ПРЕДИСЛОВИЕ

Поместить все сто́ящее в одну книгу совершенно
невозможно, и даже попытки сколь-нибудь полно осветить
хотя бы некоторые аспекты рассматриваемого предмета,
скорее всего, приведут к стремительному увеличению книги.

— ДЖЕРАЛЬД Б. ФОЛЛАНД (GERALD B. FOLLAND), *Editor's Corner* (2005)

Том 4 называется *Комбинаторные алгоритмы*. Когда я предлагал это название, я очень хотел добавить к нему подзаголовок *Мой любимый раздел программирования*. Редакторы уговорили меня не делать этого, но факт остается фактом — программы с комбинаторным привкусом всегда были моими любимицами.

С другой стороны, я часто с удивлением обнаруживал, что в сознании многих людей слово "комбинаторный" ассоциируется с вычислительной сложностью. И действительно, Сэмюэль Джонсон (Samuel Johnson) в своем знаменитом Словаре английского языка (1755) писал, что соответствующее существительное "в настоящее время обычно используется в неверном смысле". От коллег мне приходится слышать горестные рассказы о том, как "комбинаторика нас победила". Так почему же мне комбинаторика доставляет удовольствие, а у других вызывает панику?

Да, это правда, что комбинаторные задачи часто связаны с чрезвычайно большими числами. Статья в словаре Джонсона включает цитату Эфраима Чамберса (Ephraim Chambers), который указал, что общее количество слов длиной не более 24 букв при 24-буквенном алфавите равно 1 391 724 288 887 252 999 425 128 493 402 200. Соответствующее число для 10-буквенного алфавита равно 11 111 111 110; когда же количество букв алфавита равно всего лишь 5, общее число слов сокращается до 3905. Таким образом, здесь мы, определенно, наблюдаем комбинаторный взрыв с ростом размера задачи от 5 до 10 и далее до 24 и более.

На протяжении моей жизни мощность вычислительной техники выросла неимоверно. Сейчас, когда я набираю эти слова, я знаю, что мой ноутбук обрабатывает их со скоростью, более чем в сто тысяч раз большей, чем старый добрый IBM Type 650, которому я посвящал свои книги. Количество памяти моего нынешнего компьютера также примерно в сто тысяч раз больше. Компьютеры завтрашнего дня будут еще более быстрыми и с еще бо́льшим объемом памяти. Но эти удивительные достижения не уменьшили тягу людей к получению ответов на комбинаторные вопросы — скорее даже наоборот. Наша совершенно невообразимая в прошлом способность к быстрым вычислениям повысила наши ожидания и еще больше разожгла аппетит к дальнейшим исследованиям, потому что размер комбинаторной задачи может увеличиться более чем в сто тысяч раз, когда n увеличится всего лишь на единицу.

Неформально комбинаторные алгоритмы можно определить как методы высокоскоростной работы с такими комбинаторными объектами, как перестановки

или графы. Обычно мы пытаемся найти шаблоны или схемы, которые наилучшим образом удовлетворяют определенным ограничениям. Количество задач такого рода огромно, а искусство написания соответствующих программ особенно важно, поскольку одна хорошая идея способна сэкономить годы и даже века машинного времени.

В действительности тот факт, что хорошие алгоритмы для комбинаторных задач могут давать потрясающий выигрыш, привел к огромному подъему современного состояния данной науки. Многие ранее считавшиеся неразрешимыми задачи теперь могут быть с легкостью решены, а многие алгоритмы, считавшиеся ранее очень хорошими, стали гораздо лучше. Начиная примерно с 1970 года кибернетики стали сталкиваться с явлением, которое окрестили леммой Флойда: задачи, которые представляются требующими для решения n^3 операций, на самом деле могут быть решены за $O(n^2)$ операций; задачи, которые, как кажется, требуют n^2 операций, могут быть решены за $O(n \log n)$; задачи со временем решения $n \log n$ зачастую сводятся к задачам со временем решения $O(n)$. Для более сложных задач удается добиться сокращения времени выполнения с $O(2^n)$ до $O(1.5^n)$ или $O(1.3^n)$, и т. д. Прочие задачи остаются сложными в общем случае, но могут быть найдены важные частные случаи, которые решаются гораздо проще. Многие комбинаторные вопросы, о которых я думал, что ответ на них не будет получен при моей жизни, оказались успешно решенными. И это произошло главным образом благодаря усовершенствованию алгоритмов, а не повышению скорости процессоров.

