Week 1! Assignment 1

Vidyam Classes
Excellence in No Time

Sudoku Solver

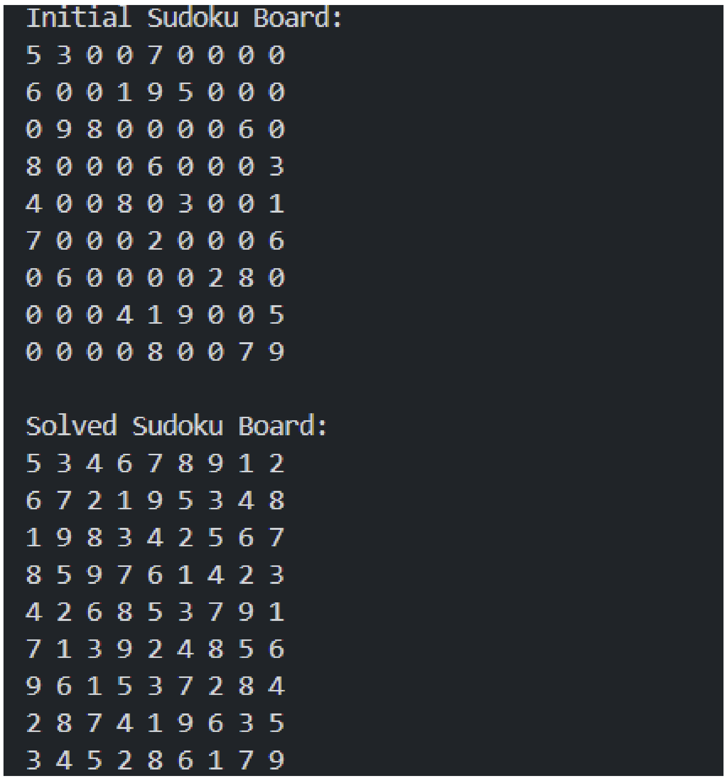
To implement a Sudoku solver using the backtracking algorithm. This assignment will help you understand the concepts of backtracking, recursion, and constraint satisfaction problems.

Problem Statement:

You are required to create a program that can solve a given 9x9 Sudoku puzzle. The program should take a partially filled Sudoku board as input and produce a completed Sudoku board as output, following the rules of Sudoku.

Techniques Used:

1. **Backtracking Algorithm**: This is a depth-first search algorithm used for solving constraint satisfaction problems. It tries to build a solution incrementally and abandons a path as soon as it determines that this path cannot lead to a valid solution.
2. **Recursion**: The backtracking algorithm is implemented using recursion, where the function calls itself to try and solve smaller sub-problems.



```
Initial Sudoku Board:
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9

Solved Sudoku Board:
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```

Assignment 2

Maze Solve (Graph)

Objective:

Implement a maze solver using graph traversal algorithms. Represent the maze as a graph where each cell in the maze is a node, and edges exist between adjacent cells (up, down, left, right) if there is no wall between them. The solver should find a path from a specified start node to an end node.

Problem Statement:

You are given a maze represented as a graph where:

- Each cell in the maze is a node labeled with integers.
- Each node connects to its adjacent cells (if they are not walls).

You need to implement a function that finds a path from the start node to the end node using graph traversal algorithms (BFS or DFS). The function should return the path as a list of node labels.

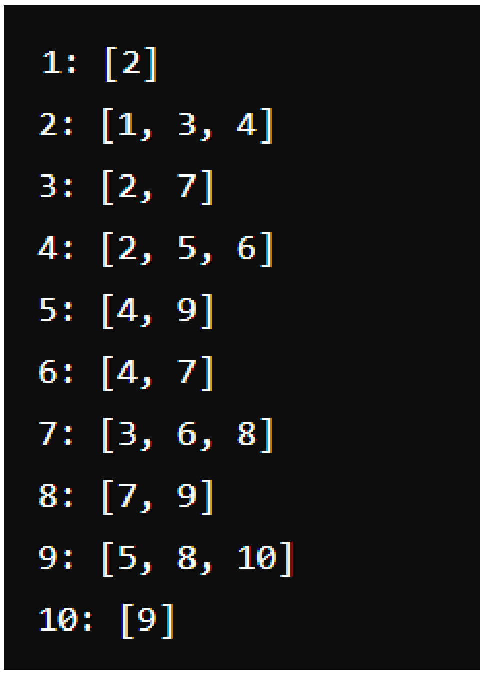
Input:

- A graph mazeGraph represented as an adjacency list.
- The label of the start node startNode.
- The label of the end node endNode.

Output:

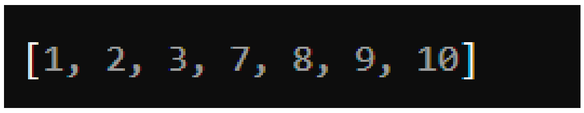
- A list of node labels representing the path from the start to the end node.
- If no path exists, return an empty list.

INPUT



```
1: [2]
2: [1, 3, 4]
3: [2, 7]
4: [2, 5, 6]
5: [4, 9]
6: [4, 7]
7: [3, 6, 8]
8: [7, 9]
9: [5, 8, 10]
10: [9]
```

OUTPUT



```
[1, 2, 3, 7, 8, 9, 10]
```