Assignment #1 Report

Challenges:

- The most significant challenge I faced was when developing the Euclidean Distance. I debugged my program several times until I realized that I was retrieving the pair of each number in the incorrect format. Eventually, I also noticed that my mathematical equation was incorrect. Another obstacle I had to overcome was just the general design and approach to this problem. I had a general understanding of how the solution would look but it took some time before I fully mapped out the object oriented design.
- Another challenge I had to overcome was trying to make the execution time faster. Originally I had utilized a priority queue and a vector to store seen and frontier values. It would take some time for the Misplaced algorithm to run. The Uniform Cost Search algorithm would take forever to run.

Design:

- The structure of this assignment is mainly stemming from the Puzzle object. Within the puzzle object, the initial state, goal state, and type of search algorithm are passed in. Most of the solving with the helper methods are done through this class. It contains several accessor methods to access values that are later used in the solution (max size of frontier, total number of nodes expanded, etc). The solving of the puzzle utilizes a graph search (record of all explored nodes).Here are the most important details regarding the implementation of the program:
- `private:`
    - `vector<vector<int>> initialState;`
    - `vector<vector<int>> finalState;`
    - `searchAlgorithm search;`
    - `int maxFrontierSize;`
    - `int expandedNodeCount;`
- `public:`
    - `void solve()`
        - Initializes root node with initialState and then performs the graph search algorithm utilizing frontier (priority queue) and explored vector. Creates two hashmaps to store frontier node states and explored states.
    - `int calculateMisplacedTitles(vector<vector<int>> currentState)`
        - Searches through the entire passed in state and matches each index with the finalState. If the current index is not a zero (blank) then it will increase the misplaced tile count and return at the very end.
    - `int calculateEuclideanDistance(vector<vector<int>> currentState)`
        - Searches through the entire passed in state and finds its equivalent numbers located in the final state in (x, y) pair format. It subtracts the location of the final state (x, y) with the current state’s (x,y) and then add to a calculatedDistance counter which returns at the end.

- `vector<Node *> possibleOperators(Node *current)`
    - `Using the current node, it finds the location of the zero (blank space). It tries to apply the four possible operators (up, down, left, right) depending on the location of zero and returns new generated nodes in a list.`

Optimization:

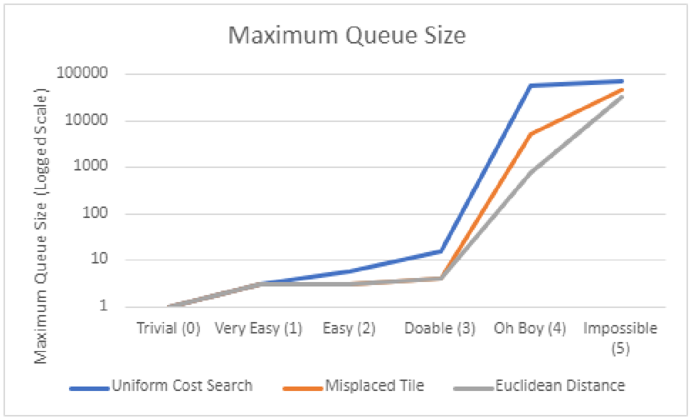
- I optimized most of the functions to pass by reference to avoid creating unnecessary copies to save some time. I also made sure to only add new states that are not in both the frontier or explored list.
- I was able to utilize a hash map (constant lookup time) for both the frontier and explored list. This made the execution time SIGNIFICANTLY faster and I was able to run the Uniform Cost Search algorithm with the Oh Boy puzzle within a few milliseconds.

Comparisons:

- Based on the results of the algorithms, it can be concluded that the algorithms utilizing heuristics are performing significantly better. More specifically the better the heuristic (Euclidean) versus the lower tier (Misplaced), there is a significant difference as the puzzle's difficulty increases. The maximum queue size varies depending on the algorithm and as the puzzle becomes harder the, the better the algorithm the less the maximum queue size is. Similarly, the number of nodes expanded follows a similar pattern. The algorithms that utilize the heuristics and the one that has the better heuristic, the number of nodes expanded is the least. The Euclidean distance outperforms both the other algorithms and misplaced tile comes in second. For problems that are very close to the solution, the algorithm does not make that big of a difference.

