Aufgabe 1: Superstar

Team: Jan Niklas Groeneveld
Team-ID: 00828

22. November 2018

Inhaltsverzeichnis

1 Lösungsidee

Um sich sicher sein zu können, dass eine Person ein der Definition entsprechender Superstar ist, muss ausgeschlossen werden, dass sie einer anderen Person folgt. Ist das gesichert, muss noch geprüft werden, dass wirklich keine andere Person dieser folgt. Dazu werden zuerst die Personen eingelesen und der Reihe nach gespeichert. Anschließend werden die Informationen, wem eine Person folgt, den Personen zugeordnet und gespeichert. Das simuliert jedoch nur die Datenbank des Netzwerkes, dessen Inhalt vom Algorithmus dann auf die geforderte Art so sparsam wie möglich abgefragt wird. Für jede Person wird gespeichert, ob sie überhaupt noch als Kandidat in Frage kommt, damit nur für diese sinnvolle Fragen gestellt werden. Dann beginnt ein Backtracking-Algorithmus, der für die erste Person feststellt, ob es sich um einen Superstar handelt. Dieser geht alle Personen durch und stellt die Frage an die Datenbank, ob die Person, die gerade als Kandidat in Betracht gezogen wird, jener Person auf der Liste folgt. Dabei darf sie nicht abfragen, ob sie sich selber folgt. Bei einer Negierung der Frage ist die Person, der nicht gefragt wird, auf jeden Fall kein Superstar. Bei ihr wird das entsprechend gespeichert, sodass sie nicht als Kandidat gehandelt wird. Wird diese Frage in einem Fall bejaht, kann ausgeschlossen werden, dass es sich beim bisherigen Kandidaten um den tatsächlichen Superstar handelt. Stattdessen wird die Person, der gefolgt wird, als Kandidat in Betracht gezogen, sofern sie nicht bereits durch andere Umstände mit entsprechender Vermerkung ausgeschlossen ist. Als Kandidat wird für sie dann die gleiche Routine aufgerufen, die die neue Person dahingehend abprüft, ob sie Superstar ist. Wichtig ist hierbei, dass sie erst ab der ihr folgenden fragt, und erst danach die Vorgänger abfragt. Warum das so wichtig ist, wird unten diskutiert. Wurde bei der Prüfung, ob eine Person einer anderen folgt, hingegen keine gefunden, der von dieser gefolgt wird, ist jene weiterhin Kandidat. Dann muss für jede andere Person in der Gruppe geprüft werden, ob sie dem Kandidaten folgt. Fällt die Auskunft hier immer positiv aus, handelt es sich um einen Superstar. Diese Information des Superstars wird dann durch alle vorigen Aufrufe zurückgegeben, um sie dann auszugeben. Kann eine Kandidatenprüfung eine Person nicht als Superstar identifizieren, gibt sie diese Information entsprechend zurück. Weil es bei diesem Ablauf vorkommt, dass Fragen doppelt gestellt werden, wird die Antwort auf eine Frage stets gespeichert. Wird eine Frage gestellt, wird aus diesem Grund vor einer kostenpflichtigen Anfrage an die simulierte Datenbank immer geprüft, ob die Antwort bereits gespeichert ist, und deswegen aus dem Speicher bezogen werden kann.

Um Aussagen darüber treffen zu können, wie teuer das Ermitteln eines Superstars in einer Gruppe mit n Teilnehmern ist, lohnt es sich, den theoretischen Fall mit den geringsten Kosten und den höchsten Kosten zu betrachten. Ein Fall mit minimalen Kosten liegt zum Beispiel dann vor, wenn keiner in der Gruppe einer anderen Person folgt. Dann prüft die erste Person für alle anderen $n-1$ Personen, ob sie einer folgt.

