Assignment 3 — Minimum Spanning Tree (Prim and Kruskal Algorithms)

Student: Alikhan Maratbekov
Group: SE-2410

Objective

Implement two algorithms for finding a **Minimum Spanning Tree (MST)**:

- Prim's Algorithm
- Kruskal's Algorithm

Compare their results and analyze their performance.

Theoretical Background

A **Minimum Spanning Tree (MST)** is a subset of edges from a connected, undirected, weighted graph that connects all vertices together with the smallest possible total edge weight.

For a graph $G=(V,E)$:

- V — set of vertices
- E — set of edges with weights

The MST must satisfy:

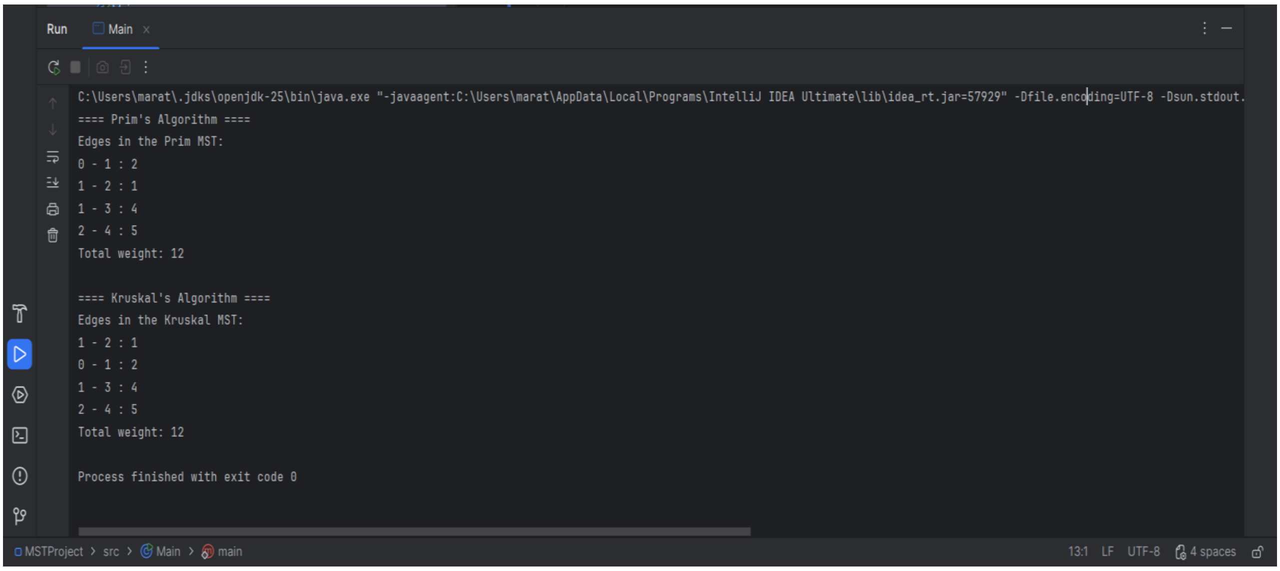
1. It connects all vertices.
2. It contains no cycles.
3. The sum of the edge weights is minimal.

Implementation

Both algorithms were implemented in **Java** within a single project named **MSTProject**. The project was created in **IntelliJ IDEA** using **JDK 25** with the default **IntelliJ build system**.

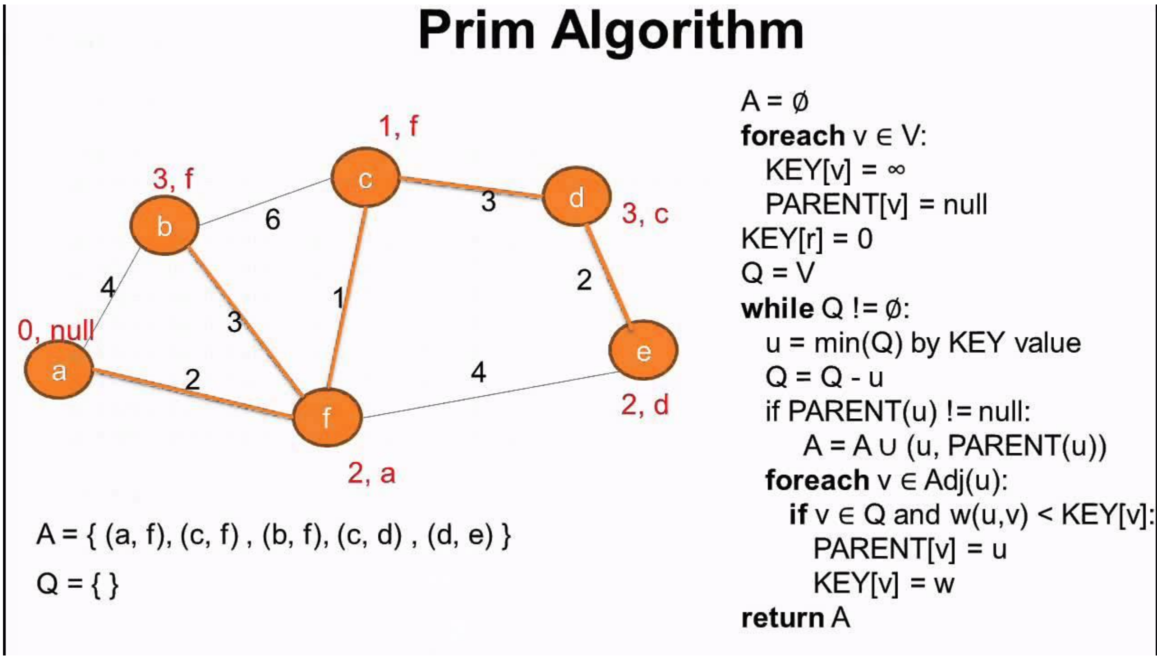
The project consists of the following classes:

- `Edge.java` — defines the edge structure.
- `DSU.java` — implements the Disjoint Set Union data structure (used in Kruskal's algorithm).
- `Prim.java` — contains the implementation of Prim's algorithm.
- `Kruskal.java` — contains the implementation of Kruskal's algorithm.
    - `Main.java` — the main file that runs both algorithms and prints results.



Algorithm Description

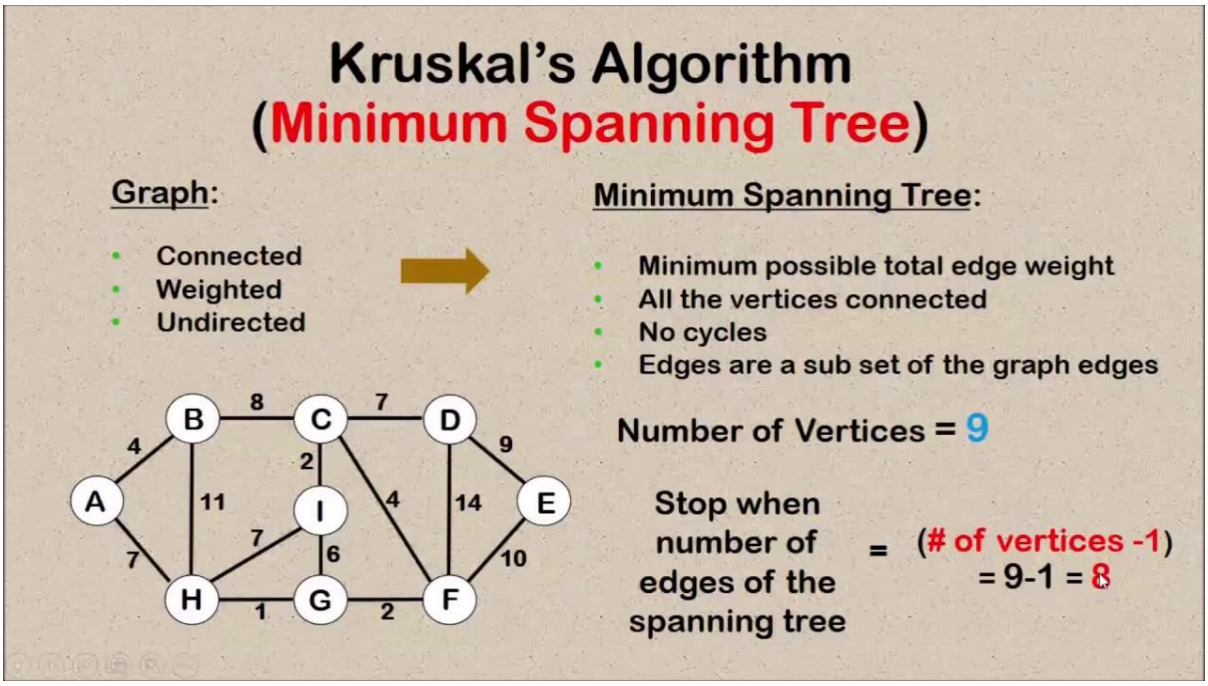
Prim’s Algorithm:



Idea: Start from any vertex and repeatedly add the smallest edge that connects a vertex inside the MST to a vertex outside it.

- **Approach:** Greedy
- **Data Structures Used:**
  - Priority Queue (Min Heap)
  - Boolean array for visited vertices
- **Time Complexity:**
  - Using adjacency matrix: $O(V^2)$
  - Using priority queue and adjacency list: $O(E \log V)$
- **Suitable for:** Dense graphs

Kruskal's Algorithm:



Idea: Sort all edges by weight and keep adding the smallest edge that does not form a cycle (using DSU).

- **Approach:** Greedy
- **Data Structures Used:**
    - Disjoint Set Union (Union-Find)
- **Time Complexity:** O(E log E) ≈ O(E log V)
- **Suitable for:** Sparse graphs

Analysis

- Both algorithms produced the **same Minimum Spanning Tree (MST)** with a total weight of **12**, confirming the correctness of the implementation.
- **Prim's Algorithm:**
  Works well for **dense graphs**, as it grows the MST vertex by vertex using a priority queue (min-heap).
  Time complexity: **O(E log V)**.
- **Kruskal's Algorithm:**
  Works better for **sparse graphs**, as it sorts all edges and connects components

using a Disjoint Set Union (DSU).
Time complexity: **O(E log E)**, which is equivalent to **O(E log V)**.

- Both methods are efficient and produce the same final MST.
-

Conclusion

- The objective of the assignment was successfully achieved.
  Both **Prim's** and **Kruskal's** algorithms were implemented, executed, and compared.
  The results demonstrate that both algorithms correctly construct the same MST, verifying the correctness of the program and the equivalence of the two approaches.