Assignment 1 AI
8-Puzzle

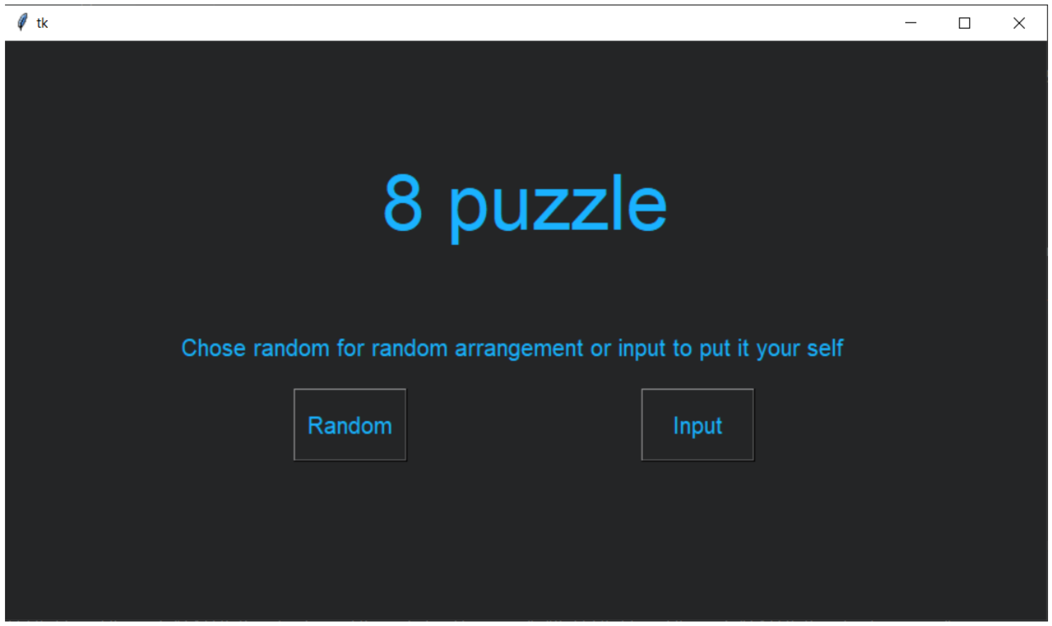
No	Name	ID
1	عبدالرحمن السيد احمد علي	19015893
2	عبدالرحمن السيد جاد السيد	19015894
3	عبدالعزيز محمد عبدالعزيز محمد	19015941
4	عمر خيرت محمد ابو ضيف	19016063

1. Main Features:

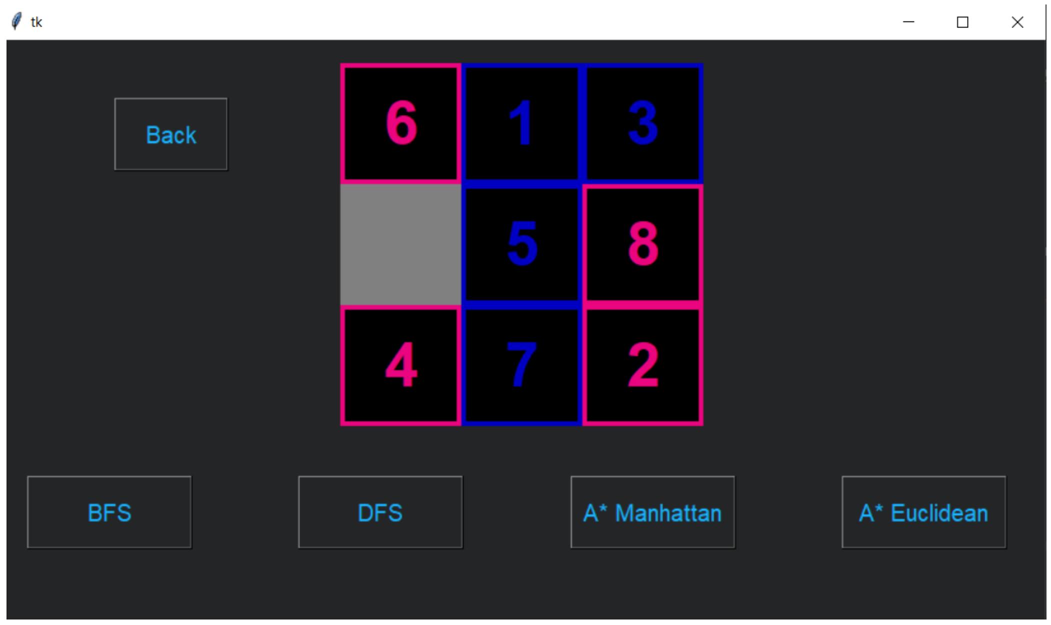
- Solve the 8-puzzle using 4 different algorithms:
  a. DFS
  b. BFS
  c. A* with Manhattan distance
  d. A* with Euclidean distance
- User friendly graphical user interface that is enjoyable and provides great experience.
- The user is allowed to enter the puzzle manually or get random arrangement to the puzzle.
- Showing the entire steps of the solution automatically or step by step and the user can go to the previous steps.
- In each search, the user is shown a table containing:
  a. cost of the path
  b. number of nodes expanded
  c. search depth
  d. the running time
- The ability to find if there is no solution of the given puzzle and show the table to let the user know the number of nodes expanded and the search depth and the running time till finding that there is no solution.

2. User Guide:

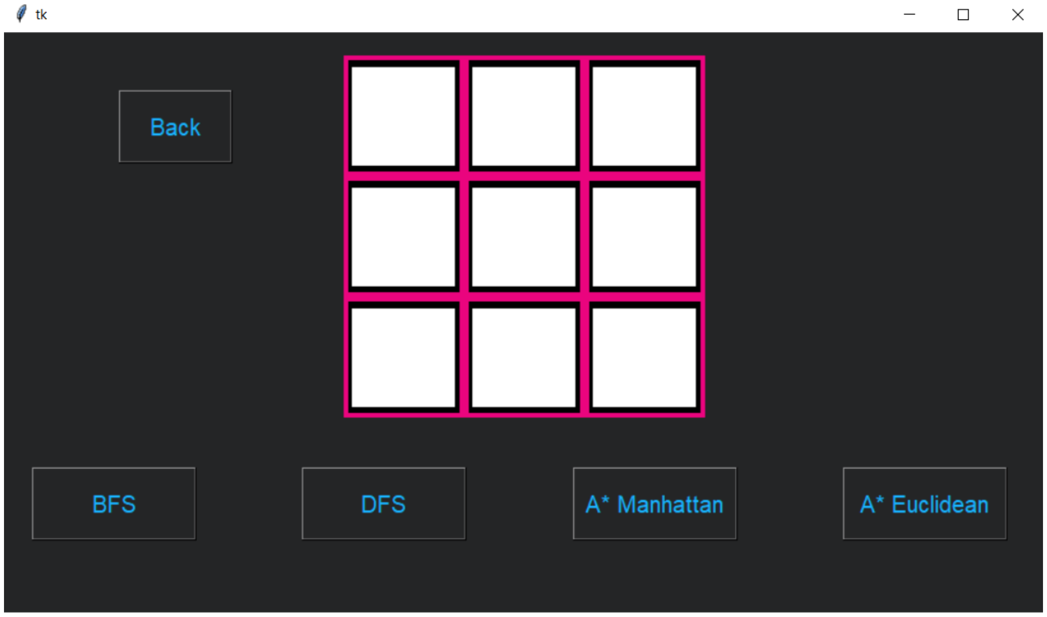
A. After running the program at the home page, the user can choose between 2 options, Random to arrange the puzzle randomly or Input to enter the arrangement himself.


tk
8 puzzle
Chose random for random arrangement or input to put it your self
Random
Input

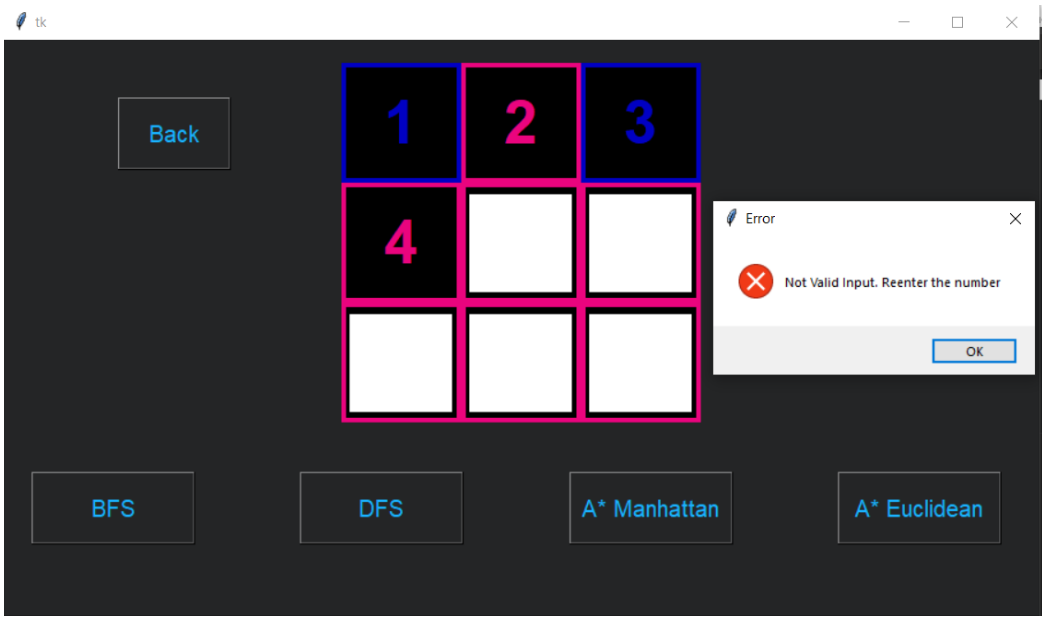
B. If the user chooses Random the random arrangement will appear and can choose the search method or go back to the home page.



C. If the user chooses Input, he can enter the number at each cell and then chose the search method or he can go back to the home page.

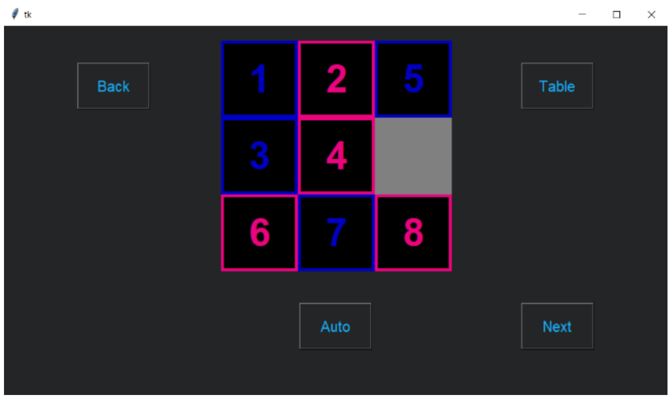


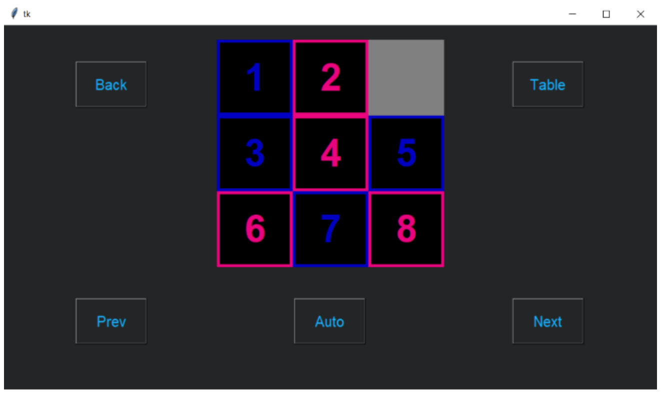
D. If the input is incorrect, error massage will appear.



