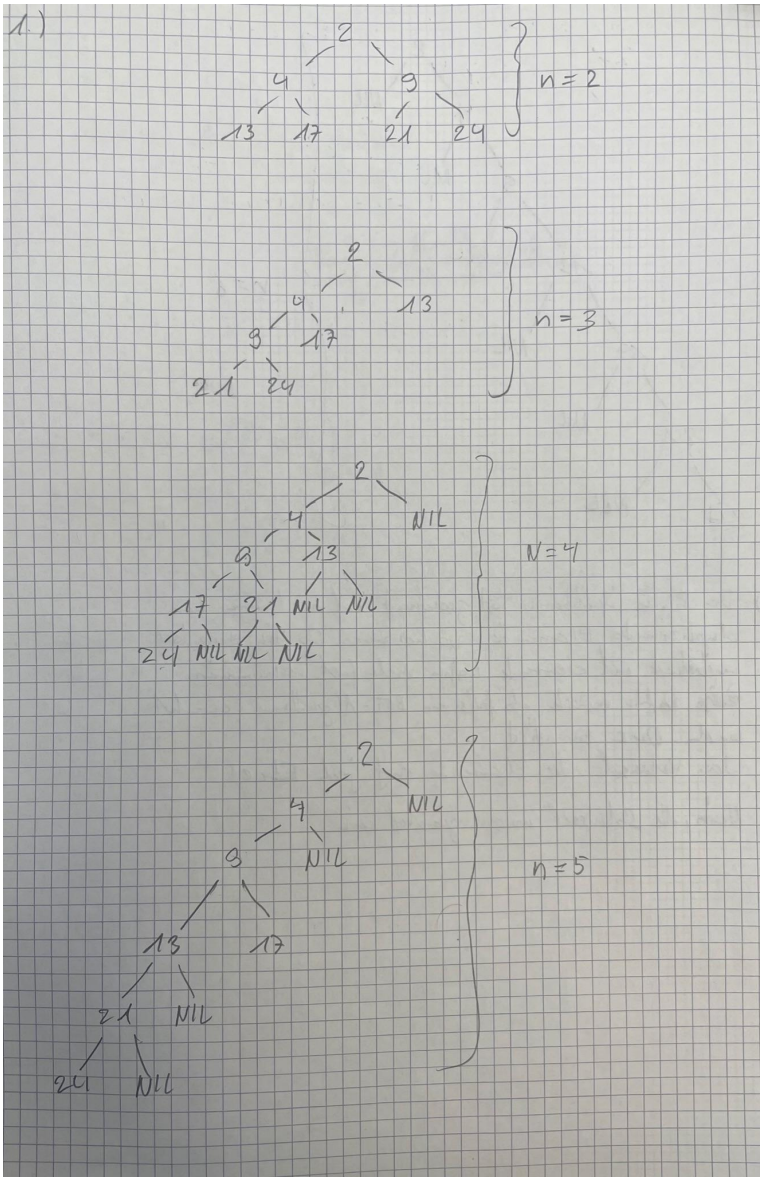
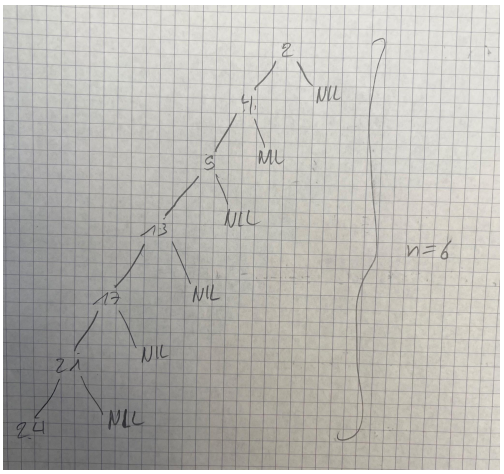
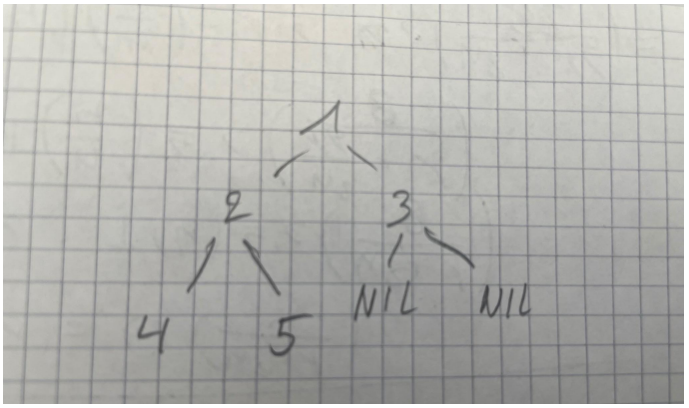
Datenstrukturen und Algorithmen, Serie 7





2.) Der binäre Suchbaum ist geordnet und es können keine duplizierten Elemente vorkommen. Im gegensatz dazu ist der Min-Heap ungeordnet und Elemente könne dupliziert vorkommen. Um einen Min-Heap in einen Binary Tree umzuwandeln, müssen auf jeder Ebene die Elemente vertauscht werden. Das ist in O(n) möglich, wobei n die Tiefe des Baumes ist. Es ist nicht möglich eine beliebige Schlüsselfolge in O(n) in einen Binary Tree umzuwandeln. Das geht nur in O(n), falls die Schlüsselfolge bereits geordnet ist.

