

Einzelprüfung „Theoretische Informatik / Algorithmen (vertieft)“

Einzelprüfungsnummer 66115 / 2020 / Herbst

## Thema 2 / Teilaufgabe 2 / Aufgabe 4

(Gutschein)

**Stichwörter:** Dynamische Programmierung

Das GUTSCHEIN-Problem ist gegeben durch eine Folge  $w_1, \dots, w_n$  von Warenwerten (wobei  $w \in \mathbb{N}_0$  für  $i = 1, \dots, n$ ) 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  $I \subseteq \{1, \dots, n\}$  gibt, sodass  $\sum_{i \in I} w_i = G$

### Exkurs: Teilsummenproblem

Das **Teilsummenproblem** (SUBSET SUM oder SSP) ist ein spezielles Rucksackproblem. Gegeben sei eine Menge von ganzen Zahlen  $I = \{w_1, w_2, \dots, w_n\}$ . Gesucht ist eine Untermenge, deren Elementsumme maximal, aber nicht größer als eine gegebene obere Schranke  $c$  ist.

(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.

Lösungsvorschlag

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

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

Lösungsvorschlag

51

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

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

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?

**Algorithmus 1: Gutschein-Problem**

```

table = boolean array  $n + 1 \times (G + 1)$  ; // Initialisiere ein boolsches Feld mit  $n + 1$ 
Zeilen für jeden Warenwert und 0 für keinen Warenwert und mit  $G + 1$  Spalten für alle
Gutscheinbetrag bis  $G$  und 0 für keinen Gutscheinbetrag

for  $k$  in  $1 \dots G$  do ; // Wenn der Gutscheinbetrag größer als 0 ist und es keine Warenwerte
gibt, d.h.  $n = 0$ , kann der Gutschein nicht eingelöst werden.

    | table[0][ $k$ ] = false
end

for  $i$  in  $0 \dots n$  do ; // Ist der Gutscheinbetrag 0, dann kann er immer eingelöst werden.

    | table[i][0] = true
end
for  $i$  in  $1 \dots n$  do ; // Durchlaufe jede Zeile der Warenwerte

    for  $k$  in  $1 \dots G$  do ; // Durchlaufe jede Spalte der Gutscheinbeträge in dieser Zeile

        table[i][ $k$ ] = table[i - 1][ $k$ ] ; // Übernehme erstmals das Ergebnis der Zelle der
        vorhergehenden Zeile in der gleichen Spalte
        if  $k \geq w_i$  und table[i][ $k$ ] noch nicht markiert then ; // Wenn der aktuelle
        Gutscheinbetrag größer als der aktuelle Warenwert und die aktuelle Zelle noch nicht
        als wahr markiert ist

            table[i][ $k$ ] = table[i - 1][ $k - w_i$ ] ; // übernimmt das Ergebnis des aktuellen
            Gutscheinbetrags minus des aktuellen Warenwerts
        ;
    end
end
end

```

```

/**
 * Nach <a href="https://www.geeksforgeeks.org/subset-sub-problem-dp-25"> Subset
 * Sum Problem auf geeksforgeeks.org</a>
 */
public class Gutschein {
    /**
     * @param G Die Indizes der GUTSCHEIN-Beträge.
     *
     * @param W Das GUTSCHEIN-Problem ist gegeben durch eine Folge  $w_1, \dots, w_n$  von
     * Warenwerten.
     *
     * @return Wahr, wenn der Gutscheinbetrag vollständig in Warenwerten eingelöst
     * werden kann, falsch wenn der Betrag nicht vollständig eingelöst
     * werden kann.
     */
}

