Symbiosis Institute of Technology

Department of Computer Science and Engineering

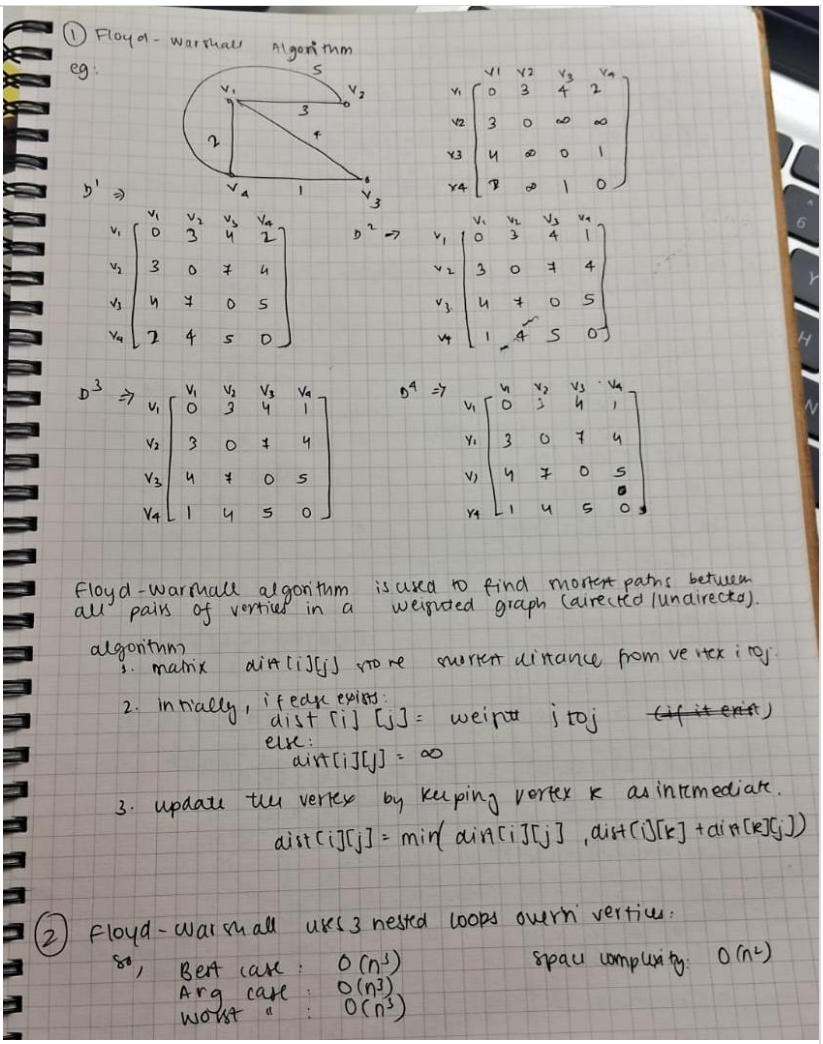
Academic Year 2025-26

Design Analysis of Algorithm– Lab

Batch 2023-27 - Sem V

Lab Assignment No:- 6	
Name of Student	Deepti Pal
PRN No.	23070122081
Batch	2023-27
Class	CSE – A3
Academic Year & Semester	2025-26 , TY, 5th sem
Date of Submission	22 sept 2025
Title of Assignment:	Assume that you are a software engineer working for a logistics company that operates a fleet of delivery trucks across multiple cities. The company wants to optimize delivery routes to minimize fuel consumption and delivery time. The road network between the cities can be represented as a weighted graph, where: • Each **vertex** represents a city. • Each **edge** represents a direct road between two cities. • Each **weight** on the edge represents the **distance in kilometers** or **average travel time** between cities.

	Problem Statement: Implement **Floyd's Algorithm** to compute the shortest paths between **every pair of cities**. Then, use the result to answer the following: 1. What is the **shortest distance** between City A and City D? 2. Which **intermediate cities** (if any) should the delivery truck pass through to achieve this shortest distance? Write a program to: ● Read the distance matrix representing the weighted graph from a file. ● Apply **Floyd's Algorithm** to compute all-pairs shortest paths. ● Output the updated shortest distance matrix. ● Allow querying of shortest paths between any two cities.
Theory: (Handwritten)	1. Explain Floyd's algorithm with an example. Show all steps 2. Discuss Complexity (Best, Worst, and Average Case) & Space Complexity



