Общие методы разработки алгоритмов

Ананий Левитин
Villanova University, USA
anany.levitin@villanova.edu

Июнь 2010

Век алгоритма

"As an old saying goes, "The man with a hammer sees every problem as a nail." Our age's hammer is the algorithm."
W. Poundstone, *How Would You Move Mount Fuji? Microsoft's Cult of the Puzzle*, 2003, p. 143.

"The algorithm's coming-of-age as the new language of science promises to be the most disruptive scientific development since quantum mechanics."
B. Chazel, *The Algorithm: Idiom of Modern Science*, *http://www.cs.princeton.edu/~chazelle/pubs* , 2006.

- Распространение компьютеров в науках и жизни выдвинули алгоритмическое решение задач на первый план.

Что такое общие методы разработки алгоритмов?

Общие методы разработки алгоритмов – это наиболее общие подходы к разработке алгоритмов, которые можно применить к самым разнообразным задачам.

Синонимы

общие методы
принципы
подходы
стратегии
парадигмы

разработка алгоритмов
создание алгоритмов
конструирование алгоритмов
дизайн алгоритмов

Важность общих методов разработки алгоритмов

"The study of algorithm design techniques has two important payoffs. First, it leads to an organized way to devise algorithms. Algorithm design techniques give guidance and direction on how to create a new algorithm. Though there are literally thousands of algorithms, there are very few design techniques. Second, the study of these techniques help us to categorize or organize the algorithms we know and in that way to understand them better."

E. Horowitz, in *Encyclopedia of Computer Science*, 3rd ed.

Основные методы разработки алгоритмов

- метод грубой силы (с исчерпывающим перебором в качестве специального случая)
- поиск с возвратом
- уменьшение размера задачи
- декомпозиция
- преобразование
- жадные алгоритмы
- итеративное улучшение
- динамическое программирование
- метод ветвей и границ

Замечание: решение некоторых задач требует применения более одного метода.

Примеры, иллюстрирующие общие методы разработки алгоритмов

- Сортировка и поиск

  Donald Knuth: “Indeed, I believe that *every* important aspect of programming arises somewhere in the context of sorting and searching” [“*The Art of Computer Programming*”, vol. 3, p. v].

- Головоломки

  показывают применимость общих методов разработки алгоритмов в областях за пределами традиционной информатики.

- Другие примеры

Метод грубой силы

Метод грубой силы − прямолинейный подход к решению задачи, обычно основанный только на постановке задачи и определениях используемых в ней понятий.

Примеры из информатики:

- сортировка выбором и пузырьковая сортировка;
- последовательный поиск;
- исчерпывающий перебор.

Алгоритм сортировки, основанный на методе грубой силы

Сортировка выбором. Найдите наименьший элемент в массиве $A[i_0..n-1]$ и поменяйте его местами с первым элементом. Затем сканируйте список, начиная со второго элемента, в поисках наименьшего среди оставшихся $n-1$ элементов и поменяйте его местами со вторым элементом. В общем случае, при i-м проходе по списку ($1 \le i \le n-1$) найдите наименьший элемент в списке $A[i-1..n-1]$

и поменяйте его местами с $A[i]$.

$A[0] \le \ldots \le A[i-1] \mid A[i], \ldots, A[min.], \ldots, A[n-1]$

в окончательных позициях

Пример : 7 3 2 5

2 3 7 5

2 3 7 5

2 3 5 7

Алгоритм поиска элемента с данным значением K в заданном массиве $A[0..n-1]$, основанный на методе грубой силы

Последовательный поиск. Этот алгоритм последовательно сравнивает элементы массива с заданным значением K до тех пор, пока не будет найден элемент с этим значением (успешный поиск) или весь список будет проверен, но требуемый элемент не найден (неудачный поиск).

Исчерпывающий перебор

Исчерпывающий перебор – это применение метода ***грубой силы*** к решению задачи поиска элемента конечного множества с заданным свойством, обычно среди комбинаторных объектов таких, как сочетания, перестановки или подмножества.

Алгоритм:

- Систематически создайте все потенциальные решения задачи, используя стандартные алгоритмы для построения комбинаторных объектов;
- Проверьте все потенциальные решения, отбрасывая те, которые не удовлетворяют условиям задачи. Для задач на оптимизацию сохраняйте лучшее решение, найденное на данный момент.
- Когда поиск закончен, выдайте решение (лучшее решение для задач на оптимизацию).

Примеры исчерпывающего перебора

Сортировка. Составьте $n!$ перестановок данных элементов и просмотрите их пока не будет найдена искомая перестановка.

Задача n ферзей. Расставить n ферзей на шахматной доске размера $n \times n$ так, что никакие два из них не стоят в одном ряду, одной колонке или на одной диагонали.

В разных рядах и разных колонках – $n!$ вариантов

Числовые ребусы. Заменить буквы на цифры так, чтобы выполнялось данное равенство, например:

SEND + MORE = MONEY.

Разумный алгоритм решит эту задачу за несколько секунд, плохой – за много часов.

