**Name: Suvra Barua**

**ID: 22010125.**

**Explanation of the DFS:**

DFS: DFS (Depth-First Search) is a graph traversal algorithm that explores as deep as possible along each branch before backtracking. It is a LIFO (Last In First Out). It uses a stack (either explicitly or via recursion) to remember nodes to explore. DFS is commonly used in tasks like path finding, detecting cycles, and solving puzzles. Its time complexity is O(V + E), where V is the number of vertices and E is the number of edges.

**Here is how this code works:**

This code builds a program to solve a maze using graph traversal. It reads a maze from a text file and finds a path from the start to the goal. The code is organized into different classes:

Node: Represents a state in the maze, keeping track of the current position, the parent node, and the action that led there (e.g., up, down).

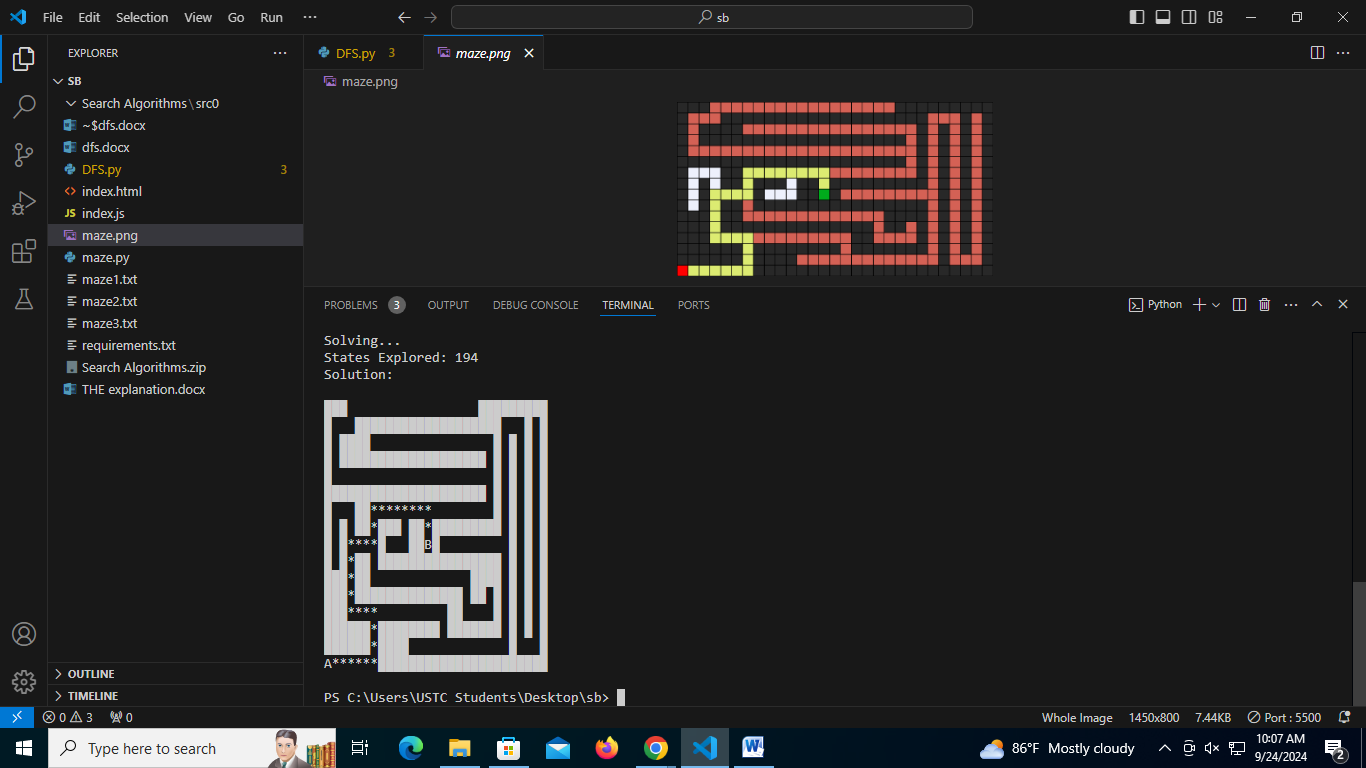
StackFrontier: Implements depth-first search (DFS) using a stack. You can add nodes, check if a state is in the frontier, and remove nodes (the last one added comes off first).

