

Einsendeaufgabe 6

Stefan Berger

3.

Die Methode insert kann unbalancierte Bäume ergeben. Geben Sie eine Reihenfolge von 10 Elementen, die einen Baum der Höhe 9 ergibt. Geben Sie eine Reihenfolge von 10 Elementen, die einen Baum der Höhe 3 ergibt. (Merke: Die Klasse `java.util.TreeMap` implementiert die Red-Black Suchbäume, die relativ balanciert bleiben.)

10 Elemente, Höhe 9: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

10 Elemente, Höhe 3: 5, 3, 8, 2, 4, 7, 9, 1, 6, 10

5.

Geben Sie Zeitkomplexität der Methoden insert, search, size, und height in asymptotischer Notation an, und begründen Sie Ihre Antwort.

INSERT (T, z):

Die grundlegende Anweisung ist der Vergleich $key[z] < key[x]$ in Zeile 5.

Die Eingabgröße ist die Anzahl der Knoten von T .

Kann der Knoten z in Ebene 1 eingefügt werden, dann wird die grundlegende Anweisung genau einmal ausgeführt. Hat der Baum mehr als einen Knoten, wird die grundlegende Anweisung höchstens so oft ausgeführt, wie der Baum hoch ist.

$$B(n) = 1$$

$$\text{INSERT} \in \Omega(1)$$

$$W(n) = h, h = \text{Höhe des Baumes}$$

$$\text{INSERT} \in \mathcal{O}(h)$$

SEARCH:

Der SEARCH Algorithmus ist rekursiv. Er hat den Verzweigungsfaktor $b = 1$, weil immer nur in einer Richtung weitergesucht wird. Die Größe der zu durchsuchenden Datenstruktur wird dadurch mit jedem Rekursionsschritt halbiert ($c = 2$). Der nicht-rekursive Anteil der Komplexität ist konstant ($k = 0$). Hat der Baum die Größe 0 findet kein rekursiver Aufruf statt und die Zeitkomplexität ist konstant.

$$T(n) = \begin{cases} \text{konstant}, & \text{falls } n = 0 \\ T(n/2) + 1, & \text{falls } n \geq 1 \end{cases}$$

$$(1/2)^0 = 1$$

$$T(n) \in \Theta(\log n)$$

SIZE:

Der SIZE Algorithmus ist rekursiv. Die rekursiven Aufrufe verzweigen sich nach beiden Kindknoten jedes Knoten ($b = 2$). Die Größe der zu durchsuchenden Datenstruktur wird dadurch mit jedem Rekursionsschritt halbiert ($c = 2$). Der nicht-rekursive Anteil der Komplexität ist konstant ($k = 0$). Hat der Baum die Größe 0 findet kein rekursiver Aufruf statt und die Zeitkomplexität ist konstant.

$$T(n) = \begin{cases} \textit{konstant}, & \textit{falls } n = 0 \\ 2T(n/2) + 1, & \textit{falls } n \geq 1 \end{cases}$$

$$2(1/2)^0 > 1$$

$$\ln 2 / \ln 2 = 1$$

$$T(n) \in \Theta(n)$$

HEIGHT:

Der HEIGHT Algorithmus ist rekursiv. Die rekursiven Aufrufe verzweigen sich nach beiden Kindknoten jedes Knoten ($b = 2$). Die Größe der zu durchsuchenden Datenstruktur wird dadurch mit jedem Rekursionsschritt halbiert ($c = 2$). Der nicht-rekursive Anteil der Komplexität ist konstant ($k = 0$). Hat der Baum die Größe 1, also Höhe 0, findet kein rekursiver Aufruf statt und die Zeitkomplexität ist konstant.

$$T(n) = \begin{cases} \textit{konstant}, & \textit{falls } n = 1 \\ 2T(n/2) + 1, & \textit{falls } n \geq 2 \end{cases}$$

$$2(1/2)^0 > 1$$

$$\ln 2 / \ln 2 = 1$$

$$T(n) \in \Theta(n)$$