# Computer Science 112
# Data Structures

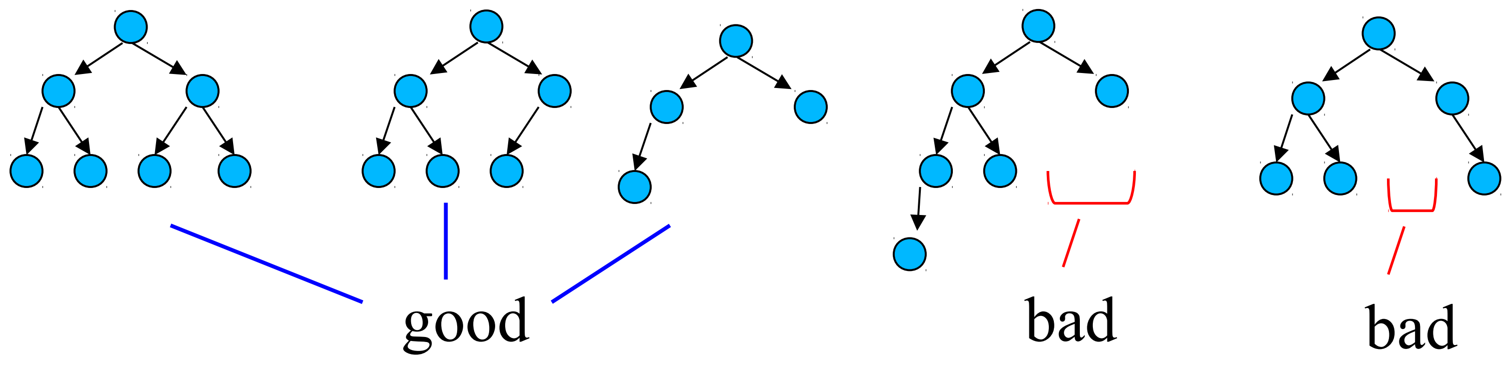
## Lecture 18:
## More Heaps
## A few practice questions for exam

## Review: Priority Queues

- **Each data item has a priority**
- **Add items to queue in any order**
- **Remove items in priority order**
  - **add A:5, B:3, C:6**
  - **remove C**
  - **add D:8**
  - **remove D, remove A**

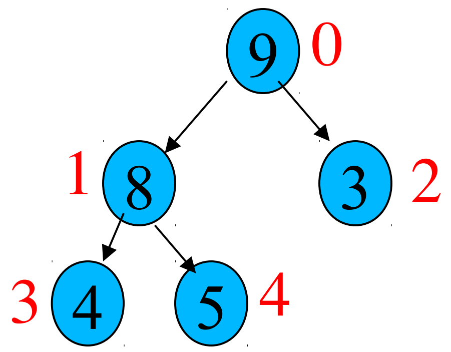
# Heap

- **A heap is a way to implement a priority queue with O(log n) complexity**
- **A heap is a complete binary tree**
  - **all levels except maybe the last are full**
  - **last level filled from left to right**



# Heap Representation

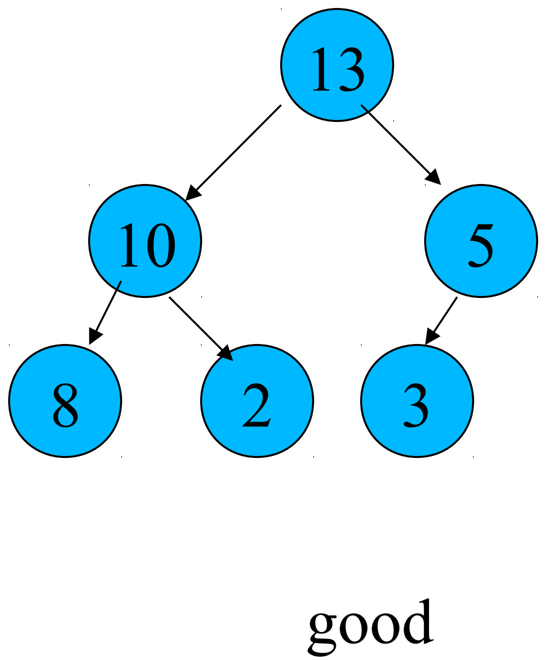
- Store heap in an array
  - For node at index j, children are at 2j+1 and 2j+2
  - Root at index **0**

