Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year:2025), B.Sc. in CSE (Day)

Lab Report 02: IDDFS

Course Title: Artificial Intelligence Lab

Course Code: CSE-316 Section:221-14

Student Details

Name		ID
1.	Md.Mohibullah	221902083

Lab Date : 26-02-2025

Submission Date : 03-04-2025

Course Teacher's Name : Md. Sabbir Hosen Mamun

Lab Report Status

Marks: …… **Signature:.....**

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

Lab Report Name: Iterative Deepening Depth-First Search (IDDFS) in Maze Navigation

1. Introduction In this lab, we implement Iterative Deepening Depth-First Search (IDDFS) to find a path in a given 2D grid-based maze. Each cell in the grid is either an empty space (0) or a wall (1). The algorithm explores possible paths from a given start node to a specified target node by iteratively deepening the search depth until a valid path is found or the maximum depth is reached.

2. Objectives

- To understand and implement the Iterative Deepening Depth-First Search (IDDFS) algorithm.
- To explore different pathfinding techniques in a grid-based environment.
- To analyze the advantages of IDDFS over traditional DFS in terms of memory efficiency.
- To test and evaluate IDDFS performance on different maze configurations.
- To develop problem-solving skills in handling constraints like walls and movement restrictions in grid-based pathfinding.

3. Problem Statement Given an N x M grid representing a maze, determine whether a valid path exists from the given start cell to the target cell. The movement is restricted to adjacent empty cells (0) in four directions: up, down, left, and right. Walls (1) cannot be traversed, and each cell may be visited only once per exploration path.

4. Algorithm Explanation IDDFS is a combination of Depth-First Search (DFS) and Depth-Limited Search (DLS). It progressively deepens the search depth until the target is found or the depth limit is exceeded.

Steps of IDDFS:

1. Start with depth = 0.
2. Perform a Depth-Limited Search (DLS) with the current depth.
3. If the target node is found, return the path.
4. If not, increment the depth and repeat until a maximum depth is reached.
5. If no path is found at the maximum depth, return "Path not found".

5. Implementation

```
class IDDFS:
    def __init__(self, grid, start, target):
        self.grid = grid
        self.N = len(grid)
        self.M = len(grid[0])
        self.start = start
```

```
        self.target = target
        self.directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]

    def dls(self, node, depth, path, visited):
        x, y = node
        if node == self.target:
            return True, path
        if depth == 0:
            return False, None

        visited.add(node)
        for dx, dy in self.directions:
            nx, ny = x + dx, y + dy
            if 0 <= nx < self.N and 0 <= ny < self.M and self.grid[nx][ny]
== 0 and (nx, ny) not in visited:
                found, new_path = self.dls((nx, ny), depth - 1, path +
[(nx, ny)], visited)
                if found:
                    return True, new_path
        visited.remove(node)
        return False, None

    def iddfs(self, max_depth):
        for depth in range(max_depth + 1):
            visited = set()
            found, path = self.dls(self.start, depth, [self.start],
visited)
            if found:
                return f"Path found at depth {depth} using IDDFS", path
        return f"Path not found at max depth {max_depth} using IDDFS",
None

# Function to take user input
def get_input():
    N, M = map(int, input().split())
    grid = [list(map(int, input().split())) for _ in range(N)]
    start_x, start_y = map(int, input().split()[1:])
    target_x, target_y = map(int, input().split()[1:])
    return grid, (start_x, start_y), (target_x, target_y)
```

```python
if __name__ == "__main__":
    grid, start, target = get_input()
    solver = IDDFS(grid, start, target)
    result, path = solver.iddfs(len(grid) * len(grid[0]))
    print(result)
    if path:
        print("Traversal Order:", path)
```

Output:

```
PS E:\8th semester\AI Lab\Lab Report 02> & "C:/Program Files/Python313/python.exe"
"
4 4
0 0 1 0
1 0 1 0
0 0 0 0
1 1 0 1
Start: 0 0
Target: 2 3
Path found at depth 5 using IDDFS
Traversal Order: [(0, 0), (0, 1), (1, 1), (2, 1), (2, 2), (2, 3)]
PS E:\8th semester\AI Lab\Lab Report 02>
```

7. Conclusion Iterative Deepening Depth-First Search (IDDFS) is a highly effective pathfinding algorithm that balances the benefits of Depth-First Search (DFS) and Breadth-First Search (BFS). It ensures memory efficiency by exploring the search space level by level rather than storing all possible paths in memory at once. This makes IDDFS well-suited for problems where memory constraints are a concern.

Our experiment demonstrates that IDDFS successfully finds a path when one exists and gracefully handles cases where no path is available. However, one downside is that the algorithm may repeatedly explore the same nodes at different depths, which can lead to redundant computations. Despite this, IDDFS remains a practical choice for search problems in constrained environments such as grid-based navigation, AI pathfinding, and puzzle-solving applications. Future enhancements could include optimizations to reduce redundant explorations or integrating heuristics to guide the search more efficiently.

GitHub Link: https://github.com/programmermahi/Artificial-Intelligence/tree/main/LabReport02-IDDFS