Programming Languages and Techniques (CIS120)

Lecture 6

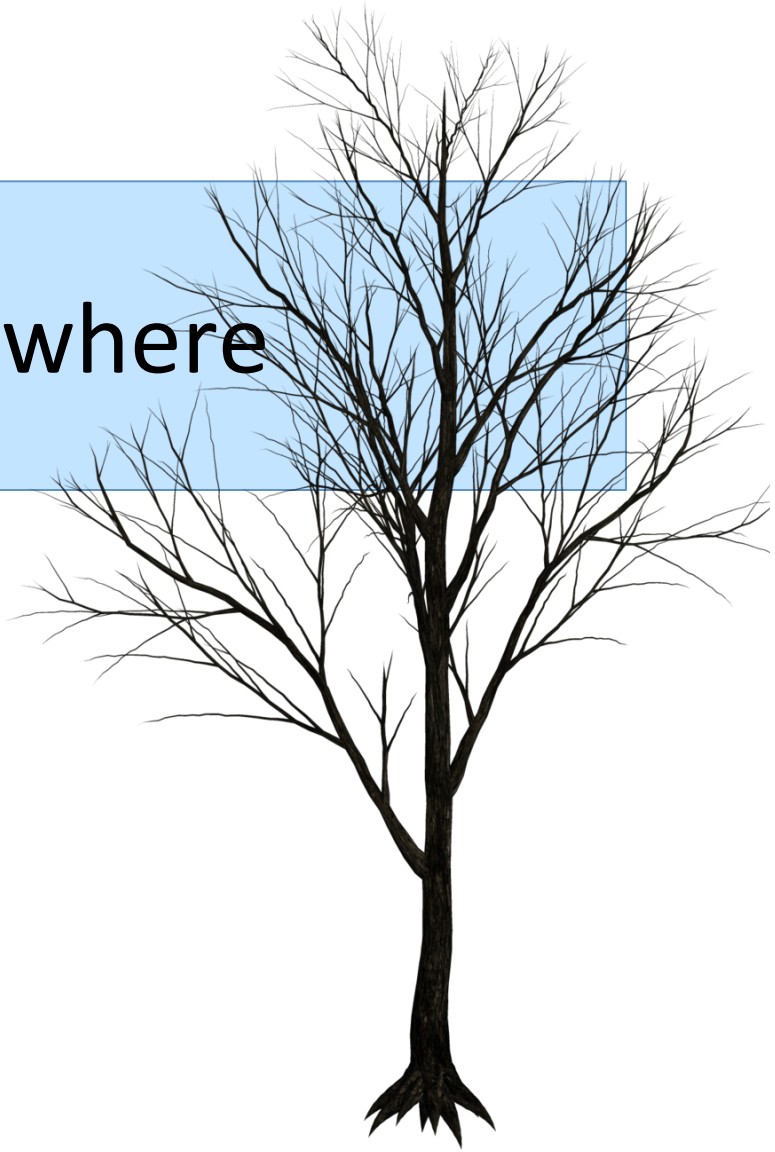
September 9th , 2015

Binary Trees and Binary Search Trees

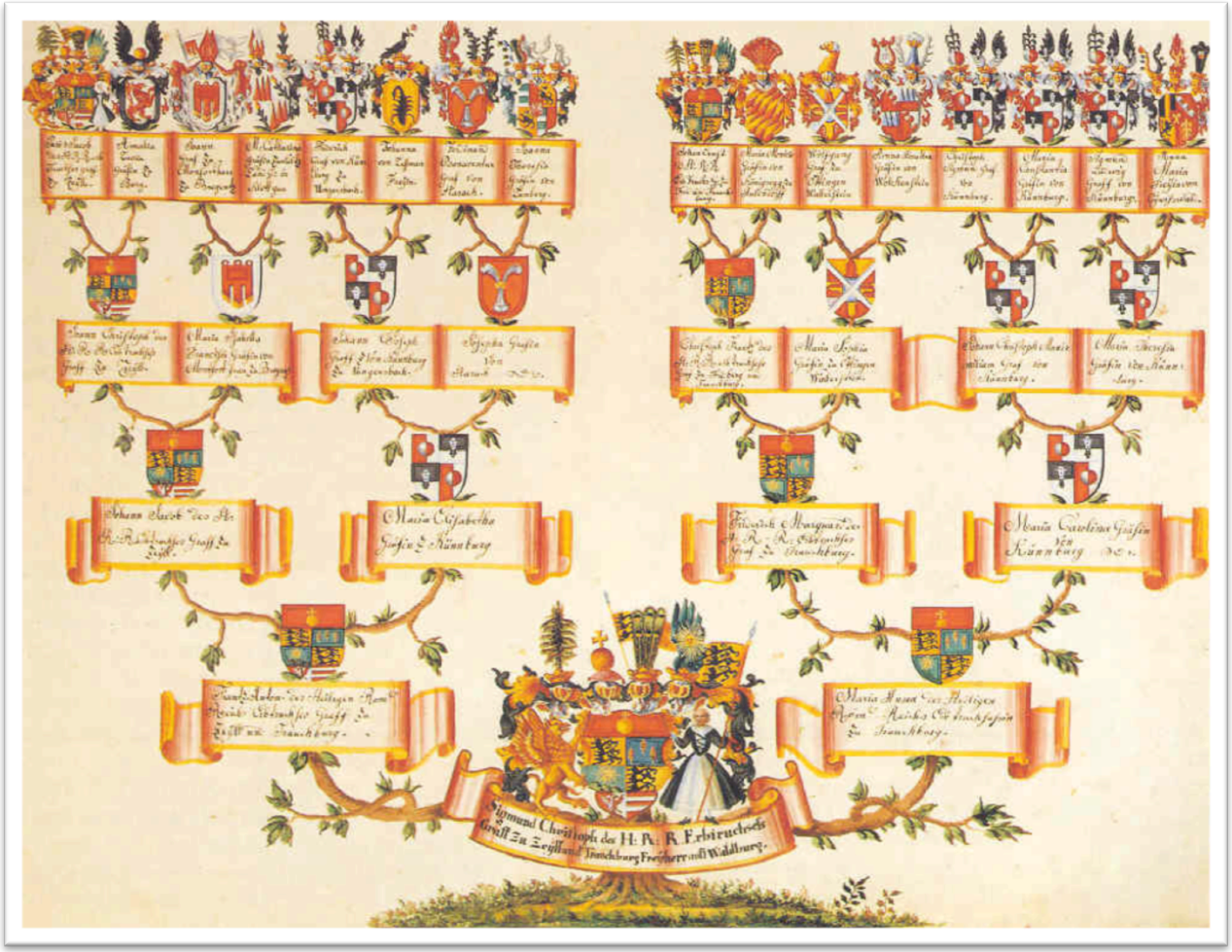
Announcements

- Great job on HW1!
- Homework 2 is available
  - due Tuesday, Sept. 15th
- Lecture attendance grade (i.e. clickers)
  - Flexibility for occasional missed lectures due to minor emergencies (i.e. it's OK to miss a few lectures)
  - No need to inform staff (or send CAR) unless you have a major emergency
- Please complete the CIS 120 Demographics Survey
  - See Piazza (or this week's labs)
- Read Chapter 6 and 7