3.) Betrachte A = [1,2,3,4,5], dann stimmt die Aussage nicht, weil 5 >= 3 ist.

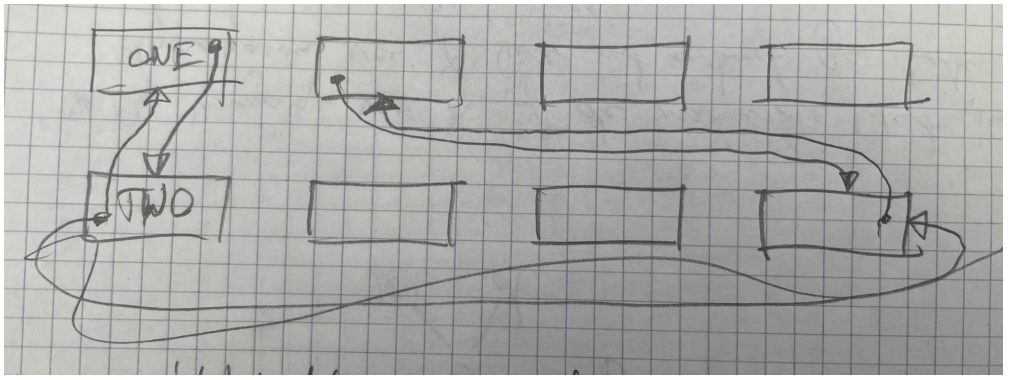


4.) DEF *INSERT(*Node, TreeNode)

```
    IF TreeNode.hasChildren?
        INSERT(Node, TreeNode.children)
    ELSE
        TreeNode.addChild(Node)
    END
END
```

5.) Der Vorgänger von X ist der maximale Wert im linken Unterbaum von X. Hat dieser ein rechtes Kind, dann existiert aufgrund der Ordnung des Binary Tree ein Wert im linken Unterbaum, welcher grösser als der Vorgänger ist. Für den Nachfolger kann analog argumentiert werden, nur dass hier ein kleineres Element existieren würde.

Repetition

1.) 

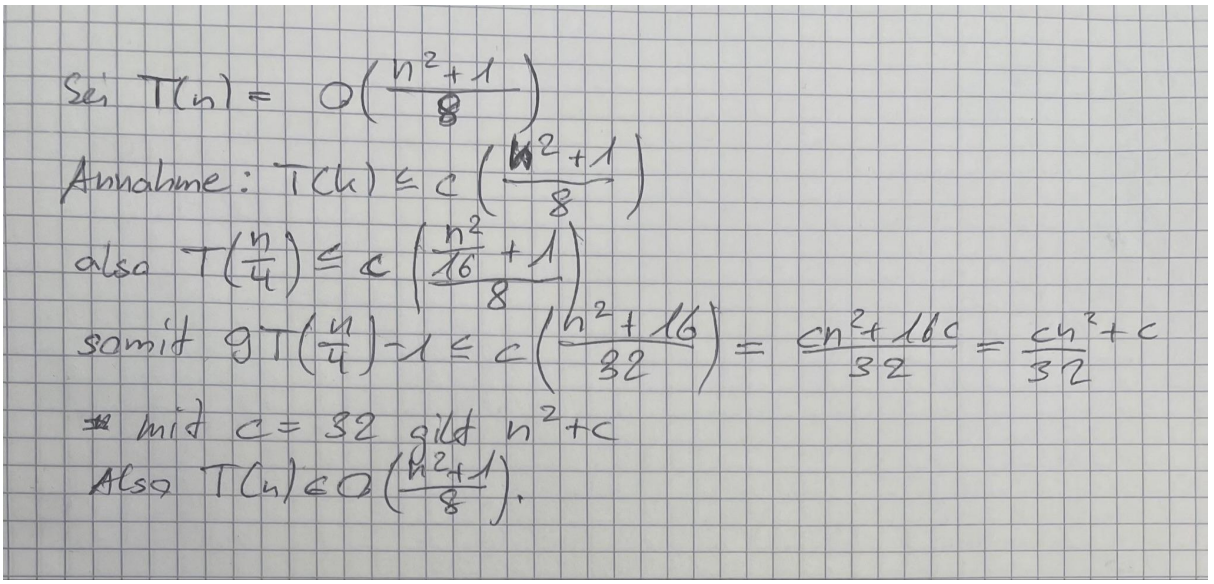
one.right.left = two.left;
two.left.right = one.right;

one.right = two;
two.left = one;

2.) a und f wachsen polynomial. b und h wachsen exponentiell. c und d wachsen faktoriell. e wächst logarithmisch. g wächst linear. i wächst linear-logarithmisch (?) und j wächst konstant.

3.) Ich denke $O(2^{log_3(n)})$.

4.)



5.) Habe ich noch nicht.