CSC-520 : Practical 1

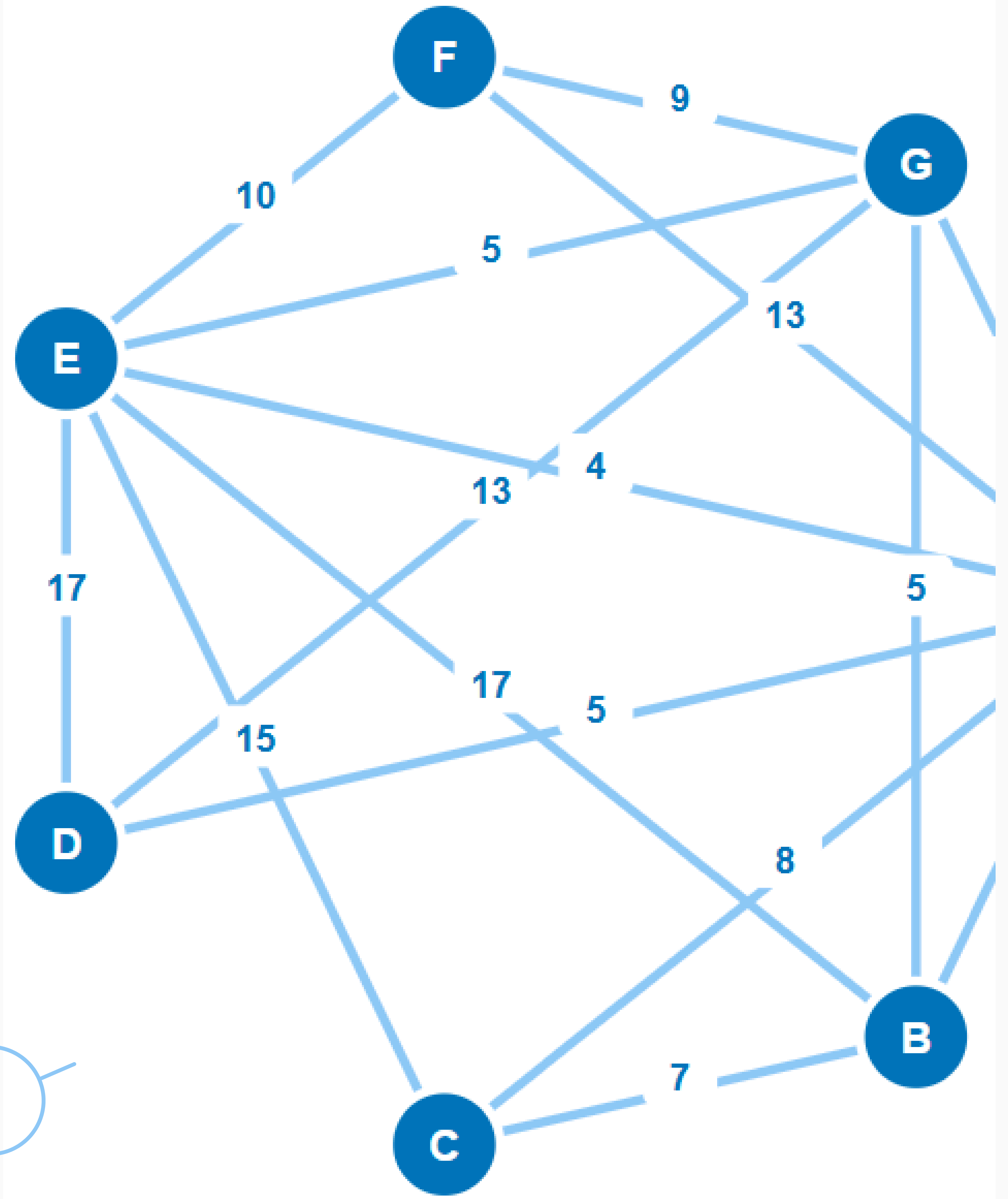
MST Visualizer

Interactive Minimum Spanning Tree Algorithm Visualizer

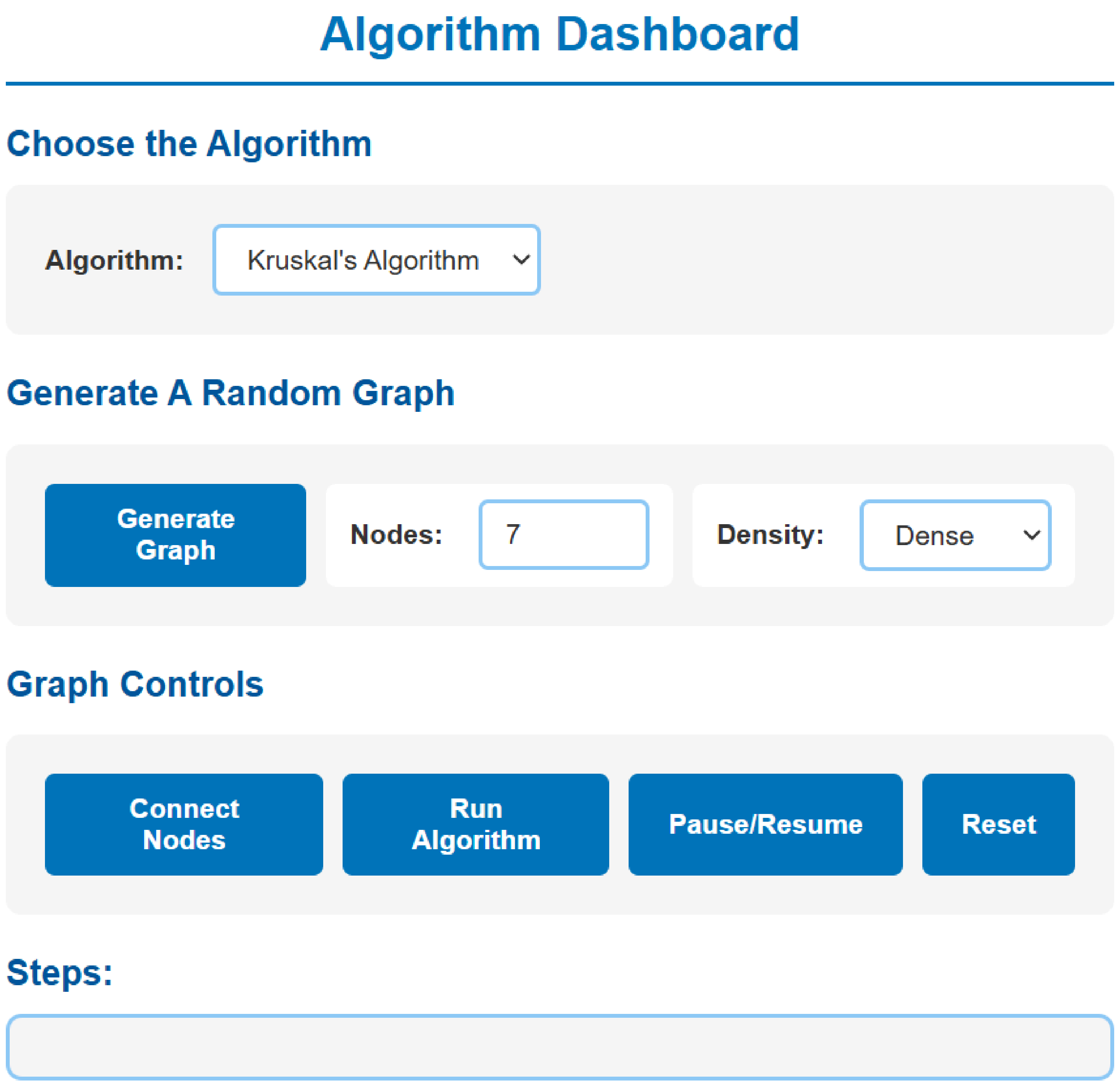
Kruskal's & Prim's Algorithms

Group No. 7

Abhishek Jaiswar - 256201
Yogiraj Bhilare - 256228
Tejas Patare - 256259



What is MST Visualizer?



Algorithm Dashboard

Choose the Algorithm

Algorithm: Kruskal's Algorithm

Generate A Random Graph

Nodes: 7

Density: Dense

Graph Controls

Steps:

Purpose

An interactive web application that visualizes Minimum Spanning Tree algorithms with step-by-step execution, helping students understand complex graph algorithms through real-time visual demonstration.

Key Features

- Interactive drag-and-drop graph creation
- Random graph generation with configurable density
- Step-by-step algorithm visualization
- Support for both Kruskal's and Prim's algorithms
- Pause/Resume control and real-time metrics

Modular Project Structure

Clean separation of concerns across three architectural layers ensures maintainability, scalability, and efficient code organization.

