Binary Search Trees
* Lecture 4
* AVL Trees

**AVL TREES**

G.M. Adelson-Velsky and E.M. Landis

“An algorithm for the organization of information”, 1962

Balanced Binary Search Tree

For any node \( n \), \( \text{height}(n) \): Length of longest path from \( n \) to a leaf node

\[
\text{height}(n) = \max(\text{height}(\text{left}(n)), \text{height}(\text{right}(n))) + 1
\]

\( \text{height}(\text{NULL}) = -1 \), \( \text{height}(y) = 0 \) \( \forall y \): leaf node

AVL invariant

\( \forall n: \text{node}, \quad |\text{height}(\text{left}(n)) - \text{height}(\text{right}(n))| \leq 1 \)

\( |h_1 - h_2| \leq 1 \)

\( \text{height}(\text{root}) < 2 \log n \)
**Exercise**

A. [Diagram of AVL tree with checks for balance]

B. [Diagram of AVL tree with checks for balance]

C. [Diagram of AVL tree with checks for balance]

\[ N_h \geq N_{h-1} + N_{h-2} + 1 \]

\[ \Rightarrow 2N_{h-2} + 1 \]

\[ \Rightarrow 1 + 2(1 + 2N_{h-4}) \]

\[ \Rightarrow 1 + 2 + 2^2(1 + 2N_{h-6}) \]

\[ \Rightarrow 1 + 2 + 2^2 + \ldots \]

\[ 2^{h/2} = 2^{h/2+1} - 1 \]

\[ N_h \geq 2^{h/2} \]

\[ h \leq 2 \log_2 N_h \]
Rotations

AVL tree before insertion/deletion

Three possible cases

NO PROBLEM

AVL invariant violated!
Consider a violation $V_2$

Case 1: goes to C

Case 2: goes to B
Case 1:

```
def left_rot(x):
    y = x.right
    x.right = y.left
    y.left = x
    return y
```
Case 2:

```
def dbl_left_rot(n):
    y = n.right
    z = y.left
    n.right = z.left
    z.left = n
    y.left = z.right
    z.right = y
    return z
```
**INSERTION AND DELETION**

Similar as in BST, followed by rotations to correct imbalances.

- **INSERTION** — One rotation is sufficient (why?)

- **DELETION** — Need to check AVL invariant from first point of discrepancy to the root.

\[O(\log n)\]
Deletion From AVL Trees Exercise

Finally AVL!!