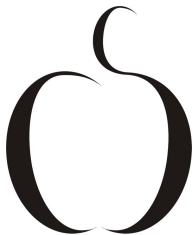


с/к “Эффективные алгоритмы”

Лекция 13: Динамическое программирование

А. Куликов

Computer Science клуб при ПОМИ
<http://logic.pdmi.ras.ru/~infclub/>



План лекции

1 Введение

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц
- 6 Кратчайшие пути
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа
 - Задача о коммивояжере

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц
- 6 Кратчайшие пути
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа
 - Задача о коммивояжере
- 7 Независимые множества в деревьях

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц
- 6 Кратчайшие пути
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа
 - Задача о коммивояжере
- 7 Независимые множества в деревьях

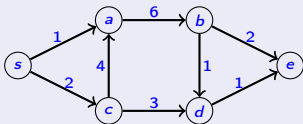
Кратчайшие пути в направленных ациклических графах

Кратчайшие пути в направленных ациклических графах

Кратчайшие пути в направленных ациклических графах

Кратчайшие пути в направленных ациклических графах

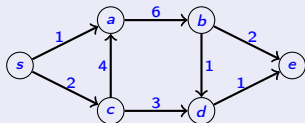
- Рассмотрим направленный ациклический граф.



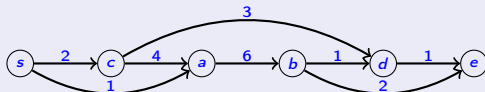
Кратчайшие пути в направленных ациклических графах

Кратчайшие пути в направленных ациклических графах

- Рассмотрим направленный ациклический направленный граф.



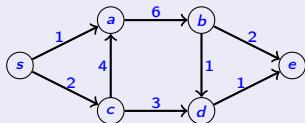
- Важным свойством такого графа является наличие топологической сортировки его вершин.



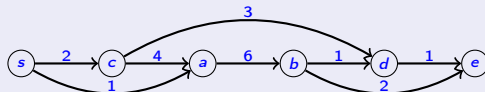
Кратчайшие пути в направленных ациклических графах

Кратчайшие пути в направленных ациклических графах

- Рассмотрим направленный ациклический граф.



- Важным свойством такого графа является наличие топологической сортировки его вершин.

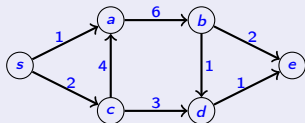


- Допустим, мы хотим найти длину кратчайшего пути от вершины s до вершины d .

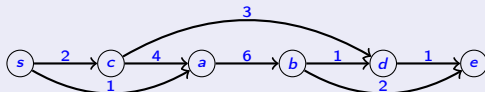
Кратчайшие пути в направленных ациклических графах

Кратчайшие пути в направленных ациклических графах

- Рассмотрим направленный ациклический направленный граф.



- Важным свойством такого графа является наличие топологической сортировки его вершин.



- Допустим, мы хотим найти длину кратчайшего пути от вершины s до вершины d .
- Ясно, что $\text{dist}(d) = \min\{\text{dist}(b) + 1, \text{dist}(c) + 3\}$.

Кратчайшие пути в направленных ациклических графах

Алгоритм

DAG-SHORTEST-PATHS(G, s)

- 1 инициализировать все значения массива dist значением ∞
- 2 $\text{dist}(s) = 0$
- 3 **for** $v \in V \setminus \{s\}$ в топологической сортировке вершин
- 4 **do** $\text{dist}(v) = \min_{(u,v) \in E} \{\text{dist}(u) + d(u, v)\}$

Кратчайшие пути в направленных ациклических графах

Алгоритм

DAG-SHORTEST-PATHS(G, s)

- 1 инициализировать все значения массива `dist` значением ∞
- 2 `dist(s) = 0`
- 3 **for** $v \in V \setminus \{s\}$ в топологической сортировке вершин
- 4 **do** `dist(v) = $\min_{(u,v) \in E} \{dist(u) + d(u, v)\}$`

Замечание

Кратчайшие пути в направленных ациклических графах

Алгоритм

DAG-SHORTEST-PATHS(G, s)

- 1 инициализировать все значения массива dist значением ∞
- 2 $\text{dist}(s) = 0$
- 3 **for** $v \in V \setminus \{s\}$ в топологической сортировке вершин
- 4 **do** $\text{dist}(v) = \min_{(u,v) \in E} \{\text{dist}(u) + d(u, v)\}$

Замечание

- Алгоритм решает множество подзадач $\{\text{dist}(u), u \in V\}$ в порядке возрастания их “сложности”.

Кратчайшие пути в направленных ациклических графах

Алгоритм

DAG-SHORTEST-PATHS(G, s)

- 1 инициализировать все значения массива dist значением ∞
- 2 $\text{dist}(s) = 0$
- 3 **for** $v \in V \setminus \{s\}$ в топологической сортировке вершин
- 4 **do** $\text{dist}(v) = \min_{(u,v) \in E} \{\text{dist}(u) + d(u, v)\}$

Замечание

- Алгоритм решает множество подзадач $\{\text{dist}(u), u \in V\}$ в порядке возрастания их “сложности”.
- Говорим, что первая задача сложнее второй, если для решения первой необходимо знать ответ для второй.

