

академия
больших
данных

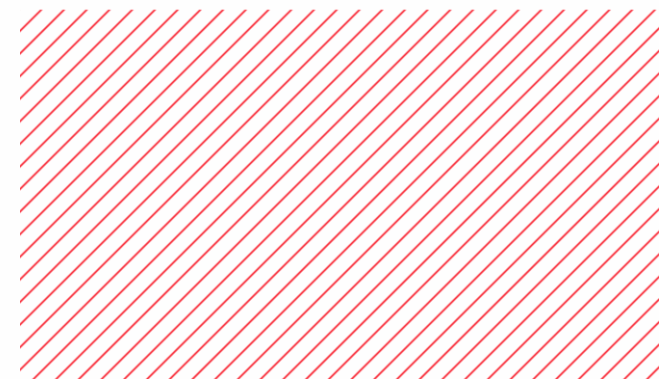
mail.ru
group

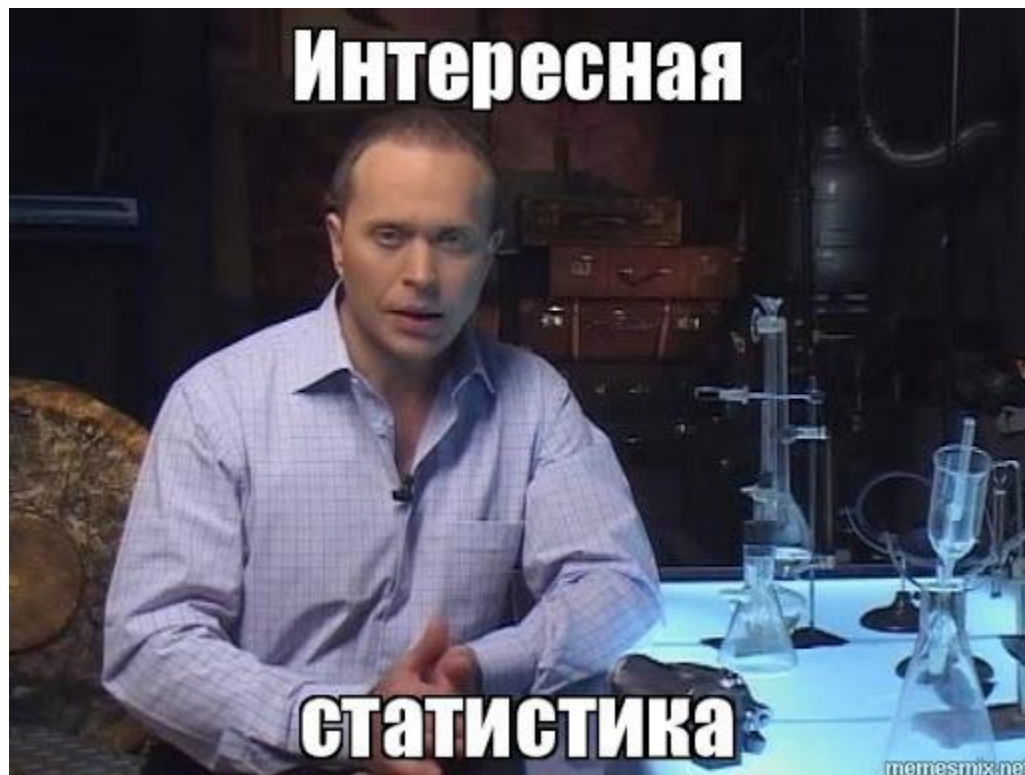


Сортировки-2 и k-я порядковая статистика

Шовкоплас Григорий

Введение в алгоритмы и структуры данных





К-я порядковая
статистика

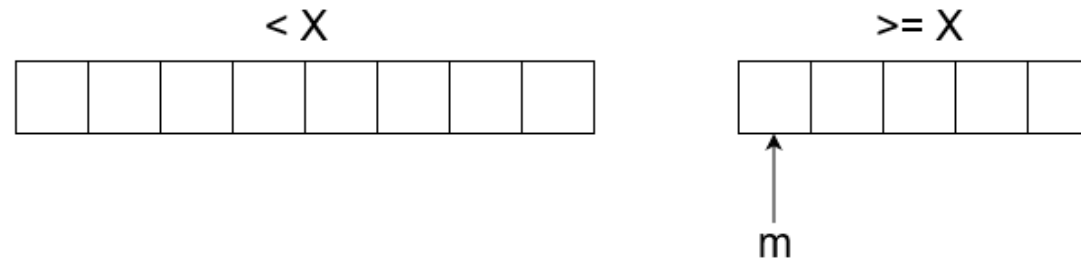


К-я порядковая статистика

- Дан массив чисел
- Требуется найти в нем К-й по величине элемент
- Как это сделать?
 - `Sort(a), return a[k]`
 - $O(n \log n)$
 - Можно ли быстрее?

Алгоритм Хоара

- Рассмотрим кусочки массива после операции $\text{split}(a, x)$
- Пусть в левой части m элементов



- Если $m > k$ рекурсивно переходим к левой части
- Иначе к правой
- Когда придем к массиву размера один, найдем ответ

Алгоритм Хоара

Псевдокод

Инвариант: $l \leq k < r$!

```
Finder
find(k, l, r)
    x = a[random(l..r-1)]
