Johannes Blömer
u.v.a.m.

Paderborn, den 2. Juli 2004

Musterlösungen zur Vorlesung

Datenstrukturen und Algorithmen

SS 2004

Blatt 10

AUFGABE 1:

1. Der Baum nach Einfügen von Knoten mit Schlüsseln $5, 3, 6, 1, 2, 9, 8, 10$ ist in Abbildung 1 dargestellt.

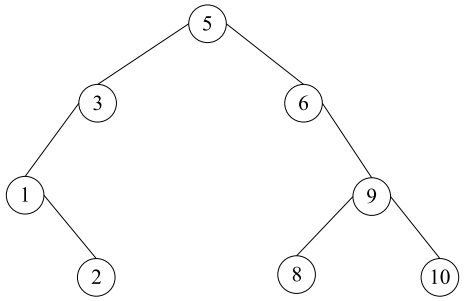

Abbildung 1: Erster Suchbaum zu Aufgabe 1

2. Der Baum nach Entfernen von Knoten mit Schlüsseln $6, 5, 1$ ist in Abbildung 2 dargestellt.

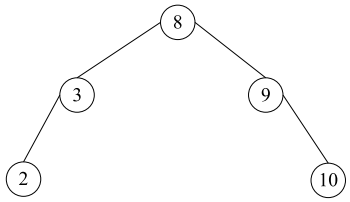

Abbildung 2: Zweiter Suchbaum zu Aufgabe 1

AUFGABE 2:

Um die Operation SELECT effizient zu unterstützen, speichern wir in jedem Knoten x eines Suchbaums im Feld $s[x]$ noch die Anzahl der Elemente im Teilbaum mit Wurzel x, deren Schlüssel höchstens so groß wie der Schlüssel $key[x]$ von x sind. Besitzt der linken Teilbaum des Baums mit Wurzel x genau l Knoten, dann ist $s[x] = l + 1$. Um nun das i-kleinste Element eines Suchbaums T zu bestimmen, beginnen wir mit der Suche in der Wurzel $x = root[T]$. Gilt $s[x] = i$, so ist $x = root[x]$ das gesuchte Element. Ist $s[x] > i$, so ist das gesuchte Element das i-kleinste Element im linken Teilbaum von x. Ist hingegen $i > s[x]$, so ist das gesuchte Element das $i - s[x]$-kleinste Element im rechten Teilbaum von x. Denn für alle Knoten y im rechten Teilbaum besitzen alle Knoten im linken Teilbaun von x sowie x selber einen kleineren Schlüssel als y. In Pseudo-Code erhalten wir dann folgenden rekursiven Algorithmus SELECT, der als Eingabe einen Wert i sowie einen Knoten x im Suchbaum erhält. Um die Suche nach dem i-kleinsten Elmement zu starten, wird der Algorithmus mit $x = root[x]$ aufgerufen.

SELECT(x, i)

```
1 if s[x] = i
2   then return x
3   else if s[x] > i
4          then SELECT(left[x], i)
5          else SELECT(right[x], i − s[x])
```

Die Korrektheit dieses Algorithmus folgt aus den Bemerkungen von oben. Für die Laufzeit L erhalten wir in Abhängigkeit von der Höhe h des Baums die Rekursion

$$L(h) \leq L(h-1) + c,$$

wobei c eine Konstante ist. Diese Rekursion ergibt sich daraus, dass wir immer nur in einem Teilbaum des Baums mit Wurzel x die Suche fortsetzen müssen. Wie wir schon häufiger in der Vorlesung gesehen haben, liefert die Rekursion $L(h) \leq L(h-1) + c$, dass $L(h) = \mathcal{O}(h)$.
Wir müssen noch zeigen, wie wir die Werte $s[x]$ bei den Operationen INSERT und DELETE aktualisieren. Betrachten wir zunächst INSERT. Fügen wir einen neuen Knoten x mit Schlüssel $key[x]$ in einen Suchbaum ein, so sucht INSERT zunächst nach der korrekten (Blatt-)Position für den Knoten x. Bei dieser Suche können wir die Werte $s[y]$ leicht aktualisieren. Wird nämlich in der Suche von einem Knoten y zu seinem linken Kind $left[y]$ gegangen, so müssen wir $s[y]$ um 1 erhöhen, da x im linken Teilbaum des Baums mit Wurzel y eingefügt wird. Wird hingegen von y zu $right[y]$ weitergegangen, so bleibt $s[y]$ unverändert. Schließlich muss noch für den neu eingefügten Knoten x der Wert $s[x]$ auf 1 gesetzt werden, denn x wird zu einem Blatt im Baum.
Beim Entfernen eines Knotens x wird entweder der Knoten x selber entfernt oder der direkte Nachfolger y von x wird entfernt. Vor dem Entfernen von y werden dabei noch die Daten von y in die Position des Knotens x kopiert. Um die Werte $s[x]$ beim Entfernen zu aktualisieren, bestimmen wir den Pfad des zu entfernenden Knoten (also entweder x oder sein direkter Nachfolger y) zur Wurzel des gesamten Baums. Wird dabei von einem Knoten z zu seinem Elternknoten w gegangen und ist $z = left[w]$, so erniedrigen wir den Wert $s[w]$ um 1. Ist hingegen $z = right[w]$, so bleibt $s[w]$ unverändert. Man überlegt sich leicht, dass dieses die Werte $s[w]$ korrekt aktualisiert.

