

The background of the slide features a close-up photograph of a large, ancient tree with a complex network of dark, twisted branches. Sunlight filters through the dense canopy of green leaves, creating bright highlights and deep shadows. The overall atmosphere is organic and natural.

Trees

Chris Piech

CS 106B
Lecture 18
Feb 8, 2016

Socrative



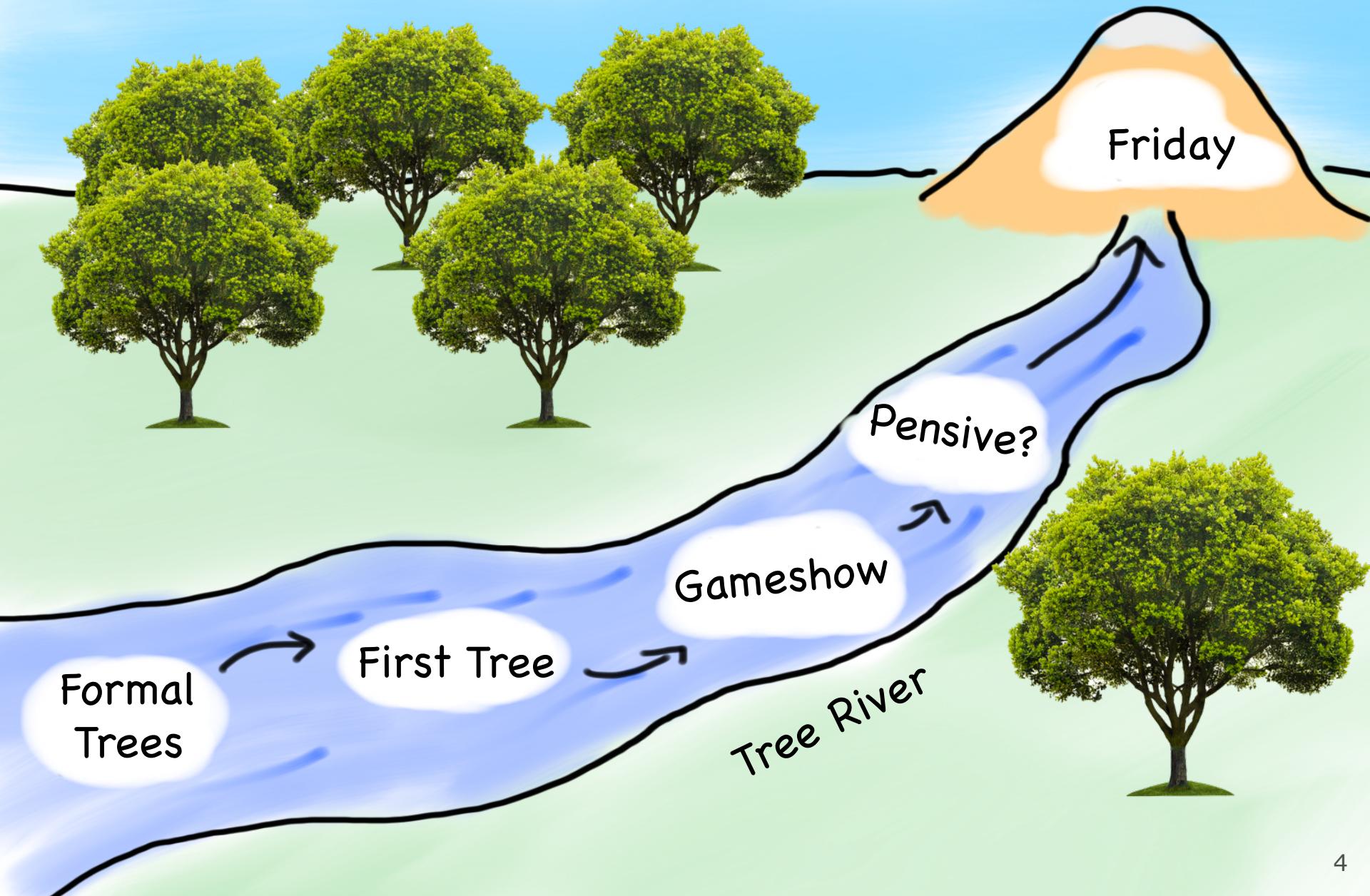
Room: **106BWIN16**

Today's Goal

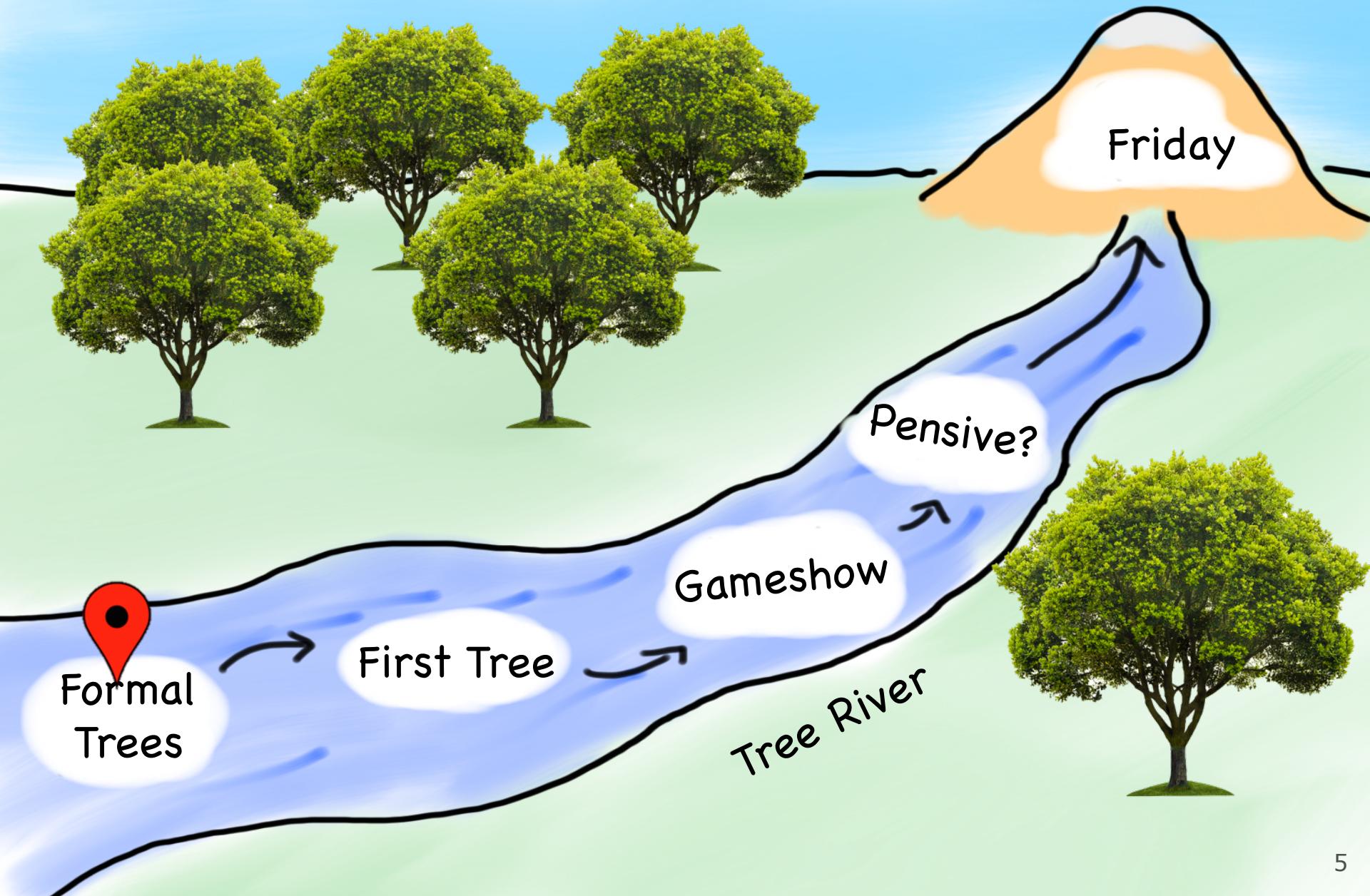
1. Be able to define a tree
2. Be able to traverse a tree



