

## Мини-задача #32 (1 балл)

В каждом элементе массива длина максимального прыжка.  
Начинаем с нулевого элемента, возможно ли пропрыгать от  
до его конца массива?

<https://leetcode.com/problems/jump-game>

## Мини-задача #33 (1 балл)

Вы брокер, который может покупать или продавать акции  
(у вас на руках только одна позиция). Дан массив цен на  
акции по дням, найти максимальную прибыль.

<https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/>

# Алгоритмы и структуры данных

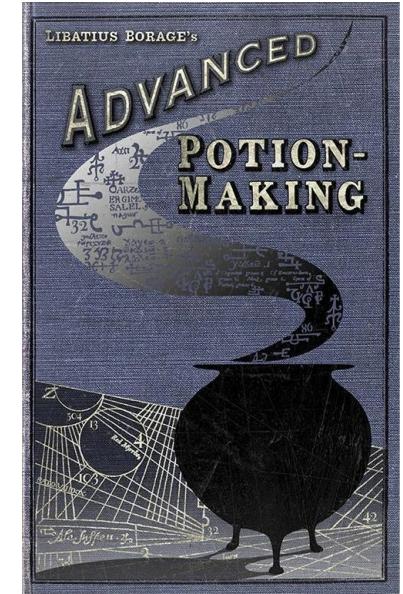
Часть 2: Организационные вопросы



# Что будем изучать?

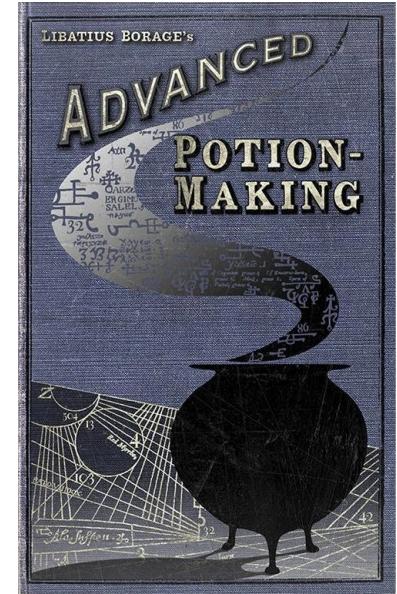
# Что будем изучать?

1. Жадные алгоритмы
2. Динамическое программирование



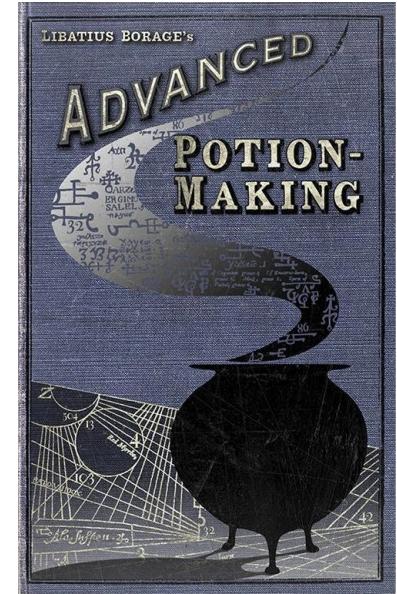
# Что будем изучать?

1. Жадные алгоритмы
2. Динамическое программирование
3. Необычные структуры данных  
(новые виды деревьев, пирамид и не только)



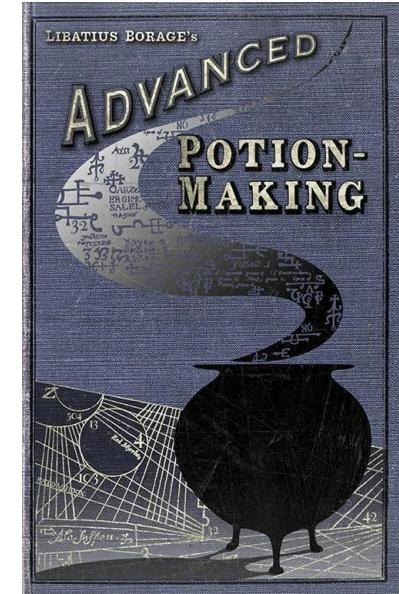
# Что будем изучать?

1. Жадные алгоритмы
2. Динамическое программирование
3. Необычные структуры данных  
(новые виды деревьев, пирамид и не только)
4. ... and beyond



# Что будем изучать?

1. Жадные алгоритмы
2. Динамическое программирование
3. Необычные структуры данных  
(новые виды деревьев, пирамид и не только)
4. ... and beyond



# Зачем будем изучать?

# Зачем будем изучать?

- База для всего остального Computer Science (в прикладном смысле)
- Огромное количество практических применений
- Алгоритмическая секция на собеседованиях
- Это весело!



Because Losing Is Fun



2. Add Two Numbers

Medium    4668    1171    Favorite    Share

Input: (2 → 4 → 3) + (5 → 6 → 4)

Output: 7 → 0 → 8

Explanation: 342 + 465 = 807.

# Как будем изучать?

# Как будем изучать?

- ~~16 лекций + 16 семинаров~~ 16 пар

# Как будем изучать?

- ~~16 лекций + 16 семинаров~~ 16 пар
- Практика = мильтки + "большие" задачи

# Как будем изучать?

- ~~16 лекций + 16 семинаров~~ 16 пар
- Практика = миньки + "большие" задачи
  - минек сильно меньше, в основном литкод
  - реализовывать на **любом** языке  
(в пределах разумного)

# Как будем изучать?

- ~~16 лекций + 16 семинаров~~ 16 пар
- Практика = миньки + "большие" задачи
  - минек сильно меньше, в основном литкод
  - реализовывать на **любом** языке  
(в пределах разумного)
  - Большим задачам — **большие** данные

# Балловая система

- ✓ Практика – [0; 50] баллов
- ✓ Сдача теории – [0; 50] баллов

# Балловая система

- ✓ Практика – [0; 50] баллов
- ✓ Сдача теории – [0; 50] баллов

На ~~экзамене~~ дифф. зачете: два вопроса по теории

# Балловая система

- ✓ Практика – [0; 50] баллов
  - ✓ Сдача теории – [0; 50] баллов
- 

- ✓ [85; 100] баллов – отл.
- ✓ [70; 85) баллов – хор.
- ✓ [55; 70) баллов – удовл.

# Балловая система

- ✓ Практика – [0; 50] баллов
- ✓ Сдача теории – [0; 50] баллов

- 
- ✓ [85; 100] баллов – отл.
  - ✓ [70; 85) баллов – хор.
  - ✓ [55; 70) баллов – удовл.

45 за практику =  
билет на дом

# Балловая система

- ✓ Практика – [0; 50] баллов
- ✓ Сдача теории – [0; 50] баллов

- 
- ✓ [85; 100] баллов – отл.
  - ✓ [70; 85) баллов – хор.
  - ✓ [55; 70) баллов – удовл.

45 за практику =  
билет на дом

50 за практику =  
-1 доп вопрос по  
теории

# Сдача задачи

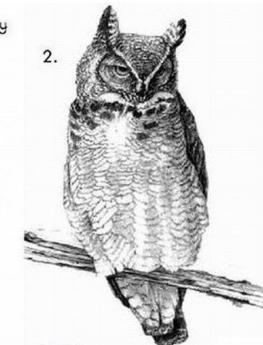
1. Получить свой вариант задачи
2. Написать решение

Как нарисовать сову

1.



2.

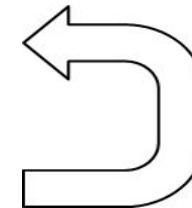


1. Рисуем кружочки

2. Рисуем остаток совы

# Сдача задачи

1. Получить свой вариант задачи
2. Написать решение
3. Пройти автоматические тесты
4. Пройти code review
5. Защитить решение на паре



**DEADLINE  
2 WEEKS**

# Дедлайны

МЫ РАЗОЗЛИЛИ ВРЕМЯ,  
И ТЕПЕРЬ У НАС ВСЕГДА  
ПОЛЧАСА ДО ДЭДЛАЙНА



# Дедлайны

1. Просрочка на неделю ⇒  
задача дешевеет в 2 раза
2. Просрочка еще больше ⇒  
задача не засчитывается

МЫ РАЗОЗЛИЛИ ВРЕМЯ,  
И ТЕПЕРЬ У НАС ВСЕГДА  
ПОЛЧАСА ДО ДЭДЛАЙНА



# Дедлайны

1. Дедлайн доп. задач - одна неделя
2. Сдача по упрощенной схеме (ревью чаще всего на паре)

МЫ РАЗОЗЛИЛИ ВРЕМЯ,  
И ТЕПЕРЬ У НАС ВСЕГДА  
ПОЛЧАСА ДО ДЭДЛАЙНА



# Материалы и референсы

1. Стэнфордский курс (вторая часть)  
*Algorithms: Design and Analysis*
2. Лекторий ФПМИ (МФТИ)
3. Алгоритмы. Построение и анализ  
Томас Кормен, ...



Ура, теперь мы готовы  
приступить к курсу!

# Простой пример

# Простой пример

Интернет, как граф:

вершины – роутеры,

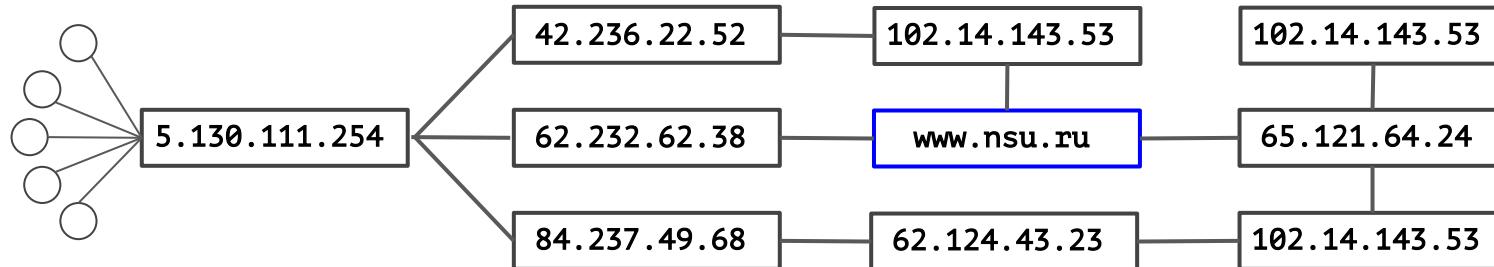
ребра – физическое соединение между роутерами  
(вес, если учитываем задержку сети)

# Простой пример

Интернет, как граф:

вершины – роутеры,

ребра – физическое соединение между роутерами  
(вес, если учитываем задержку сети)



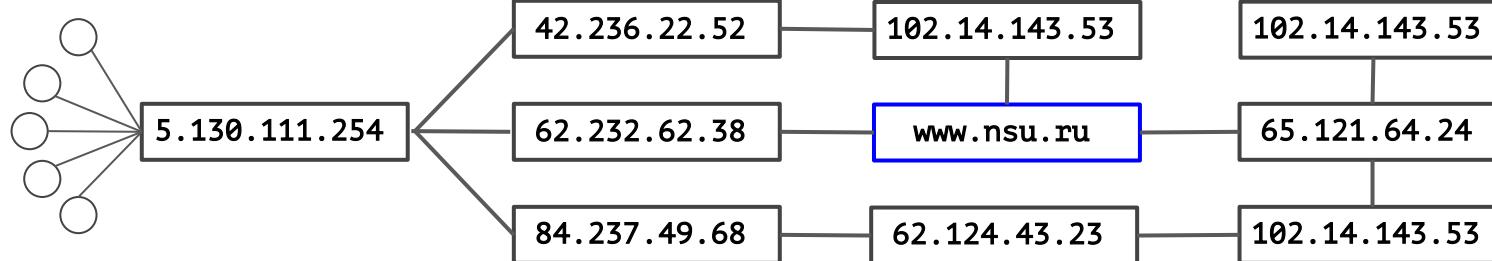
# Простой пример

Интернет, как граф:

вершины – роутеры,

ребра – физическое соединение между роутерами  
(вес, если учитываем задержку сети)

Задача: найти "кратчайший" путь от вашей машины до **цели**.  
(например, в **хопах**, т.е. переходах)