К 1975 году исследования в этой области проводились такими темпами, что значительная доля статей, опубликованных в то время в ведущих журналах по информатике, была посвящена комбинаторным алгоритмам. И этот прорыв был осуществлен не только учеными-кибернетиками; значительный вклад внесли работники в таких областях знаний, как электротехника, искусственный интеллект, исследование операций, математика, физика, статистика и многие другие. Я пытался завершить том 4 *Искусства программирования*, но вместо этого я почувствовал себя сидящим на крышке кипящего чайника: я столкнулся с комбинаторным взрывом другого рода — огромным взрывом новых идей!

Данная серия книг родилась в начале 1962 года, когда я наивно набросал список рабочих названий глав для книги, состоящей из 12 глав. В тот момент я решил включить краткую главу о комбинаторных алгоритмах просто для собственного удовольствия. "Эй, глянь — большинство людей используют компьютеры, чтобы работать с числами, но можно писать программы, которые имеют дело со схемами". В те времена было легко дать достаточно полное описание почти всех известных комбинаторных алгоритмов. И даже к 1966 году, когда я закончил первый черновой вариант из около трех тысяч рукописных страниц (который уже перерос изначально запланированную книгу), к главе 7 относилось менее ста из них. Я совершенно не представлял в тот момент, что то, что я планировал как "легкую закуску", окажется основным блюдом.

Великое комбинаторное брожение 1975 года захватило множество людей. Новые идеи улучшали старые, но при этом редко заменяли их или делали устаревшими. Все это привело к тому, что я был вынужден оставить надежду когда-либо написать достаточно полную книгу, в которой был бы упорядочен весь имеющийся материал и которая была бы достаточно универсальной, чтобы помочь каждому, перед кем

встает та или иная комбинаторная задача. Множество применяемых методов разрослось до таких размеров, что я уже практически не мог рассмотреть некоторую подтему и сказать "Это решение окончательное: на этом мы ставим точку". Вместо этого я должен был ограничиться пояснением наиболее важных принципов, которые, как мне кажется, лежат в основе всех эффективных комбинаторных методов, с которыми я встречался до сих пор. В настоящее время для тома 4 у меня накоплено в два с лишним раза больше материала, чем было собрано для всех предыдущих томов вместе взятых.

Такое количество материала неизбежно ведет к разделению запланированного "тома 4" на несколько физических томов. Сейчас перед вами том 4, А. В один прекрасный день вы увидите тома 4, Б и 4, В, если, конечно, написать их позволит мое здоровье. И, как знать, быть может, наступит очередь томов 4, Г, 4, Д ...? Впрочем, я уверен, что тома 4, Я не будет.

Я планирую по мере возможности систематически разбирать накопленные с 1962 года папки и в меру своих способностей рассказывать истории, которые все еще ждут своего часа. Я не могу гарантировать полноту изложения, но очень хочу воздать должное всем первооткрывателям ключевых идей, а потому я не буду экономить на исторических деталях. Кроме того, всякий раз, когда я знакомлюсь с чем-то, что, как мне кажется, останется важным еще лет 50 и что можно пояснить парой абзацев, я не могу пройти мимо этого и не внести его в книгу. С другой стороны, сложный материал, который требует длительных доказательств, выходит за рамки этих книг (конечно, за исключением действительно фундаментальных тем).

