Lösungsvorschläge zur 2. Übung

Aufgabe 1

In dieser Aufgabe soll eine vollständige Induktion des folgenden Typs durchgeführt werden:

1. *Induktionsverankerung(en):* Zeige, daß die Aussage für $n = n_0, n_1, \ldots$ gilt.
2. *Induktionsschritt:* Nimm an, daß die Aussage für alle $n \in \{1, \ldots, k-1\}$ gilt und zeige, daß sie auch für $n = k$ gilt.

zu a)

Da die b) schwieriger ist als die a), wird nur die b) gezeigt.

zu b)

Wir zeigen jetzt mit vollständiger Induktion für alle $n \geq 2$ zuerst die Aussage

$$T_2(n) \leq cn \log_2(n).$$

Wie man sieht kann man die obige Ungleichung leider nicht für $n = 1$ verankern, da dann die Laufzeit 0 wäre im Gegensatz zu $T_2(1) = 1$ laut Definition. Deswegen wählt man als *erste* Verankerung $n = 2$. Dann muß man sich überlegen, für welche n man durch die Berechnung von $\lfloor n/2 \rfloor$ auch Probleme der Größe 1 kommen kann. Man sieht leicht, daß das nur für $n = 3$ der Fall ist. Also verankert man die Aussage auch für $n = 3$.

Die Konstante c wird während der Induktion so bestimmt, so daß beide Verankerungen und der Induktionsschritt korrekt sind. Dabei ist es ganz **wichtig** darauf zu achten, daß c *unabhängig* von n ist!

Induktionsverankerung für $n = 2$

$$T_2(2) = 2T_2(1) + 2 = 4 \overset{!}{\leq} c2 \log_2(2) = 2c \tag{1}$$

Es muß also gelten: $c \geq 2$.

Induktionsverankerung für $n = 3$

$$T_2(3) = 2T_2(1) + 3 = 5 \overset{!}{\leq} c3 \log_2(3) \simeq 4.75c$$

Es muß also gelten $c \geq 1.05$

Induktionsschritt

Sei nun $n > 3$ beliebig aber fest. Es gilt:

$$\begin{aligned} T_2(n) &= 2T_2\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \\ &\overset{\text{Vor.}}{\leq} 2c \left\lfloor \frac{n}{2} \right\rfloor \log_2\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \end{aligned}$$

$$
\begin{aligned}
&\leq && cn\log_2\left(\frac{n}{2}\right)+n\\
&= && cn(\log_2(n)-1)+n\\
&= && cn\log_2(n)+n(1-c)\\
&\overset{*}{\leq} && cn\log_2(n)
\end{aligned}
$$

Die Ungleichung $*$ gilt für $c \geq 1$. Man kann also $c = 2$ wählen. Damit ist nun Aussage (1) gezeigt und T_2 ist Element von $O(n\log_2(n))$. Nun zur anderen Richtung. Es erweist als rechnerisch günstig für alle n die folgende Aussage zu zeigen:

$$T_2(n) \geq c(n+1)\log(n+1) \tag{2}$$

Induktionsverankerung für $n = 1$

$$T_2(1) = 1 \overset{!}{\geq} c(1+1)\log_2(1+1) = 2c$$

Also muß gelten: $c \leq \frac{1}{2}$.

Induktionsschritt

Sei nun $n > 2$ beliebig aber fest. Es gilt:

$$
\begin{aligned}
T_2(n) &= 2T_2\left(\left\lfloor\frac{n}{2}\right\rfloor\right)+n\\
&\overset{\text{Vor.}}{\geq} 2c\left(\left\lfloor\frac{n}{2}\right\rfloor+1\right)\log_2\left(\left\lfloor\frac{n}{2}\right\rfloor+1\right)+n\\
&\geq 2c\left(\frac{n-1}{2}+1\right)\log_2\left(\frac{n-1}{2}+1\right)+n\\
&= c(n+1)(\log_2(n+1)-1)+n\\
&= c(n+1)\log_2(n+1)\underbrace{-c(n+1)+n}_{>0 \text{ für } c=\frac{1}{2}}\\
&\geq c(n+1)\log_2(n+1)
\end{aligned}
$$

Man kann also $c = \frac{1}{2}$ wählen. Damit ist die Aussage (2) gezeigt. Es gilt dann natürlich auch:

$$T_2(n) = \frac{1}{2}(n+1)\log_2(n+1) \geq \frac{1}{2}n\log_2(n)$$

Also ist T_2 in $\Omega(n\log_2(n))$ und insgesamt auch in $\Theta(n\log_2(n))$.

