ГЛАВА 7

КОМБИНАТОРНЫЙ ПОИСК

...Чтобы их найти, надо искать весь день,
*а найдешь — увидишь, что и искать не стоило.**

— БАССАНИО (BASSANIO), в *Венецианский купец* (акт I, сцена 1)

Любые действия или противодействия каждой капли этого людского моря
могут сложиться в самые непредсказуемые и необъяснимые комбинации.
*Иногда происходят поразительные, невообразимые события...***

— ШЕРЛОК ХОЛМС (SHERLOCK HOLMES),
в *The Adventure of the Blue Carbuncle* (1892)

Тема комбинаторных алгоритмов слишком обширна,
чтобы охватить ее в одной статье или даже в одной книге.

— РОБЕРТ Э. ТАРЖАН (ROBERT E. TARJAN) (1976)

Постоянно сталкиваясь с массой людей, я был впечатлен
тем фактом, что наиболее успешными среди них были те,
кто обладал врожденным даром разгадывать головоломки.
Жизнь полна загадок, которые мы должны разгадывать по
мере того, как судьба подбрасывает их нам.

— УОЛТЕР ЛОЙД (СЭМ ЛОЙД-МЛ.) (SAM LOYD, JR.) (1926)

Комбинаторика изучает способы, которыми дискретные объекты могут быть упорядочены в шаблоны различного вида. Например, объекты могут представлять собой $2n$ чисел $\{1, 1, 2, 2, \ldots, n, n\}$, и мы хотим разместить их в строку таким образом, чтобы между двумя появлениями каждого числа k находилось ровно k других чисел. Когда $n = 3$, имеется по сути единственный способ размещения таких "пар Лэнгфорда", а именно 231213 (и его зеркальное отображение 312132); аналогично имеется единственное решение и для $n = 4$. Многие другие типы комбинаторных шаблонов будут рассматриваться далее в этой книге.

Обычно при изучении комбинаторных задач возникает пять основных типов вопросов, одни — потруднее, другие — попроще.

i) *Существование*: существует ли упорядочение X, отвечающее данному шаблону?
ii) *Построение*: если существует, то можно ли быстро найти такое X?

* Перевод Т. Щепкиной-Куперник.

** Перевод В. Михалюка.

iii) *Подсчет*: сколько имеется различных упорядочений X?
iv) *Генерация*: могут ли все упорядочения X_1, X_2, ... быть посещены систематическим образом?
v) *Оптимизация*: какие упорядочения максимизируют или минимизируют $f(X)$ для заданной целевой функции f?

Каждый из приведенных вопросов оказывается достаточно интересным по отношению к парам Лэнгфорда.

Например, рассмотрим вопрос существования. Методом проб и ошибок достаточно быстро выясняется, что, когда $n = 5$, разместить $\{1, 1, 2, 2, \ldots, 5, 5\}$ требуемым образом в десяти позициях невозможно. Обе единицы должны находиться либо в четных позициях, либо в нечетных. Точно так же и тройки и пятерки должны находиться парами либо в четных, либо в нечетных позициях; двойки и четверки используют по одной четной и одной нечетной позиции. Таким образом, мы не можем заполнить по пяти позиций каждой четности. Такие же рассуждения доказывают, что задача не имеет решения и при $n = 6$ или, в общем случае, когда количество нечетных значений в множестве $\{1, 2, \ldots, n\}$ нечетно.

Другими словами, решение задачи Лэнгфорда может существовать только для $n = 4m - 1$ или $n = 4m$, для некоторого целого m. И обратно, когда n имеет указанный вид, существует элегантный способ построения требуемого размещения, найденный Роем О. Дэйвисом (Roy O. Davies) и приведенный в упр. 1.

Сколько же имеется существенно разных решений задачи L_n? С ростом n — много, очень много:

$$\begin{aligned} L_3 &= 1; & L_4 &= 1; \\ L_7 &= 26; & L_8 &= 150; \\ L_{11} &= 17\,792; & L_{12} &= 108\,144; \\ L_{15} &= 39\,809\,640; & L_{16} &= 326\,721\,800; \\ L_{19} &= 256\,814\,891\,280; & L_{20} &= 2\,636\,337\,861\,200; \\ L_{23} &= 3\,799\,455\,942\,515\,488; & L_{24} &= 46\,845\,158\,056\,515\,936. \end{aligned} \tag{1}$$

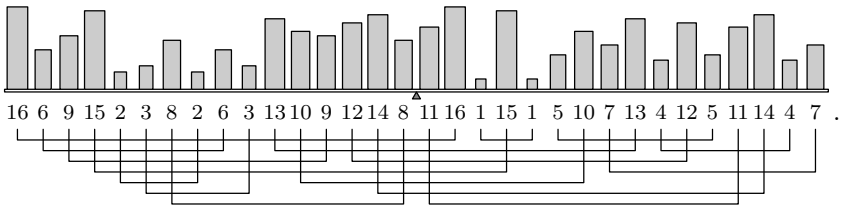
[Значения L_{23} и L_{24} были найдены в 2004 и 2005 годах М. Краецки (M. Krajecki), К. Джейллетом (C. Jaillet) и А. Бью (A. Bui); см. *Studia Informatica Universalis* **4** (2005), 151–190.] Наброски "на коленке" дают грубую оценку значения L_n (когда оно не равно нулю), равную $(4n/e^3)^{n+1/2}$ (см. упр. 5); и эта оценка оказывается в целом корректной для всех известных значений. Но простой формулы для этих значений, похоже, не существует.

Задача Лэнгфорда представляет собой простой частный случай общего класса комбинаторных задач, называющихся *задачами точного покрытия*. В разделе 7.2.2.1 мы познакомимся с так называемым алгоритмом "танцующих связей", который представляет собой удобное средство генерации всех решений таких задач. Например, при $n = 16$ этот метод требует выполнения всего лишь около 3200 обращений к памяти для каждого найденного решения задачи Лэнгфорда. Таким образом, значение L_{16} может быть вычислено за разумное время путем простой генерации всех решений и их подсчета.

Заметим, однако, что число L_{24} *огромно* — огрубленно оно равно 5×10^{16}, или около 1500 MIP-лет. (Вспомним, что "MIP-год" представляет собой количество

инструкций, которые за год может выполнить машина, выполняющая миллион инструкций в секунду, а именно 31 556 952 000 000.) Следовательно, очевидно, что точное значение L_{24} было определено с помощью иной методики, *не* включающей генерацию всех расстановок. Действительно, имеется гораздо, гораздо более быстрый способ вычисления L_n с помощью полиномиальной алгебры. Поучительный метод, описанный в упр. 6, требует $O(4^n n)$ операций, что может показаться неэффективным; но это лучше метода генерации и подсчета на громадный множитель порядка $\Theta((n/e^3)^{n-1/2})$, так что даже при $n = 16$ этот метод работает примерно в 20 раз быстрее. С другой стороны, точное значение L_{100}, вероятно, никогда не станет известным, несмотря на все бóльшую и бóльшую скорость компьютеров.

Можно также рассмотреть решения задачи Лэнгфорда, *оптимальные* в том или ином отношении. Для примера можно расположить шестнадцать пар разновесов $\{1, 1, 2, 2, \ldots, 16, 16\}$ так, чтобы они удовлетворяли условиям Лэнгфорда и обладали дополнительным свойством "сбалансированности", в том смысле, что, будучи расставленными в соответствующем порядке, они оставят в равновесии рычажные весы:



(2)

Другими словами, $15.5 \cdot 16 + 14.5 \cdot 6 + \cdots + 0.5 \cdot 8 = 0.5 \cdot 11 + \cdots + 14.5 \cdot 4 + 15.5 \cdot 7$; и в данном конкретном примере мы сталкиваемся с еще одним условием равновесия, $16 + 6 + \cdots + 8 = 11 + 16 + \cdots + 7$. Следовательно, справедливо также $16 \cdot 16 + 15 \cdot 6 + \cdots + 1 \cdot 8 = 1 \cdot 11 + \cdots + 15 \cdot 4 + 16 \cdot 7$.

Кроме того, размещение (2) имеет *минимальную ширину* среди всех решений задачи Лэнгфорда порядка 16: соединяющие линии под диаграммой показывают, что в любой точке диаграммы одновременно незавершенными оказываются не более семи пар; можно также показать, что ширина, равная шести, в данном случае недостижима (см. упр. 7).

Какое размещение $a_1 a_2 \ldots a_{32}$ чисел $\{1, 1, \ldots, 16, 16\}$ оказывается *наименее* сбалансированным, в том смысле, что сумма $\sum_{k=1}^{32} k a_k$ максимальна? Оказывается, что максимально возможное значение равно 5268. Одним из таких решений (а всего их 12 016) является

$$2\ 3\ 4\ 2\ 1\ 3\ 1\ 4\ 16\ 13\ 15\ 5\ 14\ 7\ 9\ 6\ 11\ 5\ 12\ 10\ 8\ 7\ 6\ 13\ 9\ 16\ 15\ 14\ 11\ 8\ 10\ 12. \tag{3}$$

Более интересный вопрос — о решениях задачи Лэнгфорда, являющихся минимальными и максимальными в лексикографическом порядке. Ответами для $n = 24$ являются

$$\{\texttt{abacbdecfgdoersfpgqtuwxvjklonhmirpsjqkhltiunmwvx},\ \texttt{xvwsquntkigrdapaodgiknqsvxwutmrpohljcfbecbhmfejl}\}, \tag{4}$$

если использовать буквы `a`, `b`, ..., `w`, `x` вместо чисел 1, 2, ..., 23, 24.

Мы рассмотрим множество методов комбинаторной оптимизации в следующих разделах этой главы. Наша цель, конечно же, заключается в том, чтобы решать

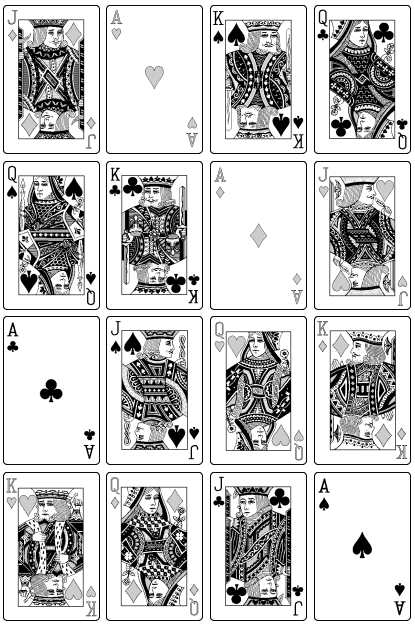


Рис. 1. Беспорядок в карточном королевстве: никакого совпадения строк. (Это всего лишь одно из множества решений популярной головоломки XVIII века.)

такие задачи с помощью изучения только небольшой части пространства всех возможных размещений.

Ортогональные латинские квадраты. Давайте ненадолго вернемся в ранние дни комбинаторики. Посмертное издание книги Жака Озанама (Jacques Ozanam) *Recreations mathematiques et physiques* (Paris: 1725) включает занимательную головоломку на с. 434 тома 4: "Возьмите всех тузов, королей, дам и валетов из обычной колоды игральных карт и расположите их в виде квадрата так, чтобы в каждой строке и каждом столбце содержались карты всех мастей и всех достоинств". Сумеете ли вы сделать это? Решение Озанама, показанное на рис. 1, делает даже больше: полный набор достоинств и мастей встречается также на обеих диагоналях квадрата.

В 1779 году подобная головоломка, распространившаяся в Санкт-Петербурге, привлекла внимание великого математика Леонарда Эйлера. "Тридцать шесть офицеров шести различных званий, взятые из шести различных полков, хотят выстроиться квадратным строем 6×6 так, чтобы в каждой строке и каждом столбце этого квадрата имелись офицеры всех званий и полков. Как они должны это сделать?" Никто не мог найти удовлетворительного решения этой задачи, так что Эйлер решил взяться за эту проблему (несмотря на то, что в 1771 году он практически ослеп и диктовал все свои работы ассистентам). Он написал большую работу на эту тему [в конце концов опубликованную в *Verhandelingen uitgegeven door het Zeeuwsch Genootschap der Wetenschappen te Vlissingen* **9** (1782), 85–239], в которой привел решения подобной задачи для n званий и n полков для $n = 1$, 3, 4, 5, 7, 8, 9, 11, 12, 13, 15, 16, ...; ему не поддались только случаи $n \bmod 4 = 2$.

Очевидно, что при $n = 2$ решения не существует. Но Эйлер был озадачен случаем $n = 6$, когда рассмотрел "очень значительное число" квадратных расста-

новок, которые не работали. Он показал, что каждое решение должно приводить ко многим другим решениям, выглядящим иными, и не мог поверить, что все такие решения сумели избежать его внимания. Поэтому он говорил: "Я не сомневаюсь в своем выводе о том, что найти полный квадрат из 36 ячеек невозможно, точно так же, как невозможно найти решения для $n = 10$, $n = 14$ ... в общем случае для всех нечетно-четных чисел".

Эйлер дал 36 офицерам имена $a\alpha$, $a\beta$, $a\gamma$, $a\delta$, $a\epsilon$, $a\zeta$, $b\alpha$, $b\beta$, $b\gamma$, $b\delta$, $b\epsilon$, $b\zeta$, $c\alpha$, $c\beta$, $c\gamma$, $c\delta$, $c\epsilon$, $c\zeta$, $d\alpha$, $d\beta$, $d\gamma$, $d\delta$, $d\epsilon$, $d\zeta$, $e\alpha$, $e\beta$, $e\gamma$, $e\delta$, $e\epsilon$, $e\zeta$, $f\alpha$, $f\beta$, $f\gamma$, $f\delta$, $f\epsilon$, $f\zeta$, основываясь на их полках и званиях. Он заметил, что любое решение состоит из двух *отдельных* квадратов, одного — для латинских букв и второго — для греческих. Каждый из этих квадратов, как предполагается, содержит разные записи в строках и столбцах; так что Эйлер начал свое исследование с изучения возможных конфигураций для $\{a, b, c, d, e, f\}$, которые он назвал *латинскими квадратами*. Латинский квадрат может быть объединен в пару с греческим квадратом, образуя "греко-латинский квадрат" только в том случае, когда квадраты *ортогональны* друг к другу, что означает, что пары букв (латинская, греческая) не могут встретиться более одного раза при наложении квадратов. Например, если положить $a = \texttt{A}$, $b = \texttt{K}$, $c = \texttt{Q}$, $d = \texttt{J}$, $\alpha = \clubsuit$, $\beta = \spadesuit$, $\gamma = \diamondsuit$ и $\delta = \heartsuit$, то рис. 1 эквивалентен латинскому, греческому и греко-латинскому квадратам

$$\begin{pmatrix} d & a & b & c \\ c & b & a & d \\ a & d & c & b \\ b & c & d & a \end{pmatrix}, \quad \begin{pmatrix} \gamma & \delta & \beta & \alpha \\ \beta & \alpha & \gamma & \delta \\ \alpha & \beta & \delta & \gamma \\ \delta & \gamma & \alpha & \beta \end{pmatrix} \quad \text{и} \quad \begin{pmatrix} d\gamma & a\delta & b\beta & c\alpha \\ c\beta & b\alpha & a\gamma & d\delta \\ a\alpha & d\beta & c\delta & b\gamma \\ b\delta & c\gamma & d\alpha & a\beta \end{pmatrix}. \tag{5}$$

Конечно, можно воспользоваться *любыми* n различными символами в латинском квадрате размером $n \times n$; все, что имеет значение, — это чтобы ни один символ не встречался дважды в одной строке или в одном столбце. Так что мы вполне можем использовать для записей числовые значения $\{0, 1, \ldots, n-1\}$. Кроме того, мы будем говорить просто о латинских квадратах, не подразделяя их на греческие или латинские, поскольку ортогональность — отношение симметричное.

Утверждение Эйлера о том, что два латинских квадрата размером 6×6 не могут быть ортогональны, было подтверждено Томасом Клаузеном (Thomas Clausen), который свел задачу к проверке 17 фундаментально различных случаев. Об этом в письме, датированном 10 августа 1842 года, К. Ф. Гауссу (C. F. Gauss) сообщил Г. К. Шумахер (H. C. Schumacher). Сам Клаузен свой анализ не публиковал. Первая появившаяся в печати демонстрация этого факта принадлежит Г. Тарри (G. Tarry) [*Comptes rendus, Association francaise pour l'avancement des sciences* **29**, part 2 (1901), 170–203], который, идя собственным путем, открыл, что латинские квадраты размером 6×6 могут быть разделены на 17 семейств. (В разделе 7.2.3 мы рассмотрим, как разложить задачу на комбинаторно неэквивалентные классы размещений.)

Гипотеза Эйлера о значениях $n = 10$, $n = 14$, ... была "доказана" трижды: Ю. Петерсеном (J. Petersen) [*Annuaire des mathématiciens* (Paris: 1902), 413–427], П. Вернике (P. Wernicke) [*Jahresbericht der Deutschen Math.-Vereinigung* **19** (1910), 264–267] и Г. Ф. Мак-Нейшем (H. F. MacNeish) [*Annals of Math.* (2) **23** (1922), 221–227]. Однако во всех трех доказательствах известны недостатки; так что вопрос оставался открытым до тех пор, пока много лет спустя не стали доступны быст-

родействующие компьютеры. Одной из первых комбинаторных задач, решенных с применением ЭВМ, стала загадка греко-латинского квадрата размером 10×10: существует он или нет?

В 1957 году Л. Д. Пейдж (L. J. Paige) и Ч. Б. Томпкинс (C. B. Tompkins) запрограммировали ЭВМ SWAC для поиска контрпримера для предсказания Эйлера. Они выбирали один отдельный латинский квадрат 10×10 "почти случайно", и их программа пыталась найти другой квадрат, ортогональный первому. Но результат оказался обескураживающим, и они решили прекратить вычисления через пять часов работы. За это время машина сгенерировала достаточно данных, чтобы можно было предсказать, что вся работа потребует как минимум 4.8×10^{11} часов машинного времени!

Вскоре после этого трем математикам удалось совершить прорыв, который вынес латинские квадраты на страницы одной из ведущих мировых газет (New York Times от 29 апреля 1959 года): Р. Ч. Бозе (R. C. Bose), Ш. Ш. Шрикханде (S. S. Shrikhande) и Э. Т. Паркер (E. T. Parker) нашли замечательный ряд построений, которые дают ортогональные квадраты размером $n \times n$ для всех $n > 6$ [*Proc. Nat. Acad. Sci.* **45** (1959), 734–737, 859–862; *Canadian J. Math.* **12** (1960), 189–203]. Таким образом, после сопротивления атакам в течение 180 лет гипотеза Эйлера оказалась почти полностью ошибочной.

Их открытие было сделано без привлечения компьютеров. Но Паркер работал на UNIVAC, и его навыки программиста помогли получить решение задачи Пейджа и Томпкинса менее чем за час машинного времени ЭВМ UNIVAC 1206 Military Computer. [См. *Proc. Symp. Applied Math.* **10** (1960), 71–83; **15** (1963), 73–81.]

Давайте взглянем, как работали программисты тех времен и как Паркер превзошел их. Пейдж и Томпкинс начали со следующего квадрата L размером 10×10 и его пока неизвестного ортогонального напарника M:

$$L = \begin{pmatrix} 0&1&2&3&4&5&6&7&8&9 \\ 1&8&3&2&5&4&7&6&9&0 \\ 2&9&5&6&3&0&8&4&7&1 \\ 3&7&0&9&8&6&1&5&2&4 \\ 4&6&7&5&2&9&0&8&1&3 \\ 5&0&9&4&7&8&3&1&6&2 \\ 6&5&4&7&1&3&2&9&0&8 \\ 7&4&1&8&0&2&9&3&5&6 \\ 8&3&6&0&9&1&5&2&4&7 \\ 9&2&8&1&6&7&4&0&3&5 \end{pmatrix} \quad \text{и} \quad M = \begin{pmatrix} 0&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 1&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 2&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 3&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 4&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 5&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 6&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 7&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 8&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 9&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \end{pmatrix}. \tag{6}$$

Без потери общности можно считать, что строки M начинаются, как показано, с 0, 1, ..., 9. Задача заключается в заполнении 90 остающихся пустыми записей, и исходная программа для SWAC работала сверху вниз и слева направо. Верхняя левая пустая запись $\sqcup$ не может содержать 0, поскольку 0 уже имеется в верхней левой строке M. Это не может быть и 1, так как пара $(1, 1)$ уже имеется слева в следующей строке квадрата (L, M). Однако мы можем умозрительно вставить в это место 2. Следующей можно вставить цифру 1; и так постепенно мы найдем лексикографически наименьшую верхнюю строку, которая может быть в M, а именно 0214365897. Аналогично наименьшими строками, которые могут размещаться под 0214365897, являются 1023456789 и 2108537946; очередная наименьшая строка имеет вид 3540619278. К сожалению, после этого мы попадаем в переплет:

нет никакой возможности завершить очередную строку так, чтобы не вступить в конфликт с уже имеющимися. Поэтому заменим 3540619278 на 3540629178 (но и этот выбор не дает возможности продолжать работу), затем на 3540698172, и так еще несколько шагов, пока наконец не достигнем строки 3546109278, за которой может следовать строка 4397028651, после которой мы снова встретимся с неприятностями.

В разделе 7.2.3 мы изучим методы оценки поведения таких поисков без реального их проведения. Такие оценки говорят нам о том, что в данном случае метод Пейджа–Томпкинса, по сути, обходит неявное дерево поиска, содержащее около 2.5×10^{18} узлов. Большинство этих узлов принадлежит только нескольким уровням дерева; больше половины из них работают с выбором правой части шестой строки M, после того как около 50 из 90 пустых мест заполнены. Типичный узел дерева поиска требует для проверки, вероятно, около 75 обращений к памяти. Следовательно, общее время работы на современном компьютере можно оценить как время, необходимое для выполнения 2×10^{20} обращений к памяти.

Паркер же вернулся к методу, который Эйлер использовал для поиска ортогональных пар в 1779 году. Сначала он находил все так называемые *секущие* (*transversals*) матрицы L, а именно все способы выбора некоторых из ее элементов так, чтобы в каждой строке и каждом столбце имелся ровно один элемент, причем чтобы были использованы все возможные значения элементов. Например, одним из сечений в записи Эйлера является 0859734216, т. е. в столбце 0 мы выбираем 0, в столбце 1 — 8, ..., в столбце 9 — 6. Каждая секущая, включающая k в крайнем слева столбце L, представляет корректный способ размещения десяти k в квадрате M. Задача поиска секущих в действительности достаточно проста. Для данной матрицы L, оказывается, их имеется ровно 808; для $k = (0, 1, \ldots, 9)$ имеется соответственно $(79, 96, 76, 87, 70, 84, 83, 75, 95, 63)$ секущих.

После того как нам становятся известными секущие, остается решить задачу точного покрытия из десяти этапов, что существенно проще, чем исходная 90-этапная задача в (6). Все, что нам нужно, — это покрыть квадрат десятью непересекающимися секущими, поскольку каждое такое множество из десяти секущих эквивалентно латинскому квадрату M, ортогональному L.

