이진 탐색트리(BST)가 주어졌을 때 L이상 R 이하의 값을 지닌 노드의 합을 구하라.

Input: root = [10,5,15,3,7,null,18], low = 7, high = 15
Output: 32

```
# 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
```

1.브루트포스

```
class Solution:
    def rangeSumBST(self, root: TreeNode, L: int, R: int) -> int:
        if not root:
            return 0

        return (root.val if L <= root.val <= R else 0) + self.rangeSumBST(root.left, L, R) +
self.rangeSumBST(root.right, L, R)
```

2.재귀

```
class Solution:
    def rangeSumBST(self, root: TreeNode, low: int, high: int) -> int:
        def check(node, val):
            if node == None:
                return val
            elif node.val < low:
                val = check(node.right, val)
            elif node.val > high:
                val = check(node.left, val)
            else:
                val += node.val
                val = check(node.left, val)
                val = check(node.right, val)
            return val

        return check(root, 0)

class Solution:
    def rangeSumBST(self, root: TreeNode, L: int, R: int) -> int:
        def dfs(node: TreeNode):
            if not node:
                return 0
            if node.val < L:
                return dfs(node.right)
            elif node.val > R:
                return dfs(node.left)
            return node.val + dfs(node.left) + dfs(node.right)
```

```
        return dfs(root)
```

3.반복

```
class Solution:
    def rangeSumBST(self, root: TreeNode, L: int, R: int) -> int:
        stack, sum = [root], 0
        while stack:
            node = stack.pop()
            if node:
                if node.val > L:
                    stack.append(node.left)
                if node.val < R:
                    stack.append(node.right)
                if L <= node.val <= R:
                    sum += node.val
        return sum
```

- 반복 dfs - 스택

```
class Solution:
    def rangeSumBST(self, root: TreeNode, L: int, R: int) -> int:
        stack, sum = [root], 0
        while stack:
            node = stack.pop(0)
            if node:
                if node.val > L:
                    stack.append(node.left)
                if node.val < R:
                    stack.append(node.right)
                if L <= node.val <= R:
                    sum += node.val
        return sum
```

- 반복 bys - 큐