Red-Black Trees

Definition
Analysis
Clusters (not in book)
Operations

Red-black trees are binary search trees that satisfy:
1. Every node is either red or black.
2. The root is black.
3. If a node is red, then its parent is black.
4. For a given node, every path to a nil has the same number of black nodes, called black-height.

The height of a red-black tree is balanced, that is, its height is $\Theta(lg n)$. Specifically, a red-black tree with $n$ nodes has height $h \leq 2 \log_2(n + 1)$.

Balanced Proof

Let $h$ be the height and $b$ be the black-height.

Point 1: $h \leq 2b$ for all red-black trees.

Basis: A r-b tree with black-height 1 has height 0 or height 1.
Assume: Assume r-b trees with black-height $b$ have height $\leq 2b$.
Show: r-b trees with black-height $b + 1$ have height $\leq 2b + 2$
Induction: Increasing the black-height by 1 increases the height by at most 2.
Balanced Proof Part 2

Point 2: Black-height \( b \) implies \( n \geq 2^b - 1 \)

Basis: If \( b = 1 \), then \( n \geq 1 = 2^1 - 1 \)
Assume: If \( b = k - 1 \), then \( n \geq 2^{k-1} - 1 \)
Show: If \( b = k \), then \( n \geq 2^k - 1 \)

Induction: 2 subtrees with black-height \( k - 1 \) implies \( n \geq 1 + 2(2^{k-1} - 1) = 2^k - 1 \)

Combining the two points:
\( n \geq 2^b - 1 \geq 2^{h/2} - 1 \) implies \( h \leq 2\lg(n + 1) \)

Clusters (not in book)
(not in book) To explain the operations, define a cluster to be a black node and its red children.

- A cluster has one, two, or three nodes.
- One node in a cluster is black.
- Any other nodes are red and are children of the black node.
- A cluster must satisfy the binary search and red-black tree properties.

Insert

\( \text{RB-Insert}(T, x) \)
\( \text{Tree-Insert}(T, x) \) into a “leaf” cluster
cluster ← cluster containing \( x \)
while cluster has too many nodes
  “split” cluster
  cluster ← parent of cluster

\[ \begin{array}{c}
A & B & C & D \\
T1 & T2 & T3 & T4 & T5
\end{array} \]

\[ \begin{array}{c}
A & B \\
T1 & T2 & T3 & T4 & T5
\end{array} \]

\[ \begin{array}{c}
A & B & C & D \\
T1 & T2 & T3 & T4 & T5
\end{array} \]
Delete

\[ \text{RB-Delete}(T, z) \]
\[ \text{Tree-Delete}(T, z) \]
\[ \text{cluster} \leftarrow \text{leaf cluster with one less node} \]
\[ \text{while} \ \text{cluster} \text{ is empty} \]
\[ \quad \text{“splice” or “shift” into} \ \text{cluster} \]
\[ \quad \text{cluster} \leftarrow \text{parent of} \ \text{cluster} \]

\[
\begin{array}{c}
\cdots A \cdots \\
\emptyset & B \\
T1 & T2 & T3
\end{array} \xrightarrow{\text{SPLICE}} \\
\begin{array}{c}
\cdots \cdots \\
A & B \\
T1 & T2 & T3
\end{array}
\]

Delete Part 2

\[
\begin{array}{c}
\cdots A \cdots \\
\emptyset & B & C \\
T1 & T2 & T3 & T4
\end{array} \xrightarrow{\text{SHIFT}} \\
\begin{array}{c}
\cdots B \cdots \\
A & C \\
T1 & T2 & T3 & T4
\end{array}
\]

\[
\begin{array}{c}
\cdots A \cdots \\
\emptyset & B & C & D \\
T1 & T2 & T3 & T4 & T5
\end{array} \xrightarrow{\text{SHIFT}} \\
\begin{array}{c}
\cdots B \cdots \\
A & C & D \\
T1 & T2 & T3 & T4 & T5
\end{array}
\]