**Depth-First Search (DFS) Algorithm**

**Aim**

To implement the Depth-First Search (DFS) algorithm in Python to find the path from a start point to a goal in a grid with obstacles, by exploring as far as possible along each branch before backtracking.

**Procedure**

1. **Input the grid dimensions (rows and columns)**, followed by the grid layout where 0 represents free spaces and 1 represents blocked spaces.
2. **Input the start and goal points** from the user to define the starting and target positions in the grid.
3. **Initialize the DFS stack**, starting with the start point, and keep track of the visited nodes to avoid revisiting.
4. **Explore the grid** by popping the stack and checking neighboring nodes (up, down, left, right). If a valid, unvisited node is found, push it onto the stack.
5. **Return the path** if the goal is reached; otherwise, return "No path" if no valid path exists.

**Program**

def dfs(grid, start, goal):

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

stack = [(start, [start])]

visited = set()

while stack:

(curr, path) = stack.pop()

if curr == goal:

return path

if curr in visited:

continue

visited.add(curr)

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

x, y = curr[0] + dx, curr[1] + dy

if 0 <= x < rows and 0 <= y < cols and grid[x][y] == 0 and (x, y) not in visited:

stack.append(((x, y), path + [(x, y)]))

return None # If no path is found

r, c = map(int, input("Enter rows and columns: ").split())

print("Enter grid (0=free, 1=blocked):")

grid = [list(map(int, input().split())) for \_ in range(r)]

start = tuple(map(int, input("Enter start (row col): ").split()))

goal = tuple(map(int, input("Enter goal (row col): ").split()))

path = dfs(grid, start, goal)

print("Path:", path if path else "No path")

**Input**

Enter rows and columns: 5 5

Enter grid (0=free, 1=blocked):

0 0 0 0 0

0 1 1 1 0

0 1 0 0 0

0 1 0 1 0

0 0 0 0 0

Enter start (row col): 0 0

Enter goal (row col): 4 4

**Output**

Path: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2), (3, 2), (4, 2), (4, 3), (4, 4)]

**Result**

The DFS algorithm successfully finds a path from the start to the goal if one exists. If no valid path is found, it returns "No path".