Aufgabe 5: Wichteln

Team-ID: 00523

Team: Einstein One

Bearbeiter/-innen dieser Aufgabe:
Stefan Cames

30. Oktober 2020

Inhaltsverzeichnis

Lösungsidee

Beim „Wichteln“ möchte jeder Schüler von einem bestimmten anderen Schüler ein Geschenk erhalten. Dabei hat jeder Schüler einen ersten, zweiten und dritten Wunsch und möchte entsprechend am liebsten, dass sein erster Wunsch erfüllt werden kann. Ziel ist es also, von so vielen Spielern wie möglich ihren ersten Wunsch zu erfüllen, ist dies bei bestimmten Schülern nicht mehr möglich, so viele zweite Wünsche wie möglich zu erfüllen und dann im gleichen Prinzip die dritten Wünsche der Schüler, von welchen erster und zweiter Wunsch nicht mehr verfügbar ist. Leider kommt es dabei auch vor, dass alle Wünsche eines Schülers besetzt sein können und er somit übrigbleibt.

Überlegt man sich, wie es am besten möglich ist, so viele Erstwünsche wie möglich zu verteilen, macht es am meisten Sinn, eindeutige Erstwünsche, welche nur ein Schüler hat, direkt zu verteilen. Haben wir hier alle Möglichkeiten verteilt, machen wir in gleicher Weise mit den zweiten Wünschen der Schüler weiter und schauen anschließend aber erst, ob nun wieder erste Wünsche eindeutig verteilt werden können, um zu garantieren, dass so viele Schüler wie möglich ihren ersten Wunsch erhalten. Letztendlich können wir mit dem gleichen Prinzip auch die dritten Wünsche der Schüler beachten und nach jedem zugewiesen Schüler prüfen, ob es vielleicht nun eindeutige Wünsche im ersten oder zweiten Wunsch gibt und verteilen diese so. Haben wir all dies fertig, macht es natürlich Sinn, auch noch einmal zu schauen, ob noch irgendein Wunsch erfüllbar ist, egal ob 1./2./3. Wunsch und erfüllen diese, um so vielen Schülern wie möglich ein Geschenk geben zu können.

Umsetzung

Setzt man die dargelegte Idee in Java um, so muss man als ersten Schritt die vorgegebene Textdatei erst einmal einlesen. Mittels eines FileReaders/BufferedReaders ist es möglich, anhand der Anzahl der Schüler, jedem Schüler einen Platz in einem Array zu geben und dabei auch all seine Wünsche in diesem mehrdimensionalen Array mit abzuspeichern. Dabei lassen wir auch noch zwei freie Felder, welche später angeben, ob und welcher seiner Wünsche erfüllt ist und welche Priorität dieser erfüllte Wunsch hat (1/2/3).

Haben wir nun das Array für die Schüler mit ihren Wünschen entsprechend vorbereitet, überprüfen wir zu Beginn erst einmal, ob es Schüler gibt, dessen erster Wunsch von keinem weiteren Schüler gewählt ist und erfüllen diese ersten Wünsche. Hier gehen wir in der Methode *earlyCheck()* mittels einer for-Schleife die ersten Wünsche aller Schüler durch und schauen für jeden Wunsch in einer anderen for-Schleife, ob dieser Wunsch noch einmal vorkommt, indem wir die Häufigkeit zählen. Ist der erste Wunsch eines Schülers schließlich nur bei ihm vorhanden, tragen wir im Array (wichteln[i][3]) für den Spieler i den Wunsch ein und bei wichteln[i][4] eine 1, um festzuhalten, dass dies sein erster Wunsch war. Zuletzt müssen wir diesen von nun an vergebenen Wunsch auch noch für alle anderen Schüler „sperren“, also kennzeichnen, dass dieser nicht mehr verfügbar ist. Dazu übergeben wir diesen vergebenen Wunsch der Methode *updateVergeben(int)*, speichern diesen in einer Arraylist *vergeben*<String> und gehen für jeden Schüler die vergebenen Wünsche durch und prüfen, ob der erste, zweite oder dritte Wunsch des jeweiligen Spielers sich eventuell bereits in der Liste der vergebenen Wünsche befindet. Ist dies der Fall, so ersetzen wir seinen Wunsch durch *0* um festzulegen, dass dieser vergeben ist.

Abgesehen von unserem nun ausgeführten *earlyCheck* besitzt unser Programm noch drei weitere Checks – für den ersten, zweiten und dritten Wunsch der Schüler – sowie eine Art „Notfallcheck“, welcher jegliche noch freien Wünsche unabhängig von der Priorität und Häufigkeit vergibt.

