CSE 326
Lecture 13

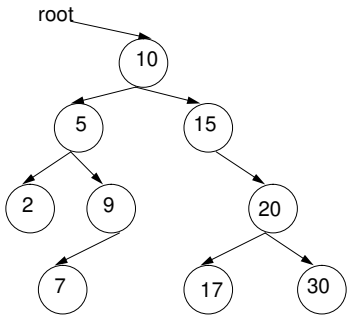
Spring Quarter 1995

Recursive Tree Deletion

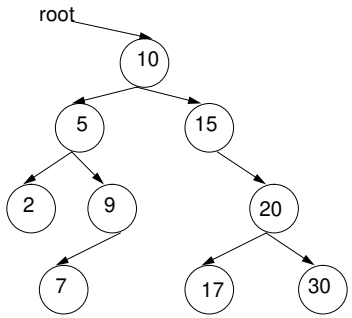
- Recursive reference parameter technique
- Delete(x, ref p) {

```
local pointer q;
if not(p=null) then
  case
    p.data > x: Delete(x, p.left_child);
    p.data < x: Delete(x. p.right_child);
    p.data = x:
        case
          p.left_child := null: p := p.right_child;
          p.right_child := null: p := p.left_child;
          otherwise: {
              q := Successor(p);
              p.data := q.data;
              Delete(q.data,p.right_child)}
        endcase
  endcase }
```


Example of Deletion



Another Example of Deletion



Value vs. Reference Parameters

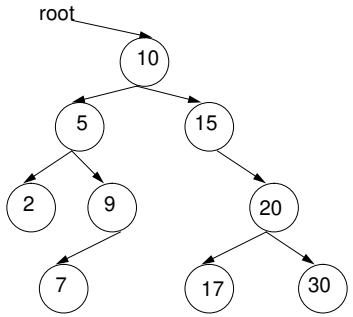
- Value parameters are generally preferred because there are no side effects.
- When to use Reference parameters

  1. The parameter is an array or some large object that needs to be acted on directly.

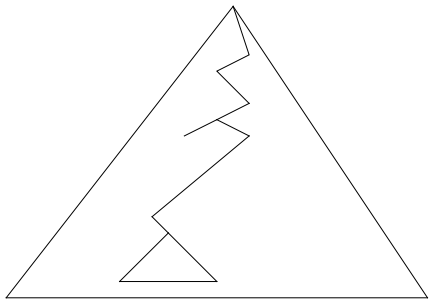
  2. The parameter is a pointer that will need to be changed as a result of the call. Because unintentioned side effects care must be taken!

Iterative Delete by Example

- Three pointers are used: "leader", "parent", and "waiter". It is tricky code with much potential for problems.



General Case of Iterative Delete



Lazy Deletion

- Instead of physically deleting nodes just mark them as deleted

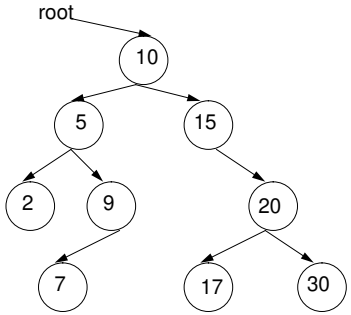
  + simpler

  + physical deletions done in batch mode

  + some insertions can just unmark nodes

  - many lazy deletions slows down insertions and finds

  - some operations like min and max may have to be modified.