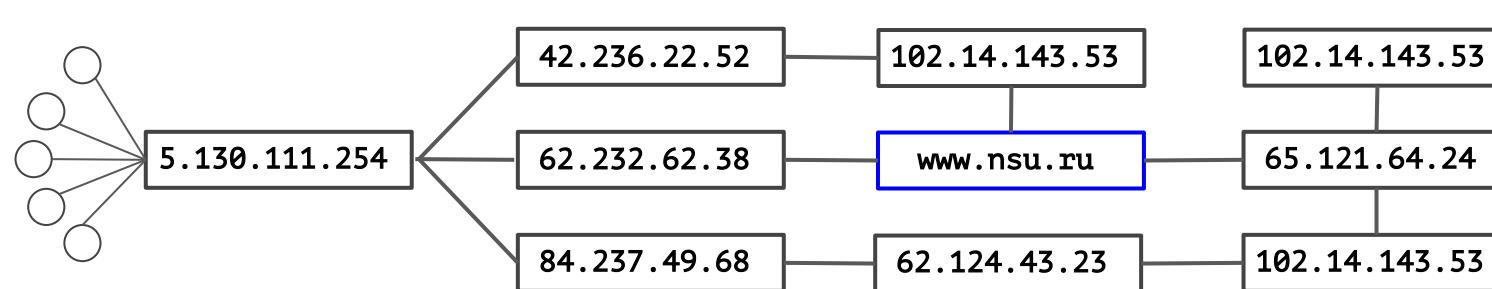
# Простой пример

Интернет, как граф:

вершины – роутеры,

ребра – физическое соединение между роутерами  
(вес, если учитываем задержку сети)

Задача: найти "кратчайший" путь от вашей машины до **цели**.  
(например, в **хопах**, т.е. переходах)



```
C:\Users\ivanu>tracert www.nsu.ru
Трассировка маршрута к www.nsu.ru [84.237.49.214]
с максимальным числом прыжков 30:

 1  <1 ms    <1 ms    <1 ms  dlinkrouter.local [192.168.0.1]
 2  1 ms     2 ms     3 ms   15-130-111-254.novtelecom.ru [5.130.111.254]
 3  1 ms     1 ms     1 ms   10.245.138.241
 4  1 ms     1 ms     1 ms   10.245.138.242
 5  1 ms     1 ms     3 ms   nsk-ix.avantel.ru [193.232.87.46]
 6  2 ms     2 ms     2 ms   host-95-170-130-190.avantel.ru [95.170.130.190]
 7  12 ms    11 ms    3 ms   host68.49.237.84.nsu.ru [84.237.49.68]
 8  3 ms     2 ms     3 ms   host25.49.237.84.nsu.ru [84.237.49.25]
 9  3 ms     3 ms     4 ms   host214.49.237.84.nsu.ru [84.237.49.214]
```

# Простой пример

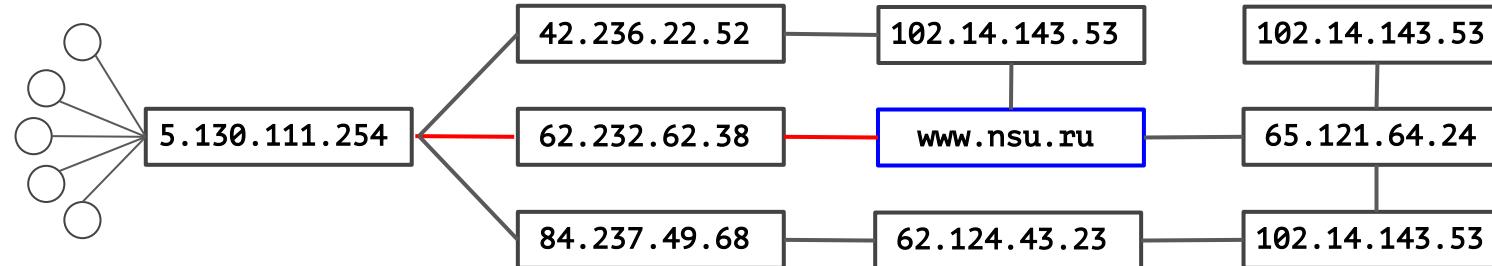
Интернет, как граф:

вершины – роутеры,

ребра – физическое соединение между роутерами  
(вес, если учитываем задержку сети)

Задача: найти "кратчайший" путь от вашей машины до **цели**.

Решение?



```
C:\Users\ivanu>tracert www.nsu.ru
Трассировка маршрута к www.nsu.ru [84.237.49.214]
с максимальным числом прыжков 30:

 1  <1 ms    <1 ms    <1 ms  dlinkrouter.local [192.168.0.1]
 2  1 ms     2 ms     1 ms   15-130-111-254.novtelecom.ru [5.130.111.254]
 3  1 ms     1 ms     1 ms   10.245.138.241
 4  1 ms     1 ms     1 ms   10.245.138.242
 5  1 ms     1 ms     3 ms   nsk-i4.avantel.ru [193.232.87.46]
 6  2 ms     2 ms     2 ms   host-95-170-130-190.avantel.ru [95.170.130.190]
 7  12 ms    11 ms    3 ms   host68.49.237.84.nsu.ru [84.237.49.68]
 8  3 ms     2 ms     3 ms   host25.49.237.84.nsu.ru [84.237.49.25]
 9  3 ms     3 ms     4 ms   host214.49.237.84.nsu.ru [84.237.49.214]
```

# Простой пример

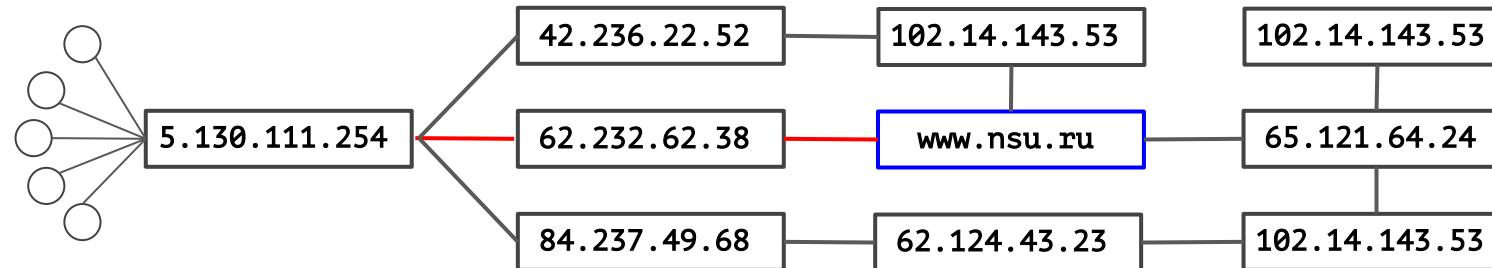
Интернет, как граф:

вершины – роутеры,

ребра – физическое соединение между роутерами  
(вес, если учитываем задержку сети)

Задача: найти "кратчайший" путь от вашей машины до **цели**.

Решение: поиск **кратчайшего пути** в графе!



```
C:\Users\ivanu>tracert www.nsu.ru
Трассировка маршрута к www.nsu.ru [84.237.49.214]
с максимальным числом прыжков 30:

 1  <1 ms    <1 ms    <1 ms  dlinkrouter.local [192.168.0.1]
 2  1 ms     2 ms     3 ms   15-130-111-254.novtelecom.ru [5.130.111.254]
 3  1 ms     1 ms     1 ms   10.245.138.241
 4  1 ms     1 ms     1 ms   10.245.138.242
 5  1 ms     1 ms     3 ms   nsk-ix.avantel.ru [193.232.87.46]
 6  2 ms     2 ms     2 ms   host-95-170-130-190.avantel.ru [95.170.130.190]
 7  12 ms    11 ms    3 ms   host68.49.237.84.nsu.ru [84.237.49.68]
 8  3 ms     2 ms     3 ms   host25.49.237.84.nsu.ru [84.237.49.25]
 9  3 ms     3 ms     4 ms   host214.49.237.84.nsu.ru [84.237.49.214]
```

# Простой пример

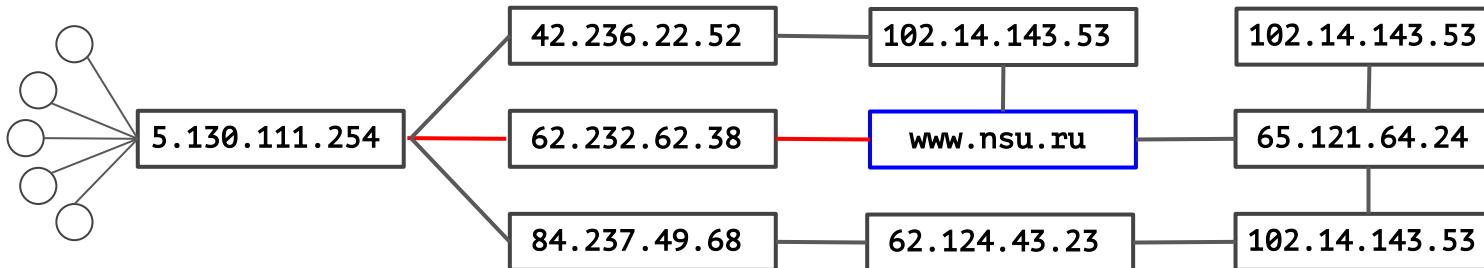
Интернет, как граф:

вершины – роутеры,

ребра – физическое соединение между роутерами  
(вес, если учитываем задержку сети)

Задача: найти "кратчайший" путь от вашей машины до **цели**.

Решение: поиск **кратчайшего пути** в графе! Дейкстра!



```
C:\Users\ivanu>tracert www.nsu.ru
Трассировка маршрута к www.nsu.ru [84.237.49.214]
с максимальным числом прыжков 30:

 1  <1 ms    <1 ms    <1 ms  dlinkrouter.local [192.168.0.1]
 2  1 ms     2 ms     3 ms   15-130-111-254.novtelecom.ru [5.130.111.254]
 3  1 ms     1 ms     1 ms   10.245.138.241
 4  1 ms     1 ms     1 ms   10.245.138.242
 5  1 ms     1 ms     3 ms   nsk-ix.avantel.ru [193.232.87.46]
 6  2 ms     2 ms     2 ms   host-95-170-130-190.avantel.ru [95.170.130.190]
 7  12 ms    11 ms    3 ms   host68.49.237.84.nsu.ru [84.237.49.68]
 8  3 ms     2 ms     3 ms   host25.49.237.84.nsu.ru [84.237.49.25]
 9  3 ms     3 ms     4 ms   host214.49.237.84.nsu.ru [84.237.49.214]
```



# Простой пример

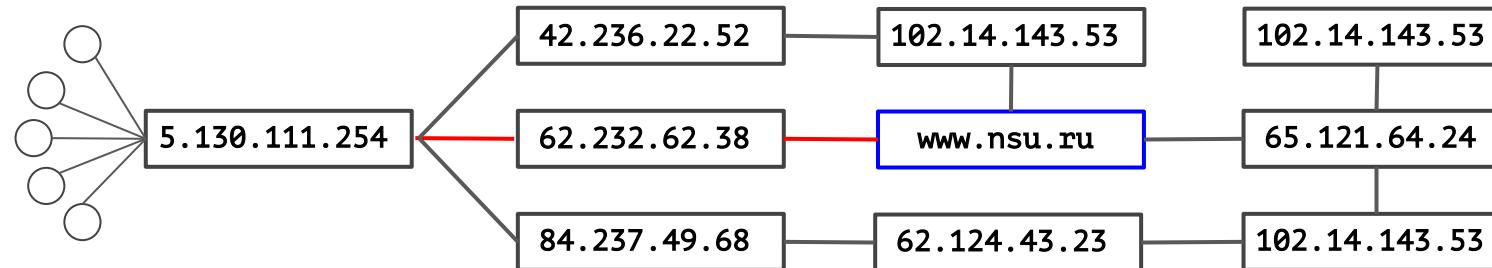
Интернет, как граф:

вершины – роутеры,

ребра – физическое соединение между роутерами  
(вес, если учитываем задержку сети)

Задача: найти "кратчайший" путь от вашей машины до **цели**.

Решение: поиск **кратчайшего пути** в графе! Дейкстры! Проблемы?

