**Prim's algorithm**

**Design and implement C/C++ Program to find Minimum Cost Spanning Tree of a given connected undirected graph using Prim's algorithm.**

#include <stdio.h>

#include <limits.h>

#define MAX 20

// Function to find the vertex with the minimum key value

int minKey(int key[], int mstSet[], int vertices)

{

int min = INT\_MAX, minIndex;

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

{

if (!mstSet[v] && key[v] < min) {

min = key[v];

minIndex = v;

}

}

return minIndex;

}

// Function to print the constructed MST stored in parent[]

void printMST(int parent[], int graph[MAX][MAX], int vertices)

{

printf("Edge \tWeight\n");

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

{

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

}

}

// Function to implement Prim's algorithm for a given graph

void primMST(int graph[MAX][MAX], int vertices)

{

int parent[MAX]; // Array to store the constructed MST

int key[MAX]; // Key values used to pick the minimum weight edge

int mstSet[MAX]; // To represent set of vertices included in MST

// Initialize all keys as INFINITE and mstSet[] as false

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

{

key[i] = INT\_MAX;

mstSet[i] = 0;

}

// Always include the first vertex in the MST

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

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

// The MST will have vertices-1 edges

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

{

// Pick the minimum key vertex from the set of vertices not yet included in the MST

int u = minKey(key, mstSet, vertices);

// Add the picked vertex to the MST Set

mstSet[u] = 1;

// Update key value and parent index of the adjacent vertices

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

{

// graph[u][v] is non-zero only for adjacent vertices of m

// mstSet[v] is false for vertices not yet included in MST

// Update the key only if the graph[u][v] is smaller than the key[v]

if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {

parent[v] = u;

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

}

}

}

// Print the constructed MST

printMST(parent, graph, vertices);

}

int main()

{

int vertices;

// Input the number of vertices

printf("Input the number of vertices: ");

scanf("%d", &vertices);

int graph[MAX][MAX];

// Input the adjacency matrix representing the graph

printf("Input the adjacency matrix for the graph:\n");

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

{

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

{

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

}

}

// Perform Prim's algorithm to find the MST

primMST(graph, vertices);

return 0;

}

**Output:**

Input the number of vertices: 5

Input the adjacency matrix for the graph:

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

Edge Weight

0 - 1 2

1 - 2 3

0 - 3 6

1 - 4 5