AUFGABE 3:

1. Wir führen den Beweis mit einer vollständigen Induktion über die Höhe h. Bei der Induktionsverankerung gilt $h = 0$ oder $h = 1$. Im Fall $h = 0$ enthält der Baum aber nur ein Blatt, also einen NIL-Knoten. Damit ist die Anzahl der inneren Knoten gegeben durch $0 = F_0 - 1$. Im Fall $h = 1$ besteht der Baum aus der Wurzel und zwei Kindern, die jeweils NIL-Knoten sind. Damit ist dann die Anzahl der inneren Knoten gegeben durch $1 = F_1 - 1$.

   Nun zum Induktionsschluss. Wir nehmen an, dass wir die Behauptung der Aufgabe für Bäume mit Höhe höchstens $h - 1, h \geq 2$, bereits bewiesen haben. Sei nun T ein AVL-Baum mit Höhe h. Da T Höhe h besitzt, muss der rechte oder linke Teilbaum der Wurzel von T Höhe $h-1$ besitzen. Da T aber ein AVL-Baum ist, folgt dann, dass einer der Teilbäume der Wurzel von T Höhe $h-1$ und der andere mindestens Höhe $h-2$ besitzt. Damit ist nach Induktionsvoraussetzung die Anzahl der Knoten in den beiden Teilbäume der Wurzel von x zusammen mindestens $F_{h-1} - 1 + F_{h-2} - 1 = F_h - 2$. Da aber auch die Wurzel von T ein innerer Knoten ist und bislang noch nicht gezählt wurde, ist die Anzahl der inneren Knoten in T gegeben durch $F_h - 2 + 1 = F_h - 1$. Damit ist der Induktionsbeweis geführt.

2. Sei $F_h \geq ac^h$ für eine geeignete Konstante a und die in der Aufgabe genannte Zahl c. Ein AVL-Baum mit Höhe h hat nach dem vorangegangenen Aufgabenteil mindestens $F_h - 1$ viele innere Knoten. Hat nun ein AVL-Baum n innere Knoten, so muss für seine Höhe h gelten

   $$F_h - 1 \leq n.$$

   Zusammen mit $F_h \geq ac^h$ liefert dieses für die Höhe h

   $$ac^h \leq n + 1 \text{ oder } h \leq \log_c((n+1)/a) = \mathcal{O}(\log(n)).$$

AUFGABE 4:
Fügen wir zunächst einen Knoten mit Schlüssel 19 in den Rot-Schwarz-Baum ein, so erhalten wir den ersten Baum in Abbildung 3. Wir erhalten dann am Knoten 19 und seinem Elternknoten eine Verletzung der Rot-Schwarz-Eigenschaft. Hierbei liegt eine Verletzung nach Fall 2) aus der Vorlesung vor. Wir führen eine Links-Rotation aus und erhalten den zweiten Baum in Abbildung 3. Immer noch liegt eine Verletzung der Rot-Schwarz-Eigenschaft vor und zwar an den Knoten 12 und 19. Jetzt liegt Fall 3) aus der Vorlesung vor. Wir führen eine Rechts-Rotation um den Großelternknoten von 12 aus und färben Knoten um. Dann erhalten wir den letzten Baum in Abbildung 3. Dieses ist ein korrekter Rot-Schwarz-Baum.

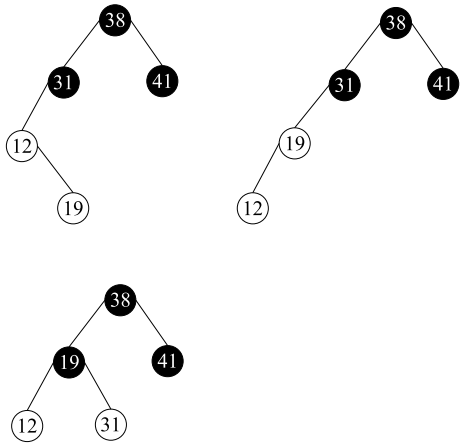

Abbildung 3: Einfügen von 19

Als nächstes fügen wir Knoten 8 ein und erhalten den ersten Baum in Abbildung 4. Wieder liegt eine Verletzung der Rot-Schwarz-Eigenschaft vor. Diesmal handelt es sich um eine Verletzung gemäß Fall 1) aus der Vorlesung. Die Behandlung dieses Falls führt aber bereits zu dem zweiten Baum in Abbildung 4, der ein korrekter Rot-Schwarz-Baum ist.

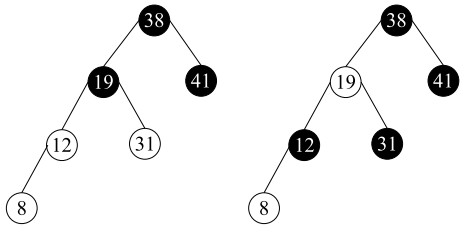

Abbildung 4: Einfügen von 8