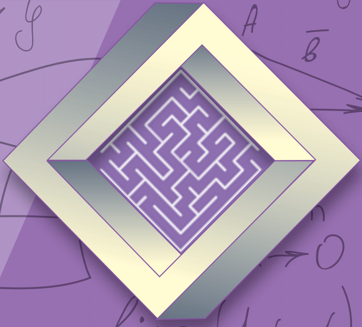
А. ЛЕВИТИН
М. ЛЕВИТИНА

Алгоритмические головоломки



Лаборатория ЗНАНИЙ

АЛГОРИТМИЧЕСКИЕ ГОЛОВОЛОМКИ

ANANY LEVITIN
MARIA LEVITIN

Algorithmic PUZZLES

OXFORD
UNIVERSITY PRESS

АНАНИЙ ЛЕВИТИН
МАРИЯ ЛЕВИТИНА

Алгоритмические ГОЛОВОЛОМКИ

2-е издание

Перевод с английского
Ж. А. Меркуловой,
Н. А. Меркулова

Москва
Лаборатория знаний

УДК 51-028.41+794
ББК 22.1я92
Л36

Левитин А.

Л36 Алгоритмические головоломки / А. Левитин, М. Левитина ; пер. с англ. Ж. А. Меркуловой, Н. А. Меркулова. — 2-е изд. — М. : Лаборатория знаний, 2019. — 325 с. : ил.
ISBN 978-5-00101-188-0

Книга является уникальной коллекцией 150 головоломок, каждая из которых снабжена указанием и решением. Задачи сгруппированы в зависимости от уровня сложности. Издание дополнено двумя обучающими разделами по стратегиям разработки и анализа алгоритмов.

В настоящее время алгоритмические головоломки часто используются на собеседованиях при приеме на работу. Они призваны развить аналитическое мышление и просто разнообразить досуг.

Для всех любителей математики.

УДК 51-028.41+794
ББК 22.1я92

12+

Научно-популярное издание

Левитин Ананий, **Левитина** Мария

АЛГОРИТМИЧЕСКИЕ ГОЛОВОЛОМКИ

Ведущий редактор *М. С. Стригунова*. Художник *В. А. Прокудин*
Оригинал-макет подготовлен *О. Г. Лапко* в пакете $\LaTeX 2_\varepsilon$

Подписано в печать 29.11.18. Формат 60×90/16.
Усл. печ. л. 20,5. Заказ

Издательство «Лаборатория знаний»
125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272, e-mail: info@pilotLZ.ru,
http://www.pilotLZ.ru

ISBN 978-5-00101-188-0

Посвящается Максу с любовью

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ В ВОПРОСАХ И ОТВЕТАХ

О ЧЕМ ЭТА КНИГА?

Эта книга представляет собой сборник алгоритмических головоломок — головоломок, для решения которых требуются, в явном или неявном виде, чётко определённые процедуры решения задач. Книга представляет собой уникальный сборник таких головоломок. В неё включены как старые классические задачи, вошедшие в фольклор математиков и специалистов по информатике, так и новые головоломки, которые предлагают решить во время собеседования в крупных компаниях при приёме на работу.

У этой книги есть две цели:

- развлечь широкий круг читателей, интересующихся головоломками;
- способствовать развитию алгоритмического мышления на высоком уровне (без компьютерного программирования) с помощью ознакомления с основными стратегиями разработки алгоритмов и методами анализа, которые были тщательно отобраны авторами.

Алгоритмы — это краеугольный камень информатики, и программирование без них невозможно. Однако было бы неправильно, как это делают многие, ставить знак равенства между алгоритмом и компьютерной программой. Некоторые алгоритмические головоломки старше компьютеров более чем на тысячу лет. Однако следует признать, что распространение компьютеров сделало доступным решение алгоритмических задач во многих областях современной жизни, от естественных и гуманитарных наук до искусства и индустрии развлечений. Решение алгоритмических головоломок — это наиболее продуктивный и уж точно самый приятный способ развить и улучшить навыки алгоритмического мышления.

ДЛЯ КОГО ЭТА КНИГА?

Эта книга может быть интересна для трёх больших категорий читателей:

- любителей головоломок;
- тех, кто хочет развить алгоритмическое мышление, в том числе учителей и учащихся;

- тех, кто готовится к собеседованию при приёме на работу, во время которого предлагают головоломки, а также тех, кто проводит эти собеседования.

Любителям головоломок понравится этот сборник, содержащий головоломки разных типов и тем. Они встретят как головоломки, любимые во все времена, так и малоизвестные «жемчужины». Никаких знаний по информатике или даже интереса к ней не требуется. Читатель, не обладающий такими знаниями, может просто пропустить описание стратегий разработки и анализа алгоритмов в разделе «Решения».

В последнее время термин «алгоритмическое мышление» очень часто употребляется преподавателями по информатике, и не без оснований: повсеместное распространение компьютеров в современном мире действительно делает необходимым овладение навыками алгоритмического мышления практически для каждого студента. Головоломки — идеальное средство для приобретения этих важных навыков в силу двух причин. Во-первых, головоломки — это занятно и человек обычно готов приложить больше усилий для решения головоломок, чем для решения обычных скучноватых упражнений. Во-вторых, алгоритмические головоломки заставляют мыслить на более абстрактном уровне. Даже студенты, обучающиеся информатике, чаще думают о решении алгоритмических задач в терминах компьютерного языка, который они знают, и не пытаются применить общие стратегии разработки и анализа алгоритмов. Головоломки помогают восполнить этот существенный недостаток.

