TIPS Interviewer Packet A - Session 3

# Interviewer:

## Behavioral:

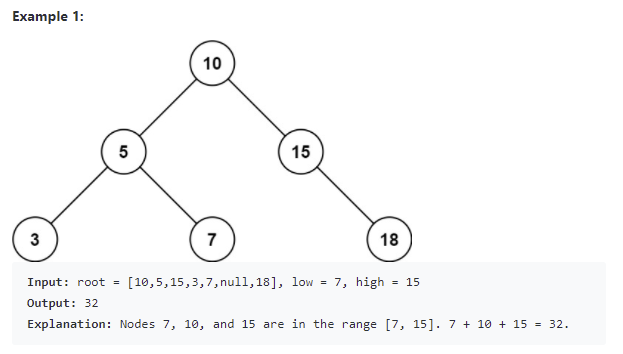
Which project are you most proud of and why?

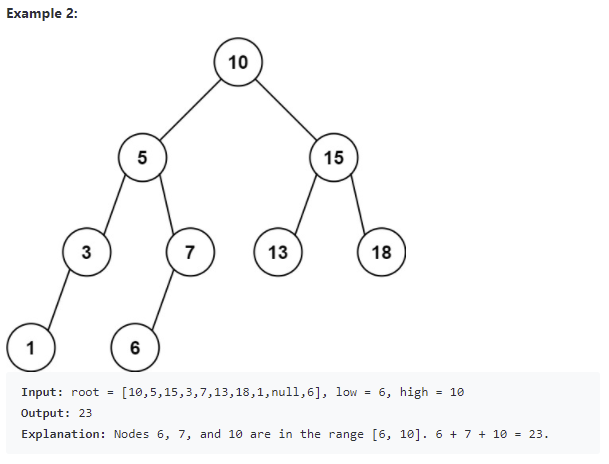
## Question:

<https://leetcode.com/problems/range-sum-of-bst/>

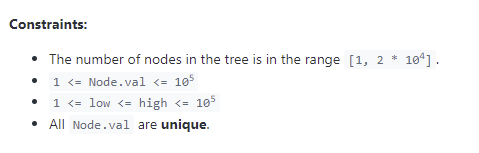
Given the root node of a binary search tree and two integers low and high, return *the sum of values of all nodes with a value in the inclusive range* [low, high].

## Examples:





## Follow up Q&A:



Is it possible for the range to include 0 nodes in the tree?

* Yes

Is it possible for the range to include all n nodes?

* Yes

Is it possible for the range to include only 1 node?

* Yes

Is it possible for the tree or root to be null?

* It would be best to make your code resilient to that edge case.

## Hint(s):

*Ask if they would like a hint before giving a hint*

## Solution(s): (General concept and time/space complexity)

### Depth First Search

We traverse the tree using a depth first search. If node.val falls outside the range [L, R], (for example node.val < L), then we know that only the right branch could have nodes with value inside [L, R].

We showcase two implementations - one using a recursive algorithm, and one using an iterative one.

**Recursive Implementation**

class Solution(object):

def rangeSumBST(self, root, L, R):

def dfs(node):

if node:

if L <= node.val <= R:

self.ans += node.val

if L < node.val:

dfs(node.left)

if node.val < R:

dfs(node.right)

self.ans = 0

dfs(root)

return self.ans

**Iterative Implementation**

class Solution(object):

def rangeSumBST(self, root, L, R):

ans = 0

stack = [root]

while stack:

node = stack.pop()

if node:

if L <= node.val <= R:

ans += node.val

if L < node.val:

stack.append(node.left)

if node.val < R:

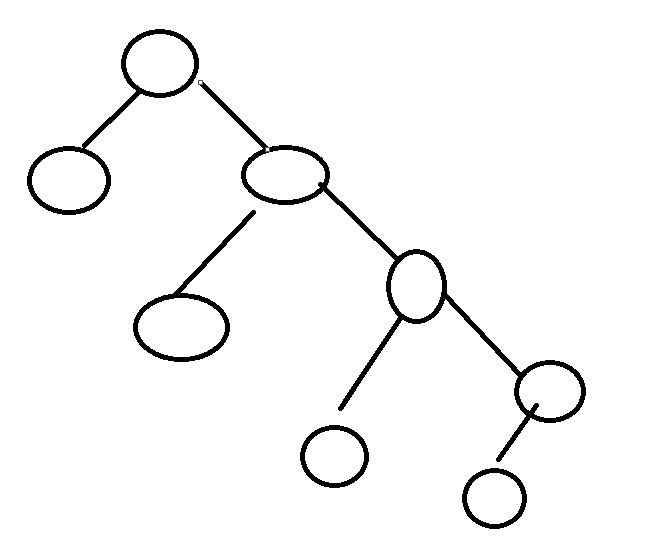
stack.append(node.right)

return ans

Time complexity: O(n)

* If the range includes all the nodes, you must visit all the nodes.