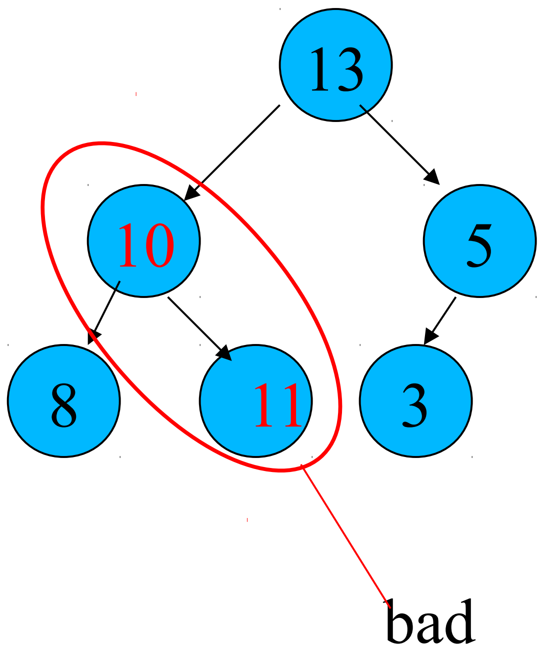


| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 9 | 8 | 3 | 4 | 5 |

## Heap

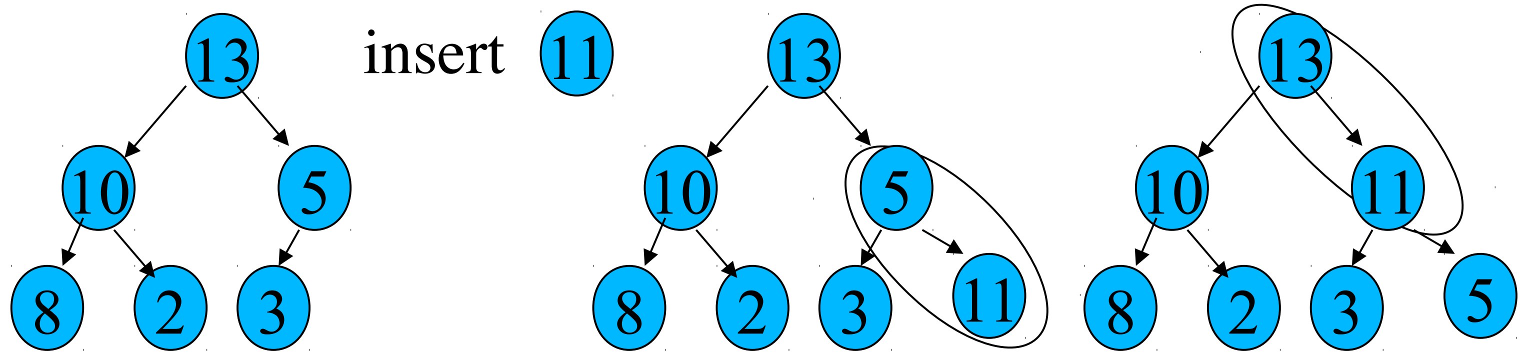
- **The number at a node is greater than the number at any descendant**

