LAB ONE – EIGHT PUZZLE SOLVER

Team Members	**Date: 29/10/2024**
▪ محمد شريف فتحى يوسف جليله	ID: 21011151
▪ محمد اشرف جابر عبده	ID: 21011082

1. Problem Statement

The 8-puzzle consists of a 3x3 grid with eight numbered tiles and one blank space. The objective is to arrange the tiles in ascending order from 0 to 8 (where 0 represents the blank) by sliding tiles into the blank space, moving from an initial configuration to the goal state, which is 012345678.

Given an initial state of the board, this program finds a sequence of moves to transition from the initial state to the goal state. The solution uses four different search algorithms to explore the search space of all possible tile configurations.

2. Data Structures and Algorithms

1) **Breadth-First Search (BFS)** - An uninformed search algorithm exploring nodes level by level. The breadth-first search algorithm required the usage of two lists, `visited` and `expanded`, which keep track of the visited nodes, and the nodes expanded respectively, as well as a double-ended queue, `frontier`, which is used to keep track of the children of a node.
2) **Depth-First Search (DFS)** - An uninformed search algorithm exploring nodes by depth. The depth-first search algorithm required the usage of two lists, `visited` and `expanded`, which keep track of the visited nodes, and the nodes expanded respectively, as well as a stack, `frontier`, which is used to find the next node to visit.
3) **Iterative Deepening Depth-First Search (IDDFS)** - An uninformed search combining DFS with breadth control. The iterative deepening depth-first search algorithm required the usage of two lists, `visited` and `expanded`, which keep track of the visited nodes, and the nodes expanded respectively, as well as a stack, `frontier`, which is used to find the next node to visit.
4) **A*** - An informed search algorithm using heuristic functions for optimization. The A* search algorithm required the usage of two lists, `visited` and `expanded`, which keep track of the visited nodes, and the nodes expanded respectively, as well as a binary heap, `frontier`, which acts as priority queue to find the next node to visit.
5) Each `solve()` function implemented in the algorithms classes returns a dictionary containing the required outputs
6) To keep track of the moves made, the directions that the empty tile can move are kept in a list, where each element is a tuple.

3. Heuristics Comparison

The A* algorithm uses two heuristics for comparison:

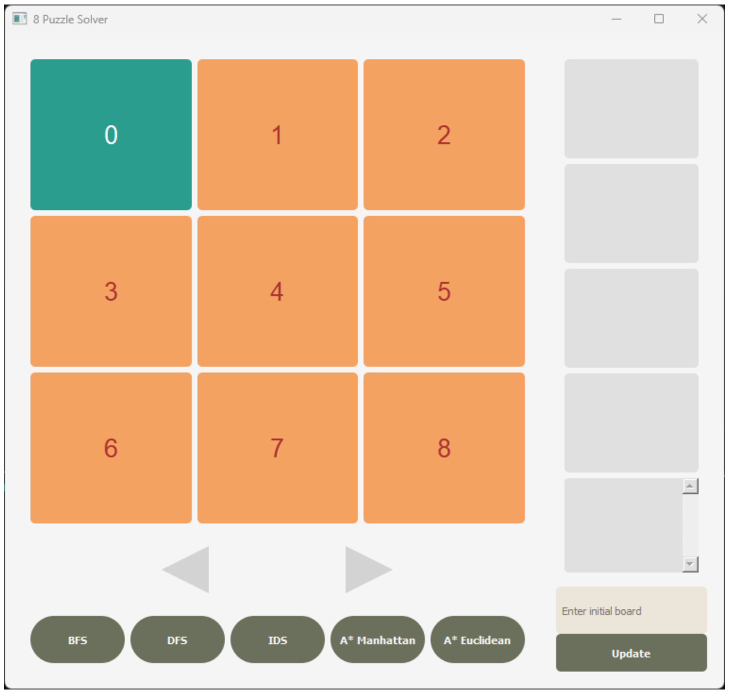
- **Manhattan Distance**: The sum of the absolute differences between the current and goal positions of tiles.
- **Euclidean Distance**: The straight-line distance between the current and goal positions of tiles.

	Nodes Expanded	
Initial Board	**Manhattan**	**Euclidean**
806547231	10670	54435
641302758	206	218
158327064	32	37
328451670	59	60
35428617	24	24
725310648	232	255
125340678	7	7
130672548	1993	3009
531428760	884	1202
42537186	222	291
312645780	8	8

We compared the number of nodes expanded using both heuristics and the Euclidean Distance heuristic expanded more nodes, which can be seen through the table provided. This is because the Manhattan Distance heuristic is more consistent with the grid structure constraints, while the Euclidean Distance heuristic is based on finding the straight-line distance between two points, which may not directly correlate with the actual number of moves needed, especially since movement is constrained to adjacent tiles, providing an overly optimistic estimate of the path cost.

