Основы информатики

Виды алгоритмов. Алгоритмы сортировки

Заикин Олег Сергеевич

zaikin.icc@gmail.com

Алгоритм сортировки

Алгоритм сортировки — алгоритм для упорядочения элементов в списке (или массиве).

Между элементами списка должны существовать отношения, допускающие полное упорядочивание (отношение порядка).

Например, отношения «меньше либо равно» и «больше либо равно» на множестве вещественных чисел являются отношениями порядка.

Классификация алгоритмов сортировки

- **Устойчивость** — устойчивая сортировка не меняет взаимного расположения равных элементов.
- **Естественность поведения** — эффективность метода при обработке уже упорядоченных, или частично упорядоченных данных.
- **Частичная упорядоченность** **-** насколько упорядочена часть массива при его прерывании
- **Использование операции сравнения.** Алгоритмы, использующие для сортировки сравнение элементов между собой, называются основанными на сравнениях. Для специальных случаев (типов данных) существуют более эффективные алгоритмы.

Методы разработки алгоритмов

Основные методы:

1. Метод грубой силы
2. Метод декомпозиции
3. Метод уменьшения размера задачи

Метод грубой силы

Прямой подход к решению задачи, обычно основан на формулировке задачи и используемых ею концепциях.

Тривиальный пример: $a^n = a*a*....*a$

Последовательный поиск - последовательный перебор всех значений до момента нахождения требуемого значения.

Реализуется при помощи циклов **while-do** и **repeat-until**

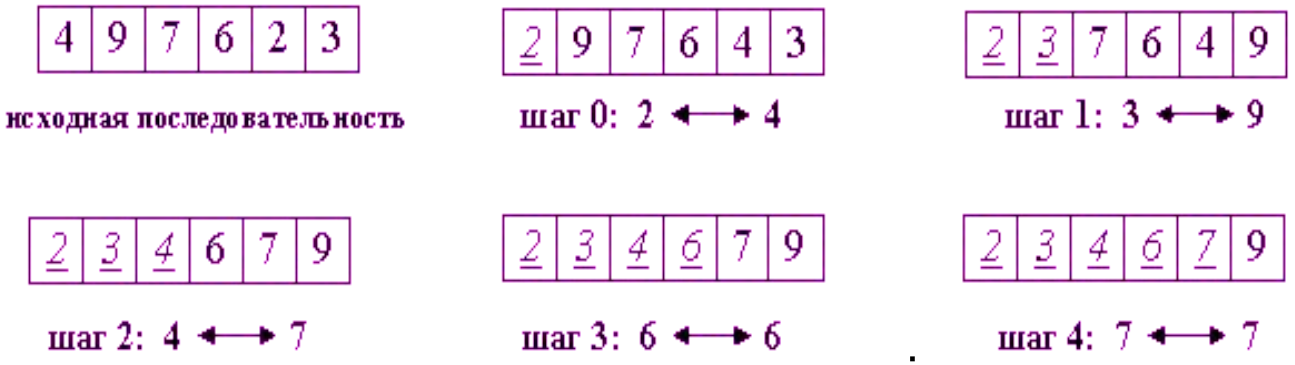
Метод грубой силы
Сортировка выбором

Идея алгоритма состоит в создании отсортированной последовательности путем присоединения к ней одного элемента за другим в правильном порядке.

Последовательность строится, начиная с левого конца массива. Алгоритм состоит из n-1 последовательных шагов, начиная от 0 и заканчивая n-2.

Метод грубой силы
Сортировка выбором

На i-м шаге выбираем наименьший из элементов a[i] … a[n-1] и меняем его местами с a[i]. Последовательность шагов при n=5 изображена на рисунке:



Вне зависимости от номера текущего шага i, последовательность a[0]…a[i] (выделена курсивом) является упорядоченной. Таким образом, на (n-1)-м шаге вся последовательность, кроме a[n] оказывается отсортированной, а a[n] стоит на последнем месте по праву: все меньшие элементы уже ушли влево.

