CS180 - Binary Search trees

Announcements

- HW due next Tuesday
- Next HW will be up today due 1 week from Monday
- Fun with trees for ~2 weeks
Last time: Priority Queues

- insert(e): add e to our data structure
- get Max(): return element with maximum key (it's e)
- remove Max(): delete element with maximum key

With vectors or lists: $O(n)$ (for one function)
Last time: Heaps

**Complete**

A binary tree where we maintain an invariant:

- Any node's value is $\leq$ its parent's value.
- Complete binary tree.

So where is maximum value?

Result: $O(\log n)$

(code is on webpage.)
Inserting + Deleting

Idea: Maintain structural property at all times.

Then bubble up or down to fix the orderings.

Runtime: $O(\log n)$
Tree Traversals: method of visiting every node

Inorder Traversal: recursive procedure:

At node v:
- recursively go left
- print v
- recursively go right

LRXBMQZYN
A use for in-order: precedence of operations

\[
(16 - (12 ÷ 3)) + 45
\]
Preorder Traversal

At node \( v \):
- print \( v \)
- recurse left
- recurse right

M X L R B Y Z Q N
Preorder (good) example:

Title
  Abstract
    Ch. 1
      1.1
      1.2
  Ch. 2
    2.1
    2.2
    2.3
  References

Title
Abstract
Ch. 1
  1.1
  1.2
Ch. 2
  2.1
  2.2
Post order Traversal

at v:
  - recurse left
  - recurse right
  - print v

RLB XQ & NYM
Post-order use: Calculating file system sizes.

```
/ usr / echambe5
  5k of files
    / cs180
      1k
        / fall11
          1k
        / fall12
          1k
      / cs341
        6k
          / examples
            20k
          / HW
            80k
```
Binary Search Trees

A binary tree where we maintain the following:

The value at any node is \( \geq \) its left child and \( < \) its right child.

\[
y \leq x \quad \Rightarrow \
x > y
\]
Example:

- User can't directly modify the tree.
- Not balanced
**Insert**: exactly 1 place a value can go

- insert (20)
- insert (5)
- insert (93)

Essentially:
- run find
- put it there
Find
Start at root
do binary search
Delete:

More complex.

delete (19)

delete (16)

delete (20)
Note: BSTs are not unique!

Can you make another BST with these elements?
Runtimes:

Find: \( O(\text{height}) = O(n) \) \( \Rightarrow \) worst case

Insert: \( O(n) \)

Delete: \( \Omega(n) \) \( \subset \) Find

We'll fix this later.
Code

- Will be pointer based. Why?
  not complete trees

(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.)