Documentation: AI Programming Assignment 1

8-Puzzle with Search

User Manual

To run the code, open a python IDLE and type the following command
>>> python <filename.py>

Then select the corresponding value from the search menu

```
------- SEARCH RUN MENU ------
1. BFS - Easy
2. BFS - Medium
3. BFS - Hard
4. DFS - Easy
5. DFS - Medium
6. DFS - Hard
7. IDS - Easy
8. IDS - Medium
9. IDS - Hard
10. A* - Easy
11. A* - Medium
12. A* - Hard
13. Greedy BFS - Easy
14. Greedy BFS - Medium
15. Greedy BFS - Hard
16. IDA* - Easy
17. IDA* - Medium
18. IDA* - Hard
Enter what would you like to run:
```

Results for 8 puzzle with Search

1. Using BFS (Breadth-First-Search) algorithm
    a. Easy Case: The BFS takes very less time with the easy case.
    Simulation Result:

```
Enter what would you like to run: 1
Initial Puzzle State:
| 1  3  4 |
| 8  6  2 |
| 7  0  5 |

Goal Puzzle State:
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'RIGHT ->', 'UP ->', 'LEFT ->', 'DOWN ->']
Moves taken:  5
Nodes visited:  40
Max. length of Node List:  54
Time taken:  0.00106
```

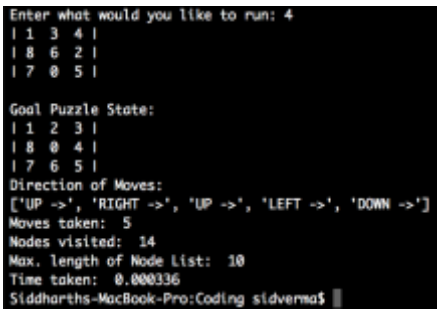
    b. Medium Case: The BFS takes comparatively more time with medium case.
    Simulation Result:

```
Enter what would you like to run: 2
| 2  8  1 |
| 0  4  3 |
| 7  6  5 |

Goal Puzzle State:
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'RIGHT ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'DOWN ->']
Moves taken:  9
Nodes visited:  341
Max. length of Node List:  362
Time taken:  0.007837
Siddharths-MacBook-Pro:Coding sidverma$
```

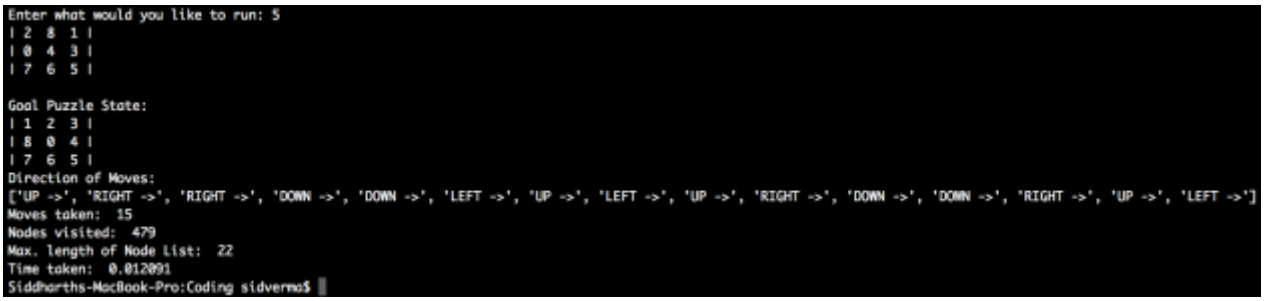
    c. Hard Case: The BFS takes very long time for the hard case (exceeding time limit). This is not because our algorithm is incorrect, but simply due to the fact that the depth of the goal state is really large, and as we increase the depth, the number of nodes to be traversed increases exponentially! Hence it takes a long time to get find the goal state with BFS for a hard input case.

