Junioraufgabe 2

Team-ID: 00112

Team-Name: LonelyFlo4TheWin

Bearbeitet von:
Florian Bange

22. November 2021

Inhaltsverzeichnis

1 Loesungsidee

1.1 Grundidee

Um das Problem der zweiten Junioraufgabe loesen zu koennen, muss man herausfinden, welches Datum den meisten Personen am besten gefaellt. Dies kann man machen, indem man zunaechst herausfindet, wie vielen Personen welches Datum nicht am besten gefaellt. Denn dann ist logischerweise erkennbar, welches Datum den wenigsten Personen am nicht am besten gefaellt.

Die Anzahl der Personen, welchen das Datum nicht am besten gefaellt, ist die Anzahl an Veraenderungen, die man taetigen muss.

Um herauszufinden, welche Daten welchen Personen nicht am besten gefallen, muss man fuer jede Person:

1. Die best-passendste Farbe finden (wenn es gruen gibt, dann gruen; wenn es kein gruen gibt, aber gelb, dann gelb; wenn es kein gruen und kein gelb gibt, dann rot).
2. Die best-passendste Farbe mit der Farbe jedes Datums vergleichen. Wenn die best-passendste Farbe besser als die des verglichenen Datums ist, kann man sich das Datum, als für die Person nicht perfekt passend, merken.

Nun kann man zaehlen, welches Datum wie oft als nicht perfekt passend gemerkt wurde. Das Datum, welches am wenigsten Personen nicht perfekt gefaellt, wird ausgewaehlt.

1.2 Bezug zum Beispiel des Aufagbenblatts

Im Beispiel wuerde das fuer Edsger so aussehen:

1. Gruen ist die best-passendste Farbe.
2. Datum 2, 3, 4 und 5 haben eine „schlechtere“ Farbe als gruen und passen Edsger somit schlechter als gruen.

Wenn man dies fuer alle Personen wiederholt, erfaehrt man, dass:

- Datum eins 4 Personen nicht am besten gefaellt
- Datum zwei 3 Personenn icht am besten gefaellt
- Datum drei 3 Personen nicht am besten gefaellt
- Datum vier 3 Personen nicht am besten gefaellt
- Datum fuenf 4 Personen nicht am besten gefaellt
- Datum sechs 2 Personen nicht am besten gefaellt
- und Datum sieben 3 Personen nicht am besten gefaellt

Somit ist Datum sechs das Datum, das den meisten Personen am besten gefaellt und man weisz, dass man zwei Eintraege veraerndern muss.

2 Umsetzung

Die Loesungsidee habe ich in Java 8 implementiert.
Am Ende des Folgenden kann man das „perfekte“ Datum und die Anzahl der benoetigten Aenderungen aus einer Instanz der Klasse „JuniorAufgabe2“ entnehmen.

2.1 Initialisierung

Zunaechst speichere ich die Werte der Tabelle aus der Datei, dessen Name angegeben werden kann, in ein zweidimensionales Integer-Array („table“) sowie die Anzahl an Personen und Daten als Integer. Alle drei als fields.

2.2 Benoetigte Aenderungen pro Datum

Nun benutze ich die Methode „getNeededChangesPerDate“ in welcher ich Folgendes tue:
Ich initialisiere ein eindimensionales Integer-Array („changesByDate“) mit der Laenge der Anzahl der Daten. Daraufhin iteriere ich fuer jede Person durch das 2D-Array („table“). Dabei gehe ich fuer jede Person durch das Array der Person - also durch die Angabe, welches Datum der Person wie gut gefaellt - und ermittle den besten/kleinsten Wert. Danach (noch immer fuer die selbe Person) gehe ich erneut durch das Array der Daten der Person und erhoehe „changesByDate“ am Index des jeweiligen Datums, wenn der Wert, den die Person fuer dieses Datum angegeben hat, groeßer/schlechter als der beste Wert ist (welchen wir zuvor ermittelt haben).

2.3 Bestes Datum finden

Schlieszlich finde ich das Datum mit den wenigsten Aenderungen, indem ich zunaechst den Integer „bestDate“ mit 0 und den Integer „changes“ mit dem Wert des Arrays „changesByDate“ an der Stelle 0 initialisiere. Danach iteriere ich durch das Array „changesByDate“ anfangend bei 1 und passe „bestDate“ und „changes“ immer an, wenn ein Wert im Array kleiner ist als der aktuelle Wert von „changes“.
Diese Werte gebe ich anschließend in einem Integer-Array zurueck, welches an erster Stelle „bestDate + 1“ (um die standartmaeszige Zaehlweise beizubehalten, +1) und an zweiter Stelle „changes“ enthaelt.

3 Beispiele

3.1 Aufgabe des Aufgabenblatts/praeferenzen0.txt

Bestes Datum: 6, noetige Veraenderungen: 2

3.2 praeferenzen1.txt

Bestes Datum: 2, noetige Veraenderungen: 1

3.3 praeferenzen2.txt

Bestes Datum: 4, noetige Veraenderungen: 0

3.4 praeferenzen3.txt

Bestes Datum: 18, noetige Veraenderungen: 7

3.5 praeferenzen4.txt

Bestes Datum: 22, noetige Veraenderungen: 14

3.6 praeferenzen5.txt

Bestes Datum: 31, noetige Veraenderungen: 34

4 Quellcode

4.1 Initialisierungen

4.1.1 Initialisierungen der Anzahl der Mitglieder und Daten

```
    // Get members (members) and dates (dates) amount
    String[] firstLineSplitted = lines.get(0).split("␣");
    this.members = Integer.parseInt(firstLineSplitted[0]);
    this.dates = Integer.parseInt(firstLineSplitted[1]);
```

4.1.2 Initialisierungen des zweidimensionalen Arrays

```
    // Initialize table
    this.table = new int[this.members][this.dates];

    // Fill table with values
    for (int i = 0; i < this.members; i++) {
      // Get the current rows values
      String[] values = lines.get(i + 1).split("␣");
      // Fill current row with values
      for (int j = 0; j < this.dates; j++)
        this.table[i][j] = Integer.parseInt(values[j]);
    }
```

4.2 Benoetigte Aenderungen pro Datum

```
// Fill changesByDate with count of members who do not find this date perfect
private void getNeededChangesPerDate() {
  // Initialize changesByDate
  this.changesByDate = new int[this.dates];
  // Go through all members
  for (int i = 0; i < this.members; i++) {
    // Get best value for current member
    int lowest = -1;
    for (int j = 0; j < this.dates; j++) {
      int current = this.table[i][j];
      if (current < lowest || lowest == -1)
        lowest = current;
    }

    // Increases changesByDate[j] by one if you would have to make a change for
    // current date for current member
    for (int j = 0; j < this.dates; j++)
      if (this.table[i][j] > lowest)
        this.changesByDate[j]++;
  }
}
```

4.3 Bestes Datum finden

```
// Returns integer array with (index 0) the best date (index of date +1) and
// (index 1) the changes you have to make for this date
private int[] getBestDate() {
  // Get best date by changes to make
  int bestDate = -1, changes = -1;
  for (int i = 0; i < this.changesByDate.length; i++) {
    int current = this.changesByDate[i];
    if (current < changes || bestDate == -1) {
      bestDate = i;
      changes = current;
    }
  }

  return new int[] { bestDate + 1, changes };
}
```