
АДРИАНОВ Н.М.
ИВАНОВ А.Б.

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

ЗАДАЧА О РЮКЗАКЕ: ДИНАМИКА

Дано:

- Рюкзак максимальной вместимости W (по весу)
- n предметов, для которых задан вес w_i и ценность v_i

Найти набор предметов $S \subset \{1, 2, \dots, n\}$ такой, что

$$\sum_{i \in S} w_i \leq W \quad \text{и} \quad \sum_{i \in S} v_i \rightarrow \max$$

Пусть W и все w_i – целые числа.

Подзадачи для динамики:

$$S \subset \{1, 2, \dots, k\} \quad \text{и} \quad \sum_{i \in S} w_i \leq x$$

Пусть $A[k, x]$ – оптимальное решение для такой задачи.

Тогда

$$A[k, x] = \max\left(A[k-1, x], A[k-1, x-w_k] + v_k\right)$$

ЗАДАЧА О РЮКЗАКЕ: ДИНАМИКА

```
procedure KnapsackDynamic( v, w )  
  
  A = 2-мерный массив  
  for x = 0,1,2,...W  
    A[0,x] = 0  
  
  for i = 1,2,...n:  
    for x = 0,1,2,...W:  
      A[i,x] = max( A[i-1,x], A[i-1,x-w[i]] + v[i] )  
  
  return A[n,W]  
end
```

Сложность: $O(nW)$

ЗАДАЧА О РЮКЗАКЕ: ЕЩЕ ОДНА ДИНАМИКА

Пусть стоимости v_i – целые числа.

Подзадачи: для $i = 0, 1, 2 \dots n$ и $x = 0, 1, 2, \dots n \cdot v_{max}$
 $A[i, x]$ = минимально нужный размер для того, чтобы достичь стоимости $\geq x$, используя только первые i предметов.

Рекуррентное соотношение:

$$A[i, x] = \min\left(A[i - 1, x], w_i + A[i - 1, x - v_i]\right)$$

(считаем $A[i - 1, x - v_i] = 0$, если $x \leq v_i$).

ЗАДАЧА О РЮКЗАКЕ: ЕЩЕ ОДНА ДИНАМИКА

```
procedure KnapsackDynamic2( W, v, w )
  A = 2-мерный массив
  A[0, x] = 0, если  $x = 0$ 
              $+\infty$ , иначе

  for i = 1, 2, ... n
    for x = 0, 1, ... n*v_max
      A[i, x] = min ( A[i-1, x], w[i] + A[i-1, x - v[i]] )

  return максимальное x такое, что A[n, x] <= W.
```

Сложность: $O(n^2 v_{max})$

ЗАДАЧА КОММИВОЯЖЁРА

Коммивояжёр находится в одном из n городов, хочет их все объехать кратчайшим путем.

Дано: полный неориентированный граф с неотрицательными длинами ребер d_{ij} .

Требуется: найти обход всех вершин по 1 разу (гамильтонов цикл) минимальной длины.

Полный перебор: $(n - 1)!$

ЗАДАЧА КОММИВОЯЖЁРА: ДИНАМИКА

Города: $\{1, 2, \dots, n\}$

Подзадачи: для каждого $S \subseteq \{1, 2, \dots, n\}$ и $j \in S$

$A[S, j]$ = кратчайший путь из 1 в j ,
проходящий через все города в S ровно по 1 разу

$$A[S, j] = \min_{i \in S, i \neq j} (A[S \setminus \{j\}, i] + d_{ij})$$

ЗАДАЧА КОММИВОЯЖЁРА: ДИНАМИКА

```
procedure TSP( p )
  A[{1}, 1] = 0
  for s = 2 .. n:
    for  $S \subseteq \{1, 2, \dots, n\}$ ,  $1 \in S$ ,  $|S| = s$ :
      A[S, 1] =  $\infty$ 
      for  $j \in S$ ,  $j \neq 1$ :
        A[S, j] = min ( A[S \ {j}, i] + d[i, j] )
          (среди всех  $i \in S$ ,  $i \neq j$ )
  return min ( A[{1, 2, \dots, n}, j] + d[j, 1] )
end
```

Сложность: $O(n^2 2^n)$

n	$n^2 2^n$	$(n - 1)!$
5	800	24
10	102400	362880
15	7372800	87178291200
20	419430400	121645100408832000
25	20971520000	620448401733239439360000

ПРОИЗВЕДЕНИЕ МАТРИЦ

Умножение матрицы $m \times n$ на матрицу $n \times p$ выполняется за $O(mnp)$. Предположим, что «стоимость» умножения равна mnp .

Даны матрицы $m_0 \times m_1, m_1 \times m_2, \dots, m_{n-1} \times m_n$, каким способом их надо перемножать, чтобы сделать это «дешевле»?

