Report

Design of Project

- Search Functions
    - Euclidean() - the A* with euclidean distance heuristic algorithm
    - Misplaced() - the A* with misplaced tile heuristic algorithm
    - UCS() - the uniform cost search algorithm
- Problem Classes
    - int Empty() - this is used to keep track of the 0 you type into the puzzle which is also blank space
    - int EuclideanFct() - this is the euclidean distance heuristic and calculates g(n)
    - int MisplacedFct() - this is the misplaced tile heuristic and calculates h(n)
    - problem* Parent - this is a pointer to the parent node
    - problem* Right() - this moves the blank space right
    - problem* Left() - this moves the blank space left
    - problem* Down() - this moves the blank space down
    - problem* Up() - this moves the blank space up
    - vector GoalPzl - this is the goal state used to compare the current puzzle to and it is the goal where you want your puzzle to look like this
    - vector CurrentPzl - this is the changing state of the puzzle so it is the current puzzle
    - void print() - after each move is made/done this prints the puzzle
    - bool Checks() - this checks to see if our final solution is the same as the goalpuzzle

Optimizing

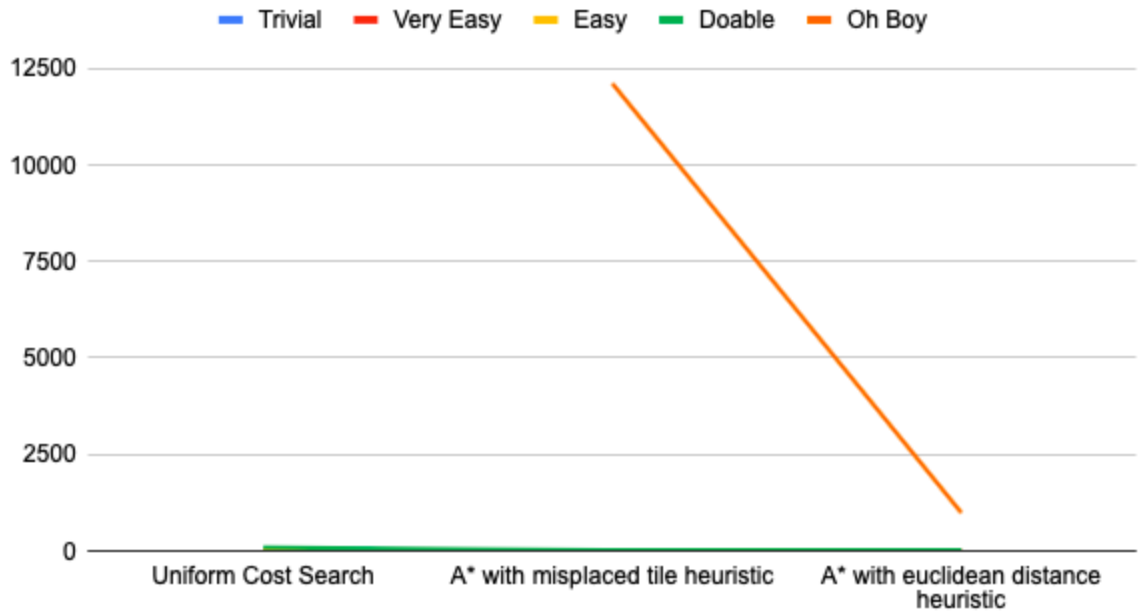
- It finds the most optimal solution per puzzle by having the least number of nodes expanded.
- Since graphs have cycles and trees do not, it means that tree searches can be done faster than graph searches because there will not be the possibility of any infinite loops and also because tree searches do not explore the same node twice. Therefore, I used a tree search over a graph search.

