Project Proposal: Implementation of Bidirectional Dijkstra's Algorithm

Team Members: Qurba Mushtaq 08232, Hiba Shahid 08036

March 30, 2025

Paper Details

- **Title:** Bidirectional Dijkstra's Algorithm is Instance-Optimal
- **Authors:** Bernhard Haeupler, Richard Hladik, Vaclav Rozhon, Robert E. Tarjan, Jakub Tetek
- **Conference:** Proceedings of SOSA (Symposium on Simplicity in Algorithms)
- **Year:** 2025
- **DOI/Link:** `https://epubs.siam.org/doi/10.1137/1.9781611978315.16`

1 Summary

This paper provides a theoretical foundation for the efficiency of **bidirectional Dijkstra's algorithm**, proving its **instance-optimality** for shortest-path computations in both weighted and unweighted graphs. The authors demonstrate that in the adjacency list query model, no correct algorithm can outperform their implementation by more than a constant factor on any input graph.

Key contributions include:

- Formal proof of instance-optimality in weighted graphs

- Near-optimal guarantees for unweighted graphs (within factor $O(\Delta)$)
- Comparative analysis with A* search

2 Justification

2.1 Theoretical Significance

- Establishes rigorous performance bounds for fundamental algorithmic technique
- Bridges theory with practical applications in routing systems

2.2 Pedagogical Value

- Reinforces core graph algorithm concepts
- Explores advanced topics like instance optimality

2.3 Implementation Potential

- Clear pseudocode (Algorithm 2) provided
- Natural comparison points against standard algorithms

3 Implementation Feasibility

3.1 Algorithm Specification

- Complete pseudocode with:
  - Bidirectional search mechanics
  - Termination conditions
  - Path reconstruction logic

3.2 Implementation Complexity

- Standard graph structures (adjacency lists)
- Basic components (priority queues, distance tracking)
- No exotic dependencies (Python/Java/C++ compatible)

3.3 Verification Methodology

- Comparison with standard Dijkstra
- Path verification in real-world graphs
- Stress testing with edge cases

3.4 Resource Availability

- **Code:** Pseudocode available (no reference implementation)
- **Data:** Real-world graphs (Kaggle) + generated graphs

3.5 Risk Mitigation

Challenge	**Mitigation Strategy**
Termination condition complexity	Step-by-step validation
Bidirectional synchronization	Thread-safe structures
Large graph handling	Progressive testing

4 Team Responsibilities

Qurba Mushtaq	**Hiba Shahid**
Core algorithm implementation	Graph generation and dataset curation
Performance benchmarking	Results analysis and visualization
Paper analysis	Report writing

5 GitHub Repository

- **URL:** `https://github.com/HibaShahidA/Bidirectional-Dijkstra`
- **Structure:**
  - `/src` - Implementation code
  - `/data` - Graph datasets
  - `/benchmarks` - Performance scripts
  - `/docs` - Technical notes

6 Next Steps

1. Implement Algorithm 2 with termination conditions
2. Develop graph generators
3. Design comparison experiments:
   - Unidirectional Dijkstra
   - A* search
4. Analyze results