# CS60020: Foundations of Algorithm Design and Machine Learning

Sourangshu Bhattacharya

# Balanced search trees

***Balanced search tree:*** A search-tree data structure for which a height of $O(\lg n)$ is guaranteed when implementing a dynamic set of $n$ items.

**Examples:**

- AVL trees
- Splay trees
- B-trees
- Red-black trees

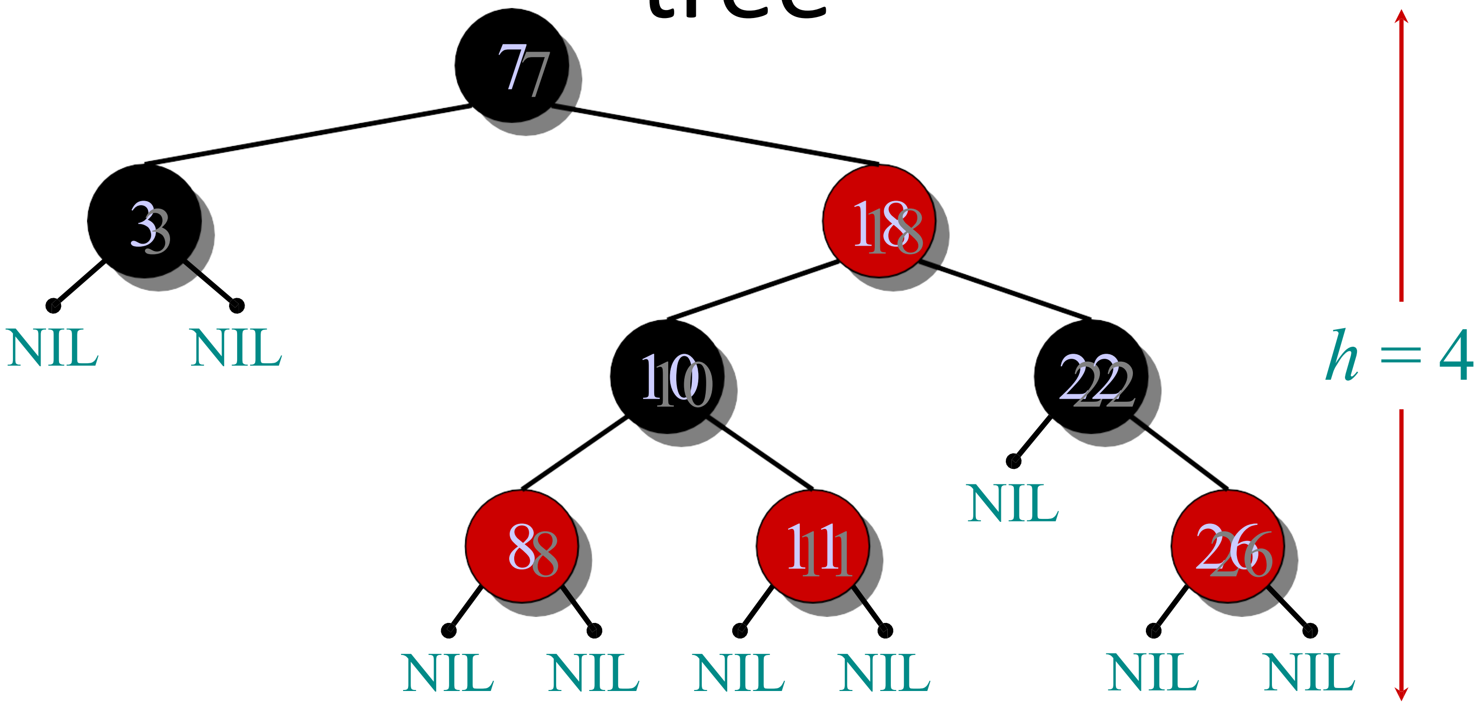
# Red-black trees

This data structure requires an extra one-bit color field in each node.

