

## Iterative Pre Post InOrder in B-Tree

Stack → Pair

↓  
Node  
state.

State = 0

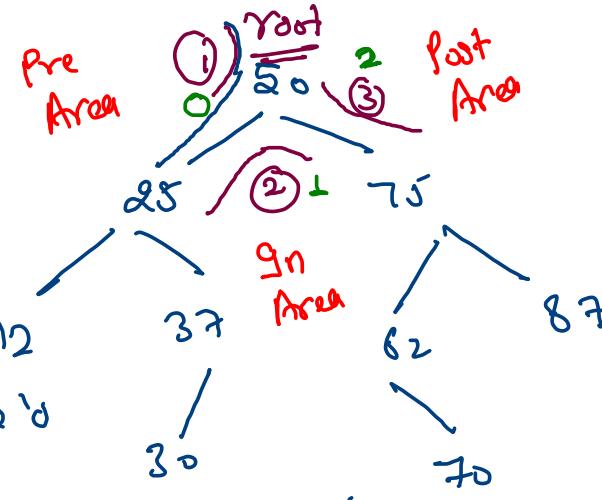
pre.add (node, data)  
add left child in stack with state '0'  
state ++

State = 1

in.add (node, data)  
add right child with state 0  
state ++

State = 2

post.add (node, data) →  
wippe out



Pre → 50 25 12 37 30 75 62 → 87

In → 12 25 30 37 50 62 70 75 87

Post → 12 30 37 25 70 62 87 75 50

content on a single level.

idx, ans  
if [index < str.length] { answer-print  
p=p+1 }

State = 0

st.push(new pair(i+1, ans+charAt()));  
state++

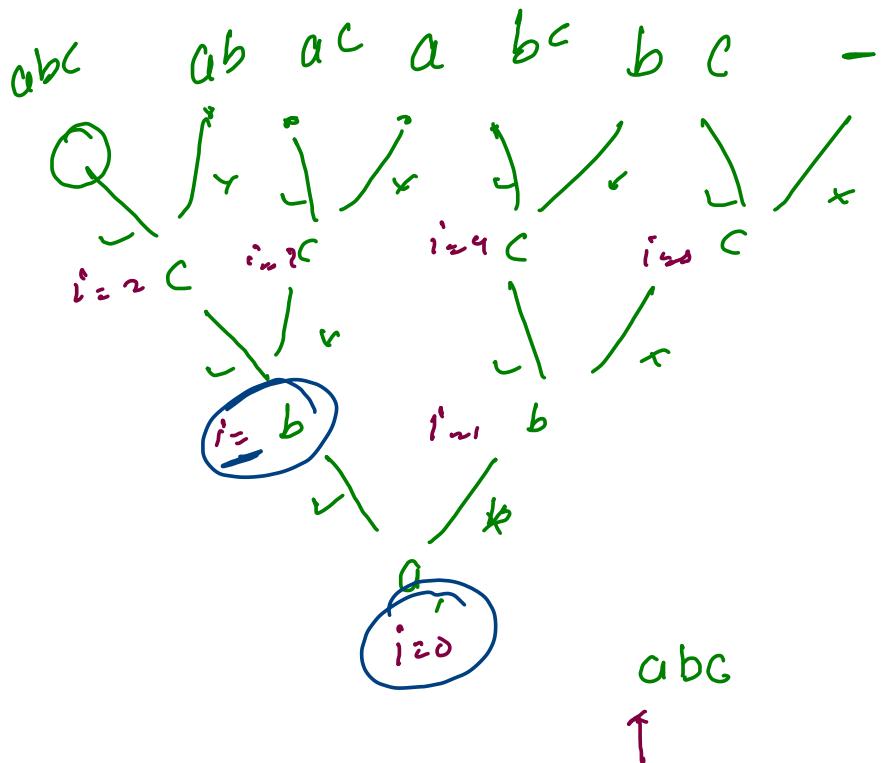
State = 1

st.push(new pair(i+1, ans+"(-)"));

state++

State = 2

st.pop()



a b c  
0 1 2

✓ ---  
✓ - - C  
✓ - b -  
✓ - bC  
✓ a - -  
✓ a - C  
✓ a b -  
✓ abc

Index, answer, state:

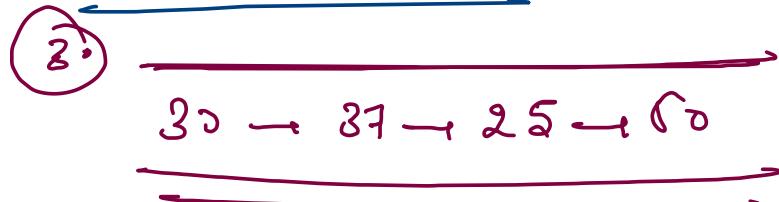
```
public static void iterativeSubseq(String str) {  
    Stack<Spair> st = new Stack<>();  
    st.push(new Spair(0, "", 0));  
  
    while(st.size() > 0) {  
        Spair p = st.peek();  
        if(p.indx == str.length()) {  
            // base case  
            System.out.println(p.ans);  
            st.pop();  
        } else if(p.state == 0) {  
            // no call  
            st.push(new Spair(p.indx + 1, p.ans + "- ", 0));  
            p.state++;  
        } else if(p.state == 1) {  
            // yes call  
            st.push(new Spair(p.indx + 1, p.ans + str.charAt(p.indx) + " ", 0));  
            p.state++;  
        } else {  
            // wipe out  
            st.pop();  
        }  
    }  
}
```

Calls  
margin

## find and Node to Root path.

① find - 30 T/F

② Node to root path.

