



Национална програма  
"Обучение за ИТ умения и кариера"  
<https://it-kariera.mon.bg>

Министерството на  
образованието и науката  
<https://www.mon.bg>

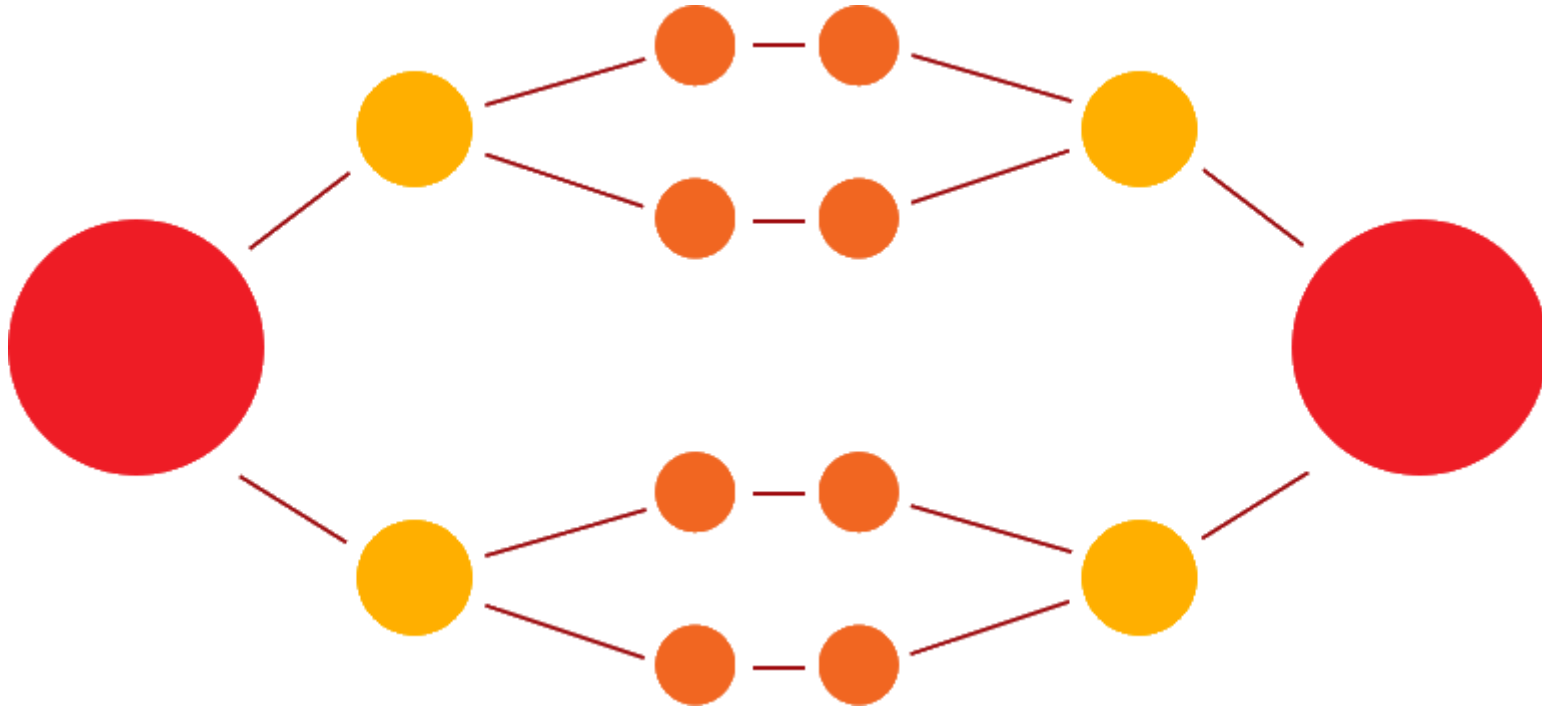


# Динамично оптимизиране

Алгоритми и структури от данни

# Съдържание

- Методът "Разделяй и владей". Динамично оптимиране
  - Упражнения: задачи върху динамично оптимиране
- Двумерно динамично оптимиране
  - Упражнения: по-сложни задачи върху динамично оптимиране



Методът "Разделяй и владей".  
Динамично оптимиране

# Методът "Разделяй и владей"

- Един от най-важните и най-широко приложим метод за проектиране на ефективни алгоритми
- Метод на декомпозицията
- Разделя дадена задача с размер  $N$  на по-малки задачи,
- На основата на решенията на по-малките задачи се получава решението на първоначалната задача.

# Динамично оптимизиране

- Метод за решаване на задачи с припокриващи се подзадачи.
- Изграждане на рекурентни връзки, свързващи решението на задачата с решенията на по-малки подзадачи от същия тип.
- Решаване на всяка една от по-малките подзадачи само веднъж, записване на резултата в таблица, от която след това се получава решение на първоначалната задача.

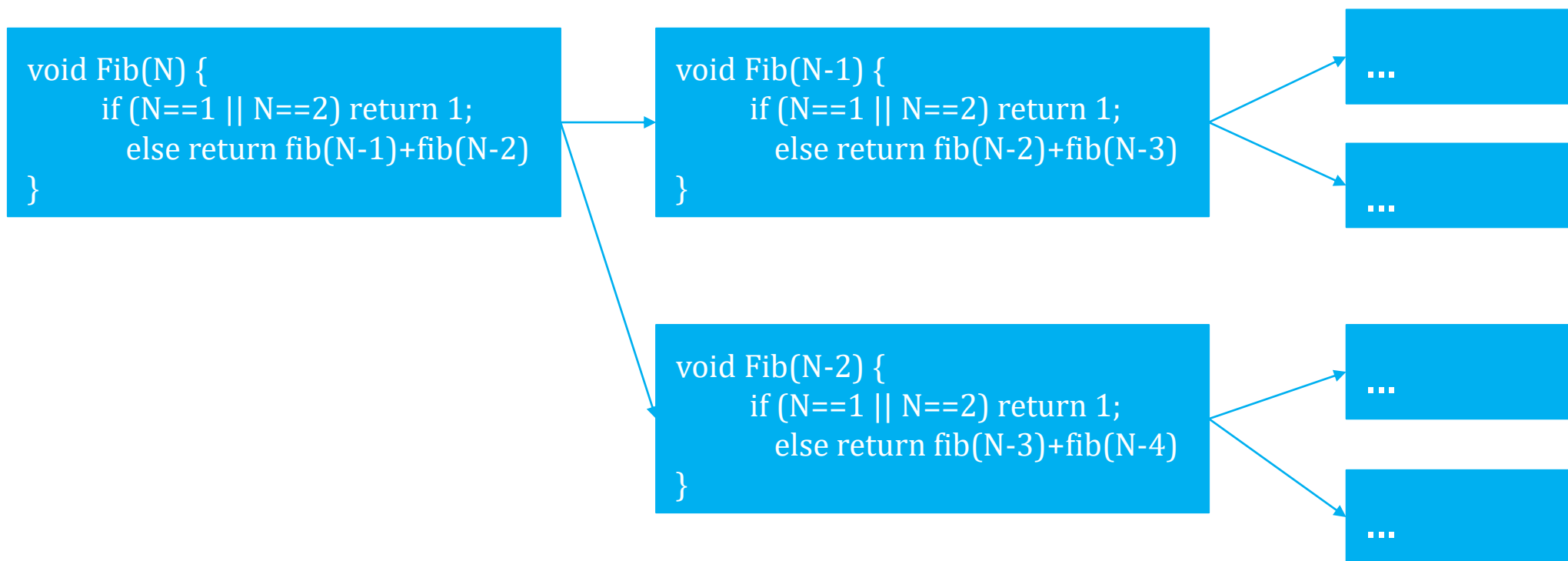
	3		2										
			1	1									
								8					

