Nathan Goynes

CSCE 420

Eight Puzzle Report

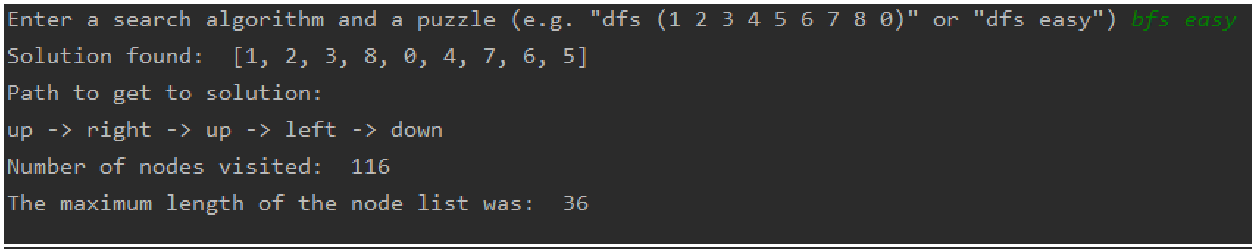
How to use 8Puzzle.py:

My program is relatively simple to use. Just type in a search algorithm and a starting state. For example, if you want to run DFS on a starting state of (1 2 3 4 5 6 7 8 0) just type in “dfs (1 2 3 4 5 6 7 8 0)”. It also has built in easy, medium, and hard starting states. To use these, just type something along the lines of “dfs easy”. For greedy and astar algorithms, all you must enter is the algorithm, the initial state, and the heuristic used (e.g. “astar (1 2 3 4 5 6 7 8 0) h1” or “astar medium h2”).

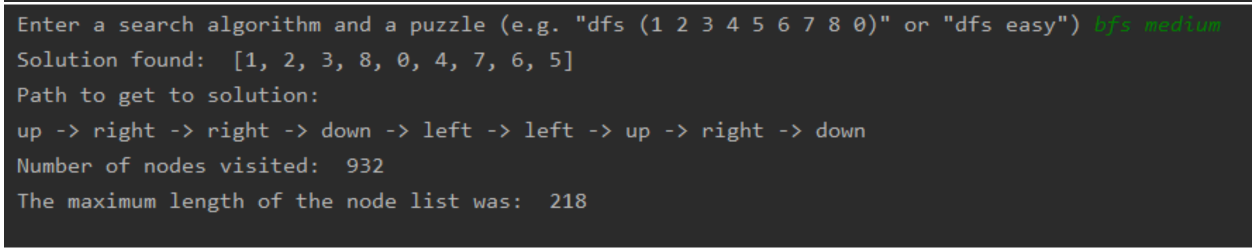
DFS:

DFS took more than ten minutes for the easy starting state. As such, it never found a solution in the time I gave it. This is probably because DFS will mostly iterate down one path of the tree for a long period of time. Because of this, it never gets close to the goal without taking an extreme amount of time.

BFS:



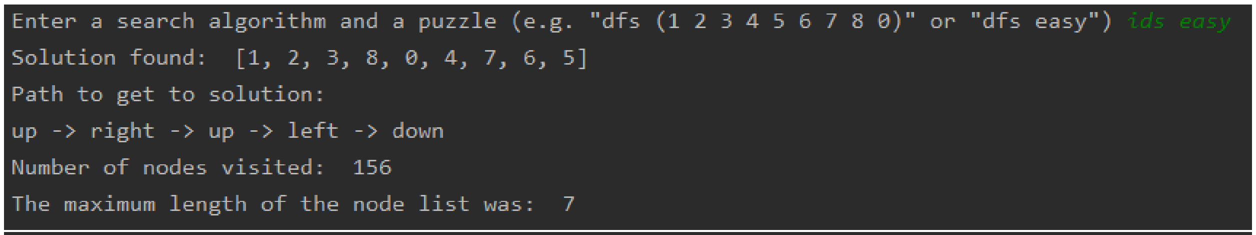
```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") bfs easy
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> up -> left -> down
Number of nodes visited:  116
The maximum length of the node list was:  36
```



