**BFS Search**

from collections import deque

# BFS implementation

def bfs(maze, start, end):

    rows, cols = len(maze), len(maze[0])

    queue = deque([(start, [start])])

    visited = set()

    visited.add(start)

    while queue:

        (x, y), path = queue.popleft()

        # Check if the end is reached

        if (x, y) == end:

            return path

        # Explore neighbors

        for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:

            nx, ny = x + dx, y + dy

            if 0 <= nx < rows and 0 <= ny < cols and maze[nx][ny] == 1 and (nx, ny) not in visited:

                visited.add((nx, ny))

                queue.append(((nx, ny), path + [(nx, ny)]))

    return "No path found"

# Example usage for BFS

maze = [

    [1, 0, 1, 1, 1],

    [1, 1, 0, 1, 0],

    [0, 1, 1, 1, 0],

    [1, 0, 0, 1, 1],

    [1, 1, 1, 0, 1],

]

start = (0, 0)

end = (3, 3)

print("BFS Path:", bfs(maze, start, end))

print("-----------------------------------------------")

print( "Name : Prateek Kumar Panda \nRoll no: 23057030")

from datetime import date

print("Date :" ,date.today())

**DFS Search**

# DFS implementation

def dfs(maze, start, end):

    rows, cols = len(maze), len(maze[0])

    stack = [(start, [start])]

    visited = set()

    visited.add(start)

    all\_paths = []

    while stack:

        (x, y), path = stack.pop()

        # Check if the end is reached

        if (x, y) == end:

            all\_paths.append(path)

            continue

        # Explore neighbors

        for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:

            nx, ny = x + dx, y + dy

            if 0 <= nx < rows and 0 <= ny < cols and maze[nx][ny] == 1 and (nx, ny) not in visited:

                visited.add((nx, ny))

                stack.append(((nx, ny), path + [(nx, ny)]))

    return all\_paths if all\_paths else "No path found"

# Example usage for DFS

maze = [

    [1, 0, 1, 1, 1],

    [1, 1, 0, 1, 0],

    [0, 1, 1, 1, 0],

    [1, 0, 0, 1, 1],

    [1, 1, 1, 0, 1],

]

start = (0, 0)

end = (4, 4)

print("DFS Paths:", dfs(maze, start, end))

print("-----------------------------------------------")

print( "Name : Prateek Kumar Panda \nRoll no: 23057030")

from datetime import date

print("Date :" ,date.today())