Aufgabe 5 – Breitensuche

Vorbereitung

- Machen Sie sich anhand der Unterlagen aus der Vorlesung mit der Breitensuche in Graphen vertraut.
- Überlegen Sie sich wie die Hauptschleife in der Breitensuche beendet werden kann, wenn eine Maximalzahl von zu durchlaufenden Kanten erreicht ist.

Aufgabenbeschreibung

1. Entwickeln Sie die Java Klasse **`Edge`** für gerichtete Kanten nach dem folgenden Muster:

```
class Edge {
    public Node dest;
    public double weight;

    public Edge(Node d, double w) { ... }
}
```

Entwickeln Sie basierend auf dieser Klasse die Java Klasse **`Node`** für Knoten in einem gerichteten Graphen nach folgendem Muster:

```
class Node {
    public String name;
    public List<Edge> neighbors;
    boolean visited;
    Node prev;
    double dist;

    public Node(String n) { ... }
}
```

Jeder Knoten besitzt einen Bezeichner **`name`** und eine Liste **`neighbors`** von Kanten zu seinen Nachfolgern. Die anderen Member der Klasse werden für die Breitensuche verwendet und könnten ergänzt oder verändert werden.

Entwickeln Sie nun damit die Java Klasse **`DirectedGraph`** für einen gerichteten und gewichteten Graph nach dem Muster:

```
class DirectedGraph {
    public static final double INFINITY =
        Double.MAX_VALUE;
    private Map<String,Node> nodes =
        new HashMap<String,Node> ();

    public static
        DirectedGraph readGraph(string file) { ... }
    public boolean
        BFS(String start, String dest, int max) { ... }
    public void printPath(String dest) { ... }
}
```

Mit der Methode **`readGraph()`** wird ein Graph durch Einlesen aus der Datei **`file`** aufgebaut. Beachten Sie, dass in der Datei zeilenweise nur Kanten in der Form

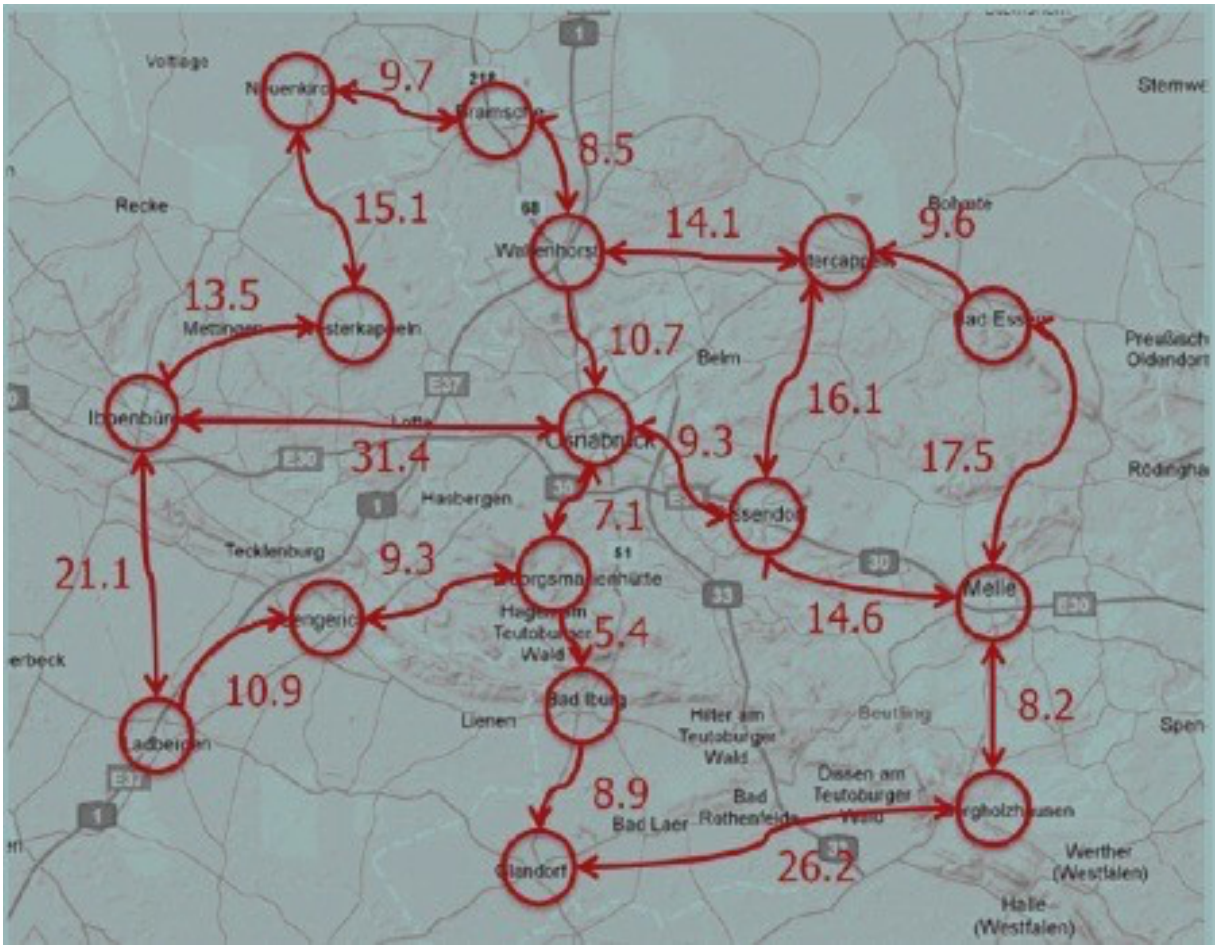
```
Knoten_A  Knoten_B  8.56
```

enthalten sind. Beim Einlesen muss auch geprüft werden, ob die Knoten bereits existieren oder neu angelegt werden müssen.

Die Methode **`BFS()`** sucht einen Pfad vom angegebenen Startknoten **`start`** zum Zielknoten **`dest`**, der über maximal **`max`** Kanten führt. Existiert ein solcher Pfad ist der Return-Wert **`true`**, ansonsten **`false`**.

Die Methode **`printPath()`** gibt den vorher ermittelten Pfad durch Rückverfolgung aus.

2. Schreiben Sie eine Java Programm, dass diese Klasse testet. Verwenden Sie dazu folgenden Graphen:



In dem Graphen sind die meisten (aber nicht alle) Kanten bidirektional, wobei hier eine Kante mit zwei Pfeilenden gezeichnet ist. Dieser Graph liegt in der Datei **`OS_Map.txt`** vor.

Abgabe

Laden Sie das Programm ihrer Gruppe im Stud.IP-Abgabebereich der Lehrveranstaltung für das Praktikum hoch. Bilden Sie dazu aus Ihrem Projekt ein ZIP-Archiv, dass nur die Quelldateien enthält. Dieses Archiv muss zu Beginn Ihres nächsten Praktikumstermins in Stud.IP vorliegen.

Bitte beachten Sie die allgemeinen Hinweise zum Praktikum.