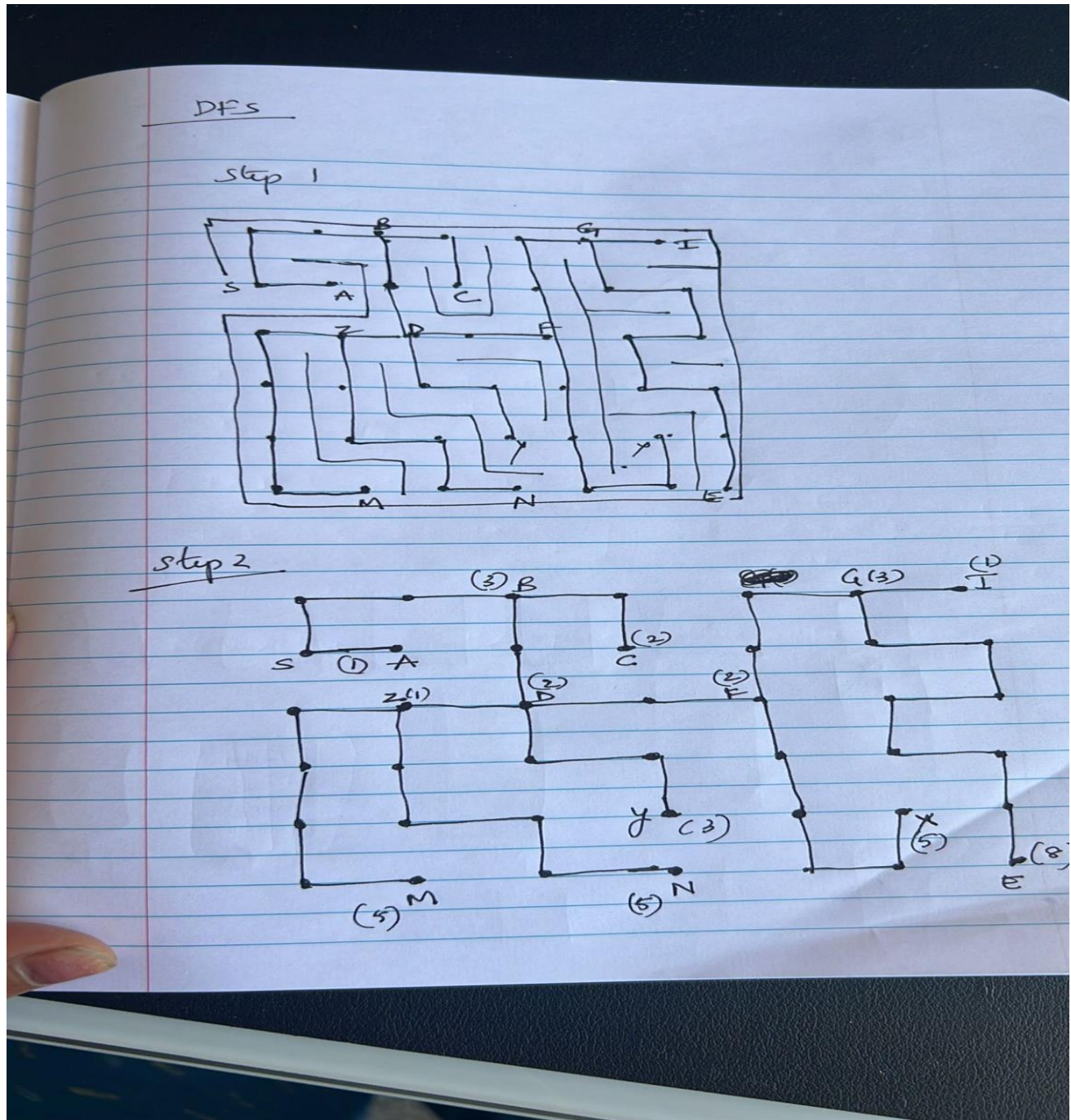
