[TAP:IVHZO] Bit-shifting

Given

int x =12 >> 3;

int y = x << 1;

- What is y?

A. 1

B. 2

C. 10

D. None of the above

E. Whatever

Today's Outline

- Tree
  - Tree
  - Binary Tree

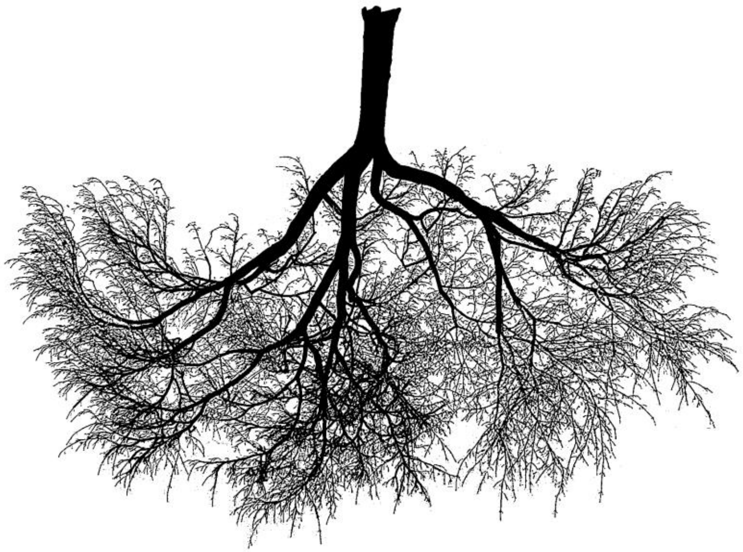
Introducing Trees

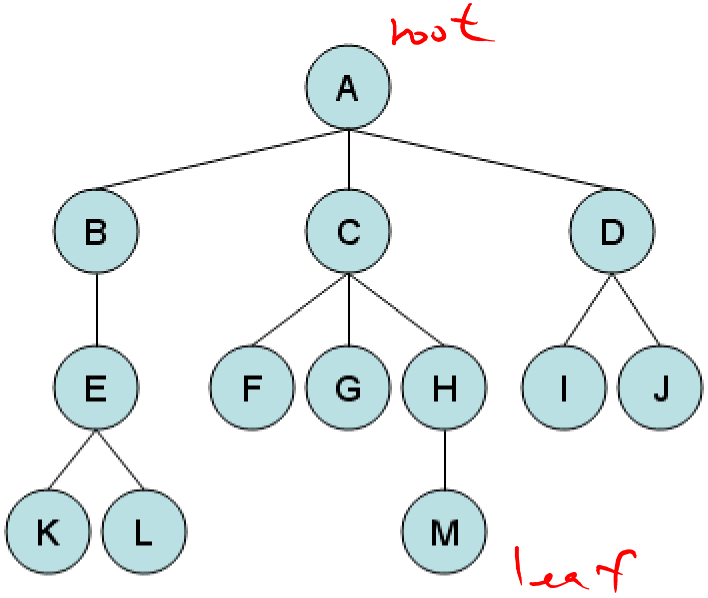
- We have been studying structures with a linear organization, i.e. each node has at most 1 successor.
  - line at a final exam
  - stacks of plates

- But you may want to allow more than 1 successor!
  - family tree ← ?
  - road network ← not a tree
  - directory hierarchy ← tree!

