

Функция **op** является **бинарным отношением** на множестве **X** если $op: X \times X \rightarrow \{\text{true}, \text{false}\}$
 Бинарное отношение можно записывать как $op(x, y)$, но мы будем чаще пользоваться $x \text{ op } y$

Бинарное отношение **op** на множестве **X** называется **отношением (нестрого) частичного порядка**, если если имеют место:

- Рефлексивность $x \text{ op } x \quad x \leq x \quad \leq \quad \geq$
- Антисимметричность $x \text{ op } y \& y \text{ op } x \rightarrow x = y \quad x \leq y \& y \leq x \rightarrow x = y$
- Транзитивность $x \text{ op } y \& y \text{ op } z \rightarrow x \text{ op } z \quad x \leq y \& y \leq z \rightarrow x \leq z$

Бинарное отношение **op** на множестве **X** называется **отношением строгого частичного порядка**, если если имеют место:

- Антирефлексивность $\neg(x \text{ op } x) \quad \cancel{x \leq x} \quad < \quad >$
- Асимметричность $x \text{ op } y \rightarrow \neg(y \text{ op } x) \quad x \leq y \quad y \cancel{\leq x} \quad \lambda \leq y \mid x = y$
- Транзитивность $x \text{ op } y \& y \text{ op } z \rightarrow x \text{ op } z \quad x < y \quad y < z \quad x < z$

Если на множестве задано отношение строгого частичного порядка **op1**, то на нем можно задать и отношение нестрогого частичного порядка $x \text{ op2 } y \Leftrightarrow (x = y) \mid x \text{ op } y$

Отношение (нестрого) частичного порядка на множестве **X** называется **линейным порядком** если любые два элемента из множества **X** сравнимы т.е. $x \text{ op } y$ или $y \text{ op } x$

В частности, отношения **больше либо равно (\geq)** и **меньше либо равно (\leq)** являются **отношениями (нестрого) частичного порядка (также являющимся линейным порядком)**, а отношения **больше ($>$)** и **меньше ($<$)** являются **отношениями строгого частичного порядка** на множестве действительных чисел.

Задача сортировки это задача упорядочивания элементов массива, в соответствии с некоторым **отношением порядка**. Мы будем рассматривать только сортировки на **линейно упорядоченных множествах**.

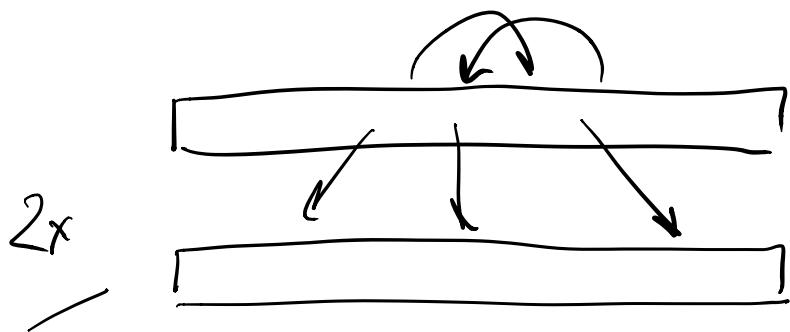
На вход алгоритм принимает упорядоченный набор элементов, на выход алгоритм выдает перестановку элементов в неубывающем порядке

Сортировка называется **стабильной**, если она не меняет местами равные элементы

Сортировка называется **сортировкой на месте (inplace)** если результат сортировки находится в той же памяти, что и входные данные. При этом алгоритм должен использовать константное количество дополнительной памяти

$$\begin{array}{cccc}
 a_1 & a_2 & a_3 & a_4 \\
 (1,2) & (2,3) & (2,4) & (3,4) \\
 (1,2) & \underline{(2,4)} & \underline{(2,3)} & (3,4) \\
 \end{array}$$

