Новинка

Кнут Искусство программирования Том 4 Выпуск 2

Т4В2

НОВЫЕ РАЗДЕЛЫ КЛАССИЧЕСКОГО ТРУДА

Искусство программирования

ТОМ 4

Генерация всех кортежей и перестановок

ВЫПУСК 2

Дональд Э. Кнут

ИСКУССТВО ПРОГРАММИРОВАНИЯ

ТОМ 4, ВЫПУСК 2

THE ART OF COMPUTER PROGRAMMING

VOLUME 4, FASCICLE 2

Generating All Tuples and Permutations

DONALD E. KNUTH *Stanford University*

ADDISON–WESLEY

Upper Saddle River, NJ · Boston · Indianapolis · San Francisco
New York · Toronto · Montréal · London · Munich · Paris · Madrid
Capetown · Sydney · Tokyo · Singapore · Mexico City

ИСКУССТВО ПРОГРАММИРОВАНИЯ

ТОМ 4, ВЫПУСК 2

Генерация всех кортежей и перестановок

ДОНАЛЬД Э. КНУТ *Стэнфордский университет*

Москва • Санкт-Петербург • Киев
2008

ББК 32.973.26–018.2.75
К53
УДК 681.142.2

Издательский дом "Вильямс"

Зав. редакцией *С. Н. Тригуб*
Перевод с английского и редакция канд. физ.-мат. наук *Ю.Г. Гордиенко*

По общим вопросам обращайтесь в Издательский дом "Вильямс"
по адресу: info@williamspublishing.com, http://www.williamspublishing.com
115419, Москва, а/я 783; 03150, Киев, а/я 152

Кнут, Дональд Эрвин.

К53 Искусство программирования, том 4, выпуск 2. Генерация всех кортежей и перестановок. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2008. — 160 с. : ил. — Парал. тит. англ.

ISBN 978–5–8459–1164–3 (рус.)

Этот выпуск представляет собой продолжение главы о комбинаторных алгоритмах, которая будет включена в четвертый том *Искусство программирования*. Поскольку часть этого тома составит большая глава о комбинаторном поиске, то этот выпуск начинается с рассмотрения генерации всех возможных объектов. Особое внимание уделяется генерации всех n-кортежей, которые расширяют эти идеи для всех перестановок. Такие алгоритмы дают естественную мотивацию, с помощью которой вводятся и развиваются многие ключевые идеи комбинаторной математики. Кнут в этом и других выпусках тома 4 иллюстрирует важные теории, рассматривая связанные с ними игры и головоломки. Даже самое серьезное программирование может быть увлекательным.

ББК 32.973.26–018.2.75

ISBN 978–5–8459–1164–3 (рус.)
ISBN 0–201–85393–0 (англ.)

СОДЕРЖАНИЕ

Я считаю, что стоило потратить Дни труда,
чтобы создать нечто об этом Искусстве или Науке,
Законы которых не должны быть утеряны или скрыты.

— РИЧАРД ДАКВОРТ (RICHARD DUCKWORTH),
Тинтинналогия, или Искусство колокольного звона (Tintinnalogia) (1668)

ОТ ИЗДАТЕЛЬСТВА

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш Web-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг.

Наши почтовые адреса:

E-mail:	info@williamspublishing.com
WWW:	http://www.williamspublishing.com

Наши электронные адреса:

в России:	115419, Москва, а/я 783
в Украине:	03150, Киев, а/я 152

ПРЕДИСЛОВИЕ

Я благодарен всем моим друзьям и выражаю здесь
и сейчас мою самую искреннюю признательность
тем из них, которые спустя значительное время
перестали спрашивать меня:
"Ну как там твоя книга, получается?"

— ПЕТЕР ДЖ. ГОМЕС (PETER J. GOMES),
Хорошая книга (The Good Book) (1996)

ЭТА БРОШЮРА содержит дополнение 2 книгн *Искусство программирования*, Том 4: *Комбинаторные алгоритмы*. Как уже говорилось в предисловии к дополнению 1 тома 1, я публикую этот материал в предварительной форме, поскольку знаю, что для завершения тома 4 потребуется много лет, а я не могу допустить, чтобы читатели так долго ждали, и мне хочется поскорее получить их ценные замечания.

Конечно, когда-ннбудь я напншу и дополненне 1 для тома 4 н даже дополненне 0, но пока что я написал дополнение 2. Опытные программисты знают, что инициализацию программы нельзя выполнить должным образом, еслн не создана основная часть программы.

Чтобы создать контекст, это дополнение содержит разделы 7.2.1.1 и 7.2.1.2 очень длинной главы о комбинаторном поиске. В конечном итоге глава 7 займет три тома (а именно тома 4A, 4B и 4C), если, конечно, мое здоровье позволит завершить эту работу. Она начинается с обзора теории графов с акцентом на некоторые интересные графы в базе данных The Stanford GraphBase, из которой я буду приводнть множество примеров. Затем идет раздел 7.1, в котором описываются битовые операции и алгоритмы, связанные с логическими (булевыми) функциями. Раздел 7.2 посвящен генерации всех возможных объектов и начинается с раздела 7.2.1 о генерацни основных комбннаторных объектов. Эти разделы, а нменно разделы 7.2.1.1 (где я начинаю изложение с описания генерации n-кортежей) и 7.2.1.2 (которые расшнряют идеи перестановок), образуют основное содержание данной брошюры. Затем идут раздел 7.2.1.3 (о комбинациях), раздел 7.2.1.4 (о целочисленных разбиениях) и раздел 7.2.1.5 (о разбненнях множеств) в дополнении 3, раздел 7.2.1.6 (о деревьях) н раздел 7.2.1.7 (об нстории генерацин комбинаций) в дополненни 4. Раздел 7.2.2 посвящен общим вопросам перебора с возвратами. И так далее, если все пойдет так, как задумано. Текущее содержание всей главы 7 регулярно обновляется на Web-странице `taocp`, которая указана на страннце 8.

С огромным удовольствием я излагал этот материал, подобно тому приятному возбуждению, с которым я писал том 2 много лет назад. При работе над томом 2 я, к своему восхищению, обнаружил, что основные принципы элементарной теорни вероятиости и теории чисел возиикают естественным образом при изучении

алгоритмов генерации случайных чисел и арифметики. Так и при работе над разделом 7.2.1 я обнаружил, что основные принципы элементарной комбинаторики возникают естественным и аргументированным образом при изучении алгоритмов генерации комбинаций. Таким образом, снова обнаружилась прекрасная история, которую следовало бы рассказать.

Моим исходным намерением было уделить меньше внимания этой теме. Однако, когда я обнаружил, насколько фундаментальными являются эти идеи для комбинаторики в целом, я не мог оставаться равнодушным и решил раскрыть эту тему более подробно. Потому я приложил все усилия для создания прочного фундамента для теоретических и практических идей, которые используются при создании многих надежных сверхструктур.

Тема генерации комбинаторных объектов дает мне возможность не только обсудить важные математические теории, но и развлечься, поскольку эта тема тесно связана с занимательными играми и головоломками. Хорошие головоломки очень помогают в процессе обучения, поэтому я решил продолжить их использование в томе 4.

> *Обычный мальчишка, который ненавидит квадратный корень или алгебру, приходит в восторг, решая головоломки, в которых используются те же принципы, и может настолько войти в курс дела, что у него разовьются такие математические и изобретательные шишки, которые приведут в изумление семейного френолога. (Френология — это лженаука о зависимости способностей человека от формы и размера его мозга, с особым вниманием на шишки, т.е. отдел мозга, где якобы концентрируются какие-то выдающиеся способности. — Примеч. ред.)*
>
> — СЭМ ЛОЙД (SAM LOYD),
> *Мир загадок (The World of Puzzledom)* (1896)

За каждую опечатку*, политически некорректное высказывание, а также ошибку, относящуюся к сути излагаемого материала или к приведенным историческим сведениям, я охотно заплачу $2,56 тому, кто первым ее найдет. То же самое вознаграждение будет назначено тому, кто найдет термины, которые я забыл упомянуть в предметно-именном указателе. Любые ценные предложения для улучшения текста будут оценены 32 центами за каждое. (Более того, если вы найдете более эффективное решение упражнения, я вознагражу вас бессмертной славой вместо презренных денег, публикуя ваше имя в окончательном варианте этой книги:–)

Я хочу поблагодарить Йоичи Харигучи (Yoichi Hariguchi) за помощь в сборке и модернизации компьютера, с помощью которого написана эта брошюра. Я также благодарю Фрэнка Раски (Frank Ruskey) за смелую попытку навязать черновик этого материала студентам колледжа и пересказ полученного им опыта при использовании этих материалов во время занятий.

Описания используемых в этой книге обозначений (если они не описаны явно в тексте) можно найти в перечне обозначений в конце томов 1, 2 или 3 со ссылками на места с более подробной информацией. Конечно, том 4 в будущем будет содержать собственный перечень обозначений.

---

* Имеется в виду оригинал настоящего издания. — *Примеч. ред.*

Примеры на машннном языке во всех будущих изданиях книги *Искусство программирования* будут основаны на компьютере MMIX, который определяется в разделе 1.3.1′ тома 1, дополнения 1. Перекрестные ссылки на разделы 1.3.1′, 1.3.2′, 1.4.1′, 1.4.2′ и 1.4.3′ в данной брошюре относятся к этому дополиеиию.

Перекрестные ссылки на еще не написаиный материал содержат обозначения '00', т.е. ссылки на несуществующие страницы, которые будут заменены фактическими страницами позднее в окончательном варианте книги.

Приятного чтения!

Станфорд, Калифорния D. E. K.
Декабрь 2004

Web-страница `http://www-cs-faculty.stanford.edu/~knuth/taocp.html` содержит иифорлацию о текущем состоянии англоязычной версии этой книги и связанных с ней других книг.

Web-страница `http://www-cs-faculty.stanford.edu/~knuth/sgb.html` содержит информацию о базе данных *The Stanford GraphBase*, включая доступное для копирования программное обеспечение для работы с графами, которые используются во многих примерах главы 7.

Web-страница `http://www-cs-faculty.stanford.edu/~knuth/mmix.html` содержит информацию о компьютере MMIX.

ГЛАВА 7

КОМБИНАТОРНЫЙ ПОИСК

Начальные разделы этой главы появятся в дополнениях 0 и 1 тома 4, который планируется опубликовать в 2007 году.

7.2. ГЕНЕРАЦИЯ ВСЕХ ВОЗМОЖНЫХ ОБЪЕКТОВ

Все на месте и под контролем, сэр.
— Традиционная форма доклада начальству в американской армии

Все на месте и в порядке, сэр.
— Традиционная форма доклада начальству в британской армии

7.2.1. Генерация основных комбинаторных структур

НАША ЦЕЛЬ в этом разделе состоит в изучении методов обработки всех возможных объектов в некой комбинаторной вселенной, поскольку часто прнходится сталкиваться с проблемами, при решении которых необходимо или желательно тщательно изучнть все возможные случаи. Например, часто требуется изучить все перестановки заданного множества.

Одни авторы иазывают эту задачу *перечислением* всех возможностей, но это не совсем верно, поскольку "перечисление" в большинстве случаев означает, что мы всего лишь хотим *подсчитать* общее количество случаев, а ие проанализировать их. Если кто-нибудь поставит задачу перечислить все перестановки миожества $\{1, 2, 3\}$, то вполне обоснованно можно дать краткий ответ, $3! = 6$, вместо более полного ответа — $\{123, 132, 213, 231, 312, 321\}$.

Другне авторы называют эту задачу *составлением перечня* всех возможностей, но это также не совсем верно. Вряд ли кто-нибудь захочет создавать перечень всех $10! = 3{,}628{,}800$ перестаиовок $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, печатая их иа тысячах страниц или вводя в огромный файл. Что действительио иужно, так это научиться представлять их в виде некоторой структуры данных, чтобы программа могла последовательно анализировать каждую перестановку.

Поэтому мы говорим о *генерации* всех иеобходимых комбинаторных объектов и *посещении* каждого объекта. В разделе 2.3.1 мы изучали алгорнтмы обхода дерева, где целью было посещение каждого узла дерева. Здесь мы также попробуем создать алгорнтмы, которые систематически обходят комбинаторное пространство возможных объектов.

Он всех их зачислил —
Он всех их зачислил;
И никто не упущен —
Никто не упущен.

— УИЛЬЯМ С. ГИЛЬБЕРТ,
Микадо (The Mikado) (1885)

7.2.1.1. Генерация всех n-кортежей. Начием с малого, рассматривая процедуру обхода всех 2^n строк, которые состоят из n двоичных цифр. Иначе говоря, попробуем обойти все n-кортежи $(a_1, \ldots, a_n)$, где каждый элемент a_j является либо 0, либо 1. Эта задача, по сути, эквивалентна проверке всех подмножеств заданного множества $\{x_1, \ldots, x_n\}$, поскольку можно сказать, что x_j находится в подмножестве тогда н только тогда, когда $a_j = 1$.

Конечно, такая задача имеет чрезвычайно простое решение. Все, что нужно, — начать с двоичного числа $(0\ldots00)_2 = 0$ и повторно добавлять 1 до тех пор, пока не достигнем $(1\ldots11)_2 = 2^n - 1$. Однако вскоре при более глубоком анализе мы увидим, что эта очень простая задача имеет удивительно интересные особенности. Изучение n-кортежей окупится позже при изучении особенностей генерации более сложных типов структур.

Прежде всего, нетрудно заметить, что двоичная запись может быть расширена на другие типы n-кортежей. Например, если нужно генерировать все варианты $(a_1, \ldots, a_n)$, в которых каждый элемент a_j является одной из десятичных цифр $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, то их можно подсчитать от $(0\ldots00)_{10} = 0$ до $(9\ldots99)_{10} = 10^n - 1$ в десятичной системе счисления. Если нужно в более общем смысле проанализировать все случаи, в которых

$$0 \le a_j < m_j \qquad \text{для } 1 \le j \le n, \tag{1}$$

где верхние границы m_j могут отличаться в разных компонентах вектора $(a_1, \ldots, a_n)$, то задача сводится к задаче повторного добавления единицы к числу

$$\begin{bmatrix} a_1, & a_2, & \ldots, & a_n \\ m_1, & m_2, & \ldots, & m_n \end{bmatrix} \tag{2}$$

в смешанной позиционной системе счисления (см. уравнение 4.1–(9) и упражнение 4.3.1–9).

Возьмем паузу и опишем этот процесс более строго.

Алгоритм M (*генерация в смешанной позиционной системе счисления*). Это алгоритм посещения всех n-кортежей, которые удовлетворяют (1) за счет повторного добавления 1 к числу в смешанной позиционной системе счисления в (2) до тех пор, пока не возникнет переполнение. Вспомогательные переменные a_0 и m_0 вводятся для удобства.

M1. [Инициализировать.] Пусть $a_j \leftarrow 0$ для $0 \le j \le n$, и установить $m_0 \leftarrow 2$.

M2. [Посетить.] Посетить n-кортеж $(a_1, \ldots, a_n)$. (Программа, в которой нужно проверить все n-кортежи, теперь приступает к этому.)

M3. [Подготовиться к прибавлению единицы.] Установить $j \leftarrow n$.

M4. [Перенести в случае необходимости.] Если $a_j = m_j - 1$, установить $a_j \leftarrow 0$, $j \leftarrow j - 1$, и повторить этот этап.

M5. [Увеличить, если это еще не сделано.] Если $j = 0$, завершить алгоритм. В противном случае установить $a_j \leftarrow a_j + 1$ и вернуться к этапу M2. ∎

Алгоритм M прост и прямолинеен, но не следует забывать, что вложенные циклы еще проще, если n является очень малой константой. Если $n = 4$, то можно было бы, например, записать следующие инструкции.

Для $a_1 = 0, 1, \ldots, m_1 - 1$ (в этом порядке) выполнить следующее:
  Для $a_2 = 0, 1, \ldots, m_2 - 1$ (в этом порядке) выполнить следующее:
    Для $a_3 = 0, 1, \ldots, m_3 - 1$ (в этом порядке) выполнить следующее: (3)
      Для $a_4 = 0, 1, \ldots, m_4 - 1$ (в этом порядке) выполнить следующее:
        Посетить (a_1, a_2, a_3, a_4).

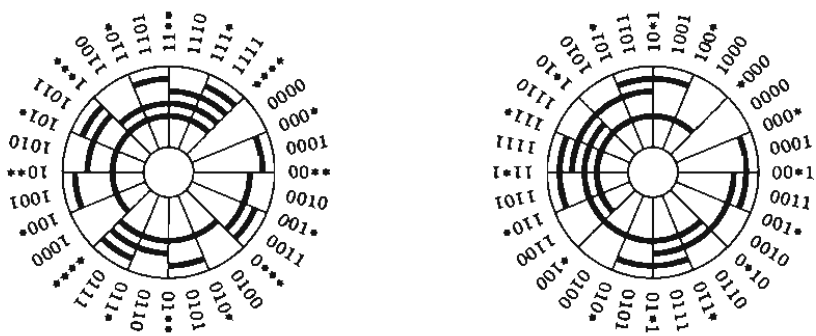


Рис. 10. (a) Лексикографический двоичный код; (b) Двоичный код Грэя

Эти инструкции эквивалентны алгоритму M и легко выражаются на любом языке программирования.

Двоичный код Грэя. Алгоритм M проходит все варианты $(a_1, \ldots, a_n)$ в лексикографическом порядке, как в словаре. Но есть много ситуаций, в которых предпочтительнее было бы посетить эти n-кортежи в некотором другом порядке. Наиболее известным альтернативным упорядочением является так называемый двоичный код Грэя, который перечисляет все 2^n строк из n бит таким образом, что только один бит изменяется каждый раз простым и регулярным способом. Например, двоичный код Грэя для $n = 4$ выглядит так:

$$0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100,$$
$$1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000. \tag{4}$$

Такой код имеет особо важное значение в приложениях, в которых аналоговая информация преобразуется в цифровую или наоборот. Предположим, что с помощью 4 сенсоров белого и черного цветов нужно идентифицировать текущее положение на вращающемся диске, который поделен на 16 секторов. Если использовать лексикографический порядок для обозначения треков от 0000 до 1111, как показано на рис. 10,*a*, то на границах секторов может возникнуть неточность измерения. А при использовании двоичного кода Грэя, как показано на рис. 10,*b*, неточностей не может быть вовсе.

Двоичный код Грэя можно определить несколькими эквивалентными способами. Например, если Γ_n обозначает двоичную последовательность Грэя из n-битовых строк, то можно определить Γ_n рекурсивно двумя правилами:

$$\begin{aligned} \Gamma_0 &= \epsilon; \\ \Gamma_{n+1} &= 0\Gamma_n,\ 1\Gamma_n^R, \end{aligned} \tag{5}$$

где ϵ обозначает пустую строку; $0\Gamma_n$ — последовательность Γ_n с префиксом 0 в каждой строке; а $1\Gamma_n^R$ — последовательность Γ_n в *обратном порядке* с префиксом 1 в каждой строке. Поскольку последняя строка в Γ_n равняется первой строке в Γ_n^R, то из (5) ясно, что в точности один бит изменяется на каждом этапе Γ_{n+1}, если Γ_n имеет то же свойство.

Еще один способ определения последовательности $\Gamma_n = g(0), g(1), \ldots, g(2^n - 1)$ заключается в задании явной формулы для отдельных элементов $g(k)$. Действи-

тельно, поскольку Γ_{n+1} начннается с $0\Gamma_n$, то неограннченная последовательность

$$\begin{aligned}\Gamma_\infty &= g(0), g(1), g(2), g(3), g(4), \ldots \\ &= (0)_2, (1)_2, (11)_2, (10)_2, (110)_2, \ldots\end{aligned} \tag{6}$$

является перестановкой всех неотрицательных целых чисел, если рассматривать каждую строку из нулей н еднннц как двоичное целое число с необязательнымн ведущими нулями. Тогда Γ_n состоит из первых 2^n элементов (6), преобразованных в n-бнтовые строки за счет вставки нулей слева, еслн необходимо.

Если $k = 2^n + r$, где $0 \le r < 2^n$, то (5) означает, что $g(k)$ равен $2^n + g(2^n - 1 - r)$. Следовательно, по нндукцни можно доказать для n, что целое число k с двоичным представленнем $(\ldots b_2 b_1 b_0)_2$ имеет эквивалент $g(k)$ в двоичном коде Грэя с представлением $(\ldots a_2 a_1 a_0)_2$, где

$$a_j = b_j \oplus b_{j+1}, \qquad \text{для } j \ge 0. \tag{7}$$

(См. упр. 6.) Например, $g\big((111001000011)_2\big) = (100101100010)_2$. Наоборот, если $g(k) = (\ldots a_2 a_1 a_0)_2$, то можно найти $k = (\ldots b_2 b_1 b_0)_2$, обращая систему равенств (7) н получая

$$b_j = a_j \oplus a_{j+1} \oplus a_{j+2} \oplus \cdots, \qquad \text{для } j \ge 0. \tag{8}$$

Эта бесконечная сумма на самом деле конечна, поскольку $a_{j+t} = 0$ для всех больших t.

Одно из многочисленных приятных следствий уравнения (7) заключается в том, что $g(k)$ можно вычислить очень легко с помощью битовой арифметики:

$$g(k) = k \oplus \lfloor k/2 \rfloor. \tag{9}$$

Аналогично, обратная функция в (8) удовлетворяет

$$g^{[-1]}(l) = l \oplus \lfloor l/2 \rfloor \oplus \lfloor l/4 \rfloor \oplus \cdots. \tag{10}$$

Однако для этой функции требуется больше вычислений (см. упр. 7.1.3–117). Из (7) можно также вывести, что если k и k' являются любыми неотрицательными целыми числами, то

$$g(k \oplus k') = g(k) \oplus g(k'). \tag{11}$$

Еще одно следствие состоит в том, что $(n+1)$-битовый двоичный код Грэя может быть записан в виде

$$\Gamma_{n+1} = 0\Gamma_n,\ (0\Gamma_n)\oplus 110\ldots0.$$

Эта структура очевидна, например, в (4). Сравнивая ее с (5), можно заметить, что обращение порядка двоичного кода Грэя эквивалентно дополнению первого бита:

$$\Gamma_n^R = \Gamma_n \oplus 1\overbrace{0\ldots0}^{n-1}. \tag{12}$$

В приведенном ниже упражнении показано, что функция $g(k)$, определенная в (7), и обратная ей функция $g^{[-1]}$, определенная в (8), имеют много других интересных свойств и приложений. Порой эти функции рассматриваются как принимающие двоичные строки и возвращающие двоичные строки. А иногда эти функции рассматриваются как принимающие целые числа и возвращающие целые числа посредством двоичной системы обозначений без учета ведущих нулей.

Двоичный код Грэя назван так в честь физика Фрэнка Грэя (Frank Gray), прославившегося изобретением, которое долгое время использовалось для обеспечения нормального приема цветных программ черно-белым телевизионным приемником [*Bell System Tech. J.* **13** (1934), p.464–515]. Он изобрел код Γ_n для приложений с кодово-импульсной модуляцией , т.е. метода аналоговой передачи цифровых сигналов [*Bell System Tech. J.* **30** (1951), p.38–40; *U.S. Patent 2632058* (17 March 1953); W. R. Bennett, *Introduction to Signal Transmission* (1971), p.238–240]. Но идея двоичного кода Грэя была известна задолго до его работы над этой проблемой, например, из патента *U.S. Patent 2307868* (12 January 1943) Джорджа Стибица (George Stibitz). Важнее другое: код Γ_5 использовался в телеграфной машине, продемонстрированной в 1878 году Эмилем Бодо (Émile Baudot), в честь которого была названа единица измерения "бод". Почти в то же время аналогичный, но более систематический код для телеграфа был независимо предложен Отто Шафлером (Otto Schäffler) [*Journal Télégraphique* **4** (1878), p.252–253; *Annales Télégraphiques* **6** (1879), 361, p.382–383].*

Интересно, что двоичный код Грэя неявно присутствует в классической игрушке, которая веками забавляла людей и известна как "китайское переплетение колец", или "Chinese ring puzzle" на английском языке, хотя англичане часто называют эту головоломку "утомительными железками" ("tiring irons"). На рис. 11 показан пример этой головоломки с 7 кольцами. Задача заключается в том, чтобы снять кольца со стержня, а кольца переплетены таким образом, что возможны только два основных действия (хотя это может быть не совсем очевидно на данном рисунке):

a) Самое правое кольцо можно снять или сдвинуть в любой момент времени;
b) Любое другое кольцо можно снять или сдвинуть тогда и только тогда, когда кольцо справа от него находится на стержне и сняты все кольца справа от него.

Текущее состояние головоломки можно представить в двоичной системе счисления, где 1 означает, что кольцо находится на стержне, а 0 — что кольцо снято. Таким образом, на рис. 11 показано состояние с кодом 1011000. (Второе кольцо слева обозначено нулем, поскольку оно полностью находится над стержнем.)

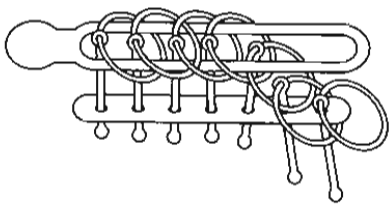


Рис. 11.
Головоломка "китайское переплетение колец"

Французский судья-магистрат Луи Грос (Louis Gros) продемонстрировал явную связь между этой головоломкой и двоичными числами в анонимно опубликованной книжке *Теория безделья (Théorie du Baguenodier)* [шик!] (Lyon: Aimé Vingtrinier, 1872). Если кольца находятся в состоянии $a_{n-1}\ldots a_0$ и если определить двоичное число $k = (b_{n-1}\ldots b_0)_2$ согласно (8), то точно на k шагов больше будет необходимо и достаточно для решения головоломки. Таким образом, Грос является истинным изобретателем двоичного кода Грэя.

* Некоторые авторы утверждают, что код Грэя был изобретен Элиша Грэем (Elisha Gray), который создал печатающую телеграфную машину в то же время, что и Бодо и Шафлер. Такие заявления не соответствуют действительности, хотя с Элиша обошлись несправедливо в отношении приоритета в изобретении телефона [L. W. Taylor, *Amer. Physics Teacher* **5** (1937), p.243–251].

Несомненно, что эта очаровательная, историческая и поучительная головоломка должна быть в каждом доме.

— ГЕНРИ Э. ДЬЮДЕНИ (HENRY E. DUDENEY) (1901)

Если кольца находятся в произвольном состоянии, отличном от состояний $00\ldots0$ или $10\ldots0$, то возможны точно два действия: одно типа (a) и одно типа (b). Только одно из этих действий дает продвижение к заданной цели, а другое является шагом назад, который нужно отменить. Действие типа (a) меняет k на $k\oplus1$. Таким образом, его нужно выполнять, если k является нечетным числом, поскольку в таком случае удается уменьшить k. Действие типа (b) в состоянии, которое оканчивается в виде $(10^{j-1})_2$ для $1\le j<n$, изменяет k на $k\oplus(1^{j+1})_2=k\oplus(2^{j+1}-1)$. Если k четно, то нужно $k\oplus(2^{j+1}-1)$ приравнять к $k-1$, что означает, что k должно быть кратно 2^j, но не кратно 2^{j+1}, или, иначе говоря,

$$j=\rho(k), \tag{13}$$

где ρ называется "масштабной функцией" уравнения 7.1.3–(44). Следовательно, для оптимального решения головоломки действия с кольцами должны подчиняться следующей закономерности. Если пронумеровать кольца 0, 1, $\ldots$, $n-1$ (начиная со свободного конца), то последовательность снятий и одеваний колец со стержня будет иметь вид последовательности чисел, которая заканчивается $\ldots$, $\rho(4)$, $\rho(3)$, $\rho(2)$, $\rho(1)$.

Идя в обратном направлении, нужно последовательно снимать или одевать кольца до тех пор, пока не будет достигнуто конечное состояние $10\ldots0$ (которое, как заметил Джон Уоллис (John Wallis) в 1693 г., гораздо сложнее получить, чем, на первый взгляд, более сложное состояние $11\ldots1$). Таким образом, получаем алгоритм решения в двоичном коде Грэя:

Алгоритм G (*генерация двоичного кода Грэя*). Этот алгоритм посещает все двоичные n-кортежи $(a_{n-1},\ldots,a_1,a_0)$, начинаясь с $(0,\ldots,0,0)$ и изменяя последовательно только по одному биту и поддерживая бит четности a_∞ в таком состоянии, что

$$a_\infty=a_{n-1}\oplus\cdots\oplus a_1\oplus a_0. \tag{14}$$

Он последовательно дополняет биты $\rho(1)$, $\rho(2)$, $\rho(3)$, $\ldots$, $\rho(2^n-1)$ и затем останавливается.

G1. [Инициализировать.] Установить $a_j\leftarrow0$ для $0\le j<n$; также установить $a_\infty\leftarrow0$.

G2. [Посетить.] Посетить n-кортеж $(a_{n-1},\ldots,a_1,a_0)$.

G3. [Изменить четность.] Установить $a_\infty\leftarrow1-a_\infty$.

G4. [Выбрать j.] Если $a_\infty=1$, то установить $j\leftarrow0$. В противном случае пусть $j\ge1$ будет таким минимальным, что $a_{j-1}=1$. (После k-го выполнения этого этапа $j=\rho(k)$.)

G5. [Дополнить координату j.] Завершить, если $j=n$. В противном случае установить $a_j\leftarrow1-a_j$ и вернуться к Г2. ▮

Бит четности a_∞ пригодится при вычислении суммы

$$X_{000}-X_{001}-X_{010}+X_{011}-X_{100}+X_{101}+X_{110}-X_{111}$$

или

$$X_\emptyset - X_a - X_b + X_{ab} - X_c + X_{ac} + X_{bc} - X_{abc},$$

где знак зависит от четности двоичной строки или количества элементов подмножества. Такие суммы часто возникают в формулах "включения-исключения", как в уравнениях 1.3.3–(29). Бит четности также необходим для эффективности: без него было бы не так просто выбрать один из двух способов определения значения j, которое соответствует выполнению действия типа (a) или типа (b) в "китайском переплетении колец". Но наиболее важной особенностью алгоритма G является то, что на этапе G5 происходит только одно изменение координаты. Следовательно, обычно требуется только простое изменение суммируемых членов X или других структур, с которыми приходится иметь дело при посещении каждого n-кортежа.

Конечно, невозможно избежать двусмысленности, связанной с цифрой самого младшего разряда, за исключением способа, с помощью которого, по слухам, одна из ирландских железных дорог попробовала решить одну проблему: они предложили удалить последний вагон во всех поездах, поскольку именно они особенно уязвимы при столкновениях поездов.

— Г. Р. СТИБИЦ (G. R. STIBITZ)
и Дж. Э. ЛАРРИВИ (J. A. LARRIVEE),
Математика и компьютеры (Mathematics и Computers) (1957)

Еще одно ключевое свойство двоичного кода Грэя было открыто Дж. Л. Уолшем (J. L. Walsh) в связи с важной последовательностью функций, которые теперь известны как *функции Уолша* [*Amer. J. Math.* **45** (1923), p. 5–24]. Пусть $w_0(x) = 1$ для всех действительных чисел x, а

$$w_k(x) = (-1)^{\lfloor 2x \rfloor \lceil k/2 \rceil} w_{\lfloor k/2 \rfloor}(2x) \qquad \text{для } k > 0. \tag{15}$$

Например, $w_1(x) = (-1)^{\lfloor 2x \rfloor}$ изменяет знак всякий раз, когда x является целым числом или целым числом плюс $\frac{1}{2}$. Отсюда следует, что $w_k(x) = w_k(x+1)$ для всех k и $w_k(x) = \pm 1$ для всех x. Гораздо важнее то, что $w_k(0) = 1$ и $w_k(x)$ *точно* k *раз меняет знак в диапазоне* $(0\,..\,1)$ и приближается к $(-1)^k$ по мере того, как x приближается к 1 слева. Следовательно, $w_k(x)$ ведет себя скорее как тригонометрическая функция $\cos k\pi x$ или $\sin k\pi x$. Поэтому другие функции можно представить в виде линейной комбинации функций Уолша таким же образом, каким они обычно представляются с помощью рядов Фурье. Этот факт, вместе с простой дискретной природой $w_k(x)$, означает, что функции Уолша чрезвычайно полезны в компьютерных вычислениях, связанных с передачей информации, обработкой изображений и многими другими приложениями.

На рис. 12 показаны первые восемь функций Уолша вместе с их тригонометрическими "собратьями". Инженеры обычно называют $w_k(x)$ функцией Уолша с *порядком* k, по аналогии с тригонометрическими функциями $\cos k\pi x$ и $\sin k\pi x$ с *частотой* $k/2$ [см., например, *H. F. Harmuth, Sequency Theory: Foundations и Applications* (New York: Academic Press, 1977)].

Хотя уравнение (15) на первый взгляд выглядит пугающе, на самом деле оно предоставляет с помощью индукции простой способ объяснения, почему $w_k(x)$ имеет точно k изменений знака. Если k четно, например $k = 2l$, то $w_{2l}(x) = w_l(2x)$ для $0 \le x < \frac{1}{2}$, т.е. функция $w_l(x)$ сжимается вдвое и $w_{2l}(x)$ содержит l изменений

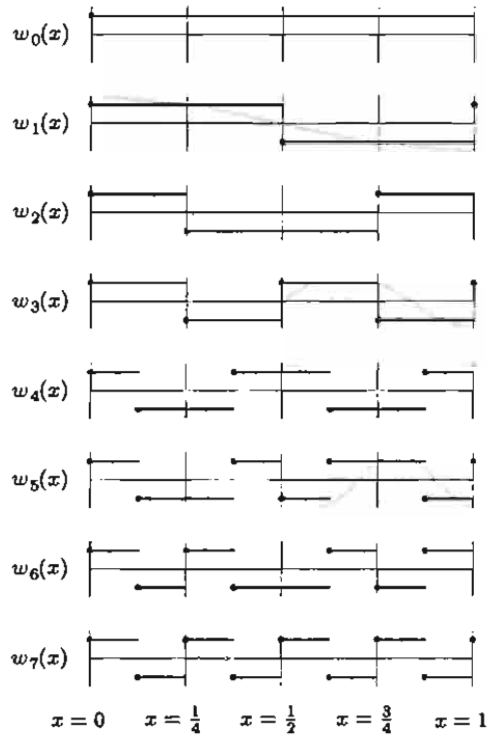


Рис. 12. Функции Уолша $w_k(x)$ для $0 \le k < 8$ и аналогичные тригонометрические функции $\sqrt{2}\cos k\pi x$, показанные для сравнения серым цветом

знака. Тогда $w_{2l}(x) = (-1)^l w_l(2x) = (-1)^l w_l(2x-1)$ в диапазоне $\frac{1}{2} \le x < 1$, что позволяет присоединить другую копию $w_l(2x)$, обращая знак в случае необходимости во избежание изменения знака при $x = \frac{1}{2}$. Функция $w_{2l+1}(x)$ аналогична, но она *вынуждает* изменить знак при $x = \frac{1}{2}$.

Какое отношение к этому имеет двоичный код Грэя? Уолш обнаружил, что все его функции можно компактно выразнть на основе более простых *функций Радемахера* [см. Hans Rademacher, *Math. Annalen* **87** (1922), p.112–138],

$$r_k(x) \;=\; (-1)^{\lfloor 2^k x \rfloor}, \tag{16}$$

которые принимают значение $(-1)^{c-k}$, где $(\ldots c_2 c_1 c_0 . c_{-1} c_{-2} \ldots)_2$ является двоичным представлением x. Действнтельно, $w_1(x) = r_1(x)$, $w_2(x) = r_1(x)r_2(x)$, $w_3(x) = r_2(x)$, и в общем случае

$$w_k(x) = \prod_{j\ge 0} r_{j+1}(x)^{b_j \oplus b_{j+1}}, \qquad \text{где } k = (b_{n-1}\ldots b_1 b_0)_2. \tag{17}$$

(См. упр. 33.) Таким образом, порядок $r_{j+1}(x)$ в $w_k(x)$ представляет собой j-й бит двоичного числа Грэя $g(k)$, согласно (7), и имеем:

$$w_k(x) \;=\; r_{\rho(k)+1}(x)\, w_{k-1}(x) \qquad \text{для } k > 0. \tag{18}$$

Уравнение (17) подразумевает удобную формулу

$$w_k(x)\, w_{k'}(x) \;=\; w_{k\oplus k'}(x), \tag{19}$$

которая намного проще, чем соответствующие формулы для произведения синуса и косинуса. Это тождество легко проверить, поскольку $r_j(x)^2 = 1$ для всех j и x, следовательно, $r_j(x)^{a\oplus b} = r_j(x)^{a+b}$. В частности, подразумевается, что $w_k(x)$ *ортогональна* $w_{k'}(x)$, если $k \neq k'$, в том смысле, что среднее значение $w_k(x)w_{k'}(x)$ равно нулю. Уравнение (17) можно использовать для определения $w_k(x)$ для дробных значений k, например $1/2$ или $13/8$.

Преобразованием Уолша 2^n чисел $(X_0, \ldots, X_{2^n-1})$ называется вектор, определенный уравнением $(x_0, \ldots, x_{2^n-1})^T = W_n(X_0, \ldots, X_{2^n-1})^T$, где W_n является матрицей $2^n \times 2^n$ с элементами $w_j(k/2^n)$ на пересечении строки j и столбца k для $0 \le j, k < 2^n$. Например, на рис. 12 показано, что преобразование Уолша при $n = 3$ имеет вид

$$\begin{pmatrix} x_{000} \\ x_{001} \\ x_{010} \\ x_{011} \\ x_{100} \\ x_{101} \\ x_{110} \\ x_{111} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & \bar{1} & \bar{1} & \bar{1} & \bar{1} \\ 1 & 1 & \bar{1} & \bar{1} & \bar{1} & \bar{1} & 1 & 1 \\ 1 & 1 & \bar{1} & \bar{1} & 1 & 1 & \bar{1} & \bar{1} \\ 1 & \bar{1} & \bar{1} & 1 & 1 & \bar{1} & \bar{1} & 1 \\ 1 & \bar{1} & \bar{1} & 1 & \bar{1} & 1 & 1 & \bar{1} \\ 1 & \bar{1} & 1 & \bar{1} & \bar{1} & 1 & \bar{1} & 1 \\ 1 & \bar{1} & 1 & \bar{1} & 1 & \bar{1} & 1 & \bar{1} \end{pmatrix} \begin{pmatrix} X_{000} \\ X_{001} \\ X_{010} \\ X_{011} \\ X_{100} \\ X_{101} \\ X_{110} \\ X_{111} \end{pmatrix} \tag{20}$$

(где $\bar{1}$ обозначает -1, а подстрочный индекс обычно рассматривается как двоичная строка 000–111 для обозначения целых чисел 0–7). Аналогично определяется *преобразование Адамара*, но вместо матрицы W_n в нем используется матрица H_n, где H_n содержит элементы $(-1)^{j\cdot k}$ на пересечении строки j и столбца k. Здесь "$j \cdot k$" обозначает скалярное произведение $a_{n-1}b_{n-1} + \cdots + a_0b_0$ двоичных представлений $j = (a_{n-1} \ldots a_0)_2$ и $k = (b_{n-1} \ldots b_0)_2$. Например, преобразование Адамара для $n = 3$ имеет вид

$$\begin{pmatrix} x'_{000} \\ x'_{001} \\ x'_{010} \\ x'_{011} \\ x'_{100} \\ x'_{101} \\ x'_{110} \\ x'_{111} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \bar{1} & 1 & \bar{1} & 1 & \bar{1} & 1 & \bar{1} \\ 1 & 1 & \bar{1} & \bar{1} & 1 & 1 & \bar{1} & \bar{1} \\ 1 & \bar{1} & \bar{1} & 1 & 1 & \bar{1} & \bar{1} & 1 \\ 1 & 1 & 1 & 1 & \bar{1} & \bar{1} & \bar{1} & \bar{1} \\ 1 & \bar{1} & 1 & \bar{1} & \bar{1} & 1 & \bar{1} & 1 \\ 1 & 1 & \bar{1} & \bar{1} & \bar{1} & \bar{1} & 1 & 1 \\ 1 & \bar{1} & \bar{1} & 1 & \bar{1} & 1 & 1 & \bar{1} \end{pmatrix} \begin{pmatrix} X_{000} \\ X_{001} \\ X_{010} \\ X_{011} \\ X_{100} \\ X_{101} \\ X_{110} \\ X_{111} \end{pmatrix}. \tag{21}$$

Это то же самое, что и дискретное преобразование Фурье на n-мерном кубе, см. уравнения 4.6.4–(38), и его можно сразу же "на месте" оценить, адаптируя метод Йетса из раздела 4.6.4:

Дано	Первый этап	Второй этап	Третий этап
X_{000}	$X_{000}+X_{001}$	$X_{000}+X_{001}+X_{010}+X_{011}$	$X_{000}+X_{001}+X_{010}+X_{011}+X_{100}+X_{101}+X_{110}+X_{111}$
X_{001}	$X_{000}-X_{001}$	$X_{000}-X_{001}+X_{010}-X_{011}$	$X_{000}-X_{001}+X_{010}-X_{011}+X_{100}-X_{101}+X_{110}-X_{111}$
X_{010}	$X_{010}+X_{011}$	$X_{000}+X_{001}-X_{010}-X_{011}$	$X_{000}+X_{001}-X_{010}-X_{011}+X_{100}+X_{101}-X_{110}-X_{111}$
X_{011}	$X_{010}-X_{011}$	$X_{000}-X_{001}-X_{010}+X_{011}$	$X_{000}-X_{001}-X_{010}+X_{011}+X_{100}-X_{101}-X_{110}+X_{111}$
X_{100}	$X_{100}+X_{101}$	$X_{100}+X_{101}+X_{110}+X_{111}$	$X_{000}+X_{001}+X_{010}+X_{011}-X_{100}-X_{101}-X_{110}-X_{111}$
X_{101}	$X_{100}-X_{101}$	$X_{100}-X_{101}+X_{110}-X_{111}$	$X_{000}-X_{001}+X_{010}-X_{011}-X_{100}+X_{101}-X_{110}+X_{111}$
X_{110}	$X_{110}+X_{111}$	$X_{100}+X_{101}-X_{110}-X_{111}$	$X_{000}+X_{001}-X_{010}-X_{011}-X_{100}-X_{101}+X_{110}+X_{111}$
X_{111}	$X_{110}-X_{111}$	$X_{100}-X_{101}-X_{110}+X_{111}$	$X_{000}-X_{001}-X_{010}+X_{011}-X_{100}+X_{101}+X_{110}-X_{111}$

Обратите внимание на то, что строки H_3 являются перестановками строк W_3. Это верно и в общем случае, а поэтому преобразование Уолша можно выполнить за счет перестановки элементов преобразования Адамара. Этот прием подробно рассматривается в упражнении 36.

Еще быстрее. При анализе 2^n возможностей обычно требуется, насколько это возможно, сократить время вычисления. Алгоритм G дополняет только один бит a_j за одно посещение $(a_{n-1},\ldots,a_0)$, но зацикливается на этапе G4 при выборе соответствующего значения j. Другой подход был предложен Гидеоном Ерлихом (Gideon Ehrlich) [*JACM* **20** (1973), p.500–513], который ввел понятие *бесцикловой* комбинаторной генерации. В бесцикловом алгоритме количество операций, выполняемых между последовательными посещениями, заранее ограничивается, поэтому никогда не приходится долго ожидать генерации новой структуры.

В разделе 7.1.3 мы узнали несколько хитроумных и быстрых способов определения количества ведущих или замыкающих 0s в двоичном числе. Эти методы могут быть использованы на этапе Г4, чтобы сделать алгоритм G бесцикловым, если n не слишком велико. Но метод Эрлиха совсем другой и более универсальный, поэтому он расширяет арсенал доступных методов эффективного вычисления. Вот как этот подход можно использовать для генерации двоичных n-кортежей [Bitner, Ehrlich, and Reingold, *CACM* **19** (1976), p.517–521].

Алгоритм L (*бесцикловая генерация двоичного кода Грэя*). В этом алгоритме, как и в алгоритме G, посещаются все двоичные n-кортежи $(a_{n-1},\ldots,a_0)$ в порядке двоичного кода Грэя. Но вместо бита четности в нем используется массив "указателей фокуса" $(f_n,\ldots,f_0)$, значение которых обсуждается ниже.

L1. [Инициализировать.] Установить $a_j \leftarrow 0$ и $f_j \leftarrow j$ для $0 \le j < n$; а также установить $f_n \leftarrow n$. (Бесцикловый алгоритм может иметь циклы на этапе инициализации, если это достаточно эффективно. В конце концов, любая программа должна быть загружена и запущена.)

L2. [Посетить.] Посетить n-кортеж $(a_{n-1},\ldots,a_1,a_0)$.

L3. [Выбрать j.] Установить $j \leftarrow f_0$, $f_0 \leftarrow 0$. (Если данный этап выполняется в k-й раз, то j теперь равно $\rho(k)$.) Завершить, если $j = n$, в противном случае установить $f_j \leftarrow f_{j+1}$ и $f_{j+1} \leftarrow j+1$.

L4. [Дополнить координату j.] Установить $a_j \leftarrow 1 - a_j$ и вернуться к этапу L2. ▮

Например, для $n = 4$ вычисления проходят следующим образом. Элемент a_j подчеркнут в этой таблице, если соответствующий бит b_j равен 1 в двоичной строке $b_3b_2b_1b_0$ так, что $a_3a_2a_1a_0 = g(b_3b_2b_1b_0)$:

a_3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
a_2	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
a_1	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
a_0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
f_3	3	3	3	3	3	3	3	3	4	4	4	4	3	3	3	3
f_2	2	2	2	2	3	3	2	2	2	2	2	2	4	4	2	2
f_1	1	1	2	1	1	1	3	1	1	1	2	1	1	1	4	1
f_0	0	1	0	2	0	1	0	3	0	1	0	2	0	1	0	4

Хотя двоичное число $k = (b_{n-1} \dots b_0)_2$ никогда не появляется явно в алгоритме L, указатели фокуса f_j представляют его неявно хитроумным образом так, что можно повторно получать $g(k) = (a_{n-1} \dots a_0)_2$, дополняя бит $a_{\rho(k)}$. Пусть *пассивность* a_j обозначается подчеркиванием, а *активность* — без подчеркивания. Тогда указатели фокуса удовлетворяют следующим инвариантным отношениям:

1) Если бит a_j пассивен, а бит a_{j-1} активен, то f_j является таким наименьшим индексом $j' > j$, что бит $a_{j'}$ активен. (Биты a_n и a_{-1} считаются активными в этом правиле, хотя они реально не представлены в алгоритме.)
2) В противном случае $f_j = j$.

Таким образом, самый правый элемент a_j блока пассивных элементов $a_{i-1} \dots a_{j+1} a_j$ с уменьшающимися индексами имеет указатель фокуса f_j, который указывает на элемент a_i с левой стороны этого блока. Все другие элементы a_j имеют указатели фокуса f_j, указывающие на себя.

На основании этого первые две операции '$j \leftarrow f_0$, $f_0 \leftarrow 0$' на этапе L3 эквивалентны высказыванию: "Установить j равным индексу самого правого активного элемента и активизировать все элементы справа от a_j". Обратите внимание, что если $f_0 = 0$, то операция $f_0 \leftarrow 0$ избыточна, но это не причиняет никакого вреда. Другие две операции на этапе L3, '$f_j \leftarrow f_{j+1}$, $f_{j+1} \leftarrow j + 1$', эквивалентны высказыванию: "Сделаем a_j пассивным", поскольку известно, что a_j и a_{j-1} активны на этом этапе вычисления. (Опять-таки операция $f_{j+1} \leftarrow j + 1$ может быть избыточной, но совершение безвредной.) Следовательно, общий результат активизации и деактивизации эквивалентен подсчетам в двоичном системе обозначений, как в алгоритме M, с пассивными битами 1 и активными битами 0.

Алгоритм L поразительно быстр, поскольку он содержит только пять операций присваивания и одну проверку завершения между каждыми посещениями генерированного n-кортежа. Но его все равно можно улучшить. Чтобы понять, как его можно улучшить, рассмотрим его приложение в занимательной лингвистике: Рудольф Кастаун (Rudolph Castown) заметил [*Word Ways* **1** (1968), p.165–169], что все 16 способов смешения букв в слове `sins` с соответствующими буквами слова `fate` приводят к получению слов, которые можно найти в обширном словаре английского языка: `sine`, `sits`, `site` и т.д. Все эти слова, кроме слов `fane`, `fite` и `sats`, широко распространены и бесспорно являются частью английского языка. Следовательно, естественно было бы задать аналогичный вопрос для слов из пяти букв: какие две строки из пяти букв порождают максимальное количество слов в базе данных Stanford GraphBase, где буквы в соответствующих положениях обмениваются всеми 32 возможными способами?

Для ответа на этот вопрос нужно проверить все $\binom{26}{2}^5 = 3{,}625{,}908{,}203{,}125$ разные пары строк. Достаточно просмотреть все $\binom{5757}{2} = 16{,}568{,}646$ пары слов в базе данных GraphBase, при условии, что по крайней мере одна из этих пар порождает не меньше 17 слов, поскольку каждый набор из 17 или более 5-буквенных слов, полученных из двух 5-буквенных строк, должен содержать две строки-антипода (без одинаковых соответствующих букв). Для каждой пары строк-антиподов нужно максимально быстро определить, дают ли 32 возможные перестановки слова английского языка.

Каждое 5-буквенное слово можно представить как 25-битовое число с помощью 5-битового представления одной буквы: от `"a"` = 00000 до `"z"` = 11001. Тогда

с помощью таблицы из 2^{25} битов или байтов можно быстро определить, является ли 5-буквенная строка словом. Таким образом, задача сводится к генерации битовых комбинаций 32 возможных слов, получаемых с помощью смешения букв двух данных слов, и поиску этих комбинаций в таблице. Для каждой пары 25-битовых слов w и w' следует выполнить следующие действия.

W1. [Проверить разницу.] Установить $z \leftarrow w \oplus w'$. Отвергнуть пару (w, w'), если $\big((z - m) \oplus z \oplus m\big)\&m' \neq 0$, где $m = 2^{20} + 2^{15} + 2^{10} + 2^5 + 1$ и $m' = 2^5 m$. Благодаря этой проверке исключаются случаи, когда w и w' имеют общую букву в некотором положении. (См. 7.1.3–(90). Оказывается, что 10 614 085 из 16 568 646 пар слов не имеют таких общих букв.)

W2. [Создать индивидуальную маску.] Устаиовить $m_0 \leftarrow z\&(2^5 - 1)$, $m_1 \leftarrow z\&(2^{10} - 2^5)$, $m_2 \leftarrow z\&(2^{15} - 2^{10})$, $m_3 \leftarrow z\&(2^{20} - 2^{15})$ и $m_4 \leftarrow z\&(2^{25} - 2^{20})$ для подготовки к следующему этапу.

W3. [Подсчитать слова.] Установить $l \leftarrow 1$ и $A_0 \leftarrow w$. Переменная l будет содержать количество найденных до сих пор слов, начиная с w. Затем выполнить описанные ниже операции $swap(4)$.

W4. [Вывести максимальное решение.] Если l больше текущего максимума или равно ему, то вывести A_j для $0 \le j < l$. ▮

Сердцем этого высокоскоростного метода является последовательность операций $swap(4)$, которая должна быть расширена (например, с помощью макропроцессора) для исключения всех ненужных накладных расходов. Она определяется на основе базовой операции

$sw(j)$: установить $w \leftarrow w \oplus m_j$.

Затем, если w является словом, установить $A_l \leftarrow w$ и $l \leftarrow l + 1$.

Для заданной операции $sw(j)$, которая переводит буквы в положение j, определяем:

$$\begin{aligned} swap(0) &= sw(0); \\ swap(1) &= swap(0), sw(1), swap(0); \\ swap(2) &= swap(1), sw(2), swap(1); \\ swap(3) &= swap(2), sw(3), swap(2); \\ swap(4) &= swap(3), sw(4), swap(3). \end{aligned} \tag{22}$$

Таким образом, $swap(4)$ расширяется в последовательиость 31 шага $sw(0)$, $sw(1)$, $sw(0)$, $sw(2)$, ..., $sw(0) = sw(\rho(1))$, $sw(\rho(2))$, ..., $sw(\rho(31))$, которые будут использоваться десять миллионов раз. Очевидно, что выигрыш в скорости получается за счет вставки значений масштабной функции $\rho(k)$ непосредственно в нашу программу, а не повторных вычислений для каждой пары слов с помощью алгоритмов M, G или L.

Победившая пара слов генерирует набор из 21 слова:

```
ducks, ducky, ducos, dunes, dunks, dinks, dinky,
dines, dices, dicey, dicky, dicks, picks, picky,
pines, piney, pinky, pinks, punks, punky, pucks.
```
(23)

Если, например, w = ducks и w' = piney, то m_0 = s ⊕ y, и первая операция $sw(0)$ заменяет строку ducks строкой ducky, которая является словом. Следующая операция $sw(1)$ применяет m_1, т.е. приводит к замене k ⊕ e буквы на предпоследнем месте, что порождает строку ducey, которая ие является словом. Применение операции $sw(0)$ заменяет строку ducey строкой duces (слово, являющееся частью юридического термиина duces tecum)* и т.д. Все пары слов можно обработать этим методом всего за несколько секунд.

Этот метод можно оптимизировать еще больше. Например, после нахождения пары, которая дает k слов, следует отвергнуть все другие пары, если они генерируют $33 - k$ строк, которые не являются словами. Но рассмотренный выше метод и так достаточно быстрый и может превзойти даже бесцикловый алгоритм L.

Приверженцы алгоритма L могут, конечно, возразить, что процесс ускорен только в частном случае, $n = 5$, а алгоритм L решает задачу генерации для произвольного n. Однако аналогичная идея применима и для $n > 5$: программу можно расширить так, чтобы она быстро генерировала все 32 положения самых правых битов $a_4a_3a_2a_1a_0$, как показано выше. Затем можно применить алгоритм L после каждых 32 шагов для последовательных изменений других битов, $a_{n-1} \ldots a_5$. Этот подход сокращает объем ненужной работы, выполняемой алгоритмом L, почти в 32 раза.

Другие двоичные коды Грэя. Двоичный код Грэя $g(0)$, $g(1)$, ..., $g(2^n - 1)$ является только одним из многих способов обхода всех возможных n-битовых строк, изменяя только один бит на каждом этапе. Назовем в общем случае циклом Грэя для двоичных n-кортежей *любую* последовательность $(v_0, v_1, \ldots, v_{2^n-1})$, которая включает каждый n-кортеж и обладает таким свойством, что v_k отличается от $v_{(k+1) \bmod 2^n}$ только одним битом. Таким образом, на языке теории графов цикл Грэя является ориентироваиным циклом Гамильтона на n-кубе. Здесь предполагается, что индексы выбраны таким образом, что $v_0 = 0 \ldots 0$.

Если v нредставить в виде двоичных чисел, то существуют такие целые числа $\delta_0 \ldots \delta_{2^n-1}$, что

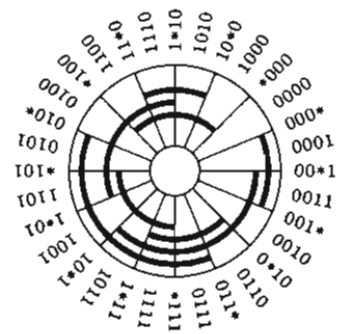
$$v_{(k+1) \bmod 2^n} = v_k \oplus 2^{\delta_k} \qquad \text{для } 0 \le k < 2^n. \tag{24}$$

Эта так называемая дельта-последовательность представляет собой еще один способ описания цикла Грэя. Например, дельта-последовательность для стандартного двоичного кода Грэя для $n = 3$ имеет внд 01020102. Это, по сути, масштабная функция $\delta_k = \rho(k + 1)$ из (13), но последнее значение δ_{2^n-1} равно $n - 1$ вместо n, поэтому цикл завершается. Поскольку отдельные элементы δ_k всегда находятся в диапазоне $0 \le \delta_k < n$, оии иазываются координатами.

Пусть $d(n)$ — это количество разных дельта-последовательностей, которые определяют n-битовый цикл Грэя, а $c(n)$ — это количество канонических дельта-последовательностей, в которых каждая координата k появляется перед первым появлением $k + 1$. Тогда $d(n) = n!\,c(n)$, поскольку каждая перестановка координат в дельта-последовательности очевидно порождает другую дельта-последовательность. Легко видеть, что единственными возможными каноническими дельта-

* Означает повестку о явке в суд для предоставления имеющихся письменных доказательств. — *Примеч. ред.*

(a)



(b)

Рис. 13. (a) Дополняющий двоичный код Грэя; (b) Сбалансированный двоичный код Грэя

последовательностями для $n \le 3$ являются

$$00; \qquad 0101; \qquad 01020102 \text{ и } 01210121. \tag{25}$$

Следовательно, $c(1) = c(2) = 1$, $c(3) = 2$; $d(1) = 1$, $d(2) = 2$, $d(3) = 12$. Прямые вычисления с помощью методов перечисления циклов Гамильтона, которые будут рассмотрены ниже, дают следующие значения:

$$\begin{aligned} c(4) &= 112; & d(4) &= 2688; \\ c(5) &= 15{,}109{,}096; & d(5) &= 1{,}813{,}091{,}520. \end{aligned} \tag{26}$$

Здесь трудно уловить закономерность, а числа растут очень быстро (см. упр. 45). Поэтому можно с уверенностью сказать, что никто никогда не скажет, какими будут значения $c(8)$ и $d(8)$.

Поскольку количество возможностей столь велико, то имеет смысл поискать дополнительные полезные свойства циклов Грэя. Например, на рис. 13,*a* показан 4-битовый цикл Грэя, в котором каждая строка $a_3a_2a_1a_0$ диаметрально противоположна своему дополненню $\bar{a}_3\bar{a}_2\bar{a}_1\bar{a}_0$. Такая схема кодирования возможна, когда количество битов является четным (см. упр. 49).

Еще больший интерес представляет цикл Грэя, найденный Дж. С. Тутнллом (G. C. Tootill) [*Proc. IEE* **103**, Part B Supplement (1956), p.435] и показанный на рис. 13,*b*. Он имеет одинаковое колнчество изменений в каждом из четырех координатных треков, поэтому все координаты в равной мере участвуют во всех операциях. Сбалансированные таким образом циклы Грэя можно создать для всех более крупных значений n с помощью следующего унивсрсального метода расшнрения цикла от n бнт до $n + 2$ бит.

Теорема D. *Пусть $\alpha_1 j_1 \alpha_2 j_2 \dots \alpha_l j_l$ является дельта-последовательностью для n-бнтового цикла Грэя, где каждое значение j_k является одной координатой, каждое значение α_k может быть пустой последовательностью координат, а l — четно. Тогда*

$$\begin{aligned} &\alpha_1(n+1)\alpha_1^R n \alpha_1 \\ &\quad j_1\alpha_2 n\alpha_2^R(n+1)\alpha_2\ j_2\alpha_3(n+1)\alpha_3^R n\alpha_3\ \dots\ j_{l-1}\alpha_l(n+1)\alpha_l^R n\alpha_l \\ &\qquad\qquad (n+1)\alpha_l^R j_{l-1}\alpha_{l-1}^R \dots \alpha_2^R j_1 \alpha_1^R n \end{aligned} \tag{27}$$

является дельта-последовательностью $(n + 2)$-битового цикла Грэя.

Например, если начать с последовательности $0\underline{1}0\underline{2}0\underline{1}0\underline{2}$ для $n=3$, где три подчеркнутых элемента являются j_1, j_2, j_3, то новая последовательность (27) для 5-битового цикла имеет вид

$$01410301020131024201043401020103. \tag{28}$$

Доказательство. Пусть α_k имеет длину m_k, а v_{kt} является достигнутой вершиной, если начать с $0\ldots0$ и применить изменения координат $\alpha_1 j_1 \ldots \alpha_{k-1} j_{k-1}$ и первые t из α_k. Нужно доказать, что все вершины $00v_{kt}$, $01v_{kt}$, $10v_{kt}$ и $11v_{kt}$ встречаются при использовании (27) для $1 \le k \le l$ и $0 \le t \le m_k$. (Самой левой координатой является $n+1$.)

Начиная с $000\ldots0 = 00v_{10}$, переходим к вершинам

$$00v_{11}, \ldots, 00v_{1m_1}, 10v_{1m_1}, \ldots, 10v_{10}, 11v_{10}, \ldots, 11v_{1m_1}.$$

Затем j_1 приводит к $11v_{20}$, за которым следует

$$11v_{21}, \ldots, 11v_{2m_2}, 10v_{2m_2}, \ldots, 10v_{20}, 00v_{20}, \ldots, 00v_{2m_2}.$$

Потом идет $00v_{30}$ и т.д., а в итоге получаем $11v_{lm_l}$. В заключение используется третья строка из (27) для генерации всех недостающих вершин $01v_{lm_l}, \ldots, 01v_{10}$ и возвращения к $000\ldots0$. ∎

Числа логических переходов $(c_0, \ldots, c_{n-1})$ в дельта-последовательности определяются, полагая, что c_j равняется количеству раз, когда $\delta_k = j$. Например, для (28) числа логических переходов равны $(12, 8, 4, 4, 4)$ и определяются на основании последовательности с числами логических переходов $(4, 2, 2)$. Если тщательно выбрать исходную дельта-последовательность и подчеркнуть соответствующий элемент j_k, то можно получить максимально близкие (насколько это возможно) числа логических переходов.

Следствие B. *Для всех $n \ge 1$ существует n-битовый цикл Грэя с числами логических переходов $(c_0, c_1, \ldots, c_{n-1})$, которые удовлетворяют условию*

$$|c_j - c_k| \le 2 \qquad \text{для } 0 \le j < k < n. \tag{29}$$

(Это наилучшее возможное условие равновесия, поскольку каждое значение c_j должно быть четным и $c_0 + c_1 + \cdots + c_{n-1} = 2^n$. Действительно, условие (29) выполняется тогда и только тогда, когда $n - r$ чисел равны $2q$ и r чисел равны $2q+2$, где $q = \lfloor 2^{n-1}/n \rfloor$ и $r = 2^{n-1} \bmod n$.)

Доказательство. Для заданной дельта-последовательности для n-битового цикла Грэя с числами логических переходов $(c_0, \ldots, c_{n-1})$ числа для цикла (27) получаются, начиная со значений $(c'_0, \ldots, c'_{n-1}, c'_n, c'_{n+1}) = (4c_0, \ldots, 4c_{n-1}, l+1, l+1)$, затем вычитанием 2 из c'_{j_k} для $1 \le k < l$ и вычитанием 4 из c'_{j_l}. Например, для $n=3$ можно получить 5-битовый сбалансированный цикл Грэя с числами логических переходов $(8-2, 16-10, 8, 6, 6) = (6, 6, 8, 6, 6)$, если применить теорему D для дельта-последовательности $0\underline{1}2\underline{1}0\underline{1}2\underline{1}$. В упр. 51 подробно рассматриваются случаи с другими значениями n. ∎

Другой важный класс n-битовых циклов Грэя, в которых каждый трек координат обладает равной ответственностью, возникает при изучении *длины серии*,

т.е. расстояния между последовательными появлениями того же значения δ. Стандартный двоичный код Грэя имеет длину серии 2 для самого младшего положения, и это может привести к потере точности при необходимости выполнения точных измерений [см., например, рассуждения Дж. М. Лоуренса (G. M. Lawrence) и В. Э. Мак-Клинтока (W. E. McClintock) в *Proc. SPIE* **2831** (1996), p.104–111]. Но все серии имеют длину 4 или более в замечательном 5-битовом цикле Грэя, дельта-последовательность которого имеет вид

$$(0123042103210423)^2. \tag{30}$$

Пусть $r(n)$ является таким максимальным значением r, что можно найти такой n-битовый цикл Грэя, в котором все серии имеют длину $\geq r$. Ясно, что $r(1) = 1$, а $r(2) = r(3) = r(4) = 2$. Легко видеть, что $r(n)$ должно быть меньше n для $n > 2$. Поэтому из (30) следует, что $r(5) = 4$. Исчерпывающий компьютерный поиск дает $r(6) = 4$ и $r(7) = 5$. Действительно, для прямого вычисления с возвратом для случая $n = 7$ требуется дерево приблизительно всего с 60 миллионами узлов для определения того, что $r(7) < 6$. В упражнении 61(a) используется 7-битовый цикл с длинами серий не менее 5. Точные значения $r(n)$ неизвестны для $n \geq 8$, но значение $r(10)$ почти определенно равно 8. Существуют интересные соображения, с помощью которых можно доказать, что $r(n) = n - O(\log n)$, если $n \to \infty$. (См. упр. 60–64.)

***Двоичные пути Грэя.** Ранее n-битовый цикл Грэя был определен как способ упорядочения всех двоичных n-кортежей в виде последовательности $(v_0, v_1, \ldots, v_{2^n-1})$ с таким свойством, что v_k является смежным для v_{k+1} в n-кубе для $0 \leq k < 2^n$, и таким, что v_{2^n-1} является смежным для v_0. Это свойство цикличности прекрасно, но не всегда имеет важное значение, и порой можно добиться большего даже без него. Поэтому n-битовым *путем Грэя*, или *кодом Грэя*, является любая последовательность, которая удовлетворяет условиям цикла Грэя, за исключением того, что последний элемент необязательно должен быть смежным с первым элементом. Иначе говоря, цикл Грэя является *циклом* Гамильтона на вершинах n-куба, но код Грэя является просто *путем* Гамильтона на этом графе.

Наиболее важным двоичным путем Грэя, который также не является циклом Грэя, является n-битовая *монотонная* последовательность $(v_0, v_1, \ldots, v_{2^n-1})$, в которой соблюдается условие

$$\nu(v_k) \leq \nu(v_{k+2}) \qquad \text{для } 0 \leq k < 2^n - 2. \tag{31}$$

(Здесь и далее ν используется для обозначения веса или продольной контрольной суммы двоичной строки, а именно количества единиц в ней.) Метод проб и ошибок свидетельствует, что существует только два монотонных n-битовых кода Грэя для каждого $n \leq 4$: один начинается с 0^n, а другой — с $0^{n-1}1$. Для $n = 3$ они имеют вид

$$000, 001, 011, 010, 110, 100, 101, 111; \tag{32}$$
$$001, 000, 010, 110, 100, 101, 111, 011. \tag{33}$$

Для $n = 4$ они менее очевидны, но определяются без особого труда.

Поскольку $\nu(v_{k+1}) = \nu(v_k) \pm 1$ всякий раз, когда v_k является смежным с v_{k+1}, то очевидно нельзя усилить (31) требованием, что все n-кортежи отсортированы

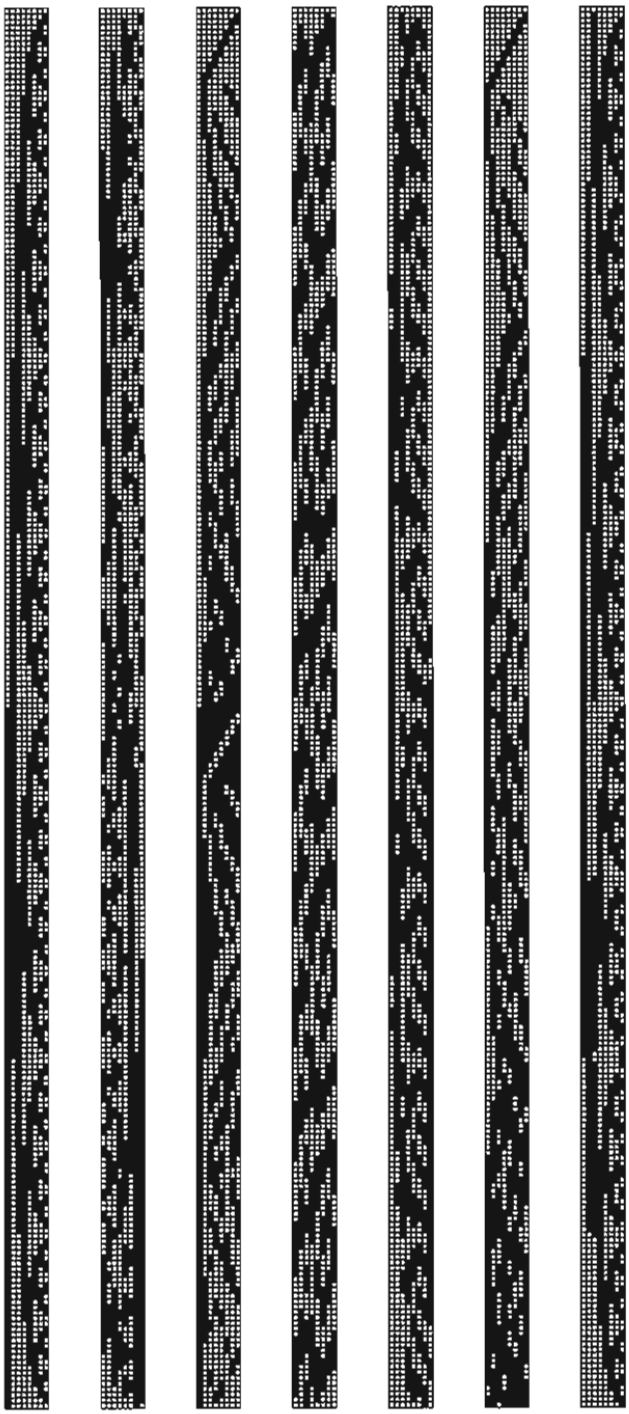


Рис. 14. Примеры 8-битового кода Грэя:

a) стандартный;
b) сбалансированный;
c) дополняющий;
d) длинносерийный;
e) нелокальный;
f) монотонный;
g) бестрендовый.

.рого по весу. Но (31) достаточно сильно для определения веса каждого v_k, для заданного k и веса v_0, поскольку нам известно, что $\binom{n}{j}$ из n-кортежей имеют вес j.

