Aufgabe 1: Superstar

Richard Wohlbold
Team-ID: 00012

5. September 2018

Inhaltsverzeichnis

1 Lösungsidee

Mein Verfahren, um zu ermitteln, ob es einen Superstar gibt und wie dieser heißt ist in zwei Phasen unterteilt:

1. In der **Ermittlungsphase** wird aus der gesamten Gruppe genau eine Person durch ein Ausschlussverfahren ermittelt, die ein Superstar sein könnte.
2. In der **Validierungsphase** wird überprüft, ob die in der Ermittlungsphase ermittelte Person tatsächlich ein Superstar ist.

Ermittlungsphase Zur Ermittlung, ob es einen Superstar gibt und wie dieser eventuell heißt, benutze ich ein Ausschlussverfahren. Am Anfang sind alle Mitglieder der Gruppe mögliche Superstars, da bisher keine Informationen vorliegen, die beweisen, dass Mitglieder keine Superstars sind. Um genau einen möglichen Superstar aus wenig Anfragen zu bestimmen, muss man aus jeder Anfrage so viele Informationen wie möglich gewinnen. Wenn X und Y zwei verschiedene Mitglieder einer Gruppe sind und beide mögliche Superstars sind, so schließt eine Anfrage der Form *Folgt X Y?* immer entweder X oder Y aus der Liste der möglichen Superstars aus, denn:

- Falls X Y folgt, so kann X kein Superstar sein, denn ein Superstar folgt keinem anderen Mitglied der Gruppe
- Falls X nicht Y folgt, so kann Y kein Superstar sein, denn alle anderen Mitglieder der Gruppe folgen einem Superstar

Um nun die Anzahl der möglichen Superstars auf einen zu reduzieren, nimmt man die ersten zwei Mitglieder X und Y aus der Liste, stellt die Anfrage *Folgt X Y?* und fügt das nichtausgeschlossene Mitglied wieder der Liste hinzu.
Da mit diesem Verfahren mit einer Anfrage je ein Mitglied aus der Liste der möglichen Superstars eliminiert werden kann und am Anfang n Mitglieder in der Liste sind und am Ende nur 1 Mitglied in der Liste verbleibt, müssen $n-1$ Anfragen gestellt werden, um die Anzahl der möglichen Superstars von n auf 1 zu reduzieren.

Validierungsphase Da jetzt nurnoch ein möglicher Superstar S vorliegt, muss noch überprüft werden, ob dieser tatsächlich einer ist. Die einzige Möglichkeit, um dies zu überprüfen, ist für jedes andere Mitglied X zwei Anfragen zu stellen:

1. Folgt X S? Falls nein, ist S kein Superstar.
2. Folgt S X? Falls ja, ist S kein Superstar.

Falls bei einer der Anfragen herauskommt, dass S kein Superstar ist, ist die Validierungsphase beendet und es gibt keinen Superstar in der Gruppe, da S der einzige mögliche Superstar vor der Validierungsphase war.
Falls bei keiner der Anfragen herauskommt, dass S kein Superstar ist, haben wir festgestellt, dass S niemandem folgt, jedoch jeder S folgt, sodass S der Superstar ist.
Falls S der Superstar ist, werden in der Validierungsphase $n-1$ Anfragen der Form *Folgt S X?* und $n-1$ Anfragen der Form *Folgt X S?* gestellt, sodass maximal $2(n-1)$ Anfragen gestellt werden.

Optimierung des Verfahrens Da in der Validierungsphase alle Anfragen, die den möglichen Superstar S enthalten, gestellt werden müssen, aber schon in der Ermittlungsphase einige dieser Anfragen gestellt werden müssen, um S zu ermitteln, kommt es zwingend vor, dass Anfragen doppelt gestellt werden. Da jede doppelt gestellte Anfrage zu vermeiden ist, können bereits gestellte Anfragen und ihre Ergebnisse gespeichert werden und für jede Anfrage kann überprüft werden, ob ihr Ergebnis bereits im Zwischenspeicher existiert, sodass die Anzahl der Anfragen leicht gesenkt werden kann. Da in der Informatik jedoch oft der Worst-Case zählt, ist meine Schätzung der Anzahl der Anfragen $3(n-1) = 3n-3$.

2 Umsetzung

Das Verfahren zur Bestimmung eines Superstars wurde in Python umgesetzt.

Dazu wird als erstes in der Funktion `eingabe_lesen` die Eingabe in die Liste der Mitglieder und die Liste der Verbindungen (X folgt Y) eingelesen. Dabei ist eine Verbindung eine Liste der beiden Namen der Form `[X, Y]`.

Die Funktion `folgt` um das Stellen der Anfrage, das Speichern, falls die gleiche Anfrage noch einmal gestellt werden sollte und um das Ausgeben von Anfragen und des Ergebnisses. Der Zwischenspeicher ist ein `dict` das als Schlüssel Tupel der Form `(X,Y)` verwendet, da Python keine Listen als Schlüssel unterstützt, und als Wert, ob X Y folgt, zurückgibt.

