Red Black Tree

Yu-Tai Ching
Department of Computer Science
National Chiao Tung University

Red Black

- Each node is colored, either red or black.
- A kind of height balance tree.
- Inplementation, each node contains the attributes, *color*, *key*, *left*, *right*, and *p*.
- If a node does not have *p*, *left*, or *right*, they point to NIL
- All NILs are replaced by a black node T.NIL.

Red Black Tree Definition

1. Every node is either red or black.
2. The root is black.
3. Every leaf is black NIL.
4. If a node is red, then both its children are black.
5. For each node, all simple paths from the node to decendant leaves contains the same number of vlack nodes.

- Figure 13.1.
- The black height of node x, $bh(x)$: number of black nodes on simple paths from node x (not including x) down to a leaf.
- Black height of a RB Tree, the black height of the root.
- Together with property 4, height of RB Tree is at most $2{*}bh(root)$.

Lemma 13.1 RB Tree with n internal nodes has height at most $2 \cdot \lg(n+1)$.

- Immediate consequence, SEARCH, MINIMUM, MAXIMUM, SUCCESSOR, PREDECESSOR can be done in $O(\lg n)$ time,.
- How about INSERTION and DELETION? Need to change color of some nodes and rotation.
- figures of rotations, left and right rotation.

BST Quick Review

- Insertion,
- Deletion, z is to be deleted,
  - z has one child or less, delete it.
  - z has two children, move the $\text{SUCCESSOR}(z)$ to z's place, delete $\text{SUCCESSOR}(z)$.

RB Insertion

- Insert a node s as insertion z into a binary search tree.
- z is inserted as a red node.
- If $z.p$ is black, we are done.
- If $z.p$ is red, we have consecutive red nodes.
- Two cases. Either z is left child or right child of $z.p$, cases depends on the color of z's uncle, y.

Case 1, y is red

- $z.p.p$ is black (why?).
- swap color of the two levels, $z.p.p$ was black, now it becomes red,
- $z.p$ and its sibling y were red, now they are black.
- Check RB Tree properties.
- $z.p.p$ is the new z, violation if $z.p$ is red.

Case 2, y is black

1. 2-1 z is right child of $z.p$,
   - left rotate about A (Figure 13.6) to get case 2-2
2. 2-2 z is left child of $z.p$.
   - color changes and right rotate about C (Figure 13.6), we are done.
   - check red-black property, and why we are done?

RB Tree Delete

- z is deleted,
- z has one child or less, its child x moves to z's place (*left* or *right* of $z.p$ points to z's child).
- We push the color of z to x.
- If z was red, x must be black, we are done. Red is removed, x now is black.
- If both z and x were black, we have a double black x.
- z has two children, z's successor w is moved to z's place, and take z's color. w is deleted, we have previous case.
- We have to take care the case that x is double black.
- A simple case, if x is the root, we simply remove the extra black.

x is doubly black, x is not root

- Assume that x is left child of $x.p$, the other case is symmetric.
- Two cases depend on the color of w, the sibling of x
- w is red, we have case 1. If w is black, we have some more sub-cases.

Case 1, w is red, Figure 13.7 (a)

- $x.p$ must be black,
- children of w must be black.
- change color of B and D, and left rotate about B,
- x's sibling is now C, C is black, we have case 2.

Case 2, w is black, both w's children are black, Figure 13.7 (b)

- $x.p$ can be red or black, children of w are black as defined.
- push one black from x and one black from w to $x.p$, x becomes single black, w becomes red,
- color of $x.p$ depends on the original color,
  - $x.p$ was red, then $x.p$ becomes black.
  - $x.p$ was black, the $x.p$ becomes double black, it is now the new x.

Case 3, w is black, one of w's children is black and the other is red

Sub-case 3.1, left child is red. Figure 13.7 (c)

- Right rotate about D, and change color,
- now C is new w, right child is red, we have sub-case 3.2,

Sub-case 3.2, right child is red. Figure 13.7 (d)

- Color changes (refer to procedure RB-DEKETE-FIXUP,
- left rotate about B,