Datenstrukturen PVL 7– SS2023

TECHNISCHE UNIVERSITÄT
IN DER KULTURHAUPTSTADT EUROPAS
CHEMNITZ

Professur Softwaretechnik

7| 2023

Administratives

Abgabe Termin

17.07.2023 - 23:59

Abgabe

Ihre Abgabe darf entweder in Java oder Python geschehen. Nutzen Sie für die Abgabe das Code Template aus dem OPAL-Bereich "Prüfungsvorleistung 7". Nichtbeachtung des Templates kann zu Punktabzügen führen! Sie dürfen zusätzliche Klassen und Methoden anlegen, um die Aufgabe zu lösen.

Laden Sie alle .java oder .py Dateien, die zum Ausführen Ihres Codes nötig sind, im OPAL-Bereich "Prüfungsvorleistung 7" hoch. Bitte geben Sie keine ganzen Ordner ab, die Quelldateien reichen. Hinweis für Java: Von uns vorgegebene Interfaces brauchen Sie ebenfalls nicht mit abzugeben.

Erlaubte Klassen

Sie dürfen Klassen und Methoden aus der Standardbibliothek von Java bzw. Python nutzen. Bitte benutzen Sie keine Bibliotheken von Dritten.

Information zu Plagiaten

Es ist **erlaubt**, Lösungen aus dem Internet in Ihre Abgabe zu integrieren. Bitte **markieren** Sie jeglichen Code, der nicht von Ihnen selbst stammt, indem Sie einen Kommentar mit der Quelle des Codes (URL ist ausreichend) an die entsprechende Stelle setzen. Dies soll verhindern, dass Ihre Lösung fälschlicherweise als Plagiat erkannt wird.
Da es sich hierbei um eine Prüfungsvorleistung handelt, ist es notwendig, dass Sie die Aufgaben selbständig bearbeiten. Daher werden alle Abgaben untereinander auf Plagiate überprüft. Sollten Plagiate erkannt werden, gilt die Abgabe von **sämtlichen** beteiligten Parteien als ungültig.

Kompilierbarkeit und Clean Code

Es werden nur Abgaben gewertet, welche sich in einem ausführbaren Zustand befinden!
Achten Sie außerdem auf die Leserlichkeit des Codes. Unverständliche oder unangemessene Bezeichner von Klassen, Variablen, etc. können zu Punkteabzügen führen.
Beachten Sie, dass Methodennamen in Java in camelCase und in Python in snake_case geschrieben werden, auch wenn das Aufgabenblatt nur die Java-Variante zeigt.

Fragen

Da es sich um eine Prüfungsvorleistung handelt, ist es wichtig, dass jeder Teilnehmer Zugang zu allen notwendigen Informationen hat. Deswegen bitten wir Sie, Fragen direkt ins Forum, im Thread "PVL 7: Fragen" zu stellen. Bitte beachten Sie, dass Sie ihre Fragen möglichst frühzeitig stellen. Kurz vor der Deadline können wir keine rechtzeitigen Antworten mehr garantieren.

Aufgabe

Implementieren Sie in der Klasse **DirectedGraph** einen gewichteten, gerichteten Graphen. In Java ist dafür das Interface **PVL7** vorgegeben.

Ein Graph sei dabei ein Tupel von Knoten V und Kanten E : G=(V,E). Der Graph soll dabei N Knoten besitzen, indiziert von 0 bis N-1. Eine Kante ist eine einseitig nutzbare Verbindung zwischen zwei Knoten. Implementieren Sie folgende Methoden, um einen gewichteten, gerichteten Graphen zu realisieren:

```
Konstruktor()
```

Einen Konstruktor in Ihrer jeweiligen Sprache, welcher keine Argumente erhält und die Klasse für die Verwendung vorbereitet.

```
newNode()
```

Welche einen neuen Knoten dem Graphen hinzufügt und dessen Nummer zurückgibt.

```
setEdge(int from, int to, int weight)
```

Welche eine Kante zwischen den Knoten FROM und TO mit dem Gewicht WEIGHT erstellt. Geben Sie "true" zurück, sollte die Kante erfolgreich erstellt worden sein, andernfalls geben Sie "false" zurück.

```
getEdges()
```

