Мини-задача #39 (2 балла)

Решить задачу распознавания строки по регулярному выражению через динамическое программирование.

https://leetcode.com/problems/regular-expression-matching

Мини-задача #40 (2 балла)

Посчитайте минимальное стоимость слияния камней в одну кучу через динамическое программирование.

https://leetcode.com/problems/minimum-cost-to-merge-stones

Алгоритмы и структуры данных

Оптимальные BST, алгоритм Нидлмана-Вунша

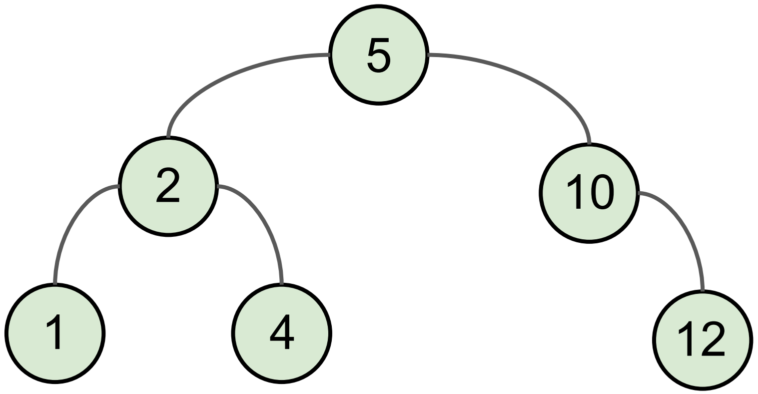
Бинарные деревья поиска

Бинарное дерево **поиска** $\begin{cases} \emptyset \\ \{\mathtt{value},\ \mathtt{left},\ \mathtt{right}\} \end{cases}$

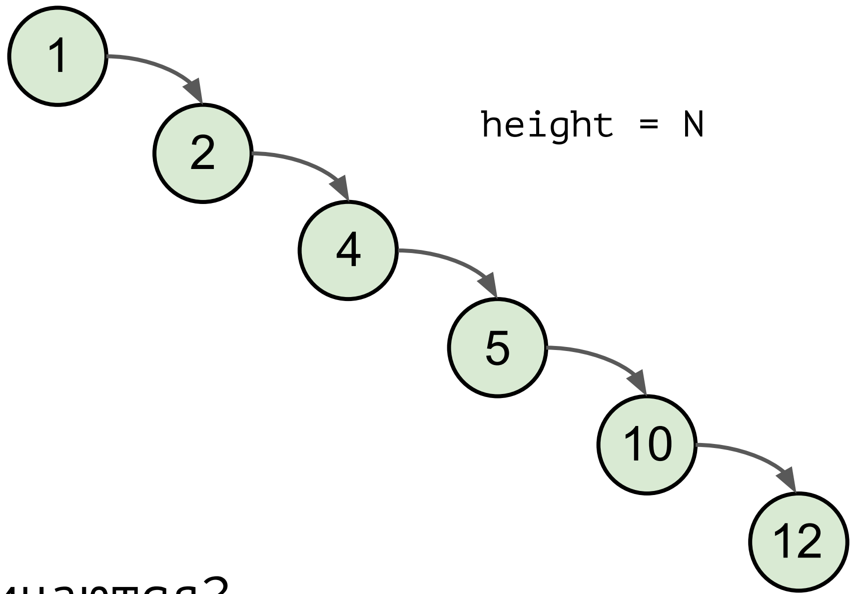
BST

где value — значение в вершине, left и right также являются бинарными деревьями ***поиска***, и <u>каждое</u> значение из left <u>меньше</u> value, а <u>каждое</u> значение из right <u>больше</u> value.

Бинарные деревья поиска



height = logN



Чем эти деревья поиска отличаются?

Высотой! Второе намного выше.

Бинарные деревья поиска

Операции:

```
1.  find(value)     -> O(height)
2.  select(i)       -> O(height)
3.  min/max         -> O(height)
4.  pred/succ(ptr)  -> O(height)
5.  rank(value)     -> O(height)
6.  вывод в пор.    -> O(N)
    возрастания
---------------------------------

7.  insert(value)   -> O(height)
8.  remove(value)   -> O(height)
```

Назревает проблема!

Обещали логарифм, а дают какой-то O(height)

Значит нам нужны такие BST, чтобы высота у дерева всегда была logN

АВЛ-деревья

Операции:

1. find(value)   -> O(logN)
2. select(i)   -> O(logN)
3. min/max   -> O(logN)
4. pred/succ(ptr)   -> O(logN)
5. rank(value)   -> O(logN)
6. вывод в пор. возрастания -> O(N)

---

7. insert(value)   -> O(logN)
8. remove(value)   -> O(logN)

В АВЛ-дереве действительно высота всегда порядка logN

И все операции продолжают работать за логарифм!

Бинарные деревья поиска

Казалось бы, что еще здесь можно обсуждать?

Бинарные деревья поиска

Пусть теперь у нас есть дополнительная информация: с какой частотой ищется тот или иной элемент.

Бинарные деревья поиска

Пусть теперь у нас есть дополнительная информация: с какой частотой ищется тот или иной элемент.

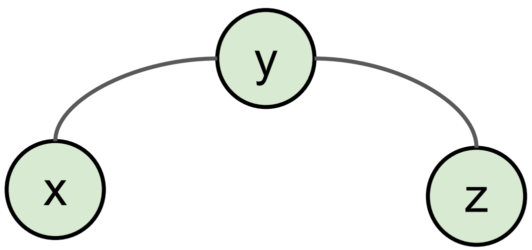
Пример: пусть есть три значения $x < y < z$. И есть информация, что: 80% запросов ищут x, 10% — y, 10% — z.

Бинарные деревья поиска

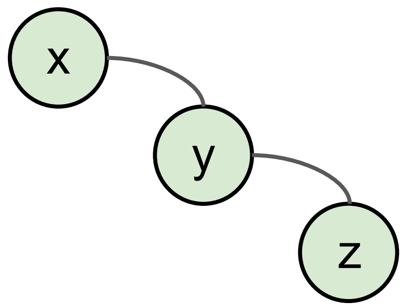
Пусть теперь у нас есть дополнительная информация: с какой частотой ищется тот или иной элемент.

Пример: пусть есть три значения x < y < z. И есть информация, что: 80% запросов ищут x, 10% — y, 10% — z.

Какое BST лучше?



сбалансированное

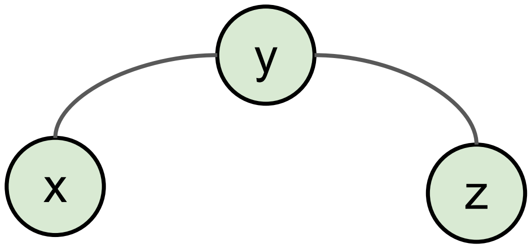


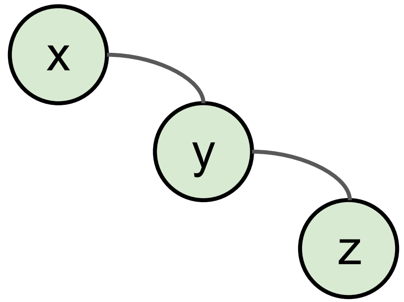
Бинарные деревья поиска

Пусть теперь у нас есть дополнительная информация: с какой частотой ищется тот или иной элемент.

Пример: пусть есть три значения x < y < z. И есть информация, что: 80% запросов ищут x, 10% — y, 10% — z.

Какое BST лучше? Введем случайную величину S: количество просмотренных узлов при запросе. И посчитаем ее мат. ожидание!





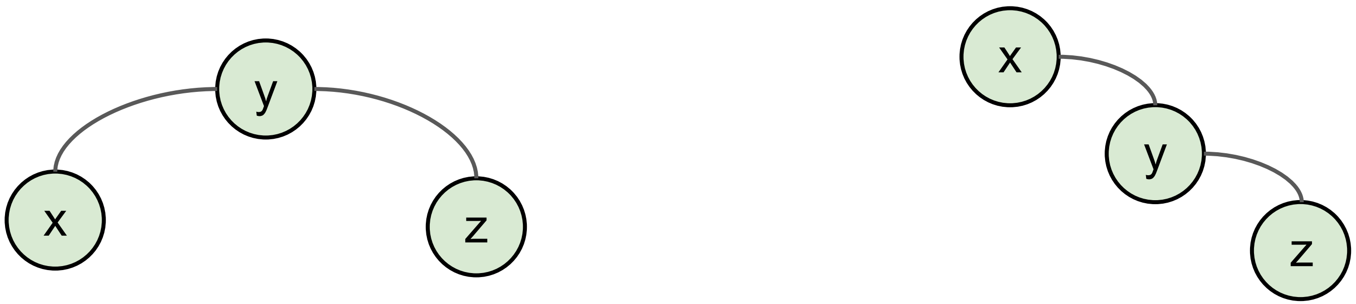
Бинарные деревья поиска

Пусть теперь у нас есть дополнительная информация: с какой частотой ищется тот или иной элемент.

