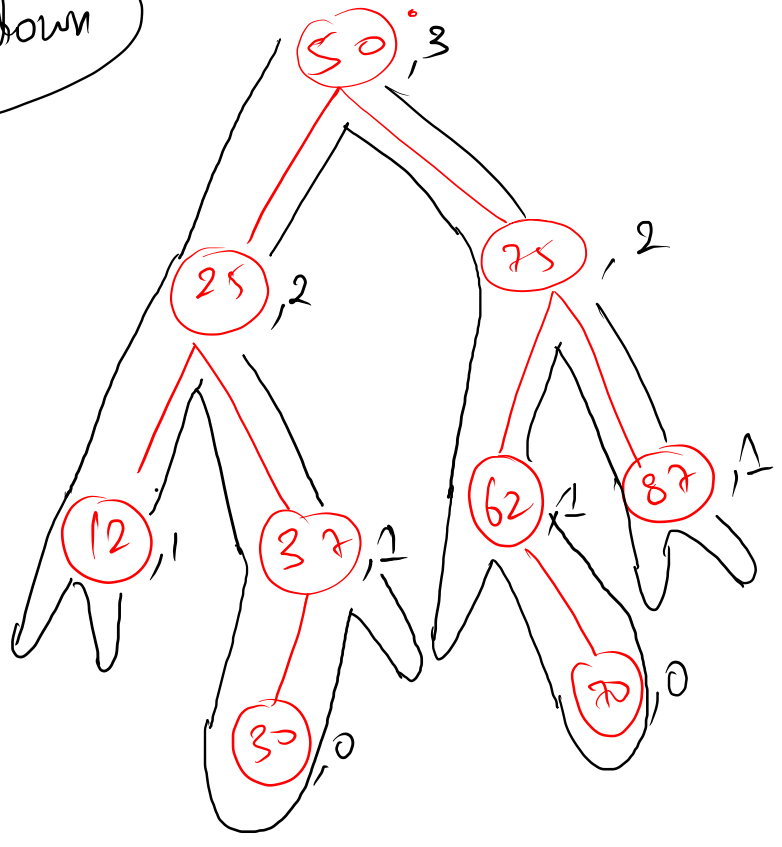
19
50 25 12 n n 37 30 n n n 75 62 n 70 n n 87 n n
3

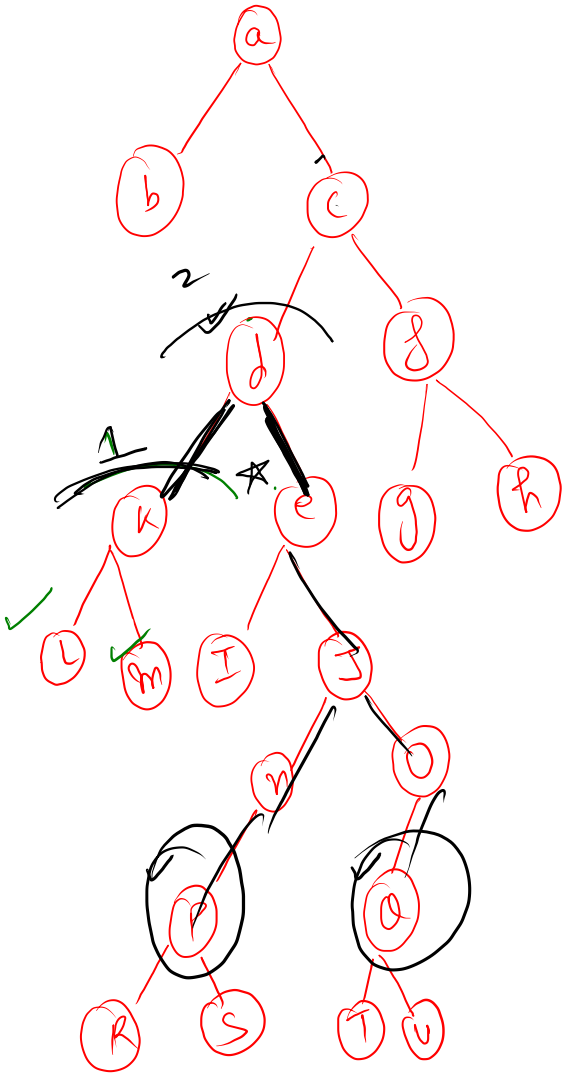
K = 3

print K levels down

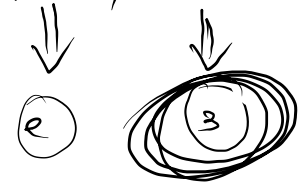
```
p s v solve(node, K){
    if(K==0){
        System.out.println(node.data);
        return;
    }
    solve(node.left, K-1);
    solve(node.right, K-1);
}
```