Конкретный квадрат L из (6) в действительности имеет ровно одну ортогональную пару:

$$\begin{pmatrix} 0&1&2&3&4&5&6&7&8&9\\ 1&8&3&2&5&4&7&6&9&0\\ 2&9&5&6&3&0&8&4&7&1\\ 3&7&0&9&8&6&1&5&2&4\\ 4&6&7&5&2&9&0&8&1&3\\ 5&0&9&4&7&8&3&1&6&2\\ 6&5&4&7&1&3&2&9&0&8\\ 7&4&1&8&0&2&9&3&5&6\\ 8&3&6&0&9&1&5&2&4&7\\ 9&2&8&1&6&7&4&0&3&5 \end{pmatrix} \perp \begin{pmatrix} 0&2&8&5&9&4&7&3&6&1\\ 1&7&4&9&3&6&5&0&2&8\\ 2&5&6&4&8&7&0&1&9&3\\ 3&6&9&0&4&5&8&2&1&7\\ 4&8&1&7&5&3&6&9&0&2\\ 5&1&7&8&0&2&9&4&3&6\\ 6&9&0&2&7&1&3&8&4&5\\ 7&3&5&1&2&0&4&6&8&9\\ 8&0&2&3&6&9&1&7&5&4\\ 9&4&3&6&1&8&2&5&7&0 \end{pmatrix}. \tag{7}$$

Алгоритм танцующих связей находит это решение и доказывает его единственность всего лишь примерно за 1.7×10^8 обращений к памяти для данных 808 секущих. Стоимость же фазы поиска секущих (около пяти миллионов обращений к памяти) пренебрежимо мала по сравнению с поиском матрицы. Таким образом, исходное

время работы, равное 2×10^{20} обращениям к памяти и считавшееся неизбежной платой за решение задачи с 10^{90} возможными способами заполнения пустых слотов, оказалось уменьшено более чем в 10^{12}(!) раз.

Позже мы увидим, что можно внести усовершенствования и в методы решения 90-уровневой задачи наподобие (6). Действительно, оказывается, (6) непосредственно представимо в качестве задачи точного покрытия (см. упр. 17), которая решается процедурой танцующих связей из раздела 7.2.2.1 ценой всего лишь 1.3×10^{11} обращений к памяти. Но даже в этом случае подход Эйлера–Паркера оказывается в тысячу раз лучше подхода Пейджа–Томпкинса. "Разложив" задачу на две отдельные фазы, одну для поиска секущих и вторую — для их объединения, Эйлер и Паркер, по сути, уменьшили стоимость вычислений с произведения $T_1 T_2$ до суммы $T_1 + T_2$.

Мораль этой истории очевидна: комбинаторные задачи могут поставить нас перед лицом огромной совокупности возможностей, но мы не должны сразу же опускать руки. Одна хорошая идея может на много порядков сократить количество требующихся вычислений.

Головоломки и реальный мир. Многие из комбинаторных задач, которые мы будем рассматривать в этой главе, наподобие задач Лэнгфорда о парах или Озанама о шестнадцати картах, изначально представляли собой не более чем головоломки. Некоторых особо серьезно настроенных читателей такая легкомысленность некоторых задач может попросту отпугнуть. В самом деле, сто́ит ли тратить попусту машинное время, неужели ему нет более полезного применения? И не должна ли книга, посвященная компьютерам и программированию, в первую очередь озаботиться важными промышленными и научными приложениями, определяющими мировой прогресс?

Начнем с того, что автор данной книги совершенно не озадачивался ее полезностью для прогресса человечества. Но он твердо убежден, что такая книга, как эта, должна обратить особое внимание на *методы* решения задач, а также на математические идеи и *модели*, которые помогут решать многие различные проблемы, а не на причины, по которым такие методы и модели могут быть полезными. Мы изучим множество красивых и эффективных способов решения комбинаторных задач, и главной причиной их изучения будет элегантность этих методов. Комбинаторные задачи встречаются повсюду, и ежедневно появляются новые способы применения описываемых в этой главе методик. Так что давайте не будем заранее ограничивать сами себя, пытаясь заранее классифицировать, для чего годятся те или иные идеи.

Например, оказывается, что ортогональные латинские квадраты чрезвычайно полезны, в особенности при планировании экспериментов. Уже в 1788 году Франсуа Кретте де Паллюль (Francois Cretté de Palluel) использовал латинский квадрат размером 4×4 для изучения, что будет с шестнадцатью овцами четырех различных пород при применении четырех различных кормов и четырех разных периодов стрижки. [*Mémoires d'Agriculture* (Paris: Société Royale d'Agriculture, trimestre d'été, 1788), 17–23.] Латинский квадрат позволил ему ограничиться 16 овцами вместо 64; при использовании греко-латинского квадрата он мог бы варьировать еще один параметр, например количество корма или вид пастбища.

Если бы мы сосредоточились на его воззрениях на животноводство, то могли бы увязнуть в деталях разведения овец, сравнении корнеплодов с зерновыми

и расходами на их выращивание и т. д. Читатели, не интересующиеся сельским хозяйством, могли бы пропустить весь материал, несмотря на то, что латинские квадраты применимы к широкому спектру исследований (представьте, например, проверку действия лекарства на пациентов на пяти стадиях некоторого заболевания, пяти возрастных и пяти весовых группах). Кроме того, концентрация на опытно-исследовательском применении латинских квадратов может привести к тому, что читатели упустят тот факт, что латинские квадраты имеют важные приложения в дискретной геометрии и кодах с исправлением ошибок (см. упр. 18–24).

Даже задача Лэнгфорда, на первый взгляд кажущаяся чистым развлечением, также имеет практическую ценность. Т. Сколем (T. Skolem) использовал последовательности Лэнгфорда для построения Штейнеровской системы троек, которую мы применяли для запросов к базам данных в разделе 6.5 [см. *Math. Scandinavica* **6** (1958), 273–280]; а в 1960-х годах Э. Д. Грот (E. J. Groth) из Motorola Corporation применил пары Лэнгфорда в разработке электронных схем для умножения. Кроме того, алгоритмы, которые эффективно решают задачу Лэнгфорда и секущих латинских квадратов, такие как метод танцующих связей, применимы к решению задач точного покрытия в общем случае; а задача точного покрытия имеет большое значение для таких важнейших проблем, как, например, справедливое распределение участков для избирательных округов и т. п.

Ни приложения, ни головоломки не являются наиболее важными вещами. Наша главная цель — внедрить в свои головы базовые концепции наподобие понятий латинских квадратов или точного покрытия. Эти концепции обеспечат нас строительными блоками, словарем и пониманием, необходимыми для решения *завтрашних* задач.

Глупо обсуждать решение задач без практического решения каких бы то ни было задач. Для стимуляции нашего мышления и воображения, упорядочения серых клеточек и знакомства с базовыми концепциями нужны подходящие задачи. И для этой цели часто лучше всего подходят именно головоломки, поскольку их можно изложить буквально несколькими словами и они не требуют сложных базовых знаний.

Жизнь — слишком сложная вещь, чтобы относиться к ней, как к очередной головоломке, которую следует решить и которую можно решить, просто снабдив машину соответствующими инструкциями и данными. Но, к счастью, есть загадки, которые *могут* быть решены! Головоломки сто́ят того, чтобы причислить их к наибольшим удовольствиям жизни, которыми, как и любыми другими удовольствиями, следует пользоваться умеренно и без злоупотреблений.

Конечно, Лэнгфорд и Озанам предназначали свои головоломки для людей, а не для компьютеров. Разве смысл не будет потерян, если мы просто передадим задачу машинам, которые будут решать ее методом грубой силы, а не с помощью искусного применения рационального мышления? Джордж Брюстер (George Brewster) в письме Мартину Гарднеру (Martin Gardner) в 1963 году выразил широко распространенное мнение следующим образом: “Скармливать головоломки для отдыха компьютерам — немногим лучше, чем глушить форель в горной реке динамитом. Оставьте человеку возможность расслабиться”.

Хотя с этим трудно не согласиться, указанная точка зрения упускает один важный момент: простые головоломки часто имеют обобщения, которые выходят за

рамки человеческих возможностей и вызывают наше любопытство. Изучение этих обобщений часто предполагает наличие поучительных методов, которые могут применяться к ряду других проблем и иметь неожиданные последствия. Действительно, многие из ключевых методов, которые мы будем изучать, родились в попытках решения различных головоломок. При написании этой главы автор не мог не наслаждаться тем фактом, что теперь, когда компьютеры становятся все более и более быстрыми, головоломки доставляют еще большее удовольствие, ведь теперь в наших руках оказывается еще более мощный динамит, с которым можно вволю поиграться. [Более подробно точка зрения автора изложена в его эссе "Полезны ли задачи-головоломки?", написанном еще в 1976 году; см. *Selected Papers on Computer Science* (1996), 169–183.]

Опасность головоломок в том, что они могут оказаться *слишком* элегантными. Хорошие головоломки, как правило, математически чисты и хорошо структурированы, но нам надо научиться систематически работать с тем грязным, хаотическим материалом, который ежедневно нас окружает. Фактически некоторые вычислительные методы важны, в первую очередь, потому, что они предоставляют мощные способы преодоления этих сложностей. Вот почему, например, в начале главы 5 рассказывалось о загадочных правилах упорядочения библиотечных каталогов, а в разделе 2.2.5 моделировалась работа реального лифта.

Чтобы в экспериментах с комбинаторными алгоритмами можно было воспользоваться данными из реального мира, подготовлена коллекция программ и данных под названием Stanford GraphBase (SGB). Она включает, например, данные об американских автострадах и модель "затраты–выпуск" американской экономики; в ней описано распределение персонажей в *Илиаде* Гомера, *Анне Карениной* Льва Толстого и ряде других романов; здесь можно найти структуру *Thesaurus* 1879 года Роджета (Roget) и результаты сотен футбольных матчей разных колледжей; в ней есть даже значения пикселей черно-белого варианта *Джоконды* (Моны Лизы) Леонардо да Винчи. Но, пожалуй, самое важное то, что SGB содержит набор пятибуквенных английских слов, которые мы и рассмотрим следующими.

Пятибуквенные английские слова. Множество примеров в этой главе будут основываться на списке пятибуквенных слов

`aargh`, `abaca`, `abaci`, `aback`, `abaft`, `abase`, `abash`, ..., `zooms`, `zowie`. (8)

(Всего в нем содержится 5757 слов — слишком много, чтобы перечислить здесь их все; но вы легко можете представить себе отсутствующие в (8) слова.) Это список, лично составленный автором книги между 1972 и 1992 годами, начиная с того времени, когда он понял, что такие слова представляют собой идеальные (`ideal`) данные для тестирования комбинаторных алгоритмов различных видов.

Этот список преднамеренно ограничен только истинными словами английского языка, в том смысле, что автор встречался с их реальным применением. Полные словари содержат тысячи слов, известных лишь особо посвященным, такие как `aalii`, `abamp`, ..., `zymin` и `zyxst`; но такие слова нужны, в первую очередь, игрокам в Scrabble®*. Однако применение таких слов уменьшает удовольствие тех, кто

* Скрэббл (от англ. *Scrabble — рыться в поисках чего-либо*) — настольная игра, в которую могут играть от 2 до 4 человек, выкладывая слова из имеющихся у них букв в поле размером 15×15. В русскоязычной среде известна под названием *Эрудит*. — *Примеч. пер.*

с ними не знаком. Поэтому в течение двадцати лет автор систематически записывал все слова, которые казались ему подходящими для дидактических целей *Искусства программирования*.

Наконец, эту коллекцию стало просто необходимо "заморозить" и прекратить пополнять хотя бы для того, чтобы иметь возможность получать воспроизводимые результаты экспериментов. Английский язык продолжает эволюционировать, но 5757 слов всегда останутся неизменными — несмотря на то, что временами автор порывался добавить к списку слова, с которыми он не был знаком в 1992 году, такие как `chads`, `stent`, `blogs`, `ditzy`, `phish`, `bling` и, возможно, `tetch`. Но это не было сделано — время для внесения изменений в SGB закончилось.

> *Этот словарь предназначен для хранения всех хорошо известных английских слов... которые можно использовать в приличном обществе и которые могут служить в качестве звеньев.*
>
> — ЛЬЮИС КЭРРОЛЛ (LEWIS CARROLL), *Doublets: A Word-Puzzle* (1879)

Собственные имена наподобие `Knuth` не рассматривались как корректные пятибуквенные слова. Однако, например, такое слово, как `gauss`, корректно, поскольку "gauss" представляет собой единицу измерения магнитной индукции. Слова в SGB состоят только из обычных строчных букв; в списке нет слов с переносами, сокращений или слов, требующих ударения, наподобие `blasé`. Таким образом, каждое слово можно рассматривать и как вектор с пятью компонентами в диапазоне $[0\,..\,26)$. В векторном смысле слова `yucca` и `abuzz` отстоят друг от друга дальше всех: евклидово расстояние между ними равно

$$\|(24,20,2,2,0)-(0,1,20,25,25)\|_2=\sqrt{24^2+19^2+18^2+23^2+25^2}=\sqrt{2415}.$$

Полностью Stanford GraphBase со всеми программами и данными можно загрузить с веб-сайта автора `http://www-cs-faculty.stanford.edu/~knuth/sgb.html`. Получить список всех слов SGB еще проще, так как он находится по тому же адресу, но в файле '`sgb-words.txt`'. Этот файл содержит 5757 строк, в каждой из которых имеется одно слово, начиная с '`which`' и заканчивая '`pupal`'. Слова расположены в порядке, соответствующем их частоте употребления; например, словами с рангами 1000, 2000, 3000, 4000 и 5000 являются соответственно `ditch`, `galls`, `visas`, `faker` и `pismo`. Запись '`WORDS`(n)' в этой главе будет использоваться для обозначения n наиболее часто встречающихся слов, в соответствии с их рангами.

Кстати, пятибуквенные слова часто включают множественное число *четырехбуквенных слов*, что, конечно, недопустимо при строгом подходе к делу, и такие слова не встретятся вам в *The Official Scrabble® Players Dictionary*, но для словаря SGB они вполне годятся. Один из способов гарантировать, что семантически неподходящие термины не попадут в статью, основанную на списке слов SGB, — ограничить рассмотрение множеством `WORDS`(n), где n равно, скажем, 3000.

Упр. 26–37 могут использоваться в качестве разминки для начального изучения слов SGB, с которыми мы встретимся в этой главе в самых разных комбинаторных контекстах. Например, пока мы еще недалеко ушли от задач покрытия, мы можем заметить, что четыре слова '`third flock began jumps`' охватывают 20 из первой 21 буквы алфавита. Пять же слов могут охватить не более 24 различных букв, как

в случае {becks, fjord, glitz, nymph, squaw}, если только мы не обратимся к редкому, отсутствующему в SGB слову наподобие waqfs (пожертвование для мусульманских религиозных или благотворительных целей), которое можно скомбинировать со словами {gyved, bronx, chimp, klutz} для охвата 25 букв.

Простых слов из WORDS(400) достаточно для образования *квадрата из слов*:

```
class
light
agree .                                   (9)
sheep
steps
```

Однако для получения *куба из слов* требуется добраться почти до WORDS(3000):

```
types  yeast  pasta  ester  start
yeast  earth  armor  stove  three
pasta  armor  smoke  token  arena .       (10)
ester  stove  token  event  rents
start  three  arena  rents  tease
```

Здесь каждый "срез" 5×5 представляет собой квадрат из слов. С помощью простого расширения базового алгоритма танцующих связей (см. раздел 7.2.2.2) можно показать ценой около 390 миллиардов обращений к памяти, что WORDS(3000) дает возможность получить только три симметричных куба из слов, таких как (10); в упр. 36 вы найдете оставшиеся два. Интересно, что из полного множества WORDS(5757) можно создать 83 576 симметричных кубов.

Графы из слов. Конечно, интересно и важно помещать объекты в строки, квадраты, кубы и прочие фигуры; но в практических приложениях имеется еще *более* интересная и важная комбинаторная структура, а именно *граф*. Вспомним из раздела 2.3.4.1, что граф представляет собой множество точек, именуемых *вершинами*, вместе с множеством линий, называющихся *ребрами*, которые соединяют определенные пары вершин. Графы применяются повсеместно, и имеется масса красивых алгоритмов, предназначенных для работы с ними, так что графы естественным образом оказываются в центре внимания многих разделов данной главы. Фактически Stanford GraphBase, в первую очередь, посвящена графам, что отражено даже в ее названии; а слова SGB собирались автором главным образом потому, что их можно использовать для определения интересных и поучительных графов.

Льюис Кэрролл (Lewis Carroll) указал путь, изобретя в конце 1877 года игру "Дублеты" (Doublets). [См. Martin Gardner, *The Universe in a Handkerchief* (1996), Chapter 6.] Идея Кэрролла, которая быстро завоевала популярность, заключалась в преобразовании одного слова в другое путем замены на каждом шаге одной буквы; например, вот как слезы (tears) превращаются в улыбку (smile):*

$$\texttt{tears} - \texttt{sears} - \texttt{stars} - \texttt{stare} - \texttt{stale} - \texttt{stile} - \texttt{smile}. \qquad (11)$$

Кратчайшая такая трансформация является кратчайшим *путем* в графе, в котором вершинами являются английские слова, а ребра соединяют пары слов, расстояние

* Вот как по тем же правилам в русском языке муха превращается в слона: муха — муза — луза — лоза — коза — кора — кара — каре — кафе — кафр — каюр — каюк — крюк — урюк — урок — срок — сток — стон — слон. — *Примеч. пер.*

Хэмминга между которыми равно 1 (т. е. они отличаются одно от другого только одной буквой).

Если ограничиться только словами из SGB, правило Кэрролла дает граф из Stanford GraphBase, официальное название которого — *words*$(5757, 0, 0, 0)$. Каждый граф, определяемый SGB, имеет свой уникальный *идентификатор*, и "кэрролло-образный" граф, порожденный из слов SGB, идентифицируется как *words*(n, l, t, s). Здесь n — количество вершин графа; l либо равно 0, либо представляет собой список весов, используемый для акцентирования различных видов словаря; t представляет собой пороговое значение, с тем чтобы можно было отвергнуть слова с малыми весами; а s — исходное значение для генерации псевдослучайных чисел, которые могут потребоваться для разбиения связей между словами с одинаковыми весами. Детальная информация в данном случае нас не интересует; нескольких примеров вполне достаточно для представления об общей идее.

- *words*$(n, 0, 0, 0)$ — граф, получающийся при применении правила Кэрролла к множеству WORDS(n), для $1 \le n \le 5757$.
- *words*$(1000, \{0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}, 0, s)$ содержит 1000 выбранных случайным образом слов SGB, обычно разных при разных значениях s.
- *words*$(766, \{0, 0, 0, 0, 0, 0, 0, 0, 1, 0\}, 1, 0)$ содержит все пятибуквенные слова из книг автора о TEX и METAFONT.

В последнем графе всего лишь 766 слов, так что мы не сможем образовать очень много длинных путей наподобие (11), хотя

$$\texttt{basic} — \texttt{basis} — \texttt{bases} — \texttt{based} — \texttt{baked} — \texttt{naked} — \texttt{named} — \texttt{names} — \texttt{games} \tag{12}$$

является одним из сто́ящих внимания примеров.

Конечно, имеется много других способов определения ребер графа, вершины которого представляют пятибуквенные слова. Можно, например, вместо расстояния Хэмминга использовать расстояние Евклида. Можно также объявить два слова смежными, если у них имеется общее подслово длиной 4 буквы; эта стратегия существенно обогащает граф, делая возможным превращение chaos в peace, даже ограничиваясь 766 словами, связанными с TEX:

$$\texttt{chaos} — \texttt{chose} — \texttt{whose} — \texttt{whole} — \texttt{holes} — \texttt{hopes} — \texttt{copes} — \texttt{scope} — \texttt{score} — \texttt{store} — \texttt{stare} — \texttt{spare} — \texttt{space} — \texttt{paces} — \texttt{peace}. \tag{13}$$

(Использованное правило позволяет удалить букву, а затем вставить другую — возможно, в другом месте.) Мы можем выбрать и совершенно иную стратегию, такую как размещение ребра между словами-векторами $a_1a_2a_3a_4a_5$ и $b_1b_2b_3b_4b_5$ тогда и только тогда, когда их скалярное произведение $a_1b_1+a_2b_2+a_3b_3+a_4b_4+a_5b_5$ кратно некоторому параметру m. Алгоритмы для работы с графами умеют одинаково успешно справляться с разными видами данных.

Слова SGB приводят также к интересному семейству *ориентированных* графов, если создавать ребра $a_1a_2a_3a_4a_5 \to b_1b_2b_3b_4b_5$ тогда, когда $\{a_2, a_3, a_4, a_5\} \subseteq \{b_1, b_2, b_3, b_4, b_5\}$ при рассмотрении слов как мультимножеств. (Удалить первую букву, добавить другую и переупорядочить.) Применение этого правила позволяет, например, преобразовать words в graph с помощью кратчайшего ориентированного

пути длиной 6:

$$\texttt{words} \to \texttt{dross} \to \texttt{soars} \to \texttt{orcas} \to \texttt{crash} \to \texttt{sharp} \to \texttt{graph}. \tag{14}$$

Теория — первый член ряда Тейлора для практики.

— TOMAC M. КОВЕР (THOMAS M. COVER) (1992)

Количество терминологических систем, используемых в настоящее время в теории графов, неплохо аппроксимируется количеством теоретиков в этой области.

— РИЧАРД П. СТЕНЛИ (RICHARD P. STANLEY) (1986)

Основы теории графов. Граф G состоит из множества вершин V и множества ребер E, которые представляют собой пары различных вершин. Мы будем считать, что V и E являются *конечными* множествами, если только явно не указано иное. Мы пишем u — v, если u и v являются вершинами, такими, что $\{u,v\} \in E$, и $u \not- v$, если u и v являются вершинами, такими, что $\{u,v\} \notin E$. Вершины с u — v называются соседними, или смежными, в G. Одно из следствий этого определения заключается в том, что u — v тогда и только тогда, когда v — u. Другое следствие заключается в том, что $v \not- v$ для всех $v \in V$; другими словами, ни одна из вершин не смежна сама себе. (Однако ниже мы рассмотрим мультиграфы, в которых разрешены петли, идущие из вершины в саму себя.)

Граф $G' = (V', E')$ является *подграфом* $G = (V, E)$, если $V' \subseteq V$ и $E' \subseteq E$. Это *остовный* подграф G, если в действительности $V' = V$. И это *порожденный* (*индуцированный*) подграф G, если, когда V' представляет собой заданное подмножество вершин, E' имеет максимально возможное количество ребер. Другими словами, когда $V' \subseteq V$, подграфом $G = (V, E)$, порожденным V', является $G' = (V', E')$, где

$$E' = \big\{\, \{u,v\} \;\big|\; u \in V',\, v \in V' \text{ и } \{u,v\} \in E \,\big\}. \tag{15}$$

Этот подграф G' обозначается как $G \,|\, V'$ и часто называется "G, ограниченный V'". В распространенном случае, когда $V' = V \setminus \{v\}$, мы записываем просто $G \setminus v$ ("G минус вершина v") как сокращение для $G \mid (V \setminus \{v\})$. Аналогичная запись $G \setminus e$ (где $e \in E$) используется для обозначения подграфа $G' = (V, E \setminus \{e\})$, получаемого путем удаления ребра, а не вершины. Заметим, что все описанные ранее SGB-графы $\mathit{words}(n, l, t, s)$ являются порожденными подграфами основного графа $\mathit{words}(5757, 0, 0, 0)$; в этих рассмотренных графах изменяется только словарь, но не правило смежности.

