max planck institut informatik

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Benjamin Doerr, Dr. Reto Spöhel **Wintersemester 2011/12**
Übungsleitung: Alexander Kobel

Übung zu Grundzüge von Algorithmen und Datenstrukturen

http://www.mpi-inf.mpg.de/departments/d1/teaching/ws11/grads/

Übungsblatt 8 Abgabe: Donnerstag, 15. Dezember 2011, 12:10 Uhr

Hinweise zur Abgabe der Aufgaben: Schreiben Sie klar und deutlich in gebührendem Abstand zu weiterem Text Ihren **Namen** und **Matrikelnummer**, ihre **Übungsgruppe als arabische Ziffer** sowie den **Namen Ihres Tutors** auf Ihre Abgabe. **Heften** Sie mehrere Blätter geeignet zusammen. Legen Sie ihre Lösung bis zum Abgabetermin in den **unteren rechten Briefkasten** im Erdgeschoss von Gebäude E1 3 (neben Hörsaal 001).

Aufgabe 1 (*Schriftliche Übung, 4 Punkte*)

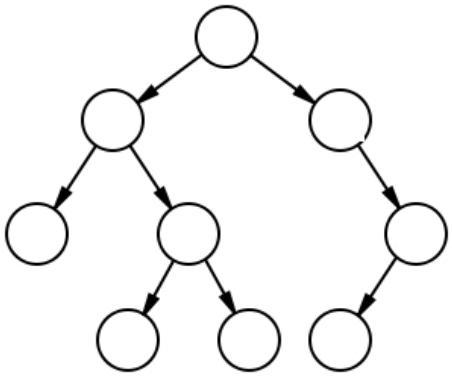
a) Definieren Sie formal korrekt, was ein geordneter ("wir unterscheiden 'linkes' und 'rechtes' Kind") binärer Baum (nicht "Suchbaum") mit Wurzel ist, und zwar über die Eigenschaften der parent-, left- und right-Abbildung. Beginnen Sie mit "Eine endliche Menge V zusammen mit drei Abbildungen parent, left, right: $V \to V \cup \{NIL\}$ heißt *geordneter binärer Baum mit Wurzel*, falls folgende Eigenschaften gelten:". Zählen Sie dann eine minimale Menge von Eigenschaften auf, die zusammen definierend für einen binären Baum sind (d.h., alles, was diese Eigenschaften erfüllt, soll ein binärer Baum sein, und alle binären Bäume sollen diese Eigenschaften erfüllen). Passen Sie auf, dass Sie keine Eigenschaft übersehen, "weil die doch eh' klar ist". Passen Sie andererseits auch auf, dass Sie nichts fordern, was sich schon aus anderen Eigenschaften (bspw. der Funktionseigenschaft von parent, left und right) ableiten lässt. [2P, Abzug für fehlende und überflüssige Eigenschaften, kein Beweis nötig]

b) Zeigen oder widerlegen Sie: Die Suchbaumeigenschaft aus Vorlesung und Skript ist äquivalent zu $(\forall v, w \in V : w = \text{left}(v) \Rightarrow \text{key}(w) \leq \text{key}(v)) \wedge (\forall v, w \in V : w = \text{right}(v) \Rightarrow \text{key}(w) \geq \text{key}(v))$. [1P, Skizze des Kernarguments/Gegenbeispiels genügt]

c) Geben Sie den Pseudocode einer nicht rekursiv definierten Funktion an, die in einem binären Suchbaum T einen Knoten mit maximalem Key berechnet. [1P, sauberer Pseudocode reicht, soweit er leicht verständlich ist]

Aufgabe 2 (*Schriftliche Übung, 4 Punkte und 2 Bonuspunkte*)

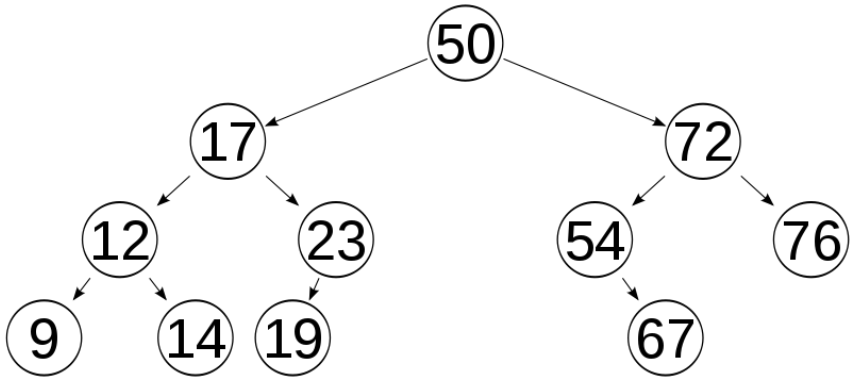
a)



Fügen Sie die Keys $1, \ldots, 9$ an die Knoten des skizzierten Baumes, so dass dieser ein binärer Suchbaum wird. [1P]

b) Beweisen Sie, dass es in a) nur eine Lösung gibt. Genauer: Zeigen Sie, dass es für jeden binären Baum mit Knotenmenge V, $|V| = n$, nur eine Funktion $\text{key} : V \to [n]$ gibt, die die Suchbaumeigenschaft erfüllt ($[n] := \{1, \ldots, n\}$). [1P]

c)



[Quelle: Wikipedia]

Beantworten Sie folgende Fragen für den skizzierten Baum. [2P]

(i) Wo fügt Insert einen Knoten mit Key 22 ein?

