CS180 - Binary Search trees

Note Title 11/7/2011

Announcements

- HW due Friday
  (make sure your code works)
- Practice exams out today
- Review Friday, test Monday
- No lab this week or next

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 lists or vectors, some operation will take $O(n)$ time

Last time: Heaps

complete

A binary tree where we maintain an invariant:

- Any node's value is ≤ its parent's value.

So where is maximum value?

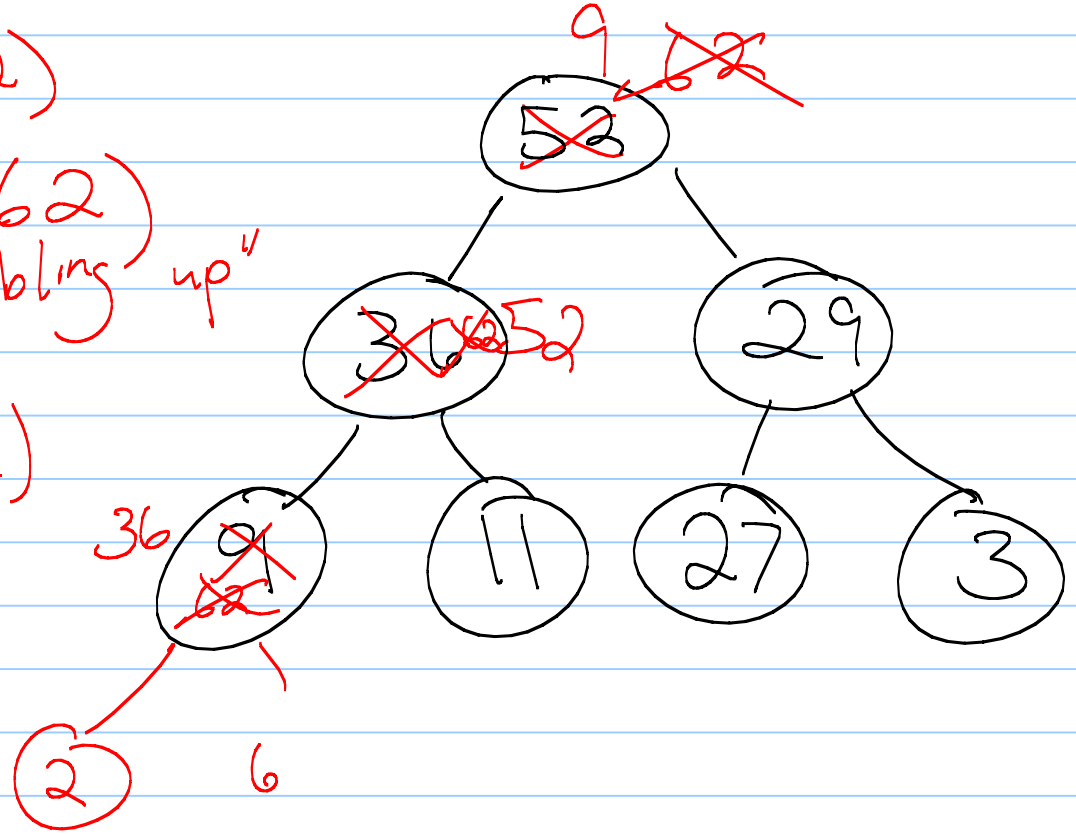
root

Inserting & Deleting

insert (2)

insert (62)
"bubbling up"

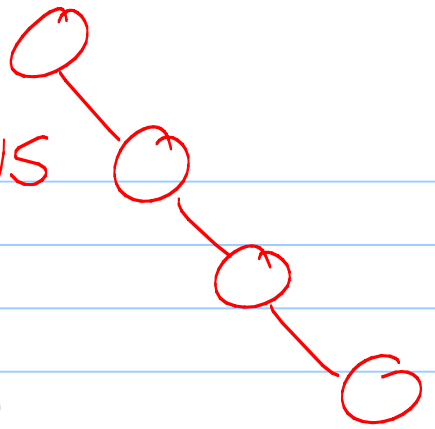
removeMax()