На рис. 14 показано всего семь из несметного количества кодов Грэя, которые перечисляют все 256 возможных 8-битовых байтов. Черные квадраты представляют единицы, а белые квадраты — нули. На рис. 14,*a* показан стандартный двоичный код Грэя,а на рис. 14,*b* показан сбалансированный двоичный код Грэя с точно $256/8 = 32$ логическими переходами в каждом положении координаты. На рис. 14,*c* показан код Грэя, аналогичный коду на рис. 13,*a*, но в котором иижние 128 кодов дополияютверхние 128 кодов. На рис. 14,*d* показан длинносерийный код, в котором логическне переходы в каждом положении координаты происходят не ближе, чем в пяти шагах. Иначе говоря, все серии имеют по крайней мере длину 5. Цикл на рис. 14,*e* является *нелокальным*, смысл которого объясняется в упражнении 59. На рис. 14,*f* показан монотонный путь для $n = 8$. Обратите внимаиие на плавное почернение рисунка в иижей части. Наконец, на рис. 14,*g* показан полностью немоиотонный код Грэя, в том смысле, что центр масс черных квадратов находится точно посередине каждого столбца. Стандартиый двончный код Грэя обладает таким же свойством в семи положениях координат, ио на рис. 14,*g* достигается полное равновесие во всех восьми столбцах. Такой код называется *бестрендовым*. Он имеет большое зиачение в аграрной отрасли и других видах деятельности (см. упр. 75 и 76).

Карла Сэвидж (Carla Savage) и Питер Винклер (Peter Winkler) [*J. Combinatorial Theory* **A70** (1995), p.230–248] нашли элегантный способ создания монотонных двоичных кодов Грэя для всех $n > 0$. Такие пути обязательно строятся на основе подчиненных путей P_{nj}, в которых все логические переходы совершаются между n-кортежами весов j и $j+1$. Сэвидж и Виикле определили подходящие подчиненные пути, рекурсивно допуская, что $P_{10} = 0, 1$, и для всех $n > 0$ имеем

$$P_{(n+1)j} = 1P^{\pi_n}_{n(j-1)},\ 0P_{nj}; \tag{34}$$

$$P_{nj} = \Lambda, \quad \text{если } j < 0 \text{ или } j \geq n, \tag{35}$$

где π_n обозиачает перестановку координат, которая описывается далее, а P^π обозиачает такую перестановку координат, что каждый элемент $a_{n-1}\ldots a_1a_0$ последовательности P заменяется элемеитом $b_{n-1}\ldots b_1b_0$, где $b_{j\pi} = a_j$. (Здесь для P^π не используется условие $b_j = a_{j\pi}$, поскольку нужно, чтобы $(2^j)^\pi$ равнялось $2^{j\pi}$.) Отсюда следует, иапример, что

$$P_{20} = 0P_{10} = 00,\ 01, \tag{36}$$

поскольку $P_{1(-1)}$ пуста; также

$$P_{21} = 1P^{\pi_1}_{10} = 10,\ 11, \tag{37}$$

поскольку P_{11} пуста и π_1 должна быть тождественной подстановкой. В общем случае P_{nj} является последовательностью n-битовых строк, содержащих точно $\binom{n-1}{j}$ строк с весом j, которые перемежаются $\binom{n-1}{j}$ строками с весом $j+1$.

Пусть α_{nj} и ω_{nj} являются первым и последним элементами P_{nj}. Тогда мы легко находим, что

$$\omega_{nj} = 0^{n-j-1}1^{j+1} \qquad \text{для } 0 \leq j < n; \tag{38}$$

$$\alpha_{n0} = 0^n \qquad \text{для } n > 0; \tag{39}$$

$$\alpha_{nj} = 1\alpha_{(n-1)(j-1)}^{\pi_{n-1}} \qquad \text{для } 1 \le j < n. \tag{4}$$

В частности, α_{nj} всегда имеет вес j и ω_{nj} всегда имеет вес $j+1$. Определим перестановки π_n of $\{0, 1, \ldots, n-1\}$ так, что обе последовательности

$$P_{n0},\ P_{n1}^R,\ P_{n2},\ P_{n3}^R,\ \ldots \tag{41}$$

$$\text{и } P_{n0}^R,\ P_{n1},\ P_{n2}^R,\ P_{n3},\ \ldots \tag{42}$$

являются монотонными двоичными путями Грэя для $n = 1, 2, 3, \ldots$. Действительно, монотонность очевидна, и сомнения остаются только по поводу приналежности к коду Грэя. Последовательности (41), (42) хорошо связываются, поскольку смежность элементов

$$\alpha_{n0} \text{ — } \alpha_{n1} \text{ — } \cdots \text{ — } \alpha_{n(n-1)}, \qquad \omega_{n0} \text{ — } \omega_{n1} \text{ — } \cdots \text{ — } \omega_{n(n-1)} \tag{43}$$

непосредственно следует из (34) независимо от перестановок π_n. Таким образом, ключевым пунктом является переход в месте запятой в формуле (34), который делает $P_{(n+1)j}$ подчиненным путем Грэя, тогда и только тогда, когда

$$\omega_{n(j-1)}^{\pi_n} = \alpha_{nj} \qquad \text{для } 0 < j < n. \tag{44}$$

Например, если $n = 2$ и $j = 1$, приходим к необходимости $(01)^{\pi_2} = \alpha_{21} = 10$, согласно (38)–(40). Следовательно, π_2 приводит к перестановке координат 0 и 1. Общая формула (см. упр. 71) принимает вид

$$\pi_n = \sigma_n \pi_{n-1}^2, \tag{45}$$

где σ_n является n-циклом $(n-1\ \ldots\ 1\,0)$. Следовательно, первые несколько случаев имеют вид

$$\pi_1 = (0), \qquad \pi_4 = (0\,3),$$
$$\pi_2 = (0\,1), \qquad \pi_5 = (0\,4\,3\,2\,1),$$
$$\pi_3 = (0\,2\,1), \qquad \pi_6 = (0\,5\,2\,4\,1\,3).$$

Похоже, не видно замкнутой формы для магических перестановок π_n. В упр. 73 показано, как могут эффективно генерироваться коды Сэвидж–Винклера.

Недвоичные коды Грэя. Мы очень подробно рассмотрели случай двоичных n-кортежей, поскольку это простейшая, классическая, наиболее часто используемая и наиболее исследованная область. Однако есть многочисленные приложения, в которых нужно генерировать $(a_1, \ldots, a_n)$ с координатами в более широком диапазоне $0 \le a_j < m_j$, как в алгоритме М. Коды Грэя также прекрасно подходят в этом случае.

Рассмотрим, например, десятичные цифры, где $0 \le a_j < 10$ для всех j. Существует ли в десятичной системе счисления способ подсчета, аналогичный двоичному коду Грэя, с изменением только одной цифры? Да, действительно, есть *две* естественные схемы. В первой схеме, которая называется *отраженным десятичным кодом Грэя*, последовательность для счета до тысячи с помощью 3-цифровых строк имеет вид

$$000, 001, \ldots, 009, 019, 018, \ldots, 011, 010, 020, 021, \ldots, 091, 090, 190, 191, \ldots, 900,$$

где каждая координата изменяется попеременно от 0 до 9 и затем обратно от 9 до 0. Во второй схеме, которая называется *модульным десятичным кодом Грэя*, цифры всегда увеличиваются на 1 по модулю 10, следовательно, онн "заворачиваются" от 9 до 0:

$$000, 001, \ldots, 009, 019, 010, \ldots, 017, 018, 028, 029, \ldots, 099, 090, 190, 191, \ldots, 900.$$

В обонх случаях цифра, которая изменяется на шаге k, определяется масштабной функцией с десятичным основанием $\rho_{10}(k)$, т.е. наибольшей степенью, в которой 10 делит k. Значит, каждый n-кортеж цифр появляется только один раз: мы генерируем 10^j разных состояний j самых правых цифр до изменения каких-либо других, для $1 \le j \le n$.

Вообще, отраженный код Грэя в любой смешанной системе счисления можно рассматривать как перестановку иеотрицательных целых чисел, т.е. как функцию, которая отображает обычиое число в смешанной системе счисления

$$k = \begin{bmatrix} b_{n-1}, & \ldots, & b_1, & b_0 \\ m_{n-1}, & \ldots, & m_1, & m_0 \end{bmatrix} = b_{n-1}m_{n-2}\ldots m_1 m_0 + \cdots + b_1 m_0 + b_0 \tag{46}$$

в ее эквивалент в отраженном коде Грэя

$$\hat{g}(k) = \begin{bmatrix} a_{n-1}, & \ldots, & a_1, & a_0 \\ m_{n-1}, & \ldots, & m_1, & m_0 \end{bmatrix} = a_{n-1}m_{n-2}\ldots m_1 m_0 + \cdots + a_1 m_0 + a_0, \tag{47}$$

как в особом случае двоичных чисел в уравнении (7). Пусть

$$A_j = \begin{bmatrix} a_{n-1}, & \ldots, & a_j \\ m_{n-1}, & \ldots, & m_j \end{bmatrix}, \qquad B_j = \begin{bmatrix} b_{n-1}, & \ldots, & b_j \\ m_{n-1}, & \ldots, & m_j \end{bmatrix}, \tag{48}$$

где $A_n = B_n = 0$ так, что при $0 \le j < n$ имеем

$$A_j = m_j A_{j+1} + a_j \qquad \text{и} \qquad B_j = m_j B_{j+1} + b_j. \tag{49}$$

Нетрудно по индукции вывести правило соединения a и b для $n - j$:

$$a_j = \begin{cases} b_j, & \text{если } B_{j+1} \text{ четно;} \\ m_j - 1 - b_j, & \text{если } B_{j+1} \text{ нечетно.} \end{cases} \tag{50}$$

(Здесь координаты n-кортежей $(a_{n-1}, \ldots, a_1, a_0)$ и $(b_{n-1}, \ldots, b_1, b_0)$ перечисляются справа налево для соответствия с (7) и соглашениями смешанной системы счисления в уравнении 4.1–(9). Читатели, которые предпочитают обозначения типа $(a_1, \ldots, a_n)$, могут замеиить j на $n - j$ во всех формулах, если есть такое желание.) В обратном направлении имеем:

$$b_j = \begin{cases} a_j, & \text{если } a_{j+1} + a_{j+2} + \cdots \text{ четно;} \\ m_j - 1 - a_j, & \text{если } a_{j+1} + a_{j+2} + \cdots \text{ нечетно.} \end{cases} \tag{51}$$

Забавно, что правило (50) н обратное ему правило (51) будут идеитичиы, если все основания m_j иечетны. Например, в троичиом коде Грэя, где $m_0 = m_1 = \cdots = 3$, имеем: $\hat{g}\big((10010211012)_3\big) = (12210211010)_3$ и $\hat{g}\big((12210211010)_3\big) = (10010211012)_3$. В упр. 78 доказываются (50) и (51) и обсуждаются аналогичиые формулы, которые справедливы в модульном случае.

Действительно, обобщая алгоритмы M и L, можно генерировать такие последовательности Грэя без циклов:

Алгоритм H (*генерация без циклов с отраженным кодом Грэя в смешанной системе счисления*). Этот алгоритм посещает все такие n-кортежи $(a_{n-1},\ldots,a_0)$, где $0\le a_j<m_j$ для $0\le j<n$, изменяя только одну координату на ±1 на каждом этапе. Он поддерживает массив указателей фокуса $(f_n,\ldots,f_0)$ для управления действиями, как в алгоритме L, вместе с массивом направлений $(o_{n-1},\ldots,o_0)$. Предполагается, что каждое основание $m_j\ge2$.

H1. [Инициализировать.] Установить $a_j\leftarrow0$, $f_j\leftarrow j$, и $o_j\leftarrow1$ для $0\le j<n$; также установить $f_n\leftarrow n$.

H2. [Посетить.] Посетить n-кортеж $(a_{n-1},\ldots,a_1,a_0)$.

H3. [Выбрать j.] Установить $j\leftarrow f_0$ и $f_0\leftarrow0$. (Как в алгоритме L, j был самой правой активной координатой, а все элементы справа теперь повторно активизированы.)

H4. [Изменить координату j.] Завершить, если $j=n$; в противном случае установить $a_j\leftarrow a_j+o_j$.

H5. [Отразить?] Если $a_j=0$ или $a_j=m_j-1$, то установить $o_j\leftarrow-o_j$, $f_j\leftarrow f_{j+1}$ и $f_{j+1}\leftarrow j+1$. (Координата j, таким образом, становится пассивной.) Вернуться к H2. ▮

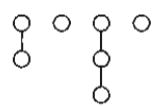
Аналогичный алгоритм генерирует модульную вариацию (см. упр. 77).

***Подчиненный лес.** Интересное и поучительное обобщение алгоритма H, открытое И. Кода (Y. Koda) и Ф. Раски (F. Ruskey) [*J. Algorithms* **15** (1993), p.324–340], пролнвает свет на дополнительные особенностн кода Грэя и бесцикловой генерации. Допустим, что мы имеем лес n узлов и нам нужно посетить все основные подлеса, а именно все такие подмножества узлов S, что если x иаходится в S и x не является корнем, то родитель x также находится в S. Например, 7-узловой лес  имеет 33 таких подмножества, соответствующих черным узлам на следующих 33 диаграммах:

 (52)

Обратите внимание на то, что если верхний ряд считывать слева направо, средиий ряд — справа налево, а нижний ряд — слева направо, то на каждом шаге изменяется состояние точно одного узла.

Если указанный лес состоит из вырожденных неветвящихся деревьев, то основные подлеса эквивалентны числам со смешаниым осиованием. Например, лес

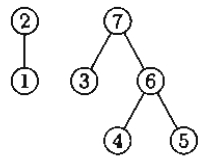


имеет $3\times2\times4\times2$ осиовных подлесов, соответствующих таким 4-ным кортежам (x_1,x_2,x_3,x_4), что $0\le x_1<3$, $0\le x_2<2$, $0\le x_3<4$ и $0\le x_4<2$. Зиаченне x_j равно количеству узлов, выделенных в j-м дереве. Если алгоритм Кода и Рас-

ки применяется к такому лесу, то он посещает подлеса в том же порядке, что и отраженный код Грэя для оснований $(3, 2, 4, 2)$.

Алгоритм К (*отраженная генерация подлесов без циклов*). Для заданного леса, узлы которого $(1, \ldots, n)$ упорядочены в обратном порядке, этот алгоритм посещает все такие двоичные n-кортежи $(a_1, \ldots, a_n)$, для которых $a_p \geq a_q$ всякий раз, когда p является родителем q. (Таким образом, $a_p = 1$ означает, что p является узлом в текущем подлесу.) Точно один бит a_j изменяется между одним и следующим посещениями. Указатели фокуса $(f_0, f_1, \ldots, f_n)$, аналогичные указателям фокуса из алгоритма L, используются вместе с дополнительными массивами указателей $(l_0, l_1, \ldots, l_n)$ и $(r_0, r_1, \ldots, r_n)$, которые представляют собой дважды связанные списки или текущую полосу. Текущая полоса содержит все узлы текущего подлеса и их дочерних узлов; r_0 указывает на самый левый узел, а l_0 — на самый правый узел.

Вспомогательный массив $(c_0, c_1, \ldots, c_n)$ определяет лес следующим образом: если p не имеет дочерних узлов, то $c_p = 0$. В противном случае c_p является самым левым (наименьшим) дочерним узлом p. Кроме того, c_0 является самым левым корнем самого леса. В начале алгоритма предполагается, что $r_p = q$ и $l_q = p$ всякий раз, когда p и q являются последовательными дочерними узлами одного семейства. Таким образом, например, лес в (52) имеет обратный порядок нумерации:



Следовательно, в этом случае имеем: $(c_0, \ldots, c_7) = (2, 0, 1, 0, 0, 0, 4, 3)$ и $r_2 = 7$, $l_7 = 2$, $r_3 = 6$, $l_6 = 3$, $r_4 = 5$ и $l_5 = 4$ в начале этапа К1.

К1. [Инициализировать.] Установить $a_j \leftarrow 0$ и $f_j \leftarrow j$ для $1 \leq j \leq n$, опустошая таким образом исходный подлес и активизируя все узлы. Установить $f_0 \leftarrow 0$, $l_0 \leftarrow n$, $r_n \leftarrow 0$, $r_0 \leftarrow c_0$ и $l_{c_0} \leftarrow 0$, помещая таким образом все корни в текущую полосу.

К2. [Посетить.] Посетить подлес, заданный $(a_1, \ldots, a_n)$.

К3. [Выбрать p.] Установить $q \leftarrow l_0$, $p \leftarrow f_q$. (Теперь p является самым правым активным узлом полосы.) Также установить $f_q \leftarrow q$ (таким образом, активизируя все узлы справа от p).

К4. [Проверить a_p.] Завершить алгоритм, если $p = 0$. В противном случае перейти к К6, если $a_p = 1$.

К5. [Вставить дочерние узлы узла p.] Установить $a_p \leftarrow 1$. Затем, если $c_p \neq 0$, установить $q \leftarrow r_p$, $l_q \leftarrow p - 1$, $r_{p-1} \leftarrow q$, $r_p \leftarrow c_p$, $l_{c_p} \leftarrow p$ (таким образом, помещая дочерние узлы p справа от p в этой полосе). Перейти к К7.

К6. [Удалить дочерние узлы p.] Установить $a_p \leftarrow 0$. Затем, если $c_p \neq 0$, установить $q \leftarrow r_{p-1}$, $r_p \leftarrow q$, $l_q \leftarrow p$ (таким образом, удаляя дочерние узлы p из этой полосы).

К7. [Сделать p пассивным.] (На этом этапе нам известно, что p активен.) Установить $f_p \leftarrow f_{l_p}$ и $f_{l_p} \leftarrow l_p$. Вернуться к К2. ▮

Читателю предлагается поэкспериментировать с этим алгоритмом на примерах типа (52), чтобы изучить прекрасный механизм, с помощью которых полоса растет и сжимается как раз в нужное время.

***Последовательности со сдвигом регистра.** Также возможен совершенно иной способ генерации всех n-кортежей m-х цифр. Можно последовательно генерировать по одному биту и повторно работать с n самыми последними генерированными цифрами, таким образом, переходя от одного n-кортежа $(x_0, x_1, \ldots, x_{n-1})$ к другому $(x_1, \ldots, x_{n-1}, x_n)$, сдвигая соответствующую новую цифру вправо. Например, на рис. 15 показано, как можно получить все 5-битовые числа, как блоки 5 последовательных битов в некоторой циклической структуре с длиной 32. Эта общая идея уже обсуждалась в некоторых упражнениях разделов 2.3.4.2 и 3.2.2, и теперь можно приступать к ее изучению.

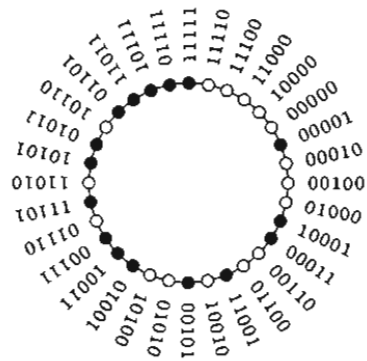


Рис. 15.
Цикл де Бруйна
для 5-битовых чисел

Алгоритм S (*обобщенная генерация со сдвигом регистра*). Этот алгоритм посещает все такие n-кортежи $(a_1, \ldots, a_n)$, что $0 \le a_j < m$ для $1 \le j \le n$, при условии, что на этапе S3 используется подходящая функция f.

S1. [Инициализировать.] Установить $a_j \leftarrow 0$ для $-n < j \le 0$ и $k \leftarrow 1$.

S2. [Посетить.] Посетить n-кортеж $(a_{k-n}, \ldots, a_{k-1})$. Завершить, если $k = m^n$.

S3. [Продвинуться.] Установить $a_k \leftarrow f(a_{k-n}, \ldots, a_{k-1})$, $k \leftarrow k + 1$ и вернуться к S2. ▮

Каждая функция f, которая подходит для алгоритма S, соответствует такому циклу m^n цифр с основанием m, что каждая комбинация n цифр возникает последовательно в этом цикле. Например, случай $m = 2$ и $n = 5$ (см. рис. 15) соответствует двоичному циклу

$$00000100011001010011101011011111; \tag{53}$$

и первые m^2 цифр бесконечной последовательности

$$00110212203132330414243 44\ldots \tag{54}$$

достигают соответствующего цикла для $n = 2$ и произвольного m. Такие циклы обычно называются m-ми *циклами де Бруйна*, поскольку Н. Г. де Бруйн рассмотрел двоичный случай для произвольного n в *Indagationes Mathematicæ* **8** (1946), p.461–467.

Таблица 1

ПАРАМЕТРЫ АЛГОРИТМА A

3 : 1	8 : 1,5	13 : 1,3	18 : 7	23 : 5	28 : 3
4 : 1	9 : 4	14 : 1,11	19 : 1,5	24 : 1,3	29 : 2
5 : 2	10 : 3	15 : 1	20 : 3	25 : 3	30 : 1,15
6 : 1	11 : 2	16 : 2,3	21 : 2	26 : 1,7	31 : 3
7 : 1	12 : 3,4	17 : 3	22 : 1,7	27 : 1,7	32 : 1,27

Элементы '$n : s$' или '$n : s, t$' означают, что многочлены $x^n + x^s + 1$ или $x^n + (x^s + 1)(x^t + 1)$ являются примитивными по модулю 2. Дополнительные значения табулированы до $n = 168$ [В. Штанке (W. Stahnke), *Math. Comp.* **27** (1973), p.977–980].

В упр. 2.3.4.2–23 доказывается, что точно $m!^{m^{n-1}}/m^n$ функций f имеют требуемые свойства. Это огромное число, но только некоторые из этих функций могут быть эффективно вычислены. Мы обсудим три типа функций f, которые представляются наиболее полезными.

Первый важный случай возникает, когда m является простым числом, а f — почти линейным рекуррентным соотношением:

$$f(x_1, \ldots, x_n) = \begin{cases} c_1, & \text{если } (x_1, x_2, \ldots, x_n) = (0, 0, \ldots, 0); \\ 0, & \text{если } (x_1, x_2, \ldots, x_n) = (1, 0, \ldots, 0); \\ (c_1 x_1 + c_2 x_2 + \cdots + c_n x_n) \bmod m & \text{в противном случае.} \end{cases} \tag{55}$$

Здесь коэффициенты $(c_1, \ldots, c_n)$ должны быть такими, что

$$x^n - c_n x^{n-1} - \cdots - c_2 x - c_1 \tag{56}$$

является примитивным многочленом по модулю m в смысле, упомянутом после формулы 3.2.2–(9). Количество таких многочленов равно $\varphi(m^n - 1)/n$, и оно достаточно велико, чтобы среди них можно было найти один, в котором только несколько коэффициентов c не равны нулю. [Эта конструкция берет свое начало в пионерской статье Виллема Мантеля (Willem Mantel), *Nieuw Archief voor Wiskunde* (2) **1** (1897), p.172–184.]

Например, предположим, что $m = 2$. Тогда можно генерировать двоичные n-кортежи с помощью очень простой бесцикловой процедуры.

Алгоритм A (*почти линейная генерация со сдвигом битов*). В этом алгоритме посещаются все n-битовые векторы либо с помощью одного специального отступа s [случай 1], либо с помощью двух специальных отступов s и t [случай 2], как показано в табл. 1.

A1. [Инициализировать.] Установить $(x_0, x_1, \ldots, x_{n-1}) \leftarrow (1, 0, \ldots, 0)$ и $k \leftarrow 0$, $j \leftarrow s$. В случае 2 также установить $i \leftarrow t$ и $h \leftarrow s + t$.

A2. [Посетить.] Посетить n-кортеж $(x_{k-1}, \ldots, x_0, x_{n-1}, \ldots, x_{k+1}, x_k)$.

A3. [Проверить на окончание.] Если $x_k \neq 0$, установить $r \leftarrow 0$; в противном случае установить $r \leftarrow r + 1$ и перейти к A6, если $r = n - 1$. (Имеем r последовательных нулей.)

A4. [Сдвинуть.] Установить $k \leftarrow (k - 1) \bmod n$ и $j \leftarrow (j - 1) \bmod n$. В случае 2 также установить $i \leftarrow (i - 1) \bmod n$ и $h \leftarrow (h - 1) \bmod n$.

A5. [Вычислить новый бит.] Установить $x_k \leftarrow x_k \oplus x_j$ [случай 1] или $x_k \leftarrow x_k \oplus x_j \oplus x_i \oplus x_h$ [случай 2]. Вернуться к A2.

A6. [Завершить.] Посетить $(0, \ldots, 0)$ и завершить. ▮

Соответствующие параметры отступа s и, возможно, t практически всегда существуют для всех n, поскольку примитивных многочленов имеется достаточно много. Например, существует восемь разных случаев (s, t) для $n = 32$, а в табл. 1 перечислены лишь наименьшие. Однако строгое доказательство их существования во всех случаях выходит за рамки современного состояния математических знаний.

Наша первая конструкция циклов де Бруйна в (55) была алгебраической и основывалась на теории конечных полей. Аналогичный метод, который работает, когда m не является простым числом, представлен в упр. 3.2.2–21. Наша следующая конструкция, наоборот, будет только комбинаторной. Действительно, она сильно связана с идеей модульных m-х кодов Грэя.

Алгоритм R (*генерация на основе рекурсивного цикла де Бруйна*). Предположим, что $f()$ является сопрограммой, которая выводит последовательные цифры m-го цикла де Бруйна длины m^n, начиная с n нулей, при повторном вызове. Этот алгоритм аналогичен сопрограмме, которая выводит цикл длины m^{n+1} при условии, что $n \geq 2$. В нем используются три закрытых переменных x, y и t, а переменная x в исходном состоянии равна нулю.

R1. [Вывести.] Вывести x. Перейти к R3, если $x \neq 0$ и $t \geq n$.

R2. [Вызвать f.] Установить $y \leftarrow f()$.

R3. [Подсчитать единицы.] Если $y = 1$, установить $t \leftarrow t + 1$; в противном случае установить $t \leftarrow 0$.

R4. [Пропустить единицу?] Если $t = n$ и $x \neq 0$, вернуться к R2.

R5. [Подогнать x.] Установить $x \leftarrow (x + y) \bmod m$ и вернуться к R1. ▮

Например, пусть $m = 3$ и $n = 2$. Если $f()$ порождает бесконечный 9-й цикл

$$001102122\ 001102122\ 0\ldots, \tag{57}$$

то алгоритм R приводит к следующему бесконечному 27-му циклу на этапе R1:

$$\begin{aligned} y &= 001021220011110212200102122\,001\ldots \\ t &= 001001000012340010000100100\,001\ldots \\ x &= 000110102220120020211122121\,0001\ldots. \end{aligned}$$

Доказательство корректной работы алгоритма R интересно и поучительно (см. упр. 93). А доказательство корректной работы следующего алгоритма, в котором используется окно, размеры которого *вдвое* больше, чем n, еще более интересно и поучительно (см. упр. 95).

Алгоритм D (*генерация на основе двойного рекурсивного цикла де Бруйна*). Предположим, что $f()$ и $f'()$ являются сопрограммами, которые выводят последовательные цифры m-го цикла де Бруйна длины m^n при повторных вызовах, начиная с n нулей. (Эти два цикла идентичны, но они должны генерироваться независимыми сопрограммами, поскольку их значения будут использоваться с разной частотой.)

Этот алгоритм аналогичен сопрограмме, которая выводит цикл длины m^{2n}. Он содержит шесть закрытых переменных x, y, t, x', y' и t', а переменные x и x' в исходном состоянии равны m.

Специальный параметр r должен быть равен константе, для которой выполняются следующие условия:

$$0 \le r \le m \qquad \text{и} \qquad \gcd(m^n - r,\ m^n + r) = 2. \tag{58}$$

Обычно $r = 1$, если m нечетно, и $r = 2$, если m четно.

D1. [Возможно вызвать f.] Если $t \neq n$ или $x \ge r$, установить $y \leftarrow f()$.

D2. [Подсчитать повторы.] Если $x \neq y$, установить $x \leftarrow y$ и $t \leftarrow 1$. В противном случае установить $t \leftarrow t + 1$.

D3. [Вывести из f.] Вывести текущее значение x.

D4. [Вызвать f'.] Установить $y' \leftarrow f'()$.

D5. [Подсчитать повторы.] Если $x' \neq y'$, установить $x' \leftarrow y'$ и $t' \leftarrow 1$. В противном случае установить $t' \leftarrow t' + 1$.

D6. [Возможно отвергнуть f'.] Если $t' = n$ и $x' < r$ и либо $t < n$, либо $x' < x$, то перейти к D4. Если $t' = n$ и $x' < r$ и $x' = x$, то перейти к D3.

D7. [Вывести из f'.] Вывести текущее значенне x'. Вериуться к D3, если $t' = n$ и $x' < r$, в противном случае вернуться к D1. ∎

Основная идея алгоритма D состоит в том, чтобы выводить из $f()$ и $f'()$ попеременно, делая специальные настройки, когда любая из последовательностей генерирует n последовательных x для $x < r$. Например, когда $f()$ и $f'()$ порождают 9-й цикл (57), берем $r = 1$ и получаем:

t на этапе D2: 12 31211112 12312111 12123121 11121231 21111212 ...
x на этапе D3: 00001102122 00011021 22000110 21220001 102122000...
t' на этапе D6: 12121111212121111212121111212121111212121111212121 ...
x' на этапе D7: 0 11021220 11021220 11021220 11021220 11021220 1 ....

Итак, на этапах D3 и **D7** получается 81-й цикл **00001011012**...**2222 00001**....

Случай $m = 2$ для алгоритма R был открыт Абрахамом Лемпелем (Abraham Lempel) [*IEEE Trans.* **C-19** (1970), p.1204–1209]. Алгоритм D был открыт только двадцать пять лет спустя [C. J. Mitchell, T. Etzion, и K. G. Paterson, *IEEE Trans.* **IT-42** (1996), p.1472–1478]. Используя их совместно, начиная с простых сопрограмм для $n = 2$ на основе (54), можно создать интересное семейство взаимодействующих сопрограмм, которые будут генерировать цикл де Бруийна длины m^n для любого заданного $m \ge 2$ и $n \ge 2$, с помощью только $O(\log n)$ простых вычислений для каждой выведенной цифры. (см. упр. 96). Более того, в простейшем случае $m = 2$, эта комбинация "R&D" обладает таким свойством, что ее k-й вывод можно вычислить непосредственно как функцию k, выполняя $O(n \log n)$ простых операций на n-битовых числах. И наоборот, для любой заданной n-битовой структуры β положение β в цикле также можно вычислить за $O(n \log n)$ шагов (см. упр. 97–99). В настоящее время неизвестны никакие другие двоичные циклы де Бруийна, которые обладали бы последним свойством.

Наша третья конструкция цикла де Бруйна основана на теории простых строк, которые будут иметь огромное значение при изучении операции сопоставления с образцом в главе 9. Пусть $\gamma = \alpha\beta$ является конкатенацией двух строк, где α называется *префиксом* γ и β — *суффиксом*. Префикс или суффикс γ называется *собственным*, если его длина положительна, но меньше длины γ. Таким образом, β является собственным суффиксом $\alpha\beta$ тогда и только тогда, когда $\alpha \neq \epsilon$ и $\beta \neq \epsilon$.

Определение P. *Строка является простой, если оне не пуста и (лексикографическн) меньше, чем все ее собственные суффиксы.*

Например, строка 01101 не является простой, поскольку она больше 01. А строка 01102 является простой, поскольку оиа меньше 1102, 102, 02 и 2. (Предполагается, что строки состоят из букв, цифр или других символов из линейно упорядоченного алфавита. Лексикографический или словарный порядок является обычным способом сравнения строк, поэтому мы записываем, что $\alpha < \beta$, и говорим, что α меньше β, когда α лексикографически меньше β. В частности, $\alpha \leq \alpha\beta$ и $\alpha < \alpha\beta$ тогда и только тогда, когда $\beta \neq \epsilon$.)

Простые строки часто называют *словами Линдона*, поскольку они были введены Р. К. Линдоном (R. C. Lyndon) [*Trans. Amer. Math. Soc.* **77** (1954), p.202–215]. Линдон назвал их стандартными последовательностями, но более обоснованным является термин "простой", согласно фундаментальной теореме разложения на множители в упр. 101. Однако из уважения к заслугам Линдона мы будем часто использовать букву λ для обозначения простых строк.

Несколько наиболее важных свойств простых строк были выведены Ченом, Фоксом и Линдоном в важной статье по теории групп [*Annals of Math.* **68** (1958), p.81–95], которая содержала следующий простой, но важный результат.

Теорема P. *Непустая строка, которая меньше всех своих циклических сдвигов, является простой.*

(Циклическими сдвигами $a_1 \ldots a_n$ являются $a_2 \ldots a_n a_1$, $a_3 \ldots a_n a_1 a_2$, $\ldots$, $a_n a_1 \ldots a_{n-1}$.)

Доказательство. Пусть строка $\gamma = \alpha\beta$ не является пустой, поскольку $\alpha \neq \epsilon$ и $\gamma \geq \beta \neq \epsilon$. Но допустим, что γ также меньше, чем ее циклический сдвиг $\beta\alpha$. Тогда условия $\beta \leq \gamma < \beta\alpha$ подразумевают, что $\gamma = \beta\theta$ для некоторых строк $\theta < \alpha$. Следовательно, если γ также меньше своего циклического сдвига $\theta\beta$, то имеем: $\theta < \alpha < \alpha\beta < \theta\beta$. Но это невозможно, поскольку α и θ имеют одинаковую длину. ▮

Пусть $L_m(n)$ является количеством m-х простых строк длиной n. Каждая строка $a_1 \ldots a_n$ вместе с ее циклическими сдвигами дает d разных строк для иекоторого делителя d для n, соответствующего точно одной строке длиной d. Например, из 010010 с помощью циклического сдвига можно получить строки 100100 и 001001, а нанменьшей из периодическнх частей $\{010, 100, 001\}$ является простая строка 001. Следовательно, имеем:

$$\sum_{d \backslash n} d L_m(d) = m^n \qquad \text{для всех } m, n \geq 1. \tag{59}$$

Это семейство уравненнй можно решить для $L_m(n)$ с помощью упражнений 4.5.3–28(а) и получить:

$$L_m(n) = \frac{1}{n}\sum_{d\backslash n}\mu(d)m^{n/d}. \tag{60}$$

В 1970-х годах Гарольд Фредриксен (Harold Fredricksen) и Джеймс Майорана (James Maiorana) открыли простой способ генерации всех m-х простых строк длины n или меньше, в порядке возрастания [*Discrete Math.* **23** (1978), p.207–210]. Для понимания этого алгоритма нужно рассмотреть *n-расширение* непустых строк λ, а именно первых n символов бесконечных строк $\lambda\lambda\lambda\ldots$. Например, 10-расширение 123 имеет вид 1231231231. Вообще, если $|\lambda| = k$, то его n-расширение имеет внд $\lambda^{\lfloor n/k\rfloor}\lambda'$, гдеλ′ — префикс λ, длина которого равна $n \bmod k$.

Определенне Q. *Строка является предпростой, если она является непустым префиксом простой строки в иекотором алфавите.*

Теорема Q. *Строка длины $n > 0$ является предпростой тогда и только тогда, когда она является n-расширением простой строки λ длины $k \le n$. Эта простая строка определяется однозначно.*

Доказательство. См. упр. 105. ∎

Теорема Q, по сути, утверждает, что существует взаимно-однозначное соответствие между простыми строками длины $\le n$ и предпростыми строками длины n. Следующий алгоритм генерирует все m-е экземпляры в порядке возрастания.

Алгоритм F (*генерация простых и предпростых строк*). Этот алгоритм посещает все такие m-е n-кортежи $(a_1,\ldots,a_n)$, что строка $a_1\ldots a_n$ является предпростой. Он также определяет такой индекс j, что $a_1\ldots a_n$ является n-расшнрением простой строки $a_1\ldots a_j$.

F1. [Инициализировать.] Установить $a_1 \leftarrow \cdots \leftarrow a_n \leftarrow 0$ и $j \leftarrow 1$; также установить $a_0 \leftarrow -1$.

F2. [Посетить.] Посетить $(a_1,\ldots,a_n)$ с индексом j.

F3. [Подготовить к увеличению.] Установить $j \leftarrow n$. Затем, если $a_j = m-1$, уменьшать j до тех пор, пока не обнаружнтся $a_j < m-1$.

F4. [Добавить единицу.] Завершить, если $j = 0$. В противном случае установить $a_j \leftarrow a_j + 1$. (Теперь строка $a_1\ldots a_j$ является простой, согласио результату в упр. 105(а).)

F5. [Сделать n-расширение.] Для $k \leftarrow j+1, \ldots, n$ (в этом порядке) установить $a_k \leftarrow a_{k-j}$. Вернуться к этапу F2. ∎

Например, алгоритм F посещает 32 троичных предпростых строки при $m = 3$ и $n = 4$:

$$\begin{array}{cccccccc}
\hat{0}000 & 00\hat{1}1 & 00\hat{2}2 & 01\hat{1}1 & 01\hat{2}2 & 02\hat{1}2 & 11\hat{1}1 & 12\hat{1}2\\
000\hat{1} & 001\hat{2} & 010\hat{1} & 011\hat{2} & 020\hat{2} & 022\hat{0} & 111\hat{2} & 122\hat{1}\\
00\hat{2}0 & 002\hat{0} & 01\hat{0}2 & 012\hat{0} & 02\hat{1}0 & 02\hat{2}1 & 11\hat{2}1 & 12\hat{2}2\\
00\hat{1}0 & 002\hat{1} & 011\hat{0} & 012\hat{1} & 021\hat{1} & 022\hat{2} & 11\hat{2}2 & 22\hat{2}2
\end{array} \tag{61}$$

(Цифры перед символом ^ являются простыми строками 0, 0001, 0002, 001, 0011, ..., 2.)

Теорема Q объясняет, почему этот алгоритм является корректным: этапы F3 и F4, очевидно, позволяют найти наименьшую m-ю строку длиной $\le n$, которая превышает предыдущую предпростую строку $a_1 \dots a_n$. Обратите внимание на то, что после увеличения a_1 от 0 до 1 этот алгоритм посещает все $(m-1)$-е простые и предпростые строки, увеличенные на $1 \dots 1$.

Алгоритм F, конечно, прекрасен, но какое отношение он имеет к циклам де Бруйна? Дело вот в чем: если выводить цифры $a_1, \dots, a_j$ на этапе F2, когда j является делителем n, то последовательность всех таких цифр образует цикл де Бруйна! Например, в случае $m = 3$ и $n = 4$ следующая 81 цифра образует:

$$0\,0001\,0002\,0011\,0012\,0021\,0022\,01\,0102\,0111\,0112$$
$$0121\,0122\,02\,0211\,0212\,0221\,0222\,1\,1112\,1122\,12\,1222\,2. \qquad (62)$$

(Здесь опущены простые строки 001, 002, 011, ..., 122 из (61), поскольку их длина не делится на 4.) Причины этих почти магических свойств описываются в упр. 108. Обратите внимание на то, что цикл имеет корректную длину, согласно (59).

В некотором смысле выход этой процедуры фактически эквивалентен "дедушке" всех циклов де Бруйна для всех m и n, а именно конструкции, впервые опубликованной в работе М. Х. Мартина (M. H. Martin) [*Bull. Amer. Math. Soc.* **40** (1934), p.859–864]. Оригинальный цикл Мартина для $m = 3$ и $n = 4$ имел вид 2222122202211...10000, т.е. дополнение до двух для (62). Действительно, Гарольд Фредриксен (Harold Fredricksen) и Джеймс Майорана (James Maiorana) открыли алгоритм F почти случайно, в поисках простого способа генерации последовательности Мартина. Явная связь между их алгоритмом и предпростыми строками не была замечена еще много лет, пока Раски, Сэвидж и Ваиг не выполнили тщательный анализ времени выполнения [*J. Algorithms* **13** (1992), p.414–430]. В упр. 107 представлены основные результаты этого анализа, а именно:

i) Среднее значение $n - j$ на этапах F3 и F5 приблизительно равно $1/(m-1)$.

ii) Общее время работы алгоритма для создания цикла де Бруйна, как в (62), равно $O(m^n)$.

УПРАЖНЕНИЯ

1. [*10*] Объясните, как генерировать все n-кортежи $(a_1, \dots, a_n)$, для которых соблюдается условие $l_j \le a_j \le u_j$, где для каждой координаты определены нижняя, l_j, и верхняя граница, u_j. (Предполагается, что $l_j \le u_j$.)

2. [*15*] Каким будет 1000000-й n-кортеж, посещенный алгоритмом M, если $n = 10$ и $m_j = j$ для $1 \le j \le n$? *Подсказка:* $\left[\begin{smallmatrix}0,&0,&1,&2,&3,&0,&2,&7,&1,&0\\1,&2,&3,&4,&5,&6,&7,&8,&9,&10\end{smallmatrix}\right] = 1000000$.

3. [*M20*] Сколько раз алгоритм M выполняет этап M4?

4. [*18*] В большинстве компьютеров быстрее вести счет по убыванию до 0, чем по возрастанию до m. Переделайте алгоритм M так, чтобы он посещал все n-кортежи в обратном порядке, начиная с $(m_1 - 1, \dots, m_n - 1)$ и заканчивая $(0, \dots, 0)$.

5. [*20*] Алгоритмы быстрого преобразования Фурье (в упражнениях 4.6.4–14) часто заканчиваются массивом ответов в порядке обращения битов с $A[(b_0 \dots b_{n-1})_2]$ вместо $A[(b_{n-1} \dots b_0)_2]$. Предложите подходящий способ переупорядочения ответов в правильном порядке. [*Подсказка.* Обратите алгоритм M.]

6. [*M17*] Докажите справедливость формулы (7), т.е. основной формулы двоичного кода Грэя.

7. [*20*] На рис. 10,*b* показан двоичный код Грэя для диска, который поделен на 16 секторов. Предложите эффективный код, аналогичный коду Грэя, для диска с 12 или 60 секторами (как на циферблате с указанием часов и минут) или с 360 секторами (как на транспортире с указанием углов в градусах).

8. [*15*] Предложите простой способ прохода всех n-битовых строк с проверкой на четность, изменяя только два бита на каждом этапе.

9. [*16*] Какой ход следует сделать в соответствии с рис. 11 при решении головоломки "китайское переплетение колец"?

▸ **10.** [*M21*] Найдите простую формулу для общего количества шагов A_n или B_n, в которой кольцо (a) снимается или (b) заменяется, в кратчайшей процедуре снятия n китайских колец. Например, $A_3 = 4$ и $B_3 = 1$.

11. [*M22*] (Х. Дж. Перкис (H. J. Purkiss), 1865.) Два самых маленьких кольца китайской головоломки можно снимать или одевать одновременно. Сколько шагов потребуется, если использовать такие ускоренные шаги?

▸ **12.** [*25*] *Композициями* n называются последовательности положительных целых чисел, которые в сумме дают n. Например, композициями 4 являются 1111, 112, 121, 13, 211, 22, 31 и 4. Целое число n имеет точно 2^{n-1} композиций, соответствующих всем подмножествам точек $\{1, \ldots, n-1\}$, которые могут использоваться для разбиения интервала $(0\,..\,n)$ на подчиненные интервалы целочисленного размера.

a) Придумайте бесцикловый алгоритм для генерации всех композиций n, представляющих каждую композицию в виде последовательного массива целых чисел $s_1 s_2 \ldots s_j$.

b) Аналогично, придумайте бесцикловый алгоритм, который неявно представляет композиции в виде массива указателей $q_0 q_1 \ldots q_t$, где элементами композиции являются $(q_0 - q_1)(q_1 - q_2)\ldots(q_{t-1} - q_t)$, а $q_0 = n$, $q_t = 0$. Например, композиция 211 может быть представлена в этой схеме указателями $q_0 = 4$, $q_1 = 2$, $q_2 = 1$, $q_3 = 0$, где $t = 3$.

13. [*21*] В продолжение предыдущего упражнения вычислите также коэффициенты многочлена $C = \binom{s_1, \ldots, s_j}{n}$ для использования в качестве посещаемой композиции $s_1 \ldots s_j$.

14. [*20*] Предложите алгоритм генерации всех таких строк $a_1 \ldots a_j$, что $0 \le j \le n$ и $0 \le a_i < m_i$ для $1 \le i \le j$, в лексикографическом порядке. Например, если $m_1 = m_2 = n = 2$, то этот алгоритм должен последовательно посещать ϵ, 0, 00, 01, 1, 10, 11.

▸ **15.** [*25*] Предложите алгоритм генерации строк из предыдущего упражнения. Все строки одинаковой длины нужно "посетить" (т.е. перебрать) в лексикографическом порядке, как и раньше, но строки разной длины можно перемешивать в любом удобном порядке. Например, 0, 00, 01, ϵ, 10, 11, 1 находятся в приемлемом порядке, когда $m_1 = m_2 = n = 2$.

16. [*23*] Бесцикловый алгоритм не может генерировать все двоичные векторы $(a_1, \ldots, a_n)$ в лексикографическом порядке, поскольку количество координат a_j, которое необходимо сменить между последовательными посещениями, не ограничено. Покажите, что бесцикловая лексикографическая генерация становится возможной, если *связанное* представление используется вместо последовательного. Предположим, что есть $2n+1$ узлов $\{0, 1, \ldots, 2n\}$, причем каждый содержит поле LINK. Двоичный n-кортеж $(a_1, \ldots, a_n)$ представляется следующим образом:

$$\texttt{LINK}(0) = 1 + na_1;$$
$$\texttt{LINK}(j - 1 + na_{j-1}) = j + na_j \qquad \text{для } 1 < j \le n;$$
$$\texttt{LINK}(n + na_n) = 0,$$

а другие поля n LINK могут иметь любые удобные значения.

17. [*20*] Хорошо известная конструкция, которая называется *картой Карно* [M. Karnaugh, *Amer. Inst. Elect. Eng. Trans.* **72**, part I (1953), p.593–599], использует двоичный код Грэя в двух измерениях для отображения всех 4-битовых чисел в виде тора 4×4:

0000	0001	0011	0010
0100	0101	0111	0110
1100	1101	1111	1110
1000	1001	1011	1010

(Элементы тора "свертываются" слева и справа, а также сверху и снизу, как если бы они были частью мозаики, покрывающей плоскость.) Покажите, что аналогично можно упорядочить все 6-битовые числа в виде тора 8×8 так, что изменяется только одна координата при перемещении на север, юг, восток или запад из любой точки.

▸ **18.** [*20*] *Вес Ли* вектора $u = (u_1, \ldots, u_n)$, где каждая компонента удовлетворяет условию $0 \le u_j < m_j$, определяется следующим соотношением:

$$\nu_L(u) = \sum_{j=1}^{n} \min(u_j, m_j - u_j),$$

а *расстояние Ли* между двумя такими векторами u и v определяется соотношением:

$$d_L(u,v) = \nu_L(u-v), \qquad \text{где } u - v = ((u_1 - v_1) \bmod m_1, \ldots, (u_n - v_n) \bmod m_n).$$

(Это минимальное количество шагов, которое необходимо для изменения u на v, если подгонять некоторые компоненты u_j на ± 1 (по модулю m_j) на каждом этапе.)

Четверичный вектор имеет $m_j = 4$ для $1 \le j \le n$, а двоичный вектор имеет все $m_j = 2$. Найдите простое взаимно-однозначное соответствие между четверичными векторами $u = (u_1, \ldots, u_n)$ и двоичными векторами $u' = (u'_1, \ldots, u'_{2n})$, которые обладают свойствами $\nu_L(u) = \nu(u')$ и $d_L(u,v) = \nu(u' \oplus v')$.

19. [*21*] (*Октакод.*) Пусть $g(x) = x^3 + 2x^2 + x - 1$.

a) Используйте один из алгоритмов этого раздела для оценки суммы

$$\sum z_{u_0} z_{u_1} z_{u_2} z_{u_3} z_{u_4} z_{u_5} z_{u_6} z_{u_\infty}$$

по всем 256 многочленам

$$(v_0 + v_1 x + v_2 x^2 + v_3 x^3) g(x) \bmod 4 = u_0 + u_1 x + u_2 x^2 + u_3 x^3 + u_4 x^4 + u_5 x^5 + u_6 x^6$$

для $0 \le v_0, v_1, v_2, v_3 < 4$, где u_∞ выбирается так, что $0 \le u_\infty < 4$ и $(u_0 + u_1 + u_2 + u_3 + u_4 + u_5 + u_6 + u_\infty) \bmod 4 = 0$.

b) Создайте множество из 256 16-битовых чисел, которые отличаются друг от друга по крайней мере шестью разными положениями битов. (Первое такое множество было открыто Нордстромом и Робинсоном в [*Information и Control* **11** (1967), p.613–616] и является действительно уникальным.)

20. [*M36*] 16-битовые комбинации кода (кодовые слова) в предыдущем упр. можно использовать для передачи 8 битов информации с возможностями исправления ошибок передачи данных в одном или двух поврежденных битах. Более того, ошибки будут обнаружены (но необязательно исправлены), если любые три бита будут получены в искаженном состоянии. Предложите алгоритм, который либо находит ближайшее кодовое слово для заданного 16-битового числа u', либо определяет, что по крайней мере три бита u' являются ошибочными. Как ваш алгоритм раскодирует число $(1100100100001111)_2$? [*Подсказка.* Используйте то, что $x^7 \equiv 1$ (по модулю $g(x)$ и 4) и что каждый четверичный многочлен степени меньше 3 сравним с $x^j + 2x^k$ (по модулю $g(x)$ и 4) для некоторых j, $k \in \{0, 1, 2, 3, 4, 5, 6, \infty\}$, где $x^\infty = 0$.]

21. [*M30*] t-подкуб n-куба можно представить в виде строки $**10**0*$, содержащей t звездочек и $n-t$ указанных битов. Если все 2^n двоичные n-кортежи записаны в лексикографическом порядке, то элемент, относящийся к такому подкубу, появится в $2^{t'}$ кластерах последовательных элементов, где t' — это количество звездочек, которые находятся слева от самого правого указанного бита. (В данном примере $n=8$, $t=5$ и $t'=4$.) Но если эти n-кортежи записаны в двоичном порядке Грэя, то количество кластеров можно сократить. Например, $(n-1)$-подкубы $*\ldots*0$ и $*\ldots*1$ возникают только в $2^{n-2}+1$ и 2^{n-2} кластерах, соответственно, когда используется двоичный порядок Грэя, а не в 2^{n-1} из них.

a) Объясните, как можно было бы вычислить $C(\alpha)$, количество двоичных кластеров Грэя для подкуба, определенного строкой α из звездочек, 0s и 1. Чему равно $C(**10**0*)$?

b) Докажите, что $C(\alpha)$ всегда лежит между $2^{t'-1}$ и $2^{t'}$ включительно.

c) Чему равно среднее значение $C(\alpha)$ по всем $2^{n-t}\binom{n}{t}$ возможным t-подкубам?

▸ **22.** [*22*] Правильным называется такой подкуб, в котором все звездочки располагаются после указанных цифр, как, например, $0110**$. Любой двоичный луч (trie) (см. раздел 6.3) можно рассматривать как способ разбиения куба на непересекающиеся правильные подкубы (рис. 16,*a*). Если поменять местами левые и правые подлучи каждого правого подлуча, проходя от корня сверху вниз, то мы получим *двоичный луч Грэя* (см. рис. 16,*b*).

Докажите, что если листья двоичного луча Грэя обходят в порядке слева направо, то последовательные листья соответствуют смежным подкубам. (Подкубы называются смежными, если они содержат смежные вершины. Например, $00**$ является смежным для $011*$, поскольку первый содержит 0010, а второй — 0110. Но $011*$ не является смежным для $10**$.)

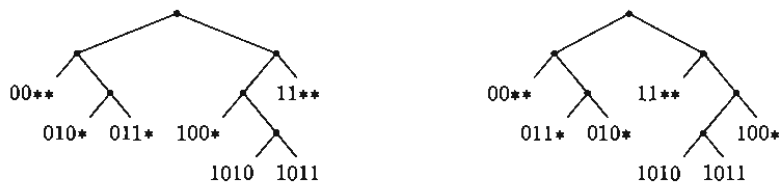


Рис. 16. (a) Нормальный двоичный луч, (b) двоичный луч Грэя

23. [*20*] Допустим, что $g(k)\oplus 2^j = g(l)$. Предложите простой способ поиска l для заданных j и k.

24. [*M21*] Рассмотрим расширение двоичной функции Грэя g на все 2-адические целые числа (см. упр. 4.1–31). Какой будет соответствующая обратная функция $g^{(-1)}$?

▸ **25.** [*M25*] Докажите, что если $g(k)$ и $g(l)$ отличаются $t>0$ битами и если $0\le k,l<2^n$, то $\lceil 2^t/3\rceil \le |k-l| \le 2^n - \lceil 2^t/3\rceil$.

26. [*25*] (Фрэнк Раски.) Для каких целых чисел N можно генерировать все неотрицательные целые числа меньше N таким образом, чтобы на каждом этапе изменялся только один бит двоичного представления?

▸ **27.** [*20*] Пусть $S_0=\{1\}$ и $S_{n+1} = 1/(2+S_n) \cup 1/(2-S_n)$; таким образом, например,

$$S_2 = \left\{ \frac{1}{2+\frac{1}{2+1}}, \frac{1}{2+\frac{1}{2-1}}, \frac{1}{2-\frac{1}{2+1}}, \frac{1}{2-\frac{1}{2-1}} \right\} = \left\{ \frac{3}{7}, \frac{1}{3}, \frac{3}{5}, 1 \right\}$$

и S_n имеет 2^n элементов, которые лежат между $\frac{1}{3}$ и 1. Вычислите 10^{10}-й наименьший элемент S_{100}.

28. [*M27*] *Медианой* n-битовых строк $\{\alpha_1,\ldots,\alpha_t\}$, где α_k имеет двоичное представление $\alpha_k = a_{k(n-1)}\ldots a_{k0}$, является строка $\hat{\alpha} = a_{n-1}\ldots a_0$, биты a_j которых для $0 \le j < n$ согласуются с большинством битов a_{kj} для $1 \le k \le t$. (Если t четно и биты α_{kj} наполовину равны 0 и наполовину равны 1, то медианный бит a_j может быть либо 0, либо 1.) Например, строки $\{0010, 0100, 0101, 1110\}$ имеют две медианы, 0100 и 0110, которые можно обозначить как 01*0.

a) Найдите простой способ описания медиан $G_t = \{g(0),\ldots,g(t-1)\}$ первых двоичных строк Грэя t, когда $0 < t \le 2^n$.

b) Докажите, что если $\alpha = a_{n-1}\ldots a_0$ является такой медианой и если $2^{n-1} < t < 2^n$, то строка β, полученная из α дополнением произвольного бита a_j, также является элементом G_t.

29. [*M24*] Если целочислеиные значеиия k передаются как n-битовые двоичные коды Грэя $g(k)$ и получаются с ошибками, описывемыми битовой структурой $p = (p_{n-1}\ldots p_0)_2$, то средняя численная ошибка равна

$$\frac{1}{2^n}\sum_{k=0}^{2^n-1}\Big|(g^{[-1]}(k)\oplus p) - k\Big|,$$

при условии, что равновероятны все значения k. Покажите, что эта сумма равиа

$$\sum_{k=0}^{2^n-1}|(k\oplus p)-k|/2^n,$$

как если бы не использовался двоичный код Грэя, и оцените ее явио.

▸ **30.** [*M27*] (*Перестановка Грэя.*) Предложите одиопроходной алгоритм для замеиы элементов массива $(X_0, X_1, X_2, \ldots, X_{2^n-1})$ $(X_{g(0)}, X_{g(1)}, X_{g(2)}, \ldots, X_{g(2^n-1)})$, используя вспомогательиое хранилище только постоянного объема. *Подсказка: рассматривая функцию $g(n)$ как перестановку всех неотрицательных целых чисел, покажите, что множество*

$$L = \{0, 1, (10)_2, (100)_2, (100*)_2, (100*0)_2, (100*0*)_2, \ldots\}$$

является множеством лидеров цикла (наименьших элементов циклов).

31. [*HM35*] (*Поле Грэя.*) Пусть $f_n(x) = g(r_n(x))$ обозначает операцию обращения битов в n-битовой двоичной строке, как показано в упр. 5, с последующим преобразованием в двоичный код Грэя. Например, операция $f_3(x)$ имеет вид $(001)_2 \mapsto (110)_2 \mapsto (010)_2 \mapsto (011)_2 \mapsto (101)_2 \mapsto (111)_2 \mapsto (100)_2 \mapsto (001)_2$, поэтому все ненулевые возможности предстают в одном цикле. Следовательно, можно использовать f_3 для определения поля из 8 элементов с $\oplus$ в качестве оператора сложения и с умножением по правилу

$$f_3^{[j]}(1) \times f_3^{[k]}(1) = f_3^{[j+k]}(1) = f_3^{[j]}(f_3^{[k]}(1)).$$

Функции f_2, f_5 и f_6 имеют такие же прекрасные свойства, а функция f_4 — нет, поскольку $f_4((1011)_2) = (1011)_2$.

Найдите все $n \le 100$, для которых f_n определяет поле 2^n элементов.

32. [*M20*] Истинно или ложно утверждение: функции Уолша удовлетворяют соотношению $w_k(-x) = (-1)^k w_k(x)$.

▸ **33.** [*M20*] Докажите закон Радемахера–Уолша (17).

34. [*M21*] *Функция Пэйли* $p_k(x)$ определяется следующими соотношениями:

$$p_0(x) = 1 \qquad \text{и} \qquad p_k(x) = (-1)^{\lfloor 2x\rfloor k} p_{\lfloor k/2\rfloor}(2x).$$

Покажите, что $p_k(x)$ имеет простое выражение на основе функции Пэйли, аналогичной (17), и свяжите фуикции Пэйли с функциями Уолша.

35. [*HM23*] Матрица Пэйли P_n с размерами $2^n \times 2^n$ получается на основе функции Пэйли так же, как матрица Уолша W_n получается на основе функции Уолша (см. (20)). Найдите соотношения между P_n, W_n и матрицей Адамара H_n. Докажите, что все три матрицы являются симметричными.

36. [*21*] Подробно опишите эффективный алгоритм вычисления преобразования Уолша $(x_0, \ldots, x_{2^n-1})$ для заданного вектора $(X_0, \ldots, X_{2^n-1})$.

37. [*HM23*] Пусть z_{kl} обозначает положение l-го изменения знака в $w_k(x)$ для $1 \le l \le k$ и $0 < z_{kl} < 1$. Докажите, что $|z_{kl} - l/(k+1)| = O((\log k)/k)$.

▸ **38.** [*M25*] Придумайте третичное обобщение функции Уолша.

▸ **39.** [*HM30*] (Дж. Дж. Сильвестер.) Строки матрицы $\left(\begin{smallmatrix} a & b \\ b & -a \end{smallmatrix}\right)$ являются ортогональными друг другу и имеют одинаковую величину. Следовательно, матричное тождество

$$\begin{aligned}(A\ B)\begin{pmatrix} a^2+b^2 & 0 \\ 0 & a^2+b^2 \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix} &= (A\ B)\begin{pmatrix} a & b \\ b & -a \end{pmatrix}\begin{pmatrix} a & b \\ b & -a \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix} \\ &= (\,Aa+Bb\ \ Ab-Ba\,)\begin{pmatrix} aA+bB \\ bA-aB \end{pmatrix}\end{aligned}$$

подразумевает тождество суммы двух квадратов $(a^2+b^2)(A^2+B^2) = (aA+bB)^2+(bA-aB)^2$. Аналогично, матрица

$$\begin{pmatrix} a & b & c & d \\ b & -a & d & -c \\ d & c & -b & -a \\ c & -d & -a & b \end{pmatrix}$$

приводит к тождеству суммы четырех квадратов:

$$\begin{aligned}(a^2+b^2+c^2+d^2)(A^2+B^2+C^2+D^2) &= (aA+bB+cC+dD)^2+(bA-aB+dC-cD)^2 \\ &\quad +(dA+cB-bC-aD)^2+(cA-dB-aC+bD)^2.\end{aligned}$$

a) Присвойте знаки матрицы H_3 в (21) символам $\{a, b, c, d, \mathrm{e}, f, g, h\}$, получая матрицу с ортогональными строками, и тождеству суммы восьми квадратов.

b) Обобщите этот принцип до H_4 и матриц более высокого порядка.

▸ **40.** [*21*] Будет ли схема вычисления на основе пятибуквенных слов давать правильный ответ, если маски на этапе W2 были вычислены в виде $m_j = x\&(2^{5j}-1)$ для $0 \le j < 5$?

41. [*25*] Ограничим задачу с пятибуквенными словами только наиболее распространенными 3000 словами, таким образом, исключая `ducky`, `duces`, `dunks`, `dinks`, `dinky`, `dices`, `dicey`, `dicky`, `dicks`, `picky`, `pinky`, `punky` и `pucks` из (23). Сколько корректных слов можно будет генерировать на основе одной пары?

42. [*35*] (М. Л. Фредман.) В алгоритме L используется $\Theta(n \log n)$ бит вспомогательной памяти для указателей фокуса, используемых для определения того двоичного бита Грэя a_j, который должен быть дополнен следующим. На каждом этапе L3 алгоритм проверяет $\Theta(\log n)$ вспомогательных битов и изменяет $\Omega(\log n)$ из них.

Покажите, что с теоретической точки зрения можно было бы достичь большего: n-битовый двоичный код Грэя можно было бы генерировать, изменяя самое большее два вспомогательных бита между посещениями. (Все еще предполагается, что $O(\log n)$ вспомогательных битов проверяется на каждом этапе так, чтобы было ясно, какие их них следовало бы изменить.)

43. [*47*] Определите $d(6)$, т.е. количество 6-битовых циклов Грэя.

44. [*M37*] Покажите, что произвольные дельта-последовательности для цикла Грэя на $n-1$ или $n-2$ битах могут использоваться для создания большого числа дельта-последовательностей для n-битовых циклов Грэя со следующими свойствами: (a) строго одно или (b) строго два имени координат встречаются только дважды.

45. [*M25*] Докажите, что последовательность $d(n)$ характеризуется дважды экспоненциальным ростом: существует такая константа $A > 1$, что $d(n) = \Omega(A^{2^n})$.

46. [*HM48*] Определите асимптотическое поведение $d(n)^{1/2^n}$ при $n \to \infty$.

47. [*M46*] (Сильверман, Виккерс и Сэмпсон.) Пусть $S_k = \{g(0), \ldots, g(k-1)\}$ образуют первые k элементов стандартного двоичного кода Грэя, а $H(k, v)$ — это количество путей Гамильтона в S_k, которые начинаются с 0 и заканчиваются v. Докажите или опровергните: $H(k, v) \le H(k, g(k-1))$ для всех $v \in S_k$, которые будут смежными для $g(k)$.

48. [*36*] Докажите, что $d(n) \le 4(n/2)^{2^n}$, если предположение из предыдущего упражнения истинно. [*Подсказка: пусть $d(n,k)$ — это количество n-битовых циклов Грэя, которые начинаются с $g(0)\ldots g(k-1)$.* Это предположение подразумевает, что $d(n) \le c_{n1}\ldots c_{n(k-1)}d(n,k)$, где c_{nk} — количество вершин, смежных с $g(k-1)$ в n-кубе, но не в S_k.]

49. [*20*] Докажите, что для всех $n \ge 1$ существует $2n$-битовый цикл Грэя, в котором $v_{k+2^{2n-1}}$ является дополнением v_k для всех $k \ge 0$.

50. [*21*] Найдите конструкцию, подобную той, которая используется в теореме D, но с четным l.

51. [*M24*] Завершите доказательство следствия B из теоремы D.

52. [*M20*] Докажите, что если числа логических переходов n-битового цикла Грэя удовлетворяют условию $c_0 \le c_1 \le \cdots \le c_{n-1}$, то $c_0 + \cdots + c_{j-1} \ge 2^j$, а равенство достигается при $j = n$.

53. [*M46*] Если числа $(c_0, \ldots, c_{n-1})$ четные и удовлетворяют условию из предыдущего упражнения, то всегда ли существует n-битовый цикл Грэя с такими числами логических переходов?

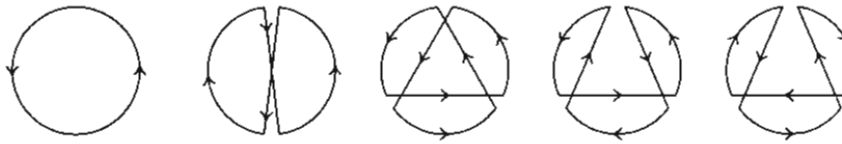
54. [*M20*] (Х. С. Шапиро, 1953.) Покажите, что если последовательность целых чисел $(a_1, \ldots, a_{2^n})$ содержит только n разных значений, то существует подпоследовательность, произведение $a_{k+1}a_{k+2}\ldots a_l$ которой является квадратом простого числа для некоторого $0 \le k < l \le 2^n$. Однако это утверждение может быть неверным, если отменить случай $l = 2^n$.

55. [*47*] (Ф. Раски и К. Сэвидж, 1993.) Если $(v_0, \ldots, v_{2^n-1})$ является n-битовым циклом Грэя, то пары $\{\{v_{2k}, v_{2k+1}\} \mid 0 \le k < 2^{n-1}\}$ образуют совершенное сочетание между вершинами с четностью и нечетностью в n-кубе. Каждое ли такое совершенное сочетание возникает как "половина" некоторого n-битового цикла Грэя?

56. [*M30*] (Э. Н. Гильберт, 1958.) Назовем два цикла Грэя эквивалентными, если их дельта-последовательности можно сделать равными за счет перестановки имен координат либо за счет обращения цикла и/или начала цикла в другом месте. Покажите, что 2688 разных 4-битовых циклов Грэя попадают только в 9 классов эквивалентности.

57. [*32*] Рассмотрим граф, вершины которого являются 2688 возможными 4-битовыми циклами Грэя, где два таких цикла являются смежными, если они связаны одним из

следующих простых преобразований:



(Изменение типа 1 возникает, когда цикл можно разбить на две части с обращением одной части. Типы 2–4 возникают, когда цикл можно разбить на три части с обращением 0, 1 или 2 частей. Части могут быть разного размера. Такие преобразования часто выполняются с циклами Гамильтона.)

Создайте программу, способную определить 4-битовые циклы Грэя, которые можно трансформировать друг в друга, на основе поиска связанных компонентов графа. Ограничьтесь только одним из четырех типов преобразования.

58. [*21*] Пусть α является дельта-последовательностью n-битового цикла Грэя, а β получается из α за счет изменения q появлений от 0 до n, где q является нечетным. Докажите, что $\beta\beta$ является дельта-последовательностью $(n+1)$-битового цикла Грэя.

59. [*22*] 5-битовый цикл Грэя из (30) является *нелокальным* в том смысле, что никакие 2^t последовательных элемента не относятся к одному t-подкубу для $1 < t < n$. Докажите, что нелокальный n-битовый цикл Грэя существует для всех $n \geq 5$. [*Подсказка.* См. предыдущее упражнение.]

60. [*20*] Покажите, что функция с ограниченнем длины прохода удовлетворяет условию $\tau(n+1) \geq \tau(n)$.

61. [*M30*] Покажите, что $\tau(m+n) \geq \tau(m) + \tau(n) - 1$, если (a) $m = 2$ и $2 < \tau(n) < 8$ или если (b) $m \leq n$ и $\tau(n) \leq 2^{m-3}$.

62. [*46*] Верно ли, что $\tau(8) = 6$?

63. [*30*] (Л. Годин.) Докажите, что $\tau(10) \geq 8$.

64. [*HM35*] (Л. Годин и П. Гвоздяк.) n-битовым *потоком Грэя* называют последовательность перестановок $(\sigma_0, \sigma_1, \ldots, \sigma_{l-1})$, где каждое σ_k является перестановкой вершин n-куба, перемещающей каждую вершину к одному из ее соседей.

a) Предположим, что $(u_0, \ldots, u_{2^m-1})$ является m-бнтовым циклом Грэя и $(\sigma_0, \sigma_1, \ldots, \sigma_{2^m-1})$ является n-битовым потоком Грэя. Пусть $v_0 = 0 \ldots 0$ н $v_{k+1} = v_k\sigma_k$, где $\sigma_k = \sigma_{k \bmod 2^m}$, если $k \geq 2^m$. При каких условиях последовательность

$$W = (u_0v_0,\ u_0v_1,\ u_1v_1,\ u_1v_2,\ \ldots,\ u_{2^{m+n-1}-1}v_{2^{m+n-1}-1},\ u_{2^{m+n-1}-1}v_{2^{m+n-1}})$$

является $(m+n)$-битовым циклом Грэя?

b) Покажите, что если m является достаточно большим, то существует n-битовый поток Грэя, удовлетворяющий условиям (a), для которого все длины серий из последовательности $(v_0, v_1, \ldots)$ удовлетворяют условию $\geq n-2$.

c) Примените эти результаты для доказательства того факта, что $\tau(n) \geq n - O(\log n)$.

65. [*30*] (Вретт Стивенс.) Пьеса Сэмюэля Беккета *Четверка (Quad)* начинается и заканчивается пустой сценой. n актеров появляются и выходят за один раз, пробегая все 2^n возможных подмножества, а актер, который покидает сцену, всегда является именно тем актером, который появился раньше других присутствующих на сцене актеров. Если $n = 4$, как в этой пьесе, то некоторые подмножества обязательно повторяются. Покажнте, однако, что существует совершенная структура точно с 2^n входами и выходами прн $n = 5$.

66. [*40*] Существует ли совершеиная структура Беккета–Грэя для восьми актеров?

67. [*20*] Иногда нужно пройти все n-битовые двоичные строки, изменяя *столько* битов, сколько возможно при переходе от одного этапа к следующему этапу, например, при проверке стабильности работы физической цепи в наихудших условиях. Объясните, как обойти все двоичные n-кортежи таким образом, чтобы на каждом этапе изменялось n или $n-1$ битов попеременно.

