CS 180 - Tree stuff

Note Title 11/4/2011

Announcements

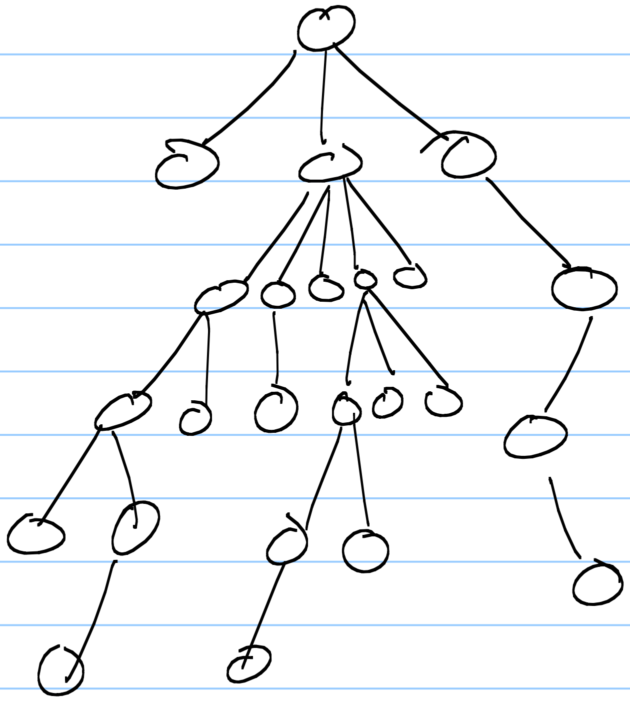
$\underrightarrow{\text{Last time}}$: Trees

Dfn: A tree T is a set of nodes storing elements in a parent-child relationship.

T has a special node r, called the root.
Each node (except r) has a unique parent.

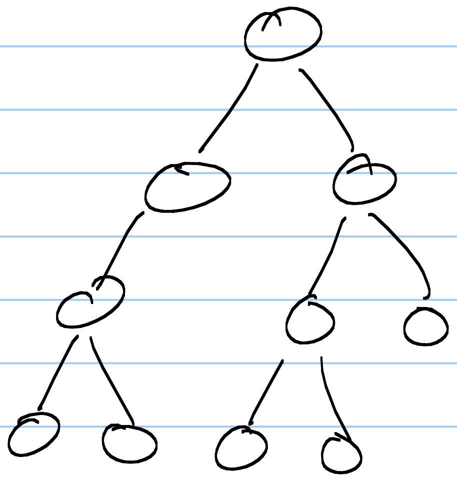
More dfns

- child
- siblings
- leaves
- internal nodes
- rooted subtree
- desendent / ancestor



Binary Tree

- Every node has ≤ 2 children.



