Сортировка подсчетом

a[ ] $a_i < C$

$O(n + C)$

1 2 1 3 4 5 1 1 → 1111 2 33 4 (4, 1, ...)

```
for (c = 1 ... C)
    for (i = 0; i < cnt[c])
        cout << c
```

$C + \underbrace{\sum_c cnt[c]}_{n}$ ⇓ $O(n + C)$

cnt
1 : 4
2 : 1
3 : 2
4 : 1

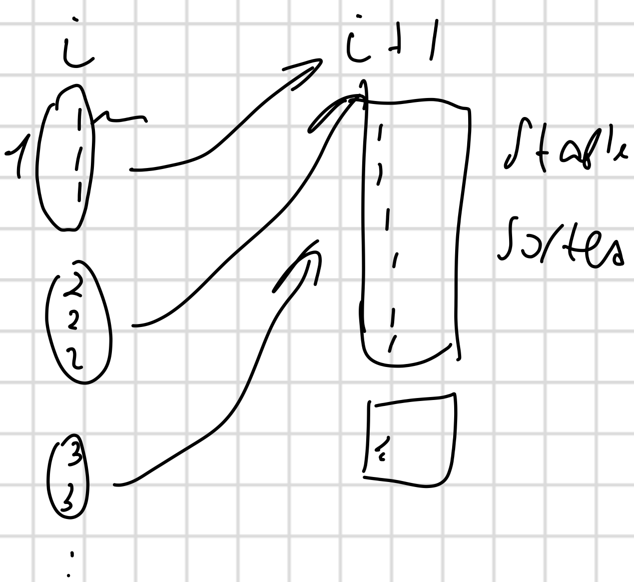
```
for (i = 0 ... n)
    cnt[a[i]]++
```

$O(n)$

Цифровая сортировка

	I	II	III
321	321	321	123
123	231	222	222
333	222	123	231
231	123	231	321
222	333	333	333

sort
stable



$i = 0$

a[i] i

cnt[c] = #(c)

for (j = 0...n)

    cnt[a[j][i]]++

for (j = 1, C)

    pos[i] = pos[i-1] + cnt[i-1]

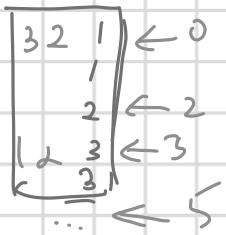
for (j = 0...n)

    ans[pos[a[j][i]]] = a[j]

    pos[a[j][i]]++

321 i=0 cnt = [0, 2, 1, 2, 0...] (0 1 2 3...)

121
333
231
222

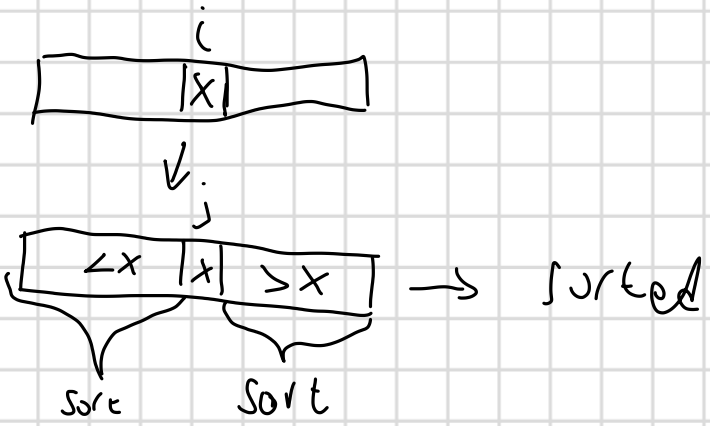


pos[i] - первая свободная позиция для строки i

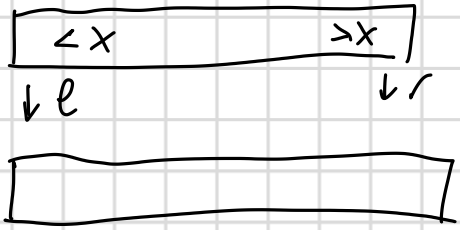
pos = [0, 0, 2, 3, 5, 5, 5]

pos[i] = pos[i-1] + cnt[i-1]

Quick Sort



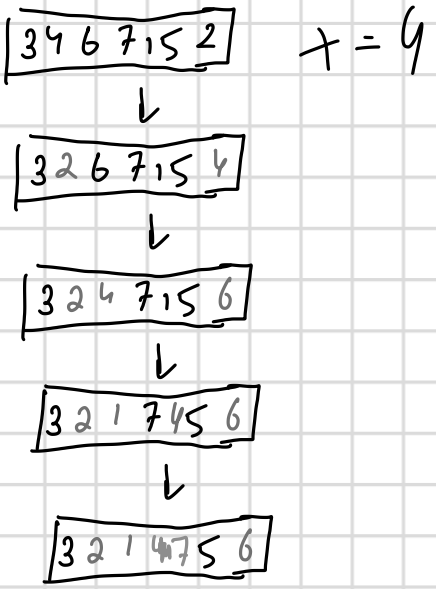

```
sort ( l, r )        [l, r)
    x = a[ rnd(l, r) ]
    j = partition ( l, r, x )   ← возвр. позицию x
     sort ( l, j )
     sort ( j+1, r )
```



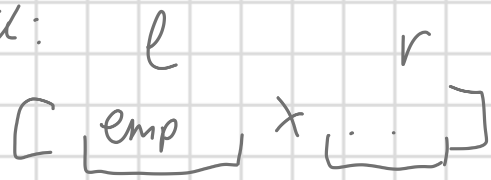
while a[l] < x => l++

while a[r] > x => r--

when Swap (a[l], a[r])

