Keerthi Srilakshmidaran

CS 4365.003

KXS180064

Report- Programming Assignment 1

1. Instructions on how to run the program is in the ReadMe document.
2. Sample input and its corresponding output:

   Input: 6 7 1 8 2 * 5 4 3 (Input is shown as the first step in solution in the screenshots)

   Output: 7 8 1 6 * 2 5 4 3 (Output is shown as the last step in solution in the screenshots)

   Depth First Search (DFS):

```
(base) keerthisri@keerthis-MBP 8puzzle % python homework1.py dfs /Users/keerthisri/input_file.txt
DFS Solution Found

Number of States Enqueued:  96

Number of Moves:  5

Solution:

[['6' '7' '1']
 ['8' '2' '*']
 ['5' '4' '3']]

[['6' '7' '1']
 ['8' '*' '2']
 ['5' '4' '3']]

[['6' '7' '1']
 ['*' '8' '2']
 ['5' '4' '3']]

[['*' '7' '1']
 ['6' '8' '2']
 ['5' '4' '3']]

[['7' '*' '1']
 ['6' '8' '2']
 ['5' '4' '3']]

[['7' '8' '1']
 ['6' '*' '2']
 ['5' '4' '3']]
```

Iterative Deepening Search (IDS):

```
(base) keerthisri@keerthis-MBP 8puzzle % python homework1.py ids /Users/keerthisri/input_file.txt
IDS Depth  1 : No Solution Found

IDS Depth  2 : No Solution Found

IDS Depth  3 : No Solution Found

IDS Depth  4 : No Solution Found

IDS Depth  5 : Solution Found

Number of States Enqueued:  60

Number of Moves:  5

Solution:

[['6' '7' '1']
 ['8' '2' '*']
 ['5' '4' '3']]

[['6' '7' '1']
 ['8' '*' '2']
 ['5' '4' '3']]

[['6' '7' '1']
 ['*' '8' '2']
 ['5' '4' '3']]

[['*' '7' '1']
 ['6' '8' '2']
 ['5' '4' '3']]

[['7' '*' '1']
 ['6' '8' '2']
 ['5' '4' '3']]

[['7' '8' '1']
 ['6' '*' '2']
 ['5' '4' '3']]
```

A* Heuristic 1- Number of Tiles in Wrong Position

```
(base) keerthisri@keerthis-MBP 8puzzle % python homework1.py astar1 /Users/keerthisri/input_file.txt
AStar1 Solution Found

Number of States Enqueued:  13

Number of Moves:  5

Solution:

[['6' '7' '1']
 ['8' '2' '*']
 ['5' '4' '3']]

[['6' '7' '1']
 ['8' '*' '2']
 ['5' '4' '3']]

[['6' '7' '1']
 ['*' '8' '2']
 ['5' '4' '3']]

[['*' '7' '1']
 ['6' '8' '2']
 ['5' '4' '3']]

[['7' '*' '1']
 ['6' '8' '2']
 ['5' '4' '3']]

[['7' '8' '1']
 ['6' '*' '2']
 ['5' '4' '3']]
```

A* Heuristic 2- Sum of Manhattan Distances

```
(base) keerthisri@keerthis-MBP 8puzzle % python homework1.py astar2 /Users/keerthisri/input_file.txt
AStar2 Solution Found

Number of States Enqueued:  11

Number of Moves:  5

Solution:

[['6' '7' '1']
 ['8' '2' '*']
 ['5' '4' '3']]

[['6' '7' '1']
 ['8' '*' '2']
 ['5' '4' '3']]

[['6' '7' '1']
 ['*' '8' '2']
 ['5' '4' '3']]

[['*' '7' '1']
 ['6' '8' '2']
 ['5' '4' '3']]

[['7' '*' '1']
 ['6' '8' '2']
 ['5' '4' '3']]

[['7' '8' '1']
 ['6' '*' '2']
 ['5' '4' '3']]
```

3. Comparative Analysis

The first A* heuristic (A*1) uses the number of tiles in the wrong position relative to the goal state while the second A* heuristic (A*2) uses the sum of Manhattan distances from each tile to the goal tile. Both A* algorithms use the number of moves from initial to the current state as g*(n). Looking at the screenshots of both these algorithms, we can see that they perform well compared to DFS and IDS. A*1 uses 5 moves with 13 states enqueued while A*2 uses 5 moves with 11 states enqueued. DFS and IDS also use 5 moves, but have higher states enqueued with DFS at 96 and IDS at 60. The lower state enqueued numbers for the A* algorithms show that they are more optimal because they visit fewer nodes and avoid expanding paths that are already expensive.

Comparing the A* algorithms to each other, we see that they both provide similar results- the only difference being in the number of states enqueued. A*2 which uses the sum of Manhattan Distance, has 2 less states enqueued than A*1 but both use the same number of moves. I would say that both heuristics performed well because they both find a solution while also eliminating bad paths to maintain optimality. However, A*2 performs a bit better than A*1 in eliminating bad paths as it has a lower number of states enqueued while also finding the solution in the same number of moves as A*1.