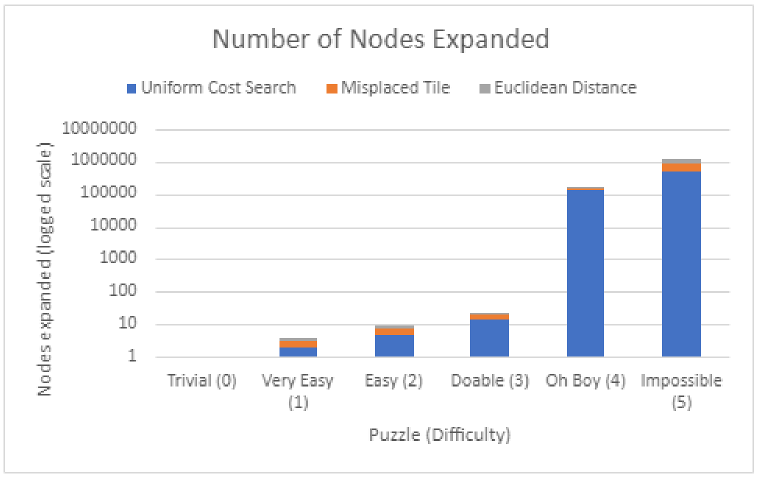
Maximum Queue Size

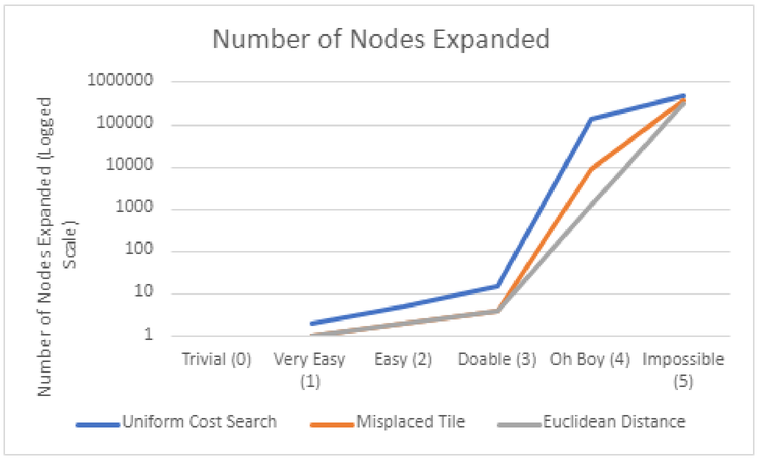
	Uniform Cost Search	Misplaced Tile	Euclidean Distance
Trivial (0)	1	1	1
Very Easy(1)	3	3	3
Easy (2)	6	3	3
Doable (3)	16	4	4
Oh Boy (4)	59805	5408	773
Impossible (5)	71272	48932	33157



Number of Nodes Expanded

	Uniform Cost Search	Misplaced Tile	Euclidean Distance
Trivial (0)	0	0	0
Very Easy(1)	2	1	1
Easy (2)	5	2	2
Doable (3)	15	4	4
Oh Boy (4)	139850	9131	1303
Impossible (5)	50081	382560	303678


Number of Nodes Expanded
Uniform Cost Search
Misplaced Tile
Euclidean Distance
Nodes expanded (logged scale)
Trivial (0)
Very Easy (1)
Easy (2)
Doable (3)
Oh Boy (4)
Impossible (5)
Puzzle (Difficulty)


Number of Nodes Expanded
Number of Nodes Expanded (Logged Scale)
Trivial (0)
Very Easy (1)
Easy (2)
Doable (3)
Oh Boy (4)
Impossible (5)
Uniform Cost Search
Misplaced Tile
Euclidean Distance