68. [*21*] Руфус К. Перверзе (R. Q. Perverse) предложил создать троичный код *анти-Грэя*, в котором каждое n-тритовое число отличается от своих соседей в положении *каждой* цифры. Возможен ли такой код для всех n?*

69. [*M25*] Измените определение двоичного кода Грэя (7), допуская, что

$$h(k) = (\ldots(b_6 \oplus b_5)(b_5 \oplus b_4)(b_4 \oplus b_3 \oplus b_2 \oplus b_0)(b_3 \oplus b_0)(b_2 \oplus b_1 \oplus b_0)b_1)_2,$$

когда $k = (\ldots b_5 b_4 b_3 b_2 b_1 b_0)_2$.

a) Покажите, что последовательность $h(0)$, $h(1)$, $\ldots$, $h(2^n-1)$ пробегает через все n-битовые числа таким образом, чтобы каждый раз изменялись точно 3 бита, когда $n > 3$.

b) Обобщите это правило для получения последовательностей, в которых на каждом этапе изменялись точно t бита, когда t нечетно и $n > t$.

70. [*21*] Сколько монотонных n-битовых кодов Грэя существует для $n = 5$ и $n = 6$?

71. [*M22*] Выведите (45), т.е. рекуррентное соотношение, которое определяет перестановки Сэвидж–Винклера.

72. [*20*] Как выглядит код Сэвидж–Винклера для перехода от 00000 к 11111?

73. [*32*] Предложите эффективный алгоритм для создания дельта-последовательности n-битового монотонного кода Грэя.

74. [*M25*] (Сэвидж и Винклер.) Насколько далеко друг от друга могут находиться смежные вершины n-куба в монотонном коде Грэя?

75. [*32*] Найдите все 5-битовые пути Грэя v_0, $\ldots$, v_{31} *без тренда*, в том смысле, что $\sum_{k=0}^{31} k(-1)^{v_{kj}} = 0$ в каждой координате j.

76. [*M25*] Докажите, что n-битовый код Грэя без тренда существует для всех $n \geq 5$.

77. [*21*] Измените алгоритм H так, чтобы можно было обойти n-кортежи со смешанным основанием в *модулярном* порядке Грэя.

78. [*M26*] Докажите формулы преобразования (50) и (51) для рефлексного кода Грэя со смешанным осиованием и выведите аналогичные формулы для модулярного случая.

79. [*M22*] Когда последний n-кортеж (a) отражеиного (рефлексивного) (b) модулярного кода Грэя со смешанным основанием является смежным для первого?

80. [*M20*] Объясните, как пройти все делители числа, при условии, что дано его разложение на простые множители $p_1^{e_1} \ldots p_t^{e_t}$, повторно умножая или деля иа одно простое число на каждом этапе.

81. [*M21*] Пусть (a_0, b_0), (a_1, b_1), $\ldots$, (a_{m^2-1}, b_{m^2-1}) является 2-значным m-м модулярным кодом Грэя. Покажнте, что если $m > 2$, то каждое ребро (x, y) — $(x, (y+1) \bmod m)$ и (x, y) — $((x+1) \bmod m, y)$ прнсутствует в одном из двух циклов:

$$(a_0, b_0) — (a_1, b_1) — \cdots — (a_{m^2-1}, b_{m^2-1}) — (a_0, b_0),$$
$$(b_0, a_0) — (b_1, a_1) — \cdots — (b_{m^2-1}, a_{m^2-1}) — (b_0, a_0).$$

---

* Здесь термин "тритовый" для троичных чисел образоваи по аналогии с термином "битовый" для двоичных чисел. — *Примеч. ред.*

▸ **82.** [*M25*] (Дж. Рингель, 1956.) Используйте предыдущее упр. для вывода того факта, что существует четыре 8-битовых цикла Грэя, таких, что вместе они покрывают все ребра 8-куба.

83. [*41*] Могут ли четыре *сбалансированных* 8-битовых цикла Грэя покрыть все ребра 8-куба?

▸ **84.** [*25*] (Ховард Л. Дыкман.) На рис. 17 показана замечательная головоломка "сумасшедшая петля" (или "гордиев узел"), в которой нужно снять гибкую веревку с жестких петель, которые окружают ее. Покажите, что решение этой головоломки тесно связано с отраженным третичным кодом Грэя.

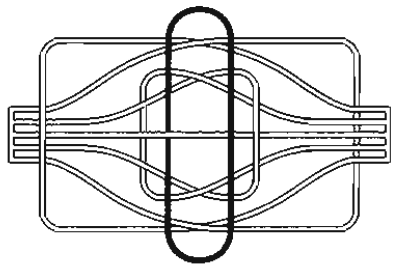


Рис. 17. Головоломка "сумасшедшая петля"

▸ **85.** [*M25*] (Дана Ричардс.) Если $\Gamma = (\alpha_0, \ldots, \alpha_{t-1})$ является произвольной последовательностью t строк и $\Gamma' = (\alpha'_0, \ldots, \alpha'_{t'-1})$ является произвольной последовательностью t' строк, то *зигзаговым произведением (boustrophedon product)* $\Gamma \wr \Gamma'$ называется последовательность из tt' строк, которая начинается с

$$(\alpha_0\alpha'_0, \ldots, \alpha_0\alpha'_{t'-1}, \alpha_1\alpha'_{t'-1}, \ldots, \alpha_1\alpha'_0, \alpha_2\alpha'_0, \ldots, \alpha_2\alpha'_{t'-1}, \alpha_3\alpha'_{t'-1}, \ldots)$$

и заканчивается $\alpha_{t-1}\alpha'_0$, если t четно, либо $\alpha_{t-1}\alpha'_{t'-1}$, если t иечетно. Например, основное определение двоичного кода Грэя в (5) можно выразить с помощью данного обозначения следующим образом: $\Gamma_n = (0,1) \wr \Gamma_{n-1}$ для $n > 0$. Докажите, что операция $\wr$ ассоциативна, т.е. $\Gamma_{m+n} = \Gamma_m \wr \Gamma_n$.

▸ **86.** [*26*] Определите неограниченный код Грэя, который проходит все возможные неотрицательные целочисленные n-кортежи $(a_1, \ldots, a_n)$ таким образом, что $\max(a_1, \ldots, a_n) \leq \max(a'_1, \ldots, a'_n)$, когда за $(a_1, \ldots, a_n)$ следует $(a'_1, \ldots, a'_n)$.

87. [*27*] В продолжение предыдущего упражнения определите неограниченный код Грэя, который проходит *все* целочисленные n-кортежи $(a_1, \ldots, a_n)$ таким образом, что $\max(|a_1|, \ldots, |a_n|) \leq \max(|a'_1|, \ldots, |a'_n|)$, когда за $(a_1, \ldots, a_n)$ следует $(a'_1, \ldots, a'_n)$.

▸ **88.** [*25*] Что произойдет после завершения алгоритма К на этапе К4, если сразу же перезапустить его на этапе К2?

▸ **89.** [*25*] (*Код Грэя для кода Морзе.*) Слова кода Морзе длины n (упр. 4.5.3–32) представляют собой строки из точек и тире, где n — это количество точек плюс удвоенное количество тире.

a) Покажите, что можно генерировать все слова кода Морзе длины n за счет последовательного изменения тире на две точки или наоборот. Например, путь для $n = 3$ будет иметь вид $\bullet\!-$, $\bullet\bullet\bullet$, $-\!\bullet$ или наоборот.

b) Какая строка идет за $\bullet\!-\!-\!-\bullet\bullet\!-\!-\!-\bullet\!-\!\bullet$ в такой последовательности для $n = 15$?

90. [*26*] Для каких n можно упорядочить слова кода Морзе в *цикле* для основных правил упр. 89? [*Подсказка: количество слов кода равно* F_{n+1}.]

▸ **91.** [*34*] Предложите бесцикловый алгоритм посещения всех таких двоичных n-кортежей $(a_1, \ldots, a_n)$, что $a_1 \leq a_2 \geq a_3 \leq a_4 \geq \cdots$. [Число таких n-кортежей равно F_{n+2}.]

92. [*M30*] Существует ли неограниченная последовательность Φ_n, первый элемент которой m^n образует m-й цикл де Бруийна для всех m? [Случай $n = 2$ решен в (54).]

▸ **93.** [*M28*] Докажите, что алгоритм R выводит цикл де Бруийна.

94. [*22*] Каким будет результат алгоритма D для $m = 5$, $n = 1$ и $r = 3$, если сопрограммы $f()$ и $f'()$ генерируют тривиальные циклы 01234 01234 01...?

▸ **95.** [*M23*] Представим, что неограниченная последовательность $a_0a_1a_2\ldots$ с периодом p перемежается с неограниченной последовательностью $b_0b_1b_2\ldots$ с периодом q с образованием неограниченной циклической последовательности

$$c_0c_1c_2c_3c_4c_5\ldots = a_0b_0a_1b_1a_2b_2\ldots.$$

a) При каких условиях $c_0c_1c_2\ldots$ имеет период pq? (Периодом последовательности $a_0a_1a_2\ldots$ в данном упражнении называется такое наименьшее целое число $p > 0$, для которого $a_k = a_{k+p}$ для всех $k \geq 0$.)

b) Какие $2n$-кортежи будут возникать в последовательных выходах алгоритма D, если этап D6 заменить следующим: "если $t' = n$ и $x' < r$, то перейти к D4"?

c) Докажите, что алгоритм D выводит цикл де Бруийна.

▸ **96.** [*M23*] Предположим, что имеется семейство сопрограмм для генерации циклов де Бруийна длины m^n с использованием алгоритмов R и D, основанных на рекурсии простых сопрограмм для осиовного случая $n = 2$, как в алгоритме S.

a) Сколько может быть сопрограмм каждого типа R_n, D_n и S_n?

b) Какое максимальное количество активизаций сопрограмм необходимо для получения вывода одной цифры высокого уровня?

97. [*M29*] Цель этого упражнения — проанализировать циклы де Бруийна, созданные с помощью алгоритмов R и D в важном особом случае $m = 2$. Пусть $f_n(k)$ является таким $(k+1)$-м битом 2^n-цикла, что $f_n(k) = 0$ для $0 \leq k < n$. Также допустим, что j_n является таким индексом, что $0 \leq j_n < 2^n$ и $f_n(k) = 1$ для $j_n \leq k < j_n + n$.

a) Запишите циклы $(f_n(0)\ldots f_n(2^n-1))$ для $n = 2$, 3, 4 и 5.

b) Докажите, что для всех четных значений n существует такое число $\delta_n = \pm 1$, что

$$f_{n+1}(k) \equiv \begin{cases} \Sigma f_n(k), & \text{если } 0 < k \leq j_n \text{ или } 2^n + j_n < k \leq 2^{n+1}; \\ 1 + \Sigma f_n(k+\delta_n), & \text{если } j_n < k \leq 2^n + j_n, \end{cases}$$

где сравнение выполняется по модулю 2. (В этой этой формуле Σf обозначает функцию суммирования $\Sigma f(k) = \sum_{j=0}^{k-1} f(j)$.) Следовательно, $j_{n+1} = 2^n - \delta_n$, когда n четно.

c) Пусть $(c_n(0)c_n(1)\ldots c_n(2^{2n}-5))$ является циклом, созданным с помощью упрощенной версии алгоритма D в упражнении 95(b), примененного для $f_n()$. Где в таком цикле возникают $(2n-1)$-кортежи 1^{2n-1} и $(01)^{n-1}0$?

d) Используйте результаты (c) для выражения $f_{2n}(k)$ на основе $f_n()$.

e) Найдите (в иекоторой степени) простую формулу для j_n, как функцию n.

98. [*M34*] В продолжение предыдущего упражнения предложите эффективный алгоритм вычисления $f_n(k)$ для заданных $n \geq 2$ и $k \geq 0$.

▸ **99.** [*M23*] Примените технологию из предыдущего упражиения для проектирования эффективиого алгоритма поиска любой заданной n-битовой строки в цикле $(f_n(0)f_n(1)\ldots f_n(2^n-1))$.

100. [*40*] Является ли цикл де Бруийна из упр. 97 полезным источником псевдослучайных битов при достаточио больших значениях n?

▸ **101.** [*M30*] (*Уникальное разложение строки на простые неувеличивающиеся строки.*)

a) Докажите, что если λ и λ' просты, то $\lambda\lambda'$ является простой, если $\lambda < \lambda'$.

b) Следовательно, каждую строку α можно записать в виде

$$\alpha = \lambda_1\lambda_2\dots\lambda_t, \qquad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_t, \qquad \text{где каждое } \lambda_j \text{ — простое число.}$$

c) На самом деле возможно только одно такое разложение. [*Подсказка.* Покажите, что λ_t должно быть лексикографически наименьшим непустым суффиксом α.]

d) Истинно или ложно утверждение: λ_1 является самым длинным простым префиксом α.

e) Найдите простые строки для следующей строки:

$$31415926535897932384626433832795028841971?$$

102. [*HM28*] Выведите количество m-х простых строк длины n на основании теоремы об однозначном разложении на простые множители в предыдущем упражнении.

103. [*M20*] Используйте (59) для доказательства малой теоремы Ферма о том, что $m^p \equiv m$ (по модулю p).

104. [*i7*] Согласно формуле (60), приблизительно $1/n$ всех n-буквенных слов являются простыми. Сколько из 5757 пятибуквенных слов GraphBase являются простыми строками? Какое из них является наименьшей непростой строкой? А какое является наибольшей простой строкой?

105. [*M31*] Пусть α является предпростой строкой длины n в неограниченном алфавите.

a) Покажите, что если увеличить последнюю букву α, то полученная строка будет простой строкой.

b) Покажите, что если α разложена на простые строки, как в упр. 101, то она является n-расширением λ_1.

c) Более того, α не может быть n-расширением двух разных простых строк.

▸ **106.** [*M30*] С помощью обратной реконструкции алгоритма F создайте алгоритм, который посещает все m-е простые и предпростые строки в порядке *убывания*.

107. [*HM30*] Оцените время выполнения алгоритма F.

108. [*M35*] Пусть $\lambda_1 < \cdots < \lambda_t$ являются m-ми простыми строками, длины которых являются множителями n, и пусть $a_1\dots a_n$ является произвольной m-й строкой. Цель этого упражнения заключается в доказательстве того, что $a_1\dots a_n$ появляется в $\lambda_1\dots\lambda_t\lambda_1\lambda_2$; следовательно, $\lambda_1\dots\lambda_t$ является циклом де Бруийна (поскольку он имеет длину m^n). Для удобства предположим, что $m = 10$ и строки соответствуют десятичным числам; такие же аргументы применимы для произвольного $m \geq 2$.

a) Покажите, что если $a_1\dots a_n = \alpha\beta$ отличается от всех его циклических сдвигов и если $\beta\alpha = \lambda_k$ — простая строка, то $\alpha\beta$ является подстрокой $\lambda_k\lambda_{k+1}$, пока не выполняется условие $\alpha = 9^j$ для некоторого $j \geq 1$.

b) Где появятся $\alpha\beta$ в $\lambda_1\dots\lambda_t$, если $\beta\alpha$ является простой строкой и α состоит из всех девяток? [*Подсказка.* Покажите, что если $a_{n+1-l}\dots a_n = 9^l$ на этапе F2 для некоторого $l > 0$ и если j не является делителем n, то на предыдущем этапе F2 имеем $a_{n-l}\dots a_n = 9^{l+1}$.]

c) Теперь рассмотрим n-кортежи вида $(\alpha\beta)^d$, где $d > 1$ является делителем n и $\beta\alpha = \lambda_k$ является простой строкой.

d) Где появятся 899135, 997879, 913131, 090909, 909090 и 911911 при $n = 6$?

e) Является ли $\lambda_1\dots\lambda_t$ лексикографически наименьшим m-м циклом де Бруийна длины m^n?

109. [*M22*] m-й тор де Бруийна с размерами $m^2 \times m^2$ для окон 2×2 — такая матрица m-х цифр d_{ij}, что каждая из m^4 подматриц

$$\begin{pmatrix} d_{ij} & d_{i(j+1)} \\ d_{(i+1)j} & d_{(i+1)(j+1)} \end{pmatrix}, \qquad 0 \le i, j < m^2$$

отлична от другой, а индексы свертываются по модулю m^2. Таким образом, любая возможная m-я 2×2 подматрица возникает только один раз. Й. Стюарт назвал этот объект m-м *ауротором* [Ian Stewart, *Game, Set, and Math* (Oxford: Blackwell, 1989), Chapter 4]. Например,

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

является двоичным ауротором. Действительно, это единственная такая матрица для $m = 2$, если исключить сдвиг и/или транспозицию.

Рассмотрим бесконечную матрицу D, элемент которой в строке $i = (\ldots a_2 a_1 a_0)_2$ и столбце $j = (\ldots b_2 b_1 b_0)_2$ is $d_{ij} = (\ldots c_2 c_1 c_0)_2$, где

$$c_0 = (a_0 \oplus b_0)(a_1 \oplus b_1) \oplus b_1;$$
$$c_k = (a_{2k} a_0 \oplus b_{2k}) b_0 \oplus (a_{2k+1} a_0 \oplus b_{2k+1})(b_0 \oplus 1) \quad \text{для } k > 0.$$

Покажите, что верхняя левая подматрица $2^{2n} \times 2^{2n}$ матрицы D является 2^n-м ауротором для всех $n \ge 0$.

110. [*M25*] В продолжение предыдущего упражнения создайте m-е ауроторы для всех m.

111. [*20*] Число 100 можно получить 12 способами за счет вставки знаков + и − в последовательность 123456789; например, $100 = 1 + 23 - 4 + 5 + 6 + 78 - 9 = 123 - 45 - 67 + 89 = -1 + 2 - 3 + 4 + 5 + 6 + 78 + 9$.

a) Какое наименьшее положительное целое число нельзя представить таким образом?

b) Рассмотрите также вставку знаков в последовательность из десяти цифр — 9876543210.

▸ **112.** [*25*] В продолжение предыдущего упражнения: как далеко можно зайти, вставляя знаки среди цифр 12345678987654321? Например, $100 = -1234 - 5 - 6 + 7898 - 7 - 6543 - 2 - 1$.

Тин тан дин дан бим бам бом бо —
тан тин дин дан бам бим бо бом —
тин тан дан дин бим бам бом бо —
тан тин дан дин бам бим бо бом —
тан дан тин бам дин бо бим бом —
. . . . Тин тан дин дан бим бам бом бо.

— ДОРОТИ Л. СЭЙЕРС (DOROTHY L. SAYERS),
Девять портных (The Nine Tailors) (1934)

Перестановка десяти десятичных цифр — это просто десятичное число из 10 цифр, в котором все цифры различны. Следовательно, достаточно сгенерировать все числа из 10 цифр и выбрать только те, цифры которых являются различными. Разве не прекрасно, что высокоскоростные вычисления спасают нас от тяжелых размышлений! Достаточно просто запрограммировать $k+1 \to k$ и проверить цифры k на наличие нежелательных совпадений цифр. Причем этот способ дает нам все перестановки в лексикографическом порядке! По зрелом размышлении ... нам не нужно задумываться о чем-то другом.

— Д. Х. ЛЕМЕР (D. H. LEHMER) (1957)

7.2.1.2. Генерация всех перестановок. После n-кортежей следующим наиболее важным элементом в перечне пожеланий практически любого специалиста в области комбинаторной генерации является задача посещения (т.е. перебора) всех *перестановок* заданного множества или мультимножества. Для решения этой задачи придумано много разных способов. Действительно, ведь для несортированного расположения элементов множества опубликовано практически такое же количество алгоритмов, как и для сортированного! В этом разделе рассматриваются наиболее важные генераторы перестановок, начиная со следующего простого и гибкого классического метода.

Алгоритм L (*генерация лексикографической перестановки*). Для заданной последовательности n элементов $a_1 a_2 \ldots a_n$, в исходном состоянии отсортированных так, что

$$a_1 \le a_2 \le \cdots \le a_n, \tag{1}$$

этот алгоритм генерирует все перестановки $\{a_1, a_2, \ldots, a_n\}$, посещая их в лексикографическом порядке. (Например, перестановки множества $\{1, 2, 2, 3\}$ имеют такой лексикографический порядок:

1223, 1232, 1322, 2123, 2132, 2213, 2231, 2312, 2321, 3122, 3212, 3221.)

Предполагается, что для удобства используется вспомогательный элемент a_0, который должен быть меньше, чем самый большой элемент a_n.

L1. [Посетить.] Посетить перестановку $a_1 a_2 \ldots a_n$.

L2. [Найти j.] Установить $j \leftarrow n-1$. Если $a_j \ge a_{j+1}$, уменьшать j на 1 повторно, пока не выполняется условие $a_j < a_{j+1}$. Завершить алгоритм, если $j = 0$. (В этом месте j является наименьшим индексом, для которого были посещены все перестановки, начиная с $a_1 \ldots a_j$. Следовательно, лексикографически следующая перестановка увеличит значение a_j.)

L3. [Увеличить a_j.] Установить $l \leftarrow n$. Если $a_j \ge a_l$, уменьшать l на 1 повторно, пока не выполняется условие $a_j < a_l$. Затем поменять местами $a_j \leftrightarrow a_l$. (Поскольку $a_{j+1} \ge \cdots \ge a_n$, элемент a_l является наименьшим элементом больше a_j, который может следовать за $a_1 \ldots a_{j-1}$ в перестановке. Перед заменой мы имели $a_{j+1} \ge \cdots \ge a_{l-1} \ge a_l > a_j \ge a_{l+1} \ge \cdots \ge a_n$; а после замены имеем $a_{j+1} \ge \cdots \ge a_{l-1} \ge a_j > a_l \ge a_{l+1} \ge \cdots \ge a_n$.)

L4. [Обратить $a_{j+1} \ldots a_n$.] Установить $k \leftarrow j+1$ и $l \leftarrow n$. Затем, если $k < l$, поменять местами $a_k \leftrightarrow a_l$, установить $k \leftarrow k+1$, $l \leftarrow l-1$ и повторять, пока не выполнится условие $k \ge l$. Вернуться к L1. ▮

Этот алгоритм придуман Пандита Нарайана еще в XIV веке в Индии (см. раздел 7.2.1.7). Он также упоминается в предисловии К. Ф. Гинденбурга (C. F. Hindenburg) к работе К. Ф. Рудигера (C. F. Rüdiger) *Specimen Analyticum de Lineis Curvis Secundi Ordinis*, C. F. Rüdiger (Leipzig, 1784), xlvi-xlvii. Он часто переоткрывался с того времени. Принцип его работы описывается в комментариях в скобках на этапах L2 и L3.

Вообще говоря, лексикографического наследника любого комбинаторного объекта $a_1 \ldots a_n$ можно получить с помощью трехэтапной процедуры.

1) Найти такое наибольшее значение j, для которого a_j может быть увеличено.
2) Увеличить a_j на наименьшее допустимое значение.
3) Найти лексикографически наименьший путь для расширения нового объекта $a_1 \ldots a_j$ до полного объекта.

Алгоритм L использует эту общую процедуру в случае генерации перестановки, точно так же, как алгоритм 7.2.1.1M использует ее в случае генерации n-кортежа. При изучении других комбинаторных объектов мы представим другие разнообразные примеры. Обратите внимание на то, что в начале этапа L4 имеем $a_{j+1} \geq \cdots \geq a_n$. Следовательно, первая перестановка, начинаясь с текущего префикса $a_1 \ldots a_j$, имеет вид $a_1 \ldots a_j a_n \ldots a_{j+1}$, а этап L4 создает ее за счет $\lfloor (n-j)/2 \rfloor$ замен (транспозиций).

На практике этап L2 находит $j = n-1$ за вдвое меньшее время, если элементы различны, поскольку точно для $n!/2$ из $n!$ перестановок выполняется $a_{n-1} < a_n$. Следовательно, алгоритм L можно ускорить, распознавая этот особый случай, без значительного усложнения (см. упр. 1). Аналогично, вероятность того, что $j \leq n-t$, равна всего $1/t!$, если a различны. Следовательно, циклы на этапах L2–L4 обычно выполняются очень быстро. В упр. 6 время выполнения анализируется в общем случае и показывается, что алгоритм L обладает сравнительно высокой эффективностью, когда присутствуют равные элементы, если только некоторые значения не появляются чаще других в мультимножестве $\{a_1, a_2, \ldots, a_n\}$.

Смежные замены. В разделе 7.2.1.1 показано, что коды Грэя эффективны для генерации n-кортежей, но аналогичные рассуждения применимы и для генерации перестановок. Простейшее возможное изменение перестановки заключается в замене смежных элементов, а из главы 5 известно, что любая перестановка может быть упорядочена, если подобрать подходящую последовательность таких замен. (Например, алгоритм 5.2.2B работает таким образом.) Следовательно, можно пойти обратным путем и получить любую нужную перестановку, начиная со всех упорядоченных элементов и заменяя соответствующие пары смежных элементов.

Естественным образом возникает вопрос: можно ли получить *все* перестановки заданного мультимножества таким образом, что на каждом этапе происходит замена (транспозиция) только двух смежных элементов? Если да, то общая программа проверки всех перестановок может во многих случаях быть проще и быстрее, поскольку ей потребуется только вычислить результат замены вместо повторной обработки всего нового массива $a_1 \ldots a_n$.

Увы, если мультимножество содержит повторяющиеся элементы, то не всегда можно найти такую последовательность, подобную коду Грэя. Например, шесть перестановок множества $\{1, 1, 2, 2\}$ связаны друг с другом следующим образом за-

менами смежных элементов:

$$1122 \longleftrightarrow 1212 \begin{array}{<} 2112 \\ 1221 \end{array} 2121 \longleftrightarrow 2211; \tag{2}$$

причем этот граф не имеет гамильтонова пути.

В большинстве приложений приходится иметь дело с перестановками *различных* элементов, а в этом случае есть один простой алгоритм, который способен генерировать все $n!$ перестановок за счет всего $n! - 1$ замен смежных элементов. Более того, еще одна такая замена возвращает множество в исходное состояние так, что имеем гамильтонов цикл, аналогичный двоичному коду Грэя.

Идея заключается в том, чтобы взять такую последовательность для $\{1, \ldots, n-1\}$ и вставить число n во все возможные перестановки, выполненные таким образом. Например, если $n = 4$, то последовательность $(123, 132, 312, 321, 231, 213)$ приводит к массиву с такими столбцами:

$$\begin{array}{cccccc} 1234 & 1324 & 3124 & 3214 & 2314 & 2134 \\ 1243 & 1342 & 3142 & 3241 & 2341 & 2143 \\ 1423 & 1432 & 3412 & 3421 & 2431 & 2413 \\ 4123 & 4132 & 4312 & 4321 & 4231 & 4213 \end{array} \tag{3}$$

где число 4 вставлено во все четыре возможные позиции. Теперь мы получим нужную последовательность, считывая вниз первый столбец, потом вверх — второй, вниз — третий, ..., вверх — последний: $(1234, 1243, 1423, 4123, 4132, 1432, 1342, 1324, 3124, 3142, \ldots, 2143, 2134)$.

В разделе 5.1.1 рассматриваются инверсии перестановки, а именно неупорядоченные пары элементов (не обязательно смежных). Каждая замена смежных элементов изменяет общее число инверсий на ± 1. Действительно, если рассмотреть так называемую таблицу инверсии $c_1 \ldots c_n$ из упр. 5.1.1–7, где c_j — это количество элементов, лежащих справа от j, которые меньше, чем j, то можно обнаружить, что перестановки в (3) имеют следующую таблицу инверсии:

$$\begin{array}{cccccc} 0000 & 0010 & 0020 & 0120 & 0110 & 0100 \\ 0001 & 0011 & 0021 & 0121 & 0111 & 0101 \\ 0002 & 0012 & 0022 & 0122 & 0112 & 0102 \\ 0003 & 0013 & 0023 & 0123 & 0113 & 0103 \end{array} \tag{4}$$

А если считать эти столбцы попеременно вниз и вверх, как прежде, то получим точно отраженный код Грэя для смешанного основания $(1, 2, 3, 4)$, как в уравнениях (46)–(51) в разделе 7.2.1.1. Такое же свойство справедливо для всех n, как заметил Э. В. Дейкстра (E. W. Dijkstra) [*Acta Informatica* **6** (1976), p.357–359], и оно приводит нас к следующей формулировке.

Алгоритм P (*простые изменения*). Для заданной последовательности $a_1 a_2 \ldots a_n$ n различных элементов этот алгоритм генерирует все их перестановки за счет повторных замен среди смежных пар. Он использует вспомогательный массив $c_1 c_2 \ldots c_n$, который представляет инверсии, как описано выше, проходя все последовательности целых чисел, таких, что

$$0 \le c_j < j \qquad \text{для } 1 \le j \le n. \tag{5}$$

Другой массив, $o_1 o_2 \ldots o_n$, отвечает за направления, в которых меняются элементы c_j.

P1. [Инициализировать.] Установнть $c_j \leftarrow 0$ и $o_j \leftarrow 1$ для $1 \le j \le n$.

P2. [Посетить.] Посетить перестановку $a_1 a_2 \ldots a_n$.

P3. [Приготовиться к изменению.] Установнть $j \leftarrow n$ и $s \leftarrow 0$. (На следующих этапах определяется координата j, для которой меняется c_j, сохраняя (5); переменная s содержит количество таких индексов $k > j$, что $c_k = k - 1$.)

P4. [Готовность к изменению?] Установить $q \leftarrow c_j + o_j$. Если $q < 0$, перейти к P7; если $q = j$, перейти к P6.

P5. [Изменить.] Заменнть $a_{j-c_j+s} \leftrightarrow a_{j-q+s}$. Затем установить $c_j \leftarrow q$ и вернуться к P2.

P6. [Увеличить s.] Завершить, если $j = 1$, в противном случае установить $s \leftarrow s+1$.

P7. [Переключить направление.] Установить $o_j \leftarrow -o_j$, $j \leftarrow j-1$, а потом вернуться к P4. ∎

Этот алгоритм, который очевидно работает для всех $n \ge 1$, обнаружен в XVII веке в Англии, когда звонари придумали обычай перезвона множества колоколов во всех возможных перестановках. Они назвали алгоритм P методом *простых изменений*. На рис. 18,*a* показана неупорядоченная особая последовательность "Кембридж 48" из 48 перестаиовок 5 колоколов, которая использовалась в начале XVI века еще до того, как согласно принципу простых изменений был обнаружен способ перебора всех $5! = 120$ вариантов. Почтенная история алгоритма P прослежена до рукописи Питера Мунди, которая хранится в Библиотеке имени Бодлея при Оксфордском университете. Она была переписана приблизительно в 1653 году и транскрибирова-иа Эрнестом Моррисом (Ernest Morris) в книге *The History and Art of Change Ringing* (1931), p.29–30. Немного позже, в 1668 году, была аноимно иапечатана знаменитая книга *Tintinnalogia (Тинтинналогия, или Искусство колокольного звона)*, авторами которой (как выяснилось теперь) были Ричард Дакворт (Richard Duckworth) и Фабиан Стедман (Fabian Stedman), посвятившие первые 60 страниц подробному описанию простых изменений, иачиная с $n = 3$ и до произвольно большого n.

> Последовательность "Кембридж 48" *в течение многих лет была величайшим Перезвоном, который когда-либо* звонил *или был изобретен; но теперь, ни* "48", *ни* "100", *ни* "720", *ни любое другое число не может сдержать нас; ведь мы открыли* алгоритм изменения звона колоколов до бесконечности.
> ... *Для четырех колоколов существует* 24 варианта Перезвона, *причем один колокол называется* "Охота", а *три другие* — "Экстремальные *Колокола"; "Охота" сдвигается, и постоянно* охотится *вверх и вниз ...; два из* "Экстремальных *колоколов" образуют* изменение *каждый раз, когда* "Охота" *возникает перед ними или после них.*
>
> — РИЧАРД ДАКВОРТ и ФАБИАН СТЕДМАН,
> *Tintinnalogia (Тинтинналогия, или Искусство колокольного звона)* (1668)

Британские эитузиасты колокольного звоиа вскоре придумали более сложные схемы, в которых две или более пары колоколов одновременио меняются местами. Например, они придумали последовательиость (рис. 18,*c*), известную как "дедовская

сдвоенная", "наилучший и наиболее искусный Перезвон, когда-либо придуманный для пяти колоколов" [*Tintinnalogia*, p. 95]. Такие методы более интересны, чем алгоритм P, с музыкальной или математической точки зрения, но они менее полезны в компьютерных приложениях, а потому мы не будем подробно рассматривать их. Заинтересованные читатели могут ознакомиться с ними в книге В. Г. Вилсона (W. G. Wilson) *Change Ringing* (1965); см. также работу А. Т. Вайта (A. T. White), опубликованную в *AMM* **103** (1996), p.771–778.


(незавершено)

(a) Последовательность "Кембридж 48"

(b) Последовательность "простые изменения"

(c) Последовательность "дедовская сдвоеиная"

(d) Последовательиость "Стедмана сдвоенная"

Рис. 18. Четыре последовательностн, которые использовались в XVII-м веке в Англии для перестановки пяти разных церковных колоколов. Объект (b) соответствует алгоритму P

X. Ф. Троттер (H. F. Trotter) опубликовал первую компьютерную реализацию простых изменений [*CACM* **5** (1962), p.434–435]. Этот алгоритм достаточно эффективный, особенно если он отлажен, как в упр. 16, поскольку $n-1$ из каждых n перестановок генерируются без нспользования этапов P6 и P7. Алгоритм L, наоборот, находится в наилучшем состоянии только в половине случаев.

Тот факт, что алгоритм P выполняет точно одиу замену за одно посещение, означает, что генерируемые им перестановки попеременно четны и нечетны (см. упр. 5.1.1–13). Следовательно, можно генерировать все четные перестановки, пропуская нечетные. Действительно, таблицы c и o упрощают слежение за текущим общим количеством инверсий, $c_1 + \cdots + c_n$.

Во многих программах требуется повторно генерировать одинаковые перестановки н в таких случаях не нужно каждый раз проходить все этапы алгоритма P. Можно просто подготовить список подходящих переходов, используя следующий метод.

Алгоритм T (*переходы простых изменений*). Этот алгоритм вычисляет такую таблицу $t[1], t[2], \ldots, t[n!-1]$, что действия алгоритма P эквивалентны последовательным обменам $a_{t[k]} \leftrightarrow a_{t[k]+1}$ для $1 \le k < n!$. Предполагаем, что $n \ge 2$.

T1. [Инициализировать.] Установить $N \leftarrow n!$, $d \leftarrow N/2$, $t[d] \leftarrow 1$ и $m \leftarrow 2$.

T2. [Цикл для m.] Завершить, если $m = n$. В противном случае установить $m \leftarrow m+1$, $d \leftarrow d/m$ и $k \leftarrow 0$. (Придерживаемся условия $d = n!/m!$.)

T3. [Охотиться вниз.] Установить $k \leftarrow k+d$ и $j \leftarrow m-1$. Пока $j > 0$, установить $t[k] \leftarrow j$, $j \leftarrow j-1$ и $k \leftarrow k+d$, пока не выполняется условие $j = 0$.

T4. [Отступ.] Установить $t[k] \leftarrow t[k]+1$ и $k \leftarrow k+d$.

T5. [Охотиться вверх.] Пока $j < m-1$, установить $j \leftarrow j+1$, $t[k] \leftarrow j$ и $k \leftarrow k+d$. Вернуться к T3, если $k < N$, в противном случае вернуться к T2. ▮

Например, если $n = 4$, то получим таблицу $(t[1], t[2], \ldots, t[23]) = (3,2,1,3,1,2,3,1,3,2,1,3,1,2,3,1,3,2,1,3,1,2,3)$.

Буквометика. Рассмотрим простую загадку, в которой полезно использовать перестановки: как разгадать загадку суммирования зашифрованных чисел

$$\begin{array}{r} \texttt{SEND} \\ +\,\texttt{MORE} \\ \hline \texttt{MONEY}, \end{array} \tag{6}$$

где каждый символ обозначает какую-то десятичную цифру? [H. E. Dudeney, *Strand* **68** (1924), 97, p.214.] Такие загадки часто называют словом "буквометика" ("alphametics"), которое придумал Дж. Э. Х. Хантер [*Globe and Mail* (Toronto: 27 October 1955), 27]. Другой термин, "криптаритм (cryptarithm)", предложил С. Ватриквант [*Sphinx* **1** (May 1931), p.50].

Классическую буквометику (6) можно легко решить вручную (см. упр. 21). Но допустим, имеется большой набор сложных буквометик, причем одни из них могут не иметь решения, а другие могут иметь десятки решений. Тогда для экономии времени можно было бы написать компьютерную программу, которая попробует перебрать все перестановки цифр, соответствующих данной загадке, и найти те перестановки, которые дают правильную сумму. (Компьютерная программа для решения буквометик была предложена Джоном Бейдлером [*Creative Computing* **4**, 6 (November–December 1978), p.110–113].)

Если отойти от частностей и рассмотреть более общий тип задач аддитивных буквометрик, то можно заметить не только простые суммы, как (6), но и другие примеры типа

$$\texttt{VIOLIN} + \texttt{VIOLIN} + \texttt{VIOLA} \;=\; \texttt{TRIO} + \texttt{SONATA}.$$

Эта задача эквивалентна следующей:

$$2(\texttt{VIOLIN}) + \texttt{VIOLA} - \texttt{TRIO} - \texttt{SONATA} \;=\; 0, \tag{7}$$

где фигурирует сумма членов с целочисленными коэффициентами, которая в итоге равна нулю, если вместо различных букв подставить различные десятичные цифры. Каждая буква в такой задаче имеет сигнатуру, которая получается путем подстановки 1 для данной буквы и подстановки 0 для других букв. Например, сигнатура для I в (7) имеет вид

$$2(010010) + 01000 - 0010 - 000000,$$

а именно 21010. Если произвольно присвоить коды $(1,2,\ldots,10)$ буквам (`V, I, O, L, N, A, T, R, S, X`), то соответствующие сигнатуры для (7) имеют вид

$$\begin{aligned} &s_1 = 210000, \quad s_2 = 21010, \quad s_3 = -7901, \quad s_4 = 210, \quad s_5 = -998, \\ &s_6 = -100, \quad s_7 = -1010, \quad s_8 = -100, \quad s_9 = -100000, \quad s_{10} = 0. \end{aligned} \tag{8}$$

(Дополнительная буква `X` добавлена, поскольку нужно десять букв.) Задача теперь сводится к поиску всех таких перестановок $a_1 \ldots a_{10}$ для $\{0,1,\ldots,9\}$, что

$$a \cdot s \;=\; \sum_{j=1}^{10} a_j s_j \;=\; 0. \tag{9}$$

Дополнительное условие заключается в том, что числа в буквометике не могут начинаться с цифры 0. Например, суммы

$$\begin{array}{r} 7316 \\ +\,0823 \\ \hline 08139 \end{array} \quad \text{и} \quad \begin{array}{r} 5731 \\ +\,0647 \\ \hline 06378 \end{array} \quad \text{и} \quad \begin{array}{r} 6524 \\ +\,0735 \\ \hline 07259 \end{array} \quad \text{и} \quad \begin{array}{r} 2817 \\ +\,0368 \\ \hline 03185 \end{array}$$

и многие другие *не* считаются корректными решениями (6). Вообще, существует множество F таких первых букв, что

$$a_j \neq 0 \qquad \text{для всех } j \in F; \tag{10}$$

множество F, соответствующее (7) и (8), имеет вид $\{1,7,9\}$.

Один из способов обработки аддитивных буквометик состоит в использовании алгоритма T для подготовки таблицы $10! - 1$ переходов $t[k]$. Тогда для каждой задачи, заданной последовательностью сигнатур $(s_1,\ldots,s_{10})$ и множеством первых букв F, решение можно искать следующим образом.

A1. [Инициализировать.] Установить $a_1 a_2 \ldots a_{10} \leftarrow 01\ldots9$, $v \leftarrow \sum_{j=1}^{10}(j-1)s_j$, $k \leftarrow 1$ и $\delta_j \leftarrow s_{j+1} - s_j$ для $1 \le j < 10$.

A2. [Проверить.] Если $v = 0$ и если справедливо (10), вывести решение $a_1 \ldots a_{10}$.

A3. [Обмен.] Остановить, если $k = 10!$. В противном случае установить $j \leftarrow t[k]$, $v \leftarrow v - (a_{j+1} - a_j)\delta_j$, $a_{j+1} \leftrightarrow a_j$, $k \leftarrow k+1$ и вернуться к A2. ▮

Этап A3 оправдан тем, что транспозиция a_j и a_{j+1} просто уменьшает $a \cdot s$ на $(a_{j+1} - a_j)(s_{j+1} - s_j)$. Хотя $10!$ равно 3 628 800 и представляет собой достаточно большое число, но операции на этапе A3 так просты, что для их выполнения потребуется всего доля секунды работы современного компьютера.

Буквометика называется *чистой*, если она имеет единственное решение. К сожалению, буквометика (7) не является чистой. Перестановки 1764802539 и 3546281970 удовлетворяют (9) и (10), следовательно, имеем два решения:

$$176478 + 176478 + 17640 \;=\; 2576 + 368020;$$

и

$$354652 + 354652 + 35468 \;=\; 1954 + 742818.$$

Более того, $s_6 = s_8$ в (8), так что можно получить еще два решения за счет обмена цифрами для букв `A` и `R`.

С другой стороны, буквометика (6) *является* чистой, хотя данный метод дает две разные перестановки для ее решения. Дело в том, что (6) включает только

восемь различных букв, следовательно, для поиска решения используются две подстановочные сигнатуры $s_9 = s_{10} = 0$. В общем случае буквометика с m различными буквами имеет $10 - m$ подстановочных сигнатур $s_{m+1} = \cdots = s_{10} = 0$, а каждое решение будет найдено $(10-m)!$ раз, если не использовать условие $a_{m+1} < \cdots < a_{10}$.

Общая структура. Очень много алгоритмов было предложено для генерации перестановок различных объектов, и наилучший способ их изучения состоит в применении мультипликативных свойств перестановок, изученных в разделе 1.3.3. Для этого нужно слегка изменить систему обозначений, используя индексирование с началом в нуле и запись $a_0 a_1 \ldots a_{n-1}$ для перестановок $\{0, 1, \ldots, n-1\}$ вместо записи $a_1 a_2 \ldots a_n$ для перестановок $\{1, 2, \ldots, n\}$. Что еще более важно, так это то, что мы будем использовать схемы генерации перестановок, в которых большинство действий происходит *слева* так, что все перестановки $\{0, 1, \ldots, k-1\}$ генерируются во время первых $k!$ этапов для $1 \le k \le n$. Например, одна такая схема для $n = 4$ имеет вид

$$\begin{aligned}&0123, 1023, 0213, 2013, 1203, 2103, 0132, 1032, 0312, 3012, 1302, 3102,\\ &0231, 2031, 0321, 3021, 2301, 3201, 1230, 2130, 1320, 3120, 2310, 3210;\end{aligned} \tag{11}$$

который называется обратным колексикографическим порядком, поскольку, если отразить строки справа налево, получим 3210, 3201, 3120, ..., 0123, т.е. обратный лексикографический порядок. Другой способ представления (11) состоит в просмотре элементов в виде $(n-a_n)\ldots(n-a_2)(n-a_1)$, где $a_1 a_2 \ldots a_n$ в лексикографическом порядке проходит через перестановки $\{1, 2, \ldots, n\}$.

Напомним из раздела 1.3.3, что перестановку типа $\alpha = 250143$ можно записать либо в форме двух строк:

$$\alpha = \binom{012345}{250143},$$

либо в более компактной циклической форме:

$$\alpha = (0\ 2)(1\ 5\ 3).$$

Это означает, что в α имеют место $0 \mapsto 2$, $1 \mapsto 5$, $2 \mapsto 0$, $3 \mapsto 1$, $4 \mapsto 4$ и $5 \mapsto 3$. Одноэлементный цикл '(4)' указывать не нужно. Поскольку цифра 4 является фиксированной точкой этой перестановки, то можно сказать, что "α фиксирует 4". Запишем $0\alpha = 2$, $1\alpha = 5$ и т.д., т.е. $j\alpha$ является образом j от α. Произведение перестановок, как, например, α и β, где $\beta = 543210$, можно записать либо в двухстрочной форме:

$$\alpha\beta = \binom{012345}{250143}\binom{012345}{543210} = \binom{012345}{250143}\binom{250143}{305412} = \binom{012345}{305412}$$

либо в циклической форме:

$$\alpha\beta = (0\ 2)(1\ 5\ 3)\ \cdot\ (0\ 5)(1\ 4)(2\ 3)\ =\ (0\ 3\ 4\ 1)(2\ 5).$$

Обратите внимание, что образом 1 от $\alpha\beta$ является $1(\alpha\beta) = (1\alpha)\beta = 5\beta = 0$ и т.д. *Предупреждение.* Почти в половине всех книг о перестановках их умножают иначе (справа налево), предполагая, что $\alpha\beta$ означает сначала применение β, а потом — α. Дело в том, что традиционная функциональная запись $\alpha(1) = 5$ вполне естественно наводит на мысль, что $\alpha\beta(1)$ означает $\alpha(\beta(1)) = \alpha(4) = 4$. Однако в данной книге

используется другая философия, и в ней перестановки всегда умножаются слева направо.

Порядок умножения нужно учитывать очень внимательно, когда перестановки представлены массивами чисел. Например, если "применить" отражение $\beta = 543210$ к перестановке $\alpha = 250143$, то результатом 341052 будет не $\alpha\beta$, а $\beta\alpha$. Вообще говоря, операция замены перестановки $\alpha = a_0 a_1 \dots a_{n-1}$ некоторым переупорядоченнем $a_{0\beta} a_{1\beta} \dots a_{(n-1)\beta}$ дает $k \mapsto a_{k\beta} = k\beta\alpha$. Перестановка *положения* с помощью β соответствует предумножению (premultiplication) на β, иначе говоря, *умножению в обратном порядке* или умножению начиная слева, т.е. замене α на $\beta\alpha$; а перестановка *значений* с помощью β соответствует *умножению в обычном порядке* на β, иначе говоря, постумножению (postmultiplication) или умножению начиная справа, т.е. замене α на $\alpha\beta$. Таким образом, например, генератор перестановок, который обменивает $a_1 \leftrightarrow a_2$, предумножает текущую перестановку на (1 2), постумножает ее на $(a_1\ a_2)$.

Следующее предложение сделано Эваристом Галуа (Évariste Galois) в 1830 году: непустое множество G перестановок образует *группу*, если оно замкнуто на операции умножения, т.е. если произведение $\alpha\beta$ принадлежит G, когда α и β являются элементами G [см. *Écrits et Mémoires Mathématiques d'Évariste Galois* (Paris, 1962), p.47]. Рассмотрим 4-куб, представленный в виде тора 4×4

$$\begin{array}{|cccc|}\hline 0 & 1 & 3 & 2 \\ 4 & 5 & 7 & 6 \\ \mathtt{c} & \mathtt{d} & \mathtt{f} & \mathtt{e} \\ 8 & 9 & \mathtt{b} & \mathtt{a} \\ \hline\end{array} \tag{12}$$

как в упр. 7.2.1.1–17, и пусть G является множеством всех перестановок вершин $\{0, 1, \dots, \mathtt{f}\}$, для которых сохраняется соседство: перестановка α принадлежит G тогда и только тогда, когда u — v подразумевает $u\alpha$ — $v\alpha$ в 4-кубе. (Здесь используются шестнадцатеричные цифры $(0, 1, \dots, \mathtt{f})$ для обозначения целых чисел $(0, 1, \dots, 15)$. Метки в (12) выбраны так, что u — v тогда и только тогда, когда u и v отличаются только одним положением бита.) Множество G очевидно является группой, а его элементы называются симметриями или автоморфизмами 4-куба.

Группы перестановок G можно очень эффективно представить в компьютере с помощью *таблицы Симса*, предложенной Чарльзом К. Симсом (Charles C. Sims) [*Computational Methods in Abstract Algebra* (Oxford: Pergamon, 1970), p.169–183]. Она является семейством подмножеств S_1, S_2, ... множества G, обладающих следующим свойством: S_k содержит только одну перестановку σ_{kj}, которая принимает $k \mapsto j$ и фиксирует значения всех элементов больше k, всякий раз, когда G содержит такую перестановку. σ_{kk} является тождественной перестановкой, которая всегда присутствует в G. Если $0 \le j < k$, то любая подходящая перестановка может быть выбрана для того, чтобы сыграть роль σ_{kj}. Основным преимуществом таблицы Симса является ее способность удобного представления целой группы.

Лемма S. *Пусть S_1, S_2, ..., S_{n-1} является таблицей Симса для группы G перестановок $\{0, 1, \dots, n-1\}$. Тогда каждый элемент α группы G имеет единственное представление:*

$$\alpha = \sigma_1 \sigma_2 \dots \sigma_{n-1}, \qquad \text{где } \sigma_k \in S_k \text{ для } 1 \le k < n. \tag{13}$$

Доказательство. Если α имеет такое представление и если σ_{n-1} является перестановкой $\sigma_{(n-1)j} \in S_{n-1}$, то α принимает значение $n-1 \mapsto j$, поскольку все элементы $S_1 \cup \cdots \cup S_{n-2}$ фиксируют зиачение $n-1$. Наоборот, если α прииимает зиачение $n-1 \mapsto j$, то имеем $\alpha = \alpha'\sigma_{(n-1)j}$, где

$$\alpha' = \alpha\sigma^-_{(n-1)j}$$

является перестановкой G, которая фиксирует $n-1$. (Как в разделе 1.3.3, σ^- обозначает иивереию σ.) Множество G' всех таких перестановок является группой, а $S_1, \ldots, S_{n-2}$ — таблицей Симса для G'. Таким образом, результат следует отсюда по индукции n. ∎

Например, вычисления показывают, что одной возможной таблицей Симса для автоморфизма группы 4-куба является

$$\begin{aligned}
S_{\mathtt{f}} &= \{(),\ (01)(23)(45)(67)(89)(\mathtt{ab})(\mathtt{cd})(\mathtt{ef}),\ \ldots,\\
&\qquad\qquad (0\mathtt{f})(1\mathtt{e})(2\mathtt{d})(3\mathtt{c})(4\mathtt{b})(5\mathtt{a})(69)(78)\};\\
S_{\mathtt{e}} &= \{(),\ (12)(56)(9\mathtt{a})(\mathtt{de}),\ (14)(36)(9\mathtt{c})(\mathtt{be}),\ (18)(3\mathtt{a})(5\mathtt{c})(7\mathtt{e})\};\\
S_{\mathtt{d}} &= \{(),\ (24)(35)(\mathtt{ac})(\mathtt{bd}),\ (28)(39)(6\mathtt{c})(7\mathtt{d})\};\\
S_{\mathtt{c}} &= \{()\};\\
S_{\mathtt{b}} &= \{(),\ (48)(59)(6\mathtt{a})(7\mathtt{b})\};\\
S_{\mathtt{a}} &= S_9 = \cdots = S_1 = \{()\}.
\end{aligned} \tag{14}$$

Здесь $S_{\mathtt{f}}$ содержит 16 перестановок $\sigma_{\mathtt{f}j}$ для $0 \le j \le 15$, которые соответственно принимают $i \mapsto i \oplus (15-j)$ для $0 \le i \le 15$. Миожество $S_{\mathtt{e}}$ содержит только четыре перестановки, поскольку автоморфизм, который фиксирует f, должен перемещать e по соседству с f. Таким образом, e будет либо e, либо d, либо b, либо 7. Множество $S_{\mathtt{c}}$ содержит только тождественную перестановку, так как автоморфизм, который фиксирует f, e и d, должеи также фиксировать c. Большинство групп имеют $S_k = \{()\}$ для всех малых значений k, как в этом примере. Следовательно, таблица Симса обычио содержит только небольшое количество перестановок, хотя сама группа может быть очень большой.

Представление Симса (13) упрощает проверку наличия перестановки α в G. Сначала определим $\sigma_{n-1} = \sigma_{(n-1)j}$, где α принимает $n-1 \mapsto j$, и предположим $\alpha' = \alpha\sigma^-_{n-1}$. Затем определим $\sigma_{n-2} = \sigma_{(n-2)j'}$, где α' принимает $n-2 \mapsto j'$, и предположим $\alpha'' = \alpha'\sigma^-_{n-2}$; и т.д. Если на любом этапе требуемая перестановка σ_{kj} не существует в S_k, то исходная перестановка α не относится к G. В случае (14) этот процесс сводит α к тождеству после обнаружеиия $\sigma_{\mathtt{f}}$, $\sigma_{\mathtt{e}}$, $\sigma_{\mathtt{d}}$, $\sigma_{\mathtt{c}}$ и $\sigma_{\mathtt{b}}$.

Например, пусть α является перестановкой (14)(28)(3c)(69)(7d)(be), которая соответствует транспозиции (12) относительно основной диагонали {0,5,f,a}. Поскольку α фиксирует f, то $\sigma_{\mathtt{f}}$ будет тождествеииой перестаиовкой (), а $\alpha' = \alpha$. Тогда $\sigma_{\mathtt{e}}$ является членом $S_{\mathtt{e}}$, который принимает e $\mapsto$ b, а нменно (14)(36)(9c)(be), и находим $\alpha'' = (28)(39)(6\mathtt{c})(7\mathtt{d})$. Эта перестановка относится к $S_{\mathtt{d}}$, так что α действнтельно является автоморфизмом 4-куба.

Наоборот, (13) также упрощает генерацию всех элемеитов соответствующей группы. Просто нужно пройти все перестановки вида

$$\sigma(1,\mathtt{c}_1)\sigma(2,\mathtt{c}_2)\ldots\sigma(n-1,\mathtt{c}_{n-1}),$$

где $\sigma(k,c_k)$ является (c_k+1)-м элементом S_k для $0 \le c_k < s_k = |S_k|$ и $1 \le k < n$, с помощью любого алгоритма из раздела 7.2.1.1, который проходит $(n-1)$-кортежи $(c_1,\ldots,c_{n-1})$ для соответствующих оснований $(s_1,\ldots,s_{n-1})$.

Применение общей структуры. Основной интерес вызывает группа *всех* перестановок $\{0,1,\ldots,n-1\}$, а в этом случае каждое множество S_k таблицы Симса будет содержать $k+1$ элементов $\{\sigma(k,0),\sigma(k,1),\ldots,\sigma(k,k)\}$, где $\sigma(k,0)$ является тождественной перестановкой и другие принимают k для $\{0,\ldots,k-1\}$ в некотором порядке. (Перестановка $\sigma(k,j)$ не должна быть такой же, как σ_{kj}, и обычно она другая.) Каждая такая таблица Симса приводит к генератору перестановок, согласно следующей схеме.

Алгоритм G (*общий генератор перестановок*). Для заданной таблицы Симса $(S_1,S_2,\ldots,S_{n-1})$, где каждое множество S_k имеет $k+1$ элементов $\sigma(k,j)$, этот алгоритм генерирует все перестановки $a_0a_1\ldots a_{n-1}$ для $\{0,1,\ldots,n-1\}$, используя вспомогательную таблицу $c_n\ldots c_2c_1$.

G1. [Инициализировать.] Установить $a_j \leftarrow j$ и $c_{j+1} \leftarrow 0$ для $0 \le j < n$.

G2. [Посетить.] (В этом месте число со смешанным основанием $\left[\begin{smallmatrix} c_{n-1}, \ldots, c_2, c_1 \\ n, \ \ldots, \ 3, \ 2 \end{smallmatrix}\right]$ является количеством перестановок, посещенных до сих пор.) Посетить перестановку $a_0a_1\ldots a_{n-1}$.

G3. [Прибавить 1 к $c_n\ldots c_2c_1$.] Установить $k \leftarrow 1$. Если $c_k = k$, установить $c_k \leftarrow 0$, то $k \leftarrow k+1$ и повторять, пока не выполняется условие $c_k < k$. Завершить алгоритм, если $k = n$; в противном случае установить $c_k \leftarrow c_k + 1$.

G4. [Переставить.] Применить перестановку $\tau(k,c_k)\omega(k-1)^-$ к $a_0a_1\ldots a_{n-1}$, как объясняется ниже, и вернуться к G2. ▮

Применение перестановки π к $a_0a_1\ldots a_{n-1}$ означает замену a_j на $a_{j\pi}$ для $0 \le j < n$. Это соответствует умножению в обратном порядке (начиная со старшего разряда) на π, как уже объяснялось ранее. Определим

$$\tau(k,j) = \sigma(k,j)\sigma(k,j-1)^- \qquad \text{для } 1 \le j \le k; \tag{15}$$

$$\omega(k) = \sigma(1,1)\ldots\sigma(k,k). \tag{16}$$

Тогда на этапах G3 и G4 сохраняется следующее свойство:

$$a_0a_1\ldots a_{n-1} \text{ является перестановкой } \sigma(1,c_1)\sigma(2,c_2)\ldots\sigma(n-1,c_{n-1}), \tag{17}$$

и лемма S доказывает, что каждая перестановка посещается строго один раз.

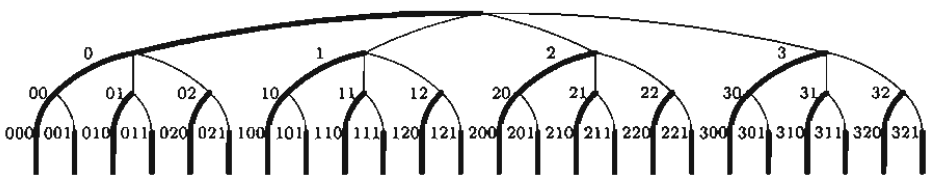


Рис. 19. Алгоритм G неявно обходит это дерево для $n = 4$

Дерево на рис. 19 иллюстрирует алгоритм G для случая $n = 4$. Согласно (17), каждая перестановка $a_0a_1a_2a_3$ of $\{0,1,2,3\}$ соответствует трехцифровой контрольной строке $c_3c_2c_1$ с $0 \le c_3 \le 3$, $0 \le c_2 \le 2$ и $0 \le c_1 \le 1$. Некоторые узлы дерева отмечены только одной цифрой c_3. Они соответствуют перестановкам $\sigma(3, c_3)$ используемой таблицы Симса. Другие узлы, отмеченные двумя цифрами c_3c_2, соответствуют перестановкам $\sigma(2, c_2)\sigma(3, c_3)$. Жирная линия соединяет узел c_3 с узлом c_30 и узел c_3c_2 — с узлом c_3c_20, поскольку $\sigma(2,0)$ и $\sigma(1,0)$ являются тождественными перестановками и эти узлы, по сути, эквивалентны. Добавленне 1 к числу со смешанным основанием $c_3c_2c_1$ на этапе G3 соответствует переходу от одного узла на рис. 19 к его наследннку в прямом порядке обхода, а преобразование на этапе G4 соответственно изменяет перестановки. Например, для $c_3c_2c_1$ происходнт изменение 121 на 200, на этапе G4 происходит умножение в обратном порядке текущей перестановки на

$$\tau(3,2)\omega(2)^- = \tau(3,2)\sigma(2,2)^-\sigma(1,1)^-.$$

Умножение в обратном порядке на $\sigma(1,1)^-$ дает переход от узла 121 к узлу 12, умножение в обратном порядке на $\sigma(2,2)^-$ — переход от узла 12 к узлу 1, а умножение в обратном порядке на $\tau(3,2) = \sigma(3,2)\sigma(3,1)^-$ — переход от узла 1 к узлу $2 \equiv 200$, который является наследником узла в прямом порядке обхода 121. Иначе говоря, умножение в обратном порядке на $\tau(3,2)\omega(2)^-$ — это именно то, что нужно для изменения $\sigma(1,1)\sigma(2,2)\sigma(3,1)$ на $\sigma(1,0)\sigma(2,0)\sigma(3,2)$ с соблюдением (17).

Алгоритм G определяет огромное количество генераторов перестановок (см. упр. 37), потому неудивительно, что в литературе встречается множество его вариантов. Конечно, некоторые из ннх более эффективны, чем другие. Рассмотрнм прнмеры наиболее эффективного соответствия операций используемого компьютера.

Например, можно получить перестановки в обратном колексикографическом порядке, как в особом случае алгоритма G (см. (11)), допуская, что $\sigma(k,j)$ является $(j+1)$-циклом:

$$\sigma(k,j) \;=\; (k{-}j \;\; k{-}j{+}1 \;\; \ldots \;\; k). \tag{18}$$

Дело в том, что $\sigma(k,j)$ должна быть перестановкой, которая соответстует $c_n \ldots c_1$ в обратном колексикографическом порядке для $c_k = j$ и $c_i = 0$ для $i \ne k$, и эта перестановка $a_0a_1 \ldots a_{n-1}$ имеет вид $01 \ldots (k{-}j{-}1)(k{-}j{+}1) \ldots (k)(k{-}j)(k{+}1) \ldots (n{-}1)$. Например, для $n = 8$ и $c_n \ldots c_1 = 00030000$ соответствующая перестановка в обратном колексикографическом порядке имеет вид 01345267, или (2 3 4 5) в циклнческой форме. Для $\sigma(k,j)$ в виде (18) уравнения (15) и (16) приводят к формулам

$$\tau(k,j) = (k{-}j \;\; k); \tag{19}$$

$$\omega(k) = (0\,1)(0\,1\,2)\ldots(0\,1\,\ldots\,k) = (0\,k)(1\,k{-}1)(2\,k{-}2)\ldots = \phi(k), \tag{20}$$

где $\phi(k)$ — это "$(k+1)$-переворот", который заменяет $a_0 \ldots a_k$ на $a_k \ldots a_0$. В этом случае $\omega(k)$ оказывается равной $\omega(k)^-$, поскольку $\phi(k)^2 = ()$.

Уравнения (19) и (20) неявно прнсутствуют за кулисами алгоритма L и в его эквиваленте в обратном лекснкографическом порядке (см. упр. 2), где на этапе L3 применяется транспозиция, а на этапе L4 — переворот. На этапе G4 сначала выполняется переворот, но тождество

$$(k{-}j\;\; k)\phi(k-1) \;=\; \phi(k-1)(j{-}1\;\; k) \tag{21}$$

показывает, что переворот и транспозиция эквивалентны (другой) транспозиции и перевороту.

Действительно, уравнение (21) является особым случаем важного тождества

$$\pi^- (j_1\ j_2\ \dots\ j_t)\pi\ =\ (j_1\pi\ j_2\pi\ \dots\ j_t\pi), \tag{22}$$

которое справедливо для *любой* перестаиовки π и любого t-цикла $(j_1\ j_2\ \dots\ j_t)$. В левой части (22) имеем, например, $j_1\pi \mapsto j_1 \mapsto j_2 \mapsto j_2\pi$, в соответствии с циклом справа. Следовательно, если α и π являются произвольными перестановками, то перестановка $\pi^-\alpha\pi$ (которая называется *сопряжением* α и π) имеет точно такую же циклическую структуру, как и α. Мы просто заменяем каждый элемент j в каждом цикле $j\pi$.

Еще один важный особый случай алгоритма G рассмотрен Р. Дж. Орд-Смитом (R. J. Ord-Smith) [*CACM* **10** (1967), 452; **12** (1969), p.638; см. также *Comp. J.* **14** (1971), p.136–139], который получается подстановкой

$$\sigma(k,j) = (k\ \dots\ 1\ 0)^j. \tag{23}$$

Теперь из (15) ясно, что

$$\tau(k,j) = (k\ \dots\ 1\ 0); \tag{24}$$

и снова имеем:

$$\omega(k) = (0\ k)(1\ k{-}1)(2\ k{-}2)\dots = \phi(k), \tag{25}$$

поскольку перестановка $\sigma(k,k) = (0\ 1\ \dots\ k)$ точно такая же, как и прежде. Интересно в этом методе то, что перестановка, необходимая на этапе G4, а именно $\tau(k,c_k)\omega(k-1)^-$, не зависит от c_k:

$$\tau(k,j)\omega(k-1)^- = (k\ \dots\ 1\ 0)\phi(k-1)^- = \phi(k). \tag{26}$$

Таким образом, алгоритм Орд-Смита является особым случаем алгоритма G, в котором на этапе G4 происходит замена $a_0 \leftrightarrow a_k$, $a_1 \leftrightarrow a_{k-1}$, ... Эта операция обычно выполняется очень быстро, поскольку k мало, и это экономит часть работы алгоритма L. (См. упр. 38 и ссылку на Дж. С. Клюгеля (G. S. Klügel) в разделе 7.2.1.7.)

Можно добиться большего, поступая так, чтобы на этапе G4 каждый раз требовалось делать только одну транспозицию, почти как в алгоритме P, но не обязательно со смежными элементами. Имеется несколько схем реализации этого плана. Самый лучший, вероятно, заключается в следующем:

$$\tau(k,j)\omega(k-1)^- = \begin{cases} (k\ 0), & \text{если } k \text{ четно;} \\ (k\ j{-}1), & \text{если } k \text{ нечетно;} \end{cases} \tag{27}$$

как предложил Б. Р. Хип (B. R. Heap) [*Comp. J.* **6** (1963), p.293–294]. Обратите внимание, что в методе Хипа транспозиция $a_k \leftrightarrow a_0$ происходит всегда, за исключением случаев, когда $k = 3, 5, \dots$, а значение k в пяти случаях на каждых 6 этапах равно 1 или 2. В упр. 40 доказывается, что метод Хипа действнтельно генернрует все перестановки.

Пропуск нежелательных блоков. Однпм важным преимуществом алгоритма G является то, что он обходит все перестановки $a_0 \dots a_{k-1}$ до достижения a_k. Затем он выполняет еще $k!$ циклов перед сменой a_k и т.д. Следовательно, еслн в пронзвольный момент времени мы дойдем до заключительных элементов $a_k \dots a_{n-1}$, которые

не имеют большого значения для решаемой задачи, то можно быстро пропустить все перестановки, которые заканчиваются неинтересующими нас суффиксами. Точнее говоря, можно заменить этап G2 следующими подчиненными этапами.

G2.0. [Нужно?] Если $a_k \dots a_{n-1}$ является ненужным нам суффиксом, то перейти к G2.1. В противном случае установить $k \leftarrow k-1$. Тогда, если $k > 0$, то повторить этот этап; если $k = 0$, то перейти к G2.2.

G2.1. [Пропустить этот суффикс.] Если $c_k = k$, применить $\sigma(k,k)^-$ к $a_0 \dots a_{n-1}$, установить $c_k \leftarrow 0$, $k \leftarrow k+1$ и повторять, пока не будет выполняться условие $c_k < k$. Завершить, если $k = n$, в противном случае установить $c_k \leftarrow c_k + 1$, применить $\tau(k, c_k)$ к $a_0 \dots a_{n-1}$ и вернуться к G2.0.

G2.2. [Посетить.] Посетить перестановку $a_0 \dots a_{n-1}$. ▮

На этапе G1 следует также установить $k \leftarrow n-1$. Обратите внимание на то, что на новых этапах тщательно выполняется условие (17). Этот алгоритм усложнился, поскольку теперь нужно знать перестановки $\tau(k,j)$ и $\sigma(k,k)$, помимо перестановок $\tau(k,j)\omega(k-1)^-$, которые появляются на этапе G4. Но дополнительные усложнения часто оправдываются, поскольку полученная в результате программа может выполняться быстрее.

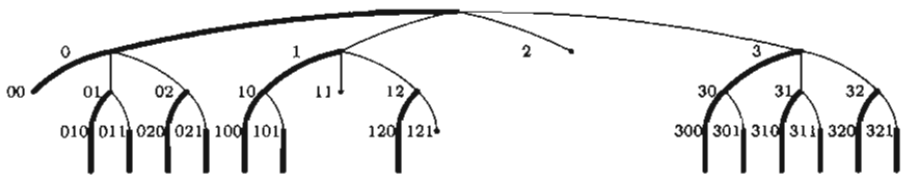


Рис. 20. Нежелательные ветви дерева с рис. 19 можно обрезать, если расширить алгоритм G

Например, на рис. 20 показано, что произойдет с деревом, приведенным на рис. 19, если не учитывать суффиксы $a_0a_1a_2a_3$, которые соответствуют узлам 00, 11, 121 и 2, не представляющим для нас интереса. (Каждый суффикс $a_k \dots a_{n-1}$ перестановки $a_0 \dots a_{n-1}$ соответствует *префиксу* $c_n \dots c_k$ контрольной строки $c_n \dots c_1$, поскольку перестановки $\sigma(1,c_1) \dots \sigma(k-1,c_{k-1})$ не влияют на $a_k \dots a_{n-1}$.) На этапе G2.1 происходит умножение в обратном порядке на $\tau(k,j)$ для перехода от узла $c_{n-1} \dots c_{k+1}j$ к его правому "собрату" $c_{n-1} \dots c_{k+1}(j+1)$ и умножение в обратном порядке на $\sigma(k,k)^-$ для перехода от узла $c_{n-1} \dots c_{k+1}k$ к его родителю $c_{n-1} \dots c_{k+1}$. Таким образом, чтобы перейти от ненужного префикса 121 к его наследнику в прямом порядке обхода, алгоритм выполняет умножение в обратном порядке на $\sigma(1,1)^-$, $\sigma(2,2)^-$ и $\tau(3,2)$, что приводит к переходу от узла 121 к узлу 12, к узлу 1 и 2. (Это, в некоторой степени, исключительный случай, поскольку префикс с $k = 1$ отвергается, только если не нужно посещать единственную перестановку $a_0a_1 \dots a_{n-1}$ с суффиксом $a_1 \dots a_{n-1}$.) После исключения узла 2 $\tau(3,3)$ переводит к узлу 3 и т.д.

Обратите внимание на то, что пропуск суффикса $a_k \dots a_{n-1}$ в этом расширении алгоритма G , по сути, то же самое, что и пропуск префикса $a_1 \dots a_j$ в исходном обозначении, если вернуться к идее о генерации перестановок $a_1 \dots a_n$ для $\{1, \dots, n\}$ и выполнять большую часть работы с правого конца. Наше исходное обозначение соответствует сначала выбору a_1, затем — a_2, ..., потом — a_n. В обозначениях

алгоритма G сиачала выбирается a_{n-1}, затем a_{n-2}, ..., потом a_0. Система обозиачений в алгоритме G имеет обратный порядок, но она значительно упрощает формулы для таблицы Симса. Опытный программист быстро и без особого труда научится переходить от одной точки зрения к другой.

