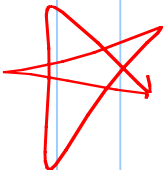


CS 180 - Heaps

Note Title

1/14/2011

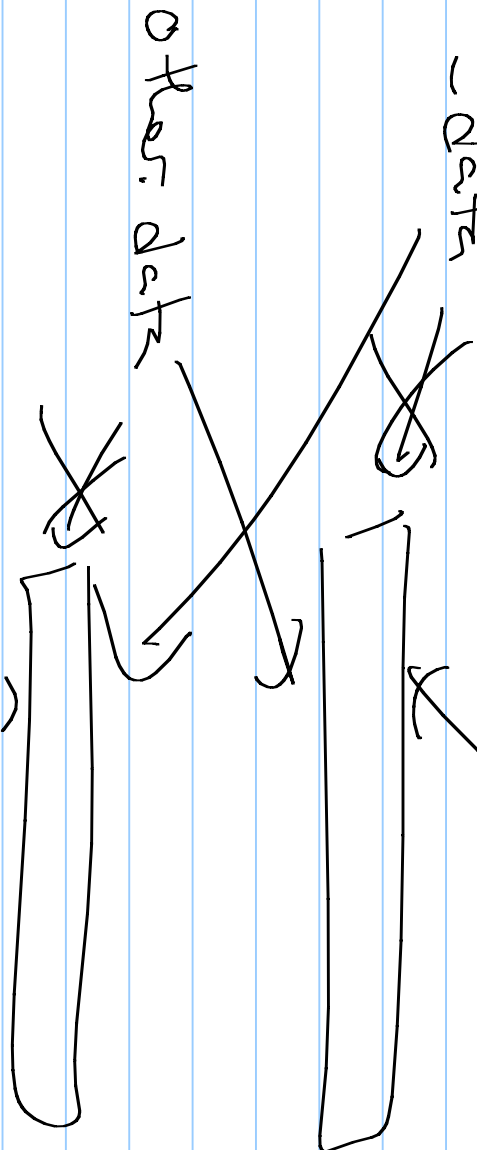
Announcements

- Midterms graded:
may redo lowest problem
(submit by Friday) 
- HW due tonight
- No lab tomorrow - lecture instead
- Office hours: tomorrow 11-12
- Boing Scholarships (104 R, Her)

Swap:

-data

temp



other data

Last time: Trees

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

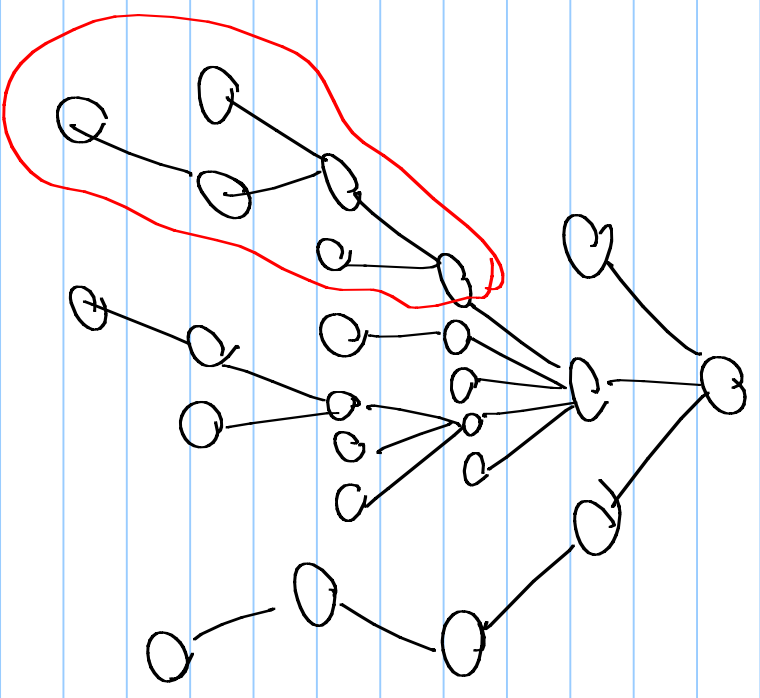
in CS

T has a special node r , called the root.

Each node (except r) has a unique parent.

