COMP304

Assignment One

Divyan Hirasen
215018696

Heuristic evaluation function used for A*

The heuristic used for the A* search algorithm is as follows:

$F(n) = G(n) + H(n)$

$G(n)$: The number of tiles out of place of state n compared to the goal state.

$H(n)$: The current depth of the state in the tree, i.e. its distance from its initial state.

Using a priority queue and implementing a comparator for it, the compare method will then order the untested states by the sum of these functions, and thus, the next state to be visited will always be the state with the least value for the sum of these functions.

This process will always yield the correct solution in the least number of moves given that the initial and goal states are solvable.

Performance Comparisons

EASY			
Characteristics	**DFS**	**BFS**	**A***
Solution Path Length	13	5	5
States Visited	14	43	7
Execution Time	0.0	0.0	0.0

MEDIUM			
Characteristics	**DFS**	**BFS**	**A***
Solution Path Length	57835	9	9
States Visited	68856	331	74
Execution Time	139.002	0.031	0.016

HARD			
Characteristics	**DFS**	**BFS**	**A***
Solution Path Length	44560	12	12
States Visited	49146	1607	377
Execution Time	74.085	0.171	0.094

Review

<u>Solution path length</u>

The BFS and A* algorithms both provided the perfect and least most moves required to solve all 3 difficulties of puzzles, while the DFS algorithm provided a solution of 8 additional steps so solve the easy, 57 862 additional steps to solve the medium and 44 548 additional steps to solve the hard puzzle. **Thus in terms of solution path length, BFS is equal to A*, while DFS is the worst performing.**

<u>States visited</u>

A* algorithm had visited the least states compared to DFS and BFS and still provided a perfect least number of moves solution. BFS was the next best which also provided a perfect least number of moves solution but had visited an extra 36, 257 and 1230 states respectively for the above difficulties. Lastly was DFS which was the least efficient which visited an additional 7, 68 782, 48 769 states compared to A*, and also provided much longer solution paths lengths. **Thus in terms of states visited, A* is the most efficient, then BFS and lastly DFS.**

<u>Execution time</u>

For the easy difficulty all searches solved a path in under a second. For medium difficulty A* had solved it 0.015s faster than BFS, and 138.986s faster than DFS. A similar order was obtained from the hard difficulty. **Thus in terms of execution time, A* was the most efficient, then BFS and lastly DFS.**

COMP304: Artificial Intelligence
Assignment One
Due Date: 18 August 2017

8-Puzzle Problem

Given an initial configuration and a goal configuration, the 8-puzzle problem involves re-arranging the tiles in an initial configuration to obtain a goal configuration. A legal move is moving any tile into the blank space except that a tile may not move diagonally into the blank space i.e. a tile may be moved up, down, left or right into the blank space. Examples of initial and goal configurations for 8-puzzle problems of varying difficulty are illustrated below:

Easy

Initial State

1	3	4
8	6	2
7		5

Goal State

1	2	3
8		4
7	6	5

Medium

Initial State

2	8	1
	4	3
7	6	5

Goal State

1	2	3
8		4
7	6	5

Hard

Initial State

2	8	1
4	6	3
	7	5

Goal State

1	2	3
8		4
7	6	5

1. Write a program that (35 marks):
   (a) Implements the depth-first search, breadth-first search and A* algorithm.
   (b) Allows a user to enter an initial and goal state for an instance of the 8-puzzle problem. The user should be allowed to enter a string representing a state for e.g. 281463b75 to represent the initial state of the Hard puzzle in the figure above.
   (c) Provides the user with the option of solving the problem using either the depth-first search, breadth-first search or A* algorithm.
   (d) Solves the instance of the 8-puzzle problem using the chosen search.
   (e) Outputs the solution path.
2. Submit a report describing (15 marks):
   (a) The heuristic evaluation function used for the A* algorithm.
   (b) A comparison of the performance of the three search methods in solving 8-puzzle problems of differing difficulty.

Notes:

- Use either Java or C++ to implement the program.
- Submit both the source code and you must submit executable programs that run without the IDE being installed on the user's system.
  - Java programs: Submit a jar file or the class files that will run. Ensure that the jar/class files can be run on a machine with only the JDK installed (i.e. without the IDE that you have used to create the program).
  - C++ programs: Ensure that you compile the program to run on machines that do not have C++.
- The interface can be text-based or graphical.
- Programs that do not run will be allocated a mark of zero.

Submission

- The assignment **must** be submitted on or before 18 August 2017.
- You must use the Course website to submit. Click on **Assignments** in the Activities block (top left). Then click on **Assignment One**. You will be taken to a page which allows you to upload a file. You can re-upload a file but this will overwrite any file that was previously uploaded.
- Please be warned against plagiarism. This is an individual assignment and group work is **not** permitted. The school has access to software to check for plagiarism. Cases of suspected plagiarism will be submitted to the University proctor.