

Основатель социальной сети Facebook Марк Цукерберг, совладелец Tesla Motors и SpaceX Элон Маск и популярный актер и венчурный инвестор Эштон Кутчер **вложили** \$40 млн в компанию **Vicarious**, которая изучает структуру коры головного мозга и пытается воссоздать ее в компьютерном коде.

Задача

Покрытие точек отрезками

Вход: множество n точек на прямой $x_1, \dots, x_n \in \mathbb{R}$.

Выход: минимальное количество отрезков единичной длины, которыми можно покрыть все точки.

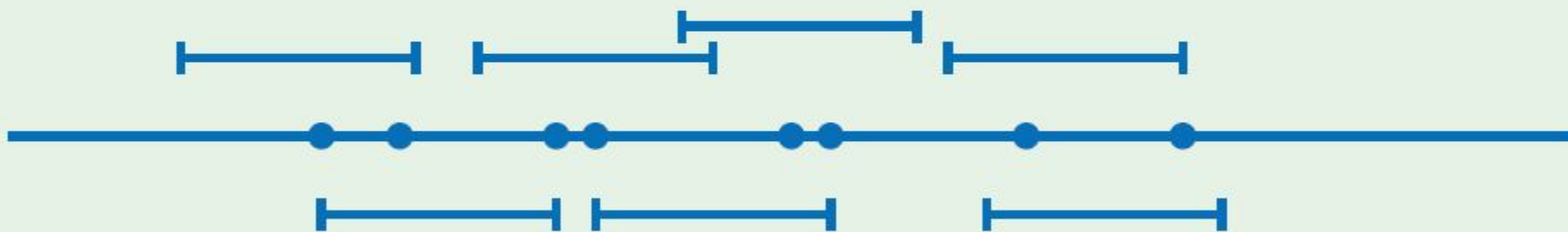
Пример



Пример

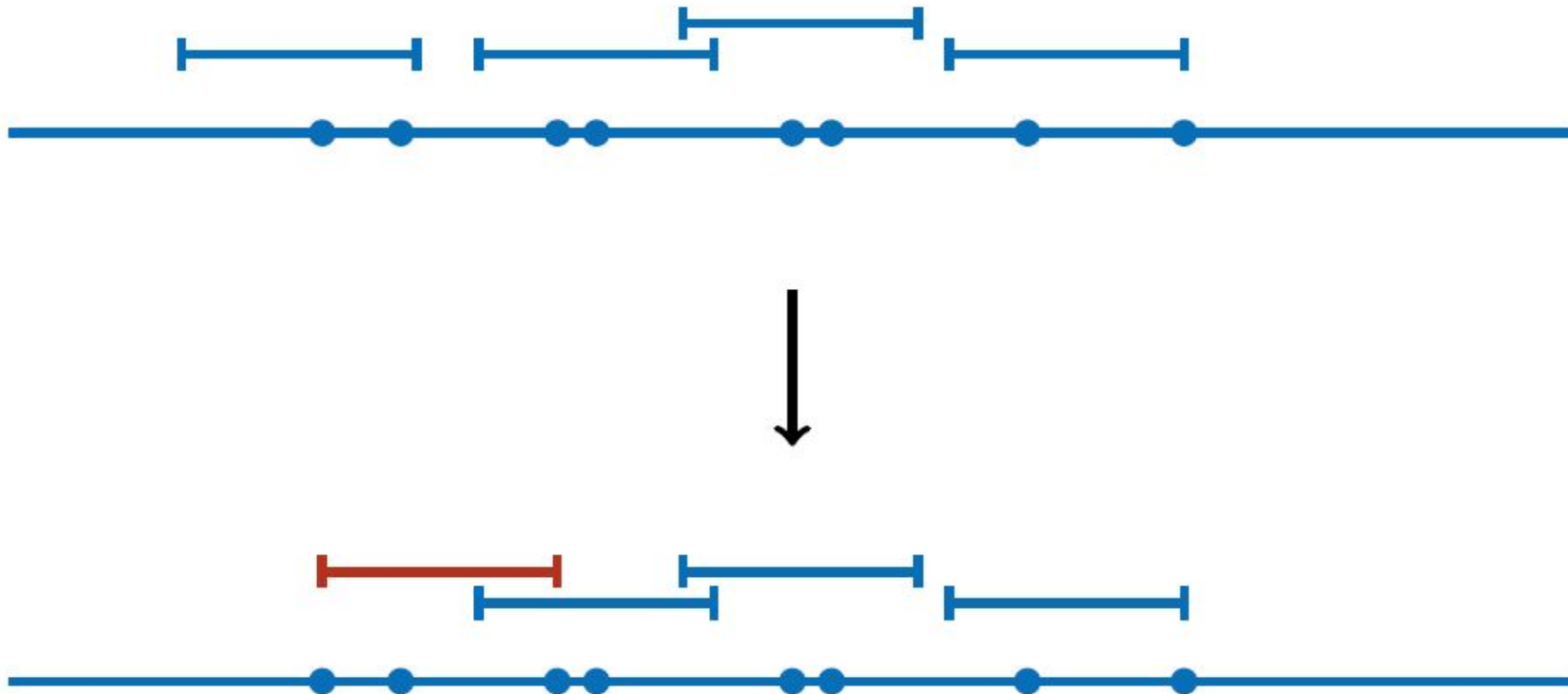


Пример



Надёжный шаг

Существует оптимальное покрытие, в котором самая левая точка покрыта левым концом отрезка.



Добавим в решение отрезок, левый край которого совпадает с крайней левой точкой.

