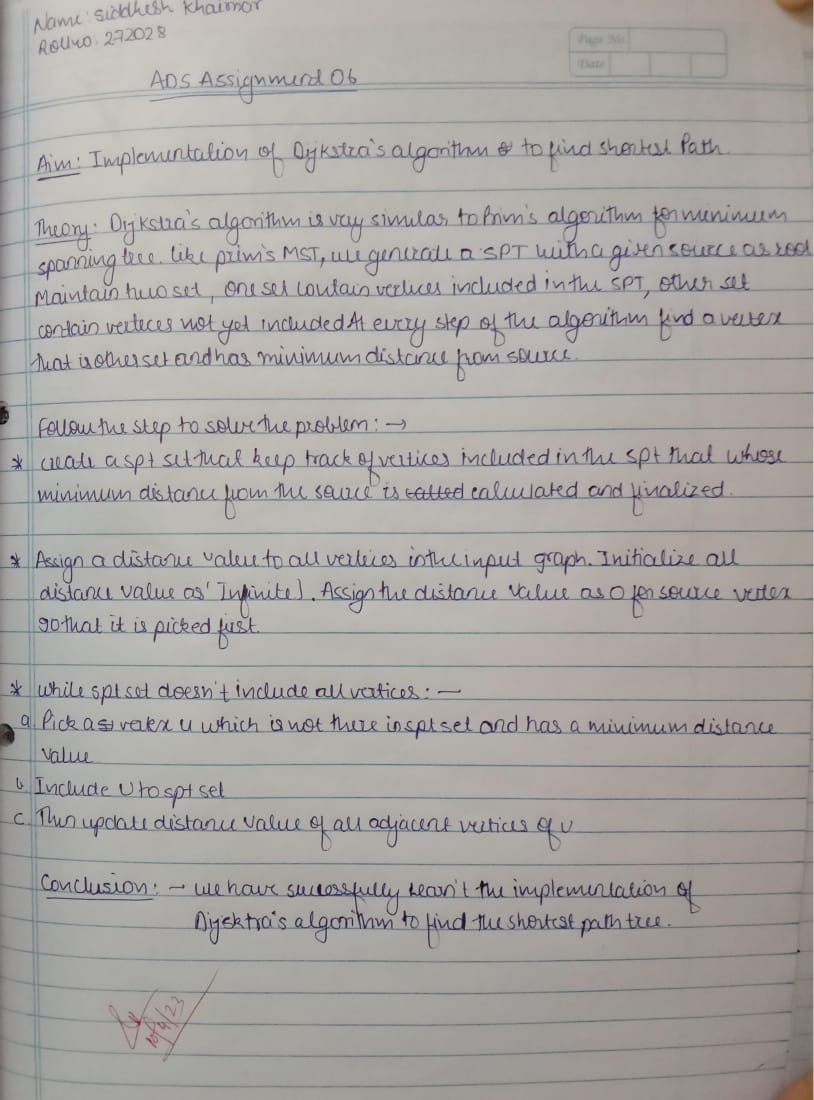
|  |  |  |  |
| --- | --- | --- | --- |
|  | Bansilal Ramnath Agarwal Charitable Trust's  Vishwakarma Institute of Information Technology  **Department of**  **Artificial Intelligence and Data Science** | | |
| Name: Siddhesh Dilip Khairnar | | | |
| Class: SY | Division: B | | Roll No: 272028 |
| Semester: IV | | Academic Year: 2022-2023 | |
| Subject Name & Code: Advance Data Structure: ADUA22202 | | | |
| Title of Assignment: For any application find Single source shortest path using Dijkstra’s algorithm | | | |
| Date of Performance: 01-03-2023 | | Date of Submission: 08-03-2023 | |

**ASSIGNMENT NO. 6**



**Program and Output:**

#include<iostream>

using namespace std;

// Number of vertices in the graph

const int V=7;

// Function to find the vertex with minimum key value

int min\_distance(int key[], bool visited[])

{

    int min = 999, min\_index;  // 999 represents an Infinite value

    for (int v = 0; v < V; v++) {

        if (visited[v] == false && key[v] < min) {

            // vertex should not be visited

            min = key[v];

            min\_index = v;

        }

    }

    return min\_index;

}

// Function to print the final MST stored in parent[]

void print\_MST(int parent[], int distance[V])

{

    int minCost=0;

    cout<<"Edge \tWeight\n";

    for (int i = 1; i< V; i++) {

        cout<<"0"<<" - "<<i<<" \t"<<distance[i]<<" \n";

        minCost+=distance[i];

    }

    cout<<"Total cost is"<<minCost;

}

// Function to find the MST using adjacency cost matrix representation

void find\_MST(int cost[V][V])

{

    int parent[V], distance[V];

    bool visited[V];

    // Initialize all the arrays

    for (int i = 0; i< V; i++) {

        distance[i] = 999;    // 999 represents an Infinite value

        visited[i] = false;

        parent[i]=-1;

    }

    distance[0] = 0;

    parent[0] = -1;

    for (int x = 0; x < V - 1; x++)

    {

        int u = min\_distance(distance, visited);

        visited[u] = true;

        for (int v = 0; v < V; v++)

        {

            int total\_distance=distance[u]+cost[u][v];

            if (cost[u][v]!=0 && visited[v] == false && total\_distance< distance[v])

            {

                parent[v] = u;

                distance[v] = total\_distance;

            }

        }

    }

    // print the final MST

    print\_MST(parent, distance);

}

// main function

int main()

{

    int cost[V][V];

    cout<<"Enter the weigth matrix for a graph with 6 vetices";

    for (int i=0;i<V;i++)

    {

        for(int j=0;j<V;j++)

        {

            cout<<"\n"<<"enter cost from vertex["<<i<<"]"<<"["<<j<<"] :";

            cin>>cost[i][j];

        }

    }

    find\_MST(cost);

    return 0;

}