Welche eine Liste von allen Kanten des Graphen zurückgibt:
Am i-ten Index der äußeren Liste soll eine Liste von Kanten repräsentieren, welche Kanten von Knoten i ausgehen. Jede einzelne Kante wird dabei durch eine Integer-Liste bestehend aus dem Index des Zielknotens und dem Gewicht der Kante dargestellt.

```
hasNegativeCircle()
```

Ein negativer Kreis sei ein Kreis, dessen Kantengewichte aufsummiert kleiner 0 sind. Geben Sie "true" zurück, falls es einen negativen Kreis gibt, ansonsten "false".

```
shortestPath(int source, int destin)
```

Geben Sie eine Liste von Kanten zurück, welche den *Kantenzug* (Weg, in welchem Dopplungen von Knoten erlaubt sind) mit dem geringsten Gewicht von SOURCE nach DESTIN beschreibt. Beachten Sie, dass der Kantenzug nicht unendlich 'kurz' werden darf. Sollte es keinen gültigen kürzesten Kantenzug oder gar keinen Kantenzug geben, geben Sie eine leere Liste zurück.

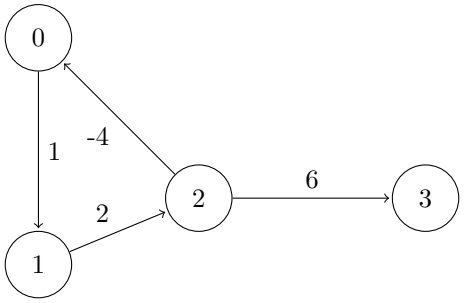
Bonusaufgabe (2 Punkte):

```
universalSink()
```

Eine universelle Senke ist ein Knoten. Dieser Knoten kann von allen anderen Knoten erreicht werden (über einen Weg), doch selbst erreicht er keinen anderen Knoten. Geben Sie den Index der universellen Senke an. Sollte es keine universelle Senke geben, geben Sie -1 zurück.

Beispiel

(a)

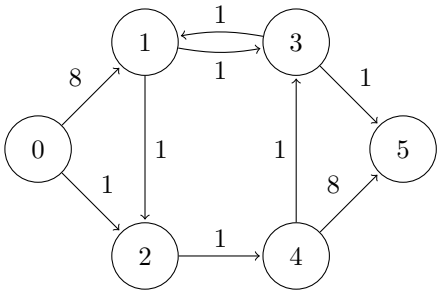



```
PVL7 graph_a = new Graph();
graph_a.newNode(); // 0
graph_a.newNode(); // 1
graph_a.newNode(); // 2
graph_a.newNode(); // 3
graph_a.setEdge(0,1,1); //true
graph_a.setEdge(1,2,2); //true
graph_a.setEdge(2,0,-4); //true
graph_a.setEdge(2,3,6); //true

graph_a.getEdges();
//[
//[[1,1]],
//[[2,2]],
//[[0,-4], [3,6]],
//[]
//]
graph_a.hasNegativeCircle(); // true
graph_a.shortestPath(0,2); // []

//Bonusaufgabe
graph_a.universalSink(); // 3
```

(b)




```
PVL7 graph_b = new Graph();
graph_b.newNode(); // 0
graph_b.newNode(); // 1
graph_b.newNode(); // 2
graph_b.newNode(); // 3
graph_b.newNode(); // 4
graph_b.newNode(); // 5
graph_b.setEdge(0,1,8);
graph_b.setEdge(0,2,1);
graph_b.setEdge(1,2,1);
graph_b.setEdge(1,3,1);
graph_b.setEdge(2,4,1);
graph_b.setEdge(3,1,1);
graph_b.setEdge(3,5,1);
graph_b.setEdge(4,3,1);
graph_b.setEdge(4,5,8);

graph_b.hasNegativeCircle(); // false
graph_b.shortestPath(0,5); // [[2,1], [4,1], [3,1], [5,1]]

//Bonusaufgabe
graph_b.universalSink(); // 5
```