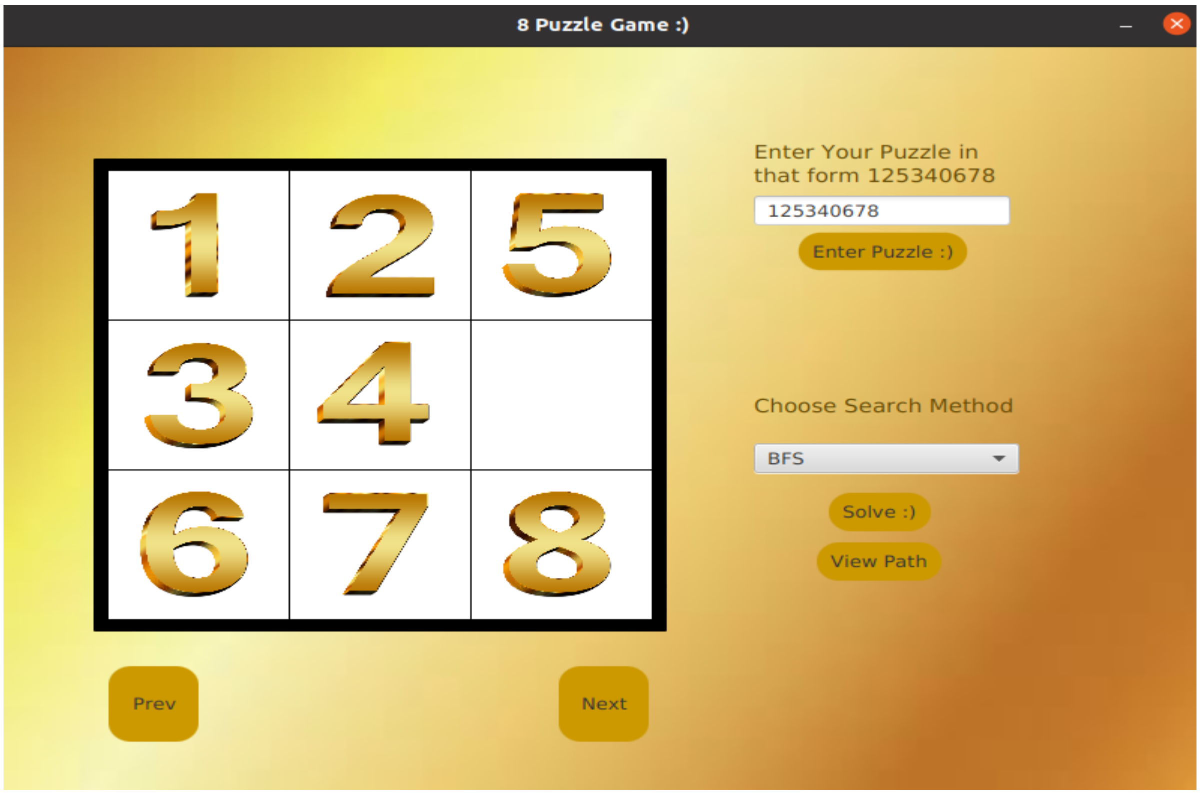
AI
Assignment 1

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



Enter Your Puzzle in that form 125340678: 125340678

Choose Search Method: BFS

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 and doesn’t attempt to solve them.

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(Cost(x) + 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 Visited	4	4
No. of Nodes Expanded	8	8
Running Time in µs	432	1515

EX 2: 768243105

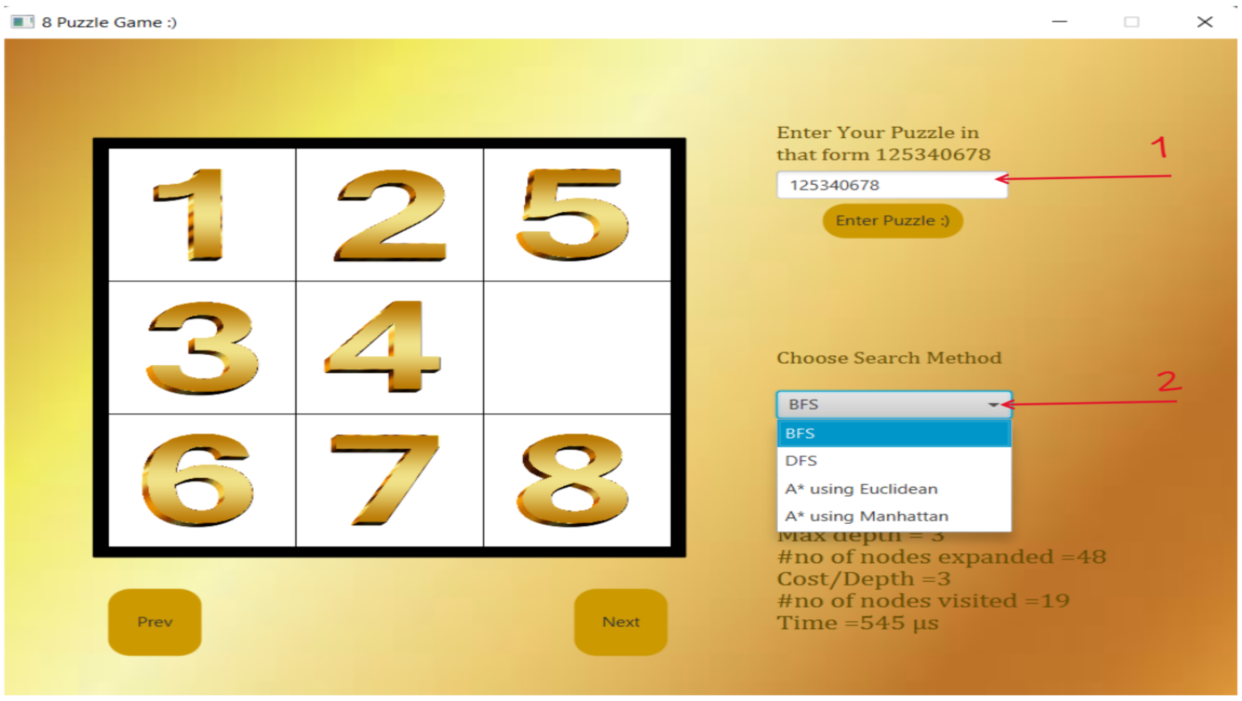
POV	Manhattan	Euclidean
Max Depth	27	27
Cost of Path	27	27
Search Path	27	27
No. of Nodes Visited	4984	10629
No. of Nodes Expanded	13567	28775
Running Time in µs	59975	320786

EX 3: 182043765

POV	Manhattan	Euclidean
Max Depth	21	21
Cost of Path	21	21
Search Path	21	21
No. of Nodes Visited	1388	2276
No. of Nodes Expanded	3783	6143
Running Time in µs	7940	26980

