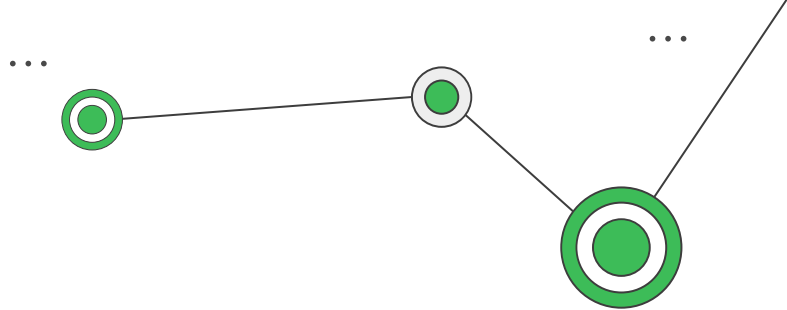
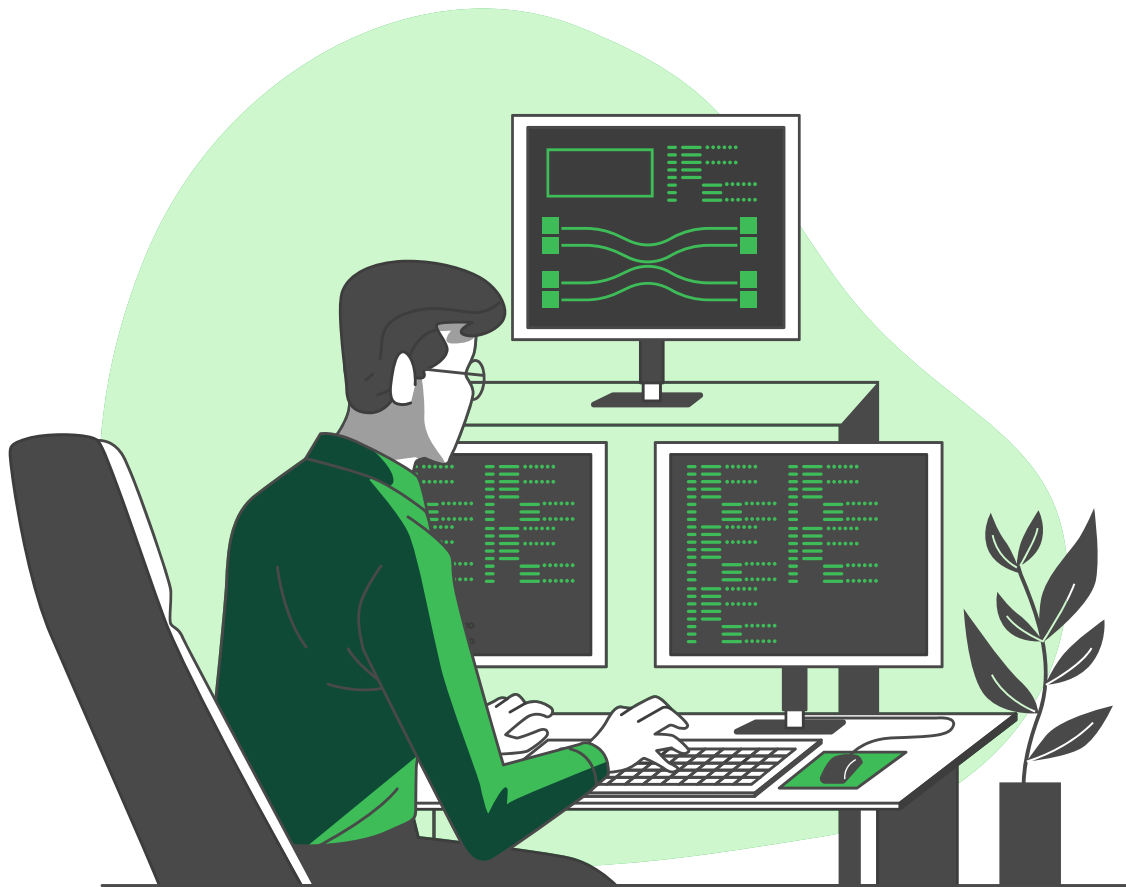
