Final Summary and Documentation

Algorithm Implementation Summary

Problem	Graph Algorithm	Time Complexity	Space Complexity	Application Domain	Key Features
Social Network Friend Suggestion	BFS / DFS	$O(V + E)$	$O(V)$	Social Media	Finds friends-of-friends, mutual connections
Google Maps Route Finding	Bellman-Ford	$O(V \times E)$	$O(V)$	Navigation	Handles negative weights, detects negative cycles
Emergency Response System	Dijkstra's	$O(E \log V)$	$O(V)$	Disaster Management	Fastest routes with positive weights, min-heap optimized
Network Cable Installation	Prim's MST	$O(E \log V)$	$O(V + E)$	Telecom Infrastructure	Minimum cost connectivity, priority queue implementation

Reflections: Real-World Context Influencing Algorithm Choice

Social Network Analysis → BFS

Why BFS over DFS?

- BFS naturally finds connections at specific depths (friends-of-friends at depth 2)
- More suitable for social networks where closer connections are more relevant
- Prevents deep traversal into unrelated parts of the network
- **Real-world impact**: Facebook/LinkedIn prioritize mutual friends over distant connections

Navigation Systems → Bellman-Ford

Why Bellman-Ford over Dijkstra?

- Real-world routes can have "negative weights" (time savings, toll discounts, rewards)
- Traffic conditions can create scenarios where longer paths become faster
- Safety feature: detects when routes can be made arbitrarily short (negative cycles)
- **Real-world impact**: Google Maps considers traffic patterns that create time savings

Emergency Response → Dijkstra

Why Dijkstra over Bellman-Ford?

- Emergency vehicle travel times are always positive
- Dijkstra's O(E log V) is faster than Bellman-Ford's O(V×E) for positive weights
- Min-heap implementation provides real-time performance
- **Real-world impact**: Ambulances need fastest routes without considering "negative travel times"

Network Installation → Prim's MST

Why Prim's over Kruskal?

- Prim's is more efficient for dense graphs (typical in infrastructure planning)

- Grows single connected component naturally
- Better for scenarios where we start from existing infrastructure
- **Real-world impact**: Telecom companies optimize cable costs while ensuring connectivity

Performance Insights from Experimental Profiling

Key Findings:

1. **BFS (O(V+E))**: Most scalable, minimal memory footprint
2. **Dijkstra/Prim's (O(E log V))**: Excellent balance of performance and functionality
3. **Bellman-Ford (O(V×E))**: Necessary for special cases but poor scalability
4. **Memory Usage**: All algorithms scale linearly with graph size

Practical Recommendations:

- Use **BFS** for connectivity and social network problems
- Choose **Dijkstra** for routing with positive weights
- Reserve **Bellman-Ford** for financial networks or negative weight scenarios
- Apply **MST algorithms** for network design and cost optimization