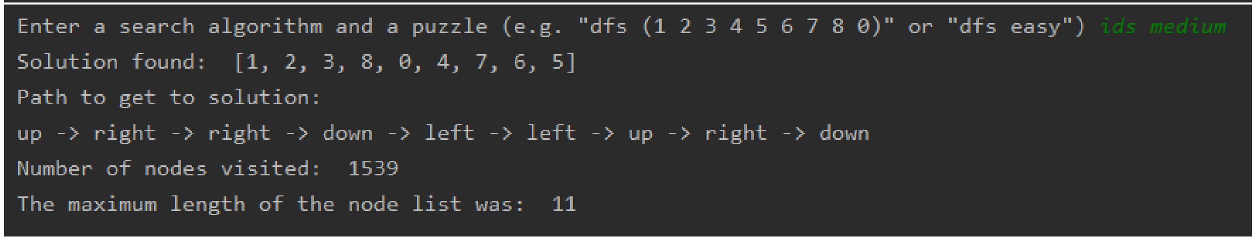
```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") bfs medium
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> right -> down -> left -> left -> up -> right -> down
Number of nodes visited:  932
The maximum length of the node list was:  218
```

BFS was able to return solutions for easy and medium starting states rather quickly, but the hard starting state was unable to find a solution in a quick amount of time. This could be because BFS just blindly follows the tree without considering the moves it made or will have to make. Because of this, it is good at finding solutions quickly that aren’t far from the start without getting stuck going down the wrong path like DFS. However, if the solution is far from the start, then it slows down significantly.

IDS:



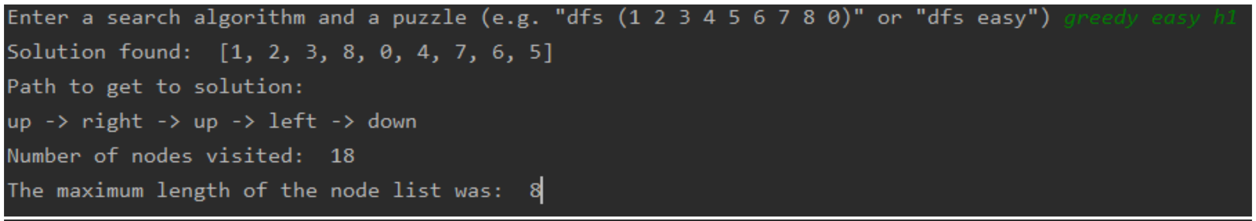
```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") ids easy
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> up -> left -> down
Number of nodes visited:  156
The maximum length of the node list was:  7
```



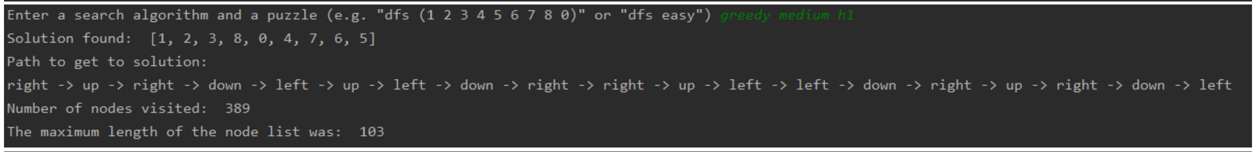
```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") ids medium
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> right -> down -> left -> left -> up -> right -> down
Number of nodes visited:  1539
The maximum length of the node list was:  11
```

IDS, like BFS, was able to execute easy and medium starting states quickly but was unable to find a solution to the hard state in good time. It is like BFS in execution, but the max size of the node list is much smaller. Because of this IDS is more efficient than BFS, but still suffers from the pitfalls of not being informed about where it’s going.