# Редица на Фибоначи

Динамично решение

# Защо динамичното решение е по-добро? [1/2]

- Рекурсивното решение за големи стойности на  $N$  е твърде нерационално
- Налага се твърде много пъти да се пресмятат някои от стойностите на редицата



## Защо динамичното решение е по-добро? [2/2]

- Динамичното решение използва вече пресметнатите стойности за по-малки стойности на търсения аргумент, без да се пресмята наново.
- Използват се рекурсивни формули, но стойностите се вземат от някаква структура от данни, в която са били попълнени в момента на пресмятането си.
- Сложността на такива задачи е линейна, с изключение на задачите, които изискват поддържането на матрица за запазване на стойности, за които сложността е квадратна.



# Редица на Фибоначи [1/7]

Да се напише програма, която извежда редицата на Фибоначи. На стандартния вход се въвежда едно цяло число  $n$  – до кой елемент да се отпечата редицата. На стандартния изход – редицата от числа, разделени с един интервал.

Примерен вход:

24

Примерен изход:

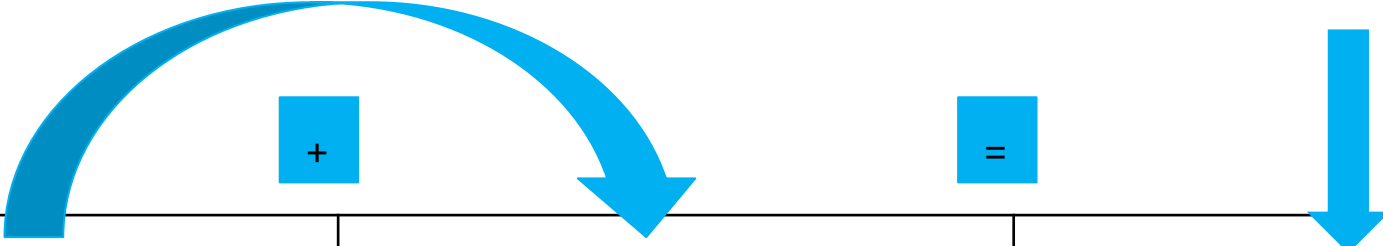
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711  
28657 46368

# Редица на Фибоначи [2/7]

- Означаваме първите два члена на редицата съответно с  $a$  и  $b$  и им даваме начална стойност 1.
- В променлива с име  $c$  ще натрупваме сумата, като в  $a$  ще помним последната стойност на  $b$ , а в  $b$  – последната стойност на  $c$ , т.е.  $c = a + b$ ;  $a = b$ ;  $b = c$ ;
- В масив с име  $arr$  се съхраняват текущите стойности на променливата  $c$ .

# Редица на Фибоначи [3/7]

Последователно попълване на текущите стойности в таблица.



a	b	c
1	1	2

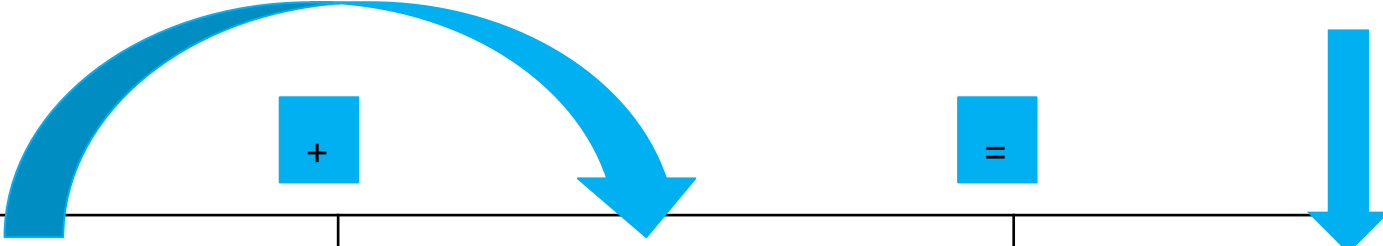
# Редица на Фибоначи [4/7]

Последователно попълване на текущите стойности в таблица.

a	b	c
1	1	2
1	2	3

# Редица на Фибоначи [5/7]

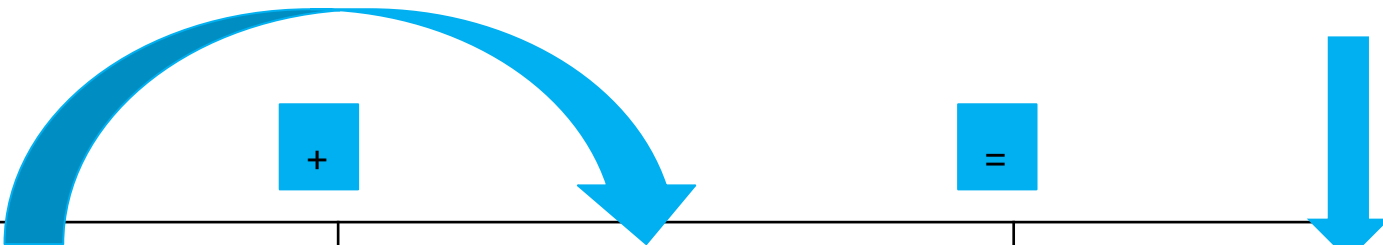
Последователно попълване на текущите стойности в таблица.



a	b	c
1	1	2
1	2	3
2	3	5

# Редица на Фибоначи [6/7]

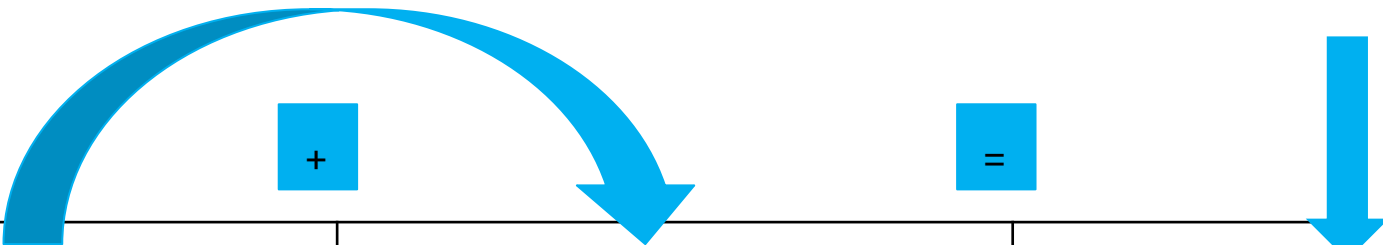
Последователно попълване на текущите стойности в таблица.



