Department of Computer Science

SRM University, AP - Amaravati
2018- 2022

CSE 413 Artificial Intelligence

'8 Puzzle Game Using BFS'

GROUP

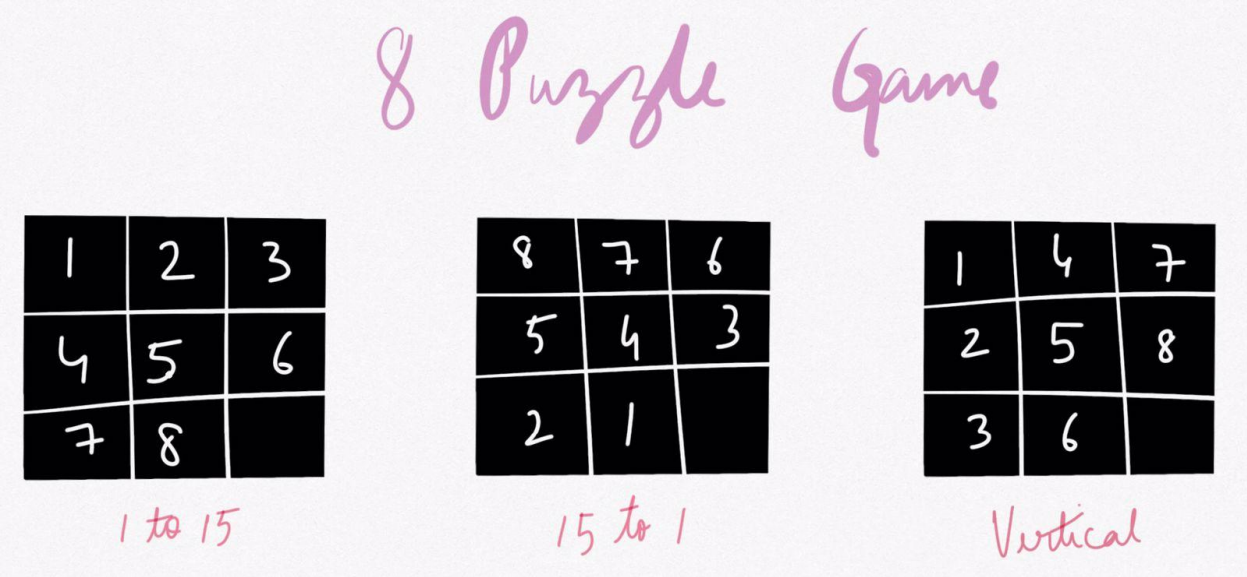
AP18110010338	Animesh Rituraj
AP18110010413	Sumanth Kongani
AP18110010436	Rahman Mohammad
AP18110010433	M Vamsikrishna
AP18110010403	Krishna Chaitanya
AP18110010441	Harsha sai Y
AP18110010407	T. Anirudh
AP18110010406	Sumedha Moturi
AP18110010438	Barukula Snehitha
AP18110010416	Lakshmi Vallala
AP18110010642	Charan Sandeep
AP18110010370	Veeranki Jagadev
AP18110010334	Yashveer Srivastava
AP18110010382	Devendra Kumar Dewangan

1. ABSTRACT

The proposed project is an implementation of solving the 8-puzzle problem using the naive search strategy BFS (Breadth-First Search). Noyes Palmer Chapman designed and popularised the 8-puzzle game in the 1870s. It is played on a three-by-three grid with eight square blocks labelled 1 through 8 and one blank square. The objective is to rearrange the blocks such that they are in the correct sequence. The order of moving the blocks can be horizontal or vertical into the empty square.

2. INTRODUCTION

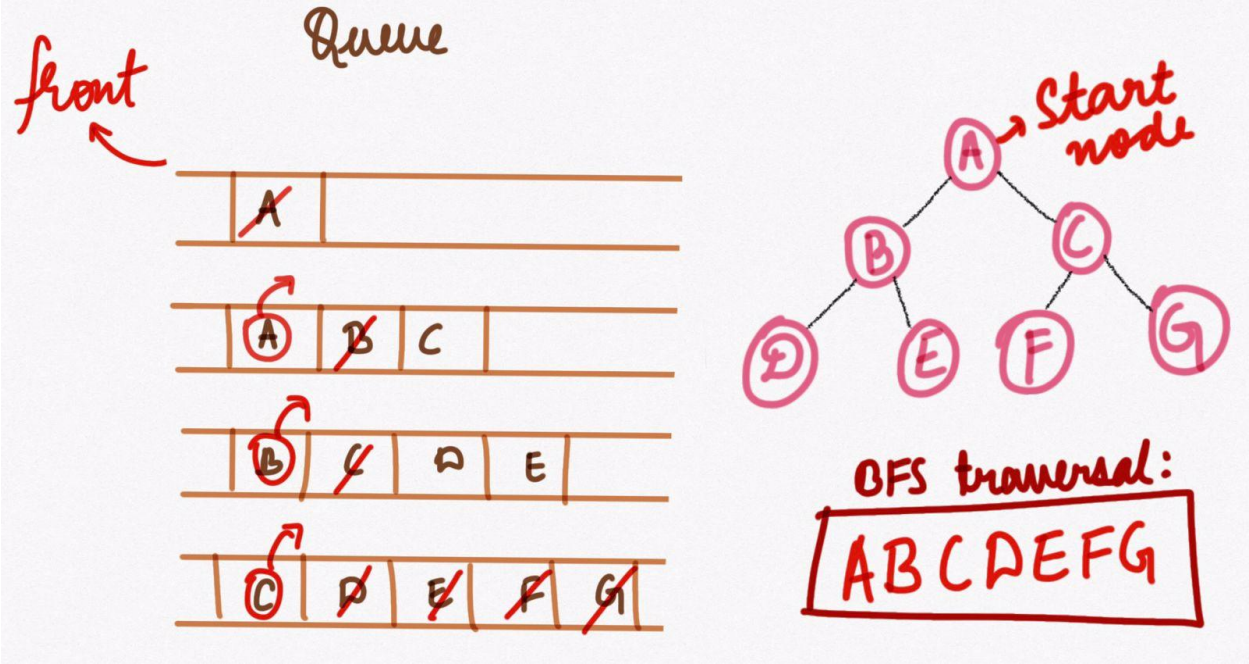
The 8 puzzle is the smaller version of the famous gem puzzle or 15 puzzle game. It is one of the oldest combination puzzle games which can be solved by achieving a particular arrangement from an initial scrambled arrangement. The 8 puzzle has 8 square tiles that are numbered 1 through 8 in a 3x3 frame and one blank unoccupied tile to move the squares around. The tiles in the same row or column of the unoccupied position can be moved by sliding them horizontally or vertically. The goal of the puzzle is to place the tiles in ascending order. In the set of numbers {1,2,3,4,5,6,7,8,9}, the initial random ordered state and our ordered goal state are different permutations of the set. Switching a pair (x,y) or the transposition of (x,y) changes the permutation leaving us closer to our final arrangement. Even though the puzzle is named 8 puzzles for the number of tiles in the frame, the 8 puzzles may also be called a 9 puzzle, alluding to the total number of tiles. The puzzle is also played in one of the variant arrangements or patterns: 1 to 15, 15 to 1, vertical, skip odd to even, vertical odd-even and skip odd even. Some of these possibilities have been numerously proved to be impossible to arrange.



