

# Алчни алгоритми

ИТ Кариера



Учителски екип

Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning>

# Съдържание

- Алчни (greedy) алгоритми и приложение
- Упражнения: алчни алгоритми



# Алчни алгоритми (Greedy)

Greedy подходът включва изграждане на решение чрез **избиране на последователни стъпки**, всяка от които произвежда частично решение на задачата, до получаване на цялостното решение. В същото време на всяка стъпка изборът трябва да бъде:

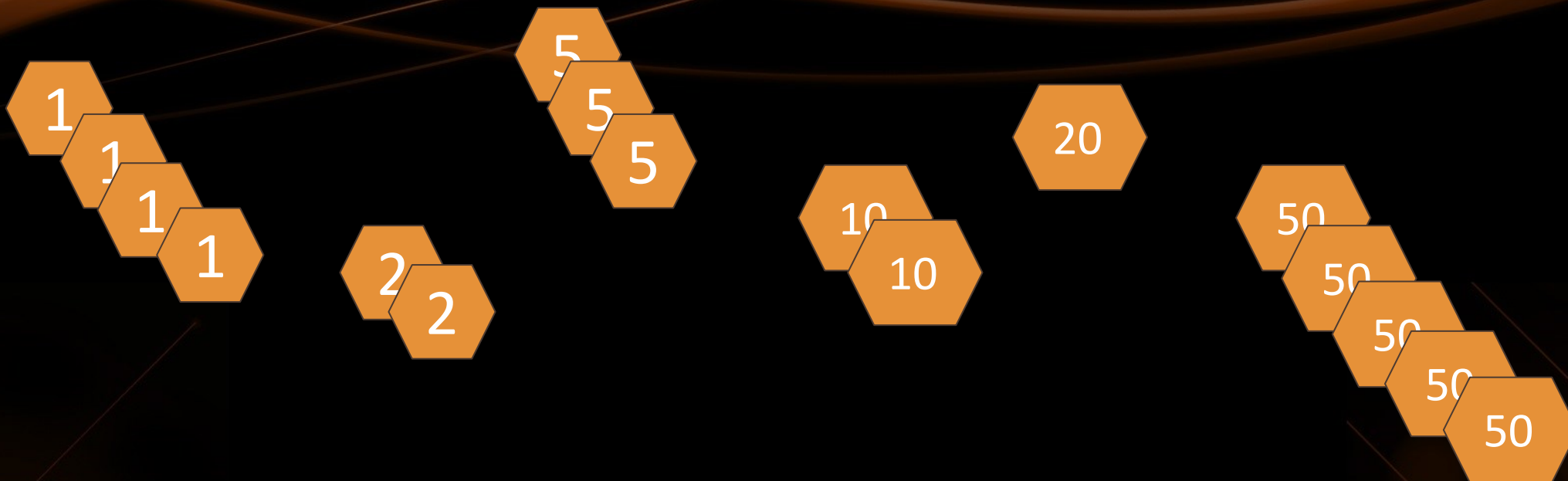
- **допустим**, т.е. да отговаря на ограниченията на задачата;
- **локално оптимален**, т.е. да е най-добрият локален избор между всички възможни варианти, налични на всяка стъпка;
- **окончателен**, т.е. веднъж направен, не може да променя следващите стъпки на алгоритъма.

# Оптимизационни решения

В компютърните науки задачите, свързани с **оптимизацията** са такива, в които е необходимо да се намери **най-доброто решение** от всички възможни решения. Пример за такива задачи са:

- Представяне на сума от неограничен брой налични монети
- Задача за египетските дроби
- Задача за раницата
- Задача за възлагане на дейности
- Задача за магнитната лента и други