Итак, понятно, что вся область комбинаторных алгоритмов огромна, и охватить ее всю не в моих силах. Какие же темы из тех, которые я опускаю, наиболее важные? Я думаю, мое наибольшее "слепое пятно" — геометрия, потому что я всегда лучше чувствовал себя при работе с визуализацией и алгебраическими формулами, чем с объектами в пространстве. Поэтому я не пытаюсь заниматься в этих книгах комбинаторными задачами, связанными с вычислительной геометрией, такими как плотная упаковка сфер, кластеризация точек данных в n-мерном евклидовом пространстве или деревьями Штайнера на плоскости. Еще более существенно то, что я уклоняюсь от такой темы, как полиэдрическая комбинаторика, и от подходов, основанных, в первую очередь, на линейном программировании, целочисленном программировании или полуопределенном программировании. Эти вопросы рассматриваются во многих других книгах по данной тематике и рассчитаны на геометрическую интуицию читателя. Для моего понимания чисто комбинаторные подходы гораздо проще.

Должен также признаться в некотором предубеждении против алгоритмов, эффективных только в асимптотическом смысле, превосходная производительность которых начинает проявляться, только когда размер задачи превосходит размер вселенной. Алгоритмам такого рода посвящено множество современных публикаций. Я могу понять, почему медитация над беспредельными пределами столь интеллектуально привлекательна и так легко попадает в академическую печать. Однако в *Искусстве программирования* любые методы, которые я сам не испытал в реальных программах, ожидает быстрая расправа. (Конечно же, из этого правила есть исключения, главным образом касающиеся базовых концепций. Одни непрактичные методы просто слишком красивы и/или интеллектуальны, чтобы их можно

было исключить; другие представляют собой поучительные примеры того, как *не* следует поступать.)

Кроме того, как и в предыдущих томах данной серии, я преднамеренно практически полностью ограничиваюсь *последовательными* алгоритмами, несмотря на все возрастающую способность компьютеров к параллельным вычислениям. Я не могу судить, какие из идей, относящихся к параллельным вычислениям, будут полезны через пять или десять лет, не говоря уж о бóльших сроках, а потому оставляю эти вопросы другим, более сведущим авторам. Последовательные методы сами по себе уже являются испытанием пределов моей способности разглядеть, что искусные программисты захотят знать завтра.

Основные решения, которые мне нужно было принять при планировании представления данного материала, — как его лучше организовать: в соответствии с решаемыми задачами или в соответствии с применяемыми методами. Так, например, глава 5 в томе 3 посвящена единственной задаче — сортировке данных; к различным ее аспектам оказались применимыми более двух десятков разнообразных методов. Комбинаторные алгоритмы, напротив, включают множество различных задач, которые, как правило, решаются с помощью существенно меньшего набора методов. В конечном итоге я пришел к выводу, что смешанная стратегия будет работать лучше любого "чистого" подхода. Так, например, в разделе 7.3 рассматривается задача поиска кратчайших путей, а в разделе 7.4.1 — задача связности. При этом многие другие разделы посвящены базовым методам, таким как применение булевой алгебры (раздел 7.1), откат (раздел 7.2.2), теория матроидов (раздел 7.6) или динамическое программирование (раздел 7.7). Знаменитая задача коммивояжера и другие классические комбинаторные задачи, связанные с покрытием, раскраской и упаковкой, не имеют собственных разделов, но неоднократно появляются в разных местах, так как они могут решаться различными способами.

Я упомянул большой прогресс в области комбинаторных вычислений, но это совсем не означает, что все комбинаторные задачи действительно "приручены". Когда время работы компьютерной программы взлетает не хуже баллистической ракеты, ее разработчикам не следует рассчитывать на то, что в этой книге найдется "серебряная пуля", некий универсальный рецепт для решения их проблемы. Описанные здесь методы зачастую будут работать намного быстрее, чем подходы, которые пытались применить эти программисты. Но давайте посмотрим правде в глаза: комбинаторные задачи очень быстро становятся невообразимо огромными. Можно даже строго доказать, что некоторая небольшая, естественная задача *никогда* не будет иметь практического решения, хотя принципиально она разрешима (см. теорему Стокмейера и Мейера в разделе 7.1.2). В других случаях мы знаем, что подходящий алгоритм для решения данной задачи вряд ли существует, хотя и не можем этого доказать. Дело в том, что любой такой эффективный алгоритм позволит решить тысячи других задач, которые давно ставят в тупик крупнейших специалистов в мире (см. описание NP-полноты в разделе 7.9).