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.**

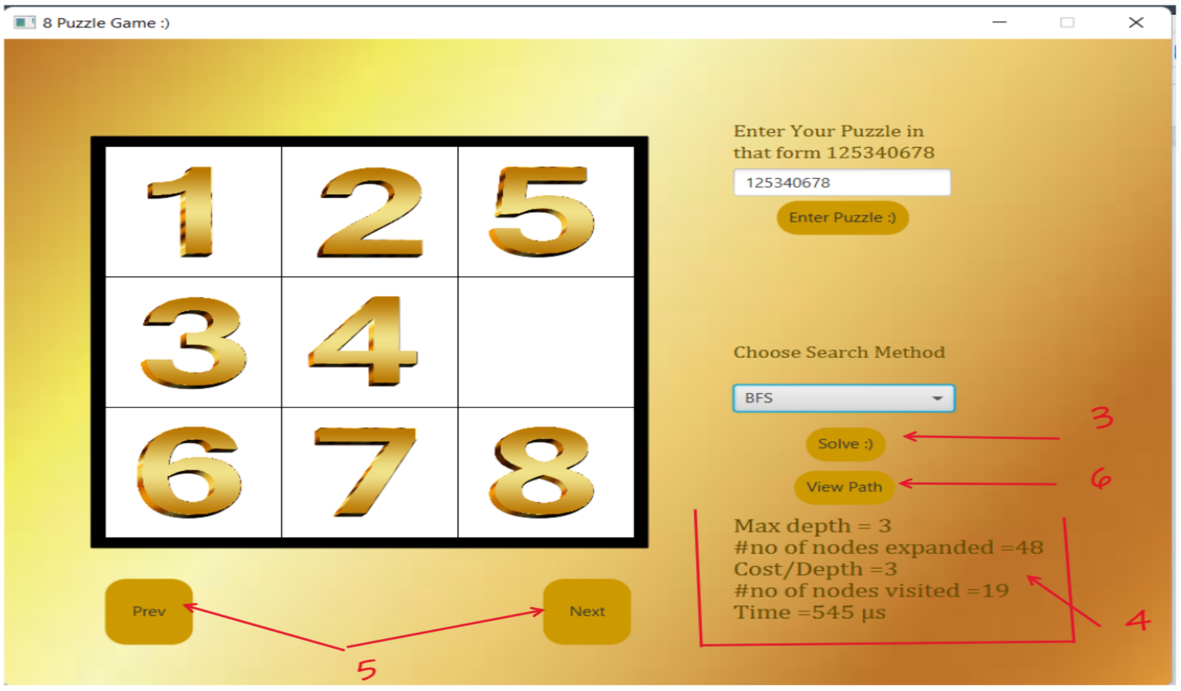
How To Use:



Max depth = 3
#no of nodes expanded =48
Cost/Depth =3
#no of nodes visited =19
Time =545 µs

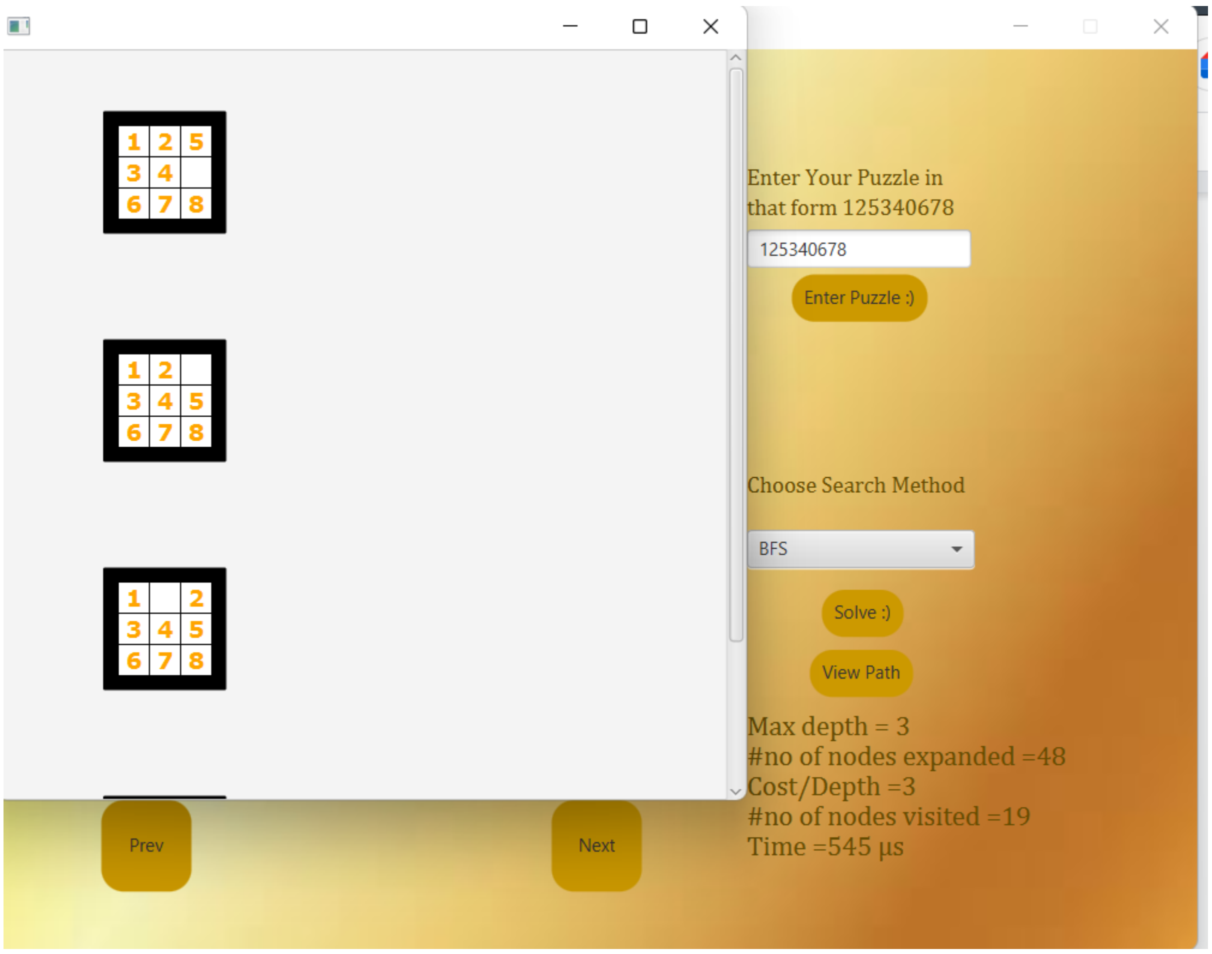
1- first user enters the initial case in that form 125340678 and then presses enter the puzzle. The puzzle will be viewed as shown.

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



Max depth = 3
#no of nodes expanded =48
Cost/Depth =3
#no of nodes visited =19
Time =545 µs

