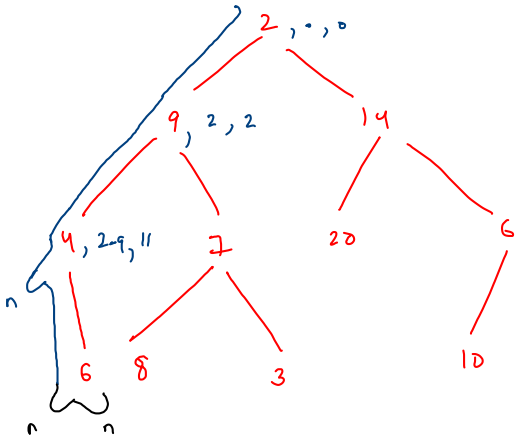
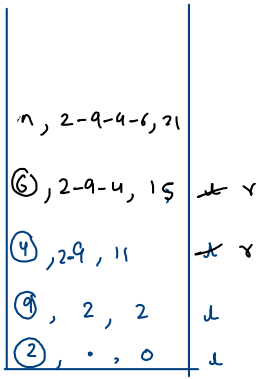


wrong

lo = 14

hi = 40



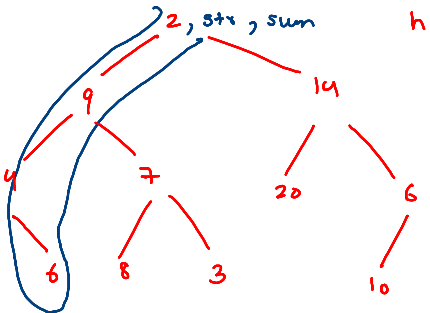
if root == null?

print ;

