AVL Trees

A balanced binary search tree is a binary search tree where the heights of the left and right subtrees of any node differ by at most one.

For any node \( n \), the height \( h(n) \) of the subtrees is given by:

- \( h(n) = \max\{h(left(n)), h(right(n))\} + 1 \)
- \( h(left(n)) = 0 \) if \( n \) is a leaf node

An AVL tree is a balanced binary search tree where the heights of the left and right subtrees of any node differ by at most one.

For an AVL tree, the height of the root is equal to \( 2^{\log(n+1)} \).

Balanced Binary Search Trees

An algorithm for the organization of information, 1962

C.M. Aideson-Veldy and E.M. Landis

AVL Trees

As AVL Trees

* Lecture 4

* Binary Search Trees

6.006 Recitation 4
13 Feb 2009
\[ n \geq \log_2 N_n \]
\[ N_n \geq 2^{\log_2 N} \]
\[ \geq 1 + 2 + 2^2 + \ldots + 2^{h/2} = 2^{h/2} - 1 \]
\[ \geq 1 + 2 + 2^2 (1 + 2N_{n-4}) \]
\[ \geq 1 + 2N_{n-2} + 1 \]
\[ \geq N_{n-1} + N_{n-2} + 1 \]

```
4
6
G
8
C
2
3
3
```

```
4
5

```

```
2
3
```

Exercise
All invariant violated!

Three possible cases

Rotations

No Problem
Case 2: goes to B

Case 1: goes to C

Insert node α

Consider a violation
Case 1:

```
def (left-rot) a:
    y = a.right
    a.right = y.left
    y.left = a
    return y
```
Symmetric Rotation for \( \text{Case 2} \):

1. Rotate 1
2. Rotate 2
null

• Deletion - need to check if all invariant from first point of discrepancy to the root
  • After rotation
    • height of root & child remains

  Similar as in BST, followed by rotations to correct imbalances

Insertion and Deletion
Deletion From AVL Trees Exercise

Finally, AVL II