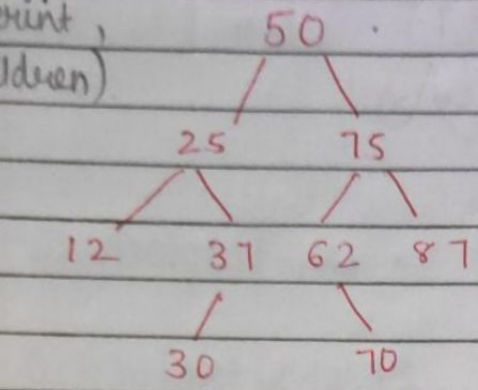
17-JAN-22

Date:
Page No.

1. SSMH.

2. levelorder Traversal Of Binary Tree.




rpa
(Remove, Print, add children)

pq | 50 | (parent queue)

cq | 25 | 75 | (child queue)

pq | 25 | 75 |

cq | 12 | 37 | 62 | 87 |

pq | 12 | 37 | 62 | 87 |

cq | 30 | 70 |

pq | 30 | 70 |

cq |

pq |

cq |

50
25 75
12 37 62 87
30 70

★ levelorder -
4 Approach
1. Parent Q, child Q.
2. Count
3. Delimitter
4. Level Pair.

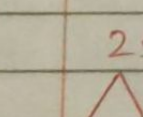
## Approach - 1 Pq, Cq

```
public static void levelOrder(Node node){

ArrayDeque<Node> pq = new ArrayDeque<>();
ArrayDeque<Node> cq = new ArrayDeque<>();

pq.add(node);
while(pq.size() >0){
        Node temp = pq.remove();
        sout(temp.data +" ");

        if(temp.left !=null){
            cq.add(temp.left);
        }

        if(temp.right !=null){
            cq.add(temp.right);
        }

        if(pq.size()==0){
            pq=cq;
            cq = new ArrayDeque<>();
            sout();
        }
}}
```

## 2nd Approach - Count.

```
public static void levelOrder2(Node node){

    ArrayDeque<Node> pq = new ArrayDeque<>();
    pq.add(node);

    while(pq.size() > 0){
        int count = pq.size();

        for(int i=0; i<count; i++){
            Node temp = pq.remove();
            sout(temp.data + " ");

            if(temp.left != null){
                pq.add(temp.left);
            }

            if(temp.right != null){
                pq.add(temp.right);
            }
        }
        sout();
    }
}
```

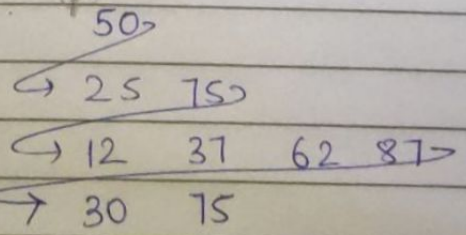
(rpa)




(remove, print, add children)

