1. Introduction

Routing is the process of path selection in any network. A computer network is made of many machines, called nodes, and paths or links that connect those nodes. Communication between two nodes in an interconnected network can take place through many different paths.

2. Problem Statement

Despite the importance of routing algorithms in a network management system, the concept behind those algorithms is not properly understood and inner working is difficult to be interpreted by traditional teaching methods. Static diagrams often fail to convey the nature of working of such algorithms, making it challenging to students to comprehend the mechanism behind those algorithms. This project seeks to address this issue by providing a user-friendly interface to students with a proper visualization experience, allowing them to gain a deeper understanding of routing algorithm concepts.

However, this system doesn’t aim to track the execution of these algorithms as a means of comparison, this option is given to user themselves outside the system boundary. User also can’t skip the run of the algorithm’s steps to arrive at the final routing table and final path.

3. Objectives

The primary goals of Network Routing Simulator are to,

1. Develop an interactive web application using HTML, CSS, JS and p5.js. p5.js is a JavaScript library for creative coding consisting of a collection of pre-written code, it provides us with tools that simplify the process of creating interactive visuals with code in the web browser [2]. We will be utilizing p5.js for graph creation and traversal of algorithm steps visually.
2. Display and update a routing table that contains information about vertex, distance and predecessor in each pass of the algorithm.
3. Develop the backend part that consists of algorithm implementation, carried out using Python.

4. Methodology

a. Requirement Identification

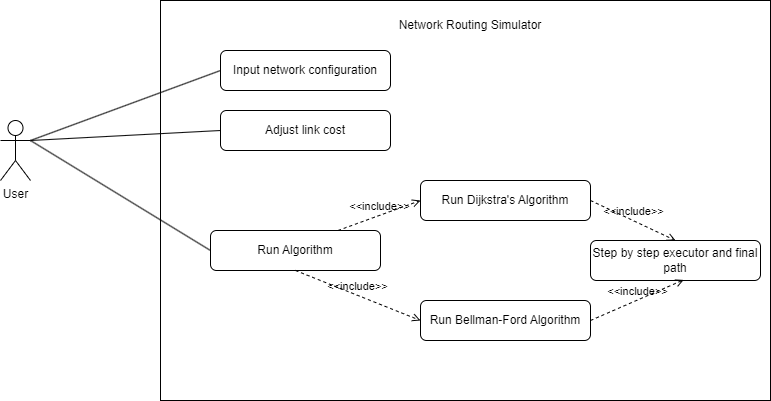


Fig 1: Use-Case Diagram

i. Study of Existing System/ Literature Review

OMNeT++:

OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators [3]. “Network” is meant in a broader sense that includes wired and wireless communication networks, on-chip networks, queueing networks, and so on [3]. Domain-specific functionality such as support for sensor networks, wireless ad-hoc networks, Internet protocols, performance modeling, photonic networks, etc., is provided by model frameworks, developed as independent projects [3]. OMNet++ can be utilized to create different network configuration and C++ code can be written to implement and visualize Dijkstra’s and Bellman-Ford algorithm to see the working of those algorithms.

Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization- Samah W.G AbuSalim et al 2020:

Research for finding the best algorithms for handling the issue of shortest path has been on going. In the real world it is easy to apply the graph theory to different types of scenarios. In shortest path algorithm, the study focuses on two nodes or vertices of the path and find the best solution for the shortest path [4]. Dijkstra’s and Bellman-Ford algorithm are the best contenders for shortest path determination so we want to create a system to visualize the working for these algorithms to invoke a sense of familiarity when using them.

E-learning Tool for Visualization of Shortest Paths Algorithms- Daniela Borissova and Ivan Mustakerov:

Visualizations of algorithms contribute to improving computer science education [5]. The process of teaching and learning of algorithms is often complex and hard to understand problem [5]. Visualization is a useful technique for learning in any computer science course [5]. This project employs these ideas for creating an educative, interactive and visual platform to depict the working mechanism of single source shortest path algorithms.

ii. Requirement Analysis

On the basis of the requirement identified, represented by the use case diagram (Fig 1) we have major parts like, creating an interface which can visualize the network configuration , implementing the working of the algorithm in a step-by-step visualization process(dynamic routing table) and a communication channel between the frontend and backend of the system.

b. Feasibility Study

i. Technical

The project can be accomplished by highlighting the relevant knowledge and skills gained from our course work. Also, the hardware and software pre-requisites in easily available to execute the system in hand. Given our technical expertise and course work we are confident in the technical feasibility of the project and its successful implementation.