```

```
public static boolean gutscheinDP(int G, int W[]) {
    // Der Eintrag in der Tabelle table[i][k] ist wahr,
    // wenn es eine Teilsumme der
    // W[0..i-1] gibt, die gleich k ist.
    int n = W.length;
    boolean table[][] = new boolean[n + 1][G + 1];

    // Wenn der Gutschein-Betrag größer als 0 ist und es keine
    // Warenwerte (n = 0) gibt, kann der Gutschein nicht eingelöst
    // werden.
    for (int k = 1; k <= G; k++) {
        table[0][k] = false;
    }

    // Ist der Gutscheinbetrag 0, dann kann er immer eingelöst werden.
    for (int i = 0; i <= n; i++) {
        table[i][0] = true;
    }

    for (int i = 1; i <= n; i++) {
        for (int k = 1; k <= G; k++) {
            table[i][k] = table[i - 1][k];
            // Warenwert
            int w = W[i - 1];
            if (k >= w && !table[i][k]) {
                table[i][k] = table[i - 1][k - w];
            }
        }
    }
    return table[n][G];
}

public static void main(String[] args) {
    System.out.println(gutscheinDP(50, new int[] { 10, 30, 40, 20, 15 }));
    System.out.println(gutscheinDP(41, new int[] { 10, 30, 40, 20, 15 }));

    System.out.println(gutscheinDP(3, new int[] { 1, 2, 3 }));
    System.out.println(gutscheinDP(5, new int[] { 1, 2, 3 }));
    System.out.println(gutscheinDP(6, new int[] { 1, 2, 3 }));
    System.out.println(gutscheinDP(2, new int[] { 1, 2, 3 }));
    System.out.println(gutscheinDP(1, new int[] { 1, 2, 3 }));
    System.out.println(gutscheinDP(7, new int[] { 1, 2, 3 }));
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/Gutschein.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/Gutschein.java)

Die äußere for-Schleife läuft  $n$  mal und die innere for-Schleife  $G$  mal.

Der letzte Eintrag in der Tabelle, also der Wert in der Zelle `table[W.length][G]`, löst das Gutscheinproblem. Steht hier `true`, dann gibt es eine Teilfolge der Waren, die den Gutscheinbetrag genau ausnutzt.

### Additum: Test-Klasse

```
import static org.junit.Assert.*;
import org.junit.Test;

public class GutscheinTest {

    int[] warenWerte = new int[] { 10, 30, 40, 20, 15 };

    private void assertEingelöst(int gutscheinBetrag, int[] warenWerte) {
        assertEquals(Gutschein.gutscheinDP(gutscheinBetrag, warenWerte), true);
    }

    private void assertNichtEingelöst(int gutscheinBetrag, int[] warenWerte) {
        assertEquals(Gutschein.gutscheinDP(gutscheinBetrag, warenWerte), false);
    }

    @Test
    public void eingelöst() {
        assertEingelöst(0, warenWerte);
        assertEingelöst(10, warenWerte);
        assertEingelöst(100, warenWerte);
        assertEingelöst(115, warenWerte);
        assertEingelöst(15, warenWerte);
        assertEingelöst(20, warenWerte);
        assertEingelöst(30, warenWerte);
        assertEingelöst(40, warenWerte);
        assertEingelöst(60, warenWerte);
        assertEingelöst(70, warenWerte);
    }

    @Test
    public void nichtEingelöst() {
        assertNichtEingelöst(11, warenWerte);
        assertNichtEingelöst(31, warenWerte);
        assertNichtEingelöst(41, warenWerte);
        assertNichtEingelöst(21, warenWerte);
        assertNichtEingelöst(16, warenWerte);
        assertNichtEingelöst(999, warenWerte);
    }
}
```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/GutscheinTest.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/GutscheinTest.java)

```
- gutscheinDP(3, new int[] { 1, 2, 3 }));: wahr (w)

[
  [w, f, f, f],
  [w, w, f, f],
  [w, w, w, w],
  [w, w, w, w]
]
```

```

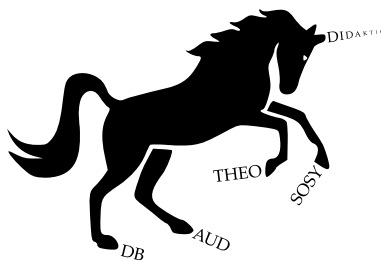
- gutscheinDP(7, new int[] { 1, 2, 3 });: falsch (f)

[ 0 1 2 3 4 5 6 7 G
  0 [w, f, f, f, f, f, f, f],
  1 [w, w, f, f, f, f, f, f],
  2 [w, w, w, w, f, f, f, f],
  3 [w, w, w, w, w, w, w, f]
  W
]

- gutscheinDP(10, new int[] { 10, 30, 40, 20, 15 });: wahr (w)

[ 0 1 2 3 4 5 6 7 8 9 10 G
  0 [w, f, f, f, f, f, f, f, f, f, f],
  10 [w, f, f, f, f, f, f, f, f, f, w],
  30 [w, f, f, f, f, f, f, f, f, f, w],
  40 [w, f, f, f, f, f, f, f, f, f, w],
  20 [w, f, f, f, f, f, f, f, f, f, w],
  15 [w, f, f, f, f, f, f, f, f, f, w]
  W
]

```



## Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.



Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an [hermine.bschlangaul@gmx.net](mailto:hermine.bschlangaul@gmx.net). Der  $\text{\LaTeX}$ -Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden: <https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2020/09/Thema-2/Teilaufgabe-2/Aufgabe-4.tex>