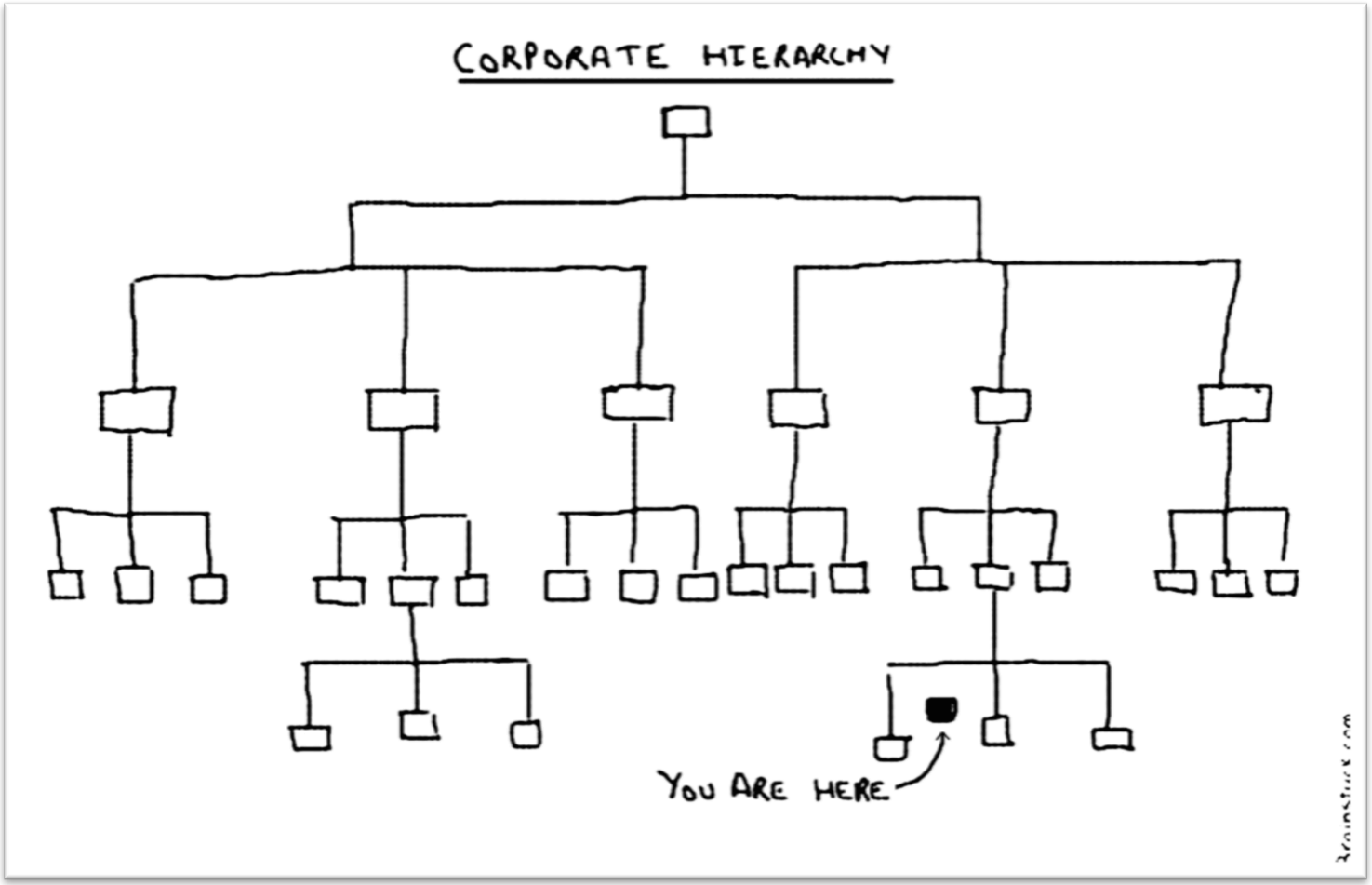
Trees are everywhere



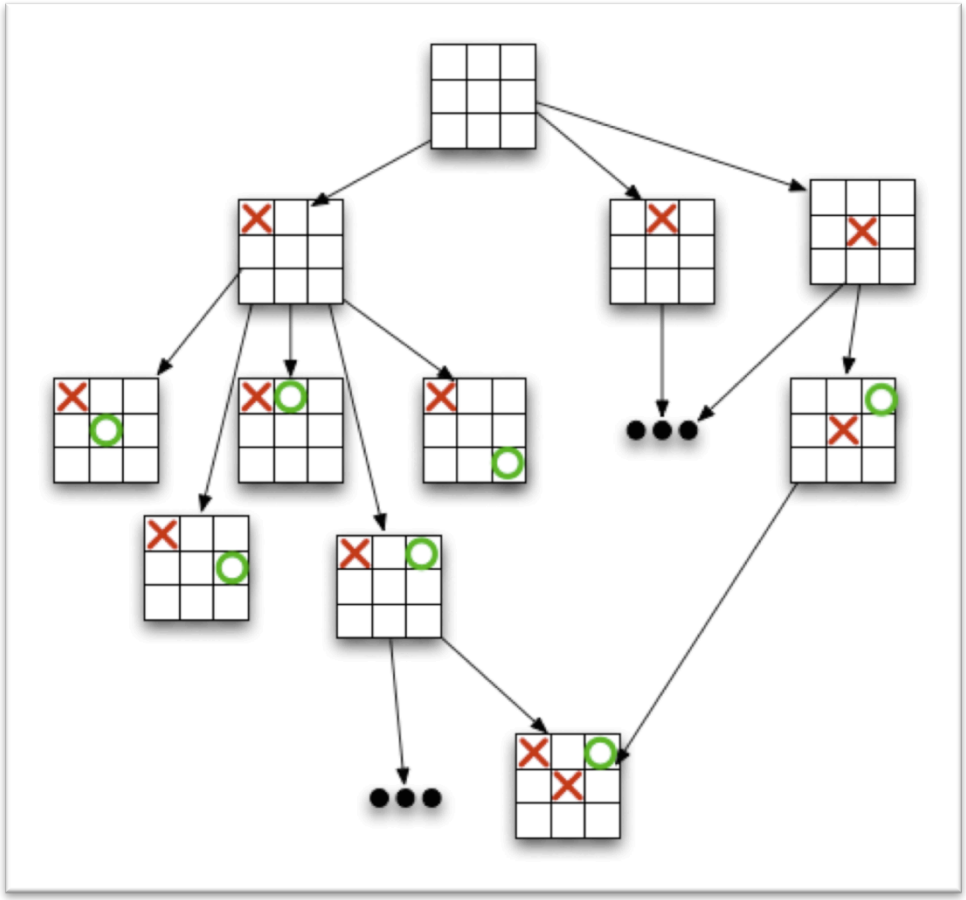
Family trees



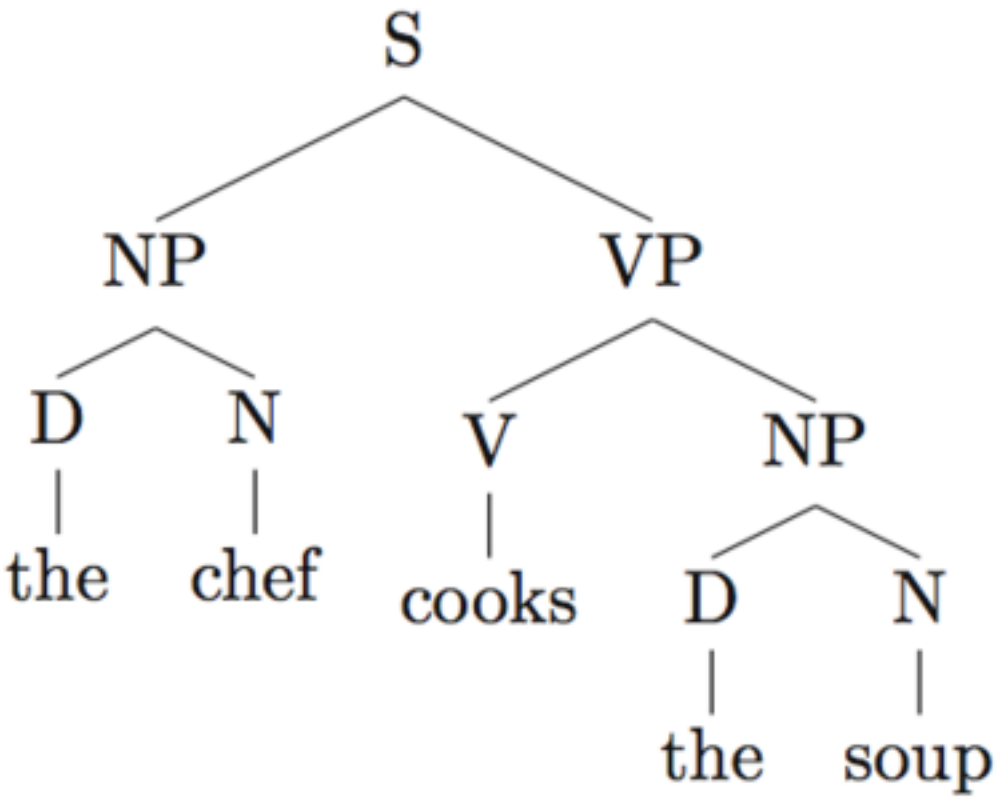