Tree

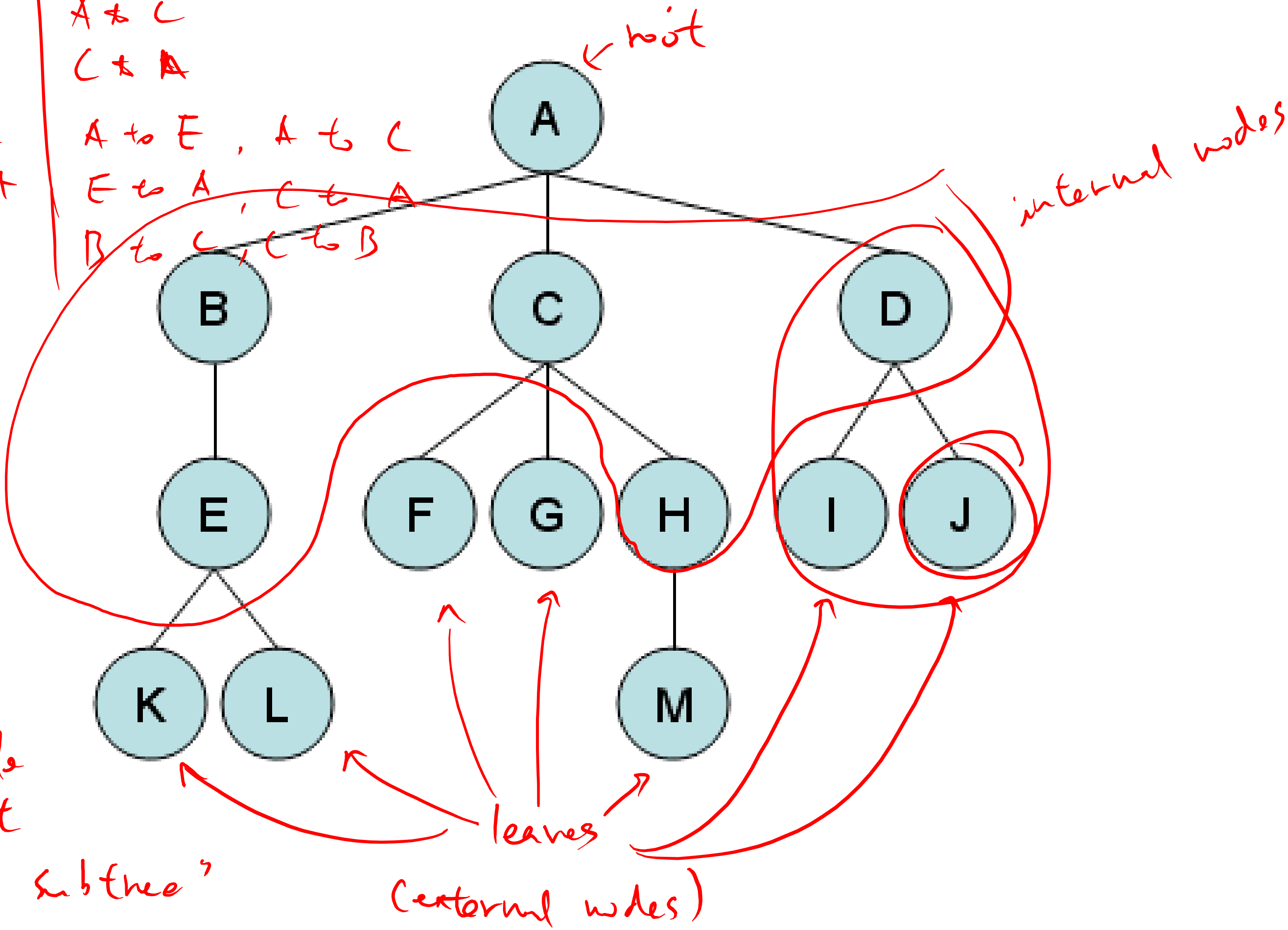
- A tree is a data structure where nodes can have:
  - one predecessor (called parent)
  - multiple successors (called children)





Tree

	ex
parent	A to C
child	C to A
ancestor	A to E, A to C
descendant	E to A, C to A
sibling	B to C, C to B



