Algorithmen I

Tutorium 1 - 7. Sitzung

Dennis Felsing

dennis.felsing@student.kit.edu
www.stud.uni-karlsruhe.de/~ubcqr/algo

2011-05-30

Überblick

Rot-Schwarz-Bäume

Rot-Schwarz-Bäume

Definition

Ein **Rot-Schwarz-Baum** ist ein Binärer Suchbaum mit einem zusätzlichen Knotenattribut **Farbe**.

Rot-Schwarz-Eigenschaften

1. Jeder Knoten ist entweder rot oder schwarz
2. Die Wurzel ist schwarz
3. Jedes Blatt (NIL) ist schwarz
4. Falls ein Knoten rot ist, sind seine Kinder schwarz
5. Für jeden Knoten enthalten alle Pfade vom Knoten zu einem Blatt die selbe Anzahl schwarzer Knoten

Höhe

Ein Rot-Schwarz-Baum mit n inneren Knoten hat höchstens die Höhe $2 \lg(n+1)$.

Einfügen

Idee

1. Füge Knoten z an passender Stelle ein
2. Färbe z rot
3. Behebe entstandene Verstöße gegen Rot-Schwarz-Eigenschaften

Verstöße

1. Onkel von z ist rot
2. Onkel von z ist schwarz und z rechtes Kind
3. Onkel von z ist schwarz und z linkes Kind

Visualisierungen von
`http://www.cs.usfca.edu/~galles/visualization`

Löschen

Vorgehen

Löschen von z ähnlich wie bei Binären Suchbäumen:

- Wenn z weniger als zwei Kinder hat, durch Kind ersetzen und $y = z$
- Wenn z zwei Kinder hat, ersetzen durch minimales Element y aus rechtem Teilbaum und y wie z färben

Löschen

Vorgehen

Löschen von z ähnlich wie bei Binären Suchbäumen:

- Wenn z weniger als zwei Kinder hat, durch Kind ersetzen und $y = z$
- Wenn z zwei Kinder hat, ersetzen durch minimales Element y aus rechtem Teilbaum und y wie z färben
- Falls y ursprünglich rot war, dann sind Rot-Schwarz-Eigenschaften erhalten (Warum?)

Löschen

Vorgehen

Löschen von z ähnlich wie bei Binären Suchbäumen:

- Wenn z weniger als zwei Kinder hat, durch Kind ersetzen und $y = z$
- Wenn z zwei Kinder hat, ersetzen durch minimales Element y aus rechtem Teilbaum und y wie z färben
- Falls y ursprünglich rot war, dann sind Rot-Schwarz-Eigenschaften erhalten (Warum?)
- Falls y ursprünglich schwarz war, dann Rot-Schwarz-Eigenschaften wiederherstellen ab x, dem ursprünglichen Kind von y

Löschen

Ausgangssituation

x ist schwarz und sein ehemaliger Vater, y, war ebenfalls schwarz $\Rightarrow$ Eigenschaft 5 verletzt. Zum Beheben färben wir x **doppelt schwarz** $\Rightarrow$ Eigenschaft 1 verletzt

Löschen

Ausgangssituation

x ist schwarz und sein ehemaliger Vater, y, war ebenfalls schwarz $\Rightarrow$ Eigenschaft 5 verletzt. Zum Beheben färben wir x **doppelt schwarz** $\Rightarrow$ Eigenschaft 1 verletzt

Wiederherstellen der RB-Eigenschaften

Sei x das linke Kind seines Vaters (symmetrisch rechtes) und w der rechte Bruder von x. Mögliche Fälle:

1. w ist rot
2. w und dessen Kinder sind schwarz
3. w und dessen rechtes Kind sind schwarz, linkes Kind rot
4. w ist schwarz und dessen rechtes Kind ist rot

$\Rightarrow$ Umfärben und rotieren bis wir Wurzel erreichen und fertig sind

B-Bäume

Motivation

Durchschnittliche Zugriffszeiten

DDR3-Hauptspeicher: 50ns

VelociRaptor-Festplatte: 5ms

Motivation

Durchschnittliche Zugriffszeiten

DDR3-Hauptspeicher: 50ns

VelociRaptor-Festplatte: 5ms

$\Rightarrow$ 100000 Zugriffe auf Hauptspeicher in der Zeit für einen Zugriff auf Festplatte

Motivation

Durchschnittliche Zugriffszeiten

DDR3-Hauptspeicher: 50ns

VelociRaptor-Festplatte: 5ms

$\Rightarrow$ 100000 Zugriffe auf Hauptspeicher in der Zeit für einen Zugriff auf Festplatte

Große Bäume passen nicht komplett in den Hauptspeicher.

