Report

Semester Project Part-2

[Design And Development of Routing Algorithms]

[Computer Networking]

[Session Fall 2024]

[66920, 68105, 67949]

[Saniya Gul, Pranjal Kumari, Maryam Ejaz]

[Sem: 2nd]

Submitted to:

[Ms. Poma Panezai]

[Department of Computer Science (CS)]

Faculty of Information and Communication Technology (FICT), BUITEMS, Quetta

Abstract

"This project is a comprehensive Routing Algorithm Simulator that utilizes Kruskal's algorithm to find the Minimum Spanning Tree (MST) of a graph. The simulator allows the user to create and modify a graph topology, select the routing source and destination nodes, and see the routing process. Highlighted features of our simulator are: Automatic generation of a Graph, Selection of Source and Destination Nodes, finding a Shortest Path, and Visualizing the Graph Traversals and Selected Paths. The simulator records every routing decision and also handles Minimum Spanning Tree Edge cases like loops and no connection (disconnected between nodes). The simulator lets users visualize and dynamically update a graph to provide an interactive surrounding to provide insight into Kruskal's algorithm in different networking conditions. This is a great tool to help know and understand Kruskal's algorithm for a variety of education and research purposes. By utilizing this simulator, users can gain a deeper understanding of graph theory and its applications in real-world networking scenarios."

Table of Contents

1. Problem Statement

As computing networks increase in both size and complexity, the need for efficient and cost-effective paths for data transmission takes on a higher importance. Routing algorithms based on traditional routing can sometimes be a poor choice for minimizing cost and reliability of the communication. A key challenge is to develop a routing algorithm such that it produces an optimal network with a minimum total link cost; that is both cycle-free and free of redundancy.

The project tackled that problem, focusing in on the design, development and implementation of routing using Kruskal's algorithm. Kruskal's algorithm is a powerful routing algorithm that produces a Minimum Spanning Tree (MST). In our project application of Kruskal's algorithm identifies the least expensive set of connections that connect all nodes in the network, minimizing the cost of transmission and avoiding unnecessary paths, productive efficiency is achieved through the things we do not realize, and action is taken in good faith. This application is ideal where cost optimization of a network is a priority.

2. Introduction to Algorithm chosen

In graph theory, one of the most fundamental problems is finding the Minimum Spanning Tree (MST) of a connected, weighted graph. This problem has far-reaching implications in various fields, including computer science, engineering, and operations research. The Minimum Spanning Tree problem involves identifying the subset of edges in a graph that connect all vertices while minimizing the total edge weight. This problem is crucial in designing efficient networks, such as telecommunications, transportation systems, and electrical grids, where the goal is to minimize costs while ensuring connectivity.

Kruskal's Algorithm:

Kruskal's algorithm is a popular and efficient solution for finding the Minimum Spanning Tree of a graph. Developed by Joseph Kruskal in 1956, this algorithm has been widely used in various applications due to its simplicity, efficiency, and scalability. Kruskal's algorithm works by sorting the edges of the graph in non-decreasing order of their weights and then selecting the smallest edge that does not form a cycle. This process is repeated until all vertices are connected, resulting in the Minimum Spanning Tree. The significance of Kruskal's algorithm extends beyond its theoretical implications. In real-world applications, finding the Minimum Spanning Tree is essential in designing efficient networks, reducing costs, and improving