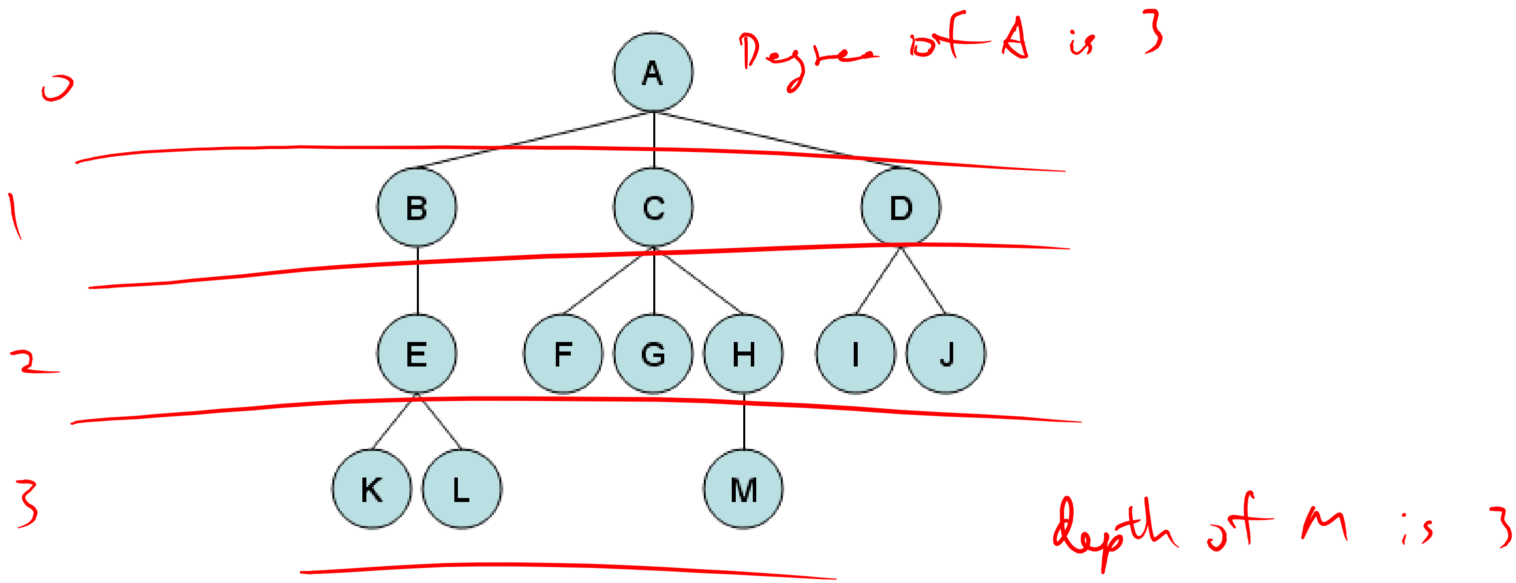
subtree:
"every node is the root of its own subtree"

Tree Logic (Natalie Jereminjenko) at Mass MoCA

https://berkshireonstage.com/2016/07/08/get-a-sneak-peek-at-mass-mocas-phase-iii-expansion-its-a-game-changer/rs32047_joe_4-copy-copy-lpr-2-838x627/

