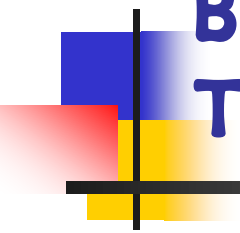


Data Structures:
Binary Search Trees, Height-Balanced Trees, AVL Tree

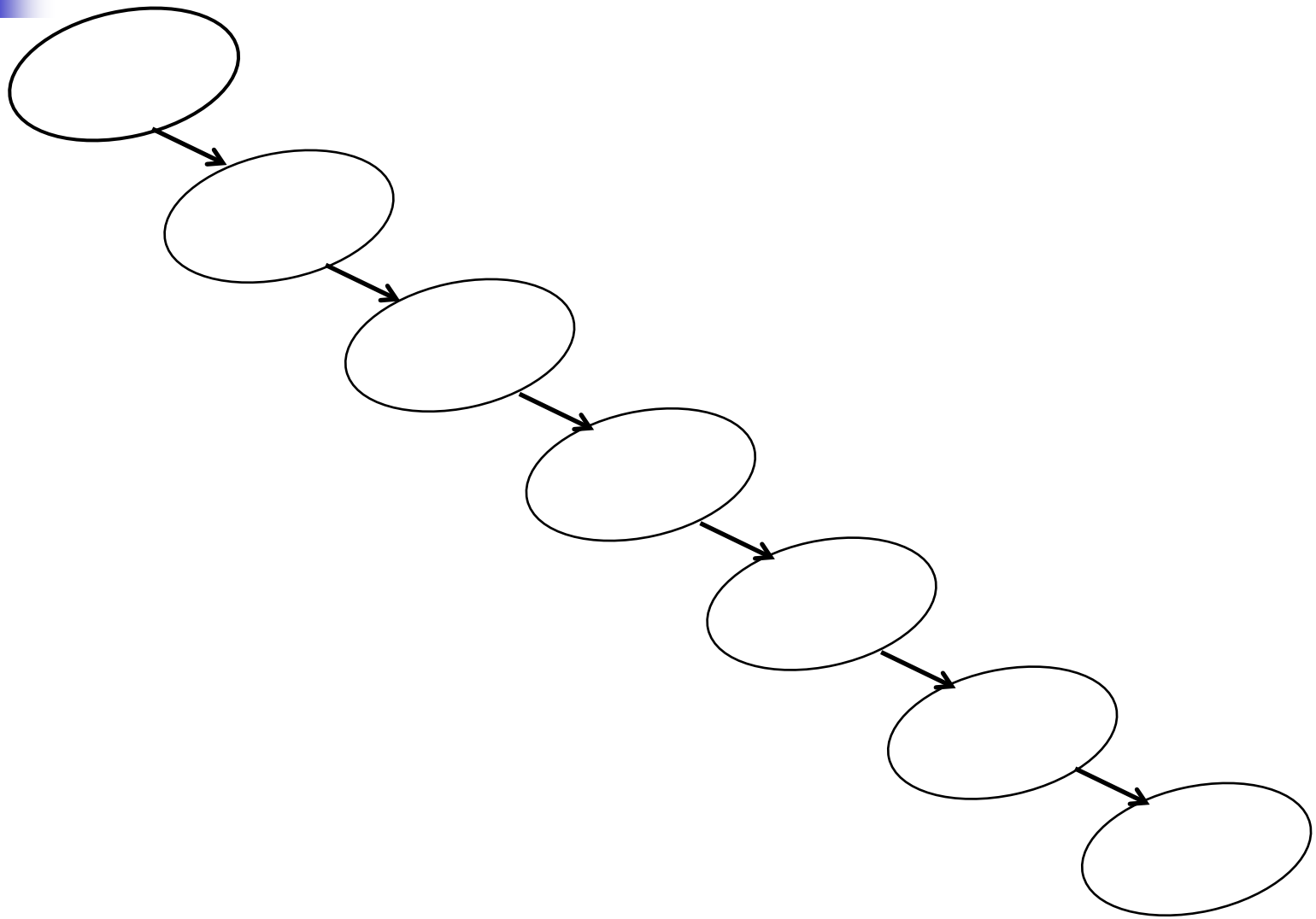
YoungWoon Cha
(Slide credits to Won Kim)
Spring 2022

Performance of BST

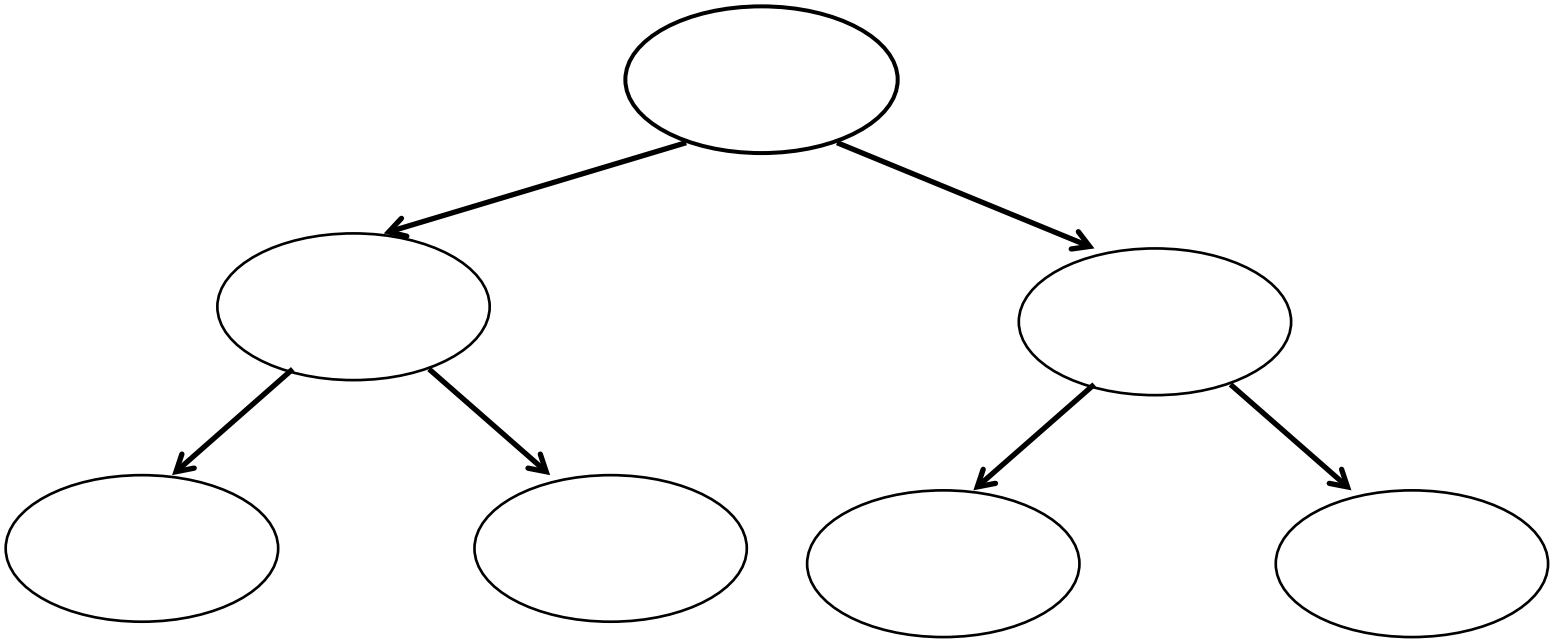
Performance of Binary Search Trees

- Search Time
- Insertion Time
- Deletion Time

Degenerate Binary Tree



A Full or Complete Binary Tree



Big O (Big Oh, Landau, Asymptotic) Notation (1/3)

- (e.g.) $T(n) = 2n^2 + 4n + 120$
  - As n grows, the n^2 term dominates
  - $T(n) = O(n^2)$
- "Big O of n^2" = $O(n^2)$
  - "Order of n^2" time complexity"

- Think of "Big" n (1,000; 1,000,000;…)

Big O (Big Oh, Landau, Asymptotic) Notation (2/3)

- O(1) : (constant and small) finding a match right away in a search, deletion of the largest key from a max heap
- $O(\log_2 n)$: (logarithmic) search of a height-balanced binary search tree, binary search of a sorted list
- O(n) : search of a binary search tree
- O(n) : (linear) search of an array, queue, linked list

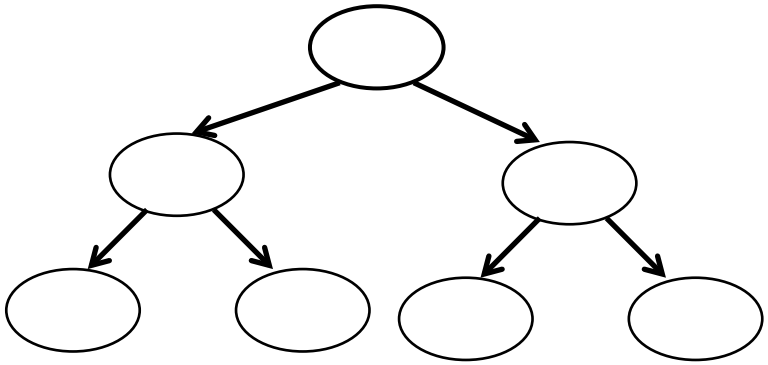
Big O (Big Oh, Landau, Asymptotic) Notation (3/3)

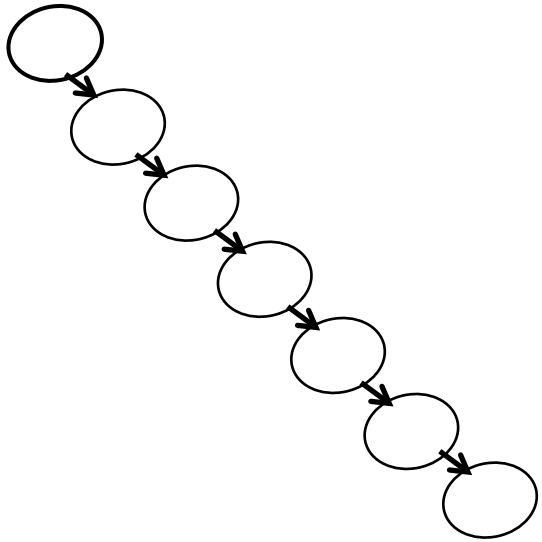
- $O(n \log_2 n)$: (loglinear) heap sort
- $O(n^2)$: (quadratic) insertion sort

- $O(n!)$: (factorial) traveling salesman problem
- $2^{O(n)}$: (exponential) general traveling salesman problem

Source of the Performance Problem of Binary Search Trees

- Height of the Binary Search Tree
  - The average case: $\log_2 n = h$, where n is the total number of nodes
  - The worst case: n

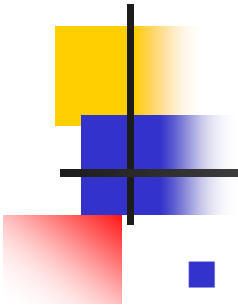




Source of the Performance Problem of Binary Search Trees

- Need to Maintain the Search Tree Height-Balanced.
  - AVL tree, red-black tree, AA tree,…
  - Search, insertion, deletion each takes $O(\log_2 n)$
  - Overhead for insertion and deletion

A Height-Balanced Tree



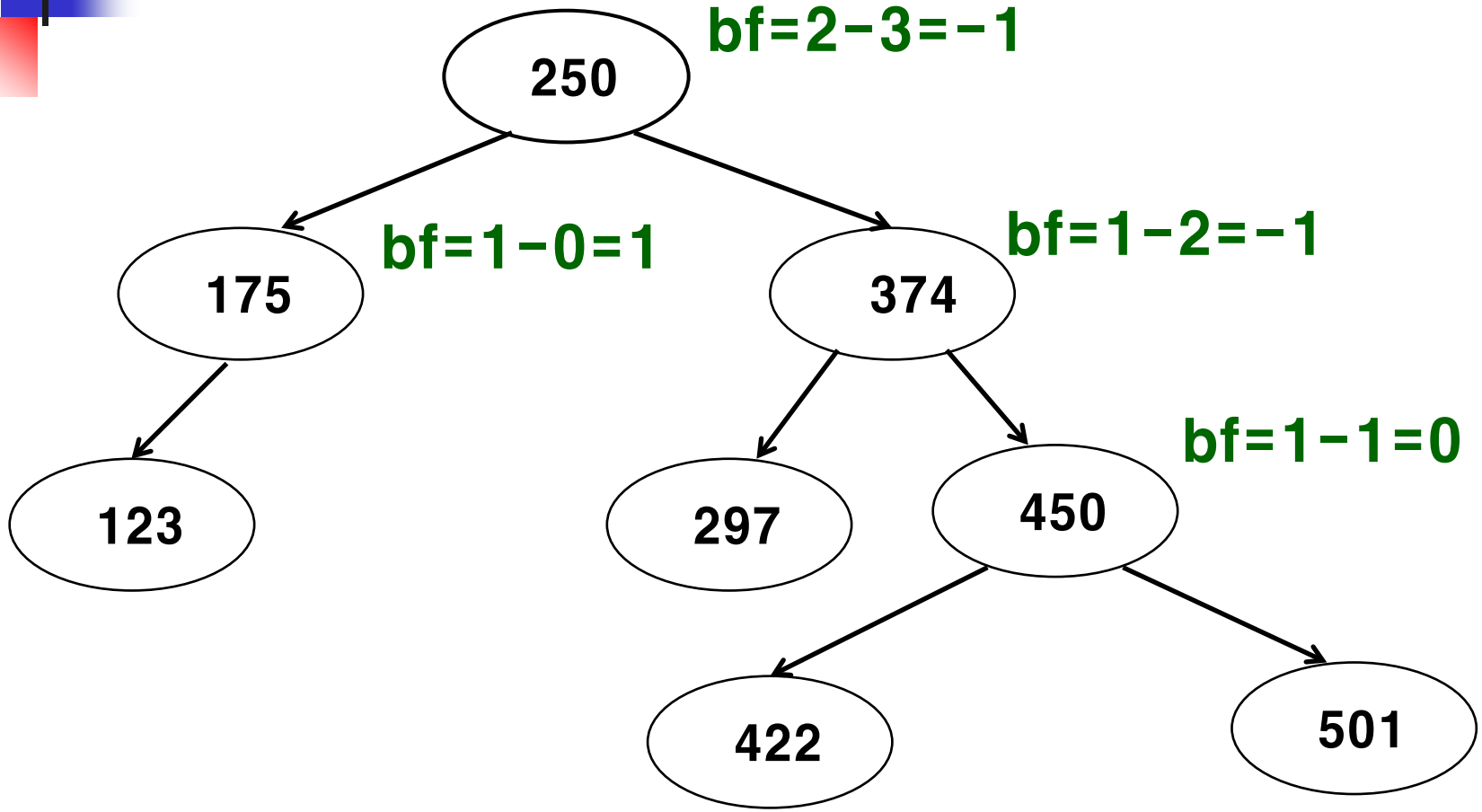
Height-Balanced Tree

- AVL Tree, (Red-Black Tree)
- (2-3 Tree, T-Tree, B-Tree, B+ Tree)

