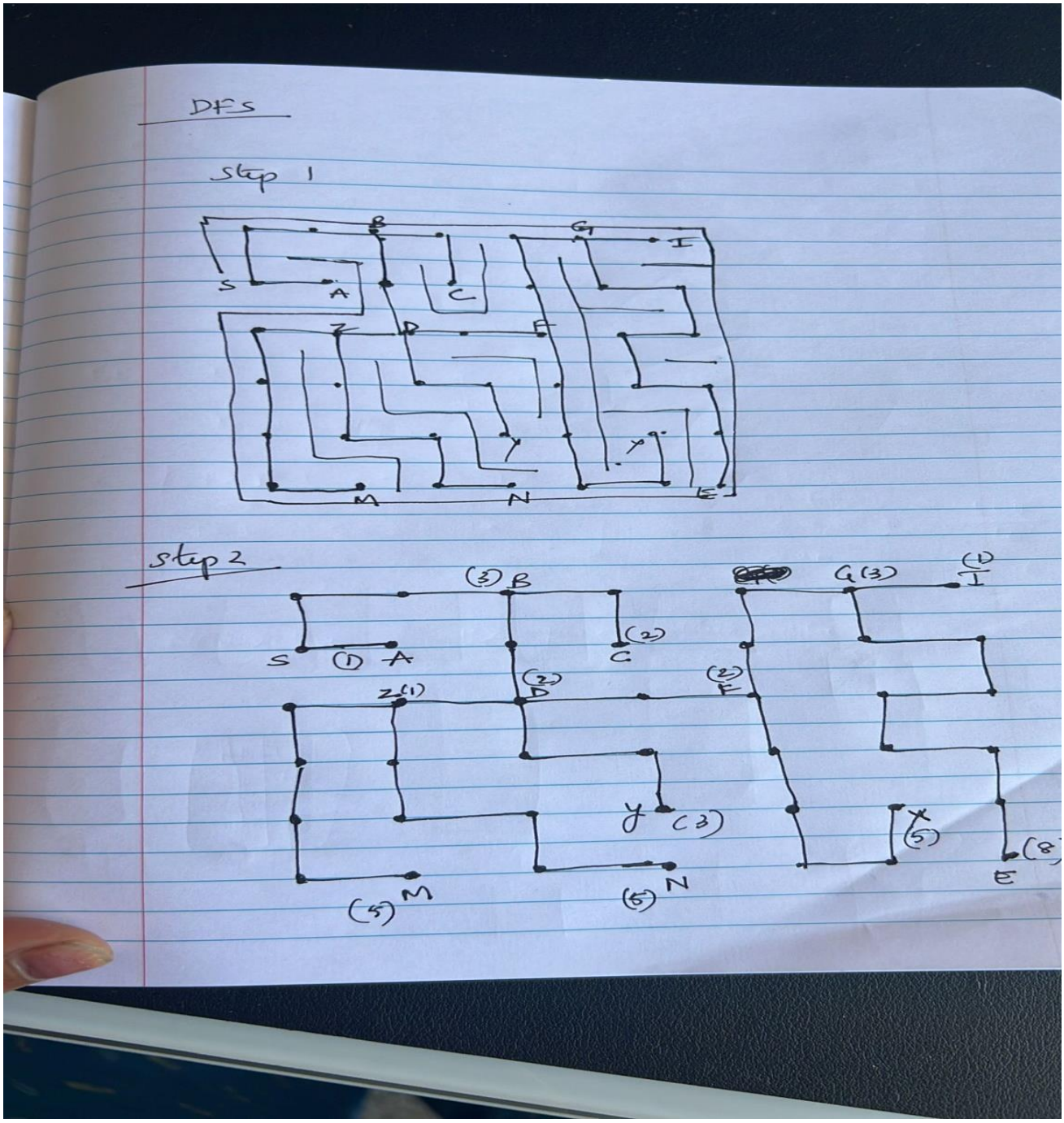
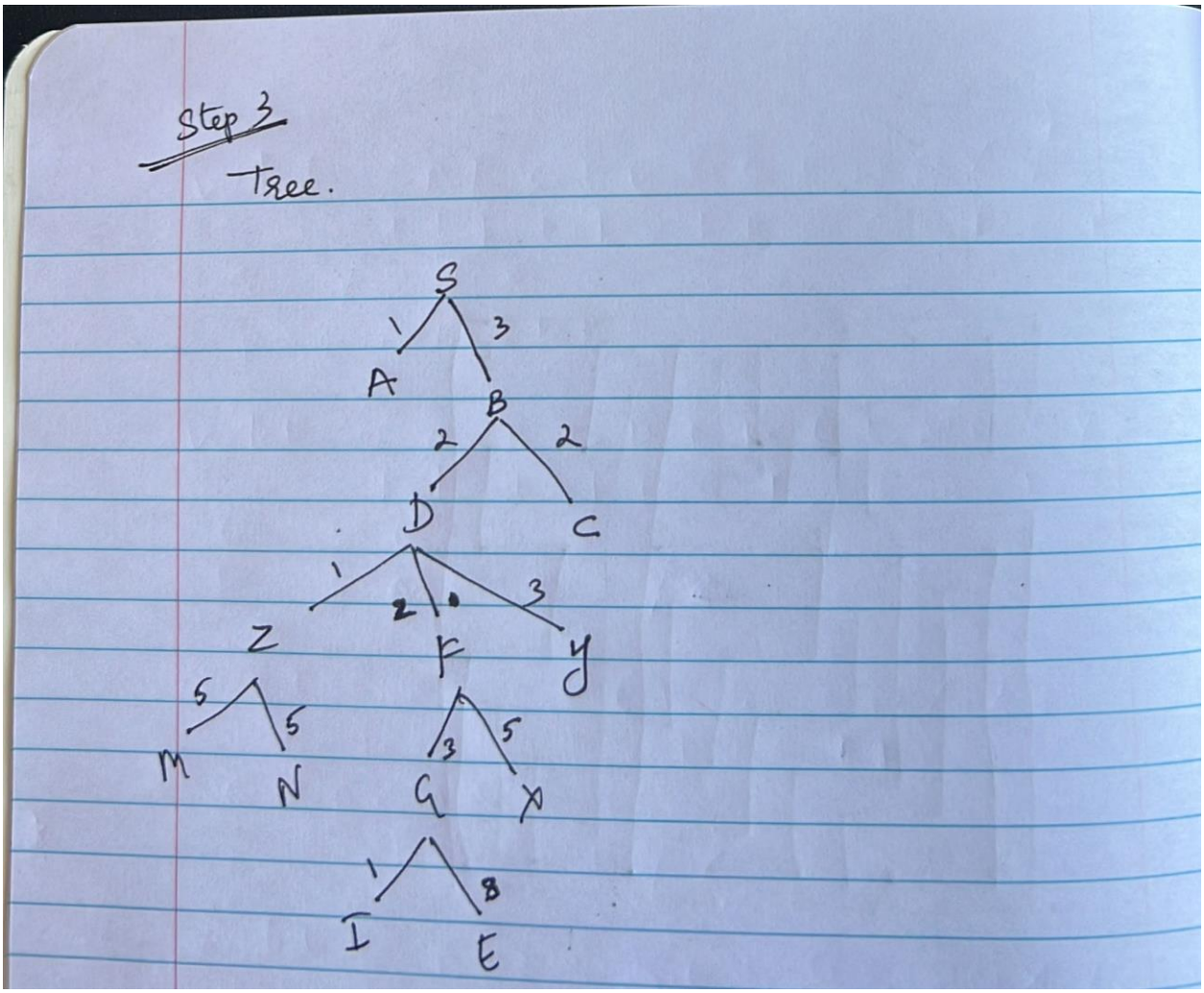
Week 11: Homework 2: Breadth-First Traversal: The Maze

