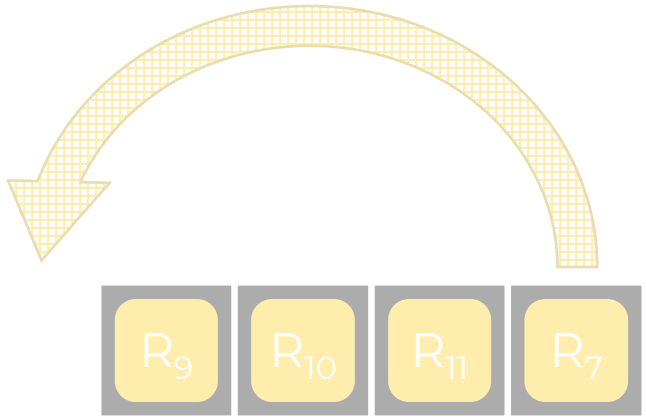


виикоорСт

Пришло время учиться оперировать данными!

Сортировки

Что у нас сегодня?

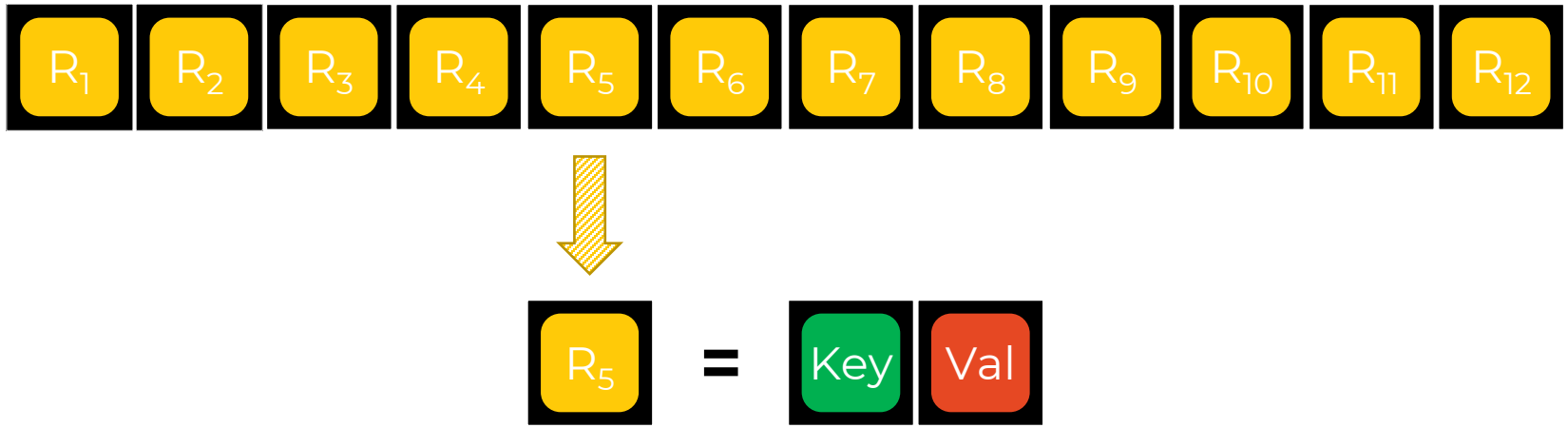
1. Сортированный массив
2. Сортировка вставками
3. Сортировка слиянием
4. Быстрая сортировка
5. k-я порядковая статистика
6. Случайные числа

Что такое сортированный массив?

Это массив, который уже отсортирован!

Что значит «отсортирован»?