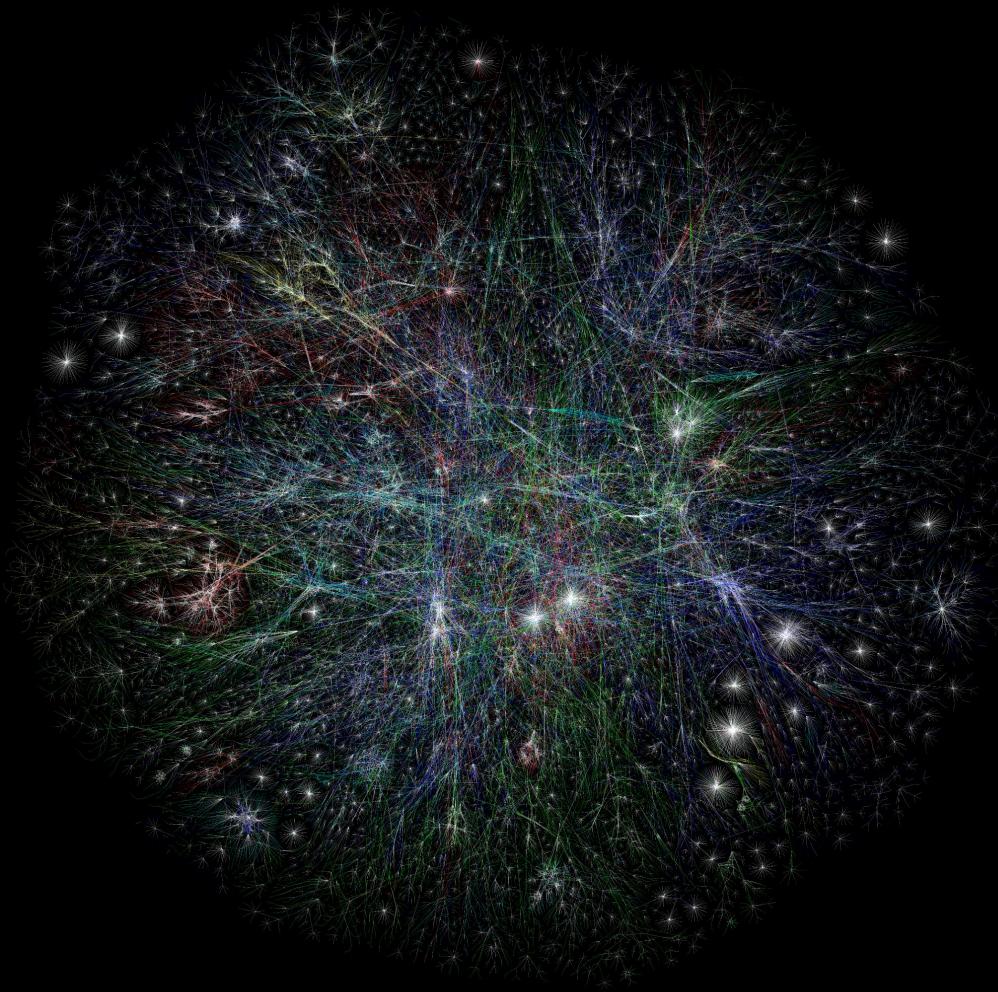


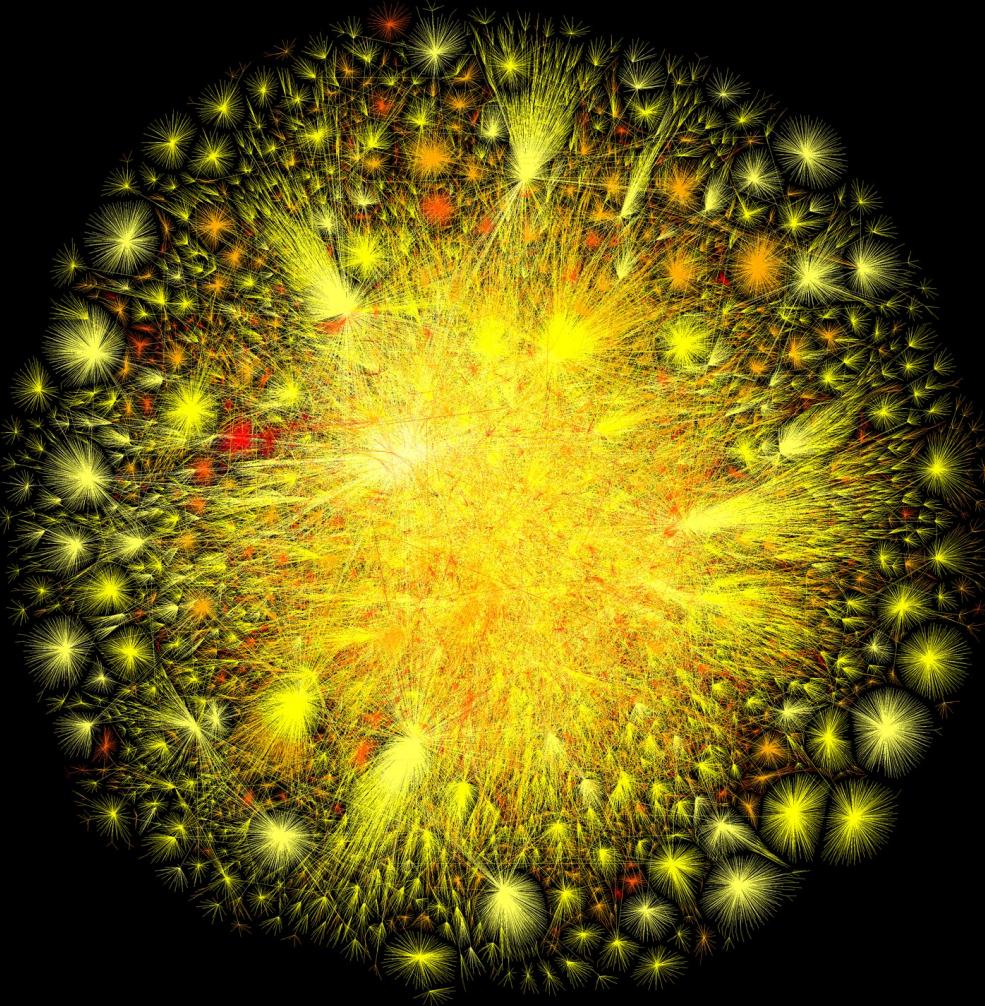
```
C:\Users\ivanu>tracert www.nsu.ru
Трассировка маршрута к www.nsu.ru [84.237.49.214]
с максимальным числом прыжков 30:

 1  <1 ms    <1 ms    <1 ms  dlinkrouter.local [192.168.0.1]
 2  1 ms     2 ms     3 ms   15-130-111-254.novtelecom.ru [5.130.111.254]
 3  1 ms     1 ms     1 ms   10.245.138.241
 4  1 ms     1 ms     1 ms   10.245.138.242
 5  1 ms     1 ms     3 ms   nsk-ix.avantel.ru [193.232.87.46]
 6  2 ms     2 ms     2 ms   host-95-170-130-190.avantel.ru [95.170.130.190]
 7  12 ms    11 ms    3 ms   host68.49.237.84.nsu.ru [84.237.49.68]
 8  3 ms     2 ms     3 ms   host25.49.237.84.nsu.ru [84.237.49.25]
 9  3 ms     3 ms     4 ms   host214.49.237.84.nsu.ru [84.237.49.214]
```



2003





2010

# Простой пример

Интернет, как граф:

вершины – роутеры,

ребра – физическое соединение между роутерами  
(вес, если учитываем задержку сети)

Задача: найти "кратчайший" путь от вашей машины до **цели**.

Решение: поиск **кратчайшего пути** в графе! Дейкстра!

Проблема: **большеват** граф то! Нет никакой возможности хранить его модель локально, чтобы запустить Дейкстру.

```
C:\Users\ivanu>tracert www.nsu.ru

Трассировка маршрута к www.nsu.ru [84.237.49.214]
с максимальным числом прыжков 30:

 1  <1 ms    <1 ms    <1 ms  dlinkrouter.local [192.168.0.1]
 2  1 ms     2 ms     3 ms  15-130-111-254.novtelecom.ru [5.130.111.254]
 3  1 ms     1 ms     1 ms  10.245.138.241
 4  1 ms     1 ms     1 ms  10.245.138.242
 5  1 ms     1 ms     3 ms  nsk-ix.avantel.ru [193.232.87.46]
 6  2 ms     2 ms     2 ms  host-95-170-130-190.avantel.ru [95.170.130.190]
 7  12 ms    11 ms    3 ms  host68.49.237.84.nsu.ru [84.237.49.68]
 8  3 ms     2 ms     3 ms  host25.49.237.84.nsu.ru [84.237.49.25]
 9  3 ms     3 ms     4 ms  host214.49.237.84.nsu.ru [84.237.49.214]
```



# Простой пример

Интернет, как граф:

вершины – роутеры,

ребра – физическое соединение между роутерами  
(вес, если учитываем задержку сети)

Задача: найти "кратчайший" путь от вашей машины до [цели](#).

Решение: поиск [кратчайшего пути](#) в графе! Дейкстра!

Проблема: **большеват** граф то! Нет никакой возможности хранить его модель локально, чтобы запустить Дейкстру.

[Идея](#): нужен алгоритм для распределенного поиска кратчайшего пути.

```
C:\Users\ivanu>tracert www.nsu.ru
Трассировка маршрута к www.nsu.ru [84.237.49.214]
с максимальным числом прыжков 30:

  1  <1 ms    <1 ms    <1 ms  dlinkrouter.local [192.168.0.1]
  2  1 ms     2 ms     3 ms  15-130-111-254.novtelecom.ru [5.130.111.254]
  3  1 ms     1 ms     1 ms  10.245.138.241
  4  1 ms     1 ms     1 ms  10.245.138.242
  5  1 ms     1 ms     3 ms  nsk-ix.avantel.ru [193.232.87.46]
  6  2 ms     2 ms     2 ms  host-95-170-130-190.avantel.ru [95.170.130.190]
  7  12 ms    11 ms    3 ms  host68.49.237.84.nsu.ru [84.237.49.68]
  8  3 ms     2 ms     3 ms  host25.49.237.84.nsu.ru [84.237.49.25]
  9  3 ms     3 ms     4 ms  host214.49.237.84.nsu.ru [84.237.49.214]
```



# Простой пример

Интернет, как граф:

вершины – роутеры,

ребра – физическое соединение между роутерами  
(вес, если учитываем задержку сети)

Задача: найти "кратчайший" путь от вашей машины до **цели**.

Решение: поиск **кратчайшего пути** в графе! Дейкстра!

Проблема: **большеват** граф то! Нет никакой возможности хранить его модель локально, чтобы запустить Дейкстру.

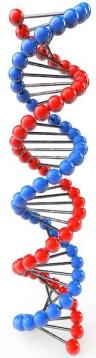
**Идея**: нужен алгоритм для распределенного поиска кратчайшего пути. **Спойлер**: это модификация Беллмана-Форда

```
C:\Users\ivanu>tracert www.nsu.ru
Трассировка маршрута к www.nsu.ru [84.237.49.214]
с максимальным числом прыжков 30:

 1  <1 ms    <1 ms    <1 ms  dlinkrouter.local [192.168.0.1]
 2  1 ms     2 ms     3 ms  15-130-111-254.novtelecom.ru [5.130.111.254]
 3  1 ms     1 ms     1 ms  10.245.138.241
 4  1 ms     1 ms     1 ms  10.245.138.242
 5  1 ms     1 ms     3 ms  nsk-ix.avantel.ru [193.232.87.46]
 6  2 ms     2 ms     2 ms  host-95-170-130-190.avantel.ru [95.170.130.190]
 7  12 ms    11 ms    3 ms  host68.49.237.84.nsu.ru [84.237.49.68]
 8  3 ms     2 ms     3 ms  host25.49.237.84.nsu.ru [84.237.49.25]
 9  3 ms     3 ms     4 ms  host214.49.237.84.nsu.ru [84.237.49.214]
```

## Простой пример #2

Пусть есть две строки,  
состоящие из символов алфавита  $\{A, C, G, T\}$



## Простой пример #2

Пусть есть две строки,  
состоящие из символов алфавита {A, C, G, T}

**Задача:** понять, "похожи" они или нет?

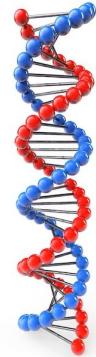


## Простой пример #2

Пусть есть две строки,  
состоящие из символов алфавита {A, C, G, T}

**Задача:** понять, "похожи" они или нет?

Пример: GTTAC и GACGT. Похожи?



