Universität Konstanz
Fachbereich Informatik & Informationswissenschaft
Prof. Dr. Sven Kosub / Michael Aichem, Julian Müller, Dagmar Sorg, Michael Strecke, Nadja Willenborg
Mathematik: Diskrete Strukturen
SS 2015

6. Übungsblatt

Ausgabe: 22.05.2015 **Abgabe:** 29.05.2015, bis spätestens 12:00 per Mail an den Tutor

Vertiefung: **10 Punkte**

(a) Welche Potenzreihe (in möglichst einfacher Darstellung mit nur einem Summensymbol) entspricht dem Produkt der Potenzreihen $\sum_{n=0}^{\infty} x^n$ und $\sum_{n=0}^{\infty}(2n+1)\cdot x^n$?

(b) Es sei $A(x)$ die erzeugende Funktion der Folge $a_0, a_1, \ldots$. Welche Folge besitzt dann die erzeugende Funktion $\int_0^x A(t)dt$?

(c) Welche erzeugende Funktion hat die Folge $(1, 1, 1, 1, 1, 1, 0, 0, \ldots, 0, \ldots)$?

(d) Welche erzeugende Funktion hat die Folge $(0, 0, 0, 0, 0, 0, 1, 1, \ldots, 1, \ldots)$?

(e) Welche erzeugende Funktion hat die Folge $(3, 2, 1, 3, 2, 1, 3, 2, 1, \ldots, 3, 2, 1, \ldots)$?

(f) Welche erzeugende Funktion hat die Folge $(-1, 1, -1, 1, -1, 1, \ldots, -1, 1, \ldots)$?

(g) Welche erzeugende Funktion hat die Folge $(0, 1, 3, 6, 10, 15, 21, 28, 36, \ldots, ?, \ldots)$?

(h) Ist $a_n =_{\mathrm{def}} (1+\sqrt{2})^n + (1-\sqrt{2})^n$ für alle $n \in \mathbb{N}$ stets eine natürliche Zahl?
Hinweis: Finden Sie eine geeignete rekursive Darstellung für a_n.

(i) Gegeben sei folgende lineare Rekursionsgleichung (aber ohne fester Ordnung!):

$$\begin{aligned} a_n &=_{\mathrm{def}} 1 + \sum_{k=0}^{n-1} a_k \qquad \text{für } n \geq 1 \\ a_0 &=_{\mathrm{def}} 1 \end{aligned}$$

Bestimmen Sie die Folgenglieder a_n mittels der erzeugenden Funktion.

(j) Bestimmen Sie die Folgenglieder zur Folge aus Teilaufgabe (i), indem Sie $a_n - a_{n-1}$ betrachten und eine einfacherere Rekursionsgleichung gewinnen.

Kreativität: **10 Punkte**

Betrachten Sie die allgemeine inhomogene Version der Fibonacci-Zahlen:

$$\begin{aligned} F_n &=_{\mathrm{def}} F_{n-1} + F_{n-2} + g(n) \qquad \text{für } n \geq 2 \\ F_1 &=_{\mathrm{def}} g(1) \\ F_0 &=_{\mathrm{def}} g(0) \end{aligned}$$

Hierbei ist $g : \mathbb{N} \to \mathbb{R}$ eine beliebige Funktion.

(a) Bestimmen Sie die erzeugende Funktion unter Verwendung von $G(x) =_{\text{def}} \sum_{n=0}^{\infty} g(n) \cdot x^n$.

(b) Lösen Sie die Rekursionsgleichung für die Funktion $g(n) =_{\text{def}} 1$.

Transfer: **10 Punkte**

In der Projekt-Dokumentation zu einem *Software Code Review* sind Sie auf die Pseudocode-Beschreibung eines implementierten Sortierverfahrens gestoßen, zusammen mit der Information, dass zum Sortieren von n Integer-Zahlen, die in einem Array A gespeichert sind, der Hauptaufruf mit SORTARRAY$(A, 0, n-1)$ erfolgen muss:

Algorithmus:	ARRAYSORT(A, i, j)
Eingabe:	Integer-Array A der Länge n, Indizes $i, j \in \{0, n-1\}$, $i \leq j$
Ausgabe:	–

```
1.   IF i = j                          /* Terminierungsfall, Array der Länge 1
2.      RETURN
3.   ELSE
4.      IF i + 1 = j                   /* Terminierungsfall, Array der Länge 2
5.         IF A[i] > A[j]                          /* Inhalte falsch angeordnet
6.            Vertausche Inhalte von A[i] und A[j]
7.         RETURN
8.      ELSE
9.         k ← ⌊(j − i + 1)/3⌋        /* Drittel der Länge berechnen, abrunden
10.         ARRAYSORT(A, i, j − k)              /* vordere zwei Drittel sortieren
11.         ARRAYSORT(A, i + k, j)              /* hintere zwei Drittel sortieren
12.         ARRAYSORT(A, i, j − k)    /* vordere zwei Drittel nochmal sortieren
13.   RETURN
```

Sie wollen sich von der Korrektheit und der Effizienz des Verfahrens überzeugen.

(a) Zeigen Sie, dass der Sortieralgorithmus mit Aufruf ARRAYSORT$(A, 0, n-1)$ ein Array A der Länge n aufsteigend sortiert.

Verwenden Sie dazu vollständige Induktion über die Länge der (Teil)Arrays, d.h., beweisen Sie, dass aus der Korrektheit des Algorithmus für alle (Teil)Arrays der Länge $m \leq n-1$ die Korrektheit des Algorithmus für alle (Teil)Arrays der Länge n folgt.

Hinweis: Überlegen Sie sich, dass nach dem zweiten rekursiven Aufruf von SORTARRAY die k größten Zahlen bereits an der richtigen Position im Array stehen.

(b) Die Laufzeit des Algorithmus hängt maßgeblich von der Anzahl der Vergleiche „$A[i] > A[j]$" in Zeile (5) ab (aber nicht von den im Array gespeicherten Integer-Zahlen!). Geben Sie eine Rekursionsgleichung für die Anzahl dieser Vergleiche V_n auf einem Array der Länge n.

Wie schnell läuft der Algorithmus in Abhängigkeit von n?