

Рекурсия, пълно изчерпване и търсене с връщане назад

ИТ Кариера



Учителски екип

Обучение за ИТ кариера

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

Съдържание

- Рекурсия и рекурсивни алгоритми. Упражнения
- Пълно изчерпване и търсене с връщане назад (backtracking).
Задача за осемте царици
- Упражнения: имплементация на backtracking алгоритъм





РЕКУРСИЯ!

Какво е рекурсия?

Какво е рекурсия?

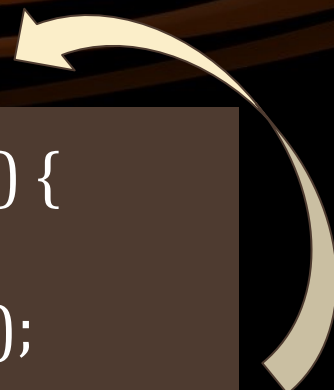
- Техника за решаване на задачи
- Разделя задачата на подзадачи от същия тип
 - Включва функция, извикваща себе си
 - Функцията трябва да има граничен случай
 - Всяка стъпка от рекурсията трябва да се движи към граничния случай

Видове рекурсия

- Пряка рекурсия

Метода А директно се обръща към себе си.

```
void A( ) {  
    ...  
    A();  
}
```

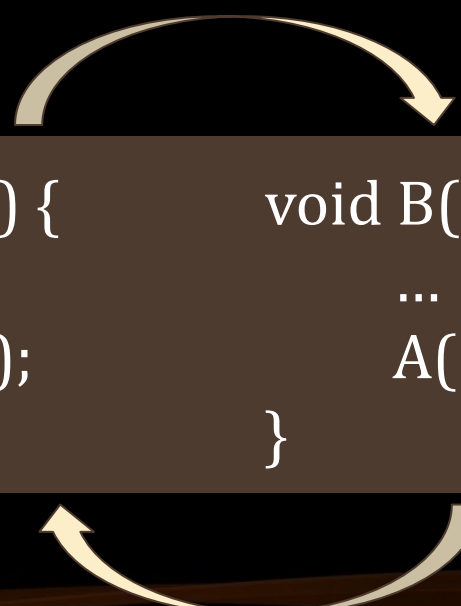


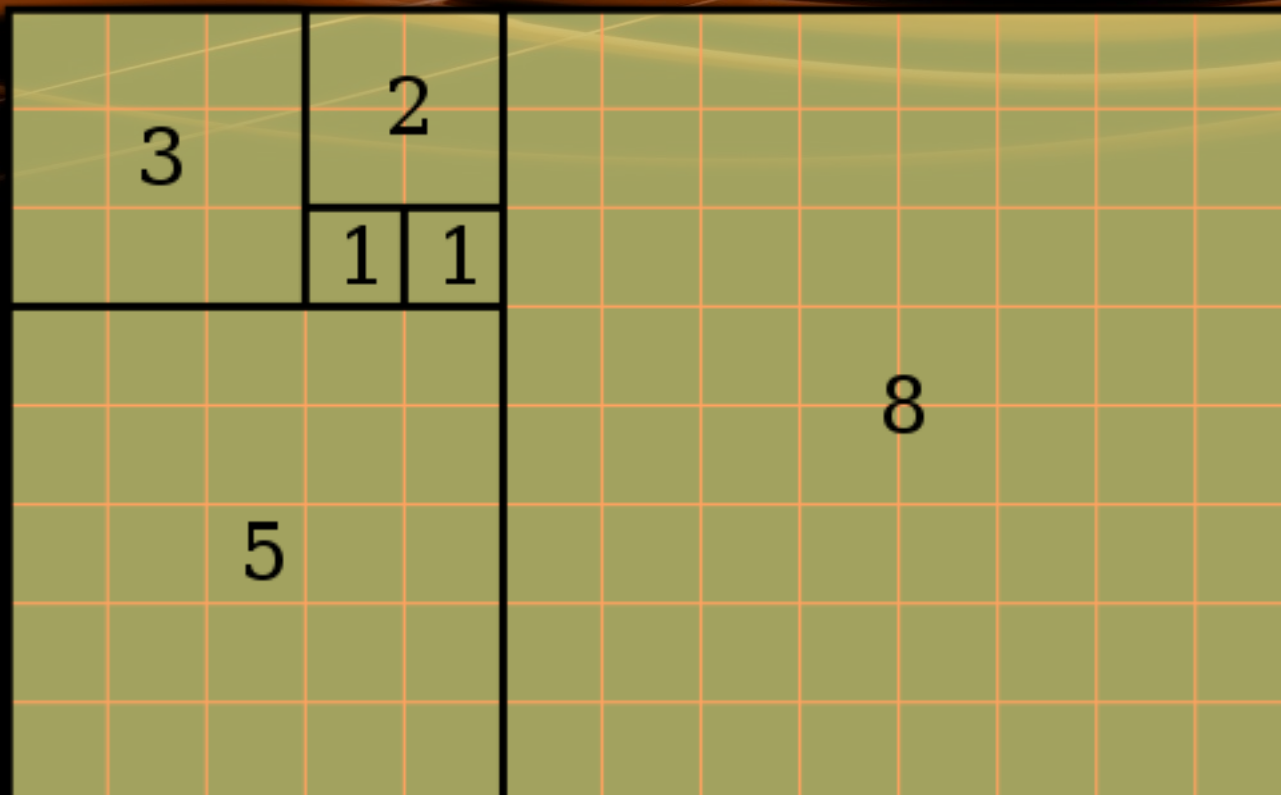
- Непряка рекурсия

А се обръща към В, В се обръща към А.

