https://course.acciojob.com/idle?question=ac71e808-aca8-4148-8e01-5b1c2422ed63

- EASY
- Max Score: 30 Points

LCA of Binary Search Tree

Given the `root` of a binary search tree, and two integers `P` and `Q`. Find the least common ancestor of `P` and `Q`.

Input Format

Here you are given an array as an input and using that array we will make a binary search tree.

The first line of input contains the number of Nodes `N` and two integers `P` and `Q`.

The second line of input contains the value of each node.

Output Format

Return the node which is LCA of P and Q in the given tree.

Example 1

Input

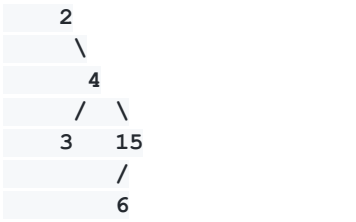
```
5 4 15
2 4 15 6 3
```

Output

```
4
```

Explanation

```
The tree looks like this:
```




```
The LCA 4 and 15 is 4.
```

Example 2

Input

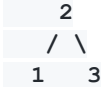
```
3 1 3
2 1 3
```

Output

```
2
```

Explanation

```
The tree looks like this:
```




```
The LCA 1 and 3 is 2.
```

Constraints

1 <= Number of nodes <= 1000

1<= value of each node <= 10000

Topic Tags

- **BST**

My code

```
// n java
import java.util.*;
class Node {
    int data;
    Node left, right;
    public Node(int item)
    {
        data = item;
        left = right = null;
    }
}

class BinarySearchTree
{
    Node constructBST(int[]arr,int start,int end,Node root)
    {
        if(start>end)
            return null;
        int mid=(start+end)/2;

        if(root==null)
```

```
            root=new Node(arr[mid]);

        root.left=constructBST(arr,start,mid-1, root.left);
        root.right=constructBST(arr,mid+1,end, root.right);

        return root;

    }
}

public class Main {
    public static void main(String[] args) throws Throwable {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int p = sc.nextInt();
        int q = sc.nextInt();
        int arr[]=new int[n];
        for (int i = 0; i < n; i++)
        {
            arr[i] = sc.nextInt();
        }

        Arrays.sort(arr);
        Node root=null;
        BinarySearchTree bst=new BinarySearchTree();
        root=bst.constructBST(arr,0,n-1,root);

        Solution Accio = new Solution();
        Node ans=Accio.LCA(root,p,q);
        System.out.println(ans.data);
```

```
        sc.close();

    }
}

class Solution
{
        static Node fun(Node node, int n, int m)
        {
                if(node ==null)
                        return null;

                if(node.data ==n || node.data ==m)
                {

        //    Node b=fun(node.left,n,m);
           // Node c=fun(node.right,n,m);
               // if(b!=null ||c!=null)
           return node;
               }
               Node b=fun(node.left,n,m);
               Node c=fun(node.right,n,m);
                if(c!=null && b!=null)
                        return node;
                if(b!=null)
                return b;
                return c;
        }
    Node LCA(Node node, int n1, int n2)
    {
```

```
        // Your code here
            return fun(node,n1,n2);
    }
}
```