8-Puzzle Solver

Omar Mohamed Iraqy (7439)

Jomana Ehab Abd El Aziz (9178)

Ahmed Mohamed Younis(7745)

1. Introduction:

- 8-Puzzle is a game where you have a 3x3 grid containing numbers from 0 to 8 shuffled with a single tile missing.
- The game Objective is to sort all the numbers in fewest moves as possible.
- Objective of the Assignment: To compare the performance of BFS, DFS, and A* algorithms in solving the 8-puzzle game.

2. Background:

2.1. Breadth-First Search (BFS):

- BFS is an uninformed search algorithm that explores all the neighbour nodes at the present depth prior to moving on to the nodes at the next depth level.
- Uses Queue to achieve visiting all nodes in same level first.
- In the context of the 8-puzzle game, BFS would explore all possible moves from the initial state before moving to the next level of moves.
- BFS guarantees the shortest path to the goal state but can be computationally expensive in terms of memory usage and time, especially for large state spaces.

2.2 Depth-First Search (DFS):

- DFS is an uninformed search algorithm that explores as far as possible along each branch before backtracking.
- Uses Stack to achieve visiting the deepest node first.
- In the context of the 8-puzzle game, DFS would explore one branch of moves as deeply as possible before exploring other branches.
- DFS does not guarantee the shortest path to the goal state and can get stuck in infinite loops or take a long time to find a solution if the search space is large and the depth limit is not set. But it works better at memory management as the amount of nodes in the memory is way less than BFS.

2.3. A* Algorithm:

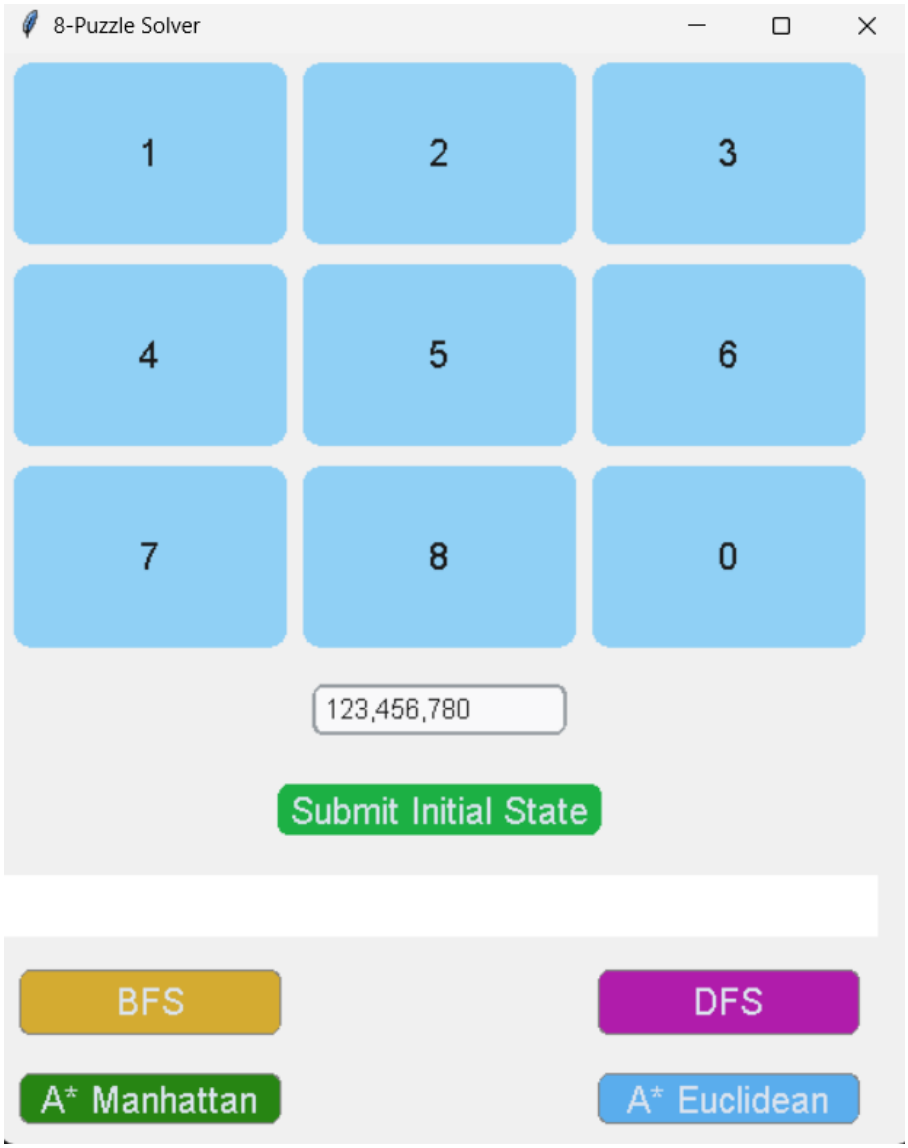
- A* is an informed search algorithm that uses a heuristic function to estimate the cost of reaching the goal state from the current state.
- Uses Heap/Priority Queue so that it visits the node with minimum cost first.
- In the context of the 8-puzzle game, A* would use a heuristic function to estimate the cost of reaching the goal state from each possible move and then choose the move with the lowest estimated cost.

- A* is guaranteed to find the shortest path to the goal state if the heuristic function is admissible.
- A* is generally more efficient than BFS and DFS in terms of time and memory usage, especially for large state spaces, but the effectiveness of A* depends heavily on the quality of the heuristic function used.

3. Results:

For the comparison between we will use a single initial state and run all the algorithms on it to get comparable data.

