Prof. Andreas Goerdt
Frank Schädlich

TU Chemnitz
Sommersemester 2003

Theoretische Informatik III
1. Übung - Lösungsvorschläge

1. Aufgabe

(a) Die Behauptung wird durch vollständige Induktion bewiesen.

Induktionsanfang:
Für den binomialen Baum B_1 gilt offensichtlich die Behauptung (siehe Abbildung 1).

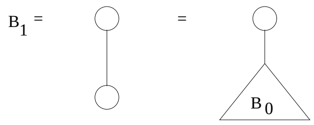

Abbildung 1: binomialer Baum B_1

Induktionsannahme:
Ein binomialer Baum B_k mit $k > 0$ hat eine Struktur wie in Abbildung 2.

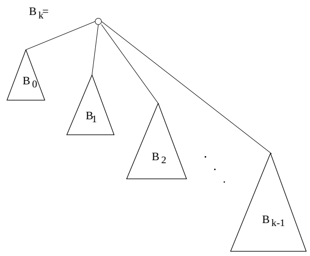

Abbildung 2: binomialer Baum B_k

Induktionsbeweis:
Der binomiale Baum B_{k+1} ist die Komposition zweier binomialer Bäume B_k (siehe Abbildung 4). Stellt man einen B_k (den linken in Abbildung 4) nach der Induktionsannahme (Abbildung 2) dar, ergibt sich sofort die Behauptung (siehe Abbildung 3).

(b) Die Behauptungen $Tiefe(B_i) = i$ und $\sharp\{$Anzahl der Knoten von B_i in Tiefe $i\} = 1$ werden durch vollständige Induktion bewiesen.

Induktionsanfang:
Der binomiale Baum B_0 besteht laut Definition aus einem Knoten. Somit ist seine Tiefe gleich 0 und die Anzahl der Knoten in Tiefe 0 beträgt 1.

Induktionsannahme:
Wir nehmen an, daß für ein $k \geq 0$ nun $Tiefe(B_k) = k$ und $\sharp\{$Anzahl der Knoten

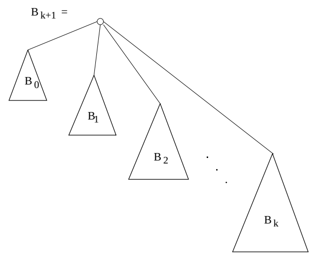

Abbildung 3: binomialer Baum B_{k+1}

von B_k in Tiefe $k\} = 1$ sei.

Induktionsbeweis:
Wir wollen zeigen, daß im binomialen Baum B_{k+1} die Aussagen $Tiefe(B_{k+1}) = k+1$ und $\sharp\{$Anzahl der Knoten von B_{k+1} in Tiefe $(k+1)\} = 1$ gelten.

Der binomiale Baum B_{k+1} ist die Komposition zweier binomialer Bäume B_k derart, daß zwei Teilbäume mit dem Tiefenunterschied 1 entstehen (siehe Abbildung 4):

$$\begin{aligned} Tiefe(B_{k+1}) &= \max\{Tiefe(B_k), Tiefe(B_k)+1\} \\ &= Tiefe(B_k)+1 \\ &= k+1 \text{ (nach Induktionsannahme).} \end{aligned}$$

Die Tiefe $(k+1)$ tritt nur im unteren binomialen Baum B_k auf und entspricht dort (intern) der Tiefe k. Nach der Induktionsannahme befindet sich dort genau ein Knoten, also gibt es in B_{k+1} genau einen Knoten in Tiefe $k+1$.

(c) Es ist $|B| = 2^i$ für alle $i \geq 0$ zu zeigen.
Der Beweis erfolgt durch vollständige Induktion.

Induktionsanfang:
Der binomiale Baum B_0 besteht laut Definition aus genau einem Knoten, es gilt also: $|B_0| = 1 = 2^0$.

Induktionsannahme:
Es gelte $|B_k| = 2^k$ für ein $k \geq 0$.