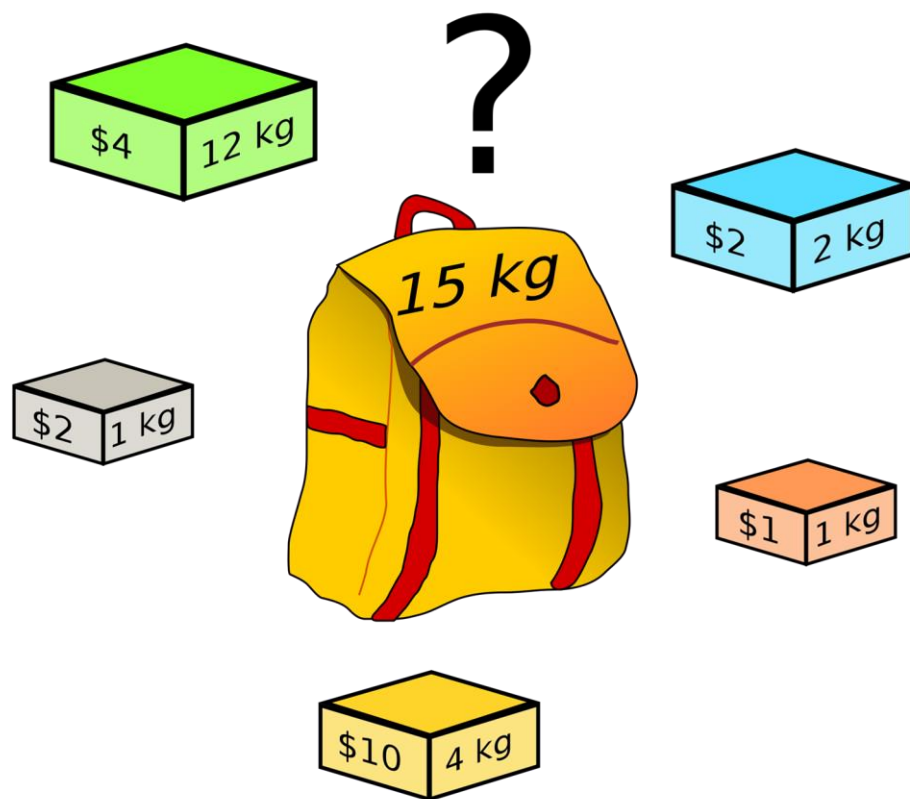
a	b	c
1	1	2
1	2	3
2	3	5
3	5	8

# Редица на Фибоначи [7/7]

Последователно попълване на текущите стойности в таблица.



a	b	c
1	1	2
1	2	3
2	3	5
3	5	8
5	8	13



# Задача за раницата

Динамично решение



# Задача за раницата [1/5]

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.

# Задача за раницата [2/5]

Дефинираме рекурентна целева функция:

$$F(0) = 0; \quad F(i) = \max \{ c_j + F(i - m_j), \quad j = 1, 2, \dots, N, \quad m_j \leq i \}, \quad i > 0$$

Методът на динамичното оптимиране изисква последователно пресмятане на стойностите на  $F(i)$ , като за това пресмятане се използват вече пресметнатите стойности за по-малки  $i$ .

# Задача за раницата [3/5]

Нека разполагаме с 8 предмета. Масивите  $m[i]$  и  $c[i]$  ще пазят съответно теглата и цените им.

```
N = 8;  
index   0 1 2 3 4 5 6 7 8  
m[MAXN] = {0, 3, 7, 6, 1, 2, 4, 5, 5};  
c[MAXN] = {0, 5, 3, 9, 1, 1, 2, 5, 2}  
M = 7;
```

# Задача за раницата [4/5]

Пресмятаме рекурентната целева функция за първите три предмета.

```
Fn[0] = 0;  
Fn[1] = max { c[4]+Fn[0] } = 1 /4/  
Fn[2] = max { c[5]+Fn[0] } = 1 /5/  
Fn[3] = max { c[1]+Fn[0], c[4]+Fn[2], c[5]+Fn[1]  
} = max{5,3,2} = 5 /1/
```

# Задача за раницата [5/5]

Пресмятаме рекурентната целева функция за останалите предмети.

$$Fn[4] = \max \{ c[1]+Fn[1], c[4]+Fn[3], c[6]+Fn[0] \} = \max\{6,6,2\} = 6$$

/1,4/

$$Fn[5] = \max \{ c[1]+Fn[2], c[5]+Fn[3], c[6]+Fn[1], c[7]+Fn[0], c[8]+Fn[0] \} = \max\{6,6,3,5,2\} = 6$$

