

# ОСНОВНЫЕ ОЦЕНКИ СЛОЖНОСТИ АЛГОРИТМОВ

модель алгоритма сортировки •  
алгоритм MERGE SORT

Нестеров Р.А., PhD, доцент  
департамента программной инженерии

02

июль 2024						
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3

# план лекции

**01**

**разделяй-и-  
властвуй:** алгоритм  
MERGE SORT

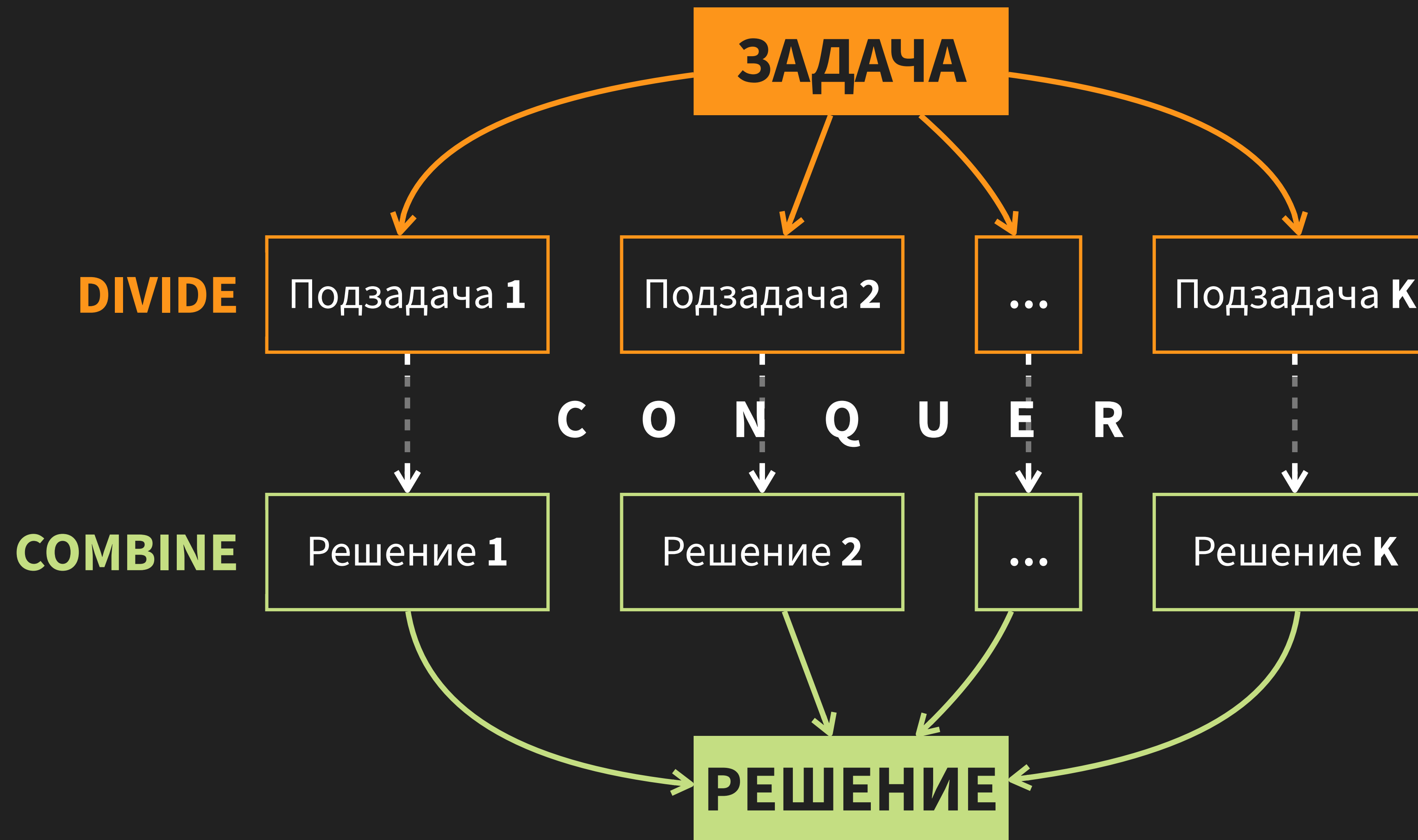
**02**

**общая модель**  
алгоритма  
сортировки

**03**

**оптимальная**  
сортировка на  
сравнениях

# разделяй–и–властвуй • MERGE SORT



# разделяй–и–властвуй • MERGE SORT

DIVIDE

разбить входную последовательность на две по [примерно] половине элементов в каждой