Induktionsbeweis:
Wir wollen $|B_{k+1}| = 2^{k+1}$ zeigen.

Laut Definition ist der binomiale Baum B_{k+1} Komposition zweier binomialer Bäume B_k (Abbildung 4).

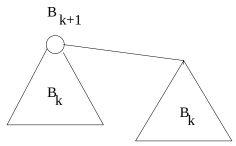

Abbildung 4: Komposition zweier binomialer Bäume

Es gilt also:

$$\begin{aligned}
|B_{k+1}| &= |B_k| + |B_k| \\
&= 2 \cdot |B_k| \\
&= 2 \cdot 2^k \text{ (nach Induktionsannahme)} \\
&= 2^{k+1}.
\end{aligned}$$

2. Aufgabe

(a) Wir verwenden die folgende Definition der Binomialkoeffizienten: $\binom{n}{k} := \frac{n!}{k!(n-k)!}$. Es gilt also:

$$\begin{aligned}
\binom{n-1}{k-1} + \binom{n-1}{k} &= \frac{(n-1)!}{(k-1)! \cdot (n-k)!} + \frac{(n-1)!}{k! \cdot (n-1-k)!} \\
&= \frac{(n-1)! \cdot k}{k! \cdot (n-k)!} + \frac{(n-1)! \cdot (n-k)}{k! \cdot (n-k)!} \\
&= \frac{(n-1)! \cdot n}{k! \cdot (n-k)!} \\
&= \frac{n!}{k! \cdot (n-k)!} \\
&= \binom{n}{k}.
\end{aligned}$$

Ein anderes Argument ist wie folgt. Der Binomialkoeffizient $\binom{n}{k}$ zählt die Anzahl k-elementiger Teilmengen einer n-elementigen Menge X. Diese Anzahl ist unabhängig von der speziellen Menge X, also sei o.B.d.A. $X = \{1, 2, \cdots, n\}$. Die Anzahl k-elementiger Teilmengen von $X = \{1, 2, \cdots, n\}$ ist aber gleich der Anzahl k-elementiger Teilmengen von $X \backslash \{n\} = \{1, \cdots, n-1\}$ (die Teilmengen, welche das Element n nicht enthalten) plus der Anzahl $(k-1)$-elementiger Teilmengen, von $X \backslash \{n\}$ (durch Hinzufügen von n entstehen alle k-elementigen Teilmengen von X, die n enthalten). Wir erhalten $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$.

(b) Nach dem Binomischen Satz gilt:

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} \cdot a^{n-i} \cdot b^i.$$

Aus $a = 1$ und $b = 1$ folgt unmittelbar die Behauptung.

Ein anderes Argument ist wie folgt. Der Binomialkoeffizient $\binom{n}{k}$ zählt die Anzahl k-elementiger Teilmengen einer n-elementigen Menge X. Mit 2^n zählen wir die Mengen aller $0,1$-Vektoren der Länge n. Jedem solchen $0,1$-Vektor entspricht genau eine Teilmenge von X, also $\sum\limits_{i=0}^{n} \binom{n}{i} = 2^n$.

(c) Wir betrachten zunächst die Spezialfälle $k, l, n = 0$. Danach führen wir einen Induktionsbeweis über $N \geq k + l + n$, d.h. die Induktion erfolgt über alle Tripel (k, l, n), deren Summe N nicht übersteigt.

Für $k = 0$ gilt:

$$\begin{aligned}
\sum_{i=0}^{l} \binom{l}{i} \cdot \binom{n}{k-i} &= \sum_{i=0}^{l} \binom{l}{i} \cdot \binom{n}{0-i} \\
&= \underbrace{\binom{l}{0} \cdot \binom{n}{0}}_{1} + \sum_{i=1}^{l} \binom{l}{i} \cdot \underbrace{\binom{n}{-i}}_{0} \\
&= 1 \\
&= \binom{l+n}{0} \\
&= \binom{l+n}{k}.
\end{aligned}$$

