CS 180 - More Trees

Note Title 10/30/2009

Announcements

- Next program is out
- Test the week after next

<u>Trees</u>: Traversals

How to display or check information stored in a tree?

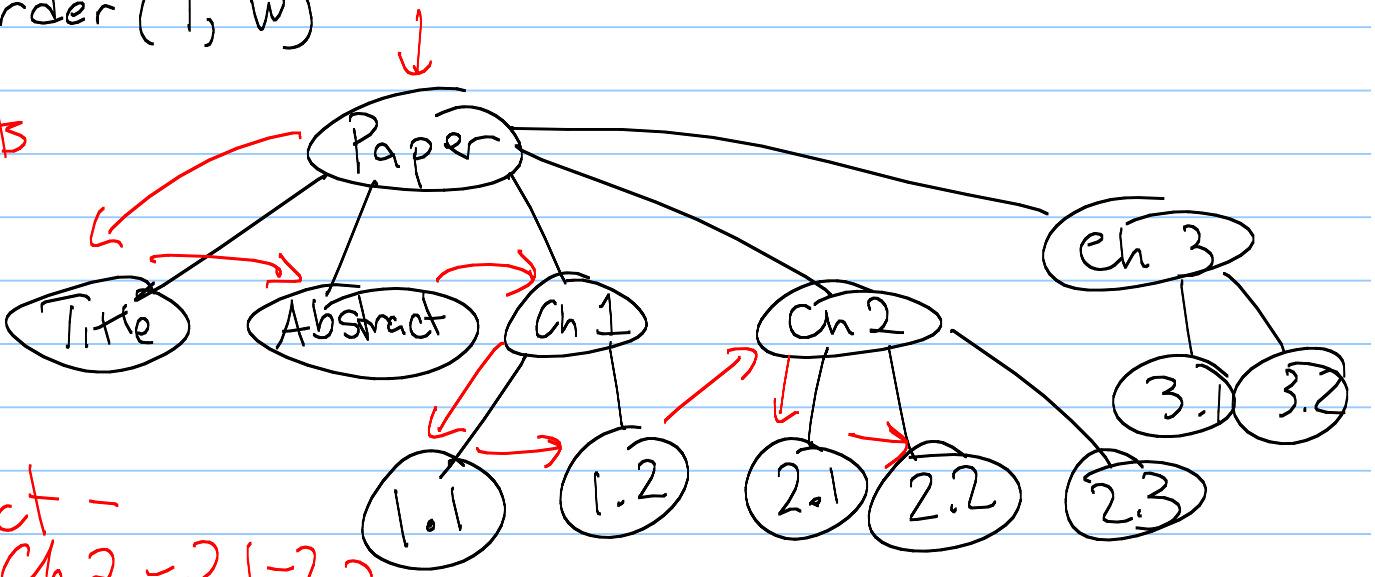
Different ways depending on what is stored in it.

- Inorder
- Preorder
- Postorder

```
Preorder(T, v):
    Perform "visit" at v
    for each child w of v:
        Preorder(T, w)
```