CONQUER

выполнить рекурсивную сортировку полученных подпоследовательностей

COMBINE

объединить две отсортированные подпоследовательности

5	2	4	7	1	3	2	6
---	---	---	---	---	---	---	---

5	2	4	7	1	3	2	6
---	---	---	---	---	---	---	---

5	2	4	7
---	---	---	---

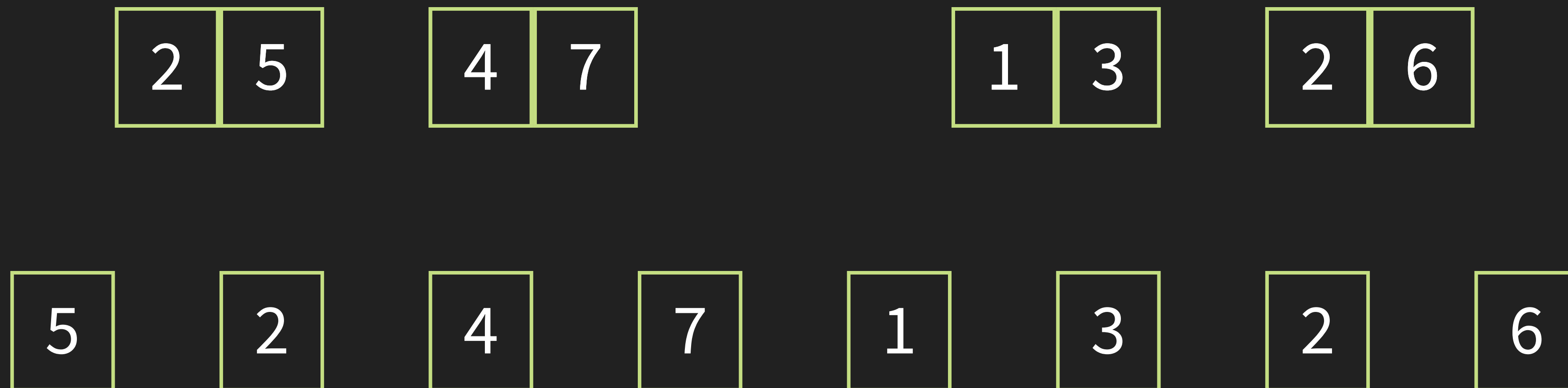
1	3	2	6
---	---	---	---











2 4 5 7

1 2 3 6

2 5

4 7

1 3

2 6

5

2

4

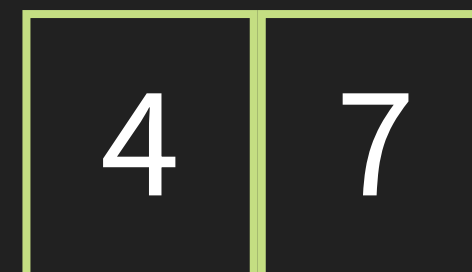
7

1

3

2

6



```

1 void mergeSort(std::vector<int> &arr,
2               size_t left,
3               size_t right) {
4
5     if (left < right) {
6         size_t mid = left + (right - left) / 2;
7
8         // DIVIDE
9         mergeSort(arr, left, mid);
10        mergeSort(arr, mid + 1, right);
11
12        // CONQUER & COMBINE
13        merge(arr, left, mid, right);
14    }
15 }

```

функция **merge** может быть реализована двумя способами

- ➔ с выделением дополнительной памяти
- ➔ без использования дополнительных массивов [in-place]

затраты

$T(n) = \dots$

$O(1)$

$2 \cdot T(n/2)$

$O(n)$

⊕ MERGE\_SORT.cpp

```
1 void mergeSort(std::vector<int> &arr,
2               size_t left,
3               size_t right) {
4
5     if (left < right) {
6         size_t mid = left + (right - left) / 2;
7
8         // DIVIDE
9         mergeSort(arr, left, mid);
10        mergeSort(arr, mid + 1, right);
11
12        // CONQUER & COMBINE
13        merge(arr, left, mid, right);
14    }
15 }
```

