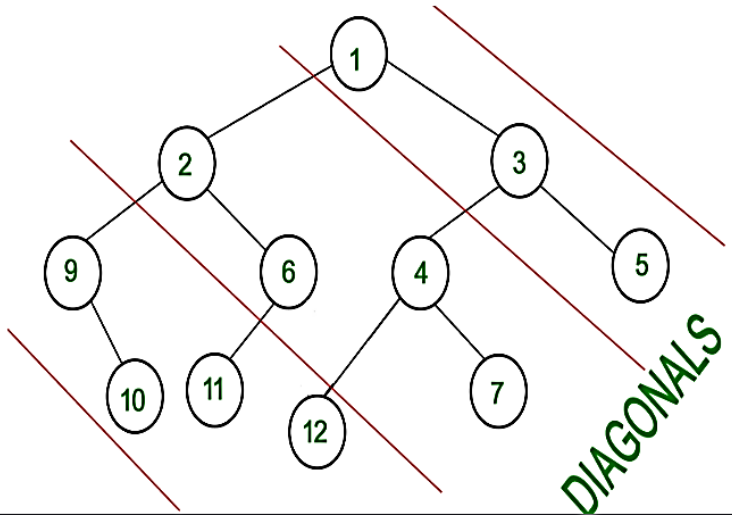
PROBLEM

Diagonal sum in binary tree

Medium Accuracy: 61.89% Submissions: 37K+ Points: 4

Consider Red lines of slope -1 passing between nodes (in following diagram). The diagonal sum in a binary tree is the sum of all node datas lying between these lines. Given a Binary Tree of size n, print all diagonal sums.

For the following input tree, output should be 9, 19, 42.
9 is sum of 1, 3 and 5.
19 is sum of 2, 6, 4 and 7.
42 is sum of 9, 10, 11 and 12.



Example 1:

Input:

```
        4
      /   \
     1     3
          /
         3
```

Output:

7 4

Example 2:

Input:

```
          10
        /    \
       8      2
      / \    /
     3   5  2
```

Output:

12 15 3

Your Task:

You don't need to take input. Just complete the function **diagonalSum()** that takes root **node** of the tree as parameter and returns an array containing the diagonal sums for every diagonal present in the tree with slope **-1**.

Expected Time Complexity: O(nlogn).

Expected Auxiliary Space: O(n).

Constraints:

$1 <= n <= 10^5$

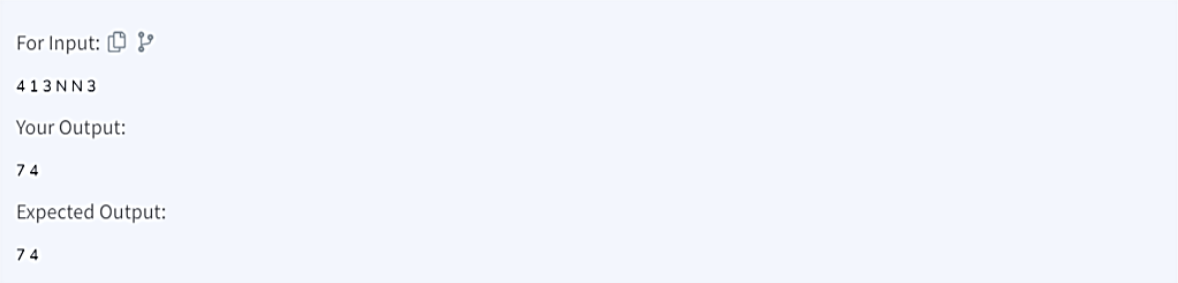
$0 <= \text{data of each node} <= 10^4$

CODE

```
/*Complete the function below
Node is as follows:
class Node{
   int data;
   Node left,right;
   Node(int d){
      data=d;
      left=right=null;
   }
}
*/
class Tree {
   public static ArrayList <Integer> diagonalSum(Node root)
   {
      // code here.
        ArrayList <Integer>arr=new ArrayList <Integer>();
      Queue<Node>q=new LinkedList<>();


      if(root==null)return arr;
      q.add(root);
      while(!q.isEmpty()){
         int n=q.size();
         int sum=0;
         for(int i=0;i<n;i++){
            Node curr=q.remove();
            while(curr!=null){
               sum+=(curr.data);
               if(curr.left!=null){
                  q.add(curr.left);
               }
               curr=curr.right;

            }

         }
```

```
            arr.add(sum);
        }
        return arr;
    }
}
```

OUTPUT



EXPLANATION

- The method diagonalSum takes a Node called root as input and returns an ArrayList<Integer> containing the diagonal sums of the binary tree.
- Inside the method, an ArrayList<Integer> called arr is initialized to store the diagonal sums.
- A Queue<Node> called q is created using a LinkedList. This queue will be used for level order traversal of the binary tree.
- If the root is null, meaning the tree is empty, the method returns the empty arr.
- Otherwise, the root is added to the queue q.
- A while loop is used to traverse the tree level by level until the queue becomes empty.
- Inside the loop, the size of the queue (n) is stored to represent the number of nodes at the current level.
- Another loop runs n times to process each node at the current level.
- For each node removed from the queue (curr), a while loop traverses the diagonal starting from that node.
- Inside the diagonal traversal, the data of each node encountered is added to the sum.
- If there is a left child of the current node, it is added to the queue for processing in the next level.
- The current node is then updated to its right child to continue diagonal traversal.
- After processing all nodes at the current level, the sum of that level is added to the arr.
- Finally, the arr containing all the diagonal sums is returned.