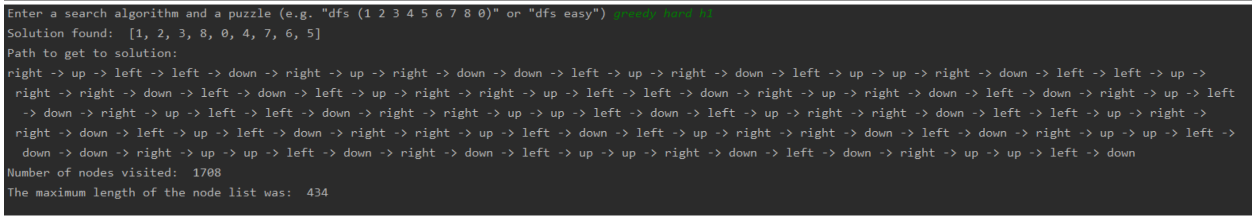
Greedy:



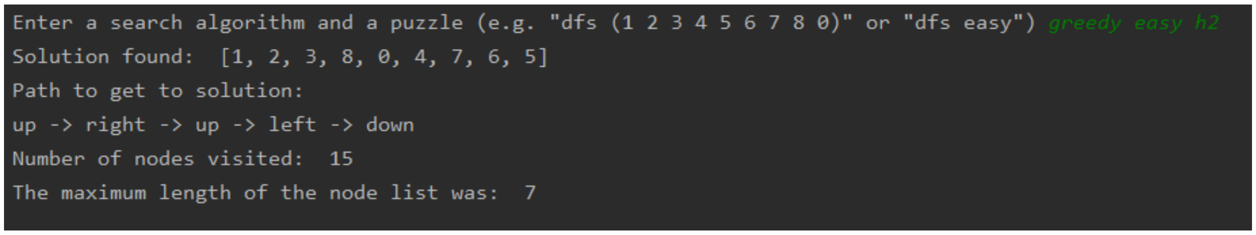
```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") greedy easy h1
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> up -> left -> down
Number of nodes visited:  18
The maximum length of the node list was:  8
```



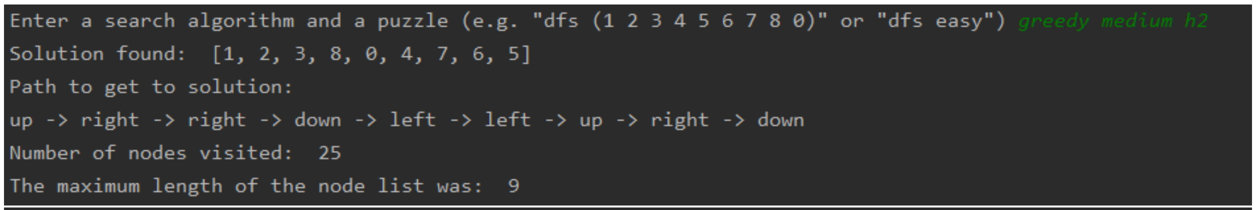
```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") greedy medium h1
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
right -> up -> right -> down -> left -> up -> left -> down -> right -> right -> up -> left -> left -> down -> right -> up -> right -> down -> left
Number of nodes visited:  389
The maximum length of the node list was:  103
```