Tables and Graphs

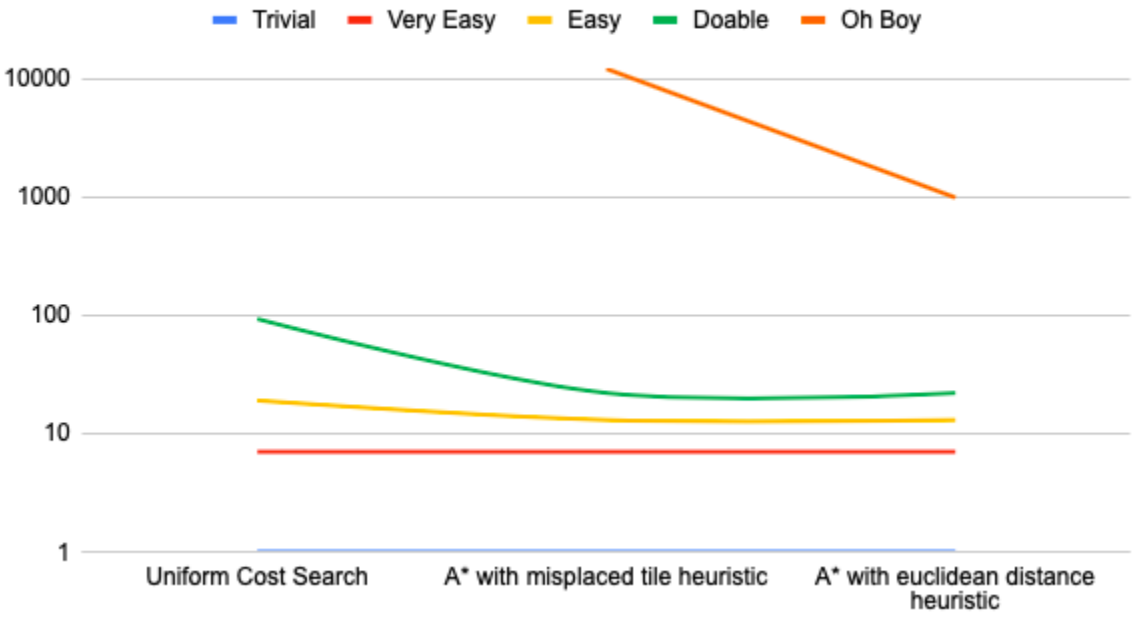
Number of Nodes Expanded

	Uniform Cost Search	**A* with misplaced tile heuristic**	**A* with euclidean distance heuristic**
Trivial	1	1	1
Very Easy	9	9	9
Easy	19	13	13
Doable	93	26	26
Oh Boy	N/A	12113	987
Impossible	∞	∞	∞