Алгоритм

Функция POINTSCOVER(x_1, \dots, x_n)

$S \leftarrow \{x_1, \dots, x_n\}$

пока S не пусто:

$x_m \leftarrow$ минимальная точка S

 добавить к решению отрезок $[l, r] = [x_m, x_m + 1]$

 выкинуть из S точки, покрытые отрезком $[l, r]$

вернуть построенное решение

Алгоритм

Функция POINTSCOVER(x_1, \dots, x_n)

$S \leftarrow \{x_1, \dots, x_n\}$

$O(n)$ пока S не пусто:

$x_m \leftarrow$ минимальная точка S $O(n)$

добавить к решению отрезок $[l, r] = [x_m, x_m + 1]$ $O(1)$

выкинуть из S точки, покрытые отрезком $[l, r]$ $O(n)$

вернуть построенное решение

Время работы: $O(n^2)$

Улучшенный алгоритм

Функция POINTSCOVER(x_1, \dots, x_n)

$x_1, \dots, x_n \leftarrow \text{SORT}(x_1, \dots, x_n)$

$i \leftarrow 1$

пока $i \leq n$:

 добавить к решению отрезок $[\ell, r] = [x_i, x_i + 1]$

$i \leftarrow i + 1$

 пока $i \leq n$ и $x_i \leq r$:

$i \leftarrow i + 1$

вернуть построенное решение

Время работы: $T(\text{SORT}) + O(n) = O(n \log n)$.