Опыт показывает, что изобретение новых комбинаторных алгоритмов будет продолжаться по-прежнему как для новых комбинаторных задач, так и для вновь выявленных вариаций или частных случаев старых. Аппетит программистов к таким алгоритмам со временем будет только расти. Когда программисты сталкиваются с задачами такого рода (а это происходит постоянно), это дает толчок искусству

программирования для достижения новых высот. Однако, скорее всего, сегодняшние методы также останутся в строю.

Бóльшая часть данной книги самодостаточна, хотя и содержит частые ссылки на темы в предыдущих томах. В них достаточно широко рассматривались детали низкоуровневого программирования на машинном языке, так что в данной книге алгоритмы обычно описываются на абстрактном уровне, не зависящем от конкретной машины. Однако некоторые аспекты комбинаторного программирования существенно зависят от низкоуровневых деталей, которые ранее не рассматривались. В таких случаях все примеры в данной книге основаны на компьютере `MMIX`, который заменяет машину `MIX`, описанную в ранних изданиях тома 1. Подробная информация о `MMIX` приведена в дополнении к этому тому, выпущенному в мягкой обложке под названием *The Art of Computer Programming*, Volume 1, Fascicle 1,* в котором содержатся разделы 1.3.1′, 1.3.2′ и т. д. Эти разделы, а также соответствующие ассемблеры и симуляторы, доступны по сети Интернет.

Еще одним загружаемым ресурсом является коллекция программ и данных под названием *The Stanford GraphBase*, постоянно упоминаемая в примерах этой книги. Читателям при изучении комбинаторных алгоритмов настоятельно рекомендуется поработать с ней — мне представляется, что это наиболее эффективный и приятный способ изучения.

Кстати, при написании вводного материала в начале главы 7 мне было приятно отметить, что в ней самым естественным образом оказались упомянутыми некоторые работы моего научного руководителя Маршалла Холла-мл. (Marshall Hall, Jr.) (1910–1990), так же как и некоторые работы *его* научного руководителя Ойстейна Оре (Oystein Ore) (1899–1968) и некоторые работы *его* научного руководителя Торальфа Сколема (Thoralf Skolem) (1887–1963). Научный руководитель Сколема, Аксель Тью (Axel Thue) (1863–1922), уже упомянут в главе 6.

Я чрезвычайно благодарен сотням читателей, которые помогли мне выявить многочисленные ошибки, допущенные в черновых вариантах этого тома, которые изначально были доступны через Интернет, а впоследствии напечатаны в виде брошюр в мягких обложках. В особенности следует отметить комментарии Торстена Далхаймера (Thorsten Dahlheimer), Марка ван Льювена (Marc van Leeuwen) и Удо Вермута (Udo Wermuth). Но, боюсь, в материале скрываются и другие ошибки, которые я хотел бы исправить как можно скорее. Поэтому я с удовольствием заплачу 2 доллара 56 центов тому, кто первым сообщит мне о любой замеченной типографской**, технической или исторической ошибке. На веб-сайте по адресу `http://www-cs-faculty.stanford.edu/~knuth/taocp.html` можно найти текущий список всех поправок к данной книге.

Станфорд, Калифорния D. E. K.
Октябрь 2010

---

* Имеется русский перевод: Кнут Д. Э. *Искусство программирования, том 1, выпуск 1. MMIX — RISC-компьютер для нового тысячелетия.* — М.: ООО “И. Д. Вильямс”, 2007. — *Примеч. пер.*

** Конечно же, имеется в виду оригинальное издание. — *Примеч. пер.*

В предисловии к первому изданию я просил
читателей не обращать внимания на ошибки.
Теперь я не хочу этого делать и благодарен тем
читателям, которые не обратили внимания на мою просьбу.

— СТЮАРТ САЗЕРЛЕНД (STUART SUTHERLAND),
The International Dictionary of Psychology (1996)

Естественно, я несу ответственность за все оставшиеся ошибки —
хотя, как мне кажется, мои друзья могли бы выловить и больше.

