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

**Faculty of Sciences and Engineering**
**Semester: (Fall, Year:2025), B.Sc. in CSE (Day)**

**LAB REPORT NO # 01**
**Course Title: Artificial Intelligence Lab**

**Course Code:**CSE 404-CSE(181) **Section:** 223_D4

**Lab Experiment Name: Implementation of IDDFS Algorithm**

**Student Details**

| Name | ID |
|---|---|
| Mujahidul Islam | 193002052 |

**Lab Date: 29/11/2025**
**Submission Date: 12/12/2025**
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:..............................**

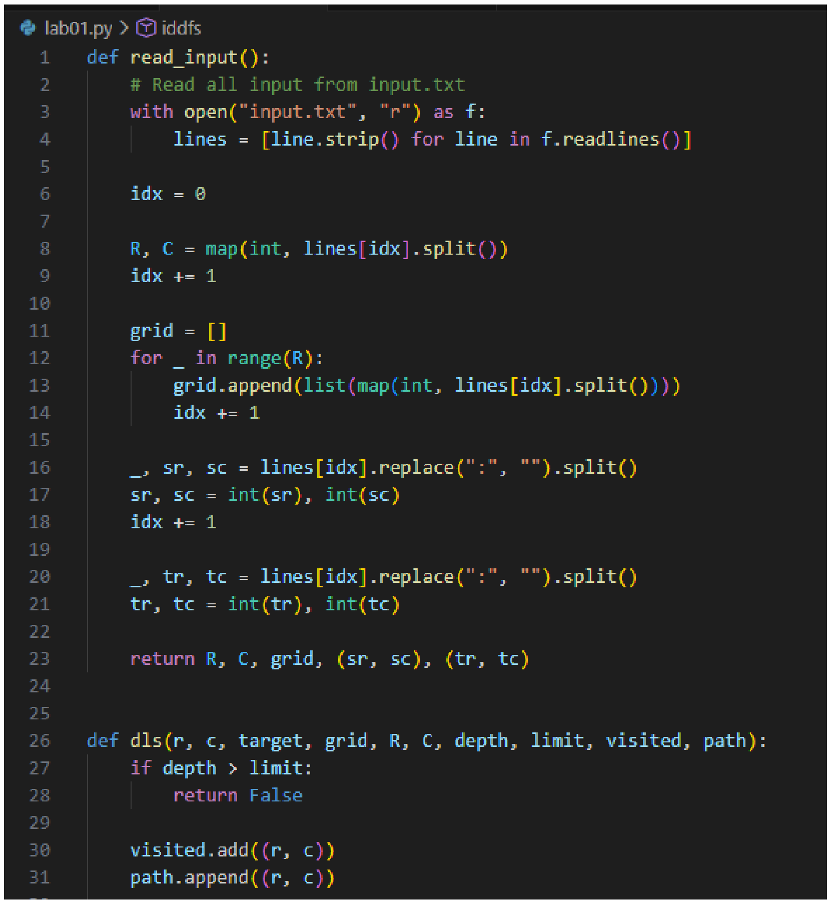
## 1. Introduction

In this lab, we implemented the Iterative Deepening Depth-First Search (IDDFS) algorithm to find a path in a 2D maze. Each cell in the maze is either an empty space or a wall, and movement is allowed only in four directions: up, down, left, and right. IDDFS combines the advantages of depth-first search and breadth-first search, providing memory efficiency while guaranteeing that the shortest path (in terms of depth) is found if it exists.