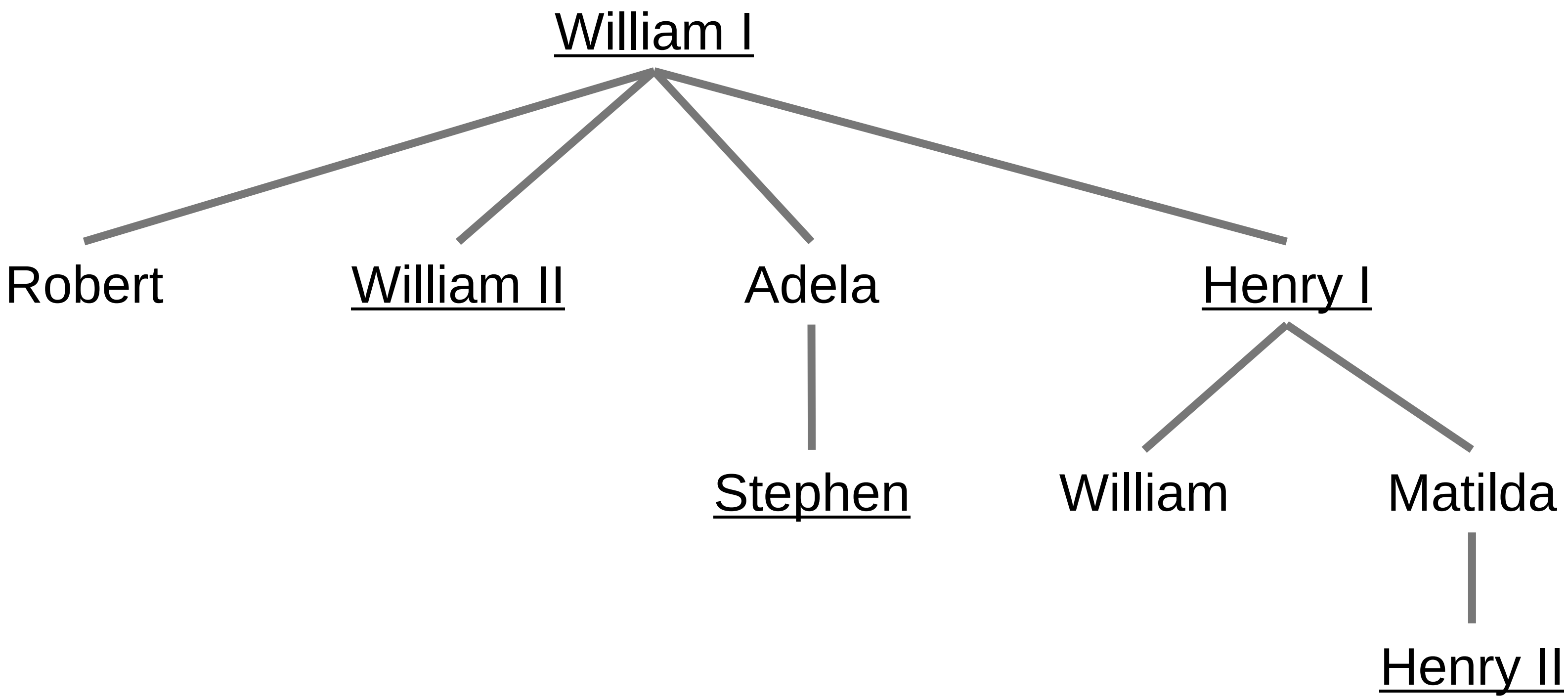
Tree Features

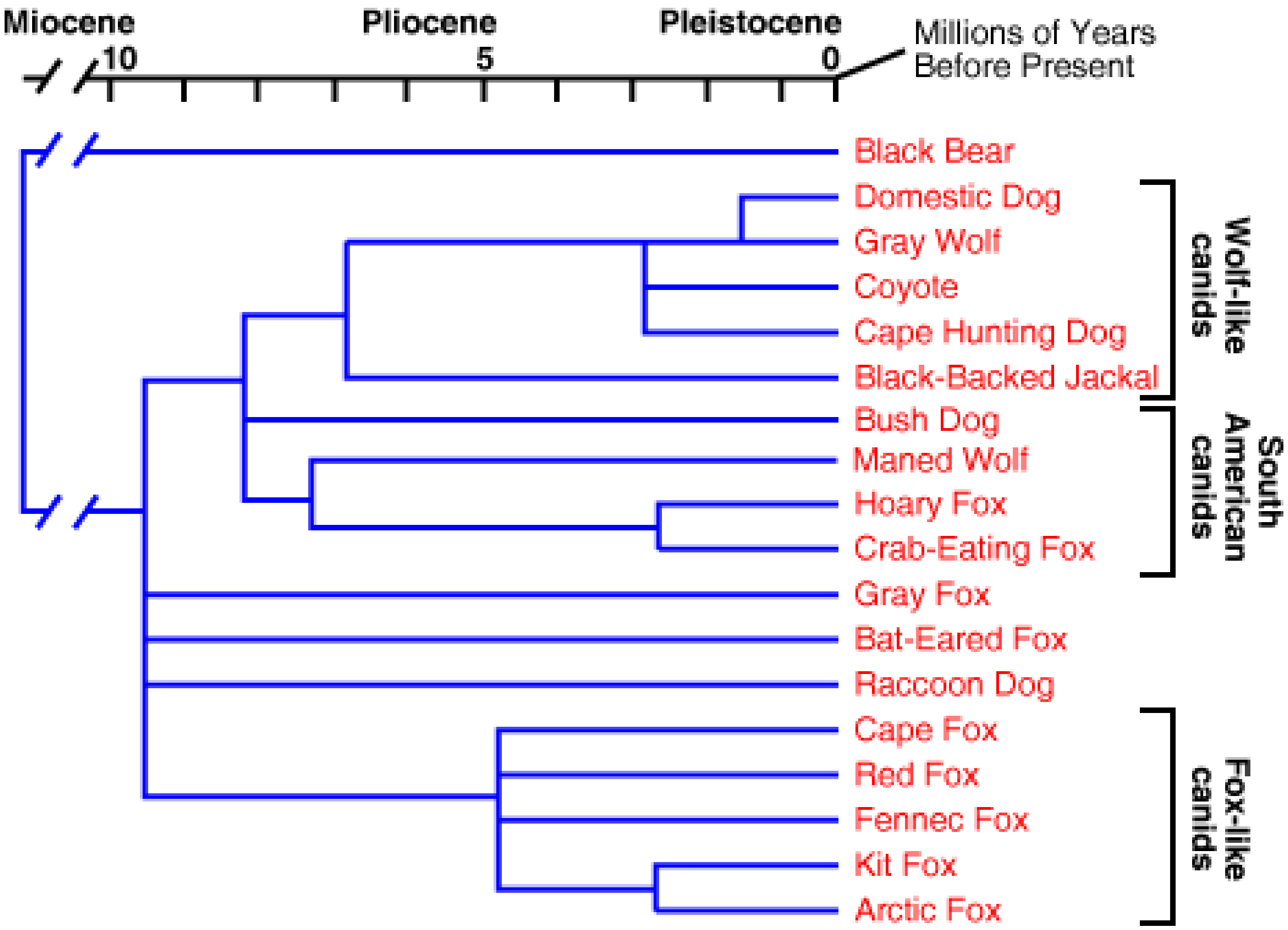
- Degree (of node): number of children of node
- Degree (of tree): maximum degree (across all nodes)
- Depth of node: number of *edges* from root to node
- Height of tree: maximum depth (across all nodes)



