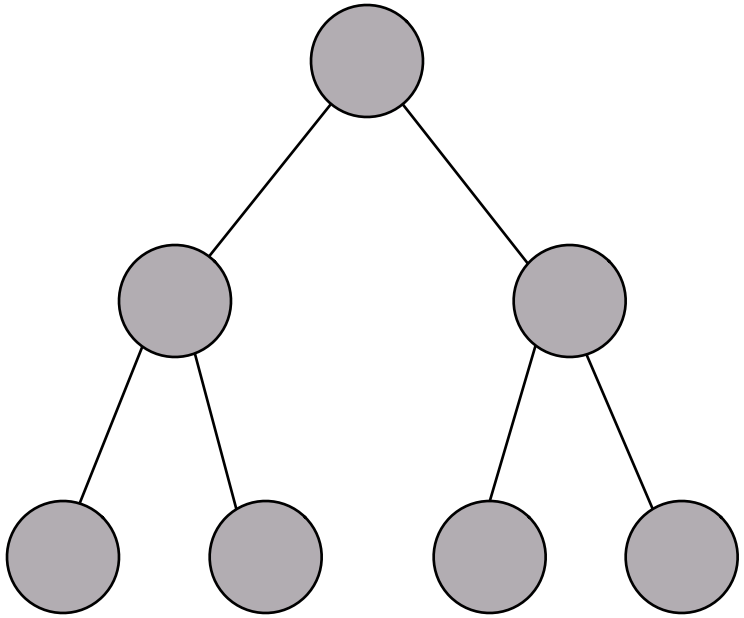
educational engineering lab
Department for Information Technology
University of Zurich

Baumstrukturen

Beispiel: Binärer Baum



Definitionen (1)

Definitionen (1)

Ein Baum (tree) ist entweder leer oder er besteht aus genau einer Wurzel (root) mit einer beliebigen Anzahl von Teilbäumen (subtrees). Jeder Teilbaum ist wieder ein Baum.

Die Wurzel des Baumes und die Wurzeln seiner Teilbäume heissen auch Knoten (nodes) des Baumes.

Knoten ohne Teilbäume sind die Blätter (leaves) des Baumes.

Definitionen (2)

Benachbarte Knoten eines Baumes sind durch Kanten (edges) verbunden.

Eine Folge von Knoten die durch Kanten miteinander verbunden sind heisst Pfad (path).

Zwischen zwei beliebigen Knoten eines Baumes gibt es genau einen Pfad.

Geordneter Baum

In einem geordneten Baum (ordered tree) ist die Reihenfolge der Teilbäume jedes Knotens relevant.

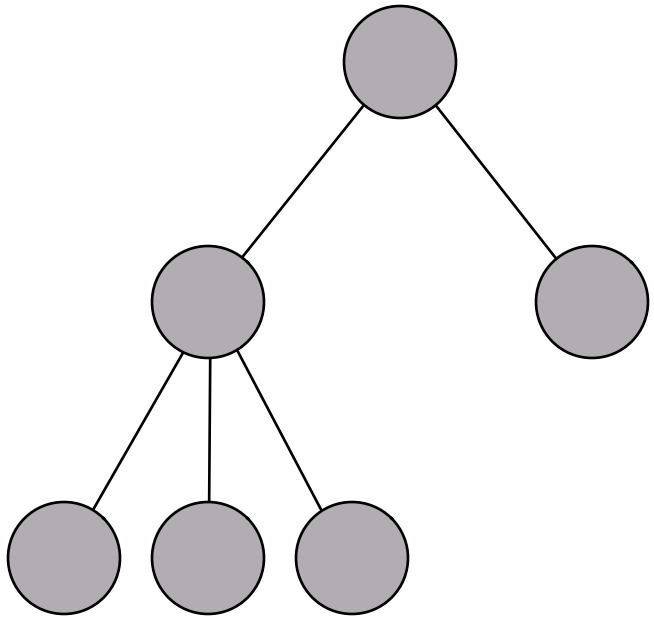
Bäume, die sich nur durch die Reihenfolge ihrer Teilbäume unterscheiden sind zueinander isomorph.

Ordnung eines Baumes

Die maximale Anzahl von Teilbäumen eines Knotens entspricht der Ordnung eines Baumes.