Für $l = 0$ gilt:

$$\begin{aligned}
\sum_{i=0}^{l} \binom{l}{i} \cdot \binom{n}{k-i} &= \sum_{i=0}^{0} \binom{0}{i} \cdot \binom{n}{k-i} \\
&= \binom{0}{0} \cdot \binom{n}{k} \\
&= \binom{n}{k} \\
&= \binom{l+n}{k}.
\end{aligned}$$

Für $n = 0$ gilt:

$$\sum_{i=0}^{l} \binom{l}{i} \cdot \binom{n}{k-i} = \sum_{i=0}^{l} \binom{l}{i} \cdot \underbrace{\binom{0}{k-i}}$$

Dieser Binomialkoeffizient ist nur gleich 1, wenn $k - i = 0$, also $k = i$, gilt. Ansonsten ist er gleich 0.

$$\begin{aligned}
&= \binom{l}{k} \\
&= \binom{l+n}{k}.
\end{aligned}$$

Induktionsanfang:
Für $N = 0$ gilt:

Das einzige Tripel (k,l,n), welches $k,l,n \geq 0$, $k+l+n \leq N$ und $k \leq l+n$ erfüllt, ist $(0,0,0)$. Es ist also $N = k = l = n = 0$. Wir erhalten damit (wie auch bereits in den obigen Spezialfällen gesehen):

$$\begin{aligned}
\sum_{i=0}^{l} \binom{l}{i} \cdot \binom{n}{k-i} &= \sum_{i=0}^{0} \binom{0}{i} \cdot \binom{0}{0-i} \\
&= \binom{0}{0} \cdot \binom{0}{0} \\
&= 1 \\
&= \binom{l+n}{k}.
\end{aligned}$$

Induktionsannahme:
Es existiert ein $N \in \mathbb{N}$, wofür gilt: Für alle Tripel (k,l,n) mit

- $k,l,n \in \mathbb{N}$,
- $k \leq l+n$ und
- $k+l+n \leq N$

gilt:

$$\binom{l+n}{k} = \sum_{i=0}^{l} \binom{l}{i} \cdot \binom{n}{k-i}.$$

Induktionsbeweis:
Wir wollen für $N' = N+1$ zeigen:
Für alle Tripel (k,l,n) mit

- $k,l,n \in \mathbb{N}$,
- $k \leq l+n$ und
- $k+l+n \leq N'$

gilt:

$$\binom{l+n}{k} = \sum_{i=0}^{l} \binom{l}{i} \cdot \binom{n}{k-i}.$$

Für $l+n=k$ ist $\binom{l+n}{k} = 1$ und

$$\begin{aligned}
\sum_{i=0}^{k-n} \underbrace{\binom{k-n}{i}}_{0 \text{ für } i > k-n} \cdot \underbrace{\binom{n}{k-i}}_{0 \text{ für } i < k-n} &= \binom{k-n}{k-n} \cdot \binom{n}{n} \\
&= 1.
\end{aligned}$$

Wir betrachten nun den Fall $k < l+n$ und $l,n,k > 0$ (anderenfalls können wir auf die eingangs betrachteten Spezialfälle zurückgreifen). Somit gilt auch $(n-1) \geq 0$

und $(k-1) \geq 0$. Es gilt dann:

$$\binom{l+n}{k} = \binom{l+(n-1)}{k} + \binom{l+(n-1)}{k-1} \qquad \text{(siehe (a))}$$

Es gilt $k < l+n$ und $k+l+n \leq N' = N+1$.

Damit ist $k+l+(n-1) \leq N$ und $(k-1)+l+(n-1) \leq N$.

In diesem Fall gilt auch $k \leq l+(n-1)$ bzw. $(k-1) \leq l+(n-1)$.

