**Aim: Implementation of Multistage Graphs.**

**Theory:**

A multistage graph **G = (V, E)** is a directed graph where vertices are partitioned into **k** (where ***k* > 1**) number of disjoint subsets ***S = {s1,s2,…,sk}*** such that edge *(u, v)* is in E, then *u Є si* and *v Є s1 + 1* for some subsets in the partition and |***s1***| = |***sk***| = 1.

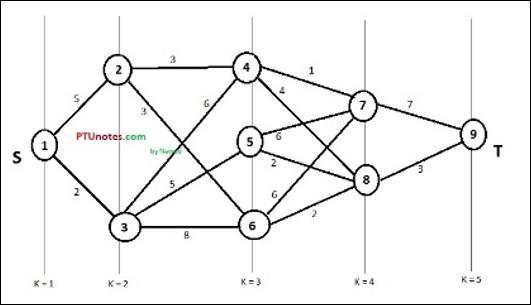
The vertex ***s Є s1*** is called the **source** and the vertex ***t Є sk*** is called **sink**.

***G*** is usually assumed to be a weighted graph. In this graph, cost of an edge *(i, j)* is represented by *c(i, j)*. Hence, the cost of path from source ***s*** to sink ***t*** is the sum of costs of each edges in this path.

The multistage graph problem is finding the path with minimum cost from source ***s*** to sink ***t***.

Example

Consider the following example to understand the concept of multistage graph.



According to the formula, we have to calculate the cost **(i, j)** using the following steps

Step-1: Cost (K-2, j)

In this step, three nodes (node 4, 5. 6) are selected as **j**. Hence, we have three options to choose the minimum cost at this step.

*Cost(3, 4) = min {c(4, 7) + Cost(7, 9),c(4, 8) + Cost(8, 9)} = 7*

*Cost(3, 5) = min {c(5, 7) + Cost(7, 9),c(5, 8) + Cost(8, 9)} = 5*

*Cost(3, 6) = min {c(6, 7) + Cost(7, 9),c(6, 8) + Cost(8, 9)} = 5*

Step-2: Cost (K-3, j)

Two nodes are selected as j because at stage k - 3 = 2 there are two nodes, 2 and 3. So, the value i = 2 and j = 2 and 3.

*Cost(2, 2) = min {c(2, 4) + Cost(4, 8) + Cost(8, 9),c(2, 6) +*

*Cost(6, 8) + Cost(8, 9)} = 8*

*Cost(2, 3) = {c(3, 4) + Cost(4, 8) + Cost(8, 9), c(3, 5) + Cost(5, 8)+ Cost(8, 9), c(3, 6) + Cost(6, 8) + Cost(8, 9)} = 10*

Step-3: Cost (K-4, j)

*Cost (1, 1) = {c(1, 2) + Cost(2, 6) + Cost(6, 8) + Cost(8, 9), c(1, 3) + Cost(3, 5) + Cost(5, 8) + Cost(8, 9))} = 12*

*c(1, 3) + Cost(3, 6) + Cost(6, 8 + Cost(8, 9))} = 13*

Hence, the path having the minimum cost is **1→ 3→ 5→ 8→ 9**.

**Complexity Analysis:**

Top of Form

Bottom of Form

Top of Form

In a multi-stage graph algorithm for shortest path, we minimize cost for every *edge* exactly once. So the Time Complexity is O(E)O(E). However, in the worst case, we get a complete graph, which has edges E=n∗(n−1)/2E=n∗(n−1)/2​, so worst time complexity then becomes O(E)=O(n2).

Bottom of Form

**Source Code:**

#include <stdio.h>

#include <limits.h>

#define n 8

int main()

{

    int stages = 4;

    printf("Number of stages=4 \n");

    // cost adjacency matrix

    int graph[n][n] =

        {{0, 1, 2, 5, 0, 0, 0, 0},

         {0, 0, 0, 0, 4, 11, 0, 0},

         {0, 0, 0, 0, 9, 5, 16, 0},

         {0, 0, 0, 0, 0, 0, 2, 0},

         {0, 0, 0, 0, 0, 0, 0, 18},

         {0, 0, 0, 0, 0, 0, 0, 13},

         {0, 0, 0, 0, 0, 0, 0, 2},

         {0, 0, 0, 0, 0, 0, 0, 0}};

    int distance[n];

    int path[stages];

    int cost[n];

    cost[n - 1] = 0;

    for (int i = n - 2; i >= 0; i--)

    {

        int min = INT\_MAX;

        for (int k = i + 1; k <= n - 1; k++)

        {

            if (graph[i][k] != 0 && graph[i][k] + cost[k] < min)

            {

                min = graph[i][k] + cost[k];

                distance[i] = k; //saving the vertex which gave minmum value

            }

        }

        cost[i] = min;

    }

    path[0] = 0; //first vertex will always be 1(added +1 in display)

    path[stages - 1] = n;

    for (int i = 1; i < stages - 1; i++)

    {

        path[i] = distance[path[i - 1]];

    }

    printf("Path is: ");

    for (int i = 0; i < stages; i++)

    {

        if (i == stages - 1)

        {

            printf("%d\n", path[i]); // last node

        }

        else

        {

            // adding +1 because we are starting from 0 and vertices starts from 1

            printf("%d -> ", path[i] + 1);

        }

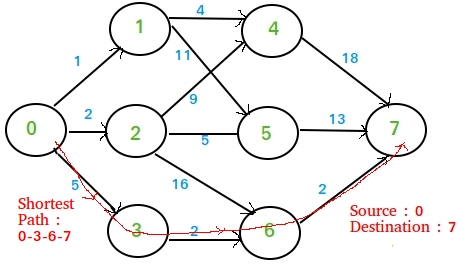
    }

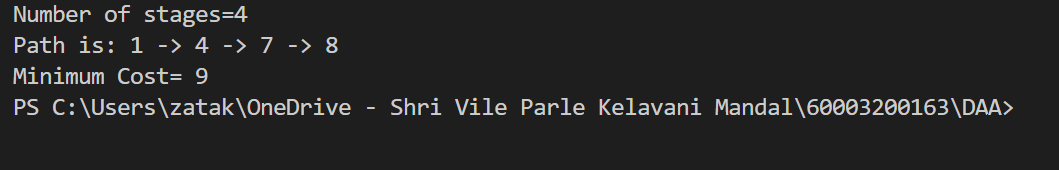
    printf("Minimum Cost= %d", cost[0]);

    return 0;

}

**Output:**



****

**Conclusion:** Thus, we have implemented multistage graph using dynamic programming and have also found optimal cost for the same. Also, the shortest path is being displayed correctly.