Proposed system

Given a 3×3 board with 8 tiles (every tile has one number from 1 to 8) and one blank tile. The objective is to place the numbers on tiles to match the final configuration using the space. We can slide four adjacent (left, right, above, and below) tiles into the blank tile.

We propose to solve the 8-puzzle using the famous graph traversal technique: Breadth-First Search (BFS). A Breadth-first search is an iterative approach exploring all the adjacent positions before exploring the next level moves. It makes use of a queue which might add too much memory while solving in a large search space.



Though deepening can solve the problems of DFS, it is inefficient as it takes multiple iterations of DFS.

```
// pseudocode
BFS(G, s)
        create a queue q
         q.push(s)
        add s to visited
         while q is not empty:
                w=q.dequeue()
                for all neighbors v of w
                        if v not visited
                                q.enqueue(v)
                                mark v as visited
```

TECHNOLOGIES/ LANGUAGES USED

Tkinter standard framework for binding python to a Graphical User Interface using TK GUI toolkit widget. Tkinter provides an object-oriented interface.

RELATED WORKS/LITERATURE SURVEY

Deniz 8 puzzle solver and tree visualiser is a web app that allows users to experiment with randomised states and supports different tree search techniques such as breadth-first search, A* Algorithm, Depth-first search, Iterative Deepening, Greedy and Uniform cost with the possibility of tuning iteration limit and depth limit. It also visualises the search tree that leads to the final state.
The 8 puzzle web app by murhafsousli was deliberately designed to understand how artificial intelligence algorithms can solve an n-puzzle game.
8 Puzzle is a game hosted on the play store by Micri-M who are dedicated to creating fun games.

The 8-puzzle problem involves moving the tiles on a 3*3 board to get all the numbers from an initial state to a goal state. In this, the board contains 8 numbered tiles and a blank tile. The set of all the possible arrangements in the problem space consists of 3,62,880 different configurations of the 8 tiles and blank space. There are numerous other search techniques and algorithms to solve the 8-puzzle game.

Using DFS-Method: We can perform this method using a state-space tree i.e. all states can be reached from the initial state. It follows the leftmost path from the root by considering the initial state. It uses a stack that is a FIFO structure.

Time complexity: O(b^m).
Space complexity: O(bm). i.e. linear space where is the max branching factor and m is the depth of the tree)

Using BFS-Method: We can perform this method on the state-space tree. It always finds a goal state nearest to the root. Regardless of what the initial state is, it attempts the same sequence of moves as DFS. It uses a queue that is a FIFO structure.

Time complexity: In the worst case, the time complexity is O(b^d).
Space complexity: O(b^d) where b is the branching factor and d is the depth of the tree.

Using Branch and Bound: it also called an approximate cost function to avoid searching in sub-trees that do not contain an answer node. It is similar to the BACK-TRACKING technique but uses the BFS-like search. It is classified into 3-types

- Live-Node
- E-Node
- Dead Node

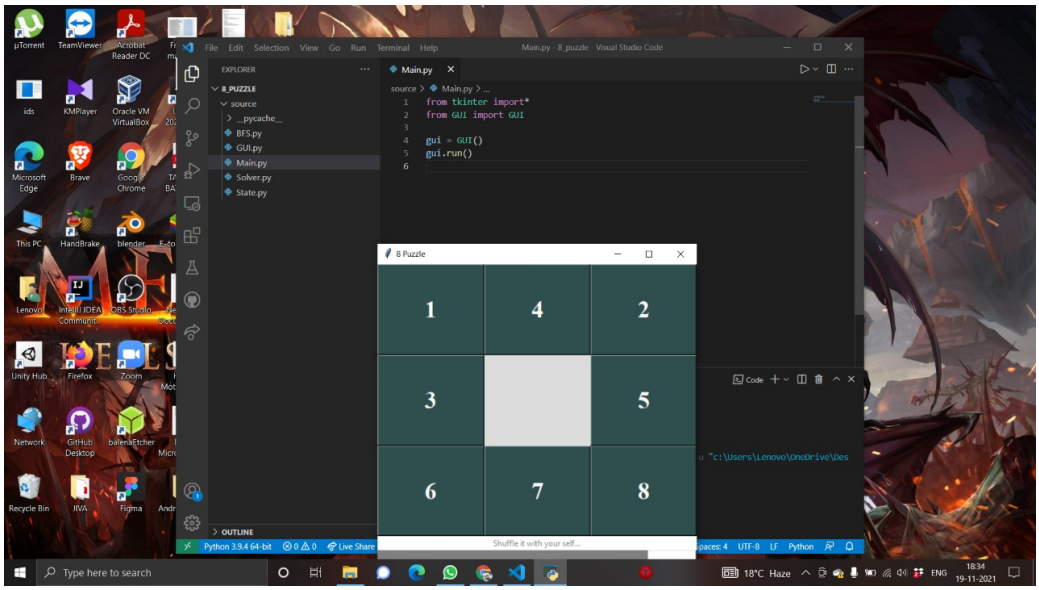
We can solve the 8-puzzle problem using the “heuristic search procedure”. It is a Best for Search procedure. In this search, we will approach.

- Or-Graph
- A* Graph.