— ХРИСТОС Х. ПАПАДИМИТРИУ (CHRISTOS H. PAPADIMITRIOU),
Computational Complexity (1994)

Мне нравится работать в разных областях,
потому что так мои ошибки размазываются более тонким слоем.

— ВИКТОР КЛИ (VICTOR KLEE) (1999)

Примечание к ссылкам. Несколько часто цитируемых журналов и трудов конференций имеют специальные коды, которые встречаются в предметно-именном указателе в конце этой книги. Различные серии *IEEE Transactions* цитируются с использованием буквенного кода, указывающего серию и приводимого полужирным шрифтом перед номером тома. Так, '*IEEE Trans.* **C-35**' означает *IEEE Transactions on Computers*, том 35. IEEE больше не использует эти столь удобные буквенные коды, но их нетрудно расшифровать: '**EC**' означает "Electronic Computers", '**IT**' — "Information Theory", '**SE**' — "Software Engineering", '**SP**' — "Signal Processing" и т. д.; '**CAD**' означает "Computer-Aided Design of Integrated Circuits and Systems".

Перекрестная ссылка вида 'упр. 7.10–00' указывает на будущее упражнение в разделе 7.10, номер которого пока что неизвестен.

Примечание к обозначениям. Простые и интуитивно понятные соглашения и обозначения для алгебраического представления математических концепций всегда были благом для прогресса, в особенности когда большинство исследователей в мире стали использовать единый символьный язык. Текущее состояние дел в комбинаторной математике, к сожалению, в этом отношении находится в беспорядке, так как одни и те же символы различными группами исследователей могут использоваться для совершенно разных целей. Некоторые специалисты, работающие в сравнительно узких областях, непреднамеренно создали противоречивую символику. Информатике — которая по своей природе очень широко взаимодействует с математикой — следует избегать этой опасности, принимая для себя, насколько это возможно, непротиворечивые обозначения. Поэтому мне часто приходилось делать выбор среди конкурирующих систем обозначений, при этом отдавая себе отчет в том, что угодить всем будет невозможно. Я изо всех сил пытался сделать все, чтобы придумать обозначения, которые, как мне кажется, будут наилучшими для будущего применения, зачастую после многих лет экспериментов и обсуждений с коллегами. Не редкостью были и метания между альтернативными вариантами, пока не удавалось остановиться на одном из них. Обычно все же удавалось найти

удобные обозначения, еще не используемые другими учеными способом, противоречащим моему.

Приложение Б представляет собой всеобъемлющий предметный указатель для всех основных обозначений, которые использованы в настоящей книге, включая и те, которые (пока что?) не являются стандартными. Если вы столкнетесь в книге с формулой, которая выглядит для вас странно и/или непонятно, достаточно высоки шансы, что приложение Б направит вас к странице, поясняющей мои намерения. Однако я хотел бы сделать несколько пояснений прямо сейчас, чтобы вы могли ознакомиться с ними еще до первого прочтения книги.

- Шестнадцатеричные константы предваряются знаком фунта (#), т. е., например, $\#123$ означает $(123)_{16}$.
- Обозначение $x \dot{-} y$ применяется для операции "монус", иногда именуемой "минус с точкой" или вычитанием с насыщением, результатом которой является $\max(0, x - y)$.
- Медиана трех чисел $\{x, y, z\}$ обозначается как $\langle xyz \rangle$.
- Множество, такое как $\{x\}$, состоящее из одного элемента, часто обозначается просто как x в контекстах наподобие $X \cup x$ или $X \setminus x$.
- Если n — неотрицательное целое число, количество единичных битов в бинарном представлении n равно νn. Кроме того, если $n > 0$, то крайний слева и крайний справа единичные биты n представляют собой соответственно $2^{\lambda n}$ и $2^{\rho n}$. Например, $\nu 10 = 2$, $\lambda 10 = 3$, $\rho 10 = 1$.
- Декартово произведение графов G и H обозначается как $G \mathbin{\square} H$. Например, $C_m \square C_n$ представляет собой тор $m \times n$, так как C_n является циклом из n вершин.

