Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement Informatik
Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger

17. November 2016

Datenstrukturen & Algorithmen Lösungen zu Blatt 8 HS 16

Lösung 8.1 *Marsmission.*

Definition der DP-Tabelle: Wir benutzen eine $(m \times n)$-Tabelle T, und $T[i,j]$ enthält den maximal erreichbaren Wert der Gesteinsproben auf einem Süd-Ost-Weg von $(1,1)$ zu (i,j).

Berechnung eines Eintrags: Für (i,j) mit $1 < i,j \leq n$ beobachten wir folgendes: Endet ein Weg in der Position (i,j), dann sind wir entweder von oben oder von links gekommen. Also ist der maximal erreichbare Wert genau

$$T[i,j] = A[i,j] + \max\{T[i-1,j], T[i,j-1]\}, \tag{1}$$

denn zum Wert $A[i,j]$ der Gesteinsprobe auf der Position (i,j) muss noch der maximale Wert der Gesteinsproben auf den vorherigen Positionen addiert werden. Für die Fälle am oberen und am linken Rand legen wir folgendes fest:

- $T[1,1] = A[1,1] = 0$, denn $(1,1)$ ist die Startposition,
- $T[i,1] = A[i,1] + T[i-1,1]$ für $i > 1$, denn die Position $(i,1)$ kann nur von der Position oben, $(i-1,1)$, erreicht werden,
- $T[1,j] = A[1,j] + T[1,j-1]$ für $j > 1$, denn die Position $(1,j)$ kann nur von der Position links, $(1,j-1)$, erreicht werden.

Berechnungsreihenfolge: Der Eintrag $T[i,j]$ hängt nur von Einträgen für kleinere Werte von i und j ab. Daher können die Einträge zum Beispiel nach aufsteigenden Werten von $i = 1,\ldots,m$ und für gleiche i jeweils nach aufsteigenden Werten von $j = 1,\ldots,n$ berechnet werden.

Auslesen der Lösung: Der maximal erreichbare *Wert* der Gesteinsproben ist am Ende im Eintrag $T[m,n]$ gespeichert. Zur Rekonstruktion des Weges starten wir im Eintrag $T[m,n]$ und geben (m,n) aus. Danach wird geprüft, ob $T[m,n] = A[m,n] + T[m-1,n]$ ist. Dann sind wir von oben, also vom Eintrag $(m-1,n)$, gekommen und fahren entsprechend dort fort. Ansonsten ist $T[m,n] = A[m,n] + T[m,n-1]$ und wir sind von links gekommen, also fahren wir beim Eintrag $(m,n-1)$ fort. Die Rekonstruktion wird so fortgesetzt, bis die Startposition $(1,1)$ erreicht wird. Wird vorzeitig der Rand der Tabelle (links oder oben) erreicht, dann wird die Rekonstruktion beim einzig möglichen Vorgängerfeld (oben oder links) fortgesetzt. Am Ende haben wir die Positionen des Weges in umgekehrter Reihenfolge ermittelt.

Laufzeit: Die Tabelle besitzt Grösse $m \cdot n$, und jeder Eintrag kann in Zeit $\Theta(1)$ berechnet werden. Damit kann die Berechnung des maximalen Werts in Zeit $\Theta(mn)$ erfolgen.

Der rekonstruierte Weg besitzt Länge $m+n-1$. Da für jede Position in Zeit $\Theta(1)$ entschieden werden kann, ob wir von oben oder von links gekommen sind, kann der gesamte Weg in Zeit $\Theta(m+n)$ ermittelt werden. Zusammen mit der Zeit zum Ausfüllen der Tabelle ergibt sich also eine Gesamtlaufzeit von $\Theta(mn)$.

Lösung 8.2 *Wanderung.*

Der Einfachheit halber definieren wir zunächst $G := \sum_{i=1}^{n} g_i$.

Definition der DP-Tabelle: Wir verwenden eine $(n+1) \times (\lfloor G/2 \rfloor + 1)$-Tabelle T mit Einträgen, die entweder "true" oder "false" sind. Für $0 \leq i \leq n$ und $0 \leq g \leq \lfloor G/2 \rfloor$ sei genau dann $T[i, g] =$ true, wenn es eine Auswahl $K \subseteq \{1, \ldots, i\}$ der ersten i Gegenstände gibt, deren Gesamtgewicht exakt g beträgt, d.h. die $\sum_{k \in K} g_k = g$ erfüllt.

Berechnung eines Eintrags: Wir unterscheiden drei Fälle.

