AVL Trees

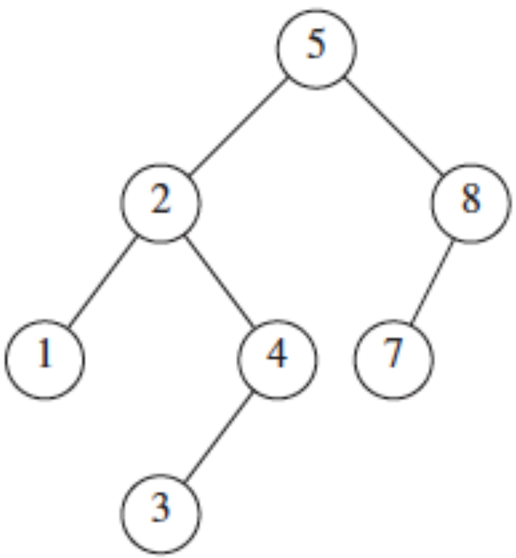
CSCI-2270

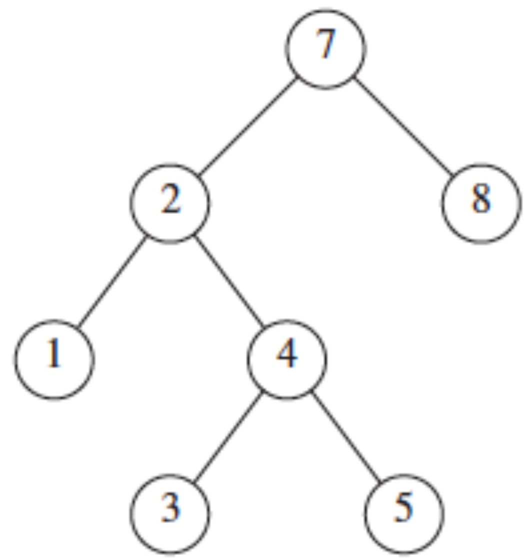
Elizabeth Boese

AVL

AVL (Adelson-Velskii and Landis)

- BST with balance
  - identical to a binary search tree, except that for every node in the tree, the height of the left and right subtrees can differ by at most 1.
- Which of the following is a binary tree? AVL tree?



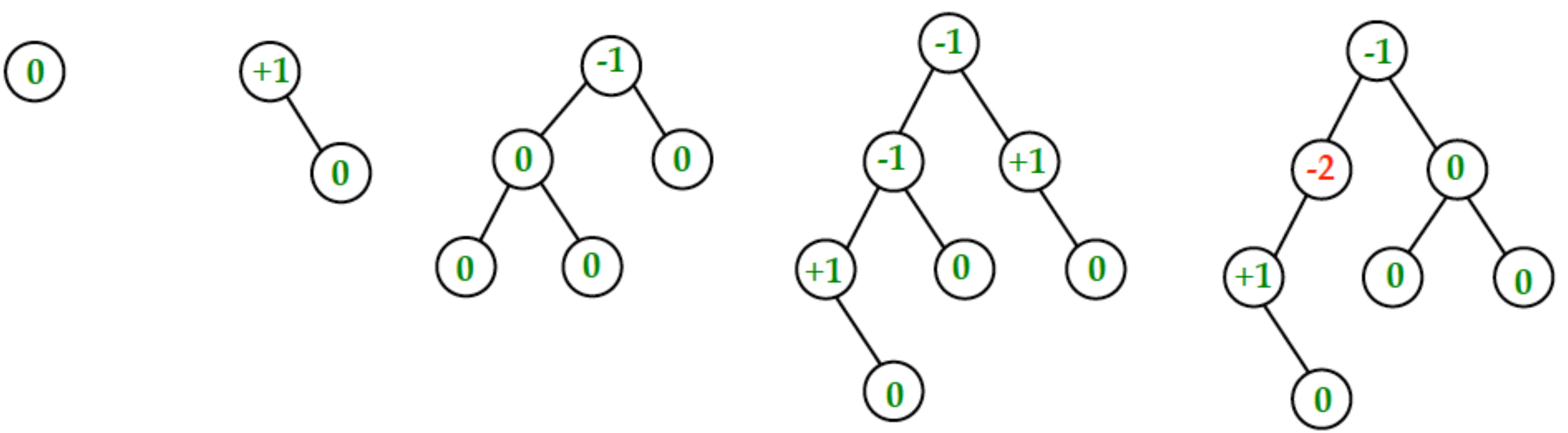


https://www.cs.umd.edu/class/spring2008/cmsc420/L7.AVLTrees.pdf

AVL

- A binary tree is an AVL tree if
  balance(u) ∈ {-1, 0, +1} for every node u

- I.e. the heights of LEFT(u) and RIGHT(u) are "about the same" for every node u.
- balance(u) := right_height(u) - left_height(u)



