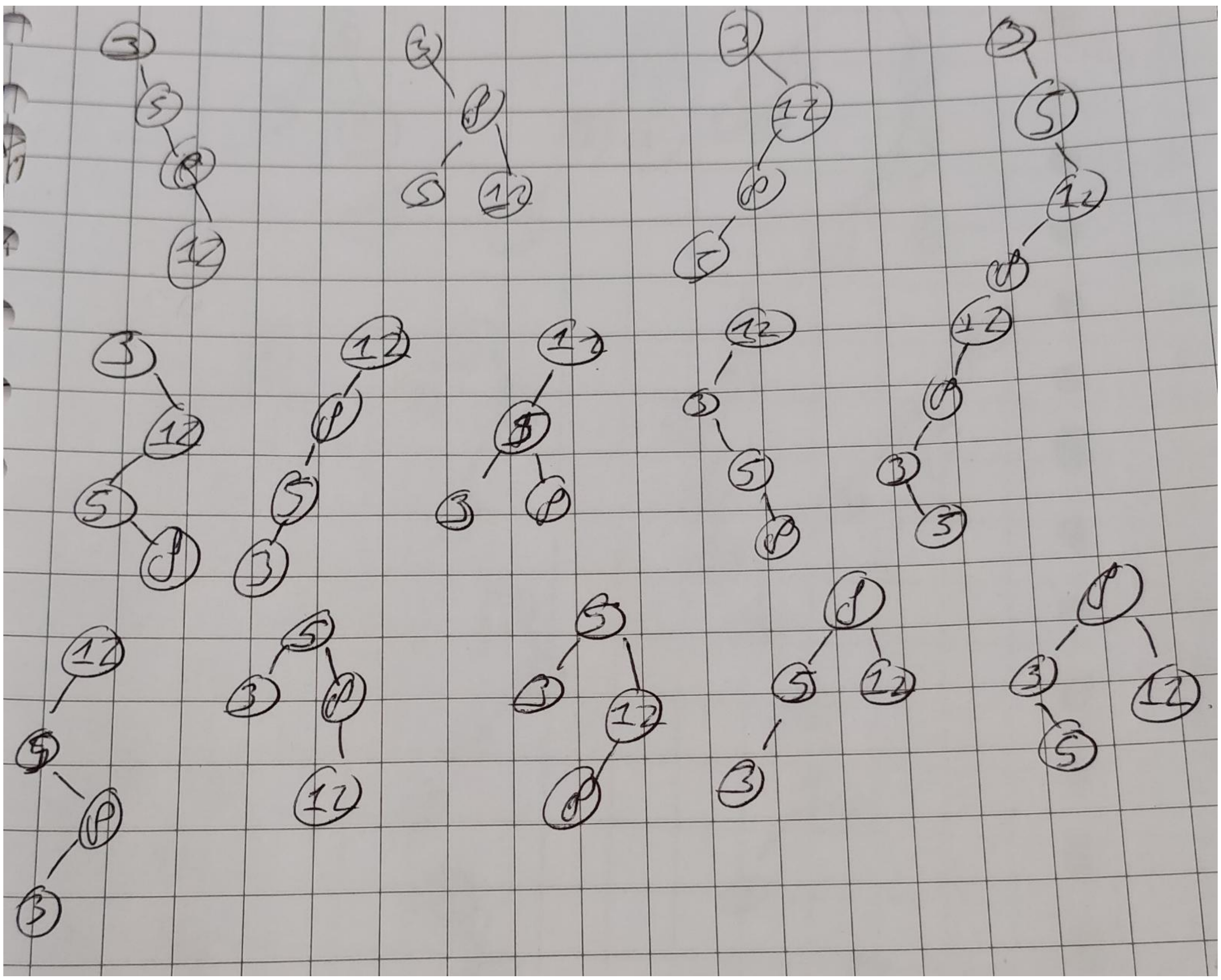
Problem session 7

Dibran Dokter 1047390

November 22, 2019

7

7.1



There are 14 trees, 5 for the lowest value in the root and the highest value node in the root and 2 for each node in between in the root. So we have 5 trees for 3, 5 trees for 12 and only 2 trees for both 5 and 8.

