THE MAZE

Dipali Gajera (19645)
San Francisco Bay University
Summer, 2022

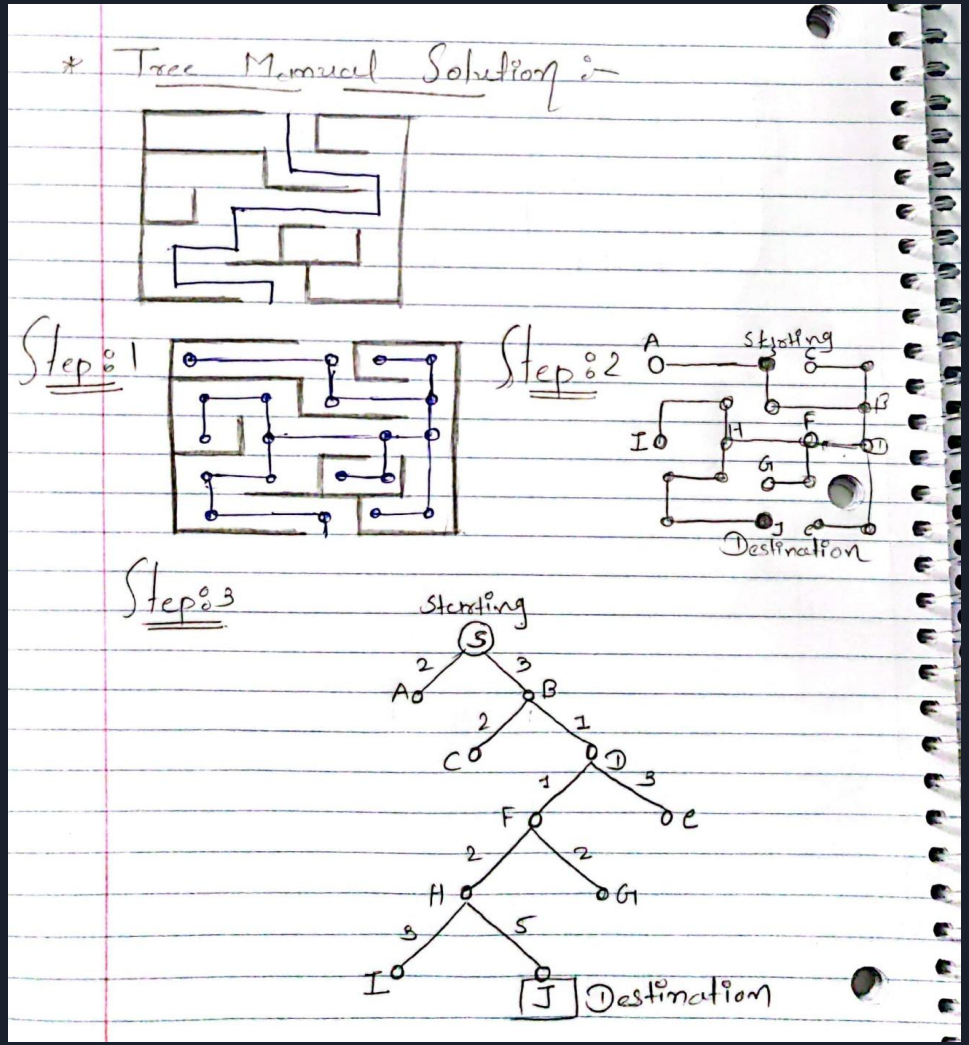
Contents

INTRODUCTION

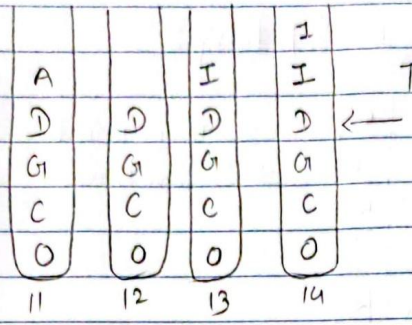
- The depth-first search algorithm is randomized in this algorithm. This method, which is frequently used with a stack, is one of the easiest for creating mazes.
- The algorithm picks a random neighboring cell that hasn't been visited yet after starting from a random cell.
- In order to facilitate backtracking, the algorithm breaks down the wall separating the two cells, marks the new cell as visited, and adds it to the stack.
- This process is carried out repeatedly by the algorithm, with a cell that has no unexplored neighbors being regarded as a dead end.

DESIGN

Tree Structure :-

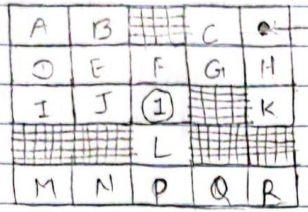


Matrix Solution



This is the final stack with the destination point.

* Matrix Manual Solution :-

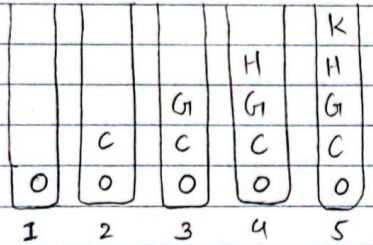


Starting Point

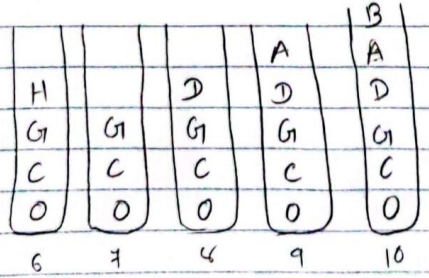
I → Destination

→ We will go do using this Search Sequence :-

R → L → T → B



Now, there is no where to go. so, we will pop out.



