AVL Tree

Balanced binary tree

- The disadvantage of a binary search tree is that its height can be as large as N-1
- This means that the time needed to perform insertion and deletion and many other operations can be O(N) in the worst case
- We want a tree with small height
- A binary tree with N node has height at least $\Theta(\log N)$
- Thus, our goal is to keep the height of a binary search tree O(log N)
- Such trees are called balanced binary search trees. Examples are AVL tree, red-black tree.

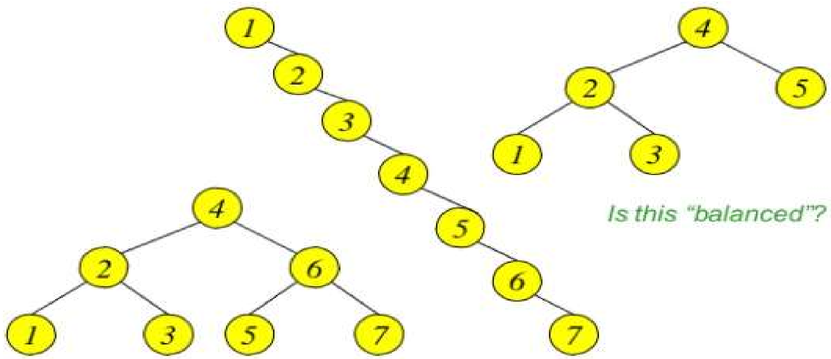
Binary Search Tree - Best Time

- All BST operations are O(h), where d is tree depth
- minimum d is $h = \lfloor \log_2 N \rfloor$ for a binary tree with N nodes
  - What is the best case tree?
  - What is the worst case tree?
- So, best case running time of BST operations is O(log N)

Binary Search Tree - Worst Time

- Worst case running time is O(N)
  - What happens when you Insert elements in ascending order?
    - Insert: 2, 4, 6, 8, 10, 12 into an empty BST
  - Problem: Lack of "balance":
    - compare depths of left and right subtree
  - Unbalanced degenerate tree

Balanced and unbalanced BST



Balancing Binary Search Trees

- Many algorithms exist for keeping binary search trees balanced
  - Adelson-Velskii and Landis (AVL) trees (height-balanced trees)
  - Splay trees and other self-adjusting trees
  - B-trees and other multiway search trees

AVL Tree is…

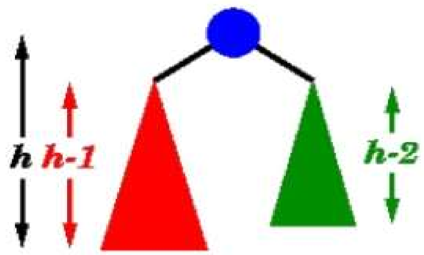
- Named after **A**delson-**V**elskii and **L**andis
- the first dynamically balanced trees to be propose
- Binary search tree with **balance condition** in which the sub-trees of each node can differ by at most 1 in their height

Definition of a balanced tree

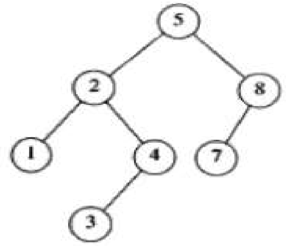
- Ensure the depth = O(log N)
- Take O(log N) time for searching, insertion, and deletion
- Every node must have left & right sub-trees of the same height

properties:

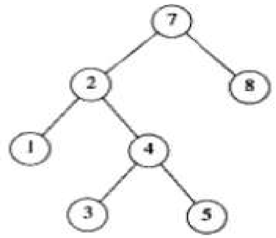
1. Sub-trees of each node can differ by at most 1 in their height
2. Every sub-trees is an AVL tree



AVL tree?



YES
Each left sub-tree has height 1 greater than each right sub-tree



NO
Left sub-tree has height 3, but right sub-tree has height 1

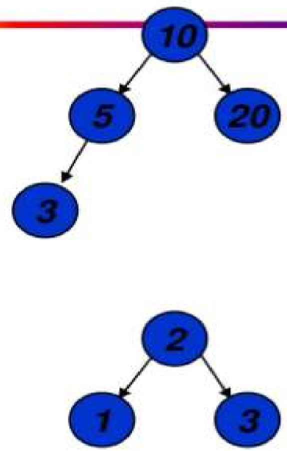
AVL tree

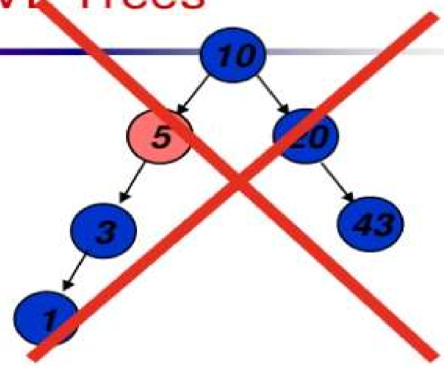
Height of a node

- The height of a leaf is 1. The height of a null pointer is zero.
- The height of an internal node is the maximum height of its children plus 1

