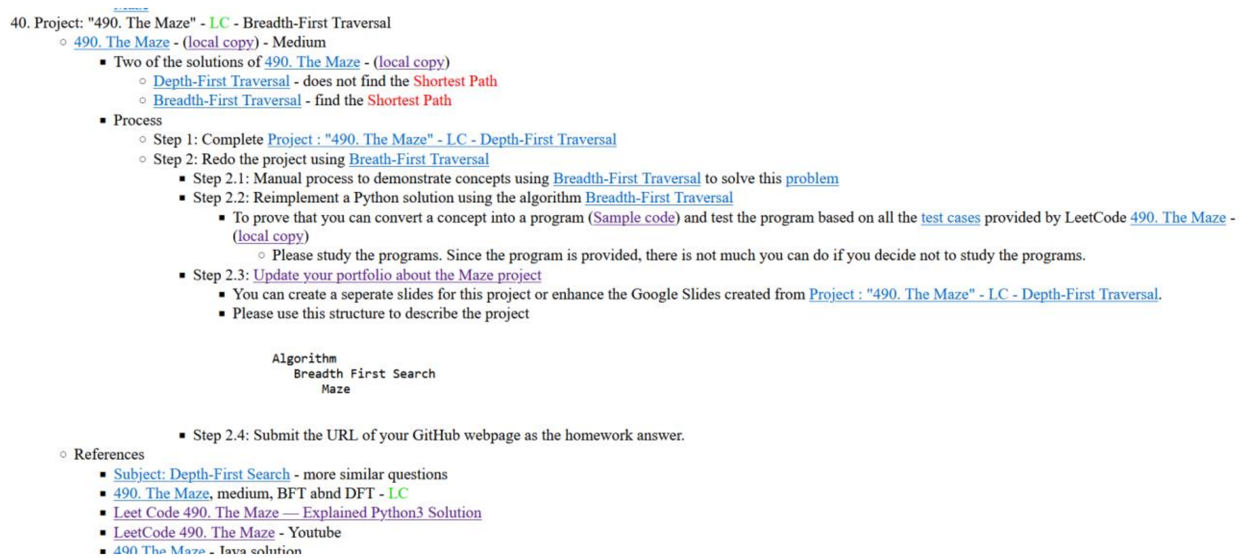


40. Project: "490. The Maze" - LC - Breadth-First Traversal

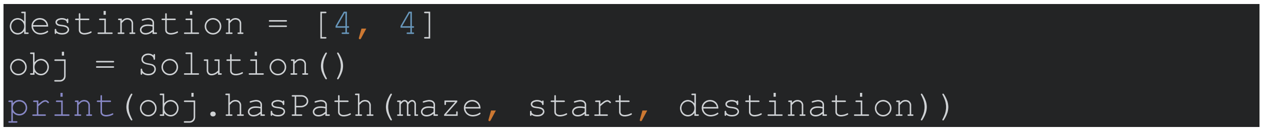
- 490. The Maze - (local copy) - Medium
  - Two of the solutions of 490. The Maze - (local copy)
    - Depth-First Traversal - does not find the Shortest Path
    - Breadth-First Traversal - find the Shortest Path
  - Process
    - Step 1: Complete Project : "490. The Maze" - LC - Depth-First Traversal
    - Step 2: Redo the project using Breath-First Traversal
      - Step 2.1: Manual process to demonstrate concepts using Breadth-First Traversal to solve this problem
      - Step 2.2: Reimplement a Python solution using the algorithm Breadth-First Traversal
        - To prove that you can convert a concept into a program (Sample code) and test the program based on all the test cases provided by LeetCode 490. The Maze - (local copy)
          - Please study the programs. Since the program is provided, there is not much you can do if you decide not to study the programs.
      - Step 2.3: Update your portfolio about the Maze project
        - You can create a seperate slides for this project or enhance the Google Slides created from Project : "490. The Maze" - LC - Depth-First Traversal.
        - Please use this structure to describe the project

```
Algorithm
   Breadth First Search
       Maze
```