Динамическое программирование

Динамическое программирование

Динамическое программирование

Динамическое программирование

- Динамическое программирование решает задачу, разбивая её на подзадачи и решая их от простых к сложным, периодически используя ответы для уже решенных подзадач.

Динамическое программирование

Динамическое программирование

- Динамическое программирование решает задачу, разбивая её на подзадачи и решая их от простых к сложным, периодически используя ответы для уже решенных подзадач.
- Подзадачи образуют направленный ациклический граф: из вершины A идет ребро в вершину B , если для решения A необходимо решить B .

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность**
- 3 Стоимость редактирования
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц
- 6 Кратчайшие пути
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа
 - Задача о коммивояжере
- 7 Независимые множества в деревьях

Наибольшая возрастающая подпоследовательность

Определение

Задача о наибольшей возрастающей последовательности (longest increasing subsequence problem) заключается в нахождении по данной последовательности её возрастающей подпоследовательности максимальной длины.

Наибольшая возрастающая подпоследовательность

Определение

Задача о наибольшей возрастающей последовательности (longest increasing subsequence problem) заключается в нахождении по данной последовательности её возрастающей подпоследовательности максимальной длины.

Пример

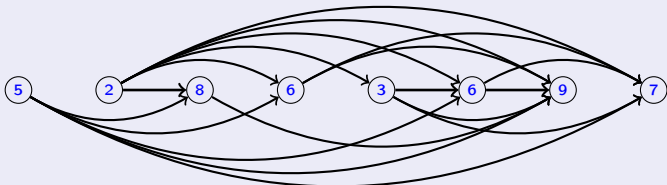
Наибольшая возрастающая подпоследовательность

Определение

Задача о наибольшей возрастающей последовательности (longest increasing subsequence problem) заключается в нахождении по данной последовательности её возрастающей подпоследовательности максимальной длины.

Пример

- Рассмотрим последовательность 5, 2, 8, 6, 3, 6, 9, 7.



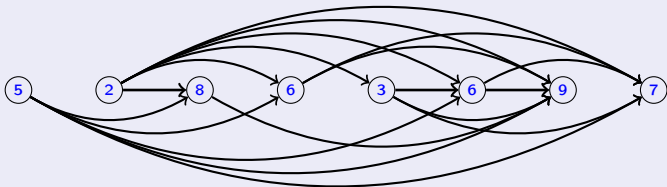
Наибольшая возрастающая подпоследовательность

Определение

Задача о наибольшей возрастающей последовательности (longest increasing subsequence problem) заключается в нахождении по данной последовательности её возрастающей подпоследовательности максимальной длины.

Пример

- Рассмотрим последовательность 5, 2, 8, 6, 3, 6, 9, 7.



- Ясно, что нам просто нужно найти длинейший путь в этом графе.

Алгоритм

Алгоритм

```
1  for  $j \leftarrow 1$  to  $n$ 
2      do  $L(j) \leftarrow 1 + \max\{L(i) : (i, j) \in E\}$ 
3           $\triangleright L(j)$  — длина максимальной последовательности,
              заканчивающейся в  $j$ -м элементе
4  return  $\max_j L(j)$ 
```

Замечания

Замечания

Замечания

Замечания

- Время работы есть $O(n^2)$.

Замечания

Замечания

- Время работы есть $O(n^2)$.
- Мы опять решали подзадачи вида $\{L(j) : 1 \leq j \leq n\}$ с важным свойством: на подзадачах задан частичный порядок и отношение, определяющее, как решить задачу по ее более простым подзадачам.

Замечания

Замечания

- Время работы есть $O(n^2)$.
- Мы опять решали подзадачи вида $\{L(j) : 1 \leq j \leq n\}$ с важным свойством: на подзадачах задан частичный порядок и отношение, определяющее, как решить задачу по ее более простым подзадачам.
- Наш алгоритм находит **длину** максимальной возрастающей подпоследовательности, но его легко модифицировать так, чтобы он находил и саму максимальную подпоследовательность.

Рекурсия и динамическое программирование

Рекурсия и динамическое программирование

Рекурсия и динамическое программирование

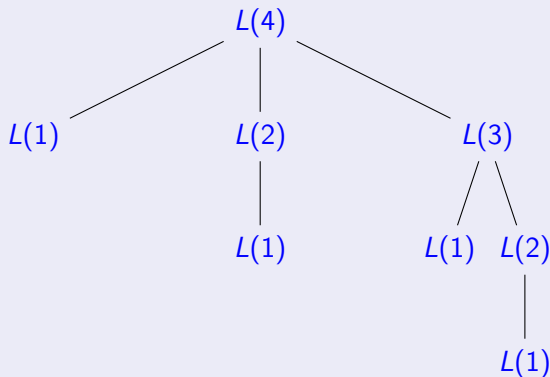
Рекурсия и динамическое программирование

- Рекурсивное определение: $L(j) = 1 + \max\{L(1), L(2), \dots, L(j-1)\}$.

