CS180 - Binary Search trees

Note Title 11/7/2011

Announcements

- HW due next Tuesday
- Next HW will be up today
  due 1 week from Monday
- Fun with trees for ~2 weeks

Last time: Priority Queues

- insert(e): add e to our data structure
- getMax(): return element with maximum key (its e)
- removeMax(): delete element with maximum key

With vectors or lists: O(n) (for one function)

Last time: Heaps

complete

A binary tree where we maintain an invariant:

- Any node's value is ≤ its parent's value.
- Complete binary tree.

So where is maximum value? root

Result: O(log n)

(Code is on webpage.)

Inserting & Deleting

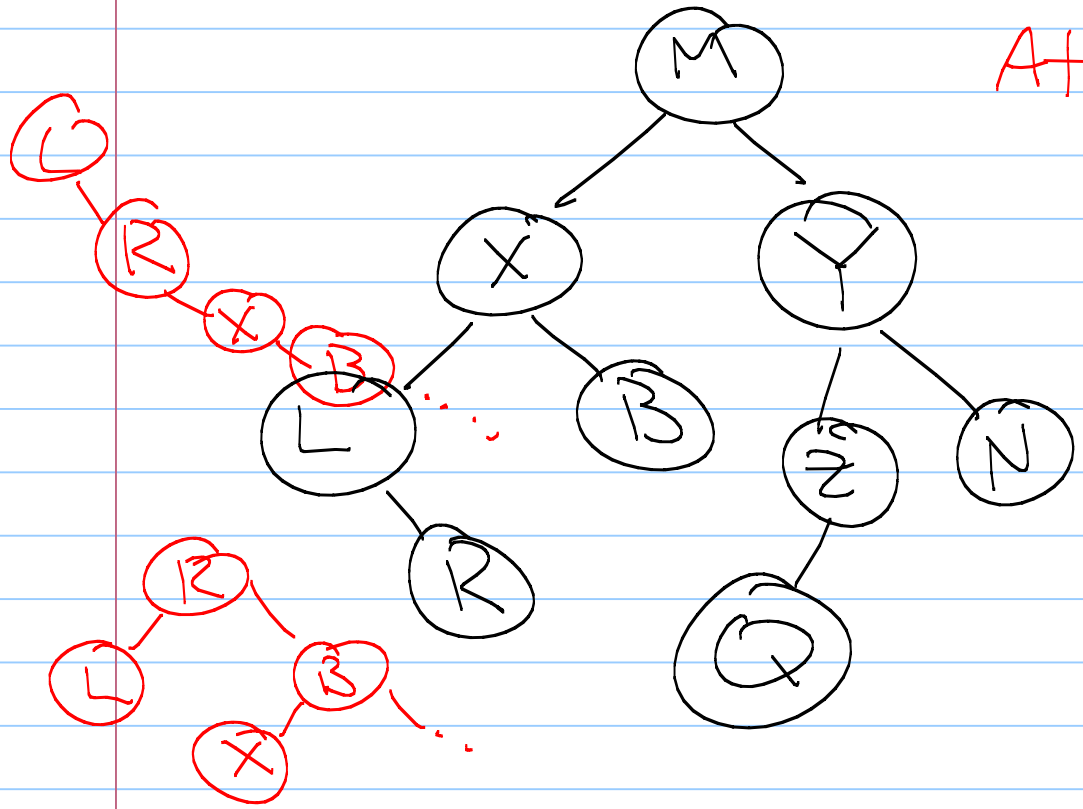
Idea: Maintain structural property at all times.

Then bubble up or down to fix the orderings.

Runtime: $O(\log n)$

Tree Traversals : method of visiting every node

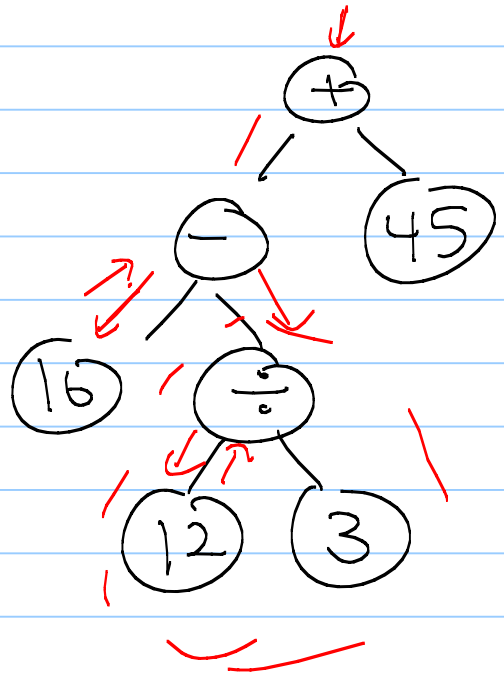
Inorder Traversal : recursive procedure:



At node v:
recursely go left
print v
recursively go right

L R X B M Q Z Y N

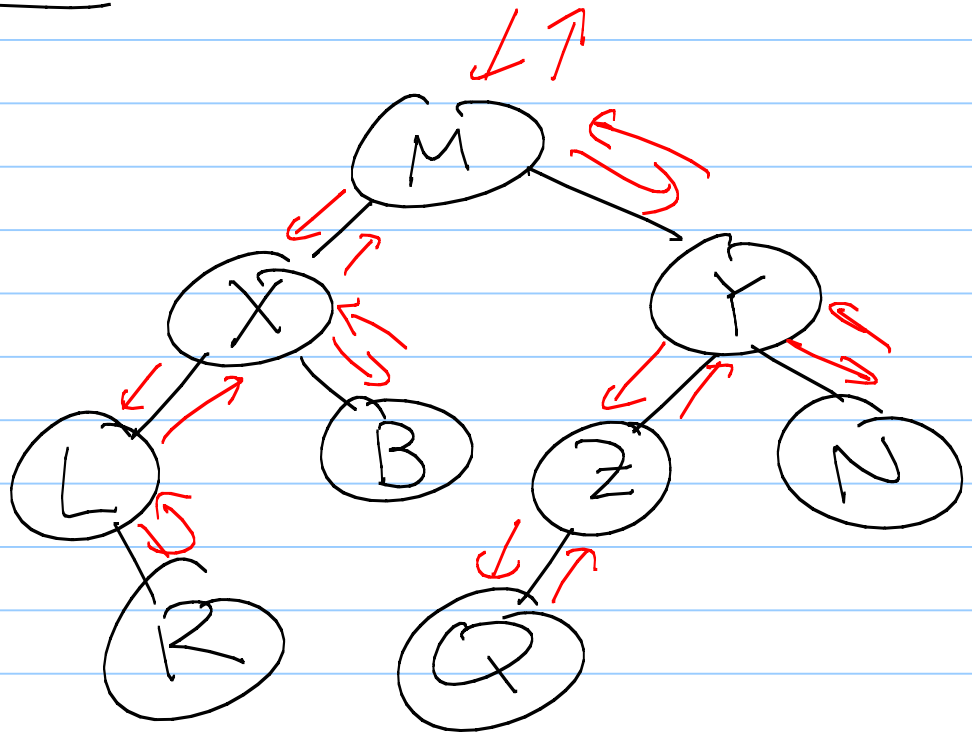
A use for inorder : precedence of operations