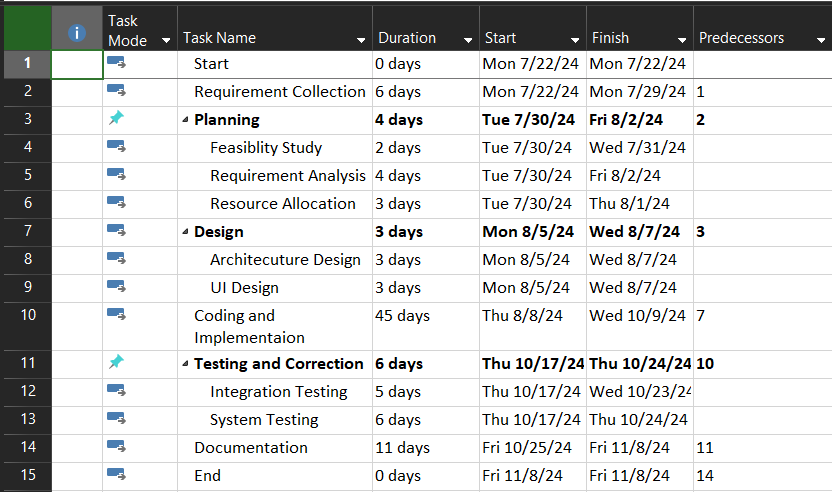
ii. Operational

The project will be managed and maintained solely by the members requiring a moderate time commitment. Given the resource and planning the operational feasibility of our project is assured with high likelihood of successful implementation and sustained operation.

iii. Economic

All the resources required to execute this project is freely and easily available with no additional expenses that is to be concurred. The expected benefit includes academic and career advantages with low to none investment.

iv. Schedule

The tentative schedule of the project is implored by a Gantt Chart, 

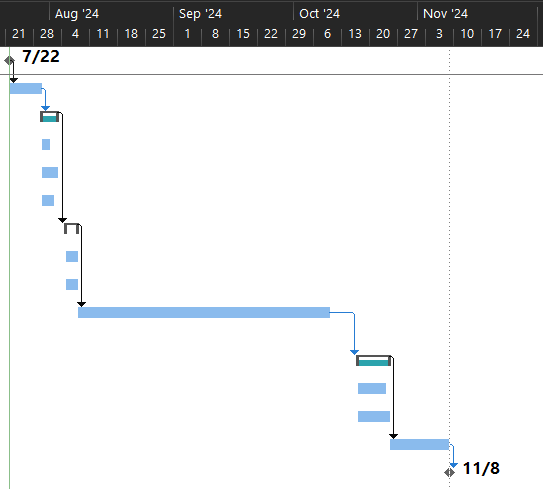


Fig 2: Gantt Chart of proposed system

c. High Level Design of System

i. Working Mechanism of Proposed System

The flowchart shown below shows the diagrammatical representation of working of the system.

