APRG Assignment 2

March 2020

1 The Problem

In this assignment, you are required to implement a data structure which allows 2 operations -

1. Insert an integer e (**Note :** It doesn't allow duplicates)
2. Get the number of integers in a range

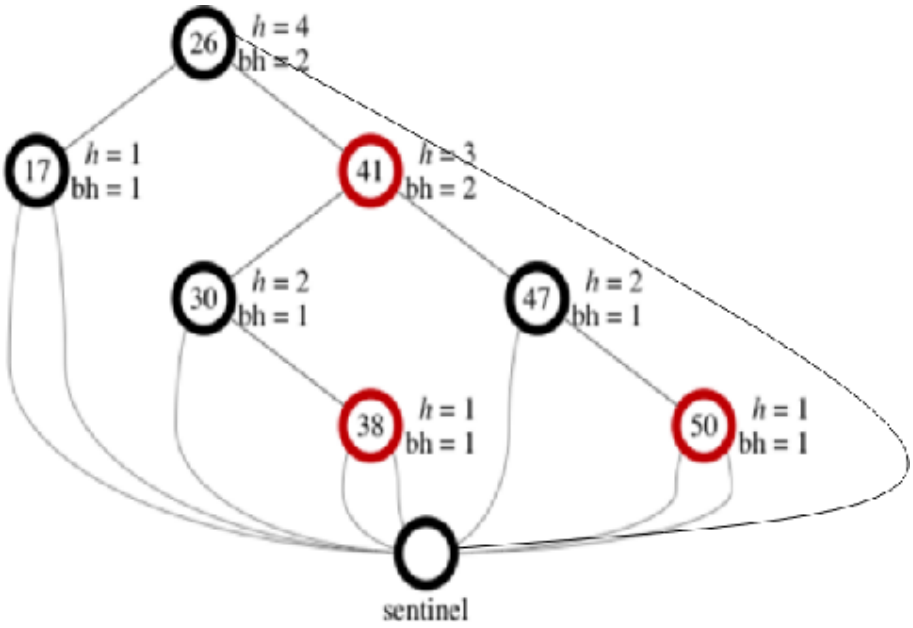
2 Red Black Trees

One way to implement this is using a data structure called Red Black Trees. Red Black Trees are a type of balanced binary search trees where the height of the tree is always $\mathcal{O}(\log n)$.

A red-black tree is a binary search tree with one extra bit per node: a color, which is either red or black.

A red-black tree obeys the five red-black properties:

1. Every node is either red or black.
2. The root is black.
3. Every leaf (the sentinel) is black.
4. If a node is red, then both its children are black. (Hence there can be no two red nodes in a row on a simple path from the root to a leaf.)
5. For each node, all paths from the node to descendant leaves contain the same number of black nodes.



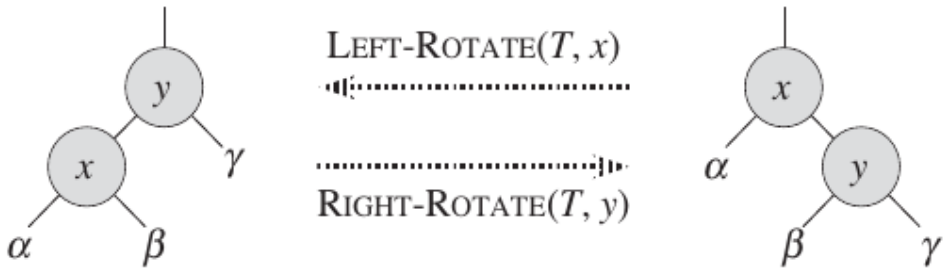
Note : Absent nodes and the parent of the root are represented by the sentinel.

The height of a node is the number of edges in a longest path to a leaf. The black-height of node x, which we write as $bh(x)$, is the number of black nodes (including the sentinel) on the path from x to a leaf, not counting x. By property 5, black-height is well defined. Here is a red-black tree with keys inside nodes and with node heights h and black-heights bh labeled.

