AI Homework 1 Report
Sourabh S. Shenoy
UIN: 225009050

8-Puzzle:

Python was used to code.

BFS:

To run the code:
Case 1)
python 8-puzzle.py bfs 134862705

Result:

```
Initial State:

-------------
| 1 | 3 | 4 |
-------------
| 8 | 6 | 2 |
-------------
| 7 | 0 | 5 |
-------------

Final State:

-------------
| 1 | 2 | 3 |
-------------
| 8 | 0 | 4 |
-------------
| 7 | 6 | 5 |
-------------

['LEFT', 'UP', 'RIGHT', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'LEFT', 'DOWN', 'RIGHT', 'UP']

11 Moves

0.0475599765778 seconds

dhcp-10-202-132-133:Desktop Sourabh$
```

Case 2)
python 8-puzzle.py bfs 281043765

Result:

```
Initial State:

-------------
| 2 | 8 | 1 |
-------------
| 0 | 4 | 3 |
-------------
| 7 | 6 | 5 |
-------------

Final State:

-------------
| 1 | 2 | 3 |
-------------
| 8 | 0 | 4 |
-------------
| 7 | 6 | 5 |
-------------

['RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'LEFT', 'DOWN', 'RIGHT', 'UP', 'RIGHT', 'DOWN', 'LEFT', 'LEFT', 'UP', 'RIGHT', 'DOWN']

17 Moves

1.08101391792 seconds

dhcp-10-202-132-133:Desktop Sourabh$
```

Case 3)
python 8-puzzle.py bfs 567408321

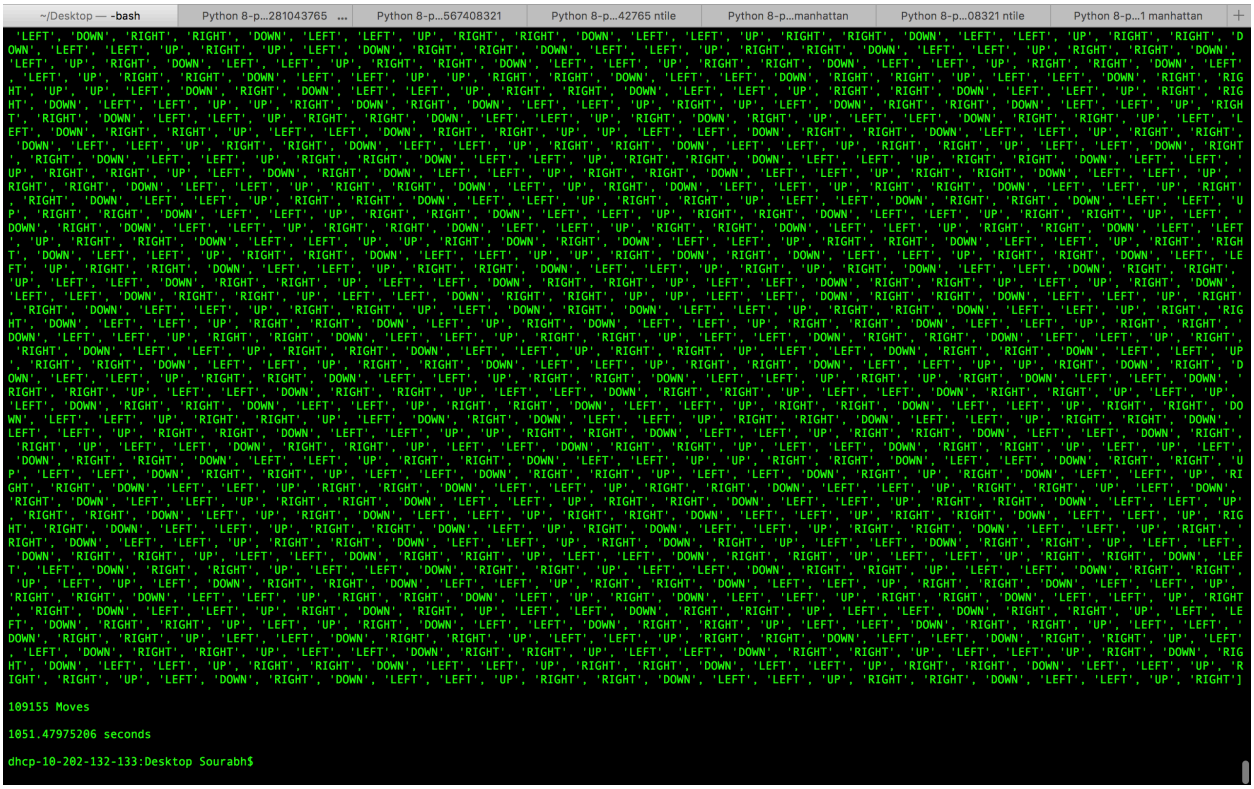
Result:

```
-------------
| 5 | 6 | 7 |
-------------
| 4 | 0 | 8 |
-------------
| 3 | 2 | 1 |
-------------

Final State:

-------------
| 1 | 2 | 3 |
-------------
| 8 | 0 | 4 |
-------------
| 7 | 6 | 5 |
-------------

['DOWN', 'LEFT', 'UP', 'UP', 'RIGHT', 'RIGHT', 'DOWN', 'DOWN', 'LEFT', 'LEFT', 'UP', 'UP', 'RIGHT', 'RIGHT', 'DOWN', 'DOWN', 'LEFT', 'LEFT', 'UP', 'UP', 'RIGHT', 'RIGHT', 'DOWN', 'D
OWN', 'LEFT', 'LEFT', 'UP', 'UP', 'RIGHT', 'DOWN']

30 Moves

46.4315569401 seconds

dhcp-10-202-132-133:Desktop Sourabh$
```

Analysis: BFS is a complete algorithm. It uses a FIFO queue. Time Complexity and space complexity, both are O(b^d). In general, BFS is not optimal, since space complexity and time complexity are large.