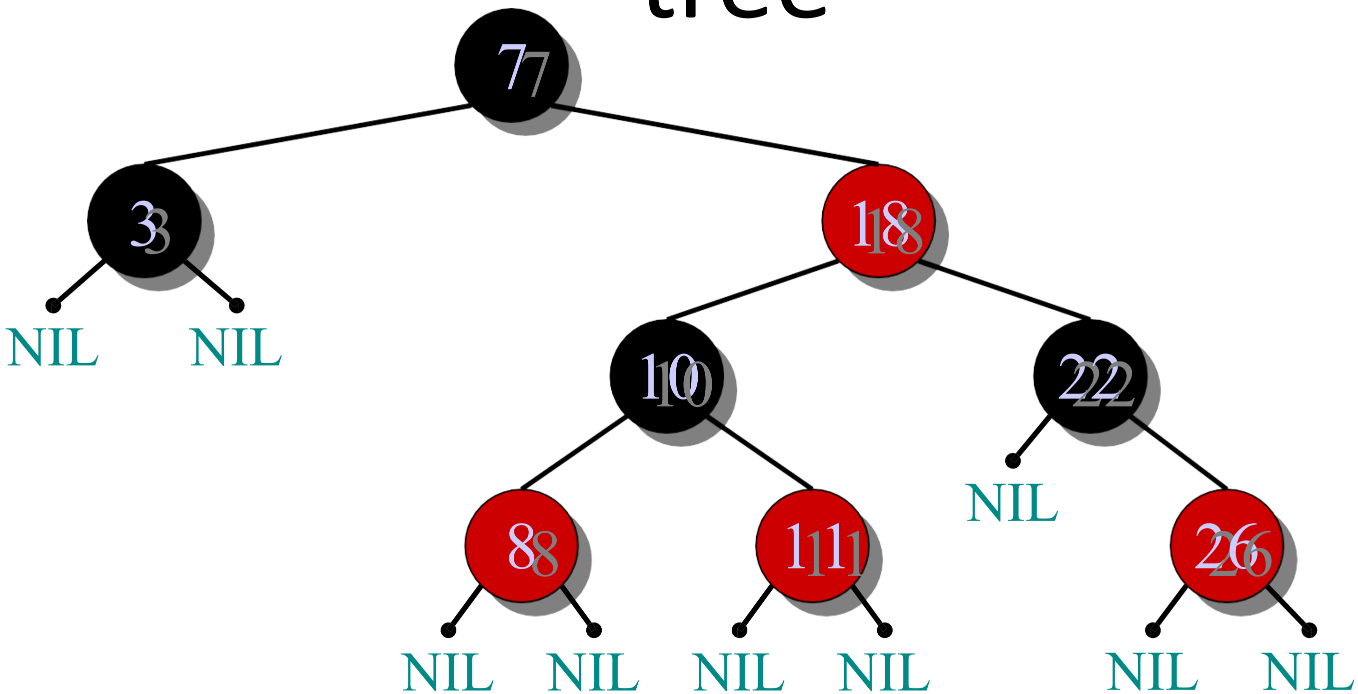
***Red-black properties:***

1. Every node is either red or black.
2. The root and leaves (NIL's) are black.
3. If a node is red, then its parent is black.
4. All simple paths from any node $x$ to a descendant leaf have the same number of black nodes = black-height($x$).