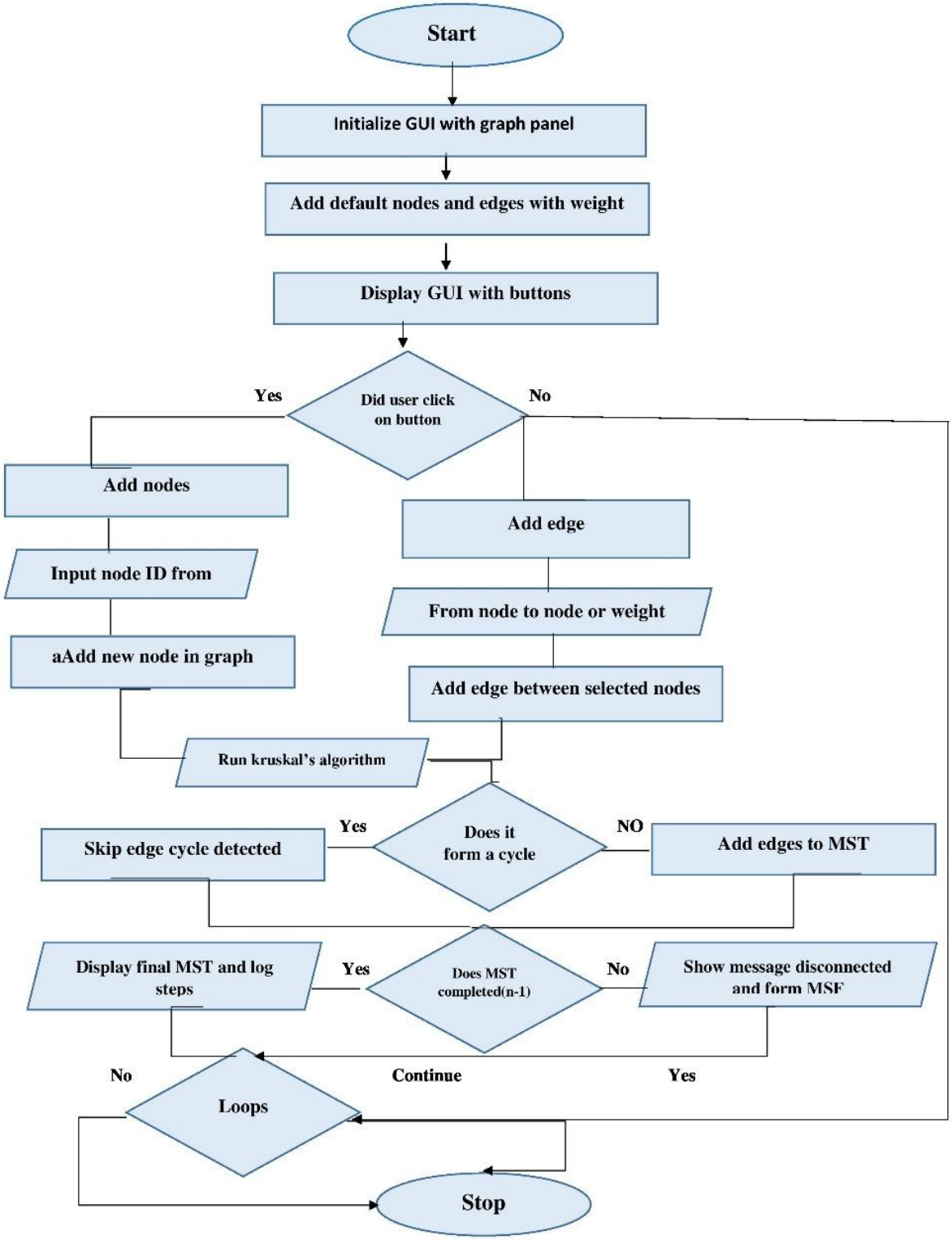
performance. For instance, in telecommunications, the MST problem is used to design network topologies that minimize the cost of installing cables while ensuring connectivity between all nodes. Similarly, in transportation systems, the MST problem is used to design routes that minimize travel time and costs.

This project aims to implement Kruskal's algorithm in Java, providing a practical solution for finding the Minimum Spanning Tree of a graph. Through this implementation, we will explore the algorithm's efficiency, scalability, and real-world applications. We will also examine the challenges and limitations of implementing Kruskal's algorithm in Java and discuss potential optimizations and improvements. By implementing Kruskal's algorithm in Java, this project will contribute to the understanding and application of graph theory in computer science and engineering. The project's findings and results will provide valuable insights into the algorithm's performance and limitations, shedding light on its potential applications and future directions. Ultimately, this project will demonstrate the power and versatility of Kruskal's algorithm in solving complex graph problems and its significance in real-world applications.