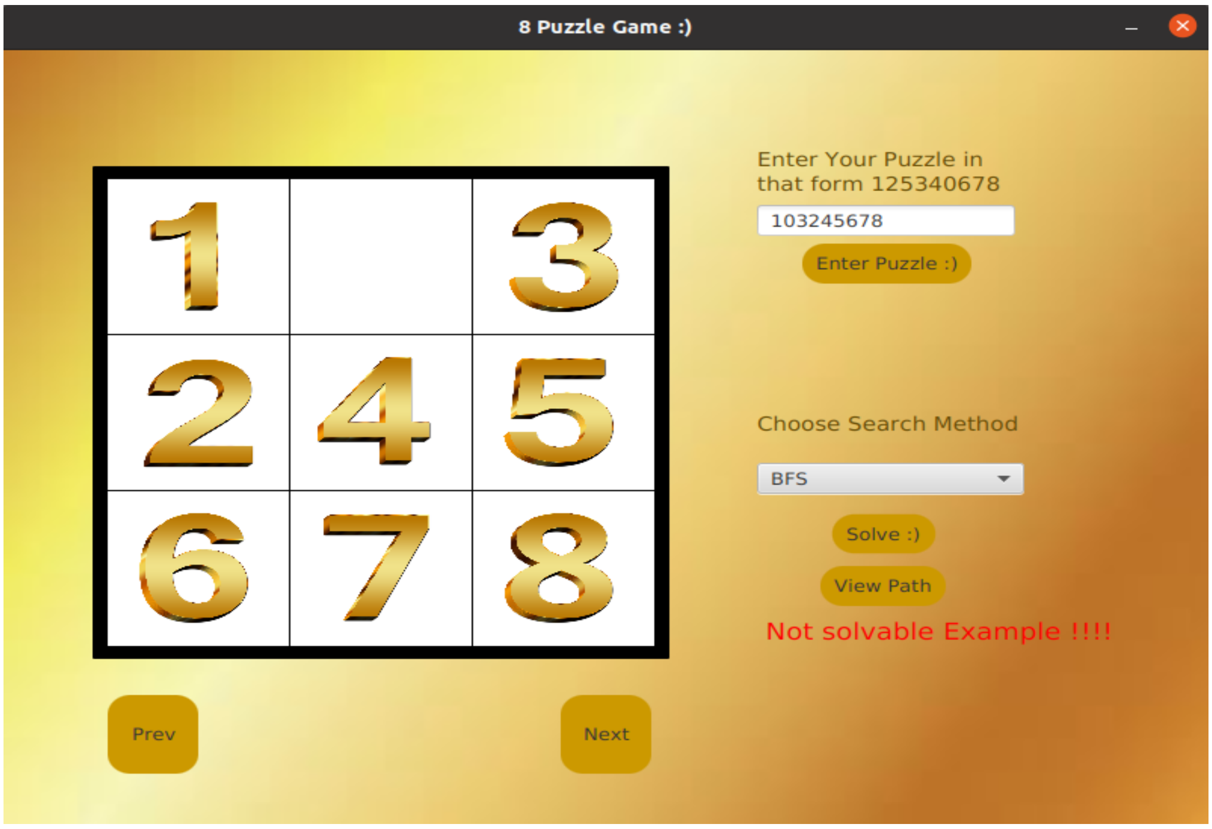
3- enter, then the results of depth ,time and cost will appear (number4).
4- the information wanted for this search.
5- you can press next to know the next move of the puzzle and keep moving till reach our goal state 012345678.
6- View path will open a new window that shows the whole path as in the following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



Max depth = 3
#no of nodes expanded =48
Cost/Depth =3
#no of nodes visited =19
Time =545 µs

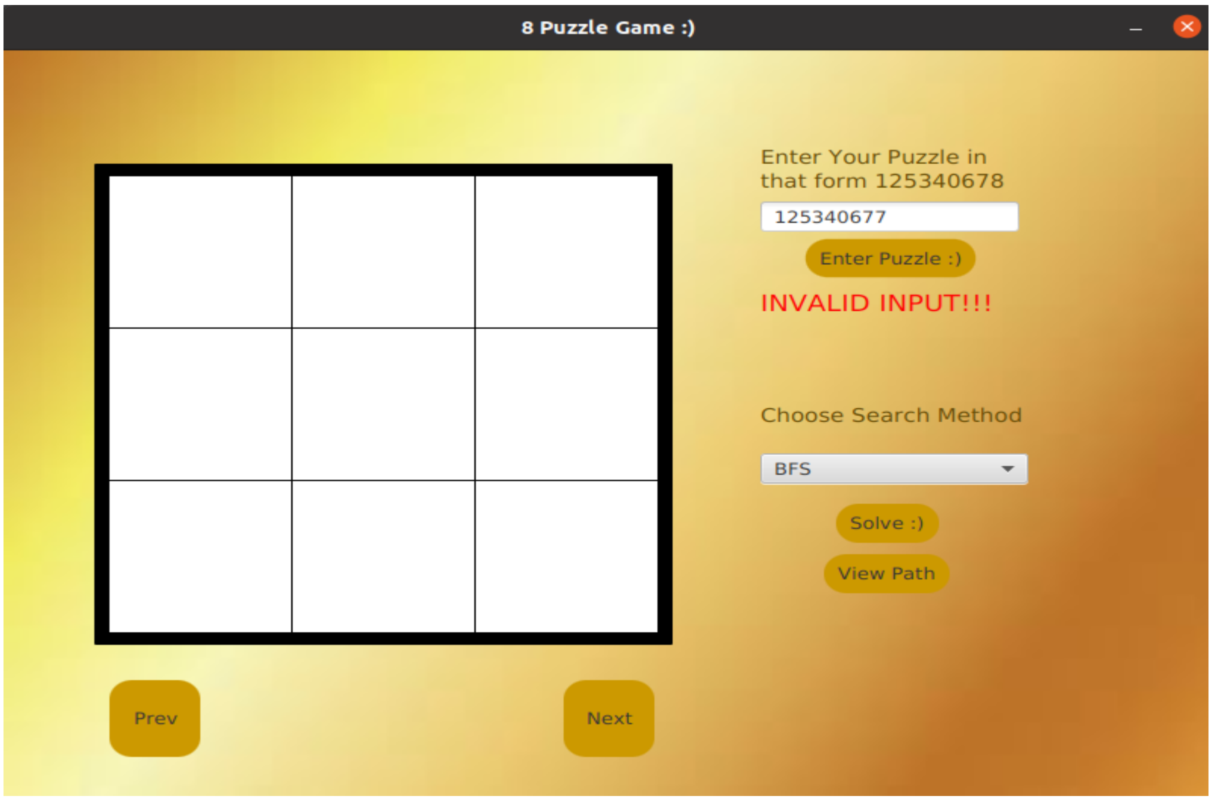
Cases that program handle it:

If the example is not solvable will print not solvable



Not solvable Example !!!!

If user input incorrect input will print "INVALID INPUT"

