*Enlargement of Data Structure*

*Background:*

There exist some public standard data structures, such as double linked list, hash table, or binary query tree, and there are also some other applications that need us to make changes or modify, actually there few applications need us to create a new type of data structure.

Normally, some extra information are needed to enlarge one standard data structure, and add into some new operations on such partially new data structure.

In this chapter would introduce the enlargement data structure of Red Black Tree to find the ith element among the certain sequence, and to find the correct interval from Red Black Tree.

*Dynamic Sequence Statistics:*

*Background:*

* The ith element in n elements collection can be simply defined as the element with the ith smallest key value. This chapter would introduce how to modify Red Black Tree to identify random sequence statistic value in O(n).
* Another task is to calculate the rank of an element at the cost of O(lgn), and the rank of an element stands for the ith position in the linear sequence, normally, the linear sequence means the Tree displayed in LDR sequence of all nodes.

*Definition:*

*Order – Statistics Tree is the kind of tree, which adds some extra information on each node among Red Black Tree.* Here, one extra field called node.size is needed to record the number of nodes under the current Sub – Tree (Include the Sub – Tree root node).

However, if we define the size of guard node as 0, then it means the size of T.nil equals to 0. The equation is as below:

*node.size = node.left.size + node.right.size + 1*

Below is the example of *Order – Statistics Tree(aka. Red Black Tree). The first part of node is the key value and the latter part of node is the size field which equals to record the number of nodes under the current node (including the current node).*

*An extra field is needed to display some other information of the Order – Statistics Tree, here we call it the Rank of Order – Statistics Tree. Rank is the sequence number among the whole tree in the LDR sequence.*

A picture containing drawing

Description automatically generated

For example, the node 14 in Black has the rank equals to 5 and the node in Red has the rank equals to 6. *Therefore, we can easily identify that the existence of rank is to eliminate the ambiguous of node with the same key value.*

*Query Element with Given Rank:*

First, we take the advantage of data structure above to realize find the specific node with the given rank, to make it easy, we just find the ith small node in the Order – Sequence Tree.

*Pseudo Code:*

Find the node with rank i in the Sub – Tree with root x:

*OS\_SELECT(x, i) {*

*r = x.left.size + 1; //r stands for the rank of node x.*

*if (i == r) {*

*return x;*

*} else if (i > r) {*

*OS\_SELECT(x.right, i - r);*

*} else if (i < r) {*

*OS\_SELECT(x.left, i);*

*}*

*}*

*Example:*

Try to find the 17th small node in the Order – Statistics Tree.

A picture containing drawing

Description automatically generated

Analyze from Root Node with key 26 and size 20, 17 > root->left->size + 1, therefore 17 > 12 + 1 = 13, so according to the function OS\_SELECT(x, i) above, we can identify that go to check the 17 – 13 = 4th node in the root->right node. Then 4 < node->left->size + 1, aka 4 < 6, so continue search in the left tree. 4 > node->left->size + 1, aka 4 > 2. Then go to check the 4 – 2 = 2th node in the root->right (38). 2 = root->left->size + 1, aka 2 = 2. Therefore, we can conclude that the node 38 is the 17th small node in the Order – Statistics (aka. Red Black Tree).

*Determine Rank of Element:*

Find the rank of one given node – means that the one given node x, make sure the rank of such node in the Tree T. Attention that, the sequence is LDR display sequence. (Note that, in this example, the node x must exist in the Tree T.)

*Rule:*

If node is the left child of its root, then rank equals to root.left.left.size + 1; Otherwise, if node is the right node of its root, then rank equals to root.right.left.size + 1.

*Pseudo Code:*

*OS\_RANK (T, x) {*

*r = x.left.size + 1;*

*y = x;*

*while (y != T.root) {*

*if (y == y.p.right) {*

*r == r + y.p.left.size + 1;*

*}*

*y = y.p;*

*}*

*return*

*}*

*Example:*

A picture containing drawing

Description automatically generated

Try to find the rank of node 38. At first, initialize the rank of node 38 which equals to 38->left->size + 1 = 1 + 1 = 2. Go upper, then 38 is the right node of its parent. Therefore, update the value of r, which equals to 38->parent->left->size + 1 + r = 1 + 1 + 2 = 4. Then in this round, update the node 38, makes it equal to node 30. 30 is its parent’s left child, so go upper. Update the node 30 and makes it equal to 41. Continue to go upper, find that the node 41 is the right node of 26, update the value of rank which equals to 41->parent->left->size + 1 = 12 + 1 + 4 = 13 + 4 = 17. To conclude, 17 is the final rank result of Order – Statistics Tree.

*Cost:*

The time cost of the algorithm is O(lgn).

*Maintain Scale of Sub-Tree:*

After giving the size field of each node, *OS\_SELECT* and *OS\_RAND* can calculate the node information in the sequence quickly. However, it would be meaningless if we do not maintain the information of each node after using *OS\_SELECT* and *OS\_RAND*.

*From the chapter before, we can tell that the insertion operation includes two phases, one is to go down along the tree root and find the specific location to finish insertion. In the second phase is to go up along the tree leaf, doing some color vary and rotation operation to keep the property of Red Black Tree.*

In order to keep the property of Red Black Tree, we need to visit each node from top root to leaf node and add the new size property. *The field size of each newly added node equals to 1, and since there exist O(lgn) nodes, therefore extra cost to maintain these information needs O(lgn).*

In the second phase, there only has some structures, color changes because of rotation, also the times to rotate at most equals to 2. Besides, rotation is a kind of local operation. It would no longer has any effect on the local size field, which need to pay more attention.

For the procedure LEFT\_ROTATION (T, x), we need to add two lines below:

*y.size = x.size;*

*x.size = x.left.size + x.right.size + 1;*

To conclude, the Red Black Tree would take *O(lgn)* at most to insert one node into the Order – Statistics Tree.

*From the chapter before, we can tell that the delete phase would also take two phases: the first phase is to delete from the Tree or just remove the tree node upper. In order to keep the field size of the whole tree, we only need to visit the simple route from the deleted node to the leaf node and update the field of each node in the route. Since the length of the simple route equals to O(lgn), so the cost of first phase equals to O(lgn). The second phase is to rotate and fix up the color of each node and it takes only O(1) times rotation.*

To conclude, deletion on Red Black Tree would just like insertion, takes *O(lgn)* times totally.