ПРИМЕЧАНИЯ К УПРАЖНЕНИЯМ

Упражнения, приведенные в этом многотомнике, предназначены как для самостоятельной проработки, так и для семинарских занятий. Очень трудно и, наверное, просто невозможно выучить предмет, только читая теорию и не применяя ее для решения конкретных задач, которые заставляют задуматься о прочитанном. Более того, мы лучше всего заучиваем то, до чего дошли самостоятельно, своим умом. Поэтому упражнения занимают важное место в данном издании. Я приложил немало усилий, чтобы сделать их как можно более информативными, а также отобрать задачи, которые были бы не только поучительными, но и позволяли бы читателю получить удовольствие от их решения.

Во многих книгах простые упражнения даются вперемешку с исключительно сложными. Это не всегда удобно, так как читателю хочется знать заранее, сколько времени ему придется затратить на решение задач (иначе в лучшем случае он их только просмотрит). В качестве классического примера подобной ситуации можно привести книгу Ричарда Беллмана (Richard Bellman) *Динамическое программирование* (М.: Изд-во иностр. лит., 1960). Это очень важная, новаторская работа, но у нее есть один недостаток: в конце некоторых глав в разделе "Упражнения и научные проблемы" среди серьезных, еще нерешенных проблем, попадаются простейшие вопросы. Говорят, что кто-то однажды спросил д-ра Беллмана, как отличить упражнения от научных проблем, и он ответил: "Если вы можете решить задачу, значит, это упражнение; в противном случае это научная проблема".

Совершенно очевидно, что в книге, подобной этой, должны быть приведены и сложные научные проблемы, и простейшие упражнения. Поэтому, чтобы читатель не ломал голову, пытаясь отличить одно от другого, были введены *рейтинги*, которые определяют степень сложности каждого упражнения. Эти рейтинги имеют следующее значение.

Рейтинг *Объяснение*

00 Чрезвычайно простое упражнение, которое можно выполнить сразу же, если прочитанный материал понят. Упражнения подобного типа почти всегда можно решить "в уме".

10 Простая задача, которая заставляет задуматься над прочитанным, но не представляет особых трудностей. На ее решение вы затратите не больше минуты; в процессе решения могут понадобиться карандаш и бумага.

20 Средняя задача, которая позволяет проверить, понял ли читатель основные положения изложенного материала. Чтобы получить исчерпывающий ответ, может понадобиться примерно 15–20 минут.

30 Задача умеренной сложности. Для ее решения может понадобиться более двух часов (а если при этом включен телевизор, то еще больше).

40 Достаточно сложная или трудоемкая задача, которую вполне можно включить в план семинарских занятий. Предполагается, что студент должен справиться с ней, затратив не слишком много времени, и решение будет нетривиальным.

50 Научная проблема, которая (насколько известно автору в момент написания книги) пока еще не получила удовлетворительного решения, хотя найти его пытались очень многие. Если вы нашли решение подобной проблемы, то опубликуйте его; более того, автор данной книги будет очень признателен, если ему сообщат решение как можно скорее (при условии, что оно правильно).

Интерполируя по этой "логарифмической" шкале, можно понять, что означает любой промежуточный рейтинг. Например, рейтинг *17* говорит о том, что упражнение немного проще, чем задача средней сложности. Если задача с рейтингом *50* будет впоследствии решена каким-либо читателем, то в следующих изданиях данной книги и в списке ошибок, опубликованных в Интернете, она может иметь рейтинг *40*.

Остаток от деления рейтинга на 5 показывает, какой объем рутинной работы потребуется для решения данной задачи. Таким образом, для выполнения упражнения с рейтингом *24* может потребоваться больше времени, чем для упражнения с рейтингом *25*, но для последнего необходим более творческий подход.

Автор очень старался правильно присвоить рейтинги упражнениям, но тому, кто составляет задачи, трудно предвидеть, насколько сложными они окажутся для кого-то другого. К тому же одному человеку некая задача может показаться простой, а другому — сложной. Таким образом, определение рейтингов — дело достаточно субъективное и относительное. Я надеюсь, что рейтинги помогут вам получить правильное представление о степени трудности задач, но их следует воспринимать в качестве ориентира, а не в качестве абсолюта.

