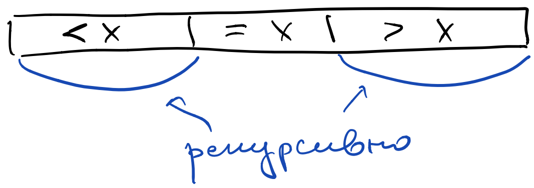
Ещё о сортировках

1. QuickSort 3

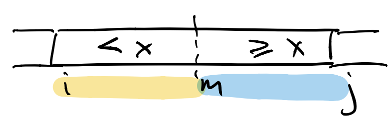


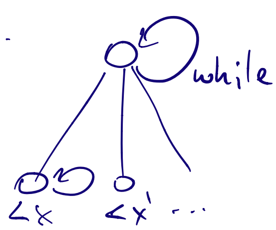
2. E[глубины рек. QS] $= O(\log n)$ (без док-ва)

3. IntroSort

Запускаем QS и если глубина рекурсии $\geq c \log n \Rightarrow$ останавливаем, запускаем Heap Sort. $\Rightarrow$ сложность $O(n \log n)$ и глубина $O(\log n)$

4. QuickSortRec(A, i, j):





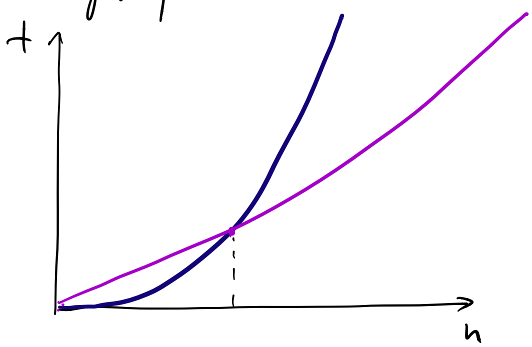

```
QuickSortRec(A, i, j):
    while j-i > 0:
        px = pivot(A, i, j)
        m = partition(A, i, j, px)
        if m - i > j - m:
            QuickSortRec(A, m+1, j)
            j = m
        else:
            QuickSortRec(A, i, m-1)
            i = m+1
```

Глубина $O(\log n)$ в худшем

5. QuickSort Stable

Реализуем partition рекурсивно через циклические сдвиги $O(n\log^2 n)$ в среднем

6. При реализации часто на маленьких массивах используют квадратичные сортировки



7. Partial Sort

Найти k самых больших эл-ов из n

Используем HeapSort $O(k\log n + n)$

$k < \frac{n}{\log n} \Rightarrow O(n)$

Линейные сортировки

1. Сортировка подсчётом (Counting Sort)

Нам дан массив размера n с целыми числами из $[0, m)$

Можно отсортировать за $O(n+m)$

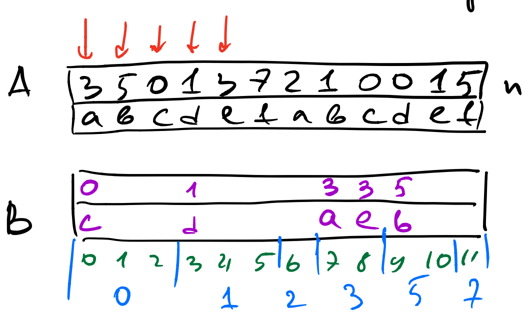
A $\boxed{3\ 5\ 0\ 1\ 3\ 7\ 2\ 1\ 0\ 0\ 1\ 5}$ n C $\boxed{3\ 3\ 1\ 2\ 0\ 2\ 0\ 1}$ m