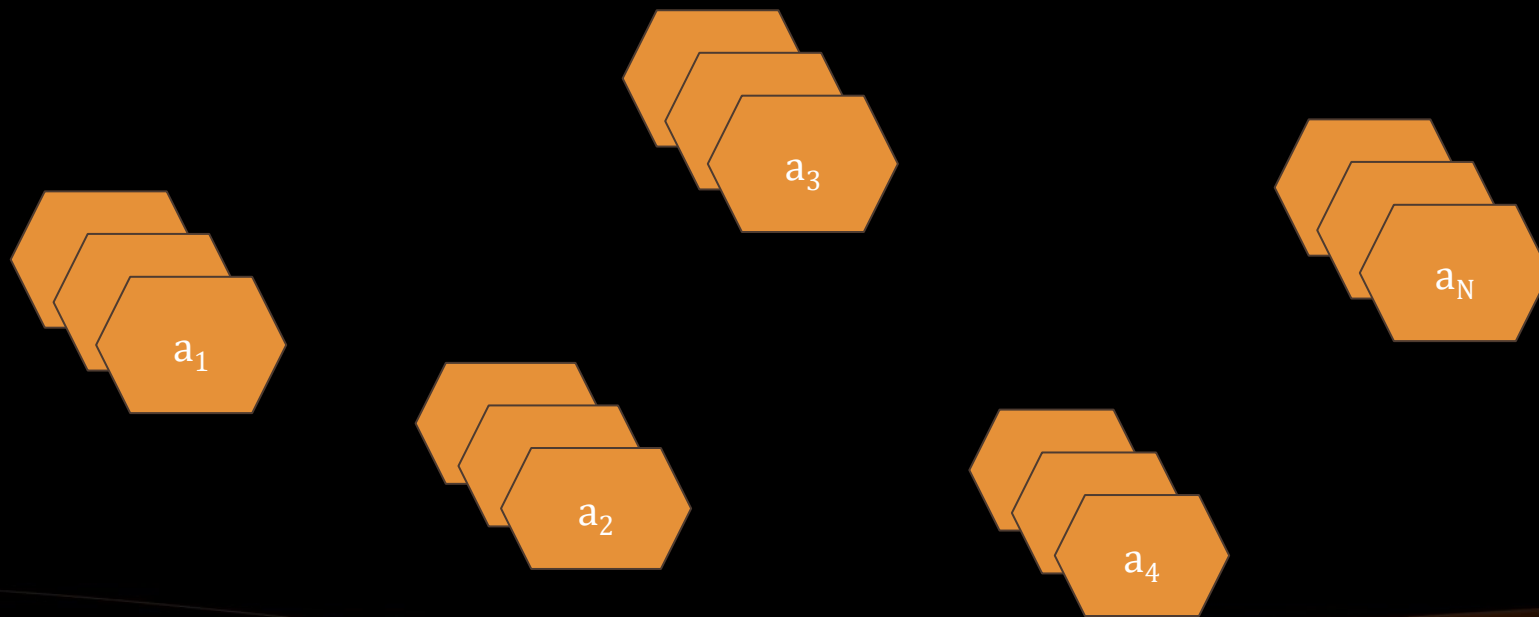


# Представяне на суми

## Задача

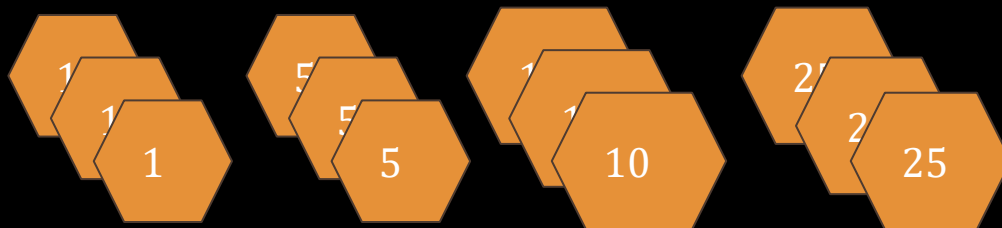
# Представяне на суми

Да се намери начин за получаване на дадена сума  $S$  ( $S$  е естествено число), като се използват минимален брой монети, с номинали от множеството  $C = \{a_1, a_2, \dots, a_N\}$ .