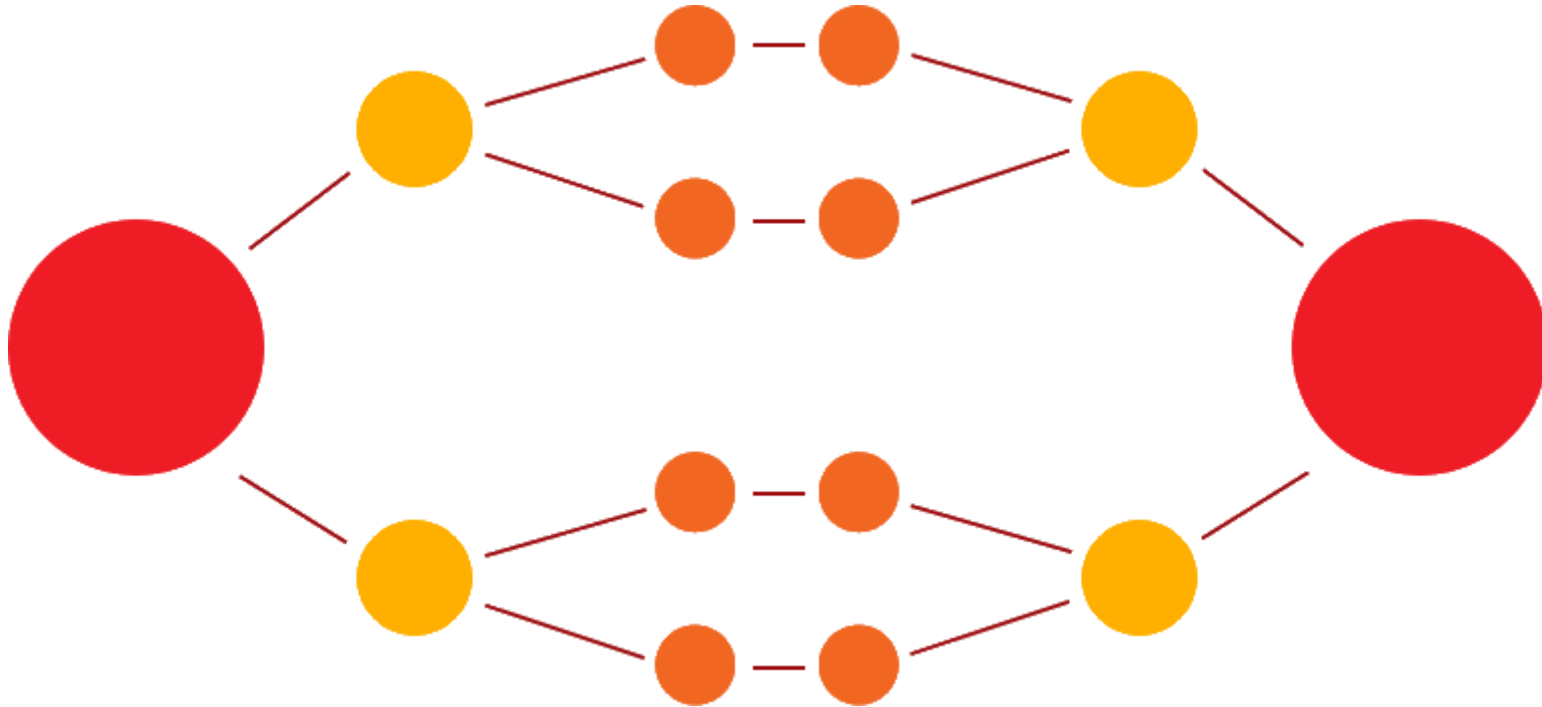
/1,5/

$$Fn[6] = \max \{ c[3]+Fn[0], c[4]+Fn[5], c[5]+Fn[4], c[6]+Fn[2], c[7]+Fn[2], c[8]+Fn[1] \} = \max\{9+0,1+5,1+6,2+1,5+1,2+1\} = 9$$

/3/

$$Fn[7] = \max \{ c[2]+Fn[0], c[3]+Fn[1], c[4]+Fn[6], c[6]+Fn[5], c[7]+Fn[2], c[8]+Fn[5] \} = \{5+0,9+1,1+9,2+6,5+1,2+6\} = 10$$

/3,4/



# Упражнения

задачи върху динамично оптимиране

# Редици от 0 и 1 [1/6]

Задача: От стандартния вход се въвежда цяло положително число  $N$ . На стандартния изход трябва да се отпечата колко на брой са редиците, с дължина  $N$ , съставени само от 0 и 1, в които няма две последователни 0.

x	x	x	...	x
---	---	---	-----	---

# Редици от 0 и 1 [2/6]

- Означаваме с  $V_k$  броя на редиците от разглеждания вид, които са с дължина  $k$ .
- Ако за последен елемент изберем 1, то предишните  $k-1$  елемента са някаква редица от разглеждания вид.

x	x	x	...	1
---	---	---	-----	---

- Броят на тези редици е  $V_{k-1}$



# Редици от 0 и 1 [3/6]

- Ако за последен елемент изберем 0, то на предпоследното място с номер  $k-1$  задължително трябва да има 1. Тогава предишните  $k-2$  елемента са някаква редица от разглеждания вид.

x	x	...	1	0
---	---	-----	---	---

- Броят на тези редици е  $V_{k-2}$

# Редици от 0 и 1 [4/6]

- В сила е следната рекурентната формула  $B_k = B_{k-1} + B_{k-2}$
- При  $k=1$   $B_k = 2$
- При  $k=2$   $B_k = 3$

```
// с цикъл  
b1=2;  
b2=3;  
for (i=3; i<=N; i++){  
    b=b2+b1;  
    b1=b2;  
    b2=b;  
}  
cout<<b;
```

```
// с рекурсия  
intB(int k){  
    if(k==1) return 2;  
    if(k==2) return 3;  
    return B(k-1)+B(k-2);  
}
```

# Редици от 0 и 1 [5/6]

- За пресмятане на големи стойности за N
- Вече веднъж пресметнатите стойности се помнят в масив v[i]

```
// с масив
intB(int k){
if (v[k]==0) v[k]=B(k-1)+B(k-2);
return v[k];
}
```

```
const Nmax=21;
int v[Nmax];
void main(){
v[1]=2;
v[2]=3;
for(int i=3; i<Nmax; i++)
v[i]=0;
cout<<B(20);
}
```

# Редици от 0 и 1 [6/6]

Дължина на редицата	Редици от разглеждания вид
1	0 1
2	01 10 11
3	110 011 111 101 010
....	....

# Задача 4=1 [1/3]

Задача: Густаво знае да брои, но сега той се учи да пише числата. Като много добър ученик той е научил 1, 2, 3 и 4. Но той не осъзнава, че 4 е различно от 1. Въпреки това той се забавлява с една игра, в която съставя числа от тези цифри и смята сбора на цифрите.