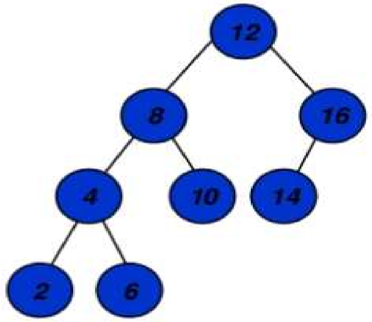
Note that this definition of height is different from the one we defined previously (we defined the height of a leaf as zero previously).

AVL Trees

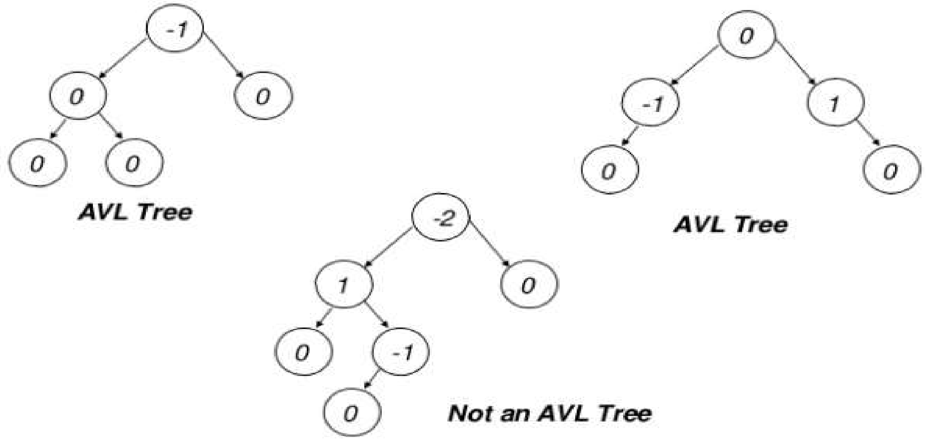




AVL Trees



AVL Tree



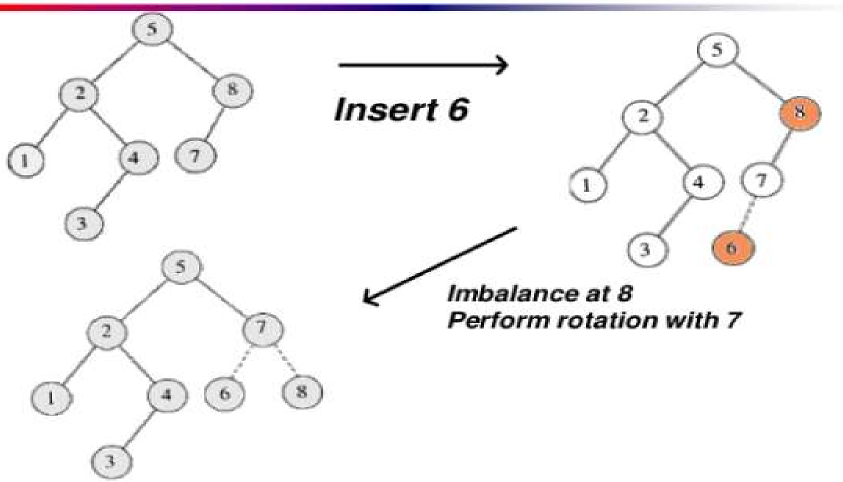
AVL - Good but not Perfect Balance

- AVL trees are height-balanced binary search trees
- Balance factor of a node
  - height(left subtree) - height(right subtree)
- An AVL tree has balance factor calculated at every node
  - For every node, heights of left and right subtree can differ by no more than 1
  - Store current heights in each node