The state we will work with:



We ran all the algorithms for the same state, these are the results we got:

3.1. Breadth-First Search (BFS):

Number of nodes visited = 80618

Max Number of nodes in memory = 110227

Number of nodes in Path = 23

Time Taken by Algorithm to find path = 1.617182970046997 seconds

3.2 Depth-First Search (DFS):

Number of nodes visited = 14306

Max Number of nodes in memory = 25430

Number of nodes in Path = 14203

Time Taken by Algorithm to find path = 0.31450891494750977 seconds

3.3. A* (Manhattan) :

Number of nodes visited by A* = 5402

Max Number of nodes in memory = 8468

Number of nodes in Path by A* = 23

Cost seen by A* = 210

Time Taken by Algorithm to find path = 4.3261637687683105 seconds

3.4. A* (Euclidean) :

Number of nodes visited by A* = 5797

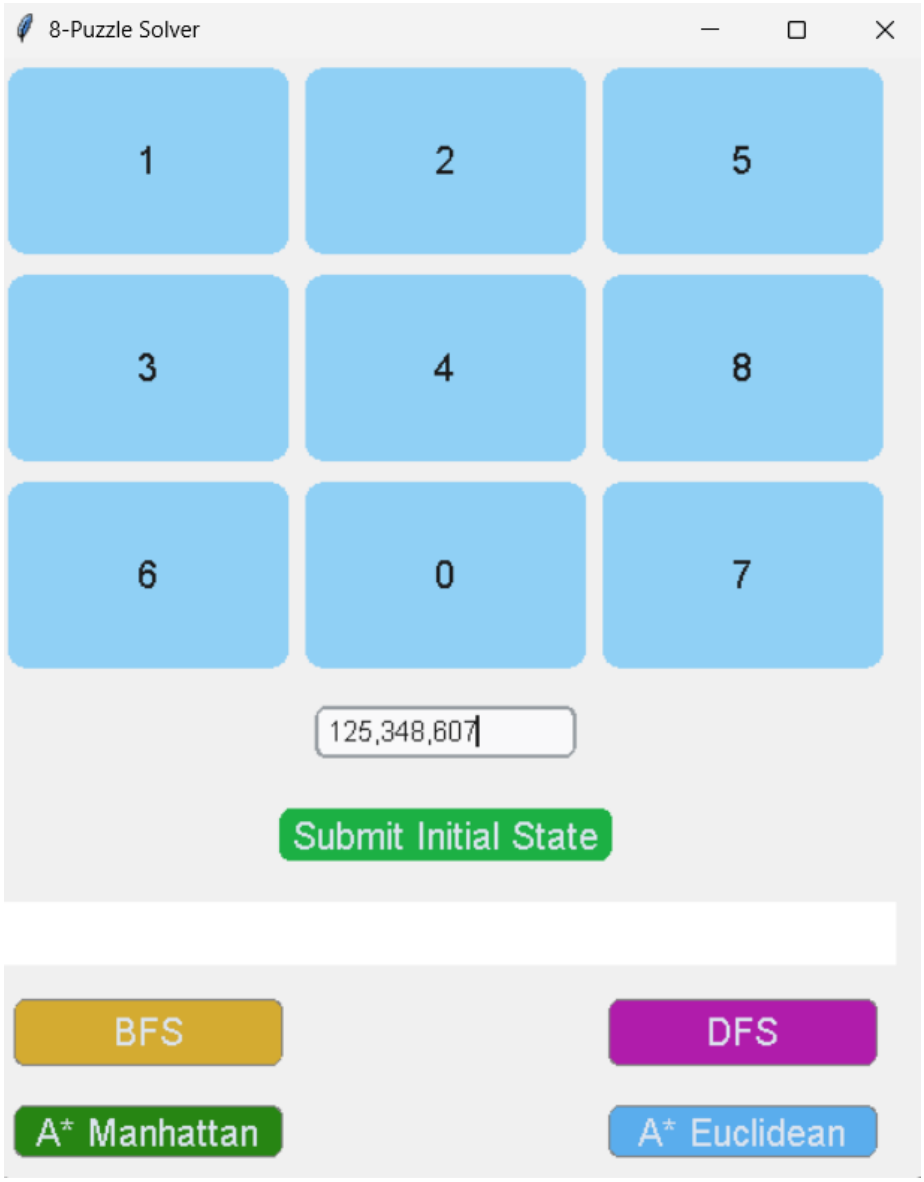
Max Number of nodes in memory = 9050

Number of nodes in Path by A* = 23

Cost seen by A* = 179.74450397542893

Time Taken by Algorithm to find path = 4.8718321323394775 seconds

Another initial state:



3.1. Breadth-First Search (BFS):

Number of nodes visited = 40

Max Number of nodes in memory = 69

Number of nodes in Path = 6

Time Taken by Algorithm to find path = 0.004019260406494141 seconds

3.2 Depth-First Search (DFS):

Number of nodes visited = 125245

Max Number of nodes in memory = 196973

Number of nodes in Path = 102328

Time Taken by Algorithm = 5.266038656234741

3.3. A* (Manhattan) :

Number of nodes visited by A* = 12

Max Number of nodes in memory = 24

Number of nodes in Path by A* = 6

Cost by A* = 28

Time Taken by Algorithm = 0.03667092323303223

3.4. A* (Euclidean) :

Number of nodes visited by A* = 12

Max Number of nodes in memory = 24

Number of nodes in Path by A* = 6

Cost by A* = 25.30056307974577

Time Taken by Algorithm = 0.001043558120727539

4. Conclusion:

By looking at the results we can see that BFS and A* algorithms are optimal while DFS is not optimal and can be very unpredictable in large state-space situations.

We can also notice that A* search uses a lot less memory than BFS.