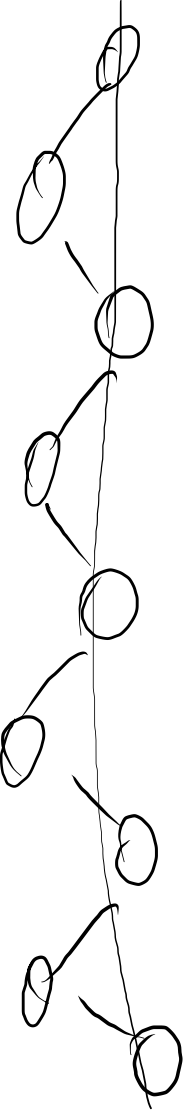
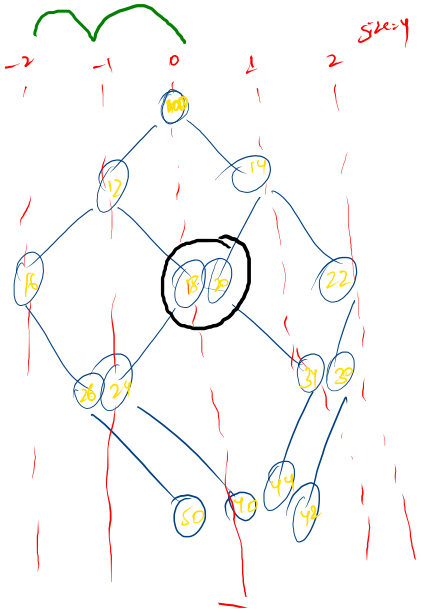
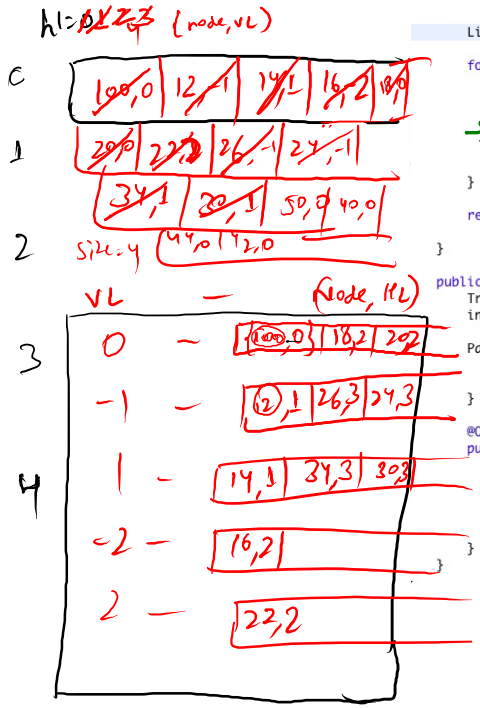
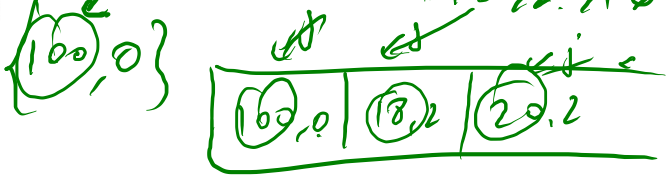