(ii) Wie sieht der Baum aus, nachdem der Knoten mit Key 50 gelöscht wurde? Bitte verwenden Sie hier und im folgenden die delete-Prozedur wie im Skript beschrieben.

(iii) Gibt es einen Knoten mit der Eigenschaft, dass Löschen und anschließendes Wieder-Einfügen die Tiefe des Baumes erhöht? Nennen Sie ggf. einen solchen Knoten.

(iv) In welcher Reihenfolge müssen Sie die vorhandenen Keys in einen anfänglich leeren Baum einfügen, damit dieser Baum dabei entsteht? Geben Sie zwei substantiell verschiedene Reihenfolgen an.

d) Beweise Sie (sauber und formal) folgende Aussage: Sei x ein Knoten in einem binären Suchbaum T mit $\text{right}(x) = NIL$. Ferner gebe es in T Knoten mit größerem Key als dem von x. Dann ist der Successor von x (der Knoten mit dem nächstgrößeren Key) ein Vorfahr von x. [2 Bonuspunkte]

Aufgabe 3 (*Schriftliche Übung, 5 Punkte*)

Erweitern Sie die Datenstruktur des binären Suchbaums geeignet, so dass sich eine Select-Funktion implementieren lässt, die $O(h)$ Zeit benötigt auf Bäumen der Höhe h. Zur Erläuterung: Analog zu sortierten Feldern soll Select bei Suchbäumen eine Funktion sein, die zu gegebenem $i \in \mathbb{N}$ einen Knoten v findet, so dass key(v) ein Element der Ordnung i in der (Multi-)Menge der Keys des Baumes ist.

Hinweis: Es genügt, an jedem Knoten eine geeignete naürliche Zahl zu speichern. Diese muss das Select in $O(h)$ Zeit ermöglichen. Sie muss aber auch bei Delete- und Insert-Operationen in Zeit $O(h)$ aktualisiert werden können.

Konkret erwarten wir folgendes von Ihnen: Beschreiben Sie präzise, welche Zahl Sie zusätzlich für jeden Knoten speichern. Geben Sie dann den Pseudocode Ihrer (vermutlich rekursiv definierten) Select-Funktion an. Beweisen Sie Korrektheit. Skizzieren Sie, wie Sie ihre Datenstruktur bei/nach einer Insert- und einer Delete-Operation aktualisieren.

Aufgabe 4 (*Schriftliche Übung, 3 Punkte*)

Ein zufälliger binärer Suchbaum mit den Zahlen $1, \ldots, n$ als Keys entsteht dadurch, dass wir diese Zahlen in einer zufälligen Reihenfolge in einen anfänglich leeren binären Suchbaum einfügen (mit der Prozedur Insert). Man kann zeigen, dass solche binären Bäume typischerweise nicht Tiefe $\Theta(n)$ haben, sondern nur $\Theta(\log n)$. Wir können hier leider nur etwas weniger leisten und zeigen, dass die typische Tiefe eines Knotens $\Theta(\log n)$ ist. Zur Vereinfachung der Sprache unterscheiden wir im folgenden nicht zwischen der Zahl x und dem Knoten v mit key$(v) = x$.

a) Betrachten Sie eine feste Reihenfolge der Insert-Operationen. Seien $x, y \in [n]$ mit $x < y$. Zeigen Sie, dass y ein Vorfahr von x ist genau dann, wenn y in der betrachteten Reihenfolge vor allen Knoten aus $[x, y-1]$ steht. [2P]

b) Folgern Sie aus a), dass die Wahrscheinlichkeit, dass bei zufälliger Reihenfolge der Inserts y ein Vorfahr von x ist, genau $1/(y-x+1)$ ist. [1P]

Nun sind wir schon fast fertig. Der Fall $x > y$ verläuft analog, so dass für beliebige x, y die Wahrscheinlichkeit, dass y ein Vorfahr von x ist, genau $p_{xy} := 1/(|y-x|+1)$ ist. Offenbar ist die Tiefe von x gleich der Anzahl seiner Vorfahren. Die erwartete Anzahl dieser ist genau $\sum_{y \neq x} p_{xy}$ [dieses einfache Argument benötigt, wenn man formal argumentieren will, den Begriff der Zufallsvariable sowie die "Linearität des Erwartungswertes"; beides lernen Sie bald in der MfI3], was maximal $2\sum_{i=2}^{n}(1/i) = O(\log n)$ ist (harmonische Reihe!).