Lecture 02/20/17

Lecturer: Xiaodi Wu

Reading: Chapter 13.1
Red-Black Trees

Red-Black trees are **binary search trees** with the following extra properties.

**Bounded Depth Binary Search Tree:** \( h = O(\log(n)) \)

- **Color Property:** every node is either red or black.
- **Root Property:** the root is black.
- **External Property:** every external node (leaf, NIL) is black.
- **Internal Property:** the children of a red node are black.
- **Depth Property:** For each node, all simple paths from the node to descendant leaves contain the same number of black nodes. (Equivalently, each node replaced by the root.)
Red-Black Trees: ignoring leaves, i.e., NIL
Properties of Red-black trees

Data Structures

- Each node contains attributes \textit{color}, \textit{key}, and normal ones like \textit{p}, \textit{left}, \textit{right}.
- Each leaf could be a separate NIL node, or more efficiently represented as a single \text{T.nil}.

Black-Height/Black-Depth

For every node \(v\) in any red-black tree
- \textit{black-height}, \(bh(v)\), is the number of black nodes from \(v\) (excluded) to any leaf.
- \textit{black depth}, is the number of black ancestors.

Depth Property: equivalently, all the external nodes have the same \textit{black depth}.
Properties of Red-black trees

Data Structures

- Each node contains attributes color, key, and normal ones like p, left, right.
- Each leaf could be a separate NIL node, or more efficiently represented as a single T.nil.

Black-Height/Black-Depth

For every node $v$ in any red-black tree

- black-height, $bh(v)$, is the number of black nodes from $v$ (excluded) to any leaf.
- black depth, is the number of black ancestors.
Properties of Red-black trees

Data Structures

▶ Each node contains attributes color, key, and normal ones like p, left, right.
▶ Each leaf could be a separate NIL node, or more efficiently represented as a single T.nil.

Black-Height/Black-Depth

For every node \( v \) in any red-black tree

▶ **black-height**, \( bh(v) \), is the number of black nodes from \( v \) (excluded) to any leaf.
▶ **black depth**, is the number of black ancestors.

Depth Property: equivalently, all the external nodes have the same black depth.
Example of Red-black trees

Every leaf, shown as a NIL, is black. Each non-NIL node is marked with its black-height; 0.

The same red-black tree but with each NIL replaced by the single sentinel T: nil, which is always black, and with black-heights omitted. The root’s parent is also the sentinel.

The same red-black tree but with leaves and the root’s parent omitted entirely. We shall use this drawing style in the remainder of this chapter.
Example of Red-black trees

Every leaf, shown as a NIL, is black. Each node is marked with its black-height; NIL shaves black-height 0.

The same red-black tree but with each NIL replaced by the single sentinel T: nil, which is always black, and with black-heights omitted. The root’s parent is also the sentinel.

The same red-black tree but with leaves and the root’s parent omitted entirely. We shall use this drawing style in the remainder of this chapter.
Example of Red-black trees

Figure 13.1
Ar ed - bl a ckt re ew i t h b lac k n o de sd a r k e n e da n dr e d n o de ss h a d ed . E v e r y n o d e i n a red-black tree is either red or black, the children of a red node are both black, and every simple path from a node to a descendant leaf contains the same number of black nodes.

(a) Every leaf, shown as a `NIL`, is black. Each non-NIL node is marked with its black-height; `NIL` shows black-height 0.

(b) The same red-black tree but with each `NIL` replaced by the single sentinel `T: nil`, which is always black, and with black-heights omitted. The root's parent is also the sentinel.

(c) The same red-black tree but with leaves and the root's parent omitted entirely. We shall use this drawing style in the remainder of this chapter.
Properties of Red-Black trees: True

True of False

- A subtree of a red-black tree is itself a red-black tree.
Properties of Red-Black trees: True

True of False

- A subtree of a red-black tree is itself a red-black tree. **False.** The root of a red-black tree must be black. Any subtree with a red root would be an adequate counterexample.
Properties of Red-Black trees: True

True of False

- A subtree of a red-black tree is itself a red-black tree. **False.** The root of a red-black tree must be black. Any subtree with a red root would be an adequate counterexample.

- The sibling of an external node is either external or it is red.
Properties of Red-Black trees: True

True of False

- A subtree of a red-black tree is itself a red-black tree. **False.** The root of a red-black tree must be black. Any subtree with a red root would be an adequate counterexample.

- The sibling of an external node is either external or it is red. **True.** If an external node had a sibling that was neither external nor red (i.e., an internal black sibling), then the black depth between the two nodes would not be the same. This violates the property of red-black trees that states all leaf nodes must have the same black depth.
Proof of Bounded Depth

Lemma (13.1)

A red-black tree with \( n \) internal nodes has height at most \( 2 \log(n + 1) \).

Proof.

By induction, the subtree rooted at any node \( x \) contains at least \( 2^{bh(x)} - 1 \) internal nodes.

- When \( bh(x) = 0 \), \( x \) is a leaf, \( 2^0 - 1 = 0 \) internal node.
- Otherwise \( bh(x) > 0 \) and \( x \) has two children, each with black-height \( bh(x) \) or \( bh(x) - 1 \). Thus, the number of internal nodes is at least

\[
(2^{bh(x)} - 1 - 1) + (2^{bh(x)} - 1 - 1) + 1 = 2^{bh(x)} - 1.
\]
Proof of Bounded Depth

Lemma (13.1)

A red-black tree with \(n\) internal nodes has height at most \(2 \log(n + 1)\).

Proof.

By induction, the subtree rooted at any node \(x\) contains at least \(2^{bh(x)} - 1\) internal nodes.

Finally, let \(h\) be the height of the red-black tree and \(bh\) its black-height. We claim that (why?)

\[ bh \geq h/2. \]

Thus, we have \(n \geq 2^{h/2} - 1\), which leads to

\[ h \leq 2 \log(n + 1). \]
How to Operate on Red-Black Trees?

Basic operations

▶ Tree-Search: the same in $O(h) = O(\log(n))$. 
How to Operate on Red-Black Trees?

Basic operations

- Tree-Search: the same in $O(h) = O(\log(n))$.
- Tree-Insert, Tree-Delete: potentially violate red-black tree properties. Restore in $O(h)$?