

CH-231-A

Algorithms and Data Structures

ADS

Lecture 24

Dr. Kinga Lipskoch

Spring 2022

Modify Operation: Deletion

```

TREE-DELETE( $T, z$ )
1  if  $z.left == \text{NIL}$ 
2      TRANSPLANT( $T, z, z.right$ )
3  elseif  $z.right == \text{NIL}$ 
4      TRANSPLANT( $T, z, z.left$ )
5  else  $y = \text{TREE-MINIMUM}(z.right)$ 
6      if  $y.p \neq z$ 
7          TRANSPLANT( $T, y, y.right$ )
8           $y.right = z.right$ 
9           $y.right.p = y$ 
10     TRANSPLANT( $T, z, y$ )
11      $y.left = z.left$ 
12      $y.left.p = y$ 

```

Time complexity: $O(h)$

Binary Search Tree: Summary

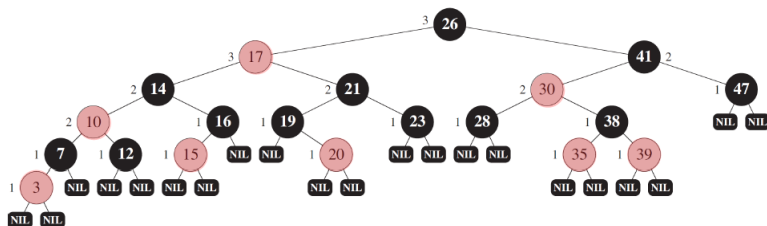
- ▶ BST provides all basic dynamic set operations in $O(h)$ running time, including:
 - ▶ Search
 - ▶ Minimum
 - ▶ Maximum
 - ▶ Predecessor
 - ▶ Successor
 - ▶ Insert
 - ▶ Delete
- ▶ Hence, BST operations are fast if h is small, i.e., if the tree is balanced. Then, $O(h) = O(\lg n)$.

Red-Black Trees: Definition

- ▶ A **red-black tree** is a BST that besides the attributes about parent, left child, right child, and key holds the attribute of a color (**red** or **black**), which is encoded in one additional bit.
- ▶ Special convention: All leaves have NIL as key.
- ▶ The node colors are used to impose constraints on the nodes such that no path from the root to a leaf is more than twice as long as any other path.
- ▶ Hence, the tree is **approximately balanced**.

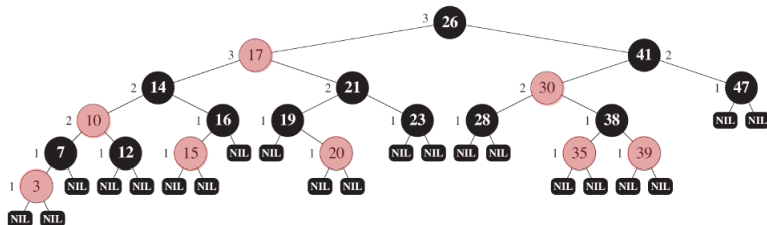
Property 1 (Duh Property)

Every node is either red or black.



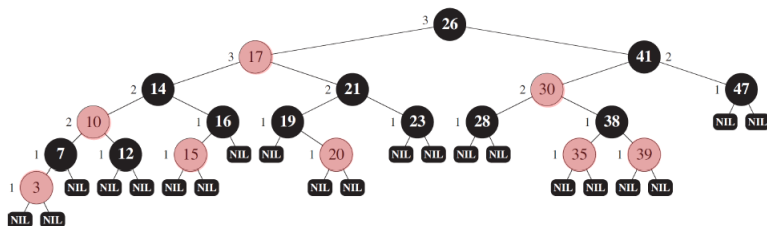
Property 2 (RooB Property)

The root is black.



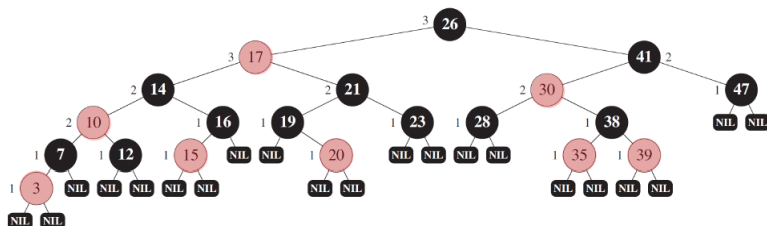
Property 3 (LeaB Property)

All leaves (NIL) are black.