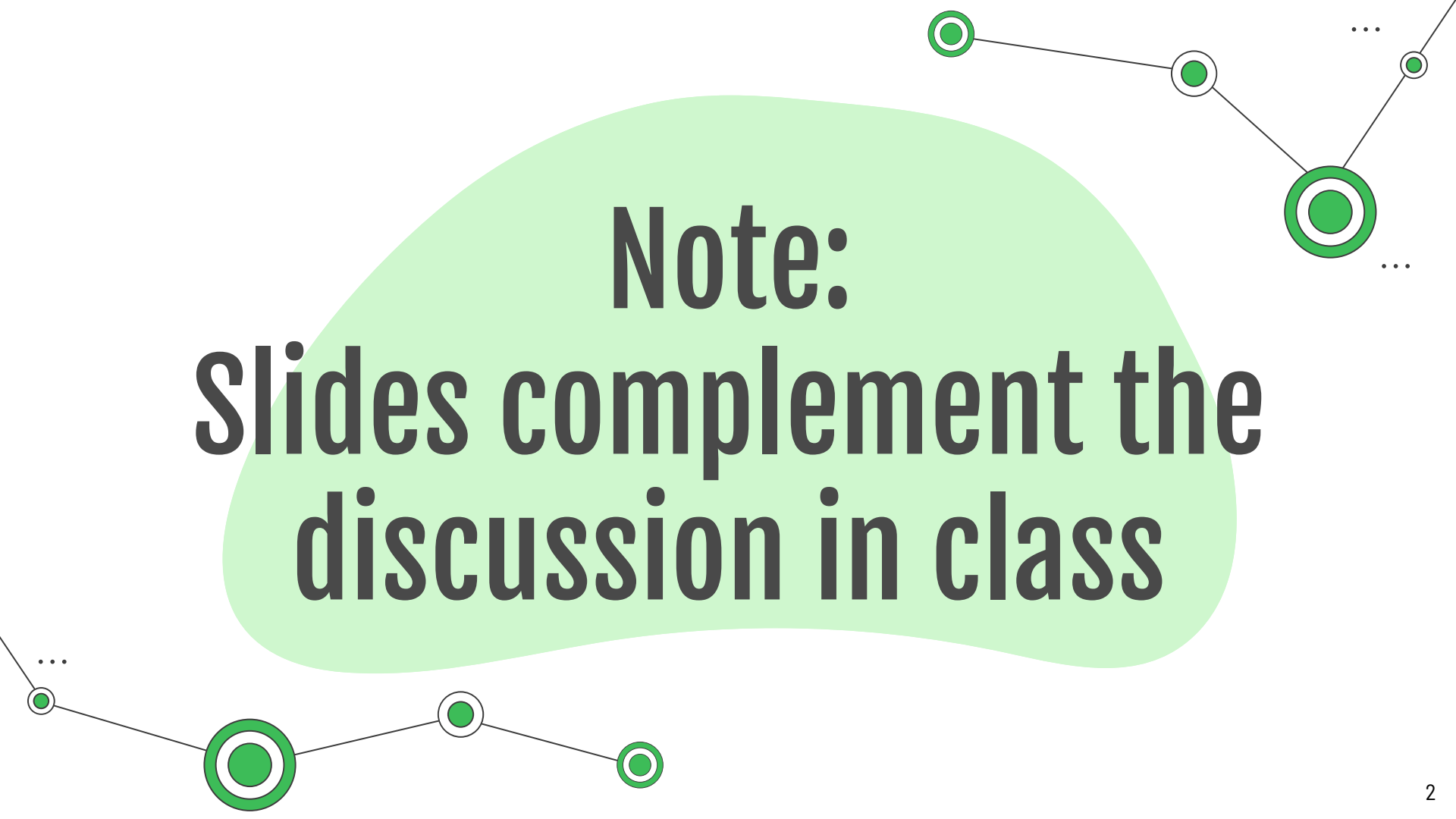




