Balancing Trees

Weiss, sections 19.4 through 19.6
(pages 706–737)
Balanced Binary Search Trees

- The tree operations take $O(\log N)$ time if trees are balanced.
- Usually, a tree is balanced if all leaves occur in levels $h$ or $h-1$, and all nodes at levels lower than $h-1$ have two children.
- Inserting and removing nodes may imbalance trees.
- Height-balanced trees maintain the balance of trees.

Level 0

Level 1

Level 2

Level 3

imbalance

balanced
AVL Trees

• For every node, the height of left and right sub-trees differ by at most 1.
  – Height of an empty tree is -1.

• Maintain balance with rotations

Not an AVL tree

An AVL tree
AVL Insert

- **Case 1 (left-left)**
- An insertion in the left sub-tree of the left child of X
- Re-balance by a *single* rotation.
  - Rotate X with its left child.
  - Switches the role of parent (X) and child (left child of X)
AVL Insert

• Case 2 (right-right)
• An insertion in the right sub-tree of the right child of X
• Re-balance by a single rotation.
  ▪ Rotate X with its right child.
  ➢ Switches the role of parent and child.

Before rotation

After rotation
AVL Insert (cont.)

- Case 3 (right-left)
- An insertion in the right sub-tree of the left child of X
- Re-balance by two single rotations (double rotation)
  - Rotate between X’s child and grandchild.
  - Rotate between X and its new left child.
AVL Insert (cont.)

• Case 4 (left-right)
• An insertion in the left sub-tree of the right child of X
• Re-balance by two single rotations:
  - Rotate between X’s child and grandchild.
  - Rotate between X and its new child.
AVL Insert (cont.)
AVL Insert (cont.)
Red-Black Tree

- **Color property**
  - Every node is red or black.
  - Root is always black.
  - Children of a red node are black.
  - No two consecutive nodes are red.

- **Height property**
  - Every path from a node to each null link (null node) contains the same number of black nodes.

- Insertion may violate the properties.

Bottom-Up Insertion

- A new node must be red because if it’s black, it violates the height property.
- If the parent (P) of the new node is red, it violates the color property.
- Two cases depending on whether the sibling (S) of parent is black or red.
  - Case 1: Sibling of parent is black.
  - Relative to the grandparent of X (G), X is an inside or outside node.
    - If X is the outside node, perform single rotation between P and G. Then, swap the colors of P and G.
    - If X is the inside node, perform a double rotation. Swap the colors of X and G.
Bottom-Up Insertion (case 1)

X is an outside node.
Bottom-Up Insertion (case 1)

X is an inside node.
Bottom-Up Insertion (case 2)

- Change the color of P and S to black. Change the color of G to red.
- If G is the root, change its color to black. Otherwise, the color property is violated if the parent of G is red as well.

X is an inside node.

Introduction to algorithms, CLRS, third edition, Chapter 13
Bottom-Up Insertion (case 2)

- Change the color of $X$ and $S$ to black. Change the color of $G$ to red.
- If $G$ is the root, change its color to black. Otherwise, the color property is violated if the parent of $G$ is red as well.

$X$ is an outside node.
Bottom-Up Insertion

Case 2

Case 1
Bottom-Up Insertion
Top-Down Insertion

- Avoids rotating up the tree.
- While descending a tree,
  - If a node has two red children, change the color of the node to red and the colors of its children to black.
- If the parent of node is red, perform a single or double rotation.
- This technique ensures that sibling of parent is not red.
Top-Down Insertion

Color flip  Case 1  Color change
Case 1

https://mypages.valdosta.edu/dgibson/courses/cs3410/notes/ch19_5.pdf