Datenstrukturen (DS)

Sommersemester 2015

Dipl-Inf. Bert Besser
Prof. Dr. Georg Schnitger
Hannes Seiwert, M.Sc.

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Institut für Informatik
AG Theoretische Informatik

Übung 6

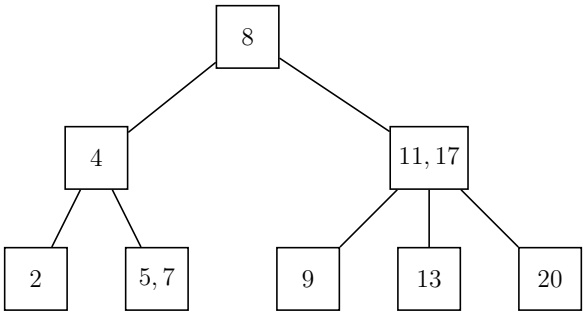
Ausgabe: 30.06.2015
Abgabe: **07.07.2015**

Die Bearbeitungszeit für dieses Übungsblatt beträgt eine Woche.

Aufgabe 6.1. *(a,b)-Bäume* (4+4 Punkte)

(Vorlesungen am 23.06.2015 und 30.06.2015, Abschnitt 5.4 im Skript)

Gegeben sei der folgende $(2,3)$-Baum B.



a) Führen Sie nacheinander die Operationen `insert(1)`, `insert(3)`, `insert(12)`, `insert(14)` auf B aus.

b) Führen Sie nacheinander die Operationen `remove(5)`, `remove(11)`, `remove(4)` auf dem ursprünglichen Baum B aus.

Stellen Sie den Baum nach jeder Operation dar und geben Sie auch Zwischenschritte an.

Aufgabe 6.2. *Hashing* (2+2+4 Punkte)

(Vorlesung am 30.06.2015, Abschnitte 5.5.1 und 5.5.2 im Skript)

Sei $k \in \{0, \ldots, 9\}$ die letzte Ziffer Ihrer Matrikelnummer und $t \in \{1, 2, \ldots, 31\}$ der Kalendertag, an dem Sie Geburtstag haben. Gegeben sei ein leeres Array A mit 11 Zellen sowie die beiden Hashfunktionen

$$f(x) = x \mod 11 \quad \text{und} \quad g(x) = 5 - (x \mod 5)\,.$$

Geben Sie t und k an!

Fügen Sie die Zahlenfolge t, 32, $200{+}7k$, $360{-}t$, 60, 35, $5t{+}k$, 37 in A ein, unter Verwendung der Verfahren

a) Hashing mit Verkettung mit der Hashfunktion f,

b) Hashing mit linearem Austesten, d.h. $h_i(x) = (f(x) + i) \mod 11 \quad (i = 0, 1, \ldots)$,

c) doppeltes Hashing, d.h. $h_i(x) = (f(x) + i \cdot g(x)) \mod 11 \quad (i = 0, 1, \ldots)$.

Es genügt jeweils, das Ergebnis anzugeben.

Bitte wenden!

Aufgabe 6.3. *Quadtrees* (4+4+4+4 Punkte)

Wir betrachten eine Baum-Datenstruktur zur Speicherung von geometrischen Objekten: Quadtrees.

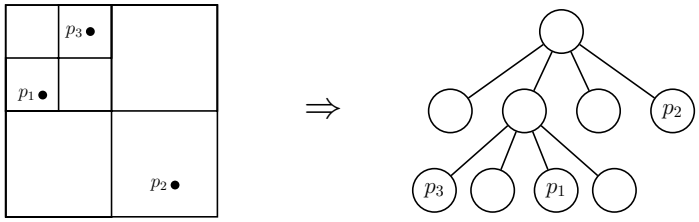
In einer einfachen Anwendung verwaltet ein Quadtree eine Menge N von n Punkten aus dem Einheitsquadrat $[0,1]\times[0,1]$. Dabei soll jeder Punkt in einem Blatt gespeichert werden und jedes Blatt genau 0 oder 1 Punkte speichern. (Innere Knoten speichern also keine Punkte.)

Wir nennen $[0,1]\times[0,1]$ eine *Zelle*. Wenn N die Punktemenge in der Zelle Z ist, dann ist der zugehörige Quadtree $T(N)$ wie folgt rekursiv definiert:

- Wenn in N nur ein einziger Punkt p enthalten ist, so besteht $T(N)$ nur aus der Wurzel und p wird in der Wurzel gespeichert.
- Befindet sich in N kein Punkt, so besteht $T(N)$ ebenfalls nur aus der Wurzel, die diesmal aber keinen Punkt speichert.
- Befinden sich mehrere Punkte in N, so wird Z in vier gleichgroße disjunkte quadratische Zellen $Z_{\mathrm{I}}, Z_{\mathrm{II}}, Z_{\mathrm{III}}, Z_{\mathrm{IV}}$ zerlegt. Die in der Zelle Z_i, $(i \in \{\mathrm{I},\mathrm{II},\mathrm{III},\mathrm{IV}\})$ enthaltene Punktemenge $N_i \subseteq N$ wird im Quadtree $T_i = T(N_i)$ gespeichert. Die vier Bäume T_i werden als Kinder an die (leere) Wurzel von $T(N)$ gehängt. (In einem Quadtree hat also jeder innere Knoten genau vier Kinder.)

II	I
III	IV

Ein kleines Beispiel:



a) Gegeben sei ein Quadtree T.

i) Implementieren Sie die Funktion `insert(p)`, welche den Punkt p in T einfügt, sowie die Funktion `remove(q)`, welche den Punkt q aus T entfernt (falls vorhanden).

Hinweis: Sie können annehmen, dass eine Funktion `kind(v,p)` zur Verfügung steht, die berechnet, in welches der vier Kinder von v der Punkt p eingefügt werden muss.

ii) Implementieren Sie in Pseudocode die Funktion `anzahl(z)`, welche die Anzahl der in Zelle z enthaltenen Punkte ausgibt.

b) i) Beschreiben Sie, wie zwei Punkte p_1 und p_2 konstruiert werden können, sodass der zugehörige Quadtree $T(\{p_1,p_2\})$ möglichst tief wird.

Fazit: Also können Quadtrees im Worstcase viel zu viele Knoten im Vergleich zur Anzahl der gespeicherten Punkte besitzen. Wir verschwenden kostbaren Speicherplatz!

ii) Wie können wir das in Teilaufgabe b)i) aufgezeigte Problem in den Griff bekommen?

Um Platz zu sparen, wollen wir lange "leere" Pfade des Quadtrees durch Direktkanten ersetzen. Dabei darf natürlich kein gespeicherter Punkt verloren gehen. Außerdem soll die Eigenschaft erhalten bleiben, dass jeder innere Knoten genau vier Kinder hat.

Beschreiben Sie, welche Knoten und Kanten Sie durch Direktkanten ersetzen.

Hinweis: Die Anzahl der Knoten im Baum nach Ersetzung muss nicht bestimmt werden. Eine lineare Anzahl $\mathcal{O}(|N|)$ von Knoten ist erreichbar.


FROHES
SCHAFFEN!