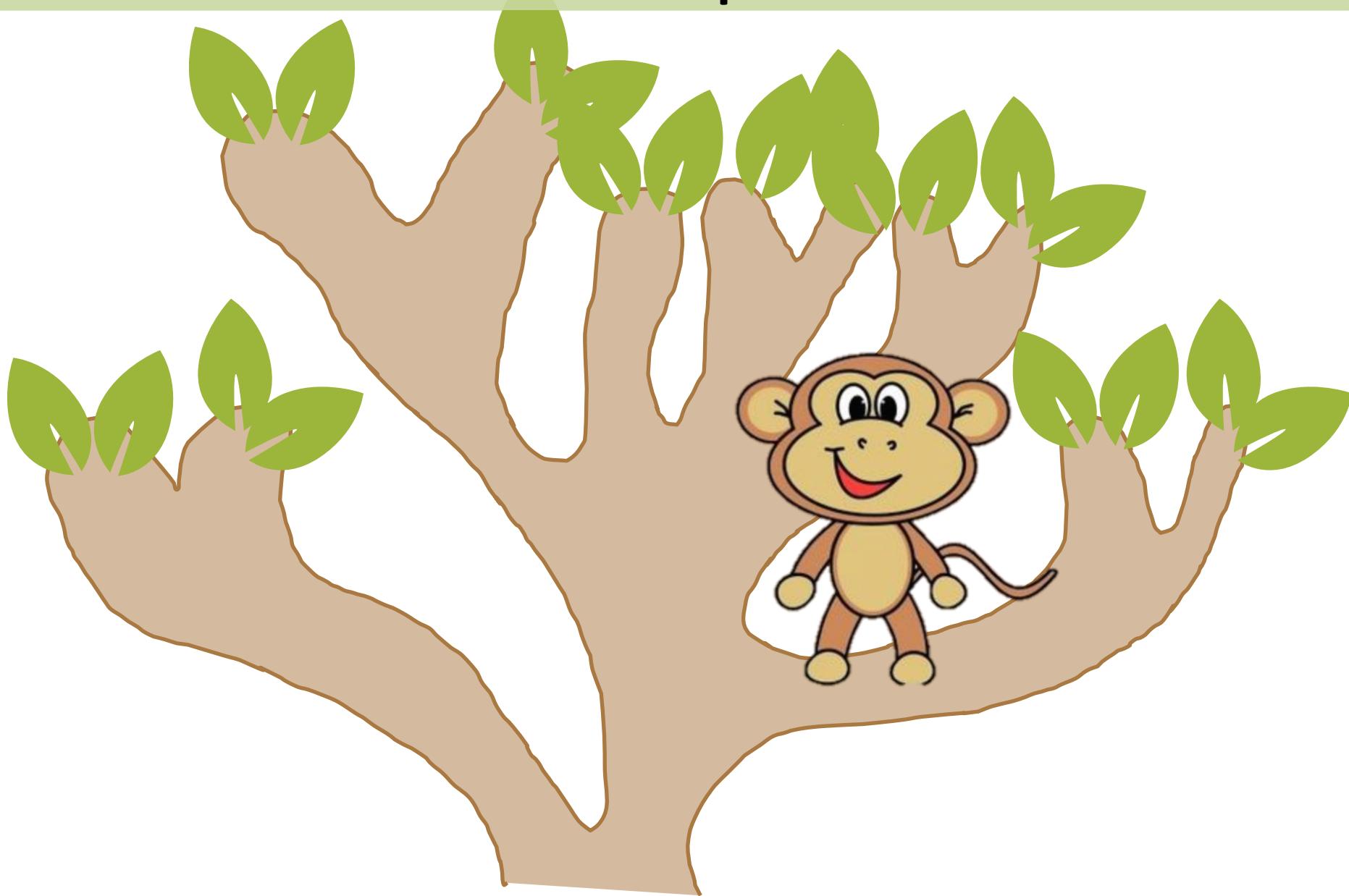
Today's Route



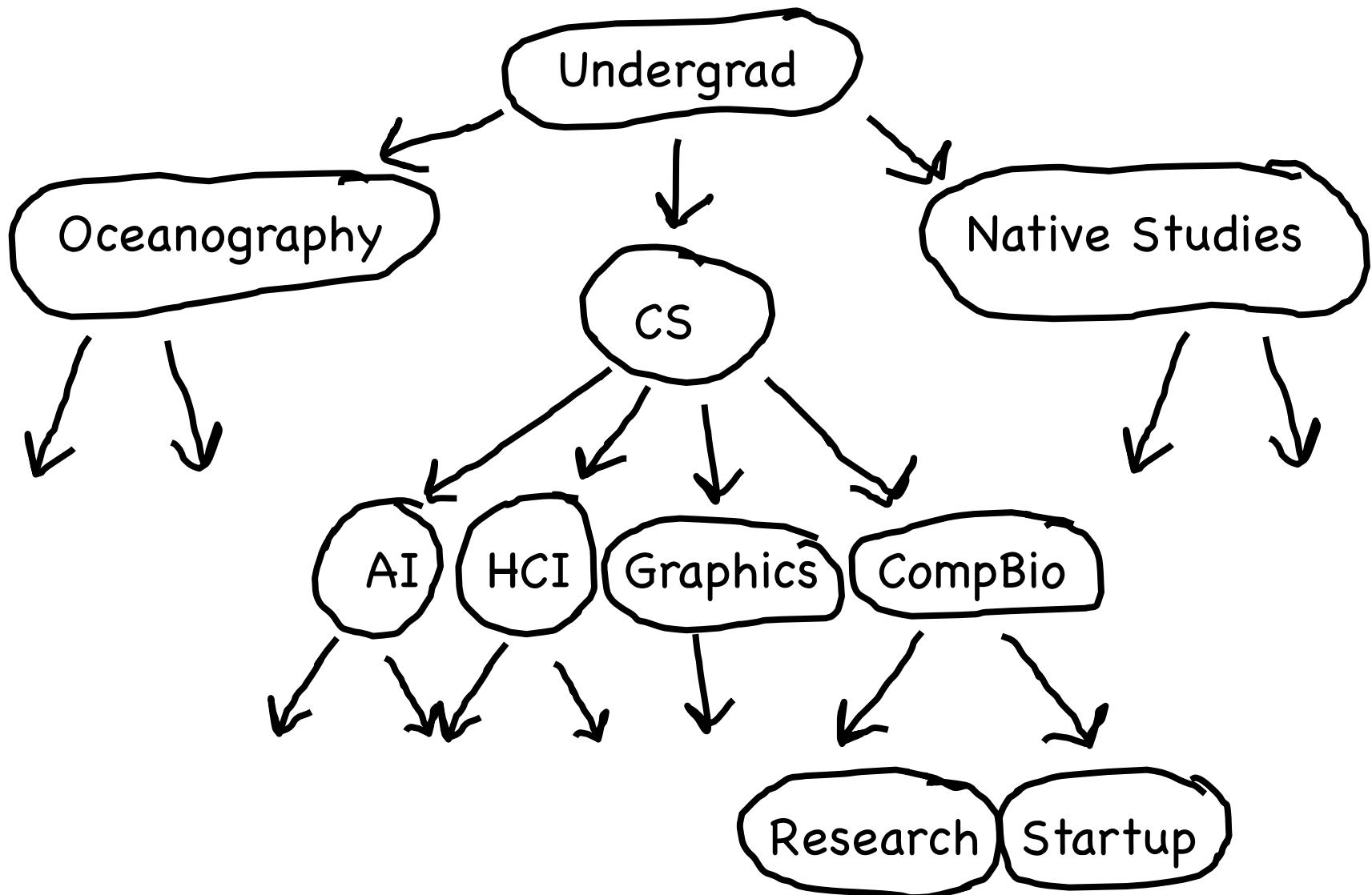
Today's Route



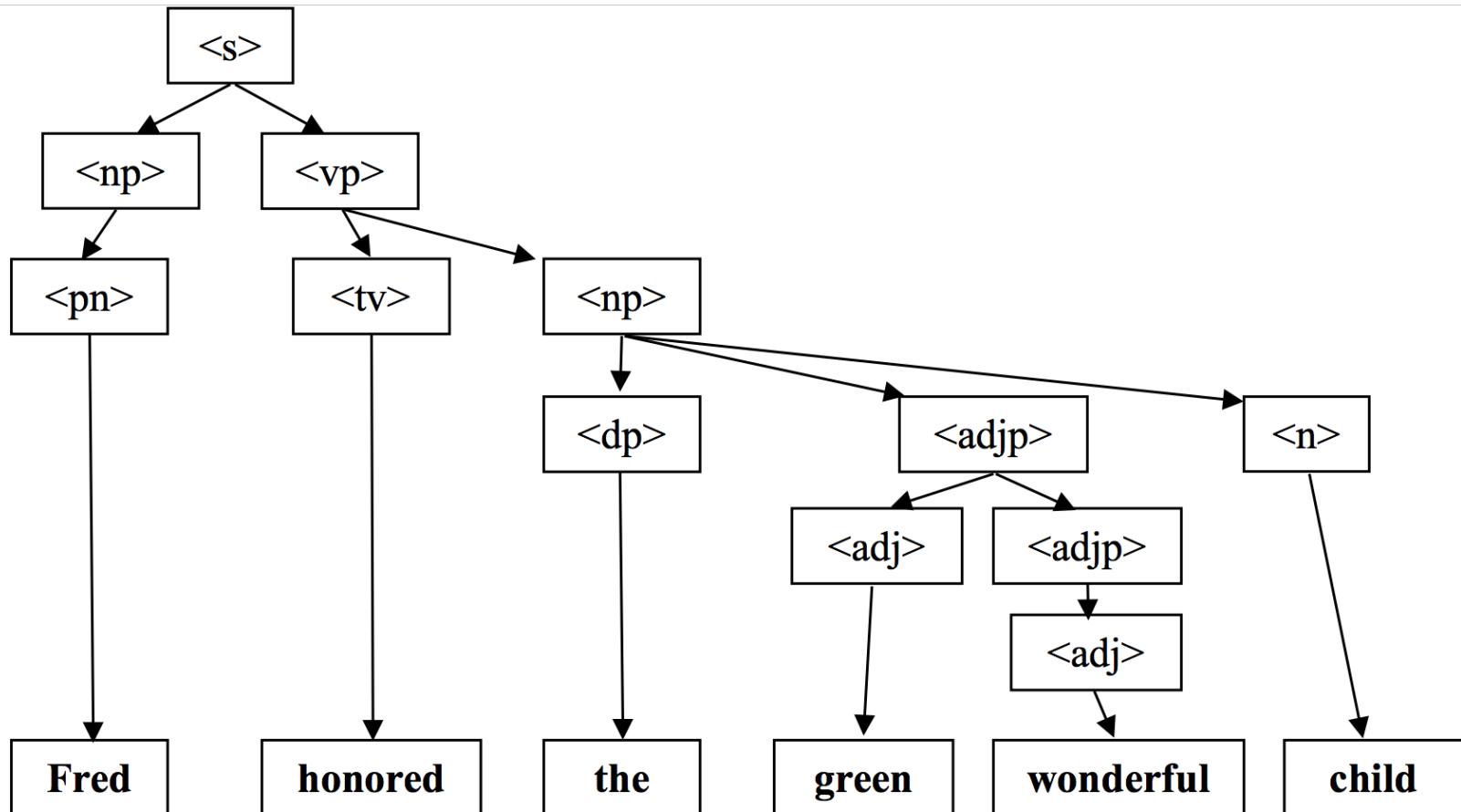
Recursive Exploration



Decision Trees

