Aufgabe 5: Wortsuche-Dokument

Team-ID: 00070

Team: Gaußgamz

Bearbeiter/-innen dieser Aufgabe:

Nils Stäcker, Manuel Pielka, Benedict Weis

16. Oktober 2021

Lösungsidee

Um die Aufgabe 5 zu lösen, haben wir uns zu jedem Gewicht zwei weitere Character gemerkt. Diese Characters können entweder den Wert 0, 1 oder # tragen.

Anfangs werden dem ersten Gewicht eine 1 und eine 0 zugewiesen. Bei den restlichen Gewichten werden jeweils eine 0 in beide Characters eingetragen.

Letztendlich wird dann jedes Gewicht, bei dem der erste Character eine 1 ist, auf die rechte Seite und jedes Gewicht bei dem der zweite Character eine 1 ist, auf die linke Seite gebracht.

Das Gewicht, welches nun abgemessen werden kann, wird abgespeichert. Dann erst wird der zweite Character, so wie beim binär addieren (nur von links nach rechts), um eins erhöht.

Hierbei wird beachtet, dass der zweite Character keine 1 sein kann, solange der erste Character des dazugehörigen Gewichts eine 1 ist.

Dies wiederholt sich so lange, bis beide Character ihr Maximum erreicht haben. Jetzt wird geschaut, ob das Gewicht von 10g bis 10000g in Zehnerschritten bereits abgespeichert ist und wenn es nicht abgespeichert ist, sucht man sich dann das nächste Ergebnis heraus.

Umsetzung

Zuerst haben wir mithilfe der Klassen „java.util.Scanner“ und „java.io.File“ die Gewichtsstücke eingelesen. Die Gewichte haben wir uns dann so in eine ArrayList abgespeichert, dass wenn wir zweimal das selbe Gewicht haben, dass das Gewicht dann auch zwei Mal in der ArrayList abgespeichert wird. Um uns die extra Character zu speichern, haben wir zwei Character-Arrays erstellt, die genau so lang sind, wie wir auch einzelne Gewichtsstücke haben. Um unsere Zwischenergebnisse, so wie auch die Endergebnisse werden in einer java.util.HashMap gespeichert.

Um sicherzugehen, dass alle möglichen Kombinationen der Characters in den beiden Char Arrays zu berechnen, haben wir zwei ineinander geschachtelte while-Schleifen geschrieben.

Hinweis zum Ausführen

Zum Ausführen des Programms rufen sie bitte die Main Klasse auf und schreiben hinter die Klasse das gewünschte Beispiel oder das Wort „all“.

Bitte wundern sie sich außerdem nicht, falls beim letzten Beispiel erstmal nichts nach dem Methodenaufruf passiert. Die Programmlaufzeit ist aufgrund von den vielen und sehr hohen Gewichten lange. Bei mir hat das Programm ungefähr 30sek gebraucht, um fertig zu werden.

Beispiele

Hier ist ein Ausschnitt der Ausgabe für gewichtsstuecke0.txt.

```
Gewichtsstuecke 0:
10g ist moeglich
linke Seite:
rechte Seite:   10


20g ist moeglich
linke Seite:
rechte Seite:   10    10


30g ist moeglich
linke Seite:
rechte Seite:   10    10    10


40g ist moeglich
linke Seite:    10
rechte Seite:   50
```

Bei 9940g tritt es zum ersten Mal auf, dass ein Gewicht nicht möglich ist. Dies sieht dann so aus:

```
9940g ist nicht moeglich
nõchster Abstand: 10
```

Um die ganze Ausgabe anzusehen, öffnen sie bitte die Textdatei: output.txt

Quellcode

Der Code zum Einlesen der Datei:

```
private void einlesen(String input) {
      File file = new File(input);
      lines = new ArrayList<String>();
```

```
        try {
            Scanner scanner = new Scanner(file);
            while (scanner.hasNext()) {
                parts = (scanner.nextLine()).split(" ");
                if (parts.length == 2) {
                    for (int i = 0; i < Integer.parseInt(parts[1]); i++) {
                        lines.add(parts[0]);
                    }
                } else {
                    lines.add(parts[0]);
                }
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
```