Insertion in Red-Black Trees

CS 251 - Data Structures
and Algorithms

A decorative network diagram consisting of several green circular nodes connected by thin black lines. Some nodes are single green circles, while others are double green circles. The nodes are arranged in a non-linear fashion, with some at the top right and others at the bottom left. Ellipses (...) are placed near some nodes to indicate a larger network.

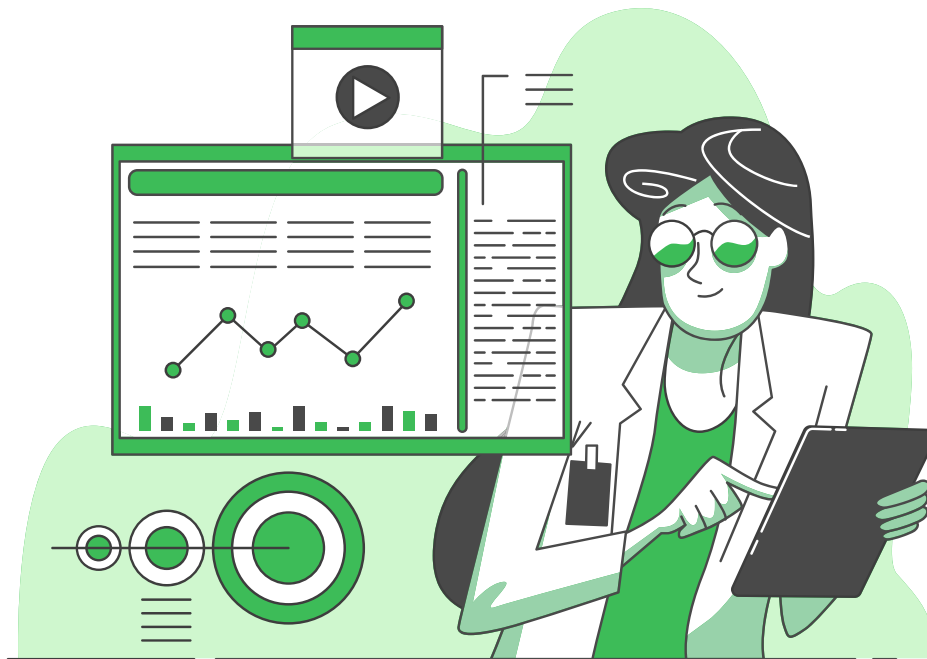
Note:
**Slides complement the
discussion in class**



Insert in Red-Black Trees

Enforcing "balance" after an insertion

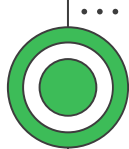
Table of Contents



01

Insert in Red-Black Trees

Enforcing "balance" after an insertion

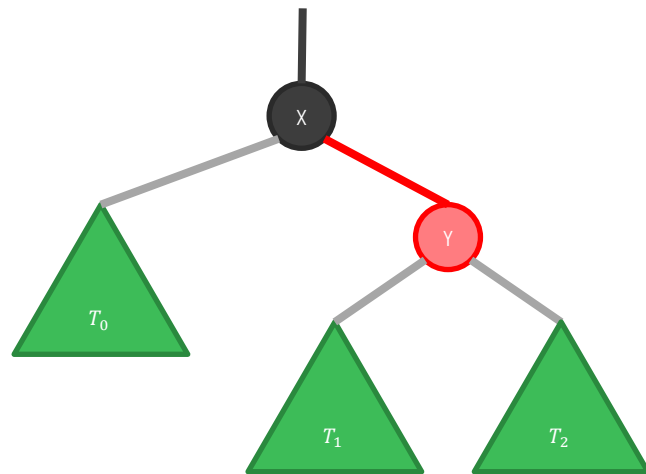
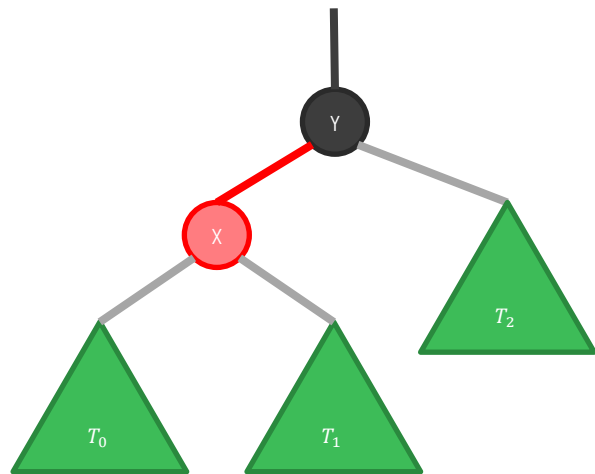


...

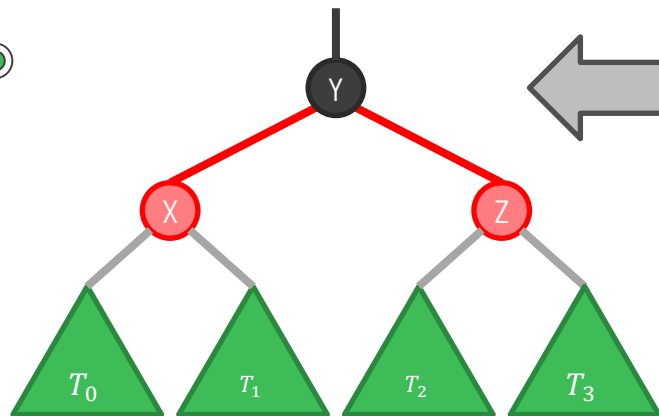
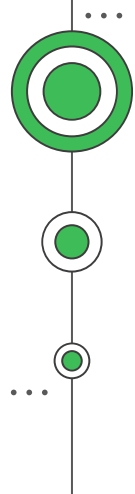


...