NOT an AVL tree

https://www.cs.umd.edu/class/spring2008/cmsc420/L7.AVLTrees.pdf

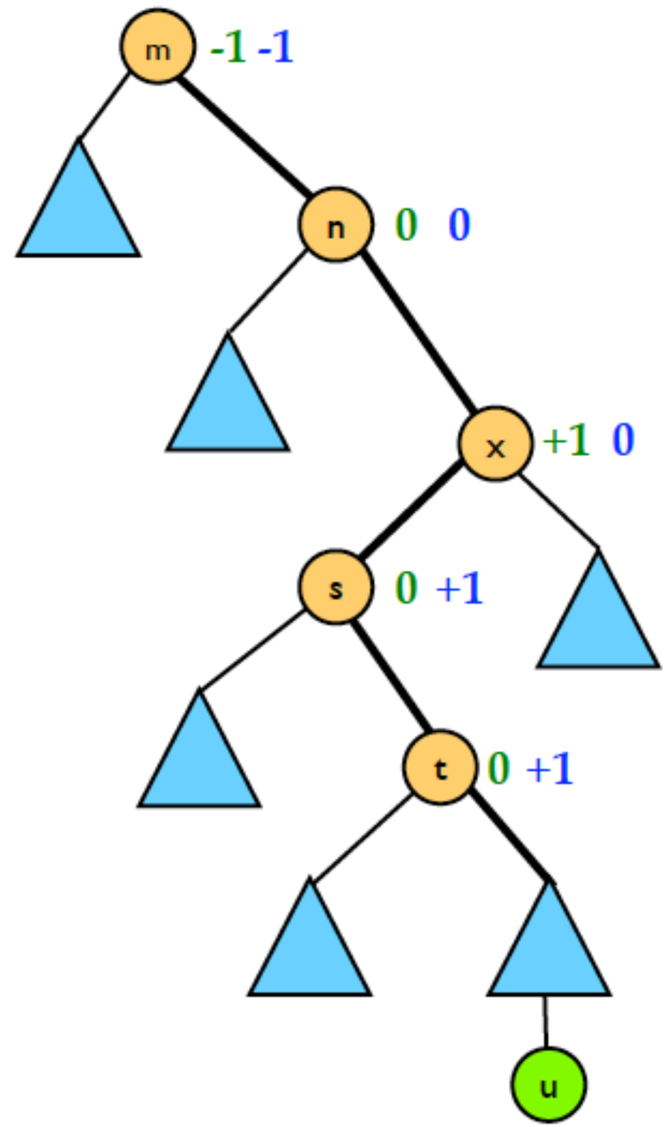
AVL Trees

- AVL tree with n nodes has height O(log n).
  - search will run in O(log n) time if AVL has binary search tree property.
- **insert**, **delete** can be implemented in **O(log n)** time.
- Good structure to implement dictionary or sorted set ADTs.

https://www.cs.umd.edu/class/spring2008/cmsc420/L7.AVLTrees.pdf

AVL Insert

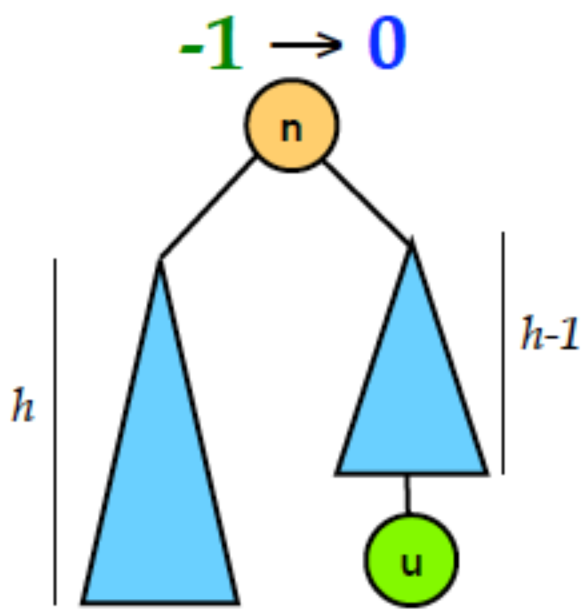
1. First, do a standard BST insert: do a find and add node where you "fall off the tree."
2. Walk insertion path back up to root, updating balances.
3. If node was added to the left subtree, *decrement* balance by 1, otherwise *increment* balance by 1. Stop when node's height doesn't change.
4. If a balance becomes +2 or -2, *fix it*.



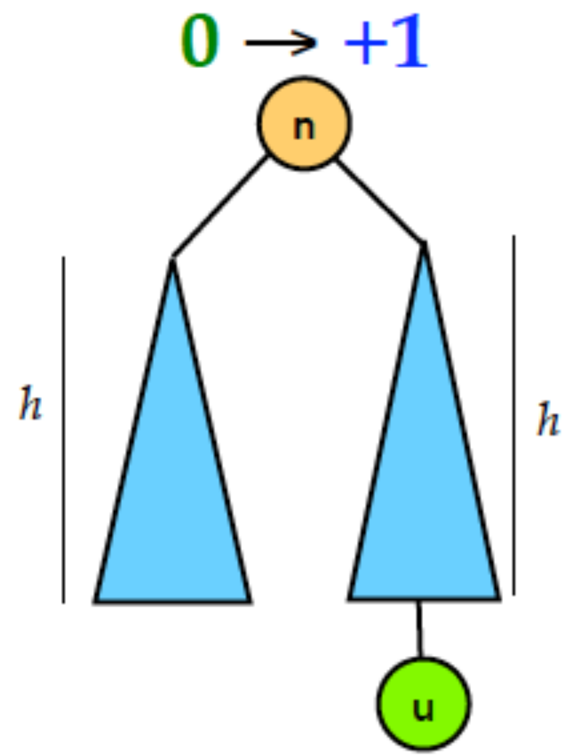
https://www.cs.umd.edu/class/spring2008/cmsc420/L7.AVLTrees.pdf