Възможно е и $A \rightarrow B \rightarrow C \rightarrow A$

```
void A( ) {  
    ...  
    B();  
}  
  
void B( ) {  
    ...  
    A();  
}
```





Редица на Фибоначи

Задача

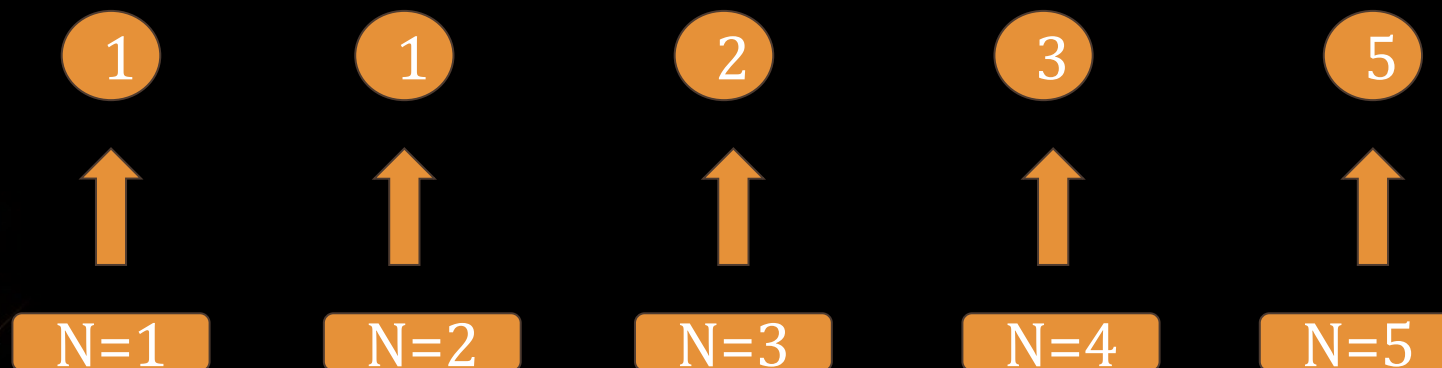
Редица на Фибоначи

- Създайте рекурсивна програма, която:
 - намира N-тия член на редицата на Фибоначи: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...
 - въведете N в конзолата



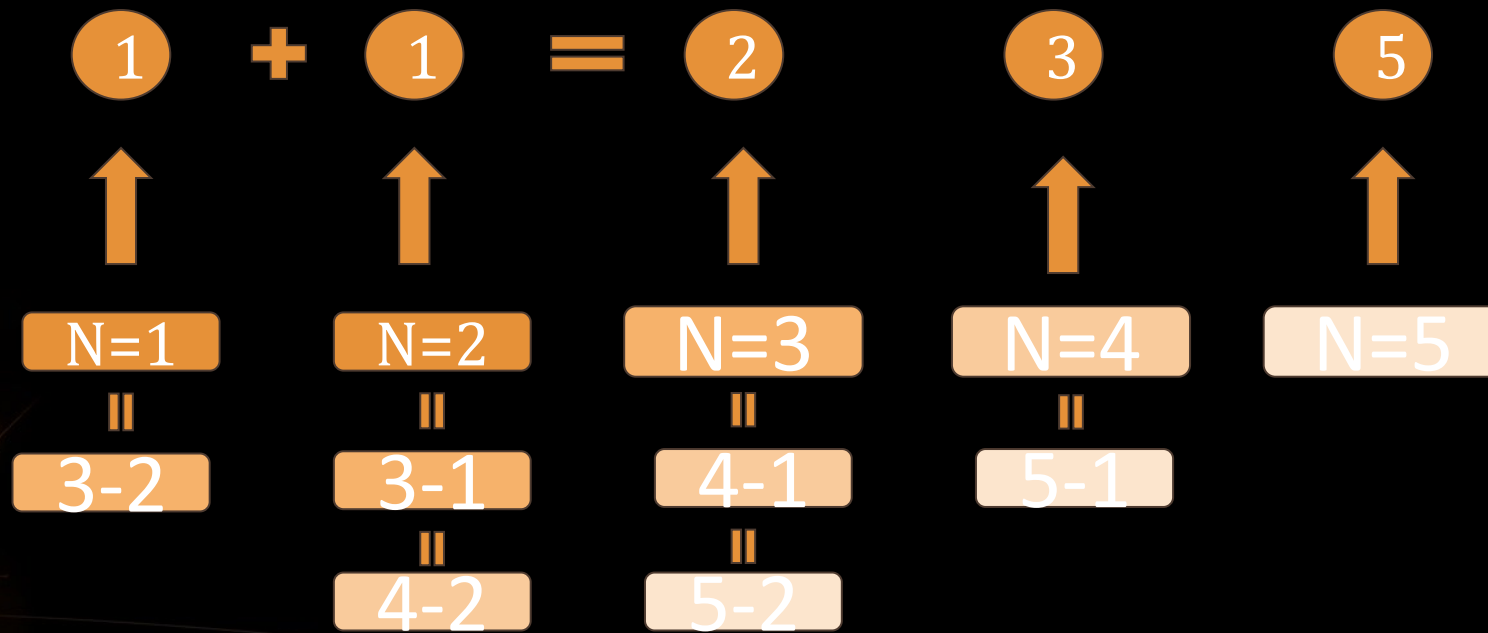
Редица на Фибоначи – етапи на изграждане на рекурсивно решение

1 стъпка: Параметризация - определят се параметрите, свързани с решението на задачата. В случая ни е необходим един параметър N , който показва кой по ред е члена на редицата (първи, втори, трети и т.н.).



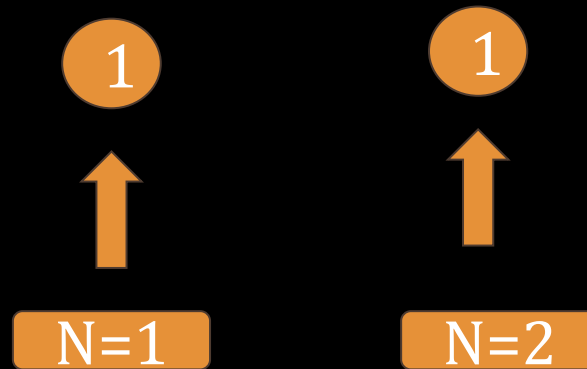
Редица на Фибоначи – етапи на изграждане на рекурсивно решение

2 стъпка: Редукция - намира се рекурсивна връзка между параметрите на дадената задача и тези на подзадачата/ите от същия вид, приличаща на изходната (рекурсивна дефиниция). Редукцията се свързва с процеса на извеждане на N-то число от редицата на Фибоначи. По дефиниция знаем, че то е сума от (N-1) - вото и (N-2) - рото число.