Also
Ex: depth in a tree

Ex: Print Contents



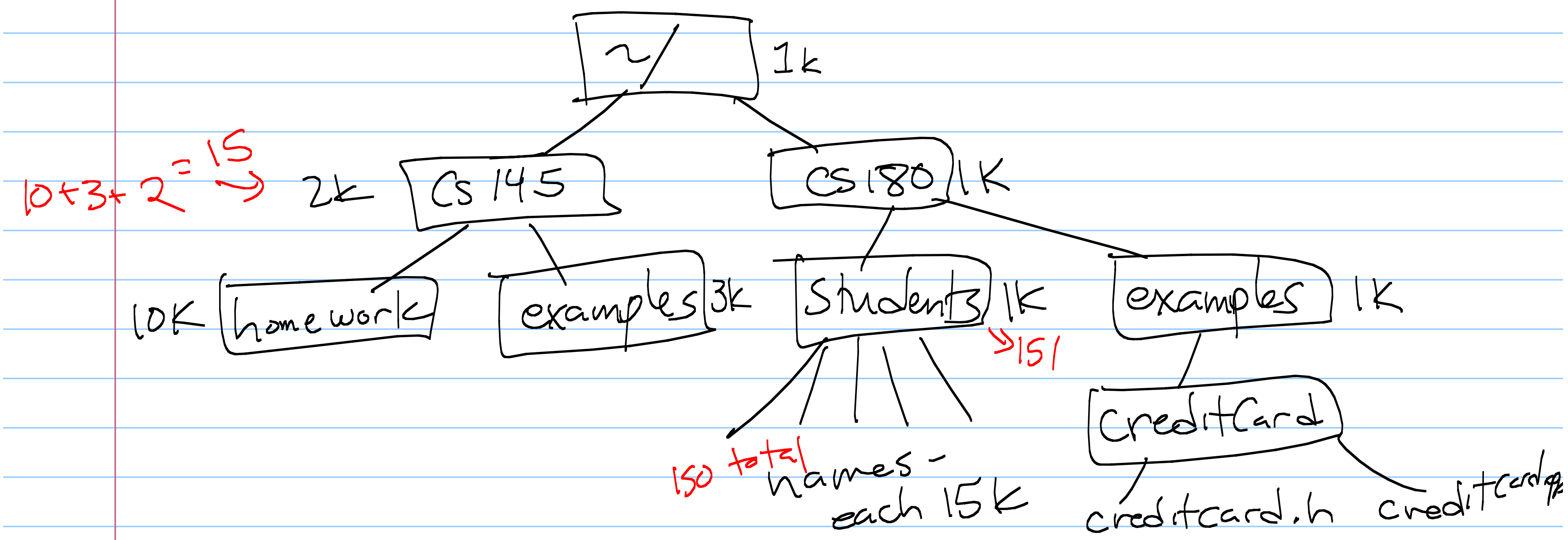
Paper – Title – Abstract – Ch 1 – 1.1 – 1.2 – Ch 2 – 2.1 – 2.2

```
Postorder(T, v):
    for each child w of v:
        postorder(T, w)
    perform action a v
```

