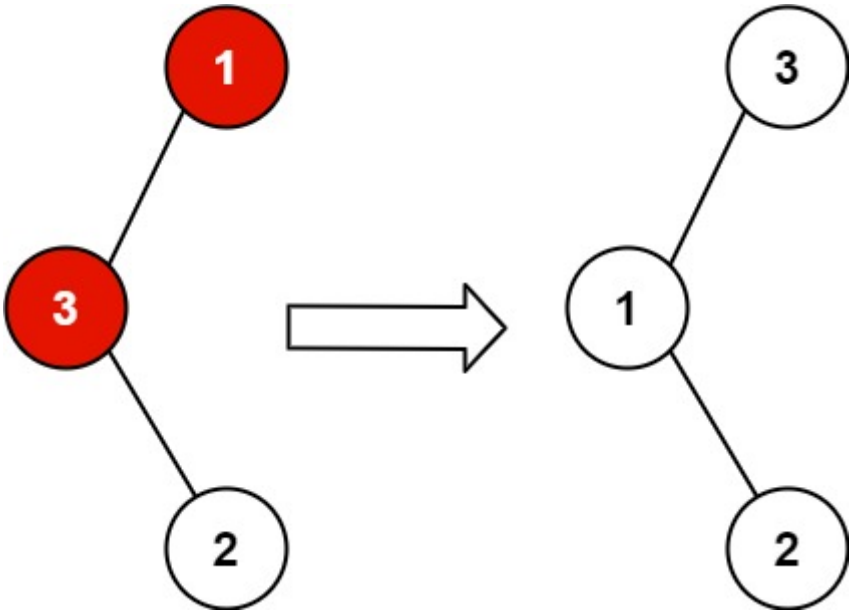
99. Recover Binary Search Tree

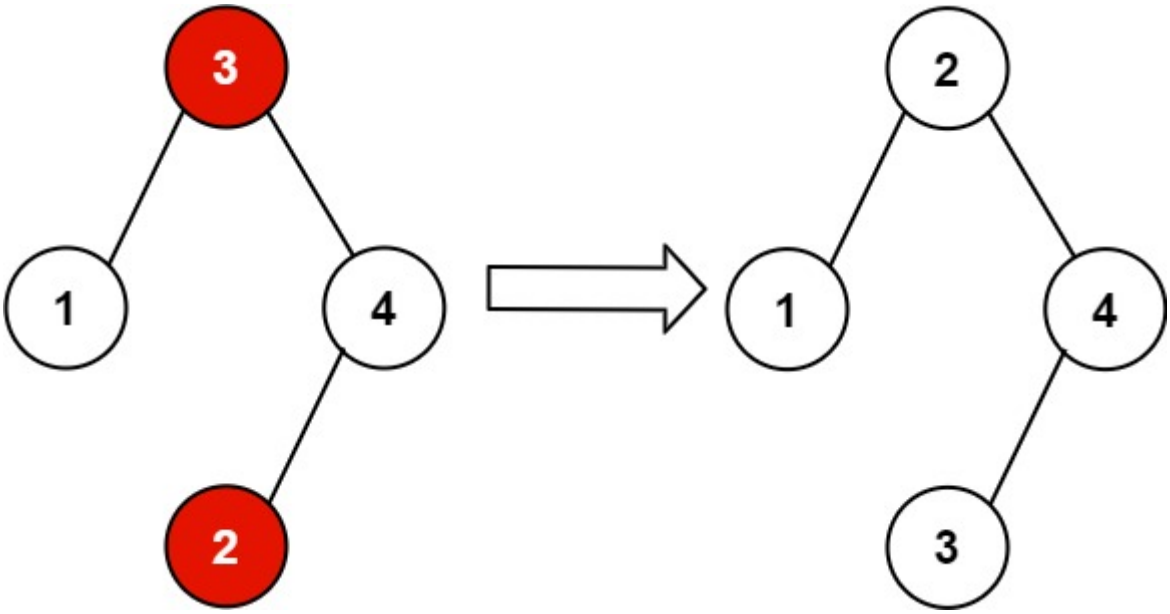
You are given the `root` of a binary search tree (BST), where the values of **exactly** two nodes of the tree were swapped by mistake. *Recover the tree without changing its structure*.

Example 1:




```
Input: root = [1,3,null,null,2]
Output: [3,1,null,null,2]
Explanation: 3 cannot be a left child of 1 because 3 > 1. Swapping 1 and 3 makes the BST valid.
```

Example 2:



```
Input: root = [3,1,4,null,null,2]
Output: [2,1,4,null,null,3]
Explanation: 2 cannot be in the right subtree of 3 because 2 < 3. Swapping 2 and 3 makes the BST valid.
```

Constraints:

- The number of nodes in the tree is in the range `[2, 1000]`.
- `-2<sup>31</sup> <= Node.val <= 2<sup>31</sup> - 1`
- ```
  # Definition for a binary tree node.
  # class TreeNode:
  #     def __init__(self, val=0, left=None, right=None):
  #         self.val = val
  #         self.left = left
  #         self.right = right
  class Solution:
      def recoverTree(self, root: Optional[TreeNode]) -> None:
          """
          Do not return anything, modify root in-place instead.
          """
          res = [None,None]
          self.prev = TreeNode(float('-inf'))
          self.recoverTreeHelper(root,res)
          res[0].val,res[1].val=res[1].val,res[0].val


      def recoverTreeHelper(self,root,res):
          if root is None:
              return
          self.recoverTreeHelper(root.left,res)
          if root.val<self.prev.val:
              if res[0]==None:
                  res[0]=self.prev
              res[1]=root
          self.prev = root
          self.recoverTreeHelper(root.right,res)
  ```
  ```