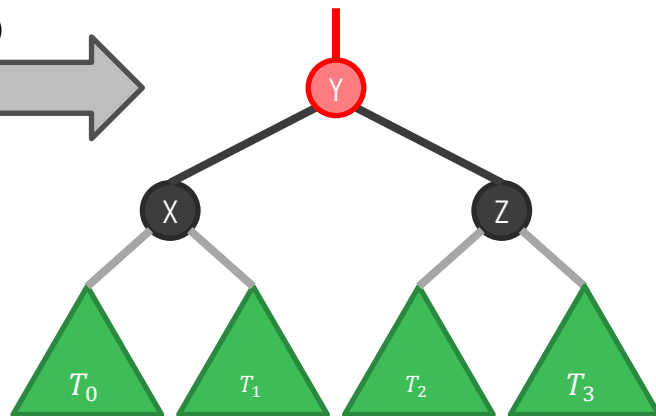
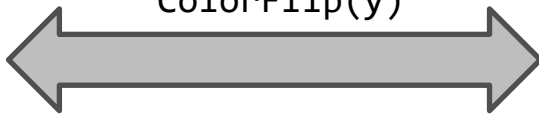
RightRotation(y)

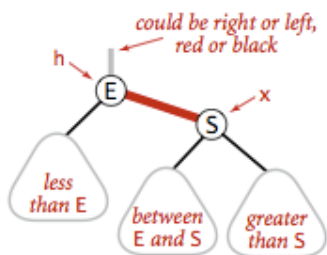


LeftRotation(x)

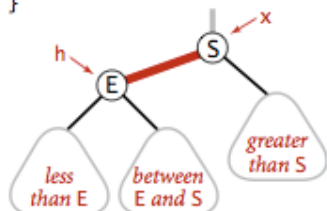


ColorFlip(y)

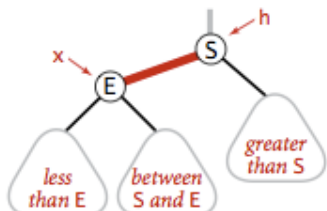




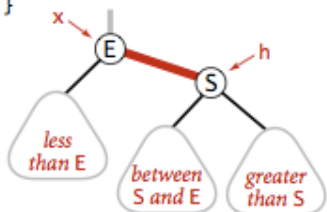
```
Node rotateLeft(Node h)
{
    Node x = h.right;
    h.right = x.left;
    x.left = h;
    x.color = h.color;
    h.color = RED;
    x.N = h.N;
    h.N = 1 + size(h.left)
        + size(h.right);
    return x;
}
```



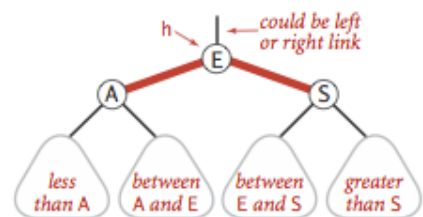
Left rotate (right link of h)



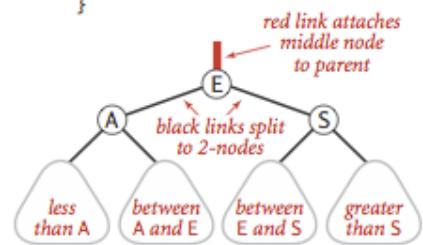
```
Node rotateRight(Node h)
{
    Node x = h.left;
    h.left = x.right;
    x.right = h;
    x.color = h.color;
    h.color = RED;
    x.N = h.N;
    h.N = 1 + size(h.left)
        + size(h.right);
    return x;
}
```



