Prof. Dr. Norbert Blum
Maria Nikolaidou
Claus Rick

Institut für Informatik, Abt. IV

Vorlesung Informatik III
WS 1998/99
8. Übung

8. 12. 1998

Aufgabe 1:

Ein binärer Suchbaum mit knotenorientierter Speicherung heißt *Rot-Schwarz-Baum* falls folgende Eigenschaften erfüllt sind:

- Jeder Knoten ist entweder rot oder schwarz gefärbt;
- jedes Blatt ist schwarz gefärbt;
- falls ein Knoten rot gefärbt ist, sind seine beiden Kinder schwarz gefärbt;
- jeder einfache Pfad von einem Knoten zu einem nachfolgenden Blatt hat die gleiche Anzahl schwarzer Knoten.

a) Zeichnen Sie einen vollständigen binären Suchbaum der Höhe 4 für die Schlüssel $\{1, 2, \ldots, 15\}$. Färben Sie den Baum in drei verschiedenen Variationen, so daß die Anzahl schwarzer Knoten auf jedem Pfad von der Wurzel zu einem Blatt jeweils zwei, drei bzw. vier beträgt. **(3P)**

b) Was ist die größte mögliche Anzahl innerer Knoten in einem *Rot-Schwarz-Baum*, wenn jeder einfache Pfad von der Wurzel zu einem Blatt k schwarze Knoten enthält? Was ist die kleinste mögliche Anzahl? **(2P)**

c) Wie kann das Einfügen eines neues Elements realisiert werden, so daß der resultierende Suchbaum immer noch ein *Rot-Schwarz-Baum* bleibt. **(4P)**

Aufgabe 2:

a) Geben Sie eine AVL-Baum T der Höhe ≥ 10 und eine Streicheoperation an, so daß jeder Knoten auf dem Suchpfad rebalanciert wird. **(1P)**

b) Nehmen Sie an, wir wollen in jedem Knoten v eines AVL-Baumes zusätzlich das maximale Element und die Höhe des Teilbaumes mit Wurzel v speichern. Zeigen Sie, daß die Operationen Zugriff, Einfügen und Streichen weiterhin in $\mathcal{O}(\log n)$ Zeit durchgeführt werden können, wobei n die Anzahl der Knoten des AVL-Baumes ist. **(4P)**

c) Geben Sie möglichst effiziente Algorithmen für die Operationen $Ord(k, S)$ (k-tes Element der linearen Anordnung von S) und $Sequ(S)$ (lineare Anordnung von S) in AVL-Bäumen an. Was ist die Laufzeit Ihrer Algorithmen? **(2P)**

Aufgabe 3:

Nehmen Sie an, wir wollen eine verkettete Liste der Länge n durchsuchen, wobei jedes Element einen Schlüssel k zusammen mit seinem Hash-Wert $h(k)$ enthält. Jeder Schlüssel ist eine lange Zeichenkette. Wie könnte man die Hashwerte ausnutzen, um die Suche nach einem Schlüssel zu beschleunigen? **(1P)**

Aufgabe 4:

Angenommen, Sie benötigen für eine Anwendung eine sehr große Hashtabelle, deren Einträge nicht vor Verwendung der Tabelle explizit initialisiert werden können. Entwickeln Sie eine Methode, um dennoch beim Zugriff auf ein Element der Tabelle feststellen zu können, ob es sich um einen gültigen Eintrag handelt oder ob erstmals auf es zugegriffen wird. Die Zugriffsoperationen sollen weiterhin in Zeit $\mathcal{O}(1)$ möglich sein. (Tip: Sie benötigen eine zusätzliche Datenstruktur, die mit der Anzahl der tatsächlich verwendeten Tabellenelemente wächst. Außerdem müssen Sie die Tabellenelemente um eine Komponente erweitern.) **(3P)**