Редица на Фибоначи – етапи на изграждане на рекурсивно решение

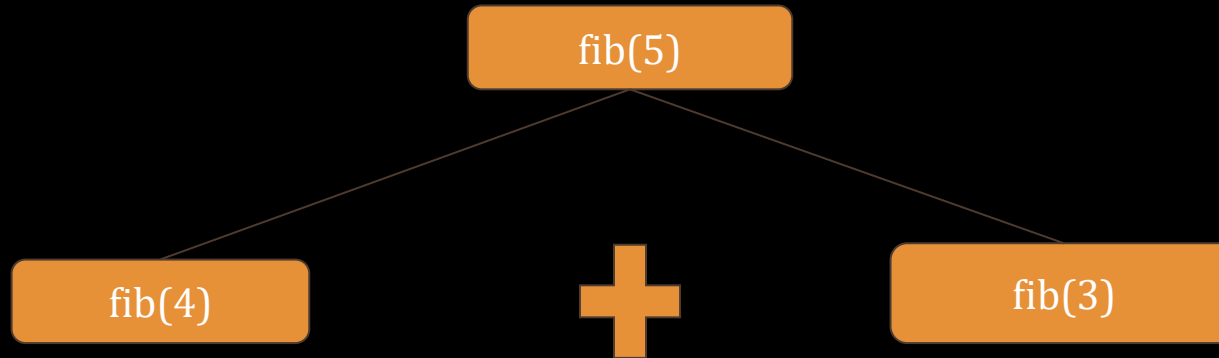
3 стъпка: Условие за край (гранично условие) - определя се подзадача, чието решение е не рекурсивно и води до завършване на алгоритъма. Обикновено се използват гранични елементи - първи, последен.



граничен случай

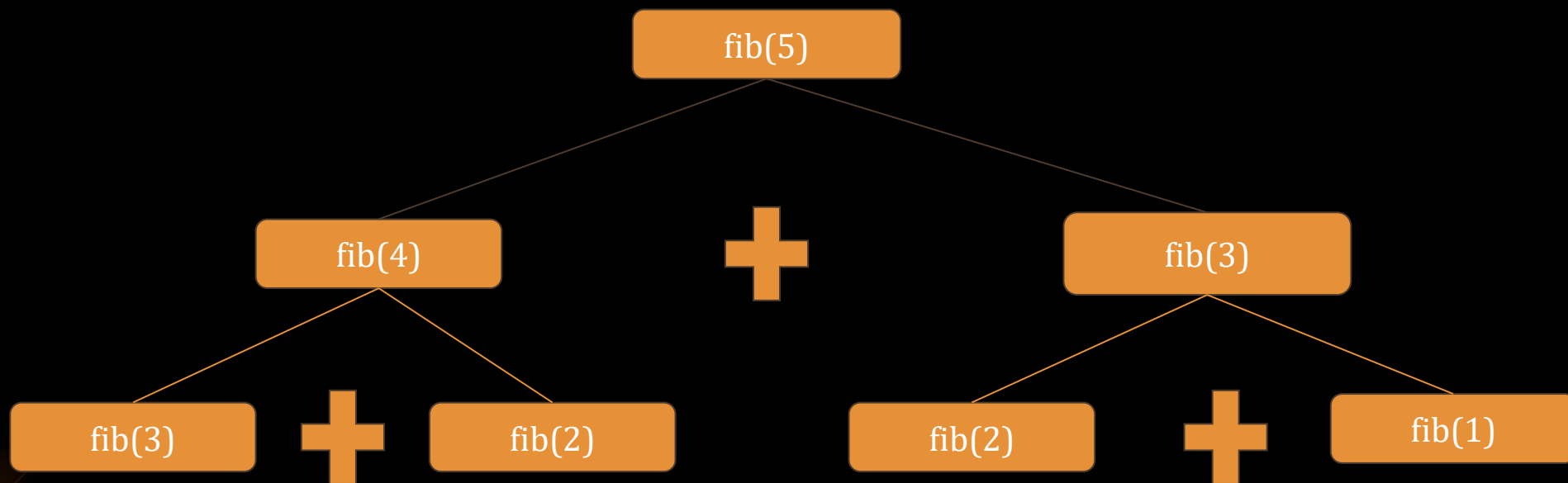
Редица на Фибоначи – механизъм на действие

Разгъване на рекурсията (слизване към дъното, към граничния случай)