Эти идеи можно применить к буквометике, поскольку, например, ясно, что большая часть вариантов для букв D, E и Y для сложения SEND и MORE ие даст в сумме MONEY. Необходимо, чтобы в этой задаче $(\mathtt{D}+\mathtt{E}-\mathtt{Y}) \bmod 10 = 0$. Следовательно, миогие перестановки можио сразу исключить из рассмотрения.

Вообще, если r_k является максимальной степенью 10, которая делит значение сигнатуры s_k, то можио отсортировать буквы и присвоить коды $\{0, 1, \ldots, 9\}$ так, что $r_0 \geq r_1 \geq \cdots \geq r_9$. Например, для решения задачи "TRIO+SONATA" в (7) можно было бы использовать $(0, 1, \ldots, 9)$ для (A, I, N, O, R, S, T, V, X, Y) соответственно, получая сигнатуры

$$s_0 = 0, \quad s_1 = -100000, \quad s_2 = 210000, \quad s_3 = -100, \quad s_4 = -100,$$
$$s_5 = 21010, \quad s_6 = 210, \quad s_7 = -1010, \quad s_8 = -7901, \quad s_9 = -998.$$

Следовательно, $(r_0, \ldots, r_9) = (\infty, 5, 4, 2, 2, 1, 1, 1, 0, 0)$. Теперь, если обратиться к этапу G2.0 для значения k с $r_{k-1} \neq r_k$, то можно сказать, что суффикс $a_k \ldots a_9$ не нужен в том смысле, что его можно пропустить, пока $a_k s_k + \cdots + a_9 s_9$ не кратно $10^{r_k - 1}$. Кроме того, из (10) следует, что суффикс $a_k \ldots a_9$ ие нужен, если $a_k = 0$ и $k \in F$. Множество первых букв F теперь имеет вид $\{1, 2, 7\}$.

Предыдущий подход для решения буквометик на этапах A1–A3 использовал метод "грубой силы" для перебора 10! вариантов. В определенных условиях он может работать очень быстро, поскольку метод перестаиовок смежиых элемеитов функционирует только с шестью ссылками на память для одной перестановки. Однако 10! равияется 3 628 800, так что для всего процесса потребуется почти 22 мегамемса независимо от типа буквометики. Наоборот, расширеиный алгоритм G на основе метода Хипа с только что онисанными пропусками найдет все четыре решеиия (7) меньше чем за 128 *кило*мемса! Таким образом, метод пропуска суффикса работает в 170 раз быстрее, чем предыдущий метод, который слепо перебирает все варианты.

Бо́льшая часть из 128 киломемсов в новом подходе тратятся на применение $\tau(k, c_k)$ на этапе G2.1. Другие обращения к памяти связаны с применением $\sigma(k, k)^-$ на этом этапе, но τ требуется 7812 раз, тогда как σ^- требуется только 2162 раза. Причину такого поведения легко понять из рис. 20, поскольку "ускоренное движение" $\tau(k, c_k)\omega(k-1)^-$ на этапе G4 вряд ли когда возникает. В этом случае оно используется только четыре раза, т.е. по одному разу для каждого решенння. Такнм образом, обход дерева в прямом порядке выполняется практически полностью за τ шагов вправо н σ^- шагов вверх. τ шагов домнннруют в данной задаче, где посещается очень мало перестановок, поскольку каждый этап $\sigma(k, k)^-$ предваряется k этапами $\tau(k, 1)$, $\tau(k, 2)$, ..., $\tau(k, k)$.

Благодаря этому анализу стало ясно, что метод Хипа, который увеличивается при оптимизации перестановок $\tau(k, j)\omega(k-1)^-$ так, что каждый переход на G4 является простой транспозицией, *не* очень хорош для расширенного алгоритма G, если на этапе G2.0 не пропускается сравнительно немного суффиксов. Более простой обратный колексикографический порядок, для которого $\tau(k, j)$ всегда является простой транснозицией, теперь гораздо нрнвлекательнее (см. (19)). Действнтельно,

алгоритм G с обратным колексикографическим порядком находит решение буквометики (7) всего за 97 киломемса.

Аналогичные результаты можно получить при решении других задач с буквометиками. Например, если применить расширенный алгоритм G для буквометики в пунктах (a)–(h) упр. 24, то для вычислений потребуется

$$\begin{aligned}&(551, 110, 14, 8, 350, 84, 153, 1598)\ \text{киломемс с методом Хипа;}\\ &(429,\ 84, 10, 5, 256, 63, 117, 1189)\ \text{кнломемс с обратным}\\ &\qquad\qquad\qquad\qquad\qquad\qquad\text{колексикографическим порядком.}\end{aligned}\tag{28}$$

Коэффициент ускорения для обратного колексикографического порядка в этнх примерах, по сравнению с методом "грубой силы" в алгоритме T, находится в пределах от 18 в пункте (h) до 4200 в пункте (d) и в среднем равняется 80. А коэффициент ускорения для метода Хипа в среднем составляет около 60.

Из алгоритма L известно, однако, что лексикографический порядок легко обрабатывается *без* усложнения контрольной таблицы $c_n \ldots c_1$, используемой в алгоритме G. Более внимательное изучение алгоритма L показывает, что его поведение можно улучшить, когда перестановки часто пропускаются, с помощью связанного списка вместо последовательного массива. Улучшенный алгоритм прекрасно подходит для широкого спектра алгоритмов генерации ограниченных классов перестановок.

Алгоритм X (*лексикографические перестановки с ограниченными префиксами*). Этот алгоритм генерирует все перестановки $a_1a_2\ldots a_n$ для $\{1,2,\ldots,n\}$, которые удовлетворяют заданной последовательности проверок

$$t_1(a_1),\quad t_2(a_1,a_2),\quad \ldots,\quad t_n(a_1,a_2,\ldots,a_n),$$

посещая их в лексикографическом порядке. В нем используется вспомогательная таблица связей $l_0, l_1, \ldots, l_n$ для поддержания циклического списка неиспользуемых элементов, так что для текущих доступных элементов

$$\{1,\ldots,n\} \setminus \{a_1,\ldots,a_k\} = \{b_1,\ldots,b_{n-k}\},\qquad \text{где } b_1 < \cdots < b_{n-k},\tag{29}$$

имеем:

$$l_0 = b_1,\quad l_{b_j} = b_{j+1}\quad \text{для } 1 \le j < n-k \quad \text{и} \quad l_{b_{n-k}} = 0.\tag{30}$$

Здесь используется всномогательная таблица $u_1\ldots u_n$ для отмеиы операцнй, которые были выполиены для массива l.

X1. [Инициализировать.] Установить $l_k \leftarrow k+1$ для $0 \le k < n$ и $l_n \leftarrow 0$. Затем устаиовнть $k \leftarrow 1$.

X2. [Перейти на уровеиь k.] Установить $p \leftarrow 0$, $q \leftarrow l_0$.

X3. [Проверить $a_1\ldots a_k$.] Установнть $a_k \leftarrow q$. Если $t_k(a_1,\ldots,a_k)$ ложно, нерейтн к X5. В противном случае, если $k = n$, посетить $a_1\ldots a_n$ и перейти к X6.

X4. [Увеличить k.] Устаиовить $u_k \leftarrow p$, $l_p \leftarrow l_q$, $k \leftarrow k+1$ и вернуться к X2.

X5. [Увеличить a_k.] Установить $p \leftarrow q$, $q \leftarrow l_p$. Если $q \neq 0$, вернуться к X3.

X6. [Уменьшить k.] Установить $k \leftarrow k-1$ и завершить, если $k = 0$. В противном случае установнть $p \leftarrow u_k$, $q \leftarrow a_k$, $l_p \leftarrow q$ и перейти к X5. ▮

Основная идея этого элегантного алгоритма принадлежит М. Ч. Эру (M. C. Er) [*Comp. J.* **30** (1987), p.282]. Его можно применить к буквометике, слегка изменяя систему обозначений, получая перестановки $a_0 \ldots a_9$ of $\{0, \ldots, 9\}$ и допуская, что l_{10} играет прежнюю роль l_0. Полученный алгоритм затратит только 49 киломемс для решения задачи "TRIO+SONATA" в (7), а буквометики из пунктов (a)–(h) упр. 24 решаются соответственно за

$$(248, 38, 4, 3, 122, 30, 55, 553) \text{ киломемс.} \tag{31}$$

Таким образом, он работает почти в 165 раз быстрее, чем метод "грубой силы".

Еще один способ применения алгоритма X для решения буквометик часто выполняется еще быстрее (см. упр. 49).

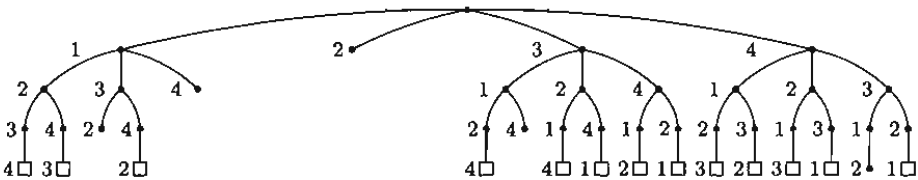


Рис. 21. Дерево, которое неявно проходится в алгоритме X для $n = 4$, если посещаются все перестановки, за исключением тех, которые начинаются с 132, 14, 2, 314 или 4312

***Дуальные методы.** Если $S_1, \ldots, S_{n-1}$ является таблицей Симса для группы перестановок G, то, как известно из леммы S, каждый элемент G можно выразить единственным образом в виде произведения $\sigma_1 \ldots \sigma_{n-1}$, где $\sigma_k \in S_k$; см. (13). В упр. 50 показано, что каждый элемент α можно выразить единственным образом в дуальной форме:

$$\alpha = \sigma_{n-1}^- \ldots \sigma_2^- \sigma_1^-, \qquad \text{где } \sigma_k \in S_k \text{ для } 1 \le k < n, \tag{32}$$

и этот факт прнводит к другому большому семейству генераторов перестановок. В частности, когда G является группой всех $n!$ перестановок, каждую перестановку можно записать в виде

$$\sigma(n-1, c_{n-1})^- \ldots \sigma(2, c_2)^- \sigma(1, c_1)^-, \tag{33}$$

где $0 \le c_k \le k$ для $1 \le k < n$ и перестановки $\sigma(k, j)$ те же, что и в алгоритме G. Теперь, однако, желательно быстрее варьировать c_{n-1} и быстрее варьировать c_1, и поэтому получаем алгоритм другого типа.

Алгоритм H (*дуальный генератор перестановок*). Для заданной таблицы Симса, как и в алгоритме G, этот алгоритм генерирует все перестановки $a_0 \ldots a_{n-1}$ из $\{0, \ldots, n-1\}$ с помощью вспомогательной таблицы $c_0 \ldots c_{n-1}$.

H1. [Инициализировать.] Установить $a_j \leftarrow j$ и $c_j \leftarrow 0$ для $0 \le j < n$.

H2. [Посетить.] (В этом месте число со смешанным основанием $\left[\begin{smallmatrix} c_1, & c_2, & \ldots, & c_{n-1} \\ 2, & 3, & \ldots, & n \end{smallmatrix}\right]$ является количеством посещенных до сих пор перестановок.) Посетить перестановку $a_0 a_1 \ldots a_{n-1}$.

H3. [Добавить 1 к $c_0 c_1 \ldots c_{n-1}$.] Установить $k \leftarrow n-1$. Если $c_k = k$, установить $c_k \leftarrow 0$, $k \leftarrow k-1$ и повторять, пока не выполняется условие $k = 0$ или условие

$c_k < k$. Завершить алгоритм, если $k = 0$; в противном случае установить $c_k \leftarrow c_k + 1$.

H4. [Переставить.] Применить перестановку $\tau(k, c_k)\omega(k+1)^-$ к $a_0 a_1 \dots a_{n-1}$, как описано ниже, и вернуться к H2. ▮

Хотя этот алгоритм кажется практически идентичным алгорнтму G, перестановкн τ и ω, которые необходимы на этапе H4, существенно отличаются от перестановок, которые необходимы на этапе G4. Новые правила, которые заменяют (15) и (16), имеют следующий внд:

$$\tau(k,j) = \sigma(k,j)^-\sigma(k,j-1) \qquad \text{для } 1 \le j \le k, \tag{34}$$

$$\omega(k) = \sigma(n-1,n-1)^-\sigma(n-2,n-2)^- \dots \sigma(k,k)^-. \tag{35}$$

Количество вариантов так же велико, как и для алгоритма G, поэтому ограничимся здесь нескольknми случаями, которые обладают особыми преимуществами. Один естественный случай, конечно, связан с таблицей Симса, которая в алгоритме G приводит к созданню обратного колексикографического порядка, а именно

$$\sigma(k,j) \;=\; (k{-}j \;\; k{-}j{+}1 \;\; \dots \;\; k), \tag{36}$$

как в (18). Полученый генератор перестановок очень похож на метод простых измененнй. Поэтому можно сказать, что алгоритмы L и P по сути дуальны по отношению друг к другу (см. упр. 52).

Другая естественная ндея состоит в создании таблицы Симса, для которой этап H4 всегда выполняет одну транспозицию двух элементов, по аналогии с коиструкцией (27), которая достигает максимальной эффективности на этапе G4. Но такая миссия оказывется невозможной: ее нельзя выполнnть даже для $n = 4$. Если начать с тождественной перестановки $a_0a_1a_2a_3 = 0123$, то переходы от контрольной таблицы $c_0c_1c_2c_3 = 0000$ к 0001, и к 0002, и к 0003 должны переместить 3. Итак, если они являются транспозициями, то онн должны быть $(3\,a)$, $(a\,b)$ н $(b\,c)$ для некоторой перестановки abc множества $\{0,1,2\}$. Перестановка, соответствующая $c_0c_1c_2c_3 = 0003$, теперь имеет вид $\sigma(3,3)^- = (b\,c)(a\,b)(3\,a) = (3\,a\,b\,c)$; а следующая перестановка, соответствующая $c_0c_1c_2c_3 = 0010$, нмеет вид $\sigma(2,1)^-$, что должно зафиксировать элемент 3. Единственная пригодная транспозиция имеет вид $(3\,c)$, следовательно, $\sigma(2,1)^-$ должна иметь вид $(3\,c)(3\,a\,b\,c) = (a\,b\,c)$. Аналогично находим, что $\sigma(2,2)^-$ должна иметь вид $(a\,c\,b)$, а перестановка, соответствующая $c_0c_1c_2c_3 = 0023$, будет иметь вид $(3\,a\,b\,c)(a\,c\,b) = (3\,c)$. На этапе H4 она преобразуется в перестановку $\sigma(1,1)^-$, которая соответствует контрольной таблице 0100, которая следует за 0023. Но единственная транспозиция, преобразующая $(3\,c)$ в перестановку, которая фиксирует 2 и 3, имеет вид $(3\,c)$; а полученная в результате перестановка также фиксирует 1, поэтому ею ие может быть $\sigma(1,1)^-$.

Из доказательства в предыдущем абзаце следует, что алгоритм H иельзя использовать для генерации всех перестановок с минимальным числом транспозиций. Но из него также следует простая схема геиерации, которая очень близка к этому минимуму, а полученый в результате алгоритм очень привлекателен, поскольку он выполняет дополнительную работу только один раз каждые $n(n-1)$ этапов (см. упр. 53).

Наконец, рассмотрим дуальный вариант метода Орд-Смита, когда

$$\sigma(k,j) = (k\ \ldots\ 1\ 0)^j, \tag{37}$$

как в (23). Зачение $\tau(k,j)$ снова не зависит от j:

$$\tau(k,j) = (0\ 1\ \ldots\ k), \tag{38}$$

и это вносит особые преимущества в алгоритме H, поскольку он позволяет избавиться от контрольной таблицы $c_0c_1\ldots c_{n-1}$. Дело в том, что $c_{n-1} = 0$ на этапе H3 тогда и только тогда, когда $a_{n-1} = n-1$ вследствие (32). Действительно, когда $c_j = 0$ для $k < j < n$ на этапе H3, имеем $c_k = 0$ тогда и только тогда, когда $a_k = k$. Следовательно, этот вариант алгоритма H можно переформулировать описанным ннже способом.

Алгоритм C (*генерация перестановок с помощью циклических сдвигов*). Этот алгоритм посещает все перестановкн $a_1\ldots a_n$ различных элементов $\{x_1,\ldots,x_n\}$.

C1. [Инициализировать.] Установить $a_j \leftarrow x_j$ для $1 \le j \le n$.

C2. [Посетить.] Посетнть перестановку $a_1\ldots a_n$ н установнть $k \leftarrow n$.

C3. [Сдвиг.] Заменить $a_1a_2\ldots a_k$ эа счет циклического сдвига на $a_2\ldots a_ka_1$ и вернуться к C2, если $a_k \ne x_k$.

C4. [Уменьшнть k.] Установить $k \leftarrow k-1$ и вернуться к C3, если $k > 1$. ▮

Например, последовательные перестановки $\{1,2,3,4\}$, сгенерированные для $n = 4$, имеют вид

1234, 2341, 3412, 4123, (1234),
2314, 3142, 1423, 4231, (2314),
3124, 1243, 2431, 4312, (3124), (1234),
2134, 1342, 3421, 4213, (2134),
1324, 3241, 2413, 4132, (1324),
3214, 2143, 1432, 4321, (3214), (2134), (1234),

где непосещенные промежуточные перестановки показаны в скобках. Этот алгоритм может быть простейшим генератором перестановок, с точки зренння минимальности размера кода программы. Его предложил Дж. Дж. Лангдон-мл. (G. G. Jr. Langdon) [см. *CACM* **10** (1967), p.298–299; **11** (1968), p.392]. Аналогичные методы были опубликованы ранее Ч. Томпкинсом (C. Tompkins) [см. *Proc. Symp. Applied Math.* **6** (1956), p.202–205] и более явно Р. Зайтцем (R. Seitz) [см. *Unternehmensforschung* **6** (1962), p.2–15]. Эта процедура особенно хороша для приложений, в которых эффективна операция циклического сдвига, например, когда последовательные перестановки хранятся в регистре компьютера, а не в массиве.

Основной недостаток дуальных методов заключается в том, что они обычно плохо адаптируются к ситуациям, когда нужно пропустить большие блоки перестановок, поскольку множество всех перестановок с заданным значением первых контрольных элементов $c_0c_1\ldots c_{k-1}$ обычно не имеет большого значения. Однако особый случай (36) иногда является исключением, поскольку $n!/k!$ перестановок с элементами $c_0c_1\ldots c_{k-1} = 00\ldots0$ в таком случае являются точно такими $a_0a_1\ldots a_{n-1}$, в которых 0 предшествует 1, 1 — 2, ..., а $k-2$ — $k-1$.

***Метод обмена Эрлиха.** Эрлих открыл совершенно иной подход для генерации перестановок на основе еще одного способа использования контрольной таблицы $c_1 \ldots c_{n-1}$. В его методе каждая перестановка получается из предыдущей за счет обмена самого левого элемента с другим элементом.

Алгоритм E (*обмены Эрлиха*). Этот алгоритм генернрует все перестановки различных элементов $a_0 \ldots a_{n-1}$ с помощью вспомогательной таблицы $b_0 \ldots b_{n-1}$ и $c_1 \ldots c_n$.

E1. [Инициализировать.] Установить $b_j \leftarrow j$ и $c_{j+1} \leftarrow 0$ для $0 \le j < n$.

E2. [Посетить.] Посетить перестановку $a_0 \ldots a_{n-1}$.

E3. [Найти k.] Установить $k \leftarrow 1$. Затем, если $c_k = k$, установить $c_k \leftarrow 0$, $k \leftarrow k+1$ и повторять, пока не выполняется условие $c_k < k$. Завершить, если $k = n$, в противном случае установить $c_k \leftarrow c_k + 1$.

E4. [Обменять.] Заменить $a_0 \leftrightarrow a_{b_k}$.

E5. [Перевернуть.] Установить $j \leftarrow 1$, $k \leftarrow k-1$. Если $j < k$, поменять $b_j \leftrightarrow b_k$, установить $j \leftarrow j+1$, $k \leftarrow k-1$ и повторять, пока не выполняется условие $j \ge k$. Вернуться к E2. ▮

Обратите внимание на то, что этапы E2 и E3 идентичны этапам G2 и G3 алгоритма G. Наиболее изумительный момент в этом алгоритме, о котором Эрлих сообщил Мартину Гарднеру (Martin Gardner) в 1987 году, заключается в том, что он работает (доказательство см. в упр. 55). Аналогичный метод, который упрощает операции на этапе E5, можно обосновать таким же образом (см. упр. 56). Среднее число замен на этапе E5 меньше 0,18 (см. упр. 57).

Алгоритм E не быстрее, чем другие упомянутые выше методы. Но он обладает тем прекрасным свойством, что изменяет каждую перестановку минимальным образом, т.е. с использованием только $n-1$ разных видов транспозиций. Тогда как в алгоритме P используются смежные замены, $a_{t-1} \leftrightarrow a_t$, в алгоритме E используются обмены первых элементов, $a_0 \leftrightarrow a_t$, которые также называются *звездными транспозициями*, для некоторой отобранной последовательности индексов $t[1]$, $t[2]$, $\ldots$, $t[n!-1]$. И если нужно повторно генерировать перестановки для такого же очень небольшого значения n, то можно предварительно вычислить эту последовательность, как это делалось в алгоритме T для последовательности индексов алгоритма P. Звездные транспозиции обладают преимуществом по сравнению со смежнымн заменами, поскольку нам всегда известно зиачение a_0 из предыдущего обмена, т.е. его не нужно считывать из памяти.

Пусть E_n является последовательностью $n!-1$ таких индексов t, что алгоритм E обменивает a_0 на a_t на этапе E4. Поскольку E_{n+1} начинается с E_n, то можно рассматривать E_n как первые $n!-1$ элементов бесконечной последовательности:

$$E_\infty = 1212132121231212132121243131321313 12 \ldots . \tag{39}$$

Например, если $n = 4$ и $a_0a_1a_2a_3 = 1234$, то перестановки, посещенные алгоритмом E, имеют вид

$$\begin{aligned} &1234,\ 2134,\ 3124,\ 1324,\ 2314,\ 3214,\\ &4213,\ 1243,\ 2143,\ 4123,\ 1423,\ 2413,\\ &3412,\ 4312,\ 1342,\ 3142,\ 4132,\ 1432,\\ &2431,\ 3421,\ 4321,\ 2341,\ 3241,\ 4231. \end{aligned} \tag{40}$$

***Малые генераторы перестановок.** После знакомства с алгоритмами P и E вполне естественно было бы спросить: а нельзя ли все перестановки получить за счет только *двух* основных операций вместо $n-1$? Например, А. Нийенхуйс (A. Nijenhuis) и Г. С. Вильф (H. S. Wilf) [см. *Combinatorial Алгоритмы* (1975), exercise 6] заметили, что все перестановкн можно генерировать для $n = 4$, если заменять $a_1a_2a_3\ldots a_n$ на каждом этапе либо на $a_2a_3\ldots a_na_1$ либо на $a_2a_1a_3\ldots a_n$. Они заиитересовались вопросом: есть ли такой же метод для всех n?

Вообще, если G является произвольной группой перестановок и если $\alpha_1, \ldots, \alpha_k$ являются элементами G, то *граф Кэли* для G с генераторами $(\alpha_1, \ldots, \alpha_k)$ является ориентированным графом, вершины которого являются π-перестановками группы G и ребра которого ндут от π к $\alpha_1\pi$, ..., $\alpha_k\pi$. [Arthur Cayley, *American J. Math.* **1** (1878), p.174–176.] Вопрос Нийенхуйса и Вильфа эквивалеитеи вопросу: имеется ли гамильтонов путь для графа Кэли для всех перестановок $\{1, 2, \ldots, n\}$ с генераторами σ и τ, где σ является циклической перестановкой $(1\ 2\ \ldots\ n)$ и τ является транспозицией $(1\ 2)$.

Согласно основной теореме Р. А. Ранкина (R. A. Rankin) [см. *Proc. Cambridge Philos. Soc.* **44** (1948), p.17–25], можно сделать вывод, что во многих случаях графы Кэли с двумя генераторами не имеют гамильтонова цикла.

Теорема R. *Пусть G является группой из g перестановок. Если граф Кэли для G с генераторами (α, β) имеет гамильтонов цикл и если перестановки $(\alpha, \beta, \alpha\beta^-)$ имеют соответственно порядок (a, b, c), то либо c четно, либо g/a и g/b иечетны.*

(*Порядком* перестановки α называется такое наименьшее положительиое целое число a, что α^a является тождественной перестановкой.)

Доказательство. См. упр. 73. ▮

В частностн, когда $\alpha = \sigma$ и $\beta = \tau$, как указано выше, то имеем $g = n!$, $a = n$, $b = 2$ и $c = n - 1$, поскольку $\sigma\tau^- = (2\ \ldots\ n)$. Следовательно, можно заключить, что гамильтонов цикл невозможен для четных $n \geq 4$. Однако гамильтонов *путь* можно легко создать для $n = 4$, так как можно объединиить 12-этапные циклы:

$$\begin{aligned} 1234 &\to 2341 \to 3412 \to 4312 \to 3124 \to 1243 \to 2431 \\ &\to 4231 \to 2314 \to 3142 \to 1423 \to 4123 \to 1234, \\ 2134 &\to 1342 \to 3421 \to 4321 \to 3214 \to 2143 \to 1432 \\ &\to 4132 \to 1324 \to 3241 \to 2413 \to 4213 \to 2134, \end{aligned} \tag{41}$$

начиная с 2341, переходя от 1234 к 2134 и заканчивая 4213.

Ф. Раски (F. Ruskey), М. Джиянг (M. Jiang) и А. Вестон (A. Weston) [см. *Discrete Applied Math.* **57** (1995), p.75–83] досконально исследовали граф σ–τ для $n = 5$ и обнаружнли, что он имеет пять существенно различных гамильтоновых циклов, один из которых ("наиболее прекрасный") показан на рис. 22,*а*. Они также обнаружили гамильтоиов путь для $n = 6$, что потребовало ненмоверных усилий, поскольку для поиска необходима была бинарная древовидиая схема принятня решений с 720 этапами. К сожалению, найденное решение не имеет очевндной логической структуры. Чуть менее сложиый путь описывается в упр. 70, ио даже этот путь нельзя назвать простым. Следовательно, подход на основе графа σ–τ, вероятно, не будет иметь практического интереса для более крупиых зиачений n, если ие будет открыта новая конструкция. Р. К. Комптон (R. C. Compton) и С. Дж. Внльямсон (S. G. Williamson)

[см. *Linear and Multilinear Algebra* **35** (1993), p.237–293] доказали, что гамильтонов цикл существует для всех n, если допустить три генератора σ, σ^- и τ вместо двух генераторов σ и τ. Их циклы обладают интересным свойством, что каждая n-я трансформация является τ-генератором, а все промежуточные $n-1$ трансформации являются либо только σ-генераторами, либо только σ^--генераторами. Но их метод очень сложен, чтобы для него можно было дать достаточно краткое объяснение.

В упр. 69 описывается общий алгоритм перестановок, который достаточно прост и требует только трех генераторов порядка 2. На рис. 22,*b* этот метод иллюстрируется для $n = 5$, мотивом для которого были упоминавшиеся ранее примеры алгоритмов перезвона колоколов.

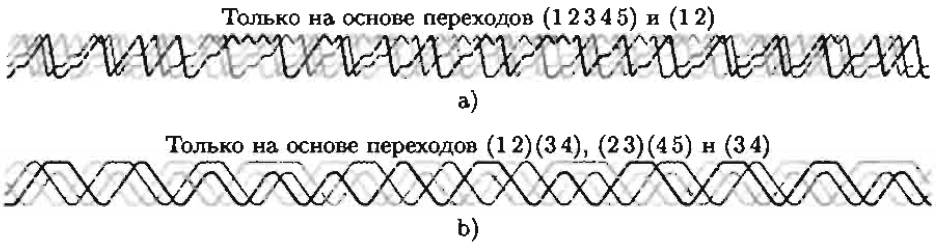


Рис. 22. Гамильтоновы циклы для 5! перестановок

Быстрее, еще быстрее. Каков самый быстрый способ генерации перестановок? Этот вопрос часто возникает в публикациях, посвященных информатике, поскольку для перебора всех $n!$ вариантов желательно максимально сократить время перебора. Ответы всегда давались крайне противоречивые; поскольку существует множество разных формулировок этого вопроса, попробуем разобраться с ними, изучая механизм скорейшей генерации перестановок на компьютере `MMIX`.

Сначала допустим, что наша цель заключается в создании перестановок в массиве n последовательно расположенных слов памяти (октабайт). Самым быстрым способом достижения этой цели среди всех рассмотренных в данном разделе является оптимизированный метод Хипа в (27), предложенный Р. Седжвиком (R. Sedgewick) [см. *Computing Surveys* **9** (1977), p.157–160].

Ключевая идея заключается в оптимизации кода для наиболее распространенных случаев на этапах G2 и G3, а именно случаев, в которых вся деятельность происходит в начале массива. Если регистры u, v и w содержат первые три слова и если следующие генерируемые шесть перестановок включают перестановку этих слов всеми шестью возможными способами, то можно выполнить эту работу следующим образом:

```
PUSHJ 0,Visit
STO v,A0;  STO u,A1;  PUSHJ 0,Visit
STO w,A0;  STO v,A2;  PUSHJ 0,Visit
STO u,A0;  STO w,A1;  PUSHJ 0,Visit
STO v,A0;  STO u,A2;  PUSHJ 0,Visit
STO w,A0;  STO v,A1;  PUSHJ 0,Visit
```
(42)

(где `A0` — это адрес октабайта a_0 и т.д.). Полная программа перестановки, которая следит за правильным размещением в регистрах u, v и w, приводится в упр. 77.

Другие инструкцин менее важны, поскольку они выполняются только $\frac{1}{6}$ часть всего времени. Общие затраты на перестановку, не считая 4υ, необходимые для выполнения операций PUSHJ и POP для каждого вызова-посещения Visit, равны приблизительно $2{,}77\mu + 5{,}69\upsilon$ в этом подходе. Если использовать четыре регистра, u, v, w, x и расширить (42) до 24 вызовов-посещений Visit, то время выполнения на одну перестановку уменьшается до $2{,}19\mu + 3{,}07\upsilon$. А с регистром r и $r!$ вызовами-посещениями Visit, как показано в упр. 78, затраты равняются $(2+O(1/r!))(\mu+\upsilon)$, что очень близко к затратам на две инструкции STO.

Последнее значение, конечно, является минимальным возможным временем для любого метода, который генерирует все перестановки в последовательном массиве. ...Не так ли? Ранее предполагалось, что в процедуре посещения необходимо просматривать все перестановки в последовательных положениях, но, возможно, эта процедура способна считывать перестановки из разных начальных положений. Тогда можно организовать работу так, чтобы зафиксировать a_{n-1} и хранить поблизости две копии других элементов:

$$a_0 a_1 \ldots a_{n-2} a_{n-1} a_0 a_1 \ldots a_{n-2}. \tag{43}$$

Если теперь рассмотреть $(n-1)!$ перестановок $a_0 a_1 \ldots a_{n-2}$, то, всегда изменяя одновременно обе копии с помощью двух команд STO вместо одной, можно допустнть, что каждый вызов Visit связан с n перестановками

$$a_0 a_1 \ldots a_{n-1}, \quad a_1 \ldots a_{n-1} a_0, \quad \ldots, \quad a_{n-1} a_0 \ldots a_{n-2}, \tag{44}$$

которые появляются последовательно. Затраты на одну перестановку сокращаются до затрат на три простые инструкции ADD, CMP, PBNZ плюс $O(1/n)$. [См. работу Я. Л. Вароля (Y. L. Varol) и Д. Ротема (Rotem, D.) *Comp. J.* **24** (1981), p.173–176.]

Более того, можно вовсе не тратить время на сохранение перестановок в памяти. Допустим, например, что наша цель заключается в генерации всех перестановок $\{0, 1, \ldots, n-1\}$. Значение n, вероятно, будет не более 16, поскольку $16! = 20\,922\,789\,888000$ н $17! = 355\,687\,428\,096000$. Следовательно, вся перестановка поместится в 16 ниблах октабайта и ее можно сохранить в одном регнстре. Это будет полезно, только если процедуре посещення не потребуется распаковывать отдельные ниблы. Предположим, что это не потребуется. Как можно быстро генерировать перестановки в ниблах 64-разрядного регистра?

Одна идея, предложенная А. Дж. Голдстейном (A. J. Goldstein) [см. *U. S. Patent 3383661* (14 May 1968)], заключается в предварительном вычислении таблицы $(t[1], \ldots, t[5039])$ переходов типа "простые изменения" для семи элемеитов с помощью алгоритма Т. Эти числа $t[k]$ лежат между 1 и 6, так что можно упаковать 20 из них в 64-разрядное слово. Число $\sum_{k=1}^{20} 2^{3k-1} t[20j+k]$ было бы удобно поместить в j-е слово вспомогательиой таблицы для $0 \le j < 252$, где $t[5040] = 1$. Например, таблица начинается с кодового слова

0000101001110010111010011010110001101001110001010011100101110000.

Следующая программа способна эффективно считывать такие коды:

```
Perm  ⟨Установить регистр a в первую перестановку.⟩
0H    LDA  p,T       p ← адрес первого кодового слова.
      JMP  3F
1H    ⟨Посетить перестановку в регистре a.⟩
      ⟨Обменять ниблы a, которые лежат в t битах справа.⟩
      SRU  c,c,3     c ← c ≫ 3.
2H    И    t,c,#1c   t ← c&(11100)₂.                          (45)
      PBNZ t,1B      Условный переход, если t ≠ 0.
      ADD  p,p,8
3H    LDO  c,p,0     c ← следующее кодовое слово.
      PBNZ c,2B      (За последним кодовым словом идет 0.)
      ⟨Если не выполнено, то продвинуть ведущие n − 7 ниблов и вернуться к 0B.⟩
```

В упр. 79 показано, как выполняется операция ⟨Обменять ниблы ... ⟩ за счет семи инструкций с помощью манипуляций с битами, которые предусмотрены на большинстве компьютеров. Следовательно, затраты на одну перестановку будут чуть больше 10υ. (Затраты на выполнение инструкций, которые извлекают новые кодовые слова, равны всего $(\mu + 5\upsilon)/20$, а на выполнение инструкций, которые продвигают ведущие $n-7$ ниблов, еще более пренебрежимо малы, поскольку меньше в 5040 раз.) Обратите внимание на то, что теперь нет необходимости в использовании операций PUSHJ и POP, которые применялись в (42) с затратами 4υ.

Однако можно повысить эффективность еще больше, адаптируя метод Лангдона с циклическим сдвигом, т.е. алгоритм C. Допустим, что в начале имеется лексикографически самая крупная перестановка и тогда нужно сделать следующее:

```
      GREG @
0H    OCTA #fedcba9876543210&(1<<(4*N)-1)
Perm  LDOU a,0B                          Установить a ← #...3210.
      JMP  2F
1H    SRU  a,a,4*(16-N)                  a ← ⌊a/16^(16−n)⌋.
      OR   a,a,t                         a ← a | t.            (46)
2H    ⟨Посетить перестановку в регистре a.⟩
      SRU  t,a,4*(N-1)                   t ← ⌊a/16^(n−1)⌋.
      SLU  a,a,4*(17-N)                  a ← 16^(17−n) a mod 16^16.
      PBNZ t,1B                          To 1B, если t ≠ 0.
      ⟨Продолжить работу методом Лангдона.⟩
```

Время выполнения на одну перестановку теперь равно всего $5\upsilon + O(1/n)$, опять-таки без необходимости использования инструкций PUSHJ и POP. (См. упр. 81 с описанием интересного способа расширения (46) до полной программы с получением очень короткой и быстрой процедуры.)

Быстрые генераторы перестановок занимательны, но на практике можно сэкономить больше времени, оптимизируя процедуру посещений, а не ускоряя генератор.

Топологическая сортировка. Вместо обработки всех $n!$ перестановок of $\{1, \ldots, n\}$ часто требуется обратить внимание только на перестановки, которые удовлетворяют определенным ограничениям. Например, нас могут интересовать только те перестановки, в которых 1 предшествует 3, 2 — 3 и 2 — 4. Для множества $\{1, 2, 3, 4\}$

существует пять таких перестановок, а именно:

$$1234,\ 1243,\ 2134,\ 2143,\ 2413. \tag{47}$$

Задача *топологической сортировки*, которая рассматривается в разделе 2.2.3 как первый пример нетривиальных структур данных, является общей задачей поиска перестановок, которые удовлетворяют m условиям $x_1 \prec y_1, \ldots, x_m \prec y_m$, где $x \prec y$ означает, что x предшествует y в перестановке. Эта задача часто возникала на практике, а поэтому успела получить несколько разных названий. Например, она часто называется задачей *линейного вложения*, поскольку объекты нужно упорядочить на одной линии с сохранением определенного порядка связей. Ее также называют задачей расширения частичного упорядочения до полного упорядочения (см. упр. 2.2.3–14).

Целью раздела 2.2.3 был поиск *одной* перестановки, которая удовлетворяла бы заданиым связям. А теперь нужно найти *все* такие перестановки, т.е. все топологические сортировки. Действительно, в данном разделе предполагается, что элементы x и y, для которых определены связи, являются целыми числами между 1 и n, причем $x < y$ всякий раз, когда $x \prec y$. Следовательно, перестановка $12\ldots n$ всегда топологически корректна. (Если это упрощеиное предположение не удовлетворяется, то можно предварительно обработать данные с помощью алгоритма 2.2.3T, чтобы переименовать объекты соответствующим образом.)

Многие важные классы перестановок являются особыми случаями этой задачи топологического упорядочения. Например, перестановки множества $\{1,\ldots,8\}$ со связями

$$1 \prec 2, \quad 2 \prec 3, \quad 3 \prec 4, \quad 6 \prec 7, \quad 7 \prec 8$$

эквивалентны перестановкам мультимножества $\{1,1,1,1,2,3,3,3\}$, поскольку можно использовать отображения $\{1,2,3,4\} \mapsto 1$, $5 \mapsto 2$ и $\{6,7,8\} \mapsto 3$. Нам уже известно, как генерировать перестановки мультимножества с помощью алгоритма L, но сейчас мы рассмотрим другой способ.

Обратите внимание на то, что x предшествует y в перестановке $a_1\ldots a_n$ тогда и только тогда, когда $a'_x < a'_y$ в обратной перестановке $a'_1\ldots a'_n$. Следовательно, рассматриваемый нами алгоритм также найдет все такие перестановки $a'_1\ldots a'_n$, что $a'_j < a'_k$ всякий раз, когда $j \prec k$. Например, из раздела 5.1.4 нам известно, что таблица Юнга является таким упорядочением $\{1,\ldots,n\}$ по строкам и столбцам, что каждая строка возрастает слева направо и каждый столбец возрастает сверху вниз. Следовательно, задача генерации всех таблиц Юнга 3×3 эквивалентна генерации всех таких $a'_1\ldots a'_9$, что

$$\begin{aligned} &a'_1 < a'_2 < a'_3, \quad a'_4 < a'_5 < a'_6, \quad a'_7 < a'_8 < a'_9, \\ &a'_1 < a'_4 < a'_7, \quad a'_2 < a'_5 < a'_8, \quad a'_3 < a'_6 < a'_9, \end{aligned} \tag{48}$$

и представляет собой особый случай топологической сортировки.

Можно было также найти все *сочетания* $2n$ элементов, а именно все варианты разбиения $\{1,\ldots,2n\}$ на n пар. Существует $(2n-1)(2n-3)\ldots(1) = (2n)!/(2^n n!)$ способов сделать это, и они соответствуют перестановкам, которые удовлетворяют условиям

$$a'_1 < a'_2, \quad a'_3 < a'_4, \quad \ldots, \quad a'_{2n-1} < a'_{2n}, \qquad a'_1 < a'_3 < \cdots < a'_{2n-1}. \tag{49}$$

Элегантный алгоритм для исчерпывающей топологической сортировки был предложен Варолем и Ротемом [см. *Comp. J.* **24** (1981), p.83–84], которые сообразили, что можно использовать метод, аналогичный методу простых изменений (алгоритм P). Предположим, что найден способ такого топологического упорядочения $\{1,\ldots,n-1\}$, что $a_1\ldots a_{n-1}$ удовлетворяет всем условиям, которые не включают n. Тогда легко можно записать все допустимые способы вставки последнего элемента n без изменения относительного порядка $a_1\ldots a_{n-1}$. Нужно просто начать с $a_1\ldots a_{n-1}n$, затем сдвигать n влево на один шаг за один раз до тех пор, пока нельзя будет сдвигать дальше. Применяя эту идею рекурсивно, получим следующую прямую процедуру.

Алгоритм V (*все топологические сортировки*). Для заданного отношения $\prec$ на множестве $\{1,\ldots,n\}$, которое обладает следующим свойством, что $x\prec y$ обозначает $x<y$, этот алгоритм генерирует все перестановки $a_1\ldots a_n$ и обратные им перестановки $a'_1\ldots a'_n$, где $a'_j<a'_k$ всякий раз, когда $j\prec k$. Предположим для удобства, что $a_0=0$ и $0\prec k$ для $1\le k\le n$.

V1. [Инициализировать.] Установить $a_j\leftarrow j$ и $a'_j\leftarrow j$ для $0\le j\le n$.

V2. [Посетить.] Посетить перестановку $a_1\ldots a_n$ и обратную ей $a'_1\ldots a'_n$. Затем установить $k\leftarrow n$.

V3. [Можно ли k переместить влево?] Установить $j\leftarrow a'_k$ и $l\leftarrow a_{j-1}$. Если $l\prec k$, перейти к V5.

V4. [Если да, переместить.] Установить $a_{j-1}\leftarrow k$, $a_j\leftarrow l$, $a'_k\leftarrow j-1$ и $a'_l\leftarrow j$. Перейти к V2.

V5. [Если нет, вернуть k назад.] Пока $j<k$, установить $l\leftarrow a_{j+1}$, $a_j\leftarrow l$, $a'_l\leftarrow j$, и $j\leftarrow j+1$. Затем установить $a_k\leftarrow a'_k\leftarrow k$. Уменьшить k на 1 и вернуться к V3, если $k>0$. ∎

Например, согласно теореме 5.1.4H, существует точно 42 таблицы Юнга с размерами 3×3. Если применить алгоритм V для связей (48) и записать обратную перестановку в виде массива

$$\begin{array}{|c|}\hline a'_1\,a'_2\,a'_3\\ a'_4\,a'_5\,a'_6\\ a'_7\,a'_8\,a'_9\\ \hline\end{array}\,, \tag{50}$$

то получим следующие 42 результата:

123 456 789	123 457 689	123 458 679	123 467 589	123 468 579	124 356 789	124 357 689	124 358 679	124 367 589	124 368 579	125 367 489	125 368 479	125 346 789	125 347 689

125 348 679	126 347 589	126 348 579	127 348 569	126 357 489	126 358 479	127 358 469	134 256 789	134 257 689	134 258 679	134 267 589	134 268 579	135 267 489	135 268 479

145 267 389	145 268 379	135 246 789	135 247 689	135 248 679	136 247 589	136 248 579	137 248 569	136 257 489	136 258 479	137 258 469	146 257 389	146 258 379	147 258 369

Пусть t_r является количеством топологических сортировок, для которых последние $n-r$ элементы находятся в их исходном положении $a_j = j$ для $r < j \le n$. Эквивалентно, t_r является количеством топологических сортировок $a_1 \ldots a_r$ для $\{1, \ldots, r\}$, если игнорировать связи, включающие элементы больше r. Тогда рекурсивный механизм алгоритма V показывает, что этап V2 выполняется N раз, а этап V3 выполняется M раз, где

$$M = t_n + \cdots + t_1 \qquad \text{и} \qquad N = t_n. \tag{51}$$

Кроме того, этап V4 и операции организации цикла V5 выполняются $N-1$ раз, а остальные операции этапа V5 выполняются $M - N + 1$ раз. Следовательно, общее время выполнения алгоритма является линейной комбинацией M, N и n.

Если индексы элементов выбраны плохо, то M может оказаться больше N. Например, если для алгоритма V используются ограничения

$$2 \prec 3, \quad 3 \prec 4, \quad \ldots, \quad n-1 \prec n, \tag{52}$$

то $t_j = j$ для $1 \le j \le n$ и имеем $M = \frac{1}{2}(n^2 + n)$, $N = n$. Но эти ограничения также эквивалентны

$$1 \prec 2, \quad 2 \prec 3, \quad \ldots, \quad n-2 \prec n-1, \tag{53}$$

после переименования элементов, и тогда M сводится к случаю $2n - 1 = 2N - 1$.

В упр. 89 показано, что на простом этапе предварительной обработки можно найти такие метки элементов, что небольшая модификация алгоритма V способна генерировать все топологические сортировки за $O(N + n)$ этапов. Таким образом, топологическую сортировку всегда можно выполнить достаточно эффективно.

Семь раз отмерь, потом переставляй. В этом разделе мы ознакомились с несколькими привлекательными алгоритмами для генерации перестановок. Однако известно множество алгоритмов, которые способны генерировать оптимальные перестановки для отдельных задач *без* перебора всех вариантов. Например, в теореме 6.1S показано, что можно найти самый лучший способ упорядочения записей в последовательном хранилище, сортируя их в соответствии с определенными критериями затрат, и для этого требуется всего $O(n \log n)$ этапов. В разделе 7.5.2 будет рассматриваться *задача о назначениях*, или *задача о присваиваниях*, в которой нужно найти такой способ перестановки столбцов квадратной матрицы, чтобы сумма диагональных элементов была максимальна. Эту задачу можно решить не более чем за $O(n^3)$ операций, поэтому было бы глупо использовать метод порядка $n!$, если n не является очень малым. Даже в таких случаях, как в задаче коммивояжера, когда неизвестен эффективный алгоритм, обычно можно найти более эффективный подход, чем простой перебор всех возможных решений. Генерацию перестановок лучше всего использовать в тех случаях, когда есть основательная причина взглянуть на каждую перестановку отдельно.

УПРАЖНЕНИЯ

1. [*20*] Объясните, как ускорить алгоритм L, оптимизируя операции, когда значение j близко к n?

2. [*20*] Перепишите алгоритм L так, чтобы он генерировал все перестановки $a_1 \ldots a_n$ в обратном колексикографическом порядке. (Иначе говоря, значения отражений $a_n \ldots a_1$

должны лексикографически уменьшаться, как в (11). Эта форма алгоритма часто проще и быстрее, чем исходная, поскольку меньшее количество вычислений зависит от значения n.)

3. [*M21*] *Рангом* комбинаторного упорядочения X по отношению к алгоритму генерации называется количество других упорядочений, которые алгоритм посещает до X. Как вычислить ранг заданной перестановки $a_1 \ldots a_n$ по отношению к алгоритму L, если $\{a_1, \ldots, a_n\} = \{1, \ldots, n\}$? Чему равен ранг 314592687?

4. [*M23*] Обобщая упр. 3, объясните, как вычислить ранг $a_1 \ldots a_n$ по отношению к алгоритму L, когда $\{a_1, \ldots, a_n\}$ является мультимножеством $\{n_1 \cdot x_1, \ldots, n_t \cdot x_t\}$? Здесь $n_1 + \cdots + n_t = n$ и $x_1 < \cdots < x_t$. (Общее количество перестановок равно, конечно, полиномиальному коэффициенту

$$\binom{n}{n_1, \ldots, n_t} = \frac{n!}{n_1! \ldots n_t!};$$

см. уравнение 5.1.2–(3).) Чему равен ранг 314159265?

5. [*HM25*] Вычислите среднее и дисперсию количества сравнений, выполняемых алгоритмом L на (a) этапе L2 и (b) этапе L3, когда элементы $\{a_1, \ldots, a_n\}$ различны.

6. [*HM34*] Выведите производящую функцию для среднего числа сравнений, выполняемых алгоритмом L на (a) этапе L2 и (b) этапе L3, когда $\{a_1, \ldots, a_n\}$ является общим мультимножеством, как в упр. 4. Также представьте результаты в замкнутой форме, когда $\{a_1, \ldots, a_n\}$ является двоичным мультмиожеством $\{s \cdot 0, (n-s) \cdot 1\}$.

7. [*HM35*] При $t \to \infty$ чему равно предельное среднее число сравнений на одну перестановку на этапе L2, если алгоритм L применяется к мультимножеству (a) $\{2 \cdot 1, 2 \cdot 2, \ldots, 2 \cdot t\}$? (b) $\{1 \cdot 1, 2 \cdot 2, \ldots, t \cdot t\}$? (c) $\{2 \cdot 1, 4 \cdot 2, \ldots, 2^t \cdot t\}$?

8. [*21*] *Вариациями* мультимножества называют перестановки всех его подчиненных мультимножеств. Например, вариациями $\{1, 2, 2, 3\}$ являются

ϵ, 1, 12, 122, 1223, 123, 1232, 13, 132, 1322,
2, 21, 212, 2123, 213, 2132, 22, 221, 2213, 223, 2231, 23, 231, 2312, 232, 2321,
3, 31, 312, 3122, 32, 321, 3212, 322, 3221.

Покажите, что простые изменения алгоритма L позволяют генерировать все вариации заданного мультимножества $\{a_1, a_2, \ldots, a_n\}$.

9. [*22*] В продолжение предыдущего упражнения создайте алгоритм генерации всех r-вариаций заданного мультимножества $\{a_1, a_2, \ldots, a_n\}$, которые также называются его r-перестановками, а именно все перестановки его r-элементных подчиненных мультимножеств. (Например, решение буквометики с r различными буквами является r-вариацией $\{0, 1, \ldots, 9\}$.)

10. [*20*] Чему будут равны $a_1 a_2 \ldots a_n$, $c_1 c_2 \ldots c_n$, и $o_1 o_2 \ldots o_n$ в конце алгоритма P, если в начале $a_1 a_2 \ldots a_n = 12 \ldots n$?

11. [*M22*] Сколько раз выполняется каждый этап алгоритма P? (Допустим, что $n \geq 2$.)

12. [*M23*] Какой будет 1000000-я перестановка, посещенная (a) алгоритмом L, (b) алгоритмом P, (c) алгоритмом C, если $\{a_1, \ldots, a_n\} = \{0, \ldots, 9\}$? *Подсказка.* в системе обозначений со смешанным основанием имеем:

$$1000000 = \begin{bmatrix} 2, & 6, & 6, & 2, & 5, & 1, & 2, & 2, & 0, & 0 \\ 10, & 9, & 8, & 7, & 6, & 5, & 4, & 3, & 2, & 1 \end{bmatrix} = \begin{bmatrix} 0, & 0, & 1, & 2, & 3, & 0, & 2, & 7, & 1, & 0 \\ 1, & 2, & 3, & 4, & 5, & 6, & 7, & 8, & 9, & 10 \end{bmatrix}.$$

13. [*M21*] (Мартин Гарднер (Martin Gardner), 1974.) Истинно или ложно: если $a_1a_2\dots a_n$ в исходном состоянии имеет вид $12\dots n$, то алгоритм P начинается с посещения всех $n!/2$ перестановок, в которых 1 предшествует 2; тогда следующей перестановкой является $n\dots21$?

14. [*M22*] Истинно или ложно: если $a_1a_2\dots a_n$ в исходном состоянии имеет вид $x_1x_2\dots x_n$ в алгоритме P, то всегда $a_{j-c_j+s}=x_j$ в начале этапа P5?

15. [*M23*] (Селмер Джонсон (Selmer Johnson), 1963.) Покажите, что переменная отступа s никогда не превышает 2 в алгоритме P.

16. [*21*] Как ускорить алгоритм P, оптимизируя его операции, когда значение j близко к n? (Эта задача аналогична упр. 1.)

▸ **17.** [*20*] Расширьте алгоритм P так, чтобы *обратная перестановка* $a'_1\dots a'_n$ была доступна для обработки, когда $a_1\dots a_n$ посещается на этапе P2. (Обратная перестановка удовлетворяет условию $a'_k=j$ тогда и только тогда, когда $a_j=k$.)

18. [*21*] (*Четочные перестановки.*) Найдите эффективный способ генерации перестановок $(n-1)!/2$, которые представляют все возможные неориентированные циклы на вершинах $\{1,\dots,n\}$; т.е. циклический сдвиг $a_1\dots a_n$ или $a_n\dots a_1$ не генерируется, если генерируется $a_1\dots a_n$. Перестановки (1234, 1324, 3124) могут, например, использоваться, когда $n=4$.

19. [*25*] Создайте алгоритм, который генерирует все перестановки n различных элементов *без циклов* в духе алгоритма 7.2.1.1L.

▸ **20.** [*20*] n-куб имеет $2^n n!$ симметрий, по одной для каждого способа перестановки и/или дополнения координат. Такую симметрию удобно представить в виде *перестановки со знаком*, а именно перестановки со вспомогательными знаками, связанными с элементами. Например, $2\bar31$ является перестановкой со знаком, которая преобразует вершины 3-куба с помощью замены $x_1x_2x_3$ на $x_2x_3\bar x_1$ так, что $000\mapsto001$, $001\mapsto011$, $\dots$, $111\mapsto110$. Предложите простой алгоритм, который генерирует все перестановки со знаками для $\{1,2,\dots,n\}$, где на каждом этапе происходит либо замена двух смежных элементов, либо отрицание первого элемента.

21. [*M21*] (Э. П. Мак-Крэйви (E. P. McCravy), 1971.) Сколько решений имеет буквометика (6) с основанием b?

22. [*M15*] Истинно или ложно: если буквометика имеет решение для основания b, то она имеет решение для основания $b+1$?

23. [*M20*] Истинно или ложно: чистая буквометика не может иметь две идентичные сигнатуры $s_j=s_k\neq0$ для $j\neq k$?

24. [*25*] Решите следующие буквометики вручную или с помощью компьютера:

a) SEND + A + TAD + MORE = MONEY.

b) ZEROES + ONES = BINARY. (Питер Мак-Дональд (Peter MacDonald), 1977)

c) DCLIX + DLXVI = MCCXXV. (Вилли Энгрен (Willy Enggren), 1972)

d) COUPLE + COUPLE = QUARTET. (Майкл Бакли (Michael Buckley), 1977)

e) FISH + N + CHIPS = SUPPER. (Роберт Винникомб (Robert Vinnicombe), 1978)

f) SATURN + URANUS + NEPTUNE + PLUTO = PLANETS. (Вилли Энгрен, 1968)

g) EARTH + AIR + FIRE + WATER = NATURE. (Герман Нийон (Herman Nijon), 1977)

h) AN+ACCELERATING+INFERENTIAL+ENGINEERING+TALE+ELITE+GRANT+FEE+ET+CETERA = ARTIFICIAL + INTELLIGENCE.

i) HARDY + NESTS = NASTY + HERDS.

▸ **25.** [*M21*] Предложите быстрый способ вычисления $\min(a\cdot s)$ и $\max(a\cdot s)$ для всех допустимых перестановок $a_1\dots a_{10}$ для $\{0,\dots,9\}$, если для буквометики задан вектор сигнатуры

$s = (s_1, \ldots, s_{10})$ и множество F первых букв. (Такая процедура делает возможным быстрое исключение многих случаев при решении большого количества буквометик, как в нескольких следующих упражнениях, поскольку решение может существовать только тогда, когда $\min(a \cdot s) \le 0 \le \max(a \cdot s)$.)

26. [*25*] Найдите единственное решение буквометики

$$\texttt{NIIHAU} \pm \texttt{KAUAI} \pm \texttt{OAHU} \pm \texttt{MOLOKAI} \pm \texttt{LANAI} \pm \texttt{MAUI} \pm \texttt{HAWAII} = 0?$$

27. [*30*] Создайте чистую буквометику сложения, в которой все слова имеют пять букв.

28. [*M25*] *Разбиением* целого числа n называется выражение $n = n_1 + \cdots + n_t$, где $n_1 \ge \cdots \ge n_t > 0$. Разбиение называется *дважды истинным*, если $\alpha(n) = \alpha(n_1) + \cdots + \alpha(n_t)$ также является чистой буквометикой, где $\alpha(n)$ является "именем" n на некотором языке. Дважды истинные буквометики были введены Аланом Уэйном (Alan Wayne) в работе [*AMM* **54** (1947), 38, p.412–414], в которой он предложил решение для $\texttt{TWENTY} = \texttt{SEVEN} + \texttt{SEVEN} + \texttt{SIX}$ и нескольких других буквометик.

a) Найдите все разбиения, которые являются дважды истинными на английском языке для $1 \le n \le 20$.

b) Уэйн также предложил пример $\texttt{EIGHTY} = \texttt{FIFTY} + \texttt{TWENTY} + \texttt{NINE} + \texttt{ONE}$. С помощью имен ONE, TWO, ..., NINETYNINE, ONEHUNDRED найдите все дважды истинные разбиения для $1 \le n \le 100$ с *различными* слагаемыми.

▸ **29.** [*M25*] В продолжение предыдущего упражнения найдите все уравнения вида $n_1 + \cdots + n_t = n'_1 + \cdots + n'_{t'}$, которые истинны математически и буквометически на английском языке, если $\{n_1, \ldots, n_t, n'_1, \ldots, n'_{t'}\}$ являются различными положительными целыми числами меньше 20. Например:

$$\texttt{TWELVE} + \texttt{NINE} + \texttt{TWO} = \texttt{ELEVEN} + \texttt{SEVEN} + \texttt{FIVE}.$$

Все буквометики должны быть чистыми.

30. [*25*] Решите эти буквометики на умножение вручную или с помощью компьютера:

a) $\texttt{TWO} \times \texttt{TWO} = \texttt{SQUARE}$. (Генри Дьюденн (Henry Dudeney), 1929)

b) $\texttt{HIP} \times \texttt{HIP} = \texttt{HURRAY}$. (Вилли Эигрен, 1970)

c) $\texttt{PI} \times \texttt{R} \times \texttt{R} = \texttt{AREA}$. (Брайан Барвелл (Brian Barwell), 1981)

d) $\texttt{NORTH/SOUTH} = \texttt{EAST/WEST}$. (Ноб Йошигахара (Nob Yoshigahara), 1995)

e) $\texttt{NAUGHT} \times \texttt{NAUGHT} = \texttt{ZERO} \times \texttt{ZERO} \times \texttt{ZERO}$. (Алан Уэйн (Alan Wayne), 2003)

31. [*M22*] (Ноб Йошигахара.) Найдите единственное решение для $\texttt{A/BC} + \texttt{D/EF} + \texttt{G/HI} = 1$, если $\{\texttt{A}, \ldots, \texttt{I}\} = \{1, \ldots, 9\}$?

32. [*M25*] (Генри Дьюденн, 1901.) Найдите все способы представления числа 100 с использованием плюса и дроби в перестановках цифр $\{1, \ldots, 9\}$. Например, $100 = 91 + 5742/638$. Учтите, что плюс всегда должен предшествовать дроби.

33. [*25*] В продолжение предыдущего упражнения найдите все положительные целые числа меньше 150, которые (a) нельзя представить указанным образом; (b) имеют единственное представление.

34. [*M26*] Сделайте уравнение $\texttt{EVEN} + \texttt{ODD} + \texttt{PRIME} = x$ дважды истинным, когда (a) x является совершенной 5-й степенью; (b) x является совершенной 7-й степенью.

▸ **35.** [*M20*] Автоморфизмы 4-куба имеют много разных таблиц Симса, а в (14) показана только одна из них. Сколько разных таблиц Симса может быть, если вершины пропумерованы, как в (12)?

36. [*M23*] Найдите таблицу Симса для группы всех автоморфизмов доски 4×4 для игры в крестики-нолики:

0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

,

т.е. найдите перестановки, которые переводят линию в линию, где под линией подразумевается множество четырех элементов в ряду, в столбце или на диагонали.

▸ **37.** [*HM22*] Сколько таблиц Симса можно использовать с алгоритмом G или H? Оцените логарифм этого количества для $n \to \infty$.

38. [*HM21*] Докажите, что среднее число транспозиций на одну перестановку при использовании алгоритма Орд-Смита (26) приблизительно равно $\sinh 1 \approx 1{,}175$.

39. [*16*] Запишите 24 перестановки, генерируемые для $n = 4$ (a) по методу Орд-Смита (26); (b) по методу Хипа (27).

40. [*M23*] Покажите, что метод Хипа (27) соответствует таблице Снмса.

▸ **41.** [*M33*] Придумайте алгоритм, который генерирует все r-вариации $\{0, 1, \ldots, n-1\}$ заменой только двух элементов при переходе от одной вариации к следующей (см. упр. 9). *Подсказка.* Обобщите метод Хипа (27), получая результаты в позициях $a_{n-r} \ldots a_{n-1}$ множества $a_0 \ldots a_{n-1}$. Например, в одиом решении для $n = 5$ и $r = 2$ используются последние два элемента соответствующих перестановок 01234, 31204, 30214, 30124, 40123, 20143, 24103, 24013, 34012, 14032, 13042, 13402, 23401, 03421, 02431, 02341, 12340, 42310, 41320, 41230.

42. [*M20*] Создайте таблицу Снмса для всех перестановок, в которых каждая $\sigma(k, j)$ и каждая $\tau(k, j)$ для $1 \le j \le k$ является циклом с длиной ≤ 3.

43. [*M24*] Создайте таблицу Симса для всех перестановок, в которых каждая $\sigma(k, k)$, $\omega(k)$ и $\tau(k, j)\omega(k-1)^-$ для $1 \le j \le k$ является циклом с длиной ≤ 3.

44. [*20*] Если блоки ненужных перестановок пропускаются расширенным алгоритмом G, то будет ли таблица Симса в методе Орд-Смита (23) лучше таблицы Симса в методе для обратного колексикографического порядка?

45. [*20*] (a) Чему равны индексы $u_1 \ldots u_9$, когда алгоритм X посещает перестановку 314592687? (b) Какая перестановка посещается, когда $u_1 \ldots u_9 = 314157700$?

46. [*20*] Истинно или ложно: когда алгоритм X посещает $a_1 \ldots a_n$, имеем $u_k > u_{k+1}$ тогда и только тогда, когда $a_k > a_{k+1}$ для $1 \le k < n$.

▸ **47.** [*M21*] Подсчитайте количество выполнений каждого этапа алгоритма X и выразите их с помощью чисел N_0, N_1, ..., N_n, где N_k — количество префиксов $a_1 \ldots a_k$, которые удовлетворяют $t_j(a_1, \ldots, a_j)$ для $1 \le j \le k$.

▸ **48.** [*M25*] Сравните время выполнения алгоритма X и алгоритма L в случаях, когда проверки $t_1(a_1)$, $t_2(a_1, a_2)$, ..., $t_n(a_1, a_2, \ldots, a_n)$ всегда дают истинный результат.

▸ **49.** [*28*] Предложенный в этом разделе метод решения буквометик сложения с помощью алгоритма X по сути выбирает цифры справа налево. Иначе говоря, он присваивает пробные значения наименее значимым цифрам до рассмотрения цифр, которые соответствуют более высоким степеням 10.

Исследуйте альтериативный подход, в котором цифры выбираются слева направо. Например, с помощью такого метода можно сразу определить, что `M` $= 1$ в буквометике `SEND` $+$ `MORE` $=$ `MONEY`. *Подсказка: см.* упр. 25.

50. [*M15*] Объясните, почему двойственная формула (32) следует из (13)?

51. [*M16*] Истинно или ложно: если множества $S_k = \{\sigma(k,0), \ldots, \sigma(k,k)\}$ образуют таблицу Симса для группы всех перестановок, то также ведут себя множества $S_k^- = \{\sigma(k,0)^-, \ldots, \sigma(k,k)^-\}$?

▸ **52.** [*M22*] Какие перестановки $\tau(k,j)$ и $\omega(k)$ возникают, когда алгоритм H используется с таблицей Симса (36)? Сравните полученный генератор с алгоритмом P.

▸ **53.** [*M26*] (Ф. М. Ивс (F. M. Ives).) Создайте таблицу Симса, для которой алгоритм H генерирует все перестановки за счет всего $n! + O((n-2)!)$ транспозиций.

54. [*20*] Будет ли алгоритм C работать, если на этапе C3 выполнить правый циклический сдвиг, задавая $a_1 \ldots a_{k-1}a_k \leftarrow a_k a_1 \ldots a_{k-1}$ вместо левого циклического сдвига?

55. [*M27*] Рассмотрим *факториальную масштабную функцию*

$$\rho_!(m) = \max\{k \mid m \bmod k! = 0\}.$$

Пусть σ_k и τ_k являются перестановками таких неотрицательных целых чисел, что $\sigma_j\tau_k = \tau_k\sigma_j$ для $j \le k$. Пусть α_0 и β_0 являются тождественными перестановками, а для $m > 0$ определим

$$\alpha_m = \beta_{m-1}^-\tau_{\rho_!(m)}\beta_{m-1}\alpha_{m-1}, \qquad \beta_m = \sigma_{\rho_!(m)}\beta_{m-1}.$$

Например, если σ_k является операцией переворота $(1\ k{-}1)(2\ k{-}2)\ldots = (0\ k)\phi(k)$ и если $\tau_k = (0\ k)$, и если алгоритм E начинается с $a_j = j$ для $0 \le j < n$, тогда α_m и β_m образуют содержимое $a_0 \ldots a_{n-1}$ и $b_0 \ldots b_{n-1}$ после выполнения m раз этапа E5.

a) Докажите, что $\beta_{(n+1)!}\alpha_{(n+1)!} = \sigma_{n+1}\sigma_n^-\tau_{n+1}\tau_n^-(\beta_n\alpha_n)^{n+1}$.

b) Используйте результат (a), чтобы установить истинность алгоритма E.

56. [*M22*] Докажите, что алгоритм E остается истинным, если этап E5 заменить следующим.

E5′. [Транспоиировать пары.] Если $k > 2$, заменить $b_{j+1} \leftrightarrow b_j$ для $j = k-2, k-4, \ldots,$ (2 или 1). Вернуться к E2. ∎

57. [*HM22*] Чему равно среднее колнчество замен на этапе E5?

58. [*M21*] Истинно или ложно: если алгоритм E начинается с $a_0 \ldots a_{n-1} = x_1 \ldots x_n$, то последняя посещаемая перестановка начинается с $a_0 = x_n$.

59. [*M20*] Некоторые авторы определяют ребра графа Кэли от π к $\pi\alpha_j$, а не от π к $\alpha_j\pi$. Насколько сильно отличаются эти два определения?

▸ **60.** [*21*] *Циклом Грэя для перестановок* называется цикл $(\pi_0, \pi_1, \ldots, \pi_{n!-1})$, который включает каждую перестановку $\{1, 2, \ldots, n\}$ и обладает таким свойством, что π_k отличается от $\pi_{(k+1) \bmod n!}$. Он также может быть описан как гамилтьтонов цикл на графе Кэли для группы всех перестановок для $\{1, 2, \ldots, n\}$ с $n-1$ генераторами $((1\ 2), (2\ 3), \ldots, (n{-}1\ n))$. *Дельта-последовательностью* такого цикла Грэя является такая последовательность целых чисел $\delta_0\delta_1 \ldots \delta_{n!-1}$, что

$$\pi_{(k+1) \bmod n!} = (\delta_k\ \delta_k{+}1)\,\pi_k.$$

(См. 7.2.1.1–(24), где описывается аналогичная ситуация для двоичных n-кортежей.) Например, на рис. 23 показан цикл Грэя, заданный простыми изменениями для $n = 4$. Его дельта-последовательность имеет вид $(32131231)^3$.

a) Найдите все циклы Грэя для перестановок $\{1, 2, 3, 4\}$.

b) Два цикла Грэя считаются эквивалентными, если их дельта-последовательности можно получить друг из друга циклическим сдвигом $(\delta_k \ldots \delta_{n!-1}\delta_0 \ldots \delta_{k-1})$, и/или обращением $(\delta_{n!-1} \ldots \delta_1\delta_0)$, и/или дополнением $((n{-}\delta_0)(n{-}\delta_1)\ldots(n{-}\delta_{n!-1}))$. Какие циклы Грэя в (a) эквивалентны друг другу?

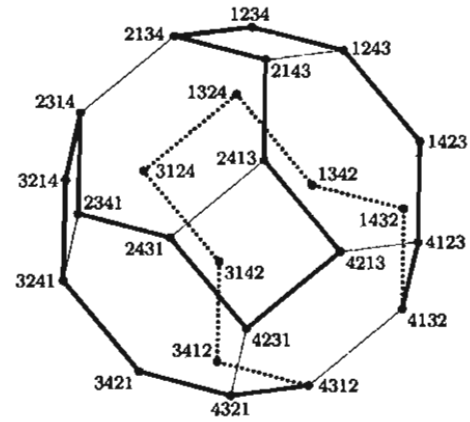


Рис. 23. Алгоритм P проходит этот гамильтонов цикл на усеченном октаэдре с рис. 5–1

61. [*21*] В продолжение предыдущего упражнения *код Грэя для перестановок* аналогичен циклу Грэя, за исключением того, что последняя перестановка $\pi_{n!-1}$ необязательно должна быть смежной для исходной перестановкн π_0. Изучите множество всех кодов Грэя для $n = 4$, которые начинаются с 1234.

▸ **62.** [*M23*] Какие перестановки могут быть последним элементом кода Грэя, который начинается с $12\ldots n$?

63. [*M25*] Оцените общее количество циклов Грэя для перестановок $\{1, 2, 3, 4, 5\}$.

64. [*23*] Дважды кодом Грэя для перестановок называется цикл Грэя с дополнительным свойством, что $\delta_{k+1} = \delta_k \pm 1$ для всех k. Р. К. Комптон и С. Дж. Вильямсон доказали, что такие коды существуют для всех $n \geq 3$. Сколько дважды кодов Грэя существует для $n = 5$?

65. [*M25*] Для каких целочисленных N существует путь Грэя по N лексикографически наименьшим перестановкам $\{1, \ldots, n\}$? (В упр. 7.2.1.1-26 решается аналогичная задача для двоичных n-кортежей.)

66. [*22*] Из метода обмена Эрлиха следует другой тип цикла Грэя для перестановок, в котором $n - 1$ генераторов являются звездными транспозициями (1 2), (1 3), ..., (1 n). Например, на рис. 24 показан соответствующий граф для $n = 4$. Проанализируйте гамильтоновы циклы этого графа.

