Project : "490. The Maze" Depth-First Traversal

By: Xinye L.

Table of Content

❑ Introduction

❑ Design

❑ Implementation

❑ Test

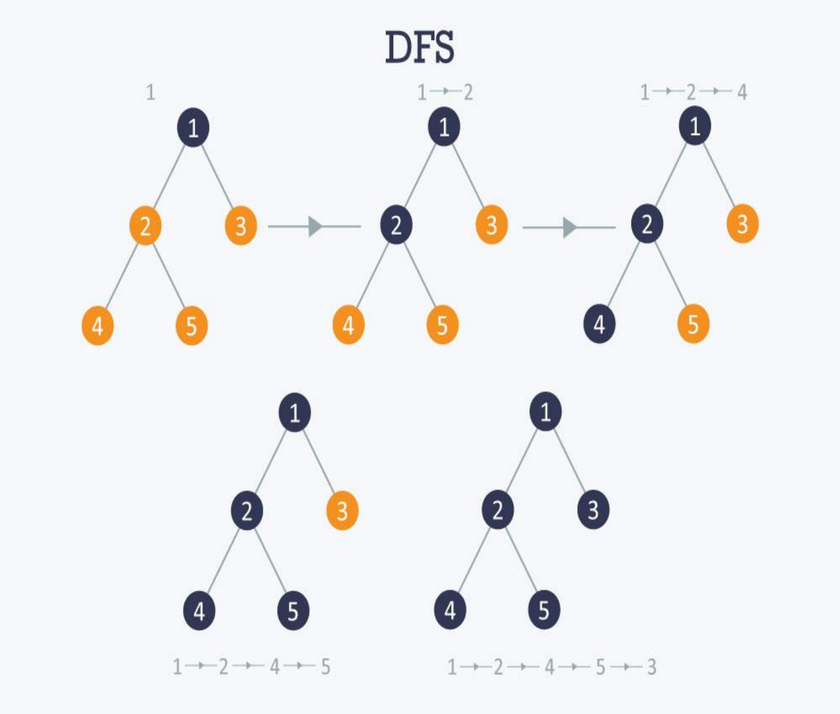
❑ Enhancement Ideas

❑ Conclusion

❑ References

Introduction

- ❏ Depth-First Search algorithm
- ❏ Manually solve the maze
- ❏ Python implementation



Depth-First Search Algorithm

❑ An algorithm for traversing or searching tree or graph data structures

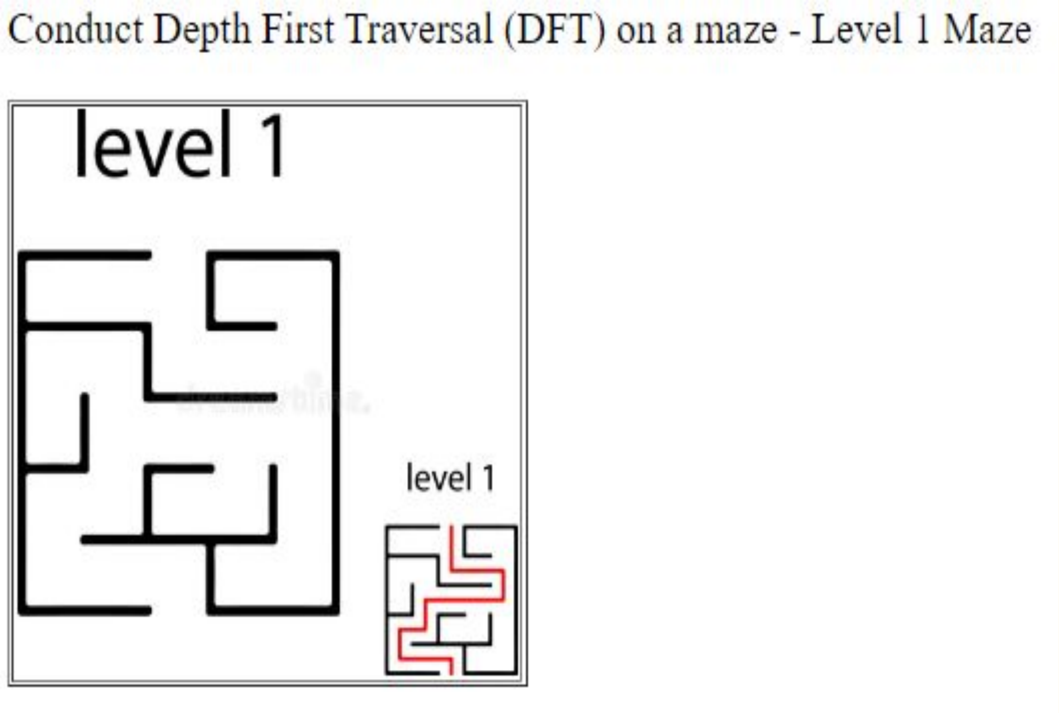
❑ Start at the root node, mark the node, and move to the adjacent unmarked node