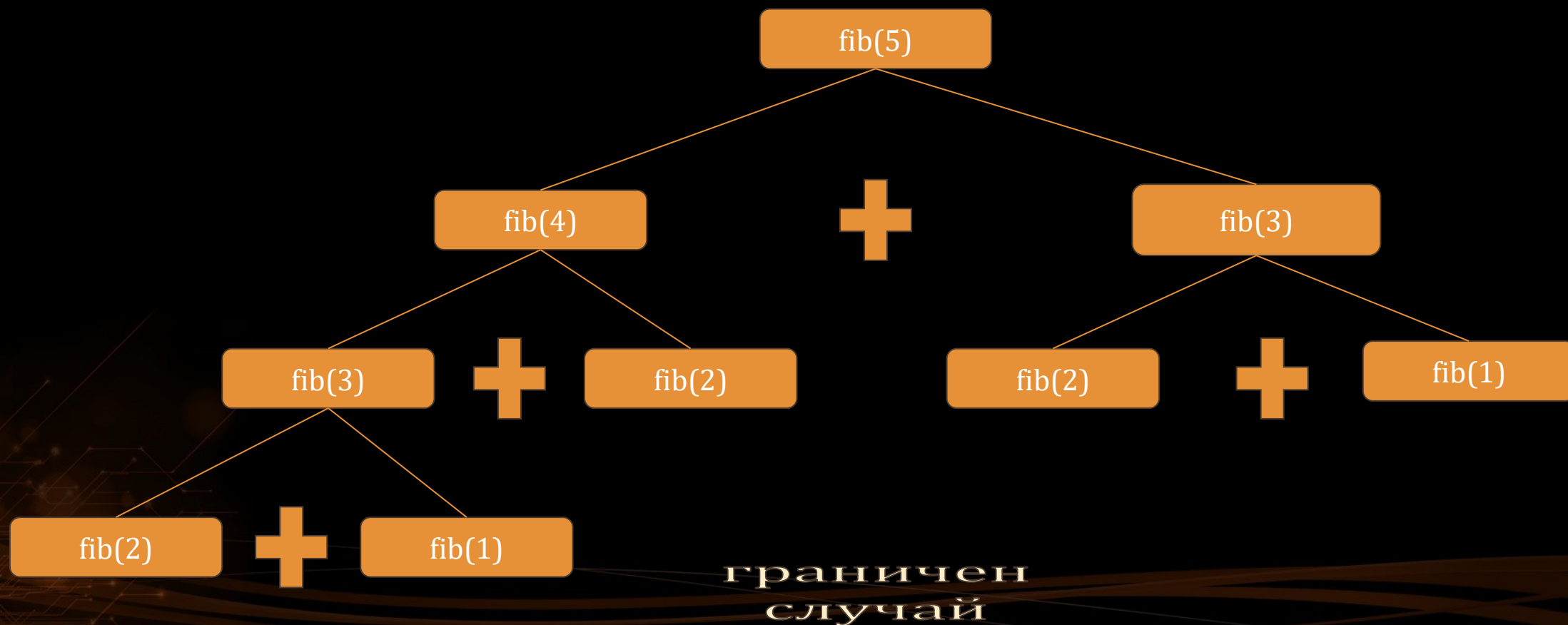
Редица на Фибоначи – механизъм на действие

Разгъване на рекурсията (слизване към дъното, към граничния случай)



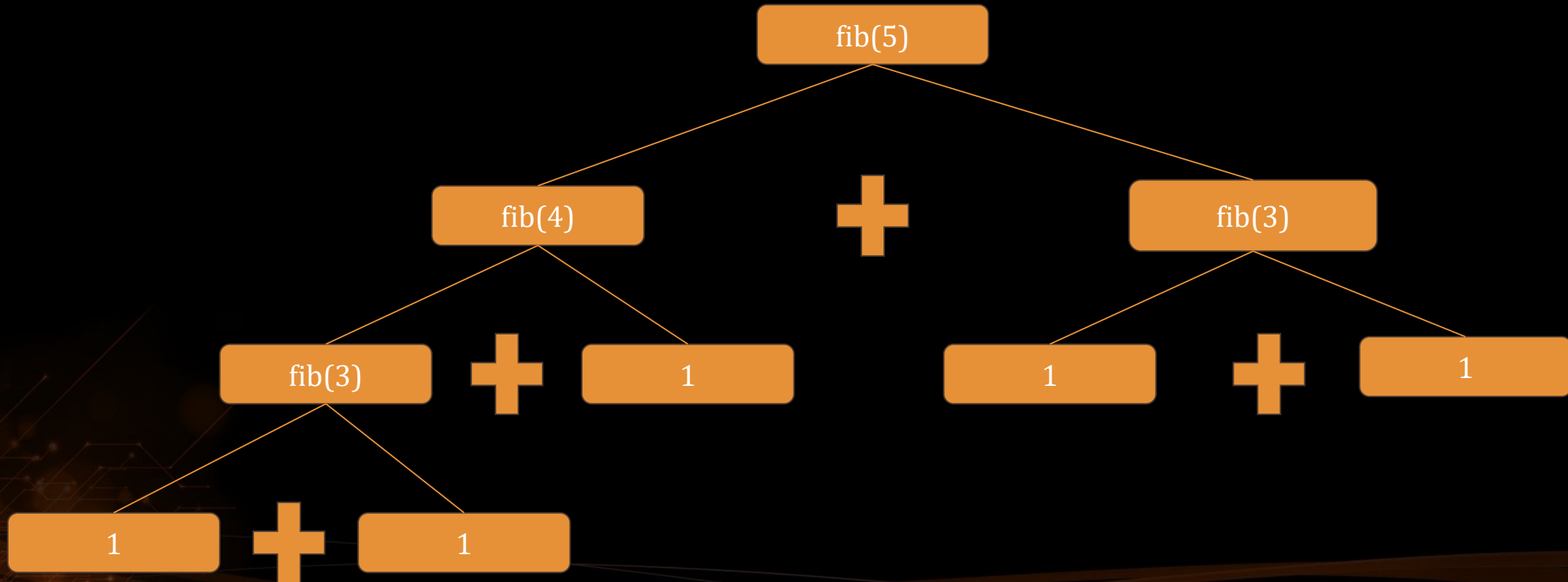
Редица на Фибоначи – механизъм на действие

Разгъване на рекурсията (слизване към дъното, към граничния случай)