Пример: пусть есть три значения x < y < z. И есть информация, что: 80% запросов ищут x, 10% — y, 10% — z.

Какое BST лучше? Введем случайную величину S: количество просмотренных узлов при запросе. И посчитаем ее мат. ожидание!

$$E[S] = 0.8 * 2 + 0.1 * 1 + 0.1 * 2 = 1.9$$



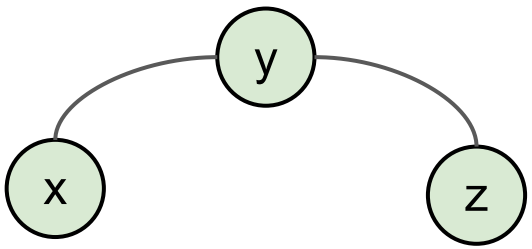
Бинарные деревья поиска

Пусть теперь у нас есть дополнительная информация: с какой частотой ищется тот или иной элемент.

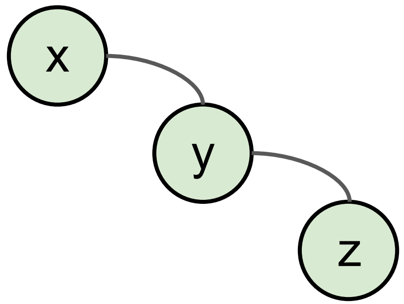
Пример: пусть есть три значения x < y < z. И есть информация, что: 80% запросов ищут x, 10% — y, 10% — z.

Какое BST лучше? Введем случайную величину S: количество просмотренных узлов при запросе. И посчитаем ее мат. ожидание!

$$E[S] = 0.8 * 2 + 0.1 * 1 + 0.1 * 2 = 1.9$$



$$E[S] = 0.8 * 1 + 0.1 * 2 + 0.1 * 3 = 1.3$$



Задача

Дано: пусть есть элементы $v_1, v_2, \ldots, v_n : v_1 < v_2 < \ldots v_n$.
И есть вероятности $p_1, p_2, \ldots, p_n$, что соответствующий элемент будут искать в каждом запросе.

Задача

Дано: пусть есть элементы $v_1, v_2, \ldots, v_n : v_1 < v_2 < \ldots v_n$.
И есть вероятности $p_1, p_2, \ldots, p_n$, что соответствующий элемент будут искать в каждом запросе.

Нужно: построить такое BST (важно, что это все еще дерево поиска) T, что $C(T) = \sum\limits_{i \in [1,n]} p_i * F_i(T)$ - будет минимальным, где $F_i(T)$ - количество посещенных вершин при поиске вершины v_i в дереве T.

Задача

Дано: пусть есть элементы $v_1, v_2, \ldots, v_n : v_1 < v_2 < \ldots v_n$.
И есть вероятности $p_1, p_2, \ldots, p_n$, что соответствующий элемент будут искать в каждом запросе.

Нужно: построить такое BST (важно, что это все еще дерево поиска) T, что $C(T) = \sum\limits_{i \in [1,n]} p_i * F_i(T)$ - будет минимальным, где $F_i(T)$ - количество посещенных вершин при поиске вершины v_i в дереве T.

Замечание: если $\sum\limits_{i \in [1,n]} p_i = 1$, то это буквально мат. ожидание. Но мы можем ослабить это ограничение (тогда речь будет идти о взвешенном времени поиска вершин).

Задача о поиске оптимального BST

Дано: пусть есть элементы $v_1, v_2, \ldots, v_n : v_1 < v_2 < \ldots v_n$.
И есть вероятности $p_1, p_2, \ldots, p_n$, что соответствующий элемент будут искать в каждом запросе.

Нужно: построить такое BST (важно, что это все еще дерево поиска) T, что $C(T) = \sum\limits_{i \in [1,n]} p_i * F_i(T)$ - будет минимальным, где $F_i(T)$ - количество посещенных вершин при поиске вершины v_i в дереве T.

Замечание: если $\sum\limits_{i \in [1,n]} p_i = 1$, то это буквально мат. ожидание. Но мы можем ослабить это ограничение (тогда речь будет идти о взвешенном времени поиска вершин).

Задача о поиске оптимального BST

Как решать? Может жадность?

Задача о поиске оптимального BST

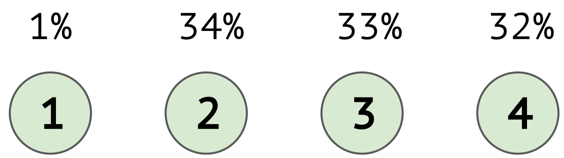
Как решать? Может жадность?

1) В корень - самую вероятную вершину (и дальше продолжаем в том же духе).

Задача о поиске оптимального BST

Как решать? Может жадность?

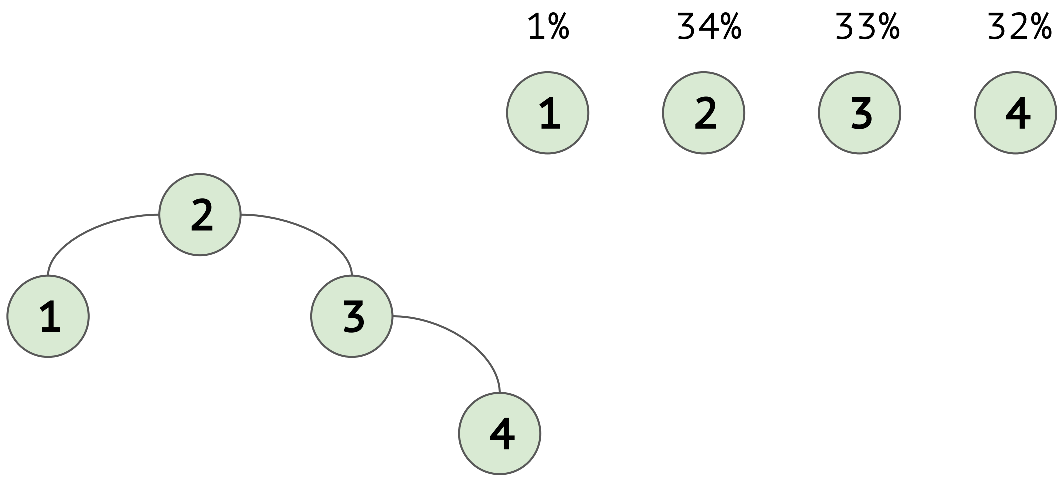
1) В корень - самую вероятную вершину (и дальше продолжаем в том же духе).



Задача о поиске оптимального BST

Как решать? Может жадность?

1) В корень - самую вероятную вершину (и дальше продолжаем в том же духе).

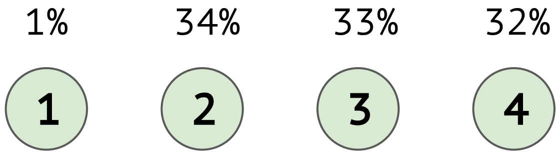


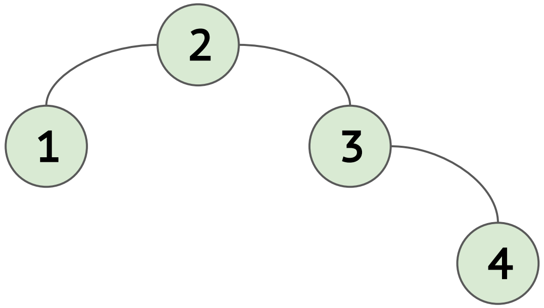
$\text{C}(T) = 0.34 * 1 + 0.01 * 2 + 0.33 * 2 + 0.32 * 3 = 1.98$

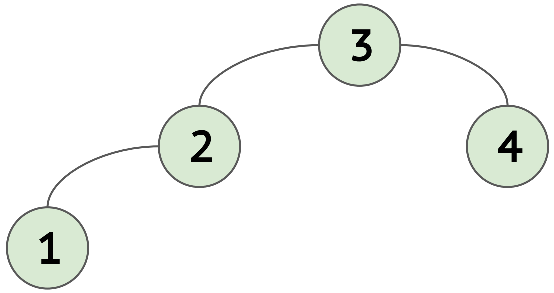
Задача о поиске оптимального BST

Как решать? Может жадность?

1) В корень - самую вероятную вершину (и дальше продолжаем в том же духе).







$\mathrm{C}(T) = 0.34 * 1 + 0.01 * 2 + 0.33 * 2 + 0.32 * 3 = 1.98$

$\mathrm{C}(T) = 0.33 * 1 + 0.34 * 2 + 0.01 * 3 + 0.32 * 2 = 1.66$

Задача о поиске оптимального BST

Как решать? Может жадность?

1) В корень - самую вероятную вершину (и дальше продолжаем в том же духе).

