Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr/ Dr. Jens Ernst
Johannes Nowak

Wintersemester 2005/06
Übungsblatt 8
14. Dezember 2005

Grundlegende Algorithmen

Abgabetermin: 21.12.2005 **nach** *der Vorlesung*

Aufgabe 1 (10 Punkte)

Geben Sie einen Algorithmus an, der die in einem binären Suchbaum gespeicherten Schlüssel in aufsteigend sortierter Reihenfolge ausgibt. Die Laufzeit soll linear in der Anzahl der Knoten des Baums sein. Der Baum sei durch einen Pointer auf die Wurzel gegeben, und an jedem Knoten seien Pointer auf dessen Kind(er) gespeichert.

Aufgabe 2 (10 Punkte)

Gegeben sei ein Array A der Größe n, in dem m Elemente mit paarweise verschiedenen Schlüsselwerten in aufsteigend sortierten Ordnung gespeichert sind. Mit $Rang(k)$ bezeichnen wir die Position des Elements mit Schlüssel k im sortierten Array. Betrachten Sie folgende Implementierung der Dictionary-Operation *is_element(k)*.

$i = 0$;
while $2^i \leq n$ and $A[2^i] < k$ **do**
  $i++$;
end while
if $2^i \leq n$ **then**
  $binary\_search(A, 2^{i-1}, 2^i, k)$;
else
  $binary\_search(A, 2^{i-1}, n, k)$;
end if

a) Beschreiben Sie, mit welcher Strategie hier nach dem Element mit Schlüssel k gesucht wird (z.B. durch Kommentierung des Pseudocodes).

b) Was sind die Kosten der Operation? Begründen Sie Ihre Antwort.

c) Für welche Schlüssel ist diese Funktion besonders effizient und warum? Wie kann man die Funktion noch verbessern, so dass die Anzahl der Operationen für *is_element* beschränkt ist durch $O(\min\{\log(\text{Rang}(k)), \log(m - \text{Rang}(k))\})$?

Aufgabe 3 (10 Punkte)

Sei ein Array A mit n paarweise verschiedenen natürlichen Zahlen gegeben. Ein Paar (i, j) heißt *Inversion* (bezüglich der aufsteigenden Sortierung), wenn $i < j$, aber $A[i] > A[j]$. Geben Sie einen Algorithmus an, der die Anzahl der Inversionen in einem Array A in Laufzeit $O(n \log n)$ bestimmt.
(*Hinweis*: Modifizieren Sie einen der in der Vorlesung behandelten Sortieralgorithmen geeignet.)

Aufgabe 4 (10 Punkte)

Gegeben sei eine Folge $(a_1, \ldots, a_n)$ ganzer Zahlen ($\in \mathbb{Z}$). Entwerfen Sie einen möglichst effizienten Algorithmus, der diejenige Teilfolge $(a_i, \ldots, a_j)$ mit $1 \leq i \leq j \leq n$ bestimmt, so daß $\sum_{k=i}^{j} a_k$ unter allen solchen Summen maximal ist. Zum Beispiel ist in der Folge $(-1, 4, -3, 5, -8, 1)$ die Teilfolge $(4, -3, 5)$ maximal bezüglich dieser Summe.