Замечания о методе грубой силы

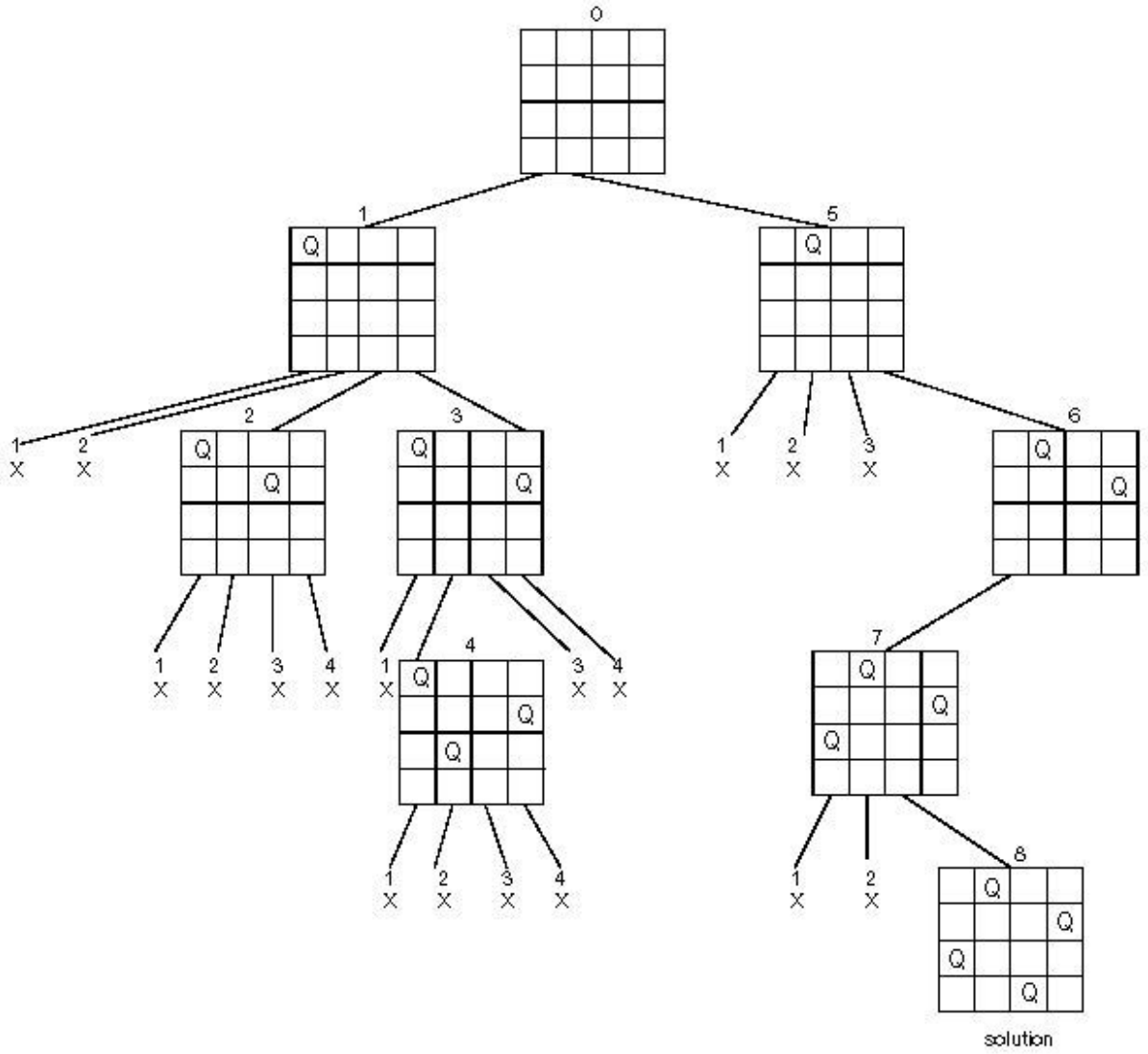
- Преимущества:
  - применим к широкому кругу дискретных задач;
  - простота;
  - трудно или невозможно превзойти для некоторых задач:
    - умножение матриц;
    - NP-сложные задачи (задача коммивояжёра, задача о рюкзаке, и сотни других задач дискретной оптимизации).
- Недостатки:
  - редко приводит к эффективным алгоритмам;
  - менее конструктивен, чем другие общие методы.
- Кроме того:
  - не упоминается в большинстве учебников (незаслуженно);
  - может быть разумной альтернативой в свете низкой стоимости и высокой скорости современных компьютеров.

Поиск с возвратом

Улучшенная версия исчерпывающего перебора, которая пытается избежать ложных потенциальных решений путём отбрасывания частично построенных кандидатов, которые не могут привести к правильному решению задачи.

- Построение ***дерева пространства состояний*:**
  - вершины: частичные решения;
  - рёбра: переходы к расширенным частичным решениям;
  - порядок построения: дерево пространства состояний строится на манер алгоритма «поиск в глубину».
- Отбрасывание ***бесперспективных вершин*:**
  - не рассматривайте ветви дерева, которые идут из ***бесперспективных вершин*** – возвращайтесь к родительской вершине.

Дерево пространства состояний для задачи четырёх ферзей



Замечания по поиску с возвратом

- Позволяет решать трудные задачи большого размера в разумное время, но это не может быть гарантировано.
- Может быть улучшен для задач оптимизации (метод ветвей и границ).
- Основной метод в области искусственного интеллекта.
- Поиск с возвратом не нужен для того, чтобы найти одно решение задачи *n* ферзей!

Метод уменьшения размера задачи

Метод уменьшения размера задачи может быть определен как подход, который использует связь между решением задачи данного размера и решением той же задачи меньшего размера.

- может быть применен либо сверху вниз, то есть рекурсивно, либо снизу вверх, то есть итеративно
- также известен под именем «индуктивный подход»

Три разновидности метода уменьшения размера задачи

- Уменьшение на постоянную величину (обычно 1)
- Уменьшение с постоянным коэффициентом (обычно вдвое)
- Переменное уменьшение размера

Алгоритм сортировки, основанный на методе уменьшения размера задачи

Сортировка вставкой. Чтобы отсортировать массив A[0..n−1], отсортируйте массив A[0..n − 2] рекурсивным методом, а затем вставьте A[n−1] в соответствующее место среди отсортированных элементов A[0..n−2].

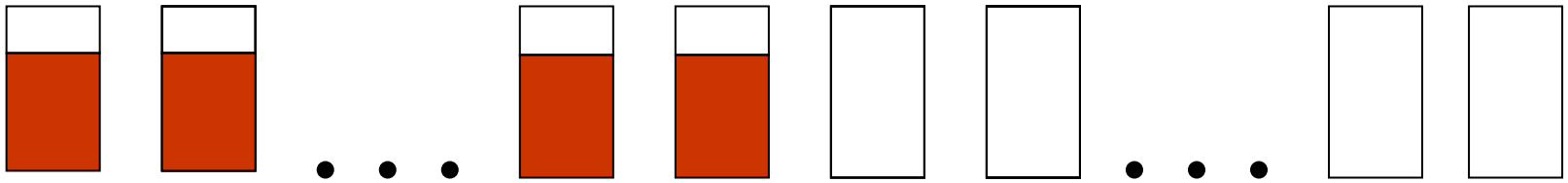
Пример: Отсортировать вставкой 6, 4, 1, 8, 5

6 | 4 1 8 5
4 6 | 1 8 5
1 4 6 | 8 5
1 4 6 8 | 5
1 4 5 6 8

Головоломка, иллюстрирующая метод уменьшения размера задачи на постоянную величину

Чередующиеся стаканы

В ряду $2n$ стаканов первые n стаканов наполнены соком, а следующие n стаканов пусты. Как получить чередующийся ряд стаканов (полный − пустой − полный − пустой и т. д.), затронув минимальное число стаканов?
(Подсказка: Разрешается переливать сок из стакана в стакан.)



Решение. Перелив сок из второго стакана в предпоследний, размер задачи уменьшается на 2. Этот оптимальный алгоритм решает задачу в $\lfloor n/2 \rfloor$ переливаний.

Уменьшение с постоянным коэффициентом

В этом варианте уменьшения размера задачи её размер уменьшается с постоянным коэффициентом (обычно вдвое).