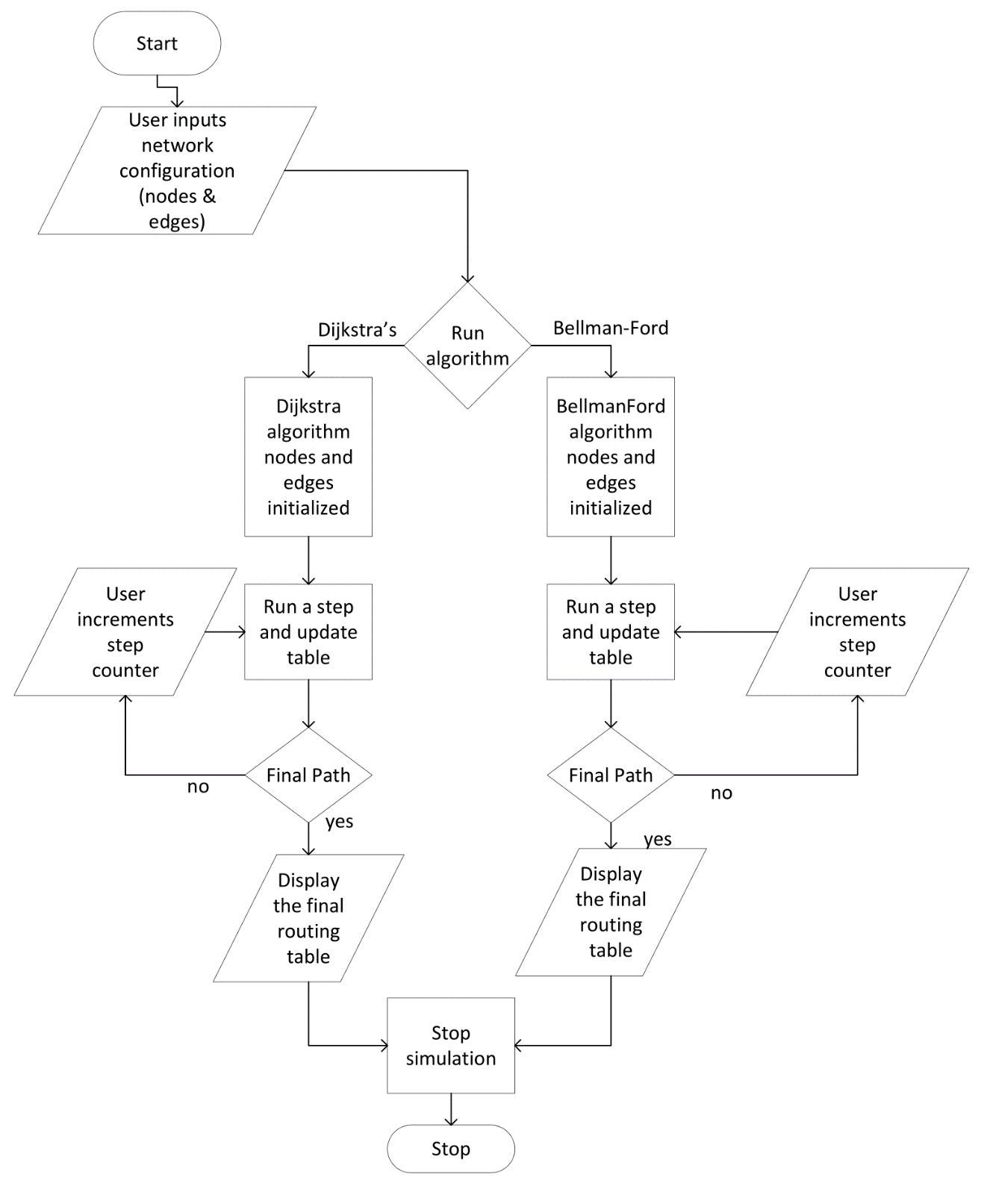


Fig 3: Flowchart of Network Routing Simulator

ii. Description of Algorithm

The single-source shortest path algorithms used in this proposed system uses the technique of relaxation. For each vertex v ∈ V, the single-source shortest paths algorithms maintain an attribute v.d(shortest-path estimate), which is an upper bound on the weight of a shortest path from source s to v [6].

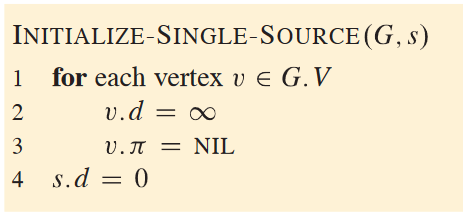


Fig 4.1: Initialization of shortest-path estimate

The process of relaxing an edge (u,v) consists of testing whether going through vertex u improves the shortest path to vertex v found so far and, if so, updating v.d and v.π [6].

