GitHub Link :

LAB-BERICHT - Graphen und kürzeste Wege (Dijkstra & DFS)

Ziel: Entwurf einer Datenstruktur für gewichtete Graphen (Adjazenzliste/Matrix) und Implementierung von Pfadsuchalgorithmen (DFS, Dijkstra). Fokus auf reale Berliner BVG-Verkehrsdaten zur Ermittlung kürzester Wege.

Pre-Lab:

- **WeightedGraph Interface:** `addVertex`, `addEdge`, `getNeighbors`, `getWeight`, `getVertices`.
- **DFS:** Findet Pfade, aber nicht unbedingt die kürzesten. Besucht rekursiv Nachbarn, markiert besuchte Knoten, stoppt bei Ziel.
- **Dijkstra:** Findet kürzesten Pfad. Initialisiert Distanzen mit ∞, Startknoten mit 0. Nutzt Priority Queue, besucht Knoten mit geringster Distanz, aktualisiert Nachbarn.
- **Adjazenzliste:** Mit `Node` Objekten (`id`, `weight`, `next`) und `Map<String, Node> adjList`.

Aufgabe:

1. Implementierung von `WeightedGraph` basierend auf Adjazenzliste (HashMap, verkettete Listen).
2. `GraphReader` Klasse zum Lesen von BVG-.txt-Dateien.
3. Methode zur Ermittlung kürzester Wege (Dijkstra mit MinHeap/PriorityQueue).
4. Reale Anfrage: Kürzeste Zeiten von „S Schöneweide" zu 4 Zielstationen.

Fragen aus dem Aufgabenblatt:

- **Gewichte speichern:** In der Adjazenzliste, jedes Edge-Objekt enthält Gewichtswert (int in Sekunden).
- **Methodenzuordnung:**
  - `addVertex`, `addEdge`, `getWeight` → `WeightedGraph`.
  - `readFromFile` → `GraphReader`.
  - `findShortestPath` → `GraphAlgorithms` (oder statische Hilfsmethode).
- **Korrektheit:** Dijkstra-Implementierung gibt korrekte kürzeste Zeiten aus.