Примеры

- Бинарный поиск и метод половинного деления
- Поиск лёгкой фальшивой монеты
- Умножение à la russe (по-русски).

Замечание: таких задач очень немного!

Бинарный поиск и метод половинного деления

Бинарный поиск. Эффективный алгоритм для поиска величины K в упорядоченном массиве A[0..n−1]

K

vs

A[0] . . . A[m] . . . A[n−1]

Если K = A[m], алгоритм останавливается (успешный поиск).

Если нет, он продолжает поиск тем же методом:

в A[0..m−1], если K < A[m],

и в A[m+1..n−1], если K > A[m].

Делает ≤ 20 сравнений для упорядоченных массивов с 10^6 элементов!

Метод половинного деления для решения уравнений с одним неизвестным $f(x) = 0$ на $[a, b]$, где $f(a)f(b) < 0$, является непрерывным аналогом бинарного поиска. Но работает медленно!

Пример головоломки

Поиск лёгкой фальшивой монеты. Среди n одинаковых на вид монет одна фальшивая. Известно, что она легче, чем настоящие монеты. Как найти фальшивую монету с помощью весов без гирь, сделав минимальное число взвешиваний?

Предупреждение: распределить монеты на две кучки с $\lfloor n/2 \rfloor$ монетами каждая – не самый эффективный способ решения этой задачи.

Решение. Распределить монеты на три кучки с $\lceil n/3 \rceil$, $\lceil n/3 \rceil$ и $n-2\lceil n/3 \rceil$ монетами каждая и взвесить первые две.

Умножение à la russe (*метод русских крестьян*)

Задача: найти произведение двух целых чисел n и m

Для чётных значений n: $n \cdot m = (n/2) \cdot 2m$

Для нечётных значений n:

$n \cdot m = (n-1)/2 \cdot 2m + m$, если $n > 1$ и m, если $n = 1$

Пример. $20 \cdot 26$

n	m	
20	26	
10	52	
5	104	104
2	208	+
1	416	416
		520

Метод сводится к сложению значений m , которые соответствуют нечётным n.

Алгоритмы с переменным уменьшением размера

В методе ***переменного уменьшения размера*** размер уменьшается нерегулярным образом.

Алгоритм Эвклида для наибольшего общего делителя:

$$\text{НОД}(m, n) = \text{НОД}(n, m \bmod n)$$

и т. д., до тех пор пока n не станет равным 0.

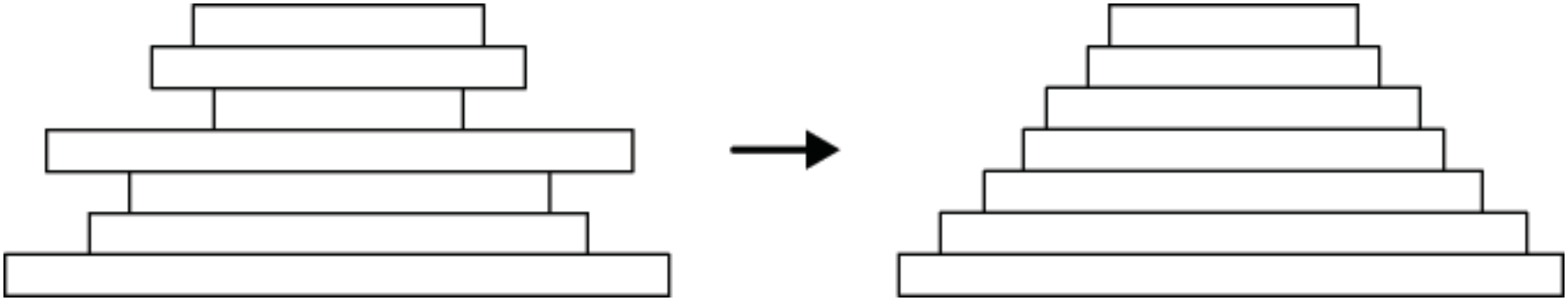
Пример: НОД(80, 44) = НОД(44, 36) = НОД(36, 8) = НОД(8, 4) = НОД(4, 0) = 4.

Метод Ньютона для решения $f(x) = 0$ и другие итеративные методы.

Пример головоломки

Сортировка блинов. *n* блинов разных размеров лежат в стопке.

Разрешается подсунуть лопаточку под один из блинов и перевернуть всю стопку над лопаточкой. Цель задачи – отсортировать блины по их размеру от самого большого внизу до самого маленького наверху. Придумайте алгоритм решения этой задачи и найдите число итераций в худшем случае.



Решение. Повторно «закиньте» наверх наибольший блин, который нарушает порядок, и затем поверните стопку так, чтобы он попал на своё место в отсортированной пирамиде. Алгоритм требует $\leq 2n-3$ переворачиваний для $n > 1$, но он не оптимален.

Метод декомпозиции

Метод декомпозиции (divide-and-conquer) можно определить как метод, который рекурсивно делит задачу на несколько подзадач, до тех пор пока они становятся достаточно простыми для прямого решения. Затем эти решения подзадач объединяются , чтобы получить решение первоначальной задачи.