❑ Continue this loop until there is no unmarked adjacent node

❑ Backtrack and check for other unmarked nodes, and traverse them

❑ Print the nodes in the path

Design - Tree

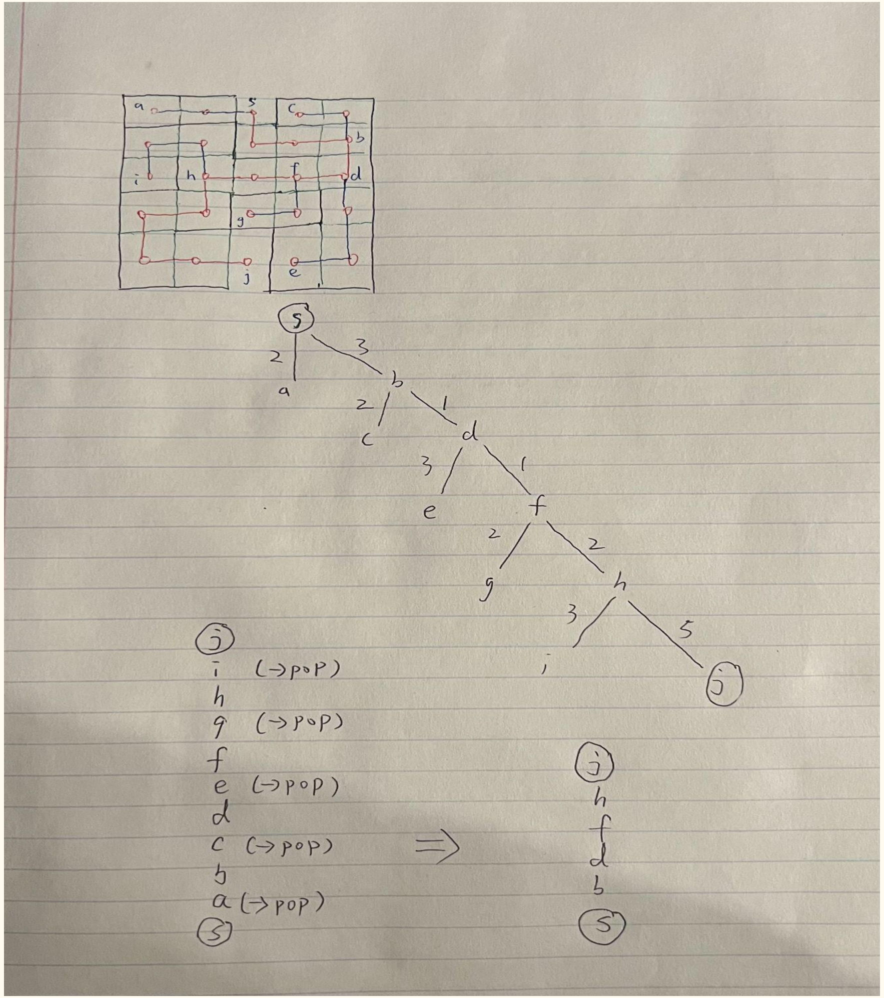