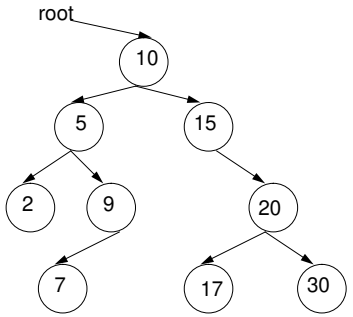
AVL Trees

- AVL trees named after Adelson-Velskii and Landis (1962) is a classic method of keeping trees relatively balanced. If there are n nodes in an AVL tree then it height is bounded by about 1.44 $\log_2 n$.
- AVL tree property:

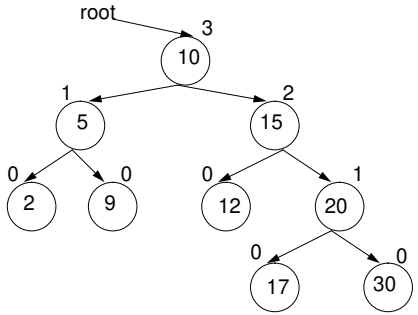
  For each node the heights of its left subtree and right subtree differ by at most 1.
- For this course we will only consider maintaining the AVL Tree property for insertions and not deletions. Deletions are harder. We can use marking and batch deletion.

Testing the AVL Tree Property

- Null's have height = -1.

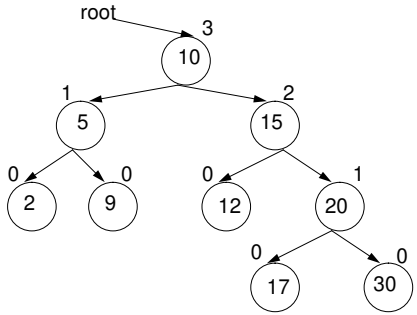


An AVL Tree

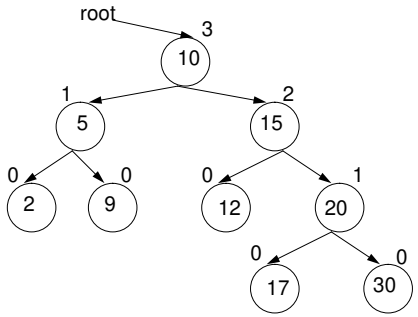


Insert 3

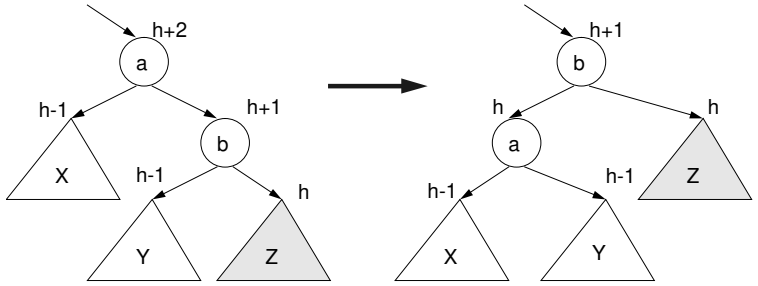
- Note that heights may change on path of insertion, but not elsewhere.



