Project 1 Proposal: Unweighted Graph Pathfinding Algorithms Comparison

1. Project Scope

This project aims to compare different path-finding algorithms in unweighted graphs to determine their efficiency in finding the shortest paths. The focus will be on generating unweighted graphs, implementing multiple path-finding algorithms, and evaluating their performance based on various metrics.

2. Objectives

- Implement a graph generator to create random unweighted graphs with varying sizes and densities.
- Select and implement at least three path-finding algorithms suitable for unweighted graphs (BFS, DFS, and Bidirectional Search).
- Develop a visualization tool to observe graph traversal and shortest path computation.
- Measure algorithm performance based on nodes explored, execution time, and memory usage.
- Analyze and compare the efficiency of different algorithms in various graph structures.
- Document findings, challenges, and performance insights.

3. Requirements

- Programming Language: Python
- Development Tools: VS Code / PyCharm
- Libraries: Matplotlib (for visualization), NetworkX (for graph representation, if needed)
- Algorithms: BFS, DFS, and Bidirectional Search
- Metrics: Number of nodes explored, execution time, memory usage

4. Algorithm Selection

The following algorithms will be implemented and compared:

- Breadth-First Search (BFS): Guarantees the shortest path in unweighted graphs.
- Depth-First Search (DFS): Can explore paths quickly but does not guarantee the shortest path.
- Bidirectional Search: Runs two simultaneous searches (from the start and goal nodes), reducing search space and improving efficiency.

6. Project Plan & Timeline

Deliverable 1: Friday 3/7 (Plan and Partially Working Code)

Date	**Task**
03/01 - 03/03	Define project scope, finalize algorithms, and set up the development environment
03/04 - 03/07	Implement graph generator and BFS algorithm, conduct preliminary testing

Deliverable 2: Friday 4/4 (Fully Working Code)

Date	Task
03/08 - 03/14	Implement DFS and Bidirectional Search
03/15 - 03/21	Develop graph visualization tool
03/22 - 03/28	Optimize algorithms and measure performance
03/29 - 04/03	Compare algorithms and analyze results

Final Deliverable: Friday 5/2

Date	Task
04/04 - 04/18	Refine and optimize project components and algorithms, address bugs
04/19 - 04/25	Test project on various scenarios and datasets, conduct final testing

04/26 - 05/01	Finalize project documentation, code, comments, and findings
05/02	Submit final project, including code, documentation, and reports