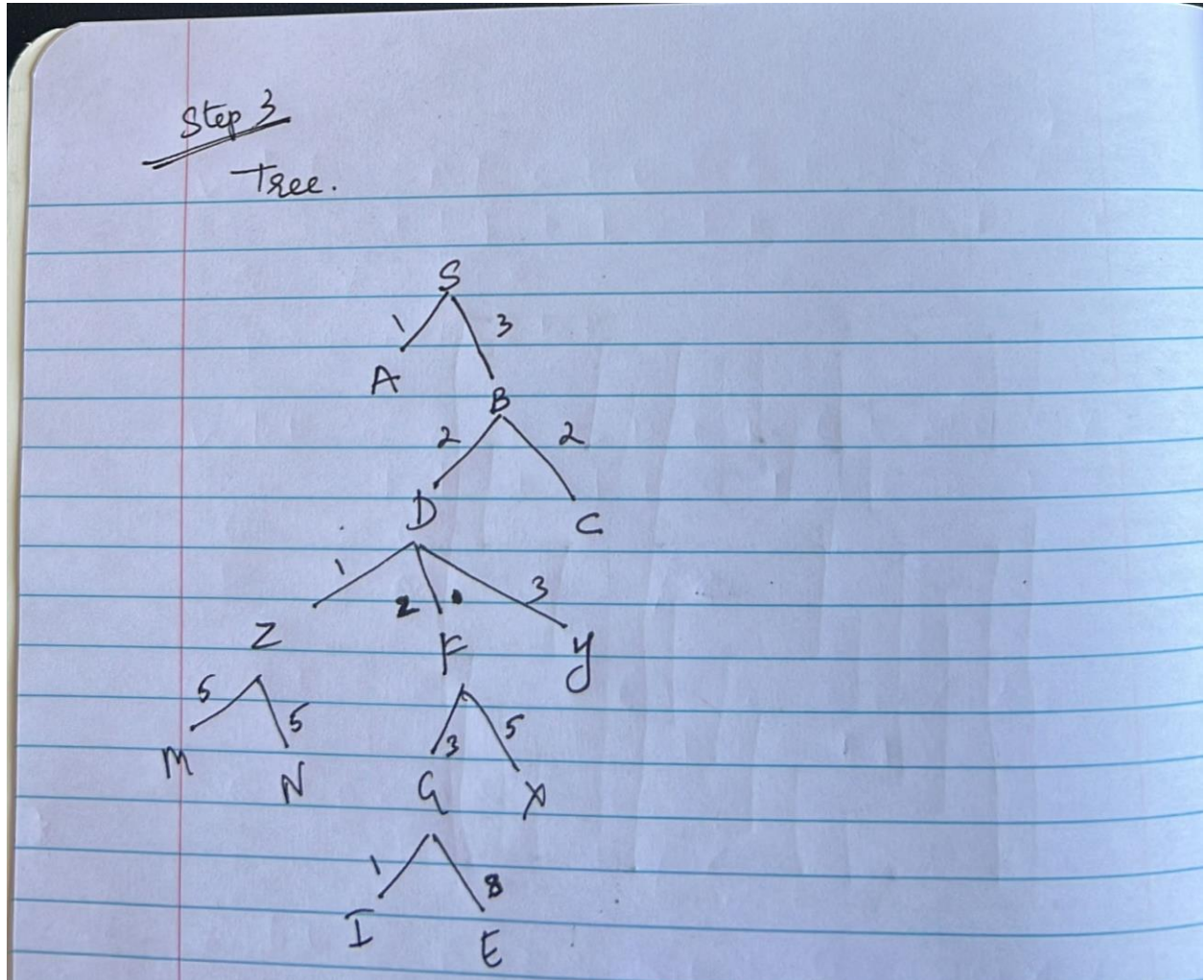