Пример

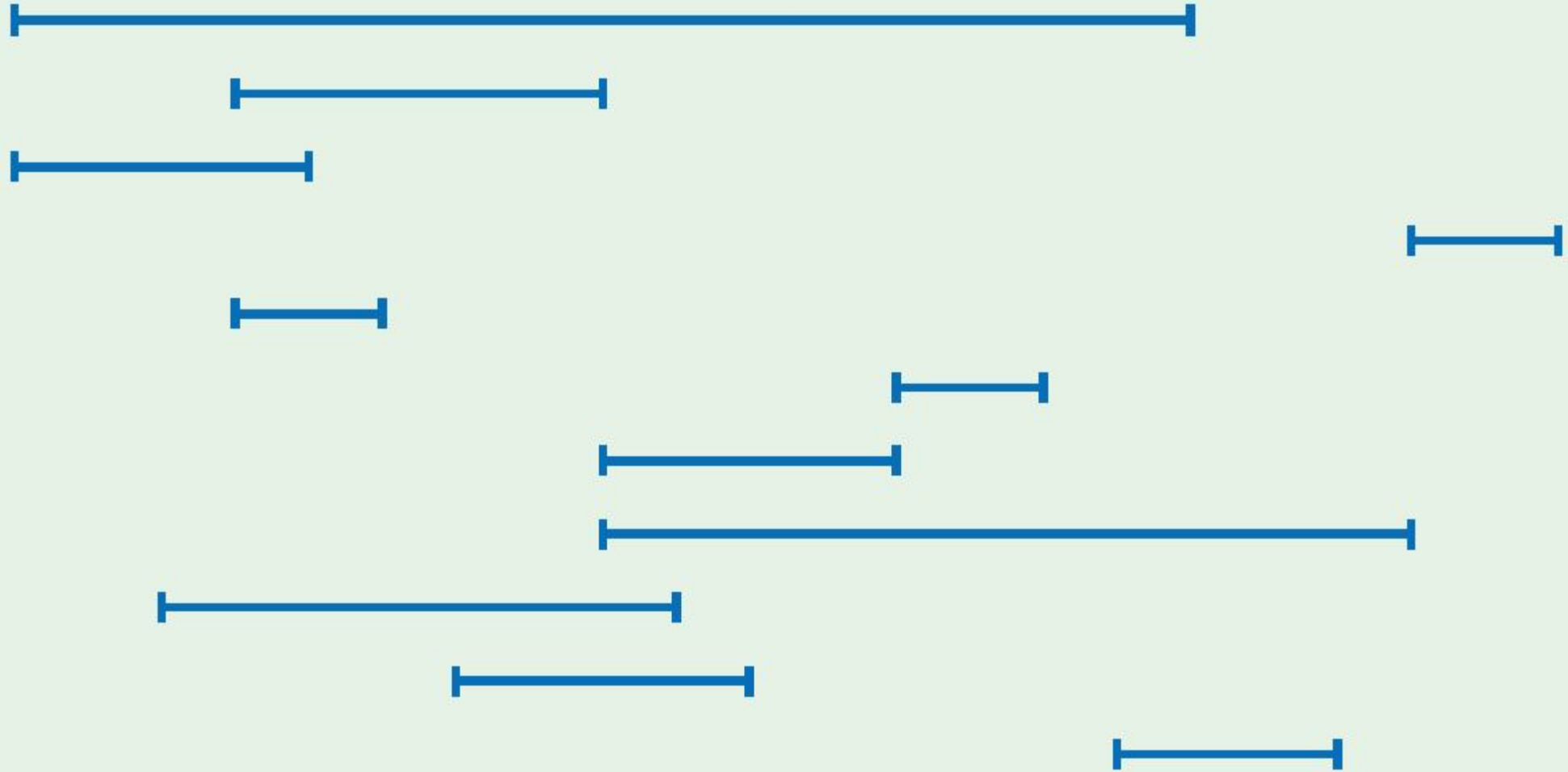


Задача о выборе заявок

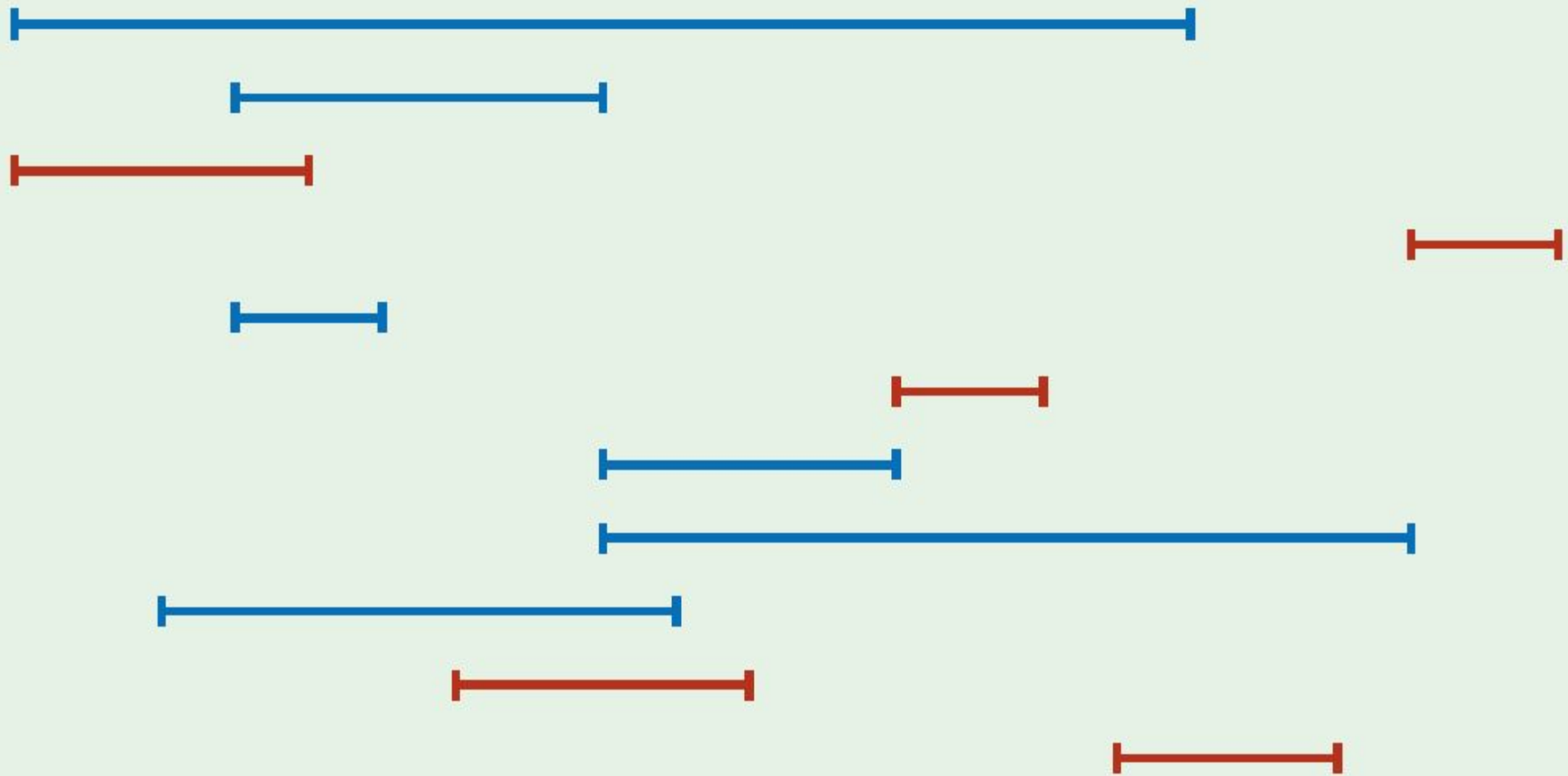
Вход: множество n отрезков на прямой.

Выход: максимальное количество попарно не пересекающихся отрезков.