Граф с n вершинами и e ребрами называется имеющим *порядок* n и *размер* e. Простейшими и наиболее важными графами порядка n являются *полный граф* K_n, *путь* P_n и *цикл* C_n. Предположим, что вершинами являются $V = \{1, 2, \ldots, n\}$. Тогда

- K_n имеет $\binom{n}{2} = \frac{1}{2}n(n-1)$ ребер u — v для $1 \le u < v \le n$; каждый граф с n вершинами является остовным подграфом K_n;
- P_n имеет $n-1$ ребер v — $(v{+}1)$ для $1 \le v < n$, где $n \ge 1$; это — путь длиной $n{-}1$ от 1 до n;
- C_n имеет n ребер v — $((v \bmod n){+}1)$ для $1 \le v \le n$, где $n \ge 1$; он является графом, только когда $n \ge 3$ (C_1 и C_2 являются мультиграфами).

В действительности можно определить K_n, P_n и C_n на вершинах $\{0,1,\ldots,n-1\}$ или на *любом* n-элементном множестве V вместо $\{1,2,\ldots,n\}$, поскольку два графа, отличающихся только именами вершин, но не структурой ребер, комбинаторно эквивалентны.

Формально графы $G=(V,E)$ и $G'=(V',E')$ *изоморфны*, если существует взаимно однозначное соответствие φ между V и V', такое, что u — v в G тогда и только тогда, когда $\varphi(u)$ — $\varphi(v)$ в G'. Для указания изоморфности графов G и G' часто используется обозначение $G\cong G'$; впрочем, часто мы будем менее строги, рассматривая изоморфные графы, как если бы они были эквивалентными, и время от времени записывая $G=G'$, даже когда множества вершин G и G' не строго идентичны.

Небольшие графы можно определить, просто начертив диаграмму, в которой вершины изображены маленькими кружка́ми, а ребра — линиями между ними. На рис. 2 показано несколько важных примеров, свойства которых мы изучим позже. Граф Петерсена на рис. 2, (д) назван по имени Юлиуса Петерсена (Julius Petersen), одного из основателей теории графов, который использовал его для опровержения одного правдоподобного предположения [L'*Intermédiaire des Mathématiciens* **5** (1898), 225–227]; в действительности это замечательная конфигурация, которая служит контрпримером для многих оптимистических предсказаний о том, что может оказаться истинно для всех графов в целом. Граф Чватала (см. рис. 2, (е)) описан Вацлавом Чваталом (Václav Chvátal) в *J. Combinatorial Theory* **9** (1970), 93–94.

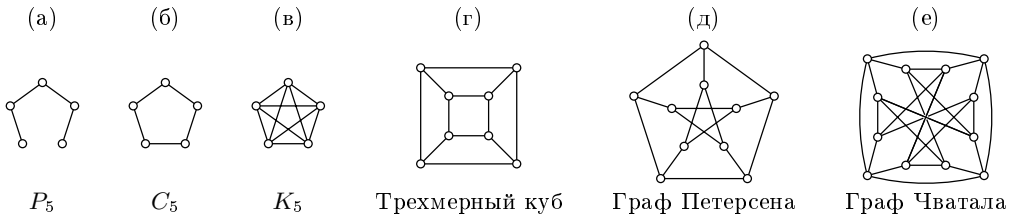
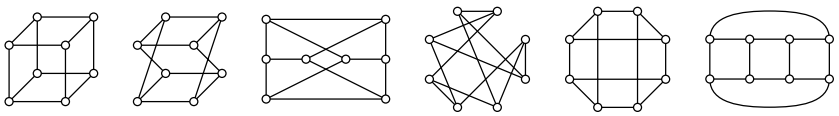


Рис. 2. Шесть примеров графов с $(5,5,5,8,10,12)$ вершинами и $(4,5,10,12,15,24)$ ребрами соответственно.

Линии на диаграмме могут пересекаться одна с другой в точках, не являющихся вершинами. Например, центральная точка на рис. 2, (е) *не* является вершиной графа Чватала. Граф называется *планарным*, если существует способ изобразить его без каких бы то ни было пересечений. Ясно, что P_n и C_n всегда планарны; представленная на рис. 2, (г) диаграмма показывает, что трехмерный куб также является планарным графом. Но K_5 имеет слишком много вершин, чтобы быть планарным (см. упр. 46).

Степенью вершины является количество ее соседей. Если все вершины имеют одну и ту же степень, граф называется *регулярным*. На рис. 2, например, P_5 является нерегулярным графом, так как две его вершины имеют степень 1, а три — степень 2. Остальные пять графов регулярны, со степенями соответственно $(2,4,3,3,4)$. Регулярный граф степени 3 часто называется кубическим или трехвалентным.

Имеется много способов изобразить тот или иной граф, и одни из них более выразительны, чем другие. Например, каждая из шести диаграмм

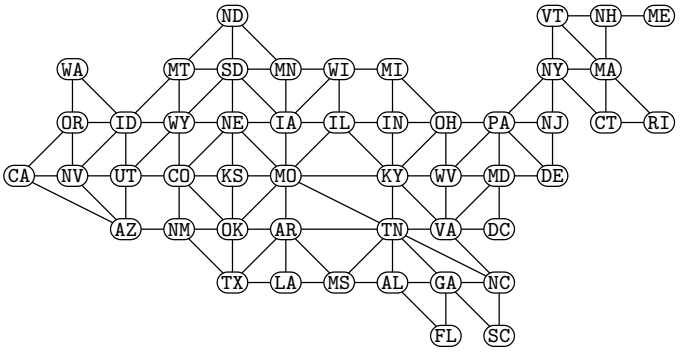


(16)

изоморфна трехмерному кубу (рис. 2, (г)). Диаграмма графа Чватала, показанная на рис. 2, (е) и раскрывающая неожиданную симметрию графа, придумана Адрианом Бонди (Adrian Bondy) через много лет после публикации статьи Чватала.

Симметриями графа, известными также как его *автоморфизмы*, являются перестановки его вершин, сохраняющие отношение смежности. Другими словами, перестановка φ является автоморфизмом G, если $\varphi(u)$ — $\varphi(v)$, когда u — v в графе G. Хорошо продуманный чертеж наподобие рис. 2, (е) может выявить симметрию графа, но одна диаграмма не всегда в состоянии показать все существующие симметрии. Например, трехмерный куб имеет 48 автоморфизмов, а граф Петерсена — 120. Мы изучим алгоритмы для работы с изоморфизмами и автоморфизмами в разделе 7.2.3. Симметрии часто помогают избавиться от излишних вычислений, делая алгоритмы при работе с графами, обладающими k автоморфизмами, почти в k раз быстрее.

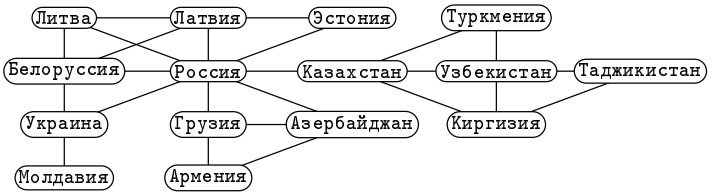
Графы, связанные с объектами реального мира, обычно существенно отличаются от чисто математических графов наподобие приведенных на рис. 2. Например, вот достаточно знакомый американцам граф, вовсе не имеющий симметрии, но зато обладающий достоинством планарности.



(17)

Он представляет внутренние границы США, и позже мы используем его в нескольких примерах. Для удобства вместо изображения пустых кружков 49 вершин на этой диаграмме помечены двухбуквенными кодами штатов.*

* *От переводчика*: пожалуй, читателям переводного издания книги более знакомыми окажутся подобные географические графы, представляющие, например, республики бывшего СССР:



(17, с)

Пути и циклы. Остовный путь P_n графа называется *гамильтоновым путем*, и остовный цикл C_n — *гамильтоновым циклом*, по имени У. Р. Гамильтона (W. R. Hamilton), предложившего в 1856 году головоломку, состоящую в поиске таких путей по ребрам додекаэдра. Независимо от него эту задачу для многогранников в общем случае изучал Т. П. Киркман (T. P. Kirkman) в *Philosophical Transactions* **146** (1856), 413–418; **148** (1858), 145–161. [См. N. L. Biggs, E. K. Lloyd, R. J. Wilson *Graph Theory 1736–1936* (1998), Chapter 2.] Однако задача поиска остовного пути или цикла гораздо старше; на самом деле ее можно рассматривать как наиболее старую из комбинаторных задач, поскольку поиск путей обхода шахматной доски конем имеет долгую историю, ведущую вглубь веков — в Индию IX века (см. раздел 7.3.3). Граф называется *гамильтоновым*, если он имеет гамильтонов цикл. (Граф Петерсена, кстати, является наименьшим 3-регулярным графом, не являющимся ни планарным, ни гамильтоновым; см. C. de Polignac, *Bull. Soc. Math. de France* **27** (1899), 142–145.)

Обхватом (*girth*) графа называется длина его наименьшего цикла; обхват ациклического графа (не содержащего циклов) бесконечен. Например, шесть графов на рис. 2 имеют обхваты $(\infty, 5, 3, 4, 5, 4)$ соответственно. Нетрудно доказать, что граф с минимальной степенью k и обхватом 5 должен иметь как минимум k^2+1 вершин. Дальнейший анализ показывает, что фактически этот минимум достижим, только если $k = 2$ (C_5), $k = 3$ (граф Петерсена), $k = 7$ или, вероятно, $k = 57$ (см. упр. 63 и 65.)

Расстоянием $d(u, v)$ между двумя вершинами u и v является минимальная длина пути от u до v в графе; оно равно бесконечности, если такого пути не существует. Ясно, что $d(v, v) = 0$ и что $d(u, v) = d(v, u)$. Справедливо также неравенство треугольника

$$d(u, v) + d(v, w) \;\geq\; d(u, w). \tag{18}$$

Ибо, если $d(u, v) = p$, $d(v, w) = q$, $p < \infty$ и $q < \infty$, то существуют пути

$$u = u_0 — u_1 — \cdots — u_p = v \quad \text{и} \quad v = v_0 — v_1 — \cdots — v_q = w, \tag{19}$$

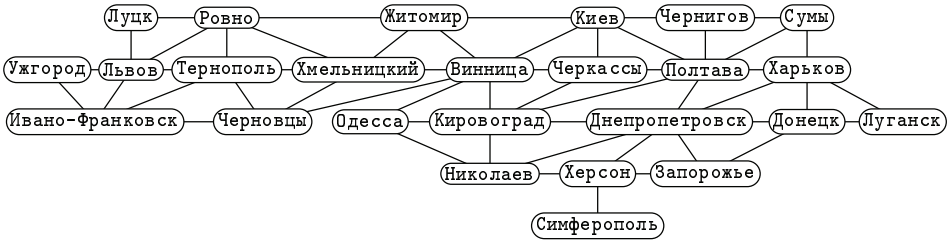
и можно найти наименьший индекс r, такой, что $u_r = v_s$ для некоторого s. Тогда

$$u_0 — u_1 — \cdots — u_{r-1} — v_s — v_{s+1} — \cdots — v_q \tag{20}$$

представляет собой путь длиной $\leq p + q$ от u до w.

Диаметром графа является максимум $d(u, v)$ по всем вершинам u и v. Граф называется *связным*, если его диаметр конечен. Вершины графа всегда могут быть

или области Украины (соответствующий граф для России занял бы слишком много места):



(17, у)

(Здесь вершины графа помечены областными центрами и столицей автономной республики Крым, а не названиями соответствующих областей.)

разбиты на связные *компоненты*, где две вершины, u и v, принадлежат одному и тому же компоненту тогда и только тогда, когда $d(u,v) < \infty$.

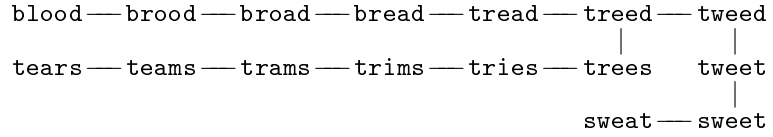
В графе *words*$(5757,0,0,0)$, например, $d(\texttt{tears},\texttt{smile}) = 6$, поскольку (11) представляет собой кратчайший путь от `tears` к `smile`. Кроме того, $d(\texttt{tears},\texttt{happy}) = 6$, $d(\texttt{smile},\texttt{happy}) = 10$ и $d(\texttt{world},\texttt{court}) = 6$. Но $d(\texttt{world},\texttt{happy}) = \infty$; таким образом, рассматриваемый граф не является связным. В действительности он содержит 671 слово наподобие `aloof`, которые не имеют соседей и сами по себе образуют связные компоненты порядка 1. Пары слов наподобие `alpha` — `aloha`, `droid` — `druid` и `opium` — `odium` дают еще 103 связных компонента порядка 2. Одни компоненты порядка 3 наподобие `chain` — `chair` — `choir` являются путями; другие, такие как {`getup`, `letup`, `setup`}, представляют собой циклы. Имеется также немного большее количество небольших компонентов, наподобие занимательного пути

$$\texttt{login} — \texttt{logic} — \texttt{yogic} — \texttt{yogis} — \texttt{yogas} — \texttt{togas}, \tag{21}$$

слова в котором не имеют других соседей. Но подавляющее большинство пятибуквенных слов принадлежит гигантскому компоненту порядка 4493. Если вы можете пройти два шага от имеющегося у вас слова, меняя две различные буквы, то шансы, что это слово принадлежит указанному гигантскому компоненту, превышают 15 к 1.

Аналогично граф *words*$(n,0,0,0)$ имеет при $n = (5000,4000,3000,2000,1000)$ гигантский компонент порядка $(3825,2986,2056,1186,224)$ соответственно. Но если n мало́, то ребер для связности оказывается недостаточно. Например, *words*$(500,0,0,0)$ имеет 327 различных компонентов, причем среди них нет ни одного компонента порядка 15 или выше.

Концепцию расстояния можно обобщить и рассматривать величину $d(v_1,v_2,\ldots,v_k)$ для любого значения k, которая представляет собой минимальное количество ребер в связанном подграфе, содержащем вершины $\{v_1,v_2,\ldots,v_k\}$. Например, $d(\texttt{blood},\texttt{sweat},\texttt{tears})$ оказывается равным 15, поскольку подграф



(22)

имеет 15 ребер, а подходящего подграфа из 14 ребер не существует.

В разделе 2.3.4.1 мы упоминали, что связный граф с наименьшим количеством ребер называется *свободным деревом*. Подграф, соответствующий обобщенному расстоянию $d(v_1,\ldots,v_k)$, всегда будет свободным деревом. Его ошибочно называют *деревом Штейнера*, поскольку Якоб Штейнер (Jacob Steiner) однажды упомянул случай $k = 3$ для точек $\{v_1,v_2,v_3\}$ на евклидовой плоскости [*Crelle* **13** (1835), 362–363]. Франц Хайнен (Franz Heinen) решил эту задачу в *Über Systeme von Kräften* (1834); Гаусс (Gauss) распространил анализ на случай $k = 4$ в письме к Шумахеру (Schumacher) от 21 марта 1836 года.

Раскраска графа. Граф называется *k-дольным*, или *k-раскрашиваемым*, если его вершины могут быть разбиты на k или менее частей так, что концы каждого ребра принадлежат различным частям — или, что то же самое, если имеется способ раскрасить его вершины не более чем k различными цветами, так, чтобы смежные

вершины не были окрашены в один и тот же цвет. Знаменитая Теорема четырех красок, сформулированная в 1852 году Ф. Гутри (F. Guthrie) и окончательно доказанная с применением компьютера К. Аппелем (K. Appel), В. Хакеном (W. Haken) и Д. Кохом (J. Koch) [*Illinois J. Math.* **21** (1977), 429–567], гласит, что *любой планарный граф является 4-раскрашиваемым*. Простое доказательство этой теоремы неизвестно, но частные случаи наподобие (17) могут быть раскрашены на глаз (см. упр. 45); в общем случае для 4-раскрашивания планарного графа достаточно $O(n^2)$ шагов [N. Robertson, D. P. Sanders, P. Seymour и R. Thomas, *STOC* **28** (1996), 571–575].

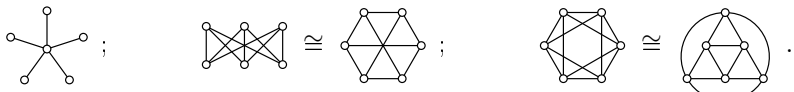
На практике особенно важен случай 2-раскрашиваемых графов. Такие графы в общем случае называются *двудольными* или просто *биграфами*; каждое ребро такого графа имеет по одной конечной точке в каждой из частей.

Теорема B. *Граф является двудольным тогда и только тогда, когда он не содержит циклов нечетной длины.*

Доказательство. [См. D. König, *Math. Annalen* **77** (1916), 453–454.] Каждый подграф k-дольного графа является k-дольным. Следовательно, цикл C_n может быть подграфом двудольного графа, только если C_n сам является двудольным, а в этом случае n должно быть четным.

И наоборот, если граф не содержит циклов нечетной длины, мы можем раскрасить его вершины двумя цветами $\{0, 1\}$, придерживаясь следующей процедуры. Изначально все вершины не окрашены. Если все соседи окрашенных вершин уже окрашены, выберем неокрашенную вершину w и окрасим ее в цвет 0. В противном случае выберем окрашенную вершину u, которая имеет неокрашенного соседа v; назначим v противоположный u цвет. В упр. 48 доказывается, что в конечном итоге при этом получится корректная раскраска. ∎

Полный двудольный граф $K_{m,n}$ представляет собой наибольший двудольный граф, вершины которого разделены на две части размерами m и n. Его можно определить на множестве вершин $\{1, 2, \ldots, m+n\}$, говоря, что u — v тогда, когда $1 \le u \le m < v \le m + n$. Другими словами, $K_{m,n}$ имеет mn ребер, по одному для каждого выбора одной вершины из одной части и второй — из другой. Аналогично *полный k-дольный граф* $K_{n_1,\ldots,n_k}$ имеет $N = n_1 + \cdots + n_k$ вершин, разделенных на части размерами $\{n_1, \ldots, n_k\}$, и имеет ребра между любыми двумя вершинами, не принадлежащими одной и той же части. Вот несколько примеров для $N = 6$:



$K_{1,5}$ $K_{3,3}$ $K_{2,2,2}$ (23)

Обратите внимание, что $K_{1,n}$ представляет собой свободное дерево; его часто называют *звездой* порядка $n + 1$.

Далее я буду говорить "орграф" вместо "ориентированный граф". Это просто и кратко и явно войдет в моду.

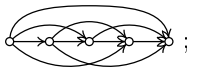
— ГЁРГИ ПОЙА (GYÖRGY PÓLYA),
письмо ФРЭНКУ ХАРАРИ (FRANK HARARY) (ок. 1954 года)

Ориентированные графы. В разделе 2.3.4.2 мы определили *ориентированные графы* (или *орграфы*), которые очень похожи на графы, с тем отличием, что вместо ребер они имеют *дуги*. Дуга $u \longrightarrow v$ направлена от одной вершины к другой, в то время как ребро u — v соединяет две вершины, не различая их. Кроме того, ориентированные графы могут иметь петли $v \longrightarrow v$, идущие из вершины в нее же, а также между двумя вершинами u и v может находиться более одной дуги $u \longrightarrow v$.

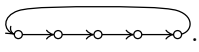
Формально ориентированным графом $D = (V, A)$ порядка n и размера m являются множество V из n вершин и мультимножество A из m упорядоченных пар (u, v), где $u \in V$ и $v \in V$. Упорядоченные пары называются дугами, и мы записываем $u \longrightarrow v$, когда $(u, v) \in A$. Ориентированный граф называется *простым*, если A является множеством, а не мультимножеством, т. е. если для всех u и v имеется не более одной дуги (u, v). Каждая дуга (u, v) имеет начальную вершину u и конечную вершину v, именуемую также верхушкой (tip). Каждая вершина имеет *исходящую степень* $d^+(v)$, равную количеству дуг, для которых v является начальной вершиной, и *входящую степень* $d^-(v)$, равную количеству дуг, для которых v является верхушкой. Вершина с нулевой входящей степенью называется "источником", а вершина с нулевой исходящей степенью называется "стоком". Заметим, что $\sum_{v\in V} d^+(v) = \sum_{v\in V} d^-(v)$, поскольку обе суммы равны m — общему количеству дуг.

Большинство концепций, определенных для графов, естественным путем распространяются и на ориентированные графы. Достаточно просто вставить слово "ориентированный", когда возникает необходимость различать ребра и дуги. Например, ориентированные графы имеют ориентированные подграфы, которые могут быть остовными, порожденными или ни теми, ни другими. Изоморфизмом между ориентированными графами $D = (V, A)$ и $D' = (V', A')$ является взаимно однозначное соответствие φ между V и V', при котором количество дуг $u \longrightarrow v$ в D равно количеству дуг $\varphi(u) \longrightarrow \varphi(v)$ в D' для всех $u, v \in V$.

На диаграммах ориентированных графов вместо линий между вершинами используются стрелки. Простейшими и наиболее важными ориентированными графами порядка n являются ориентированные варианты графов K_n, P_n и C_n, а именно *транзитивный турнир* $K_n^{\rightarrow}$, *ориентированный путь* $P_n^{\rightarrow}$ и *ориентированный цикл* $C_n^{\rightarrow}$. Схематически для $n = 5$ они показаны на следующих диаграммах:


$K_5^{\rightarrow}$

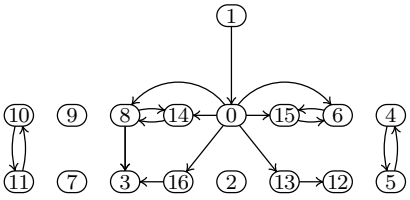
$P_5^{\rightarrow}$


$C_5^{\rightarrow}$ (24)

Имеется также *полный ориентированный граф* J_n, который представляет собой наибольший простой ориентированный граф с n вершинами; он имеет n^2 дуг $u \longrightarrow v$, по одной для каждого варианта выбора u и v.

На рис. 3 показана более сложная диаграмма порядка 17, которую можно назвать отчетливо ориентированной: это ориентированный граф, описанный Эркюлем Пуаро (Hercule Poirot) в детективе Агаты Кристи (Agatha Christie) *Убийство в Восточном экспрессе* (1934).* Вершины соответствуют местам в описанном в детективе

* Непереводимая игра слов: "отчетливо ориентированная" ("expressly oriented") перекликается с оригинальным названием повести А. Кристи *Murder on the Orient Express*. Имена персонажей на диаграмме взяты из перевода Л. Беспаловой (Агата Кристи. *Восточный экспресс. Десять негритят. Романы.* — М., 1990). — *Примеч. пер.*



Условные обозначения
0 — Пьер Мишель, француз, проводник
1 — Эркюль Пуаро, бельгиец, детектив
2 — Сэмьюэл Эдуард Рэтчетт, погибший американец
3 — Каролина Марта Хаббард, американка, домохозяйка
4 — Эдуард Генри Мастермэн, англичанин, слуга
5 — Антонио Фоскарелли, итальянец, агент по продажам
6 — Гектор Уиллард Маккуин, американец, секретарь
7 — Харви Харрис, англичанин, не явился
8 — Хильдегарда Шмидт, немка, горничная
9 — (Пустое место)
10 — Грета Ольсон, шведка, экономка
11 — Мэри Хермиона Дебенхэм, англичанка, гувернантка
12 — Хелена Мария Андрени, прекрасная графиня
13 — Рудольф Андрени, венгр, граф, дипломат
14 — Наталья Драгомирова, русская, княгиня
15 — Полковник Арбэтнот, англичанин, офицер из Индии
16 — Сайрус Бетман Хардман, американец, детектив

Рис. 3. Ориентированный граф порядка 17 размером 18, придуманный Агатой Кристи.

