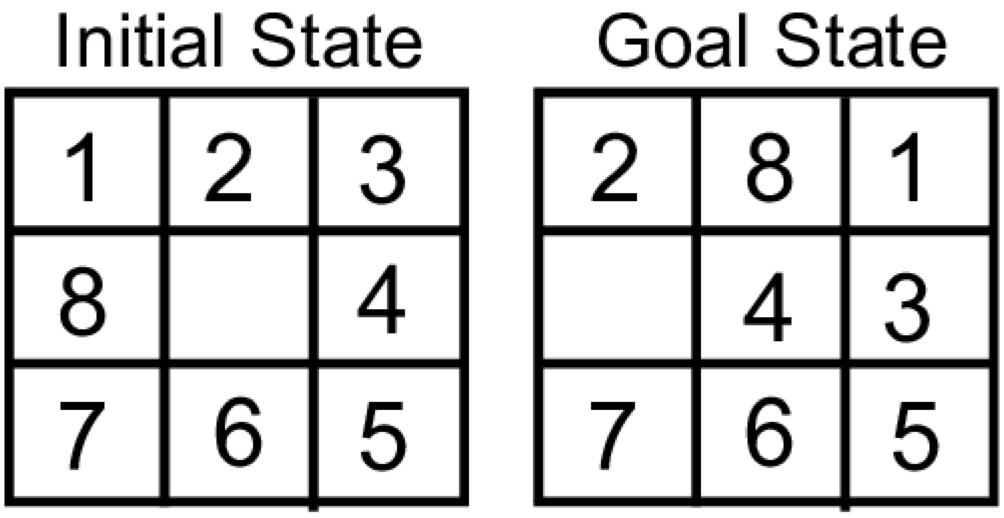
CS451 ASSIGNMENT 1

ÖMER FARUK DALĞIN

S012251

1.INTRODUCTION

In this assignment, I coded Python for N-Puzzle problem. N-Puzzle is a popular puzzle problem consisting of N tiles where N can be 8, 15, 24 and so on (M2-1). In this assignment, N = 8. The puzzle is divided into square root of N+1 rows and columns (e.g. 15-Puzzle will have 4 rows and 4 columns, and an 8-Puzzle will have 3 rows and 3 columns). The puzzle consists of N tiles and one empty space where the tiles can be moved. Start and Goal configurations(also called state) of the puzzle are provided. The puzzle can be solved by moving the tiles one by one in the single empty space and thus achieving the Goal configuration. The structure of the problem is below as an example(for N=8).



2.ALGORITHMS

I tested 4 different algorithms to solve the N-puzzle problem. These are;

- *BFS (Breadth-First Search) : **Breadth-first search (BFS)** is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.*

- *DFS (Depth-First Search) : **Depth-first search (DFS)** is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.*

- *UCS (Uniform Cost Search) : **Uniform Cost Search** is an algorithm used to move around a directed weighted search space to go from a start node to one of the ending nodes with a minimum cumulative cost.*

- *A* (A star Search) : **A*search** is a graph traversal and path search algorithm, which is often used in many fields of computer science due to its completeness, optimality, and optimal efficiency. It can be seen as an extension of Dijkstra's algorithm. A* achieves better performance by using heuristics to guide its search.*

3.IMPLEMENTATION DETAILS

I was provided a code baseline, I had to be strict on this baseline. I did not use any other implementation for this assignment. Node and Graph implementations were given to me to deeply understand what they actually do, and I was not allowed to change these files. Also I had main.py file to run codes. It makes possible to select the algorithm that the user wants to apply to the problem. Before that, you need to enter the grid size as 8, and it takes 2 states from console for n-puzzle, first one is initial state and second is goal state. After entering what is required, the program will work with the algorithm you have chosen and will give you the following results(for proper inputs):

Match found!

