CHEAPEST WAY TO FIX A COMPUTER NETWORK USING KRUSKAL AND PRIMS ALGORITHM

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfilment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

Prakriti Yadav | AP22110011514

Sagar Husen | AP22110011517

Sarbesh Yadav | AP22110011510

Neshab Alam Ansari | AP22110011516

Anand Kumar Pandit | AP22110011513

Under the Guidance of

Dr.Elakkiya E

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

Certificate

This is to certify that the work present in this Project entitled "CHEAPEST WAY TO FIX A COMPUTER NETWORK USING KRUSKAL AND PRIMS ALGORITHM " has been carried out by all the members under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in School of Engineering and Sciences.

Supervisor

(Signature)
Prof. / Dr. [Elakkiya E]
Designation,
Affiliation.

Table of Contents

1.AIM:

The aim of this project is to determine the most cost-effective method for repairing or optimizing a computer network. This is achieved by employing two well-known algorithms, Kruskal's Algorithm and Prim's Algorithm, to identify the Minimum Spanning Tree (MST) within the network. The Minimum Spanning Tree represents the subset of connections that efficiently connects all network nodes while minimizing the overall cost or distance

2.**Project Description**: Finding the Cheapest Way to Fix a Computer Network

Objective:

The primary objective of this project is to develop a solution for efficiently repairing a computer network by determining the most cost-effective way to establish or reestablish connections between network components. This is achieved through the implementation of two widely used algorithms: Prim's Algorithm and Kruskal's Algorithm.

Key Components:

Graph Representation:

The computer network is represented as a graph using an adjacency matrix, where nodes represent network components, edges signify potential connections, and edge weights correspond to the repair or optimization costs associated with each connection.

Prim's Algorithm:

Prim's Algorithm is employed to systematically construct the Minimum Spanning Tree (MST) of the network.

Initialization involves setting key values to infinity, marking all vertices as not included in the MST, and initializing the parent array.

The algorithm iteratively selects the vertex with the minimum key value that is not yet included in the MST, adding it to the MST set and updating key values and parent indices for adjacent vertices.

Kruskal's Algorithm:

Kruskal's Algorithm is utilized to find the Minimum Spanning Tree by selecting edges in ascending order of their associated costs.

The algorithm maintains subsets of vertices, ensuring that adding an edge does not form a cycle in the MST. Edges are sorted by weight and added to the MST if they do not create a cycle.

Result Display:

The results of both algorithms are displayed, showcasing the edges of the Minimum Spanning Tree along with their corresponding weights.

This provides a clear visualization of the most cost-effective way to establish connections within the computer network.

Educational Notes:

The code includes educational notes to aid understanding, explaining that edges represent network connections, and weights indicate the associated repair or optimization costs. This documentation enhances the educational value of the project.

Project Notes:

Additional notes highlight the specific context in which the project operates, emphasizing that the network is considered as a computer network, where edges represent network connections, and weights represent the cost-effectiveness of establishing these connections.

Conclusion:

This project serves as a valuable tool for network administrators and engineers, offering a systematic approach to finding the cheapest way to fix a computer network. By utilizing Prim's and Kruskal's algorithms, the project provides practical solutions for optimizing network connectivity while minimizing repair costs. Additionally, the educational notes enhance the project's value as a learning resource for those studying graph algorithms and network optimization strategies.

3.Individual members roles and responsibility:

Prakriti yadav: (AP22110011514)

—>Introduction about the projects:

Anand Kumar Pandit: (AP22110011513)

—> Explanation about Minimum spanning Tree(MST)Print function

Neshab Alam Ansari: (AP22110011516)

—>Explanation about prims Algorithm function

Sagar Husen: (AP22110011517)

—> Explanation about Kruskal's Algorithm function

Sarbesh Yadav:(AP22110011510)