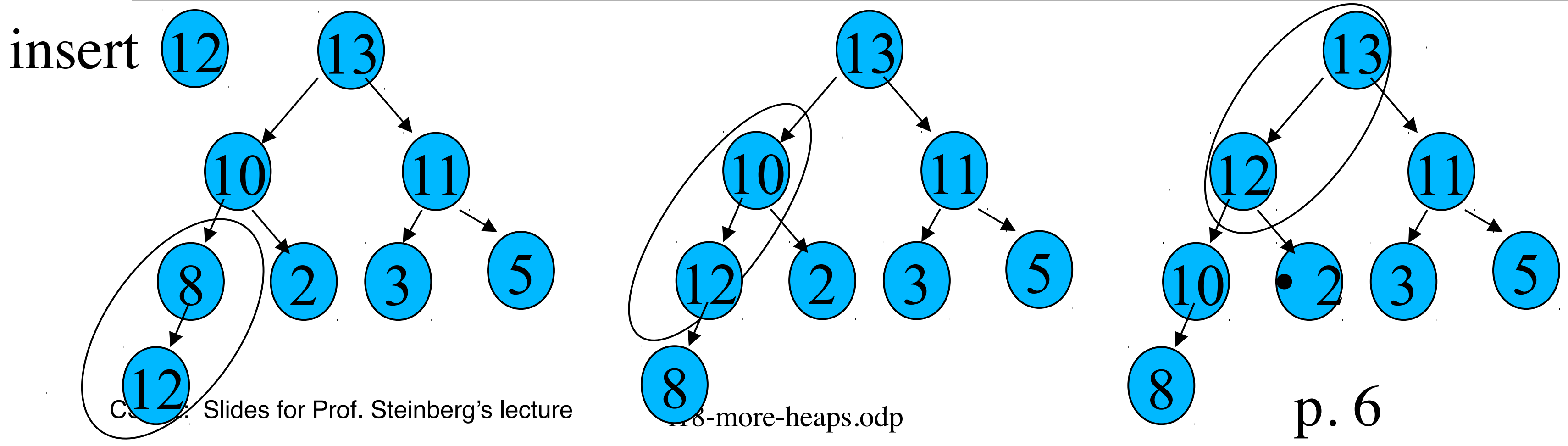




# Heap Insert

- **Add node at end of last level**
- **Move up restoring order (*filter up*)**



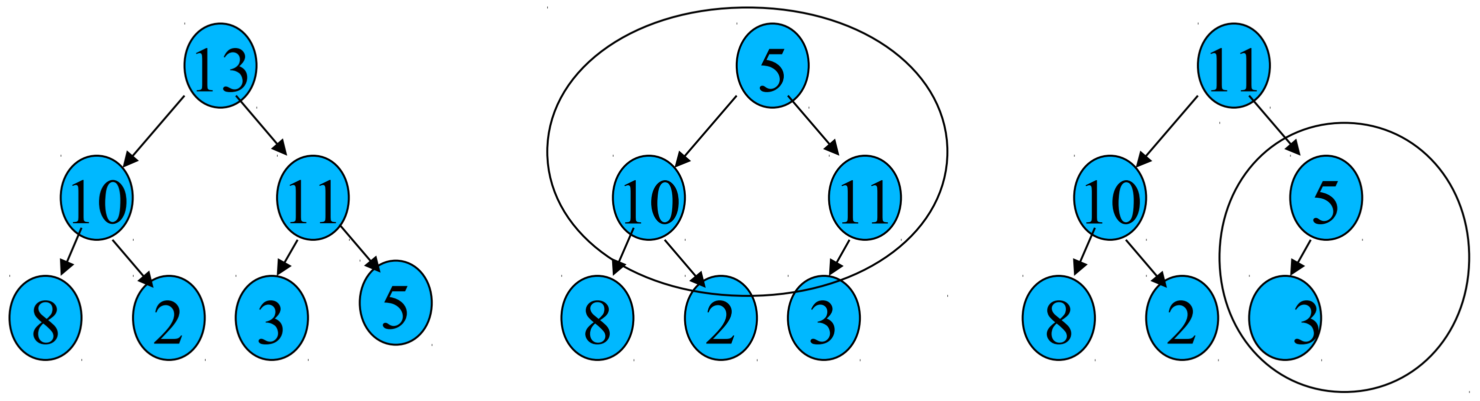


## Big O of insertion

- O(H) where H is height of while tree
  = O(log(n)) where n is number of nodes

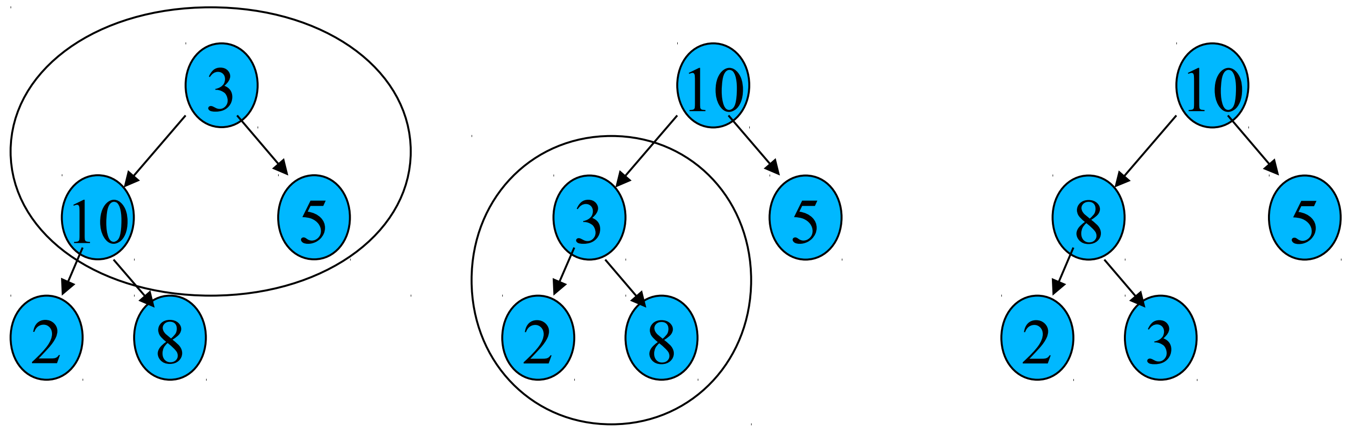
# Heap Deletion

- **Copy out data at root**
- **Delete last node on last row & put data in root**
- **Move down restoring order (*filter down*)**



## Heap Deletion

- Compare current node and two children
  - if current node largest, stop
  - if left child is largest swap current and left
  - similar if right child is largest