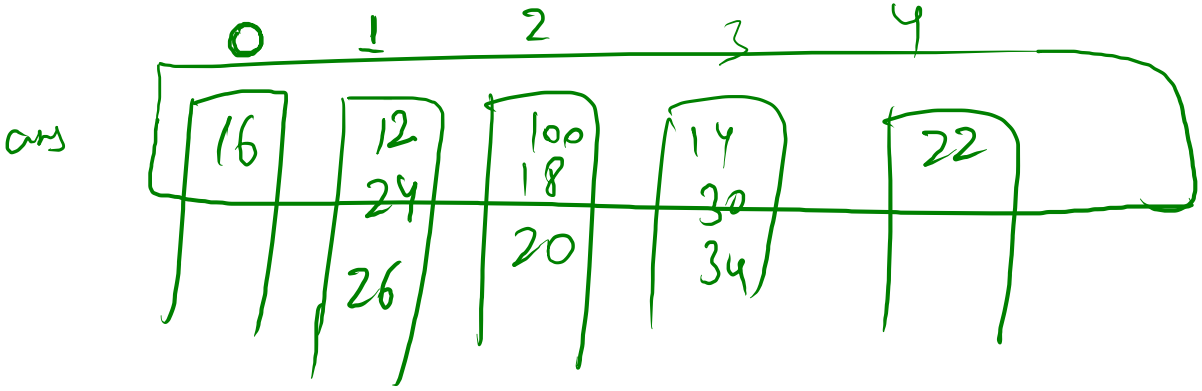




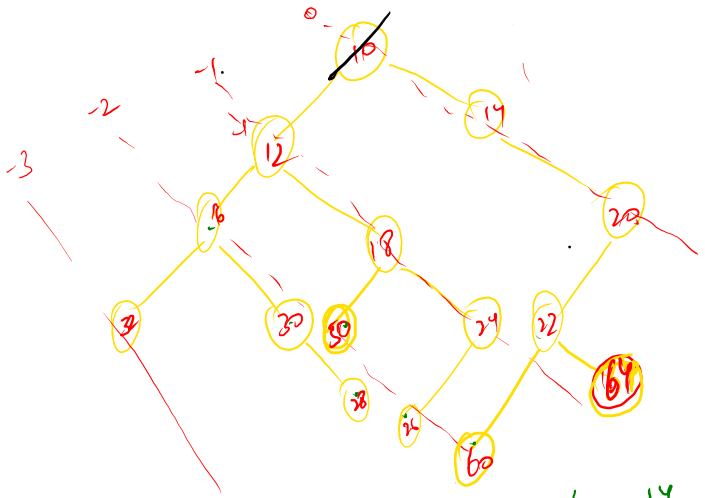

```java
List<List<Integer>> ans = new ArrayList<>();

for (int i = min; i <= max; i++) {
    ArrayList<Pair> temp = map.get(i);
    Collections.sort(temp);
    ans.add(new ArrayList<>());
    for (int j = 0; j < temp.size(); j++) {
        ans.get(i-min).add(temp.get(j).node.val);
    }
}

return ans;
}

public class Pair implements Comparable<Pair> {
    TreeNode node;
    int level;

    Pair(TreeNode node, int level) {
        this.node = node;
        this.level = level;
    }

    @Override
    public int compareTo(Pair o) {
        if (this.level != o.level) {
            return this.level - o.level;
        } else {
            return this.node.val - o.node.val;
        }
    }
}
```

i-min





18
16
~~12~~
14



~~10,0~~	~~12,-1~~	~~14,0~~	~~20,0~~	~~16,-2~~	~~18,-1~~	~~24,-1~~
~~22,-1~~	~~32,-3~~	~~30,-2~~	~~28,-2~~			
~~50,-2~~	~~26,-2~~	~~60,-2~~	~~64,-1~~			