спальном вагоне Стамбул–Кале, а дуги $u \longrightarrow v$ означают, что пассажир с места u подтверждает алиби пассажира с места v. В этом графе имеется шесть связных компонентов, а именно $\{0,1,3,6,8,12,13,14,15,16\}$, $\{2\}$, $\{4,5\}$, $\{7\}$, $\{9\}$ и $\{10,11\}$, поскольку связность ориентированного графа определяется путем рассмотрения дуг как ребер.

Две дуги являются *последовательными*, если верхушка первой является начальной вершиной второй. Ряд последовательных дуг $(a_1, a_2, \ldots, a_k)$ называется *обходом* (*walk*) длиной k; его можно определить с помощью как дуг, так и вершин:

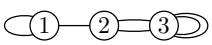
$$v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} v_2 \cdots v_{k-1} \xrightarrow{a_k} v_k. \tag{25}$$

В простом ориентированном графе достаточно просто указать вершины; например, $1 \longrightarrow 0 \longrightarrow 8 \longrightarrow 14 \longrightarrow 8 \longrightarrow 3$ представляет собой обход для графа на рис. 3. Обход (25) представляет собой ориентированный путь, если вершины $\{v_0, v_1, \ldots, v_k\}$ различны; он является ориентированным циклом, если все вершины различны, за исключением $v_k = v_0$.

В ориентированном графе ориентированное расстояние $d(u,v)$ представляет собой количество дуг в кратчайшем *ориентированном* пути от u к v, которое также равно длине кратчайшего обхода от u к v. Оно может отличаться от $d(v,u)$; но неравенство треугольника (18) остается корректным.

Каждый граф может рассматриваться как ориентированный граф, поскольку ребро u — v по сути эквивалентно соответствующей паре дуг, $u \longrightarrow v$ и $v \longrightarrow u$. Ориентированный граф, полученный таким образом, обладает всеми свойствами исходного графа; например, степень каждой вершины графа становится исходящей степенью ориентированного графа, а также ее входящей степенью. Кроме того, расстояния остаются неизменными.

Мультиграф (V,E) подобен графу, с тем отличием, что его ребра E могут быть *мультимножеством* пар $\{u,v\}$; в мультиграфе разрешены также ребра v — v, представляющие собой петли из вершины в нее же, соответствующие "мультипарам" $\{v,v\}$. Например,



(26)

представляет собой мультиграф порядка 3 с 6 ребрами, $\{1,1\}$, $\{1,2\}$, $\{2,3\}$, $\{2,3\}$, $\{3,3\}$ и $\{3,3\}$. Степени вершин в этом примере равны $d(1) = d(2) = 3$ и $d(3) = 6$, поскольку каждая петля вносит в степень вершины вклад, равный 2. При рассмот-

рении мультиграфа как ориентированного графа петля v — v становится *двумя* дугами-петлями $v \longrightarrow v$.

Представление графов и ориентированных графов. Любой ориентированный граф, а значит, любой граф или мультиграф, полностью описывается его *матрицей смежности* $A = (a_{uv})$, имеющей n строк и n столбцов при наличии у ориентированного графа n вершин. Каждый элемент a_{uv} этой матрицы определяет количество дуг от u до v. Например, матрицами смежности для $K_3^{\rightarrow}$, $P_3^{\rightarrow}$, $C_3^{\rightarrow}$, J_3 и (26) являются соответственно

$$K_3^{\rightarrow} = \begin{pmatrix} 011 \\ 001 \\ 000 \end{pmatrix}, \quad P_3^{\rightarrow} = \begin{pmatrix} 010 \\ 001 \\ 000 \end{pmatrix}, \quad C_3^{\rightarrow} = \begin{pmatrix} 010 \\ 001 \\ 100 \end{pmatrix}, \quad J_3 = \begin{pmatrix} 111 \\ 111 \\ 111 \end{pmatrix}, \quad A = \begin{pmatrix} 210 \\ 102 \\ 024 \end{pmatrix}. \tag{27}$$

Мощный математический инструментарий теории матриц позволяет доказывать многие нетривиальные результаты для графов путем изучения их матриц смежности; в упр. 65 представлены особенно впечатляющие примеры того, что может быть сделано таким путем. Одной из основных причин этого является то, что умножение матриц в контексте ориентированных графов имеет простую интерпретацию. Рассмотрим квадрат A, в котором элемент на пересечении строки u и столбца v по определению равен

$$(A^2)_{uv} = \sum_{w \in V} a_{uw} a_{wv}. \tag{28}$$

Поскольку a_{uw} — количество дуг от u до w, мы видим, что $a_{uw}a_{wv}$ — количество обходов вида $u \longrightarrow w \longrightarrow v$. Следовательно, $(A^2)_{uv}$ является общим количеством обходов длиной 2 от u до v. Аналогично элементы A^k дают общее количество обходов длиной k между любыми упорядоченными парами вершин для всех $k \geq 0$. Например, матрица A в (27) дает

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & 2 \\ 0 & 2 & 4 \end{pmatrix}, \qquad A^2 = \begin{pmatrix} 5 & 2 & 2 \\ 2 & 5 & 8 \\ 2 & 8 & 20 \end{pmatrix}, \qquad A^3 = \begin{pmatrix} 12 & 9 & 12 \\ 9 & 18 & 42 \\ 12 & 42 & 96 \end{pmatrix}; \tag{29}$$

таким образом, существует 12 обходов длиной 3 от вершины 1 мультиграфа (26) до вершины 3 и 18 таких обходов из вершины 2 назад в нее же.

Переупорядочение вершин заменяет матрицу смежности A на $P^{-}AP$, где P — матрица перестановки (0–1-матрица, в каждой строке и каждом столбце которой имеется ровно одна единица), а $P^{-} = P^T$ представляет собой матрицу для обратной перестановки. Таким образом,

$$\begin{pmatrix} 210 \\ 102 \\ 024 \end{pmatrix}, \quad \begin{pmatrix} 201 \\ 042 \\ 120 \end{pmatrix}, \quad \begin{pmatrix} 012 \\ 120 \\ 204 \end{pmatrix}, \quad \begin{pmatrix} 021 \\ 240 \\ 102 \end{pmatrix}, \quad \begin{pmatrix} 402 \\ 021 \\ 210 \end{pmatrix} \quad \text{и} \quad \begin{pmatrix} 420 \\ 201 \\ 012 \end{pmatrix} \tag{30}$$

представляют собой все матрицы смежности для (26), и никаких других матриц смежности для этого мультиграфа не существует.

Имеется более чем $2^{n(n-1)/2}/n!$ графов порядка n, при $n > 1$, и почти все они требуют $\Omega(n^2)$ бит данных при наиболее экономичном кодировании. Следовательно, для подавляющего большинства всех возможных графов наилучший способ их представления в компьютере с точки зрения требующейся памяти — это работа с их матрицами смежности.

Однако графы, возникающие в практических задачах, по своим характеристикам существенно отличаются от графов, выбираемых случайным образом из

множества всех возможных. Такие реальные графы, как правило, оказываются "разреженными", имеющими, скажем, $O(n \log n)$ ребер вместо $\Omega(n^2)$, если только n не слишком мало́, поскольку $\Omega(n^2)$ бит данных сложно генерировать. Например, предположим, что вершины соответствуют людям, а ребра — отношению дружбы. Если мы рассмотрим 5 иллиардов людей, мало у кого из них будет более чем 10 тысяч друзей. Но даже если каждый в среднем имеет 10 тысяч друзей, то у графа все равно будет только 2.5×10^{13} ребер, в то время как почти все графы с порядком 5 миллиардов имеют примерно 6.25×10^{18} ребер.

Таким образом, наилучший способ представления графа в машине обычно оказывается отличным от записи n^2 значений a_{uv} матрицы смежности. Вместо этого алгоритмы Stanford GraphBase разработаны для работы со структурой данных, похожей на структуру для связного представления разреженных матриц, рассматривавшейся в разделе 2.2.6, хотя и несколько упрощенной. Такой подход не только доказанно универсален и эффективен, но и прост в использовании.

Представление ориентированного графа в SGB представляет собой комбинацию последовательного и связанного распределения с применением узлов двух базовых типов. Ряд узлов представляет вершины, другие представляют дуги. (Имеется и третий тип узлов, которые представляют весь граф в целом, предназначенный для алгоритмов, работающих одновременно с несколькими графами. Но каждый граф требует только одного узла графа, так что узлы вершин и дуг доминируют.)

Вот как это работает. Каждый ориентированный граф SGB порядка n и размера m построен на последовательном массиве из n узлов вершин, что облегчает доступ к вершине k для $0 \le k < n$. Напротив, m узлов дуг связаны вместе в общем пуле памяти, который, по сути, неструктурирован. Каждый узел вершины обычно занимает 32 байт, а каждый узел дуги — 20 байт (узел графа занимает 220 байт); но размеры узлов могут быть легко изменены. Несколько полей каждого узла фиксированы и имеют определенное значение в любом случае. Остальные поля могут использоваться для различных целей разными алгоритмами или на разных этапах одного и того же алгоритма. Фиксированные части узла называются стандартными полями, а многоцелевые части — сервисными полями.

Каждый узел вершины имеет два стандартных поля, `NAME` и `ARCS`. Если v — переменная, указывающая на узел вершины, мы называем ее *переменной вершины*. Тогда `NAME`(v) указывает на строку символов, которые могут использоваться для идентификации соответствующей вершины удобным для восприятия человеком способом; например, 49 вершин графа (17) имеют имена наподобие `CA`, `WA`, `OR`, ..., `RI`. Другое стандартное поле, `ARCS`(v), существенно важнее для алгоритмов: оно указывает на узел дуги, первый в односвязном списке длиной $d^+(v)$, в котором имеется по одному узлу для каждой дуги, исходящей из вершины v.

Каждый узел дуги имеет два стандартных поля, `TIP` и `NEXT`; переменная a, которая указывает на узел дуги, называется *переменной дуги*. `TIP`(a) указывает на узел вершины, представляющий верхушку дуги a; `NEXT`(a) указывает на узел дуги, представляющий следующую дугу, начальной вершиной которой является та же вершина, что и у дуги a.

Вершина v с исходящей степенью 0 представляется соответствующим узлом путем присваивания `ARCS`$(v) = \Lambda$ (нулевой указатель). В противном случае, если, скажем, исходящая степень равна 3, структура данных содержит три узла дуг

с $\texttt{ARCS}(v) = a_1$, $\texttt{NEXT}(a_1) = a_2$, $\texttt{NEXT}(a_2) = a_3$ и $\texttt{NEXT}(a_3) = \Lambda$; и эти три дуги из вершины v ведут в $\texttt{TIP}(a_1)$, $\texttt{TIP}(a_2)$, $\texttt{TIP}(a_3)$.

Предположим, например, что мы хотим вычислить исходящую степень вершины v и сохранить ее в сервисном поле ODEG. Это просто.

$$\begin{aligned}&\text{Установить } a \leftarrow \texttt{ARCS}(v) \text{ и } d \leftarrow 0.\\ &\text{Пока } a \neq \Lambda, \text{ устанавливать } d \leftarrow d+1 \text{ и } a \leftarrow \texttt{NEXT}(a).\\ &\text{Установить } \texttt{ODEG}(v) \leftarrow d.\end{aligned} \tag{31}$$

Когда граф или мультиграф рассматривается как ориентированный граф, как упоминалось выше, каждое его ребро u — v эквивалентно двум дугам, $u \longrightarrow v$ и $v \longrightarrow u$. Эти дуги называются парными (mate) и занимают два узла дуг, скажем, a и a', где a появляется в списке дуг, исходящих из u, а a' — в списке дуг, исходящих из v. Тогда $\texttt{TIP}(a) = v$ и $\texttt{TIP}(a') = u$. Мы также записываем

$$\texttt{MATE}(a) = a' \qquad \text{и} \qquad \texttt{MATE}(a') = a \tag{32}$$

в алгоритмах, которые должны быстро переходить от одного списка к другому. Однако обычно нам не требуется хранить явно указатель от дуги на парную ей или использовать сервисное поле MATE в каждом узле дуги, поскольку необходимые связи можно вывести *неявно* при правильном построении структур данных.

Неявный вывод работает примерно следующим образом: при создании каждого ребра u — v неориентированного графа или мультиграфа мы вводим *последовательные* узлы дуг для $u \longrightarrow v$ и $v \longrightarrow u$. Например, если узлу дуги отводится 20 байт, мы резервируем 40 последовательных байтов для каждой новой пары. Мы можем также обеспечить, чтобы адрес в памяти первого байта был кратен 8. Тогда, если узел дуги a находится в памяти по адресу α, узел парной дуги находится по адресу

$$\left\{\begin{matrix}\alpha+20, & \text{если } \alpha \bmod 8 = 0\\ \alpha-20, & \text{если } \alpha \bmod 8 = 4\end{matrix}\right\} \;=\; \alpha - 20 + \big(40 \mathbin{\&} ((\alpha \mathbin{\&} 4) - 1)\big). \tag{33}$$

Такой трюк очень полезен в комбинаторных задачах, когда операции могут выполняться триллионы раз, поскольку при этом экономия в 3.6 нс на одной операции приведет к тому, что программа завершит работу на час раньше. Но способ (33) не является непосредственно "переносимым" из одной реализации в другую. Если размер узла дуги изменится, например, с 20 до 24, то числа 40, 20, 8 и 4 в (33) нужно будет заменить на 48, 24, 16 и 8 соответственно.

В алгоритмах в этой книге не делается никаких предположений о размерах узлов. Вместо этого мы примем соглашение языка программирования C и его потомков о том, что если a указывает на узел дуги, то '$a+1$' представляет собой указатель на узел дуги, следующий за первым в памяти. В общем случае

$$\texttt{LOC}(\texttt{NODE}(a+k)) \;=\; \texttt{LOC}(\texttt{NODE}(a)) + kc, \tag{34}$$

если размер каждого узла дуги равен c байт. Аналогично если v представляет собой переменную вершины, то '$v+k$' будет k-м узлом вершины, следующим за узлом, на который указывает v; фактический адрес памяти этого узла будет равен v плюс умноженное на размер узла вершины значение k.

Стандартные поля узла графа g включают общее количество дуг $\texttt{M}(g)$ и общее количество вершин $\texttt{N}(g)$; указатель на первый узел вершины в последовательном

списке всех узлов вершин VERTICES(g); идентификатор графа ID(g), представляющий собой строку наподобие words(5757,0,0,0); а также некоторые другие поля, необходимые для выделения и освобождения памяти при увеличении или уменьшении графа, или для экспорта графа во внешний формат для взаимодействия с другими пользователями и системами для работы с графами. Но нам редко придется как обращаться к этим полям узла графа, так и давать полное описание формата SGB, поскольку в этой главе мы будем описывать почти все алгоритмы для работы с графами с помощью словесного описания естественным языком на достаточно абстрактном уровне, а не на битовом уровне машинных программ.

Простой алгоритм для работы с графом. Для иллюстрации алгоритма средневысокого уровня наподобие тех, с которыми мы будем иметь дело позже, давайте превратим доказательство теоремы B в пошаговую процедуру, окрашивающую вершины данного графа в два цвета, если граф является двудольным.

Алгоритм B (*Проверка двудольности*). Для заданного графа, представленного в формате SGB, этот алгоритм либо находит 2-раскрашивание с COLOR(v) $\in \{0,1\}$ в каждой вершине v, либо завершается неудачей, если корректное 2-раскрашивание невозможно. Здесь COLOR представляет собой сервисное поле в каждом узле вершины. Другое сервисное поле, LINK(v), представляет собой указатель на вершину и используется для поддержки стека всех окрашенных вершин, соседи которых еще не протестированы. Вспомогательная переменная вершины s указывает на вершину этого стека. Алгоритм также использует переменные u, v, w для вершин и a для дуг. Предполагается, что узлы вершин представляют собой $v_0 + k$ для $0 \le k < n$.

B1. [Инициализация.] Установить COLOR($v_0 + k$) $\leftarrow -1$ для $0 \le k < n$. (Сейчас все вершины не окрашены.) Затем установить $w \leftarrow v_0 + n$.

B2. [Выполнено?] (В этот момент все вершины $\ge w$ окрашены, и то же относится и к соседям всех окрашенных вершин.) Завершить работу алгоритма успешно, если $w = v_0$. В противном случае установить $w \leftarrow w - 1$ (следующему узлу вершины в направлении уменьшения индекса).

B3. [Окраска w при необходимости.] Если COLOR(w) ≥ 0, вернуться к шагу B2. В противном случае установить COLOR(w) $\leftarrow 0$, LINK(w) $\leftarrow \Lambda$ и $s \leftarrow w$.

B4. [Стек $\Rightarrow u$.] Установить $u \leftarrow s$, $s \leftarrow$ LINK(s), $a \leftarrow$ ARCS(u). (Мы будем проверять всех соседей окрашенной вершины u.)

B5. [Завершено с u?] Если $a = \Lambda$, перейти к шагу B8. В противном случае установить $v \leftarrow$ TIP(a).

B6. [Обработка v.] Если COLOR(v) < 0, установить COLOR(v) $\leftarrow 1 -$ COLOR(u), LINK(v) $\leftarrow s$ и $s \leftarrow v$. Если же COLOR(v) $=$ COLOR(u), завершить работу алгоритма неудачей.

B7. [Цикл по a.] Установить $a \leftarrow$ NEXT(a) и вернуться к шагу B5.

B8. [Стек не пустой?] Если $s \ne \Lambda$, вернуться к шагу B4. В противном случае вернуться к шагу B2. ∎

Этот алгоритм представляет собой вариант общей процедуры обхода графа, именуемой поиском в глубину, которую мы детально изучим в разделе 7.4.1. Время его работы составляет $O(m + n)$ при наличии m дуг и n вершин (см. упр. 70);

следовательно, он хорошо приспособлен к распространенному случаю разреженных графов. Путем небольших изменений в случае неудачного завершения алгоритм можно заставить выводить цикл нечетной длины, тем самым доказывая невозможность 2-раскрашивания (см. упр. 72).

Примеры графов. Stanford GraphBase включает библиотеку более чем из трех десятков подпрограмм генерации, способных создавать разнообразные ориентированные и неориентированные графы для использования в экспериментах. Мы уже рассматривали графы *слов*; давайте теперь взглянем на некоторые другие, чтобы увидеть имеющиеся возможности.

• *roget*$(1022, 0, 0, 0)$ представляет собой ориентированный граф с 1022 вершинами и 5075 дугами. Вершины представляют категории слов или концепций, которые П. М. Роджет (P. M. Roget) и Д. Л. Роджет (J. L. Roget) включили в свой знаменитый *Thesaurus* (London: Longmans, Green, 1879). Дуги представляет собой перекрестные ссылки между категориями, имеющиеся в книге. Например, типичными дугами являются `water` ⟶ `moisture`, `discovery` ⟶ `truth`, `fool` ⟶ `sage`, `preparation` ⟶ `learning`, `vulgarity` ⟶ `ugliness`, `wit` ⟶ `amusement`.

• *book*$(\texttt{'jean'}, 80, 0, 1, 356, 0, 0, 0)$ представляет собой граф с 80 вершинами и 254 ребрами. Вершины представляют персонажей романа *Отверженные* (*Les Misérables*) Виктора Гюго (Victor Hugo); ребра соединяют персонажей, встречавшихся в романе друг с другом. Типичными ребрами являются `Fantine` — `Javert`, `Cosette` — `Thénardier`.

• *bi_book*$(\texttt{'jean'}, 80, 0, 1, 356, 0, 0, 0)$ представляет собой двудольный граф с 80+356 вершинами и 727 ребрами. Вершины представляют персонажей и главы упомянутого романа *Отверженные*; ребра соединяют персонажей с главами, в которых они появляются (например, `Napoleon` — `2.1.8`, `Marius` — `4.14.4`).

• *plane_miles*$(128, 0, 0, 0, 1, 0, 0)$ представляет собой планарный граф со 129 вершинами и 381 ребром. Вершины представляют 128 городов США и Канады, а также специальную вершину `INF` для "точки в бесконечности". Ребра определяют так называемую *триангуляцию Делоне* этих городов, основанную на их широте и долготе на плоскости; это означает, что u — v тогда и только тогда, когда существует окружность, проходящая через u и v и не охватывающая никакую иную вершину. Ребра имеются также между `INF` и всеми вершинами, лежащими на выпуклой оболочке всех местоположений городов. Типичными ребрами являются `Seattle, WA` — `Vancouver, BC` — `INF`; `Toronto, ON` — `Rochester, NY`.

• *plane_lisa*$(360, 250, 15, 0, 360, 0, 250, 0, 2295000)$ представляет собой планарный граф с 3027 вершинами и 5967 ребрами. Он получен с помощью обработки оцифрованного изображения *Моны Лизы* Леонардо да Винчи, имеющего 360 строк и 250 столбцов пикселей, состоящей в огрублении интенсивностей пикселей до 16 уровней серого от 0 (черный) до 15 (белый). Полученные в результате 3027 областей одинаковой яркости, связанных "ходом ладьи", рассматриваются затем как соседние, если имеют общую границу между пикселями (рис. 4).

• *bi_lisa*$(360, 250, 0, 360, 0, 250, 8192, 0)$ представляет собой двудольный граф с $360+250 = 610$ вершинами и 40 923 ребрами. Это другой взгляд на знаменитую картину Леонардо да Винчи, на этот раз связывающий строки и столбцы, на пересечении

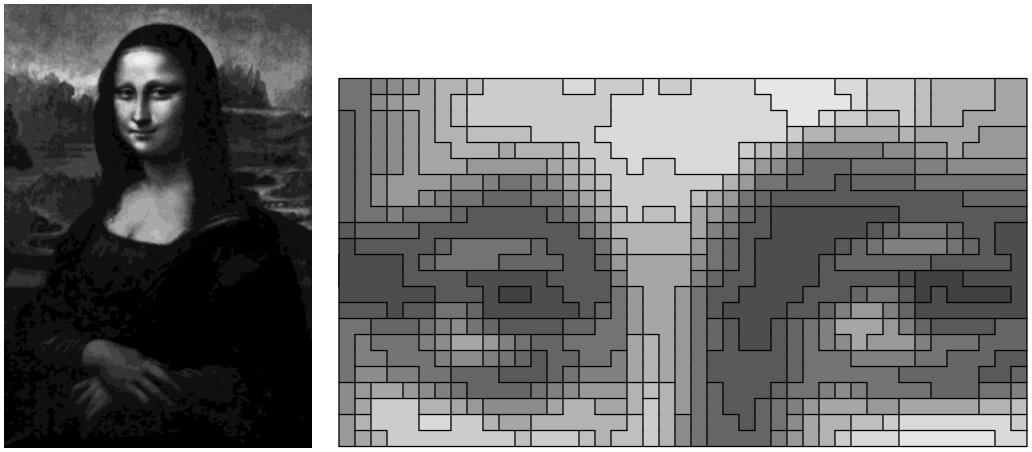


Рис. 4. Цифровая копия *Моны Лизы* и крупный план фрагмента (который лучше рассматривать издали).

которых уровень яркости равен как минимум 1/8. Например, ребро `r102` — `c113` приходится прямо на средину "улыбки Джоконды".