Q4 => Conduct Breadth First Traversal (BFT) on the following maze



Week 11: Homework 2: Breadth-First Traversal: The Maze



Week 11: Homework 2: Breadth-First Traversal: The Maze

Step 4 BFS.

	S	A	B	C	D	E	F	G	I	M	N	X	Y	Z
	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Visited	S	A	B	C	D	E	F	G	I	M	N	X	Y	Z

Q : S

P :

	S	A	B	C	D	E	F	G	I	M	N	X	Y	Z
Visited	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Q : AB

P : S

	S	A	B	C	D	E	F	G	I	M	N	X	Y	Z
Visited	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Q : B.

P : SA

	S	A	B	C	D	E	F	G	I	M	N	X	Y	Z
Visited	1	1	1	1	1	0	0	0	0	0	0	0	0	0

Q : CD

P : SAB

	S	A	B	C	D	E	F	G	I	M	N	X	Y	Z
Visited	1	1	1	1	1	0	0	0	0	0	0	0	0	0

Q : D

P : SABC

	S	A	B	C	D	E	F	G	I	M	N	X	Y	Z
Visited	1	1	1	1	1	0	1	0	0	0	0	0	1	1

Q : FYZ

P : SA BCD

Week 11: Homework 2: Breadth-First Traversal: The Maze

visited : S A B C D E F G I M N X Y Z
1 1 1 1 1 1 0 1 1 0 0 0 1 1 1
Q Y Z G X
P S A B C D F

visited S A B C D E F G I M N X Y Z
1 1 1 1 1 1 0 1 1 0 0 0 1 1 1
Q: Z G X
P: S A B C D F Y

visited S A B C D E F G I M N X Y Z
1 1 1 1 1 1 0 1 1 0 1 1 1 1 1
Q: G X M N
P: S A B C D F Y Z

visited S A B C D E F G I M N X Y Z
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Q: X M N I E
P: S A B C D F Y Z G

visited S A B C D E F G I M N X Y Z
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Q: M N I E
P: S A B C D F Y Z G X

visited S A B C D E F G I M N X Y Z
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Q: N I E
P: S A B C D F Y Z G X M

