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

Name: Mhd Yazan Hammoud

Modul: Informatik 2

Datum: 19.07.2025

## ZIEL DES VERSUCHS

Ziel des Labors war es, eine Datenstruktur für gewichtete Graphen zu entwerfen, die über eine Adjazenzliste oder -matrix verwaltet wird, und darauf Algorithmen zur Pfadsuche (DFS, Dijkstra) zu implementieren. Der Fokus lag dabei auf realen Verkehrsdaten der Berliner BVG. Es sollten kürzeste (zeitlich günstigste) Wege zwischen Haltestellen gefunden werden.

## PRE-LAB

### P1. Interface für gewichtete Graphen

```
public interface WeightedGraph {
   void addVertex(String id);
   void addEdge(String from, String to, int weight);
   List<String> getNeighbors(String vertex);
   int getWeight(String from, String to);
   Set<String> getVertices();
}
```

### P2. DFS – Tiefensuche zur Pfadsuche

DFS kann verwendet werden, um einen Pfad zu finden, jedoch nicht unbedingt den kürzesten, da sie nicht die Pfadlänge berücksichtigt.

Algorithmus-Skizze:
- Rekursiv alle Nachbarn besuchen
- Markiere besuchte Knoten
- Stoppe, wenn Ziel erreicht

### P3. Dijkstra – Algorithmus für den kürzesten (günstigsten) Pfad

- Initialisiere alle Distanzen mit ∞, Startknoten mit 0
- Verwende eine PriorityQueue
- Besuche stets den Knoten mit geringster Distanz
- Aktualisiere Nachbarknoten, wenn neuer Pfad kürzer ist

### P4. Adjazenzliste mit verketteten Listen

```
class Node {
   String id;
   int weight;
```

```
    Node next;
}
```

Map<String, Node> adjList;

Benötigte Methoden: addVertex, addEdge, getNeighbors, getWeight

## AUFGABE

1. Datenstruktur WeightedGraph implementieren – basierend auf einer Adjazenzliste mit HashMap und verketteten Listen.

2. Parser-Klasse GraphReader – liest .txt-Dateien im BVG-Format und ruft addVertex und addEdge auf.

3. Methode zur Ermittlung kürzester Wege – Dijkstra-Algorithmus mit MinHeap oder PriorityQueue.

4. Reale Anfrage: Kürzeste Zeiten von „S Schöneweide“ zu 4 Zielstationen:

[[60068201511, 660], [60066102852, 1224], [60053301433, 1950], [60120003653, 504]]

## FAKULTATIV (optional)

- Methode printAllAtDistance(graph, start, steps)
- Liste aller Reisezeiten von S Schöneweide
- Mehrere minimale Wege (Backtracking & Pfadvergleich)

## FRAGEN AUS DEM AUFGABENBLATT

Ex. 1: Wie speicherst du die Gewichte?

→ In der Adjazenzliste: jedes Edge-Objekt enthält einen Gewichtswert (int) in Sekunden.

Ex. 3: Zu welcher Klasse gehören diese Methoden?

- addVertex, addEdge, getWeight → WeightedGraph
- readFromFile → GraphReader
- findShortestPath → GraphAlgorithms oder als statische Hilfsmethode

Korrektheit:

→ Die Dijkstra-Implementierung gibt die korrekten kürzesten Zeiten aus (siehe Vergleich mit Beispieldaten).