$1+2=3$

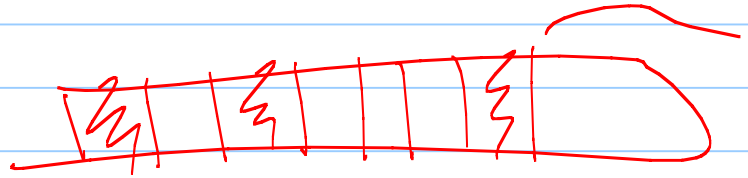
$1+2+4=7$

$1+2+4+8=15$



Code for heaps

- Start on Wednesday
- Array based. (How?)

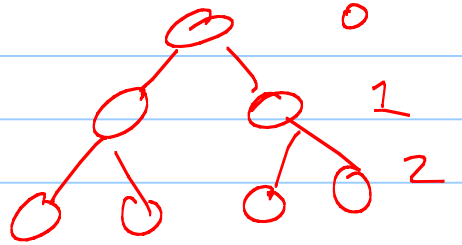


Running times?

$O(h) = O(\log n)$

$$\sum_{i=0}^{h} 2^i = n \qquad 2^{h+1} - 1 = n$$

$$\log_2(2^{h+1} - 1) = \log_2 n \Rightarrow h + 1 = \log_2 n$$

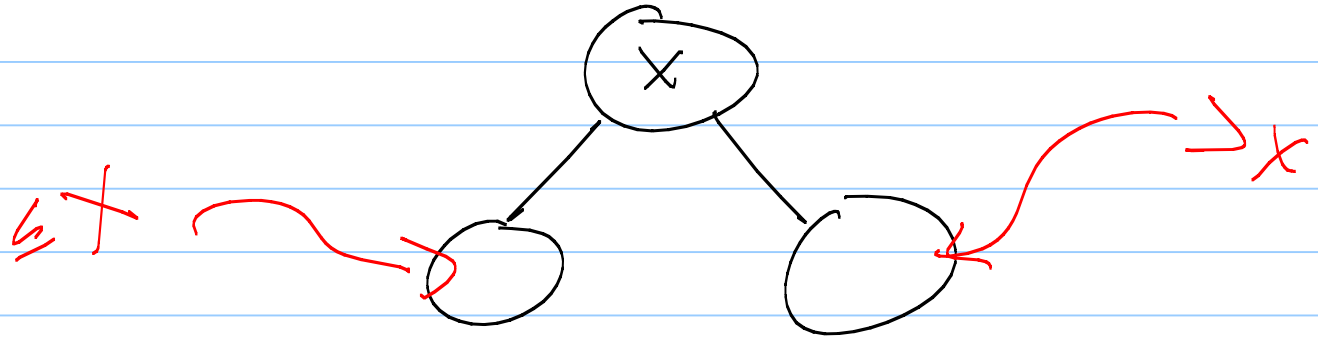


level i: 2^i nodes

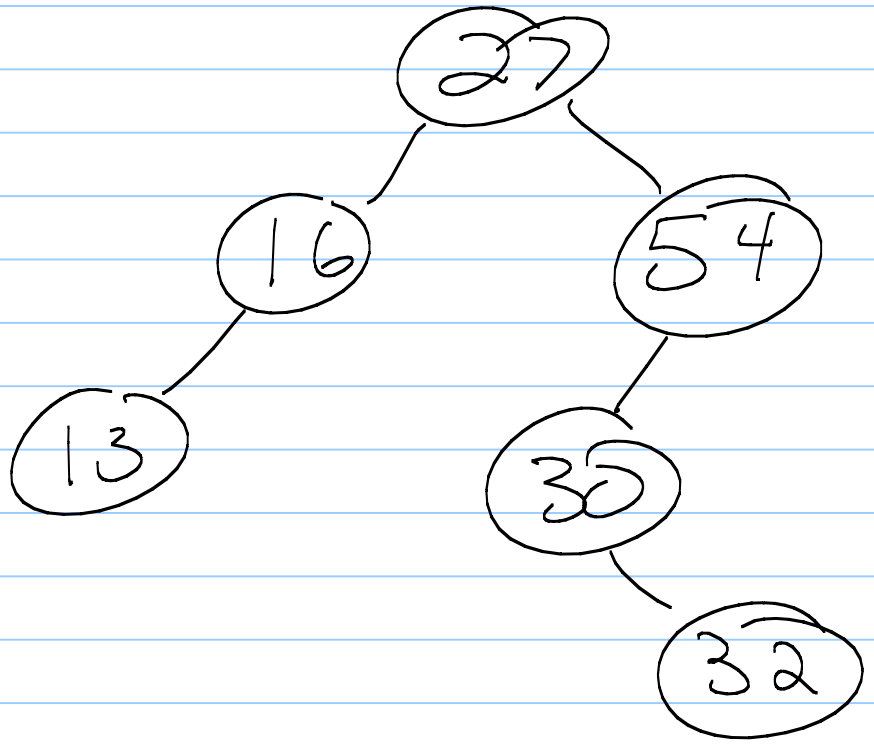
Binary Search Trees

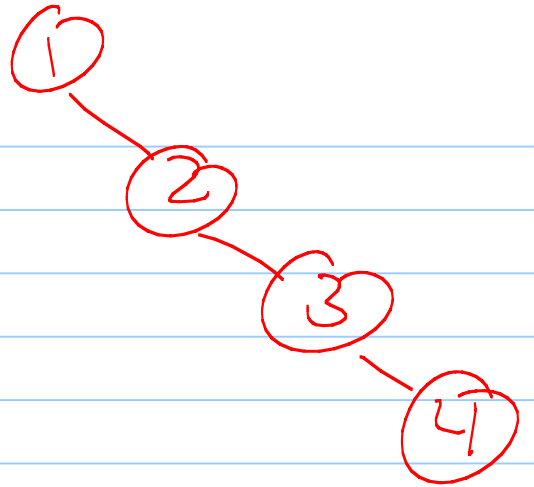
A binary tree where we maintain the following:

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



Example:



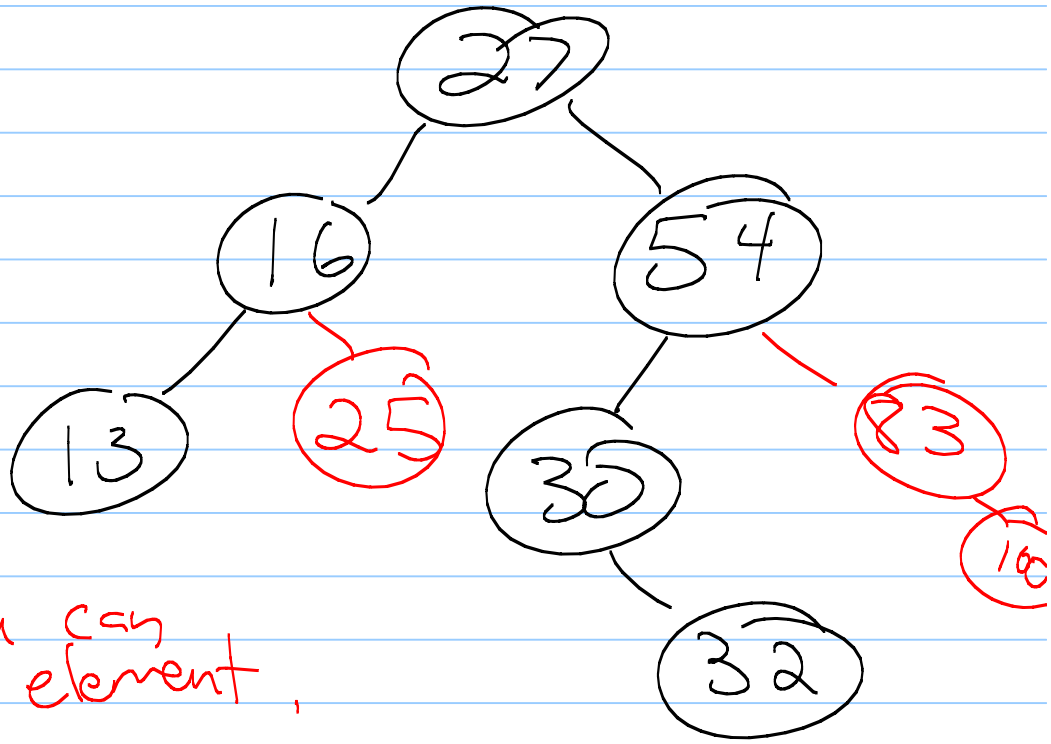


Insert

insert (83)

insert (100)

insert (25)