4. Graphical User Interface

The Graphical User Interface was made using the PyQt5 library, where the main window is divided into two widgets horizontally, `left_widget` and `right_widget`.
The left widget itself is divided into two widgets vertically, with one being the widget containing the grid for the puzzle, which displays every move made until the goal state is reached, and the other containing all the buttons. The two arrow buttons are to analyse each move made by going backwards or forwards after the goal state is reached. Algorithms are each represented using the five buttons at the bottom.



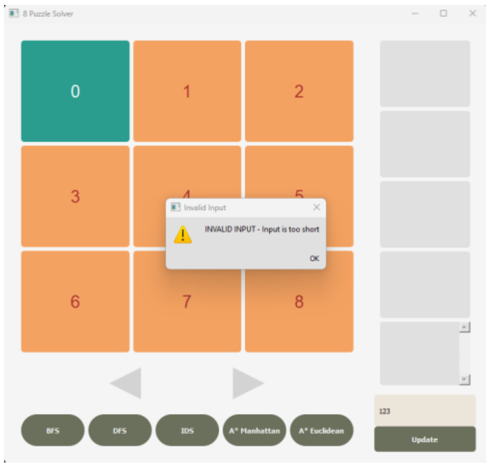
The right widget, as well, is divided vertically, where the top widget is where the output is displayed, the scrollable widget being reserved for the moves, as it can be quite long, and the bottom widget is for the input of the initial state.

The initial state is written into the text field, and when the Update button is pressed, the input is first validated, and would output an alert message if the input string:

- is shorter or longer than 9 digits long
- contains non-numeric characters
- contains repeated digits
- contains numbers outside the set 0,1,2,3,4,5,6,7,8
- is unsolvable (this is checked through the number of inversions, where an odd number of inversions deems the board unsolvable)

5. Sample Runs

<u>Test Case 01</u>
Input: ‘123’
Expected Output: ‘INVALID INPUT - Input is too short’



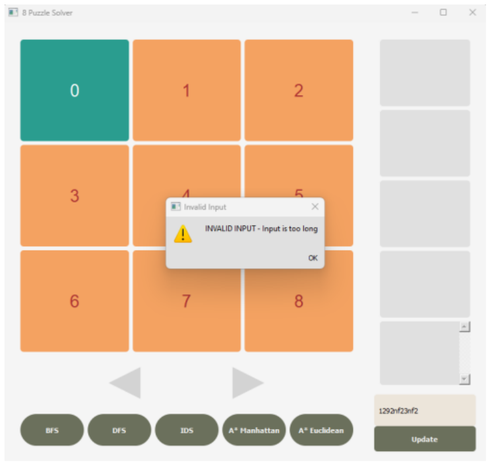
Invalid Input

INVALID INPUT - Input is too short

OK

123

<u>Test Case 02</u>
Input: ‘1292nf23nf2’
Expected Output: ‘INVALID INPUT - Input is too long’



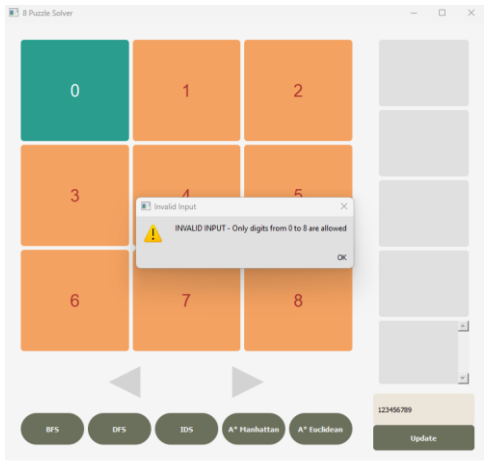
Invalid Input

INVALID INPUT - Input is too long

OK

1292nf23nf2

<u>Test Case 03</u>
Input: ‘123456789’
Expected Output: ‘INVALID INPUT - Only digits from 0 to 8 are allowed’