Weil das immer negativ ausfällt, ist sie noch Superstar-Kandidat. Gleichzeitig ist zu diesem Zeitpunkt bereits ausgeschlossen, dass eine andere Person Superstar ist. Anschließend wird der Algorithmus für die nächste Person prüfen, ob sie dem Kandidaten folgt. Weil das auch nicht der Fall ist, ist nach n Fragen bekannt, dass es keinen Superstar gibt. Demnach gilt für die Mindestkosten K_{min} in Euro:

$$K_{min} = n \tag{1}$$

Bezüglich der Kosten im Fall der höchsten Zahlung lohnt es sich, ein Beispiel zu betrachten: In einer Gruppe sind die Personen A, B, C und D. Die Namen sind in dieser Reihenfolge angegeben. Jede Person außer D folgt seinem Nachfolger, also folgt A B, B folgt C und C folgt D. Darüber hinaus folgen alle Personen D (mit Ausnahme von D selbst, da sich niemand selbst folgen kann).Zuerst wird der Algorithmus besprochen, bei dem jeder Kandidat wieder von vorn beginnt, die Personen danach abzufragen, ob er ihnen folgt: Zuerst wird erfragt, dass A B folgt, dann fragt B zwei Fragen (ob er A und C folgt). Weil er C folgt, muss dieser drei Fragen stellen (bis er weiß, dass er D folgt und die Routine bei D aufrufen kann). D muss drei mal Fragen, damit man weiß, dass er niemandem folgt. Danach muss er sich vergewissern, dass ihm alle anderen Personen folgen. Dazu müsste er drei Fragen stellen. Weil jedoch bereits einmal erfragt wurde, dass C D folgt, fallen nur für zwei Fragevorgänge kosten an. In diesem Fall bedeutet das Kosten in Höhe von 11€. Allgemein lässt sich das folgendermaßen formulieren: Es wird die Summe der Fragen derjenigen Personen summiert, die nicht der Superstar sind, also $\sum_{i=1}^{n-1} i$. Dazu müssen $n-1$ addiert werden, weil ausgeschlossen werden muss, dass D einer anderen Person folgt. Anschließend werden weitere $n-2$ fällig, weil noch einmal für jeden Gruppenteilnehmer außer sich selbst und denjenigen, der einen aufgerufen hat und deswegen die Frage bereits gestellt hat, erfragt werden muss, dass er dem Kandidaten folgt. Deswegen gilt:

$$K_{max} = \sum_{i=1}^{n-1}(i) + (n-1) + (n-2) \tag{2}$$

Wird die Summenformel gelöst und vereinfacht, bleibt die Gleichung einer Parabel:

$$K_{max} = \frac{n^2}{2} + \frac{3n}{2} - 3 \tag{3}$$

Jetzt wird erläutert, warum der durch den oben angesprochenen kleinen Kunstgriff optimierte Algorithmus wesentlich kostensparender ist: Der Unterschied besteht darin, dass beim Versuch des Ausschlusses, dass ein Kandidat anderen folgt, jetzt mit Priorität diejenigen abgefragt werden, die noch als Kandidaten in Frage kommen, weil diejenigen, die in der Liste vor der eigenen Person stehen, bereits als Superstars ausgeschlossen wurden. Deswegen wird nur abgefragt, ob ihnen gefolgt wird, wenn keine andere Person als Superstar noch möglich ist. In der Rechnung wirkt sich das folgendermaßen aus: Im kostenintensivsten Fall müssen $n-1$ Fragen gestellt werden, bis die letzte Person (D) Kandidat ist. Sie muss daraufhin noch einmal $(n-1)+(n-2)$ Fragen stellen, um als Superstar festzustehen. Deswegen wachsen die maximalen Kosten hier nur linear:

$$K_{max} = (n-1) + (n-1) + (n-2) = 3n - 4 \tag{4}$$

Deswegen kann man jetzt die Aussage treffen, dass für die Kosten K für $n \geq 2$gilt: $n \leq K \leq 3n-4$. Weil die Wahrscheinlichkeit für das Vorhandensein eines Superstars nicht bekannt ist, müssen für die nähere Betrachtung diese beiden Fälle unterschieden werden. Wenn ein Superstar in einer Gruppe ist, müssen auf jeden Fall die Kosten $(n-1)+(n-2) = 2n-3$ aufgewendet werden, um das zu beweisen. Deswegen gilt für die Kosten K_S", wenn es einen Superstar in der Gruppe gibt, $2n-3 \leq K_S \leq 3n-4$. Der Mittelwert dessen K_{SD} liegt bei $K_{SD} = (n-1)/2 + (n-1) + (n-2) = (5n-7)/2$. Gibt es keinen Superstar, ist das im Allgemeinen bereits nach geringerer Zahlung bewiesen.

