United International University
Department of Computer Science and Engineering
CSI 342: Artificial Intelligence Laboratory
Lab 02: A* Search
Fall 2015

Today we will solve the same problem, that we solved in the last class. However, today we will use a different algorithm called 'A* search'. In stead of a queue frontier it uses a frontier that acts like a priority queue. The key of this priority queue is the summation of the cost function $g(x)$ and the heuristic function $h(x)$. First, we have to implement these two functions. Lets have a recap of the problem.

We are playing the 8-puzzle game today. Consider the following: In a 8-puzzle game we start with an arbitrary node and set to find the actions that are needed to find the following goal state:

0	1	2
3	4	5
6	7	8

Suppose our initial node is something like the following:

1	2	5
3	7	4
6	8	0

Allowable actions are up,down, left and right and only the tile with zero can move to an adjacent place. Our actions are already implemented and goal checking is also done.

1 Task 1: Cost Function $g(x)$

This cost is the actual cost from the start node to the current node x. Suppose, you are given any state with the steps that are required to generate the state from the initial state or start state. Then the cost is simply the sum of the cost of the individual costs of each step. In our case we assume that each action (right,left,up, down costs us the same). First, function that we need to write is the cost function that returns the cost to reach the state from the initial node. Lets write it in pathCost.m file.

2 Heuristic Function $h(x)$

Now that you have implemented the cost function its time to implement the heuristic function for any given node. In todays, class, we will use the Manhattan distance as the heuristic function. Manhattan distance is simply the sum of the individual Manhattan distances. For any particular piece the Manhattan distance is the moves needed up-down or left-right to reach the position that the piece holds in the goal state. Write your own code for Manhattan distance in the manhattanDistance.m file.

3 Task 3: A* Search

A node in the A* tree will have two elements. Firstly, the state of the puzzle and secondly, the action sequence. MATLAB allows us to create structures that can handle such nodes. We have already seen this in the last class. However today, we also need the cost function to be added to this states. The inital node does not contain any steps, but has got a state and the value of the cost function. Here is how it can be created:

```
field1='state';
field2='steps';
```

```
field3='cost';
value1=[1 3 6 2 7 8 5 4 0];
value2=[];
value3=0;
initialNode=struct(field1,value1,field2,value2,field3,pathCostGx(stateInitial)+manhattanDistance(stateIn
```

Now, we can easily add this initialNode to a frontier.

```
frontier=[initialNode];
```

Now, A* runs until the frontier is empty or the goal is found! We need a while loop for that. Each time check if the size of the frontier is 0 or not and extract a node from the queue. Now, check if that node contains the goal. If goal check then just print the steps and that is your solution. Else, expand the node using the actions and add them into the frontier too!

Another thing, you need to do is to maintain a closed List. It will contain all the visited states and thus states in this list will not be added to the frontier.

Use astar.m to finish your code.

Beware, the frontier is a priority queue. Each time you need to sort it before removing the first element.

Answer for this particular initial solution is:

[3 0 2 0 3 3]

Here, 0 = UP, 1= DOWN, 2= RIGHT, 3=LEFT

4 Task 4: Comparison

1. Try to find the solution of the problem with initial node [0 1 3 7 8 6 2 5 4] using both bfs and A* search. See how much time they take and how many nodes they explore, in terms of frontier and closed list.