Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr/Dr. Sven Kosub
Stefan Eckhardt

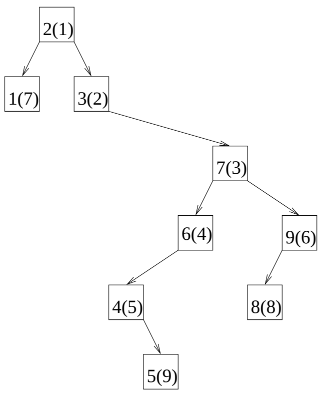
Wintersemester 2004/05
Übungsblatt 4
29. November 2004

Effiziente Algorithmen und Datenstrukturen I

Abgabetermin: 22.11.2004 **vor** *der Vorlesung*

Aufgabe 1

Sei A ein Array mit n Elementen. Der *kartesische Baum* des Arrays A ist ein wie folgt rekursiv definierter Binärbaum: Die Wurzel des Baumes enthält den Index m und den Wert des kleinsten Elementes von A, der linke Teilbaum der Wurzel ist ein kartesischer Baum des Teilarrays $A[1, \dots, m-1]$, der rechte Teilbaum der Wurzel ist ein kartesischer Baum des Teilarrays $A[m+1, \dots, n]$. Das folgende Beispiel zeigt einen kartesischen Baum für ein Array mit 9 Elementen.



7	1	2	5	9	4	3	8	6

Beispiel: Kartesischer Baum und Array. Die Knoten des Baumes sind mit Index und Wert (in Klammern) bezeichnet.

Geben Sie einen effizienten Algorithmus zur Konstruktion eines kartesischen Baums aus einem Array $A[1, \dots, n]$ an, der die Elemente aus dem Array sukzessive in den Baum einfügt, und analysieren Sie die Laufzeit Ihres Algorithmus. Verwenden Sie Amortisationsanalyse.

Aufgabe 2

Ein $m \times n$-**Young-Tableau** ist eine $m \times n$ Matrix mit Einträgen aus $\mathbb{N} \cup \{\infty\}$. Die Einträge in der i-ten Zeile sind von links nach rechts aufsteigend sortiert und die Einträge in der j-ten Spalte sind von oben nach unten aufsteigend sortiert. In einem Young-Tableau können also $r \leq m \cdot n$ Elemente verwaltet werden. (Hinweis: die ∞-Einträge werden wie *nicht existierende* Einträge behandelt)

Die Operation Extract-Min löscht das minimale Element aus einem Young-Tableau. Implementieren Sie Extract-Min so, dass die Laufzeit für eine Operation auf einem nicht leeren Young-Tableau $O(n+m)$ ist. Verwenden Sie dazu einen rekursiven Algorithmus, der das Problem der Größe $m \times n$ löst, indem er entweder das Teilproblem der Größe $m-1 \times n$ oder das Teilproblem der Größe $m \times n-1$ löst.

Aufgabe 3

Wie groß ist die Wahrscheinlichkeit dafür, dass beim Hashing mit Verkettung (Chaining) die Anzahl der Sondierungen bei der erfolgreichen Suche um mehr als 33% von der erwarteten Anzahl abweicht?

Aufgabe 4

Als offenes Hashverfahren werde „Double Hashing“ verwendet, wobei als Hashfunktion

$$h(k,i) = (h_1(k) + i \cdot h_2(k)) \bmod n$$

benutzt wird. Zeigen Sie, dass für $\gcd(n, h_2(k)) = d$, bei einer erfolglosen Suche $1/d$-tel der Hashtabelle untersucht wird, ehe die Suche zur Ausgangsposition $h_1(k)$ zurückkehrt und abgebrochen wird. D.h. für $d = 1$ wird die gesamte Hashtabelle durchsucht, ehe die Suche abbricht.