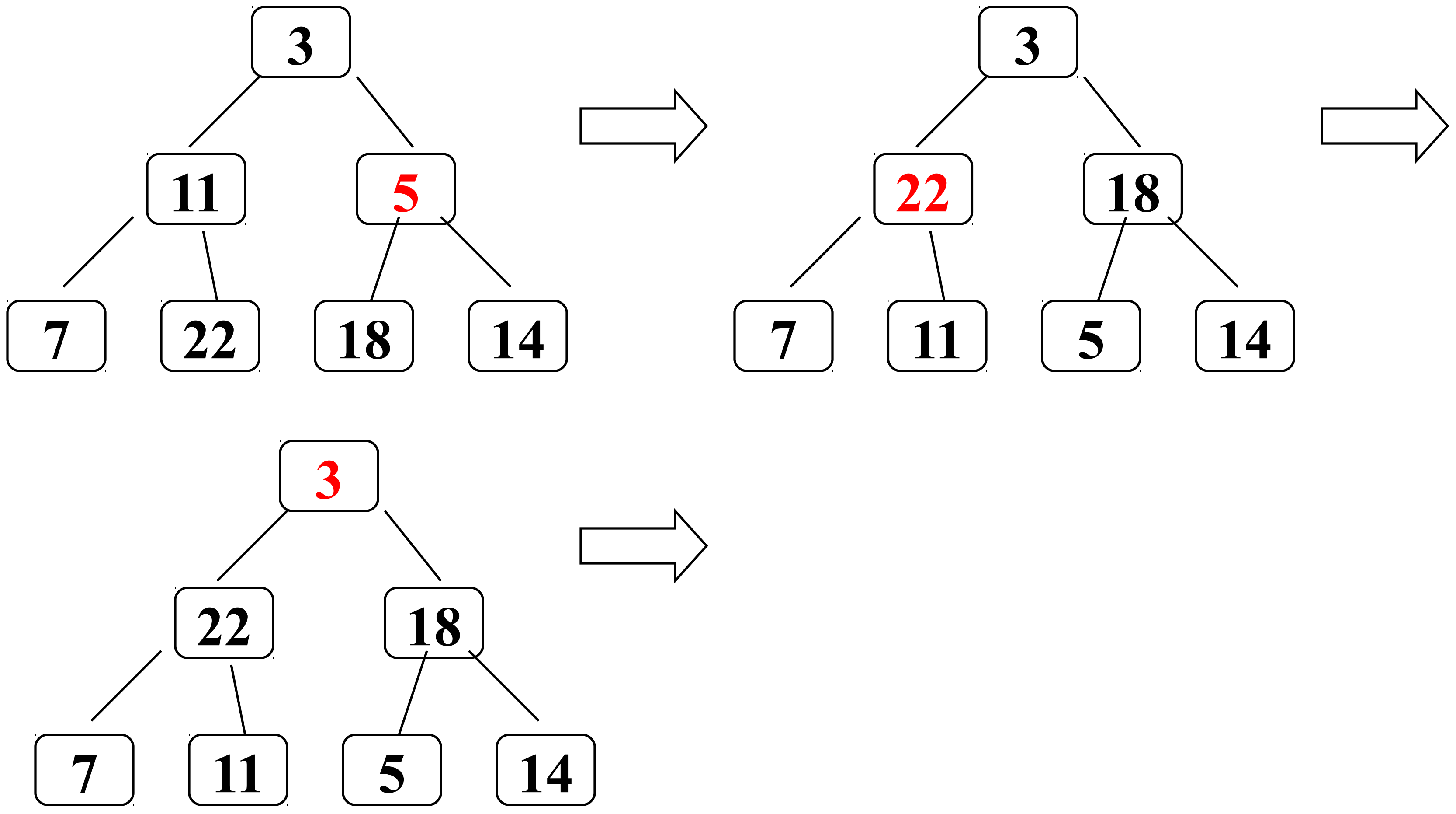
## Big O of deletion

- O(H) where H is height of whole tree
  = O(log(n)) where n is number of nodes

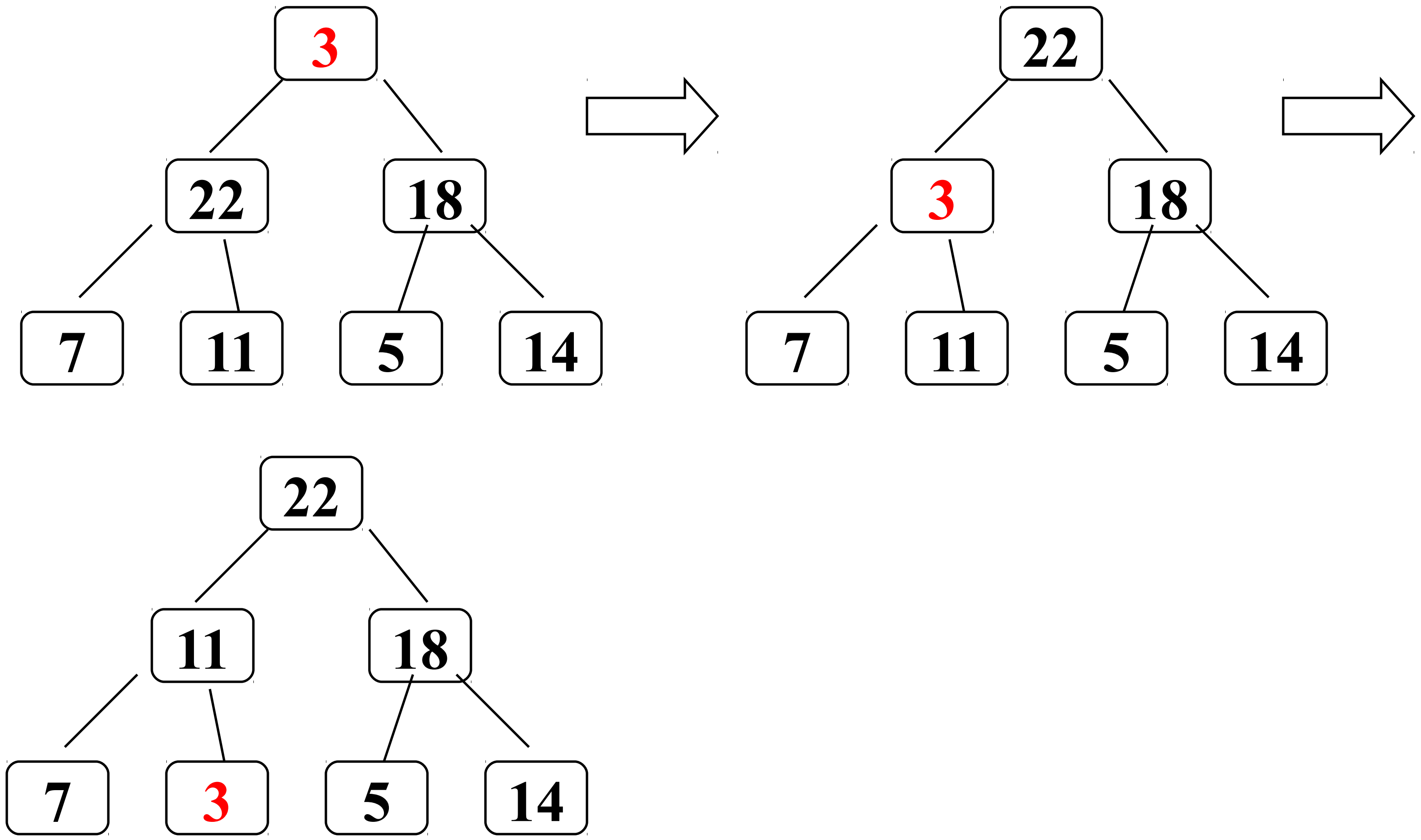
## Building a Heap from an Array

- Go from last non-leaf to index 0
  - At each node, do filter-down

# Building a Heap from an Array



# Building a Heap from an Array



## Building a Heap from an Array

- Work at a node is O(h) where h is height of subtree rooted at that node
- In a complete binary tree, majority of nodes close to bottom, so adds up to O(n)

## New:  Implementing a heap

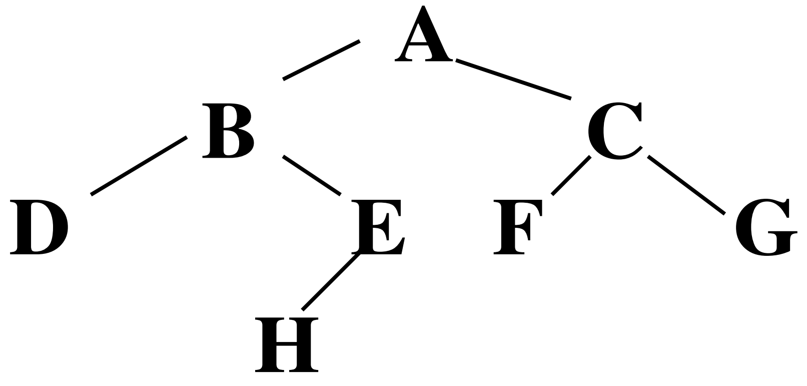
- See trees/Heap.java

# Heap Sort

- Insert all the data into the heap, one by one: n*O(log n) = O(n log n)
- Remove all the data from the heap, one by one: n*O(log n) = O(n log n)
- Total: O(n log n)

## Practice for exam

**For this tree, if we process the nodes in a pre-order traversal, in what order will the nodes be processed? Inorder?**



**Suppose you used an array to implement a hash table. Suppose each entry in the array is a hash bucket represented as a linked list, i.e. collisions are handled by chaining. Suppose the hash table is to represent a set of positive ints, and the hashing function is hash(n) = n mod 10 (ie remainder of n/10). Suppose we add the following numbers to the table:**

