Today's agenda

- ArrayList
- Size of the tree
- Sum of all nodes
- level order
- reverse level order

AlgoPrep

ArrayList → nothing but array → dynamic Array.

List<Integer> ls = new ArrayList<>(); (ls: name)

ArrayList<Integer> ls = new ArrayList<>(); (ls: name)

↓

ls

0	1	2
10	20	10

ls.add(10); → O(1)
ls.add(20);
ls.add(10);

→ ls.get(1); → 20 → O(1)

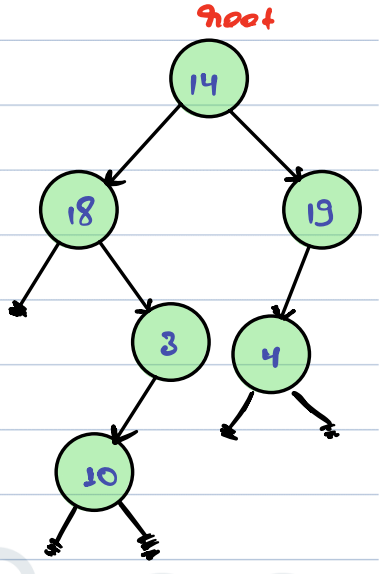
→ ls.size(); → 3 → O(1)

ls.remove(idx) → last idx → O(1)
→ oth idx → O(n)

AlgoPrep

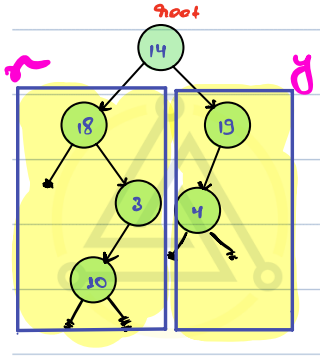
Q) Size of a tree

↳ Given a tree, calculate no. of nodes in it.



Faith: Given root node, find and return no. of node.

Main logic:



→ x + y + 1

Base Case:

```
if (root == null) { return 0; }
```

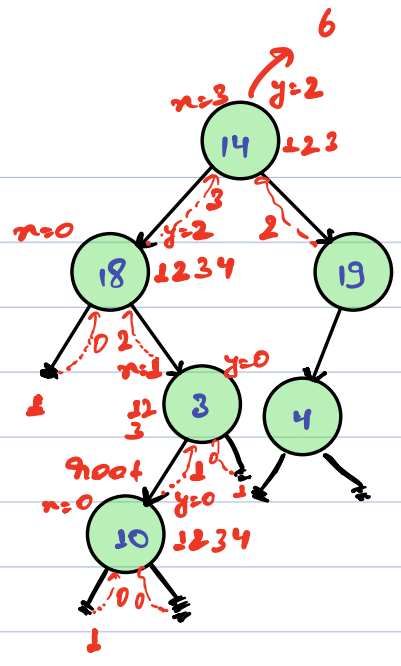
T.C: O(N)

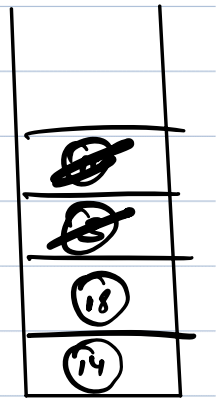
S.C: O(N)

```
int size (Node root){
    if (root == null) { return 0; }
    int x = size(root.left);
    int y = size(root.right);
    return x+y+1;
}
```

AlgoPrep

```
int Size (node root){
1  if (root == null){return 0;}
2  int x = Size(root.left);
3  int y = Size (root.right);
4  return x+y+1;
}
```



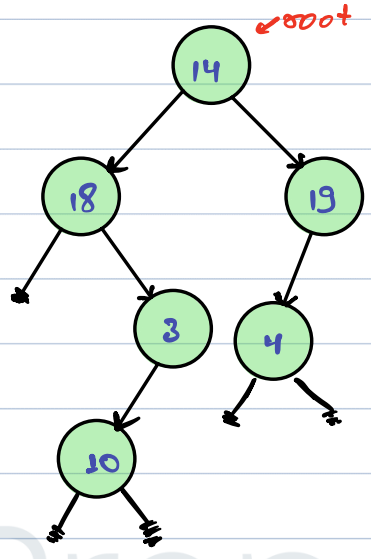


S.C: O (no. of levels) = O(N)