}

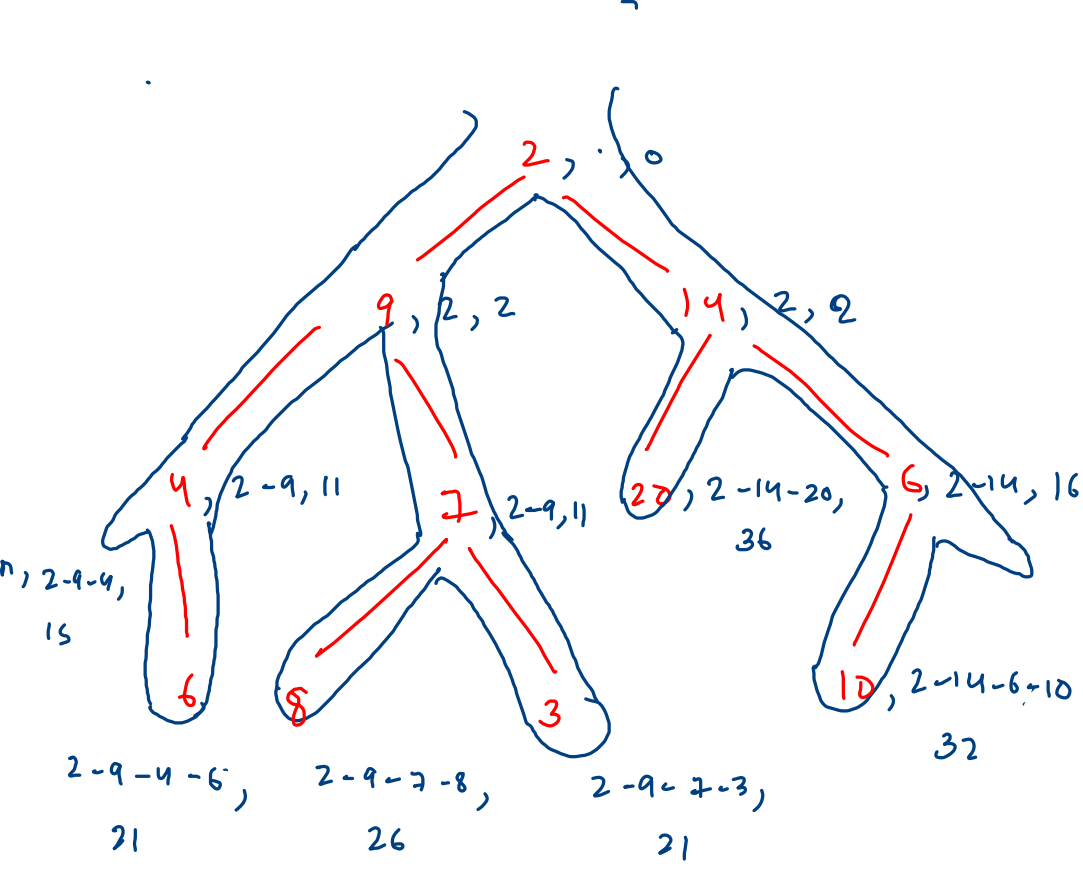
~~2~~ - 9 - 4

[2 - 9 - 4 - 6
  2 - 9 - 4 - 6



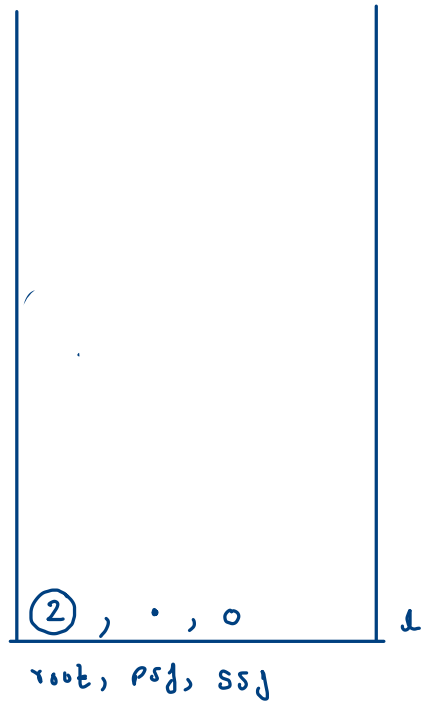
lo = 24

hi = 40



$h_0 = 24$

$h_i = 40$

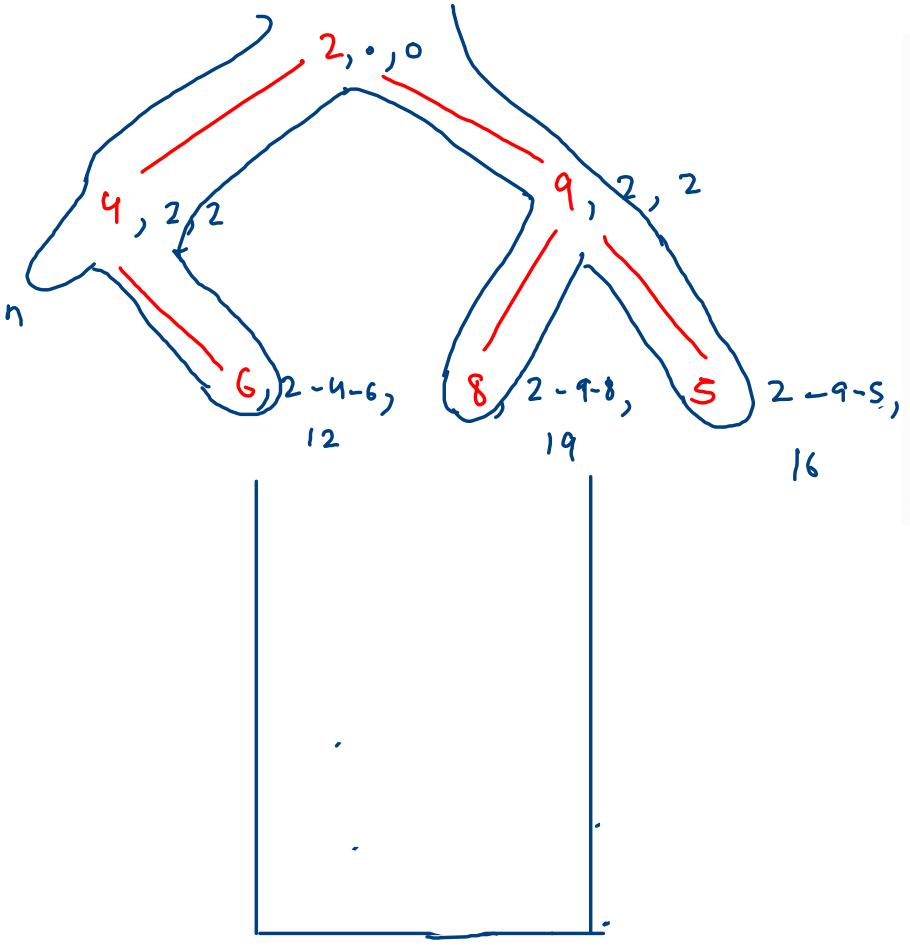


if (root is a leaf node)

2-9-7-8

2-14-20

2-14-6-10



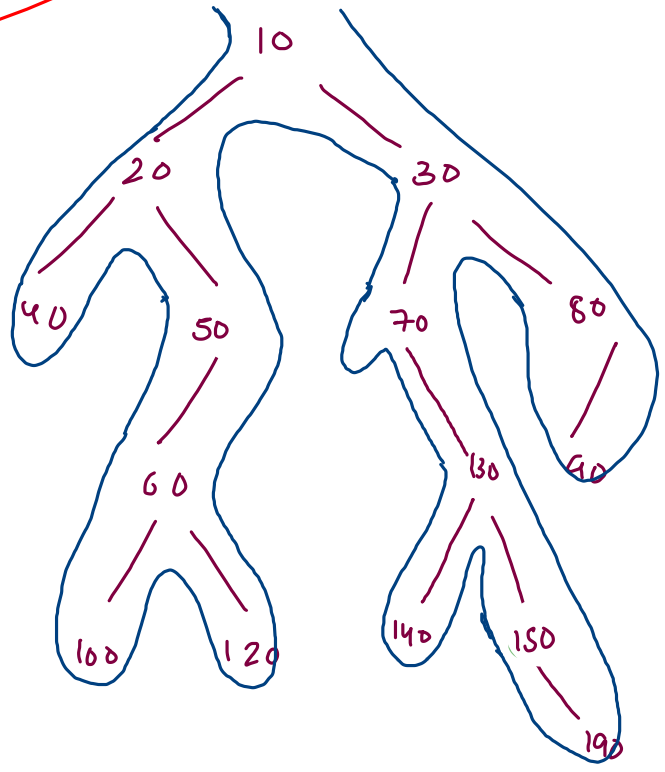

```java
public static void pathToLeafFromRoot(Node node, String path, int sum, int lo, int hi){
  if(node == null) {
      return;
  }

  //Leaf node
  if(node.left == null && node.right == null) {
      //Leaf is not added yet in the path
      path += node.data;
      sum += node.data;

      if(sum >= lo && sum <= hi) {
          System.out.println(path);
      }

      return;
  }

  pathToLeafFromRoot(node.left,path + node.data + " ", sum + node.data, lo,hi);
  pathToLeafFromRoot(node.right, path + node.data + " ",sum + node.data,lo,hi);
}
```

