

Aufgabe 1 (Rekurrenzen)

[Σ 8] Punkte

a) • Schritt 1:

$$\begin{aligned}
 g_n &= \sum_{i=0}^{n-1} g_i = \sum_{i=0}^{n-2} g_i + g_{n-1} \\
 &= 2g_{n-1} \text{ für } n > 1. \\
 \Rightarrow g_n &= 2g_{n-1} + [n=0] - [n=1] \\
 &\text{Ausgleich für } n=1 \text{ nötig!}
 \end{aligned}$$

• Schritt 2:

$$\begin{aligned}
 G(z) &= \sum_n g_n z^n \\
 &= 2z \cdot \sum_n g_{n-1} z^{n-1} + 1 - z \\
 &= 2z \cdot G(z) + 1 - z
 \end{aligned}$$

• Schritt 3:

$$\begin{aligned}
 (1 - 2z) \cdot G(z) &= 1 - z \\
 \Leftrightarrow G(z) &= \frac{1 - z}{1 - 2z}
 \end{aligned}$$

• Schritt 4: siehe unten

Alternative Lösung mit Cauchy-Produkt:

• Schritt 1: $g_0 = 1$, aber $\sum_{i=0}^{-1} g_i = 0$.
 Also $g_n = \sum_{i=0}^{n-1} g_i + [n=0]$

• Schritt 2:

$$\begin{aligned}
 G(z) = \sum_n g_n z^n &= \sum_n \left(\sum_{i=0}^{n-1} g_i \right) z^n + 1 \quad (\rightarrow \text{Indexvertauschung}) \\
 &= z \cdot \sum_n \left(\sum_{j=0}^{n-1} g_{n-1-j} \right) z^{n-1} + 1 \quad (\rightarrow \text{Indexverschiebung}) \\
 &= z \cdot \sum_k \left(\sum_{j=0}^k g_{k-j} \right) z^k + 1 \\
 &= z \cdot \left(\sum_k g_k z^k \right) \cdot \left(\sum_k z^k \right) + 1 \quad (\rightarrow \text{Cauchy-Produkt}) \\
 &= z \cdot G(z) \cdot \frac{1}{1-z} + 1
 \end{aligned}$$

- Schritt 3:

$$\begin{aligned} G(z) &= z \cdot G(z) \cdot \left(\frac{1}{1-z}\right) + 1 \\ \Rightarrow -1 &= G(z) \cdot \left(\frac{z}{1-z} - 1\right) \\ \Rightarrow G(z) &= \frac{1-z}{1-2z} \end{aligned}$$

- Schritt 4 (für beide Varianten):

$$\begin{aligned} G(z) &= \frac{1}{1-2z} - \frac{z}{1-2z} \\ &= \sum_n 2^n z^n - z \cdot \sum_n 2^{n-1} z^n \\ \Rightarrow g_n &= 2^n - 2^{n-1} = 2^{n-1} \text{ für } n > 0, g_0 = 1. \end{aligned}$$

b)

$$\begin{aligned} a_n - a_{n-1} &= c_n \\ \Rightarrow a_n &= a_{n-1} + c_n \\ \Rightarrow a_n &= \sum_{k=0}^n c_k \\ &= \sum_{k=0}^n 2^k = 2^{n+1} - 1 \end{aligned}$$

Aufgabe 2 (Greedy-Algorithmen)

[Σ 10] Punkte

- a) Sortiere zunächst die Objekte der Größe nach und initialisiere die Behälter als leer. Wiederhole bis keine Objekte mehr vorhanden sind:

- Nimm das größte noch nicht eingefügte Objekt und packe es in den ersten nicht leeren Behälter, der noch genug Platz hat. Gibt es keinen solchen Behälter, dann nimm einen neuen hinzu.

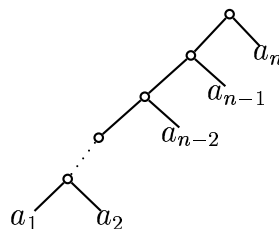
- b) Betrachte die folgende Instanz:

$$b = 10, a_1 = 5, a_2 = 4, a_3 = 4, a_4 = 3, a_5 = 2, a_6 = 2.$$

Greedy braucht 3 Behälter: $b_1 = 5 + 4, b_2 = 4 + 3 + 2, b_3 = 2$.

Optimal ist aber: $b_1 = 5 + 3 + 2, b_2 = 4 + 2 + 2 + 2$.

- c) Der Baum sieht so aus:



Die Codierung braucht für a_n ein Bit, für a_{n-1} zwei Bit, usw. Am Ende stehen a_1 und a_2 , die jeweils $n - 1$ Bit benötigen. Somit werden zur Codierung des Textes

$$\sum_{k=2}^n 2^k \cdot (n - k + 1) + 2(n - 1)$$

Bits benötigt.

Aufgabe 3 (Aufwandsanalyse)

[Σ 9] Punkte

- a) Ist der Exponent n ungerade so wird das Ergebnis mit x multipliziert und n um Eins verkeinert. Im nächsten Schritt ist n also gerade.

Ist n gerade, so wird n durch zwei geteilt und das Ergebnis quadriert.

Im günstigsten Fall ist n immer gerade, also eine Zweierpotenz $n = 2^k$, $k \in \mathbb{N}$.