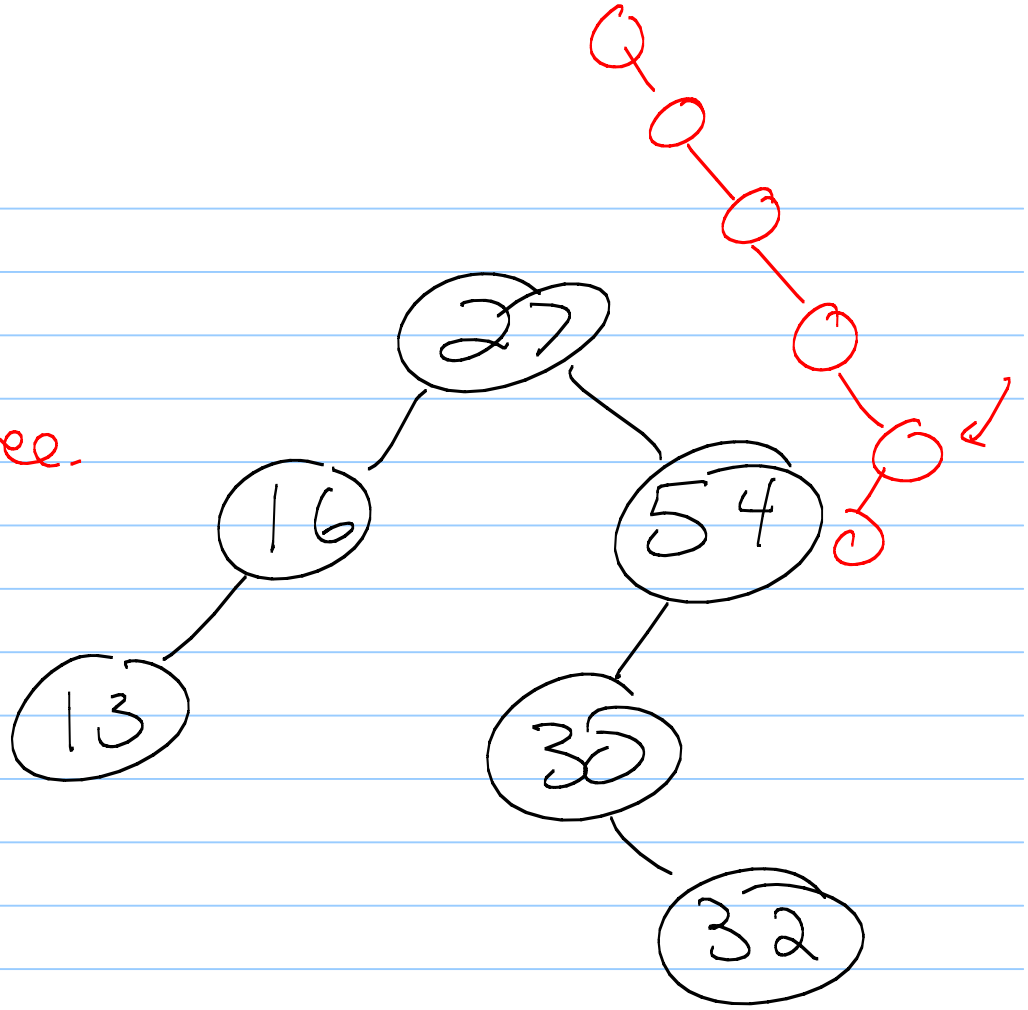
Only 1 position you can insert a given element.