AVL Insert

Easy Cases



Node was added to the shorter subtree

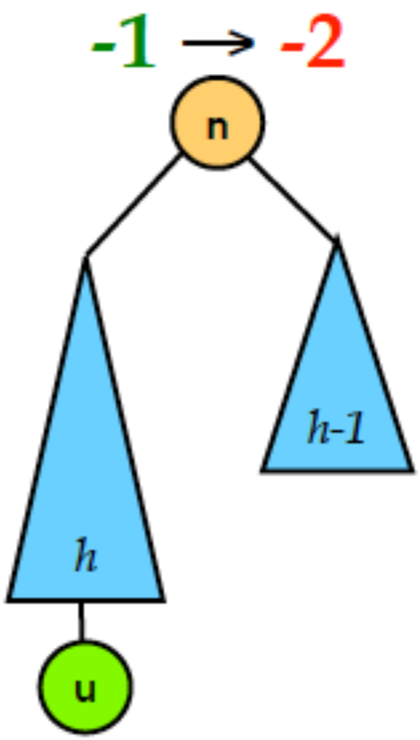


Subtrees were equal, now slightly unbalanced

https://www.cs.umd.edu/class/spring2008/cmsc420/L7.AVLTrees.pdf

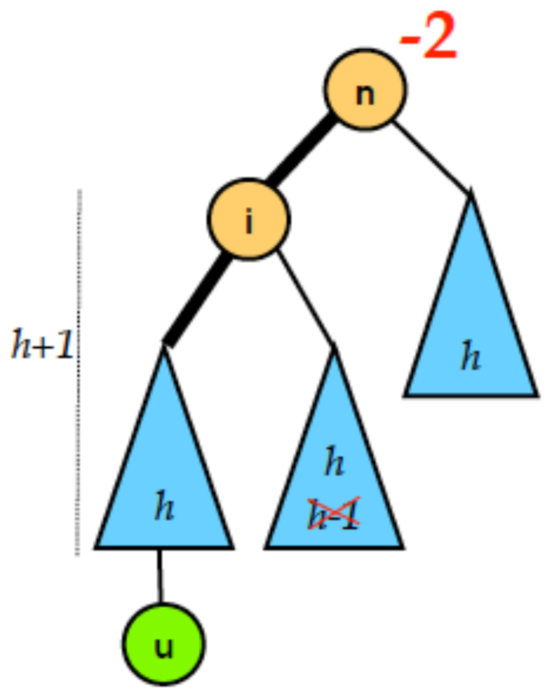
AVL Insert

What to do?

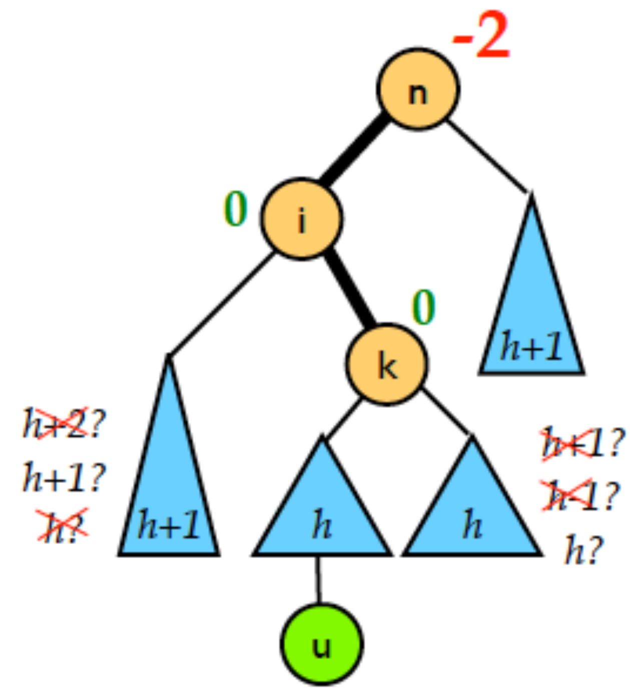


Suppose n is the lowest node that would become -2

Two cases:



Left, Left

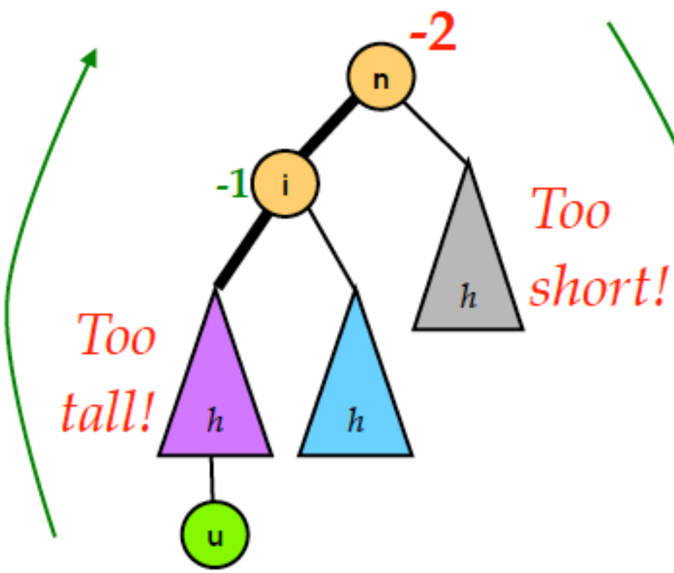


Left, Right

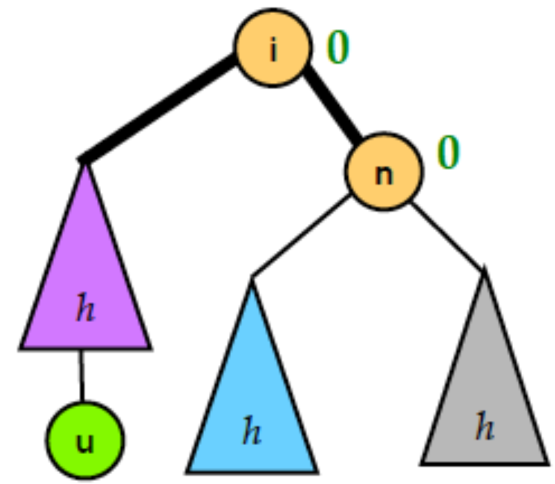
https://www.cs.umd.edu/class/spring2008/cmsc420/L7.AVLTrees.pdf

AVL Insert

Left, Left Case



Right rotation
(aka clockwise rotation)

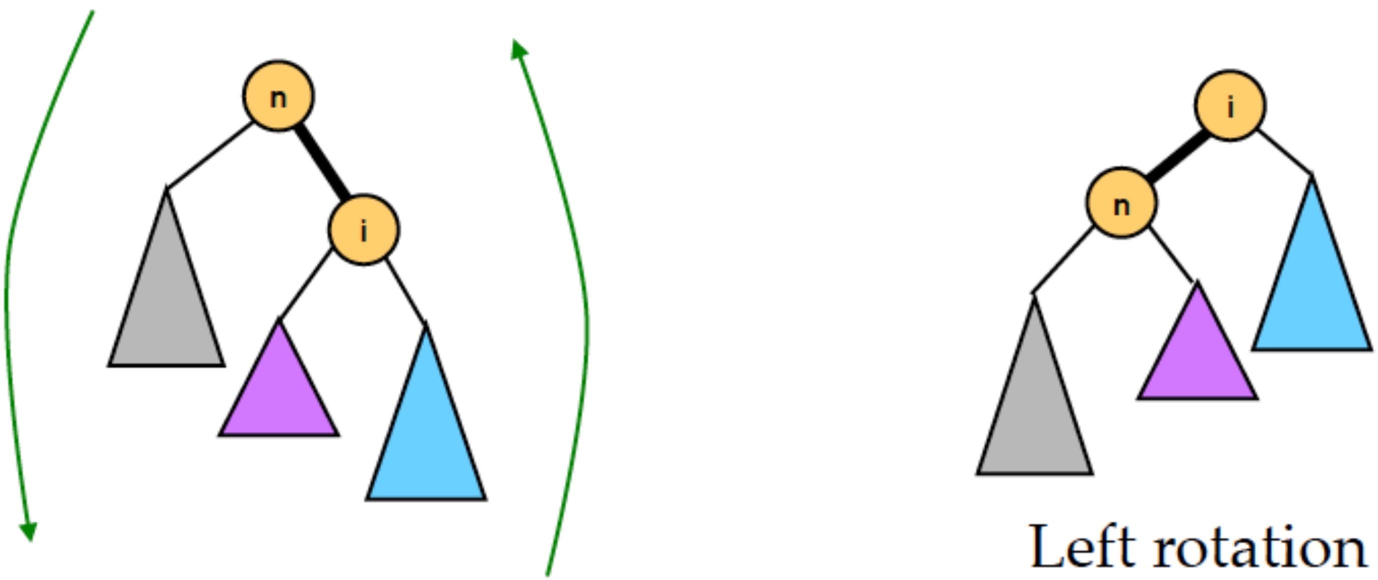


Why does obey BST ordering?

https://www.cs.umd.edu/class/spring2008/cmsc420/L7.AVLTrees.pdf

AVL Insert

Symmetric Left Rotation



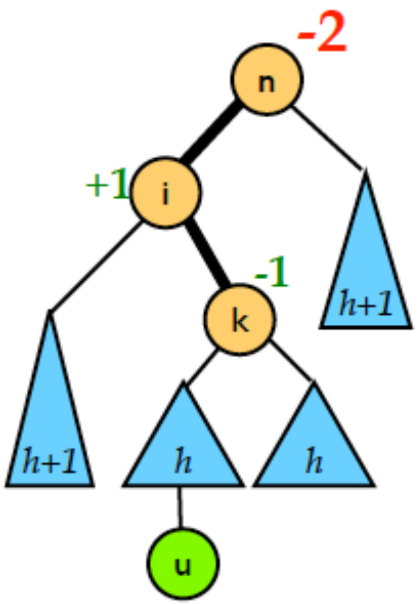
Left rotation
(aka counterclockwise rotation)