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

## ЗАДАЧА

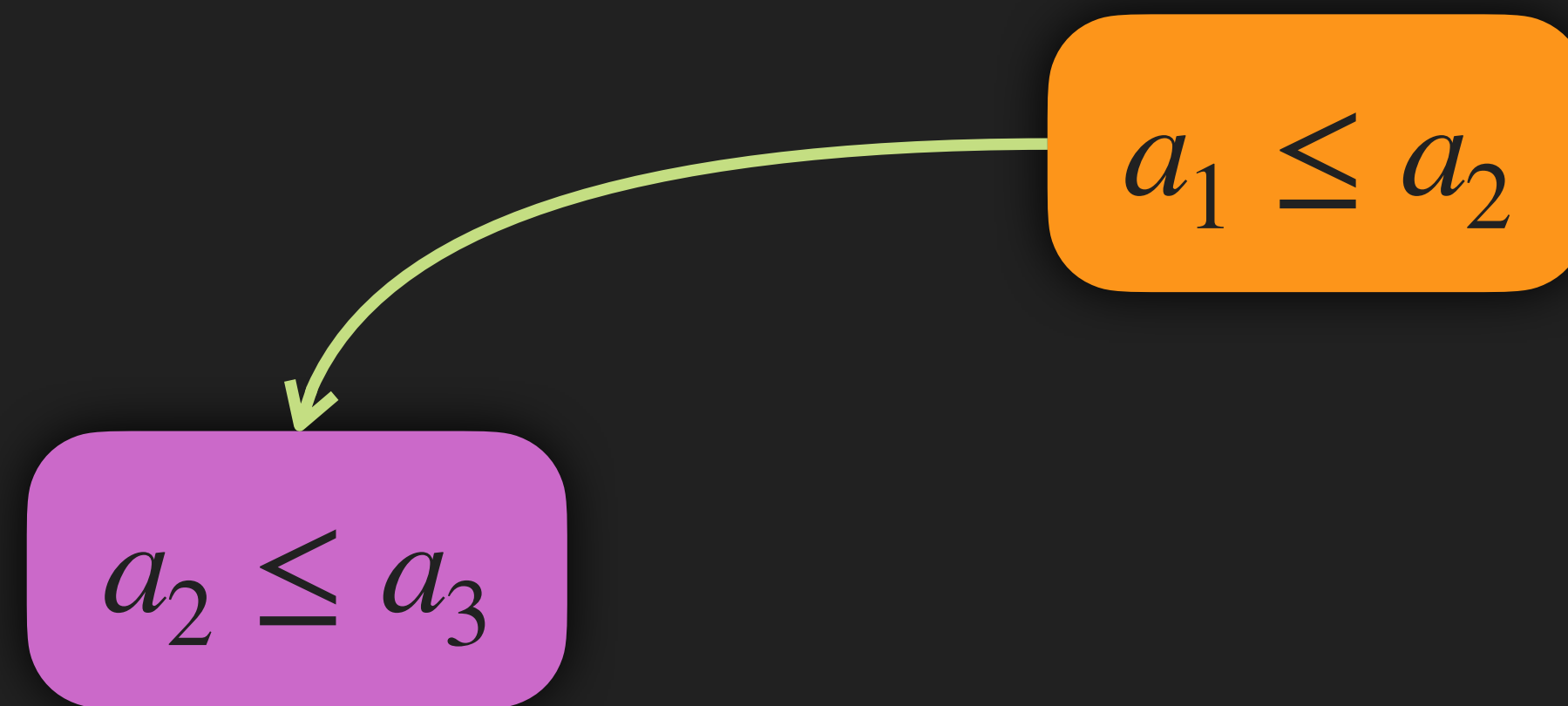
Найти  $g(n)$ , для которой  $T(n) = 2 \cdot T(n/2) + O(n) = O(g(n))$ .