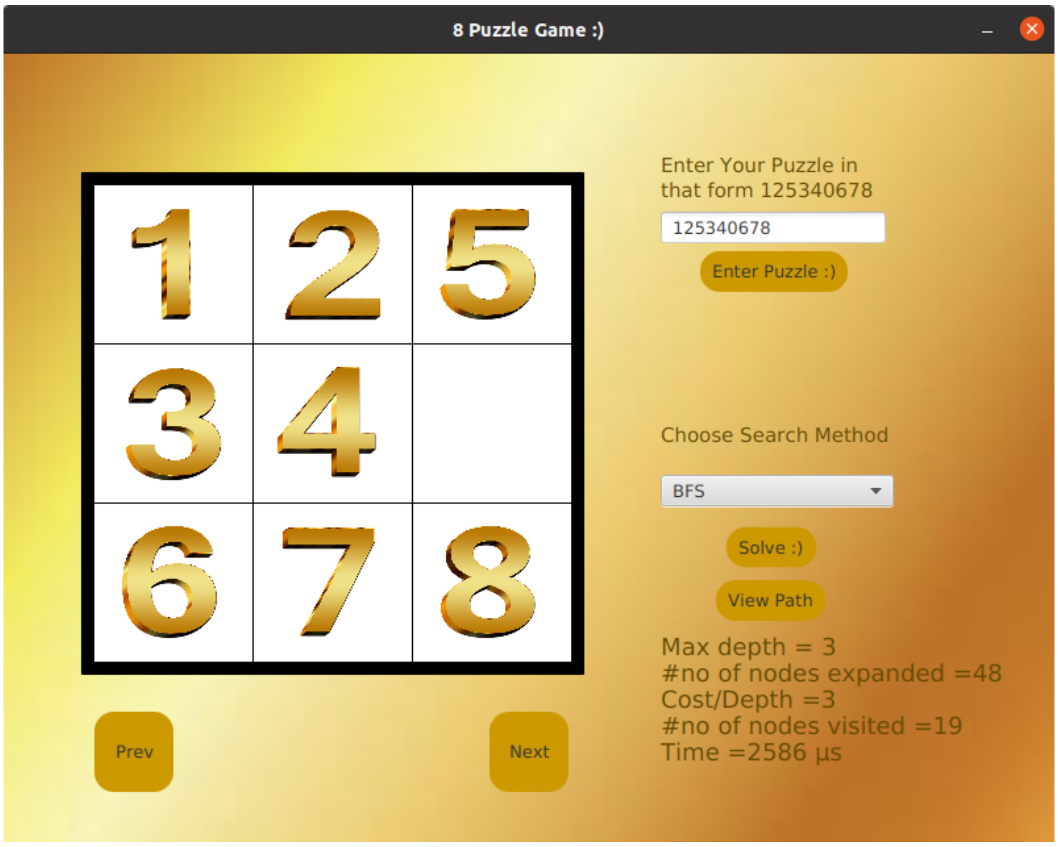


INVALID INPUT!!!

Sample Runs:

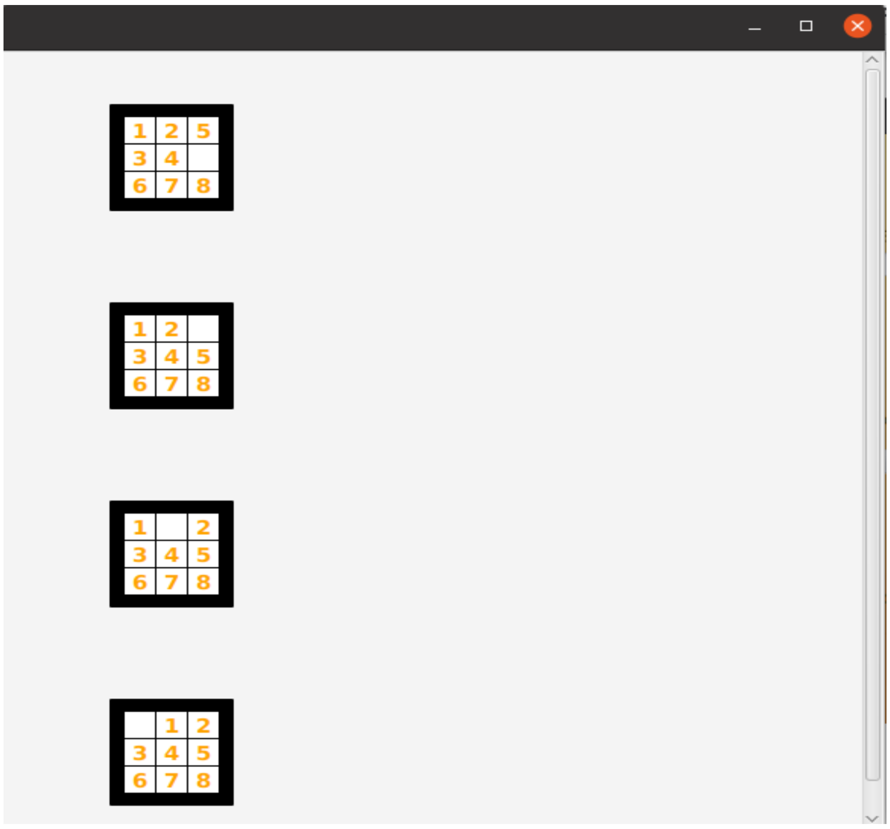
EX 1: 125340678

BFS:

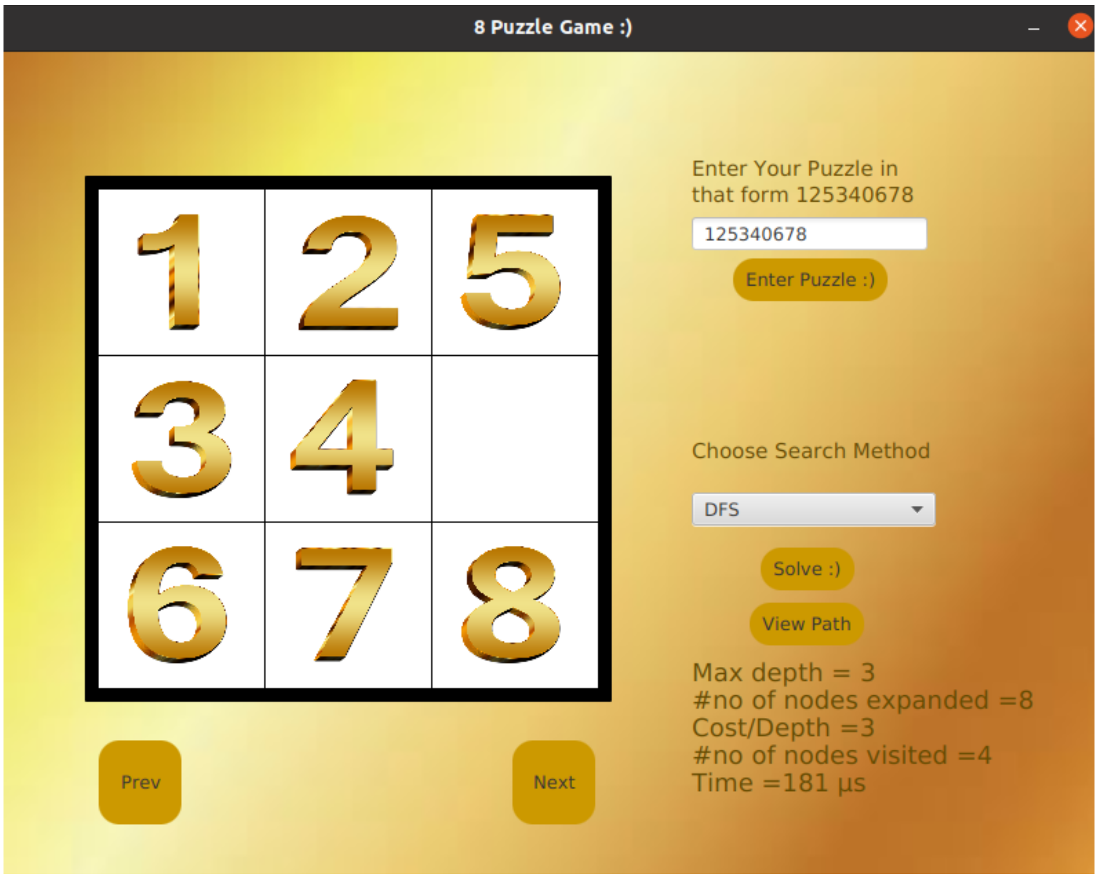


Max depth = 3
#no of nodes expanded =48
Cost/Depth =3
#no of nodes visited =19
Time =2586 µs

Path:



DFS:



Enter Your Puzzle in that form 125340678

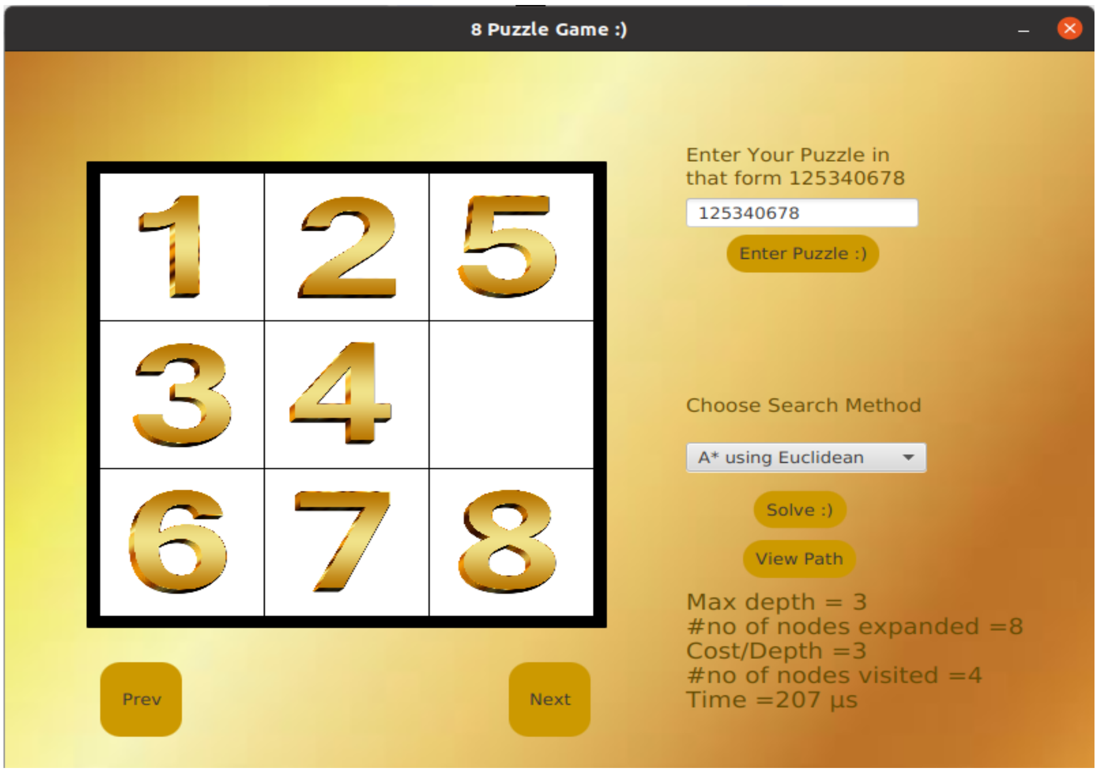
125340678

Choose Search Method

DFS

Max depth = 3
#no of nodes expanded =8
Cost/Depth =3
#no of nodes visited =4
Time =181 µs

A* using Euclidean:



Enter Your Puzzle in that form 125340678

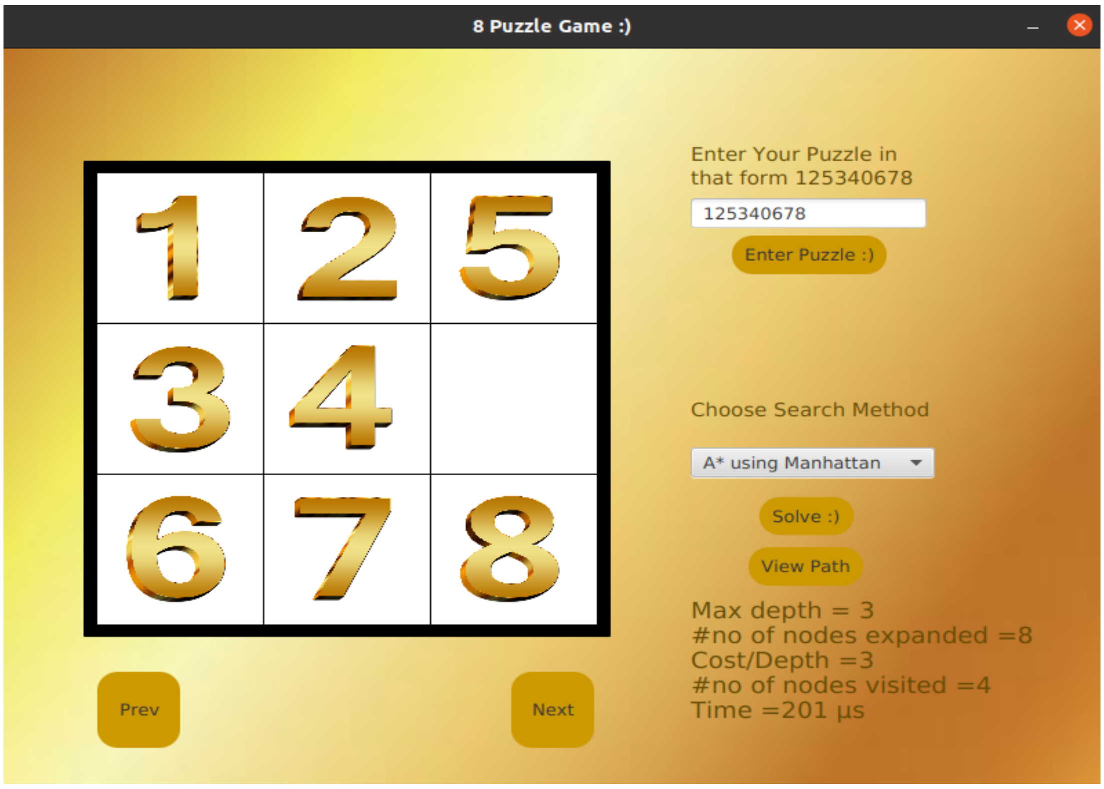
125340678

Choose Search Method

A* using Euclidean

Max depth = 3
#no of nodes expanded =8
Cost/Depth =3
#no of nodes visited =4
Time =207 µs

A* using Manhattan:



Enter Your Puzzle in that form 125340678

125340678

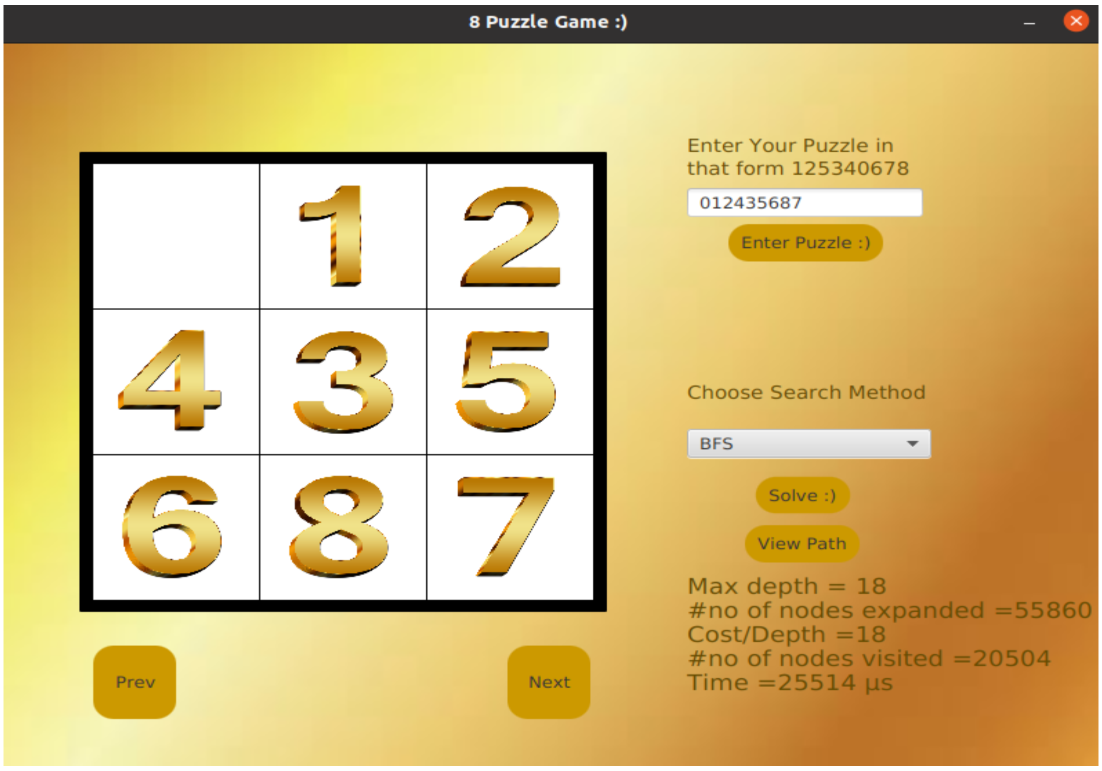
Choose Search Method

A* using Manhattan

```
Max depth = 3
#no of nodes expanded =8
Cost/Depth =3
#no of nodes visited =4
Time =201 µs
```

EX 2: 012435687

BFS:



Enter Your Puzzle in that form 125340678

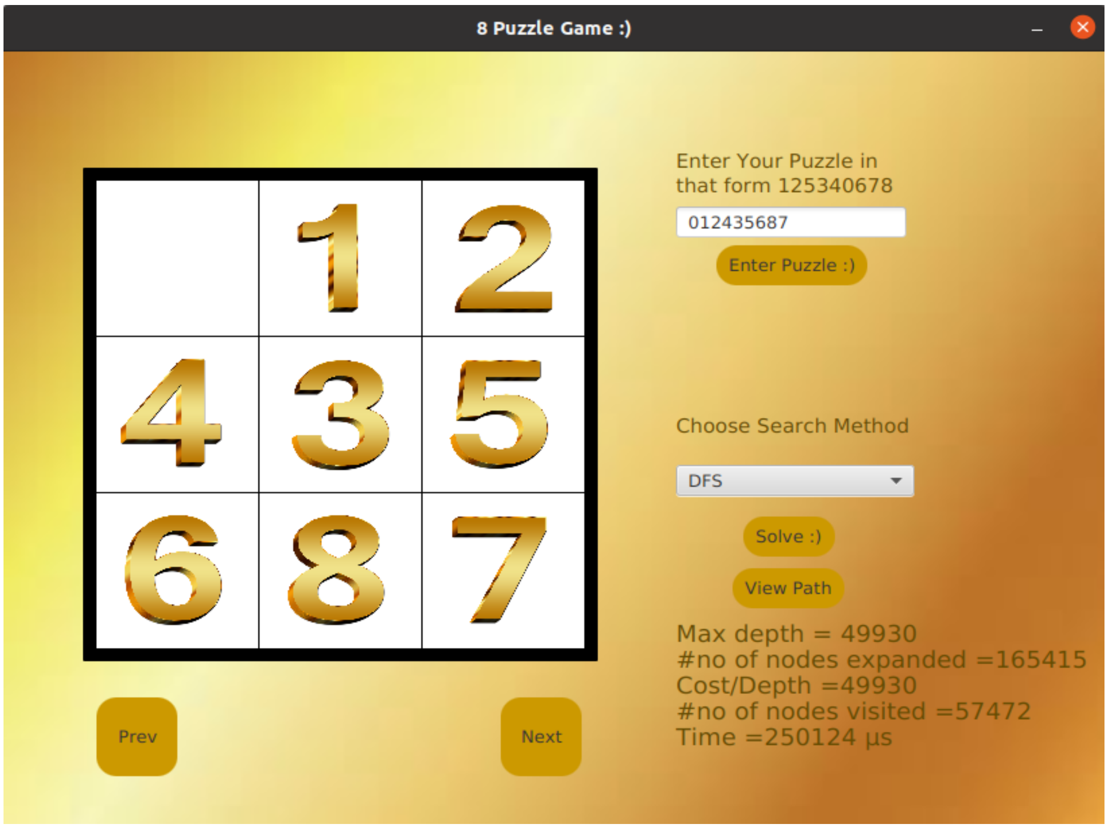
012435687

Choose Search Method

BFS

```
Max depth = 18
#no of nodes expanded =55860
Cost/Depth =18
#no of nodes visited =20504
Time =25514 µs
```

DFS:



Enter Your Puzzle in that form 125340678

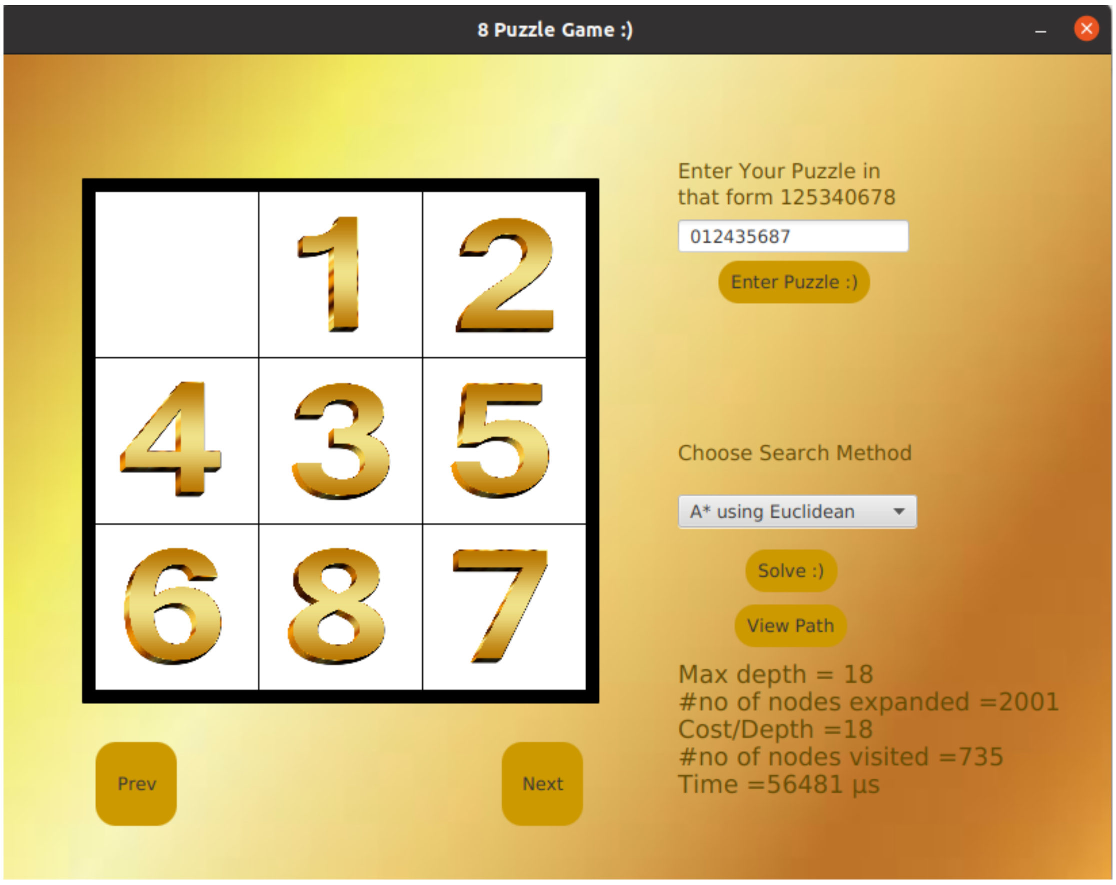
012435687

Choose Search Method

DFS

```
Max depth = 49930
#no of nodes expanded =165415
Cost/Depth =49930
#no of nodes visited =57472
Time =250124 µs
```