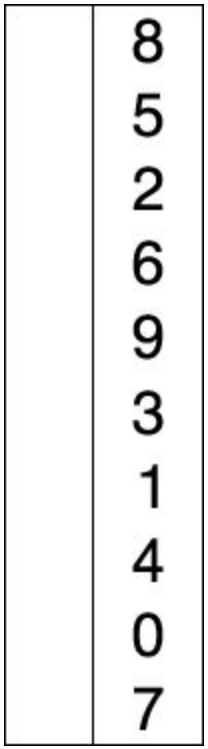
Метод грубой силы
Сортировка выбором

```
for (int i=0; i < n-1; i++)
{
  min = i;
  for (int j=i+1; j < n; j++)
   If a[j] < a[min] then
      min = j;
  t = a[i];
  a[i] = a[min];
  a[min] = t;
}
```

Шаги алгоритма:

1. находим минимальное значение в текущем списке
2. производим обмен этого значения со значением на текущей позиции
3. сортируем хвост списка, исключив из рассмотрения уже отсортированные элементы

Метод грубой силы
Сортировка выбором



Метод грубой силы
Пузырьковая сортировка

Алгоритм состоит в повторяющихся проходах по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов.

Проходы по массиву повторяются до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован.

При проходе алгоритма, элемент, стоящий не на своём месте, «всплывает» до нужной позиции как пузырёк в воде, отсюда и название алгоритма.

Метод грубой силы
Пузырьковая сортировка. Пример

Массив с числами «5 1 4 2 8». Первый проход:

(**5 1** 4 2 8) (**1 5** 4 2 8), Здесь алгоритм сравнивает два первых элемента и меняет их местами.

(1 **5 4** 2 8) (1 **4 5** 2 8), Меняет местами, так как 5 > 4

(1 4 **5 2** 8) (1 4 **2 5** 8), Меняет местами, так как 5 > 2

(1 4 2 **5 8**) (1 4 2 **5 8**), 8 > 5, не меняем

Второй проход:

(**1 4** 2 5 8) (**1 4** 2 5 8) 4 > 1, не меняем

(1 **4 2** 5 8) (1 **2 4** 5 8), Меняет местами, так как 4 > 2

(1 2 **4 5** 8) (1 2 **4 5** 8) 5 > 4, не меняем

(1 2 4 **5 8**) (1 2 4 **5 8**) 8 > 5, не меняем

Метод грубой силы
Пузырьковая сортировка

Вход: массив A, состоящий из элементов A[0], A[1], ..., A[n-1]

```
t := истина
цикл пока t:
    t := ложь
    цикл для j = 0, 1, ..., n − 2:
        если A[j] > A[j+1], то:
            поменять местами элементы A[j] и A[j+1]
            t := истина
```

Методы разработки алгоритмов

Основные методы:

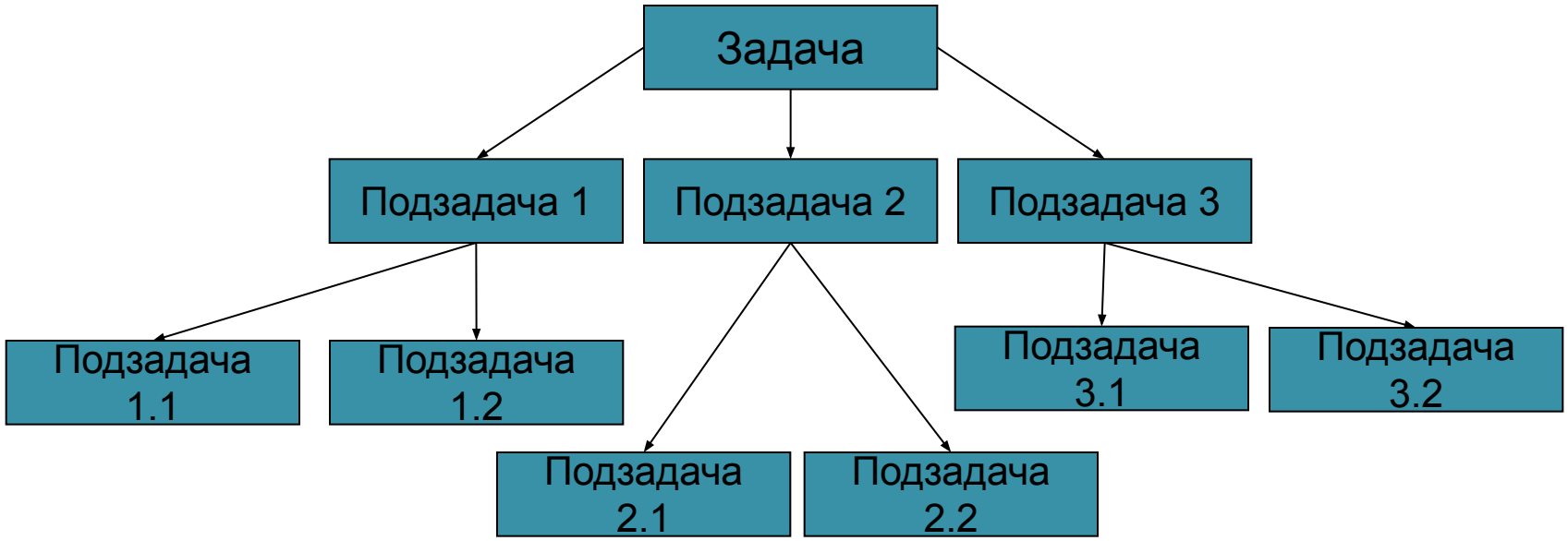
1. Метод грубой силы
2. **Метод декомпозиции**
3. Метод уменьшения размера задачи

Метод декомпозиции

Схема алгоритмов, основанных на методе декомпозиции:

1. Экземпляр задачи разбивается на несколько “более простых” подзадач.
2. Решаются подзадачи
3. При необходимости решение исходной задачи находится путем комбинации решений подзадач

Пример иерархии задач



Разложение задачи в последовательность разнородных подзадач

В методе выделяют относительно небольшое число разнородных подзадач.

Разложение задачи в последовательность однородных подзадач

В данном подходе алгоритм решения разбивается на части – отдельные компоненты вектора.

Например, задача P сводится к n экземплярам более простой задачи R и к простой задаче Q, объединяющей n решений.

Разложение задачи в последовательность однородных подзадач

Например – скалярное произведение 2 векторов.

$$\bar{a} = (a_1, a_2, \ldots, a_n)$$

$$\bar{b} = (b_1, b_2, \ldots, b_n)$$

$$\langle \bar{a}, \bar{b} \rangle = a_1 \cdot b_1 + a_2 \cdot b_2 + \ldots + a_n \cdot b_n$$

