Problemstellung / Aufgabe

Unsere Aufgabe ist es den Superstar einer Gruppe zu ermitteln. Dies sollten wir in so wenig Abfragen wie möglich verwirklichen, da es pro Abfrage einen Euro Aufwandsentschädigung kostet. Abgefragt werden darf nur „Folgt A, B?“, andere Abfragen sind nicht zulässig. Der Superstar muss mithilfe dieser Art von Abfragen gefunden werden.

Was muss man wissen?

Um die Aufgabe meistern zu können, muss man zu aller erst verstehen, wie die Rohdaten aufgebaut sind. Wir bekommen Beispiele, auf welches unser Programm angewendet werden soll, in Form von Textdateien. Bei den Beispielen steht in der ersten Zeile immer eine Liste an Namen, getrennt durch ein Leerzeichen. Die Namen geben Aufschluss darüber, wer alles zu der Gruppe gehört.

In den weiteren Zeilen stehen immer die Beziehungen zueinander. Eine denkbare Zeile wäre z.B.: „Chantal Moritz“ dies steht dafür, dass Chantal, Moritz folgt.

Weiterhin sollte man wissen welche Bedingungen es gibt, um Superstar zu werden. Ein Superstar ist eine Person, dem alle anderen Personen einer bestimmten Gruppe folgen, er wiederum folgt niemandem aus dieser Gruppe.

Diese Bedingungen geben einen Hinweis darauf, dass es maximal einen Superstar pro Beispiel geben kann. Es ist unmöglich, dass zwei Personen Superstar sind, da sie nicht zur gleichen Zeit beide Voraussetzungen erfüllen können.

Lösungsidee

Wir haben uns, zur Veranschaulichung ein eigenes kleines Beispiel ausgedacht, dafür haben wir die folgenden Beziehungen genauer betrachtet:

/	**A folgt**	**B folgt**	**C folgt**	**D folgt**
A	/	Ja	Nein	Ja
B	Ja	/	Nein	Nein
C	Ja	Ja	/	Ja
D	Nein	Ja	Nein	/

Abbildung 1

Wir suchen also den Superstar in der Gruppe, bestehend aus den Personen A, B, C und D.

Die Beziehungen zueinander sind aus der Abbildung 1 abzulesen.

“Wie finden wir den Superstar, mit möglichst wenig abfragen?“

Unsere Idee besteht aus 2 Arbeitsschritten.

1. Schließe mit jeder gemachten Abfrage eine Person aus, bis nur noch einer übrig ist und merk dir was abgefragt wurde.
2. Finde heraus ob der potenzielle Superstar wirklich der Superstar ist.
    a. Folgt der potenzielle Superstar keinem?
    b. Folgen alle dem potenziellen Superstar?

Es gibt dann keinen Superstar, wenn mindestens eine der Fragestellungen, unter 2a oder 2b, nicht positiv beantwortet werden konnte.

Da in der Aufgabenstellung explizit jede andere Art der Abfrage, außer „Folgt Person X, Person Y?“, untersagt wurde, haben wir uns das folgende Schema ausgedacht.

