35. Wechselgeld, 10 Punkte

Idee: liste sortieren, mit divmod(Betrag, Geldwert) Anzahl der Münzen und Restbetrag bestimmen, den Geldwert Anzahl-mal auf einen Stack speichern. Sollte die Anzahl o sein, dann divmod(Betrag, nächster Geldwert), wenn diese Anzahl ebenfalls o ist, dann hole eine Münze vom Stack, addiere Wert auf Betrag und berechne divmod(Betrag, Geldwert).
Eingabe: Betrag, Geldwerte(Liste, unsortiert)

1. sortiere Geldwerte absteigend
2. `Restbetrag = Betrag`
3. für jeden Geldwert `g` aus Liste
   (a) berechne `Anzahl, Restbetrag = divmod(Restbetrag, g)`
   (b) wenn `Anzahl == 0` und der Stack ist nicht leer und `a==0` aus `a,r = divmod(Restbetrag, g.next())`, dann `Restbetrag +` letzter Stackwert
   (c) berechne `Anzahl, Restbetrag = divmod(Restbetrag, g)`
   (d) lege Geldwert `g` `Anzahl`-mal auf Stack
4. Wenn `Restbetrag = 0`, dann gebe alle Elemente vom Stack zurück, sonst gebe `"Nicht lösbar."` zurück

Laufzeit

$\mathcal{O}(c * n)$ wobei n die Anzahl der Geldwerte ist. Teuerste Operation ist das Speichern auf dem Stack.

Speicherbedarf

$\mathcal{O}(c * n)$ wobei n die Anzahl der Geldwerte ist und c die Anzahl der verwendeten Münzen.

38. Beste unabhängige Knotenmenge in Bäumen, 10 Punkte

Teilprobleme

Ein Knoten ist Teil der besten unabhängigen Knotenmenge (folgend BUK), wenn sein Wert die Gesamtmenge erhöht. Ist ein Knoten Teil der BUK, dann können nur noch die Enkelkinder zur BUK gehören. Wenn er kein Teil der BUK wird, dann müssen seine Kinder Teil der BUK sein. Ziel ist es für jeden Knoten (von der Wurzel aus startend) die maximale BUK zu finden.

Rekursion

$$\text{BUK}(K) = \max\{\sum \text{BUK(Kinder von } K), \sum \text{BUK(Enkelkinder von } K)\}$$

Die Funktion muss abbrechen sobald man an den Blättern (Wert vom Blatt zurückgeben) angekommen ist.

Laufzeit

$\mathcal{O}(2^n)$, wobei n die Anzahl der Knoten ist. Man rechnet mit dieser Rekursion mehrfach alle Teilergebnisse aus. Besser wäre es, diese Teilergebnisse in eine Tabelle zu Speichern und die Rekursion dahingehend zu erweitern, dass bereits berechnete Teilergebnisse berücksichtigt werden.

Speicher

Diese naive Lösung verwendet keine Speicherstruktur. Daher $\mathcal{O}(n)$.