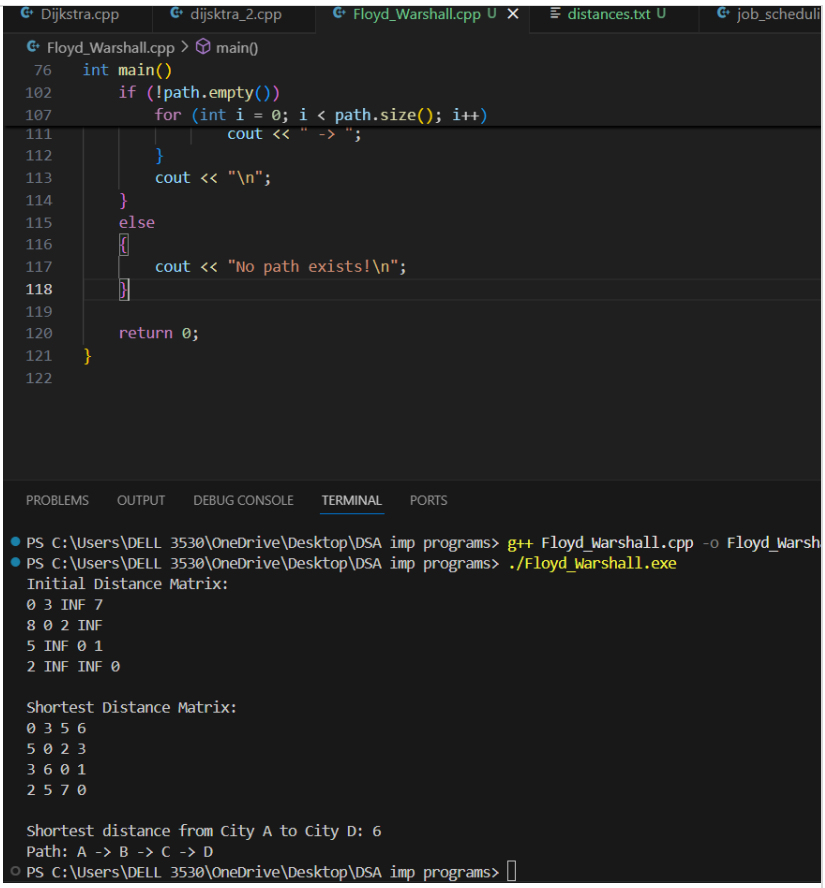
Source code

```
#include <iostream>
#include <vector>
#include <algorithm>

const int INF = 1e9;

vector<vector<int>> readMatrix(const string &filename)
{
    ifstream file(filename);
    vector<vector<int>> matrix;
    string token;

    while (true)
    {
        vector<int> row;
        string line;
```

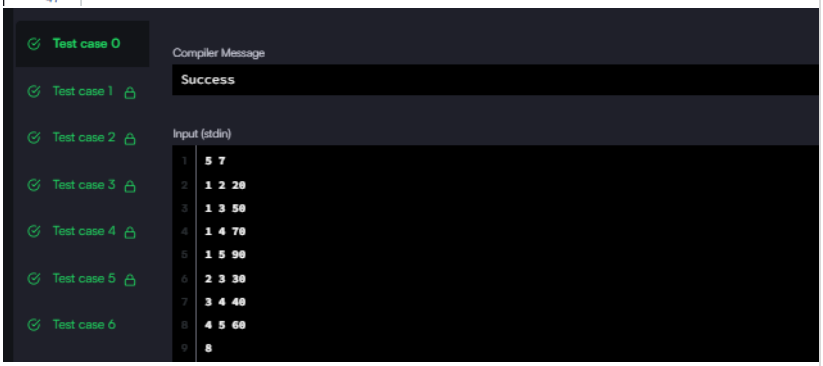
```
        if (!getline(file, line))
            break;
        stringstream ss(line);

        while (ss >> token)
        {
            if (token == "INF")
                row.push_back(INF);
            else
                row.push_back(stoi(token));
        }
        if (!row.empty())
            matrix.push_back(row);
    }
    return matrix;
}

void floydWarshall(vector<vector<int>> &dist,
vector<vector<int>> &next)
{
    int n = dist.size();
    next.assign(n, vector<int>(n, -1));

    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if (dist[i][j] != INF)
                next[i][j] = j;

    for (int k = 0; k < n; k++)
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                if (dist[i][k] + dist[k][j] < dist[i][j])
                {
                    dist[i][j] = dist[i][k] + dist[k][j];
                    next[i][j] = next[i][k];
                }
}

vector<int> getPath(int u, int v, const
vector<vector<int>> &next)
{
    if (next[u][v] == -1)
        return {};
    vector<int> path = {u};
    while (u != v)
```

```
    {
        u = next[u][v];
        path.push_back(u);
    }
    return path;
}

void printMatrix(const vector<vector<int>> &matrix)
{
    for (auto &row : matrix)
    {
        for (int val : row)
            cout << (val == INF ? "INF" : to_string(val))
<< " ";
        cout << "\n";
    }
}

int main()
{

    vector<vector<int>> dist =
readMatrix("distances.txt");

    if (dist.empty())
    {
        cout << "Error: distance matrix is empty. Check
file path/format!\n";
        return 0;
    }

    cout << "Initial Distance Matrix:\n";
    printMatrix(dist);

    vector<vector<int>> next;
    floydWarshall(dist, next);

    cout << "\nShortest Distance Matrix:\n";
    printMatrix(dist);

    // example
    int start = 0, end = 3;
    string cityNames = "ABCD";

    vector<int> path = getPath(start, end, next);
```

```
    if (!path.empty())
    {
        cout << "\nShortest distance from City " << cityNames[start]
             << " to City " << cityNames[end] << ": " << dist[start][end] << "\n";
        cout << "Path: ";
        for (int i = 0; i < path.size(); i++)
        {
            cout << cityNames[path[i]];
            if (i < path.size() - 1)
                cout << " -> ";
        }
        cout << "\n";
    }
    else
    {
        cout << "No path exists!\n";
    }

    return 0;
}
```