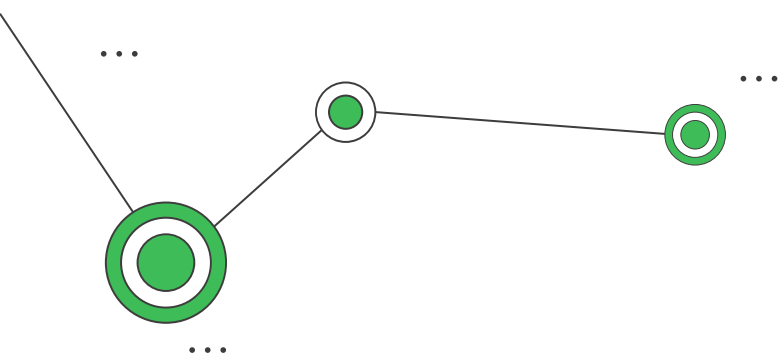
Right rotate (left link of h)



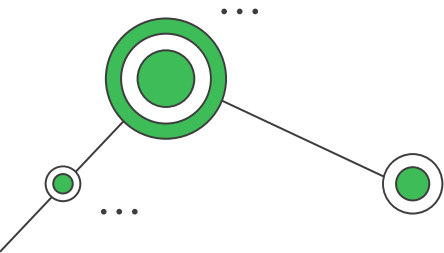
```
void flipColors(Node h)
{
    h.color = RED;
    h.left.color = BLACK;
    h.right.color = BLACK;
}
```



Flipping colors to split a 4-node



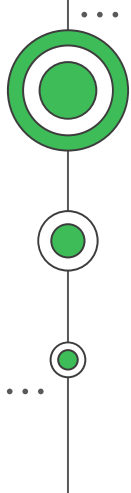
Insert(k)



Insert the key as usual in BSTs.

New nodes are always **red** (it doesn't upset the black height).

Use transformations to **rebalance the black height** of the RBT.



Insert 5, 4, 3, 2, 1 in a Red-Black Tree:

Insert 5:

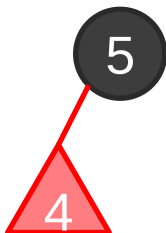


Root Color Flip



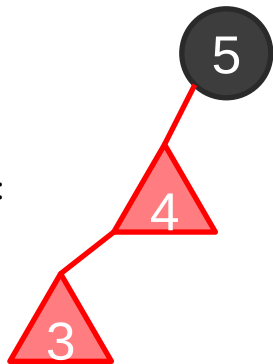
Height = 0
Black height = 1

Insert 4:

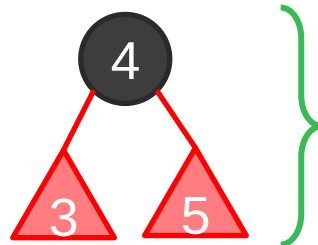


Height = 1
Black height = 1

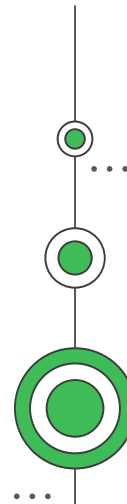
Insert 3:

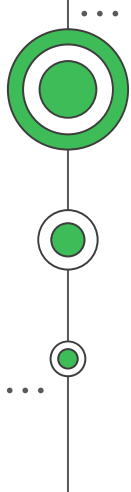


rightrotation(5)

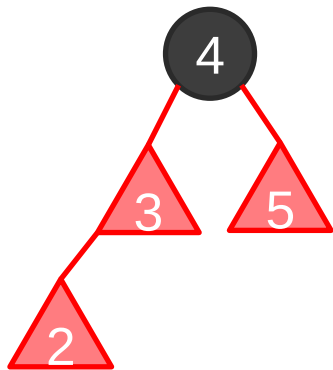


Height = 1
Black height = 1

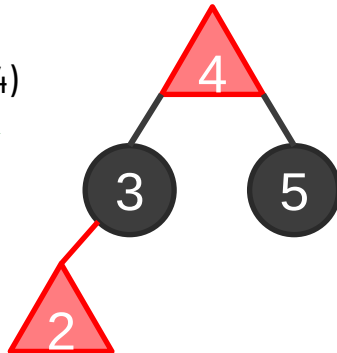




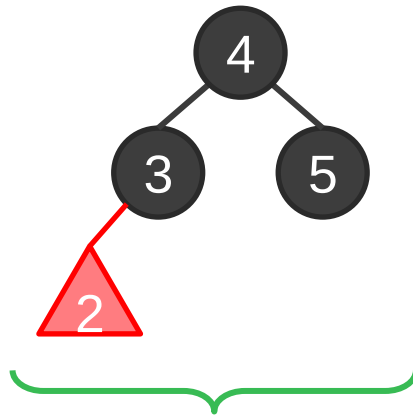
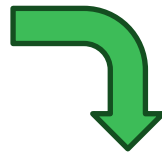
Insert 2:



colorflip(4)