• *raman*$(31, 23, 3, 1)$ представляет собой граф совершенно отличной от других рассмотренных ранее графов SGB природы. Вместо связывания объектов языка, литературы или иных продуктов человеческой культуры, этот так называемый "граф-расширитель Рамануджана" ("Ramanujan expander graph") основан на строгих математических принципах. Каждая из его $(23^3-23)/2 = 6072$ вершин имеет степень 32; следовательно, он имеет 97 152 ребра. Вершины соответствуют классам эквивалентности матриц 2×2, являющихся несингулярными по модулю 23; типичным ребром является `(2,7;1,1)` — `(4,6;1,3)`. Графы Рамануджана важны, в первую очередь, из-за необычно большого обхвата и малого диаметра для их размеров и степеней. У данного графа и обхват, и диаметр равны 4.

• *raman*$(5, 37, 4, 1)$ — аналогично регулярный граф степени 6 с 50 616 вершинами и 151 848 ребрами. Имеет обхват 10, диаметр 10, а кроме того, этот граф двудолен.

• *random_graph*$(1000, 5000, 0, 0, 0, 0, 0, 0, 0, s)$ представляет собой граф с 1000 вершинами, 5000 ребрами и исходным значением для генерации псевдослучайных чисел s. Граф "эволюционирует", начиная с безреберного состояния, путем многократного выбора псевдослучайных номеров вершин $0 \le u, v < 1000$ и добавления ребра u — v, если только не $u = v$ или такое ребро уже имеется в наличии. При $s = 0$ все вершины принадлежат гигантскому компоненту порядка 999, за исключением одной изолированной вершины 908.

• *random_graph*$(1000, 5000, 0, 0, 1, 0, 0, 0, 0, 0)$ представляет собой ориентированный граф с 1000 вершина и 5000 дугами, полученный с помощью аналогичной эволюции. (В действительности каждая из его дуг является также частью *random_graph*$(1000, 5000, 0, 0, 0, 0, 0, 0, 0, 0)$.)

• *subsets*$(5, 1, -10, 0, 0, 0, \#1, 0)$ представляет собой граф с $\binom{11}{5} = 462$ вершинами, по одной для каждого пятиэлементного подмножества множества $\{0, 1, \ldots, 10\}$. Две

вершины являются смежными, если соответствующие подмножества непересекающиеся; таким образом, граф является регулярным со степенью 6 и имеет 1386 ребер. Его можно рассматривать как обобщение графа Петерсена, SGB-имя которого — $subsets(2,1,-4,0,0,0,\#1,0)$.

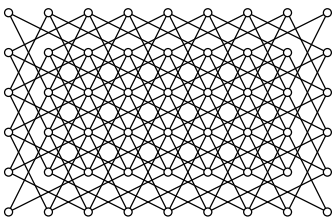
• $subsets(5,1,-10,0,0,0,\#10,0)$ имеет те же 462 вершины, но теперь вершины являются смежными, если соответствующие подмножества имеют четыре общих элемента. Данный граф регулярен со степенью 30 и имеет 6930 ребер.

• $parts(30,10,30,0)$ представляет собой еще один SGB-граф с математической основой. Он имеет 3590 вершин, по одной для каждого разбиения 30 на не более чем 10 частей. Два разбиения рассматриваются как смежные, если одно из них получается путем деления части другого; это правило определяет 31 377 ребер. Ориентированный граф $parts(30,10,30,1)$ аналогичен, но его 31 377 дуг направлены от более коротких к более длинным разбиениям (например, 13+7+7+3 ⟶ 7+7+7+6+3).

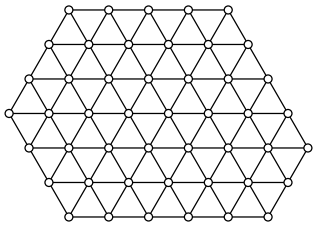
• $simplex(10,10,10,10,10,0,0)$ представляет собой граф с 286 вершинами и 1320 ребрами. Его вершинами являются целочисленные решения $x_1+x_2+x_3+x_4=10$, где $x_i\ge 0$, а именно "композиции 10 на четыре неотрицательные части"; их можно также рассматривать как барицентрические координаты точек внутри тетраэдра. Ребра, такие как 3.1.4.2 — 3.0.4.3, соединяют композиции, которые максимально близки одна к другой.

• $board(8,8,0,0,5,0,0)$ и $board(8,8,0,0,-2,0,0)$ представляют собой графы с 64 вершинами, 168 или 280 ребер которых соответствуют ходам коня или слона в шахматах.

Несметное количество других примеров можно легко получить, варьируя параметры генераторов графов SGB. Например, на рис. 5 показаны два простых варианта *board* и *simplex*; несколько загадочные правила *board* поясняются в упр. 75.



$board(6,9,0,0,5,0,0)$
(ходы коня на доске размером 6×9)



$simplex(10,8,7,6,0,0,0)$
(усеченная треугольная решетка)

Рис. 5. Образцы SGB-графов, связанных с настольными играми.

Алгебра графов. Мы также можем получать новые графы с помощью операций над уже имеющимися. Например, если $G=(V,E)$ — некоторый граф, то его *дополнение* $\overline{G}=(V,\overline{E})$ получается, если положить

$$u \text{ — } v \text{ в } \overline{G} \qquad \Longleftrightarrow \qquad u\ne v \text{ и } u \not\text{—} v \text{ в } G. \tag{35}$$

Таким образом, не ребра становятся ребрами, и наоборот. Обратите внимание, что $\overline{\overline{G}}=G$ и что $\overline{K_n}$ не имеет ребер. Соответствующие матрицы смежности A и $\overline{A}$ удовлетворяют условию

$$A+\overline{A} \;=\; J-I; \tag{36}$$

здесь J — матрица, все элементы которой является единицами, а I — тождественная матрица, так что J и $J-I$ являются соответственно матрицами смежности для J_n и K_n, когда G имеет порядок n.

Кроме того, каждый граф $G=(V,E)$ приводит к *реберному графу* $L(G)$, вершины которого являются ребрами E; два ребра графа G смежны в $L(G)$, если они имеют общую вершину. Таким образом, например, реберный граф $L(K_n)$ имеет $\binom{n}{2}$ вершин и является регулярным графом степени $2n-4$ при $n\ge 2$ (см. упр. 82). Граф называется *k-реберно-раскрашиваемым*, если его реберный граф является k-раскрашиваемым.

Для двух данных графов $G=(U,E)$ и $H=(V,F)$ их *объединением* (*union*) $G\cup H$ является граф $(U\cup V, E\cup F)$, получаемый путем объединения вершин и ребер. Например, предположим, что G и H представляют собой графы ходов ладьи и слона в шахматах; тогда $G\cup H$ является графом ходов ферзя и его официальное SGB-имя имеет вид

$$\mathit{gunion}\,(\mathit{board}\,(8,8,0,0,-1,0,0),\mathit{board}\,(8,8,0,0,-2,0,0),0,0). \tag{37}$$

В частном случае, когда множества вершин U и V непересекающиеся, объединение $G\cup H$ не требует согласованной идентификации вершин для взаимной корреляции; мы получаем диаграмму $G\cup H$, просто изображая диаграмму G, а после нее диаграмму H. Этот частный случай называется соприкосновением (juxtaposition) или *прямой суммой* G и H, и мы будем обозначать ее как $G\oplus H$. Например, легко увидеть, что

$$K_m\oplus K_n \;\cong\; \overline{K_{m,n}} \tag{38}$$

и что каждый граф является прямой суммой своих связных компонентов.

Уравнение (38) является частным случаем общей формулы

$$K_{n_1}\oplus K_{n_2}\oplus\cdots\oplus K_{n_k} \;\cong\; \overline{K_{n_1,n_2,\ldots,n_k}}\,, \tag{39}$$

которая справедлива для полных k-дольных графов при $k\ge 2$. Но (39) не работает при $k=1$ из-за позорного факта: стандартные обозначения теории графов противоречивы! Действительно, $K_{m,n}$ означает полный двудольный граф, но K_n *не* означает полный однодольный граф. Теоретики, работающие в области теории графов, невероятным образом ухитряются десятилетиями жить с этой аномалией и не обезуметь.

Еще одним важным способом комбинации двух непересекающихся графов G и H является образование их *соединения* (*join*) $G\mathbin{—}H$, которое состоит из $G\oplus H$ вместе со всеми ребрами $u\mathbin{—}v$ для $u\in U$ и $v\in V$. [См. А. А. Зыков, *Мат. сборник* **24** (1949), 163–188, §I.3.] А если G и H являются непересекающимися *ориентированными графами*, то их *ориентированное соединение* $G\longrightarrow H$ аналогично, но добавляет к $G\oplus H$ только дуги $u\longrightarrow v$, идущие от U к V.

Прямая сумма двух матриц A и B получается путем помещения B по диагонали ниже и правее A:

$$A\oplus B \;=\; \begin{pmatrix} A & O\\ O & B\end{pmatrix}, \tag{40}$$

где каждое O в данном примере представляет собой матрицу, состоящую из одних нулей, с количеством строк и столбцов, необходимых для корректного выравнивания

получающейся матрицы. Наше обозначение $G \oplus H$ для прямой суммы графов легко запомнить, так как матрица смежности для $G \oplus H$ строго равна прямой сумме соответствующих матриц смежности A и B для G и H. Аналогично матрицами смежности для $G \mathbin{—\!\!—} H$, $G \longrightarrow H$ и $G \longleftarrow H$ являются

$$A \mathbin{—\!\!—} B = \begin{pmatrix} A & J \\ J & B \end{pmatrix}, \qquad A \longrightarrow B = \begin{pmatrix} A & J \\ O & B \end{pmatrix}, \qquad A \longleftarrow B = \begin{pmatrix} A & O \\ J & B \end{pmatrix} \tag{41}$$

соответственно, где J — матрица, состоящая из одних единиц, как в (36). Эти операции ассоциативны и связаны операцией дополнения:

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C, \qquad A \mathbin{—\!\!—} (B \mathbin{—\!\!—} C) = (A \mathbin{—\!\!—} B) \mathbin{—\!\!—} C; \tag{42}$$

$$A \longrightarrow (B \longrightarrow C) = (A \longrightarrow B) \longrightarrow C, \qquad A \longleftarrow (B \longleftarrow C) = (A \longleftarrow B) \longleftarrow C; \tag{43}$$

$$\overline{A \oplus B} = \overline{A} \mathbin{—\!\!—} \overline{B}, \qquad \overline{A \mathbin{—\!\!—} B} = \overline{A} \oplus \overline{B}; \tag{44}$$

$$\overline{A \longrightarrow B} = \overline{A} \longleftarrow \overline{B}, \qquad \overline{A \longleftarrow B} = \overline{A} \longrightarrow \overline{B}; \tag{45}$$

$$(A \oplus B) + (A \mathbin{—\!\!—} B) \;=\; (A \longrightarrow B) + (A \longleftarrow B). \tag{46}$$

Обратите внимание, что, комбинируя (39) с (42) и (44), мы получаем

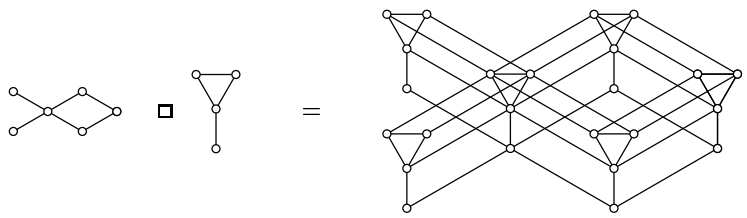
$$K_{n_1,n_2,\ldots,n_k} \;=\; \overline{K_{n_1}} \mathbin{—\!\!—} \overline{K_{n_2}} \mathbin{—\!\!—} \cdots \mathbin{—\!\!—} \overline{K_{n_k}} \tag{47}$$

при $k \geq 2$. Кроме того,

$$K_n = K_1 \mathbin{—\!\!—} K_1 \mathbin{—\!\!—} \cdots \mathbin{—\!\!—} K_1 \quad \text{и} \quad \vec{K_n} = K_1 \longrightarrow K_1 \longrightarrow \cdots \longrightarrow K_1\,, \tag{48}$$

с n копиями K_1, что показывает, что $K_n = K_{1,1,\ldots,1}$ является полным n-дольным графом.

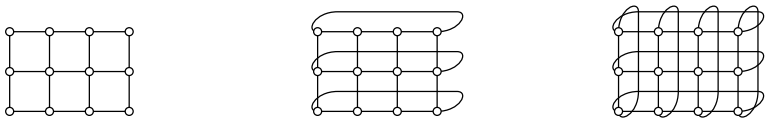
Прямые суммы и соединения аналогичны сложению, поскольку $\overline{K_m} \oplus \overline{K_n} = \overline{K_{m+n}}$ и $K_m \mathbin{—\!\!—} K_n = K_{m+n}$. Можно также скомбинировать графы с помощью алгебраической операции, аналогичной умножению. Например, операция *декартова произведения* образует граф $G \mathbin{\square} H$ порядка mn из графа $G = (U, E)$ порядка m и графа $H = (V, F)$ порядка n. Вершинами $G \mathbin{\square} H$ являются упорядоченные пары (u, v), где $u \in U$ и $v \in V$; ребрами являются $(u, v) \mathbin{—\!\!—} (u', v)$, когда $u \mathbin{—\!\!—} u'$ находится в G, вместе с $(u, v) \mathbin{—\!\!—} (u, v')$, когда $v \mathbin{—\!\!—} v'$ находится в H. Другими словами, $G \mathbin{\square} H$ образуется путем замены каждой вершины G копией H и замены каждого ребра G ребрами между соответствующими вершинами надлежащих копий:



(49)

Как обычно, простейшие частные случаи этого общего построения оказываются особо важными на практике. Когда и G, и H являются путями или циклами, мы получаем "графы в клеточку", а именно $m \times n$-*решетку* $P_m \mathbin{\square} P_n$, $m \times n$-*цилиндр*

$P_m \mathbin{\square} C_n$ и $m \times n$-*тор* $C_m \mathbin{\square} C_n$, показанные ниже для $m = 3$ и $n = 4$:



$P_3 \mathbin{\square} P_4$ (3×4-решетка) $\quad$ $P_3 \mathbin{\square} C_4$ (3×4-цилиндр) $\quad$ $C_3 \mathbin{\square} C_4$ (3×4-тор) (50)

Четыре других заслуживающих внимания пути определения произведений графов также оказываются полезными. В каждом случае вершинами произведения графов являются упорядоченные пары (u, v).

- *Прямое произведение* $G \otimes H$, именуемое также "конъюнкцией" G и H, или их "категорийное произведение" (categorical product) содержит $(u, v) \mathrel{—} (u', v')$, когда $u \mathrel{—} u'$ имеется в G, а $v \mathrel{—} v'$ имеется в H.
- *Сильное произведение* $G \boxtimes H$ объединяет ребра $G \mathbin{\square} H$ с ребрами $G \otimes H$.
- *Нечетное произведение* $G \triangle H$ содержит ребро $(u, v) \mathrel{—} (u', v')$, когда либо $u \mathrel{—} u'$ содержится в G, либо $v \mathrel{—} v'$ содержится в H, но не одновременно.
- *Лексикографическое произведение* $G \circ H$, именуемое также композицией G и H, содержит $(u, v) \mathrel{—} (u', v')$, когда $u \mathrel{—} u'$ содержится в G, и $(u, v) \mathrel{—} (u, v')$, когда $v \mathrel{—} v'$ содержится в H.

Все пять этих операций естественным образом распространяются на произведения $k \geq 2$ графов $G_1 = (V_1, E_1)$, $\ldots$, $G_k = (V_k, E_k)$, вершины которых представляют собой упорядоченные k-кортежи $(v_1, \ldots, v_k)$, такие, что $v_j \in V_j$ для $1 \leq j \leq k$. Например, когда $k = 3$, декартовы произведения $G_1 \mathbin{\square} (G_2 \mathbin{\square} G_3)$ и $(G_1 \mathbin{\square} G_2) \mathbin{\square} G_3$ изоморфны, если рассматривать составные вершины $(v_1, (v_2, v_3))$ и $((v_1, v_2), v_3)$ как одинаковые с (v_1, v_2, v_3). Следовательно, можно записывать декартово произведение без скобок, как $G_1 \mathbin{\square} G_2 \mathbin{\square} G_3$. Наиболее важным примером декартова произведения с k сомножителями является k-мерный куб,

$$P_2 \mathbin{\square} P_2 \mathbin{\square} \cdots \mathbin{\square} P_2; \tag{51}$$

2^k его вершин $(v_1, \ldots, v_k)$ являются смежными, когда расстояние Хэмминга между ними равно 1.

В общем случае предположим, что $v = (v_1, \ldots, v_k)$ и $v' = (v'_1, \ldots, v'_k)$ являются k-кортежами вершин, где $v_j \mathrel{—} v'_j$ в G_j для ровно a из индексов j, а $v_j = v'_j$ в точности для b из индексов. Тогда имеем:

- $v \mathrel{—} v'$ в $G_1 \mathbin{\square} \cdots \mathbin{\square} G_k$ тогда и только тогда, когда $a = 1$ и $b = k - 1$;
- $v \mathrel{—} v'$ в $G_1 \otimes \cdots \otimes G_k$ тогда и только тогда, когда $a = k$ и $b = 0$;
- $v \mathrel{—} v'$ в $G_1 \boxtimes \cdots \boxtimes G_k$ тогда и только тогда, когда $a + b = k$ и $a > 0$;
- $v \mathrel{—} v'$ в $G_1 \triangle \cdots \triangle G_k$ тогда и только тогда, когда a нечетно.

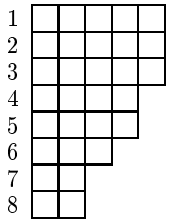
Лексикографическое произведение несколько отличается, поскольку оно не коммутативно; в $G_1 \circ \cdots \circ G_k$ мы имеем $v \mathrel{—} v'$ для $v \neq v'$ тогда и только тогда, когда $v_j \mathrel{—} v'_j$, где j — минимальный индекс, такой, что $v_j \neq v'_j$.

В упр. 91–102 исследуются некоторые фундаментальные свойства произведений графов. См. также книгу *Product Graphs* Вильфреда Имриха (Wilfried Imrich)

и Санди Клавжара (Sandi Klavzar) (2000), в которой содержится исчерпывающее введение в теорию графов, включая алгоритмы для разложения графов на "простые" подграфы.

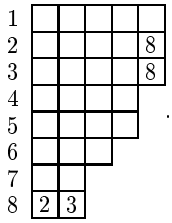
***Графические последовательности степеней.** Последовательность $d_1 d_2 \ldots d_n$ неотрицательных целых чисел называется *графической* (*graphical*), если существует как минимум один граф с вершинами $\{1, 2, \ldots, n\}$, такой, что вершина k имеет степень d_k. Можно считать, что $d_1 \geq d_2 \geq \cdots \geq d_n$. Ясно, что в любом таком графе $d_1 < n$ и что сумма $m = d_1 + d_2 + \cdots + d_n$ любой графической последовательности всегда четна, поскольку она равна удвоенному количеству ребер. Кроме того, легко увидеть, что последовательность 3311 не является графической; следовательно, графические последовательности должны удовлетворять дополнительным условиям. Каким?

Простой способ выяснить, является ли данная последовательность $d_1 d_2 \ldots d_n$ графической, и построить соответствующий граф в случае, если он существует, был открыт В. Гавелом (V. Havel) [*Casopis pro Pěstováni Matematiky* **80** (1955), 477–479]. Мы начинаем с пустой таблицы, имеющей d_k ячеек в строке k; эти ячейки представляют "слоты", в которые мы будем помещать соседей вершины k в строящемся графе. Пусть c_j — количество ячеек в столбце j; таким образом, $c_1 \geq c_2 \geq \cdots$, и при $1 \leq k \leq n$ мы имеем $c_j \geq k$ тогда и только тогда, когда $d_k \geq j$. Предположим, например, что $n = 8$ и $d_1 \ldots d_8 = 55544322$. Тогда исходная таблица имеет вид



(52)

и мы имеем $c_1 \ldots c_5 = 88653$. Идея Гавела состоит в соединении вершины n с d_n вершин с наивысшими степенями. В данном случае, например, мы создаем два ребра, 8 — 3 и 8 — 2, и таблица принимает следующий вид:



(53)

(Мы не используем 8 — 1, поскольку пустые слоты должны продолжать образовывать таблицу; ячейки каждого столбца должны заполняться снизу вверх.) Затем мы устанавливаем $n \leftarrow 7$ и создаем два очередных ребра, 7 — 1 и 7 — 5. Затем наступает очередь следующих трех ребер, 6 — 4, 6 — 3, 6 — 2, и таблица становится

почти наполовину заполненной:

1					7
2				6	8
3				6	8
4				6	
5				7	
6	2	3	4		
7	5	1			
8	2	3			

. (54)

Мы свели задачу к поиску графа с последовательностью степеней $d_1 \dots d_5 = 43333$; в этот момент мы также имеем $c_1 \dots c_4 = 5551$. Читателю предлагается самостоятельно заполнить оставшиеся пустые места, перед тем как посмотреть ответ в упр. 103.

Алгоритм H (*Генератор графа для определенных степеней*). Этот алгоритм для заданных $d_1 \geq \cdots \geq d_n \geq d_{n+1} = 0$ создает ребра между вершинами $\{1, \dots, n\}$ таким образом, что ровно d_k ребер связаны с вершиной k, для $1 \leq k \leq n$, если, конечно, последовательность $d_1 \dots d_n$ является графической. Массив $c_1 \dots c_{d_1}$ используется в качестве вспомогательного.

H1. [Установка c.] Начать с $k \leftarrow d_1$ и $j \leftarrow 0$. Затем, пока $k > 0$, выполнять следующие операции: установить $j \leftarrow j + 1$; пока $k > d_{j+1}$, устанавливать $c_k \leftarrow j$ и $k \leftarrow k - 1$. Успешно завершить работу алгоритма, если $j = 0$ (все d равны нулю).

H2. [Поиск n.] Установить $n \leftarrow c_1$. Успешно завершить работу алгоритма, если $n = 0$; завершить работу алгоритма неудачей, если $d_1 \geq n > 0$.

H3. [Начало цикла по j.] Установить $i \leftarrow 1$, $t \leftarrow d_1$, $r \leftarrow c_t$ и $j \leftarrow d_n$.

H4. [Генерация нового ребра.] Установить $c_j \leftarrow c_j - 1$ и $m \leftarrow c_t$. Создать ребро n — m и установить $d_m \leftarrow d_m - 1$, $c_t \leftarrow m - 1$, $j \leftarrow j - 1$. Если $j = 0$, вернуться к шагу H2. В противном случае если $m = i$, установить $i \leftarrow r+1$, $t \leftarrow d_i$ и $r \leftarrow c_t$ (см. упр. 104); повторить шаг H4. ∎

Когда алгоритм H завершается успешно, он строит граф с интересующими нас ребрами. Но если он завершается неудачно, то как мы можем убедиться, что задача неразрешима? Ключевой факт основан на важной концепции, именуемой "мажоризация": если $d_1 \dots d_n$ и $d'_1 \dots d'_n$ являются двумя разбиениями одного и того же числа (т. е. если $d_1 \geq \cdots \geq d_n$, $d'_1 \geq \cdots \geq d'_n$ и $d_1 + \cdots + d_n = d'_1 + \cdots + d'_n$), то мы говорим, что $d_1 \dots d_n$ *мажоризирует* $d'_1 \dots d'_n$, если $d_1 + \cdots + d_k \geq d'_1 + \cdots + d'_k$ для $1 \leq k \leq n$.

