

# Динамическое программирование.

## Задача о рюкзаке

# Условие задачи о рюкзаке

- Есть набор товаров ( $n$  штук), каждый из которых характеризуется весом  $q_i$  и стоимостью  $p_i$ .
  - $(6, 9.5), (4, 6), (4, 6), (3, 4)$
- Рюкзак, загрузка которого не должна превышать  $Q_{max}$ 
  - 12
- Необходимо выбрать такой набор вещей, которые максимизируют стоимость при условии выполнения требования по максимальной загрузке

# Перебор (в том числе, полный)

- Перебрать все варианты:  $O(2^n)$

- $\{\}$  -> (0, 0)
- $\{1\}$  -> (6, 9.5)
- $\{2\}$  -> (4, 6)
- $\{3\}$  -> (4, 6)
- $\{1,2,3\}$  -> (**14**, 21.5)
- ...
- $\{2,3,4\}$  -> (11, 16)

16 вариантов

- Улучшение:

- Поиск с возвратами
- Метод ветвей и границ
- (\*) – вернуться к этому позже

<i>i</i>	<i>q</i>	<i>p</i>
1	6	9.5
2	4	6
3	4	6
4	3	4
$Q_{max}$	12	

# «Жадные» эвристики

<i>i</i>	<i>q</i>	<i>p</i>
1	6	9.5
2	4	6
3	4	6
4	3	4
$Q_{max}$	12	

- «Жадный» алгоритм – принятие локально-оптимальных решений на каждом шаге...
- Эвристика (эвристический алгоритм) – алгоритм, не являющийся гарантированно точным или оптимальным (как правило, построенный в соответствии со «здравым смыслом»)
- Варианты:
  - Берем самый дорогой из тех, что можем: 1, 2 → 15.5
  - Берем с наибольшим удельным весом: 1, 2 → 15.5
- Быстро –  $O(n^2)$  или даже  $O(n \log n)$ , но не находит максимума
- NP-трудная

# Динамическое программирование

- Принцип ДП (принцип Беллмана):
  - Каково бы ни было **состояние системы** в результате какого-то числа шагов, мы должны выбирать управление на ближайшем шаге так, чтобы оно, **в совокупности с оптимальным управлением на всех последующих шагах**, приводило к максимальному выигрышу на всех оставшихся шагах, включая **данный**.

$$W_i(S_i) = \max_{u_i \in U_i(S_i)} \{w_i(S_i, u_i) + W_{i+1}(\varphi_i(S_i, u_i))\}$$

- $W_i(S_i)$  – условный оптимальный выигрыш на всех шагах от  $i$ -того и до последнего;
- $u_i(S_i)$  – управление на  $i$ -том шаге. То управление, при котором  $W_i(S_i)$  максимально – условно-оптимальное управление.

# Динамическое программирование. Алгоритм (1)

- Шаг 1: Выбрать способ описания процесса.
  - **Этапы** – предметы в некотором порядке (фиксированном)
    - $(6, 9.5), (4, 6), (4, 6), (3, 4)$
  - **Выигрыш** – это стоимость предметов в рюкзаке.
  - **Управление** – это то, что мы можем контролировать. В данном случае – положить предмет или не положить предмет. Но управляем мы в разных точках (их 4) поэтому  $u_i, i \in \{1..4\}$ .
  - **Состояние.** Неформально это можно определить как все характеристики системы, важные с точки зрения решаемой задачи. В задачах распределения ресурсов для идентификации параметров состояния может быть важным анализ того, что именно **ограничивает** управление. В данном случае, это вместимость рюкзака.
    - Сначала (в первой точке) это  $Q_{max}$ . Потом – неясно.

# Динамическое программирование. Алгоритм (2)

- Шаг 2: Записать выигрыш на  $i$ -том шаге в зависимости от состояния системы и управления.
  - $w_i(S_i, u_i) = p_i * u_i$
- Шаг 3: Записать для  $i$ -того шага функцию, выражающую изменения состояния системы под влиянием управления  $u_i$ :
  - $\varphi_i(S_i, u_i) = S_i - q_i * u_i$
- Шаг 4: Записать **основное функциональное уравнение**, включающее функцию  $W_i(S_i)$  через  $W_{i+1}(S_i)$ :

$$W_i(S_i) = \max_{u_i \in \{u | u \in \{0,1\}, u q_i \leq S_i\}} \{p_i u_i + W_{i+1}(S_i - q_i u_i)\}$$