**100412 532 300 442 713**

**Assuming these numbers are representative of the data we will enter, have we chosen a good hash function? Why or why not?**

**Draw the hash table resulting from entering this data in the given order. Assume entries are added at the head of a list where possible.**

**100 412 532 300 442 713**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |  |  |

**Suppose we test each of the following numbers to see if it is in the table. (If a number is not in the table do not add it – table contents stay unchanged.) For each number, how many times do we compare it with a number in the hash table?**

**200**
**543**
**532**
**400**

**Suppose we had a BinarySearchTree made out of Nodes declared as follows:**

```
public class Node{
        public int data;
       public Node lst; // left sub tree
       public Node rst; // right sub tree
// …. etcetera}
```

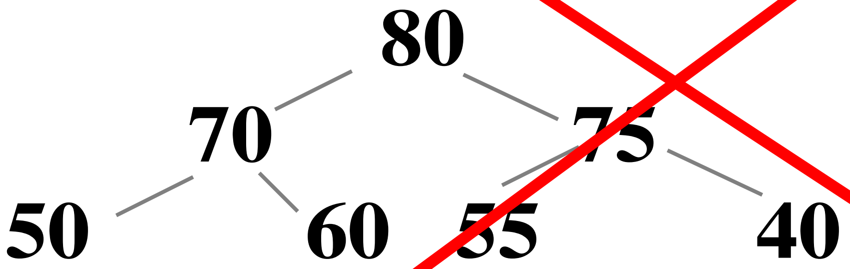
**Finish the following code to print out the data in a tree in numerical order.  It must be recursive.**

```
public static printData(Node root){


if (________________________){


System.out.println(root.data);
```

## Heaps not on exam 2 !!

~~For the MaxHeap below~~

~~What would be the first three numbers removed?~~

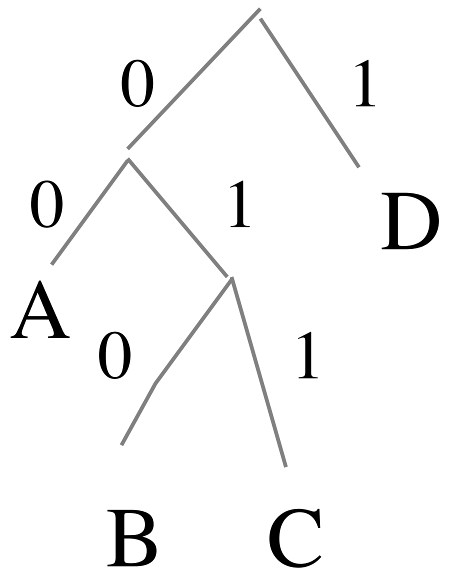


~~Redraw the heap as it would be after all three have been removed.~~

**For the Huffman code represented by the tree below,**

**What is the shortest code for any character?**

**What sequence of characters does 100100011001represent?**

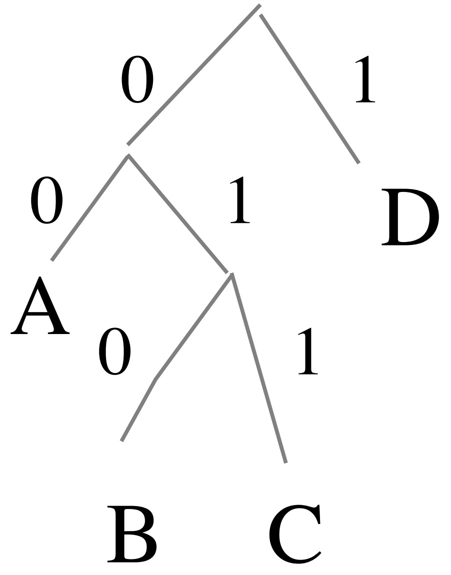


For the Huffman code represented by the tree below,

What is the shortest code for any character? 1 = D

What sequence of characters does 1|001|1|0001|1|001|1represent?