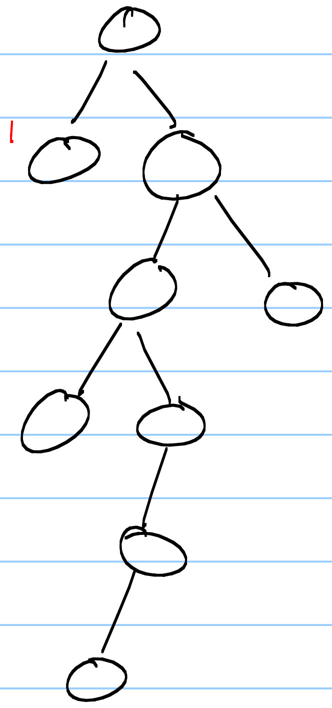
Depth + Height - defined recursively

depth : depth(r) = 0

depth(v) = depth(parent(v)) + 1

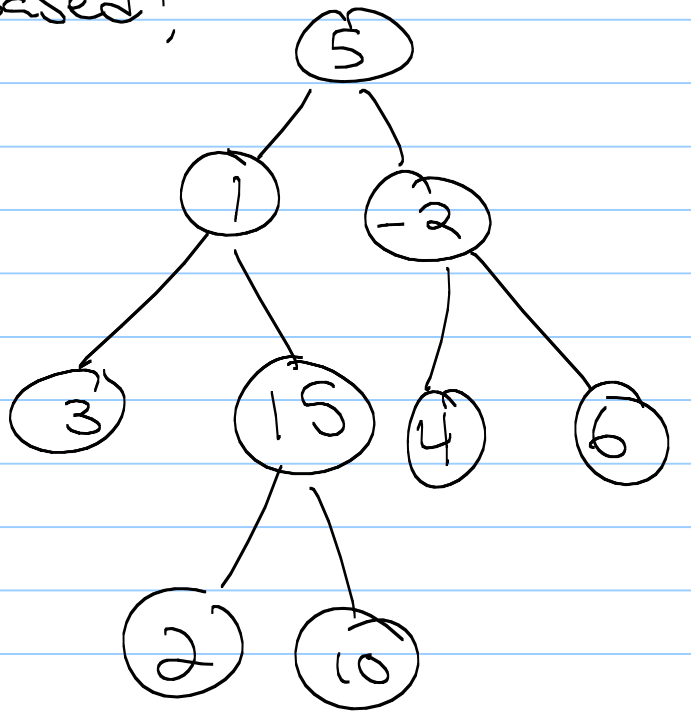
height : height(leaf) = 0

height(v) = max(height of children) + 1

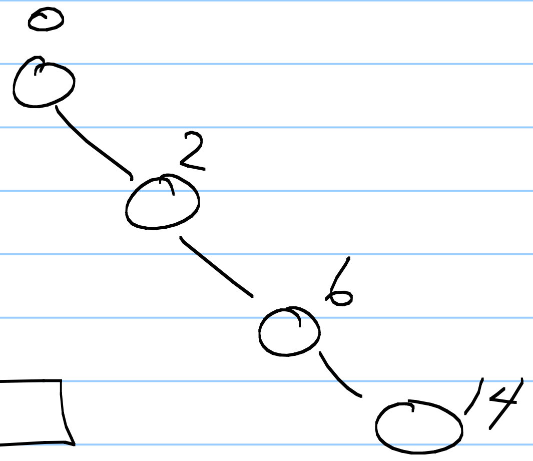


Nice trick

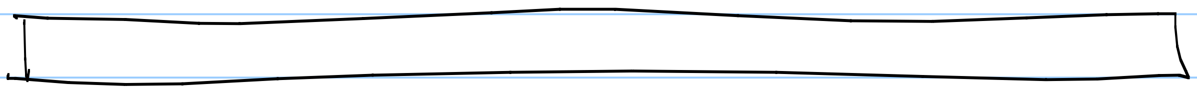
Can be pointers or array based!



Potential downside (of array)



Array:



How big?

Data Structure:

Priority Queue: supports the following operations

insert(e) : adds element e to the data structure

removeMax(): removes maximum element

getMax() : returns maximum element

How to build?

Why?

Good if you need limited sorting.

Ex:

How?

Maintaining with list or vector:

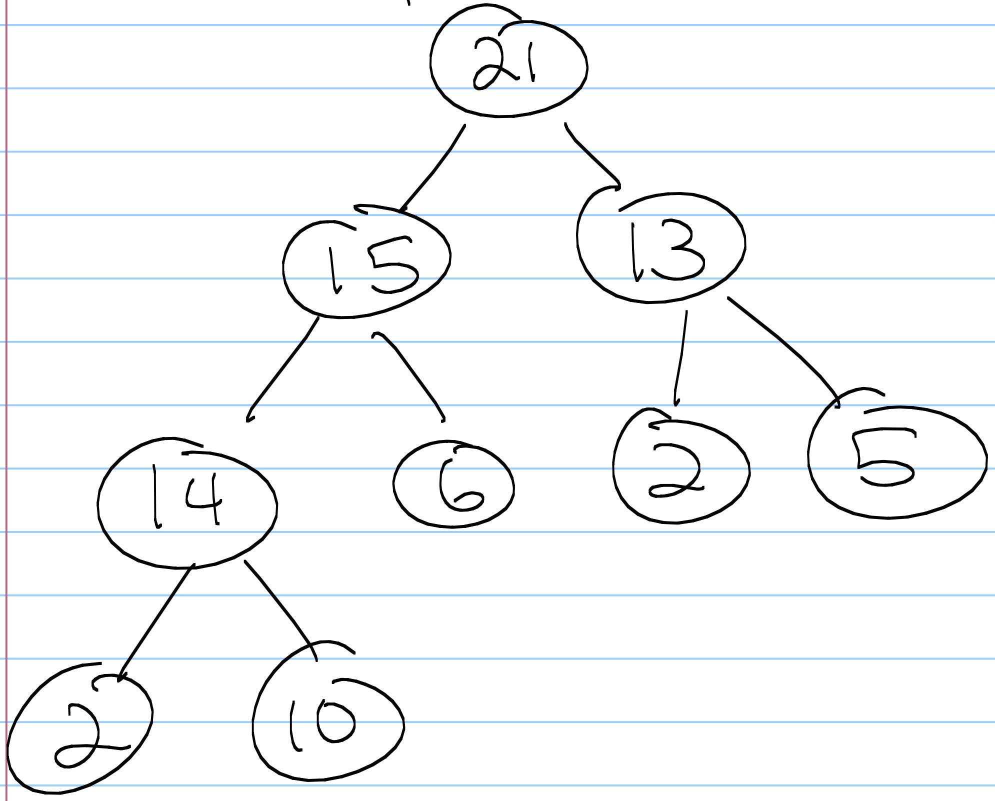
Heaps

A binary tree where:

- For every node v (other than root), the key stored at v is $\leq$ key stored at v's parent

- The tree is complete: levels 0 to h-1 are full, and level h is filled in left to right order