- $T[i, 0] =$ true für jedes $i \in \{0, \ldots, n\}$, denn die Menge $\{1, \ldots, i\}$ enthält die leere Menge $K = \emptyset$ mit Gewicht 0.
- $T[0, g] =$ false für jedes $g \in \{1, \ldots, \lfloor G/2 \rfloor\}$, denn mit der leeren Menge kann kein Gewicht $g > 0$ erzielt werden.
- Für alle $i \in \{1, \ldots, n\}$ und $g \in \{1, \ldots, \lfloor G/2 \rfloor\}$ setzen wir

$$T[i, g] = \begin{cases} T[i-1, g] & \text{falls } g_i > g \\ T[i-1, g] \vee T[i-1, g-g_i] & \text{sonst.} \end{cases}$$

  Für $g_i > g$ wiegt der Gegenstand i mehr als das zulässige Gesamtgewicht der Auswahl, folglich kann er niemals in dieser enthalten sein und es gilt $T[i, g] = T[i-1, g]$. Ansonsten kann der Gegenstand i in der Auswahl entweder enthalten sein, oder nicht, d.h., mit den Gegenständen $\{1, \ldots, i-1\}$ muss entweder das Gewicht $g - g_i$ erzielt werden können (falls i benutzt wird), oder das Gewicht g (falls i nicht benutzt wird).

Berechnungsreihenfolge: Wir berechnen die Einträge $T[i, g]$ für aufsteigende Werte von i, und für gleiche Werte von i für aufsteigende Werte von g.

Auslesen der Lösung: Für jedes $g \in \{0, \ldots, \lfloor G/2 \rfloor\}$ ist genau dann $T[n, g] =$ true, wenn eine Teilmenge von Gegenständen aus $\{1, \ldots, n\}$ mit Gesamtgewicht g existiert. Wir bestimmen zunächst den grössten Wert g_{max} mit $T[n, g_{max}] =$ true. Dies legt das maximal mögliche *Gewicht* eines Rucksacks mit Gewicht $\leq \lfloor G/2 \rfloor$ fest.

Nun setzen wir $i \leftarrow n$ und $g \leftarrow g_{max}$ und verfahren wie folgt. Ist $i = 0$, dann sind wir fertig. Ansonsten prüfen wir, ob $T[i, g] = T[i-1, g]$ gilt. Falls ja, dann setzen wir $i \leftarrow i-1$ und fahren wie beschrieben fort. Ansonsten gilt $T[i, g] = T[i-1, g-g_i]$, wir geben i aus, setzen $i \leftarrow i-1$ und $g \leftarrow g - g_i$ und fahren wie beschrieben fort. Die auf diese Art ausgegebenen Gegenstände werden in den Rucksack von Alice gelegt, die anderen in den Rucksack von Bob (oder umgekehrt).

Laufzeit: Die DP-Tabelle hat $\Theta(nG)$ viele Einträge, und jeder Eintrag kann in konstanter Zeit aus früher berechneten Einträgen berechnet werden. Damit kann die Tabelle in Zeit $\Theta(nG)$ ausgefüllt werden. Zur Rekonstruktion der Lösung werden weitere n Schritte benötigt. Jeder Schritt ist in konstanter Zeit ausführbar. Damit hat der gesamte Algorithmus Laufzeit $\Theta(nG)$, was lediglich pseudopolynomiell ist.

Lösung 8.3 *Zahlenrätsel.*

a) *Definition der DP-Tabelle*: Wir verwenden eine Tabelle S der Grösse $(n+1) \times (\sigma+1)$. Der Eintrag an der Stelle $S[n', \sigma']$ ist 'true', falls die Summe σ' aus den ersten n' Ziffern gebildet werden kann, und 'false' ansonsten.

   Berechnung eines Eintrags: Wir setzen $S[0, 0] \leftarrow$ 'true', und für alle $0 < \sigma' \leq \sigma$ setzen wir $S[0, \sigma'] \leftarrow$ 'false'. Um einen anderen Eintrag $S[n', \sigma']$ zu berechnen, iterieren wir über

alle $i = 1, 2, \ldots, n'$ und berechnen jeweils die Zahl z, die aus den letzten i Ziffern besteht. Falls $z > \sigma'$ ist, brechen wir ab, ansonsten prüfen wir den Wert $S[n' - i, \sigma' - z]$. Dieser beschreibt, ob die Zahl $\sigma' - z$ aus den ersten $n' - i$ Ziffern gebildet werden kann. Falls ja, können wir σ' aus den ersten n' Ziffern bilden, indem wir die letzten i Ziffern addieren. Wir setzen dann also $S[n', \sigma'] \leftarrow$ 'true' und brechen die Iteration ab. Ansonsten fahren wir mit der nächsten Wahl für i fort. Haben wir alle Möglichkeiten für i durchlaufen, dann setzen wir $S[n', \sigma'] \leftarrow$ 'false', da es keine Möglichkeit gibt, die Summe zu bilden.