Find

Check if 58 is in tree.

Find 30

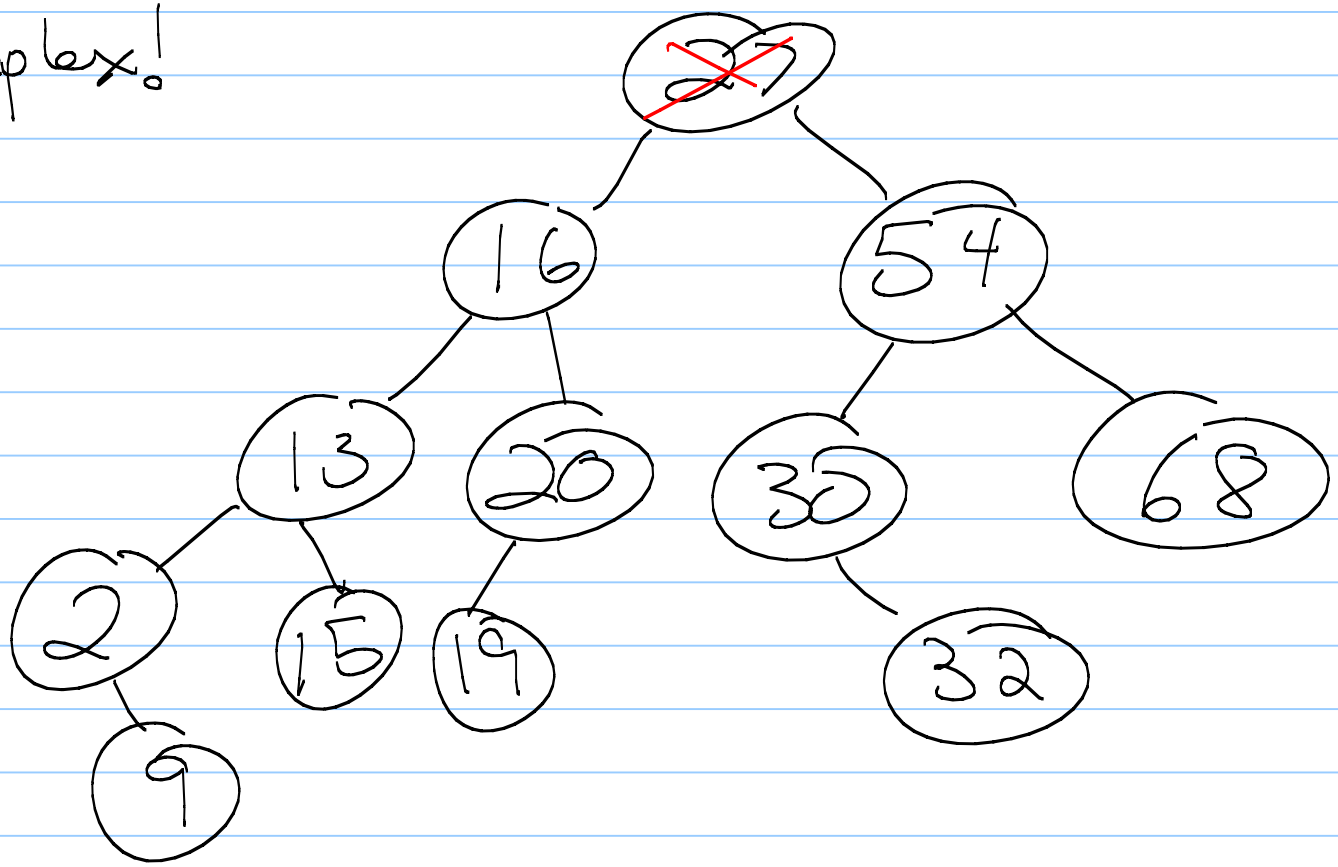
$O(h) = O(n)$



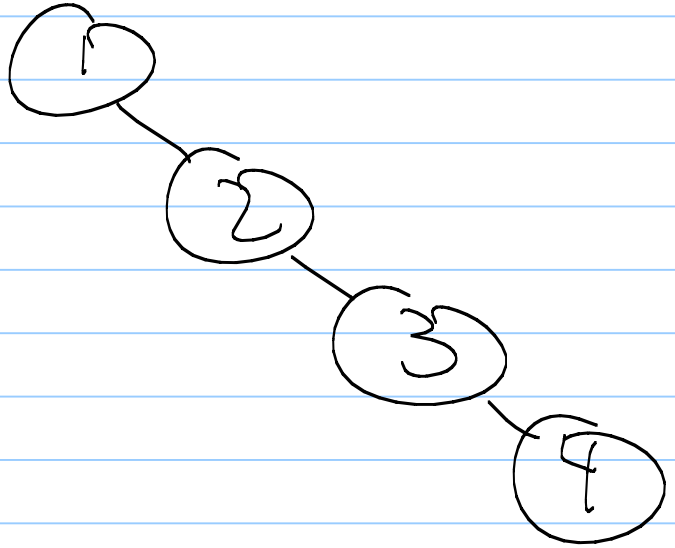
Delete:

More complex!

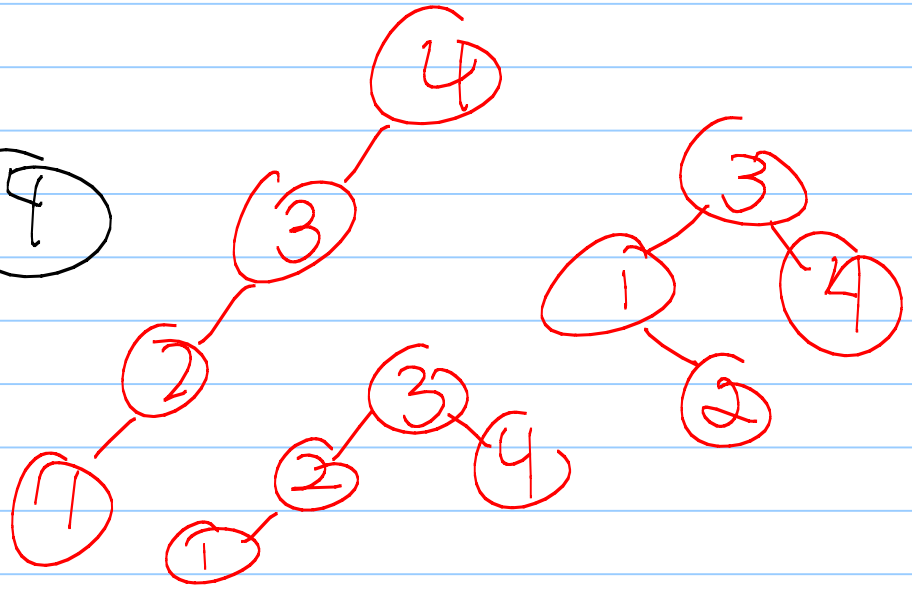
remove (27)



Note: BSTs are not unique!



Can you make another BST with these elements?