Now, Again will gonna pop out B and A

IMPLEMENTATION

```python
from typing import List

class Solution:
    def DFS(self, maze: List[List[int]], start: List[int], destination: List
        direction = [(1,0), (-1,0), (0,-1), (0,1)]
        m = len(maze)
        n = len(maze[0])

        stack = []
        seen = set()

        stack.append((start[0], start[1]))
        seen.add((start[0], start[1]))

        while stack:
            cur_i, cur_j = stack.pop()

            for d in direction:
                i = cur_i
                j = cur_j

                while 0 <= i < m and 0 <= j < n and maze[i][j] == 0:
                    i += d[0]
                    j += d[1]

                i -= d[0]
                j -= d[1]

                if i ==destination[0] and j == destination[1]:
                    return True

                if (i, j) not in seen:
                    stack.append((i,j))
                    seen.add((i, j))

        return False
```

```python
def main():
    obj = Solution()
    maze = []
    val = int(input("How many array do you want to enter : "))

    for i in range(0,val):
        arr = []
        n = int(input("How many elements do you want to enter : "))
        for j in range(0, n):
            ele = int(input("Enter array values : "))
            arr.append(ele)
        maze.append(arr)

    print()
    val = 2
    destination = []
    starting_position = []
    for k in range(0,val):
        pos = int(input("Enter the start position : "))
        destination.append(pos)

    print()
    for d in range(0,val):
        pos = int(input("Enter the position for Destination : "))
        starting_position.append(pos)

    print()
    result = obj.DFS(maze, destination, starting_position)
    print("-----------------------------------")
    print("The Maze output will be :-", '"', result, '"')
    print("-----------------------------------")

if __name__ == '__main__':
    main()
```

TEST

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

```
dipaligajera@Dipalis-MBP Algoritham % /usr/bin/python3 "/Users/dipaligajera/Desktop/untitled folder 2/Algoritham
/Maze/Final.py"
How many array do you want to enter : 5
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 0
Enter array values : 1
Enter array values : 0
Enter array values : 0
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 1
Enter array values : 0
How many elements do you want to enter : 5
Enter array values : 1
Enter array values : 1
Enter array values : 0
Enter array values : 1
Enter array values : 1
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0

Enter the start position : 0
Enter the start position : 4

Enter the position for Destination : 4
Enter the position for Destination : 4

------------------------------------
The Maze output will be :- " True "
------------------------------------
dipaligajera@Dipalis-MBP Algoritham %
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

```
dipaligajera@Dipalis-MBP Algoritham % /usr/bin/python3 "/Users/dipaligajera/Desktop/untitled folder 2/Algoritham
/Maze/Final.py"
How many array do you want to enter : 5
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 0
Enter array values : 1
Enter array values : 0
Enter array values : 0
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 1
Enter array values : 0
How many elements do you want to enter : 5
Enter array values : 1
Enter array values : 1
Enter array values : 0
Enter array values : 1
Enter array values : 1
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0

Enter the start position : 0
Enter the start position : 4

Enter the position for Destination : 3
Enter the position for Destination : 2

------------------------------------
The Maze output will be :- " False "
------------------------------------
dipaligajera@Dipalis-MBP Algoritham %
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

```
dipaligajera@Dipalis-MBP Algoritham % /usr/bin/python3 "/Users/dipaligajera/Desktop/untitled folder 2/Algoritham
/Maze/Final.py"
How many array do you want to enter : 5
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0
How many elements do you want to enter : 5
Enter array values : 1
Enter array values : 1
Enter array values : 0
Enter array values : 0
Enter array values : 1
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0
Enter array values : 0
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 1
Enter array values : 0
Enter array values : 0
Enter array values : 1
How many elements do you want to enter : 5
Enter array values : 0
Enter array values : 1
Enter array values : 0
Enter array values : 0
Enter array values : 0

Enter the start position : 4
Enter the start position : 3

Enter the position for Destination : 0
Enter the position for Destination : 1

------------------------------------
The Maze output will be :- " False "
------------------------------------
dipaligajera@Dipalis-MBP Algoritham %
```

ENHANCEMENT IDEAS

- This maze problem, we can also solve this using shortest path algorithm. Like, Dijkstra's Algorithm and Bellman Ford's Algorithm.
- Another way for this problem is using shortest path, Minimum Spanning Tree(MST).
- Converting the maze into X and Y squares is the either way and convert into tree using nodes.
- This shortest path is the easiest way to solve this maze problem .
- We can apply this algorithm in real world.
  - For example; Map.

CONCLUSION

- From this project, i learned a lot about Depth First Search(DFS) Algorithm and also compare with the other algorithms.
- The N, E, W, and S make it easier to move in the necessary directions in order to arrive to the target.

REFERENCE

- https://github.com/Areesha-Tahir/BFS-DFS-Maze-Solver-In-Python/blob/main/main.py
- https://www.youtube.com/watch?v=sTRK9mQgYuc
- https://codereview.stackexchange.com/questions/197356/searching-a-maze-using-dfs-and-bfs-in-python-3
- https://makeschool.org/mediabook/oa/tutorials/trees-and-mazes/generating-a-maze-with-dfs/

THANK YOU