Lehrstuhl für Informatik 3
Rheinisch-Westfälische Technische Hochschule
Aachen
Univ.-Prof. Dr.-Ing. M. Nagl

Aachen, 23. 06. 2006

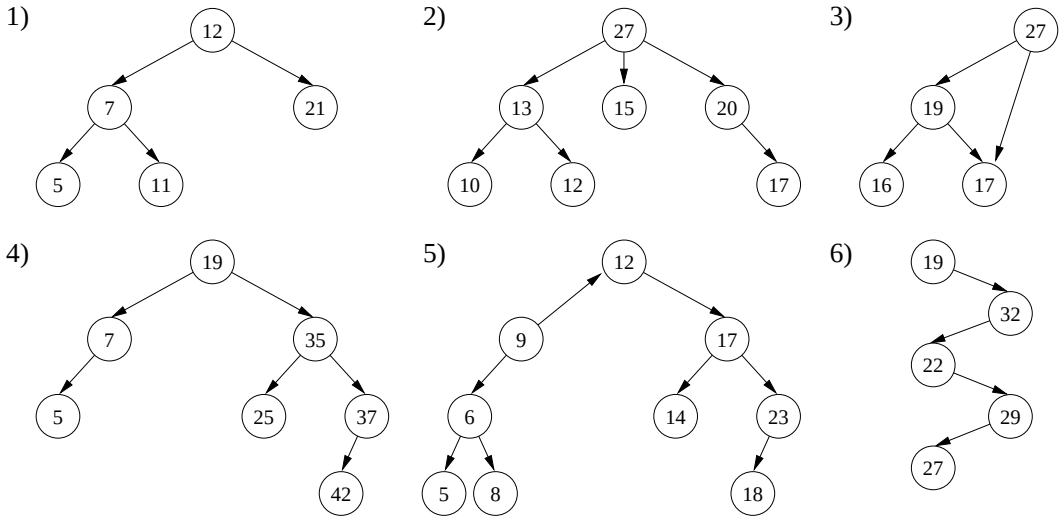
Nagl, Becker, Ranger, Wörzberger

Übungen zur Vorlesung „Grundgebiete der Informatik 2: Algorithmen und Programmiertechniken“

— Blatt 10 —

23. Aufgabe *Graphen und Bäume*: *(7 Punkte)*

(a) Welche der folgenden Graphen sind Bäume, Binärbäume, binäre Suchbäume? Geben Sie gegebenenfalls jeweils den maximalen Auslaufgrad an und bestimmen Sie die Höhe der Bäume. *(3 Punkte)*



(b) Wieviele Blätter und Knoten hat ein binärer Baum der Höhe n maximal? Wie hoch muß ein binärer Suchbaum zur Speicherung von m Werten minimal sein? *(4 Punkte)*

24. Aufgabe *Operationen auf sortierten Bäumen*: *(13 Punkte)*

(a) Fügen Sie die folgenden Zahlen in der gegebenen Reihenfolge in einen binären Suchbaum ein (ausgehend von einem leeren Baum). Skizzieren Sie den resultierenden Baum und bestimmen Sie seine Höhe.

15, 12, 19, 8, 14, 13, 16, 22, 17 *(2 Punkte)*

(b) Löschen Sie aus dem in (a) aufgebauten Baum nacheinander die Werte 13, 14 und 19 und skizzieren Sie den Baum nach jeder Löschoperation. *(4 Punkte)*

(c) Skizzieren Sie den binären Suchbaum, der entsteht, wenn Zahlen aus (a) in sortierter Reihenfolge (8, 12, 13, 14, 15, 16, 17, 19, 22) einfügt werden. Welcher Baum eignet sich besser zum Suchen? Warum? *(2 Punkte)*

(d) Ein Binärbaum heißt balanciert, wenn für jeden Knoten gilt, daß die Höhe des linken Teilbaums sich von der Höhe des rechten Teilbaums höchstens um den Wert eins unterscheidet. Schreiben Sie eine Funktion `istBalanciert`, die rekursiv ermittelt, ob ein gegebener Binärbaum balanciert ist oder nicht. Der Zugriff auf den Binärbaum erfolgt über das aus Aufgabe 22 bekannte Interface. *(5 Punkte)*

Funktionaler Modul: `istBalanciert.h`

```
#include <BinBaum.h>

bool istBalanciert(BinBaumKnotenT const & baum);
```

Datentypmodul: `BinBaum.h`

```
/* Datentypmodul BinBaum --- Binaerer Baum
   Baeume dieses Datentyps speichern vorzeichenbehaftete ganze Zahlen.  Die
   verkapselte Entwurfsentscheidung ist die interne Speicherung der Knoten.
   Die Namen befolgen die Baum-Metapher so weit wie moeglich.
   Die Kommentare sind aus Platzgruenden sparsamer.
*/
#ifndef __INC_BinBaum_h__
#define __INC_BinBaum_h__

struct BinBaumKnotenT;          // Der opake Eintragstyp.

/* Erzeugungs- und Loeschoperationen */
BinBaumKnotenT & neuerKnoten();
void zerstoereKnoten(BinBaumKnotenT & opfer);

/* Zugriff auf den gespeicherten Wert */
int  leseWert (BinBaumKnotenT const & knoten);
void setzeWert(BinBaumKnotenT       & knoten, int neuerWert);

/* Navigation zu den Nachbarn.  Liefern NULL und geben eine Fehlermeldung
   aus, falls linkerAstExistiert (etc.) false liefern. */
BinBaumKnotenT * linkerAst (BinBaumKnotenT const & stamm);
BinBaumKnotenT * rechterAst(BinBaumKnotenT const & stamm);
BinBaumKnotenT * stamm     (BinBaumKnotenT const & ast  );

/* Abfragen zur Struktur */
bool linkerAstExistiert (BinBaumKnotenT const & knoten);
bool rechterAstExistiert(BinBaumKnotenT const & knoten);
bool stammExistiert     (BinBaumKnotenT const & knoten);

/* Struktur des Baumes aendern: Aeste hinzufuegen oder absaegen.  Der Stamm
   bleibt.  */
/* Neue Aeste anbringen.  Eventuell vorhandene Aeste werden dabei abgesaegt,
   aber nicht zerstoert. */
void setzeLinkenAst (BinBaumKnotenT & stammKnoten,
                     BinBaumKnotenT & neuerLinkerAstKnoten);
void setzeRechtenAst(BinBaumKnotenT & stammKnoten,
                     BinBaumKnotenT & neuerRechterAstKnoten);

/* Die Aeste werden abgesaegt, aber nicht zerstoert. */
void loescheLinkenAst (BinBaumKnotenT & stammKnoten);
void loescheRechtenAst(BinBaumKnotenT & stammKnoten);
#endif
```

Abgabe: 30. 06. 2006 vor der Vorlesung.