

Set 11: Ordered Dictionary Abstract Data Types: AVL Trees

CS240: Data Structures and Data Management

Jérémy Barbay

Outline

AVL Trees

Definition

Height of AVL Trees

Balancing operations

Binary Search Trees

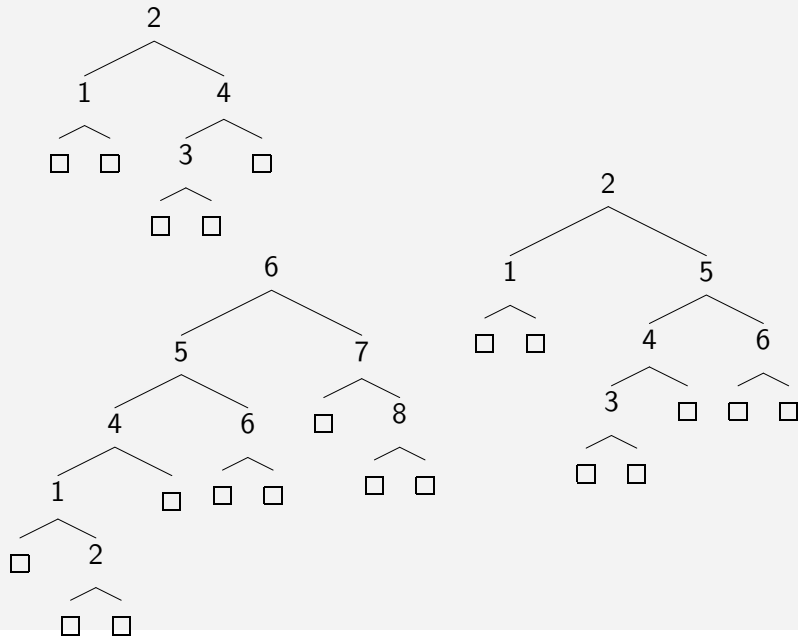
- ▶ The worst-case performance is
- ▶ Randomly built trees perform well
 - ▶ Expected height $h = 1.386 \log(n + 1)$
- ▶ Sequence of n^2 alternating inserts/deletes
 - ▶ Expected height $h \in \Theta(\sqrt{n})$
- ▶ Possible improvements?

Height Balanced Trees

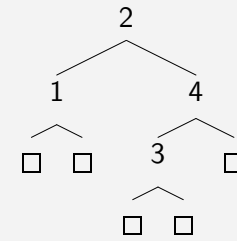
- ▶ Can we **guarantee** tree height?
 - ▶ Try to keep our search trees **balanced**
 - ▶ Must not affect the running time
- ▶ **Balanced Node**
The heights of its subtrees differ by at most one
- ▶ **AVL Tree**
A Binary Search Tree such that every node is balanced
 - ▶ *Adel'son-Vel'skii and Landis, 1962*

Which Trees are AVL?

Which nodes are balanced?



Recording Balance



- We can explicitly record a height for each node
It would take bits per node.
- Or we can use condition codes
 - = - Balanced
 - > - Left-heavy (by one)
 - < - Right-heavy (by one)
 It would take bits per node.

AVL Tree Height

General Idea

Let $S(h)$ be the fewest possible nodes for an AVL tree of height h (including placeholders)

$S(1)$

$S(2)$

$S(3)$

$S(h) =$

AVL Tree Height

General Idea

Theorem

h is $\Theta(\log n)$ for an AVL tree of height h and n internal nodes.

Proof:

- Recurrence relation (close to Fibonacci Sequence) gives

$$S(h) = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{h+2}}{\sqrt{5}} + 1$$

- Note: $S(h) \leq n$.

$$\begin{aligned} h &\leq \frac{\lg n}{\lg \frac{1+\sqrt{5}}{2}} + o(1) \\ &\approx 1.44 \lg n \end{aligned}$$

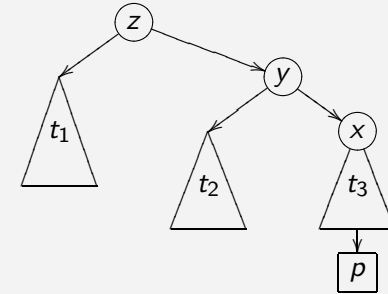
Operations

- ▶ Find:
 - ▶ As in a Binary Search Tree (BST).
- ▶ Insert
 - ▶ Find and insert as in a BST.
 - ▶ Update heights (codes) on path back to root
 - ▶ Locate a possible unbalanced node, z
 - ▶ Perform a **rotation** (see two next slides)
- ▶ Delete
 - ▶ Find and delete as in a BST
 - ▶ Update heights (codes) on path back to root
 - ▶ Locate possible unbalanced nodes and **rotate** them.

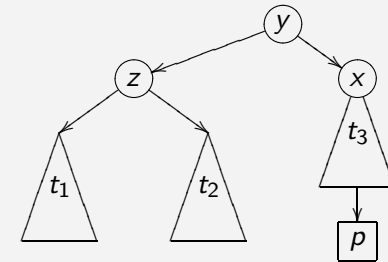
The complexity of Find is .