Головоломки в этой книге, безусловно, можно использовать для самостоятельного обучения. Вместе с обучающим материалом они, на наш взгляд, позволяют ознакомиться с основными алгоритмическими понятиями и методами. Головоломки могут использовать и преподаватели компьютерных курсов как в университетах, так и в средней школе, в качестве вспомогательных упражнений и материала для проектов. Эта книга также может быть интересна для обучения на курсах по принятию решений, особенно на тех курсах, которые основаны на решении головоломок.

Для людей, готовящихся к собеседованию, эта книга будет полезна по двум причинам. Во-первых, в ней есть много примеров головоломок, с которыми они могут встретиться, с полным решением и комментариями. Во-вторых, в книге содержится краткий учебный материал по стратегиям разработки и методам анализа алгоритмов. По словам менеджеров, предлагающих на собеседовании решить головоломки, для них в большей степени важно увидеть правильный подход к решению головоломки, а не само решение. На потенциального работодателя произведёт очень хорошее впечатление, если вы продемонстрируете ваш опыт в применении общих стратегий разработки алгоритмов.

КАКИЕ ГОЛОВОЛОМКИ ВКЛЮЧЕНЫ В КНИГУ?

Алгоритмические головоломки составляют малую часть среди тысяч математических головоломок, придуманных на протяжении многих лет. Подбирая головоломки для этой книги, мы руководствовались следующими критериями.

Во-первых, мы хотели, чтобы головоломки иллюстрировали некоторые общие принципы разработки и анализа алгоритмов.

Во-вторых, мы хотели, чтобы они были красивыми и элегантными, хотя эти качества, конечно, субъективны.

В-третьих, мы хотели, чтобы головоломки были различного уровня сложности. Сложность головоломки трудно точно определить; иногда головоломки, которые легко решают ученики средней школы, ставят в тупик профессоров математики. Тем не менее, мы поделили нашу книгу на три раздела — головоломки лёгкие, средней сложности и сложные — чтобы помочь читателю оценить уровень головоломки. В пределах каждого раздела мы также отсортировали головоломки по уровню сложности. Для решения головоломок из раздела «Лёгкие головоломки» нужна только математика средней школы. Для решения некоторых задач из следующих двух разделов применяется метод индукции, но в общем, чтобы решить все головоломки в книге, будет достаточно школьной математики. Кроме того, во второй части учебного раздела вкратце освещаются такие темы, как бинарные числа и простые рекуррентные соотношения. Конечно, это не означает, что все головоломки в книге — лёгкие. Некоторые из них — особенно в конце последнего раздела — являются по-настоящему сложными. Но их трудность вовсе не в том, что они требуют сложной математики, поэтому они не должны отпугивать читателя.

В-четвёртых, мы чувствовали себя обязанными включить несколько головоломок, имеющих историческое значение.

В заключение отметим, что мы включили в книгу головоломки только с ясной постановкой задачи и решением, избегая таких трюков, как преднамеренная двусмысленность, игра слов и т. д.

Нужно сделать ещё одно замечание. Многие головоломки из этой книги можно решить с помощью полного перебора или поиска с возвратом (эти стратегии описаны в первой части учебного раздела). Однако, это *не тот* подход, которым нужно наслаждаться при решении этих головоломок. Он требуется только в случаях, когда это специально указано. Поэтому мы исключили такие категории головоломок, как судоку и криптарифмы, которые можно решить или полным перебором (перебором с возвратом), или же благодаря гениальной проницательности и озарению. Мы также решили не включать головоломки, относящиеся к физическим объектам, которые не очень легко описать, такие как Китайские кольца и кубик Рубика.

ПОДСКАЗКИ, РЕШЕНИЯ И КОММЕНТАРИИ

В книге для каждой головоломки приведены подсказка, решение и комментарии. Книги по головоломкам редко включают подсказки, но мы считаем их ценным дополнением. Они могут слегка подтолкнуть в правильном направлении, оставляя, тем не менее, шанс читателю самому решить головоломку. Все подсказки собраны в отдельном разделе.

К каждой головоломке приведено решение. Как правило, оно начинается с короткого ответа. Это сделано для того, чтобы дать читателю последнюю возможность самому решить головоломку: если ответ читателя не тот, что в решении, он может прекратить чтение решения и попытаться снова решить головоломку.

Алгоритмы описаны без использования специального формата или псевдокода. Акцент делается на идеях, а не на незначительных деталях. Переписать решения в более формальном виде — это уже само по себе полезное упражнение.

Большинство комментариев обращают внимание на общую идею алгоритма, которую демонстрируют головоломка и её решение. Иногда в комментариях также есть ссылки на похожие головоломки из этой книги или из других источников.

В большинстве книг, посвящённых головоломкам, не указывается происхождение головоломки. Обычно это объясняется тем, что пытаться найти автора головоломки — всё равно, что пытаться найти автора шутки. Хотя в этом и есть значительная доля правды, мы решили упомянуть о самых ранних источниках головоломок, которые нам известны. Однако читатель должен иметь в виду, что мы не производили хоть сколько-нибудь серьёзного поиска источников головоломок. Если бы мы это сделали, получилась бы совсем другая книга.

ЧТО ПРЕДСТАВЛЯЕТ СОБОЙ УЧЕБНЫЙ РАЗДЕЛ?

В книгу включён учебный раздел, где на примерах головоломок описаны общие стратегии разработки и методы анализа алгоритмов. Хотя почти все головоломки в книге можно решить без малейшего знания тем, освещённых в учебном разделе, вне сомнения он поможет решить головоломки гораздо легче и, что важно, с большей пользой. Кроме того, в решениях, комментариях и некоторых подсказках используется та же терминология, объяснение которой даётся в учебном разделе.

Учебный раздел написан на максимально доступном уровне, чтобы быть понятным широкому кругу читателей. Если читатель — специалист по информатике, то он вряд ли найдёт там для себя что-то новое, разве что примеры головоломок. В то же время, такой читатель может быстро освежить свои знания фундаментальных идей разработки и анализа алгоритмов.