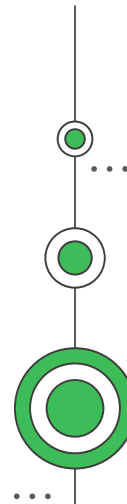


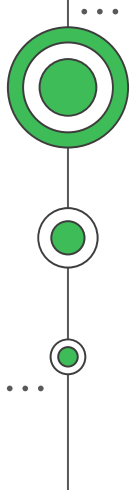
rootcolorflip()



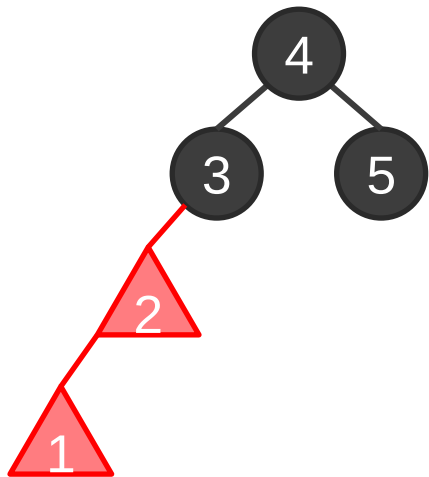
Height = 2

Black height = 2

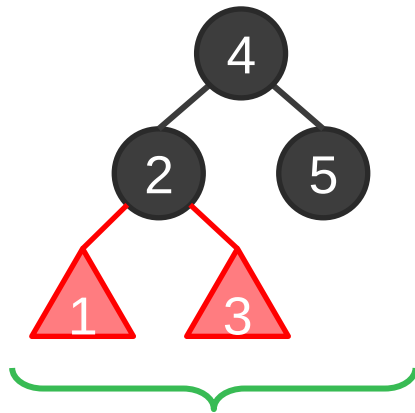




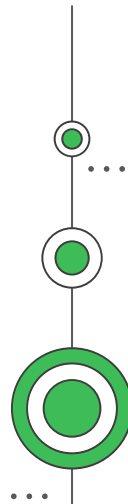
Insert 1:



rightrotation(3)



Height = 2
Black height = 2



Data Structure Visualizations

About
Algorithms
F.A.Q
Known Bugs /
Feature Requests
Java Version
Flash Version
Create Your Own /
Source Code
Contact

David Galles
Computer Science
University of San
Francisco

Currently, we have visualizations for the following data structures and algorithms:

- Basics
 - Stack: Array Implementation
 - Stack: Linked List Implementation
 - Queues: Array Implementation
 - Queues: Linked List Implementation
 - Lists: Array Implementation (available in java version)
 - Lists: Linked List Implementation (available in java version)
- Recursion
 - Factorial
 - Reversing a String
 - N-Queens Problem
- Indexing
 - Binary and Linear Search (of sorted list)
 - Binary Search Trees
 - AVL Trees (Balanced binary search trees)
 - Red-Black Trees
 - Splay Trees
 - Open Hash Tables (Closed Addressing)
 - Closed Hash Tables (Open Addressing)
 - Closed Hash Tables, using buckets
 - Trie (Prefix Tree, 26-ary Tree)
 - Radix Tree (Compact Trie)
 - Ternary Search Tree (Trie with BST of children)
 - B Trees
 - B+ Trees
- Sorting
 - Comparison Sorting
 - Bubble Sort
 - Selection Sort
 - Insertion Sort
 - Shell Sort
 - Merge Sort
 - Quick Sort
 - Bucket Sort
 - Counting Sort
 - Radix Sort
 - Heap Sort
- Heap-like Data Structures
 - Heaps
 - Binomial Queues
 - Fibonacci Heaps
 - Leftist Heaps
 - Skew Heaps
- Graph Algorithms
 - Breadth-First Search
 - Depth-First Search
 - Connected Components
 - Dijkstra's Shortest Path
 - Prim's Minimum Cost Spanning Tree
 - Topological Sort (Using Indegree array)
 - Topological Sort (Using DFS)

Visit <https://www.cs.usfca.edu/~galles/visualization/Algorithms.html> to
practice insertions in Red-Black Trees.

Done!

Do you have any questions?

CREDITS: This presentation template was created by [Slidesgo](#), including icons by [Flaticon](#), infographics & images by [Freepik](#) and illustrations by [Stories](#)