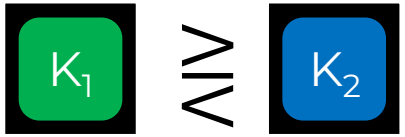
Дан массив элементов:



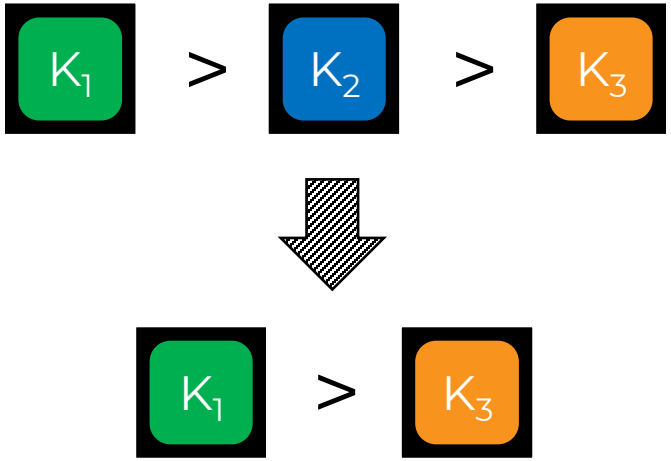
Условия для ключей

Для любых ключей

- Доступно сравнение:

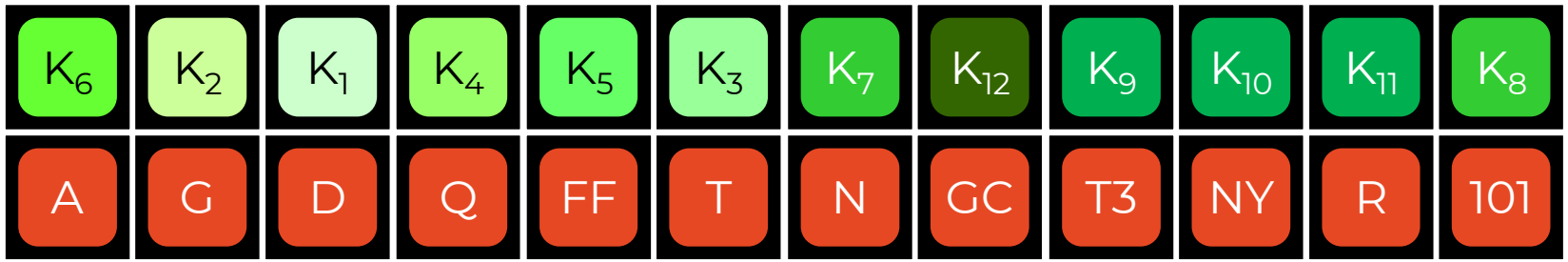


- Сравнение транзитивно:



Что значит «отсортирован»?

Дан массив элементов:

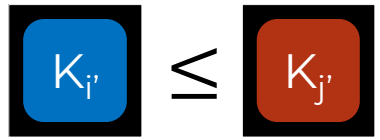


Задача сортировки

Дан массив элементов:

Нужно переставить элементы в новом порядке:

так, чтобы для любых 

, где $i' < j'$: 

Задача сортировки

Таким образом, задача сортировки элементов:

Эквивалентна задаче сортировки их ключей:

Так как в большинстве случаев ключи – **числа** или сводимые к ним сущности, то без ограничения общности можем рассматривать **сортировку чисел**.

Сортированный массив

В общем случае это тот массив, который удовлетворяет задаче сортировки.

На каком из массивов – сортированном или несортированном – проще производить операции?

- Операцию поиска?
- Операцию индексации?
- Операцию вставки/удаления?

Алгоритмы сортировки

Какие алгоритмы сортировки вы бы могли предложить?

Сортировка вставками

Как сортировать?

- **База**: Массив из одного элемента уже отсортирован;
- **Шаг**: Вставляем новый элемент в уже отсортированную часть массива.
- По индукции, этот способ сможет отсортировать массив любой длины.

Как делать шаг?