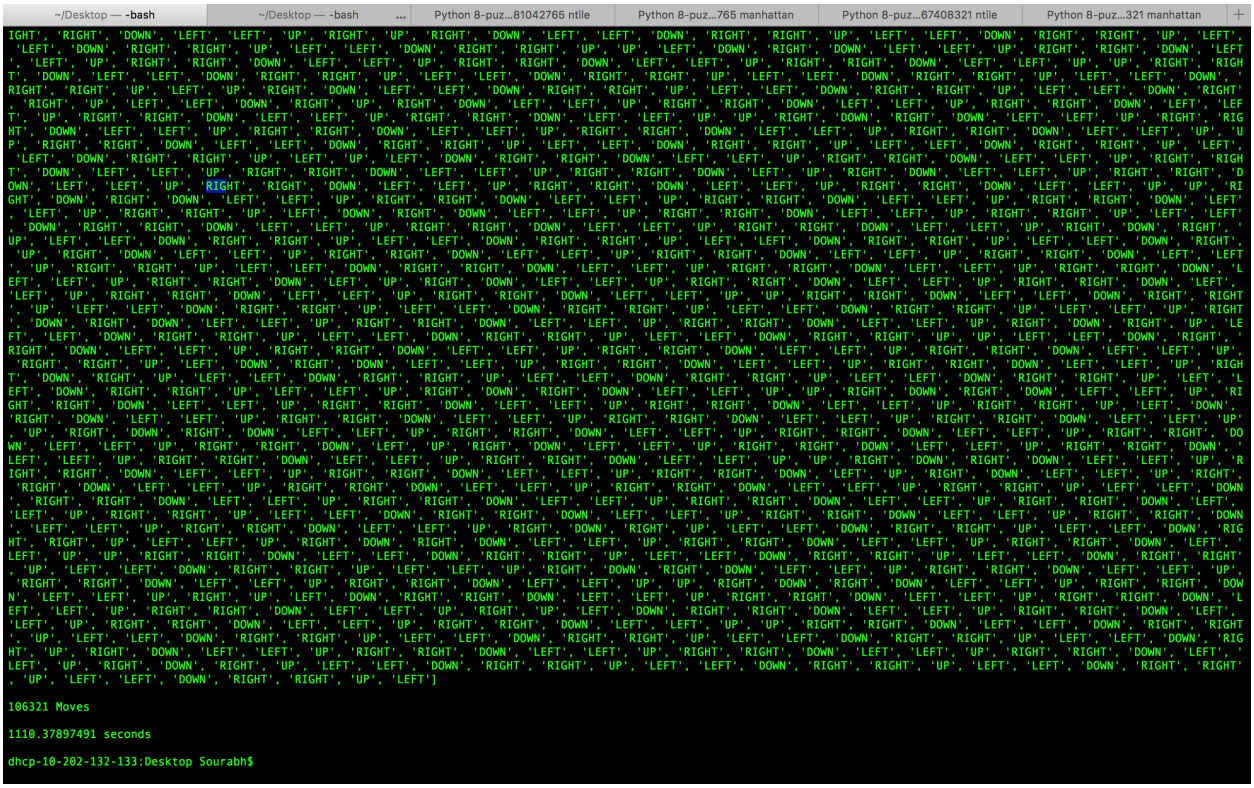
DFS:

To run the code:
Case 1)
python 8-puzzle.py dfs 134862705



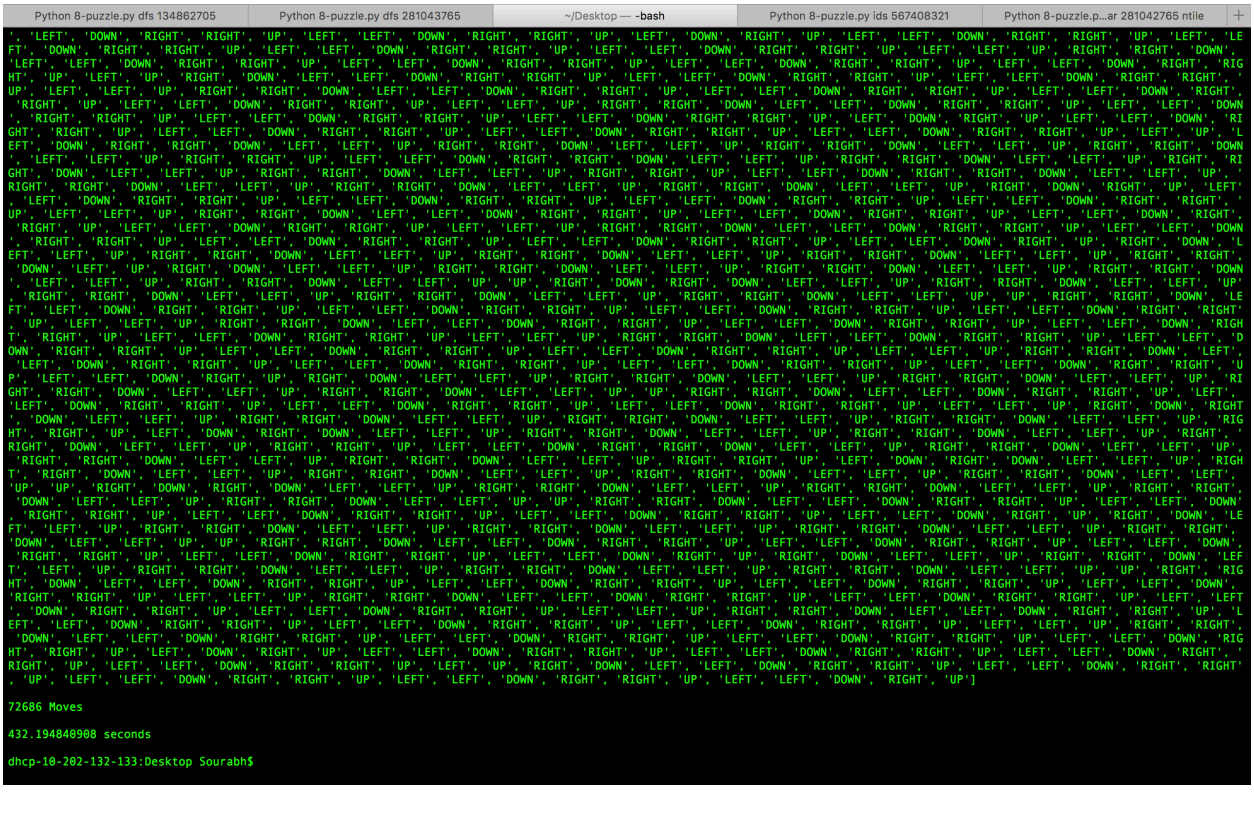
Case 2)
python 8-puzzle.py dfs 281043765

Result:



Case 3)
python 8-puzzle.py dfs 567408321

Result:

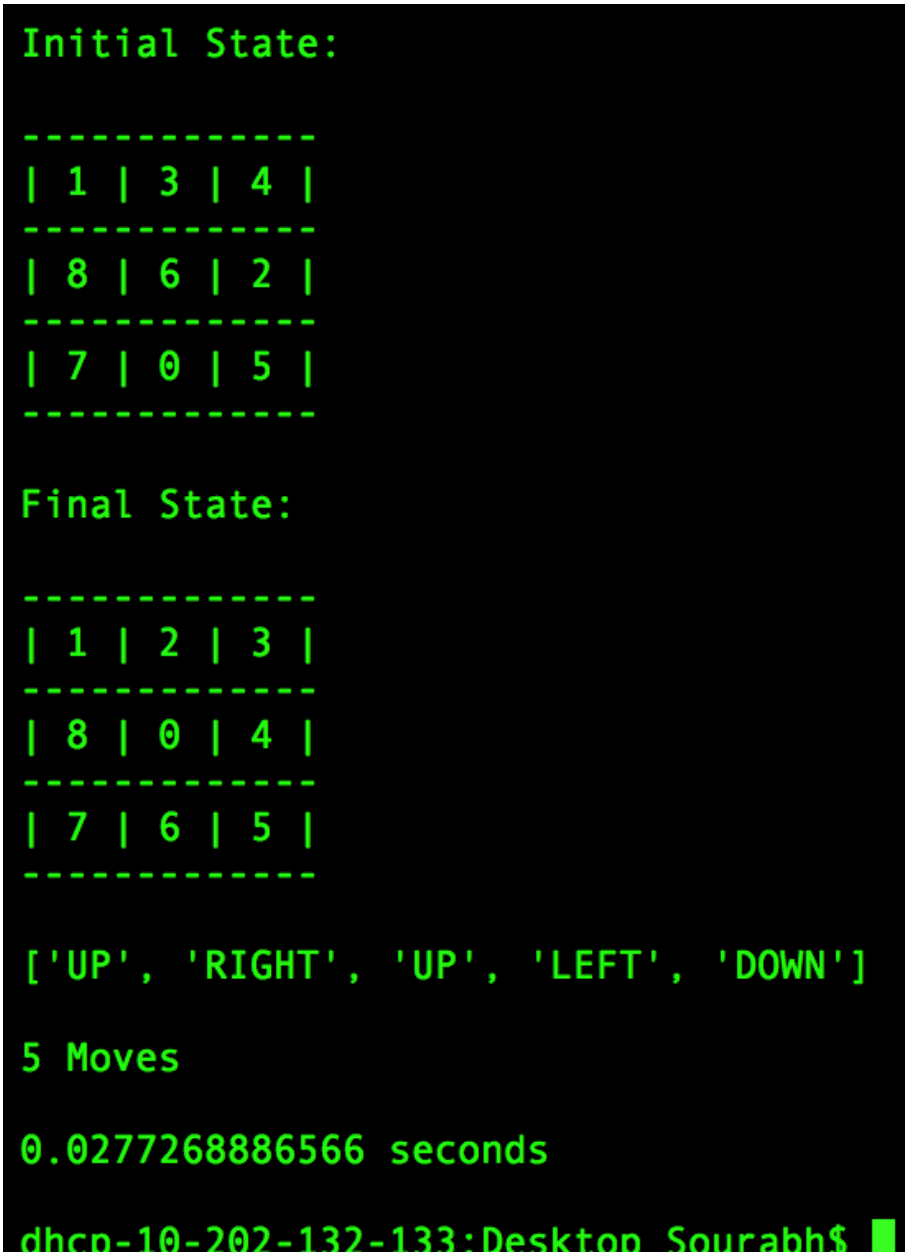


Analysis: Uses a LIFO Queue. DFS is complete in finite spaces, but incomplete otherwise. Time is O(b^m) and Space Complexity is O(bm). DFS is also not optimal due to time complexity issues.

IDS:

To run the code:
Case 1)
python 8-puzzle.py ids 134862705

Result:



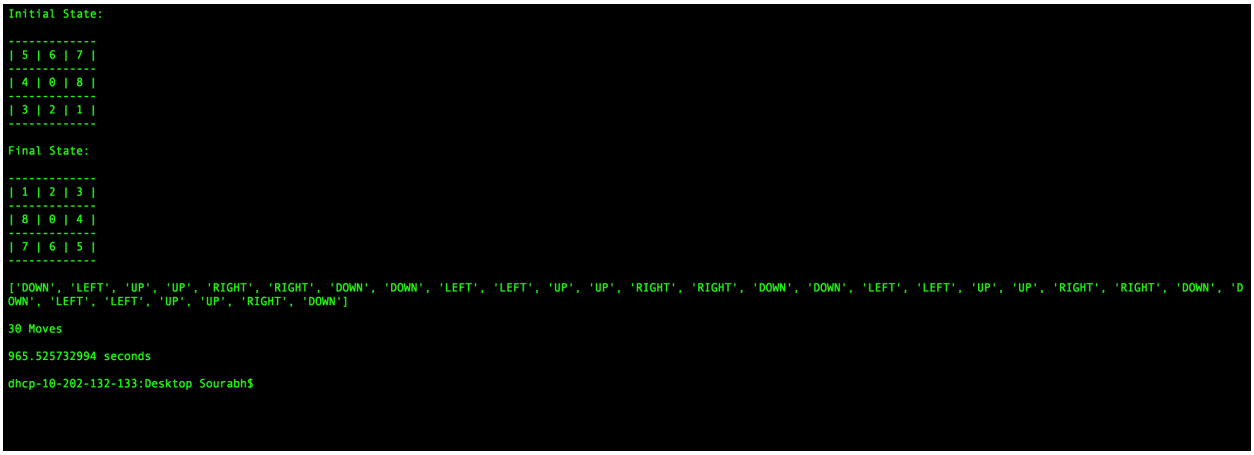
Case 2)
python 8-puzzle.py ids 281043765

Result:

```
Initial State:

-------------
| 2 | 8 | 1 |
-------------
| 0 | 4 | 3 |
-------------
| 7 | 6 | 5 |
-------------

Final State:

-------------
| 1 | 2 | 3 |
-------------
| 8 | 0 | 4 |
-------------
| 7 | 6 | 5 |
-------------

['UP', 'RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'LEFT', 'UP', 'RIGHT', 'DOWN']

9 Moves

0.278384923935 seconds

dhcp-10-202-132-133:Desktop Sourabh$
```

Case 3)
python 8-puzzle.py dfs 567408321

Result:

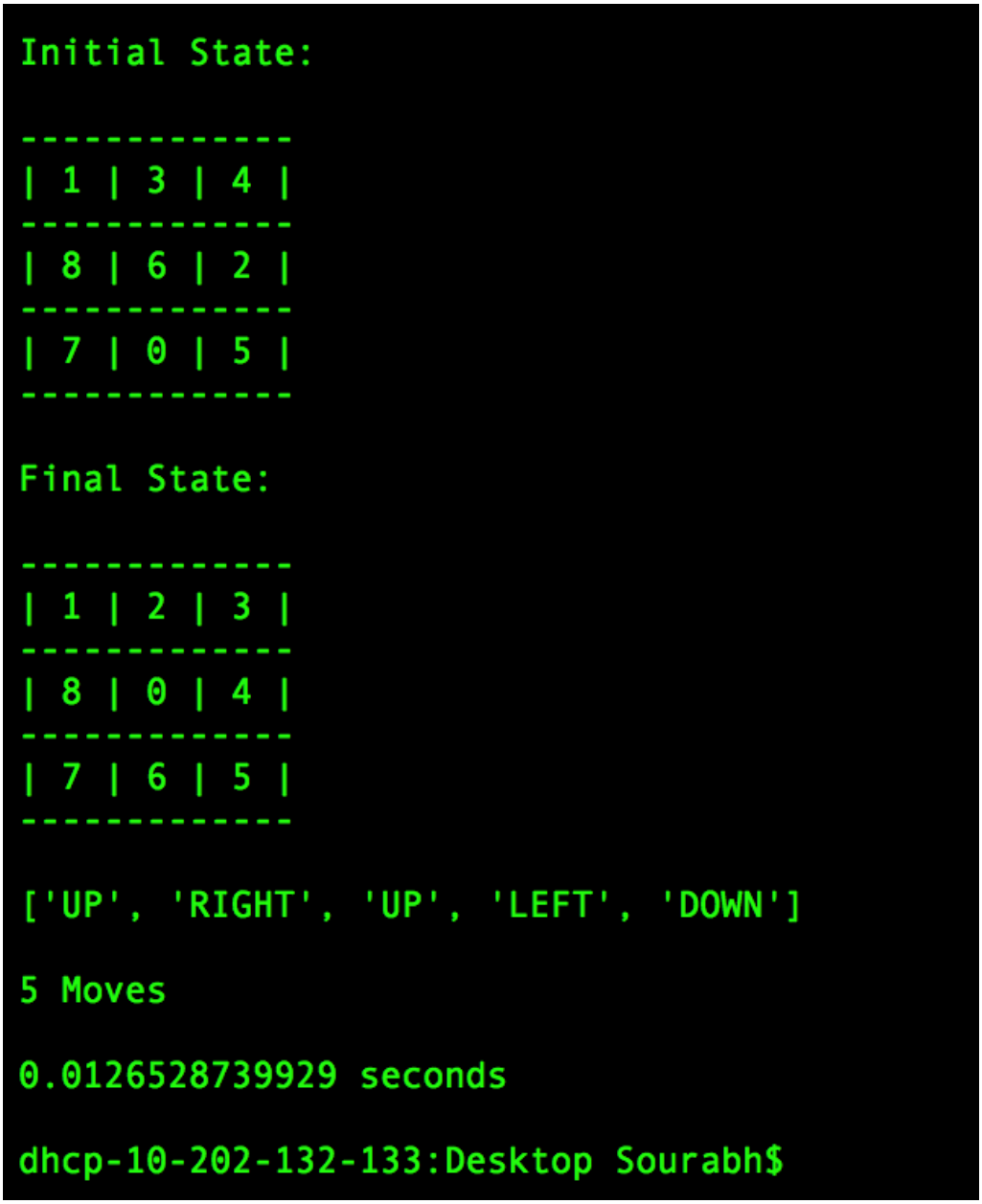


Analysis: IDS uses DFS with incremental depths per iteration. In essence, it combines the best features of both BFS and DFS. Although it performs repetitive tasks, in general it reaches the answer much faster. Also, since it is a variant of DFS, we only have one path in the memory at a time.

Greedy Best First Search:

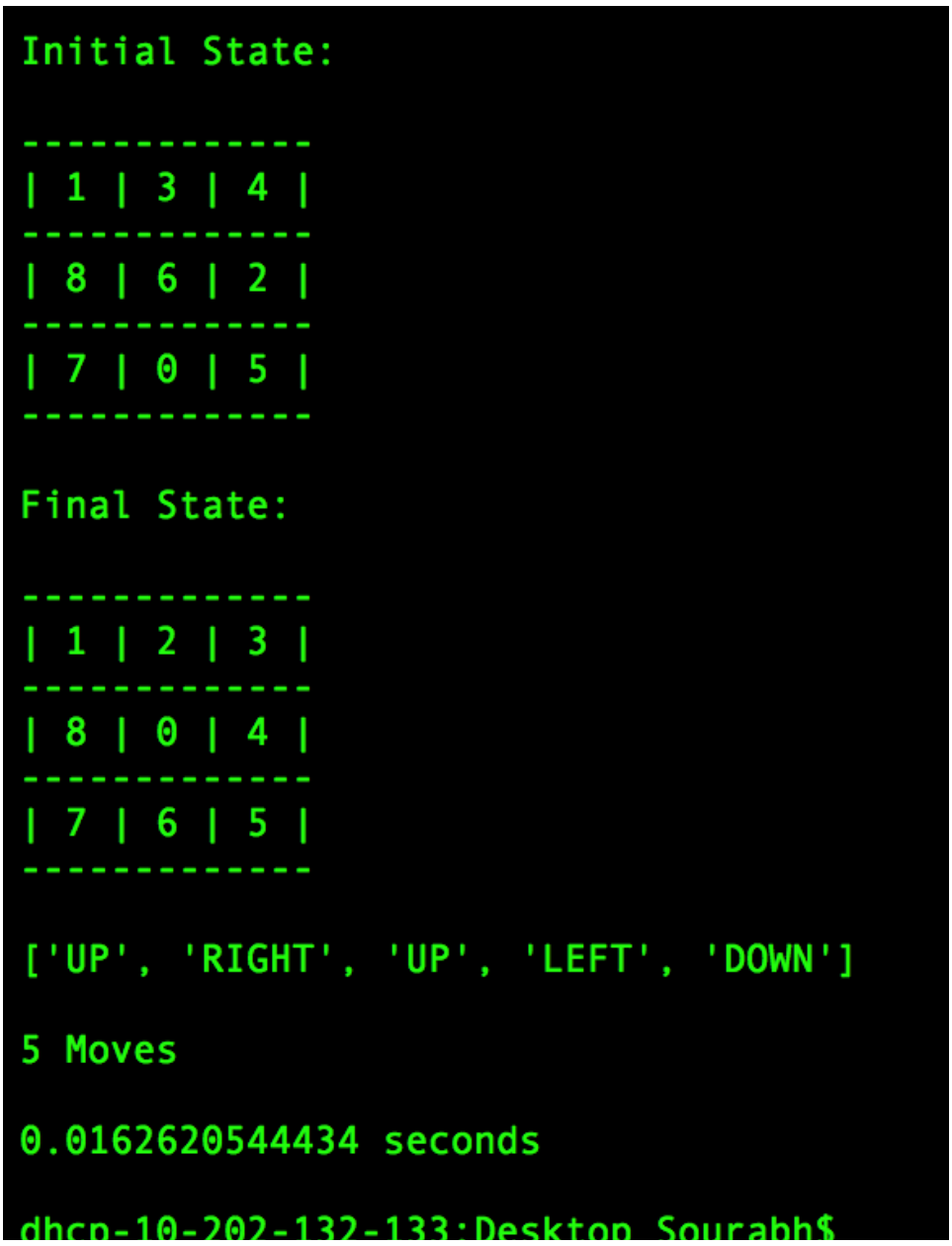
To run the code:
Case 1.a)
python 8-puzzle.py greedy 134862705 ntiles