Укажите задачи P, R, Q

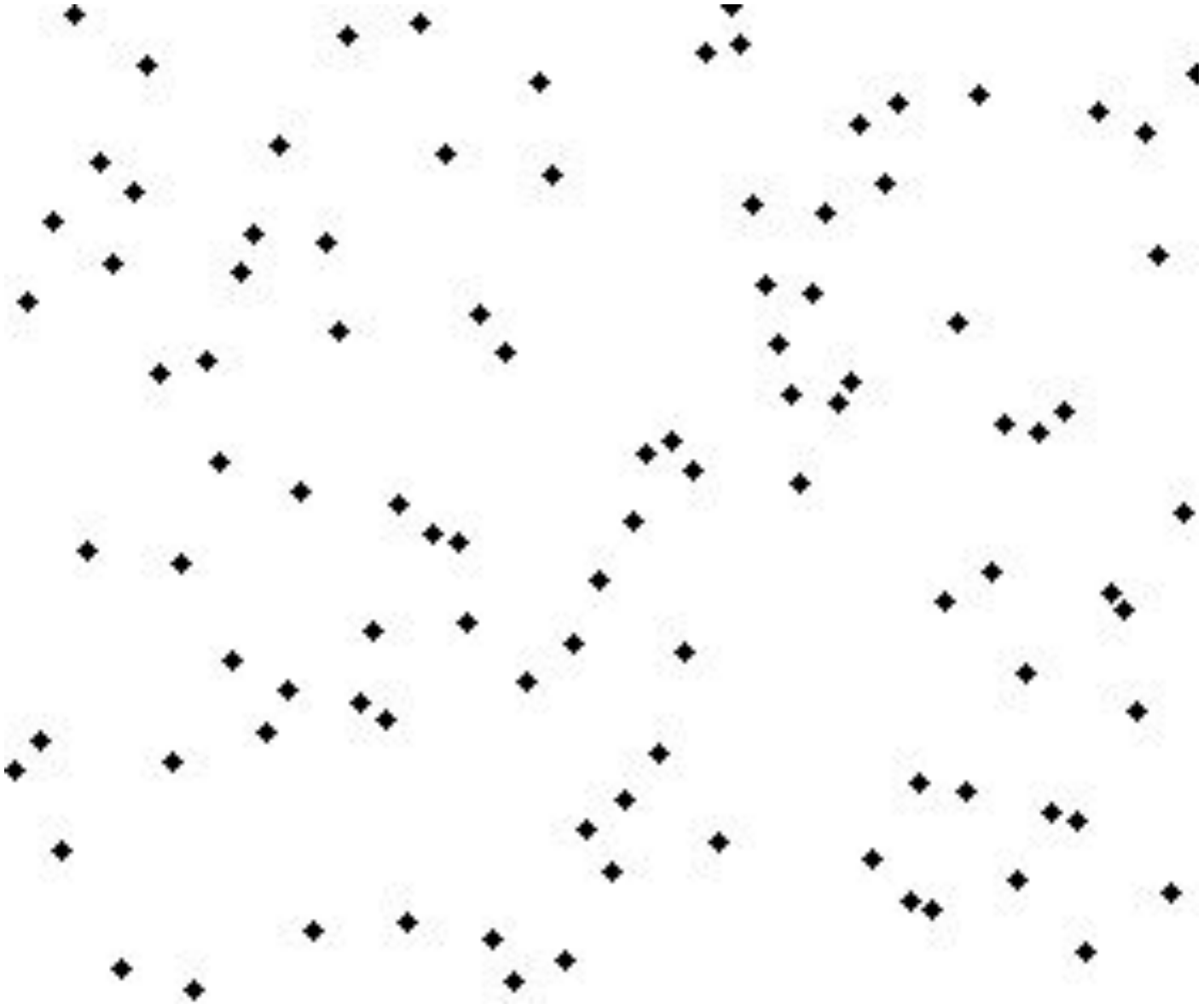
Сортировка слиянием

Этапы решения задачи сортировки:

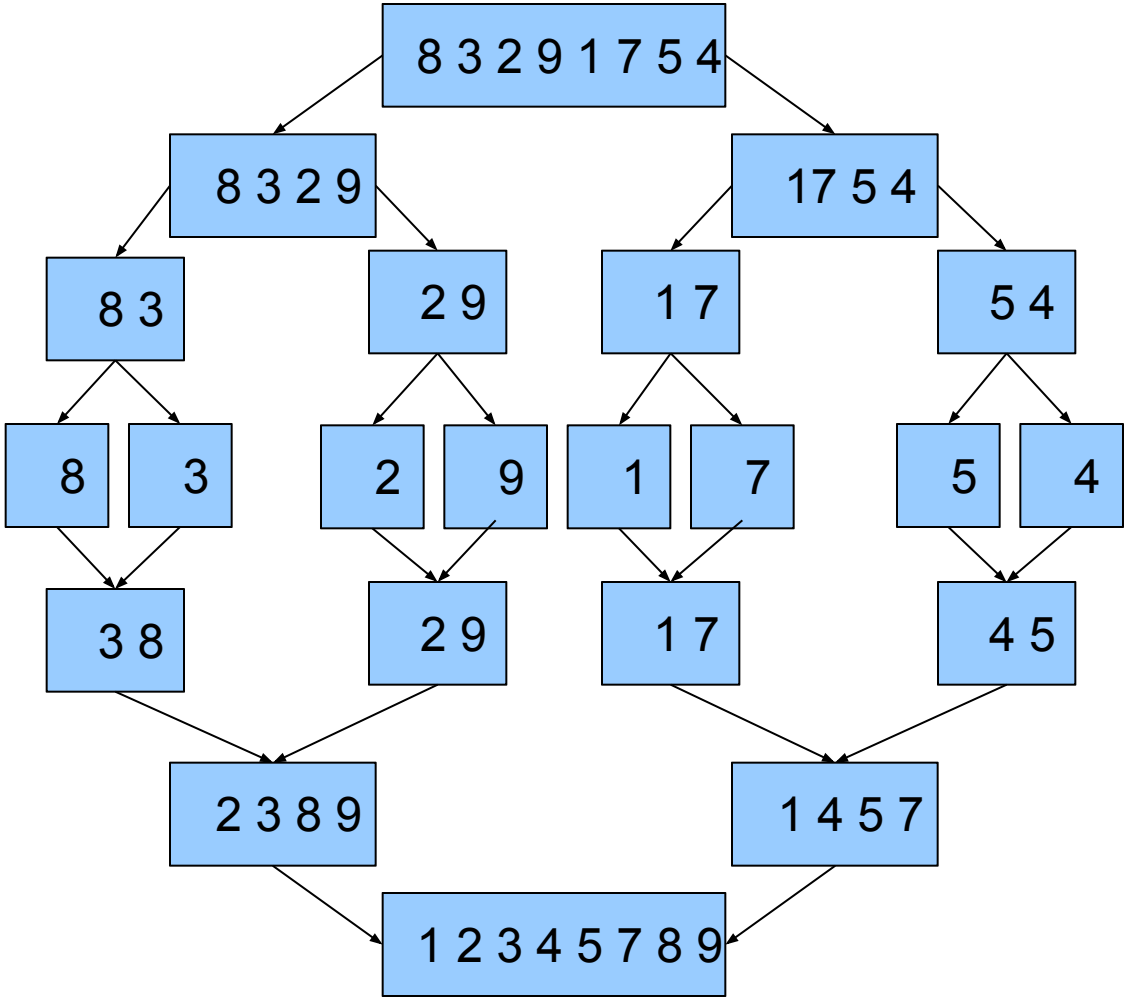
1) Сортируемый массив разбивается на две части меньшего размера;
2) Каждая из получившихся частей сортируется отдельно;
3) Два упорядоченных массива меньшего размера соединяются в один.

Рекурсивное разбиение задачи на меньшие происходит до тех пор, пока размер массива не достигнет единицы (любой массив длины 1 можно считать упорядоченным).