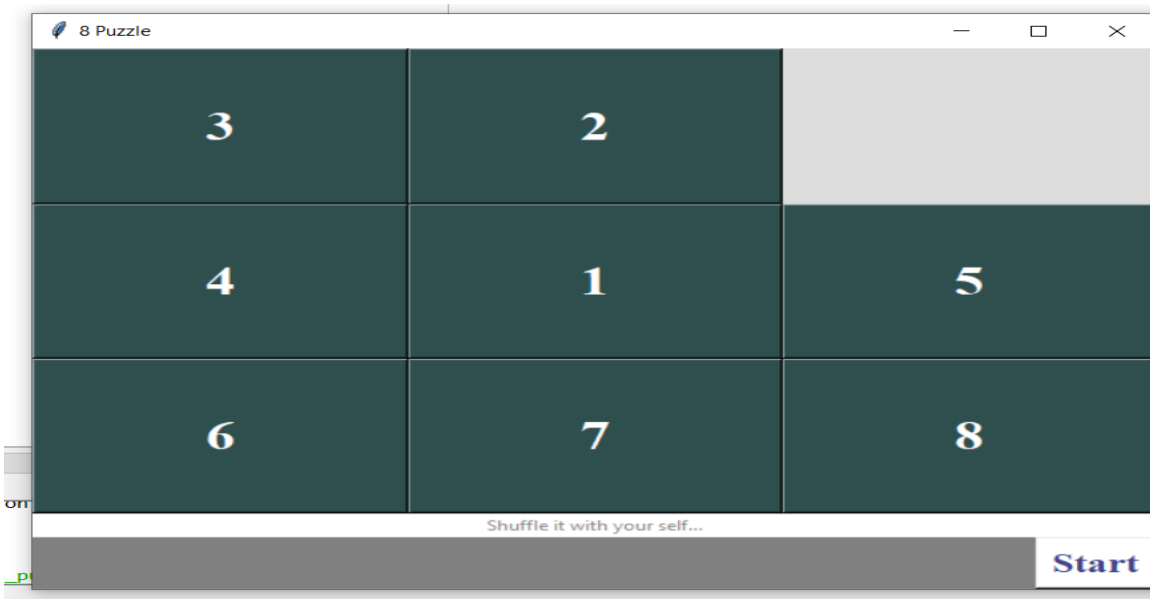
6. RESULTS/PERFORMANCE ANALYSIS

An individual who logs into the platform can successfully access and visualise the 8-puzzle using BFS. Below are the snapshots of the implementation of the game by our group members.