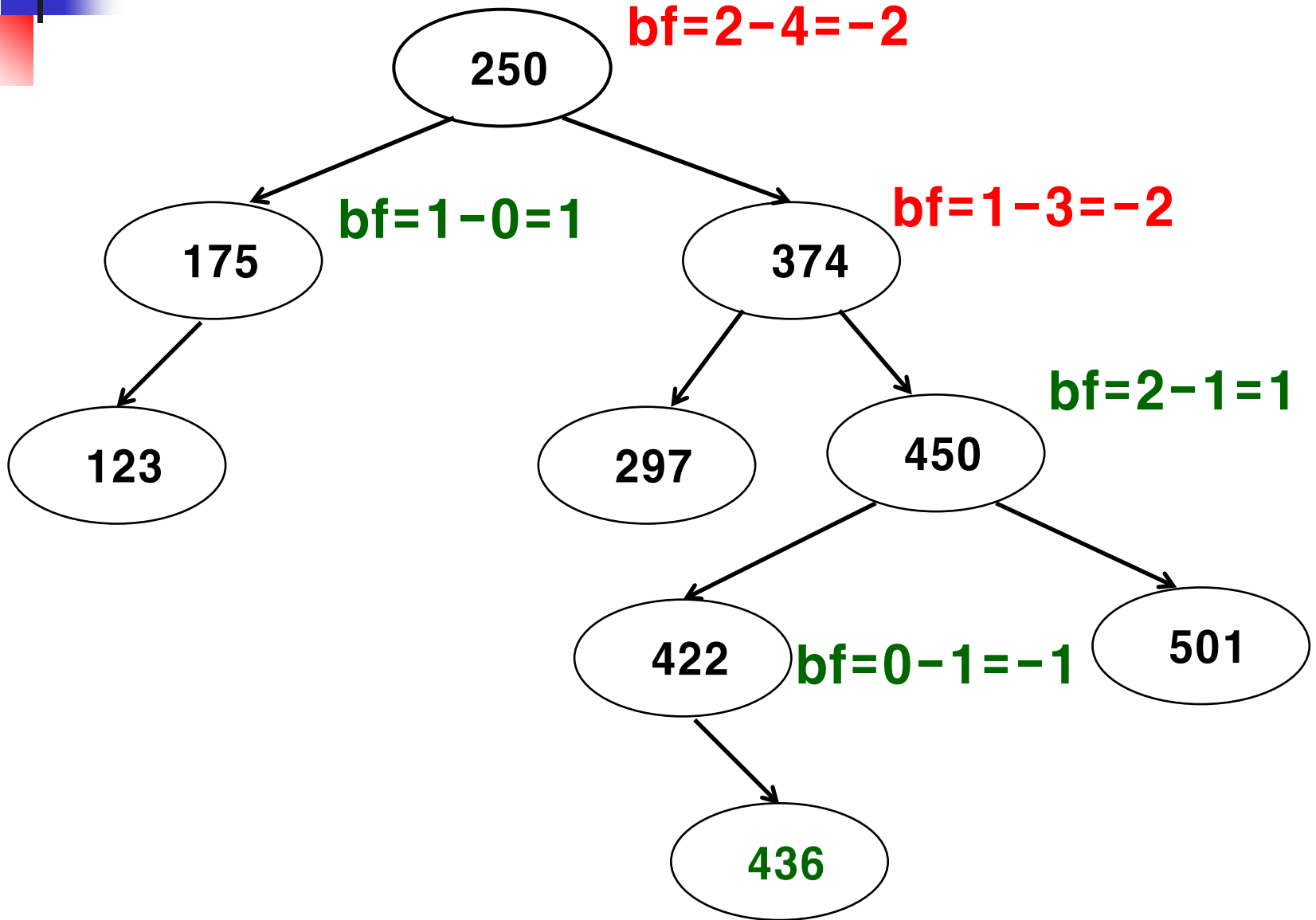
AVL Tree

- Invented by G.M. Adelson-Velskii and E.M. Landis (1962)
- A Height-Balanced Binary Search Tree
  - "not perfectly balanced"
- Balance Factor of Each Node Must Be 0, +1, or -1.
  - (Balance Factor = $h^L - h^R$, where h^L is the height of left subtree and h^R is the height of the right subtree.)

An AVL Tree

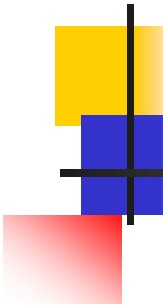


Not an AVL Tree



AVL Tree Rotations

- If the Balance Factor of a Node Becomes 2 or -2, Upon Insertion or Deletion of a Node
  - The tree must be re-balanced by performing “tree rotations” around the node whose balance factor becomes 2 or -2.
- Two Objectives
  - Maintain AVL tree height balanced.
  - Maintain AVL tree as a binary search tree.

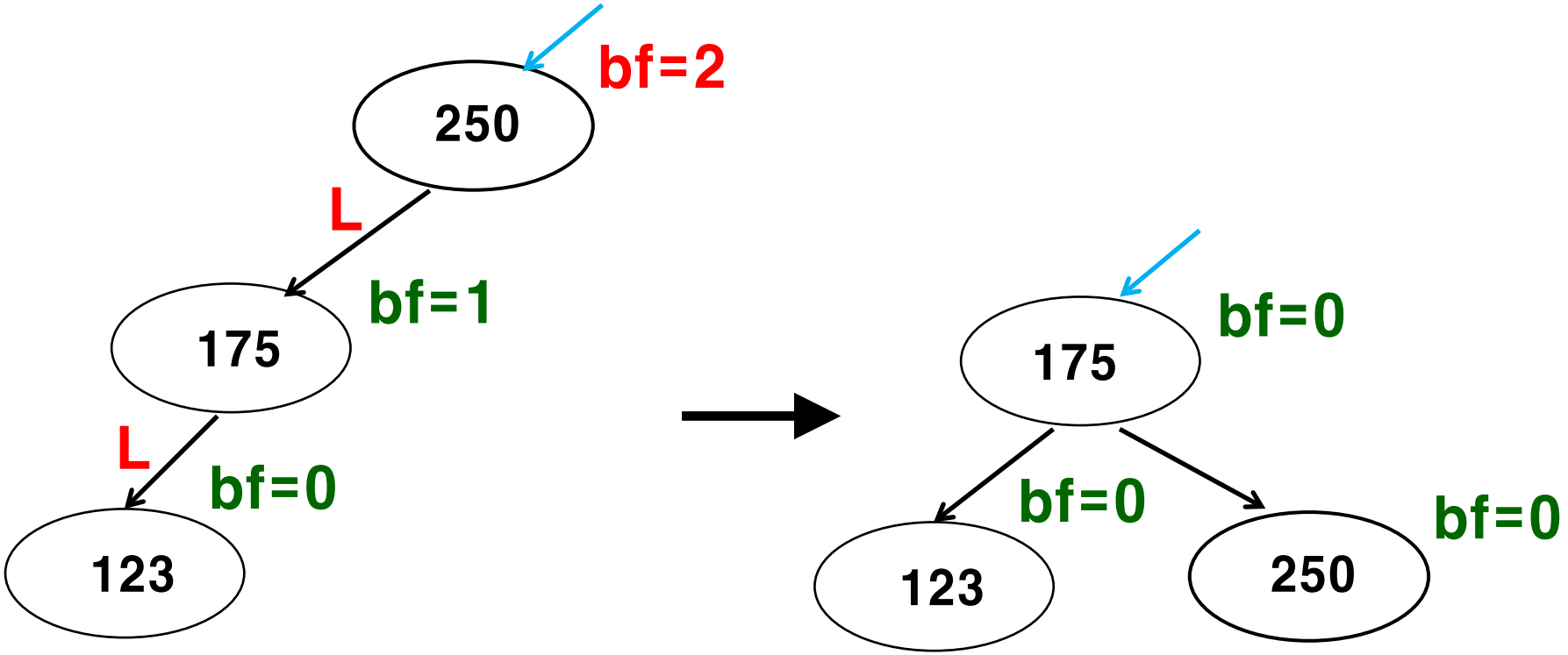


AVL Tree Rotations

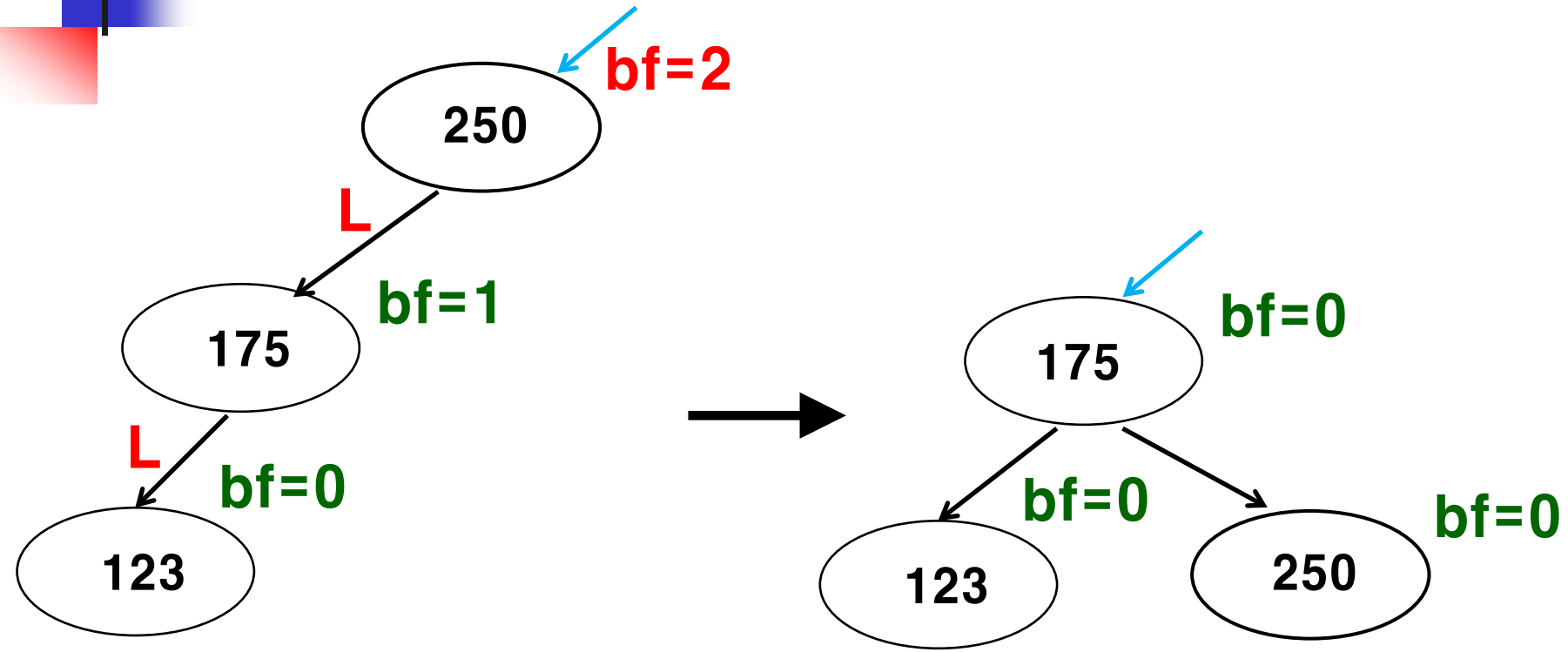
- 4 Types of Rotations
  - form a chain of 3 nodes down from the node for a single rotation (right or left), or
  - form a chain of 3 or 4 nodes down from the node for a double rotation (right and left, or left and right)

AVL Tree Rotations (1/4)

LL: Single Rotation (Right)



Implementing the LL Rotation

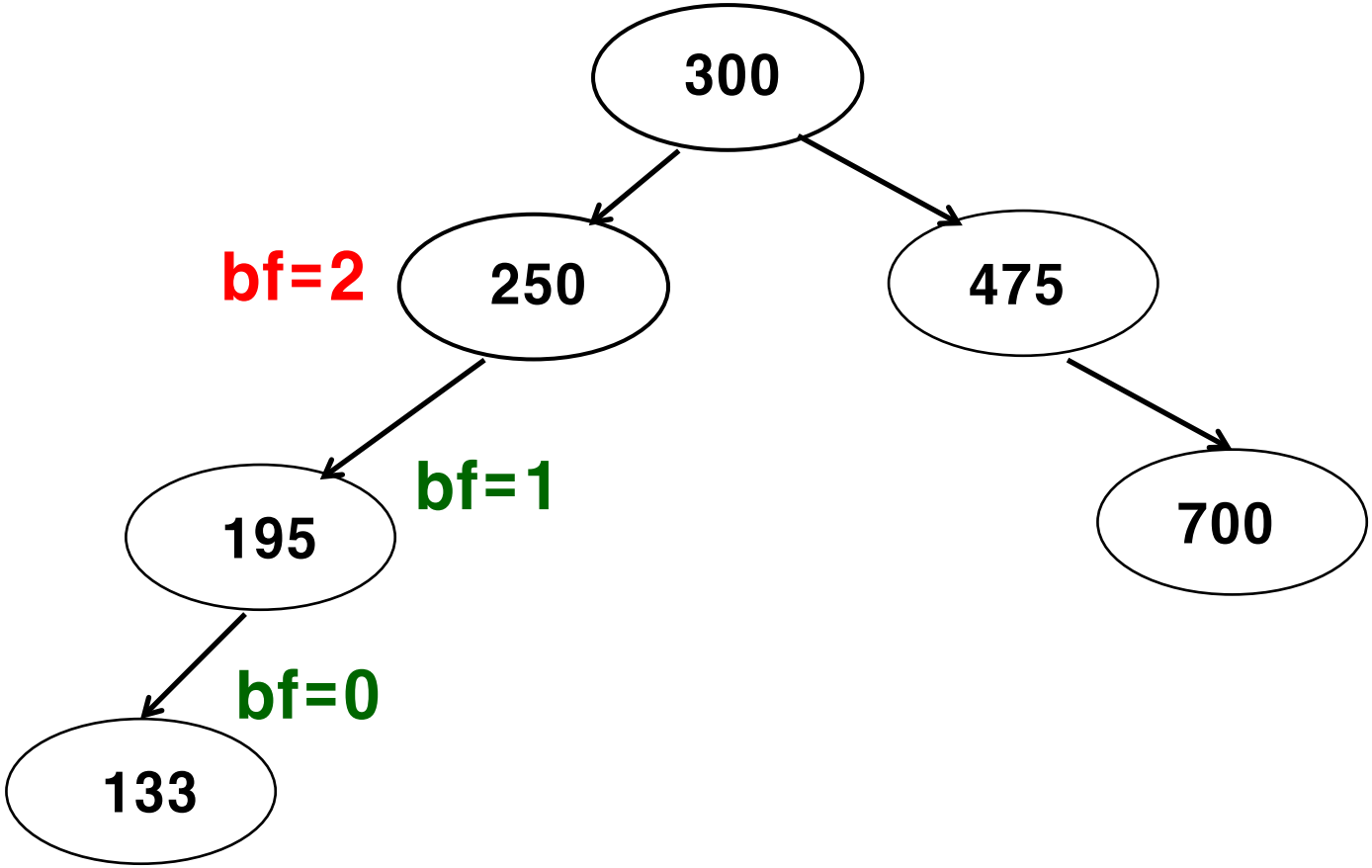


change node 175's right child pointer from NULL to node 250

change the left child pointer of the parent of node 250
from node 250 to node 175

