Report

Semester Project Part-2

[Design and Visualization of Prim's Algorithm for MST]

[Computer Networks]

[Fall 2024]

[66608, 67656, 66755]

[Suhana, Muhammad Usman, Rafia Saleem]

[2nd Semester]

Submitted to:

[Ms. Poma Panezai]

[Department of Computer Science]

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

Abstract

In this project, we implemented and visualized Prim's Algorithm for generating a Minimum Spanning Tree (MST). Prim's Algorithm is a greedy algorithm that grows the MST by selecting the lowest weight edge at every step. MSTs are useful in network routing because they help connect all the nodes with the least total cost and no loops. We used Java as the programming language and tools like GraphStream and JavaFX for the visual part of the simulation. The application allows users to create custom graphs by adding nodes and weighted edges, choose a starting point, and watch how MST is formed step by step. This made it easier to understand how Prim's Algorithm works in real time. This project enhances the understanding of MST based routing and its efficiency in building loop free, minimum cost topologies. We also improved our skills in Java programming, user interface design, and algorithm visualization.

Table of Contents

1. Problem Statement

In today's computer networks, routing plays a critical role in ensuring that data is transferred quickly, efficiently and reliably between different devices or nodes. As the network grows larger and complex, it becomes important to use algorithms that can find the best paths to send the data while avoiding redundancy and unnecessary delays. One effective approach is to build a Minimum Spanning Tree (MST), which connects all nodes in the network with the lowest total connection cost, without any loops.

Prim's Algorithm is one such tree based routing method, known for its greedy strategy of always selecting the next shortest edge. It ensures the construction of a loop free, cost efficient network. I this project, we used the Prim's Algorithm to understand how MSTs work and how they can be applied to network routing, we aim to implement the algorithm and with that visualize the entire process.

2. Algorithm Chosen

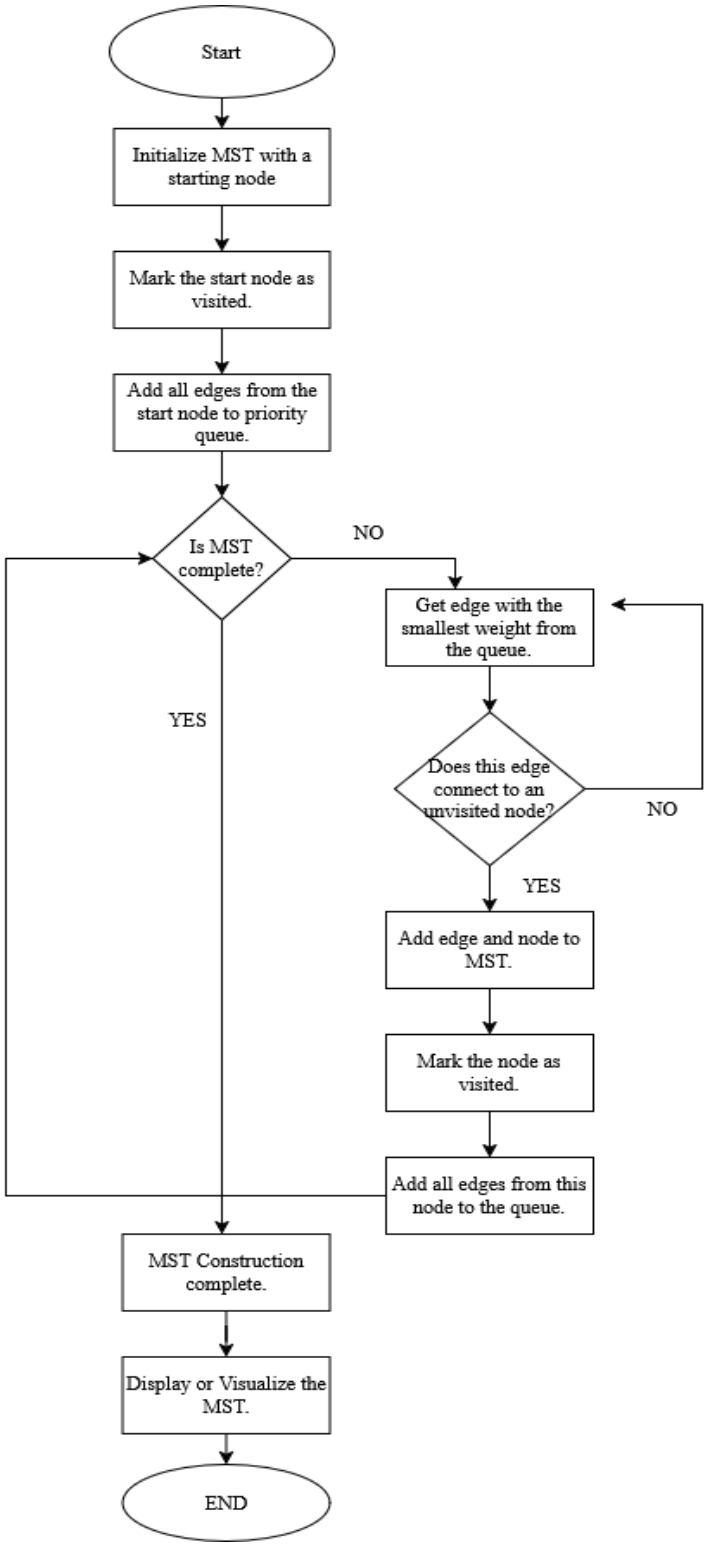
Prim's Algorithm – Greedy MST Construction