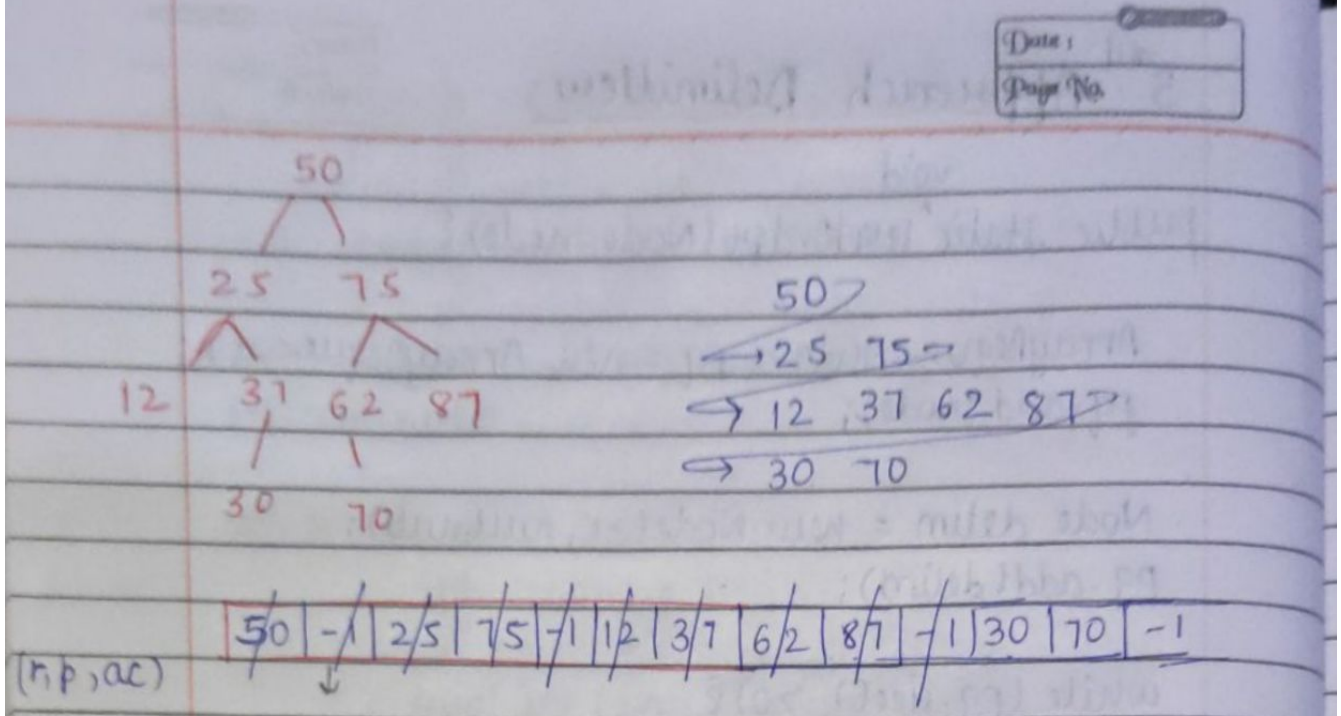
pq | 50 | 25 | 75 | 12 | 37 | 62 | 87 | 30 | 75 |

(b)



# 3rd Approach Delimitter

```
public static void levelOrder(Node node) {

    ArrayDeque<Node> pq = new ArrayDeque<>();
    pq.add(node);

    Node delim = new Node(-1, null, null);
    pq.add(delim);

    while (pq.size() > 0) {
        Node temp = pq.remove();

        if (temp.data == -1) {
            soutln();
            if (pq.size() > 0) {
                pq.add(temp);
            }
            continue; → se loop vapis upr pohoch jaata h,
                        Nichle vali chlti nhi h
        }
        sout(temp.data + " ");

        if (temp.left != null) {
            pq.add(temp.left);
        }

        if (temp.right != null) {
            pq.add(temp.right);
        }
    }
}
```


50
25 75
12 37 62 87
30 70
50
25 75
12 37 62 87
30 70
50 -1 25 75 -1 12 37 62 87 -1 30 70 -1
(r,p,ac)
level ka end signify krta h

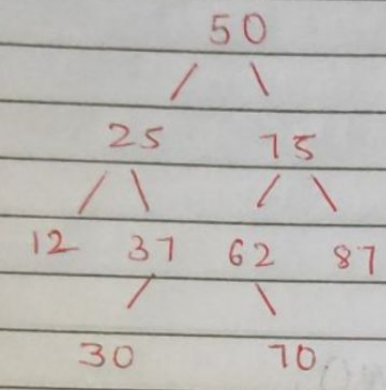
## 4th level Pair Approach.

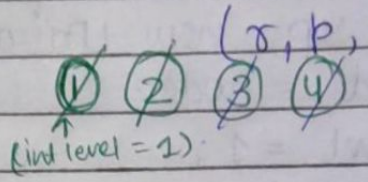
Date :
Page No.

```
static class LPair {
        Node node;
        int level;
}

public static void levelOrder(Node node) {
    ArrayDeque<LPair> q = new ArrayDeque<>();
    LPair rp = new LPair();
    rp.node = node;
    rp.level = 1;
    q.add(rp);

    int level = 1;
    while (q.size() > 0) {
        LPair temp = q.remove();

        if (temp.level > level) {
            level = temp.level;
            System.out.println();
        }

        System.out.print(temp.node.data + " ");
        if (temp.node.left != null) {
            LPair leftp = new LPair();
            leftp.node = temp.node.left;
            leftp.level = temp.level + 1;
            q.add(leftp);
        }
```

```
if (temp.node.right != null) {
    LPair.rightp = new LPair();
    rightp.node = temp.node.right;
    rightp.level = temp.level + 1;
    q.add(rightp);
}
```

33.




| $50^1$ | $25^2$ | $75^2$ | $12^3$ | $37^3$ | $62^3$ | $87^3$ | $30^4$ | $70^4$ |
|---|---|---|---|---|---|---|---|---|

50

25 75

12 37 62 87

30 70

Date :
Page No.