Motivation

Durchschnittliche Zugriffszeiten

DDR3-Hauptspeicher: 50ns

VelociRaptor-Festplatte: 5ms

$\Rightarrow$ 100000 Zugriffe auf Hauptspeicher in der Zeit für einen Zugriff auf Festplatte

Große Bäume passen nicht komplett in den Hauptspeicher.
Festplatten lesen gleich ganze **Seiten** aus, die 2^{11} bis 2^{14} Bytes groß sind.

Motivation

Durchschnittliche Zugriffszeiten

DDR3-Hauptspeicher: 50ns

VelociRaptor-Festplatte: 5ms

$\Rightarrow$ 100000 Zugriffe auf Hauptspeicher in der Zeit für einen Zugriff auf Festplatte

Große Bäume passen nicht komplett in den Hauptspeicher. Festplatten lesen gleich ganze **Seiten** aus, die 2^{11} bis 2^{14} Bytes groß sind.

Idee

Anzahl von Zugriffen auf Hintergrundspeicher minimieren, indem man mehrere Verzweigungen pro Knoten speichert. Auslesen eines Knotens entspricht dann Auslesen einer Seite.

Definition

Knotenattribute

- n Schlüssel s_x
- $n + 1$ Zeiger p_x auf Kinder

Definition

Knotenattribute

- n Schlüssel s_x
- $n+1$ Zeiger p_x auf Kinder

Dabei gilt:

- p_0 verweist auf Teilbaum mit Schlüsseln kleiner s_1
- p_i verweist auf Teilbaum mit Schlüsseln zwischen s_i und s_{i+1}
- p_l verweist auf Teilbaum mit Schlüsseln größer s_l

Definition

Eigenschaften von B-Baum der Ordnung m

- Alle Blätter haben die gleiche Tiefe
- Jeder Knoten hat mindestens $\lceil \frac{m}{2} \rceil$ Kinder
- Die Wurzel hat mindestens 2 Kinder
- Jeder Knoten hat höchstens m Kinder
- Jeder innere Knoten mit i Kindern hat $i - 1$ Schlüssel

Höhe

Für die Höhe h eines B-Baumes gilt: $h \leq \log_{\lceil \frac{m}{2} \rceil} n$

Vergleich zu Rot-Schwarz-Bäumen

Keine Vor- oder Nachteile im O-Kalkül gegenüber Rot-Schwarz-Bäumen:

	Average Case	Worst Case
Platz	$O(n)$	$O(n)$
Search	$O(\log n)$	$O(\log n)$
Insert	$O(\log n)$	$O(\log n)$
Delete	$O(\log n)$	$O(\log n)$

Aber: Anzahl der Zugriffe auf Hintergrundspeicher proportional zur Höhe

Suchen

Vorgehen

Suchen

Vorgehen

1. Beginne bei Wurzel
2. Falls Schlüssel im Knoten, dann zurückgeben
3. Ansonsten zu passendem Kind gehen und bei Schritt 2 weitermachen

Suchen

Vorgehen

1. Beginne bei Wurzel
2. Falls Schlüssel im Knoten, dann zurückgeben
3. Ansonsten zu passendem Kind gehen und bei Schritt 2 weitermachen

Zeitaufwand?

- Anzahl Zugriffe auf Hintergrundspeicher: $O(\log_m n)$

Suchen

Vorgehen

1. Beginne bei Wurzel
2. Falls Schlüssel im Knoten, dann zurückgeben
3. Ansonsten zu passendem Kind gehen und bei Schritt 2 weitermachen

Zeitaufwand?

- Anzahl Zugriffe auf Hintergrundspeicher: $O(\log_m n)$
- Innerhalb eines Knotens: $O(m)$.

Suchen

Vorgehen

1. Beginne bei Wurzel
2. Falls Schlüssel im Knoten, dann zurückgeben
3. Ansonsten zu passendem Kind gehen und bei Schritt 2 weitermachen

Zeitaufwand?