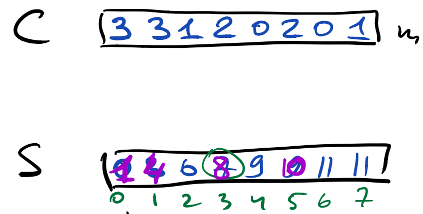
B |0 0 0 1 1 1 2 3 3 5 5 7|

Время $O(n+m)$

Память $O(m)$

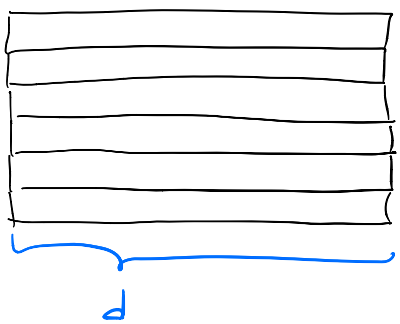
1.1. Стабильная сортировка подсчётом

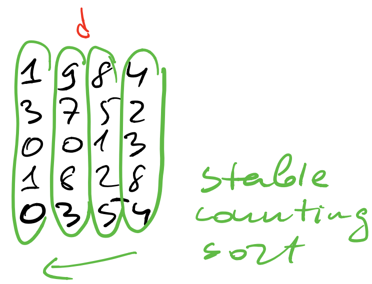




$$S[k] = \sum_{i=0}^{k-1} C[i]$$

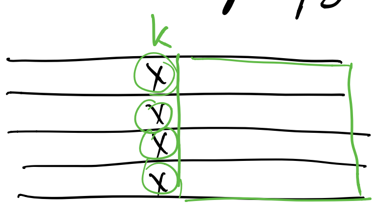
2. Цифровая сортировка (Radix Sort)





Время $O(d \cdot n)$

когда мы сортируем по k-ому разряду



3. Bucket Sort (сортировка вычёрпыванием, вёдерная сортировка)

Вещественные числа на $[0, 1]$

В $\forall$ ведре используем сортировку вставками.

Утв: В предположении о равномерном распределении эл-ов, время работы алгоритма $O(n)$ в среднем.

$\exists$ n_i – кол-во эл-ов в ведре i.

$$\mathbb{E}[T(n)] = \mathbb{E}\Big[\sum_{i=0}^{n} O(n_i^2)\Big] = O(n)$$

$$\mathbb{E}[n_i] = 1$$

$$\mathbb{E}[n_i^2] = D[n_i] + \mathbb{E}^2[n_i] = 2 - \frac{1}{n}$$

$$p = \frac{1}{n} \Rightarrow 1 \qquad P[n_i = k] = C_n^k p^k (1-p)^{n-k}$$

$$1 - p \Rightarrow 0$$

$$D[n_i] = np(1-p) = n \cdot \frac{1}{n} \cdot \frac{n-1}{n} = 1 - \frac{1}{n}$$

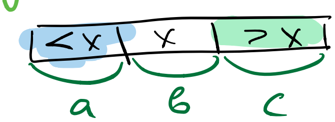
Порядковые статистики

$\equiv$ k-ая порядковая статистика массива A – это эл-т, который будет стоять на позиции k в отсортированном A.

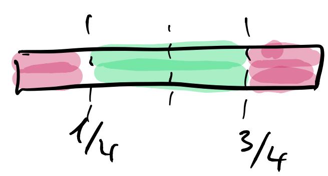
Утв: Порядковую статистику можно найти за $O(n)$ в худшем (см. „Медиана медиан“)

Нахождение k-ой статистики за $O(n)$ в среднем.

```
Randomized-Selection(A, i, j, k)
    x ← Pivot(A, i, j)        // случайный
    Partition3(A, i, j, x)
    if k ≤ a:
        Randomized-Selection(A, i, i+a, k)
    else if k > a+b:
        Randomized-Selection(A, i+a+b, j, k-a-b)
    else:
        return x
```



≡ Разбиение хорошее:



Сколько вызовов нужно, чтоб размер массива уменьшился в $4/3$ раза в среднем

$$\mathbb{E}[\#\text{ разбиений до хорошего}] = 2$$

$$\mathbb{E}[T(n)] \leqslant \mathbb{E}\left[T\left(\tfrac{3}{4}n\right)\right] + \underbrace{O(n)}_{\text{partition}} \cdot \underbrace{\mathbb{E}[\#\text{разбиений}\ldots]}_{=2}$$

$$\leqslant \mathbb{E}\left[T\left(\tfrac{3}{4}n\right)\right] + O(n) = O(n)$$

$$T'(n) = \mathbb{E}[T(n)]$$

$$T'(n) \leqslant T'\left(\tfrac{3}{4}n\right) + O(n)$$