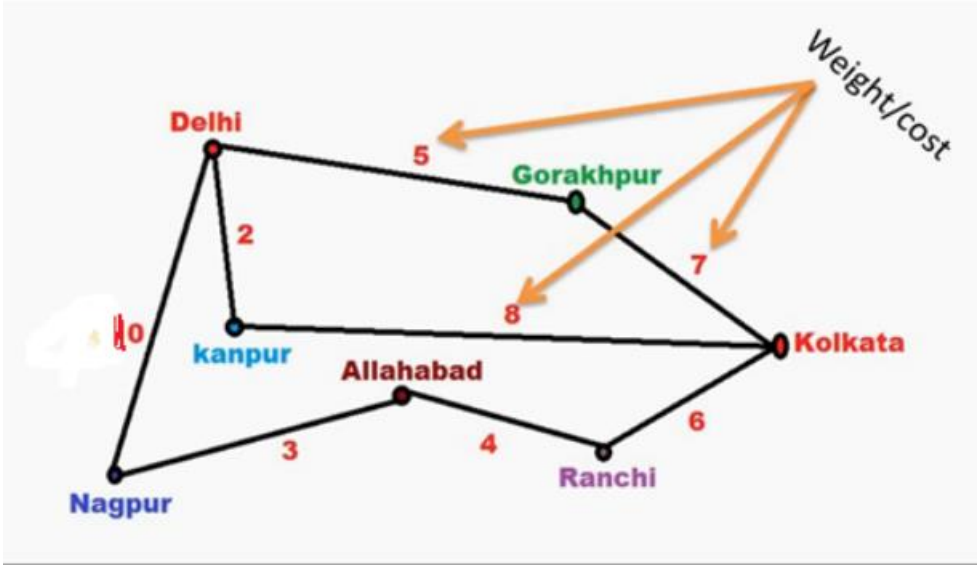
DSA LAB

Assignment 11

Name: Madhav Jha
Roll no.: E3-48
Branch: CSE (AI & ML)

Write a program to find shortest path from Delhi to Kolkata using Dijkstra and Floyd Warshall Algorithm.

Graph:



Code:

```c
#include <stdio.h>
#include <stdlib.h>
#define MAXNUM 20000

int edges[1000][1000];
int apspArr[1000][1000];
int arr[1000][1000];
char nodes[1000][1000];

void edgesPrint(int n) {
    printf("\n\n  | ");
    for (int i = 0;i < n;++i) {
        printf(" %d ", i);
    }
    printf("\n--|");
    for (int i = 0;i < n;++i) {
        printf("---");
    }
```

```
    for (int i = 0;i < n;i++) {
        printf("\n%d | ", i);
        for (int j = 0;j < n;++j) {
            if (edges[i][j] == MAXNUM) {
                printf(" 0 ");
                continue;
            }
            if (edges[i][j] < 9) {
                printf(" ");
            }
            printf("%d ", edges[i][j]);
        }
    }
    printf("\n");
}

void apspPrint(int n) {
    printf("\n\n  | ");
    for (int i = 0;i < n;++i) {
        printf(" %d ", i);
    }
    printf("\n--|");
    for (int i = 0;i < n;++i) {
        printf("---");
    }
    for (int i = 0;i < n;i++) {
        printf("\n%d | ", i);
        for (int j = 0;j < n;++j) {
            if (apspArr[i][j] == MAXNUM) {
                printf(" 0 ");
                continue;
            }
            if (apspArr[i][j] < 9) {
                printf(" ");
            }
            printf("%d ", apspArr[i][j]);
        }
    }
    printf("\n");
}
```

```
int min(int a, int b) {
    return (a < b) ? a : b;
}

void apsp(int n) {
    for (int i = 0;i < n;i++) {
        for (int j = 0;j < n;j++) {
            apspArr[i][j] = edges[i][j];
        }
    }
    for (int i = 0;i < n;++i) {
        for (int j = 0;j < n;++j) {
            for (int k = 0;k < n;++k) {
                if (j == i || k == i || j == k) continue;
                apspArr[j][k] = min(apspArr[j][k], apspArr[j][i] +
apspArr[i][k]);
            }
        }
    }
}

int dijkstraAlgo(int a, int b, int n) {
    for (int i = 0;i < n;i++) {
        for (int j = 0;j < n;j++) {
            arr[i][j] = MAXNUM;
        }
    }
    int isVisited[n];
    for (int i = 0;i < n;i++) {
        isVisited[i] = 0;
    }

    for (int i = 0;i < n;i++) {
        if (edges[a][i] != MAXNUM) {
            arr[a][i] = edges[a][i];
        }
    }
    isVisited[a] = 1;
```

```
    int prev = a, count = n - 1;
    while (count > 0) {
        int mn = MAXNUM, k = -1;
        for (int j = 0;j < n;j++) {

            if (arr[prev][j] < mn && isVisited[j] == 0) {
                mn = arr[prev][j];
                k = j;
            }
        }

        if (mn == MAXNUM) continue;

        for (int j = 0;j < n;j++) {
            arr[k][j] = min(arr[prev][j], arr[prev][k] + edges[k][j]);
        }
        isVisited[k] = 1;
        prev = k;
        count--;
    }

    int res = MAXNUM;
    for (int i = 0;i < n;i++) {
        res = min(res, arr[i][b]);
    }
    return res;
}

int main(void) {

// #ifndef ONLINE_JUDGE
//     freopen("input.txt", "r", stdin);
// #endif

    printf("\n\n:::::All Pair Shortest Path:::::\n\n");

    int n = 0;
    printf("Enter number of nodes: ");
    scanf("%d", &n);
```

```
    printf("\nEnter node/city values/name:\n");
    for (int i = 0;i < n;++i) {
        printf("Name of node %d: ", i);
        scanf("%s", &nodes[i]);
    }
    for (int i = 0;i < n;++i) {
        printf("\nNode %d: %s", i, nodes[i]);
    }printf("\n");

    // put all edges as 0
    for (int i = 0;i < n;i++) {
        for (int j = 0;j < n;++j) {
            edges[i][j] = MAXNUM;
        }
    }

    int e = 0;
    printf("\nEnter number of edges: ");
    scanf("%d", &e);

    if (e < n - 1) {
        printf("\n\nERROR: Edges should be >= %d\n\n", n - 1);
        return 0;
    }

    printf("\nEnter edges in following format: from to weight \n(e.g.:
2 4 3)\n");
    for (int i = 0;i < e;i++) {
        int a, b, c;
        printf("Edge: ");
        scanf("%d", &a);
        scanf("%d", &b);
        scanf("%d", &c);
        edges[a][b] = c;
        edges[b][a] = c;
    }

    printf("\n\nInitial edge matrix:");
    edgesPrint(n);
    apsp(n);
```

```
    int inp = 0;
    do {
        printf("\n\n0. Exit!!!");
        printf("\n1. Find shortest distance between two nodes using
dijkstra algorithm.");
        printf("\n2. Find shortest distance between two nodes using
Floyd Warshall algorithm.");
        printf("\n3. Find shortest distance between all nodes.");
        printf("\nYour Choice: ");
        scanf("%d", &inp);

        int a, b, val;
        switch (inp)
        {
        case 1:
            printf("\nEnter current node number: ");
            scanf("%d", &a);
            printf("Enter destination node number: ");
            scanf("%d", &b);

            val = dijkstraAlgo(a, b, n);
            printf("\nShortest path distance between %s and %s is:
%d", nodes[a], nodes[b], val);
            break;
        case 2:
            printf("\nEnter current node number: ");
            scanf("%d", &a);
            printf("Enter destination node number: ");
            scanf("%d", &b);

            val = apspArr[a][b];
            printf("\nShortest path distance between %s and %s is:
%d", nodes[a], nodes[b], val);
            break;
        case 3:
            printf("\n\nAll Pair Shortest Distance: ");
            apspPrint(n);
            break;
        case 0:
```

```
            printf("\nExit...!");
            break;
        default:
            printf("\nERROR: Invalid option!!!\n");
            break;
        }

    } while (inp != 0);

    printf("\n\n");
    return 0;
}
```

