Programmierkurs 2
Sommersemester 2023

7. Übungsblatt (BLOCK 2)

Abgabe bis 26.06.2023

Legen Sie in Ihrem git-Ordner einen Unterordner **blatt7** an. Checken Sie dann den Code für das Übungsblatt dort ein.

Kürzeste Wege

Implementieren Sie den Algorithmus von Dijkstra. Verwenden Sie als Prioritätswarteschlange eine eigene **MinHeap** Implementierung. Verzichten Sie in Ihrer Implementierung auf die decreasyKey() Operation (es wird also nur insert() und extractMin() benötigt). Wenn der Distanzwert eines Knotens w nach Relaxierung einer Kante sinkt, fügen Sie ihn einfach erneut mit der verringerten Distanz in den MinHeap ein. Achten Sie darauf, dass der MinHeap die Distanzwerte und nicht die KnotenIDs als Schlüssel benutzt, aber immer klar ist, welcher Schlüssel zu welchem Knoten gehört.

Das Einleseformat des Graphs ist dasselbe wie auf dem letzten Übungsblatt. Dieses Mal sind aber die Kantengewichte von Bedeutung. Beim Aufruf wird zunächst der Graphfile übergeben, und dann, im zweiten Eingabefile, eine Liste von KnotenID-Paaren (eine Paar pro Zeile, separiert durch Leerzeichen). Es soll dann mit Hilfe des Dijkstra-Algorithmus der kürzeste Pfad vom ersten zum zweiten Knoten in jeder Zeile berechnet und dessen Kosten in den Ausgabefile geschrieben werden (wieder ein Wert pro Zeile). Sollte der Zielknoten nicht vom Startknoten aus erreichbar sein, soll -1 ausgegeben werden.

Speichern Sie Ihren Code in Datei **Dijkstra.java** ab.

COMPILE

```
javac Dijkstra.java
```

AUFRUF:

```
java Dijkstra <EingabeFile1> <EingabeFile2> <AusgabeFile>
```

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

```
java Dijkstra graph-klein2.txt st-klein.txt distanzen-klein.txt
```

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

```
java Dijkstra graph-gross.txt st-gross.txt distanzen-gross.txt
```