Пример

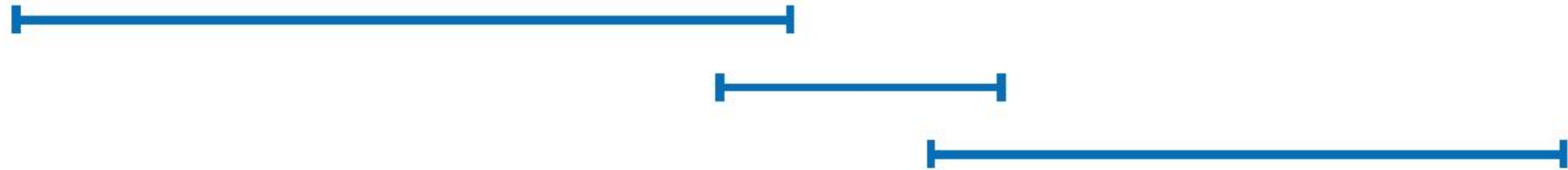


Пример



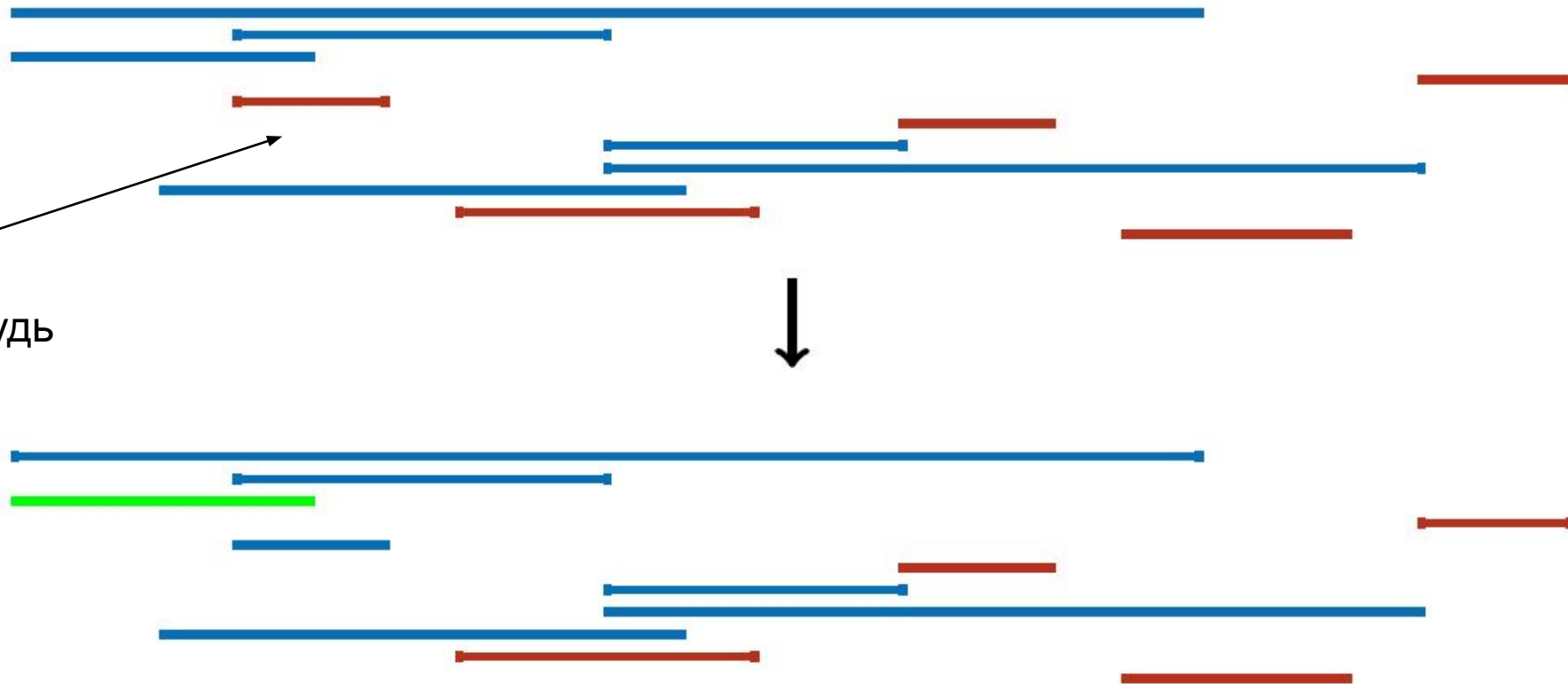
Замечание

Выбирая в первую очередь более короткие отрезки, можно получить неоптимальное решение.



Надёжный шаг

Существует оптимальное решение, содержащее отрезок, правый конец которого минимален.



Хоть какое-нибудь
решение

Можно сразу добавить в решение отрезок, правый конец которого минимален.

Алгоритм

Функция ACTSEL($l_1, r_1, \dots, l_n, r_n$)

$S \leftarrow \{[l_1, r_1], \dots, [l_n, r_n]\}$

пока S не пусто:

$[l_m, r_m] \leftarrow$ отрезок из S с мин. правым концом

 добавить $[l_m, r_m]$ к решению

 выкинуть из S отрезки, пересекающиеся с $[l_m, r_m]$

вернуть построенное решение

Время работы: $O(n^2)$.

Улучшенный алгоритм

Функция $ACTSEL(\ell_1, r_1, \dots, \ell_n, r_n)$

отсортировать n отрезков по правым концам
для всех отрезков в полученном порядке:

 если текущий отрезок не пересекает

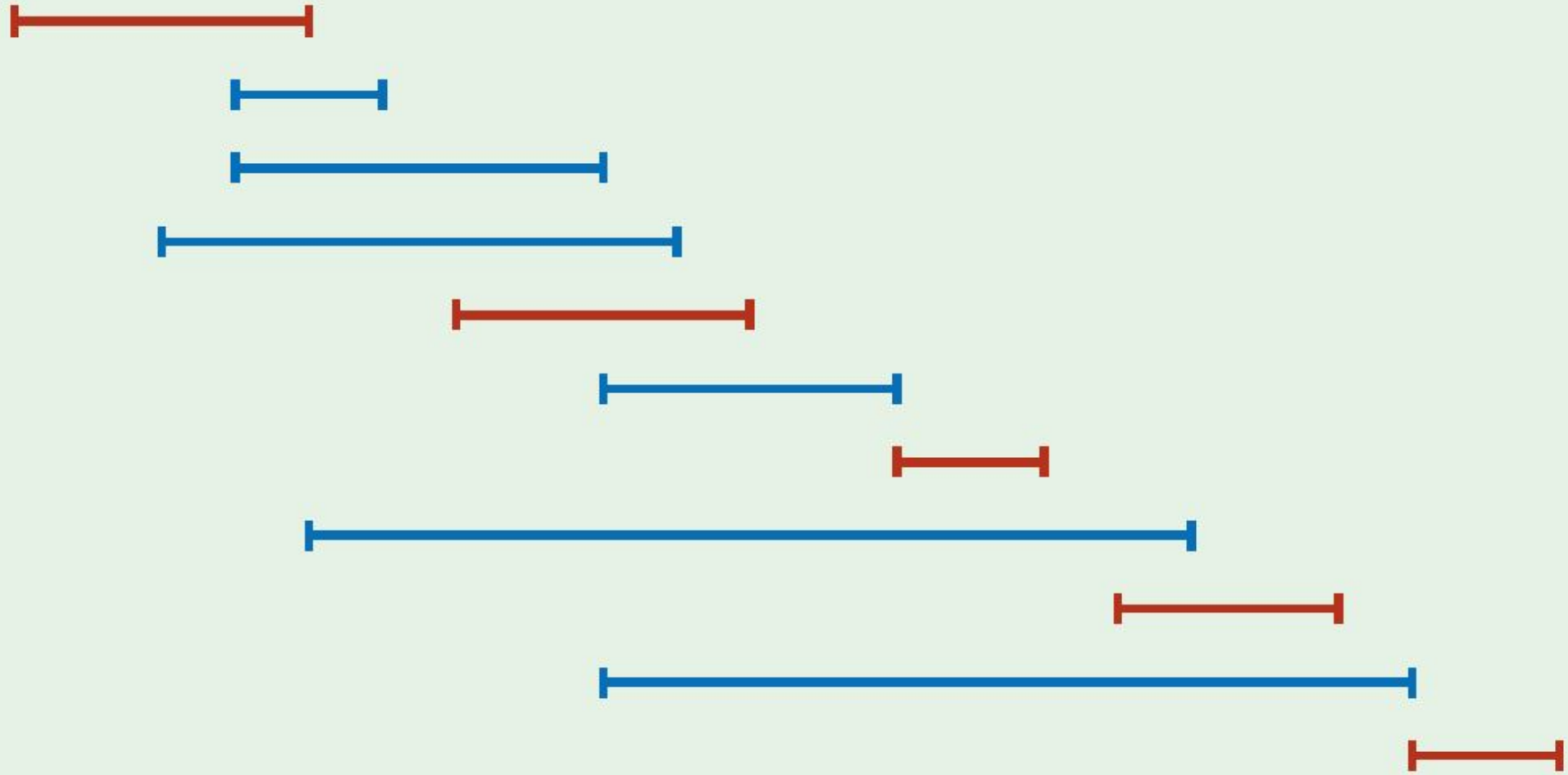
 последний добавленный:

 взять его в решение

вернуть построенное решение

Время работы: $T(\text{SORT}) + O(n) = O(n \log n)$.

Пример

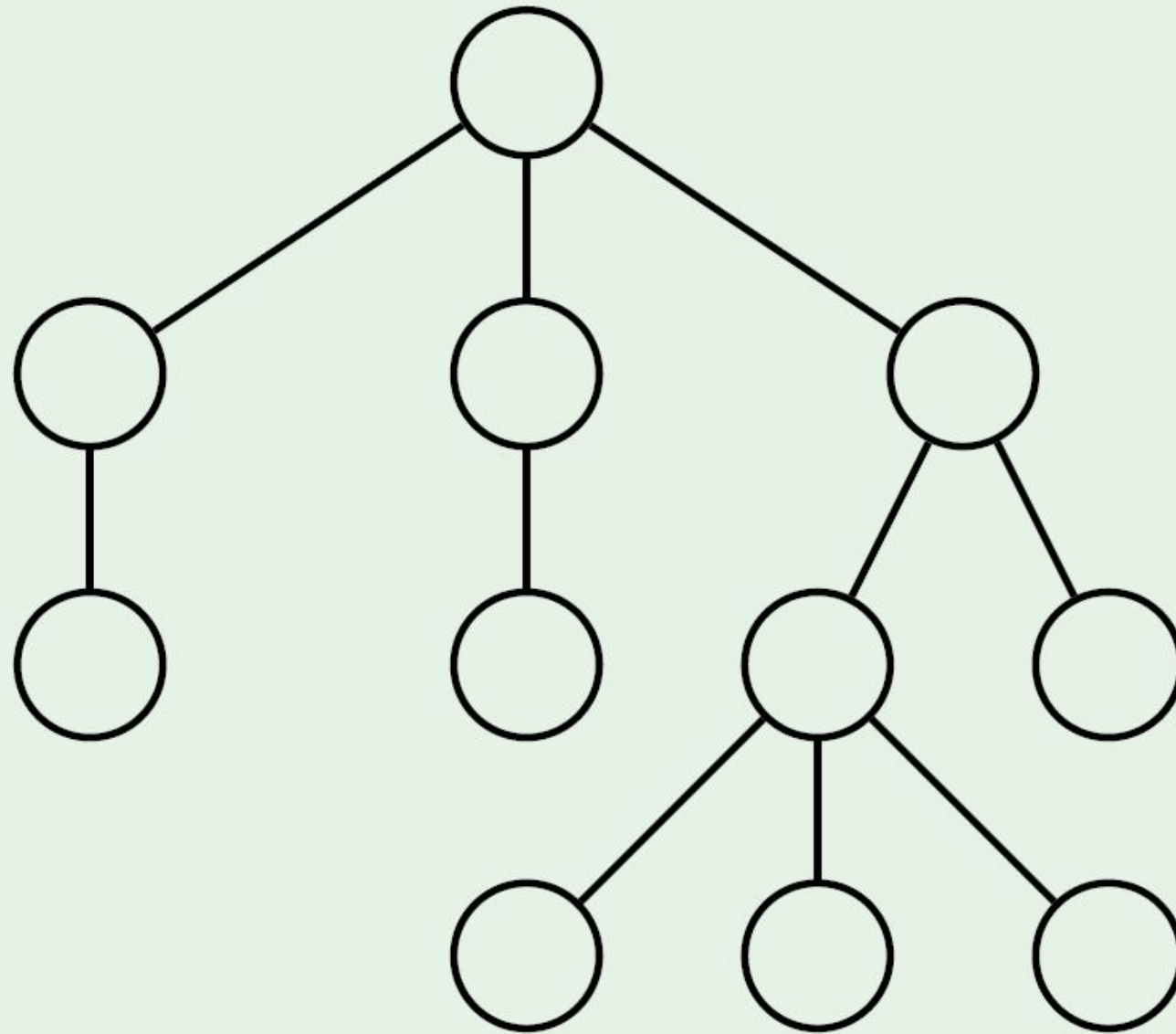