QueueFrontier: Inherits from StackFrontier but modifies remove() to follow a queue behavior, useful for breadth-first search (BFS).

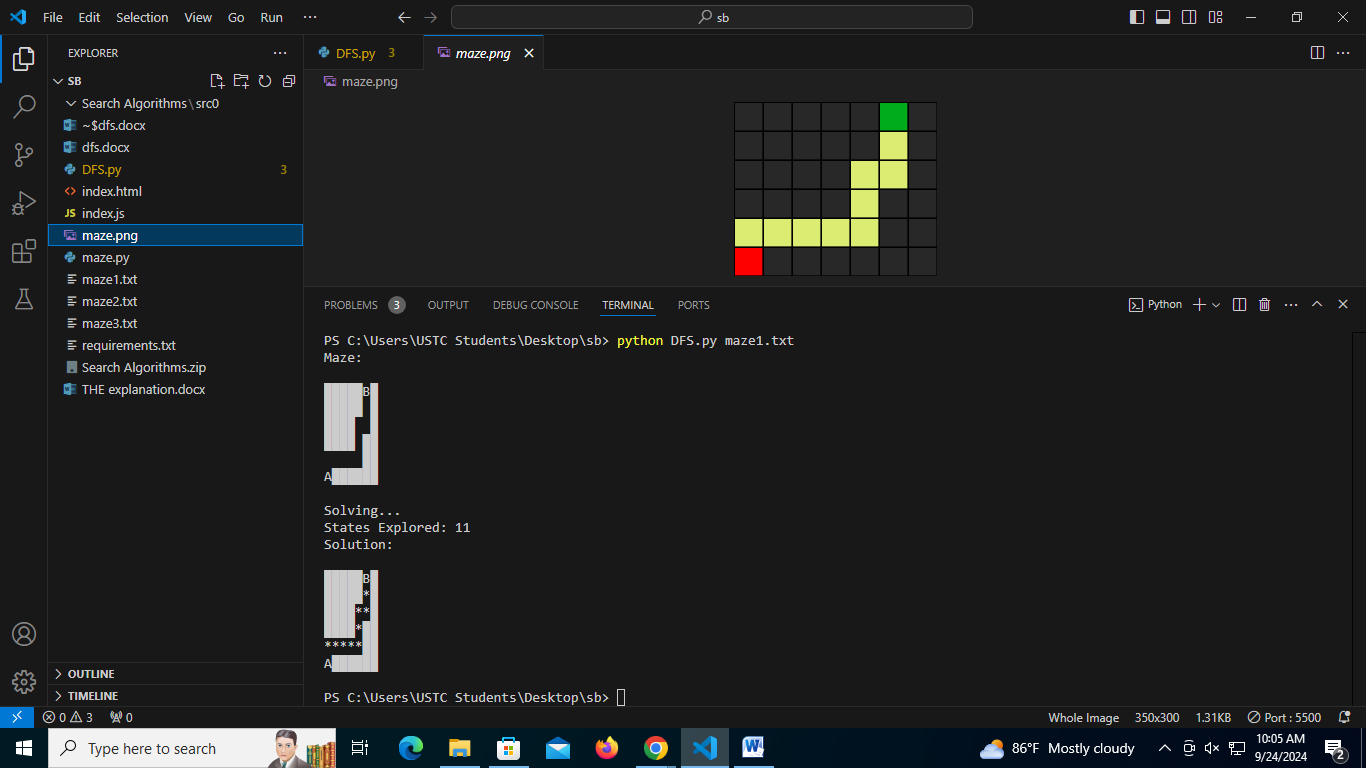
Maze: Reads the maze from a file, builds the maze's layout (walls, start, and goal), and provides methods to print, solve, and export an image of the maze. The solve() method uses DFS to explore paths until it finds the goal, tracing the path back to the start.