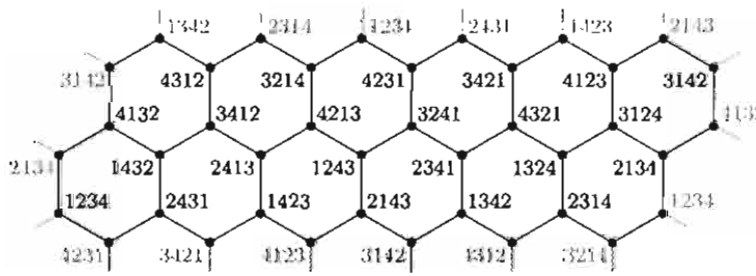


Рис. 24. Граф Кэли для перестановок $\{1, 2, 3, 4\}$, генерированный звездными транспозициями (1 2), (1 3) и (1 4), изображенными в виде скрученного тора

67. [*26*] В продолжение предыдущего упражнения найдите цикл Грэя с обменом первыми элементами для $n = 5$, в котором каждая звездная транспозиция (1 j) происходит 30 раз для $2 \leq j \leq 5$.

68. [*M30*] (В. Л. Компельмахер и В. А. Лисковец, 1975.) Пусть G является графом Кэли для всех перестановок $\{1,\ldots,n\}$ с генераторами $(\alpha_1,\ldots,\alpha_k)$, где каждый генератор α_j является транспозицией $(u_j\ v_j)$. Также пусть A является графом с вершинами $\{1,\ldots,n\}$ и ребрами u_j — v_j для $1\le j\le k$. Докажите, что G имеет гамильтонов цикл тогда и только тогда, когда A является связным. (На рис. 23 показан особый случай, когда A является путем, а на рис. 24 показан особый случай, когда A является звездным графом.)

▸ **69.** [*28*] Если $n\ge 4$, то следующий алгоритм генерирует все перестановки $A_1A_2A_3\ldots A_n$ of $\{1,2,3,\ldots,n\}$ с помощью только трех преобразований:

$$\rho=(1\,2)(3\,4)(5\,6)\ldots,\qquad \sigma=(2\,3)(4\,5)(6\,7)\ldots,\qquad \tau=(3\,4)(5\,6)(7\,8)\ldots,$$

причем ρ и τ никогда не применяются вслед друг за другом. Объясните принцип работы этого алгоритма.

Z1. [Инициализировать.] Установить $A_j\leftarrow j$ для $1\le j\le n$. Также установить $a_j\leftarrow 2j$ для $j\le n/2$ и $a_{n-j}\leftarrow 2j+1$ для $j<n/2$. Затем вызвать алгоритм P, но с параметром $n-1$ вместо n. Рассмотрим этот алгоритм как сопрограмму, которая должна возвращать управление всякий раз при посещении $a_1\ldots a_{n-1}$ на этапе P2. В нем используются те же переменные, за исключением n.

Z2. [Установить x и y.] Снова вызвать алгоритм P, получая новую перестановку $a_1\ldots a_{n-1}$ и новое значение j. Если $j=2$, заменить $a_{1+s}\leftrightarrow a_{2+s}$ (таким образом отменяя результат этапа P5) и повторить этот этап. В этом случае мы находимся посередине алгоритма P. Если $j=1$ (так, что алгоритм P завершает работу), установить $x\leftarrow y\leftarrow 0$ и перейти к Z3. В противном случае установить

$$x\leftarrow a_{j-c_j+s+[o_j=-1]},\qquad y\leftarrow a_{j-c_j+s-[o_j=+1]};$$

именно эти два элемента заменяются наиболее часто на этапе P5.

Z3. [Посетить.] Посетить перестановку $A_1\ldots A_n$. Затем перейти к Z5, если $A_1=x$ и $A_2=y$.

Z4. [Применить ρ, потом σ.] Заменить $A_1\leftrightarrow A_2$, $A_3\leftrightarrow A_4$, $A_5\leftrightarrow A_6$, ... Посетить $A_1\ldots A_n$. Потом заменить $A_2\leftrightarrow A_3$, $A_4\leftrightarrow A_5$, $A_6\leftrightarrow A_7$, ... Завершить, если $A_1\ldots A_n=1\ldots n$, в противном случае вернуться к Z3.

Z5. [Применить τ, потом σ.] Заменить $A_3\leftrightarrow A_4$, $A_5\leftrightarrow A_6$, $A_7\leftrightarrow A_8$, ... Посетить $A_1\ldots A_n$. Потом заменить $A_2\leftrightarrow A_3$, $A_4\leftrightarrow A_5$, $A_6\leftrightarrow A_7$, ..., и вернуться к Z2. ▮

Подсказка: сначала покажите, что алгоритм работает, если он изменен так, что $A_j\leftarrow n+1-j$ и $a_j\leftarrow j$ на этапе Z1, и если перестановки "переброса"

$$\rho'=(1\ n)(2\ n{-}1)\ldots,\qquad \sigma'=(2\ n)(3\ n{-}1)\ldots,\qquad \tau'=(2\ n{-}1)(3\ n{-}2)\ldots$$

используются вместо ρ, σ, τ на этапах Z4 и Z5. В этой модификации на этапе Z3 происходит переход к этапу Z5, если $A_1=x$ и $A_n=y$.

▸ **70.** [*M33*] Два 12-цикла (41) можно рассматривать как циклы σ–τ для 12 перестановок $\{1,1,3,4\}$:

$$\begin{aligned}1134\to 1341\to 3411\to 4311\to 3114\to 1143\to 1431\\ \to 4131\to 1314\to 3141\to 1413\to 4113\to 1134.\end{aligned}$$

Заменяя $\{1,1\}$ на $\{1,2\}$, получим непересекающиеся циклы, а с помощью перехода от одного к другому — также гамильтонов путь. Можно ли аналогичным образом получить путь σ–τ для всех перестановок шести элементов на основе 360-цикла для перестановок $\{1,1,3,4,5,6\}$?

71. [*48*] Будет ли граф Кэли с генераторами $\sigma=(1\,2\ldots n)$ и $\tau=(1\,2)$ иметь гамильтонов цикл для нечетных $n\ge 3$?

72. [*M21*] Для графа Кэли с генераторами $(\alpha_1, \ldots, \alpha_k)$ предположим, что каждый генератор α_j преобразует $x \mapsto y$. (Например, σ и τ в упр. 71 преобразуют $1 \mapsto 2$.) Докажите, что любой гамильтонов путь, начинающийся с $12 \ldots n$ в G, должен заканчиваться перестановкой, которая преобразует $y \mapsto x$.

▸ **73.** [*M30*] Пусть α, β и σ являются перестановками множества X, где $X = A \cup B$. Допустим, что $x\sigma = x\alpha$, когда $x \in A$, и $x\sigma = x\beta$, когда $x \in B$, а порядок $\alpha\beta^-$ нечетный.

a) Докажите, что все три перестановки, α, β, σ, имеют одинаковый знак, т.е. все они четны или все нечетны. *Подсказка.* Перестановка имеет нечетный порядок тогда и только тогда, когда все ее циклы имеют нечетную длину.

b) Выведите теорему R из (a).

74. [*M30*] (P. A. Ранкин.) Предполагая, что $\alpha\beta = \beta\alpha$ в теореме R, докажите, что гамильтонов цикл существует тогда и только тогда, когда существует такое число k, что $0 \le k \le g/c$ и $t + k \perp c$, где $\beta^{g/c} = \gamma^t$, $\gamma = \alpha\beta^-$. *Подсказка.* Представьте элементы группы в форме $\beta^j\gamma^k$.

75. [*M25*] Ориентированный тор $O_m \times O_n$ имеет mn вершин (x, y) для $0 \le x < m$, $0 \le y < n$ и ребра $(x, y) \to (x, y)\alpha = ((x + 1) \bmod m, y)$, $(x, y) \to (x, y)\beta = (x, (y + 1) \bmod n)$. Докажите, что если $m > 1$ и $n > 1$, то количество гамильтоновых циклов этого диграфа равно:

$$\sum_{k=1}^{d-1} \binom{d}{k} [\gcd((d-k)m, kn) = d], \qquad d = \gcd(m, n).$$

76. [*M31*] Клетки, пронумерованные числами 0, 1, ..., 63 на рис. 25, иллюстрируют *северо-восточный проход конем* на торе 8×8: если k находится в клетке (x_k, y_k), то $(x_{k+1}, y_{k+1}) \equiv (x_k + 2, y_k + 1)$ или $(x_k + 1, y_k + 2)$ по модулю 8, а $(x_{64}, y_{64}) = (x_0, y_0)$. Сколько таких проходов может быть на $m \times n$ торе, для $m, n \ge 3$?

29	24	19	14	49	44	39	34
58	53	48	43	38	9	4	63
23	18	13	8	3	62	33	28
52	47	42	37	32	27	22	57
17	12	7	2	61	56	51	46
6	41	36	31	26	21	16	11
35	30	1	60	55	50	45	40
0	59	54	25	20	15	10	5

Рис. 25. Северо-восточный проход конем

▸ **77.** [*22*] Создайте полную программу `MMIX`, внутренний цикл которой представлен в (42) с помощью метода Хипа (27).

78. [*M23*] Проанализируйте время выполнения программы в упр. 77, обобщая ее так, чтобы во внутреннем цикле выполнялось $r!$ посещений (с $a_0 \ldots a_{r-1}$ в глобальных регистрах).

79. [*20*] Какие семь инструкций `MMIX` реализуют этап ⟨ Обменять ниблы ... ⟩ в (45)? Например, если регистр `t` содержит значение 4 и регистр `a` содержит ниблы `#12345678`, то регистр `a` должен измениться на `#12345687`.

80. [*21*] Решите предыдущее упражнение с помощью только пяти `MMIX` инструкций. *Подсказка.* Используйте `MXOR`.

▸ **81.** [*22*] Завершите программу `MMIX` для (46), предлагая способ выполнения последнего этапа ⟨ Продолжить работу методом Лангдона. ⟩.

82. [*M21*] Проанализируйте время выполнения программы в упр. 81.

83. [*22*] Используйте путь σ–τ из упр. 70 для создания процедуры `MMIX`, аналогичной (42), которая генерирует все перестановки `#123456` в регистре `a`.

84. [*20*] Предложите способ генерации всех $n!$ перестановок $\{1,\ldots,n\}$, выполняемых p процессорами в параллельном режиме работы. .

▸ **85.** [*25*] Допустим, что значение n настолько мало, что $n!$ помещается в рамках компьютерного слова. Предложите эффективный способ преобразования заданной перестановки $\alpha = a_1 \ldots a_n$ множества $\{1,\ldots,n\}$ в целочисленную функцию $k = r(\alpha)$ в диапазоне $0 \le k < n!$? Обе функции, $k = r(\alpha)$ и $\alpha = r^{[-1]}(k)$, должны вычисляться всего за $O(n)$ этапов.

86. [*20*] Отношение частичного порядка предполагается транзитивным, т.е. из $x \prec y$ и $y \prec z$ следует $x \prec z$. Но для удовлетворения этого условия в алгоритме V не требуется этого исходного отношения.

Покажите, что если $x \prec y$ и $y \prec z$, то алгоритм V дает идентичные результаты, независимо от того, выполняется ли условие $x \prec z$ или нет.

87. [*20*] (Ф. Раски (F. Ruskey).) Рассмотрим таблицы инверсии $c_1 \ldots c_n$ перестановок, посещенных алгоритмом V. Каким замечательным свойством они обладают? (Сравните с таблицами инверсии из (4) в алгоритме P.)

88. [*21*] Покажите, что алгоритм V можно использовать для генерации всех способов разбиения цифр $\{0,1,\ldots,9\}$ на два 3-элементных и два 2-элементных множества.

▸ **89.** [*M30*] Рассмотрим числа t_0, t_1, ..., t_n в (51). Ясно, что $t_0 = t_1 = 1$.

a) Назовем индекс j тривиальным, если $t_j = t_{j-1}$. Например, 9 является тривиальным для отношений в таблице Юнга (48). Как можно было бы изменить алгоритм V, чтобы переменная k принимала только нетривиальные значения.

b) Проанализируйте время выполнения модифицированного алгоритма. Какие формулы заменят (51)?

c) Скажем, что интервал $[j\,..\,k]$ не является цепью, если не выполняется $l \prec l+1$ для $j \le l < k$. Докажите, что в таком случае $t_k \ge 2t_{j-1}$.

d) Каждая обратная топологическая сортировка $a'_1 \ldots a'_n$ задает маркировку, которая соответствует отношениям $a'_{j_1} \prec a'_{k_1}$, ..., $a'_{j_m} \prec a'_{k_m}$, эквивалентным исходным отношениям $j_1 \prec k_1$, ..., $j_m \prec k_m$. Объясните, как найти такую маркировку, что $[j\,..\,k]$ не является цепью, когда j и k являются последовательными нетривиальными индексами.

e) Докажите, что с помощью такой маркировки $M < 4N$ в формулах из п. (b).

90. [*M21*] Алгоритм V можно использовать для генерации всех перестановок, которые h-упорядочены для всех h в заданном множестве, а именно для всех таких $a'_1 \ldots a'_n$, что $a'_j < a'_{j+h}$ для $1 \le j \le n-h$ (см. раздел 5.2.1). Проанализируйте время выполнения алгоритма V, когда он генерирует все перестановки, которые одновременно 2-упорядочены и 3-упорядочены.

91. [*HM21*] Проанализируйте время выполнения алгоритма V, когда он используется с условиями (49) для поиска сочетаний.

92. [*M18*] Сколько перестановок вероятно посетит алгоритм V "случайным" образом? Пусть P_n является количеством частичных упорядочений $\{1,\ldots,n\}$, а именно количеством рефлексивных, антисимметричных и транзитивных отношений. Пусть Q_n является количеством таких отношений с дополнительным свойством $j < k$ для $j \prec k$. На основе P_n и Q_n выразите ожидаемое количество способов топологической сортировки n элементов, усредненное по всем частичным упорядочениям.

93. [*35*] Докажите, что все топологические сортировки можно генерировать таким образом, что только одна или две смежные транспозиции выполняются на каждом этапе. (Пример $1 \prec 2$, $3 \prec 4$ показывает, что одна транспозиция на этап не всегда может быть достигнута, даже если допустить несмежные обмены, поскольку только две из шести подходящих перестановок являются нечетными.)

▸ **94.** [*25*] Покажите, что в случае с сочетаниями с помощью отношений в (49) все топологические сортировки можно генерировать всего с одной транспозицией на каждом этапе.

95. [*21*] Как генерировать все *перестановки вверх-вниз* $\{1,\ldots,n\}$, т.е. такие перестановки $a_1\ldots a_n$, в которых $a_1 < a_2 > a_3 < a_4 > \cdots$.

96. [*21*] Как генерировать все *циклические перестановки* $\{1,\ldots,n\}$, т.е. такие перестановки $a_1\ldots a_n$, циклическое представление которых состоит из одного n-цикла.

97. [*21*] Как генерировать все *неупорядочения* $\{1,\ldots,n\}$, т.е. такие перестановки $a_1\ldots a_n$, для которых $a_1\neq 1$, $a_2\neq 2$, $a_3\neq 3$, ...

98. [*HM23*] Проанализируйте асимптотическое время выполнения метода из предыдущего упражнения.

99. [*M30*] Для заданного $n\geq 3$ покажите, что все неупорядочения $\{1,\ldots,n\}$ можно генерировать, выполняя не больше двух транспозиций между посещениями.

100. [*21*] Как генерировать все *неразложимые* перестановки $\{1,\ldots,n\}$, т.е. такие перестановки $a_1\ldots a_n$, что $\{a_1,\ldots,a_j\}\neq\{1,\ldots,j\}$ для $1\leq j<n$.

101. [*21*] Как генерировать все *инволюции* $\{1,\ldots,n\}$, т.е. такие перестановки $a_1\ldots a_n$, для которых $a_{a_1}\ldots a_{a_n}=1\ldots n$.

102. [*M30*] Покажите, что все инволюции $\{1,\ldots,n\}$ можно генерировать за счет не более двух транспозиций между посещениями.

103. [*M32*] Покажите, что все четные перестановки $\{1,\ldots,n\}$ можно генерировать последовательными *поворотами трех последовательных элементов*.

▸ **104.** [*M22*] Перестановка $a_1\ldots a_n$ of $\{1,\ldots,n\}$ называется *хорошо сбалансированной*, если

$$\sum_{k=1}^{n} ka_k=\sum_{k=1}^{n}(n+1-k)a_k.$$

Например, 3142 является хорошо сбалансированной для $n=4$.

a) Докажите, что ни одна перестановка не может быть хорошо сбалансированной для $n \bmod 4=2$.

b) Докажите, что если $a_1\ldots a_n$ является хорошо сбалансированной, то таковыми являются обратная ей перестановка $a_n\ldots a_1$, ее дополнение $(n+1-a_1)\ldots(n+1-a_n)$ и инверсиая ей перестановка $a'_1\ldots a'_n$.

c) Определите количество хорошо сбалансированных перестановок для малых значений n.

▸ **105.** [*26*] *Слабым порядком* называется отношение $\preceq$, которое является транзитивным ($x\preceq y$ и $y\preceq z$ подразумевает $x\preceq z$) и полным (всегда выполняется $x\preceq y$ или $y\preceq x$). Можно записать $x\equiv y$, если $x\preceq y$ и $y\preceq x$; $x\prec y$, если $x\preceq y$ и $y\not\preceq x$. Для трех элементов $\{1,2,3\}$ существует 13 слабых порядков, а именно:

$$1\equiv 2\equiv 3,\quad 1\equiv 2\prec 3,\quad 1\prec 2\equiv 3,\quad 1\prec 2\prec 3,\quad 1\equiv 3\prec 2,\quad 1\prec 3\prec 2,$$
$$2\prec 1\equiv 3,\quad 2\prec 1\prec 3,\quad 2\equiv 3\prec 1,\quad 2\prec 3\prec 1,\quad 3\prec 1\equiv 2,\quad 3\prec 1\prec 2,\quad 3\prec 2\prec 1.$$

a) Как систематично генерировать все слабые порядки $\{1,\ldots,n\}$, как последовательности цифр, разделенных символом $\equiv$ или $\prec$.

b) Слабый порядок можно также представить в виде последовательности $a_1\ldots a_n$, где $a_j=k$, если j предшествуют k знаков $\prec$. Например, 13 слабых порядков $\{1,2,3\}$ равны соответственно 000, 001, 011, 012, 010, 021, 101, 102, 100, 201, 110, 120, 210 в этой форме. Найдите простой способ генерации всех таких последовательностей длины n.

106. [*M40*] Можно ли решить упр. 105,*b* с помощью кода, подобного коду Грэя?

▸ **107.** [*30*] (Джон Х. Конвэй, 1973.) Пасьянс "топсвопс" начинается с тасовки колоды из n карт, помеченных $\{1,\ldots,n\}$, и размещения их лицевой стороной вверх. Тогда, если верхняя карта $k>1$, снять верхние k карт по одной и положить их снова наверх колоды, таким образом заменяя перестановку $a_1\ldots a_n$ перестановкой $a_k\ldots a_1a_{k+1}\ldots a_n$. Продолжать, пока верхней картой не будет 1. Например, 7-этапная последовательность

$$31452 \to 41352 \to 53142 \to 24135 \to 42135 \to 31245 \to 21345 \to 12345$$

может произойти при $n=5$. Какой будет самая длинная возможная последовательность для $n=13$?

108. [*M27*] Докажите, что если самый длинный розыгрыш пасьянса "топсвопс" с n картами имеет длину $f(n)$, то $f(n)\le F_{n+1}-1$.

109. [*M47*] Найдите верхнюю и нижнюю границы для функции $f(n)$ в пасьянсе "топсвопс".

▸ **110.** [*25*] Найдите все такие перестановки $a_0\ldots a_9$ of $\{0,\ldots,9\}$, что

$$\begin{aligned}\{a_0,a_2,a_3,a_7\}&=\{2,5,7,8\},\\ \{a_1,a_4,a_5\}&=\{0,3,6\},\\ \{a_1,a_3,a_7,a_8\}&=\{3,4,5,7\},\\ \{a_0,a_3,a_4\}&=\{0,7,8\}.\end{aligned}$$

Также предложите алгоритм для решения более крупных задач этого типа.

▸ **111.** [*M25*] Известно несколько аналогов цикла де Бруийна, которые ориентированы на перестановки. Самый простой и самый красивый из них называется *универсальным циклом перестановок* и был предложен Б. В. Джексоном в *Discrete Math.* **117** (1993), p.141–150. Он представляет собой цикл из $n!$ таких цифр, что каждая перестановка $\{1,\ldots,n\}$ возникает лишь однажды в виде блока $n-1$ последовательных цифр (с избыточным финальным элементом). Например, (121323) является универсальным циклом перестановок для $n=3$, причем единственным таким циклом.

Найдите универсальный цикл перестановок для $n=4$ и докажите, что такие циклы существуют для всех $n\ge 2$.

▸ **112.** [*HM43*] Сколько существует универсальных циклов для перестановок ≤ 9 объектов?

ОТВЕТЫ К УПРАЖНЕНИЯМ

РАЗДЕЛ 7.2.1.1

1. Пусть $m_j = u_j - l_j + 1$, и посетим $(a_1 + l_1, \ldots, a_n + l_n)$ вместо $(a_1, \ldots, a_n)$ в алгоритме M. Или заменим '$a_j \leftarrow 0$' на '$a_j \leftarrow l_j$' и '$a_j = m_j - 1$' на '$a_j = u_j$' в этом алгоритме и установим $l_0 \leftarrow 0$, $u_0 \leftarrow 1$ на этапе M1.

2. $(0, 0, 1, 2, 3, 0, 2, 7, 0, 9)$.

3. Этап M4 выполняется $m_1 m_2 \ldots m_k$ раз, если $j = k$; следовательно, общее количество равно $\sum_{k=0}^{n} \prod_{j=1}^{k} m_j = m_1 \ldots m_n (1 + 1/m_n + 1/m_n m_{n-1} + \cdots + 1/m_n \ldots m_1)$. Если все m_j равны или больше 2, то оно меньше $2m_1 \ldots m_n$. (Таким образом, следует иметь в виду, что необычные методы на основе кода Грэя, которые изменяют только одну цифру за посещение, фактически сокращают общее число изменений цифр, но максимум в два раза.)

4. **N1.** [Инициализировать.] Установить $a_j \leftarrow m_j - 1$ для $0 \le j \le n$, где $m_0 = 2$.

N2. [Посетить.] Посетить n-кортеж $(a_1, \ldots, a_n)$.

N3. [Подготовить к вычитанию единицы.] Установить $j \leftarrow n$.

N4. [Заем в случае необходимости.] Если $a_j = 0$, то установить $a_j \leftarrow m_j - 1$, $j \leftarrow j - 1$, а потом повторить этот этап.

N5. [Уменьшить, если это еще не сделано.] Если $j = 0$, то завершить этот алгоритм. В противном случае установить $a_j \leftarrow a_j - 1$ и вернуться к этапу N2. ▮

5. Обращение битов легко выполняется на компьютерах типа MMIX, но на других компьютерах можно поступить следующим образом:

R1. [Инициализнровать.] Установить $j \leftarrow k \leftarrow 0$.

R2. [Переставнть.] Поменять $A[j + 1] \leftrightarrow A[k + 2^{n-1}]$. Также, если $j < k$, поменять $A[j] \leftrightarrow A[k]$ и $A[j + 2^{n-1} + 1] \leftrightarrow A[k + 2^{n-1} + 1]$.

R3. [Продвинуть k.] Установить $k \leftarrow k + 2$ и завершить, если $k \ge 2^{n-1}$.

R4. [Продвинуть j.] Установить $h \leftarrow 2^{n-2}$. Если $j \ge h$, повторно устанавливать $j \leftarrow j - h$ и $h \leftarrow h/2$ до тех пор, пока не будет выполняться условие $j < h$. Затем установить $j \leftarrow j + h$. (Теперь $j = (b_0 \ldots b_{n-1})_2$, если $k = (b_{n-1} \ldots b_0)_2$.) Вернуться к этапу R2. ▮

6. Если $g((0b_{n-1} \ldots b_1 b_0)_2) = (0(b_{n-1}) \ldots (b_2 \oplus b_1)(b_1 \oplus b_0))_2$, то $g((1b_{n-1} \ldots b_1 b_0)_2) = 2^n + g((0\bar{b}_{n-1} \ldots \bar{b}_1 \bar{b}_0)_2) = (1(\bar{b}_{n-1}) \ldots (\bar{b}_2 \oplus \bar{b}_1)(\bar{b}_1 \oplus \bar{b}_0))_2$, где $\bar{b} = b \oplus 1$.

7. Для $2r$ секторов можно использовать $g(k)$ для $2^n - r \le k < 2^n + r$, где $n = \lceil \lg r \rceil$, поскольку $g(2^n - r) \oplus g(2^n + r - 1) = 2^n$ by (5). [G. C. Tootill, *Proc. IEE* **103**, Part B Supplement (1956), p.434.] См. также упр. 26.

8. Используйте алгоритм G с $n \leftarrow n - 1$ и включите бит четности a_∞ справа. (Таким образом, можно получить $g(0)$, $g(2)$, $g(4)$, ...)

9. Замените самое правое кольцо, поскольку $\nu(1011000)$ нечетно.

10. $A_n + B_n = g^{[-1]}(2^n - 1) = \lfloor 2^{n+1}/3 \rfloor$ и $A_n = B_n + n$. Следовательно, $A_n = \lfloor 2^n/3 + n/2 \rfloor$ и $B_n = \lfloor 2^n/3 - n/2 \rfloor$.

Историческое замечание. Японский математик Йориюки Арима (1714–1783) рассмотрел эту задачу [см. *Shūki Sanpō* (1769), Problem 44], заметив, что головоломка с n-кольцами сводится к головоломке с $(n-1)$-кольцами после некоторго числа шагов. Пусть $C_n = A_n - A_{n-1} = B_n - B_{n-1} + 1$ является числом колец, снятых во время этого упрощения задачи. Арима заметил, что $C_n = 2C_{n-1} - [n \text{ четно}]$. Таким образом, он мог бы вычислить $A_n = C_1 + C_2 + \cdots + C_n$ для $n = 9$, фактически не зная формулы $C_n = \lceil 2^{n-1}/3 \rceil$.

Больше чем двести лет назад Кардан уже упоминал "complicati annuli" [*De Subtilitate Libri XXI* (Nuremberg: 1550), Book 15]. Он считал их "бесполезными и даже поразительно неуловимыми", ошибочно заявляя, что достаточно 95 шагов для снятия семи колец и еще 95 для их возврата в исходное положение. Джон Валлис (John Wallis) посвятил этой головоломке семь страниц в латинском издании своей книги *Algebra* **2** (Oxford: 1693), Chapter 111. Он представил весьма подробное описание не очень эффективиого метода для случая с 9 кольцами. Он включил операции сдвига кольца по стержню, а также снятие и одевание колец и намекнул, что возможны более эффективные методы, но не предлагал их.

11. Решение имеет вид $S_n = S_{n-2} + 1 + S_{n-2} + S_{n-1}$, если $S_1 = S_2 = 1$ is $S_n = 2^{n-1} - [n \text{ четно}]$. [*Math. Quest. Educational Times* **3** (1865), p.66–67.]

12. (a) Согласно теории, $n-1$ двоичный код Грэя позволяет получить композиции в удобном порядке (4, 31, 211, 22, 112, 1111, 121, 13):

A1. [Инициализировать.] Установить $t \leftarrow 0$, $j \leftarrow 1$, $s_1 \leftarrow n$. (Предполагается, что $n > 1$.)

A2. [Посетить.] Посетить $s_1 \ldots s_j$. Затем установить $t \leftarrow 1 - t$ и перейти к A4, если $t = 0$.

A3. [Нечетный этап.] Если $s_j > 1$, то установить $s_j \leftarrow s_j - 1$, $s_{j+1} \leftarrow 1$, $j \leftarrow j + 1$, в противном случае установить $j \leftarrow j - 1$ и $s_j \leftarrow s_j + 1$. Вернуться к A2.

A4. [Четный этап.] Если $s_{j-1} > 1$, то установить $s_{j-1} \leftarrow s_{j-1} - 1$, $s_{j+1} \leftarrow s_j$, $s_j \leftarrow 1$, $j \leftarrow j + 1$. В противном случае установить $j \leftarrow j - 1$, $s_j \leftarrow s_{j+1}$, $s_{j-1} \leftarrow s_{j-1} + 1$ (но завершить, если $j - 1 = 0$). Вернуться к A2. ∎

(b) Теперь $q_1, \ldots, q_{t-1}$ представляют кольца на стержне.

B1. [Инициализировать.] Установить $t \leftarrow 1$, $q_0 \leftarrow n$. (Предполагается, что $n > 1$.)

B2. [Посетить.] Установить $q_t \leftarrow 0$ и посетить $(q_0 - q_1) \ldots (q_{t-1} - q_t)$. Перейти к B4, если t четно.

B3. [Нечетный этап.] Если $q_{t-1} = 1$, то установить $t \leftarrow t - 1$. В противном случае установить $q_t \leftarrow 1$ и $t \leftarrow t + 1$. Вериуться к этапу B2.

B4. [Четный этап.] Если $q_{t-2} = q_{t-1} + 1$, то установить $q_{t-2} \leftarrow q_{t-1}$ и $t \leftarrow t - 1$ (но завершить, если $t = 2$), в противном случае установить $q_t \leftarrow q_{t-1}$, $q_{t-1} \leftarrow q_t + 1$, $t \leftarrow t + 1$. Вернуться к B2. ∎

Эти алгоритмы [см. J. Misra, *ACM Trans. Math. Software* **1** (1975), p.285] являются бесцикловыми даже на этапах инициализации.

13. На этапе A1 также установить $C \leftarrow 1$. На этапе A3 установить $C \leftarrow s_j C$, если $s_j > 1$, в противном случае — $C \leftarrow C/(s_{j-1}+1)$. На этапе A4 установить $C \leftarrow s_{j-1}C$ if $s_{j-1} > 1$, в противном случае — $C \leftarrow C/(s_{j-2}+1)$. Аналогичные модификации применимы к этапам B1, B3 и B4. Достаточная точность необходима для подгонки к значению $C = n!$ для композиции $1 \ldots 1$. Расширим определение бесцикловости, предполагая, что для выполнения арифметической операции требуется единица времени.

14. **S1.** [Инициализировать.] Установить $j \leftarrow 0$.

S2. [Посетить.] Посетить строки $a_1 \ldots a_j$.

S3. [Удлинить.] Если $j < n$, то установнть $j \leftarrow j+1$, $a_j \leftarrow 0$ и вернуться к S2.

S4. [Увеличить.] Если $a_j < m_j - 1$, то установить $a_j \leftarrow a_j + 1$ и вернуться к S2.

S5. [Сократить.] Установить $j \leftarrow j-1$ и вернуться к S4, если $j > 0$. ▮

15. **T1.** [Иинициализировать.] Установить $j \leftarrow 0$.

T2. [Четное посещение.] Если j четно, то в таком случае посетить строку $a_1 \ldots a_j$.

T3. [Удлинить.] Если $j < n$, то устаиовить $j \leftarrow j+1$, $a_j \leftarrow 0$ и вернуться к T2.

T4. [Нечетное посещение.] Если j нечетно, то посетить строку $a_1 \ldots a_j$.

T5. [Увеличить.] Если $a_j < m_j - 1$, то установить $a_j \leftarrow a_j + 1$ и вернуться к T2.

T6. [Сократить.] Установить $j \leftarrow j-1$ и вернуться к T4, если $j > 0$. ▮

Этот алгоритм является бесцикловым, хотя на первый взгляд может показаться, что он содержит циклы. Не более четырех шагов отделяют последовательные посещения. Основная идея связана с результатами упражнения 2.3.1–5 и алгоритмом предпостпорядкового обхода ("prepostorder") (алгоритм 7.2.1.6Q).

16. Предположим, что $\texttt{LINK}(j-1) = j + nb_j$ для $1 \le j \le n$ и $\texttt{LINK}(j-1+n) = j + n(1-b_j)$ для $1 < j \le n$. Эти поля связи представляют $(a_1, \ldots, a_n)$ тогда и только тогда, когда $g(b_1 \ldots b_n) = a_1 \ldots a_n$, так что мы можем использовать бесцикловый генератор двоичного кода Грэя для достижения нужного результата.

17. Поместите конкатеиацию 3-битовых кодов $(g(j), g(k))$ в ряду j и столбце k для $0 \le j, k < 8$. [Нетрудно доказать, что это, по сути, *единственное* решение, за исключением перестановок, и/или дополнения координат, и/или вращения строк, поскольку изменяемая координата при движении на север или иа юг зависит только от строки, и аналогичное утверждение применимо к столбцам. Об изоморфизме Карно между 4-кубом и тором 4×4 можно прочесть в *The Design of Switching Circuits*, W. Keister, A. E. Ritchie и S. H. Washburn (1951), p.174. Между прочим, Кайстер придумал остроумный вариант китайской головоломки с кольцамн, который называется SpinOut, и ее обобщение Hexadecimal Puzzle [*U.S. Patents 3637215–3637216* (1972).]

18. Используйте 2-битовый код Грэя для представления цифр $u_j = (0, 1, 2, 3)$ соответственно как битовые пары $u'_{2j-1}u'_{2j} = (00, 01, 11, 10)$. [К. Й. Ли предложил свой вариант в *IEEE Trans.* **IT-4** (1958), p.77–82. Аналогичная $m/2$-битовая кодировка работает для четных значений m. Например, если $m = 8$, то можно представить $(0, 1, 2, 3, 4, 5, 6, 7)$ в виде $(0000, 0001, 0011, 0111, 1111, 1110, 1100, 1000)$. Но такая схема оставляет в стороне некоторые из двоичных структур при $m > 4$.]

19. (a) Модулярный четверичный алгоритм Грэя требует чуть меньше вычислеиий, чем алгоритм M, но это не важно, поскольку число 256 невелико. Результат имеет вид $z_0^8 + z_1^8 + z_2^8 + z_3^8 + 14(z_0^4 z_2^4 + z_1^4 z_3^4) + 56 z_0 z_1 z_2 z_3 (z_0^2 + z_2^2)(z_1^2 + z_3^2)$.

(b) Замена (z_0, z_1, z_2, z_3) на $(1, z, z^2, z)$ дает $1 + 112z^6 + 30z^8 + 112z^{10} + z^{16}$. Таким образом, все ненулевые веса Ли ≥ 6. Теперь примените эту конструкцию в предыдущем упражненин для преобразования каждого вектора $(u_0, u_1, u_2, u_3, u_4, u_5, u_6, u_\infty)$ в 16-битовое число.

20. Восстановите четверичный вектор $(u_0, u_1, u_2, u_3, u_4, u_5, u_6, u_\infty)$ из u' и используйте алгоритм 4.6.1D для поиска остатка от деления $u_0+u_1x+\cdots+u_6x^6$ на $g(x)$ по модулю 4. Этот алгоритм может быть использован, несмотря на то, что эти коэффициенты не относятся к полю, поскольку миогочлен $g(x)$ является нормированным. Выразим остаток в виде $x^j + 2x^k$ (по модулю $g(x)$ и 4) и предположим, что $d = (k-j) \bmod 7$, $s = (u_0+\cdots+u_6+u_\infty) \bmod 4$.

Случай 1, $s = 1$: если $k = \infty$, то ошибка равна x^j. Иначе говоря, правильный вектор имеет $u_j \leftarrow (u_j - 1) \bmod 4$), а в противном случае будет три или более ошибки.

Случай 2, $s = 3$: если $j = k$, ошибка равна $-x^j$, в противном случае будет три или более ошибки.

Случай 3, $s = 0$: если $j = k = \infty$, ошибок не будет; если $j = \infty$ и $k < \infty$, то по крайней мере будет четыре ошибки. В противном случае ошибки будут иметь вид $x^a - x^b$, где $a = (j + (\infty, 6, 5, 2, 3, 1, 4, 0)) \bmod 7$ и соответственно $d = (0, 1, 2, 3, 4, 5, 6, \infty)$ и $b = (j+2d) \bmod 7$.

Случай 4, $s = 2$: если $j = \infty$, то ошибки будут иметь вид $2x^k$. В противном случае ошибки будут иметь вид

$$\begin{aligned} x^j + x^\infty&, \text{ если } k = \infty;\\ -x^j - x^\infty&, \text{ если } d = 0;\\ x^a + x^b&, \text{ если } d \in \{1,2,4\},\ a = (j-3d) \bmod 7,\ b = (j-2d) \bmod 7;\\ -x^a - x^b&, \text{ если } d \in \{3,5,6\},\ a = (j-3d) \bmod 7,\ b = (j-d) \bmod 7. \end{aligned}$$

Для заданного $u' = (1100100100001111)_2$ имеем: $u = (2, 0, 3, 1, 0, 0, 2, 2)$ и $2 + 3x^2 + x^3 + 2x^6 \equiv 1 + 3x + 3x^2 \equiv x^5 + 2x^6$; также $s = 2$. Таким образом, ошибки будут иметь вид $x^2 + x^3$, а ближайшее безошибочное кодовое слово имеет вид $(2, 0, 2, 0, 0, 0, 2, 2)$. Согласно алгоритму 4.6.1D, $2 + 2x^2 + 2x^6 \equiv (2 + 2x + 2x^3)g(x)$ (по модулю 4). Поэтому восемь информационных битов соответствуют $(v_0, v_1, v_2, v_3) = (2, 2, 0, 2)$. (Более "умный" алгоритм мог бы заметить: "Ага, ведь это первые 16 битов числа π".)

Об обобщениях других эффективных схем кодирования на основе четверичных векторов см. в классической работе A. R. Hammons, P. V. Kumar, A. R. Calderbank, N. J. A. Sloane, P. Solé, *IEEE Trans.* **IT-40** (1994), p.301–319.

21. (a) $C(\epsilon) = 1$, $C(0\alpha) = C(1\alpha) = C(\alpha)$ и $C(*\alpha) = 2C(\alpha) - [10\ldots0 \in \alpha]$. Итерирование этой последовательности дает $C(\alpha) = 2^t - 2^{t-1}e_t - 2^{t-2}e_{t-1} - \cdots - 2^0e_1$, где $e_j = [10\ldots0 \in \alpha_j]$ и α_j является суффиксом α, следующим за j-й звездочкой. В данном примере имеем $\alpha_1 = *10{*}{*}0*$, $\alpha_2 = 10{*}{*}0*$, $\ldots$, $\alpha_5 = \epsilon$. Таким образом, $e_1 = 0$, $e_2 = 1$, $e_3 = 1$, $e_4 = 0$ и $e_5 = 1$ (по соглашению), следовательно, $C(**10{*}{*}0*) = 2^5 - 2^4 - 2^2 - 2^1 = 10$.

(b) Можно было бы удалить концевые звездочки, чтобы $t = t'$. Тогда $e_t = 1$ подразумевает $e_{t-1} = \cdots = e_1 = 0$. (Случай $C(\alpha) = 2^{t'-1}$ возникает тогда и только тогда, когда α заканчивается $10^j{*}^k$.)

(c) При вычислении суммы $C(\alpha)$ по всем t-подкубам обратите внимание на то, что $\binom{n}{t}$ кластеров начинаются с n-кортежа $0\ldots0$, а $\binom{n-1}{t}$ кластеров — с последующего n-кортежа (а именно по одному кластеру для каждого t-подкуба, содержащего этот n-кортеж и измененный бит). Таким образом, среднее равно $\big(\binom{n}{t} + (2^n - 1)\binom{n-1}{t}\big)/2^{n-t}\binom{n}{t} = 2^t(1 - t/n) + 2^{t-n}(t/n)$. (Формулы в (c) справедливы для *любого* n-битового пути Грэя, а формулы в (a) и (b) характерны для отраженного двоичного кода Грэя. Эти результаты опубликованы в работе К. Фалутсоса (C. Faloutsos) [*IEEE Trans.* **SE-14** (1988), p.1381–1393].)

22. Пусть $\alpha{*}^j$ и $\beta{*}^k$ являются последовательными листьями двоичного луча Грэя, где α и β являются двоичными строками и $j \le k$. Тогда последние $k - j$ битов α являются такой строкой α', что α и $\beta\alpha'$ являются последовательными элементами двоичного кода Грэя, следовательно, смежными. (Интересные применения этого свойства к кубически

связанным (cube-connected) параллельным компьютерам со связью путем передачи сообщений рассматриваются в работе Вильяма Дж. Делли (W. J. Dally) *A VLSI Architecture для Concurrent Data Structures* (Kluwer, 1987), Chapter 3.)

23. $2^j = g(k) \oplus g(l) = g(k \oplus l)$ подразумевает, что $l = k \oplus g^{[-1]}(2^j) = k \oplus (2^{j+1} - 1)$. Иначе говоря, если $k = (b_{n-1} \dots b_0)_2$ имеем $l = (b_{n-1} \dots b_{j+1} \bar{b}_j \dots \bar{b}_0)_2$.

24. Определяя $g(k) = k \oplus \lfloor k/2 \rfloor$ как обычно, находим $g(k) = g(-1-k)$. Следовательно, существуют *два* таких 2-адических целых числа k, что $g(k)$ имеет заданное 2-адическое значение l. Одно из них четное, а другое — нечетное. Было бы удобно определить, что $g^{[-1]}$ является четным решеннем. Тогда (8) заменяется на $b_j = a_{j-1} \oplus \cdots \oplus a_0$ для $j \geq 0$. Например, $g^{[-1]}(1) = -2$ по этому определению; если l является целым числом, то "знак" $g^{[-1]}(l)$ — четностью l.

25. Пусть $p = k \oplus l$; из упражнения 7.1.3–3 известно, что $2^{\lfloor \lg p \rfloor + 1} - p \leq |k - l| \leq p$. Имеем $\nu(g(p)) = \nu(g(k) \oplus g(l)) = t$ тогда и только тогда, когда существуют положительные целые числа $j_1, \dots, j_t$, такие, что $p = (1^{j_1} 0^{j_2} 1^{j_3} \dots (0 \text{ или } 1)^{j_t})_2$. Наибольшее возможное число $p < 2^n$ появляется, когда $j_1 = n + 1 - t$ и $j_2 = \cdots = j_t = 1$, достигая $p = 2^n - \lceil 2^t/3 \rceil$. Наименьшее возможное число $2^{\lfloor \lg p \rfloor + 1} - p = (1^{j_2} 0^{j_3} \dots (1 \text{ или } 0)^{j_t})_2 + 1$ возникает, если $j_2 = \cdots = j_t = 1$, достигая $p = \lceil 2^t/3 \rceil$ [см. C. K. Yuen, *IEEE Trans.* **IT-20** (1974), p.668; S. R. Cavior, *IEEE Trans.* **IT-21** (1975), p.596].

26. Пусть $N = 2^{n_t} + \cdots + 2^{n_1}$, где $n_t > \cdots > n_1 \geq 0$. Также предположим, что Γ_n является произвольным кодом Грэя для $\{0, 1, \dots, 2^n - 1\}$, который начинается с 0 и заканчивается 1, за исключением того, что Γ_0 просто 0. Используйте

$$\Gamma^R_{n_t},\ 2^{n_t} + \Gamma_{n_{t-1}},\ \dots,\ 2^{n_t} + \cdots + 2^{n_3} + \Gamma^R_{n_2},\ 2^{n_t} + \cdots + 2^{n_2} + \Gamma_{n_1},\ \text{если } t \text{ четно};$$
$$\Gamma_{n_t},\ 2^{n_t} + \Gamma^R_{n_{t-1}},\ \dots,\ 2^{n_t} + \cdots + 2^{n_3} + \Gamma^R_{n_2},\ 2^{n_t} + \cdots + 2^{n_2} + \Gamma_{n_1},\ \text{если } t \text{ нечетно}.$$

27. Вообще, если $k = (b_{n-1} \dots b_0)_2$, то $(k+1)$-й наибольший элемент S_n равен

$$1/(2 - (-1)^{a_{n-1}}/(2 - \cdots/(2 - (-1)^{a_1}/(2 - (-1)^{a_0}))\dots)),$$

со структурой знаков $g(k) = (a_{n-1} \dots a_0)_2$. Таким образом, можно вычислить любой элемент S_n за $O(n)$ шагов, зная его ранг. Задавая $k = 2^{100} - 10^{10}$ и $n = 100$, получим ответ 373065177/1113604409. (Всякий раз, когда $f(x)$ является положительной и монотонной функцией, 2^n элементов $f(\pm f(\dots \pm f(\pm x)\dots))$ упорядочены согласно двончному коду Грэя, как показал Х. Э. Зальцер в своей работе [*CACM* **16** (1973), p.180]. В этом частном случае, однако, существует другой способ получения ответа, поскольку мы также имеем $S_n = //2, \pm 2, \dots, \pm 2, \pm 1//$, используя представленне из раздела 4.5.3. Непрерывная дробь в этом виде упорядочена дополняющими противоположными битами k.)

28. (a) Поскольку $t = 1, 2, \dots$, бит a_j медианы (G_t) пробегает периодическую последовательность

$$0, \dots, 0, *, 1, \dots, 1, *, 0, \dots, 0, *, \dots$$

со звездочками на каждом 2^{1+j}-м этапе. Таким образом, строки, соответствующие двоичному представлению $\lfloor (t-1)/2 \rfloor$ и $\lfloor t/2 \rfloor$, являются медианами. И этн строки действительно являются экстремальными случаями в том смысле, что все медианы согласуются с общими битами $\lfloor (t-1)/2 \rfloor$ и $\lfloor t/2 \rfloor$, следовательно, звездочки встречаются там, где согласования нет. Например, если $t = 100 = (01100100)_2$ и $n = 8$, имеем медиану $(G_{100}) = 001100{*}{*}$.

(b) Поскольку $G_{2t} = 2G_t \cup (2G_t + 1)$, можно предположить, что $t = (a_{n-2} \dots a_1 a_0 1)_2$ является нечетным. Если α равио $g(p)$ и β равио $g(q)$ в двоичиом коде Грэя, то имеем $p = (p_{n-1} \dots p_0)_2$, $q = (p_{n-1} \dots p_{j+1} \bar{p}_j \dots \bar{p}_0)_2$ и $a_{n-1} a_{n-2} = 01 = p_{n-1} p_{n-2}$. Здесь не выполняется $p < t \leq q$, поскольку подразумевается, что $j = n - 1$ и $p_{n-3} = p_{n-4} =$

$\cdots = p_0 = 1$ [см. A. J. Bernstein, K. Steiglitz, J. E. Hopcroft, *IEEE Trans.* **IT-12** (1966), p.425–430].

29. Предположим, что $p \neq 0$, let $l = \lfloor \lg p \rfloor$ и $S_a = \{s \mid 2^l a \le s < 2^l(a+1)\}$ для $0 \le a < 2^{n-l}$. Тогда $(k \oplus p) - k$ имеет постоянный знак для всех $k \in S_a$, а

$$\sum_{k \in S_a} \Big|(k \oplus p) - k\Big| = 2^l |S_a| = 2^{2l}.$$

Кроме того, $g^{[-1]}(g(k) \oplus p) = k \oplus g^{[-1]}(p)$, и $\lfloor \lg g^{[-1]}(p) \rfloor = \lfloor \lg p \rfloor$. Следовательно,

$$\frac{1}{2^n} \sum_{k=0}^{2^n-1} \Big|g^{[-1]}(g(k) \oplus p) - k\Big| = \frac{1}{2^n} \sum_{a=0}^{2^{n-l}-1} \sum_{k \in S_a} \Big|(k \oplus g^{[-1]}(p)) - k\Big| = \frac{1}{2^n} \sum_{a=0}^{2^{n-l}-1} 2^{2l} = 2^l.$$

[см. Morgan M. Buchner, Jr., *Bell System Tech. J.* **48** (1969), p.3113–3130.]

30. Цикл с $k > 1$ имеет длину $2^{\lfloor \lg \lg k \rfloor + 1}$, поскольку легко показать из уравнения (7), что если $k = (b_{n-1} \ldots b_0)_2$ то имеем:

$$g^{[2^l]}(k) = (c_{n-1} \ldots c_0)_2, \qquad \text{где } c_j = b_j \oplus b_{j+2^l}.$$

Чтобы переставить все элементы k, такие, что $\lfloor \lg k \rfloor = t$, есть два варианта: если t является степенью 2, то цикл, содержащий $2\lfloor k/2 \rfloor$, также содержит $2\lfloor k/2 \rfloor + 1$. Поэтому необходимо удвоить лидеров циклов для $t-1$. В противном случае цикл, содержащий $2\lfloor k/2 \rfloor$, является непересекающимся с циклом, содержащим $2\lfloor k/2 \rfloor + 1$, а значит, $L_t = (2L_{t-1}) \cup (2L_{t-1}+1) = (L_{t-1}*)_2$. Этот аргумент, обнаруженный Йоргом Арндтом в 2001 году, дает подсказку и приводит к следующему алгоритму:

P1. [Инициализировать.] Установить $t \leftarrow 1$, $m \leftarrow 0$. (Можно предположить, что $n \ge 2$.)

P2. [Цикл по лидерам.] Установить $r \leftarrow m$. Выполнить алгоритм Q с $k = 2^t + r$, тогда если $r > 0$, то установить $r \leftarrow (r-1)\&m$ и повторять до тех пор, пока не будет выполнено условие $r = 0$ (см. упр. 7.1.3–79).

P3. [Увеличить $\lg k$.] Установить $t \leftarrow t+1$. Завершить, если t равно n. В противном случае установить $m \leftarrow 2m + [t\&(t-1) \neq 0]$ и вернуться к P2. ▮

Q1. [Начать цикл.] Установить $s \leftarrow X_k$, $l \leftarrow k$, $j \leftarrow l \oplus \lfloor l/2 \rfloor$.

Q2. [Выполнить цикл.] Если $j \neq k$, то установить $X_l \leftarrow X_j$, $l \leftarrow j$, $j \leftarrow l \oplus \lfloor l/2 \rfloor$ и повторять до тех пор, пока не будет выполнено условие $j = k$. Затем установить $X_l \leftarrow s$. ▮

31. Поле на основе операции f_n можно получить тогда и только тогда, когда можно получить поле на основе операции $f_n^{[2]}$, которая переводит $(a_{n-1} \ldots a_0)_2$ в $((a_{n-1} \oplus a_{n-2})(a_{n-1} \oplus a_{n-3})(a_{n-2} \oplus a_{n-4}) \ldots (a_2 \oplus a_0)(a_1))_2$. Пусть $c_n(x)$ является характеристическим многочленом матрицы A, определяющей это преобразование, по модулю 2. Тогда $c_1(x) = x+1$, $c_2(x) = x^2 + x + 1$ и $c_{j+1}(x) = xc_j(x) + c_{j-1}(x)$. Поскольку $c_n(A)$ является нулевой матрицей по теореме Кэли–Гамильтона, то поле получается тогда и только тогда, когда $c_n(x)$ является примитивным многочленом, а это условие можно проверить, как в разделе 3.2.2. Первыми такими величинами n являются 1, 2, 3, 5, 6, 9, 11, 14, 23, 26, 29, 30, 33, 35, 39, 41, 51, 53, 65, 69, 74, 81, 83, 86, 89, 90, 95.

(Прокручивая эту последовательность назад, можно показать, что $c_{-j-1}(x) = c_j(x)$, следовательно, $c_j(x)$ делит $c_{(2j+1)k+j}(x)$, например, $c_{3k+1}(x)$ всегда кратно $x+1$. Следовательно, все числа n вида $2jk+j+k$ исключаются, когда $j > 0$ и $k > 0$. Многочлены $c_{18}(x)$, $c_{50}(x)$, $c_{98}(x)$ и $c_{99}(x)$ являются неприводимыми, но не примитивными.)

32. Почти всегда истинно, но ложно в точках, где $w_k(x)$ изменяет знак. (Уолш сначала предположил, что функция $w_k(x)$ должна быть равна нулю в таких точках, но принятое здесь соглашение лучше, поскольку делает более простые формулы, как (15)–(19), верными для всех x.)

33. По индукции для k имеем:

$$w_k(x) = w_{\lfloor k/2\rfloor}(2x) = r_1(2x)^{b_1+b_2} r_2(2x)^{b_2+b_3} \ldots = r_1(x)^{b_0+b_1} r_2(x)^{b_1+b_2} r_3(x)^{b_2+b_3} \ldots$$

для $0 \le x < \frac12$, поскольку $r_j(2x) = r_{j+1}(x)$ и $r_1(x) = 1$ в этом диапазоне. И когда $\frac12 \le x < 1$,

$$\begin{aligned} w_k(x) = (-1)^{\lceil k/2\rceil} w_{\lfloor k/2\rfloor}(2x-1) &= r_1(x)^{b_0+b_1} r_1(2x-1)^{b_1+b_2} r_2(2x-1)^{b_2+b_3} \ldots \\ &= r_1(x)^{b_0+b_1} r_2(x)^{b_1+b_2} r_3(x)^{b_2+b_3} \ldots, \end{aligned}$$

поскольку $\lceil k/2\rceil \equiv b_0 + b_1$ (по модулю 2) и $r_j(2x-1) = r_{j+1}(x-\frac12) = r_{j+1}(x)$ для $j \ge 1$.

34. $p_k(x) = \prod_{j\ge0} r_{j+1}^{b_j}(x)$; следовательно, $w_k(x) = p_k(x)p_{\lfloor k/2\rfloor}(x) = p_{g(k)}(x)$ [см. R. E. A. C. Paley, *Proc. London Math. Soc.* (2) **34** (1932), p.241–279].

35. Если $j = (a_{n-1}\ldots a_0)_2$ и $k = (b_{n-1}\ldots b_0)_2$, то элемент в строке j и столбце k равен $(-1)^{f(j,k)}$, где $f(j,k)$ является суммой всех $a_r b_s$, таких, что $r = s$ (матрица Адамара); $r + s = n-1$ (матрица Пэйли); $r + s = n$ или $n-1$ (матрица Уолша).

Пусть R_n, F_n и G_n являются матрицами перестановок от $j = (a_{n-1}\ldots a_0)_2$ к $k = (a_0\ldots a_{n-1})_2$, $k = 2^n - 1 - j = (\overline{a}_{n-1}\ldots\overline{a}_0)_2$, и $k = g^{[-1]}(j) = ((a_{n-1})\ldots(a_{n-1}\oplus\cdots\oplus a_0))_2$ соответственно. Тогда, используя произведение матриц, получим рекурсивные формулы:

$$R_{n+1} = \begin{pmatrix} R_n\otimes(1\ 0) \\ R_n\otimes(0\ 1)\end{pmatrix}, \qquad F_{n+1} = F_n\otimes\begin{pmatrix}0&1\\1&0\end{pmatrix}, \qquad G_{n+1} = \begin{pmatrix}G_n&0\\0&G_nF_n\end{pmatrix},$$

$$H_{n+1} = H_n\otimes\begin{pmatrix}1&1\\1&\overline{1}\end{pmatrix}, \qquad P_{n+1} = \begin{pmatrix}P_n\otimes(1\ 1)\\P_n\otimes(1\ \overline{1})\end{pmatrix}, \qquad W_{n+1} = \begin{pmatrix}W_n\otimes(1\ 1)\\F_nW_n\otimes(1\ \overline{1})\end{pmatrix}.$$

Таким образом, $W_n = G_n^T P_n = P_n G_n$; $H_n = P_n R_n = R_n P_n$; и $P_n = W_n G_n^T = G_n W_n = H_n R_n = R_n H_n$.

36. W1. [Преобразование Адамара.] Для $k = 0, 1, \ldots, n-1$, замените пару (X_j, X_{j+2^k}) парой $(X_j + X_{j+2^k}, X_j - X_{j+2^k})$ для всех j с четными $\lfloor j/2^k\rfloor$, $0 \le j < 2^n$. (Эти операции эффективно выполняют преобразование $X^T \leftarrow H_n X^T$.)

W2. [Обращение битов.] Примените алгоритм из упражнения 5 для вектора X. (Эти операции эффективно выполняют преобразование $X^T \leftarrow R_n X^T$ в системе обозначений из упр. 35.)

W3. [Двоичная перестановка Грэя.] Примените алгоритм из упр. 30 к вектору X. (Эти операции эффективно выполняют преобразование $X^T \leftarrow G_n^T X^T$.) ▮

Если n имеет одно из особых значений из упражнения 31, то эффективнее объединить этапы W2 и W3 в одном этапе перестановки.

37. Если $k = 2^{e_1} + \cdots + 2^{e_t}$ с $e_1 > \cdots > e_t \ge 0$, то изменение знака происходит в $S_{e_1} \cup \cdots \cup S_{e_t}$, где

$$S_0 = \left\{\frac12\right\}, \quad S_1 = \left\{\frac14, \frac34\right\}, \quad \ldots, \quad S_e = \left\{\frac{2j+1}{2^e} \;\middle|\; 0 \le j < 2^e\right\}.$$

Следовательно, количество изменений знака в $(0\,..\,x)$ равно $\sum_{j=1}^{t} \lfloor 2^{e_j}x + \frac12\rfloor$. Подстановка $x = l/(k+1)$ дает $l + O(t)$ изменений, т.е. l-е изменение происходит самое большее на расстоянии $O(\nu(k))/2^{\lfloor \lg k\rfloor}$ от $l/(k+1)$.

(Этот аргумент позволяет утверждать, что существует бесконечно много пар (k,l) с $|z_{kl} - l/(k+1)| = \Omega((\log k)/k)$. Но пока не видно явной конструкции для таких "плохих" пар.)

38. Пусть $t_0(x) = 1$ и $t_k(x) = \omega^{\lfloor 3x\rfloor\lceil 2k/3\rceil} t_{\lfloor k/3\rfloor}(3x)$, где $\omega = e^{2\pi i/3}$. Тогда $t_k(x)$ обращается вокруг начала $\frac{2}{3}k$ раз при увеличении x от 0 до 1. Если $s_k(x) = \omega^{\lfloor 3^k x\rfloor}$ является третичным аналогом функции Радемахера $r_k(x)$, то имеем $t_k(x) = \prod_{j\ge 0} s_{j+1}(x)^{b_j - b_{j+1}}$, когда $k = (b_{n-1}\dots b_0)_3$, как в модулярном третичном коде Грэя.

39. (a) Воспользуемся символами $\{x_0, x_1, \dots, x_7\}$ вместо $\{a, b, c, d, e, f, g, h\}$. Нам нужно найти такую перестановку p of $\{0, 1, \dots, 7\}$, что матрица с $(-1)^{j\,k} x_{p(j\oplus k}$ в строке j и столбце k имеет ортогональные строки. Это условие эквивалентно требованию, что

$$(j + j')\cdot(p(j) + p(j')) \equiv 1 \text{ (по модулю 2)} \qquad \text{для } 0 \le j < j' < 8.$$

Одно решение имеет вид $p(0)\dots p(7) = 0\,1\,7\,2\,5\,6\,3\,4$, что дает $(a^2+b^2+c^2+d^2+e^2+f^2+g^2+h^2)(A^2+B^2+C^2+D^2+E^2+F^2+G^2+H^2) = \mathcal{A}^2+\mathcal{B}^2+\mathcal{C}^2+\mathcal{D}^2+\mathcal{E}^2+\mathcal{F}^2+\mathcal{G}^2+\mathcal{H}^2$, где

$$\begin{pmatrix}\mathcal{A}\\ \mathcal{B}\\ \mathcal{C}\\ \mathcal{D}\\ \mathcal{E}\\ \mathcal{F}\\ \mathcal{G}\\ \mathcal{H}\end{pmatrix} = \begin{pmatrix} a & b & c & d & e & f & g & h\\ b & -a & d & -c & f & -e & h & -g\\ h & g & -f & -e & d & c & -b & -a\\ c & -d & -a & b & g & -h & -e & f\\ f & e & h & g & -b & -a & -d & -c\\ g & -h & e & -f & -c & d & -a & b\\ d & c & -b & -a & -h & -g & f & e\\ e & -f & -g & h & -a & b & c & -d\end{pmatrix}\begin{pmatrix}A\\ B\\ C\\ D\\ E\\ F\\ G\\ H\end{pmatrix}.$$

(Это тождество было открыто К. Ф. Дегеном [см. *Mémoires de l'Acad. Sci. St. Petersbourg* (5) **8** (1818), p.207–219]. Связанные октонионы рассматриваются в интересном обзоре Дж. К. Баец [см. *Bull. Amer. Math. Soc.* **39** (2002), p.145–205; **42** (2005), p.213, p.229–243]. Кватернионы и октонионы рассматриваются также в книге Дж. Х. Конвэя и Д. Э. Смита, [J. H. Conway, D. A. Smith, *On Quaternions and Octonions* (2003)].

(b) Решения для 16×16 *не* существует. Ближайшее решение имеет вид

$$p(0)\dots p(15) = 0\ 1\ 11\ 2\ 14\ 15\ 13\ 4\ 9\ 10\ 7\ 12\ 5\ 6\ 3\ 8,$$

которое нарушается тогда и только тогда, когда $j \oplus j' = 5$. (Дж. Дж. Сильвестр (J. J. Sylvester) в §9–11,13 своей статьи [*Philos. Mag.* **34** (1867), p.461–475] доказал основные результаты, которые теперь известны как преобразование Адамара, хотя сам Дж. С. Адамар ссылается на Сильвестра [*Bull. des Sciences Mathématiques* (2) **17** (1893), p.240–246]. Более того, Сильвестр вводит преобразование m^n элементов в §14, используя m-е корни единицы.)

40. Да; это изменение действительно проходило бы через переставленные подмножества в лексикографическом двоичном порядке, а не в двоичном порядке Грэя. (Любая матрица 5×5 из 0 и 1, которая не является сингулярной по модулю 2, будет генерировать все 32 варианта при обходе всех линейных комбинаций ее строк.) Наиболее важным аспектом является внешний вид масштабной функции или некоторой другой дельта-последовательности кода Грэя, а не тот факт, что только один элемент a_j изменяется на каждом этапе, в таких случаях, как здесь, когда любое количество a_j может меняться одновременно при тех же затратах.

41. Самое большее 16, например `fired`, `fires`, `finds`, `fines`, `fined`, `fares`, `fared`, `wares`, `wards`, `wands`, `wanes`, `waned`, `wines`, `winds`, `wires`, `wired`. Точно так же 16 слов можно получить на основе `paced/links` и `paled/mints` и, возможно, на основе смешения слова с несловом-антиподом.

42. Предположим, что $n \le 2^{2^r} + r + 1$ и $s = 2^r$. Используем вспомогательную таблицу 2^{r+s} битов f_{jk} для $0 \le j < 2^s$ и $0 \le k < s$, представляющую указатели фокуса, как в алгоритме L, вместе со вспомогательным s-битовым регистром $j = (j_{s-1}\dots j_0)_2$ и $(r+2)$-битовым

счетчиком программы $p = (p_{r+1} \ldots p_0)_2$. На каждом этапе проверяем счетчик программы и, возможно, j-й регистр и один из f битов. Затем на основе просмотренных битов дополняем бит кода Грэя, дополняем бит счетчика программы и, возможно, изменяем j или f бит, таким образом, эмулируя этап L3 по отношению к наиболее значимым $n-r-2$ битам.

Например, существует конструкция для $r = 1$:

$p_2p_1p_0$	Изменить	Установить		$p_2p_1p_0$	Изменить	Установить	
0 0 0	a_0, p_0	$j_0 \leftarrow f_{00}$	$j \leftarrow f_0$	1 1 0	a_0, p_0	$f_{j0} \leftarrow f_{(j+1)0}$	$f_j \leftarrow f_{j+1}$
0 0 1	a_1, p_1	$j_1 \leftarrow f_{01}$		1 1 1	a_1, p_1	$f_{j1} \leftarrow f_{(j+1)1}$	
0 1 1	a_0, p_0	$f_{00} \leftarrow 0$	$f_0 \leftarrow 0$	1 0 1	a_0, p_0	$f_{(j+1)0} \leftarrow (j+1)_0$	$f_{j+1} \leftarrow j+1$
0 1 0	a_2, p_2	$f_{01} \leftarrow 0$		1 0 0	a_{j+3}, p_2	$f_{(j+1)1} \leftarrow (j+1)_1$	

Этот процесс нрекращается, когда предпринимается нопытка нзменить бит a_n.

(На самом деле нужно изменить только *один* вспомогательный бит за этап, если допустить проверку только некоторых двоичных битов Грэя и вспомогательных битов, поскольку $p_r \ldots p_0 = a_r \ldots a_0$, и можно установить $f_0 \leftarrow 0$ более эффективным способом, когда j еще ие принимает свое итоговое значение $2^s - 1$. Эта конструкция, предложенная М. Л. Фредманом (M. L. Fredman) в 2001 году, лучше предложенной им же ранее [см. *SICOMP* **7** (1978), p.134–146.] С помощью улучшенной коиструкции можно сократить количество вспомогательиых битов до $O(n)$.)

43. Это количество было оценеио Дж. Сильверманом, В. Э. Виккерсом и Дж. Л. Сэмпсоном [*IEEE Trans.* **IT-29** (1983), p.894–901] и оказалось равным 7×10^{22}. Можно выполнить и точное вычнсление, поскольку каждый 6-битовый цикл Грэя имеет только пять или менее сегментов, которые лежат в 5-кубе, соответствующем по крайней мере одной из шести координат. (В своей неопубликованной работе Стив Винкер использовал аналогичную идею для оценки $d(5)$ за менее чем 15 минут на типичном компьютере в 1972 году.)

44. Все $(n+1)$-битовые дельта-последовательности только с двумя случаями координаты j получаются с помощью следующей конструкции: пусть $\delta_1 \ldots \delta_{2^n-1}$ и $\varepsilon_1 \ldots \varepsilon_{2^n-1}$ являются n-битовыми дельта-последовательностями для *путей* Грэя с $2^{\delta_1} \oplus \cdots \oplus 2^{\delta_{2^n-1}} = 2^{\varepsilon_1} \oplus \cdots \oplus 2^{\varepsilon_{2^n-1}}$. Создайте цикл

$$\delta_{k+1} \ldots \delta_{2^n-1}\, n\, \varepsilon_1 \ldots \varepsilon_{2^n-1}\, n\, \delta_1 \ldots \delta_k$$

для некоторых k с $0 \le k < 2^n$, затем поменяйте $n \leftrightarrow j$.

Все $(n+2)$-битовые дельта-последовательности только с двумя случаямн координат h и j (с h перед j) аналогично получаются на основе четырех n-битовых последовательностей $\delta_1 \ldots \delta_{2^n-1}, \ldots, \eta_1 \ldots \eta_{2^n-1}$, где $2^{\delta_1} \oplus \cdots \oplus 2^{\eta_{2^n-1}} = 0$ заменой $n \leftrightarrow h$ и $n+1 \leftrightarrow j$ в

$$\delta_{k+1} \ldots \delta_{2^n-1}\, n\, \varepsilon_1 \ldots \varepsilon_{2^n-1}\, (n+1)\, \zeta_1 \ldots \zeta_{2^n-1}\, n\, \eta_1 \ldots \eta_{2^n-1}\, (n+1)\, \delta_1 \ldots \delta_k.$$

Пусть $a(n)$ и $b(n)$ являются количествами n-битовых циклов, определенных в пуиктах (a) и (b); тогда $(a(1), \ldots, a(5)) = (1, 0, 0, 1920, 318996480)$ и $(b(1), \ldots, b(5)) = (0, 2, 12, 384, 4200960)$. Приведенные выше конструкции доказывают, что $a(n+1) + 2b(n+1) = 2^n(n+1)A(n)$ и $b(n+2) = 2^n(n+2)(n+1)B(n)$, если существуют способы $A(n)$ и $B(n)$ выбрать соответствующие последовательности δ, ε, ζ и η. Если ограничиться случаями, где пути Грэя расширяются до циклов Грэя с $\delta_0 = \varepsilon_0 = \zeta_0 = \eta_0$, то получим последовательности $a'(n+1)$ и $b'(n+2)$, где $a'(n+1) + 2b'(n+1) = 2^n(n+1)d(n)^2/n$ и $b'(n+2) = 2^n(n+2)(n+1)d(n)^4/n^3$.

45. Имеем $d(n+1) \ge 2^n d(n)^2/n$, поскольку $2^n d(n)^2/n$ является более низкой границей для количества $(n+1)$-битовых дельта-последовательностей с точио двумя появлениями 0. Следовательно, $d(n+1)^{1/2^{n+1}} > d(n)^{1/2^n}$; а $d(n) \ge \frac{5}{32}\alpha^{2^n}$ для $n \ge 5$, где $\alpha = (\frac{32}{5}d(5))^{1/32} \approx 2{,}06$.

На самом деле можно обнаружить еще более быстрый рост, используя предыдущее упражнение, поскольку $d(n+1) \ge a'(n+1) + b'(n+1)$ и $b'(n+1) \le \frac{25}{64}(n+1)\,d(n)^2/n$ для $n \ge 5$. Следовательно, $d(n+1) \ge (2^n - \frac{25}{64})(n+1)\,d(n)^2/n$ для $n \ge 5$, а итерация этого отношения дает

$$\lim_{n\to\infty} d(n)^{1/2^n} \ge d(5)^{1/32} \prod_{n=5}^{\infty} \Big(2^n - \frac{25}{64}\Big)^{1/2^{n+1}} \Big(\frac{n+1}{n}\Big)^{1/2^{n+1}} \approx 2{,}3606.$$

[См. R. J. Douglas, *Disc. Math.* **17** (1977), p.143–146; M. Mollard, *European J. Comb.* **9** (1988), p.49–52.] Истинное значение этого предела, вероятно, равно ∞.

46. Лео Мозер (неопубликованный результат) предположил, что асимптотнка имеет вид $\sim n/e$. До сих пор установлена только верхняя граннца около $n/\sqrt{2}$; см. ссылку в ответе к предыдущему упражнению.

