Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: 4th (Spring, Year: 2023), B.Sc. in CSE (Day)

Efficient Pathfinding using Dijkstra's Algorithm: Visualizing the Shortest Route between Two Points

---

Course Title: Object Oriented Programming Lab
Course Code: CSE 202 / Section: DI

Students Details

Name	ID
KHONDOKAR SAIM	**221902353**

Submission Date: 21 /06/2023
Course Teacher's Name: Md. Parvez Hossain

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

The "Efficient Pathfinding using Dijkstra's Algorithm: Visualizing the Shortest Route between Two Points" is a Java project aimed at implementing Dijkstra's Algorithm for efficient pathfinding in a graph. The project focuses on finding the shortest route between two points and provides a visual representation of the path for enhanced understanding and user experience.

1.2 Motivation

The motivation behind developing our project aims to streamline the -

[1] Process of finding the shortest route
[2] Enhance decision-making capabilities
[3] Contribute to advancements in various domains that heavily rely on efficient pathfinding.

1.3 Problem Definition

1.3.1 Problem Statement

The problem at hand is to develop a Java program that efficiently finds and visualizes the shortest route between two points using Dijkstra's Algorithm. The program should take a graph as input, with nodes representing locations and edges representing connections between them. The goal is to calculate the shortest path from a given source node to a target node, considering the weights assigned to each edge, and present it visually to the user.

1.3.2 Complex Engineering Problem

The complex engineering problem in this project lies in efficiently implementing Dijkstra's Algorithm to find the shortest route between two points in a given graph.

[1] This requires handling large-scale graphs with numerous nodes and edges
[2] Optimizing the algorithm's time complexity
[3] Ensuring accurate results.

Additionally,This project requires a combination of

[1] Algorithmic expertise
[2] Data structure optimization
[3] Graphical representation skills to overcome these engineering challenges and provide an efficient and visually appealing solution for pathfindin

Chapter 2

Design/Development/Implementation of the Project

2.1 Project Details

The Efficient Pathfinding project aims to implement Dijkstra's Algorithm in Java to find the shortest route between two points in a graph. The project focuses on visualizing the obtained shortest route for enhanced understanding and user experience.

