PROJECT REPORT

Title:
Dynamic Graph System with Dijkstra's Algorithm and Search/Sort Utilities

Team Members:

- Mueez Waqar (231672)
- Faizan Mehdi (231257)
- Abu Huraira (231257)

Course: Data Structures and Algorithms
Semester: Spring 2025
Institute: [Your Institute Name Here]

---

1. Objective

The goal of this project is to implement a dynamic graph data structure in C++ that allows users to interactively:

- Add/remove vertices and edges
- Find the shortest path using **Dijkstra's Algorithm**
- Perform **QuickSort**, **MergeSort**, **Linear Search**, and **Binary Search** on vertex IDs

---

2. Tools & Technologies

- Language: **C++**
- Compiler: Any standard C++11+ compiler
- IDE: Visual Studio Code / Dev-C++ / Code::Blocks
- Standard Libraries: `<iostream>`, `<limits>`, `<climits>`, `<iomanip>`, `<string>`

---

3. Project Description

The project is a menu-driven console application featuring:

- **Dynamic Vertex and Edge Storage:** Vertices and their adjacency lists (edges) are managed through dynamically resizing arrays.
- **Graph Representation:** The graph is represented using an adjacency list approach, enabling efficient storage and manipulation.

- **Shortest Path Finder:** Dijkstra's algorithm is implemented from scratch to compute the minimum path between two nodes.
- **Sorting Algorithms:** QuickSort and MergeSort are applied to vertex IDs.
- **Searching:** Users can locate vertices using Linear and Binary Search.

---

4. Implementation Details

□ Classes Implemented:

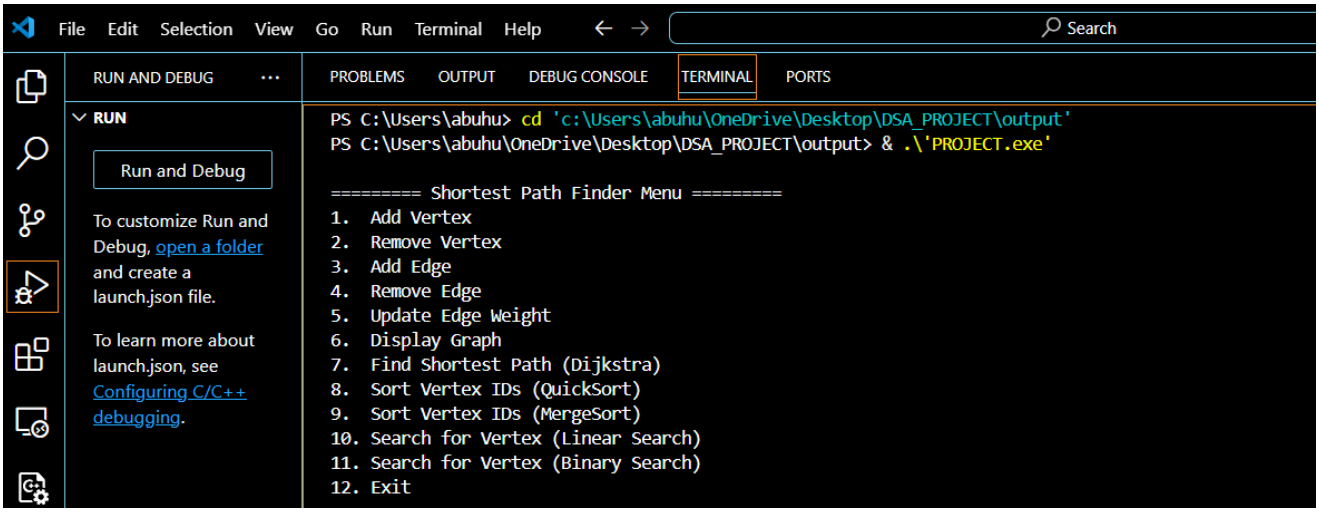
- `Edge`: Represents an edge with destination and weight.
- `DynamicEdgeArray`: Manages dynamic list of edges.
- `Vertex`: Represents a vertex and its outgoing edges.
- `DynamicVertexArray`: Manages dynamic array of Vertex pointers.
- `Graph`: Main graph class with operations to modify and display the graph.
- `MinHeap`: A binary heap used in Dijkstra's algorithm.
- `ShortestPathFinder`: Encapsulates Dijkstra's algorithm logic.

□ Algorithms Used:

- **Dijkstra's Algorithm**: For shortest path using a priority queue (min-heap).
- **QuickSort / MergeSort**: For sorting vertex IDs.
- **Linear Search / Binary Search**: For searching vertex IDs.

🛍 Features:

- Add/remove vertices or edges
- Display full graph structure
- Run Dijkstra's shortest path
- Sort vertex IDs using QuickSort or MergeSort
- Search vertex using Linear or Binary methods



Sample Dijkstra Output:

```
Shortest distance from 1 to 5 is 14.
Path: 1 -> 3 -> 5
```

6. Results

- The graph operations are functional and optimized for dynamic use.
- Dijkstra's algorithm correctly identifies the shortest path.
- Sort and search algorithms work efficiently on vertex ID lists.

---

7. Conclusion

This project demonstrates an end-to-end implementation of a dynamic graph and essential graph algorithms. It combines fundamental data structures with sorting and searching techniques, providing a rich learning experience and strong foundation in algorithm design.

---

8. Limitations & Future Work

- Current graph is **directed**; undirected support can be added.
- GUI interface or visualization could enhance usability.

- Edge deletion does not handle backward links in undirected graphs.
- BFS and DFS algorithms could be added for traversal.