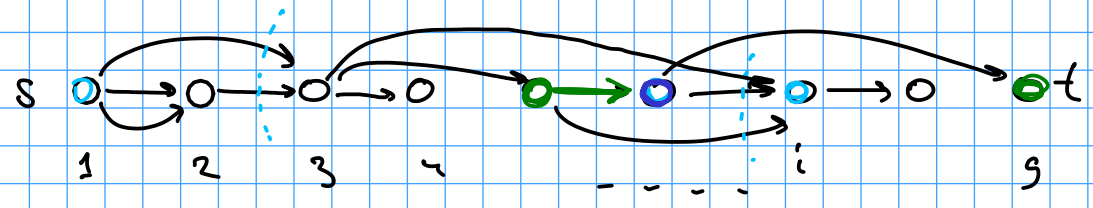
Динамическое программирование

Кратчайшие пути в ациклических граф



dist(s, t) - ?

d[1] = 0

- $d[i] = \min_{(v,i)\in E} [\, w(v,i) + d[v] \,]$ (*)

$O(|V| + |E|)$ $\quad dist(s,t) = d[t]$

Шаги решения задачи динамическим програм.

1. Выделение подзадачу
2. Определить соотношение на подзадачах
3. Порядок вычисление подзадач.

Задача о самой длинной возраст. подпосл.

M

2	3	7	4	10	15	8	6

1. L[i] - длина самой большой ↑↑ подпосл., которая заканчивается в ячейке с № i
2. $L[1] = 1 \quad L[i] = \max_{\substack{j<i \\ M[j]<M[i]}} [L[j]] + 1$ $\quad O(n^2)$
3. Порядок: слева направо. $\quad O(n \log n)$

NB: соотношение задаёт граф на подзадачах

Задача о рюкзаке

Рюкзак вместимости W

В хранилище хранятся предметы:

ценность	v_1	v_2	...	v_n
вес/объём	w_1	w_2	...	w_n

Вопрос: множество товаров I:

$$\sum_{i \in I} w_i \leq W, \quad \sum_{i \in I} v_i \to \max$$

1. "непрерывный рюкзак"

Жадный алгоритм

из соображений удельной стоимости $\frac{v_i}{w_i}$

2. дискретный рюкзак с повторениями

Веса целые.

$W = 10$ $\quad w_1 = 6 \quad v_1 = 24$

$w_2 = 5 \quad v_2 = 15$

Жадное решение: взять 1 й товар. Итого 24.

Оптимальное решение: 2 товара №2. Итого 30

1. $V[i]$ - оптимальное решение для рюкзака размером i

2. $V[0] = 0$

$$V[i] = \max_{\substack{1 \leq k \leq n \\ w_k \leq i}} [V[i - w_k] + v_k]$$

3. Порядок вычислений $O(Wn)$

3. дискретный рюкзак без повторений

1. $V[i, k]$ – оптимальное решение для рюкзака веса i и первых k предметов

2. $V[0, k] = 0$

$$V[i, k] = \max[V[i, k-1], V[i-w_k, k-1] + v_k]$$

k-й предмет отсутствует

k-ый предмет есть в решении

3. Порядок

for $k = 1 \dots n$

for $i = 1 \dots W$

$V[i, k] = \dots$

$O(nW)$

Памяти $O(nW)$

Можно $O(W)$ *

Перемножение матриц

Вход: $A_1 \dots A_n$

Хотим вычислить $A_1 \times A_2 \times \dots \times A_n$

$A \times B \times C$

$10 \times 20 \quad 20 \times 40 \quad 40 \times 100$

1. $(A \times B) \times C = 10 \cdot 20 \cdot 40 + 10 \cdot 40 \cdot 100$ — 48000

2. $A \times (B \times C) = 20 \cdot 40 \cdot 100 + 10 \cdot 20 \cdot 100$ — 100000

$C = A \times B$ – потребуется $n \cdot k \cdot m$ умножений

$n \times k \quad n \times m \quad m \times k$

Задача: вычислить наименьшее кол-во операций для умножения n матриц.

Вход: $m_1, m_2, m_3 \dots m_{n+1}$

$(m_1 \times m_2) \times (m_2 \times m_3) \dots (m_n \times m_{n+1})$

1. $C[i, j]$ – стоимость операций для перемножения $A_i \times \dots \times A_j$

2. $C[i, i] = 0$

$$(A_i \times A_{i+1}) \overset{k}{\times} (\dots \times A_j)$$

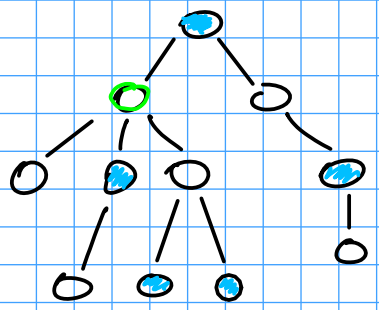
$$C[i, j] = \min_{i \le k < j} \left[C[i, k] + C[k+1, j] + m_i \cdot m_{k+1} \cdot m_{j+1} \right]$$

Время $O(n^3)$ Память $O(n^2)$

3. Порядок

```
for s = 1 to n
    for i = 1 .. n - s
        C[i, i + s] = ....
```

Максимальное независимое множество в дереве



IS – independent set

никакие два вершины не соединены рёбрами.

Жадный алгоритм и дин. прогр.

1. $IS[v]$ – макс. нез. мн-во для поддерева с корнем в v.

2. $IS[l] = 1$, l – лист

$$IS[v] = \max \left[\sum_{w\text{-ребёнок } v} IS[w], \; 1 + \sum_{w\text{ - внук } v} IS[w] \right]$$

3. "от листьев к корню"

Замечания:

1. Некоторые из таких алгоритмов можно реализовать рекурсивно + "memoisation"
   ⇒ эффективный алгоритм
2. Как восстановить оптимальное решение?
   Давайте в каждой подзадаче запоминать также те подзадачи, на которых мы получили оптимум.
   Можно пройти по такому пути с конца.