2.1.1 Efficient Pathfinding project GUI interface

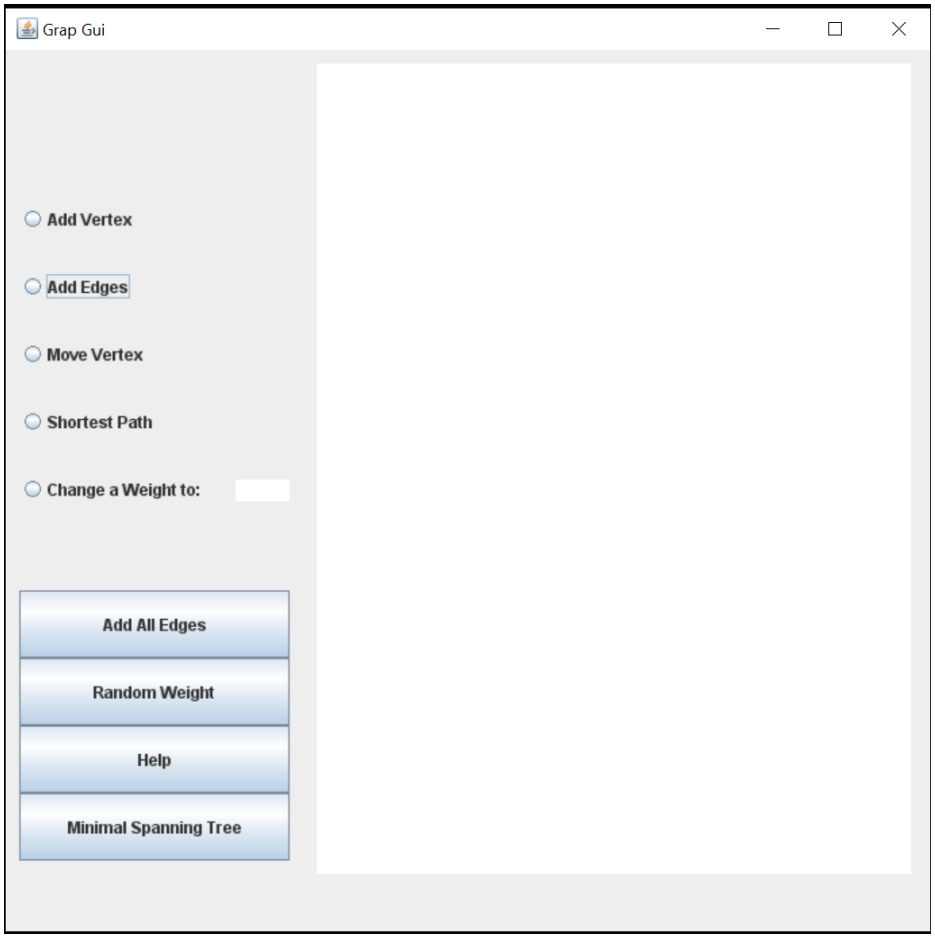


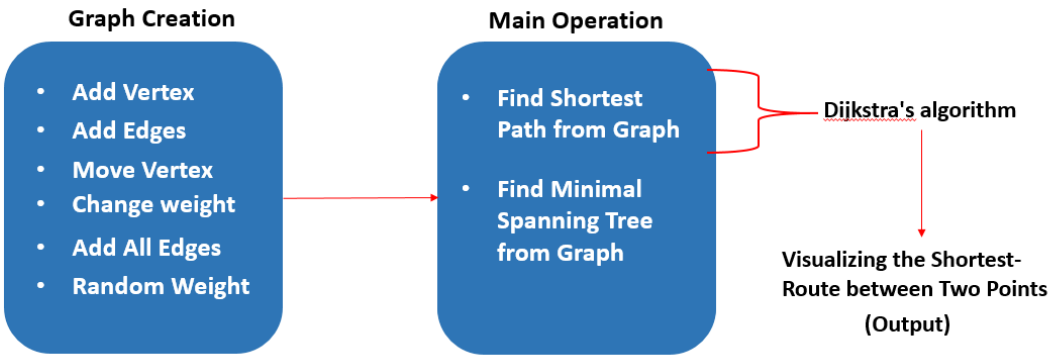
Figure 2.1: GUI interface

2.2 Implementation

All the implementation details of this project include in this section, alongwith many subsections.

2.2.1 Implementation Procedure

The workflow



Tools and libraries

Tools

The implementation of the algorithm and visualization can be done using java Programming Language. Because this language offer extensive libraries and frameworks for graph manipulation, data structures, and user interface development. For helps in coding, debugging, and managing the project effectively I am going to use ***Apache NetBeans 17 IDE*** as a tool.

Libraries

Here's an explanation of the libraries used in the code:

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
```

```
import java.util.LinkedList;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import javax.swing.*;
```

These libraries provide essential classes and functionalities for GUI programming, handling events, data structures, and random number generation, enabling the code to create interactive graphical applications and manipulate data effectively.

Implementation details (with screenshots and programming codes)

- Select Add Vertex and left click on the right empty space to create a dot or Vertex.--> Create more than one vertex

- Select Add Edge and Connect the vertex to create a link between two vertex.

- Select Move Vertex to move vertex to another position.

- Select Shortest Path to find the shortest path from one vertex to another.

- Select on Change weight to, and then enter a weight then Click on a Edge to change the weight of specific edge.

- Click on Add All Edge to connect all the edges.

- Click on Add Random Weight to provide random weight to the edges.

- Click on Minimal Spanning Tree to get the minimal spanning tree

- There might be some issue if you clicked on Minimal Spanning Tree and then Shorted Path.

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

For helps in coding, debugging, and managing the project effectively I am going to use ***Apache NetBeans 17 IDE*** as a Simulation Environment for this project.

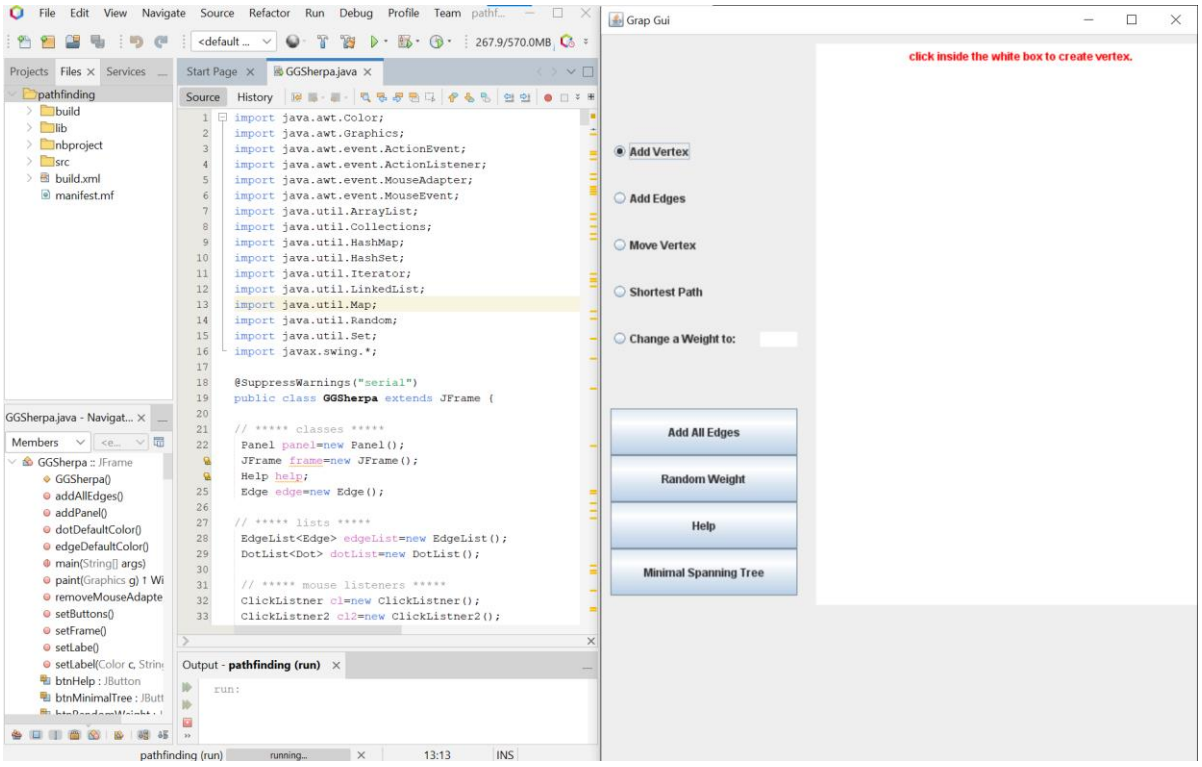


Figure 3.1: Simulation Environment

3.2 Results Analysis/Testing