Эта книга написана для читателей с различным уровнем математической подготовки и научного кругозора, поэтому некоторые упражнения рассчитаны исключительно на тех, кто серьезно интересуется математикой или занимается ею профессионально. Если рейтингу предшествует буква *M*, значит, математические понятия и обоснования используются в упражнении в большей степени, чем это необходимо тому, кто интересуется в основном программированием алгоритмов. Если же упражнение отмечено буквами *HM*, то для его решения необходимо знание высшей математики в большем объеме, чем дается в настоящей книге. Но пометка *HM* совсем *не*обязательно означает, что упражнение трудное.

Перед некоторыми упражнениями стоит стрелка "▸", которая означает, что они особенно поучительны и их очень рекомендуется выполнить. Само собой разумеется, никто не ожидает, что читатель (или студент) будет решать *все* задачи, поэтому наиболее важные из них и были выделены. Но это ни в коем случае не означает, что другие упражнения выполнять не стоит! Каждый читатель должен хотя бы попытаться решить все задачи, рейтинг которых меньше или равен *10*. Стрелки помогут выбрать задачи с более высокими рейтингами, которые следует решать в первую очередь.

В некоторых разделах имеется более сотни упражнений. Как же найти свой путь среди такого множества? В целом последовательность упражнений стремится

следовать последовательности идей в основном тексте. Идущие подряд упражнения могут основываться одно на другом, как в новаторских задачниках Пойя (Pólya) и Сегё (Szegő). В последних упражнениях разделов часто охватывается материал всего раздела или вводятся дополнительные темы.

К большинству упражнений приведены ответы, помещенные в отдельном разделе в конце книги. Пожалуйста, пользуйтесь ими разумно: ответ смотрите только после того, как приложите все усилия, чтобы решить задачу самостоятельно, либо если у вас совершенно нет времени на ее решение. Ответ будет поучителен и полезен для вас только в том случае, если вы ознакомитесь с ним *после* того, как найдете свое решение или изрядно потрудитесь над задачей. Ответы к задачам излагаются очень кратко и схематично, так как предполагается, что читатель честно пытался решить задачу собственными силами. Иногда в приведенном решении дается меньше информации, чем спрашивалось, но чаще бывает наоборот. Вполне возможно, что полученный вами ответ окажется лучше того, который помещен в книге, или вы найдете ошибку в ответе. В таком случае автор был бы очень признателен, если бы вы как можно скорее подробно сообщили ему об этом; тогда в последующих изданиях книги будет опубликовано более удачное решение, а также имя его автора.

Решая задачи, вы, как правило, можете пользоваться ответами к предыдущим упражнениям, за исключением случаев, когда это будет оговорено особо. Рейтинги упражнениям присваивались в расчете именно на это, и вполне возможно, что рейтинг упражнения $n + 1$ ниже рейтинга упражнения n, даже если результат упражнения n является его частным случаем.

Условные обозначения		*00*	Простейшее (ответ дать немедленно)
		10	Простое (на одну минуту)
▶	Рекомендуется	*20*	Средней трудности (на четверть часа)
M	С математическим уклоном	*30*	Повышенной трудности
HM	Требует знания высшей математики	*40*	Высокой трудности
		50	Научная проблема

УПРАЖНЕНИЯ

▶ **1.** [*00*] Что означает рейтинг *M20*?

2. [*10*] В чем ценность упражнений в данной книге для читателя?

3. [*HM45*] Докажите, что всякое односвязное компактное трехмерное многообразие гомеоморфно трехмерной сфере.

Искусство является значительной составляющей такого благотворного занятия, как противостояние предубеждениям.

— ГЕНРИ ДЖЕЙМС (HENRY JAMES), *The Art of Fiction* (1884)

Я признателен всем моим друзьям, но особая моя благодарность тем из них, которые по прошествии времени перестают теребить меня вопросом "Какая книга будет следующей?"