$a[0] \leq b[0]$

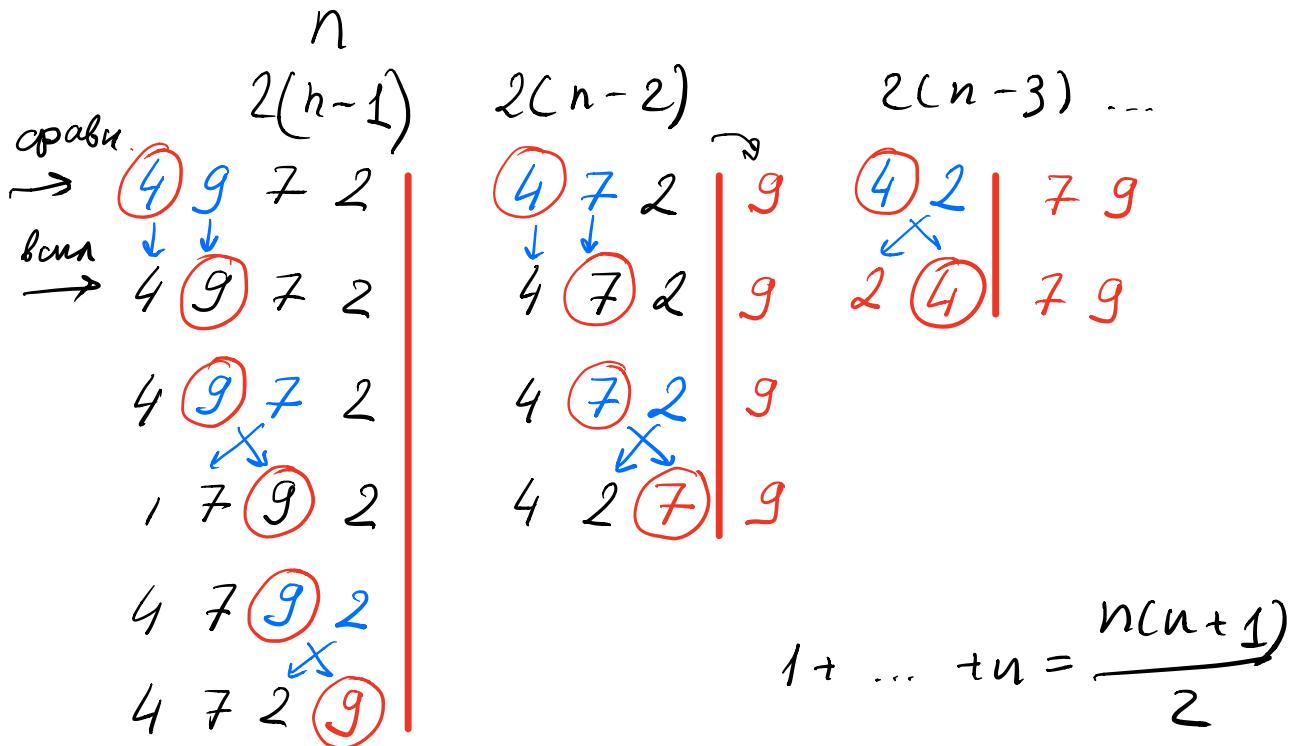


["Mama",
"Малышка",
"Мила",
"Рада",]

["a", "u", "A", "m", "y"]

]

Bubble Sort (сортировка пузырьком)