Ex: height in a tree

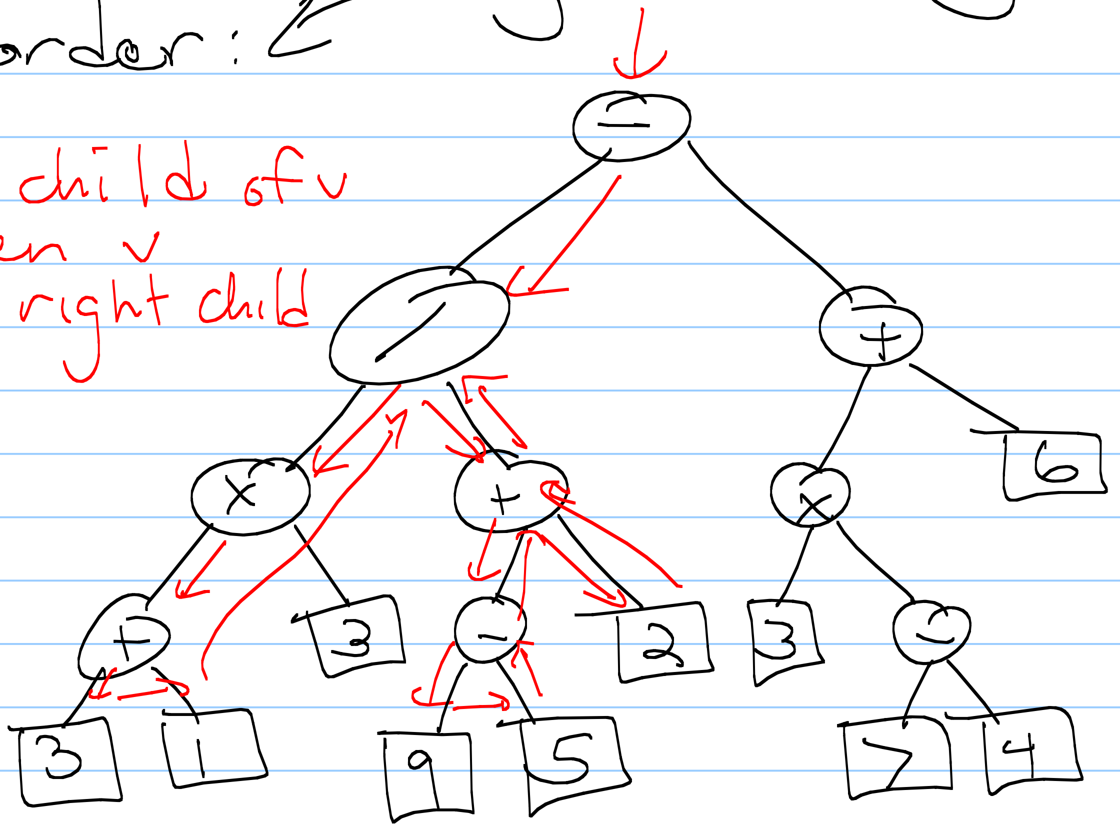
Example: size of a directory

- need to know size of children before we can compute current directory's size:



Inorder: ← only for binary trees

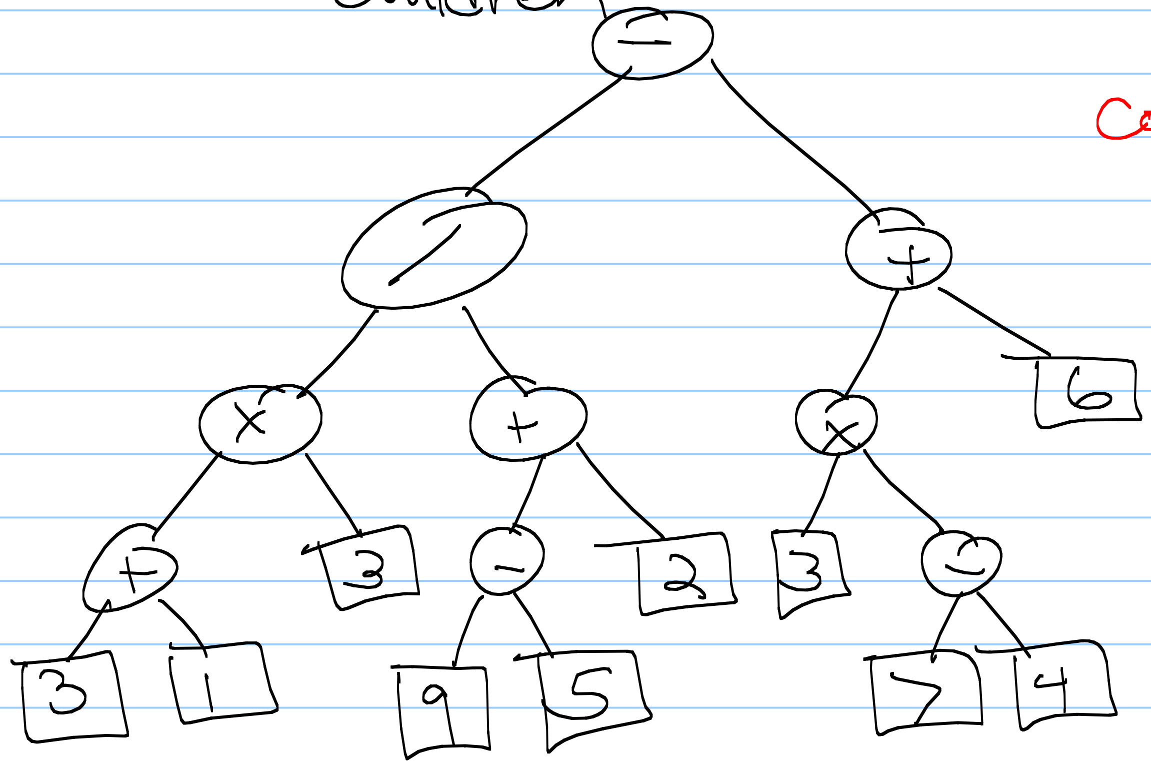
left child of v
then v
then right child



$(((3+1)\times 3)/((9-5)+2))-$

Binary Trees

- each internal node has exactly 2 children

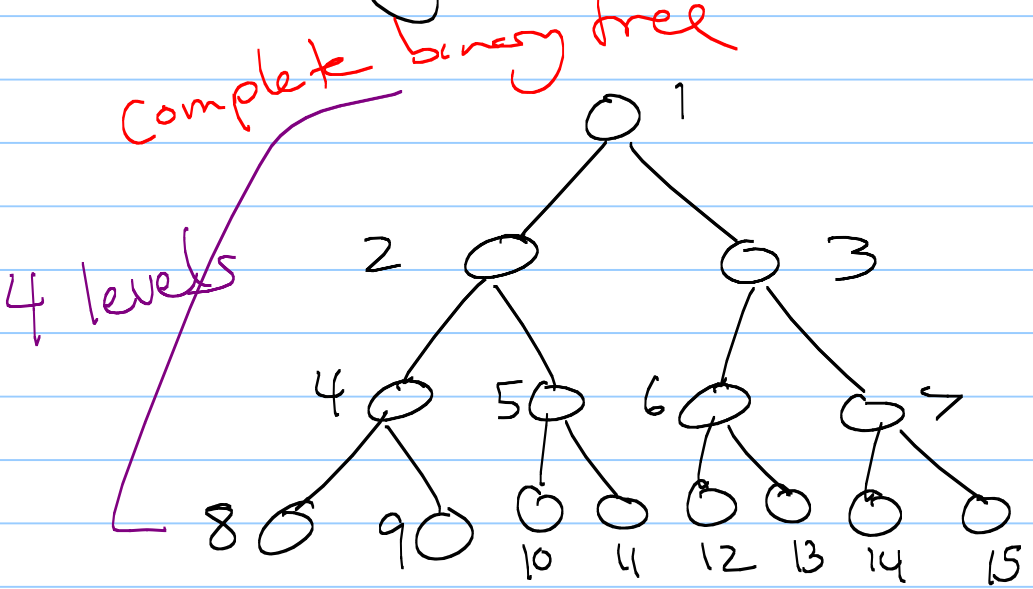


Complete: all leaves have same depth

Representations of Binary trees:

Level numbering

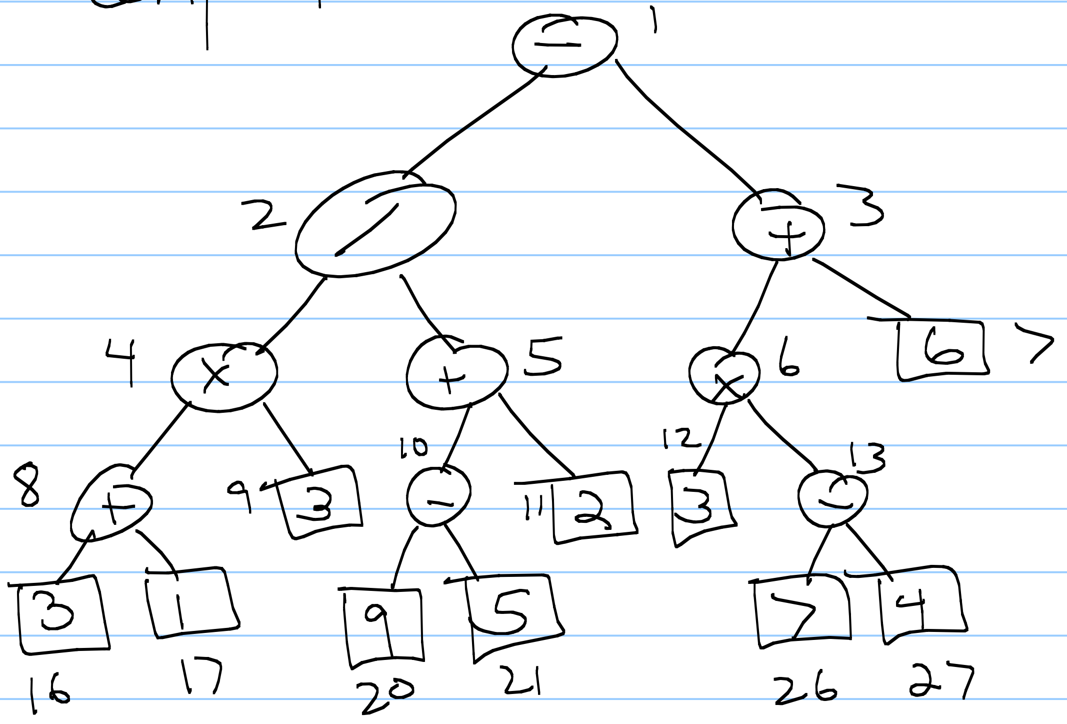
- If v is the root, $p(v) = 1$
- If v is left child of u, $p(v) = 2p(u)$
- If v is right child of u, $p(v) = 2p(u) + 1$



Note: i levels ⇒ $2^i - 1$ nodes

What if not complete?

- If v is the root, p(v) = 1
- If v is left child of u, p(v) = 2p(u)
- If v is right child of u, p(v) = 2p(u) + 1



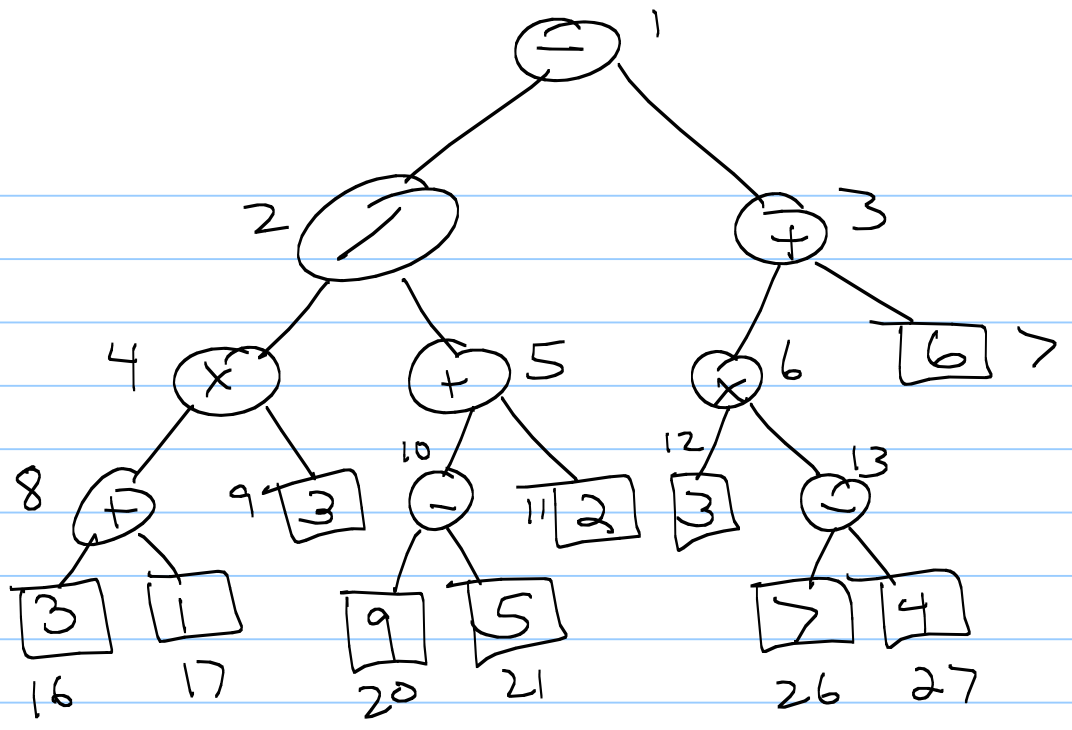
What type of underlying structure does this suggest?