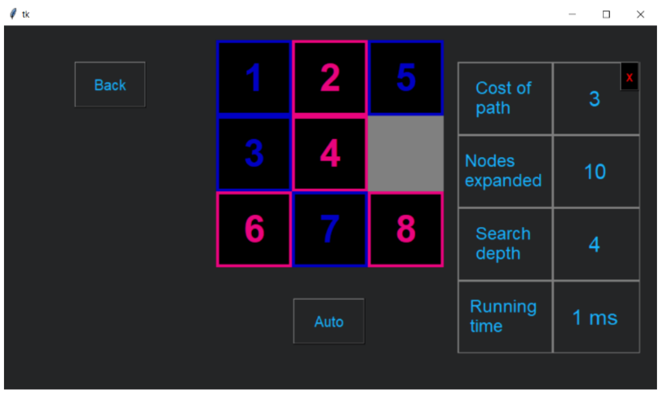
Not Valid Input. Reenter the number

E. After choosing the search method, steps page will appear. The user can go to the next step and move step by step or go to the previous at any step or click auto to make all steps to the end.

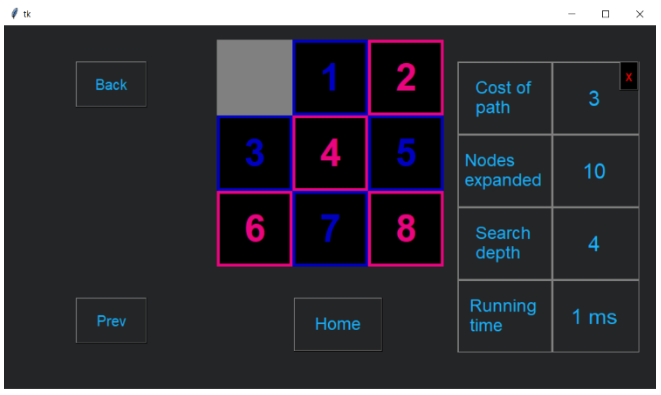




F. The user can show the table from the Table button at any time, or it will be shown automatically after the last step.



Cost of path	3
Nodes expanded	10
Search depth	4
Running time	1 ms



Cost of path	3
Nodes expanded	10
Search depth	4
Running time	1 ms

G. User can choose back to return to enter the input or chose another method or he can go to the home page after it finish.

3. Data Structure Used:

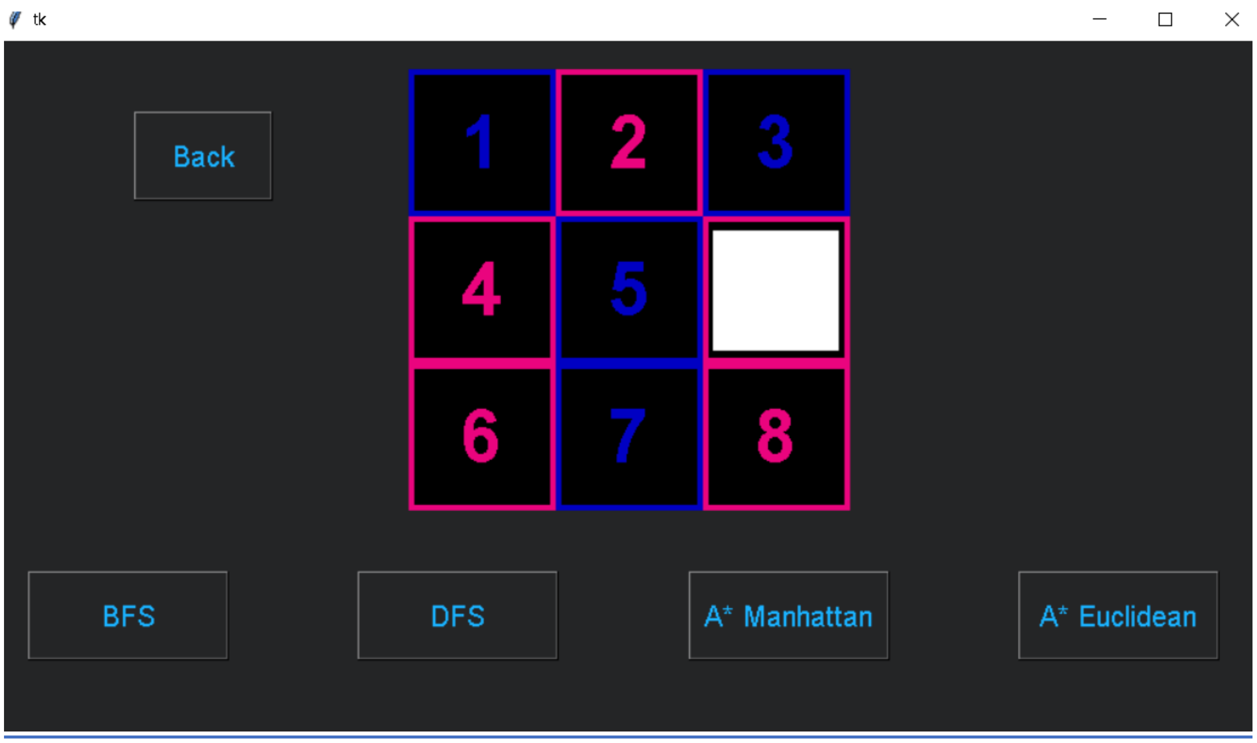
- **Array**: It is used to store the states of the puzzle in addition to the path.
- **Set**: It is used to store the states we visited in order we find the child of each state to add it to fringe we first check that they are not visited.
- **Stack**: It is used in DFS as fringe for expanded nodes.
- **Queue**: It is used in BFS as fringe for expanded nodes.
- **Priority Queue**: It is used in A* as fringe for expanded nodes where nodes are ordered according to the specified heuristic function in addition to the cost of the node.

4. Design Decisions:

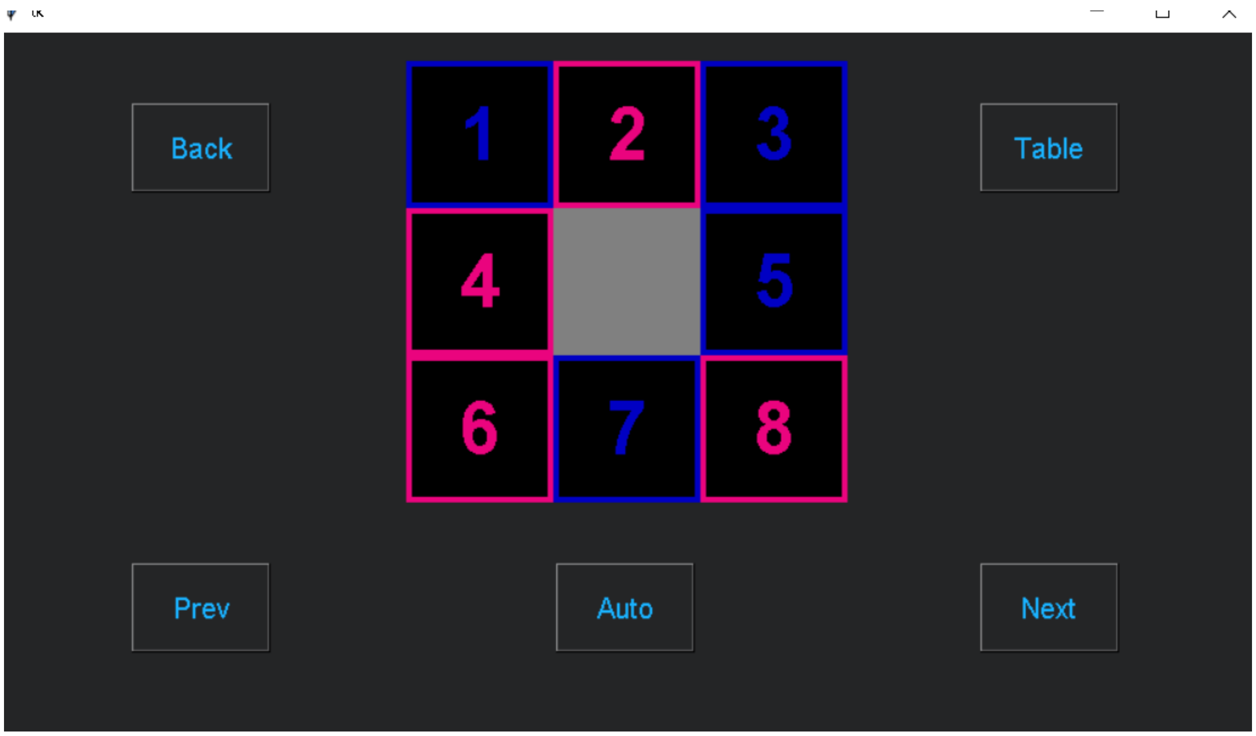
- The controller direct communication between graphical user interface and the search agent class.
- Services packages contain all modules that preform operations depending on Model package modules.
- The search agent class is responsible for solving the puzzle with given search algorithm from user and initial puzzle which is passed to from controller which give algorithm through passing it its corresponding class and here strategy design pattern is applied.
- The fringe in each search algorithm is given from FringeFactory class where factory design pattern is applied to give the desired fringe to the class it needs.
- All python built-in data structures are adapted for the fringe interface (Adapter design pattern).
- The steps of the solution to the puzzle and all the information needed for it is found in Answer class which have only one instance (Singleton design pattern).

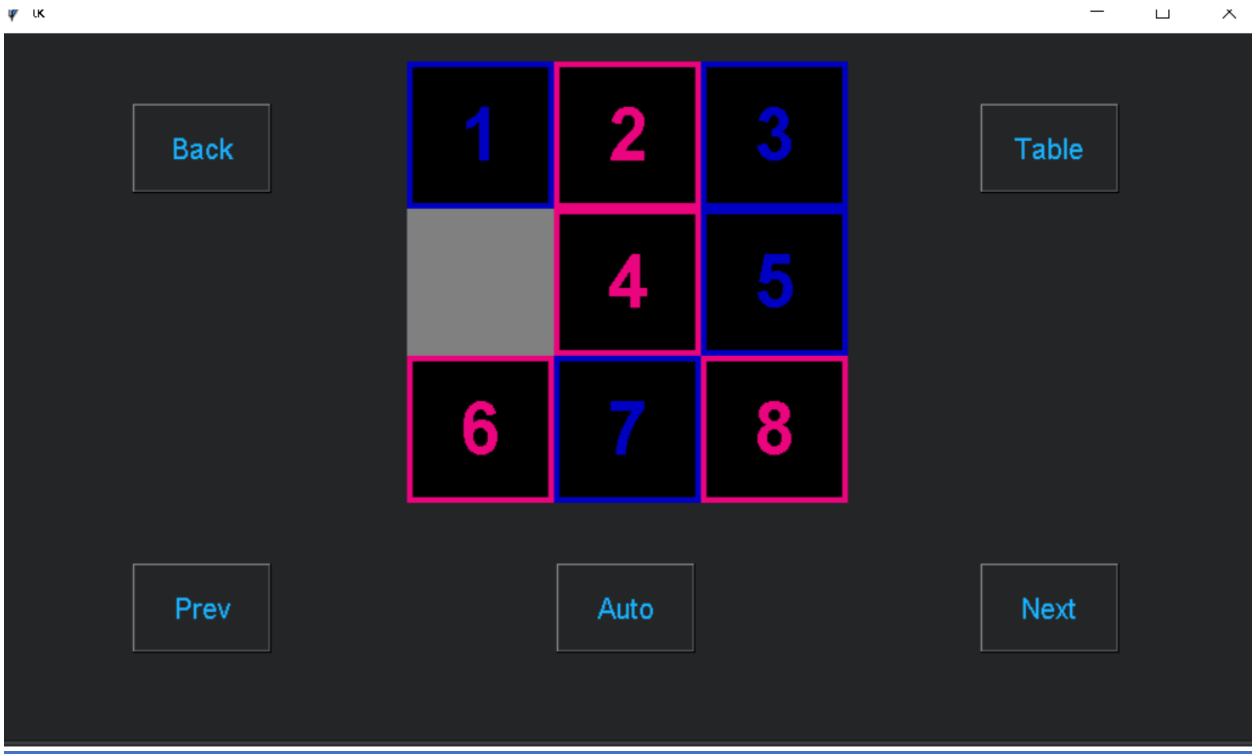
5. Sample Runs:

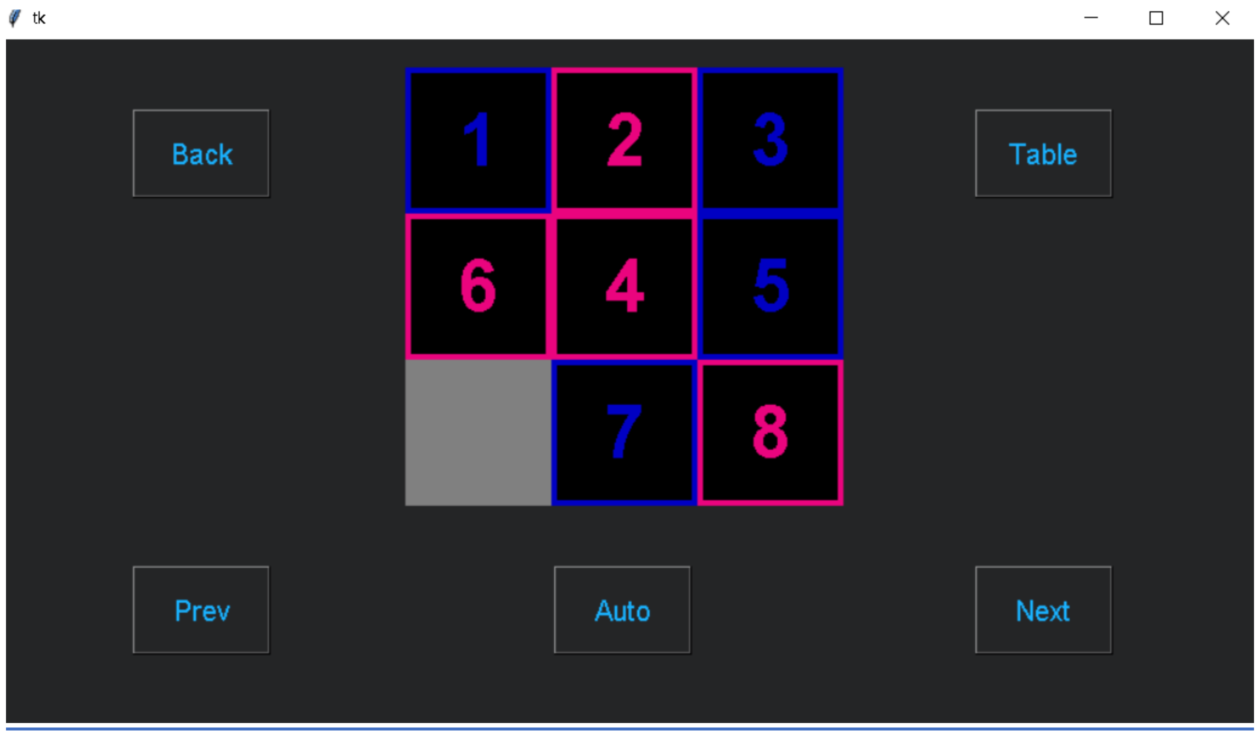
First case (input taken from user) :

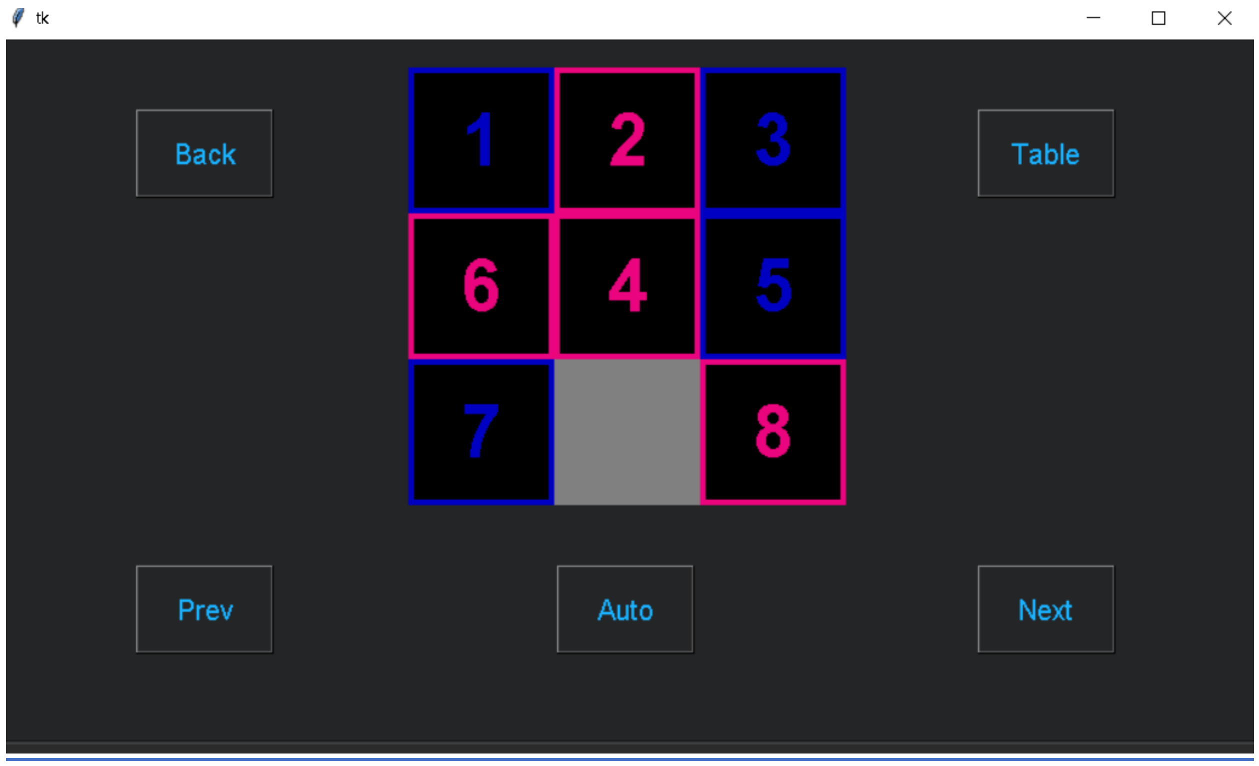


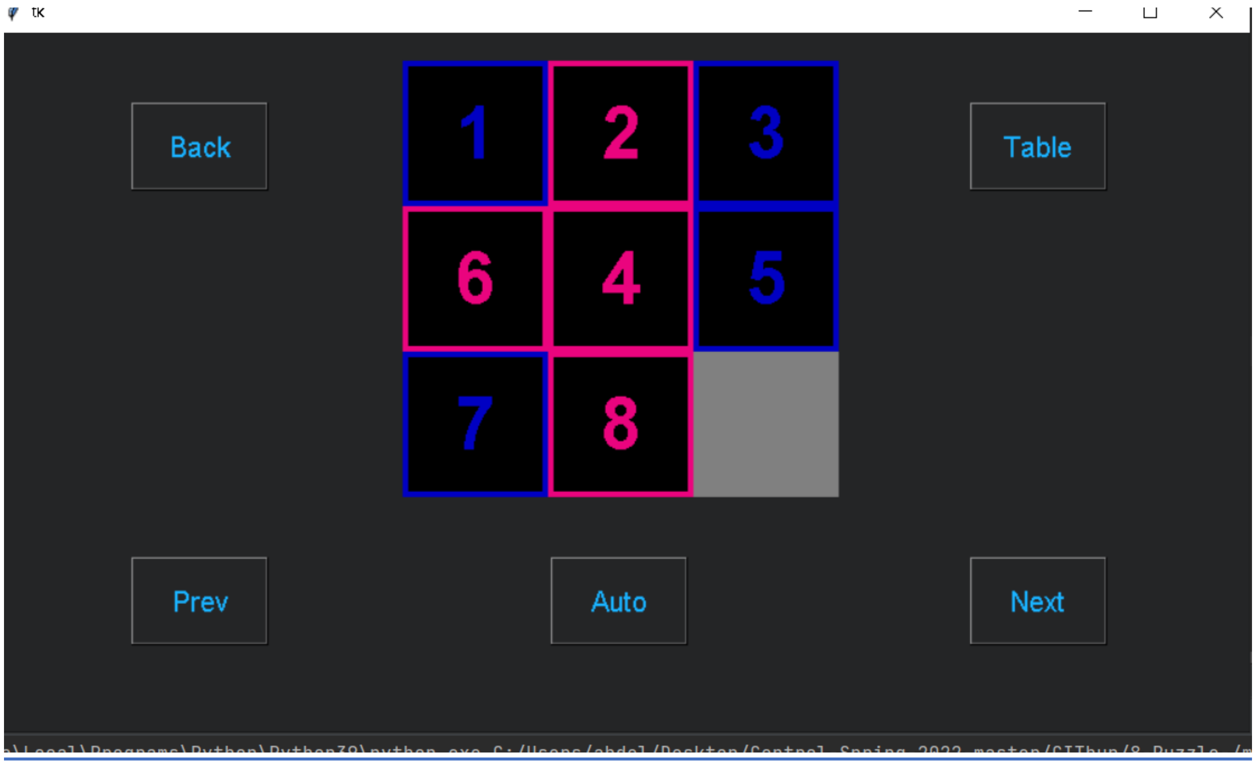
Bfs:


tk
Back
1
2
3
Table
4
5
6
7
8
Prev
Auto
Next

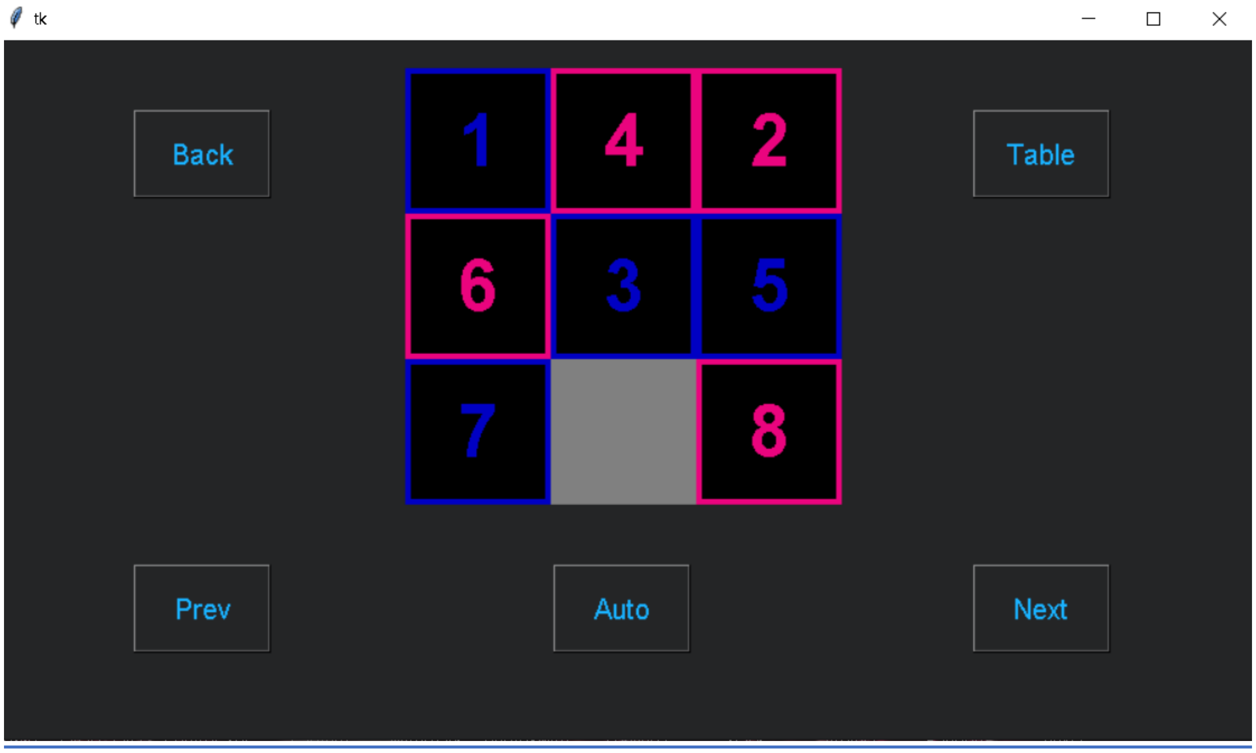

tk
Back
1
2
3
Table
4
5
6
7
8
Prev
Auto
Next

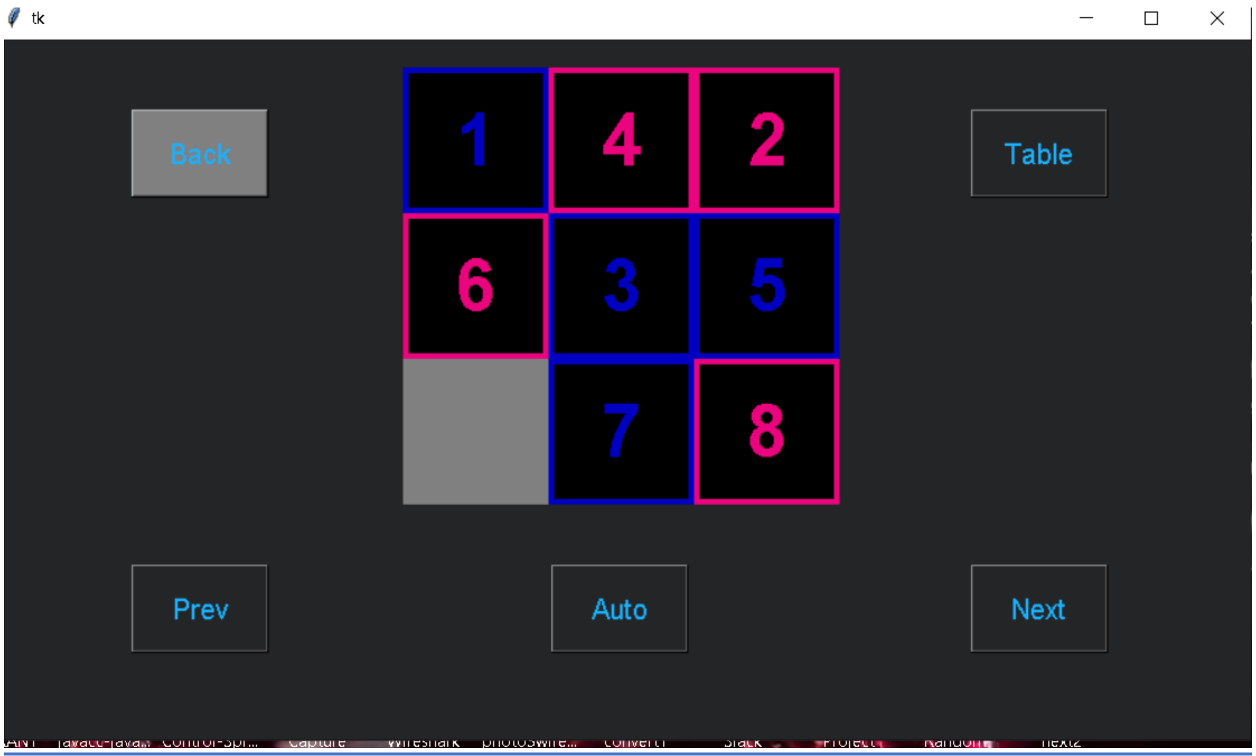

tk
Back
Table
1
2
3
6
4
5
7
8
Prev
Auto
Next

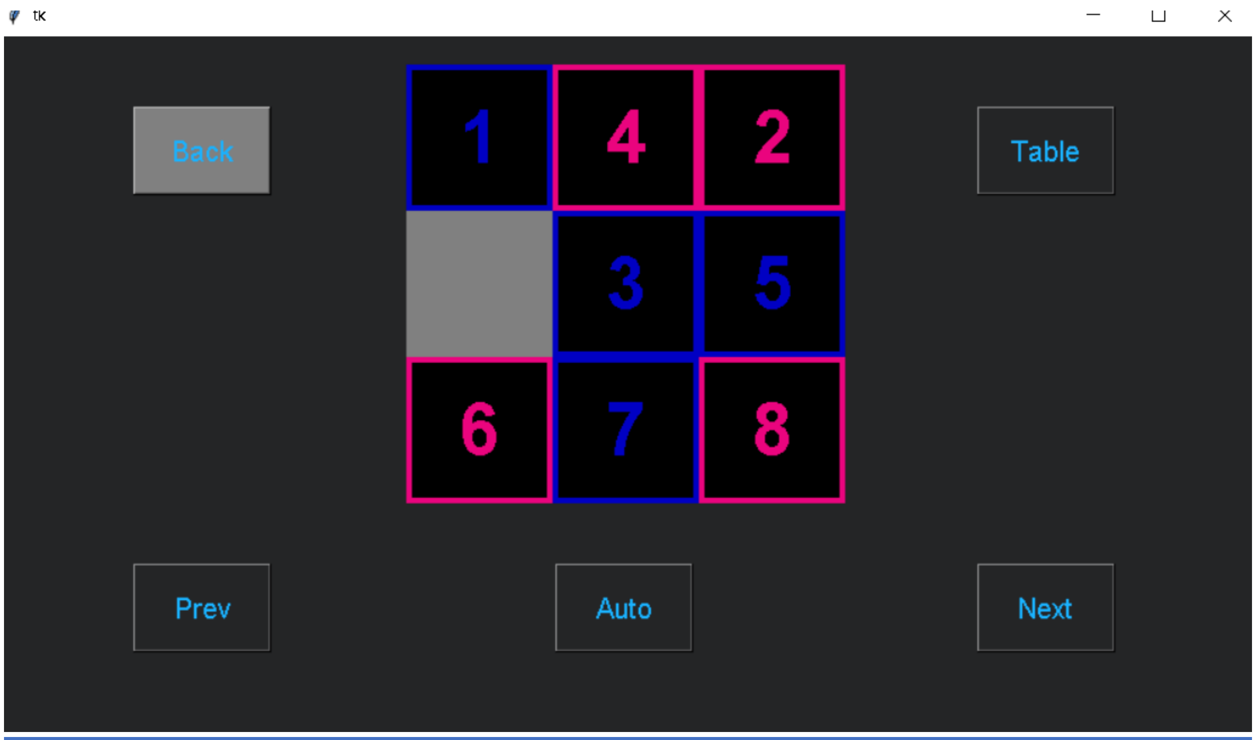

tk
Back
Table
1
2
3
6
4
5
7
8
Prev
Auto
Next

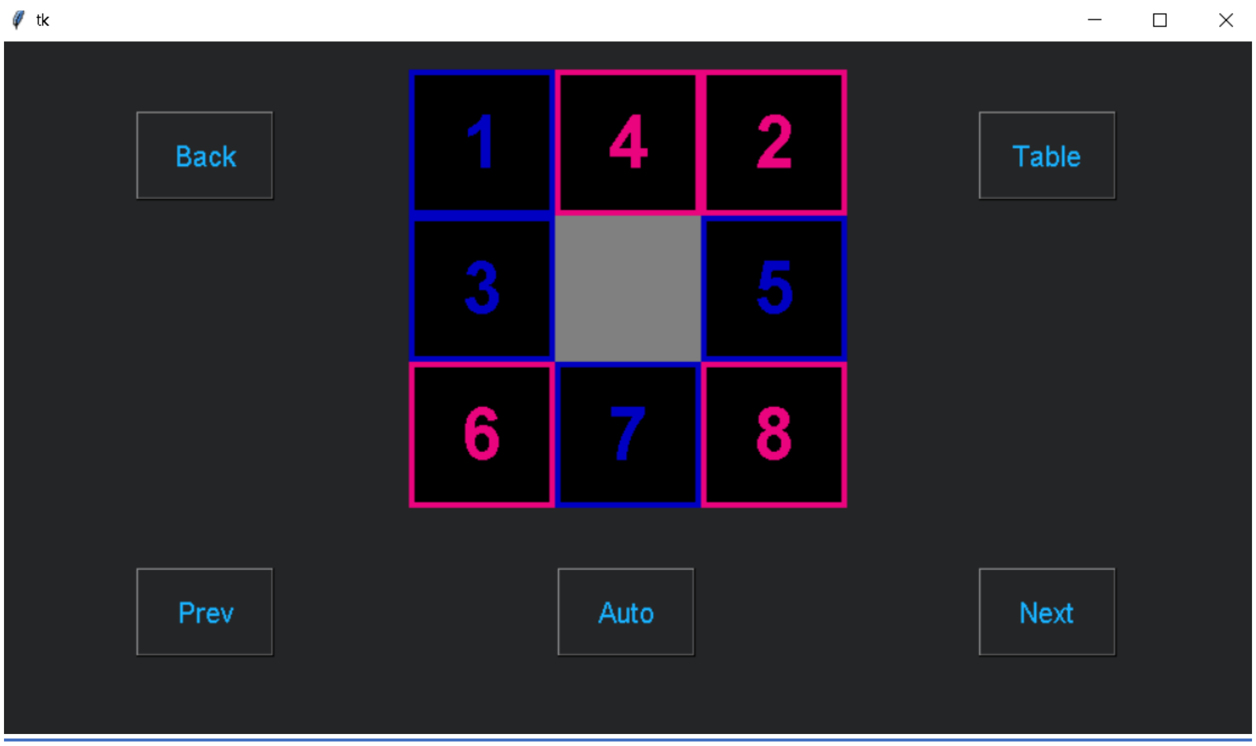


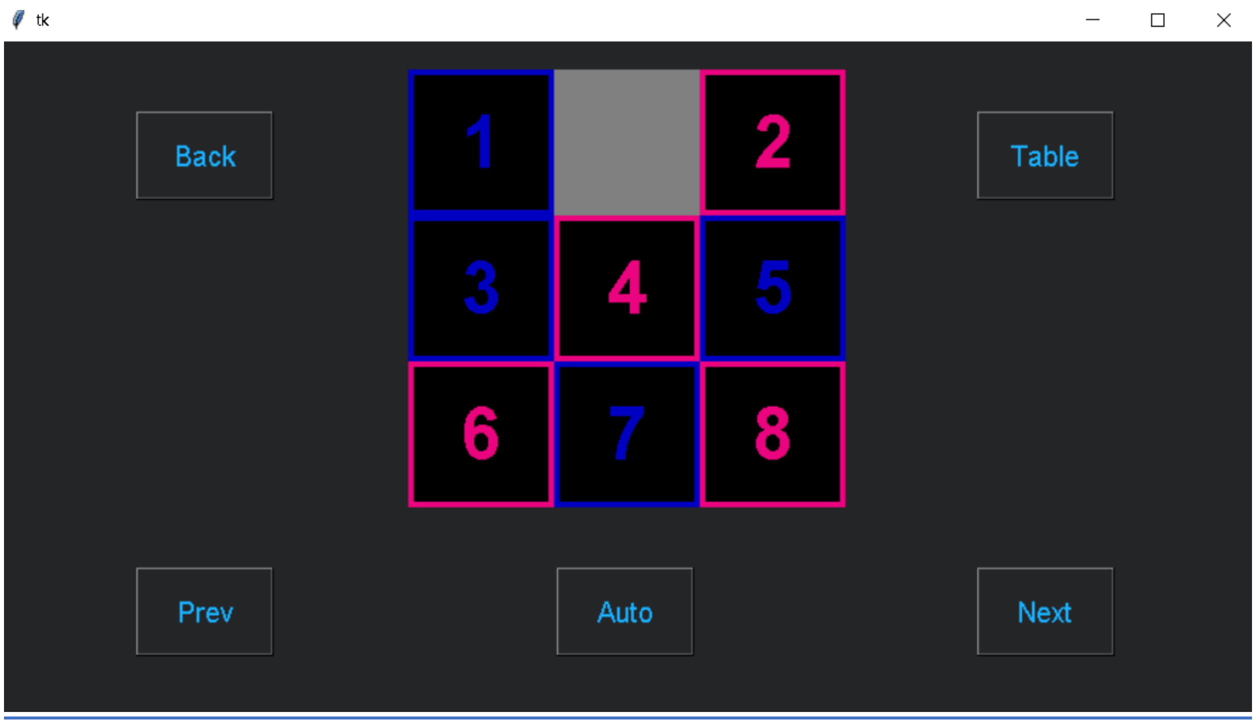
```
a\Local\Programs\Python\Python38\python.exe C:/Users/abdel/Desktop/Control Spring 2022 master/GITHub/8 Puzzle /m
```

