## CHANDIGARH UNIVERSITY

## UNIVERSITY INSTITUTE OF NGINEERING

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



|  |  |
| --- | --- |
| **Submitted By: Submitted To:**  Vivek Kumar(21BCS8129) Neha Dutta(E12830) | |
| **Subject Name** | Design and Analysis of Algorithm Lab |
| **Subject Code** | 20CSP-312 |
| **Branch** | Computer Science and Engineering |
| **Semester** | 5th |

**Experiment - 9**

**Student Name: Vivek Kumar UID: 21BCS8129**

**Branch: BE-CSE(LEET) Section/Group: 20BCS-WM-616/A**

**Semester: 5th Date of Performance: 07/11/2022**

**Subject Name: DAA Lab Subject Code: 20CSP-312**

1. **Aim/Overview of the practical:**

Code and analyze to find the shortest paths in a graph with positive edge weights using Dijkstra’s algorithm.

**2. Task to be done/ Which logistics used:**

Code and analyze to find the shortest paths in a graph with positive edge weights using Dijkstra’s algorithm.

**3. Requirements (For programming-based labs):**

* Laptop or PC.
* Operation system (Mac, Windows, Linux, or any)
* Vs-Code with MinGw or any C++ Compiler

**4. Steps for experiment/practical/Code:**

#include <iostream>

#include <stdio.h>

using namespace std;

#define INFINITY 9999

#define max 5

void dijkstra(int G[max][max], int n, int startnode);

int main()

{

    int G[max][max] = {{0, 1, 0, 3, 10}, {1, 0, 5, 0, 0}, {0, 5, 0, 2, 1}, {3, 0, 2, 0, 6}, {10, 0, 1, 6, 0}};

    int n = 5;

    int u = 0;

    dijkstra(G, n, u);

    return 0;

}

void dijkstra(int G[max][max], int n, int startnode)

{

    int cost[max][max], distance[max], pred[max];

    int visited[max], count, mindistance, nextnode, i, j;

    for (i = 0; i < n; i++)

        for (j = 0; j < n; j++)

            if (G[i][j] == 0)

                cost[i][j] = INFINITY;

            else

                cost[i][j] = G[i][j];

    for (i = 0; i < n; i++)

    {

        distance[i] = cost[startnode][i];

        pred[i] = startnode;

        visited[i] = 0;

    }

    distance[startnode] = 0;

    visited[startnode] = 1;

    count = 1;

    while (count < n - 1)

    {

        mindistance = INFINITY;

        for (i = 0; i < n; i++)

            if (distance[i] < mindistance && !visited[i])

            {

                mindistance = distance[i];

                nextnode = i;

            }

        visited[nextnode] = 1;

        for (i = 0; i < n; i++)

            if (!visited[i])

                if (mindistance + cost[nextnode][i] < distance[i])

                {

                    distance[i] = mindistance + cost[nextnode][i];

                    pred[i] = nextnode;

                }

        count++;

    }

    for (i = 0; i < n; i++)

        if (i != startnode)

        {

            cout <<endl<< "Distance of node "<<i<<" = "<<distance[i];

            cout << " Path = "<<i;

            j = i;

            do

            {

                j = pred[j];