Лемма M. *Если последовательность $d_1 \dots d_n$ является графической и $d_1 \dots d_n$ мажоризирует $d'_1 \dots d'_n$, то последовательность $d'_1 \dots d'_n$ также является графической.*

Доказательство. Достаточно доказать утверждение, когда $d_1 \dots d_n$ и $d'_1 \dots d'_n$ отличаются только в двух местах,

$$d'_k = d_k - [k = i] + [k = j], \qquad \text{где } i < j, \tag{55}$$

поскольку любая последовательность, мажоризируемая $d_1 \dots d_n$, может быть получена неоднократной мини-мажоризацией, аналогичной показанной. (Детально мажоризация рассматривается в упр. 7.2.1.4–55.)

Из условия (55) вытекает, что $d_i > d'_i \geq d'_{i+1} \geq d'_j > d_j$. Так что любой граф с последовательностью степеней $d_1 \dots d_n$ содержит вершину v, такую, что v — i и v +— j. Удаление ребра v — i и добавление ребра v — j дает граф с последовательностью степеней $d'_1 \dots d'_n$, как и требовалось. ▮

Следствие H. *Алгоритм H завершается успешно, если последовательность $d_1 \dots d_n$ является графической.*

Доказательство. Можно считать, что $n > 1$. Предположим, что G — произвольный граф с вершинами $\{1, \dots, n\}$ с последовательностью степеней $d_1 \dots d_n$, и пусть G' — подграф, порожденный $\{1, \dots, n-1\}$; другими словами, G' получается путем удаления вершины n и d_n ребер, связанных с ней. Последовательность степеней $d'_1 \dots d'_{n-1}$ графа G' получается из $d_1 \dots d_{n-1}$ путем уменьшения некоторых d_n элементов на 1 и сортировкой их в невозрастающем порядке. По определению последовательность $d'_1 \dots d'_{n-1}$ графическая. Новая последовательность степеней $d''_1 \dots d''_{n-1}$, получаемая с помощью стратегии, реализованной на шагах H3 и H4, задумана как мажоризируемая каждой такой последовательностью $d'_1 \dots d'_{n-1}$, поскольку уменьшает наибольшие возможные d_n элементов на 1. Таким образом, новая последовательность $d''_1 \dots d''_{n-1}$ является графической. Алгоритм H, устанавливающий $d_1 \dots d_{n-1} \leftarrow d''_1 \dots d''_{n-1}$, будет, таким образом, успешен по индукции по n. ▮

Время работы алгоритма H грубо пропорционально количеству генерируемых ребер, которое может быть порядка n^2. В упр. 105 представлен более быстрый метод, который за $O(n)$ шагов выясняет, является ли данная последовательность $d_1 \dots d_n$ графической, но сам граф при этом не строится.

Над графами. Если вершины и/или дуги графа или ориентированного графа оснащены дополнительными данными, мы называем его *сетью* (*network*). Например, каждая вершина *words*$(5757, 0, 0, 0)$ имеет связанный с ней ранг, соответствующий популярности соответствующего пятибуквенного слова. Каждая вершина *plane_lisa*$(360, 250, 15, 0, 360, 0, 250, 0, 2295000)$ имеет связанную с ней яркость пикселей между 0 и 15. Каждая дуга *board*$(8, 8, 0, 0, -2, 0, 0)$ имеет связанную с ней длину, отражающую расстояние перемещения фигуры по доске: перемещение слона из угла в угол имеет длину 7. Stanford GraphBase включает несколько генераторов, которые не упоминались выше, поскольку они используются, в первую очередь, для генерации интересных сетей, а не графов с интересной структурой.

• *miles*$(128, 0, 0, 0, 0, 127, 0)$ представляет собой сеть со 128 вершинами, соответствующими тем же городам Северной Америки, что и в графе *plane_miles*, описанном ранее. Но *miles*, в отличие от *plane_miles*, представляет собой полный граф с $\binom{128}{2}$ ребрами. Каждое ребро имеет целую длину, представляющую расстояние в милях, которое автомобиль должен был преодолеть при поездке в 1949 году из одного города в другой. Например, в сети *miles* '`Vancouver, BC`' отстоит на 3496 миль от '`West Palm Beach, FL`'.

• *econ*$(81, 0, 0, 0)$ представляет собой сеть с 81 вершиной и 4902 дугами. Ее вершины представляют отрасли экономики США, а дуги — потоки денег из одной отрасли в другую в 1985 году, измеренные в миллионах долларов. Например, значение потока от `Apparel` в `Household furniture` равно 44, а это означает, что мебельная

промышленность заплатила в этом году швейной промышленности 44 миллиона долларов. Суммы потоков, входящих в каждую вершину, равны суммам исходящих потоков. Дуга имеется только в том случае, если поток ненулевой. Специальная вершина под названием `Users` получает потоки, которые представляют общую потребность в продукте; некоторые из этих потоков отрицательны в связи со способом рассмотрения правительственными экономистами импортируемых товаров.

• *games*(120, 0, 0, 0, 0, 0, 128, 0) представляет собой сеть со 120 вершинами и 1276 дугами. Вершины этого графа представляют футбольные команды американских колледжей и университетов. Дуги проходят между командами, игравшими одна с другой во время сезона 1990 года, и помечены числами, соответствующими выигранным очкам. Например, дуга `Stanford`⟶`California` имеет значение 27, а дуга `California`⟶`Stanford` имеет значение 25, поскольку 17 ноября 1990 года Cardinal из Станфорда победили Golden Bears из Беркли со счетом 27:25.

• *risc*(16) представляет собой сеть совершенно иного вида. В ней 3240 вершин и 7878 дуг, определяющих *ориентированный ациклический граф*, т. е. ориентированный граф, не содержащий ориентированных циклов. Вершины представляют логические элементы с логическими значениями; дуга наподобие `Z45`⟶`R0:7~` означает, что значение логического элемента `Z45` является входным для элемента `R0:7~`. Каждый элемент имеет код типа (AND, OR, XOR, NOT, защелка или внешний ввод); каждая дуга имеет длину, описывающую величину задержки. Сеть содержит полную логику небольшой микросхемы RISC, способной подчиняться простым командам, определяемым шестнадцатью регистрами размером 16 бит каждый.

Полное детальное описание всех генераторов SGB можно найти в книге автора *The Stanford GraphBase* (New York: ACM Press, 1994) вместе с десятками небольших демонстрационных программ, поясняющих, как работать с генерируемыми графами и сетями. Например, программа LADDERS показывает, как найти кратчайший путь между двумя пятибуквенными словами. Программа TAKE_RISC демонстрирует, как нанокомпьютер выполняет свою работу, имитируя работу сети, построенной из логических элементов *risc*(16).

Гиперграфы. Графы и сети могут быть просто обворожительны, но это ни в коем случае не конец истории. Множество важных комбинаторных алгоритмов разработано для работы с *гиперграфами*, которые являются более обобщенными по сравнению с графами, поскольку их ребра могут быть *произвольными* подмножествами вершин.

Например, у нас может быть семь вершин, идентифицируемых ненулевыми бинарными строками $v = a_1a_2a_3$, вместе с семью ребрами, идентифицируемыми ненулевыми бинарными строками в квадратных скобках $e = [b_1b_2b_3]$, где $v \in e$ тогда и только тогда, когда $(a_1b_1 + a_2b_2 + a_3b_3) \bmod 2 = 0$. Каждое из этих ребер содержит ровно три вершины:

$$\begin{aligned} [001] = \{010, 100, 110\};\quad [010] = \{001, 100, 101\};\quad [011] = \{011, 100, 111\};\\ [100] = \{001, 010, 011\};\quad [101] = \{010, 101, 111\};\\ [110] = \{001, 110, 111\};\quad [111] = \{011, 101, 110\}. \end{aligned} \tag{56}$$

В силу симметрии каждая вершина принадлежит ровно трем ребрам. Ребра, содержащие три и более вершин, иногда называют гиперребрами, чтобы отличать их от ребер обычных графов. Но их можно называть и просто ребрами.

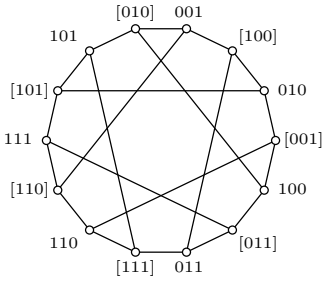
Гиперграф называется *r-однородным*, если каждое ребро содержит ровно r вершин. Таким образом, (56) является 3-однородным гиперграфом, а 2-однородный гиперграф представляет собой ни что иное, как обычный граф. Полный r-однородный гиперграф $K_n^{(r)}$ имеет n вершин и $\binom{n}{r}$ ребер.

Большинство основных концепций теории графов естественным образом может быть распространено на гиперграфы. Например, если $H = (V, E)$ представляет собой гиперграф и если $U \subseteq V$, то подгиперграф $H \mid U$, порожденный U, имеет ребра $\{e \mid e \in E \text{ и } e \subseteq U\}$. Дополнение $\overline{H}$ r-однородного гиперграфа имеет ребра $K_n^{(r)}$, которые не являются ребрами H. k-раскрашиванием гиперграфа называется назначение цветов его вершинам таким образом, чтобы ни одно ребро не было монохромным. И т. д.

Гиперграфы могут иметь много других имен, поскольку одни и те же свойства могут быть сформулированы многими различными способами. Например, каждый гиперграф $H = (V, E)$ представляет собой, по сути, *семейство множеств*, поскольку каждое ребро является подмножеством V. 3-однородный гиперграф называется также *системой троек*. Гиперграф эквивалентен также матрице B, состоящей из нулей и единиц, в которой имеется по одной строке для каждой вершины v и по одному столбцу для каждого ребра e; на пересечении строки v и столбца e этой матрицы находится значение $b_{ve} = [v \in e]$. Матрица B называется *матрицей инциденций* H, и мы говорим, что вершина "v инцидентна e", когда $v \in e$. Кроме того, гиперграф эквивалентен *двудольному графу* с множеством вершин $V \cup E$ и с ребром v — e в случае инцидентности v с e. Гиперграф называется *связным* тогда и только тогда, когда связным является соответствующий двудольный граф. *Цикл* длиной k в гиперграфе определяется как цикл длиной $2k$ в соответствующем двудольном графе.

Например, гиперграф (56) может быть определен эквивалентной матрицей инциденций или эквивалентным двудольным графом следующим образом.

$$\begin{array}{c} \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{array} \begin{array}{c} \begin{array}{ccccccc} [001] & [010] & [011] & [100] & [101] & [110] & [111] \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array} \qquad \text{(57)}$$



Он содержит 28 циклов длиной 3, таких как

$$[101] — 101 — [010] — 001 — [100] — 010 — [101]. \qquad (58)$$

Гиперграф H^T, *дуальный* гиперграфу H, получается путем взаимообмена ролей вершин и ребер, но с неизменным отношением инцидентности. Другими словами,

он соответствует транспонированной матрице инциденций. Обратите внимание, что, например, дуальным к r-регулярному графу является r-однородный гиперграф.

Матрицы инциденций и двудольные графы могут соответствовать гиперграфам, в которых некоторые ребра встречаются больше одного раза, поскольку различные столбцы матрицы могут быть одинаковыми. Если гиперграф $H = (V, E)$ не имеет повторяющихся ребер, он соответствует еще одному комбинаторному объекту, а именно *булевой функции* (логической функции). Ибо если, скажем, множество вершин V равно $\{1, 2, \ldots, n\}$, функция

$$h(x_1, x_2, \ldots, x_n) \;=\; \big[\{j \mid x_j = 1\} \in E\big] \tag{59}$$

характеризует ребра H. Например, булева формула

$$\begin{aligned}(x_1 \oplus x_2 \oplus x_3) \wedge (x_2 \oplus x_4 \oplus x_6) \wedge (x_3 \oplus x_4 \oplus x_7)\\ \wedge (x_3 \oplus x_5 \oplus x_6) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4)\end{aligned} \tag{60}$$

представляет собой другой способ описания гиперграфа (56) и (57).

Тот факт, что комбинаторные объекты можно рассматривать столькими разными способами, может показаться невероятным. Но это исключительно полезно, поскольку предполагает различные способы решения эквивалентных задач. Когда мы рассматриваем задачу с разных точек зрения, мозг естественным образом думает о разных способах ее решения. Иногда озарение приходит, когда мы думаем о том, как работать со строками и столбцами матрицы. В другой раз прорыв происходит, когда мы представляем вершины и пути или при визуализации кластеров точек в пространстве. Возможно, что наилучшим способом окажется применение булевой алгебры. Если мы застряли на одном из представлений, на помощь может прийти другое.

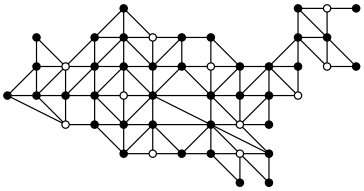
Покрытие и независимость. Если $H = (V, E)$ является графом или гиперграфом, множество вершин U называется *покрытием* H, если каждое ребро содержит как минимум один из членов U. Множество вершин W называется *независимым* (или устойчивым) в H, если в нем не содержится полностью ни одно ребро.

С точки зрения матрицы инциденций покрытие является множеством строк, сумма которых не равна нулю в каждом столбце. В частном случае, когда H является графом, каждый столбец матрицы содержит только две единицы; следовательно, независимое множество в графе соответствует множеству взаимно ортогональных строк, т. е. множеству, в котором скалярное произведение любых двух разных строк равно нулю.

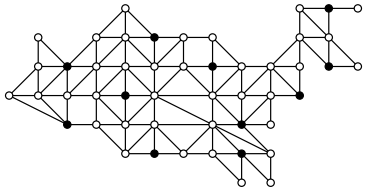
Эти концепции являются оборотными сторонами одной медали. Если U покрывает H, то $W = V \setminus U$ независимо в H; и наоборот, если W независимо в H, то $U = V \setminus W$ покрывает H. Оба утверждения эквивалентны утверждению, что порожденный гиперграф $H \mid W$ не имеет ребер.

Это дуальное отношение между покрытием и независимостью, которое, похоже, впервые было открыто Клодом Бергом (Claude Berge) [*Proc. National Acad. Sci.* **43** (1957), 842–844], несколько парадоксально. Хотя оно логически очевидно и легко проверяемо, на интуитивном уровне оно кажется удивительным. Когда мы смотрим на граф и пытаемся найти большое независимое множество, обычно мы мыслим совершенно иначе, чем когда смотрим на тот же граф и пытаемся найти малое вершинное покрытие; но цели в обоих случаях одинаковы.

Покрывающее множество U *минимально* (*minimal*), если $U \setminus u$ не является покрытием для всех $u \in U$. Аналогично независимое множество W является *максимальным* (*maximal*), если $W \cup w$ не является независимым для всех $w \notin W$. Вот, например, как выглядит минимальное покрытие 49-вершинного графа США (17) и соответствующее ему максимальное независимое множество.



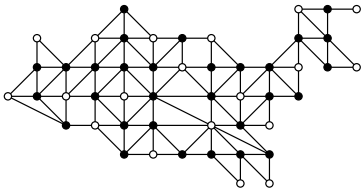
Минимальное вершинное покрытие
из 38 вершин



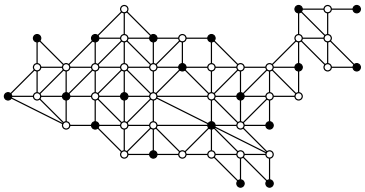
Максимальное независимое множество
из 11 вершин

(61)

Покрытие называется *наименьшим* (*minimum*), если оно имеет наименьший возможный размер, а независимое множество называется *наибольшим* (*maximum*), если оно имеет наибольший возможный размер. Например, в случае графа (17) можно добиться лучшего результата, чем представляет собой (61).



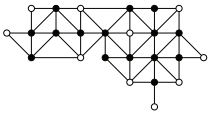
Наименьшее вершинное покрытие
из 30 вершин



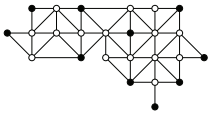
Наибольшее независимое множество
из 19 вершин

(62)

От переводчика. Для графа Украины (17, у) соответствующие результаты имеют следующий вид.



Наименьшее вершинное покрытие
из 15 вершин



Наибольшее независимое множество
из 10 вершин

Обратите внимание на тонкую разницу между “минимальным” и “наименьшим”: в общем случае люди, работающие с комбинаторными алгоритмами, в отличие от словаря английского языка, используют слова на ‘-al’ (наподобие “minimal” или “optimal”) для комбинаторных конфигураций, которые являются наилучшими *локально*, в том смысле, что минимальные изменения их не улучшают. Слова же на ‘-um’, такие как “minimum” или “optimum”, оставлены для конфигураций, наилучших *глобально*, т. е. при рассмотрении всех возможностей.* Легко найти решение любой задачи оптимизации, являющееся всего лишь оптимальным в слабом локальном смысле, просто забираясь на вершину ближайшего холма. Но обычно

* К счастью, русский язык позволяет использовать по-разному звучащие слова — скажем, “оптимальный” и “наилучший”. — *Примеч. пер.*

гораздо труднее найти наилучшее решение, истинный оптимум, который представляет собой вершину самой высокой горы. Например, из раздела 7.9 вы узнаете, что задача поиска наибольшего независимого множества данного графа принадлежит классу сложных задач, который называется *NP-полным*.

Даже если задача является NP-полной, не стоит опускать руки. Мы рассмотрим методы поиска наименьших покрытий в нескольких частях данной главы, и эти методы неплохо работают для задач небольшого размера; наилучшее решение (62) найдено менее чем за секунду, после исследования очень небольшой доли всех 2^{49} возможных вариантов. Кроме того, нередко частные случаи NP-полных задач оказываются проще, чем задачи в общем случае. В разделе 7.5.1 мы увидим, что можно быстро найти наименьшее вершинное покрытие для любого двудольного графа или для любого гиперграфа, являющегося дуальным к графу. А в разделе 7.5.5 мы изучим эффективные способы наибольшего *паросочетания*, которое представляет собой наибольшее независимое множество в реберном графе данного графа.

Задача максимизации размера независимого множества возникает достаточно часто, чтобы получить собственное обозначение: если H представляет собой некоторый гиперграф, то число

$$\alpha(H) \;=\; \max\big\{|W| \;\big|\; W \text{ является независимым множеством вершин в } H\big\} \qquad (63)$$

называется *числом независимости* (или числом устойчивости) гиперграфа H. Аналогично

$$\chi(H) \;=\; \min\{k \mid H \text{ является } k\text{-раскрашиваемым}\} \qquad (64)$$

называется *хроматическим числом* H. Обратите внимание, что $\chi(H)$ представляет собой размер наименьшего покрытия H независимыми множествами, поскольку вершины, получающие некоторый определенный цвет, должны быть независимыми в соответствии с нашими определениями.

Эти определения $\alpha(H)$ и $\chi(H)$ применимы, в частности, в случае, когда H представляет собой обычный граф, но, конечно, в этих случаях мы обычно пишем $\alpha(G)$ и $\chi(G)$. Графы имеют другое важное число, именуемое их *кликовым числом*,

$$\omega(G) \;=\; \max\big\{|X| \;\big|\; X \text{ является кликой в } G\big\}, \qquad (65)$$

где "клика" представляет собой множество взаимно смежных вершин. Ясно, что

$$\omega(G) \;=\; \alpha(\overline{G}), \qquad (66)$$

поскольку клика в G является независимым множеством в дополнении графа. Аналогично мы можем видеть, что $\chi(\overline{G})$ является наименьшим размером "кликового покрытия", которое представляет собой множество клик, которое покрывает в точности все вершины.

Несколько экземпляров "задач точного покрытия" упоминались ранее в этом разделе без точного пояснения, что же в действительности означает такая задача. Наконец, мы созрели для определения: для заданной матрицы инциденций или гиперграфа H *точное покрытие* H представляет собой множество строк, сумма которых равна $(1\,1\,\ldots\,1)$. Другими словами, точное покрытие представляет собой множество вершин, которые соединяются с каждым гиперребром ровно по одному разу; обычное покрытие требует соединения с каждым гиперребром *как минимум* по одному разу.

Упражнения

1. [*25*] Предположим, что $n = 4m-1$. Постройте размещение пар Лэнгфорда для чисел $\{1,1,\ldots,n,n\}$, обладающее тем свойством, что мы также получим решение для $n = 4m$, заменяя первое '$2m{-}1$' на '$4m$' и добавляя '$2m{-}1\ 4m$' справа. *Указание*: поместите слева $m-1$ четных чисел $4m{-}4$, $4m{-}6$, ..., $2m$.

2. [*20*] Для каких n можно расположить числа $\{0,0,1,1,\ldots,n{-}1,n{-}1\}$ в виде пар Лэнгфорда?

3. [*22*] Предположим, что мы разместили числа $\{0,0,1,1,\ldots,n{-}1,n{-}1\}$ по *окружности*, а не на прямой, с расстоянием k между двумя k. Получим ли мы решение, существенно отличное от решения упр. 2?

4. [*M20*] (T. Сколем (T. Skolem), 1957.) Покажите, что рассматривающаяся в упр. 1.2.8–36 строка Фибоначчи $S_\infty = babbababbabba\ldots$ приводит непосредственно к бесконечной последовательности 0012132453674 ... пар Лэнгфорда для множества *всех* неотрицательных целых чисел, если просто независимо заменять a и b слева направо на 0, 1, 2 и т. д.

▶ **5.** [*HM22*] Какова вероятность того, что при случайной перестановке чисел $\{1,1,2,2,\ldots,n,n\}$ два k будут находиться ровно на расстоянии k позиций одно от другого, для заданного k? Воспользуйтесь этой формулой для оценки величины чисел Лэнгфорда L_n в (1).

▶ **6.** [*M28*] (M. Годфри (M. Godfrey), 2002.) Пусть

$$f(x_1,\ldots,x_{2n}) = \prod_{k=1}^{n}(x_k x_{n+k}\sum_{j=1}^{2n-k-1} x_j x_{j+k+1}).$$

а) Докажите, что $\sum_{x_1,\ldots,x_{2n}\in\{-1,+1\}} f(x_1,\ldots,x_{2n}) = 2^{2n+1}L_n$.

б) Поясните, как вычислить эту сумму за $O(4^n n)$ шагов. Сколько битов точности потребуют при этом арифметические операции?

в) Усильте результат в восемь раз, используя тождества

$$f(x_1,\ldots,x_{2n}) = f(-x_1,\ldots,-x_{2n}) = f(x_{2n},\ldots,x_1) = f(x_1,-x_2,\ldots,x_{2n-1},-x_{2n}).$$

7. [*M22*] Докажите, что любое размещение Лэнгфорда чисел $\{1,1,\ldots,16,16\}$ должно иметь при чтении слева направо в некоторой точке семь незавершенных пар.

8. [*23*] Простейшая последовательность Лэнгфорда не только хорошо сбалансирована, но и *планарна*, в том смысле, что ее пары могут быть соединены без пересечения линий, как в (2): $\overset{\lceil\ \ \ \ \ \ \rceil}{2\,3\,1\,2\,1\,3}$. Найдите все планарные последовательности Лэнгфорда для $n \le 8$.

9. [*24*] (*Тройки Лэнгфорда.*) Сколькими способами можно разместить числа $\{1,1,1,2,2,2,\ldots,9,9,9\}$ в строке так, чтобы последовательные k находились на расстоянии k друг от друга для $1 \le k \le 9$?

