Ausgabedatum: 05.12.2013
Version: 2013-12-04 17:23

8. Übungsblatt für Track ε zur Vorlesung Entwurf und Analyse von Algorithmen, WS 13/14

Abgabe: *Bis* ***Freitag****, 13.12.2013, 12:00 Uhr, Kasten im Treppenhaus 48-6.*

Bitte denken Sie an die Anmeldung zur Zwischenklausur und den Datentest; nur noch bis Mittwoch, den 11.12.!

Der Fehler der Woche

Mit T_{BC}, T_{AC} und T_{WC} der Laufzeit eines Algorithmus im Best-, Average- und Worst-Case gilt stets, dass

$$T_{\mathrm{AC}} = \frac{T_{\mathrm{BC}} + T_{\mathrm{WC}}}{2} .$$

Was ist falsch?

Basisaufgaben

B1: Aufbauen eines AVL-Baums — 2 Punkte

Konstruieren Sie aus der Schlüsselfolge $[91, 11, 49, 85, 56, 77, 32]$ den zugehörigen AVL-Baum. Starten Sie also mit dem leeren Baum, in den Sie den Schlüssel 91 einfügen. Fügen Sie in den resultierenden AVL-Baum den Schlüssel 11 ein, und so weiter.

Stellen Sie jeden so entstehenden Baum einzeln graphisch dar, wobei an jedem Knoten sein Balancegrad notiert werden soll. Wird nach dem Einfügen eine Rotation notwendig, so geben Sie an, welcher Typ von Rotation an welchem Knoten angewendet werden muss; der aus der Rotation resultierende Baum soll dann als neue Graphik dargestellt werden.

Geben Sie auch den aus der Schlüsselfolge $[91, 11, 49, 85, 56, 32, 77]$ resultierenden AVL-Baum an. Was fällt beim Vergleich der beiden Bäume auf, insbesondere in Hinblick auf die Laufzeit von Suchanfragen?

B2: Einfügen in Suchbäume

2 Punkte

Stellen Sie für die Folge

$$3, 6, 16, 17, 30, 31, 46, 60, 68, 69,\ 76, 78, 87, 95, 125$$

jeweils das Ergebnis dar, wenn die Elemente von links nach rechts in die anfänglich leere Datenstruktur eingefügt werden; und zwar für:

a) B-Baum der Ordnung $m = 4$

b) Digitaler Suchbaum

c) Trie

Hinweis: Für die Aufgabenteile b) und c) wandeln Sie die Elemente zunächst in 7-Bit Binärzahlen um.

B3/4: Operationen auf Tries

4 + 4 Punkte

Implementieren Sie Algorithmen für folgende Operationen im Trie und bestimmen Sie jeweils die Laufzeit im Worst-Case als Θ-Klasse.

B3: Einfügen eines Elements

B4: Löschen eines (gegebenen) Elements

Elemente werden dabei als binärer String der Länge `k` (globale Konstante) an die Prozeduren übergeben. Der vorhandene Trie soll jeweils im Sinne der Operation modifiziert werden.

Begründen Sie die Korrektheit Ihrer Implementierungen und die behaupteten Laufzeitschranken.

Aufbauaufgaben

22. Aufgabe

2 Punkte

Erläutern Sie, wo und wie in der Analyse der Suchkosten binärer Suchbäume das Random Permutation Model einfließt. Was müssten Sie ändern, wenn eine andere Verteilung angenommen werden müsste?

Wiederholung

Wir erinnern uns: Ein gewichtsbalancierter Baum ist ein binärer Baum, in dem für jeden Knoten $v \in V$ der Quotient $\frac{1+|V_l|}{1+|V|}$ einen Wert aus dem Intervall $[\alpha, 1-\alpha]$ annimmt, α fest aus $[0, 1/2]$. Dabei ist V die Menge der Knoten des Teilbaums mit Wurzel v und V_l die Menge der Knoten im linken Teilbaum des Knotens v.

23. Aufgabe

3 + 2 Punkte

a) Sei $\alpha \in (0, 1/2]$ fest und T ein beliebiger α-balancierter Baum mit $n \geq 1$ Knoten. Zeigen Sie für $h(T)$ die Höhe von T, dass

$$h(T) \leq 1 + \frac{\log_2(n+1) - 1}{\log_2\left(\frac{1}{1-\alpha}\right)} .$$

b) Zeigen oder widerlegen Sie: Es gibt $n_0 \in \mathbb{N}$ und $\alpha \in (0, 1/2]$, sodass alle AVL-Bäume mit $n \geq n_0$ Knoten α-balanciert sind.

24. Aufgabe

$2 + 2 + 3$ Punkte

Ein *Rot-Schwarz-Baum* ist ein binärer Suchbaum mit den folgenden Eigenschaften:

(1) Jeder Knoten ist entweder rot oder schwarz.

(2) Die Wurzel ist schwarz.

(3) Ist ein Knoten rot, so sind alle seine Kinder schwarz.

(4) Sei v ein beliebiger Knoten und T_v der Teilbaum, dessen Wurzel v ist. Dann haben alle (einfachen) Pfade von v zu fiktiven Blättern von T_v die gleiche Anzahl an schwarzen Knoten.

In einem Rot-Schwarz-Baum müssen alle fiktiven Blätter die selbe (fiktive) Farbe haben; per Konvention legen wir fest, dass alle fiktiven Blätter schwarz sind.

a) Zeigen oder widerlegen Sie: Es gibt $n_0 \in \mathbb{N}$, sodass alle Rot-Schwarz-Bäume mit $n \geq n_0$ Knoten höhenbalanciert sind.

b) Zeigen oder widerlegen Sie: Es gibt $n_0 \in \mathbb{N}$ und $\alpha \in (0, 1/2]$, sodass alle Rot-Schwarz-Bäume mit $n \geq n_0$ Knoten α-balanciert sind.

c) Zeigen oder widerlegen Sie: in Rot-Schwarz-Bäumen ist die Suchzeit – erfolgreich wie -los – in $\mathcal{O}(\log n)$, n die Knotenzahl.

25. Aufgabe

$3 + 3$ Punkte

a) Zeigen oder widerlegen Sie für jedes $\alpha \in [0, 1/2]$: Nach Einfügen eines neuen Elements in einen α-balancierten Baum genügt stets eine konstante Anzahl der bekannten AVL-Rotationen, um α-Balance wiederherzustellen.

b) Recherchieren Sie nach Antworten auf die folgenden Fragen:

   i) Was für Implementierungen von gewichts- bzw. α-balancierten Suchbäumen gibt es?

   ii) Wie verhalten sich die Laufzeiten der wesentlichen Operationen, insbesondere Einfügen und Suchen von Schlüsseln?

   Listen Sie Ihre Funde mit kurzen Zusammenfassungen in einer strukturierte Übersicht, natürlich mit sachgemäßen Referenzen!

   Hinweis: Als Ergebnis sind sowohl eine Liste von (nicht zwingend *allen* erdachten) Implementierungen, aber auch Verweise auf nicht konstruktive Positiv- wie Negativergebnisse denkbar! Eigene Ansätze sind natürlich ebenfalls gerne gesehen.