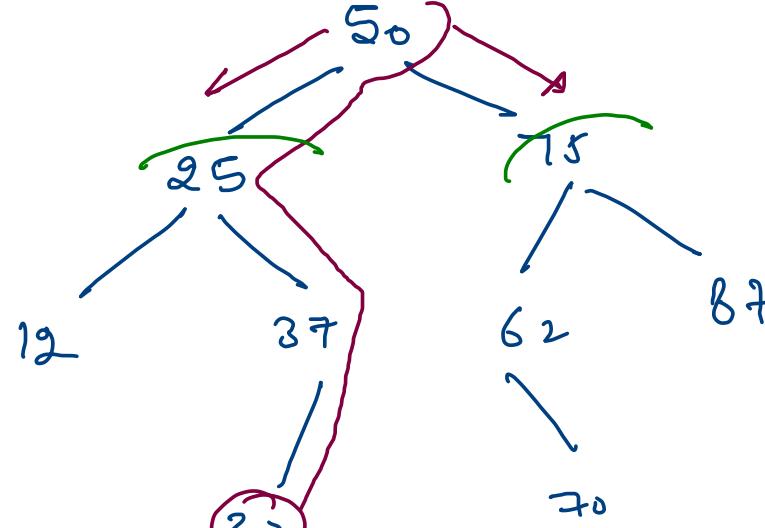


find → Expectation

null → return false

self check  
left check  
right check

Merging



True (Present)

find(50) → True (Present)

find(50.left) → T  
find(50.right) → F

find(50.right) → F

# Arrays

① subarray.

② subset

Subarray → {1, 2, 3}

$$\frac{3 \times 2^2}{2} = ⑥$$

$$n \cdot = \frac{n(n+1)}{2}$$

- ① {1}
- ② {1, 2}
- ③ {1, 2, 3}
- ④ {2}
- ⑤ {2, 3}
- ⑥ {3}

Subset - → {1, 2, 3}

$$n \cdot = 2^n$$

- ① {}
- ② {1}
- ③ {2}
- ④ {3}
- ⑤ {1, 2}
- ⑥ {1, 3}
- ⑦ {2, 3}
- ⑧ {1, 2, 3}

$$= 2^3 = 8$$

# Strings

① Substring -

② Subseq

Substring -

abc

$$n \cdot = \frac{n(n+1)}{2}$$

- ① a
- ② ab
- ③ abc
- ④ b
- ⑤ bc
- ⑥ c

Subseq -

abc

- - -
- - c
- b -
- bc
- a --
- c - c
- a b -
- a b c

- ① -
- ② c
- ③ b
- ④ bc
- ⑤ a
- ⑥ ac
- ⑦ ab
- ⑧ abc

```

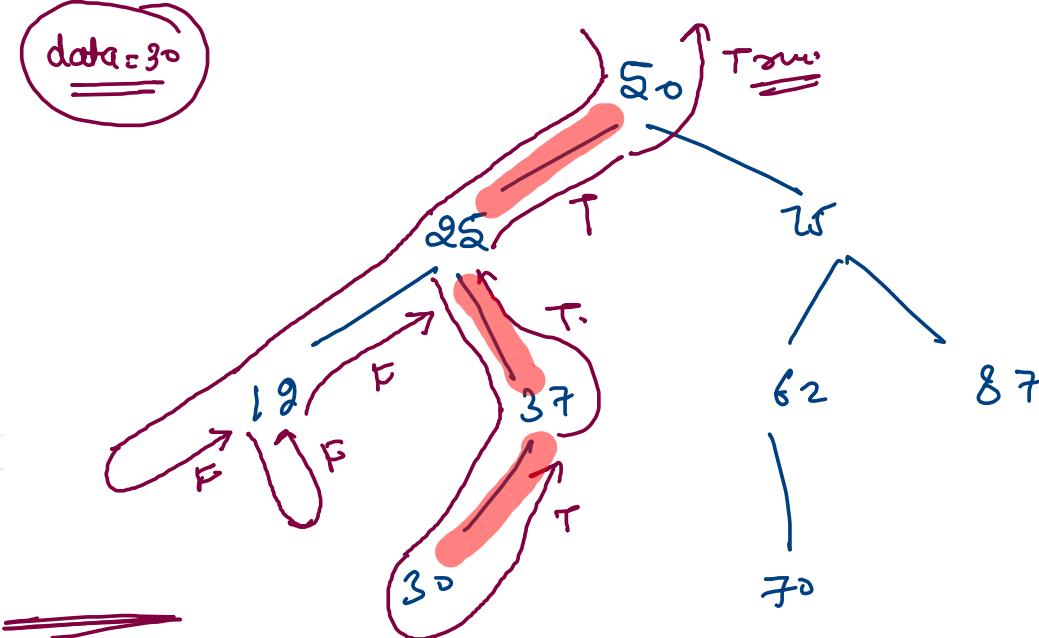
public static boolean find(Node node, int data){
    if(node == null) return false;

    if(node.data == data) return true;

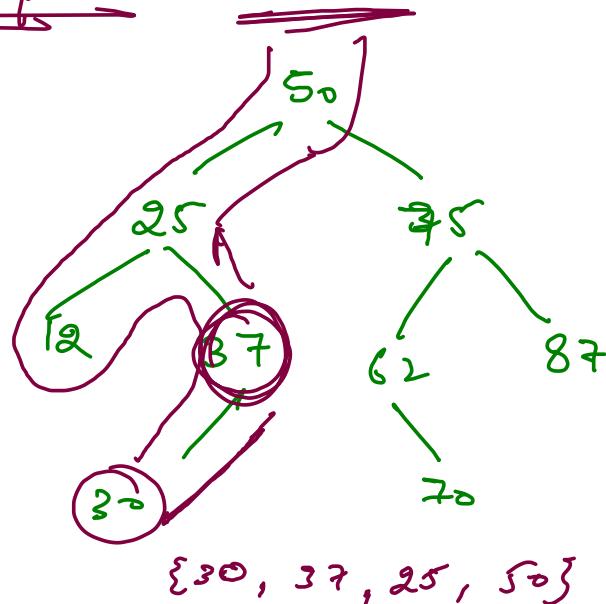
    boolean res = false;
    res = find(node.left, data);
    if(res == true) return true;

    res = find(node.right, data);
    return res;
}