## Простой пример #2

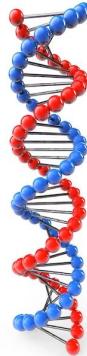
Пусть есть две строки,  
состоящие из символов алфавита {A, C, G, T}

**Задача:** понять, "похожи" они или нет?

Пример:

GTTAC  
GACGT

Похожи?



## Простой пример #2

Пусть есть две строки,  
состоящие из символов алфавита {A, C, G, T}

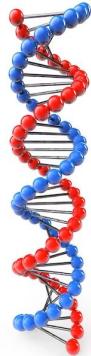
**Задача:** понять, "похожи" они или нет?

Пример:

GTTAC--

G--ACGT

Похожи? А так?



## Простой пример #2

Пусть есть две строки,  
состоящие из символов алфавита {A, C, G, T}

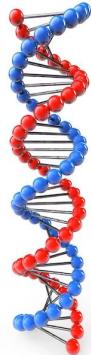
**Задача:** понять, "похожи" они или нет?

Пример:

GTTAC--

G--ACGT

Похожи? А так?



## Простой пример #2

Пусть есть две строки,  
состоящие из символов алфавита {A, C, G, T}

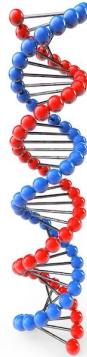
**Задача:** понять, "похожи" они или нет?

Пример:

GTT**AC**--

G--**ACGT**

**Задача:** найти оптимальное **выравнивание**.



## Простой пример #2

Пусть есть две строки,  
состоящие из символов алфавита {A, C, G, T}

**Задача:** понять, "похожи" они или нет?

Пример:

GTTAC--

G---ACGT

Вводится  $S(x, y)$  - похожесть символов  $x$  и  $y$ ;  
 $d < 0$  - штраф за разрыв;

**Задача:** найти выравнивание, на котором максимизируется  
сумма  $S$  символов по позициям



## Простой пример #2

Пусть есть две строки,  
состоящие из символов алфавита {A, C, G, T}

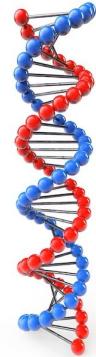
**Задача:** понять, "похожи" они или нет?

Пример:

GTT**A**C--

G--**A**CGT

**Решение:** Алгоритм Нидлмана – Вунша (еще один классический пример алгоритма динамического программирования)



# Алгоритмы и структуры данных

Жадные алгоритмы: начало



# Жадный алгоритм

Жадный алгоритм на каждой итерации принимает "локально" оптимальный выбор.



# Жадный алгоритм

Жадный алгоритм на каждой итерации принимает "локально" оптимальный выбор.

Т.е. на каждом шаге принимает решение, которое сулит наибольшую выгоду прямо сейчас.

(точное математическое определение будет позже)



# Жадный алгоритм

Жадный алгоритм на каждой итерации принимает "локально" оптимальный выбор.

Т.е. на каждом шаге принимает решение, которое сулит наибольшую выгоду прямо сейчас.

(точное математическое определение будет позже)

Пример?



# Жадный алгоритм

Жадный алгоритм на каждой итерации принимает "локально" оптимальный выбор.

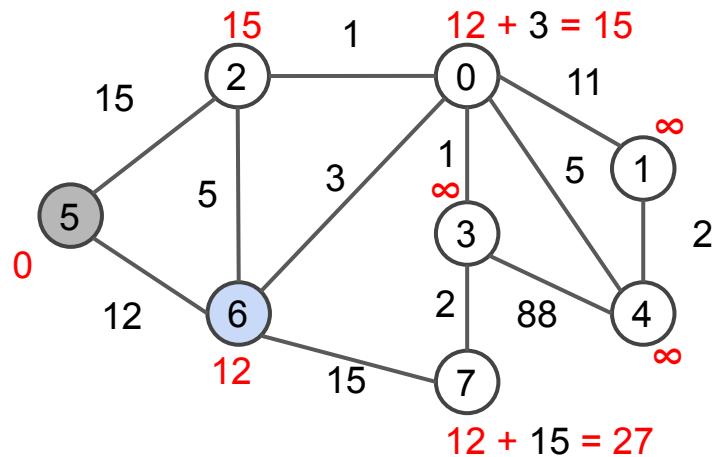
Т.е. на каждом шаге принимает решение, которое сулит наибольшую выгоду прямо сейчас.

(точное математическое определение будет позже)

Пример: тот же Дейкстра!



# Алгоритм Дейкстры поиска кратчайших путей



Ищем расстояние от **пятой** вершины до всех остальных.

Изначально все остальные пометки - **бесконечность**

visited = 

0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

dists = 

15	$\infty$	15	$\infty$	$\infty$	0	12	27
----	----------	----	----------	----------	---	----	----

1. Улучшаем **текущее** расстояние до соседних с текущей (еще не обработанных) вершинах
2. Пометить текущую, как обработанную
3. В качестве следующей вершины берем ту, в которой еще не были, и у которой **минимальный** dist

# Жадный алгоритм

Жадный алгоритм на каждой итерации принимает "локально" оптимальный выбор.

Т.е. на каждом шаге принимает решение, которое сулит наибольшую выгоду прямо сейчас.

(точное математическое определение будет позже)

Пример: тот же Дейкстра!

(т.к. выбираем всегда ближайшую вершину)



# Жадный алгоритм

Жадный алгоритм на каждой итерации принимает "локально" оптимальный выбор.

Обычно жадные алгоритмы просто:

1. предложить
2. реализовать
3. оценить их сложность

# Жадный алгоритм

Жадный алгоритм на каждой итерации принимает "локально" оптимальный выбор.

Обычно жадные алгоритмы просто:

1. предложить
2. реализовать
3. оценить их сложность

Что сложно в жадных алгоритмах:

1. получить корректное решение

# Жадный алгоритм

Жадный алгоритм на каждой итерации принимает "локально" оптимальный выбор.

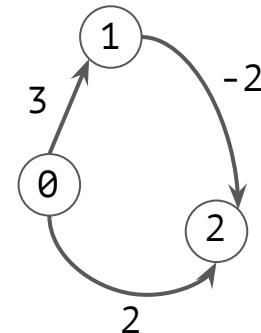
Обычно жадные алгоритмы просто:

1. предложить
2. реализовать
3. оценить их сложность

Что сложно в жадных алгоритмах:

1. получить корректное решение

Какой кратчайший путь из 0 в 2 выдаст алгоритм Дейкстры?



# Жадный алгоритм

Жадный алгоритм на каждой итерации принимает "локально" оптимальный выбор.

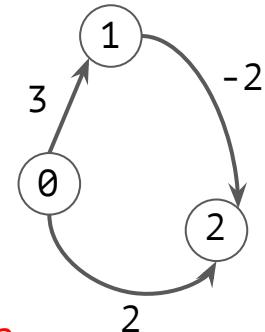
Обычно жадные алгоритмы просто:

1. предложить
2. реализовать
3. оценить их сложность

Что сложно в жадных алгоритмах:

1. получить корректное решение

Какой кратчайший путь из 0 в 2 выдаст алгоритм Дейкстры?



Выдаст 2, хотя правильный ответ — 1.

# Жадный алгоритм

Жадный алгоритм на каждой итерации принимает "локально" оптимальный выбор.

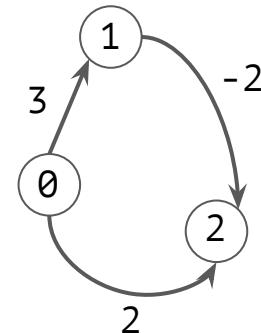
Обычно жадные алгоритмы просто:

1. предложить
2. реализовать
3. оценить их сложность

Что сложно в жадных алгоритмах:

1. получить корректное решение

Какой кратчайший путь из 0 в 2 выдаст алгоритм Дейкстры?



Из-за отрицательных весов жадный алгоритм перестал быть корректным!

# Задача планирования

# Задача планирования

Пусть есть один исполнитель и набор из задач, которые нужно выполнить.

# Задача планирования

Пусть есть один **исполнитель** и набор из задач, которые нужно выполнить.

Для каждой задачи  $j$  определены:

1. Длительность:  $l_j$
2. Вес (приоритет):  $w_j$

# Задача планирования

Пусть есть один **исполнитель** и набор из задач, которые нужно выполнить.

Для каждой задачи  $j$  определены:

1. Длительность:  $l_j$
2. Вес (приоритет):  $w_j$

Кроме того, введем  $c_j$  время окончание работы  $j$ , как сумму длительностей задач от 0 до  $j$

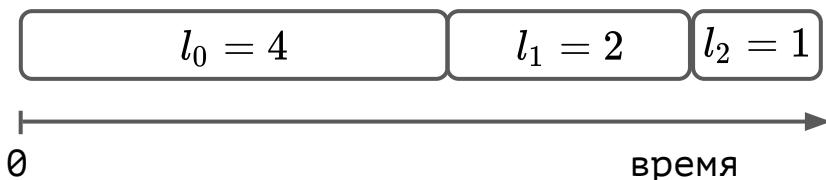
# Задача планирования

Пусть есть один **исполнитель** и набор из задач, которые нужно выполнить.

Для каждой задачи  $j$  определены:

1. Длительность:  $l_j$
2. Вес (приоритет):  $w_j$

Кроме того, введем  $c_j$  время окончание работы  $j$ , как сумму длительностей задач от 0 до  $j$



# Задача планирования

Пусть есть один **исполнитель** и набор из задач, которые нужно выполнить.

Для каждой задачи  $j$  определены:

1. Длительность:  $l_j$
2. Вес (приоритет):  $w_j$

Кроме того, введем  $c_j$  время окончание работы  $j$ , как сумму длительностей задач от 0 до  $j$

$$c_0 = 4$$

$$c_1 = 6 \quad c_2 = 7$$

$$l_0 = 4$$

$$l_1 = 2$$

$$l_2 = 1$$



# Задача планирования

Пусть есть один **исполнитель** и набор из задач, которые нужно выполнить.

Для каждой задачи  $j$  определены:

1. Длительность:  $l_j$
2. Вес (приоритет):  $w_j$

Кроме того, введем  $c_j$  время окончание работы  $j$ , как сумму длительностей задач от 0 до  $j$

$$c_0 = 4$$

$$c_1 = 6 \quad c_2 = 7$$

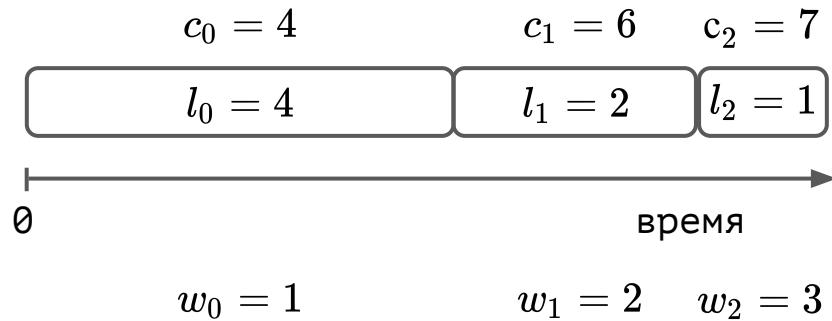
$$l_0 = 4$$

$$l_1 = 2 \quad l_2 = 1$$



**Задача:** найти такой порядок задач  $\sigma$ , чтобы функция  $S(\sigma) = \sum c_j * w_j$  принимала минимальное значение