UI Layer

- index.html
- style.css
- dom.js
- graphDrawing.js
- modal.js

Logic Layer

- state.js
- interactions.js
- graphGeneration.js
- utils.js

Algorithm Layer

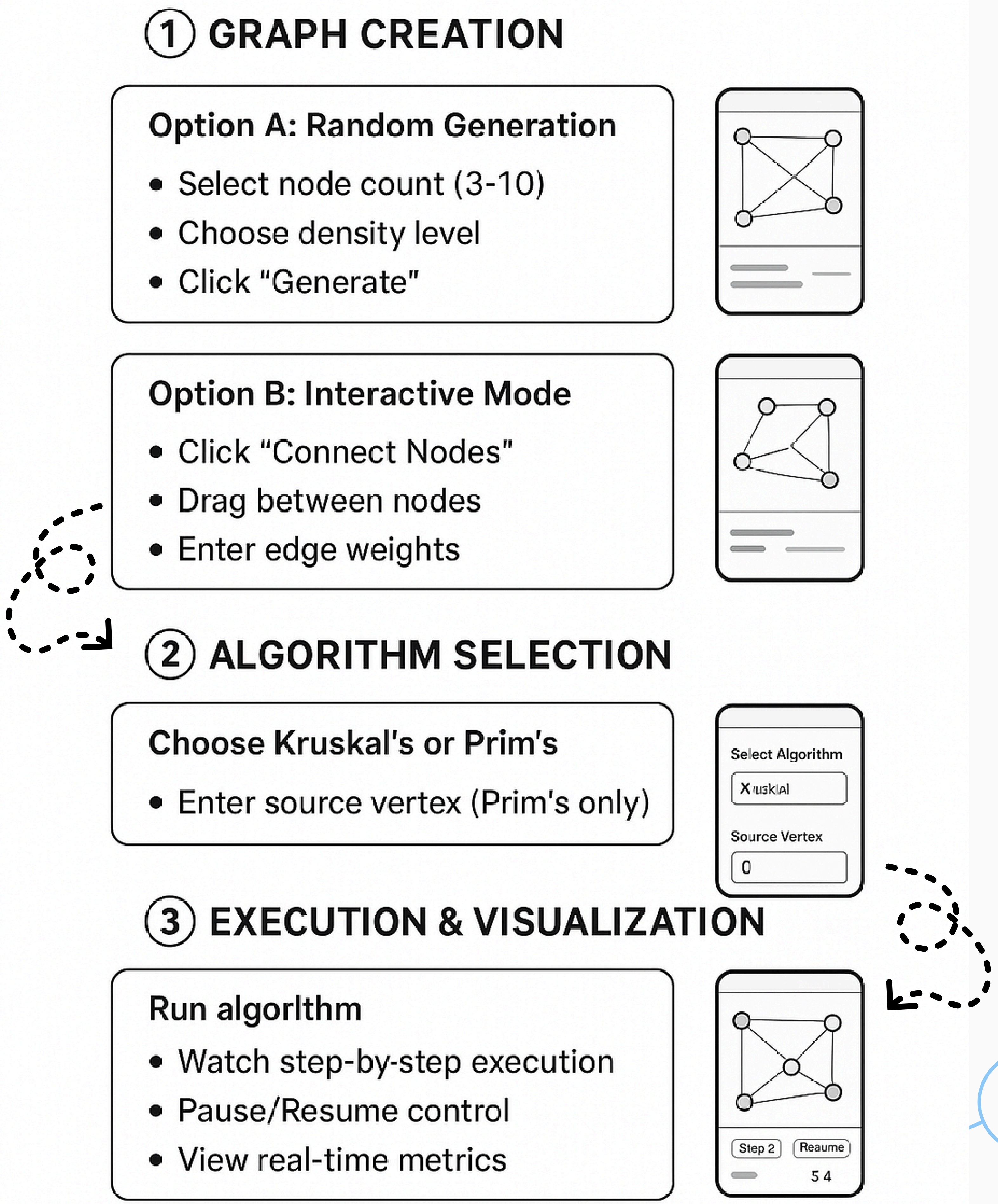
- algorithms.js
- uiControls.js

Separation of Concerns

Maintainable Code

Scalable Design

User Interaction Flow



① GRAPH CREATION

Option A: Random Generation

- Select node count (3-10)
- Choose density level
- Click "Generate"

Option B: Interactive Mode

- Click "Connect Nodes"
- Drag between nodes
- Enter edge weights

② ALGORITHM SELECTION

Choose Kruskal's or Prim's

- Enter source vertex (Prim's only)

Select Algorithm: Xuskial

Source Vertex: 0

③ EXECUTION & VISUALIZATION

Run algorlthm

- Watch step-by-step execution
- Pause/Resume control
- View real-time metrics

01

Create Graph

Choose between random generation (select node count and density) or interactive mode (drag nodes to connect and set edge weights).

02

Select Algorithm

Pick Kruskal's or Prim's algorithm. For Prim's, specify your source vertex to begin tree construction.