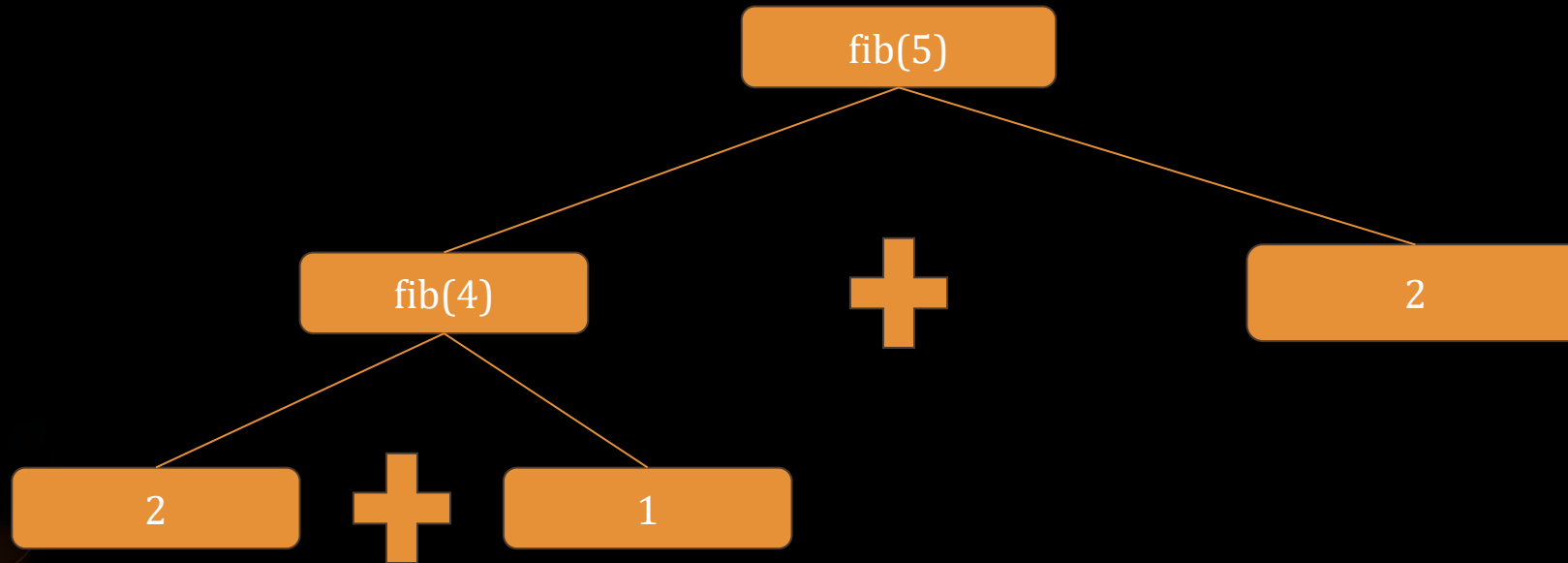
Редица на Фибоначи – механизъм на действие

Свиване на рекурсията



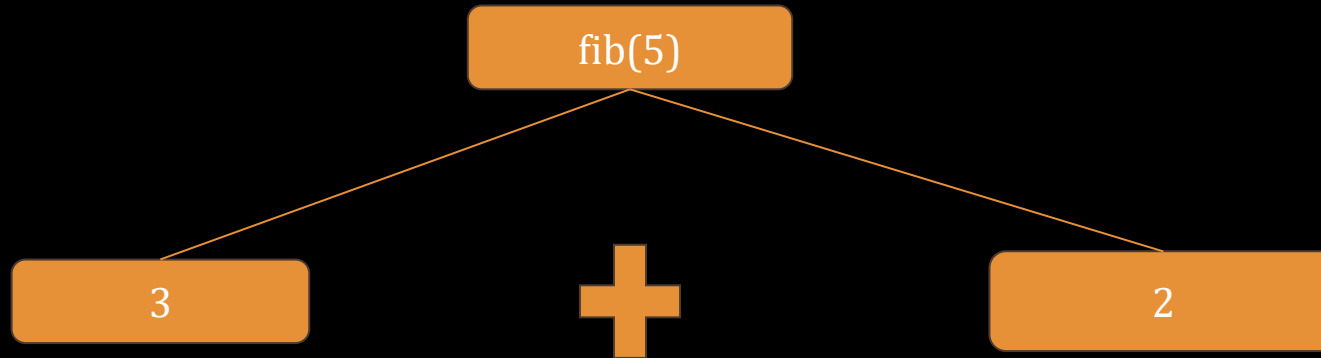
Редица на Фибоначи – механизъм на действие

Свиване на рекурсията



Редица на Фибоначи – механизъм на действие

Свиване на рекурсията



Редица на Фибоначи – механизъм на действие

Свиване на рекурсията

5

край!

Редица на Фибоначи – решение

```
static void Main(string[] args)
{
    int n = 5;

    int fibN = Fib(n);

    Console.WriteLine(fibN);
}

public static int Fib(int n)
{
    if (n == 1 || n == 2)
    {
        return 1;
    }

    return Fib(n - 1) + Fib(n - 2);
}
```




Намиране на най-голям общ делител

Задача

НОД

- Създайте рекурсивна програма, която:
- намира най-големия общ делител на две цели числа a и b
- въведете стойностите на a и b в конзолата