Runtimes:

Find:

Insert:

Delete:

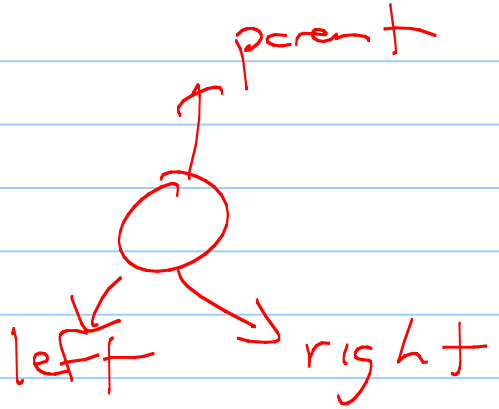
$O(n)$ (because in worst tree, $h = n$)

Code

- Will be pointer based. Why?

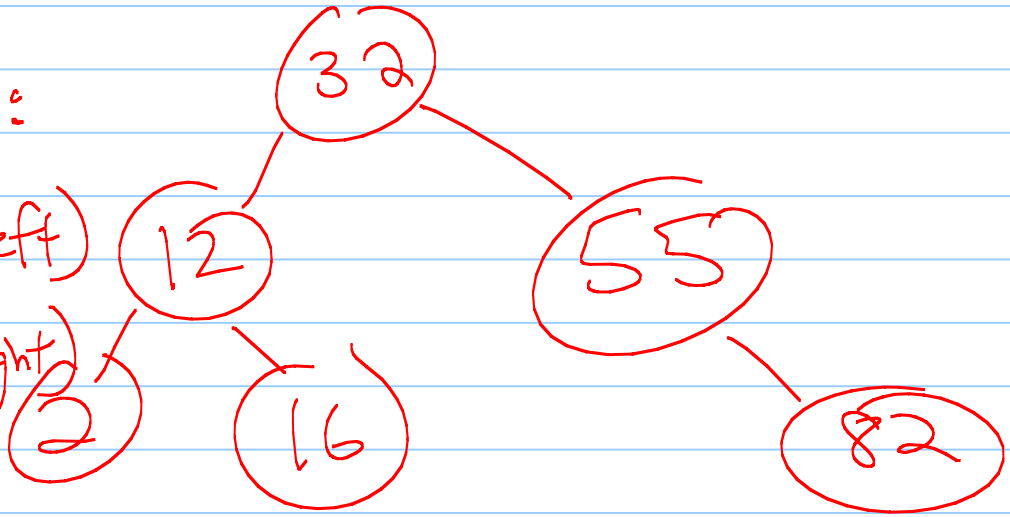
not complete, so array may waste space

(Need nodes, iterators, etc.)



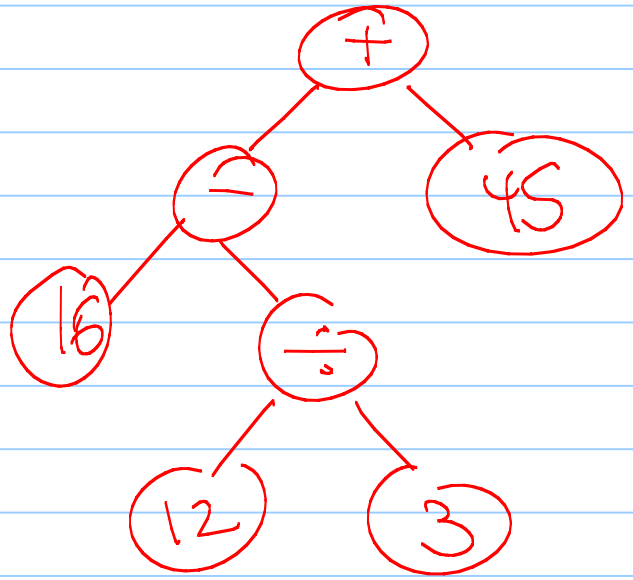
Tree traversals

```
InOrderPrint(n):
  if n != NULL
    InorderPrint(n->left)
    print n
    InorderPrint(n->right)
```



Inorder traversal: 2, 12, 16, 32, 55, 82

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



Pre order & past order

print n
Preorder (n->left)
Preorder (n->right)

postorder (n->left)
postorder (n->right)
print n

examples next time