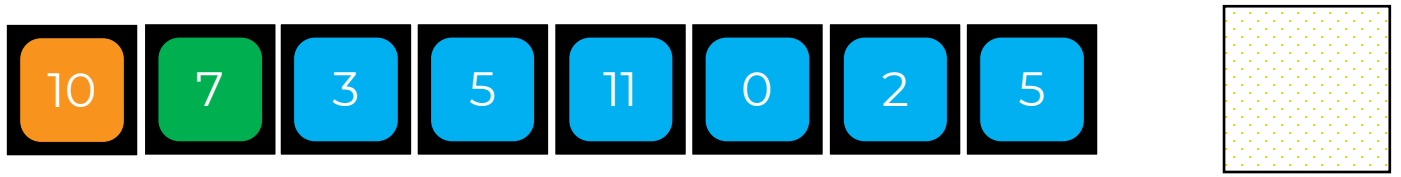
Сортировка вставками:

Пусть у нас есть массив:

Подмассив длиной 1 уже отсортирован.

Сортировка вставками:

Хотим вставить следующую переменную в отсортированный подмассив:



Заведем переменную для хранения промежуточного значения и скопируем туда 7.

Теперь у нас есть место, чтобы «сдвигать» элементы уже отсортированного массива, если они больше, чем значение в переменной.

Загадка

*Сколько времени будет работать сортировка вставками на отсортированном массиве длины **N**?*

- В случае сортировки вставками мы вообще не будем перемещать элементы. Ответ: **O(N)**.
- Таким образом, мы можем надеяться на **почти отсортированных массивах** работать линейное время.

Какова асимптотика сортировки вставками?

В лучшем случае?

O(N)

В среднем случае?

$O(N^2)$

В худшем случае?

$O(N^2)$

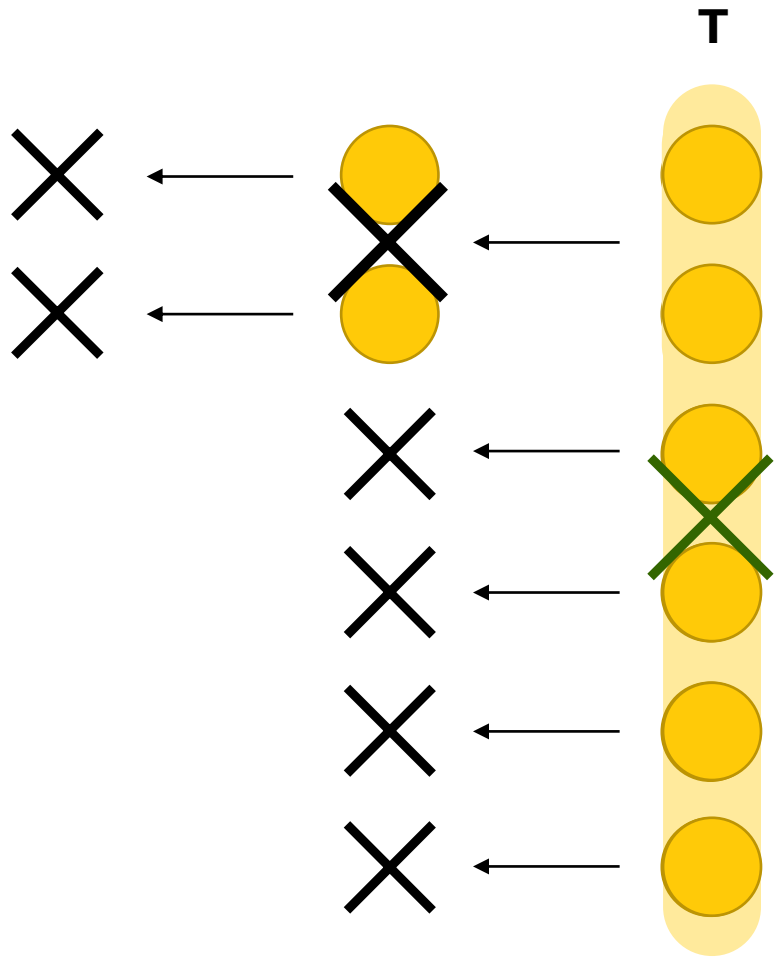
Затраты памяти?

O(1)

Стратегия «Divide and conquer»

Пусть у нас есть проблема **T**.

- **Divide**: разделим на несколько проблем поменьше.
- **Conquer**: разбиваем до тех пор, пока не получим такие части, для которых есть тривиальное решение.
- **Combine**: объединяя ответы на маленькие задачи, получаем ответы для больших.



