Markus Nebel/Alexander Sczyrba/Dirk Frettlöh 13.2.2024

Abschlussprojekt Prinzipien der Programmierung

Sie dürfen eines der beiden folgenden Projekte auswählen. Sie bearbeiten eines der Projekte und geben ihre Programmdaten bis zum Sonntag, 3.3.2024 um 23:59 ab. Die Abgabe erfolgt wie bei den Übungen auch per Moodle.

Projekt A ist thematisch im Bereich der Verarbeitung großer Datenmengen angesiedelt, Projekt B im Bereich der Algorithmik. Beides sind Algorithmen auf Graphen, wie sie in dem PdP-Praktikum und "Mathe für Informatik I" gezeigt wurden.

Teil (1) besteht bei beiden Projekten darin, Daten aufzuarbeiten und in eine für Teil (2) nützliche Form zu bringen. Teil (2) besteht bei beiden Projekten darin, auf diesen Daten einen Algorithmus laufen zu lassen, der das jeweils gestellte Problem löst. Teil (1) wird heute bekanntgegeben, Teil (2) am 16.2.2024. Für Teil (1) brauchen Sie in beiden Fällen die Methoden aus dem PdP-Praktikum (13.2.-16.2.2024), für Teil (2) brauchen Sie jeweils das, was Sie während des Semesters in den Vorlesungen und Übungen gelernt haben.

(Projekt A auf der nächsten Seite)

Projekt A: Sie schreiben ein Programm zum Berechnen des 'Abstands' zwischen zwei Schauspielern. Der Abstand wird gemessen als die Länge der kürzesten Kette von Schauspielern, die gemeinsam in einem Film gespielt haben. (Eine präzisere Erklärung folgt am Freitag 16.2.)

(1) *(10 Punkte)* Sie finden hier:

`https://www.math.uni-bielefeld.de/~frettloe/teach/pdp-prakt24.html`

unter "Projekt A" mehrere Dateien, die eine Liste von Schauspielern enthalten und in denen die Filme stehen, in denen sie mitwirkten.

- Die Datei `data_actorLabel.wtf` ordnet Schauspielern eine Id zu.
- Die Datei `data_filmLabel.wtf` ordnet Filmen eine Id zu.
- In der Datei `data_relations.wtf` enthält jeder Eintrag die Id eines Schauspielers und die Id eines Films, in dem er mitgewirkt hat.

Schreiben Sie ein Programm für die Linux-Shell, welches diese Dateien in eine einzelne Datei für einen Graphen überträgt, die sich einfach mit einem Javaprogramm einlesen lässt, und den Knoten im Graphen die Namen der Schauspieler zuordnet. Die Datei soll von folgender Form sein: Jede Zeile steht für eine Zuordnung eines Schauspielers zu einem Film. Jede Zeile enthält die Id und den Namen des Schauspielers, sowie die Id und den Namen des Films, die durch Komma (oder Semikolon oder ihr Lieblings-Trennzeichen) getrennt sind. Der Cast des Films Independence Day (Id: Q105387) sähe zum Beispiel so aus:

```
Q10306924,Jim Piddock,Q105387,Independence Day
Q106706,Jeff Goldblum,Q105387,Independence Day
Q11057445,Nectar Rose,Q105387,Independence Day
Q1125651,Thom Barry,Q105387,Independence Day
Q1139435,Robert Pine,Q105387,Independence Day
...
```

(2) *(40 Punkte)* Schreiben Sie ein Javaprogramm, das als Eingabe die Namen zweier Schauspieler bekommt und

- falls einer der Schauspieler nicht in der Liste der Schauspieler vorhanden ist eine dementsprechende Fehlermeldung ausgibt;
- falls eine Kette von Schauspielern, die gemeinsam in einem Film gespielt haben, zwischen diesen Schauspielern existiert, die Länge der kürzesten solchen Kette, zwischen diesen ausgibt;
- falls keine solche Kette existiert, als Länge "Unendlich" (oder ∞) ausgibt.

