Project #1 Report

I faced several challenges when doing project #1. Some of the challenges that I faced during this project were implementing my frontier so that it was sorted in order of costs. In python, it was difficult to use the priorityQueue and I was running into several issues with it, so I had to use a list and sort it every time in order for the total costs of the nodes to be in order. Additionally, I had to account for the visited nodes and make sure to check for them in the right place, because if I didn't include a visited list, my program would run in an infinite loop. Lastly, I had to make sure that I had all possible operators depending on where the blank space was which took a lot of time and effort to implement, since it was an exhaustive list. Lastly, for the Oh boy problem, I had to use a different puzzle to demonstrate the change in number of nodes expanded and max queue size, since it took a long time to run for the uniform cost search and misplaced tile heuristic, since their heuristic was not as strong as the euclidean distance.

The design of my program was broken down into different functions and a node class. The node class contained the g(n) and h(n) costs, as well as the puzzle for each state. I had functions to retrieve the type of puzzle the user wanted to enter, as well as the type of heuristic they wanted to use. I had an expansion function for the different possible operations that would then be added to a node, which would be appended to the frontier if it was not visited. I also had 3 heuristic functions that would calculate the heuristic for a given puzzle and set its heuristic cost for that puzzle in its node before it was appended to the frontier. Lastly, in my main function, I implemented my graph search algorithm that would go through the frontier and pop the node with the smallest class which would check to see if it was in the goal state. If not, it would expand that node and add those nodes to the frontier. There was also a visitor list that would append expanded nodes to ensure that the same puzzle would not be expanded in the frontier again.

I tried to optimize my code through having a priority queue structure for the frontier. However, because of issues and errors, I had to settle with a list that would be sorted upon every iteration of the while loop. I used a graph tree search in order to keep account for previous nodes that were visited. I was able to optimize the code through utilizing a visitor list that would prevent the program from running in an infinite loop. It would detect identical puzzles before it was expanded to the frontier.

For the comparisons of the heuristic searches, I used another puzzle instead of the oh boy to compare the 3 search algorithms. This is because the oh boy puzzle took a very long time for the Uniform Cost Search and Misplaced Tile Heuristic algorithm to run since the solution was at depth 22. So I used another puzzle to save time and space. I put estimates for those 2 heuristic algorithms for the oh boy puzzle based on what I observed. I listed the test case for the medium puzzle below. Additionally, the impossible puzzle was never able to find a solution.

Number of Nodes Expanded

	Uniform Cost Search	Misplaced Tile Heuristic	Euclidean Distance Heuristic
Trivial	0	0	0
Very Easy	3	1	1
Easy	4	2	2
Doable	27	4	4
Oh Boy	Around 4^22	Around 5,000,000	974
Medium*	308	22	11

Medium Puzzle:
1 8 2
0 4 3
7 6 5

Max Queue Size

	Uniform Cost Search	Misplaced Tile Heuristic	Euclidean Distance Heuristic
Trivial	1	1	1
Very Easy	6	3	3
Easy	7	4	4
Doable	20	5	5
Oh Boy	Around 25,000	Around 6,500	570
Medium*	213	19	13

Medium Puzzle:
1 8 2
0 4 3
7 6 5

As the complexity increases, the Euclidean Distance Heuristic has the smallest max queue size and the number of nodes expanded. The Misplaced Tile Heuristic has the second smallest max queue size and the number of nodes expanded. The Uniform Cost Search has the largest max queue size and the number of nodes expanded.

To Run the program you type python3 ProjectPuzzle.py in the terminal. Python3 is used for running the python program.