7.2

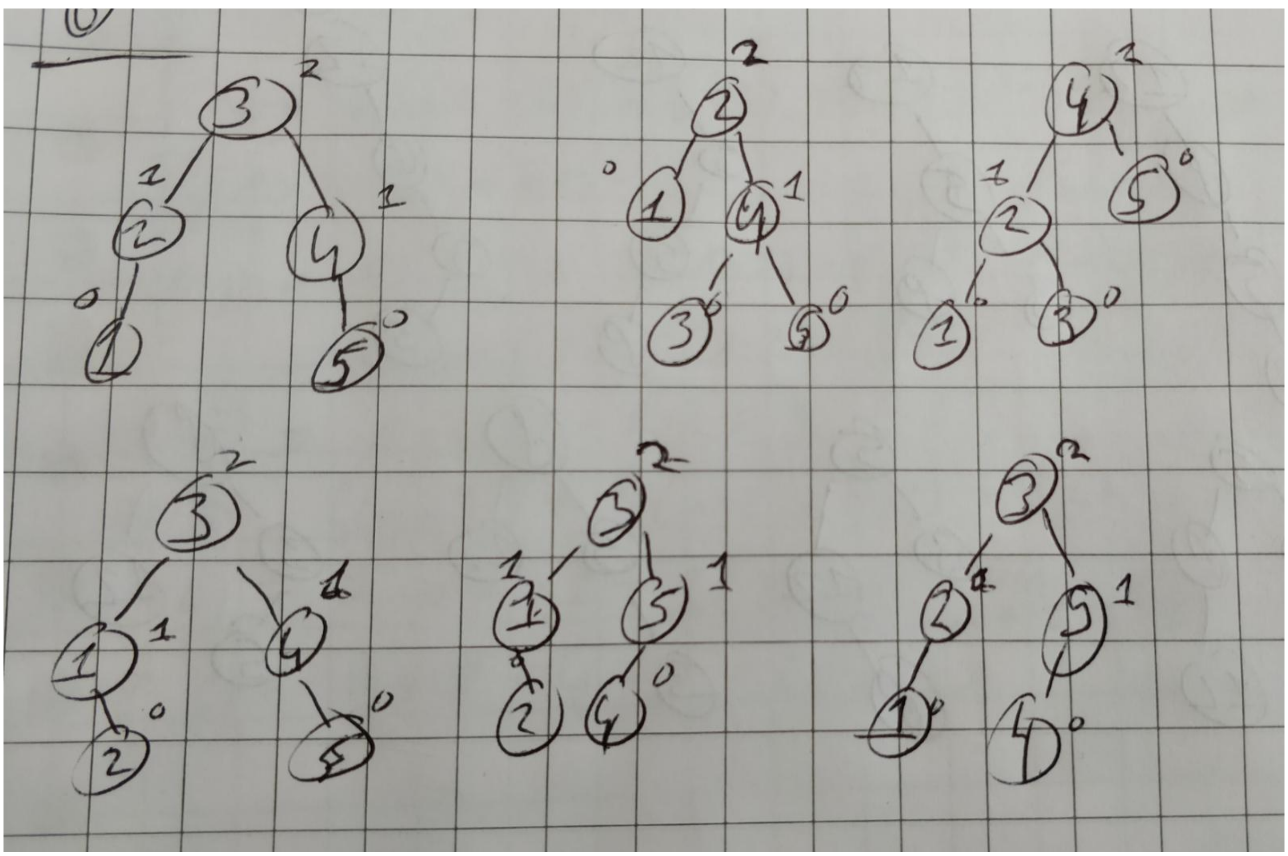
We can use the following for $f(n) : f(n) = f(n-1) + f(n-2) \cdot f(1) + f(n-3) * f(2)$ .... We do this until f(n-1) reaches f(0).

When we do this for $f(6)$ then we get $f(6) = f(5)+f(4)\cdot f(1)+f(3)\cdot f(2)+f(2)\cdot f(3)+f(1)\cdot f(4)+f(0)\cdot f(5)$.

f(0) = 1
f(1) = 1
f(2) = 2
f(3) = 5
f(4) = 14
f(5) = $(f(4) + f(3) \cdot f(1) + f(2) \cdot f(2) + f(1) \cdot f(3) + f(0) \cdot f(4) = 42$
f(6) = $(f(5) + f(4) \cdot f(1) + f(3) \cdot f(2) + f(2) \cdot f(3) + f(1) \cdot f(4)) + f(0) \cdot f(5) = 132$.