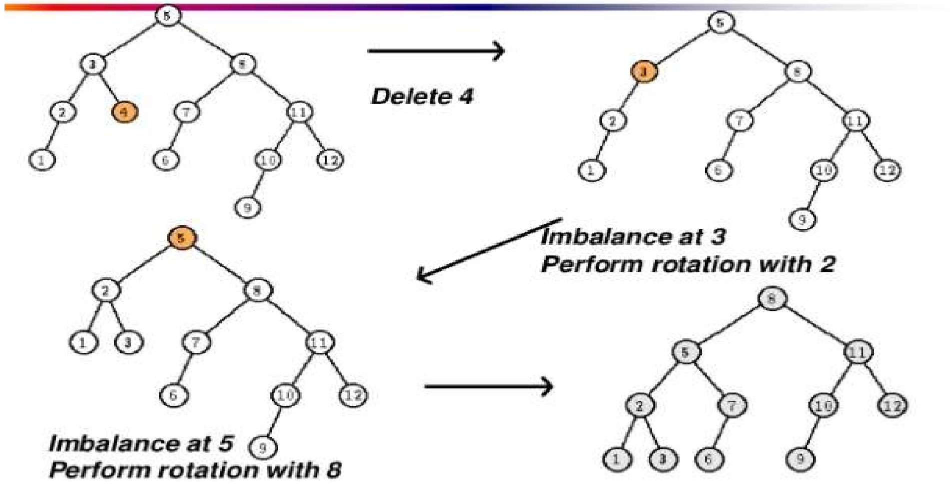
AVL Tree Operations

- Left Rotate
- Right Rotate
- Left Right Rotate
- Right Left Rotate

Insertion



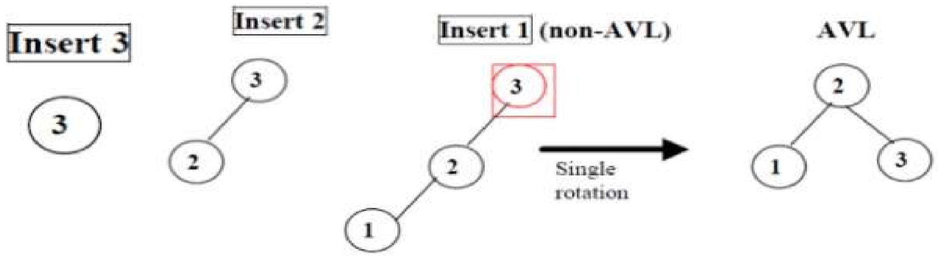
Deletion



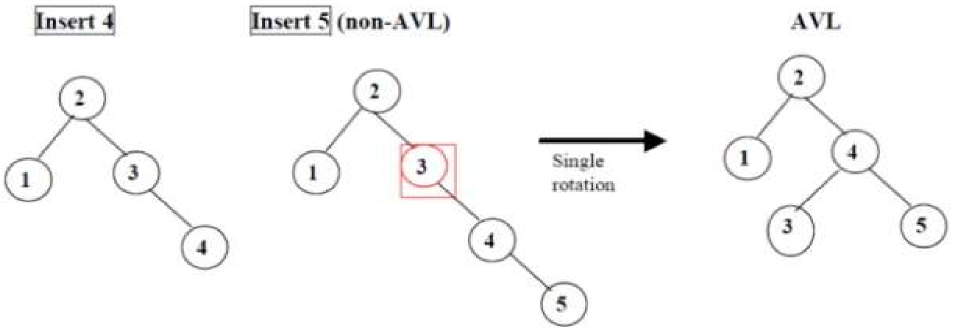
Key Points

- AVL tree remain balanced by applying rotations, therefore it guarantees O(log N) search time in a dynamic environment
- Tree can be re-balanced in at most O(log N) time