$(16 - (12 \div 3)) + 45$

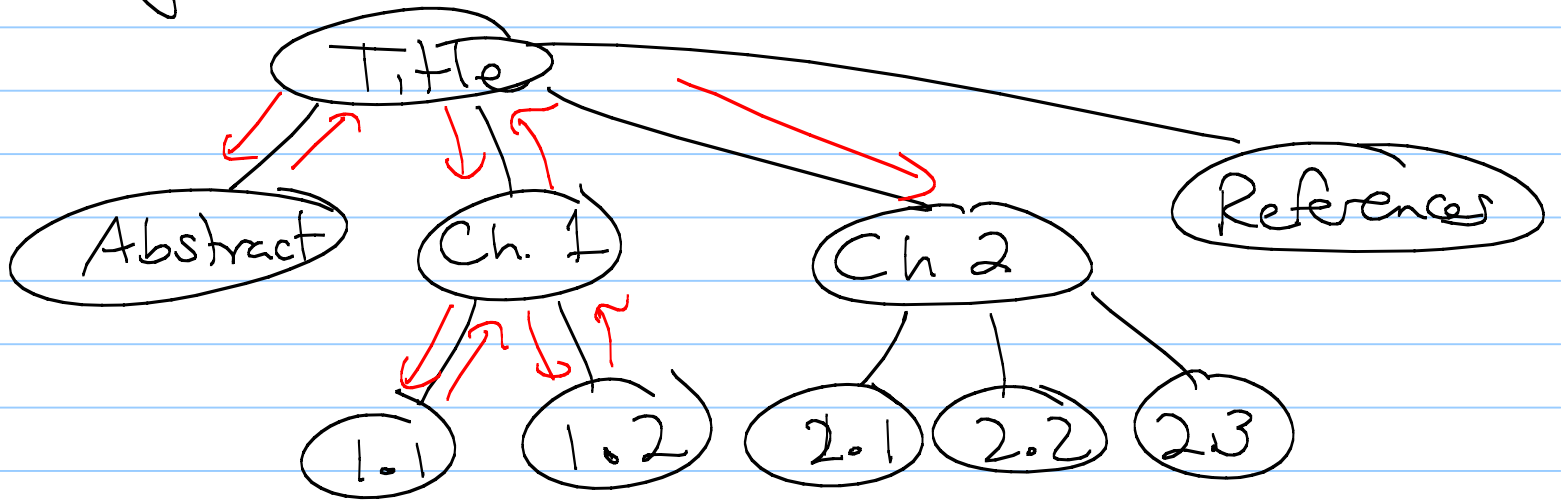
Preorder Traversal

At v:
print v
recurse left
recurse right



MXLRBYZQN

Preorder (good) example:



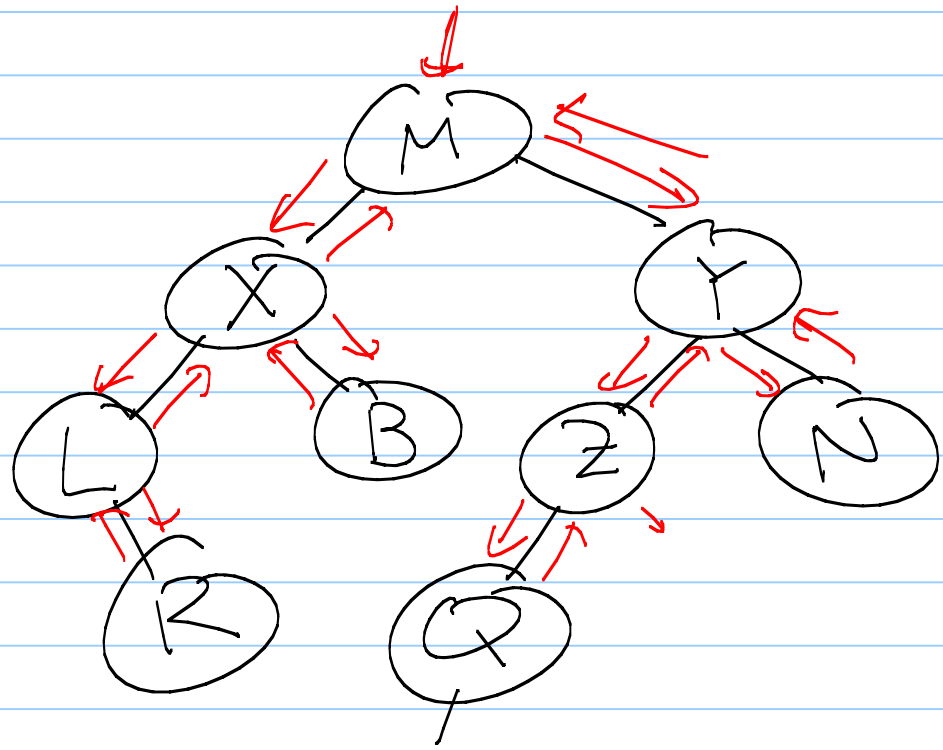
Title
Abstract
Ch 1
1.1
1.2
Ch 2
2.1
2.2

