Project 1
CS170 Introduction to Artificial Intelligence
Niloofar Montazeri
Name: Joshua McIntyre
SID: 862054277
Email: jmcin005@ucr.edu
Name: Edwin Leon
SID: 862132870
Email: eleon029@ucr.edu
April 27, 2021

In completing this assignment we consulted:

- The Project 1 - Eight Puzzle pdf
- The Blind Search lecture slides
- The Heuristic Search lecture slides
- Python3 documentation

Most of the code is original except for some built-in functions and subroutines:

- Heapq: used to implement the priority queue

CS170: Project 1 Write Up

Joshua McIntyre, SID: 862054277

Edwin Leon, SID: 862132870

Introduction:

The following write up is to document our findings in the comparison of three algorithms used to solve the eight puzzle problem. The project goes over Uniform Cost Search, A* with Misplaced Tile Heuristic, and A* with Euclidean Distance Heuristic. To produce and test the algorithms we decided to use Python and the code is provided in a Github repository.

Algorithms:

The project covers three algorithms: Uniform Cost Search, A* with Misplaced Tile Heuristic, and A* with Euclidean Distance Heuristic.

Uniform Cost Search:

The Uniform Cost Search algorithm solely uses g(n) as its heuristic function since h(n) is set to 0. The g(n) is the cost of the path from one node to the root node (ie. the depth of the node). The Uniform Cost Search algorithm will expand to the cheapest node in the frontier (ie. the closest node to the root node). To solve the eight puzzle problem, the Uniform Cost Search algorithm will essentially perform a breadth first search as it expands the nodes until the goal is reached.

A* with Misplaced Tile Heuristic:

The A* an algorithm with a misplaced tile heuristic uses g(n) and h(n) as components for its heuristic function. Similarly, as in the Uniform Cost Search algorithm g(n) is the cost from one node to the root node. The h(n) component is determined by the number of misplaced tiles of the current node without counting the blank tile. The heuristic value (sum of the g(n) and h(n)) will be assigned to each node. The algorithm will determine which node to expand by selecting the node with the smallest heuristic value.

Ex:

Assume the following state/node is the initial state/root node.

[[1, 2, 3],
  [4, 8, 0],
  [7, 6, 5]]

The g(n) = 1 and h(n) = 3 because the cost of the root node to an expanded node whose parent is the root node is 1, and the number of misplaced tiles is 3.

A* with Euclidean Distance Heuristic:

The A* an algorithm with a Euclidean distance heuristic uses g(n) and h(n) as components for its heuristic function. Similarly, as in the Uniform Cost Search algorithm g(n) is the cost from one node to the root node. The h(n) component is determined by the sum of the distances of every misplaced tile (except the blank tile) to its goal state position.The heuristic value (sum of the g(n) and h(n)) will be assigned to each node. The algorithm will determine which node to expand by selecting the node with the smallest heuristic value.

Ex:

Assume the following state/node is the initial state/root node.

[[1, 2, 3],
  [4, 8, 0],
  [7, 6, 5]]

Assume the following state/node is the goal state/node.

[[1, 2, 3],
  [4, 5, 6],
  [7, 8, 0]]

The g(n) = 1 and h(n) = 3.82 because the cost of the root node to an expanded node whose parent is the root node is 1, and the sum of the distance of every misplaced tile to its goal state is 3.82 (1 (8 tile) + 1.41 (5 tile) + 1.41 (6 tile)).

Implementation:

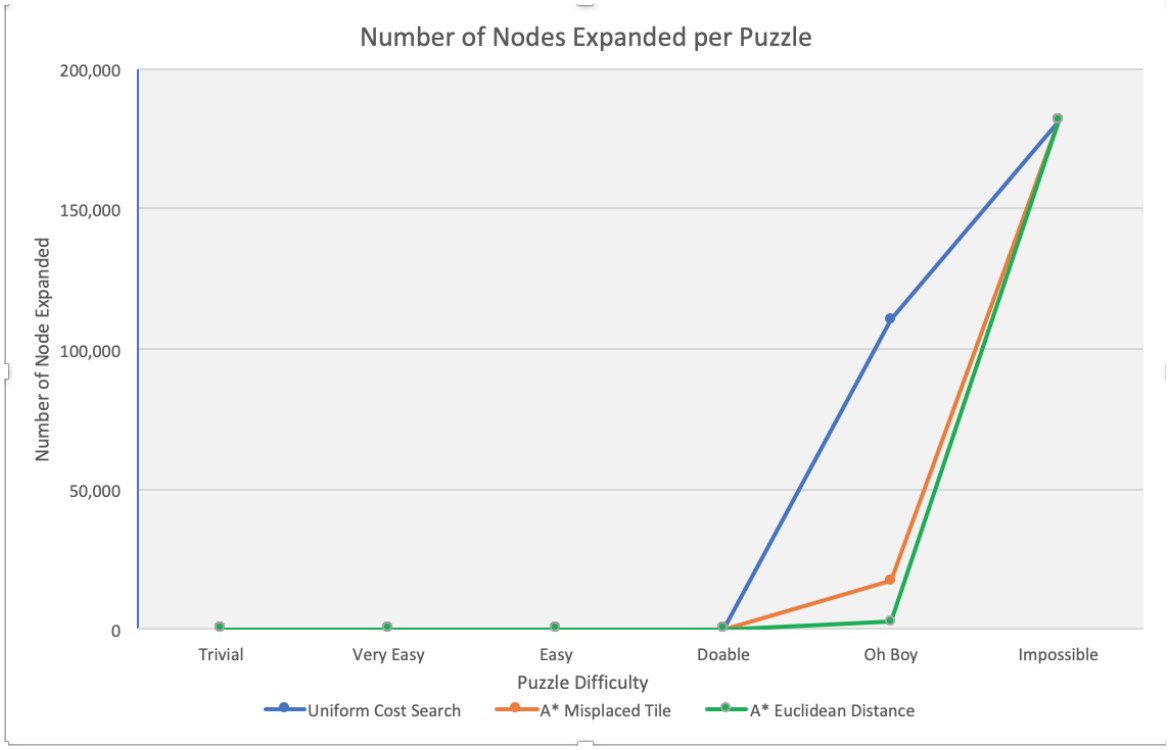
We implemented a graph-search algorithm to the Uniform Cost Search, A* with Misplaced Tile, and A* with Euclidean Distance. We created an “explored” set to keep a record

of all the explored nodes. This allowed us to prevent duplicates from being expanded because before expanding it checks if the node is in the frontier or in the explored set.

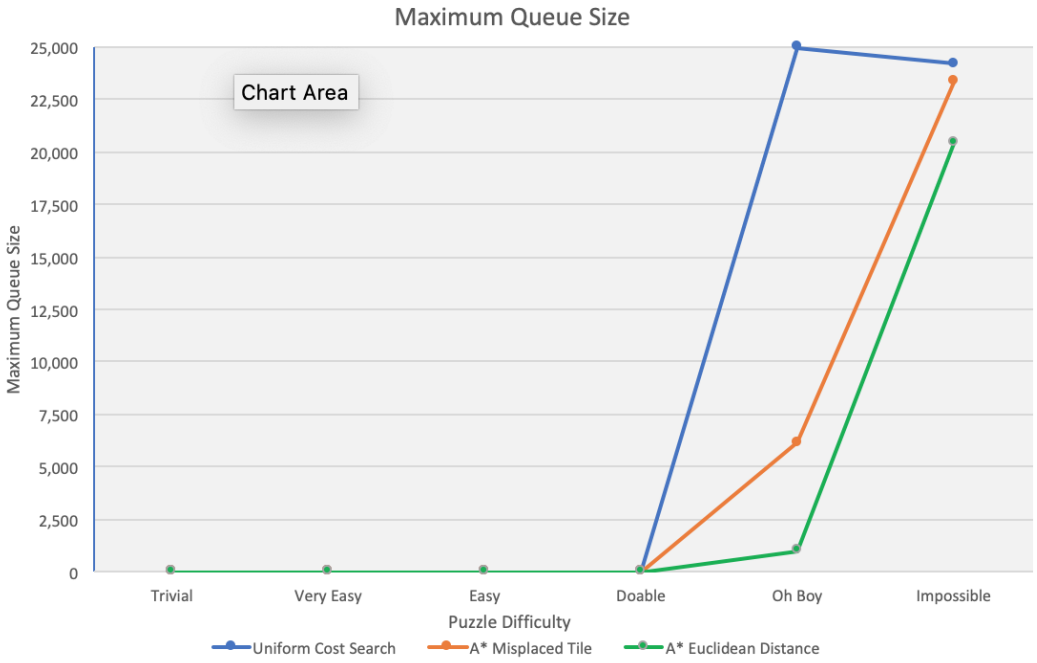
Comparing Algorithms:

When comparing the three algorithms the results varied depending on the difficulty of the puzzle. We found ourselves testing the six different puzzles provided in the "Project 1 - Report Guidelines" document. As we tested the algorithms we noticed that there was nearly no difference between the algorithms when the puzzles were extremely simple. However, as the difficulty of the puzzle increased the heuristic of the algorithms played an important role on the number of nodes expanded and the maximum size of the queue. We found that the A* with Euclidean Distance Heuristic is the best heuristic to use to solve the 8 puzzle problem.