    m = split(l, r, x)
    if k < m
        return find(k, l, m)
    else
        return find(k, m, r)
```

Алгоритм Хоара

Опять забыли терминальное условие!

```
find(k, l, r)
    if r - l == 1
        return a[k]
    x = a[random(l..r-1)]
    m = split(l, r, x)
    if k < m
        return find(k, l, m)
    else
        return find(k, m, r)
```



Анализ

- Какие бы у нас могли возникнуть проблемы, если бы мы были любителями срезов из питона?
- Почему это будет работать быстрее, чем сортировка?
 - $E(T(n)) = 3 \times \left(n + \frac{2}{3}n + \frac{4}{9}n + \dots \right) =$
 - $= 3n \times \sum_{i=0}^{\infty} \left(\frac{2}{3}\right)^i =$
 - $= 3n \times \frac{1}{1-\frac{2}{3}} = 9n$
 - Работает за $O(n)$ в среднем получается!



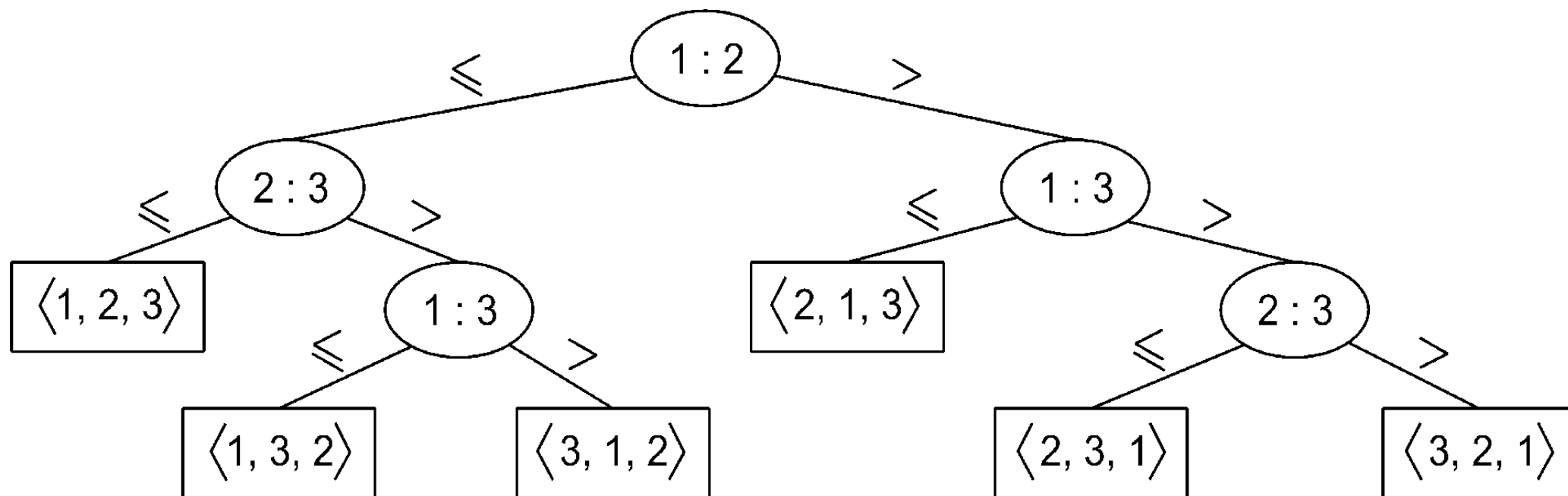
Есть ли сортировки
быстрее быстрой?