array based implementation

How?

```
private:
    int[maxsize] mytree;
```

Ex:

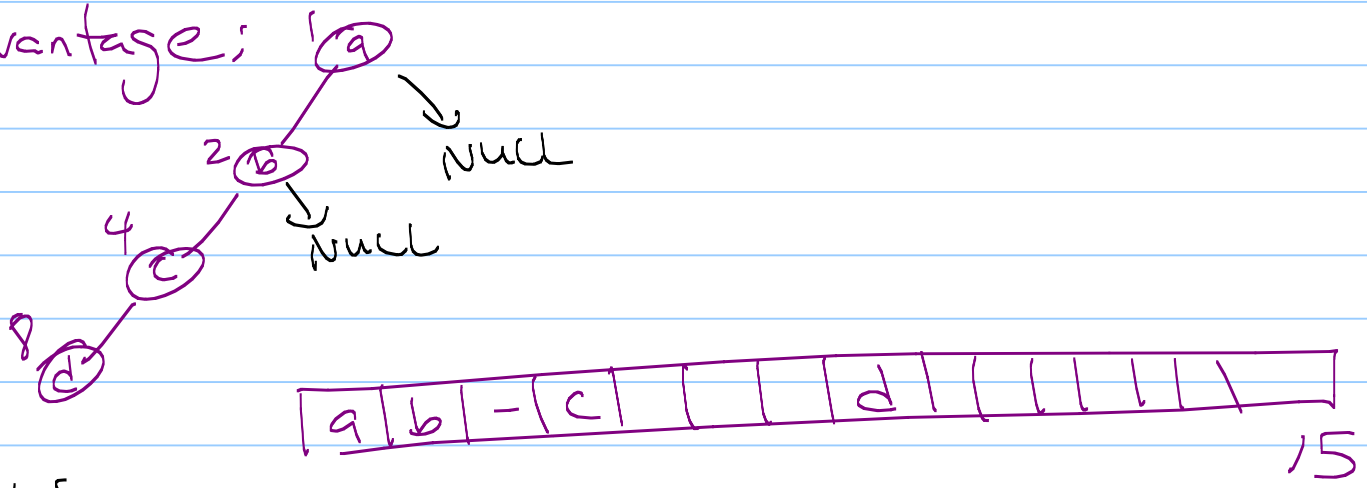


−	/	+	x	+	x	6	+	3		
1	2	3	4	5	6	7	8	9		

Advantage for array based:

- Fast
- If the tree is complete, this is more space efficient

Disadvantage:



want linked representation

Alternative: Linked structure (for binary)

```
struct Node {
    Object  element;
    Node*   parent;
    Node*   left;
    Node*   right;

    Node() : element(Object()) {
        parent = left = right = NULL;
    }
```

Priority Queue ADT (Ch. 7)

Keys versus values

↑ sort based on these

↑ data stored

Ex: Standby list for a flight

values = names of people

key = calculated based on freq. flyer, order of request, + price

A note about keys:

Properties: need to be able to compare them

- reflexive propery: $k \le k$
- transitive property: if $k_1 \le k_2$ and $c_2 \le k_3 \implies k_1 \le k_3$
- anti-symmetric: if $k_1 \le k_2$ and $k_2 \le k_1 \implies k_1 = k_2$

P. Q. ADT:

Methods:

- insertItem(k,e)
- minElement(): returns the element with the smallest key
- removeMin(): removes element with minimum key

Next time – code these w/ an array based binary tree