Algorithmen I

Tutorium 1 - 6. Sitzung

Dennis Felsing

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

2011-05-23

Überblick

Binäre Suchbäume

Binäre Suchbäume

Idee

Suchen in $O(\log n)$

Definition

Ein **Binärer Suchbaum** ist ein Binärer Baum, für den die **Binäre-Suchbaum-Eigenschaft** gilt.

Binäre Suchbäume

Binäre-Suchbaum-Eigenschaft

Sei x ein Knoten in einem Binären Suchbaum.
Falls y ein Knoten im linken Unterbaum von x ist, dann gilt $y.\text{schlüssel} \leq x.\text{schlüssel}$.
Falls y ein Knoten im rechten Unterbaum von x ist, dann gilt $y.\text{schlüssel} \geq x.\text{schlüssel}$.

Erinnerung: Heapeigenschaft

Bei Max-Heaps: $A[\textsc{Parent}(i)] \geq A[i]$

Unterschiede? Bei Binären Suchbäumen ist sortierte Ausgabe einfacher.

Suchen

TREE-SEARCH(x, k)

```
1  if x == NIL oder k == x.schlüssel
2        return x
3  if k < x.schlüssel
4        return TREE-SEARCH(x.links, k)
5  else return TREE-SEARCH(x.rechts, k)
```

Ohne Rekursion?

ITERATIVE-TREE-SEARCH(x, k)

```
1  while x ≠ NIL und k ≠ x.schlüssel
2        if k < x.schlüssel
3              x = x.links
4        else x = x.rechts
5  return x
```

Einfache Operationen

TREE-MINIMUM(x)

```
1  while x.links ≠ NIL
2        x = x.links
3  return x
```

TREE-MAXIMUM(x)

```
1  while x.rechts ≠ NIL
2        x = x.rechts
3  return x
```

Einfügen

```
TREE-INSERT(T, z)
 1  y = NIL
 2  x = T.wurzel
 3  while x ≠ NIL
 4        y = x
 5        if z.schlüssel < x.schlüssel
 6              x = x.links
 7        else x = x.rechts
 8  z.vater = y
 9  if y == NIL
10        T.wurzel = z
11  elseif z.schlüssel < y.schlüssel
12        y.links = z
13  else y.rechts = z
```

Löschen

```
TRANSPLANT(T, u, v)
1   if u.vater == NIL
2         T.wurzel = v
3   elseif u == u.vater.links
4         u.vater.links = v
5   else u.vater.rechts = v
6   if v ≠ NIL
7         v.vater = u.vater
```

Löschen

```
TREE-DELETE(T, z)
 1  if z.links == NIL // Fall a)
 2        TRANSPLANT(T, z, z.rechts)
 3  elseif z.rechts == NIL // Fall b)
 4        TRANSPLANT(T, z, z.links)
 5  else y = TREE-MINIMUM(z.rechts) // Fall c) und d)
 6        if y.vater ≠ z // Fall d)
 7              TRANSPLANT(T, y, y.rechts)
 8              y.rechts = z.rechts
 9              y.rechts.vater = y
10        TRANSPLANT(T, z, y)
11        y.links = z.links
12        y.links.vater = y
```

Kreativaufgabe

Aufgabe

Schreibe eine Prozedur INORDER-TREE-WALK(w), so dass INORDER-TREE-WALK($T.wurzel$) alle Elemente eines Binären Suchbaumes T in sortierter Reihenfolge ausgibt.

- Einfach: Rekursiv
- Mittel: Nichtrekursiv mit Stack
- Schwer: Nichtrekursiv ohne zusätzliche Datenstrukturen

Welchen Zeit- und Platzaufwand hat dein Algorithmus?
Kommt dein Algorithmus ohne Vater-Zeiger aus?

Probleme

Schnelle Laufzeiten bei balancierten Suchbäumen. Balance kann aber nicht garantiert werden.
Abhilfe schaffen **Rot-Schwarz-Bäume**.

Rot-Schwarz-Bäume

Rot-Schwarz-Bäume

Definition

Ein **Rot-Schwarz-Baum** ist ein Binärer Suchbaum mit einem zusätzlichen Knotenattribut **Farbe**, dass entweder den Wert ROT oder $\mathrm{SCHWARZ}$ hat.

Rot-Schwarz-Baum-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

Definition

Die **Schwarz-Höhe eines Knotens** x, $bh(x)$, ist die Anzahl der schwarzen Knoten zwischen x und einem Blatt, ohne x selbst.
Die **Schwarz-Höhe eines Baumes** ist die Schwarz-Höhe der Wurzel.