lo = 15

hi = 18

2 - 9 - 5

print single child nodes



60 , 130

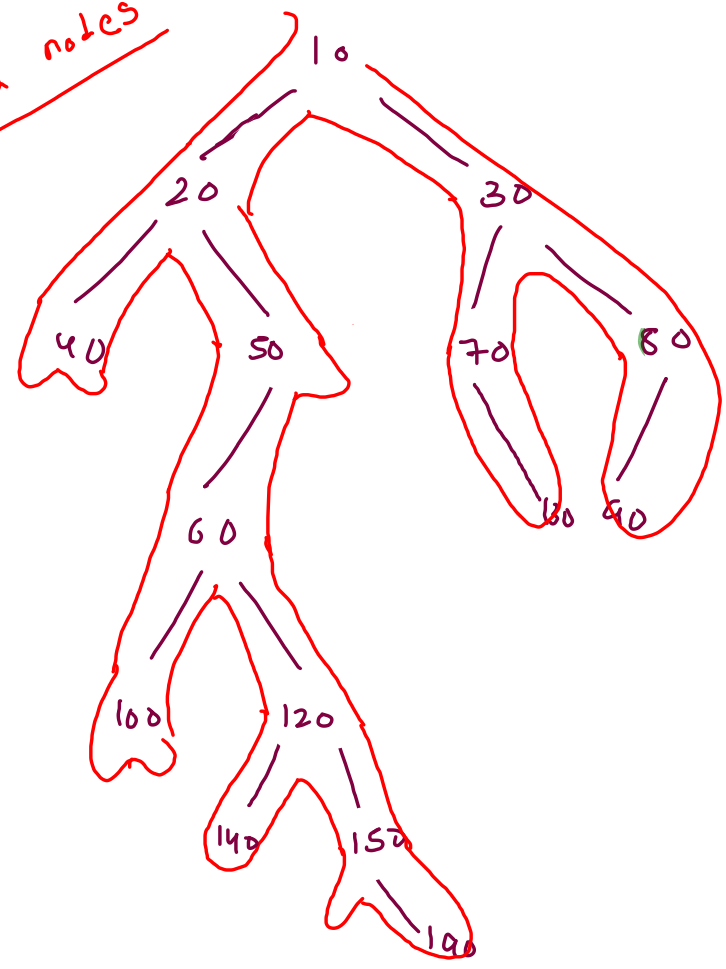
60, 90

Single child node

```
if(node.left != null && node.right==null){
    Syso(node.left.data);
}
if(node.left == null && node.right!=null){
    Syso(node.right.data);
}
```

130, 190

print single child nodes




```java
public static void printSingleChildNodes(Node node) {
    if(node == null) {
        return;
    }

    if(node.left != null && node.right == null) {
        //node is a single child parent, so print its single child
        System.out.println(node.left.data);
    }

    if(node.left == null && node.right != null) {
        //node is a single child parent, so print its single child
        System.out.println(node.right.data);
    }


    printSingleChildNodes(node.left);
    printSingleChildNodes(node.right);
}
```