48. Если $d(n,k,v)$ циклов начинаются с $g(0)\ldots g(k-1)v$, то это предположение подразумевает, что $d(n,k,v) \le d(n,k,g(k))$, поскольку обратный цикл для цикла Грэя также является циклом Грэя. Таким образом, подсказка следует из $d(n) = d(n,1)$ и

$$d(n,k) = \sum_v \{\, d(n,k,v) \mid v \text{ — } g(k-1), v \notin S_k \,\} \le c_{nk} d(n,k,g(k)) = d(n,k+1).$$

Наконец, $d(n,2^n) = 1$, следовательно, $d(n) \le \prod_{k=1}^{2^n-1} c_{nk} = \prod_{k=1}^{n} k^{\binom{n}{k}} = n\prod_{k=1}^{n-1}(k(n-k))^{\binom{n}{k}/2} \le n\prod_{k=1}^{n-1}(n/2)^{\binom{n}{k}} = n(n/2)^{2^n-2}$ [см. *IEEE Trans.* **IT-29** (1983), p.894–901].

49. Возьмите любой путь Гамильтона P от $0\ldots0$ до $1\ldots1$ в $(2n-1)$-кубе, например, как в коде Сэвидж–Винклера, и используйте $0P$, $1\overline{P}$. (Все такие циклы получаются с помощью такой конструкции, когда $n = 1$ или $n = 2$, но гораздо больше возможностей существует, когда $n > 2$.)

50. $\alpha_1(n{+}1)\alpha_1^R n\alpha_1 j_1 \alpha_2 n\alpha_2^R(n{+}1)\alpha_2 \ldots j_{l-1}\alpha_l n\alpha_l^R(n{+}1)\alpha_l n\alpha_l^R j_{l-1}\ldots j_1\alpha_1^R n.$

51. Можно предположить, что $n > 3$ и имеется n-битовый цикл Грэя с количеством переходов $c_j = 2\lfloor(2^{n-1}+j)/n\rfloor$. Нужно создать $(n+2)$-битовый цикл с количеством переходов $c'_j = 2\lfloor(2^{n+1}+j)/(n+2)\rfloor$. Если $2^{n+1} \bmod (n+2) \ge 2$, то можно применить теорему D с $l = 2\lfloor 2^{n+1}/(n+2)\rfloor + 1$, подчеркивая b_j копий из j, где $b_j = 4\lfloor(2^{n-1}+j)/n\rfloor - \lfloor(2^{n+1}+j)/(n+2)\rfloor - [j=0]$, и размещая подчеркнутый 0 последним. Это всегда просто сделать, поскольку $|b_j - 2^{n+2}/n(n+2)| < 5$. Аналогичная конструкция работает, если $2^{n+1} \bmod (n+2) \le n$, с $l = 2\lfloor 2^{n+1}/(n+2)\rfloor - 1$ и $b_j = 4\lfloor(2^{n-1}+j)/n\rfloor - \lfloor(2^{n+1}+j+2)/(n+2)\rfloor - [j=0]$. Действительно, $2^{n+1} \bmod (n+2)$ всегда $\le n$ [см. K. Kedlaya, *Electronic J. Combinatorics* **3** (1996), комментарий к #R25 (9 апреля 1997 года)]. Основная идея этого доказательства принадлежит Дж. П. Робинсону и М. Кону [см. *IEEE Trans.* **C-30** (1981), p.17–23].

52. Количество разных конструкций кода в наименьших положениях координаты j не более $c_0 + \cdots + c_{j-1}$.

53. Обратите внимание на то, что в теореме D рождаются только циклы с $c_j = c_{j+1}$ для некоторого j, поэтому не могут возникать числа $(2, 4, 6, 8, 12)$. Расширение упр. 50 дает также $c_j = c_{j+1} - 2$, но оно не приводит к $(6, 10, 14, 18, 22, 26, 32)$. Наборы чисел, удовлетворяющие условиям упр.52, — точно такие, которые получаются, начиная с $\{2, 2, 4, \ldots, 2^{n-1}\}$, с повторной заменой некоторой пары $\{c_j, c_k\}$ с $c_j < c_k$ парой $\{c_j + 2, c_k - 2\}$.

54. Возьмем значения $\{p_1, \ldots, p_n\}$ и допустим, что x_{jk} является количеством раз, когда p_j встречается в $(a_1, \ldots, a_k)$. Получим $(x_{1k}, \ldots, x_{nk}) \equiv (x_{1l}, \ldots, x_{nl})$ (по модулю 2) для некоторого $k < l$. Но если p являются простыми числами, варьирующимися как дельта-последовательность n-битового цикла Грэя, то единственным решением является $k = 0$ и $l = 2^n$ [см. *AMM* **60** (1953), 418; **83** (1976), p.54].

56. [*Bell System Tech. J.* **37** (1958), p.815–826.] 112 канонических дельта-последовательностей дают следующее.

Класс	Пример	t	Класс	Пример	t	Класс	Пример	t
A	0102101302012023	2	D	0102013201020132	4	G	0102030201020302	8
B	0102303132101232	2	E	0102032021202302	4	H	0102101301021013	8
C	0102030130321013	2	F	0102013102010232	4	I	0102013121012132	1

Здесь B является сбалансированным кодом (рис. 13,b), G является стандартным двоичным кодом Грэя (рнс. 10,b), а H — дополняющим кодом (рис. 13,a). Класс H также является эквивалентным модулярному $(4,4)$ коду Грэя в соответствии с упр. 18. Класс с t автоморфизмами соответствует $32 \times 24/t$ из 2688 разных дельта-последовательностей $\delta_0\delta_1\ldots\delta_{15}$.

Аналогично (см. упр. 7.2.3–00) 5-битовый цикл Грэя относится к 237 675 разным классам эквивалентности.

57. Только для типа 1 изолированными являются 480 вершин, а именно вершины классов D, F, G в предыдущем ответе. Только для типа 2 граф имеет 384 компонента, 288 из которых являются изолированными вершинами классов F и G. Есть 64 компонента размера 9, каждый из которых содержнт 3 вершины нз E и 6 — из A; 16 компонентов размера 30, каждый из которых содержит 6 вершины из H и 24 — из C; 16 компонеитов размера 84, каждый из которых содержит 12 вершины из D, 24 — из B, 48 — из I. Только для типа 3 (или типа 4) весь граф является связным. (Аналогично все 91 392 4-битовых *путей* Грэя являются связанными, если путь $\alpha\beta$ считается смежным для пути $\alpha^R\beta$. В. Э. Виккерс и Дж. Сильверман [*IEEE Trans.* **C-29** (1980), p.329–331] предположили, что изменения типа 3 достаточно для связывания графа n-битовых циклов Грэя для всех $n \geq 3$.)

58. Если некоторая иепустая подстрока $\beta\beta$ включает каждую координату четное количество раз, то такая подстрока не может иметь длину $|\beta|$, поэтому некоторый циклический сдвиг β имеет префикс γ с тем же свойством четности. Но тогда α не определяет цикл Грэя, поскольку можио обращать в 0 каждое значение n из γ.

59. Если α является нелокальным в упр. 58, то иелокальным является и $\beta\beta$ при условии, что $q > 1$ и 0 возникает больше чем $q+1$ раз в α. Следовательно, начиная с α из (30), но с 0 и 1, которые поменялись местами, получим иелокальные циклы для $n \geq 5$, в которых координата 0 измеияется точио шесть раз [см. Mark Ramras, *Discrete Math.* **85** (1990), p.329–331]. С другой стороны, 4-битовый цикл Грэя не может быть нелокальным, поскольку ои всегда имеет проход длины 2; если $\delta_k = \delta_{k+2}$, то элемеиты $\{v_{k-1}, v_k, v_{k+1}, v_{k+2}\}$ образуют 2-подкуб.

60. Используйте конструкцию из упр. 58 с $q = 1$.

61. Идея состоит в том, чтобы перемежать m-битовый цикл $U = (u_0, u_1, u_2, \ldots)$ с n-битовым циклом $V = (v_0, v_1, v_2, \ldots)$ с образованием конкатеиаций

$$W = (u_{i_0}v_{j_0},\ u_{i_1}v_{j_1},\ u_{i_2}v_{j_2},\ \ldots), \qquad i_k = \bar{a}_0 + \cdots + \bar{a}_{k-1}, \quad j_k = a_0 + \cdots + a_{k-1},$$

где $a_0a_1a_2\ldots$ является периодической строкой управляющих битов $\alpha\alpha\alpha\ldots$. Если $a_k = 0$, то продвигаемся к следующему элементу U, в противном случае — к следующему злементу V.

Если α является любой строкой длины $2^m \leq 2^n$, содержащей s бит из 0 и $t = 2^m - s$ бит из 1, то W будет $(m+n)$-битовым циклом Грэя, если s и t являются иечетными. Имеем $i_{k+l} \equiv i_k$ (по модулю 2^m) и $j_{k+l} \equiv j_k$ (по модулю 2^n), только если l кратно 2^m, так как $i_k + j_k = k$. Допустим, что $l = 2^m c$, тогда $j_{k+l} = j_k + tc$, так что c кратно 2^n.

(a) Пусть $\alpha = 0111$; тогда серии длины 8 возннкают в левых 2 битах, а серии длины $\geq \lfloor \frac{4}{3}r(n) \rfloor$ — в правых n бнтах.

(b) Пусть s является самым крупным нечетным числом $\le 2^m r(m)/(r(m)+r(n))$. Предположим, что $t = 2^m - s$ и $a_k = \lfloor (k+1)t/2^m \rfloor - \lfloor kt/2^m \rfloor$, так что $i_k = \lceil ks/2^m \rceil$ и $j_k = \lfloor kt/2^m \rfloor$. Если серия длины l возникает в левых m битах, то имеем $i_{k+l+1} \ge i_k + r(m) + 1$, следовательно, $l+1 > 2^m r(m)/s \ge r(m)+r(n)$. А если она возникает в правых n битах, то имеем $j_{k+l+1} \ge j_k + r(n) + 1$, следовательно,

$$l+1 > 2^m r(n)/t > 2^m r(n)/(2^m r(n)/(r(m)+r(n)) + 2)$$
$$= r(m) + r(n) - \frac{2(r(m)+r(n))^2}{2^m r(n) + 2(r(m)+r(n))} > r(m) + r(n) - 1,$$

поскольку $r(m) \le r(n)$.

Эта конструкция часто работает также в менее ограниченных случаях. Более подробно серии в коде Грэя рассмотреиы Л. Годдином (L. Goddyn), Дж. М. Лоуренсом (G. M. Lawrence) и Э. Немет (E. Nemeth) [*Utilitas Math.* **34** (1988), p.179–192].

63. Установить $a_k \leftarrow k \bmod 4$ для $0 \le k < 2^{10}$, за исключением того, что $a_k = 4$, если $k \bmod 16 = 15$, или $k \bmod 64 = 42$, или $k \bmod 256 = 133$. Также установить $(j_0, j_1, j_2, j_3, j_4) \leftarrow (0, 2, 4, 6, 8)$. Затем для $k = 0, 1, \ldots, 1023$ установить $\delta_k \leftarrow j_{a_k}$ и $j_{a_k} \leftarrow 1 + 4a_k - j_{a_k}$. (Эта конструкция обобщает метод из упр. 61.)

64. (a) Каждый элемент u_k появляется вместе с $\{v_k, v_{k+2^m}, \ldots, v_{k+2^m(2^{n-1}-1)}\}$ и $\{v_{k+1}, v_{k+1+2^m}, \ldots, v_{k+1+2^m(2^{n-1}-1)}\}$. Таким образом, перестановка $\sigma_0 \ldots \sigma_{2^m-1}$ должна быть 2^{n-1}-циклом n-битовых четных вершин, который умножен на произвольную перестановку других вершин. Это условие является достаточным.

(b) Пусть τ_j является такой перестановкой, что $v \mapsto v \oplus 2^j$, и пусть $\pi_j(u, w)$ является перестановкой $(uw)\tau_j$. Если $u \oplus w = 2^i + 2^j$, то $\pi_j(u, w)$ принимает $u \mapsto u \oplus 2^i$ и $w \mapsto w \oplus 2^i$, тогда как $v \mapsto v \oplus 2^j$ для всех других вершин v, что переводит каждую вершину в соседнюю.

Если S является произвольным множеством $\subseteq \{0, \ldots, n-1\}$, то пусть $\sigma(S)$ является потоком всех перестановок τ_j для всех $j \in \{0, \ldots, n-1\} \setminus S$, в порядке возрастания j, повторенном дважды. Например, если $n = 5$, то имеем: $\sigma(\{1, 2\}) = \tau_0\tau_3\tau_4\tau_0\tau_3\tau_4$. Тогда поток Грэя

$$\Sigma(i, j, u) = \sigma(\{i, j\})\pi_j(u, u \oplus 2^i \oplus 2^j)\sigma(\{i, j\})\tau_j\sigma(\{j\})$$

состоит из $6n-8$ перестановок, произведение которых является транспозицией $(u\, u \oplus 2^i \oplus 2^j)$. Более того, если этот поток применяется к любой n-битовой вершине v, то все его серии имеют длину $n-2$ или больше.

Можно предположить, что $n \ge 5$. Пусть $\delta_0 \ldots \delta_{2^n-1}$ является дельта-последовательностью для n-битового цикла Грэя $(v_0, v_1, \ldots, v_{2^n-1})$ со всеми сериями длины 3 или больше. Тогда произведение всех перестановок в

$$\Sigma = \prod_{k=1}^{2^{n-1}-1} \big(\Sigma(\delta_{2k-1}, \delta_{2k}, v_{2k-1})\, \Sigma(\delta_{2k}, \delta_{2k+1}, v_{2k})\big)$$

имеет вид $(v_1\, v_3)(v_2\, v_4) \ldots (v_{2^n-3}\, v_{2^n-1})(v_{2^n-2}\, v_0) = (v_{2^n-1} \ldots v_1)(v_{2^n-2} \ldots v_0)$, т.е. удовлетворяет условию цикла (a).

Более того, все степени $(\sigma(\emptyset)\Sigma)^i$ порождают серии длины $\ge n-2$, когда применяются к любой вершине v. Повторяя отдельные множители $\sigma(\{i, j\})$ или $\sigma(\{j\})$ в Σ столько, сколько нужно, можно настроить длину $\sigma(\emptyset)\Sigma$, получая $2n + (2^{n-1}-1)(12n-16) + 2(n-2)a + 2(n-1)b$ для любых целых чисел $a, b \ge 0$. Таким образом, можно точно увеличить ее длину до 2^m, при условии, что $2^m \ge 2n + (2^{n-1}-1)(12n-16) + 2(n^2-5n+6)$, согласно упражнению 5.2.1–21.

(c) Ограничение $r(n) \ge n - 4\lg n + 8$ может быть доказано для $n \ge 5$ следующим образом. Сначала заметим, что оно справедливо для $5 \le n < 33$, согласно методу из

упр. 60–63. Затем можно заметить, что каждое целое число $N \geq 33$ можно записать в виде $N = m + n$ или $N = m + n + 1$ для некоторого $m \geq 20$, где

$$n = m - \lfloor 4 \lg m \rfloor + 10.$$

Если $m \geq 20$, то 2^m является достаточно большим, чтобы была верна конструкция в п. (b). Следовательно,

$$\begin{aligned} r(N) \geq r(m+n) \geq 2\min(r(m), n-2) &\geq 2(m - \lfloor 4 \lg m \rfloor + 8) \\ &= m + n + 1 - \lfloor 4 \lg(m+n) - 1 + \epsilon \rfloor + 8 \\ &\geq N - 4 \lg N + 8, \end{aligned}$$

где $\epsilon = 4 \lg(2m/(m+n)) < 1$ [см. *Electronic Journal of Combinatorics* **10** (2003), #R27, p.1–10]. Действительно, рекурсивное применение (b) дает $r(1024) \geq 1000$.

65. Компьютерный поиск дает восемь существенно разных возможных структур (и их обратные структуры). Одна из них имеет дельта-последовательность 0102031420302404 1234214103234103 и достаточно близка к двум другим.

66. (Решение Марка Кука) Одной такой дельта-последовательностью является 01234560 70121324356576071021353462670153741236256701731426206570134214656057310246453757102043537614073630464273703564027132750541210275641502403654250136025416156043125760325720431576243217604520417516354767035647570625437242132624161523417514367143164314. (Решения для $n > 8$ все еще неизвестны.)

67. Пусть $v_{2k+1} = \bar{v}_{2k}$ и $v_{2k} = 0u_k$, где $(u_0, u_1, \ldots, u_{2^n-1})$ является произвольным $(n-1)$-битовым циклом Грэя [см. J. P. Robinson, M. Cohn, *IEEE Trans.* **C-30** (1981), p.17–23].

68. Да. Простейший способ, вероятно, состоит в том, чтобы взять $(n-1)$-тритовое число и добавить $0 \ldots 0$, $1 \ldots 1$, $2 \ldots 2$ в каждую строку (по модулю 3). Например, для $n = 3$ код имеет вид 000, 111, 222, 001, 112, 220, 002, 110, 221, 012, 120, 201, ..., 020, 101, 212.

69. (a) Необходимо проверить только те изменения в h, когда биты $b_{j-1} \ldots b_0$ являются одновременно дополненными для $j = 1, 2, \ldots$. И такими изменениями являются соответственно $(1110)_2$, $(1101)_2$, $(0111)_2$, $(1011)_2$, $(10011)_2$, $(100011)_2$, ... Для доказательства того, что возникает каждый n-кортеж, заметим, что $0 \leq h(k) < 2^n$, где $0 \leq k < 2^n$ и $n > 3$. Кроме того, $h^{[-1]}((a_{n-1} \ldots a_0)_2) = (b_{n-1} \ldots b_0)_2$, где $b_0 = a_0 \oplus a_1 \oplus a_2 \oplus \cdots$, $b_1 = a_0$, $b_2 = a_2 \oplus a_3 \oplus a_4 \oplus \cdots$, $b_3 = a_0 \oplus a_1 \oplus a_3 \oplus \cdots$ и $b_j = a_j \oplus a_{j+1} \oplus \cdots$ для $j \geq 4$.

(b) Пусть $h(k) = (\ldots a_2 a_1 a_0)_2$, где $a_j = b_j \oplus b_{j+1} \oplus b_0[j \leq t] \oplus b_{t-1}[t-1 \leq j \leq t]$.

70. Как в (32) и (33), можно удалить $n!$, предполагая, что строки с весом 1 упорядочены. Тогда есть 14 решений для $n = 5$, начинающихся с 00000, и 21 решение, начинающееся с 00001. Для $n = 6$ существует 46 935 решений каждого типа (связанных обращением и дополнением). Для $n = 7$ количество решений очень велико, гораздо больше, но очень мало по сравнению с общим числом 7-битовых кодов Грэя.

71. Допустим, что $\alpha_{n(j+1)}$ отличается от α_{nj} в координате t_j для $0 \leq j < n-1$. Тогда $t_j = j\pi_n$, согласно (44) и (38). Теперь (34) означает, что $t_0 = n-1$. И если $0 < j < n-1$, то имеем: $t_j = ((j-1)\pi_{n-1})\pi_{n-1}$ согласно (40). Таким образом, $t_j = j\sigma_n\pi_{n-1}^2$ для $0 \leq j < n-1$, а значение $(n-1)\pi_n$ определяется тем, что осталось. (Обозначения этих перестановок чрезвычайно сложные, поэтому рекомендуется всегда тщательно проверять их на простых случаях.)

72. Дельта-последовательность имеет вид 0102132430201234012313041021323.

73. Пусть $Q_{nj} = P^R_{nj}$, и обозначим (41) и (42) как S_n и T_n. Таким образом, $S_n = P_{n0}Q_{n1}P_{n2}\ldots$ и $T_n = Q_{n0}P_{n1}Q_{n2}\ldots$, если опустить запятые. Тогда имеем:

$$S_{n+1} = 0P_{n0}\ 0Q_{n1}\ 1Q^{\pi}_{n0}\ 1P^{\pi}_{n1}\ 0P_{n2}\ 0Q_{n3}\ 1Q^{\pi}_{n2}\ 1P^{\pi}_{n3}\ 0P_{n4}\ \ldots,$$
$$T_{n+1} = 0Q_{n0}\ 1P^{\pi}_{n0}\ 0P_{n1}\ 0Q_{n2}\ 1Q^{\pi}_{n1}\ 1P^{\pi}_{n2}\ 0P_{n3}\ 0Q_{n4}\ 1Q^{\pi}_{n3}\ \ldots,$$

где $\pi = \pi_n$, которые демонстрируют сравнительно простую совместную рекурсию между дельта-последовательностями Δ_n и E_n из S_n и T_n. А именно, если записать

$$\Delta_n = \phi_1\, a_1\, \phi_2\, a_2 \ldots \phi_{n-1}\, a_{n-1}\, \phi_n, \qquad E_n = \psi_1\, b_1\, \psi_2\, b_2 \ldots \psi_{n-1}\, b_{n-1}\, \psi_n,$$

где все ϕ_j и ψ_j являются строками длины $2\binom{n-1}{j-1} - 1$, то следующие последовательности имеют вид

$$\Delta_{n+1} = \phi_1\ a_1\ \phi_2\ n\ \psi_1\pi\ b_1\pi\ \psi_2\pi\ n\ \phi_3\ a_3\ \phi_4\ n\ \psi_3\pi\ b_3\pi\ \psi_4\pi\ n\ \ldots$$
$$E_{n+1} = \psi_1\ n\ \phi_1\pi\ n\ \psi_2\ b_2\ \psi_3\ n\ \phi_2\pi\ a_2\pi\ \phi_3\pi\ n\ \psi_4\ b_4\ \psi_5\ n\ \phi_4\pi\ a_4\pi\ \phi_5\pi\ n\ \ldots$$

Например, имеем: $\Delta_3 = 0\underline{1}0\,2\,1\underline{0}\,1$ и $E_3 = 0\underline{2}\,1\,2\,0\underline{2}\,1$, если подчеркнуть a и b, чтобы отличить их от ϕ и ψ; и

$$\Delta_4 = 0\,1\,0\,2\,1\,3\,0\pi\ 2\pi\ 1\pi\ 2\pi\ 0\pi\ 3\,1\,3\,1\pi = 0\,\underline{1}\,0\,2\,1\,3\,2\,\underline{1}\,0\,1\,2\,3\,1\,\underline{3}\,0,$$
$$E_4 = 0\,3\,0\pi\ 3\,1\,2\,0\,2\,1\,3\,0\pi\ 2\pi\ 1\pi\ 0\pi\ 1\pi = 0\,\underline{3}\,2\,3\,1\,2\,0\,\underline{2}\,1\,3\,2\,1\,0\,\underline{2}\,0;$$

где $a_3\phi_4$ и $b_3\psi_4$ — пустые. Элементы подчеркнуты для следующего этапа.

Таким образом, дельта-последовательности можно подсчитать следующим образом. Здесь $p[j] = j\pi_n$ для $1 \le j < n$; $s_k = \delta_k$, $t_k = \varepsilon_k$ и $u_k = [\delta_k$ и ε_k подчеркнуты] — для $0 \le k < 2^n - 1$.

R1. [Инициализировать.] Установить $n \leftarrow 1$, $p[0] \leftarrow 0$, $s_0 \leftarrow t_0 \leftarrow u_0 \leftarrow 0$.

R2. [Продвинуть n.] Выполнить алгоритм S, приведенный ниже, который вычисляет массивы s', t' и u' для следующего n. Затем установить $n \leftarrow n + 1$.

R3. [Готово?] Если n достаточно велико, нужная дельта-последовательность Δ_n находится в массиве s'; завершить. В противном случае продолжить выполнение.

R4. [Вычислить π_n.] Установить $p'[0] = n - 1$ и $p'[j] = p[p[j-1]]$ для $1 \le j < n$.

R5. [Подготовиться к продвижению.] Установить $p[j] \leftarrow p'[j]$ для $0 \le j < n$; установить $s_k \leftarrow s'_k$, $t_k \leftarrow t'_k$ и $u_k \leftarrow u'_k$ для $0 \le k < 2^n - 1$. Вернуться к R2. ▮

На следующих этапах выражение "Передавать нечто(l, j), пока $u_j = 0$" означает: "Если $u_j = 0$, повторно выполнять нечто(l, j), $l \leftarrow l + 1$, $j \leftarrow j + 1$, пока не выполняется условие $u_j \ne 0$".

S1. [Подготовиться к вычислению Δ_{n+1}.] Установить $j \leftarrow k \leftarrow l \leftarrow 0$ и $u_{2^n-1} \leftarrow -1$.

S2. [Продвинуть j.] Передавать $s'_l \leftarrow s_j$ и $u'_l \leftarrow 0$, пока $u_j = 0$. Затем перейти к S5, если $u_j < 0$.

S3. [Продвинуть j и k.] Установить $s'_l \leftarrow s_j$, $u'_l \leftarrow 1$, $l \leftarrow l + 1$, $j \leftarrow j + 1$. Затем передавать $s'_l \leftarrow s_j$ и $u'_l \leftarrow 0$, пока $u_j = 0$. Затем установить $s'_l \leftarrow n$, $u'_l \leftarrow 0$, $l \leftarrow l + 1$. Затем передавать $s'_l \leftarrow p[t_k]$ и $u'_l \leftarrow 0$, пока $u_k = 0$. Затем установить $s'_l \leftarrow p[t_k]$, $u'_l \leftarrow 1$, $l \leftarrow l + 1$, $k \leftarrow k + 1$. И еще раз передавать $s'_l \leftarrow p[t_k]$ и $u'_l \leftarrow 0$, пока $u_k = 0$.

S4. [Готово с Δ_{n+1}?] Если $u_k < 0$, то перейти к S6. В противном случае установить $s'_l \leftarrow n$, $u'_l \leftarrow 0$, $l \leftarrow l + 1$, $j \leftarrow j + 1$, $k \leftarrow k + 1$ и вернуться к S2.

S5. [Завершить Δ_{n+1}.] Установить $s'_l \leftarrow n$, $u'_l \leftarrow 1$, $l \leftarrow l + 1$. Затем передавать $s'_l \leftarrow p[t[k]]$ и $u'_l \leftarrow 0$, пока $u_k = 0$.

S6. [Подготовиться к вычислению E_{n+1}.] Установить $j \leftarrow k \leftarrow l \leftarrow 0$. Передавать $t'_l \leftarrow t_k$, пока $u_k = 0$. Затем установить $t'_l \leftarrow n$, $l \leftarrow l+1$.

S7. [Продвинуть j.] Передавать $t'_l \leftarrow p[s_j]$, пока $u_j = 0$. Завершить, если $u_j < 0$. В противном случае установить $t'_l \leftarrow n$, $l \leftarrow l+1$, $j \leftarrow j+1$, $k \leftarrow k+1$.

S8. [Продвинуть k.] Передавать $t'_l \leftarrow t_k$, пока $u_k = 0$. Затем перейти к S10, если $u_k < 0$.

S9. [Продвинуть k и j.] Установить $t'_l \leftarrow t_k$, $l \leftarrow l+1$, $k \leftarrow k+1$. Затем передавать $t'_l \leftarrow t_k$, пока $u_k = 0$. Затем установить $t'_l \leftarrow n$, $l \leftarrow l+1$. Затем передавать $t'_l \leftarrow p[s_j]$, пока $u_j = 0$. Затем установить $t'_l \leftarrow p[s_j]$, $l \leftarrow l+1$, $j \leftarrow j+1$. Вернуться к S7.

S10. [Завершить E_{n+1}.] Установить $t'_l \leftarrow n$, $l \leftarrow l+1$. Затем передавать $t'_l \leftarrow p[s_j]$, пока $u_j = 0$. ▮

Для генерации монотонного кода Сэвидж–Винклера для достаточно больших n можно сначала сгенерировать Δ_{10} и E_{10} или даже Δ_{20} и E_{20}. С помощью этих таблиц подходящая рекурсивная процедура сможет достигнуть высоких значений n с очень малыми вычислительными накладными расходами в среднем на один этап.

74. Если монотонный путь имеет вид $v_0, \ldots, v_{2^n-1}$ и если v_k имеет вес j, то имеем:

$$2\sum_{t>0}\binom{n}{j-2t} + ((j+\nu(v_0)) \bmod 2) \le k \le 2\sum_{t\ge 0}\binom{n}{j-2t} + ((j+\nu(v_0)) \bmod 2) - 2.$$

Следовательно, максимальное расстояние между вершинами весов j и $j+1$ равно $2(\binom{n-1}{j-1} + \binom{n-1}{j} + \binom{n-1}{j+1}) - 1$. Максимальное значение, приблизительно равное $3\cdot 2^n/\sqrt{2\pi n}$, возникает, если j приблизительно равно $n/2$. (Оно всего в три раза больше наименьшего значения, достижимого в *произвольном* упорядочении вершин, которое равно $\sum_{j=0}^{n-1}\binom{j}{\lfloor j/2\rfloor}$ согласно упр. 7.10–00).

75. Есть только пять существенно различных решений, все из которых действительно являются *циклами* Грэя. Эти дельта-последовательности имеют вид

0 1 2 3 0 1 2 4 2 1 0 3 2 1 0 1 2 1 0 3 2 1 0 4 0 1 2 3 0 1 2 (1)
0 1 2 3 0 1 2 4 2 1 0 3 2 1 0 1 3 0 1 2 3 0 1 4 1 0 3 2 1 0 3 (1)
0 1 2 3 0 1 2 4 2 1 0 3 2 1 0 2 0 3 2 1 0 3 2 4 2 3 0 1 2 3 0 (2)
0 1 2 3 0 1 2 4 2 3 0 1 2 3 0 2 0 1 2 3 0 1 2 4 2 3 0 1 2 3 0 (2)
0 1 2 3 4 1 0 1 2 1 0 3 0 1 4 3 2 1 0 3 0 1 4 1 0 1 2 3 4 1 0 (3)

76. Если $v_0, \ldots, v_{2^n-1}$ не имеет тренда, то не имеет тренда $(n+1)$-битовый цикл $0v_0$, $1v_0$, $1v_1$, $0v_1$, $0v_2$, $1v_2$, $\ldots$, $1v_{2^n-1}$, $0v_{2^n-1}$. На рис. 14,*g* показана более интересная конструкция, которая обобщает первое решение упр. 75 до $(n+2)$-битового цикла:

$$00\Gamma''^R,\ 01\Gamma'^R,\ 11\Gamma',\ 10\Gamma'',\ 10\Gamma,\ 11\Gamma''',\ 01\Gamma'''^R,\ 00\Gamma^R,$$

где Γ является n-битовой последовательностью $g(1), \ldots, g(2^n-1)$ и $\Gamma' = \Gamma \oplus g(1)$, $\Gamma'' = \Gamma \oplus g(2^{n-1})$, $\Gamma''' = \Gamma \oplus g(2^{n-1}+1)$. ($n$-битовая структура без тренда, которая *почти* является кодом Грэя, имеющая только четыре этапа, на которых $\nu(v_k \oplus v_{k+1}) = 2$, была найдена для всех $n \ge 3$ К. С. Ченгом [см. *Proc. Berkeley Conf. Neyman and Kiefer* **2** (Hayward, Calif.: Inst. of Math. Statistics, 1985), p.619–633].)

77. Замените массив $(o_{n-1}, \ldots, o_0)$ массивом сигнальных значений $(s_{n-1}, \ldots, s_0)$ с $s_j \leftarrow m_j - 1$ на этапе H1. Установите $a_j \leftarrow (a_j+1) \bmod m_j$ на этапе H4. Если $a_j = s_j$ на этапе H5, то установите $s_j \leftarrow (s_j-1) \bmod m_j$, $f_j \leftarrow f_{j+1}$, $f_{j+1} \leftarrow j+1$.

78. Обратите внимание на то, что B_{j+1} в (50) является количеством отражений (рефлексий) в координате j, поскольку координата j пропускается на этапах, кратных $m_j \ldots m_0$. Следовательно, если $b_j < m_j$, то увеличение b_j на 1 приводит a_j к увеличению или уменьшению на 1. Более того, если $b_i = m_i - 1$ для $0 \le i < j$, то замена все этих b_i на 0 при увеличении на 1 b_j приведет к увеличению каждого $B_0, \ldots, B_j$ на 1, оставляя таким образом значения $a_0, \ldots, a_{j-1}$ неизменными в (50). Обратите внимание на то, что в (51) выполняется $B_j = m_j B_{j+1} + b_j \equiv m_j B_{j+1} + a_j + (m_j - 1)B_{j+1} \equiv a_j + B_{j+1}$ (по модулю 2). Следовательно, $B_j \equiv a_j + a_{j+1} + \cdots$, и (51) очевидно эквивалентны (50).

В модулярном коде Грэя для общего основания $(m_{n-1}, \ldots, m_0)$ допустнм

$$\bar{g}(k) = \begin{bmatrix} a_{n-1}, & \ldots, & a_2, & a_1, & a_0 \\ m_{n-1}, & \ldots, & m_2, & m_1, & m_0 \end{bmatrix},$$

когда k определяется (46). Тогда $a_j = (b_j - B_{j+1}) \bmod m_j$, поскольку координата j увеличена по модулю m_j точно $B_j - B_{j+1}$ раз, если начать с $(0, \ldots, 0)$. Обратная функция, которая определяет b на основании модулярного кода Грэя a, имеет вид $b_j = (a_j + a_{j+1} + a_{j+2} + \cdots) \bmod m_j$ в особом случае, когда каждое m_j является делителем m_{j+1} (например, если все m_j равны). Но в общем случае обратная функция не имеет простоого вида. Ее можно вычислить с помощью рекуррентных соотношений $b_j = (a_j + B_{j+1}) \bmod m_j$, $B_j = m_j B_{j+1} + b_j$ для $j = n-1, \ldots, 0$, начиная с $B_n = 0$.

(Отраженный (рефлексивный) код Грэя для основания $m > 2$ предложен Иваном Флоресом [см. *IRE Trans.* **EC-5** (1956), p.79–82]. Он вывел (50) и (51) для случая, когда все m_j равны. Модулярный код Грэя с общим смешанным основанием неявно обсуждался Джозефом Розенбаумом [см. *AMM* **45** (1938), p.694–696], но без формул преобразования. Формулы преобразования для всех m_j, имеющих общее значение m, опубликованы Мартином Коном [см. *Info. и Control* **6** (1963), p.70–78].)

79. (a) Для последнего n-кортежа всегда выполняется $a_{n-1} = m_{n-1} - 1$, ноэтому он находится на расстоянии одного шага от $(0, \ldots, 0)$, только если $m_{n-1} = 2$. А это условие является достаточным для финального n-кортежа $(1, 0, \ldots, 0)$. (Аналогично финальный подлес из вывода алгоритма K является смежным к исходному тогда и только тогда, когда самое левое дерево является изолированной вершиной.)

(b) Последний n-кортеж имеет вид $(m_{n-1} - 1, 0, \ldots, 0)$ тогда и только тогда, когда $m_{n-1} \ldots m_{j+1} \bmod m_j = 0$ для $0 \le j < n-1$, поскольку $b_j = m_j - 1$ и $B_j = m_{n-1} \ldots m_j - 1$.

80. Пройдите все $p_1^{a_1} \ldots p_t^{a_t}$ с помощью отраженного кода Грэя с основаниями $m_j = e_j + 1$.

81. Первый цикл содержит ребро от (x, y) до $(x, (y+1) \bmod m)$ тогда и только тогда, когда $(x + y) \bmod m \ne m - 1$, тогда и только тогда, когда второй цикл содержит ребро от (x, y) до $((x+1) \bmod m, y)$.

82. Существуют два 4-битовых цикла Грэя $(u_0, \ldots, u_{15})$ и $(v_0, \ldots, v_{15})$, которые покрывают все ребра 4-куба. (Действительно, неребра классов A, B, D, H и I в упр. 56 образуют циклы Грэя в тех же классах, что и их дополиения.) Следовательно, с номощью 16-го модулярного кода Грэя можно образовать четыре искомых цикла: $(u_0u_0, u_0u_1, \ldots, u_0u_{15}, u_1u_{15}, \ldots, u_{15}u_0)$, $(u_0u_0, u_1u_0, \ldots, u_{15}u_0, u_{15}u_1, \ldots, u_0u_{15})$, $(v_0v_0, \ldots, v_{15}v_0)$, $(v_0v_0, \ldots, v_0v_{15})$.

Аналогичным образом можно показать, что существует $n/2$ n-битовых цикла Грэя типа "ребро-непересечение" для n is 16, 32, 64 и т.д. [см. *Abhandlungen Math. Sem. Hamburg* **20** (1956), p.13–16.] Дж. Оберт и Б. Шнайдер доказали [см. *Discrete Math.* **38** (1982), p.7–16], что аналогичными свойствами обладают *все* четные значения $n \ge 4$, но ни одной простой конструкции неизвестно.

83. Марк Кук нашел следующее абсолютно несимметричное решение в декабре 2002 года:

(1) 27374650573202656123165467436105251060520424163143721451014217372506246064173213107351607103156205713172463452102434643207054702414735614673762504735074513062065641507312373142737643256124026430167354675324025246374752176402707360651052151060735754632531056;

(2) 06167134172321751716715404602471647424732025316216735316327360526710141503047313570615453627623241426465272021632075363710750740315767476154565275651045102402310735342465123040654530621371053726205017524534067034373435315026024630456276741527524060216104346;

(3) 37010637515071312362437657351030120423537472074104736216172473246505132565057121565024570473247421427640231034362703262764130574056062034174561315175631470272172520561321260405350626046017364267176417435134012453602417306365450615630274145356764326257450516;

(4) 67065464356721472362104054320545107374051705321454316364305046734560621206416201320742373627204506473140171020514126107452343672132045275235341051542637060136356730710542016315121053506173123642725371656172175425107602154623754526742570373464036473762716576.

(Каждая из этих дельта-последовательностей должна начинаться с той же вершины куба.) Существует ли симметричное решение этого упражнения?

84. Если обозначить исходное положение как $(2,2)$, то решение из 8 этапов будет выглядеть, как показано на рис. A-1, т.е. как последовательность, прогрессирующая до $(0,0)$. На первом этапе, например, передняя половина веревки обходит правую расческу и располагается под ней, потом обходит большую правую петлю. Среднюю линию нужно считывать справа налево. Обобщение до n пар петель, аналогично, требует $3^n - 1$ этапов.

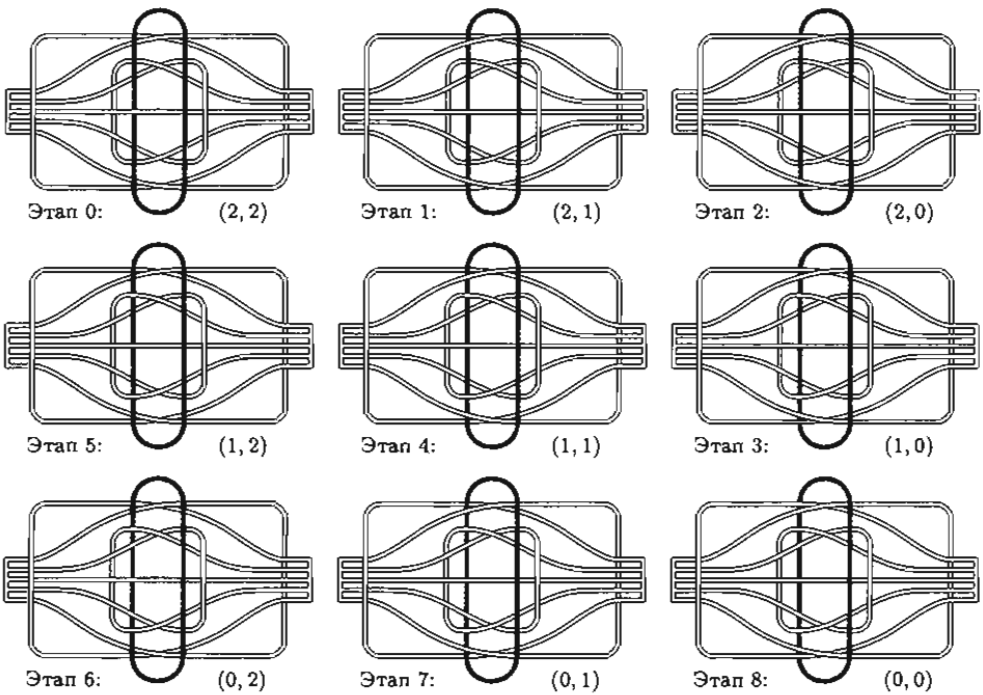


Рис. A-1.

[Откуда взялась эта замечательная головоломка, не очень ясно. В книге Джерри Слокама и Джека Ботерманса *The Book of Ingenious & Diabolical Puzzles* (1994) приведена 2-петлевая версия, вырезанная из рога, вероятно, сделанная в Китае приблизительно в 1850 году [p. 101], а также современная 6-петлевая версия, сделанная в Малайзии в 1988 году [p. 93]. У Слокама также есть 4-петлевая версия, сделанная из бамбука в Англии в 1884 году. Он нашел ее в книгах Генри Новра [см. *Catalogue of Conjuring Tricks и Puzzles* (1858 или 1859)] и В. Х. Кремера [см. *Games, Amusements, Pastimes и Magic*, 1867], а также в каталоге В. Хамли 1895 года под названием головоломка "удивительное каноэ". [См. также *Патенты США 2091191* (1937), *D172310* (1954), *3758114* (1973), *D406866* (1999).] Дыкман заметил ее связь с отраженным третичным кодом Грэя в письме к Мартину Гарднеру от 2 августа 1972 года.)

85. Согласно (50), элемент $\left[\begin{smallmatrix} b; & b' \\ t; & t' \end{smallmatrix}\right]$ произведения $\Gamma\wr\Gamma'$ имеет вид $\alpha_a\alpha'_{a'}$, если $\hat{g}(\left[\begin{smallmatrix} b; & b' \\ t; & t' \end{smallmatrix}\right]) = \left[\begin{smallmatrix} a; & a' \\ t; & t' \end{smallmatrix}\right]$ в отраженном коде Грэя для осиований (t, t'). Можно показать, что элемент $\left[\begin{smallmatrix} b; & b'; & b'' \\ t; & t'; & t'' \end{smallmatrix}\right]$ произведений $(\Gamma\wr\Gamma')\wr\Gamma''$ и $\Gamma\wr(\Gamma'\wr\Gamma'')$ является $\alpha_a\alpha'_{a'}\alpha''_{a''}$, если $\hat{g}(\left[\begin{smallmatrix} b; & b'; & b'' \\ t; & t'; & t'' \end{smallmatrix}\right]) = \left[\begin{smallmatrix} a; & a'; & a'' \\ t; & t'; & t'' \end{smallmatrix}\right]$ в отраженном коде Грэя для оснований (t, t', t''). См. упр. 4.1–10 и обратите внимание также на закон смешанного основания:

$$m_1 \ldots m_n - 1 - \begin{bmatrix} x_1, & \ldots, & x_n \\ m_1, & \ldots, & m_n \end{bmatrix} = \begin{bmatrix} m_1 - 1 - x_1, & \ldots, & m_n - 1 - x_n \\ m_1, & \ldots, & m_n \end{bmatrix}.$$

В общем, отраженный код Грэя для оснований $(m_1, \ldots, m_n)$ имеет вид $(0, \ldots, m_1 - 1)\wr\cdots\wr(0, \ldots, m_n - 1)$ [см. *Information Processing Letters* **22** (1986), p.201–205].

86. Пусть Γ_{mn} является отраженным m-м кодом Грэя, который определяется $\Gamma_{m0} = \epsilon$, и

$$\Gamma_{m(n+1)} = (0, 1, \ldots, m-1)\wr\Gamma_{mn}, \qquad n \geq 0.$$

Этот путь проходит от $(0, 0, \ldots, 0)$ до $(m-1, 0, \ldots, 0)$ для четного m. Рассмотрим путь Грэя Π_{mn}, заданный $\Pi_{m0} = \emptyset$ и

$$\Pi_{m(n+1)} = \begin{cases} (0, 1, \ldots, m-1)\wr\Pi_{mn},\ m\Gamma^R_{(m+1)n}, & \text{если } m \text{ нечетно;} \\ (0, 1, \ldots, m)\wr\Pi_{mn},\ m\Gamma^R_{mn}, & \text{если } m \text{ четно.} \end{cases}$$

Этот путь проходит все $(m+1)^n - m^n$ неотрицательные целые n-кортежи, для которых $\max(a_1, \ldots, a_n) = m$, начинаясь с $(0, \ldots, 0, m)$ и заканчиваясь $(m, 0, \ldots, 0)$. Искомый неограниченный путь Грэя имеет вид Π_{0n}, Π^R_{1n}, Π_{2n}, Π^R_{3n}, ...

87. Это невозможно для нечетных n, поскольку n-кортежи с $\max(|a_1|, \ldots, |a_n|) = 1$ содержат $\frac{1}{2}(3^n+1)$ для нечетных и $\frac{1}{2}(3^n-3)$ — для четных. Если $n = 2$, то можно использовать спираль Σ_0, Σ_1, Σ_2, ..., где Σ_m закручивается против часовой стрелки от $(m, 1-m)$ до $(m, -m)$, когда $m > 0$. Для четных значений $n \geq 2$, если T_m является путем n-кортежей от $(m, 1-m, m-1, 1-m, \ldots, m-1, 1-m)$ до $(m, -m, m, -m, \ldots, m, -m)$, то можно использовать $\Sigma_m\wr(T_0, \ldots, T_{m-1}), (\Sigma_0, \ldots, \Sigma_m)^R\,\bar{\wr}\,T_m$ для $(n+2)$-кортежей с теми же свойствами, где $\bar{\wr}$ является дуальным зигзаговым произведением (dual boustrophedon product):

$$\Gamma\,\bar{\wr}\,\Gamma' = (\alpha_0\alpha'_0, \ldots, \alpha_{t-1}\alpha'_0, \alpha_{t-1}\alpha'_1, \ldots, \alpha_0\alpha'_1, \alpha_0\alpha'_2, \ldots, \alpha_{t-1}\alpha'_2, \alpha_{t-1}\alpha'_3, \ldots).$$

(Неограниченные n-мерные коды Грэя *без* ограничения величины впервые были получены в работе Э. Вазони [см. *Acta Litterarum ac Scientiarum*, sectio Scientiarum Mathematicarum **9** (Szeged: 1938), p.163–173].)

88. Он снова посетит все подлеса, но в обратном порядке, заканчивая $(0, \ldots, 0)$ и возвращаясь к состоянию, которое было после инициализации на этапе K1. (Этот принцип отражения действительно является ключом к пониманию работы алгоритма K.)

89. (a) Пусть $M_0 = \epsilon$, $M_1 = \bullet$, и $M_{n+2} = \bullet M_{n+1}^R$, $-M_n^R$. Эта конструкция работает, поскольку последний элемент M_{n+1}^R является первым элементом M_{n+1}, а именно точкой, вслед за которой идет первый элемент M_n^R.

(b) Для заданной строки $d_1 \ldots d_l$, где каждый элемент d_j является точкой ($\bullet$) или тире ($-$), можно найти следующий элемент, предполагая $k = l - [d_l = \bullet]$ и поступая следующим образом: если k нечетно и $d_k = \bullet$, заменить $d_k d_{k+1}$ на $-$; если k четно и $d_k = -$, заменить d_k на $\bullet\bullet$; в противном случае уменьшить k на 1 и повторять до тех пор, пока не происходят изменения или не достигнуто значение $k = 0$. Следующей строкой для заданной будет $\bullet - - \bullet\bullet\bullet - \bullet - \bullet$.

90. Цикл может существовать, только если количество слов кода четно, поскольку количество тире изменяется на ± 1 на каждом этапе. Таким образом, $n \bmod 3 = 2$. Пути Грэя M_n из упр. 89 не годятся; они начинаются с $(\bullet -)^{\lfloor n/3 \rfloor} \bullet^{n \bmod 3}$ и заканчиваются $(- \bullet)^{\lfloor n/3 \rfloor} \bullet^{[n \bmod 3 = 1]} \_^{[n \bmod 3 = 2]}$. Но $M_{3k+1}\bullet$, $M_{3k}^R -$ является гамильтоновым циклом (Hamiltonian cycle) в графе кода Морзе для $n = 3k + 2$.

91. Эквивалентно n-кортежи $a_1 \bar{a}_2 a_3 \bar{a}_4 \ldots$ не содержат двух последовательных единиц. Такие n-кортежи соответствуют последовательностям кода Морзе длины $n + 1$, если присоединить 0 и затем представить $\bullet$ и $-$ соответственно элементами 0 и 10. В таком случае можно преобразовать путь M_{n+1} из упр. 89 в процедуру типа алгоритма К с "каймой" индексов, обозначающими, где начинается каждая точка или тире (за исключением финальной точки).

Q1. [Инициализировать.] Установить $a_j \leftarrow \lfloor ((j-1) \bmod 6)/3 \rfloor$ и $f_j \leftarrow j$ для $1 \le j \le n$. Также установить $f_0 \leftarrow 0$, $r_0 \leftarrow 1$, $l_1 \leftarrow 0$, $r_j \leftarrow j + (j \bmod 3)$ и $l_{j+(j \bmod 3)} \leftarrow j$ для $1 \le j \le n$, а если $j + (j \bmod 3) > n$, то установить $r_j \leftarrow 0$ и $l_0 \leftarrow j$. ("Кайма" теперь содержит 1, 2, 4, 5, 7, 8, ...)

Q2. [Посетить.] Посетить n-кортеж $(a_1, \ldots, a_n)$.

Q3. [Выбрать p.] Установить $q \leftarrow l_0$, $p \leftarrow f_q$, $f_q \leftarrow q$.

Q4. [Проверить a_p.] Завершить алгоритм, если $p = 0$. В противном случае установить $a_p \leftarrow 1 - a_p$ и перейти к Q6, если $a_p + p$ четно.

Q5. [Вставить $p + 1$.] Если $p < n$, установить $q \leftarrow r_p$, $l_q \leftarrow p + 1$, $r_{p+1} \leftarrow q$, $r_p \leftarrow p + 1$, $l_{p+1} \leftarrow p$. Перейти к Q7.

Q6. [Удалить $p + 1$.] Если $p < n$, установить $q \leftarrow r_{p+1}$, $r_p \leftarrow q$, $l_q \leftarrow p$.

Q7. [Сделать p пассивным.] Установить $f_p \leftarrow f_{l_p}$ и $f_{l_p} \leftarrow l_p$. Вернуться к Q2. ▮

Этот алгоритм также можно вывести как особый случай значительно большего общего метода Ганга Ли, Фрэнка Раски и Дональда Кнута, который расширяет алгоритм К, позволяя пользователю указать либо $a_p \ge a_q$, либо $a_p \le a_q$ для каждой пары (p, q) типа (родительский, дочерний) [см. книгу Киута и Раски, *Lecture Notes in Computer Science* **2635** (2004), p.183–204]. Обобщение в другом направлении, которое порождает все строки длины n, которые не содержат некоторых подстрок, было открыто М. Б. Сквайром [*Electronic J. Combinatorics* **3** (1996), #R17, p.1–29].

Забавно, что отображение $k \mapsto g(k)/2$ является взаимнооднозначным соответствием между всеми двоичными n-кортежами без серий единиц нечетной длины и всеми двоичными n-кортежами без серий единиц длины 2.

92. Да, поскольку граф всех $(n-1)$-кортежей $(x_1, \ldots, x_{n-1})$ с $x_1, \ldots, x_{n-1} \le m$ и с дугами $(x_1, \ldots, x_{n-1}) \to (x_2, \ldots, x_n)$ всякий раз, когда $\max(x_1, \ldots, x_n) = m$ является связным и сбалансированным; см. теорему 2.3.4.2G. Действительно, такая последовательность получается из алгоритма F, если заметить, что финальные элементы k^n простых строк длины, делящей n, при вычитании из $m - 1$ одинаковы для всех $m \ge k$. Если $n = 4$, например,

то первая 81 цифра последовательности Φ_4 имеет вид $2-\alpha^R = 0\,0001\,01\,0011\ldots$, где α является строкой (62). (Существуют также неограниченные m-е последовательности, первые m^n элементов которых являются циклами де Бруийна для всех n, для произвольного фиксированного $m \geq 3$ [см. работу Л. Дж. Каммингса и Д. Видемана *Cong. Numerantium* **53** (1986), p.155–160].)

93. Цикл, генерированный $f()$, является циклической перестановкой $\alpha 1$, где α имеет длину m^n-1 и завершается 1^{n-1}. Цикл, генерированный алгоритмом R, является циклической перестановкой $\gamma = c_0\ldots c_{m^{n+1}-1}$, где $c_k = (c_0+b_0+\cdots+b_{k-1}) \bmod m$ и $b_0\ldots b_{m^{n+1}-1} = \beta = \alpha^m 1^m$.

Если $x_0\ldots x_n$ возникает в γ, скажем, $x_j = c_{k+j}$ для $0\leq j\leq n$, тогда $y_j = b_{k+j}$ для $0\leq j<n$, где $y_j = (x_{j+1}-x_j) \bmod m$. (Это связь с модулярным m-м кодом Грэя; см. упр. 78.) Теперь, если $y_0\ldots y_{n-1} = 1^n$, то имеем: $m^{n+1}-m-n < k \leq m^{n+1}-n$; в противном случае существует такой индекс k', что $-n<k'<m^n-n$ и $y_0\ldots y_{n-1}$ возникает в β в положениях $k = (k'+r(m^n-1)) \bmod m^{n+1}$ для $0\leq r<m$. В обоих случаях m вариантов выбора k имеют разные значения x_0, поскольку сумма всех элементов в α равна $m-1$ (по модулю m) для $n\geq 2$. (Алгоритм R справедлив также для $n=1$, если $m \bmod 4 \neq 2$, поскольку $m \perp \sum\alpha$ в этом случае.)

94. 0<u>01</u>0<u>2</u>0304<u>1</u><u>2</u><u>1</u>3<u>1</u>4<u>2</u><u>2</u>3<u>2</u>43<u>4</u>344. (Подчеркнутые цифры вставлены в перемежающиеся строки 00112234 и 34. Алгоритм D можно использовать в общем случае для $n=1$ и $r = m-2\geq 0$; но это бессмыслению делать в связи с (54).)

95. (a) Пусть $c_0c_1c_2\ldots$ имеет период τ. Если τ нечетно, то имеем $p=q=r$, так что $r=pq$ только в тривиальном случае, когда $p=q=1$ и $a_0=b_0$. В противном случае $r/2 = \operatorname{lcm}(p,q) = pq/\gcd(p,q)$ согласно 4.5.2-(10), следовательно, $\gcd(p,q)=2$. В последнем случае $2n$-кортежи $c_lc_{l+1}\ldots c_{l+2n-1}$ будут иметь вид $a_jb_k\ldots a_{j+n-1}b_{k+n-1}$ для $0\leq j<p$, $0\leq k<q$, $j\equiv k$ (по модулю 2), а $b_ka_j\ldots b_{k+n-1}a_{j+n-1}$ для $0\leq j<p$, $0\leq k<q$, $j\not\equiv k$ (по модулю 2).

(b) В выводе будут перемежаться две последовательности, $a_0a_1\ldots$ и $b_0b_1\ldots$, периоды которых соответственно равны m^n+r и m^n-r. Последовательность из a образует цикл $f()$ с x^n, замененными на x^{n+1}, и последовательность из b образует цикл $f'()$ с x^n, замененными на x^{n-1} для $0\leq x<\tau$. Согласно (58) и пункту (a), период имеет длину $m^{2n}-r^2$, а каждый $2n$-кортеж возникает с исключением $(xy)^n$ для $0\leq x,y<r$.

(c) Реальный этап D6 изменяет поведение (b) на переход к D3, когда $t\geq n$, $t'=n$ и $0\leq x'=x<\tau$. Это изменение приводит к выводу дополнительного x сразу после вывода x^{2n-1} и перед выводом цифры b, которая выводится вслед за x^n в цикле. D6 также позволяет передать управление D7 и затем D3 с $t'=n$ в случае, когда $t\geq n$ и $x<x'<r$. При таком поведении дополнительное $x'x$ выводится сразу после вывода $(xx')^{n-1}x$ и перед выводом b. Эти дополнительные r^2 цифр дают r^2 недостающих $2n$-кортежей из (b).

96. (a) Рекуррентные соотношения $S_2=1$, $S_{2n+1}=S_{2n}=2S_n$, $R_2=0$, $R_{2n+1}=1+R_{2n}$, $R_{2n}=2R_n$, $D_2=0$, $D_{2n+1}=D_{2n}=1+2D_n$ приводят к решению $S_n=2^{\lfloor\lg n\rfloor-1}$, $R_n = n-2S_n$, $D_n=S_n-1$. Таким образом, имеем: $S_n+R_n+D_n=n-1$.

(b) Каждый вывод высокого уровня обычно включает $\lfloor\lg n\rfloor-1$ D-активизаций и $\nu(n)-1$ R-активизаций плюс одна базовая активизация на нижнем уровне. Но со следующим исключением: алгоритм R может вызывать свою сопрограмму $f()$ дважды, если первая активизация завершается последовательностью 1^n, а иногда алгоритму R не требуется вызывать $f()$ вовсе. Алгоритм D может вызывать свою сопрограмму $f'()$ дважды, если первая активизация завершается последовательностью $(x')^n$, а иногда алгоритму D не требуется вызывать $f()$ или $f'()$.

Алгоритм R завершает последовательность x^{n+1} тогда и только тогда, когда его дочерняя сопрограмма $f()$ только что завершила последовательность 0^n. Алгоритм D завершает

последовательность x^{2^n} для $x < r$ тогда и только тогда, когда только что совершен переход от D6 к D3 без вызова любой дочерней сопрограммой.

На основании этих наблюдений можно заключить, что по крайней мере $\lfloor \lg n \rfloor + \nu(n) + 1$ активизаций возможно на каждый вывод высокого уровня, если $r > 1$. Такой случай возникает, когда алгоритм D для $n = 6$ переходит от D6 к D4. Но для $r = 1$ можно получить $2\lfloor \lg n \rfloor + 3$ активизаций, например, когда алгоритм R для $n = 25$ переходит от R4 к R2.

97. (a) (0011), (00011101), (0000101001111011)и(0000011000101101111100111010100 1). Таким образом, $j_2 = 2$, $j_3 = 3$, $j_4 = 9$, $j_5 = 15$.

(b) Очевидно, имеем: $f_{n+1}(k) = \Sigma f_n(k) \bmod 2$ для $0 \le k < j_n + n$. Следующее значение, $f_{n+1}(j_n+n)$, зависит от того, происходит ли переход с этапа R4 на этап R2 после вычисления $y = f_n(j_n+n-1)$. Если происходит (а именно, если $f_{n+1}(j_n+n-1) \neq 0$), то имеем: $f_{n+1}(k) \equiv 1 + \Sigma(k+1)$ для $j_n + n \le k < 2^n + j_n + n$; в противном случае имеем $f_{n+1}(k) \equiv 1 + \Sigma(k-1)$ для таких значений k. В частности, $f_{n+1}(k) = 1$, когда $2^n \le k + \delta_n \le 2^n + n$. Предложенная формула, которая имеет более простые диапазоны для индекса k, справедлива, поскольку $1 + \Sigma(k \pm 1) \equiv \Sigma(k)$ для $j_n < k < j_n + n$ или $2^n + j_n < k < 2^n + j_n + n$.

(c) Перемежаемый цикл имеет $c_n(2k) = f_n^+(k)$ и $c_n(2k+1) = f_n^-(k)$, где

$$f_n^+(k) = \begin{cases} f_n(k-1), & \text{если } 0 < k \le j_n+1; \\ f_n(k-2), & \text{если } j_n+1 < k \le 2^n+2; \end{cases} \qquad f_n^-(k) = \begin{cases} f_n(k+1), & \text{если } 0 \le k < j_n; \\ f_n(k+2), & \text{если } j_n \le k < 2^n-2; \end{cases}$$

$f_n^+(k) = f_n^+(k \bmod (2^n+2))$, $f_n^-(k) = f_n^-(k \bmod (2^n-2))$. Следовательно, подпоследовательность $1^{2^{n-1}}$ начинается с положения $k_n = (2^{n-1}-2)(2^n+2) + 2j_n + 2$ в цикле c_n; это означает, что j_{2n} нечетно. Подпоследовательность $(01)^{n-1}0$ начинается с положения $l_n = (2^{n-1}+1)(j_n-1)$, если $j_n \bmod 4 = 1$, с положения $l_n = (2^{n-1}+1)(2^n+j_n-3)$, если $j_n \bmod 4 = 3$. Также $k_2 = 6$, $l_2 = 2$.

(d) Алгоритм D вставляет четыре элемента в цикл c_n. Следовательно,

$$f_{2n}(k) = \begin{cases} c_n(k-1), & \text{если } 0 < k \le l_n+2; \\ c_n(k-3), & \text{если } l_n+2 < k \le k_n+3; \\ c_n(k-4), & \text{если } k_n+3 < k \le 2^{2n}; \end{cases}$$

когда $j_n \bmod 4 < 3$ $(l_n < k_n)$;

$$= \begin{cases} c_n(k-1), & \text{если } 0 < k \le k_n+1; \\ c_n(k-2), & \text{если } k_n+1 < k \le l_n+3; \\ c_n(k-4), & \text{если } l_n+3 < k \le 2^{2n}. \end{cases}$$

когда $j_n \bmod 4 = 3$ $(k_n < l_n)$:

(e) Следовательно, $j_{2n} = k_n + 1 + 2[j_n \bmod 4 < 3]$. Действительно, элемент, предшествующий 1^{2^n}, состоит из $2^{n-2} - 1$ полных периодов $f_n^+()$, перемежаемых 2^{n-2} полными периодами $f_n^-()$, с одним вставленным элементом 0 и одним вставленным элементом 10, если $l_n < k_n$, за которым следует $f_n(1)f_n(1)f_n(2)f_n(2)\ldots f_n(j_n-1)f_n(j_n-1)$. Сумма всех этих элементов нечетна, если не выполняется условие $l_n < k_n$; следовательно, $\delta_{2n} = 1 - 2[j_n \bmod 4 = 3]$.

Пусть $n = 2^t q$, где q нечетно и $n > 2$. Эти рекуррентные последовательности подразумевают, что если $q = 1$, то имеем $j_n = 2^{n-1} + b_t$, где $b_t = 2^t/3 - (-1)^t/3$. И если $q > 1$, то имеем $j_n = 2^{n-1} \pm b_{t+2}$, где знак + выбирается тогда и только тогда, когда $\lfloor \lg q \rfloor + [\lfloor 4q/2^{\lfloor \lg q \rfloor} \rfloor = 5]$ четно.

98. Если $f(k) = g(k)$, когда k лежит в определенном диапазоне, то существует такая константа C, что $\Sigma f(k) = C + \Sigma g(k)$ для k в этом диапазоне. Следовательно, можно продолжить и легко вывести дополнительные следствия: если $n > 1$, то

$$\Sigma f_{2n}(k) \equiv \begin{cases} \Sigma c_n(k-1), & \text{если } 0 < k \le l_n+2; \\ 1+\Sigma c_n(k-3), & \text{если } l_n+2 < k \le k_n+3; \\ \Sigma c_n(k-4), & \text{если } k_n+3 < k \le 2^{2n}; \end{cases}$$

когда $j_n \bmod 4 < 3$ $(l_n < k_n)$;

$$\equiv \begin{cases} \Sigma c_n(k-1), & \text{если } 0 < k \le k_n+1; \\ 1+\Sigma c_n(k-2), & \text{если } k_n+1 < k \le l_n+3; \\ \Sigma c_n(k-4), & \text{если } l_n+3 < k \le 2^{2n}. \end{cases}$$

когда $j_n \bmod 4 = 3$ $(k_n < l_n)$:

$$\Sigma c_n(k) \equiv \Sigma f_n^+(\lceil k/2\rceil) + \Sigma f_n^-(\lfloor k/2\rfloor).$$

$$\Sigma f_n^+(k) \equiv \begin{cases} \Sigma f_n(k-1), & \text{если } 0<k\le j_n+1;\\ 1+\Sigma f_n(k-2), & \text{если } j_n+1<k\le 2^n+2;\end{cases} \quad \Sigma f_n^-(k) \equiv \begin{cases} \Sigma f_n(k+1), & \text{если } 0\le k<j_n;\\ 1+\Sigma f_n(k+2), & \text{если } j_n\le k<2^n-2;\end{cases}$$

$$\Sigma f_n^\pm(k) \equiv \lfloor k/(2^n\pm 2)\rfloor + \Sigma f_n^\pm(k \bmod (2^n\pm 2)); \quad \Sigma f_n(k) = \Sigma f_n(k \bmod 2^n).$$

$$\Sigma f_{2n+1}(k) \equiv \begin{cases} \Sigma\Sigma f_{2n}(k), & \text{если } 0<k\le j_{2n} \text{ или } 2^{2n}+j_{2n}<k\le 2^{2n+1};\\ 1+k+\Sigma\Sigma f_{2n}(k+\delta_{2n}), & \text{если } j_{2n}<k\le 2^{2n}+j_{2n}.\end{cases}$$

$\Sigma\Sigma f_{2n}(k)$, когда $j_n \bmod 4<3$ $(l_n<k_n)$:

$$\equiv \begin{cases} \Sigma\Sigma c_n(k-1), & \text{если } 0<k\le l_n+2;\\ 1+k+\Sigma\Sigma c_n(k-3), & \text{если } l_n+2<k\le k_n+3;\\ \Sigma\Sigma c_n(k-4), & \text{если } k_n+3<k\le 2^{2n};\end{cases}$$

когда $j_n \bmod 4=3$ $(k_n<l_n)$:

$$\equiv \begin{cases} \Sigma\Sigma c_n(k-1), & \text{если } 0<k\le k_n+1;\\ 1+k+\Sigma\Sigma c_n(k-2), & \text{если } k_n+1<k\le l_n+3;\\ 1+\Sigma\Sigma c_n(k-4), & \text{если } l_n+3<k\le 2^{2n}.\end{cases}$$

$$\Sigma\Sigma f_{2n}(k) \equiv [j_n \bmod 4 < 3]\lfloor k/2^{2n}\rfloor + \Sigma\Sigma f_{2n}(k \bmod 2^{2n}).$$

И тогда существует замыкание:

$$\Sigma\Sigma c_n(2k) = \Sigma f_n^+(k), \qquad \Sigma\Sigma c_n(2k+1) = \Sigma f_n^-(k).$$

Если $n = 2^t q$, когда q нечетно, то время вычисления $f_n(k)$ этим методом с использованием рекуррентной формулы равно $O(t + S(q))$, где $S(1) = 1$, $S(2k) = 1 + 2S(k)$, и $S(2k+1) = 1 + S(k)$. Ясно, что $S(k) < 2k$, поэтому вычисления содержат не более $O(n)$ простых операций с n-битовыми числами. На самом деле этот метод часто значительно быстрее. Если усреднить $S(k)$ по всем k с $\lfloor \lg k\rfloor = s$, то получим $(3^{s+1} - 2^{s+1})/2^s$, который меньше $3k^{\lg(3/2)} < 3k^{0.59}$. (Между прочим, если $k = 2^{s+1} - 1 - (2^{s-e_1} + 2^{s-e_2} + \cdots + 2^{s-e_t})$, то имеем: $S(k) = s + 1 + e_t + 2e_{t-1} + 4e_{t-2} + \cdots + 2^t e_1$.)

99. Строка, которая начинается с места k, в $f_n()$ начинается с места $k^+ = k + 1 + [k > j_n]$ в $f_n^+()$ и с места $k^- = k - 1 - [k > j_n]$ в $f_n^-()$, за исключением того, что 0^n и 1^n встречаются дважды в $f_n^+()$, но их нет совсем в $f_n^-()$.

Чтобы найти $\gamma = a_0 b_0 \ldots a_{n-1} b_{n-1}$ в цикле $f_{2n}()$, допустим, что $\alpha = a_0 \ldots a_{n-1}$ и $\beta = b_0 \ldots b_{n-1}$. Допустим, что α начинается с места j, а β — с места k в $f_n()$. Предположим, что ни α, ни β не равно 0^n или 1^n. Если $j^+ \equiv k^+$ (по модулю 2), предположим, что $l/2$ является решением уравнения $j^+ + (2^n+2)x = k^- + (2^n-2)y$. Можно получить $l/2 = k + (2^n - 2)(2^{n-3}(j-k) \bmod (2^{n-1}+1))$, если $j \ge k$, а в противном случае $l/2 = j + (2^n+2)(2^{n-3}(k-j) \bmod (2^{n-1}-1))$. В противном случае допустим, что $(l-1)/2 = k^+ + (2^n+2)x = j^- + (2^n-2)y$. Тогда γ начинается с места l в цикле $c_n()$, т.е. с положения $l + 1 + [l \ge k_n] + 2[l \ge l_n]$ в цикле $f_{2n}()$. Аналогичные формулы справедливы, когда справедливо $\alpha \in \{0^n, 1^n\}$ или $\beta \in \{0^n, 1^n\}$ (но не оба сразу). Наконец, 0^{2n}, 1^{2n}, $(01)^n$ и $(10)^n$ начинаются соответственно с мест 0, j_{2n}, $l_n + 1 + [k_n < l_n]$, и $l_n + 2 + [k_n < l_n]$.

Чтобы найти $\beta = b_0 b_1 \ldots b_n$ в $f_{n+1}()$, когда n четно, допустим, что n-битовая строка $(b_0 \oplus b_1)\ldots(b_{n-1} \oplus b_n)$ начинается с места j в $f_n()$. Тогда β начинается с места $k = j - \delta_n[j \ge j_n] + 2^n[j = j_n][\delta_n = 1]$, если $f_{n+1}(k) = b_0$; в противном случае в месте $k + (2^n - \delta_n, \delta_n, 2^n + \delta_n)$, согласно $(j<j_n, j=j_n, j>j_n)$.

Время выполнения этой рекурсии равно $T(n) = O(n) + 2T(\lfloor n/2\rfloor)$, т.е. оно порядка $O(n \log n)$. (Упр. 97–99 основаны на работе Дж. Тулиаии, который также развил методы для некоторых больших значений m [см. *Discrete Math.* **226** (2001), p.313–336].)