Week 11: Homework 2: Breadth-First Traversal: The Maze

Visited S A B C D E F G I M N X Y Z

Q I E

P S A B C D F Y Z G X M N

Visited S A B C D E F G I M N X Y Z

Q E

P S A B C D F Y Z G X M N I

Visited S A B C D E F G I M N X Y Z.

Q ——

P : S A B C D F Y Z G X M I N E

Week 11: Homework 2: Breadth-First Traversal: The Maze

Step1 : Traced the path in Maze

Step 2 : The actual path I got

Step 3: Maze converted into Tree

Step 4: Breadth First Traversal

Code

```
from  typing import List
import collections

class Solution:
    def hasPath(self, maze: List[List[int]], start: List[int], destination:
List[int]) -> bool:
        visited = set()
        dirs = [0, 1, 0, -1, 0]
        row, col = len(maze), len(maze[0])

        def bfs(i, j):
            queue = collections.deque([[i, j]])
            visited.add((i, j))
            while queue:
                i, j = queue.pop()
                if [i, j] == destination: return True
                for k in range(4):
                    move_i, move_j = i, j
                    while 0 <= (t1 := move_i + dirs[k]) < row and 0 <= (t2 :=
move_j + dirs[k + 1]) < col and maze[t1][t2] != 1:
                        move_i, move_j = t1, t2
                    if move_i == i and move_j == j: continue
                    if (move_i, move_j) not in visited:
                        visited.add((move_i, move_j))
                        queue.appendleft([move_i, move_j])
            return False

        return bfs(start[0], start[1])
```

Week 11: Homework 2: Breadth-First Traversal: The Maze

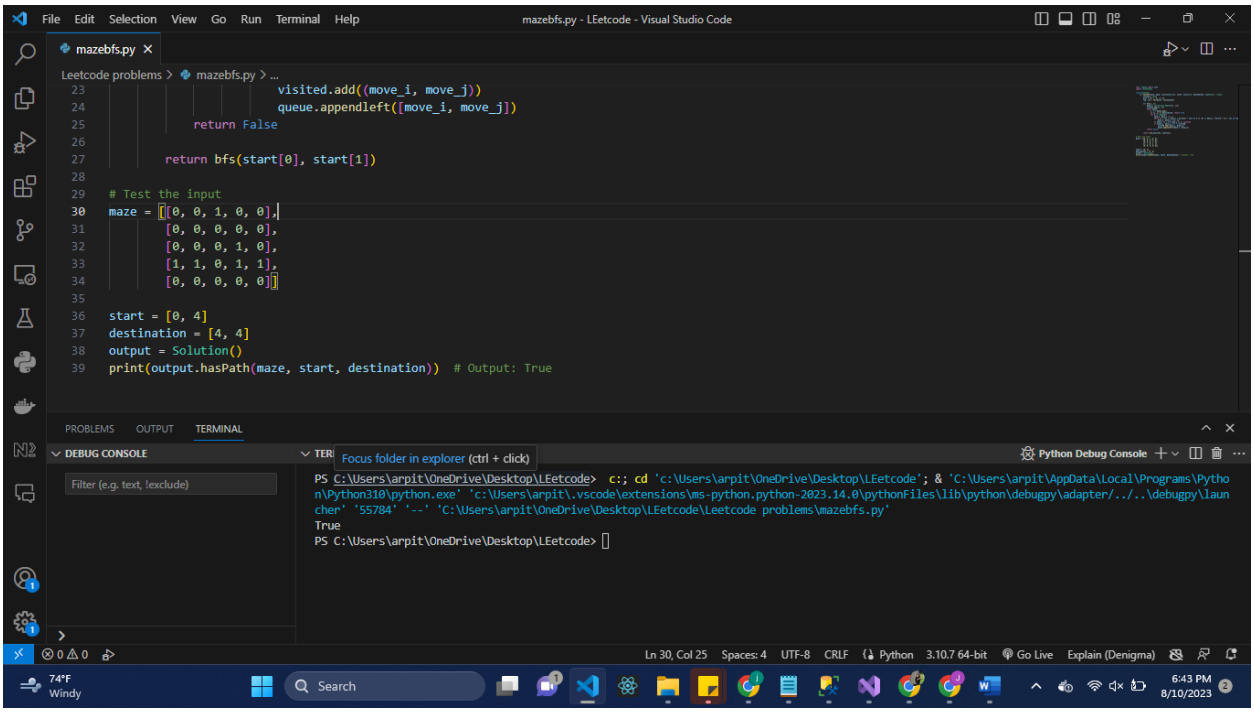
```
# Test the input
maze = [[0, 0, 1, 0, 0],
        [0, 0, 0, 0, 0],
        [0, 0, 0, 1, 0],
        [1, 1, 0, 1, 1],
        [0, 0, 0, 0, 0]]

start = [0, 4]
destination = [4, 4]
output = Solution()
print(output.hasPath(maze, start, destination))  # Output: True
```