```

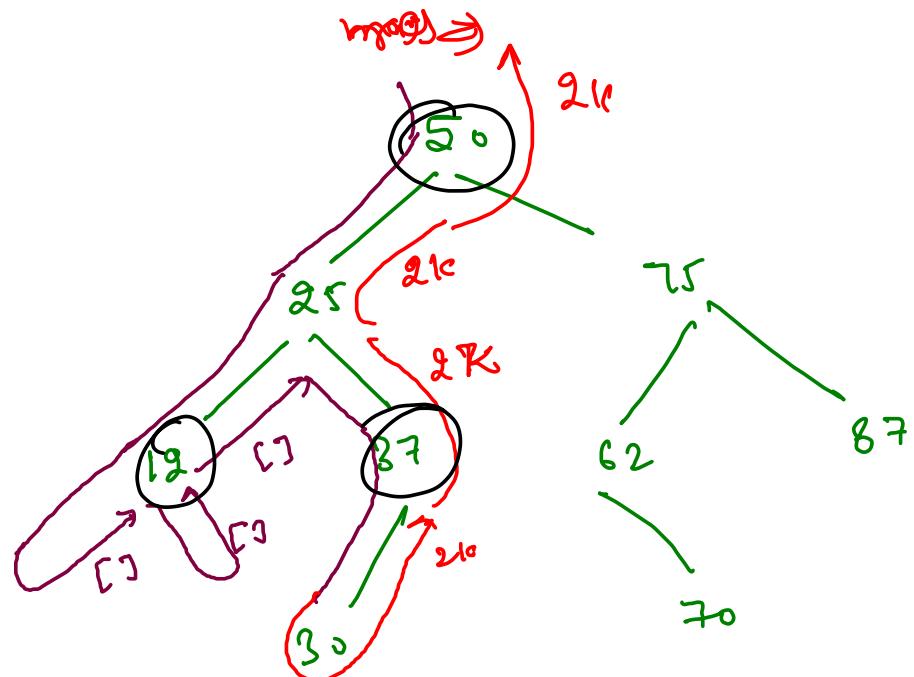


node to Root path



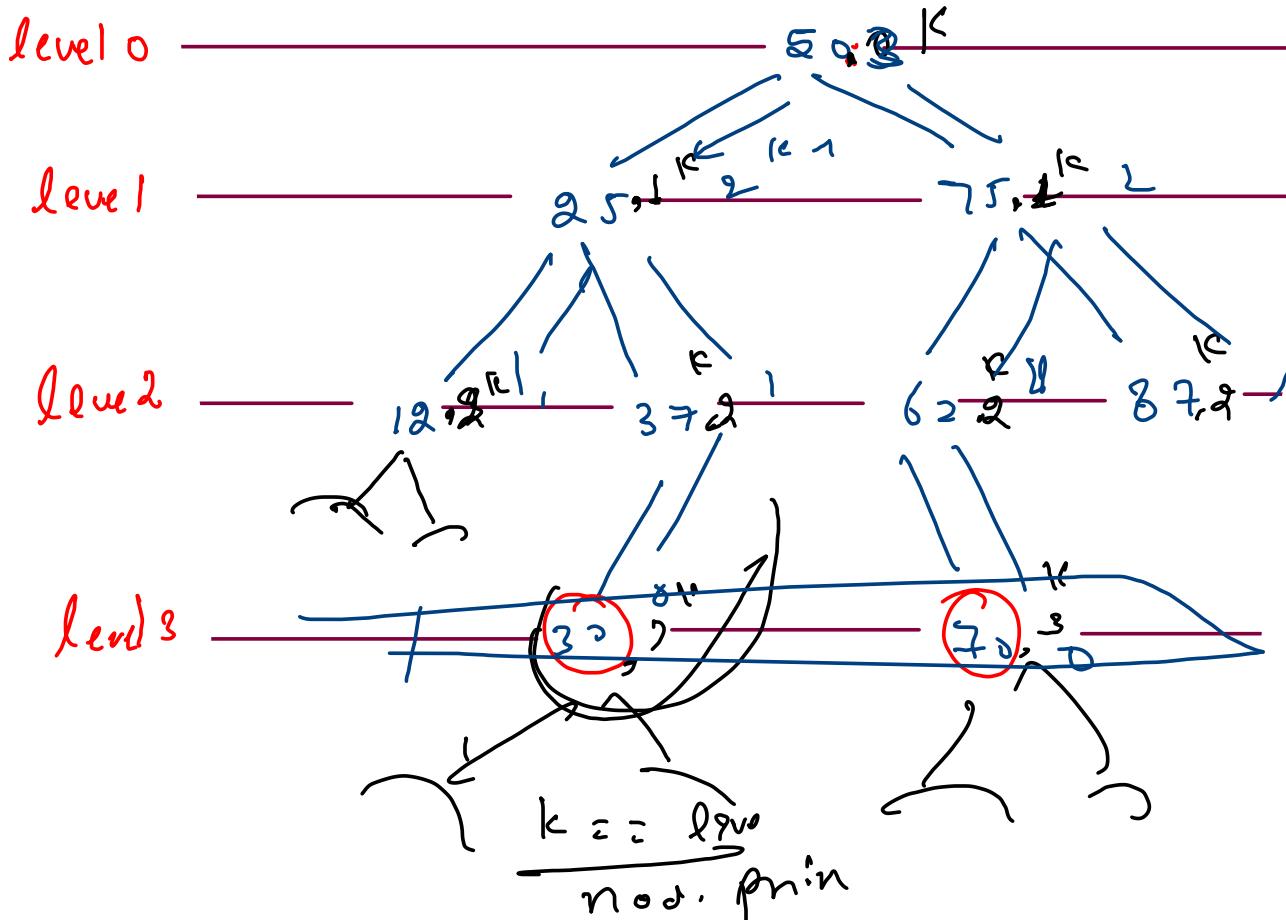
(30)

```
public static ArrayList<Integer> nodeToRootPath(Node i  
    if(node == null) return new ArrayList<>();  
  
    if(node.data == data) {  
        ArrayList<Integer> bres = new ArrayList<>();  
        bres.add(node.data);  
        return bres;  
    }  
  
    ArrayList<Integer> res;  
    res = nodeToRootPath(node.left, data);  
    if(res.size() > 0) {  
        res.add(node.data);  
        return res;  
    }  
  
    res = nodeToRootPath(node.right, data);  
    if(res.size() > 0) {  
        res.add(node.data);  
        return res;  
    }  
  
    return new ArrayList<>();  
}
```