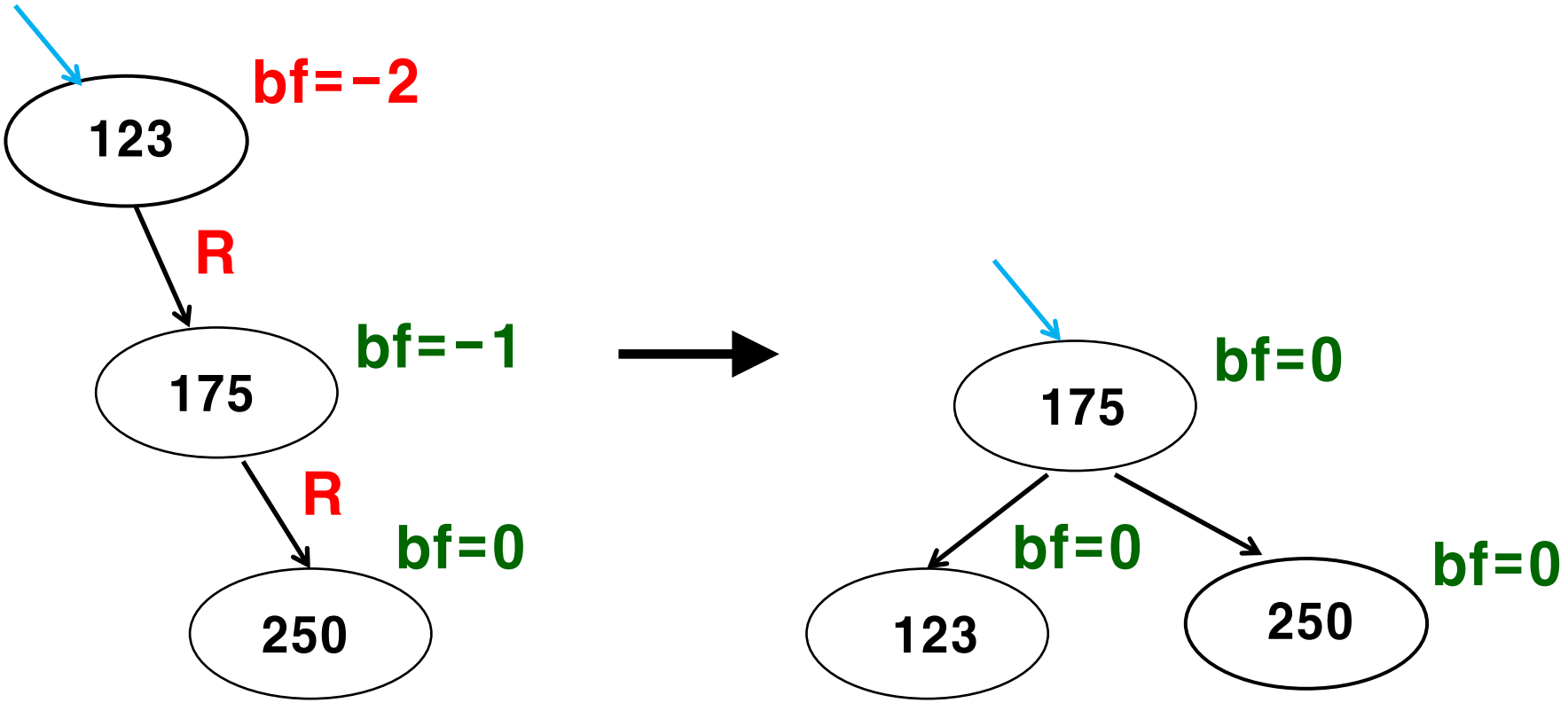
change the balance factor in the affected nodes

Exercise: Re-Balance the AVL Tree

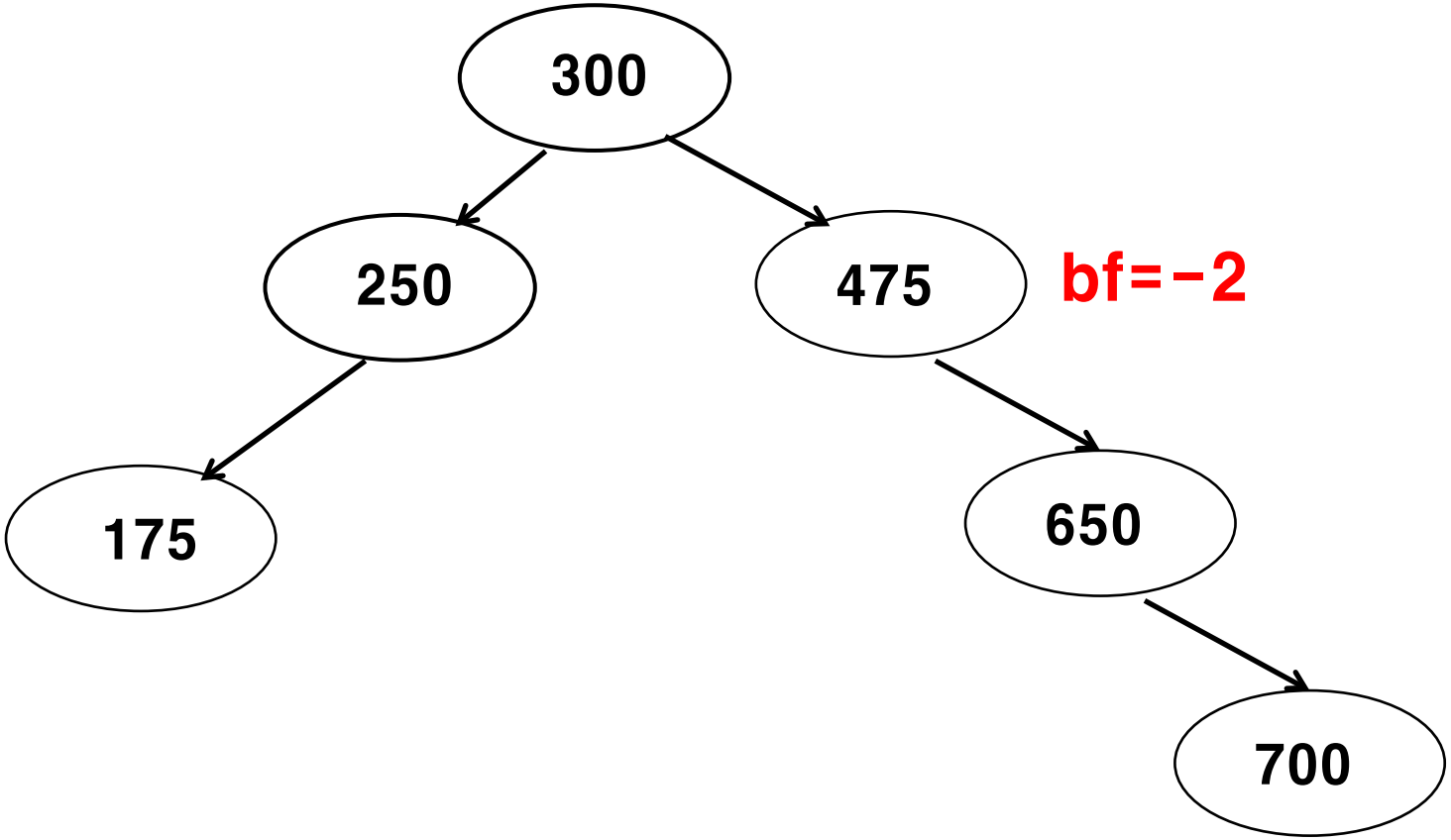


AVL Tree Rotations (2/4)

RR: Single Rotation (Left)

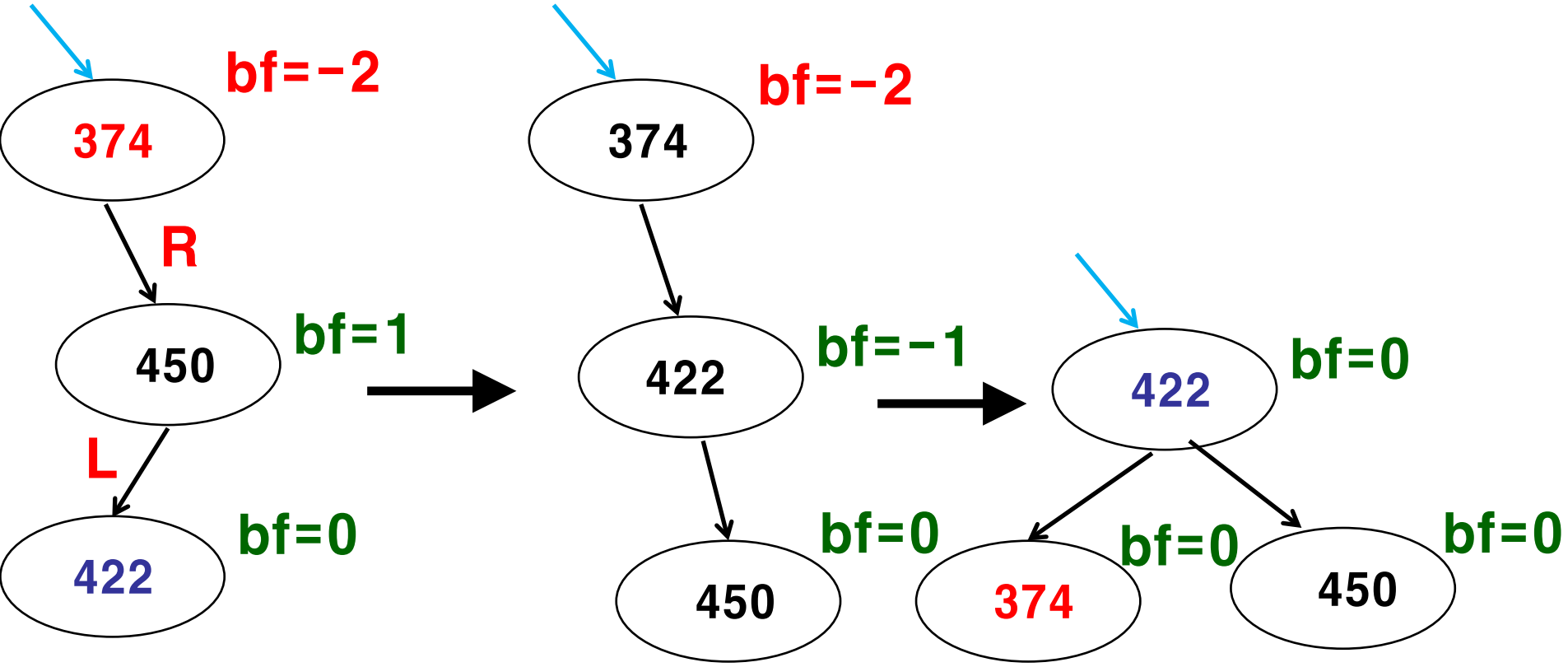


Exercise: Re-Balance the AVL Tree

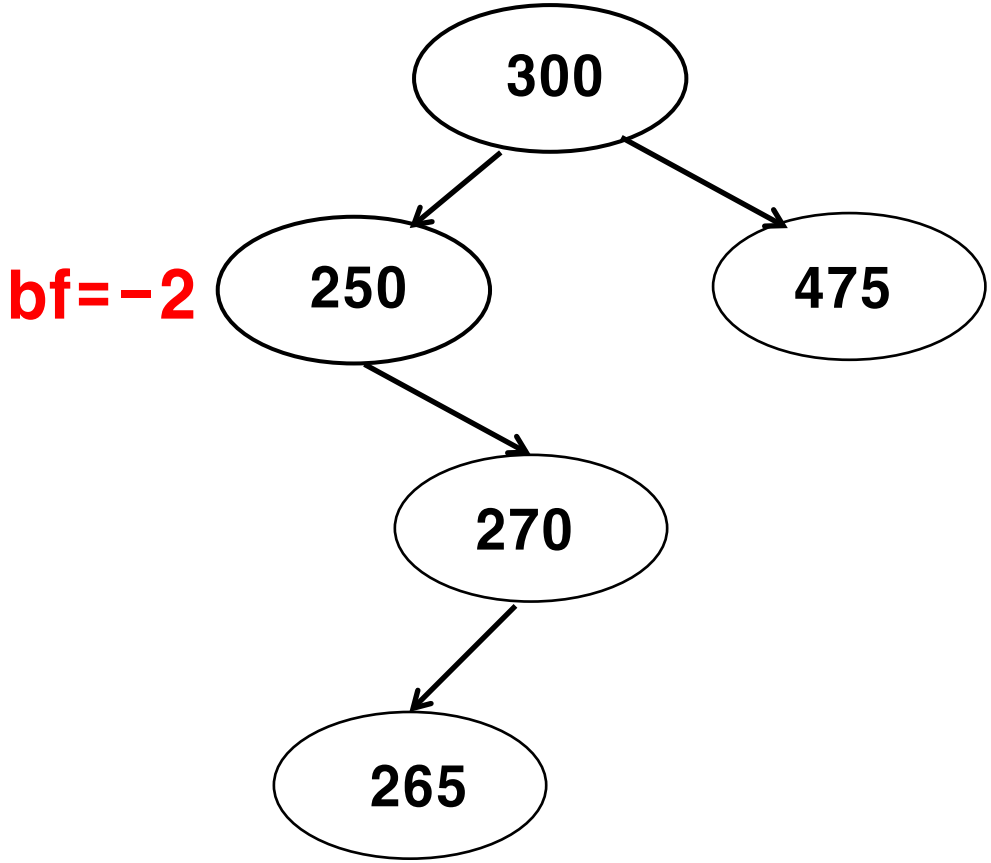


AVL Tree Rotations (3/4)

RL: Double Rotation (Right and Left)