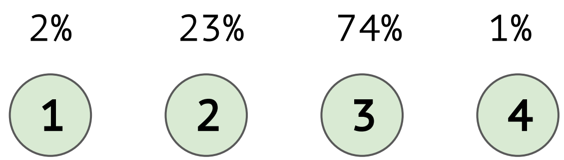
2) В листья - самые маловероятные вершины (и дальше поднимаемся).

Задача о поиске оптимального BST

Как решать? Может жадность?

1) В корень - самую вероятную вершину (и дальше продолжаем в том же духе).

2) В листья - самые маловероятные вершины (и дальше поднимаемся).

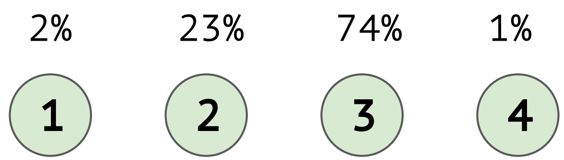


Задача о поиске оптимального BST

Как решать? Может жадность?

1) В корень - самую вероятную вершину (и дальше продолжаем в том же духе).

2) В листья - самые маловероятные вершины (и дальше поднимаемся).



Задача о поиске оптимального BST

Но вообще, искать правильный корень - звучит перспективно.

Задача о поиске оптимального BST

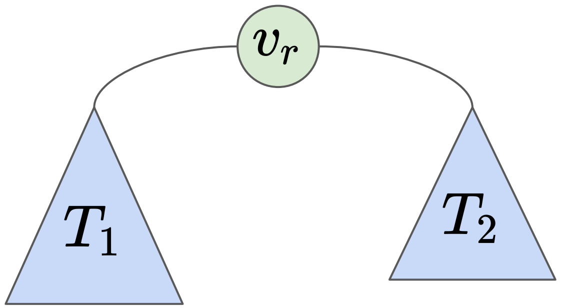
Но вообще, искать правильный корень - звучит перспективно.

Пусть мы знаем правильный для **оптимального** BST корень v_r. Что можете сказать про поддеревья?

Задача о поиске оптимального BST

Но вообще, искать правильный корень - звучит перспективно.

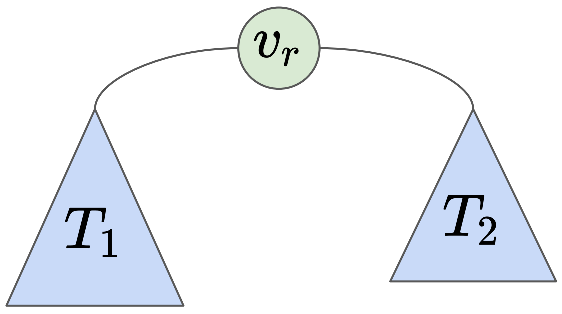
Пусть мы знаем правильный для **оптимального** BST корень v_r. Что можете сказать про поддеревья T_1 и T_2?



Задача о поиске оптимального BST

Но вообще, искать правильный корень - звучит перспективно.

Пусть мы знаем правильный для **оптимального** BST корень v_r. Что можете сказать про поддеревья T_1 и T_2?

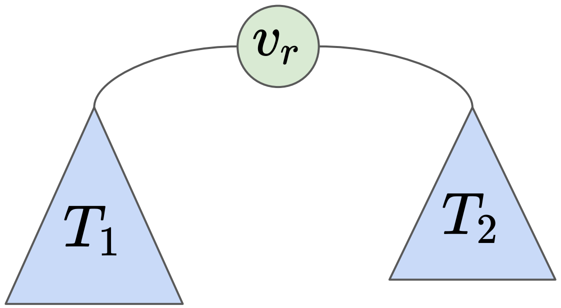


1) Они являются BST

Задача о поиске оптимального BST

Но вообще, искать правильный корень - звучит перспективно.

Пусть мы знаем правильный для **оптимального** BST корень v_r . Что можете сказать про поддеревья T_1 и T_2 ?



1) Они являются BST,

2) В левом вершины $v_1, v_2, \ldots, v_{r-1}$
   В правом вершины $v_{r+1}, v_{r+2}, \ldots, v_n$

Задача о поиске оптимального BST

Дано: пусть есть элементы $v_1, v_2, \ldots, v_n : v_1 < v_2 < \ldots v_n$.
И есть вероятности $p_1, p_2, \ldots, p_n$, что соответствующий элемент будут искать в каждом запросе.

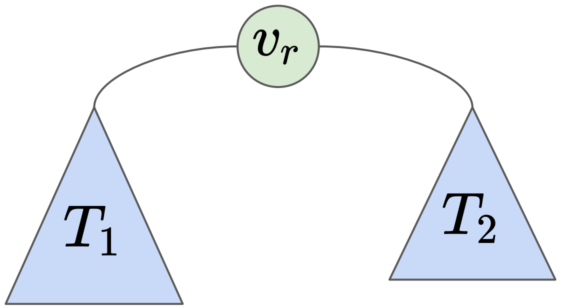
Нужно: построить такое BST (важно, что это все еще дерево поиска) T, что $C(T) = \sum_{i \in [1,n]} p_i * F_i(T)$ - будет минимальным, где $F_i(T)$ - количество посещенных вершин при поиске вершины v_i в дереве T.

Замечание: если $\sum_{i \in [1,n]} p_i = 1$, то это буквально мат. ожидание. Но мы можем ослабить это ограничение (тогда речь будет идти о взвешенном времени поиска вершин).

Задача о поиске оптимального BST

Но вообще, искать правильный корень - звучит перспективно.

Пусть мы знаем правильный для **оптимального** BST корень v_r. Что можете сказать про поддеревья T_1 и T_2?



1) Они являются BST,

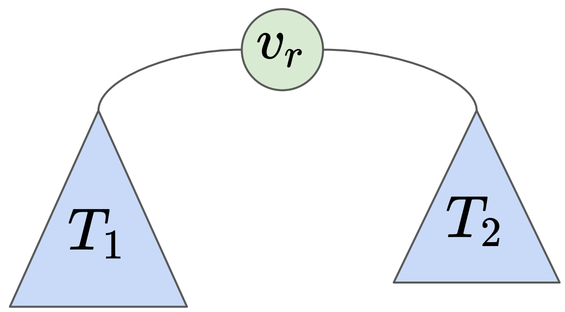
2) В левом вершины $v_1, v_2, \ldots, v_{r-1}$
В правом вершины $v_{r+1}, v_{r+2}, \ldots, v_n$

3) Они являются оптимальными BST для этих наборов вершин

Задача о поиске оптимального BST

T_1 и T_2 являются оптимальными BST для набор вершин $v_1, v_2, \ldots, v_{r-1}$ и $v_{r+1}, v_{r+2}, \ldots, v_n$ соответственно.

Док-во: предположим, что T_1 не является оптимальным. Т.е. некое BST T_1^* из вершин $v_1, v_2, \ldots, v_{r-1}$: $C(T_1^*) < C(T_1)$.

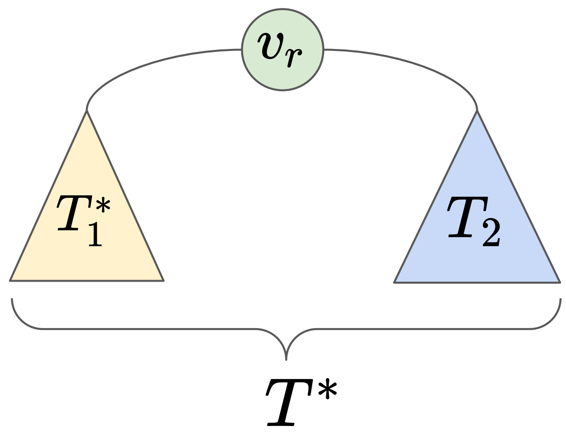


Задача о поиске оптимального BST

T_1 и T_2 являются оптимальными BST для набор вершин $v_1, v_2, \ldots, v_{r-1}$ и $v_{r+1}, v_{r+2}, \ldots, v_n$ соответственно.

Док-во: предположим, что T_1 не является оптимальным. Т.е. некое BST T_1^* из вершин $v_1, v_2, \ldots, v_{r-1}$: $C(T_1^*) < C(T_1)$.

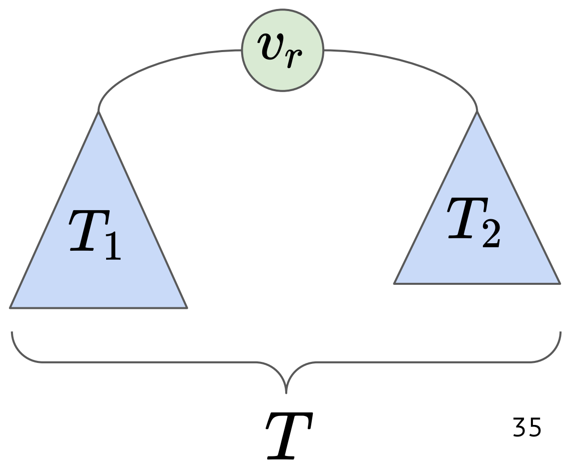
Теперь покажем, что в таком случае, дерево T^* будет более оптимальным, чем исходное (что даст нам противоречие).



