Announcements

- HW is up
- Turn in exam redo
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

A binary tree where we maintain 2 invariants:
- Tree is complete.
- Any node's value is ≤ its parent's value.

Runtimes: $O(\log n)$ for insert, delete
$O(1)$ - get Max

(Code is on webpage.)
Complete trees:

- Each level has $2^i$ nodes

$$n = \sum_{i=0}^{\text{depth} - 1} 2^i = 2^{d+1} - 1$$

Depth?

$$n = 2^{d+1} - 1$$

$$n+1 \approx 2^{d+1}$$

$$\Rightarrow \log_2(n+1) = d + 1$$

$d = \lceil \log_2 n \rceil$
**Binary Search Trees**

A binary tree where we maintain the following:

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

$y \leq x$

$z > x$
Example:

Sorted order: inorder

\[13, 16, 27, 30, 32, 54\]

Go left
Print myself
Go right
Insert: unique location in current tree where new value could be added

insert (20)
insert (5)
insert (93)
Find: traverse tree, comparing left if smaller, right if greater.

find(62)
Delete:

More complex!

del (16)
del (36)
del (16)
default(16)
candidates to replace:
neighbor from inorder traversal
Note: BSTs are not unique!

Can you make another BST with these elements?
Runtimes:
Find: $O(\text{depth tree})$
Insert: 
Delete: 
\[ \text{depth} = O(n) \]
Code

- Will be pointer based. Why?
  (Could be unbalanced)

(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.)

AVL.h will balance binary search trees.