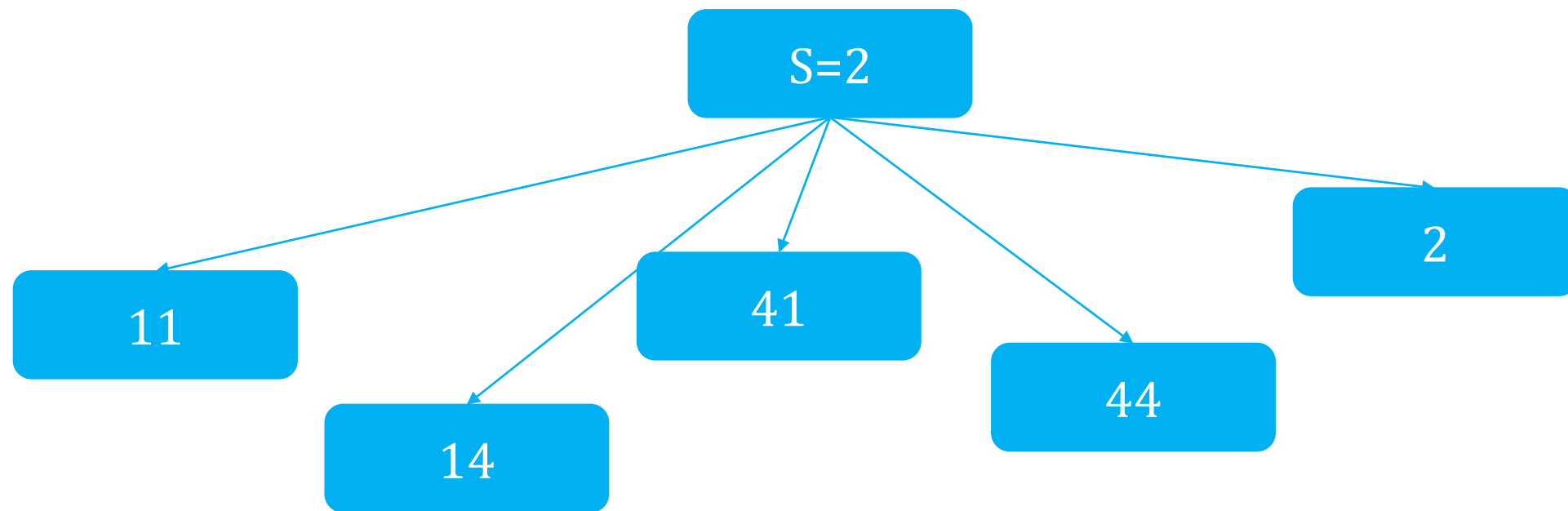
Например :

$$132 = 1 + 3 + 2 = 6$$

$$112314 = 1 + 1 + 2 + 3 + 1 + 1 = 9 \text{ (запомнете, че Густаво мисли, че } 4 = 1)$$

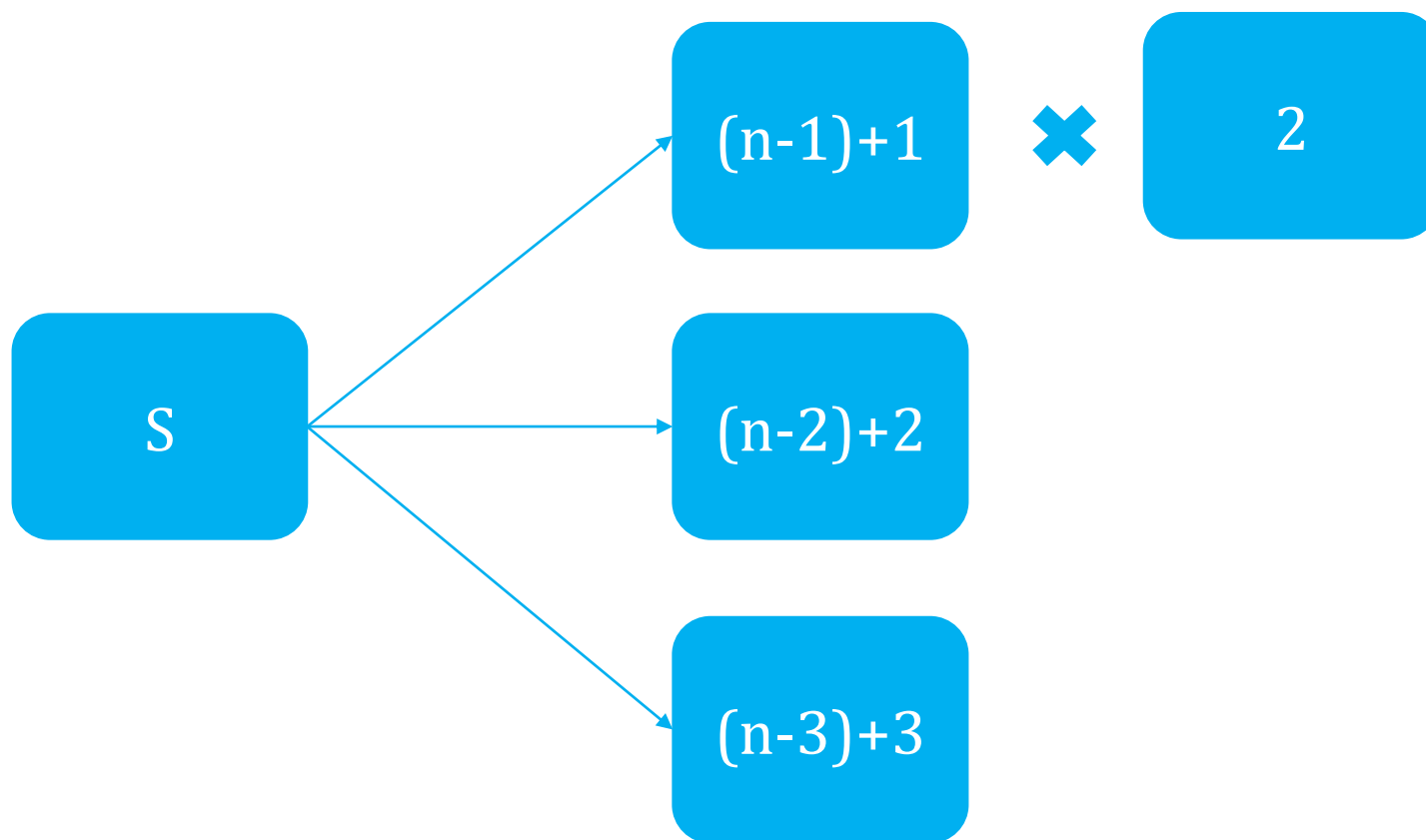
# Задача 4=1 [2/3]

Да се напише програма, която извежда броя на числата, чиято сума е равна на  $S$ . На стандартния вход се въвежда едно цяло число  $S$ .



# Задача 4=1 [3/3]

Нека в масив  $b[]$  пазим броя на числата, чиято сума е равна на  $S$ . Тогава  $b[i] = 2*b[i-1] + b[i-2] + b[i-3]$ ;

