Адрианов Н.М.
Иванов А.Б.

Алгоритмы и структуры данных

Динамическое программирование

Динамическое программирование

Термин: Ричард Беллман, 1950-е годы

The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word, research. I'm not using the term lightly; I'm using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term, research, in his presence. You can imagine how he felt, then, about the term, mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. What title, what name, could I choose? In the first place I was interested in planning, in decision making,... But planning, is not a good word for various reasons. I decided therefore to use the word, "programming"... [Dynamic] has a very interesting property as an adjective, and that is its impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. Its impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities.

Максимальное независимое множество

Maximal Weighted Independent Set (MAX-WIS)

Множество вершин графа называется *независимым*, если в нем нет соседних вершин.

Дано: граф, в каждой вершине задан *неотрицательный* вес.

Требуется: найти независимое множество вершин с максимальной суммой весов.

Задача сложная (NP-полная). Однако если граф – дерево, то существует алгоритм, работающий за линейное время.

MAX-WIS в дереве

Пусть граф G = дерево-цепочка $\circ$–$\circ$–$\circ$–$\circ$ с n вершинами.
В вершинах заданы веса $w_i \geq 0$, $i = 1, ...n$.
Как решить задачу о максимальном независимом множестве?

MAX-WIS в дереве

Пусть граф G = дерево-цепочка ○–○–○–○ с n вершинами.
В вершинах заданы веса $w_i \geq 0$, $i = 1, ...n$.
Как решить задачу о максимальном независимом множестве?

Пусть G' – дерево с удаленной вершиной n,
G'' – дерево с удаленными вершинами n и $n-1$.

- Если вершина n не входит в оптимальное решение, то оптимальное решение является оптимальным решением для G'.
- Если вершина n входит в оптимальное решение, то оптимальное решение получается из оптимального решения для G'' добавлением вершины n.

MAX-WIS в дереве: динамика

Обозначим сумму весов в оптимальном решении MAX-WIS(G). Тогда

$$\text{MAX-WIS}(G) = \max\Big(\text{MAX-WIS}(G'),\ \text{MAX-WIS}(G'') + w_n\Big)$$

- Итак, задача легко решается, если известны решения двух задач меньшего размера.

MAX-WIS в дереве: динамика

Обозначим сумму весов в оптимальном решении MAX-WIS(G). Тогда

$$\text{MAX-WIS}(G) = \max\Big(\text{MAX-WIS}(G'),\ \text{MAX-WIS}(G'') + w_n\Big)$$

- Итак, задача легко решается, если известны решения двух задач меньшего размера.
- Нельзя использовать рекурсию!

MAX-WIS в дереве: динамика

Обозначим сумму весов в оптимальном решении MAX-WIS(G). Тогда

$$\text{MAX-WIS}(G) = \max\Big(\text{MAX-WIS}(G'),\ \text{MAX-WIS}(G'') + w_n\Big)$$

- Итак, задача легко решается, если известны решения двух задач меньшего размера.
- Нельзя использовать рекурсию!
- Решаем последовательно задачи для G_1, G_2, …, G_n.
- Формула дает максимальный вес, а как восстановить максимальное множество?

MAX-WIS в дереве: динамика

Обозначим сумму весов в оптимальном решении MAX-WIS(G). Тогда

$$\text{MAX-WIS}(G) = \max\Big(\text{MAX-WIS}(G'),\ \text{MAX-WIS}(G'') + w_n\Big)$$

- Итак, задача легко решается, если известны решения двух задач меньшего размера.
- Нельзя использовать рекурсию!
- Решаем последовательно задачи для G_1, G_2, …, G_n.
- Формула дает максимальный вес, а как восстановить максимальное множество?
  - можно хранить для каждой решенной задачи множество (память $= O(n^2)$)
  - но лучше хранить признаки – включается ли вершина k в решение задачи G_k, тогда максимальное множество восстанавливается за линейное время.