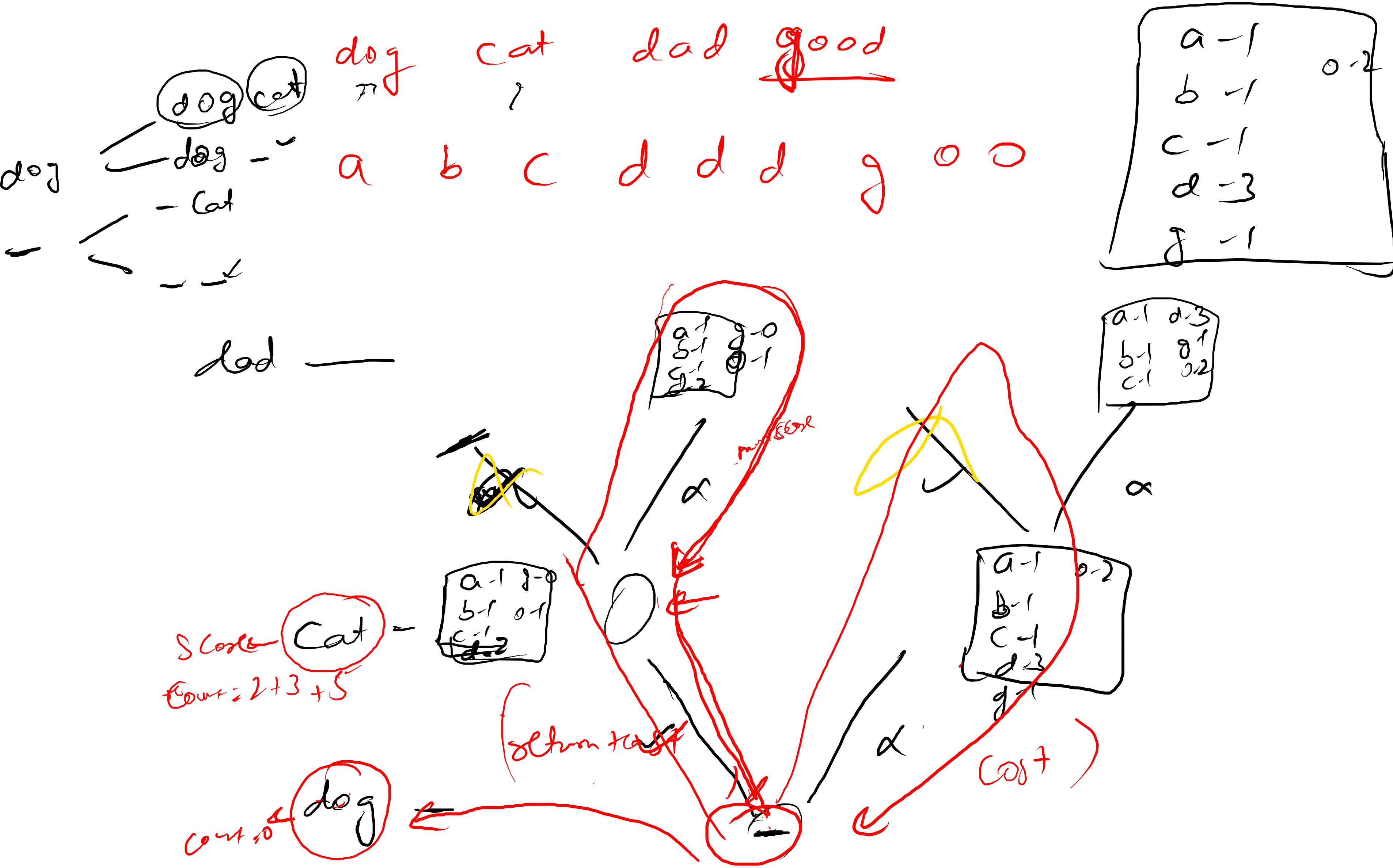
10 14 20

12 18 24 22 64

16 30 ~~28~~ 50 26 60

32

0	10,14,20
-1	12,18,24
-2	16,30,28,50,26,60
-3	32

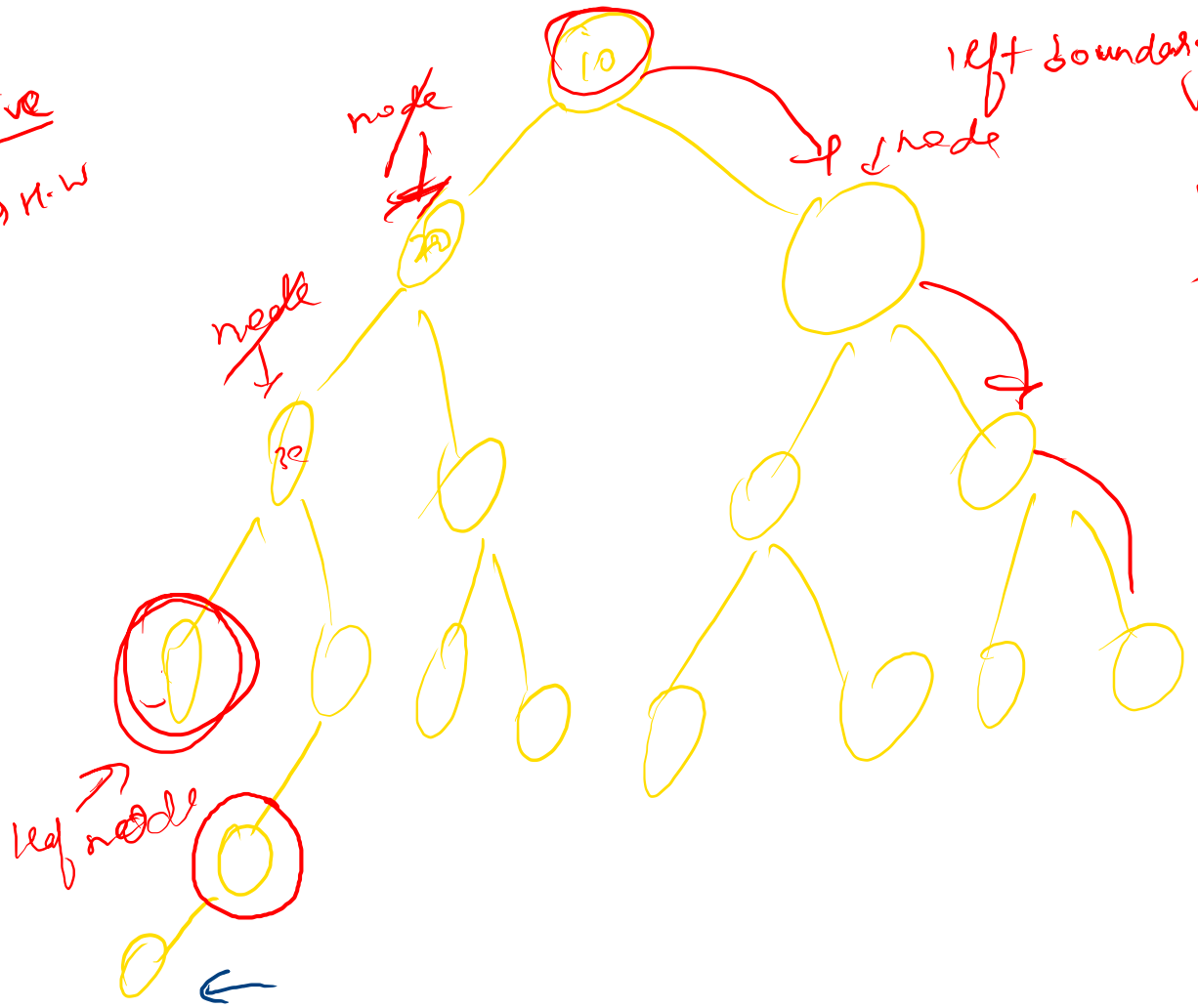


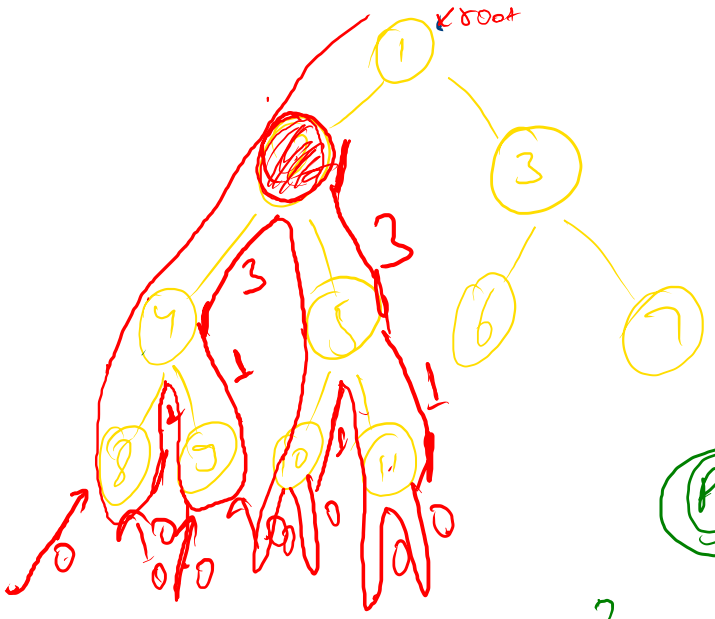
dog cat dad good
a b c d d d g o o
a-1
b-1
c-1
d-3
g-1
o-2
dog
Cat
dad
α
Score
Cat
Count = 2+3+5
dog
count = 0
cost

boundary traversal

recursive

↳ H.W





red = x = 2

blue = y =

$> \frac{11}{2} = 6$

$> \frac{n}{2}$

PC > lc + rc + 1

lc = 3

rc = 3

PC = 4

4 $> \frac{11}{2} \Rightarrow 6$

int lc = 3

int rc = 3

```
int traversal(Node root){
    if(root == null){
        return 0;
    }
    int left = traversal(root.left);
    int right = traversal(root.right);
    if(x == root.val){
        lc = left;
        rc = right;
    }
    return left + right + 1;
}
```