2 Umsetzung

In der Java-Implementierung dieses Problems enthält die Hauptklasse die Funktion `main()`. In ihr findet die Kommunikation mit dem Nutzer mithilfe der Klasse `SimpleInput` statt, und das Auslesen der Datei mit der Klasse `ZeilenweiseAuslesen`. Sind die Namen eingelesen, werden sie nach Leerzeichen getrennt und mit ihren Namen wird eine Instanz der Klasse `Person` erzeugt. Diese hat als Eigenschaften den Namen der Person, und mehrere Arrays mit Zeigern auf Instanzen der Klasse »Person«, die jeweils die Länge der Anzahl an Personen haben. In einen werden über die Methoden `hinzufuegen_folgen()` alle Personen hinzugefügt, denen die Person folgt. Dieser Array ist die simulierte Datenbank, in dem die Informationen von Beginn an gespeichert werden. Den Arrays `folgenfrei` und `nichtfolgen` werden die Informationen

hinzugefügt, die erfragt werden. Sie sind dadurch der Speicher, den der Algorithmus kostenfrei auslesen kann. Darüber hinaus wird in der `main()`-Methode eine Instanz der Klasse `Konto` erzeugt. Diese Instanz speichert die bisher fällige Zahlung. Über ihre Methode `naechste_zahlung()` wird der Wert der bisherigen Zahlung um eins erhöht. Um das Ergebnis in der `main()`-Methode auszugeben, wird hier ein String der Ausgaben zurückgegeben.

Zentralelement des Backtracking-Algorithmus ist die Methode `wer_ist_der_superstar()`, der ein Zeiger auf die Instanz des Kontos und ein Array mit Zeigern auf alle Personen übergeben wird. Damit eine Person in dieser Rekursion nicht zweimal aufgerufen wird, und auch keine Person abgefragt wird, die überhaupt kein Kandidat mehr ist, werden diese beiden Fälle am Anfang abgefangen. Im Anschluss wird der Variable `bin_ich_gerade_kandidat` der Wert `true` gesetzt, um die Person zu blockieren. Jetzt beginnt die Schleife, die durch die Personen durchzählt, um zu prüfen, wem die Person folgt. Der Kunstgriff, der ermöglicht, dass die im Array nach einem stehenden Personen zuerst abgefragt werden, wird dadurch verwirklicht, dass die Funktion `get_num()`, der der Array der Personen übergeben wird, die Position der abgefragten Person im Array zurückgibt. Von diesem Wert wird dann bis unter die Summe aus diesem Wert und der Länge des Arrays gezählt, und die Position im Array durch eine Modulooperation der Zählvariable und der Anzahl der Personen berechnet. Bezüglich dieser Personen werden dann einige Fälle der Reihe nach abgefragt:

1. Gleichen sich die Person, für die die Methode aufgerufen wurde, und die Person, auf die der Zeiger im Array zeigt, muss die Schleife durch `continue` in den nächsten Durchlauf springen.

2. Wenn die Person, für die die Methode aufgerufen wurde, selber kein Kandidat mehr ist, und die Person, die im Array an der Stelle der Zählvariable steht, auch kein Kandidat mehr ist, wäre eine Abfrage sinnlos, da sie keine verwertbaren Informationen mit sich trüge. In diesem Fall wird auch über `continue` weitergesprungen.