## 2. Objectives:

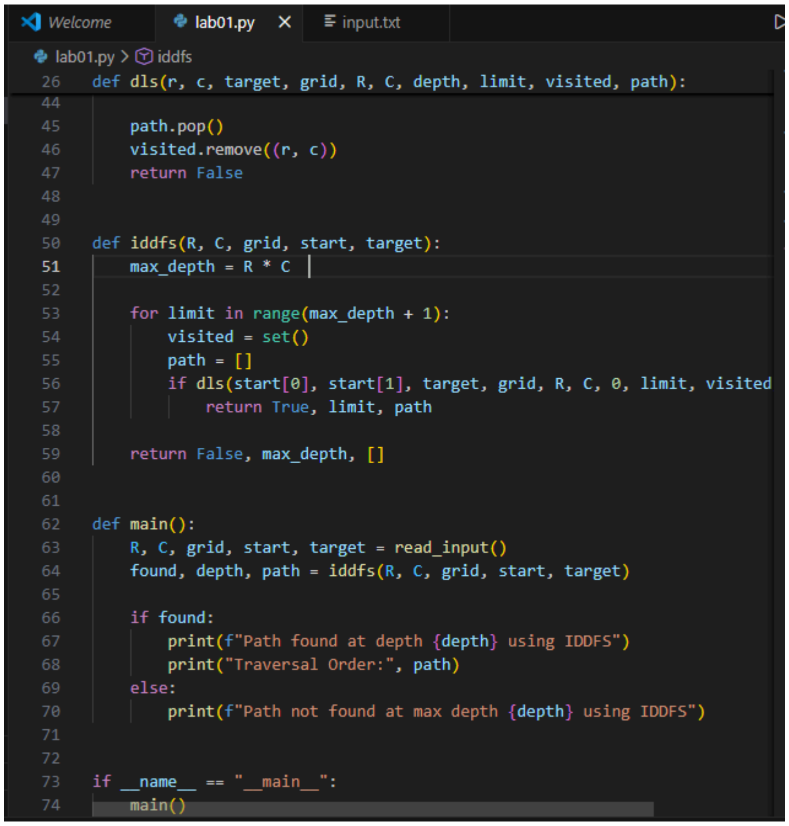
- Implement the IDDFS algorithm to explore a maze.
- Read the maze and start/target positions from an input.txt file.
- Determine whether a valid path exists between the start and target cells.
- Output the depth at which the path is found and the traversal order.
- Understand the trade-offs between DFS, BFS, and IDDFS in terms of memory and completeness.

## 3. Implementation:



lab01.py > iddfs

```python
def read_input():
    # Read all input from input.txt
    with open("input.txt", "r") as f:
        lines = [line.strip() for line in f.readlines()]

    idx = 0

    R, C = map(int, lines[idx].split())
    idx += 1

    grid = []
    for _ in range(R):
        grid.append(list(map(int, lines[idx].split())))
        idx += 1

    _, sr, sc = lines[idx].replace(":", "").split()
    sr, sc = int(sr), int(sc)
    idx += 1

    _, tr, tc = lines[idx].replace(":", "").split()
    tr, tc = int(tr), int(tc)

    return R, C, grid, (sr, sc), (tr, tc)


def dls(r, c, target, grid, R, C, depth, limit, visited, path):
    if depth > limit:
        return False

    visited.add((r, c))
    path.append((r, c))
```