# Динамическое программирование. Алгоритм (3)

- Шаг 5: Найти функцию условного оптимального выигрыша для последнего этапа:

$$W_i(S_i) = \max_{u_i \in \{u | u \in \{0,1\}, uq_i \leq S_i\}} \{p_i u_i\}$$

- Шаг 6: Вычислить  $W_i(S_i)$ , зная  $W_{i+1}(S_i)$ .
- Шаг 7: Зная  $W_1(Q_{max})$  и оптимальное управление  $u_1^* = u_1^*(Q_{max})$ , определить  $u_2^* = Q_{max} - u_1^* q_1$  и далее остальные  $u_i^*$



# Динамическое программирование.

## Пример вычислений. Этап 4

Этап (предмет) 4

Вес: 3

Стоимость: 4

$S_4$	$W_4(S_4)$	$U_4(S_4)$
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		

$i$	$q$	$p$
1	6	9.5
2	4	6
3	4	6
4	3	4
$Q_{max}$	12	

$$W_4(S_4) = \max_{u_i \in \{u|u \in \{0,1\}, uq_i \leq S_i\}} \{p_i u_i\}$$

# Динамическое программирование.

## Пример вычислений. Этап 3

$$W_3(S_3) = \max\{p_3 u_3 + W_4(S_3 - q_3 u_3)\}$$

**Этап (предмет) 3**

Вес: 4

Стоимость: 6

$S_3$	$W_3(S_4)$	$U_3(S_3)$
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		

**Этап (предмет) 4**

Вес: 3

Стоимость: 4

$S_4$	$W_4(S_4)$	$U_4(S_4)$
0	0	0
1	0	0
2	0	0
3	4	1
4	4	1
5	4	1
6	4	1
7	4	1
8	4	1
9	4	1
10	4	1
11	4	1
12	4	1

$i$	$q$	$p$
1	6	9.5
2	4	6
3	4	6
4	3	4
$Q_{max}$	12	

# Динамическое программирование.

## Пример вычислений. Этап 1

### Этап (предмет) 1

Вес: 6

Стоимость: 9.5

$S_1$	$W_1(S_1)$	$U_1(S_1)$
12		

$U_1=0$ :

$$0 + W_2(12) = 16$$

$U_1=1$ :

$$9.5 + W_2(6) = 15.5$$

### Этап (предмет) 2

Вес: 4

Стоимость: 6

$S_2$	$W_2(S_2)$	$U_2(S_2)$
0	0	0
1	0	0
2	0	0
3	4	0
4	6	1
5	6	1
6	6	1
7	10	1
8	10	1
9	10	1
10	10	1
11	16	1
12	16	1

### Этап (предмет) 3

Вес: 4

Стоимость: 6

$S_3$	$W_3(S_3)$	$U_3(S_3)$
0	0	0
1	0	0
2	0	0
3	4	0
4	6	1
5	6	1
6	6	1
7	10	1
8	10	1
9	10	1
10	10	1
11	10	1
12	10	1

### Этап (предмет) 4

Вес: 3

Стоимость: 4

$S_4$	$W_4(S_4)$	$U_4(S_4)$
0	0	0
1	0	0
2	0	0
3	4	1
4	4	1
5	4	1
6	4	1
7	4	1
8	4	1
9	4	1
10	4	1
11	4	1
12	4	1

$i$	$q$	$p$
1	6	9.5
2	4	6
3	4	6
4	3	4
$Q_{max}$	12	

# Сложность точных алгоритмов для задачи о рюкзаке

- Перебор:  $O(2^n)$
- Псевдополиномиальный алгоритм (динамическое программирование):  $O(n * Q_{max})$

(Собирались вернуться позже)

$$\sum_{i=1}^4 u_i p_i \rightarrow \max$$

$$\sum_{i=1}^4 u_i q_i \leq Q_{max}$$

$$u_i \in \{0,1\}$$

# Ha GMPL

```
param p{i in 1..4} >= 0;  
param q{i in 1..4} >= 0;  
param Qmax >= 0;
```

```
var u{i in 1..4} binary;
```

