Programmierkurs 2
Sommersemester 2023

6. Übungsblatt (BLOCK 2)

Abgabe bis 19.06.2023

Legen Sie in Ihrem git-Ordner einen Unterordner **blatt6** an. Checken Sie dann den Code für das Übungsblatt dort ein.

Graphen

Implementieren Sie die Graphdatenstruktur Adjazenzliste. Legen Sie dazu zunächst eine Knoten- und eine Kantenklasse an. Die Knotenklasse sollte die KnotenID, sowie die Koordinaten des Knotens beinhalten und eine Liste der eingehenden und eine Liste der ausgehenden Kanten. Die Kantenklasse sollte die Start- und Endknoten ID speichern, sowie das Gewicht der Kante. Lesen Sie dann den Graph aus dem EingabeFile ein und speichern Sie ihn als Adjazenzliste ab. Der Graph hat n Knoten und m Kanten und ist folgendem Format gegeben:

```
<n>
<m>
<LatitudeKnoten 0> <LongitudeKnoten 0>
<LatitudeKnoten 1> <LongitudeKnoten 1>
...
<LatitudeKnoten n-1> <LongitudeKnoten n-1>
<StartknotenIDKante 0> <EndknotenIDKante 0> <GewichtKante 0>
<StartknotenIDKante 1> <EndknotenIDKante 1> <GewichtKante 1>
...
<StartknotenIDKante m-1> <EndknotenIDKante m-1> <GewichtKante m-1>
```

Latitude und Longitude sind Gleitkommazahlen, alles andere nicht-negative Ganzzahlen. Beachten Sie außerdem, dass die Kanten gerichtet sind (vom Startknoten zum Endknoten).

Fügen Sie dann Ihrer Graphklasse die Funktion getDegree(int ID) hinzu, welche den Gesamtgrad des Knotens mit der übergebenen ID zurückgibt. Schreiben Sie dann in AusgabeFile1 die Knotengrade gemäß der Einlesereihenfolge bzw. ID, jeweils einen Wert pro Zeile.

Implementieren Sie dann die Funktion getNeighbors(int ID), die zum Knoten mit der entsprechenden ID zugehörigen adjazenten Knoten gemäß der ausgehenden Kanten zurückgibt, sowie getNeighborhood(int ID, int k), die für $k = 0$ einfach nur die Eingabe ID zurückgibt, für $k = 1$ dieselbe Menge wie getNeighbors(int ID) und für $k \geq 2$ die Menge der Nachbarn der Knoten aus dem Aufruf getNeighborhood(ID, $k-1$), allerdings ohne Knoten aus den Mengen getNeighborhood(ID, k') für $0 \leq k' \leq k-1$. Die entsprechende Knoten ID zum Aufruf sowie k werden als Parameter übergeben. Die berechneten Knotenmengen sollen dann in AusgabeFile2 geschrieben werden. Dabei soll in Zeile i die Rückgabemenge von getNeighborhood(ID,i) stehen, jeweils durch Leerzeichen separiert für $i = 0, \ldots, k$.

Speichern Sie Ihren Code in der Datei Graph.java ab.

COMPILE

```
javac Graph.java
```

AUFRUF:

```
java Graph <EingabeFile> <ID> <k> <AusgabeFile1> <AusgabeFile2>
```

TEST RUNS (müssen durchlaufen, korrekte Ausgaben im ILIAS zum Vergleich):

```
java Graph graph-klein.txt 1 3 grad-klein.txt nachbarn-klein.txt
```

FINAL RUNS (sollen in weniger als 5 Minuten durchlaufen):

```
java Graph graph-gross.txt 12345 10 grad-gross.txt nachbarn-gross.txt
```