ПОЧЕМУ В КНИГЕ ДВА УКАЗАТЕЛЯ?

В дополнение к стандартному указателю в книге есть ещё указатель, где головоломки сгруппированы по стратегии разработки или типу анализа соответствующего алгоритма. Этот указатель призван помочь читателю соотнести задачу с конкретной стратегией или методом решения, тем самым являясь дополнительной подсказкой.

Мы надеемся, что читатели найдут книгу как занимательной, так и полезной. Мы также надеемся, что они разделят наше восхищение красотой и удивительной человеческой изобретательностью, которые видны во многих головоломках этой книги.

Ананий Левитин, Мария Левитина
Май 2011 г.
algorithmicpuzzles.book@gmail.com

БЛАГОДАРНОСТИ

Мы хотели бы выразить нашу глубокую благодарность рецензентам книги: Тиму Шартье (Колледж Дэвидсона), Стивену Лукасу (Университет Джеймса Мэдисона) и Лоре Таалман (Университет Джеймса Мэдисона). Их горячая поддержка идеи книги и конкретные предложения по её содержанию были, безусловно, нам очень полезны.

Мы также благодарны Симону Берковичу из Университета Джорджа Вашингтона за обсуждение тем головоломок и за чтение части рукописи книги.

Мы благодарны всем сотрудникам издательства *Oxford University Press* и их коллегам, которые работали над книгой. Мы особенно благодарны нашему редактору Филлис Коэн за её неустанные усилия сделать книгу лучше. Мы также благодарим помощника редактора Халли Стеббинс, дизайнера обложки Наталью Балнову и менеджера по продажам Мишель Келли. Мы ценим работу Ричарда Кампа, редактора, подготовившего рукопись к печати, а также усилия Дженифер Коуинг и Кирана Кумара, которые отвечали за выпуск книги.

СПИСОК ГОЛОВОЛОМОК

ГОЛОВОЛОМКИ УЧЕБНОГО РАЗДЕЛА

Данный список содержит все головоломки, включённые в учебный раздел. Головоломки перечислены в том порядке, в котором они появляются в книге. Номер страницы указывает на ту страницу, где описана сама головоломка, а их решения даны непосредственно в учебном разделе, следом за формулировкой.

ГОЛОВОЛОМКИ ОСНОВНОГО РАЗДЕЛА

В этом списке указаны все 150 головоломок, включённых в основной раздел книги. Три номера после головоломки указывают на страницы, где приведены условие, подсказка и решение соответственно.

ГОЛОВОЛОМКА В КАЧЕСТВЕ ЭПИГРАФА: КТО ЭТО СКАЗАЛ?

Определите, какому автору из перечисленных ниже принадлежат эти цитаты.

Человек с молотком подходит к каждой проблеме, как к забиванию гвоздей. Большой молоток нашего века — это алгоритм.

Решение задач — это практический навык, как, скажем, плавание. Все практические навыки мы приобретаем с помощью подражания и практики.

Нет лучшего способа развеять скуку, как ввести в курс занимательные темы и элементы развлечения — игру, юмор, красоту и сюрприз.

Не само знание, но процесс обучения, не обладание, но движение к цели, — вот что гарантирует настоящее наслаждение.

Если я случайно упустил что-нибудь более или менее уместное или необходимое, я прошу снисходительности, поскольку нет ни одного человека, который совершенно безупречен и предусмотрителен по отношению ко всем вещам.

Уильям Паундстоун,
автор книги *«Как сдвинуть гору Фудзи? Подходы ведущих мировых компаний к поиску талантов»*.

Дьёрдь Пойа (1887–1985),
выдающийся венгерский математик, автор книги *«Как решать задачу»*, классической книги по решению задач.

Мартин Гарднер (1914–2010),
американский писатель, особенно хорошо известный по рубрике «Математические игры» в журнале *Scientific American* и книгам по занимательной математике.

Карл Фридрих Гаусс (1777–1855),
великий немецкий математик.

Леонардо Пизанский (Фибоначчи) (1170 – ок. 1250),
выдающийся итальянский математик, автор *Liber Abaci* («*Книга абака*»), одной из важнейших книг по математике в истории.

Глава 1

УЧЕБНЫЙ РАЗДЕЛ

ОБЩИЕ СТРАТЕГИИ РАЗРАБОТКИ АЛГОРИТМОВ

Цель данного раздела — дать краткий обзор общих стратегий разработки алгоритмов. Хотя каждая из этих стратегий применима не ко всем головоломкам, все вместе они представляют собой мощный набор средств. Неудивительно, что эти стратегии используются также для решения многих задач в информатике. Таким образом, обучение тому, как применять эти стратегии к решению головоломок, может служить прекрасным введением в эту важную область.

Прежде чем мы приступим к рассмотрению основных стратегий разработки алгоритмов, нужно сделать важное замечание о двух типах алгоритмических головоломок. В каждой головоломке есть входные данные, по которым определяется, о каком *случае* задачи идёт речь. Случай может быть либо частным (например, найти фальшивую монету среди восьми монет с помощью взвешивания) или общим (например, найти фальшивую монету среди n монет путём взвешивания). Если читатель имеет дело с частным случаем головоломки, он не обязан решать её для общего случая. Иногда другие случаи имеют другое решение или даже вообще не имеют решения. Но с другой стороны, конкретное число в формулировке головоломки может вовсе не иметь никакого значения. Тогда решение головоломки для общего случая не только принесёт большее удовлетворение, но может оказаться проще. Но, независимо от того, задана ли головоломка для частного случая или в общей форме, всегда полезно решить её для нескольких простых случаев. Иногда это может навести читателя на неверный путь, но гораздо чаще будет способствовать более глубокому пониманию задачи.