c

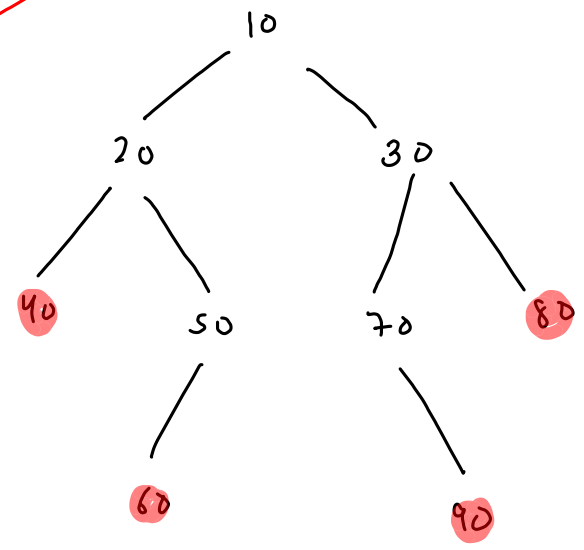
l

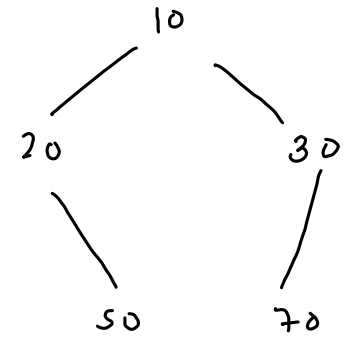
r

60 190 130

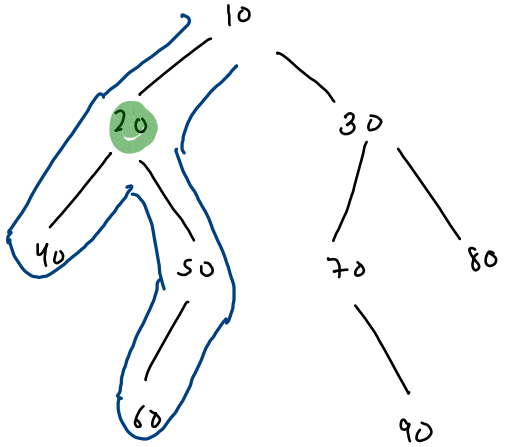
90

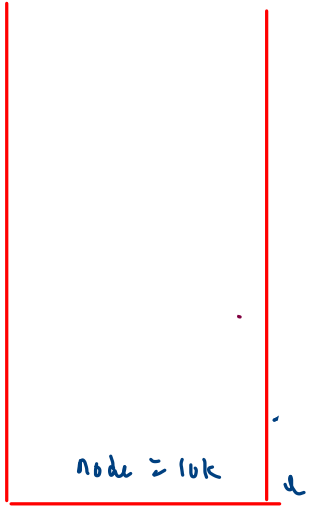
remove leaves

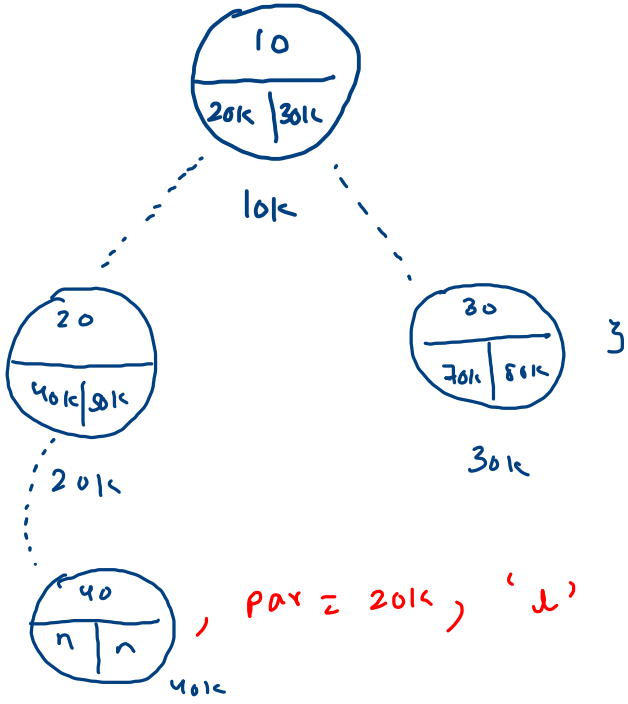




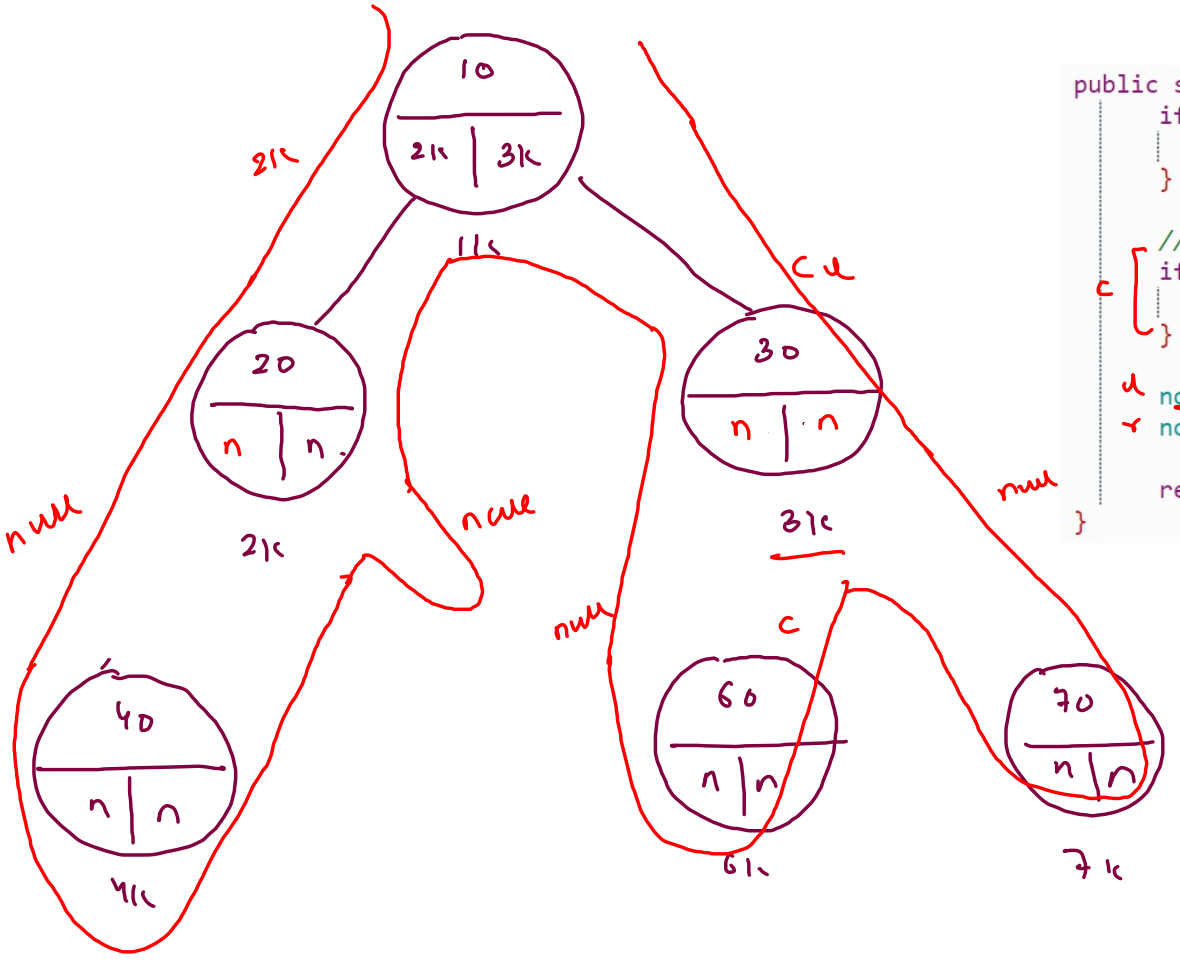

```
void rL (Node node) {
    if(node == null) {
        return;
    }
    if (node.left == null && node.right == null){
        node = null;
        return;
    }
    rL(node.left);
    rL (node.right);
}
```