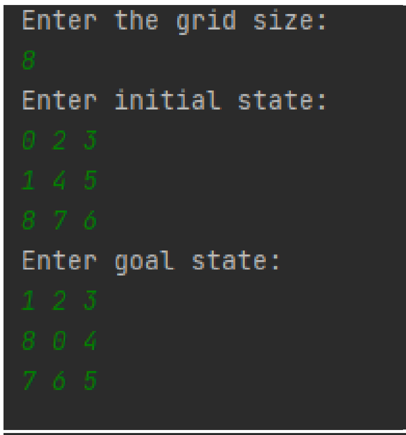
Num. of visited nodes: …

Depth of graph: …

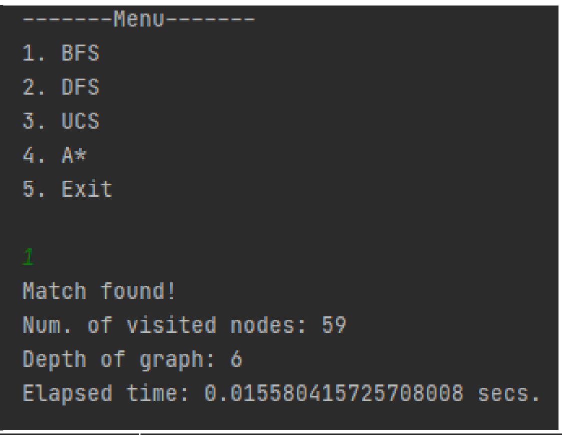
Elapsed time: … secs

4.RESULTS

You can see the grid size, initial state and goal state in the screenshots I added below. The results obtained after testing with these values are as follows for each algorithms:

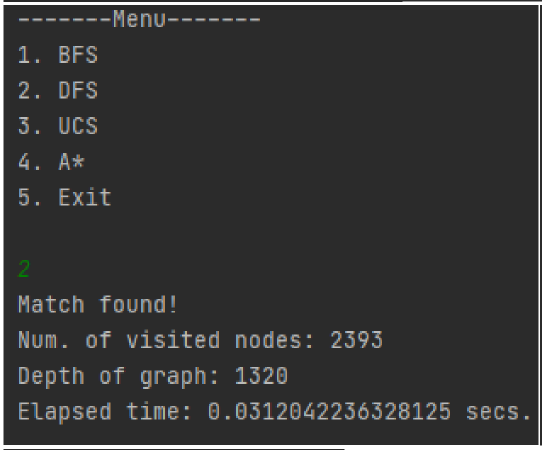


```
Enter the grid size:
8
Enter initial state:
0 2 3
1 4 5
8 7 6
Enter goal state:
1 2 3
8 0 4
7 6 5
```



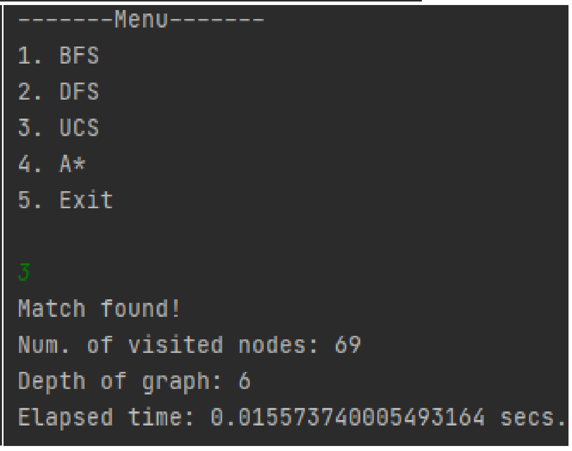
```
-------Menu-------
1. BFS
2. DFS
3. UCS
4. A*
5. Exit

1
Match found!
Num. of visited nodes: 59
Depth of graph: 6
Elapsed time: 0.015580415725708008 secs.
```



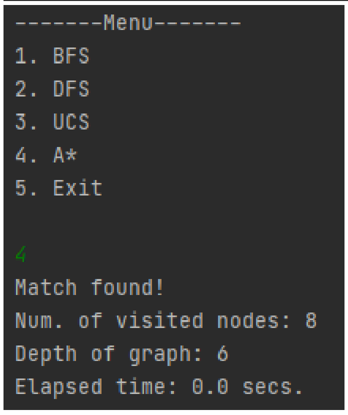
```
-------Menu-------
1. BFS
2. DFS
3. UCS
4. A*
5. Exit

2
Match found!
Num. of visited nodes: 2393
Depth of graph: 1320
Elapsed time: 0.0312042236328125 secs.
```



```
-------Menu-------
1. BFS
2. DFS
3. UCS
4. A*
5. Exit

3
Match found!
Num. of visited nodes: 69
Depth of graph: 6
Elapsed time: 0.015573740005493164 secs.
```



```
-------Menu-------
1. BFS
2. DFS
3. UCS
4. A*
5. Exit

4
Match found!
Num. of visited nodes: 8
Depth of graph: 6
Elapsed time: 0.0 secs.
```

5.CONCLUSION

I studied the N-puzzle problem and learned how algorithms work, which ones are faster, and their contents. The A* search algorithm was the best and the fastest one. I think it's a search algorithm that is used frequently in many fields of computer science. DFS stood out as the slowest running algorithm in this test. DFS is trying to reach the deepest somehow by diving into the bottom and it was not very efficient for this problem.