Aufgabe 4

Das GUTSCHEIN-Problem ist gegeben durch eine Folge $w_1, \ldots, w_n$ von Warenwerten (wobei $w \in N_0$ für $i = 1, \ldots, n$) und einem Gutscheinbetrag $G \in \mathbb{N}_0$.

Da Gutscheine nicht in Bargeld ausgezahlt werden können, ist die Frage, ob man eine Teilfolge der Waren findet, sodass man genau den Gutschein ausnutzt. Formal ist dies die Frage, ob es eine Menge von Indizes I mit $I \subseteq \{1, \ldots, n\}$ gibt, sodass $\sum_{i \in I} w_i = G$

(a) Sei $w_1 = 10, w_2 = 30, w_3 = 40, w_4 = 20, w_5 = 15$ eine Folge von Warenwerten.

(i) Geben Sie einen Gutscheinbetrag $40 < G < 115$ an, sodass die GUTSCHEIN-Instanz eine Lösung hat. Geben Sie auch die lösende Menge $I \subseteq \{1, 2, 3, 4, 5\}$ von Indizes an.

> 50
> $I = \{1, 3\}$

(ii) Geben Sie einen Gutscheinbetrag G mit $40 < G < 115$ an, sodass die GUTSCHEIN-Instanz keine Lösung hat.

> 51

(b) Sei *table* eine $(n \times (G + 1))$-Tabelle mit Einträgen table$[i, k]$, für $1 < i < n$ und $0 \leq k \leq G$, sodass

$$\text{table}[i,k] = \begin{cases} \textbf{true} & \text{falls es } I \subseteq \{1,\ldots,n\} \text{ mit } \sum_{i \in I} w_i = G \text{ gibt} \\ \textbf{false} & \text{sonst} \end{cases}$$

Geben Sie einen Algorithmus in Pseudo-Code oder Java an, der die Tabelle *table* mit *dynamischer Programmierung* in Worst-Case-Laufzeit $\mathcal{O}(n \times G))$ erzeugt. Begründen Sie die Korrektheit und die Laufzeit Ihres Algorithmus. Welcher Eintrag in *table* löst das GUTSCHEIN-Problem?

```
3  import org.bschlangaul.helfer.Konsole;
4
5  public class Gutschein {
6    /**
7     * Das GUTSCHEIN-Betrag von 0, 1, ...
8     */
9    int gutscheinBetrag;
10
11   /**
12    * Das GUTSCHEIN-Problem ist gegeben durch eine Folge w1, ..., wn
   ↪  von
13    * Warenwerten.
14    */
15   int[] warenWerten;
16
17   boolean[][] tabelle;
18
19   public Gutschein(int G, int[] w) {
```

```
20      tabelle = new boolean[w.length][G + 1];
21    }
22
23    /**
24     * Helfer-Methode, die größere Zahl zurückgibt.
25     *
26     * @param a Die Zahl a.
27     * @param b Die Zahl b
28     *
29     * @return Die größere Zahl.
30     */
31    static int max(int a, int b) {
32      return (a > b) ? a : b;
33    }
34
35    static int gutscheinDP(int gutscheinBetrag, int warenWerten[], int
      ↪  werte[]) {
36      int w, g;
37      int n = warenWerten.length;
38      int tabelle[][] = new int[n + 1][gutscheinBetrag + 1];
39
40      for (w = 0; w <= n; w++) {
41        for (g = 0; g <= gutscheinBetrag; g++) {
42          if (w == 0 || g == 0)
43            tabelle[w][g] = 0;
44          else if (warenWerten[w - 1] <= g)
45            tabelle[w][g] = max(werte[w - 1] + tabelle[w - 1][g -
              ↪  warenWerten[w - 1]], tabelle[w - 1][g]);
46          else
47            tabelle[w][g] = tabelle[w - 1][g];
48        }
49      }
50
51      Konsole.zeige2DIntFeld(tabelle);
52
53      return tabelle[n][gutscheinBetrag];
54    }
55
56
57    public static void main(String[] args) {
58      //new Gutschein(50, new int[] { 10, 30, 40, 20, 15 });
59
60      System.out.println(gutscheinDP(51, new int[] { 10, 30, 40, 20, 15
        ↪  }, new int[] { 1, 1, 1, 1, 1 }));
61    }
62
63  }
```

github: raw