Grundlegende Algorithmen und Datenstrukturen - Übungsblatt 1

1.)
Im Folgenden bezeichne log wie in der Vorlesung den Logarithmus zur Basis 2. Ordnen sie die 10 Funktionen aus der folgenden Tabelle absteigend nach ihrem asymptotischen Wachstum, d.h., geben sie eine Anordnung $g_1(n), g_2(n), ..., g_{10}(n)$ an, so dass $g_i(n) = \Omega(g_{i+1}(n))$ für $i = 1, ..., 9$. Antwort:

$$nlogn \quad (\sqrt{2})^{logn} \quad n^2 \quad n! \quad 4^{logn}$$

$$(\frac{3}{2})^n \quad n^3 \quad log(n!) \quad 2^{2^n} \quad n^{loglogn}$$

$$g_0 = \Omega(g_1) \Rightarrow g_0 \text{ ist untere Schranke von } g_1$$

Antwort:

$$n^{loglogn} < \sqrt{2}^{logn} < 4^{logn} < log(n!) < nlogn < (\frac{3}{2})^n < n^2 < n^3 < n! < 2^{2^n}$$

2.)
Entscheiden sie für jede der folgenden Aussagen, ob sie für asymptotisch nichtnegative Funktionen f und g immer wahr, niemals wahr oder manchmal wahr sind:

$a) f(n) + g(n) = \Theta(max(f(n), g(n)))$
$b) f(n) + O(f(n)) = \Theta(f(n))$
$c) f(n) = O(f(n)^2)$

Begründen sie ihre Antwort.

Antwort:
a) Korrekt, da die kleinere der beiden Laufzeiten jeweils quasi durch das Wachstum der größeren "verdeckt" wird.
b)falsch, da Θ die untere Schranke beschreibt, und durch Addition der Oberen Schranke auf die Funktion würde die Funktion den von Θ abgegrenzten Bereich verlassen.
c)Korrekt, da eine Funktion immer in ihrer quadratischen Laufzeit abgelaufen seien kann.

3.)
Geben sie ein allgemeines Verfahren an, mit dem man jeden vergleichsbasierten Sortieralgorithmus stabil machen kann.

Antwort:
Man könnte den Elementen im Ausgangsfeld einen weiteren Wert and die Hand geben, zum Beispiel aus einem einzelnen Wert ein Tupel machen, dann kann man erst nach dem Schlüsselwert sortieren und dann nach noch einmal darüber iterieren und bei gleichen Werten noch einmal nach dem zweiten Wert im Tupel sortieren.

4.)
Wir betrachten ein spezielles Sortierproblem: die n zu sortierenden Werte sind als Folge von $\frac{n}{k}$ aufeinanderfolgenden Teilfolgen der Länge jeweils k gegeben. Für jede dieser Teilfolgen gilt, dass ihre Elemente alle größer sind, als die Elemente der vorherigen Teilfolge.Geben sie eine asymptotische untere Schranke für die Anzahl der benötigten Vergleiche für dieses spezielle Sortierproblem an.

Antwort:
Für alle Folgen gilt, dass die Elemente einer Folge größer sind, als die der kleineren, und demnach auch kleiner als die der nächstgrößeren.

$$\forall x \in G_0, \forall y \in G_1, \forall z \in G_2 : x < y < z$$

kann man einen normalen Sortieralgorithmus wie z.B. Mergesort auf die jeweils ersten Elemente der n Folgen anwenden, somit erhält man eine Laufzeit von $n * log * n$ für die Vergleiche, die Vertauschungen nach dem Sortieren nehmen im worst-case einen Aufwand von $n * k$ in Anspruch

5.) Geben sie einen Algorithmus an, der eine Folge von n ganzen Zahlen Bereich $[1..k]$ so vorverarbeitet, dass Anfragen, wie viele Zahlen aus der Folge in einem Anfragebereich $[a..b]$ liegen, wobei $1 \leq a \leq b \leq k$, in Zeit O(1) beantwortet werden können. Ihr Algorithmus sollte Laufzeit $O(n + k)$ erreichen.

Antwort:
Eine Vorverarbeitung der Zahlenfolge von n Zahlen im Bereich $[1..k]$ könnte gelöst werden, in dem man einem Feld der Länge k die bisherige Anzahl der Zahlen bis zu diesem Wert zuordnet.

Eingabe A:
0 0 1 2 2 2 2 3 3 4 4 5 6 6 8 8 8 9 12

Verarbeitungsfeld B:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	3	7	9	11	12	14	14	17	18	18	18	18	18	18	18	18	18	18	18	18

Nun kann die Anfrage $[a, b]$ wie folgt beantwortet werden:

$$B[b] - B[a]$$

Das “indexieren” hat eine Laufzeit von $n + k$, da es k Zugriffe auf das Feld B gibt, und einmalige iterieren über Feld A in Zeit n durchläuft.