$a=35$ $b=14$

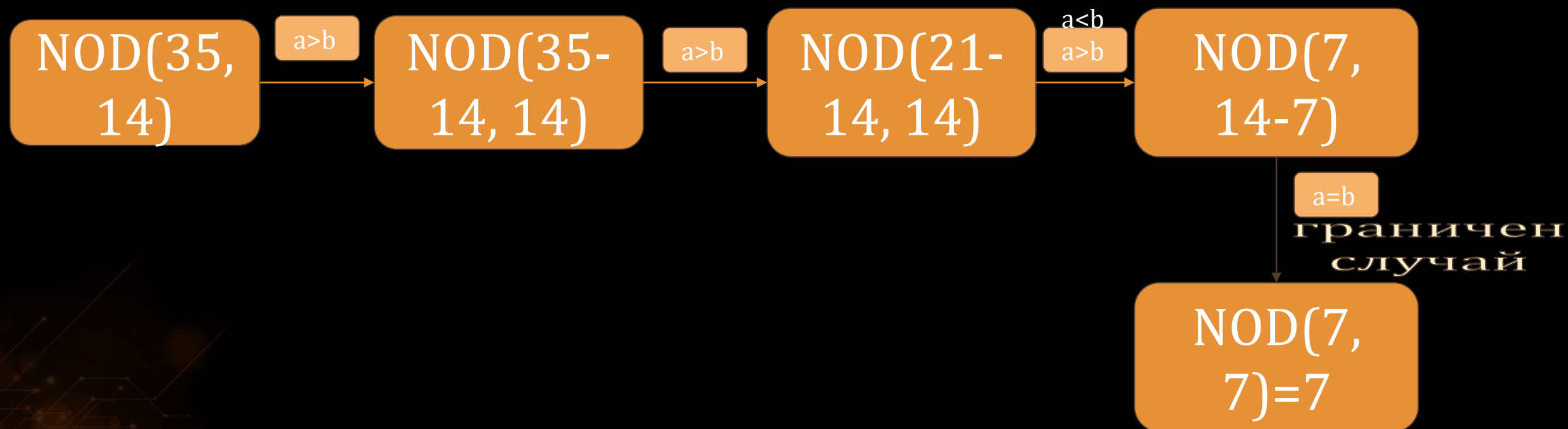


7

НОД. Алгоритъм на Евклид с изваждане – алгоритъм

- Стъпка 1. Въведете стойности за a и b .
- Стъпка 2. Ако $a \neq b$ изпълни изпълни стъпка 3.
 - ако $a = b$ изпълни стъпка 5.
- Стъпка 3. Ако $a > b$ изпълни $a = a - b$;
 - ако $a < b$ изпълни $b = b - a$;
- Стъпка 4. Изпълни стъпка 2.
- Стъпка 5. Изведи a .
- Стъпка 6. Край на алгоритъма.

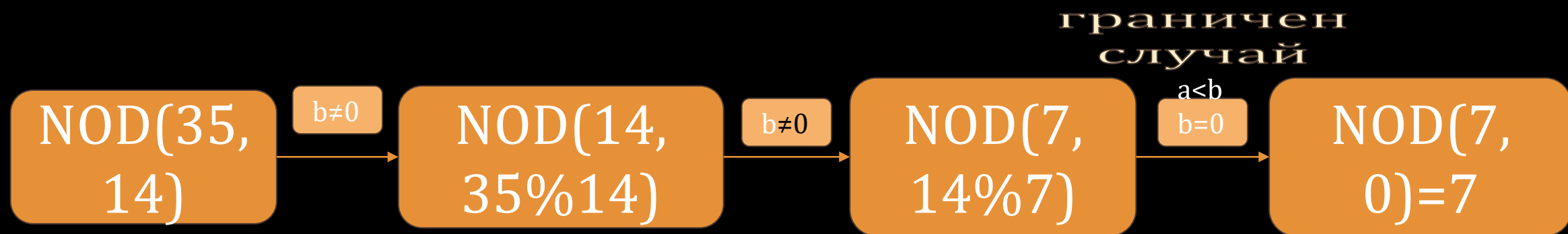
НОД. Алгоритъм на Евклид с изваждане



НОД. Алгоритъм на Евклид с деление – алгоритъм

- Стъпка 1. Въведете стойности за a и b .
- Стъпка 2. Ако $b > 0$ изпълни изпълни стъпка 3.
 - ако $b=0$ изпълни стъпка 4.
- Стъпка 3. Изпълни:
 - $r=b$;
 - $b=a\%b$;
 - $a=r$;
- Стъпка 4. Изведи a .
- Стъпка 5. Край на алгоритъма.

НОД. Алгоритъм на Евклид с деление



Намиране на най-малко общо кратко Задача

НОК

- Създайте рекурсивна програма, която:
 - намира най-малкото общо кратно на две цели числа a и b
 - въведете стойностите на a и b в конзолата



$$\text{НОК}(a, b) = \frac{a * b}{\text{НОД}(a, b)}$$

Какво е Backtracking?

- Клас алгоритми за намиране на всички решения на някаква комбинаторна задача.
- В това число намиране на всички пътища от София до Варна

Как работи backtracking?

- На всяка стъпка рекурсивно се преглеждат всички перспективни възможности
 - Всички не перспективни възможности се отхвърлят възможно най-рано
 - Времето на работа е експоненциално

Backtracking

- Backtracking поставя въпроси като:
 - Колко начина за...съществуват?
 - Има ли начин за...
 - Кой са всички възможни решения?
 - Кое е оптималното решение?