Wir wenden also zweimal die Induktionsannahme an:

$$= \sum_{i=0}^{l} \binom{l}{i} \cdot \binom{n-1}{k-i} + \sum_{i=0}^{l} \binom{l}{i} \cdot \binom{n-1}{k-1-i}$$

$$= \sum_{i=0}^{l} \binom{l}{i} \cdot \left[\binom{n-1}{k-i} + \binom{n-1}{k-i-1} \right]$$

erneutes Anwenden von (a)

$$= \sum_{i=0}^{l} \binom{l}{i} \cdot \binom{n}{k-i}.$$

„Kombinatorische Interpretationsmöglichkeit“: Zahlenlotto. Aus $l+n$ Zahlen werden k Stück ohne Zurücklegen und ohne Beachtung der Reihenfolge gezogen. Die Anzahl aller möglichen Zahlenkombinationen ist dann $\binom{l+n}{k}$. Nun gebe es in diesem Zahlenlotto l Gewinnzahlen, der Rest sind n Nieten. Ein Tip, der exakt i Gewinnzahlen und $n-i$ Nieten enthält, kann auf $\binom{l}{i} \cdot \binom{n}{n-i}$ verschiedene Arten gezogen werden. Summiert man anschließend über die mögliche Anzahl von Gewinnzahlen (also $0, \ldots, l$) in einem Tip auf, erhält man alle möglichen Tips und damit die gewünschte Identität: $\binom{l+n}{k} = \sum_{i=0}^{l} \binom{l}{i} \cdot \binom{n}{k-i}$.

3. Aufgabe

Wir bestimmen die explizite Form der Fibonacci-Zahlen, die zunächst implizit wie folgt gegeben sind:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_i &= F_{i-1} + F_{i-2} \quad (\text{ für } i \geq 2). \end{aligned}$$

Das Problem wird mittels der sogenannten „Potenzmethode“ gelöst, indem zunächst $F_i = t^i$ gesetzt wird:

$$\begin{aligned} t^i &= t^{i-1} + t^{i-2} \quad | : t^{i-2} \\ 0 &= t^2 - t - 1. \end{aligned}$$

Die Lösungen dieser quadratischen Gleichung sind:

$$t_1 = \frac{1+\sqrt{5}}{2}$$

$$t_2 = \frac{1-\sqrt{5}}{2}.$$

Mit den so errechneten Nullstellen wird die explizite Form der Rekursion wie folgt angesetzt:

$$F_i = k_1 \cdot \left(\frac{1+\sqrt{5}}{2}\right)^i + k_2 \cdot \left(\frac{1-\sqrt{5}}{2}\right)^i.$$

Die Konstanten k_1 und k_2 erhält man, indem die Startwerte der Rekursion ($F_0 = 1$ und $F_1 = 1$) eingesetzt werden und das resultierende Gleichungssystem gelöst wird:

$$k_1 = \frac{1}{\sqrt{5}} \qquad k_2 = -\frac{1}{\sqrt{5}}.$$

Mit der berechneten expliziten Form kann die Gültigkeit der Behauptung nun leicht nachgewiesen werden:

$$F_i = \frac{(1+\sqrt{5})^i - (1-\sqrt{5})^i}{2^i\sqrt{5}} \geq \left(\frac{1+\sqrt{5}}{2}\right)^{i-2}.$$

Denn es gilt:

$$\begin{aligned}
\frac{(1+\sqrt{5})^i - (1-\sqrt{5})^i}{2^i\sqrt{5}} &\geq \frac{(1+\sqrt{5})^{i-2}}{2^i} \\
\Longleftrightarrow \frac{(1+\sqrt{5})^{i-2}[(1+\sqrt{5})^2-1]}{2^i\sqrt{5}} &\geq \frac{(1-\sqrt{5})^i}{2^i} \\
\Longleftrightarrow \frac{(1+\sqrt{5})^{i-2}[5+2\sqrt{5}]}{2^i\sqrt{5}} &\geq \frac{(1-\sqrt{5})^i}{2^i}
\end{aligned}$$

Die linke Seite ist jedoch für $i \geq 2$ immer größer als 1, die rechte immer kleiner als 1, da $\sqrt{5} \approx 2,236$ ist.

