Nicole Kaiser fl Tutorium: Do. 12-14 Uhr fl Tutor: Martin Klossek

04.01.2002

Theoretische Informatik 1, WS 2001/2002

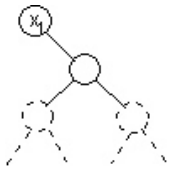
Übungsblatt 9

Keine Garantie für die Korrektheit der hier vorgestellten Lösungen.

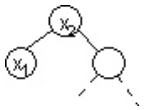
Aufgabe 9.1

Angenommen x_i sei im Splay-Baum. Nach der Operation $splay(x_i)$ ist der Knoten mit Schlüssel x_i die Wurzel.
Nach $splay(x_1)$ ist x_1 an der Wurzel. Die Schlüssel $x_2,...,x_n$ hängen rechts von x_1 , weil nach der Operation $splay(x_i)$ die Eigenschaften des binären Suchbaums erhalten bleiben.
Jetzt werden nacheinander $splay(x_2),\ldots,splay(x_n)$ ausgeführt. x_n ist dann Wurzel und $x_{n-1},...,x_1$ hängen sortiert links von x_n .

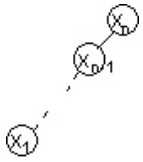
Nach $splay(x_1)$ sieht der Baum in etwa so aus:



Nach $splay(x_2)$ sieht er in etwa so aus:



Nach allen splay-Operationen hat der Baum dann schließlich diese Form:



Aufgabe 9.2

(a)

Sei u Vaterknoten von w . Der Geschwisterknoten w' bestehe aus folgenden Schlüsseln: $y_1<y_2<\ldots<y_{w'}$. w' ist immer ein direkter Geschwisterknoten, liegt also genau neben w .

Fall 2.1:

Ein Geschwisterknoten w' von w hat a Schlüssel. w klaut einen Schlüssel von u .

Fall a:
Sei w' ein linker Geschwisterknoten. Der Schlüssel $y_{w'}$ aus w' wird in den Vaterknoten eingefügt. Der rechte Teilbaum dieses Schlüssels wir als linker Teilbaum an w angehängt.

Fall b:
Sei w' ein rechter Geschwisterknoten. Der Schlüssel y_1 aus w' wird in den Vaterknoten eingefügt. Der linke Teilbaum dieses Schlüssels wir als rechter Teilbaum an w angehängt.

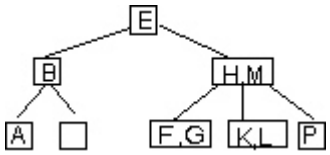
Fall 2.2:

Beide Geschwisterknoten von w haben nur $<a$ Schlüssel.
Verschmelze w und seinen Geschwisterknoten w' zu einem neuen Knoten w'' und füge einen Schlüssel des Vaters u hinzu.

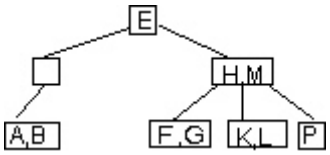
Fertig, wenn u mindestens $a-1$ Schlüssel hat. Wenn nicht, wende Fall 2.1 oder 2.2 rekursiv auf u an.

(b)

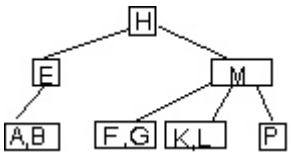
remove(D):



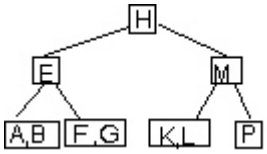
Fall 2.2:



Fall 2.1.b:



und umhängen:



Aufgabe 9.3

Gegeben sei ein AVL-Baum. Bilde beim kleinsten Schlüssel des Baums beginnend eine Liste mit der Inorder des Baumes. Dabei wird der Baum einmal von unten nach oben durchlaufen ($\rightarrow O(\log n)$). Der Index eines Schlüssels in der Liste bezeichnet dann den Rang des Schlüssels im Baum. Beim Erstellen der Liste schreibe den Index des aktuellen Schlüssels auch in den Knoten mit dem entsprechenden Schlüssel.

lookup(x):
in AVL-Bäumen $O(\log n)$.

select(k):
Greife auf das Element in der Liste über den entsprechenden Index in der Liste zu. $\rightarrow O(1)$

rang(x)=k:
Führe ein lookup(x) durch, um x zu finden. Der Rang von x steht im Knoten, in dem auch x steht. $\rightarrow O(\log n)$

insert(x):
in AVL-Bäumen $O(\log n)$. Zusätzlich muss nach dem insert(x) die Liste neu aufgebaut werden $\rightarrow O(\log n)$. Insgesamt also $2 \cdot O(\log n) = O(\log n)$.

remove(x):
analog zu insert(x). Insgesamt $O(\log n)$.