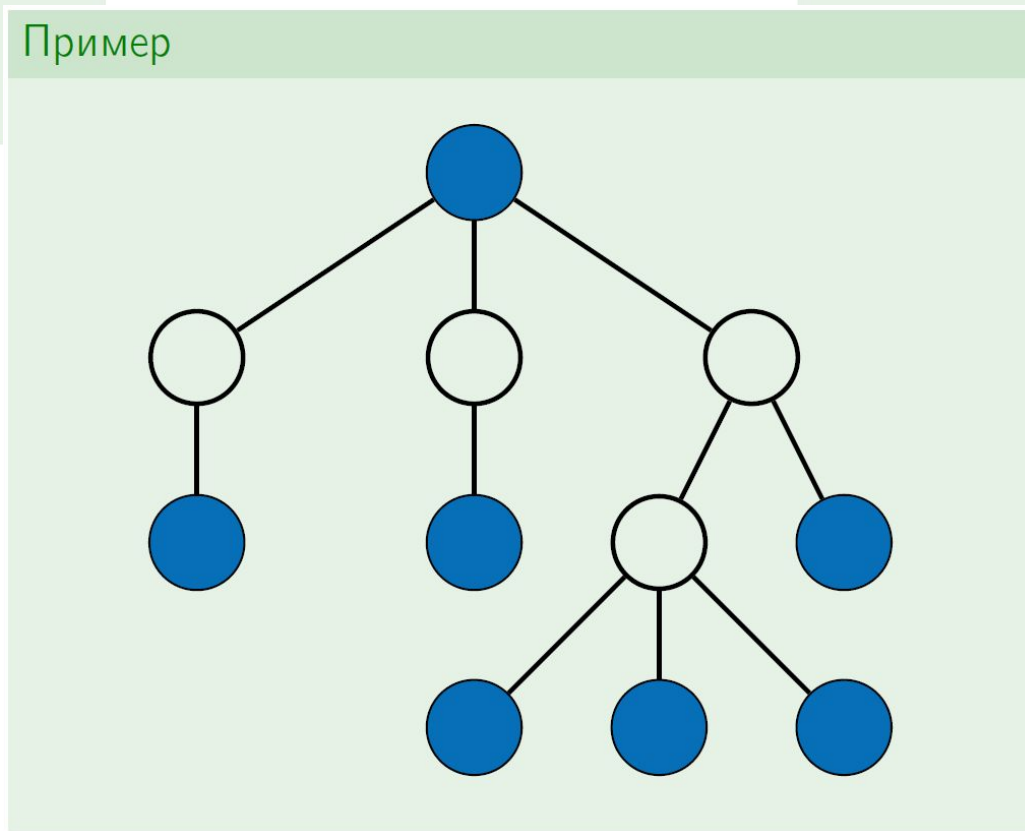
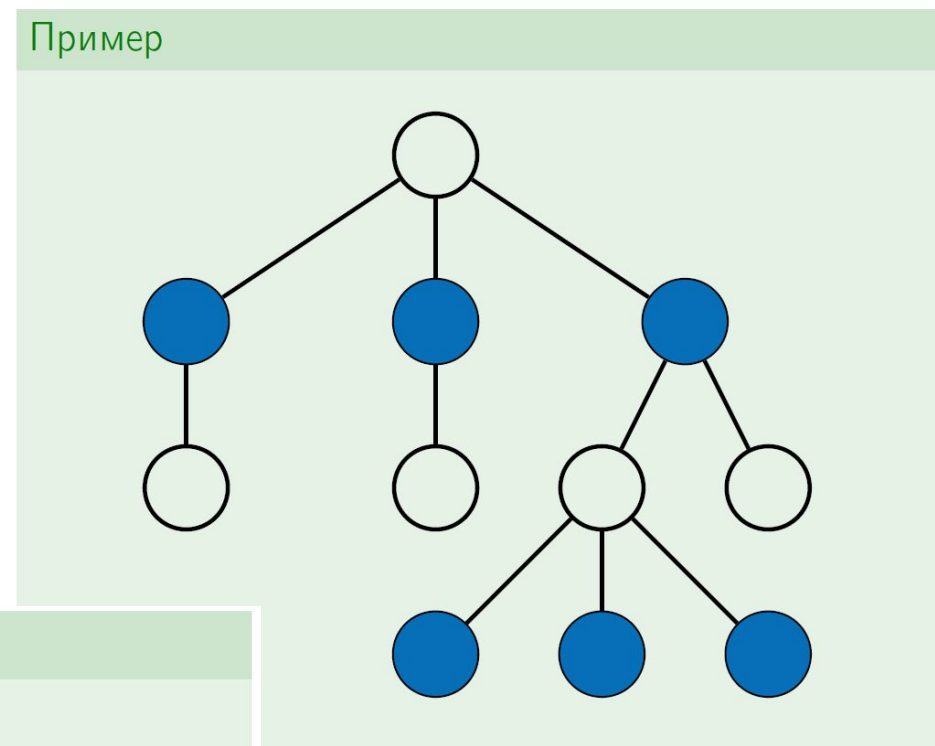
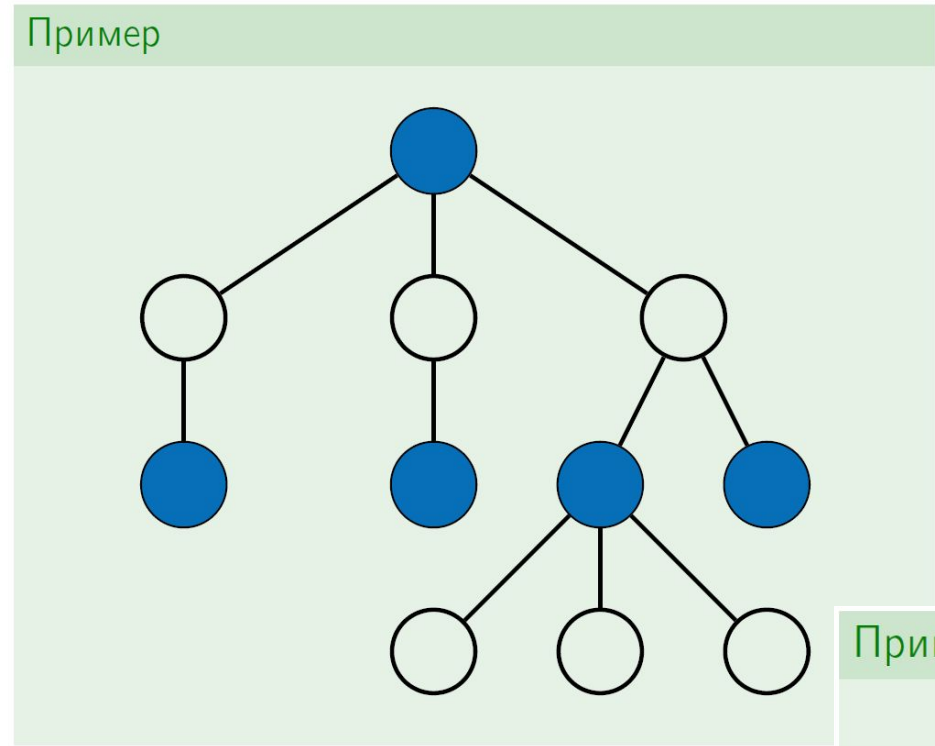
Планирование вечеринки в компании