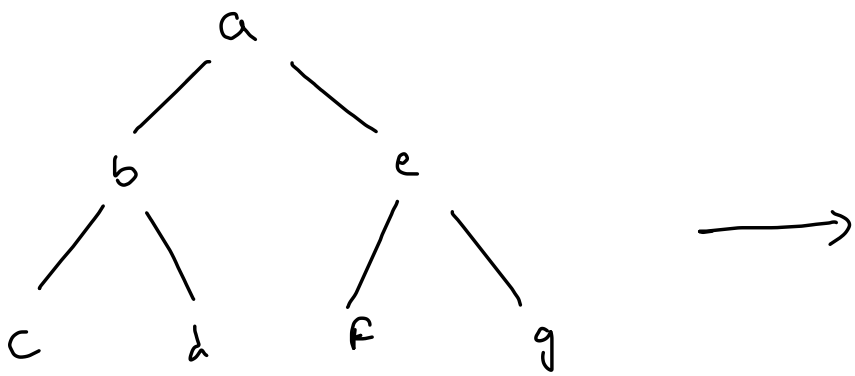


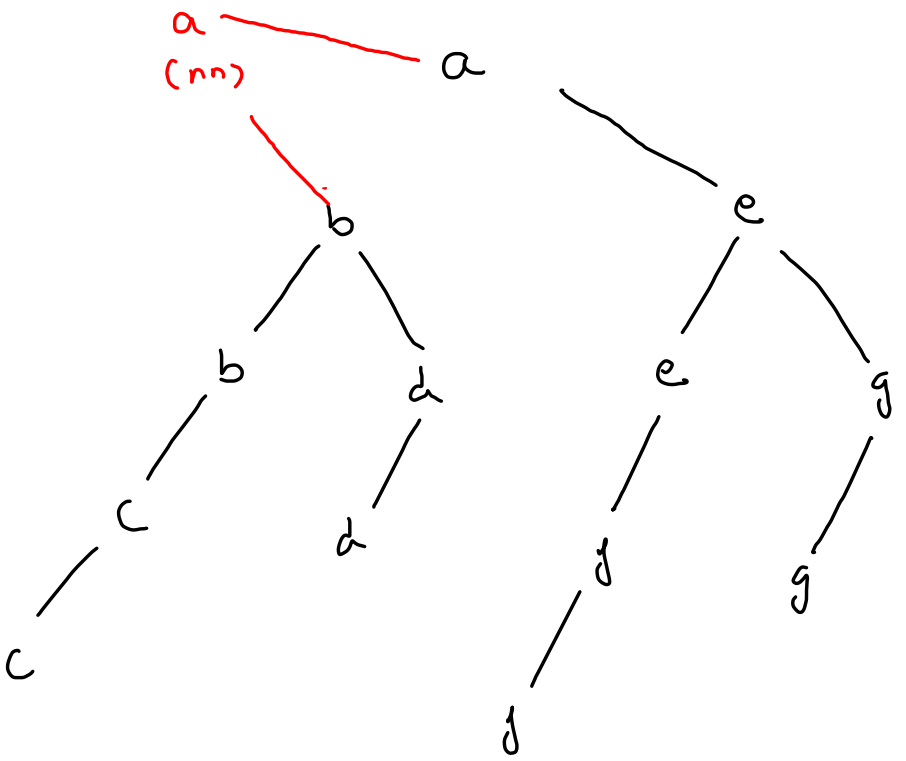

```
void rL (Node node) {
    if(node == null) {
        return;
    }
    if (node.left == null && node.right == null){
        node = null;
        return;
    }
    rL (node.left);
    rL (node.right);
}
```



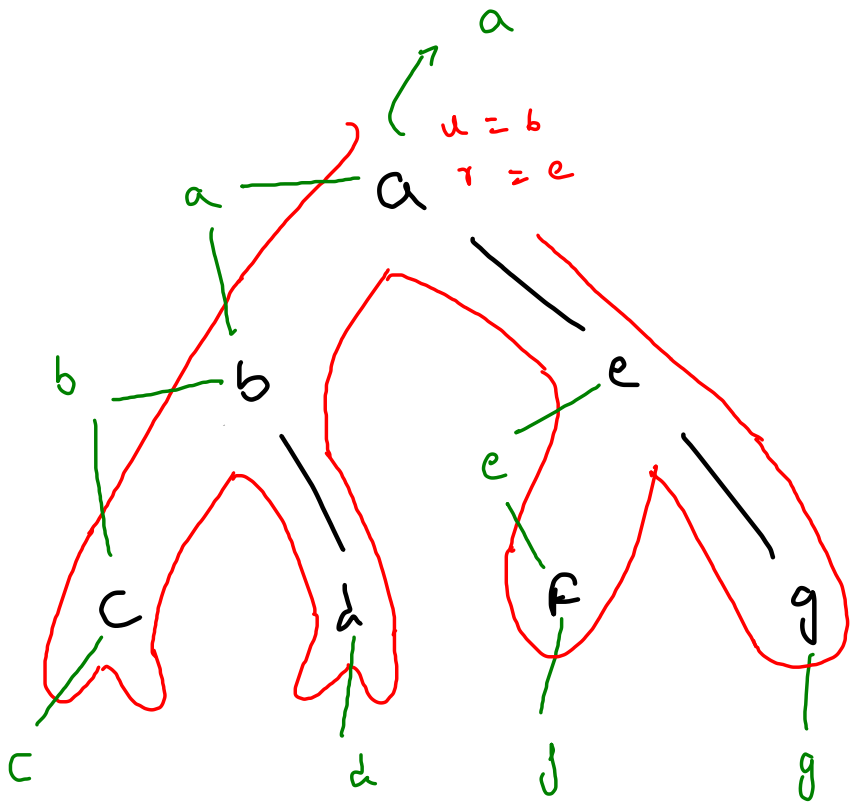

```java
public static Node removeLeaves(Node node){
    if(node == null) {
        return null;
    }

    //Leaf node
    if(node.left == null && node.right == null) {
        return null;
    }

    node.left = removeLeaves(node.left);
    node.right = removeLeaves(node.right);

    return node;
}
```

