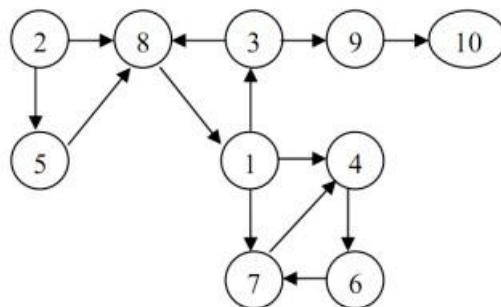


Algorithmen und Datenstrukturen

Graphen

Aufgabe 1: Manuelle Graphen-Analyse [2 Punkte] Gegeben ist folgender Graph G:



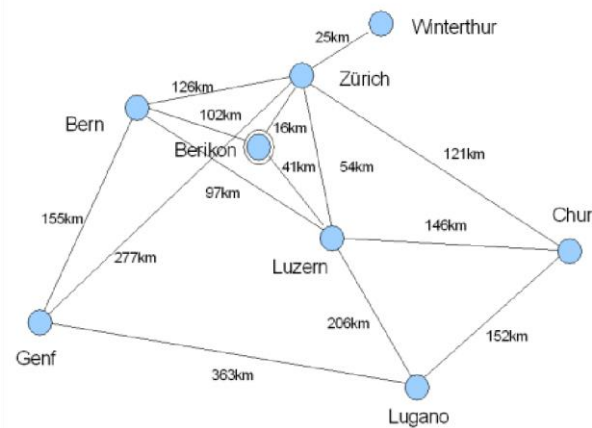
Führen Sie für den Graphen G die Breitensuche durch, beginnend im Knoten 1.
Geben Sie die Reihenfolge der Knotenbesuche an. "Besuchen" Sie dabei für einen Knoten seine Nachbarn immer in aufsteigender Reihenfolge (also von Knoten 1 aus zunächst 3, dann 4, dann 7).

Aufgabe 2: Graph-Repräsentation [2 Punkte]

Graphen kann man als Adjazenz-Matrix oder mit Hilfe von Adjazen-Listen darstellen. In dieser Aufgabe untersuchen Sie, welche Darstellungsform für welche Anwendung sinnvoll ist.

- Geben Sie mindestens drei verschiedene Operationen (wie z.B. einfügen eines Knoten, Löschen einer Kante etc.) an, die in einer Adjazenzmatrix schneller sind als in Adjazenz-Listen. Begründen Sie, warum.
- Geben Sie umgekehrt mindestens drei Operationen an, die mit Adjazenz-Listenschneller sind.

Aufgabe 3: Erzeugen des Graphen [3 Punkte]



Lesen Sie die Distanzen zwischen den Städten aus der Datei Swiss.txt ein und erstellen Sie einen entsprechenden **ungerichteten** Graphen. Für diesen Zweck soll eine Klasse RouteServer erstellt werden, die den CommandExecutor implementiert – wie gehabt.

Hinweise zur Implementation des RouteServer:

- Sie können das Interface Graph bzw. die Klasse AdjListGraph verwenden.
- Da der Knoten, die für den Algorithmus notwendigen Felder (dist, mark und prev) nicht enthält, müssen Sie den DijkstraNode verwenden.
- Definition des Graphen: `Graph<DijkstraNode, Edge> graph = new AdjListGraph<DijkstraNode, Edge>(DijkstraNode.class, Edge.class);`
- addEdge Methode des Graphen verwenden.
- Eine ungerichtete Kante erhalten Sie, indem Sie gerichtete in beide Richtungen einfügen.

Aufgabe 4: Kürzeste Strecke [3 Punkte]

Traversieren Sie den Graphen und bestimmen Sie die kürzeste Verbindung nach dem in Folien beschriebenen Dijkstra-Algorithmus (Methode `private void dijkstra(String from, String to)` in RouteServer implementieren). Geben Sie die gefundene Strecke aus.

Hinweise:

- Den Algorithmus (in Pseudocode) aus den Folien verwenden.
- Die Pseudo-Anweisung `for all` kann einfach in die entsprechende Java 5 Anweisung übersetzt werden.
- Mit der Graph-Methode `findNode("Winterthur")` kann der Start-Knoten gefunden werden.
- Als PriorityQueue nehmen Sie die JDK Queue `java.util.Queue<DijkstraNode> pq = new java.util.PriorityQueue<DijkstraNode>();`
- Die Ordnung/Priorität wird durch das Comparable bzw `compareTo` bestimmt (DijkstradeNode muss korrekt ergänzt werden).
- Im RouteServer nach dem Einlesen der Daten und erstellen des Graphen:

```
...
String start = "Winterthur";
String goal = "Lugano";
dijkstra(start, goal);
<Ausgabe des Resultates>
```

- Resultatausgabe: Vom Zielknoten (Lugano) mittels den gesetzten prev Verweisen zurück bis nach Winterthur.