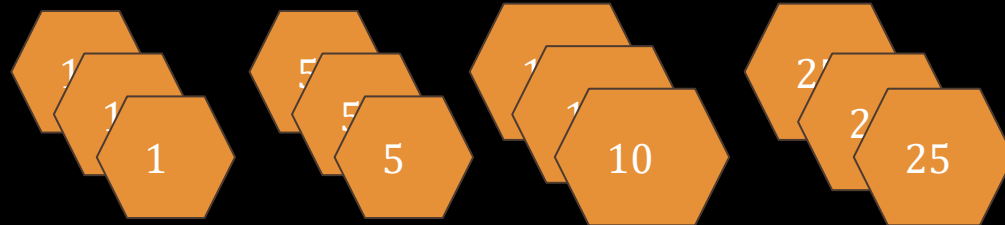
# Представяне на суми

Сума: 48



# Представяне на суми

Сума: 48



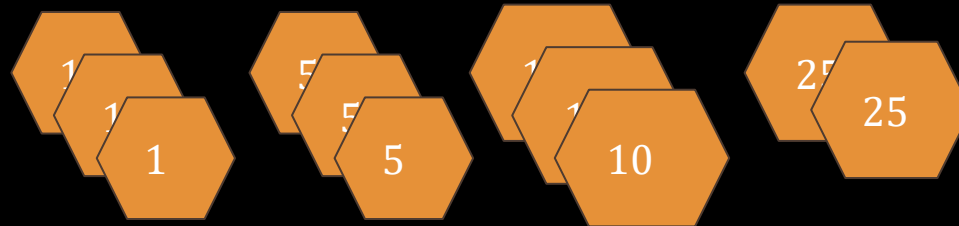
Начална стойност: 0



# Представяне на суми

Взимайте от най-голямата монета, докато е възможно.

Сума: 48



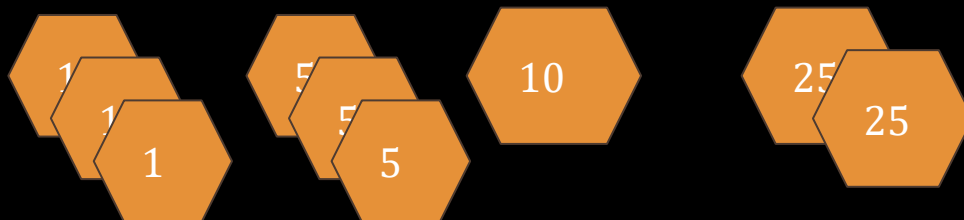
Актуализация: 25

25

# Представяне на суми

Вземете необходимия брой от втората по големина.

Сума: 48



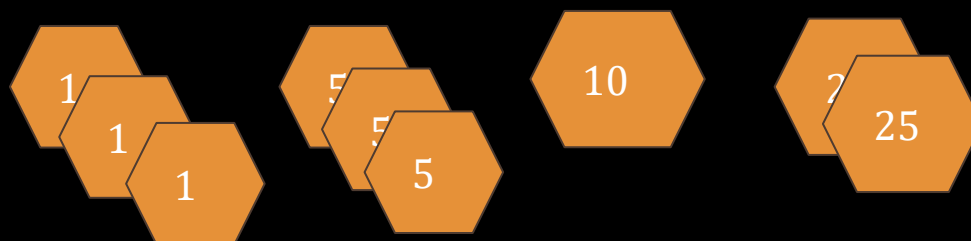
Актуализация: 45



# Представяне на суми

Вземете необходимия брой от третата по големина.

Сума: 48



ОК!

Актуализация: 48



# Представяне на суми – решение

```
int finalSum = 18;  
int currentSum = 0;  
int[] coins = { 10, 10, 5, 5, 2, 2, 1, 1 };
```

```
Queue<int> resultCoins = new Queue<int>();
```

```
// Следващия слайд
```

```
Console.WriteLine("Sum not found");
```