Задача о поиске оптимального BST

$$C(T) = \sum_{i \in [1,n]} p_i * F_i(T) =$$

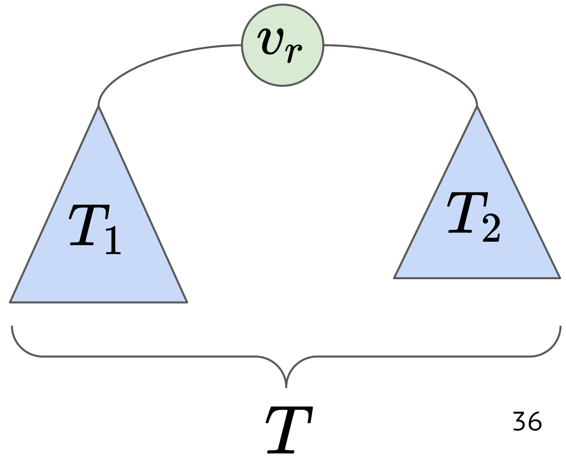
$F_i(T)$ - количество посещенных вершин в T при поиске вершины i.



Задача о поиске оптимального BST

$$C(T) = \sum_{i\in[1,n]} p_i * F_i(T) = p_r * 1 + \sum_{i\in[1,r-1]} p_i F_i(T) + \sum_{i\in[r+1,n]} p_i F_i(T) =$$

$F_i(T)$ - количество посещенных вершин в T при поиске вершины i.

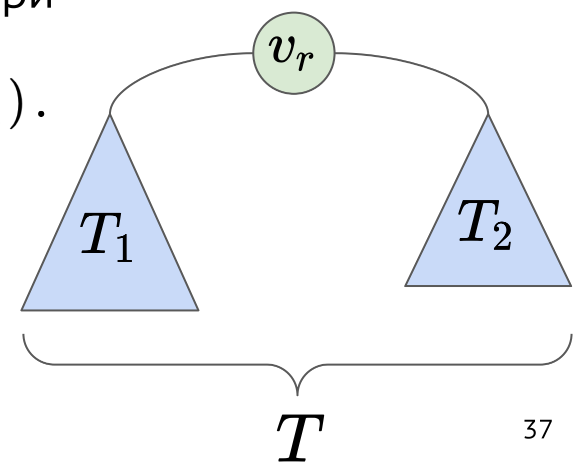


Задача о поиске оптимального BST

$$C(T) = \sum_{i \in [1,n]} p_i * F_i(T) = p_r * 1 + \sum_{i \in [1,r-1]} p_i F_i(T) + \sum_{i \in [r+1,n]} p_i F_i(T) =$$

$F_i(T)$ - количество посещенных вершин в T при поиске вершины i.

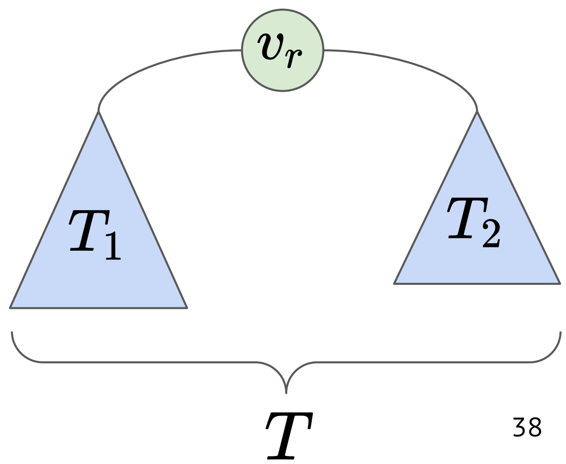
Для всех вершин, кроме v_r количество шагов при поиске в поддереве меньше на единицу, чем поиск в основном дереве. Т.е. $F_i(T) = 1 + F_i(T_1)$.



Задача о поиске оптимального BST

$$C(T) = \sum_{i\in[1,n]} p_i * F_i(T) = p_r * 1 + \sum_{i\in[1,r-1]} p_i F_i(T) + \sum_{i\in[r+1,n]} p_i F_i(T) =$$

$$= \sum_{i\in[1,n]} p_i + \sum_{i\in[1,r-1]} p_i F_i(T_1) + \sum_{i\in[r+1,n]} p_i F_i(T_2) =$$



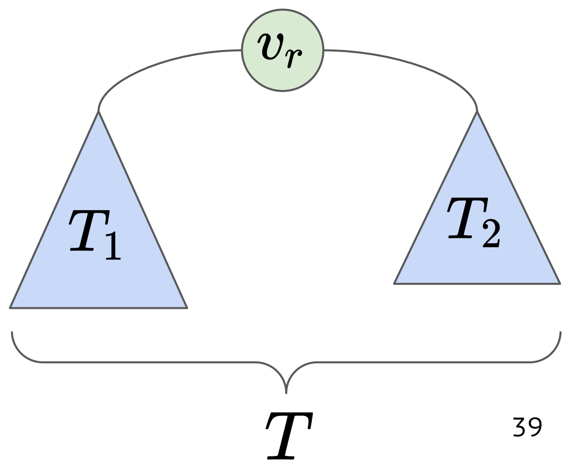
Задача о поиске оптимального BST

$$C(T) = \sum_{i\in[1,n]} p_i * F_i(T) = p_r * 1 + \sum_{i\in[1,r-1]} p_i F_i(T) + \sum_{i\in[r+1,n]} p_i F_i(T) =$$

$$= \sum_{i\in[1,n]} p_i + \sum_{i\in[1,r-1]} p_i F_i(T_1) + \sum_{i\in[r+1,n]} p_i F_i(T_2) =$$

$$= \sum_{i\in[1,n]} p_i + C(T_1) + C(T_2)$$

не зависит от
поддеревьев!

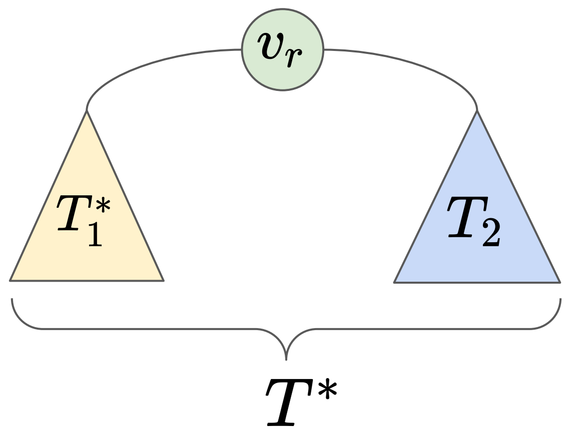


Задача о поиске оптимального BST

$$C(T) = \sum_{i \in [1,n]} p_i * F_i(T) = p_r * 1 + \sum_{i \in [1,r-1]} p_i F_i(T) + \sum_{i \in [r+1,n]} p_i F_i(T) =$$

$$= \sum_{i \in [1,n]} p_i + \sum_{i \in [1,r-1]} p_i F_i(T_1) + \sum_{i \in [r+1,n]} p_i F_i(T_2) =$$

$$= \sum_{i \in [1,n]} p_i + C(T_1) + C(T_2)$$



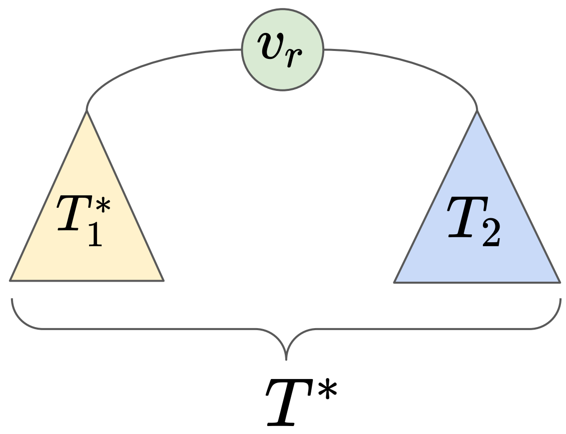
Аналогично:

$$C(T^*) = \sum_{i \in [1,n]} p_i + C(T_1^*) + C(T_2)$$

Задача о поиске оптимального BST

$$C(T) = \sum_{i\in[1,n]} p_i + C(T_1) + C(T_2)$$

$$C(T^*) = \sum_{i\in[1,n]} p_i + C(T_1^*) + C(T_2)$$



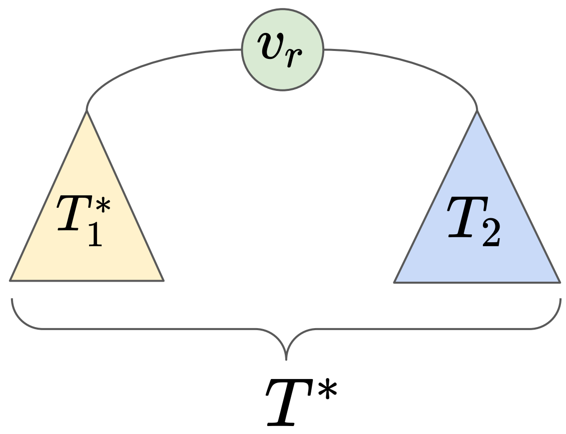
Задача о поиске оптимального BST

$$C(T) = \sum_{i \in [1,n]} p_i + C(T_1) + C(T_2)$$

$$C(T^*) = \sum_{i \in [1,n]} p_i + C(T_1^*) + C(T_2) \quad \Longrightarrow \quad C(T^*) < C(T)$$