Only a constant # of pointers need to be updated for a rotation: O(1) time

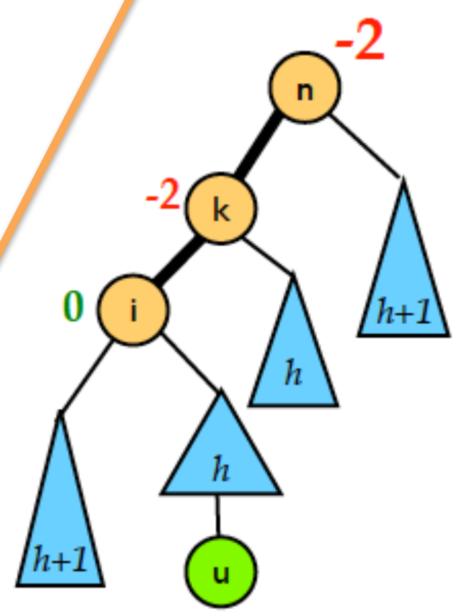
https://www.cs.umd.edu/class/spring2008/cmsc420/L7.AVLTrees.pdf

AVL Insert

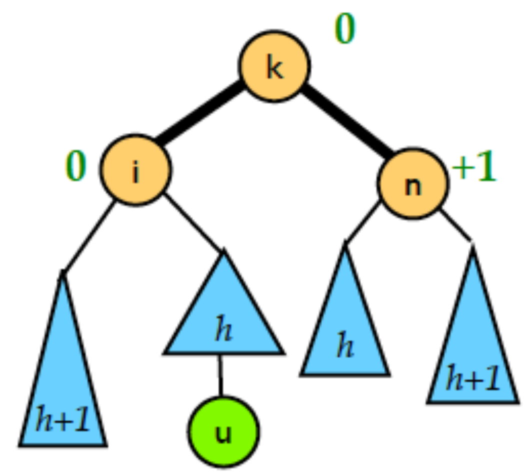
Left, Right Case



Left, Right



(1) Left rotation at i



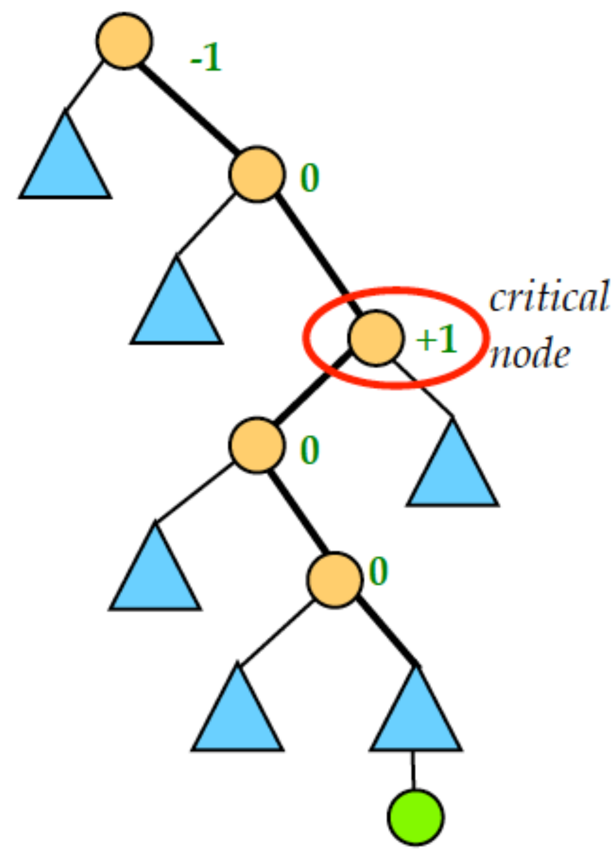
(2) Then right rotation at n

https://www.cs.umd.edu/class/spring2008/cmsc420/L7.AVLTrees.pdf

AVL Insert

- Critical Node
  - node on the insertion path closest to the leaves with balance ≠ 0

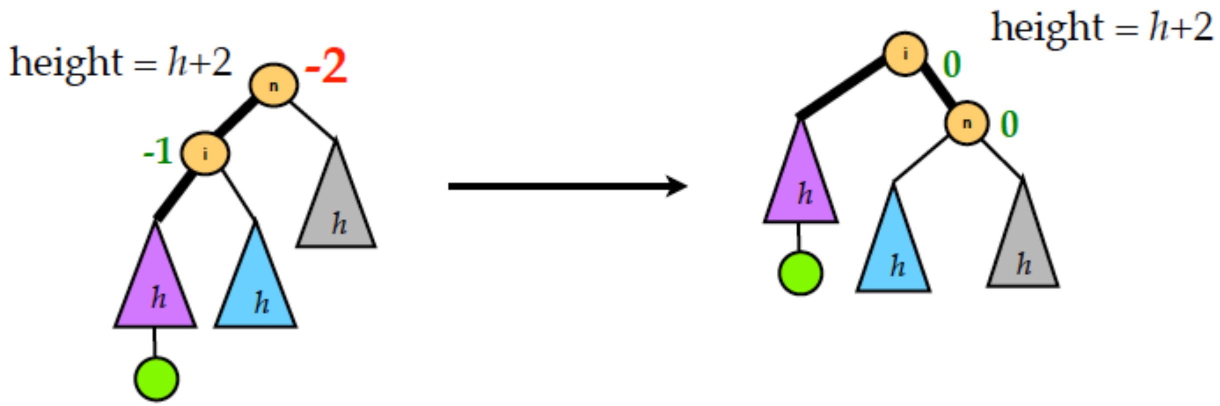
- Rotations leave subtree rooted at critical node balanced with unchanged height.



