CSCI 2100

Heap Recap
Binary Trees
Search Trees
Recap

- HW due today
- Next HW posted after class, due in 1 week (pen + paper)
- Lab tomorrow
Last "time":

- Trees: binary tree (height + depth)

- Priority Queues: ADT supporting:
  - insert
  - get Max
  - remove Max

- Heaps:
  Binary tree implementation of PQ
  Value at node is ≥ children's value

(all operations are O(log n))
Heap example:

Insert: 8, 11, 3, 4, 10, 1, 12, 9 (in this order!)

Then:
1. remove Max()
2. remove Max()
Aside: Tree traversals

3 ways to traverse a tree:

Starting at root:

Preorder $(v) =$

visit

Preorder $(v \rightarrow \text{left})$

Preorder $(v \rightarrow \text{right})$

traversal order: root, usr, echamber5, csci2100, csci3200, letsher, bin, ls, cd,
In order(v) =
Inorder(v → left), visit v, Inorder(v → right)

Ex:

traversal: 5, *, 7, -, 2, +, 6, /, 3

"middle"
Post order \( (v) = \)
\[
\text{post order (v \to left)}
\]
\[
\text{post order (v \to right)}
\]
\[
\text{visit v}
\]

Ex:

\[
\text{traversal} : 4, 6, 9, 5, 17, 14, 19, 11
\]
Next: 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.

Ex:
Finding in a BST:

- Check if root = target value
  - return true
- If root > target
  - If left ≠ NULL
    - else return false
  - else recurse on left child
- If root < target
  - else recurse on right child

Ex:
find (58)
find (50)

if hit a leaf return false
Inserting!

Given a BST, insert is done by finding the (unique) leaf location where the value fits.

Ex: Insert (4), Insert (27), Insert (15)

Find, but when hit a null insert the new node
Delete: More Complex

remove (16): delete 20's node
remove (27): copy 30 up to root
Cases:

- Target could be a leaf

- Target could have only 1 child: remove (15)
Cases (cont):

- Target could have 2 children

removing (20):

```
   20
   /  \
  10   30
  /   /  \
  5   15 25
   \   /  \
    27 35
```
Remove(x):
  Find(x)

If x is a leaf
  delete x's node
else if x has one child
  can remove x's node
  and "promote" it's only child to its spot

Else:
  Find smallest value > x
  (Note: this is next value in an inorder traversal!)
  Copy that value to x's spot
  & remove that node, promoting its right child in its spot (if it has one)
  Note: it will have no left child
Code:

- Pointer based.

Reason: we need to move around entire subtrees.

Also, tree is not complete!

Valid BST: 

- not space efficient
Consider

Can you make another BST with these values?