More defs

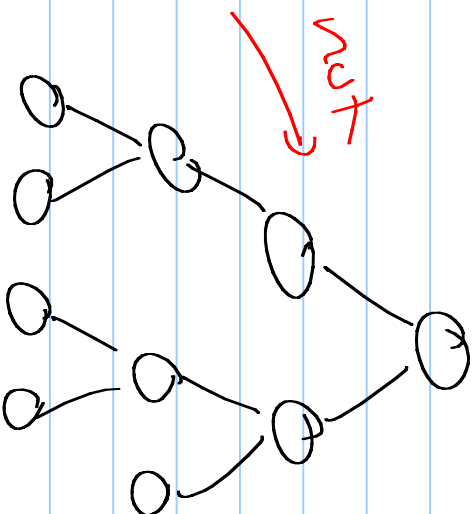
- child
- siblings
- leaves
- internal nodes
- rooted subtree
- descendant / ancestor



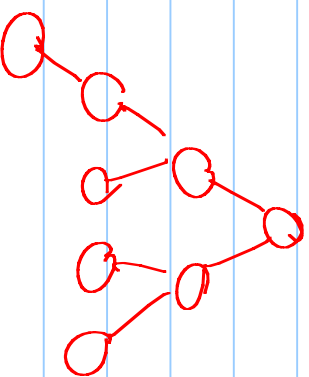
Binary Tree

- Every node has ≤ 2 children.

Full: exactly 2 or 0 children



Complete: all levels filled



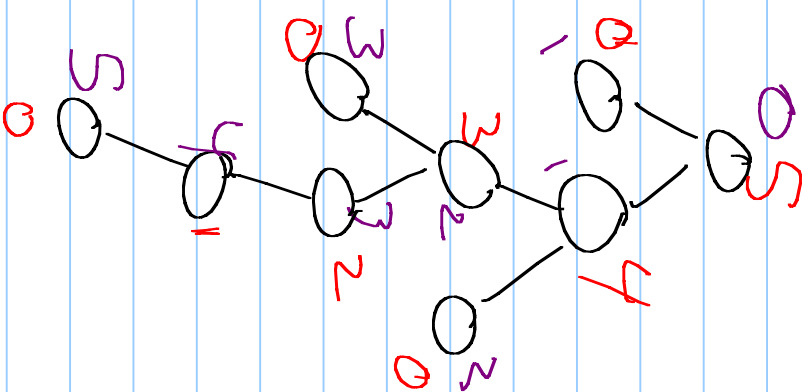
Depth & Height - defined recursively

~~depth~~ : $\text{depth}(r) = 0$

$\text{depth}(v) = \text{depth}(\text{parent}(v)) + 1$

~~height~~ : $\text{height}(\text{leaf}) = 0$

$\text{height}(v) = \max(\text{height of children}) + 1$



Nice trick

pointer : Node:

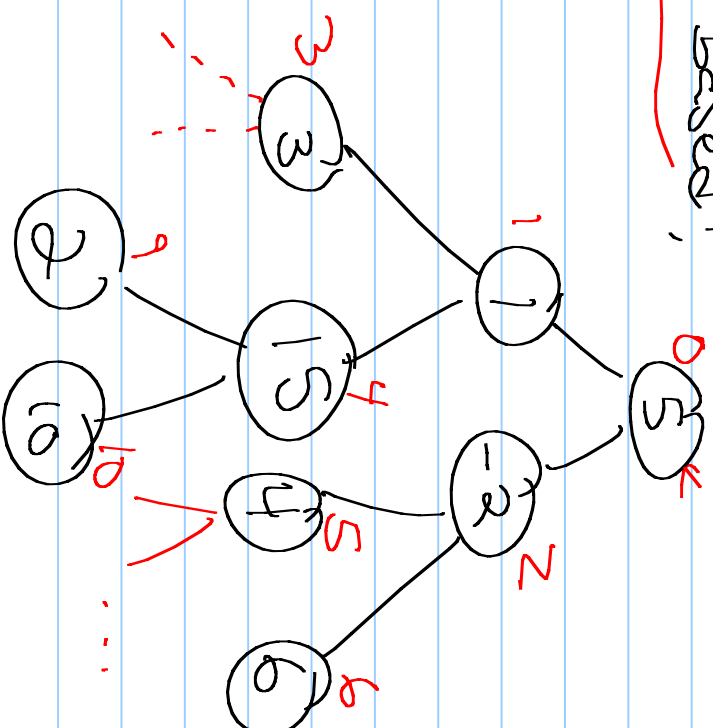
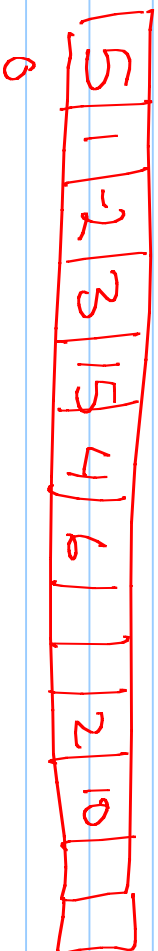
- data
- left
- right
- parent