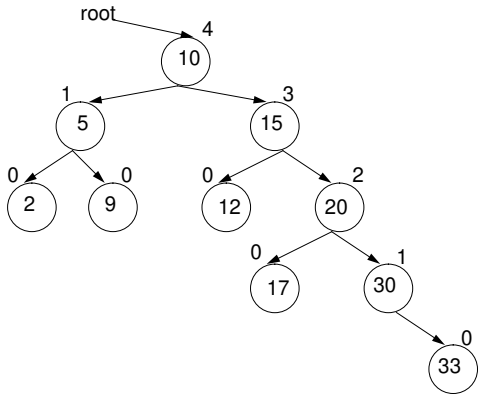
Insert 33



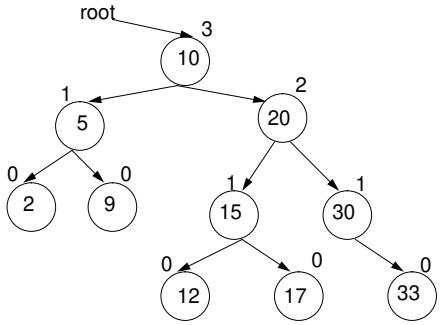
The Single Rotation



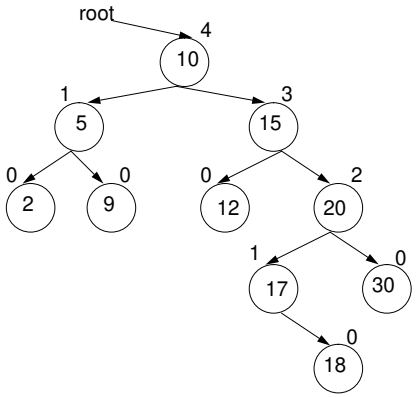
Single Rotation Example



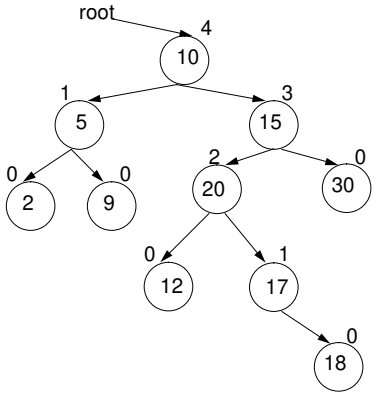
Tree After Single Rotation



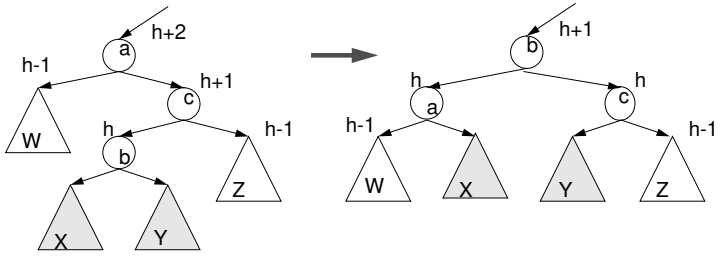
Insert 18



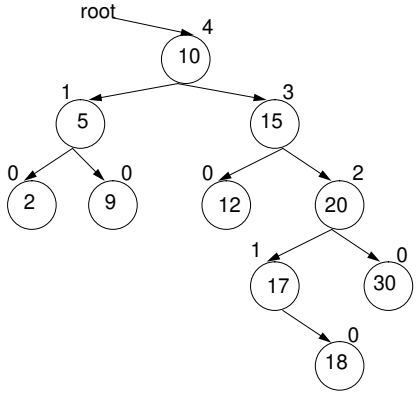
Single Rotation Won't Help



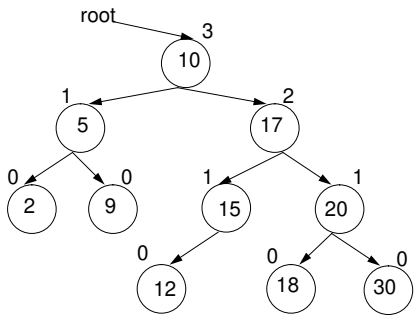
Double Rotation



Double Rotation Example



Tree After Double Rotation



Summary of AVL Insertion

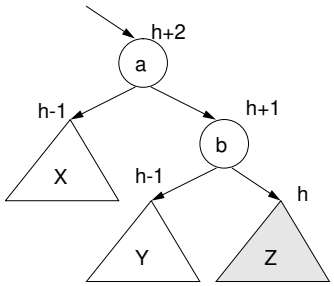
1. Insert the new node keeping track of the insertion path in a stack.
2. Pop the stack incrementing the height field of the nodes found if needed, until a node n is found which is not balanced.
3. if insertion path is right-right or left-left from n then do a single rotation at n.

   if insertion path is left-right or right-left then do a double rotation at n.

Single Rotation Code

- Left-Rotation(n) {

  local node pointer x,y,z; integer a;

  x := n.left;
  y := n.right.left;
  z := n.right.right;
  n.left := n.right;
  n.left.left := x;
  n.left.right := y;
  n.right := z;
  n.height := n.height - 1;
  n.left.height := n.left.height -1;
  a := n.data;
  n.data := n.left.data;
  n.left.data := a }

- No knowledge of parent of n needed