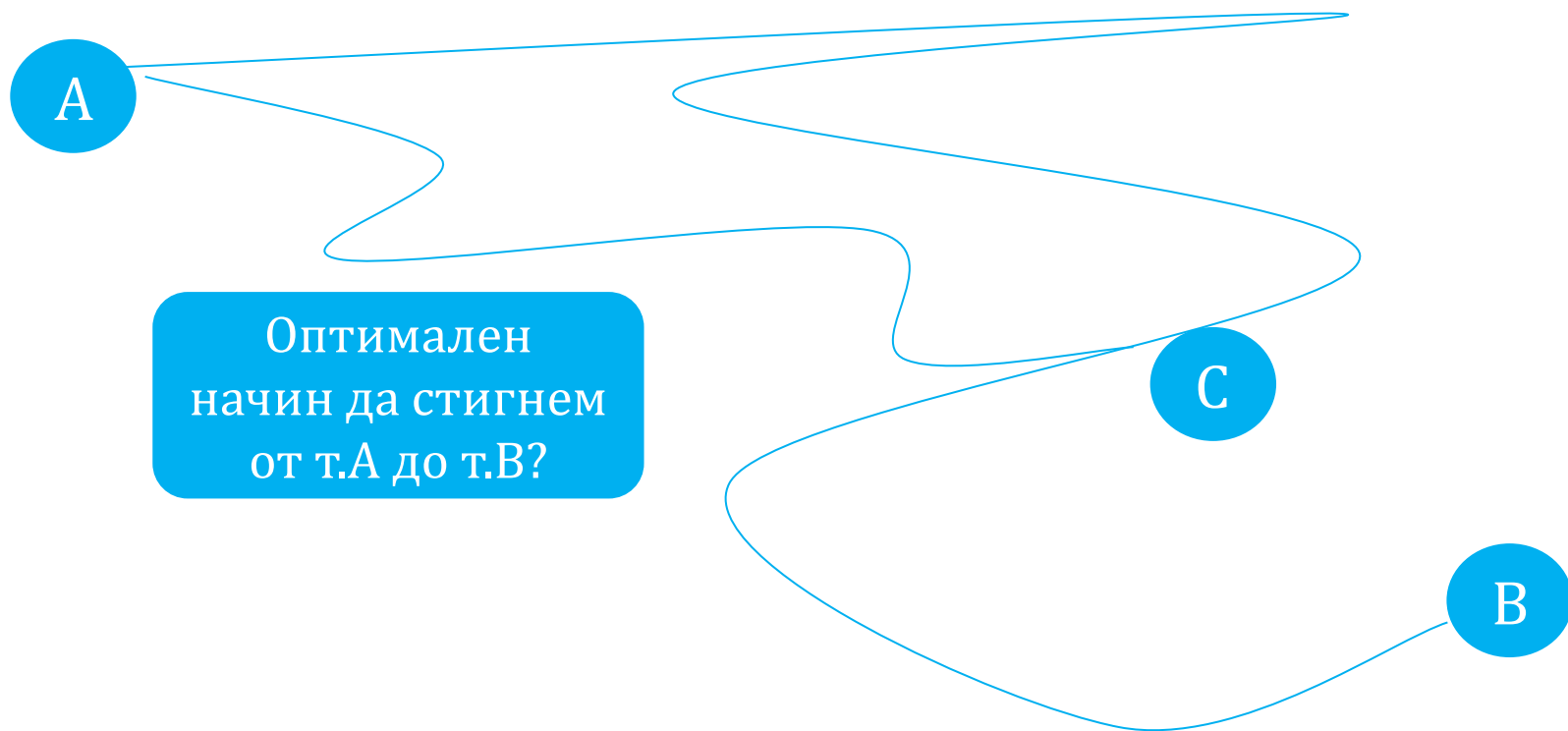






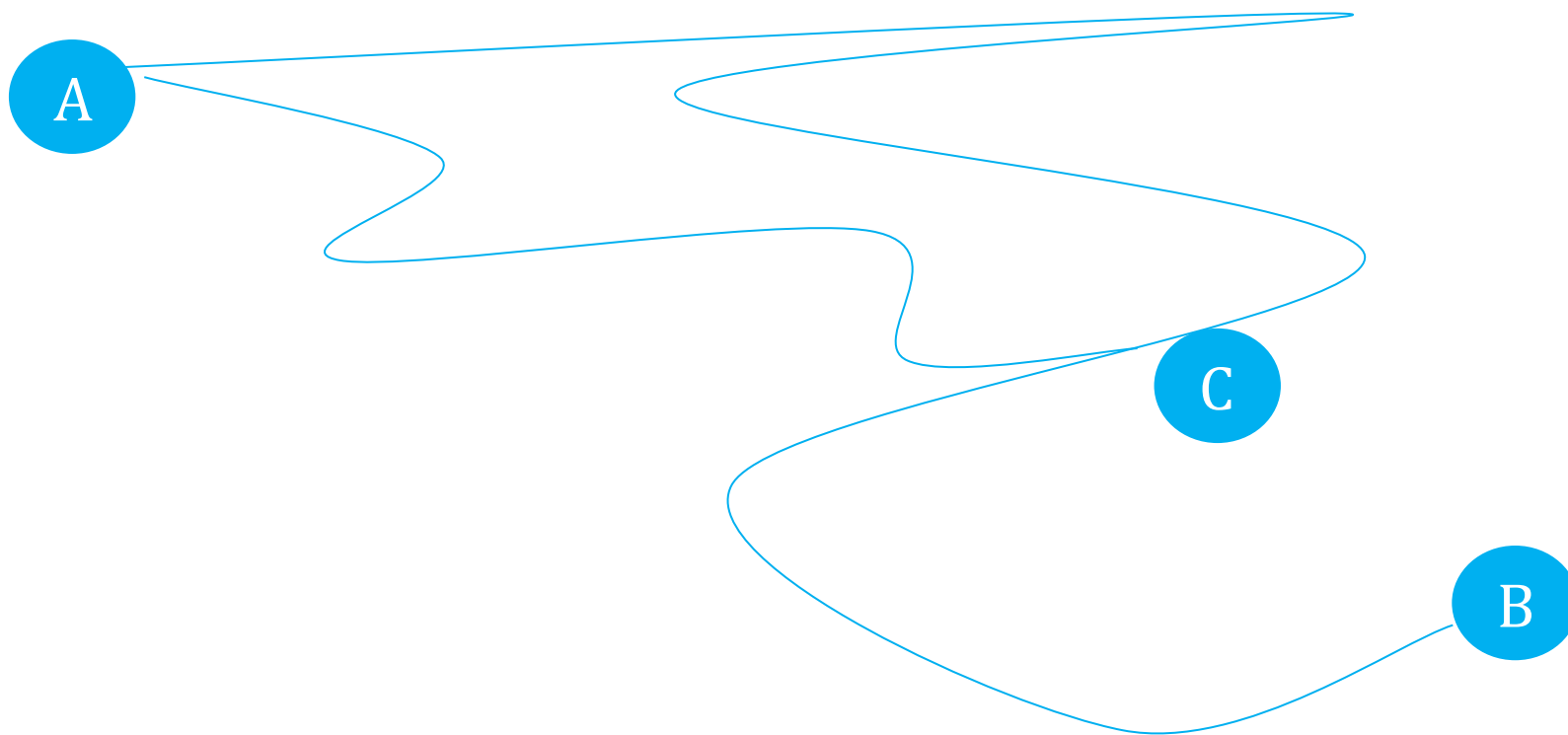
# Двумерно динамично оптимиране [1/3]

Основополагаща идея, на която е базиран методът на динамичното оптимиране - произволна част от оптимална траектория също е оптимална траектория.