Can be pointers or array based!

$$\text{left}(v) = 2 \times v + 1$$

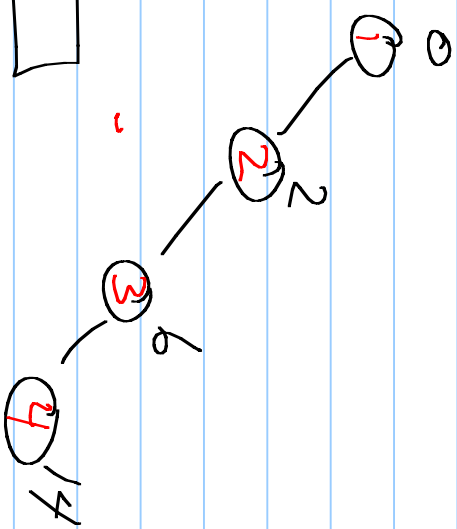
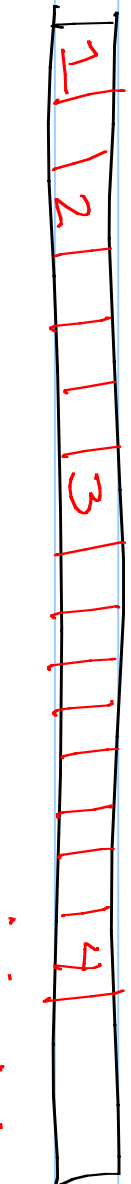
$$\text{right}(v) = 2(v+1)$$

parent?



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 ^{sortings.}

Ex: repeated access to highest weight item in structure

How?

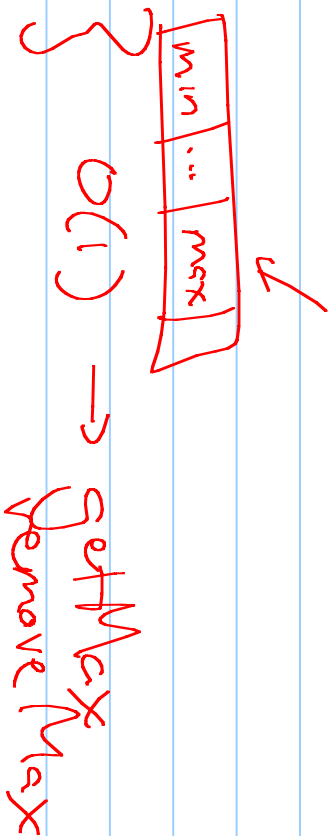
Maintaining with list or vector:

Vector: [1|2|]

2, 1.5

Sort vector

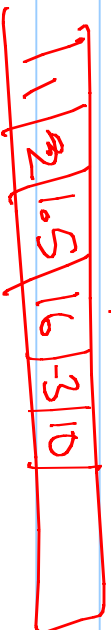
max is at
end



insert: $O(n)$

Another (vector)

don't sort



max = 3

O(1): insert: put at end

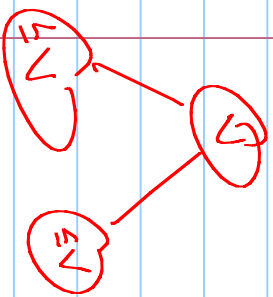
O(n): remove Max: if bigger than current max, update
remove max from vector
find new max

O(1): get Max:

Heaps

A binary tree where:

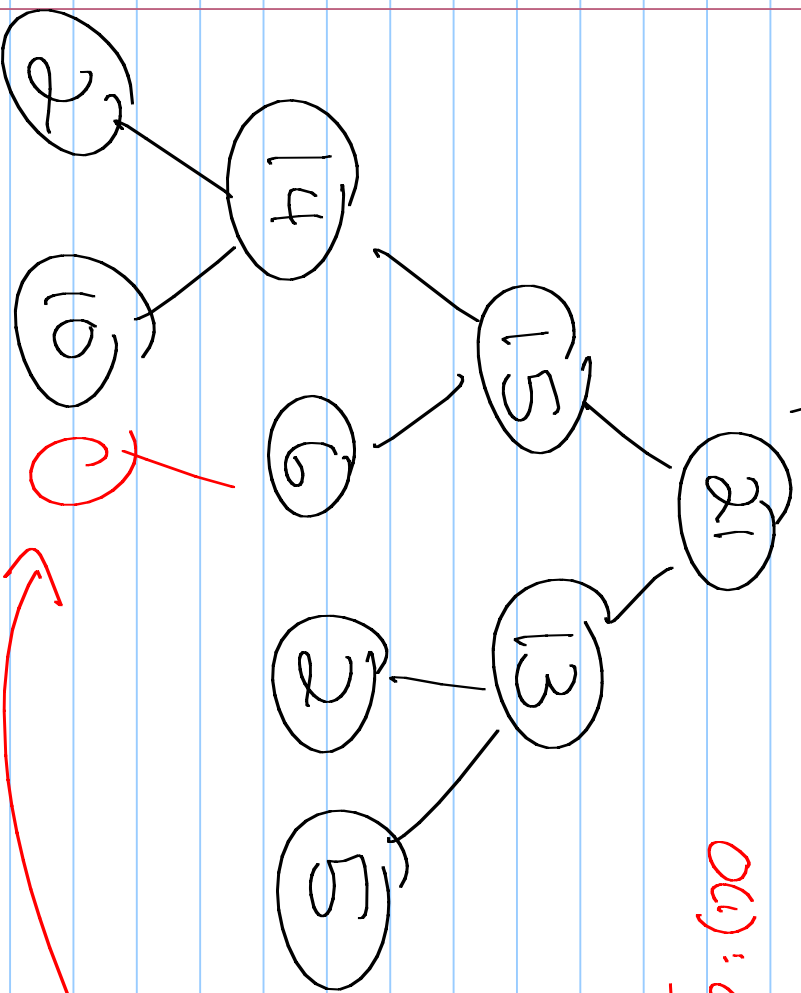
- For every node v (other than root),
the key stored at v is \leq key
stored at v 's parent



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

Max Heap : not a bst

o(1) : getMax : return root



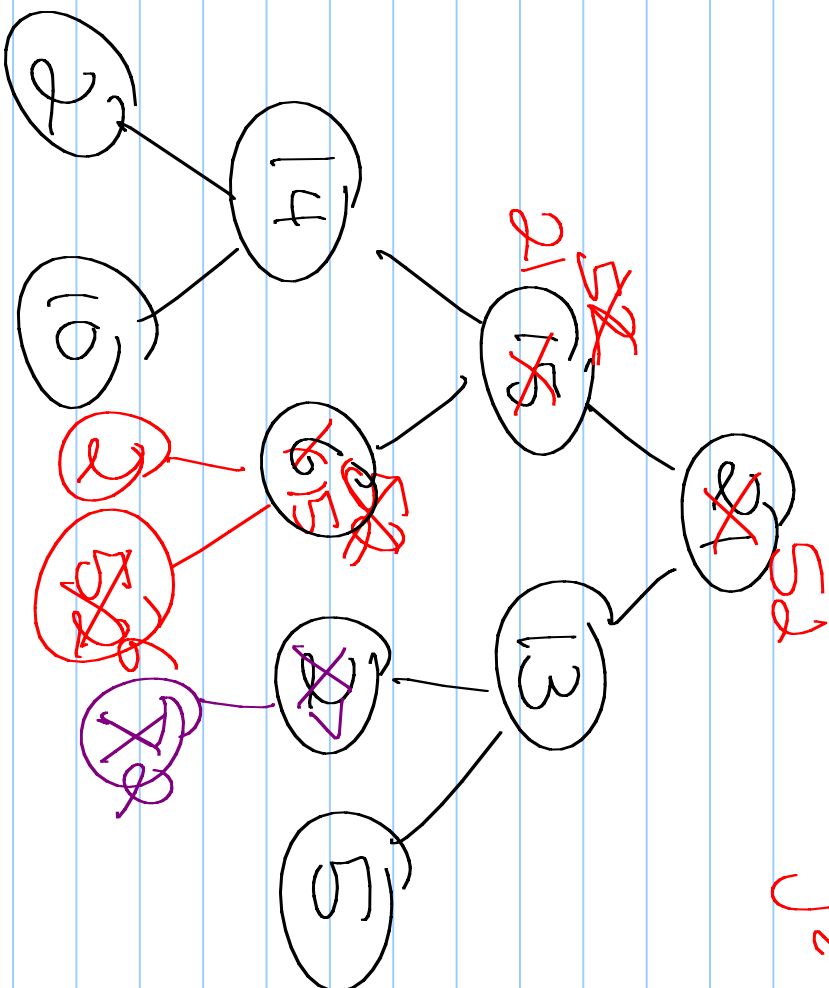
insert :