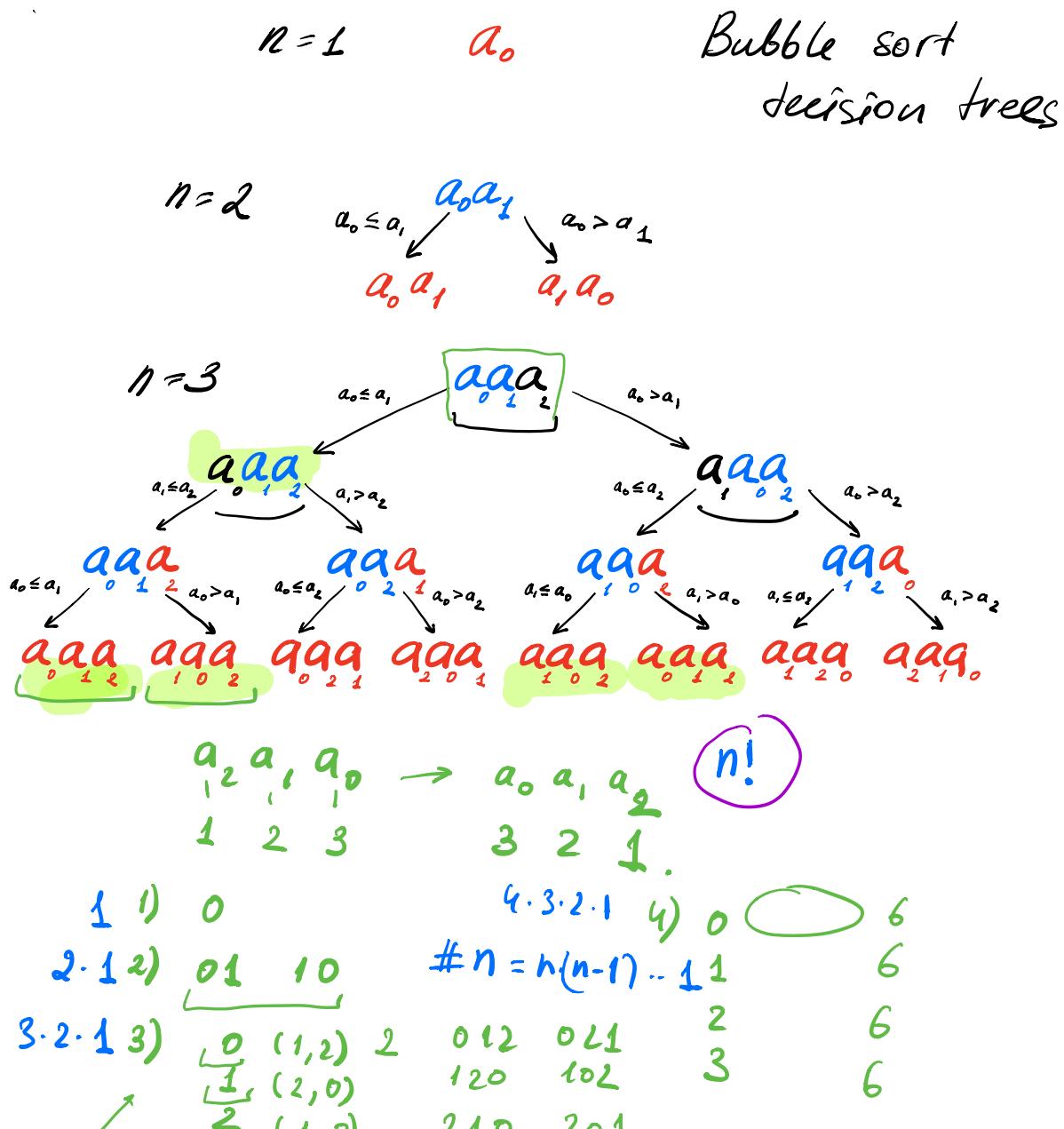


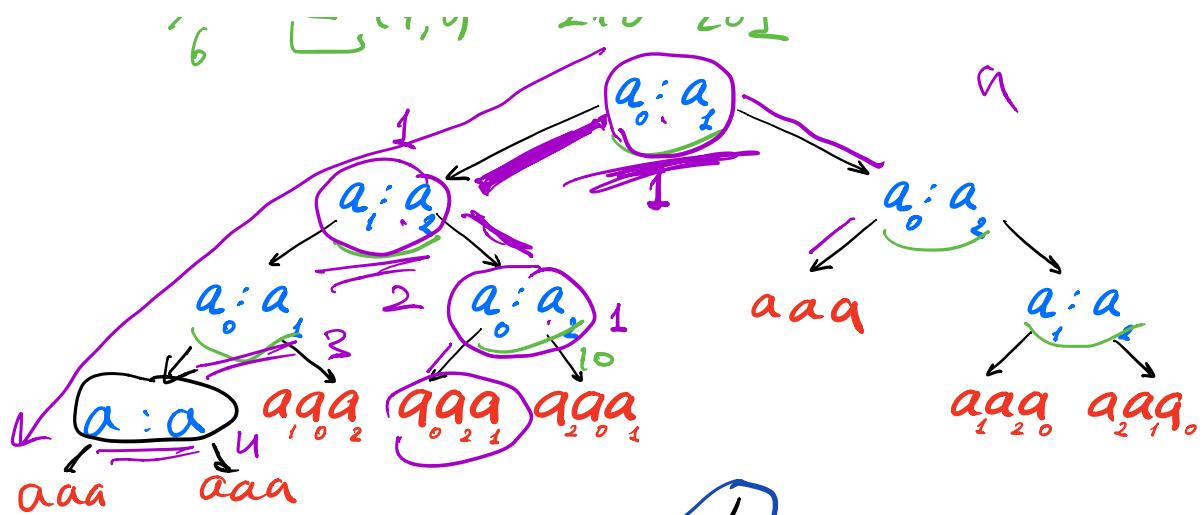
$$\begin{aligned}
 & 2(n-1) + 2(n-2) + \dots + \underbrace{2 \cdot 1} = \\
 & = 2(1 + 2 + \dots + (n-1)) = \\
 \rightarrow & 1 + 2 + \dots + n-2 + n-1 + \\
 \rightarrow & (n-1) + (n-2) + \dots + 2 + 1 \\
 & \cancel{n} + n + \dots + n + n = n(n-1) \\
 & \quad \# n-1
 \end{aligned}$$

