Übungen zu Algorithmen
WSI für Informatik
Kaufmann/Kuckuk/Förster

Wintersemester 2021/22
Universität Tübingen
20. Januar 2022

Aufgabensammlung Teil 11

Thema: (Binäre) Suchbäume und AVL-Bäume

Aufgabe 1: Knotenorientierte Binäre Suchbäume

a) Gegeben ein Pfad P von der Wurzel zum Blatt eines binären Suchbaumes. Sei B die Menge der Knoten auf diesem Pfad, A alle Knoten links von P und C alle Knoten rechts von P. Gilt dann $a \leq b \leq c$ für alle $a \in A$, $b \in B$ und $c \in C$?

b) Zeigen Sie: Hat ein Knoten in einem binären Suchbaum zwei Kinder, so hat sein Nachfolger kein linkes Kind und sein Vorgänger kein rechtes Kind. Dabei ist der Nachfolger (bzw. Vorgänger) eines Schlüssels der nächstgrößere (bzw. nächstkleinere) Schlüssel im Suchbaum.

   Gilt diese Aussage auch für ein Knoten mit nur einem Kind?

c) Zeigen oder widerlegen Sie: Sei T ein binärer Suchbaum und seien x und y Elemente in T. Der binäre Suchbaum, der sich nach Löschen von x und y (in dieser Reihenfolge) ergibt ist derselbe Suchbaum, der sich nach Löschen von y und x (in dieser Reihenfolge) ergibt.

Aufgabe 2: 1-3-Bäume

Dies ist eine alte Klausuraufgabe!

Wir betrachten eine spezielle Klasse von Bäumen, die wir *1-3-Bäume* nennen. In diesen Bäumen mit Wurzel hat jeder innere Knoten entweder 1 oder 3 Kinder, aber **niemals** 2.

a) Geben Sie an, wie viele Blätter ein 1-3-Baum mit n inneren Knoten mindestens und höchstens haben kann.

b) Wir wollen 1-3-Bäume als Suchbäume mit knotenorientierter Speicherung benutzen. Für 1-3-Suchbäume sollen nun entsprechende Regeln entworfen werden.

   (i) Wie sind die Regeln für eine Suchoperation nach einem Schlüssel x?

   (ii) Welche Schwierigkeit ergibt sich bei einer Einfügeoperation und wie könnten Sie diese beseitigen?

   Tipp: Hier gibt es keine richtigen und falschen Regeln, geben Sie eine Strategie an, die funktionieren würde!

Aufgabe 3: Bestimmung des Medians in AVL-Bäumen

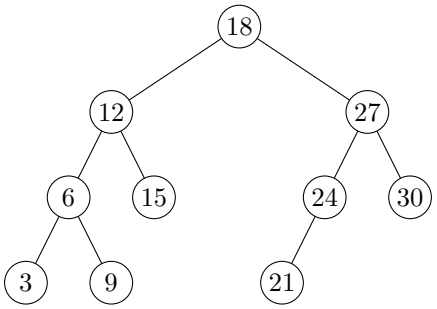
Dies ist eine alte Klausuraufgabe!

Sei $S \subset \mathbb{N}$ eine Menge von n Schlüsseln, die in einem AVL-Baum T abgespeichert sind. Für ein Element v in T sei $T(v)$ der Teilbaum von T mit der Wurzel v. Nehmen Sie an, dass jedes Element v im Baum T die Anzahl der Elemente in $T(v)$ gespeichert hat.
Entwerfen Sie einen Algorithmus in Pseudocode, der den Median der Menge S in Zeit $O(\log n)$ berechnet. Erklären Sie die Idee Ihres Algorithmus und begründen Sie außerdem die Korrektheit und die Laufzeit.

Aufgabe 4: AVL-Bäume

Dies ist eine alte Klausuraufgabe!

Gegeben sei der folgende AVL-Baum:



a) Fügen Sie die Zahlen 10, 8, 25 und 20 in dieser Reihenfolge in den gegebenen AVL-Baum ein. Geben Sie Ihren AVL-Baum nach jeder Einfüge-Operation an. Haben Sie Rotation verwendet? Wenn ja, welche?

b) Löschen Sie die Zahl 18 aus dem AVL-Baum, den Sie in a) am Ende erhalten haben. Brauchen Sie hier eine Rotation? Wenn ja, welche?

c) Gegeben sei ein Array A, das n Zahlen enthält. Können Sie die Zahlen in A mit Hilfe eines AVL-Baums **in der Zeit** $\mathcal{O}(n \log n)$ sortieren? Wenn ja, beschreiben Sie **kurz** wie.

Lösungen:

1. a) Stimmt nicht.

   b) Da der Knoten zwei Kinder hat, muss sein Nachfolger im rechten Teilbaum sein. Dort darf er aber kein linkes Kind haben. Gleiches Argument für den Vorgänger.

   Hat ein Knoten nur ein linkes Kind, kann sein Nachfolger z.B. sein Parent sein.

   c) Stimmt nicht. Man beachte, dass es verschiedene Reparaturstrategien gibt, je nachdem, ob der gelöschte Knoten ein oder zwei Kinder hat.

2. a) Mindestens 1, höchstens $2n + 1$

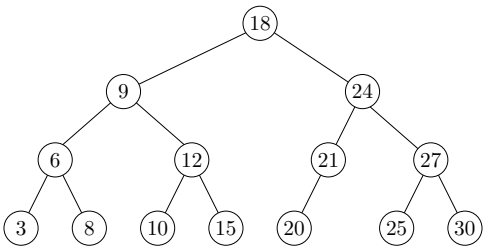
   b) Z.B. könnte der Schlüssel des Knotens v das Maximum der Schlüssel innerhalb seines Teilbaums $T(v)$ sein.

      (i) Dann wird bei der Suche geschaut, in welchem T0eilbaum der zu suchende Schlüssel liegt.

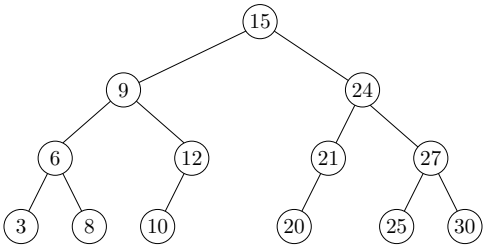
      (ii) Problematisch ist hier generell, dass man bei 1 oder 3 Kindern pro Knoten bleiben muss. Fügt man den neuen Schlüssel x bei einem Blatt ein, erhält dieses 1 Kind. Fügt man x unter einem Knoten mit einem Kind ein, so ist x größer als der Schlüssel des Kindes. Dann kann x zwischen die beiden gesetzt werden. Gleiches gilt bei der Einfügung unter einem Knoten v mit drei Kindern c_1, c_2, c_3 - x ist der größte Schlüssel. Man kann dann x als einziges Kind von v setzen c_1, c_2, c_3 als dessen drei Kinder nutzen.

3. Der Median ist (sortiert) der $\lceil n/2 \rceil$-te Schlüssel. Dadurch, dass jedes v in T weiß wie groß der Teilbaum ist, ist es einfach möglich, rauszufinden, ob der Mediam im linken oder rechten Teilbaum oder auf dem aktuellen Knoten v steht. Da nur einmal von Wurzel bis maximal zum Blatt gelaufen wird, ist die Laufzeit durch die Höhe des Baumes $\mathcal{O}(\log n)$ beschränkt.

4. a) Endresultat:



   b) Keine Rotation nötig:



   c) Ja, Elemente iterativ einfügen und in-order ausgeben.