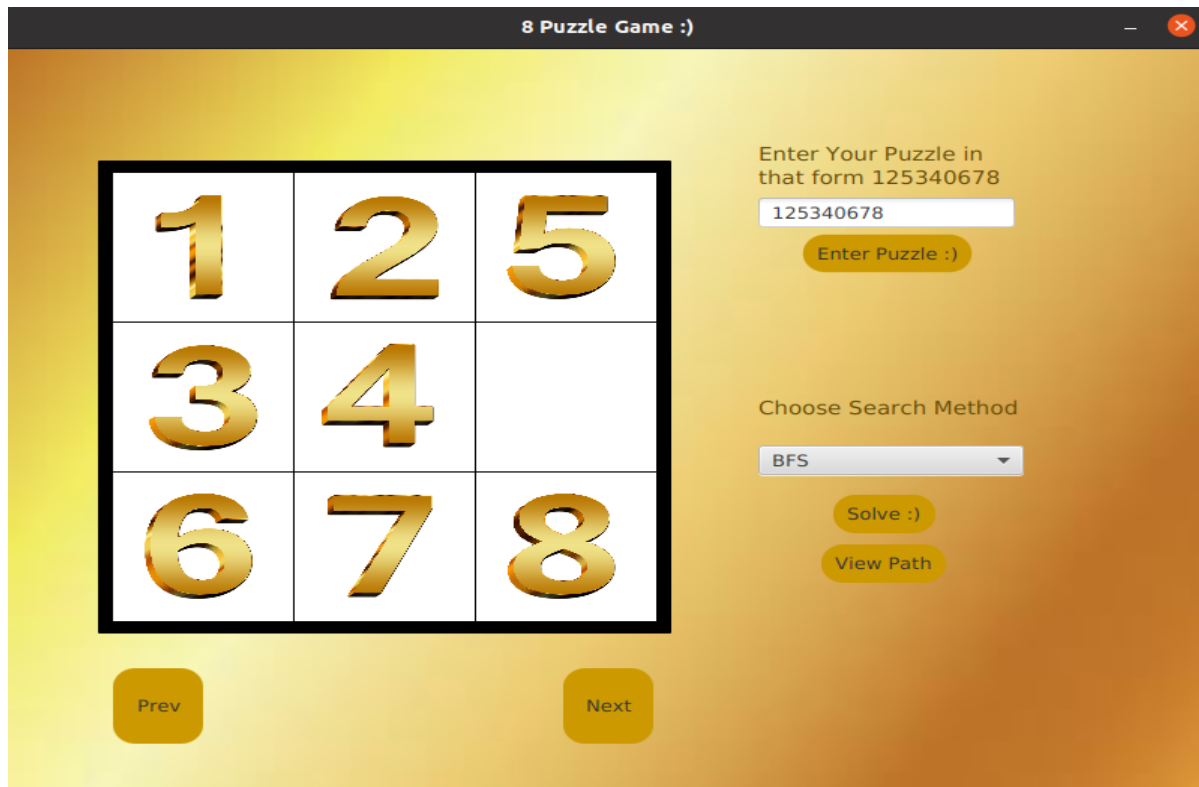


AI

Assignment 1

Name	ID
Mai Ahmed Hussein	19016736
Habiba Osama Zaki	19015575
Basel Ahmed Awad	19015513
Ali Hassan Alsharawy	19016013



Problem statement:

Solving the 8-puzzle game using BFS, DFS, and A* with the heuristic function as euclidean distance or manhattan distance to reach goal state "012345678".

Assumptions and Details:

- The default algorithm will be **BFS** if the user didn't choose an algorithm.
- Code detects invalid and unsolvable states.

Data Structures:

- **HashSet:**

Used to store the visited (explored) states.

Also used for the frontier list in BFS and DFS to check if it contains a certain state or not in $O(1)$ instead of $O(n)$ of the Queue and Stack (.contains) method.

- **Queue:**

Used in BFS as the frontier list.

- **Stack:**

Used in DFS as the frontier list.

- **PriorityQueue:**

Used in A* as the frontier list.

- **HashMap:**

Used in A* to store the lowest cost for the states currently in the frontier only.

