Iterative Splaying

Each time a value is inserted or looked up it is splayed to the root of the splay tree through a series of rotations as described in the previous section. The double rotation operations will either move the value to the root or the child of the root of the tree. If the double rotates result in the newly inserted value at the child of the root of the tree, a single rotate is used to move the newly inserted value to the root as depicted in Fig. 10.14 when 30 and 15 are inserted into the splay tree.

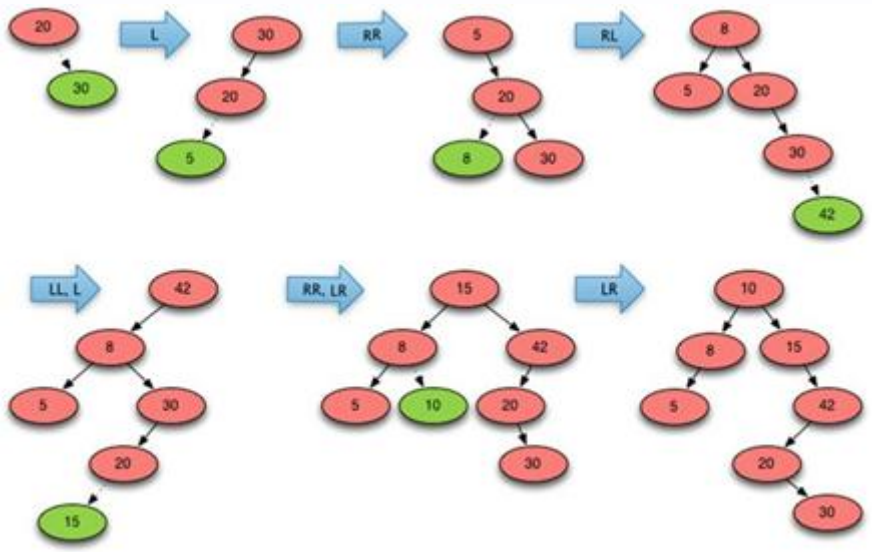

Fig. 10.14 Splay Tree Example

Inserting a new value into a binary search tree without recursion is possible using a while loop. The while loop moves from the root of the tree to the leaf node which will become the new node's parent at which point the loop terminates, the new node is created, and the parent is hooked up to its new child.

After inserting the new node, it must be splayed to the top. To splay it is necessary to know the path that was taken through the tree to the newly inserted node. This path can be recorded using a stack. As the insert loop passes through another node in the tree, it is pushed onto the stack. The end result is that all nodes, from the root to the new child, on the path to the new child are pushed onto this path stack.

Finally, splaying can occur by emptying this path stack. First the child is popped from the stack. Then, the rest of the stack is emptied as follows.

• If two more nodes are available on the stack they are the parent and grandparent of the newly inserted node. In that case a double rotate can be performed resulting in the root of the newly rotated subtree being the newly inserted node. Which double

rotation is required can be determined from the values of the grandparent, parent, and child.

• If only one node remains on the stack it is the parent of the newly inserted node. A single rotation will bring the newly inserted node to the

root of the splay tree.

Implementing splay in the manner described here works well when looking up a value in the tree, whether it is found or not. When a value is found it will be added to the path stack. When a value is not found, the parent should be splayed to the top, which naturally occurs when the looked up value is not found because the parent will be left on the top of the path stack when splaying is performed.

One method of deleting a node from a splay tree is accomplished by deleting just as you would in a binary search tree. If the node to delete has zero or one child it is trivial to delete the node. If the node to delete has two children, then the leftmost value in its right subtree can replace the value in the node to delete and the leftmost value can be deleted from the right subtree. The parent of the deleted node is splayed to the top of the tree.

Another method of deletion requires splaying the deleted node to the root of the tree first. Then the rightmost value of the left subtree is splayed to the root. After splaying the left subtree, its root node's right subtree is empty and the original right subtree can be added to it. The original left subtree becomes the root of the newly constructed splay tree.

Recursive Splaying

Implementing splaying recursively follows the recursive insert operation on binary search trees. The splaying is combined with this recursive insert function. As the recursive insert follows the path down the tree it builds a rotate string of “R” and “L”. If the new item is inserted to the right of the current root node, then a left rotate will be required to splay the newly inserted node up the tree and an “L” is added to the rotate string. Otherwise, a right rotate will be required and an “R” is added to the rotate string.

As the recursive insert function returns, the path to the newly inserted node is retraced by the returning function. The last two characters in the rotate string dictate what double rotation is required. A dictionary or hash table takes care of mapping “RR”, “RL”, “LR”, and “LL” to the appropriate rotate functions. The hash table lookup is used to call the appropriate rotation and the rotate string is truncated (or re-initialized to the empty string depending on when “R” and “L” are added to the rotate string). When the recursive insert is finished, any required single rotation will be recorded in the rotate string and can be performed.

It should be noted that implementing splaying using a rotate string and hash table like this requires about one half the conditional statements to determine the required rotations as compared to the iterative algorithm described above. When inserting a new node the path must be determined by comparing the value to insert to each node on the path to its location in the tree. In the iterative description above, the values on the path are again compared during splaying. In this recursive description the new item is only compared to each item on the path once. This has an impact on performance as shown later in the chapter.

Looking up a value using this recursive implementation works similarly to insert either splaying the found value or its parent if it is not found to the root of the tree. Deleting a value again can be done recursively by first looking up the value to delete resulting in it being splayed to the root of the tree and then performing the method of root removal described in the previous section.