Aufgabe 7

zu a)

Es sind die geordneten Paare $(3,2)$ $(5,2)$ $(9,2)$ $(6,2)$ $(9,6)$.

zu b)

Es gibt natürlich die meisten Inversionen, wenn man die Reihenfolge der Element genau invertiert: $\{n, n-1, \ldots, 1\}$. Dann bildet *jedes* Element der Menge mit *jedem anderen* eine Inversion. Das wären $n(n-1)$. Allerdings hat man jede Inversion doppelt gezählt, da wenn n_1 mit n_2 eine Inversion bildet, natürlich automatisch n_2 auch eine Inversion mit n_1 bildet. Also gilt:

$$\text{Maximale Anzahl der Inversionen} \quad = \quad \frac{n(n-1)}{2} \qquad (= O(n^2))$$

zu c)

siehe Vorlesung.

zu d)

Man wandelt den Mergesort–Algorithmus wie folgt ab.

1. *Devide:* Teile die Menge M in zwei möglichst gleichgroße Mengen M_1 und M_2.
2. *Conquer:* Ermittle die Anzahl der Inversionen $I(M_1)$ und $I(M_2)$ und sortiere M_1 und M_2 rekursiv. Abbruchbedingung für die Rekursion ist wie vorher jede einelementige Menge. Diese hat per definitionem keine Inversion.
3. *Merge:* Mische die beiden Mengen. Jedesmal wenn aus der rechten Menge ein Element r_i genommen wird, wird überprüft, wie viele Elemente N_i aus der linken Menge *noch nicht gemischt* worden sind. r_i wird also beim Einsortieren an N_i Elementen aus der linken Menge 'vorbeigeschoben' und macht damit N_i Inversionen rückgängig. Summiere alle diese N_i des Mischvorgangs und gebe als Anzahl der Inversionen den Wert
$$I(M) = I(M_1) + I(M_2) + \sum_{a_i \in M_2} N_i$$
zurück.

Aufgabe 8

Funktionsweise

FUNCTION Binom (n, m)
    BEGIN
    IF $(m = 0)$ or $(n = m)$ **THEN**
        RETURN 1
    ELSE
        RETURN $Binom(n-1, m) + Binom(n-1, m-1)$
    END

Für den korrekten Ablauf des Algorithmus müssen n und m ganzzahlig und größer oder gleich null sein. Außerdem muß $m \leq n$ gelten. Nur dann ist garantiert, daß eine der beiden Abbruchbedingungen erreicht wird.

Laufzeit

Man sieht leicht im Algorithmus, daß jeder Aufruf von *Binom* zu je zwei neuen Aufrufen von *Binom* führt. Die Anzahl der Berechnungen $T(n)$ kann man deshalb abschätzen durch:

$$T(n) \geq 2^{\text{minimale Anzahl der Aufrufebenen}}$$

Nimmt man $m = \lfloor n/2 \rfloor$, so müssen auf jeden Fall m Ebenen berechnet werden, bis n durch Subtraktion von 1 auf m (linker Summand), oder m durch Subtraktion 1 auf null (rechter Summand) reduziert worden ist, also eine der beiden Abbruchbedingungen zutrifft. Man erhält:

$$T_(n) \geq 2^{\frac{n}{2}}$$

Der Algorithmus hat also im schlechtesten Fall exponentielle Laufzeit.

Verbesserungen

Will man die rekursive Struktur beibehalten und hat man genügend Speicher zur Verfügung, so kann man schon berechnete Werte in einer $m \times n$–Matrix abspeichern. Man erhält somit eine polynominale Laufzeit. Noch schneller geht es iterativ durch Ausnutzung der Definition über die Faktultäten. Damit kann den Binominalkoeffizienten in linearer Zeit (in n) berechnen.