Number of Nodes Expanded			
	Uniform Cost Search	A* Misplaced Tile	A* Euclidean Distance
Trivial (0)	0	0	0
Very Easy (1)	7	7	7
Easy (2)	6	6	6
Doable (3)	45	12	12
Oh Boy (4)	110421	17137	2515
Impossible (5)	181439	181439	181439



Maximum Size of Queue			
	Uniform Cost Search	A* Misplaced Tile	A* Euclidean Distance
Trivial (0)	1	1	1
Very Easy (1)	4	4	4
Easy (2)	3	3	3
Doable (3)	17	6	6
Oh Boy (4)	24968	6162	991
Impossible (5)	24174	23318	20383



Uniform Cost Search:

```
Welcome to the 8 puzzle solver created by 862054277 and 862132870.
Authors: Josh McIntyre and Edwin Leon
Type "1" to use a default puzzle, or "2" to enter your own puzzle: 1
Enter 1 for Uniform Cost Search
Enter 2 for A* with the Misplaced Tile heuristic
Enter 3 for A* with the Eucledian distance heuristic
1
You have selected Uniform Cost Search
1 0 3
4 2 6
7 5 8
The best state to expand with g(n) = 1 and h(n) = 0 is...

1 2 3
4 0 6
7 5 8
The best state to expand with g(n) = 2 and h(n) = 0 is...

1 2 3
4 5 6
7 0 8
The best state to expand with g(n) = 3 and h(n) = 0 is...

1 2 3
4 5 6
7 8 0
Number of nodes expanded is 18
The maximum number of nodes in the queue at any one time: 9
Solution found in 3 moves
```

A* with Misplaced Tile Heuristic:

```
Welcome to the 8 puzzle solver created by 862054277 and 862132870.
Authors: Josh McIntyre and Edwin Leon
Type "1" to use a default puzzle, or "2" to enter your own puzzle: 1
Enter 1 for Uniform Cost Search
Enter 2 for A* with the Misplaced Tile heuristic
Enter 3 for A* with the Eucledian distance heuristic
2
You have selected A* with the Misplaced Tile heuristic
1 0 3
4 2 6
7 5 8
The best state to expand with g(n) = 1 and h(n) = 3 is...

1 2 3
4 0 6
7 5 8
The best state to expand with g(n) = 2 and h(n) = 2 is...

1 2 3
4 5 6
7 0 8
The best state to expand with g(n) = 3 and h(n) = 1 is...

1 2 3
4 5 6
7 8 0
Number of nodes expanded is 11
The maximum number of nodes in the queue at any one time: 6
Solution found in 3 moves
```

A* with Euclidean Distance Heuristic:

```
Welcome to the 8 puzzle solver created by 862054277 and 862132870.
Authors: Josh McIntyre and Edwin Leon
Type "1" to use a default puzzle, or "2" to enter your own puzzle: 1
Enter 1 for Uniform Cost Search
Enter 2 for A* with the Misplaced Tile heuristic
Enter 3 for A* with the Eucledian distance heuristic
3
You have selected A* with the Eucledian distance heuristic
1 0 3
4 2 6
7 5 8
The best state to expand with g(n) = 1 and h(n) = 3.0 is...

1 2 3
4 0 6
7 5 8
The best state to expand with g(n) = 2 and h(n) = 2.0 is...

1 2 3
4 5 6
7 0 8
The best state to expand with g(n) = 3 and h(n) = 1.0 is...

1 2 3
4 5 6
7 8 0
Number of nodes expanded is 11
The maximum number of nodes in the queue at any one time: 6
Solution found in 3 moves
```

Conclusion:

After analyzing our data, A* search algorithm with the Euclidean distance heuristic is the better algorithm of the three for solving the 8-puzzle problem. The next best algorithm to use is the misplaced tile heuristic in the A* search algorithm, and the least efficient of the three is uniform cost search.