UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+
Accredited University

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Mini Project Report

On

Animated Minimum Spanning Tree (MST) Visualizer using Prim's and Kruskal's Algorithm

SUBMITTED BY:

Vikash

UID:25MCI10384

CLASS: MCA (AI&ML)

SECTION: 25MAM6-A

SUBMITTED TO:

Ms. Gurpreet Kaur

Table of Content

1.Acknowledgement

I would like to express my sincere gratitude to my subject teacher for their valuable guidance, support, and encouragement throughout the development of this project. Their insightful feedback and constant motivation inspired me to work hard and improve my understanding of algorithms.

I also want to thank my classmates and friends for their constructive suggestions and constant help during the creation of this project. This project helped me gain practical knowledge about how theoretical algorithms can be implemented and visualized using Python programming.

Finally, I would like to thank my family for their constant encouragement and support during this project work.

2. Introduction

In the world of computer science, algorithms play a crucial role in solving real-life computational problems efficiently. One such important problem is finding a *Minimum Spanning Tree (MST)* in a connected, weighted graph. The MST problem is widely used in network design, such as in computer networks, road maps, and telecommunication systems.

This project, **"Animated MST Visualizer using Prim's and Kruskal's Algorithm"**, demonstrates how two famous greedy algorithms — Prim's and Kruskal's — can be used to generate a minimum spanning tree visually. The project not only computes the MST but also animates its formation step-by-step for better understanding.

3. Background

The Minimum Spanning Tree (MST) is a subgraph of a connected, weighted graph that connects all vertices together with the minimum possible total edge weight and without forming any cycles.

The two most commonly used algorithms to find MST are:

1. **Prim's Algorithm** — starts from one vertex and grows the MST by adding the smallest possible edge connecting the tree to a new vertex.
2. **Kruskal's Algorithm** — sorts all edges by weight and adds them one by one to the MST, ensuring no cycles are formed.

Both algorithms are based on the **Greedy approach**, where the locally optimal choice is made at each step to eventually reach a globally optimal solution.

4. Objective of the Project

The main objective of this project is:

- To design and implement an animated visualization tool that demonstrates how Prim's and Kruskal's algorithms construct a Minimum Spanning Tree (MST).
- To make the learning process interactive and visual.
- To help students understand how MST is formed step by step.
- To compare both algorithms visually and computationally.

5. Purpose of the Project

The purpose of this project is to provide an interactive and easy-to-understand visual tool for learning greedy algorithms, especially MST algorithms.
It serves as a practical example for algorithm analysis and implementation in Python, integrating both logic and visualization.

It also demonstrates how GUI development and algorithmic logic can be combined to create useful educational tools.

6. Technologies Used

Technology	Description
Language: Python	High-level programming language used for implementation
GUI Library: Tkinter	Used to design interactive graphical user interface
Graph Library: NetworkX	Used to generate and handle graph data structures
Visualization: Matplotlib	Used to display and animate graph structures
Threading: Python Threading	Used to handle animation without freezing the GUI

7. Requirements

Hardware Requirements:

- Processor: Intel i3 or above
- RAM: Minimum 4 GB
- Storage: 1 GB Free Space

Software Requirements:

- Operating System: Windows / Linux
- IDE: Visual Studio Code / PyCharm
- Python Version: 3.8 or higher
- Required Libraries: Tkinter, NetworkX, Matplotlib, Threading

8. System Design

The system follows the input-process-output model:

User Interface (Tkinter)

Dropdown: Select Algorithm (Prim / Kruskal)

Button: Generate Random Graph

Button: Run Algorithm (Animation)

Button: Save Graph as PNG

Text Box: Algorithm Steps Log

Output: Animated Graph Visualization

Displays total MST cost and step updates

The flow of the system:

1. User selects algorithm → Prim or Kruskal
2. Random connected graph is generated
3. Animation starts showing edges being added step-by-step
4. Total MST cost is calculated and displayed
5. User can save the result as an image

9. Algorithm Explanation

Prim's Algorithm (Greedy Method)

Steps:

1. Start from any vertex.
2. Choose the edge with the smallest weight connecting the current tree to a new vertex.
3. Add this edge and vertex to the MST.
4. Repeat until all vertices are included.