3. Methodology (Flowchart)

To accomplish this project, we followed a systematic approach that included designing and implementing Kruskal's algorithm using Java, testing it with various datasets, and evaluating its performance. We ensured the algorithm's correctness through detailed testing and checked its efficiency and scalability by applying it to different network sizes and complexities. This practical approach allowed us to gain hands-on experience with the algorithm and understand its strengths and limitations, providing valuable insights for real-world applications of Kruskal's algorithm in network design and optimization.

Flowchart:



Visual representation of algorithm logic and work flow.

4. Implementation details

Our project is a graph visualizer that illustrates the process of using Kruskal's algorithm to find the Minimum Spanning Tree (MST) of a graph. The user can create nodes and edges, run the algorithm, and visualize it.

The project has several classes; all have their own instance variables and methods, including Graph, Node, Edge, DrawingCanvas, GraphPanel and Kruskal Algorithm. The Graph class represents a graph that is created with nodes and edges; the Node class creates a node; the Edge classes create a directed edge between nodes. The Drawing Canvas class is used to create the visual presentation of the graph, such a drawing the nodes and drawing the edges; whereas the GraphPanel has the graph the buttons for user input and a log area. Lastly Kruskal Algorithm class uses the Kruskal algorithm to find the MST.

The graph is represented by an adjacency list in which for each node there is a list of edges that belongs to the node. In order to implement the Kruskal algorithm, we used a disjoint set data structure, to keep track connected components. The edges are first sorted according to weight; we then taken each edge and see if can be added to the MST without creating a cycle.

The visualization is done through Java's Graphics API, where nodes are depicted as circles and edges are represented as lines. our program is interactive, allowing the user to create nodes and edges, run the algorithm and visualize the algorithm's process. User actions are happening through JButton and JOptionPane. The code implementation details include the use of a disjoint-set data structure to keep track of connected components, sorting edges by weight, and iterating through the sorted edges (in order to find the MST). The visualization updated during the algorithm as it was being animated, so the viewer could see the MST being built up as it would have been done by Kruskal's algorithm.

In summary, this project shows how Kruskal's algorithm is used to calculate the MST of a graph, while also portraying a nice tool for visualizing and understanding how the algorithm works.

Key Implementation Details:

1. LANGUAGE: Java.

2. TOOL: Java swing and java Graphics API (used for visualizing a graph)

3. IDE USED: Intellij IDEA.

4. STRUCTURE OF THE CODE: Multiple classes (MainApp, Graph, Node, Edge, DrawingCanvas, GraphPanel, Kruskal's Algorithm) using OOP with adjacency list and disjoint set data structure.

5. Visualization Approach

The visualization strategy adopted in this project clearly demonstrates the working of kruskal's Algorithm for finding minimum spanning tree (MST) of a connected, weighted, and undirected graph. The intention for this visualization is to allow end-users a step-by-step, real-time depiction of how the algorithm processes a graph, selects edges, and constructs the final MST. This interactive and dynamic depiction serves to provide a more conceptual understanding for the user by changing an algorithmic representation into a visual representation that is more intuitive.

Graphical Representation:

The graph is visually constructed using a set of nodes and weighted edges. Each node represented as a circle, virtually, labelled with a letter (A, B, C, etc.,) each edge as a line connecting two nodes. Each edge's weight is displayed directly on the edge as a numerical value, which lets the user know the "cost" of the edges in connection to the other nodes. This graphical representation will provide an opportunity to see how Kruskal's algorithm sorts and chooses edges based on weight.

Animated MST construction

The visualization highlights the step-by-step construction of the Minimum Spanning Tree (MST) using Kruskal's algorithm. Edges are processed in order of weight, and valid edges are added to the MST, typically shown in black and green. Rejected edges (those forming cycles) are skipped and often shown in red. The animation clearly illustrates each step.

Interactive Log Panel

Along with the visual animation, the system will also provide a log area or message panel that will chronicle each step of the algorithm in a textual form as well.

This will include:

1. Which edge is being considered (and the weight of it)

2. Why it is included in the MST or why it is rejected

3. The status of the node groups - or components merged.

The log serves as a running commentary of the algorithm's execution and will provide the user with two forms of learning - visually and textually. The log is beneficial to learners and users for debugging, presentation, or demonstration purposes.