Im schlimmsten Fall ist n ungerade und $\frac{n-1}{2}$ ungerade, usw. Oder von hinten nach vorn betrachtet ist $n = (\dots(((1 \cdot 2) + 1) \cdot 2) + 1) \cdots + 1$. Also $n = \sum_{i=0}^k 2^i = 2^{k+1} - 1$ für $k \in \mathbb{N}$.

- b) Die Rekurrenz ist:

$$T(0) = 0, \quad T(n) = T((n-1) \operatorname{div} 2) + 2$$

Hierbei wurde der Fall “ungerade” und “gerade” zusammengefaßt, da $n-1$ immer ungerade sein soll.

- c) Da im O -Kalkül abgeschätzt wird sei $n = 2^{k+1} - 1$.

Es werden $k+1$ Divisionen benötigt, bis der Basisfall erreicht wurde. Nach jeder Division wird eine Multiplikation durchgeführt und einmal quadriert. Das sind insgesamt $2k+2$ Multiplikationen. Da $n \approx 2^k$ ist, ist die Berechnung mit $O(\log n)$ Multiplikation durchzuführen.

Aufgabe 4 (Dynamische Programmierung)

[Σ 10] Punkte

- a) • Die Werte werden alle in einer Tabelle gespeichert. Zum Berechnen eines nicht-trivialen Falles werden die drei schon zuvor berechneten Vorgänger aus der Tabelle gelesen. Um die Zeile n voll zu bekommen sind maximal $6n$ Ablesevorgänge nötig und es gibt n Vorgängerzeilen. Also $O(n^2)$ Aufwand (leider auch beim Speicher).

- Eine Implementierung war alternativ auch möglich:

```
trinom n k = (alltris!!n)!!k
```

```
alltris = [1]:mktris [1]
```

```
mktris (x:xs) = erg:mktris erg
  where erg = mktri (0:0:x:xs)
```

```
mktri [x,y] = [x+y,y]
```

```
mktri (x:y:z:xs) = (x+y+z):mktri (y:z:xs)
```

b) Bemerkung: Es gilt nach Definition:

$$\begin{aligned} \text{trinom } n \ 0 &= \text{trinom } n \ (2*n) \text{ und} \\ \text{trinom } n \ 1 &= \text{trinom } n \ (2*n-1) \end{aligned}$$

Beweis der Aussage durch vollständige Induktion über n :

Induktionshypothese: Es gilt für ein n und alle $k=0..2*n$ $\text{trinom } n \ k = \text{trinom } n \ (2*n-k)$

Induktionsanfang: $n=0$ siehe Bemerkung oben. Induktionsschritt: $n \rightarrow n+1$:

Die Aussagen für $k=0, 1, 2*(n+1)-1, 2*(n+1)$ sind nach der Bemerkung oben erfüllt. Sei nun also k ungleich der vier Werte.

$$\begin{aligned} \text{trinom } (n+1) \ k &= \text{trinom } n \ (k-2) + \text{trinom } n \ (k-1) \\ &\quad + \text{trinom } n \ k \text{ nach Def} \\ &= \text{trinom } n \ (2*n-(k-2)) + \text{trinom } n \ (2*n-(k-1)) \\ &\quad + \text{trinom } n \ (2*n-k) \text{ drei mal I.H.} \\ &= \text{trinom } n \ (2*(n+1) -k) + \text{trinom } n \ (2*(n+1) -(k-1)) \\ &\quad + \text{trinom } n \ (2*(n+1)-(k-2)) \\ &= \text{trinom } (n+1) \ (2*(n+1)-k) \text{ nach Def.} \end{aligned}$$

Qed.

Aufgabe 5 (Logik)

[Σ 14] Punkte

- a) \mathcal{A} ist ein Modell für F , wenn \mathcal{A} eine zu F passende Struktur ist und $\mathcal{A}(F) = 1$ gilt. Eine Struktur heißt zu F passend, wenn jedem k -stelligem Prädikatsymbol bzw. Funktionssymbol ein k -stelliges Prädikat bzw. eine k -stellige Funktion und jeder freien Variable ein Element aus $U_{\mathcal{A}}$ zugeordnet wird.
- b) Zu zeigen ist, daß $H = F \rightarrow G$ eine Tautologie ist. Zeige also Unerfüllbarkeit von $\neg H = F \wedge \neg G$ mit Resolution:

$$\text{SkNF}(\neg H) = \forall x \forall y \left(\underbrace{(P(x) \vee Q(y))}_{F_1} \wedge \underbrace{\neg P(a)}_{F_2} \wedge \underbrace{\neg Q(a)}_{F_3} \right)$$

$$\text{Resolution: } R_1 = \{F_1, F_2\}[x/a] \quad R_2 = \{R_1, F_3\}[y/a] = \{\}$$

- c) $F = \exists x P(x)$ und $\text{SkNF}(F) = P(a)$. Dann ist $\mathcal{A} = (\mathbb{N}, I_{\mathcal{A}})$ mit $I_{\mathcal{A}}: P \mapsto \{x | x \text{ gerade}\}$, $a \mapsto 3$ ein Modell für F . $\text{SkNF}(F)$ hingegen wird unter der Struktur \mathcal{A} zu falsch ausgewertet.
- d)
- Basisfall:
Atomare Formeln enthalten keine Operatoren.
 - Schritt:
Seien F und G Formeln, die nur Implikationen enthalten.
Aus $\neg F$ wird $F \rightarrow 0$.
Aus $(F \vee G)$ wird $(\neg F \rightarrow G)$.
Aus $(F \wedge G)$ wird $\neg(F \rightarrow \neg G)$.