# Example of a red-black tree

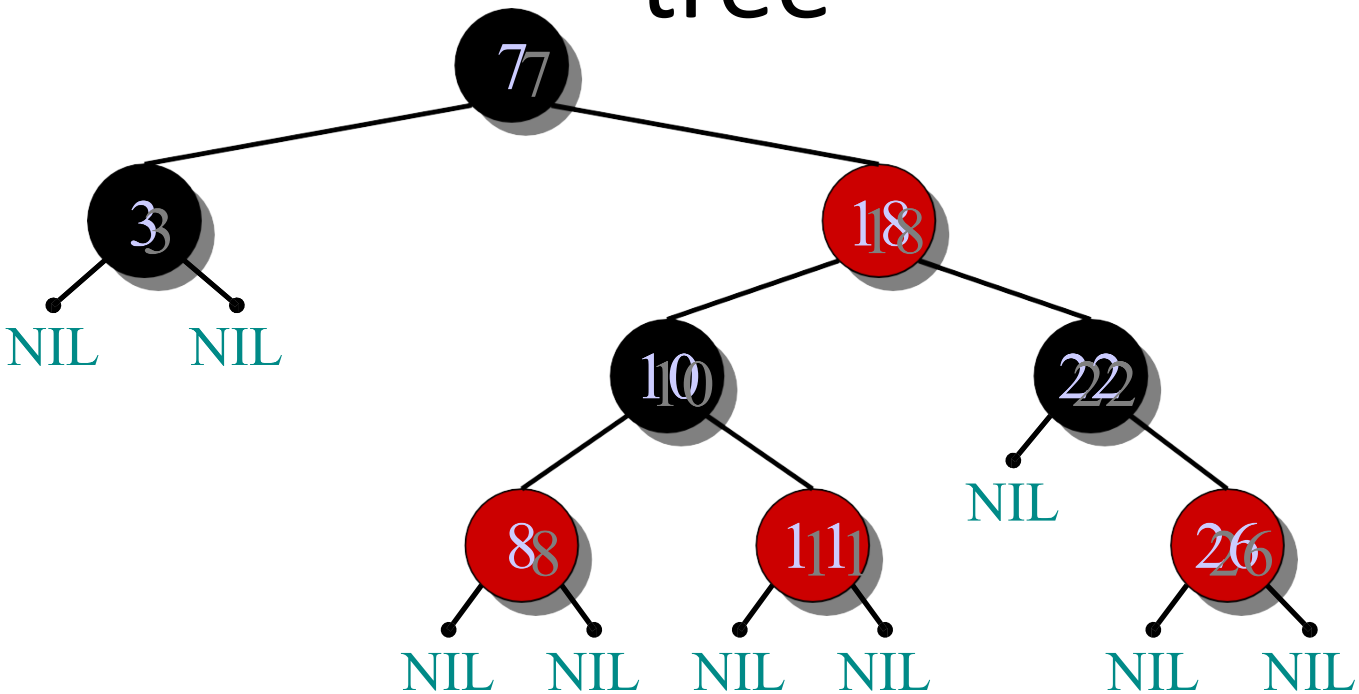


# Example of a red-black tree



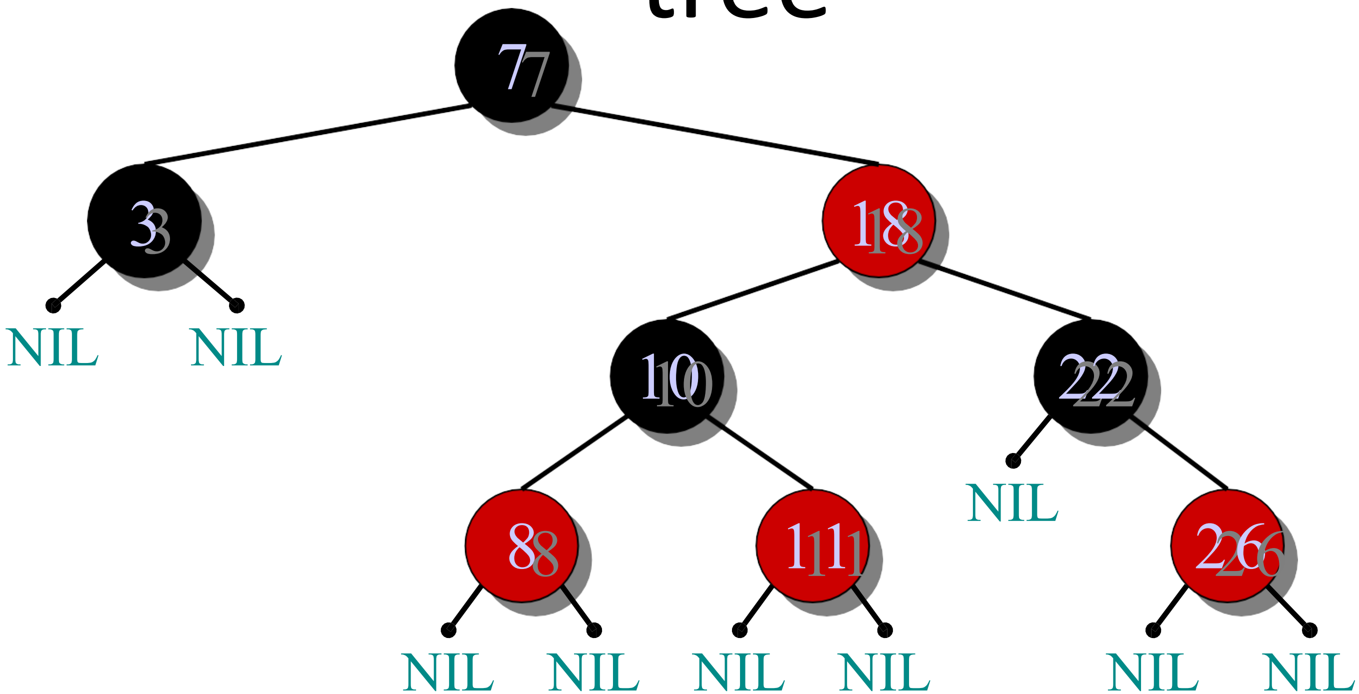

1. Every node is either red or black.

# Example of a red-black tree



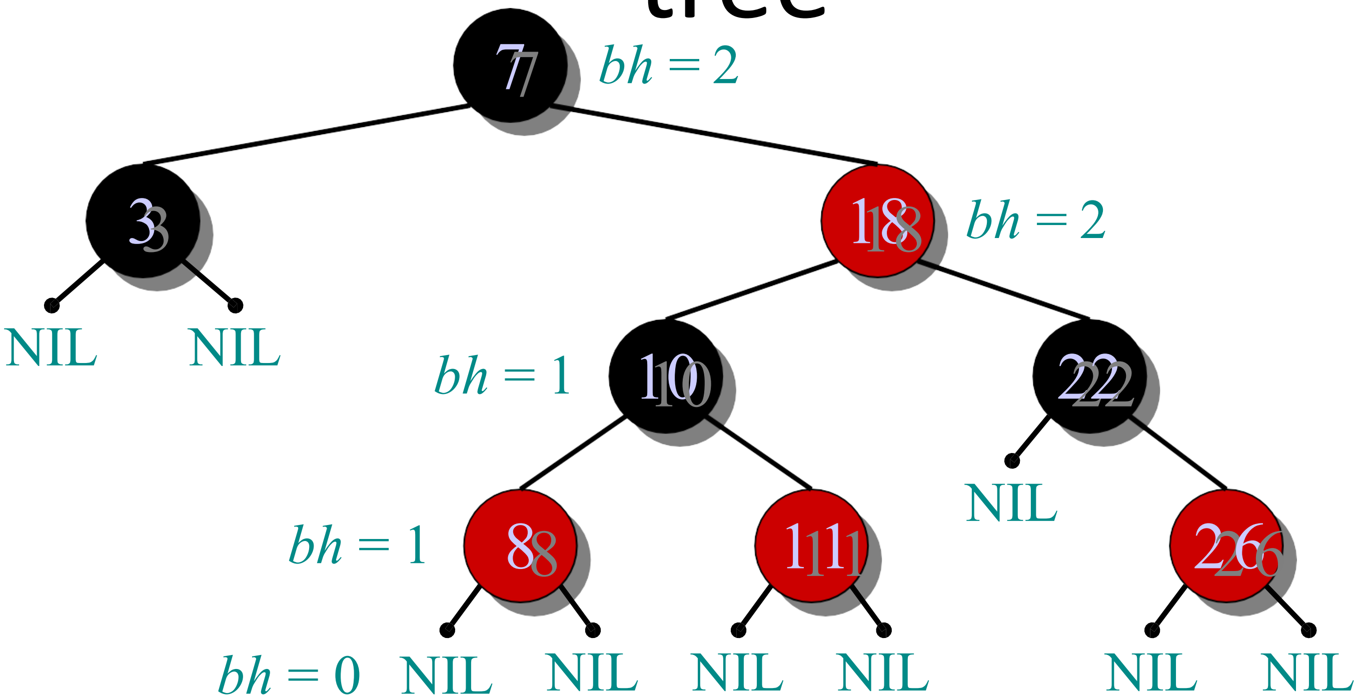

2. The root and leaves (NIL's) are black.

# Example of a red-black tree

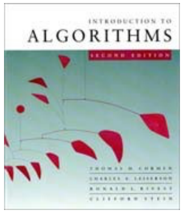