$2^k$  {30, 37, 25, 50}

Print k level Down →



k level Down

123

# Recursion

null → check \

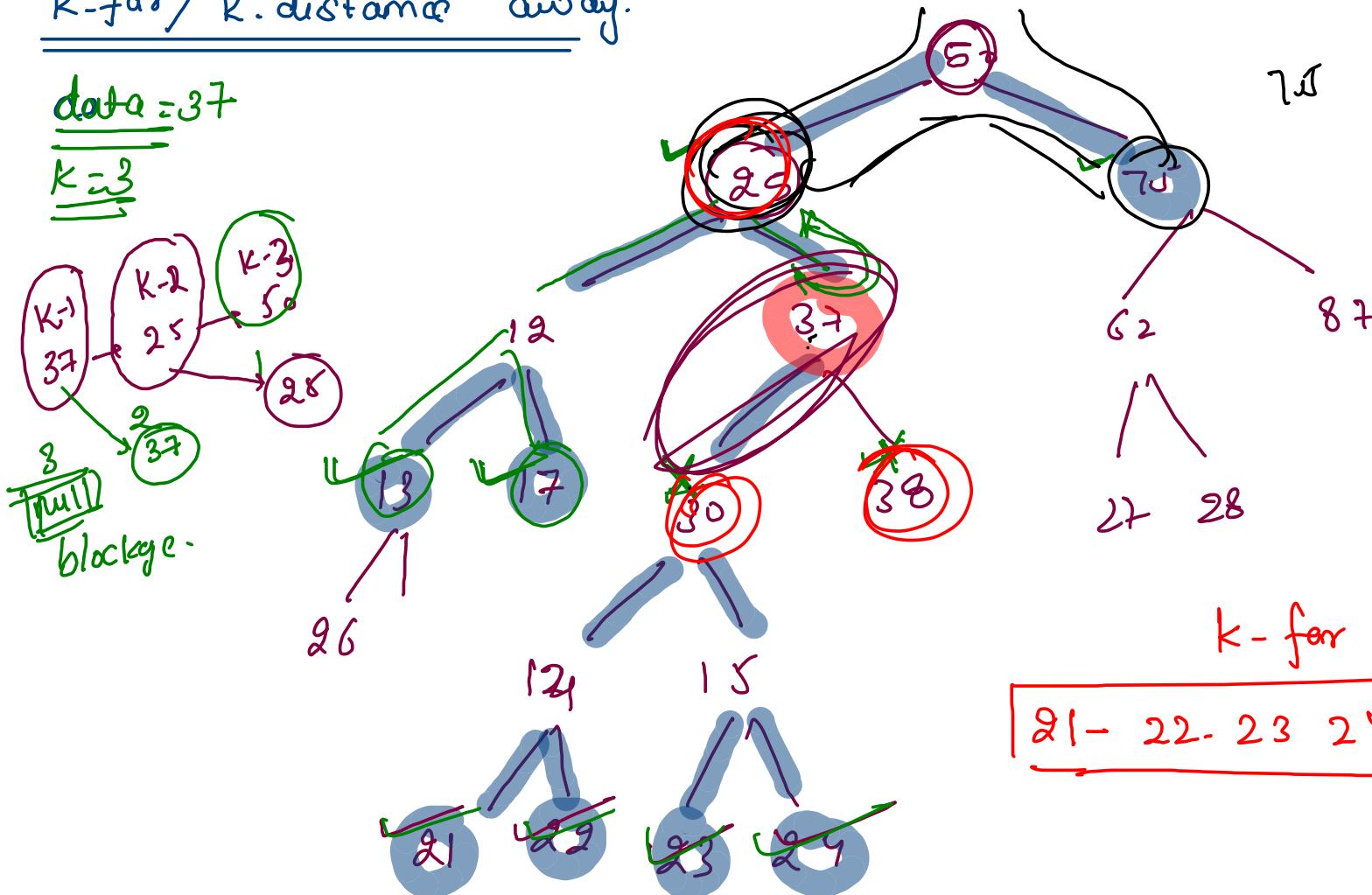
I am checker

left →  
right →

k-far / k.distance away.