Organizational charts



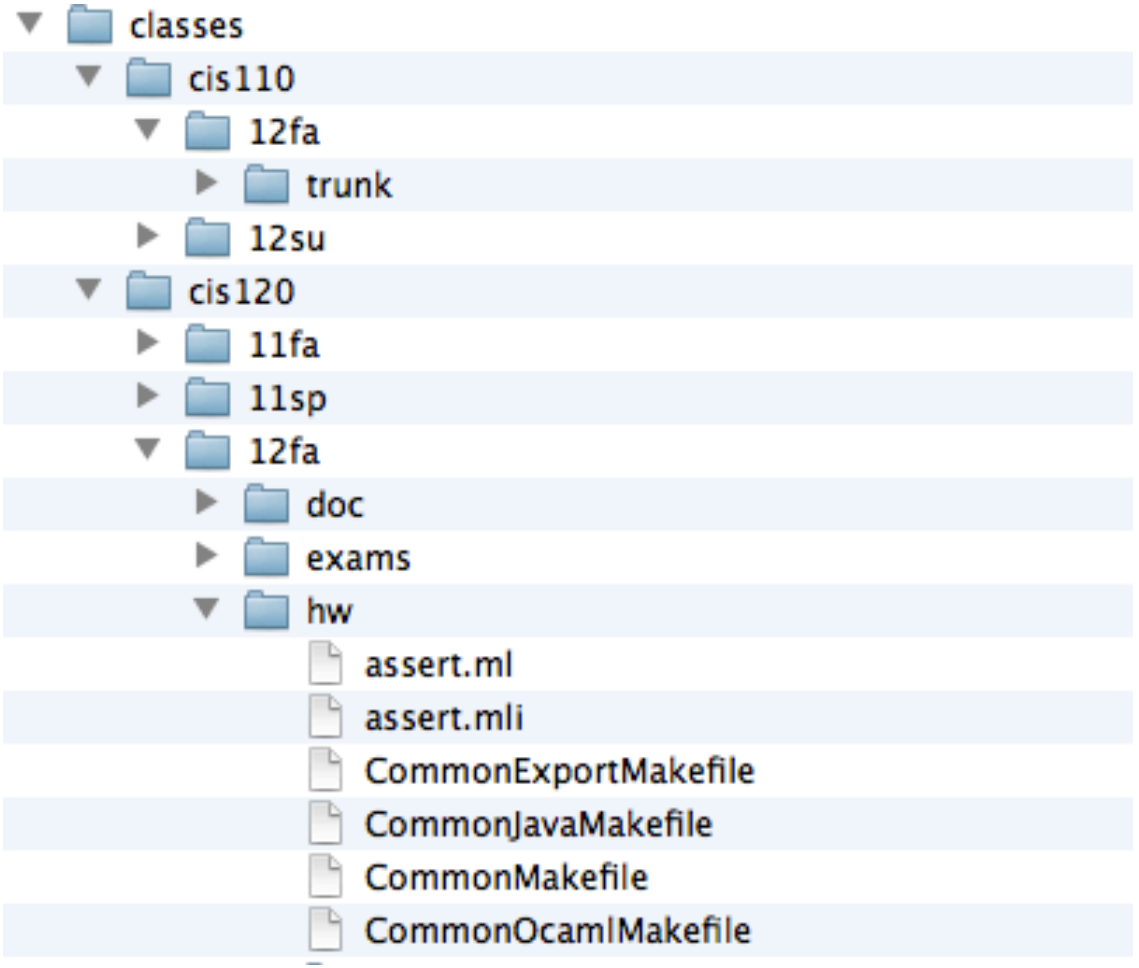
Game trees



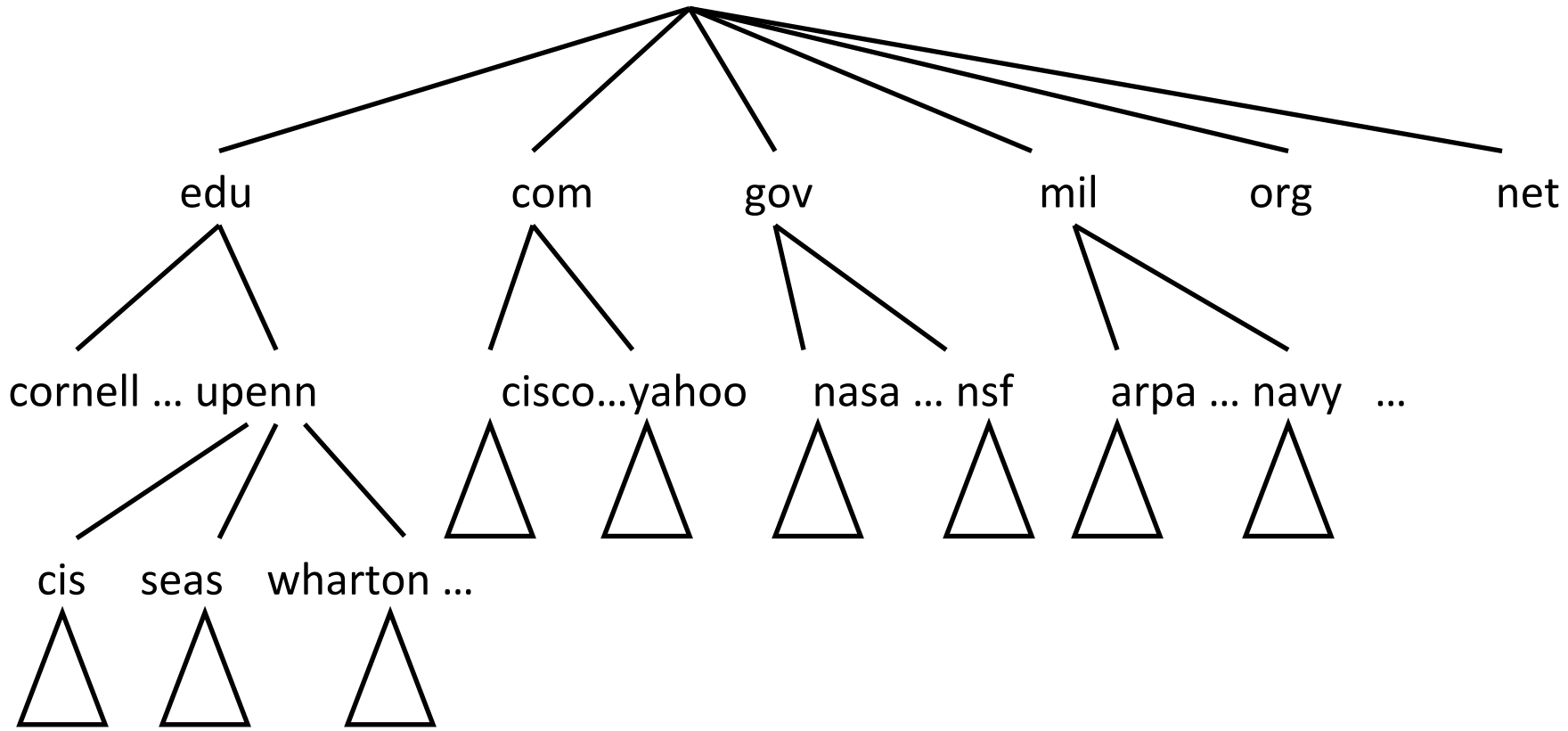
Natural-Language Parse Trees



