Algorithmen und Berechnungskomplexität I, WS 12/13
Präsenzübungsblatt 2
Universität Bonn, Institut für Informatik, Abteilung I

- *Hier kann man sich zu unserer Mailingliste anmelden: https://lists.iai.uni-bonn.de/mailman/listinfo.cgi/vl-algber1*
- *Wer noch keiner Übungsgruppe zugeordnet ist und dennoch am Übungsbetrieb teilnehmen möchte, kontaktiert bitte Rainer Penninger (penninge@cs.uni-bonn.de).*

Aufgabe 1: Asymptotisches Wachstum

Gegeben sei eine Abbildung $f : \mathbb{N} \to \mathbb{R}$ mit

$$f(n) = \begin{cases} 3n - 6 & \text{falls } \log n \geq 10^5 \\ 5n^2 - 31n & \text{falls } \log n < 10^5. \end{cases}$$

Bestimmen Sie eine möglichst einfache Funktion $g : \mathbb{N} \to \mathbb{R}$ mit $f = \Theta(g)$.

Aufgabe 2: Suche in einem Array

Gegeben ein Array A, in dem n Zahlen aufsteigend sortiert vorliegen. Das *Suchproblem* in A ist, gegeben eine Zahl x, die Frage nach dem Index von x in A, falls x in A vorkommt, oder andernfalls festzustellen, dass x nicht in A vorkommt. Das Verfahren *Binärsuche* vergleicht x mit dem Element y in A an der Position $n/2$. Ist $y > x$ dann wird rekursiv auf den Positionen $1..(n/2-1)$ weitergesucht. Ist $x = y$ so wird die Position von y ausgegeben, und falls $y < x$ gilt, wird auf den Positionen $(n/2+1)..n$ rekursiv weitergesucht. Zeigen Sie: Die Laufzeit der Binärsuche liegt in $\Omega(\log n)$. Es genügt zu zeigen, dass die Binärsuche für einen Beweis der Aussage „$x \notin A$“ mindestens $\Omega(\log n)$ Vergleiche der Form $y > x$?, $y = x$? oder $y < x$? benötigt. Folgern Sie, dass die Laufzeit für *jedes* Verfahren zur Suche in A Laufzeit $\Omega(\log n)$ hat.

Bittte wenden!

Aufgabe 3: Sortieren rückgängig machen

Formulieren Sie einen Algorithmus für das folgende Problem: Eine sortierte Zahlenfolge der Länge n soll „entsortiert“ werden, d.h. in eine unsortierte umgewandelt werden. Versuchen Sie, ein Verfahren zu formulieren, in dem jede unsortierte Reihenfolge mit gleicher Wahrscheinlichkeit erzeugt wird. Verwenden Sie dabei Pseudocode, wie er auch in der Vorlesung bzw. in Skript verwendet wird. Die sortierte Folge liege in einer Liste L vor, aus der das x-te Element jeweils mit `pop`(x) entnommen werden kann. Die verbleibenden Elemente von L rutschen anschließend zusammen. Die Resultatfolge soll in ein Array A der Größe n geschrieben werden. Sie dürfen davon ausgehen, dass eine Funktion `rand`(n) existiert, die bei jedem Aufruf mit Eingabewert n zufällig gleichverteilt eine ganze Zahl aus dem Intervall $[1, n]$ zurückgibt.

Aufgabe 4: Bubble-Sort

Der Algorithmus Bubble-Sort sortiert ein Array von n Zahlen wie folgt:

Algorithmus *BUBBLE-SORT(array A)*
for $i = n - 1$ downto 1 do
    BUBBLE(i,A);

Algorithmus *BUBBLE(integer i, Array A)*
for $j = 1$ to i do
    if $(A[j] > A[j + 1])$ then tausche $A[j]$ und $A[j + 1]$ in A

Sortieren Sie die folgende Zahlenfolge mit Hilfe des Bubble-Sort-Algorithmus:

$$56, 34, 47, 2, 55, 87, 74, 28, 104, 6, 3, 65, 51, 62, 58, 12$$

Geben Sie dabei in Form einer Tabelle nach jedem Aufruf der Funktion BUBBLE() das sortierte Teilergebnis an sowie die Anzahl der Vertauschungen, die dazu nötig waren.