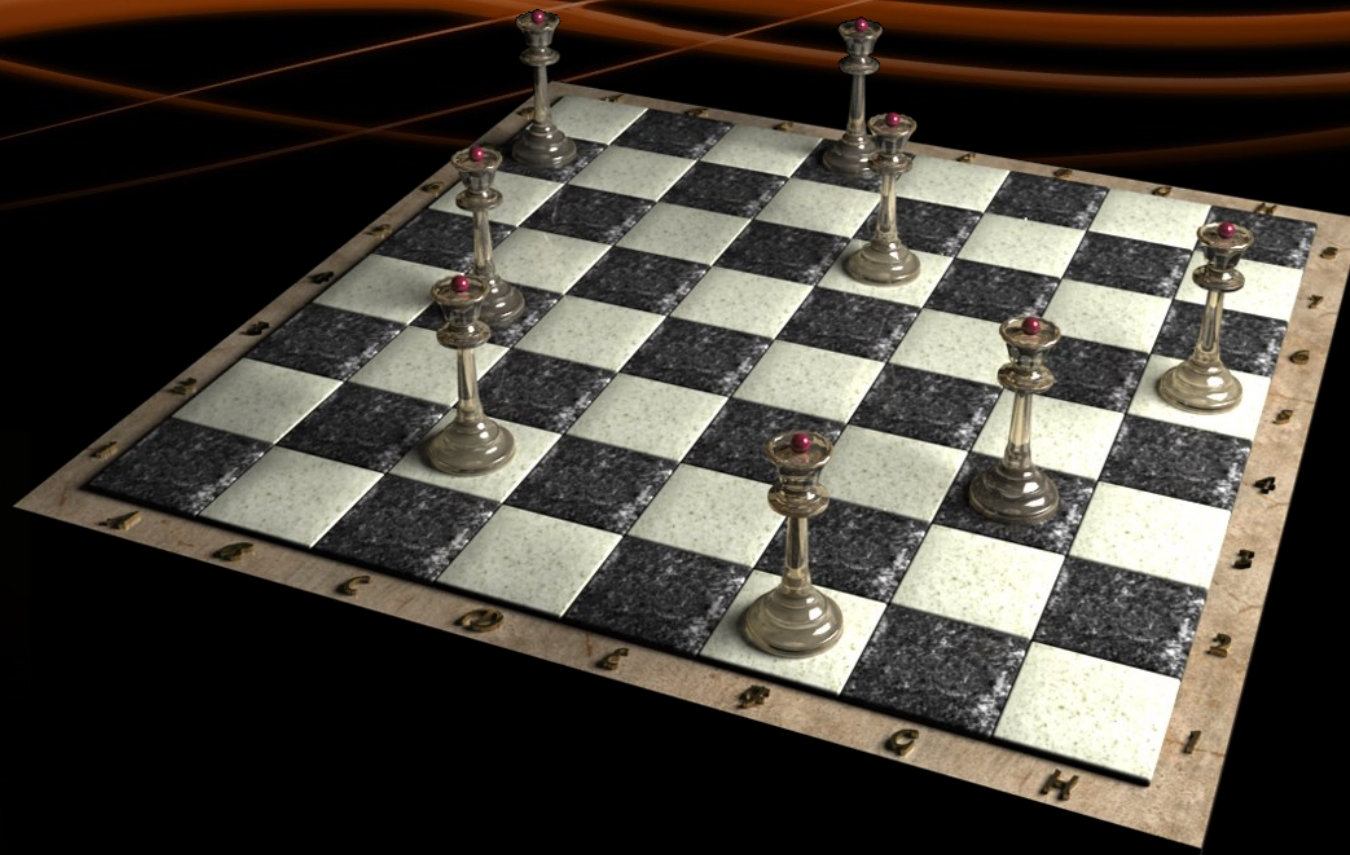
Backtracking – алгоритъм

1 стъпка: Конструира се едно частично решение.

2 стъпка: Проверява се дали частичното решение е общо (търсеното). Ако е така, решението се запомня или извежда и процесът на търсене или завършва, или продължава по същата схема, докато бъдат генерирани всички възможни решения.

3 стъпка: В противен случай се прави опит текущото частично решение да се продължи (според условието на задачата).

4 стъпка: Ако на някоя стъпка се окаже невъзможно ново разширяване, извършва се връщане назад към предишното частично решение и се прави нов опит то да се разшири (продължи) по друг, различен от предишния, начин.



Задача за осемте царици

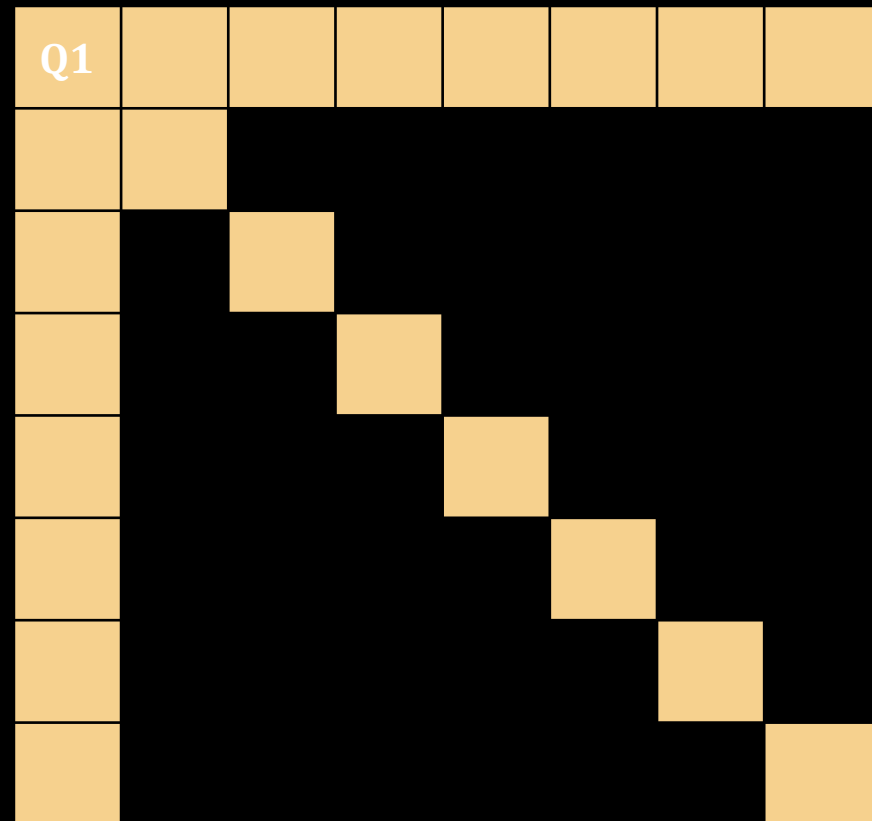
Задача за осемте царици

Шахматната дъска е разделена на 64 клетки, образуващи осем хоризонтала и осем вертикала. Най-силната шахматна фигура е царицата, тъй като тя може да атакува всяка клетка по хоризонтала, вертикала и диагонала. В задачата с осемте царици се търси решение на това как те да бъдат разположени на шахматната дъска така, че никоя от тях да не е нападната от останалите.

Задача за осемте царици

Ще поставяме по една царица (Q1, Q2,Q8) във всяка вертикала, започвайки с първата клетка по вертикал и хоризонтал.