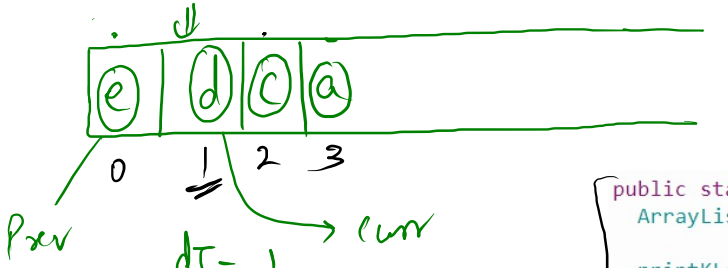
30
70



node, distance (k) — print Nodes k distance away



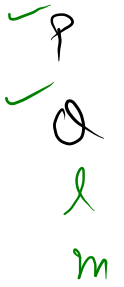
① nodeToRoot path <~~Integer~~ Nodes>



dT = 1

rD = TD − dT ⇒ 3 − 1 ⇒ 2

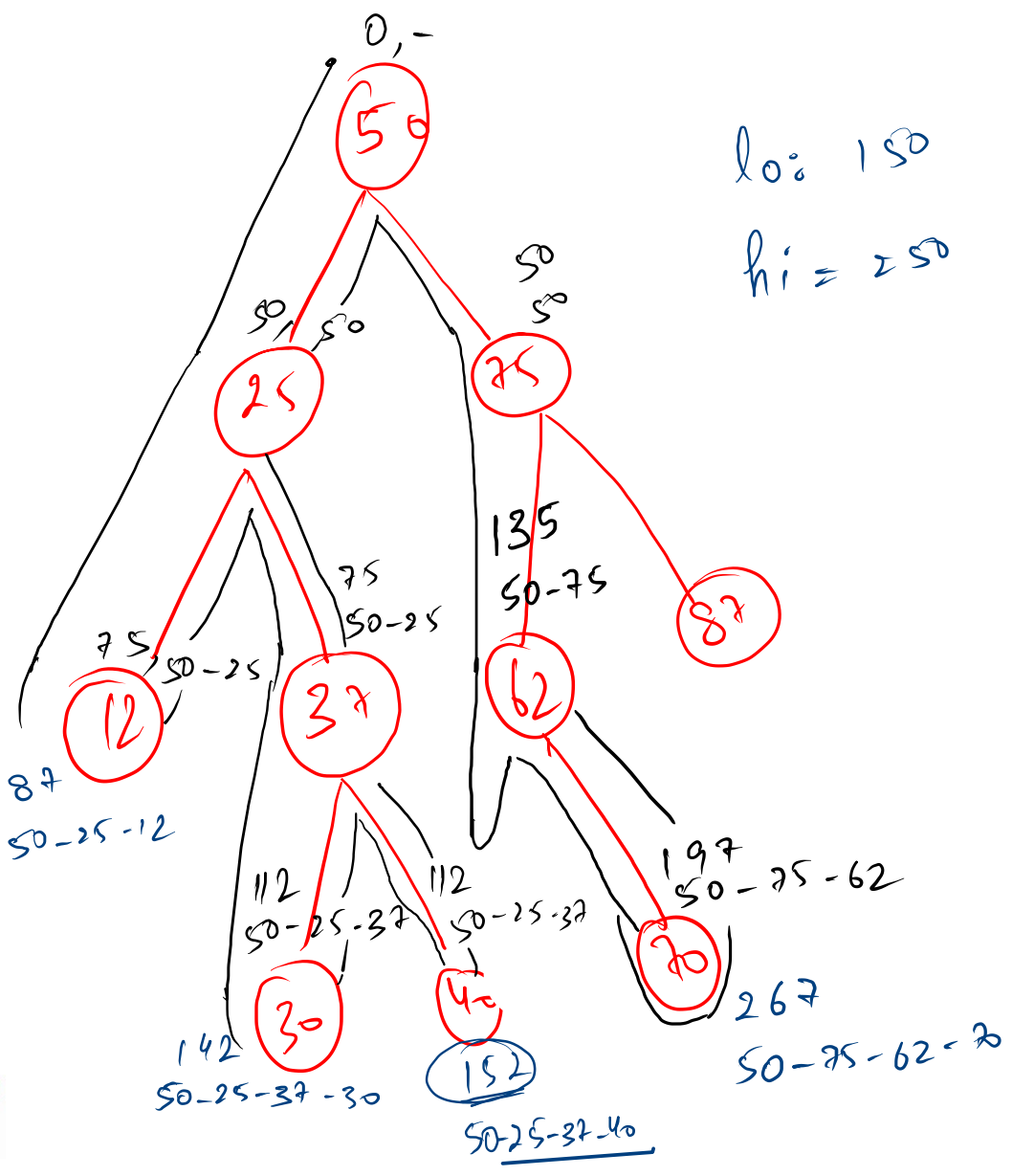
② Traversal




```java
public static void printKNodesFar(Node node, int data, int k) {
    ArrayList<Node> list = nodeToRootPath(node,data);

    printKLevelsDown(list.get(0) , k);

    for(int idx = 1 ; idx < list.size() ; idx++){
        int rdist = k - idx;
        Node curr = list.get(idx) , prev = list.get(idx-1);

        if(rdist == 0){
            System.out.println(curr.data);
            return;
        }else{
            if(curr.left == prev){
                printKLevelsDown(curr.right,rdist-1);
            }else{
                printKLevelsDown(curr.left,rdist-1);
            }
        }
    }
}
```

```java
public static void pathToLeafFromRoot(Node node, String path, int sum, in
  if(node == null){
      return;
  }
  if(node.left == null && node.right == null){
      sum += node.data;
      path += node.data;
      if(lo <= sum && 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);
}
```