Сортировка слиянием применительно к случайным точкам

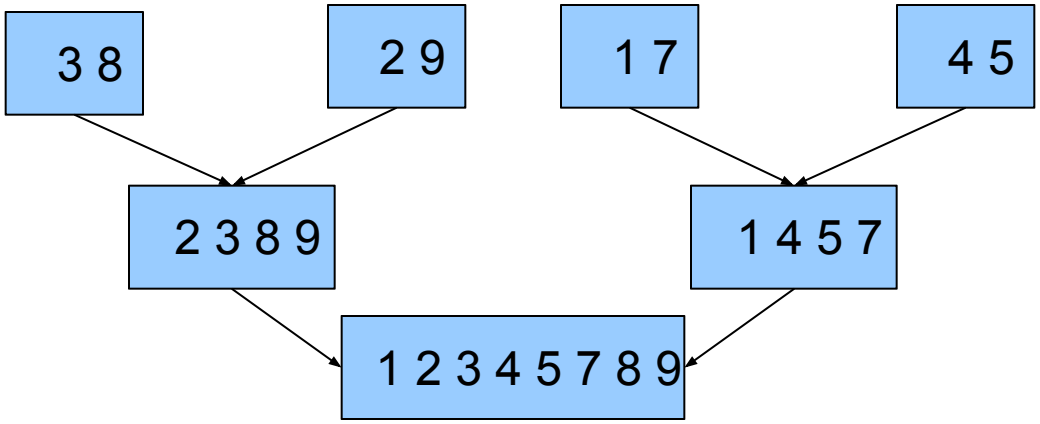


Пример работы алгоритма сортировки слиянием



Пример работы алгоритма сортировки слиянием

Как именно два упорядоченных массива меньшего размера соединяются в один?



Быстрая сортировка

- Разработана английским информатиком Чарльзом Хоаром в 1960 году в Московском университете, где он обучался компьютерному переводу и занимался разработкой русско-английского разговорника.

В то время словарь хранился на магнитной ленте, и если упорядочить все слова в тексте, их переводы можно получить за один прогон ленты.

Наиболее популярный алгоритм сортировки.

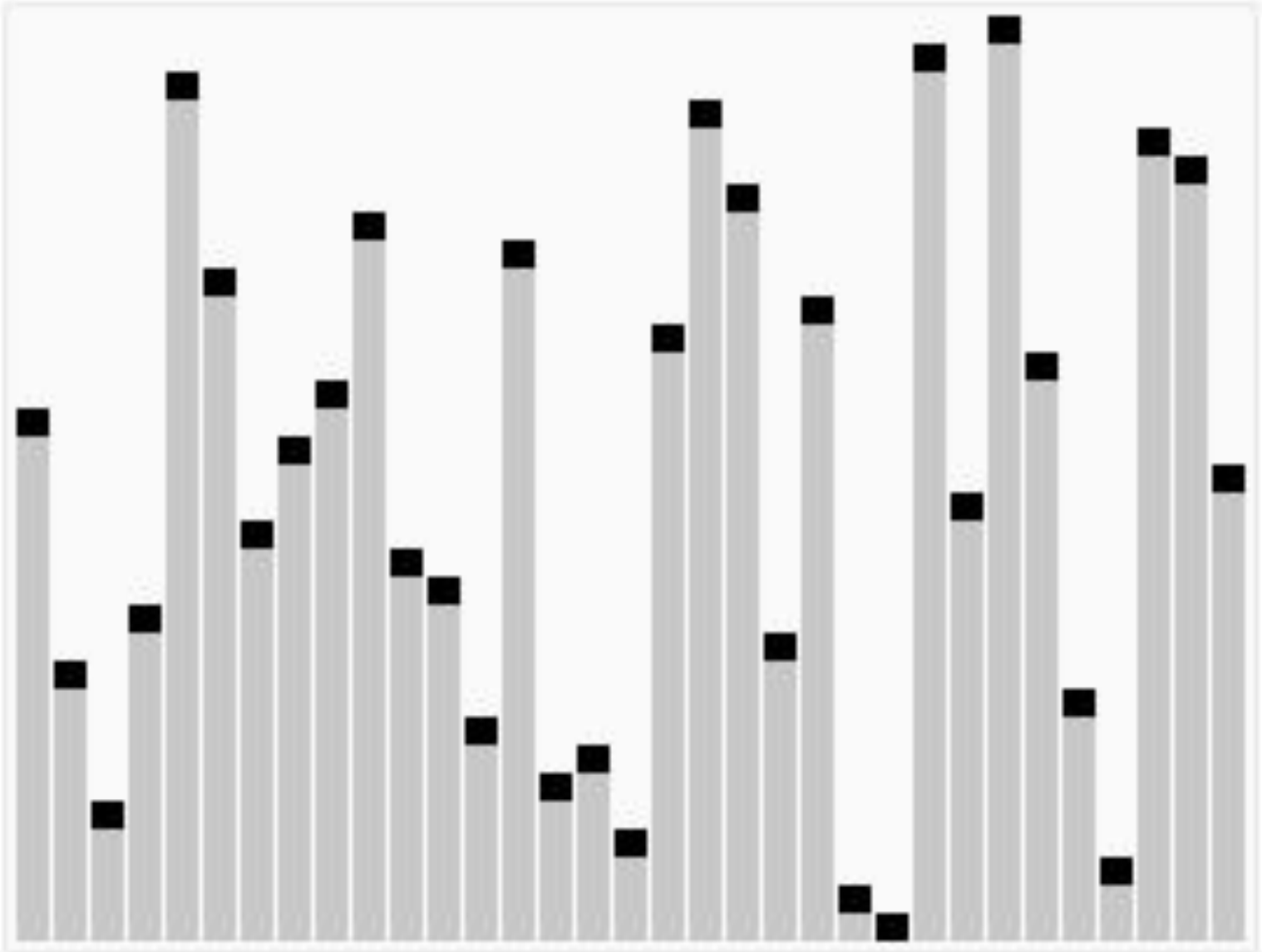
В стандартной библиотеке C++ реализация называется qsort.