Рекурсия и динамическое программирование

Рекурсия и динамическое программирование

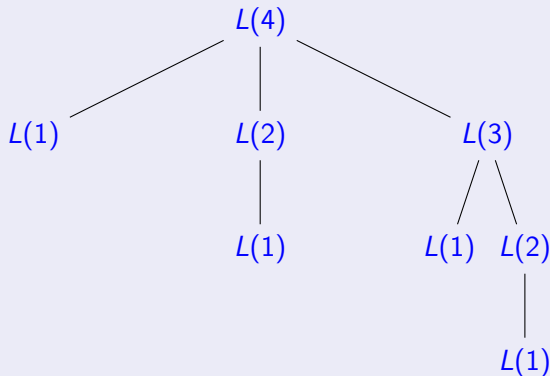
- Рекурсивное определение: $L(j) = 1 + \max\{L(1), L(2), \dots, L(j-1)\}$.



Рекурсия и динамическое программирование

Рекурсия и динамическое программирование

- Рекурсивное определение: $L(j) = 1 + \max\{L(1), L(2), \dots, L(j-1)\}$.



- Эффективность динамического программирования достигается, таким образом, за счет разумного перечисления подзадач и порядка на них.

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования**
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц
- 6 Кратчайшие пути
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа
 - Задача о коммивояжере
- 7 Независимые множества в деревьях

Стоимость редактирования

Определение

Задача о стоимости редактирования (edit distance problem)

заключается в нахождении минимального количества вставок, удалений и замен букв, необходимых для того, чтобы из одного входного слова получить другое.

Стоимость редактирования

Определение

Задача о стоимости редактирования (edit distance problem)

заключается в нахождении минимального количества вставок, удалений и замен букв, необходимых для того, чтобы из одного входного слова получить другое.

Пример

E X P O N E N - T I A L
- - P O L Y N O M I A L

Стоимость равна 6.

Подзадачи

Подзадачи

Подзадачи

Подзадачи

- Важным моментом динамического программирования является выбор подзадач.

Подзадачи

Подзадачи

- Важным моментом динамического программирования является выбор подзадач.
- Нам нужно найти стоимость редактирования строчек $x[1 \dots m]$ и $y[1 \dots n]$.

Подзадачи

Подзадачи

- Важным моментом динамического программирования является выбор подзадач.
- Нам нужно найти стоимость редактирования строчек $x[1 \dots m]$ и $y[1 \dots n]$.
- Естественной подзадачей является $E(i, j)$ — стоимость редактирования префикса длины i строчки x и префикса длины j строчки y .

Подзадачи

Подзадачи

- Важным моментом динамического программирования является выбор подзадач.
- Нам нужно найти стоимость редактирования строчек $x[1 \dots m]$ и $y[1 \dots n]$.
- Естественной подзадачей является $E(i, j)$ — стоимость редактирования префикса длины i строчки x и префикса длины j строчки y .
- Нашей целью является вычисление $E(m, n)$.

Подзадачи

Подзадачи

- Важным моментом динамического программирования является выбор подзадач.
- Нам нужно найти стоимость редактирования строчек $x[1 \dots m]$ и $y[1 \dots n]$.
- Естественной подзадачей является $E(i, j)$ — стоимость редактирования префикса длины i строчки x и префикса длины j строчки y .
- Нашей целью является вычисление $E(m, n)$.
- $E(i, j) = \min\{1 + E(i - 1, j), 1 + E(i, j - 1), \text{diff}(i, j) + E(i - 1, j - 1)\}$.

Подзадачи

Подзадачи

- Важным моментом динамического программирования является выбор подзадач.
- Нам нужно найти стоимость редактирования строчек $x[1 \dots m]$ и $y[1 \dots n]$.
- Естественной подзадачей является $E(i, j)$ — стоимость редактирования префикса длины i строчки x и префикса длины j строчки y .
- Нашей целью является вычисление $E(m, n)$.
- $E(i, j) = \min\{1 + E(i - 1, j), 1 + E(i, j - 1), \text{diff}(i, j) + E(i - 1, j - 1)\}$.
- Значения $E(i, j)$ записываются в таблицу, обходить которую можно в любом порядке, лишь бы $E(i, j)$ всегда шло позже, чем $E(i - 1, j)$, $E(i, j - 1)$, $E(i - 1, j - 1)$.