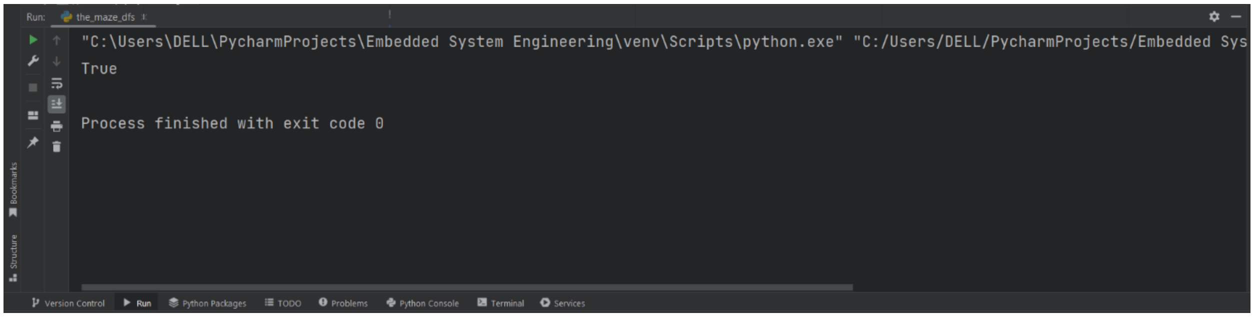
      - Step 2.4: Submit the URL of your GitHub webpage as the homework answer.
- References
  - Subject: Depth-First Search - more similar questions
  - 490. The Maze, medium, BFT abnd DFT - LC
  - Leet Code 490. The Maze — Explained Python3 Solution
  - LeetCode 490. The Maze - Youtube
  - 490 The Maze - Java solution

Step 1: 490. The Maze (DFS)

```
class Solution:
    def hasPath(self, maze, start, destination):
        m, n, stopped = len(maze), len(maze[0]), set()

        def dfs(x, y):
            if (x, y) in stopped:
                return False
            stopped.add((x, y))
            if [x, y] == destination:
                return True
            for i, j in (-1, 0), (1, 0), (0, -1), (0, 1):
                newX, newY = x, y
                while 0 <= newX + i < m and 0 <= newY + j < n and maze[newX + i][newY + j] != 1:
                    newX += i
                    newY += j
                if dfs(newX, newY):
                    return True
            return False

        return dfs(*start)


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]
```



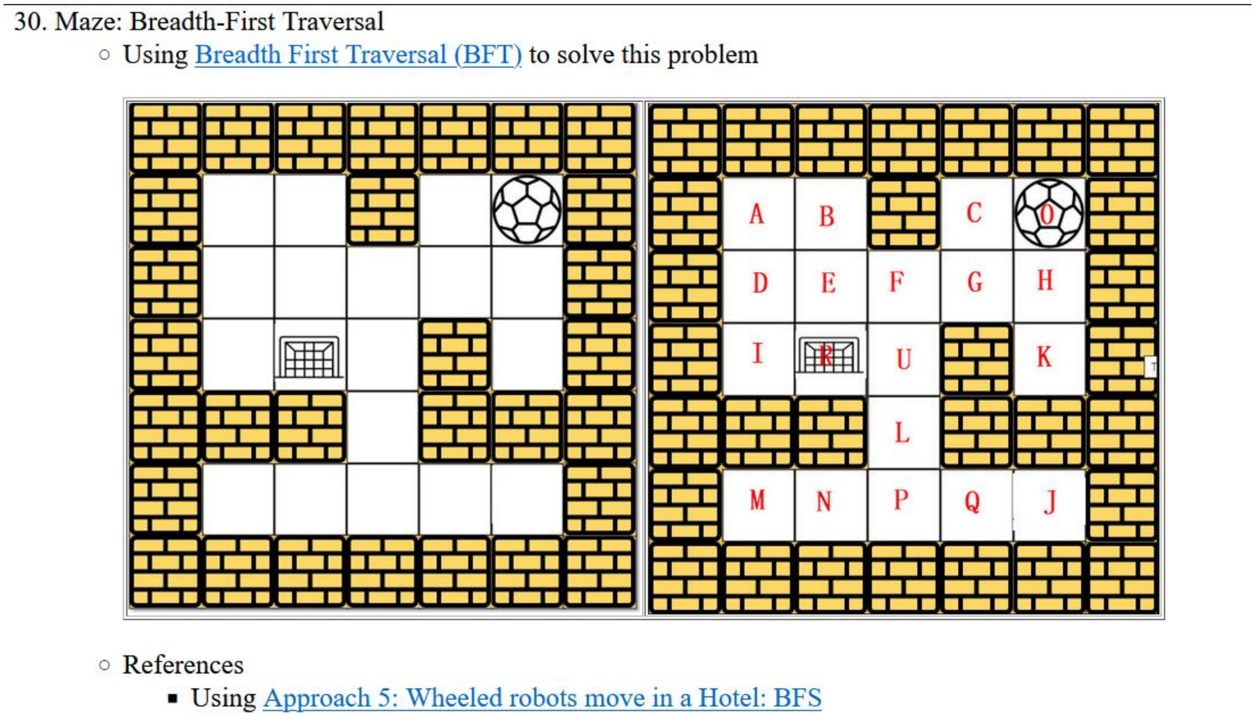
```
destination = [4, 4]
obj = Solution()
print(obj.hasPath(maze, start, destination))
```



```
"C:\Users\DELL\PycharmProjects\Embedded System Engineering\venv\Scripts\python.exe" "C:/Users/DELL/PycharmProjects/Embedded Sys
True

Process finished with exit code 0
```