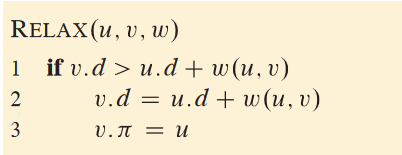


Fig 4.2: Relaxing an edge (u,v) with weight w

Bellman-Ford Algorithm:

The Bellman-Ford algorithm solves the single-source shortest-paths problem in the general case in which edge weights may be negative. The procedure BELLMAN-FORD relaxes edges, progressively decreasing an estimate v.d on the weight of a shortest path from the source s to each vertex v ∈ V until it achieves the actual shortest-path weight δ (s,v). The algorithm returns TRUE if and only if the graph contains no negative-weight cycles that are reachable from the source [6].

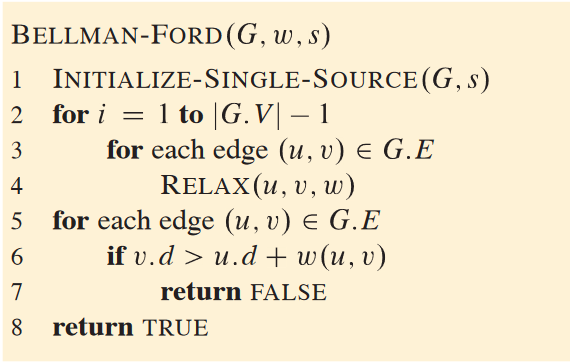


Fig 4.3: Bellman-Ford algorithm pseudocode

Dijkstra’s Algorithm:

Dijkstra’s algorithm solves the single-source shortest-paths problem on a graph G= (V, E), but it requires nonnegative weights on all edges: w (u, v) ≥ 0 for each edge (u, v) ∈ E. Dijkstra’s algorithm maintains a set S of vertices whose final shortest-path weights from the source s have already been determined. The algorithm repeatedly selects the vertex u ∈ V - S with the minimum shortest-path estimate, adds u into S, and relaxes all edges leaving u. The procedure DIJKSTRA uses a min-priority queue Q of vertices, keyed by their d values [6].

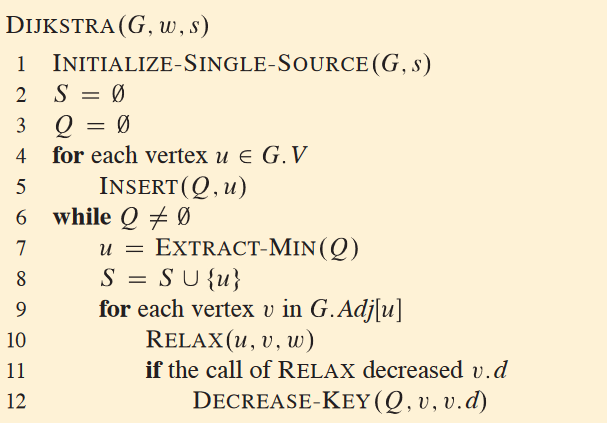


Fig 4.4: Dijkstra’s algorithm pseudocode

5. Expected Outcome

The User Interface (UI) will have provision for the user to input custom network configurations with buttons like add node and add edge. Add Edge will require the start node, end node and the cost of link between them. Two additional buttons will be placed to start Dijkstra’s and Bellman-Ford algorithm simulation. Also, a step-by-step routing table will be updated on each step incremented by the user until final routing table can be determined. Sample wireframe to depict the behavior of our system is created using Balsamiq.