Die Methode *checkOnlySecondPriorität()* prüft mittels einer for-Schleife, ob es Schüler gibt, welchen nur noch der erste Wunsch möglich ist, da der zweite/dritte Wunsch bereits vergeben ist (*wichteln[i][1] == 0 && wichteln[i][2] == 0*) und verteilt diese, woraufhin der Wunsch mittels der Methode *updateVergeben*(wichteln[i][0]) für alle anderen Schüler gesperrt wird und die Methode beendet wird. Vorher wird hierbei noch geprüft, ob dem Schüler bereits ein Wunsch zugeordnet wurde und überspringt solche. Auch wird geprüft, ob alle Wünsche des Schülers bereits vergeben sind, also ob alle drei Wünsche *0* sind und setzt – ist dies der Fall – das Feld *wichteln*[i][3] auf *-1* um so festzuhalten, dass dieser Schüler keine Möglichkeiten mehr hat und am Ende übrig bleiben wird. Entsprechend wird *true* zurückgegeben, sobald ein Wunsch verteilt worden konnte, andernfalls *false*, wenn kein einziger Wunsch verteilt wurde.

Die Methode *checkOnlySecondPriority()* prüft hierbei genau wie die erste Methode in der for-Schleife zuerst, ob der jeweilige Schüler bereits einen Wunsch zugewiesen bekommen hat und, ob Schüler komplett rausfallen, also am Ende übrig bleiben. Anschließend wird geschaut, ob bei dem jeweiligen Schüler noch noch der zweite Wunsch erfüllbar ist, also ob *wichteln*[i][0] und *wichteln*[i][2] beide *0* sind, also vergeben. Falls dies bei einem Schüler zutrifft, erhält er den jeweiligen Wunsch, es wird die Methode *updateVergeben*(wichteln[i][1]) aufgerufen und

anschließend der aktuelle Durchlauf beendet, wobei wie bereits im letzten Check *true* zurückgeben wird, außer es konnte kein Schüler gefunden werden, für welchen nur der zweite Wunsch möglich ist – dann würde *false* zurückgegeben werden.

Die Methode checkOnlyThirdPriority() prüft letztendlich im gleichen Prinzip wie bisher, allerdings auf die Frage hin, ob der jeweilige Schüler nur noch seinen dritten Wunsch frei hat (*wichteln[i][0] == 0 && wichteln[i][1] == 0*) und verteilt diesen. Auch hier wird der dritte Wunsch für alle anderen Schüler nun gesperrt und die Methode wird beendet. Dabei wird auch hier entsprechend wieder *true* oder *false* zurückgegeben – je nachdem, ob ein Wunsch verteilt werden konnte oder nicht.

Letztendlich existiert noch eine Methode *finalNotfallCheck()*, welche auch für alle Spieler prüft, ob ihr erster Wunsch noch nicht vergeben ist, würde diesen entsprechend zuordnen, *updateVergeben()* ausführen und *true* zurückgeben. In gleicher Weise wird dies auch für den zweiten und schließlich für den dritten Wunsch geprüft. Besteht hier keine Möglichkeit mehr, wird *false* zurückgeben, womit das Programm beendet ist.

Wichtig anzumerken dabei ist hier, dass jede Methode nur solange prüft, bis ein Spieler gefunden wurde, auf den die jeweilige Bedingung zutrifft, setzt seine jeweiligen Befehle um und beendet dann die Methode und gibt *true* zurück, außer es nach Durchlauf aller Spieler niemand gefunden, auf den die Bedingung zutrifft, dann wird *false* zurückgegeben.

All diese soeben beschriebenen Methode werden nun nach folgendem Schema ausgeführt:

```
/*
Solange 1. Check true zurückgibt (Solange Schülern ihr 1. Wunsch erfüllt werden
kann), führe weiterhin den 1. Check aus

Ist dies nicht mehr möglich...:
Wenn der 2. Check true zurückgibt (Wenn einem Schüler sein 2. Wunsch erfüllt
werden kann), beginne wieder von Vorne

Ist dies nicht möglich...:
Wenn der 3. Check true zurückgibt (Wenn einem Schüler sein 3. Wunsch erfüllt
werden kann), beginne wieder von Vorne

Ist dies nicht möglich...:
Überprüfe einen letzten "Notfall"-Check, welcher dazu da ist, noch etwas zu ver-
teilen, wenn irgendwie möglich. Wenn dies möglich ist, beginne wieder von Vorne

Ist dies nicht möglich...:
Beende das Programm (setze active auf false)
*/
```