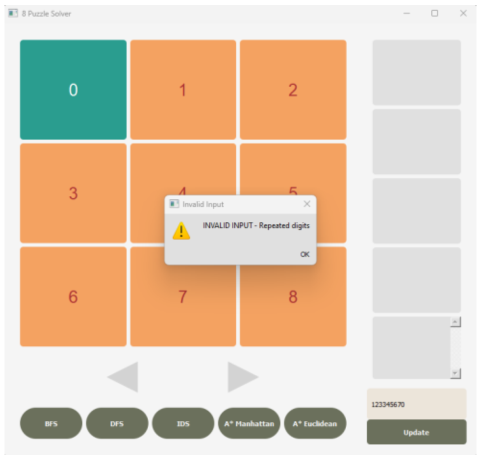
Invalid Input

INVALID INPUT - Only digits from 0 to 8 are allowed

OK

123456789

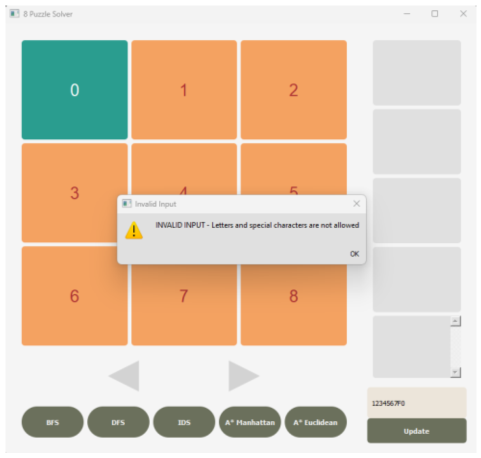
<u>Test Case 04</u>
Input: ‘123345670’
Expected Output: ‘INVALID INPUT - Repeated digits’



123345670

INVALID INPUT - Repeated digits

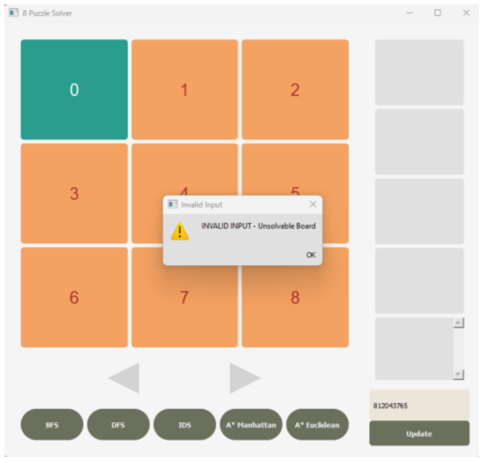
<u>Test Case 05</u>
Input: ‘1234567F0’
Expected Output: ‘INVALID INPUT - Letters and special characters are not allowed’



1234567F0

INVALID INPUT - Letters and special characters are not allowed

<u>Test Case 06</u>
Input: ‘812043765’
Expected Output: ‘INVALID INPUT - Unsolvable Board’



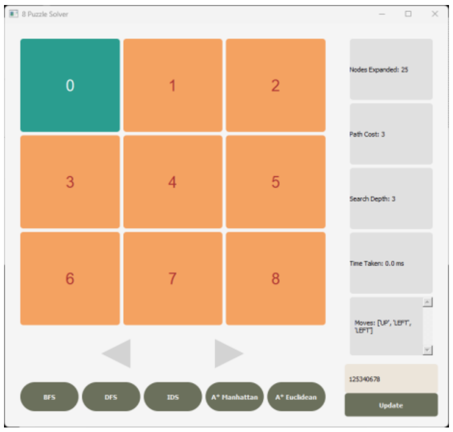
812043765

INVALID INPUT - Unsolvable Board

Test Case 07

Input: ‘125340678’

Breadth-First Search



Nodes Expanded: 25

Path Cost: 3

Search Depth: 3

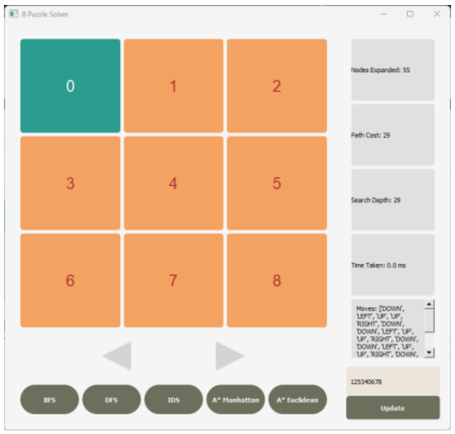
Time Taken: 0.0 ms

Moves: ['UP', 'LEFT', 'LEFT']

