Aufgabe 3: Voll Daneben

Team-ID: 00381

Team-Name: EinSteine

Bearbeiter/-innen dieser Aufgabe:
Oliver Schirmer, Konrad Walter, Paskal Paesler, Corey Attireh

20. November 2018

Inhaltsverzeichnis

Lösungsidee

Da die Glückszahlen im Bereich von 1 bis 1000 liegen und für Al 10 Zahlen ausgewählt werden sollen, lässt sich der Gesamtbereich in 10 gleichmäßige Teilbereiche unterteilen. In diesem Fall also 1-100, 101-200, 201-300 usw. Dafür werden 10 kleinere Listen erstellt und alle Glückszahlen in die jeweiligen Bereiche eingeordnet. Um nun die Distanz, und damit die Kosten, zu minimieren, muss ein Schwerpunkt in dem jeweiligen Bereich ermittelt werden. Dafür werden alle Zahlen in dem Bereich addiert und am Ende durch ihre Anzahl geteilt. Damit erhalten wir für jeden Bereich einen Durchschnitt.

Durch die Unterteilung in die 10 Bereiche wird jedoch kein optimales Ergebnis erreicht, da sich Schwerpunkte bei Betrachtung des gesamten Zahlenbereichs auch über mehrere Bereiche erstrecken. Das Berechnen der Durchschnitte, damit der Akzente, für jeden Bereich ist natürlich hilfreich, reicht aber nicht aus, um eine gesamte Ausgewogenheit herzustellen.
Um dem entgegenzuwirken wurde eine Art Gravitationssystem entwickelt. Da eine von Al‘s Zahlen jeweils einen Zahlenbereich abdeckt, genügt es die „Gravitation“ zu den benachbarten Bereichen zu ermitteln.
Dafür muss grundlegend die Konzentration der Zahlen für jeden Bereich berechnet werden. Das geschieht durch das Bilden des Verhältnisses von tatsächlicher Glückszahl-Anzahl in einem Bereich und durchschnittlicher Glückszahl-Anzahl, welche man errechnen kann, indem man die Anzahl aller Glückszahlen durch 10 teilt.

Nachdem dann der Durchschnitt für jeden Bereich ermittelt wurde, werden die Differenzen der Konzentration des momentanen Bereichs und der Konzentration des darüber & des darunter liegenden Bereichs gebildet. Damit erhält man die Stärke, von der der momentane Bereich von den anliegenden Bereichen „angezogen“ wird. Von diesen beiden Differenzen, wird nochmals die Differenz gebildet, wodurch sich die endgültige Ausrichtung des momentanen Bereichs ergibt.
Um wie viele Zahlen der Durchschnitt jetzt nach oben bzw. unten geändert wird hängt von der Anzahl der Glückszahlen im Bereich ab. D.h. die errechnete Ausrichtung/Gravitation wird mit der Hälfte der Anzahl der Glückszahlen multipliziert.

Umsetzung

Für die anfängliche Sortierung wird für jeden Bereich eine Ziffer festgelegt: Für 1-100 0, 101-200 1, 201-300 2 usw. Dann wird über alle Glückszahlen iteriert und diese in den jeweiligen Bereich einsortiert, d.h. der Liste für diesen Bereich hinzugefügt.

Bei der Ermittlung der Konzentration/Wichtung ist nur zu erwähnen, dass die errechneten Werte als Schlüssel-Wert-Paare gespeichert werden: Bereich-ID & Konzentration – bspw. 0 & 0.9; 1 & 1.43

Die Errechnung des Durchschnitts erfolgt mithilfe eines BigInteger-Objekts, da dieses gewährleistet, dass (theoretisch) unendlich viele Zahlen addiert werden können und keine Fehler auftreten, im Gegensatz zu einem Integer- bzw. Long-Objekt.
Der Durchschnitt selbst ist eine Zahl in dem jeweiligen Bereich, weshalb dieser maximal den Wert 1000 annehmen kann und somit als Integer gespeichert werden kann. Da ein BigInteger jedoch nur als String ausgegeben werden kann muss dieser String zu einem Integer mithilfe der Integer.parseInt Methode umgewandelt werden.