Output:

```
PS E:\Google Drive\Classroom\SEM-3\DSA_lab-sem3> cd "e:\Google
Drive\Classroom\SEM-3\DSA_lab-sem3\lab-11_(graph-traversal)\"
; if ($?) { gcc graphTraversal.c -o graphTraversal } ; if ($?)
{ .\graphTraversal }

:::::All Pair Shortest Path:::::

Enter number of nodes: 7

Enter node/city values/name:
Name of node 0: Delhi
Name of node 1: Gorakhpur
Name of node 2: Kolkata
Name of node 3: Ranchi
Name of node 4: Allahabad
Name of node 5: Nagpur
Name of node 6: Kanpur

Node 0: Delhi
Node 1: Gorakhpur
Node 2: Kolkata
Node 3: Ranchi
Node 4: Allahabad
Node 5: Nagpur
Node 6: Kanpur

Enter number of edges: 8

Enter edges in following format: from to weight
(e.g.: 2 4 3)
Edge: 0 1 5
Edge: 1 2 7
Edge: 2 3 6
Edge: 3 4 4
Edge: 4 5 3
Edge: 5 0 10
Edge: 0 6 2
Edge: 6 2 8
```

```
Initial edge matrix:

   |  0  1  2  3  4  5  6
---|--------------------
0  |  0  5  0  0  0 10  2
1  |  5  0  7  0  0  0  0
2  |  0  7  0  6  0  0  8
3  |  0  0  6  0  4  0  0
4  |  0  0  0  4  0  3  0
5  | 10  0  0  0  3  0  0
6  |  2  0  8  0  0  0  0


0. Exit!!!
1. Find shortest distance between two nodes using dijkstra
algorithm.
2. Find shortest distance between two nodes using Floyd
Warshall algorithm.
3. Find shortest distance between all nodes.
Your Choice: 1

Enter current node number: 0
Enter destination node number: 2

Shortest path distance between Delhi and Kolkata is: 10

0. Exit!!!
1. Find shortest distance between two nodes using dijkstra
algorithm.
2. Find shortest distance between two nodes using Floyd
Warshall algorithm.
3. Find shortest distance between all nodes.
Your Choice: 2

Enter current node number: 0
Enter destination node number: 2

Shortest path distance between Delhi and Kolkata is: 10
```

```
0. Exit!!!
1. Find shortest distance between two nodes using dijkstra
algorithm.
2. Find shortest distance between two nodes using Floyd
Warshall algorithm.
3. Find shortest distance between all nodes.
Your Choice: 3


All Pair Shortest Distance:

   |  0  1  2  3  4  5  6
---|--------------------
0  |  0  5 10 16 13 10  2
1  |  5  0  7 13 17 15  7
2  | 10  7  0  6 10 13  8
3  | 16 13  6  0  4  7 14
4  | 13 17 10  4  0  3 15
5  | 10 15 13  7  3  0 12
6  |  2  7  8 14 15 12  0


0. Exit!!!
1. Find shortest distance between two nodes using dijkstra
algorithm.
2. Find shortest distance between two nodes using Floyd
Warshall algorithm.
3. Find shortest distance between all nodes.
Your Choice: 0

Exit...!

PS E:\Google Drive\Classroom\SEM-3\DSA_lab-sem3\lab-11_(graph-
traversal)>
```