Die Funktion `superstar_bestimmen` ist die Funktion, die aus den gelesenen Mitgliedern den Superstar bestimmen. Dazu werden die Bestimmungs- und Validierungsphase wie oben beschrieben ausgeführt. Um zu bestimmen, ob ein Mitglied einem anderen folgt, wird `folgt` ausgeführt, damit die Anfrage richtig verarbeitet wird.

Das erste Argument des Skripts sollte die Eingabedatei sein, sonst wird eine Fehlermeldung ausgegeben. Das Skript verlässt sich auf die Gültigkeit der Eingabedatei; falls diese nicht richtig formattiert ist, kann es zu unerwarteten Fahlern führen.

3 Beispiele

Die erste Zeile ist nicht Teil des Programms, sondern dient zur Illustration, wie das Skript aufgerufen werden sollte. Die letzte leere Zeile und das $ soll zeigen, dass die Ausgabe hier endet.

Beispiel 1 von der bwinf-Webseite:

```
$ python superstar.py ../beispieldaten/superstar1.txt
Folgt Hailey Justin?  Ja!
Folgt Justin Selena?  Nein!
Folgt Selena Justin?  Ja!
Folgt Justin Hailey?  Nein!
Superstar: Justin
3 Mitglieder
4 gestellte Anfragen

$
```

Beispiel 2 von der bwinf-Webseite:

```
$ python superstar.py ../beispieldaten/superstar2.txt
Folgt Codd Knuth?   Ja!
Folgt Knuth Dijkstra?   Ja!
Folgt Dijkstra Hoare?   Nein!
Folgt Dijkstra Turing?  Nein!
Folgt Turing Dijkstra?  Ja!
Folgt Hoare Dijkstra?   Ja!
Folgt Dijkstra Knuth?   Nein!
Folgt Codd Dijkstra?  Ja!
Folgt Dijkstra Codd?  Nein!
Superstar: Dijkstra
5 Mitglieder
9 gestellte Anfragen

$
```

Beispiel 3 von der bwinf-Webseite:

```
$ python superstar.py ../beispieldaten/superstar3.txt
Folgt Sjoukje Rinus?  Nein!
Folgt Sjoukje Rineke?   Nein!
Folgt Sjoukje Pia?  Nein!
Folgt Sjoukje Peter?  Nein!
Folgt Sjoukje Jorrit?   Nein!
Folgt Sjoukje Jitse?  Nein!
Folgt Sjoukje Edsger?   Ja!
Folgt Jitse Edsger?   Ja!
Folgt Edsger Jitse?   Ja!
Kein Superstar
8 Mitglieder
9 gestellte Anfragen

$
```

Beispiel 4 von der bwinf-Webseite (Die Liste der Anfragen wurde hier aus Platzgründen auf die erste und lezte Zeile reduziert):

```
$ python superstar.py ../beispieldaten/superstar4.txt
Folgt Rut Penny?  Nein!
...
Folgt Folke Rut?  Nein!
Superstar: Folke
80 Mitglieder
211 gestellte Anfragen

$
```

4 Quellcode (ausschnittsweise)

Die Funktion `superstar_bestimmen`, die den Superstar ermittelt, falls es einen gibt:

```python
def superstar_bestimmen(mitglieder, verbindungen):
    # Ermittlungsphase
    moegliche_superstars: list = mitglieder.copy()

    # Wiederholen, bis es nurnoch einen moeglichen Superstar gibt
    while len(moegliche_superstars) > 1:

        mitglied1 = moegliche_superstars.pop()
        mitglied2 = moegliche_superstars.pop()

        # Eines der beiden Mitglieder ausschliessen
        if folgt(mitglied1, mitglied2, verbindungen):
            moegliche_superstars.append(mitglied2)
        else:
            moegliche_superstars.append(mitglied1)

    # Validierungsphase

    vermuteter_superstar = moegliche_superstars[0]

    # Fuer jedes Mitglied fragen, ob es dem vermuteten Superstar folgt oder andersherum
    for mitglied in mitglieder:
        if mitglied != vermuteter_superstar:
            if not folgt(mitglied, vermuteter_superstar, verbindungen):
                return None
            if folgt(vermuteter_superstar, mitglied, verbindungen):
                return None

    return vermuteter_superstar
```

superstar_bestimmen.py

Die Funktion `folgt`, die eine Anfrage stellt, sie zwischenspeichert, die Anfrage und das Ergebnis ausgibt und die Gesamtkosten verfolgt:

```python
verbindungen_zwischenspeicher = {}
euro = 0

# Folgt X Y?
def folgt(x, y, verbindungen):

    # Aufruf aus Zwischenspeicher beantworten, falls moeglich
    if verbindungen_zwischenspeicher.get((x,y)) is not None:
        return verbindungen_zwischenspeicher.get((x,y))

    # Anfrage stellen
    ergebnis = [x,y] in verbindungen

    # Anfrage und Ergebnis zwischenspeichern
    verbindungen_zwischenspeicher[(x,y)] = ergebnis

    # Anzahl der Anfragen erhoehen
    global euro
    euro += 1

    # Anfrage und Ergebnis ausgeben
    if ergebnis:
        print("Folgt", x, y + "?␣\tJa!")
    else:
        print("Folgt", x, y + "?␣\tNein!")

    return ergebnis
```

folgt.py