Result:



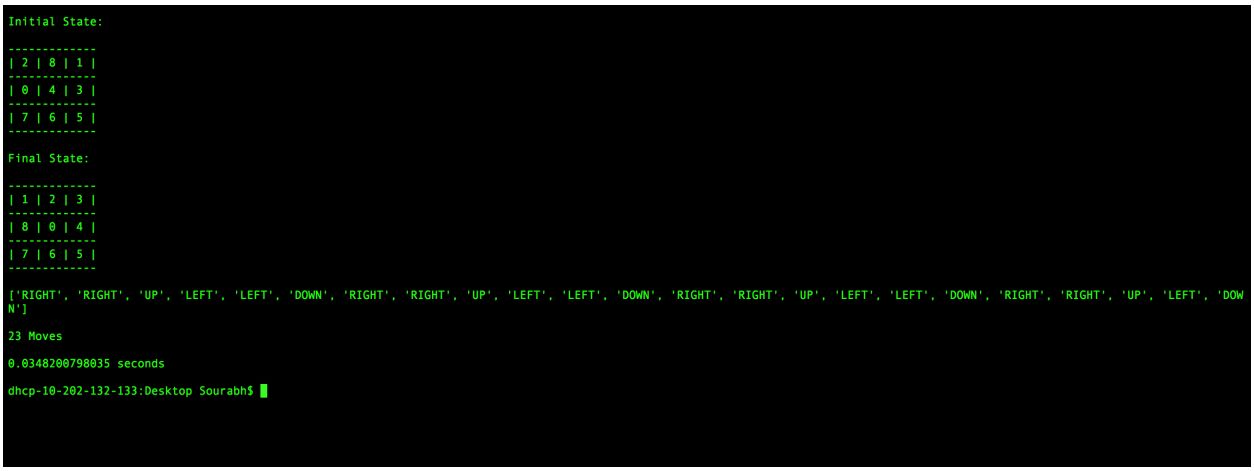
Case 1.b)
python 8-puzzle.py greedy 134862705 manhattan

Result:



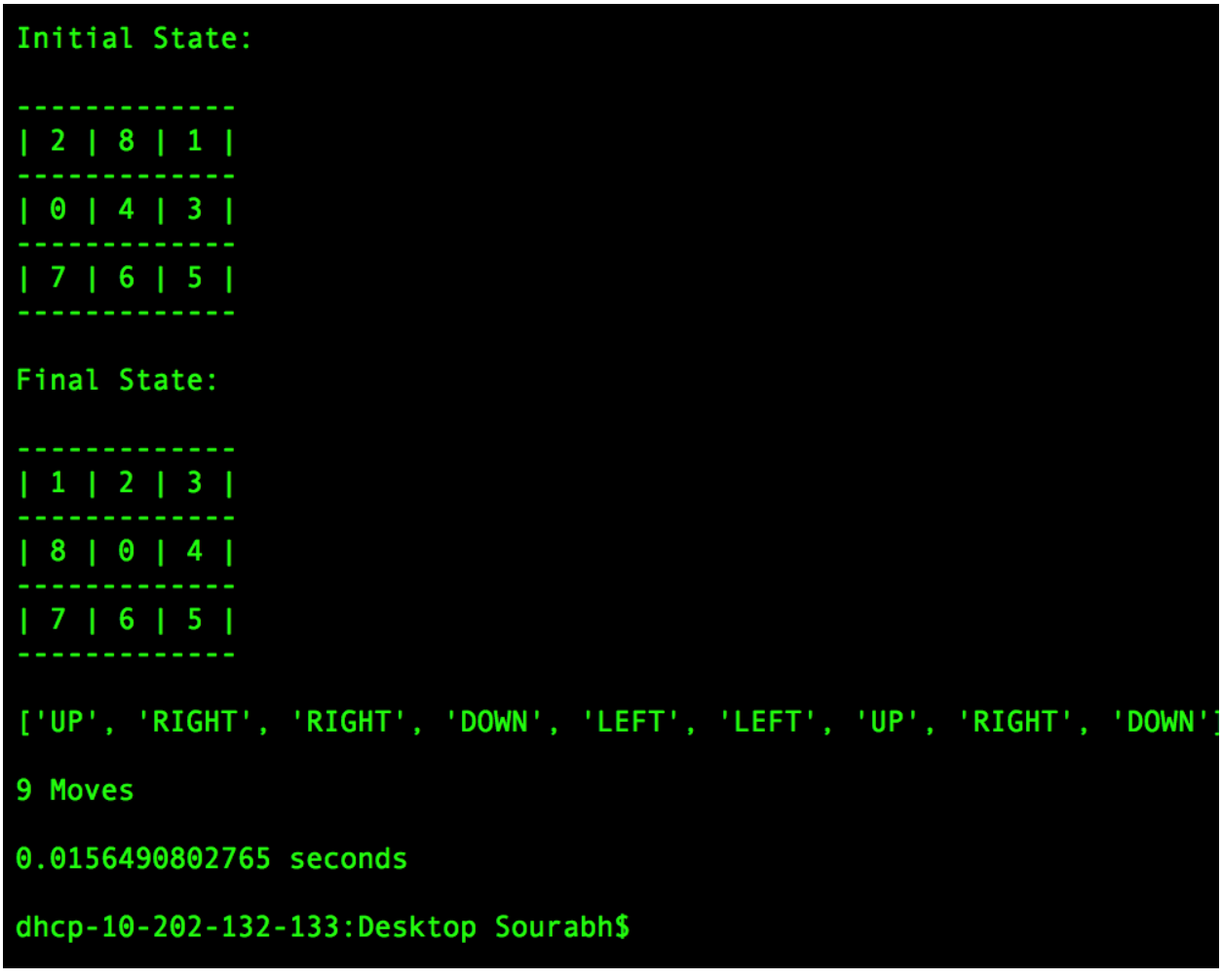
Case 2.a)
python 8-puzzle.py greedy 281043765 ntiles

Result:



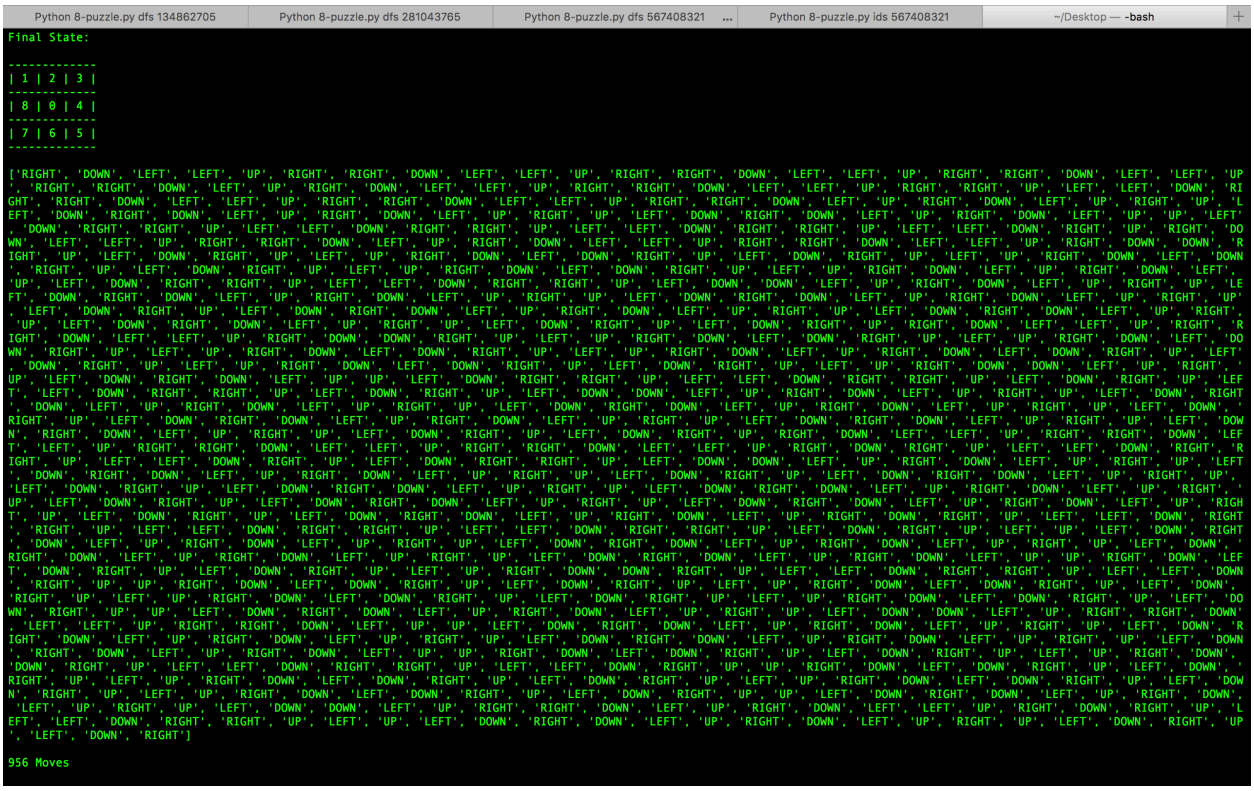
Case 2.b)
python 8-puzzle.py greedy 281043765 manhattan

Result:



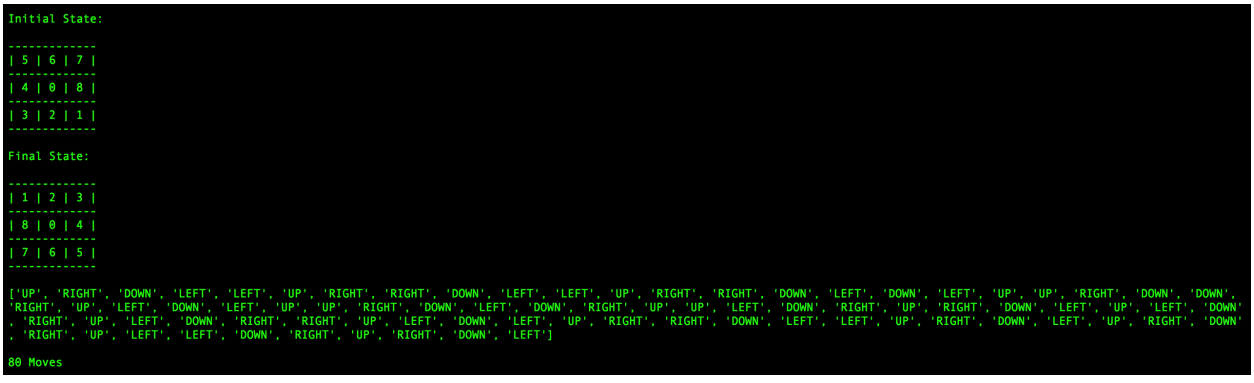
Case 3.a)
python 8-puzzle.py greedy 567408321 ntiles

Result:



Case 3.b)
python 8-puzzle.py greedy 567408321 manhattan

Result:



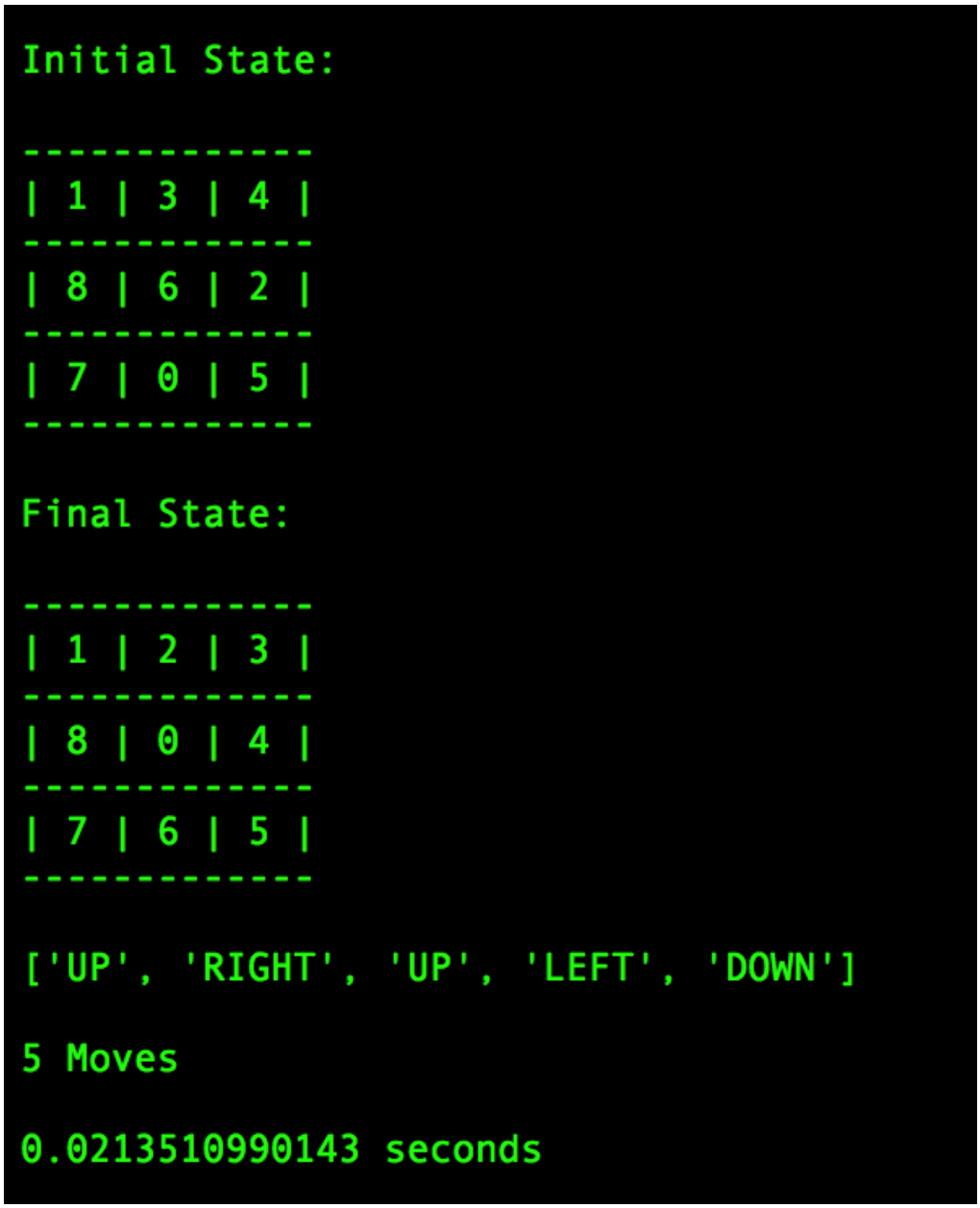
Analysis:
We use 2 popular heuristic functions

- Hamming priority function. The number of blocks in the wrong position, plus the number of moves made so far to get to the search node. Intutively, a search node with a small number of blocks in the wrong position is close to the goal, and we prefer a search node that have been reached using a small number of moves.
- Manhattan priority function. The sum of the Manhattan distances (sum of the vertical and horizontal distance) from the blocks to their goal positions, plus the number of moves made so far to get to the search node.

A* Search:

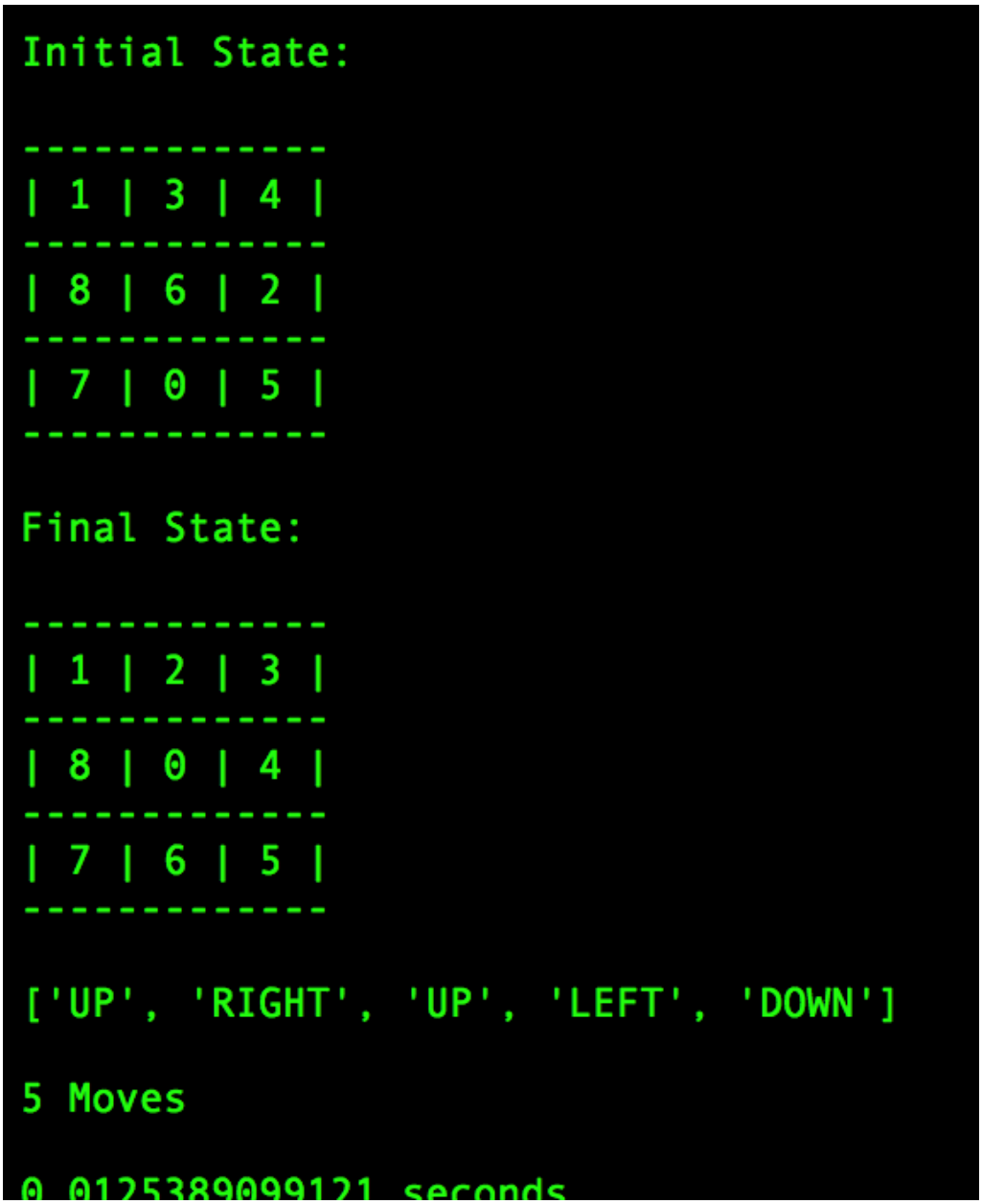
To run the code:
Case 1.a)
python 8-puzzle.py astar 134862705 ntiles

Result:



Case 1.b)
python 8-puzzle.py astar 134862705 manhattan

Result:


Initial State:
1	3	4
8	6	2
7	0	5
Final State:
1	2	3
8	0	4
7	6	5
['UP', 'RIGHT', 'UP', 'LEFT', 'DOWN']
5 Moves
0.012538909912 seconds