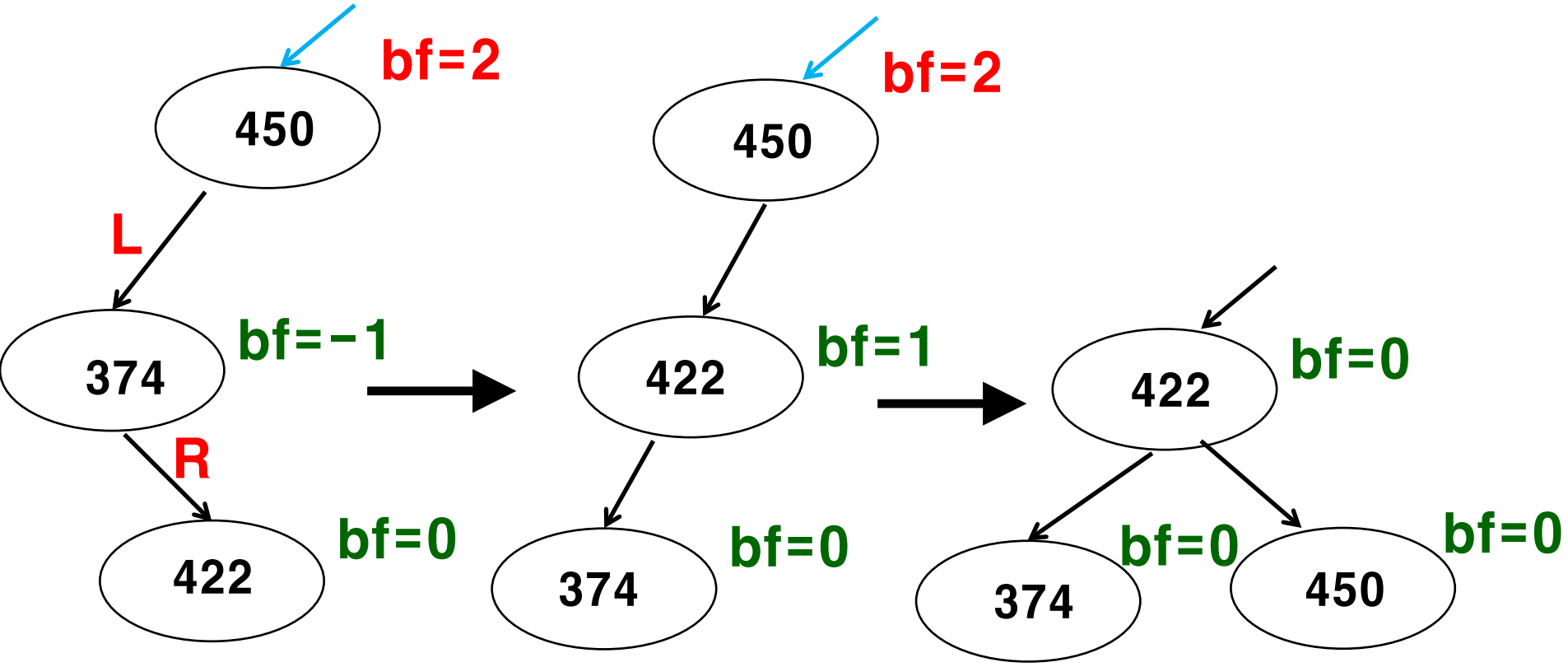


Exercise: Re-Balance the AVL Tree

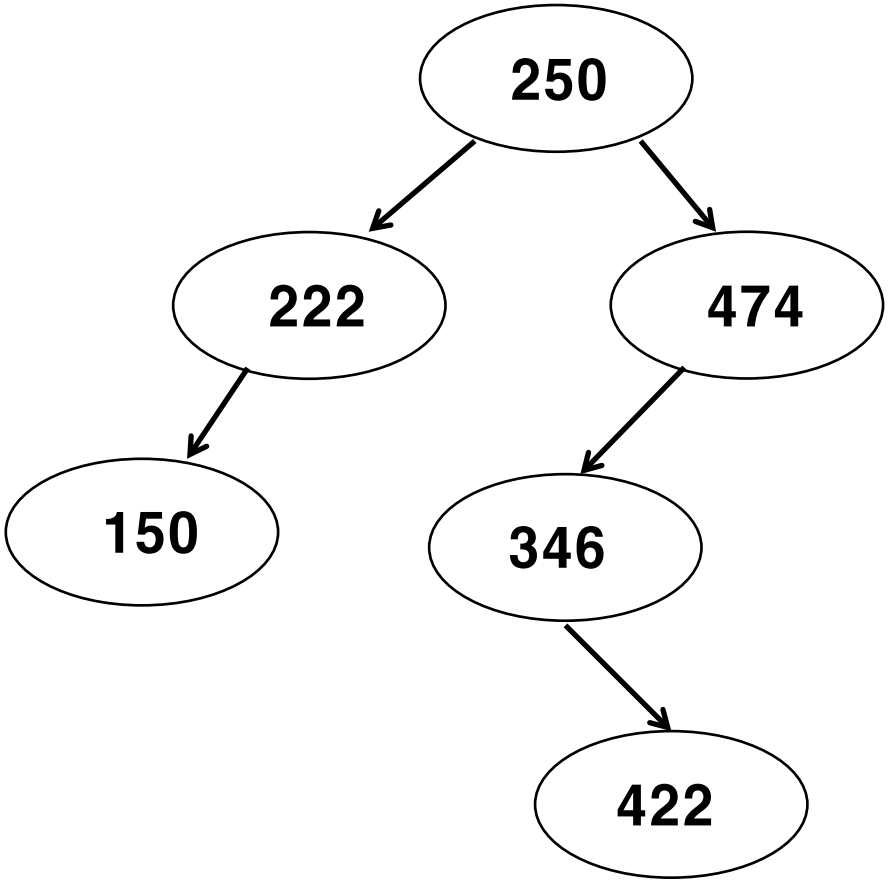


AVL Tree Rotations (4/4)

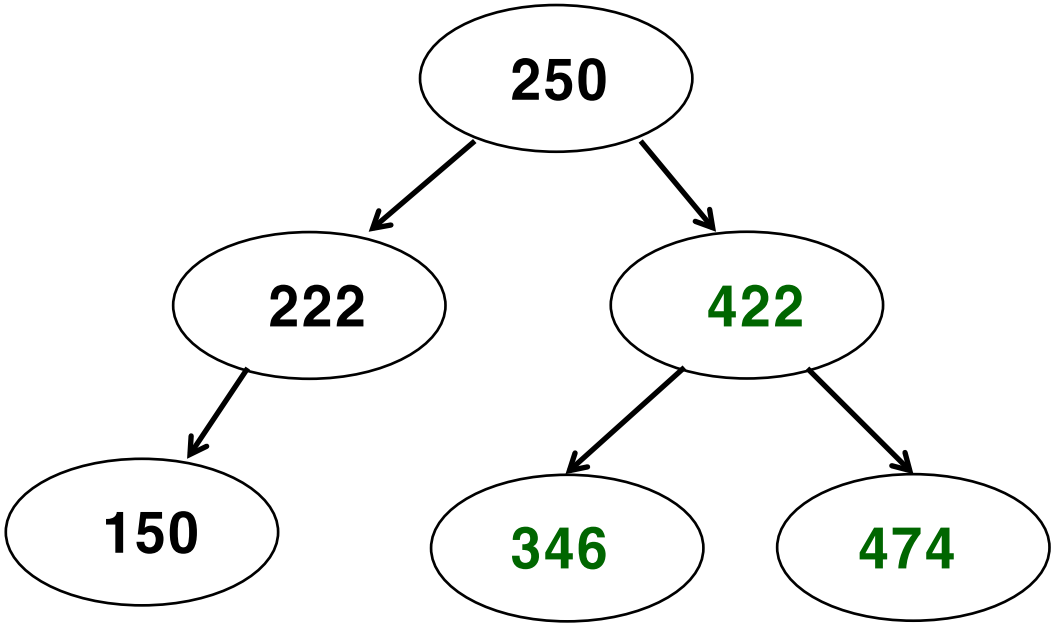
LR: Double Rotation (Left and Right)



