Algorithms & Analysis 2025A – Group Project

Undergraduate Project – Sudoku Solver

In this project, you will design and implement a Sudoku Solver - a Java program that automatically finds solutions for Sudoku puzzles using efficient algorithms.

Sudoku Overview

Sudoku is a 9x9 grid-based puzzle where each row, column, and 3x3 subgrid must contain the numbers 1 to 9 without repetition. The solver should take an incomplete Sudoku board as input and fill in the missing numbers while ensuring all constraints are met.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

(Image source: https://en.wikipedia.org/wiki/Sudoku_solving_algorithms)

There are many approaches to solving Sudoku, which you can explore here:

https://en.wikipedia.org/wiki/Sudoku_solving_algorithms

You are encouraged to research additional resources and apply relevant techniques in this group project. However, you must cite any external resources appropriately.

Implementation Requirements

You must create a class RMIT_Sudoku_Solver with the following method:

```
public class RMIT_Sudoku_Solver {
    public int[][] solve(int[][] puzzle) {
        // Implement your solution here
    }
}
```

The method solve(int[][] puzzle):

- Accepts a 2D integer array (9x9) representing the Sudoku puzzle.
- Each cell contains an integer from 0 to 9:
    - 1-9 => Pre-filled numbers (must remain unchanged in the solution)
    - 0 => Empty cells that your program must solve.
- Returns a 9x9 2D array containing the solved Sudoku puzzle.
- Each cell in the returned array must contain a value from 1 to 9.

If your program cannot find a solution within 2 minutes (on any computer), it must raise an exception.

You may create additional classes and methods needed to organize your code effectively.

You may implement multiple approaches to solve puzzles and compare their performance and accuracy. This can help you analyze which method is more efficient and suitable for different types of Sudoku puzzles.

Examples

You can visit this page to find sample Sudoku puzzles and solutions:

https://sandiway.arizona.edu/sudoku/examples.html

Good luck, and happy coding!

Master Project – 15 Puzzle Solver

In this project, you will design and implement a 15-Puzzle Solver - a Java program that automatically finds solutions for 15-puzzle problems using efficient algorithms.

15-Puzzle Overview

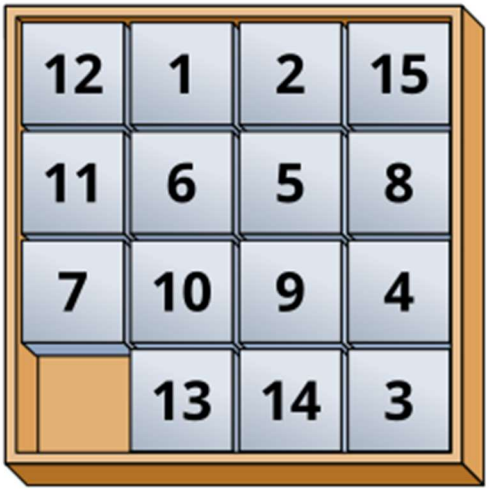
The 15-puzzle is a sliding tile puzzle consisting of 15 square tiles numbered 1 to 15, arranged in a 4x4 grid, with one empty position. Tiles can slide horizontally or vertically into the empty space. The goal is to rearrange the tiles in numerical order (from left to right, top to bottom).

This picture represents the solution of the 15-Puzzle

(https://personal.math.ubc.ca/~cass/courses/m308-02b/projects/grant/fifteen.html)

This picture represents a sample puzzle

(https://en.wikipedia.org/wiki/15_puzzle)



Implementation Requirements

You must create a class RMIT_15_Puzzle_Solver with the following method:

```
public class RMIT_15_Puzzle_Solver {
    public String solve(int[][] puzzle) {
        // Implement your solution here
    }
}
```

The method solve(int[][] puzzle):

- Accepts 4x4 2D integer array representing an instance of the 15-Puzzle. Each cell in the input contains an integer from 0 to 15:
    - 1-15 => Numbered tiles
    - 0 => Empty cell
- Output: A String representing the sequence of moves to solve the puzzle. Each character in the string represents a move:
    - U => Move the tile below the empty cell up
    - D => Move the tile above the empty cell down
    - L => Move the tile right of the empty cell left
    - R => Move the tile left of the empty cell right
- Executing these moves in order must solve the given puzzle. If the String length is greater than 1,000,000 (one million), raise an exception

For example, if the input puzzle is

```
[
  [1, 2, 3, 4],
  [5, 6, 7, 8],
  [9, 10, 11, 12],
  [13, 0, 14, 15]
]
```

The solve() method must return “LL” to solve this instance. This means moving tile 14 left and then moving tile 15 left will solve the puzzle.

You may create additional classes and methods needed to organize your code effectively.

You may implement multiple approaches to solve puzzles and compare their performance and accuracy. This will help analyze efficiency and determine the most suitable method for different types of 15-puzzles.

Good luck, and happy coding!