Berechnungsreihenfolge: Jeder Eintrag $S[n', \sigma']$ hängt nur von Einträgen für kleinere Werte von n' und σ' ab. Daher können wir die Einträge zum Beispiel nach aufsteigenden Werten von n' und für gleiche n' jeweils nach aufsteigenden Werten von σ' berechnen.

Auslesen der Lösung: Die Lösung steht am Ende im Eintrag $S[n, \sigma]$.

Laufzeit: Die Anzahl an Einträgen in der Tabelle ist $\Theta(n\sigma)$. Für jeden Eintrag $S[n', \sigma']$ iterieren wir über die Werte $i = 1, 2, \ldots, n'$. Ausserdem müssen wir jeweils die Zahl berechnen, die aus den letzten i Ziffern gebildet wird. Wir können diese Zahl jeweils in konstanter Zeit aus der Zahl ableiten, die aus den letzten $i - 1$ Ziffern gebildet wurde (bzw. einfach ablesen, falls $i = 1$ ist). Alternativ können wir diese Zahlen auch für alle möglichen Kombinationen von n' und i vorberechnen.

Insgesamt erhalten wir eine Laufzeit von $\mathcal{O}(n^2\sigma)$. Wegen der Abhängigkeit von σ ist dies *nicht* polynomiell in n, sondern lediglich *pseudopolynomiell*. Die Laufzeit wird polynomiell, falls σ vorab durch ein Polynom in n beschränkt ist (also $\sigma \in \mathcal{O}(n^c)$ mit konstantem c).

b) Die Tabelle wird genauso gefüllt wie bisher. Zum Auffinden aller Möglichkeiten entwerfen wir einen rekursiven Algorithmus LIST, der drei Parameter n' (initial n), σ' (initial σ) sowie eine Zeichenkette *str* (initial leer) erhält. Wie zur Berechnung eines Eintrags iterieren wir über alle $i = 1, 2, \ldots, n'$ und berechnen jeweils die Zahl z, die aus den letzten i Ziffern besteht. Falls $S[n' - i, \sigma' - z] =$ 'false' ist, dann ist nichts weiter zu tun und wir fahren mit dem nächsten i fort. Ist dagegen $S[n' - i, \sigma' - z] =$ 'true', dann rufen wir LIST$(n' - i, \sigma' - z, \text{“}z + \text{”} \oplus str)$ rekursiv auf (wobei “$z +$ ” $\oplus$ *str* die Konkatenation der Zeichenketten “$z +$ ” und *str* bedeutet). Wichtig ist, dass wir jetzt anders als bei der Berechnung der Tabelle die Schleife nicht schon abbrechen, sobald ein i mit $S[n'-i, \sigma'-z] =$ 'true' gefunden wird, sondern alle Möglichkeiten anschauen, bis entweder $i = n'$ oder $z > \sigma'$ gilt. Die Rekursion terminiert, sobald $n' = 0$ gilt, und der Algorithmus gibt dann die Zeichenkette *str* (mit Ausnahme des letzten Zeichens, das immer “+” ist) aus.

Der Algorithmus wirkt auf den ersten Blick nicht sehr effizient, da anscheinend sehr viele rekursive Aufrufe stattfinden. Man kann aber beobachten, dass ein rekursiver Aufruf nur dann stattfindet, wenn er am Ende zu einer ausgegebenen Möglichkeit führt. Betrachten wir den Rekursionsbaum des Algorithmus, dann stellen wir fest, dass in jedem Blatt eine Lösung ausgegeben wird. Da in jedem Schritt n' um mindestens 1 verringert wird, ist die Länge jedes Weges zwischen zwei Blättern im Rekursionsbaum maximal $2n$. Da pro Knoten nur Zeit $\mathcal{O}(n)$ anfällt und in jedem Blatt eine Lösung ausgegeben wird, ist die Gesamtlaufzeit in $\mathcal{O}(n^2M)$, wobei M die Anzahl der ausgegebenen Möglichkeiten ist.

Man kann sogar erreichen, dass die Zeit zwischen der Ausgabe zweier Zeichenketten insgesamt nur in $\mathcal{O}(n)$ ist, wenn man zusätzlich bei der Berechnung jedes Tabelleneintrags das grösste i festhält, für das $S[n' - i, \sigma' - z] =$ 'true' ist. Da in jedem Schritt der Rekonstruktion n' umso stärker verringert wird, je grösser i ist (d.h., je mehr Aufwand beim Iterieren über i anfiel), reduziert sich die Laufzeit auf $\mathcal{O}(nM)$. Da jede ausgegebene Zeichenkette Länge $\Theta(n)$ hat (genau n Ziffern und bis zu $n - 1$ viele Pluszeichen) und es M Möglichkeiten gibt, ist die Laufzeit also linear in der Grösse der Ausgabe!