



Faculty of Engineering

Computer and Communications Program

SEARCH ALGORITHMS FOR 8 PUZZLE GAME

Language: Python 3.6

Presented by:

Mohamed Khaled 4783

Zeyad Mahmoud Fathy 4988

Abdallah El-Sayed 4883

Abdelrahman Ashraf 4912

Description: Some search solutions for 8 puzzle game, implementing these algorithms:

An instance of the 8-puzzle game consists of a board holding 8 distinct movable tiles, plus an empty space. For any such board, the empty space may be legally swapped with any tile horizontally or vertically adjacent to it. In this assignment, the blank space is going to be represented with the number 0.

Given an initial state of the board, the search problem is to find a sequence of moves that transitions this state to the goal state, that is, the configuration with all tiles arranged in ascending order 0,1,2,3,4,5,6,7,8 .

The search space is the set of all possible states reachable from the initial state. The blank space may be swapped with a component in one of the four directions 'Up', 'Down', 'Left', 'Right', one move at a time.

It's required to implement the game solution using different algorithms, BFS, DFS, A* (Manhattan Distance & Euclidean Distance).

Implemented Algorithms:

1. BFS

The algorithm idea is to horizontally explore the nodes within the tree until we reach the goal.

In general its search depth value is fine compared to other algorithms as DFS.

The main defect is the long running time because BFS is an uninformed search so we have no information about the path that may help us reach the goal.

BFS is implemented using Queue data structure.

2. DFS

The algorithm idea is to deeply explore the nodes within the tree until we reach the goal.

In general its search depth value is bad due to searching until end of depth of each node in tree.

Same as BFS the main defect is the long running time because DFS is an uninformed search so we have no information about the path that may help us reach the goal.

DFS is implemented using Stack data structure.

3. A*

The algorithm idea is to work on finding the goal with putting into consideration the Heuristics that would help reaching the goal in shortest path.

A* is an informed search method, so we have some knowledge (Heuristics) that lead us to reach the goal in the shortest and fastest way so its running time is a lot better than any uninformed search method.

A* is implemented using priority queue data structure.

For the Heuristics we compute it using any of the following 2 methods (Both are implemented):

1)Manhattan Distance:

It is the sum of absolute values of difference's in the goal's x and y coordinates and the current cell's x and y coordinates respectively.

2)Euclidean Distance:

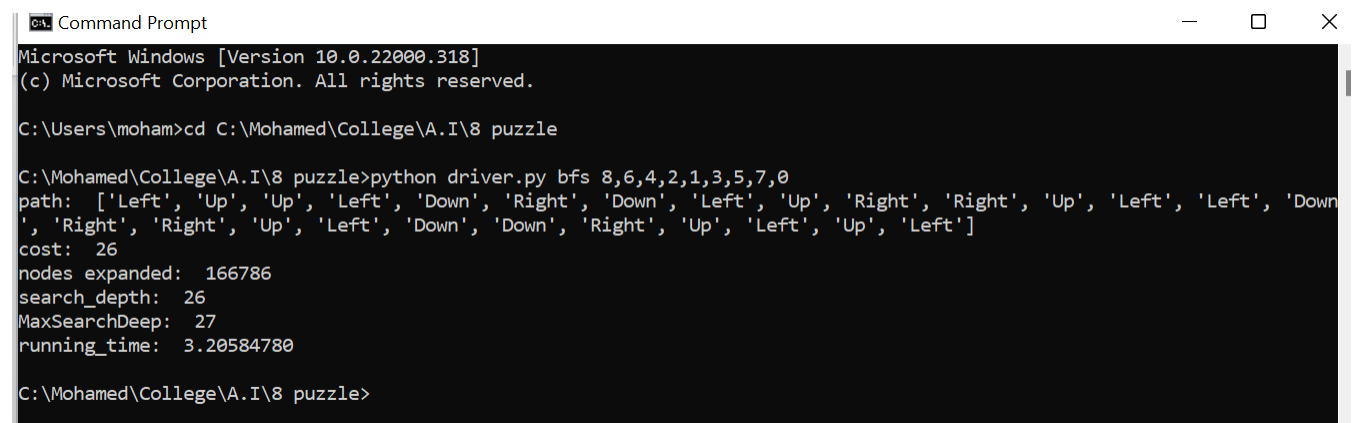
It is the distance between the current cell and the goal cell using the distance formula.

Test Cases:

1)BFS

Input : python driver.py bfs 8,6,4,2,1,3,5,7,0

Output:



```
Microsoft Windows [Version 10.0.22000.318]
(c) Microsoft Corporation. All rights reserved.

C:\Users\moham>cd C:\Mohamed\College\A.I\8 puzzle

C:\Mohamed\College\A.I\8 puzzle>python driver.py bfs 8,6,4,2,1,3,5,7,0
path: ['Left', 'Up', 'Up', 'Left', 'Down', 'Right', 'Down', 'Left', 'Up', 'Right', 'Right', 'Up', 'Left', 'Left', 'Down', 'Right', 'Right', 'Up', 'Left', 'Down', 'Down', 'Right', 'Up', 'Left', 'Up', 'Left']
cost: 26
nodes expanded: 166786
search_depth: 26
MaxSearchDeep: 27
running_time: 3.20584780

C:\Mohamed\College\A.I\8 puzzle>
```

2)DFS

Input : python driver.py dfs 1,2,5,3,4,8,6,7,0

Output:

```
Command Prompt
Microsoft Windows [Version 10.0.22000.318]
(c) Microsoft Corporation. All rights reserved.

C:\Users\moham>cd C:\Mohamed\College\A.I\8 puzzle

C:\Mohamed\College\A.I\8 puzzle>python driver.py dfs 1,2,5,3,4,8,6,7,0
path: ['Up', 'Up', 'Left', 'Left']
cost: 4
nodes expanded: 181437
search_depth: 4
MaxSearchDeep: 66126
running_time: 2.77127880

C:\Mohamed\College\A.I\8 puzzle>
```

3)A*

Input : python driver.py ast 1,2,5,3,4,8,6,7,0

Output:

```
C:\Mohamed\College\A.I\8 puzzle>python driver.py ast 1,2,5,3,4,8,6,7,0
path: ['Up', 'Up', 'Left', 'Left']
cost: 4
nodes expanded: 4
search_depth: 4
MaxSearchDeep: 4
running_time: 0.00018620

C:\Mohamed\College\A.I\8 puzzle>_
```

Conclusion:

1. The cost path is close in all algorithms except for DFS it has very large depth value
2. In both uninformed search algorithms (BFS, DFS), the number of expanded nodes is too large, while for informed search algorithm (A*) the nodes expanded is much smaller, and that's why Heuristics are powerful.
3. The running time truly shows how informed search is much better than uninformed search, as the time difference between BFS,DFS (that are close in running time), and A* search is great, so that's why A* is the best algorithm to solve this game as it expand least nodes and reach the goal in the fastest way.

Finally concerning the different Heuristics calculating ways, in most cases Manhattan is faster and expands less nodes to reach goal.