Говорят, что алгоритм сортировки **основан на сравнениях**, если он никак не использует внутреннюю структуру сортируемых элементов? а лишь сравнивает их и после и после некоторого числа сравнений выдает ответ

Для таких сортировок мы можем ввести понятие **решающего дерева**.

Решающее дерево это бинарное дерево, представляющее последовательность сравнений элементов для конкретного алгоритма на входах конкретной длины. В узлах дерева находятся пары элементов, сравниваемые алгоритмом, а в листьях - перестановки исходного массива





$$= \frac{5}{n!} \leq \# \text{nodes} \leq 2^h$$

$$\rightarrow n! \leq 2^h$$

$$\log_2 n! \leq h$$

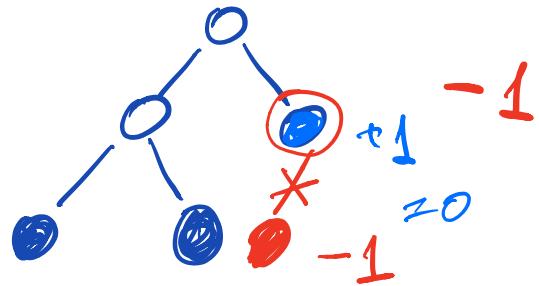
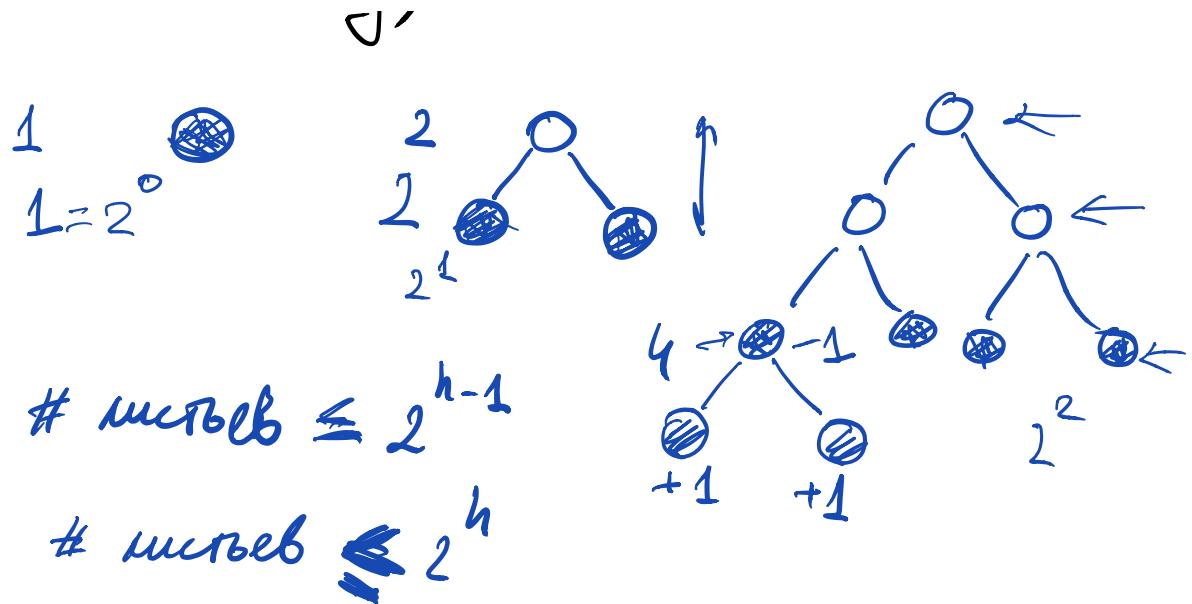
$$\left\lfloor \frac{n}{2} \right\rfloor \left\lceil \frac{\frac{n}{2}}{2} \right\rceil \geq 3.5 \\ \left\lceil \frac{\frac{n}{2}}{2} \right\rceil = 4$$