Step 2: Depth-First Traversal

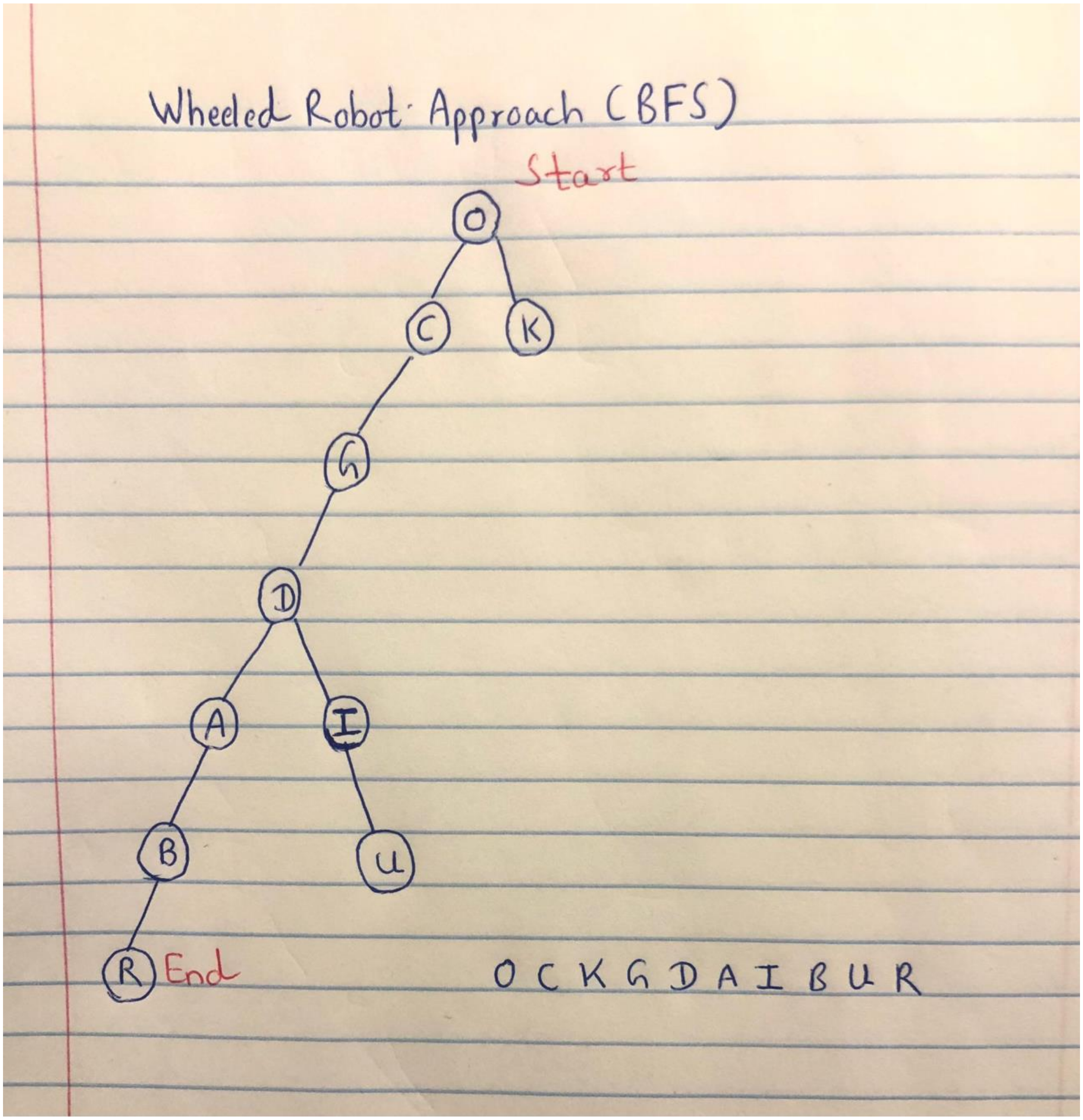
Step 2.1: Manual Process to Demonstrate Concepts Using BFS