- Anzahl Zugriffe auf Hintergrundspeicher: $O(\log_m n)$
- Innerhalb eines Knotens: $O(m)$. Mit binärer Suche: $O(\log m)$

Suchen

Vorgehen

1. Beginne bei Wurzel
2. Falls Schlüssel im Knoten, dann zurückgeben
3. Ansonsten zu passendem Kind gehen und bei Schritt 2 weitermachen

Zeitaufwand?

- Anzahl Zugriffe auf Hintergrundspeicher: $O(\log_m n)$
- Innerhalb eines Knotens: $O(m)$. Mit binärer Suche: $O(\log m)$
- Insgesamt: $O(\log m \log n)$

Einfügen

Vorgehen

- Suche nach einzufügendem Schlüssel bis zum passenden Blatt
- Falls im Blattknoten noch Platz, einfach an entsprechender Stelle einfügen
- Ansonsten Überlauf:
  - Mittleres Element wird in Vaterknoten verschoben
  - Überprüfen ob bei Vaterknoten Überlauf

Löschen

Vorgehen

- Suche nach zu löschendem Schlüssel
- Fall 1: Schlüssel in Blatt: Schlüssel entfernen, möglichen Unterlauf behandeln durch Knotenverschmelzung
- Fall 2: Schlüssel in innerem Knoten: Finde neuen Seperator der zu entfernendes Objekt ersetzt

Quiz

- Binäre Suche läuft im Worst Case in $O(\log n)$

Quiz

- Binäre Suche läuft im Worst Case in $O(\log n)$ ✓
- Rot-Schwarz-Bäume werden häufig bei Datenbanken eingesetzt

Quiz

- Binäre Suche läuft im Worst Case in $O(\log n)$ ✓
- Rot-Schwarz-Bäume werden häufig bei Datenbanken eingesetzt ×
- Binäre Suchbäume, RB-Bäume und B-Bäume haben die selben Laufzeiten für Suchen, Einfügen, Löschen

Quiz

- Binäre Suche läuft im Worst Case in $O(\log n)$ ✓
- Rot-Schwarz-Bäume werden häufig bei Datenbanken eingesetzt ×
- Binäre Suchbäume, RB-Bäume und B-Bäume haben die selben Laufzeiten für Suchen, Einfügen, Löschen ×
- Binäre Suchbäume sind eine Sonderform von B-Bäumen mit $m = 2$

Quiz

- Binäre Suche läuft im Worst Case in $O(\log n)$ ✓
- Rot-Schwarz-Bäume werden häufig bei Datenbanken eingesetzt ×
- Binäre Suchbäume, RB-Bäume und B-Bäume haben die selben Laufzeiten für Suchen, Einfügen, Löschen ×
- Binäre Suchbäume sind eine Sonderform von B-Bäumen mit $m = 2$ ×
- B-Bäume sind auch mit SSDs sinnvoll

Quiz

- Binäre Suche läuft im Worst Case in $O(\log n)$ ✓
- Rot-Schwarz-Bäume werden häufig bei Datenbanken eingesetzt ×
- Binäre Suchbäume, RB-Bäume und B-Bäume haben die selben Laufzeiten für Suchen, Einfügen, Löschen ×
- Binäre Suchbäume sind eine Sonderform von B-Bäumen mit $m = 2$ ×
- B-Bäume sind auch mit SSDs sinnvoll ✓

Quiz

- Ein RB-Baum, der nur durch Aufrufe von RB-Insert erzeugt wurde, enhält mindestens einen roten Knoten

Quiz

- Ein RB-Baum, der nur durch Aufrufe von RB-Insert erzeugt wurde, enhält mindestens einen roten Knoten ×
- Ein RB-Baum, der nur durch (mind. 2) Aufrufe von RB-Insert erzeugt wurde, enhält mindestens einen roten Knoten

Quiz

- Ein RB-Baum, der nur durch Aufrufe von RB-Insert erzeugt wurde, enhält mindestens einen roten Knoten ×
- Ein RB-Baum, der nur durch (mind. 2) Aufrufe von RB-Insert erzeugt wurde, enhält mindestens einen roten Knoten ✓
- Wie viele interne Knoten kann ein RB-Baum mit Schwarz-Höhe k minimal haben?