- **LinkedList:**

Used to store the final path states.

Also used to store the state's children list(neighbour list).

- **Node:**

Added data structure that contains puzzle state stored as int, depth, reference to parent node, reference to children linkedlist(neighbour list), and the index of the zero in the puzzle state.

Algorithms:

BFS Algorithm:

Start by checking the shallowest node first, level by level using Queue as frontier.

DFS Algorithm:

Start by checking the deepest node first, using Stack as frontier.

A* Algorithm:

Searching according to the $\min(\text{Cost}(x) + \text{heuristic}(x))$ using Priority Queue as frontier.

The heuristic is calculated using 2 different methods Manhattan and Euclidean.

public void tracePath(Node n) :

Getting the final path starting from goal state node to the root node (initial state).

Manhattan Vs Euclidean:

EX 1: 125340678

POV	Manhattan	Euclidean
Max Depth	3	3
Cost of Path	3	3
Search Path	3	3
No. of Nodes Expanded	4	4
Running Time in μ s	120	177

EX 2: 768243105

POV	Manhattan	Euclidean
Max Depth	27	27
Cost of Path	27	27
Search Path	27	27
No. of Nodes Expanded	2196	8176
Running Time in μ s	13028	114496

EX 3: 182043765

POV	Manhattan	Euclidean
Max Depth	21	21
Cost of Path	21	21
Search Path	21	21
No. of Nodes Expanded	993	1429
Running Time in μ s	5744	14587

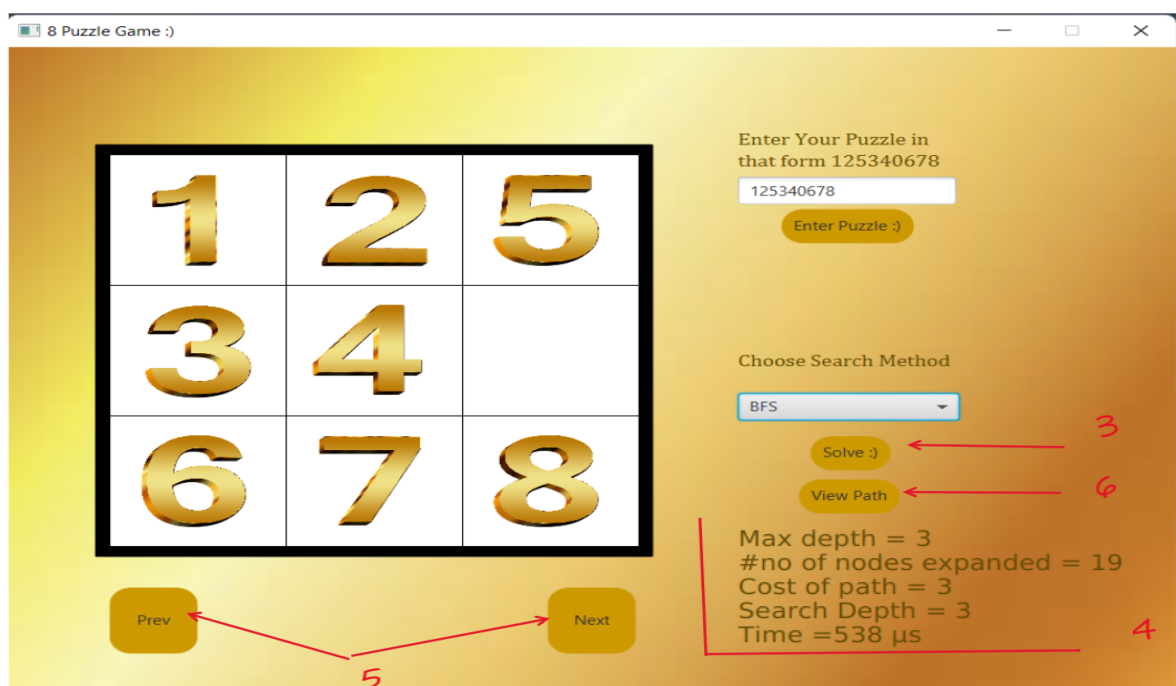
From the table shown above, it appears that Manhattan expands less nodes and has shorter running time than Euclidean. So, **Manhattan Distance is more admissible.**

