Brandon Gopar

Study Plan: Minimum Spanning Tree using Prim's Algorithm

1. Study Plan

Topic Chosen:

Minimum Spanning Trees using Prim's Algorithm

Why It was chosen:

I chose Prim's Algorithm because it demonstrates how greedy strategies can be effectively applied to real-world graph problems. It also shows how to practice using custom data structures and priority queues.

Learning Goals:

- Understand the process of Prim’s algorithm.
- Gain practical experience with graph representation using classes and structs.
- Apply smart pointers for safe and efficient memory management.
- References: GeeksforGeeks, C++ STL documentation

2. What Was Learned

Concept Summary:

- A weighted graph is a graph where edges have associated weights or costs.
- A Minimum Spanning Tree (MST) is a subset of the edges that connects all vertices with the minimum possible total edge weight.
- Prim's Algorithm builds the MST incrementally by always choosing the minimum weight edge that connects a new vertex to the growing tree. This is a greedy algorithm.

Insights:

- The use of priority queue significantly improves performance and efficiency.
- Managing custom data types and structuring classes was initially challenging but rewarding.
- Implementing edge comparison and handling bidirectional edges carefully was a key insight.

Time Complexity:

- Prim’s Algorithm (heap): O(E log V)
- Prim’s Algorithm (array): $O(V^2)$
- Space Complexity: Varies

3. Implementation Summary

Approach:

Graph representation, algorithm choice, and structure usage.

Key Files / Functions:

- Catch.hpp
- Graph.hpp
- Test.cpp

Sample Input & Output:

Graph with 4 vertices and 5 edges:
(0–1, 1), (0–2, 2), (1–2, 4), (1–3, 3), (2–3, 5)

Expected MST Output:
0 - 1 (Weight: 1)
0 - 2 (Weight: 2)
1 - 3 (Weight: 3)
Total Weight: 6

4. Reflection

What was interesting or difficult?
- Integrating smart pointers and managing object ownership in C++ was a challenge but helped my understanding of memory management.
- Using a queue with custom structs required careful setup of comparison operators.

How might this apply to future projects?
- This project has algorithm design principles useful in networking, operations research, and game development.
- It provided a strong foundation for solving optimization problems using greedy algorithms.

Would you choose another graph algorithm project?
- Yes! I'd be interested in exploring Dijkstra's shortest path or Bellman-Ford for graphs with negative weights in future projects.