**Red-Black** Trees

A *red-black tree* is a binary search tree where

1) every node is red or black

2) the root is black

3) every leaf is black

4) no red node has a red child

5) any node $x$ has the same number of black nodes on all simple paths to its descendant leaves

---

**Red-Black** Trees (cont)

Every red node must have two black children.

Every internal black node must have two children.

$bh(x)$= # of black nodes on paths to $x$’s descendant leaves (not counting $x$)

$bh(x) \leq h(x) =$ height of $x$

$\forall x$ The subtree rooted at $x$ has at least $2^{bh(x)} - 1$ internal nodes.

---

**Red-Black** Trees (cont)

Proof: By induction on $h(x)$.

**Base**: $h(x)=0$. Then $bh(x)=0$ and $x$ is a leaf and its subtree has 0 internal nodes and $0 = 2^0 - 1$.

**Step**: $h(x)>0$. $x$ is not a leaf so it has two children, $y$ and $z$ of lesser height.

Both $bh(y)$ and $bh(z)$ are either $bh(x)$ or $bh(x) - 1$.

The number of internal nodes in $x$’s subtree is at least

$$1 + 2^{bh(y)} - 1 + 2^{bh(z)} - 1 \geq 2 \cdot 2^{bh(x)} - 1 - 1 = 2^{bh(x)} - 1$$

QED
**Red-Black Trees (cont)**

If $T$ has $n$ internal nodes then
\[
\begin{align*}
    n &\geq 2^{h(root)} - 1 \\
    \lg(n+1) &\geq \frac{h(root)}{2} \\
    h(T) &\leq 2 \cdot \lg(n+1)
\end{align*}
\]

So operations which are $O(h)$ on binary trees are $O(\lg n)$ on red-black trees, provided these operations preserve the properties of red and black trees.

Assume root is black.

---

**Red-Black Trees (cont)**

Rotations

Preserves binary-search-tree property.

---

**Red-Black Trees (cont)**

Insertion - Use normal Insert operation then repair.

Let $z$ be the inserted node (has 2 black nil children)

If $z$ is not the root, make $z$ red.

If $p[z]$ is black then done, otherwise we have two adjacent red nodes.

Let $y$ be $z$'s uncle.

**Case 1 - $y$ is red.**

And repeat if $x$’s parent is red.

If $x$ is the root, make $x$ black.
**Red-Black** Trees (cont)  Insertion

**Case 2** - $y$ is **black** and $z$ is a right child.
    Rotate left at $p[z]$ and go to Case 3.

![Tree Diagram Case 2]

**Red-Black** Trees (cont)  Insertion

**Case 3** - $y$ is **black** and $z$ is a left child.
    Rotate right at $x$ and exchange colors of $p[z]$ and $x$.

![Tree Diagram Case 3]

**Red-Black** Trees (cont)  Insertion Flowchart

Case 1 can repeat at most $h/2 = O(\lg n)$ times.
At most 2 rotations will be needed.

**Red-Black** Trees (cont)  Example: Insertion

DONE
Red-Black Trees (cont)
Example: Insertion

Case 1

Case 2

Case 3
Red-Black Trees (cont) Deletion

Deletion - Assume y is the node to remove.
y has a child x (could be a nil)
If y is red then no change is needed.
If y is black and x is red, then make x black.
Otherwise y and x are black.
Let z be y’s parent and assume y was a left child.

Extra black node needed to maintain black height.

Before

After

Red-Black Trees (cont) Deletion

Case 1 - z is black and w is red.

Left-Rotate at z and exchange colors of z and w.
Go to Case 2, 3 or 4.

Red-Black Trees (cont) Deletion

Case 2 - w and its two children are black.

Make w red and pull a black node above z.
If z is red then absorb extra black node by making z black.
Done or move up tree.

Red-Black Trees (cont) Deletion

Case 3 - w and its right child are black while its left child is red.

Right-Rotate at w and exchange colors of w and its left child.
Go to Case 4.
**Red-Black** Trees (cont)  Deletion

**Case 4** - *w* is **black** and its right child is **red**.

![Diagram of a tree](image)

Left-Rotate at *z* and exchange colors of *z* and *w*.
Extra **black** node is absorbed by *R*.  Done.

---

**Red-Black** Trees (cont)  Deletion

**Flowchart**

```
Start --> Case 1 (Left-Rotate) --> Case 2
        |                   |
        v                   v
Case 3 (Right-Rotate) --- Case 4 (Left-Rotate) --> Case 2
                             |                    |
                             v                    v
                           End
```

Case 2 can repeat at most $h = O(\lg n)$ times.
At most 3 rotations will be needed.

---

**Red-Black** Trees (cont)  Example: Deletion

- **Case 2**
- **Case 4**

![Deletion Flowchart](image)
Red-Black Trees (cont)
Example: Deletion