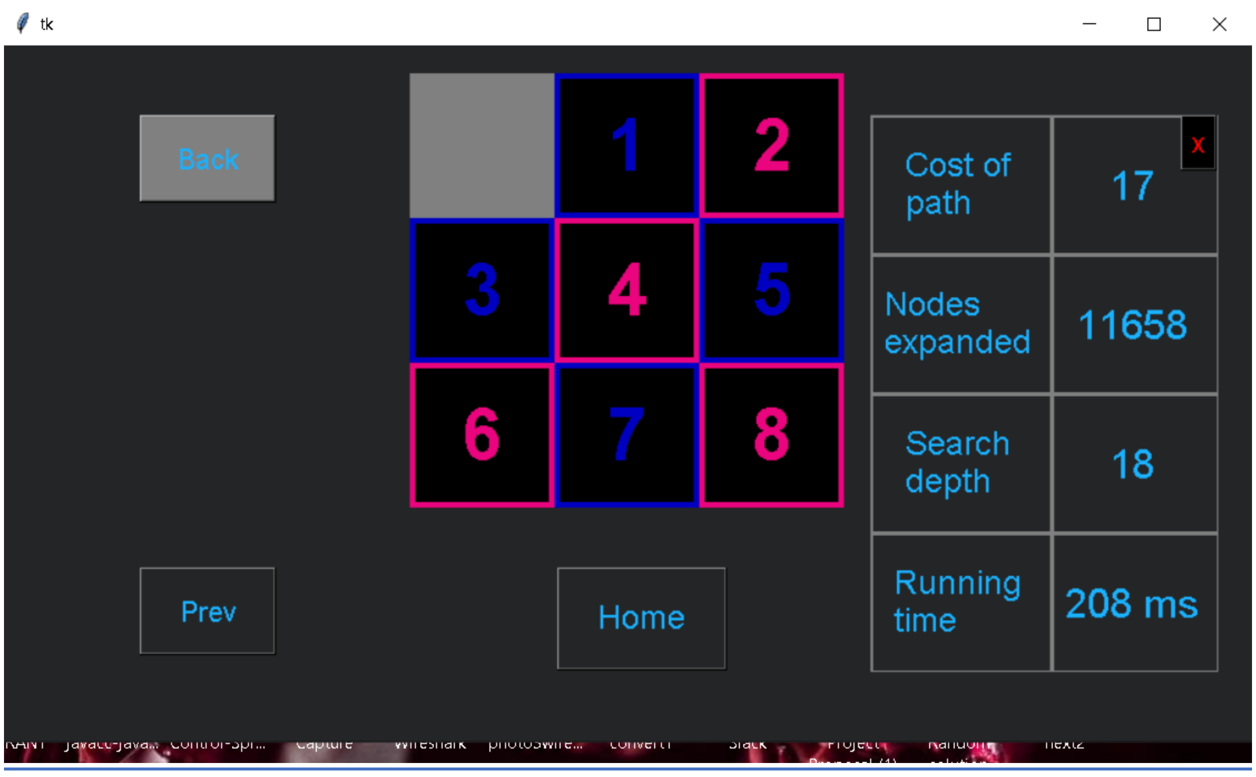



tk
Back
1
4
2
6
3
5
7
8
Table
Prev
Auto
Next


tk
Back
1
4
2
3
5
6
7
8
Table
Prev
Auto
Next

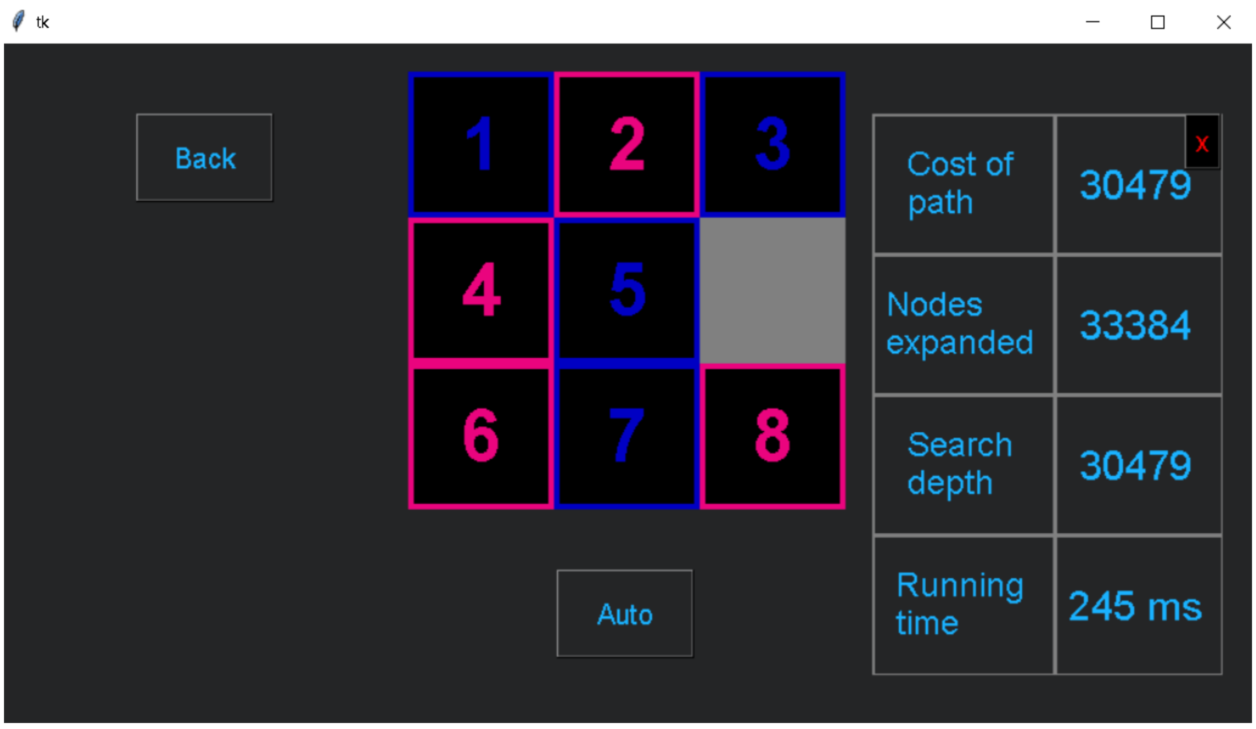

tk
Back
1
4
2
Table
3
5
6
7
8
Prev
Auto
Next


tk
Back
1
2
Table
3
4
5
6
7
8
Prev
Auto
Next



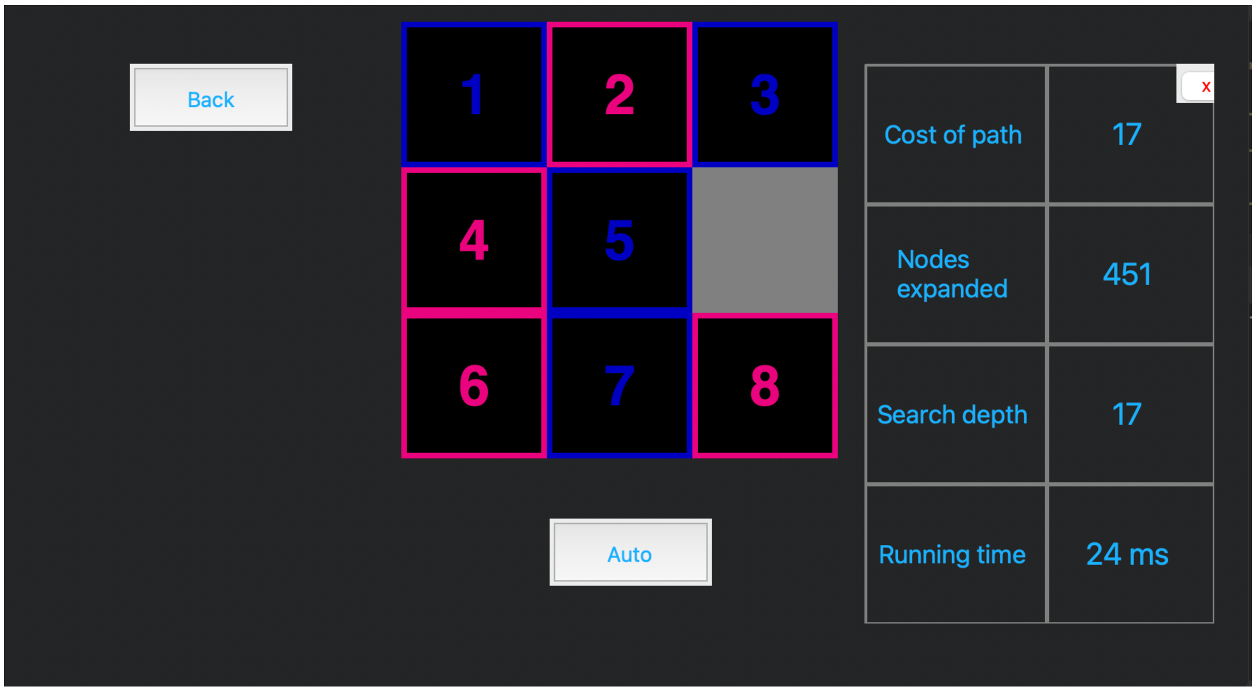
Cost of path	17
Nodes expanded	11658
Search depth	18
Running time	208 ms

Dfs:



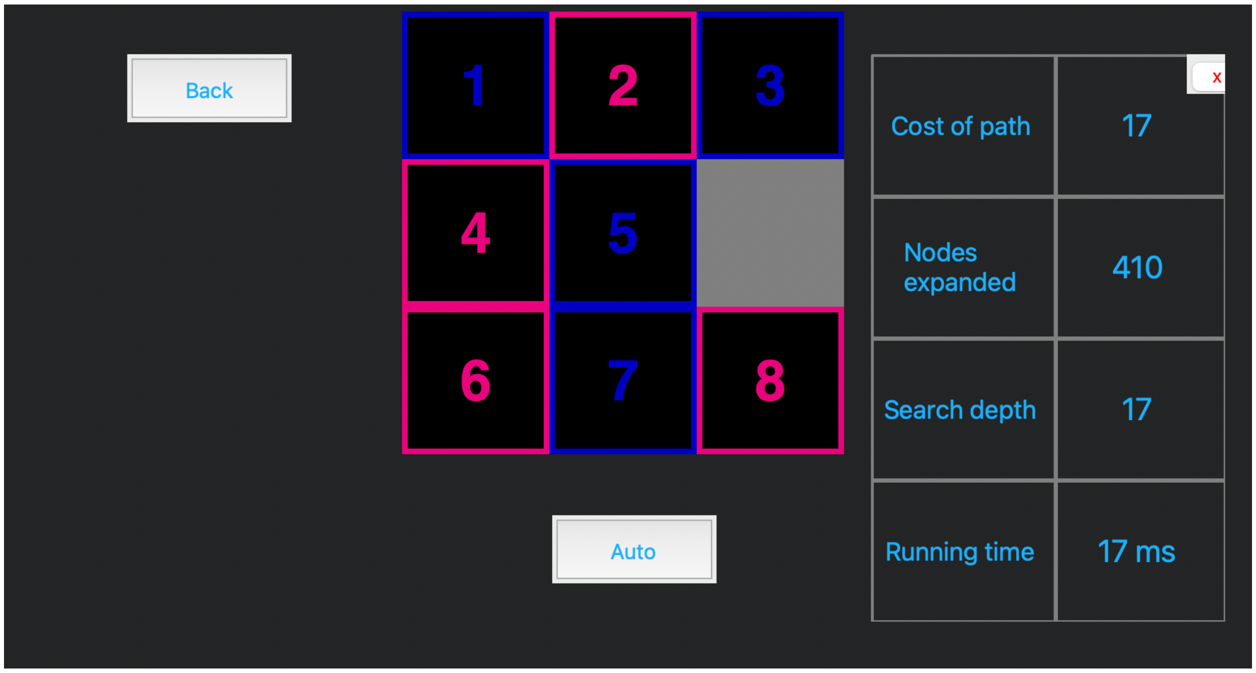
Cost of path	30479
Nodes expanded	33384
Search depth	30479
Running time	245 ms

A* Euclidean:



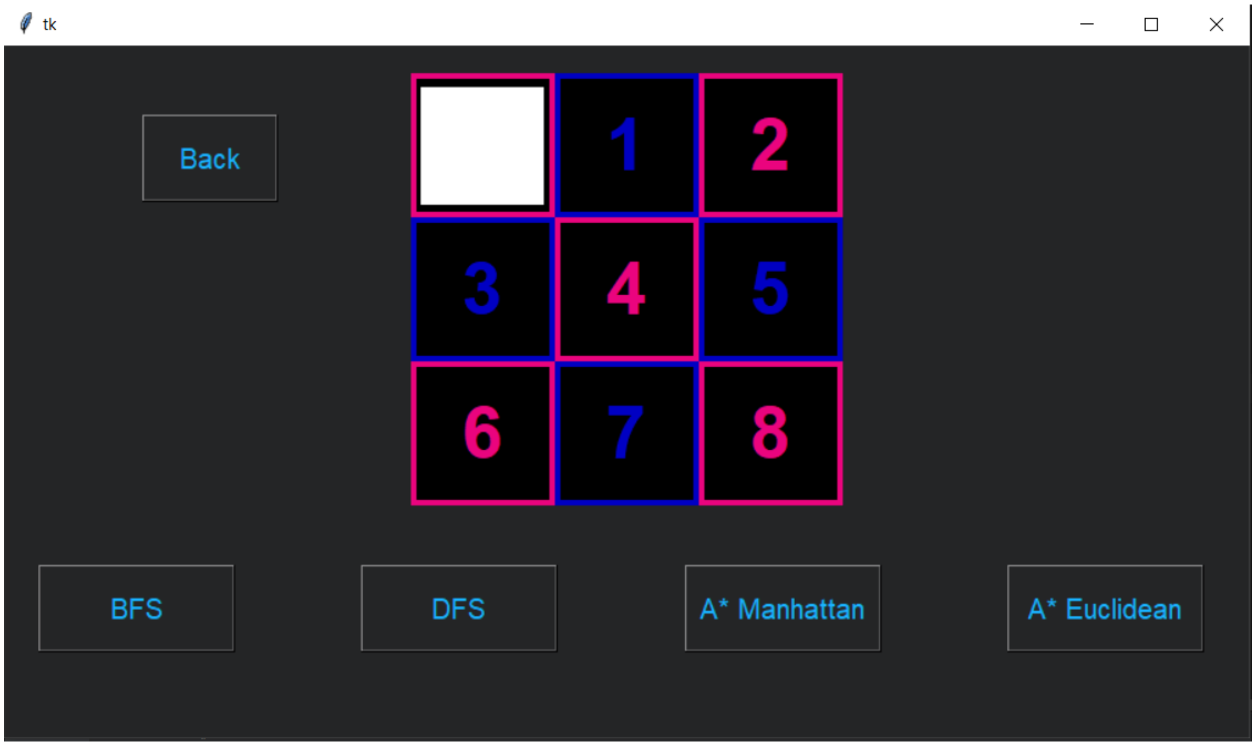
Cost of path	17
Nodes expanded	451
Search depth	17
Running time	24 ms

A* Manhattan:

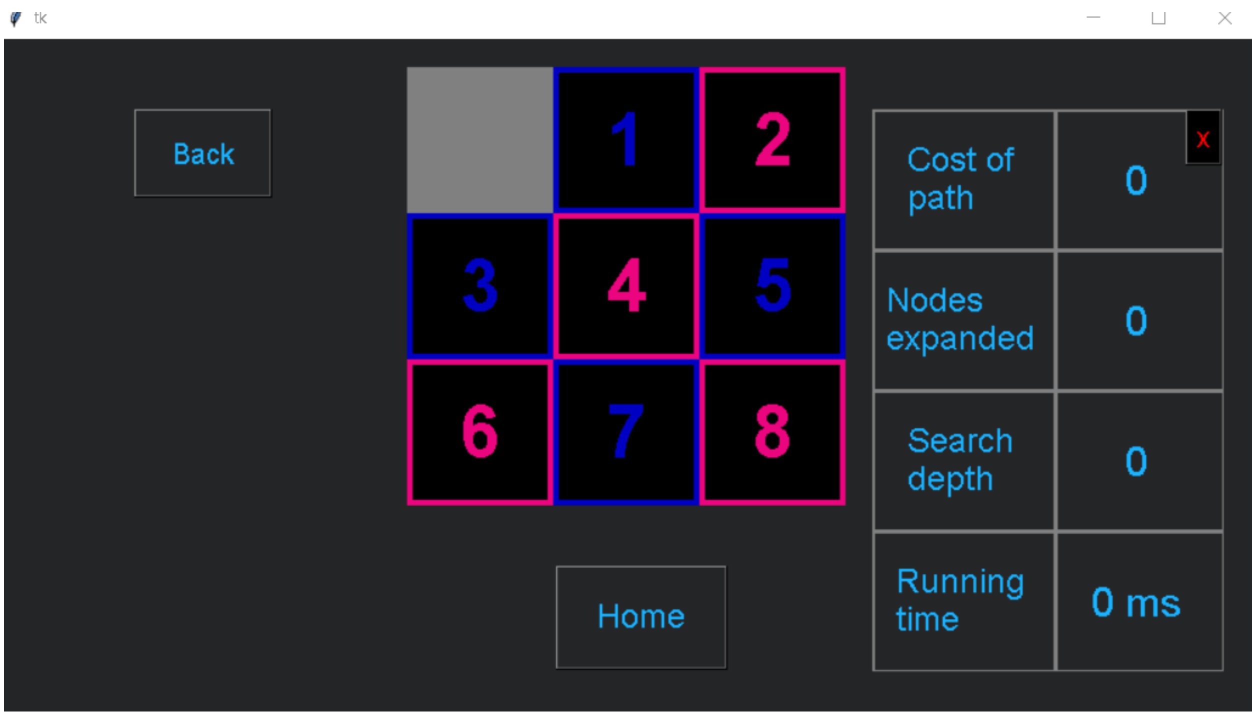


Cost of path	17
Nodes expanded	410
Search depth	17
Running time	17 ms

Second case (input by user):

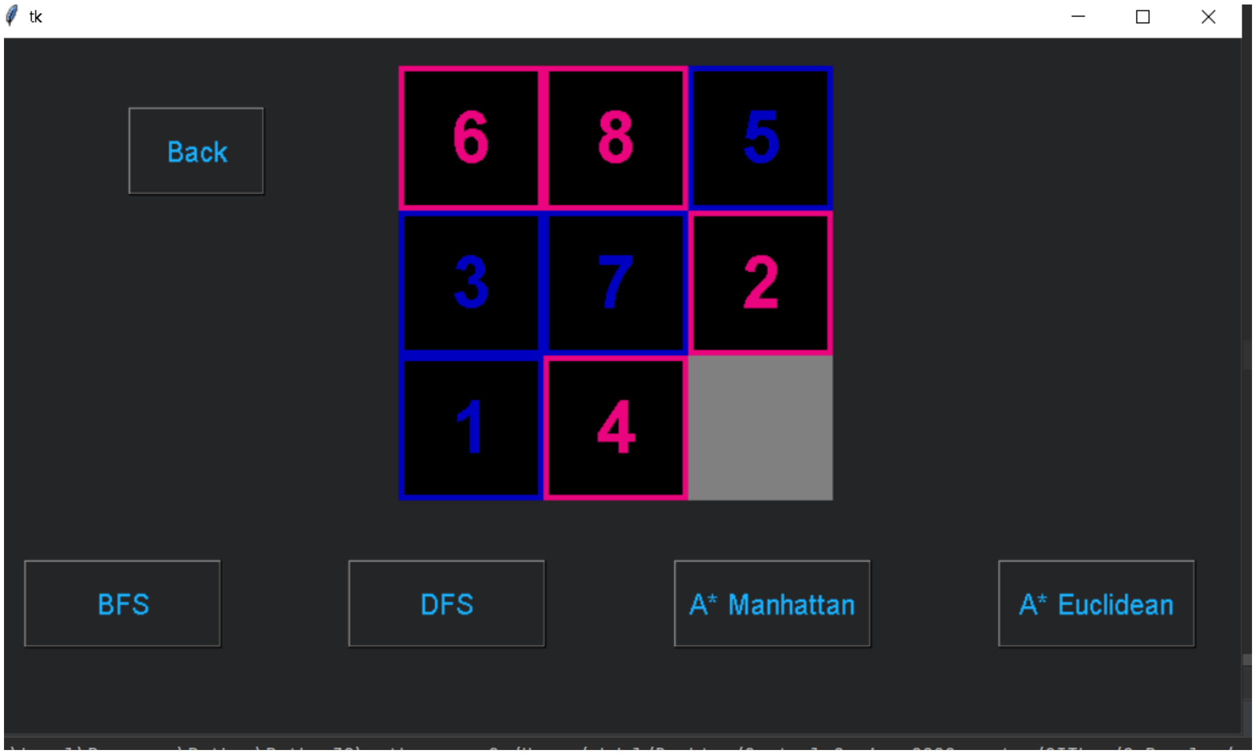


Bfs , dfs , A* Euclidean , A* Manhattan :

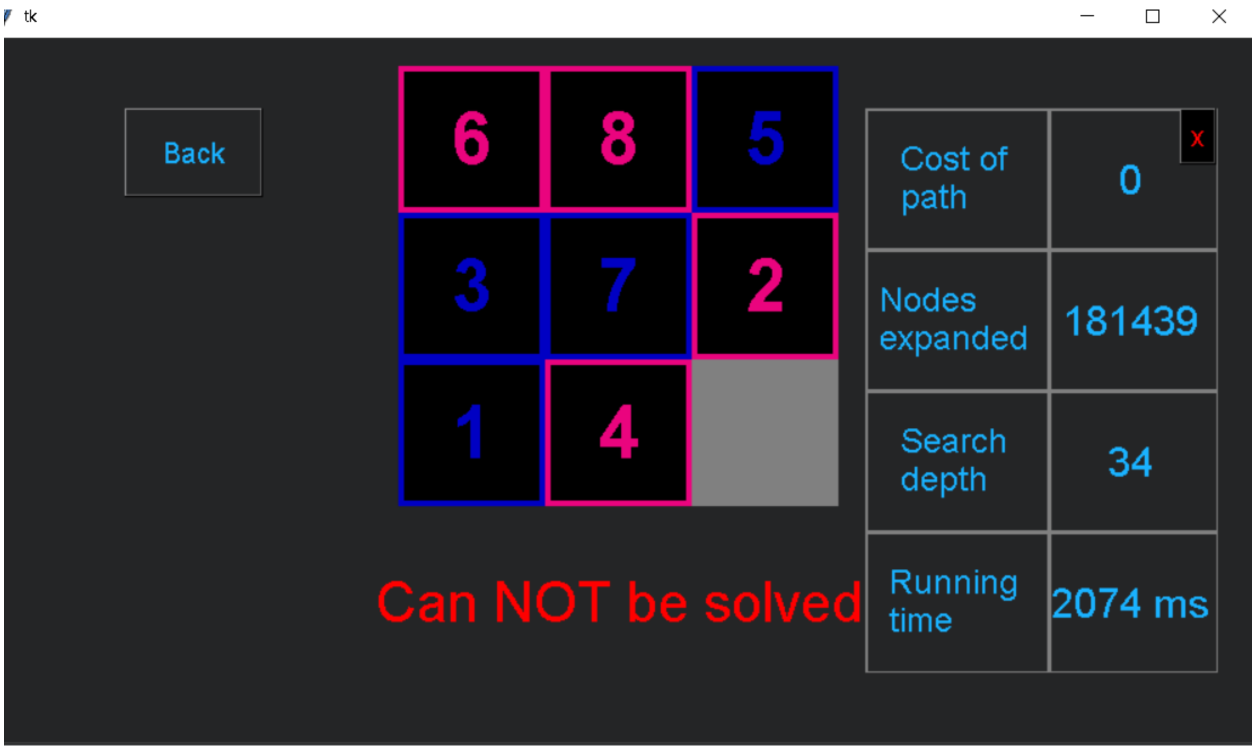


Cost of path	0
Nodes expanded	0
Search depth	0
Running time	0 ms

Third test case (Random input No solution):