3. Ist keiner der vorherigen Fälle eingetreten, wird nun gefragt, ob die Person, die von der aktuellen Instanz repräsentiert wird, der entsprechenden Person im Array folgt. Wenn das der Fall ist, ist die aktuelle Person in jedem Fall als Superstar auszuschließen. Dann wird für den neuen Kandidaten die Methode `wer_ist_der_superstar()` aufgerufen. Das Ergebnis wird dann folgendermaßen evaluiert. Wird ein Zeiger ungleich `null` zurückgegeben, bedeutet das, dass ein sicherer Superstar gefunden wurde. Dieser wird weiter zurückgegeben. Wenn bei der vorherigen Abfrage jedoch ermittelt wird, dass die aktuelle Person der entsprechenden Person an der Stelle des Arrays der Zählvariable nicht folgt, wird bei dieser Person die Variable `bin_ich_ueberhaupt_noch_kandidat` auf `false` gesetzt, sie ist also ausgeschlossen.

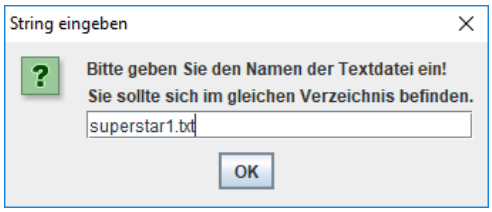
Wenn dieser Schleifendurchlauf so beendet wurde, dass die aktuelle Person immer noch Kandidat ist, muss noch für alle Personen im Array sichergestellt werden, dass sie aktuellen Kandidaten wirklich folgen. Dabei muss immer der Fall abgefangen werden, in dem die Routine versuchen würde, abzufragen, ob sich die Person selbsst folgt. Sollte sich herausstellen, dass irgendeine Person dem aktuellen Kandidaten nicht folgt, muss dies in der Variable `bin_ich_ueberhaupt_noch_kandidat()` gespeichert werden, und anschließend null zurückgegeben werden. Selbiges muss auch zurückgegeben werden, wenn sich bereits in der ersten Schleife herausgestellt hat, dass eine Person kein Kandidat mehr ist. Sollten alle Prüfungen so ausfallen, dass es sich bei einer Person um den Superstar handelt, muss der Zeiger auf sich selbst zurückgegeben werden.

Ebenfalls bedeutsam ist die Methode `folgst_Du_()`, bei der abgefragt wird, ob eine Person einer anderen folgt. Sie iteriert zuerst durch die Arrays `folgenfrei` und `nichtfolgen`, in denen die bekannten Antworten gespeichert werden. Trifft sie hier auf die gesuchte Person, kann sie `true` zurückgeben, wenn die Person im ersten Array gefunden wurde, und `false`, wenn das für den zweiten der Fall ist. Wurde in diesen Arrays die Person nicht gefunden, wird eine Zahlung fällig. Dem übergebenen Zeiger auf die Instanz der Klasse `Konto` wird der Auftrag zur Erhöhung des Kontostandes erteilt, anschließend wird der Array, der die Datenbank simuliert, durchgezählt. Wird dabei auf die gesuchte Person getroffen, bedeutet das, dass ihr gefolgt wird. Dem Array `folgenfrei` wird diese Person als bekannte Person hinzugefügt. Trifft die Schleife jedoch auf ein Arrayelement, dessen Zeiger `null` ist, bedeutet das, dass dieser Person nicht gefolgt wird. Dann wird diese Information im Array `nichtfolgen` gespeichert.

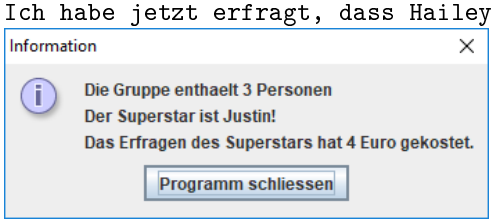
Ist die Rückgabeinformation aus diesen Routinen wieder in der `main()`-Methode angekommen, wird der Superstar oder die Information, dass es keinen Superstar gibt, zusammen mit der Kontoinformation zurückgegeben.