Folgt A, B ?: *Ja*
A wird entfernt

Folgt B, C ?: *Ja*
B wird entfernt

Folgt C, D ?: *Nein*
D wird entfernt

C ist der potenzielle Superstar

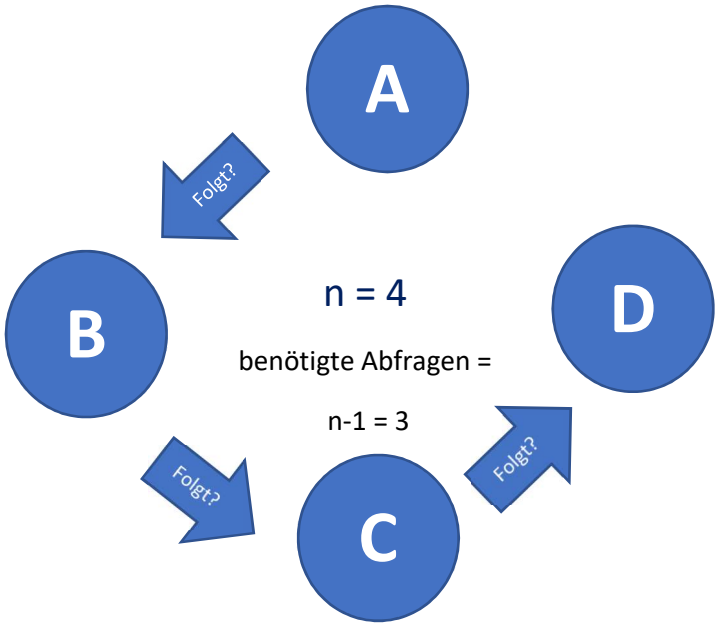

Abbildung 2

Wir gehen also die Liste an Personen durch und Fragen ab, ob Person X, Person Y folgt oder nicht.

Wenn ja, dann wird Person X aus der Liste entfernt. Die Bedingung, dass der Superstar keinem folgen darf ist für Person X also nicht mehr gegeben. Person X kann daher nicht der Superstar sein.

Wenn nein, dann wird Person Y aus der Liste entfernt. Die Bedingung, dass dem Superstar von jedem gefolgt wird, ist für Person Y also nicht mehr gegeben. Person Y kann daher nicht der Superstar sein.

Dieser Vorgang wird so lange wiederholt, bis nur noch eine Person in der Liste vorhanden ist.

In unserem Beispiel, bleibt nach 3 Abfragen nur die Person C übrig.

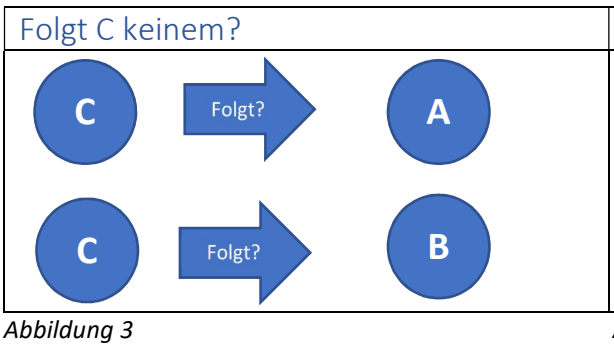

Abbildung 3

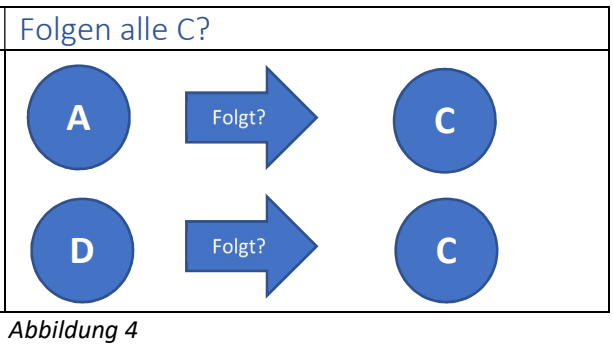

Abbildung 4

In unseren Abbildungen 3 und 4 ist zu sehen, dass wir nur 4 weitere Abfragen benötigen um sicher zu sein, dass Person C der Superstar ist.

Im eben genannte Beispiel entstanden 7 Euro an Kosten, für die Benutzung des Algorithmus. Diese 7 Euro ergeben sich aus den ersten 3 Abfragen (siehe Abb. 2) und den jeweils 2 Abfragen (siehe Abb. 3 und 4).

Die entstehenden Kosten können vorher mathematisch abgeschätzt werden.

Im besten Fall, gibt es keinen Superstar.

/	A folgt	B folgt	C folgt	D folgt
A	/	Ja	Ja	Ja
B	Ja	/	Ja	Ja
C	Ja	Ja	/	Ja
D	Ja	Ja	Ja	/

Abbildung 5

Hierbei benötigt unser Algorithmus eine Anzahl von n Abfragen. ‚n' steht für die Anzahl aller Mitglieder, in der zu überprüfenden Gruppe.

Im ersten Durchlauf, in welchem der potenzielle Superstar ermittelt wird, werden n-1 schritte benötigt, wenn alle allen folgen. Die eigentliche Konstellation, wer wem folgt, ist für das Endresultat irrelevant, da sich unser Programm die bereits getätigten abfragen merkt. Wenn etwas doppelt überprüft werden müsste, wird im weiteren Verlauf nur das vorherige Ergebnis abgerufen und demnach gehandelt.
Im zweiten Durchlauf, wird überprüft ob der potenzielle Superstar wirklich der Superstar ist. Nach einer weiteren Abfrage, wie zum Beispiel ob A unserem potenziellen Superstar D folgt, könnte es sein, dass D nicht der Superstar sein kann.