How To Use:

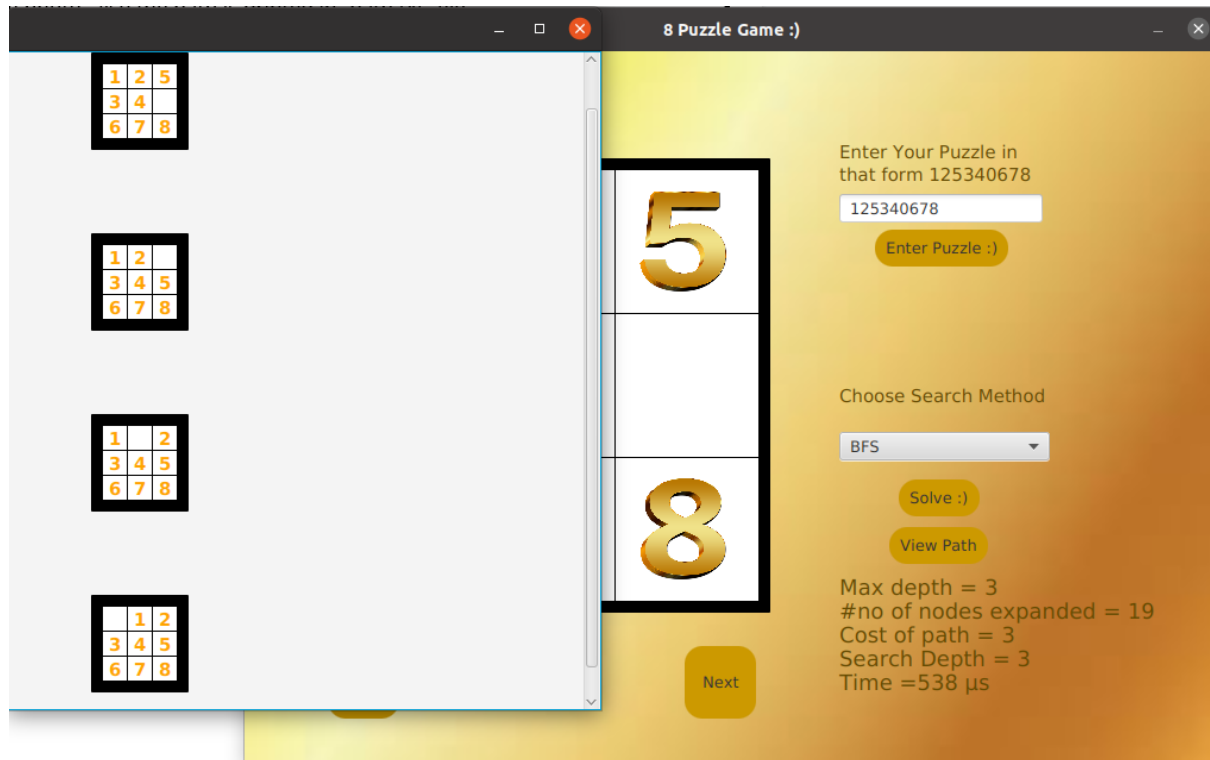


1- first user enters the initial case in that form 125340678 and then presses enter the puzzle. The puzzle will be viewed as shown.

2- choose the search method from the drop down menu . The default is BFS.