50 25 37 40

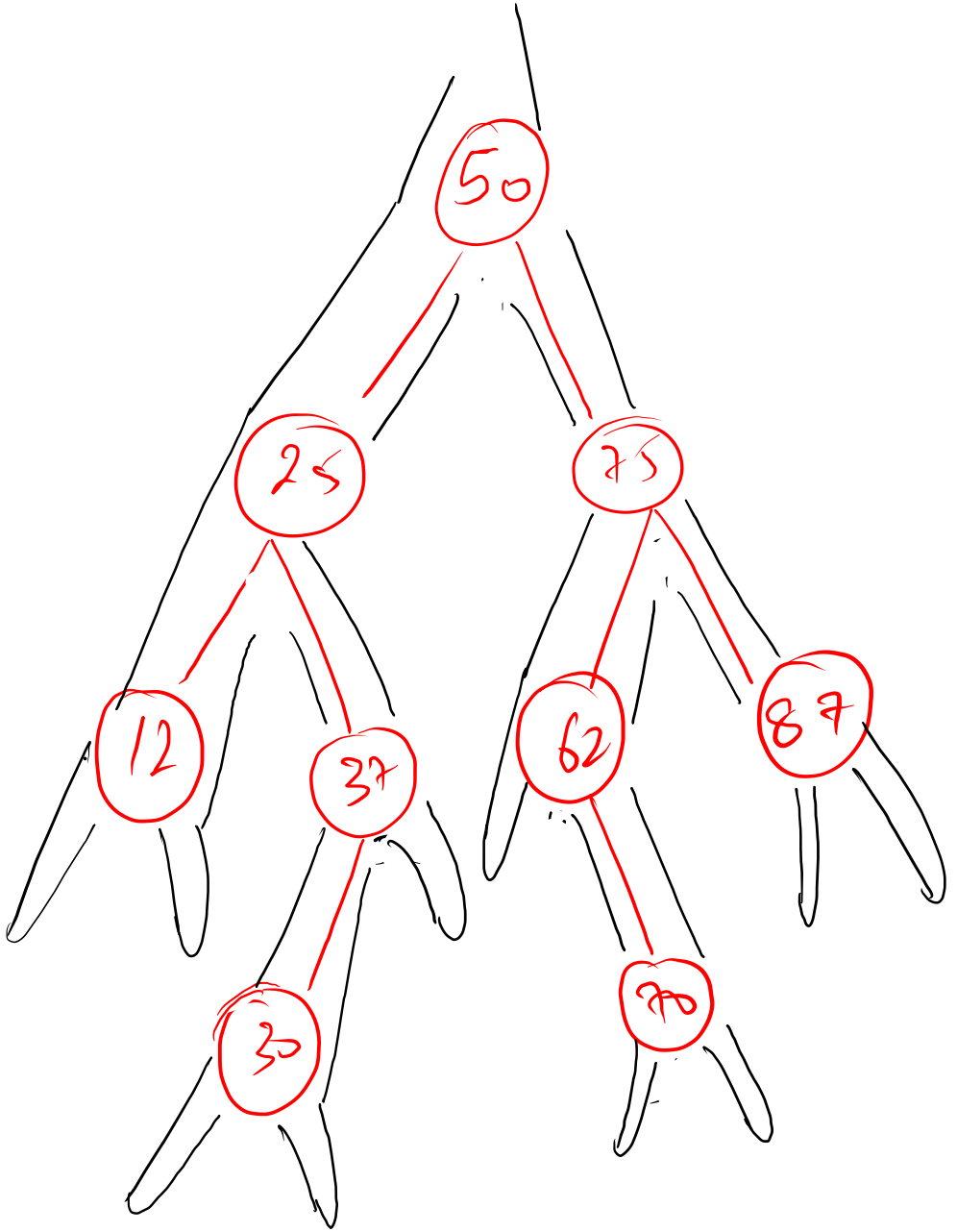



```java
pathToLeafFromRoot(root, "", 0, lo, hi);
```

```java
public static void pathToLeafFromRoot(Node node, String path, int sum, int lo, int hi){
  // write your code here
}
```

30

70




```java
public static void printSingleChildNodes(Node node){
    if(node == null){
        return;
    }
    if(node.left == null && node.right != null){
        System.out.println(node.right.data); — 1
    }else if(node.left != null && node.right == null){
        System.out.println(node.left.data);   — 2
    }

    printSingleChildNodes(node.left);        — 3
    printSingleChildNodes(node.right);       — 4
}
```