Быстрая сортировка

Алгоритм:

1. выбрать элемент, называемый опорным. Можно выбирать различными способами. Например, средний.
2. сравнить все остальные элементы с опорным, на основании сравнения разбить множество на три — «меньшие опорного», «равные» и «большие», расположить их в порядке меньшие-равные-большие.
3. повторить рекурсивно для «меньших» и «больших». Базой рекурсии являются наборы, состоящие из одного или двух элементов. Первый возвращается в исходном виде, во втором, при необходимости, сортировка сводится к перестановке двух элементов.

Быстрая сортировка. Пример



Быстрая сортировка. Пример

6 5 3 1 8 7 2 4

Быстрая сортировка

- Поскольку на каждом следующем уровне рекурсии длина обрабатываемого отрезка массива уменьшается, по меньшей мере, на единицу, терминальная ветвь рекурсии будет достигнута обязательно и обработка гарантированно завершится.

Быстрая сортировка

На практике обычно разделяют сортируемое множество не на три, а на две части: например, «меньшие опорного» и «равные и большие».

Такой подход в общем случае оказывается эффективнее, так как для осуществления такого разделения достаточно одного прохода по сортируемому множеству и однократного обмена лишь некоторых выбранных элементов.

Быстрая сортировка

- Быстрая сортировка является существенно улучшенным вариантом пузырьковой сортировки, эффективность которого низка.

Принципиальное отличие состоит в том, что после каждого прохода элементы делятся на две независимые группы.

Улучшение самого неэффективного прямого метода сортировки дало в результате эффективный улучшенный метод.

Быстрая сортировка. Оценка эффективности

Лучший случай. В каждой итерации массив разделяется на два равных по величине подмассива. Это дает наименьшее время сортировки.

Худший случай. В каждой итерации массив разделяется на вырожденный подмассив из одного опорного элемента и на подмассив из всех остальных элементов. Такое может произойти, если в качестве опорного на каждом этапе будет выбран элемент либо наименьший, либо наибольший из всех обрабатываемых.

Методы разработки алгоритмов

Основные методы:

1. Метод грубой силы
2. Метод декомпозиции
3. **Метод уменьшения размера задачи**

Метод уменьшения задачи

Основан на использовании связи между решением данного экземпляра задачи и решением меньшего экземпляра той же задачи.

Варианты:

- Сверху вниз (рекурсивно)
- Снизу вверх (без рекурсии)

Метод уменьшения задачи

Варианты уменьшения размера:

1. Уменьшение на постоянную величину
2. Уменьшение на постоянный множитель
3. Уменьшение переменного размера

Уменьшение на постоянную величину

Вычисление a^n

$$f(n)=\begin{cases} f(n-1)\cdot a, при\ n>1 \\ a \qquad\qquad\quad , при\ n=1 \end{cases}$$

На какую постоянную величину уменьшается задача?