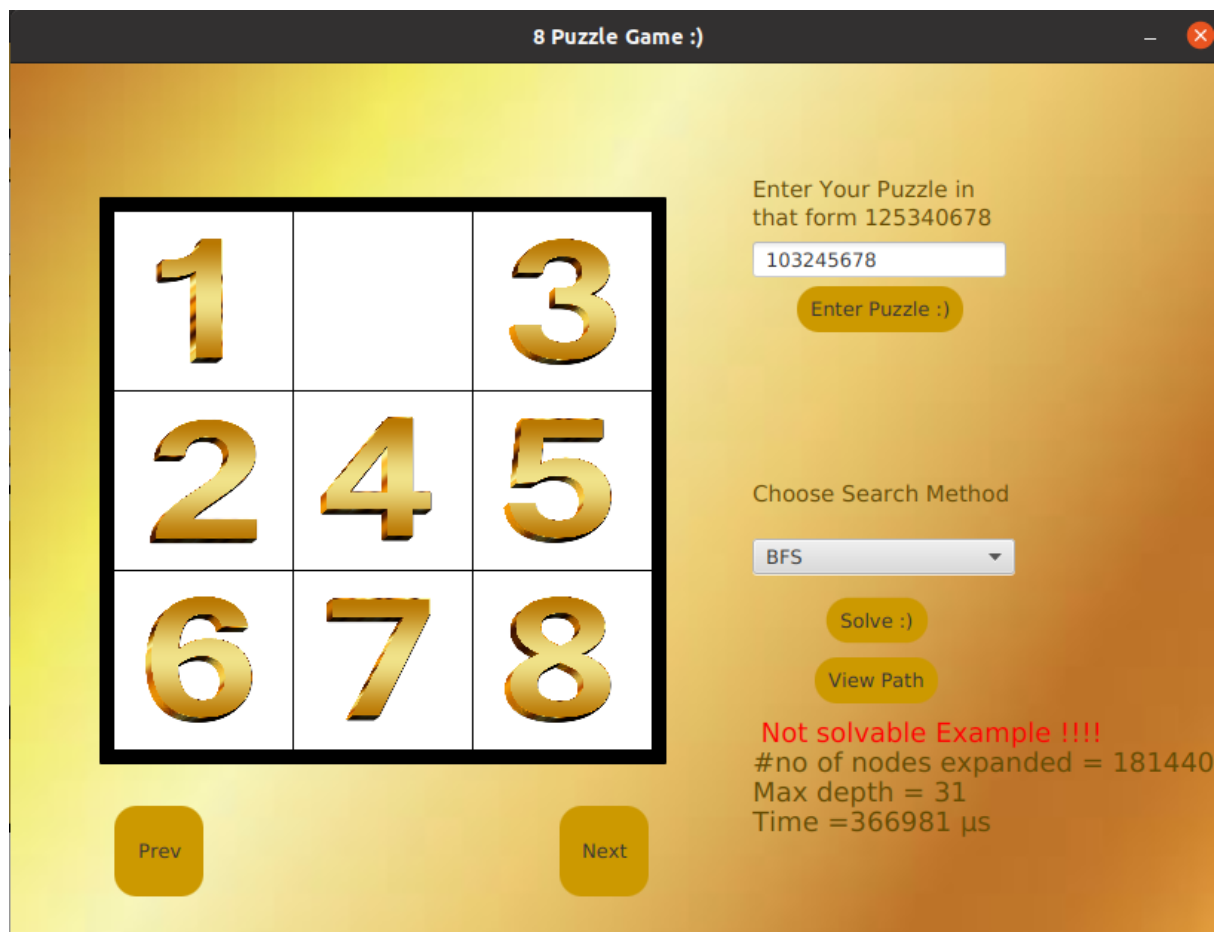


- 3- enter, then the results of depth ,time and cost will appear (number4).
- 4- the information wanted for this search.
- 5- you can press next to know the next move of the puzzle and keep moving till reach our goal state 012345678.
- 6- View path will open a new window that shows the whole path as in the following photo, also the path is printed in "path.txt" file.



Cases that program handle it:

If the example is not solvable will print not solvable



The screenshot shows the '8 Puzzle Game :)' window. On the left is a 3x3 grid with tiles containing numbers 1 through 8. The top row has tiles 1, an empty space, and 3. The middle row has tiles 2, 4, and 5. The bottom row has tiles 6, 7, and 8. On the right, there is a text input field with '103245678' and a yellow 'Enter Puzzle :)' button. Below that is a 'Choose Search Method' dropdown menu set to 'BFS', with yellow 'Solve :)' and 'View Path' buttons. At the bottom are yellow 'Prev' and 'Next' buttons. On the right side, there is a red text message: 'Not solvable Example !!!!' followed by statistics: '#no of nodes expanded = 181440', 'Max depth = 31', and 'Time = 366981 μs'.