модель сортировки для  $A = \{a_1, a_2, a_3\}$

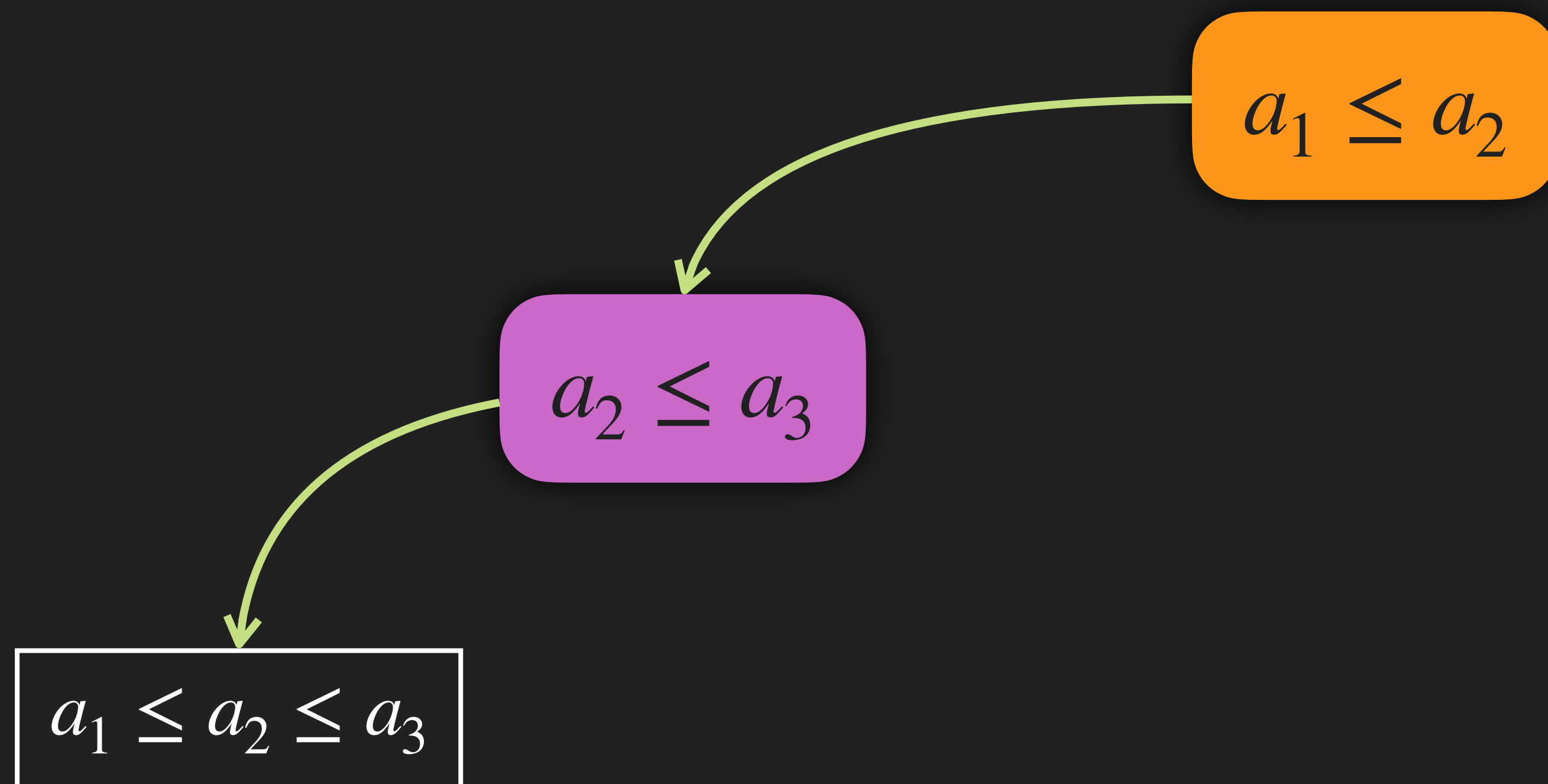
$$a_1 \leq a_2$$



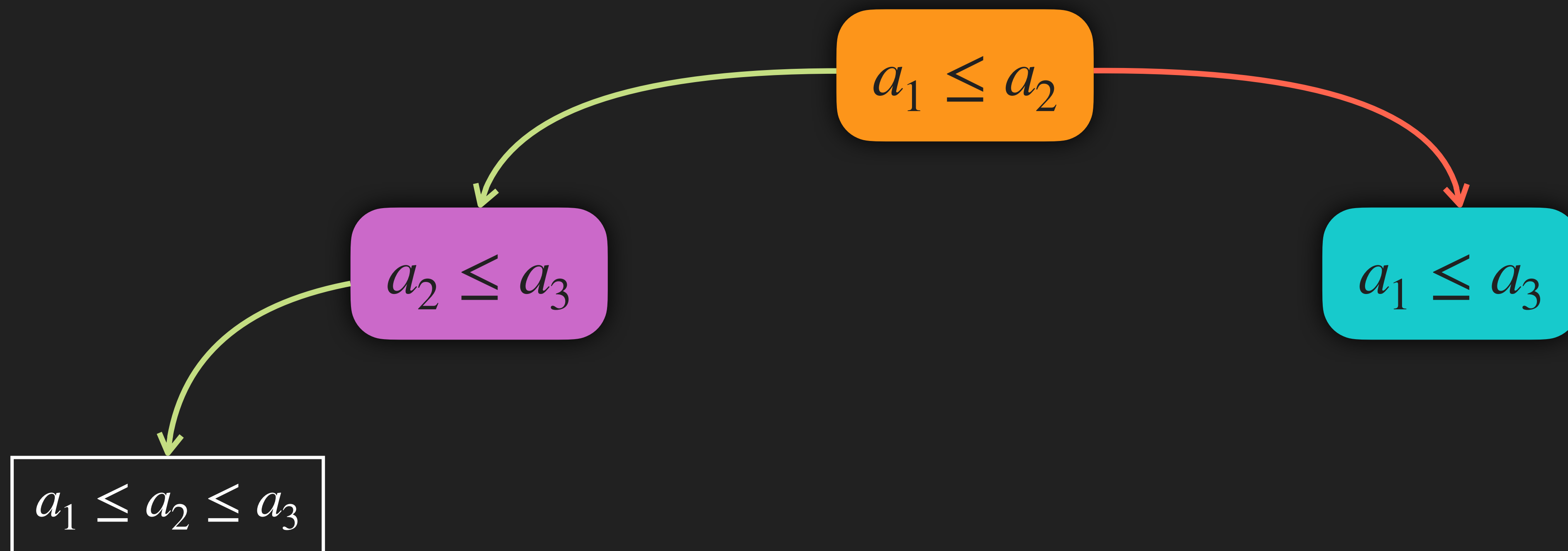
# модель сортировки для $A = \{a_1, a_2, a_3\}$