— ПИТЕР Д. ГОМЕС (PETER J. GOMES), *The Good Book* (1996)

Наконец-то после долгих обещаний я представляю миру мою работу. Однако я боюсь, что по отношению к ней были поставлены слишком высокие ожидания. Причина задержки ее публикации в значительной мере кроется в чрезвычайном рвении, которое было проявлено высокопочтенными особами со всех сторон в обеспечении меня дополнительной информацией.

— ДЖЕЙМС БОСВЕЛЛ (JAMES BOSWELL), *The Life of Samuel Johnson, LL. D.* (1791)

Автор особенно признателен издательству Addison-Wesley за терпение, проявленное в течение десятилетия ожидания этой книги после подписания контракта.

— ФРЭНК ХАРАРИ (FRANK HARARY), *Graph Theory* (1969)

Средний парень, ненавидящий квадратные корни или алгебру, приходит в восторг от головоломок, построенных на тех же принципах, и может таким способом развить свои математические и изобретательские шишки, которые приведут в изумление семейного френолога.*

— СЭМ ЛОЙД (SAM LOYD), *The World of Puzzledom* (1896)

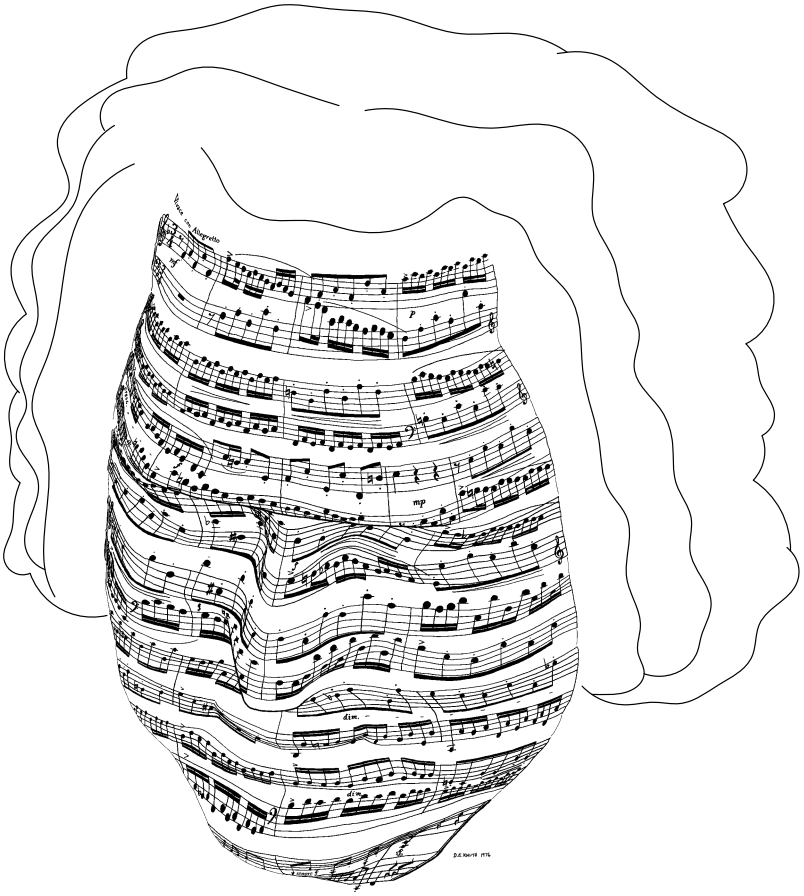
Пожалуйста, еще немного!

— Слоган Bitburger Brauerei** (1951)

---

* Френология — псевдонаука о связи психики человека и строения поверхности его черепа. Создателем френологии является австрийский врач и анатом Ф. Й. Галль (F. J. Gall), который утверждал, что все психические свойства, якобы локализующиеся в полушариях мозга, при развитии вызывают разрастание определенного участка мозга, а это, в свою очередь, — образование выпуклости ("шишки") на соответствующем участке черепа. — *Примеч. пер.*

** Одно из крупнейших пивоваренных предприятий Германии. — *Примеч. пер.*



Hommage à Bach.