Conduct Depth First Traversal (DFT) on a maze - Level 1 Maze

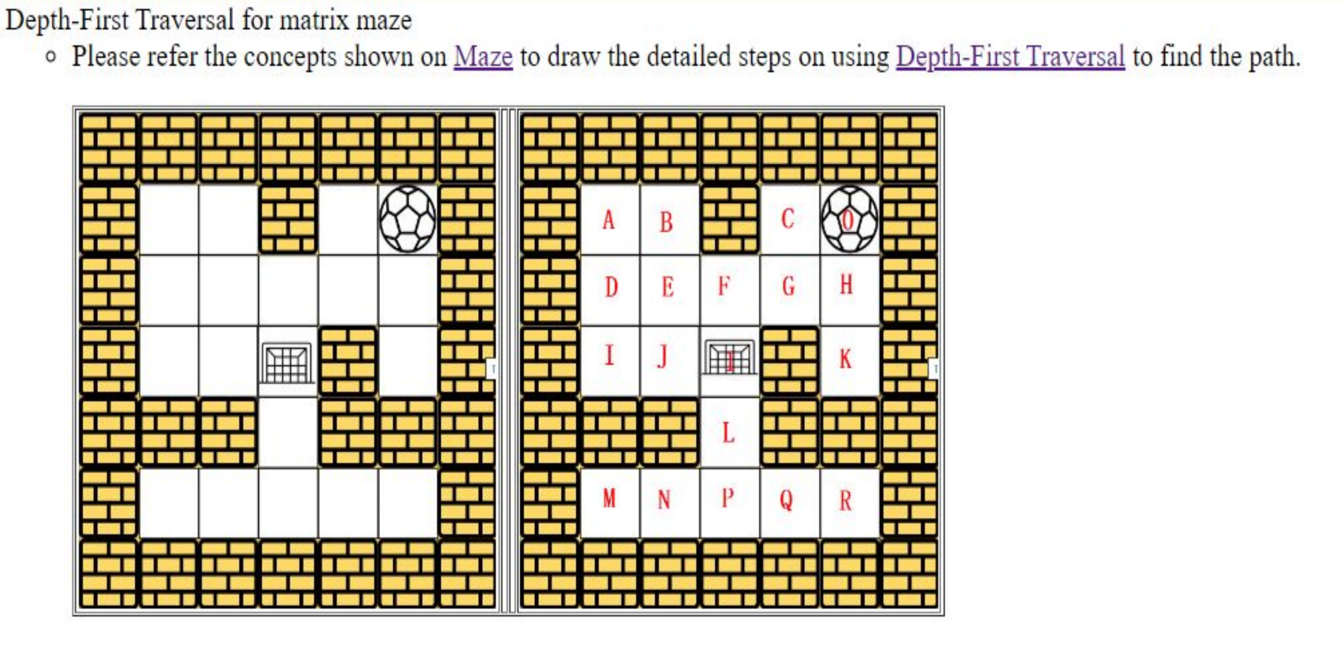
- Legged Robot moves on streets: DFS

Solution - Tree

- Mase
- Tree
- DFT



Design - Mase Matrix