Полный перебор

Теоретически многие головоломки можно решить с помощью полного перебора — стратегии решения задач, при которой в условие задачи просто подставляются все возможные варианты решения, пока решение не будет найдено. При применении полного перебора обычно не нужно много изобретательности. Поэтому человеку (в отличие от компьютера) редко предлагают решить головоломку таким способом. При применении полного перебора самым главным ограничением является его неэффективность: как правило, число возможных вариантов, которые надо перебрать, растёт по меньшей мере экспоненциально с увеличением размера задачи, тем самым делая такой подход нецелесообразным не только для человека, но и для компьютера. В качестве примера рассмотрим задачу построения *магического квадрата* третьего порядка.

Магический квадрат. Заполните таблицу 3×3 девятью различными целыми числами от 1 до 9 так, чтобы сумма чисел в каждой строке, каждом столбце и двух главных диагоналях была одинакова (рис. 1.1).

?	?	?
?	?	?
?	?	?

Рис. 1.1. Таблица 3×3, которую нужно заполнить целыми числами от 1 до 9, чтобы получился магический квадрат

Сколькими способами можно заполнить такую таблицу? Допустим, мы ставим за один раз одно число, сначала поставим где-нибудь 1, а в самом конце — 9. Есть девять способов поставить 1, восемь способов поставить 2 и т. д.; последнюю цифру 9 можно поставить в единственную оставшуюся свободной ячейку в таблице. Таким образом, существует $9! = 9\cdot8\cdot\ldots\cdot1 = 362\,880$ способов разместить девять чисел в таблице 3×3. (Мы только что использовали стандартное обозначение $n!$, называемое *n факториал*, для произведения последовательных целых чисел от 1 до n.) Таким образом, если решать эту задачу путём полного перебора, пришлось бы расставить числа от 1 до 9 в таблице 362 880 возможными способами и для каждого случая проверить, не является ли сумма в каждой строке, столбце и главной диагонали одинаковой. Такой объём работы вне сомнения невозможно сделать вручную.

На самом деле, нетрудно решить эту головоломку, доказав сначала, что значение суммы равняется 15 и что нужно поместить 5 в центральную ячейку (см. головоломку «*Ещё раз о магическом квадрате*», № 29 в основном разделе книги). Кроме того, можно воспользоваться несколькими известными алгоритмами построения магических квадратов произвольного порядка $n \geqslant 3$, что особенно эффективно для нечётных n (см., например, [Pic02]). Конечно же, эти алгоритмы не основаны на полном переборе: количество вариантов решения, которое нужно проверить перебором, становится непомерно большим уже для n, равного всего лишь 5. Действительно, $(5^2)! \cong 1,5 \cdot 10^{25}$ и, следовательно, компьютеру, который делает 10 триллионов операций в секунду, понадобится около 49 000 лет, чтобы закончить работу.

Поиск с возвратом

В методе полного перебора есть две основные проблемы. Первая заключается в механизме определения всех возможных вариантов решения. Для некоторых задач мы можем составить хорошо структурированный набор вариантов. Например, варианты расстановки первых девяти положительных целых чисел в ячейках таблицы 3×3 (см. выше пример «*Магического квадрата*») можно получить с помощью перестановок этих чисел; для поиска же всех перестановок известно несколько алгоритмов. Однако во многих задачах варианты решения не образуют такую регулярную структуру. Вторая, и более сложная, проблема заключается в количестве вариантов, которые нужно проверить. Как правило, это количество растёт по крайней мере экспоненциально с увеличением размера задачи. Поэтому полный перебор разумно применять только для задач маленького размера.

Поиск с возвратом гораздо лучше по сравнению с примитивным «лобовым» подходом полного перебора. Это удобный метод сгенерировать возможные варианты решения, при этом исключая ненужные варианты. Главная его идея состоит в том, чтобы построить варианты решения по принципу «один компонент за один раз» и оценить такие частично построенные варианты решения следующим образом: если такой частично построенный вариант не нарушает ограничений задачи, то далее подбирается первый из допустимых вариантов для следующего компонента решения. Если для следующего компонента нет допустимого варианта, то не нужно рассматривать варианты ни для какого из оставшихся компонентов. В таком случае алгоритм возвращается на предыдущий шаг и заменяет последний компонент частично построенного решения на следующий из возможных вариантов.

Поиск с возвратом, в принципе, предполагает отсечение определённого количества неверных вариантов — чем больше это количество, тем быстрее алгоритм находит решение. В худшем варианте поиск с возвратом может в итоге сгенерировать такое же количество вариантов решения, как и полный перебор, но это происходит редко.

Поиск с возвратом можно интерпретировать как процесс построения дерева, на котором видны принимаемые решения. В компьютерных науках термин «дерево» используется для описания иерархических структур (например, генеалогическое древо, организационные схемы). *Дерево* обычно изображают следующим образом: *корень* (единственная узловая точка без *родителей*) сверху, а его *листья* (узлы без *детей*) — внизу диаграммы. Это просто удобное общепринятое изображение. Для перебора с возвратом такое дерево называется *деревом пространства состояний*. Корень этого дерева соответствуют началу процесса построения решения; мы считаем, что корень — нулевой уровень дерева. Дети корня — на первом уровне дерева — соответствуют возможным вариантам выбора первого компонента решения (например, ячейка в магическом квадрате содержит 1). Их дети — узлы на втором уровне — соответствуют возможным вариантам следующего компонента решения, и т. д. Листья могут быть двух видов. Первый вид — *бесперспективные узлы*, или *тупики* — представляют частично построенный вариант, который не ведёт к окончательному решению. Установив, что данный узел бесперспективен, алгоритм перебора с возвратом закрывает его (часть дерева обрезается), отменяет решение о последнем компоненте, возвращается обратно к *родителю* бесперспективного узла и рассматривает другой вариант выбора для этого компонента. Второй вид листьев даёт решение задачи. Если достаточно одного решения, алгоритм останавливается. Если нужны ещё решения, алгоритм продолжает поиск, возвращаясь к родителю листа.