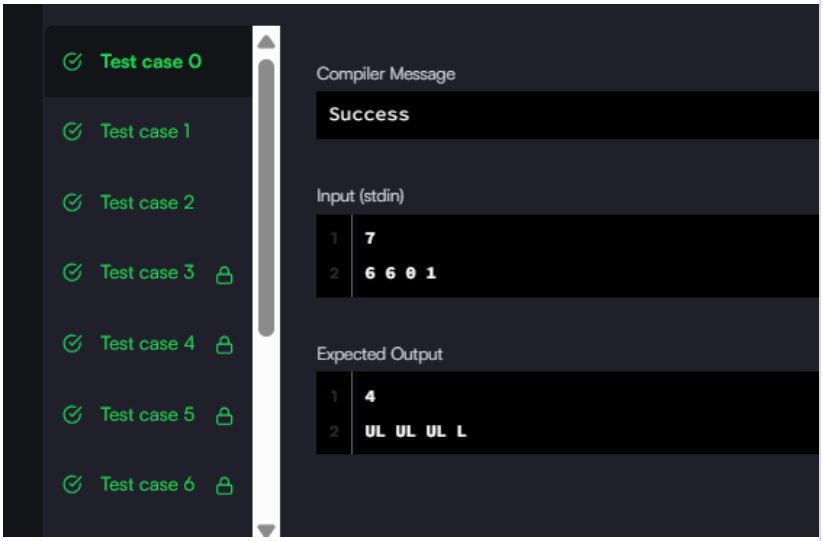
Output Screenshots (if applicable)	
Problems Solved from Hacker Rank	1. https://www.hackerrank.com/challenges/floyd-city-of-blinding-lights/problem

```cpp
#include <bits/stdc++.h>
using namespace std;

const int INF = 1000000000;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    // Initialize distance matrix
    vector<vector<int>> dist(n + 1, vector<int>(n + 1, INF));
    for (int i = 1; i <= n; i++) dist[i][i] = 0;

    // Read edges
    for (int i = 0; i < m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        dist[u][v] = w;  // overwrite last edge if multiple
    }

    // Floyd-Warshall algorithm
    for (int k = 1; k <= n; k++) {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (dist[i][k] < INF && dist[k][j] < INF) {
                    dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);
                }
            }
        }
    }

    // Process queries
    int q;
    cin >> q;
    while (q--) {
        int u, v;
        cin >> u >> v;
        if (dist[u][v] >= INF) cout << -1 << "\n";
        else cout << dist[u][v] << "\n";
    }

    return 0;
}
```



2. https://www.hackerrank.com/challenges/red-knights-shortest-path/problem

```cpp
#include <bits/stdc++.h>
using namespace std;

struct Cell {
    int r, c;
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;
    int i_start, j_start, i_end, j_end;
    cin >> i_start >> j_start >> i_end >> j_end;

    const int dr[6] = { -2, -2,  0,  2,  2,  0 };
    const int dc[6] = { -1,  1,  2,  1, -1, -2 };
    const string moveName[6] = { "UL", "UR", "R", "LR", "LL", "L" };


    vector<vector<int>> dist(n, vector<int>(n, -1));

    vector<vector<pair<Cell, int>>> parent(n, vector<pair<Cell, int>>(n, { {-1, -1}, -1 }));

    queue<Cell> q;
    dist[i_start][j_start] = 0;
    q.push({ i_start, j_start });

    while (!q.empty()) {
        Cell cur = q.front();
        q.pop();

        if (cur.r == i_end && cur.c == j_end) {
            break;
        }

        for (int m = 0; m < 6; m++) {
            int nr = cur.r + dr[m];
            int nc = cur.c + dc[m];
            if (nr < 0 || nr >= n || nc < 0 || nc >= n) continue;
            if (dist[nr][nc] != -1) continue;  // already visited

            dist[nr][nc] = dist[cur.r][cur.c] + 1;
            parent[nr][nc] = { { cur.r, cur.c }, m };
            q.push({ nr, nc });
        }
    }

```



3. https://www.hackerrank.com/challenges/clues-on-a-binary-path/problem
4. https://www.hackerrank.com/challenges/subtrees-and-paths/problem

Conclusion	Thus, we have studied Floyd's algorithm using a dynamic method.