**Write a c program to find single source shortest path using Dijkstra’s algorithm**

**Dijkstra’s Algorithm for single source shortest path**

#include <stdio.h>

#include <conio.h>

#define INFINITY 9999

#define MAX 10

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

int main()

{

    int G[MAX][MAX], i, j, n, u;

    printf("Enter no. of vertices:");

    scanf("%d", &n);

    printf("\nEnter the adjacency matrix:\n");

    for (i = 0; i < n; i++)

        for (j = 0; j < n; j++)

            scanf("%d", &G[i][j]);

    printf("\nEnter the starting node:");

    scanf("%d", &u);

    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;

    // pred[] stores the predecessor of each node

    // count gives the number of nodes seen so far

    // create the cost matrix

    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];

    // initialize pred[],distance[] and visited[]

    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;

        // nextnode gives the node at minimum distance

        for (i = 0; i < n; i++)

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

            {

                mindistance = distance[i];

                nextnode = i;

            }

        // check if a better path exists through nextnode

        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++;

    }

    // print the path and distance of each node

    for (i = 0; i < n; i++)

        if (i != startnode)

        {

            printf("\nDistance of node%d=%d", i, distance[i]);

            printf("\nPath=%d", i);

            j = i;

            do

            {

                j = pred[j];

                printf("<-%d", j);

            } while (j != startnode);

        }

}

OUTPUT

Enter no. of vertices:4

Enter the adjacency matrix:

2 3 4 1

4 2 9 6

2 3 888 888

4 888 1 9

Enter the starting node:2

Distance of node0=2

Path=0<-2

Distance of node1=3

Path=1<-2

Distance of node3=3

Path=3<-0<-2

Write a C program to solve Travelling Salesman problem using dynamic method

#include <stdio.h>

int tsp\_g[10][10] = {

    {12, 30, 33, 10, 45},

    {56, 22, 9, 15, 18},

    {29, 13, 8, 5, 12},

    {33, 28, 16, 10, 3},

    {1, 4, 30, 24, 20}};

int visited[10], n, cost = 0;

/\* creating a function to generate the shortest path \*/

void travellingsalesman(int c)

{

    int k, adj\_vertex = 999;

    int min = 999;

    /\* marking the vertices visited in an assigned array \*/

    visited[c] = 1;

    /\* displaying the shortest path \*/

    printf("%d ", c + 1);

    /\* checking the minimum cost edge in the graph \*/

    for (k = 0; k < n; k++)

    {

        if ((tsp\_g[c][k] != 0) && (visited[k] == 0))

        {

            if (tsp\_g[c][k] < min)

            {

                min = tsp\_g[c][k];

            }

            adj\_vertex = k;

        }

    }

    if (min != 999)

    {

        cost = cost + min;

    }

    if (adj\_vertex == 999)

    {

        adj\_vertex = 0;

        printf("%d", adj\_vertex + 1);

        cost = cost + tsp\_g[c][adj\_vertex];

        return;

    }

    travellingsalesman(adj\_vertex);

}

/\* main function \*/

int main()

{

    int i, j;

    n = 5;

    for (i = 0; i < n; i++)

    {

        visited[i] = 0;

    }

    printf("\n\nShortest Path:\t");

    travellingsalesman(0);

    printf("\n\nMinimum Cost: \t");

    printf("%d\n", cost);

    return 0;

}

OUTPUT

Shortest Path: 1 5 4 3 2 1

Minimum Cost: 99