3 Beispiele

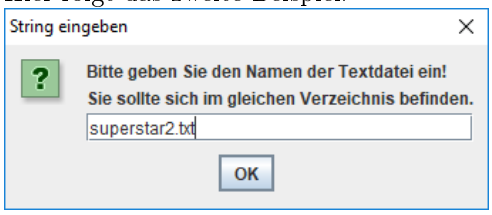
In diesen Beispielen werden einmal die Ein- und Ausgabefenster ausgedruckt, sowie zur zusätzlichen Information über die Funktionsweise des Programmes die getätigten Abfragen, die im abgegebenen Code auskommentiert sind. Auf letzteres wird beim Beispiel »superstar4.txt« aus Platzgründen verzichtet. Das ist das erste Beispiel:



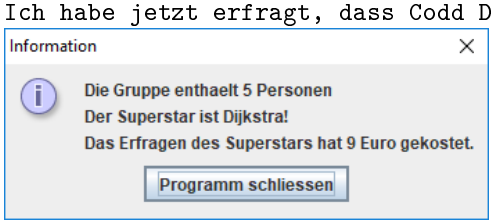

```
Ich habe jetzt erfragt, dass Selena Justin folgt.
Ich habe jetzt erfragt, dass Justin Hailey nicht folgt.
Ich habe jetzt erfragt, dass Justin Selena nicht folgt.
Ich weiss bereits, dass Selena Justin folgt.
Ich habe jetzt erfragt, dass Hailey Justin folgt.
```



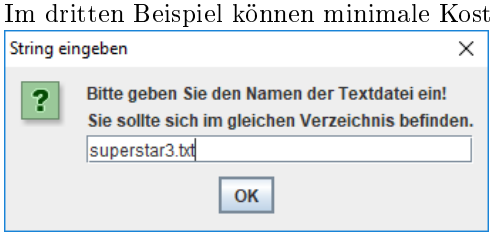
Hier folgt das zweite Beispiel:



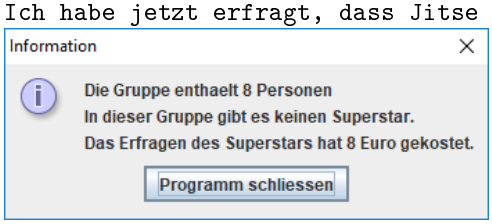

```
Ich habe jetzt erfragt, dass Turing Hoare folgt.
Ich habe jetzt erfragt, dass Hoare Dijkstra folgt.
Ich habe jetzt erfragt, dass Dijkstra Knuth nicht folgt.
Ich habe jetzt erfragt, dass Dijkstra Codd nicht folgt.
Ich habe jetzt erfragt, dass Dijkstra Turing nicht folgt.
Ich habe jetzt erfragt, dass Dijkstra Hoare nicht folgt.
Ich habe jetzt erfragt, dass Turing Dijkstra folgt.
Ich weiss bereits, dass Hoare Dijkstra folgt.
Ich habe jetzt erfragt, dass Knuth Dijkstra folgt.
Ich habe jetzt erfragt, dass Codd Dijkstra folgt.
```



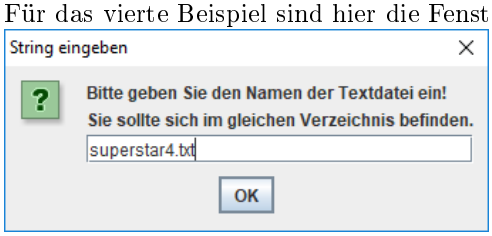
Im dritten Beispiel können minimale Kosten erreicht werden:

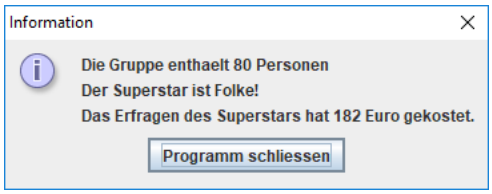



```
Ich habe jetzt erfragt, dass Edsger Jitse folgt.
Ich habe jetzt erfragt, dass Jitse Jorrit nicht folgt.
Ich habe jetzt erfragt, dass Jitse Peter nicht folgt.
Ich habe jetzt erfragt, dass Jitse Pia nicht folgt.
Ich habe jetzt erfragt, dass Jitse Rineke nicht folgt.
Ich habe jetzt erfragt, dass Jitse Rinus nicht folgt.
Ich habe jetzt erfragt, dass Jitse Sjoukje nicht folgt.
Ich habe jetzt erfragt, dass Jitse Edsger folgt.
```



Für das vierte Beispiel sind hier die Fenster ausgedruckt:





4 Quellcode

```
public static void main(String[] args) {
    // In den folgenden Zeilen wird die vom Nutzer geforderte Textdatei ausgelesen
    SimpleInput si = new SimpleInput();
    String input = si.getString("Bitte geben Sie den Namen der Textdatei ein!\nSie sollte
     sich im gleichen Verzeichnis befinden.");
    ZeilenweiseAuslesen za = new ZeilenweiseAuslesen(input);
    String text = za.auslesen(1);
    // Die erste Zeile wird so in die einzelnen Namen zerlegt, weil die Namen durch
    Leerzeichen getrennt sind
    String[] personenstring = text.split(" ");
    int anzahlpersonen = personenstring.length;
    // Hier werden Instanzen der Klasse "Person" mit den jeweiligen Namen angelegt
    Person[] personen = new Person[anzahlpersonen];
    for (int i = 0; i < anzahlpersonen; i++)
        personen[i] = new Person(personenstring[i], anzahlpersonen);
    // In der folgenden Routine werden die folgenden Zeilen, in denen die Beziehungen der
     Personen gesschrieben sind, ausgelesen
    int zaehler = 2;
    while (true) {
        String ausgelesen = za.auslesen(zaehler);
        if (ausgelesen == null) break;
        if (ausgelesen.length() <= 1) break;
        zaehler++;
        anfuegen(ausgelesen, personen);
    }
    // Bei Bedarf kann hier die Liste derjenigen ausgegeben werden, denen von einer
    Person gefolgt wird
    /*for (int i = 0; i < anzahlpersonen; i++)
        personen[i].wem_ich_folge();*/
    Konto konto = new Konto(); // Hier wird das Konto, das die Kosten zaehlt,
    initialisiert
    String ausgabe = "Die Gruppe enthaelt " + anzahlpersonen + " Personen\n";
    Person superstar = personen[0].wer_ist_der_superstar(konto, personen); // Diese
    Routine beginnt die Backtracking-Suche nach dem Superstar
    // In der folgenden Abfrage wird der Ausgabetext gesetzt
    if (superstar != null) ausgabe += "Der Superstar ist " + superstar.name + "!";
    else ausgabe += "In dieser Gruppe gibt es keinen Superstar.";
    ausgabe += "\n";
    ausgabe += konto.wie_teuer_war_es();
    si.message(ausgabe);
    //System.out.println(ausgabe);
    System.exit(0);
}
```

```
public Person wer_ist_der_superstar(Konto konto, Person[] personen) {
    if (bin_ich_gerade_kandidat || !bin_ich_ueberhaupt_noch_kandidat)
        return null; // Sollte diese Person bereits einmal in der Rekursion vorkommen,
    wird zurueckgesprungen
    bin_ich_gerade_kandidat = true; // Damit eine Person nicht zweimal abgefragt wird,
    wird das hier vermerkt
    // Diese Schleife zaehlt durch die Personen, um rekursiv durchzufragen
    int start = get_num(personen);
    for (int z = start; z < start + personen.length; z++) {
        int i = z % personen.length;
        if (personen[i].name.equals(this.name))
            continue; // Die Aufgabe wird von einer Person nicht an sich selbst
    weitergegeben
        if (!personen[i].bin_ich_ueberhaupt_noch_kandidat &&
    bin_ich_ueberhaupt_noch_kandidat)
            continue; // Wenn eine Person kein Kandidat mehr ist, und man selber auch
    nicht, ist diese Person uninteressant
        if (folgst_Du_(personen[i], konto)) {