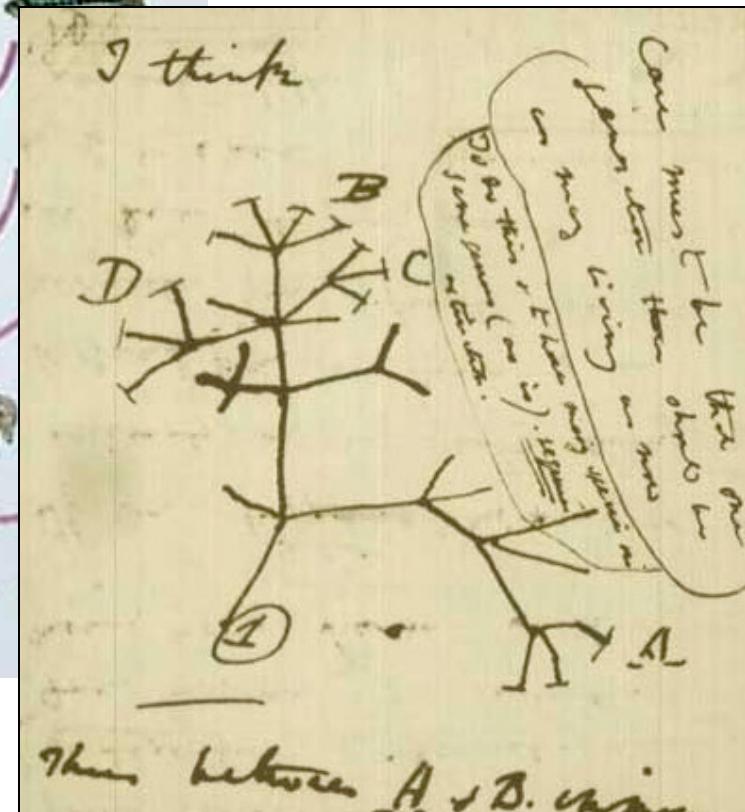
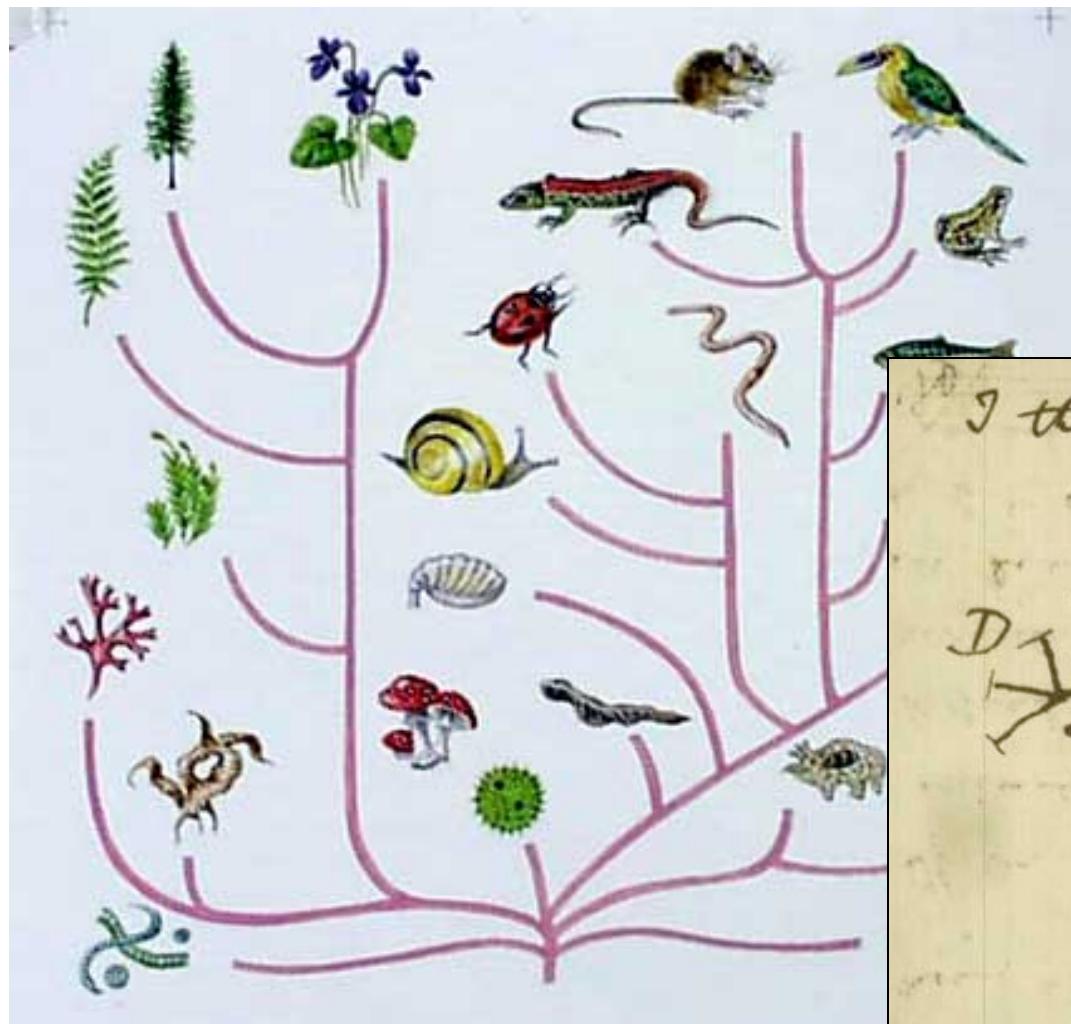


Syntax Tree

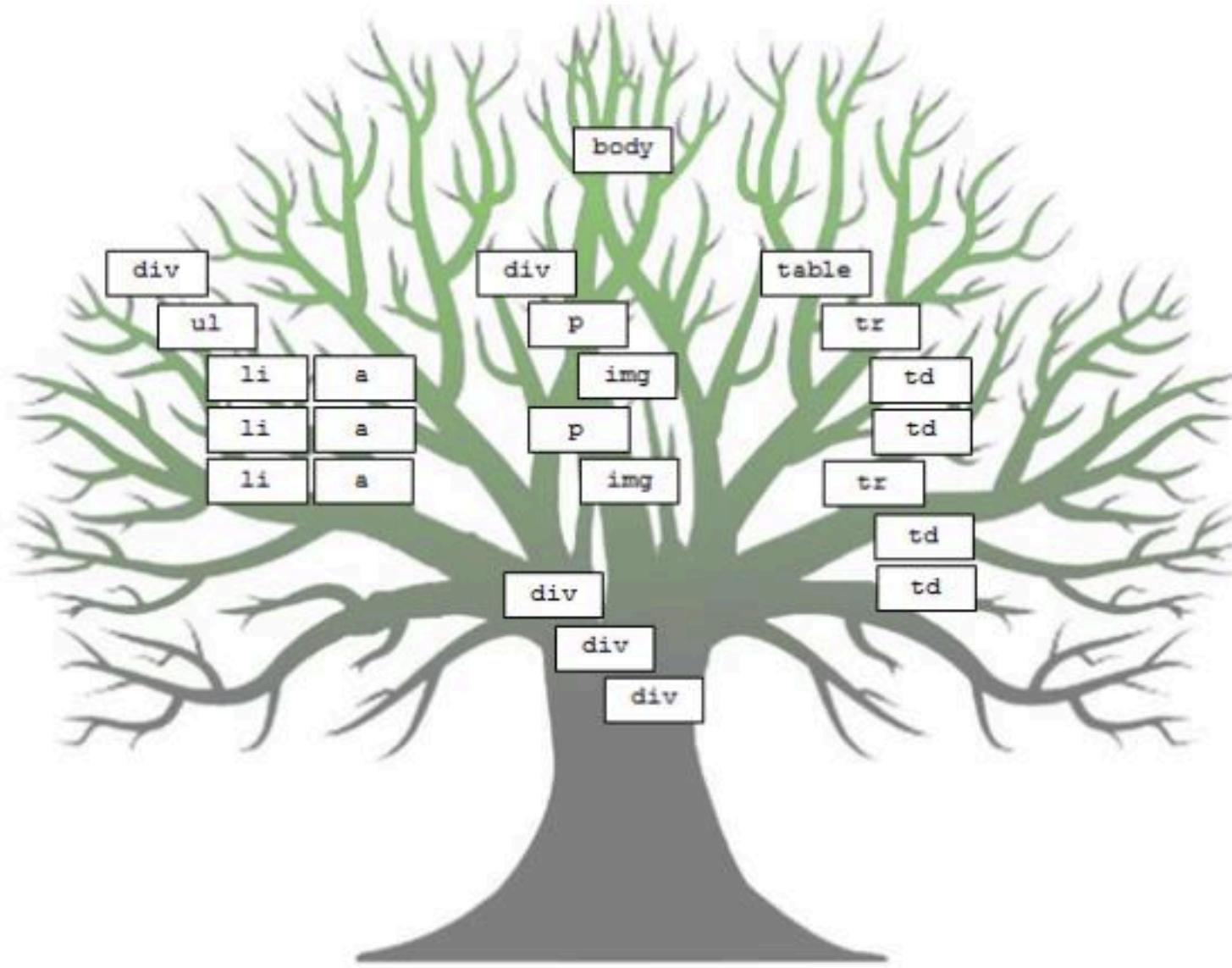


Random expansion from sentence.txt grammar for symbol "<s>"