Bemerkung: Eine andere Beweisidee, die für Fibonacci-Zahlen (fast) immer funktioniert, ist die vollständige Induktion. Hier wird zusätzlich berücksichtigt, daß $1+\varphi = \varphi^2$ mit $\varphi = \frac{1+\sqrt{5}}{2}$ ist.

Induktionsanfang:
Für $i = 2$ ist $F_2 = 1 = \varphi^0$ und $F_3 = 2 \geq \frac{1+\sqrt{5}}{2} = \varphi^1$.

Induktionsannahme:
Sei für ein $k \geq 3$ und alle i mit $2 \leq i \leq k$ schon bewiesen, daß $F_i \geq \varphi^{i-2}$ gilt.

Induktionsbeweis:
Wir wollen $F_{k+1} \geq \varphi^{k-1}$ zeigen.

Nach Definition der Fibonacci-Zahlen gilt $F_{k+1} = F_k + F_{k-1}$. Mit der Induktionsannahme folgt

$$\begin{aligned}
F_{k+1} &= F_k + F_{k-1} \\
&\geq \varphi^{k-2} + \varphi^{k-3} \qquad \text{(Induktionsannahme)} \\
&= \varphi^{k-3} \cdot (\varphi + 1) \\
&= \varphi^{k-3} \cdot \varphi^2 \\
&= \varphi^{k-1}.
\end{aligned}$$

4. Aufgabe

Sind $a, b\colon \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ Funktionen sowie $n \in \mathbb{R}^+$, so ist $a(n)$ der Art $O(b(n))$, genau dann, wenn ein $c \in \mathbb{R}^+ (c > 0)$ und ein $n_0 \in \mathbb{R}^+$ existieren, so daß $a(n) \leq c \cdot b(n)$ für alle $n \geq n_0$ gilt. $O(b(n))$ beschreibt somit eine Menge von Funktionen. Als Schreibweise hat sich auch $a(n) = O(b(n))$ für $a(n) \in O(b(n))$ eingebürgert. Diese Aussage muß von links nach rechts gelesen werden. Es gilt **nicht** $O(b(n)) = a(n)$.

(a) Sei $g(n)$ eine beliebige Funktion der Art $O(f(n))$. Es existieren also $c_1 > 0$ und $n_1 > 0$, so daß für alle $n \geq n_1$ gilt: $g(n) \leq c_1 \cdot f(n)$. Für jedes $c > 0$ ist dann $c \cdot g(n) \leq c \cdot c_1 \cdot f(n)$, also $c \cdot g(n) \leq c_2 \cdot f(n)$ mit $c_2 = c \cdot c_1$.
Es folgt $c \cdot g(n) \in O(f(n))$.

(b) Für alle $n \geq 0$ sowie für jedes $c \geq 1$ und jede Funktion $f\colon \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ gilt: $f(n) \leq c \cdot f(n)$, also $f(n) \in O(f(n))$.

(c) Aus $g(n) \in O(f(n))$ folgt $\exists c_g > 0, \exists n_g \forall n \geq n_g : g(n) \leq c_g \cdot f(n)$.
Aus $h(n) \in O(g(n))$ folgt $\exists c_h > 0, \exists n_h \forall n \geq n_h : h(n) \leq c_h \cdot g(n)$.

Sei $c = c_g \cdot c_h$ und $n_0 = \max\{n_g, n_h\}$, so gilt für alle $n \geq n_0$:

$$\begin{array}{rcll} h(n) & \leq & c_h \cdot g(n) & \\ & \leq & c_h \cdot c_g \cdot f(n) & (\text{da } g(n) \leq c_g \cdot f(n)) \\ & = & c \cdot f(n) & \end{array}$$

und somit ist $h(n) \in O(f(n))$.