Benefits

Overall, visualization provides various benefits with respect to learning and engagement, allowing the user/instruction to be easier to follow. It presents the important step feedback of Kruskal's algorithm to visually depict how the Minimum Spanning Tree (MST) is created, and allows the user to follow the steps along the way. In addition to highlighting the edges being selected and the adding log messages, the visualization offers a visual representation of important concepts of edge selection, cycle detection, and merging components together. Lastly, the ability to adjust functionality encourages experimentation on the part of the user. The main benefits of the visualization are that it improves conceptual understanding, supports active learning, and provides a pathway to introduce, explain and prepare learners studying graph theory and algorithms.

6. Results and Screenshots

The implementation of Kruskal's algorithm within our project has produced valid Minimum Spanning Trees (MSTs) from a variety of graph types. Users were able to create graphs visually and interactively, as they assigned weights on edges and their progress was displayed as they watched their MST grow. Players selected edges of minimum weight in Kruskal's algorithm without forming cycles, and maintained proper connectivity through a disjoint set data structure that kept track of connected components. The edges clearly presented visual feedback: the selected edges were colored (green) while rejected edges that would happen to form cycles were otherwise colored (red). The log panel provided real time updates of each move, offered a narrative to each decision of accept/reject edge, and others for merging components. The system is tested with different size graphs, consistently demonstrating correctness in every case. The system was able to handle invalid or incomplete user input such as unconnected nodes and the use of special characters, without failure. The visualization allowed each step of constructing the MST to be visualized, as a whole increase the level of attention from the user. The project met its objectives, providing correct, interactive and educational demonstration of Kruskal's Algorithm and its application in designing networks at cost efficient level.

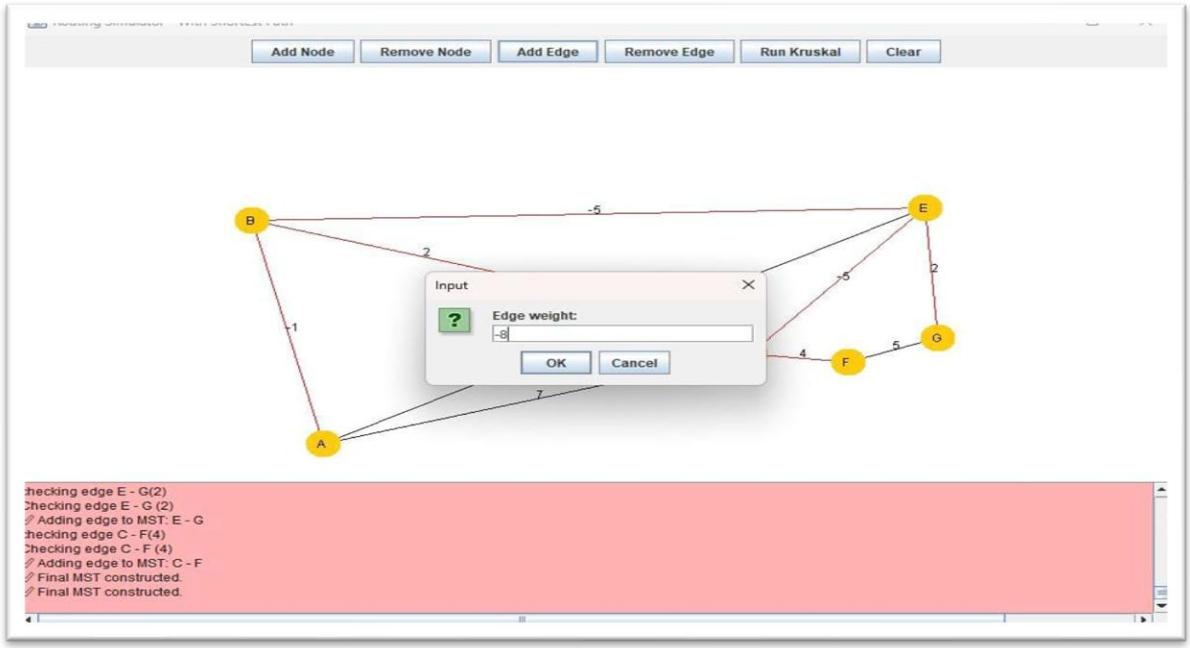


Figure 1 description: This screenshot shows the edge weight input dialog for a Kruskal's algorithm simulation. Users enter the desired weight (positive or negative) of the edge they selected, and that weight is utilized to build the Minimum Spanning Tree (MST). The console log below demonstrates the steps taken, indicated by the final console log of successfully completing the MST.