Bei der Ermittlung der „Gravitation“ sind nur einige Umsetzungsdetails zu erwähnen:
Für die Berechnung der Gravitation zum nächstgelegenen Bereich wird darauf geachtet, dass es sich beim momentanen Bereich nicht um den letzten Bereich handelt. Ist dies der Fall wird der Wert „0“ eingesetzt. Das gleiche wird bei der Berechnung der Gravitation zum vorherigen Bereich gemacht, wenn es sich beim momentanen Bereich um den ersten Bereich handelt.
Zusätzlich, liegt die Gravitation im negativen Bereich, wird diese auch auf 0 gesetzt. Somit ist gewährleistet, dass es keinen negativen Gravitationswert, also „Abstoßung“ gibt, was sich in unserem Beispiel negativ auf den Gewinn Al‘s auswirken würde.
Die Differenz der beiden Gravitationswerte kann durchaus negativ sein, was jedoch nur für die Richtung, in die der Durchschnitt verschoben wird steht, es sich jedoch trotz dessen um eine Verschiebung, aufgrund von hohen Konzentrationen in benachbarten Bereichen, handelt.
Bei der Berechnung der tatsächlichen Verschiebung, wurde das Produkt aus der Hälfte der Gruppengröße und der Gravitation gebildet. Warum die Hälfte der Gruppengröße gewählt wurde, lässt sich an der modellhaften Vorstellung festmachen, dass der Durchschnitt eines jeden Bereiches, bei einer komplett gleichen Verteilung, exakt in der Mitte des Bereiches liegen würde und die Verschiebung in Richtung eines benachbarten Bereiches jeweils die Hälfte der Bereichsgröße in eine Richtung betragen würde.
Die Multiplikation mit der Gravitation lässt sich daran erklären, dass es sich bei der „Gravitation“ nur um einen Faktor handelt, der angibt inwiefern ein benachbarter Bereich eine höhere Konzentration an Glückszahlen besitzt als der momentane.

Beispiele

In diesem Abschnitt gehe ich auf die Originalbeispiele der Webseite (https://bwinf.de/bundeswettbewerb/37-bwinf/1-runde/material-371/) ein:

In der Praxis lässt sich nun erkennen, dass die Ergebnisse nicht ganz optimal sind. Im 1. Beispiel müssten die Zahlen auf 50 enden, um ein optimales Ergebnis von 25$ Gewinn zu erhalten.
Das liegt an den Rundungsungenauigkeiten des BigInteger-Objekts und der nicht mit einkalkulierten unterschiedlichen Gruppengrößen, bei einer nicht durch 10 teilbaren Anzahl an Glückszahlen, welche die Gravitation beeinflussen. Das 1. Problem würde sich durch die Verwendung der BigDecimal Klasse und das 2. Problem durch das Nutzen einer Verhältnisgleichung lösen lassen, jedoch ist die Abweichung minimal und die Umsetzung würde das Programm nur unnötig verkomplizieren.

Beispiel 1	Beispiel 2	Beispiel 3
---ERGEBNIS--- Glückszahlen: 199 Einzahlung: 4975$ Al's Zahlen: 52 152 252 352 452 552 652 752 852 949 Auszahlung: 4951$ Saldo: 24$ Al Capone: Gotcha!	---ERGEBNIS--- Glückszahlen: 100 Einzahlung: 2500$ Al's Zahlen: 55 155 246 354 428 549 666 763 849 941 Auszahlung: 2261$ Saldo: 239$ Al Capone: Gotcha!	---ERGEBNIS--- Glückszahlen: 100 Einzahlung: 2500$ Al's Zahlen: 66 156 256 362 459 551 662 750 860 959 Auszahlung: 2396$ Saldo: 104$ Al Capone: Gotcha!