2.1 Operations

2.1.1 Rotations

The height of the red black tree is maintained using the rotate operation, which can be implemented in $\mathcal{O}(1)$



Note : Here α, β, γ are subtrees

2.1.2 Insertion

We create a new node (z) with **red** color and insert into the tree, the same way that we do for a normal binary search tree. But this might violate property 2 (if z is the root) or 4 (parent of z is red). In this case we apply the `fix_rbproperty` operation

The `fix_rbproperty` operation works by maintaining the following in-variants

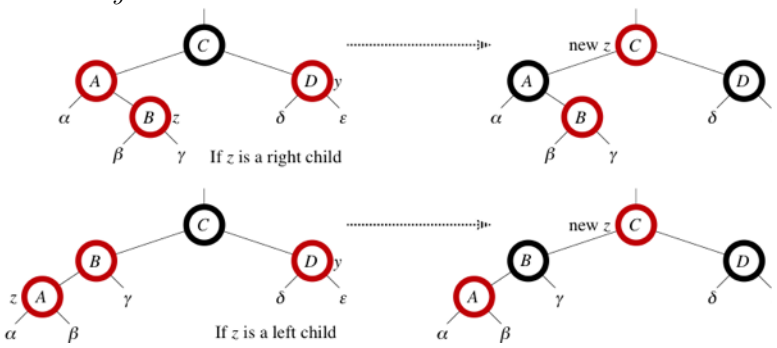
1. z is red
2. Either property 2 is violated or 4 but not both

The operation stops only when $z's$ parent is black. This fixes property 4. To fix the violation of property 2, you can explicitly make the root's color black after completing `fix_rbproperty` operation.

To maintain the loop invariant there are 6 cases. Three cases are described here (z's uncle is to right of $z's$ grandfather, the other three are symmetrical to these.)

Naming: z is the new node which is inserted. y is $z's$ parent's sibling (can be a sentinel too)

1. **Case 1:** y is red

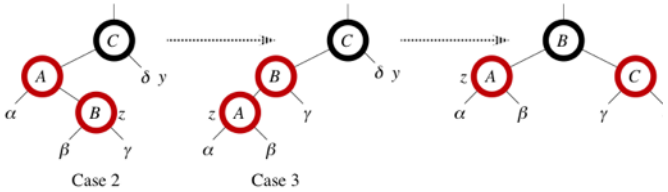


   - z.parent.parent (z's grandparent) must be black, since z and z.parent are both red and there are no other violations of property 4.
   - Make z.parent and y black, so that now z and z.parent are not both red. But property 5 might be violated.
   - Make z.parent.parent red to restore property 5.
   - The next iteration has z.parent.parent as the new z (i.e., z moves up two levels).

2. **Case 2:** y is black, and z is a right child.

- Left rotate around z.parent, so that now z is a left child, and both z and z.parent are red.
- Now this reduces to case 3

3. **Case 3:** y is black, and z is a left child
   - Make z.parent black and z.parent.parent red.
   - Then right rotate on z.parent.parent.
   - We no longer have two red nodes in a row.
   - z.parent is now black, and so the loop test fails and the loop terminates.



2.1.3 Size of subtree

We can augment the tree by storing the size of subtree rooted at each node. This can be later used to implement the range query.

Note - Make sure to update the size of the nodes after each rotation operation.

2.2 Implementation

Implement the following classes

1. `Node` - A class which represents a single node of the red black tree

   Class variables

   - `value` - The integer stored at the current node
   - `left, right, parent` - The reference to respective nodes
   - `size` - The size of the subtree rooted at the current node
   - `color` - Color of the node

2. `RedBlackTree` - A class which implements the red black tree and other methods to perform range queries on the tree.

   Class variables

- `root` - Reference to the root node of the tree

Methods

- `insert(x)` - Insert the integer x into the tree
- `rebalance(node)` - Rebalance the tree according to the 6 cases stated above after insertion.
- `left_rotate(node), right_rotate(node)` - Implement rotation operation at the given node.
- `count_less_than(x)` - Returns the number of nodes with values less than x

Apart from the above methods you can have other helper methods to solve the problem.

3 Input

The first line of the input denotes the number of operations (Q)

The next Q lines are either of the form -

1. `+ x` - Insert x into the data structure
2. `? l r` - Query the number of elements in the range l to r (both inclusive)

4 Output

For each query of the type `? l r`, print the appropriate answer

5 References

1. https://www.cs.dartmouth.edu/ thc/cs10/lectures/0519/0519.html