3. If a node is red, then its parent is black.

# Example of a red-black tree




4. All simple paths from any node *x* to a descendant leaf have the same number of black nodes = *black-height*(*x*).




# Height of a red-black tree

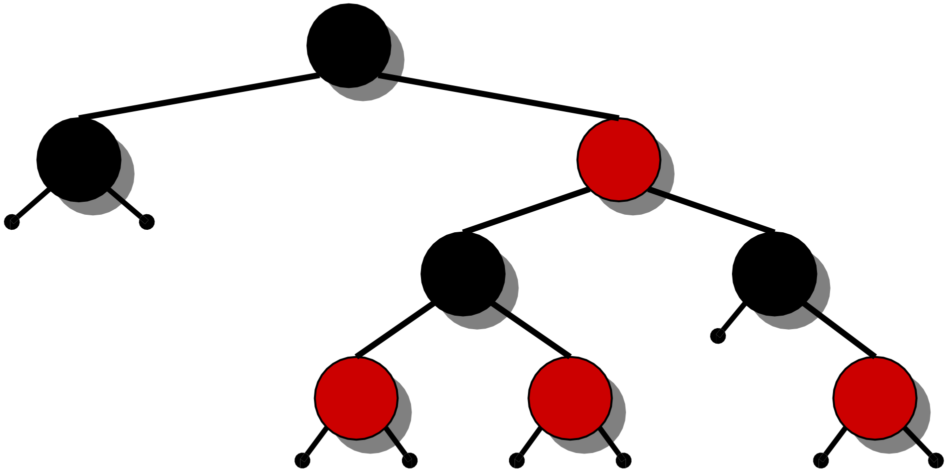
**Theorem.** A red-black tree with $n$ keys has height

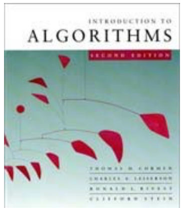