transform to left cloned



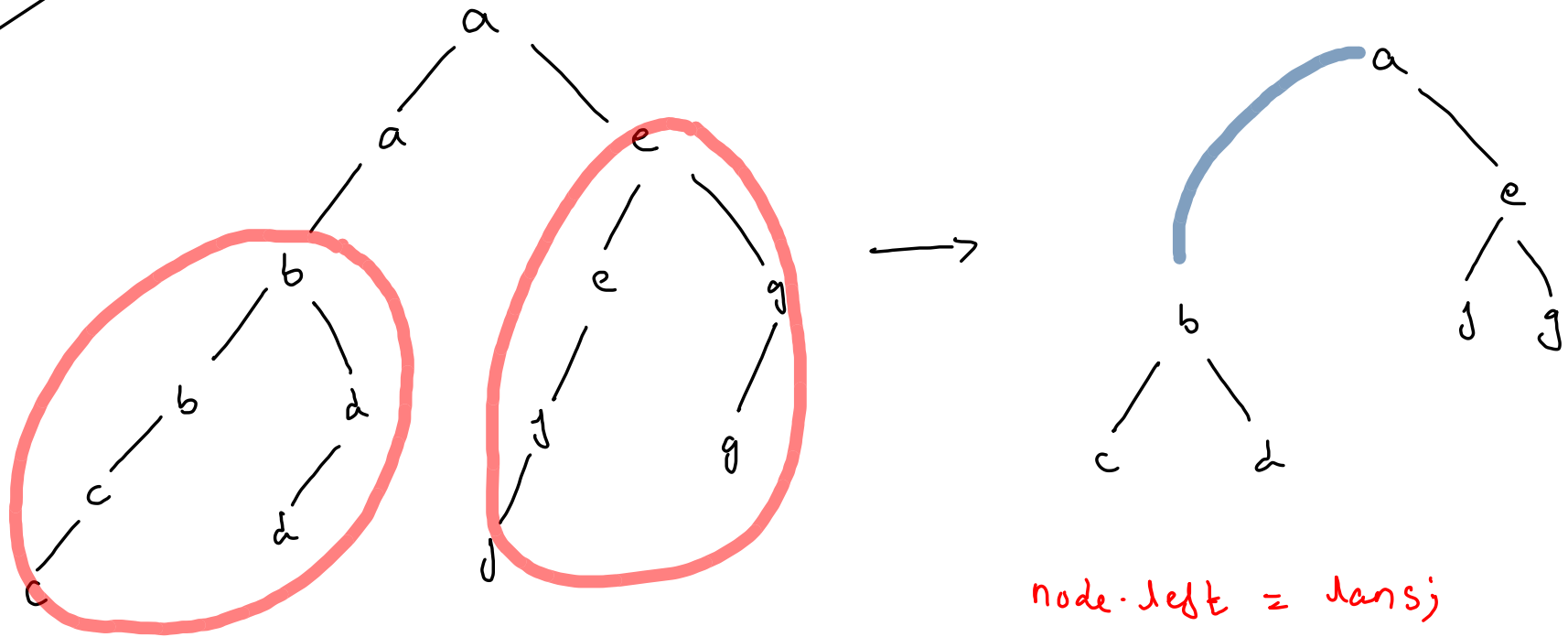


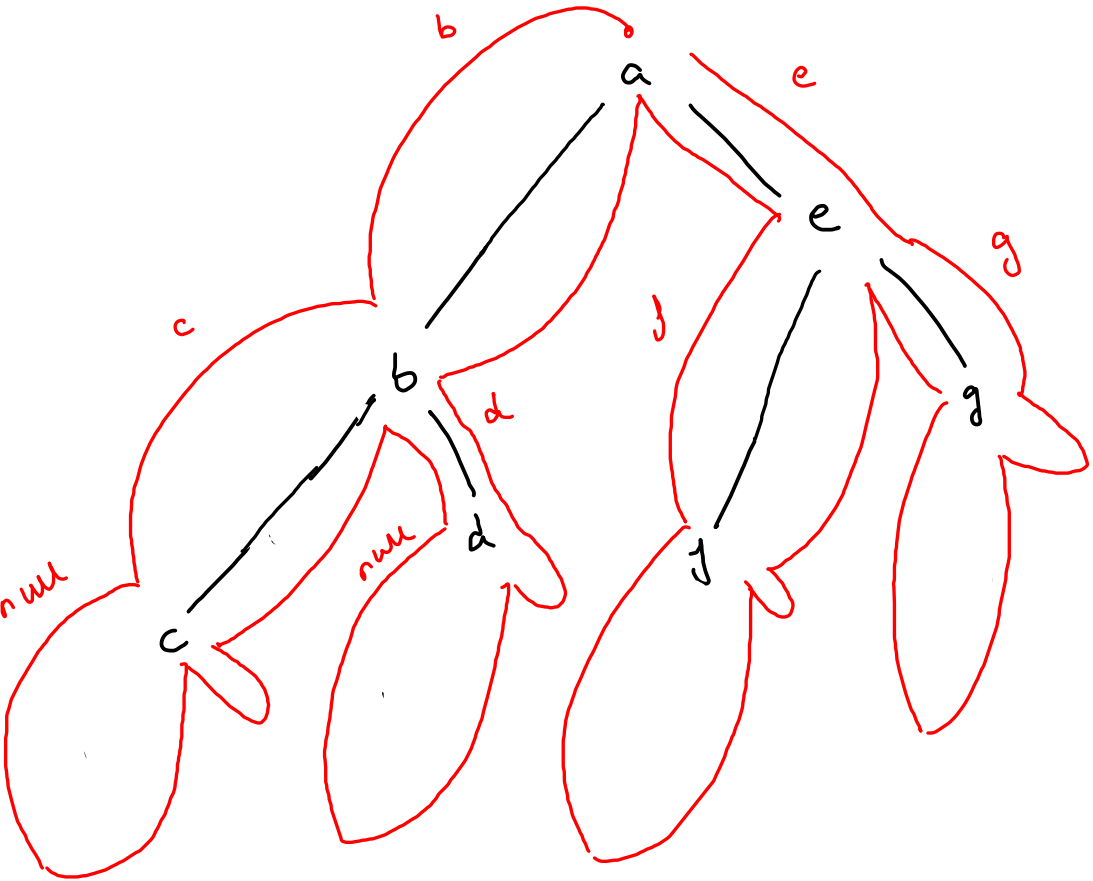

```
Node nn = new Node(node.data);
nn.left = fun(node.left);
node.left = nn;
node.right = fun(node.right);
return node;
```



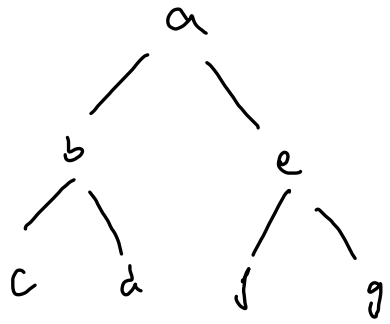

```java
public static Node createLeftCloneTree(Node node){
    if(node == null) {
        return null;
    }

    Node lans = createLeftCloneTree(node.left);
    Node rans = createLeftCloneTree(node.right);

    Node nn = new Node(node.data);
    node.left = nn;
    nn.left = lans;
    node.right = rans;

    return node;
}
```

