Trees

A binary tree is either empty or it consists of a node, called the root, and two binary trees, called the left subtree and the right subtree.

Each node of a full binary tree is either internal—with two non-empty sub-trees—or a leaf—with two empty sub-trees.

A complete binary tree has all levels full except possibly the last, which is filled from left to right.

Tree Traversal

Often we want to do something for each node in a tree.

Denote:

V – perform the action on the root node
L – perform the action on the left sub-tree
R – perform the action on the right sub-tree

Six possible orders: VLR, LVR, LRV, VRL, RVL, RLV

Most common orders:

preorder – VLR
inoder – LVR
postorder – LRV
Binary Search Trees

Every node has a key and

1) The key of the root is greater than the key of any node in the left sub-tree.

2) The key of the root is less than the key of any node in the right sub-tree.

3) The left and right sub-trees are binary search trees.

AVL Trees

Goal: Keep binary search tree almost balanced — searching will be faster.

An AVL Tree is a binary search tree in which the heights of the left and right subtrees of the root differ by at most 1 and in which the left and right subtrees are also AVL trees.

Mark each node with a balance factor:

left-higher (Symbol: /) — left subtree is higher.

right-higher (Symbol: \)

equal-height (Symbol: —)
**AVL Insertion**

- A
- B
- C
- D
- E
- F

**Rebalance Right**

**Pre-condition:** \( \text{height}(r->right) = \text{height}(r->left)+2 \)

**Case a)** Right is Right heavy: rotate left

- A
- B
- C
- D
- E
- F
- G

- A
- B
- C
- D
- E
- F
- G
- H
- I
- J
- K
Case b) Right is Left heavy: double rotate left

sub-case i) Left child of Right is balanced:

sub-case ii) Left child of Right is Left heavy:

sub-case iii) Left child of Right is Right heavy:

Case c) Right is balanced: rotate left

Height of an AVL Tree

To determine the worst case number of comparisons needed to find an element in an AVL tree, we need to know the maximum possible height of the tree.

What is the minimum number of nodes in an AVL tree of height $h$?

- $F_h$ denote an AVL tree of height $h$ containing the minimum possible number of nodes, and
- $|F_h|$ be the number of nodes in that tree

$|F_1| = 1, |F_2| = 2, |F_3| = 4$
\[ |F_h| = 1 + |F_{h-1}| + |F_{h-2}| \]

This is a Fibonacci sequence, so
\[ |F_h| + 1 \approx \frac{1}{\sqrt{5}} \left( \frac{1+\sqrt{5}}{2} \right)^{h+2} \]

\[ h \approx 1.44 \log |F_h| \] — in the worst case an AVL tree is 1.44 times as tall as a perfectly balanced tree.