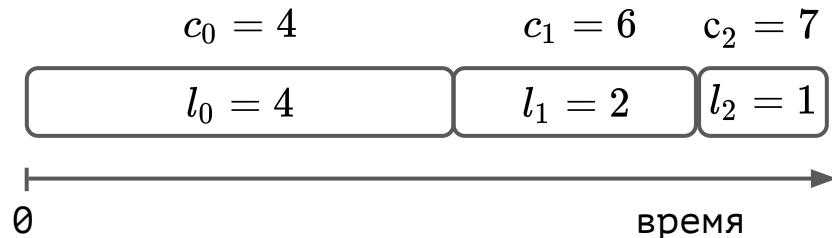
# Задача планирования: пример



**Задача:** минимизировать  
функцию  $S(\sigma) = \sum c_j * w_j$

$$\sum c_j * w_j = 1 * 4 + 2 * 6 + 3 * 7 = 37$$

# Задача планирования: пример

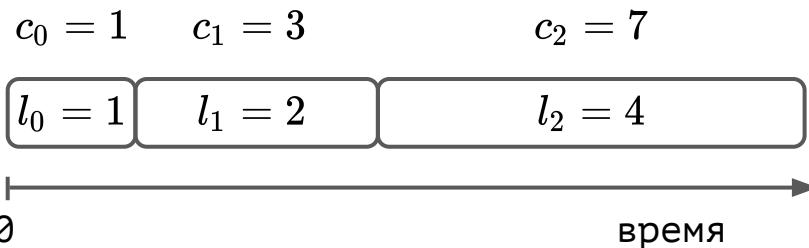


$$w_0 = 1$$

$$w_1 = 2 \quad w_2 = 3$$

**Задача:** минимизировать  
функцию  $S(\sigma) = \sum c_j * w_j$

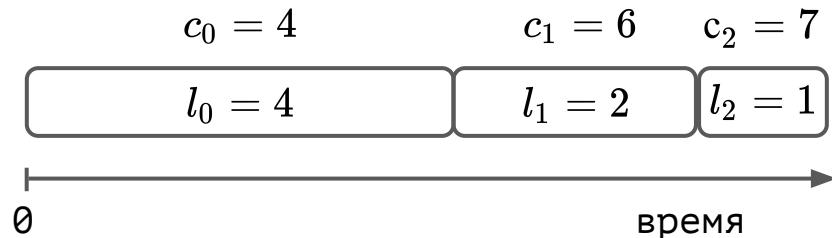
$$\sum c_j * w_j = 1 * 4 + 2 * 6 + 3 * 7 = 37$$



$$w_0 = 3 \quad w_1 = 2$$

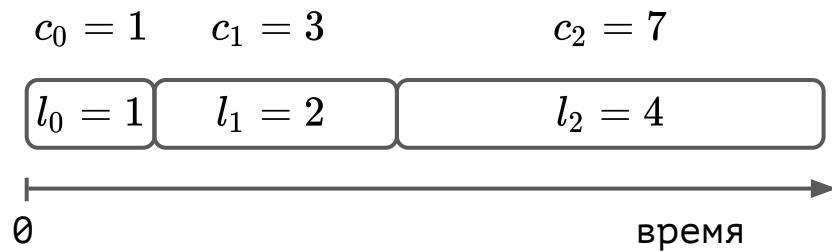
$$w_2 = 1$$

# Задача планирования: пример



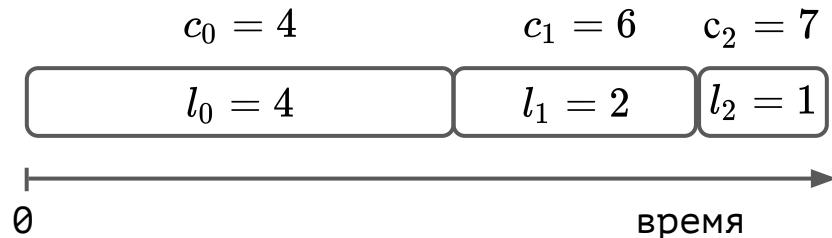
**Задача:** минимизировать  
функцию  $S(\sigma) = \sum c_j * w_j$

$$\sum c_j * w_j = 1 * 4 + 2 * 6 + 3 * 7 = 37$$



$$\sum c_j * w_j = 3 * 1 + 2 * 3 + 1 * 7 = \underline{\underline{16}}$$

# Задача планирования: пример

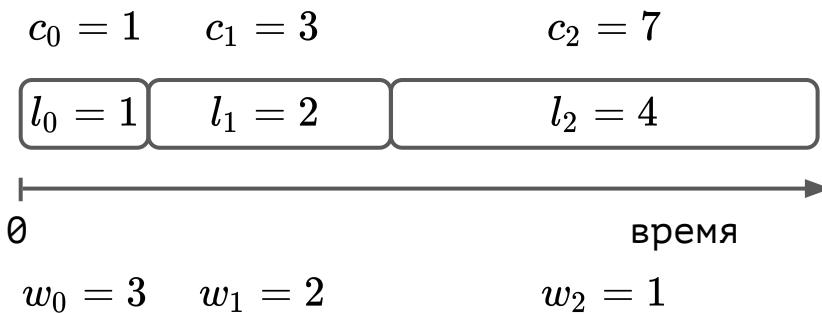


$$w_0 = 1$$

$$w_1 = 2 \quad w_2 = 3$$

**Задача:** минимизировать функцию  $S(\sigma) = \sum c_j * w_j$

$$\sum c_j * w_j = 1 * 4 + 2 * 6 + 3 * 7 = 37$$



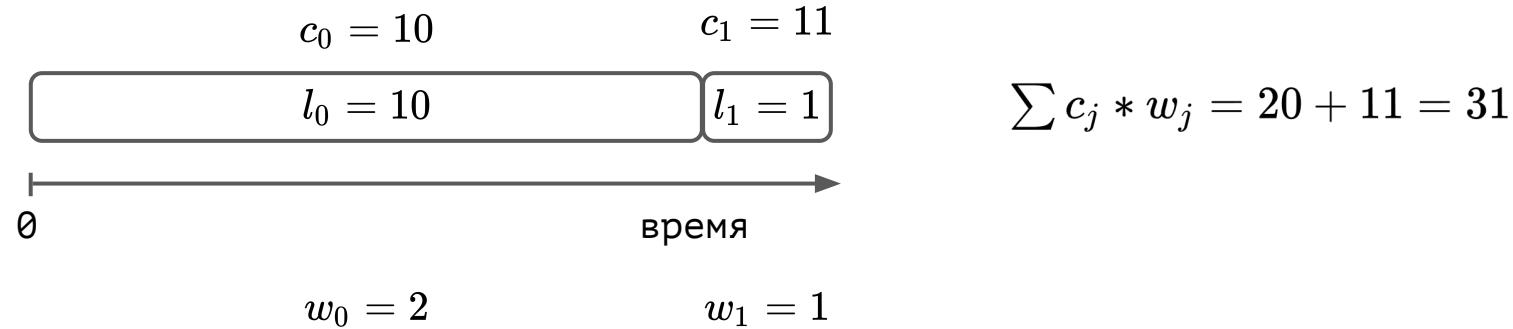
$$w_0 = 3 \quad w_1 = 2$$

$$w_2 = 1$$

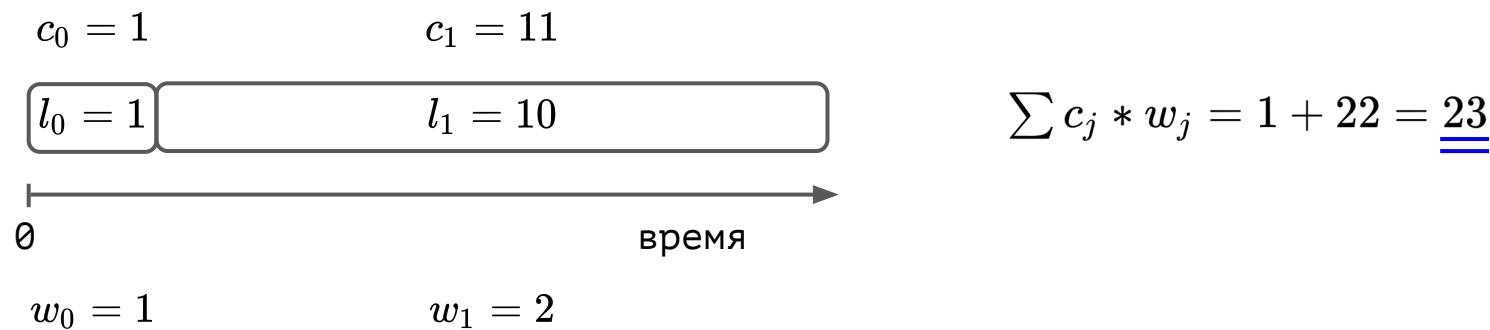
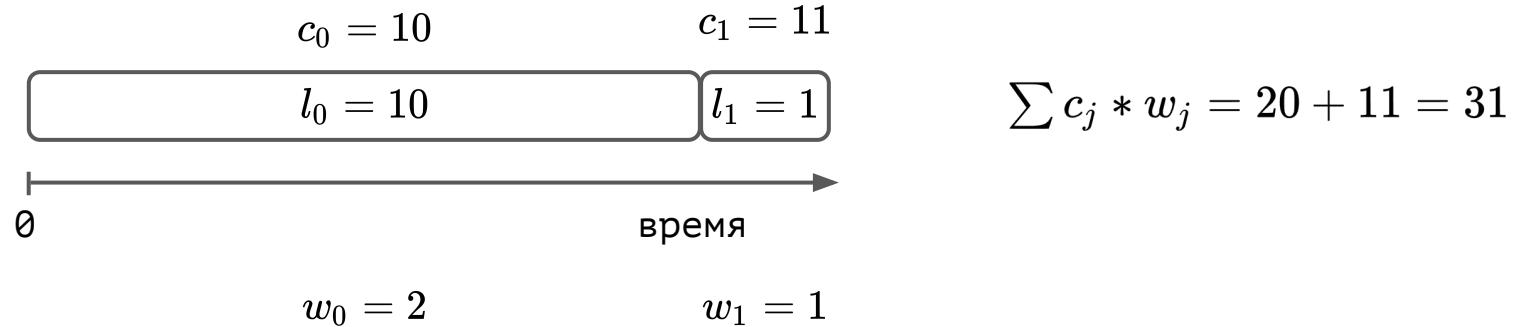
$$\sum c_j * w_j = 3 * 1 + 2 * 3 + 1 * 7 = \underline{\underline{16}}$$

Всегда ли достаточно ставить вперед наивысший приоритет?

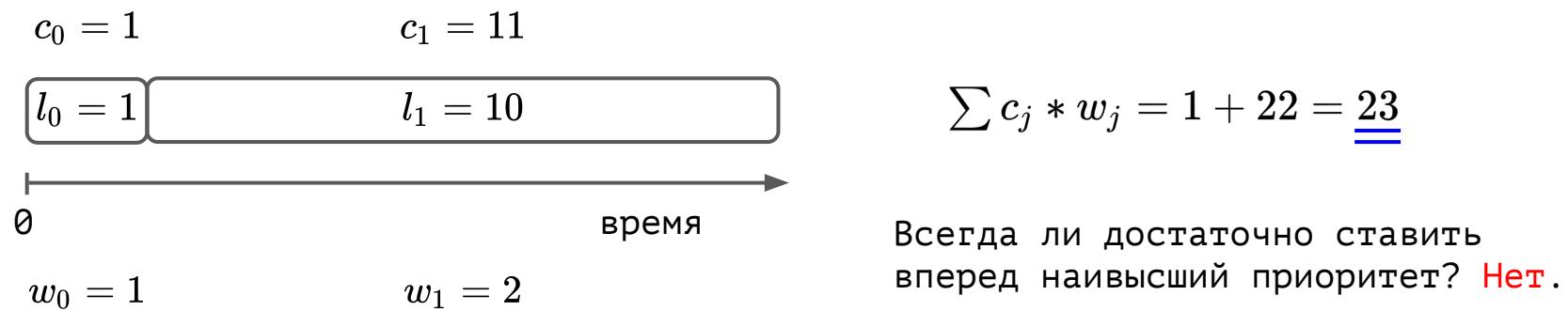
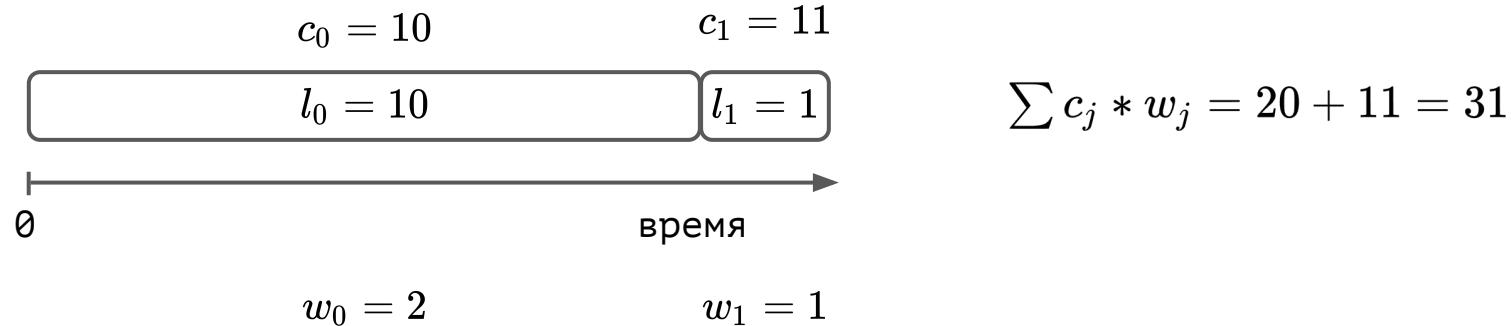
# Задача планирования: пример