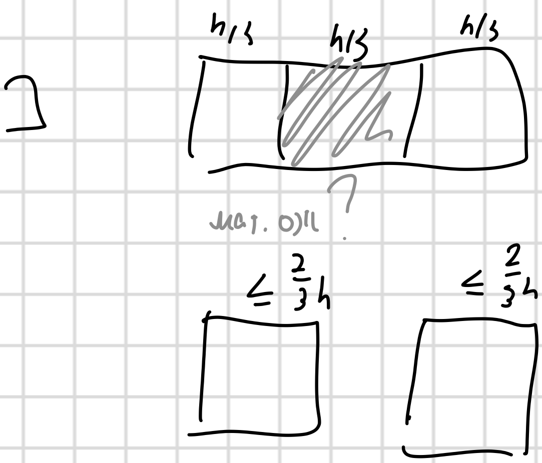


иск:



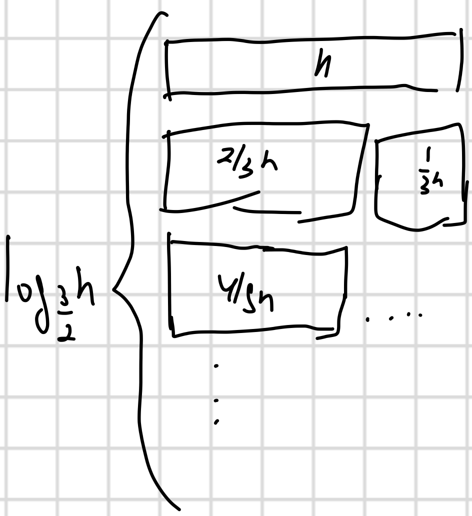
$T(n) = T(n-1) + n$ в худш. случ. ($O(n^2)$)

$T(n) = T(\frac{n}{2}) + T(\frac{n}{2}) + n$ — из лучш. случая



$T(n-1) + T(1) + n$

$T(n) = T(\frac{n}{3}) + T(\frac{2}{3}n) + n = n \cdot \log_{\frac{3}{2}} n = O(n \log n)$



посчитаем мат ожидание попасть в центр

исходы:

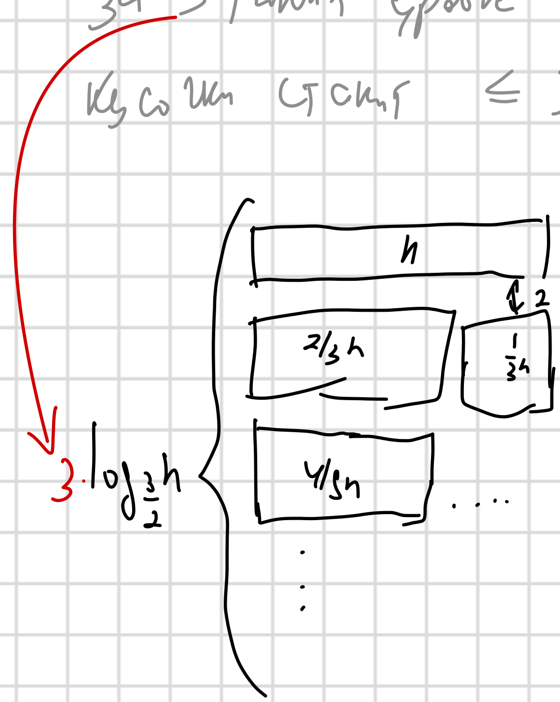
1) попал на i шаге

P попал → $\frac{1}{3}$ $\left(\frac{2}{3}\right)^{i-1}$ ← P не попал

$X = i$

$$\sum i \cdot \frac{1}{3} \cdot \left(\frac{2}{3}\right)^{i-1} = 3$$

примерно за 3 текущих уровня все эти кусочки станут $\leq \frac{2}{3}n$



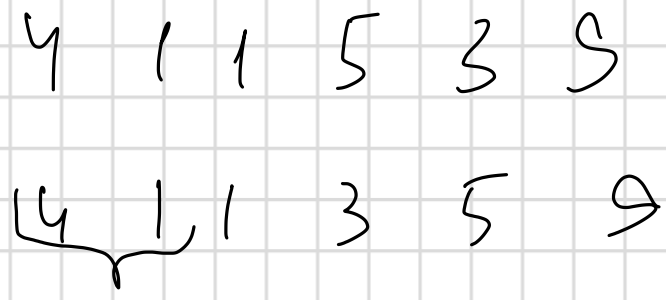

```
sort(l, r)
    if (r - l == 1)
        return
    x = a[rand(l, r)]
    m = partition(l, r, x)
    sort(l, m)
    sort(m, r)

partition(l, r, x)
    i = l
    j = r - 1
    while (i < j)
```

```
if (a[i] < x)
    i++
    continue
if (a[j] > x)
    j--
    continue
swap(a[i], a[j])
i++
j--
```

T - какъв тип

```
quicksort(a,0,length[a]−1)
void quicksort(a: T[n], int l, int r)
    if l < r
        int q = partition(a, l, r)
        quicksort(a, l, q)
        quicksort(a, q + 1, r)
```



```
int partition(a: T[n], int l, int r)
    T v = a[(l + r) / 2]
    int i = l
    int j = r
    while (i ⩽ j)
        while (a[i] ⩽ v)
            i++
        while (a[j] > v)
            j--
        if (i ⩾ j)
            break
        swap(a[i++], a[j--])
    return j
```