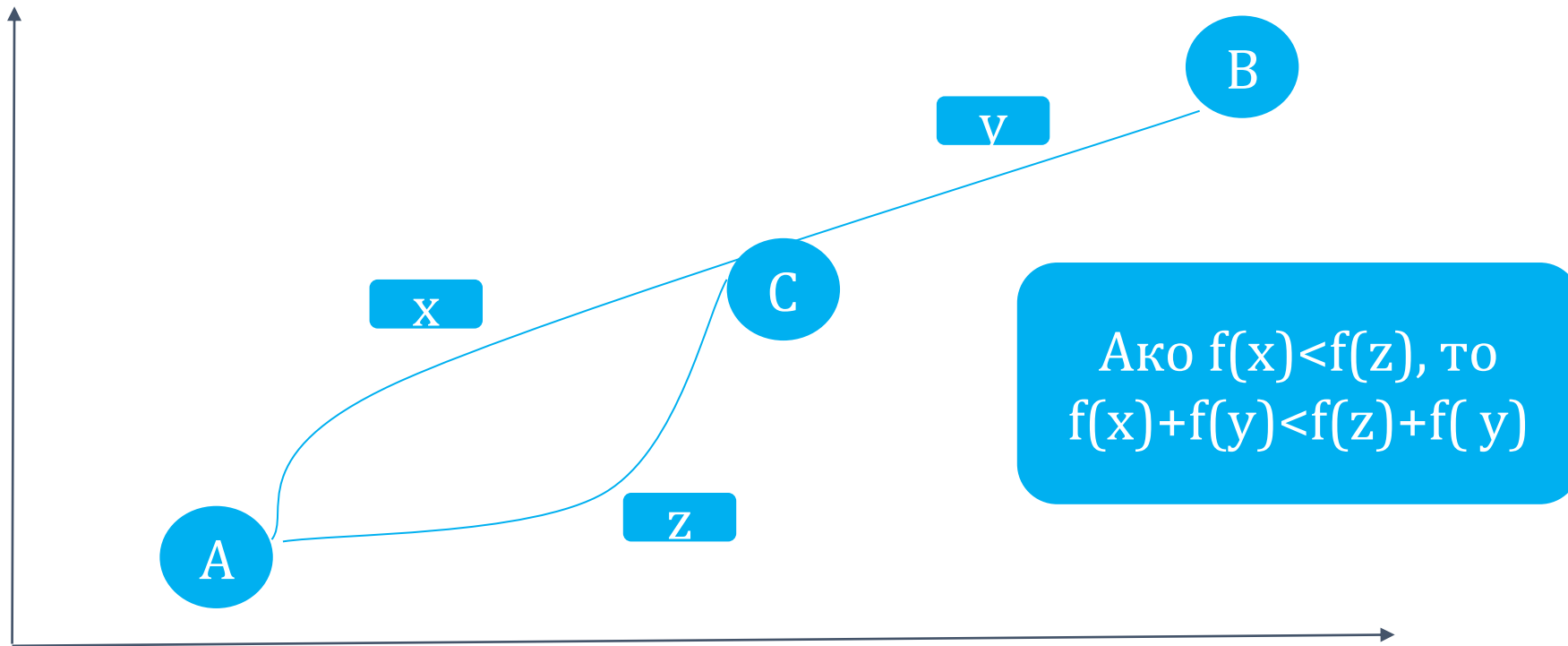


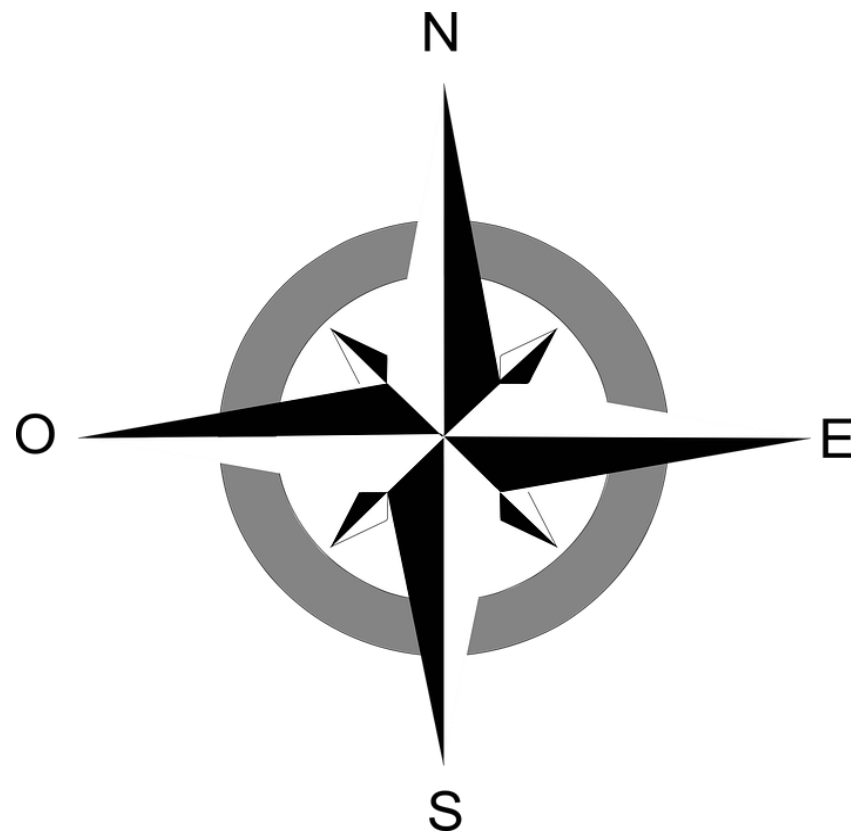
# Двумерно динамично оптимиране [2/3]

Понятието оптимален - най-къс път, който може да се осъществи или минимален разход на гориво, ако движението се извършва с превозно средство.