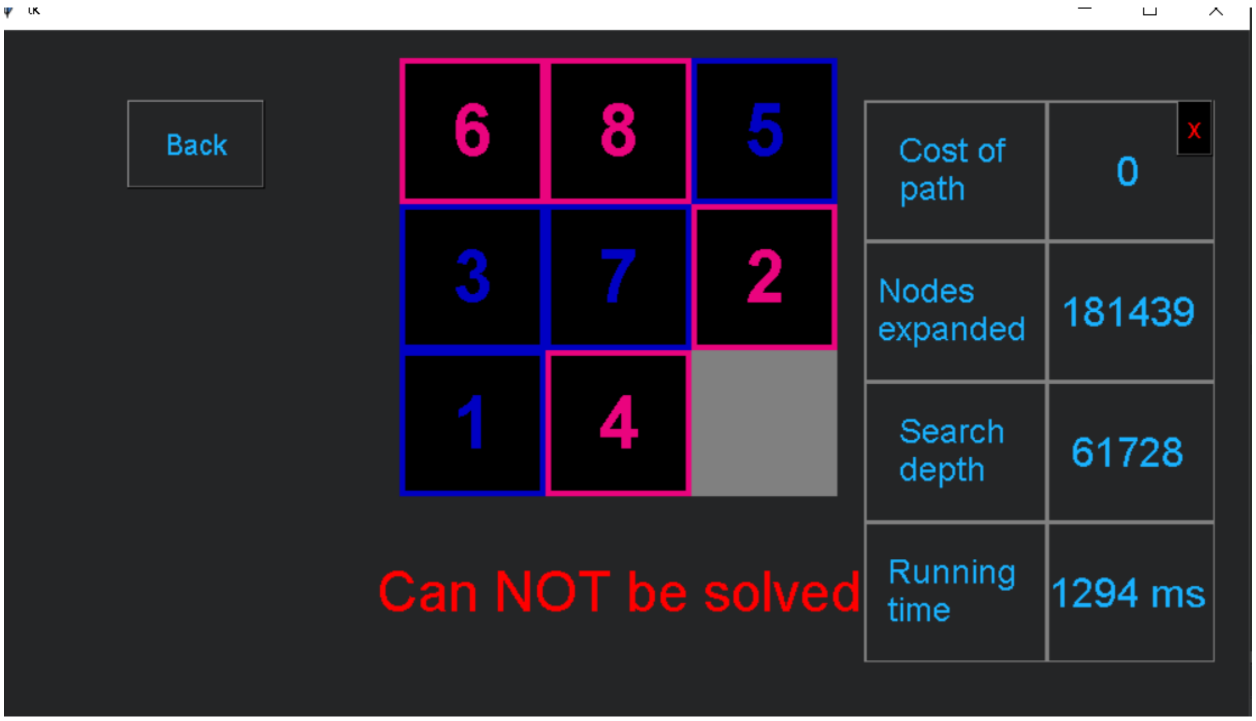
Bfs:



Cost of path	0
Nodes expanded	181439
Search depth	34
Running time	2074 ms

Can NOT be solved

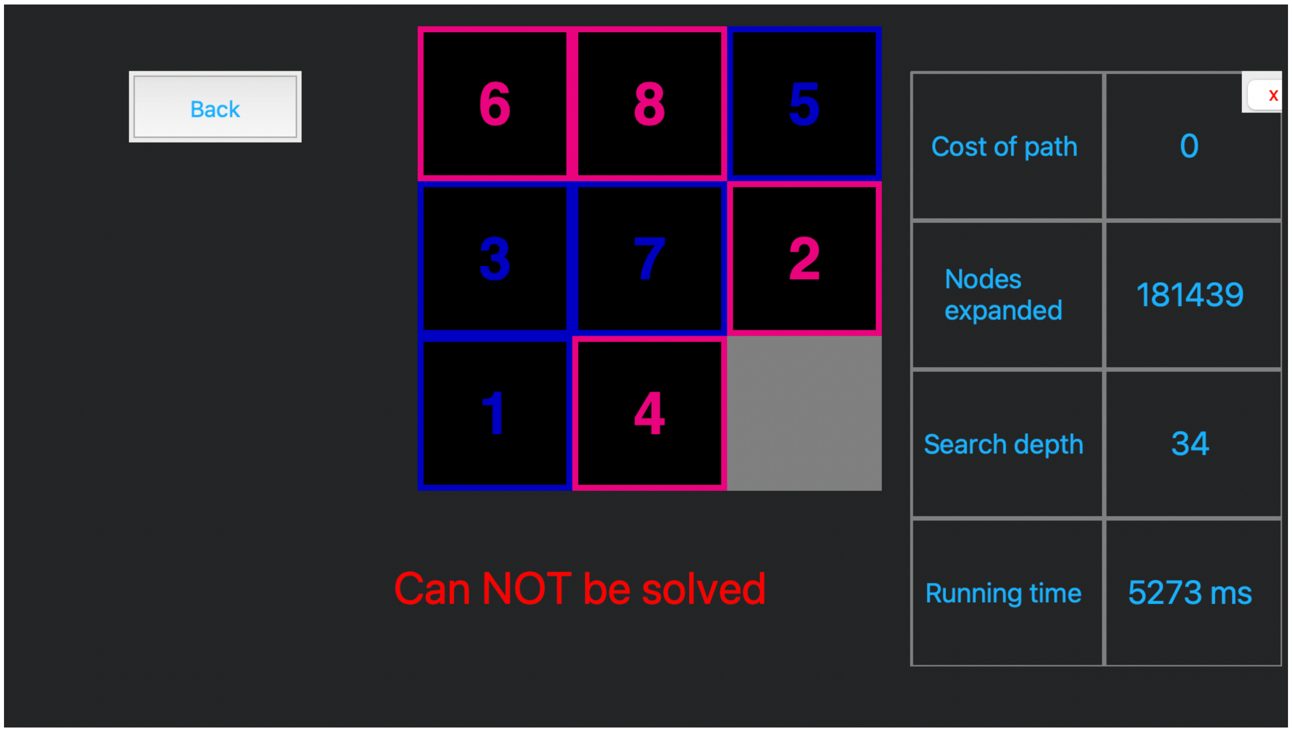
Dfs:



Can NOT be solved

Cost of path	0
Nodes expanded	181439
Search depth	61728
Running time	1294 ms

A* Euclidean:

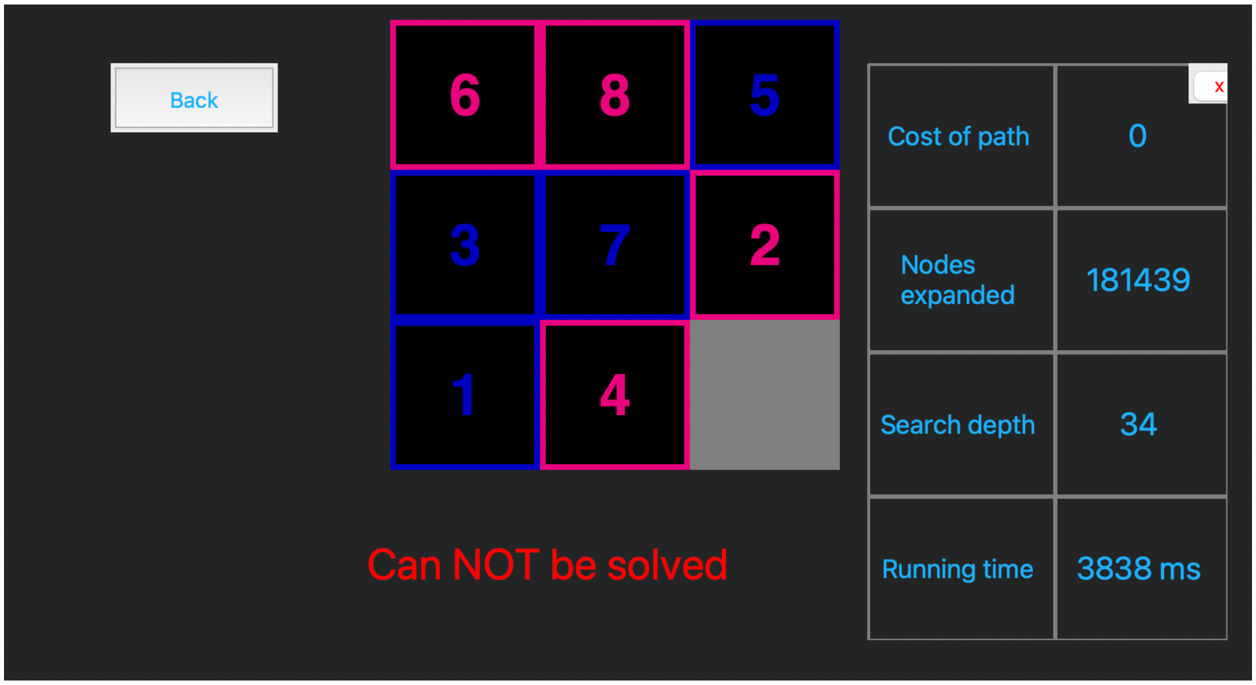


6	8	5
3	7	2
1	4	

Can NOT be solved

Cost of path	0
Nodes expanded	181439
Search depth	34
Running time	5273 ms

A* Manhattan:

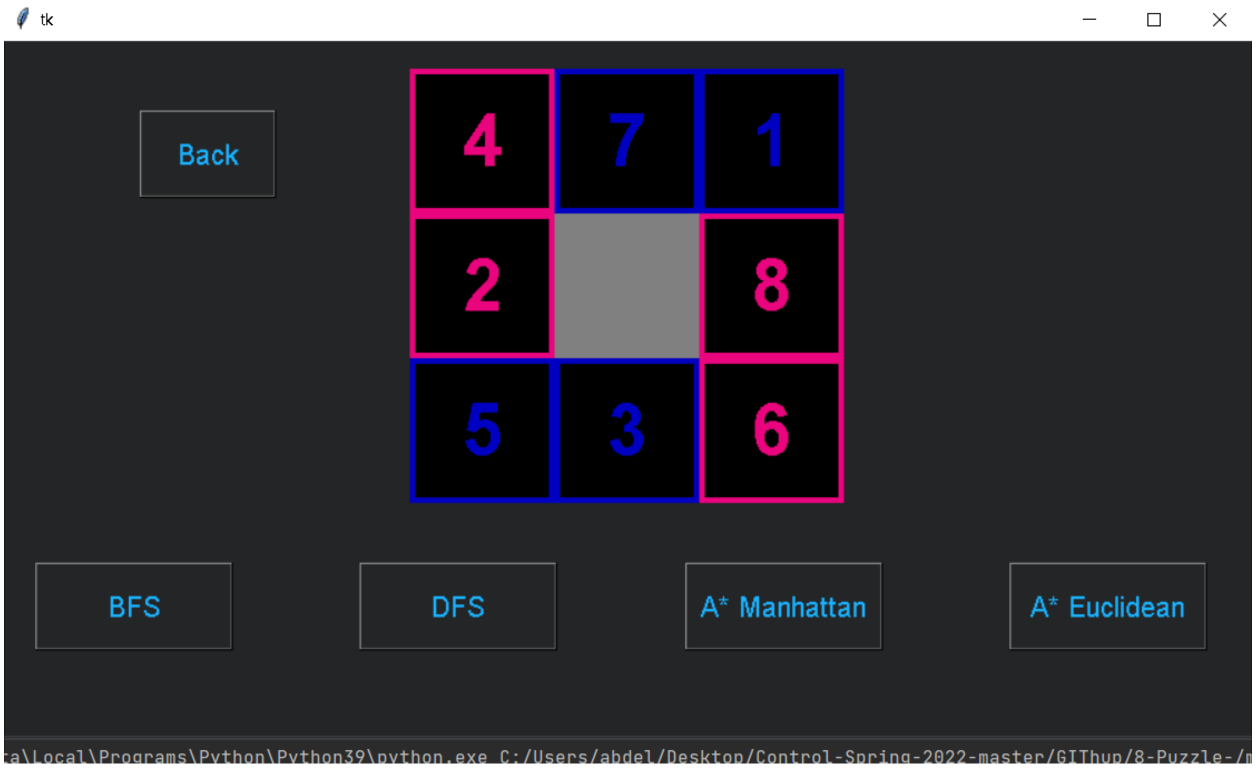


6	8	5
3	7	2
1	4	

Can NOT be solved

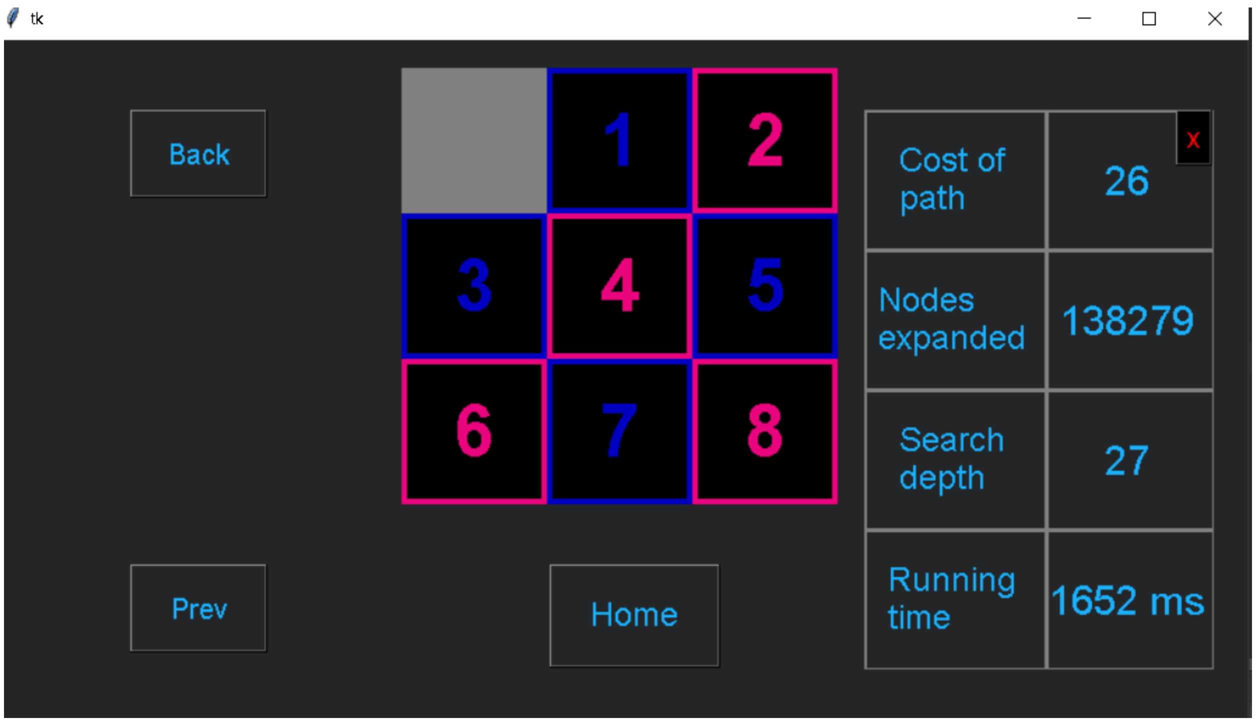
Cost of path	0
Nodes expanded	181439
Search depth	34
Running time	3838 ms

Fouth test case (Random input):



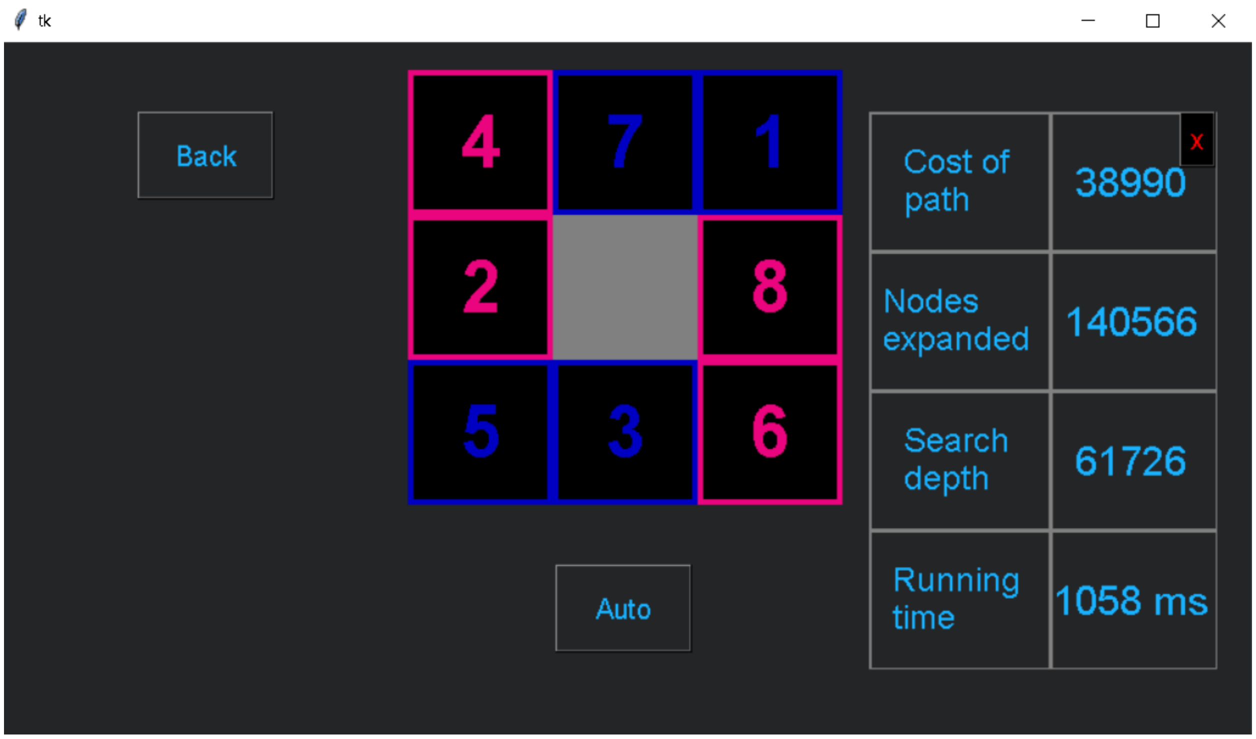
```
a\Local\Programs\Python\Python39\python.exe C:/Users/abdel/Desktop/Control-Spring-2022-master/GIThup/8-Puzzle-/
```

Bfs:



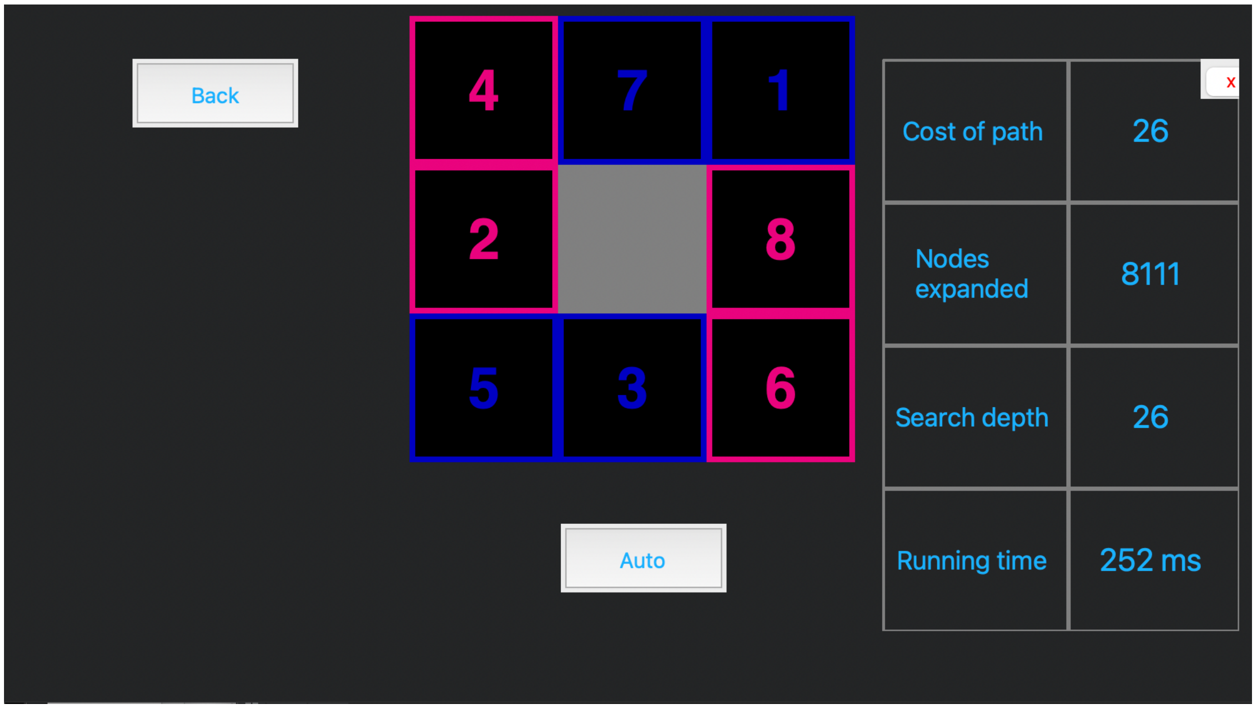
Cost of path	26
Nodes expanded	138279
Search depth	27
Running time	1652 ms

Dfs:



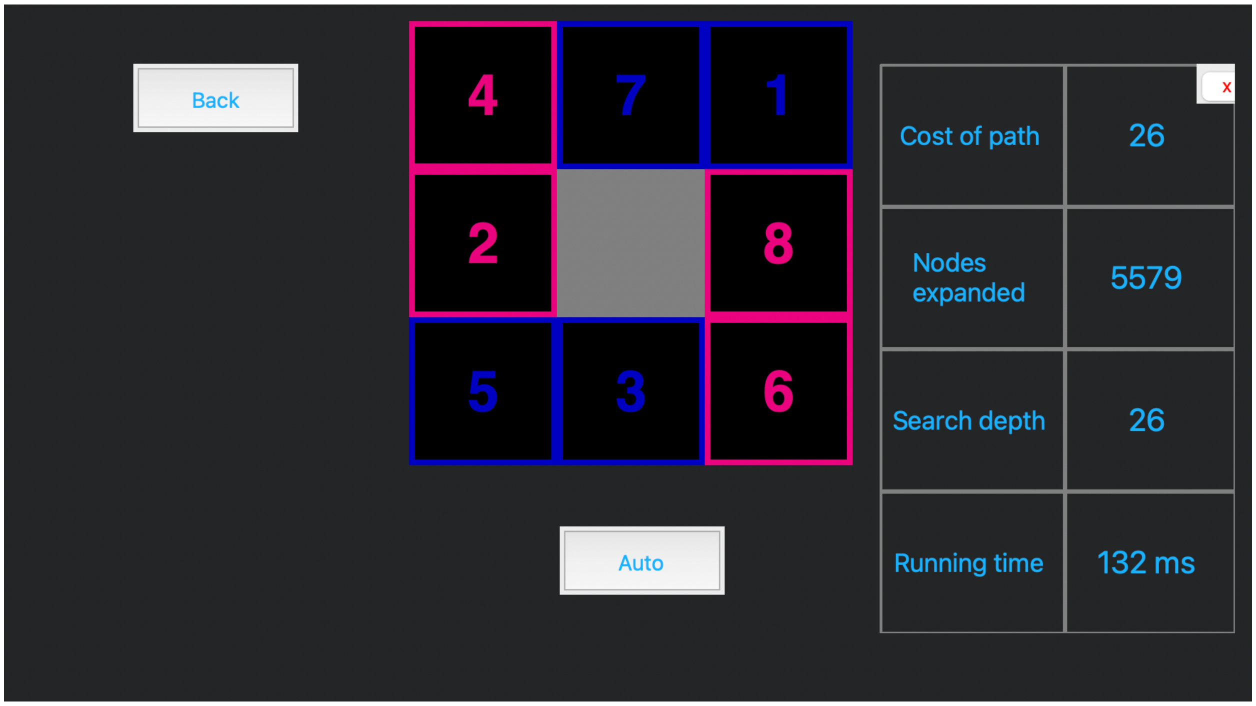
Cost of path	38990
Nodes expanded	140566
Search depth	61726
Running time	1058 ms

A* Euclidean:



Cost of path	26
Nodes expanded	8111
Search depth	26
Running time	252 ms

A* Manhattan:



Cost of path	26
Nodes expanded	5579
Search depth	26
Running time	132 ms

6. Comparison:

A. For compex input:

	cost of path	nodes expanded	search depth	running time (ms)
bfs	26	138279	27	1278
dfs	38990	140566	61726	772
A* Euclidean	26	8111	26	252
A* Manhattan	26	5579	26	153

B. For simple input:

	cost of path	nodes expanded	search depth	running time (ms)
bfs	17	11658	18	121
dfs	30479	33384	30479	207

A* Euclidean	17	451	17	24
A* Manhatten	17	410	17	17

C. For no solution input:

	cost of path	**nodes expanded**	**search depth**	**running time (ms)**
bfs	–	181439	34	2074
dfs	–	181439	61728	1294
A* Euclidean	–	181439	34	5273
A* Manhatten	–	181439	34	3838