Exercise: Rebalance the AVL Tree

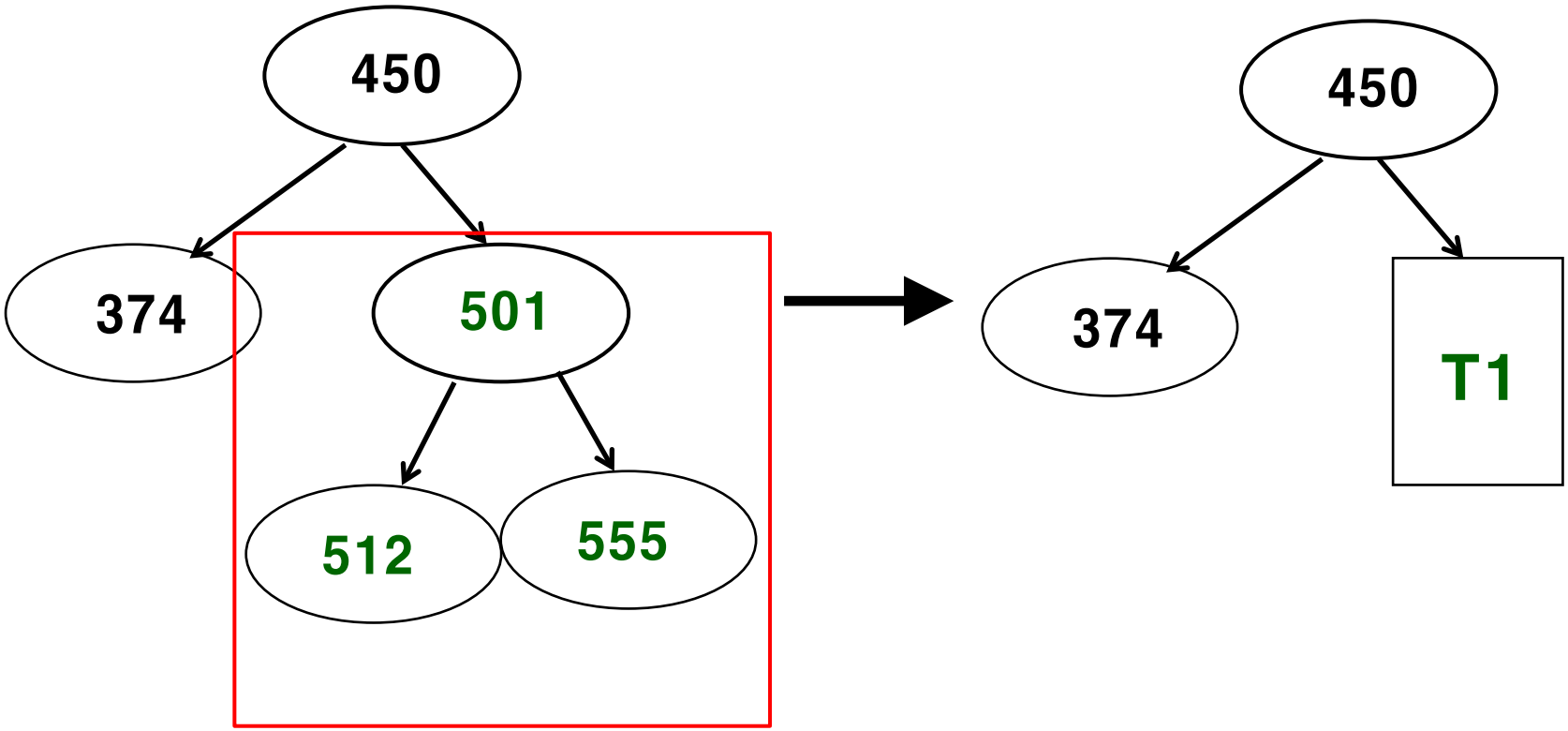


Exercise: Result



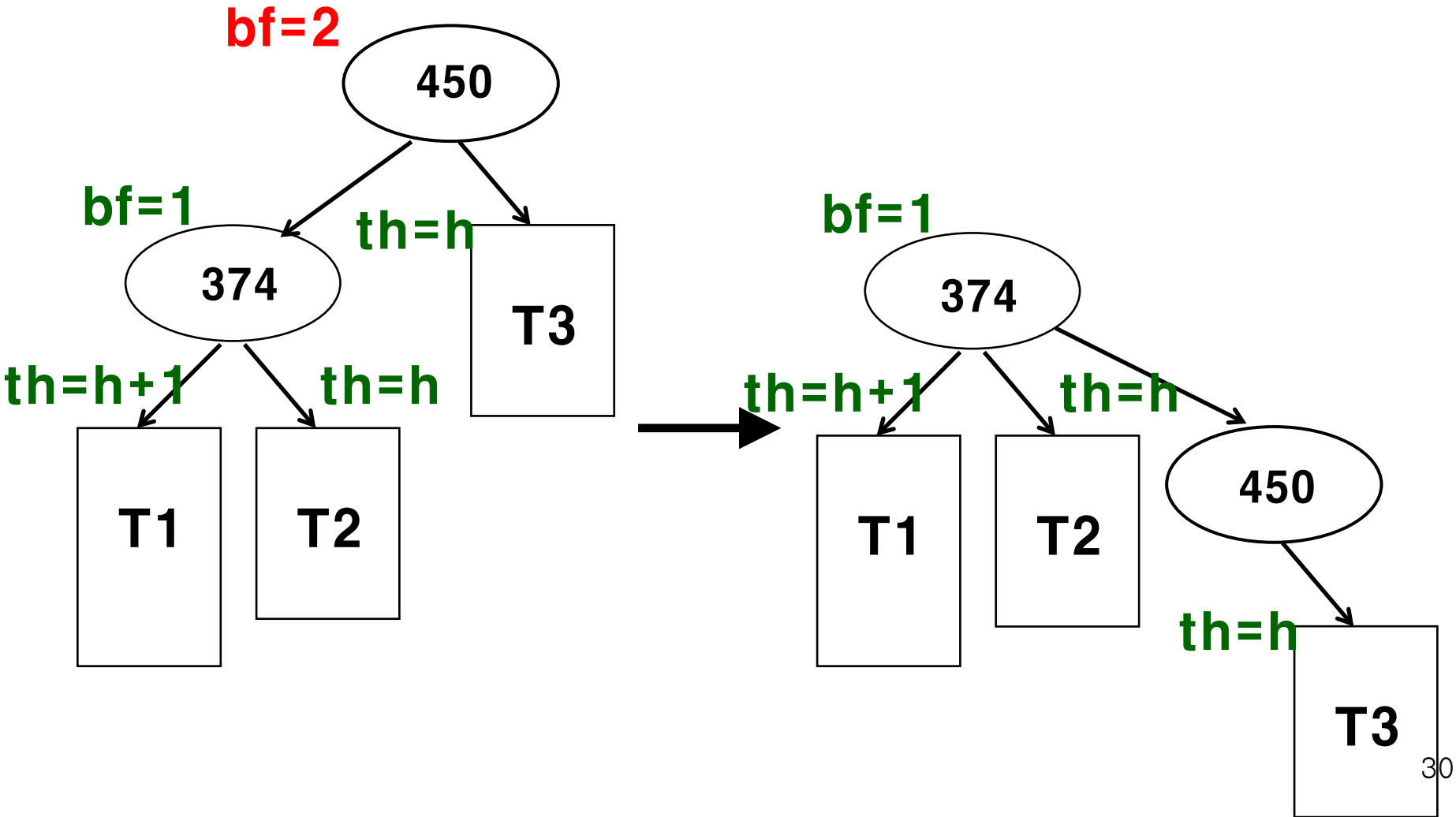
Rotation of an Entire Subtree

<Notation>



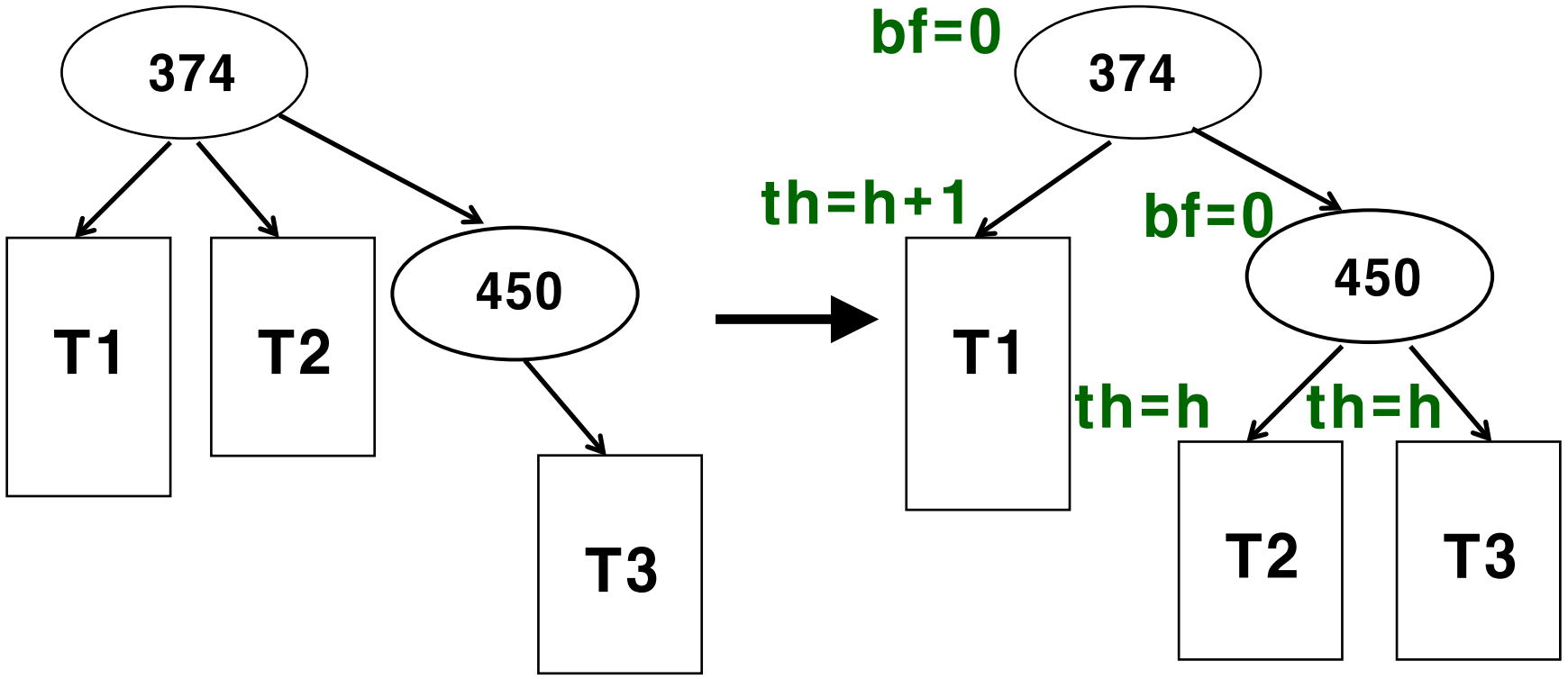
AVL Tree Rotations: General (1/4)

LL: Single Rotation (Right)



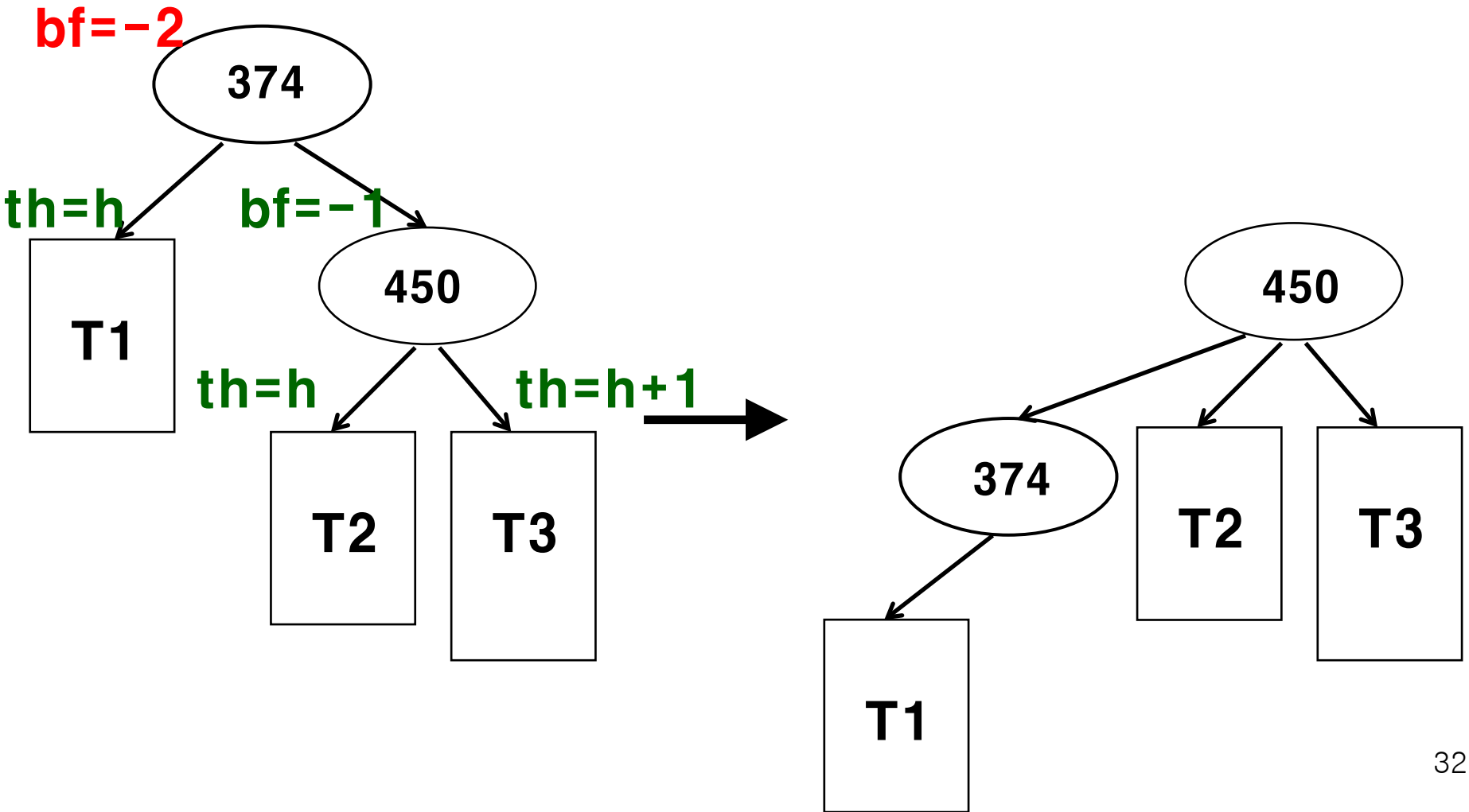
AVL Tree Rotations: General (1/4) cont'd

LL: Single Rotation (Right)



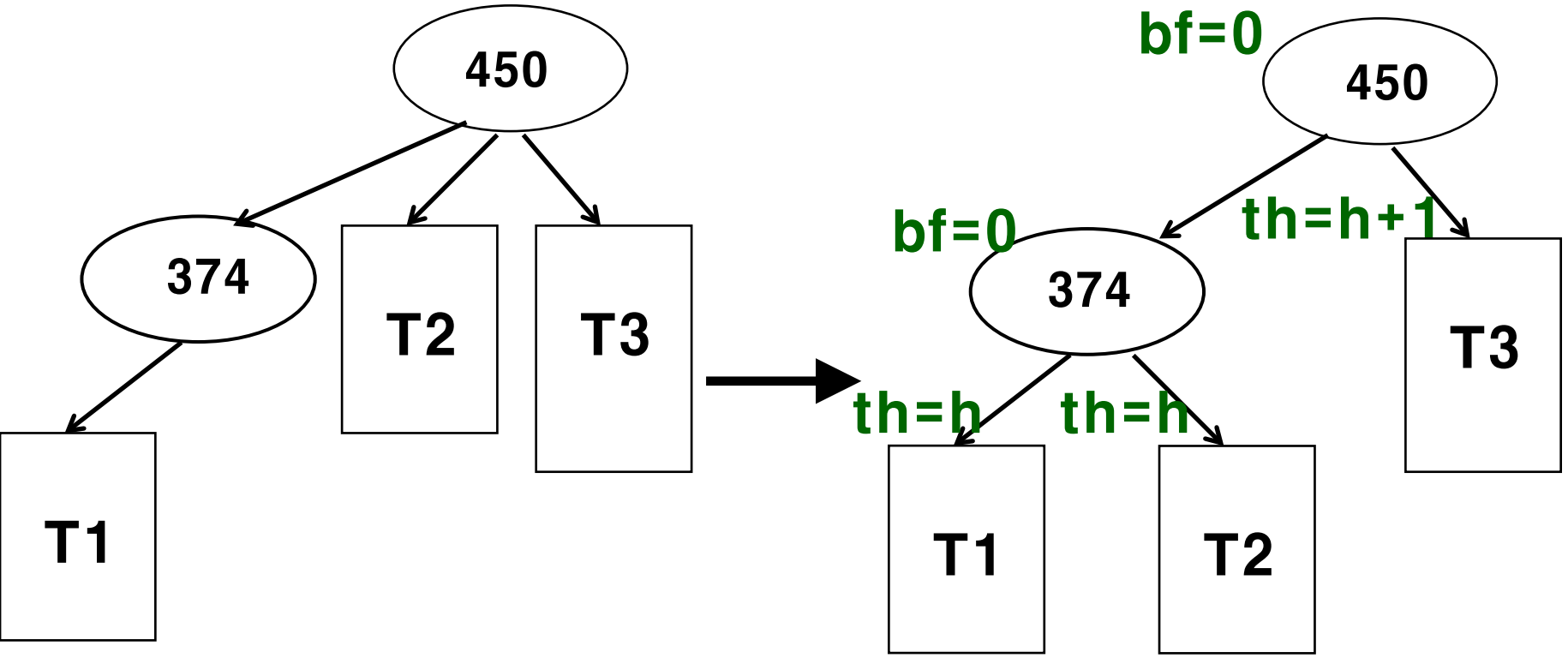
AVL Tree Rotations: General (2/4)

RR: Single Rotation (Left)



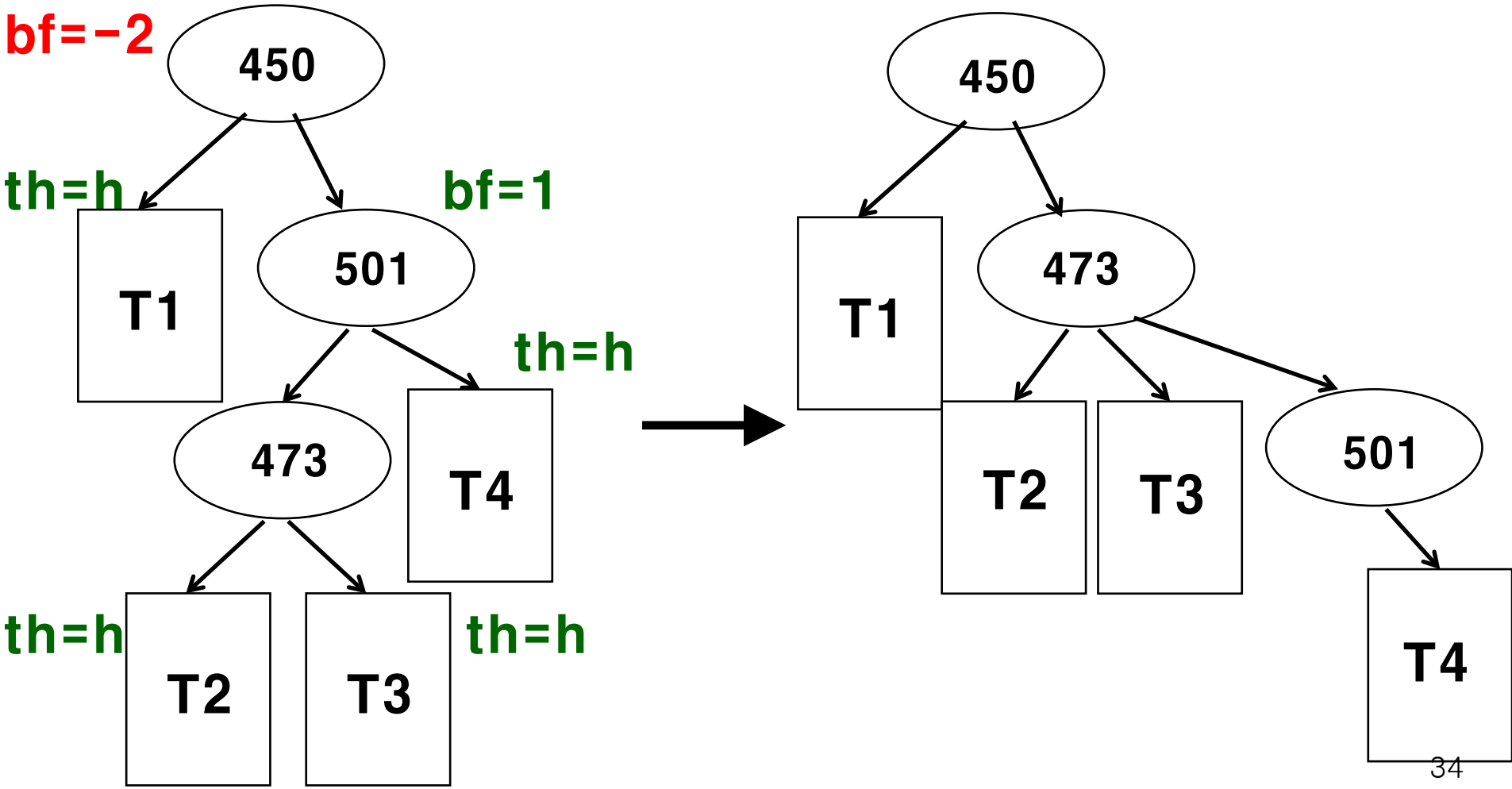
AVL Tree Rotations: General (2/4) cont'd

RR: Single Rotation (Left)