# Двумерно динамично оптимиране [3/3]





# Задача за пешеходеца

Динамично решение

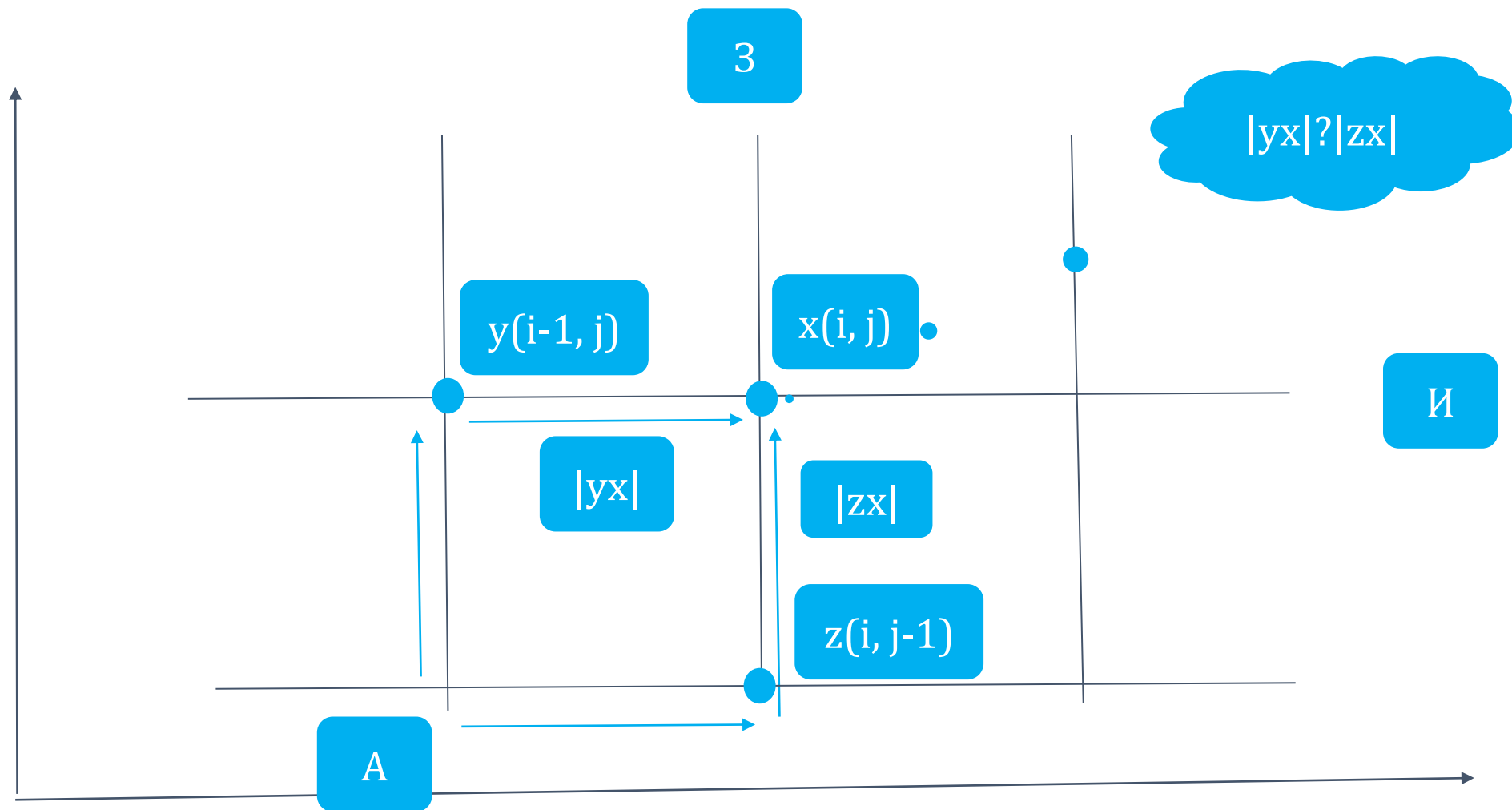
# Задача за пешеходеца [1/5]

Пешеходец трябва да започне движението си от  $m$ . А и да отиде до  $n$ . В, като спазва правилото, че от всяко кръстовище може да тръгне или на север или на изток. За преминаването на всяка отсечка от улица, заключена между две кръстовища, се заплаща определена такса. Какъв маршрут да се избере, така че общата платена сума да е минимална?

# Задача за пешеходец [2/5]

- Формулираме една фамилия от подзадачи
  - за всяко кръстовище  $(i, j)$  означаваме с  $v[i][j]$  оптималното решение на задачата за намиране на маршрут с минимална такса, тръгвайки от  $m$ . А и стигайки до кръстовището с координати  $(i, j)$ .
  - $v[0][0] = 0$ , тъй като цената за преминаване от  $m$ . А до същата точка е нула.
  - $v[0][1]$  е равно на таксата за преминаване по отсечката от  $m$ . А до съседното ѝ кръстовище с координати  $(0,1)$
  - последователно пресмятаме стойностите на  $v[i][j]$  чрез запълването на този масив по редове. Първо зареждаме стойности в  $v[0][0], v[0][1], \dots, v[0][N]$ , след това в  $v[1][0], v[1][1], \dots, v[1][N]$  и т. н.). Получаваме  $v[N][N]$ .

# Задача за пешеходец [3/5]



# Задача за пешеходец [4/5]

- $a[i][j]$  - таксата за преминаване по вертикалната отсечка от кръстовище  $(i, j-1)$  до кръстовище  $(i, j)$ ,
- с  $b[i][j]$  - таксата за преминаване по хоризонталната отсечка от  $(i-1, j)$  до  $(i, j)$ .
- $v[i][j] = \min\{v[i][j-1] + a[i][j], v[i-1][j] + b[i][j]\}$
- $v[i][0] = v[i-1][0] + b[i][0]$  - рекурентна зависимост за първия ред
- $v[0][j] = v[0][j-1] + a[0][j]$  - рекурентна зависимост за първия стълб



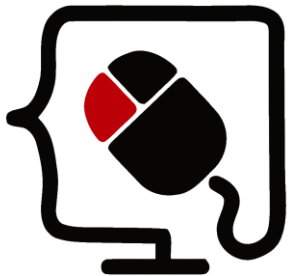
# Задача за пешеходец [5/5]

За да получим самия маршрут:

- при зареждането на всяка от стойностите на  $v[i][j]$  , запомняме откъде е получена тази стойност; по-точно - при кой от двата случая е бил достигнат минимумът: дали при идване „отдолу“ или при идване „отляво“ върху кръстовището  $(i, j)$ .
- използваме допълнителен масив  $w[i][j]$ , чиито елементи да зареждаме с 1, ако в точката  $(i, j)$  оптималният маршрут е дошъл „отляво“ и с 0, ако в същата точка оптималният маршрут е дошъл „отдолу“. Освен това за начална стойност присвояваме  $w[0][0]=-1$ .

# Обобщение

- Динамичното оптимиране
  - решава подзадачи, които се припокриват
  - избира оптималните решения на подзадачите
  - комбинира оптималните решения на подзадачите и получава оптимално общо решение



Национална програма  
"Обучение за ИТ умения и кариера"  
<https://it-kariera.mon.bg>

Министерството на  
образованието и науката  
<https://www.mon.bg>



Документът е разработен за нуждите на Национална програма "Обучение за ИТ умения и кариера" на Министерството на образованието и науката (МОН) и се разпространява под свободен лиценз CC-BY-NC-SA (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).