30. Maze: Breadth-First Traversal

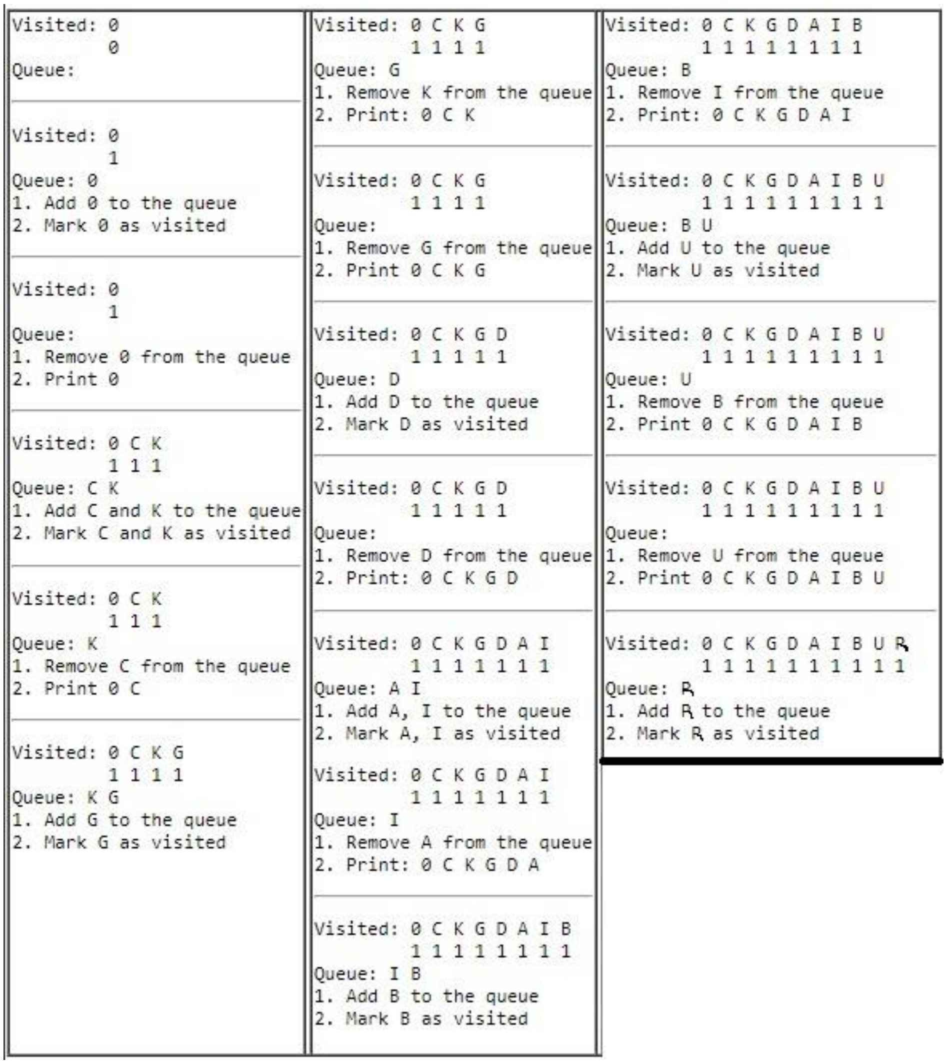
- Using Breadth First Traversal (BFT) to solve this problem

- References
  - Using Approach 5: Wheeled robots move in a Hotel: BFS



We are going to use the wheeled robot approach for this problem which means the ball can go through the empty spaces by rolling right, left, up, down, but it won't stop rolling until hitting a wall. When the ball stops, it could choose the next direction.

