34. Search for a Range

作者：qianrong wu

思路

（1）首先从左往右找target，如果没有就返回[-1, -1]；再从右往左重复此操作。 （2）因为这个array已经sort好，可以用binary search

Python3 Code

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution:
    def lowestCommonAncestor(self, root, p, q):
        """
        :type root: TreeNode
        :type p: TreeNode
        :type q: TreeNode
        :rtype: TreeNode
        """
        if not root or not p or not q:
            return none
        if max(p.val, q.val) < root.val:
            return self.lowestCommonAncestor(root.left, p, q)
        elif min(p.val, q.val) > root.val:
            return self.lowestCommonAncestor(root.right, p, q)
        else:
            return root
```

Python Code

```
class Solution(object):
    def searchRange(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: List[int]
        """
        n = len(nums)
        res = [-1, -1]
        if not n:
            return res

        l = 0
        r = n-1
        while l < r:
            m = (l+r)/2
            if nums[m] < target:
                l = m+1
            else:
                r = m
        if nums[l] != target:
            return res
        res[0] = l

        r = n-1
        while l < r:
            m = (l+r)/2+1
            if nums[m] > target:
                r = m-1
            else:
                l = m
        res[1] = r
        return res
```

总结

1. 思路一时间复杂度: O(n); 空间复杂度:O(1)
2. 思路二时间复杂度: O(logn); 空间复杂度:O(1)