Max Heap

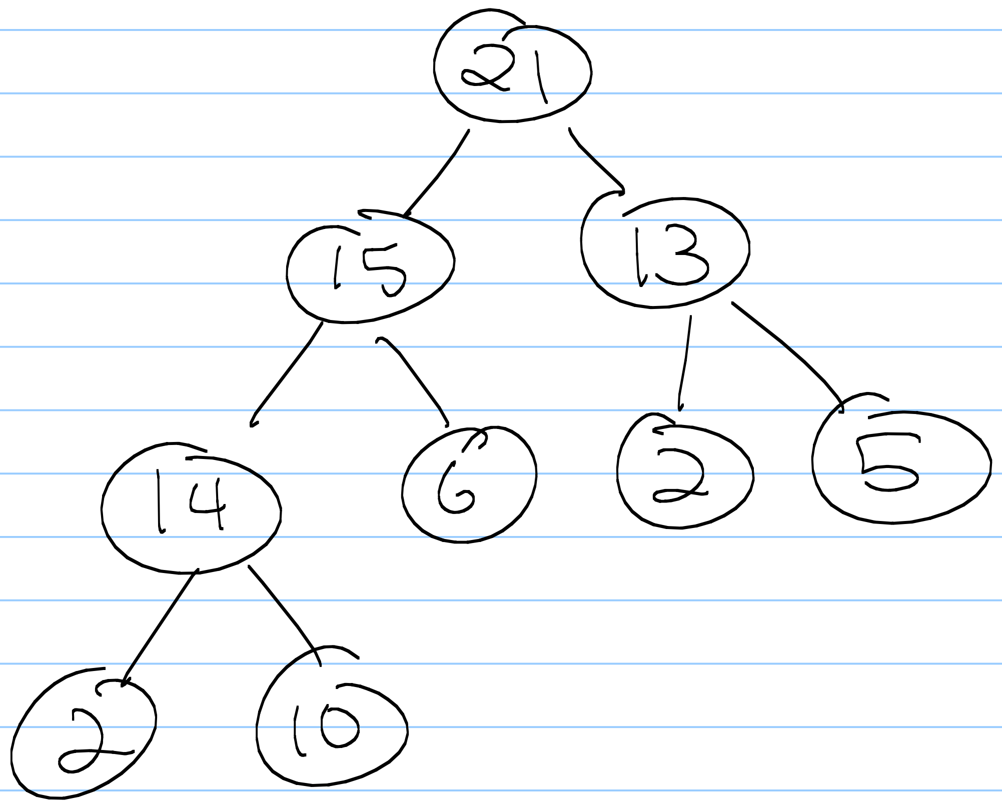


Insert

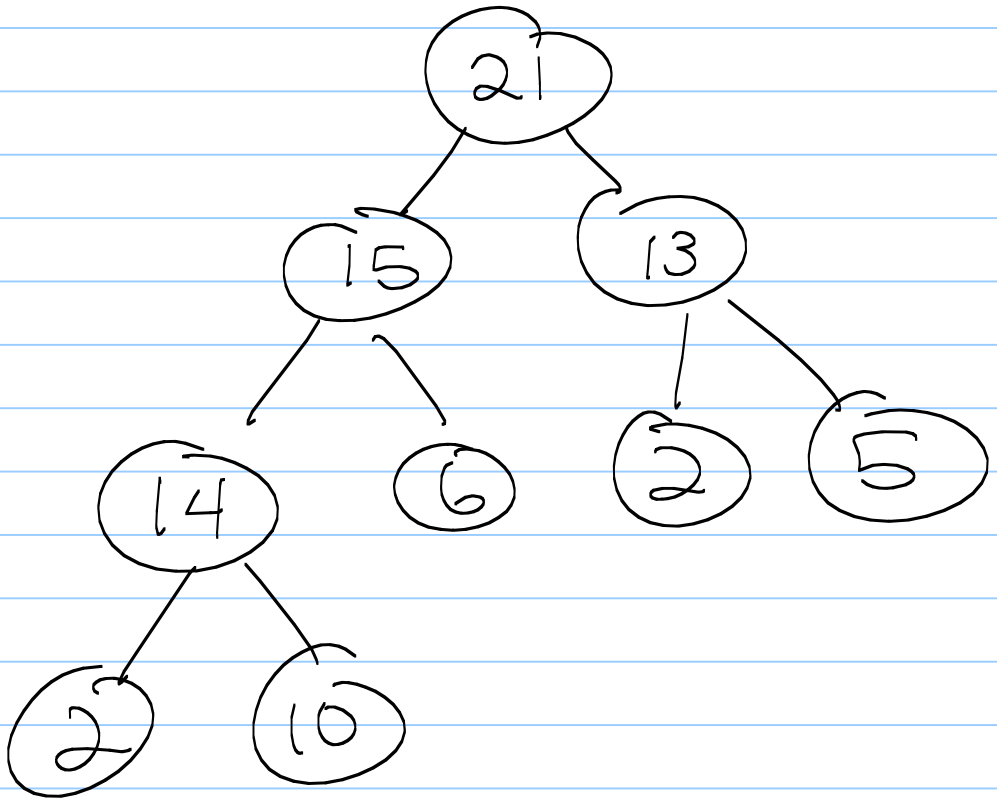
insert(2)

insert(52)

insert(7)



Remove



Running times

How many comparisons / swaps?

Code for this class

- Array-Based. Why?