A* using Euclidean:



Enter Your Puzzle in that form 125340678

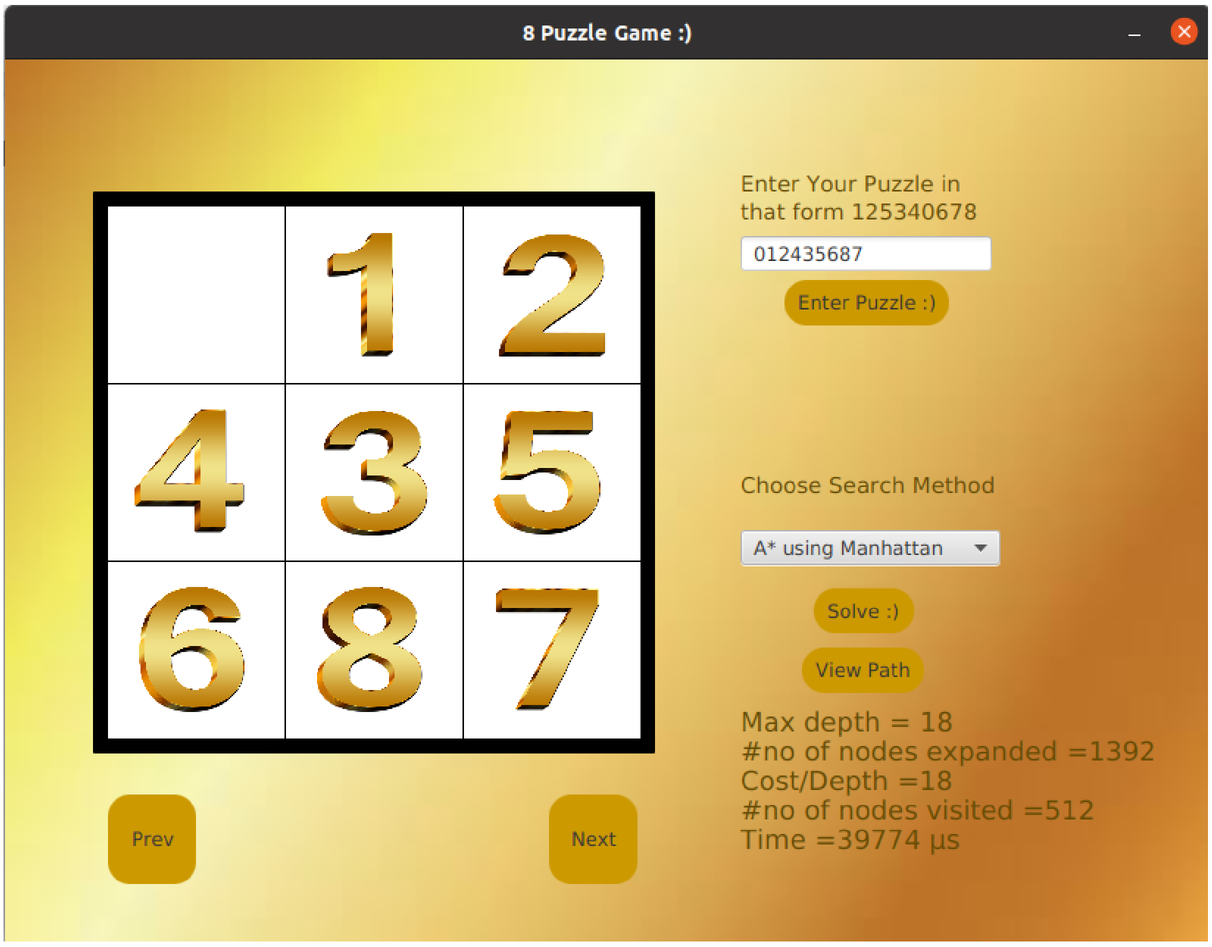
012435687

Choose Search Method

A* using Euclidean

```
Max depth = 18
#no of nodes expanded =2001
Cost/Depth =18
#no of nodes visited =735
Time =56481 µs
```

A* using Manhattan:


8 Puzzle Game :)

Enter Your Puzzle in that form 125340678

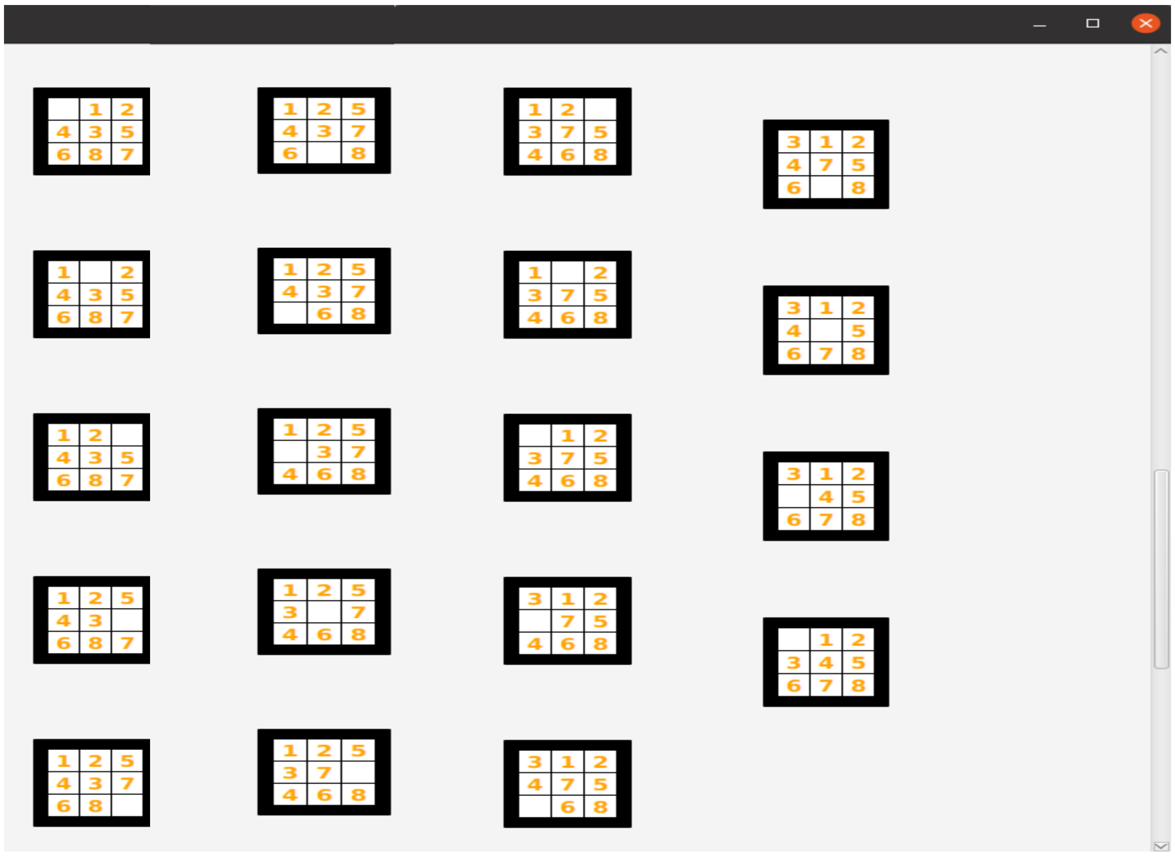
012435687

Choose Search Method

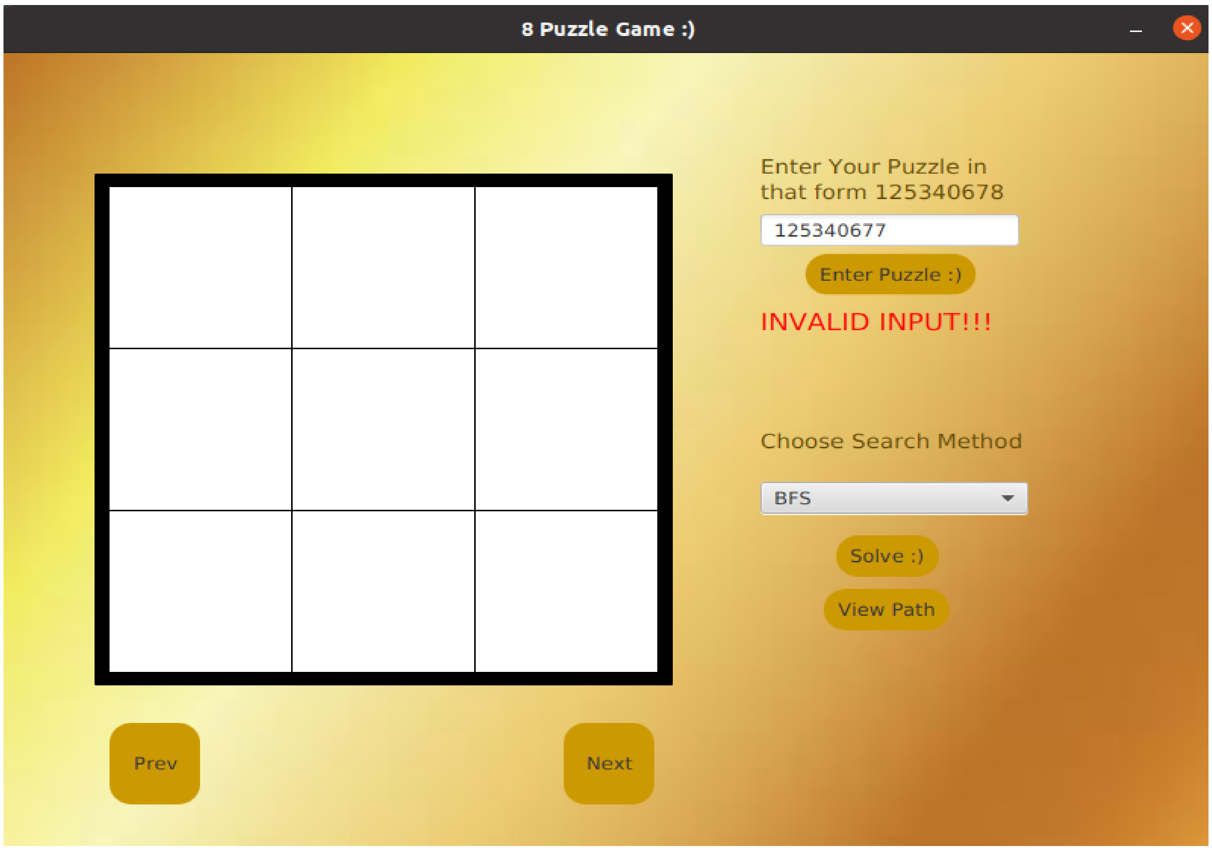
A* using Manhattan

```
Max depth = 18
#no of nodes expanded =1392
Cost/Depth =18
#no of nodes visited =512
Time =39774 µs
```

Path:



Invalid Input:



Enter Your Puzzle in that form 125340678

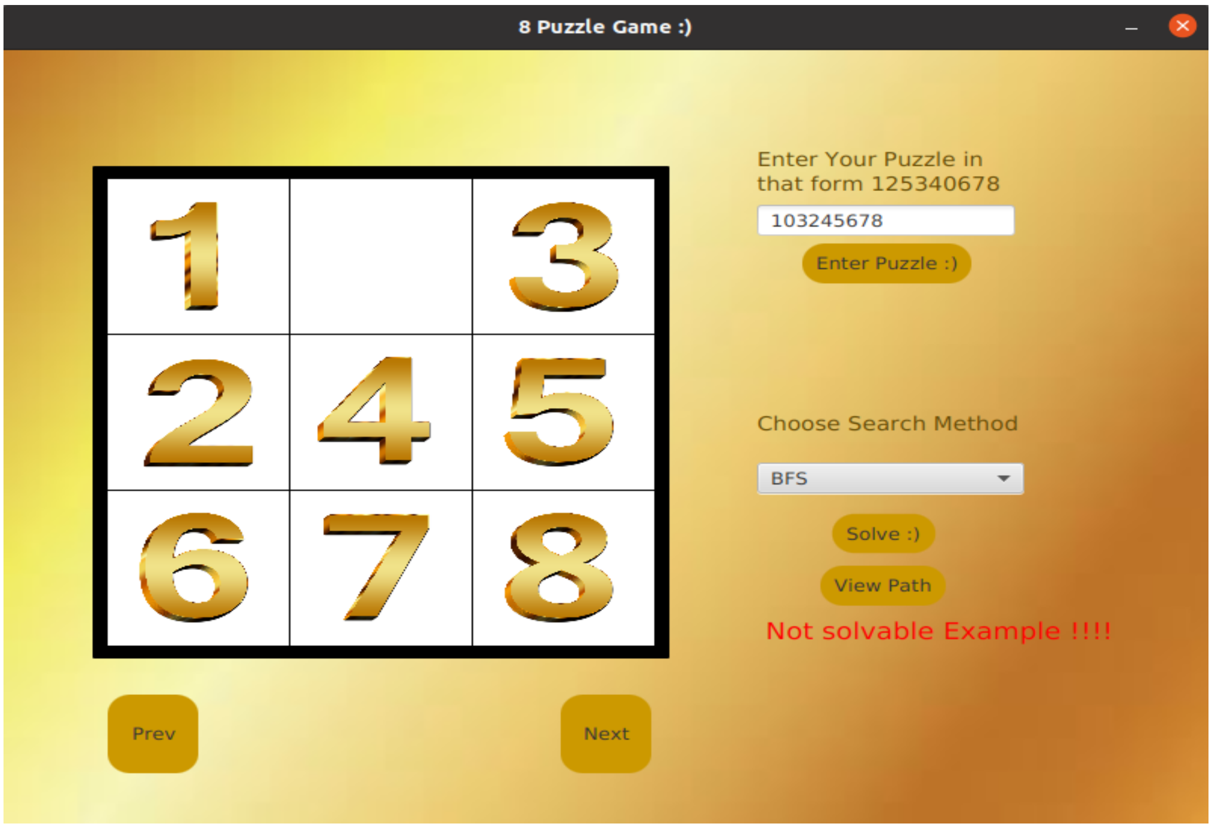
125340677

INVALID INPUT!!!

Choose Search Method

BFS

Unsolvable State:



Enter Your Puzzle in that form 125340678

103245678

Choose Search Method

BFS

Not solvable Example !!!!