# Задача планирования: пример



# Задача планирования: пример



# Задача планирования: пример

Рассмотрим два частных случая:

# Задача планирования: пример

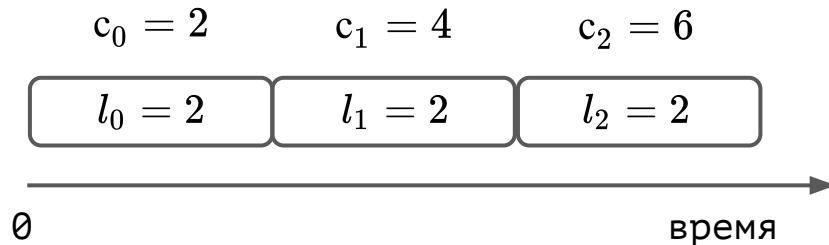
Рассмотрим два частных случая:

1. Пусть все задания одной **длительности**

# Задача планирования: пример

Рассмотрим два частных случая:

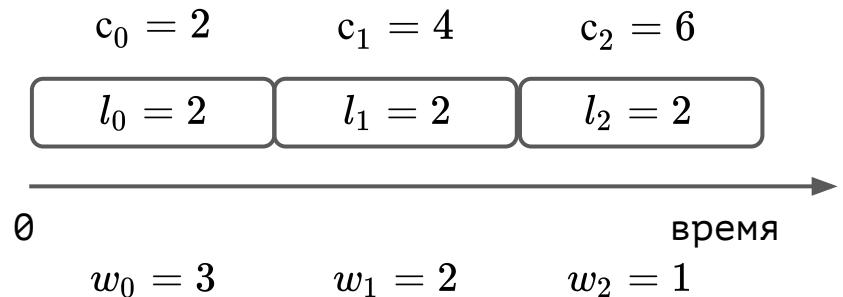
1. Пусть все задания одной длительности



# Задача планирования: пример

Рассмотрим два частных случая:

- Пусть все задания одной длительности



Тогда с большим приоритетом исполняем раньше.

# Задача планирования: пример

Рассмотрим два частных случая:

1. Пусть все задания одной **длительности**  
(тогда с более высоким приоритетом исполняем раньше)
2. Пусть у всех заданий одинаковый приоритет, но разная длительность

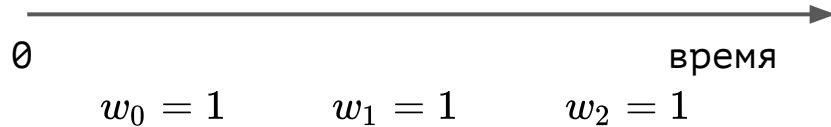
# Задача планирования: пример

Рассмотрим два частных случая:

1. Пусть все задания одной **длительности**

(тогда с более высоким приоритетом исполняем раньше)

2. Пусть у всех заданий одинаковый приоритет, но разная длительность



# Задача планирования: пример

Рассмотрим два частных случая:

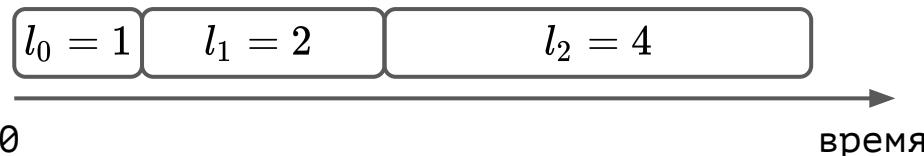
- Пусть все задания одной **длительности**

(тогда с более высоким приоритетом исполняем раньше)

- Пусть у всех заданий одинаковый приоритет, но разная длительность

$$c_0 = 1 \quad c_1 = 3$$

$$c_2 = 7$$



$$\sum c_j = 11$$

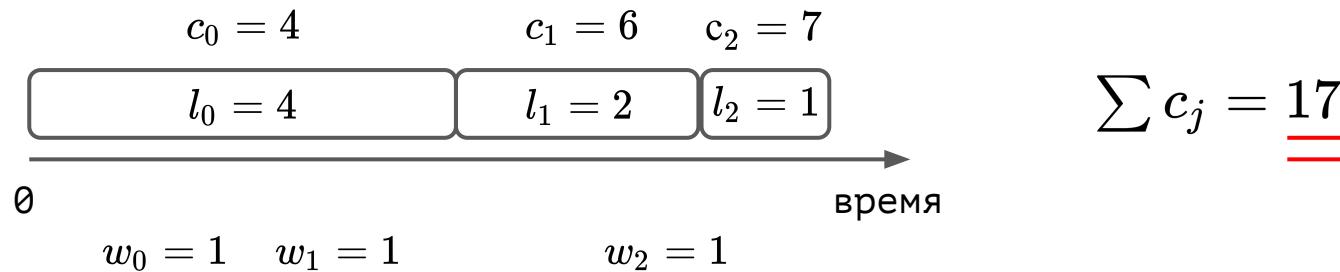
# Задача планирования: пример

Рассмотрим два частных случая:

- Пусть все задания одной **длительности**

(тогда с более высоким приоритетом исполняем раньше)

- Пусть у всех заданий одинаковый приоритет, но разная длительность



# Задача планирования: пример

Рассмотрим два частных случая:

1. Пусть все задания одной **длительности**

(тогда с более высоким приоритетом исполняем раньше)

2. Пусть у всех заданий одинаковый приоритет, но разная длительность

(тогда более короткие исполняем раньше)

# Задача планирования: пример

Рассмотрим два частных случая:

1. Пусть все задания одной **длительности**

(тогда с **более высоким приоритетом** исполняем раньше)

2. Пусть у всех заданий одинаковый приоритет, но разная длительность

(тогда **более короткие** исполняем раньше)

Значит нужно для каждого задания получить **метрику**, которая растет при росте приоритета и уменьшается при росте длины.

# Задача планирования

Значит нужно для каждого задания получить [метрику](#), которая растет при росте приоритета и уменьшается при росте длины.

Варианты?

# Задача планирования

Значит нужно для каждого задания получить [метрику](#), которая растет при росте приоритета и уменьшается при росте длины.

Варианты?

1. разность  $w_j - l_j$

2. отношение  $\frac{w_j}{l_j}$

# Задача планирования

Значит нужно для каждого задания получить [метрику](#), которая растет при росте приоритета и уменьшается при росте длины.

Варианты?

1. разность  $w_j - l_j$

2. отношение  $\frac{w_j}{l_j}$

Жадный алгоритм: на каждом шаге берем работу с наибольшей метрикой (пусть пока все метрики получились разными)

# Задача планирования

Значит нужно для каждого задания получить [метрику](#), которая растет при росте приоритета и уменьшается при росте длины.

Варианты?

1. разность  $w_j - l_j$

2. отношение  $\frac{w_j}{l_j}$

Получили сразу два жадных алгоритма!

При этом такие стратегии явно дает разные последовательности элементов

Жадный алгоритм: на каждом шаге берем работу с наибольшей метрикой (пусть пока все метрики получились разными)

# Задача планирования

Значит нужно для каждого задания получить **метрику**, которая растет при росте приоритета и уменьшается при росте длины.

Варианты?

$$1. \text{ разность } w_j - l_j$$

$$2. \text{ отношение } \frac{w_j}{l_j}$$

Получили сразу два жадных алгоритма!

При этом такие стратегии явно дает разные последовательности элементов  $\Rightarrow$  разные ответы.

Жадный алгоритм: на каждом шаге берем работу с наибольшей метрикой (пусть пока все метрики получились разными)

# Задача планирования

Значит нужно для каждого задания получить **метрику**, которая растет при росте приоритета и уменьшается при росте длины.

Варианты?

$$1. \text{ разность } w_j - l_j$$

$$2. \text{ отношение } \frac{w_j}{l_j}$$

Получили сразу два жадных алгоритма!

При этом такие стратегии явно дает разные последовательности элементов  $\Rightarrow$  разные ответы. Так какая стратегия лучше?

Жадный алгоритм: на каждом шаге берем работу с наибольшей метрикой (пусть пока все метрики получились разными)

## Задача планирования: пример

Предложите пример, на котором эти два жадных алгоритма поведут себя по-разному? (а мы поймем, кто из них хуже)

## Задача планирования: пример

Предложите пример, на котором эти два жадных алгоритма поведут себя по-разному? (а мы поймем, кто из них хуже)

$$w_0 = 3 \quad w_1 = 1$$

$$l_0 = 5 \quad l_1 = 2$$

# Задача планирования: пример

Предложите пример, на котором эти два жадных алгоритма поведут себя по-разному? (а мы поймем, кто из них хуже)

$$w_0 = 3 \quad w_1 = 1$$

$$l_0 = 5 \quad l_1 = 2$$

1-ый алгоритм:  
(разность)  $l_1 = 2$   $l_0 = 5$   $\sum c_j * w_j = 2 + 7 * 3 = 23$

$$w_1 = 1 \quad w_0 = 3$$

# Задача планирования: пример

Предложите пример, на котором эти два жадных алгоритма поведут себя по-разному? (а мы поймем, кто из них хуже)

$$w_0 = 3 \quad w_1 = 1$$

$$l_0 = 5 \quad l_1 = 2$$

1-ый алгоритм:  
(разность)  $l_1 = 2$   $l_0 = 5$   $\sum c_j * w_j = 2 + 7 * 3 = 23$

$w_1 = 1$   $w_0 = 3$

2-ой алгоритм:  
(отношение)  $l_0 = 5$   $l_1 = 2$   $\sum c_j * w_j = 15 + 7 = 22$

$w_0 = 3$   $w_1 = 1$

# Задача планирования: пример

Предложите пример, на котором эти два жадных алгоритма поведут себя по-разному? (а мы поймем, кто из них хуже)

$$w_0 = 3 \quad w_1 = 1$$

$$l_0 = 5 \quad l_1 = 2$$

1-ый алгоритм:  
(разность)  $l_1 = 2$   $l_0 = 5$   $\sum c_j * w_j = 2 + 7 * 3 = 23$   
 $w_1 = 1$   $w_0 = 3$

2-ой алгоритм:  
(отношение)  $l_0 = 5$   $l_1 = 2$   $\sum c_j * w_j = 15 + 7 = 22$   
 $w_0 = 3$   $w_1 = 1$



## Задача планирования: корректность

**Утверждение:** если отсортировать работы по убыванию отношения  $\frac{w_j}{l_j}$ , полученное решение будет оптимальным.

## Задача планирования: корректность

**Утверждение:** если отсортировать работы по убыванию отношения  $\frac{w_j}{l_j}$ , полученное решение будет оптимальным.

**Допущение:** пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

# Задача планирования: корректность

**Утверждение:** если отсортировать работы по убыванию отношения  $\frac{w_j}{l_j}$ , полученное решение будет оптимальным.

**Допущение:** пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство:** обозначим порядок работ, полученный жадным алгоритмом через  $\sigma$ .

# Задача планирования: корректность

**Утверждение:** если отсортировать работы по убыванию отношения  $\frac{w_j}{l_j}$ , полученное решение будет оптимальным.

**Допущение:** пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство:** обозначим порядок работ, полученный жадным алгоритмом через  $\sigma$ . Пусть есть другой действительно оптимальный порядок, его обозначим через  $\sigma^*$ .

# Задача планирования: корректность

**Утверждение:** если отсортировать работы по убыванию отношения  $\frac{w_j}{l_j}$ , полученное решение будет оптимальным.

**Допущение:** пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство:** обозначим порядок работ, полученный жадным алгоритмом через  $\sigma$ . Пусть есть другой действительно оптимальный порядок, его обозначим через  $\sigma^*$ .

Пусть в  $\sigma$  работы были занумерованы  $0, 1, \dots, n - 1$ .

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** обозначим порядок работ, полученный жадным алгоритмом через  $\sigma$ . Пусть есть другой действительно оптимальный порядок, его обозначим через  $\sigma^*$ .

Пусть в  $\sigma$  работы были занумерованы от  $0, 1, \dots, n - 1$ .

