Билет 18. k-ая порядковая статистика

Основная идея: Найти k-ый по величине элемент, не сортируя весь массив. Зачем работать, если можно работать меньше?

Определение

k-ая порядковая статистика — это k-ый по величине элемент в массиве.

- **1-ая порядковая статистика** = минимальный элемент
- **n-ая порядковая статистика** = максимальный элемент

Наивный подход (плохой)

```
1. Отсортировать весь массив - O(n log n)
2. Взять элемент на позиции k - O(1)
Итого: O(n log n) - слишком много работы!
```

Умный алгоритм (QuickSelect)

Идея: Используем разбиение из быстрой сортировки, но рекурсивно идём только в нужную часть!

```
Алгоритм QuickSelect(arr, k):
1. Выбрать pivot случайным образом
2. Разбить массив на три части:
   - left: элементы < pivot
   - equal: элементы = pivot
   - right: элементы > pivot
3. Если k в left: искать в left
   Если k в equal: вернуть pivot (уже нашли!)
   Если k в right: искать в right (скорректировав k)
```

Пример работы

Найти 4-ую порядковую статистику в массиве: [3, 1, 4, 1, 5, 9, 2, 6]

```
Шаг 1: pivot = 4 (случайно выбрали)
Разбиение: [3, 1, 1, 2] | [4] | [5, 9, 6]
           left (len=4)  equal right (len=3)

k = 4:
- left содержит элементы с индексами 0..3
- equal содержит элемент с индексом 4
- right содержит элементы с индексами 5..7

Наша k=4 попадает в equal → нашли! Ответ: 4
```

Более интересный пример: Найти 6-ую порядковую статистику

```
k = 6:
- left: индексы 0..3 (4 элемента)
- equal: индекс 4 (1 элемент)
- right: индексы 5..7 (3 элемента)

k = 6 > (4 + 1) = 5 → ищем в right
Новое k = 6 - 5 = 1 (первый элемент в right)

Рекурсивно в right [5, 9, 6]:
pivot = 6
Разбиение: [5] | [6] | [9]
k = 1 попадает в left → ответ: 5
```

Анализ времени работы

Худший случай:

$O(n^2)$ - постоянно невезучий выбор pivot

Средний случай (при случайном выборе pivot):

$O(n)$ - вот ради этого всё и затевалось!

Почему в среднем O(n)?

Математика:

$$T(n) = T(n/2) + O(n) \Rightarrow T(n) = O(n)$$

Интуиция:

- В среднем pivot делит массив пополам
- Каждый раз работаем только с половиной
- Сумма геометрической прогрессии: $n + n/2 + n/4 + ... = 2n = O(n)$

Визуализация уменьшения работы

```
Уровень 0: обрабатываем n элементов
Уровень 1: обрабатываем ~n/2 элементов
Уровень 2: обрабатываем ~n/4 элементов
Уровень 3: обрабатываем ~n/8 элементов
...
Сумма: n + n/2 + n/4 + n/8 + ... = 2n
```