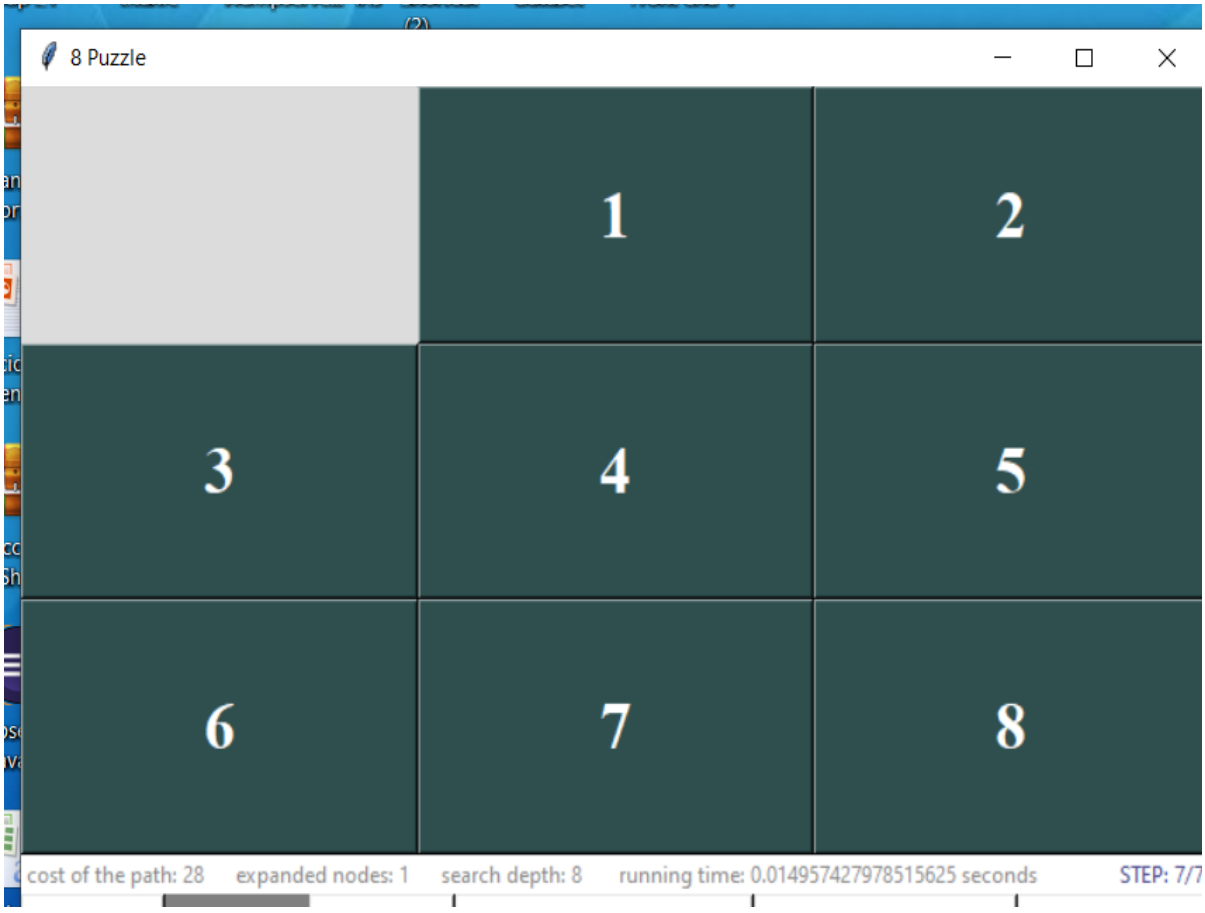
Animesh Rituraj



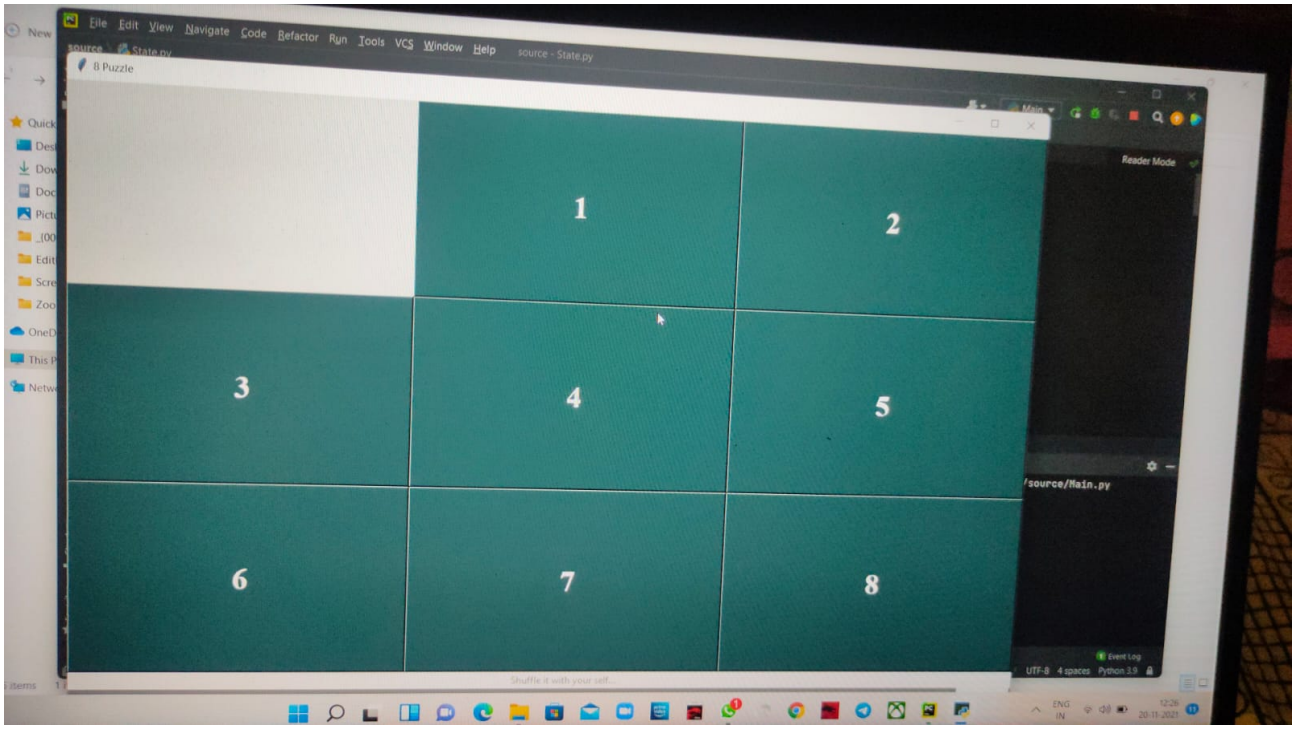
Sumanth Kongani



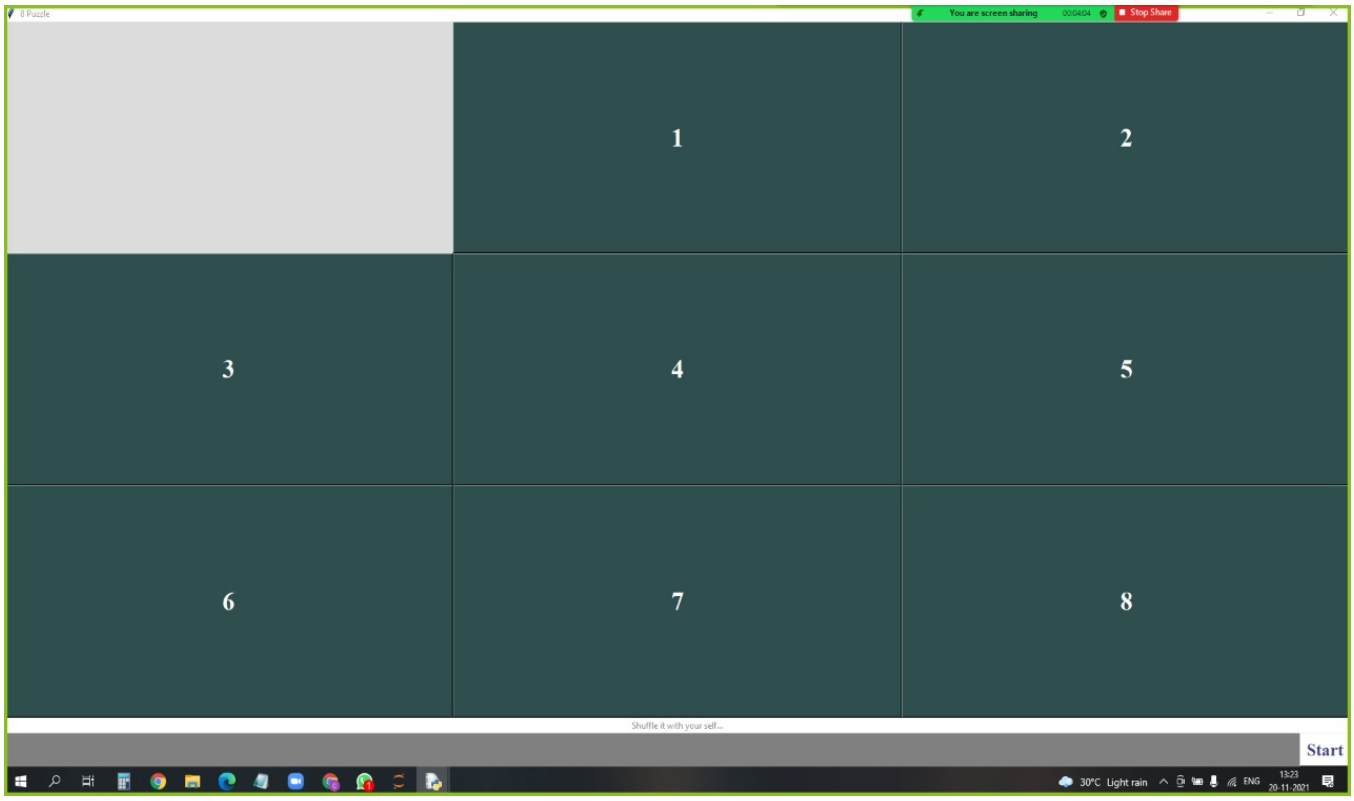
Khaleelur Rahman Mohammad



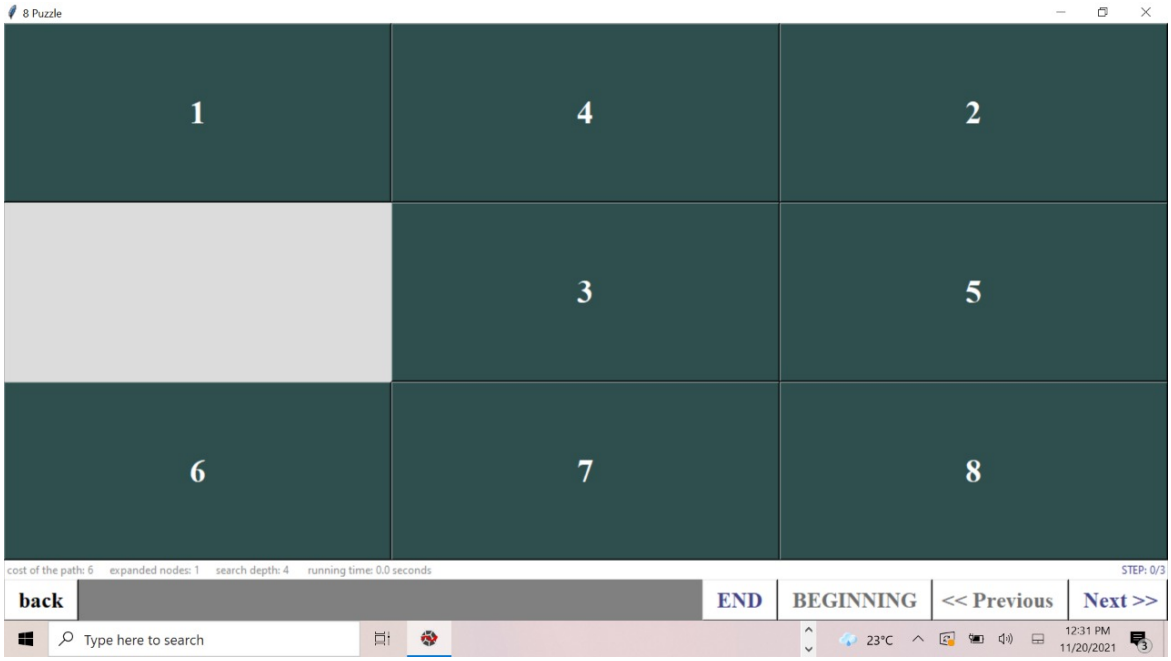
M Vamsikrishna



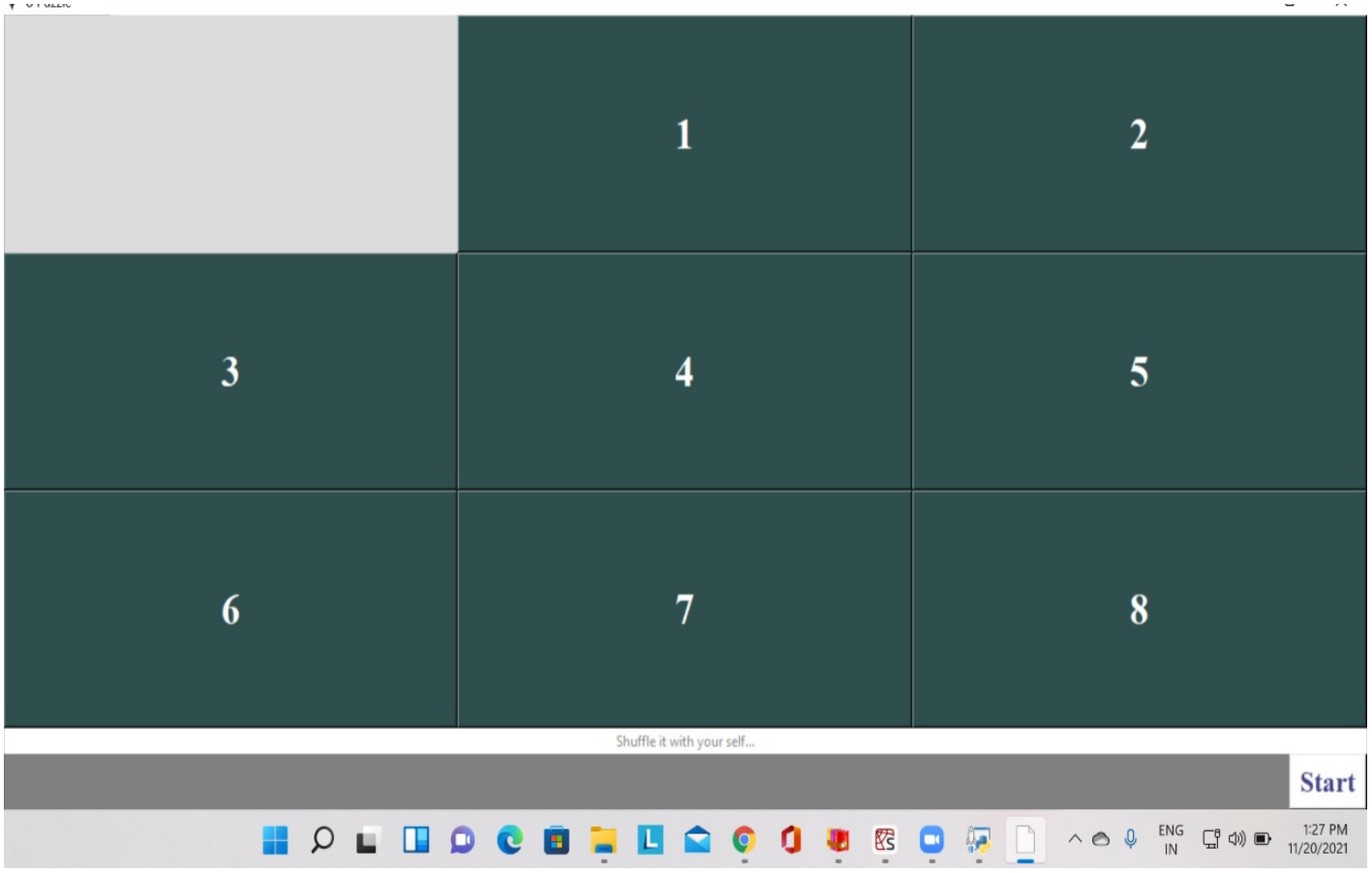
Krishna Chaitanya



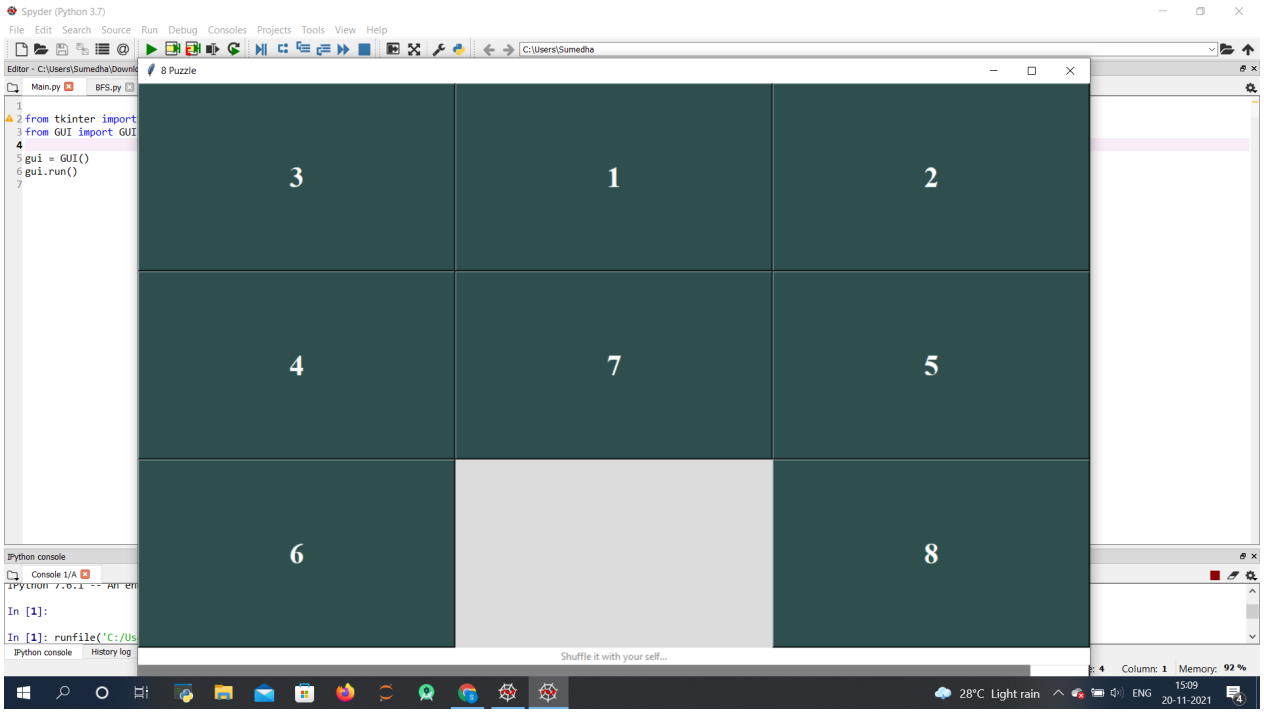