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** обозначим порядок работ, полученный жадным алгоритмом через  $\sigma$ . Пусть есть другой действительно оптимальный порядок, его обозначим через  $\sigma^*$ .

Пусть в  $\sigma$  работы были занумерованы от  $0, 1, \dots, n - 1$ .

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

$\sigma$  и  $\sigma^*$  различны  $\Rightarrow$  в  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

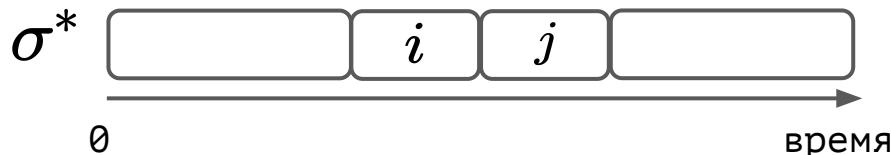
**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** обозначим порядок работ, полученный жадным алгоритмом через  $\sigma$ . Пусть есть другой действительно оптимальный порядок, его обозначим через  $\sigma^*$ .

Пусть в  $\sigma$  работы были занумерованы от  $0, 1, \dots, n - 1$ .

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

$\sigma$  и  $\sigma^*$  различны  $\Rightarrow$  в  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .



**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

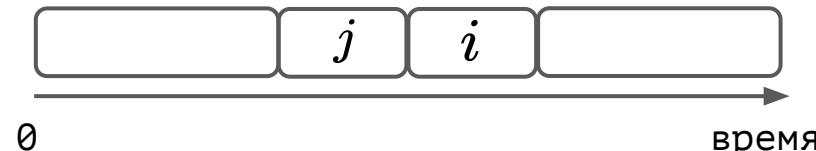
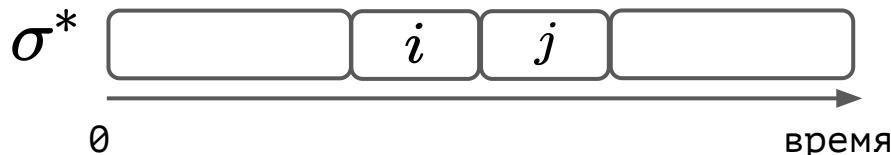
**Доказательство:** обозначим порядок работ, полученный жадным алгоритмом через  $\sigma$ . Пусть есть другой действительно оптимальный порядок, его обозначим через  $\sigma^*$ .

Пусть в  $\sigma$  работы были занумерованы от  $0, 1, \dots, n - 1$ .

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

$\sigma$  и  $\sigma^*$  различны  $\Rightarrow$  в  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

Тогда рассмотрим  $\sigma^{**}$ :

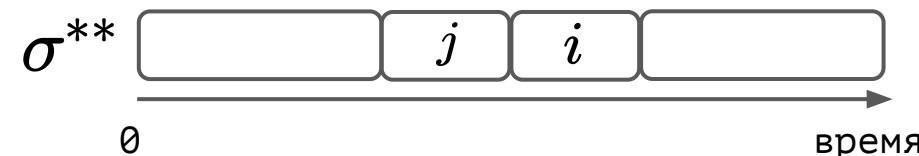
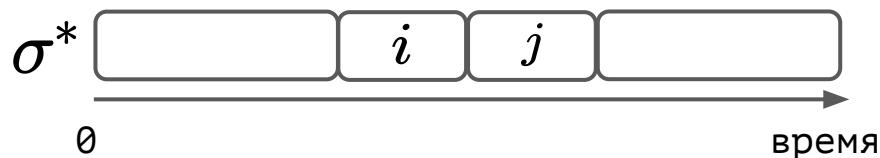


**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** ...

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .



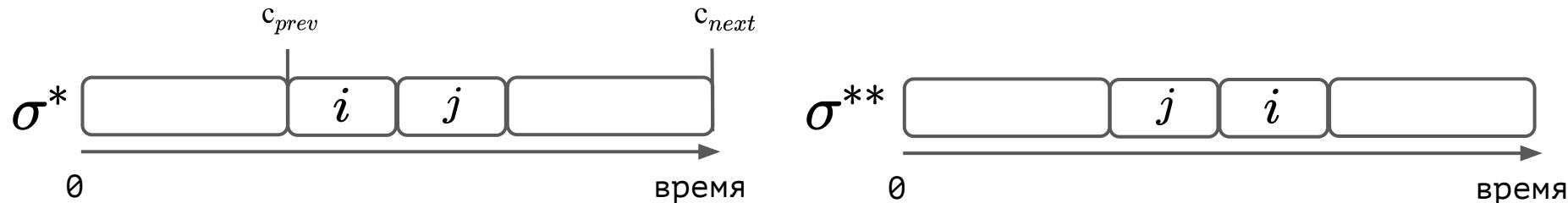
Как от такой перестановки поменяются времена окончания задач?

**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** ...

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .



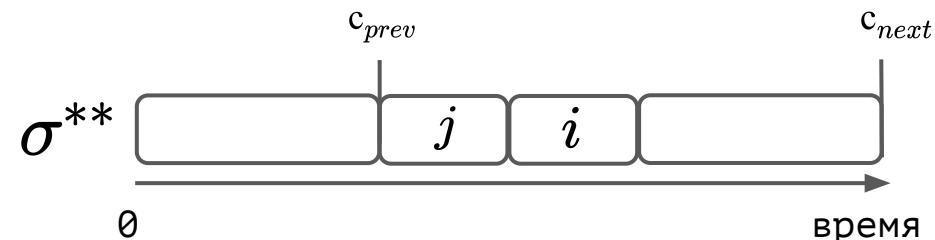
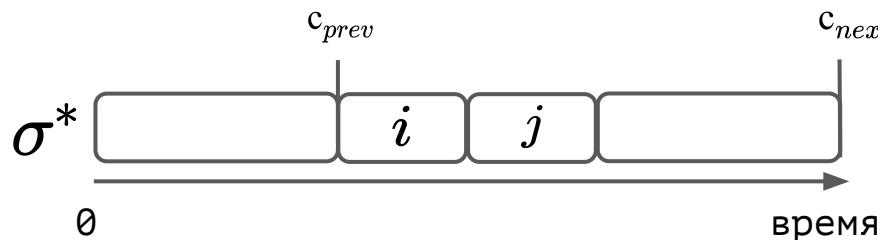
Как от такой перестановки поменяются времена окончания задач?

**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** ...

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .



Как от такой перестановки поменяются времена окончания задач?

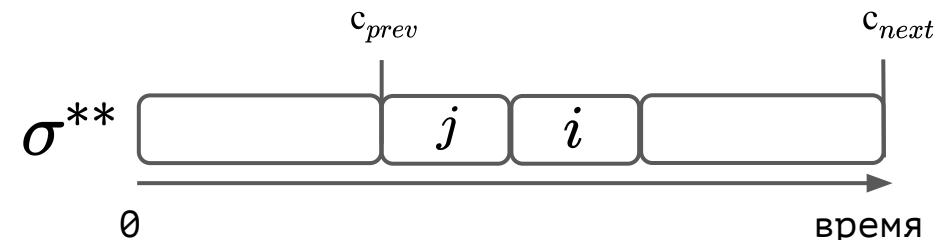
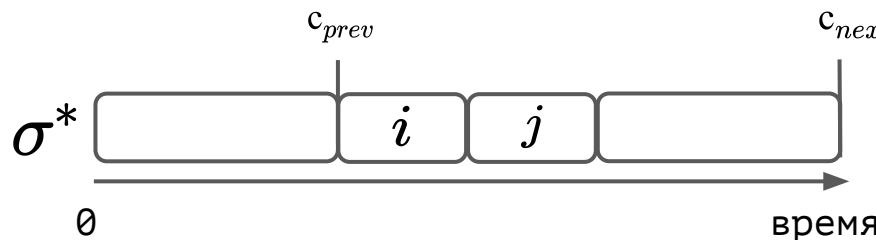
Для всех задач, кроме  $i$  и  $j$  времена окончания не изменятся.

**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** ...

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .



Как от такой перестановки поменяются времена окончания задач?

Для всех задач, кроме  $i$  и  $j$  времена окончания не изменятся.  
А для них изменение будет:  $c'_i \rightarrow c_i + l_j$  и  $c'_j \rightarrow c_j - l_i$

**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** ...

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

Тогда, если  $S(\sigma^*) = x + w_i * c_i + w_j * c_j + y$ , то

$$S(\sigma^{**}) = x + w_i * (c_i + l_j) + w_j * (c_j - l_i) + y$$

**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** ...

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

Тогда, если  $S(\sigma^*) = x + w_i * c_i + w_j * c_j + y$ , то

$$\begin{aligned} S(\sigma^{**}) &= x + w_i * (c_i + l_j) + w_j * (c_j - l_i) + y = \\ &= S(\sigma^*) + w_i * l_j - w_j * l_i \end{aligned}$$

**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** ...

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

Тогда, если  $S(\sigma^*) = x + w_i * c_i + w_j * c_j + y$ , то

$$\begin{aligned} S(\sigma^{**}) &= x + w_i * (c_i + l_j) + w_j * (c_j - l_i) + y = \\ &= S(\sigma^*) + w_i * l_j - w_j * l_i \end{aligned}$$

Но:  $i > j \Rightarrow \frac{w_i}{l_i} < \frac{w_j}{l_j}$

**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** ...

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

Тогда, если  $S(\sigma^*) = x + w_i * c_i + w_j * c_j + y$ , то

$$\begin{aligned} S(\sigma^{**}) &= x + w_i * (c_i + l_j) + w_j * (c_j - l_i) + y = \\ &= S(\sigma^*) + w_i * l_j - w_j * l_i \end{aligned}$$

Но:  $i > j \Rightarrow \frac{w_i}{l_i} < \frac{w_j}{l_j} \Rightarrow w_i l_j < w_j l_i$

**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** ...

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

Тогда, если  $S(\sigma^*) = x + w_i * c_i + w_j * c_j + y$ , то

$$\begin{aligned} S(\sigma^{**}) &= x + w_i * (c_i + l_j) + w_j * (c_j - l_i) + y = \\ &= S(\sigma^*) + w_i * l_j - w_j * l_i \end{aligned}$$

Но:  $i > j \Rightarrow \frac{w_i}{l_i} < \frac{w_j}{l_j} \Rightarrow w_i l_j < w_j l_i$

Все значения положительные  $\Rightarrow$  мы прибавили к  $S(\sigma^*)$  меньше, чем отняли  $\Rightarrow S(\sigma^{**}) < S(\sigma^*)$

**Допущение:** пусть для начала **нет равных** значений  $\frac{w_j}{l_j}$ .

**Доказательство:** ...

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

Тогда, если  $S(\sigma^*) = x + w_i * c_i + w_j * c_j + y$ , то

$$\begin{aligned} S(\sigma^{**}) &= x + w_i * (c_i + l_j) + w_j * (c_j - l_i) + y = \\ &= S(\sigma^*) + w_i * l_j - w_j * l_i \end{aligned}$$

Но:  $i > j \Rightarrow \frac{w_i}{l_i} < \frac{w_j}{l_j} \Rightarrow w_i l_j < w_j l_i$

Все значения положительные  $\Rightarrow$  мы прибавили к  $S(\sigma^*)$  меньше, чем отняли  $\Rightarrow S(\sigma^{**}) < S(\sigma^*)$

Получили противоречие с тем, что  $\sigma^*$  - оптимальное решение.  
А значит, оптимальным было  $\sigma$ .  $\square$

~~Допущение~~: пусть для начала ~~нет равных~~ значений  $\frac{w_j}{l_j}$ .

**Доказательство:** ...

Тогда имеем:  $\frac{w_0}{l_0} > \frac{w_1}{l_1} > \dots > \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

Тогда, если  $S(\sigma^*) = x + w_i * c_i + w_j * c_j + y$ , то

$$\begin{aligned} S(\sigma^{**}) &= x + w_i * (c_i + l_j) + w_j * (c_j - l_i) + y = \\ &= S(\sigma^*) + w_i * l_j - w_j * l_i \end{aligned}$$

Но:  $i > j \Rightarrow \frac{w_i}{l_i} < \frac{w_j}{l_j} \Rightarrow w_i l_j < w_j l_i$

Все значения положительные  $\Rightarrow$  мы прибавили к  $S(\sigma^*)$  меньше, чем отняли  $\Rightarrow S(\sigma^{**}) < S(\sigma^*)$

Получили противоречие с тем, что  $\sigma^*$  - оптимальное решение.  
А значит, оптимальным было  $\sigma$ .  $\square$

~~Допущение~~: пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство**: пусть в  $\sigma$  мы разрешали ничьи как угодно.

