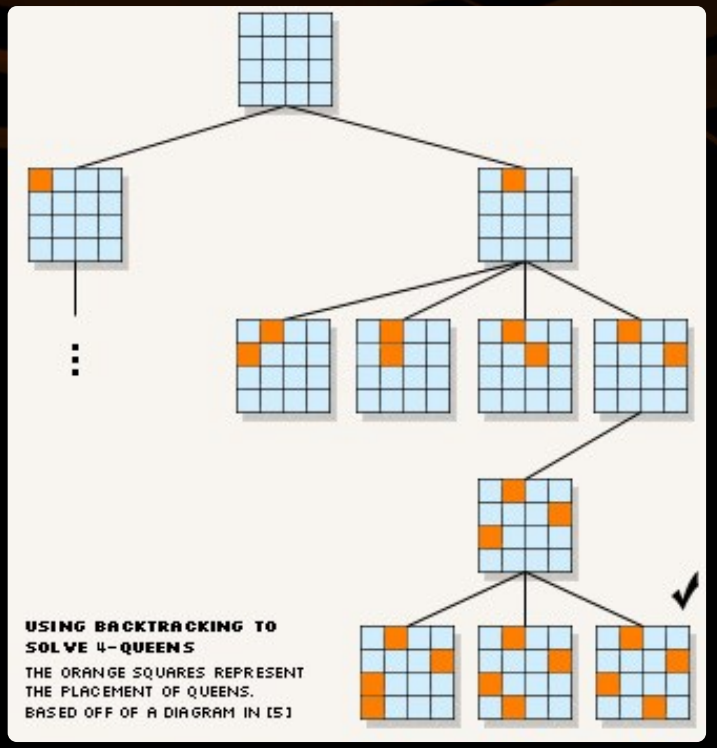
Алгоритми върху линейни структури от данни. Бектракинг

Учителски екип

Обучение за ИТ кариера

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

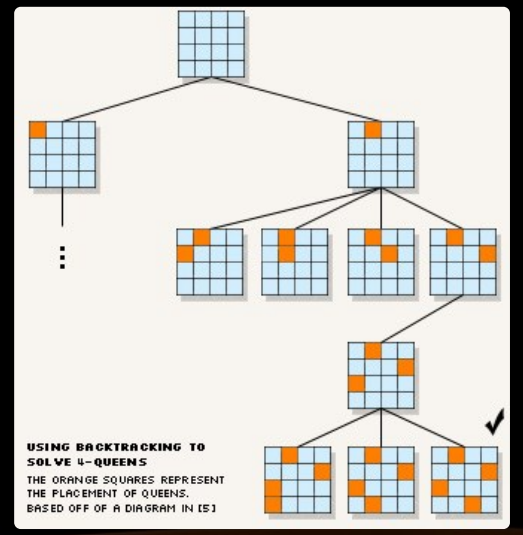


Съдържание

1. Що е "бектракинг" (търсене с връщане назад)
2. Задачата за 8-те царици
3. Рекурсивно намиране на всички пътища в лабиринт

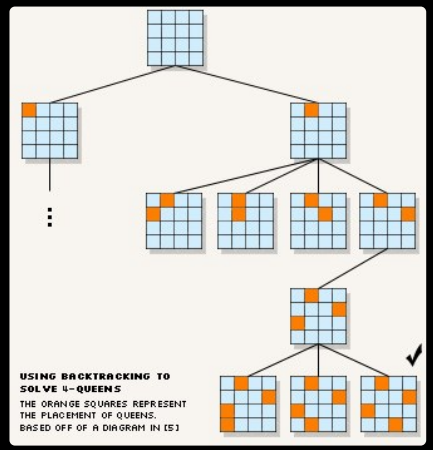
Бектракинг (търсене с връщане назад)

- Какво е **бектракинг**? Генериране на всички кандидати
  - Множество от алгоритми за **намиране на всички решения** за дадена комбинаторна задача
    - Например: Намиране на всички пътища от София до Варна



Бектракинг

- Как работи връщането назад?
  - На всяка стъпка рекурсивно **се опитват всички перспективни възможности**
  - **Отхвърлят се** всички **неперспективни възможности** колкото е възможно по-рано
- Връщането назад има **експоненциално време за изълнение**!