AlgoPrep

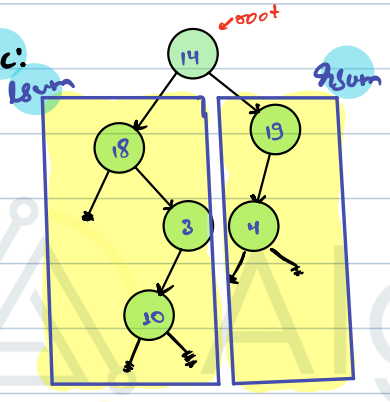
Q) Sum of a tree

↳ Given a tree, calculate sum of all nodes data in it.



Faith: Given root node, find and return sum of node.

Main logic:



→ lsum + rsum + root.val

Base case:

```
if(root == null){return 0;}
```

//Pseudo code

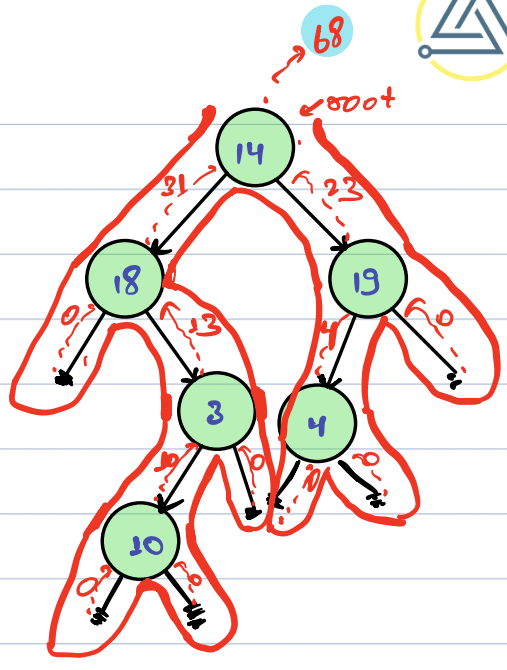
```
int Sum(Node root){
    if(root == null){return 0;}

    int lsum = Sum(root.left);
    int rsum = Sum(root.right);

    return lsum + rsum + root.val;
}
```

T.C: O(N)

S.C: O(N)

AlgoPrep

```
int Sum (Node root){
  if (root == null){return 0;}

    int lSum = Sum (root.left);
    int rSum = Sum (root.right);

      return lSum + rSum + root.val;

}
```

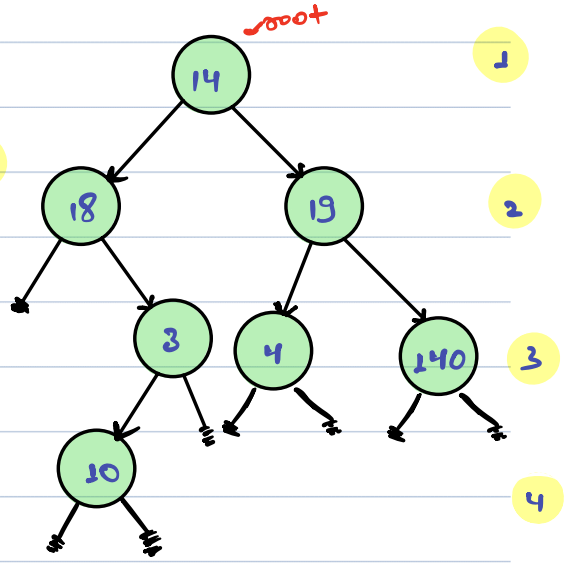


AlgoPrep

Break till 9:23 PM

Pre (NLR) In (LNR) Post (LRN)

Q) Level order of tree



level order: 14 18 19 3 4 140 10

<Node> Q;

rem: 10

Output:

14 18 19 3 4 140 10

AlgoPrep

//Pseudo code

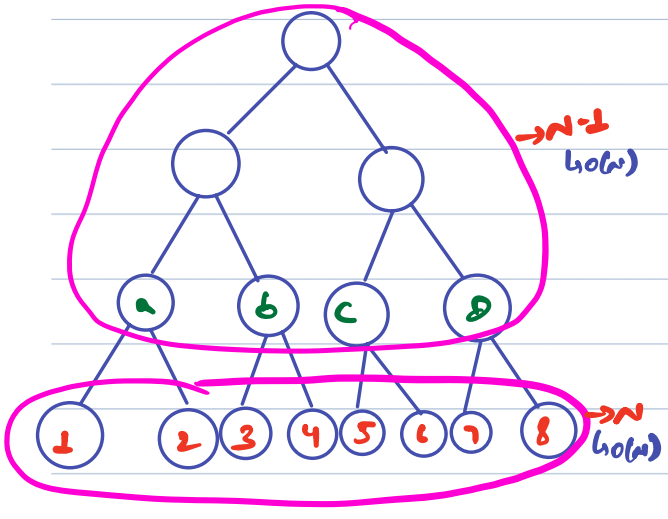
```
void levelorder (Node root) {
    Queue<Node> q;
    q.add(root);
    while (q.size()>0) {
        Node rem = q.remove();
        System.out.print(rem.val);
        if(rem.left!=null){ q.add(rem.left);}

        if(rem.right!=null){q.add(rem.right);}
    }
}
```

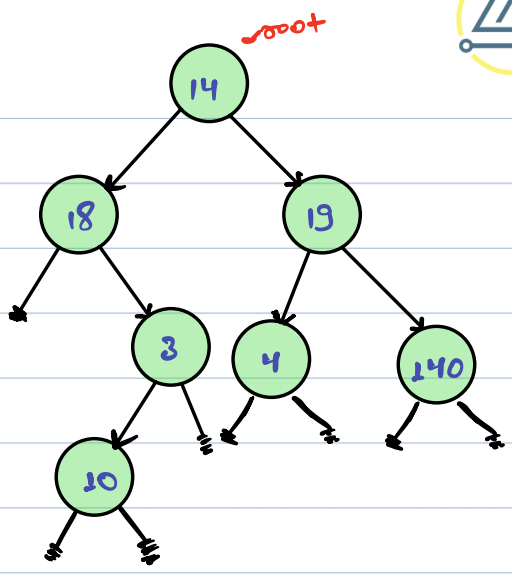
T.C: O(N)

S.C: O(N)

Total no. of nodes : $2n-1$

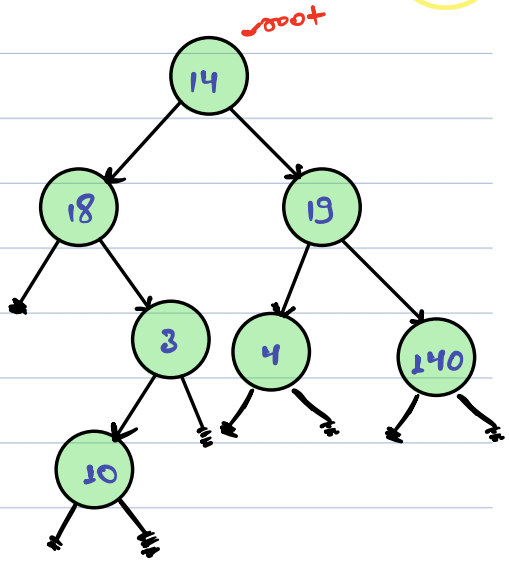
$\Rightarrow O(n)$





AlgoPrep

Q) Level order of tree 2



14

18 19

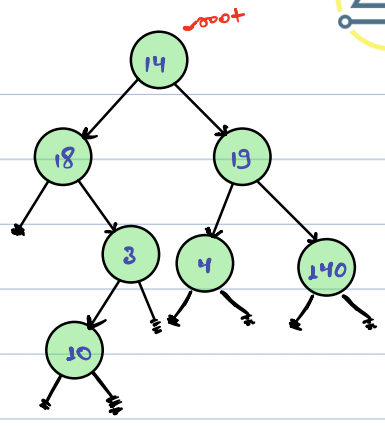
3 4 140

10

IIPSuedo code

```
void levelorder (Node root ) {
    Queue<Node> q;
    q.add(root);

    while ( q.size() >0){
        int n= q.size();
        for(int i=1; i<=n; i++ ) {
          Node rem = q.remove();
          System.out.print (rem.val);
          if(rem.left != null ){ q.add(rem.left);}

          if(rem.right != null){ q.add (rem.right);

        }

        System.out.println();
    }
}
```