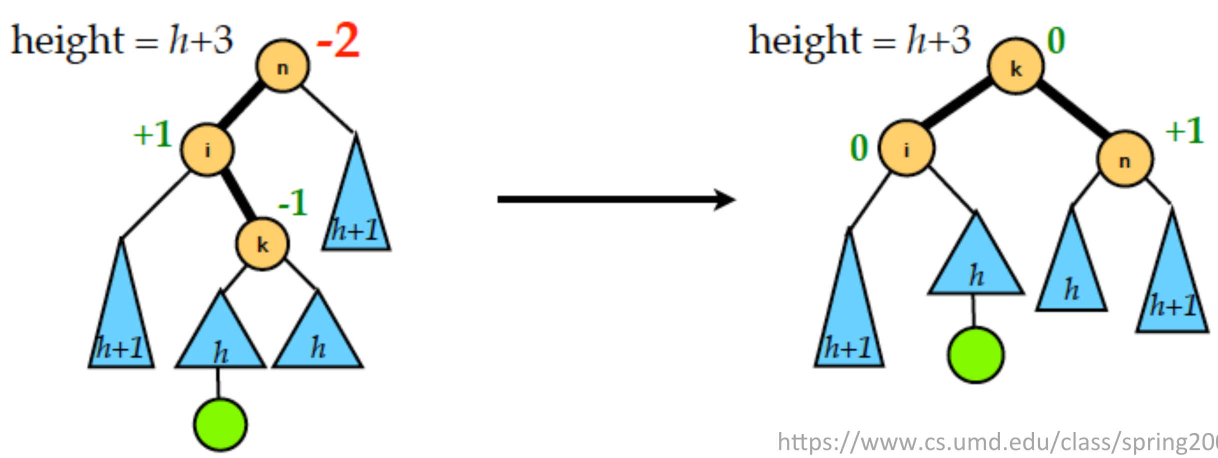
AVL Insert

- Rotations preserve height of critical subtree

Left, Left Case:



Left, Right Case:



https://www.cs.umd.edu/class/spring2008/cmsc420/L7.AVLTrees.pdf

AVL Insert

- Because height of critical subtree doesn’t change, it can’t effect the balance of any nodes higher up in the tree.
- We can stop processing once we process the critical node.
- Therefore, only one rotation will occur.
- Optimization:
  - on first pass down the tree to insert a node, remember the critical node (last node with non-zero balance)
  - Then, to adjust balances, start at critical node and rewalk the path down to inserted node.

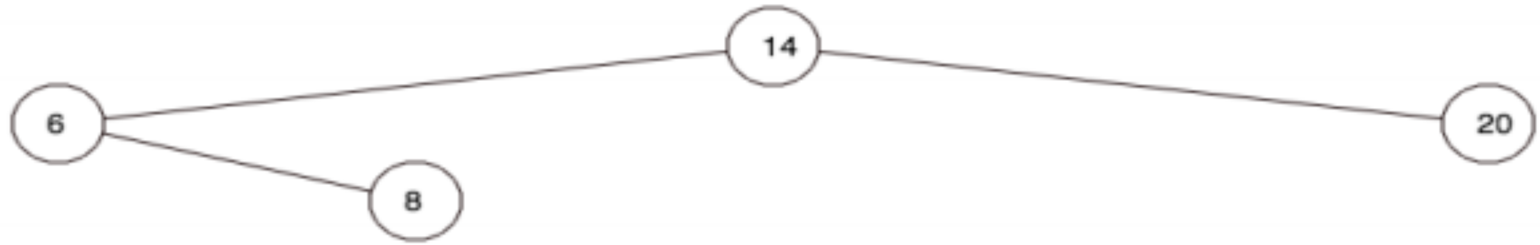
https://www.cs.umd.edu/class/spring2008/cmsc420/L7.AVLTrees.pdf