125340678

Update

Depth-First Search



Nodes Expanded: 55

Path Cost: 29

Search Depth: 29

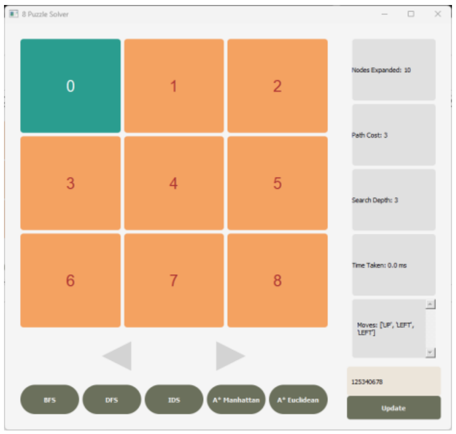
Time Taken: 0.0 ms

Moves: ['DOWN', 'LEFT', 'UP', 'UP', 'RIGHT', 'DOWN', 'DOWN', 'LEFT', 'UP', 'UP', 'RIGHT', 'DOWN', 'DOWN', 'LEFT', 'UP', 'UP', 'RIGHT', 'DOWN',

125340678

Update

IDDFS



Nodes Expanded: 10

Path Cost: 3

Search Depth: 3

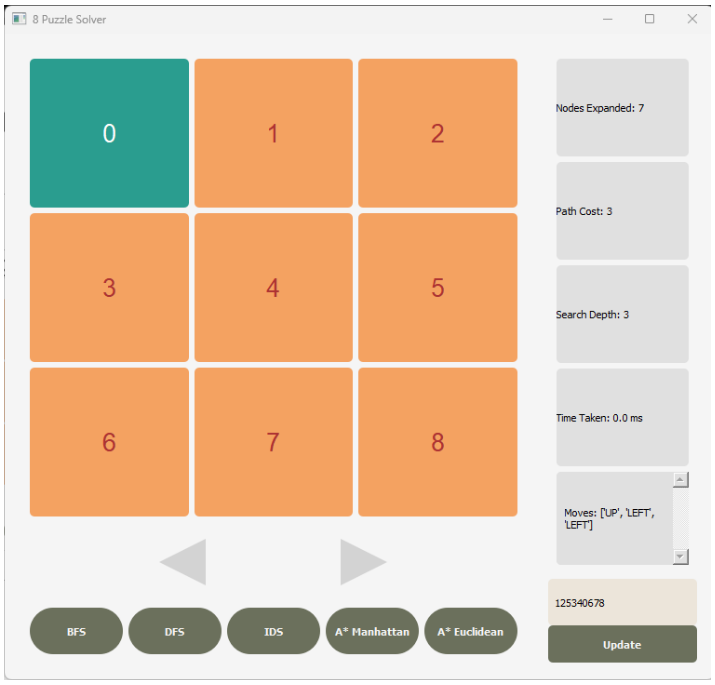
Time Taken: 0.0 ms

Moves: ['UP', 'LEFT', 'LEFT']

125340678

Update

A* - Manhattan



Nodes Expanded: 7

Path Cost: 3

Search Depth: 3

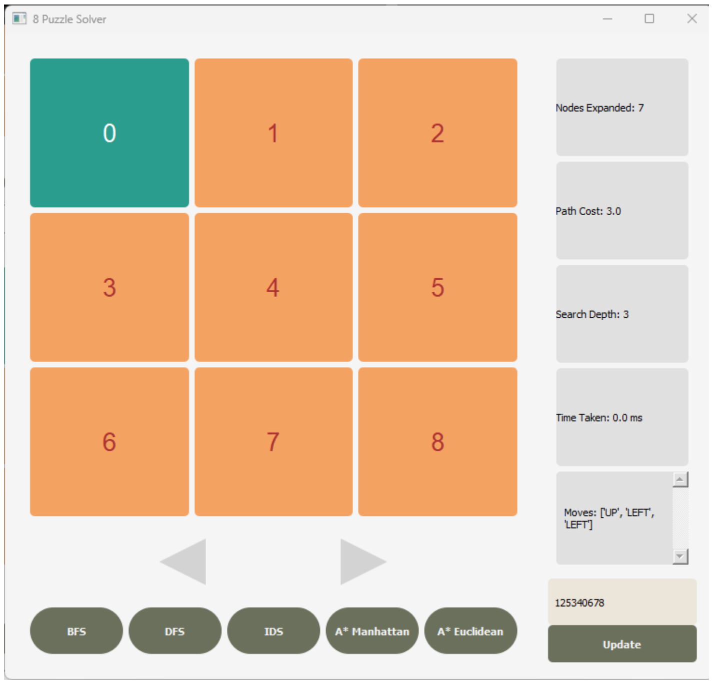
Time Taken: 0.0 ms

Moves: ['UP', 'LEFT', 'LEFT']

125340678

Update

A* - Euclidean



Nodes Expanded: 7

Path Cost: 3.0

Search Depth: 3

Time Taken: 0.0 ms

Moves: ['UP', 'LEFT', 'LEFT']

125340678

Update