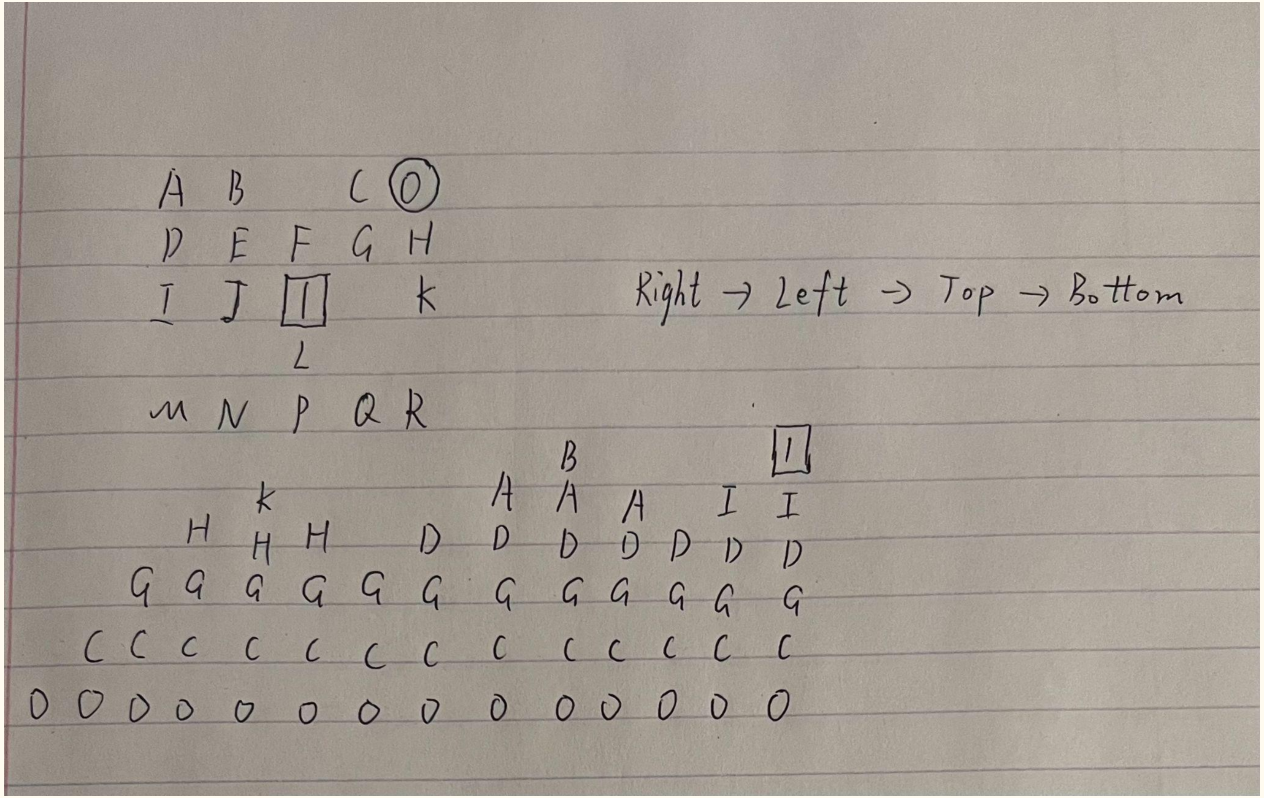


Depth-First Traversal for matrix maze

- Please refer the concepts shown on Maze to draw the detailed steps on using Depth-First Traversal to find the path.

