**HW1**

**Reinforcement Learning**

**044203**

May 2023

Ayala Luz, 318298452

Gil Kapel, 305323776

1. Solving 8-Puzzle with Dijkstra’s Algorithm:
   1. Denote the size of a board , we would have different states for a single puzzle (for a board of 3x3 we have 9! Options (362,880) (– 9 places to locate the hole (‘0’), then 8 options to locate ‘1’ and so on. Because the search graph would be enormous, we would run Dijkstra algorithm one step at a time, build the next nodes from the current node’s possible actions and finish building when we get to the goal state (we can use BFS algo to ensure optimality)
   2. Code
2. Solving 8-Puzzle with A\* Algorithm:
   1. We can use the Manhattan distance between two states as a heuristic function – by flattening each state to a vector and find the sum of Manhattan distances between each cell. This function is admissible because in the best situation the shortest path is exactly the sum of Manhattan distances between each cell, and in other situations the Manhattan distance is smaller (if we must use a circulate path to get to the destination) 🡪
   2. Code
   3. ***2,931*** nodes were expanded using ***incorrect tiles count heuristic*** vs ***351*** nodes in ***Manhattan distance heuristic***. The *incorrect tiles heuristic* is admissible because as before, it would take at least n steps to go if n tiles are incorrect, but most of the times it would take a longer path  
       🡪 .
   4. ***Code*** – Dijkstra took 15 second vs A\* that takes 0.3 second – both with the same solution ‘ddrrullurrdlulddrruulldrr’
   5. Heuristic function analysis:
      1. For we get that the heuristic function doesn’t contribute to the algorithm, and we collapse to the Dijkstra algorithm.   
         For we get the exact definition of A\* demonstrated in the lecture.  
         If the algorithm won’t use the real distance between two nodes   
           
         at all, this could accelerate the execution time (if there are obstacles on the graph, it could expand unnecessary nodes) but make the search sub-optimal.

In general, for we give more weight to the heuristic values, that could accelerate the time to solve the problem with cost of harming the optimality.

* + 1. Comparison:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Path length | 25 | 25 | 25 | 45 |
| Plan time | 23.2 seconds | 0.3 seconds | 0.05  seconds | 0.01 seconds |
| Explored nodes | 166,159 | 3,589 | 769 | 242 |

The results demonstrate what we explained in section 5.1 – is optimal and fast, is optimal but slow and  is sub-optimal and fast.

We added a test for and as it appears, for this specific case the algorithm found the best path one scale faster than the regular A\*.