data = 37

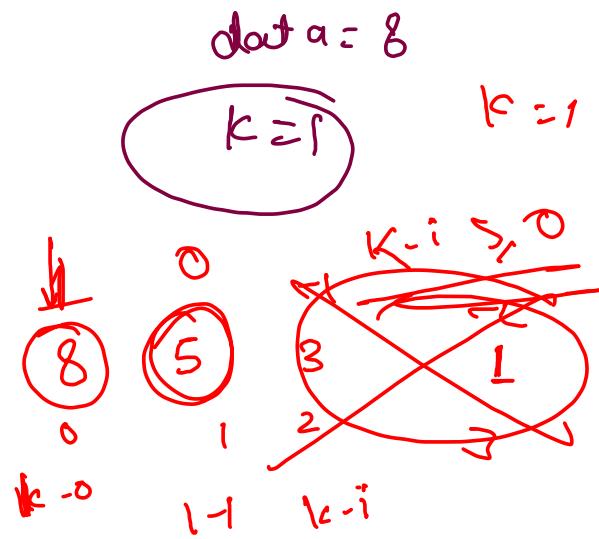
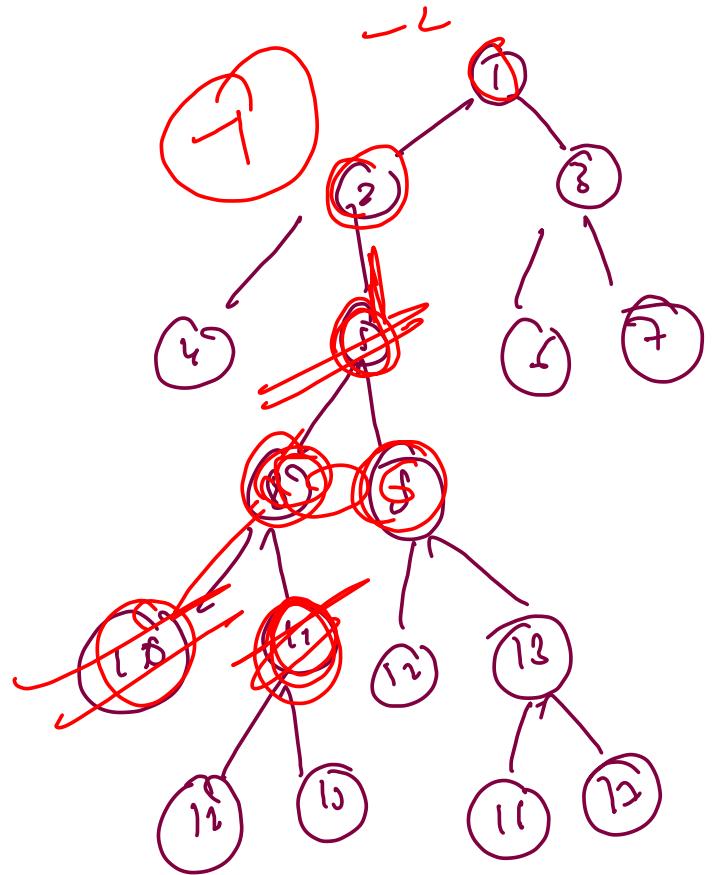
k = 3



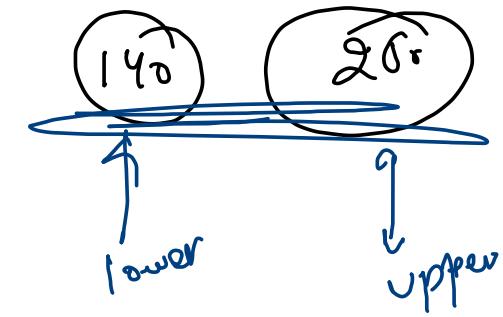
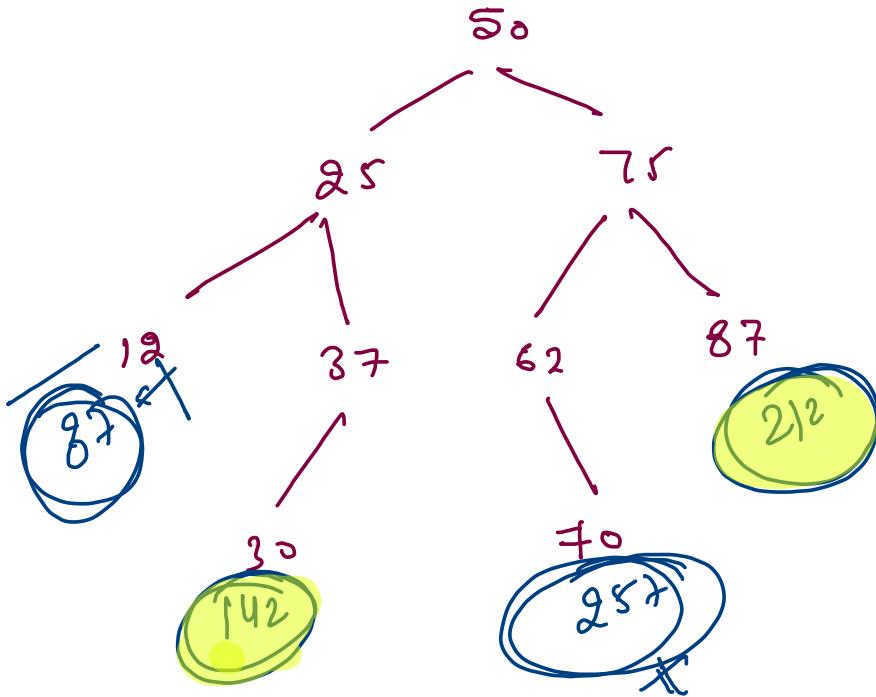
Hint →  
# Nodes to  
root  
path  
+  
# k level down