противоречие с
оптимальностью T

$$C(T_1^*) < C(T_1)$$



Задача о поиске оптимального BST

$$C(T) = \sum_{i \in [1,n]} p_i + C(T_1) + C(T_2)$$

$$C(T^*) = \sum_{i \in [1,n]} p_i + C(T_1^*) + C(T_2)$$

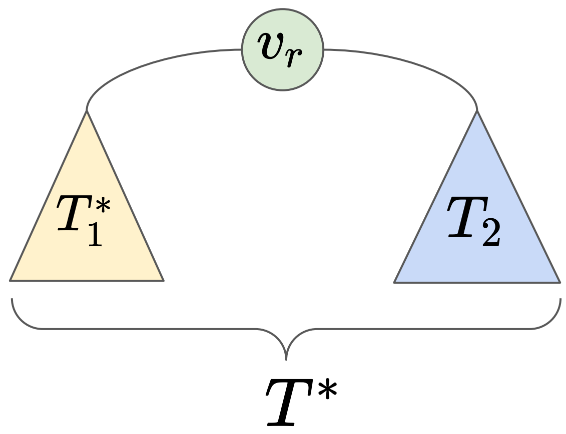
$$\Longrightarrow$$

противоречие с
оптимальностью T

$$C(T^*) < C(T)$$

$$C(T_1^*) < C(T_1)$$

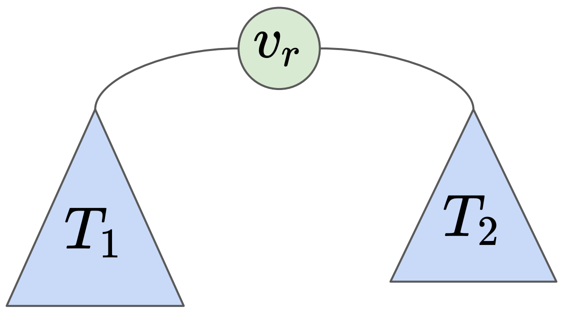
Аналогичные рассуждения с T2. $\square$



Задача о поиске оптимального BST

Но вообще, искать правильный корень - звучит перспективно.

Пусть мы знаем правильный для **оптимального** BST корень v_r . Что можете сказать про поддеревья T_1 и T_2 ?



1) Они являются BST,

2) В левом вершины $v_1, v_2, \ldots, v_{r-1}$
   В правом вершины $v_{r+1}, v_{r+2}, \ldots, v_n$

3) Они являются оптимальными BST для этих наборов вершин $\square$

Оптимальная подструктура

Но вообще, искать правильный корень - звучит перспективно.

Пусть мы знаем правильный для **оптимального** BST корень v_r . Тогда T_1 и T_2 - оптимальные BST для $v_1, v_2, \ldots, v_{r-1}$ и $v_{r+1}, v_{r+2}, \ldots, v_n$ соответственно.

Осталось найти подходящий r. Для этого будем перебирать все варианты и брать тот, который минимизирует C(T).

Оптимальная подструктура

Но вообще, искать правильный корень - звучит перспективно.

Пусть мы знаем правильный для **оптимального** BST корень v_r . Тогда T_1 и T_2 - оптимальные BST для $v_1, v_2, ..., v_{r-1}$ и $v_{r+1}, v_{r+2}, ..., v_n$ соответственно.

Осталось найти подходящий r. Для этого будем перебирать все варианты и брать тот, который минимизирует C(T). При этом зависим мы только от задач меньшей размерности: от префикса и суффикса (элементы слева и справа от r)

Оптимальная подструктура

Но вообще, искать правильный корень - звучит перспективно.

Пусть мы знаем правильный для **оптимального** BST корень v_r . Тогда T_1 и T_2 - оптимальные BST для $v_1, v_2, \ldots, v_{r-1}$ и $v_{r+1}, v_{r+2}, \ldots, v_n$ соответственно.

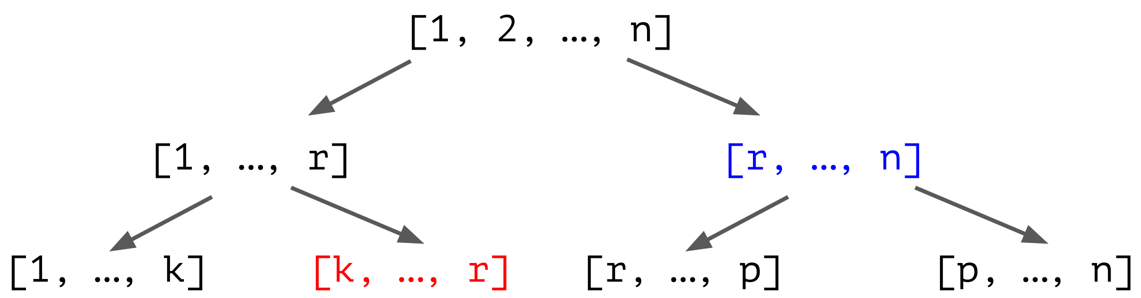
Осталось найти подходящий r. Для этого будем перебирать все варианты и брать тот, который минимизирует C(T). При этом зависим мы только от задач меньшей размерности: от префикса и суффикса (элементы слева и справа от r)

Глобально какие именно подзадачи мы будем рассматривать?
Для какого подмножества $S \subseteq \{v_1, v_2, \ldots, v_n\}$ будем решать подзадачу?

Осталось найти подходящий r. Для этого будем перебирать все варианты и брать тот, который минимизирует C(T). При этом зависим мы только от задач меньшей размерности: от префикса и суффикса (элементы слева и справа от r)

Глобально какие именно подзадачи мы будем рассматривать?
Для какого подмножества $S \subseteq \{v_1, v_2, ..., v_n\}$ будем решать подзадачу?

Хотя на каждом уровне рекурсии мы смотрим только на префиксы и суффиксы, глобально мы рассмотрим промежутки $[v_i, v_{i+1}, ..., v_j]$ для индексов $1 \leq i < j \leq n$



Рекуррентное соотношение

Введем для $1 \leq i < j \leq n$ обозначение C_{ij} - взвешенное время поиска элемента в оптимальном BST из вершин $\{v_i, v_{i+1}, \ldots, v_j\}$.

Рекуррентное соотношение

Введем для $1 \le i < j \le n$ обозначение C_{ij} - взвешенное время поиска элемента в оптимальном BST из вершин $\{v_i, v_{i+1}, \dots, v_j\}$.

Тогда: $C_{ij} = \min\limits_{r=i}^{j}\{\sum\limits_{k\in[i,j]} p_k + C_{i,r-1} + C_{r+1,j}\}$

Рекуррентное соотношение

Введем для $1 \leq i < j \leq n$ обозначение C_{ij} - взвешенное время поиска элемента в оптимальном BST из вершин $\{v_i, v_{i+1}, \ldots, v_j\}$.

Тогда: $C_{ij} = \min\limits_{r=i}^{j}\{\sum\limits_{k \in [i,j]} p_k + C_{i,r-1} + C_{r+1,j}\}$

Есть краевые случаи: $C_{i,i-1}$ и $C_{j+1,j}$, их значения доопределим нулями.

Рекуррентное соотношение

Введем для $1 \le i < j \le n$ обозначение C_{ij} - взвешенное время поиска элемента в оптимальном BST из вершин $\{v_i, v_{i+1}, \ldots, v_j\}$.

Тогда: $C_{ij} = \min\limits_{r=i}^{j}\{\sum\limits_{k \in [i,j]} p_k + C_{i,r-1} + C_{r+1,j}\}$

Есть краевые случаи: $C_{i,i-1}$ и $C_{j+1,j}$, их значения доопределим нулями, как и для любого случая, когда i > j.

Рекуррентное соотношение

Введем для $1 \leq i < j \leq n$ обозначение C_{ij} - взвешенное время поиска элемента в оптимальном BST из вершин $\{v_i, v_{i+1}, \ldots, v_j\}$.

Тогда: $C_{ij} = \min\limits_{r=i}^{j}\{\sum\limits_{k\in[i,j]} p_k + C_{i,r-1} + C_{r+1,j}\}$

Есть краевые случаи: $C_{i,i-1}$ и $C_{j+1,j}$, их значения доопределим нулями, как и для любого случая, когда i > j.

