TECHNISCHE UNIVERSITÄT DORTMUND

Wintersemester 2007/08

Einführung in die Informatik für Naturwissenschaftler und Ingenieure

(alias **Einführung in die Programmierung)**

(Vorlesung)

Prof. Dr. Günter Rudolph
Fakultät für Informatik
Lehrstuhl für Algorithm Engineering

Inhalt

- Definition ☑
- ADT Keller ☑
- ADT Schlange ☑
- ADT Liste ☑
- ADT Binärbaum ☑
  - Exkurs: Einfache Dateibehandlung
  - Exkurs: Strings (C++)

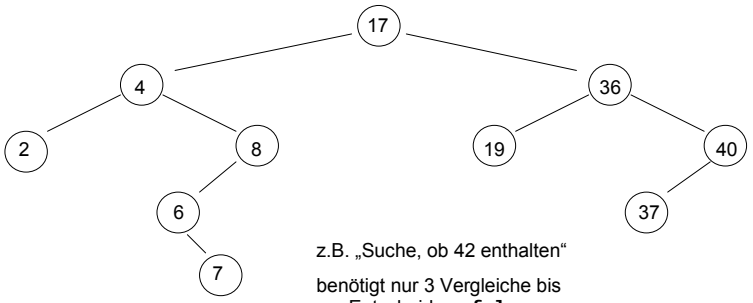
… endlich! ☺

ADT Binäre Bäume

Beispiel:

Zahlenfolge 17, 4, 36, 2, 8, 19, 40, 6, 7, 37

kleiner	: nach links
größer	: nach rechts



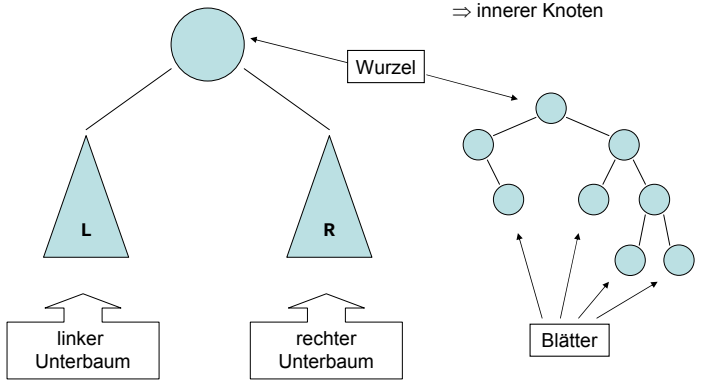
z.B. „Suche, ob 42 enthalten“

benötigt nur 3 Vergleiche bis zur Entscheidung **`false`**

ADT Binäre Bäume: Terminologie

keine Wurzel und kein Blatt

⇒ innerer Knoten



ADT Binäre Bäume: Datenstruktur

```
struct BinTree {
  T data;                  // Nutzdaten
  BinTree *lTree, *rTree;  // linker und rechter Unterbaum
};
```

Falls ein Unterbaum nicht existiert, dann zeigt der Zeiger auf 0.

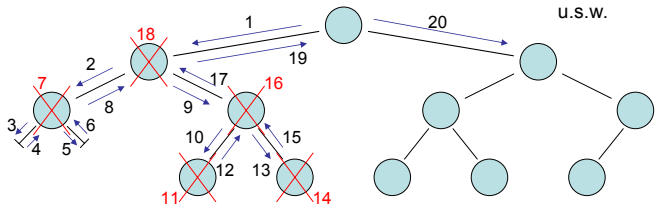
```
bool IsElement(int key, BinTree *tree) {
  if (tree == 0) return false;
  if (tree->data == key) return true;
  if (tree->data < key) return IsElement(key, tree->rTree);
  return IsElement(key, tree->lTree);
}
```


ADT Binäre Bäume: Einfügen

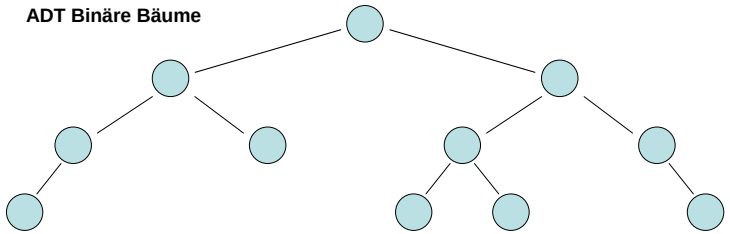
```
BinTree *Insert(int key, BinTree *tree) {
  if (tree == 0) {
    BinTree *b = new BinTree;
    b->data = key;
    b->lTree = b->rTree = 0;
    return b;
  }
  else {
    if (tree->data < key)
      tree->rTree = Insert(key, tree->rTree);
    else if (tree->data > key)
      tree->lTree = Insert(key, tree->lTree);
    return tree;
  }
}
```


ADT Binäre Bäume: Aufräumen

```
void Clear(BinTree *tree) {
  if (tree == 0) return;   // Rekursionsabbruch
  Clear(tree->lTree);      // linken Unterbaum löschen
  Clear(tree->rTree);      // rechten Unterbaum löschen
  delete tree;             // aktuellen Knoten löschen
}
```



ADT Binäre Bäume



Höhe := Länge des längsten Pfades von der Wurzel zu einem Blatt.

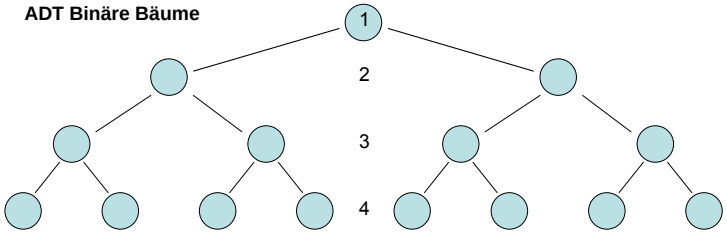
Höhe(leerer Baum) = 0

Höhe(nicht leerer Baum) = 1 + max { Höhe(linker U-Baum), Höhe(rechter U-Baum) }

Anmerkung: rekursive Definition!

(U-Baum = Unterbaum)

ADT Binäre Bäume



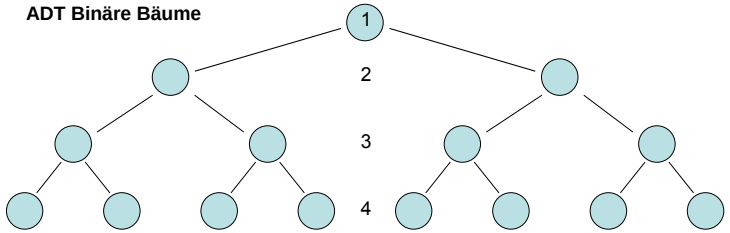
Auf Ebene k können jeweils zwischen 1 und 2^{k-1} Elemente gespeichert werden.

⇒ In einem Baum der Höhe h können also zwischen h und

$$\sum_{k=1}^{h} 2^{k-1} = 2^h - 1$$

Elemente gespeichert werden!

ADT Binäre Bäume



- Ein **vollständiger Baum** der Höhe h besitzt $2^h - 1$ Knoten.
  Man braucht maximal h Vergleiche, um Element (ggf. nicht) zu finden.
  Bei $n = 2^h - 1$ Elementen braucht man $\log_2(n) < h$ Vergleiche!
- Ein **degenerierter Baum** der Höhe h besitzt h Knoten (= lineare Liste).
  Man braucht maximal h Vergleiche, um Element (ggf. nicht) zu finden.
  Bei n = h braucht man also n Vergleiche!