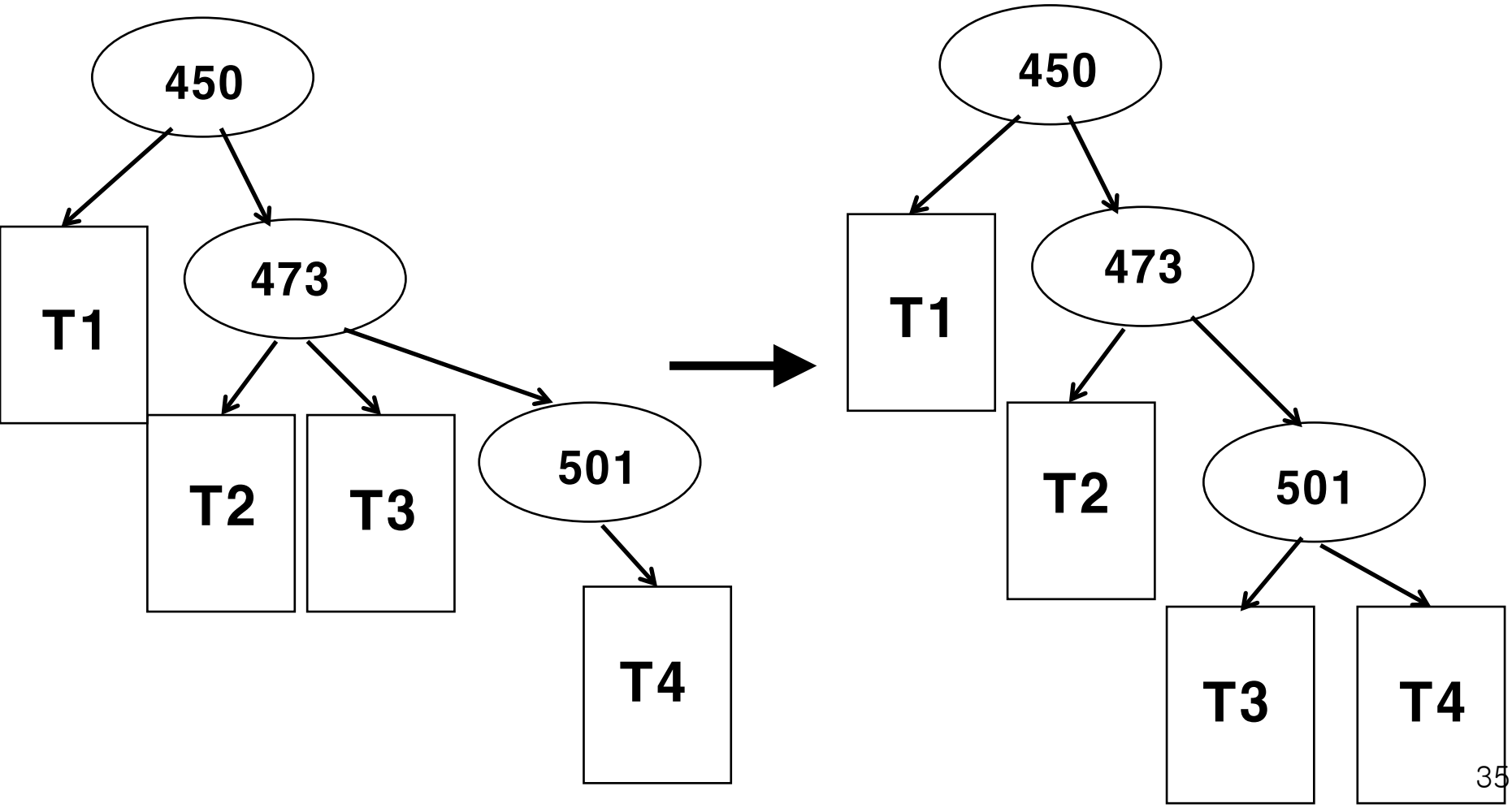
AVL Tree Rotations: General (3/4)

RL: Double Rotation (Right and Left)



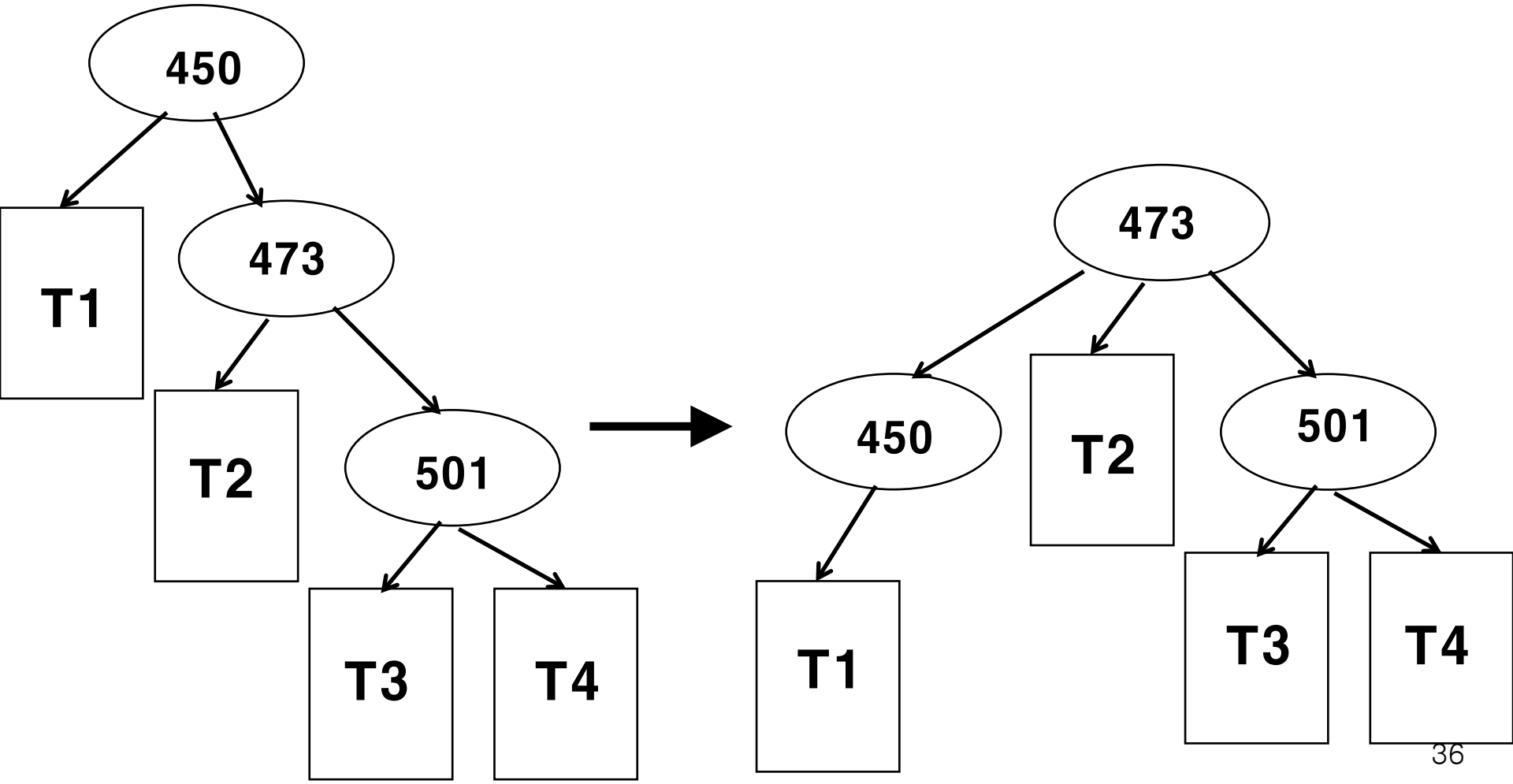
AVL Tree Rotations: General (3/4) cont'd

RL: Double Rotation (Right and Left)



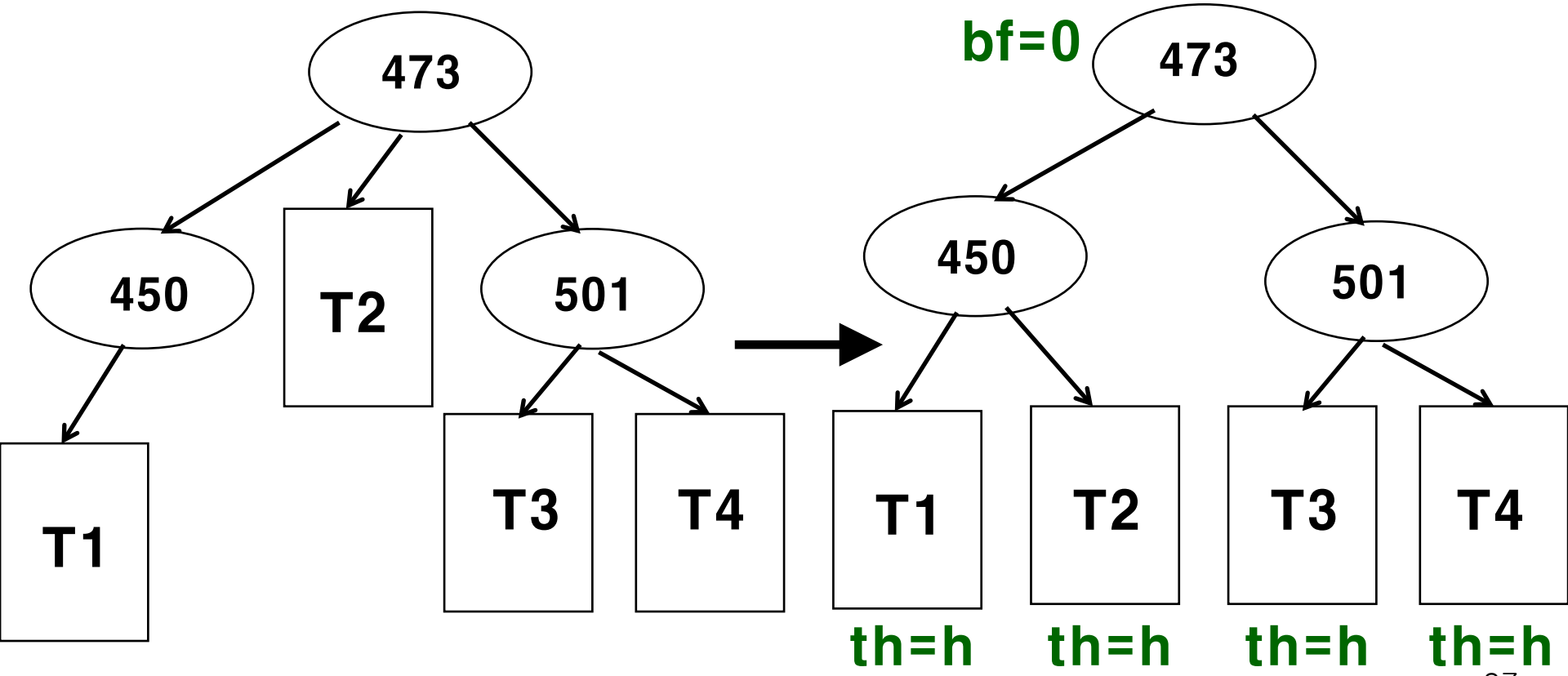
AVL Tree Rotations: General (3/4) cont'd

RL: Double Rotation (Right and Left)



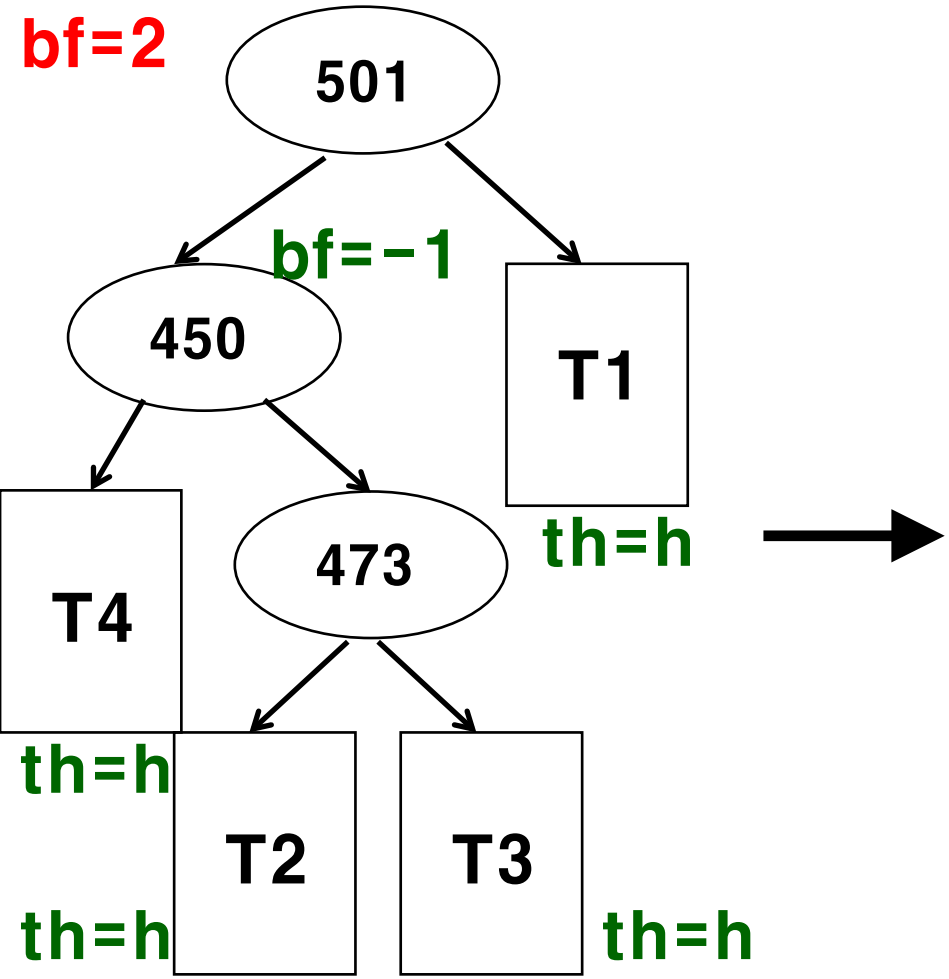
AVL Tree Rotations: General (3/4) cont'd

RL: Double Rotation (Right and Left)