$$h \le 2\lg(n+1).$$

*Proof.* (The book uses induction. Read carefully.)

**INTUITION:**

- Merge red nodes into their black parents.






# Height of a red-black tree

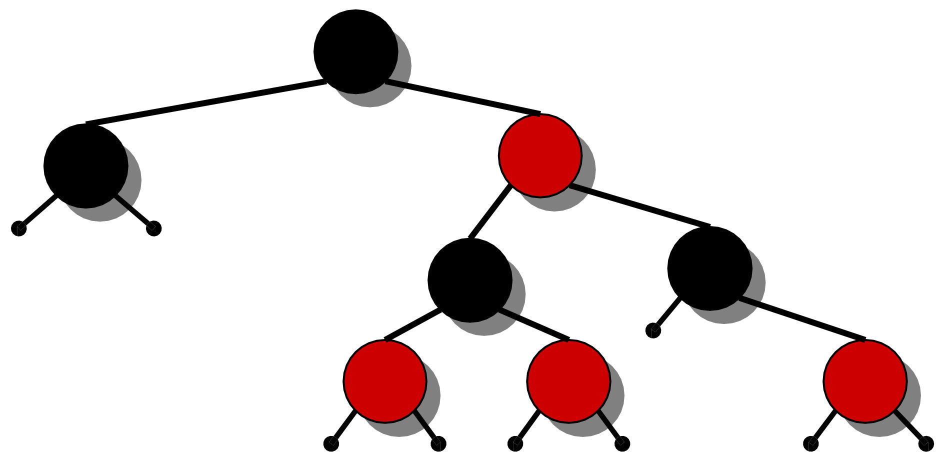
**Theorem.** A red-black tree with n keys has height

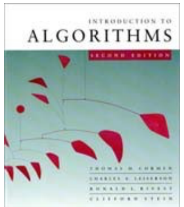
$$h \leq 2 \lg(n + 1).$$

*Proof.* (The book uses induction. Read carefully.)