$$n! = 1 \cdot 2 \cdot 3 \cdots \left\lfloor \frac{n}{2} \right\rfloor \left(\left\lfloor \frac{n}{2} \right\rfloor + 1 \right) \cdots n \geq \\ \geq \underbrace{1 \cdot 1 \cdots 1}_{\left\lfloor \frac{n}{2} \right\rfloor} \cdot \underbrace{\Gamma \left(\frac{n}{2} \right)}_{\left\lceil \frac{n}{2} \right\rceil} \cdot \underbrace{\Gamma \left(\frac{n}{2} \right)}_{\left\lceil \frac{n}{2} \right\rceil} \cdots \underbrace{\Gamma \left(\frac{n}{2} \right)}_{\left\lceil \frac{n}{2} \right\rceil} = \\ = \Gamma \left(\frac{n}{2} \right)^{\left\lceil \frac{n}{2} \right\rceil} \geq \left(\frac{n}{2} \right)^{\frac{n}{2}}$$

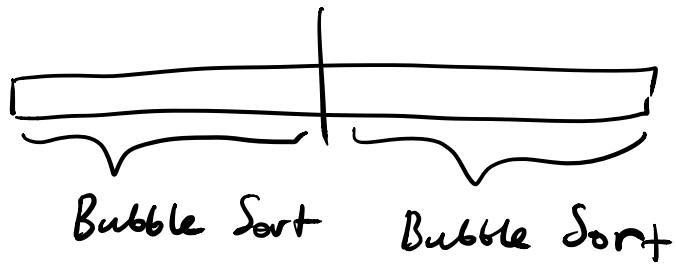
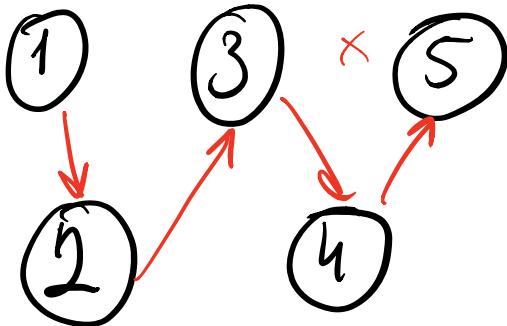
$$\log_2 \left(\frac{n}{2} \right)^{\frac{n}{2}} = \frac{n}{2} \log_2 \frac{n}{2} \leq \log_2 n! \leq h$$