## 3. FIND and NODEtoRootPath fn Binary Tree.

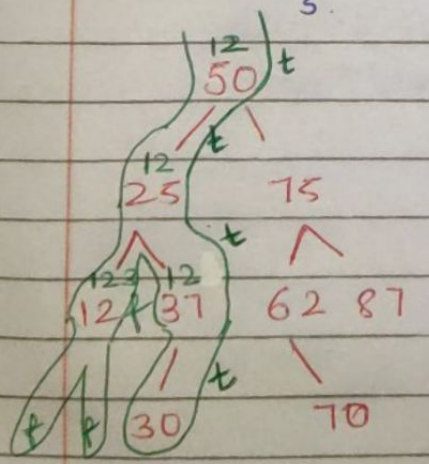
### Find -

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

    boolean fil = find(node.left, data);
    if (fil == true) {
        return true;
    }

    boolean fir = find(node.right, data);
    if (fir == true) {
        return true;
    }
    return false;
}
```

1 — base case + node match

2 — found in left

3 — found in right

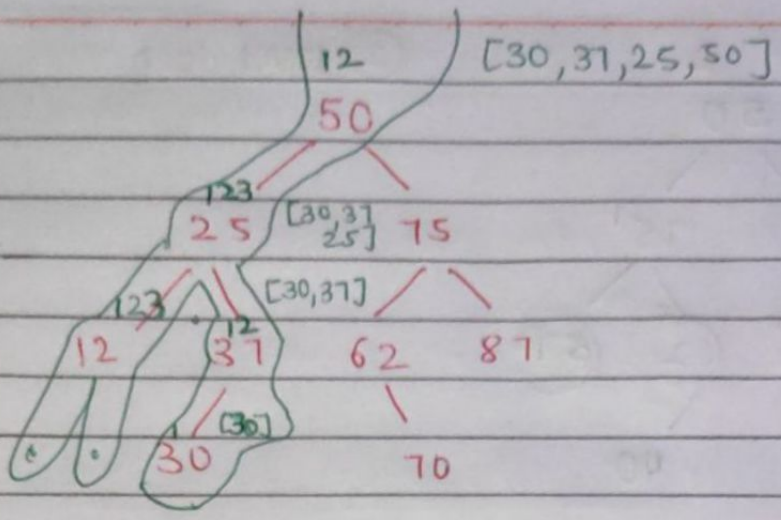


[Desired Node se Root Node tk ka path return in AL]

Date :
Page No.

## Nodetorootpath -

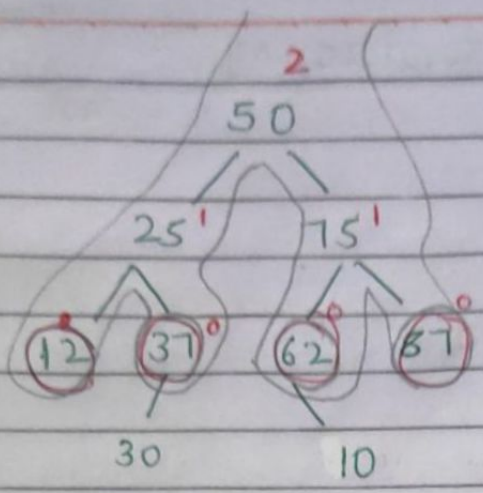
```
public static ArrayList<Integer> nodeToRootPath
                (Node node, int data) {

    if(node == null){
        return new ArrayList<>();
    }

    if(node.data == data){                                    ①
        ArrayList<Integer> list = new ArrayList<>();
        list.add(node.data);
        return list;
    }

                                                    data);
    ArrayList<Integer> llist = nodeToRootPath(node.left,     ②
    if(llist.size() > 0){
            llist.add(node.data);
            return llist;
    }

                                                    data);    ③
    ArrayList<Integer> rlist = nodeToRootPath(node.right,
    if(rlist.size() > 0){
            rlist.add(node.data);
            return rlist;
    }

    return new ArrayList<>();
}
```




## 4. Print K Level Down - [Print element at 2 level down from root]