# Представяне на суми – решение

```
for (int i = 0; i < coins.Length; i++)
{
    if (currentSum + coins[i] > finalSum) continue;

    currentSum += coins[i];
    resultCoins.Enqueue(coins[i]);
    if (currentSum == finalSum)
    {
        // Sum Found
    }
}
```



$$7/9 = 1/2 + 1/4 + 1/36$$

$$7/9 = 1/3 + 1/3 + 1/9$$

$$7/9 = 1/9 + 1/9 + 1/9 + 1/9 + 1/9 + 1/9 + 1/9$$

# Египетски дроби

## Задача

# Египетски дроби

Древните египтяни са използвали означение само за дробите с числител единица. Всяка друга дроб  $p/q$  представяли и записвали като сума от такива дроби (с числител единица). Нека  $p$  и  $q$  са две естествени числа ( $q \neq 0$ ,  $p < q$ ;  $p, q \in \mathbb{N}$ ). Да се намери представяне на дробта  $p/q$  във вид на сума:

$$p/q = 1/a_1 + 1/a_2 + \dots + 1/a_N,$$

при което знаменателите да бъдат различни ( $a_i \neq a_j$ ,  $1 \leq i, j \leq N$ ,  $i \neq j$ ,  $a_i \geq 2$ ,  $a_j \geq 2$ ,  $a_i, a_j \in \mathbb{N}$ ).

# Египетски дроби

Търсим най-голямата възможна дроб, която не надвишава  $7/9$ .

Дроб:  $7/9$

$$1/a_1 = 1/2$$

# Египетски дроби

Търсим следващия член в сумата -  $1/a_2$ , който трябва да бъде максималната дроб, която може да се добави към  $1/2$  така, че резултатът да не надвишава  $7/9$ .

Дроб:  $7/9$

$$1/a_1 = 1/2$$

$$1/4$$

$$1/a_2 \leq 7/9 - 1/2 \leq 5/18$$

# Египетски дроби

Търсим следващия член в сумата -  $1/a_3$ , който трябва да бъде максималната дроб, която може да се добави към  $\frac{1}{2} + \frac{1}{4}$  така, че резултатът да не надвишава  $\frac{7}{9}$ .

Дроб:  $\frac{7}{9}$

$$1/a_1 = 1/2$$

$$1/a_2 = 1/4$$

$$1/36$$

$$1/a_3 \leq \frac{7}{9} - \frac{1}{2} - \frac{1}{4} \leq \frac{5}{18} - \frac{1}{4} \leq \frac{2}{72}$$



# Египетски дроби

Дроб:  $7/9$

$$1/a_1 = 1/2$$

$$1/a_2 = 1/4$$

$$1/a_2 = 1/36$$

$$1/2 + 1/4 + 1/36 = 7/9$$

OK!

# Египетски дроби – алгоритъм

Стъпка 1. Дайте стойности за числителя и знаменателя на дробта  $p/q$ .

$p=7; q=9;$

Стъпка 2. Докато числителят е по-голям от 1 търсим максималната дроб  $1/r$ , ненадвишаваща  $p/q$  ( $q \neq 0$ ).

```
r=(p+q)/p;  
//r=(7+9)/7  
//r=2
```

Изход:  
 $1/2 +$

# Египетски дроби – алгоритъм

Стъпка 3. Разликата  $p/q - 1/r$  се пресмята чрез привеждане под общ знаменател. Така, новите стойности за  $p$  и  $q$  ще бъдат:

```
p=p*r-q;  
//p=7*2-9 = 5  
q=q*r;  
//q=9*2=18
```

Стъпка 4. Проверяваме дали новите стойности на числителя и знаменателя са кратни.

```
is divided(p, q)  
//is divided(5, 18)
```

Изход:  
 $1/2 +$

# Египетски дроби – алгоритъм

Стъпка 5. Числителя е по-голям от 1. Изпълняваме стъпка 2 с новите стойности за  $p$  и  $q$ .

```
r=(p+q)/p  
//r=(5+18)/5  
//r=4
```

Изход:  
 $1/2 + 1/4 +$

# Египетски дроби – алгоритъм

Стъпка 5. Изпълняваме стъпка 3 с новите стойности за  $p$  и  $q$ .

```
p=p*r-q  
//p=5*4-18 = 2  
q=q*r;  
//q=18*4=72
```

Стъпка 6. Изпълняваме стъпка 4 за новите стойности на  $p$  и  $q$ .

```
is divided(p, q)  
//is divided(2, 72) ->1, 36
```

