https://course.acciojob.com/idle?question=5c814944-2552-47b2-999d-e85df0cba97f

- MEDIUM
- Max Score: 40 Points

Recover BST

You are given the root of a binary search tree (BST), where the values of exactly two nodes of the tree were swapped by mistake.

Your task is to recover the tree without changing its structure.

Note

You are Not Allowed To Use Extra Space.

Input Format

Input is managed for you. (The input given in the example is the preorder traversal of the binary search tree.)

Output Format

Output is managed for you.

Example 1

Input

```
9
4 2 3 -1 -1 -1 5 -1 -1
```

Output

```
3 -> 4 <- 5
2 -> 3 <- .
. -> 2 <- .
. -> 5 <- .
```

Explanation

You need to complete the function. Input and Output are handled by driver code.

Example 2

Input

```
13
7 3 2 -1 -1 10 -1 -1 5 -1 12 -1 -1
```

Output

```
3 -> 7 <- 10
2 -> 3 <- 5
. -> 2 <- .
. -> 5 <- .
. -> 10 <- 12
. -> 12 <- .
```

Explanation

You need to complete the function. Input and Output are handled by driver code.

Constraints

0 <= Number of Nodes <= 10^9

-10^9 <= value of Node data <= 10^9

Topic Tags

- **BST**

My code

```
// n java
import java.util.Scanner;

public class Main {
    public static Scanner scn = new Scanner(System.in);

    public static class TreeNode {
        int val = 0;
        TreeNode left = null;
        TreeNode right = null;

        TreeNode(int val) {
            this.val = val;
        }
    }
        public static void inOrder(TreeNode root){
                if(root==null) return;
         //LEFT
                 inOrder(root.left);
                // work
                if(first==null && prev.val>root.val)
                        first=prev;
                if(first!=null && prev.val>root.val)
                        second=root;

                 prev=root;
```

```
            // right
            inOrder(root.right);
        }
    static TreeNode first=null;
        static TreeNode second=null;
        static TreeNode prev= new TreeNode(Integer.MIN_VALUE);
    public static void recoverTree(TreeNode root) {
        //Write code here
            inOrder(root);
            int temp=first.val;
            first.val=second.val;
            second.val=temp;
    }

    //
input_section=========================================
========

    public static void display(TreeNode node) {
        if (node == null)
            return;

        StringBuilder sb = new StringBuilder();
        sb.append((node.left != null ? node.left.val : "."));
        sb.append(" -> " + node.val + " <- ");
        sb.append((node.right != null ? node.right.val : "."));

        System.out.println(sb.toString());

        display(node.left);
```

```
    display(node.right);

}

public static TreeNode createTree(int[] arr, int[] IDX) {
    if (IDX[0] > arr.length || arr[IDX[0]] == -1) {
        IDX[0]++;
        return null;
    }

    TreeNode node = new TreeNode(arr[IDX[0]++]);
    node.left = createTree(arr, IDX);
    node.right = createTree(arr, IDX);

    return node;
}

public static void solve() {
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++)
        arr[i] = scn.nextInt();

    int[] IDX = new int[1];
    TreeNode root = createTree(arr, IDX);
    recoverTree(root);
    display(root);
}

public static void main(String[] args) {
```

```
        solve();
    }
}
```