Следующий пример является классическим для иллюстрации применения поиска с возвратом для конкретной задачи.

Задача об *n* ферзях. Разместите n ферзей на шахматной доске $n \times n$ таким образом, чтобы ни один из них не находился под боем другого. (Один ферзь бьёт другого, если находится с ним на одной линии по вертикали, горизонтали или диагонали.)

При $n = 1$ задача имеет тривиальное решение, и легко увидеть, что при $n = 2$ и $n = 3$ решения нет. Поэтому рассмотрим задачу о 4 ферзях и решим её с помощью поиска с возвратом. Поскольку каждого из четырёх ферзей нужно поместить на свою вертикаль, всё, что нам необходимо сделать, — определить для каждого ферзя клетку по горизонтали шахматной доски, показанной на рис. 1.2.

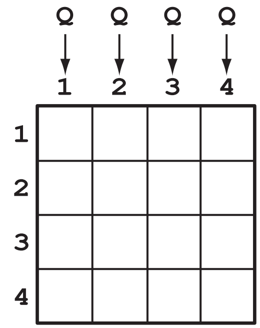


Рис. 1.2. Задача о 4 ферзях на шахматной доске

Начнём с пустой доски и поставим ферзя 1 в первую возможную позицию — позицию на горизонтали 1 и на вертикали 1. Затем ставим ферзя 2: после неуспешных попыток поставить его на горизонталях 1 и 2 и на второй вертикали помещаем его в первую возможную позицию — на клетку (3, 2), т. е. клетку на горизонтали 3 и на вертикали 2. Это оказывается тупиком, поскольку для ферзя 3 нет подходящей позиции на вертикали 3. Поэтому алгоритм возвращается назад и размещает ферзя 2 в следующую возможную позицию (4, 2). Затем ферзь 3 ставится на клетку (2, 3), что также оказывается тупиком. Алгоритм возвращается назад на все шаги к ферзю 1 и размещает его в позицию (2, 1). Ферзь 2 затем идёт на (4, 2), ферзь 3 на (1, 3) и ферзь 4 на (3, 4), что даёт решение задачи. Дерево пространства состояний данного поиска показано на рис. 1.3.

Если необходимо найти другие решения (а для задачи о 4 ферзях есть ещё одно решение), алгоритм просто возобновляет свои шаги с того листа дерева состояний, на котором он был остановлен. В качестве альтернативы для поиска второго решения в данном случае можно использовать симметричность доски.

Насколько быстрее можно найти решение с помощью поиска с возвратом по сравнению с полным перебором? Количество всех возможных позиций для четырёх ферзей на четырёх различных клетках доски 4×4 равно

$$\frac{16!}{4!(16-4)!} = \frac{16 \cdot 15 \cdot 14 \cdot 13}{4 \cdot 3 \cdot 2} = 1820.$$

Это общая формула, описывающая количество способов, которыми можно выбрать k различных элементов (неважно, в каком порядке) из множества n различных элементов. Это количество называется *сочетанием* из *n по k*, обозначается $\binom{n}{k}$ или $C(n,k)$ и равно $\frac{n!}{k!(n-k)!}$.

Если мы рассмотрим размещение ферзей только в различных колонках доски, общее количество возможных решений уменьшается до $4^4 = 256$. Если мы ещё добавим ограничение, что ферзи должны быть и в различных позициях по

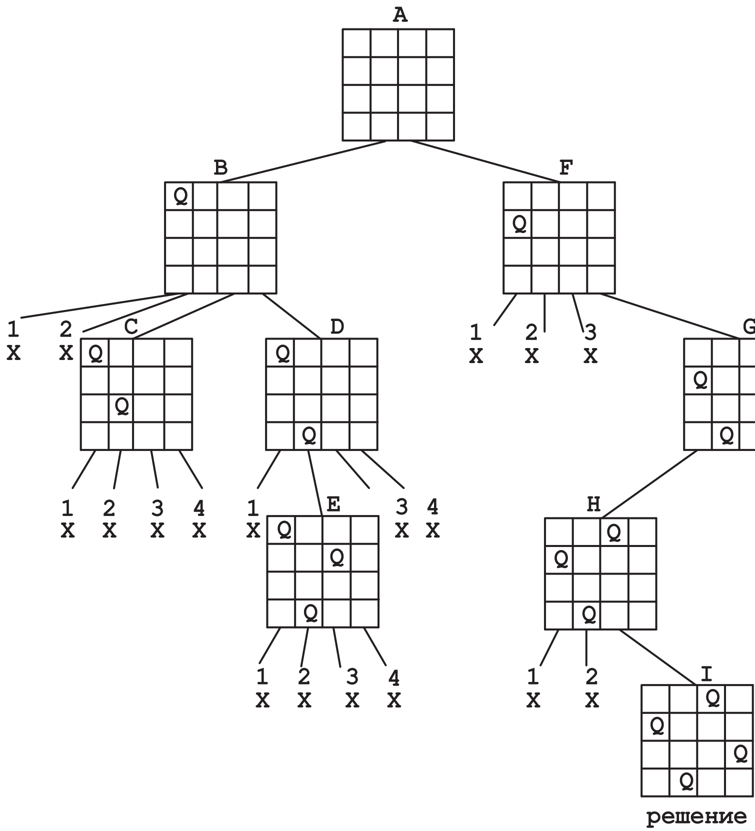


Рис. 1.3. Дерево пространства состояний, показывающее решение задачи о 4 ферзях (ферзи обозначены буквой Q) методом поиска с возвратом. Буква X обозначает неудачные попытки поместить ферзя на указанной вертикали. Буквы над узлами показывают, в какой последовательности генерируются узлы

горизонтали, количество вариантов уменьшается до $4! = 24$. Это количество вполне приемлемо, но не для более сложных частных случаев. Например, для обычной шахматной доски 8×8 количество возможных решений $8! = 40\,320$.