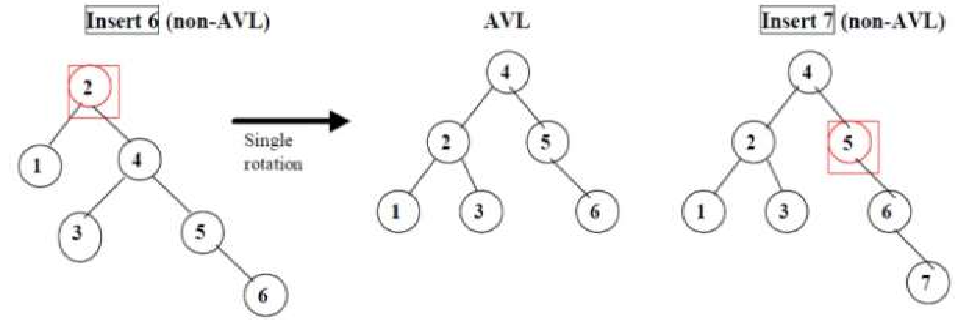
AVL Trees Example



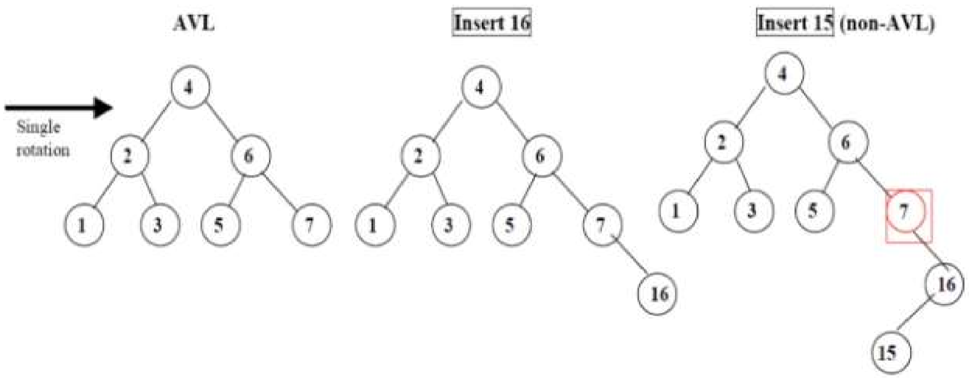
AVL Trees Example



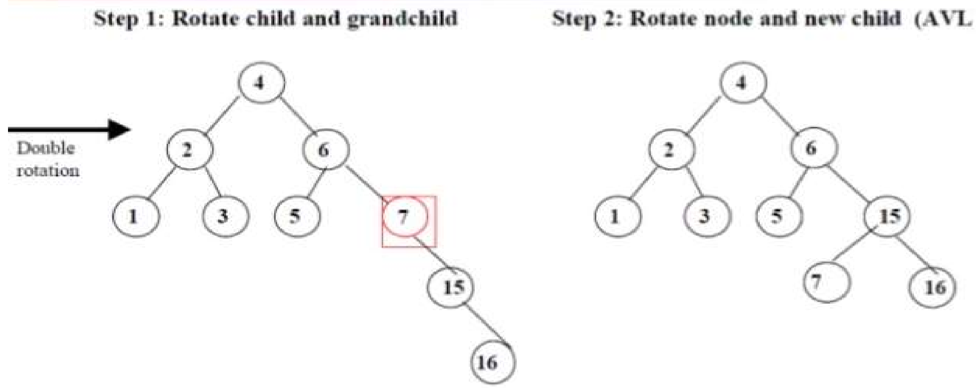
AVL Trees Example



AVL Trees Example

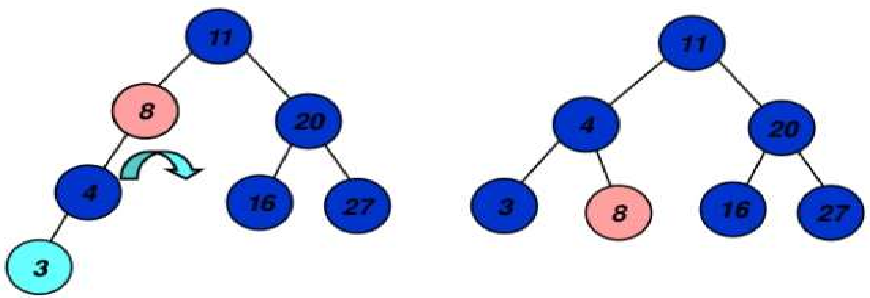


AVL Trees Example



Example

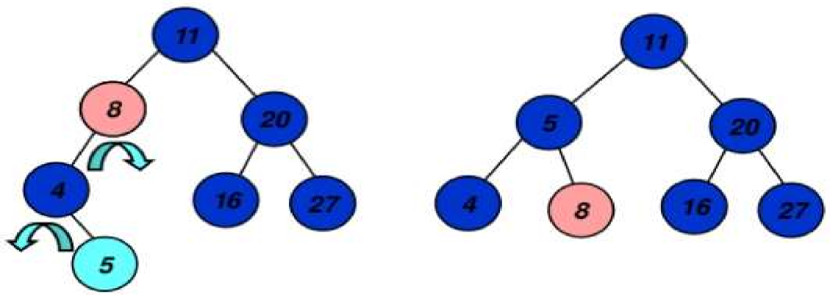
- Insert 3 into the AVL tree



5/22/2012

Example

- Insert 5 into the AVL tree



5/22/2012

https://www.slideshare.net/sandpoonia/lecture11-33728696