Die damit benötigten Abfragen entsprechen n-1+1, gekürzt also n.

Im besten Fall werden demnach n schritte benötigt, danach ist klar, dass es keinen Superstar gibt.

Im ungünstigsten Fall,

/	A folgt	B folgt	C folgt	D folgt
A	/	irrelevant	irrelevant	Nein
B	Ja	/	irrelevant	Nein
C	irrelevant	Ja	/	Ja
D	Ja	Ja	Ja	/

Abbildung 6

benötigt unser Algorithmus eine Anzahl von 3n-4 Abfragen.

Hier sind nach der Ermittlung des potenziellen Superstars, in diesem Fall D, wieder n-1 Schritte durchlaufen. Um zu überprüfen ob alle D folgen werden weiterhin n-2 Abfragen benötigt, und n-1 Abfragen gebraucht um zu schauen ob D keinem folgt. Das Endresultat, ob D der Superstar ist oder nicht, hat hierbei keinen Einfluss auf die Anzahl der Abfragen.

n-1+n-2+n-1 = 3n-4

Umsetzung

Bei der Umsetzung in Quellcode haben wir uns für C# entschieden, da wir mit Hilfe der objektorientierten Programmierung und seiner dynamischen Listen einen für den Menschen natürliche Programmstruktur verwirklichen konnten.

Ein Objekt der Klasse ‚Person' speichert:

- Seinen eigenen Namen
- Eine Liste an Personen, welchen er folgt
- Eine Liste an Personen, wer ihm folgt
- Eine Liste an Personen die schon überprüft wurden, mit A_Follows_Bt

In der Methode ‚FindSuperstar' wird eine Kopie der Liste aller Personen übergeben, und mit einer foreach Schleife durchlaufen.

Es ist zu beachten, dass die Methode rekursiv ist.

In jedem Durchgang, wird zuerst überprüft ob wir unseren Rekursionsanker bereits erreicht haben. Dieser tritt auf, wenn nur noch eine Person in der Liste vorhanden ist. Dann haben wir den potenziellen Superstar gefunden und weitere Methoden zur Validierung werden aufgerufen.
Wenn der Rekursionsanker noch nicht erreicht ist, dann wird mit einer weiteren for Schleife die Liste aller Personen, überprüft.

Sollte der Name der Person A aus der äußeren Schleife und der Person „B" aus der inneren nicht gleich sein, dann wird geschaut ob „A", "B" folgt. Wie vorhin bereits in der Lösungsidee erklärt, wird dann je nachdem eine Person aus der Liste entfernt. Das verursacht eine InvalidOperationException, diese fangen wir ab und die Methode ‚FindSuperstar' startet sich mit den veränderten Werten neu. So verkürzt sich die Liste mit jedem Durchgang um eine Person, bis am Ende nur noch eine Person übrig ist.

Ist der Rekursionsanker erst einmal erreicht, geht es an die Kontrolle. Denn noch können wir nicht wissen, ob der potenzielle Superstar wirklich der Superstar ist. Dafür wird die Methode ‚ValidateStar' aufgerufen, welche einen boolischen Wert zurück gibt und ein Objekt der Klasse ‚Person' an nimmt.

Diese hat ein 2 stufiges Validierungsverfahren. Im ersten Schritt wird geschaut ob alle dem potenziellen Superstar folgen. Dafür haben wir die Methode ‚EveryOneFollows', welche ebenfalls einen boolean zurück gibt. Ist diese Bedingung erfüllt, dann wird die Methode ‚FollowsNoOne' aufgerufen. Sie funktioniert nahezu identisch, wie die ‚EveryOneFollows', nur wird das Verfahren bei ihr gespiegelt angewendet. Ist diese Bedingung ebenfalls erfüllt, dann haben wir den Superstar gefunden.