transform left cloned tree to normal

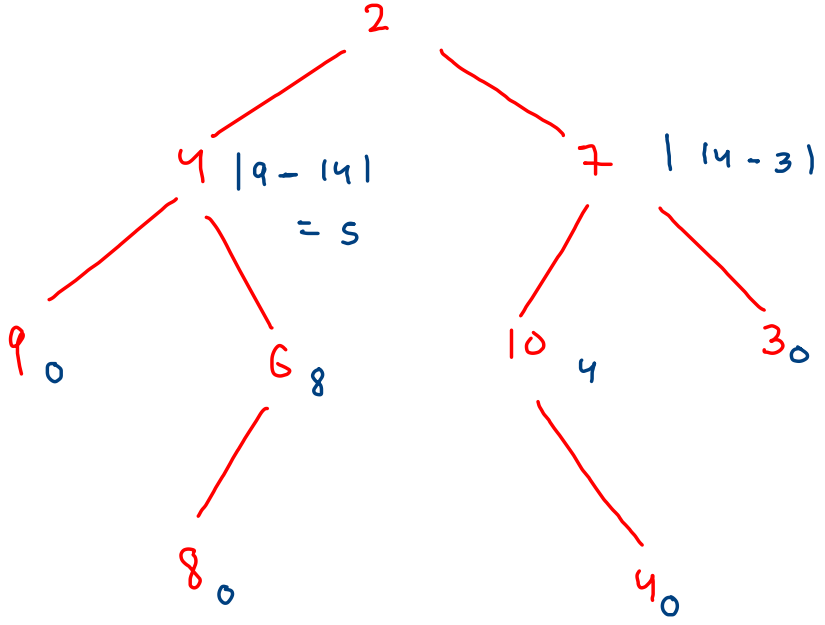





```java
public static Node transBackFromLeftClonedTree(Node node){
    if(node == null) {
        return null;
    }

    node.left = transBackFromLeftClonedTree(node.left.left);
    node.right = transBackFromLeftClonedTree(node.right);

    return node;
}
```



