5. Динамическое программирование

Об идее, как и о привидении,…

следует немного

поговорить, прежде

чем она явится.

Ч. Диккенс

Динамическое программирование – метод проектирования алгоритмов.

Предложен американским математиком Ричардом Беллманом как <u>общий метод</u> оптимизации многостадийных процессов принятия решений.

Динамическое программирование – метод решения задач с перекрещивающимися подзадачами.

Иллюстрация метода на примере чисел Фибоначчи

0, 1, 1, 2, 3, 5, 8, 13…

$F(n)=F(n-1) + F(n-2),\ n \geq 2 \qquad (1)$

Начальные условия:

$F(0)=0,\ F(1)=1 \qquad (2)$

5.1. Вычисление биномиальных коэффициентов

Биномиальным коэффициентом C(n, k) называется количество комбинаций (подмножеств) из k элементов из n-элементного множества ($0 \le k \le n$).

Название «биномиальные коэффициенты» происходит от участия этих чисел в формуле бинома:

$(a+b)^n = C(n,0)a^n + \ldots + C(n,k)a^{n-k}b^k + \ldots + C(n,n)b^n$

$C(n,k) = C(n-1,k-1) + C(n-1,k)$ при $n>k>0$ (3)

$C(n,0)=C(n,n)=1$ (4)

Таблица для вычислений биномиальных коэффициентов

	0	1	2	…	k-1	k
0	1					
1	1	1				
2	1	2	1			
…						
k	1					1
…						
n-1	1				C(n-1,k-1)	C(n-1,k)
n	1					C(n,k)

Алгоритм Binomial (n,k)

// Вх. данные: Пара неотрицательных чисел $n \geq k \geq 0$

// Bsx. данные: Значение C(n,k)

```
for i←0 to n do
   for j ←0  to min(i,k) do
      if j=0 or j=I
         C[i,j] ←1
      else
         C[i,j] ← C[i-1,j-1] + C[i-1,j]
return C(n,k)
```

Оценка эффективности алгоритма вычисления биномиальных коэффициентов

Базовая операция – сложение.

A(n,k) – общее количество сложений при вычислении C(n,k).

$$A(n,k)=\sum_{i-1}^{k}\ \sum_{j=1}^{i-1} 1 + \sum_{i=k+1}^{n}\ \sum_{j=1}^{k} 1 = \sum_{i=1}^{k}(i-1) + \sum_{i=k+1}^{n} k =$$

$$=[(k-1)k]/2+k(n-k) \in O(n\times k)$$

5.2. Задача о рюкзаке и функции с запоминанием

<u>Дано</u>: Рюкзак вместимостью W

Количество предметов: n

Веса предметов: $w_1, w_2, \ldots, w_n$

Стоимости предметов: $v_1, v_2, \ldots, v_n$

<u>Требуется найти</u>: наиболее ценное подмножество, помещающееся в рюкзаке.

Экземпляр задачи, определяемый первыми i предметами ($1 \le i \le n$)

Веса: $w_1, w_2, \ldots, w_n$

Стоимости: $\mathbf{v_1, v_2, \ldots, v_n}$

Ёмкость рюкзака: $\mathbf{1 \le j \le W}$.

V[i,j] – значение **оптимального** решения этого экземпляра задачи, т.е. **стоимость наиболее ценного подмножества предметов из первых i предметов, которые помещаются в рюкзак ёмкости j.**

Все подмножества первых i предметов, которые помещаются в рюкзак ёмкостью j делятся на две категории:

1. те, которые **не включают** i-й предмет;
2. те, которые его **включают**.

Замечания.

2. Среди подмножеств (ПМ), которые не включают i-й предмет, стоимость оптимального ПМ – $\mathbf{V[i\text{-}1,j]}$.
3. Среди ПМ, которые включают i-й предмет, оптимальное ПМ составляется из оптимального ПМ из этого предмета и первых **(i-1)** предметов, которые размещаются в рюкзаке ёмкостью $\mathbf{(j-w_i)}$. Стоимость такого оптимального ПМ равна $\mathbf{v_i+V[i\text{-}1,j-w_i]}$.

Рекуррентное соотношение

$$V[i,j] = \begin{cases} \max\{V[i-1,j],\ v_i + V[i-1,\ j-w_i]\}, & \text{если } j-w_i \geq 0 \\ V[i-1,j], & \text{если } j-w_i < 0 \end{cases}$$

<u>**Начальные условия**</u>: $V[0,j]=0$ при $j \geq 0$

$V[i,0]=0$ при $i \geq 0$

<u>**Цель**</u>: Найти **V[n,W]**, т.е. максимальную стоимость подмножества из **n** предметов, которое помещается в рюкзак ёмкостью **W**, и само это подмножество.

Таблица для решения задачи о рюкзаке методом динамического программирования

	0	…	$j-w_i$	j	…	W
0	0		0	0		0
…						
w_i, v_i (i-1)	0		$V[i-1, j-w_i]$	$V[i-1, j]$		
	0			$V[i, j]$		
n	0					Целевое значение

Пример 1. W=5

Предмет	Вес	Стоимость
1	2	12
2	1	10
3	3	20
4	2	15

Ёмкость j

	i	0	1	2	3	4	5
	0	0	0	0	0	0	0
$w_1 = 2$ $v_1 = 12$	1	0	0	12	12	12	12
$w_2 = 1$ $v_2 = 10$	2	0	10	12	22	22	22
$w_3 = 3$ $v_3 = 20$	3	0	10	12	22	30	32
$w_4 = 2$ $v_4 = 15$	4	0	10	15	25	30	**37**

Максимальная стоимость: V[4,5]=37.

1.Поскольку V[4,5]≠ V[3,5], то предмет 4 был <u>включен</u> в решение вместе с оптимальным подмножеством, заполняющим оставшиеся 5-2=3 единицы ёмкости рюкзака. Это элемент V[3,3].

2.Поскольку V[3,3]= V[2,3], то предмет 3 не является частью оптимального подмножества.

3. Поскольку V[2,3] ≠ V[1,3], то предмет 2 также является частью оптимального подмножества.

4. V[1,3-1] остается в качестве определения оставшейся части подмножества. Т.к. V[1,2] ≠ V[0,2], то предмет 1 входит в оптимальное подмножество.

Эффективность алгоритма $\in$ **O(n*W)**

Время поиска оптимального подмножества $\in$ **O(n+W)**.