2. Using DFS (Depth-First-Search) algorithm
    DFS runtime is highly dependent on the limit/depth to which we are conducting the DFS. Hence if we have an easy case (i.e number of moves to reach goal state is less), we can safely assume that the depth of the goal node won't be too large. Hence when we run the DFS with a small depth value ~5 we get the following results
    a. Easy Case: The DFS takes very less time with the easy case.
    Simulation Result:



b. Medium Case: The DFS takes comparatively more time with medium case (since the goal node will be at a further depth than that in the easy case, we must increase the search depth for our DFS Algorithm appropriately.

Simulation Result:



c. Hard Case: The DFS takes very long time for the hard case. This is not because our algorithm is incorrect, but simply due to the fact that the depth of the goal state is really large, and as we increase the depth, the number of nodes to be traversed increases exponentially! Hence it takes a long time to get find the goal state with DFS for a hard input case.

3. Using IDS (Iterative-Deepening-Search) algorithm
Similar to the DFS in working, IDS works from a certain depth d. If the goal state is not found in that step, IDS increments the depth to D+1 and re-runs the DFS approach. This process is continued till a goal is found.

a. Easy Case: For the easy case IDS had a run-time depending on the depth level. We are running at a higher depth of ~150 hence should expect to see a greater run time.

Simulation Result:

b. Medium Case: The medium case required more run-time compared to the easy case.

```
Goal Puzzle State:
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOW
N ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'UP ->',
'LEFT ->', 'DOWN ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'LEF
T ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'LEFT ->', 'DOWN ->', 'DOWN ->'
, 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'LEFT ->', 'DOWN ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'LEFT ->', 'UP ->', 'RIGHT ->',
'RIGHT ->', 'DOWN ->', 'LEFT ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'DOWN ->', 'RIGHT ->', 'RIGHT ->', 'UP ->', 'UP ->', 'LEFT ->', 'D
OWN ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'DOWN ->']
Moves taken:  97
Nodes visited:  109283
Max. length of Node List:  133
Time taken:  452.88427
Siddharths-MacBook-Pro:Coding sidverma$
```

c. Hard Case: Since IDS is based entirely on DFS, it takes very long time for the hard case. But since it traverses the depth first, it is possible to find a goal state at a lower depth than an easier state in upper depth if it is further to the right of the tree. In the case below, we see such a situation, where the hard case goal state is reached before the medium case (depicted above).

Simulation Result:

```
Enter what would you like to run: 9
| 5  6  7 |
| 4  0  8 |
| 3  2  1 |

Goal Puzzle State:
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'LEFT ->', 'DOWN ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'UP ->', 'LEFT ->', 'DOWN ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'UP ->', 'LEFT ->', 'DOWN ->', 'DOWN
 ->', 'RIGHT ->', 'UP ->', 'UP ->', 'LEFT ->', 'DOWN ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'UP ->', 'LEFT ->', 'DOWN ->', 'DOWN ->', 'RIGHT ->', 'RIGHT ->', 'UP ->'
, 'UP ->', 'LEFT ->', 'DOWN ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOW
N ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'UP ->',
'LEFT ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP
->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'UP ->', 'LEFT ->', 'LEFT ->', 'DOWN ->', 'DOWN ->', 'RIGHT
->', 'RIGHT ->', 'UP ->', 'LEFT ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'DOWN ->']
Moves taken:  100
Nodes visited:  63516
Max. length of Node List:  132
Time taken:  140.229953
```

4. Using GBFS(Greedy-Best-First-Search)

a. Easy Case: GBFS has a faster rum-time than BFS with an admissible heuristics h1(n) and h2(n)