k-far from 37

21 - 22 - 23 24 13 17 75



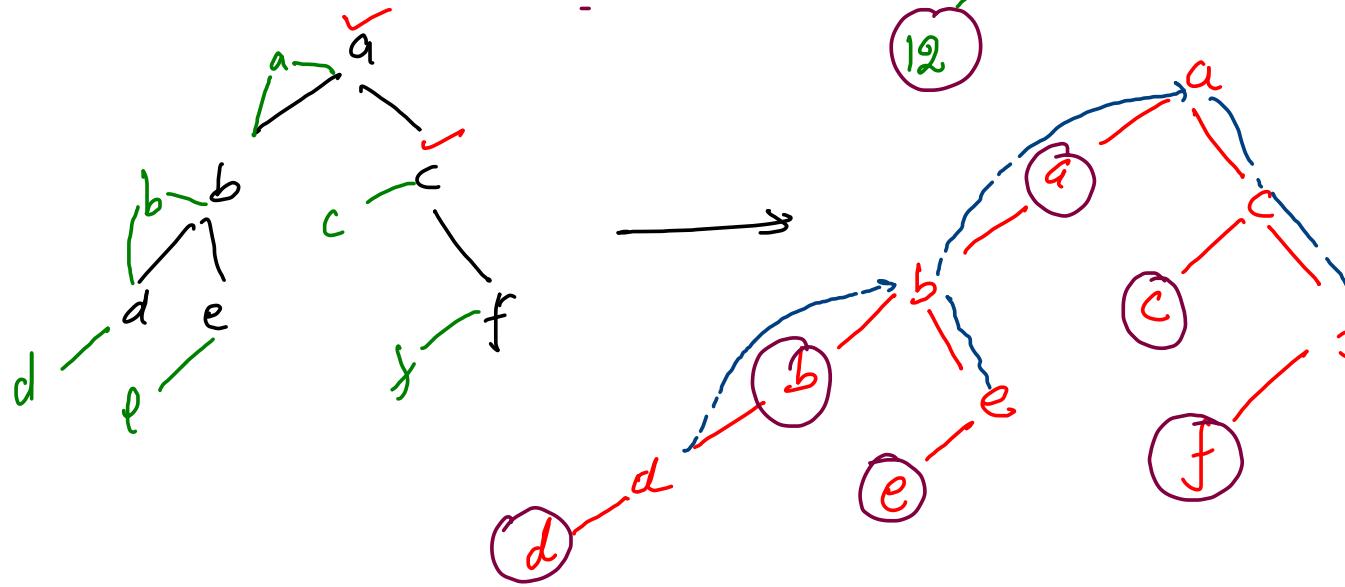
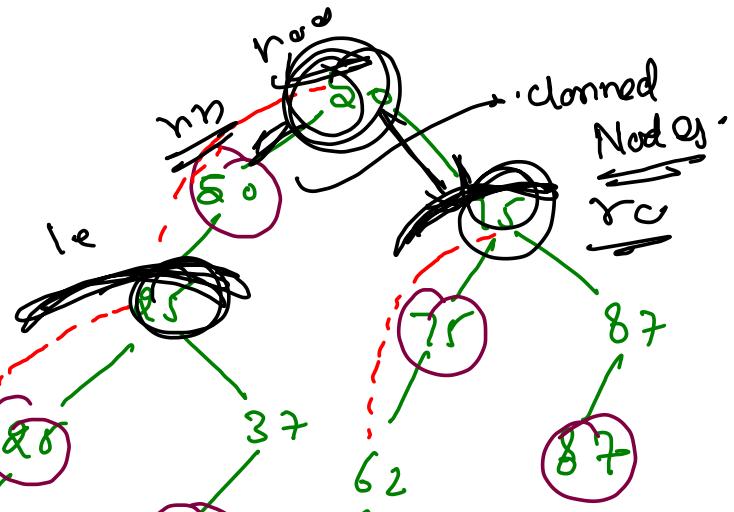
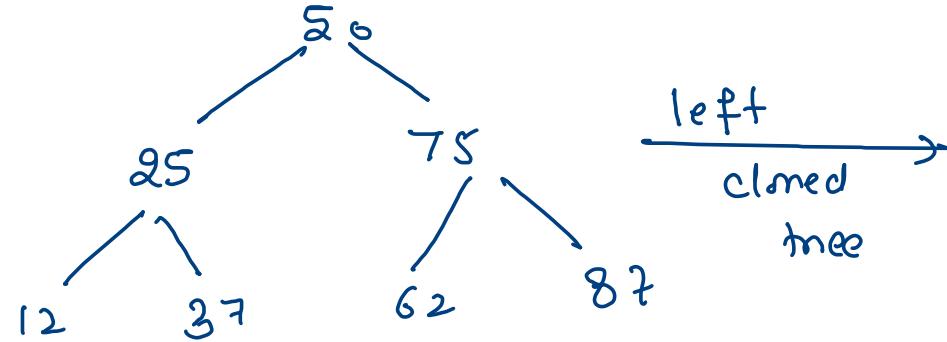
Path in Range  
 node to  
leaf