Какой алгоритм из уже изученных в курсе использует эту стратегию?

Бинарный поиск

Сортировка слиянием (Merge sort)

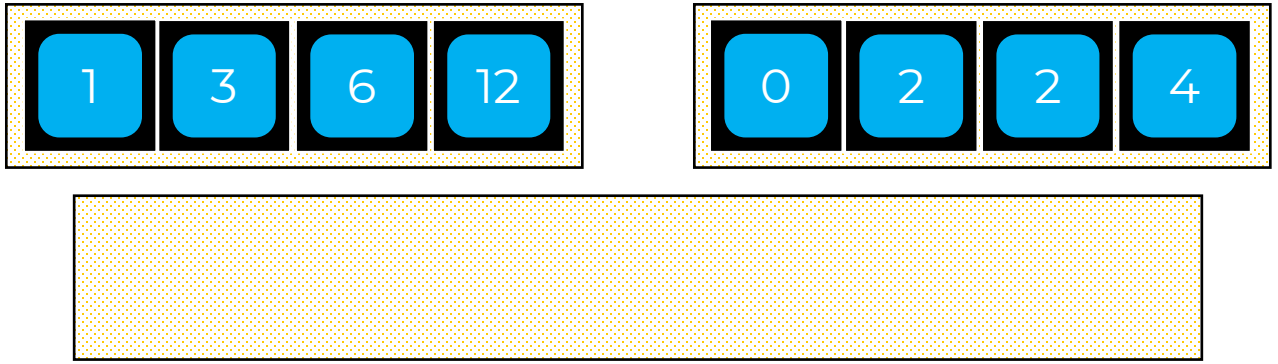
Как сортировать?

- **База**: Массив из одного элемента уже отсортирован;
- **Шаг**: Объединяем два отсортированных подмассива.
- По индукции, этот способ сможет отсортировать массив любой длины.

Как делать шаг?

Процедура Merge

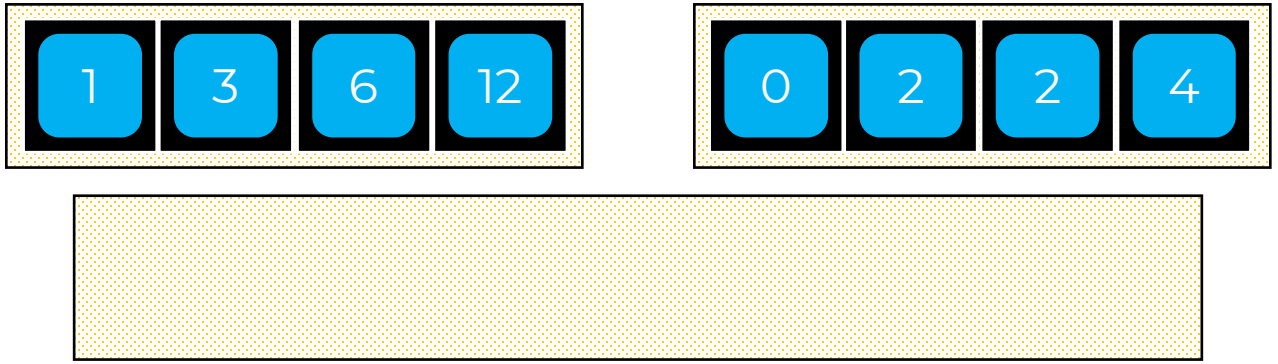
Пусть у нас уже есть два отсортированных подмассива:



Создаем пустой массив, размер которого равен суммарному размеру двух подмассивов.

