Rachel Danover
Rdano001
862210840

GitHub Link: https://github.com/RDanover/AI-Project-1

Challenges:

The biggest challenges I faced was with ensuring the program could be easily altered to work with larger or smaller sliding block puzzles, primarily with the get_possible_moves function and the Manhattan distance calculation. I also misunderstood the heuristic and implemented manhattan distance instead of euclidean distance. I eventually decided to keep both heuristics in just for comparisons.

Design:

I used an object oriented design. My object Puzzle_State has a vector representing the location of all the tiles, and an integer g to represent the depth/cost, and double h to represent the heuristic value. I do not keep track of the parent and child nodes.

Optimizing:

I used the std:sort method to improve the sorting time of the queue.

To improve this program in the future, adding a hashmap to check previously visited states would be much much faster and would bring down the is_duplicate function from O(N) to O(1).

Graph Search:

I used a vector to check previously visited nodes. It's a very very slow choice, but I had difficulty implementing a hash map for the program.

Rachel Danover
Rdano001
862210840

Heuristic Functions:

Uniform cost:

This was the worst performer for all of the solvable puzzles, however ran the fastest for the impossible puzzle.

Misplaced Tile:

This was tied with Euclidean and Manhattan distances until the “oh boy” puzzle where it took significantly longer than either.

Euclidean Distance:

This was the second best heuristic overall it improved the oh boy case drastically compared to uniform cost and misplaced tile

Manhattan Distance:

This was the best performer overall, especially for the “Oh Boy” case where the puzzle was solved almost immediately while euclidean distance took quite a bit longer.

Findings:

For the shallow problems the heuristics made very little difference, however as the problem depth became larger the more detail about the quality of state the heuristic provided (aka a better heuristic)the quicker the puzzle was solved. However for the impossible cases, having the heuristics made the program take longer to run since it had more code to run per each state.

Rachel Danover
Rdano001
862210840

Test data:

Trival

```
1 2 3
4 5 6
7 8 *
```

Easy

```
1 2 *
4 5 3
7 8 6
```

Very Easy

```
1 2 3
4 5 6
7 * 8
```

doable

```
* 1 2
4 5 3
7 8 6
```

Oh Boy

```
8 7 1
6 * 2
5 4 3
```

Trace

```
1 * 3
4 2 6
7 5 8
```

IMPOSSIBLE: The following puzzle is impossible to solve, if you *can* solve it, you have a bug in your code.