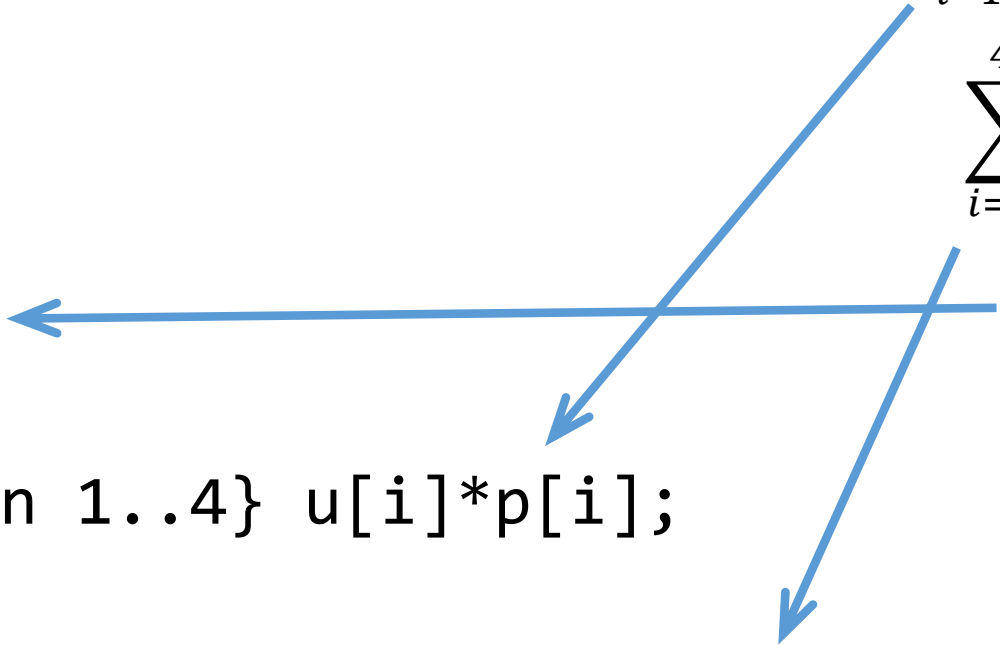
```
maximize profit: sum {i in 1..4} u[i]*p[i];
```

```
s.t. weight: sum {i in 1..4} u[i]*q[i] <= Qmax;
```

```
solve;
```

$$\sum_{i=1}^4 u_i p_i \rightarrow \max$$

$$\sum_{i=1}^4 u_i q_i \leq Q_{\max}$$

$$u_i \in \{0,1\}$$


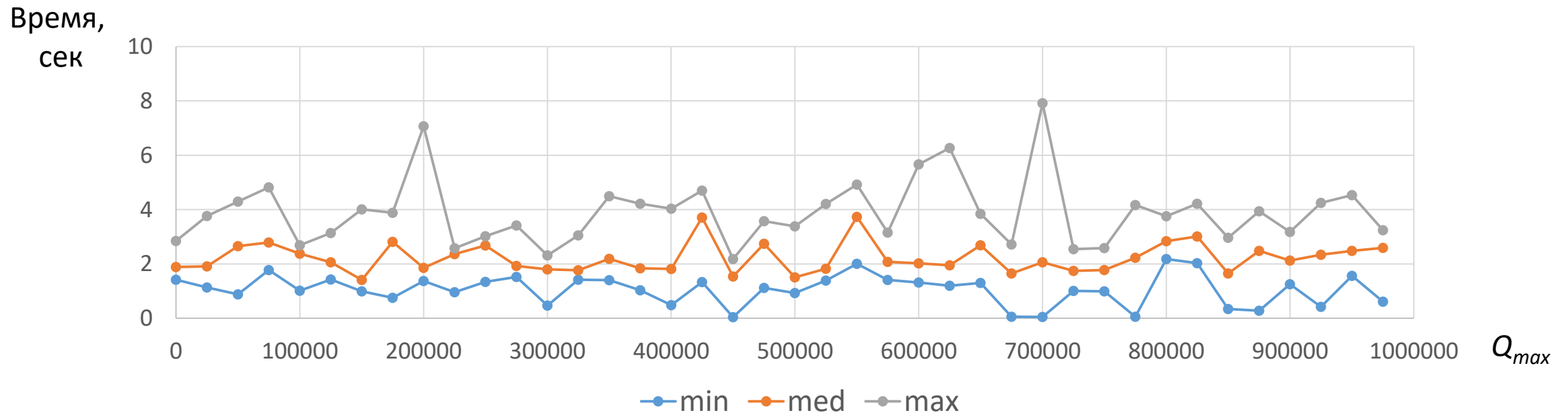
# Время решения

- Тестовые задачи:
  - Количество объектов: 2000
  - Вес и стоимость: равномерно случайно распределены на [50; 1000]
  - Размер рюкзака изменяется от 10 до 1000000
- Время решения:

???

# Время решения

- Тестовые задачи:
  - Количество объектов: 2000
  - Вес и стоимость: равномерно случайно распределены на [50; 1000]
  - Размер рюкзака изменяется от 10 до 1000000
- Время решения:





# Промежуточное резюме

- Динамическое программирование – метод, позволяющий конструировать псевдополиномиальные алгоритмы для сложных задач комбинаторной оптимизации
- Несмотря на экспоненциальную сложность *некоторых* задач (целочисленного линейного программирования в общем), конкретные экземпляры задач комбинаторной оптимизации могут быть достаточно «простыми» для решателей
  - Поэтому всегда имеет смысл моделировать задачу и пытаться решить ее стандартным решателем
  - Только если это не приводит к желаемому результату, начинать разработку специфической процедуры решения – точной или приближенной