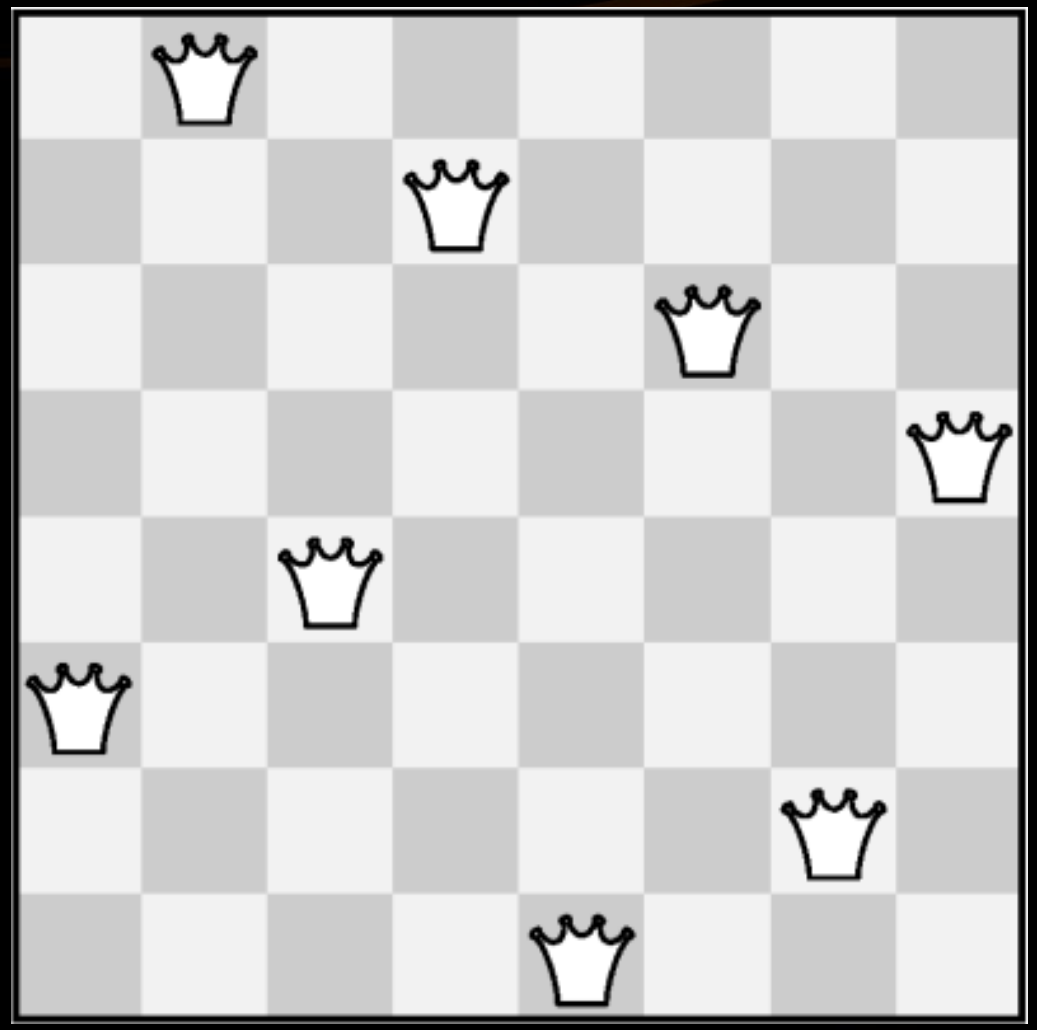


Бектракинг алгоритми (псевдокод)

```
void Backtracking(Node node)
{
  if (node is solution)
    PrintSolution(node);
  else
    for each child c of node
      if (c is perspective candidate)
      {
        MarkPositionVisited(c);
        Backtracking(c);
        UnmarkPositionVisited(c);
      }
}
```

Пъзелът „8-те царици“

- Напишете програма, която да намери всички възможни разположения на
  - 8 царици на шахматното поле
  - така, че да няма две царици, които да могат да се нападнат взаимно http://en.wikipedia.org/wiki/Eight_queens_puzzle