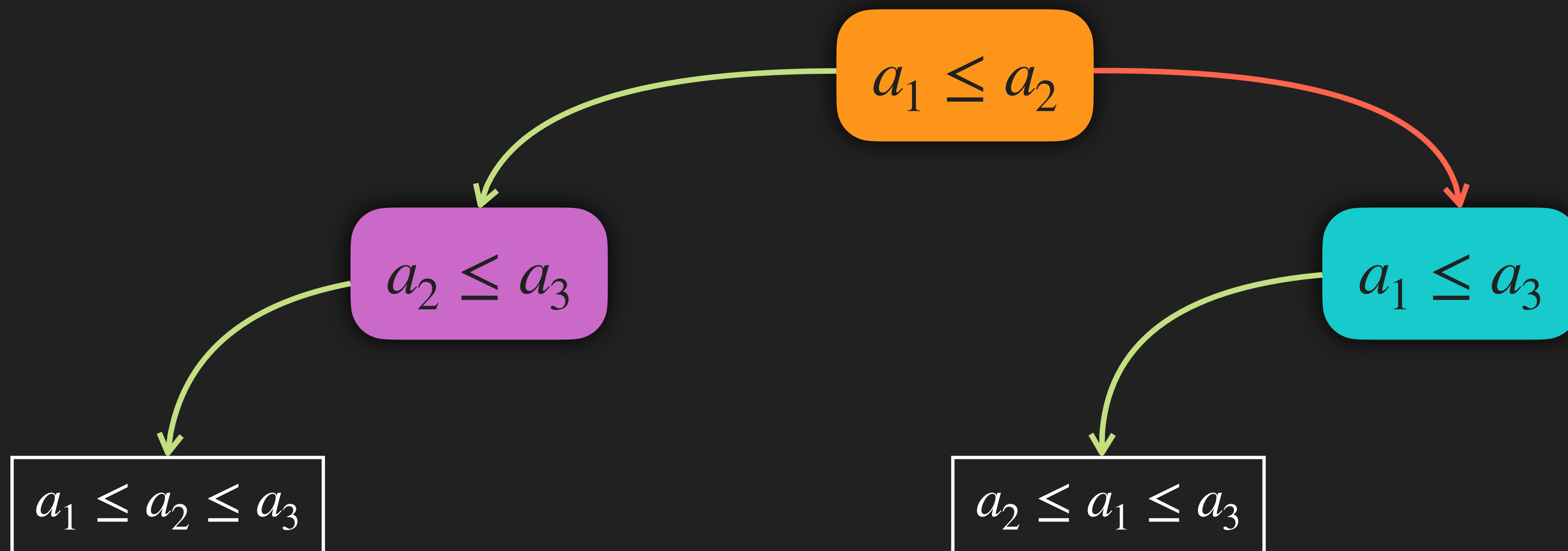
# модель сортировки для $A = \{a_1, a_2, a_3\}$



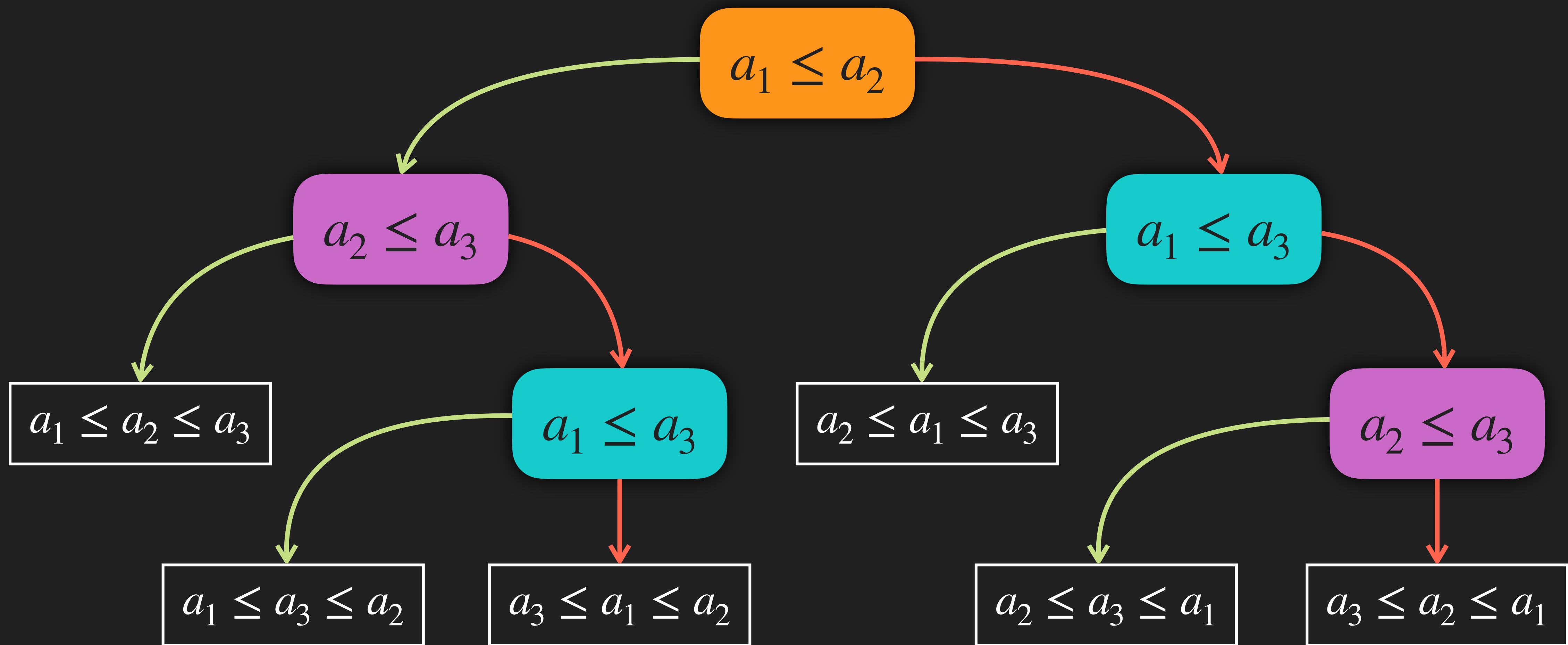
# модель сортировки для $A = \{a_1, a_2, a_3\}$