a

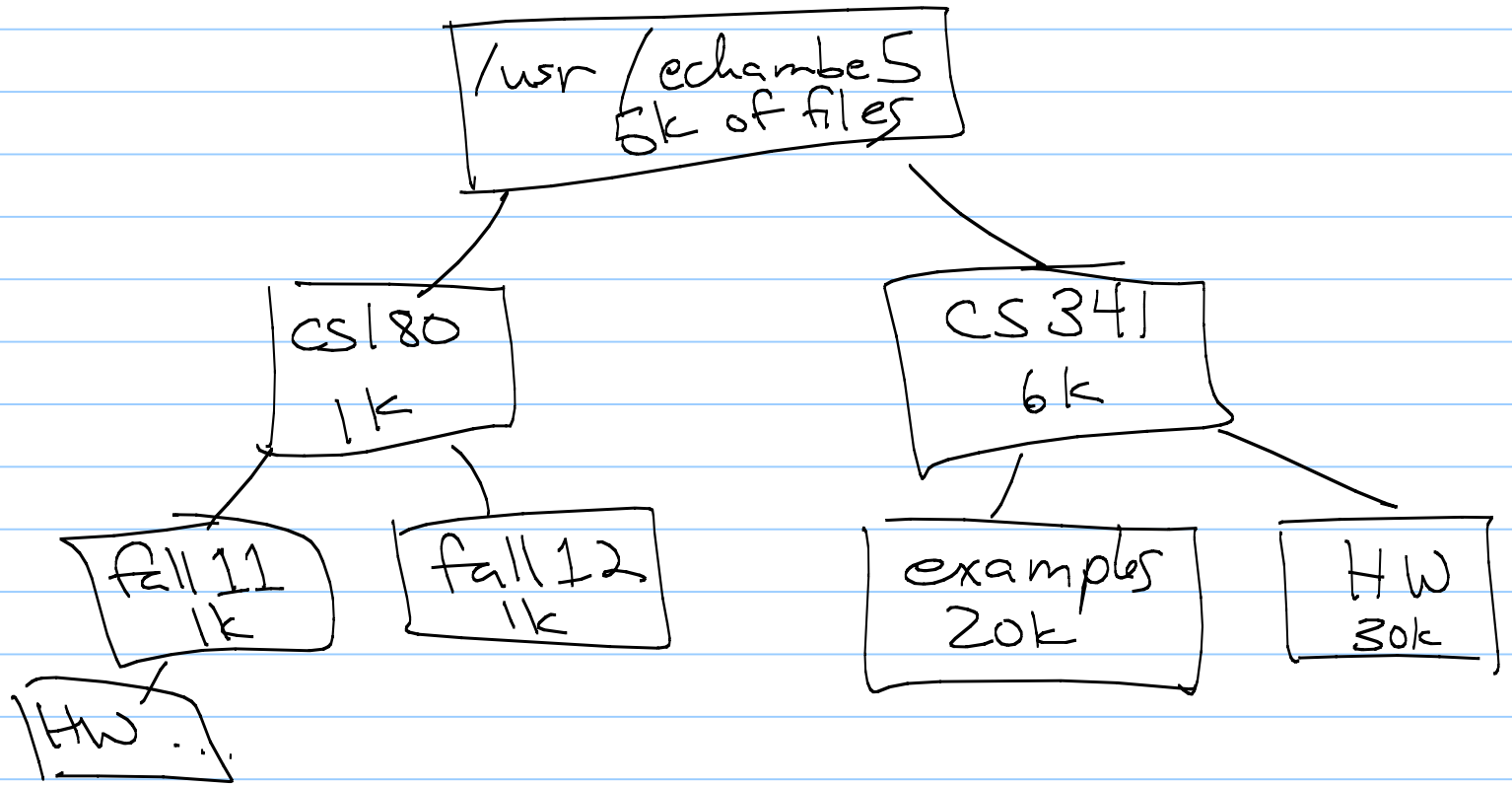
Post order Traversal

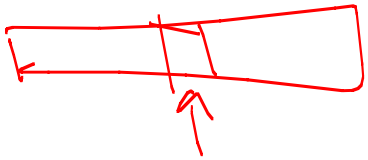
at v:
recurse left
→recurse right
print v

RLBXQZNYM



Postorder use: Calculating file system sizes!





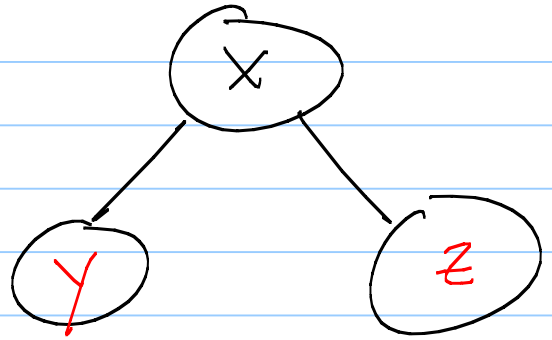
Binary Search Trees

A binary tree where we maintain the following:

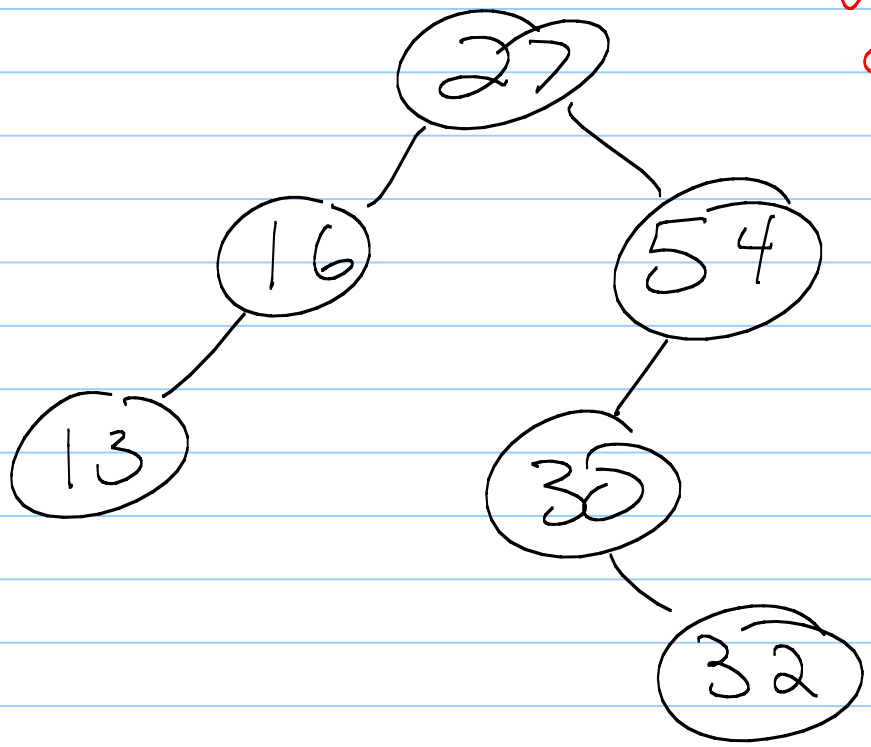
The value at any node is $\geq$ its left child and $<$ its right child.

$y \leq x$

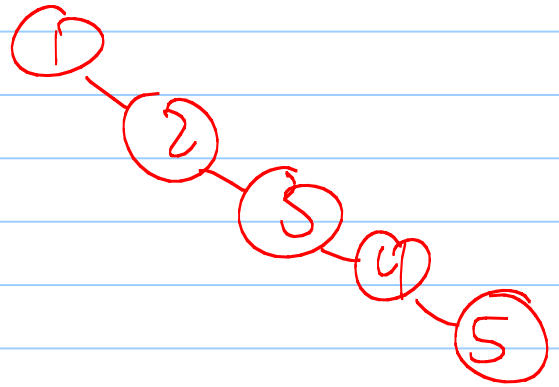
$z > x$



Example:



- User can't directly modify the tree.
- Not balanced

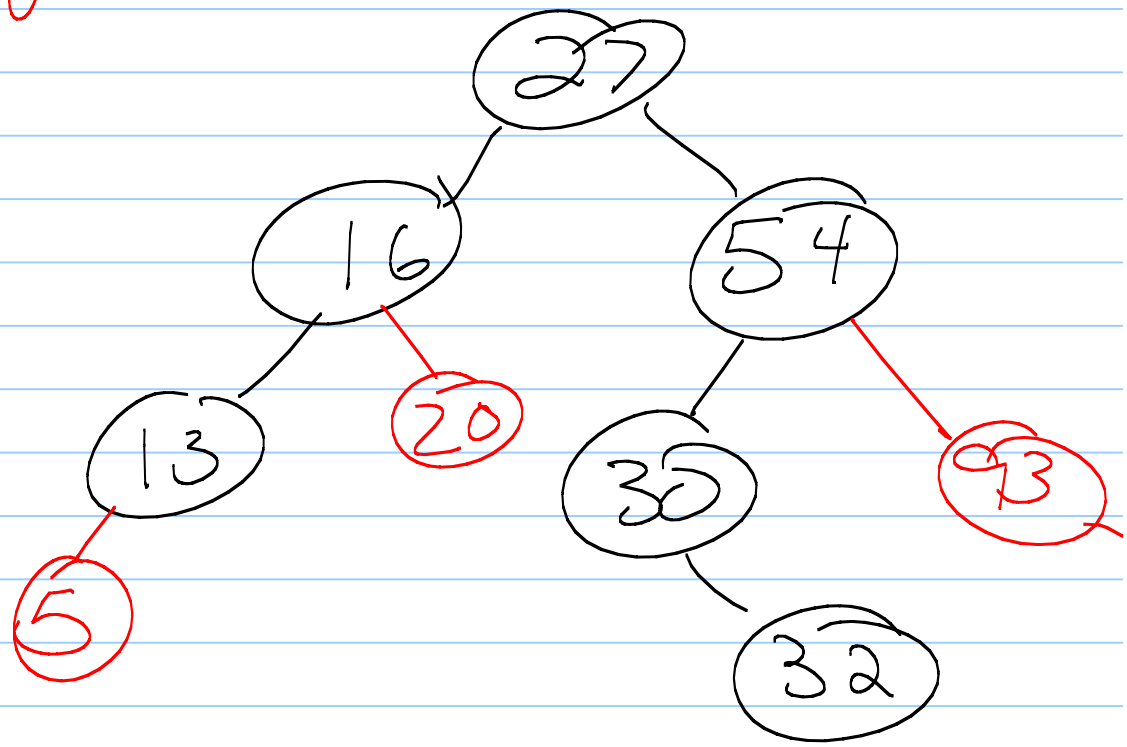


Insert : exactly 1 place a value can go

insert (20)

insert (5)

insert (93)