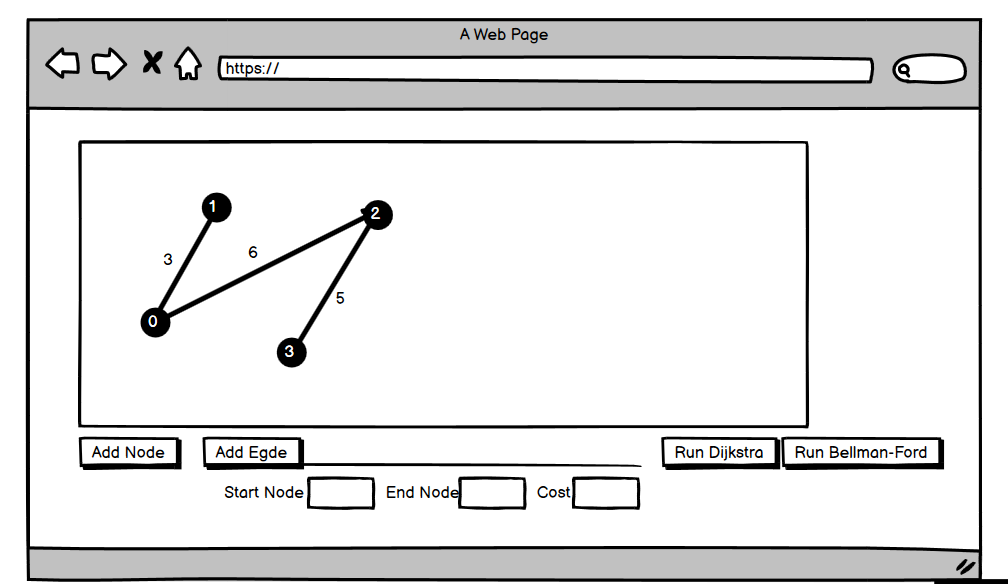


Fig 5.1: UI wireframe

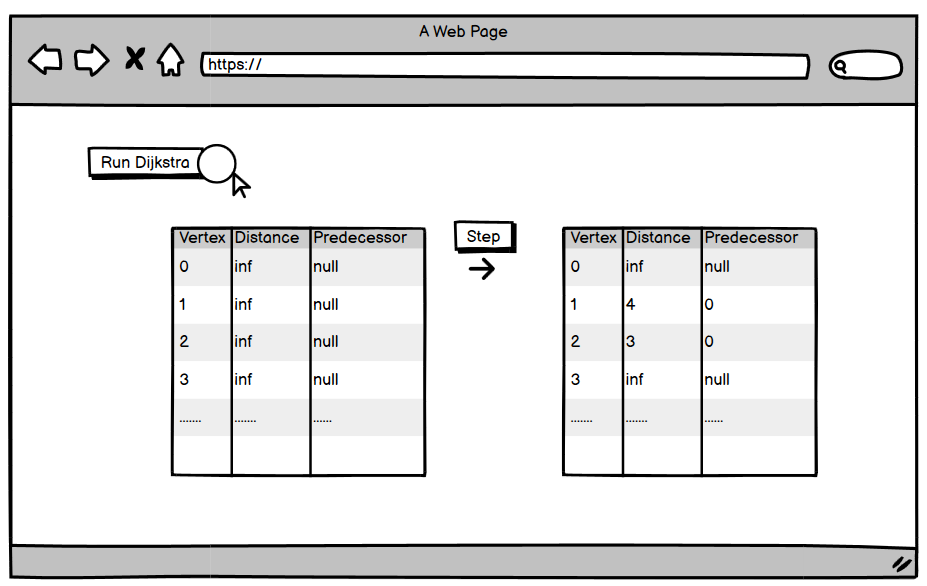


Fig 5.2: Depiction of routing table being updated after step counter is clicked

6. References

[1] AWS, “What is Routing - Beginner’s Guide to Network Routing - AWS,” *Amazon Web Services, Inc.*, 2023. <https://aws.amazon.com/what-is/routing/>

[2] “Learn p5.js: Introduction to Creative Coding Cheatsheet,” *Codecademy*. https://www.codecademy.com/learn/learn-p5js/modules/p5js-introduction-to-creative-coding/cheatsheet

[3] “What is OMNeT++?,” *Omnetpp.org*, 2019. <https://omnetpp.org/intro/>

[4] S. W. G. AbuSalim, R. Ibrahim, M. Zainuri Saringat, S. Jamel, and J. Abdul Wahab, “Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization,” *IOP Conference Series: Materials Science and Engineering*, vol. 917, p. 012077, Sep. 2020, doi: <https://doi.org/10.1088/1757-899x/917/1/012077>.

[5] D. Borissova and I. Mustakerov, “E-learning Tool for Visualization of Shortest Paths Algorithms,” *Trends Journal of Sciences Research*, vol. 2, no. 3, pp. 84–89, Sep. 2015, doi: https://doi.org/10.31586/informationprocesses.0203.01.

[6] T. H. Cormen, Charles Eric Leiserson, R. L. Rivest, and C. Stein, Introduction to algorithms. Cambridge, Massachusett: The Mit Press, 2022.