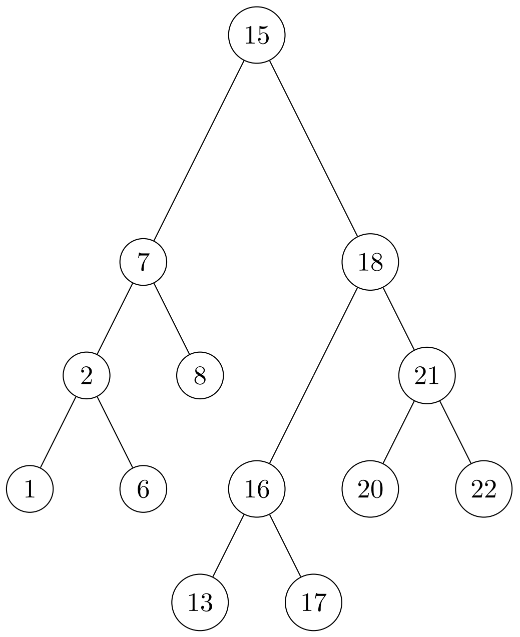
Algorithmen und Datenstrukturen Sheet 5 **Maximilian von Sternberg**



1.

addRoot(15)
addLeft(15, 7)
addRight(15, 18)
addLeft(7, 2)
addRight(7, 8)
addLeft(2, 1)
addRigt(2, 6)
addLeft(18, 16)
addRight(18, 21)
addLeft(16, 13)
addRight(16, 17)
addLeft(21, 20)
addRight(21, 22)

2. (a) Wenn alle Blatt-Knoten diesselbe Tiefe besitzen

(b)

$$\begin{aligned} H &= \log_2(N+1) - 1 \quad \mid +1 \\ H + 1 &= \log_2(N+1) \quad \mid 2^{()} \\ 2^{H+1} &= N + 1 \quad \mid -1 \\ 2^{H+1} - 1 &= N \quad \square \end{aligned}$$

(c)

$$\begin{aligned} n_E &\leq 2^h \quad \mid \log_2 \\ log_2 n_E &\leq h \quad \square \end{aligned}$$

Listing 1: Ein Beispiel

```java
package week05;

import java.util.LinkedList;
import java.util.Queue;

public class BreadthFirstIterator {

    private final Queue<Node> queue;

    public static class Node {
        int value;
        Node left;
        Node right;

        Node(int x) {
            value = x;
        }
    }

    public static void main(String[] args) {
        Node root = new Node(13);
        root.left = new Node(10);
        root.right = new Node(19);
        root.left.left = new Node(3);
        root.left.right = new Node(7);
        root.left.left.left = new Node(1);
        root.left.left.right = new Node(4);
        root.right.left = new Node(15);
        root.right.left.right = new Node(17);
        root.right.right = new Node(22);

        BreadthFirstIterator iter = new BreadthFirstIterator(root);

        System.out.println("hasNext: " + iter.hasNext());
        for (int i = 0; i < 10; i++)
            System.out.println("next: " + iter.next());
        System.out.println("hasNext: " + iter.hasNext());
    }

    public BreadthFirstIterator(Node root) {
        queue = new LinkedList<>();
        queue.add(root);
    }

    /**
     * @return whether we have a next node
     */
    public boolean hasNext() {
        return !queue.isEmpty();
    }

    /**
     * @return the next node
     */
    public int next() {
        Node current = queue.remove();
        System.out.println(current.value);
        if(current.left != null){
            queue.add(current.left);
        }
        if(current.right != null){
            queue.add(current.right);
        }
        return current.value;
    }
}
```