Изход:  
 $1/2 + 1/4 +$

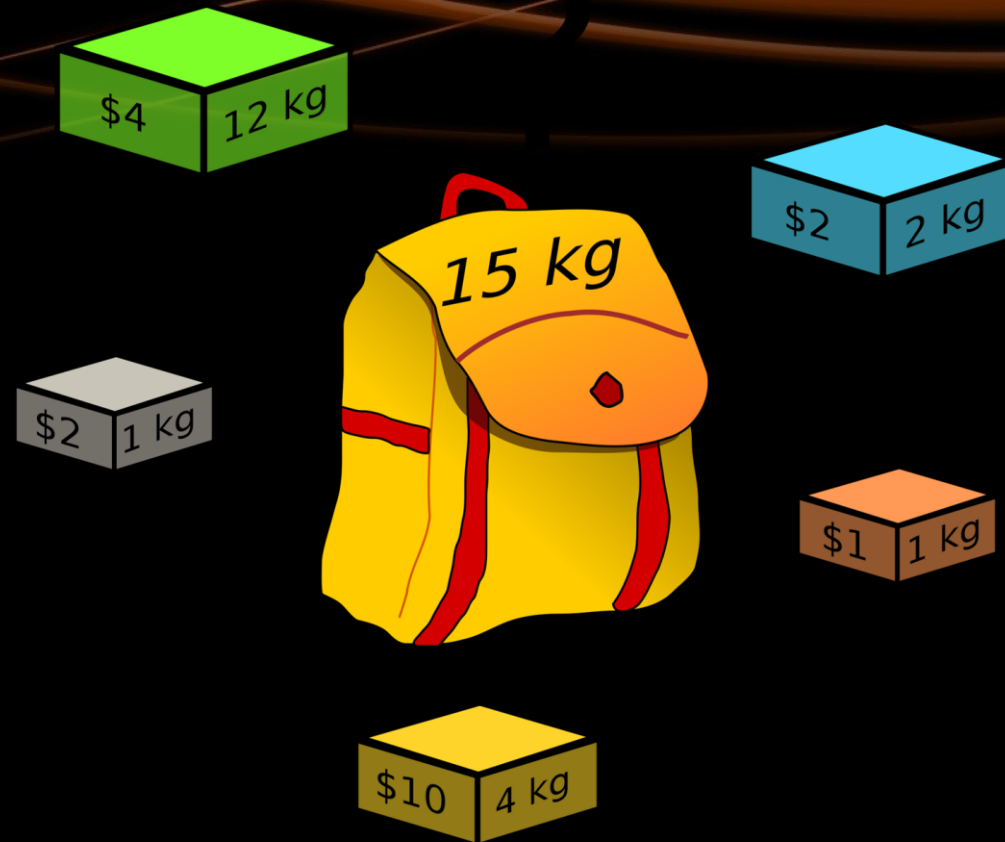


# Египетски дроби – алгоритъм

Стъпка 7. Числителят след съкращението на дробта е 1. Добавяме съкратената дроб към получения до тук израз.

Изход:  
 $1/2 + 1/4 + 1/36$

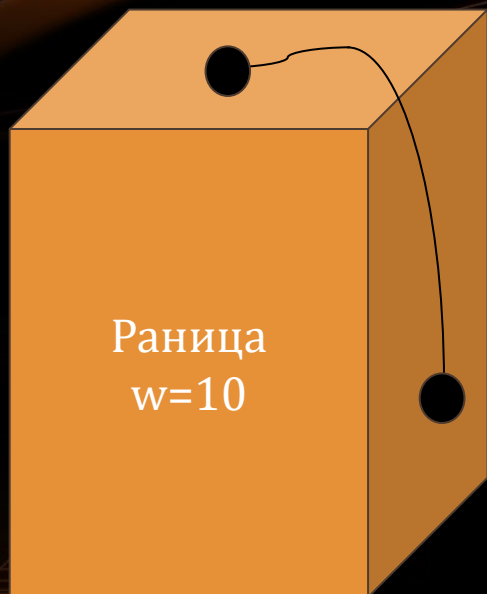
Стъпка 8. Край на алгоритъма.