3.2.1 Result_portion_1

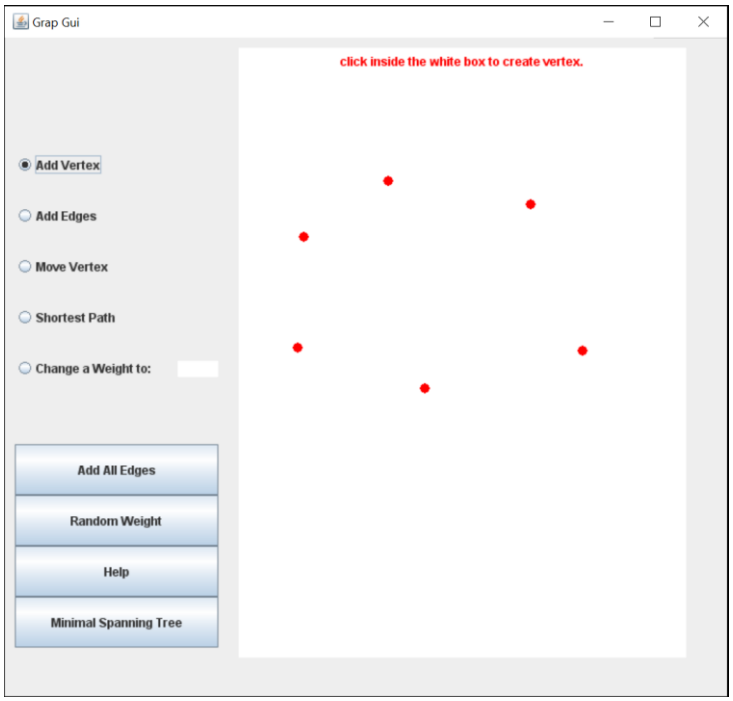


Figure 3.2: Add Vertex

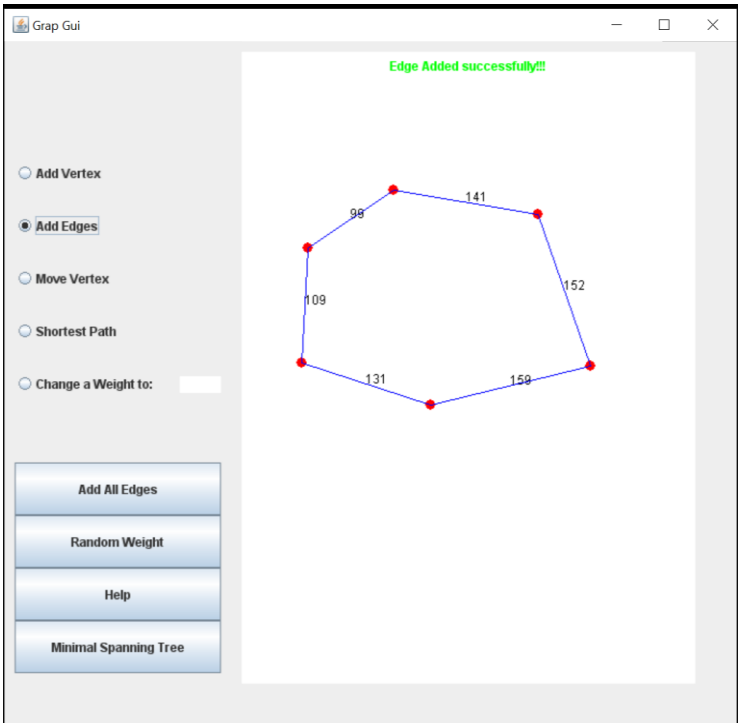


Figure 3.3: Add Edges

3.2.2 Result_portion_2

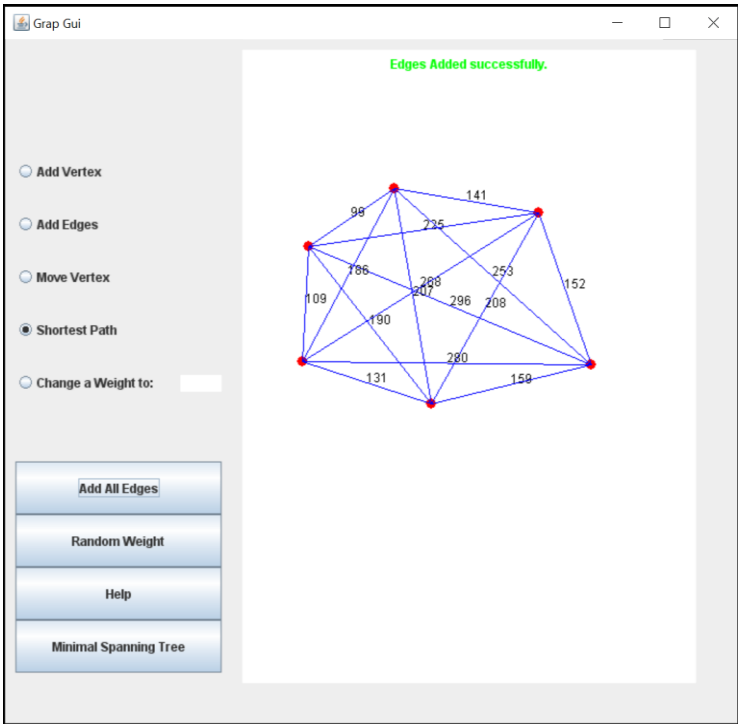


Figure 3.4: Add All Edges

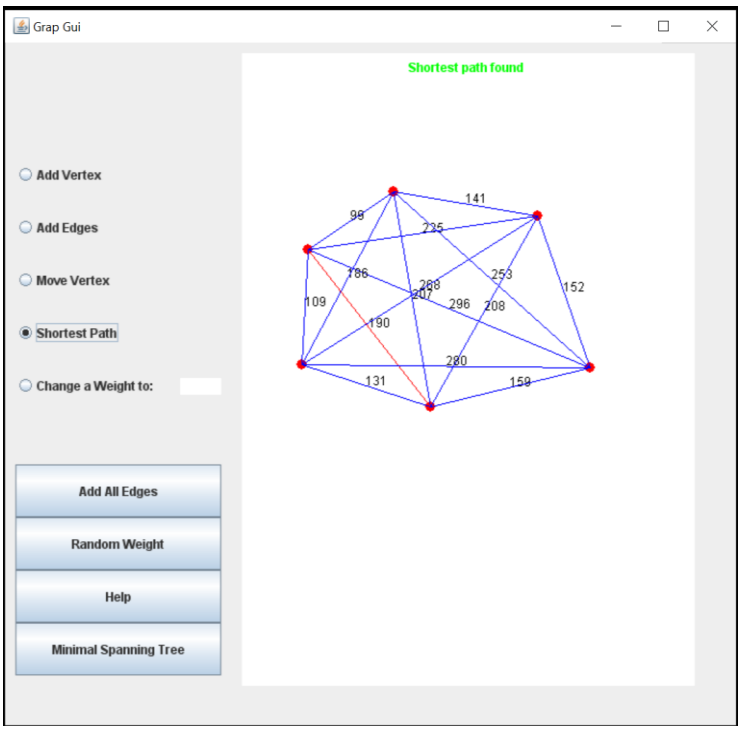


Figure 3.5: Find Shortest Path found from Graph

3.2.3 Result_portion_3

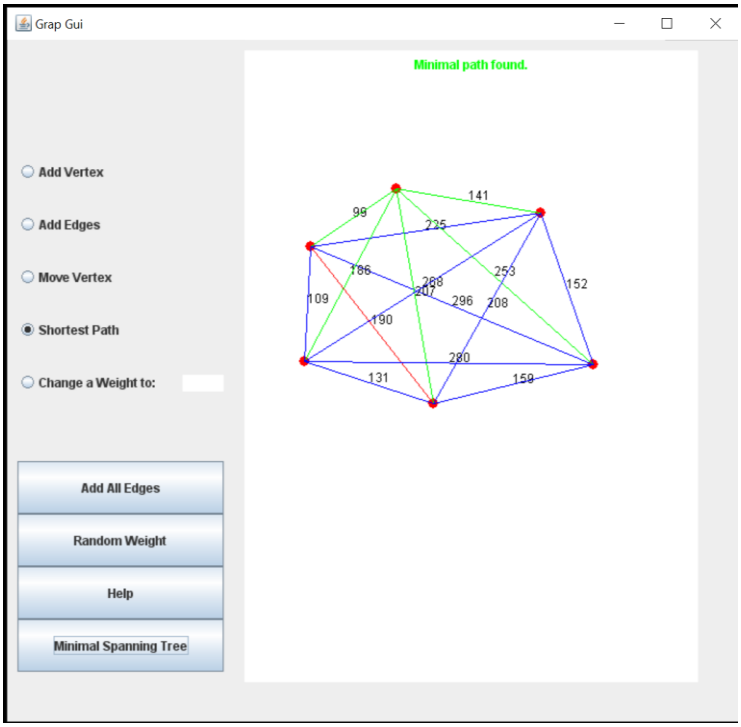


Figure 3.6: Minimal Spanning Tree found from Graph

3.3 Results Overall Discussion

Overall, evaluating the accuracy of Dijkstra's Algorithm involves comparing its output with known

[1] Optimal routes

[2] Verifying results through test cases

[3] Assessing its handling of graph constraints

[4] Analyzing sensitivity to input changes

[5] Comparing with alternative algorithms

[6] Conducting statistical analysis

These evaluations collectively provide insights into the algorithm's accuracy in finding the shortest route between two points.

Chapter 4

Conclusion

4.1 Discussion

The project's results have demonstrated the effectiveness of Dijkstra's Algorithm in accurately finding the shortest route between two points. By comparing the algorithm's output with known optimal routes, conducting test cases, and analyzing statistical data, the accuracy of the algorithm has been verified. The visualization feature has proven valuable in providing users with a visual representation of the graph and the shortest route, enhancing their comprehension and interaction with the pathfinding process.

4.2 Limitations