——————  
 left check  
 Right check  
 ——————

↙ 50    25    37    90  
 ↘ 50    25    87

Transform to left cloned tree

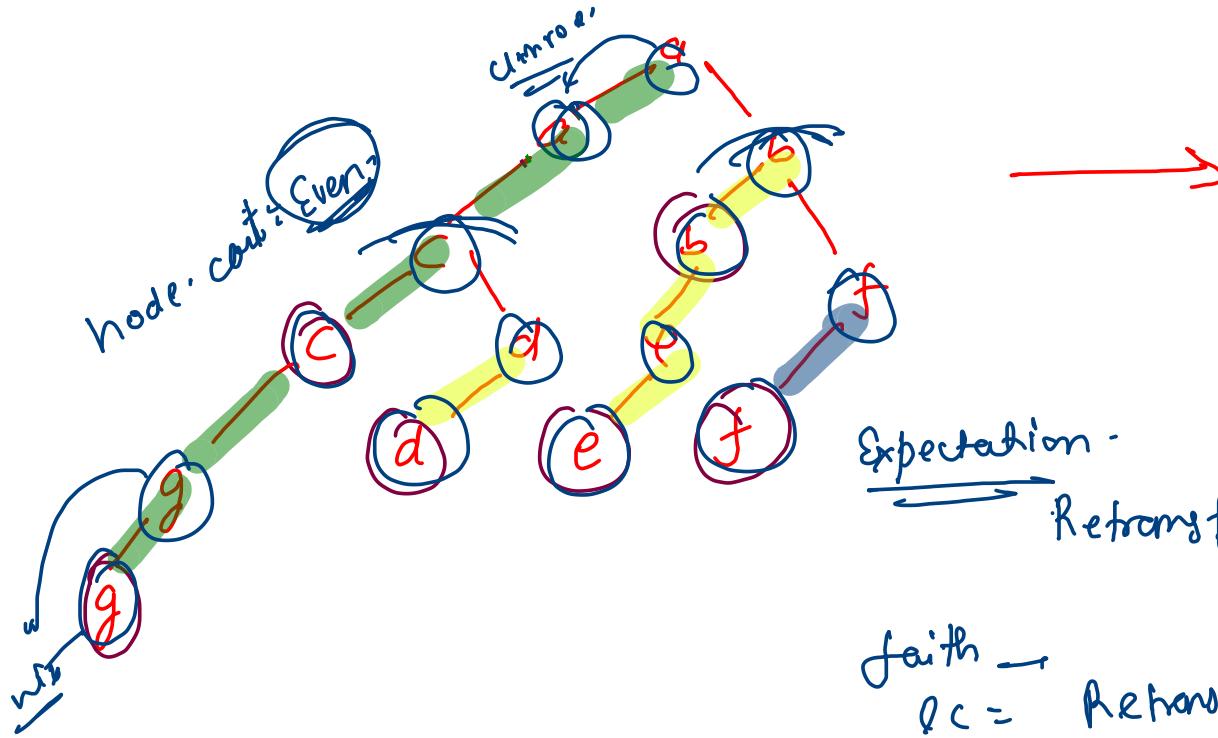


Expectation -  
transform(50)

faith = transform(25);  
rc = true for (75)

nn  
 $\text{node}.\text{left} = \text{nn}; \text{return } \text{nn}$   
 $\text{nn}.\text{left} = \text{lc}$ .

## left Cloned tree to Normal tree

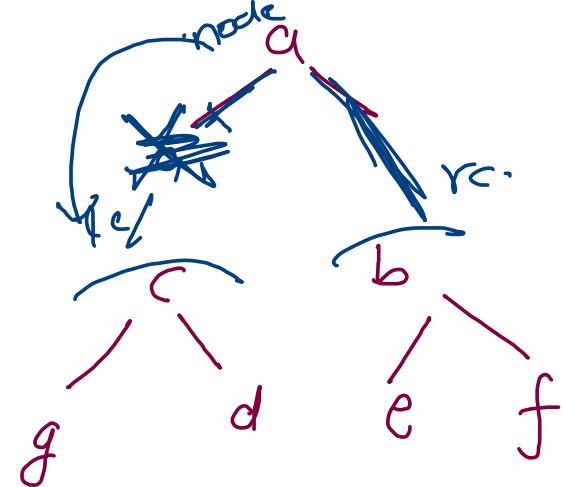


Expectation -  
Retransform (a)