Тогда имеем:  $\frac{w_0}{l_0} \geq \frac{w_1}{l_1} \geq \dots \geq \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

Тогда, если  $S(\sigma^*) = x + w_i * c_i + w_j * c_j + y$ , то

$$\begin{aligned} S(\sigma^{**}) &= x + w_i * (c_i + l_j) + w_j * (c_j - l_i) + y = \\ &= S(\sigma^*) + w_i * l_j - w_j * l_i \end{aligned}$$

Но:  $i > j \Rightarrow \dots \Rightarrow w_i l_j - w_j l_i \leq 0$

~~Допущение~~: пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство**: пусть в  $\sigma$  мы разрешали ничьи как угодно.

Тогда имеем:  $\frac{w_0}{l_0} \geq \frac{w_1}{l_1} \geq \dots \geq \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

Тогда, если  $S(\sigma^*) = x + w_i * c_i + w_j * c_j + y$ , то

$$\begin{aligned} S(\sigma^{**}) &= x + w_i * (c_i + l_j) + w_j * (c_j - l_i) + y = \\ &= S(\sigma^*) + w_i * l_j - w_j * l_i \end{aligned}$$

Но:  $i > j \Rightarrow \dots \Rightarrow w_i l_j - w_j l_i \leq 0$ . Этого недостаточно, чтобы сразу сделать вывод, ведь получилось, что  $\sigma^{**}$  не хуже! Но может быть и таким же.

~~Допущение~~: пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство**: пусть в  $\sigma$  мы разрешали ничьи как угодно.

Тогда имеем:  $\frac{w_0}{l_0} \geq \frac{w_1}{l_1} \geq \dots \geq \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

Тогда, если  $S(\sigma^*) = x + w_i * c_i + w_j * c_j + y$ , то

$$\begin{aligned} S(\sigma^{**}) &= x + w_i * (c_i + l_j) + w_j * (c_j - l_i) + y = \\ &= S(\sigma^*) + w_i * l_j - w_j * l_i \end{aligned}$$

Но:  $i > j \Rightarrow \dots \Rightarrow w_i l_j - w_j l_i \leq 0$ . Этого недостаточно, чтобы сразу сделать вывод, ведь получилось, что  $\sigma^{**}$  не хуже! Но может быть и таким же.

Но что здесь произошло: в  $\sigma^*$  была локальная инверсия, мы ее исправили и получили результат не хуже.

~~Допущение~~: пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство**: пусть в  $\sigma$  мы разрешали ничьи как угодно.

Тогда имеем:  $\frac{w_0}{l_0} \geq \frac{w_1}{l_1} \geq \dots \geq \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

...

Но:  $i > j \Rightarrow \dots \Rightarrow w_i l_j - w_j l_i \leq 0$ . Этого недостаточно, чтобы сразу сделать вывод, ведь получилось, что  $\sigma^{**}$  не хуже! Но может быть и таким же.

Но что здесь произошло: в  $\sigma^*$  была локальная инверсия, мы ее исправили и получили результат не хуже.

А сколько может быть всего инверсий? (даже не локальных)

~~Допущение~~: пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство**: пусть в  $\sigma$  мы разрешали ничьи как угодно.

Тогда имеем:  $\frac{w_0}{l_0} \geq \frac{w_1}{l_1} \geq \dots \geq \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

...

Но:  $i > j \Rightarrow \dots \Rightarrow w_i l_j - w_j l_i \leq 0$ . Этого недостаточно, чтобы сразу сделать вывод, ведь получилось, что  $\sigma^{**}$  не хуже! Но может быть и таким же.

Но что здесь произошло: в  $\sigma^*$  была локальная инверсия, мы ее исправили и получили результат не хуже. При этом новых инверсий мы не создали!!!

~~Допущение~~: пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство**: пусть в  $\sigma$  мы разрешали ничьи как угодно.

Тогда имеем:  $\frac{w_0}{l_0} \geq \frac{w_1}{l_1} \geq \dots \geq \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

...

Но:  $i > j \Rightarrow \dots \Rightarrow w_i l_j - w_j l_i \leq 0$ . Этого недостаточно, чтобы сразу сделать вывод, ведь получилось, что  $\sigma^{**}$  не хуже! Но может быть и таким же.

Но что здесь произошло: в  $\sigma^*$  была локальная инверсия, мы ее исправили и получили результат не хуже.

А сколько может быть всего инверсий?  $\binom{N}{2}$

~~Допущение~~: пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство**: пусть в  $\sigma$  мы разрешали ничьи как угодно.

Тогда имеем:  $\frac{w_0}{l_0} \geq \frac{w_1}{l_1} \geq \dots \geq \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

...

Но:  $i > j \Rightarrow \dots \Rightarrow w_i l_j - w_j l_i \leq 0$ . Этого недостаточно, чтобы сразу сделать вывод, ведь получилось, что  $\sigma^{**}$  не хуже! Но может быть и таким же.

Но что здесь произошло: в  $\sigma^*$  была локальная инверсия, мы ее исправили и получили результат не хуже.

А сколько может быть всего инверсий?  $\binom{N}{2}$

Т.е. за  $\binom{N}{2}$  шагов мы точно избавимся от инверсий без ухудшения результата!

~~Допущение~~: пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство**: пусть в  $\sigma$  мы разрешали ничьи как угодно.

Тогда имеем:  $\frac{w_0}{l_0} \geq \frac{w_1}{l_1} \geq \dots \geq \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

...

Но:  $i > j \Rightarrow \dots \Rightarrow w_i l_j - w_j l_i \leq 0$ . Этого недостаточно, чтобы сразу сделать вывод, ведь получилось, что  $\sigma^{**}$  не хуже! Но может быть и таким же.

Но что здесь произошло: в  $\sigma^*$  была локальная инверсия, мы ее исправили и получили результат не хуже.

А сколько может быть всего инверсий?  $\binom{N}{2}$

Т.е. за  $\binom{N}{2}$  шагов мы точно избавимся от инверсий без ухудшения результата! При этом получим  $\sigma$ !

~~Допущение~~: пусть для начала нет равных значений  $\frac{w_j}{l_j}$ .

**Доказательство**: пусть в  $\sigma$  мы разрешали ничьи как угодно.

Тогда имеем:  $\frac{w_0}{l_0} \geq \frac{w_1}{l_1} \geq \dots \geq \frac{w_{n-1}}{l_{n-1}}$

В  $\sigma^*$  есть пара последовательных работ  $i, j$  такая, что  $i > j$ .

...

Но:  $i > j \Rightarrow \dots \Rightarrow w_i l_j - w_j l_i \leq 0$ . Этого недостаточно, чтобы сразу сделать вывод, ведь получилось, что  $\sigma^{**}$  не хуже! Но может быть и таким же.

Но что здесь произошло: в  $\sigma^*$  была локальная инверсия, мы ее исправили и получили результат не хуже.

А сколько может быть всего инверсий?  $\binom{N}{2}$

Т.е. за  $\binom{N}{2}$  шагов мы точно избавимся от инверсий без ухудшения результата! При этом получим  $\sigma$ !

Значит он и есть искомый.

# Оптимальное кэширование

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

a			
---	--	--	--

Ввод: a

1 cache-miss

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

a	b		
---	---	--	--

Ввод: a b

2 cache-misses

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

a	b	c	d
---	---	---	---

Ввод: a b c d

4 cache-misses

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

a	b	c	d
---	---	---	---

Кого выкинуть из кэша?

Ввод: a b c d e

5 cache-misses

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

a	b	c	d
---	---	---	---

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Ввод: a b c d e

5 cache-misses

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

e	b	c	d
---	---	---	---

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Ввод: a b c d e

5 cache-misses

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

e	b	c	d
---	---	---	---

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Ввод: a b c d e f

6 cache-misses

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

e	f	c	d
---	---	---	---

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Ввод: a b c d e f

6 cache-misses

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

e	f	c	d
---	---	---	---

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Ввод: a b c d e f a

Можно ли было этого избежать?

7 cache-misses

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

e	f	a	d
---	---	---	---

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Ввод: a b c d e f a b

Можно ли было этого избежать?

8 cache-misses

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

e	f	a	b
---	---	---	---

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Ввод: a b c d e f a b

Можно ли было этого избежать?

8 cache-misses

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

c	d	a	b
---	---	---	---

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Ввод: a b c d e f a b c d

Можно ли было этого избежать?

10 cache-misses

# Оптимальное кэширование

**Задача:** есть кэш для  $k$  элементов и алфавит из  $n$  элементов.  $k < n$ . На вход подается последовательность значений из этого алфавита.

Предложите оптимальную стратегию кэширования (минимизирующую количество cache misses)

Кэш: 

c	d	a	b
---	---	---	---

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Ввод: a b c d e f a b c d

А это точно была хорошая идея?

**10** cache-misses



# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Еще его называют  
алгоритмом ясновидящего  
(*clairvoyant algorithm*)



# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

a	b	c	d
---	---	---	---

Ввод: a b c d e f a b c d



Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

4 cache-misses

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

a	b	c	d
---	---	---	---

Ввод: a b c d e f a b c d



Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

5 cache-misses

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

a	b	c	d
---	---	---	---

Ввод: a b c d e f a b c d



5 cache-misses

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Нет, того, кто будет использоваться позже всего в будущем!

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

a	b	c	e
---	---	---	---

Ввод: a b c d e f a b c d



5 cache-misses

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Нет, того, кто будет использоваться позже всего в будущем!

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

a	b	c	e
---	---	---	---

Ввод: a b c d e f a b c d



Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

6 cache-misses

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

a	b	c	e
---	---	---	---

Ввод: a b c d e f a b c d



6 cache-misses

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Нет, того, кто будет использоваться позже всего в будущем!

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

a	b	c	f
---	---	---	---

Ввод: a b c d e f a b c d



6 cache-misses

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Нет, того, кто будет использоваться позже всего в будущем!

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

a	b	c	f
---	---	---	---

Ввод: a b c d e f a b c d



Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

6 cache-misses

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

a	b	c	f
---	---	---	---

Ввод: a b c d e f a b c d



Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

6 cache-misses

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

a	b	c	f
---	---	---	---

Ввод: a b c d e f a b c d



Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

6 cache-misses

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

a	b	c	f
---	---	---	---

Ввод: a b c d e f a b c d



Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

7 cache-misses

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

d	b	c	f
---	---	---	---

Ввод: a b c d e f a b c d

Кого выкинуть из кэша?

Последнего используемого? (LRU-cache)

Нет, того, кто будет использоваться позже всего в будущем!

7 cache-misses

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Кэш: 

d	b	c	f
---	---	---	---

Ввод: a b c d e f a b c d

Кого выкинуть из кэша?  
Последнего используемого? (LRU-cache)

7 cache-misses < 10 cache-misses

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Теорема: алгоритм Белади дает оптимальное решение задачи кэширования.

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Теорема: алгоритм Белади дает оптимальное решение задачи кэширования.

Доказательство совсем не очевидное! Можете найти его здесь: <https://sites.cs.ucsb.edu/~suri/ccs130a/Caching.pdf>

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Теорема: алгоритм Белади дает оптимальное решение задачи кэширования.

Доказательство совсем не очевидное! Можете найти его здесь: <https://sites.cs.ucsb.edu/~suri/ccs130a/Caching.pdf>

Зачем такой алгоритм нужен?

# Оптимальное кэширование

Алгоритм Белади. Используется следующее **жадное** правило вытеснения из кэша: удаляется элемент, который понадобится в будущем **позже всего**.

Теорема: алгоритм Белади дает оптимальное решение задачи кэширования.

Доказательство совсем не очевидное! Можете найти его здесь: <https://sites.cs.ucsb.edu/~suri/ccs130a/Caching.pdf>

Зачем такой алгоритм нужен?

Это **верхняя граница** качества алгоритмов кэширования.

## Мини-задача #32 (1 балл)

В каждом элементе массива длина максимального прыжка.  
Начинаем с нулевого элемента, возможно ли пропрыгать от  
до его конца массива?

<https://leetcode.com/problems/jump-game>

## Мини-задача #33 (1 балл)

Вы брокер, который может покупать или продавать акции  
(у вас на руках только одна позиция). Дан массив цен на  
акции по дням, найти максимальную прибыль.

<https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/>