# Задача за раницата

# Задача за раницата

Дадени са  $N$  предмета с тегла  $w_1, w_2, \dots, w_N$  и съответните им цени  $v_1, v_2, \dots, v_N$ , както и раница, която може да издържи тегло  $W$ . Необходимо е да се намери подмножество от предмети, които могат да бъдат поставени в раницата и които в същото време да имат максимална цена.



Раница  
 $w=10$

$w_1=7$   
 $v_1=42$

Предмет 1

$w_2=3$   
 $v_2=12$

Предмет 2

$w_3=4$   
 $v_3=40$

Предмет 3

$w_4=5$   
 $v_4=25$

Предмет 4

Ще подредим резултатите в таблица, разглеждайки всички възможни подмножества.



# Задача за раницата – алгоритъм

- Разглеждат се всички подмножества от 4 елемента, като е необходимо:
  - тяхното общо тегло да е по-малко или равно на теглото на раницата
  - тяхната обща цена да е максимална
- Прилага се метода на изчерпващото търсене, т.е. преглеждат се всички подмножества и се търси това, което отговаря на условието.
- Необходими са два масива за пазене на стойностите съответно на теглата  $m[i]$  и цените  $c[i]$  на всеки един от предметите.



# Задача за раницата – алгоритъм

Разглеждаме подмножеството, състоящо се от първия предмет. Неговото общо тегло е 7, а общата му цена е 42. Към него се опитваме да добавим нов предмет, който не е взет до момента. Запазваме максималната обща цена, намерена до момента и отговаряща на тегло по-малко или равно на 10. (MAXC=42)

Подмножес тво	Общо тегло	Обща цена
{1}	7	42
{1, 2}	10	36
{1, 3}	11	недопустим
{1, 4}	12	недопустим

Подмножес тво	Общо тегло	Обща цена
{1, 2, 3}	14	недопустим
{1, 2, 4}	15	недопустим
{1, 3, 4}	16	недопустим
{1, 2, 3, 4}	19	недопустим

# Задача за раницата – алгоритъм

Разглеждаме подмножеството, състоящо се от втория предмет. Неговото общо тегло е 3, а общата му цена е 12. Към него се опитваме да добавим нов предмет, който не е взет до момента. Запазваме максималната обща цена, намерена до момента и отговаряща на тегло по-малко или равно на 10. ( $42 < 52 \rightarrow \text{MAXC} = 52$ )

Подмножество	Общо тегло	Обща цена
{2}	3	12
{2, 3}	7	52
{2, 4}	8	37
{2, 3, 4}	12	недопустим

