CS180 - Binary Search trees

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

- `get Max()` : return element with maximum key (its `e`)

- `remove Max()` : delete element with maximum key

With lists or vectors, some operation will take \( \Theta(n) \) 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"

Remove Max()

2

36

11

27

3
**Code for heaps**

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

**Running times?**

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

$h \leq 2^i = n \Rightarrow 2^h - 1 = n$

level $i$: $2^i$ nodes

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

A binary tree where we maintain the following:

The value at any node is ≥ 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: $O(n)$ (because in worst tree, $h = n$)
Insert: $O(n)$
Delete: $O(n)$
Code
- Will be pointer-based. Why?
  not complete, so array may waste space

(Need nodes, iterators, etc.)
Tree traversals

Inorder Print(n):
- If n != NULL
  - Inorder Print(n->left)
  - Print n
  - Inorder Print(n->right)

Inorder traversal: 2, 12, 16, 32, 55, 82
\[(16 - (12 \div 3)) + 45\]
Pre order + post order

↓

print n

Preorder (n→left)

Preorder (n→right)

post order (n→left)

post order (n→right)

print n

examples next time