            // Wenn die Person einer anderen folgt, ist sie der Definition entsprechend
    kein Kandidat mehr. Einer Person, der gefolgt wird, wird nun die Aufgabe des
    Durchfragens uebergeben
            bin_ich_ueberhaupt_noch_kandidat = false;
            Person superstar = null;
            superstar = personen[i].wer_ist_der_superstar(konto, personen);
            if (superstar != null)
```

```
                return superstar; // Wenn eine andere Person der Superstar ist, wird ein Zeiger auf ihn zurueckgegeben
            } else personen[i].bin_ich_ueberhaupt_noch_kandidat = false;
        }
        if (bin_ich_ueberhaupt_noch_kandidat) // Wenn die Person noch Kandidat ist, muss geprueft werden, ob ihr alle anderen folgen
        {
            for (int i = 0; i < personen.length; i++) {
                if (personen[i].name.equals(this.name))
                    continue; // Niemand kann sich selber folgen, deswegen muss dieser Fall uebersprungen werden
                if (!personen[i].folgst_Du_(this, konto)) {
                    bin_ich_ueberhaupt_noch_kandidat = false;
                    return null; // Wenn irgendeine Person dieser nicht folgt, ist sie kein Superstar
                }
            }
        } else {
            bin_ich_gerade_kandidat = false;
            return null; // Wenn eine Person nach dieser Ueberpruefung kein Superstar mehr ist, muss "null" zurueckgegeben werden
        }
        bin_ich_gerade_kandidat = false;
        return this; // Hier kommt die Funktion genau dann an, wenn die Person der Superstar ist. Auf ihn wird ein Zeiger zurueckgegeben
    }
```

```
    public boolean folgst_Du_(Person gesucht, Konto konto) {
        //Die unter der naechsten Zeile eingeleitete Schleife, prueft zuerst, ob sich die Frage bereits mit dem bisherigen Wissen beantworten laesst. Wenn der Name der erfragten Person in dem Array "folgenfrei" steht, ist klar, dass dieser Person gefolgt wird, wenn der Name im Array "nichtfolgen" enthalten ist, kann zurueckgegeben werden, dass der gefragten Person nicht gefolgt wird.
        int maxloop = (nichtfolgen.length > folgenfrei.length) ? nichtfolgen.length : folgenfrei.length;
        for (int i = 0; i < maxloop; i++) {
            if (folgenfrei[i] != null)
                if (folgenfrei[i].name.equals(gesucht.name)) {
                    System.out.printf("Ich weiss bereits, dass %s %s folgt.\n", this.name, gesucht.name);
                    return true;
                }
            if (nichtfolgen[i] != null)
                if (nichtfolgen[i].name.equals(gesucht.name)) {
                    System.out.printf("Ich weiss bereits, dass %s %s nicht folgt.\n", this.name, gesucht.name);
                    return false;
                }
        }
        // Sollte noch kein Wert zurueckgegeben sein, bedeutet das, dass das Verhaeltnis ungeklaert ist. Dann wird eine Zahlung faellig
        konto.naechste_zahlung();
        // Diese Schleife zaehlt durch den Array der Personen, in dem eingangs alle gespeichert wurden, denen die Person folgt
        for (int i = 0; i < folgen.length; i++) {
            if (folgen[i] == null) {
                hinzufuegen_nichtfolgen(gesucht);
                //System.out.printf("Ich habe jetzt erfragt, dass %s %s nicht folgt.\n", this.name, gesucht.name);
                return false;
            }
            if (folgen[i].name.equals(gesucht.name)) {
                hinzufuegen_folgenfrei(gesucht);
                //System.out.printf("Ich habe jetzt erfragt, dass %s %s folgt.\n", this.name, gesucht.name);
                return true;
            }
        }
        return false;
    }
```