Filesystem Directory Structure



Domain Name Hierarchy



Clickers, please…

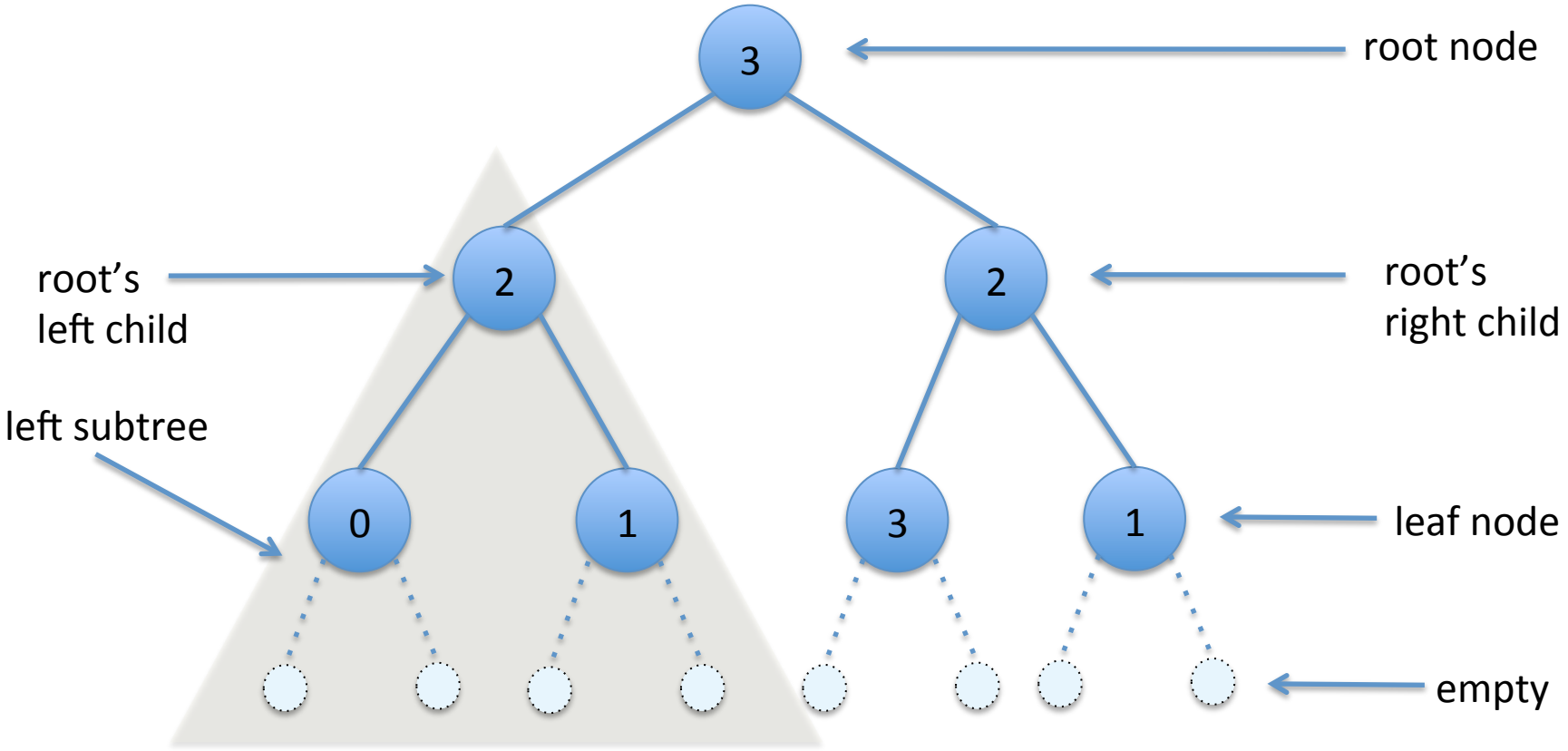
Have you ever programmed with trees before?