$$\boxed{\frac{n}{2} \log \frac{n}{2} \leq h} \leq T(n)$$

$O(n \log n)$



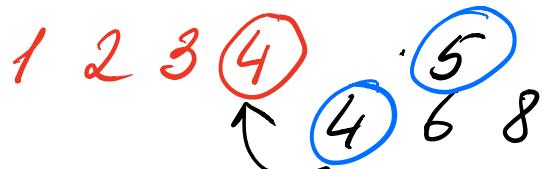
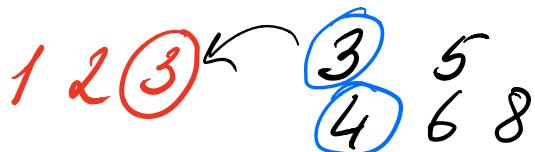
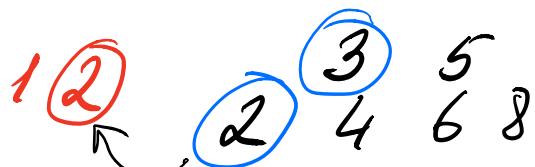
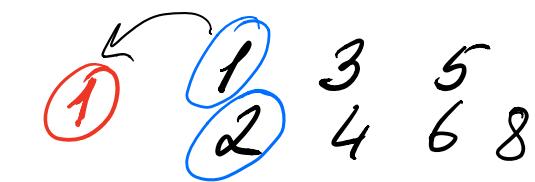
Merge



```
def merge_sort(array):  
    ↗ x = sort(array[: $\frac{n}{2}$ ])  
    ↗ y = sort(array[ $\frac{n}{2}$ :])  
    if len(array) ≤ 1  
        return array  
    else:  
        return merge(x, y)
```