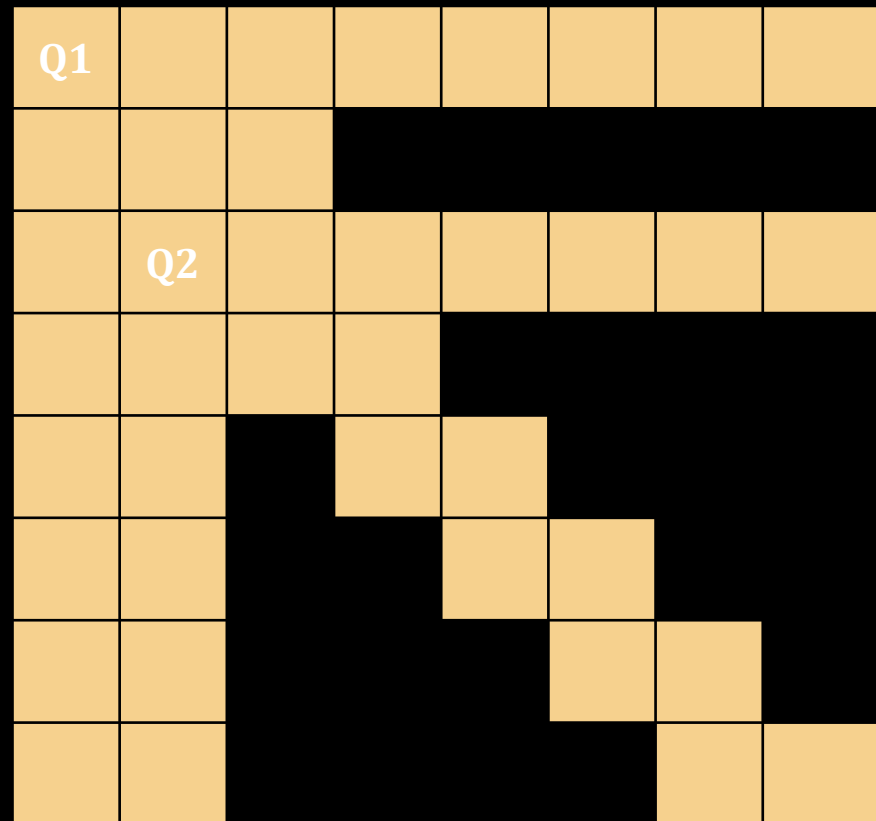
На фигурата вдясно са оцветени клетките, които тя може да атакува.



Задача за осемте царици

Проследяваме възможностите за поставяне на втората царица по втория вертикал.

На фигурата вдясно са оцветени клетките, които са атакувани до момента.



Задача за осемте царици

Проследяваме възможностите за поставяне на третата царица по третия вертикал.

На фигурата вдясно са оцветени клетките, които са атакувани до момента.

Q1							
	Q2						
		Q3					

Задача за осемте царици

Проследяваме възможностите за поставяне на четвъртата царица по четвъртия вертикал.

На фигурата вдясно са оцветени клетките, които са атакувани до момента.

Q1							
			Q4				
	Q2						
		Q3					

Задача за осемте царици

Проследяваме възможностите за поставяне на петата царица в петия вертикал.

На фигурата вдясно са оцветени клетките, които са атакувани до момента.

Оказа се, че стигнахме до “задънена улица”.

Q1							
			Q4				
	Q2						
				Q5			
		Q3					

Задача за осемте царици

Връщаме се на предходната стъпка, за да помислим за друга алтернатива.

Q1							
			Q4				
	Q2						
		Q3					

Задача за осемте царици

Поставяме Q5 в петата клетка на последния ред. В предишното решение тя беше в четвъртата клетка отгоре надолу.

На фигурата вдясно са оцветени клетките, които са атакувани до момента.

Q1							
			Q4				
	Q2						
		Q3					
				Q5			

Задача за осемте царици

Поглеждаме към шестата вертикала и виждаме, че там нямаме възможност да поставим царица.

Изчерпвайки всички възможни места за царицата по петата вертикала, трябва да се върнем към четвъртата вертикала.

Q1							
			Q4				
	Q2						
		Q3					
				Q5			

Задача за осемте царици

Проследяваме нова възможност за поставяне на четвъртата царица в четвъртия вертикал (в предишното решение тя беше във втората клетка).

Сега ще я поставим в седма клетка отгоре надолу.

На фигурата вдясно са оцветени клетките, които са атакувани до момента.

Q1							
	Q2						
		Q3					
			Q4				

Задача за осемте царици

Поставяме Q5 в петия вертикал, втора клетка отгоре надолу.

На фигурата вдясно са оцветени клетките, които са атакувани до момента.

Q1							
				Q5			
	Q2						
		Q3					
			Q4				

Задача за осемте царици

Поставяме Q6 в шести вертикал.

На фигурата вдясно са оцветени клетките, които са атакувани до момента.

Q1							
				Q5			
	Q2						
					Q6		
		Q3					
			Q4				

Задача за осемте царици

Поставяме Q7 в седми вертикал.

На фигурата вдясно са оцветени клетките, които са атакувани до момента.

Оказва се, че в осми вертикал е невъзможно да поставим последната царица. Отново “задънена улица”.

Q1							
				Q5			
	Q2						
					Q6		
		Q3					
						Q7	
			Q4				

Задача за осемте царици

Отново се връщаме на предходните стъпки и мислим за други алтернативи.

На фигурата вдясно е показано крайното решение на задачата.

Q1							
						Q7	
				Q5			
							Q8
	Q2						
			Q4				
					Q6		
		Q3					

Обобщение

- Рекурсията е:
 - Метод, който се обръща към себе си
 - Много мощна техника за прилагане на комбинаторни алгоритми
- Всяка рекурсия трябва да има граничен случай
- Рекурсията може да бъде вредна, ако не се използва правилно



Обобщение

- Backtracking намира:
 - Всички решения/оптимално решение на комбинаторна задача чрез генериране на всички възможности
 - Отстранява не перспективните възможности



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**