- Wheeled robots moves in a hotel: DFS

Solution - Matrix Path



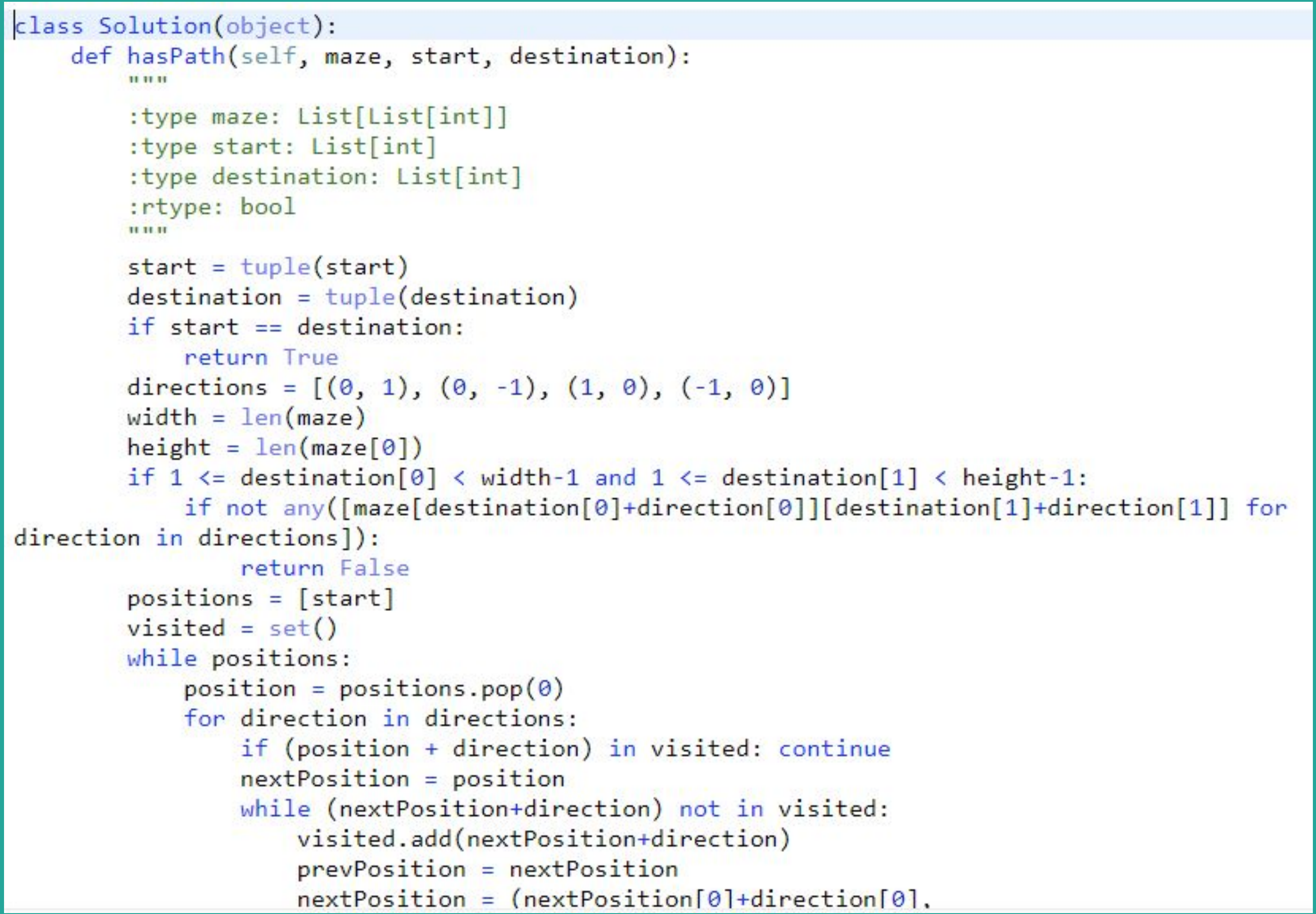
```
A  B     C  O
D  E  F  G  H
I  J  1     K
      L
m  N  P  Q  R
```

Right → Left → Top → Bottom

```
                           B           1
            K           A  A  A     I  I
         H  H  H     D  D  D  D  D  D  D
      G  G  G  G  G  G  G  G  G  G  G  G
   C  C  C  C  C  C  C  C  C  C  C  C  C
O  O  O  O  O  O  O  O  O  O  O  O  O  O
```