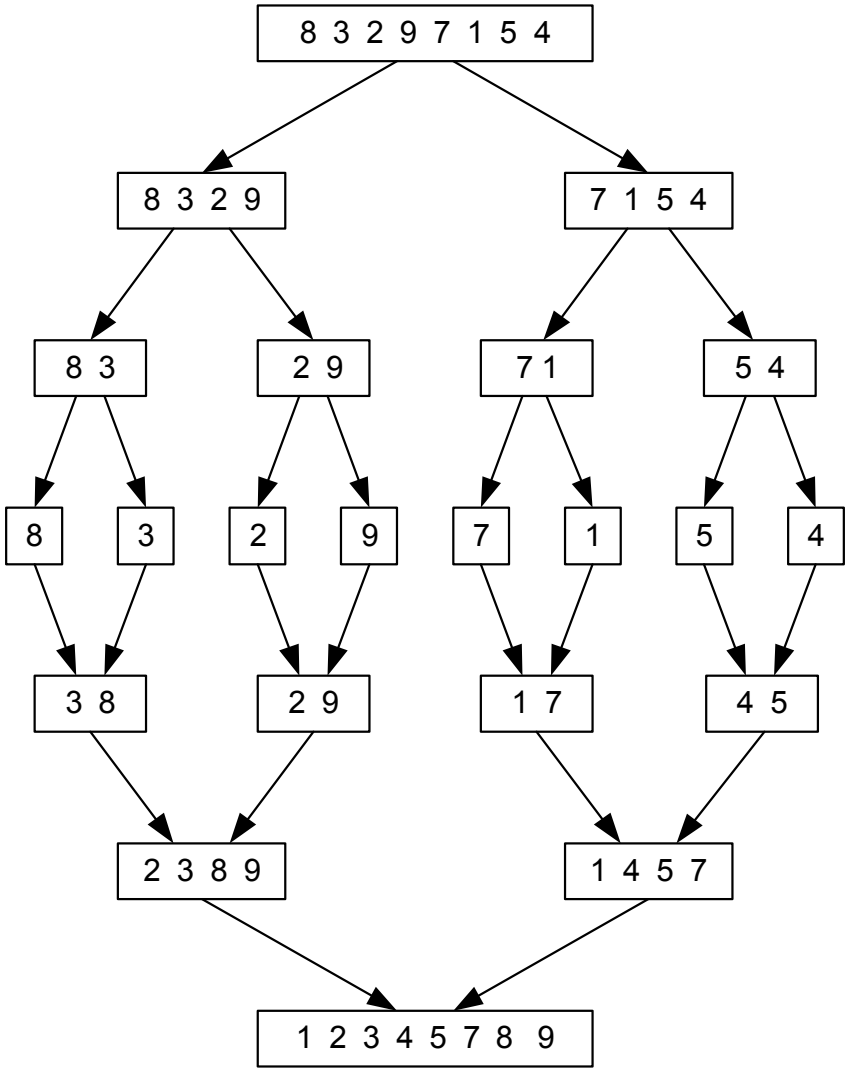
- сортировка слиянием и быстрая сортировка
- метод Карацубы для умножения больших чисел
- умножение матриц методом Штрассена
- обход бинарного дерева и родственные алгоритмы
- декомпозиционные методы для нахождения ближайшей пары точек и построения выпуклой оболочки
- быстрое преобразование Фурье

Сортировка слиянием

- Разделите массив A[0..$n-1$] на две примерно одинаковые части и скопируйте каждую часть в массивы B и C. Отсортируйте B и C тем же методом.

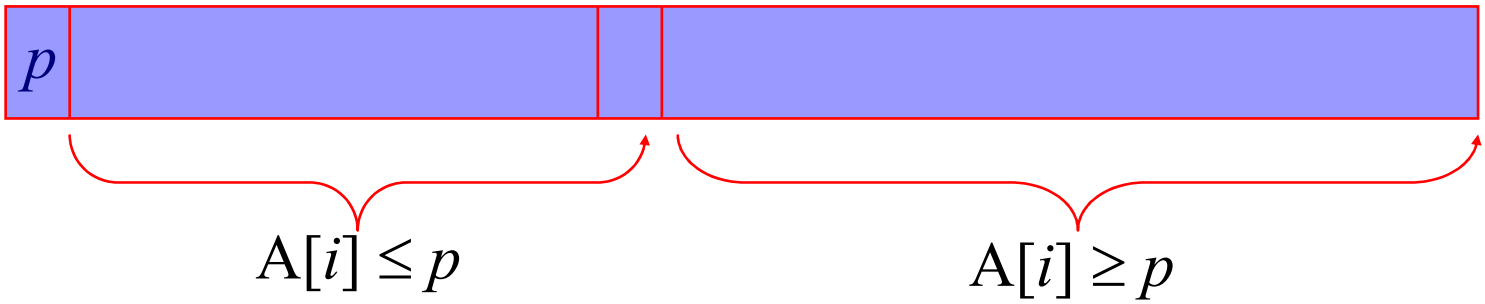
- Слейте отсортированные массивы B и C в массив A, сравнивая первые элементы в оставшихся частях B и C и отсылая в A наименьший из них. Когда в одном из этих массивов не остается элементов, отошлите в A элементы, оставшиеся в другом массиве.

Пример сортировки слиянием



Быстрая сортировка

- Выберите опорный элемент p (для простоты – первый элемент массива).
- Перегруппируйте элементы массива так, чтобы первые s элементов были не больше опорного элемента, а все элементы в остальных $n–s$ позициях были не меньше его.



Поменяйте местами опорный элемент с A[s] – это поставит опорный элемент в его правильную позицию в упорядоченном массиве.

- Отсортируйте тем же методом A[0..s–1] и A[s+1..n–1].

Пример быстрой сортировки

Разбиение массива можно получить, просмотрев его слева направо (индекс *i),* пропуская элементы меньшие, чем опорный, и справа налево (индекс *j*), пропуская элементы бóльшие, чем опорный.

5	3	1	9	8	2	4	7	
			i			*j*		*i* и *j* не пересеклись – переставить элементы
5	3	1	4	8	2	9	7	
				i	*j*			*i* и *j* не пересеклись – переставить элементы
5	3	1	4	2	8	9	7	
				j	*i*			*i* и *j* пересеклись – переставить 5 и 2 (разделение)
2	3	1	4	5	8	9	7	продолжить сортировку тем же методом 2, 3, 1, 4 и 8, 9, 7

Быстрая сортировка – самый эффективный алгоритм для сортировки больших массивов с неизвестными свойствами.

Алгоритм Карацубы для умножения целых чисел

Если $A = A_1A_2$ и $B = B_1B_2$, где A и B − два целых числа с n цифрами, разбитые на две половины A_1, A_2 и B_1, B_2 с $n/2$ цифрами, то

$$A \cdot B = A_1 \cdot B_1 \cdot 10^n + (A_1 \cdot B_2 + A_2 \cdot B_1) \cdot 10^{n/2} + A_2 \cdot B_2.$$

Использование этого равенства по-прежнему требует n^2 умножений двух цифр, что следует из решения рекурсивного уравнения для числа умножений

$$M(n) = 4M(n/2), \quad M(1) = 1.$$

Но

$$(A_1 \cdot B_2 + A_2 \cdot B_1) = (A_1 + A_2) \cdot (B_1 + B_2) - A_1 \cdot B_1 - A_2 \cdot B_2$$

уменьшает число умножений до

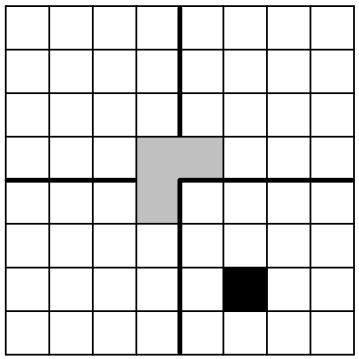
$$M(n) = 3M(n/2), \quad M(1) = 1,$$

то есть $M(n) = n^{\log_2 3} \approx n^{1,585}$.

Пример головоломки

Задача о тромино. Замостить доску размером $2^n \times 2^n$ с одной вырезанной клеткой прямоугольными тромино. Прямоугольное тромино − это фигура, образованная склеиванием трёх квадратов в форме буквы Г, но оно может быть ориентировано любым способом.