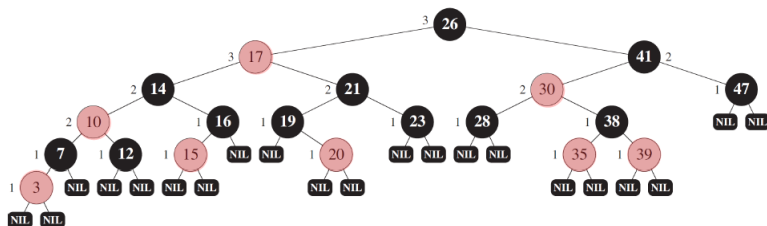
Property 4 (BredB Property)

If a node is red, then both children are black.



Property 5 (BH Property)

For each node all paths from the node to a leaf have the same number of black nodes.

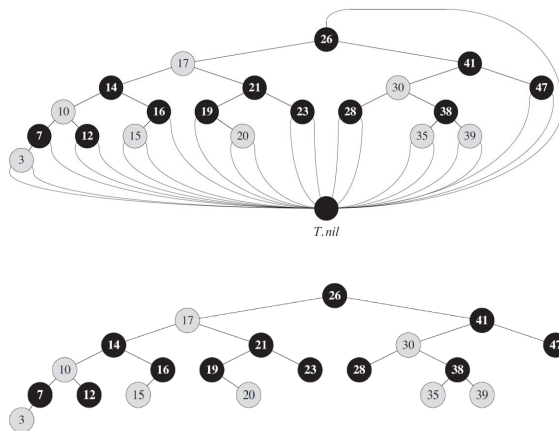


For each node x , we can define a unique black height $bh(x)$.

Properties

1. Every node is either red or black (Duh)
2. The root is black (RooB)
3. All leaves are black (LeaB)
4. If a node is red, then both children are black (BredB)
5. For each node all paths from the node to a leaf have the same number of black nodes (BH)

NIL Sentinel



Number of Nodes vs. Black-Height

Lemma 1:

Let $n(x)$ be the number of non-leaf nodes of a red-black subtree rooted at x . Then, $n(x) \geq 2^{bh(x)} - 1$.

Proof (by induction on height $h(x)$ of node x):

- ▶ $h(x) = 0$: x is a leaf. $bh(x) = 0$. $2^{bh(x)} - 1 = 0$. $n(x) \geq 0$. True.
- ▶ $h(x) > 0$: x is a non-leaf node. It has two children c_1 and c_2 . If c_i is red, then $bh(c_i) = bh(x)$, else $bh(c_i) = bh(x) - 1$. Use assumption, since $h(c_i) < h(x)$,

$$n(c_i) \geq 2^{bh(c_i)} - 1 \geq 2^{bh(x)-1} - 1.$$
 Thus,

$$n(x) = n(c_1) + n(c_2) + 1 \geq 2(2^{bh(x)-1} - 1) + 1 = 2^{bh(x)} - 1.$$

Height vs. Black-Height

Lemma 2:

Let h be the height of a red-black tree with root r . Then,
 $bh(r) \geq h/2$.

Proof:

- ▶ Let r, v_1, v_2, \dots, v_h be the longest path in the tree.
- ▶ The number of black nodes in the path is $bh(r)$.
- ▶ Thus, the number of red nodes is $h - bh(r)$.
- ▶ Since v_h is black (LeaB property) and every red node in the path must be followed by a black one (BredB property), we have $h - bh(r) \leq bh(r)$.
- ▶ Hence, $bh(r) \geq h/2$.

Height of a Red-Black Tree

Theorem:

A red-black tree with n non-leaf nodes has height $h \leq 2 \lg(n + 1)$.

Proof:

- ▶ Lemma 1: $n \geq 2^{bh(r)} - 1$ (r being the root).
- ▶ Lemma 2: $bh(r) \geq h/2$.
- ▶ Thus, $n \geq 2^{h/2} - 1$.
- ▶ So, $h \leq 2 \lg(n + 1)$.

Corollary:

The height of a red-black tree is $O(\lg n)$.

All dynamic set operations can be performed in $O(\lg n)$, if we maintain the red-black tree properties.