$|(4+9+6+8) - (7+10+4+3)| = 3$



tilt of tree = 3 + 5 + 11 + 8 + 4

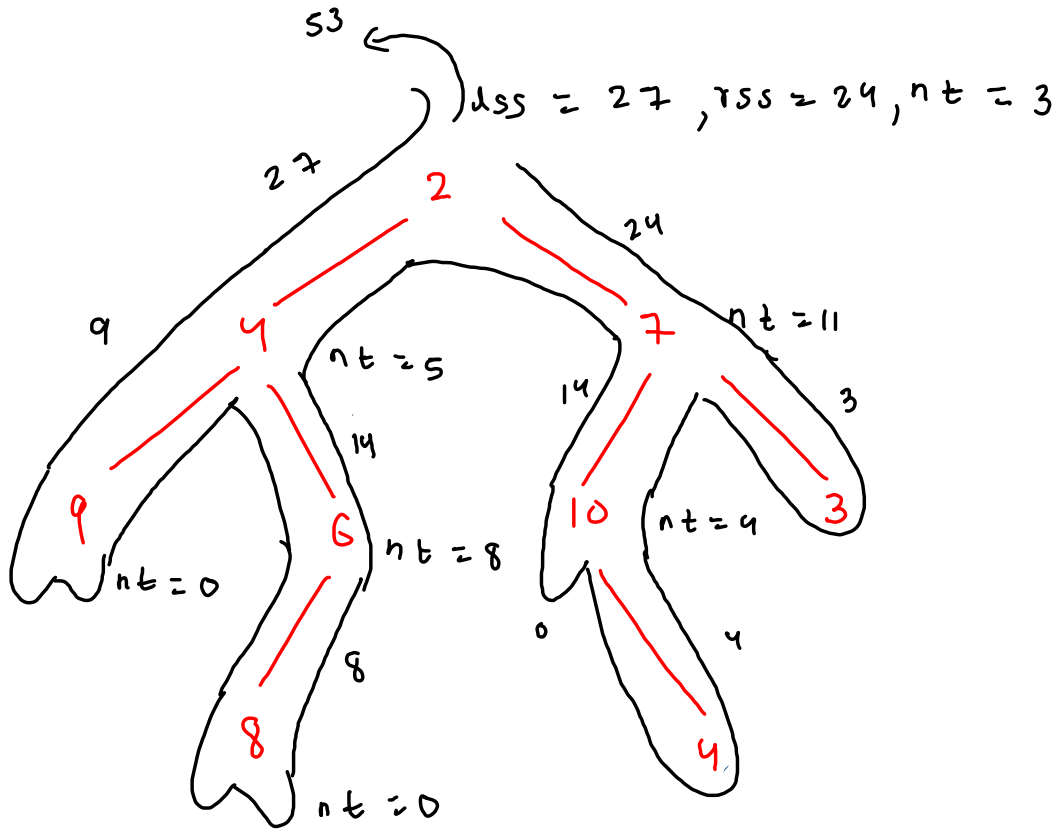
= 31

tilt of a node

| left subtree sum - right subtree sum |

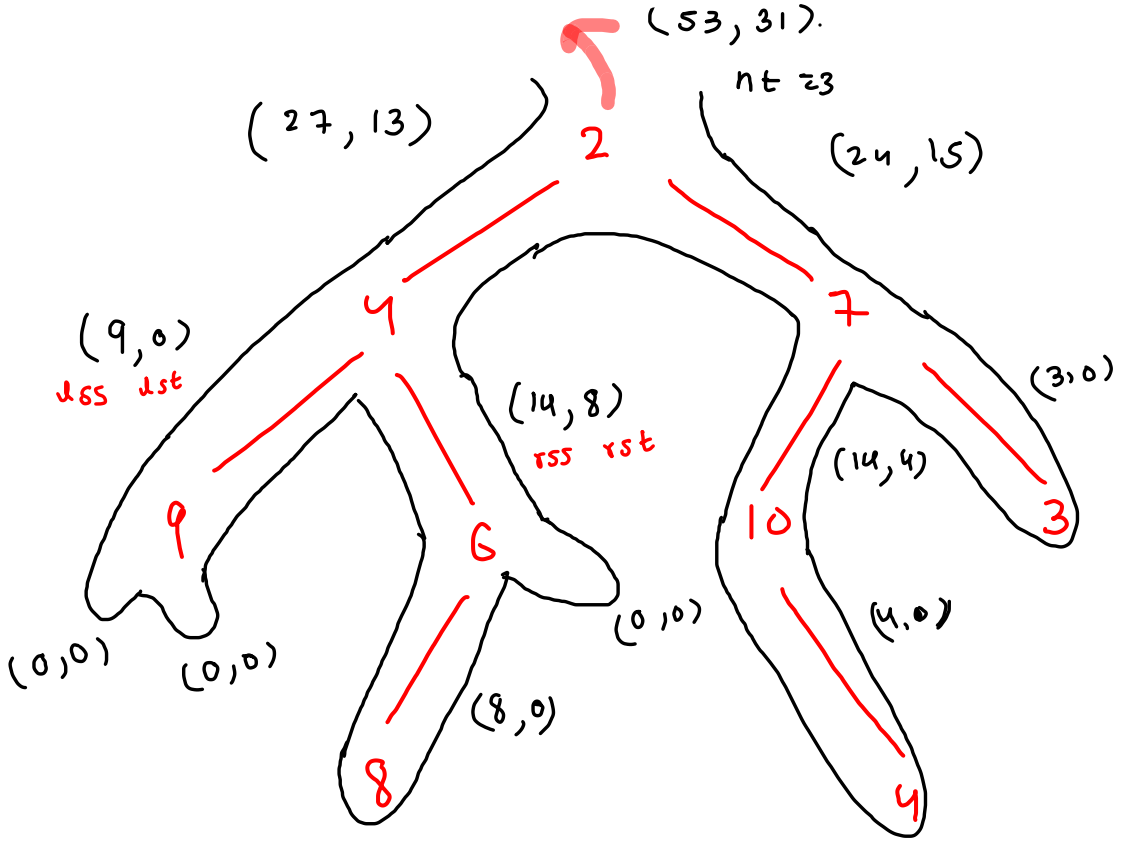
tilt of a tree

$\sum$ tilts $\forall$ nodes



otidt = 0 + 8 + 5 + 4 + 11 + 3

(tidt of a tree)

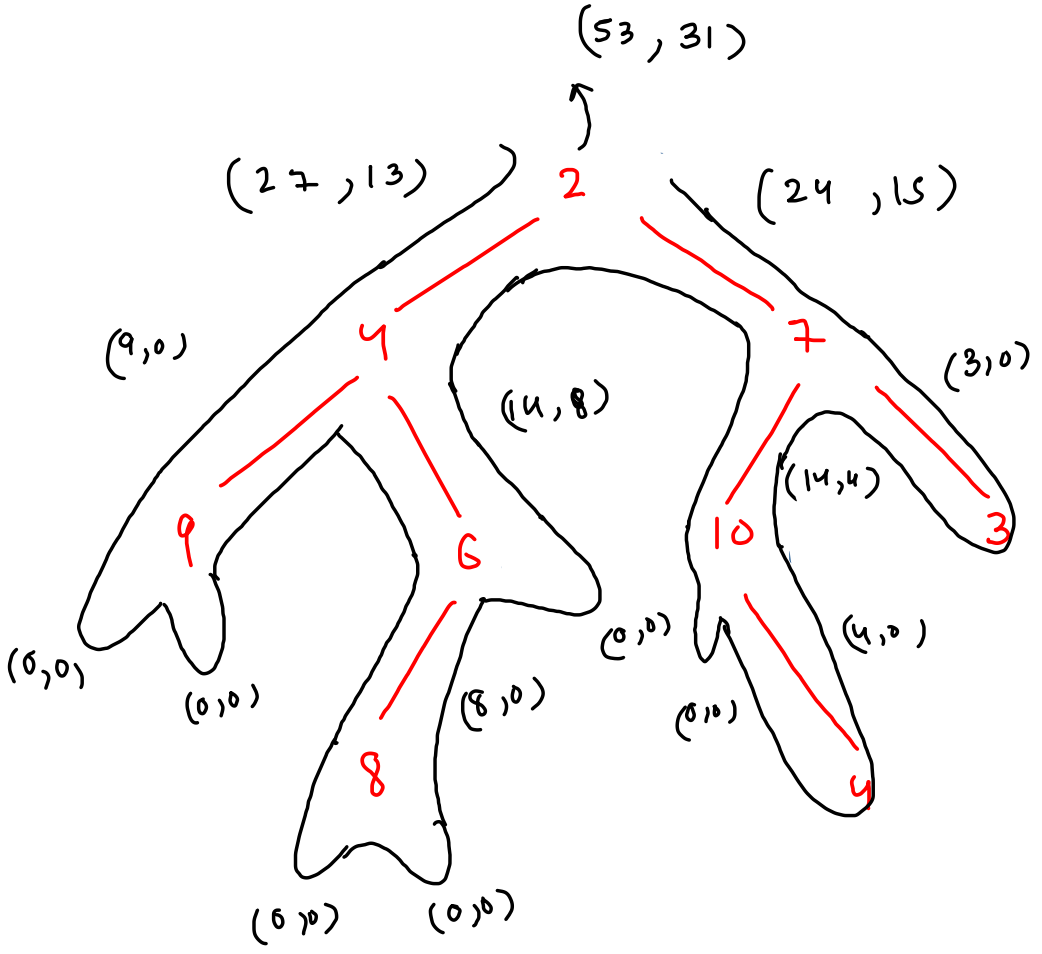


subtree sum, tilt

```
pair {
    int ss;
    int st;
}
```

ss -> subtree sum

st -> subtree tilt



(subtree sum, subtree tilt)

```java
public static pair helper(Node node) {
    if(node == null) {
        return new pair(0,0);
    }

    pair lp = helper(node.left);
    pair rp = helper(node.right);

    int ss = lp.ss + rp.ss + node.data;
    int node_tilt = Math.abs(lp.ss - rp.ss);
    int st = node_tilt + lp.st + rp.st;

    return new pair(ss,st);
}
```