1		3
2	4	5
6	7	8

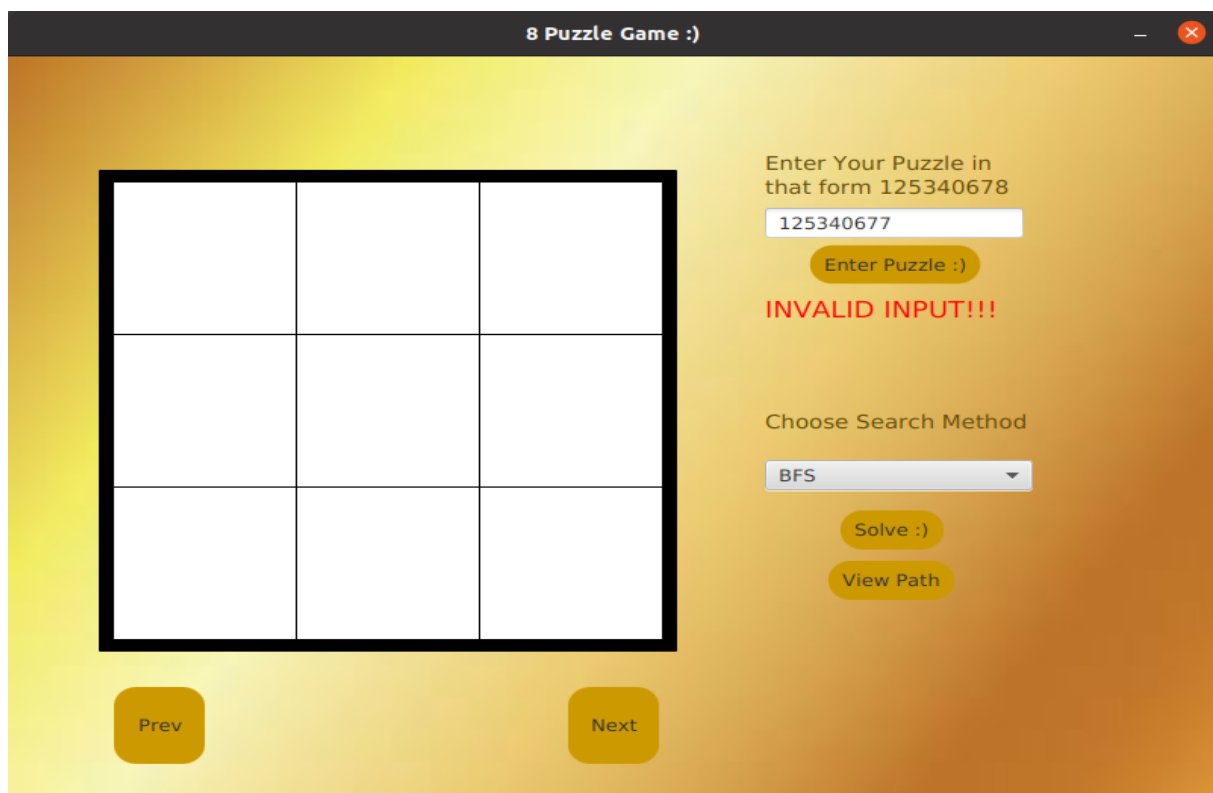
Enter Your Puzzle in that form 125340678
103245678
Enter Puzzle :)

Choose Search Method
BFS
Solve :)
View Path

Not solvable Example !!!!
#no of nodes expanded = 181440
Max depth = 31
Time = 366981 μs

Prev Next

If user input incorrect input will print "INVALID INPUT"



The screenshot shows the '8 Puzzle Game :)' window with the 3x3 grid empty. On the right, the text input field contains '125340677' and the yellow 'Enter Puzzle :)' button is highlighted. Below it, the 'Choose Search Method' dropdown menu is set to 'BFS', with yellow 'Solve :)' and 'View Path' buttons. At the bottom are yellow 'Prev' and 'Next' buttons. A red text message 'INVALID INPUT!!!' is displayed on the right side.

Enter Your Puzzle in that form 125340678
125340677
Enter Puzzle :)

INVALID INPUT!!!