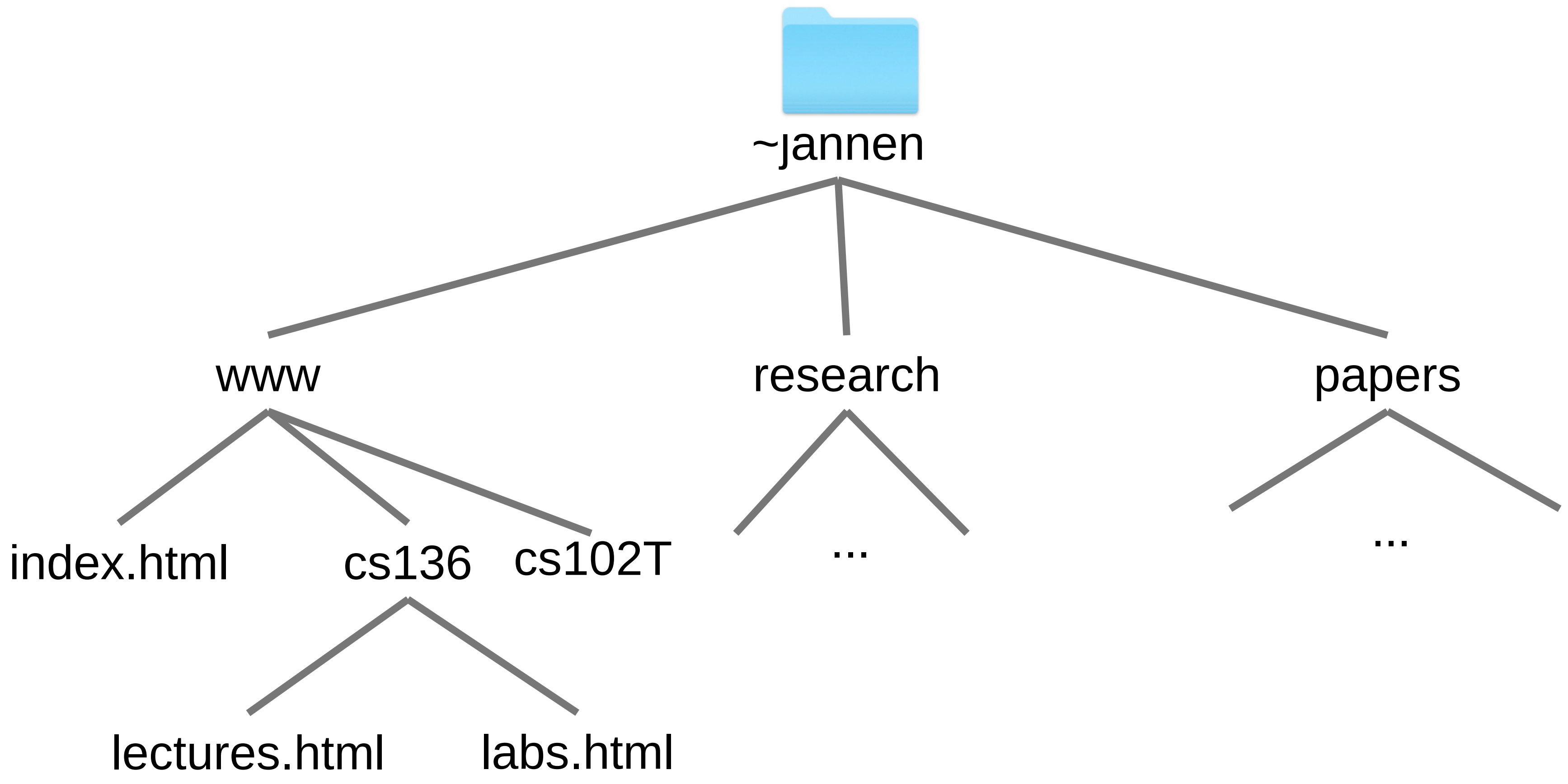
Tree examples

House of Normandy, Battle of Hastings, 1066

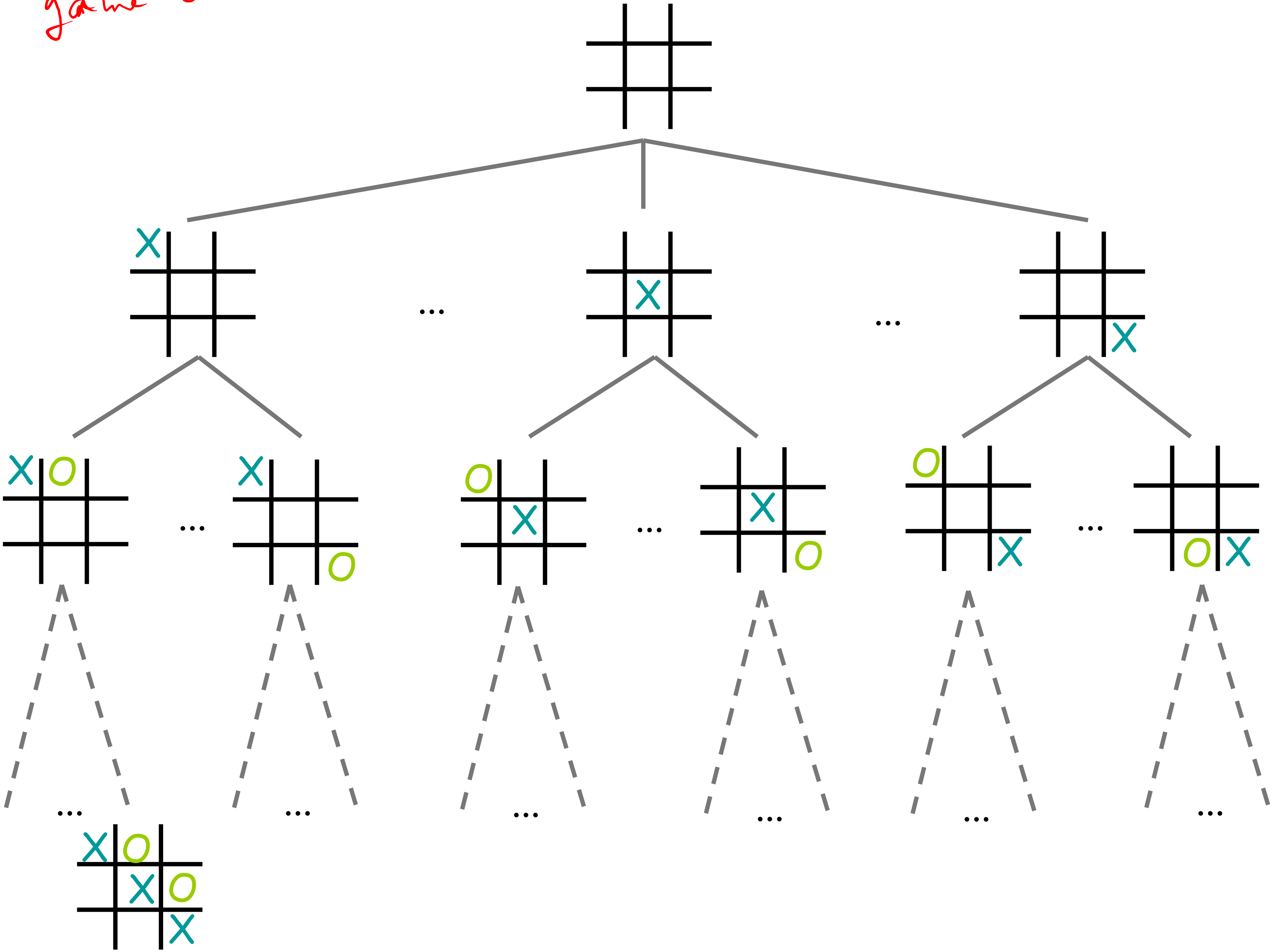




Miocene
Pliocene
Pleistocene
10
5
0
Millions of Years Before Present
Black Bear
Domestic Dog
Gray Wolf
Coyote
Cape Hunting Dog
Black-Backed Jackal
Bush Dog
Maned Wolf
Hoary Fox
Crab-Eating Fox
Gray Fox
Bat-Eared Fox
Raccoon Dog
Cape Fox
Red Fox
Fennec Fox
Kit Fox
Arctic Fox
Wolf-like canids
South American canids
Fox-like canids


~jannen
www
research
papers
index.html
cs136
cs102T
...
...
lectures.html
labs.html