Вход: дерево.

Выход: независимое множество (множество не соединённых друг с другом вершин)
максимального размера.

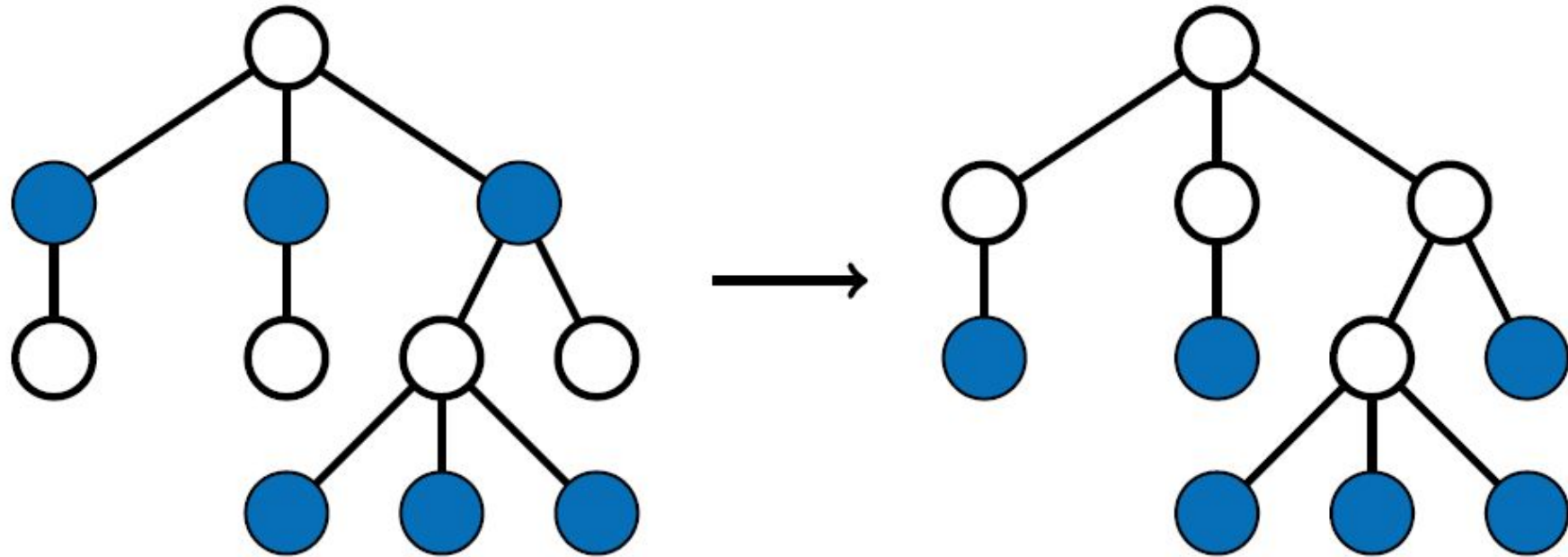
Пример





Надёжный шаг

Существует оптимальное решение, содержащее каждый лист дерева.



Можно взять в решение все листья.

Алгоритм

Функция $\text{MAXINDEPENDENTSET}(T)$

пока T не пусто:

 взять в решение все листья

 выкинуть их и их родителей из T

вернуть построенное решение

Время работы: $O(|T|)$.

Непрерывный рюкзак

Вход: веса w_1, \dots, w_n и стоимости c_1, \dots, c_n данных n предметов; вместимость рюкзака W .

Выход: максимальная стоимость частей предметов суммарного веса не более W .

Пример

20 руб.



18 руб.