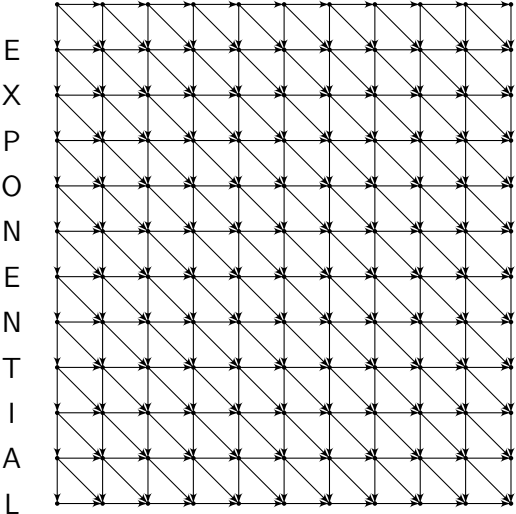
Алгоритм

Алгоритм

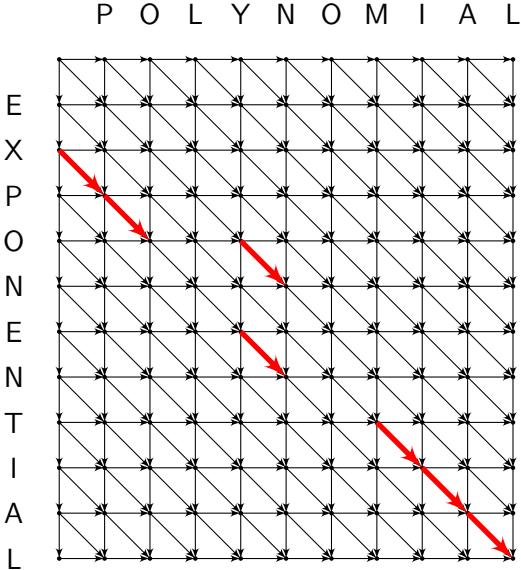
```
1  for  $i \leftarrow 0$  to  $m$ 
2      do  $E(i, 0) = i$ 
3  for  $j \leftarrow 0$  to  $n$ 
4      do  $E(0, j) = j$ 
5  for  $i \leftarrow 1$  to  $m$ 
6      do for  $j \leftarrow 1$  to  $n$ 
7          do  $E(i, j) \leftarrow \min\{E(i - 1, j) + 1, E(i, j - 1) + 1,$ 
               $E(i - 1, j - 1) + \text{diff}(i, j)\}$ 
8  return  $E(m, n)$ 
```

Соответствующий граф

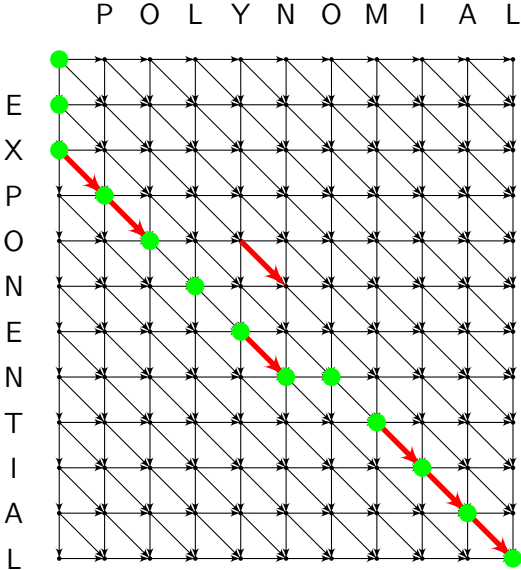
P O L Y N O M I A L



Соответствующий граф



Соответствующий граф



Стандартные способы выбора подзадач

Стандартные способы выбора подзадач

ВХОД

x_1, \dots, x_n

x_1, \dots, x_n и y_1, \dots, y_m

x_1, \dots, x_n

дерево

подзадача

x_1, \dots, x_i

x_1, \dots, x_i и y_1, \dots, y_j

x_i, \dots, x_j

поддерево

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке**
- 5 Перемножение нескольких матриц
- 6 Кратчайшие пути
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа
 - Задача о коммивояжере
- 7 Независимые множества в деревьях

Задача о рюкзаке

Определение

Задача о рюкзаке (knapsack problem) заключается в нахождении по данному набору из n предметов со стоимостями v_1, \dots, v_n и весами w_1, \dots, w_n , а также общему весу W поднабора веса не более W максимальной стоимости.

Задача о рюкзаке

Определение

Задача о рюкзаке (knapsack problem) заключается в нахождении по данному набору из n предметов со стоимостями v_1, \dots, v_n и весами w_1, \dots, w_n , а также общему весу W поднабора веса не более W максимальной стоимости.

Алгоритм для рюкзака с повторениями

```
1  ▷  $K(w)$  — максимальная стоимость при весе не более  $w$ 
2   $K(0) = 0$ 
3  for  $w \leftarrow 1$  to  $W$ 
4      do  $K(w) \leftarrow \max\{K(w - w_i) + v_i : w_i \leq w\}$ 
5  return  $K(W)$ 
```

Рюкзак без повторений

Алгоритм для рюкзака без повторений

```
1  ▷  $K(w, j)$  — максимальная стоимость при весе не более  $w$   
   среди первых  $j$  предметов  
2  for all  $j, w$   
3     do  $K(0, j) = K(w, 0) = 0$   
4  for  $j \leftarrow 1$  to  $n$   
5     do for  $w \leftarrow 1$  to  $W$   
6         do if  $w_j > w$   
7             then  $K(w, j) \leftarrow K(w, j - 1)$   
8             else  $K(w, j) \leftarrow \max\{K(w, j - 1),$   
                 $K(w - w_j, j - 1) + v_j\}$   
9  return  $K(W, n)$ 
```

Запоминание

Запоминание

Запоминание

Запоминание

- Как мы уже видели, реализация вычисления решения для задачи при помощи подзадач простой рекурсией может быть очень не эффективной по той причине, что одни и те же вычисления будут производиться многократно.