## Week 11: Homework 2: Breadth-First Traversal: The Maze

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



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





Step 4 BRS.

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

Q: S

P:

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

1 1 1 0 0 0 0 0 0 0 0 0 0 0

Q: AB

P: S

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

1 1 1 0 0 0 0 0 0 0 0 0 0 0

Q: B

P: SA

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

1 1 1 1 0 0 0 0 0 0 0 0 0 0

Q: CD

P: SAB

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

1 1 1 1 1 0 0 0 0 0 0 0 0 0

Q: D

P: SABC

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

1 1 1 1 1 0 1 0 0 0 0 0 1 1

Q: FYZ

P: SABC



## 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 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 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 0 1 1 0 0 0 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  
 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  
 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  
 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
```

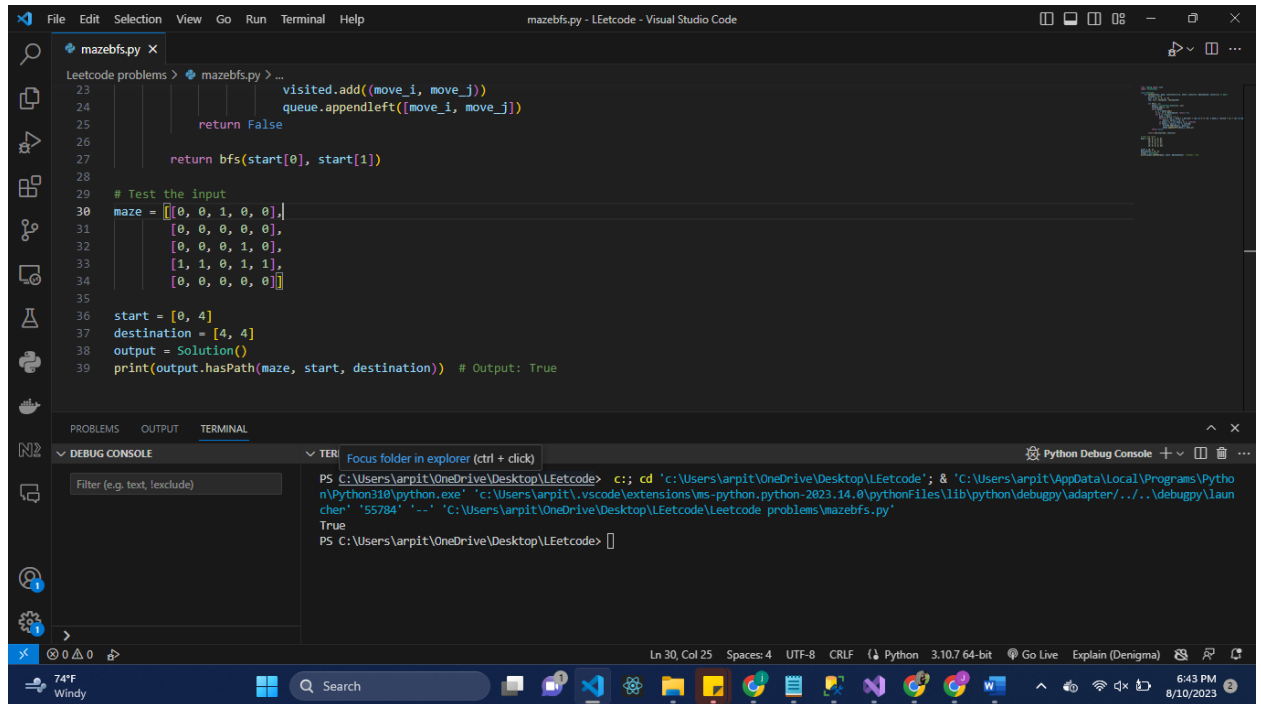
Test case 1

**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



```
File Edit Selection View Go Run Terminal Help
mazebfs.py - LeetCode - Visual Studio Code

Leetcode problems > mazebfs.py > ...
23         visited.add((move_i, move_j))
24         queue.appendleft([move_i, move_j])
25     return False
26
27     return bfs(start[0], start[1])
28
29 # Test the input
30 maze = [[0, 0, 1, 0, 0],
31         [0, 0, 0, 0, 0],
32         [0, 0, 0, 1, 0],
33         [1, 1, 0, 1, 1],
34         [0, 0, 0, 0, 0]]
35
36 start = [0, 4]
37 destination = [4, 4]
38 output = Solution()
39 print(output.hasPath(maze, start, destination)) # Output: True

PROBLEMS OUTPUT TERMINAL
DEBUG CONSOLE
Filter (e.g. text, exclude)
TERMINAL
Focus folder in explorer (ctrl + click)
Python Debug Console
PS C:\Users\arpit\OneDrive\Desktop\LeetCode> c:: cd 'c:\Users\arpit\OneDrive\Desktop\LeetCode'; & 'C:\Users\arpit\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\arpit\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55784' '-.' 'C:\Users\arpit\OneDrive\Desktop\LeetCode\LeetCode problems\mazebfs.py'
True
PS C:\Users\arpit\OneDrive\Desktop\LeetCode> []