Höhe

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

Was heißt das? Jeder Pfad von der Wurzel zu einem Blatt ist maximal doppelt so lang wie jeder andere. Operationen Tree-Search, Tree-Minimum, Tree-Maximum lassen sich in $O(\lg n)$ ausführen. Tree-Insert und Tree-Delete können wir nicht direkt verwenden, da Rot-Schwarz-Baum-Eigenschaften nicht erhalten bleiben.

Rot-Schwarz-Bäume

Aufgabe

Zeichne einen Binären Suchbaum der Höhe 3 mit den Schlüsseln $\{1, 2, ..., 15\}$. Füge die NIL-Blätter hinzu und färbe die Knoten auf drei verschiedene Arten, so dass die Schwarz-Höhe der resultierenden Rot-Schwarz-Bäume 2, 3 und 4 ist.

Drehungen

LEFT-ROTATE(T, x)

```
 1  y = x.rechts
 2  x.rechts = y.links
 3  if y.links ≠ T.nil
 4        y.left.vater = x
 5  y.vater = x.vater
 6  if x.vater == T.nil
 7        T.wurzel = y
 8  elseif x == x.vater.links
 9        x.vater.links = y
10  else x.vater.rechts = y
11  y.links = x
12  x.vater = y
```

Drehungen erhalten die Binärer-Suchbaum-Eigenschaft.

Einfügen

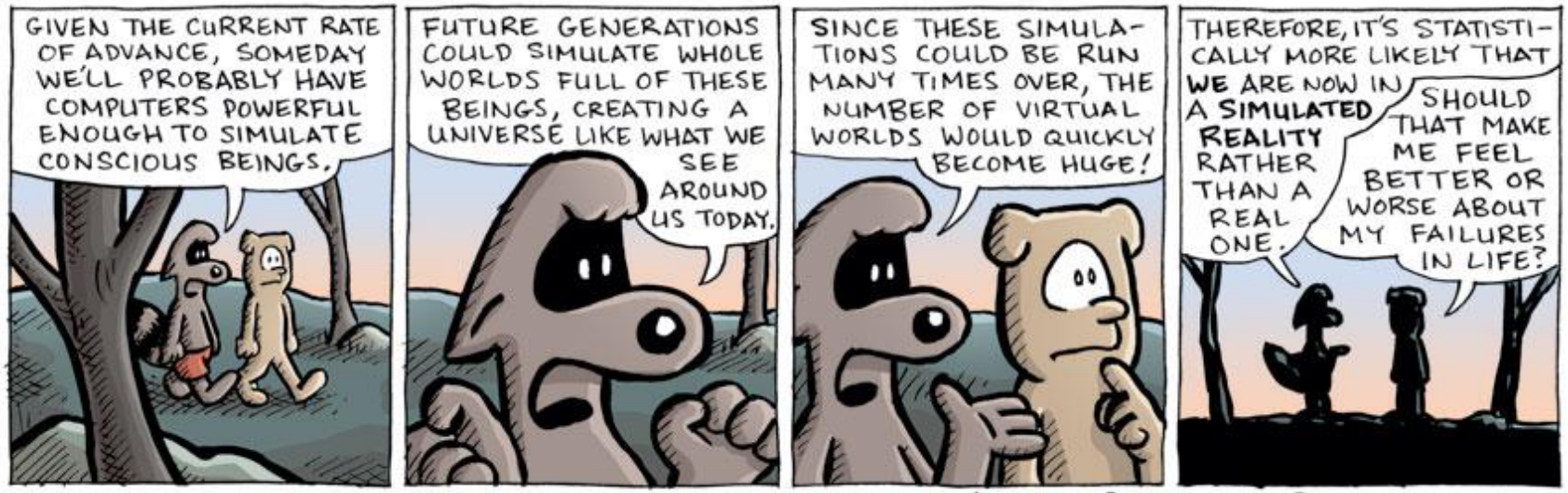
Idee

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

Übersicht


GIVEN THE CURRENT RATE OF ADVANCE, SOMEDAY WE'LL PROBABLY HAVE COMPUTERS POWERFUL ENOUGH TO SIMULATE CONSCIOUS BEINGS.
FUTURE GENERATIONS COULD SIMULATE WHOLE WORLDS FULL OF THESE BEINGS, CREATING A UNIVERSE LIKE WHAT WE SEE AROUND US TODAY.
SINCE THESE SIMULATIONS COULD BE RUN MANY TIMES OVER, THE NUMBER OF VIRTUAL WORLDS WOULD QUICKLY BECOME HUGE!
THEREFORE, IT'S STATISTICALLY MORE LIKELY THAT WE ARE NOW IN A SIMULATED REALITY RATHER THAN A REAL ONE.
SHOULD THAT MAKE ME FEEL BETTER OR WORSE ABOUT MY FAILURES IN LIFE?
calamitiesofnature.com © 2010 Tony Piro