Number of Nodes Expanded

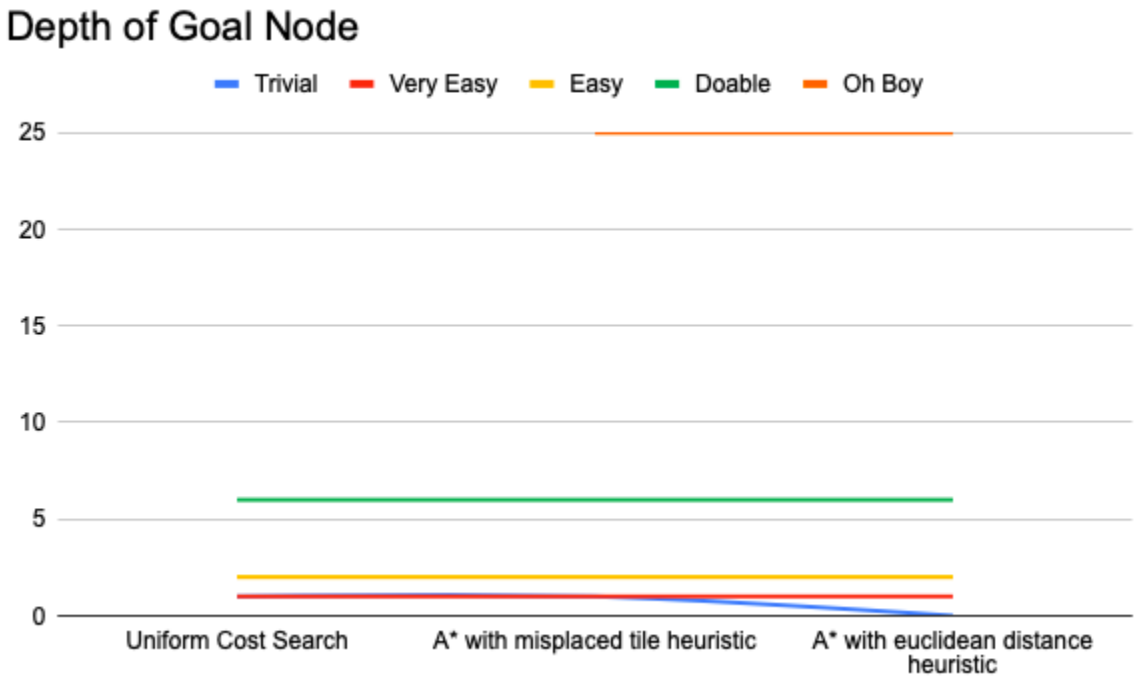


Adjusted Number of Nodes Expanded



Depth of Goal Node

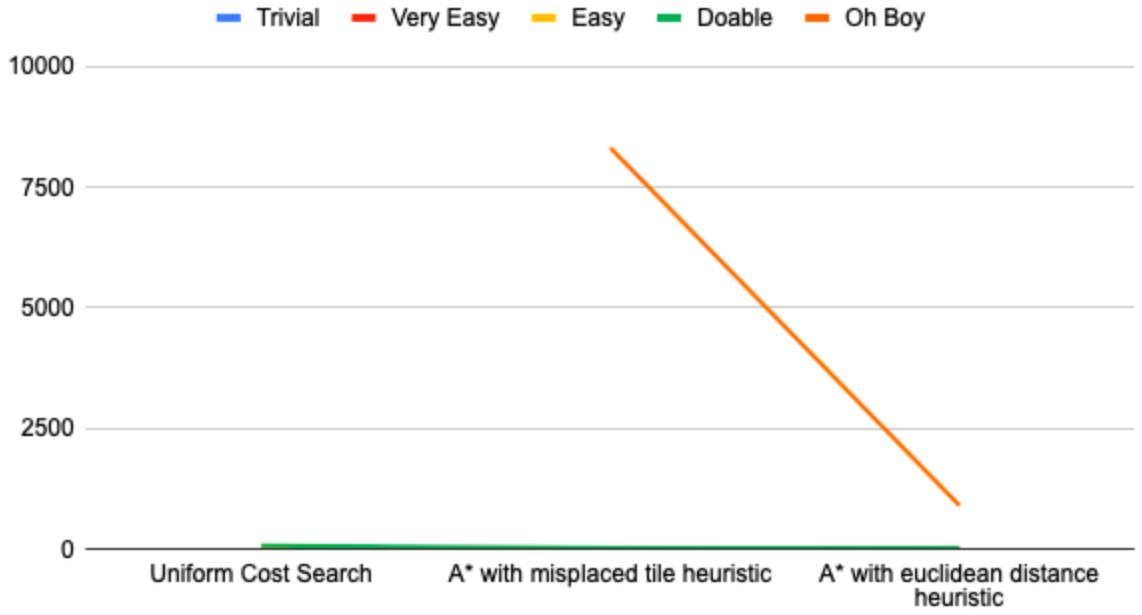
	Uniform Cost Search	A* with misplaced tile heuristic	A* with euclidean distance heuristic
Trivial	1	1	0
Very Easy	1	1	1
Easy	2	2	2
Doable	6	6	6
Oh Boy	N/A	25	25
Impossible	∞	∞	∞



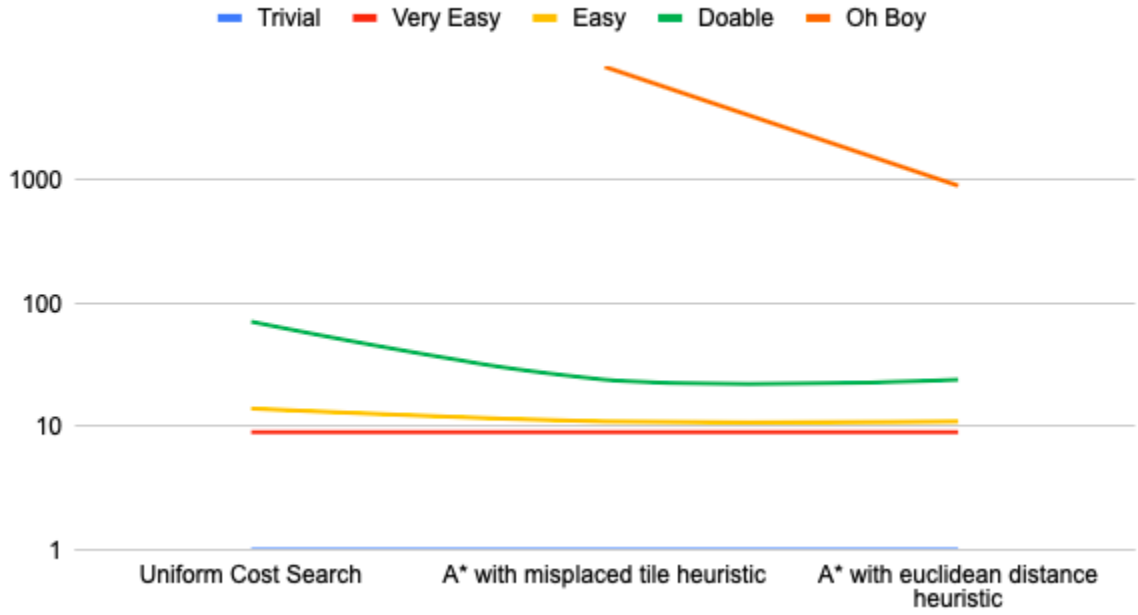
Max Queue Size

	Uniform Cost Search	**A* with misplaced tile heuristic**	**A* with euclidean distance heuristic**
Trivial	1	1	1
Very Easy	7	7	7
Easy	14	11	11
Doable	71	24	24
Oh Boy	N/A	8316	901
Impossible	∞	∞	∞