Despite the success of the project, there are a few limitations that should be considered:

Time Complexity:

Dijkstra's Algorithm has a time complexity of O((V + E) log V), where V is the number of vertices and E is the number of edges in the graph. This time complexity can be a limitation for very large graphs, as the algorithm may take a considerable amount of time to find the shortest route.

Memory Usage:

The algorithm requires storing information about each vertex and its distance from the source node, which can consume significant memory for large graphs. Memory usage may become a limitation when dealing with extremely large or dense graphs.

4.3 Scope of Future Work

There are several areas for future work and improvements in the Efficient Pathfinding using Dijkstra's Algorithm project:

[1] Explore integrating the pathfinding algorithm and visualization tool into real-life applications, such as map navigation systems, logistics optimization, or network routing algorithms. This integration will enhance the practical value of the project.

[2] Conduct further performance analysis, including benchmarking the algorithm against larger and more complex graphs. This analysis will provide a deeper understanding of the algorithm's scalability and efficiency.

References

[1] Dijkstra's Shortest Path Algorithm | Brilliant Math & Science Wiki

[2] Dijkstra Algorithm in Java | Baeldung

[3] Introduction to GUI Building (apache.org)

[4] Dijkstra’s Shortest Path Algorithm | Algorithms - Computer Science Engineering (CSE) (edurev.in)