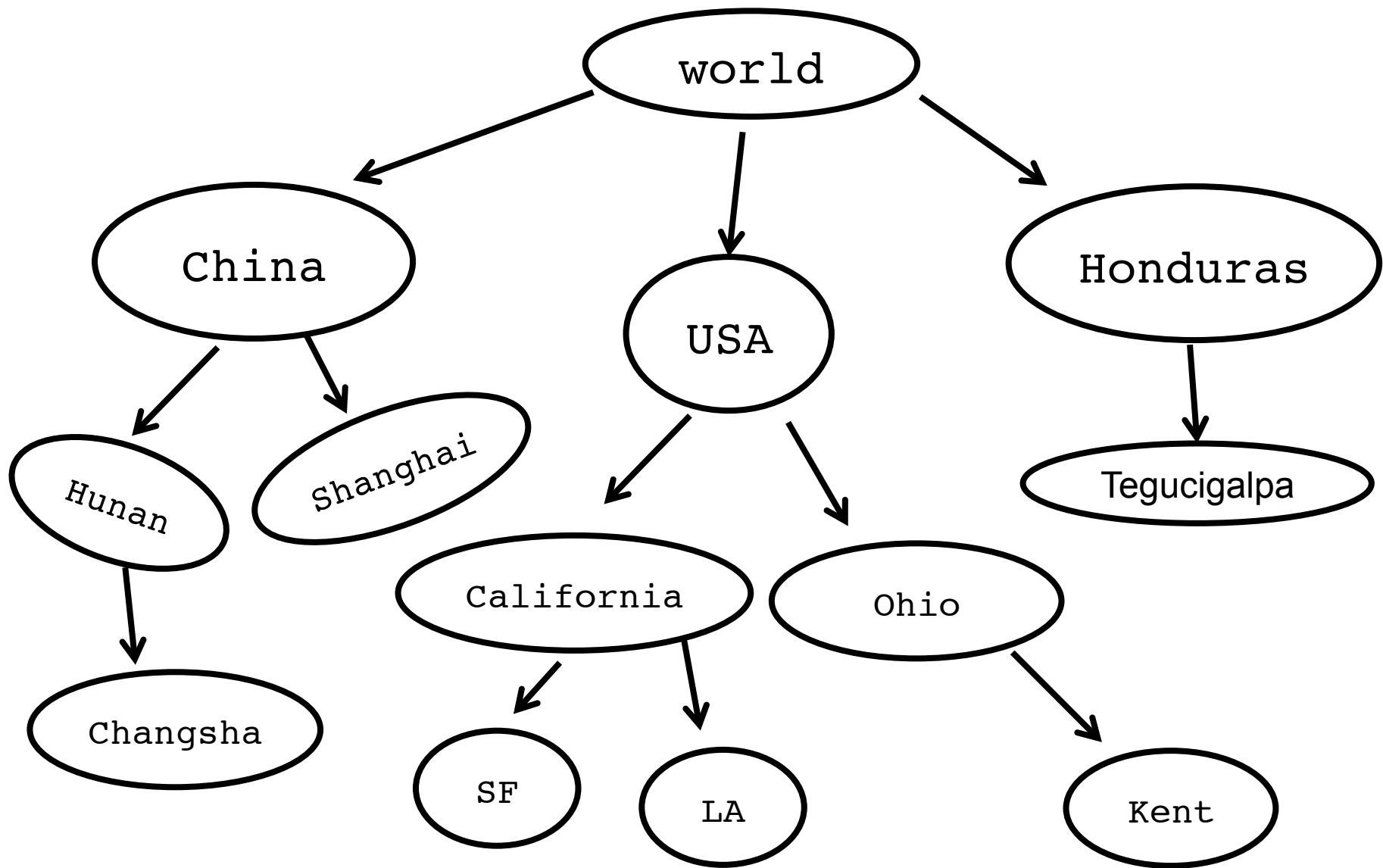
Animal Tree



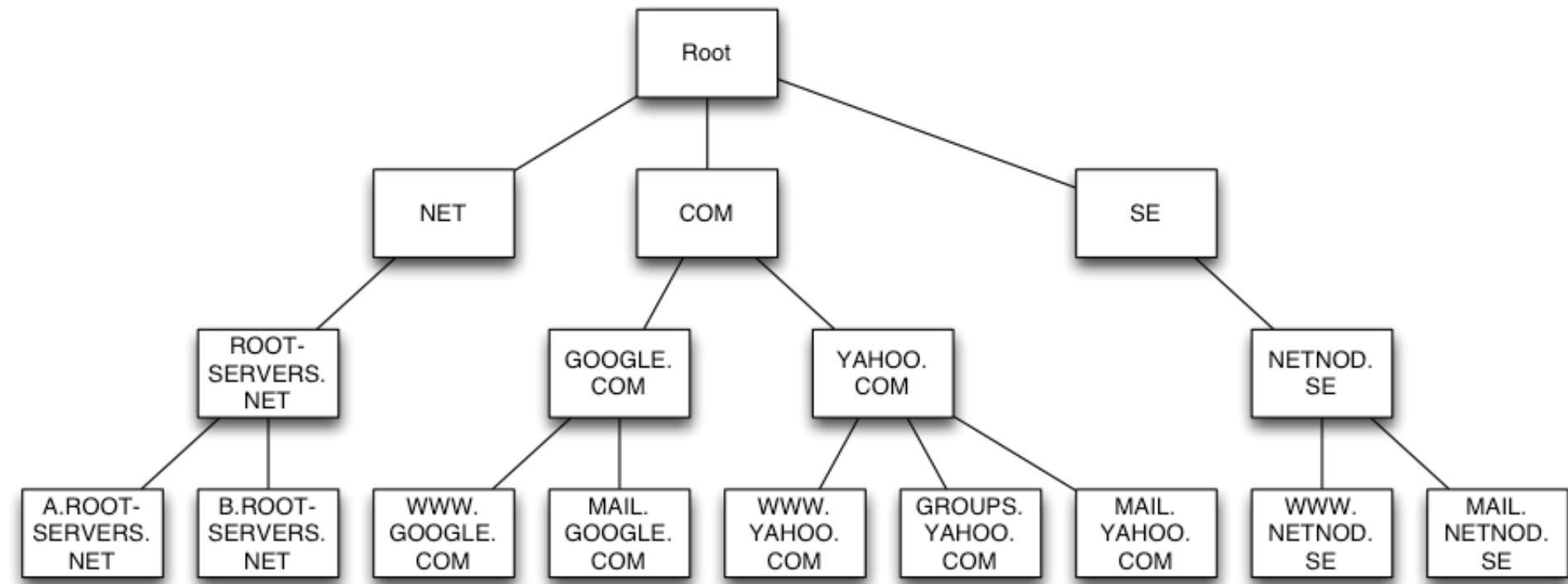
Websites



Hierarchies

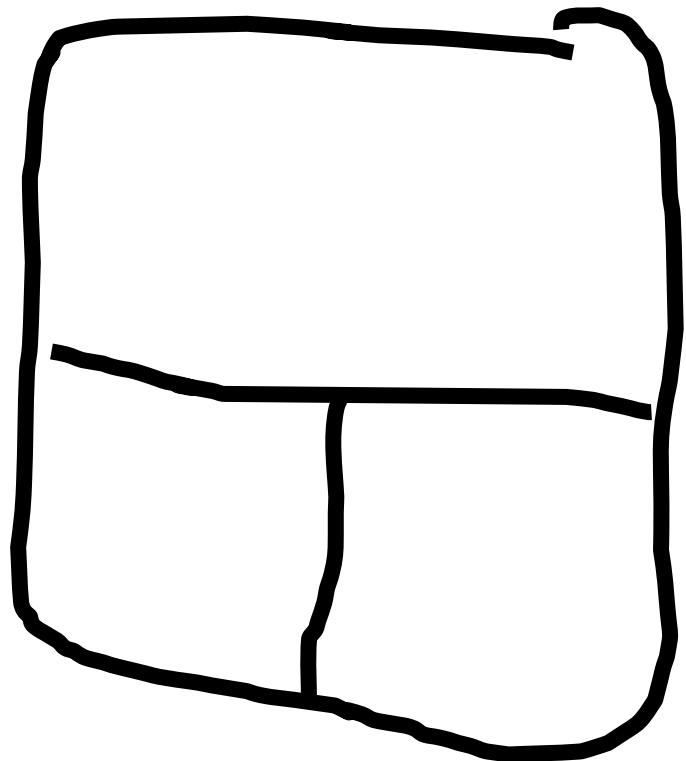


Internet Domains