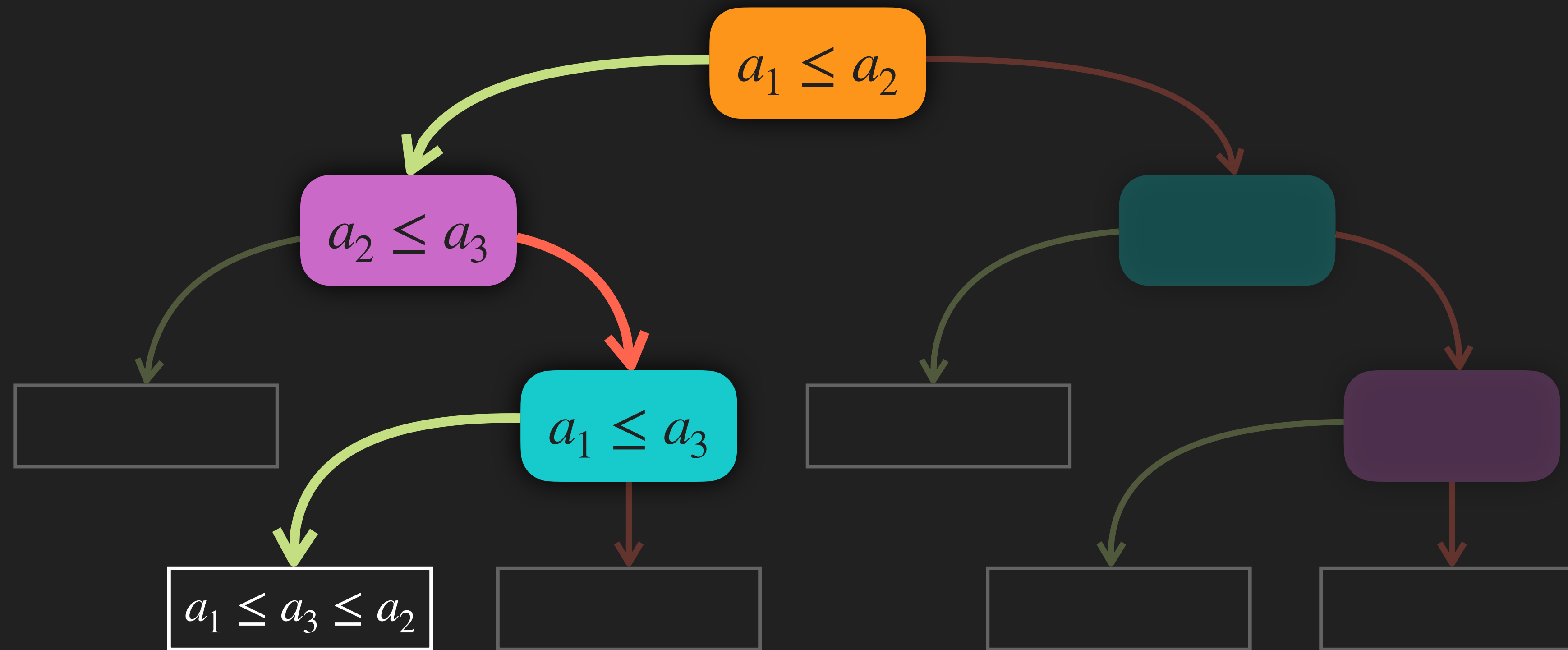


# модель сортировки для $A = \{a_1, a_2, a_3\}$

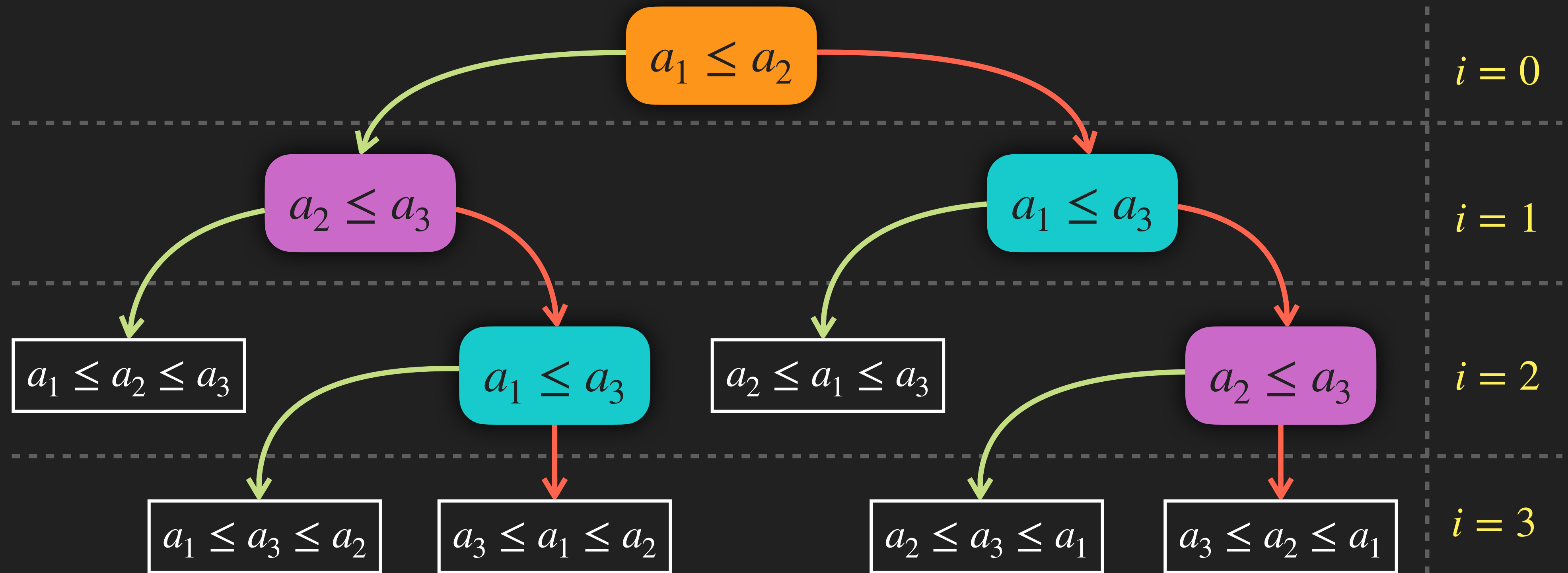


# модель сортировки для $A = \{a_1, a_2, a_3\}$





выполнение сортировки —  
путь из корня до листа  
в дереве решений



выполнение сортировки —  
 путь из корня до листа  
 в дереве решений

**НИЖНЯЯ** граница высоты  
 дерева — минимальное  
 количество сравнений

## ТЕОРЕМА

Высота дерева решений для сортировки —  $\Omega(n \log n)$ .



## ТЕОРЕМА

Высота дерева решений для сортировки —  $\Omega(n \log n)$ .

## ДОК-ВО

Оценим  $h$  — высоту дерева решений алгоритма сортировки массива из  $n$  элементов.

(1) Число листьев дерева решений составляет  $n!$  — количество перестановок.

(2) Бинарное дерево высоты  $h$  имеет не более  $2^h$  листьев. Тогда:

$$2^h \geq n! \Rightarrow h \geq \log_2(n!).$$

(3) Воспользуемся формулой Стирлинга  $n! > (n/e)^n$  и получим:


$$h \geq \log_2(n!) \Rightarrow h \geq \log_2(n/e)^n \Leftrightarrow h \geq n \log_2 n - n \log_2 e.$$

Следовательно,  $h = \Omega(n \log n)$  для  $c = 0.5$  и достаточно больших  $n$ .

