Implementation of Splay Tree.

R Siva Girish
Computer Science
PES University
PES1201700159
sivagirish81@gmail.com

Abstract

A Splay tree is a self adjusting binary search tree which implements the concept of locality of reference like that of cache memory.Each operation performed on the splay tree results in the latest accessed node to become the root node of the splay tree.

Keywords

Splay tree; Binary Search Tree(BST); inorder Successor; remove; insert; order

I. Problem Statement

The Problem at hand is to develop a self adjusting binary tree along with an additional condition that the most recently accessed node is made the root node of the tree(splayed).

II. Implementation

Splay tree is an implementation of locality of reference in a binary search tree.

The implementation provides us with certain functionality :

- get_num_nodes():Returns the number of nodes
- find():Returns 1 or 0 depending on whether the element exists in the tree and splays the node.
- insert(key):Inserts an element into the binary tree in a BST approach and splays the node.
- remove(key): If the tree is not empty it removes the node with the key and splays it's parent.If the node to be removed is the root node it simply removes the node.
- Post_order():Returns a vector containing the post order traversal of the splay tree.
- Pre_order():Returns a vector containing the pre order traversal of the splay tree.

III. Analysis

Amortized analysis of splay tree using potential method.The concept revolving around potential method is to fix a potential function and make sure that during a sequence of operations the potential function can never become negative. In the splay tree the potential of each node is equal to the floor(log(num_of_nodes) attached to it i.e is floor(log(n)).A set of parameters are defined and used for this proof.

The Parameters used are-:

- size(node) : Number of nodes in subtree of node.
- rank(node) :Floor of logarithm of size of node.
- Potential function $\phi = \Sigma$rank(node)
- ϕ' denotes the new potential value
- Amortized cost = M

The major operations in the splay function are the double rotations and the single rotations.
For the sake of this proof let us take the double rotation operation of the splay function into consideration.
Let x be the node to be splayed,y be the parent of x and z be the grandparent of x and both x and y lie in the left subtree of z.Then the implementation requires us to perform a double right rotation.
It can be observed that $\phi'(x) = \phi(z)$,$\phi(y) \geqslant \phi(x)$,$\phi'(y) < \phi'(x)$.
Amortized Cost= Actual cost(2 for 2 rotations) + ΔPotential
$\Delta P = \phi'(x) + \phi'(y) + \phi'(z) - \phi(x) - \phi(y) - \phi(z)$
Let us eliminate y from this equation and add an inequality
$\Delta P \leqslant (\phi(x') - \phi(z)) + \phi'(y) + (\phi'(z) - \phi(x)) - \phi(x) - \phi(x)$
$\Delta P \leqslant 0 + \phi'(x) - 2\phi(x) - \phi'(z)$
$M \leqslant 2 + \phi'(x) - 2\phi(x) - \phi'(z)$ -(i)
Log function is concave.Therefore -:

$\frac{(log(size(x))+log(size'(x))}{2} \leqslant \log(\frac{(size(x)+size'(x))}{2})$ $(size'(x)=size'(z))$
$\frac{(log(size(x))+log(size'(z))}{2} \leqslant \log(\frac{(size(x)+size'(z))}{2})$ $(\log(size(x))=\phi(x))$
$\frac{\phi(x) + \phi'(z)}{2} \leqslant \log(\frac{(size(x)+size(z'))}{2})$ $((size(x) + size'(z))=size'(x))$
$\frac{\phi(x) + \phi'(z)}{2} \leqslant \log(\frac{size'(x)}{2})$ $(\log(\frac{size'(x)}{2})=\phi'(x)-1)$ $\log 2=1$

$\phi'(z) \leqslant 2\phi'(x) - \phi(x) - 2$
Substituting above equation in equation (i) we get :
$M = 3(\phi'(x) - \phi(x))$
The above equation indicates the amortized cost of splaying in case of double rotation. Similarly for single rotation the amortized cost will be :
$M = 3(\phi'(x) - \phi(x)) + 1$ the extra one can be inferred from the fact that there is only one rotation taking place instead of 2 and that extra one will account for a future rotation.
M=3(log(n))+1

IV. Conclusion

Therefore the Amortized cost of splaying a node in a splay tree is log(n).We can further prove that the amortized cost of the find,insert,remove operations are all of O(log(n)) amortized cost in a similar fashion.