Case 2.a)
python 8-puzzle.py astar 281043765 ntiles

Result:

Algorithm did not give any output

Case 2.b)
python 8-puzzle.py astar 281043765 manhattan

Result:

Algorithm did not give any output

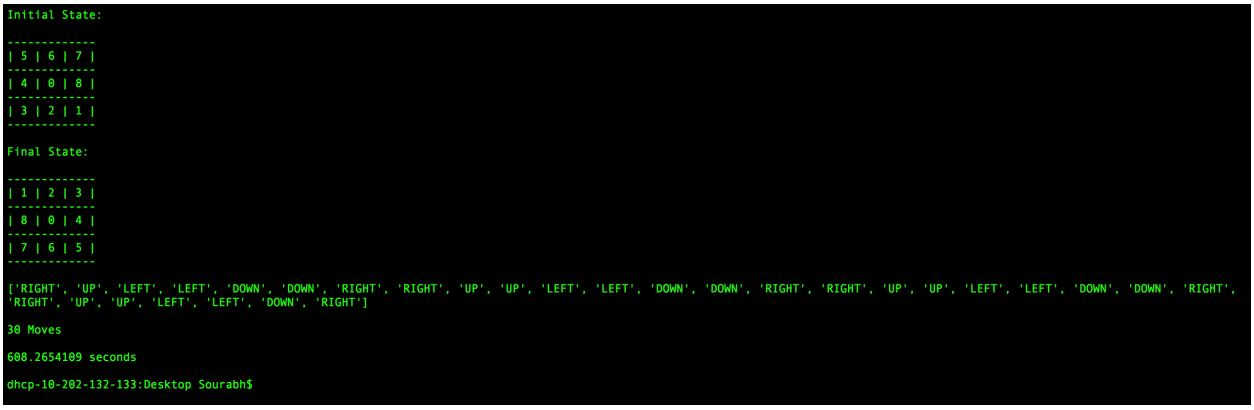
Case 3.a)
python 8-puzzle.py astar 567408321 ntiles

Result:

Algorithm did not give any output

Case 3.b)
python 8-puzzle.py astar 567408321 manhattan

Result:



IDA* Search:

To run the code:
Case 1.a)
python 8-puzzle.py idastar 134862705 ntiles

Result:

Case 1.b)
python 8-puzzle.py idastar 134862705 manhattan

Result:

To run the code:
Case 2.a)
python 8-puzzle.py idastar 281043765 ntiles

Result:

```
Initial State:

-------------
| 2 | 8 | 1 |
-------------
| 0 | 4 | 3 |
-------------
| 7 | 6 | 5 |
-------------

Final State:

-------------
| 1 | 2 | 3 |
-------------
| 8 | 0 | 4 |
-------------
| 7 | 6 | 5 |
-------------

['UP', 'RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'LEFT', 'UP', 'RIGHT', 'DOWN']

9 Moves

0.449097156525 seconds

dhcp-10-202-132-133:Desktop Sourabh$
```

Case 2.b)
python 8-puzzle.py idastar 281043765 manhattan

Result:

```
Initial State:

-------------
| 2 | 8 | 1 |
-------------
| 0 | 4 | 3 |
-------------
| 7 | 6 | 5 |
-------------

Final State:

-------------
| 1 | 2 | 3 |
-------------
| 8 | 0 | 4 |
-------------
| 7 | 6 | 5 |
-------------

['UP', 'RIGHT', 'RIGHT', 'DOWN', 'LEFT', 'LEFT', 'UP', 'RIGHT', 'DOWN']

9 Moves

0.0158090591431 seconds

dhcp-10-202-132-133:Desktop Sourabh$
```

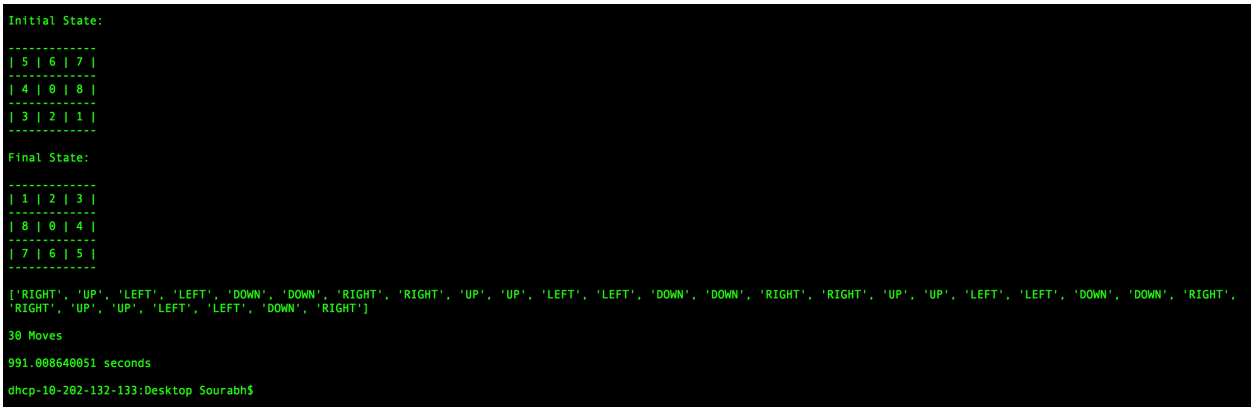
Case 3.a)
python 8-puzzle.py idastar 567408321 ntiles

Result:

Algorithm did not give any output

Case 3.b)
python 8-puzzle.py idastar 567408321 manhattan

Result:



Analysis: Iterative Deepening A* Search uses A* search, but increases the depth incrementally, to avoid the problems caused by A* Algorithm.

Recursion Depth in IDA*

-> For Case 1, with Manhattan heuristic

```
Initial State:

-------------
| 1 | 3 | 4 |
-------------
| 8 | 6 | 2 |
-------------
| 7 | 0 | 5 |
-------------

Final State:

-------------
| 1 | 2 | 3 |
-------------
| 8 | 0 | 4 |
-------------
| 7 | 6 | 5 |
-------------

['UP', 'RIGHT', 'UP', 'LEFT', 'DOWN']

5 Moves

0.00604295730591 seconds
```

Why some programs do not give the answer?
There are certain board states, which do not give an answer. If the goal state lies after such a node state in the tree, then we cannot reach the goal node and hence the algorithm does not give any output

References:

1) Some references are included in the code wherever a snippet/idea was used
2) https://courses.cs.washington.edu/courses/cse473/12au/slides/lect3.pdf
3) https://www.cs.princeton.edu/courses/archive/fall12/cos226/assignments/8puzzle.html
4) http://stackoverflow.com/questions/12033252/iterative-deepening-a-star-ida-to-solve-n-puzzle-sliding-puzzle-in-java