Artificial Intelligence CA

8-Puzzle Game

Question 1.1

The 8-puzzle game works by allowing the non-empty tile to move into the empty tile space. The aim of this game is to achieve the given goal state.

Here are the following rules that needs to be consider when solving the game:

- **States:** describes specific location of each of the tiles, and the empty tile in one of nine squares.
- **Initial state:** designing any state as an initial state and any given goal can be reached from exactly half of the possible initial states.
- **Actions:** defining the action as a movement of the blank space such as up, right, left and down and different subsets of these are possible depending on where the empty tile is located.
- **Transition model:** giving a state and action to return the resulting state.
- **Goal test:** checks whether the start state matches the goal state.
- **Path cost:** each step costs 1 and the path cost is the number of steps taken to solve the puzzle.

The action is abstracted to the puzzle's start state and goal state, ignoring the intermediate locations where the block is sliding. For example, shaking the board when pieces get stuck and ruled out extracting the pieces and putting them back again. Also, the empty title can't move diagonally and can take only one step at a time by the Breadth-first search. This example is cheaper but it's an incomplete search.

Question 1.2

Question 1.2.1

The A* search algorithm calculates the shortest path by comparing estimated cost from node n to the goal node using heuristic function h(n). Also, the cost to reach the node n from the start node g(n). This search algorithm expands less search tree and provide optimal results faster. As a result, A* algorithm computes by picking the node according to a f value, which is equal to the sum of g and h. At each step the algorithm selects a node having the lowest cost f and process that node.

The cost is calculated by using f(n) = g(n) + h(n).

- f(n) = estimated cost of the cheapest solution.
- g(n) = cost to reach next node n from the start state.
- h(n) = cost to reach goal node from current node.

Question 1.2.2

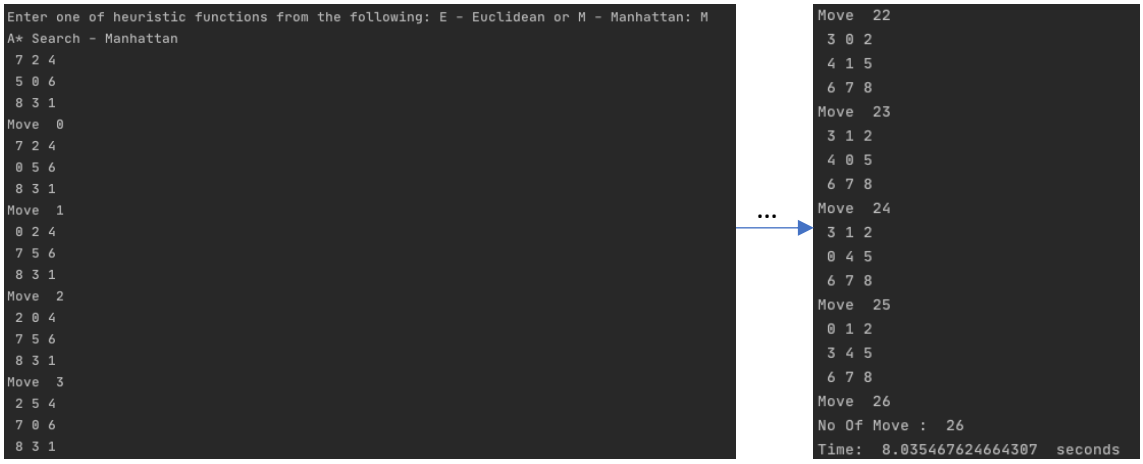
One of the admissible heuristic function for the 8-puzzle problem is the Euclidean method, which takes the distance between two points and measures the length of a segment connecting the two points. This function done by taking the square root of the sum of absolute values of differences in the current positions (x and y) squared and its goal positions (x and y) squared respectively. Also, Euclidian is an admissible heuristic, because in every move one tile will only move closer to its goal by one step and the Euclidian distance is not greater than the number of steps required to move a tile to its goal position.

Another admissible heuristic function is the Manhattan method, which takes the distance between two points and measured along x and y axes. This function works by finding the sum of absolute values of differences in the current position (x and y) and its goal position (x and y) respectively. Also, Manhattan is an admissible heuristic because in every move, one tile can only move closer to its goal by one step.

Question 1.2.3

The 8-puzzle game program is located in the module file called eight_puzzle_game.py.

For the program to compute the algorithm as intended the user must input 0 to represent the empty tile in the 8-puzzle game. For the option of using the one of the heuristic functions, the user must input E for Euclidean or M for Manhattan.



Question 1.2.4

While running computing the A* algorithm for the 8-puzzle game I have noticed that the Manhattan method is quicker than the Euclidian method. This is because Manhattan method takes around 8 seconds to solve, whereas Euclidian method takes around 30 seconds when running on my machine. As a result of Euclidean method allows to find all possible true values as it is less than or equal to true value at every node and still become an admissible, but Manhattan method allows the tile to move to any adjacent square to an optimal solution.

Question 1.3

The 8-puzzle game program is located in the module file called eight_puzzle_game.py.

I have created a generic version of the 8-puzzle game where the user can input the start state, goal state and chooses one of the heuristic functions from the given options.

```
Enter your start state

Enter the values and 0 value as an empty tile.
[1,1]: 7
Enter the values and 0 value as an empty tile.
[1,2]: 2
Enter the values and 0 value as an empty tile.
[1,3]: 4
Enter the values and 0 value as an empty tile.
[2,1]: 5
Enter the values and 0 value as an empty tile.
[2,2]: 0
Enter the values and 0 value as an empty tile.
[2,3]: 6
Enter the values and 0 value as an empty tile.
[3,1]: 8
Enter the values and 0 value as an empty tile.
[3,2]: 3
Enter the values and 0 value as an empty tile.
[3,3]: 1
Start Board:  [[7, 2, 4], [5, 0, 6], [8, 3, 1]]
```

```
Enter your goal state

Enter the values and 0 value as an empty tile.
[1,1]: 0
Enter the values and 0 value as an empty tile.
[1,2]: 1
Enter the values and 0 value as an empty tile.
[1,3]: 2
Enter the values and 0 value as an empty tile.
[2,1]: 3
Enter the values and 0 value as an empty tile.
[2,2]: 4
Enter the values and 0 value as an empty tile.
[2,3]: 5
Enter the values and 0 value as an empty tile.
[3,1]: 6
Enter the values and 0 value as an empty tile.
[3,2]: 7
Enter the values and 0 value as an empty tile.
[3,3]: 8
Goal Board:  [[0, 1, 2], [3, 4, 5], [6, 7, 8]]
```