Insert

insert (2) ✓

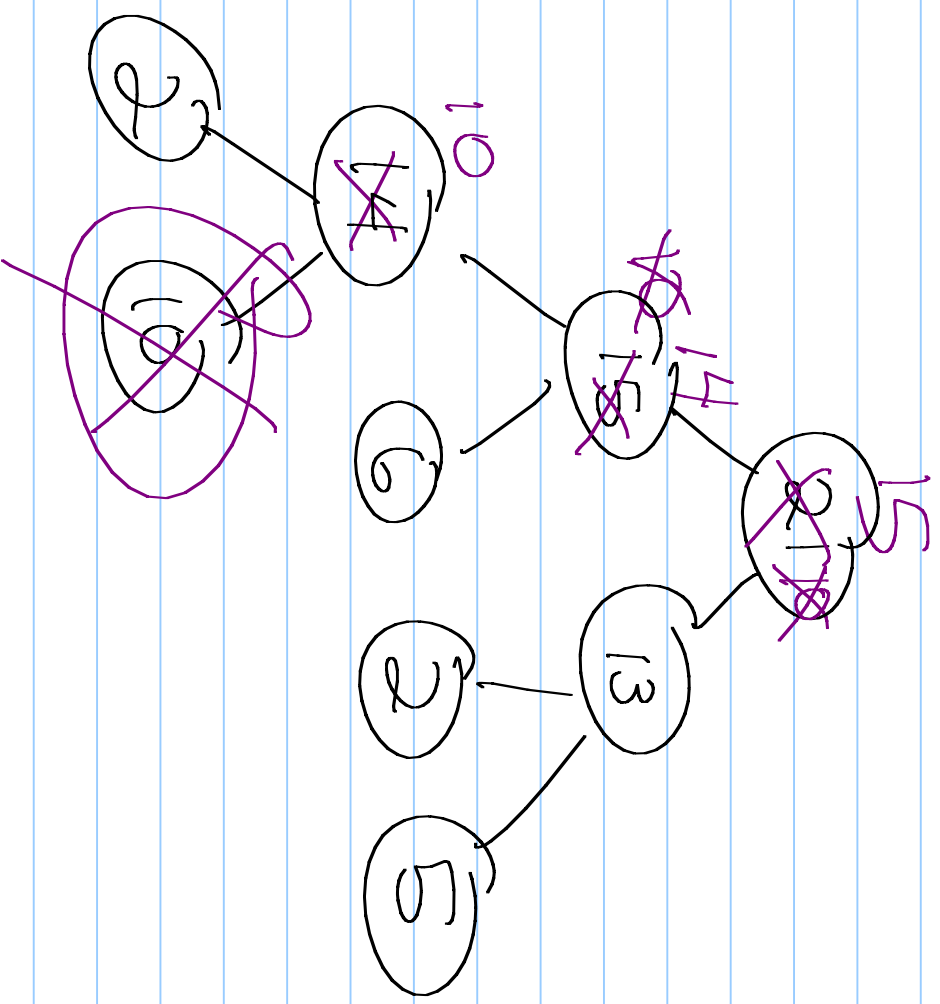
insert (52)

insert (7)



$$\sum_{i=0}^n 2^i = 2^{n+1} - 1 = n \approx 2 \log_2 n$$

Remove



Running times

How many comparisons/swaps?

next time

Code for this class

- Array - Based. Why?