Читателю может быть интересно, что общее количество различных решений задачи о 8 ферзях равно 92, 12 из них качественно различны, а 80 других получаются из основных двенадцати с помощью поворотов и отражений. Что касается общей задачи об n ферзях, она имеет решение для любого $n \geqslant 4$; однако подходящей формулы, описывающей количество решений для любого n, не найдено. Известно, что количество решений очень быстро растёт с увеличением n. Например, количество решений при $n = 10$ равно 724, из них 92 качественно различаются, а при $n = 12$ эти количества равны 14 200 и 1787 соответственно.

С помощью поиска с возвратом можно решить многие головоломки из этой книги. Однако для каждой из них есть более эффективный алгоритм, который и должен найти читатель. В частности, для решения головоломки «*Снова задача об n ферзях*» (№ 140) из основного раздела книги нужно найти гораздо более быстрый алгоритм.

Уменьшай и властвуй

Стратегия «*уменьшай и властвуй*» *(метод уменьшения размера задачи — прим. ред.)* основана на нахождении соотношения между решением данной задачи и решением её более простого частного случая. Когда это соотношение найдено, оно естественным образом приводит к рекурсивному алгоритму, который последовательно сводит задачу ко всё более и более простым частным случаям до тех пор, пока частный случай не становится простым настолько, что его можно решить.[1] Приведём пример.

Задача о знаменитости. Знаменитость в группе из n людей — это человек, который не знает никого, но которого знают все. Задача — найти знаменитость, задавая людям один вопрос «Ты знаешь этого человека?»

Пусть, для простоты, нам известно, что знаменитость находится в группе из n людей. Задача может быть решена следующим образом с помощью алгоритма «уменьшить на единицу». Если $n = 1$, то этот единственный человек — знаменитость просто по определению. Если $n > 1$, выбираем двух человек из группы, например A и B, и спрашиваем A, знает ли он B. Если A знает B, удаляем A из группы людей, которые могут быть знаменитостью. Если A не знает B, удаляем B из группы. Затем решаем задачу рекурсивно (т. е. тем же методом) для оставшейся группы из $n - 1$ человек.

[1] Понятие *рекурсии* является одним из самых важных в информатике. Если читатель не знаком с ним, он может найти достаточно информации, например, в Википедии, по ссылкам в статье «Рекурсия (информатика)».

В качестве лёгкого упражнения читатель может решить головоломку «*Отряд солдат*» (№ 4) в основном разделе книги.

В принципе, более простые частные случаи в стратегии «*уменьшай и властвуй*» необязательно должны получаться при уменьшении n на 1. Хотя «уменьшить на единицу» это наиболее распространённый метод уменьшения, есть примеры уменьшения и на большее число. Можно получить гораздо более быстрый алгоритм, если нам удастся на каждом шаге уменьшать размер в постоянное число раз, например вдвое. Хорошо известный пример этого алгоритма применяется в следующей игре.

Угадай число (двадцать вопросов). Угадать число, находящееся в пределах от 1 до n, задавая вопросы, на которые можно ответить «да» и «нет».

Наиболее быстрый алгоритм для решения этой задачи — задавать вопрос, ответ на который уменьшает диапазон чисел, содержащих задуманное число, примерно в два раза на каждом шаге. Например, первый вопрос может быть, больше ли задуманное число, чем $\left\lceil \frac{n}{2} \right\rceil$ (это принятое обозначение для $\frac{n}{2}$, округлённого до ближайшего целого числа)[1]. Если ответ «нет», то задуманное число находится между целыми числами 1 и $\left\lceil \frac{n}{2} \right\rceil$. Если ответ «да», то задуманное число — между целыми числами $\left\lceil \frac{n}{2} \right\rceil + 1$ и n. В любом случае алгоритм уменьшает задачу размера n до частного случая этой же задачи размером примерно вдвое меньше изначальной. Повторяем этот шаг до тех пор, пока размер задачи уменьшится до 1, и задача решена.

Поскольку этот алгоритм уменьшает размер частного случая задачи (диапазон чисел, в которых содержится задуманное число) примерно в два раза на каждом шаге, он работает потрясающе быстро. Например, при $n = 1\,000\,000$ для алгоритма нужно не более 20 вопросов! Ещё быстрее был бы алгоритм, который уменьшает размер задачи в большее число раз, например в три раза.

[1]Число $\lceil x \rceil$ называется потолком действительного числа x и равняется наименьшему целому числу, которое больше или равно x. Например, $\lceil 2{,}3 \rceil = 3$; $\lceil 2 \rceil = 2$. Число $\lfloor x \rfloor$ называется полом x и равняется наибольшему целому числу, меньшему или равному x. Например, $\lfloor 2{,}3 \rfloor = 2$; $\lfloor 2 \rfloor = 2$.

Головоломка «*Фальшивая монета из восьми*» (№ 10) из основного раздела книги иллюстрирует стратегию «уменьшай в постоянное число раз», вариант стратегии «уменьшай и властвуй». Эта головоломка может служить хорошим упражнением.

Нужно отметить, что иногда проще установить соотношение между бóльшим и меньшим случаями в обратном порядке. Это означает, что нужно сначала решить головоломку для самого маленького частного случая, потом следующего случая и т. д. Этот метод иногда называют «подходом по возрастающей». Примером является первое решение головоломки «*Разделение прямоугольника*» (№ 3) в основном разделе книги.