AVL Trees

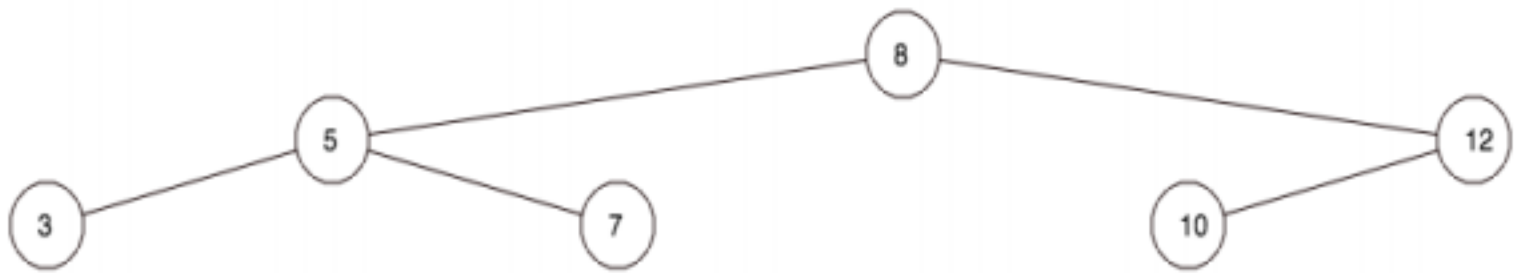
- Nice Features:
  - Worst case O(log n) performance guarantee
  - Fairly simple to implement
- Problems:
  - Have to maintain extra balance factor storage at each node.

QUESTIONS TO PONDER

1. [2 Points] If 10 is inserted into the following AVL tree, draw the resulting tree.

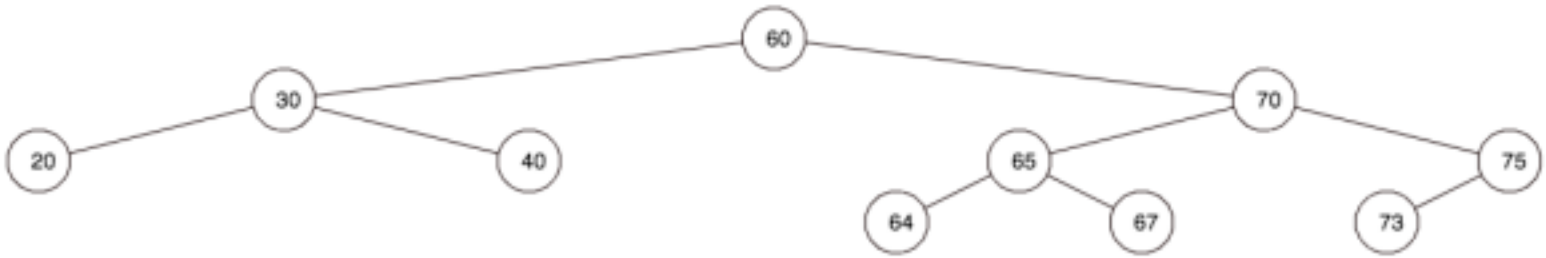


2. [2 Points] If 14 is inserted into the following AVL tree, draw the resulting tree.

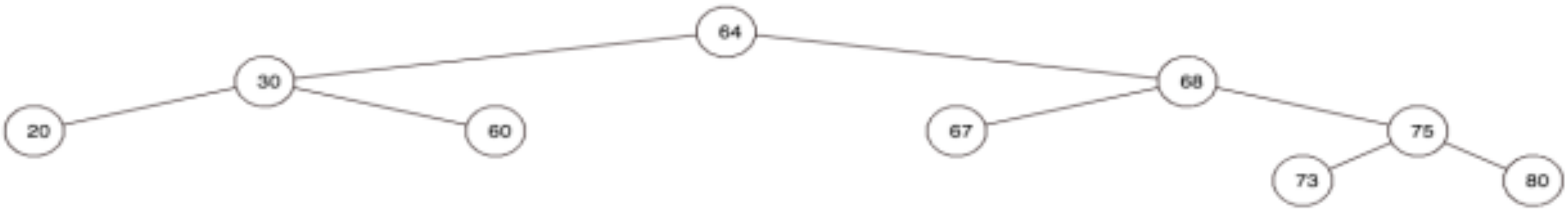


3. [2 Points] If 12 is inserted into the following AVL tree, draw the resulting tree.

4. [2 Points] If 68 is inserted into the following AVL tree, draw the resulting tree.