Осталось решить восходящим анализом!

Алгоритм

A - снова двумерный массив.

```
for s in [0, n - 1]:
```

выбираем длину промежутка
между i и j

Алгоритм

А - снова двумерный массив.

```
for s in [0, n - 1]:
    for i in [1, n]:
```

выбираем длину промежутка между i и j

и для этого промежутка перебираем i

Алгоритм

A - снова двумерный массив.

```
for s in [0, n - 1]:        ← выбираем длину промежутка
                              между i и j
    for i in [1, n]:        ← и для этого промежутка
                              перебираем i
```

A[i, i + s] = $\min\limits_{r=[i,i+s]}\left\{\sum\limits_{k\in[i,i+s]} p_i + A[i, r-1] + A[r+1, i+s]\right\}$

Алгоритм

A - снова двумерный массив.

```
for s in [0, n - 1]:
    for i in [1, n]:
```

выбираем длину промежутка между i и j

и для этого промежутка перебираем i

A[i, i + s] = $\min_{r=[i,i+s]} \{ \sum_{k\in[i,i+s]} p_i + A[i, r-1] + A[r+1, i+s] \}$

Важно: если первый индекс оказывается больше второго, сразу возвращаем 0.

Алгоритм

A - снова двумерный массив.

```
for s in [0, n - 1]:
    for i in [1, n]:
```

выбираем длину промежутка между i и j

и для этого промежутка перебираем i

A[i, i + s] = $\min\limits_{r=[i,i+s]} \left\{ \sum\limits_{k \in [i,i+s]} p_i + A[i, r-1] + A[r+1, i+s] \right\}$

Важно: если первый индекс оказывается больше второго, сразу возвращаем 0. Поэтому все под главной диагональю сразу занулено.

Алгоритм

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix}$$

A - снова двумерный массив.

```
for s in [0, n - 1]:        ← выбираем длину промежутка между i и j

    for i in [1, n]:        ← и для этого промежутка перебираем i
```

A[i, i + s] = $\min\limits_{r=[i,i+s]} \left\{ \sum\limits_{k \in [i,i+s]} p_i + A[i, r-1] + A[r+1, i+s] \right\}$

Важно: если первый индекс оказывается больше второго, сразу возвращаем 0. Поэтому все под главной диагональю сразу занулено.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{bmatrix}$$

Алгоритм

A - снова двумерный массив.

```
for s in [0, n - 1]:
```

выбираем длину промежутка между i и j

```
    for i in [1, n]:
```

и для этого промежутка перебираем i

A[i, i + s] = $\min\limits_{r=[i,i+s]} \left\{ \sum\limits_{k\in[i,i+s]} p_i + A[i, r-1] + A[r+1, i+s] \right\}$

Важно: если первый индекс оказывается больше второго, сразу возвращаем 0. Поэтому все под главной диагональю сразу занулено.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix}$$

Алгоритм

A - снова двумерный массив.

```
for s in [0, n - 1]:
    for i in [1, n]:
```

выбираем длину промежутка между i и j

и для этого промежутка перебираем i

A[i, i + s] = $\min\limits_{r=[i,i+s]} \left\{ \sum\limits_{k\in[i,i+s]} p_i + A[i, r-1] + A[r+1, i+s] \right\}$

Важно: если первый индекс оказывается больше второго, сразу возвращаем 0. Поэтому все под главной диагональю сразу занулено.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix}$$

Алгоритм

`A - снова двумерный массив.`

```
for s in [0, n - 1]:
    for i in [1, n]:
```

выбираем длину промежутка между i и j

и для этого промежутка перебираем i

`A[i, i + s] =` $\min\limits_{r=[i,i+s]} \left\{ \sum\limits_{k\in[i,i+s]} p_i + A[i, r-1] + A[r+1, i+s] \right\}$

Важно: если первый индекс оказывается больше второго, сразу возвращаем 0. Поэтому все под главной диагональю сразу занулено.

Алгоритм

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix}$$ ответ

A - снова двумерный массив.

```
for s in [0, n - 1]:
    for i in [1, n]:
```

выбираем длину промежутка между i и j

и для этого промежутка перебираем i

A[i, i + s] = $\min\limits_{r=[i,i+s]} \left\{ \sum\limits_{k\in[i,i+s]} p_i + A[i, r-1] + A[r+1, i+s] \right\}$

Важно: если первый индекс оказывается больше второго, сразу возвращаем 0. Поэтому все под главной диагональю сразу занулено.

Алгоритм

`Сложность?`

Алгоритм

Сложность?

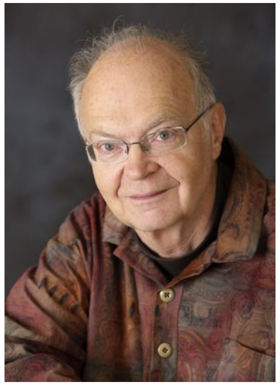
1. Подзадач всего квадратичное число, O(n^2)
2. В каждой задаче линейное количество работы (от i до j)
3. Получаем O(n^3)

Алгоритм

Сложность?

1. Подзадач всего квадратичное число, O(n^2)
2. В каждой задаче линейное количество работы (от i до j)
3. Получаем O(n^3)

Есть и другой алгоритм динамического программирования, который решает задачу за O(n^2) за авторством Дональда Кнута.



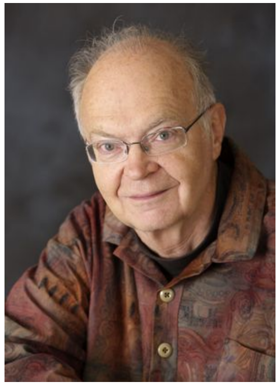
Алгоритм

Сложность?

1. Подзадач всего квадратичное число, O(n^2)
2. В каждой задаче линейное количество работы (от i до j)
3. Получаем O(n^3)

Есть и другой алгоритм динамического программирования, который решает задачу за O(n^2) за авторством Дональда Кнута.

Кроме того, мы говорили только про статические деревья, которые не изменяются между запросами.

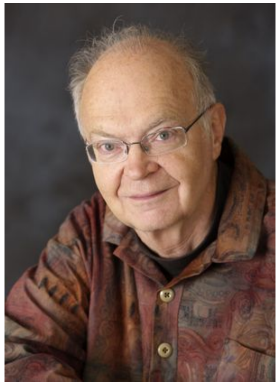


Алгоритм

Сложность?

1. Подзадач всего квадратичное число, O(n^2)
2. В каждой задаче линейное количество работы (от i до j)
3. Получаем O(n^3)

Есть и другой алгоритм динамического программирования, который решает задачу за O(n^2) за авторством Дональда Кнута.

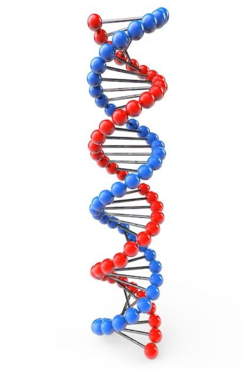


Кроме того, мы говорили только про статические деревья, которые не изменяются между запросами.

Про динамические поговорим позже (splay-trees).

Выравнивание последовательностей

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

Выравнивание последовательностей

Дано: строки $X = x_1 x_2 \ldots x_m$ и $Y = y_1 y_2 \ldots y_n$

Выравнивание последовательностей

Дано: строки $X = x_1 x_2 \ldots x_m$ и $Y = y_1 y_2 \ldots y_n$

$d_{gap} \geq 0$ - штраф за пропуск,

$d_{ab} \geq 0$ - штраф за ситуацию, когда символы a и b оказались на одном месте (0, когда a = b)

Выравнивание последовательностей

Дано: строки $X = x_1 x_2 \ldots x_m$ и $Y = y_1 y_2 \ldots y_n$

$d_{gap} \geq 0$ - штраф за пропуск,

$d_{ab} \geq 0$ - штраф за ситуацию, когда символы a и b оказались на одном месте (0, когда a = b)

Найти: такое выравнивание последовательностей X и Y (т.е. дополнение их до одинаковой длины, возможно большей, чем максимальная из изначальных длин), чтобы сумма штрафов по каждой позиции была минимальной.