Решение. Если $n = 1$, задача имеет тривиальное решение. Если $n > 1$, поместите одно тромино в центре доски так, чтобы оно разбило доску на четыре части размером $2^{n-1} \times 2^{n-1}$, у каждой из которых «вырезан» один квадрат и, следовательно, каждая может быть покрыта этим же алгоритмом (рекурсивно).



Замечания о методе декомпозиции

- Несколько весьма важных алгоритмов попадают в эту категорию.
- Метод декомпозиции не следует путать с методом уменьшения размера задачи.
- Алгоритмы, основанные на методе декомпозиции, особенно удобны для параллельных вычислений.

Метод преобразования

Преобразовать

- в более простой случай той же задачи
- в другую форму той же задачи
- в другую задачу с известным решением

Примеры преобразования в более простой случай

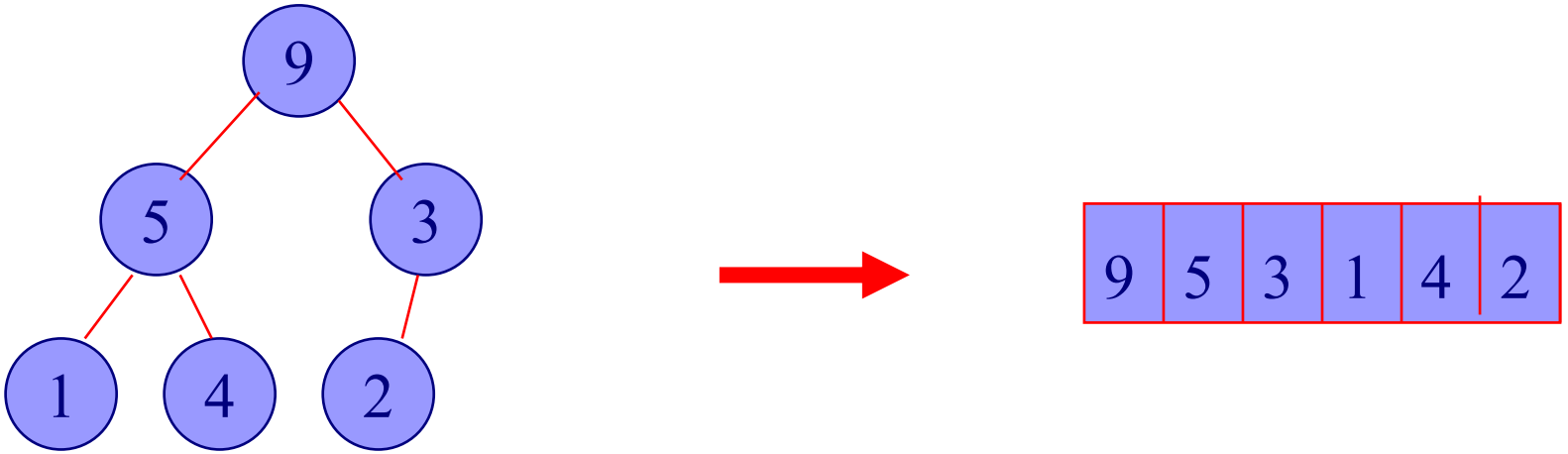
- Сортировка исходных данных (например, задача о наличии равных элементов в $A[0..n-1]$)
- Топологическая сортировка ориентированного ациклического графа
- Метод исключения Гаусса
- Преобразование общей задачи линейного программирования к задаче в канонической форме

Примеры изменения формы задачи

- Пирамидальная сортировка
- Деревья поиска
- Использование бинарной и других систем счисления
- Метод Горнера вычисления значения многочлена
- Быстрое преобразование Фурье

Пирамидальная сортировка

Пирамидой называется полное бинарное дерево с ключами (числами или строками) в его узлах при условии, что величина ключа в каждом узле $\geq$ величине ключей в его детях.



Часть 1. Постройте пирамиду для данного списка с n ключами.

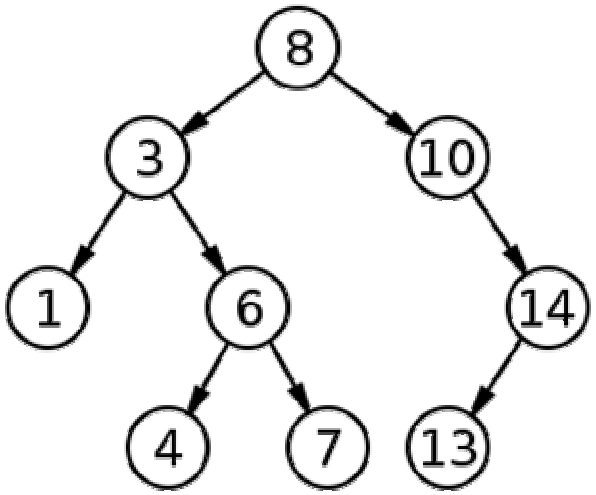
Часть 2. Повторите следующую операцию, удаляющую наибольший элемент $n-1$ раз:

- поменять местами первый (корневой) и последний ключи;
- уменьшить размер пирамиды на 1;
- если нужно, обменять новый корневой ключ с бóльшим ключом в его детях, чтобы восстановить требования пирамиды.

Бинарное дерево поиска (БДП)

Представление списка ключей бинарным деревом поиска:

левое поддерево каждого узла содержит ключи, которые меньше, чем ключ в этом узле; правое поддерево содержит ключи, которые больше или равны ключу в узле.



БДП – одна из самых важных идей в информатике, с многочисленными обобщениями и применениями, включая поиск и сортировку.

Примеры преобразования в другую задачу

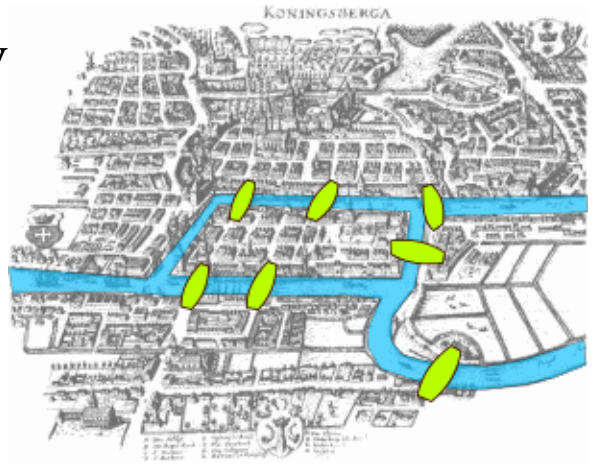
- Вычисление наименьшего общего кратного:
$$\text{НОК}(m, n) = mn/\text{НОД}(m, n).$$

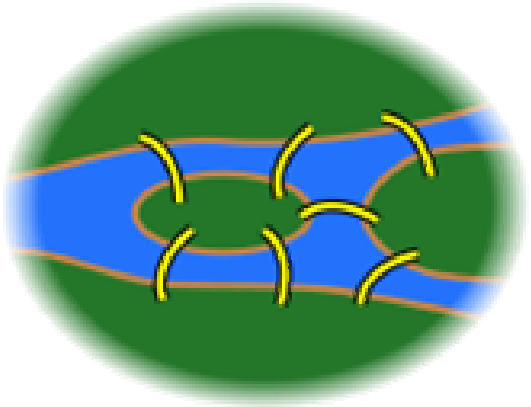
- Сведение к задачам о графах
(кёнигсбергские мосты, графы пространства состояний).

