Koç University
COMP202
Data Structures and Algorithms
Assignment 6

Due: 04 June 2023

This assignment will involve the implementation of Graph data structures and graph traversal algorithms, as well as an application of these algorithms.

This assignment must be completed individually. While discussing general ideas and approaches is allowed, any sharing of code is explicitly prohibited, and will be considered plagiarism. Additionally, any use of AI tools such as ChatGPT is prohibited as well.

1 Introduction

In this assignment, you will implement the following:

- A graph data structure.
- Multiple commonly used graph traversal algorithms.
- An application of a graph traversal algorithm to solving the game Sudoku.

The files are commented to include more details about the implementation. Additionally, you can refer to the pseudocode in the lecture slides, which will be very helpful when writing the code.

2 Graph Data Structure (40 pts)

In this part of the assignment, you will need to develop a graph data structure, which represents a directed, weighted graph. We will then ask you to write wrappers around this data structure, to handle two other types of graphs: undirected and weighted, directed and unweighted. While we don't require you to implement the unweighted, undirected case, this can easily be done by combining the ideas for the other two. You will need to modify the bolded files below. The following files are provided for you:

- *iGraph.java*: This is an interface defining the graph ADT. Make sure all the necessary methods are implemented in your concrete implementation, and pay attention to all of the comments provided in this file.
- **DirectedWeightedGraph.java**: Most of your graph implementation will be in this file. You will implement all the necessary methods for a directed, weighted graph.
- **DirectedUnweightedGraph.java**: Implement a directed, unweighted graph by extending the directed, weighted graph, and using all weights to be equal (such as 1).
- **UndirectedWeightedGraph.java**: Implement an undirected, weighted graph by extending the directed, weighted graph, and adding paths going each direction.

3 Graph Algorithms (40 pts)

This part of the assignment requires you to implement a few common graph algorithms: Depth-First Search (DFS), Dijkstra's single-source, all-destinations shortest path (aka smallest cost) algorithm, and the directed, cycle-finding algorithm.

- **GraphAlgorithms.java**: Implement the provided algorithms. You may add any needed methods and fields, but cannot modify what is provided.
- *AlgTesting.java*: Tests the graph algorithms for the autograder. This can be run independently from the rest of the autograder.

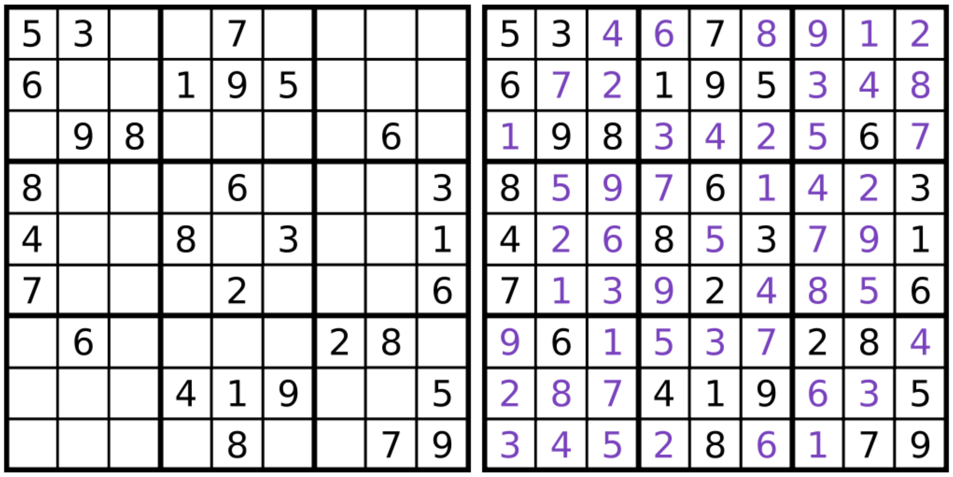
4 Application: Solving Sudoku (20 pts)

In this final part of the homework, you will use your knowledge of graph algorithms to solve Sudoku puzzles.

Sudoku is a logic game in which there is a 9x9 grid of squares, with some initially filled with numbers. The goal is to find the unique arrangement of numbers such that each row, each column, and each of the 9 different 3x3 squares contains all the numbers between 1 and 9 exactly once. You can see Figure 1 for an example.

To solve this using graph algorithms, we can implicitly represent each valid board combination as a vertex of a graph, and have a (directed) edge connecting two boards if we can get from the first board to the second board by adding a single valid number (by valid number, we mean that the inclusion of this number at a given location doesn't violate any of the constraints mentioned above).

There are three important files for this part:



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

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

Figure 1: A player begins with the left board, where many of the squares are left blank. The goal is to produce the board on the right, where you can see that no two of the same number are in the same row, column, or 3x3 grids.
Src: https://blogs.unimelb.edu.au/sciencecommunication/files/2016/10/sudoku-p14bi6.png

- **Sudoku.java**: This contains the implementation of the Sudoku board, as well as some helper functions that will be useful for the implementation of the search. You will be required to implement some of these helper functions, checking for the validity of certain moves. You may add additional fields/methods as necessary.
- **SudokuSolver.java**: Here, you will need to implement the algorithm that solves the provided Sudoku board. There is only a single method in the provided file, but you may add additional fields/methods as necessary.
- *SudokuTesting.java*: Tests the sudoku code using the autograder. Do not modify this file. You can run this independently of the other parts of the homework.

5 Grading

As with previous assignments, you will be graded with an autograder, which is included in the files. Do not change variable or method names unless told that you are allowed to, as this can cause issues with the grading process.

The autograder can be a very useful tool for debugging and understanding problems in your code. If you have trouble with this, refer to the video provided on Blackboard.

Note that the autograder producing 100% for you does not mean your code is running perfectly. We will be running additional checks that are not in the version of the autograder provided to you.

6 Submission

You will need to submit a zip file containing your solution to Blackboard, as well as uploading all your code to github classroom. **Make sure to do both submissions, so there is no issues with the grading process**.

6.1 Submission Instructions

- Make sure that the file structure and names match the originally provided ones. Otherwise, the autograder will not be able to properly grade your code.
- After submitting on Blackboard, download your submission and make sure it is correct.
- Make sure that the submitted code compiles. If it does not compile, you will receive a 0 for this assignment.
- Do not submit code that does not terminate, or blows up the memory. This will result in a 0 for this assignment.

7 Additional Hints

- If a specific part of the code is not working properly, you may find it useful to write your own main class and run it with many print statements.
- You may use data structures such as Stacks, Queues, and Lists, that are provided by the Java language.
- All the information you need to implement the data structure and algorithms will be found in the lecture notes, the textbook, and in the comments of the code.