Choose Search Method
BFS
Solve :)
View Path

Prev Next

Sample Runs:

EX 1: 125340678

BFS:

8 Puzzle Game :)

1	2	5
3	4	
6	7	8

Enter Your Puzzle in that form 125340678

125340678

Enter Puzzle :)

Choose Search Method

BFS

Solve :)

View Path

Max depth = 3
#no of nodes expanded = 19
Cost of path = 3
Search Depth = 3
Time = 68 μ s

Prev

Next

Path:

1	2	5
3	4	
6	7	8

1	2	
3	4	5
6	7	8

1		2
3	4	5
6	7	8

	1	2
3	4	5
6	7	8

DFS:

8 Puzzle Game :)

1	2	5
3	4	
6	7	8

PrevNext

Enter Your Puzzle in that form 125340678

125340678

Enter Puzzle :)

Choose Search Method

DFS

Solve :)

View Path

Max depth = 3
#no of nodes expanded = 4
Cost of path = 3
Search Depth = 3
Time =44 μ s

A* using Euclidean:

8 Puzzle Game :)

1	2	5
3	4	
6	7	8

PrevNext

Enter Your Puzzle in that form 125340678

125340678

Enter Puzzle :)

Choose Search Method

A* using Euclidean

Solve :)

View Path

Max depth = 3
#no of nodes expanded = 4
Cost of path = 3
Search Depth = 3
Time =177 μ s

A* using Manhattan:

8 Puzzle Game :)

1	2	5
3	4	
6	7	8

Enter Your Puzzle in that form 125340678

125340678

Enter Puzzle :)

Choose Search Method

A* using Manhattan

Solve :)

View Path

Max depth = 3
#no of nodes expanded = 4
Cost of path = 3
Search Depth = 3
Time = 120 μ s

Prev Next

EX 2: 012435687

BFS:

8 Puzzle Game :)

	1	2
4	3	5
6	8	7

Enter Your Puzzle in that form 125340678

012435687

Enter Puzzle :)

Choose Search Method

BFS

Solve :)

View Path

Max depth = 18
#no of nodes expanded = 20504
Cost of path = 18
Search Depth = 18
Time = 19099 μ s

Prev Next

DFS:

8 Puzzle Game :)

	1	2
4	3	5
6	8	7

Enter Your Puzzle in that form 125340678

012435687

Enter Puzzle :)

Choose Search Method

DFS

Solve :)

View Path

Max depth = 49930
#no of nodes expanded = 57472
Cost of path = 49930
Search Depth = 49930
Time = 130767 μ s

Prev Next

A* using Euclidean:

8 Puzzle Game :)

	1	2
4	3	5
6	8	7

Enter Your Puzzle in that form 125340678

012435687

Enter Puzzle :)

Choose Search Method

A* using Euclidean

Solve :)

View Path

Max depth = 18
#no of nodes expanded = 585
Cost of path = 18
Search Depth = 18
Time = 28518 μ s

Prev Next

A* using Manhattan:

8 Puzzle Game :)

	1	2
4	3	5
6	8	7

Prev

Next

Enter Your Puzzle in that form 125340678

012435687

Enter Puzzle :)

Choose Search Method

A* using Manhattan

Solve :)

View Path

Max depth = 18
#no of nodes expanded = 533
Cost of path = 18
Search Depth = 18
Time = 4754 μ s

Path:

1	2	
4	3	5
6	8	7

1	2	5
4	3	7
6		8

1	2	
3	7	5
4	6	8

3	1	2
4	7	5
6		8

3	1	2
4		5
6	7	8

3	1	2
	4	5
6	7	8

	1	2
3	4	5
6	7	8

Invalid Input:

8 Puzzle Game :)

Enter Your Puzzle in that form 125340678

125340677

Enter Puzzle :)

INVALID INPUT!!!

Choose Search Method

BFS

Solve :)

View Path

Prev Next

Unsolvable State:

8 Puzzle Game :)

1		3
2	4	5
6	7	8

Enter Your Puzzle in that form 125340678

103245678

Enter Puzzle :)

Choose Search Method

BFS

Solve :)

View Path

Not solvable Example !!!!
#no of nodes expanded = 181440
Max depth = 31
Time = 366981 μ s

Prev Next