Uzair Akram

Algorithms

Assignment 1 - Part 2 Red Black Tree - III

Consider the case where in an R-B tree, a red node can have a red child, if its parent is black in color. All other constraints of the R-B tree are maintained.

Discuss whether (or not) the tree will still have O(log(n)) height.

Red Black trees are self balancing binary search trees. Trees operations have O(h) complexity which means that the time complexity for tree operations is proportional to its height. It is most advantageous if the height is reduced to a minimum. The minimum height is achieved when the tree is balanced, the height then is O(log(N)) and the time complexity for tree operations is O(log(N)). The complexity can be maintained by self balancing trees. Red Black tree is one such tree and balances the tree using this criteria:

1. Every node is colored: Red or Black
2. The root is always black
3. No adjacent red nodes: If a node is red both its children are black
4. Every path from a node to its descendant NULL nodes has the same number of black nodes between.

If the third criteria is removed, and instead we add a new criteria which allows only two adjacent red nodes: a red node can have a red child, if its parent is black in color.

1. Every node is colored: Red or Black
2. The root is always black
3. ~~No adjacent red nodes: If a node is red both its children are black~~
4. Only two adjacent red nodes are allowed: a red node can have a red child, if its parent is black in color
5. Every path from a node to its descendant NULL nodes has the same number of black nodes between.

I think an O(log(N)) time complexity can be maintained using the criteria above. If we only allow 2 adjacent red nodes and maintain all other criteria. However maintaining the criteria becomes more complex. Consider two distant branches in the tree; one with a large number of adjacent red nodes and one with few, the other criteria becomes complex to maintain. However this will reduce the number of rotains potentially by half.

Discuss an insertion scheme that will maintain the height and the constraints.

So to maintain the property of Red Black tree with two adjacent nodes we will keep track of great uncle instead of Uncle, this will allow two nodes red adjacent nodes instead of only one.

Insertion

Find proper position of the node being inserted in the binary tree

Insert the node and color it red

Uzair Akram

If the parent node is red and grandparent node is red it is a violation

Fix the violation

The algorithm would have to

Case1:

Red Parent, Red Grandparent, Black Great Grandparent, Great Uncle Red.

- FIX: Recolor: Great Grandparent node red, Grandparent node and Great uncle node Black.

Case2a:

Grandparent Red, Great uncle Black, Great GrandParent Black.

- Straight line among Great Grandparent, Grandparent, parent, child
- If grandparent is left(right) child of great grandparent, parent is left(right) of grandparent and, child is left(right) child of parent
- FIX: Rotate: GrandParent is new parent and great grandparent is the child
- FIX: Recolor: exchange color of grandparent and great grandParent

Case 2b:

Grandparent red, parent red, great uncle black, great grandparent black.

- Zigzag among great grandparent, grandparent, parent and child.
- If grandparent is left(right) child of great grandparent, parent is right(left) of grandparent and child is left(right) child of parent
- FIX:Rotate: Child becomes Grandparent and Grandparent becomes child.
- Go to case 2a