03

Visualize Execution

Run the algorithm step-by-step with pause/resume control. Monitor real-time metrics, edge selections, and tree growth as the algorithm progresses.

Kruskal's Algorithm

Edge-Based Greedy Approach

Kruskal's algorithm processes edges in ascending order by weight, building the MST by selectively adding edges that don't create cycles.

Algorithm Steps

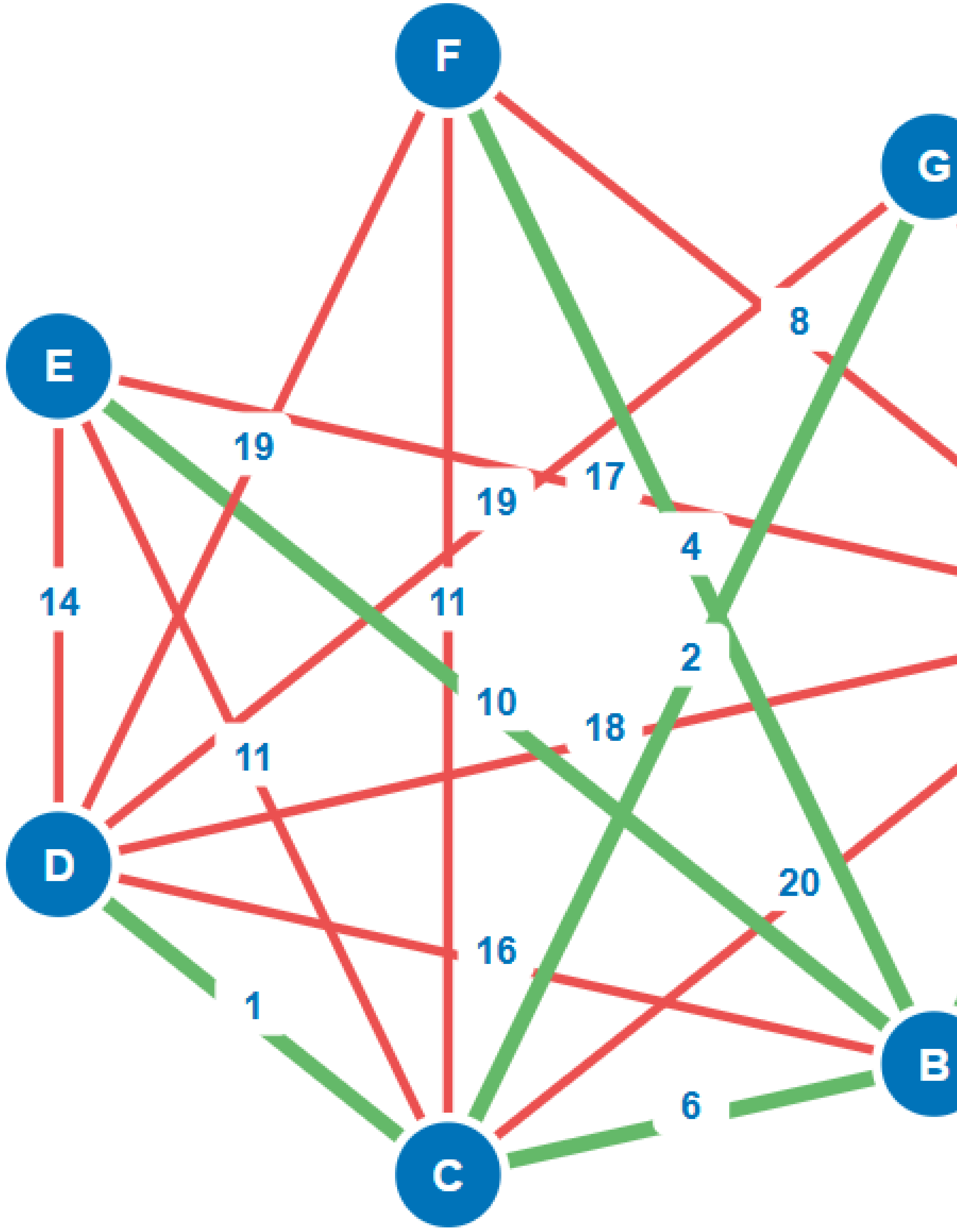
- Sort all edges by weight (ascending order)
- Examine each edge sequentially
- Use Union-Find to detect cycles
- Add edge to MST if no cycle (GREEN)
- Skip edge if cycle detected (RED)
- Continue until all vertices are connected