10. [*M20*] Поясните, как построить *магический квадрат* непосредственно по рис. 1. (Преобразуйте каждую карту в число от 1 до 16 так, чтобы суммы в каждой строке, каждом столбце и каждой диагонали были равны 34.)

11. [*20*] Распространите (5) на "иврито-греко-латинский" квадрат, добавляя по одной из букв $\{\text{א},\text{ב},\text{ג},\text{ד}\}$ к двухбуквенным строкам в каждой ячейке. Никакие пары букв (латинская, греческая), (латинская, иврит) и (греческая, иврит) не должны встречаться более чем в одном месте.

▶ **12.** [*M21*] (Л. Эйлер (L. Euler).) Пусть $L_{ij} = (i+j) \bmod n$ для $0 \le i,j < n$ представляет собой таблицу сложения для целых чисел по модулю n. Докажите, что латинский квадрат, ортогональный L, существует тогда и только тогда, когда n нечетно.

13. [*M25*] Квадрат 10×10 можно разделить на четыре четверти размером по 5×5. Латинский квадрат 10×10, образованный цифрами $\{0, 1, \ldots, 9\}$, имеет k "нарушителей", если его верхняя левая четверть имеет ровно k элементов ≥ 5. (См. в упр. 14, (д) пример с $k = 3$.) Докажите, что квадрат не имеет ортогонального к нему квадрата, если только в нем нет как минимум трех нарушителей.

14. [*29*] Найдите все квадраты, ортогональные следующим латинским квадратам:

(а)	(б)	(в)	(г)	(д)
3145926870	2718459036	0572164938	1680397425	7823456019
2819763504	0287135649	6051298473	8346512097	8234067195
9452307168	7524093168	4867039215	9805761342	2340178956
6208451793	1435962780	1439807652	2754689130	3401289567
8364095217 .	6390718425 .	8324756091 .	0538976214 .	4012395678
5981274036 '	4069271853 '	7203941586 '	4963820571 '	5678912340 '
4627530981	3102684597	5610473829	7192034658	6789523401
0576148329	9871546302	9148625307	6219405783	0195634782
1730689452	8956307214	2795380164	3471258906	1956740823
7093812645	5643820971	3986512740	5027143869	9567801234

15. [*50*] Найдите три латинских квадрата 10×10, взаимно ортогональных друг другу.

16. [*48*] (Г. Д. Райзер (H. J. Ryser), 1967.) Латинский квадрат называется "порядка n", если он имеет n строк, n столбцов и n символов. Каждый ли латинский квадрат нечетного порядка имеет секущую?

17. [*25*] Пусть L — латинский квадрат с элементами L_{ij} для $0 \leq i, j < n$. Покажите, что задачи (а) поиска всех секущих L и (б) поиска всех ортогональных L квадратов являются частными случаями общей задачи точного покрытия.

18. [*M26*] Строка $x_1 x_2 \ldots x_N$ называется n-арной, если каждый элемент x_j принадлежит множеству $\{0, 1, \ldots, n-1\}$ n-арных цифр. Две строки, $x_1 x_2 \ldots x_N$ и $y_1 y_2 \ldots y_N$, называются *ортогональными*, если N пар (x_j, y_j) различны для $1 \leq j \leq N$. (Следовательно, две n-арные строки не могут быть ортогональны, если их длина N превышает n^2.) n-арная матрица с m строками и n^2 столбцами, строки которых ортогональны одна другой, называется *ортогональным массивом* порядка n и глубиной m.

Найдите соответствие между ортогональными массивами глубиной m и списками из $m - 2$ взаимно ортогональных латинских квадратов. Какой ортогональный массив соответствует упр. 11?

▸ **19.** [*M25*] Продолжая выполнение упр. 18, докажите, что ортогональный массив порядка $n > 1$ и глубиной m возможен, только если $m \leq n + 1$. Покажите, что этот верхний предел достижим, когда n является простым числом p. Приведите пример для $p = 5$.

20. [*HM20*] Покажите, что если каждый элемент k в ортогональном массиве заменить на $e^{2\pi ki/n}$, то его строки станут ортогональными векторами в обычном смысле (их скалярное произведение будет равно нулю).

▸ **21.** [*M21*] *Геометрическая сеть* представляет собой систему точек и прямых, подчиняющуюся трем аксиомам.

i) Каждая прямая есть множество точек.

ii) Различные прямые имеют не более одной общей точки.

iii) Если p является точкой, а L — прямая, такая, что $p \notin L$, то существует ровно одна прямая M, такая, что $p \in M$ и $L \cap M = \emptyset$. Если $L \cap M = \emptyset$, мы говорим, что L *параллельна* M, и записываем $L \parallel M$.

a) Докажите, что прямые геометрической сети могут быть разбиты на классы эквивалентности, когда две прямые находятся в одном классе тогда и только тогда, когда они совпадают или параллельны.

б) Покажите, что если имеется как минимум два класса параллельных прямых, то каждая прямая содержит то же количество точек, что и другие прямые в ее классе.

в) Кроме того, если имеется как минимум три класса, существуют числа m и n, такие, что каждая точка принадлежит ровно m прямым, и все прямые содержат ровно n точек.

▸ **22.** [*M22*] Покажите, что каждый ортогональный массив можно рассматривать как геометрическую сеть. Справедливо ли обратное утверждение?

23. [*M23*] (*Коды с коррекцией ошибок.*) "Расстоянием Хэмминга" $d(x,y)$ между двумя строками, $x = x_1 \ldots x_N$ и $y = y_1 \ldots y_N$, является количество позиций j, в которых $x_j \neq y_j$. *b-арный код с n информационными и r контрольными разрядами* представляет собой множество $C(b,n,r)$, состоящее из b^n строк $x = x_1 \ldots x_{n+r}$, где $0 \le x_j < b$ для $1 \le j \le n+r$. При передаче кодового слова x и получении сообщения y величина $d(x,y)$ равна количеству ошибок передачи. Код называется *исправляющим t ошибок*, если можно реконструировать значение x при полученном сообщении y, таком, что $d(x,y) \le t$. *Расстоянием* кода является минимальное значение $d(x,x')$, взятое по всем парам кодовых слов $x \neq x'$.

а) Докажите, что код является исправляющим t ошибок тогда и только тогда, когда его расстояние превышает $2t$.

б) Докажите, что исправляющий одну ошибку b-арный код с двумя информационными и двумя контрольными разрядами эквивалентен паре ортогональных латинских квадратов порядка b.

в) Кроме того, код $C(b,2,r)$ с расстоянием $r+1$ эквивалентен множеству r взаимно ортогональных латинских квадратов порядка b.

▸ **24.** [*M30*] Геометрическая сеть с N точками и R прямыми естественным образом приводит к бинарному коду $C(2,N,R)$ с кодовыми словами $x_1 \ldots x_N x_{N+1} \ldots x_{N+R}$, определяемыми битами четности

$$x_{N+k} = f_k(x_1,\ldots,x_N) = \big(\sum\{x_j \mid \text{точка } j \text{ лежит на прямой } k\}\big) \bmod 2.$$

а) Докажите, что этот код имеет расстояние $m+1$, если сеть имеет m классов параллельных прямых.

б) Найдите эффективный способ исправления до t ошибок в этом коде, в предположении, что $m = 2t$. Проиллюстрируйте процесс декодирования для случая $N = 25$, $R = 30$, $t = 3$.

25. [*27*] Найдите латинский квадрат, строки и столбцы которого представляют собой пятибуквенные слова. (Для выполнения этого упражнения вам понадобятся большие словари.)

▸ **26.** [*25*] Составьте имеющее смысл английское предложение, состоящее только из пятибуквенных слов.

27. [*20*] Сколько SGB-слов содержат ровно k различных букв для $1 \le k \le 5$?

28. [*20*] Имеются ли пары SGB-слов, рассматриваемых как векторы, все соответствующие компоненты которых отличаются на ± 1?

29. [*20*] Найдите все SGB-слова, являющиеся *палиндромами* (остающимися неизменными при чтении справа налево) или зеркальными парами (наподобие `regal lager`).

▸ **30.** [*20*] Буквы слова `first` находятся в алфавитном порядке слева направо. Какое из таких пятибуквенных слов является лексикографически *первым*? Какое последним?

31. [*21*] (К. Мак-Манус (C. McManus).) Найдите все множества из трех SGB-слов, находящихся в арифметической прогрессии и не имеющих общих букв ни в одной фиксированной позиции. (Одним таким примером является {`power`, `slugs`, `visit`}.)

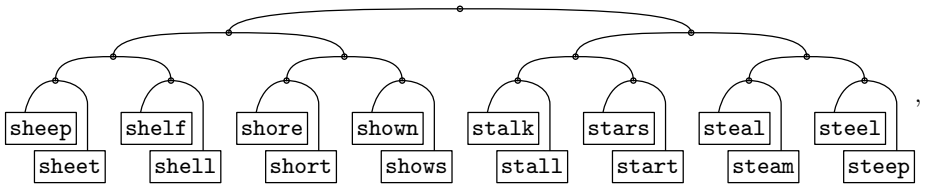
32. [*23*] Содержит ли английский язык хотя бы одно десятибуквенное слово $a_0a_1\ldots a_9$, для которого и $a_0a_2a_4a_6a_8$, и $a_1a_3a_5a_7a_9$ являются SGB-словами?

33. [*20*] (Скот Моррис (Scot Morris).) Завершите следующий список из 26 интересных SGB-слов:

about, bacon, faced, under, chief, ..., pizza.

▶ **34.** [*21*] Для каждого SGB-слова, не включающего букву y, получим 5-битовое бинарное число, заменяя гласные {a,e,i,o,u} единицами, а остальные буквы — нулями. Каковы наиболее распространенные слова для каждых из 32 бинарных значений?

▶ **35.** [*26*] Шестнадцать тщательно отобранных элементов WORDS(1000) приводят к ветвящейся структуре



которая представляет собой полный бинарный луч слов, начинающихся с буквы s. Однако такой структуры для слов, начинающихся с буквы a, не существует, даже если рассмотреть всю коллекцию WORDS(5757).

Какие буквы алфавита могут быть использованы в качестве начальной буквы шестнадцати слов, образующих полный бинарный луч в WORDS(n) для заданного n?

36. [*M17*] Поясните симметрии, наблюдающиеся в кубе из слов (10). Покажите также, что еще два таких куба можно получить изменением только двух слов — {stove,event}.

37. [*20*] Какие вершины графа *words*(5757, 0, 0, 0) имеют максимальную степень?

38. [*22*] Используя правило ориентированного графа из (14), замените tears на smile за три шага, *не прибегая к помощи компьютера.*

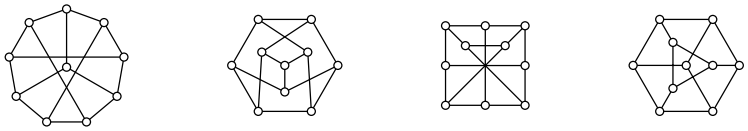
39. [*M00*] Является ли $G \setminus e$ порожденным подграфом G? Является ли он остовным подграфом?

40. [*M15*] Сколько (а) остовных и (б) порожденных подграфов имеет граф $G = (V, E)$, когда $|V| = n$ и $|E| = e$?

41. [*M10*] Для каких целых чисел n мы имеем: (а) $K_n = P_n$? (б) $K_n = C_n$?

42. [*15*] (Д. Г. Лемер (D. H. Lehmer).) Пусть G — граф с 13 вершинами, каждая вершина которого имеет степень 5. Сформулируйте нетривиальное утверждение о G.

43. [*23*] Является ли какой-либо из приведенных графов графом Петерсена?



44. [*M23*] Сколько симметрий имеет граф Чватала? (См. рис. 2, (е).)

45. [*20*] Найдите простой способ 4-раскрашивания планарного графа (17). Хватит ли вам трех красок?

46. [*M25*] Пусть G представляет собой граф с $n \geq 3$ вершинами, определяемыми планарной диаграммой, и являющийся "максимальным" в том смысле, что между несмежными вершинами нельзя провести никаких дополнительных линий без пересечения уже существующих.

а) Докажите, что диаграмма разбивает плоскость на области, каждая из которых имеет на своих границах ровно три вершины. (Одной из этих областей являются все точки, лежащие за пределами диаграммы.)

б) Следовательно, G имеет ровно $3n - 6$ ребер.

47. [*M22*] Докажите, что полный биграф $K_{3,3}$ не является планарным.

48. [*M25*] Завершите доказательство теоремы B, показав, что описанная процедура никогда не дает один и тот же цвет двум соседним вершинам.

49. [*18*] Начертите диаграммы для всех кубических графов не более чем с 6 вершинами.

50. [*M24*] Найдите все двудольные графы, которые могут быть 3-раскрашены ровно 24 способами.

▶ **51.** [*M22*] Для заданной геометрической сети, такой как описанная в упр. 21, постройте двудольный граф, вершинами которого являются точки p и прямые L сети, причем $p \mathrel{—} L$ тогда и только тогда, когда $p \in L$. Чему равен *обхват* этого графа?

52. [*M16*] Найдите простое неравенство, которое связывает диаметр и его обхват. (Насколько малым может быть диаметр при большом обхвате?)

53. [*15*] Какое из слов `world` и `happy` принадлежит гигантскому компоненту графа $words(5757, 0, 0, 0)$?

▶ **54.** [*21*] Вот список из 49 почтовых кодов в графе (17) в алфавитном порядке: `AL`, `AR`, `AZ`, `CA`, `CO`, `CT`, `DC`, `DE`, `FL`, `GA`, `IA`, `ID`, `IL`, `IN`, `KS`, `KY`, `LA`, `MA`, `MD`, `ME`, `MI`, `MN`, `MO`, `MS`, `MT`, `NC`, `ND`, `NE`, `NH`, `NJ`, `NM`, `NV`, `NY`, `OH`, `OK`, `OR`, `PA`, `RI`, `SC`, `SD`, `TN`, `TX`, `UT`, `VA`, `VT`, `WA`, `WI`, `WV`, `WY`.

а) Предположим, что мы рассматриваем два штата как соседние, если их почтовые коды совпадают в одной букве (т. е. `AL` — `AR` — `OR` — `OH` и т. д.). Каковы компоненты данного графа?

б) Образуем ориентированный граф с `XY` $\longrightarrow$ `YZ` (например, `AL` $\longrightarrow$ `LA` $\longrightarrow$ `AR` и т. д.). Что собой представляют *сильно связанные компоненты* этого ориентированного графа? (См. раздел 2.3.4.2.)

в) В США, помимо почтовых кодов в (17), имеются дополнительные почтовые коды, а именно `AA`, `AE`, `AK`, `AP`, `AS`, `FM`, `GU`, `HI`, `MH`, `MP`, `PW`, `PR`, `VI`. Рассмотрите заново вопрос из п. (б) с использованием всех 62 почтовых кодов.

55. [*M20*] Сколько ребер имеется в полном k-дольном графе $K_{n_1,\ldots,n_k}$?

▶ **56.** [*M10*] Истинно или ложно утверждение "мультиграф является графом тогда и только тогда, когда соответствующий ориентированный граф является простым"?

57. [*M10*] Истинно или ложно утверждение "вершины u и v находятся в одном и том же связном компоненте ориентированного графа тогда и только тогда, когда либо $d(u, v) < \infty$, либо $d(v, u) < \infty$"?

58. [*M17*] Опишите все (а) графы и (б) мультиграфы, являющиеся регулярными степени 2.

▶ **59.** [*M23*] *Турнир* порядка n представляет собой ориентированный граф из n вершин, имеющий ровно $\binom{n}{2}$ дуг, либо $u \longrightarrow v$, либо $v \longrightarrow u$ для каждой пары различных вершин $\{u, v\}$.

а) Докажите, что каждый турнир содержит ориентированный остовный путь $v_1 \longrightarrow \cdots \longrightarrow v_n$.

б) Рассмотрим турнир на вершинах $\{0, 1, 2, 3, 4\}$, в котором $u \longrightarrow v$ тогда и только тогда, когда $(u - v) \bmod 5 \geq 3$. Сколько ориентированных остовных путей он имеет?

в) Является ли $K_n^{\rightarrow}$ единственным турниром порядка n, который имеет единственный ориентированный остовный путь?

▸ **60.** [*M22*] Пусть u — вершина с наибольшей исходящей степенью в турнире и пусть v — любая другая вершина. Докажите, что $d(u,v) \le 2$.

61. [*M16*] Постройте ориентированный граф, имеющий k обходов длиной k из вершины 1 в вершину 2.

62. [*M21*] *Перестановочный ориентированный граф* представляет собой ориентированный граф, каждая вершина которого имеет входящую и исходящую степени, равные 1; следовательно, его компонентами являются ориентированные циклы. Если он имеет n вершин и k компонентов, то мы назовем его *четным* при четном $n-k$ и *нечетным* при $n-k$ нечетном.

а) Пусть G — ориентированный граф с матрицей смежности A. Докажите, что количество остовных перестановочных ориентированных графов G равно $\operatorname{per} A$ (перманенту A).

б) Интерпретируйте определитель $\det A$ в терминах остовных перестановочных ориентированных графов.

63. [*M23*] Пусть G — граф с обхватом g, каждая вершина которого имеет как минимум d соседей. Докажите, что G имеет как минимум N вершин, где

$$N = \begin{cases} 1 + \sum_{0\le k<t} d(d-1)^k, & \text{если } g = 2t+1; \\ 1 + (d-1)^t + \sum_{0\le k<t} d(d-1)^k, & \text{если } g = 2t+2. \end{cases}$$

▸ **64.** [*M21*] Продолжая выполнять упр. 63, покажите, что имеется *единственный* граф с обхватом 4, минимальной степенью d и порядком $2d$ для каждого $d \ge 2$.

▸ **65.** [*HM31*] Предположим, что граф G имеет обхват 5, минимальную степень d и $N = d^2+1$ вершин.

а) Докажите, что матрица смежности A графа G удовлетворяет уравнению $A^2 + A = (d-1)I + J$.

б) Поскольку A — симметричная матрица, она имеет N ортогональных собственных векторов x_j с соответствующими собственными значениями λ_j, такими, что $Ax_j = \lambda_j x_j$ для $1 \le j \le N$. Докажите, что каждое λ_j равно либо d, либо $(-1 \pm \sqrt{4d-3})/2$.

в) Покажите, что если $\sqrt{4d-3}$ иррационально, то $d = 2$. *Указание*: $\lambda_1 + \cdots + \lambda_N = \operatorname{trace}(A) = 0$.

г) А если $\sqrt{4d-3}$ рационально, то $d \in \{3, 7, 57\}$.

66. [*M30*] Продолжая выполнять упр. 65, постройте такой граф для $d = 7$.

67. [*M48*] Существует ли регулярный граф со степенью 57, порядком 3250 и обхватом 5?

68. [*M20*] Сколько различных матриц смежности имеет граф G с n вершинами?

▸ **69.** [*20*] Расширяя (31), поясните, как вычислить и исходящую степень `ODEG(`v`)`, и входящую степень `IDEG(`v`)` для *всех* вершин v графа, представленного в SGB-формате.

▸ **70.** [*M20*] Как часто выполняется каждый шаг алгоритма B, когда этот алгоритм успешно выполняет 2-раскрашивание графа с m дугами и n вершинами?

71. [*26*] Реализуйте алгоритм B для компьютера `MMIX` на языке ассемблера `MMIXAL`. Считайте, что в момент начала работы вашей программы регистр `v0` указывает на узел первой вершины, а регистр `n` содержит количество вершин.

▸ **72.** [*M22*] Когда `COLOR(`v`)` установлен на шаге B6, назовем u *родителем* v; но когда `COLOR(`w`)` установлен на шаге B3, будем говорить, что w не имеет родителя. Определим (включительно) *предков* вершины v рекурсивно как v вместе с предками родителя v (если таковой имеется).

а) Докажите, что если v в процессе работы алгоритма B находится в стеке ниже u, то родителем v является предок u.

б) Кроме того, если COLOR(v) = COLOR(u) на шаге B6, v находится в данный момент в стеке.

в) Воспользуйтесь этими фактами для расширения алгоритма B таким образом, чтобы, если заданный граф не является двудольным, выводились имена вершин цикла нечетной длины.

73. [*15*] Какое другое имя имеет граф *random_graph*$(10, 45, 0, 0, 0, 0, 0, 0, 0, 0)$?

74. [*21*] Какая вершина *roget*$(1022, 0, 0, 0)$ имеет наибольшую исходящую степень?

75. [*22*] Генератор графов SGB *board*$(n_1, n_2, n_3, n_4, p, w, o)$ создает граф, вершинами которого являются t-мерные целочисленные векторы $(x_1, \ldots, x_t)$ для $0 \le x_i < b_i$, определяемые первыми четырьмя параметрами (n_1, n_2, n_3, n_4) следующим образом. Установим $n_5 \leftarrow 0$, и пусть значение $j \ge 0$ — минимальное такое, что $n_{j+1} \le 0$. Если $j = 0$, установим $b_1 \leftarrow b_2 \leftarrow 8$ и $t \leftarrow 2$; это стандартная шахматная доска размером 8×8. В противном случае если $n_{j+1} = 0$, установим $b_i \leftarrow n_i$ для $1 \le i \le j$ и $t \leftarrow j$. Наконец, если $n_{j+1} < 0$, установим $t \leftarrow |n_{j+1}|$ и установим b_i равным i-му элементу периодической последовательности $(n_1, \ldots, n_j, n_1, \ldots, n_j, n_1, \ldots)$. (Например, значения $(n_1, n_2, n_3, n_4) = (2, 3, 5, -7)$ дают семимерную шахматную доску с $(b_1, \ldots, b_7) = (2, 3, 5, 2, 3, 5, 2)$, следовательно, граф имеет $2{\cdot}3{\cdot}5{\cdot}2{\cdot}3{\cdot}5{\cdot}2 = 1800$ вершин.)

Остальные параметры (p, w, o) указывают часть, завертывание и ориентацию, определяя дуги графа. Положим сначала, что $w = o = 0$. Если $p > 0$, имеем $(x_1, \ldots, x_t) \longrightarrow (y_1, \ldots, y_t)$ тогда и только тогда, когда $y_i = x_i + \delta_i$ для $1 \le i \le t$, где $(\delta_1, \ldots, \delta_t)$ представляет собой целочисленное решение уравнения $\delta_1^2 + \cdots + \delta_t^2 = |p|$. А если $p < 0$, мы также позволяем $y_i = x_i + k\delta_i$ для $k \ge 1$, что соответствует k ходам в том же направлении.

Если $w \ne 0$, пусть в двоичной записи w имеет вид $(w_t \ldots w_1)_2$. Тогда мы разрешаем "завертывание" $y_i = (x_i + \delta_i) \bmod b_i$ или $y_i = (x_i + k\delta_i) \bmod b_i$ по каждой координате i, для которой $w_i = 1$.

Если $o \ne 0$, граф ориентированный; смещения $(\delta_1, \ldots, \delta_t)$ дают дуги только тогда, когда они лексикографически больше, чем $(0, \ldots, 0)$. Но если $o = 0$, граф неориентированный.