Prim's Algorithm is a greedy algorithm used to find the MST of a weighted, undirected graph. Prim's Algorithm works by repeatedly building a tree, starting from single vertex, and adding the smallest edge that connects e current tree to a new vertex not yet in the tree. This process continues until all the vertices are included in the tree, resulting in an MST with the minimum total edge weight.

How Prim's Algorithm Works

1. **Initialization**
   Start with a single node and mark it as visited.
2. **Edge Selection**
   Continuously add the minimum weight edge that connects a visited node to an unvisited node.
3. **Expansion**
   Mark the newly added node as visited node and its outgoing edges to a priority queue.
4. **Termination**
   Repeat until all nodes are included in the MST.

3. Flowchart


Start
Initialize MST with a starting node
Mark the start node as visited.
Add all edges from the start node to priority queue.
Is MST complete?
NO
YES
Get edge with the smallest weight from the queue.
Does this edge connect to an unvisited node?
NO
YES
Add edge and node to MST.
Mark the node as visited.
Add all edges from this node to the queue.
MST Construction complete.
Display or Visualize the MST.
END

4. Implementation Details

Development Environment

Language Used – Java.

IDE – IntelliJ IDEA.

Visualization Libraries – JavaFX, GraphStream, Swing.

- **JavaFX –** for the GUI interaction.
- **Swing –** for additional user interface components.
- **GraphStream –** for graph drawing and dynamic updates.

Graph Representation

- Nodes class with ID and position attributes.
- Edge class with source and weight.

Algorithm Implementation

Prim's Algorithm class with methods

- Initialization
- Finding minimum key
- Updating adjacent nodes
- Constructing MST

Algorithm Tracing

- Console output of each step
- Highlighting of current operations
- Final MST display with total weight

5. Visualization Approach

We used the graph stream library to dynamically show how the Minimum Spanning Tree (MST) grows step by step during the execution of Prim's Algorithm. Nodes and edges are visually updated to reflect their inclusion in the MST. JavaFX was used to create smooth graphical effects and animations. While Swing was used to handle user interface components. The combination of both made the interface both interactive and user friendly. Users can add or remove nodes, define edge weights and initiate the algorithm visually. The visualization uses GraphStream's force directed layout algorithm to automatically arrange nodes in a readable format. Nodes are representedf as circles with unique IDs, while edges are drawn as lines with weight labels.

Animation Sequence

- **Initial State**
- **Selection Phase**
- **Evaluation Phase**
- **Selection Phase**
- **Update Phase**
- **Completion**

6. Results and Screenshots

The program was successfully constructed with a visual of showing nodes connected to one another showing the shortest path , upon initiating Prim's Algorithm for a given start and goal the algorithm begins , the nodes light up in order and edges are explored. The nodes are colored as they were explored, and the resultant shortest path is of different color. Hence, we were able to successfully make our project work and was able to show the shortest path between two nodes the nodes that were visited plus the nodes that were not yet visited.

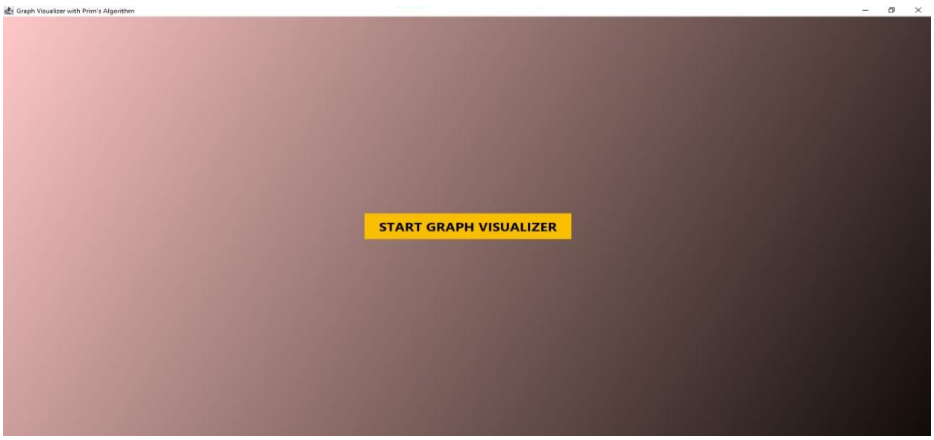


Figure 1 Application Startup Screen

Initial interface of the Graph Visualizer implementing Prim's Algorithm, showing the launch point for the simulation

Figure 2 Main Menu Interface

Users can choose between manual graph input or random generation, with navigation controls to exit or return to previous screens.