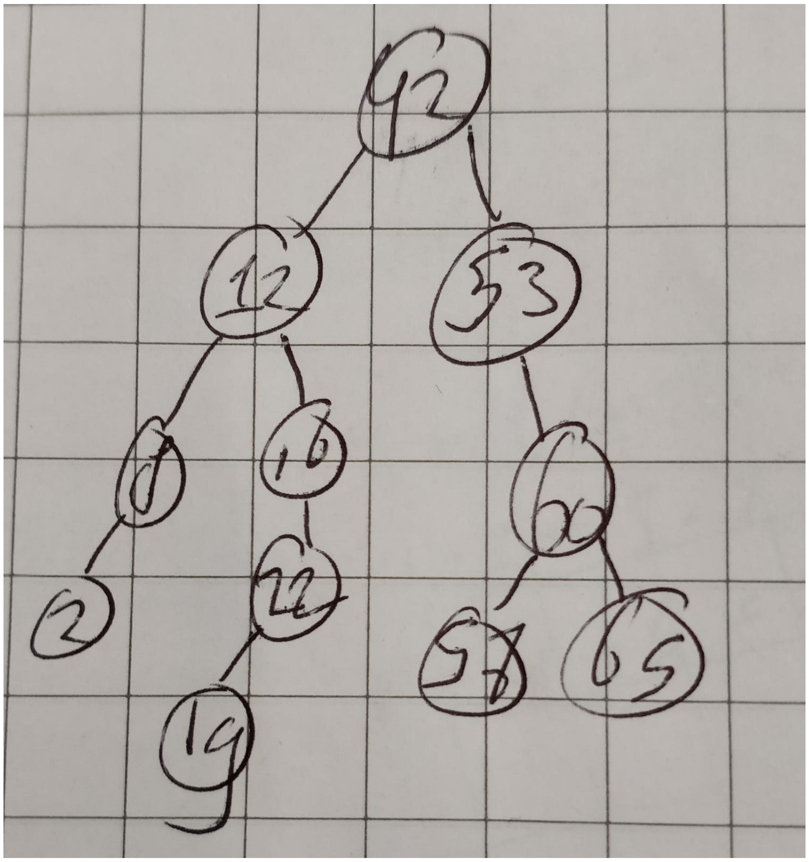
7.3

There are 6 AVL trees that can be made with the nodes $\{1, 2, 3, 4, 5\}$. We can check that they are AVL trees by checking that the height of the children of the root do not differ by more than 1.

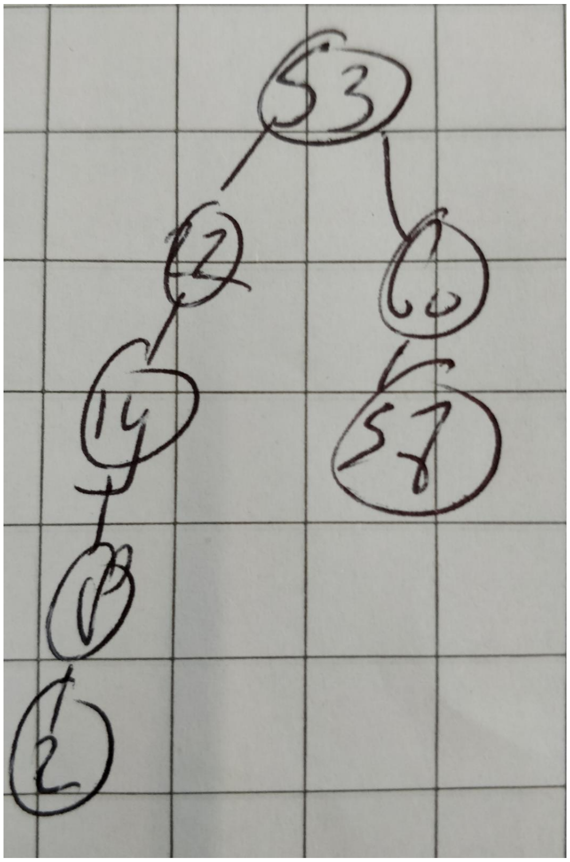


7.4

1)



2,3,4,5)



7.5

To do this we need to switch the children of each node's children. We do this by going from the root through all the child nodes.

This gives to following algorithm:

```
root = tree.root
recursiveSwap(root)

recursiveSwap(x)
{
    while x != NIL //We not have a leaf
    {
        swap(x.left, x.right)
        recursiveSwap(x.left)
        recursiveSwap(x.right)
    }
}
```

7.6

To prove this we need to realize there is only one parent pointer that is NIL. Namely the root node.
For all the other nodes we either have 2 NIL values for the children or only one. But when they have only one NIL then the other node will have 2 NIL at some point in the tree because they will become a leaf node.

Thus we can have 1 for the root node + 2 for every leaf node and + 1 for every node with only one NIL. This added up is n+2. Since we have 1 for the root node and n+1 for the remaining nodes.