Time Complexity: $O(V^2)$ using adjacency matrix, or $O(E \log V)$ using priority queue.
Space Complexity: $O(V + E)$

Kruskal's Algorithm (Greedy Method)

Steps:

1. Sort all edges in increasing order of their weights.
2. Pick the smallest edge that doesn't form a cycle (using Union-Find).
3. Add it to the MST.
4. Repeat until MST has (V−1) edges.

Time Complexity: O(E log E)
Space Complexity: O(V)

10. Source Code

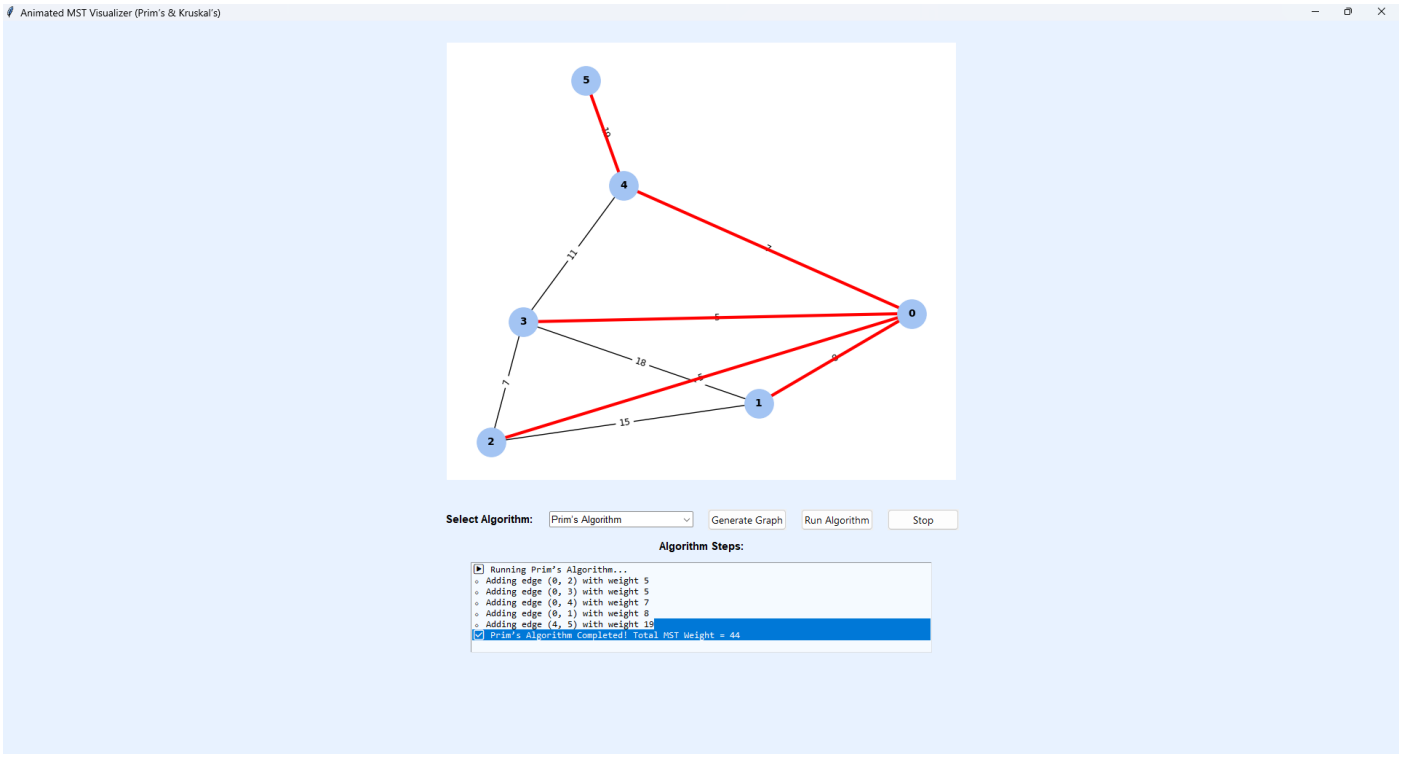
```
import tkinter as tk
from tkinter import ttk
import networkx as nx
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import random, time, threading


def create_graph():
    global G, pos
    G = nx.Graph()
    n = 6
    for i in range(n):
        G.add_node(i)
    for i in range(1, n):
        G.add_edge(i-1, i, weight=random.randint(2, 20))
    for _ in range(3):
        u, v = random.sample(range(n), 2)
        if not G.has_edge(u, v):
            G.add_edge(u, v, weight=random.randint(2, 20))
```

```
    pos = nx.spring_layout(G)


def run_algorithm(algo):
    if algo == "Prim":
        mst = nx.minimum_spanning_tree(G, algorithm='prim')
    else:
        mst = nx.minimum_spanning_tree(G, algorithm='kruskal')
    return mst
```