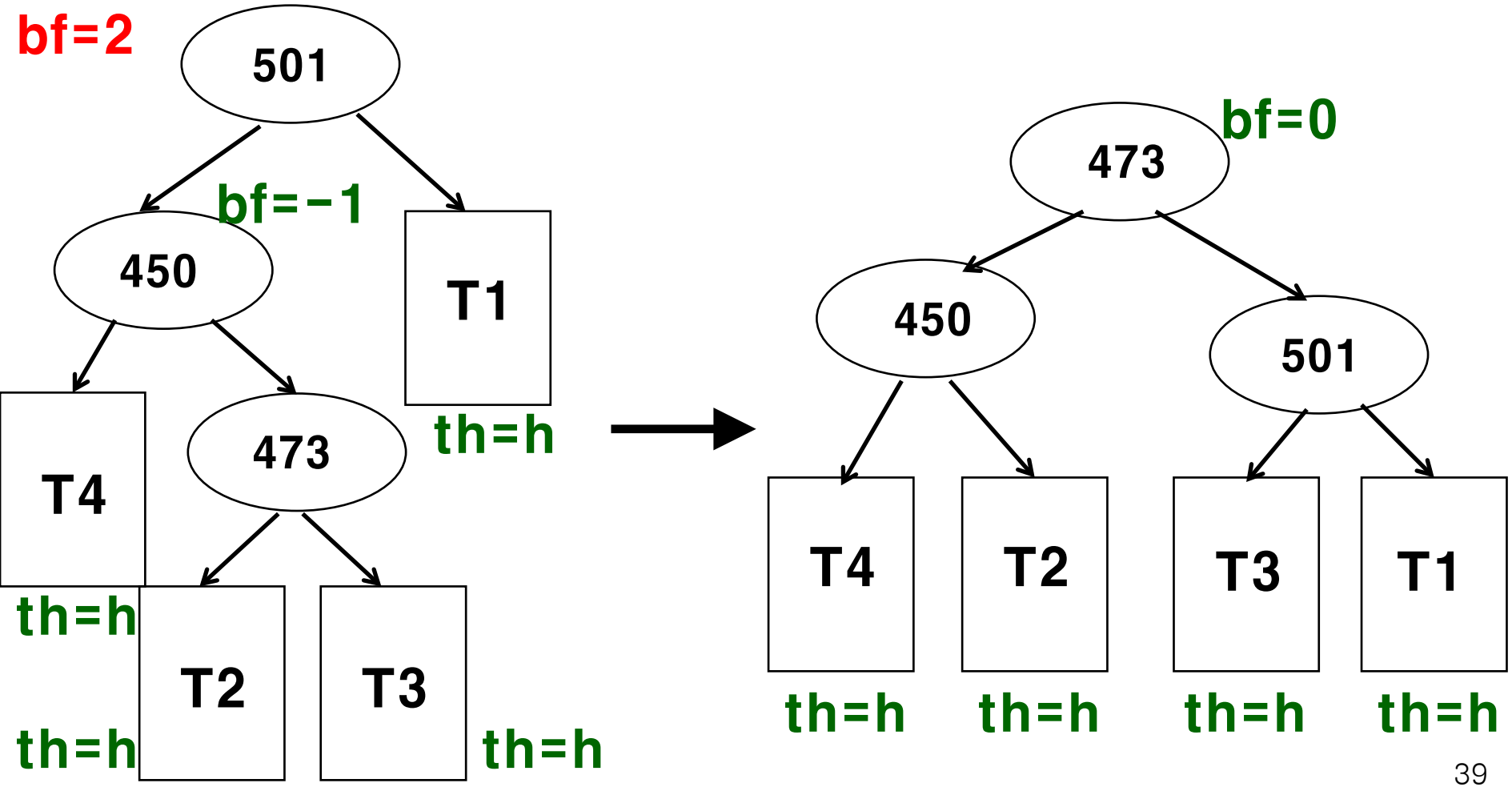
AVL Tree Rotations: General (4/4)

LR: Double Rotation (Left and Right)

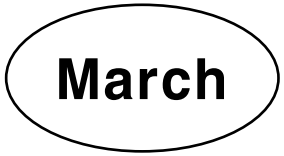


AVL Tree Rotations: General (4/4)

LR: Double Rotation (Left and Right)

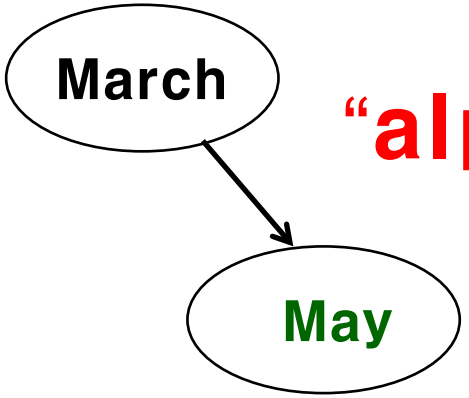


Example: Build an AVL Tree (1/12)



insert May

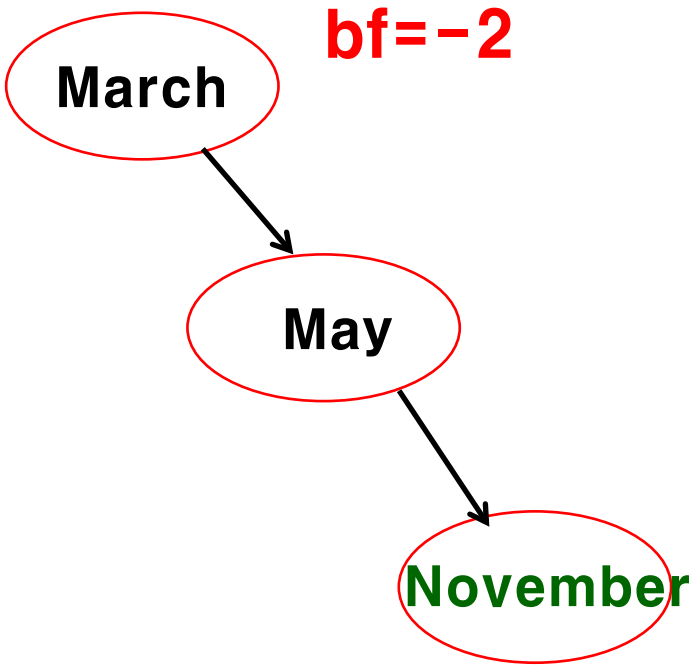
Example: Build an AVL Tree (2/12)



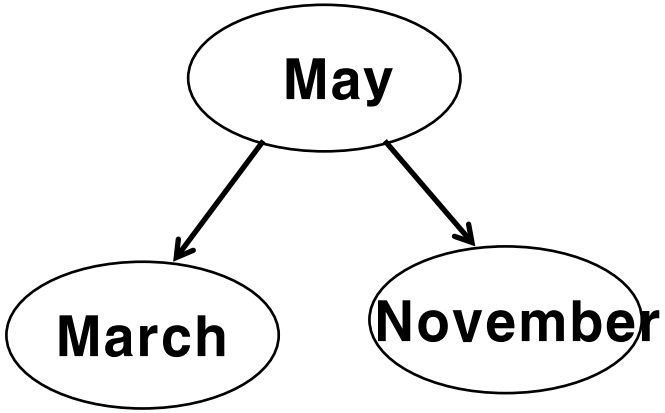
"alphabetical order"

insert November

Exercise: Build an AVL Tree (3/12)



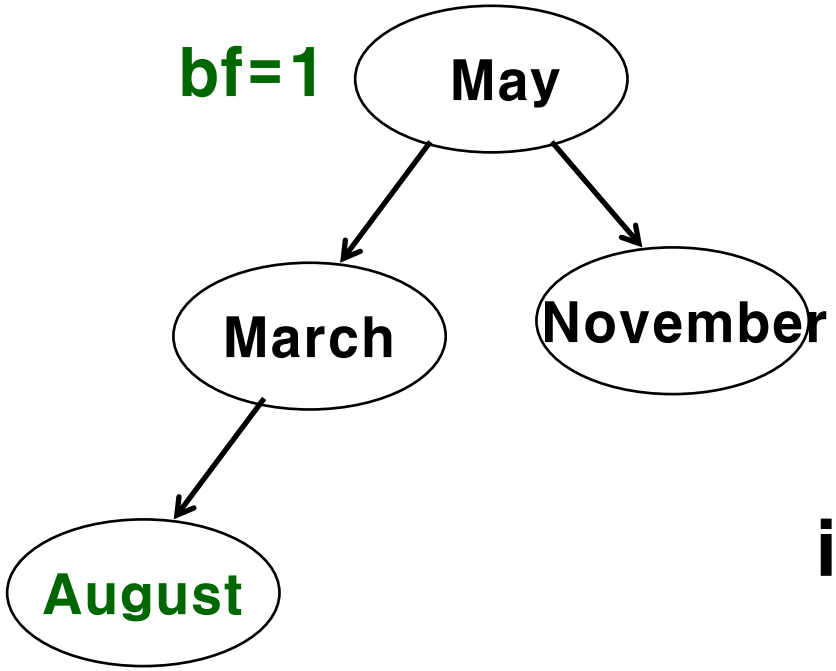
Exercise: Build an AVL Tree (4/12)



RR rotation

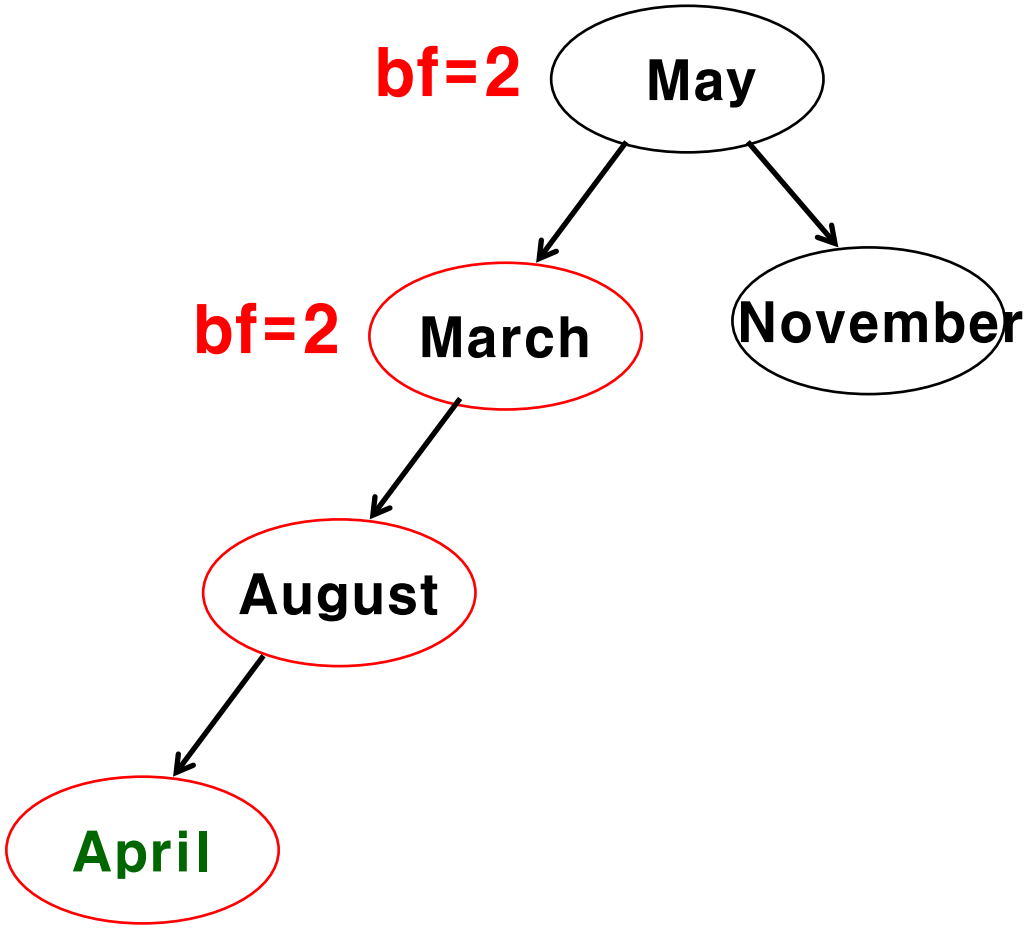
insert August

Example: Build an AVL Tree (5/12)

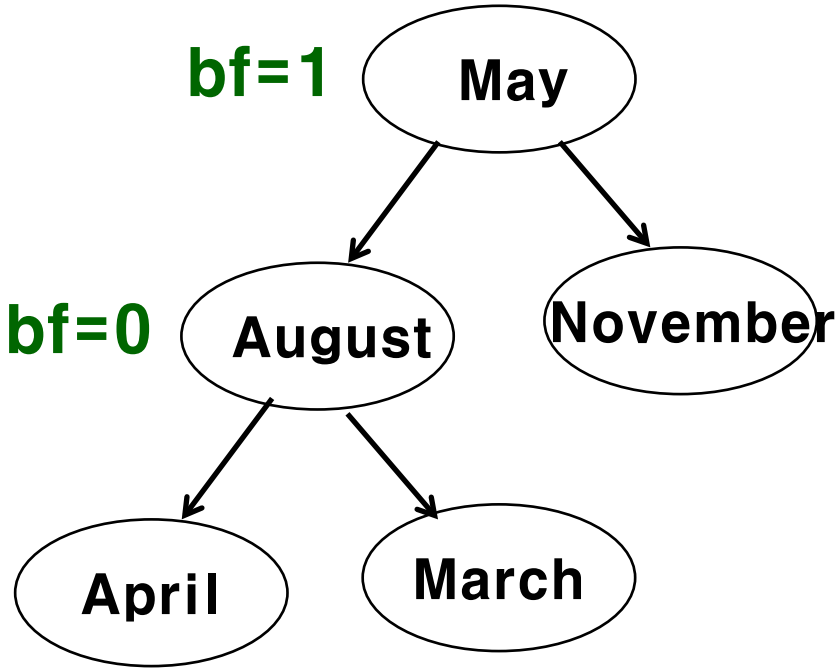


insert April

Exercise: Build an AVL Tree (6/12)



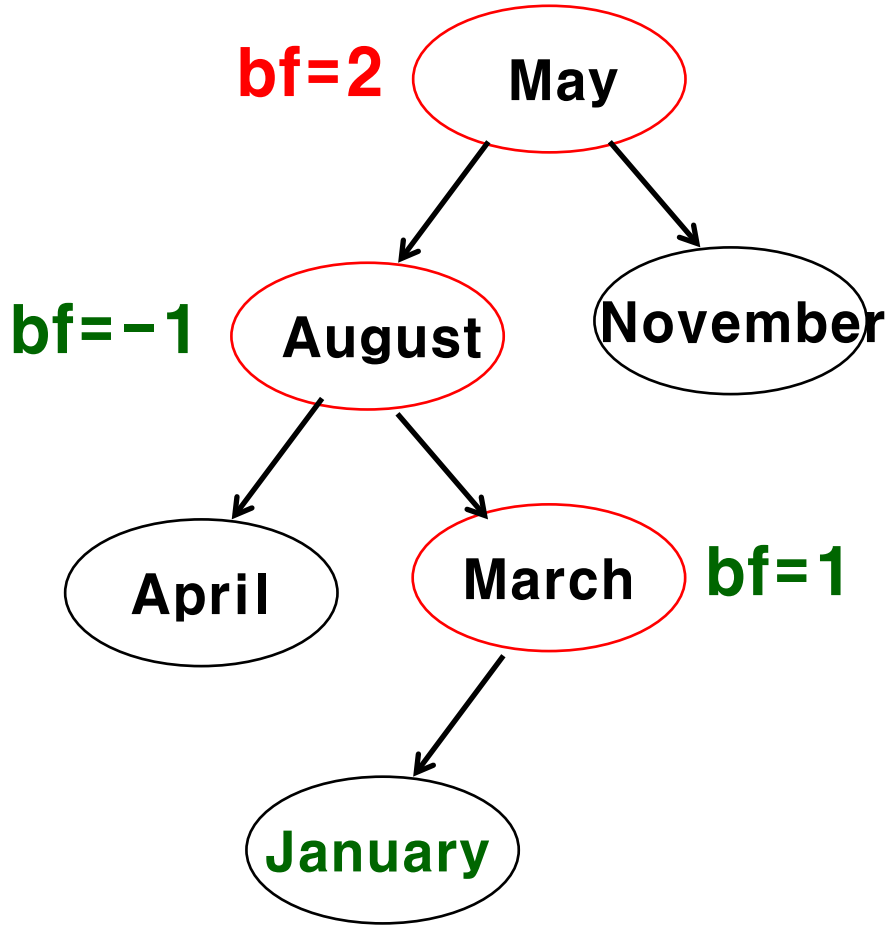
Example: Build an AVL Tree (7/12)