# Задача за раницата – алгоритъм

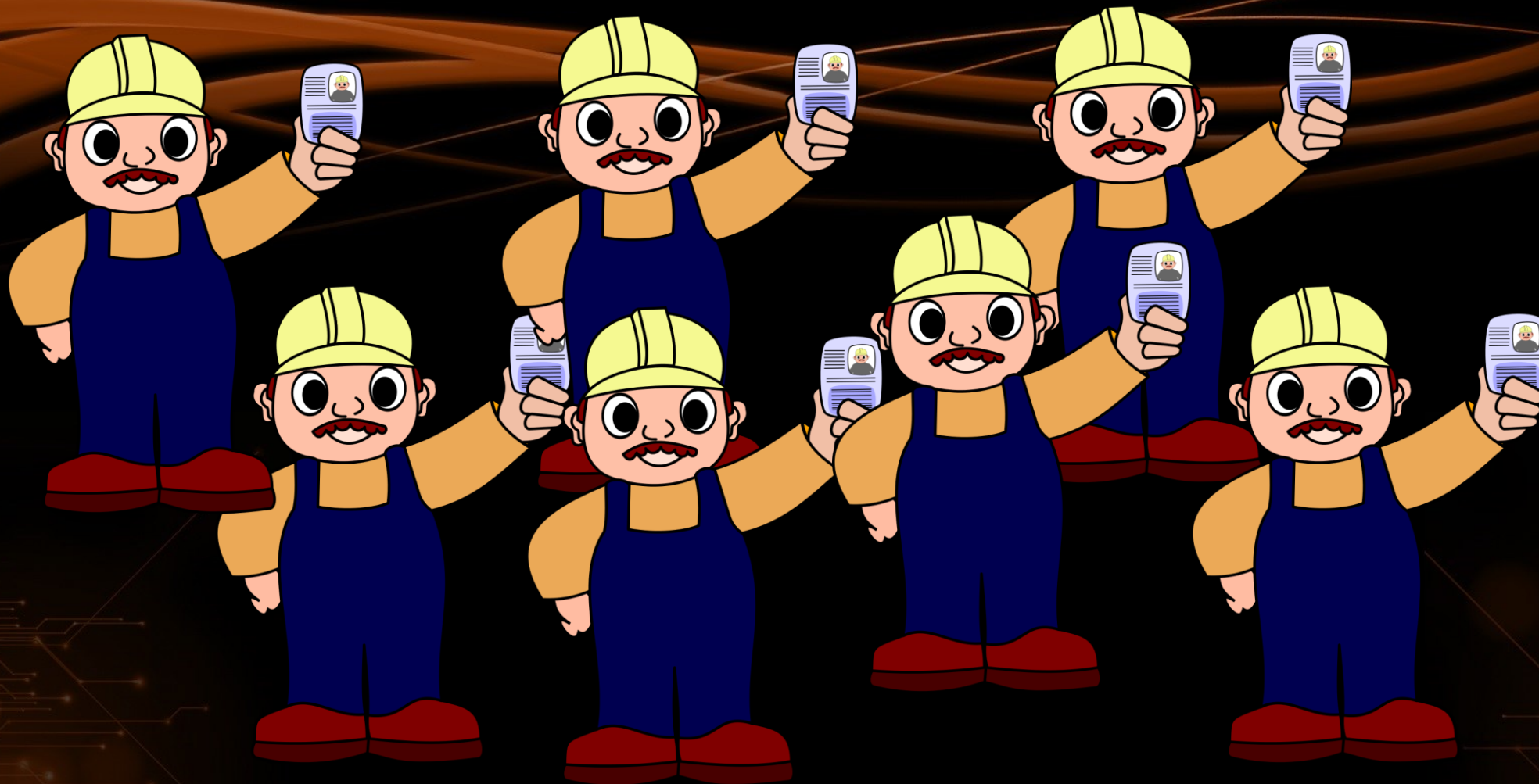
Разглеждаме подмножеството, състоящо се от третия предмет. Неговото общо тегло е 4, а общата му цена е 40. Към него се опитваме да добавим нов предмет, който не е взет до момента и отговаряща на тегло по-малко или равно на 10. Запазваме максималната обща цена, намерена до момента. ( $52 < 65 \rightarrow \text{MAXC} = 65$ )

Подмножество	Общо тегло	Обща цена
{3}	4	40
{3, 4}	9	65

# Задача за раницата

Разглеждаме подмножеството, състоящо се от четвъртия предмет. Неговото общо тегло е 5, а общата му цена е 25. Към него се опитваме да добавим нов предмет, който не е взет до момента и отговаряща на тегло по-малко или равно на 10. Запазваме максималната обща цена, намерена до момента. (MAXC=65)

Подмножество	Общо тегло	Обща цена
{4}	5	25



**Задача за възлагане на дейности**



# Задача за възлагане на дейности

Нека имаме  $N$  служители, които трябва да изпълнят  $N$  дейности, по една дейност всеки (т.е. всеки служител е назначен да изпълнява само една дейност, а всяка дейност е възложена само на един човек).

Разходите за изпълнение на  $j$ -тата дейност от  $i$ -тия служител са известни и са равни на  $C[i, j]$  за всички двойки  $i, j = 1, \dots, N$ .

Задачата е следната: необходимо е да се разпределят дейностите между работниците, така че те да бъдат изпълнени с най-ниска обща цена.

# Задача за възлагане на дейности

Задачата може да се представи чрез матрицата на разходите. Идеята е да се избере по един елемент от всеки ред на матрицата, така че избраните елементи да са в различни колони и общото им количество да има най-малката възможна стойност.