Наибольшая возраст. последовательность

Дано: последовательность чисел $a_1, a_2, \ldots a_n$

Найти: самую длинную возрастающую подпоследовательность $a_{i_1}, a_{i_2}, \ldots a_{i_k}$:

$$i_1 < i_2 < \ldots < i_k$$

$$a_{i_1} < a_{i_2} < \ldots < a_{i_k}$$

5 2 8 6 3 6 9 7 3

Наибольшая возраст. последовательность

5 2 8 6 3 6 9 7 3

$$A[j] = \text{максимальная длина возрастающей подпоследовательности, заканчивающейся в } j$$

$$A[j] = 1 + \max\{A[i] : i < j,\, a_i < a_j\}$$

Решение начальной задачи: $\max(A[j])$

Идея динамического программирования

1. исходная задача включается в семейство задач;
2. решение каждой задачи выражается через решения задач “меньшего размера” ;
3. решаем задачи последовательно в правильном порядке.

1-мерная динамика

- Максимальное независимое множество в дереве
- Наибольшая возрастающая последовательность

Далее: 2-мерная динамика

- Расстояние редактирования
- Задача о рюкзаке
- Задача коммивояжёра (динамика по подмножествам)
- Замощение домино (динамика по профилю)

Выравнивание (расстояние редактирования)

(Sequence alignment, minimal edit distance)

Дано:

- слова $X = x_1 x_2 \dots x_m$ и $Y = y_1 y_2 \dots y_n$
- стоимость замены α_{ab} для каждой пары букв a, b
- стоимость вставки буквы α_+, стоимость удаления α_-

Выравнивание (расстояние редактирования)

(Sequence alignment, minimal edit distance)

Дано:

- слова $X = x_1 x_2 \dots x_m$ и $Y = y_1 y_2 \dots y_n$
- стоимость замены α_{ab} для каждой пары букв a, b
- стоимость вставки буквы α_+, стоимость удаления α_-

Пример: $X =$ `AGGGCT`, $Y =$ `AGGCAC`

```
AGGGCT-
AGG-CAC
```

стоимость $= \alpha_- + \alpha_{TA} + \alpha_+$

Требуется: найти расстояние редактирования = выравнивание последовательностей минимальной стоимости.

(расстояние Левенштейна: $\alpha_- = \alpha_+ = \alpha_{xy} = 1$ при $x \neq y$)

Выравнивание: динамика

$X_i :=$ первые i букв слова X
$Y_j :=$ первые j букв слова Y
$A[i, j] :=$ стоимость оптимального выравнивания X_i и Y_j.

$$A[i,j] = \min\Big(\alpha_{x_i y_j} + A[i-1, j-1],\ \alpha_- + A[i-1, j],\ \alpha_+ + A[i, j-1]\Big)$$

Выравнивание: пример

$$A[i,j] = \min\Big(\delta_{x_i y_j} + A[i-1, j-1], \quad 1 + A[i-1, j], \quad 1 + A[i, j-1]\Big)$$

		S	I	T	T	I	N	G
K								
I								
T								
T								
E								
N								

Выравнивание: динамика

```
procedure SequenceAlignment( X, Y, α )
  A = 2-мерный массив размера)
  for i = 0,1,2,...m:
    A[i,0] = i*α_
  for i = 0,1,2,...n:
    A[0,i] = i*α+

  for i = 1,2,...m:
    for j = 1,2,...n:
       A[i,j] = min( A[i-1,j-1] + α_{x_i y_j},
                     A[i-1,j] + α_,
                     A[i,j-1] + α+)
  return A[m,n]
end
```

Сложность: $O(mn)$

При заполненном массиве A восстанавливаем оптимальное выравнивание, двигаясь в обратном направлении, за $O(m+n)$.

Задача о рюкзаке: динамика

- 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

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

Пусть W и все w_i – целые числа.

Задача о рюкзаке: динамика

- 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

$$\sum_{i \in S} w_i \le W \quad \text{и} \quad \sum_{i \in S} v_i \to max$$

Пусть W и все w_i – целые числа.