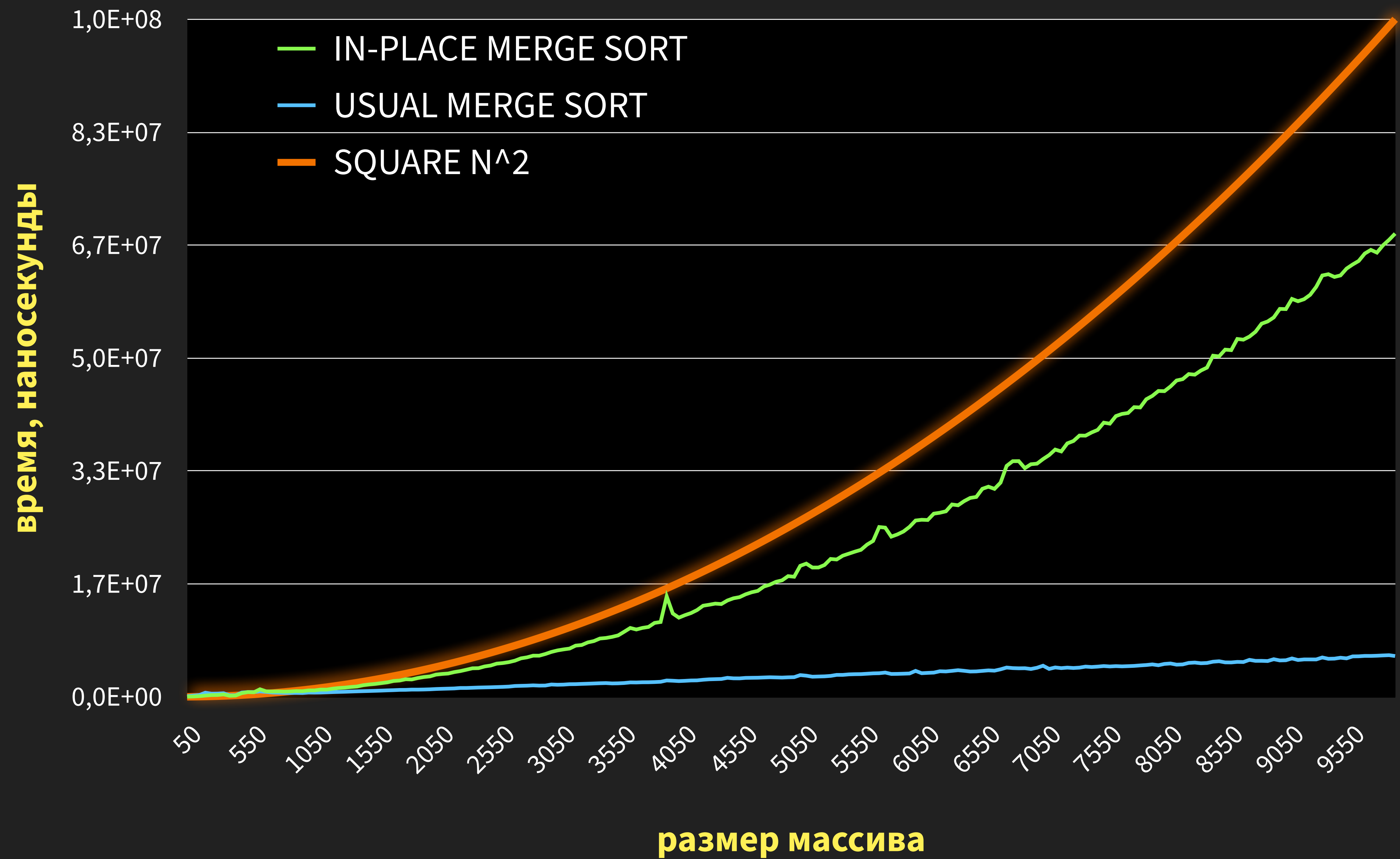


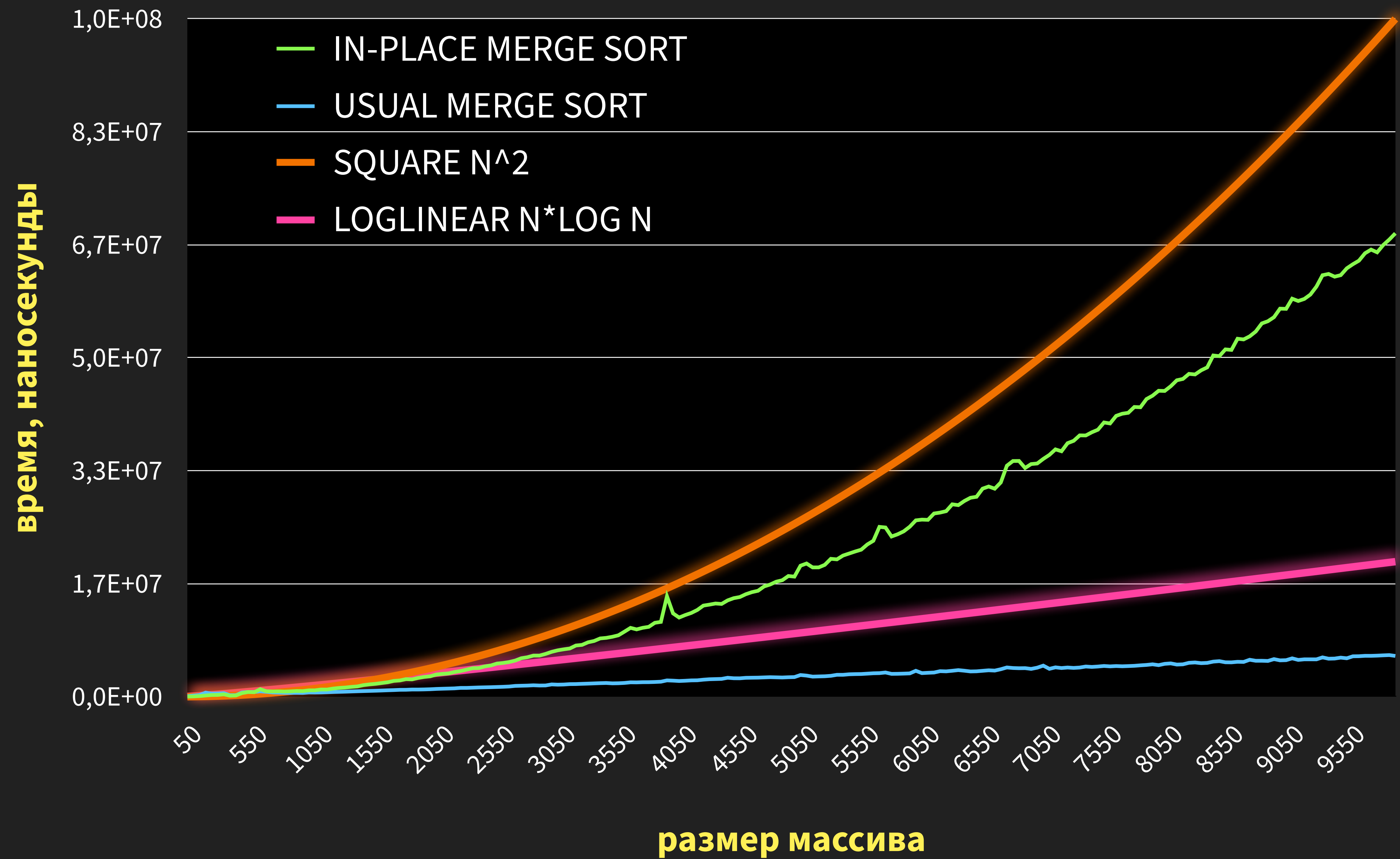
любой алгоритм сортировки,  
использующий сравнения,  
потребует  $\Omega(n \log n)$  операций

# MERGE SORT — оптимальный



## алгоритм сортировки на сравнениях







хранение упорядоченных  
данных в массиве — плохая  
идея...