Запоминание

Запоминание

- Как мы уже видели, реализация вычисления решения для задачи при помощи подзадач простой рекурсией может быть очень не эффективной по той причине, что одни и те же вычисления будут производиться многократно.
- Можно, однако, избежать повторений в рекурсии.

Запоминание

Запоминание

- Как мы уже видели, реализация вычисления решения для задачи при помощи подзадач простой рекурсией может быть очень не эффективной по той причине, что одни и те же вычисления будут производиться многократно.
- Можно, однако, избежать повторений в рекурсии.

Рекурсивный алгоритм для рюкзака с повторениями

KNAPSACK(w)

- 1 **if** w есть в хэш-таблице
- 2 **then return** $K(w)$
- 3 $K(w) \leftarrow \max\{K(w - w_i) + v_i : w_i \leq w\}$
- 4 добавить пару $(w, K(w))$ в хэш-таблицу
- 5 **return** $K(w)$

Запоминание

Запоминание

Запоминание

Запоминание

- Данный алгоритм не решает по несколько раз одну и ту же задачу, но все-таки тратит какое-то лишнее время на рекурсивные вызовы.

Запоминание

Запоминание

- Данный алгоритм не решает по несколько раз одну и ту же задачу, но все-таки тратит какое-то лишнее время на рекурсивные вызовы.
- Впрочем, такой трюк может иногда давать и выигрыш: метод динамического программирования решает **каждую подзадачу**, в то время как рекурсия с запоминанием решает только **подзадачи, необходимые для вычисления конечного результата**.

Запоминание

Запоминание

- Данный алгоритм не решает по несколько раз одну и ту же задачу, но все-таки тратит какое-то лишнее время на рекурсивные вызовы.
- Впрочем, такой трюк может иногда давать и выигрыш: метод динамического программирования решает **каждую подзадачу**, в то время как рекурсия с запоминанием решает только **подзадачи, необходимые для вычисления конечного результата**.
- Например, если общий вес рюкзака, а также веса всех предметов кратны **100**, то рекурсия с запоминанием будет рассматривать только подзадачи, где вес также кратен **100**, в то время как метод динамического программирования переберет и все оставшиеся подзадачи.

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц**
- 6 Кратчайшие пути
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа
 - Задача о коммивояжере
- 7 Независимые множества в деревьях

Перемножение нескольких матриц

Перемножение нескольких матриц

Перемножение нескольких матриц

Перемножение нескольких матриц

- Пусть нам нужно вычислить произведение $A \times B \times C \times D$ матриц размера 50×20 , 20×1 , 1×10 , 10×100 , соответственно.

Перемножение нескольких матриц

Перемножение нескольких матриц

- Пусть нам нужно вычислить произведение $A \times B \times C \times D$ матриц размера 50×20 , 20×1 , 1×10 , 10×100 , соответственно.
- Как известно, умножение матриц ассоциативно, поэтому порядок, в котором мы будем перемножать, на результат никак не влияет.

Перемножение нескольких матриц

Перемножение нескольких матриц

- Пусть нам нужно вычислить произведение $A \times B \times C \times D$ матриц размера 50×20 , 20×1 , 1×10 , 10×100 , соответственно.
- Как известно, умножение матриц ассоциативно, поэтому порядок, в котором мы будем перемножать, на результат никак не влияет.
- Он, однако, может повлиять на время, необходимое для перемножения.

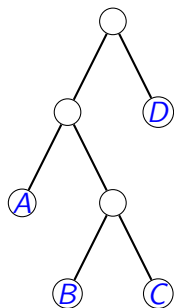
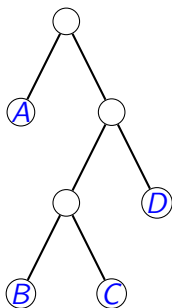
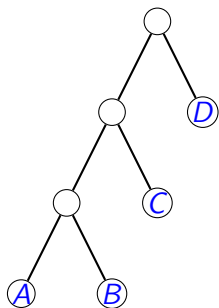
Перемножение нескольких матриц

Перемножение нескольких матриц

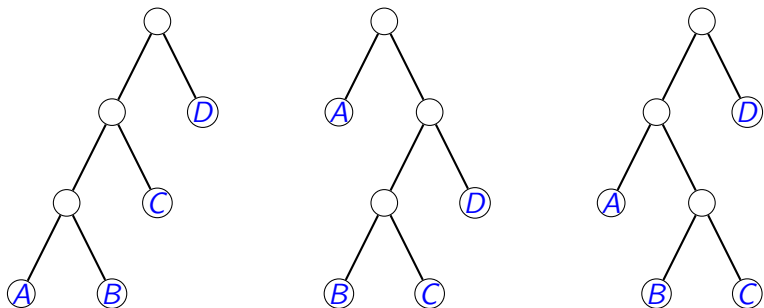
- Пусть нам нужно вычислить произведение $A \times B \times C \times D$ матриц размера 50×20 , 20×1 , 1×10 , 10×100 , соответственно.
- Как известно, умножение матриц ассоциативно, поэтому порядок, в котором мы будем перемножать, на результат никак не влияет.
- Он, однако, может повлиять на время, необходимое для перемножения.
- Простое перемножение матриц размера $m \times n$ и $n \times p$ требует mnp умножений.