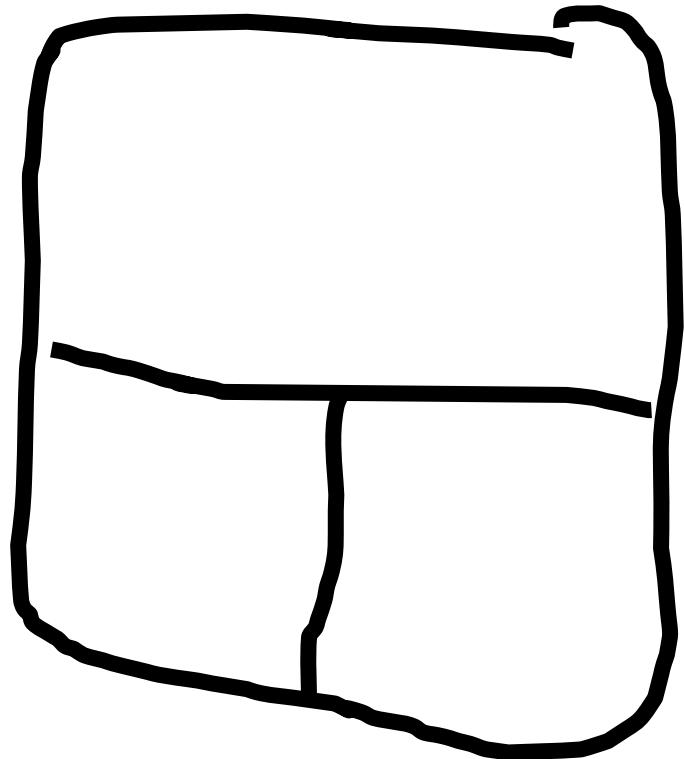
We have seen: Recursion on Trees

But How About Making a Tree Variable?

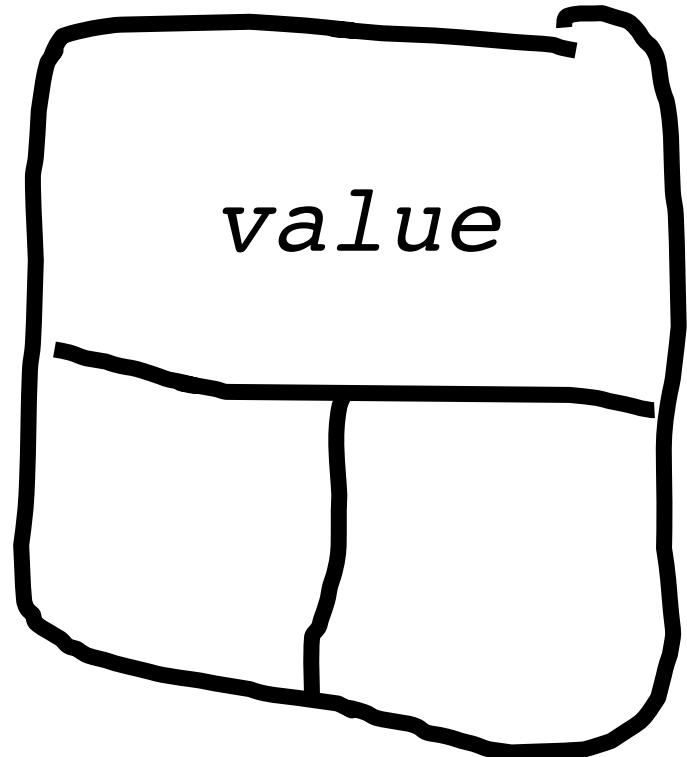




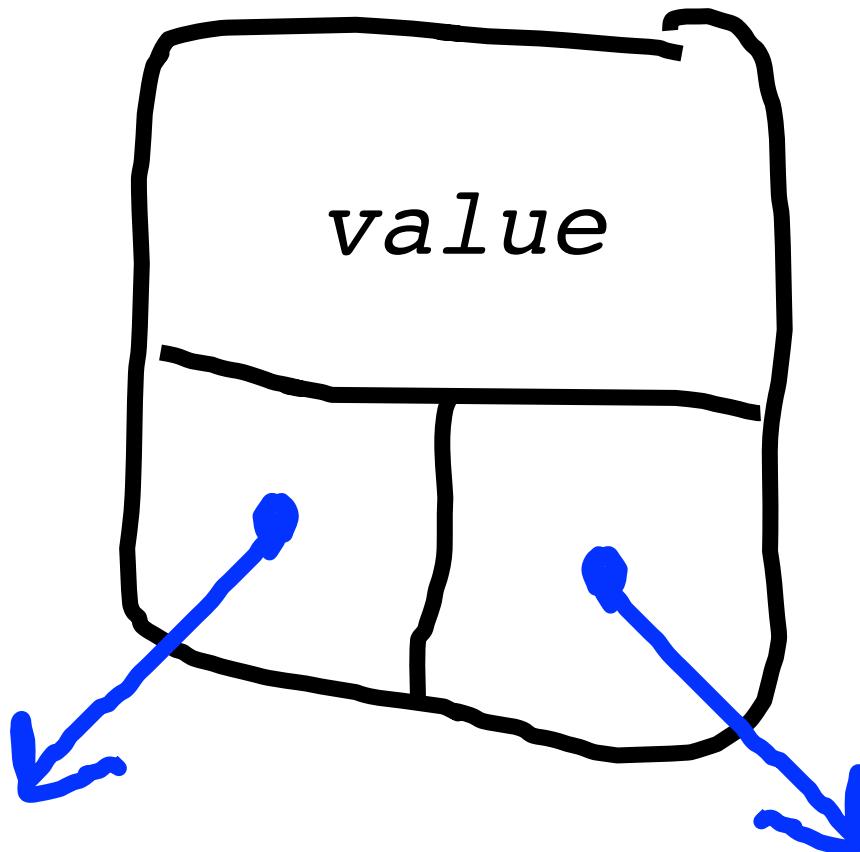
Binary Tree:

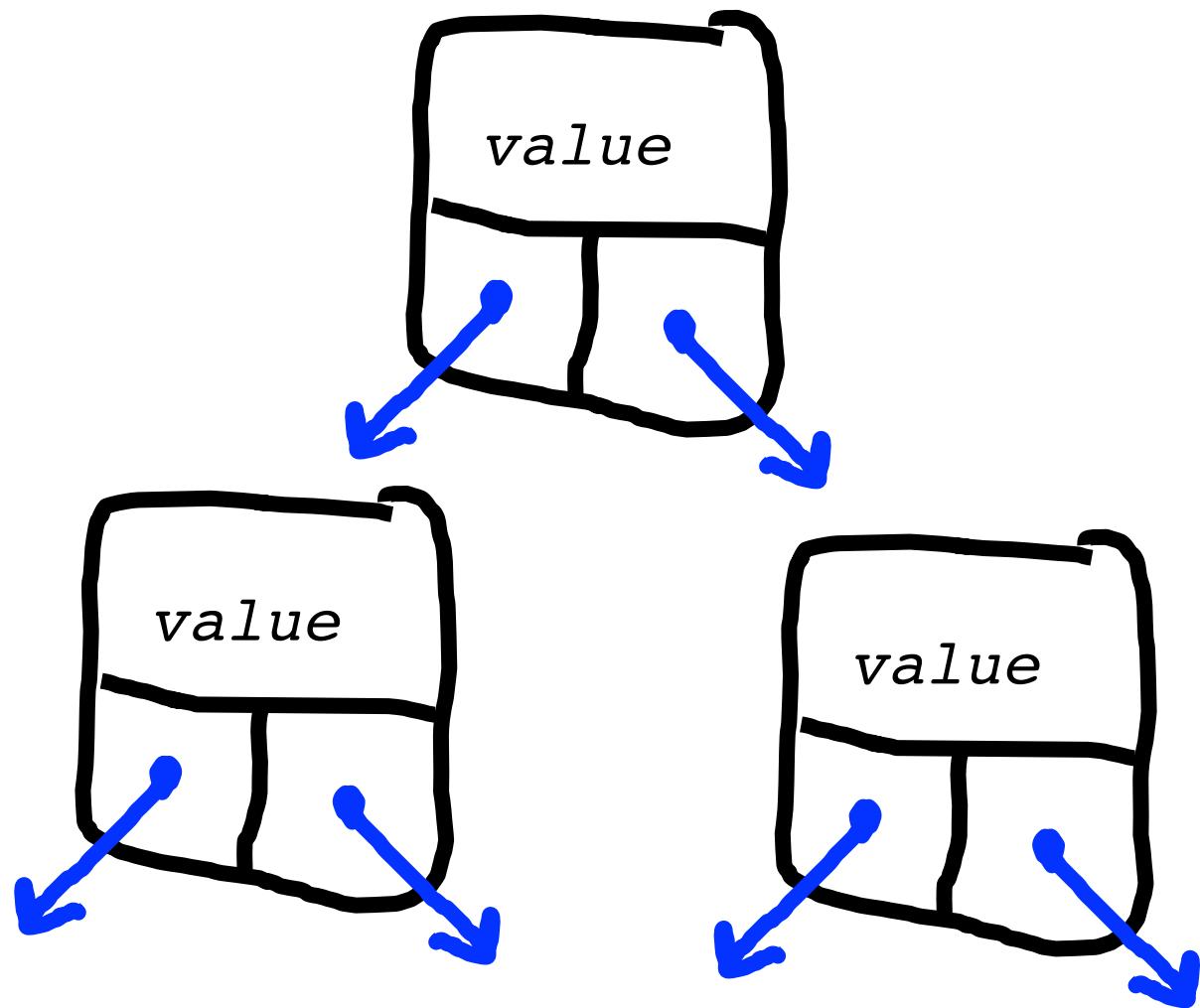


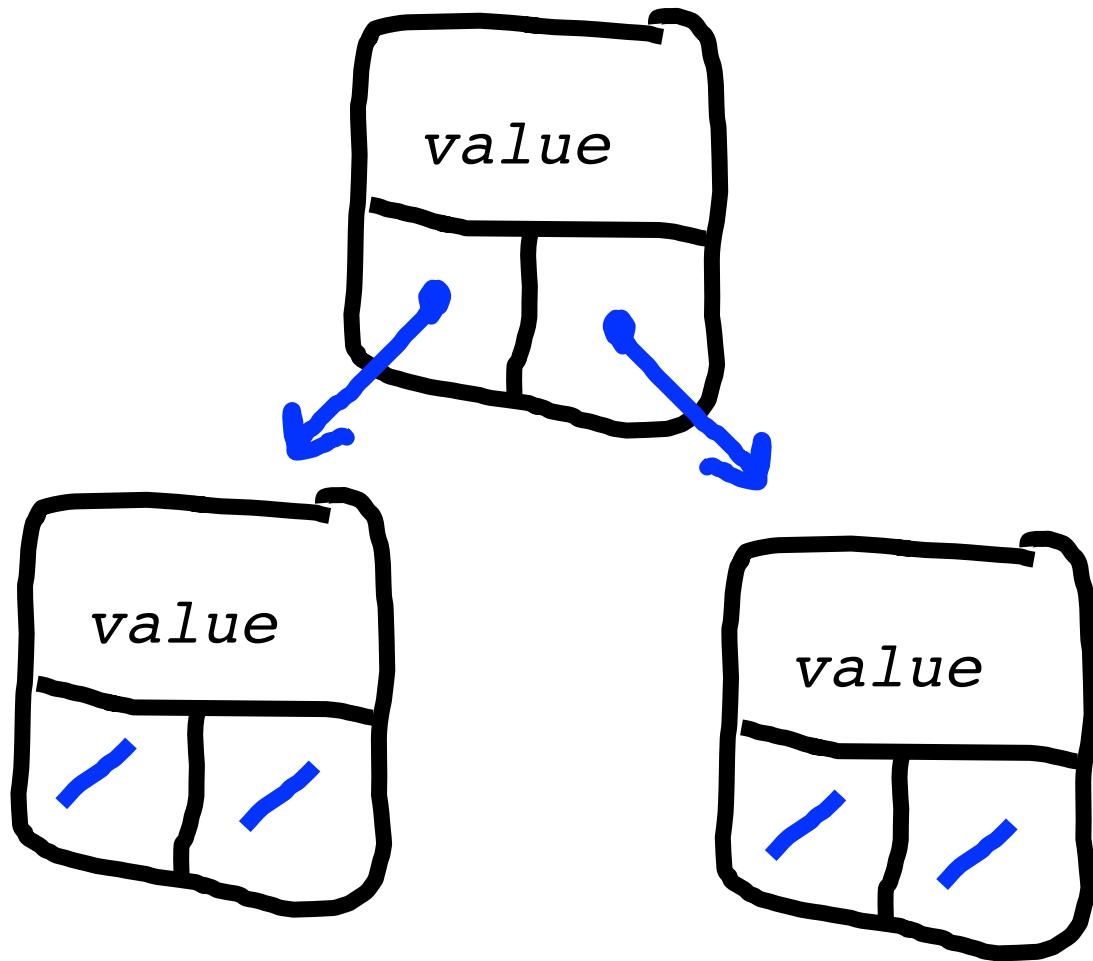
Binary Tree:



Binary Tree:



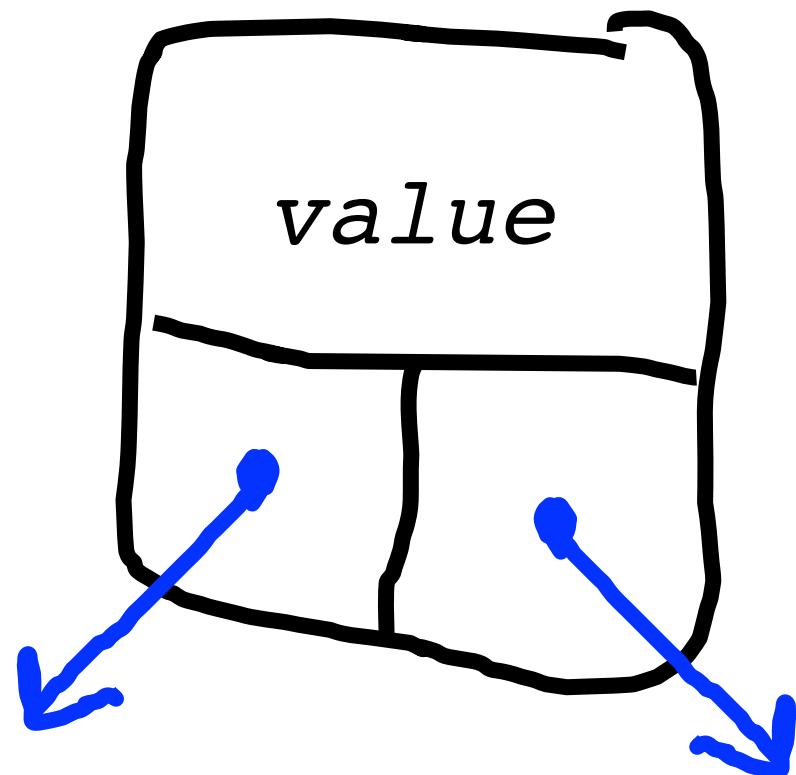




Most Important Slide

Binary Tree

```
struct Tree {  
    string value;  
    Tree * left;  
    Tree * right;  
};
```

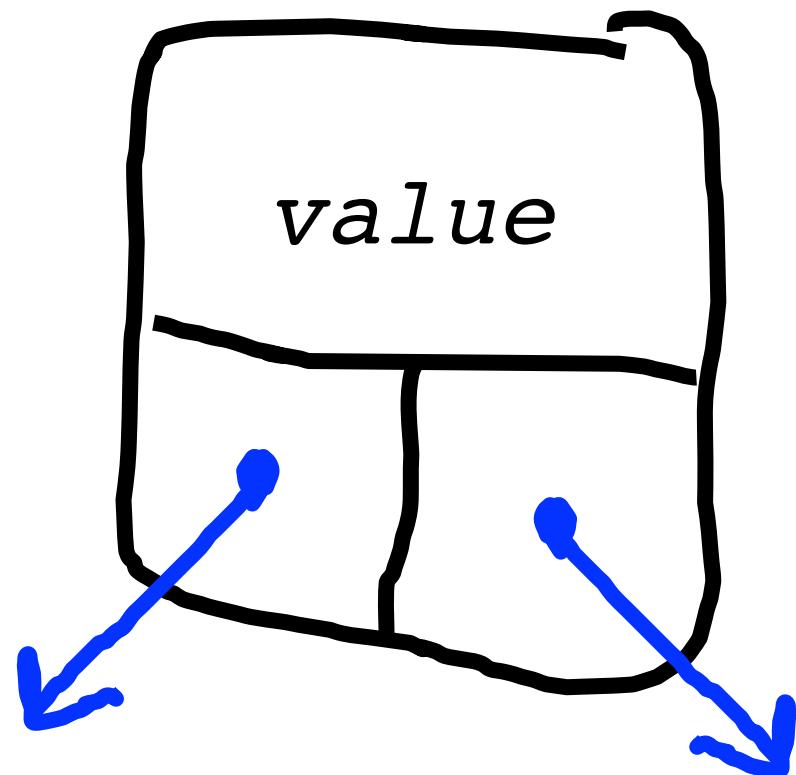


Does value have to be a string?