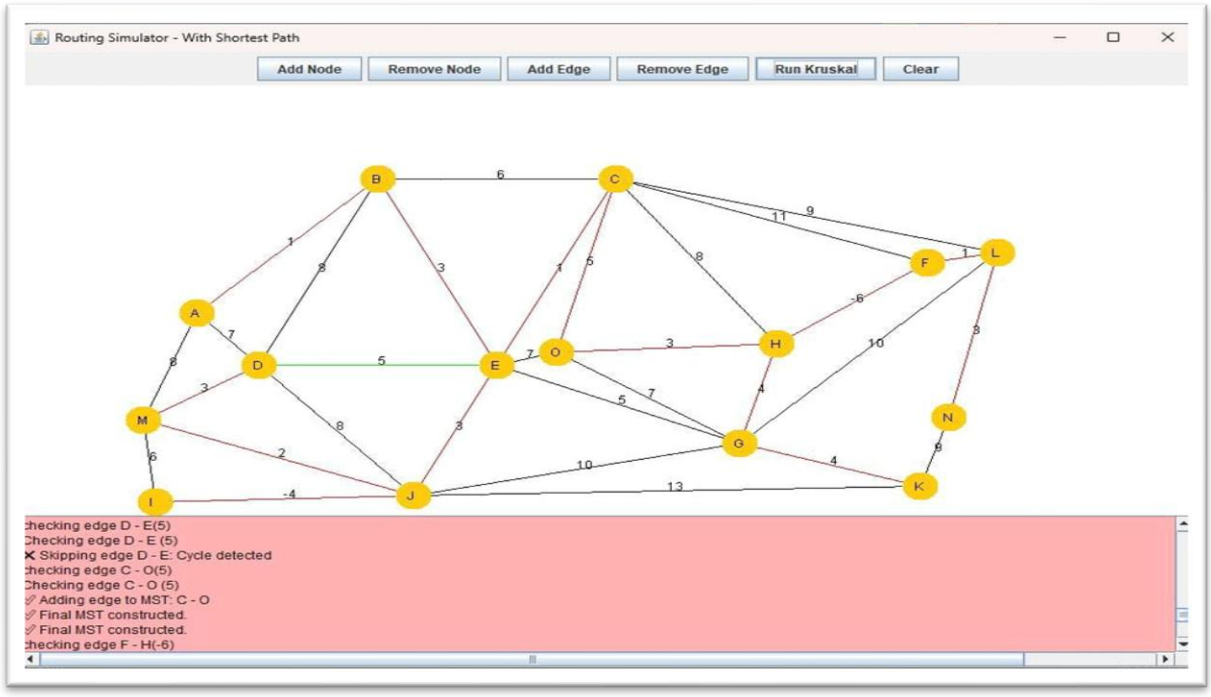


Figure 2 description: This screenshot shows the Kruskal's algorithm adding edges of varying weight and checking for cycles. Edges that form cycles will be skipped as seen in the console log so that only valid edges are added in order to build the MST.