Beispiel:

Baum der Ordnung 3



Binärer Baum

Ein binärer Baum (binary tree) ist ein Baum in dem jeder Knoten maximal zwei Teilbäume hat.

Binäre Bäume haben die Ordnung 2.

Ist der binäre Baum geordnet, so sprechen wir von linken und rechten Teilbäumen.

Perfekter Binärer Baum

Ein binärer Baum heisst perfekt oder voll wenn der linke und rechte Teilbaum jedes Knotens aus gleich vielen Knoten besteht.

In einem perfekten binären Baum ist jeder Knoten entweder ein Blatt oder er hat genau zwei nicht leere Teilbäume.

Höhe eines Baumes

Die Höhe (height) eines Baumes ist die maximale Pfadlänge von der Wurzel zu den Blättern des Baumes.

Für einen perfekten binären Baum mit n Knoten und der Höhe h gilt

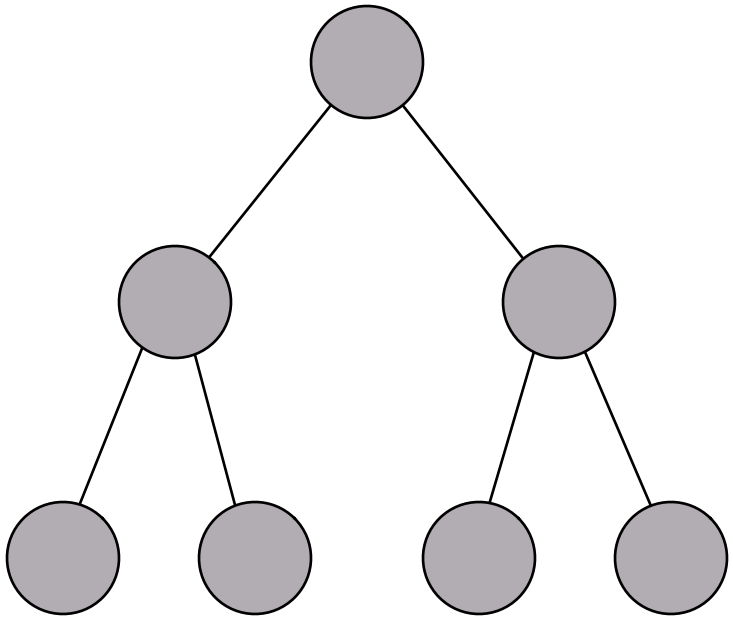
$$n = 2^{h+1} - 1$$

und

$$h = ld\ (n+1) - 1$$

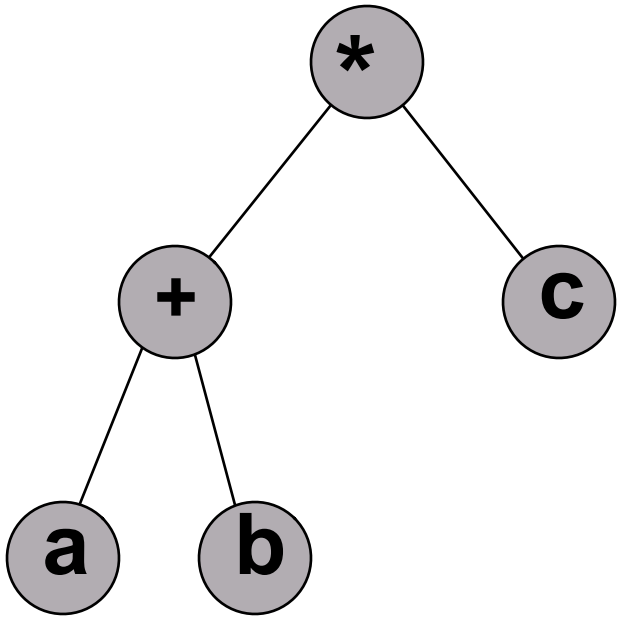
Ein perfekter binärer Baum der Höhe h hat genau 2^h Blätter.