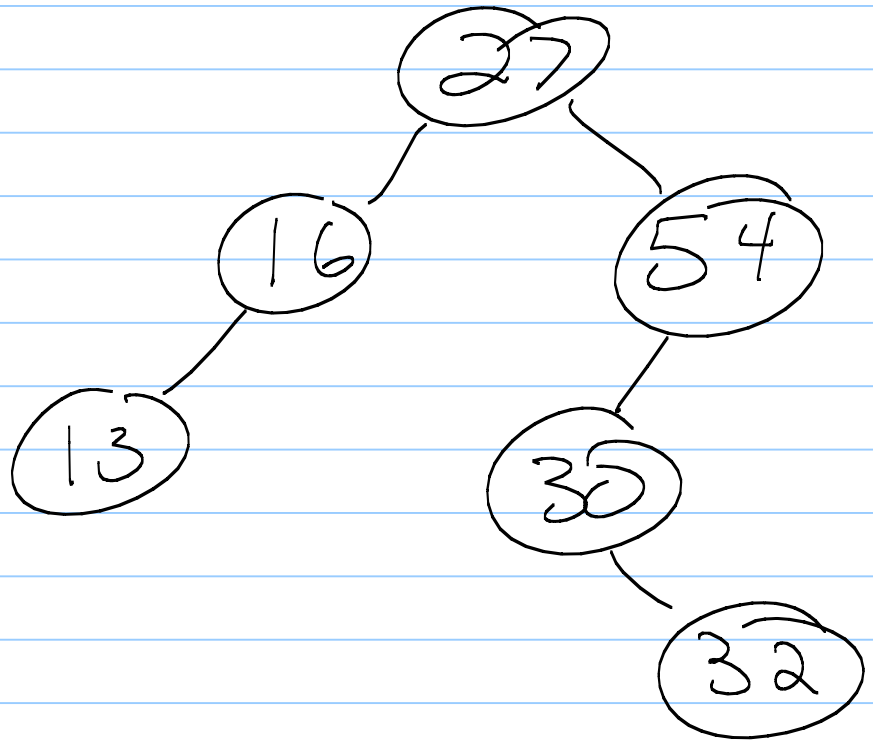


Essentially:
- run find
- put it there

Find

Start at root

do binary search



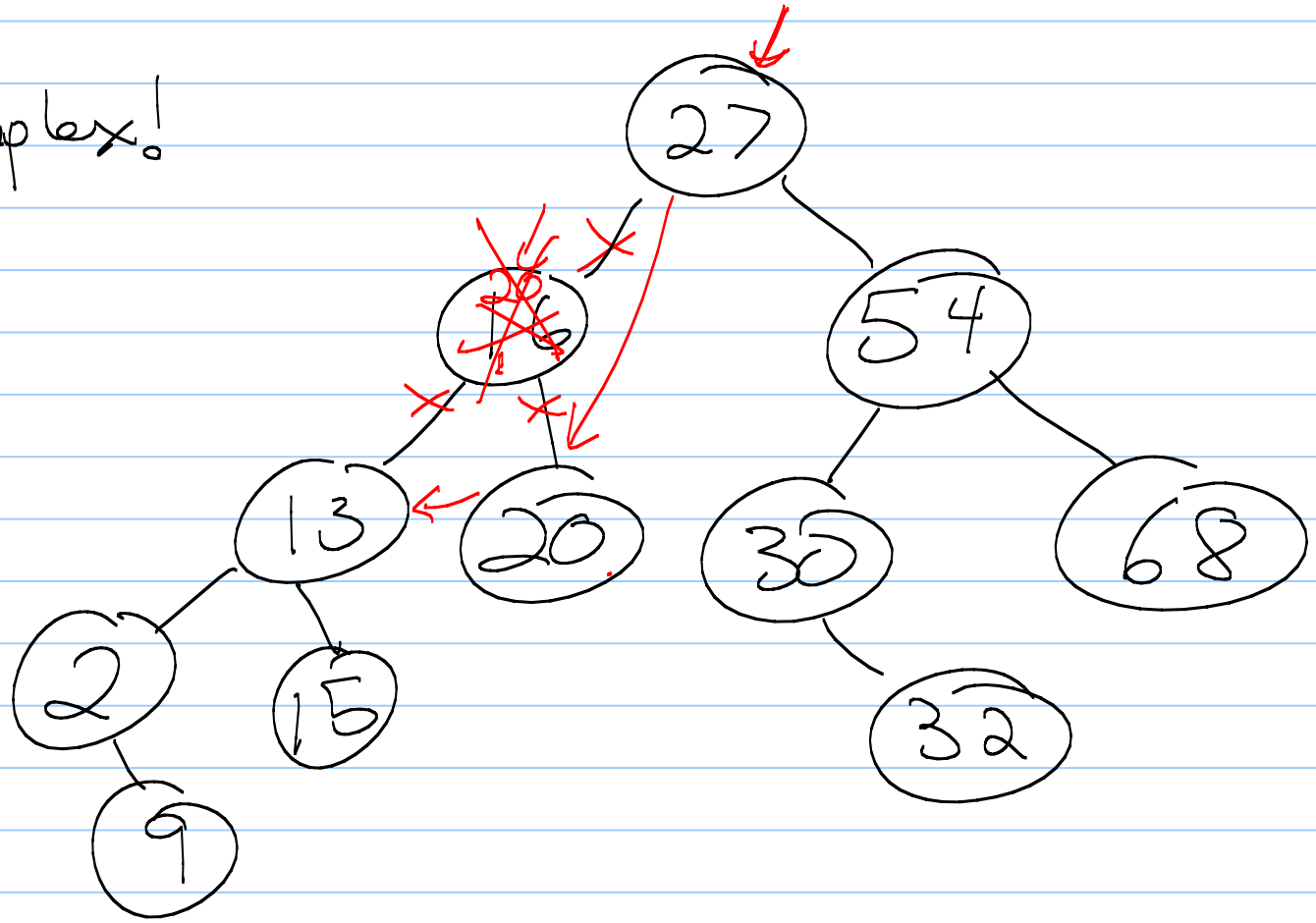
Delete:

More complex!