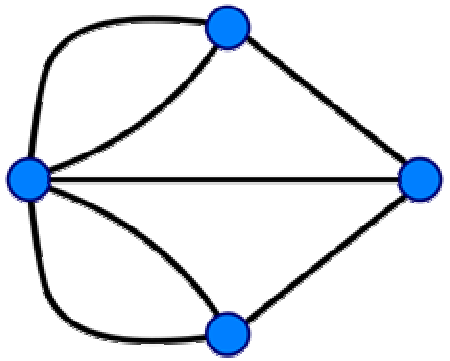
- Математическое программирование и
математическое моделирование.

Задача о кёнигсбергских мостах (Эйлер, 1736)

Можно ли было пройти по каждому из 7 мостов Кёнигсберга один раз в течение той же прогулки?



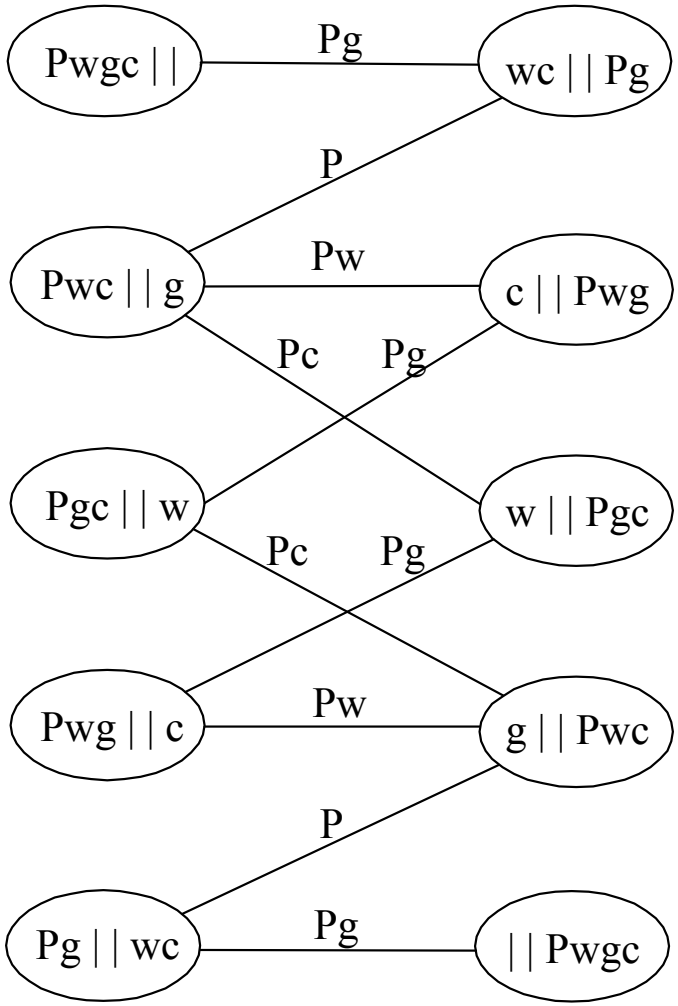




Рисунки взяты из статьи в Википедии

Пример графа пространства состояний

Граф пространства состояний для головоломки о волке, козе и капусте



Общие замечания о методе преобразования

- Очень важная идея в математике и информатике, теоретически и практически (например, понятие NP-сложных задач).
- Этот метод не так конструктивен, как, скажем, метод декомпозиции, но его три разновидности – упрощение, изменение формы и сведéние к другой задаче – могут быть полезны.
- Разница между изменением формы и сведéнием к другой задаче не всегда очевидна.

Жадный метод

Жадный метод – это подход к решению задач оптимизации, в которых на каждом шагу делается выбор, который

- удовлетворяет всем ограничениям задачи;
- локально оптимален;
- необратим.

Жадный подход приводит к корректному решению некоторых задач оптимизации, но не работает для других.

Примеры жадных алгоритмов

- задача о размене
- алгоритмы Прима и Крускала для нахождения минимального остовного дерева (МОД)
- алгоритм Дейкстры
- коды Хаффмана

- приближённые алгоритмы для задач комбинаторной оптимизации (например, правило ближайшего соседа в задаче коммивояжёра)
- градиентные методы спуска для минимизации функций

Задача о размене

Выплатить сумму n наименьшим количеством монет номиналом $d_1 > \ldots > d_m = 1$

Жадное решение: Повторно используйте максимальное число раз монету наибольшего возможного номинала

Пример: $n = 83$ коп. и

$d_1 = 50$ коп., $d_2 = 10$ коп., $d_3 = 5$ коп., $d_4 = 1$ коп.

Жадное решение: $83 = 1 \times 50 + 3 \times 10 + 0 \times 5 + 3 \times 1$

Жадное решение

- оптимально для любой суммы n и «обычных» номиналов
- может не быть оптимальным для произвольных номиналов (например, $n = 10$, $d_1 = 7$, $d_2 = 5$, $d_3 = 1$)

Задача о минимальном остовном дереве (МОД)

Для данного связного графа с весами на его рёбрах найти МОД – его связный ациклический подграф (то есть дерево), который содержит все вершины графа и имеет наименьшую сумму весов на своих рёбрах.