Simulation Result:

```
Enter what would you like to run: 13
| 1  3  4 |
| 8  6  2 |
| 7  0  5 |

Goal Puzzle State:
Checking state [1, 3, 4, 8, 6, 2, 7, 0, 5]  with direction:  None
Checking state [1, 3, 4, 8, 0, 2, 7, 6, 5]  with direction:  UP ->
Checking state [1, 0, 4, 8, 3, 2, 7, 6, 5]  with direction:  UP ->
Checking state [1, 3, 4, 8, 0, 2, 7, 6, 5]  with direction:  DOWN ->
Checking state [1, 3, 4, 8, 2, 0, 7, 6, 5]  with direction:  RIGHT ->
Checking state [1, 3, 0, 8, 2, 4, 7, 6, 5]  with direction:  UP ->
Checking state [1, 0, 3, 8, 2, 4, 7, 6, 5]  with direction:  LEFT ->
Checking state [1, 2, 3, 8, 0, 4, 7, 6, 5]  with direction:  DOWN ->
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'RIGHT ->', 'UP ->', 'LEFT ->', 'DOWN ->']
Moves taken:  5
Nodes visited:  5
Max. length of Node List:  12
Time taken:  0.000711
Siddharths-MacBook-Pro:Coding sidverma$
```

```
Enter what would you like to run: 13
| 1  3  4 |
| 8  6  2 |
| 7  0  5 |

Goal Puzzle State:
Checking state [1, 3, 4, 8, 6, 2, 7, 0, 5]  with direction:  None
Checking state [1, 3, 4, 8, 0, 2, 7, 6, 5]  with direction:  UP ->
Checking state [1, 0, 4, 8, 3, 2, 7, 6, 5]  with direction:  UP ->
Checking state [1, 3, 4, 8, 0, 2, 7, 6, 5]  with direction:  DOWN ->
Checking state [1, 3, 4, 8, 2, 0, 7, 6, 5]  with direction:  RIGHT ->
Checking state [1, 3, 0, 8, 2, 4, 7, 6, 5]  with direction:  UP ->
Checking state [1, 0, 3, 8, 2, 4, 7, 6, 5]  with direction:  LEFT ->
Checking state [1, 2, 3, 8, 0, 4, 7, 6, 5]  with direction:  DOWN ->
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'RIGHT ->', 'UP ->', 'LEFT ->', 'DOWN ->']
Moves taken:  5
Nodes visited:  5
Max. length of Node List:  12
Time taken:  0.000941
```

b. Medium Case: For the medium case, the GBFS will have a larger run-time, but will still be lesser than that of BFS for the normal case

Simulation Result

```
Checking state [0, 1, 3, 8, 2, 4, 7, 6, 5]  with direction:  UP ->
Checking state [1, 0, 3, 8, 2, 4, 7, 6, 5]  with direction:  RIGHT ->
Checking state [1, 2, 3, 8, 0, 4, 7, 6, 5]  with direction:  DOWN ->
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'RIGHT ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'DOWN ->']
Moves taken:  9
Nodes visited:  156
Max. length of Node List:  126
Time taken:  0.065799
```

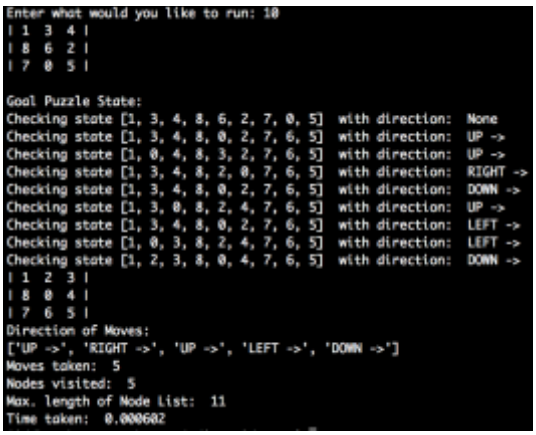
c. Hard Case: GBFS will have a much larger running time for the hard case compared to other cases, but yet again, it will be faster then regular BFS and DFS algorithms

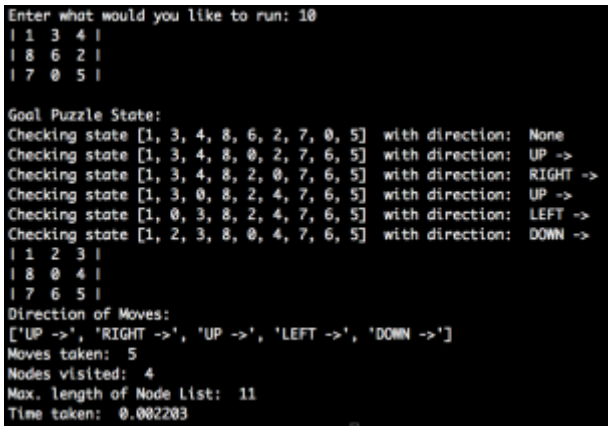
Simulation Result:

5. Using A* (A – Star Heuristic Search) algorithm

a. Easy Case: The A* algorithm uses a heuristic function f(n) = g(n) + h(n) and performs much better than BFS, DFS, IDS and Greedy BFS.

