Programming Assignment 1
Questions
Question 1

Implement the A* algorithm and the Eight-Puzzle in a programming language of your choice.

The most challenging part will probably be the queue of fringe nodes (ordered by f-value). For the Eight-Puzzle an array of 32 queues for the different possible f-values will do. A more general solution would be a priority queue implemented as a binary tree. Some programming languages have an appropriate data structure in their standard library.

The state of the puzzle can be represented by a simple array of numbers that lists the tiles in the puzzle row by row. The number zero can be used for the empty position. For example, the goal state can be represented by the array [0 1 2 3 4 5 6 7 8].

Test your implementation with the initial state [1 6 4 8 7 0 3 2 5]. What is the optimal solution path length?

Marks available: 2
Question 2

Now extend your implementation of A* to turn it into a graph search algorithm.

To make the implementation reasonably efficient, the set of all nodes can be put into a hash table. A good hash function will make a difference here, so keep that in mind.

Test your implementation with the initial state [8 1 7 4 5 6 2 0 3]. What is the optimal solution path length?

Marks available: 2
Question 3

How many different states of the Eight-Puzzle are exactly 27 steps from the solution, where the distance to the solution is the length of the shortest path?

Hint: You can generate the search space breadth-first, using a hash table to test for repeated states, and starting from the goal state. This works because all the actions in the search space are reversible. While you enumerate the states in this way, simply count the ones at depth 27.

Marks available: 2
Question 4

An essential component of the forward state-space search planning algorithm is the code that identifies the actions (ground instances of operators) that are applicable in a given state. Implement a function that performs this task.

Depending on the programming language you are using, you may want to implement a PDDL parser for planning domains and problems, or you can just hard-wire the components into your code. The choice is yours.

Test your code on the random domain and problem. How many applicable actions are there in the initial state?

Marks available: 2
Question 5

An essential component of the ground backward search planning algorithm is the code that identifies the actions (ground instances of operators) that are relevant for a given goal. Implement a function that performs this task.

Test your code on the random domain and problem (same as before). How many relevant actions are there for the given goal?

Marks available: 2