порядок	количество перемножений
$A \times ((B \times C) \times D)$	120 000
$(A \times (B \times C)) \times D$	60 000
$(A \times B) \times (C \times D)$	7 000

Представление порядков бинарными деревьями



Представление порядков бинарными деревьями



Количество таких деревьев экспоненциально.

Подзадачи

Подзадачи

Подзадачи

Подзадачи

- Ясно, что у оптимального дерева поддеревья также оптимальны.

Подзадачи

Подзадачи

- Ясно, что у оптимального дерева поддеревья также оптимальны.
- В каждом поддереве вычисляется произведение вида $A_i \times A_{i+1} \times \dots \times A_j$.

Подзадачи

Подзадачи

- Ясно, что у оптимального дерева поддеревья также оптимальны.
- В каждом поддереве вычисляется произведение вида $A_i \times A_{i+1} \times \dots \times A_j$.
- Пусть $C(i, j)$ — минимальное количество умножений, необходимых для вычисления $A_i \times A_{i+1} \times \dots \times A_j$.

Подзадачи

Подзадачи

- Ясно, что у оптимального дерева поддеревья также оптимальны.
- В каждом поддереве вычисляется произведение вида $A_i \times A_{i+1} \times \dots \times A_j$.
- Пусть $C(i, j)$ — минимальное количество умножений, необходимых для вычисления $A_i \times A_{i+1} \times \dots \times A_j$.
- Тогда $C(i, j) = \min_{i \leq k < j} \{C(i, k) + C(k + 1, j) + m_{i-1} \cdot m_k \cdot m_j\}$.

Алгоритм

Алгоритм

```
1  for  $i \leftarrow 1$  to  $n$ 
2      do  $C(i, i) = 0$ 
3  for  $s \leftarrow 1$  to  $n - 1$ 
4      do for  $i \leftarrow 1$  to  $n - s$   $\triangleright s$  — размер подзадачи
5          do  $j = i + s$ 
6               $C(i, j) = \min_{i \leq k < j} \{ C(i, k) + C(k + 1, j) + m_{i-1} \cdot m_k \cdot m_j \}$ 
7  return  $C(1, n)$ 
```

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц
- 6 Кратчайшие пути**
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа
 - Задача о коммивояжере
- 7 Независимые множества в деревьях

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц
- 6 Кратчайшие пути**
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа
 - Задача о коммивояжере
- 7 Независимые множества в деревьях

Кратчайшие надежные пути

Определение

Задача о кратчайшем надежном пути (shortest reliable path problem) заключается в нахождении по данному взвешенному графу с двумя выделенными вершинами s и t , а также числу k кратчайшего пути из s в t , состоящего не более чем из k ребер.

Кратчайшие надежные пути

Определение

Задача о кратчайшем надежном пути (shortest reliable path problem) заключается в нахождении по данному взвешенному графу с двумя выделенными вершинами s и t , а также числу k кратчайшего пути из s в t , состоящего не более чем из k ребер.

Подзадачи

Кратчайшие надежные пути

Определение

Задача о кратчайшем надежном пути (shortest reliable path problem) заключается в нахождении по данному взвешенному графу с двумя выделенными вершинами s и t , а также числу k кратчайшего пути из s в t , состоящего не более чем из k ребер.

Подзадачи

- Подзадачи нужно выбрать так, чтобы каким-нибудь образом запоминалась информация о длине пути.

Кратчайшие надежные пути

Определение

Задача о кратчайшем надежном пути (shortest reliable path problem) заключается в нахождении по данному взвешенному графу с двумя выделенными вершинами s и t , а также числу k кратчайшего пути из s в t , состоящего не более чем из k ребер.

Подзадачи

- Подзадачи нужно выбрать так, чтобы каким-нибудь образом запоминалась информация о длине пути.
- $\text{dist}(v, i)$ есть длина кратчайшего пути, состоящего ровно из i ребер, из s в v

Кратчайшие надежные пути

Определение

Задача о кратчайшем надежном пути (shortest reliable path problem) заключается в нахождении по данному взвешенному графу с двумя выделенными вершинами s и t , а также числу k кратчайшего пути из s в t , состоящего не более чем из k ребер.

Подзадачи

- Подзадачи нужно выбрать так, чтобы каким-нибудь образом запоминалась информация о длине пути.
- $\text{dist}(v, i)$ есть длина кратчайшего пути, состоящего ровно из i ребер, из s в v
- $\text{dist}(v, i) = \min_{(u,v) \in E} \{\text{dist}(u, i-1) + l(u, v)\}$

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц
- 6 Кратчайшие пути**
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа**
 - Задача о коммивояжере
- 7 Независимые множества в деревьях

Кратчайшие пути между всеми парами вершин графа

Определение

Задача о кратчайших путях между всеми парами вершин (all-pairs shortest paths problem) заключается в нахождении по данному взвешенному графу кратчайших расстояний между всеми парами вершин. Предполагаем, что граф не содержит циклов отрицательного веса.