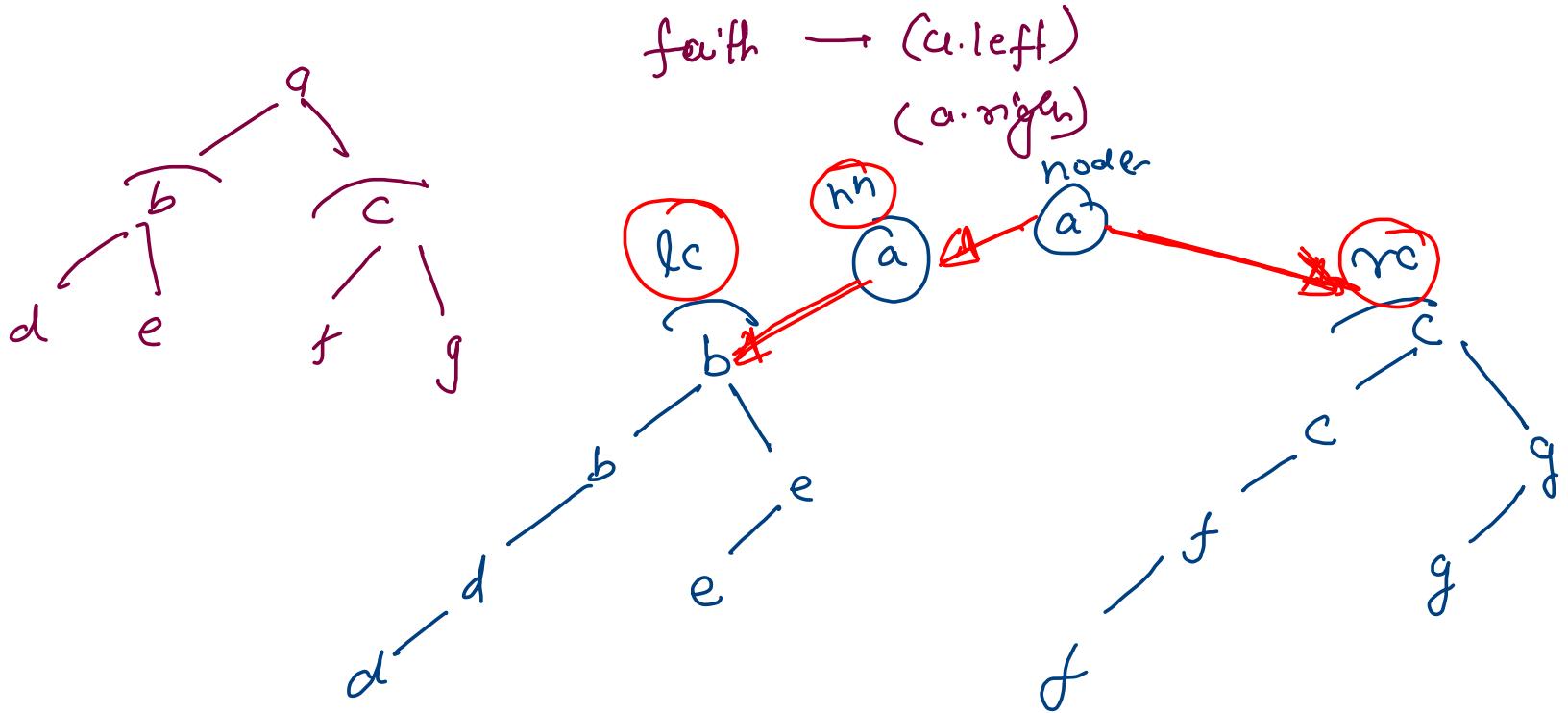
faith →  
 $lc = \text{Retransform } (\underline{\underline{a.left.left}});$

$\text{node.left} = lc;$   $\text{re} \rightarrow \text{Retransform (arright)};$   
 $\text{node.right} = rc;$

return node;



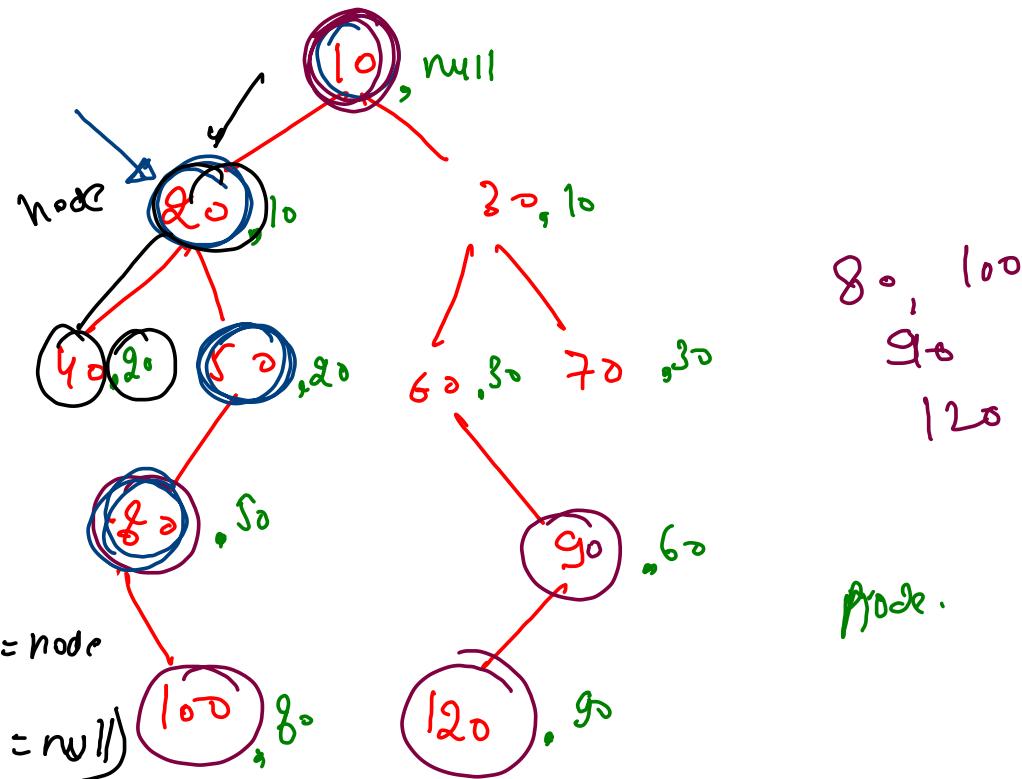
pointer Except  
Null Encouter  
X



} Node nn = new Node(node, data);  
 nn.left = lc;  
 node.left = nn;  
 node.right = rc;  
 return node;

## Print single child.

left child of my parent  
right child of my parent



(parent != null && parent.left == node  
&& parent.right == null)

node is single left child of  
parent

(parent != null && parent.right == node && parent.left == null)  
node is single right child