No

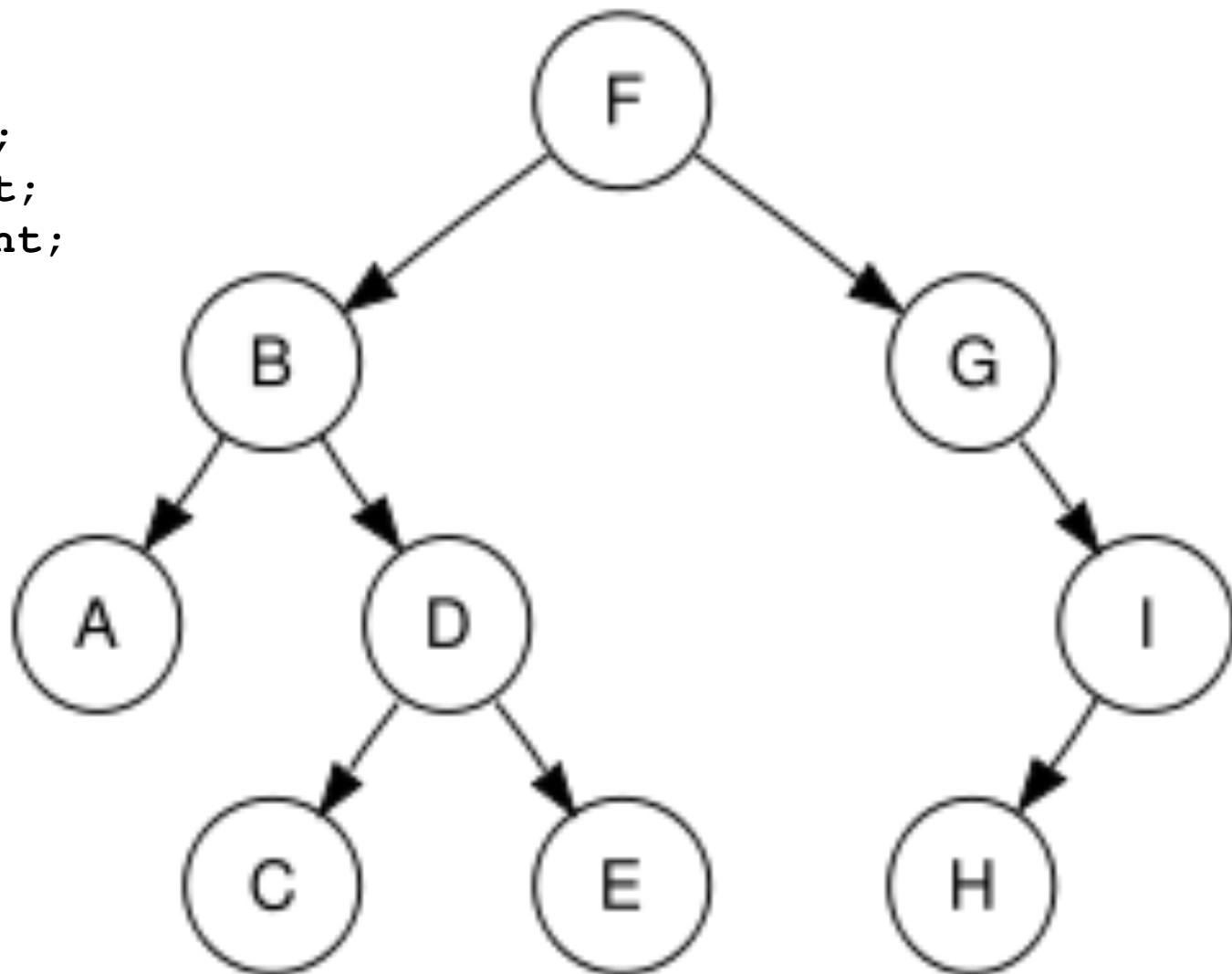
Binary Tree

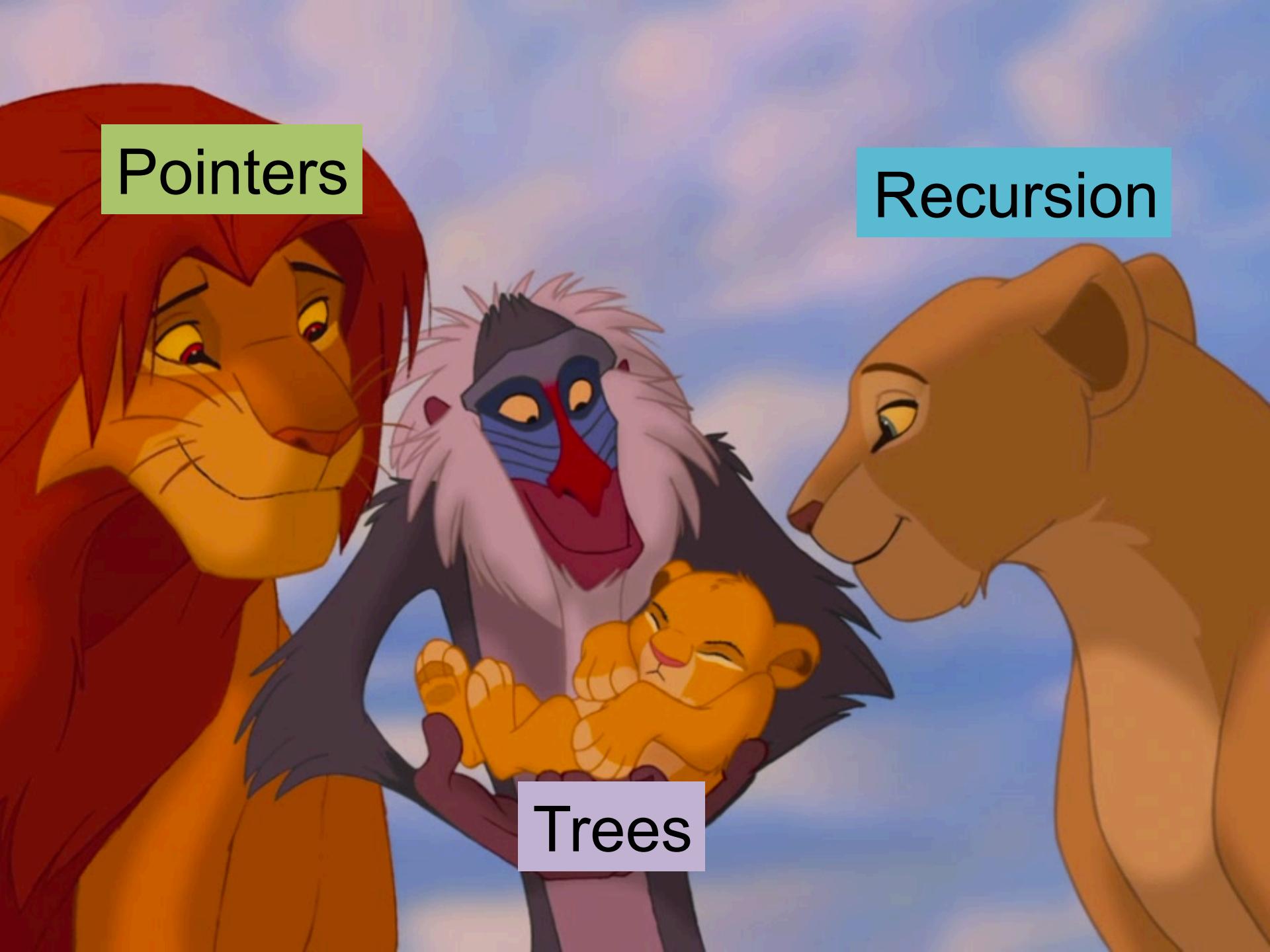
```
struct Tree {  
    string value;  
    Tree * left;  
    Tree * right;  
};
```



Example Binary Tree

```
struct Tree {  
    char value;  
    Tree * left;  
    Tree * right;  
};
```



A scene from Disney's The Lion King. Simba, the young lion cub, is being held by his father Mufasa. Timon and Pumbaa are standing behind them, looking down at Simba. Nala is partially visible on the right side of the frame. The background shows a sunset or sunrise over a savanna landscape.