**INTUITION:**

- Merge red nodes into their black parents.






# Height of a red-black tree

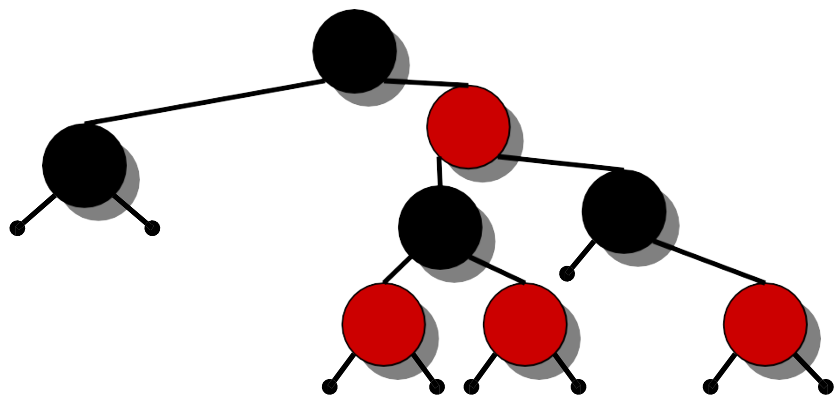
**Theorem.** A red-black tree with n keys has height

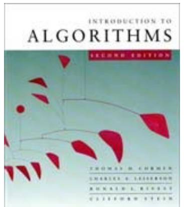
$$h \leq 2 \lg(n + 1).$$

*Proof.* (The book uses induction. Read carefully.)

**INTUITION:**

- Merge red nodes into their black parents.






# Height of a red-black tree

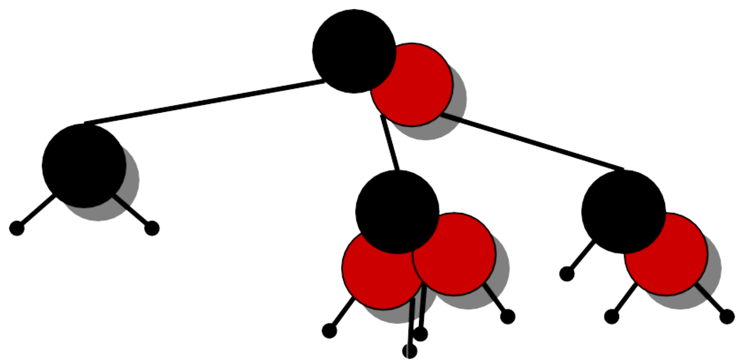
**Theorem.** A red-black tree with n keys has height

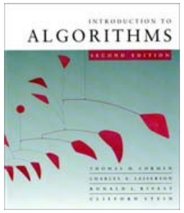
$$h \leq 2 \lg(n + 1).$$

*Proof.* (The book uses induction. Read carefully.)

**INTUITION:**

- Merge red nodes into their black parents.






# Height of a red-black tree

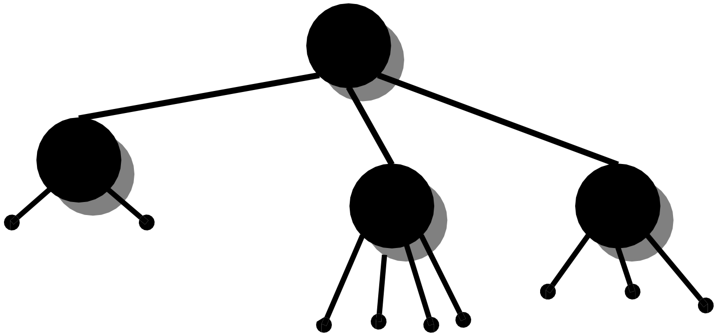
**Theorem.** A red-black tree with n keys has height

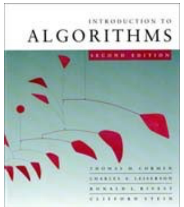
$$h \leq 2\lg(n+1).$$

*Proof.* (The book uses induction. Read carefully.)

**INTUITION:**

- Merge red nodes into their black parents.