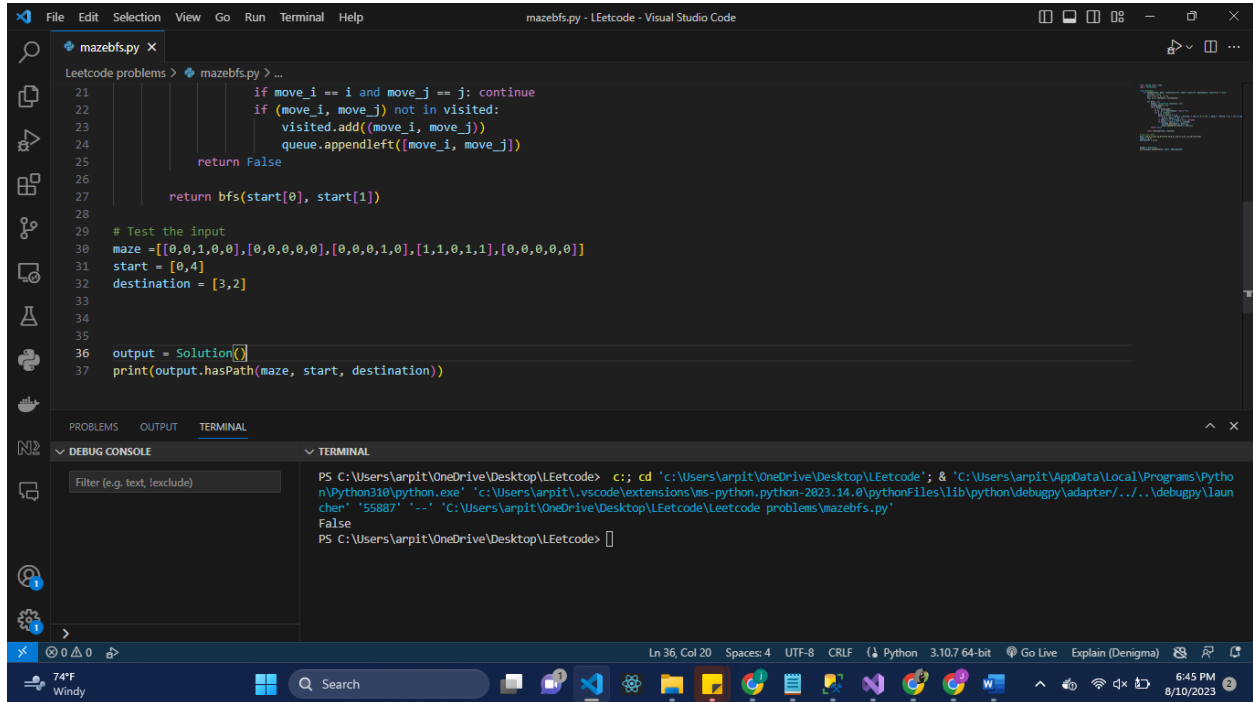
Ln 30, Col 25 Spaces: 4 UTF-8 CRLF Python 3.10.7 64-bit Go Live Explain (Denigma)
74°F Windy 6:43 PM 8/10/2023
```

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



```
File Edit Selection View Go Run Terminal Help
mazebfs.py - LeetCode - Visual Studio Code

Leetcode problems > mazebfs.py > ...
21         if move_i == i and move_j == j: continue
22         if (move_i, move_j) not in visited:
23             visited.add((move_i, move_j))
24             queue.appendleft((move_i, move_j))
25         return False
26
27     return bfs(start[0], start[1])
28
29 # Test the input
30 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]]
31 start = [0,4]
32 destination = [3,2]
33
34
35
36 output = Solution()
37 print(output.hasPath(maze, start, destination))