Harsha sai Y



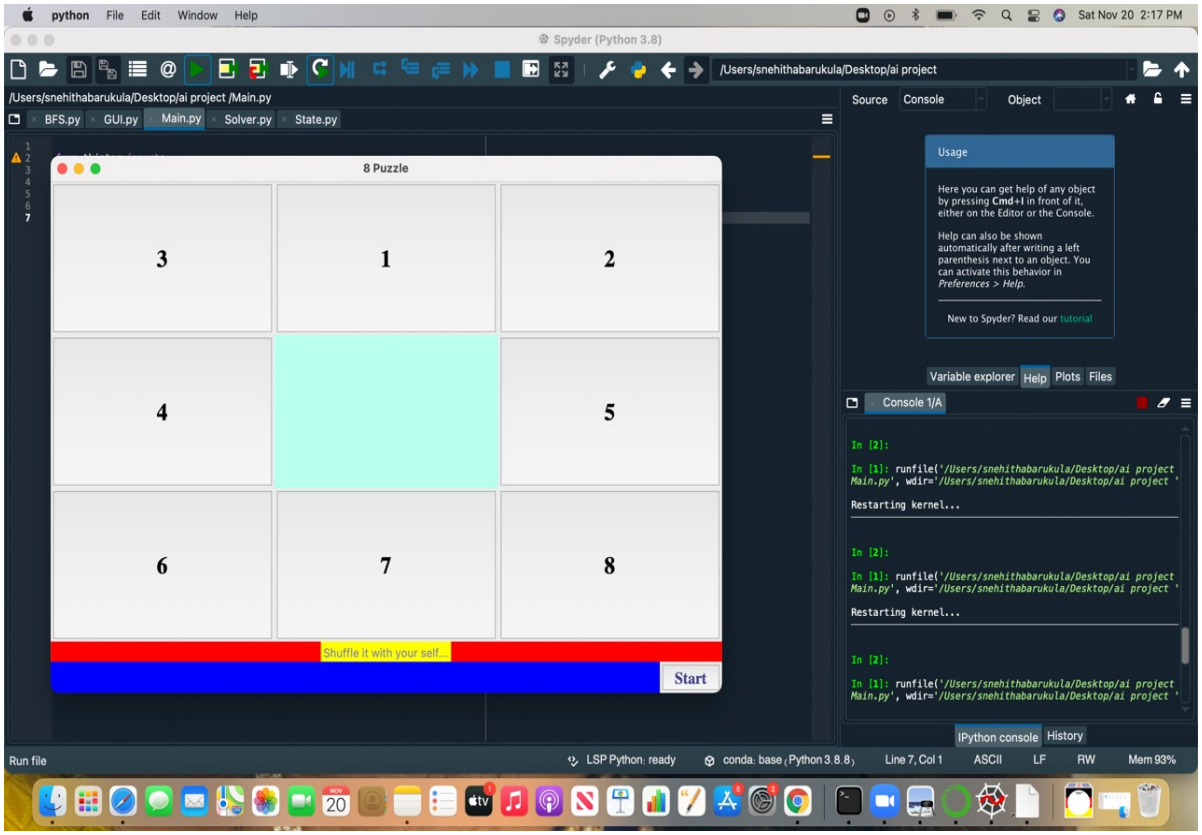
T. Anirudh



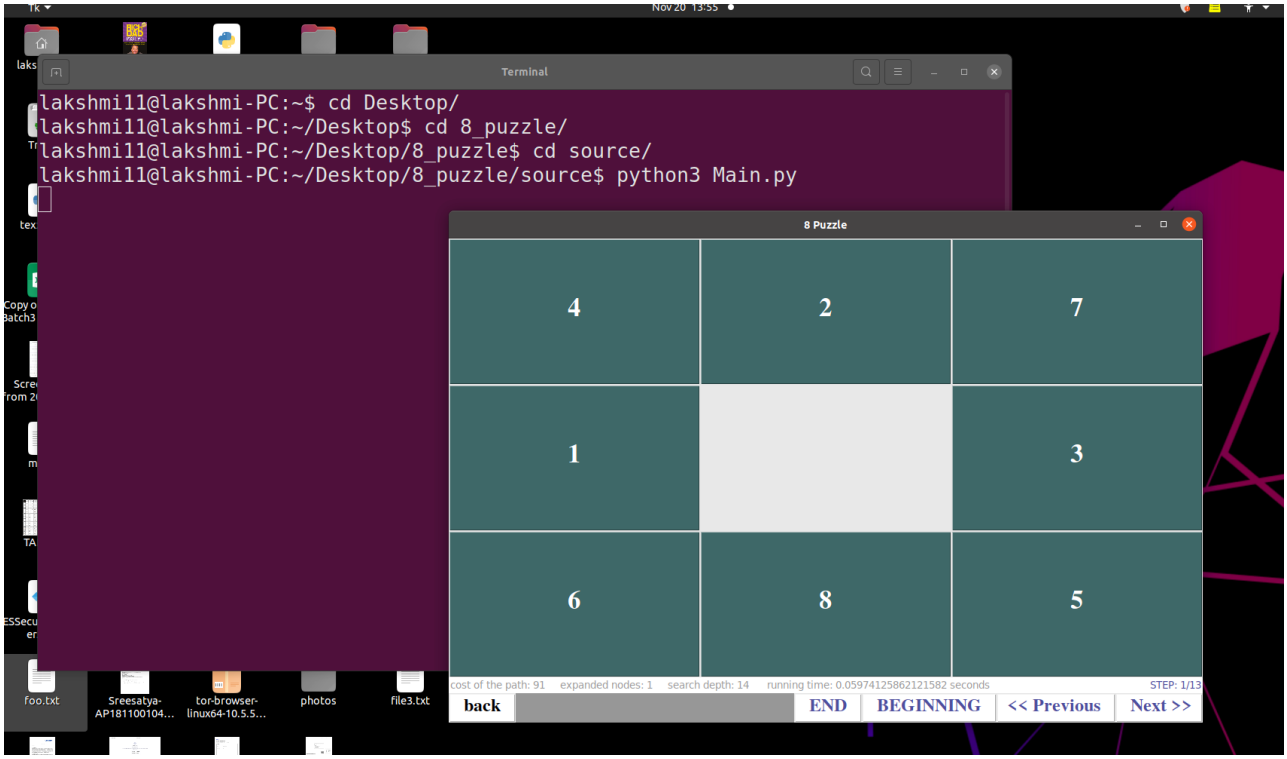
Sumedha Moturi



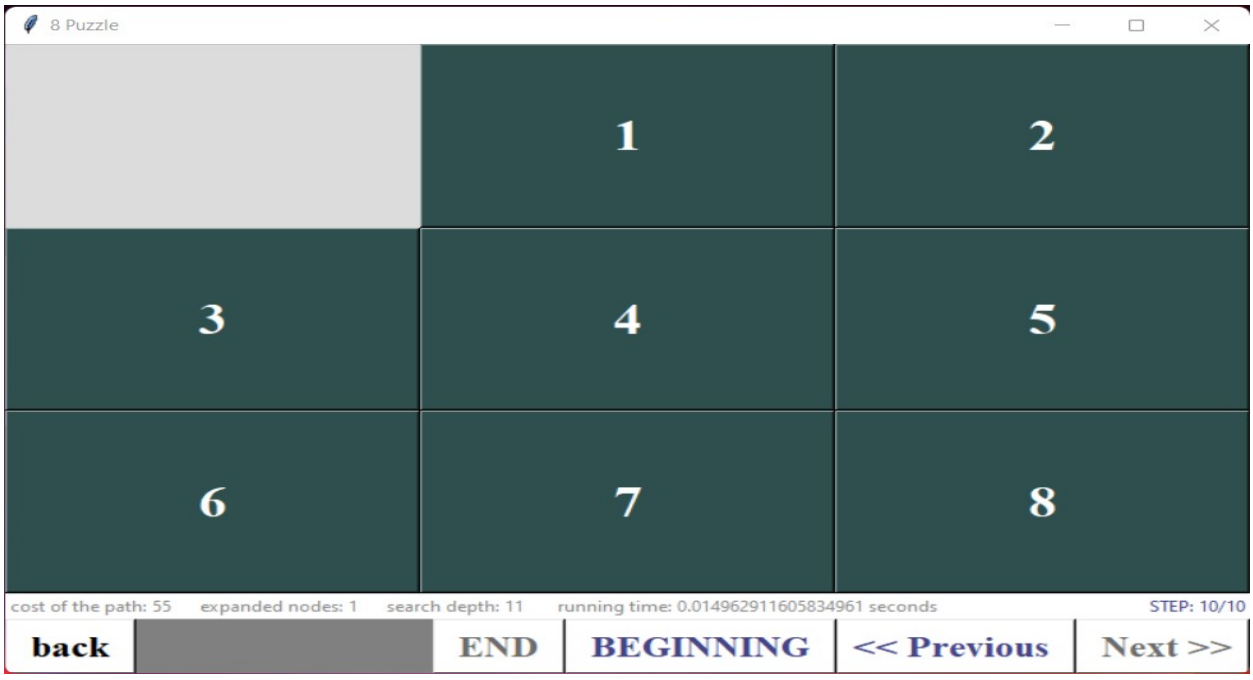
Barukula Snehitha



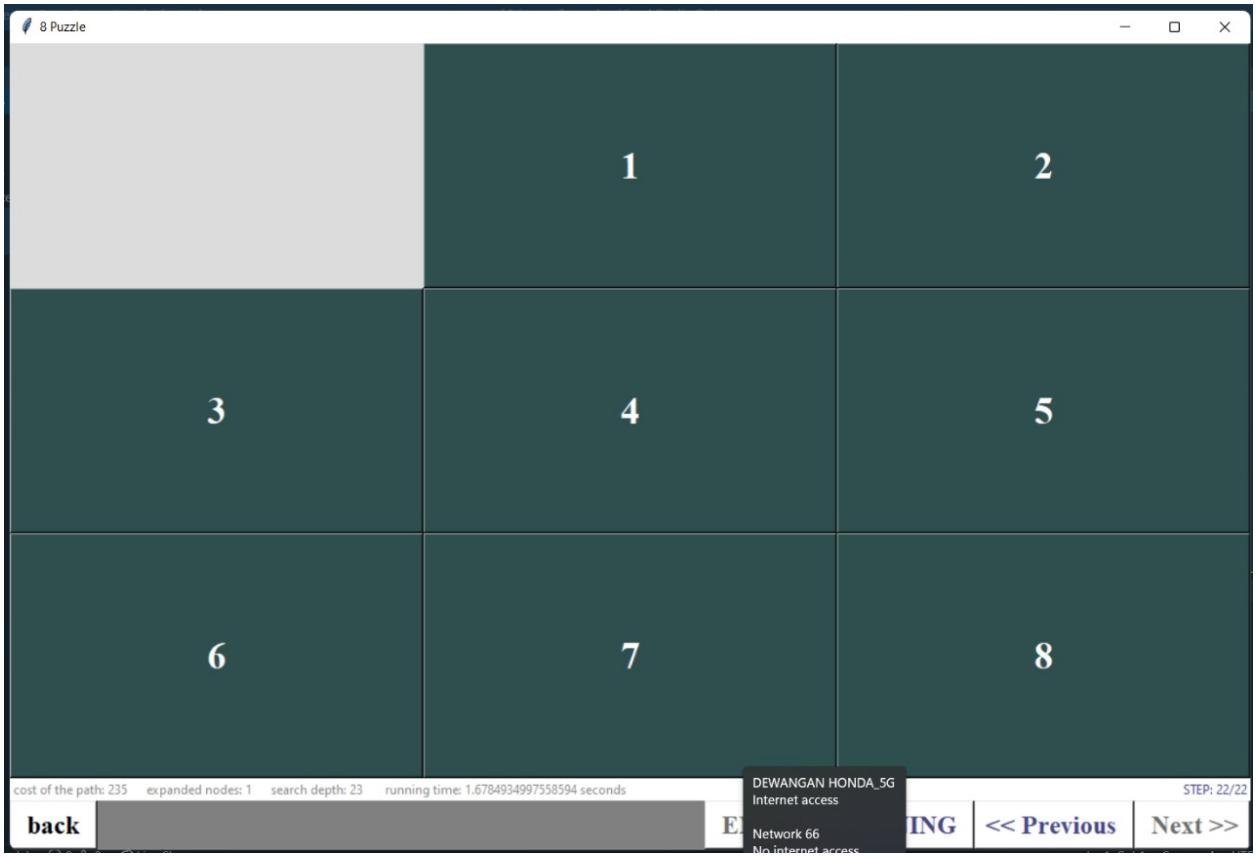
Lakshmi Vallala



Veeranki Jagadev

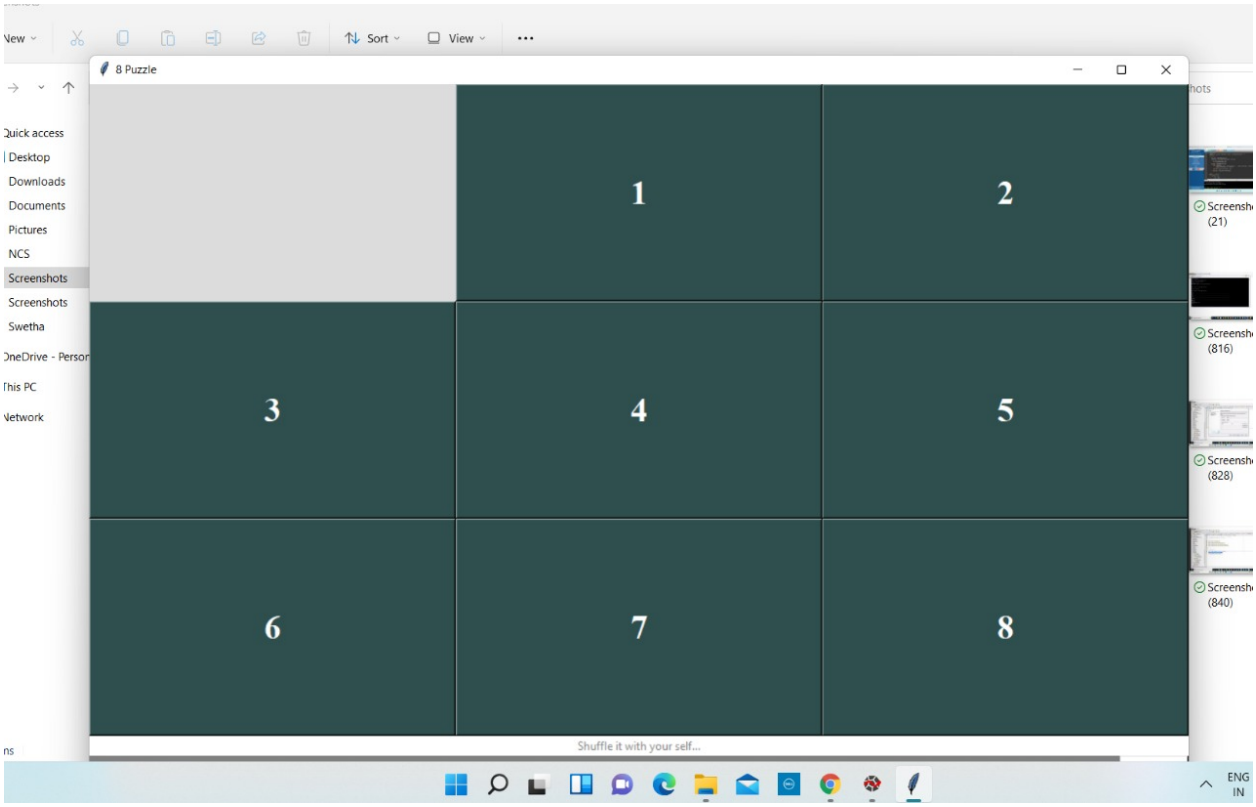


Yashveer Srivastava



Devendra Kumar Dewangan

Charan Sandeep



7. CONCLUSION

We have successfully created the 8-puzzle game with a GUI using the breadth-first search technique. Our future implementations are to beautify the UI and allow real-time tree visualisation.

7. CONTRIBUTION

Everybody has executed the code, worked on it and contributed to each section in the report but our major contribution is as follows:

AP18110010338	**Animesh Rituraj - Code**
AP18110010413	**Sumanth Kongani - Report**
AP18110010436	**Rahman Mohammad - Report**
AP18110010433	**M Vamsikrishna - Report**
AP18110010403	**Krishna Chaitanya - Report**
AP18110010441	**Harsha sai Y - Report**
AP18110010407	**T. Anirudh - Report**
AP18110010406	**Sumedha Moturi - Code & Report**

AP18110010438	**Barukula Snehitha - Report**
AP18110010416	**Lakshmi Vallala - Code & Report**
AP18110010642	**Charan Sandeep - Report**
AP18110010370	**Veeranki Jagadev - Code & Video**
AP18110010334	**Yashveer Srivastava - Code & Video**
AP18110010382	**Devendra Kumar Dewangan - Code & Video**