Подзадачи для динамики:

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

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

Тогда

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

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 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: $\{1, 2, \ldots n\}$

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

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

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

Задача коммивояжёра: динамика

```
procedure TSP( p )
  A[{1}, 1] = 0
  for s = 2 .. n:
    for S ⊆ {1,2,...n}, 1 ∈ S, |S| = s:
      A[S, 1] = ∞
      for j ∈ S, j ≠ 1:
        A[S, j] = min (A[S \ {j}, i] + d[i,j])
                    (среди всех i ∈ S, i ≠ j)
  return min (A[{1,2,...n}, j] + d[j,1]
end
```

Задача коммивояжёра: динамика

```
procedure TSP( p )
  A[{1}, 1] = 0
  for s = 2 .. n:
    for S ⊆ {1, 2, ... n}, 1 ∈ S, |S| = s:
      A[S, 1] = ∞
      for j ∈ S, j ≠ 1:
        A[S, j] = min (A[S \ {j}, i] + d[i,j])
                  (среди всех i ∈ S, i ≠ j)
  return min (A[{1, 2, ... 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	6204484017332394393600000

Произведение матриц

Умножение матрицы $m \times n$ на матрицу $n \times p$ выполняется за ...

Произведение матриц

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

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

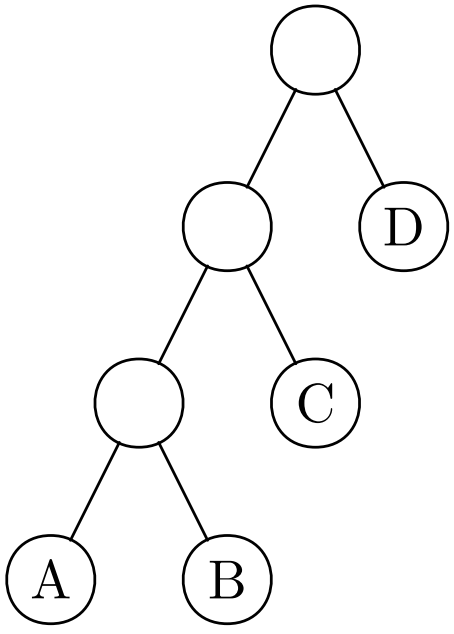
$A = (50 \times 20)$ $B = (20 \times 1)$ $C = (1 \times 10)$ $D = (10 \times 100)$

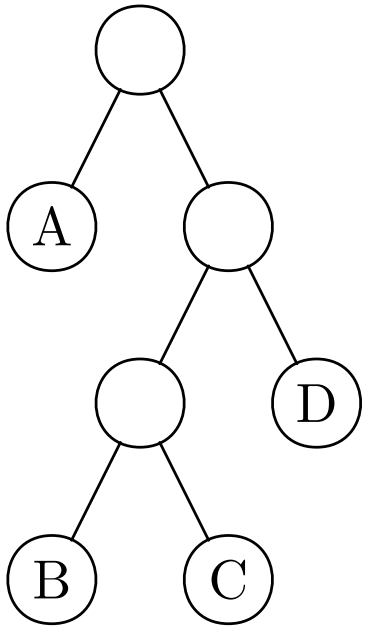
$A \times ((B \times C) \times D)$

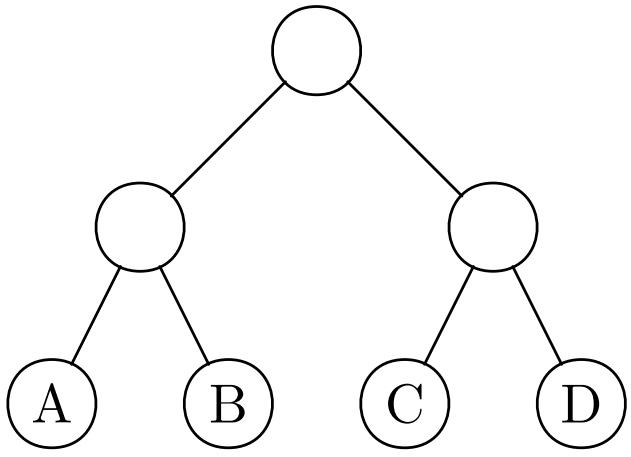
$(A \times (B \times C)) \times D$

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

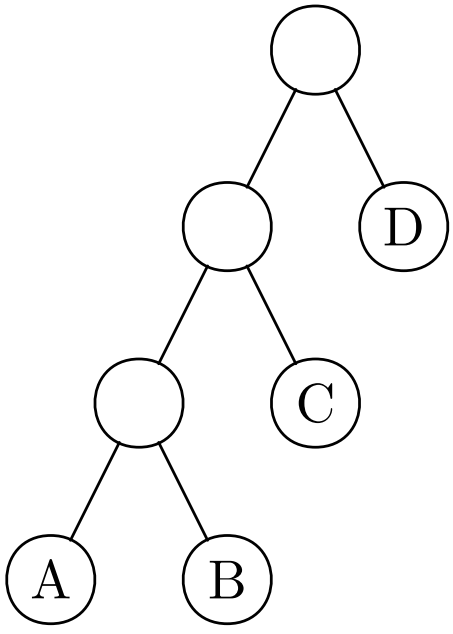
Произведение матриц

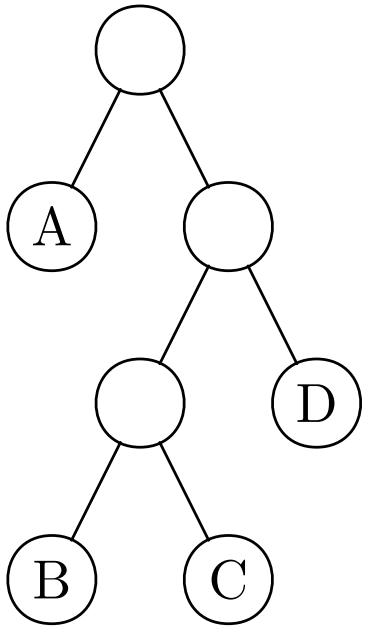


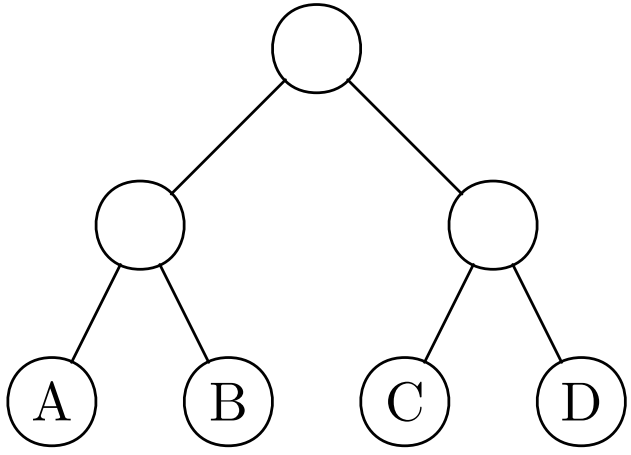




Произведение матриц







$C[i,\, j] =$ минимальнасть стоимость вычисления $A_i \times \ldots \times A_j$

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

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 $1, 2, \ldots (i-1)$,
T_2 - оптимальное дерево для $(i+1) \ldots n$.

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

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

(считаем $C_{xy} = 0$, если $x > y$).

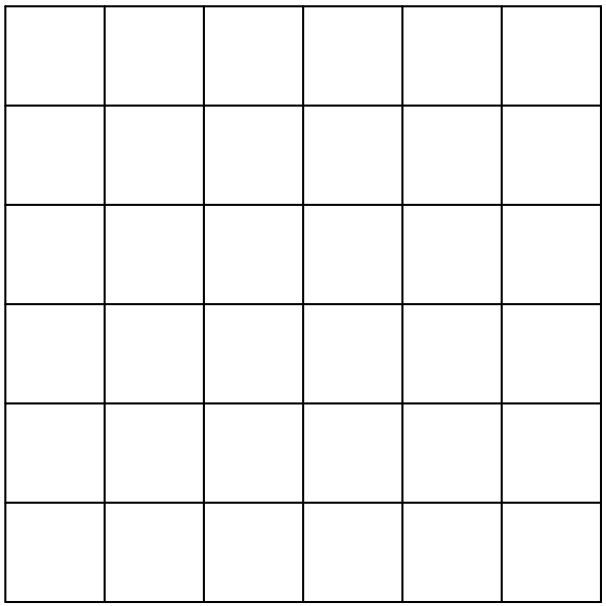
Оптимальные бинарные деревья поиска

