**Übungsblatt 1 für "Diskrete und geometrische Algorithmen"**

1.) (a) Selection Sort ist ein einfacher Sortieralgorithmus: sei $A$ eine Liste mit $n$ Elementen, so sucht man zunächst das kleinste Element und bringt es an die erste Position. Anschließend sucht man das zweitkleinste Element und bringt es an die zweite Position, etc.
Überlegen Sie sich einen Pseudocode für diesen Algorithmus und bestimmen Sie die Anzahl der notwendigen Schritte, die (in Ihrem Pseudocode) nötig sind, um eine $n$-elementige Menge zu sortieren.

(b) Sequentielle Suche: Überlegen Sie sich einen Pseudocode, welcher in einem $(n-1)$-elementigen Datensatz $A[1..n-1]$ einen Wert $x$ durch Vergleich mit den Datenelementen $A[1]$, $A[2]$, ... sucht und $j \in \{1, \ldots, n-1\}$ ausgibt, falls $x = A[j]$ und $n$ sonst. Machen Sie bei Ihrem Algorithmus eine

i. best-case Analyse (es werden Eingaben betrachtet, die minimale Laufzeiten (wie groß sind diese?) liefern).

ii. worst-case Analyse (es werden Eingaben betrachtet, die maximale Laufzeiten (wie groß sind diese?) liefern).

iii. average-case Analyse (Mittelwert der Laufzeiten für alle Eingaben) für das "random permutation model" für die Elemente des Datensatzes und im "erfolgreichen Fall", also unter der Annahme, daß der gesuchte Datensatz sich tatsächlich im Feld befindet.

2.) (a) Binäre Suche. Gegeben sei ein (aufsteigend) sortiertes Datenfeld $A[1..n]$ und ein Wert $x$. Das sogenannte Suchproblem, also einen Index $j$ mit $x = A[j]$ auszugeben, falls $x$ in $A$ vorkommt (erfolgreicher Fall) und anderenfalls einen speziellen Wert NIL auszugeben, falls $x$ nicht in $A$ vorkommt (erfolgloser Fall), kann dann einfach gelöst werden, indem $x$ mit dem mittleren Element der Folge verglichen wird und man nach diesem Vergleich entweder fündig geworden ist oder nur mehr das halbe Datenfeld mit der gleichen Prozedur zu betrachten braucht. Die binäre Suche ist ein Algorithmus, der genau diese Idee verwendet. 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) Im in der Vorlesung kennengelernten Algorithmus Insertion-Sort wird die Sequentielle Suche verwendet, um das sortierte Teilfeld $A[1..j-1]$ (rückwärts) zu durchsuchen. Können wir statt dessen die binäre Suche benutzen, um die Laufzeit von Insertion Sort im schlechtesten Fall auf $\Theta(n \log n)$ zu verbessern?

3.) Das Horner-Schema dient zur Auswertung von Polynomen. Die Grundidee ist die Umformung

$$p(x) = \sum_{k=0}^{n} a_k x^k = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n) \cdots )).$$

(a) Wiederholen Sie die genaue Funktionsweise des Horner-Schemas und überlegen Sie sich einen Pseudocode für das Horner-Schema.

(b) Überlegen Sie sich einen Pseudocode für die direkte Auswertung von Polynomen (direktes Einsetzen).

(c) Vergleichen Sie die Schrittzahlen der beiden Codes.

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 $\sum_{k=1}^{n} \frac{1}{k}$

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

(b) mit dem Cauchy'schen Integralkriterium abschätzen.

5.) (a) Zeigen Sie mit vollständiger Induktion, daß ein Algorithmus, dessen Laufzeit $T(n)$ (in Abhängigkeit von der Eingabegröße $n = 2^k$, mit einer ganzen Zahl $k \geq 1$) 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) Analog zeigen Sie mit vollständiger Induktion, daß ein Algorithmus, dessen Laufzeit $T(n)$ (in Abhängigkeit von der Eingabegröße $n = 2^k$, mit einer ganzen Zahl $k \geq 0$) 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.

6.) 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 $o$, $O$, $\omega$, $\Omega$, $\Theta$ der anderen Funktion ist.

(b) Vergleichen Sie das aymptotische 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 $o$, $O$, $\omega$, $\Omega$, $\Theta$ 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) Ist $f(n) = O((f(n))^2)$ ?

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

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

7.) Unimodal search: ein $n$-elementiges Array $A[1..n]$ heißt unimodal, falls es ein $m \in \{1, \ldots, n\}$ gibt, sodaß $A[1..m]$ eine streng monoton wachsende Folge und $A[m..n]$ eine streng monoton fallende Folge bildet. Insbesondere ist $A[m]$ das maximale Element des Arrays.
Geben Sie einen Divide and Conquer Algorithmus an, welcher in $O(\log n)$ Schritten das maximale Element eines unimodalen Arrays liefert.

8.) Man beschreibe einen Algorithmus der Ordnung $\Theta(n \log n)$ (im worst-case), der für ein gegebenes Feld $S[1..n]$ von $n$ ganzen Zahlen und einer ganzen Zahl $x$ bestimmt, ob es zwei Elemente in $S$ gibt, deren Summe genau $x$ ist. Die Beschreibung kann auch verbal erfolgen, sie muß aber zusammen mit der Laufzeitanalyse so detailliert sein, daß das Laufzeitverhalten $\Theta(n \log n)$ tatsächlich gezeigt wird.
**Anmerkung:** Wir dürfen annehmen, daß die Integer-Arithmetik des Prozessors die Grundrechnungsarten und die Betragsbildung jedenfalls einschließt und somit solch eine Operation nur $\Theta(1)$ benötigt.
**Hinweis:** Daß sich ein Feld der Länge $n$ in $\Theta(n \log n)$ (beispielsweise durch Merge-Sort) sortieren läßt, mag hierbei von Bedeutung sein.