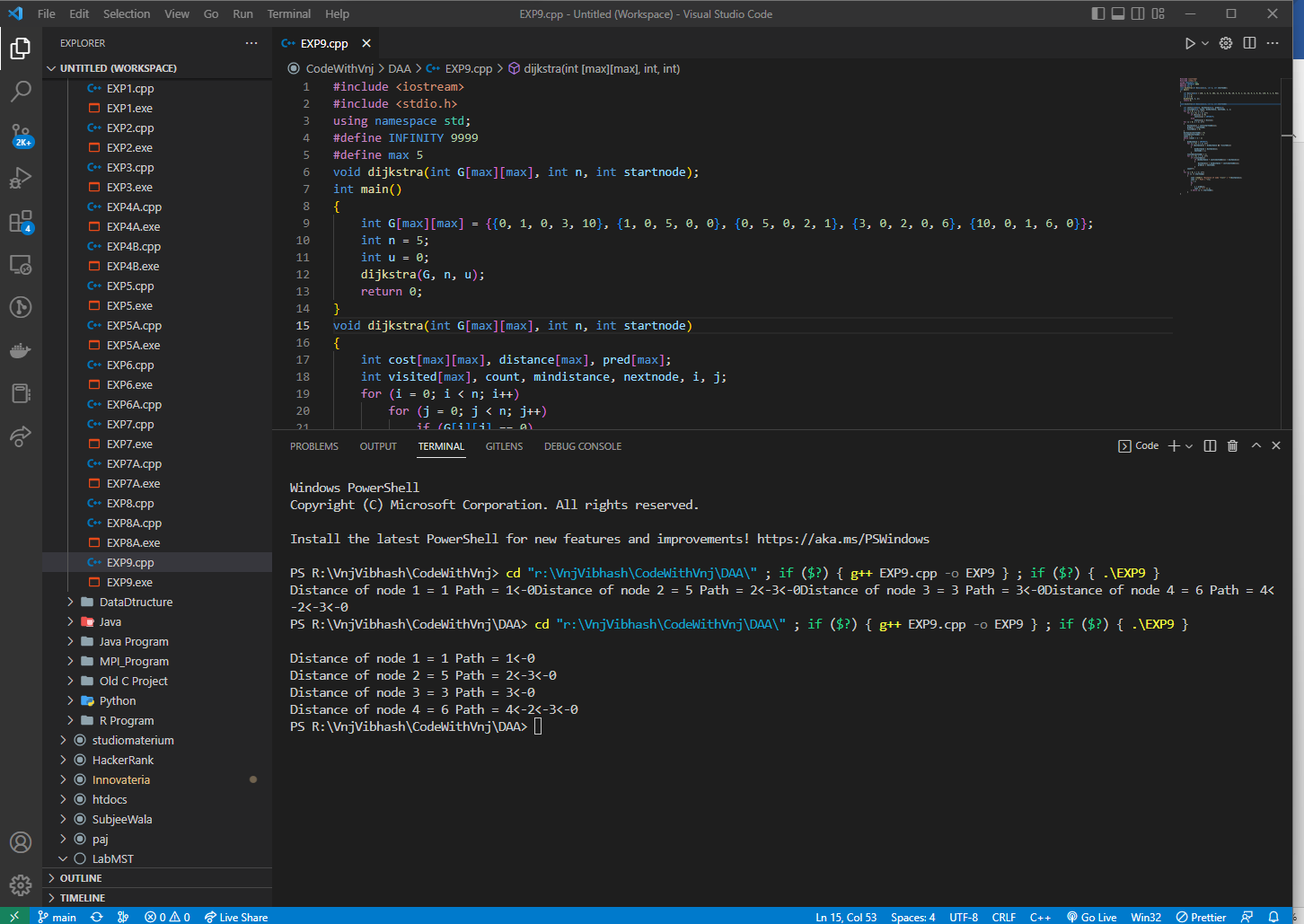
                cout << "<-" << j;

            } while (j != startnode);

        }

}

**5. Output:**

****

**Learning outcomes (What I have learnt):**

1. How to solve the Djkstra algorithm using dynamic programming.

**Evaluation Grid (To be created per the faculty's SOP and Assessment guidelines):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. | Worksheet completion including writing learning objectives/Outcomes.  (To be submitted at the end of the day). |  |  |
| 2. | Post-Lab Quiz Result. |  |  |
| 3. | Student Engagement in  Simulation/Demonstration/Performance and Controls/Pre-Lab Questions. |  |  |
|  | Signature of Faculty (with Date): | Total Marks Obtained: |  |