Universität des Saarlandes
FR 6.2 Informatik
Prof. Dr. Hans-Peter Lenhof
Dipl. Inform. Andreas Hildebrandt

9. Übung zur Vorlesung Programmierung II, SS 2003

Aufgabe 1: Binäre Suchbäume (10 Punkte)
Die in der Vorlesung vorgestellte Implementierung von binären Suchbäumen ist nicht darauf ausgelegt, viele Elemente mit identischen Schlüsseln abzuspeichern.

- Bestimmen Sie die asymptotische Gesamtlaufzeit einer Sequenz von n `tree_insert`-Aufrufen, mit denen n Elemente mit identischem Schlüssel in einen anfangs leeren binären Suchbaum eingefügt werden.
- Um dieses Problem zu lösen kann man in der Implementierung von `tree_insert` den Fall `x->key == z->key` in Zeile 7 sowie den Fall `z->key == p->key` in Zeile 13 abzufragen und separat zu behandeln. In diesem Fall soll eine der drei folgenden Strategien verwendet werden:[1]

  (a) Jeder Knoten erhält eine `bool` Variable `links`, die mit `true` initialisiert wird und jedes Mal, wenn der Knoten während einem Aufruf von `tree_insert` besucht wird, logisch negiert wird. Hat `links` den Wert `true`, wird in Zeile 7 `x->left_child` gewählt, ansonsten `x->right_child`.

  (b) Jeder Knoten speichert eine Liste mit Elementen mit gleichen Schlüsseln, an die `z` einfach angehängt wird.

  (c) `x` wird zufällig auf `x->left_child` oder `x->right_child` gesetzt.

  Geben Sie für die Strategien (i) und (ii) die asymptotische Gesamtlaufzeit einer wie oben definierten Sequenz von n `tree_insert` Aufrufen an. Bestimmen Sie für Strategie (iii) die worst-case Laufzeit und geben Sie eine informelle Herleitung für die durchschnittliche Laufzeit dieser Strategie an.

Aufgabe 2: Rot-Schwarz-Bäume, Teil 1 (10 Punkte)

(a) Zeichnen Sie den Rot-Schwarz-Baum der sich ergibt, wenn Sie der Reihe nach die Schlüssel $\{41, 38, 31, 12, 19, 8\}$ in einen anfangs leeren Baum einfügen.

(b) Entfernen Sie nun der Reihe nach die Schlüssel $\{8, 12, 19, 31, 38, 41\}$ aus dem Baum und zeichnen Sie nach jedem Schritt den sich ergebenden Rot-Schwarz-Baum.

[1] Die Beschreibung der Strategien bezieht sich auf Zeile 7. Für die Anwendung in Zeile 13 muß lediglich `x->key` durch `p->key` ersetzt werden.

Aufgabe 3: Erwartungswerte (10 Punkte)
Angenommen Sie nehmen an einem Gewinnspiel teil, das Sie beliebig oft wiederholen dürfen. Dabei beschreibt die Zufallsvariable X die Anzahl an Versuchen, die Sie durchführen müssen, bevor Sie den Gewinn kassieren können, d.h. $X = n$ bedeutet, dass die ersten $n-1$ Versuche erfolglos, der n-te hingegen erfolgreich waren. Die Wahrscheinlichkeit $p(\{X \geq i\})$ dafür, daß Sie mindestens i Versuche benötigen, ist Ihnen im Voraus bekannt und es gilt

$$p(\{X \geq i\}) = \begin{cases} 0 & i = 0 \\ \frac{1}{i} \sum_{k=1}^{i} \frac{1}{k^2} & \text{sonst} \end{cases}$$

Bestimmen Sie den Erwartungswert $E[X]$ der Anzahl benötigter Versuche.

Aufgabe 4: 2–4 Bäume (10 Punkte)
In einem 2–4 Baum hat jeder Knoten mindestens 2 und höchstens 4 Kinder und alle Pfade von der Wurzel zu einem beliebigen Blatt haben die gleiche Länge h. h ist daher die *Höhe des Baums*.

(a) Bestimmen Sie für einen 2–4 Baum mit n Blättern eine möglichst große Funktion $f(n)$ und eine möglichst kleine Funktion $g(n)$, so dass gilt: $f(n) \leq h \leq g(n)$.

(b) 2–4 Bäume können als Suchbäume verwendet werden. Eine aufsteigend sortierte Folge $x_1, \ldots, x_n$ wird dabei folgendermaßen abgespeichert: im i-ten *Blatt* von links speichert man die Zahl x_i. In jedem *inneren Knoten* k wird die Zahl abgelegt, die im rechtesten Blatt des Unterbaumes mit Wurzel k gespeichert ist.

(i) Beschreiben Sie die Suche in einem solchen 2–4 Baum und bestimmen Sie deren Laufzeit.

(ii) Beschreiben Sie das Einfügen und Löschen eines Elementes in einen solchen Suchbaum. Dabei dürfen Sie die 2–4 Eigenschaft des Baumes zerstören.

(iii) Beschreiben Sie drei Operationen *Verschmelzen*, *Aufspalten* und *Stehlen eines Kindes*, mit denen Sie die 2–4 Eigenschaft des Baumes nach einer Einfüge- oder Löschoperation wiederherstellen können.

Zusatzaufgabe 5: Binäre Suchbäume, Implementierung (10 Bonuspunkte)
Implementieren Sie eine Template-Klasse `BinSuchbaum<T>`. Diese sollte die in der Vorlesung vorgestellten Operationen `tree_search`, `tree_insert`, `tree_delete`, `tree_successor`, `tree_predecessor`, `tree_minimum` und `tree_maximum` enthalten.

Abgabe: 27.06.2003