Алгоритмы сортировки

Лекция 1

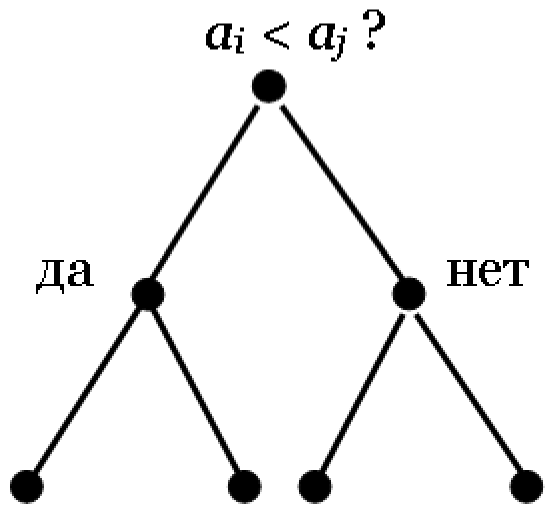
Сортировка

- *Сортировкой* называют упорядочение множества объектов по неубыванию или невозрастанию какого-нибудь параметра.

- Алгоритм пузырька (волновой алгоритм) — $O(n^2)$.
- Алгоритм Фон-Неймана — $O(n \log n)$.
- Пирамидальный алгоритм — $O(n \log n)$.
- QuickSort и др.

Сортировка с помощью сравнений

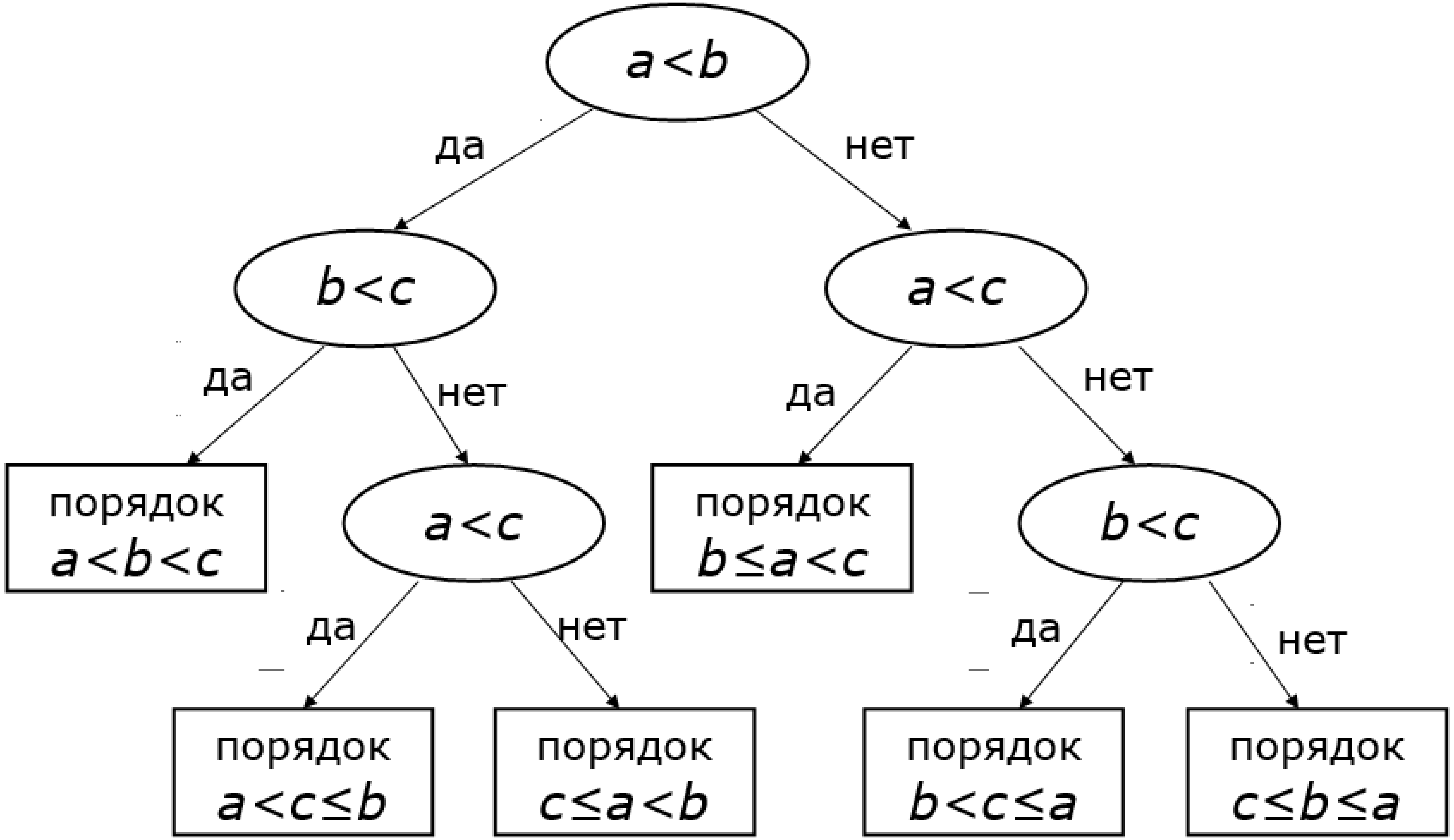
Лемма 1. Бинарное дерево высоты h содержит не более 2^h листьев.