Уменьшение на постоянный множитель

Вычисление a^n

$$a^n = \begin{cases} \left(a^{\frac{n}{2}}\right)^2, при\ n > 1 \\ a \qquad , при\ n = 1 \end{cases}$$

Для каких n справедлива данная формула?

Как нужно поменять формулу, чтобы учитывать все натуральные n?

Уменьшение на постоянный множитель

Вычисление a^n

$$a^n = \begin{cases} \left(a^{\frac{n}{2}}\right)^2, при\ n\ положительном\ четном \\ \left(a^{\frac{n-1}{2}}\right)^2 \cdot a, при\ нечетном\ n > 1 \\ a \quad , при\ n = 1 \end{cases}$$

Формула справедлива для любых натуральных значений n.

Уменьшение на переменный множитель

Алгоритм Евклида поиска НОД.

Основан на 2 утверждениях

1. НОД(m,n) = НОД(n,m mod n)
2. НОД(m,0) = m

Аргументы в правой части всегда меньше, чем в левой (как минимум со 2-ой итерации), но они не меньше ни на постоянное значение ни на постоянный множитель.

Метод уменьшения размера задачи Сортировка вставкой

Метод уменьшения на единицу применим к сортировке.

Предположим, что уже отсортирован массив длиной n – 1.

Чтобы отсортировать массив длины n, нужно найти позицию элемента A[n-1] в отсортированной части длины n-1.

Метод уменьшения размера задачи Сортировка вставкой

На каждом шаге алгоритма мы выбираем один из элементов входных данных и вставляем его на нужную позицию в уже отсортированном списке, до тех пор пока набор входных данных не будет исчерпан.

Метод выбора очередного элемента из исходного массива произволен; может использоваться практически любой алгоритм выбора.

Метод уменьшения размера задачи Сортировка вставкой

Пример:

89 **45** 12 68 -> 89 89 12 68 -> 45 89 12 68

45 89 **12** 68 -> 45 89 89 68 -> 45 45 89 68 -> 12 45 89 68

12 45 89 **68** -> 12 45 89 89 -> 12 45 68 89

Метод уменьшения размера задачи Сортировка вставкой

Вход: массив A из n элементов: A[0] … A[n-1]

```
for (int i = 1; i < n; i++)
{
   key = A[i];
   j := i – 1;
   while ( (j >= 0) && (A[j] > key) )
   {
    A[j + 1] = A[j];
    j = j – 1;
   }
   A[j + 1] = key;
}
```

Сортировка подсчетом

Сортировка подсчётом — алгоритм сортировки, в котором используется диапазон чисел сортируемого массива (списка) для подсчёта совпадающих элементов.

Применение сортировки подсчётом целесообразно лишь тогда, когда сортируемые числа имеют (или их можно отобразить в) диапазон возможных значений, который достаточно мал по сравнению с сортируемым множеством, например, миллион натуральных чисел меньших 1000.

Сортировка связных списков

В связных списках обращение к элементу по его номеру — ресурсоёмкая операция, требующая в среднем n/2 сравнений и обращений к памяти. В результате применение типичных алгоритмов сортировки становится крайне неэффективным, т.к. они требуют для своей работы возможности обращения к элементам сортируемого списка по их индексам.

Однако у связных списков есть преимущество: возможность быстро объединить два отсортированных списка в один.

Объединение двух отсортированных списков

Началом результирующего списка из них выбирается элемент с наименьшим ключом.

Затем в качестве следующих элементов результирующего списка выбирается последующие элементы из первого или второго исходного списка, с меньшим значением ключа.

Когда достигнут последний элемент одного из исходных списков, указатель последнего элемента результирующего списка устанавливается на остаток другого входного списка.