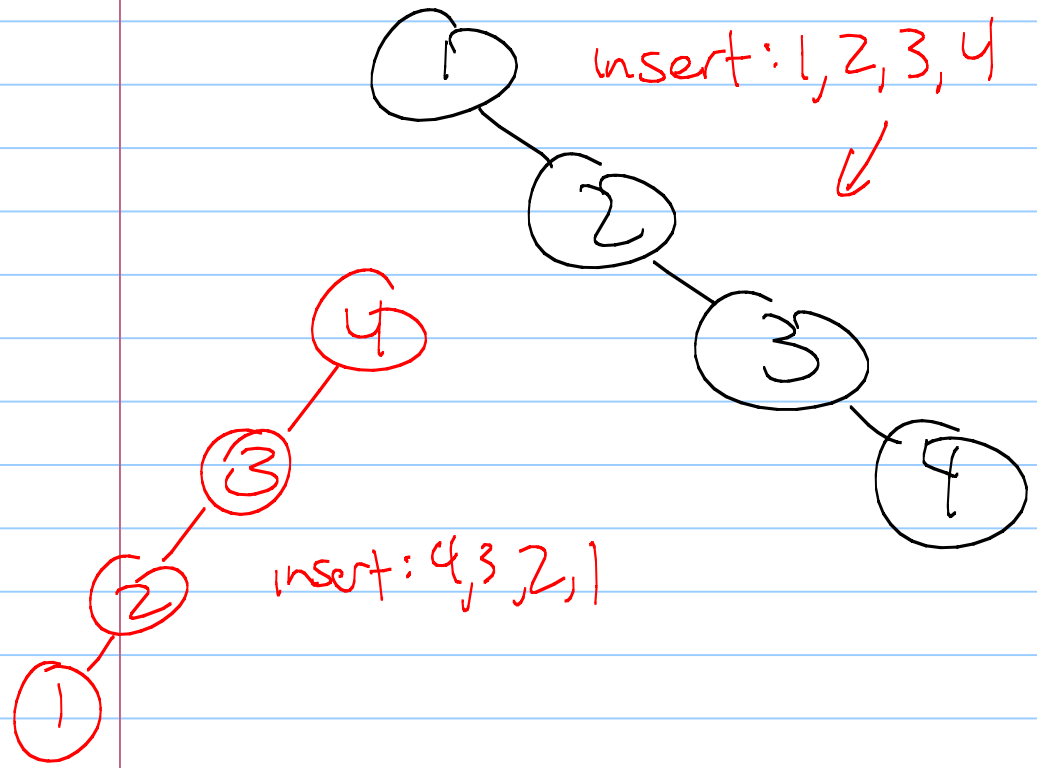
delete (19)

delete (16)

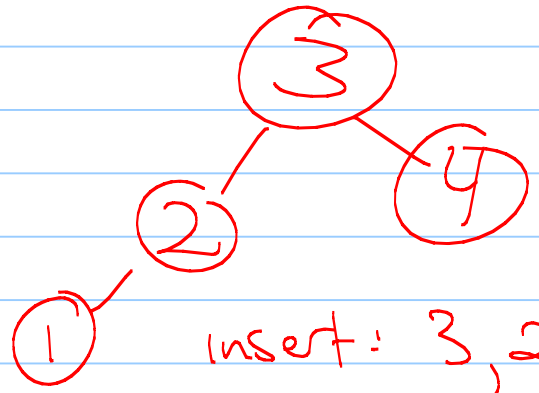
delete (20)



Note: BSTs are not unique!



Can you make another ~~BST~~ with these elements?



3, 4, 2, 1

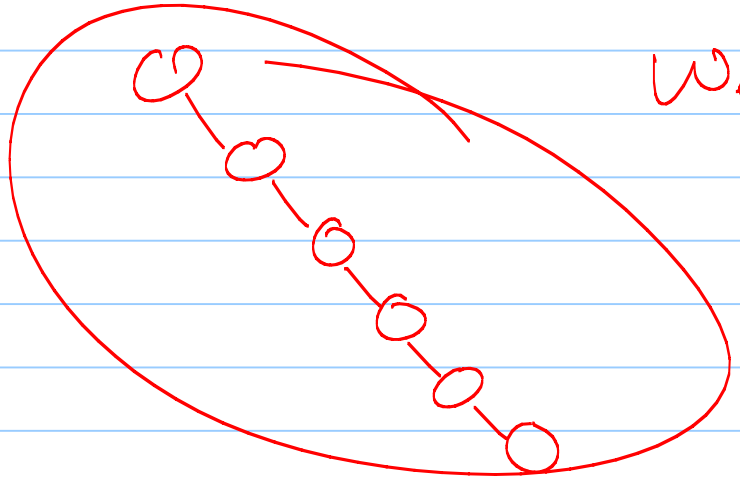
Runtimes:

Find: $O(\text{height}) = O(n)$

Insert: $O(n)$ ← find

Delete: $O(n)$ ← find

} worst case



We'll fix this later.

Code

- Will be pointer based. Why?

not complete trees

(Need nodes, iterators, etc.)

Today:

Code for generic binary trees.

BinaryTree.h will be generic — not BSTs.

BST.h will inherit from BinaryTree.h (but so will other classes.)