Найдите установки $(n_1, n_2, n_3, n_4, p, w, o)$, для которых *board* будет давать следующие фундаментальные графы: (а) полный граф K_n; (б) путь P_n; (в) цикл C_n; (г) транзитивный турнир $\vec{K_n}$; (д) ориентированный путь $\vec{P_n}$; (е) ориентированный цикл $\vec{C_n}$; (ж) $m \times n$-решетку $P_m \,\square\, P_n$; (з) $m \times n$-цилиндр $P_m \,\square\, C_n$; (и) $m \times n$-тор $C_m \,\square\, C_n$; (к) $m \times n$-граф ладьи $K_m \,\square\, K_n$; (л) $m \times n$-ориентированный тор $\vec{C_m} \,\square\, \vec{C_n}$; (м) нулевой граф $\overline{K_n}$; (н) n-мерный куб $P_2 \,\square \cdots \square\, P_2$ с 2^n вершинами.

76. [*20*] Может ли *board*$(n_1, n_2, n_3, n_4, p, w, o)$ создавать петли или параллельные (повторяющиеся) ребра?

77. [*M20*] Докажите, что $\overline{G}$ имеет диаметр ≤ 3, если граф G имеет диаметр ≥ 3.

78. [*M27*] Пусть $G = (V, E)$ — граф с $|V| = n$ и $G \cong \overline{G}$. (Другими словами, граф G *самодополнительный*: существует перестановка φ множества V, такая, что u — v тогда и только тогда, когда $\varphi(u) \not- \varphi(v)$ и $u \ne v$. Можно представить, что ребра K_n раскрашены черным и белым цветами; белые ребра определяют граф, изоморфный графу с черными ребрами.)

а) Докажите, что $n \bmod 4 = 0$ или 1. Начертите диаграммы всех таких графов с $n < 8$.

б) Докажите, что если $n \bmod 4 = 0$, то каждый цикл перестановки φ имеет длину, кратную 4.

в) И обратно, каждая перестановка φ с такими циклами возникает в некотором таком графе G.

г) Распространите эти результаты на случай $n \bmod 4 = 1$.

▸ **79.** [*M22*] Для заданного $k \geq 0$ постройте граф на вершинах $\{0, 1, \ldots, 4k\}$, который одновременно является регулярным и самодополнительным.

▸ **80.** [*M22*] Самодополнительный граф должен иметь диаметр, равный 2 или 3 в соответствии с упр. 77. Для заданного $k \geq 2$ постройте самодополнительные графы обоих возможных диаметров, когда (а) $V = \{1, 2, \ldots, 4k\}$; (б) $V = \{0, 1, 2, \ldots, 4k\}$.

81. [*20*] Дополнение простого ориентированного графа без циклов определяется путем расширения (35) и (36), так что мы имеем $u \to v$ в $\overline{D}$ тогда и только тогда, когда $u \neq v$ и $u \not\to v$ в D. Что собой представляют самодополнительные ориентированные графы порядка 3?

82. [*M21*] Истинны или ложны приведенные далее утверждения о реберных графах?
а) Если G содержится в G', то $L(G)$ является порожденным подграфом $L(G')$.
б) Если G является регулярным графом, то же справедливо и для $L(G)$.
в) $L(K_{m,n})$ является регулярным для всех $m, n > 0$.
г) $L(K_{m,n,r})$ является регулярным для всех $m, n, r > 0$.
д) $L(K_{m,n}) \cong K_m \,\square\, K_n$.
е) $L(K_4) \cong K_{2,2,2}$.
ж) $L(P_{n+1}) \cong P_n$.
з) Графы G и $L(G)$ имеют одно и то же количество компонентов.

83. [*16*] Начертите граф $\overline{L(K_5)}$.

▸ **84.** [*M21*] Является ли $L(K_{3,3})$ самодополнительным?

85. [*M22*] (О. Оре (O. Ore), 1962.) Для каких графов G выполняется $G \cong L(G)$?

86. [*M20*] (Р. Д. Вильсон (R. J. Wilson).) Найдите граф G порядка 6, для которого $\overline{G} \cong L(G)$.

87. [*20*] Является ли граф Петерсена (а) 3-раскрашиваемым? (б) 3-реберно раскрашиваемым?

88. [*M20*] Граф $W_n = K_1 \,\text{—}\, C_{n-1}$ называется *колесом* порядка n, где $n \geq 4$. Сколько циклов содержит этот граф в качестве подграфов?


W_8

89. [*M20*] Докажите законы ассоциативности (42) и (43).

▸ **90.** [*M24*] Граф называется *сографом* (*cograph*), если он может быть построен алгебраически из одноэлементных графов посредством операций дополнения и прямого суммирования. Например, имеется четыре неизоморфных графа порядка 3, и все они являются сографами: $\overline{K_3} = K_1 \oplus K_1 \oplus K_1$ и его дополнение, K_3; $\overline{K_{1,2}} = K_1 \oplus K_2$ и его дополнение, $K_{1,2}$, где $K_2 = \overline{K_1 \oplus K_1}$.

Исчерпывающее перечисление показывает, что имеется 11 неизоморфных графов порядка 4. Приведите алгебраические формулы, доказывающие, что 10 из них являются сографами. Какой из них не является сографом?

▸ **91.** [*20*] Начертите диаграммы графов с четырьмя вершинами: (а) $K_2 \,\square\, K_2$; (б) $K_2 \otimes K_2$; (в) $K_2 \boxtimes K_2$; (г) $K_2 \vartriangle K_2$; (д) $K_2 \circ K_2$; (е) $\overline{K_2} \circ K_2$; (ж) $K_2 \circ \overline{K_2}$.

92. [*21*] Пять типов произведений графов, определенных в тексте, хорошо работают как для простых ориентированных графов, так и для обычных графов. Начертите диаграммы для ориентированных графов с четырьмя вершинами (а) $\vec{K_2} \,\square\, \vec{K_2}$; (б) $\vec{K_2} \otimes \vec{K_2}$; (в) $\vec{K_2} \boxtimes \vec{K_2}$; (г) $\vec{K_2} \vartriangle \vec{K_2}$; (д) $\vec{K_2} \circ \vec{K_2}$.

93. [*15*] Какие из пяти произведений графов превращают K_m и K_n в K_{mn}?

94. [*10*] Являются ли SGB-графы *words* порожденными подграфами $P_{26} \,\square\, P_{26} \,\square\, P_{26} \,\square\, P_{26} \,\square\, P_{26}$?

95. [*M20*] Если вершина u графа G имеет степень d_u, а вершина v графа H имеет степень d_v, то какова степень вершины (u,v) в (а) $G \mathbin{\square} H$? (б) $G \otimes H$? (в) $G \mathbin{\boxtimes} H$? (г) $G \mathbin{\triangle} H$? (д) $G \circ H$?

▶ **96.** [*M22*] Пусть A является матрицей $m \times m'$ с элементами $a_{uu'}$ на пересечении строки u и столбца u'; и пусть B является матрицей $n \times n'$ с элементами $b_{vv'}$ на пересечении строки v и столбца v'. *Прямое произведение* $A \otimes B$ представляет собой матрицу $mn \times m'n'$ с элементами $a_{uu'}b_{vv'}$ на пересечении строки (u,v) и столбца (u',v'). Таким образом, $A \otimes B$ представляет собой матрицу смежности для $G \otimes H$, если A и B являются матрицами смежности G и H.

Найдите аналогичные формулы для матриц смежности (а) $G \mathbin{\square} H$; (б) $G \mathbin{\boxtimes} H$; (в) $G \mathbin{\triangle} H$; (г) $G \circ H$.

97. [*M25*] Найдите столько интересных алгебраических соотношений между суммами и произведениями графов, сколько сможете. (Например, закон дистрибутивности $(A \oplus B) \otimes C = (A \otimes C) \oplus (B \otimes C)$ для прямых сумм и произведений матриц влечет за собой $(G \oplus G') \otimes H = (G \otimes H) \oplus (G' \otimes H)$. Мы также имеем $\overline{K_m} \mathbin{\square} H = H \oplus \cdots \oplus H$ с m копиями H, и т. д.)

98. [*M20*] Если граф G имеет k компонентов, а граф H имеет l компонентов, то сколько имеется компонентов в графах $G \mathbin{\square} H$ и $G \mathbin{\boxtimes} H$?

99. [*M20*] Пусть $d_G(u,u')$ — расстояние от вершины u до вершины u' в графе G. Докажите, что $d_{G \mathbin{\square} H}((u,v),(u',v')) = d_G(u,u') + d_H(v,v')$, и найдите аналогичную формулу для $d_{G \mathbin{\boxtimes} H}((u,v),(u',v'))$.

100. [*M21*] Для каких связных графов $G \otimes H$ связно?

▶ **101.** [*M25*] Найдите все связные графы G и H, такие, что $G \mathbin{\square} H \cong G \otimes H$.

102. [*M20*] Какова простейшая алгебраическая формула для графа *ходов королем* (по одной клетке по горизонтали, вертикали или диагонали) на доске $m \times n$?

103. [*20*] Завершите таблицу (54). Примените также алгоритм H к последовательности 866444444.

104. [*18*] Поясните манипуляции переменными i, t и r на шагах H3 и H4.

105. [*M38*] Предположим, что $d_1 \geq \cdots \geq d_n \geq 0$, и пусть $c_1 \geq \cdots \geq c_{d_1}$ — сопряжение этой последовательности, как в алгоритме H. Докажите, что последовательность $d_1 \ldots d_n$ является графической тогда и только тогда, когда $d_1 + \cdots + d_n$ четно и $d_1 + \cdots + d_k \leq c_1 + \cdots + c_k - k$ для $1 \leq k \leq s$, где s — максимальное значение, при котором $d_s \geq s$.

106. [*20*] Истинно или ложно следующее утверждение? Если $d_1 = \cdots = d_n = d < n$ и nd четно, алгоритм H строит *связный* граф.

107. [*M21*] Докажите, что последовательность степеней $d_1 \ldots d_n$ самодополнительного графа удовлетворяет условиям $d_j + d_{n+1-j} = n - 1$ и $d_{2j-1} = d_{2j}$ для $1 \leq j \leq n/2$.

▶ **108.** [*M23*] Разработайте алгоритм, аналогичный алгоритму H, который строит *простой ориентированный граф* на вершинах $\{1, \ldots, n\}$, имеющий определенные значения d_k^- и d_k^+ в качестве входящей и исходящей степеней каждой вершины k, если существует как минимум один такой граф.

109. [*M20*] Разработайте алгоритм, аналогичный алгоритму H, который строит *двудольный граф* на вершинах $\{1, \ldots, m+n\}$, имеющий определенные степени d_k для каждой вершины k, когда это возможно; все ребра j — k должны иметь $j \leq m$ и $k > m$.

110. [*M22*] Не прибегая к алгоритму H, покажите путем непосредственного построения, что последовательность $d_1 \ldots d_n$ является графической, когда $n > d_1 \geq \cdots \geq d_n \geq d_1 - 1$ и $d_1 + \cdots + d_n$ четно.

▸ **111.** [*25*] Пусть G представляет собой граф на вершинах $V = \{1, \ldots, n\}$ со степенью d_k у вершины k и $\max(d_1, \ldots, d_n) = d$. Докажите, что существует целое число N ($n \le N \le 2n$) и граф H на вершинах $\{1, \ldots, N\}$, такой, что H является регулярным графом степени d и $H \mid V = G$. Поясните, как построить такой регулярный граф с наименьшим возможным значением N.

▸ **112.** [*20*] Имеются ли в сети *miles*$(128, 0, 0, 0, 0, 127, 0)$ три эквидистантных города? Если нет, какие три города оказываются ближе всего к вершинам равностороннего треугольника?

113. [*05*] Если H является гиперграфом с m ребрами и n вершинами, то сколько строк и столбцов имеет его матрица инциденций?

114. [*M20*] Предположим, что мультиграф (26) рассматривается как гиперграф. Какова соответствующая матрица инциденций? Какой вид имеет соответствующий двудольный мультиграф?

▸ **115.** [*M20*] Пусть B представляет собой матрицу инциденций графа G. Поясните смысл симметричных матриц B^TB и BB^T.

116. [*M17*] Опишите ребра полного двудольного r-однородного гиперграфа $K_{m,n}^{(r)}$.

117. [*M22*] Сколько неизоморфных 1-однородных гиперграфов имеют m ребер и n вершин? (Ребра могут повторяться.) Перечислите их все для $m = 4$ и $n = 3$.

118. [*M20*] "Гиперлесом" называется гиперграф, не имеющий циклов. Если гиперлес имеет m ребер, n вершин и p компонентов, то чему равна сумма степеней его вершин?

119. [*M18*] Какой гиперграф соответствует (60) без последнего члена $(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4)$?

120. [*M20*] Определите *ориентированный гиперграф*, обобщая концепцию ориентированных графов.

121. [*M19*] Для заданного гиперграфа $H = (V, E)$ положим $I(H) = (V, F)$, где F — семейство всех максимальных независимых множеств гиперграфа H. Выразите $\chi(H)$ через $|V|$, $|F|$ и $\alpha(I(H)^T)$.

▸ **122.** [*M24*] Найдите наибольшее независимое множество и минимальное раскрашивание следующих систем троек: (а) гиперграф (56); (б) граф, дуальный графу Петерсена.

123. [*17*] Покажите, что оптимальное раскрашивание $K_n \,\square\, K_n$ эквивалентно решению знаменитой комбинаторной задачи.

124. [*M22*] Каково хроматическое число графа Чватала (рис. 2, (е))?

125. [*M48*] Для каких значений g существует 4-регулярный 4-хроматический граф с обхватом g?

▸ **126.** [*M22*] Найдите оптимальное раскрашивание "королевского тора" $C_m \boxtimes C_n$ при значениях $m, n \ge 3$.

127. [*M22*] Докажите, что (а) $\chi(G) + \chi(\overline{G}) \le n + 1$ и (б) $\chi(G)\chi(\overline{G}) \ge n$, когда G является графом порядка n, и найдите графы, для которых выполняется равенство.

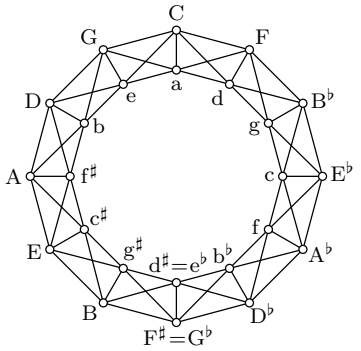
128. [*M18*] Выразите $\chi(G \,\square\, H)$ через $\chi(G)$ и $\chi(H)$, когда G и H являются графами.

129. [*23*] Опишите максимальные клики графа ходов ферзя (37).

130. [*M20*] Сколько максимальных клик в полном k-дольном графе?

131. [*M30*] Пусть $N(n)$ — наибольшее количество максимальных клик, которое может иметь граф с n вершинами. Докажите, что $3^{\lfloor n/3 \rfloor} \le N(n) \le 3^{\lceil n/3 \rceil}$.

▸ **132.** [*M20*] Назовем G *плотно раскрашиваемым* (*tightly colorable*), если $\chi(G) = \omega(G)$. Докажите, что $\chi(G \boxtimes H) = \chi(G)\chi(H)$, если G и H плотно раскрашиваемы.

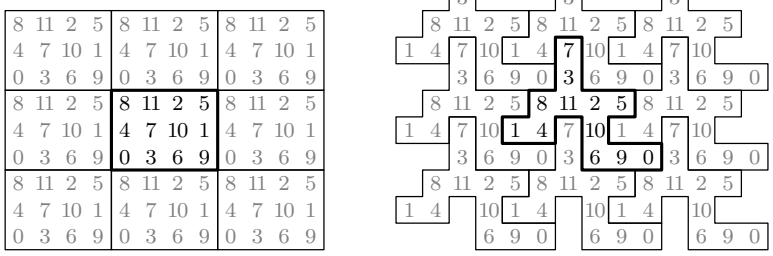


133. [*21*] Показанный "музыкальный граф" предоставляет хорошую возможность наглядно представить многочисленные определения, дававшиеся в этом разделе, поскольку его свойства легко проанализировать. Определите для данного графа (а) порядок; (б) размер; (в) обхват; (г) диаметр; (д) число независимости, $\alpha(G)$; (е) хроматическое число, $\chi(G)$; (ж) реберно-хроматическое число, $\chi(L(G))$; (з) количество клик, $\omega(G)$; (и) алгебраическую формулу графа, выражающую его как произведение хорошо известных меньших графов. Чему равен размер (к) наименьшего вершинного покрытия? (л) наибольшего паросочетания? Является ли граф G (м) регулярным? (н) планарным? (о) связным? (п) ориентированным? (р) свободным деревом? (с) гамильтоновым графом?

134. [*M22*] Сколько автоморфизмов имеет музыкальный граф?

▸ **135.** [*HM26*] Предположим, что композитор осуществляет случайное блуждание в музыкальном графе, начиная с вершины C, и делая на каждом шаге выбор одного из пяти ребер с равной вероятностью. Покажите, что после четного количества шагов блуждание с более высокой вероятностью завершится в вершине C, чем в любой другой. Какова точная вероятность попасть из вершины C в вершину C в результате блуждания в 12 шагов?

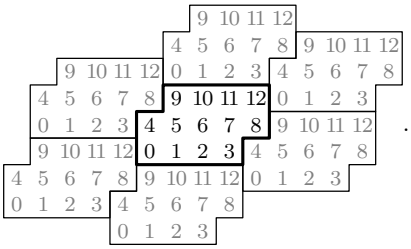
136. [*HM23*] *Ориентированный граф Кейли* представляет собой ориентированный граф, вершины V которого являются элементами группы, а его дугами являются $v \longrightarrow v\alpha_j$ для $1 \le j \le d$ и всех вершин v, где $(\alpha_1, \ldots, \alpha_d)$ — фиксированные элементы группы. *Граф Кейли* — это ориентированный граф Кейли, который одновременно представляет собой граф. Является ли граф Петерсена графом Кейли?



▸ **137.** [*M25*] (*Обобщенные торы.*) Тор размером $m \times n$ можно рассматривать как замощение плоскости. Например, можно представить, что бесконечное количество копий торов 3×4 из (50) размещаются вместе в виде решетки, как показано на приведенном выше рисунке слева; из каждой вершины можно перемещаться на север юг, восток или запад в следующую вершину тора. Вершины здесь были пронумерованы таким образом, чтобы перемещение на север из вершины v приводило в вершину $(v + 4) \bmod 12$, перемещение на восток — в вершину $(v + 3) \bmod 12$, и т. д. В правой части рисунка показан тот же тор, но с несколько иной формой мозаичного элемента; *любой* вариант выбора двенадцати ячеек с номерами $\{0, 1, \ldots, 11\}$ будет замащивать плоскость, с точно тем же лежащим в основе замощения графом.

Смещенные копии одной плитки будут также заполнять плоскость, если они образуют *обобщенный* тор, в котором ячейка (x, y) соответствует той же вершине, что и ячейки

$(x+a, y+b)$ и $(x+c, y+d)$, где (a,b) и (c,d) — целочисленные векторы, и $n = ad - bc > 0$. Тогда обобщенный тор будет иметь n точек. Эти векторы (a,b) и (c,d) в приведенном выше примере 3×4 представляют собой $(4,0)$ и $(0,3)$; а когда они становятся соответственно $(5,2)$ и $(1,3)$, мы получаем



Здесь $n = 13$, и перемещение в северном направлении из v дает $(v+4) \bmod 13$; перемещение в восточном направлении приводит в $(v+1) \bmod 13$.

Докажите, что если $\gcd(a,b,c,d) = 1$, то вершинам такого обобщенного тора всегда можно присвоить целочисленные метки $\{0, 1, \ldots, n-1\}$ таким образом, что соседями v являются $(v \pm p) \bmod n$ и $(v \pm q) \bmod n$ для некоторых целых чисел p и q.

138. [*HM27*] Продолжая выполнять упр. 137, придумайте эффективный способ присваивания меток k-мерным вершинам $x = (x_1, \ldots, x_k)$, когда заданы целочисленные векторы α_j, такие, что каждый вектор x эквивалентен $x + \alpha_j$ для $1 \le j \le k$. Проиллюстрируйте свой метод для случая $k = 3$, $\alpha_1 = (3,1,1)$, $\alpha_2 = (1,3,1)$, $\alpha_3 = (1,1,3)$.

▸ **139.** [*M22*] Пусть H представляет собой фиксированный граф порядка h, а $\#(H{:}G)$ — количество раз, когда H оказывается порожденным подграфом данного графа G. Если G выбран случайным образом из множества всех $2^{n(n-1)/2}$ графов на вершинах $V = \{1, 2, \ldots, n\}$, то чему равно среднее значение $\#(H{:}G)$ для случаев, когда H представляет собой (а) K_h; (б) P_h для $h > 1$; (в) C_h для $h > 2$; (г) произвольный граф?

140. [*M30*] Граф G называется *пропорциональным*, если каждое из его количеств порожденных подграфов $\#(K_3{:}G)$, $\#(\overline{K_3}{:}G)$ и $\#(P_3{:}G)$ согласуется с ожидаемыми значениями, полученными в упр. 139.

а) Покажите, что колесо W_8 из упр. 88 пропорционально в данном смысле.

б) Докажите, что граф G пропорционален тогда и только тогда, когда $\#(K_3{:}G) = \frac{1}{8}\binom{n}{3}$ и последовательность степеней $d_1 \ldots d_n$ его вершин удовлетворяет тождествам

$$d_1 + \cdots + d_n = \binom{n}{2}, \qquad d_1^2 + \cdots + d_n^2 = \frac{n}{2}\binom{n}{2}. \tag{$*$}$$

141. [*26*] Условия из упр. 140, (б) могут выполняться, только если $n \bmod 16 \in \{0, 1, 8\}$. Напишите программу для поиска всех пропорциональных графов с $n = 8$ вершинами.

142. [*M30*] (С. Янсон (S. Janson) и Я. Кратохвил (J. Kratochvil), 1991.) Докажите, что не существует графа G на четырех или более вершинах, являющегося "сверхпропорциональным" в том смысле, что количество его подграфов $\#(H{:}G)$ согласуется с ожидаемыми значениями из упр. 139 для каждого из одиннадцати неизоморфных графов H порядка 4, которые рассматривались в упр. 90. *Указание*: обратите внимание, что $(n-3)\#(K_3{:}G) = 4\#(K_4{:}G) + 2\#(K_{1,1,2}{:}G) + \#(\overline{K_{1,3}}{:}G) + \#(\overline{K_1 \oplus K_{1,2}}{:}G)$.

▸ **143.** [*M25*] Пусть A является произвольной матрицей с $m > 1$ различными строками и $n \ge m$ столбцами. Докажите, что как минимум один столбец A может быть удален так, что при этом никакие две строки не станут равными.

▸ **144.** [*21*] Пусть X — матрица размером $m \times n$, элементами x_{ij} которой являются 0, 1 или $*$. "Дополнением" X является матрица X^*, в которой каждая $*$ заменена на 0 либо 1.

Покажите, что задача поиска дополнения с наименьшим количеством различных строк эквивалентна задаче поиска хроматического числа графа.

▸ **145.** [*25*] (Р. С. Бойер (R. S. Boyer) и Д. С. Мур (J. S. Moore), 1980.) Предположим, что массив $a_1 \dots a_n$ содержит *мажоритарный элемент*, т. е. значение, которое встречается более чем $n/2$ раз. Разработайте алгоритм, который находит его менее чем за n сравнений. *Указание*: если $n \geq 3$ и $a_{n-1} \neq a_n$, мажоритарный элемент $a_1 \dots a_n$ является также мажоритарным элементом $a_1 \dots a_{n-2}$.