AIM: Implement broadcast tree for a given subnet of hosts

A **minimum spanning tree** (MST) or minimum weight spanning tree for a weighted, connected and undirected graph is a spanning tree with weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.

A minimum spanning tree has (V – 1) edges where V is the number of vertices in the given graph.

The steps for finding MST using Kruskal’s algorithm

1. Sort all the edges in non-decreasing order of their weight.
2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so If cycle is not formed, include this edge. Else, discard it.
3. Repeat step#2 until there are (V-1) edges in the spanning

The steps for finding MST using Kruskal’s algorithm

* Sort all the edges in non-decreasing order of their weight.
* Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so If cycle is not formed, include this edge. Else, discard it.
* Repeat step#2 until there are (V-1) edges in the spanning tree

#include <stdio.h>

#define MAX 20

int parent[MAX];

// Find operation for union-find

int find(int i) {

while (parent[i] != i)

i = parent[i];

return i;

}

// Union operation

void union\_sets(int i, int j) {

int a = find(i);

int b = find(j);

parent[a] = b;

}

int main() {

int n, i, j, u, v, a, b;

int cost[MAX][MAX];

int min, mincost = 0, edges = 1;

printf("Enter the number of hosts (nodes): ");

scanf("%d", &n);

printf("Enter the cost adjacency matrix (use 999 for no link):\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; // avoid self-loop

}

}

}

// Initialize parent array

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

parent[i] = i;

}

printf("\nEdges in the Broadcast Tree (MST):\n");

while (edges < n) {

min = 999;

// Find minimum edge

for (i = 1; 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 including this edge doesn't form a cycle

if (u != v) {

printf("Edge %d: (%d -- %d) cost: %d\n", edges++, a, b, min);

mincost += min;

union\_sets(u, v);

}

cost[a][b] = cost[b][a] = 999; // remove edge

}

printf("\nMinimum Cost of Broadcast Tree = %d\n", mincost);

return 0;

}

**OUTPUT:**

Enter the number of hosts (nodes): 4

Enter the cost adjacency matrix (use 999 for no link):

0 5 999 999

5 0 10 999

999 10 0 8

999 999 8 0

Edges in the Broadcast Tree (MST):

Edge 1: (1 -- 2) cost: 5

Edge 2: (3 -- 4) cost: 8

Edge 3: (2 -- 3) cost: 10

Minimum Cost of Broadcast Tree = 23