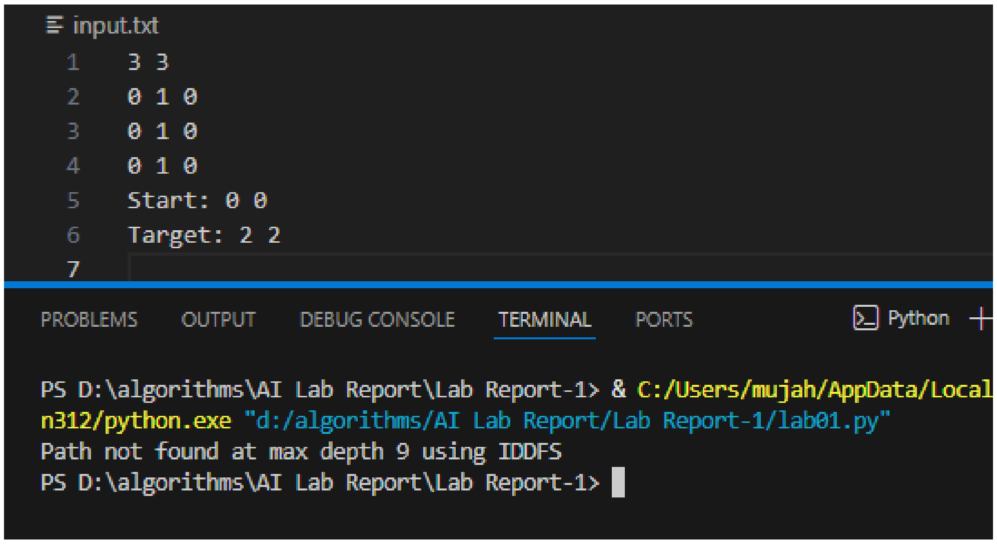


Welcome lab01.py input.txt

lab01.py > iddfs

```python
def dls(r, c, target, grid, R, C, depth, limit, visited, path):

    path.pop()
    visited.remove((r, c))
    return False


def iddfs(R, C, grid, start, target):
    max_depth = R * C

    for limit in range(max_depth + 1):
        visited = set()
        path = []
        if dls(start[0], start[1], target, grid, R, C, 0, limit, visited
            return True, limit, path

    return False, max_depth, []


def main():
    R, C, grid, start, target = read_input()
    found, depth, path = iddfs(R, C, grid, start, target)

    if found:
        print(f"Path found at depth {depth} using IDDFS")
        print("Traversal Order:", path)
    else:
        print(f"Path not found at max depth {depth} using IDDFS")


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

## 4. Output:

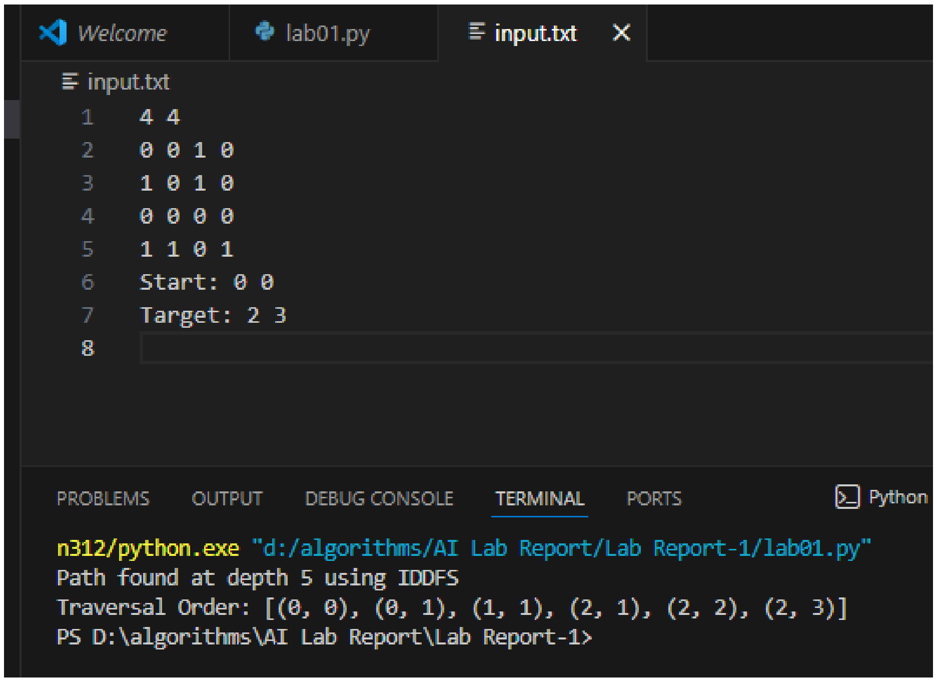


input.txt

```
3 3
0 1 0
0 1 0
0 1 0
Start: 0 0
Target: 2 2

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
PS D:\algorithms\AI Lab Report\Lab Report-1> & C:/Users/mujah/AppData/Local
n312/python.exe "d:/algorithms/AI Lab Report/Lab Report-1/lab01.py"
Path not found at max depth 9 using IDDFS
PS D:\algorithms\AI Lab Report\Lab Report-1>
```



```
4 4
0 0 1 0
1 0 1 0
0 0 0 0
1 1 0 1
Start: 0 0
Target: 2 3

```

```
n312/python.exe "d:/algorithms/AI Lab Report/Lab Report-1/lab01.py"
Path found at depth 5 using IDDFS
Traversal Order: [(0, 0), (0, 1), (1, 1), (2, 1), (2, 2), (2, 3)]
PS D:\algorithms\AI Lab Report\Lab Report-1>
```

**5. Discussion:**

In this lab, we implemented IDDFS (Iterative Deepening Depth-First Search) to find a path in a 2D maze. The program reads the maze from an input file and checks if a path exists from start to target. IDDFS combines DFS and BFS, using less memory while still finding the shortest path.

The algorithm increases depth limits iteratively until the target is found or maximum depth is reached. It correctly finds paths when possible and reports if no path exists. This lab helped understand how iterative deepening works and how it improves DFS for unknown depth problems.