Кратчайшие пути между всеми парами вершин графа

Определение

Задача о кратчайших путях между всеми парами вершин (all-pairs shortest paths problem) заключается в нахождении по данному взвешенному графу кратчайших расстояний между всеми парами вершин. Предполагаем, что граф не содержит циклов отрицательного веса.

Подзадачи

Кратчайшие пути между всеми парами вершин графа

Определение

Задача о кратчайших путях между всеми парами вершин (all-pairs shortest paths problem) заключается в нахождении по данному взвешенному графу кратчайших расстояний между всеми парами вершин. Предполагаем, что граф не содержит циклов отрицательного веса.

Подзадачи

- $\text{dist}(i, j, k)$ — длина кратчайшего пути из i в j , все промежуточные вершины которого принадлежат множеству $\{1, \dots, k\}$.

Кратчайшие пути между всеми парами вершин графа

Определение

Задача о кратчайших путях между всеми парами вершин (all-pairs shortest paths problem) заключается в нахождении по данному взвешенному графу кратчайших расстояний между всеми парами вершин. Предполагаем, что граф не содержит циклов отрицательного веса.

Подзадачи

- $\text{dist}(i, j, k)$ — длина кратчайшего пути из i в j , все промежуточные вершины которого принадлежат множеству $\{1, \dots, k\}$.
- $\text{dist}(i, j, k) = \min\{\text{dist}(i, k, k-1) + \text{dist}(k, j, k-1), \text{dist}(i, j, k-1)\}$

Кратчайшие пути между всеми парами вершин графа

Определение

Задача о кратчайших путях между всеми парами вершин (all-pairs shortest paths problem) заключается в нахождении по данному взвешенному графу кратчайших расстояний между всеми парами вершин. Предполагаем, что граф не содержит циклов отрицательного веса.

Подзадачи

- $\text{dist}(i, j, k)$ — длина кратчайшего пути из i в j , все промежуточные вершины которого принадлежат множеству $\{1, \dots, k\}$.
- $\text{dist}(i, j, k) = \min\{\text{dist}(i, k, k-1) + \text{dist}(k, j, k-1), \text{dist}(i, j, k-1)\}$
- Время работы соответствующего алгоритма — $O(|V|^3)$.

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц
- 6 Кратчайшие пути**
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа
 - **Задача о коммивояжере**
- 7 Независимые множества в деревьях

Задача о коммивояжере

Определение

Задача о коммивояжере (traveling salesman problem) заключается в нахождении по данному полному взвешенному графу гамильтонова цикла минимальной стоимости.

Задача о коммивояжере

Определение

Задача о коммивояжере (traveling salesman problem) заключается в нахождении по данному полному взвешенному графу гамильтонова цикла минимальной стоимости.

Замечания

Задача о коммивояжере

Определение

Задача о коммивояжере (traveling salesman problem) заключается в нахождении по данному полному взвешенному графу гамильтонова цикла минимальной стоимости.

Замечания

- Количество различных циклов равно $(n - 1)!$.

Задача о коммивояжере

Определение

Задача о коммивояжере (traveling salesman problem) заключается в нахождении по данному полному взвешенному графу гамильтонова цикла минимальной стоимости.

Замечания

- Количество различных циклов равно $(n - 1)!$.
- Естественной подзадачей в данном случае является начальная часть цикла.

Задача о коммивояжере

Определение

Задача о коммивояжере (traveling salesman problem) заключается в нахождении по данному полному взвешенному графу гамильтонова цикла минимальной стоимости.

Замечания

- Количество различных циклов равно $(n - 1)!$.
- Естественной подзадачей в данном случае является начальная часть цикла.
- Об этой начальной части мы должны знать её последнюю вершину и множество внутренних вершин.

Задача о коммивояжере

Определение

Задача о коммивояжере (traveling salesman problem) заключается в нахождении по данному полному взвешенному графу гамильтонова цикла минимальной стоимости.

Замечания

- Количество различных циклов равно $(n - 1)!$.
- Естественной подзадачей в данном случае является начальная часть цикла.
- Об этой начальной части мы должны знать её последнюю вершину и множество внутренних вершин.
- Для подмножества вершин $S \subseteq \{1, \dots, n\}$, содержащего 1 и вершины $j \in S$ $C(S, j)$ есть длина кратчайшего пути, начинающегося в вершине 1 , заканчивающегося в вершине j и проходящего ровно по разу все вершины множества S .

Алгоритм

Алгоритм

TSP(G)

```
1   $C(\{1\}, 1) = 0$ 
2  for  $s \leftarrow 2$  to  $n$ 
3      do for  $S \subseteq \{1, 2, \dots, n\}$ , таких что  $|S| = s$  и  $1 \in S$ 
4          do  $C(S, 1) = \infty$ 
5             for  $j \in S, j \neq 1$ 
6                 do  $C(S, j) = \min_{i \in S, i \neq j} \{C(S \setminus \{j\}, i) + d_{ij}\}$ 
7  return  $\min_j C(\{1, \dots, n\}, j) + d_{j1}$ 
```

Алгоритм

Алгоритм

TSP(G)

```
1   $C(\{1\}, 1) = 0$ 
2  for  $s \leftarrow 2$  to  $n$ 
3      do for  $S \subseteq \{1, 2, \dots, n\}$ , таких что  $|S| = s$  и  $1 \in S$ 
4          do  $C(S, 1) = \infty$ 
5              for  $j \in S, j \neq 1$ 
6                  do  $C(S, j) = \min_{i \in S, i \neq j} \{C(S \setminus \{j\}, i) + d_{ij}\}$ 
7  return  $\min_j C(\{1, \dots, n\}, j) + d_{j1}$ 
```

Время работы

Время работы алгоритма есть $O(n^2 2^n)$. Полученный алгоритм хоть и гораздо быстрее полного перебора, но все же совершенно бесполезен на практике: экспоненциально не только время работы, но еще и память.

