

# 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 общий метод 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...

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

Начальные условия:

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

## 5.1. Вычисление биномиальных коэффициентов

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

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

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

$$C(n,k) = C(n-1,k-1) + C(n-1,k) \quad \text{при } n > k > 0 \quad (3)$$

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

# Таблица для вычислений биномиальных коэффициентов

	<b>0</b>	<b>1</b>	<b>2</b>	<b>...</b>	<b>k-1</b>	<b>k</b>
<b>0</b>	1					
<b>1</b>	1	1				
<b>2</b>	1	2	1			
<b>...</b>						
<b>k</b>	1					1
<b>...</b>						
<b>n-1</b>	1				<b>C(n-1,k-1)</b>	<b>C(n-1,k)</b>
<b>n</b>	1					<b>C(n,k)</b>

Алгоритм Binomial (n,k)

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

// Вых. данные: Значение  $C(n,k)$

**for**  $i \leftarrow 0$  **to**  $n$  **do**

**for**  $j \leftarrow 0$  **to**  $\min(i,k)$  **do**

**if**  $j=0$  **or**  $j=i$

$C[i,j] \leftarrow 1$

**else**

$C[i,j] \leftarrow C[i-1,j-1] + C[i-1,j]$

**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

Дано: Рюкзак вместимостью  $W$

Количество предметов:  $n$

Весы предметов:  $w_1, w_2, \dots, w_n$

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

Требуется найти: наиболее ценное

подмножество, помещающееся в рюкзаке.



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

**Весы:**  $W_1, W_2, \dots, W_n$

**Стоимости:**  $V_1, V_2, \dots, V_n$

**Ёмкость рюкзака:**  $1 \leq j \leq W$ .

$V[i, j]$  – 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 –  $V[i-1, j]$ .
3. Среди ПМ, которые **включают**  $i$ -й предмет, оптимальное ПМ состоит из оптимального ПМ из этого предмета и первых  $(i-1)$  предметов, которые размещаются в рюкзаке ёмкостью  $(j-w_i)$ . Стоимость такого оптимального ПМ равна  $v_i + V[i-1, j-w_i]$ .

## Рекуррентное соотношение

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

**Начальные условия:**  $V[0,j]=0$  при  $j \geq 0$

$$V[i,0]=0 \text{ при } i \geq 0$$

**Цель:** 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	<b>37</b>

**Максимальная стоимость:**  $V[4,5]=37$ .

1. Поскольку  $V[4,5] \neq V[3,5]$ , то предмет 4 был включен 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  $V[2,3] \neq V[1,3]$ , то предмет 2 также является частью оптимального подмножества.

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

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

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