Space complexity: O(n)

* For the recursive implementation, the recursion will consume additional space in the function call stack. In the worst case, the tree is of chain shape, and we will reach all the way down to the leaf node.
* For the iterative implementation, in the worst case we can have a binary search tree like:   
  since we visit the right subtree first (it is added on top of the left child if it exists) all of the left subtree node could be on the stack, so we would have at most O(n/2) nodes on the stack. That makes the space complexity O(n).

### Other questions follow up

*Ask if there is more than 5 minutes remaining when they finish their code and testing.*

How would you change your code to allow for up to 2 inclusive ranges and to return the sum of the nodes with values within at least one of the ranges? (Note: the 2 ranges would have potential overlap (e.g. [0, 10] and [5,20])

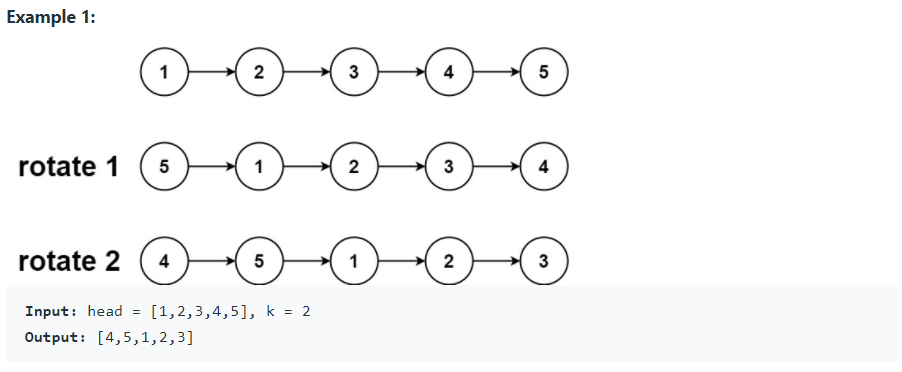
# Interviewee:

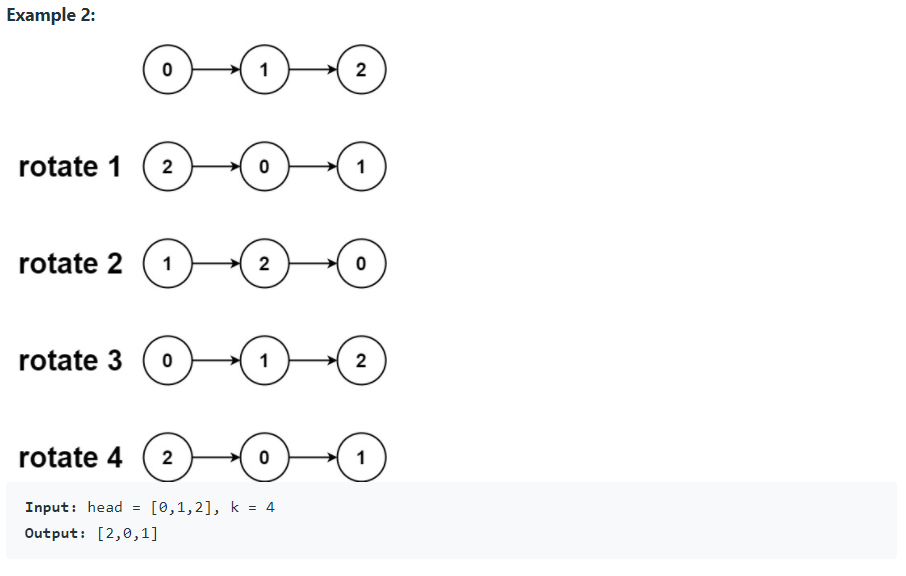
## Question:

<https://leetcode.com/problems/rotate-list/>

Given the head of a linked list, rotate the list to the right by k places.

## Examples:





## Code below or on leetcode