"Single" Rotation

- ▶ node z fails the AVL test after adding node p :

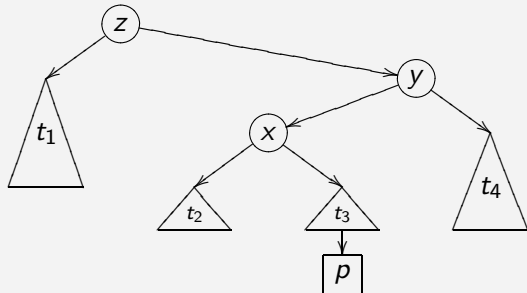


- ▶ single rotation regains balance:

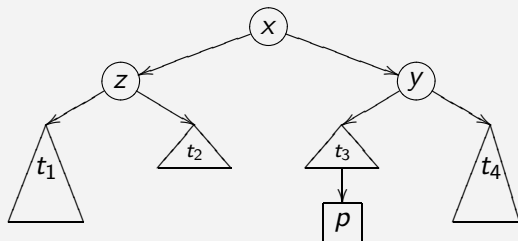


"Double" Rotation

- ▶ Node z fails the AVL test after adding node p :



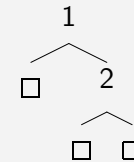
- ▶ Double rotation regains balance:



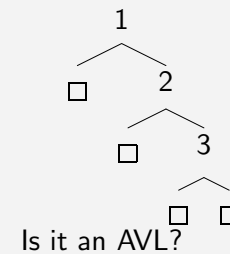
Examples of Insertion

"Single" Rotation

- ▶ From an empty tree: Insert(1), Insert(2)



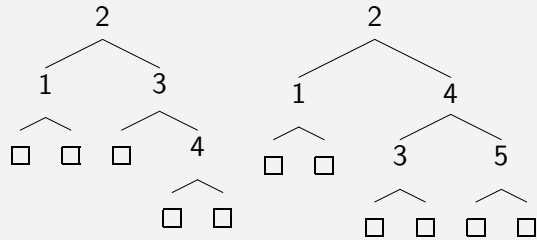
- ▶ Insert(3)



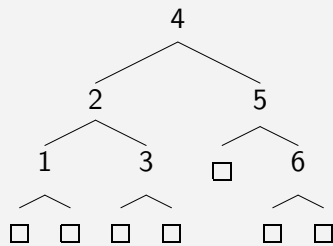
Examples of Insertion

"Single" Rotation (Cont)

- Insert(4), Insert(5)



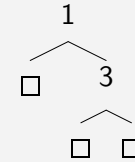
- Insert(6)



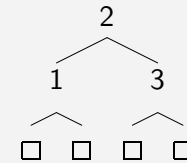
Examples of Insertion

"Double" Rotation

- From an empty tree: Insert(1), Insert(3)

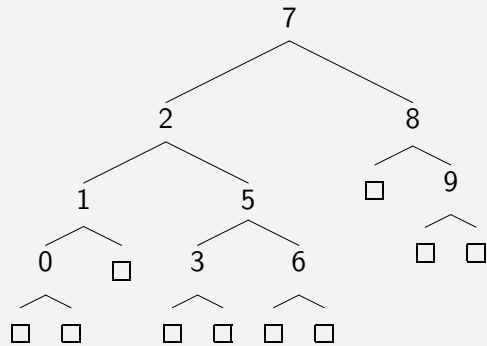


- Insert(2)



Examples of Insertion

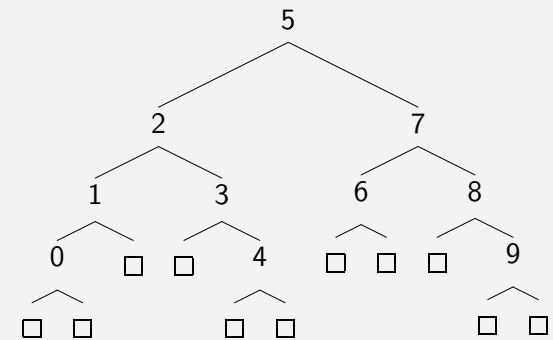
"Double" Rotation, Larger Example



- Insert(4)

Examples of Insertion

"Double" Rotation, Solution of the large example.

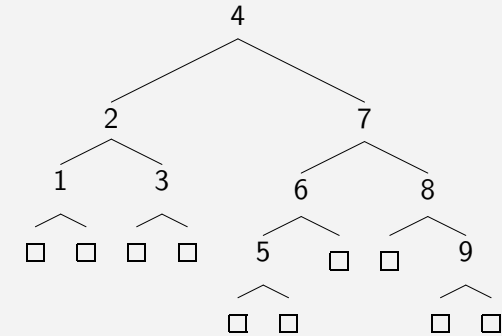


Cost of Insert

- ▶ How many rotations may be required for an Insert?
(A double rotation counts as one rotation.)
- ▶ How expensive is a rotation?
- ▶ Worst-case running time for Insert?