Test case 1

Input:	maze = [[0,0,1,0,0],[0,0,0,0,0],[0,0,0,1,0],[1,1,0,1,1],[0,0,0,0,0]], start = [0,4], destination = [4,4]
Output:	true
Explanation:	One possible way is : left -> down -> left -> down -> right -> down -> right.

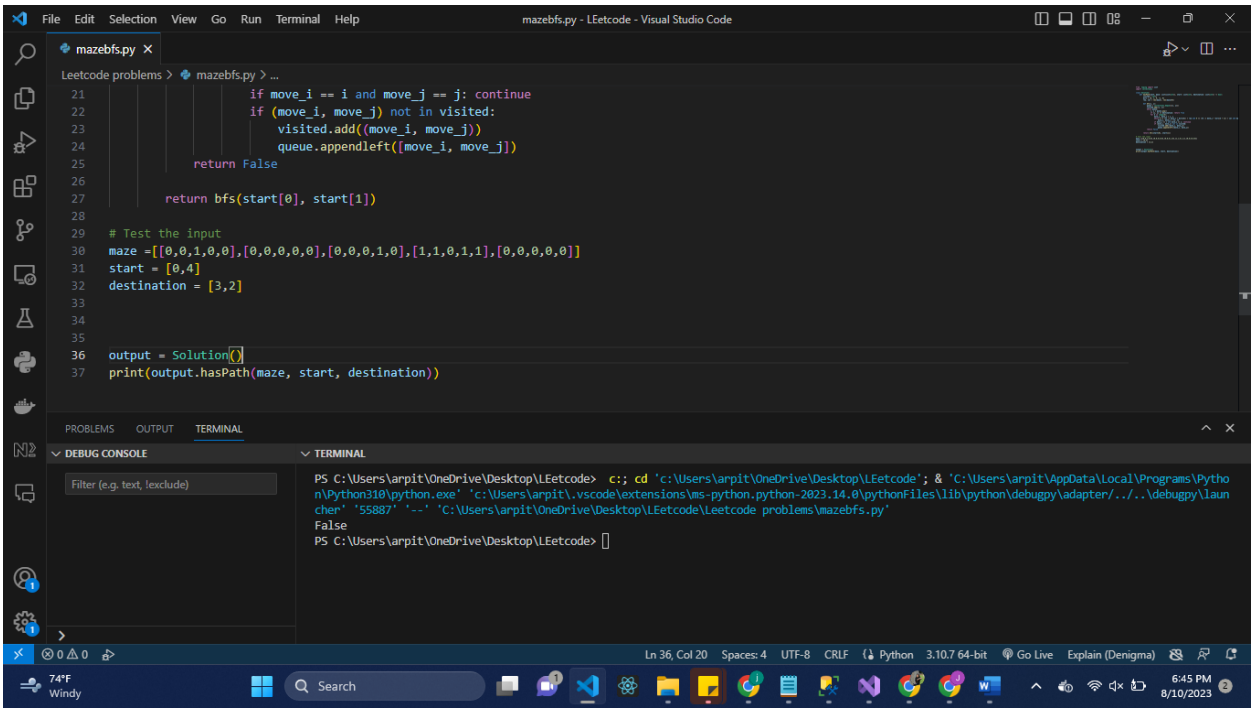
Week 11: Homework 2: Breadth-First Traversal: The Maze



Test case 2

Input: maze = [[0,0,1,0,0],[0,0,0,0,0],[0,0,0,1,0],[1,1,0,1,1],[0,0,0,0,0]], start = [0,4], destination = [3,2]
Output: false
Explanation: There is no way for the ball to stop at the destination. Notice that you can pass through the destination but you cannot stop there.

Week 11: Homework 2: Breadth-First Traversal: The Maze



Test case 3

Input: maze = [[0,0,0,0,0],[1,1,0,0,1],[0,0,0,0,0],[0,1,0,0,1],[0,1,0,0,0]], start = [4,3], destination = [0,1] **Output:** false

Week 11: Homework 2: Breadth-First Traversal: The Maze