Разделяй и властвуй

Стратегия «*разделяй и властвуй*» (*метод декомпозиции — прим. ред.*) заключается в том, чтоб разделить задачу на несколько более лёгких подзадач (обычно такого же или похожего типа и желательно одного размера), решить каждую из них и, если это необходимо, скомбинировать их решения, чтобы получить решение исходной задачи. Эта стратегия лежит в основе многих эффективных алгоритмов, использующихся для решения важных задач в информатике. Удивительно, но с помощью алгоритмов «разделяй и властвуй» можно решить не так уж много головоломок. Приведём, однако, хорошо известный пример, который идеально демонстрирует эту стратегию.

Головоломка «Тримино». Замостить доску размером $2^n \times 2^n$, у которой нет одной клетки, угловыми тримино, т. е. фигурками L-образной формы, состоящими из трёх примыкающих друг к другу квадратов. Отсутствовать может любая из клеток доски. Тримино должны покрыть все клетки, кроме отсутствующей, без наложения друг на друга.

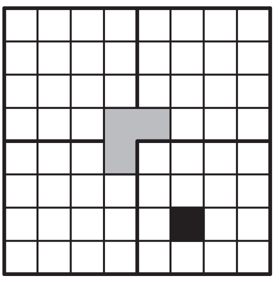


Рис. 1.4. Первый шаг алгоритма «разделяй и властвуй» в задаче покрытия фигурками тримино доски размером $2^n \times 2^n$, на которой отсутствует одна клетка

Задача может быть решена при помощи рекурсивного алгоритма «разделяй и властвуй». Поместим одно тримино в центр доски таким образом, что задача для случая n упрощается до четырёх случаев $n - 1$ (рис. 1.4).

Алгоритм останавливается после того, как каждая часть доски размером 2×2 с одной отсутствующей клеткой покрыта одним тримино.

С помощью этого алгоритма читатель может в качестве быстрого, но полезного упражнения завершить покрытие доски размером 8×8, изображённой на рис. 1.4.

Большинство алгоритмов «разделяй и властвуй» решают меньшие подзадачи рекурсивно, поскольку, как в приведённом выше примере, они представляют собой меньшие частные случаи той же задачи. Однако так бывает не всегда. Для решения некоторых задач о досках, например, доску нужно поделить на части («поддоски»), которые необязательно меньшие версии изначальной. В качестве примеров решите головоломки «*Задача о 2n шашках*» (№ 37) и «*Замощение прямыми тримино*» (№ 78) из основного раздела книги.

О стратегии «разделяй и властвуй» необходимо сделать ещё одно замечание. Хотя некоторые считают стратегию «уменьшай и властвуй» (которая описана выше) особым случаем «разделяй и властвуй», лучше рассматривать её как отдельную стратегию. Принципиальная разница между ними заключается в количестве меньших подзадач, которые нужно решить на каждом шаге: несколько подзадач в алгоритмах типа «разделяй и властвуй» и всего одну в алгоритмах «уменьшай и властвуй».

Преобразуй и властвуй

«Преобразуй и властвуй» — это широко известный подход к решению задач, который основан на идее преобразования. Задача решается в два этапа. Сначала, на этапе преобразования, задача модифицируется или преобразуется в другую задачу, которую легче решить в силу тех или иных причин. На следующем этапе «властвования» она решается. В области решения задач с помощью алгоритмов можно выделить три варианта этой стратегии. Первый вариант называется *упрощение частного случая*; при этом задача решается путём преобразования исходного частного случая в другой частный случай той же задачи, который обладает некоей особенностью, позволяющей решить задачу легче. Второй вариант, называемый *изменение представления*, основан на преобразовании входа задачи в более подходящий для эффективного решения с помощью алгоритма. Третьим вариантом стратегии преобразования является *упрощение задачи*, при котором частный случай исходной задачи преобразуется в частный случай другой задачи.

В качестве первого примера приведём задачу-головоломку из книги Джона Бентли «Жемчужины программирования» [Ben00, p. 15–16].

Поиск анаграмм. Анаграммы — это слова, состоящие из одних и тех же букв. Например, слова «воз» и «зов», «атлас» и «салат». Нужно разработать алгоритм для поиска всех анаграмм в большом списке слов.

Эффективный алгоритм для решения этой задачи работает в две стадии. Сначала он присваивает каждому слову «ключ», состоящий из букв этого слова, расположенных в алфавитном порядке (изменение представления), а затем сортирует полученные «ключи» по алфавиту (сортировка данных относится к упрощению частного случая). Таким образом, анаграммы в полученном списке стоят рядом.

В качестве упражнения читателю предлагается решить головоломку «*Расстановка чисел*» (№ 43), основанную на этой же идее.

Другой пример изменения представления, который иногда бывает полезен, — это представить вход задачи в двоичной или троичной системе счисления. Если читатель не знаком с этой важной темой, дадим краткие пояснения. В десятичной позиционной системе, которую бóльшая часть человечества использует на протяжении последних восьмисот лет, целое число представляется в виде комбинации степеней числа 10. Например, $1069 = 1 \cdot 10^3 + 0 \cdot 10^2 + 6 \cdot 10^1 + 9 \cdot 10^0$. В двоичной и троичной системах число представляется в виде комбинации степеней чисел 2 и 3 соответственно. Например, в двоичной системе $1069_{10} = 10000101101_2$, потому что $1069 = 1{\cdot}2^{10}+0{\cdot}2^9+0{\cdot}2^8+0{\cdot}2^7+0{\cdot}2^6+1{\cdot}2^5+0{\cdot}2^4+1{\cdot}2^3+1{\cdot}2^2+0{\cdot}2^1+1{\cdot}2^0$. А в троичной системе $1069_{10} = 1110121_3$, поскольку $1069 = 1 \cdot 3^6 + 1 \cdot 3^5 + 1 \cdot 3^4 + 0 \cdot 3^3 + 1 \cdot 3^2 + 2 \cdot 3^1 + 1 \cdot 3^0$. Десятичное число записывается десятью цифрами (от 0 до 9), двоичное число — двумя (0 и 1), троичное число — тремя (0, 1 и 2). Каждое десятичное целое число имеет своё уникальное представление в каждой из этих систем. Это представление можно получить, последовательно деля число на 2 и 3 соответственно. Двоичная система является особенно важной, поскольку она оказалась наиболее удобной для компьютерной обработки данных.