# Height of a red-black tree

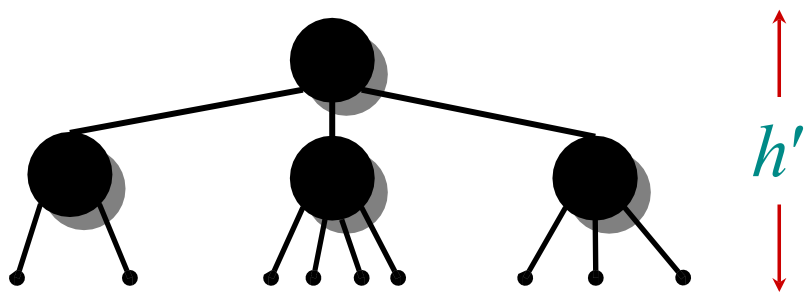
**Theorem.** A red-black tree with n keys has height

$$h \leq 2 \lg(n+1).$$

*Proof.* (The book uses induction. Read carefully.)

**INTUITION:**

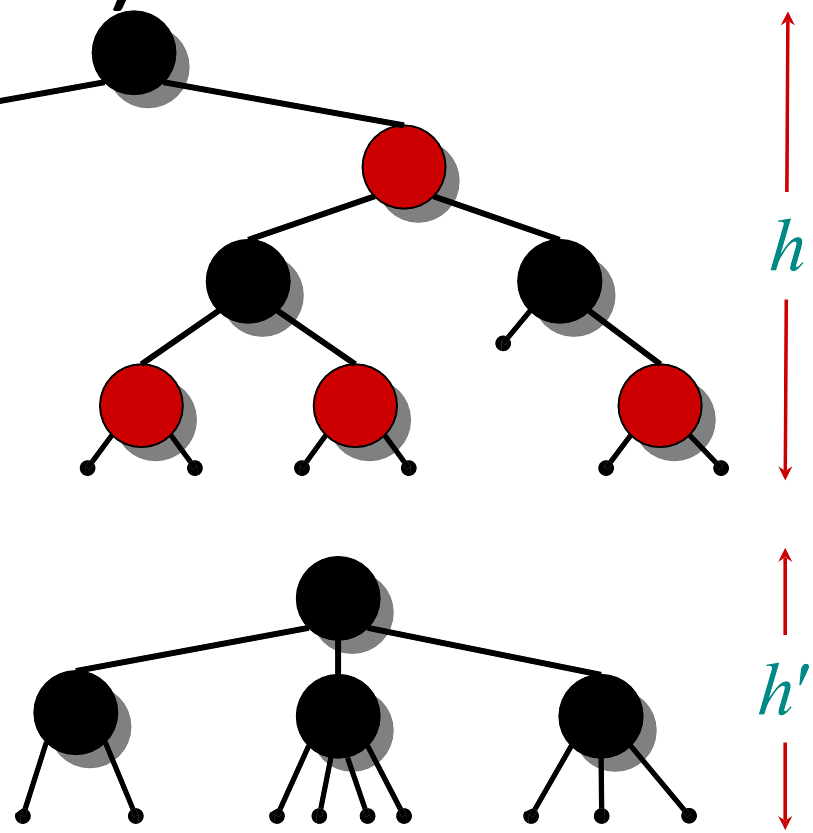
- Merge red nodes into their black parents.




- This process produces a tree in which each node has 2, 3, or 4 children.
- The 2-3-4 tree has uniform depth $h'$ of leaves.

# Proof (continued)

- We have $h' \geq h/2$, since at most half the leaves on any path are red.
- The number of leaves in each tree is $n + 1$
  $\Rightarrow n + 1 \geq 2^{h'}$
  $\Rightarrow \lg(n + 1) \geq h' \geq h/2$
  $\Rightarrow h \leq 2 \lg(n + 1)$. □



# Query operations

**Corollary.** The queries SEARCH, MIN, MAX, SUCCESSOR, and PREDECESSOR all run in $O(\lg n)$ time on a red-black tree with $n$ nodes.

# Modifying operations

The operations INSERT and DELETE cause modifications to the red-black tree:

- the operation itself,
- color changes,
- restructuring the links of the tree via ***"rotations"***.