```
procedure OptimalBST( p )
  A = 2-мерный массив n*n
  for s = 0 .. n-1:
    for i = 1 .. n-s:
      A[i, i+s] = min (∑_{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

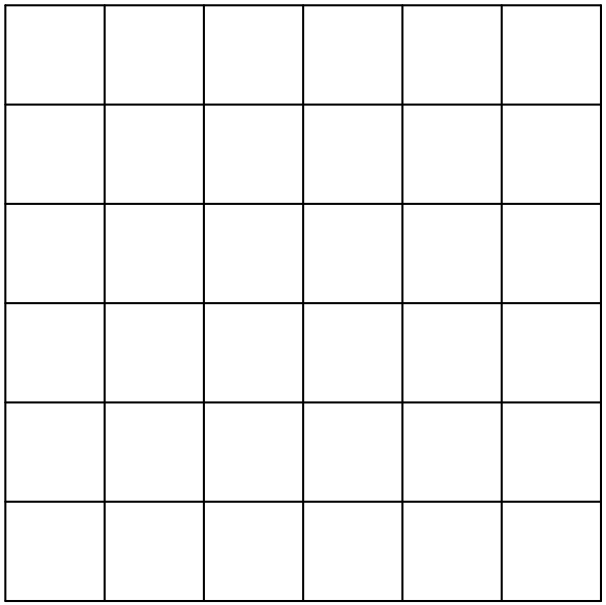
Дана доска $m \times n$. Найти количество различных ее замощений при помощи домино 1×2.



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

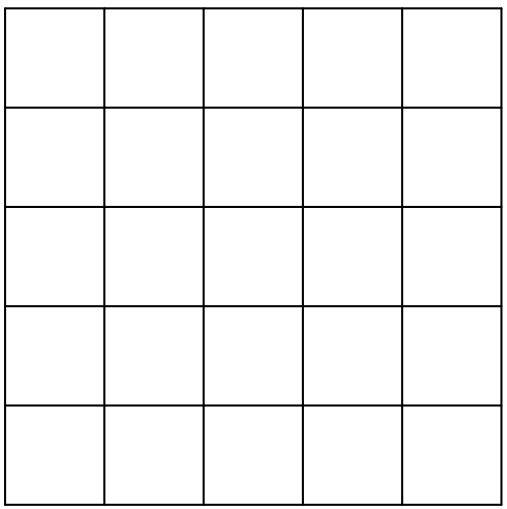
Динамика по профилю: короли

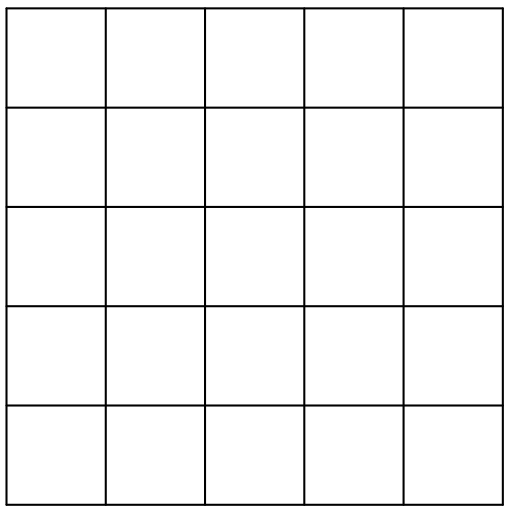
Дана доска $m \times n$. Найти количество различных способов расставить королей так, чтобы они не били друг друга.



Динамика по профилю: “ЕГЭ”

Дана таблица $m \times n$. Каждую клетку можно либо оставить пустой, либо зачеркнуть по диагонали. Нельзя, чтобы две диагонали выходили из одной вершины. Требуется найти максимально возможное количество зачеркнутых вершин.





Динамика по изломанному профилю