Merge sort

One pass all
merge

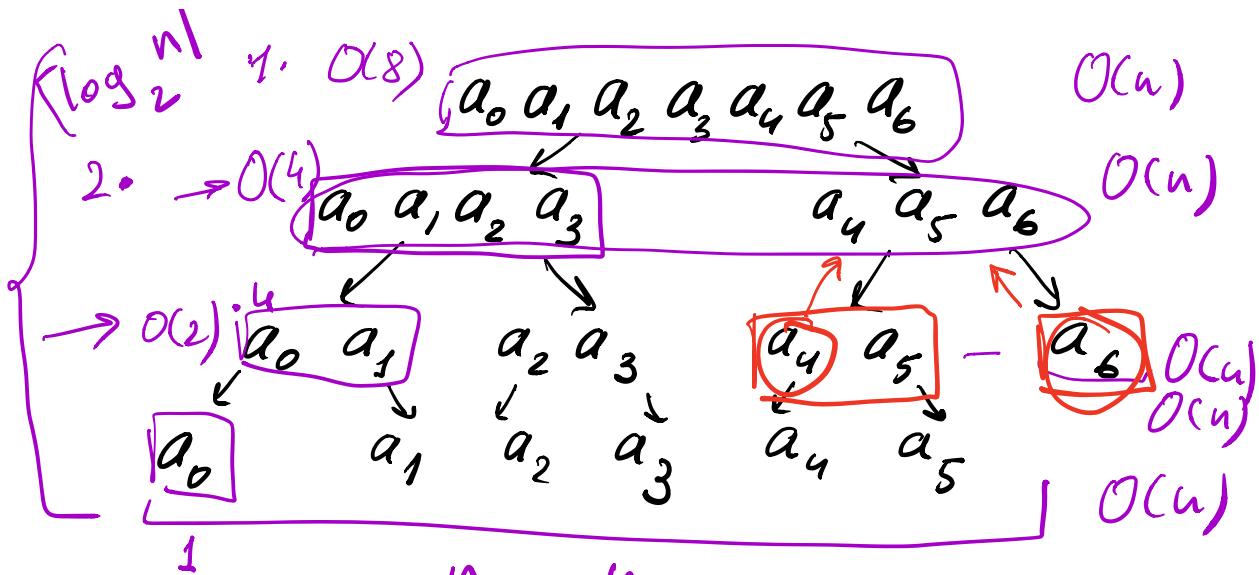


1 2 3 4 5 6 8

1 2 3 4 5 6 8

$\mathcal{O}(n+m)$

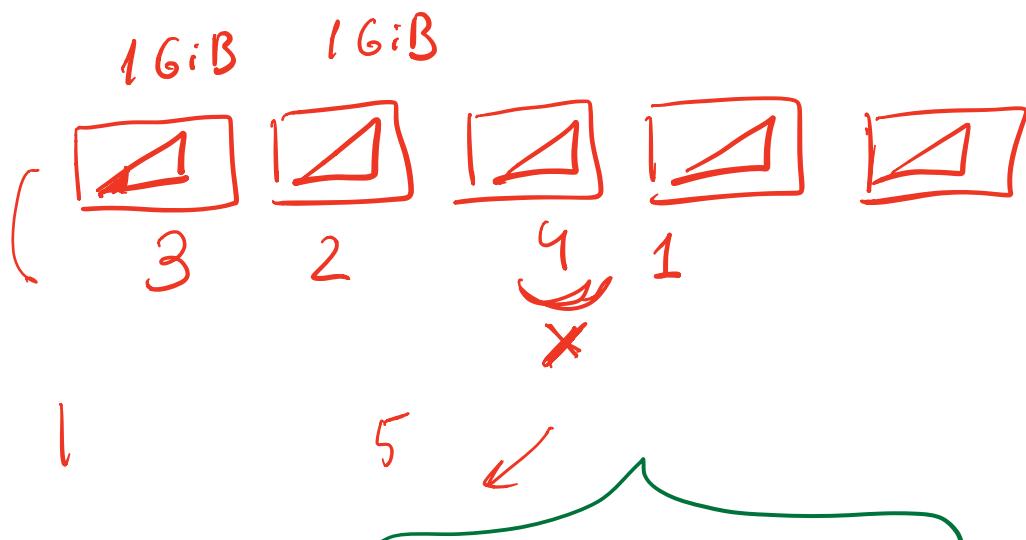
^

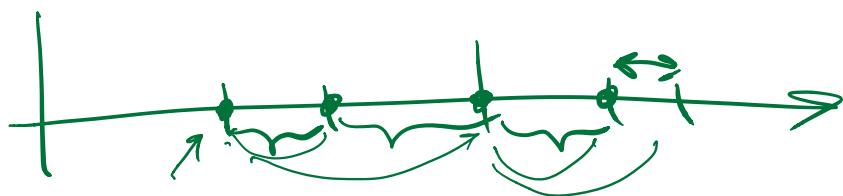
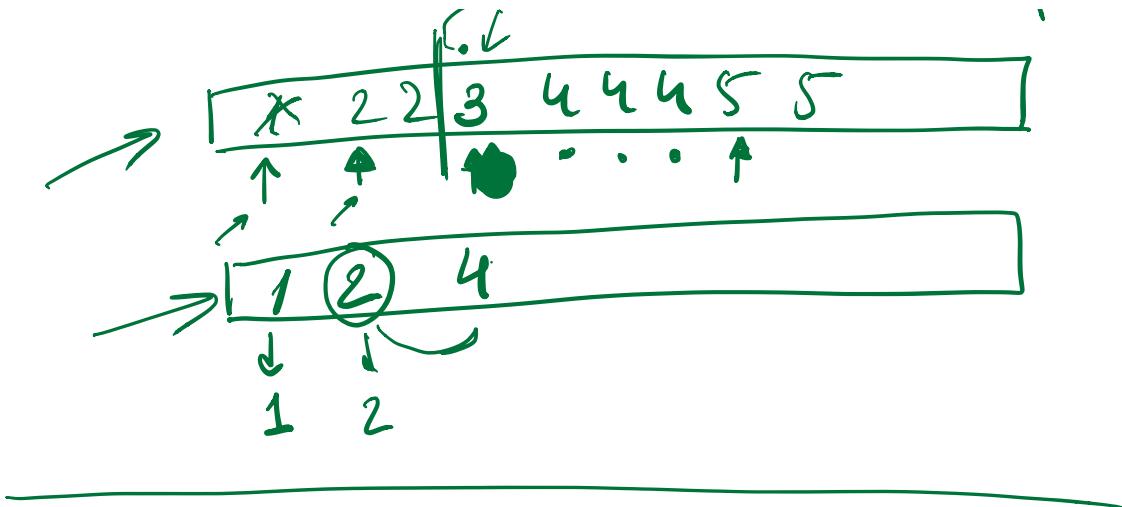


$$\begin{aligned}
 n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} \dots &= \\
 = n \left(1 + \frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{2^k} \right) &= \\
 = n \left(2 - \frac{1}{2^k} \right) &\leq 2n
 \end{aligned}$$

$O(n \log n)$

TiB





$f(t)$ - можно ли расставить
коробки на расстоянии