100. Здесь нет очевидных дефектов; прежде чем рекомендовать любую последовательность, все равно необходима тщательная проверка. Наоборот, цикл де Бруийна, который создается неявно в алгоритме F, является неудачным источником предположительно случайных битов, даже если он имеет n-распределение в смысле определения 3.5D, поскольку 0s доминирует в начале. Действительно, если n простое число, то биты $tn + 1$ этой последовательности являются нулями для $0 \le t < (2^n - 2)/n$.

101. (a) Пусть β является суффиксом $\lambda\lambda'$ с $\beta \le \lambda\lambda'$. Либо β является суффиксом λ', откуда $\lambda < \lambda' \le \beta$ либо $\beta = \alpha\lambda'$, и имеем $\lambda < \alpha < \beta$.

Теперь $\lambda < \beta \le \lambda\lambda'$ подразумевает, что $\beta = \lambda\gamma$ для некоторых $\gamma \le \lambda'$. Но γ является суффиксом β с $1 \le |\gamma| = |\beta| - |\lambda| < |\lambda'|$; следовательно, γ является суффиксом λ' и $\lambda' < \gamma$. Противоречие.

(b) Произвольная строка длины 1 является простой. Комбинируем смежные простые строки, согласно (a), в произвольном порядке, пока нельзя будет получить никакой другой комбинации. [См. описание более общих результатов в работе М. П. Шутценбергера; *Proc. Amer. Math. Soc.* **16** (1965), p.21–24.]

(c) Если $t \ne 0$, то пусть λ является наименьшим суффиксом $\lambda_1 \dots \lambda_t$. Тогда λ является простой строкой по определению и имеет вид $\beta\gamma$, где β является непустым суффиксом некоторой строки λ_j. Следовательно, $\lambda_t \le \lambda_j \le \beta \le \beta\gamma = \lambda \le \lambda_t$, так что получим $\lambda = \lambda_t$. Удалим λ_t и будем повторять до тех пор, пока не будет выполняться условие $t = 0$.

(d) Истинно. Если $\alpha = \lambda\beta$ для некоторой простой строки λ с $|\lambda| > |\lambda_1|$, то можно было бы присоединить множители β для получения другого разложения на простые множители α.

(e) $3 \cdot 14159265358979323846264338327\,95 \cdot 02884197$. (Эффективный алгоритм представлен в упр. 106. Знание большего числа цифр π не меняет первые два множителя. Бесконечное разложение на десятичные дроби любого числа, которое является "нормальным" в смысле Э. Бореля (см. раздел 3.5), дает простые числа ограниченной длины.)

102. Имеем: $1/(1 - mz) = 1/\prod_{n=1}^{\infty}(1 - z^n)^{L_m(n)}$. Это подразумевает (60), как в упр. 4.6.2–4.

103. Если $n = p$ — простое, то из (59) следует, что $L_m(1) + pL_m(p) = m^p$, и имеем: $L_m(1) = m$. (Это комбинаторное доказательство забавно контрастирует с традиционным алгебраическим доказательством теоремы 1.2.4F.)

104. 4483 непростыми строками являются `abaca`, `agora`, `ahead`, ...; 1274 простыми строками являются ..., `rusts`, `rusty`, `rutty`. (Поскольку строка `prime` (простая) не является простой, то, вероятно, правильнее было бы называть простую строку термином `lowly` (скромная).)

105. (a) Пусть α' является α с увеличенной последней буквой, и предположим, что $\alpha' = \beta\gamma'$, где $\alpha = \beta\gamma$ и $\beta \ne \epsilon$, $\gamma \ne \epsilon$. Пусть θ является префиксом α, причем $|\theta| = |\gamma|$. Согласно предположению, существует такая строка ω, что $\alpha\omega$ является простой. Следовательно, $\theta \le \alpha\omega < \gamma\omega$, так что имеем $\theta \le \gamma$. Далее $\theta < \gamma'$, и имеем $\alpha' < \gamma'$.

(b) Пусть $\alpha = \lambda_1\beta = a_1 \dots a_n$, где $\lambda_1\beta\omega$ является простой. Условие $\lambda_1\beta\omega < \beta\omega$ подразумевает, что $a_j \le a_{j+r}$ для $1 \le j \le n - r$, где $r = |\lambda_1|$. Но не может быть $a_j < a_{j+r}$, в противном случае α начиналась бы с простой строки длиннее λ_1, что противоречит упр. 101(d).

(c) Если α является n-расширением обеих строк λ и λ', где $|\lambda| > |\lambda'|$, то получим $\lambda = (\lambda')^q\theta$, где θ является непустым префиксом λ'. Но тогда $\theta \le \lambda' < \lambda < \theta$.

106. **B1.** [Инициализировать.] Установить $a_1 \leftarrow \cdots \leftarrow a_n \leftarrow m - 1$, $a_{n+1} \leftarrow -1$, и $j \leftarrow 1$.

B2. [Посетить.] Посетить $(a_1, \dots, a_n)$ с индексом j.

B3. [Вычесть единицу.] Завершить, если $a_j = 0$. В противном случае установить $a_j \leftarrow a_j - 1$ и $a_k \leftarrow m - 1$ для $j < k \le n$.

B4. [Подготовиться к разложению.] (Согласно упр. 105(b), нужно найти первый простой множитель λ_1 of $a_1 \dots a_n$.) Установить $j \leftarrow 1$ и $k \leftarrow 2$.

B5. [Найти новое j.] (Теперь $a_1 \dots a_{k-1}$ является $(k-1)$-расширением простой строки $a_1 \dots a_j$.) Если $a_{k-j} > a_k$, то вернуться к B2. В противном случае, если $a_{k-j} < a_k$, то установнть $j \leftarrow k$. Затем увеличить k на 1 и новторить этот этап. ▮

Эффективный алгоритм разложения на простые множители на этапах B4 и B5 принадлежит Дж. П. Дювалю [см. *J. Algorithms* **4** (1983), p.363–381]. Более подробную информацию можно найти в работе Каттелла, Раскн, Савада, Серра и Миерса [см. *J. Algorithms* **37** (2000), p.267–282].

107. Количество посещенных n-кортежей равно: $P_m(n) = \sum_{j=1}^{n} L_m(j)$. Поскольку $L_m(n) = \frac{1}{n}m^n + O(m^{n/2}/n)$, то имеем: $P_m(n) = Q(m,n) + O(Q(\sqrt{m},n))$, где

$$Q(m,n) = \sum_{k=1}^{n} \frac{m^k}{k} = \frac{m^n}{n} R(m,n);$$

$$R(m,n) = \sum_{k=0}^{n-1} \frac{m^{-k}}{1-k/n} = \sum_{k=0}^{n/2} \frac{m^{-k}}{1-k/n} + O(nm^{-n/2})$$

$$= \frac{m}{m-1} \sum_{j=0}^{t-1} \frac{1}{n^j} \sum_{l} \left\{ {j \atop l} \right\} \frac{l!}{(m-1)^l} + O(n^{-t}).$$

Таким образом, $P_m(n) \sim m^{n+1}/((m-1)n)$. Основной вклад во время выполнения дают циклы на этапах F3 и F5, затраты на которые равны $n-j$ для каждой простой строки длины j, следовательно, общее время выполнения равно: $nP_m(n) - \sum_{j=1}^{n} jL_m(j) = m^{n+1}(1/((m-1)^2 n) + O(1/(mn^2)))$. Это меньше, чем время, необходимое для вывода отдельных цифр m^n цикла де Бруийна.

108. (a) Если $\alpha \neq 9\ldots9$, то имеем $\lambda_{k+1} \leq \beta 9^{|\alpha|}$, поскольку последняя строка является простой.

(b) Предположим, что β состоит не только из нулей, поскольку $9^j 0^{n-j}$ является подстрокой $\lambda_{t-1}\lambda_t\lambda_1\lambda_2 = 89^n0^n1$. Пусть значение k минимально для $\beta \leq \lambda_k$, тогда $\lambda_k \leq \beta\alpha$, поэтому β является префиксом λ_k. Поскольку β является предпростой строкой, то она является $|\beta|$-расширением некоторой простой строки $\beta' \leq \beta$. Предпростая строка, посещенная в алгоритме F сразу перед β', имеет вид $(\beta'-1)9^{n-|\beta'|}$, согласно упр. 106, где $\beta'-1$ обозначает десятичное число, которое на единицу меньше β'. Таким образом, если β' не является λ_{k-1}, то подсказка (которая также следует из упр. 106) состоит в том, что λ_{k-1} заканчивается по крайней мере строкой $n-|\beta'| \geq n-|\beta|$ 9s, а α является суффиксом λ_{k-1}. С другой стороны, если $\beta' = \lambda_{k-1}$, то α является суффиксом λ_{k-2} и β является суффиксом $\lambda_{k-1}\lambda_k$.

(c) Если $\alpha \neq 9\ldots9$, то имеем: $\lambda_{k+1} \leq (\beta\alpha)^{d-1}\beta 9^{|\alpha|}$, поскольку последняя строка является простой. В противном случае λ_{k-1} заканчивается по крайней мере строкой $(d-1)|\beta\alpha|$ 9s, и $\lambda_{k+1} \leq (\beta\alpha)^{d-1}9^{|\beta\alpha|}$, а $(\alpha\beta)^d$ является подстрокой of $\lambda_{k-1}\lambda_k\lambda_{k+1}$.

(d) Внутри простых строк 135899135914, 787899787979, 129999131313 14, 09090911, 089999 09 090911, 118999 119 119122.

(e) Да. Во всех случаях положение $a_1\ldots a_n$ предшествует положению подстроки $a_1\ldots a_{n-1}(a_n+1)$, если $0 \leq a_n < 9$ (и если предположить, что строки like 9^j0^{n-j} появляются в начале). Более того, 9^j0^{n-j-1} возникает только после того, как $9^{j-1}0^{n-j}a$ появится для $1 \leq a \leq 9$, поэтому не нужно располагать 0 после 9^j0^{n-j-1}.

109. Предположим, что нужно найти подматрицу

$$\begin{pmatrix} (w_{n-1}\ldots w_1w_0)_2 & (x_{n-1}\ldots x_1x_0)_2 \\ (y_{n-1}\ldots y_1y_0)_2 & (z_{n-1}\ldots z_1z_0)_2 \end{pmatrix}.$$

Двоичный случай $n=1$ уже рассмотрен в условии, а если $n>1$, то по индукции можно предположить, что необходимо только определить ведущие биты a_{2n-1}, a_{2n-2}, b_{2n-1}

и b_{2n-2}. Случай $n = 3$ является типичным: нужно решить следующую систему уравнений:

$$\begin{aligned}
b_5 &= w_2, & b_4 &= x_2, & a_5 \oplus b_5 &= y_2, & a_4 \oplus b_4 &= z_2, & &\text{если } a_0 = 0,\ b_0 = 0;\\
b_4 &= w_2, & b'_5 &= x_2, & a_4 \oplus b_4 &= y_2, & a_5 \oplus b'_5 &= z_2, & &\text{если } a_0 = 0,\ b_0 = 1;\\
a_5 \oplus b_5 &= w_2, & a_4 \oplus b_4 &= x_2, & b_5 &= y_2, & b_4 &= z_2, & &\text{если } a_0 = 1,\ b_0 = 0;\\
a_4 \oplus b_4 &= w_2, & a_5 \oplus b'_5 &= x_2, & b_4 &= y_2, & b'_5 &= z_2, & &\text{если } a_0 = 1,\ b_0 = 1;
\end{aligned}$$

здесь $b'_5 = b_5 \oplus b_4 b_3 b_2 b_1$ учитывает перенос, когда j становится равным $j + 1$.

110. Пусть $a_0 a_1 \ldots a_{m^2-1}$ является таким m-м циклом де Бруийна, как первые m^2 элементов (54). Если m нечетно, пусть $d_{ij} = a_j$, когда i четно, а $d_{ij} = a_{(j+(i-1)/2 \bmod m^2}$, когда i нечетно. (Первым среди многих людей, открывших эту конструкцию, похоже, был Джон К. Кок, который также создал торы де Бруийна других форм и размеров [*Discrete Math.* **70** (1988), p.209–210].)

Если $m = m'm''$, где $m' \perp m''$, используем китайскую теорему об остатках для определения

$$d_{ij} \equiv d'_{ij} \text{ (по модулю } m') \qquad \text{и} \qquad d_{ij} \equiv d''_{ij} \text{ (по модулю } m'')$$

на основе матриц, которые решили эту задачу для m' и m''. Таким образом, предыдущее упр. приводит к решению для произвольного m.

Другое интересное решение для четных m было найдено Анталом Ивани и Золтаном Тотом [см. *2nd Conf. Automata, Languages, and Programming Systems* (1988), p.165–172; а также работу Херльберта и Айзека, *Contemp. Math.* **178** (1994), p.153–160]. Первые m^2 элементов a_j бесконечной последовательности

$$0011\,021331203223\,041524355342514054 45\,0617263746577564 \ldots 07667\,08 \ldots$$

определяют цикл де Бруийна с тем свойством, что расстояние между появлениями ab и ba всегда четно. Затем можно допустить, что $d_{ij} = a_j$, если $i + j$ четно, а $d_{ij} = a_i$, если $i + j$ нечетно. Например, когда $m = 4$, имеем:

$$\begin{pmatrix}
0&0&1&0&0&2&1&2&2&0&3&0&2&2&3&2\\
0&0&0&1&0&2&0&3&2&0&2&1&2&2&2&3\\
0&1&1&1&0&3&1&3&2&1&3&1&2&3&3&3\\
1&0&1&1&1&2&1&3&3&0&3&1&3&2&3&3\\
0&0&1&0&0&2&1&2&2&0&3&0&2&2&3&2\\
0&2&0&3&0&0&0&1&2&2&2&3&2&0&2&1\\
0&1&1&1&0&3&1&3&2&1&3&1&2&3&3&3\\
1&2&1&3&1&0&1&1&3&2&3&3&3&0&3&1\\
0&0&1&0&0&2&1&2&2&0&3&0&2&2&3&2\\
2&0&2&1&2&2&2&3&0&0&0&1&0&2&0&3\\
0&1&1&1&0&3&1&3&2&1&3&1&2&3&3&3\\
3&0&3&1&3&2&3&3&1&0&1&1&1&2&1&3\\
0&0&1&0&0&2&1&2&2&0&3&0&2&2&3&2\\
2&2&2&3&2&0&2&1&0&2&0&3&0&0&0&1\\
0&1&1&1&0&3&1&3&2&1&3&1&2&3&3&3\\
3&2&3&3&3&0&3&1&1&2&1&3&1&0&1&1
\end{pmatrix}$$

(упр. 109);

$$\begin{pmatrix}
0&0&1&0&0&0&1&0&3&0&2&0&3&0&2&0\\
0&0&0&1&0&2&0&3&0&1&0&0&0&2&0&3\\
0&1&1&1&0&1&1&1&3&1&2&1&3&1&2&1\\
1&0&1&1&1&2&1&3&1&1&1&0&1&2&1&3\\
0&0&1&0&0&0&1&0&3&0&2&0&3&0&2&0\\
2&0&2&1&2&2&2&3&2&1&2&0&2&2&2&3\\
0&1&1&1&0&1&1&1&3&1&2&1&3&1&2&1\\
3&0&3&1&3&2&3&3&3&3&1&3&0&3&2&3\\
0&3&1&3&0&3&1&3&3&3&3&2&3&3&2&3\\
1&0&1&1&1&2&1&3&1&1&1&0&1&2&1&3\\
0&2&1&2&0&2&1&2&3&2&2&2&2&3&2&2\\
0&0&0&1&0&2&0&3&0&1&0&0&0&2&0&3\\
0&3&1&3&0&3&1&3&3&3&3&2&3&3&2&3\\
2&0&2&1&2&2&2&3&2&1&2&0&2&2&2&3\\
0&2&1&2&0&2&1&2&3&2&2&2&2&3&2&2\\
3&0&3&1&3&2&3&3&3&1&3&0&3&2&3&3
\end{pmatrix}$$

(Тот).

111. (a) Пусть $d_j = j$ и $0 \le a_j < 3$ для $1 \le j \le 9$, $a_9 \ne 0$. Образуем последовательности s_j, t_j по правилам $s_1 = 0$, $t_1 = d_1$; $t_{j+1} = d_{j+1} + 10t_j[a_j = 0]$ для $1 \le j < 9$; $s_{j+1} = s_j + (0, t_j, -t_j)$ для $a_j = (0, 1, 2)$ и $1 \le j \le 9$. Тогда s_{10} является возможным результатом, и нам нужно только запомнить наименьшие значения, которые возникают. Более половины работы можно сэкономить, допуская $a_k = 2$, когда $s_k = 0$, используя $|s_{10}|$ вместо s_{10}.

Поскольку нужно проверить меньше $3^8 = 6561$ возможностей, то здесь разумно было бы применить грубую силу на основе третичной версии алгоритма М. Потребуется меньше 24 000 мемсов и 1600 умножений для вывода результата, что так можно представнть все целые числа меньше 211, но не 211.

Другой подход на основе использования кода Грэя для варьирования знаками после разбиения цифр на блоки 2^8 возможными способами сокращает количество умножений до 255, но за счет 500 дополнительных мемсов. Следовательно, код Грэя не дает преимущества в этом примере.

(b) В этом случае (за счет 73 000 мемсов и 4900 умножений) таким образом можно представить все числа меньше 241, но не 241. Есть 46 способов представления 100, включая замечательный способ $9 - 87 + 6 + 5 - 43 + 210$.

(Впервые эту "задачу века" сформулировал Г. Э. Дьюдени [*The Weekly Dispatch* (4 and 18 June 1899)]; см. также работы Мартина Гарднера [*The Numerology of Dr. Matrix*, Chapter 6] и Стивена Каана [*J. Recreational Math.* **23** (1991), p.19–25].)

112. Для выполнения метода нз упр. 111 теперь потребуется больше 167 миллионов мемсов и 10 миллионов умножений, поскольку 3^{16} гораздо больше 3^8. Этот метод можно значительно улучшить (до 10,4 миллиона мемсов н 1100 умножений), табулируя возможиости, полученные для первых k и последних k цифр, для $1 \le k < 9$, а затем рассматривая все блоки цифр, в которых используется 9. Всего существует 60318 способов представления 100, а первым числом, для которого нельзя найти такое представление, является число 16040.

РАЗДЕЛ 7.2.1.2

1. (Дж. П. Н. Филипс.) [J. P. N. Phillips, *Comp. J.* **10** (1967), p.311.] Предполагая, что $n \geq 3$, можно заменить этапы L2–L4 следующими.

L2′. [Простейший случай?] Установить $y \leftarrow a_{n-1}$ и $z \leftarrow a_n$. Если $y < z$, то установить $a_{n-1} \leftarrow z$, $a_n \leftarrow y$ и вернуться к L1.

L2.1′. [Следующий простейший случай?] Установить $x \leftarrow a_{n-2}$. Если $x \geq y$, то перейти к этапу L2.2′. В противном случае установить $(a_{n-2}, a_{n-1}, a_n) \leftarrow (z, x, y)$, если $x < z$, либо $(a_{n-2}, a_{n-1}, a_n) \leftarrow (y, z, x)$, если $x \geq z$. Вернуться к L1.

L2.2′. [Найти j.] Установить $j \leftarrow n-3$ и $y \leftarrow a_j$. Если $y \geq x$, то установить $j \leftarrow j-1$, $x \leftarrow y$, $y \leftarrow a_j$ и повторять, пока не будет выполняться условие $y < x$. Завершить, если $j = 0$.

L3′. [Легко увеличить?] Если $y < z$, то установить $a_j \leftarrow z$, $a_{j+1} \leftarrow y$, $a_n \leftarrow x$ и перейти к L4.1′.

L3.1′. [Увеличить a_j.] Установить $l \leftarrow n-1$; если $y \geq a_l$, повторно уменьшить l на 1, пока не будет выполняться условие $y < a_l$. Затем установить $a_j \leftarrow a_l$ и $a_l \leftarrow y$.

L4′. [Начать в обратную сторону.] Установить $a_n \leftarrow a_{j+1}$ и $a_{j+1} \leftarrow z$.

L4.1′. [Обратить $a_{j+2} \ldots a_{n-1}$.] Установить $k \leftarrow j+2$, $l \leftarrow n-1$. Затем, если $k < l$, поменять $a_k \leftrightarrow a_l$, установить $k \leftarrow k+1$, $l \leftarrow l-1$ и повторять, пока не будет выполняться условие $k \geq l$. Вернуться к L1. ▮

Программа может выполняться еще быстрее, если a_t хранить в памяти по адресу $\mathtt{A}[n-t]$ для $0 \leq t \leq n$ или если использовать обратный колексикографический порядок, как в следующем упражнении.

2. Снова предположим, что в исходном состоянии $a_1 \leq a_2 \leq \cdots \leq a_n$. Однако перестановки, генерированные из $\{1, 2, 2, 3\}$, будут иметь вид 1223, 2123, 2213, ..., 2321, 3221. Пусть a_{n+1} является вспомогательным элемеитом, который *больше*, чем a_n.

L1. [Посетить.] Посетить перестановку $a_1 a_2 \ldots a_n$.

L2. [Найти j.] Установить $j \leftarrow 2$. Если $a_{j-1} \geq a_j$, то увеличить j на 1, пока не будет выполняться условие $a_{j-1} < a_j$. Завершить, если $j > n$.

L3. [Умеиьшить a_j.] Устаиовить $l \leftarrow 1$. Если $a_l \geq a_j$, то увеличить l, пока не будет выполняться условие $a_l < a_j$. Затем обменять местами $a_l \leftrightarrow a_j$.

L4. [Обратить $a_1 \ldots a_{j-1}$.] Установить $k \leftarrow 1$ и $l \leftarrow j-1$. Затем, если $k < l$, обменять $a_k \leftrightarrow a_l$, установить $k \leftarrow k+1$, $l \leftarrow l-1$ и повторять, пока не будет выполняться условие $k \geq l$. Вернуться к L1. ▮

3. Пусть $C_1 \ldots C_n = c_{a_1} \ldots c_{a_n}$ является таблицей обратного преобразования, как в упр. 5.1.1–7. Тогда рангом $\operatorname{rank}(a_1 \ldots a_n)$ является число со смешанным основанием $\left[\begin{smallmatrix} C_1, & \ldots, & C_{n-1}, & C_n \\ n, & \ldots, & 2, & 1 \end{smallmatrix}\right]$. [См. работу Х. А. Роте (H. A. Rothe) *Sammlung combinatorisch-analytischer Abhandlungen* **2** (1800), p.263–264, а также пионерскую работу Пандита Нарайана (Paṇḍita Nārāyaṇa), ссылки на которую приводились в разделе 7.2.1.7.] Например, 314592687 имеет ранг $\left[\begin{smallmatrix} 2, & 0, & 1, & 1, & 4, & 0, & 0, & 1, & 0 \\ 9, & 8, & 7, & 6, & 5, & 4, & 3, & 2, & 1 \end{smallmatrix}\right] = 2 \cdot 8! + 6! + 5! + 4 \cdot 4! + 1! = 81577$; это факториальная система счисления, представленная в уравнении 4.1–(10).

4. Используйте рекуррентное соотношение $\operatorname{rank}(a_1 \ldots a_n) = \frac{1}{n} \sum_{j=1}^{l} n_j [x_j < a_1] \binom{n}{n_1, \ldots, n_l} + \operatorname{rank}(a_2 \ldots a_n)$. Например, $\operatorname{rank}(314159265)$ равен:

$$\tfrac{3}{9}\binom{9}{2,1,1,1,2,1,1} + 0 + \tfrac{2}{7}\binom{7}{1,1,1,2,1,1} + 0 + \tfrac{1}{5}\binom{5}{1,2,1,1} + \tfrac{3}{4}\binom{4}{1,1,1,1} + 0 + \tfrac{1}{2}\binom{2}{1,1} = 30991.$$

5. (а) Этап L2 выполняется $n!$ раз. Вероятность того, что выполняется точно k сравнений, равна $q_k - q_{k+1}$, где q_t — это вероятность того, что $a_{n-t+1} > \cdots > a_n$, а именно $[t \leq n]/t!$.

Следовательно, среднее равно $\sum k(q_k - q_{k+1}) = q_1 + \cdots + q_n = \lfloor n!\,e \rfloor/n! - 1 \approx e - 1 \approx 1{,}718$, а дисперсия равна:

$$\sum k^2(q_k - q_{k+1}) - \text{среднее}^2 = q_1 + 3q_2 + \cdots + (2n-1)q_n - (q_1 + \cdots + q_n)^2 \approx e(3-e) \approx 0{,}766.$$

(О более высоких моментах, см. работу Р. Кемпа [*Acta Informatica* **35** (1998), p.17–89, Theorem 4].)

Кстати, отсюда следует, что среднее число операций замены на этапе L4 равно: $\sum\lfloor k/2\rfloor(q_k - q_{k+1}) = q_2 + q_4 + \cdots \approx \cosh 1 - 1 = (e + e^{-1} - 2)/2 \approx 0.543$. Этот результат принадлежит Р. Дж. Орд-Смиту (R. J. Ord-Smith) [см. *Comp. J.* **13** (1970), p.152–155].

(b) Этап L3 выполняется только $n! - 1$ раз, но предположим для удобства, что он выполняется на один раз больше (без сравнений). Тогда вероятность того, что выполняется точно k сравнений, равна $\sum_{j=k+1}^{n} 1/j!$ для $1 \le k < n$ и $1/n!$ — для $k = 0$. Следовательно, среднее равно: $\frac{1}{2}\sum_{j=0}^{n-2} 1/j! \approx e/2 \approx 1{,}359$; согласно упр. 1, это число сокращается на $\frac{2}{3}$. Дисперсия равна: $\frac{1}{3}\sum_{j=0}^{n-3} 1/j! + \frac{1}{2}\sum_{j=0}^{n-2} 1/j! - \text{среднее}^2 \approx \frac{5}{6}e - \frac{1}{4}e^2 \approx 0{,}418$.

6. (a) Пусть $e_n(z) = \sum_{k=0}^{n} z^k/k!$; тогда количество разных префиксов $a_1 \ldots a_j$ равно: $j!\,[z^j]\,e_{n_1}(z)\ldots e_{n_t}(z)$. Это в $N = \binom{n_1,\ \ldots,\ n_t}{n}$ раз больше вероятности q_{n-j} того, что по крайней мере $n - j$ сравнений выполняются на этапе L2. Следовательно, среднее равно: $\frac{1}{N}w(e_{n_1}(z)\ldots e_{n_t}(z)) - 1$, где $w(\sum x_k z^k/k!) = \sum x_k$. В двоичном случае среднее равно: $M/\binom{n}{s} - 1$, где $M = \sum_{l=0}^{s}\sum_{k=l}^{n-s+l}\binom{k}{l} = \sum_{l=0}^{s}\binom{n-s+l+1}{l+1} = \binom{n+2}{s+1} - 1 = \binom{n}{s}\left(2 + \frac{s}{n-s+1} + \frac{n-s}{s+1}\right) - 1$.

(b) Если $\{a_1, \ldots, a_j\} = \{n'_1 \cdot x_1, \ldots, n'_t \cdot x_t\}$, то префикс $a_1 \ldots a_j$ дает вклад $\sum_{1\le k<l\le t}(n_k - n'_k)[n_l > n'_l]$ в общее количество сравнений, выполняемых на этапе L3. Таким образом, среднее равно: $\frac{1}{N}\sum_{1\le k<l\le t} w(f_{kl}(z))$, где

$$f_{kl}(z) = \Bigg(\prod_{\substack{1\le m\le t\\ m\ne k,\ m\ne l}} e_{n_m}(z)\Bigg)\Bigg(\sum_{r=0}^{n_k}(n_k - r)\frac{z^r}{r!}\Bigg)e_{n_l-1}(z)$$

$$= e_{n_1}(z)\ldots e_{n_t}(z)(n_k - z\tau_k(z))\tau_l(z), \qquad \text{где } \tau_k(z) = \frac{e_{n_k-1}(z)}{e_{n_k}(z)}.$$

В двузначном случае эта формула сводится к $\frac{1}{N}w\big((se_s(z) - ze_{s-1}(z))e_{n-s-1}(z)\big) = \frac{s}{N}\left(\binom{n+1}{s+1} - 1\right) - \frac{1}{N}\left(\binom{n+1}{s+1}\left(s - \frac{s+1}{n-s+1}\right) + 1\right) = \frac{1}{N}\left(-s - 1 + \binom{n+1}{s}\right) = \frac{n+1}{n-s+1} - \frac{s+1}{N}$.

7. В обозначениях ответа к предыдущему упражнению $\frac{1}{N}w(e_{n_1}(z)\ldots e_{n_t}(z)) - 1$ равно:

$$\frac{n_1 + \cdots + n_t}{n} + \frac{(n_1n_2 + n_1n_3 + \cdots + n_{t-1}n_t) + n_1(n_1-1) + \cdots + n_t(n_t-1)}{n(n-1)} + \cdots.$$

Используя уравнение 1.2.9–(38), можно показать, что предел равен: $-1 + \exp\sum_{k\ge1}\tau_k/k$, где $\tau_k = \lim_{t\to\infty}(n_1^k + \cdots + n_t^k)/(n_1 + \cdots + n_t)^k$. В случаях (a) и (b) имеем $\tau_k = [k=1]$, поэтому предел равен: $e - 1 \approx 1{,}71828$. В случае (c) имеем $\tau_k = 1/(2^k - 1)$, поэтому предел равен $-1 + \exp\sum_{k\ge1} 1/(k(2^k - 1)) \approx 2{,}46275$.

8. Допустим, что j в исходном состоянии равно нулю, и заменим этап L1 на

L1′. [Посетить.] Посетить вариацию $a_1 \ldots a_j$. Если $j < n$, то установить $j \leftarrow j + 1$ и повторить этот этап. ∎

Этот алгоритм предложен Л. Дж. Фишером (L. J. Fischer) и К. К. Краузе (K. C. Krause) [см. *Lehrbuch der Combinationslehre und der Arithmetik* (Dresden, 1812), p.55–57].

Кстати, общее число вариаций равно $w(e_{n_1}(z)\ldots e_{n_t}(z))$ в обозначениях ответа к упр. 6. Эта задача была впервые рассмотрена Джеймсом Бернулли (J. Bernoulli) [*Ars Conjectandi* (1713), Part 2, Chapter 9].

9. Предположим, что $r > 0$ и начнем с $a_0 < a_1 \le a_2 \le \cdots \le a_n$. **R1.** [Посетить.] Посетить вариацию $a_1 \ldots a_r$. (В этом месте $a_{r+1} \le \cdots \le a_n$.)

R2. [Простой случай?] Если $a_r < a_n$, то заменить $a_r \leftrightarrow a_j$, где j является таким наименьшим индексом, что $j > r$ и $a_j > a_r$, и вернуться к R1.

R3. [Обратить.] Установить $(a_{r+1}, \ldots, a_n) \leftarrow (a_n, \ldots, a_{r+1})$, как на этапе L4.

R4. [Найти j.] Установить $j \leftarrow r-1$. Если $a_j \ge a_{j+1}$, то уменьшить j на 1 повторно, пока не будет выполняться условие $a_j < a_{j+1}$. Завершить, если $j = 0$.

R5. [Увеличить a_j.] Установить $l \leftarrow n$. Если $a_j \ge a_l$, то уменьшить l на 1 повторно, пока не будет выполняться условие $a_j < a_l$. Потом заменить $a_j \leftrightarrow a_l$.

R6. [Снова обратить.] Установить $(a_{j+1}, \ldots, a_n) \leftarrow (a_n, \ldots, a_{j+1})$, как на этапе L4, и вернуться к R1. ▮

Количество вариаций равно $r![z^r]\,e_{n_1}(z)\ldots e_{n_t}(z)$. Конечно, если элементы различны, то оно равно n^r.

10. $a_1a_2\ldots a_n = 213\ldots n$, $c_1c_2\ldots c_n = 010\ldots 0$, $o_1o_2\ldots o_n = 1(-1)1\ldots 1$, если $n \ge 2$.

11. Этап (P1, ..., P7) выполняется $(1, n!, n!, n!+x_n, n!, (x_n+3)/2, x_n)$ раз, где $x_n = \sum_{k=1}^{n-1} k!$, поскольку P7 выполняется $(j-1)!$ раз, если $2 \le j \le n$.

12. Нужно найти перестановку с рангом 999999. Вот ответы: (a) 2783915460, согласно упр. 3; (b) 8750426319, поскольку отраженным числом со смешанным основанием для $\left[\begin{smallmatrix}0,&0,&1,&2,&3,&0,&2,&7,&0,&9\\1,&2,&3,&4,&5,&6,&7,&8,&9,&10\end{smallmatrix}\right]$ является $\left[\begin{smallmatrix}0,&0,&1,&3-2,&3,&5-0,&2,&7,&8-0,&9-9\\1,&2,&3,&4,&5,&6,&7,&8,&9,&10\end{smallmatrix}\right]$, согласно 7.2.1.1–(50); (c) произведение $(0\ 1\ \ldots\ 9)^9(0\ 1\ \ldots\ 8)^0(0\ 1\ \ldots\ 7)^7(0\ 1\ \ldots\ 6)^2\ldots(0\ 1\ 2)^1$, а именно 9703156248.

13. Первое утверждение истинно для всех $n \ge 2$. Но когда 2 пересекает 1, а именно когда c_2 изменяется с 0 на 1, то имеем: $c_3 = 2$, $c_4 = 3$, $c_5 = \cdots = c_n = 0$, а следующей перестановкой для $n \ge 5$ является $432156\ldots n$ [см. *Time Travel* (1988), p. 74].

14. Истинно в начале этапов P4–P6, поскольку точно $j - 1 - c_j + s$ элементов лежат слева от x_j, а именно $j - 1 - c_j$ из $\{x_1, \ldots, x_{j-1}\}$ и s из $\{x_{j+1}, \ldots, x_n\}$. (В некотором смысле эта формула является сутью алгоритма P.)

15. Если $\left[\begin{smallmatrix}b_{n-1},&\ldots,&b_0\\1,&\ldots,&n\end{smallmatrix}\right]$ соответствует отраженному коду Грэя $\left[\begin{smallmatrix}c_1,&\ldots,&c_n\\1,&\ldots,&n\end{smallmatrix}\right]$, то переходим к этапу P6 тогда и только тогда, когда $b_k = k-1$ для $j \le k \le n$ и B_{n-j+1} четно, согласно 7.2.1.1–(50). Но $b_{n-k} = k-1$ для $j \le k \le n$ подразумевает, что B_{n-k} нечетно для $j < k \le m$. Следовательно, $s = [c_{j+1} = j] + [c_{j+2} = j+1] = [o_{j+1} < 0] + [o_{j+2} < 0]$ на этапе P5 [см. *Math. Comp.* **17** (1963), p.282–285].

16. **P1′.** [Инициализировать.] Установить $c_j \leftarrow j$ и $o_j \leftarrow -1$ для $1 \le j < n$; также установить $z \leftarrow a_n$.

P2′. [Посетить.] Посетить $a_1 \ldots a_n$. Затем перейти к P3.5′, если $a_1 = z$.

P3′. [“Охотиться” вниз.] Для $j \leftarrow n-1, n-2, \ldots, 1$ (в этом порядке) установить $a_{j+1} \leftarrow a_j$, $a_j \leftarrow z$ и посетить $a_1 \ldots a_n$. Затем установить $j \leftarrow n-1$, $s \leftarrow 1$ и перейти к P4′.

P3.5′. [“Охотиться” вверх.] Для $j \leftarrow 1, 2, \ldots, n-1$ (в этом порядке) установить $a_j \leftarrow a_{j+1}$, $a_{j+1} \leftarrow z$ и посетить $a_1 \ldots a_n$. Затем установить $j \leftarrow n-1$, $s \leftarrow 0$.

P4′. [Готовы к изменению?] Установить $q \leftarrow c_j + o_j$. Если $q = 0$, то перейти к P6′; если $q > j$, перейти к P7′.

P5′. [Изменить.] Заменить $a_{c_j+s} \leftrightarrow a_{q+s}$. Затем установить $c_j \leftarrow q$ и вернуться к P2′.

P6′. [Увеличить s.] Завершить, если $j = 1$; в противном случае установить $s \leftarrow s+1$.

P7′. [Переключить направленне.] Установить $o_j \leftarrow -o_j$, $j \leftarrow j-1$ и перейти назад к P4′. ▮

17. В исходном состоянии $a_j \leftarrow a'_j \leftarrow j$ для $1 \le j \le n$. На этапе P5 следует установить $t \leftarrow j - c_j + s$, $u \leftarrow j - q + s$, $v \leftarrow a_u$, $a_t \leftarrow v$, $a'_v \leftarrow t$, $a_u \leftarrow j$, $a'_j \leftarrow u$, $c_j \leftarrow q$ (см. упр. 14).

С помощью требуемой и доступной обратной перестановки можно значительно упростить алгоритм, избегая использования переменной отступа s и позволяя контрольной таблице $c_1 \ldots c_n$ отсчитывать только убывания, как заметил Г. Эрлих (G. Ehrlich) [см. *JACM* **20** (1973), p.505–506].

Q1. [Инициализировать.] Установить $a_j \leftarrow a'_j \leftarrow j$, $c_j \leftarrow j-1$ и $o_j \leftarrow -1$ для $1 \le j \le n$. Также установить $c_0 = -1$.

Q2. [Посетить.] Посетить перестановку $a_1 \ldots a_n$ и обратную ей перестановку $a'_1 \ldots a'_n$.

Q3. [Найти k.] Установить $k \leftarrow n$. Затем, если $c_k = 0$, установить $c_k \leftarrow k-1$, $o_k \leftarrow -o_k$, $k \leftarrow k-1$ и повторять, пока не будет выполняться условие $c_k \neq 0$. Завершить, если $k = 0$.

Q4. [Изменить.] Установить $c_k \leftarrow c_k - 1$, $j \leftarrow a'_k$, и $i = j + o_k$. Затем установить $t \leftarrow a_i$, $a_i \leftarrow k$, $a_j \leftarrow t$, $a'_t \leftarrow j$, $a'_k \leftarrow i$ и вернуться к Q2. ▮

18. Установить $a_n \leftarrow n$ и использовать $(n-1)!/2$ итераций алгоритма P для генерации всех таких перестановок $\{1, \ldots, n-1\}$, что 1 предшествует 2 [см. M. K. Roy, *CACM* **16** (1973), p.312–313, а также упр. 13].

19. Например, можно воспользоваться идеей алгоритма P с n-кортежами $c_1 \ldots c_n$, изменяющимися, как в алгоритме 7.2.1.1H, по отношению к основаниям $(1, 2, \ldots, n)$. Этот алгоритм корректно поддерживает направления, хотя имеет другую нумерацию индексов. Отступ s, необходимый для алгоритма P, можно вычислить, как в ответе к упр. 15; обратную перестановку можно выполнить, как в упр. 17 [см. работу Г. Эрлиха в *CACM* **16** (1973), p.690–691]. Другие алгоритмы, как метод Хипа, также могут быть реализованы без циклов.

(*Замечание.* В большинстве приложений генерации перестановок важно сократить до минимума *общее* время выполнения, а не максимальное время между последовательными посещениями. С этой точки зрения отсутствие циклов обычно не требуется, за исключением счета на параллельном компьютере. Хотя остается интеллектуальное удовлетворение от осознания того факта, что существует бесцикловый алгоритм, независимо от его практической необходимости.)

20. Например, для $n = 3$ начинаем с 123, 132, 312, $\bar{3}12$, $1\bar{3}2$, $12\bar{3}$, $21\bar{3}$, ..., 213, $\bar{2}13$, ... Если дельта-последовательность для n имеет вид $(\delta_1\delta_2 \ldots \delta_{2^n-1})$, то соответствующая последовательность для $n+1$ имеет вид $(\Delta_n\delta_1\Delta_n\delta_2 \ldots \Delta_n\delta_{2^n-1})$, где Δ_n является последовательностью $2n+1$ операций $n\ n{-}1\ \ldots\ 1\ -\ 1\ \ldots\ n{-}1\ n$. Здесь $\delta_k = j$ означает $a_j \leftrightarrow a_{j+1}$ и $\delta_k = -$ означает $a_1 \leftarrow -a_1$.

(Перестановки со знаком предстают в другом виде в упр. 5.1.4–43 и 44. Множество всех перестановок со знаками называется группой октаэдра.)

21. Очевидно, что $\mathtt{M} = 1$, следовательно, O должно быть равно 0 и S должно быть равно $b-1$. Тогда $\mathtt{N} = \mathtt{E}+1$, $\mathtt{R} = b-2$, и $\mathtt{D}+\mathtt{E} = b+\mathtt{Y}$. Это дает точно $\max(0, b-7-k)$ вариантов для E, когда $\mathtt{Y} = k \ge 2$, следовательно, общее количество решений равно: $\sum_{k=2}^{b-7}(b-7-k) = \binom{b-8}{2}$ для $b \ge 8$ [см. *Math. Mag.* **45** (1972), p.48–49]. Кстати, Д. Эпштейн (D. Eppstein) доказал, что задача решения буквометики с заданным основанием является NP-полной [см. *SIGACT News* **18**, 3 (1987), p.38–40].

22. $(\mathtt{XY})_b + (\mathtt{XX})_b = (\mathtt{XYX})_b$ разрешима только для $b = 2$.

23. Почти истинно, поскольку количество решений четно, *если только не выполняется условие* $[j \in F] \neq [k \in F]$. (Рассмотрим третичную буквометику $\mathtt{X} + (\mathtt{XX})_3 + (\mathtt{YY})_3 + (\mathtt{XZ})_3 = (\mathtt{XYX})_3$.)

24. (a) $9283+7+473+1062 = 10825$. (b) $698392+3192 = 701584$. (c) $63952+69275 = 133227$. (d) $653924 + 653924 = 1307848$. (e) $5718 + 3 + 98741 = 104462$. (f) $127503 + 502351 + 3947539 + 46578 = 4623971$. (g) $67432 + 704 + 8046 + 97364 = 173546$. (h) $59 + 577404251698 + 69342491650 + 49869442698 + 1504 + 40614 + 82591 + 344 + 41 + 741425 = 5216367650 + 691400684974$. (Все решения единственны. [См. в отношении (b)–(g): *J. Recreational Math.* **10** (1977), p.115; **5** (1972), p.296; **10** (1977), p.41; **10** (1978), p.274; **12** (1979), p.133–134; **9** (1977), p.207.])

(i) В этом случае существует $\frac{8}{10}10! = 2903040$ решений, поскольку годится *каждая* перестановка $\{0, 1, \ldots, 9\}$, за исключением тех, в которых буква H или N присваивается 0. (Тщательно продуманная программа решения общих буквометик сложения должна сокращать объем выводимых данных в таких случаях.)

25. Допустим, что $s_1 \le \cdots \le s_{10}$. Пусть i является наименьшим индексом $\notin F$, установим $a_i \leftarrow 0$, а потом расположим остальные элементы a_j в порядке возрастания j. Доказательство, аналогичное доказательству теоремы 6.1S, показывает, что эта процедура максимизирует $a \cdot s$. Аналогичная процедура достигает минимума, поскольку $\min(a \cdot s) = -\max(a \cdot (-s))$.

26. $400739 + 63930 - 2379 - 1252630 + 53430 - 1390 + 738300$.

27. Смогут ли читатели улучшить следующие примеры: BLOOD + SWEAT + TEARS = LATER; EARTH + WATER + WRATH = HELLO + WORLD; AWAIT + ROBOT + ERROR = SOBER + WORDS; CHILD + THEME + PEACE + ETHIC = IDEAL + ALPHA + METIC. (Это упражнение возникло после знакомства с буквометикой WHERE + SEDGE + GRASS + GROWS = MARSH [см. A. W. Jr. Johnson, *J. Recr. Math.* **15** (1982), p.51], которая была бы изумительно чистой, за исключением того, что D и 0 имеют одинаковые сигнатуры.)

28. (a) $11 = 3+3+2+2+1$, $20 = 11+3+3+3$, $20 = 11+3+3+2+1$, $20 = 11+3+3+1+1+1$, $20 = 8+8+2+1+1$, $20 = 7+7+6$, $20 = 7+7+2+2+2$, $20 = 7+7+2+1+1+1+1$, $20 = 7+5+5+2+1$, $20 = 7+5+2+2+2+1+1$, $20 = 7+5+2+2+1+1+1+1$, $20 = 7+3+3+2+2+1+1+1$, $20 = 7+3+3+1+1+1+1+1+1+1$, $20 = 5+3+3+3+3+3$. (Эти четырнадцать решений были впервые получены Роем Чайлдсом (Roy Childs) в 1999 году. Следующие дважды разбиваемые значения n: 30 (в 20 вариантах), 40 (в 94 вариантах), 41 (в 67), 42 (в 57), 50 (в 190 вариантах, включая $50 = 2 + 2 + \cdots + 2$) и т.д.)

(b) $51 = 20 + 15 + 14 + 2$, $51 = 15 + 14 + 10 + 9 + 3$, $61 = 19 + 16 + 11 + 9 + 6$, $65 = 17 + 16 + 15 + 9 + 7 + 1$, $66 = 20 + 19 + 16 + 6 + 5$, $69 = 18 + 17 + 16 + 10 + 8$, $70 = 30 + 20 + 10 + 7 + 3$, $70 = 20 + 16 + 12 + 9 + 7 + 6$, $70 = 20 + 15 + 12 + 11 + 7 + 5$, $80 = 50 + 20 + 9 + 1$, $90 = 50 + 12 + 11 + 9 + 5 + 2 + 1$, $91 = 45 + 19 + 11 + 10 + 5 + 1$. (Два варианта с числом 51 получены Стивеном Кааном (Steven Kahan); [см. *Have Some Sums To Solve* (Farmingdale, New York: Baywood, 1978), p.36–37,84,112]. Изумительные примеры с 17 различными слагаемыми на итальянском языке и 58 различными слагаемыми в виде чисел с римскими цифрами получены Джулио Чезаре (Giulio Cesare) [см. *J. Recr. Math.* **30** (1999), p.63].

Замечание. Прекрасный пример THREE = TWO + ONE + ZERO [см. [Richard L. Breisch, *Recreational Math. Magazine* **12** (December 1962), p.24], к сожалению, не отвечает нашим соглашениям. Общее число дважды истинных разбиений на различные части, вероятно, конечно на английском языке, хотя номенклатура для произвольно больших целых чисел не является стандартной. Существует ли пример больше следующего (предложенного Гонсалесом-Моррисом (G. González-Morris)) примера

NINETYNINENONILLIONNINETYNINESEXTILLIONSIXTYONE

= NINETYNINENONILLIONNINETYNINESEXTILLIONNINETEEN + SIXTEEN + ELEVEN + NINE + SIX?

29. $10 + 7 + 1 = 9 + 6 + 3$, $11 + 10 = 8 + 7 + 6$, $12 + 7 + 6 + 5 = 11 + 10 + 9$, $\ldots$, $19 + 10 + 3 = 14 + 13 + 4 + 1$ (всего 31 пример).

30. (a) $567^2 = 321489$, $807^2 = 651249$ или $854^2 = 729316$. (b) $958^2 = 917764$. (c) $96 \times 7^2 = 4704$. (d) $51304/61904 = 7260/8760$. (e) $328509^2 = 4761^3$ [см. *Strand* **78** (1929), 91, p.208; *J. Recr. Math* **3** (1970), p.43; **13** (1981), p.212; **27** (1995), p.137; **31** (2003), p.133]. Решения (b), (c), (d) и (e) являются единственными. С помощью подхода справа налево на основе алгоритма X эти ответы можно найти за (14, 13, 11, 3423, 42) киломемс соответственно. Ноб Йошигахара также заметил, что NORTH/SOUTH = WEST/EAST имеет единственное решение: $67104/27504 = 9320/3820$.

31. $5/34 + 7/68 + 9/12$(!). Единственность можно проверить с помощью алгоритма X, используя дополнительное условие A < D < G приблизительно за 265 Kμ [см. *Quark Visual Science Magazine*, No. 136 (Tokyo: Kodansha, October 1993)]. Забавно, что очень похожая головоломка также имеет единственное решение: $1/(3 \times 6) + 5/(8 \times 9) + 7/(2 \times 4) = 1$ [см. работу С. Морриса в *Omni* **17**, 4 (January 1995), p.97].

32. Существует 11 способов, из которых наиболее удивительным является $3 + 69258/714$ [см. *The Weekly Dispatch* (9 и 23 June 1901); *Amusements* в *Mathematics* (1917), p.158–159].

33. (a) 1, 2, 3, 4, 15, 18, 118, 146. (b) 6, 9, 16, 20, 27, 126, 127, 129, 136, 145 [см. *The Weekly Dispatch* (11 and 30 November, 1902); *Amusements* в *Math.* (1917), p.159].

В этом случае одна приемлемая стратегия заключается в поиске всех вариаций, в которых $a_k \ldots a_{l-1}/a_l \ldots a_9$ является целым числом, и записи решений для всех перестановок $a_1 \ldots a_{k-1}$. Существует точно 164959 целых чисел с единственным решением, причем самое большое равно 9876533. Существуют решения для всех годов XXI века, за исключением 2091. Наибольшее количество решений (125) найдено для $n = 6443$. Самый длинный диапазон нредставимых n найден для $5109 < n < 7060$. Дьюдени смог получить правильные ответы вручную для малых n, "отбрасывая девятки".

34. (a) $x = 10^5$, $7378 + 155 + 92467 = 7178 + 355 + 92467 = 1016 + 733 + 98251 = 100000$. (b) $x = 4^7$, $3036 + 455 + 12893 = 16384$ является единственным решением. Скорейший способ решения этой задачи, вероятно, заключается в том, чтобы начать со списка 2529 простых чисел, которые состоят из пяти различных цифр (а именно 10243, 10247, ..., 98731), и переставлять пять оставшихся цифр.

Кстати, неограниченная буквометика EVEN + ODD = PRIME имеет десять решеиий; причем ODD и PRIME являются простыми только в одном из них [см. статью М. Арисава в *J. Recr. Math.* **8** (1975), p.153].

35. Вообще, если $s_k = |S_k|$ для $1 \le k < n$, то существует $s_1 \ldots s_{k-1}$ способов выбора каждого нетождественного элемента из S_k. Следовательно, ответ имеет вид $\prod_{k=1}^{n-1}(\prod_{j=1}^{k-1} s_j^{s_k-1})$, т.е. в данном случае $2^2 \cdot 6^3 \cdot 24^{15} = 436196692474023836123136$.

(Но если перенумеровать вершины, то значения s_k могут измениться. Например, если вершины (0, 3, 5) of (12) меняются с (e, d, c), то имеем $s_{14} = 1$, $s_{13} = 6$, $s_{12} = 4$, $s_{11} = 1$ и $4^5 \cdot 24^{15}$ таблиц Симса.)

36. Поскольку каждый из элементов $\{0, 3, 5, 6, 9, \mathtt{a}, \mathtt{c}, \mathtt{f}\}$ располагается на трех линиях, но каждый из остальных элементов располагается только на двух линиях, то ясно, что можно допустить $S_{\mathtt{f}} = \{(), \sigma, \sigma^2, \sigma^3, \alpha, \alpha\sigma, \alpha\sigma^2, \alpha\sigma^3\}$, где σ = (03fc)(17e4)(2bd4)(56a9) является поворотом на 90°, а α = (05)(14)(27)(36)(8d)(9c)(af)(be) является кручением изнутри наружу. Кроме того, $S_{\mathtt{e}} = \{(), \beta, \gamma, \beta\gamma\}$, где β = (14)(28)(3c)(69)(be) является транспозицией и γ = (12)(48)(5a)(69)(7b)(de) является другим кручением; $S_{\mathtt{d}} = \cdots = S_1 = \{()\}$. (Всего $4^7 - 1$ альтернативных ответов.)

37. Множество S_k можно выбрать $k!^{k-1}$ способами (см. упр. 35), а его нетождественные элементы можно присвоить $\sigma(k, 1)$, ..., $\sigma(k, k)$, причем $k!$ способами. Поэтому ответ имеет

вид $A_n = \prod_{k=1}^{n-1} k!^k = n!^{\binom{n}{2}} / \prod_{k=1}^{n} k^{\binom{k}{2}}$. Например, $A_{10} \approx 6{,}256 \times 10^{148}$. Имеем:

$$\sum_{k=1}^{n-1} \binom{k}{2} \ln k \approx \frac{1}{2} \int_1^n x(x-1) \ln x \, dx + O(n^2 \log n) = \frac{1}{6} n^3 \ln n + O(n^3)$$

по формуле суммирования Эйлера. Таким образом, $\ln A_n = \frac{1}{3} n^3 \ln n + O(n^3)$.

38. Вероятность того, что $\phi(k)$ потребуется на этапе G4, равна $1/k! - 1/(k+1)!$ для $1 \le k < n$, но вероятность непопадания на этап G4 равна $1/n!$. Поскольку $\phi(k)$ выполняет $\lceil k/2 \rceil$ транспозиций, то среднее равно: $\sum_{k=1}^{n-1} (1/k! - 1/(k+1)!) \lceil k/2 \rceil = \sum_{k=1}^{n-1} (\lceil k/2 \rceil - \lceil (k-1)/2 \rceil)/k! - \lceil (n-1)/2 \rceil / n! = \sum_{k \text{ odd}} 1/k! + O(1/(n-1)!)$.

39. (a) 0123, 1023, 2013, 0213, 1203, 2103, 3012, 0312, 1302, 3102, 0132, 1032, 2301, 3201, 0231, 2031, 3021, 0321, 1230, 2130, 3120, 1320, 2310, 3210; (b) 0123, 1023, 2013, 0213, 1203, 2103, 3102, 1302, 0312, 3012, 1032, 0132, 0231, 2031, 3021, 0321, 2301, 3201, 3210, 2310, 1320, 3120, 2130, 1230.

40. По индукции находим $\sigma(1,1) = (0\ 1)$, $\sigma(2,2) = (0\ 1\ 2)$,

$$\sigma(k,k) = \begin{cases} (0\ k)(k{-}1\ k{-}2\ \ldots\ 1), & \text{если } k \ge 3 \text{ нечетно;} \\ (0\ k{-}1\ k{-}2\ 1\ \ldots\ k{-}3\ k), & \text{если } k \ge 4 \text{ четно,} \end{cases}$$

а также $\omega(k) = (0\ k)$, если k четно, $\omega(k) = (0\ k{-}2\ \ldots\ 1\ k{-}1\ k)$, если $k \ge 3$ нечетно. Таким образом, если $k \ge 3$ нечетно, $\sigma(k,1) = (k\ k{-}1\ 0)$ и $\sigma(k,j)$ принимает $k \mapsto j-1$ для $1 < j < k$; если $k \ge 4$ четно, $\sigma(k,j) = (0\ k\ k{-}3\ \ldots\ 1\ k{-}2\ k{-}1)^j$ для $1 \le j \le k$.

Замечание. Первая схема, которая использует алгоритм G для генерации всех перестановок на основе одних транспозиций, была предложена Марком Уэллсом (Mark Wells) [см. *Math. Comp.* **15** (1961), p.192–195], но она была значительно более сложной. В. Липскимл. (W. Jr. Lipski) изучал такие схемы в более общем случае и обнаружил несколько других методов [см. *Computing* **23** (1979), p.357–365].

41. Предположим, что $r < n$. Алгоритм G будет генерировать r-вариации для произвольной таблицы Симса, если просто заменить '$k \leftarrow 1$' на '$k \leftarrow n - r$' на этапе G3, при условии, что $\omega(k)$ переопределяется и равно $\sigma(n-r, n-r) \ldots \sigma(k,k)$, вместо использования (16).

Если $n - r$ нечетно, то метод (27) еще применим, хотя формулы в ответе к упр. 40 нужно исправить для $k < n - r + 2$. Новые формулы имеют вид $\sigma(k,j) = (k\ j{-}1\ \ldots\ 1\ 0)$ и $\omega(k) = (k\ \ldots\ 1\ 0)$ для $k = n - r$; $\sigma(k,j) = (k\ \ldots\ 1\ 0)^j$ для $k = n - r + 1$.

Если $n - r$ четно, то можно использовать (27) с четными и нечетными обратными, если $r \le 3$. Но для $r \ge 4$ потребуется более сложная схема, поскольку такая фиксированная транспозиция, как $(k\ 0)$, может использоваться для нечетных k, только если $\omega(k-1)$ является k-циклом, т.е. $\omega(k-1)$ должна быть четной перестановкой; но $\omega(k)$ нечетно для $k \ge n - r + 2$.

Следующая схема работает, если $n - r$ четно: пусть $\tau(k,j)\omega(k-1)^- = (k\ k{-}j)$ для $1 \le j \le k = n - r$, а для $k > n - r$ используем (27). Затем, если $k = n - r + 1$, имеем $\omega(k-1) = (0\ 1\ \ldots\ k{-}1)$, следовательно, $\sigma(k,j)$ принимает $k \mapsto (2j-1) \bmod k$ для $1 \le j \le k$, и $\sigma(k,k) = (k\ k{-}1\ k{-}3\ \ldots\ 0\ k{-}2\ \ldots\ 1)$, $\omega(k) = (k\ \ldots\ 1\ 0)$, $\sigma(k+1,j) = (k{+}1\ \ldots\ 0)^j$.

42. Если $\sigma(k,j) = (k\ j{-}1)$, то имеем $\tau(k,1) = (k\ 0)$ и $\tau(k,j) = (k\ j{-}1)(k\ j{-}2) = (k\ j{-}1\ j{-}2)$ для $2 \le j \le k$.

43. Конечно, $\omega(1) = \sigma(1,1) = \tau(1,1) = (0\ 1)$. Следующая конструкция образует $\omega(k) = (k{-}2\ k{-}1\ k)$ для всех $k \ge 2$: пусть $\alpha(k,j) = \tau(k,j)\omega(k-1)^-$, где $\alpha(2,1) = (2\ 0)$, $\alpha(2,2) = (2\ 0\ 1)$, $\alpha(3,1) = \alpha(3,3) = (3\ 1)$, $\alpha(3,2) = (3\ 1\ 0)$; это дает $\sigma(2,2) = (0\ 2)$, $\sigma(3,3) = (0\ 3\ 1)$. Затем для $k \ge 4$ допустим, что $\alpha(k,1) = (k\ k{-}4)$, $\alpha(k,j) = (k\ k{-}3{-}j\ k{-}2{-}j)$ для $1 < j <$

$k-2$ и

$$\begin{array}{rcccccc} & k \bmod 3 = 0 & & k \bmod 3 = 1 & & k \bmod 3 = 2 \\ \alpha(k,k{-}2) = & (k\ k{-}2\ 0), & \text{или} & (k\ k{-}3\ 0), & \text{или} & (k\ k{-}1\ 0); \\ \alpha(k,k{-}1) = & (k\ k{-}2\ k{-}3), & \text{или} & (k\ k{-}3), & \text{или} & (k\ k{-}1\ k{-}3); \\ \alpha(k,k) = & (k\ k{-}2), & \text{или} & (k\ k{-}3\ k{-}2), & \text{или} & (k\ k{-}2), \end{array}.$$

Это дает $\sigma(k,k) = (k{-}3\ k\ k{-}2)$, что и требовалось.

44. Нет, поскольку $\tau(k,j)$ является $(k+1)$-циклом, а не транспозицией (см. (19) и (24)).

45. (a) 202280070, поскольку $u_k = \max(\{0,1,\ldots,a_k-1\} \setminus \{a_1,\ldots,a_{k-1}\})$. (На самом деле u_n никогда не устанавливается алгоритмом, но можно предположить, что он равен нулю.) (b) 425368917.

46. Истинно (предполагая, что $u_n = 0$). Если $u_k > u_{k+1}$ или $a_k > a_{k+1}$, то получим $a_k > u_k \geq a_{k+1} > u_{k+1}$.

47. Этапы $(\mathrm{X1}, \mathrm{X2}, \ldots, \mathrm{X6})$ выполняются соответственно $(1, A, B, A-1, B-N_n, A)$ раз, где $A = N_0 + \cdots + N_{n-1}$ и $B = nN_0 + (n-1)N_1 + \cdots + 1N_{n-1}$.

48. Этапы $(\mathrm{X2}, \mathrm{X3}, \mathrm{X4}, \mathrm{X5}, \mathrm{X6})$ выполняются соответственно $A_n + (1, n!, 0, 0, 1)$ раз, где $A_n = \sum_{k=1}^{n-1} n^k = n! \sum_{k=1}^{n-1} 1/k! \approx n!\,(e-1)$. Предполагая, что затраты на них равны соответственно $(1,1,3,1,3)$ мемсов, для операций с a_j, l_j или u_j общие затраты равны около $9e - 8 \approx 16{,}46$ мемса на перестановку.

Алгоритм L затрачивает приблизительно $(e, 2 + e/2, 2e + 2e^{-1} - 4)$ мемсов на перестановку на этапах $(\mathrm{L2}, \mathrm{L3}, \mathrm{L4})$, а общие затраты равны $3{,}5e + 2e^{-1} - 2 \approx 8{,}25$ (см. упр. 5).

Алгоритм X можно оптимизировать в этом случае для значений k, близких к n. То же самое можно сделать и с алгоритмом L, как показано в упр. 1.

49. Упорядочьте сигнатуры так, чтобы $|s_0| \geq \cdots \geq |s_9|$. Приготовьте таблицы $w_0 \ldots w_9$, $x_0 \ldots x_9$, $y_0 \ldots y_9$ так, чтобы для сигнатур $\{s_k, \ldots, s_9\}$ выполнялись условия $w_{x_k} \leq \cdots \leq w_{y_k}$. Например, для SEND + MORE = MONEY имеем $(s_0, \ldots, s_9) = (-9000, 1000, -900, 91, -90, 10, 1, -1, 0, 0)$ для соответствующих букв (M, S, O, E, N, R, D, Y, A, B); $(w_0, \ldots, w_9) = (-9000, -900, -90, -1, 0, 0, 1, 10, 91, 1000)$ и $x_0 \ldots x_9 = 0112233344$, $y_0 \ldots y_9 = 9988776554$. Еще одна таблица $f_0 \ldots f_9$ содержит $f_j = 1$, если цифра, соответствующая w_j, не может быть нулем, и в этом случае $f_0 \ldots f_9 = 1000000001$. Эти таблицы упрощают вычисление самого крупного и самого малого значений

$$s_k a_k + \cdots + s_9 a_9$$

для всех вариантов $a_k \ldots a_9$ из остальных цифр с помощью метода из упр. 25, поскольку связи l_j информируют нас об этих цифрах в порядке возрастания.

Этот метод требует очень затратных вычислений в каждом узле дерева поиска, но зато он часто позволяет поддерживать малым размер дерева. Например, он разрешает первые восемь буквометик из упр. 24 всего за 7, 13, 7, 9, 5, 343, 44 и 89 киломемс. В случаях (a), (b), (e) и (h) наблюдается существенное улучшение, а в случае (f) — существенное ухудшение. Еще один неудачный пример связан с буквометикой 'CHILD' из упр. 27, где метод "слева направо" требует 2947 киломемс по сравнению с 588 киломемсами для метода "справа налево". Однако метод "слева направо" показывает более эффективную работу для BLOOD + SWEAT + TEARS (73 по сравнению с 360) и HELLO + WORLD (340 по сравнению с 410).

50. Если α входит в группу перестаиовок, то в иее входят все ее степени α^2, α^3, ..., включая $\alpha^{m-1} = \alpha^-$, где m является порядком α (наименьшее общее кратное длин ее циклов). А (32) эквивалентно $\alpha^- = \sigma_1\sigma_2 \ldots \sigma_{n-1}$.

51. Ложно. Например, обе $\sigma(k,i)^-$ и $\sigma(k,j)^-$ могут принимать значения $k \mapsto 0$.

52. $\tau(k,j) = (k{-}j\ k{-}j{+}1)$ является смежной заменой, а

$$\omega(k) = (n{-}1\ \ldots\ 0)(n{-}2\ \ldots\ 0)\ldots(k\ \ldots\ 0) = \phi(n-1)\phi(k-1)$$

является k-переворотом, за которым следует n-переворот. Перестановка, соответствующая контрольной таблице $c_0 \ldots c_{n-1}$ в алгоритме H, содержит c_j элементов справа от j, которые меньше j для $0 \le j < n$. Поэтому она точно такая же, как и перестановка, соответствующая $c_1 \ldots c_n$ в алгоритме P, за исключением того, что индексы сдвинуты на 1.

Единственным существенным различием между алгоритмом P и этой версией алгоритма H является то, что алгоритм P использует отраженный код Грэя для перебора всех вариантов своей контрольной таблицы, тогда как алгоритм H перебирает числа со смешанным основанием в (лексикографическом) порядке возрастания.

Действительно, код Грэя можно использовать вместе с любой таблицей Симса, изменяя алгоритм G или H. Тогда все переходы на основе $\tau(k,j)$ или $\tau(k,j)^-$, а также перестановки $\omega(k)$ окажутся неуместными.

53. В этом разделе в доказательстве того факта, что $n! - 1$ транспозиций нельзя достичь для $n = 4$, также показано, что можно свести эту задачу от n к $n-2$ за счет одной транспозиции $(n{-}1\ n{-}2)$, т.е. '$(3\ c)$' в обозначениях данного доказательства.

Таким образом, можно генернровать все перестановки с помощью следующих преобразований на этапе H4: если $k = n{-}1$ или $k = n{-}2$, то транспонировать $a_{j \bmod n} \leftrightarrow a_{(j-1) \bmod n}$, где $j = c_{n-1} - 1$. Если $k = n-3$ или $k = n-4$, то транспонировать $a_{n-1} \leftrightarrow a_{n-2}$ и $a_{j \bmod (n-2)} \leftrightarrow a_{(j-1) \bmod (n-2)}$, где $j = c_{n-3} - 1$. И вообще, если $k = n - 2t - 1$ или $k = n - 2t - 2$, то транспонировать $a_{n-2i+1} \leftrightarrow a_{n-2i}$ для $1 \le i \le t$ и также $a_{j \bmod (n-2t)} \leftrightarrow a_{(j-1) \bmod (n-2t)}$, где $j = c_{n-2t-1} - 1$ [см. *CACM* **19** (1976), p.68–72].

Соответствующая таблица Симса для перестановок может быть записана следующим образом, хотя они не присутствуют явно в самом алгоритме:

$$\sigma(k,j)^- = \begin{cases} (0\ 1\ \ldots\ j{-}1\ k), & \text{если } n-k \text{ нечетно;} \\ (0\ 1\ \ldots\ k)^j, & \text{если } n-k \text{ четно.} \end{cases}$$

Значение $a_{j \bmod (n-2t)}$ будет равно $n-2t-1$ после замены. Для повышения эффективности можно также использовать тот факт, что значение k обычно равно $n-1$. Общее число транспоэиций равно $\sum_{t=0}^{\lfloor n/2 \rfloor}(n-2t)! - \lfloor n/2 \rfloor - 1$.

54. Да; это преобразование может быть любым k-циклом по поэициям $\{1, \ldots, k\}$.

55. (a) Поскольку $\rho_l(m) = \rho_l(m \bmod n!)$ для $n > \rho_l(m)$, то имеем $\rho_l(n! + m) = \rho_l(m)$ для $0 < m < n \cdot n! = (n+1)! - n!$. Следовательно, $\beta_{n!+m} = \sigma_{\rho_l(n!+m)} \cdots \sigma_{\rho_l(n!+1)} \beta_{n!} = \sigma_{\rho_l(m)} \cdots \sigma_{\rho_l(1)} \beta_{n!} = \beta_m \beta_{n!}$ для $0 \le m < n \cdot n!$, и имеем, в частности,

$$\beta_{(n+1)!} = \sigma_{n+1} \beta_{(n+1)!-1} = \sigma_{n+1} \beta_{n!-1} \beta_{n!}^n = \sigma_{n+1} \sigma_n^- \beta_{n!}^{n+1}.$$

Аналогично, $\alpha_{n!+m} = \beta_{n!}^- \alpha_m \beta_{n!} \alpha_{n!}$ для $0 \le m < n \cdot n!$.

Поскольку $\beta_{n!}$ коммутирует с τ_n и τ_{n+1}, то находим $\alpha_{n!} = \tau_n \alpha_{n!-1}$, и

$$\begin{aligned} \alpha_{(n+1)!} = \tau_{n+1} \alpha_{(n+1)!-1} &= \tau_{n+1} \beta_{n!}^- \alpha_{(n+1)!-1-n} \beta_{n!} \alpha_{n!} = \cdots \\ &= \tau_{n+1} \beta_{n!}^{-n} \alpha_{n!-1} (\beta_{n!} \alpha_{n!})^n \\ &= \beta_{n!}^{-n-1} \tau_{n+1} \tau_n^- (\beta_{n!} \alpha_{n!})^{n+1} \\ &= \beta_{(n+1)!}^- \sigma_{n+1} \sigma_n^- \tau_{n+1} \tau_n^- (\beta_{n!} \alpha_{n!})^{n+1}. \end{aligned}$$

(b) В этом случае $\sigma_{n+1}\sigma_n^- = (n\ n{-}1\ \ldots\ 1)$ и $\tau_{n+1}\tau_n^- = (n{+}1\ n\ 0)$, и имеем $\beta_{(n+1)!}\alpha_{(n+1)!} = (n{+}1\ n\ \ldots\ 0)$ по индукции. Следовательно, $\alpha_{jn!+m} = \beta_{n!}^{-j} \alpha_m (n\ \ldots\ 0)^j$ для $0 \le j \le n$ и $0 \le m < n!$. Все перестановки $\{0, \ldots, n\}$ достигаются, поскольку $\beta_{n!}^{-j}\alpha_m$ фиксирует n и $(n\ \ldots\ 0)^j$ принимает $n \mapsto n-j$.

56. Если установить $\sigma_k = (k{-}1\ k{-}2)(k{-}3\ k{-}4)\ldots$ в предыдущем упражнении, то по индукции можно найти, что $\beta_{n!}\alpha_{n!}$ является $(n+1)$-циклом $(0\ n\ n{-}1\ n{-}3\ \ldots$ (2 или 1) (1 или 2) $\ldots\ n{-}4\ n{-}2)$.