—>Explanation of main function and showing input/output

//CODE

```
#include <stdio.h>
#include <limits.h>
#define V 5
void printMST(int parent[], int graph[V][V]) {
   printf("Edge \tWeight\n");
   for (int i = 1; i < V; i++)
      printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);
}
void primMST(int graph[V][V]) {
   int parent[V];
   int key[V];
```

```
    int mstSet[V];

     for (int i = 0; i < V; i++) {
        key[i] = INT_MAX;
        mstSet[i] = 0;
        parent[i] = -1;
    }
    key[0] = 0;
    for (int count = 0; count < V - 1; count++) {
        int u = -1;
        for (int v = 0; v < V; v++) {
            if (!mstSet[v] && (u == -1 || key[v] < key[u])) {
                u = v;
            }
        }
        mstSet[u] = 1;
        for (int v = 0; v < V; v++) {
            if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph[u][v];
            }
        }
    }
    printf("Prim's Algorithm Result:\n");
    printMST(parent, graph);
}
```

```
void kruskalMST(int graph[V][V]) {
    int parent[V];
    int rank[V];
    struct Edge {
        int src, dest, weight;
    } result[V];
    for (int v = 0; v < V; v++) {
        parent[v] = v;
        rank[v] = 0;
    }
    struct Edge edges[V * V];
    int e = 0;
    for (int i = 0; i < V; i++) {
        for (int j = i + 1; j < V; j++) {
            if (graph[i][j] != 0) {
                edges[e].src = i;
                edges[e].dest = j;
                edges[e].weight = graph[i][j];
                e++;
            }
        }
    }
    for (int i = 0; i < e - 1; i++) {
        for (int j = 0; j < e - i - 1; j++) {
            if (edges[j].weight > edges[j + 1].weight) {
```

```
            struct Edge temp = edges[j];
            edges[j] = edges[j + 1];
            edges[j + 1] = temp;
        }
    }
}
int i = 0, count = 0;
while (count < V - 1 && i < e) {
    struct Edge next_edge = edges[i++];
    int x = next_edge.src;
    int y = next_edge.dest;
    int xroot = x;
    int yroot = y;
    while (parent[xroot] != xroot)
        xroot = parent[xroot];
    while (parent[yroot] != yroot)
        yroot = parent[yroot];

    if (xroot != yroot) {
        result[count++] = next_edge;
        if (rank[xroot] < rank[yroot])
            parent[xroot] = yroot;
        else if (rank[xroot] > rank[yroot])
            parent[yroot] = xroot;
        else {
            parent[yroot] = xroot;
```

```
            rank[xroot]++;
        }
    }
}

    printf("\nKruskal's Algorithm Result:\n");
    printMST(parent, graph);
}

int main() {
    int graph[V][V] = {{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}};

    primMST(graph);
    kruskalMST(graph);
    printf("-->>NOTES<<--");
    printf("\n!HERE WE HAVE CONSIDERED A **GRAPH** AS A
COMPUTER NETWORK!");
    printf("\n(1).Edges==Networks");
    printf("\n(2).weights=cheapest Way/Cost");

    return 0;
}
```

4.Code Explanation: Finding the Cheapest Way to Fix a Computer **Network**

The provided C code implements two well-known algorithms, Prim's Algorithm and Kruskal's Algorithm, to find the Minimum Spanning Tree (MST) of a computer network represented as a graph. The primary goal is to determine the most cost-effective way to establish or repair connections within the network.

Functions:

printMST:

This function prints the edges of the Minimum Spanning Tree along with their weights.

primMST:

-> Implements Prim's Algorithm to find the MST.

->Initializes key values, MST set, and parent array.

->Iteratively selects the vertex with the minimum key value, updates key values, and adds vertices to the MST set.

kruskalMST:

->Implements Kruskal's Algorithm to find the MST.

->Initializes subsets, sorts edges by weight, and processes edges to form the MST without creating cycles.

Main:

->Defines an example graph represented by an adjacency matrix.

->Calls both Prim's and Kruskal's algorithms and prints the results.

->Provides additional notes indicating that the graph represents a computer network, where edges symbolize network connections, and weights represent the cost of establishing these connections.

Project Context:

The project assumes the graph to represent a computer network, where:

->Edges signify network connections.

->Weights on edges represent the cost-effectiveness or cost of establishing connections.

Conclusion:

This C code provides a practical implementation for network administrators and engineers to find the cheapest way to fix or optimize a computer network. By employing Prim's and Kruskal's algorithms, the code facilitates efficient decision-making in establishing network connections while minimizing costs. The inclusion of informative notes enhances its educational value for those studying graph algorithms and network optimization strategies.

5.Test cases and their outputs

Inputs:

{{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}};

Outputs:

Prim's Algorithm Result:

Edge Weight

0 - 1 2

1 - 2 3

0 - 3 6

1 - 4 5

Kruskal's Algorithm Result:

Edge Weight

0 - 1 2

0 - 2 0

0 - 3 6

0 - 4 0

-->>NOTES<<--

!HERE WE HAVE CONSIDERED A *GRAPH* AS A COMPUTER NETWORK!

(1).Edges==Networks

(2).weights=cheapest Way/Cost

******THE END******