Technische Universität München
Institut für Informatik
Prof. Dr. J. Esparza/Dr. S. Gerke

WS 2000/01
Übungsblatt 4
14.November 2000

Effiziente Algorithmen und Datenstrukturen

(Abgabetermin: Dienstag, den 21.November 2000, in der Vorlesung)

Aufgabe 1

In dieser Aufgabe betrachten wir Suchbäume, die durch sukzessives Einfügen der Elemente $1, \ldots, N$ in einen leeren Baum entstanden sind. (Unter Einfügen verstehen wir das in der Vorlesung besprochene Verfahren). Wir nehmen an, daß die $N!$ verschiedenen Anordnungen der Zahlen $1, \ldots, N$ gleich wahrscheinlich sind. Zeigen Sie, daß die durchschnittliche Anzahl von Vergleichen, um ein Element zu finden $O(\log N)$ beträgt.
Eine Lösungsstrategie ist die folgende:
a) Sei $S(n)$ die durchschnittliche Anzahl von Vergleichen, um ein Element in einem Baum zu finden, der durch Einfügen der Elemente $1, 2, \ldots, n$ entstanden ist, wobei jede Anordnung der Elemente gleich wahrscheinlich ist. Wir sind also an $S(N)$ interessiert. Zeigen Sie,

$$S(n) = \frac{1}{n} \sum_{i=1}^{n} \left(1 + \frac{n-i}{n} S(n-i) + \frac{i-1}{n} S(i-1)\right).$$

(Tip: Mit Wahrscheinlichkeit $1/n$ ist das Element i die Wurzel.)
b) Setzen Sie $T(n) = nS(n)$ und verifizieren Sie die folgende Gleichung.

$$nT(n) - (n-1)T(n-1) = 2T(n-1) + 2n - 1.$$

c) Folgern Sie, daß

$$T(n) \leq \frac{n+1}{n} T(n-1) + 2.$$

d) Lösen sie die Rekursion, um zum folgenden Ergebnis zu kommen:

$$T(n) \leq 2(n+1)(\sum_{i=1}^{n+1} 1/i) - 2(n+1)$$

Aus der zweiten Aufgabe des 1.Übungsblattes folgt nun die Behauptung.

Aufgabe 2

Zeigen Sie, daß die erwartete Anzahl von Hashtabellenplätzen, die bei einer erfolgreichen Suche (mit gleicher Wahrscheinlichkeit für alle Schlüssel) inspiziert werden, bei Hashing mit linearem Sondieren nicht von der Reihenfolge abhängt, in der die Schlüssel in die anfangs leere Hashtabelle eingefügt worden sind.
Gilt die entsprechende Aussage auch für quadratisches Sondieren?

Aufgabe 3

Sei T ein Baum in einer Fibonacci-Heap-Datenstruktur mit n Knoten.
Zeigen oder widerlegen Sie, daß T die Höhe $O(\log n)$ hat.

Aufgabe 4

Führen Sie folgende Operationen auf dem angegebenen Fibonacci Heap (markierte Knoten sind grau unterlegt) aus, und stellen Sie den jeweils resultierenden Fibonacci Heap (gegebenfalls mit Zwischenstadien) wieder graphisch dar:

Insert(42), Delete_Min, Decrease_Key(91,60), Decrease_Key(71,61).