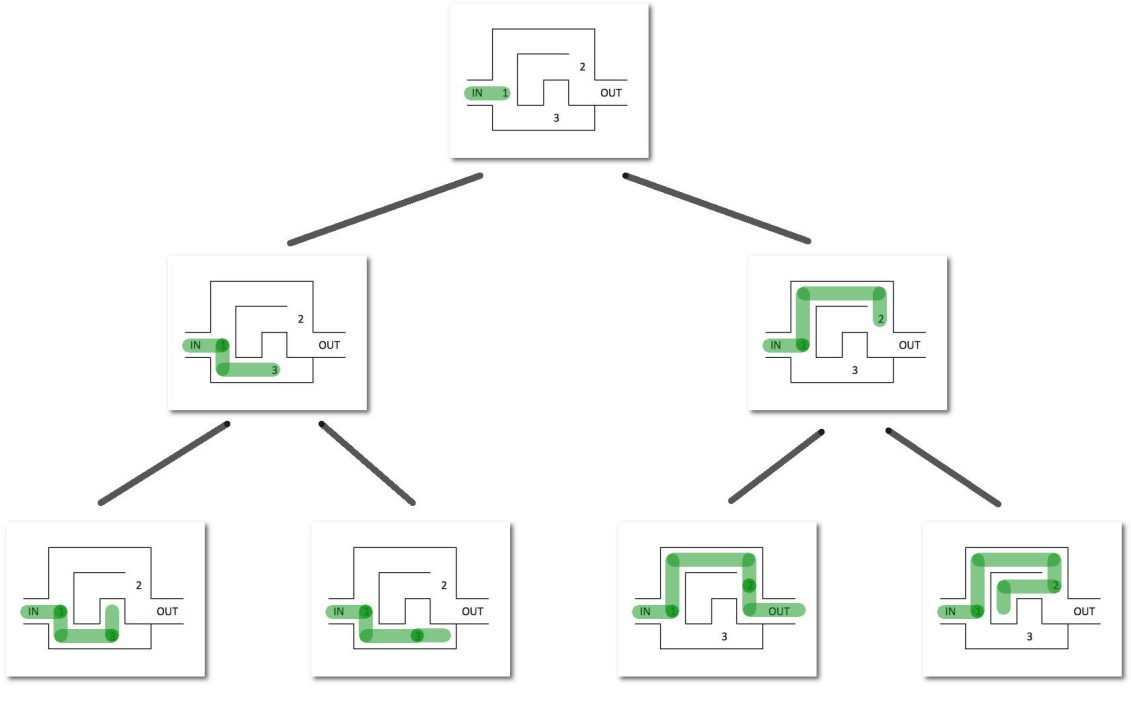
Търсене с връщане назад (Backtracking)

УП Практикум, 2ра група
Виолета Кастрева
Богомил Стоянов

Какво е бектрекинг? 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



Псевдокод

```
void FIND_SOLUTIONS( parameters):
  if (valid solution):
    store the solution
    Return
  for (all choice):
    if (valid choice):
      APPLY (choice)
      FIND_SOLUTIONS (parameters)
      BACKTRACK (remove choice)
  Return
```

За какво се изполва най-често бектрекинга?

Решаване на пъзели:

- Судоку: Попълване на судоку дъска при спазване на нейните ограничения.
- Кръстословици: Откриване на думи в матрица въз основа на пресичащи се букви.
- The N-Queens Problem: Поставяне на N дами на N×N шахматна дъска, така че две дами да могат да се ударят една друга.

За какво се изполва най-често бектрекинга?

Комбинаторни проблеми:

- Пермутации и комбинации: Генериране на всички възможни подредби на даден набор от елементи.
- Сума на подмножество: Намиране на подмножества на множество, които като сбор дават дадена сума.
- Доста други задачи, свързани с подмножества

Алгоритми с графи:

- Хамилтонови цикли: Намиране на цикъл, който посещава всеки връх в графа точно веднъж.
- Задачи с оцветяване: Присвояване на цветове на върховете на графа, така че да няма два съседни върха с един и същи цвят (и подобни).

Пример: Решаване на судоку с бектрекинг

Първо си въвеждаме помощна функция

```
bool isValid(char board[9][9], int row, int col, char num) {
    for (int x = 0; x < 9; x++) {
        if (board[row][x] == num || board[x][col] == num ||
            board[3 * (row / 3) + x / 3][3 * (col / 3) + x % 3] == num) {
            return false;
        }
    }
    return true;
}

```

```
bool solveSudoku(char board[9][9]) {
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            if (board[i][j] == '.') {
                for (char c = '1'; c <= '9'; c++) {
                    if (isValid(board, i, j, c)) {
                        board[i][j] = c;
                        if (solveSudoku(board))
                            return true;
                        board[i][j] = '.'; // backtrack
                    }
                }
                return false; // trigger backtracking
            }
        }
    }
    return true; // solution found
}
```

Край