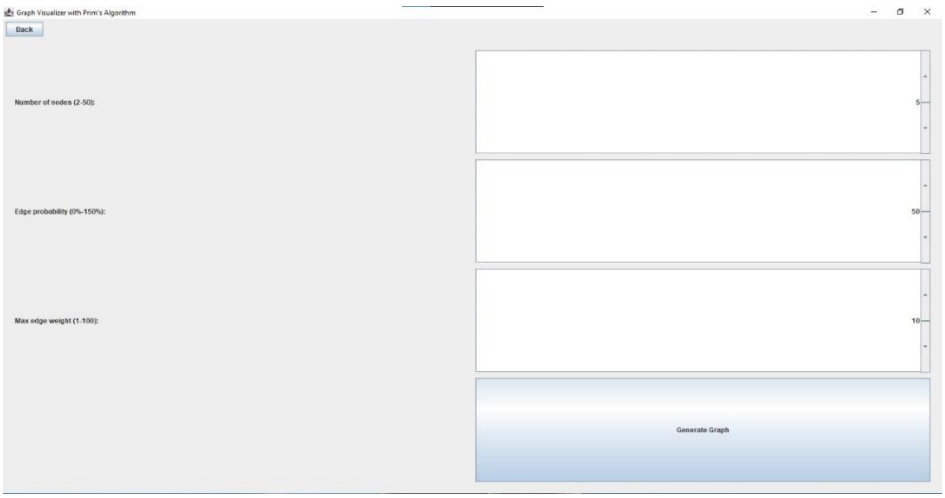


Figure 3 Random Graph Configuration

Parameters for automated graph generation including node count (2-50), edge probability (0-150%), and maximum edge weight (1-100).

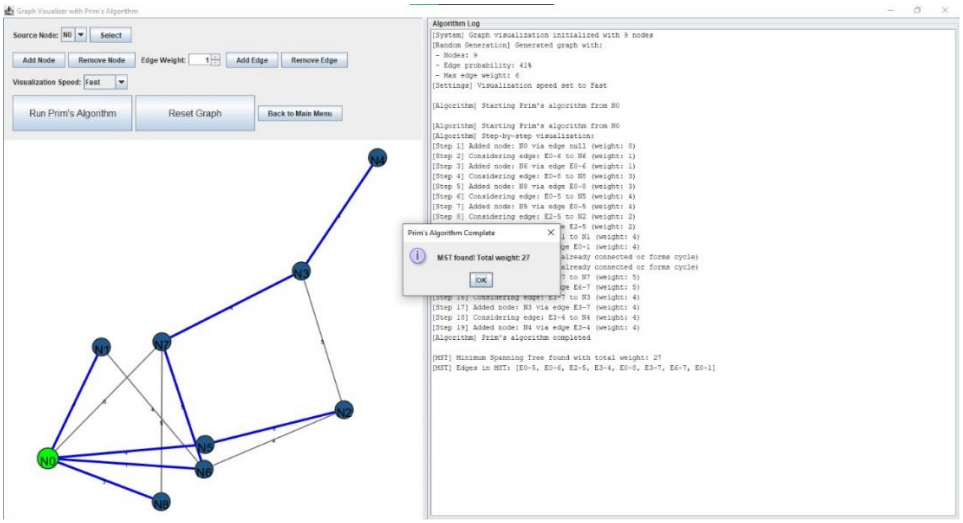


Figure 4 Algorithm Execution Log

Step by Step visualization of Prim's Algorithm building a MST with total weight 27, showing node additions and edge evaluations.

7. Conclusion

In this project we successfully implemented Prim's Algorithm to find the shortest path in an unweighted graph and used javaFX to visualize the process. All the applications including the JavaFX was downloaded smoothly without freezing but we did face a bit difficulty in updating the libraries , we also faced challenges in connecting the nodes correctly, as well as giving them a different color each, these were overcome by testing and stepwise debugging. Overall, it was a successful project

References

https://www.geeksforgeeks.org/prims-algorithm-with-a-java-implementation/

https://openjfx.io/

http://graphstream-project.org