Структура оптимального решения

Пусть в конце получим выровненное оптимальное решение:

$$---X + gaps ---$$

$$---Y + gaps ---$$

Структура оптимального решения

Пусть в конце получим выровненное оптимальное решение:

$$---X + gaps --\boxed{-}$$

$$---Y + gaps --\boxed{-}$$

Какие варианты есть для двух последних элементов?

$$X = x_1 x_2 \ldots x_m$$
$$Y = y_1 y_2 \ldots y_n$$

Структура оптимального решения

Пусть в конце получим выровненное оптимальное решение:

$$\begin{array}{l} - - - X + gaps - -\boxed{-} \\ - - - Y + gaps - -\boxed{-} \end{array} \Longrightarrow \quad \begin{array}{ll} 1) & x_m \\ & y_n \end{array}$$

Какие варианты есть для двух последних элементов?

$$X = x_1 x_2 \dots x_m$$
$$Y = y_1 y_2 \dots y_n$$

Структура оптимального решения

Пусть в конце получим выровненное оптимальное решение:

$$\begin{array}{l} ---X + gaps --- \\ ---Y + gaps --- \end{array} \Longrightarrow \quad 1)\ \begin{array}{c} x_m \\ y_n \end{array} \qquad 2)\ \begin{array}{c} \_\_ \\ y_n \end{array} \qquad 3)\ \begin{array}{c} x_m \\ \_\_ \end{array}$$

Какие варианты есть для двух последних элементов?

$$X = x_1 x_2 \ldots x_m$$
$$Y = y_1 y_2 \ldots y_n$$

Структура оптимального решения

Пусть в конце получим выровненное оптимальное решение:

$$\begin{array}{l} - - -X + gaps - -\boxed{-} \\ - - -Y + gaps - -\boxed{-} \end{array} \Longrightarrow \quad 1)\ \begin{array}{c} x_m \\ y_n \end{array} \quad 2)\ \begin{array}{c} \_\_ \\ y_n \end{array} \quad 3)\ \begin{array}{c} x_m \\ \_\_ \end{array}$$

Какие варианты есть для двух последних элементов?

Два пропуска на последнем месте не имеют смысла, т.к. их можно просто убрать и сразу улучшить счет => в оптимальном решении их быть не может.

Оптимального подструктура

Пусть в конце получим выровненное оптимальное решение:

$$\begin{array}{l} ---X+gaps--\boxed{-} \\ ---Y+gaps--\boxed{-} \end{array} \quad \Longrightarrow \quad 1)\ \begin{array}{c} x_m \\ y_n \end{array} \qquad 2)\ \begin{array}{c} \_\_ \\ y_n \end{array} \qquad 3)\ \begin{array}{c} x_m \\ \_\_ \end{array}$$

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение:

Оптимального подструктура

Пусть в конце получим выровненное оптимальное решение:

$$\boxed{\begin{array}{c} ---X + gaps --- \\ ---Y + gaps --- \end{array}} \Longrightarrow \quad 1)\ \begin{array}{c} x_m \\ y_n \end{array} \quad 2)\ \begin{array}{c} \_ \\ y_n \end{array} \quad 3)\ \begin{array}{c} x_m \\ \_ \end{array}$$

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение:

1. в первом случае префиксные последовательности из X' и Y' выровнены оптимальным образом;

Оптимального подструктура

Пусть в конце получим выровненное оптимальное решение:

$$\boxed{\begin{matrix} ---X+gaps-- \\ ---Y+gaps-- \end{matrix}}\begin{matrix}- \\ -\end{matrix} \quad \Rightarrow \quad 1)\ \begin{matrix} x_m \\ y_n \end{matrix} \qquad 2)\ \begin{matrix} \_ \\ y_n \end{matrix} \qquad 3)\ \begin{matrix} x_m \\ \_ \end{matrix}$$

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение:

1. в первом случае префиксные последовательности из X' и Y' выровнены оптимальным образом;
2. во втором случае префиксные последовательности из X и Y' выровнены оптимальным образом;

Оптимального подструктура

Пусть в конце получим выровненное оптимальное решение:

$$\boxed{\begin{array}{l} ---X + gaps --- \\ ---Y + gaps --- \end{array}} \Rightarrow \quad 1)\ \begin{array}{c} x_m \\ y_n \end{array} \quad 2)\ \begin{array}{c} \_ \\ y_n \end{array} \quad 3)\ \begin{array}{c} x_m \\ \_ \end{array}$$

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение:

1. в первом случае префиксные последовательности из X' и Y' выровнены оптимальным образом;
2. во втором случае префиксные последовательности из X и Y' выровнены оптимальным образом;
3. в третьем случае префиксные последовательности из X' и Y выровнены оптимальным образом.

Оптимального подструктура

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение: в первом случае (когда в последней позиции оптимального решения стоят x_m и y_n) префиксные последовательности из X' и Y' выровнены оптимальным образом.

Док-во: от противного. Пусть есть более оптимальное выравнивание для X' и Y'.

Оптимального подструктура

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение: в первом случае (когда в последней позиции оптимального решения стоят x_m и y_n) префиксные последовательности из X' и Y' выровнены оптимальным образом.

Док-во: от противного. Пусть есть более оптимальное выравнивание для X' и Y'. Т.е. если суммарный штраф для X' и Y' в оптимальном решении равен p, но есть другое выравнивание, где он будет равен p^* и $p^* < p$.

Оптимального подструктура

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение: в первом случае (когда в последней позиции оптимального решения стоят x_m и y_n) префиксные последовательности из X' и Y' выровнены оптимальным образом.

Док-во: от противного. Пусть есть более оптимальное выравнивание для X' и Y'. Т.е. если суммарный штраф для X' и Y' в оптимальном решении равен p, но есть другое выравнивание, где он будет равен p^* и $p^* < p$.

Тогда возьмем это более оптимальное выравнивание и добавим к нему в конец элементы x_m и y_n .

Оптимального подструктура

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение: в первом случае (когда в последней позиции оптимального решения стоят x_m и y_n) префиксные последовательности из X' и Y' выровнены оптимальным образом.

Док-во: от противного. Пусть есть более оптимальное выравнивание для X' и Y'. Т.е. если суммарный штраф для X' и Y' в оптимальном решении равен p, но есть другое выравнивание, где он будет равен p^* и $p^* < p$.

Тогда возьмем это более оптимальное выравнивание и добавим к нему в конец элементы x_m и y_n . Тогда суммарный штраф получившегося выравнивания для элементов X' и Y' будет равен

$p^* + d_{x_m y_n}$

Оптимального подструктура

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение: в первом случае (когда в последней позиции оптимального решения стоят x_m и y_n) префиксные последовательности из X' и Y' выровнены оптимальным образом.

Док-во: от противного. Пусть есть более оптимальное выравнивание для X' и Y'. Т.е. если суммарный штраф для X' и Y' в оптимальном решении равен p, но есть другое выравнивание, где он будет равен p^* и $p^* < p$.

Тогда возьмем это более оптимальное выравнивание и добавим к нему в конец элементы x_m и y_n . Тогда суммарный штраф получившегося выравнивания для элементов X' и Y' будет равен

$p^* + d_{x_m y_n} < p + d_{x_m y_n}$

Оптимального подструктура

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение: в первом случае (когда в последней позиции оптимального решения стоят x_m и y_n) префиксные последовательности из X' и Y' выровнены оптимальным образом.

Док-во: от противного. Пусть есть более оптимальное выравнивание для X' и Y'. Т.е. если суммарный штраф для X' и Y' в оптимальном решении равен p, но есть другое выравнивание, где он будет равен p^* и $p^* < p$.

Тогда возьмем это более оптимальное выравнивание и добавим к нему в конец элементы x_m и y_n . Тогда суммарный штраф получившегося выравнивания для элементов X' и Y' будет равен

$p^* + d_{x_m y_n} < p + d_{x_m y_n}$ ⟵ суммарный штраф за оптимальное решение!

Оптимального подструктура

Получаем противоречие с оптимальностью изначального решения $\square$

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение: в первом случае (когда в последней позиции оптимального решения стоят x_m и y_n) префиксные последовательности из X' и Y' выровнены оптимальным образом.

Док-во: от противного. Пусть есть более оптимальное выравнивание для X' и Y'. Т.е. если суммарный штраф для X' и Y' в оптимальном решении равен p, но есть другое выравнивание, где он будет равен p^* и $p^* < p$.

Тогда возьмем это более оптимальное выравнивание и добавим к нему в конец элементы x_m и y_n . Тогда суммарный штраф получившегося выравнивания для элементов X' и Y' будет равен

$p^* + d_{x_m y_n} < p + d_{x_m y_n}$ ⟵ суммарный штраф за оптимальное решение!

Оптимального подструктура

Получаем противоречие
с оптимальностью
изначального решения $\square$

Для случаев 2 и 3 -
аналогично.

Обозначим $X' = X - x_m$ и $Y' = Y - y_n$.

Утверждение: в первом случае (когда в последней позиции оптимального решения стоят x_m и y_n) префиксные последовательности из X' и Y' выровнены оптимальным образом.

Док-во: от противного. Пусть есть более оптимальное выравнивание для X' и Y'. Т.е. если суммарный штраф для X' и Y' в оптимальном решении равен p, но есть другое выравнивание, где он будет равен p^* и $p^* < p$.

Тогда возьмем это более оптимальное выравнивание и добавим к нему в конец элементы x_m и y_n . Тогда суммарный штраф получившегося выравнивания для элементов X' и Y' будет равен

$p^* + d_{x_m y_n} < p + d_{x_m y_n}$ $\longleftarrow$ суммарный штраф за оптимальное решение!

