

Aufgabe 1: Superstar

Richard Wohlbold

Team-ID: 00012

5. September 2018

Inhaltsverzeichnis

1	Lösungsidee	2
2	Umsetzung	3
3	Beispiele	4
4	Quellcode (ausschnittsweise)	5

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 nun noch 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 S jeder folgt, sodass S ein Superstar ist.

Falls S ein Superstar ist, werden in der Validierungsphase werden $n - 1$ 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 möglichen Anfragen, die den möglichen Superstar S enthalten, gestellt werden müssen, aber schon in der Validierungsphase 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 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 $[X, Y]$. Dabei kümmert sich die Funktion `folgt` um das Stellen der Anfrage, das Speichern, falls die gleiche Anfrage noch einmal gestellt werden sollte und um das Ausgeben der Anfrage 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.

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:

```
1 $ python superstar.py ../beispieldaten/superstar1.txt
  Folgt Hailey Justin?  Ja!
3 Folgt Justin Selena?  Nein!
  Folgt Selena Justin?  Ja!
5 Folgt Justin Hailey?  Nein!
  Superstar: Justin
7 3 Mitglieder
  4 gestellte Anfragen
9
$
```

Beispiel 2 von der bwinf-Webseite:

```
$ python superstar.py ../beispieldaten/superstar2.txt
2 Folgt Codd Knuth?    Ja!
  Folgt Knuth Dijkstra?  Ja!
4 Folgt Dijkstra Hoare?  Nein!
  Folgt Dijkstra Turing? Nein!
6 Folgt Turing Dijkstra? Ja!
  Folgt Hoare Dijkstra?  Ja!
8 Folgt Dijkstra Knuth?  Nein!
  Folgt Codd Dijkstra?   Ja!
10 Folgt Dijkstra Codd?  Nein!
   Superstar: Dijkstra
12 5 Mitglieder
   9 gestellte Anfragen
14
$
```

Beispiel 3 von der bwinf-Webseite:

```
1 $ python superstar.py ../beispieldaten/superstar3.txt
  Folgt Sjoukje Rinus?  Nein!
3 Folgt Sjoukje Rineke?  Nein!
  Folgt Sjoukje Pia?    Nein!
5 Folgt Sjoukje Peter?  Nein!
  Folgt Sjoukje Jorrit?  Nein!
7 Folgt Sjoukje Jitse?  Nein!
  Folgt Sjoukje Edsger?  Ja!
9 Folgt Jitse Edsger?   Ja!
  Folgt Edsger Jitse?   Ja!
11 Kein Superstar
   8 Mitglieder
13 9 gestellte Anfragen
15 $
```

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

```
1 $ python superstar.py ../beispieldaten/superstar4.txt
  Folgt Rut Penny?  Nein!
3 ...
  Folgt Folke Rut?  Nein!
5 Superstar: Folke
  80 Mitglieder
7 211 gestellte Anfragen
9 $
```

4 Quellcode (ausschnittsweise)

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

```

1 def superstar_bestimmen(mitglieder, verbindungen):
2     # Ermittlungsphase
3     moegliche_superstars: list = mitglieder.copy()
4
5     # Wiederholen, bis es nur noch einen moeglichen Superstar gibt
6     while len(moegliche_superstars) > 1:
7
8         mitglied1 = moegliche_superstars.pop()
9         mitglied2 = moegliche_superstars.pop()
10
11        # Eines der beiden Mitglieder ausschliessen
12        if folgt(mitglied1, mitglied2, verbindungen):
13            moegliche_superstars.append(mitglied2)
14        else:
15            moegliche_superstars.append(mitglied1)
16
17    # Validierungsphase
18
19    vermuteter_superstar = moegliche_superstars[0]
20
21    # Fuer jedes Mitglied fragen, ob es dem vermuteten Superstar folgt oder andersherum
22    for mitglied in mitglieder:
23        if mitglied != vermuteter_superstar:
24            if not folgt(mitglied, vermuteter_superstar, verbindungen):
25                return None
26            if folgt(vermuteter_superstar, mitglied, verbindungen):
27                return None
28
29    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:

```

1 verbindungen_zwischenspeicher = {}
2 euro = 0
3
4 # Folgt X Y?
5 def folgt(x, y, verbindungen):
6
7     # Aufruf aus Zwischenspeicher beantworten, falls moeglich
8     if verbindungen_zwischenspeicher.get((x,y)) is not None:
9         return verbindungen_zwischenspeicher.get((x,y))
10
11    # Anfrage stellen
12    ergebnis = [x,y] in verbindungen
13
14    # Anfrage und Ergebnis zwischenspeichern
15    verbindungen_zwischenspeicher[(x,y)] = ergebnis
16
17    # Anzahl der Anfragen erhoehen
18    global euro
19    euro += 1
20
21    # Anfrage und Ergebnis ausgeben
22    if ergebnis:
23        print("Folgt", x, y + "?_\tJa!")
24    else:
25        print("Folgt", x, y + "?_\tNein!")
26
27    return ergebnis

```

folgt.py