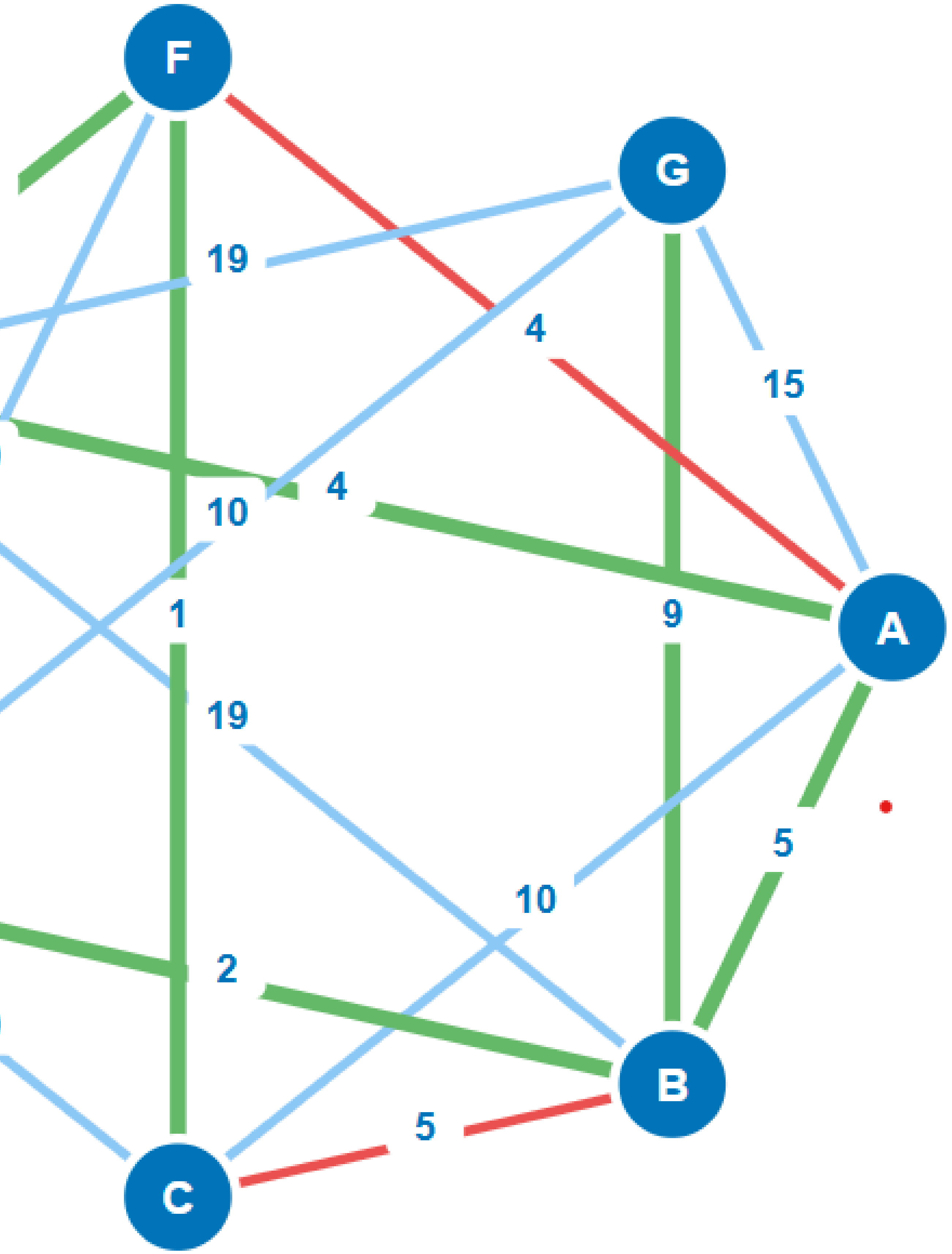
Union-Find Data Structure

- **find(x):** Locates set with path compression
- **union(a,b):** Connects two disjoint sets

Complexity Analysis

- **Time:** O(E log E)
- **Space:** O(V)
- **Best For:** Sparse graphs with fewer edges





Prim's Algorithm

Vertex-Based Greedy Approach

Prim's algorithm grows the MST from a source vertex, continuously adding minimum-weight edges connecting visited and unvisited vertices.

Algorithm Steps

- Start from designated source vertex
- Add source to visited set
- Examine all edges from visited vertices
- Select minimum weight edge to unvisited vertex
- Add edge to MST (GREEN) and mark vertex as visited
- Repeat until all vertices are visited

Priority Queue Structure

- Maintains available edges efficiently
- Sorted by edge weight
- Enables quick minimum selection

Complexity Analysis

- **Time:** O(E log V)
- **Space:** O(V)
- **Best For:** Dense graphs with many edges

Thank you

Group No. 7