The following diagram contains the steps to get to the destination.



```
Visited: 0
         0
Queue:

Visited: 0
         1
Queue: 0
1. Add 0 to the queue
2. Mark 0 as visited

Visited: 0
         1
Queue:
1. Remove 0 from the queue
2. Print 0

Visited: 0 C K
         1 1 1
Queue: C K
1. Add C and K to the queue
2. Mark C and K as visited

Visited: 0 C K
         1 1 1
Queue: K
1. Remove C from the queue
2. Print 0 C

Visited: 0 C K G
         1 1 1 1
Queue: K G
1. Add G to the queue
2. Mark G as visited
```

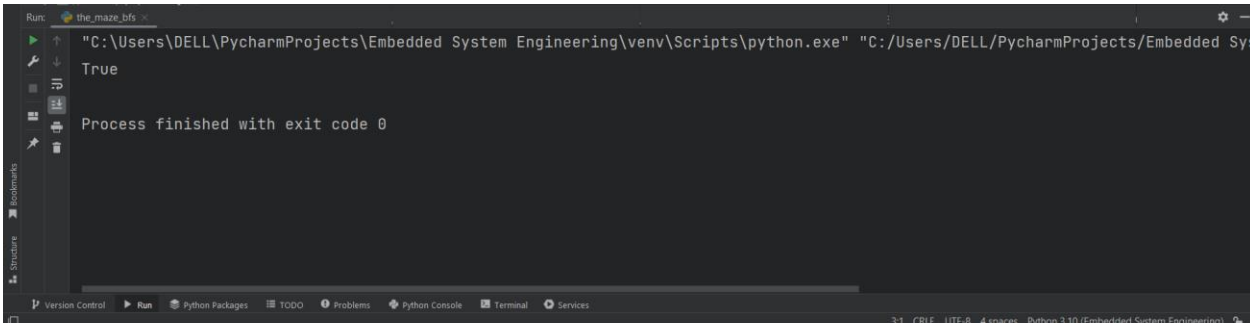
```
Visited: 0 C K G
         1 1 1 1
Queue: G
1. Remove K from the queue
2. Print: 0 C K

Visited: 0 C K G
         1 1 1 1
Queue:
1. Remove G from the queue
2. Print 0 C K G

Visited: 0 C K G D
         1 1 1 1 1
Queue: D
1. Add D to the queue
2. Mark D as visited

Visited: 0 C K G D
         1 1 1 1 1
Queue:
1. Remove D from the queue
2. Print: 0 C K G D

Visited: 0 C K G D A I
         1 1 1 1 1 1 1
Queue: A I
1. Add A, I to the queue
2. Mark A, I as visited

Visited: 0 C K G D A I
         1 1 1 1 1 1 1
Queue: I
1. Remove A from the queue
2. Print: 0 C K G D A

Visited: 0 C K G D A I B
         1 1 1 1 1 1 1 1
Queue: I B
1. Add B to the queue
2. Mark B as visited
```

```
Visited: 0 C K G D A I B
         1 1 1 1 1 1 1 1
Queue: B
1. Remove I from the queue
2. Print: 0 C K G D A I

Visited: 0 C K G D A I B U
         1 1 1 1 1 1 1 1 1
Queue: B U
1. Add U to the queue
2. Mark U as visited

Visited: 0 C K G D A I B U
         1 1 1 1 1 1 1 1 1
Queue: U
1. Remove B from the queue
2. Print 0 C K G D A I B

Visited: 0 C K G D A I B U
         1 1 1 1 1 1 1 1 1
Queue:
1. Remove U from the queue
2. Print 0 C K G D A I B U

Visited: 0 C K G D A I B U R
         1 1 1 1 1 1 1 1 1 1
Queue: R
1. Add R to the queue
2. Mark R as visited
```

0 C K G D A I B U R

Step 2.2: 490. The Maze (BFS)

```
class Solution:
    def hasPath(self, maze, start, destination):

        Q = [start]
        n = len(maze)
        m = len(maze[0])
        dirs = ((0, 1), (0, -1), (1, 0), (-1, 0))

        while Q:
            i, j = Q.pop(0)
            maze[i][j] = 2
            if i == destination[0] and j ==
destination[1]:
                return True

            for x, y in dirs:
                row = i + x
                col = j + y
                while 0 <= row < n and 0 <= col < m and
maze[row][col] != 1:
                    row += x
                    col += y
                row -= x
                col -= y
                if maze[row][col] == 0:
                    Q.append([row, col])

        return False


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]
obj = Solution()
print(obj.hasPath(maze, start, destination))
```



```
"C:\Users\DELL\PycharmProjects\Embedded System Engineering\venv\Scripts\python.exe" "C:/Users/DELL/PycharmProjects/Embedded Sy
True


Process finished with exit code 0
```