План лекции

- 1 Введение
- 2 Наибольшая возрастающая подпоследовательность
- 3 Стоимость редактирования
- 4 Задача о рюкзаке
- 5 Перемножение нескольких матриц
- 6 Кратчайшие пути
 - Кратчайшие надежные пути
 - Кратчайшие пути между всеми парами вершин графа
 - Задача о коммивояжере
- 7 Независимые множества в деревьях

Независимые множества в деревьях

Определение

Задача о максимальном независимом множестве (independent set problem) заключается в нахождении по данному графу множества попарно не соединенных ребрами вершин максимального размера.

Независимые множества в деревьях

Определение

Задача о максимальном независимом множестве (independent set problem) заключается в нахождении по данному графу множества попарно не соединенных ребрами вершин максимального размера.

Идеи

Независимые множества в деревьях

Определение

Задача о максимальном независимом множестве (independent set problem) заключается в нахождении по данному графу множества попарно не соединенных ребрами вершин максимального размера.

Идеи

- В общем случае задача NP-трудна, но если входной граф является деревом, то может быть решена за линейное время.

Независимые множества в деревьях

Определение

Задача о максимальном независимом множестве (independent set problem) заключается в нахождении по данному графу множества попарно не соединенных ребрами вершин максимального размера.

Идеи

- В общем случае задача NP-трудна, но если входной граф является деревом, то может быть решена за линейное время.
- Пусть $I(u)$ — размер максимального независимого множества в поддереве с корнем в u .

Независимые множества в деревьях

Определение

Задача о максимальном независимом множестве (independent set problem) заключается в нахождении по данному графу множества попарно не соединенных ребрами вершин максимального размера.

Идеи

- В общем случае задача NP-трудна, но если входной граф является деревом, то может быть решена за линейное время.
- Пусть $I(u)$ — размер максимального независимого множества в поддереве с корнем в u .

- $$I(u) = \max \left\{ 1 + \sum_{w - \text{внук } u} I(w), \sum_{w - \text{сын } u} I(w) \right\}$$

Об используемой памяти

Об используемой памяти

Об используемой памяти

Об используемой памяти

- Как правило, время работы алгоритма, основанного на методе динамического программирования, равно количеству вершин в соответствующем графе.

Об используемой памяти

Об используемой памяти

- Как правило, время работы алгоритма, основанного на методе динамического программирования, равно количеству вершин в соответствующем графе.
- Ясно, что объема памяти, пропорционального количеству вершин, будет достаточно, но иногда можно обойтись и меньшим объемом.

Об используемой памяти

Об используемой памяти

- Как правило, время работы алгоритма, основанного на методе динамического программирования, равно количеству вершин в соответствующем графе.
- Ясно, что объема памяти, пропорционального количеству вершин, будет достаточно, но иногда можно обойтись и меньшим объемом.
- Например, если для подсчета значения $\text{dist}(\cdot, \cdot, k + 1)$ необходимы только значения $\text{dist}(\cdot, \cdot, k)$, то нет никакой необходимости в каждый момент хранить значения для всех k .

Что мы узнали за сегодня?

Что мы узнали за сегодня?

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Метод динамического программирования применим тогда, когда решение задачи сводится к решению нескольких более простых перекрывающихся задач.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Метод динамического программирования применим тогда, когда решение задачи сводится к решению нескольких более простых перекрывающихся задач.
- Основным моментом метода является определение подзадач и способа построения решения из решений для более простых задач.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Метод динамического программирования применим тогда, когда решение задачи сводится к решению нескольких более простых перекрывающихся задач.
- Основным моментом метода является определение подзадач и способа построения решения из решений для более простых задач.
- По задаче, таким образом, строится ориентированный ациклический граф.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Метод динамического программирования применим тогда, когда решение задачи сводится к решению нескольких более простых перекрывающихся задач.
- Основным моментом метода является определение подзадач и способа построения решения из решений для более простых задач.
- По задаче, таким образом, строится ориентированный ациклический граф.
- Время работы, как правило, равно количеству вершин этого графа.

Что мы узнали за сегодня?

Что мы узнали за сегодня?

- Метод динамического программирования применим тогда, когда решение задачи сводится к решению нескольких более простых перекрывающихся задач.
- Основным моментом метода является определение подзадач и способа построения решения из решений для более простых задач.
- По задаче, таким образом, строится ориентированный ациклический граф.
- Время работы, как правило, равно количеству вершин этого графа.
- Используемая память может быть и меньше.

Спасибо за внимание!