Решаване на пъзела „8-те царици“

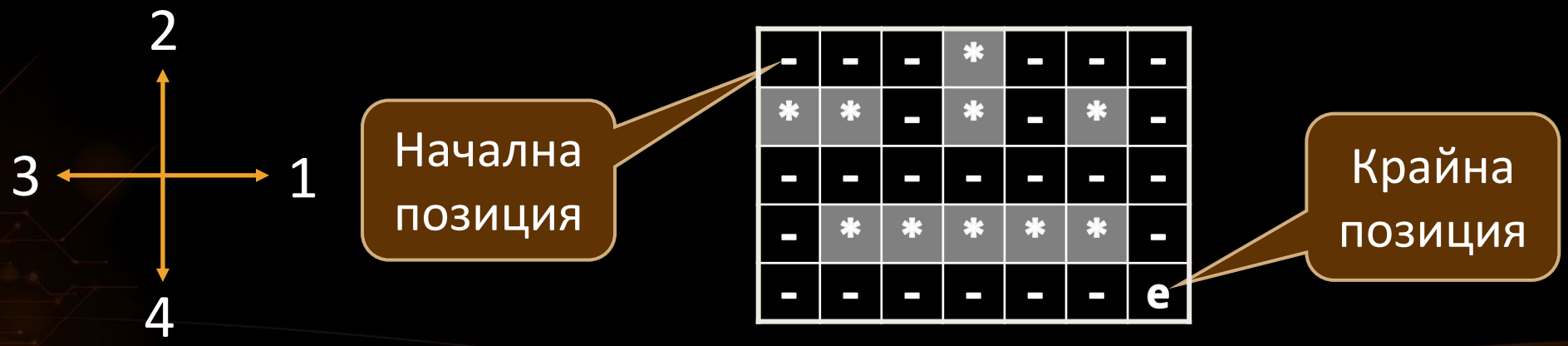
- Намери всички решения на "8 царици"
- На всяка стъпка:
  - Сложете царицата на свободна позиция
  - Извикайте рекурсивно функциятая
  - Махнете царица

```
void PutQueens(row)
{
  if (row == 8)
    PrintSolution();
  else
    for (col = 0 … 7)
      if (CanPlaceQueen(row, col))
      {
        MarkAllAttackedPositions(row, col);
        PutQueens(row + 1);
        UnmarkAllAttackedPositions(row, col);
      }
}
```

Намиране на всички пътища в лабиринт

- Даден е **лабиринт**
  - Представен като матрица от клетки с размер M x N

  Са проходими, останалите с (*) – не са
- **Започваме от горния ляв** ъгъл и **се движим в 4-те посоки**
- Търсим **всички пътища до изхода**, маркирани с '**e**'



Намиране на всички пътища в лабиринт (2)

- Съществуват **3 различни пътища** от горния ляв ъгъл д долния десен ъгъл:

1)

0	1	2	*			
*	*	3	*		*	
6	5	4				
7	*	*	*	*	*	
8	9	10	11	12	13	14

RRDDLLDDRRRRRR

2)

0	1	2	*	8	9	10
*	*	3	*	7	*	11
		4	5	6		12
	*	*	*	*	*	13
						14

RRDDRRUURRDDDD

3)

0	1	2	*			
*	*	3	*		*	
		4	5	6	7	8
	*	*	*	*	*	9
						10

RRDDRRRRDD

Намиране на всички пътища: Алгоритъм

- Структурата от данни е двумерен масив → матрица от символи:

```
static char[,] lab =
{
    {'-', '-', '-', '*', '-', '-', '-'},
    {'*', '*', '-', '*', '-', '*', '-'},
    {'-', '-', '-', '-', '-', '-', '-'},
    {'-', '*', '*', '*', '*', '*', '-'},
    {'-', '-', '-', '-', '-', '-', 'e'},
};
```

- Тиретата '-' са **проходими** клетки
- Звездичките '*' са **непроходими**
- Символът '**e**' **е край** (може да го има няколко пъти)

Намиране на всички пътища: Алгоритъм (2)

```
private static void FindPath(int row, int col)
{
  if (!IsInBounds(row, col)) { return; }

  if (IsExit(row, col)) { Console.WriteLine("Path found!"); }
  else if (!IsVisited(row, col) && IsPassable(row, col))
  {
    Mark(row, col);
    FindPath(row, col + 1); // Right
    FindPath(row + 1, col); // Down
    FindPath(row, col - 1); // Left
    FindPath(row - 1, col); // Up
    Unmark(row, col);
  }
}
```

Намерете всички пътища и ги изведете

- Създайте `List<char>`, който ще пази пътя
- Преминете в посока при всяко рекурсивно повикване (**L**, **R**, **U** или **D** - за ляво, дясно, горе, долу)
- В началото на всяко рекурсивно обръщение (извикване) → **Добавете посока**
- В края на всяко рекурсивно обръщение (извикване) → **Премахнете посока**

Намерете всички пътища и ги изведете(2)

```
private static void FindPath(int row, int col, char direction)
{
  if (!IsInBounds(row, col)) { return; }

  path.Add(direction);

  if (IsExit(row, col)) { PrintPath(); }
  else if (!IsVisited(row, col) && IsFree(row, col))
  {
    Mark(row, col);
    FindPath(row, col + 1, 'R');
    FindPath(row + 1, col, 'D');
    FindPath(row, col - 1, 'L');
    FindPath(row - 1, col, 'U');
    Unmark(row, col);
  }

  path.RemoveAt(path.Count - 1);
}
```

Рекурсията в бектракинга

- Бектракинг (връщане назад) служи за намиране на всички/оптимални решения на комбинаторни проблеми, чрез създаване на всички възможни решения
  - Премахват се неперспективните кандидати

Алгоритми върху линейни структури от данни. Бектракинг (търсене с връщане назад)

Въпроси?

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

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"

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**