1. yes
2. no
3. *not sure*

Binary Trees

A particular form of tree-structured data

Binary Trees



A binary tree is either *empty*, or a *node* with at most two children, both of which are also binary trees.

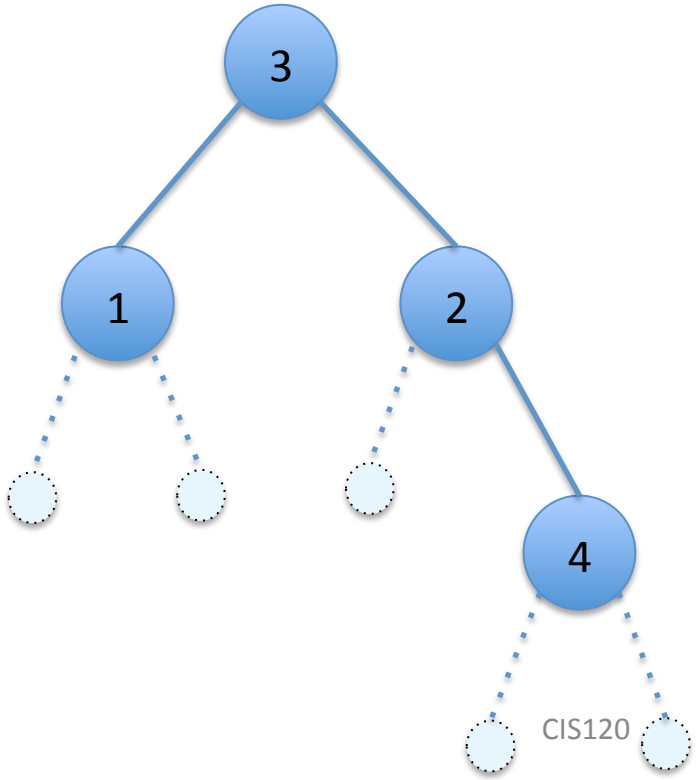
A *leaf* is a node whose children are both empty.