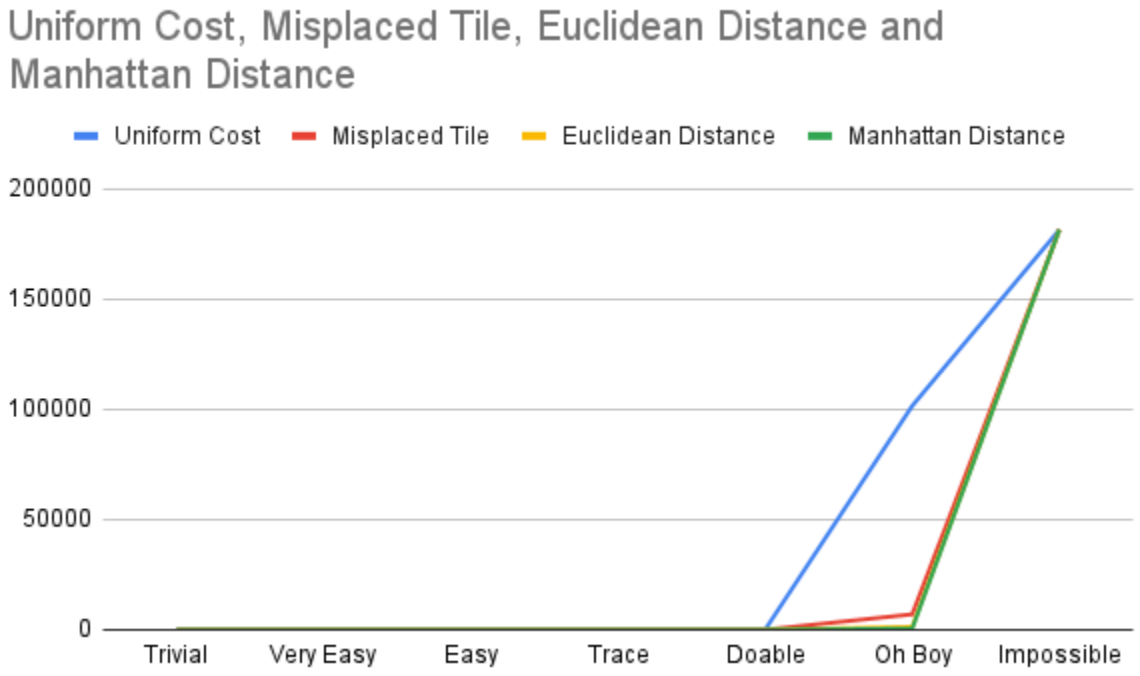
```
1 2 3
4 5 6
8 7 *
```

Rachel Danover
Rdano001
862210840

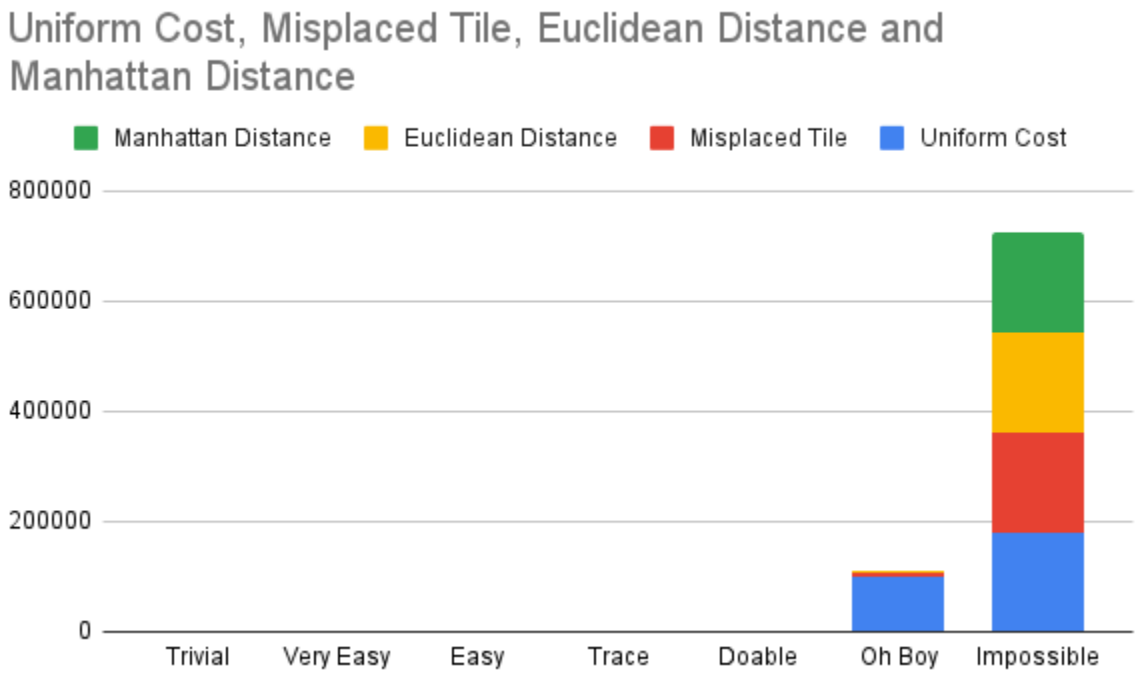
Graphs:

Nodes Expanded

	Trivial	Very Easy	Easy	Trace	Doable	Oh Boy	Impossible
Uniform Cost	0	2	3	9	29	101547	181440
Misplaced Tile	0	1	2	3	4	7012	181440
Euclidean Distance	0	1	2	3	4	1321	181440
Manhattan Distance	0	1	2	3	4	576	181440



Rachel Danover
Rdano001
862210840



Rachel Danover
Rdano001
862210840

Maximum Queue Size

	Trivial	Very Easy	Easy	Trace	Doable	Oh Boy	Impossible
Uniform Cost	0	5	4	10	18	32171	32806
Misplaced Tile	0	3	3	6	4	4038	29856
Euclidean Distance	0	3	3	6	4	842	26685
Manhattan Distance	0	3	3	6	4	382	26805