CSC263 Winter 2019
Lecture Notes (Archived)

Yuchen Wang

March 15, 2025

Contents

1 Notes On AVL Trees

Definition of AVL Trees A binary tree is *height-balanced* if the heights of the left and right subtrees of *every* node differ by at most one. An AVL tree is a height-balanced binary search tree.

Remark By convention, the height of an empty tree is -1; the height of a tree consisting of a single node is 0.

Definition of balance factor Let h_R and h_L be the heights of the right and left subtrees of a node m in a binary tree respectively. The *balance factor* of m, $BF[m]$, is defined as $BF[m] = h_R - h_L$.
For an AVL tree, the balanced factor of any node is $-1, 0$ or $+1$.

1. if $BF[m] = +1, m$ is right heavy
2. if $BF[m] = -1, m$ is left heavy
3. if $BF[m] = 0, m$ is balanced

In AVL trees we will store $BF[m]$ in each node m

Algorithm Search Treat T as an ordinary binary search tree

Algorithm Insert First insert x in T as in ordinary binary search trees: trace a path from the root downward, and insert a new node with key x in it in the proper place, so as to preserve the binary search tree property. This may destroy the integrity of our AVL tree in that

1. The addition of a new leaf may have destroyed the height-balance of some nodes
2. The balance factors of some nodes must be updated to take into account the new leaf

Steps for Insert as following
Insert **x** into T as in any BST:
  x is now a leaf
  Set BF(x) to 0
Go up from x to the root and for each node v in this path

Adjust the BF:
  if x is in right subtree of v: Increment BF(v)
  if x is in left subtree of v: Decrement BF(v)
Rebalance if necessary:
if BF(v) = +2:
  if BF(v.right) = +1
    Do Left Rotation, update BFs of rotated nodes, and stop
  if BF(v.right) = -1
    Do Right-Left Rotation, update BFs of rotated nodes, and stop
  if BF(v.right) = -2
    Symmetric to above case