Binary Trees in OCaml

```
type tree =
| Empty
| Node of tree * int * tree
```

```
let t : tree =
  Node (Node (Empty, 1, Empty),
    3,
    Node (Empty, 2,
      Node (Empty, 4, Empty)))
```

=



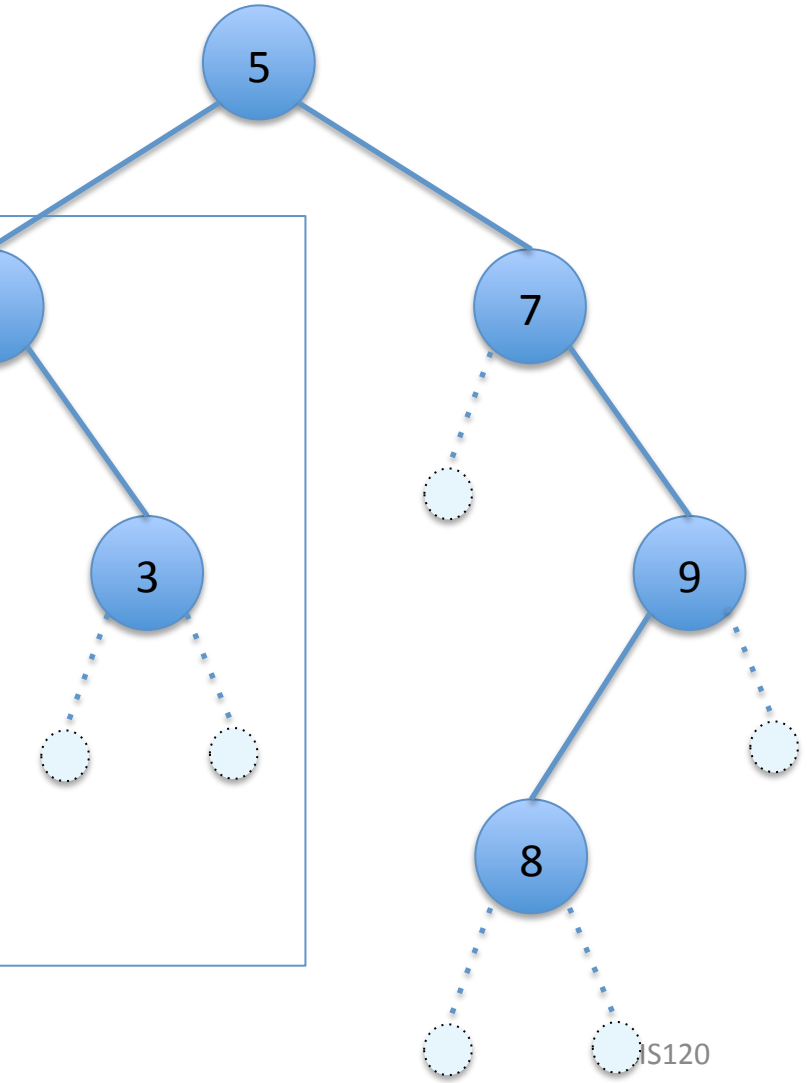
Representing trees

```
type tree =
| Empty
| Node of tree * int * tree
```

```
Node (Node (Empty, 0, Empty),
      1,
      Node (Empty, 3, Empty))
```

```
Node (Empty, 0, Empty)
```