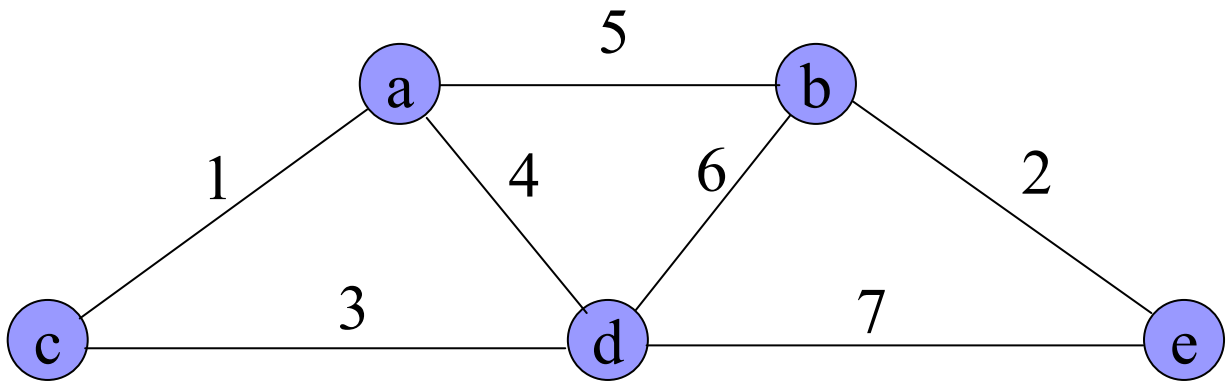
Удивительно, что задача может быть решена двумя разными жадными алгоритмами:

- алгоритмом Прима (вершины)
- алгоритм Крускала (рёбра)

Алгоритм Прима

- Начните с произвольной вершины.
- Постройте последовательность расширяющихся поддеревьев $T_1, T_2, \ldots, T_n$, добавляя к T_i ближайшую новую вершину, т.е. вершину, соединённую с вершиной из T_i ребром минимального веса.
- Алгоритм останавливается, когда все n вершин включены в T_n = МОД.

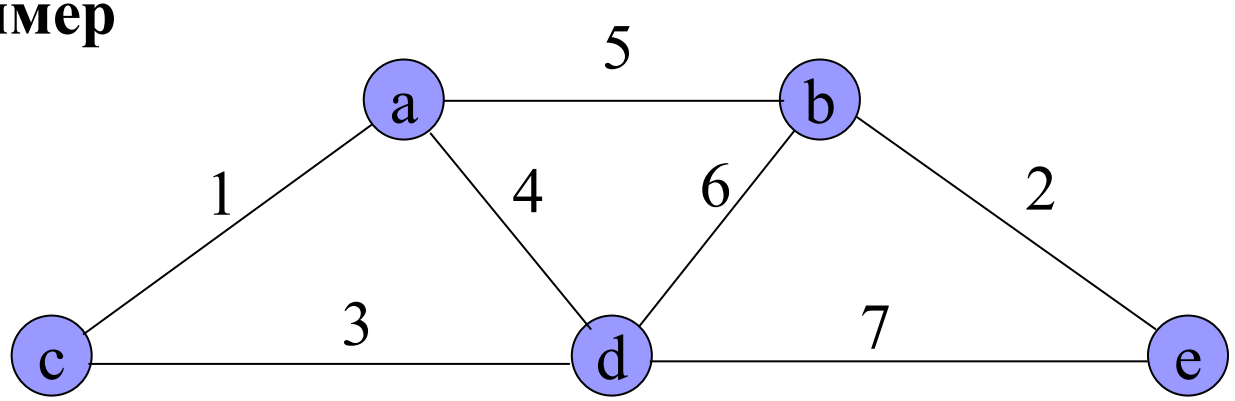
Пример



Алгоритм Крускала

- Отсортируйте рёбра в возрастающем порядке весов.
- Начните с включения в G_1 первого ребра в списке.
- Постройте последовательность расширяющихся подграфов $G_1, G_2, \ldots, G_{n-1}$, добавляя к G_i следующее ребро в списке при условии, что такое включение не создаст цикла. Если цикл возникает, просто перейдите к следующему ребру в списке.
- Алгоритм останавливается после включения $n-1$ ребра в G_{n-1} = МОД.

Пример



Пример головоломки

Задача о гирях. Найти оптимальный набор n гирь весом $\{w_1, w_2, ..., w_n\}$, позволяющий взвесить на рычажных весах любой целочисленный вес l от 1 до наибольшего значения W в случае, если

- гири могут находится только на одной чашке весов;
- гири могут находится на обеих чашках весов.

Решение. Начиная с $w_1 = 1$ и всегда выбирая такой вес w_i, чтобы можно было взвесить максимальный интервал последовательных весов с $\{w_1, w_2, ..., w_i\}$, придём к максимальному интервалу в обоих случаях:

а) $w_1 = 1, w_2 = 2, \ldots, w_n = 2^{n-1}$ в интервале от 1 до $W = 2^n - 1$;

б) $w_1 = 1, w_2 = 3, \ldots, w_n = 3^{n-1}$ в интервале от 1 до $W = (3^n - 1)/2$.

Замечания по поводу жадного подхода

- построение жадного алгоритма обычно гораздо легче, чем доказательство его корректности
- даже если этот подход не работает, он может быть полезен для получения первого приближения
- жадный подход – это единственный общий метод разработки алгоритмов, который был формально изучен (используя понятие «матроида»)

Метод последовательного улучшения

“Unlike greedy algorithms, *iterative methods* start with any feasible solution and proceed to improve upon the solution by repeated applications of a simple step. The step typically involves a small, localized change which improves the value of the objective function.” (Moret and Shapiro, “Algorithms from P to NP”)

Примеры

- симплекс-метод
- алгоритм Форда − Фалкерсона о максимальном потоке в сетях
- алгоритмы локального поиска, например, 2-опт и 3-опт для задачи коммивояжёра
- головоломки, основанные на идее полуинварианта − некоторой характеристики с монотонной последовательностью значений, последнее из которых является решением задачи.

Пример головоломки

Положительные изменения. Дана таблица размера $m \times n$, содержащая действительные числа. Существует ли алгоритм, делающий все суммы по строкам и по столбцам неотрицательными, если единственная разрешённая операция − изменение знаков всех чисел какой-нибудь строки или какого-нибудь столбца?

Решение. Последовательно найдите строку или столбец с отрицательной суммой и измените там знаки всех чисел. Это увеличит общую сумму всех элементов таблицы. Эта сумма является *полуинвариантом* задачи.

