NAME: Hithesh Shanmugam

Code: CSC 480

ID:2068266

Video link: https://youtu.be/nDdpXMmTJBc

8-PUZZLE:

Length=Length of the solution path

Cost= Cost of the solution path

Time=Number of nodes popped

Space=Size of the queue at its max

State: EASY

Algorithm	Length	Cost	Time	Space
BFS	6	17	43	35
DFS	558	2351	609	453
IDS	6	17	107	6
UCS	6	17	24	15
GBFS	6	17	7	7
A*1	6	17	13	8
A*2	6	17	9	7
A*3	6	17	6	6

By analysing the data, we got from the table we can find the differences of search of how do they work. We can say that there are different approaches for this puzzle. The length and cost are similar for almost every algorithm except Depth-first search, it has the largest value for the length and cost. By comparing the time and space for the rest of the search algorithms A*3 search has the best solution of all. In the easy state the best algorithm to use would be the A*3 search and the DFS search would not be preferred here.

State: MEDIUM

Algorithm	Length	Cost	Time	Space
BFS	10	31	310	221
DFS	14	49	14	12
IDS	10	31	1312	2
UCS	10	31	149	98
GBFS	Terminated (runs more than 5 mins)			
A*1	10	31	67	51
A*2	10	31	44	36
A*3	10	31	26	22

We can see the difference between the easy and the medium states the factors vary and takes long to execute the program. In terms of length and cost except DFS and GBFS remaining all the search algorithms have the same values. In terms of the time DFS has the least value and IDS pops a greater number of nodes. In terms of space IDS is best suitable because the size of the queue at the end is just 2. GBFS is terminated because the it gets stuck in the loop and it is not optimal.

State: HARD

Algorithm	Length	Cost	Time	Space
BFS	Terminated (Runs more than 5 mins)			
DFS				
IDS				
UCS				
GBFS				
A*1				
A*2				
A*3	33	144	1480	982

Here the only algorithm works is the A*3 search and it takes more time than the other two states i.e., the time taken for the hard state is much big than easy and medium states. The remaining algorithms are terminated because,

BFS- memory constraints as it stores all the nodes of the present node to go to the next node.

DFS-the states keep reoccurring and it might also end up in infinite loop.

IDS- if there is a node close to root, but not in first few subtrees explored by DFS, then DFS reaches that node very late.

UCS- the open list is required to be kept sorted as priorities in priority queue needs to be maintained.

GBFS- it gets stuck in the loop and it is not optimal

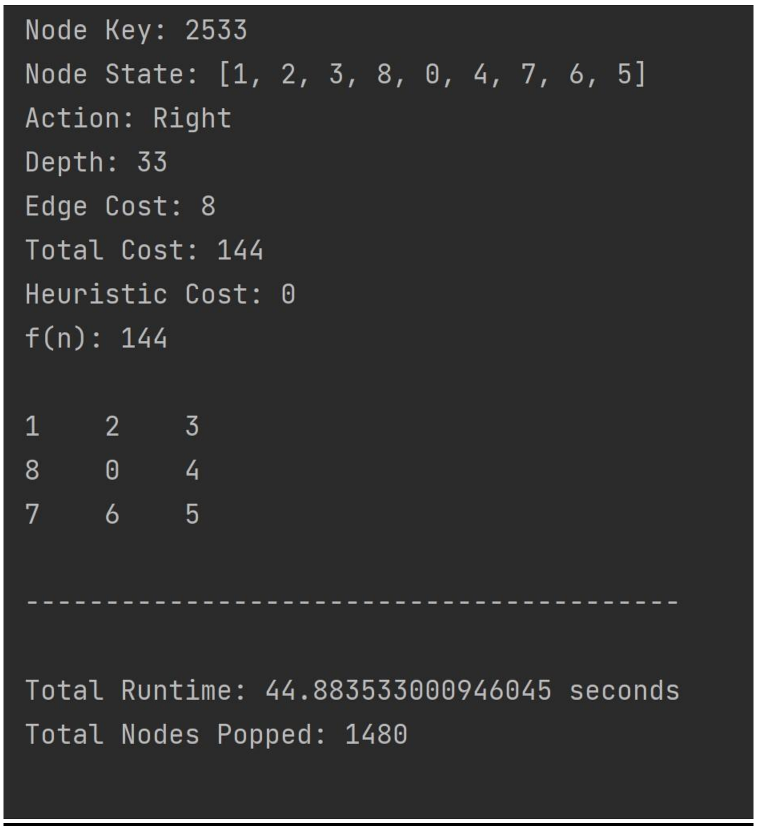
A*1- the performance of A* search is dependent on accuracy of heuristic algorithm used to compute the function h(n)

A*2- same as A*1 search

To do another heuristic I constructed a heuristic algorithm as A*3 search.

A*3 search:

The first search h1(n) is based on the number of tiles out of the place, and second search h2(n) is based on the number of tiles is in correct position. Now to construct the third heuristic function h3(n) I constructed it based on the product of the number of positions each tile is in correct position and their own cost.



```
Node Key: 2533
Node State: [1, 2, 3, 8, 0, 4, 7, 6, 5]
Action: Right
Depth: 33
Edge Cost: 8
Total Cost: 144
Heuristic Cost: 0
f(n): 144

1    2    3
8    0    4
7    6    5

-----------------------------------------


Total Runtime: 44.883533000946045 seconds
Total Nodes Popped: 1480
```

This is the final output for the A*3 search in hard state.

The other outputs I submitted as pdf files.