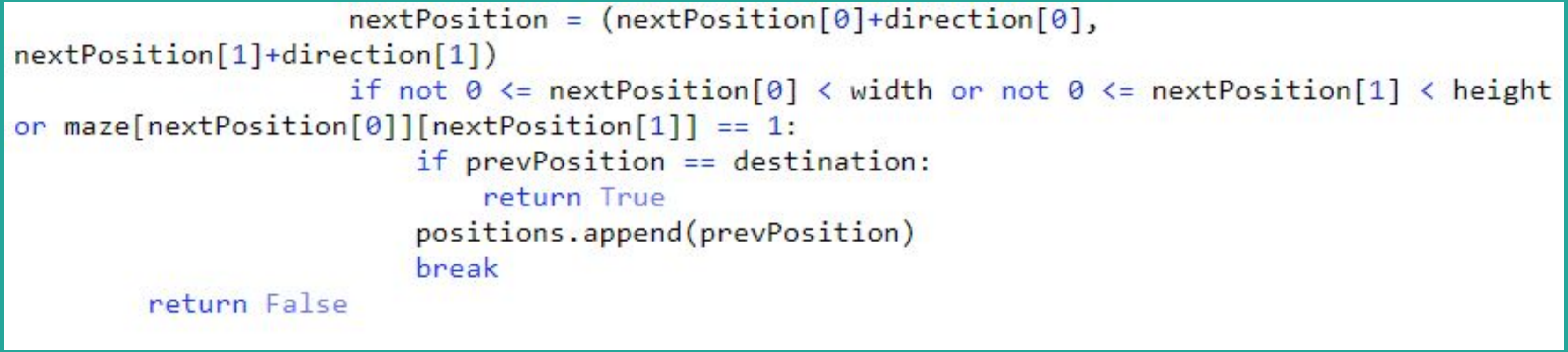
Implementation

- ❏ There is only one ball and one destination in the maze
- ❏ Both the ball and the destination exist on an empty space, and they will not be at the same position initially
- ❏ The given maze does not contain border but the border of the maze are all walls
- ❏ The maze contains at least 2 empty spaces, and both the width and height of the maze won’t exceed 100



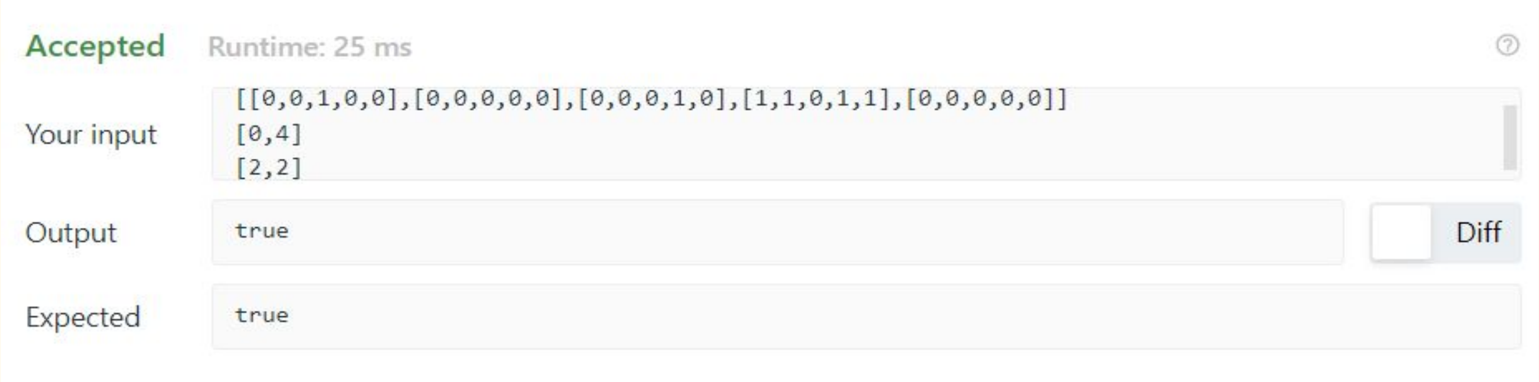
```
class Solution(object):
    def hasPath(self, maze, start, destination):
        """
        :type maze: List[List[int]]
        :type start: List[int]
        :type destination: List[int]
        :rtype: bool
        """
        start = tuple(start)
        destination = tuple(destination)
        if start == destination:
            return True
        directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]
        width = len(maze)
        height = len(maze[0])
        if 1 <= destination[0] < width-1 and 1 <= destination[1] < height-1:
            if not any([maze[destination[0]+direction[0]][destination[1]+direction[1]] for 
direction in directions]):
                return False
        positions = [start]
        visited = set()
        while positions:
            position = positions.pop(0)
            for direction in directions:
                if (position + direction) in visited: continue
                nextPosition = position
                while (nextPosition+direction) not in visited:
                    visited.add(nextPosition+direction)
                    prevPosition = nextPosition
                    nextPosition = (nextPosition[0]+direction[0],
```