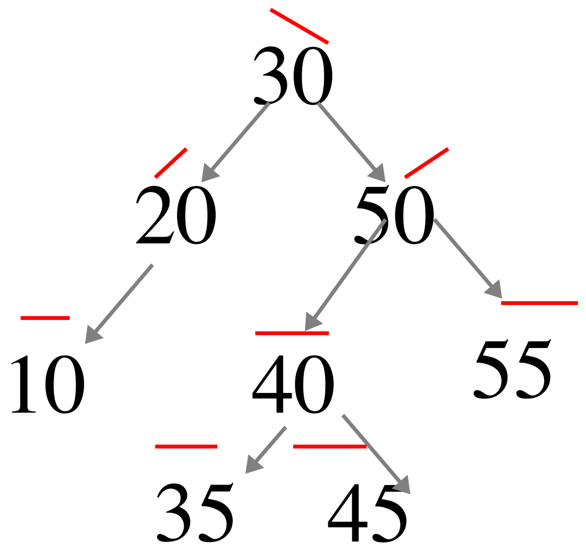
DADACAD

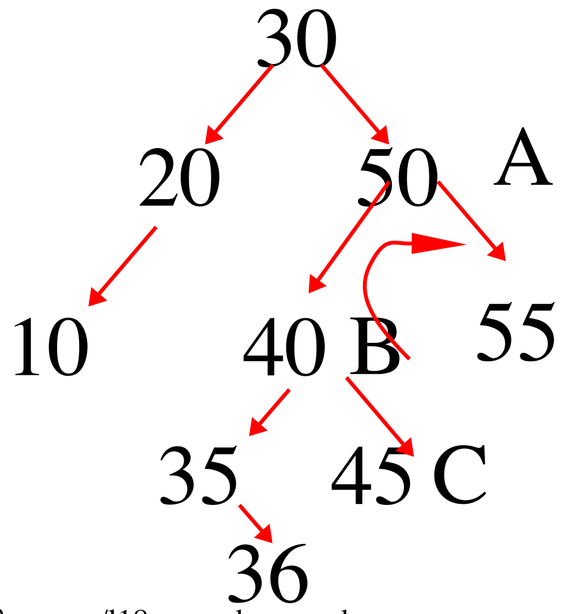


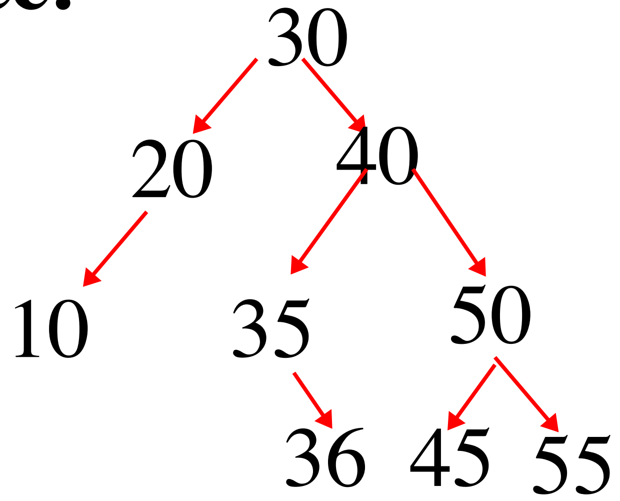
**Mark balance factors in the nodes of the following AVL tree. Then insert 36, and draw the resulting AVL tree.**

$$\begin{tikzcd}
 & & & 30 \arrow[dl] \arrow[dr] & & \\
 & & 20 \arrow[dl] & & 50 \arrow[dl] \arrow[dr] & \\
 & 10 & & 40 \arrow[dl] \arrow[dr] & & 55 \\
 & & 35 & & 45 &
\end{tikzcd}$$

## Mark balance factors in the nodes of the following AVL tree. Then insert 36, and draw the resulting AVL tree.


30
20
50
10
40
55
35
45


30
20
50
A
10
40 B
55
35
45 C
36


30
20
40
10
35
50
36
45
55