# TOPIC:- Programs on Prim's and Kruskal's Algorithm.

# Objective:

Implementation of prim’s and Kruskal’s algorithm

Problem Statement:

Prim's Algorithm is a greedy algorithm that is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.

In Kruskal Algorithm, initially, all the nodes of the graph are separated from each other, which means they don’t have an edge between them. Then to obtain the minimum spanning tree from that graph we first sort the edges of the graph in a non-decreasing fashion. Then we pick the edges from left to right and connect the graph. Now there are two possibilities, first one is if the current picked edge is already connected in the tree so in this case, we will just continue our process and if the current picked edge is not connected then we will just connect those two nodes with dsu (disjoint set union). In this manner, we finally conclude with the final MST of the given graph.

Prim’s Algorithm:

STEP 1 : Choose a Node and add it to MST

STEP 2 : Check all of its adjacent nodes, if they are already present in MST or not if they are not present, add edge weight to that edge into Min-Heap.

STEP 3 : Add the smallest weighted edge and its node in Min-Heap to MST. Repeat step 2 and step 3 for Node added.

STEP 4 : Repeat step 2 , step 3 for recently added node.

**Code:**

#include<stdio.h>

#include<stdbool.h>

#define INF 9999999

// number of vertices in graph

// create a 2d array of size 5x5

//for adjacency matrix to represent graph

int main() {

int V;

scanf("%d",&V);

int G[V][V];

for(int i=0;i<V;i++)

{

for(int j=0;j<V;j++)

{

scanf("%d",&G[i][j]);

}

}

int no\_edge; // number of edge

// create a array to track selected vertex

// selected will become true otherwise false

int selected[V];

// set selected false initially

memset(selected, false, sizeof(selected));

// set number of edge to 0

no\_edge = 0;

// the number of egde in minimum spanning tree will be

// always less than (V -1), where V is number of vertices in

//graph

// choose 0th vertex and make it true

selected[0] = true;

int x; // row number

int y; // col number

// print for edge and weight

printf("Edge : Weight\n");

while (no\_edge < V - 1) {

//For every vertex in the set S, find the all adjacent vertices

// , calculate the distance from the vertex selected at step 1.

// if the vertex is already in the set S, discard it otherwise

//choose another vertex nearest to selected vertex at step 1.

int min = INF;

x = 0;

y = 0;

for (int i = 0; i < V; i++) {

if (selected[i]) {

for (int j = 0; j < V; j++) {

if (!selected[j] && G[i][j]) { // not in selected and there is an edge

if (min > G[i][j]) {

min = G[i][j];

x = i;

y = j;

}

}

}

}

}

printf("%d - %d : %d\n", x, y, G[x][y]);

selected[y] = true;

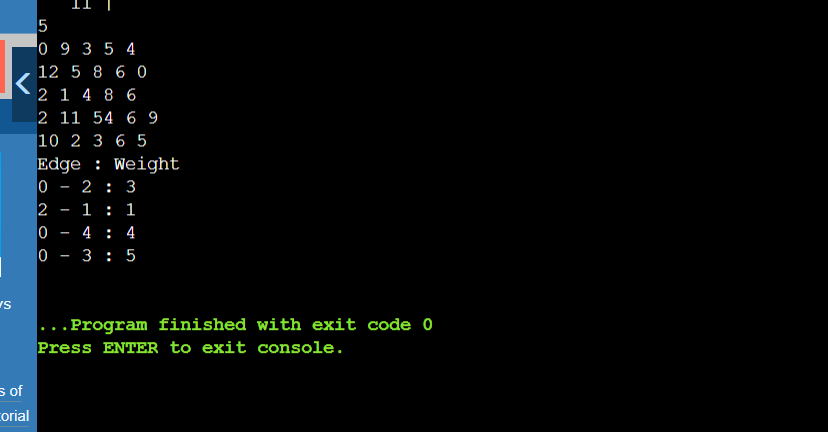
no\_edge++;

}

  return 0;

}

**OUTPUT**:



Kruskal’s Algorithm:

STEP 1: Sort the given edges

STEP 2: Check each edge in sorted order if it forms a cycle with already selected edges. If not Add it to list of MST . If it does move to next edge.

STEP 3: Run steps 1 and 2 till v-1 edges are selected (v=no.of.vertices).

**Code:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int i,j,k,a,b,u,v,n,ne=1;

int min,mincost=0,cost[9][9],parent[9];

int find(int);

int uni(int,int);

void main()

{

printf("\n\tImplementation of Kruskal's algorithm\n");

printf("\nEnter the no. of vertices:");

scanf("%d",&n);

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

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

scanf("%d",&cost[i][j]);

if(cost[i][j]==0)

cost[i][j]=999;

}

}

printf("The edges of Minimum Cost Spanning Tree are\n");

while(ne < n)

{

for(i=1,min=999;i<=n;i++)

{

for(j=1;j <= n;j++)

{

if(cost[i][j] < min)

{

min=cost[i][j];

a=u=i;

b=v=j;

}

}

}

u=find(u);

v=find(v);

if(uni(u,v))

{

printf("%d edge (%d,%d) =%d\n",ne++,a,b,min);

mincost +=min;

}

cost[a][b]=cost[b][a]=999;

}

printf("\n\tMinimum cost = %d\n",mincost);

getch();

}

int find(int i)

{

while(parent[i])

i=parent[i];

return i;

}

int uni(int i,int j)

{

if(i!=j)

{

parent[j]=i;

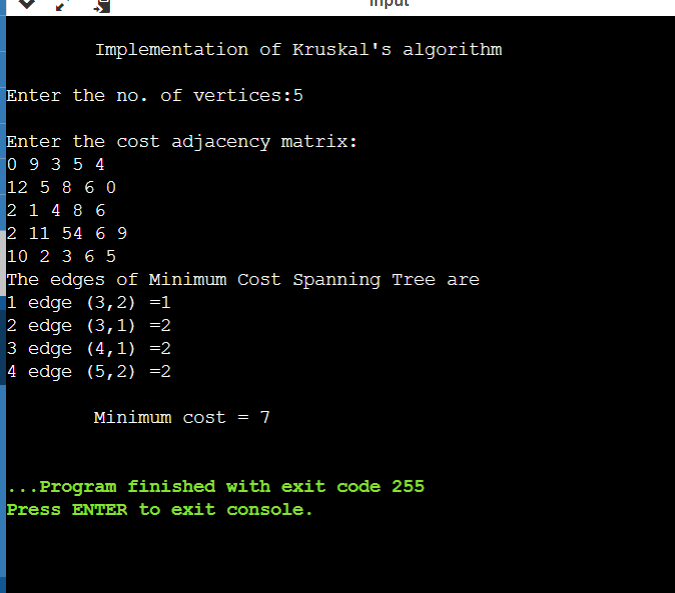
return 1;

}

return 0;

}

**OUTPUT:**

****

The number of vertices is taken as input which is 5 and the adjacency matrix in the form of the 2D array is entered by the user which indicates the weights from edge to edge. The edges are sorted based on the cost of the edges. The starting vertex is the vertex of the edge with minimum weight. The edges are traversed in this way and give the minimum cost. Here the edges and corresponding weights are printed, and the minimum cost obtained is 7.

**Problems Faced:**

In this experiment, I didn’t face any problem. It is quite easy topic.

**Conclusion:**

With the help of this assignment, I understood the concept of prims and Kruskal’s algorithm.