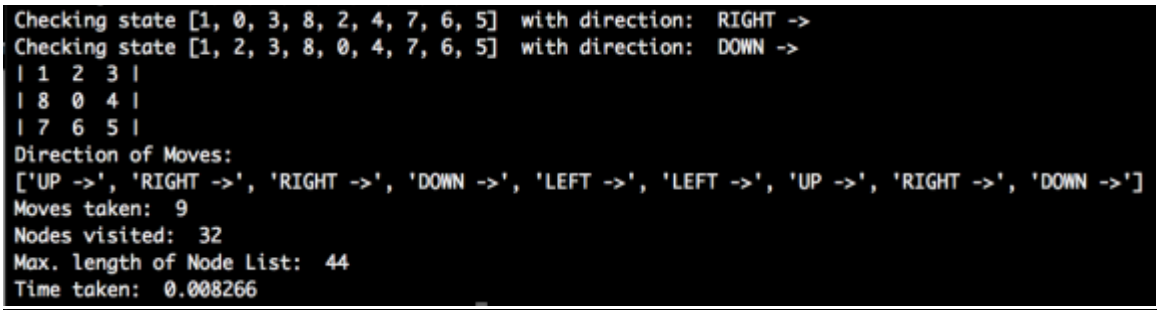
Simulation Result:

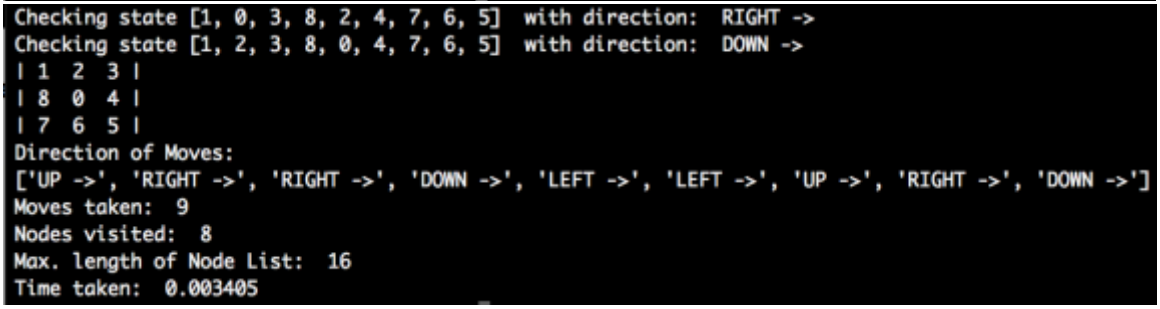




b. Medium Case: A* takes more time for the medium case, but still lesser than all previous search algorithms discussed.

Simulation Result:





c. Hard Case: A* exceeds time limit with heuristic h1(n) for the hard case as compared to Greedy BFS. With h2(n) it performs better than Greedy BFS

Simulation Result:

```
Checking state [1, 2, 3, 8, 4, 5, 7, 6, 0]  with direction:  RIGHT ->
Checking state [1, 2, 3, 8, 4, 0, 7, 6, 5]  with direction:  UP ->
Checking state [1, 2, 3, 8, 0, 4, 7, 6, 5]  with direction:  LEFT ->
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'LEFT ->', 'DOWN ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'UP ->', 'LEFT ->', 'DOWN ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'UP ->', 'LEFT ->', 'DOWN ->', 'RIGH
T ->', 'RIGHT ->', 'UP ->', 'LEFT ->', 'LEFT ->', 'DOWN ->', 'RIGHT ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'UP ->', 'LEFT ->', 'DOWN ->', 'LEF
T ->', 'UP ->', 'RIGHT ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'LEFT ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'LEFT ->', 'UP ->', 'RIG
HT ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'LE
FT ->', 'DOWN ->', 'RIGHT ->', 'RIGHT ->', 'UP ->', 'LEFT ->']
Moves taken:  66
Nodes visited:  300
Max. length of Node List:  259
Time taken:  1.892483
```