Beispiele

Siehe folgende Seiten…

Wichteln1.txt

1: 2 | 10 | 6
2: 2 | 7 | 3
3: 4 | 7 | 1
4: 3 | 4 | 9
5: 3 | 7 | 9
6: 4 | 3 | 2
7: 7 | 6 | 2
8: 10 | 2 | 4
9: 9 | 8 | 1
10: 4 | 9 | 6

7: 7 | 6 | 2 // 0. Check - 1. Wunsch nur einmal vorhanden -> Zuweisen
8: 10 | 2 | 4 // 0. Check - 1. Wunsch nur einmal vorhanden -> Zuweisen
9: 9 | 8 | 1 // 0. Check - 1. Wunsch nur einmal vorhanden -> Zuweisen
5: 3 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen
2: 2 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen
6: 4 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen
1: 0 | 0 | 6 // 3. Check - Dritte Priorität als einzige Möglichkeit? -> Zuweisen
3: 0 | 0 | 1 // 3. Check - Dritte Priorität als einzige Möglichkeit? -> Zuweisen
---
Der Schüler 1 erhält ein Geschenk von Schüler 6 - Das war sein 3er Wunsch
Der Schüler 2 erhält ein Geschenk von Schüler 2 - Das war sein 1er Wunsch
Der Schüler 3 erhält ein Geschenk von Schüler 1 - Das war sein 3er Wunsch
Der Schüler 4 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 5 erhält ein Geschenk von Schüler 3 - Das war sein 1er Wunsch
Der Schüler 6 erhält ein Geschenk von Schüler 4 - Das war sein 1er Wunsch
Der Schüler 7 erhält ein Geschenk von Schüler 7 - Das war sein 1er Wunsch
Der Schüler 8 erhält ein Geschenk von Schüler 10 - Das war sein 1er Wunsch
Der Schüler 9 erhält ein Geschenk von Schüler 9 - Das war sein 1er Wunsch
Der Schüler 10 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.

Normaler Programmablauf

1: 4 | 6 | 5
2: 5 | 4 | 6
3: 6 | 4 | 5
4: 6 | 4 | 5
5: 5 | 4 | 6
6: 4 | 6 | 5
7: 4 | 5 | 6
8: 5 | 4 | 6
9: 6 | 5 | 4
10: 4 | 5 | 6

1: 4 | 6 | 5 // Notfall-Check - Erste Priorität wird 'zufällig' verteilt -> Zuweisen
2: 5 | 0 | 6 // Notfall-Check - Erste Priorität wird 'zufällig' verteilt -> Zuweisen
3: 6 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen

---

Der Schüler 1 erhält ein Geschenk von Schüler 4 - Das war sein 1er Wunsch
Der Schüler 2 erhält ein Geschenk von Schüler 5 - Das war sein 1er Wunsch
Der Schüler 3 erhält ein Geschenk von Schüler 6 - Das war sein 1er Wunsch
Der Schüler 4 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 5 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 6 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 7 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 8 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 9 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 10 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.

Sonderfall: Bei 10 Schülern kommen insgesamt nur drei Wünsche vor, somit auch nur drei Wünsche insgesamt erfüllt werden… (Unter anderem hierfür ist der „finalNotfallCheck“)

1: 2 | 30 | 26
2: 20 | 29 | 30
3: 29 | 16 | 28
4: 8 | 6 | 30
5: 20 | 29 | 28
6: 3 | 4 | 30
7: 3 | 30 | 27
8: 12 | 10 | 14
9: 4 | 3 | 8
10: 28 | 18 | 29
11: 6 | 2 | 19
12: 9 | 30 | 15
13: 4 | 14 | 12
14: 3 | 23 | 26
15: 26 | 10 | 30
16: 30 | 20 | 8
17: 9 | 4 | 10
18: 8 | 7 | 30
19: 16 | 14 | 10
20: 10 | 29 | 13
21: 3 | 19 | 23
22: 8 | 4 | 6
23: 27 | 4 | 16
24: 3 | 15 | 26
25: 29 | 19 | 17
26: 26 | 27 | 20
27: 27 | 8 | 16
28: 27 | 21 | 17
29: 6 | 20 | 27
30: 30 | 10 | 26