Лемма 2. Высота любого дерева решений, упорядочивающего последовательность из n различных элементов, не менее $\log n!$.

Доказательство. Так как результатом может быть любая из $n!$ перестановок, то в дереве решений должно быть не менее $n!$ листьев.
Тогда по лемме 1 высота дерева не меньше $\log n!$.

Дерево решений для последовательности <a,b,c>



Нижняя оценка на количество сравнений

Теорема. В любом алгоритме, упорядочивающем с помощью сравнений, на упорядочивание последовательности из n элементов тратится не менее $cn\log n$ сравнений при некотором $c>0$ и достаточно большом n.

Доказательство. При $n \geq 4$ имеем

$$n! \geq n(n-1)\,(n-2)\ldots(\lceil \tfrac{n}{2} \rceil) \geq (\tfrac{n}{2})^{n/2} \quad ,$$

тогда

$$\log n! \geq (\tfrac{n}{2}) \log (\tfrac{n}{2}) \geq (\tfrac{n}{4}) \log n.$$

Алгоритм Фон-Неймана

- На вход подается последовательность чисел $a_1, \dots, a_n$.
- Алгоритм работает $\lceil \log_2 n \rceil$ итераций.
- Перед началом итерации с номером k ($k = 1, 2, \dots, \lceil \log_2 n \rceil$) имеется последовательность $a_{i_1}, \dots, a_{i_n}$ тех же чисел, разбитая на группы по 2^{k-1} элементов (последняя группа может быть неполной). Внутри каждой группы элементы упорядочены по не убыванию. Итерация состоит в том, что эти группы разбиваются на пары соседних групп, и элементы упорядочиваются внутри этих новых в два раза больших групп. При этом используется то, что внутри исходных групп элементы уже упорядочены.

$$\left.\begin{matrix} x_1,\dots,x_m \\ y_1,\dots,y_m \end{matrix}\right\} \Rightarrow z_1,\dots,z_{2m}$$

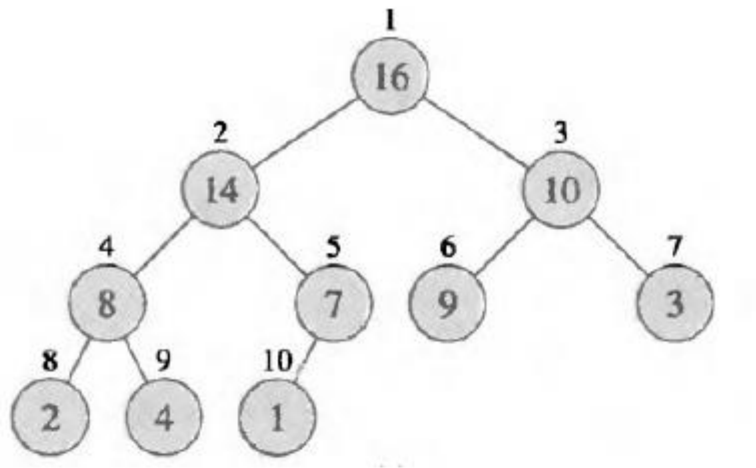
слияние за линейное время O(m)

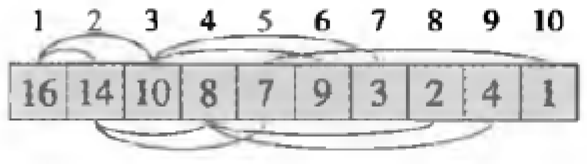
Пирамидальный алгоритм

- 1 этап: Построение пирамиды

(для каждого i: $a_i \geq a_{2i}$, $a_i \geq a_{2i+1}$)

Пирамида строится с середины!