Beispiel: Perfekter Binärer Baum



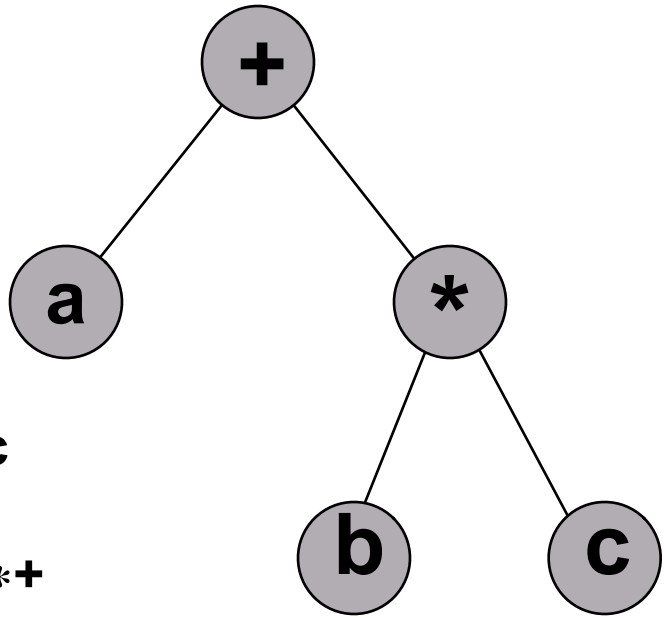
n = 7
h = 2

Beispiel: Arithmetischer Ausdruck (a+b)∗c



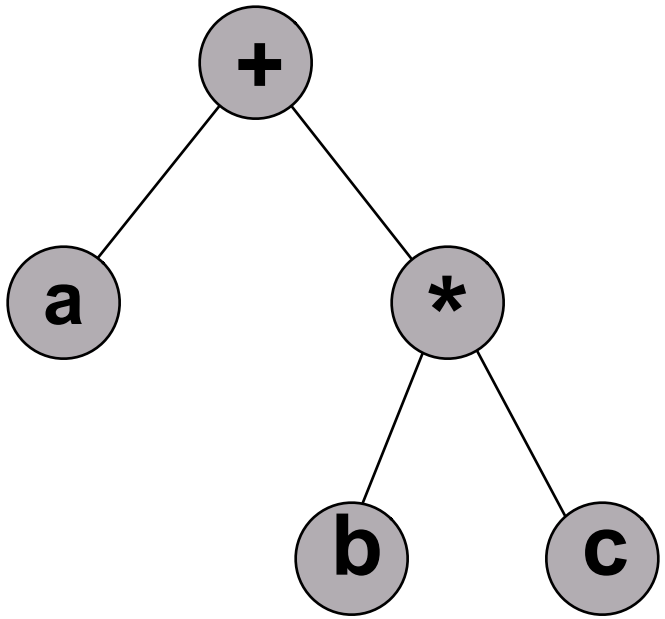
preorder: ∗+abc
inorder: (a+b)∗c
postorder: ab+c∗

Beispiel: Arithmetischer Ausdruck a+b∗c



preorder: +a*bc
inorder: a+b∗c
postorder: abc∗+

Beispiel: Arithmetischer Ausdruck a+b*c

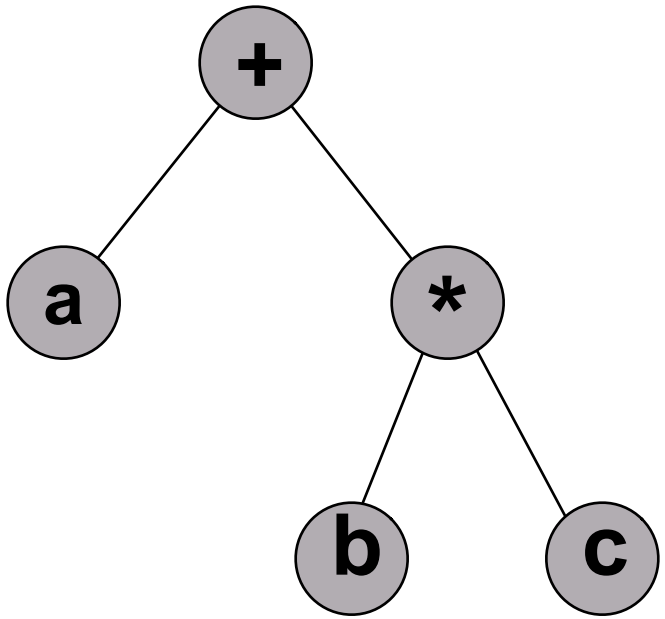


Preorder Traversieren (rekursiv)

```
preOrder() {
        if (!isEmpty()) {
                visit(info);
                left.preOrder();
                right.preOrder();
        }
}
```

