The 8 puzzle project Report

In this project we are solving the 8-puzzle by using the 3 following search algorithms:

1. Uniform Cost Search
2. A* with the Misplaced Tile heuristic.
3. A* with the Euclidean Distance heuristic.

Test cases are used: (trivial case is same as our goal state)

```
trival_case    very_easy_case  easy_case    easy_case2    doable_case  doable_case2  oh_boy_case  impossible_case
[1  2  3]        [1  2  3]     [1  2  0]    [4  1  2]     [0  1  2]    [3  1  2]     [8  7  1]    [1  2  3]
[4  5  6]        [4  5  6]     [4  5  3]    [7  5  0]     [4  5  3]    [4  5  0]     [6  0  2]    [4  5  6]
[7  8  0]        [7  0  8]     [7  8  6]    [8  6  3]     [7  8  6]    [7  8  6]     [5  4  3]    [8  7  0]
```

My program is successfully finding the goal state of the 7 out of the 8 test cases. All the 3 search algorithms failed to find the goal state for the “impossble_case.” In this test case, the program was terminal at depth 31 or 30, and all had expended 181440 nodes.

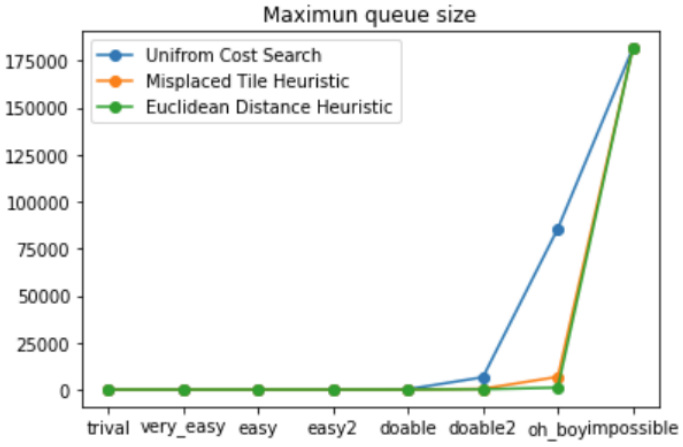
depth of goal state

	Unifrom Cost Search	Misplaced Tile Heuristic	Euclidean Distance Heuristic
trival	0	0	0
very_easy	1	1	1
easy	2	2	2
easy2	4	4	4
doable	9	9	9
doable2	15	15	15
oh_boy	22	22	22
impossible	31	31	30

According to the result of the depth of the goal state table, all cases result in the same result of the depth of the goal state. Except for the impossible case, the euclidean distance was terminal at different depths. According to the table and graph below, I observed that uniforms cost search is taken up most of the space of complexity than the other two.

Max queue size

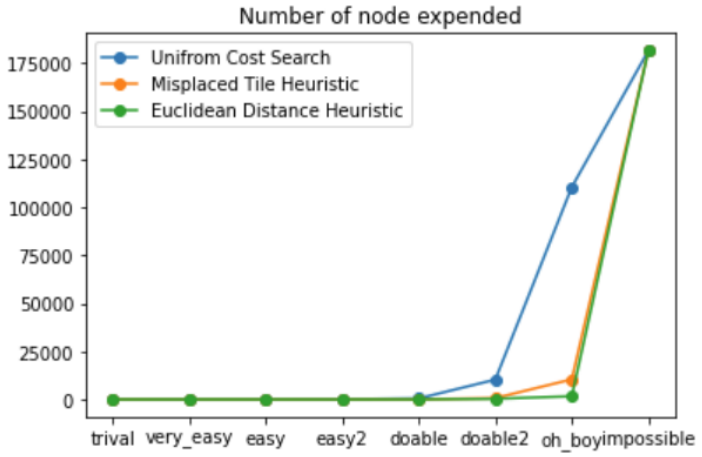
	Unifrom Cost Search	Misplaced Tile Heuristic	Euclidean Distance Heuristic
trival	1	1	1
very_easy	3	2	2
easy	4	3	3
easy2	29	5	5
doable	196	13	10
doable2	6550	420	176
oh_boy	85639	6758	1037
impossible	181440	181440	181440



According to the table and the graph of the maximum queue at any one time, the uniform cost search is obvious to use more memory than the other two. Because it had the largest amount of expanded nodes compared to the other two. In simple cases, both misplaced tiles and Euclidean distance heuristics may cost memory of the same size. But for the hardest case, the Euclidean distance heuristic uses less memory, it uses a smaller queue size. This means the better heuristic helps us to search for the goal state more efficiently.

Number of Node expand

	Unifrom Cost Search	Misplaced Tile Heuristic	Euclidean Distance Heuristic
trival	0	0	0
very_easy	6	3	3
easy	6	4	4
easy2	45	7	7
doable	500	21	15
doable2	10275	659	286
oh_boy	110520	10424	1625
impossible	181439	181439	181439



Based on the table and graph of the number of nodes already expanded, the uniform cost search obviously expanded more nodes. This will cause the uniform cost search to spend a lot of time searching for the goal state or a specific state. According to a large difference in the quantity of node expansion. In contrast to the other two, when I running my program I can strongly feel the slowness of the uniform cost search. overall, all test cases of uniform cost search were taking the longest time to search the goal state.

For the node expand and max queue, the line graphs are merged to one point and the table shows the expanded same number of nodes or max queue size in the “impossible case” section. This is because our program failed to search for the goal state and terminated it. Also, this is the case that consumes the largest amount of time for debugging, waiting, and testing the program

In last, the best search algorithm for solving this 8-puzzle problem of these 3 is the A* with Euclidean Distance Heuristic. Because it uses the smallest size of the queue and expends a less number of nodes. Consequently, this algorithm would take less amount of memory and time to search the goal state. (it has less time and space complexity). It’s ok to use the Misplaced Tile Heuristic because it would use slightly more memory and takes more time than Euclidean Distance Heuristic. Owing to the largest amount of memory usage and time, the uniform cost search would not be recommended use to solve the 8 puzzles. Additionally, I implemented another heuristic for Manhattan distance that appeared in our lectures. It uses a slightly smaller queue and expends fewer nodes than the Euclidean distance, but it has the same problem it is terminal and fails to find the goal state for the “impossible case.”