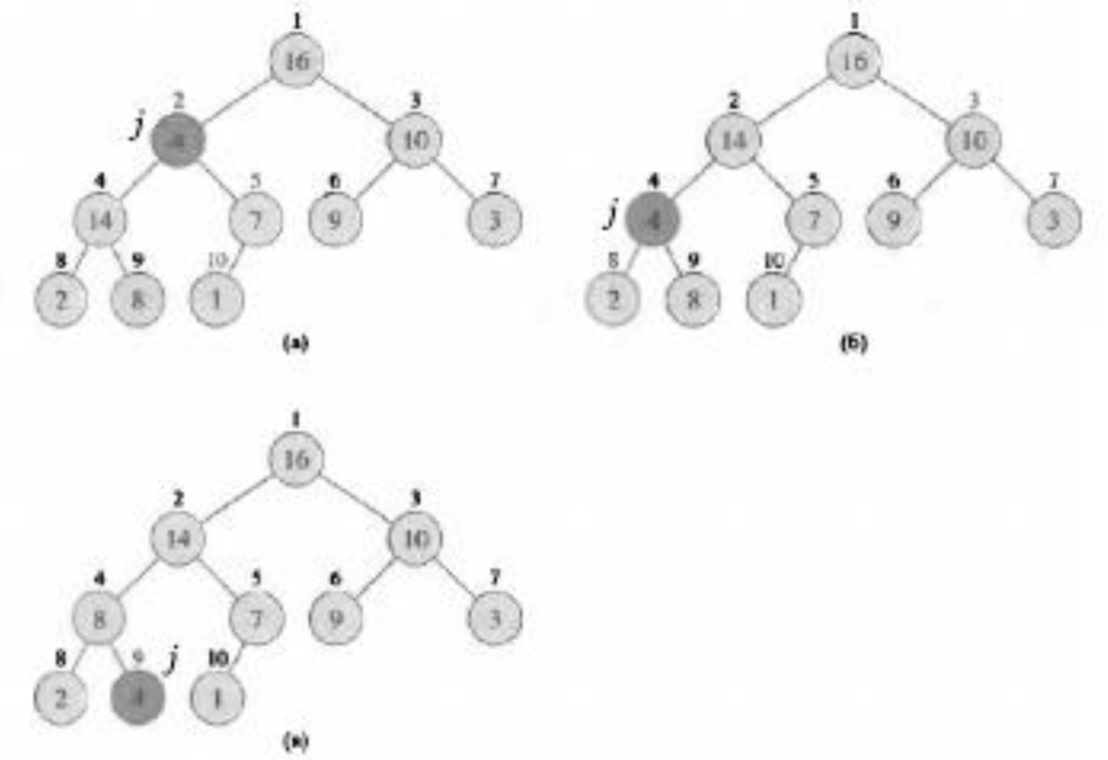
- 2 этап: Сортировка, используя пирамиду

Процедура «ПИРАМИДКА»

Алгоритм 1: Процедура **Пирамидка**(j, n)

if $(j \leq \frac{n}{2})$ **then**
  if $(A[2j] \leq A[2j+1])$ **then**
    $max \leftarrow 2j + 1;$
  else
    $max \leftarrow 2j;$
  if $(A[j] < A[max])$ **then**
    Поменять местами элементы $A[j]$ и $A[max]$;
    выполнить **Пирамидка**(max, n);

Пример процедуры Пирамидка(2,10)



Пирамидальная сортировка

- $\boldsymbol{For}\ j \leftarrow \left[\frac{n}{2}\right]\ to\ 1\ \boldsymbol{do}$
  - Выполнить Пирамидка (j,n);

1 этап

- $\boldsymbol{For}\ j \leftarrow 1\ to\ n\ \boldsymbol{do}$
  - Элемент a_1 складывается на полку;
  - Последний элемент пирамиды $a_{n\text{-}j+1}$ поставить на место элемента a_1;
  - Выполнить Пирамидка (1, n-j)

2 этап

Сортировка подсчётом

На входе n чисел $a[1],\ldots,a[n]$. Известно, что $0 \le a[i] \le K$.
Хорошо, если $K << n$.

for $i \leftarrow 1$ **to** K **do**
    $c[i] = 0;$
for $i \leftarrow 1$ **to** n **do**
    $c[a[i]] = c[a[i]] + 1;$ `// Подсчитываем, сколько раз встречается каждое значение`
$m = 0;$
for $i \leftarrow 1$ **to** K **do**
    for $j \leftarrow 1$ **to** $c[i]$ **do**
        $m = m + 1;$
        $a[m] = i;$ `// Заполняем упорядоченный массив`