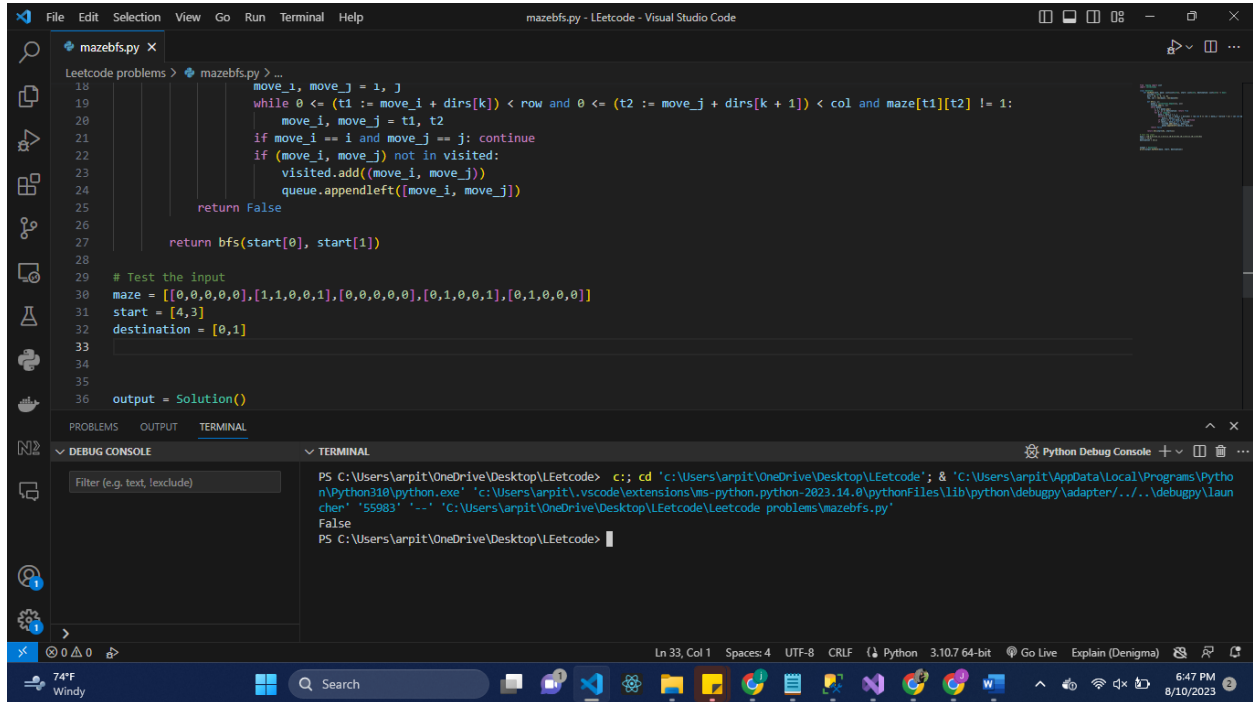
PROBLEMS OUTPUT TERMINAL
DEBUG CONSOLE TERMINAL
Filter (e.g. text, exclude)
PS C:\Users\arpit\OneDrive\Desktop\LeetCode> c:: cd 'c:\Users\arpit\OneDrive\Desktop\LeetCode'; & 'C:\Users\arpit\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\arpit\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55887' '--' 'C:\Users\arpit\OneDrive\Desktop\LeetCode\LeetCode problems\mazebfs.py'
False
PS C:\Users\arpit\OneDrive\Desktop\LeetCode>

Ln 36, Col 20 Spaces: 4 UTF-8 CRLF Python 3.10.7 64-bit Go Live Explain (Denigma) 74°F Windy 6:45 PM 8/10/2023
```

Test case 3

<b>Input:</b>	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]
<b>Output:</b>	false

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



The screenshot shows the Visual Studio Code editor with a file named `mazebfs.py` open. The code implements a Breadth-First Traversal (BFS) algorithm to solve a maze problem. The maze is represented as a 2D grid where 0 is an open path and 1 is a wall. The goal is to find the shortest path from a start point to a destination point.

```
18 move_i, move_j = i, j
19 while 0 <= (t1 := move_i + dirs[k]) < row and 0 <= (t2 := move_j + dirs[k + 1]) < col and maze[t1][t2] != 1:
20     move_i, move_j = t1, t2
21     if move_i == i and move_j == j: continue
22     if (move_i, move_j) not in visited:
23         visited.add((move_i, move_j))
24         queue.appendleft((move_i, move_j))
25     return False
26
27 return bfs(start[0], start[1])
28
29 # Test the input
30 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]]
31 start = [4,3]
32 destination = [0,1]
33
34
35
36 output = Solution()
```

The terminal window shows the command prompt output, indicating that the program executed successfully and returned `False`.

```
PS C:\Users\arpit\OneDrive\Desktop\LEetcode> cd 'c:\Users\arpit\OneDrive\Desktop\LEetcode'; & 'C:\Users\arpit\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\arpit\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55983' '--' 'C:\Users\arpit\OneDrive\Desktop\LEetcode\LEetcode problems\mazebfs.py'
False
PS C:\Users\arpit\OneDrive\Desktop\LEetcode>
```