LEC 14

CSE 123

Modifying Binary Trees

Questions during Class?
Raise hand or send here

sli.do #cse123

BEFORE WE START

Talk to your neighbors:

What was your favorite event in the Winter Olympics?

Respond on sli.do!

Instructor: Brett Wortzman

TAs:
Arohan
Ashar
Sean
Chloe
Jenny
Nate
Jonah
Brice
Chris
Elden
Ishita
Kuhu
Kavya
Misha
Kieran
Sahana
Anirudh
Crystal
Eeshani
Aidan
Cora
Dixon
Nhan
Trien
Evan
Rena
Katharine
Anya

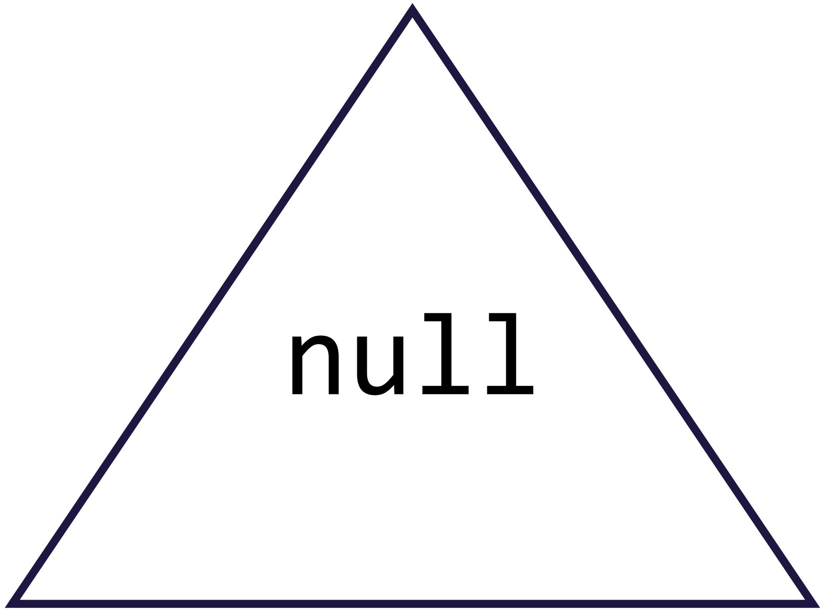
Now playing: 🎶 CSE 123 26wi Lecture Tunes 🎶

Announcements

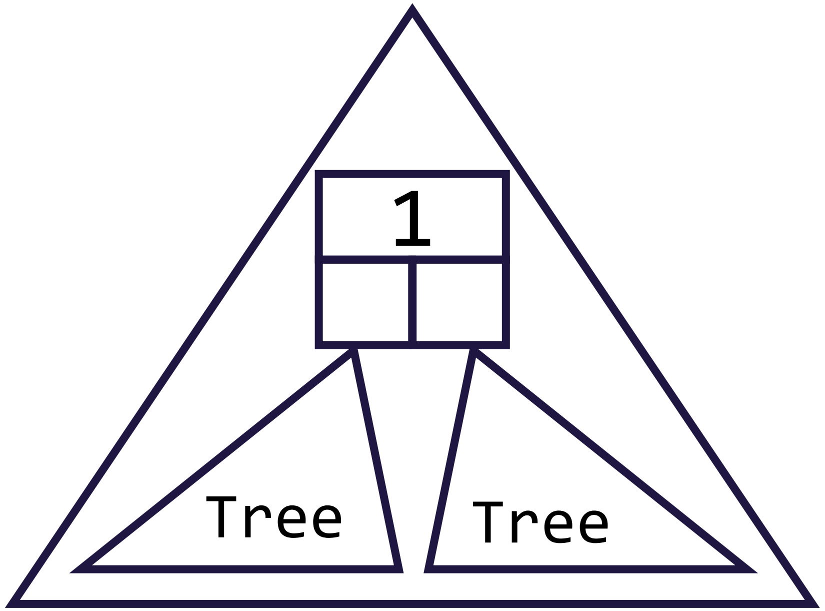
- Programming Assignment 2 due tonight at 11:59pm
- *Programming Assignment* 3 out tomorrow
  - due next Friday(!) (3/6)
- Quiz 2 next Tuesday (3/3)
  - Practice quizzes out soon; solutions on Saturday
- R-Brett added, due 3/15
  - R7, R-Bonus also due 3/15
- Creative Project 2 and Resub 4 feedback out today
- Quiz 1 feedback out today or tomorrow

Review: Binary Trees

- A Binary Tree is either:



Empty tree



Node w/ two subtrees

This is a recursive definition!
A tree is either empty or a node with two more trees!

Review: Binary Tree Programming

- Programs look very similar to Recursive LinkedList!

- Guaranteed base case: empty tree
  - Simplest possible input, should immediately know the return
- Guaranteed public / private pair
  - Need to know which subtree you're currently processing
- If modifying, we use `x = change(x)`
  - Don't stop early, return updated subtree (`IntTreeNode`)

Review: Binary Tree Traversals

- 3 different primary traversals
  - All concerned with when you process your current root

- Pre-order traversal:
  - Process **root**, left subtree, right subtree
- In-order traversal:
  - Process left subtree, **root**, right subtree
- Post-order traversal:
  - Process left subtree, right subtree, **root**

Sometimes different traversals yield different results

Modifying Binary Trees

- Like linked lists, cannot modify nodes
  - Because data field is `final` (there are good reasons for this)
- Will need to create and insert new nodes
- Use `x = change(x)`, usually ***3 times***
  - overall root (in public method)
  - left subtree
  - right subtree
- Order might matter!
  - Does operation on root depend on children?