1: 2 | 30 | 26 // 0. Check - 1. Wunsch nur einmal vorhanden -> Zuweisen
8: 12 | 10 | 14 // 0. Check - 1. Wunsch nur einmal vorhanden -> Zuweisen
10: 28 | 18 | 29 // 0. Check - 1. Wunsch nur einmal vorhanden -> Zuweisen
19: 16 | 14 | 10 // 0. Check - 1. Wunsch nur einmal vorhanden -> Zuweisen
20: 10 | 29 | 13 // 0. Check - 1. Wunsch nur einmal vorhanden -> Zuweisen
3: 29 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen
5: 20 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen
2: 0 | 0 | 30 // 3. Check - Dritte Priorität als einzige Möglichkeit? -> Zuweisen
15: 26 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen
26: 0 | 27 | 0 // 2. Check - Zweite Priorität als einzige Möglichkeit? -> Zuweisen
7: 3 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen
29: 6 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen
4: 8 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen
9: 4 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen
17: 9 | 0 | 0 // 1. Check - Erste Priorität als einzige Möglichkeit? -> Zuweisen
13: 0 | 14 | 0 // 2. Check - Zweite Priorität als einzige Möglichkeit? -> Zuweisen
14: 0 | 23 | 0 // 2. Check - Zweite Priorität als einzige Möglichkeit? -> Zuweisen
18: 0 | 7 | 0 // 2. Check - Zweite Priorität als einzige Möglichkeit? -> Zuweisen
21: 0 | 19 | 0 // 2. Check - Zweite Priorität als einzige Möglichkeit? -> Zuweisen

24: 0 | 15 | 0 // 2. Check - Zweite Priorität als einzige Möglichkeit? -> Zuweisen
25: 0 | 0 | 17 // 3. Check - Dritte Priorität als einzige Möglichkeit? -> Zuweisen
28: 0 | 21 | 0 // 2. Check - Zweite Priorität als einzige Möglichkeit? -> Zuweisen
---
Der Schüler 1 erhält ein Geschenk von Schüler 2 - Das war sein 1er Wunsch
Der Schüler 2 erhält ein Geschenk von Schüler 30 - Das war sein 3er Wunsch
Der Schüler 3 erhält ein Geschenk von Schüler 29 - Das war sein 1er Wunsch
Der Schüler 4 erhält ein Geschenk von Schüler 8 - Das war sein 1er Wunsch
Der Schüler 5 erhält ein Geschenk von Schüler 20 - Das war sein 1er Wunsch
Der Schüler 6 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 7 erhält ein Geschenk von Schüler 3 - Das war sein 1er Wunsch
Der Schüler 8 erhält ein Geschenk von Schüler 12 - Das war sein 1er Wunsch
Der Schüler 9 erhält ein Geschenk von Schüler 4 - Das war sein 1er Wunsch
Der Schüler 10 erhält ein Geschenk von Schüler 28 - Das war sein 1er Wunsch
Der Schüler 11 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 12 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 13 erhält ein Geschenk von Schüler 14 - Das war sein 2er Wunsch
Der Schüler 14 erhält ein Geschenk von Schüler 23 - Das war sein 2er Wunsch
Der Schüler 15 erhält ein Geschenk von Schüler 26 - Das war sein 1er Wunsch
Der Schüler 16 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 17 erhält ein Geschenk von Schüler 9 - Das war sein 1er Wunsch
Der Schüler 18 erhält ein Geschenk von Schüler 7 - Das war sein 2er Wunsch
Der Schüler 19 erhält ein Geschenk von Schüler 16 - Das war sein 1er Wunsch
Der Schüler 20 erhält ein Geschenk von Schüler 10 - Das war sein 1er Wunsch
Der Schüler 21 erhält ein Geschenk von Schüler 19 - Das war sein 2er Wunsch
Der Schüler 22 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 23 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 24 erhält ein Geschenk von Schüler 15 - Das war sein 2er Wunsch
Der Schüler 25 erhält ein Geschenk von Schüler 17 - Das war sein 3er Wunsch
Der Schüler 26 erhält ein Geschenk von Schüler 27 - Das war sein 2er Wunsch
Der Schüler 27 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.
Der Schüler 28 erhält ein Geschenk von Schüler 21 - Das war sein 2er Wunsch
Der Schüler 29 erhält ein Geschenk von Schüler 6 - Das war sein 1er Wunsch
Der Schüler 30 konnte leider nicht zugeteilt werden, da alle seine Wünsche besetzt sind.

Normaler Programmablauf: Alle Wünsche werden nach Möglichkeit in den Prioritäten 1. Wunsch, dann 2. Wunsch und anschließend 3. Wunsch verteilt. Manche Schüler bleiben leider übrig, da all ihre Wünsche bereits besetzt sind.

Quellcode

Der Quellcode befindet sich im Anhang (Main.java).