Процедура Merge

Смотрим на начала подмассивов и копируем наименьший элемент из двух:

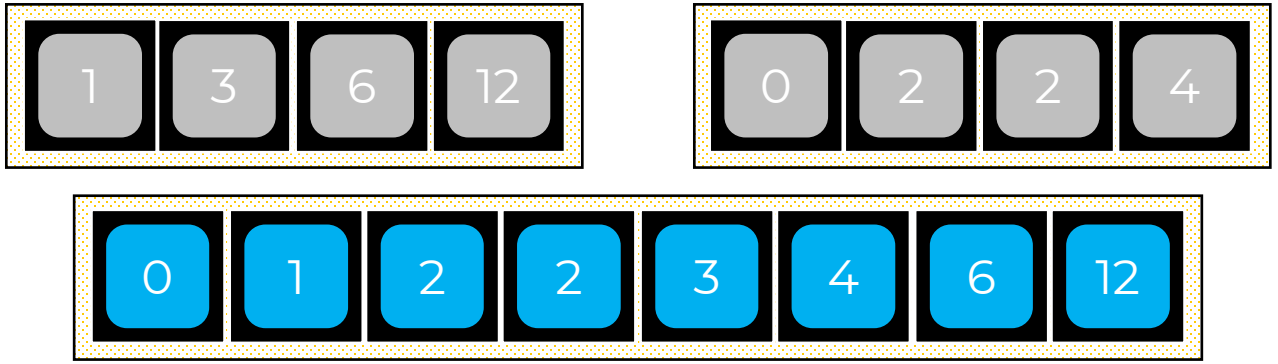


Сдвигаемся на один элемент в массиве, из которого копировали элемент на прошлом шаге.

Когда элементов в одном из массивов не осталось – просто копируем оставшиеся из другого

Процедура Merge

Смотрим на начала подмассивов и копируем наименьший элемент из двух:



Сдвигаемся на один элемент в массиве, из которого копировали элемент на прошлом шаге.

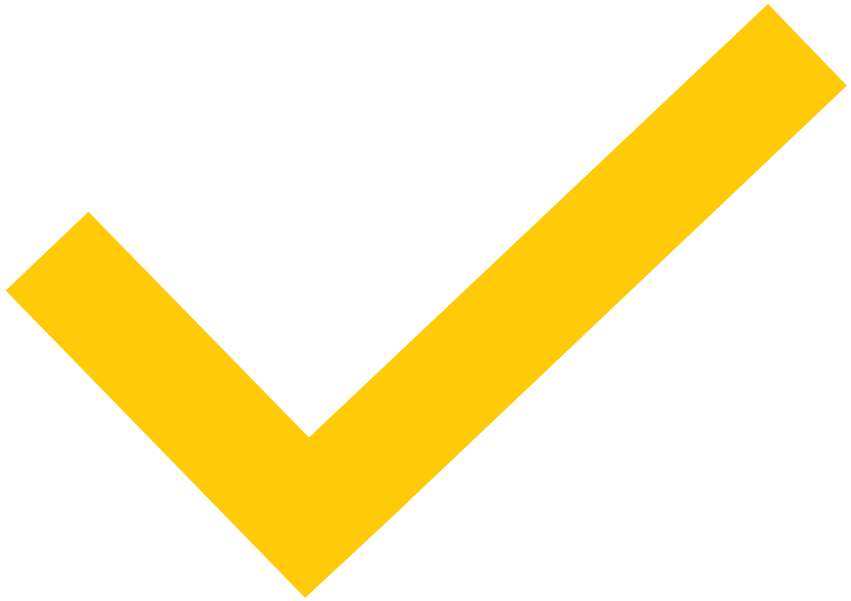
Когда элементов в одном из массивов не осталось – просто копируем оставшиеся из другого

Как произвести сортировку слиянием?

Делим массив до подмассивов длины 1

Подмассивы длины 1 уже отсортированы

Объединяем подмассивы Merge



Пишем псевдокод!

Какова асимптотика **MergeSort**?

В лучшем случае?

O(NlogN)

В среднем случае?

O(NlogN)

В худшем случае?

O(NlogN)

Затраты памяти?

По крайней мере O(N)

Быстрая сортировка

«Быстрая» сортировка

- **Divide**: Массив отсортирован тогда и только тогда, когда для каждого его элемента все элементы слева меньше или равны ему, а справа – больше или равны;
- **Conquer**: Массив из одного элемента уже отсортирован;
- **Combine**: Стараемся избежать явной комбинации для сохранения памяти.

