1 Insertion in red-black trees

Insertion of a node \( z \) in an r-b tree is similar to insertion in a BST tree.

Case 1. If node \( z \) is inserted into an empty tree, we color \( z \) BLACK, and make \( z \) the root of the tree.
Otherwise, the tree is not empty and,

Case 2. We perform the standard BST-Insert operations and color \( z \) red.

Possible Problem. When we color \( z \) red, if the parent \( p_z \) of \( z \) is also red, we have a problem. Note that in that case the grandparent of \( z \) must be black. Towards this we need to apply a function \( \text{FIX}(z) \) to fix the RED color of \( z \). If \( z \) is the root, fixing \( z \) is straightforward!

1.1 Case 1: LLr, LRr and RRr, RLr

Case 1. The first case involves the Insertion subcases \( \text{LRr} \) and \( \text{LLr} \) which are shown. Cases \( \text{RRr} \) and \( \text{RLr} \) are not shown but are symmetric. These cases require node recolorings only. Note that if \( g_z \) is the root its color cannot change; this causes an increase to the blackheight of descendant nodes. \( \text{FIX} \) may cause a total of \( O(\log n) \) recursive calls higher in the tree.

// LRr, LLr shown (RRr, RLr symmetric and not shown)
Case 1a: LRr :: \( g_z \) b->r; \( g_z \) and \( p(g_z) \) may become red; Call \( \text{Fix}(g_z) \) next.

\[
\begin{array}{c}
gz/b \\
/ \ /
/ \\
pz/r uz/r \\
/ \\
1 \ 4 5
\end{array}
\begin{array}{c}
gz/r \\
/ \\
/ \\
z/r* 3 4 5
\end{array}
\]

Case 1b: LLr :: Same as before (\( g_z \) b--->r)

\[
\begin{array}{c}
gz/b \\
/ \ /
/ \\
pz/r uz/r \\
/ \\
z/r* 3 4 5
\end{array}
\begin{array}{c}
gz/r \\
/ \\
/ \\
1 2
\end{array}
\]
1.2 Case 2: LRb, and Case 3: LLb

Case 2 covers the case of LRb, and Case 3 the case of LLb. Case 2 is ALWAYS FOLLOWED by Case 3. RLb and RRb are symmetric (not shown).

***** A star (*) shows the node on which FIX is run.
Case 2: LRb is reduced to Case 3: LLb and resolved.

Case 2: LRb

\[ \begin{array}{cccc}
\text{gz/b} & \text{gz/b} \\
/ & \backslash & / \\
\text{pz/r} & \text{uz/b} & \text{z/r} & \text{uz/b} \\
/ & \backslash & / \\
1 & \text{z/r} & 4 & 5 \\
\end{array} \]

-----LRo(pz)---

\[ \begin{array}{cccc}
/ & \backslash & / \\
1 & 2 & 3 & 4 \\
\end{array} \]

Case 3: LLb

\[ \begin{array}{cccc}
\text{gz/b} & \text{pz/b} \\
/ & \backslash & / \\
\text{pz/r} & \text{uz/b} & \text{z/r} & \text{gz/r} \\
/ & \backslash & / \\
3 & 4 & 5 & 1 \\
\end{array} \]

-----RRo(gz)-->

\[ \begin{array}{cccc}
/ & \backslash & / \\
1 & 2 & 3/b & \text{uz/b} \\
\end{array} \]

After the single rotation is performed there can be no way that there are two consecutive RED nodes in a path from the root to \( pz \) (root of the subtree in Case 3). Therefore after a single rotation we are done.

**Conclusion.** Insertion requires \( O(\lg n) \) recolorings (Case 1) and \( O(1) \) rotations (Cases 2 and 3).
2 Deletion in red-black trees

Deletion in an r-b tree is identical to a Deletion in a BST tree. When we perform Delete(z), a node is spliced out; this node is called x. If z has no or one child, then x is z otherwise x is the successor (or the predecessor) of z. In any of these three cases we call y the only child of x, and p(y) the new parent of y which was previously the parent of x.

1. If x is red, then y must be black and p(y) must be black or otherwise x should have been black. Splicing out x causes no violations whatsoever.

2. If x is black, we have a violation of RB3. We distinguish the following subcases.

   3. If y is red, we recolor y black and the violation is resolved.
   4. If y is black and y becomes the root of the tree, no RB3 violation occurs, because all the paths from the root y will have black height one less.
   5. If y is black but not the root, we have a violation of RB3 that can not be resolved immediately. We “transfer” the BLACK color of x to y by coloring y DOUBLE-BLACK. We then need to fix y by calling FIXDELETE(y), i.e. a node calling FIXDELETE is a node “colored” BLACK twice.

2.1 Cases 1 and 2

Comment: Lr1 and Lr2 are not possible; a red node CANNOT have red children.

Case 1: Lr0 : A left rotation is performed and then Case 2a or 3 or 4 applies.

\[
\begin{array}{c}
\text{py/b} \\
/ \ /
** y/b *v/r \quad \text{---LRo(py)} \quad \rightarrow \quad \text{py/r B/b}
\end{array}
\]

** REMARK 1:**

\[
\begin{array}{c}
/ \ /
** y/b \quad *v/r \quad \rightarrow \quad \text{py/r B/b}
\end{array}
\]

\[
\begin{array}{c}
/ \ /
** y/b \quad \rightarrow \quad \text{py/r B/b}
\end{array}
\]

Case 2. Lb0

Subcase 2a. If py is r (Case 1) color py with b and color v with r and stop

\[
\begin{array}{c}
\text{py/r} \\
/ \ /
** y/b *v/b \\
/ \ /
\end{array}
\]

\[
\begin{array}{c}
\text{y/b v/r} \quad \text{and stop}
\end{array}
\]

Subcase 2b, If py is b

\[
\begin{array}{c}
\text{FIXDELETE(py)} \quad \text{py plays the role of y and py}
\end{array}
\]

\[
\begin{array}{c}
\text{also carries the extra black inherited by y due to the x splice out.}
\end{array}
\]

\[
\begin{array}{c}
\text{py/b} \\
/ \ /
** y/b *v/b \\
/ \ /
\end{array}
\]

\[
\begin{array}{c}
\text{y/b v/r}
\end{array}
\]

\[
\begin{array}{c}
/ \ /
A/b B/b
\end{array}
\]
2.2 Cases 3 and 4

Case 3: Lb1rb: Is transformed into case 4 immediately A,v exchanging colors.

```
py/? py/?
/ \ / \ The suffix rb in Lb1rb
** y/b *v/b ---RRo(v)--> **y/b A/b * denotes the children of v
/ \ / \ / \ / \\
1 2 A/r B/b 1 2 3 v/r
/ \ / \ / \ / \\
3 4 5 6 4 B/b
/ \ \\
5 6
```

Case 4: Lb2 and Lb1br (there is neither Lb1bb=Lb0 neither Lb1rr=Lb2)

```
py/? v/? Case4 terminates Case3
/ \ / \ 
** y/b *v/b ---LRo(py)-> py/b B/b
/ \ / \ / \ / \\
1 2 A/? B/r y/b A/? 5 6
/ \ / \ / \ / \\
3 4 5 6 1 2 3 4
```

Running time is $O(\lg n)$ as well. Cases 1, 2a, 3, 4 terminate in $O(1)$ time, Case 2b advances (moves towards the top) one level every time it is executed, and the height of the RB tree is $O(\lg n)$. 