preorder: +a*bc

Beispiel: Arithmetischer Ausdruck a+b∗c

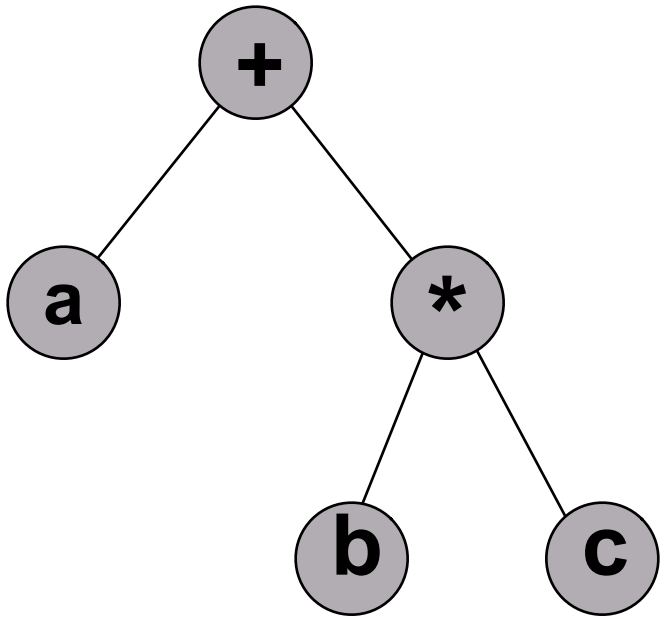


inorder: a+b∗c

Inorder Traversieren (rekursiv)

```
inOrder() {
        if (!isEmpty()) {
                left.inOrder();
                visit(info);
                right.inOrder();
        }
}
```

Beispiel: Arithmetischer Ausdruck a+b∗c

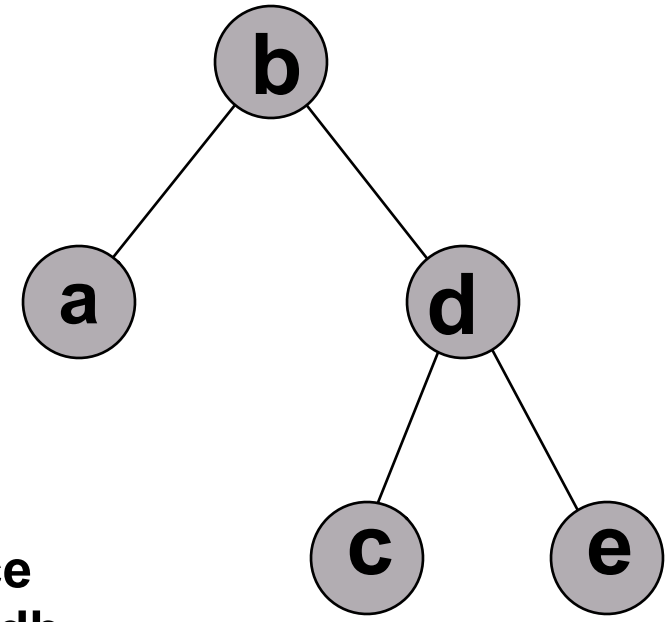


postorder: abc∗+

Postorder Traversieren (rekursiv)

```
postOrder() {
        if (!isEmpty()) {
                left.postOrder();
                right.postOrder();
                visit(info);
        }
}
```

Binärer Sortierbaum



inorder: abcde
preorder: badce
postorder: acedb
levelorder: badce

Binärer Sortierbaum

Ein binärer Sortierbaum ist ein geordneter binärer attributierter Baum für den beim inorder-Traversieren alle Attribute gemäss ihrer Ordnungsrelation durchlaufen werden.

Bezeichnet x.key das Attribut eines Knotens x und x.left den linken und x.right den rechten Teilbaum des Knotens x so gilt bei steigender Sortiertheit eines Sortierbaumes T:

$$\text{All } x: x \in T:$$

$$(\text{All } y: y \in x.\text{left}: y.\text{key} \leq x.\text{key}) \wedge (\text{All } y: y \in x.\text{right}: y.\text{key} \geq x.\text{key})$$

Achtung: Es müssen auch alle Teilbäume Sortierbäume sein!