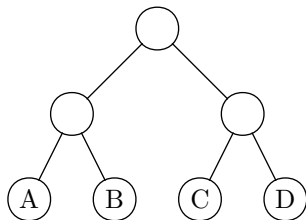
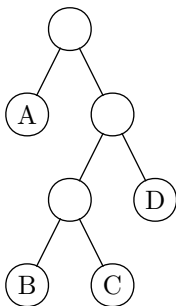
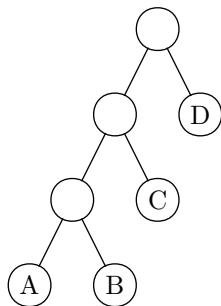
$$A = (50 \times 20) \quad B = (20 \times 1) \quad C = (1 \times 10) \quad D = (10 \times 100)$$

$$A \times ((B \times C) \times D)$$

$$(A \times (B \times C)) \times D$$

$$(A \times B) \times (C \times D)$$

ПРОИЗВЕДЕНИЕ МАТРИЦ



$C[i, j]$ = минимальная стоимость вычисления $A_i \times \dots \times A_j$

$$C[i, j] = \min_{i \leq k < j} \left(C[i, k] + C[k + 1, j] + m_{i-1} \cdot m_k \cdot m_j \right)$$

ОПТИМАЛЬНЫЕ БИНАРНЫЕ ДЕРЕВЬЯ ПОИСКА

Сбалансированные бинарные деревья минимизируют время поиска случайного элемента в дереве, если все элементы в дереве ищутся равновероятно. Время поиска = глубина элемента в дереве.

Дано: веса (вероятности) p_i , $i = 1, 2 \dots n$.

Требуется: построить бинарное дерево поиска T , минимизирующее величину

$$C(T) = \sum_i p_i * [\text{глубина элемента } i \text{ в } T].$$

Похоже на код Хаффмана. Однако есть существенные отличия:

- код Хаффмана prefix-free, т.е. символы находятся только в листьях дерева
- в коде Хаффмана не требуется, чтобы дерево было деревом поиска (упорядоченность)

ОПТИМАЛЬНЫЕ БИНАРНЫЕ ДЕРЕВЬЯ ПОИСКА

Пусть

T – оптимальное дерево поиска, в вершине которого находится i ,
 T_1 – левое поддерево, T_2 – правое поддерево.

Тогда

T_1 – оптимальное дерево для $1, 2, \dots, (i - 1)$,

T_2 – оптимальное дерево для $(i + 1) \dots n$.

Пусть C_{ij} – вес оптимального дерева для $\{i, i + 1, \dots, j\}$. Тогда

$$C_{ij} = \min_{i \leq r \leq j} \left(\sum_{k=i}^j p_k + C_{i(r-1)} + C_{(r+1)j} \right)$$

(считаем $C_{xy} = 0$, если $x > y$).

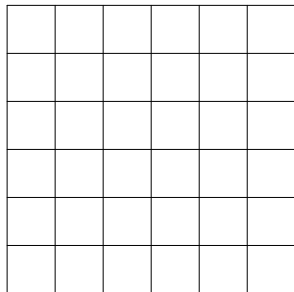
ОПТИМАЛЬНЫЕ БИНАРНЫЕ ДЕРЕВЬЯ ПОИСКА

```
procedure OptimalBST( p )
  A = 2-мерный массив n*n
  for s = 0 .. n-1:
    for i = 1 .. n-s:
      A[i, i+s] = min (  $\sum_{k=i}^{i+s} p_k + A[i, r-1] + A[r+1, i+s]$  )
                      (среди всех r: i <= r <= i+s)
end
```

Сложность: $O(n^3)$

ДИНАМИКА ПО ПРОФИЛЮ: ДОМИНО

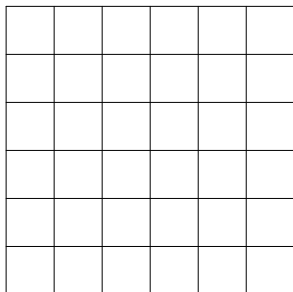
Дана доска $m \times n$. Найти количество различных ее замощений при помощи домино 1×2 .



$A[k, p]$ = количество замощений таких, что полностью заполнены k столбцов, в $k + 1$ покрыты клетки, соответствующие профилю p , но не лежит ни одной доминошки целиком.

ДИНАМИКА ПО ПРОФИЛЮ: КОРОЛИ

Дана доска $m \times n$. Найти количество различных способов расставить k королей так, чтобы они не били друг друга.



$A[i, p, j]$ = количество способов расставить j королей на первых i столбцах так, что p - профиль для i -го столбца.