Im folgenden Beispiel hat die kürzeste Kette zwischen `Halle Berry` und `Silvia Colloca` (`Halle Berry -> Hugh Jackman -> Silvia Colloca`) die Länge 3:

```
Q1033016,Halle Berry,Q106182,X-Men
Q129591,Hugh Jackman,Q106182,X-Men
Q129591,Hugh Jackman,Q211009,Van Helsing
Q271625,Silvia Colloca,Q211009,Van Helsing
```

(Projekt B auf der nächsten Seite)

Projekt B: Sie schreiben ein Programm zum Berechnen der Anzahl aller verschiedenen offenen Eulerzüge in einem Graphen; siehe Skript zur Vorlesung Mathe I, Kapitel 5, oder wikipedia:
`https://de.wikipedia.org/wiki/Eulerkreisproblem`.

Obacht: hier betrachten wir *offene* Eulerzüge, also solche, die jede Kante des Graphen genau einmal enthalten, aber deren Start- und Ziel-Knoten verschieden sein dürfen. Ein Graph, der einen offenen Eulerzug besitzt, heißt *semi-eulersch*.

(1) *(10 Punkte)* Sie finden hier:

`https://www.math.uni-bielefeld.de/~frettloe/teach/pdp-prakt24.html`

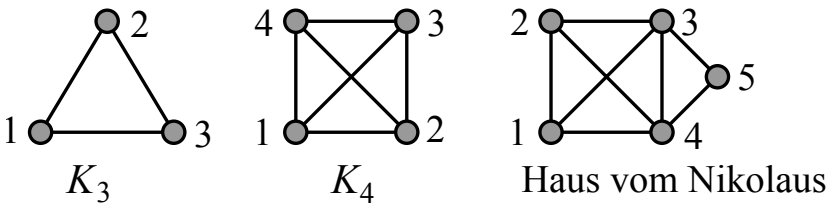
unter "Projekt B" mehrere Dateien, die jeweils einen Graphen im GraphML-Format enthalten. Schreiben Sie ein Programm für die Linux-Shell, das eine solche Datei bearbeitet, so dass Sie eine Datei für einen Graphen erhalten, der sich einfach mit einem Javaprogramm einlesen lässt. Die Datei soll von folgender Form sein: Jede Zeile steht für eine Kante im Graphen. Jede Zeile enthält daher einfach zwei Namen (oder Nummern oder Label oder...) von Knoten, die durch Komma (oder Semikolon oder ihr Lieblings-Trennzeichen) getrennt sind. Der vollständige Graph K_3 auf drei Knoten sähe also zum Beispiel einfach so aus:

```
n1;n2
n1;n3
n2;n3
```

(2) *(40 Punkte)* Schreiben Sie ein Javaprogramm, das jeweils eine der aufbereiteten Dateien einliest und

- die Anzahl aller verschiedenen offenen Eulerzüge ausgibt, falls der Graph semi-eulersch ist;
- "nicht semi-eulersch" ausgibt, falls der Graph nicht semi-eulersch ist.

Es gibt verschiedene Arten, die offenen Eulerzüge zu zählen: vorwärts und rückwärts als verschieden auffassen, oder als gleich; bei Kreisen: Startpunkte unterscheiden oder nicht. Wir wollen hier so zählen, das ein Vorwärts-Kantenzug (z.B. 1-2-3-1) verschieden ist vom Rückwärtskantenzug (hier z.B. 1-3-2-1). Wenn der Kantenzug geschlossen ist (also Startpunkt gleich Zielpunkt), dann zählen wir den mehrfach und nicht einfach (also 1-2-3-1 ist ungleich 2-3-1-2 ist ungleich 3-1-2-3). Ein paar Beispiele:



Beispiele: Die Antwort für den vollständigen Graphen K_3 auf drei Knoten ist "6", die für den vollständigen Graphen K_4 auf vier Knoten ist "nicht semi-eulersch", die für das Haus vom Nikolaus ist z.B. "88", siehe wikipedia. (Obacht, bei wikipedia steht: "44", aber dort wird vorwärts und rückwärts als gleich aufgefasst und bei uns als verschieden.)