Рекуррентное соотношение

Обозначим X_i и Y_j - префиксы из первых i и j букв в последовательностях X и Y соответственно.

Рекуррентное соотношение

Обозначим X_i и Y_j - префиксы из первых i и j букв в последовательностях X и Y соответственно.

В качестве подзадач нас будут интересовать пары (X_i, Y_j), т.к. мы всегда откусываем буквы с правого конца.

Рекуррентное соотношение

Обозначим X_i и Y_j - префиксы из первых i и j букв в последовательностях X и Y соответственно.

В качестве подзадач нас будут интересовать пары (X_i, Y_j), т.к. мы всегда откусываем буквы с правого конца.

Введем p_{ij} - суммарный штраф при оптимальном выравнивании X_i и Y_j

Рекуррентное соотношение

Обозначим X_i и Y_j - префиксы из первых i и j букв в последовательностях X и Y соответственно.

В качестве подзадач нас будут интересовать пары (X_i, Y_j), т.к. мы всегда откусываем буквы с правого конца.

Введем p_{ij} - суммарный штраф при оптимальном выравнивании X_i и Y_j

Тогда для $i \in [1, m]$ и $j \in [1, n]$ верно:

$$p_{ij} = \min \begin{cases} d_{x_i y_j} + p_{i-1,j-1} \\ d_{gap} + p_{i-1,j} \\ d_{gap} + p_{i,j-1} \end{cases}$$

Рекуррентное соотношение

Обозначим X_i и Y_j - префиксы из первых i и j букв в последовательностях X и Y соответственно.

В качестве подзадач нас будут интересовать пары (X_i, Y_j), т.к. мы всегда откусываем буквы с правого конца.

Введем p_{ij} - суммарный штраф при оптимальном выравнивании X_i и Y_j

Тогда для $i \in [1, m]$ и $j \in [1, n]$ верно:

$$p_{ij} = \min \begin{cases} d_{x_i y_j} + p_{i-1,j-1} \\ d_{gap} + p_{i-1,j} \\ d_{gap} + p_{i,j-1} \end{cases}$$

Доопределим $p_{i,0}$ и $p_{0,i}$ как $i * d_{gap}$

Действительно, выравниваем пустую оследовательность до длины i.

Алгоритм (Нидлмана-Вунша)

1) Заводим двумерный массив A размера m x n.

Алгоритм (Нидлмана-Вунша)

1) Заводим двумерный массив A размера m x n.

2) Инициализируем его края: A[0][k] = A[k][0] = $k * d_{gap}$

Алгоритм (Нидлмана-Вунша)

1) Заводим двумерный массив A размера m x n.

2) Инициализируем его края: A[0][k] = A[k][0] = $k * d_{gap}$

3) Заполняем массив:

```
for i in range(1, m):
    for j in range(1, n):
```

Алгоритм (Нидлмана-Вунша)

1) Заводим двумерный массив A размера m x n.

2) Инициализируем его края: A[0][k] = A[k][0] = $k * d_{gap}$

3) Заполняем массив:

```
for i in range(1, m):
   for j in range(1, n):
```

$$A[i][j] = \min \begin{cases} A[i-1][j-1] + d_{x_i, y_j} \\ A[i-1][j] + d_{gap} \\ A[i][j-1] + d_{gap} \end{cases}$$

Алгоритм (Нидлмана-Вунша)

1) Заводим двумерный массив A размера m x n.

2) Инициализируем его края: A[0][k] = A[k][0] = $k * d_{gap}$

3) Заполняем массив:

```
for i in range(1, m):
    for j in range(1, n):
```

$$A[i][j] = \min \begin{cases} A[i-1][j-1] + d_{x_i, y_j} \\ A[i-1][j] + d_{gap} \\ A[i][j-1] + d_{gap} \end{cases}$$

Сложность: O(nm)

Восстановление решения

1) Начинаем с позиции A[m][n]

Восстановление решения

1) Начинаем с позиции A[m][n]

2) Каждый раз думаем, из какой позиции мы пришли в текущую: из (i-1, j-1), из (i, j-1) или из (i-1, j). "Думаем" - значит выбираем минимум из 3 значений (знач. в ячейках + стоимость)

Восстановление решения

1) Начинаем с позиции A[m][n]

2) Каждый раз думаем, из какой позиции мы пришли в текущую: из (i-1, j-1), из (i, j-1) или из (i-1, j). "Думаем" - значит выбираем минимум из 3 значений (знач. в ячейках + стоимость)

3) Если сделали переход из (i-1,j-1), значит элементы x_i, y_j стоят на одной позиции (берем их оба).

   Если из (i-1, j), значит берем элемент x_i, а напротив в X ставим пропуск.

   Если из (i, j-1), значит берем элемент y_j, а напротив в Y ставим пропуск.

Работает за O(m + n) 🥳

Восстановление решения

1) Начинаем с позиции A[m][n]

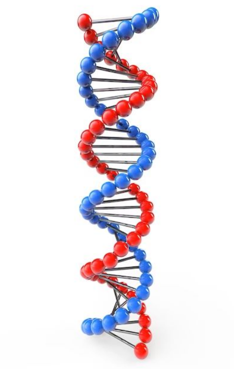
2) Каждый раз думаем, из какой позиции мы пришли в текущую: из (i-1, j-1), из (i, j-1) или из (i-1, j). "Думаем" - значит выбираем минимум из 3 значений (знач. в ячейках + стоимость)

3) Если сделали переход из (i-1,j-1), значит элементы x_i, y_j стоят на одной позиции (берем их оба).

   Если из (i-1, j), значит берем элемент x_i, а напротив в X ставим пропуск.

   Если из (i, j-1), значит берем элемент y_j, а напротив в Y ставим пропуск.

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

```
X = GTTAC
Y = GACGT
```

Простой пример #2

X = GTTAC

Y = GACGT

		G	A	C	G	T
G						
T						
T						
A						
C						

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	?				
T	2					
T	3					
A	4					
C	5					

Простой пример #2

X = GTTAC

Y = GACGT

$$d_{gap} = 1$$

$$x \neq y \Rightarrow d_{x,y} = 2$$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	?				
T	2					
T	3					
A	4					
C	5					

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

$0 + d_{1,1} = 0 + 0 = 0$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0				
T	2					
T	3					
A	4					
C	5					

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	?			
T	2					
T	3					
A	4					
C	5					

$$0 + d_{gap} = 0 + 1 = 1$$

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	?			
T	2					
T	3					
A	4					
C	5					

Простой пример #2

X = GTTAC

Y = GACGT

$$d_{gap} = 1$$

$$x \neq y \Rightarrow d_{x,y} = 2$$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	?		
T	2					
T	3					
A	4					
C	5					

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

$1 + d_{gap} = 1 + 1 = 2$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	?		
T	2					
T	3					
A	4					
C	5					

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

🤔

$$3 + d_{gap} = 3 + 1 = 4$$

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

🤔

$3 + d_{gap} = 3 + 1 = 4$

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

Ответ:

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

🤔

–
T

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

Ответ:

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

🤔

$\begin{matrix} - \\ T \end{matrix}$

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

Ответ:

$2 + d_{gap} = 2 + 1 = 3$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

🤔

–
T

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

Ответ:

$$2 + d_{gap} = 2 + 1 = 3$$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

🤔

\- -

G T

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

Ответ:

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

```
– –
G T
```

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

Ответ:

```
C _ _
C G T
```

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

Ответ:

```
A C _ _
A C G T
```

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

Ответ:

```
T A C _ _
_ A C G T
```

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

Ответ:

T T A C _ _

_ _ A C G T

Простой пример #2

X = GTTAC

Y = GACGT

$d_{gap} = 1$

$x \neq y \Rightarrow d_{x,y} = 2$

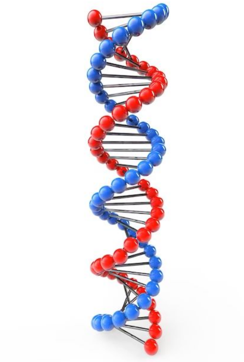
Ответ:

G T T A C _ _

G _ _ A C G T

😎		G	A	C	G	T
	0	1	2	3	4	5
G	1	0	1	2	3	4
T	2	1	2	3	4	3
T	3	2	3	4	5	4
A	4	3	2	3	4	5
C	5	4	3	2	3	4

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

Мини-задача #39 (2 балла)

Решить задачу распознавания строки по регулярному выражению через динамическое программирование.

https://leetcode.com/problems/regular-expression-matching

Мини-задача #40 (2 балла)

Посчитайте минимальное стоимость слияния камней в одну кучу через динамическое программирование.

https://leetcode.com/problems/minimum-cost-to-merge-stones

Takeaways

- Сбалансированные BST могут быть превзойдены, если есть дополнительная информация о ключах.

- Поиск (статического) оптимального BST - задачи динамического программирования (похожа на порядок перемножения матриц)

- Выравнивание последовательности - еще один пример задачи динамического программирования.