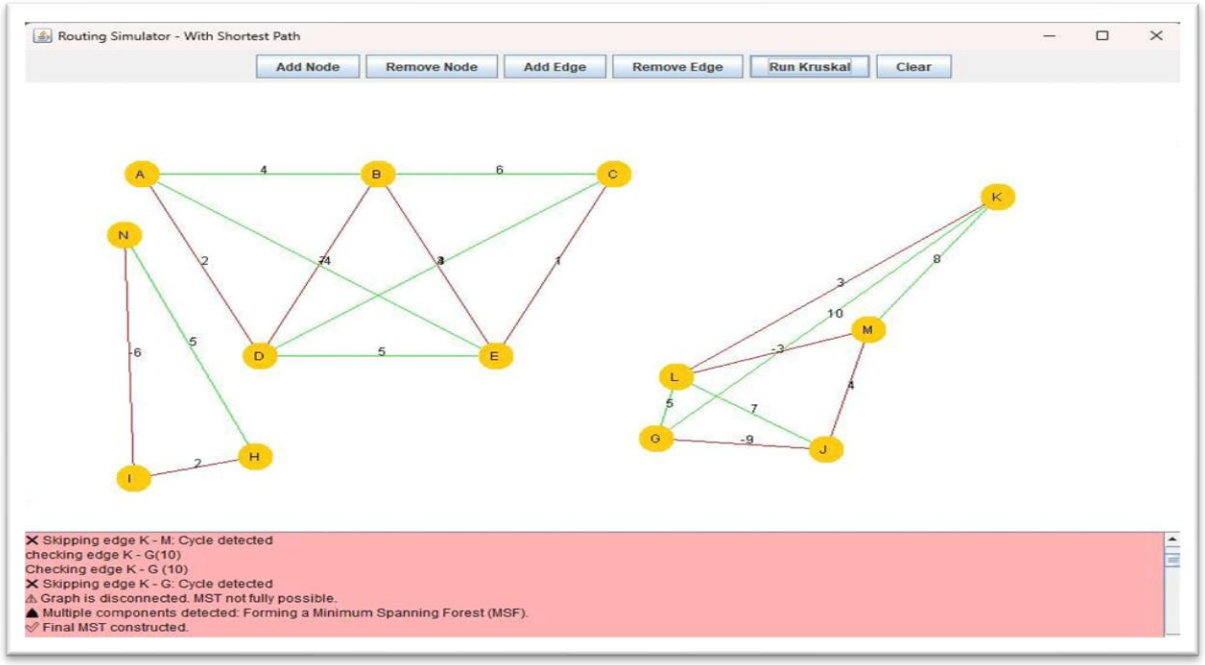


Figure 3 description: This screenshot shows the Kruskal's algorithm on a disconnected graph, which results in the creation of a Minimum Spanning Forest (MSF) as opposed to a single MST. The algorithm identifies edges within each connected component, while ignoring edges that create cycles, as indicated in the console. The green edges represent the selected paths for each sub-tree of the MSF.

7. Conclusion

In summary, Kruskal's algorithm is a powerful and efficient method for finding the Minimum Spanning Tree of a weighted, connected graph. We have demonstrated the algorithm in Java and its application for optimizing design and improving overall performance. The efficiency of Kruskal's algorithm on large-scale networks is useful for telecommunications, transportation and electrical grid systems. Utilization of the Kruskal's algorithm gives developers and engineers the ability to build networks capable of providing maximum effectiveness with the lowest cost, while improving network reliability and performance. The scalability and reproducibility of the Kruskal's algorithm provides an opportunity to provide solutions to complex network problems, and our implementation in the Java programming language shows the potential of utilizing it to find real world solutions. The success of the implementation of Kruskal's algorithm in Java demonstrates the importance of graph theory in the fields of computer science and engineering. With the continuing increase in network complexity and size, the role of efficient algorithms, such as Kruskal's algorithm, will be significant in developing and optimizing network systems. Overall, this project demonstrates the importance of the Kruskal's algorithm in graph theory as we tackle various types of complex problems.

References

1. For understanding the concept of spanning tree.

https://youtu.be/h6cUkkaKNHw?si=qDp9sn1HTy-nMY3j

2. For understanding the basic concept of kruskal’s Algorithm.

https://youtu.be/huQojf2tevI?si=i79q6QyBPOdgvFu0

3. Union find kruskal’s Algorithm.

https://youtu.be/JZBQLXgSGfs?si=rhISSmnDDb2-UfCm

4. Kruskal’s Algorithm (Minimum Spanning Tree)

https://www.geeksforgeeks.org/kruskals-algorithm-in-java/