14 руб.



рюкзак

Пример

20 руб.

18 руб.

14 руб.

4

3

2

20

18

4

3

рюкзак

всего: 38 руб.

Пример

20 руб.

18 руб.

14 руб.

4

3

2

20

14

18/3

4

2

1

рюкзак

всего: 40 руб.

Пример

20 руб.

18 руб.

14 руб.

4

3

2

14

18

20/2

2

3

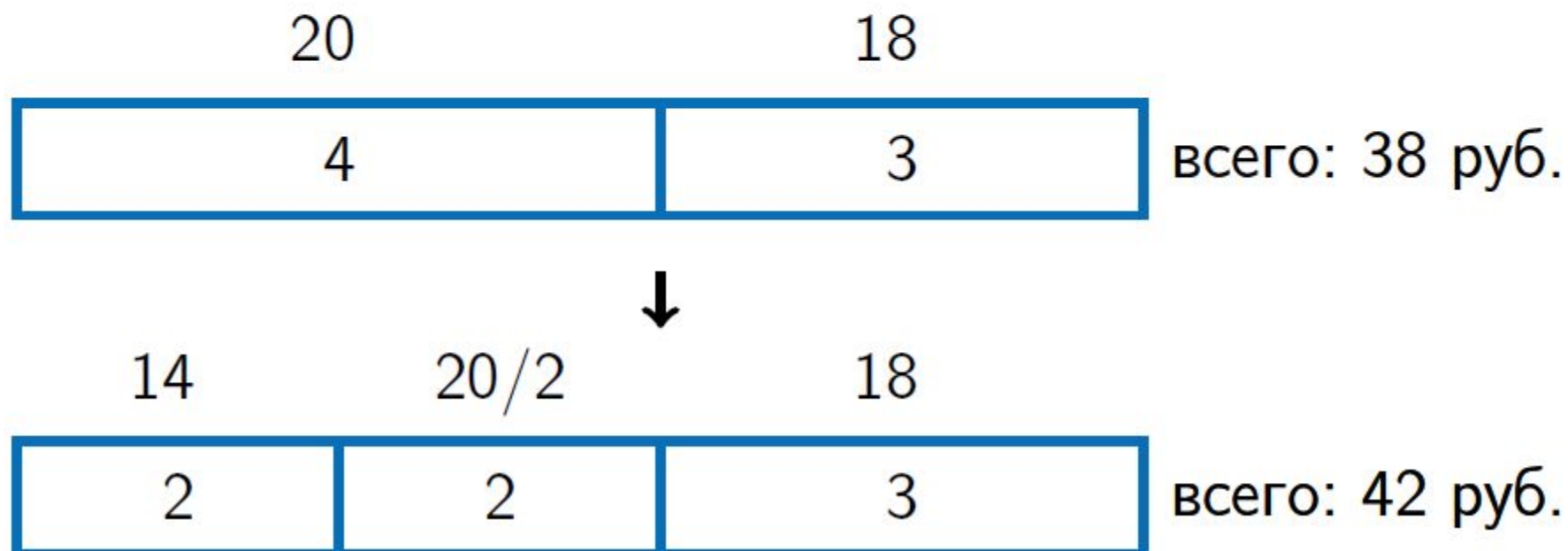
2

рюкзак

всего: 42 руб.

Надёжный шаг

Существует оптимальное решение, содержащее максимально возможную часть предмета, стоимость которого за килограмм максимальна.



Алгоритм

Функция $\text{KNAPSACK}(w_1, c_1, \dots, w_n, c_n)$

отсортировать предметы по убыванию c/w
для всех предметов в полученном порядке:
 взять по максимуму текущего предмета
вернуть построенное решение

Время работы: $T(\text{SORT}) + O(n) = O(n \log n)$.

Основные идеи

Надёжный шаг. Существует оптимальное решение, согласованное с локальным жадным шагом.

Оптимальность подзадач. Задача, остающаяся после жадного шага, имеет тот же тип.

Задача на программирование: покрыть отрезки точками

По данным n отрезкам необходимо найти множество точек минимального размера, для которого каждый из отрезков содержит хотя бы одну из точек.

В первой строке дано число $1 \leq n \leq 100$ отрезков. Каждая из последующих n строк содержит по два числа $0 \leq l \leq r \leq 10^9$, задающих начало и конец отрезка. Выведите оптимальное число m точек и сами m точек. Если таких множеств точек несколько, выведите любое из них.

Sample Input 1:

```
3
1 3
2 5
3 6
```

Sample Output 1:

```
1
3
```

Задача на программирование: непрерывный рюкзак

Первая строка содержит количество предметов $1 \leq n \leq 10^3$ и вместимость рюкзака $0 \leq W \leq 2 \cdot 10^6$. Каждая из следующих n строк задаёт стоимость $0 \leq c_i \leq 2 \cdot 10^6$ и объём $0 < w_i \leq 2 \cdot 10^6$ предмета (n, W, c_i, w_i – целые числа). Выведите максимальную стоимость частей предметов (от каждого предмета можно отделить любую часть, стоимость и объём при этом пропорционально уменьшатся), помещающихся в данный рюкзак, с точностью не менее трёх знаков после запятой.

Sample Input:

```
3 50
60 20
100 50
120 30
```

Sample Output:

```
180.000
```

Задача на программирование: различные слагаемые

По данному числу $1 \leq n \leq 10^9$ найдите максимальное число k , для которого n можно представить как сумму k различных натуральных слагаемых. Выведите в первой строке число k , во второй — k слагаемых.

Sample Input 1:

4

Sample Output 1:

2

1 3

Sample Input 2:

6

Sample Output 2:

3

1 2 3