Python implementation continued...



```
                    nextPosition = (nextPosition[0]+direction[0],
nextPosition[1]+direction[1])
                    if not 0 <= nextPosition[0] < width or not 0 <= nextPosition[1] < height
or maze[nextPosition[0]][nextPosition[1]] == 1:
                        if prevPosition == destination:
                            return True
                        positions.append(prevPosition)
                        break
        return False
```

Test cases



Accepted Runtime: 25 ms

Your input

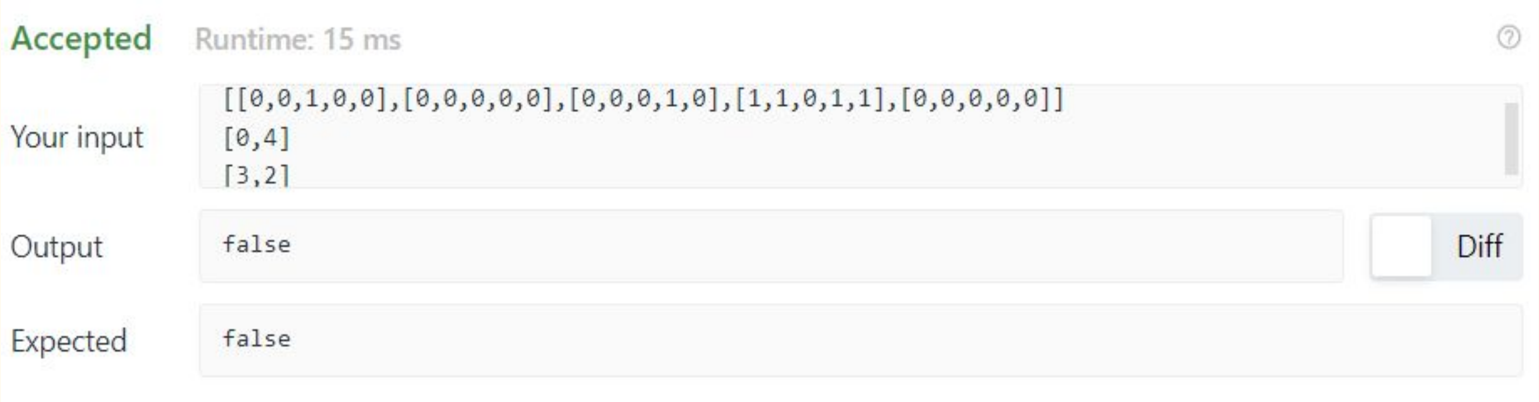
```
[[0,0,1,0,0],[0,0,0,0,0],[0,0,0,1,0],[1,1,0,1,1],[0,0,0,0,0]]
[0,4]
[2,2]
```

Output

```
true
```

Expected

```
true
```



Accepted Runtime: 15 ms

Your input

```
[[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]]
[0,4]
[3,2]
```

Output

```
false
```

Expected

```
false
```

Enhancement ideas

- ❑ Time complexity
- ❑ Auxiliary Space

Conclusion

- We introduced the depth-first search algorithm
- DFT problems can be solved manually
- We implemented and tested its python implementation

thank you

References

- ❑ GeeksforGeeks. (2019, February 4). *Depth First Search or DFS for a Graph - GeeksforGeeks*. GeeksforGeeks. https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/

- ❑ Jeffrey. (2020, March 22). *leetcode 490. The Maze (Python)*. (Jeffrey’s Blog). https://zhenyu0519.github.io/2020/03/22/lc490/