6. Using IDA*(Iterative-Deepening-A –Star Heuristic Search) algorithm
   a. Easy Case: The different results for the two heuristics is mentioned below

Simulation Result:

```
Checking state [1, 3, 0, 8, 2, 4, 7, 6, 5]  with direction:  UP ->
Checking state [1, 0, 3, 8, 2, 4, 7, 6, 5]  with direction:  LEFT ->
Checking state [1, 2, 3, 8, 0, 4, 7, 6, 5]  with direction:  DOWN ->
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'RIGHT ->', 'UP ->', 'LEFT ->', 'DOWN ->']
Moves taken:  5
Nodes visited:  5
Max. length of Node List:  12
Time taken:  0.0014
```

```
Enter what would you like to run: 16
| 1  3  4 |
| 8  6  2 |
| 7  0  5 |

Goal Puzzle State:
Checking state [1, 3, 4, 8, 6, 2, 7, 0, 5]  with direction:  None
Checking state [1, 3, 4, 8, 0, 2, 7, 6, 5]  with direction:  UP ->
Checking state [1, 3, 4, 8, 2, 0, 7, 6, 5]  with direction:  RIGHT ->
Checking state [1, 3, 0, 8, 2, 4, 7, 6, 5]  with direction:  UP ->
Checking state [1, 0, 3, 8, 2, 4, 7, 6, 5]  with direction:  LEFT ->
Checking state [1, 2, 3, 8, 0, 4, 7, 6, 5]  with direction:  DOWN ->
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'RIGHT ->', 'UP ->', 'LEFT ->', 'DOWN ->']
Moves taken:  5
Nodes visited:  4
Max. length of Node List:  11
Time taken:  0.002889
```

   b. Medium Case: The results for the medium case for IDA* are shown below

Simulation Result:

```
Checking state [0, 1, 3, 8, 2, 4, 7, 6, 5]  with direction:  UP ->
Checking state [1, 0, 3, 8, 2, 4, 7, 6, 5]  with direction:  RIGHT ->
Checking state [1, 2, 3, 8, 0, 4, 7, 6, 5]  with direction:  DOWN ->
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'RIGHT ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'DOWN ->']
Moves taken:  9
Nodes visited:  22
Max. length of Node List:  37
Time taken:  0.007303
```

```
Checking state [0, 1, 3, 8, 2, 4, 7, 6, 5]  with direction:  UP ->
Checking state [1, 0, 3, 8, 2, 4, 7, 6, 5]  with direction:  RIGHT ->
Checking state [1, 2, 3, 8, 0, 4, 7, 6, 5]  with direction:  DOWN ->
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['UP ->', 'RIGHT ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'DOWN ->']
Moves taken:  9
Nodes visited:  8
Max. length of Node List:  16
Time taken:  0.005166
```

   c. Hard Case: The results for the hard case are as shown. For h1(n), the run exceeded the time limit.

Simulation Results:

```
Checking state [1, 0, 3, 8, 2, 4, 7, 6, 5]  with direction:  LEFT ->
Checking state [1, 2, 3, 8, 0, 4, 7, 6, 5]  with direction:  DOWN ->
| 1  2  3 |
| 8  0  4 |
| 7  6  5 |
Direction of Moves:
['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 ->', 'U
P ->', 'LEFT ->', 'LEFT ->', 'UP ->', 'RIGHT ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'LEFT ->', 'DOWN ->', 'RIGHT ->', 'RIGHT ->', 'UP
 ->', 'LEFT ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'LEFT ->', 'DOWN ->', 'RIGHT ->', 'DOWN ->', 'LEFT ->', 'UP ->', 'UP ->', 'RIGHT ->', 'DOWN -
>', 'LEFT ->', 'DOWN ->', 'RIGHT ->', 'UP ->', 'UP ->', 'LEFT ->', 'DOWN ->']
Moves taken:  60
Nodes visited:  206
Max. length of Node List:  173
Time taken:  1.670813
```