```
public static void printKLevelDown(Node node, int K){
    if(node == null || K<0){
        return;
    }
    if(K==0){
        sout(node.data);
        return;
    }

    printKLevelDown(node.left, K-1);
    printKLevelDown(node.right, K-1);
```

| I/P | O/P |
|---|---|
| 19 | 30 |
| 50 25 12 nn 37 30 nn 75 62 n 70 n n 87 n n. | 70 |
| 3 | |




(2) level deep

o/p → 12 37 62 87.

IMPORTANT

## 5. Print Nodes K Distance Away -

→ [Nodetoroot path + K level down].

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 50 | e | b | a | h' |

← NodeToRootPath in AL form.

(50 se 4 down vle)

NOW
K down [50 se 4 down].

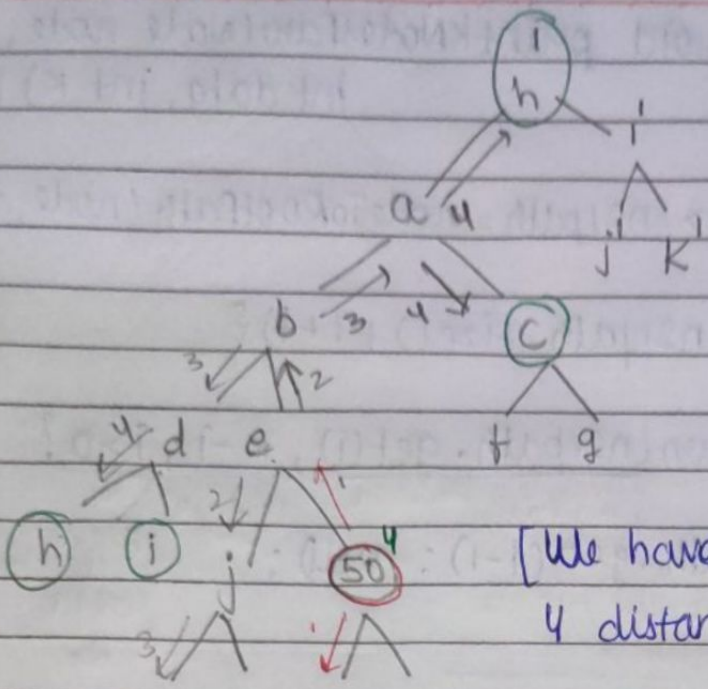
| 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| n | 50 | e | b | a |

(50 ke wale ko print) →

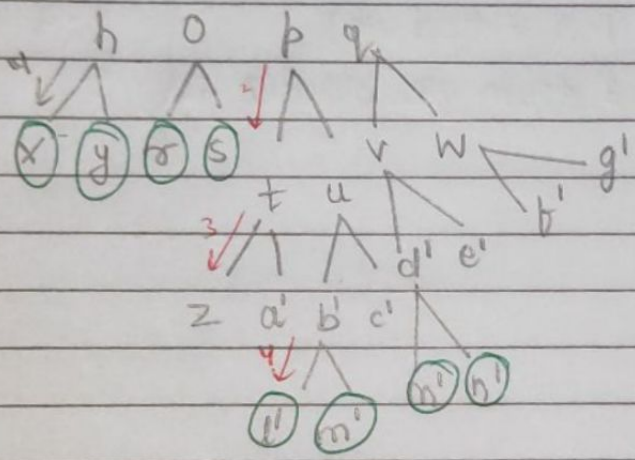
(Works as blocker)

(e se 3 down vle. print hone chiye)

TC = k.n




[We have to print nodes that are 4 distance away from 50].




50 - 4 down
e - 3 down but not on 50.
b - 2 down but not on e
a - 1 down but not on b
h' - 0 dow but not on a

```
public static void printKNodesFar(Node node,
                                  int data, int K) {

    ArrayList<Node> n2rpath = nodeToRootPath(node, data);

    for(int i = 0; i < n2rpath.size(); i++) {

        printKLevelDown(n2rpath.get(i), K - i, i > 0 ?
                        n2rapath.get(i-1) : null);
    }
}
```

0th node pass kiya, K-0 (K far), i > 0 ? → piche vla uska blocker bn jaega

i th pass

i ke liye i-1th vla blocker h

3.

(0) 50 ko bola K-0 (4-0, 4 dur vle)

(1) e ko bola K-1 (3 down vle blocker so

e ke liye so blocker h.