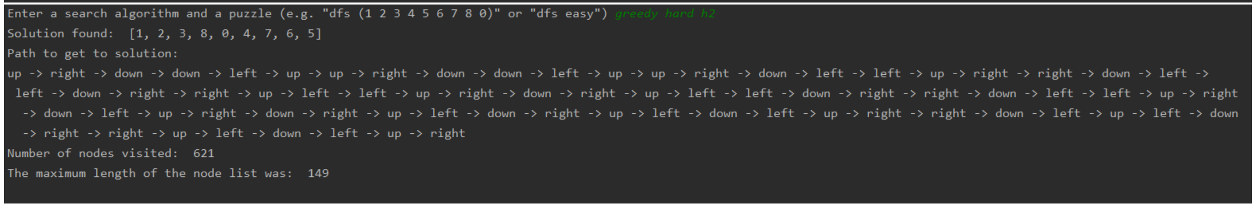
```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") greedy hard h1
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
right -> up -> left -> left -> down -> right -> up -> right -> down -> down -> left -> up -> right -> down -> left -> up -> up -> right -> down -> left -> left -> up ->
 right -> right -> down -> left -> down -> left -> up -> right -> right -> up -> left -> left -> down -> right -> up -> right -> down -> left -> down -> right -> up -> left
  -> down -> right -> up -> left -> left -> down -> right -> right -> up -> up -> left -> down -> left -> up -> right -> right -> down -> left -> left -> up -> right ->
 right -> down -> left -> up -> left -> down -> right -> right -> up -> left -> down -> left -> up -> right -> right -> down -> left -> down -> right -> up -> up -> left ->
  down -> down -> right -> up -> up -> left -> down -> right -> down -> left -> up -> up -> right -> down -> left -> down -> right -> up -> up -> left -> down
Number of nodes visited:  1708
The maximum length of the node list was:  434
```



```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") greedy easy h2
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> up -> left -> down
Number of nodes visited:  15
The maximum length of the node list was:  7
```



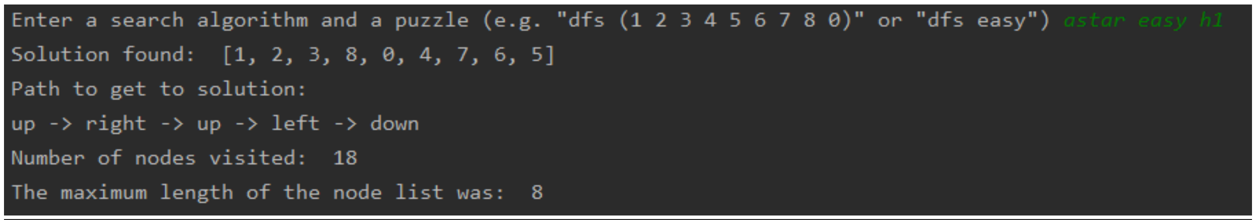
```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") greedy medium h2
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> right -> down -> left -> left -> up -> right -> down
Number of nodes visited:  25
The maximum length of the node list was:  9
```



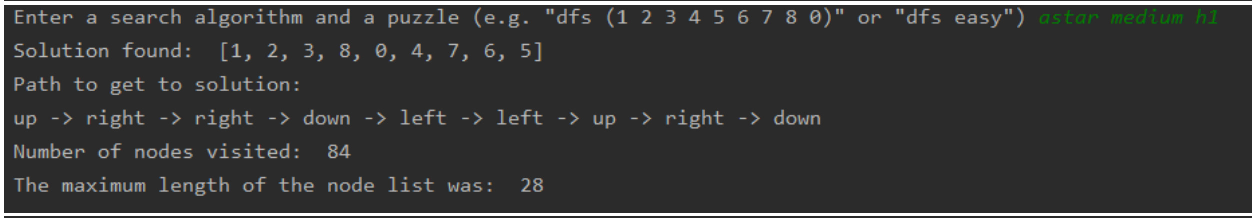
```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") greedy hard h2
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> down -> down -> left -> up -> up -> right -> down -> down -> left -> up -> up -> right -> down -> left -> left -> up -> right -> right -> down -> left ->
 left -> down -> right -> right -> up -> left -> left -> up -> right -> down -> right -> up -> left -> left -> down -> right -> right -> down -> left -> left -> up -> right
  -> down -> left -> up -> right -> down -> right -> up -> left -> down -> right -> up -> left -> down -> left -> up -> right -> right -> down -> left -> up -> left -> down
  -> right -> right -> up -> left -> down -> left -> up -> right
Number of nodes visited:  621
The maximum length of the node list was:  149
```

The greedy algorithm was able to find solutions to all starting states using both heuristics in a quick amount of time. In comparing h1 to h2, h2 visits far less nodes and the maximum length of the node list is much less.

