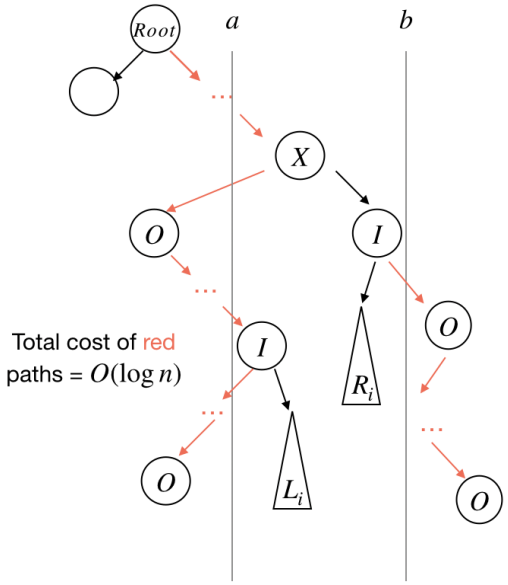
Question 11

Solution (Part A)

Assume that `insert_into_output` takes $O(C)$ time to insert a key. Under this assumption, consider the given `find` method, starting at the root. As long as keys are outside the range $[a, b]$, only one branch is taken, until the first node in $[a, b]$ is found, call it X. Both branches are taken at X.



Consider the left subtree of X. It has no keys greater than X (and hence greater than b), and evaluate the cost of `find` here.
In this subtree, any time an I inside the range $[a, X]$ is enountered, all of its right subtree is guaranteed to be in the range $[I, X]$ and hence in $[a, b]$, so this cost is included in O(m). On the other hand, any time $O < a$ is encountered, the traversal goes right. Either way, at max one extra node(meaning outsite the range $[a, b]$) is explored.
Following a similar argument for the right subtree, at max one extra node is explored per level.

Extra Cost = Cost till you reach X + Extra cost in subtrees

$$= O(h) + 2 * O(log(n) - h)$$
$$= O(log(n))$$

Now to count the right-ward traversal at any I on this path, note that all nodes on each such right are in the range $[a, b]$. Hence the number of steps taken in that subtree is simply a pre-order traversal of that subtree. This is $O(L_i)$, the number of nodes in that subtree. Thus the total on the left of X is $O(\log(n) + \sum L_i)$, where $\sum L_i$ is the number of nodes on the left of X, which are in the range $[a, b]$.

Similarly, the right subtree of X takes $O(\log(n) + \sum R_i)$, where $\sum R_i$ is the number of nodes in the right subtree of X that are in the range $[a, b]$. Thus the total time is $O(\log(n) + m)$, since $\sum L_i + \sum R_i = m$.

Overall Time Complexity taking the assumption into account is $= O(m + log(n) + mC)$

Less tight bound

A less tight bound may be obtained by performing the following simpler analysis. Consider first the cost of reaching the first node within the range, which is $O(\log(n))$. For each node within the range, the cost of traversing the part of its subtree that is outside the range is $O(\log(n))$, since only one branch is taken for nodes outside the range. Since there are m nodes in the range, this total cost is upper bounded by $O(m \log(n))$. Adding the cost to reach the first node in the range, we obtain the total time complexity of `find` as $O(m \log(n) + \log(n)) = O(m \log(n))$.

Overall Time Complexity taking the assumption into account is $= O(mlog(n) + mC)$

Solution (Part B)

The cost of `insert_into_output` is assumed to be $O(C)$ while maintaining the output sorted. It is necessary to have the final output sorted since there is no extra code for sorting it afterwards.

The cost associated with `insert_into_output` therefore should be minimal. Options include-

- C = m. The insertion happens in an array or Linked List using insertion sort or BST(not necessarily balanced) is used and the output is by default sorted since BST.
- C = log(m) BBST or Heap or Skip List all ensure insertion in log(m) time. All give the result in a sorted manner as BBST Inorder is sorted, Heap extract-min gives elements in sorted manner, and Skip List has the elements sorted in the lowest level. Making it the best option in traversal