1. **MST using greedy techniques**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <limits.h>

#define V 100 // Maximum number of vertices

int minKey(int key[], bool mstSet[]) {

int min = INT\_MAX, min\_index;

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

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

min = key[v], min\_index = v;

return min\_index;

}

void printMST(int parent[], int graph[V][V], int n) {

printf("Edge \tWeight\n");

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

printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);

}

void primMST(int graph[V][V], int n) {

int parent[V]; // Array to store constructed MST

int key[V]; // Key values used to pick the minimum weight edge in cut

bool mstSet[V]; // To represent set of vertices not yet included in MST

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

key[i] = INT\_MAX, mstSet[i] = false;

key[0] = 0; // Make key 0 so that this vertex is picked as first vertex

parent[0] = -1; // First node is always root of MST

for (int count = 0; count < n - 1; count++) {

int u = minKey(key, mstSet);

mstSet[u] = true;

for (int v = 0; v < n; v++)

if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v])

parent[v] = u, key[v] = graph[u][v];

}

printMST(parent, graph, n);

}

int main() {

int n; // Number of vertices

printf("Enter the number of vertices (maximum %d): ", V);

scanf("%d", &n);

int graph[V][V]; // Adjacency matrix representing the graph

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

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

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

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

}

}

// Print the MST

printf("Minimum Spanning Tree:\n");

primMST(graph, n);

return 0;

}

**Output:**

Enter the number of vertices (maximum 100): 5

Enter the adjacency matrix:

0 2 0 6 0

2 0 3 8 5

0 3 0 0 7

6 8 0 0 9

0 5 7 9 0

Minimum Spanning Tree:

Edge Weight

0 - 1 2

1 - 2 3

0 - 3 6

1 - 4 5

--------------------------------

Process exited after 34.97 seconds with return value 0

Press any key to continue . . .