Процедура Partition

Выбираем какой-то элемент массива (например, первый). Назовем его **опорным** (pivot). Идем с **правого** конца, пока не встретим элемент **меньше опорного**.

Когда встретили элемент, который **меньше опорного**, меняем его местами с опорным.

Процедура Partition

Дальше идем с **левого** конца, пока не встретим элемент, который **больше опорного**.

Когда встретили элемент, который **больше опорного**, меняем его местами с опорным.

Процедура Partition

Снова идем с **правого** конца, пока не встретим элемент, который **меньше** **опорного**.

Когда встретили элемент, который **меньше** **опорного**, меняем его местами с опорным.

Продолжаем операцию, пока не дойдем до **опорного** элемента.

За какое время работает процедура *Partition*?

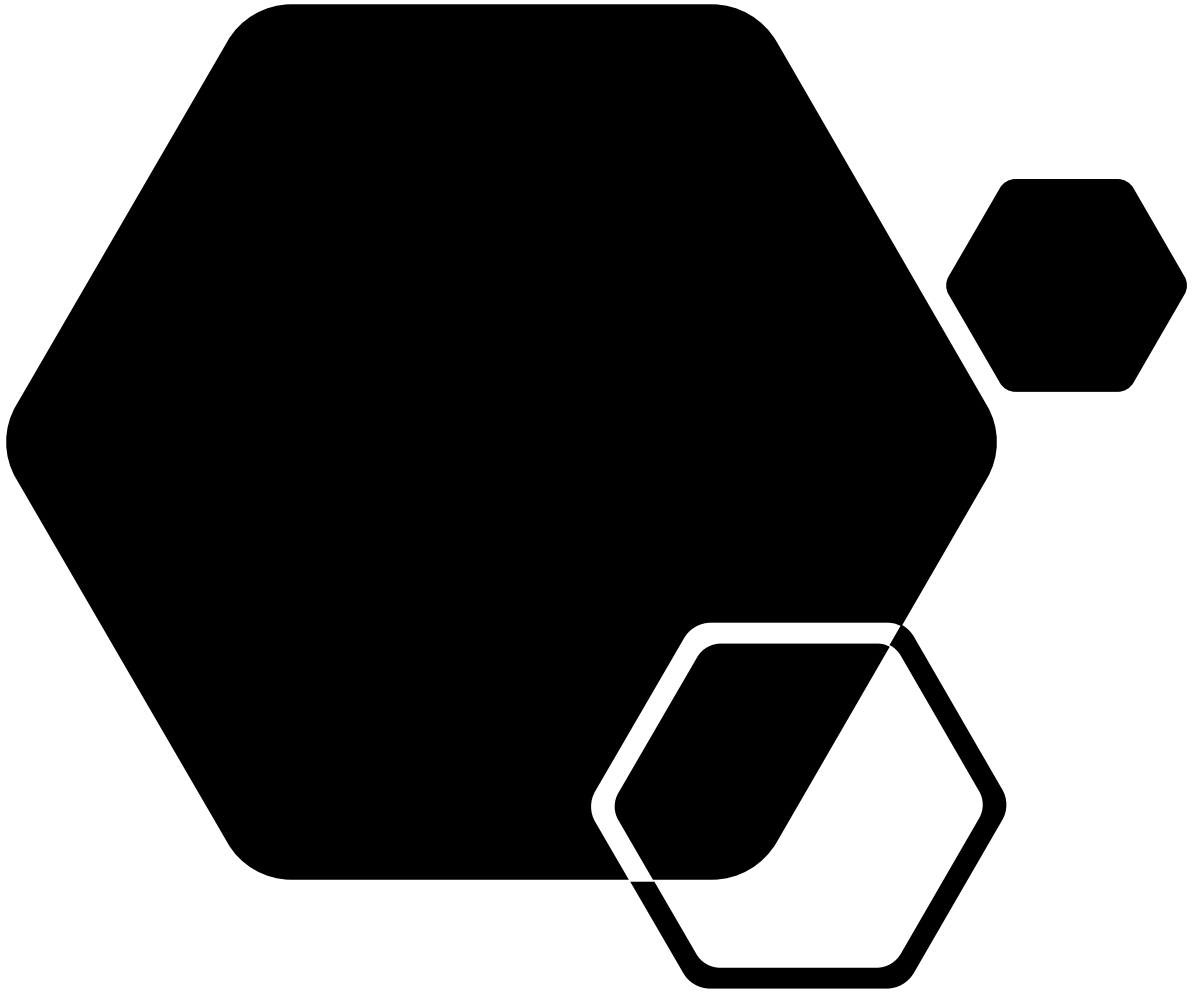
За O(N)

