# Theoretische Informatik 1 / Algorithmentheorie

Wintersemester 2012/13

Prof. Dr. Georg Schnitger,
Dipl. Inf. Bert Besser,
Dipl. Inf. Matthias Poloczek

Arbeitsgruppe Theoretische Informatik, Institut für Informatik

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

---

# Blatt 6

Ausgabe: 29.11.2012
Abgabe: 06.12.2012 **vor** der Vorlesung

### 6.1. Aufgabe (1+2+3+1+4+3)

*Investieren geht über Studieren*

Ein Unternehmen hat $m$ Millionen Euro, die es in seine $n$ Abteilungen investieren will. Jede Abteilung $i$ macht $\ell_i$ Vorschläge für Projekte, in die Geld investiert werden kann und bei denen Gewinn zu erwarten ist. Die Verwirklichung des Vorschlags $V_{i,j}$ von Abteilung $i$ kostet $c_{i,j}$ Millionen Euro und bringt $r_{i,j}$ Millionen Euro ein; die Verwirklichung des Vorschlags ergibt also den Gewinn $r_{i,j} - c_{i,j}$. Jede Abteilung kann *höchstens* einen Vorschlag verwirklichen.

Alle Angaben für $m, n$ und $c_{i,j}$ sind natürliche Zahlen, insbesondere stellen die Abteilungen also Anträge über "glatte" Millionenbeträge. Die Erträge $r_{i,j}$ sind beliebig.

Ziel ist es, den Gewinn, also die Summe der Einzelgewinne $r_{i,j} - c_{i,j}$ zu maximieren. Dabei darf nur eine Gesamtsumme von höchstens $m$ Millionen Euro investiert werden.

Beispiel: 5 Millionen Euro stehen zur Verfügung, 3 Abteilungen machen jeweils $\ell_1 = 2$, $\ell_2 = 3$ beziehungsweise $\ell_3 = 1$ Vorschläge.

Abteilung 1

| Vorschlag | Kosten | Ertrag |
|---|---|---|
| $V_{1,1}$ | 1 | 6 |
| $V_{1,2}$ | 2 | 8 |

Abteilung 2

| Vorschlag | Kosten | Ertrag |
|---|---|---|
| $V_{2,1}$ | 1 | 9 |
| $V_{2,2}$ | 3 | 12 |
| $V_{2,3}$ | 4 | 16 |

Abteilung 3

| Vorschlag | Kosten | Ertrag |
|---|---|---|
| $V_{3,1}$ | 1 | 5 |

Hier bringt die Wahl der Vorschläge $V_{1,1}, V_{2,2}, V_{3,1}$ oder $V_{1,2}, V_{2,1}, V_{3,1}$ mit 18 Millionen Euro jeweils eine Gewinnmaximierung, wobei im ersten Fall die zur Verfügung stehenden 5 Millionen Euro komplett ausgegeben werden, während im zweiten Fall 4 Millionen ausgegeben werden und eine Million nicht investiert wird.

Wir wollen das Problem mit dynamischer Programmierung lösen. Betrachte hierzu die Teilprobleme

$$\text{Gewinn}(k, x) = \left\{ \begin{array}{l} \text{maximaler Gewinn bei Investition von } x \text{ Millionen Euro} \\ \text{unter Berücksichtigung der Vorschläge der Abteilungen 1 bis } k \end{array} \right.$$

für $1 \leq k \leq n$ und $0 \leq x \leq m$.

a.) **Gib** die Gewinnwerte aller Teilprobleme Gewinn$(k, x)$ für $1 \leq k \leq 3$ und $0 \leq x \leq 5$ im Beispiel an.

b.) Wie sieht die Initialisierung aus? Welche Teilprobleme können sofort gelöst werden?

c.) Wie können schwierige Teilprobleme mit Hilfe mit bereits gelöster Teilprobleme berechnet werden? **Gib** eine Rekursionsgleichung für die Teilprobleme an.

d.) Welches Teilproblem Gewinn$(k, x)$ liefert den maximalen Gewinn des Gesamtproblems?

e.) Rekonstruktion der optimalen Lösung: **Gib** einen Algorithmus mit dynamischer Programmierung für das Gesamtproblem an, der neben der Berechnung des maximalen Gewinns auch ausgibt, welche Vorschläge dafür gewählt werden.

f.) Wir nehmen nun an, dass ein nicht investierter Betrag $x$ auf die Bank gebracht werden kann und mit 5% verzinst wird. (Im Beispiel ist unter dieser Annahme die Lösung $V_{1,2}, V_{2,1}, V_{3,1}$ vorzuziehen, da sie 18,05 Millionen einbringt.) Wir möchten aber weiter einen Algorithmus verwenden, der das alte Problem löst. Wie ist *die Eingabe* für den Algorithmus zu erweitern, um zu berücksichtigen, dass ein nicht investierter Betrag Zinsen erbringt?

## 6.2. Aufgabe (6) *Maximale aufsteigende Teilfolge*

Wir betrachten das Problem der maximalen aufsteigenden Teilfolge. Gegeben ist eine Folge von Zahlen $a_1, a_2, \ldots, a_n$. Gesucht ist eine aufsteigende Teilfolge $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ maximaler Länge. Eine Teilfolge $(i_1 < i_2 < \cdots < i_k)$ ist aufsteigend, wenn $a_{i_1} < a_{i_2} < \cdots < a_{i_k}$ gilt.

**Gib** einen möglichst effizienten Algorithmus für dieses Problem an und **begründe** seine Korrektheit.

## 6.3. Aufgabe (2+2) *Visualisierung von Dynamischer Programmierung*

Algorithmen, die auf dynamischer Programmierung beruhen, lassen sich als gerichtete, azyklische Graphen darstellen, die wir als Visualisierungsgraphen bezeichnen. Die Knoten entsprechen dabei den zu lösenden Teilproblemen. Wir ziehen eine Kante $(a, b)$ ein, wenn die Berechnung des zu $b$ gehörenden Teilproblems auf die zu $a$ gehörende Lösung zurückgreift.

**Gib** für die beiden nachfolgend beschriebenen Visualisierungsgraphen jeweils die Menge der Kanten an.

a.) Beim Paarweisen Alignment Problem für zwei Strings $u, v$ mit $|u| = n$ und $|v| = m$ (vgl. S.71ff. im Skript) hat der Visualisierungsgraph die Knotenmenge

$$\{(i, j) \mid 0 \leq i \leq n, 0 \leq j \leq m\}.$$

(Zur Erinnerung: $D_{i,j}$ ist der Wert eines optimalen Alignments der Präfixe $u^{(i)}$ und $v^{(j)}$.)

b.) Beim Algorithmus von Floyd ist die Eingabe durch einen Graphen $G = (V, E)$ gegeben und wir wollen für alle Knotenpaare $u, v \in V$ die Länge eines kürzesten Weges bestimmen (vgl. S.68 im Skript).

Die Knoten des Visualisierungsgraphen für diesen Algorithmus sind durch die Tripel $(k, u, v)$ für $0 \leq k \leq |V|$ und $u, v \in V$ gegeben.

(Zur Erinnerung: distanz$_k[u][v]$ ist die Länge eines kürzesten Weges von $u$ nach $v$, der nur innere Knoten in $\{0, 1, \ldots, k\}$ besucht.)