game tree



Today's Outline

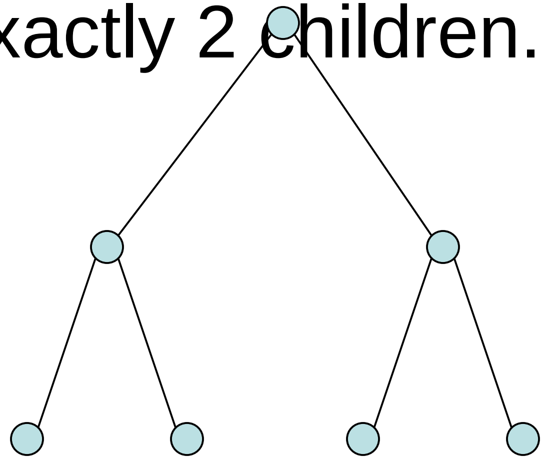
- Tree
  - Tree
  - Binary Tree

Binary Trees

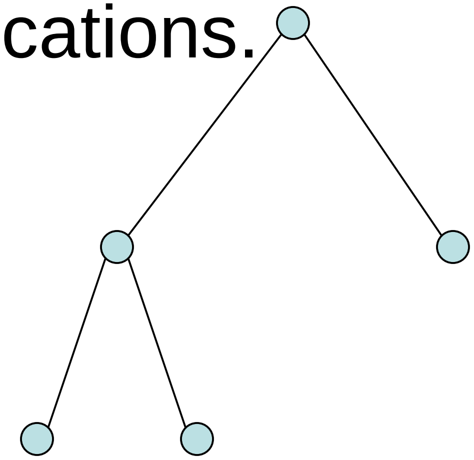
- Binary Tree: Tree with Degree of each node <= 2
- Recursively defined. A tree can either be:
  - Empty
  - Root with left and right subtrees

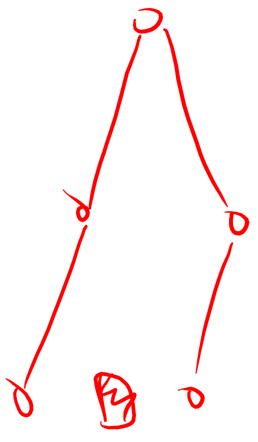
Full vs. Complete

- Full tree – A full binary tree of height h has *leaves only* on level h, and each internal node has exactly 2 children.



- Complete tree – A *complete* binary tree of height h is *full* to height h-1 and has all leaves at level h in leftmost locations.

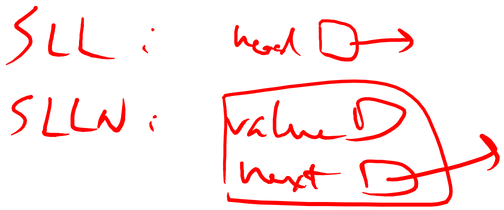


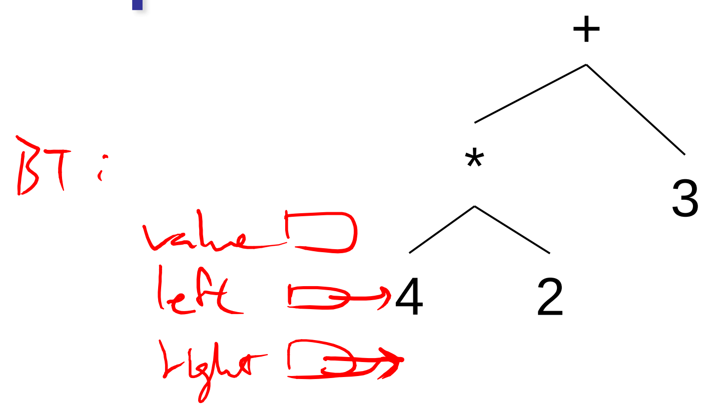


All full trees are complete, but not all complete trees are full!

Example: Expression Trees

4 * 2 + 3





Build using constructor

```
new BinaryTree<E>(value, leftSubTree, rightSubTree)
```

```
BinaryTree<String> fourTimesTwo =
                new BinaryTree<String>(“*”,
                new BinaryTree<String>(“4”),
                new BinaryTree<String>(“2”));
BinaryTree<String> fourTimesTwoPlusThree =
                new BinaryTree<String>(“+”,
                fourTimesTwo,
                new BinaryTree<String>(“3”));
```

Evaluating Expression Trees

- Starting at the root,
  - Evaluate left subree
  - Evaluate right subtree
  - Perform operation (+, -, *, /) with left and right

```
int evaluate(BinaryTree<String> tree) {
    //base case
    if (tree.height() == 0)
        return Integer.parseInt(tree.value());

    //recursive case
    int left = evaluate(tree.left());
    int right = evaluate(tree.right());

    switch(tree.value()){
        case "+":
            return left + right;
        case "-":
            return left - right;

        case "*":
            return left * right;
        case "/":
            return left / right;
    }
    return ERROR_CODE;
}
```