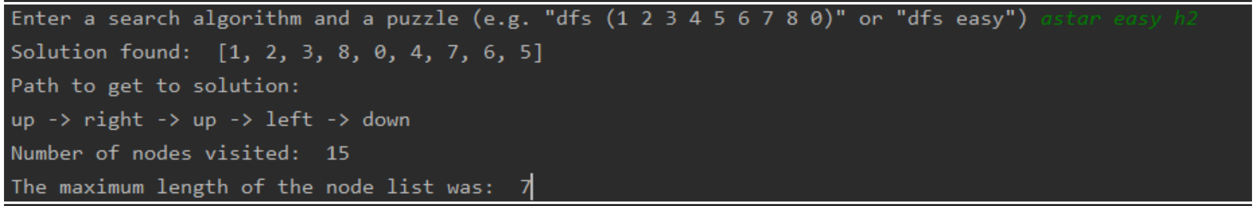
A*:



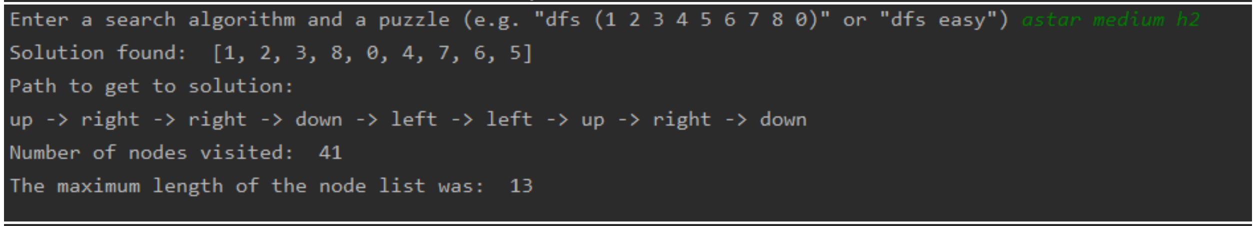
```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") astar easy h1
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> up -> left -> down
Number of nodes visited:  18
The maximum length of the node list was:  8
```



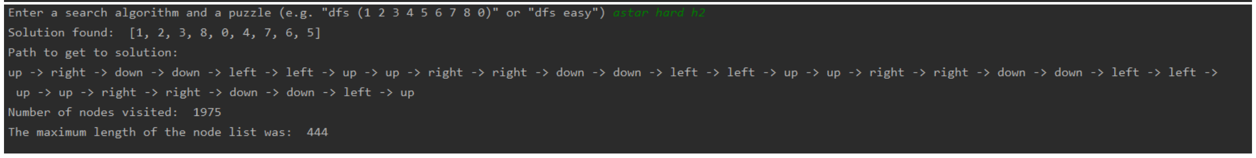
```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") astar medium h1
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> right -> down -> left -> left -> up -> right -> down
Number of nodes visited:  84
The maximum length of the node list was:  28
```



```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") astar easy h2
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> up -> left -> down
Number of nodes visited:  15
The maximum length of the node list was:  7
```



```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") astar medium h2
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> right -> down -> left -> left -> up -> right -> down
Number of nodes visited:  41
The maximum length of the node list was:  13
```



```
Enter a search algorithm and a puzzle (e.g. "dfs (1 2 3 4 5 6 7 8 0)" or "dfs easy") astar hard h2
Solution found:  [1, 2, 3, 8, 0, 4, 7, 6, 5]
Path to get to solution:
up -> right -> down -> down -> left -> left -> up -> up -> right -> right -> down -> down -> left -> left -> up -> up -> right -> right -> down -> down -> left -> left ->
 up -> up -> right -> right -> down -> down -> left -> up
Number of nodes visited:  1975
The maximum length of the node list was:  444
```

A* was also able to find solutions to all starting states. The main difference between A* and Greedy being that A* finds a much cheaper solution than Greedy for harder starting states. This is because A* keeps in mind the current cost when calculating the next cost of traversal. However, it visits more nodes and the maximum length of the node list is similar to Greedy.