Max Queue Size



Adjusted Max Queue Size



Findings

The algorithms have the same maximum queue size when the puzzle difficulty levels are trivial and very easy and very close/similar maximum queue size when the puzzle difficulty level is easy. The maximum queue size for the Uniform Cost Search (UCS) algorithm is 14 whereas both the heuristic cost search algorithms have a maximum queue size of 11. There is a bit of a larger difference, but not too drastic, when it comes to the difficulty level of doable. However, this slight difference is only in the UCS algorithm (71) when the puzzle has a doable difficulty whereas the two heuristic algorithms have the same maximum queue size (24).The algorithms start to have very different maximum queue sizes when it comes to the difficulty level of oh boy. There is a large difference between the heuristic algorithms, which is thousands more. The UCS algorithm did not do the oh boy puzzle. The impossible level has an infinite maximum queue size for all the algorithms.

For the number of nodes expanded, the changes and differences are similar to that of the maximum queue size. For the puzzle difficulty levels of trivial and easy, the number of nodes expanded is the same for all of the algorithms. However, again, there is a slight difference in the UCS algorithm when it comes to the difficulty level of easy. For the number of nodes expanded, the UCS algorithm has a bit more number of nodes expanded (19) than the heuristic algorithms whereas the heuristic algorithms have the same amount of number of nodes expanded (13). There is a bit of a larger difference, but not too drastic, when it comes to the difficulty level of doable. However, this slight difference is only in the UCS algorithm (93) when the puzzle has a doable difficulty whereas the two heuristic algorithms have the same maximum queue size (26). The UCS algorithm did not do the oh boy puzzle. For the difficulty level of oh boy of the puzzle, the misplaced tile heuristic (MTH) algorithm has more expanded number of nodes than the euclidean distance heuristic (EDH) algorithm. The impossible level has an infinite number of nodes expanded for all the algorithms.

The number of nodes expanded as well as the maximum queue size are dependent on the difficulty level of the puzzle where for both it increases as the difficulty level of the puzzle increases. For both the number of nodes expanded as well as the maximum queue size, the UCS algorithm ends up having a much larger amount than the MTH and EDH algorithms. For both the number of nodes expanded as well as the maximum queue size, the UCS algorithm cannot do the oh boy difficulty level of the puzzle. For both the number of nodes expanded as well as the maximum queue size, the MTH algorithm ends up having more than the EDH algorithm. For both the number of nodes expanded as well as the maximum queue size, there is an infinite amount for impossible difficulty level puzzle. This is observed by analyzing the graphs and tables.

References

- https://www.geeksforgeeks.org/a-search-algorithm/?ref=lbp
- https://www.geeksforgeeks.org/convert-a-string-to-integer-array-in-c-c/
- https://www.geeksforgeeks.org/uniform-cost-search-dijkstra-for-large-graphs/

A trace of the Euclidean distance A* on the given problem

```
Welcome to asimi003 8 puzzle solver.
Type "1" to use default puzzle, or "2" to enter your own puzzle.
2
Enter your puzzle, use a zero to represent the blank
Enter the array, put a comma and a space after each number except the last
Example: 1, 2, 3, 4, 5, 6, 7, 8, 0
1, 2, 3, 4, 8, 0, 7, 6, 5
Enter your choice of algorithm
1. Uniform Cost Search
2. A* with the Misplaced Tile heuristic.
3. A* with the Eucledian distance heuristic.
3

Expanding state
1 2 3
4 8 0
7 6 5

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

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

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

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

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

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

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

The best state to expand with g(n) = 5 and h(n) = 0 is...
1 2 0
```

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

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

The best state to expand with g(n) = 5 and h(n) = 0 is...
1 2 3
4 5 6
7 8 0
To solve this problem the search algorithm expanded a total of 9 nodes.
The maximum number of nodes in the queue at any one time: 10.
The depth of the goal node was 5.
```