(d) Es gilt für alle n: $f(n) \leq \max\{f(n), g(n)\}$ und $g(n) \leq \max\{f(n), g(n)\}$, also $f(n) + g(n) \leq 2 \cdot \max\{f(n), g(n)\}$.
Für eine beliebige Funktion $h(n) \in O(f(n) + g(n))$ gilt: $\exists c_0, \exists n_0 : h(n) \leq c_0 \cdot (f(n) + g(n))$ für alle $n \geq n_0$. Damit haben wir $h(n) \leq c_0 \cdot 2 \cdot \max\{f(n), g(n)\} = c \cdot \max\{(f(n), g(n))\}$ mit $c = 2 \cdot c_0$ für alle $n \geq n_0$. Es folgt $h(n) \in O(\max\{f(n), g(n)\}$.

(e) Nach der Annahme $h(n) \in O(f(n))$ und $l(n) \in O(g(n))$ haben wir:
$\exists c_h > 0, \exists n_h \forall n \geq n_h : h(n) \leq c_h \cdot f(n)$ und analog gilt
$\exists c_l > 0, \exists n_l \forall n \geq n_l : l(n) \leq c_l \cdot g(n)$.
Sei $c = c_h \cdot c_g$ und $n_0 = \max\{n_h, n_g\}$. Die beiden Ungleichungen gelten dann auch für alle $n \geq n_0$. Alle betrachteten Funktionen sind der Art $\mathbb{R}^+ \to \mathbb{R}^+$, also gilt (für alle $n \geq n_0$) $h(n) \cdot l(n) \leq c_h \cdot f(n) \cdot c_l \cdot g(n) = c \cdot f(n) \cdot g(n)$, folglich $h(n) \cdot l(n) \in O(f(n) \cdot g(n))$.

5. Aufgabe

Wir nehmen im Folgenden an, daß der Eingabegraph zusammenhängend ist, ansonsten kann man vorab mit Depth-First-Search den Graphen in der Zeit $O(|V| + |E|)$ auf Zusammenhang testen. Folgende Datenstrukturen kommen zum Einsatz:

T: Einfach verkettete Liste.

V_i: Union-Find-Strukturen mit Heuristik Union-By-Size.

E_i: Binomiale Heaps mit Kanten als Objekten, geordnet nach dnm Kantengewichten.

V[1..n]: Feld von n Adjazenzlisten, die den (zusammenhängenden und ungerichteten) Graphen $G = (V, E)$ mit der Punktmenge $V = \{1, \cdots, n\}$ beschreiben.

Programm:

```
1   T := NIL;
2   anzahl := 0;
3   for i := 1 to n do
4       anzahl := anzahl + 1;
5       V_i:=NewUnion();
6       Union(V_i, i);
7       E_i:=BinMakeHeap();
8       for each Kante({x, y}) ∈V[i] do
9           BinInsert(E_i, Kante({x,y}));
10  while anzahl > 0 do
11      Kante(u,v):=BinMin(E_1);
12      BinDelMin(E_1);
13      i := Find(u);
14      j := Find(v);
15      if i ≠ j then
16          ListAppend(T, Kante({u,v}));
17          Union(V_i, V_j);
18          BinMeld(E_i, E_j);
19          anzahl := anzahl − 1;
```

Zur Laufzeit:

1-2: $O(1)$
4-7: $O(1)$
9: $O(1)$
3-9: $O(|V| + |E|)$ (Es wird jede Kante eingetragen und jeder Knoten bearbeitet.)
11: $O(1)$
12: $O(\log |E|) = O(\log |V|^2) = O(2 \cdot \log |V|) = O(\log |V|)$, da $|E|$ immer der Art $O(|V|^2)$ ist.
13-14: $O(\log |V|)$
16,17: $O(1)$
18: $O(\log |E|) = O(\log |V|)$
19: $O(1)$
15-19: $O(\log |V|)$
11-19: $O(\log |V|)$
10-19: $O((|V| + |E|) \log |V|)$, da die Schleife maximal $(|V| + |E|)$-mal durchlaufen wird.

Die Gesamtlaufzeit beträgt also $O((|V| + |E|) \cdot \log |V|)$.