Quellcode

```
package de.ncrypted.volldaneben;

/**
 * @author ncrypted
 */
public class VollDaneben {

    private static List<Integer> luckNumbers = new ArrayList<>();
    private static List<Integer> alNumbers = new ArrayList<>();

    public static void main(String[] args) {
        readLuckNumbers(numbersFileName);
        getAlNumbers();
        System.out.println("---ERGEBNIS---");
        int deposit = luckNumbers.size() * 25;
        int payout = 0;
        for (Integer luckNumber : luckNumbers) {
            payout += getPayout(luckNumber);
        }
        int saldo = deposit - payout;
        System.out.println("Glückszahlen: " + luckNumbers.size());
        System.out.println("Einzahlung: " + deposit + "$");
        System.out.println("Al's Zahlen:");
        alNumbers.forEach(number -> System.out.println(number));
        System.out.println("Auszahlung: " + payout + "$");
        System.out.println("Saldo: " + saldo + "$");
        System.out.println("Al Capone: " + (saldo > 0 ?
"Gotcha!" : saldo == 0 ? "Grmpf..." : "RAGE!"));
    }

    private static void getAlNumbers() {
        alNumbers.clear();
        List<List<Integer>> luckNumberGroups = new ArrayList<>();

        // Erstelle die Glueckszahlen-Gruppen
        // 0: 1-100 | 1: 101-200 | 2: 201-300 ...
        for (int i = 0; i < 10; i++) {
```

```
            luckNumberGroups.add(i, new ArrayList<>());
            for (Integer luckNumber : luckNumbers) {
                if (luckNumber > i * 100 && luckNumber <= i * 100
+ 100) {
                    luckNumberGroups.get(i).add(luckNumber);
                }
            }
        }

        // Ermittle Wichtung je Bereich (100 Zahlen), aufgrund
vieler Glueckszahlen
        // 0: 1-100 | 1: 101-200 | 2: 201-300 ... || Konzentration
verglichen zum Durchschnitt aller Zahlen
        Map<Integer, Float> concentration = new HashMap<>();
        int total = luckNumbers.size();
        float groupSize = (float) total / 10;
        for (int i = 0; i < 10; i++) {
            int count = luckNumberGroups.get(i).size();
            concentration.put(i, count / groupSize);
        }

        // Bilde den Durchschnitt der Glueckszahl-Gruppen,
        // verschiebe Sie in Richtung höher konzentrierterer
benachbarter Bereiche,
        // und setze sie als Al's Zahl
        for (int i = 0; i < luckNumberGroups.size(); i++) {
            List<Integer> luckNumberGroup =
luckNumberGroups.get(i);
            BigInteger sum = BigInteger.ZERO;
            for (Integer luckNumber : luckNumberGroup) {
                sum = sum.add(BigInteger.valueOf(luckNumber));
            }
            int ratio =
Integer.parseInt(sum.divide(BigInteger.valueOf(luckNumberGroup.siz
e())).toString());

            // Ermittle "Gravitation"
            float conc = concentration.get(i);
            float gravPos = 0;
            if (i != luckNumberGroups.size() - 1) {
                gravPos = concentration.get(i + 1) - conc;
            }
            gravPos = gravPos < 0 ? 0 : gravPos;
            float gravNeg = 0;
            if (i != 0) {
                gravNeg = concentration.get(i - 1) - conc;
            }
            gravNeg = gravNeg < 0 ? 0 : gravNeg;
            float grav = gravPos - gravNeg;
            int gravSteps = (int)
Math.floor(luckNumberGroup.size() / 2F * grav);
            ratio += gravSteps;

            alNumbers.add(ratio);
        }
    }

    private static int getPayout(int luckNumber) {
        int distance = -1;
        for (Integer alNumber : alNumbers) {
            int currentDistance = Math.abs(alNumber - luckNumber);
```

```
                if (distance == -1 || currentDistance < distance) {
                    distance = currentDistance;
                }
            }
            return distance;
        }
    }
```