Combine в QuickSort

- После каждой процедуры Partition опорный элемент занимает свое итоговое место.
- Перемещение элементов происходит «на месте».

Благодаря этим двум факторам мы можем обойтись без явного шага **Combine**.

Пишем псевдокод!



Какова асимптотика **QuickSort**?

В лучшем случае?

O(NlogN)

В среднем случае?

O(NlogN)

В худшем случае?

$O(N^2)$

Затраты памяти?

O(logN)

Что лучше:
MergeSort или
QuickSort?

k-я порядковая статистика

k-й порядковой статистикой набора элементов линейно упорядочиваемого множества называется такой его элемент, который является **k**-м элементом набора в порядке сортировки.

TL;DR: Ответ на вопрос: «Какой элемент окажется на **k**-м месте?»

Как искать k-ю порядковую статистику?

Подсказка: вспомнить процедуру *Partition*

Действительно, если опорный элемент оказался на k-й позиции, то k-я порядковая статистика найдена!

Если же нет, рекурсивно найдем ее в одной из частей массива.

Пишем псевдокод!

Какова асимптотика поиска k-й порядковой статистики?

В лучшем случае?

O(N)

В среднем случае?

O(N)

В худшем случае?

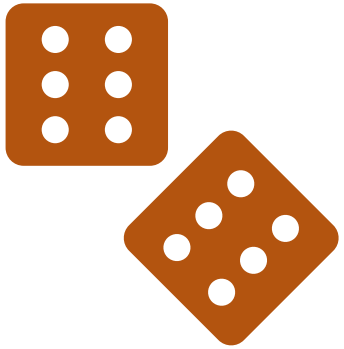
$O(N^2)$

Затраты памяти?

O(1)

Как устроена псевдослучайность?

Для получения **псевдослучайного числа** необходимы:

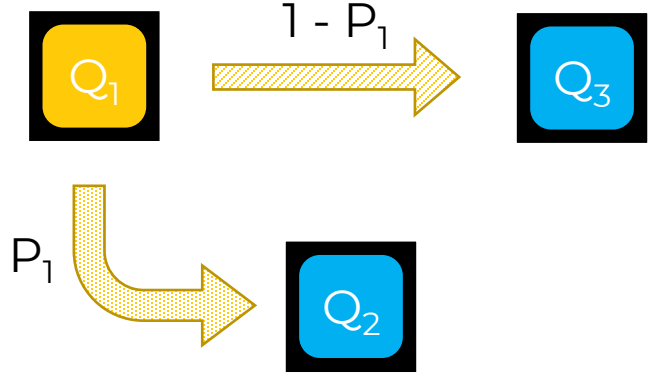


- Входные данные – **сид**;
- **Псевдослучайный генератор** – функция, генерирующая более информативные данные по сложному правилу.

Данные, которые выдает генератор, не являются случайными на самом деле, но их значения похожи на те, которые можно получить истинно случайной генерацией.

Рандомизированные алгоритмы

Рандомизированным называется алгоритм, в котором выполнение одного или несколько шагов основано на **случайном правиле**, т. е. среди многих детерминированных правил одно выбирается случайно в соответствии с **вероятностью P**.



Как можно улучшить работу **QuickSort** в худших случаях с помощью рандомизации?

Перемешаем массив перед сортировкой!

Спасибо за внимание!