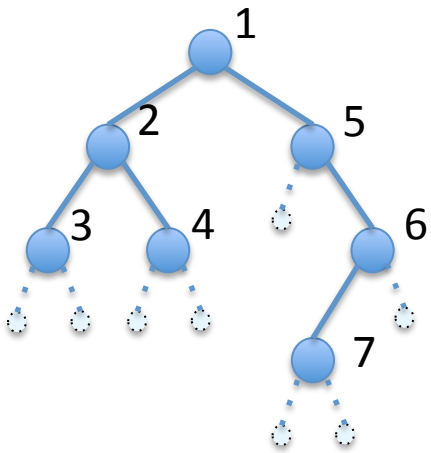
```
Empty
```



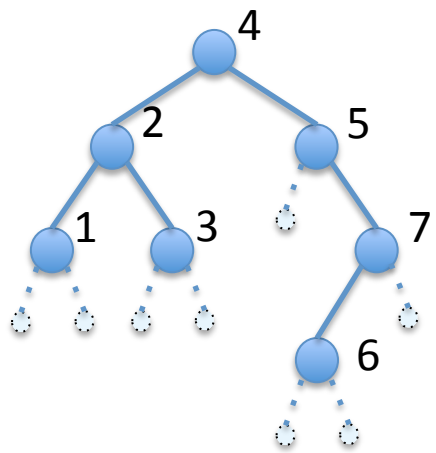
Demo

see trees.ml

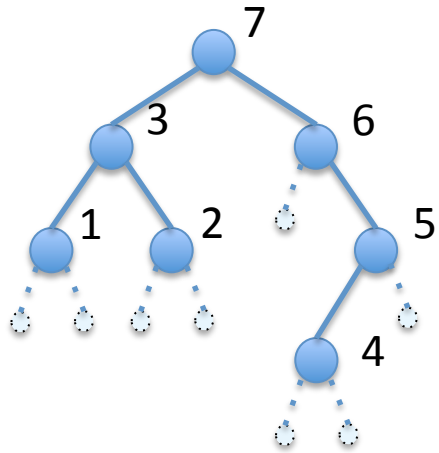
Recursive Tree Traversals



Pre-Order
Root – Left – Right



In Order
Left – Root – Right



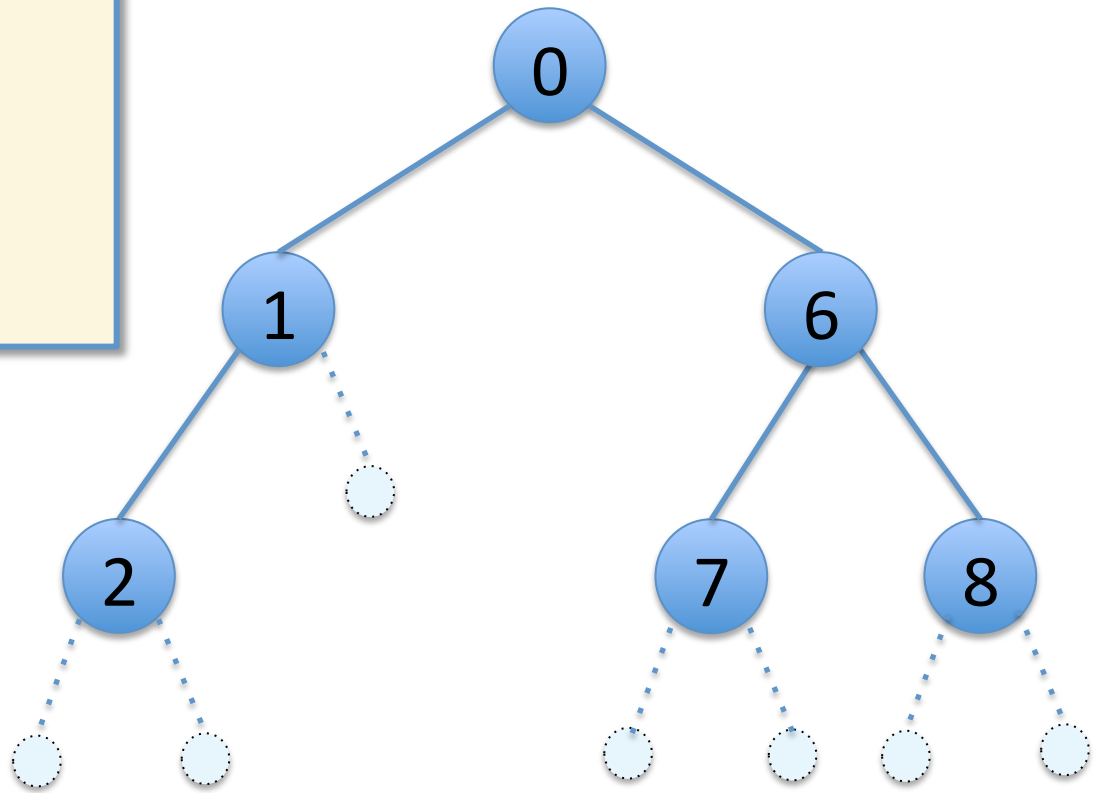
Post-Order
Left – Right – Root

```
(* Code for Pre-Order Traversal *)
let rec f (t:tree) : … =
  begin match t with
  | Empty -> …
  | Node(l, x, r) ->
    let root = … x … in (* process root *)
    let left =  f l  in (* recursively process left *)
    let right = f r  in (* recursively process right *)
        combine root left right
  end
```

The traversals vary the order in which these are computed…

In what sequence will the nodes of this tree be visited by a post-order traversal?

1. [0;1;6;2;7;8]
2. [0;1;2;6;7;8]
3. [2;1;0;7;6;8]
4. [7;8;6;2;1;0]
5. [2;1;7;8;6;0]



Post-Order
Left – Right – Root

Answer: 5

Demo

trees.ml treeExamples.ml