Hier der Code um alle möglichen Ergebnisse berechnet:

```
public void alleberechnen() {
        long gegengewicht = 0;
        long gewichtsadd = 0;
        for (int i = 0; i < binarGG.length; i++) {
            binarGG[i] = '0';
            binarGA[i] = '0';
        }
        binarGG[0] = '1';

        while (true) {
            if (binarGG[0] == '#') {
                return;
            }
            for (int i = 0; i < binarGA.length; i++) {
                binarGA[i] = '0';
            }
            gewichtsadd = 0;

            gegengewicht = 0;
            for (int i = 0; i < binarGG.length; i++) {
                if (binarGG[i] == '1') {
                    gegengewicht += Long.parseLong(lines.get(i + 1));
                }
            }

            while (true) {
                long gewicht = gegengewicht - gewichtsadd;
                if (gewicht > 10000) {
```

```
            if ((gewicht < groesstesGewicht)) {
                alleErgebnis.remove(groesstesGewicht);
                alleErgebnisseKeys.remove(groesstesGewicht);
                Daten neu = new Daten(gewicht);
                if (!alleErgebnis.containsKey(gewicht)) {
                    alleErgebnis.put(gewicht, neu);
                    alleErgebnisseKeys.add(gewicht);
                }
            }
        } else {
            Daten neu = new Daten(gewicht);
            if ((gewicht % 10 == 0) && (gewicht <= 10000)) {
                neu.links = binarGA.clone();
                neu.rechts = binarGG.clone();
            }
            if (!alleErgebnis.containsKey(gewicht)) {
                alleErgebnis.put(gewicht, neu);
                alleErgebnisseKeys.add(gewicht);
            }
        }
        if (gewicht < groesstesGewicht && gewicht > 10000) {
            groesstesGewicht = gewicht;
        }

        long neuGA = Long.valueOf(gewichtsadd);
        while (neuGA <= gewichtsadd) {
            binarGA = trinarAddieren(binarGA, binarGG);
            neuGA = 0;
            if (binarGA[0] == '#') {
                break;
            }
            for (int i = 0; i < binarGG.length; i++) {
                if (binarGA[i] == '1') {
                    neuGA += Long.parseLong(lines.get(i + 1));
                }
            }
        }
        if (binarGA[0] == '#') {
            break;
        }
        gewichtsadd = Long.valueOf(neuGA);
    }
    long neuGG = Long.valueOf(gegengewicht);
    while (neuGG <= gegengewicht) {
        binarGG = binarAddieren(binarGG);
        neuGG = 0;
        if (binarGG[0] == '#') {
            break;
```

```
            }
            for (int i = 0; i < binarGG.length; i++) {
                if (binarGG[i] == '1') {
                    neuGG += Long.parseLong(lines.get(i + 1));
                }
            }
        }
        if (binarGG[0] == '#') {
            break;
        }
        gegengewicht = Long.valueOf(neuGG);
    }
}
```

Hier der Code, der alle aus allen Ergebnisse nur die gewollten holt:

```
 public void umrechnen() {
    Collections.sort(alleErgebnisseKeys);
    for (int i = 10; i <= 10000; i += 10) {
        if (alleErgebnis.containsKey(Long.valueOf(i))) {
            zehnerSchritte.put(i, alleErgebnis.get(Long.valueOf(i)));
        } else {
            int index = 0;
            for (int h = 0; h < alleErgebnisseKeys.size() - 1; h++) {
                if (i < alleErgebnisseKeys.get(h)) {
                    break;
                }
                index = h;
            }
            long abstandK = (i - alleErgebnisseKeys.get(index));
            long abstandG = (alleErgebnisseKeys.get(index + 1) - i);

            if (abstandK <= abstandG) {
                zehnerSchritte.put(i, alleErgebnis.get(Long.valueOf(alleErgebnisseKeys.get(index))));
            } else {
                zehnerSchritte.put(i, alleErgebnis.get(Long.valueOf(alleErgebnisseKeys.get(index + 1))));
            }
        }
    }
}
```

Hier der Code, der die Ergebnisse dann ausgibt:

```
public void ausgabe() {
    for (int i = 10; i <= 10000; i += 10) {
        String rechts = "";
```

```
            String links = "";

            Daten aktuell = zehnerSchritte.get(i);
            if (aktuell.gewicht == i) {
                System.out.println(i + "g ist moeglich");
                for (int h = 0; h < aktuell.rechts.length; h++) {
                    if (aktuell.rechts[h] == '1') {
                        rechts += (lines.get(h + 1) + "    ");
                    }
                }

                for (int h = 0; h < aktuell.links.length; h++) {
                    if (aktuell.links[h] == '1') {
                        links += (lines.get(h + 1) + "    ");
                    }
                }

                System.out.println("linke Seite:\t" + links);
                System.out.println("rechte Seite:\t" + rechts);
                System.out.println();

            } else {
                System.out.println(i + "g ist nicht moeglich");
                System.out.println("naechster Abstand: " + Math.abs((aktuell.gewicht - i)));
            }
            System.out.println();
        }
    }
```