AlgoPrep




```
Queue<Node> q;
q.add(root);

while (q.size() > 0){
    int n = q.size();
    for(int i=1; i<=n; i++){
        Node rem = q.remove();
        System.out.print(rem.val+" ");
        if(rem.left != null){ q.add(rem.left);}

        if(rem.right != null){ q.add(rem.right);
    }
    System.out.println();
}
```

q:

n: 1

rem: 10

14

18 19

3 4 140

10

→

1 → 1

2 → n+1

3 → n2

4 → i

5 → i

. → i

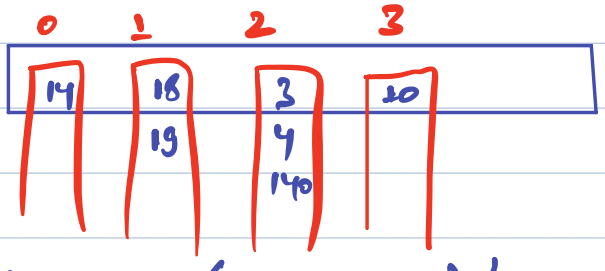
. → i

n → n

O(no. of nodes)

→ List<List<Integer>> ans;

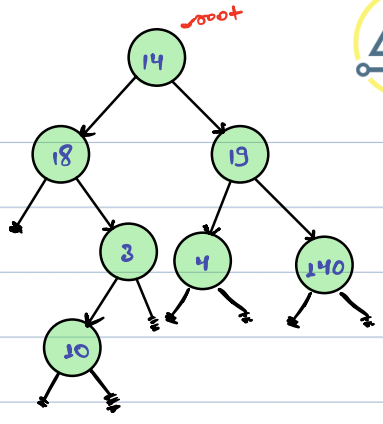
ans:




```
List<List<>>
void levelorder (Node root ) {
    Queue<Node> q;
    q.add(root);
    List<List<Integer>> ans = new ArrayList<>();
    while (q.size()>0){
        int n= q.size();
        List<Integer> temp = new ArrayList<>();
        for(int i=1; i<=n; i++ ) {
            Node rem = q.remove();
            temp.add(rem.val);
            if(rem.left!=null){ q.add(rem.left);}

            if(rem.right!=null){q.add(rem.right);

        }
            ans.add(temp);
    }
    return ans;
}
```

AlgoPrep

```
List<List<>> void levelOrder(Node root){
    Queue<Node> q;
    q.add(root);
    List<List<Integer>> ans = new ArrayList<>();
    while(q.size()>0){
        int n= q.size();
        List<Integer> temp = new ArrayList<>();
        for(int i=1; i<=n; i++){
            Node rem = q.remove();
            temp.add(rem.val);
            if(rem.left!=null){q.add(rem.left);}
            if(rem.right!=null){q.add(rem.right);}
        }
        ans.add(temp);
    }
}
```



q: 14 18 19 3 4 140

n = 2

temp: 18 19

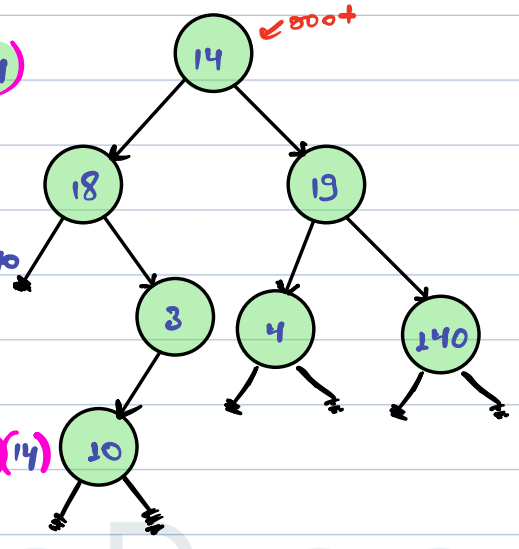
ans: 14 | 18 19

AlgoPrep

Q) Reverse level order

↳ Print the level order from last level to first.

output: (10) (3 4 140) (18 19) (14)



↳ level order: 14 18 19 3 4 140 10

↓

(10) (140 4 3) (19 18) (14)

AlgoPrep

R-L level order

↳ 14 19 18 140 4 3 10

↓ reverse

(10) (3 4 140) (18 19) (14)