After that we generated an image to showcase the maze. We also outputted the maze and solved maze in the command line.

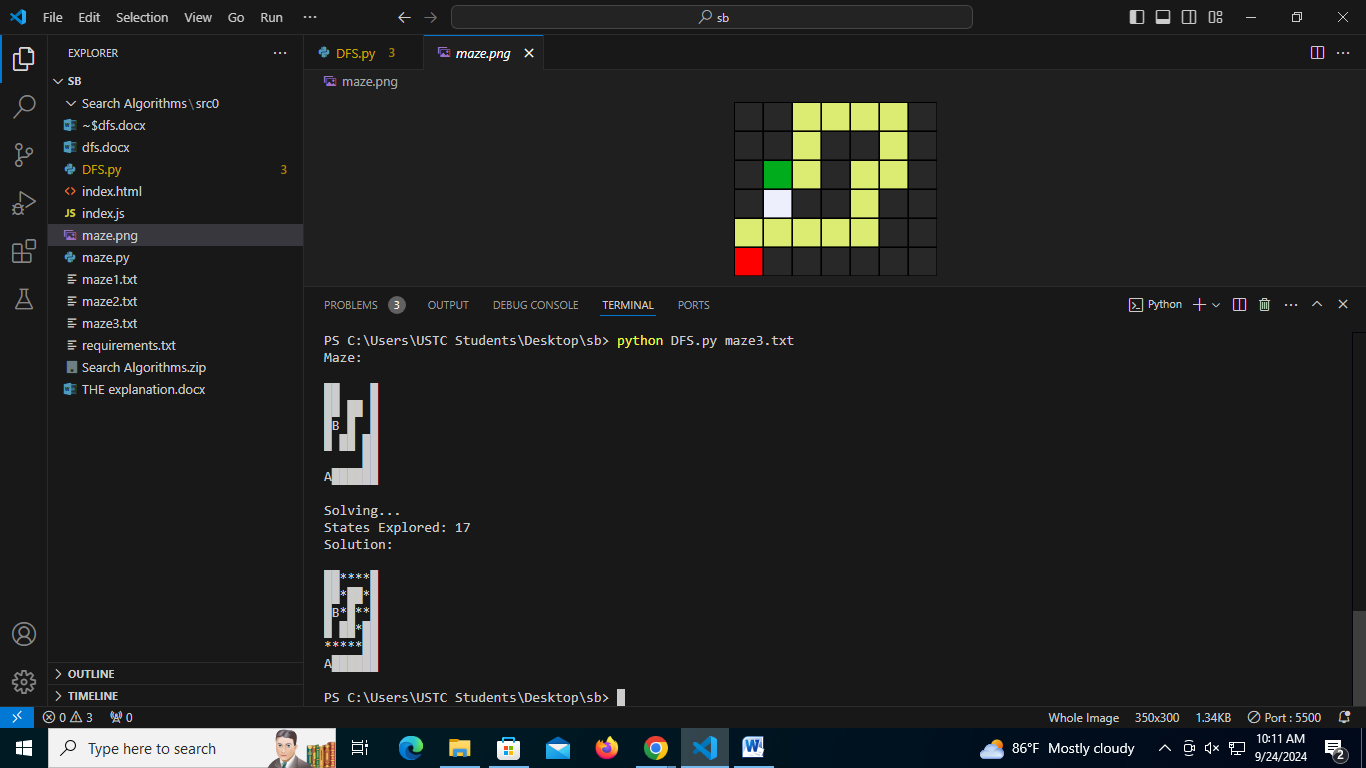
To run it, we have to pass in a text file containing the maze as an argument.



Output of maze1.txt



Output of maze2.txt



Output of maze3.txt