Examples of Deletion

"Simple Rotation"

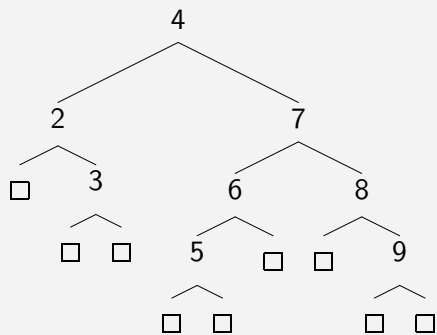


- ▶ Delete(1)
- ▶ Delete(2)

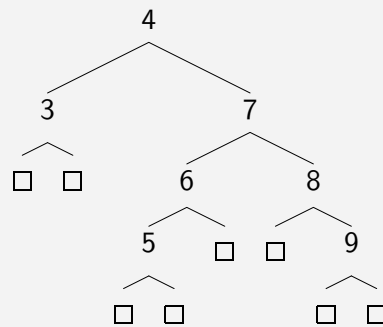
Examples of Deletion

"Simple Rotation", solutions

Delete(1)

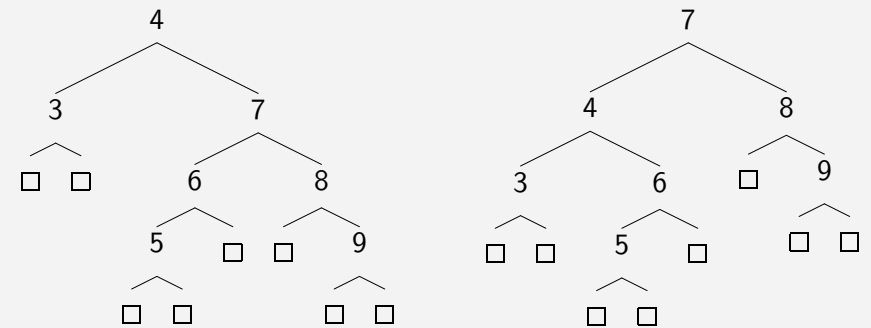


Delete(2)



Examples of Deletion

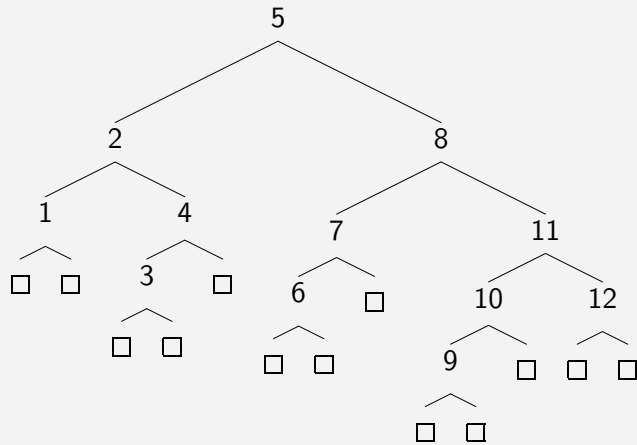
"Simple Rotation", solutions (cont)



The unbalanced node is
We perform a rotation.

Delete

Larger Example

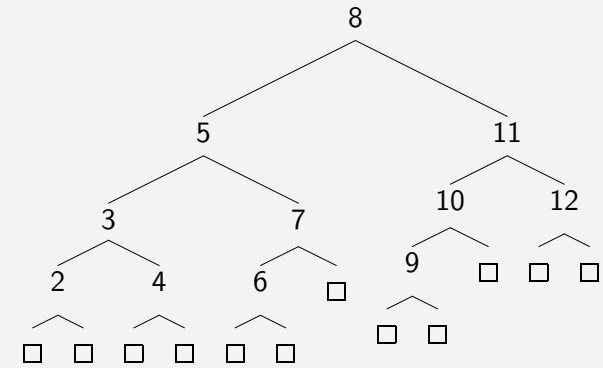


► Delete(1)

Delete

Solution of large Example

We need to rotate first around , and then around .



Cost of Delete

- How many rotations may be required for a Delete?
- How expensive is a rotation?
- Worst-case running time for Delete?

Final Thoughts

- All major binary search tree operations have **guaranteed worst-case** $\Theta(\log n)$ performance
- Fairly large constant hidden in order notation
- Each internal node stores a condition code (or height)
- Condition code is usually represented by $\{-1, 0, 1\}$

Summary

- ▶ AVL Trees are
- ▶ Their height is
- ▶ The time in which the operators are supported is
 - ▶ Search in
 - ▶ Insertion in
 - ▶ Deletion in

References:

- ▶ Goodrich and Tamassia: pp. 152-158
- ▶ Cormen, Leiserson, Rivest, Stein: pp. 296 (poorly covered)