В качестве примера головоломки, для решения которой используется преимущество двоичной системы, рассмотрим задачу из книги У. Паундстоуна [Pou03, p. 84].

Конверты с банкнотами. У вас есть тысяча банкнот по 1 доллару. Разложите их по 10 конвертам таким образом, чтобы можно

было выдать любую сумму от 1 до 1000 долларов, комбинируя эти конверты. Конечно же, выдача сдачи не предполагается.

Положим банкноты на сумму 1, 2, $2^2, \ldots, 2^8$ в первые девять конвертов, а оставшуюся сумму $1000 - (1+2+\cdots+2^8) = 489$ в десятый конверт. Любое число A, меньшее 489, может быть получено как комбинация степеней числа 2: $b_8 \cdot 2^8 + b_7 \cdot 2^7 + \cdots + b_0 \cdot 1$, где коэффициенты $b_8, b_7, \ldots, b_0$ равны 0 или 1. (Эти коэффициенты составляют представление числа A в двоичной системе. Самое большое число, которое можно представить с помощью девятизначного двоичного числа, это $2^8 + 2^7 + \cdots + 1 = 2^9 - 1 = 511$.) Любое число A от 489 до 1000 включительно можно представить как $489 + A'$, где $0 \leqslant A' \leqslant 511$. Таким образом, нужная сумма денег для такого A может быть получена как содержимое десятого конверта и комбинация первых девяти, которая задаётся двоичным представлением A'. Обратите внимание, что для некоторых A решение головоломки не единственное.

Хорошее упражнение для читателя — решение двух версий головоломки «*Задача Баше о гирях*» (№ 115), в которых используется двоичная и вариация троичной систем соответственно.

И наконец, многие задачи можно решить с помощью графов. *Граф* — это конечное множество точек на плоскости, некоторые из которых соединены линиями. Точки и линии называются *вершинами* и *рёбрами* графа соответственно. Рёбра могут не иметь ориентации или могут быть ориентированы от одной вершины к другой. В первом случае граф называется неориентированным, а во втором — ориентированным графом или кратко орграфом. В применении к головоломкам или играм вершины графа обычно представляют возможные состояния рассматриваемой задачи, а рёбра указывают на допустимые переходы между этими состояниями. Одна из вершин графа представляет исходное состояние, а другая — состояние, которое нужно достичь (таких вершин может быть несколько). Такой граф называется *графом пространства состояний*. Таким образом, представление задачи в виде графа сводит задачу к поиску пути между начальной вершиной и целевыми вершинами.

В качестве примера рассмотрим вариант очень старой и известной головоломки.[1)]

[1)]Классическая версия этой головоломки для трёх супружеских пар была включена в самый ранний из известных сборников математических задач на латыни *Propositiones ad Acuendos Juvenes*

[. . .]

Алгоритмические головоломки

МНОГИЕ СЧИТАЮТ, ЧТО АЛГОРИТМЫ ОТНОСЯТСЯ ИСКЛЮЧИТЕЛЬНО К ОБЛАСТИ ИНФОРМАТИКИ, НО СУТЬ АЛГОРИТМИЧЕСКОГО МЫШЛЕНИЯ – ИСПОЛЬЗОВАНИЕ АНАЛИТИЧЕСКОЙ ЛОГИКИ ДЛЯ РЕШЕНИЯ ЗАДАЧ.

Эта логика распространяется далеко за пределы информатики в обширный и интересный мир головоломок. В своей книге «Алгоритмические головоломки» Ананий и Мария Левитины показывают, как применять аналитическое мышление для решения головоломок с помощью строго определенных методов, используя в качестве примеров как классические головоломки, так и новые задачи, которые работодатели в больших компаниях предлагают решить во время собеседования при приеме на работу.

Уникальный сборник головоломок в этой книге дополнен учебным разделом по методам разработки и анализа алгоритмов, чтобы показать читателю различные подходы к решению алгоритмических задач. Для каждой из 150 головоломок в книге приводятся подсказка и решение, а также комментарии о происхождении головоломки и методах ее решения.

«Алгоритмические головоломки» – книга в своем роде уникальная, в ней есть головоломки разного уровня сложности. Читатели со средней математической подготовкой смогут развить навыки алгоритмического мышления с помощью головоломок элементарного уровня, а более подготовленные получат удовольствие от решения сложных головоломок.

АНАНИЙ ЛЕВИТИН – профессор информатики в Университете Вилланова. Он является автором популярного учебника по разработке и анализу алгоритмов, который был переведен на китайский, греческий, корейский и русский языки. Он также автор статей, посвященных математической теории оптимизации, разработке программного обеспечения, управлению данными, разработке алгоритмов и обучению информатике.

МАРИЯ ЛЕВИТИНА является независимым консультантом. Она проработала несколько лет в ведущих компаниях по разработке программного обеспечения и занималась прикладными бизнес-приложениями для больших корпораций; сейчас она специализируется на интернет-приложениях и беспроводных технологиях. Мария Левитина закончила Московский государственный университет по специальности «Математика».