Introduction

Coursework 1 studies a Matlab program which is adapted from MathWorks. The question is based on your understanding of A* algorithm and extends your skills to develop well-known tree search techniques to solve the maze problem. The primary goal is to give you first-hand experience in implementing search techniques for solving the shortest route problem in Matlab, one of the most adapted high-level languages and analytic tools in business and research.

Marks

Coursework 1 accounts for 15% of the COMP1037 marks. Corresponding marks will be awarded for answering each of the sub-questions in FAIcw1.doc. Report should be written using this file as the template (keeping the same format of margin, font size and spacing, etc.).

Plagiarism vs. Group Discussions

As you should know, there is **no tolerance of plagiarism** and any breach of which will be dealt with according to the University's standard policies. Please be very careful not to cross the boundary into plagiarism while having general discussions regarding the coursework to promote the generation of new ideas and to enhance the learning experience. The important part is that when you sit down to actually do the work and write the answers, you do it individually. If you do this, and you truly understand what you have written, you will not be guilty of plagiarism. Do NOT, under any circumstances, share code or share figures, graphs or charts, etc.

Deadline and submission procedure

The submission deadline is 12pm on the 4th April 2019 via Moodle. Late submission results into a 5% reduction of your coursework mark for each weekday. Any work handed in after the 11th April will receive zero marks.

Name your submission file: ***FAIcw1-XXX.zip*** (incl. 1. your report '***FAIcw1-XXX.pdf***', 2. your ***Astar-MazeSolver-XXX***' code folder, 3. Your ***Greedy-MazeSolver-XXX***' code folder 4. your ***DFS-MazeSolver-XXX***' code folder), where XXX should be your student ID number, and submit a single compressed file via Moodle.

Your code needs to be runnable by calling the function with name '**AStarMazeSolver**', '**GreedyMazeSolver**' and '**DFSMazeSolver**' under each corresponding code folder, the input of function is a 'maze' generated by the maze generator. Any code cannot be successfully called by the above function name will be given zero marks.

If you can't submit your coursework on time due to Extenuating Circumstances, please contact your personal tutor first. I am only granting an extension of submission based on his / her recommendation.

Please remove the above text while writing your coursework report.

This part is based on the maze generator demo (MazeGeneration-master). The maze generator is a project written by some student using Matlab. He has adopted a tree search approach to randomly generate a maze with user-defined size and difficulty. ***(15 marks)***

(a) Read the MazeGeneration-master code, identify which line(s) of code is used to implement the tree search approach, explain the logic and the data structure used by the student to implement the tree search. (3 marks)

(b) Identify the logic problem of this maze generator if there is any. (1 marks)

(c) Write a maze solver using A* algorithm. (5 marks)

i) The solver needs to be called by command '**AStarMazeSolver(maze**)' within the Matlab command window, with the assumption that the 'maze' has already been generated by the maze generator.
ii) In the report, show what changes you have made and explain why you make these changes. You can use a screenshot to demonstrate your code verification.
iii) The maze solver should be able to solve any maze generated by the maze generator
iv) Your code need display all the routes that A* has processed with RED color.
v) Your maze solver should be about to display the final solution with BLACK color.

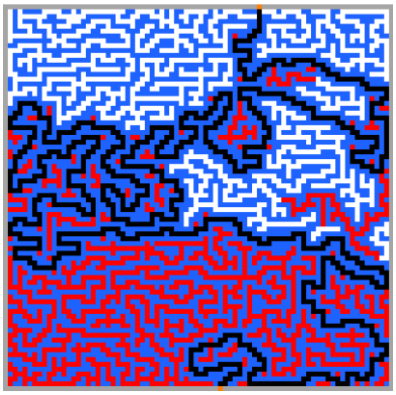

Figure 1. Sample output

(d) Based on the previous AStarMazeSolver, implement a maze solver using the DFS algorithm. Similar to question (c), the solver need by called and ran by command '**DFSMazeSolver(maze)**'. (3 marks)

(e) Based on the previous AStarMazeSolver, implement a maze solver using the Greedy search algorithm. Similar to question (c), the solver need by called by command '**GreedyMazeSolver(maze)**'. (1 marks)

(f) You are required to use the Matlab basics from the first lab session to show the evaluation results of the three searching methods you've implemented in (c), (d) and (e) (hint: bar/plot) with respect to the '**total path cost**', '**number of nodes discovered**' and '**number of nodes expanded**'. Explain how you can extract the related information from data stored in variable '**QUEUE'.** (2 marks).