Beispiele

Beispiel	Formel	1	2	3	4
Superstar	/	Justin	Dijkstra	Es gibt keinen	Folke
Tatsächliche Kosten in Euro	bA(n, i) = 2(n-1) + i - 2	4€	9€	8€	182€
Im besten Fall	bA(n) = n	n = 3	n = 5	n = 8	n = 80
Im ungünstigsten Fall	bA(n) = 3n-4	3*3-4 = = 5	3*5-4= =11	3*8-4 = = 20	3*80-4 = = 236

Abbildung 7 |bA = benötigte Abfragen; n = Anzahl Mitglieder; i = Position in der Liste der Gruppe

Quelltext

Teeniegram.cs

```
public static Person FindSuperstar() => FindSuperstar(new List<Person>(Person.AllUsers));

        private static Person FindSuperstar(List<Person> listOf_PersonB)
        {
            try
            {
                foreach (Person personA in listOf_PersonB)
                {
                    if (listOf_PersonB.Count != 1)
                    {
                        for (int i = listOf_PersonB.IndexOf(personA) + 1; i < listOf_PersonB.Count; i++)
                        {
                            if (personA.Name != listOf_PersonB[i].Name)
                            {
                                if (A_follows_B(personA, listOf_PersonB[i]))
                                {
                                    Console.WriteLine("entferne: " + personA.Name + "\n");
                                    listOf_PersonB.Remove(personA);
                                    //restart because the list size changed
                                    return FindSuperstar(new List<Person>(listOf_PersonB));
                                }
                                else
                                {
                                    Console.WriteLine("entferne: " + listOf_PersonB[i].Name + "\n");
                                    listOf_PersonB.Remove(listOf_PersonB[i]);
                                    //restart because the list size changed
                                    return FindSuperstar(new List<Person>(listOf_PersonB));
                                }
                            }
                        }
                    }
                    else
                    {
                        Console.WriteLine(personA.Name + " ist wahrscheinlich der superstar!"
                            + "\nÜberprüfe verdacht...\n"
                            );
                        Console.WriteLine(personA.Name +
                            " Index: " + Person.AllUsers.IndexOf(personA));
                        Console.WriteLine("Anzahl aller Benutzer: " + Person.AnzahlUser);

                        bool valid = ValidateStar(personA);
                        if (valid)
                        {
                            //Console.WriteLine(personA.Name + " ist der Superstar");
                            Console.WriteLine("Kosten in Euro:" + cost);
                            return personA;
                        }
                        else
                        {
                            Console.WriteLine(personA.Name + " ist nicht der Superstar");
                            Console.WriteLine("Kosten in Euro:" + cost);
                            return null;
                        }
                    }
                }
            }
            catch (System.InvalidOperationException)
            {
                return FindSuperstar(new List<Person>(listOf_PersonB));
            }

            //niemand ist der Superstar
            return null;
        }
```

```
private static bool ValidateStar(Person person)
        {
            //first validation
            if (EveryOneFollows(person))
            {
                Console.WriteLine("Alle folgen " + person.Name + "\n");
                //second validation
                if (FollowsNoOne(person))
                {
                    Console.WriteLine("\n" + person.Name + " folgt keinem!\n");
                    return true;
                }
                else
                {
                    Console.WriteLine($"{person.Name} folgt jemandem!");
                }
            }
            else
            {
                Console.WriteLine($"Nicht alle folgen {person.Name}!");
            }
            return false;
        }

        private static bool FollowsNoOne(Person person)
        {
            for (int i = 0; i < Person.AnzahlUser; i++)
            {
                if (person.Name != Person.AllUsers[i].Name)
                {
                    if (!person.IsInspected(Person.AllUsers[i]))
                    {
                        if (A_follows_B(person, Person.AllUsers[i]))
                        {
                            return false;
                        }
                    }
                }
            }
            return true;
        }

        private static bool EveryOneFollows(Person star)
        {
            foreach (Person B in Person.AllUsers)
            {
                if (!star.Equals(B))
                {
                    if (!B.IsInspected(star))
                    {
                        if (!A_follows_B(B, star))
                        {
                            return false;
                        }
                    }
                }
            }

            return true;
        }

        private static bool A_follows_B(Person a, Person b)
        {
            cost++;
            Console.WriteLine("Aufruf-Nummer: " + cost);
            Console.WriteLine("folgt: {0} : {1} ? [{2}]"
                              , a.Name
                              , b.Name,
                                a.Follows(b)
                              );

            a.Inspected.Add(b);
            return a.Follows(b);
        }
```