10. Output Screens



11. FUNCTIONALITIES

Feature	Description
Generate Graph	Creates a random connected weighted graph
Algorithm Selection	Dropdown menu to select Prim's or Kruskal's algorithm
Run Algorithm	Animates MST formation step by step
Stop Animation	Stops the ongoing animation instantly
Save Graph	Saves the current MST visualization as a PNG image
Log Box	Displays the sequence of algorithm operations

12. Conclusion

This project successfully implements and visualizes the construction of a Minimum Spanning Tree using two famous greedy algorithms: Prim's and Kruskal's.
The graphical and animated approach helps users easily understand how MSTs are formed step by step.

It bridges the gap between theoretical understanding and visual learning, making algorithm learning more interactive and interesting.

13. Future Work

Future Enhancements:

- Add side-by-side visualization of both algorithms simultaneously.
- Include animation speed control (slow/medium/fast).
- Add other graph algorithms like Dijkstra's or Bellman-Ford.
- Export algorithm log report as PDF.

14. Learning Outcomes

1. Practical understanding of MST formation.
2. Hands-on experience with Python GUI (Tkinter) and visualization.
3. Deep understanding of greedy algorithms.
4. Enhanced debugging and logical thinking skills.

.

15. References

1. GeeksforGeeks – Minimum Spanning Tree algorithms
2. Python Official Documentation (Tkinter, NetworkX, Matplotlib)
3. "Introduction to Algorithms" by Cormen et al.
4. TutorialsPoint – DAA concepts
5. YouTube tutorials on Graph Visualization using NetworkX

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+
Accredited University

Certificate

This is certified that Vikash, a student of Master of Artificial Intelligence & Machine Learning (AI & ML) has successfully completed the Minor Project titled "Animated Minimum Spanning Tree (MST) Visualizer using Prim's and Kruskal's Algorithm" under the esteemed guidance of Ms Gurpreet Kaur, Assistant Professor, University Institute of Computing (UIC), Chandigarh University.

This project was undertaken as a part of the academic curriculum and is submitted in partial fulfilment of the requirements for the MAM program. The work presented in this project is a result of group research, diligent effort, and dedication, demonstrating the student's ability to apply theoretical knowledge to practical problem-solving

The project **"Animated Minimum Spanning Tree (MST) Visualizer using Prim's and Kruskal's Algorithm"** is a **Python-based desktop application** designed to visualize how minimum spanning trees are formed in a weighted graph. It integrates **Tkinter** for the graphical user interface, **NetworkX** for graph generation and algorithm implementation, and **Matplotlib** for dynamic graph visualization. The project focuses on providing an **interactive and animated learning experience** that demonstrates the working of **Prim's and Kruskal's algorithms** step-by-step in an intuitive and visually engaging way.

I hereby confirm that this project is originally carried out by the student and has not been submitted elsewhere for the award of any other degree, diploma, or certification

Project Guide:

Ms Gurpreet kaur

Assistant Professor

University Institute of Computing

Chandigarh University