Динамическое программирование

• В теории оптимизации этот подход интерпретируется как метод решения оптимизационных задач, которые удовлетворяют принципу оптимальности. (Ричард Беллман, 1950-е годы)

• В информатике ДП интерпретируется как подход к решению задач с пересекающимися подзадачами, не обязательно оптимизационного характера:

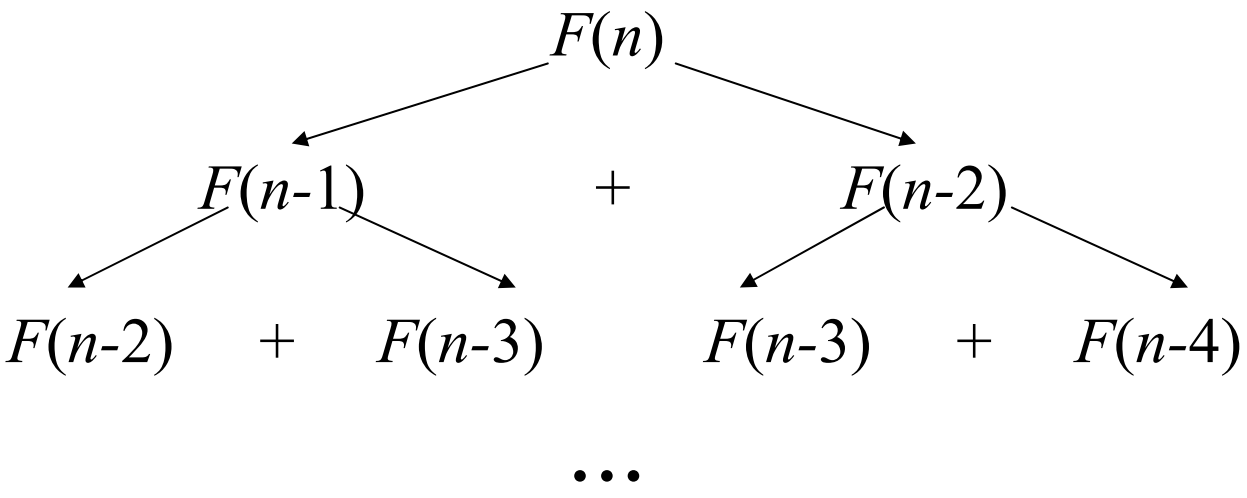
- решите все меньшие подзадачи, но один раз;
- занесите решения в таблицу;
- возьмите решение для нужного случая из этой таблицы.

Простой пример

Вычисление n-го числа Фибоначчи:

$$F(n) = F(n-1) + F(n-2) \text{ for } n > 1$$
$$F(0) = 0$$
$$F(1) = 1$$

Рекурсивное вычисление n-го числа Фибоначчи очень неэффективно:



ДП вычисление *n*-го числа Фибоначчи

Вычисление *n*-го числа Фибоначчи методом ДП:

$F(0) = 0$
$F(1) = 1$
$F(2) = 1+0 = 1$
•
•

$F(n) = F(n-1) + F(n-2)$

0	**1**	**1**	**. . .**	***F*(*n*-2)**	***F*(*n*-1)**	***F*(*n*)**

Стандартные ДП алгоритмы в информатике

- Вычисление биномиальных коэффициентов (треугольник Паскаля)
- Оптимальное умножение нескольких матриц
- Построение оптимального бинарного дерева поиска
- Алгоритм Воршалла для транзитивного замыкания
  Алгоритм Флойда для поиска кратчайших путей между всеми парами вершин взвешенного графа
- Некоторые трудные задачи дискретной оптимизации (например, задача о рюкзаке)

Задача о рюкзаке

Даны n предметов с

целочисленными весами: $w_1 \quad w_2 \ldots \quad w_n$

стоимостями: $v_1 \quad v_2 \quad \ldots \quad v_n$

и рюкзак вместимостью W; требуется найти наиболее ценное подмножество предметов, помещающееся в рюкзаке.

Рассмотрим случай первых i предметов и вместимость j ($j \le W$). Пусть $V[i, j]$ будет максимальная ценность в этом случае. Тогда

$$V[i,j] = \begin{cases} \max\{V[i-1, j],\ v_i + V[i-1, j-w_i]\}, \text{ если } j-w_i \ge 0 \\ \\ V[i-1, j], \text{ если } j-w_i < 0 \end{cases}$$

Начальные условия: $V[0, j] = 0$ и $V[i, 0] = 0$

Задача о рюкзаке (продолжение)

Пример. Вместимость рюкзака $W = 5$

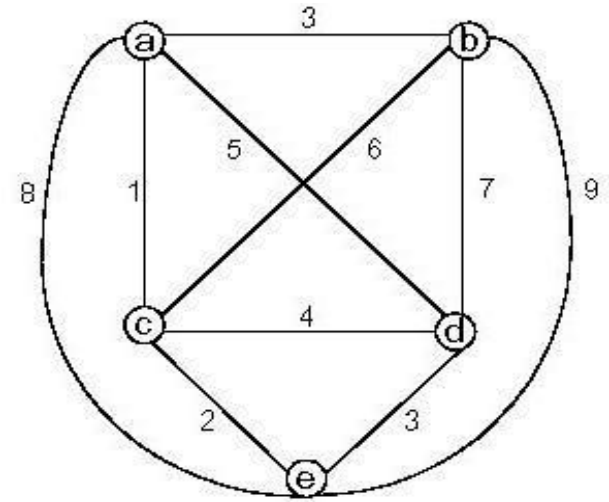
предмет	вес	стоимость
1	2	12
2	1	10
3	3	20
4	2	15

		вместимость j					
		0	1	2	3	4	5
	0	0	0	0	0	0	0
$w_1 = 2, v_1 = 12$	1	0	0	12	12	12	12
$w_2 = 1, v_2 = 10$	2	0	10	12	22	22	22
$w_3 = 3, v_3 = 20$	3	0	10	12	22	30	32
$w_4 = 2, v_4 = 15$	4	0	10	15	25	30	**37**

Метод ветвей и границ

- Улучшение поиска с возвратом для оптимизационных задач
- Для каждого узла (частично построенного решения) в дереве пространства состояний метод вычисляет оценку величины оптимизируемой функции во всех потомках этого узла (расширений этого частичного решения)
- Оценка используется для
  - отбрасывания бесперспективных вершин
  - определения порядка построения дерева (узел с лучшей оценкой обычно исследуется раньше остальных)

Применение к задаче коммивояжёра



lb = сумма длин двух самых коротких рёбер для всех вершин, делённая на 2

Ещё раз: общие методы разработки алгоритмов

метод грубой силы
  исчерпывающий перебор

поиск с возвратом

уменьшение размера задачи
  на постоянную величину
  с постоянным коэффициентом
  переменное уменьшение размера

декомпозиция

преобразование
  к другому случаю
  в другую форму
  в другую задачу

жадный подход

метод ветвей и границ

итеративное улучшение

динамическое программир.

Заключительные замечания об общих методах разработки алгоритмов

- Применение общего метода к той же задаче может дать разные алгоритмы (например, метод сливания и быстрая сортировка, алгоритмы Прима и Крускала для построения минимального остовного дерева)

- Некоторые алгоритмы используют идеи нескольких методов (например, быстрое преобразование Фурье)

- Классификация некоторых алгоритмов может быть неоднозначна (например, сортировка перебором)