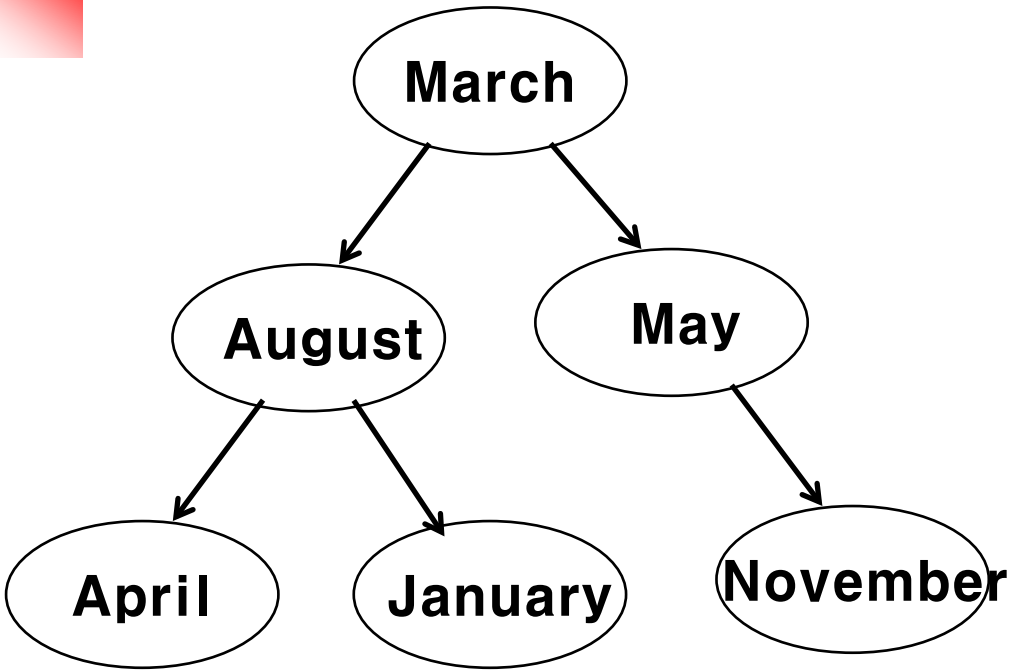
insert January

LL rotation

Exercise: Build an AVL Tree (8/12)



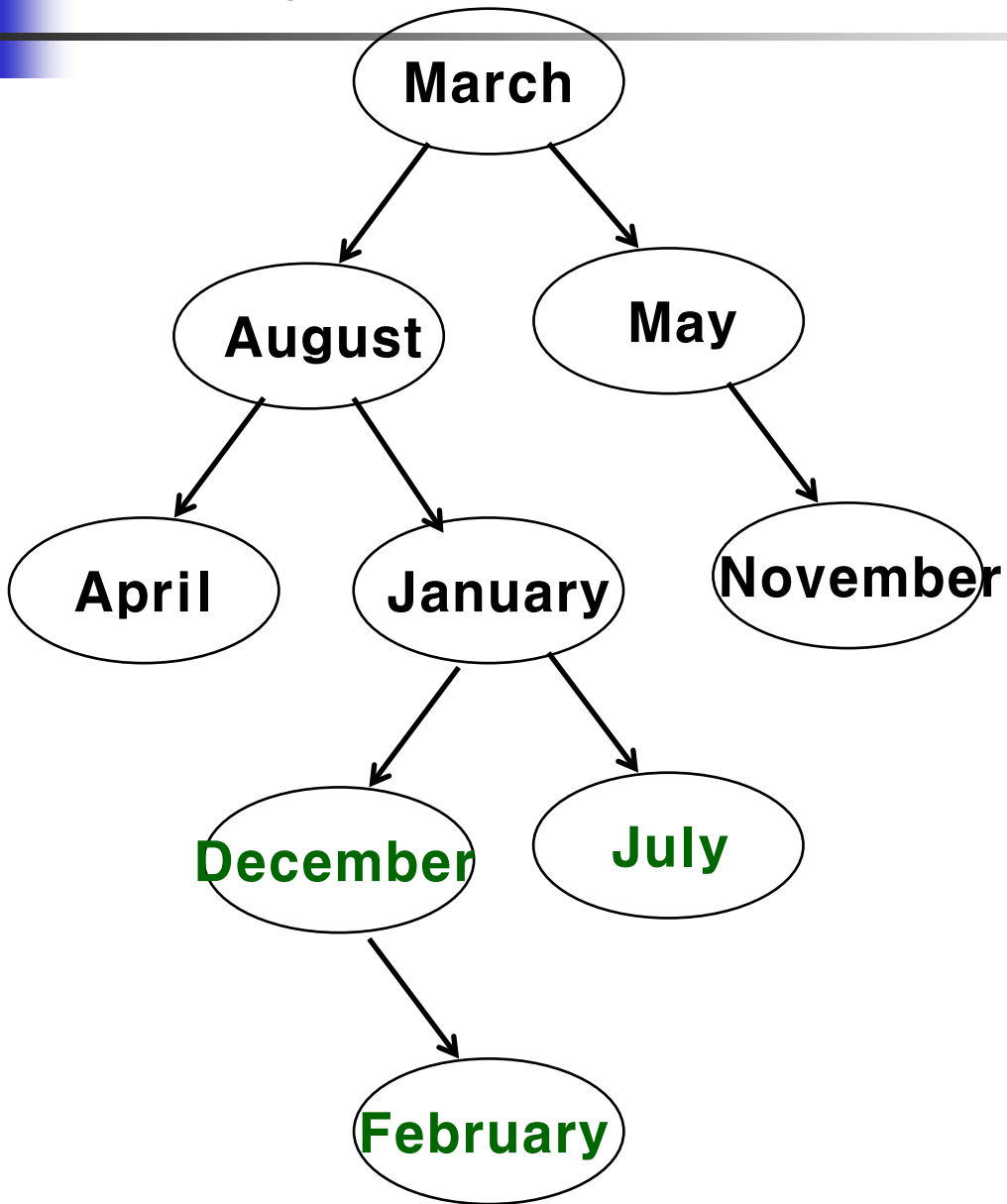
Example: Build an AVL Tree (9/12)



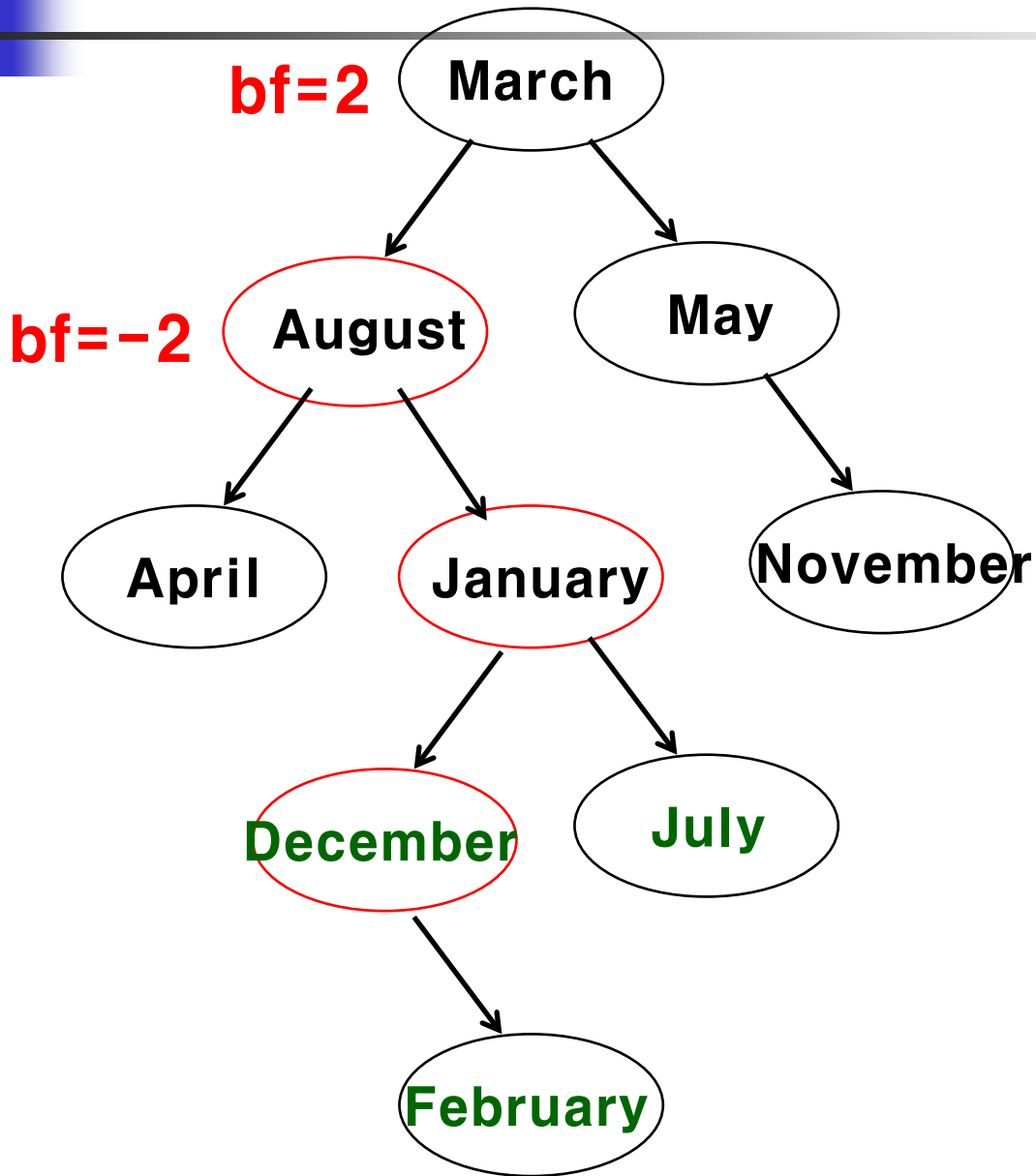
insert December
insert July
insert February

LR rotation

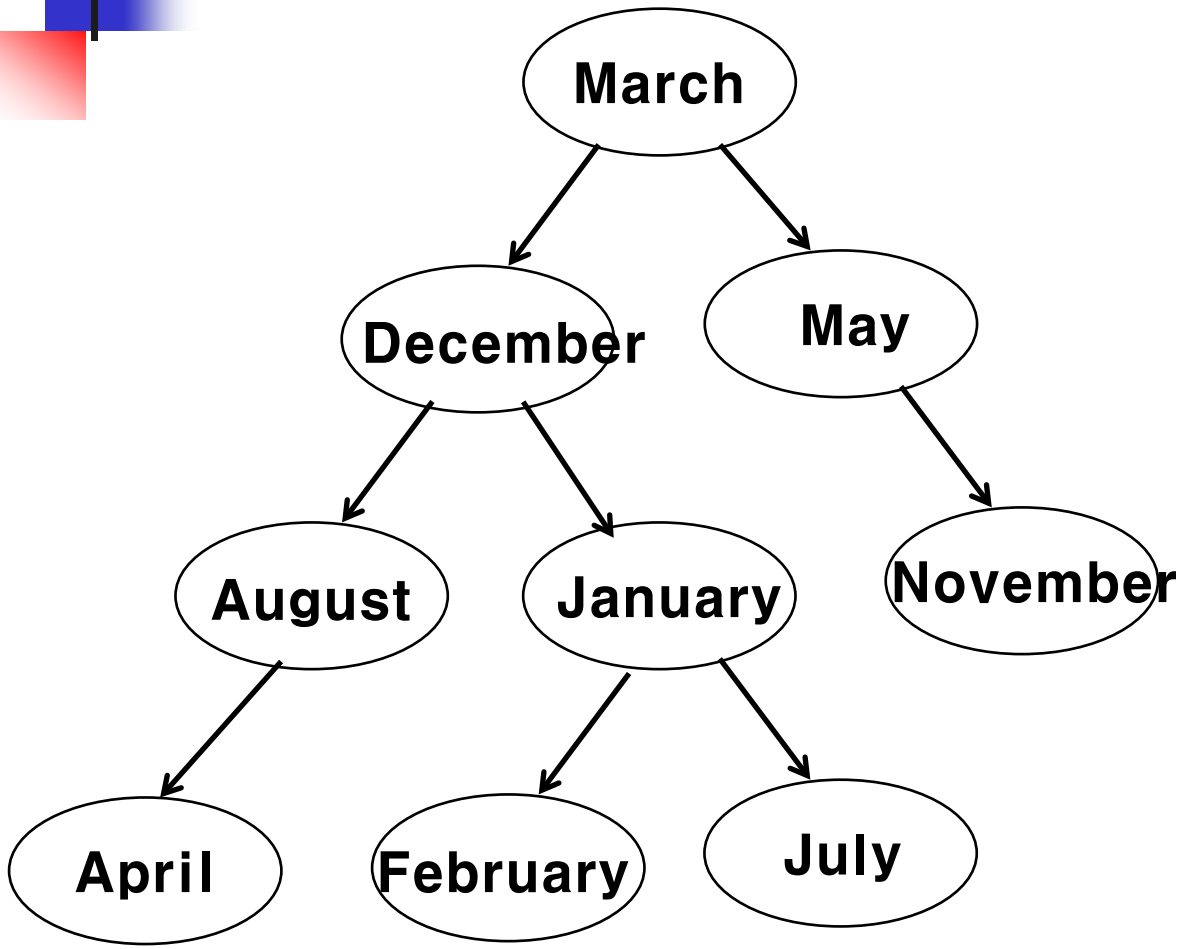
Example: Build an AVL Tree (10/12)



Exercise: Build an AVL Tree (11/12)



Exercise: Build an AVL Tree (12/12)



RL rotation

Assignment 6

Practice: AVL Programs

- https://www.thecrazyprogrammer.com/2014/03/c-program-for-avl-tree-implementation.html
- Hand trace the Above program in doing the following sequence of 5 inserts:
  - Insert: fish, dog, cat, mouse, lion
  - (for the purpose of easier conceptualization, assume the node key is a string, and the keys are arranged in alphabetical order.)
  - "hand tracing a program" means following the program line by line (as if you are the computer), plugging in the values for all the variables, and the function calls and returns.

Note on Hand Tracing of Programs

- Hand tracing programs is extremely important.
- Code inspection, code review, and debugging are required for software engineers.
- Code inspection
  - Hand tracing your own program or others' programs.
- Code review
  - Hand tracing others' programs
- Debugging
  - Hand tracing to find the source of errors discovered.

HW 6-1: Build an AVL Tree for Insertions

- Draw Trees and Rotations in doing the following sequence of inserts:
  - (must draw the results for each operation)

- Insert
  - fish, dog, cat, mouse, lion

HW 6-2: AVL Tree Insert and Delete

- Draw Trees in doing the following sequence of inserts and deletes:
  - (must draw the results for each operation)
  - (use the AVL program)

- Insert
  - Cat Dog Bat Fish Chicken Cow Tiger Eagle Lion Snake Bird Owl Mouse

- Delete
  - Cow Snake Owl Cat Mouse Eagle Bird

End of Lecture