Green University of Bangladesh
Department of Computer Science and Engineering(CSE)
Faculty of Sciences and Engineering
Semester: (Spring, Year:2024), B.Sc. in CSE (Day)

Course Code:CSE-316 Section: D-14

Student Details

Name	ID
Ashraful Islam Miraj	221902231

Lab Date : 19-02-2025
Submission Date : 25-02-2025
Course Teacher's Name : Md. Sabbir Hosen Mamun

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: …… Signature:.....

Comments:..... Date:.....

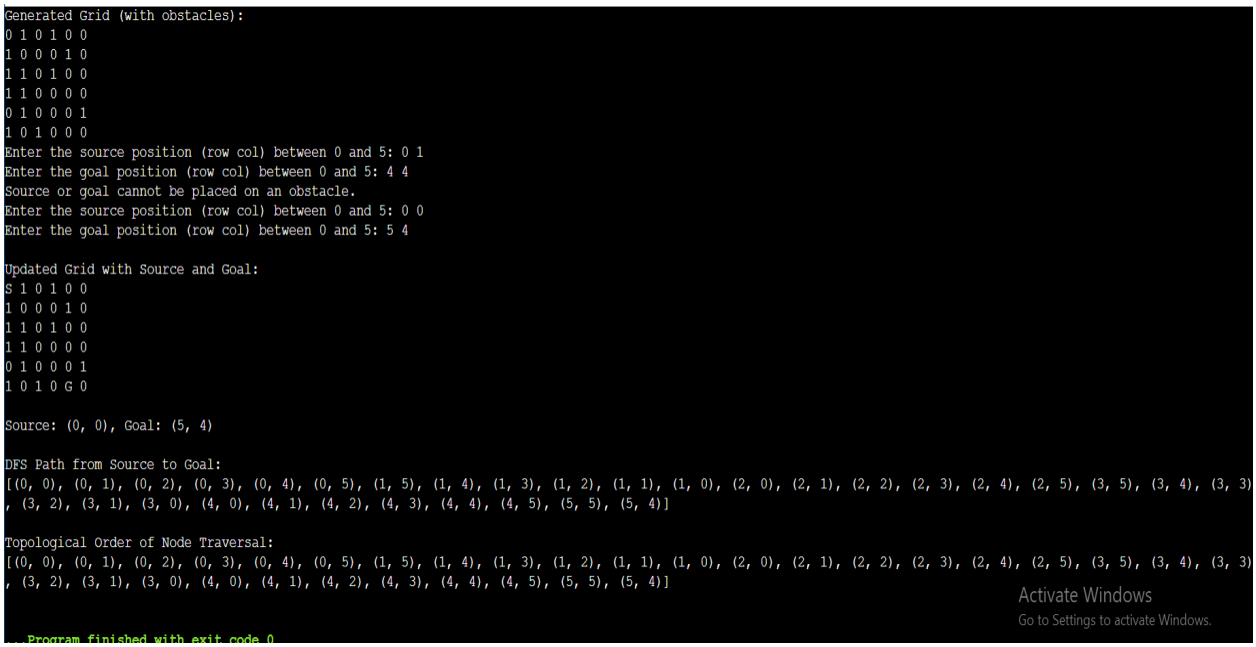
Code:

```
import random

def generate_grid(N):
    grid = [['0' for _ in range(N)] for _ in range(N)]

    for i in range(random.randint(5, N * N // 2)):
        x, y = random.randint(0, N-1), random.randint(0, N-1)
        while grid[x][y] == 'S' or grid[x][y] == 'G':
            x, y = random.randint(0, N-1), random.randint(0, N-1)
        grid[x][y] = '1'
    return grid


def dfs(grid, source, goal, N):
    stack = [source]
    visited = set()
    parent = {source: None}

    directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
    topological_order = []

    while stack:
        current = stack.pop()


        if current not in visited:
            visited.add(current)
            topological_order.append(current)


            if current == goal:
                path = []
                while current != source:
                    path.append(current)
                    current = parent[current]
                path.append(source)
```

```
            path.reverse()
            return path, topological_order


        for d in directions:
            next_node = (current[0] + d[0], current[1] + d[1])
            if 0 <= next_node[0] < N and 0 <= next_node[1] < N and next_node not in visited and
grid[next_node[0]][next_node[1]] != '#':
                stack.append(next_node)
                parent[next_node] = current

    return None, topological_order


def print_grid(grid):
    for row in grid:
        print(" ".join(row))


def main():

    while True:
        try:
            N = int(input("Enter the size of the grid (between 4 and 7): "))
            if 4 <= N <= 7:
                break
            else:
                print("Please enter a number between 4 and 7.")
        except ValueError:
            print("Invalid input. Please enter a valid integer.")


    grid = generate_grid(N)

    print("\nGenerated Grid (with obstacles):")
    print_grid(grid)


    while True:
        try:
```

```
        source = tuple(map(int, input(f"Enter the source position (row col) between 0 and {N-1}: ").split()))
        goal = tuple(map(int, input(f"Enter the goal position (row col) between 0 and {N-1}: ").split()))


        if (0 <= source[0] < N and 0 <= source[1] < N) and (0 <= goal[0] < N and 0 <= goal[1] < N) and source != goal:
            if grid[source[0]][source[1]] != '1' and grid[goal[0]][goal[1]] != '1':
                break
            else:
                print("Source or goal cannot be placed on an obstacle.")
        else:
            print("Invalid positions. Make sure both positions are within the grid and source != goal.")
    except ValueError:
        print("Invalid input. Please enter valid row and column numbers.")


  grid[source[0]][source[1]] = 'S'
  grid[goal[0]][goal[1]] = 'G'

  print("\nUpdated Grid with Source and Goal:")
  print_grid(grid)
  print(f"\nSource: {source}, Goal: {goal}")


  path, topological_order = dfs(grid, source, goal, N)

  if path:
      print("\nDFS Path from Source to Goal:")
      print(path)
  else:
      print("\nNo path found from Source to Goal.")

  print("\nTopological Order of Node Traversal:")
  print(topological_order)

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

OUTPUT:



https://github.com/Aashu002-ux/GraphTraversal./edit/main/README.md