	Дейност 1	Дейност 2	Дейност 3	Дейност 4
Работник 1	9	2	7	8
Работник 2	6	4	3	7
Работник 3	5	8	1	8
Работник 4	7	6	9	4

# Задача за възлагане на дейности

Очевидната стратегия за решаване на тази задача е да изберете най-малките елементи във всеки ред, но тя реално не е вярна, тъй като елементите се появяват в една и съща колона. Всъщност най-малките елементи въобще може да не влизат в оптималното решение.

	Дейност 1	Дейност 2	Дейност 3	Дейност 4
Работник 1	9	2	7	8
Работник 2	6	4	3	7
Работник 3	5	8	1	8
Работник 4	7	6	9	4

NO!

# Задача за възлагане на дейности

{1, 2, 3, 4}

Случай в който на 1-ия работник е възложена първата дейност, на 2-ия - втората, на 3-ия - третата и на 4-ия - четвъртата.

	Дейност 1	Дейност 2	Дейност 3	Дейност 4
Работник 1	9	2	7	8
Работник 2	6	4	3	7
Работник 3	5	8	1	8
Работник 4	7	6	9	4

18

# Задача за възлагане на дейности

{1, 2, 4, 3}

Случай в който на 1-ия работник е възложена първата дейност, на 2-ия - втората, на 3-ия - четвъртата и на 4-ия - третата.

	Дейност 1	Дейност 2	Дейност 3	Дейност 4
Работник 1	9	2	7	8
Работник 2	6	4	3	7
Работник 3	5	8	1	8
Работник 4	7	6	9	4

30

# Задача за възлагане на дейности

{1, 3, 2, 4}

Случай в който на 1-ия работник е възложена първата дейност, на 2-ия - третата, на 3-ия - втората и на 4-ия - четвъртата.

	Дейност 1	Дейност 2	Дейност 3	Дейност 4
Работник 1	9	2	7	8
Работник 2	6	4	3	7
Работник 3	5	8	1	8
Работник 4	7	6	9	4

24



# Задача за възлагане на дейности

{1, 3, 4, 2}

Случай в който на 1-ия работник е възложена първата дейност, на 2-ия - третата, на 3-ия - четвъртата и на 4-ия - втората.

	Дейност 1	Дейност 2	Дейност 3	Дейност 4
Работник 1	9	2	7	8
Работник 2	6	4	3	7
Работник 3	5	8	1	8
Работник 4	7	6	9	4

26

# Задача за възлагане на дейности

{1, 4, 2, 3}

Случай в който на 1-ия работник е възложена първата дейност, на 2-ия - четвъртата, на 3-ия - втората и на 4-ия - третата.

	Дейност 1	Дейност 2	Дейност 3	Дейност 4
Работник 1	9	2	7	8
Работник 2	6	4	3	7
Работник 3	5	8	1	8
Работник 4	7	6	9	4

33

# Задача за възлагане на дейности

{1, 4, 3, 2}

Случай в който на 1-ия работник е възложена първата дейност, на 2-ия - четвъртата, на 3-ия - третата и на 4-ия - втората.

	Дейност 1	Дейност 2	Дейност 3	Дейност 4
Работник 1	9	2	7	8
Работник 2	6	4	3	7
Работник 3	5	8	1	8
Работник 4	7	6	9	4

23

# Задача за възлагане на дейности

В задачата за възлагане на дейности, броят на разглежданите пермутации са равни на  $N!$ . При  $N=4$  е необходимо да разгледаме 24 случая. Ние разгледахме само 6. Оказва се, че изчерпващото търсене е непрактично за всички стойности на  $N$ , с изключение на малките. Този проблем има значително по-ефективно решение, наречен унгарски метод в чест на унгарските математици Kőnig и Egerváry, които са го открили.

# Обобщение

- Greedy алгоритмите се използват за решаване на оптимизационни задачи
- Обикновено са по-ефективни от другите алгоритми, но може да доведат и до не толкова оптимален резултат
- Алчните алгоритми избират най-доброто локално решение
- Алчните алгоритми предполагат, че винаги изборът на локално оптимално решение води до глобално такова, но понякога не е така





# Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"



Министерство  
на образованието  
и науката



Национална  
програма  
„Обучение за  
ИТ кариера“

- Курсът се разпространява под свободен лиценз **CC-BY-NC-SA**