57. Поступая, как в ответе к упр. 5, получим $\sum_{k=2}^{n-1}[k\,\text{odd}]/k! \sim (\lfloor n/2\rfloor - 1)/n! = \sinh 1 - 1 - O(1/(n-1)!)$.

58. Истинно. По формулам из упр. 55 имеем $\alpha_{n!-1} = (0\ n)\beta_{n!}^{-}(n\ \ldots\ 0)$ и $0 \mapsto n-1$, поскольку $\beta_{n!}$ фиксирует n. (Следовательно, алгоритм E определит гамильтонов цикл на графе из упр. 66 тогда и только тогда, когда $\beta_{n!} = (n{-}1\ \ldots\ 2\ 1)$, и это выполняется тогда и только тогда, когда длина каждого цикла $\beta_{(n-1)!}$ является делителем n. Последнее истинно для $n = 2, 3, 4, 6, 12, 20$ и 40, но не для других $n \le 250{,}000$.)

59. Граф Кэли с генераторами $(\alpha_1, \ldots, \alpha_k)$ в определении в данном разделе изоморфен графу Кэли с генераторами $(\alpha_1^{-}, \ldots, \alpha_k^{-})$ в альтернативном определении, поскольку $\pi \to \alpha_j\pi$ в первом определении тогда и только тогда, когда $\pi^{-} \to \pi^{-}\alpha_j^{-}$ в последнем определении.

60. Имеется 88 дельта-последовательностей, которые сводятся к четырем классам: $P = (32131231)^3$ (простые изменения, представленные 8 разными дельта-последовательностями); $Q = (32121232)^3$ (двойной вариант Грэя простых изменений с 8 представителями); $R = (121232321232)^2$ (двойной код Грэя с 24 представителями); $S = 2\alpha 3\alpha^R$, $\alpha = 12321312121$ (с 8 представителями). Классы P и Q являются циклическими сдвигами их дополнений; классы P, Q и S являются сдвигами их обращений; класс R является сдвинутым обращением своего дополнения [см. статью А. Л. Л. Сильвера (A. L. L Silver) *Math. Gazette* **48** (1964), p.1–16].

61. Существует соответственно $(26, 36, 20, 26, 28, 40, 40, 20, 26, 28, 28, 26)$ путей, которые заканчиваются следующим образом: $(1243, 1324, 1432, 2134, 2341, 2413, 3142, 3214, 3421, 4123, 4231, 4312)$.

62. Есть только два пути для $n = 3$, которые заканчиваются перестановками 132 и 213. Но для $n \ge 4$ существуют коды Грэя, ведущие от $12\ldots n$ к любой нечетной перестановке $a_1a_2\ldots a_n$. В упр. 61 это доказано для $n = 4$, а здесь можно доказать это по индукции для $n > 4$ следующим образом.

Пусть $A(j)$ является множеством всех таких перестановок, которые начинаются с j, и пусть $A(j,k)$ является множеством всех таких перестановок, которые начинаются с jk. Если $(\alpha_0, \alpha_1, \ldots, \alpha_n)$ — это такие нечетные перестановки, что $\alpha_j \in A(x_j, x_{j+1})$, тогда $(1\,2)\alpha_j$ является четной перестановкой в $A(x_{j+1}, x_j)$. Следовательно, если $x_1x_2\ldots x_n$ является перестановкой $\{1, 2, \ldots, n\}$, то существует по крайней мере один гамильтонов путь вида

$$(1\,2)\alpha_0 \text{ — } \cdots \text{ — } \alpha_1 \text{ ~~} (1\,2)\alpha_1 \text{ — } \cdots \text{ — } \alpha_2 \text{ — } \cdots \text{ — } (1\,2)\alpha_{n-1} \text{ — } \cdots \text{ — } \alpha_n;$$

а подчиненный путь от $(1\,2)\alpha_{j-1}$ к α_j включает все элементы $A(x_j)$.

Эта конструкция решает задачу по крайней мере $(n-2)!^n/2^{n-1}$ различными способами, если $a_1 \ne 1$, поскольку можно принять $\alpha_0 = 21\ldots n$ и $\alpha_n = a_1a_2\ldots a_n$. Есть $(n-2)!$ способов выбора $x_2\ldots x_{n-1}$ и $(n-2)!/2$ способов выбора каждой из $\alpha_1, \ldots, \alpha_{n-1}$.

Наконец, если $a_1 = 1$, возьмем любой путь $12\ldots n$ — $\cdots$ — $a_1a_2\ldots a_n$, который проходит все $A(1)$, и выберем произвольный этап α — α' с $\alpha \in A(1,j)$ и $\alpha' \in A(1,j')$ для некоторого $j \ne j'$. Заменим этот этап на

$$\alpha \text{ — } (1\,2)\alpha_1 \text{ — } \cdots \text{ — } \alpha_2 \text{ — } \cdots \text{ — } (1\,2)\alpha_{n-1} \text{ — } \cdots \text{ — } \alpha_n \text{ — } \alpha',$$

используя конструкцию, как в показанном выше гамильтоновом пути, но теперь с $\alpha_1 = \alpha$, $\alpha_n = (1\,2)\alpha'$, $x_1 = 1$, $x_2 = j$, $x_n = j'$, и $x_{n+1} = 1$. (В этом случае все перестановки α_1, $\ldots$, α_n могут быть четными.)

63. Оценки по методу Монте-Карло с использованием технологий из раздела 7.2.3 дают для общего количества классов эквивалентности приблизительное значение $1,2 \times 10^{21}$. Большая часть этих классов будет содержать 480 циклов Грэя.

64. Точно 2 005 200 дельта-последовательностей обладают свойством дважды кода Грэя. Они относятся к 4206 классам эквивалентности для циклического сдвига, обращения и/или дополнения. Девять классов, например, как код $2\alpha 2\alpha^R$, где

$$\alpha = 123432343212321212323212321212343432121234321234321212321,$$

являются сдвигами их обращения; 48 классов состоят из повторяющихся 60-циклов. Один их наиболее интересных классов последнего типа имеет вид $\alpha\alpha$, где

$$\alpha = \beta 2\beta 4\beta 4\beta 4\beta 4, \qquad \beta = 32121232123.$$

65. Такой путь существует для произвольного заданного $N \le n!$: пусть N-я перестановка имеет вид $\alpha = a_1 \ldots a_n$, а $j = a_1$. Пусть Π_k является множеством всех перестановок $\beta = b_1 \ldots b_n$, для которых $b_1 = k$ и $\beta \le \alpha$. По индукции для N существует путь Грэя P_1 для Π_j. Тогда можно создать пути Грэя P_k для $\Pi_j \cup \Pi_1 \cup \cdots \cup \Pi_{k-1}$ для $2 \le k \le j$, последовательно комбинируя P_{k-1} с циклом Грэя для Π_{k-1}. (См. конструкцию "поглощение" в ответе к упр. 62. Действительно, P_j будет *циклом* Грэя, когда N кратно 6.)

66. Определяя дельта-последовательность с помощью правила $\pi_{(k+1) \bmod n!} = (1\ \delta_k)\pi_k$, можно найти точно 36 таких последовательностей, которые являются циклическими сдвигами структуры типа $(xyzyzyxzyzyz)^2$. (Следующий случай, $n = 5$, вероятно, имеет около 10^{18} решений, которые неэквивалентны по отношению к циклическому сдвигу, обращению и перестановке координат, т.е. около 6×10^{21} разных дельта-последовательностей.) Кстати, Игорь Пак (Igor Pak) показал, что в общем случае граф Кэли, генерированный звездными транспозициями, является $(n-2)$-мерным тором.

67. Если предположить, что π эквивалентно $\pi(12345)$, то получим сокращенный граф с 24 вершинами, который имеет 40768 гамильтоновых циклов, 240 из которых приводят к дельта-последовательностям вида α^5, в которых α использует каждую транспозицию шесть раз (например, $\alpha = 3542325342345324543524 52$). Общее количество решений этой задачи приблизительно равно 10^{16}.

68. Если A не является связным, то и G не является связным. Если A является связным, то можно предположить, что он является свободным деревом. Более того, в этом случае можно доказать обобщение результата из упр. 62: для $n \ge 4$ существует гамильтонов путь в G от тождественной перестановки к произвольной нечетной перестановке. Можно предположить без утраты общности, что A содержит ребро $1 \text{---} 2$, где 1 является листком дерева, а далее доказательство аналогично доказательству из упр. 62.

[Эта элегантная конструкция предложена М. Тчуенте (M. Tchuente) [см. *Ars Combinatoria* **14** (1982), p.115–122]. Широкие обобщения рассмотрены Ф. Раски и К. Сэвидж [см. *SIAM J. Discrete Math.* **6** (1993), p.152–166]. См. также публикацию на русском языке [*Кибернетика* **11**, 3 (1975), p.17–25], или ее английский перевод [см. *Cybernetics* **11** (1975), p.362–366].

69. Согласно подсказке, модифицированный алгоритм будет вести себя, как показано ниже, при $n = 5$.

1234	1243	1423	4123	4132	1432	1342	1324	3124	3142	3412	4312
↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑
54321	24351	24153	54123	14523	14325	24315	24513	54213	14253	14352	54312
12345	15342	35142	32145	32541	52341	51342	31542	31245	35241	25341	21345
15342	12435	32415	35412←31452		51432	52431	32451←35421		31425	21435	25431
23451	53421	51423	21453→25413		23415	13425	15423→12453		52413	53412	13452
21543	51243	53241	23541	23145	25143	15243	13245	13542	53142	52143	12543
34512	34215	14235	14532	54132	34152	34251	54231	24531	24135	34125	34521
32154→35124		15324→12354		52314	32514←31524		51324	21354→25314		35214→31254	
45123←42153		42351←45321		41325	41523→42513		42315	45312←41352		41253←45213	
43215	43512←41532		41235	45231→43251		43152→45132		42135	42531←43521		43125
51234	21534→23514		53214	13254←15234		25134←23154		53124	13524→12534		52134
↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑

Здесь столбцы представляют множества перестановок, которые циклически поворачиваются и/или отражаются всеми $2n$ способами. Следовательно, каждый столбец содержит точно одну четочную перестановку (см. упр. 18). Алгоритм P можно применить для систематического перебора всех четочных перестановок, зная, что пара xy появляется перед yx в его столбце, тогда τ' вместо ρ' переводит нас вправо или влево. На этапе Z2 пропускается замена $a_1 \leftrightarrow a_2$, таким образом приводя к повтору перестановок $a_1 \ldots a_{n-1}$ в обратном направлении. (Здесь неявно используется тот факт, что $t[k] = t[n! - k]$ на выходе алгоритма T.)

Теперь, если заменить $1 \ldots n$ на $24 \ldots 31$ и $A_1 \ldots A_n$ на $A_1 A_n A_2 A_{n-1} \ldots$, получим немодифицированный алгоритм, результат выполнения которого показан на рис. 22,*b*.

Этот метод инспирирован (неконструктивной) теоремой Э. С. Рапопорта (E. S. Rapoport) [см. *Scripta Math.* **24** (1959), p.51–58]. Он иллюстрирует более общий факт, замеченный Карлой Сэвидж в 1989 году: граф Кэли для *произвольной* группы, генерированной тремя инволюциями ρ, σ, τ, имеет гамильтонов цикл, когда $\rho\tau = \tau\rho$ [вскоре будет опубликована посвященная этой теме работа И. Пака (I. Pak) и Р. Радойчича (R. Radoičić), "Hamiltonian paths in Cayley graphs" (Гамильтоновы пути на графах Кэли)].

70. Нет. Самый длинный цикл в этом диграфе имеет длину 358. Но есть пары непересекающихся 180-циклов, на основе которых можно вывести гамильтонов путь длины 720. Например, рассмотрим циклы $\alpha\sigma\beta\sigma$ и $\gamma\sigma\sigma$, где

$$\alpha = \tau\sigma^5\tau\sigma^5\tau\sigma^3\tau\sigma^2\tau\sigma^5\tau\sigma^3\tau\sigma^2\tau\sigma^5\tau\sigma^5\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^5\tau\sigma^5\tau\sigma^5\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^3\tau\sigma^2\tau\sigma^1\tau\sigma^1;$$

$$\beta = \sigma^3\tau\sigma^5\tau\sigma^2\tau\sigma^2\tau\sigma^5\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^5\tau\sigma^1\tau\sigma^3\tau\sigma^5\tau\sigma^5\tau\sigma^5\tau\sigma^3\tau\sigma^2\tau\sigma^1\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^3\tau\sigma^2\tau\sigma^4;$$

$$\gamma = \sigma\tau\sigma^5\tau\sigma^5\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^3\tau\sigma^2\tau\sigma^5\tau\sigma^2\tau\sigma^3\tau\sigma^5\tau\sigma^1\tau\sigma^5\tau\sigma^3\tau\sigma^2\tau\sigma^1\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^3\tau\sigma^2$$
$$\tau\sigma^5\tau\sigma^5\tau\sigma^5\tau\sigma^3\tau\sigma^5\tau\sigma^2\tau\sigma^5\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^5\tau\sigma^1\tau\sigma^3\tau\sigma^3\tau\sigma^5\tau\sigma^5\tau\sigma^1\tau\sigma^5\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^2.$$

Если начать с 134526 и далее продолжить $\alpha\sigma\beta\tau$, то получим 163452; если далее продолжить $\gamma\sigma\tau$, то получим 126345; если далее продолжить $\sigma\gamma\tau$, то получим 152634; если далее продолжить $\beta\sigma\alpha$, то в итоге получим 415263.

71. Брендан Мак-Кэй (Brendan McKay) и Фрэнк Раски (Frank Ruskey) нашли такие циклы с помощью компьютера для $n = 7$, 9 и 11, но какую-либо интересную структуру им обнаружить не удалось.

72. Любой гамильтонов путь включает $(n-1)!$ вершин, которые преобразуются $y \mapsto x$, причем за каждой из вершин (если она не последняя) следует вершина, которая преобразуется $x \mapsto x$. Одна вершина должна быть последней, а другие $(n-1)! + 1$ вершин преобразуются $x \mapsto x$.

73. (a) Сначала предположим, что β является тождественной перестановкой (). Тогда каждый цикл α, который содержит элемент A, лежит целиком внутри A. Следовательно, циклы σ получаются за счет пропуска всех циклов α, которые не содержат элементов A. Все остальные циклы имеют нечетную длину, поэтому σ является четной перестановкой.

Если β не является четной перестановкой, то применим этот аргумент к $\alpha' = \alpha\beta^-$, $\beta' = ()$ и $\sigma' = \sigma\beta^-$. Отсюда следует, что σ' является четной перестановкой. Таким образом, σ и β имеют одинаковый знак.

Аналогично σ и α имеют одинаковый знак, поскольку $\beta\alpha^- = (\alpha\beta^-)^-$ имеют одинаковый порядок, как $\alpha\beta^-$.

(b) Пусть X является множеством вершин графа Кэли в теореме R, а α является перестановкой X, которая переводит вершину π в $\alpha\pi$. Эта перестановка имеет g/a циклов длины a. Аналогично определим перестановку β. Тогда $\alpha\beta^-$ имеет g/c циклов длины c. Если c нечетно, то любой гамильтонов цикл в графе определяет цикл σ, который содержит все вершины и удовлетворяет гипотезе (a). Следовательно, α и β имеют нечетное число циклов, поскольку знак перестановки n элементов с r циклами равен $(-1)^{n-r}$ (см. упр. 5.2.2–2).

(Это доказательство, согласно которому X не может быть объединением любого нечетного числа циклов, получено Р. А. Ранкином (R. A. Rankin) [*Proc. Cambridge Phil. Soc.* **62** (1966), p.15–16].)

74. Представление $\beta^j\gamma^k$ единственно, если потребовать $0 \le j < g/c$ и $0 \le k < c$. Если имеем $\beta^j = \gamma^k$ для некоторого j с $0 < j < g/c$, то группа будет иметь не более jc элементов. Отсюда следует, что $\beta^{g/c} = \gamma^t$ для некоторого t.

Пусть σ является гамильтоновым циклом, как в ответе к предыдущему упражнению. Если $\pi\sigma = \pi\alpha$, то $\pi\gamma\sigma$ должно быть равно $\pi\gamma\alpha$, поскольку $\pi\gamma\beta = \pi\alpha$. А если $\pi\sigma = \pi\beta$, то $\pi\gamma\sigma$ не может равняться $\pi\gamma\alpha$, поскольку это подразумевало бы $\pi\gamma^2\sigma = \pi\gamma^2\alpha$, ..., $\pi\gamma^c\sigma = \pi\gamma^c\alpha$. Таким образом, все элементы $\pi\gamma^k$ имеют эквивалентное поведение по отношению к их наследникам в σ.

Обратите внимание на то, что если $j \ge 0$, то существует такое $k \le j$, что $\pi\sigma^j = \pi\alpha^k\beta^{j-k} = \pi\beta^j\gamma^k$. Следовательно, $\pi\sigma^{g/c} = \pi\gamma^{t+k}$ эквивалентно π, и такое же поведение повторится. Первый возврат к π за g этапов происходит тогда и только тогда, когда $t + k$ является взаимно простым для c.

75. Примените результат из предыдущего упражнения для $g = mn$, $a = m$, $b = n$, $c = mn/d$. Значение t удовлетворяет условиям $t \equiv 0$ (по модулю m) и $t + d \equiv 0$ (по модулю n). Отсюда следует, что $k + t \perp c$ тогда и только тогда, когда $(d - k)m/d \perp kn/d$.

Замечание. Модулярный код Грэя из упр. 7.2.1.1–78 является гамильтоновым путем от $(0, 0)$ к $(m - 1, (-m) \bmod n)$. Он является гамильтоновым циклом тогда и только тогда, когда m кратно n. Естественно было бы сделать предположение (ложное), что существует по крайней мере один гамильтонов цикл для $d > 1$. Но П. Эрдеш (P. Erdős) и В. Т. Троттер (W. T. Trotter) обнаружили [см. *J. Graph Theory* **2** (1978), p.137–142], что если p и $2p + 1$ являются нечетными простыми числами, то нет подходящего k для $m = p(2p + 1)(3p + 1)$ и $n = (3p + 1)\prod_{q=1}^{3p} q^{[q \text{ простое}][q \ne p][q \ne 2p+1]}$.

См. работу Дж. Э. Галлиана (J. A. Gallian) в *Mathematical Intelligencer* **13**, 3 (Summer 1991), p.40–43, где приводятся интересные факты о других типах циклов в $O_m \times O_n$.

76. Предположим, что проход начинается в нижнем левом углу. Для m и n, кратных 3, нет решений, поскольку 2/3 клеток недостижимы в таком случае. В противном случае, допуская $d = \gcd(m, n)$ и поступая, как в предыдущем упражнении, но с $(x, y)\alpha =$

$((x+2) \bmod m, (y+1) \bmod n)$ и $(x,y)\beta = ((x+1) \bmod m, (y+2) \bmod n)$, находим ответ:

$$\sum_{k=0}^{d} \binom{d}{k} [\gcd((2d-k)m, (k+d)n) = d \text{ или } (mn \perp 3 \text{ и } \gcd((2d-k)m, (k+d)n) = 3d)].$$

77.
```
01  * Генератор перестановок по типу Хипа.
02  N     IS    10              Значение n (3 илн больше, но не слишком).
03  t     IS    $255
04  j     IS    $0              8j
05  k     IS    $1              8k
06  ak    IS    $2
07  aj    IS    $3
08        LOC   Data_Segment
09  a     GREG  @               Базовый адрес для a_0 ... a_{n-1}.
10  A0    IS    @
11  A1    IS    @+8
12  A2    IS    @+16
13        LOC   @+8*N           Пространство для a_0 ... a_{n-1}.
14  c     GREG  @-8*3           Положение 8c_0.
15        LOC   @-8*3+8*N       8c_3 ... 8c_{n-1}, в исходном состоянии нуль.
16        OCTA  -1              8c_n = -1, удобная сигнальная метка.
17  u     GREG  0               Содержимое a_0, кроме внутреннего цикла.
18  v     GREG  0               Содержимое a_1, кроме внутреннего цикла.
19  w     GREG  0               Содержимое a_2, кроме внутреннего цикла.
20        LOC   #100
21  1H    STCO  0,c,k           B-A   c_k ← 0.
22        INCL  k,8             B-A   k ← k+1.
23  0H    LDO   j,c,k           B     j ← c_k.
24        CMP   t,j,k           B
25        BZ    t,1B            B     Цикл, если c_k = k.
26        BN    j,Done          A     Завершить, если c_k < 0 (k = n).
27        LDO   ak,a,k          A-1   Извлечь a_k.
28        ADD   t,j,8           A-1
29        STO   t,c,k           A-1   c_k ← j+1.
30        AND   t,k,#8          A-1
31        CSZ   j,t,0           A-1   Установить j ← 0, если k четно.
32        LDO   aj,a,j          A-1   Извлечь a_j.
33        STO   ak,a,j          A-1   Заменить на a_k.
34        CSZ   u,j,ak          A-1   Установить u ← a_k, если j = 0.
35        SUB   j,j,8           A-1   j ← j-1.
36        CSZ   v,j,ak          A-1   Установить v ← a_k, если j = 0.
37        SUB   j,j,8           A-1   j ← j-1.
38        CSZ   w,j,ak          A-1   Установить w ← a_k, если j = 0.
39        STO   aj,a,k          A-1   Заменить a_k тем, что было a_j.
40  Inner PUSHJ 0,Visit         A
          ...                         (См. (42))
55        PUSHJ 0,Visit         A
56        SET   t,u             A     Обменять u ↔ w.
57        SET   u,w             A
58        SET   w,t             A
```

```
59          SET   k,8*3          A     k ← 3.
60          JMP   0B             A
61    Main  LDO   u,A0           1
62          LDO   v,A1           1
63          LDO   w,A2           1
64          JMP   Inner          1     ▮
```

78. Строки 31–38 дают $2r-1$ инструкций; строки 61–63 — r и строки 56–58 — $3+(r-2)[r$ четное$]$ инструкций (см. $\omega(r-1)$ в ответе к упр. 40). Следовательно, общее время выполнения равно $((2r!+2)A+2B+r-5)\mu+((2r!+2r+7+(r-2)[r\text{ четно}])A+7B-r-4)\upsilon$, где $A=n!/r!$ и $B=n!(1/r!+\cdots+1/n!)$.

79. `SLU u,[#f],t; SLU t,a,4; XOR t,t,a; И t,t,u; SRU u,t,4; OR t,t,u; XOR a,a,t`. Здесь, как и в ответе к упр. 1.3.1'–34, '`[#f]`' обозначает регистр с константой `#f`.

80. `SLU u,a,t; MXOR u,[#8844221188442211],u; И u,u,[#ff000000]; SRU u,u,t; XOR a,a,u`. Это уловка, которая преобразует #12345678 в #13245678, когда $t=4$, но (45) все еще работает.

Даже более быстрое и хитроумное решение будет процедурой, аналогичной (42). Рассмотрим

```
PUSHJ 0,Посетить; MXOR a,a,c1; PUSHJ 0,Посетить; ... MXOR a,a,c5; PUSHJ 0,Посетить
```

где `c1`, ..., `c5` являются константами, с помощью которых #12345678 последовательно преобразуется в #12783456, #12567834, #12563478, #12785634, #12347856. Другие инструкции, выполняемые с относительной частотой только 1/6 или 1/24, могут позаботиться о перестановке ниблов внутри байтов и между ними. Остроумный способ, но он не лучше (46), в связи с накладными расходами на выполнение `PUSHJ/POP`.

81.
```
t   IS   $255 ;k IS $0 ;kk IS $1 ;c IS $2 ;d IS $3
    SET  k,1      k ← 1.
3H  SRU  d,a,60   d ← самый левый нибл.
    SLU  a,a,4    a ← 16a mod 16^16.
    CMP  c,d,k
    SLU  kk,k,2
    SLU  d,d,kk
    OR   t,t,d    t ← t + 16^k d.
    PBNZ c,1B     Вернуться к основному циклу, если d ≠ k.
    INCL k,1      k ← k + 1.
    PBNZ a,3B     Вернуться ко второму циклу, если k < n.   ▮
```

82. $\mu+(5n!+11A-(n-1)!+6)\upsilon=((5+10/n)\upsilon+O(n^{-2}))n!$ плюс время посещения, где $A=\sum_{k=1}^{n-1}k!$ обозначает количество циклов на этапе 3H.

83. С должной инициализацией и 13-октабайтовой таблицей потребуется всего несколько инструкций MMIX:

```
magic  GREG #8844221188442211
0H     ⟨Посетить регистр a⟩
       PBN  c,Sigma
Tau    MXOR t,magic,a; ANDNL t,#ffff; JMP 1F
Sigma  SRU  t,a,20; SLU a,a,4; ANDNML a,#f00
1H     XOR  a,a,t; SLU c,c,1
2H     PBNZ c,0B; INCL p,8
3H     LDOU c,p,0; PBNZ c,0B                    ▮
```

84. Предполагая, что все процессоры обладают одинаковой мощностью, допустим, что k-й процессор генерирует все перестановки ранга r для $(k-1)n!/p \le r < kn!/p$ с помощью любого метода на основе контрольных таблиц $c_1 \ldots c_n$. Стартовая и конечная контрольные таблицы легко вычисляются за счет преобразования их рангов в систему обозначений со смешанным основанием (упр. 12).

85. Можно применить технологию, использованную в алгоритме 3.4.2P. Чтобы вычислить $k = r(\alpha)$, сначала следует установить $a'_{a_j} \leftarrow j$ для $1 \le j \le n$ (обратная перестановка). Затем установить $k \leftarrow 0$ и для $j = n, n-1, \ldots, 2$ (в этом порядке) установить $t \leftarrow a'_j$, $k \leftarrow kj + t - 1$, $a_t \leftarrow a_j$, $a'_{a_j} \leftarrow t$. Чтобы вычислить $r^{[-1]}(k)$, начнем с $a_1 \leftarrow 1$. Тогда для $j = 2, \ldots, n-1, n$ (в этом порядке) установить $t \leftarrow (k \bmod j)+1$, $a_j \leftarrow a_t$, $a_t \leftarrow j$, $k \leftarrow \lfloor k/j \rfloor$ [см. статью C. Плещински (S. Pleszczyński) в *Inf. Proc. Letters* **3** (1975), p.180–183; а также статью В. Мирвольда (W. Myrvold) и Ф. Раски в *Inf. Proc. Letters* **79** (2001), p.281–284].

Другой метод предпочтительнее, если нужно использовать ранжирование и неранжирование только для $n^{\underline{m}}$ *вариаций* $a_1 \ldots a_m$ of $\{1, \ldots, n\}$: чтобы вычислить $k = r(a_1 \ldots a_m)$, начнем с $b_1 \ldots b_n \leftarrow b'_1 \ldots b'_n \leftarrow 1 \ldots n$; затем для $j = 1, \ldots, m$ (в этом порядке) установить $t \leftarrow b'_{a_j}$, $b_t \leftarrow b_{n+1-j}$ и $b'_{b_t} \leftarrow t$; наконец установить $k \leftarrow 0$ и для $j = m, \ldots, 1$ (в этом порядке) установить $k \leftarrow k \times (n+1-j) + b'_{a_j} - 1$. Чтобы вычислить $r^{[-1]}(k)$, начнем с $b_1 \ldots b_n \leftarrow 1 \ldots n$; затем для $j = 1, \ldots, m$ (в этом порядке) установить $t \leftarrow (k \bmod (n+1-j))+1$, $a_j \leftarrow b_t$, $b_t \leftarrow b_{n+1-j}$, $k \leftarrow \lfloor k/(n+1-j) \rfloor$. (См. упр. 3.4.2–15 для случаев больших n и малых m.)

86. Если $x \prec y$ и $y \prec z$, то алгоритм никогда не переместит ни y влево от x, ни z влево от y, поэтому не удастся проверить x по отношению к z.

87. Они имеют лексикографический порядок, а в алгоритме P используется отраженный порядок Грэя.

88. Создайте обратные перестановки с помощью $a'_0 < a'_1 < a'_2$, $a'_3 < a'_4 < a'_5$, $a'_6 < a'_7$, $a'_8 < a'_9$, $a'_0 < a'_3$, $a'_6 < a'_8$.

89. (a) Пусть $d_k = \max\{j \mid 0 \le j \le k$ и j является нетривиальным$\}$, где 0 считается нетривиальным. Эта таблица легко предварительно вычисляется, поскольку j является тривиальным тогда и только тогда, когда он идет после $\{1, \ldots, j-1\}$. Установить $k \leftarrow d_n$ на этапе V2 и $k \leftarrow d_{k-1}$ — на этапе V5. (Предполагая $d_n > 0$.)

(b) Теперь $M = \sum_{j=1}^{n} t_j[j$ является нетривиальным$]$.

(c) Есть по крайней мере две топологические сортировки $a_j \ldots a_k$ множества $\{j, \ldots, k\}$, и любая из них может быть помещена после любой топологической сортировки $a_1 \ldots a_{j-1}$ of $\{1, \ldots, j-1\}$.

(d) алгоритм 2.2.3T повторно выводит минимальные элементы (элементы без предшествующих им элементов), удаляя их из графа отношений. Используем его в обратном порядке, повторно удаляя и присваивая высшие маркеры *максимальным* элементам (элементам без последующих элементов). Если существует только один максимальный элемент, то он является тривиальным. Если k и l максимальны, то оба они выводятся перед любым элементом x с $x \prec k$ или $x \prec l$, поскольку на этапах T5 и T7 максимальные элементы хранятся в очереди (а не в стеке). Таким образом, если k является нетривиальным и выводится первым, то элемент l может стать тривиальным, но следующий нетривиальный элемент j не будет выводиться до l; а k не связан с l.

(e) Пусть для нетривиального t выполняется $s_1 < s_2 < \cdots < s_r = N$. Тогда имеем $s_j \ge 2s_{j-2}$, согласно (c). Следовательно, $M = s_2 + \cdots + s_r \le s_r(1 + \frac{1}{2} + \frac{1}{4} + \cdots) + s_{r-1}(1 + \frac{1}{2} + \frac{1}{4} + \cdots) < 4s_r$.

(На самом деле, как заметил М. Печарски (M. Peczarski), можно получить более точную оценку. Пусть $s_0 = 1$, нетривиальными индексами являются $0 = k_1 < k_2 < \cdots < k_r$,

и пусть $k'_j = \max\{k \mid 1 \le k < k_j,\ k \not\prec k_j\}$ для $j > 1$. Тогда $k'_j \ge k_{j-1}$. Существует s_j топологических сортировок $\{1, \ldots, k_{j+1}\}$, которые завершаются k_{j+1}, и существует по крайней мере s_{j-1} топологических сортировок, которые завершаются k'_{j+1}, поскольку каждая из s_{j-1} топологических сортировок $\{1, \ldots, k_j - 1\}$ может быть расширена. Следовательно,

$$s_{j+1} \ \ge\ s_j + s_{j-1} \qquad \text{для } 1 \le j < \tau.$$

Теперь пусть $y_0 = 0$, $y_1 = F_2 + \cdots + F_\tau$, и $y_j = y_{j-2} + y_{j-1} - F_{\tau+1}$ для $1 < j < \tau$. Тогда

$$F_{\tau+1}(s_1 + \cdots + s_\tau) + \sum_{j=1}^{\tau-1} y_j(s_{\tau+1-j} - s_{\tau-j} - s_{\tau-1-j}) \ =\ (F_2 + \cdots + F_{\tau+1})s_\tau,$$

и каждое значение $y_j = F_{\tau+1} - 2F_j - (-1)^j F_{\tau+1-j}$ является неотрицательным. Следовательно, $s_1 + \cdots + s_\tau \le \big((F_2 + \cdots + F_{\tau+1})/F_{\tau+1}\big)s_\tau \approx 2{,}6s_\tau$. В следующем упражнении показано, что это ограничение является наилучшим из возможных.)

90. Число N таких перестановок равно F_{n+1}, согласно упр. 5.2.1–25. Следовательно, $M = F_{n+1} + \cdots + F_2 = F_{n+3} - 2 \approx \phi^2 N$. Кстати, обратите внимание, что все такие перестановки удовлетворяют $a_1 \ldots a_n = a'_1 \ldots a'_n$. Их можно упорядочить в виде пути Грэя (упр. 7.2.1.1–89).

91. Поскольку $t_j = (j-1)(j-3)\ldots(2 \text{ или } 1)$, находим $M = \big(1 + 2/\sqrt{\pi n} + O(1/n)\big)N$.

Замечание. Таблицы инверсии $c_1 \ldots c_{2n}$ для перестановок, удовлетворяющих условиям (49), характеризуются следующими условиями: $c_1 = 0$, $0 \le c_{2k} \le c_{2k-1}$, $0 \le c_{2k+1} \le c_{2k-1} + 1$.

92. Общее число пар (R, S), где R является частичным упорядочением, а S — линейным упорядочением, которое включает R, равно произведению P_n и ожидаемому числу топологических сортировок; оно также равно произведению Q_n и $n!$. Отсюда следует ответ: $n!\,Q_n/P_n$.

Процесс вычисления P_n и Q_n рассматривается в разделе 7.2.3. Для $1 \le n \le 12$ ожидаемое число топологических сортировок приблизительно равно

$$(1,\ 1.33,\ 2.21,\ 4.38,\ 10.1,\ 26.7,\ 79.3,\ 262,\ 950,\ 3760,\ 16200,\ 74800).$$

Асимптотические значения для $n \to \infty$ были выведены Г. Р. Брайтвеллом (G. R. Brightwell), Х. Ю. Промелем (H. J. Prömel) и А. Стегер (A. Steger) [см. *J. Combinatorial Theory* **A73** (1996), p.193–206], но асимптотическое поведение очень отличается от поведения n в диапазонах, которые встречаются на практике. Значения Q_n впервые были определены Ш. П. Аванном (S. P. Avann) для $n \le 5$ [см. *Æquationes Math.* **8** (1972), p.95–102].

93. Основная идея заключается в введении подстановочных элементов $n+1$ и $n+2$ с $j \prec n+1$ и $j \prec n+2$ для $1 \le j \le n$ и поиске всех топологических сортировок такого расширенного отношения с помощью смежных замен. Затем нужно взять каждую *вторую* перестановку, игнорируя подстановочные элемеиты. Для этого можно использовать алгоритм, аналогичный алгоритму V, ио с рекурсией, которая сводит n к $n-2$ за счет вставки $n-1$ и n среди $a_1 \ldots a_{n-2}$ всеми возможными способами, предполагая, что $n-1 \not\prec n$, иногда обменивая $n+1$ с $n+2$. [См. статью Г. Пруссе и Ф. Раски в *SICOMP* **23** (1994), p.373–386. Бесцикловая реализация описана Э. Р. Канфилдом и С. Дж. Вильямсоном в *Order* **12** (1995), p.57–75.]

94. Случай $n = 3$ иллюстрирует общую идею структуры, которая начинается с $1 \ldots (2n)$ и заканчивается $1(2n)2(2n-1)\ldots n(n+1)$: 123456, 123546, 123645, 132645, 132546, 132456, 142356, 142536, 142635, 152634, 152436, 152346, 162345, 162435, 162534.

Сочетания можно рассматривать как инволюции $\{1, \ldots, 2n\}$, которые содержат n циклов. В таком представлении эта структура использует две транспозиции на один этап.

Обратите внимание на то, что таблицы инверсии C только что перечисленных перестановок имеют следующий вид: 000000, 000100, 000200, 010200, 010100, 010000, 020000, 020100, 020200, 030200, 030100, 030000, 040000, 040100, 040200. Вообще, $C_1 = C_3 = \cdots = C_{2n-1} = 0$, и n-кортежи $(C_2, C_4, \ldots, C_{2n})$ проходят отраженный код Грэя на основаниях $(2n-1, 2n-3, \ldots, 1)$. Таким образом, процесс генерации может легко стать бесцикловым, если это потребуется [см. статью Т. Уолша (T. Walsh) в *J. Combinatorial Math. and Combinatorial Computing* **36** (2001), p.95–118, Section 1].

Замечание. Алгоритмы генерации всех сочетаний восходят еще к Й. Ф. Пфаффу [см. *Abhandlungen Akad. Wissenschaften* (Berlin: 1814–1815), p.124–125], который описал две такие процедуры. Первый метод является лексикографическим, который также соответствует лексикографическому порядку таблиц инверсии C. Его второй метод соответствует *колексикографическому* порядку этих таблиц. Четная и нечетная перестановки меняются в обоих случаях.

95. Нужно генерировать обратные перестановки с $a'_1 < a'_n > a'_2 < a'_{n-1} > \cdots$, используя алгоритм V. (О количестве решений см. упр. 5.1.4–23.)

96. Например, можно начать с $a_1 \ldots a_{n-1} a_n = 2 \ldots n1$ и $b_1 b_2 \ldots b_n b_{n+1} = 12 \ldots n1$ и использовать алгоритм P для генерации $(n-1)!$ перестановок $b_2 \ldots b_n$ для $\{2, \ldots, n\}$. Сразу после этого алгоритм выполняет $b_i \leftrightarrow b_{i+1}$, затем $a_{b_{i-1}} \leftarrow b_i$, $a_{b_i} \leftarrow b_{i+1}$, $a_{b_{i+1}} \leftarrow b_{i+2}$ и посещает $a_1 \ldots a_n$.

97. Используйте алгоритм X с $t_k(a_1, \ldots, a_k) = \text{`}a_k \neq k\text{'}$.

98. С помощью обозначений из упр. 47 имеем: $N_k = \sum \binom{k}{j}(-1)^j (n-j)^{\underline{k-j}}$ по методу (упр. 1.3.3–26). Если $k = O(\log n)$, тогда $N_{n-k} = (n! e^{-1}/k!)(1 + O(\log n)^2/n)$; следовательно, $A/n! \approx (e-1)/e$ и $B/n! \approx 1$. Количество ссылок на память, согласно условиям упр. 48, следовательно, равно $\approx A + B + 3A + B - N_n + 3A \approx n!\,(9 - \frac{8}{e}) \approx 6{,}06n!$, т.е. приблизительно 16,5 на одно неупорядочение. [См. описание аналогичного метода в работе С. Дж. Акля (S. G. Akl), *BIT* **20** (1980), p.2–7.]

99. Допустим, что L_n генерирует $D_n \cup D_{n-1}$, начиная с $(1\ 2\ \ldots\ n)$, затем $(2\ 1\ \ldots\ n)$, и заканчивая $(1\ \ldots\ n{-}1)$; например, $L_3 = (1\,2\,3)$, $(2\,1\,3)$, $(1\,2)$. Тогда можно генерировать D_{n+1} как $K_{nn}, \ldots, K_{n2}, K_{n1}$, где $K_{nk} = (1\ 2\ \ldots\ n)^{-k}(n\ n{+}1) L_n (1\ 2\ \ldots\ n)^k$; например, D_4 имеет вид

$$(1\,2\,3\,4),\ (2\,1\,3\,4),\ (1\,2)(3\,4),\ (3\,1\,2\,4),\ (1\,3\,2\,4),\ (3\,1)(2\,4),\ (2\,3\,1\,4),\ (3\,2\,1\,4),\ (2\,3)(1\,4).$$

Обратите внимание на то, что K_{nk} начинается с цикла $(k{+}1\ \ldots\ n\ 1\ \ldots\ k\ n{+}1)$ и заканчивается $(k{+}1\ \ldots\ n\ 1\ \ldots\ k{-}1)(k\ n{+}1)$, поэтому умножение в обратном порядке на $(k{-}1\ k)$ переводит нас от K_{nk} к $K_{n(k-1)}$. Кроме того, умножение в обратном порядке на $(1\ n)$ возвращает нас от последнего элемента D_{n+1} к первому. Умножение в обратном порядке на $(1\ 2\ n{+}1)$ возвращает нас от последнего элемента D_{n+1} к элементу $(2\ 1\ 3\ \ldots\ n)$, с которого можно вернуться к элементу $(1\ 2\ \ldots\ n)$, следуя за циклом для D_n в обратном направлении, таким образом, завершая список L_{n+1}, как и требовалось.

100. Используйте алгоритм X с $t_k(a_1, \ldots, a_k) = \text{`}p > 0$ или $l[q] \neq k+1\text{'}$.

Замечание. Количество неразложимых перестановок равно: $[z^n]\,(1 - 1/\sum_{k=0}^{\infty} k!\, z^k)$; см. работу Л. Комтета (L. Comtet) в *Comptes Rendus Acad. Sci.* **A275** (Paris, 1972), p.569–572. А. Д. Кинг (A. D. King) [*Discrete Math.* **306** (2006), 508–516] показал, что неразложимые перестановки можно эффективно генерировать, делая только одну транспозицию на каждом этапе. Действительно, для этого достаточно *смежных* транспозиций. Например, для $n = 4$ неразложимыми перестановками являются 3142, 3412, 3421, 3241, 2341, 2431, 4231, 4321, 4312, 4132, 4123, 4213, 2413.

101. В данном случае генератор лексикографических инволютивных перестановок аналогичен алгоритму X.

Y1. [Инициализировать.] Установить $a_k \leftarrow k$ и $l_{k-1} \leftarrow k$ для $1 \le k \le n$. Затем установить $l_n \leftarrow 0$, $k \leftarrow 1$.

Y2. [Перейти на уровень k.] Если $k > n$, посетить $a_1 \ldots a_n$ и перейти к Y3. В противном случае установить $p \leftarrow l_0$, $u_k \leftarrow p$, $l_0 \leftarrow l_p$, $k \leftarrow k+1$ и повторить этот этап. (Предполагается, что $a_p = p$.)

Y3. [Уменьшить k.] Установить $k \leftarrow k-1$ и завершить, если $k = 0$. В противном случае установить $q \leftarrow u_k$ и $p \leftarrow a_q$. Если $p = q$, установить $l_0 \leftarrow q$, $q \leftarrow 0$, $r \leftarrow l_p$ и $k \leftarrow k+1$ (подготовка к $a_p > p$). В противном случае установить $l_{u_{k-1}} \leftarrow q$, $r \leftarrow l_q$ (подготовка к $a_p > q$).

Y4. [Увеличить a_p.] Если $r = 0$, то перейти к Y5. В противном случае установить $l_q \leftarrow l_r$, $u_{k-1} \leftarrow q$, $u_k \leftarrow r$, $a_p \leftarrow r$, $a_q \leftarrow q$, $a_r \leftarrow p$, $k \leftarrow k+1$ и перейти к Y2.

Y5. [Восстановить a_p.] Установить $l_0 \leftarrow p$, $a_p \leftarrow p$, $a_q \leftarrow q$, $k \leftarrow k-1$ и вернуться к Y3. ∎

Пусть $t_{n+1} = t_n + nt_{n-1}$, $a_{n+1} = 1 + a_n + na_{n-1}$, $t_0 = t_1 = 1$, $a_0 = 0$, $a_1 = 1$ (см. уравнения 5.1.4-(40)). Этап Y2 выполняется t_n раз с $k > n$ и a_n раз с $k \le n$. Этап Y3 выполняется a_n раз с $p = q$ и всего $a_n + t_n$ раз. Этап Y4 выполняется $t_n - 1$ раз, а этап Y5 — a_n раз. Общее число мемсов для всех t_n выводов, следовательно, равно $11a_n + 12t_n$, где $a_n < 1.25331414t_n$. (Конечно, этот алгоритм можно оптимизировать в случае необходимости.)

102. Создадим список L_n, который начинается с () и заканчивается $(n{-}1\ n)$, начиная с $L_3 = (), (1\ 2), (1\ 3), (2\ 3)$. Если n нечетно, то L_{n+1} имеет вид $L_n, K_{n1}^R, K_{n2}, \ldots, K_{nn}^R$, где $K_{nk} = (k\ \ldots\ n)^- L_{n-1}(k\ \ldots n)(k\ n{+}1)$. Например:

$$L_4 = (), (1\ 2), (1\ 3), (2\ 3), (2\ 3)(1\ 4), (1\ 4), (2\ 4), (1\ 3)(2\ 4), (1\ 2)(3\ 4), (3\ 4).$$

Если n четно, то L_{n+1} имеет вид $L_n, K_{n(n-1)}, K_{n(n-2)}^R, \ldots, K_{n1}, (1\ n{-}2)L_{n-1}^R(1\ n{-}2)(n\ n{+}1)$.

Более подробно эта тема рассматривается в работе Т. Уолша (см. в ответе к упр. 94).

103. Следующее элегантное решение предложено К. Сэвидж. Требуется всего $n-2$ разных операций ρ_j для $1 < j < n$, где ρ_j заменяет $a_{j-1}a_ja_{j+1}$ на $a_{j+1}a_{j-1}a_j$, если j четно, $a_ja_{j+1}a_{j-1}$, если j нечетно. Предположим, что $n \ge 4$; пусть $A_4 = (\rho_3\rho_2\rho_2\rho_3)^3$. Вообще, A_n начинается и заканчивается ρ_{n-1} и содержит $2n-2$ появлений ρ_{n-1}. Чтобы получить A_{n+1}, заменим k-ю операцию ρ_{n-1} в A_n на $\rho_nA_n'\rho_n$, где $k = 1, 2, 4, \ldots, 2n-2$, если n четно, и $k = 1, 3, \ldots, 2n-3, 2n-2$, если n нечетно, и где A_n' равно A_n с удаленнымн первым или последним элементом. Тогда, если начать с $a_1 \ldots a_n = 1 \ldots n$, операцин ρ_{n-1} в A_n приведут к тому, что a_n будет проходить последовательные значения $n \to p_1 \to n \to p_2 \to \cdots \to p_{n-1} \to n$, где $p_1 \ldots p_{n-1} = (n{-}1{-}[n \text{ четное}]) \ldots 4213 \ldots (n{-}1{-}[n \text{ нечетное}])$; финальная перестановка снова будет иметь вид $1 \ldots n$.

104. (a) Для хорошо сбалансированной перестановки $\sum_{k=1}^n ka_k = n(n+1)^2/4$.

(b) Замените k на a_k при суммировании по k.

(c) Достаточно быстрый способ подсчета, когда n не очень велико, можно предложить на основе оптимизнрованного алгоритма простых изменений из упр. 16, поскольку величина $\sum ka_k$ изменяется простым способом с каждой смежной заменой и поскольку $n-1$ из каждых n этапов являются "охотами", которые могут быть выполнены очень быстро. Работу можно сократить вдвое, рассматривая только перестановки, в которых 1 предшествует 2. Значения для $1 \le n \le 15$ равны 0, 0, 0, 2, 6, 0, 184, 936, 6688, 0, 420480, 4298664, 44405142, 0, 6732621476.

105. (a) Для каждой перестановки $a_1 \ldots a_n$, вставьте $\prec$ между a_j и a_{j+1}, если $a_j > a_{j+1}$; вставьте между ними либо $\equiv$, либо $\prec$, если $a_j < a_{j+1}$. (Следовательно, перестановка с k "восхождениями" достигает 2^k слабых порядков. Слабый порядок иногда называется

"предпочтительным упорядочением"; в упр. 5.3.1–4 показано, что их существует приблизительно $n!/(2(\ln 2)^{n+1})$.) Код Грэя для слабого порядка, в котором на каждом этапе меняется $\prec \leftrightarrow \equiv$ и/или $a_j \leftrightarrow a_{j+1}$, можно получить, комбинируя алгоритм P с двоичным кодом Грэя на восхождениях.

(b) Начнем с $a_1 \ldots a_n a_{n+1} = 0 \ldots 00$ и $a_0 = -1$. Будем выполнять алгоритм L, пока он не остановится с $j = 0$. Далее нужно найти такое k, что $a_1 > \cdots > a_k = a_{k+1}$, и завершить, если $k = n$. В противном случае установить $a_l \leftarrow a_{k+1} + 1$ для $1 \le l \le k$ и перейти к этапу L4. (См. работу М. Мора (M. Mor) и А. С. Френкеля (A. S. Fraenkel) в *Discrete Math.* **48** (1984), p.101–112. Слабо упорядоченные последовательности характеризуются тем, что если появляется k и $k > 0$, то появляется и $k - 1$.)

106. Все слабо упорядоченные последовательности можно получить с помощью последовательности элементарных операций $a_i \leftrightarrow a_j$ или $a_i \leftarrow a_j$. (Возможно, можно было бы сильнее ограничить эти преобразования, допуская только $a_j \leftrightarrow a_{j+1}$ или $a_j \leftarrow a_{j+1}$ для $1 \le j < n$.)

107. На каждом этапе величина $\sum_{k=1}^{n} 2^k [a_k = k]$ растет, как заметил Х. С. Вильф, поэтому игра должна завершиться. Можно применить по крайней мере три подхода для решения: один плохой, один хороший и один еще лучше.

Плохой заключается в розыгрыше всех 13! тасовок и записи самой длинной. Этот метод дает правильный ответ, но 13! равно 6 227 020 800, и в среднем игра продолжается приблизительно 8,728 этапа.

Хороший способ [см. Э. Пеппердайн (A. Pepperdine), *Math. Gazette* **73** (1989), p.131–133] заключается в розыгрыше в обратном порядке, начиная с последней позиции $1* \ldots *$, где $*$ обозначает карту, повернутую лицевой стороной вниз. Карта переворачивается, только если она имеет существенное значение. Для продвижения назад от заданной позиции $a_1 \ldots a_n$ рассмотрим все такие $k > 1$, что либо $a_k = k$, либо $a_k = *$ и k еще не были перевернуты вверх. Таким образом, следующими для последней позиции являются $21* \ldots *$, $3*1* \ldots *$, $\ldots$, $n* \ldots *1$. Некоторые позиции (как $6{*}{*}213$ для $n = 6$) не имеют предшественников, даже если не все карты повернуты вверх. Легко систематически исследовать дерево потенциальных обратных игр и действительно показать, что количество узлов с t символами $*$ точно равно $(n-1)!/t!$. Следовательно, общее число рассматриваемых узлов точно равно $\lfloor (n-1)!\, e \rfloor$. Для $n = 13$ оно равно 1,302,061,345.

Но значительно лучший способ заключается в розыгрыше в прямом порядке, начиная с исходной позиции $* \ldots *$ и переворачивая верхнюю карту, если она повернута лицевой стороной вниз, пробегая все $(n-1)!$ перестановок $\{2, \ldots, n\}$ по мере переворачивания карт. Если известно, что нижние $n - m$ карт равны $(m+1)(m+2) \ldots n$ в этом порядке, то возможно не больше $f(m)$ дальнейших действий. Таким образом, не нужно дальше прослеживать линию игры, если она не будет продолжаться достаточно долго, чтобы это было интересно. Генератор перестановок, как алгоритм X, позволяет использовать одно вычисление для всех перестановок с одинаковым префиксом и отвергнуть не имеющие большого значения префиксы. Карта в позиции j необязательно должна принимать значение j, когда она переворачивается. Для $n = 13$ в этом методе необходимо рассмотреть только $(1, 11, 940, 6960, 44745, 245083, 1118216, 4112676, 11798207, 26541611, 44380227, 37417359)$ ветвей на уровнях $(1, 2, \ldots, 12)$ и выполнить в общем только 482 663 902 действия вперед. Хотя он повторяет некоторые линии игры, но раннее обрезание невыгодных ветвей ускоряет вычисления в 11 раз по сравнению с обратным методом для $n = 13$.

Единственный способ достижения длины 80 заключается в том, чтобы начать с 2 9 4 5 11 12 10 1 8 13 3 6 7.

108. Этот результат верен для любой игры, в которой

$$a_1 \ldots a_n \to a_k a_{p(k,2)} \ldots a_{p(k,k-1)} a_1 a_{k+1} \ldots a_n$$

для $a_1 = k$, где $p(k,2)\dots p(k,k-1)$ является произвольной перестановкой $\{2,\dots,k-1\}$. Допустим, a_1 принимает точно m различных значений $d(1) < \cdots < d(m)$ во время игры. Докажем, что при этом произойдет не более F_{m+1} перестановок, включая исходную тасовку. Это утверждение очевидно для $m = 1$.

Пусть $d(j)$ является исходным значением $a_{d(m)}$, где $j < m$, и допустим, что $a_{d(m)}$ изменяется на этапе r. Если $d(j) = 1$, то количество перестановок равно $r + 1 \le F_m + 1 \le F_{m+1}$. В противном случае $r \le F_{m-1}$ и не более F_m дальнейших перестановок последуют за этапом r [см. *SIAM Review* **19** (1977), p.739–741].

Значения $f(n)$ для $1 \le n \le 16$ имеют вид (0, 1, 2, 4, 7, 10, 16, 22, 30, 38, 51, 65, 80, 101, 113, 139) и достижимы за (1, 1, 2, 2, 1, 5, 2, 1, 1, 1, 1, 1, 1, 4, 6, 1) способов соответственно. Единственная самая длинная перестановка для $n = 16$ имеет вид

$$9\ 12\ 6\ 7\ 2\ 14\ 8\ 1\ 11\ 13\ 5\ 4\ 15\ 16\ 10\ 3.$$

109. Согласно прямому методу из ответа к упр. 107, фуикция $f(n)$ растет с интенсивиостью, по крайней мере равиой $n \log n$ (по сравиению с процедурой раскладывания (coupon collecting)).

110. Для $0 \le j \le 9$ создадим битовые векторы $A_j = [a_j \in S_1]\dots[a_j \in S_m]$ и $B_j = [j \in S_1]\dots[j \in S_m]$. Тогда количество j, таких, что $A_j = v$, должно равняться количеству k, таких, что $B_k = v$, для всех битовых векторов v. И в таком случае значения $\{a_j \mid A_j = v\}$ следует присвоить перестановкам $\{k \mid B_k = v\}$ всеми возможными способами.

Например, битовые векторы в данной задаче имеют вид

$$(A_0,\dots,A_9) = (9,6,8,\mathtt{b},5,4,0,\mathtt{a},2,0), \qquad (B_0,\dots,B_9) = (5,0,8,6,2,\mathtt{a},4,\mathtt{b},9,0),$$

в шестнадцатеричной системе счисления Следовательно, $a_0\dots a_9 = 8327061549$ или 8327069541.

В более крупной задаче битовые векторы следовало бы разместить в хеш-таблице. Было бы лучше дать этот ответ на осиове классов эквивалентности, а не перестановок. Действительно, эта задача имеет мало общего с перестановками.

111. В ориентированном графе с $n!/2$ вершинами $a_1\dots a_{n-2}$ и $n!$ ребрами $a_1\dots a_{n-2} \to a_2\dots a_{n-1}$ (по одному для каждой перестановки $a_1\dots a_n$) каждая вершина имеет полустепень входа 2 и полустепень выхода 2. Более того, анализируя пути $a_1\dots a_{n-2} \to a_2\dots a_{n-1} \to a_3\dots a_n \to a_4\dots a_n a_2 \to a_5\dots a_n a_2 a_1 \to \cdots \to a_2 a_1 a_3\dots a_{n-2}$, можно прийти к выводу, что любая вершина достижима из другой. Следовательно, по теореме 2.3.4.2G, существует контур Эйлера или эйлеров контур (Eulerian trail), причем этот контур эквивалентен универсальному циклу перестановок. Лексикографически наименьшим примером для $n = 4$ является (123124132134214324314234).

(Г. Херлберт (G. Hurlbert) и Г. Айзек (G. Isaak) [см. *Discrete Math.* **149** (1996), p.123–129] предложили другой привлекательиый подход: назовем *модулярным универсальным циклом перестановок* такой цикл из $n!$ цифр $\{0,\dots,n\}$, который обладает следующим свойством: каждая перестановка $a_1\dots a_n$ of $\{1,\dots,n\}$ возникает из последовательных цифр $u_1\dots u_n$ с допущением $a_j = (u_j - c) \bmod (n+1)$, где c является "отсутствующей" цифрой в $\{u_1,\dots,u_n\}$. Например, модулярный универсальный цикл (012032) является существенно единственным для $n = 3$, а лексикографически наименьшим для $n = 4$ является (012301420132014321430243). Если вершины $a_1\dots a_{n-2}$ и $a'_1\dots a'_{n-2}$ в диграфе (т.е. ориентированном графе) из предыдущего абзаца считаются эквивалентными для $a_1 - a'_1 \equiv \cdots \equiv a_{n-2} - a'_{n-2}$ (по модулю n), то получим ориентированный граф с $(n-1)!/2$ вершинами, контуры Эйлера которого соответствуют модулярным универсальным циклам перестановок для $\{1,\dots,n-1\}$.)

112. Согласно упр. 2.3.4.2–22, достаточно подсчитать орнентированные деревья с корнями в местах $12\ldots(n-2)$ диграфа из ответа к предыдущему упражнению. И эти деревья можно подсчитать, как в упр. 2.3.4.2–19. Для $n \le 6$ числа U_n оказываются соблазнительно простыми: $U_2 = 1$, $U_3 = 3$, $U_4 = 2^7 \cdot 3$, $U_5 = 2^{33} \cdot 3^8 \cdot 5^3$, $U_6 = 2^{190} \cdot 3^{49} \cdot 5^{33}$. (Здесь (121323) считается таким же циклом, как и (213231), но отличным от (131232).)

Марк Кук (Mark Cooke) обнаружил следующий поучительный и эффективный способ вычисления этих значеиий. Сначала обратим виимание на то, что универсальный цикл перестановок также эквивалентен *гамильтонову* циклу на графе Кэли с генераторамн $\sigma = (1\ 2\ \ldots\ n)$ и $\rho = (1\ 2\ \ldots\ n{-}1)$. Например, цикл в ответе к предыдущему упр. для $n = 4$ соответствует циклу $\sigma^3\rho^2\sigma\rho\sigma^2\rho^2\sigma^3\rho\sigma^2\rho^2\sigma\rho\sigma^2\rho$.

Теперь рассмотрнм матрицу $M = 2I - R - S$ с размерамн $n! \times n!$, где $R_{\pi\pi'} = [\pi' = \pi\rho]$ и $S_{\pi\pi'} = [\pi' = \pi\sigma]$. Существует такая матрица H, что $H^{-}RH$ и $H^{-}SH$ имеют диагональный блок, состоящий из k_λ копий $k_\lambda \times k_\lambda$ матриц R_λ и S_λ для каждого разбиения λ из n, где k_λ является $n!$, делениым на произведение длин уголков формы λ (теорема 5.1.4H), и где R_λ и S_λ являются матричными представлениями ρ и σ на основе диаграммы Юнга. (Доказательство можно иайти в книге Брюса Сагана [Bruce Sagan, *The Symmetric Group* (Pacific Grove, Calif.: Wadsworth & Brooks/Cole, 1991)].) Например, для $n = 3$ имеем:

$$R = \begin{pmatrix} 0&0&0&1&0&0\\ 0&0&0&0&0&1\\ 0&0&0&0&1&0\\ 1&0&0&0&0&0\\ 0&0&1&0&0&0\\ 0&1&0&0&0&0 \end{pmatrix},\quad S = \begin{pmatrix} 0&1&0&0&0&0\\ 0&0&1&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1\\ 0&0&0&1&0&0 \end{pmatrix},\quad H = \begin{pmatrix} 1&1&1&-1&1&0\\ 1&1&-1&0&0&-1\\ 1&1&0&1&-1&1\\ 1&-1&-1&1&0&1\\ 1&-1&1&0&1&-1\\ 1&-1&0&-1&-1&0 \end{pmatrix},$$

$$H^{-}RH = \begin{pmatrix} 1&0&0&0&0&0\\ 0&-1&0&0&0&0\\ 0&0&0&1&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&0&1\\ 0&0&0&0&1&0 \end{pmatrix},\qquad H^{-}SH = \begin{pmatrix} 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&0&-1&0&0\\ 0&0&1&-1&0&0\\ 0&0&0&0&0&-1\\ 0&0&0&0&1&-1 \end{pmatrix},$$

где строки и столбцы индексируются соответствующими перестановками 1, σ, σ^2, ρ, $\rho\sigma$, $\rho\sigma^2$; здесь $k_3 = k_{111} = 1$ и $k_{21} = 2$. Следовательно, собственные значения M являются объединением k_λ-кратных повторяющихся собственных значений для $k_\lambda \times k_\lambda$ матриц $2I - R_\lambda - S_\lambda$. В этом примере собственными значениями (0), (2) и двух $\left(\begin{smallmatrix} 2&0\\ -2&3 \end{smallmatrix}\right)$ являются $\{0\}$, $\{2\}$ и два $\{2,3\}$.

Собственные значения M непосредственно связаны с собственными значениямн матрицы A в упр. 2.3.4.2–19. Действительно, каждый собственный вектор A дает собственный вектор M, еслн приравнять компоненты для перестановок π и $\pi\rho\sigma^{-}$, поскольку строки π и $\pi\rho\sigma^{-}$ матриц $R + S$ равны. Например:

$$A = \begin{pmatrix} 2&-1&-1\\ -1&2&-1\\ -1&-1&2 \end{pmatrix} \text{ имеет собственные векторы } \begin{pmatrix} 1\\1\\1 \end{pmatrix}, \begin{pmatrix} 1\\-1\\0 \end{pmatrix}, \begin{pmatrix} 1\\0\\-1 \end{pmatrix} \text{ для собственных значений 0, 3, 3,}$$

что дает собственные векторы $(1,1,1,1,1,1)^T$, $(1,-1,0,0,-1,1)^T$, $(1,0,-1,-1,0,1)^T$ матрицы M для тех же собственных значений. Матрица M имеет $n!/2$ дополнительных собственных вектора со всеми нулевыми компоиентами, за исключением тех, которые индексированы π и $\pi\sigma^{-}\rho$ для некоторой π, поскольку только строки $\pi\rho^{-}$ и $\pi\sigma^{-}$ матрицы $R+S$ имеют ненулевые элементы в столбцах π и $\pi\sigma^{-}\rho$. Такие векторы дают $n!/2$ дополнительных собственных значений, равных 2.

Следовательно, значение U_n, т.е. произведение $2/n!$ и ненулевых собственных значений A, равно $2^{1-n!/2}/n!$ произведению ненулевых собственных значений M.

К сожалению, удивительная закономерность малых простых чисел-множителей прекращается: U_7 равно $2^{1217}3^{123}5^{119}7^{5}11^{28}43^{35}73^{20}79^{21}109^{35}$, а U_9 кратно 59229013196333^{168}.

По крайней мере один из этих циклов почти наверняка должен просто описываться и вычисляться, как это было для циклов де Бруийна в разделе 7.2.1.1. Однако ни одной простой конструкции пока не найдено.

ПРЕДМЕТНО-ИМЕННОЙ УКАЗАТЕЛЬ

Если какой-либо элемент предметно-именного указателя указывает на страницу с упражнением, обязательно просмотрите *ответ* к этому пражненню. Там может содержаться полезная информация по искомой вами теме. Страница ответа указывается здесь только в том случае, если в ответе есть ссылка на тему, не включениую в формулировку упражнения.

Научно-популярное издание

Дональд Эрвин Кнут

Искусство программирования
Том 4, Выпуск 2

Генерация всех кортежей и перестановок

Литературный редактор *И. А. Попова*
Верстка *А. Н. Полинчик*
Художественный редактор *С. А. Чернокозинский*
Корректор *Л. А. Гордиенко*

Издательский дом "Вильямс"
127055, г. Москва, ул. Лесная, д. 43, стр. 1.

Подписано в печать 27.08.2007. Формат 70×100/16.
Гарнитура LH. Печать офсетная.
Усл. печ. лист. 12,9. Уч.-изд. лист. 11,8.
Тираж 3000 экз. Заказ № 4013.

Отпечатано по технологии CtP
в ОАО "Печатный двор" им. А. М. Горького.
197110, Санкт-Петербург, Чкаловский пр., 15.

ASCII СИМВОЛЫ

	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#a	#b	#c	#d	#e	#f	
#2x		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	#2x
#3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	#3x
#4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	#4x
#5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	#5x
#6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	#6x
#7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~		#7x
	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#a	#b	#c	#d	#e	#f	

MMIX КОДЫ ОПЕРАЦИЙ (ОПКОДЫ)

	#0	#1	#2	#3	#4	#5	#6	#7	
#0x	TRAP 5υ	FCMP υ	FUN υ	FEQL υ	FADD 4υ	FIX 4υ	FSUB 4υ	FIXU 4υ	#0x
	FLOT[I] 4υ		FLOTU[I] 4υ		SFLOT[I] 4υ		SFLOTU[I] 4υ		
#1x	FMUL 4υ	FCMPE 4υ	FUNE υ	FEQLE 4υ	FDIV 40υ	FSQRT 40υ	FREM 4υ	FINT 4υ	#1x
	MUL[I] 10υ		MULU[I] 10υ		DIV[I] 60υ		DIVU[I] 60υ		
#2x	ADD[I] υ		ADDU[I] υ		SUB[I] υ		SUBU[I] υ		#2x
	2ADDU[I] υ		4ADDU[I] υ		8ADDU[I] υ		16ADDU[I] υ		
#3x	CMP[I] υ		CMPU[I] υ		NEG[I] υ		NEGU[I] υ		#3x
	SL[I] υ		SLU[I] υ		SR[I] υ		SRU[I] υ		
#4x	BN[B] υ+π		BZ[B] υ+π		BP[B] υ+π		BOD[B] υ+π		#4x
	BNN[B] υ+π		BNZ[B] υ+π		BNP[B] υ+π		BEV[B] υ+π		
#5x	PBN[B] 3υ−π		PBZ[B] 3υ−π		PBP[B] 3υ−π		PBOD[B] 3υ−π		#5x
	PBNN[B] 3υ−π		PBNZ[B] 3υ−π		PBNP[B] 3υ−π		PBEV[B] 3υ−π		
#6x	CSN[I] υ		CSZ[I] υ		CSP[I] υ		CSOD[I] υ		#6x
	CSNN[I] υ		CSNZ[I] υ		CSNP[I] υ		CSEV[I] υ		
#7x	ZSN[I] υ		ZSZ[I] υ		ZSP[I] υ		ZSOD[I] υ		#7x
	ZSNN[I] υ		ZSNZ[I] υ		ZSNP[I] υ		ZSEV[I] υ		
#8x	LDB[I] μ+υ		LDBU[I] μ+υ		LDW[I] μ+υ		LDWU[I] μ+υ		#8x
	LDT[I] μ+υ		LDTU[I] μ+υ		LDO[I] μ+υ		LDOU[I] μ+υ		
#9x	LDSF[I] μ+υ		LDHT[I] μ+υ		CSWAP[I] 2μ+2υ		LDUNC[I] μ+υ		#9x
	LDVTS[I] υ		PRELD[I] υ		PREGO[I] υ		GO[I] 3υ		
#Ax	STB[I] μ+υ		STBU[I] μ+υ		STW[I] μ+υ		STWU[I] μ+υ		#Ax
	STT[I] μ+υ		STTU[I] μ+υ		STO[I] μ+υ		STOU[I] μ+υ		
#Bx	STSF[I] μ+υ		STHT[I] μ+υ		STCO[I] μ+υ		STUNC[I] μ+υ		#Bx
	SYNCD[I] υ		PREST[I] υ		SYNCID[I] υ		PUSHGO[I] 3υ		
#Cx	OR[I] υ		ORN[I] υ		NOR[I] υ		XOR[I] υ		#Cx
	AND[I] υ		ANDN[I] υ		NAND[I] υ		NXOR[I] υ		
#Dx	BDIF[I] υ		WDIF[I] υ		TDIF[I] υ		ODIF[I] υ		#Dx
	MUX[I] υ		SADD[I] υ		MOR[I] υ		MXOR[I] υ		
#Ex	SETH υ	SETMH υ	SETML υ	SETL υ	INCH υ	INCMH υ	INCML υ	INCL υ	#Ex
	ORH υ	ORMH υ	ORML υ	ORL υ	ANDNH υ	ANDNMH υ	ANDNML υ	ANDNL υ	
#Fx	JMP[B] υ		PUSHJ[B] υ		GETA[B] υ		PUT[I] υ		#Fx
	POP 3υ	RESUME 5υ	[UN]SAVE 20μ+υ		SYNC υ	SWYM υ	GET υ	TRIP 5υ	
	#8	#9	#A	#B	#C	#D	#E	#F	

$\pi = 2\upsilon$, если условный переход выполняется; $\pi = 0$, если условный переход не выполняется.

Искусство программирования

ДОНАЛЬД Э. КНУТ

Эта многотомная работа по анализу алгоритмов давно считается исчерпывающим описанием классической информатики. Три завершенных тома, вышедших в свет к этому времени, представляют собой неоценимый источник информации для теории и практики программирования. Многочисленные читатели этого труда говорят о его огромном влиянии на них и их профессионализм. Ученых потрясает красота и элегантность анализа Кнута, программисты-практики используют эти книги как справочник для решения возникающих перед ними проблем. И все восхищены обширностью, ясностью, точностью и юмором *Искусства программирования*.

В настоящее время, работая над четвертым и последующими томами и над обновлением уже вышедших, Кнут создал серию небольших брошюр, называемых *выпусками*, которые планирует регулярно издавать. Каждый выпуск содержит один или несколько разделов с новым или обновленным материалом. В конечном счете материалы выпусков войдут в окончательные версии каждого тома, и начатая в 1962 году гигантская работа будет завершена.

Том 4, выпуск 2

Данный выпуск представляет собой продолжение главы о комбинаторных алгоритмах, которая будет включена в четвертый том *Искусства программирования*. Поскольку часть этого тома составит большая глава о комбинаторном поиске, то этот выпуск начинается с рассмотрения генерации всех возможных объектов. Особое внимание уделяется генерации всех n-кортежей, которые расширяют эти идеи для всех перестановок. Такие алгоритмы дают естественную мотивацию, с помощью которой вводятся и развиваются многие ключевые идеи комбинаторной математики. Кнут в этом и других выпусках тома 4 иллюстрирует важные теории, рассматривая связанные с ними игры и головоломки, и убеждает в том, что даже самое серьезное программирование может быть увлекательным.

Дональд Э. Кнут известен всему миру своей пионерской работой по алгоритмам и методам программирования, разработками издательских систем TEX и METAFONT, а также научными работами. Заслуженный профессор Искусства программирования Станфордского университета в настоящее время посвящает все свое время работе над завершением данных выпусков и всех семи томов своей великой работы.

ISBN 978-5-8459-1164-3

Издательский дом "ВИЛЬЯМС"
www.williamspublishing.com

ADDISON-WESLEY
Pearson Education

www.awprofessional.com