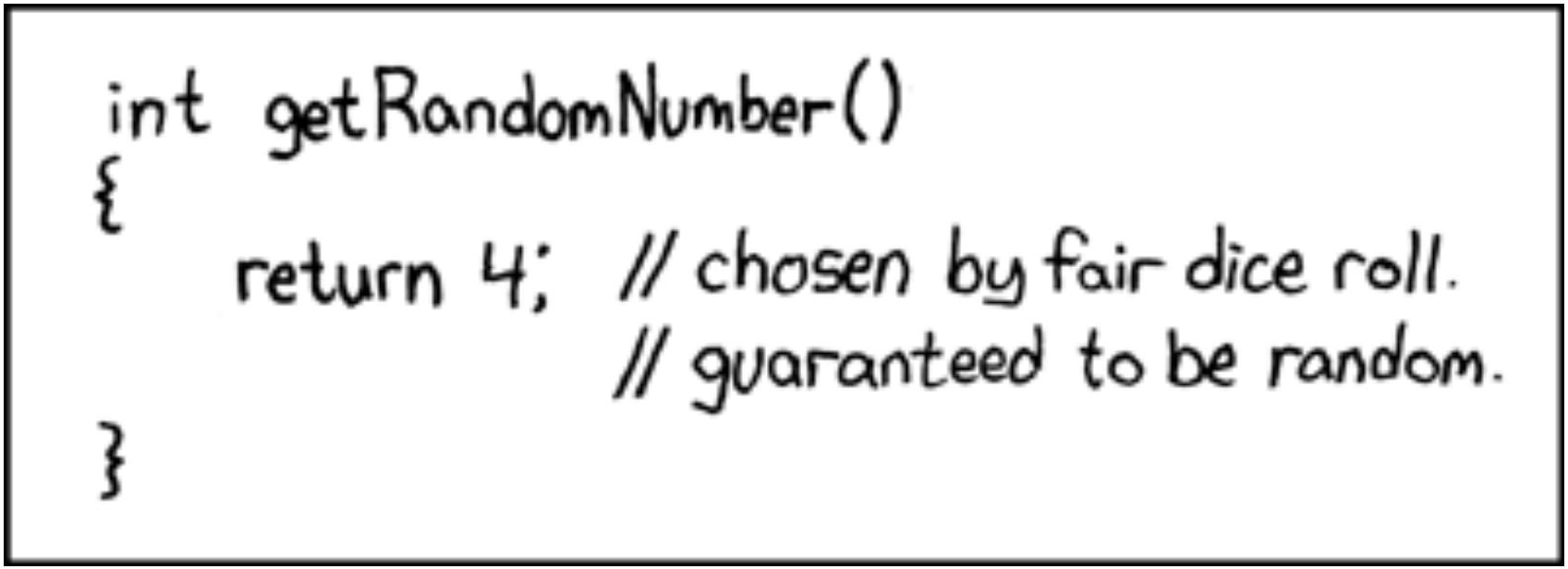
Quiz

- Ein RB-Baum, der nur durch Aufrufe von RB-Insert erzeugt wurde, enhält mindestens einen roten Knoten ×
- Ein RB-Baum, der nur durch (mind. 2) Aufrufe von RB-Insert erzeugt wurde, enhält mindestens einen roten Knoten ✓
- Wie viele interne Knoten kann ein RB-Baum mit Schwarz-Höhe k minimal haben? $\sum_{i=0}^{k-1} 2^i = 2^k - 1$
- Wie viele interne Knoten kann ein RB-Baum mit Schwarz-Höhe k maximal haben?

Quiz

- Ein RB-Baum, der nur durch Aufrufe von RB-Insert erzeugt wurde, enhält mindestens einen roten Knoten ×
- Ein RB-Baum, der nur durch (mind. 2) Aufrufe von RB-Insert erzeugt wurde, enhält mindestens einen roten Knoten ✓
- Wie viele interne Knoten kann ein RB-Baum mit Schwarz-Höhe k minimal haben? $\sum_{i=0}^{k-1} 2^i = 2^k - 1$
- Wie viele interne Knoten kann ein RB-Baum mit Schwarz-Höhe k maximal haben? $\sum_{i=0}^{2k-1} 2^i = 2^{2k} - 1$

Übersicht


int getRandomNumber()
{
return 4; // chosen by fair dice roll.
// guaranteed to be random.
}