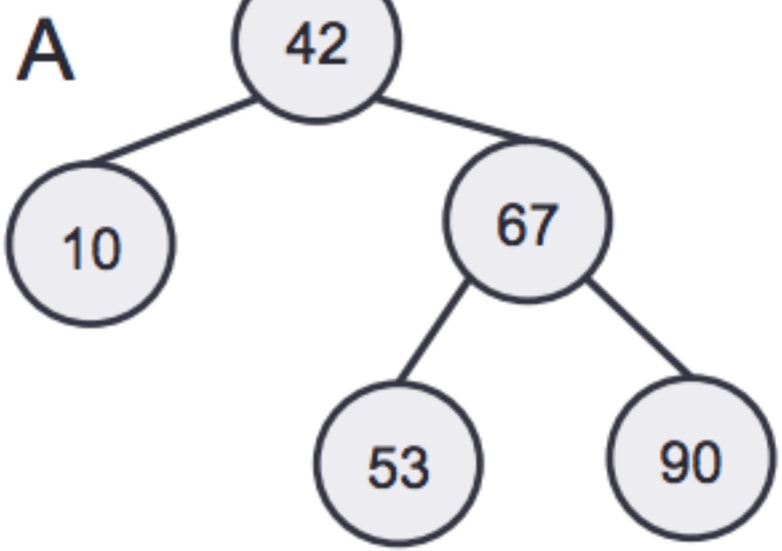


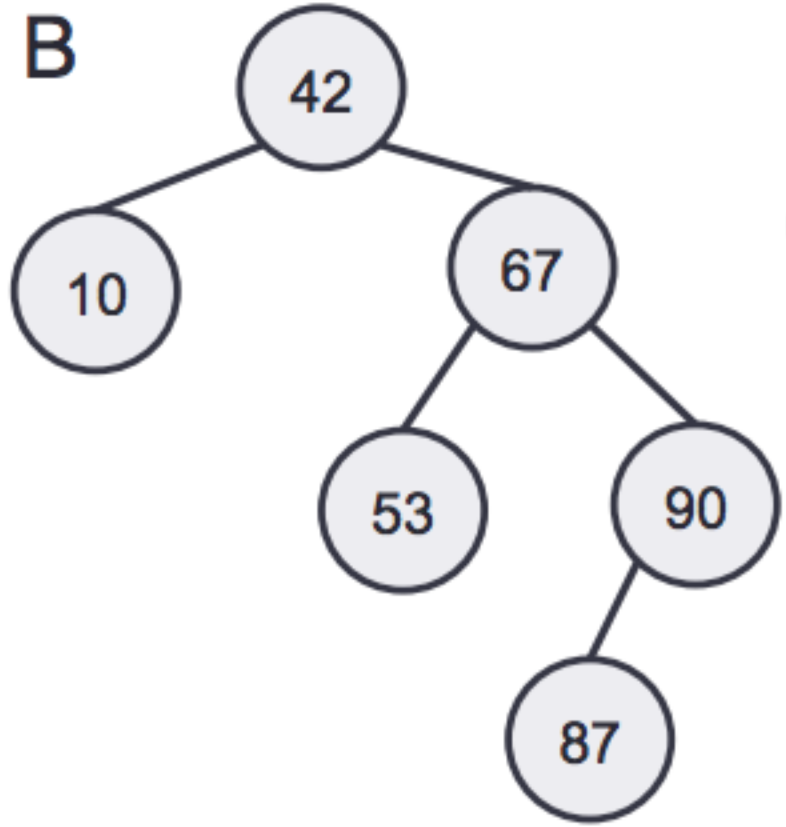
5. [2 Points] If 90 is inserted into the following AVL tree, draw the resulting tree.



Questions to Ponder

6. Which of the following is a balanced AVL tree?





Questions to Ponder

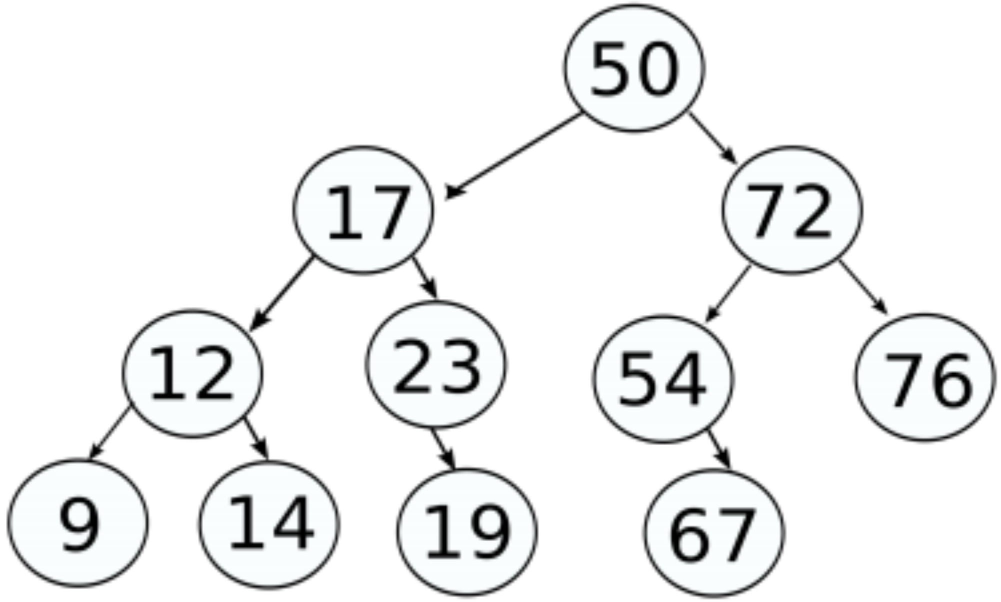
7.

(e) **T F** An AVL tree is balanced, therefore a median of all elements in the tree is always at the root or one of its two children.
Explain:

https://courses.csail.mit.edu/6.006/oldquizzes/solutions/q1-f2008-sol.pdf

8. What are the two invariants that an AVL tree must hold?

5. Given this tree structure:



(a) Assuming the tree is a binary search tree, and not an AVL tree, draw the tree structure created by the following code. Hint: Don't try to do it all at once in your head. Draw each insertion/deletion as an entirely separate tree.

```
t->insert(71);
t->delete(19);
t->delete(23);
t->delete(50);
t->insert(77);
t->insert(78);
t->insert(79);
t->insert(80);
```

(b) Repeat part (a) assuming the tree is an AVL tree.