Время теорем

- Все рассмотренные сортировки требуют от элементов массива отношение порядка
- Давайте докажем, что нельзя сортировать быстрее, чем за $n \log n$
- Теорема о нижней оценке для сортировки сравнениями:
 - В худшем случае любой алгоритм сортировки сравнениями выполняет $\Omega(n \log n)$ сравнений

Теорема о нижней оценке для сортировки сравнениями





Теорема о нижней оценке для сортировки сравнениями

- Какая высота у данного дерева?
- Дерево двоичное $\Rightarrow h \geq \log_2(\text{leafs})$
- Всего перестановок $n! \Rightarrow h \geq \log_2(n!)$
- $\geq cn \log n$
- $h = \Omega(n \log n)$
- Можно ли сортировать быстрее?

"SORTING ALGORITHMS"

SelectionSort
 $\Omega(n^2)$



BubbleSort
 $\Omega(n)$



MergeSort
 $\Omega(n \log(n))$



CountingSort
 $O(n+k)$



Сортировка подсчетом



Сортировка подсчетом

- Дан целочисленный массив a , размера n
- $0 \leq a_i < m$
- Заведём вспомогательный массив cnt размера m
- Посчитаем сколько раз каждое число встречается в массиве
- Перезапишем исходный массив

Сортировка подсчетом

Псевдокод

Сколько работает?

$O(n + m)$

```
for i = 0 to n - 1  
    cnt[a[i]]++
```

```
i = 0  
for j = 0 to m - 1  
    while cnt[j] > 0  
        a[i++] = j  
        cnt[j]--
```



Сортировка подсчетом

- Можно сортировать отрицательные числа?
- Можно ли сортировать большие числа?
- Анаграммы?
- Можно ли сортировать кортежи по «маленькому» ключу?
 - Найдем диапазоны, в которых будут лежать элементы

Сортировка подсчетом пар по первому элементу

Псевдокод

Сколько работает?


$O(n + m)$

Бонус: устойчивость

```
Finder
for i = 0 to n - 1
    cnt[a[i].first]++

p[0] = 0
for i = 1 to m - 1
    p[i] = p[i - 1] + cnt[i - 1]

for i = 0 to n - 1
    a'[p[a[i].first]++] = a[i]
```

Цифровая
сортировка



Цифровая сортировка

- Дан целочисленный массив a , размера n
- $0 \leq a_i < m^k$
- Представим числа в m -ичной системе счисления
- У каждого k цифр (возможны ведущие нули)
- Будем сортировать их как кортежи по цифрам



Цифровая сортировка

- Если сортировать от старших разрядам к младшим долго 😞
- А можно ли от младших к старшим?
- Да, если у нас есть устойчивая сортировка... секундочку!
- Идем по возрастанию разряда и сортируем устойчиво по каждой следующей цифре
- За сколько работает?
 - $O(k(n + m))$
- Память $O(n + m)$



Bce!