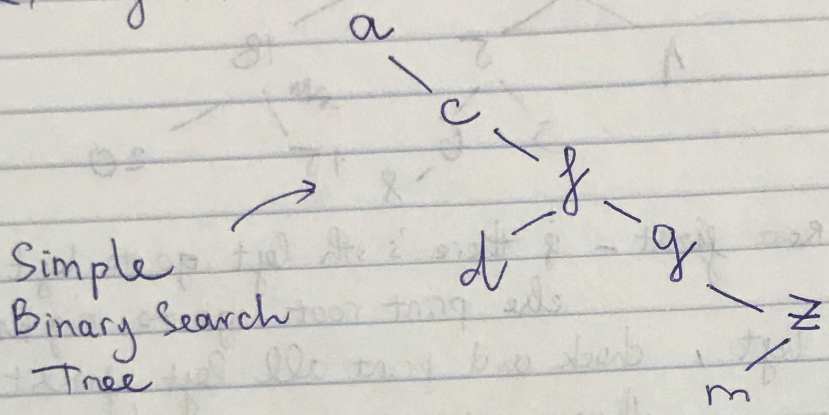
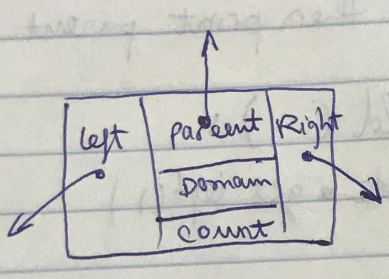
[a, c, f, d, g, z, m]

First will be root, others will append according by, left <, right >



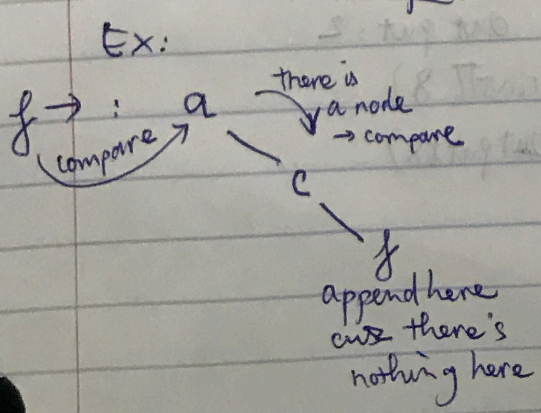
Class_Node:

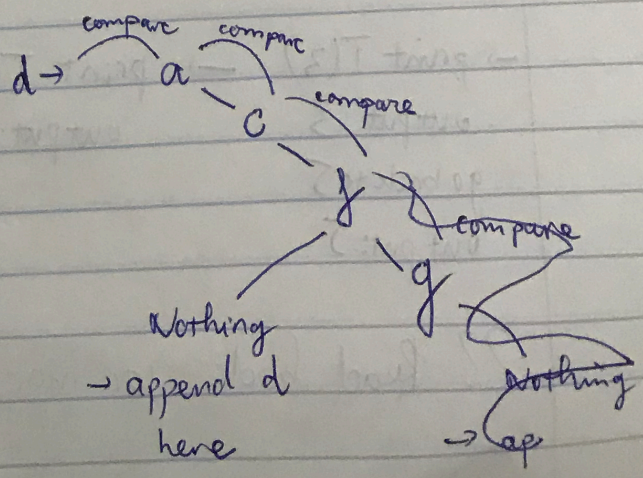


Class Binary Search Tree: size, root, depth or height

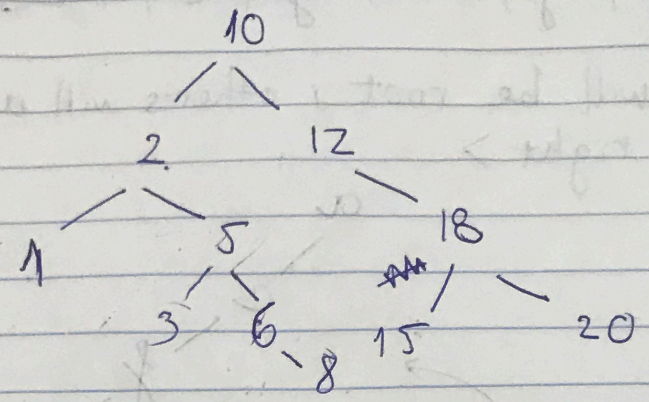
- .is_empty() – Size == 0 ?
- .getRoot() – Return root
- .add or increment()
  * compare starting from Root then go down until height/depth = 0 ?? No need

Ex:





?? Print Tree:



- Check Root first - if there's sth left, go to left
  else print root, go to right
- If go left, check and print all left first
  then print parent → right → come back to root

```
PrintTree (Node)
If IS-left-child ( ):
    PrintTree (Node.getleft())
Else:
    Print (Node)
    ~~PrintTree (Node.getright())~~
    If IS-right-child ()
        PrintT( Node.getright() ) parent
    ~~Else~~: → go back to parent → print parent → printT(parent.right)
```

Step 1: printT(10) → printT(2) → printT(1) → print T(5)
(output: 1)
go back to (2)
output: 2

→ print T(3) → print T(6) → printT(8)
output: 3 output: 6 output 8)
go back to 5
output: 5

?? Read book for more