Übungsblatt 1

1.) Überlegen Sie sich einen Pseudocode für die folgenden Algorithmen und bestimmen Sie die Anzahl der notwendigen Schritte, die (in Ihrem Pseudocode) nötig sind, um eine n-elementige Menge zu sortieren. Wenden Sie die Algorithmen auf den Datensatz 6, 77, 45, 103, 4, 17 an.

   (a) Bubble-Sort: Der Algorithmus vergleicht der Reihe nach je zwei benachbarte Zahlen und vertauscht diese, falls sie nicht in der richtigen Reihenfolge angeordnet sind. Dieses Verfahren wird so lange wiederholt, bis alle Zahlen der Eingabe sortiert sind.

   (b) Selection-Sort: Der Algorithmus sucht zunächst das kleinste Element und bringt es an die erste Position. Anschließend sucht er das zweitkleinste Element und bringt es an die zweite Position, usw.

2.) Sequentielle Suche: In einem Datensatz $A[1\,..\,n]$ aus n Zahlen soll ein Wert x durch Vergleich mit den Datenelementen $A[1]$, $A[2]$, ... gesucht werden. Die Ausgabe ist j, falls $x = A[j]$ und NIL sonst.

   (a) Überlegen Sie sich einen Pseudocode für sequentielles Suchen.

   (b) Führen Sie eine Best-case- und eine Worst-case-Analyse durch. Führen Sie weiters eine Average-case-Analyse unter folgenden Annahmen durch: (i) Es soll das Modell der Zufallspermutationen (alle Permutationen gleich wahrscheinlich) zugrunde gelegt werden. (ii) Der gesuchte Datensatz ist im Feld A tatsächlich vorhanden.

3.) (a) Binäre Suche: Gegeben sei ein (aufsteigend) sortiertes Datenfeld $A[1\,..\,n]$ und ein Wert x. Die Ausgabe ist j, falls $x = A[j]$ und NIL sonst. Der Wert x wird mit dem mittleren Element $A[\lfloor n/2 \rfloor]$ der Folge verglichen. Dann ist entweder $A[\lfloor n/2 \rfloor] = x$, oder es muss nur mehr das halbe Datenfeld mit der gleichen Prozedur betrachtet werden. Schreiben Sie ein iteratives oder rekursives Programm in Pseudocode für die binäre Suche. Begründen Sie, warum die Laufzeit der binären Suche im schlechtesten Fall $\Theta(\log n)$ ist

   (b) Insertion-Sort verwendet die sequentielle Suche, um das sortierte Teilfeld $A[1\,..\,j-1]$ (rückwärts) zu durchsuchen. Kann statt dessen die binäre Suche benutzt werden, um die Laufzeit von Insertion-Sort im schlechtesten Fall auf $\Theta(n \log n)$ zu verbessern?

4.) Zeigen Sie für die harmonischen Zahlen $H_n = \sum_{k=1}^{n} \frac{1}{k}$ die Abschätzung $H_n = O(\log n)$, indem Sie

   (a) sie durch $N = \lfloor \log_2 n \rfloor$ Blöcke der Gestalt $\sum_{j=0}^{2^i-1} \frac{1}{2^i+j}$, $i = 1, \ldots, N$, abschätzen und anhand dieser Aufteilung $\sum_{k=1}^{n} \frac{1}{k} \leq \log_2(n) + 1$ verifizieren.

   (b) mit dem Cauchy'schen Integralkriterium abschätzen.

5.) Beweisen Sie die Identität

$$\sum_{1 \leq k \leq N} a_k b_k = a_N \sum_{1 \leq k \leq N} b_k - \sum_{1 \leq k < N} (a_{k+1} - a_k) \sum_{1 \leq i \leq k} b_i$$

   und berechnen Sie damit $\sum_{1 \leq k < n} \binom{k}{m} H_k$.

6.) (a) Zeigen Sie mit vollständiger Induktion, dass ein Algorithmus, dessen Laufzeit $T(n)$ (mit $n = 2^k$, $k \in \mathbb{N} \setminus \{0\}$) der Rekursion

$$T(n) = \begin{cases} 2, & \text{für } n = 2, \\ 2T(\frac{n}{2}) + n, & \text{für } n = 2^k,\ k = 2, 3, 4, \ldots \end{cases}$$

genügt, $T(n) = n \log_2 n$ erfüllt.

(b) Zeigen Sie mit vollständiger Induktion, dass ein Algorithmus, dessen Laufzeit $T(n)$ (mit $n = 2^k$, $k \in \mathbb{N} \setminus \{0\}$) der Rekursion

$$T(n) = \begin{cases} 1, & \text{für } n = 1, \\ 2T(\frac{n}{2}) + n^2, & \text{für } n = 2^k,\ k = 1, 2, 3, \ldots \end{cases}$$

genügt, $T(n) = 2n^2 - n$ erfüllt.

7.) Zum asymptotischen Vergleich von Folgen.

(a) Vergleichen Sie das asymptotische Verhalten von $f(n) = n!$ und $g(n) = (n+2)!$, d.h. überlegen Sie sich ob eine (welche) der Funktionen ein o, O, ω, Ω, Θ der anderen Funktion ist.

(b) Vergleichen Sie das asymptotische Verhalten von $f(n) = n^{\log_2 4}$ und $g(n) = 3^{\log_2(n)}$, d.h. überlegen Sie sich ob eine (welche) der Funktionen ein o, O, ω, Ω, Θ der anderen Funktion ist.

(c) Zeigen Sie an Hand der Definition, dass für positive Funktionen f und g die Beziehung

$$\max(f(n), g(n)) = \Theta(f(n) + g(n))$$

gilt.

(d) Folgt aus $f(n) = O(g(n))$, dass $2^{f(n)} = O(2^{g(n)})$?

(e) Gilt für alle positiven Funktionen f die Beziehung $f(n) = O((f(n))^2)$?

(f) Finden Sie eine Funktion f, sodass weder $f(n) = O(n)$ noch $f(n) = \Omega(n)$ gilt.

(g) Beweisen Sie, dass $\sum_{j=0}^{n} a^j = \Theta(a^n)$, für $a > 1$.

8.) Gegeben seien k sortierte Listen $A[n(i-1)+1\,..\,ni]$, $i = 1, \ldots, k$, die zu einer sortierten Liste $A[1\,..\,kn]$ verschmolzen werden sollen. Die folgende Vorgangsweise liegt nahe: Zuerst `Merge`$(A, 1, n, 2n)$, dann `Merge`$(A, 1, 2n, 3n)$, usw.

(a) Analysieren Sie die Laufzeit dieses Algorithmus.

(b) Finden Sie einen besseren Algorithmus, indem Sie die Strategie Divide & Conquer verwenden.