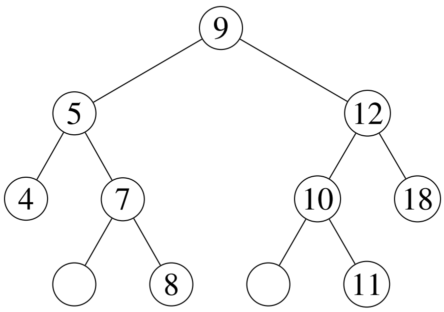
Algorithms and Datastructures

Assignment 11

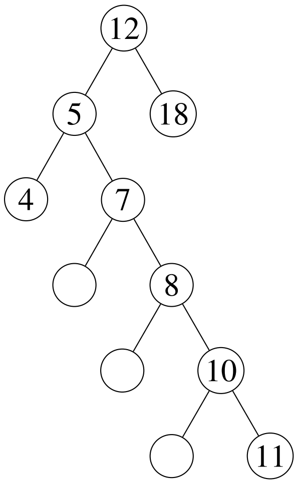
Lucas van der Laan
s1047485

1

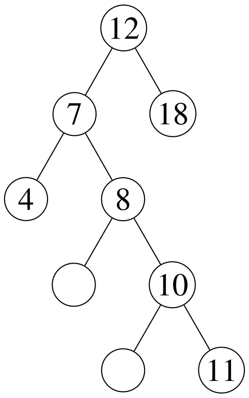
Initial



9 Removed



5 Removed



2

Algorithm 1 Second biggest element in BST

```
procedure Main(tree)
    Require: tree - The root of the binary search tree
    if tree is NIL or (tree.left and tree.right is NIL) then
        return NIL
    end if
    (biggest, second) ← Recursion(tree, 0, 0)
    return (biggest, second)
end procedure
procedure Recursion(tree, biggest, second)
    if second > tree.value and tree.right is NIL then
        return (biggest, second)
    end if
    nodeChoice ← NIL
    if tree.right ≠ NIL then
        nodeChoice ← tree.right
    else if tree.left ≠ NIL then
        nodeChoice ← tree.left
    end if
    if nodeChoice ≠ NIL then
        (biggestChild, secondChild) ← Recursion(nodeChoice, biggest, second)
        if biggest < biggestChild then
            second ← Max(biggest, secondChild)
            biggest ← biggestChild
        else
            second ← Max(second, biggestChild)
        end if
    end if
    return second
end procedure
```

We know for this algorithm that it can never exceed n, because it will only ever touch a node that exists, so that makes it already maximum $\mathcal{O}(n)$. We also know that for a specific tree, namely a tree with only right sided children, we would have to go through the entire tree to find the largest and second largest element, as this is a property of BSTs. Thus we know that for a specific tree it has to be $\mathcal{O}(n)$ and we know the maxmium can only ever be $\mathcal{O}(n)$, thus it has to be $\mathcal{O}(n)$.

5

First sort the array using merge sort, this has a complexity of $\mathcal{O}(n \lg n)$.
We then add the median of the array, and use that median as a pivot and then do this until everything has been added. This is $\mathcal{O}(n \lg n)$ because the sorting takes priority.