3. Übung zur Vorlesung
"Algorithmische Massenspektrometrie"

Wintersemester 2014/2015

Kai Dührkop

Ausgabe: 21. November 2014, Abgabe: 24. November 2014 in der Übung

1. **Coin-Changing-Problem:**

   Gegeben seien k Münzwerte $a_1, a_2, ..., a_k \in \mathbb{N}$, so dass $a_1 > a_2 > ... > a_{k-1}, a_k = 1$ und ein Betrag $M \in \mathbb{N}$. Betrachten Sie das folgende *Coin-Changing-Problem*:

   > Finden Sie ein Kompomer $C = c_1, c_2, ..., c_k$ über dem Alphabet der Münzwerte, für dass gilt $\sum_{i=1}^{k} c_i \cdot a_i = M$, wobei $c_1, c_2, ..., c_k \in \mathbb{N}$.

   Angenommen, wir haben den folgenden Algorithmus `UP_BOTTOM_CHANGE` zur Lösung dieses Problems gefunden:

```
UP_BOTTOM_CHANGE(M, a, k)
  r ← M
  FOR i ← 1 TO k
    c_i ← r / a_i
    r ← r - c_i · a_i
  END FOR;
  IF (∑_{i=1}^{k} c_i · a_i = M) THEN
    RETURN (c_1, c_2, ...c_k);
  END IF;
END.
```

   Wir sagen, dass ein Algorithmus zur Lösung des *Coin-Changing-Problem*s optimal ist, wenn er für jede Eingabeinstanz die Lösung, falls eine existiert, mit der kleinsten möglichen Anzahl von Münzen liefert.

   Beweisen Sie, dass der Algorithmus `UP_BOTTOM_CHANGE` nicht optimal ist. Wie nennt man den Ansatz, den man in dem Algorithmus verwendet? Überlegen Sie sich Bedingungen die Münzwerte a_i erfüllen müssen, damit der Algorithmus optimal ist.

   (4 Punkte)

2. **BONUS: Anzahl Strings:** Finden Sie einen effizienten Algorithmus, der berechnet, wie viele verschiedene **Strings** mit einer gegebenen Masse M über dem Alphabet $\Sigma = \{a, b, c, d\}$ mit den Massen $\mu(a) = 3, \mu(b) = 6, \mu(c) = 8, \mu(d) = 9$ existieren. Wie ist die Laufzeit dieses Algorithmus? Wie viel Speicherplatz braucht er? (Wie immer, jeweils in O-Notation.) Berechnen Sie, wie viele verschiedene Strings mit Masse 25 es gibt. Hinweis: Verwenden Sie eindimensionales DP.

   (8 Bonuspunkte)