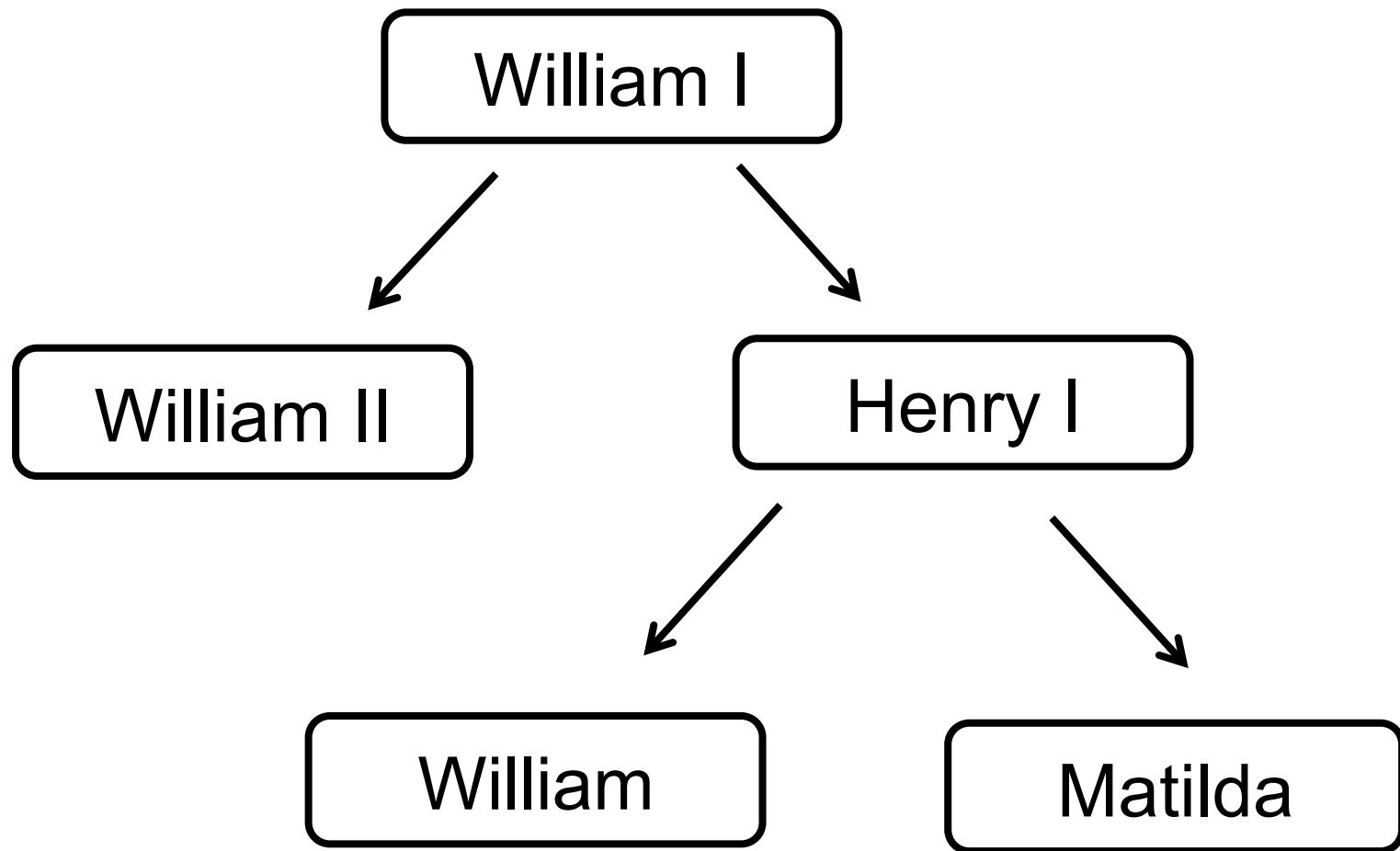
Pointers

Recursion

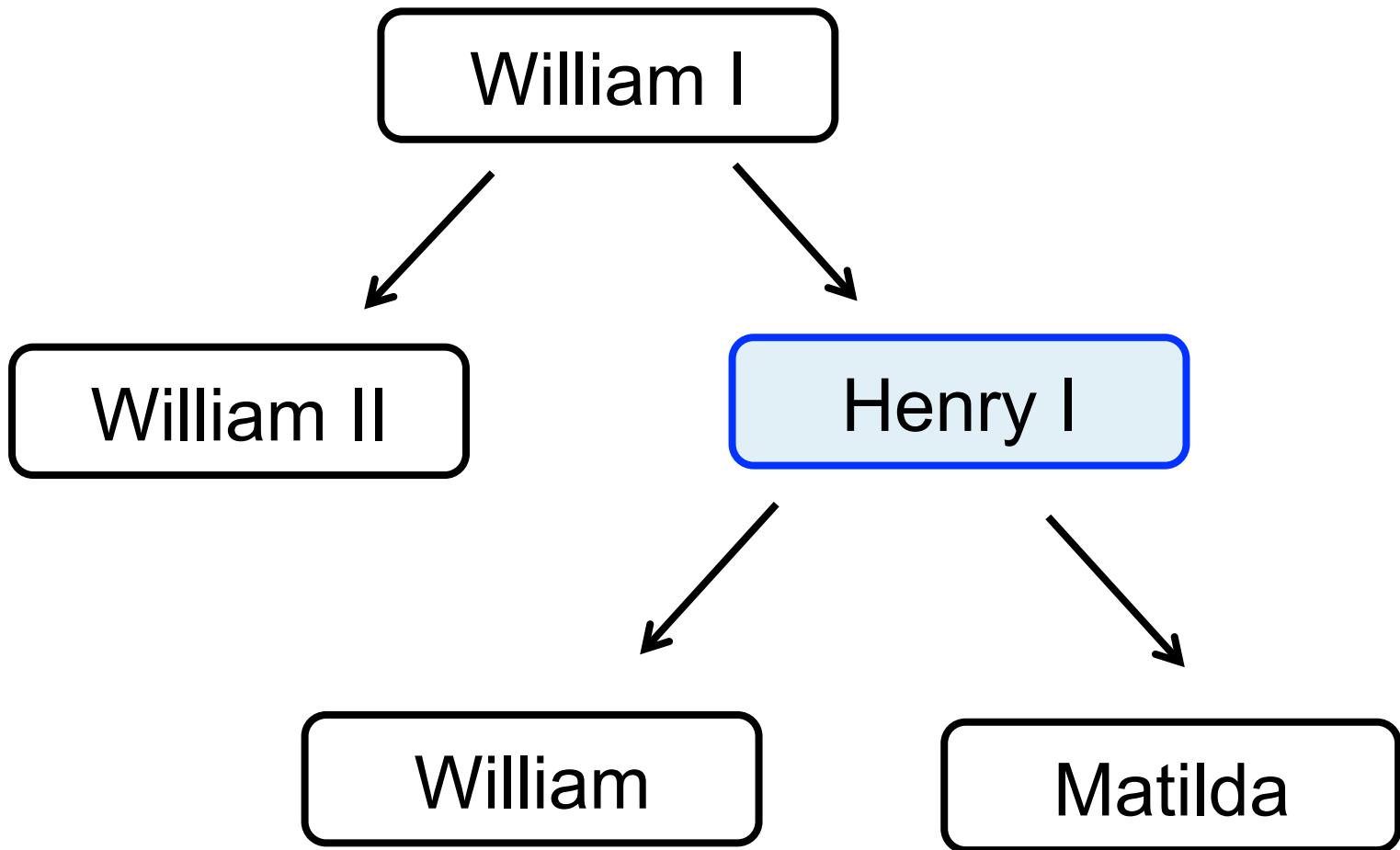
Trees

Terminology

Terminology

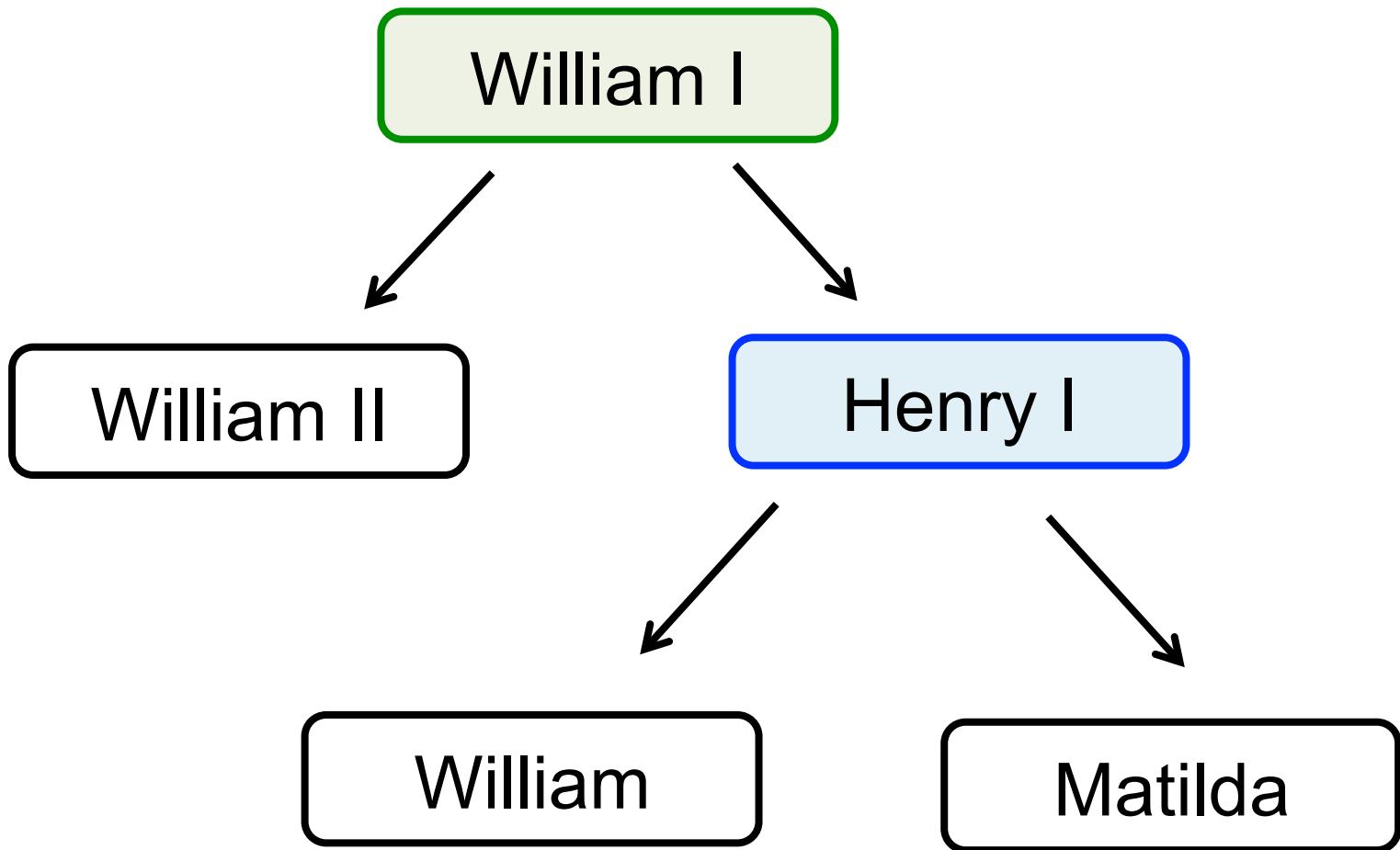


Terminology



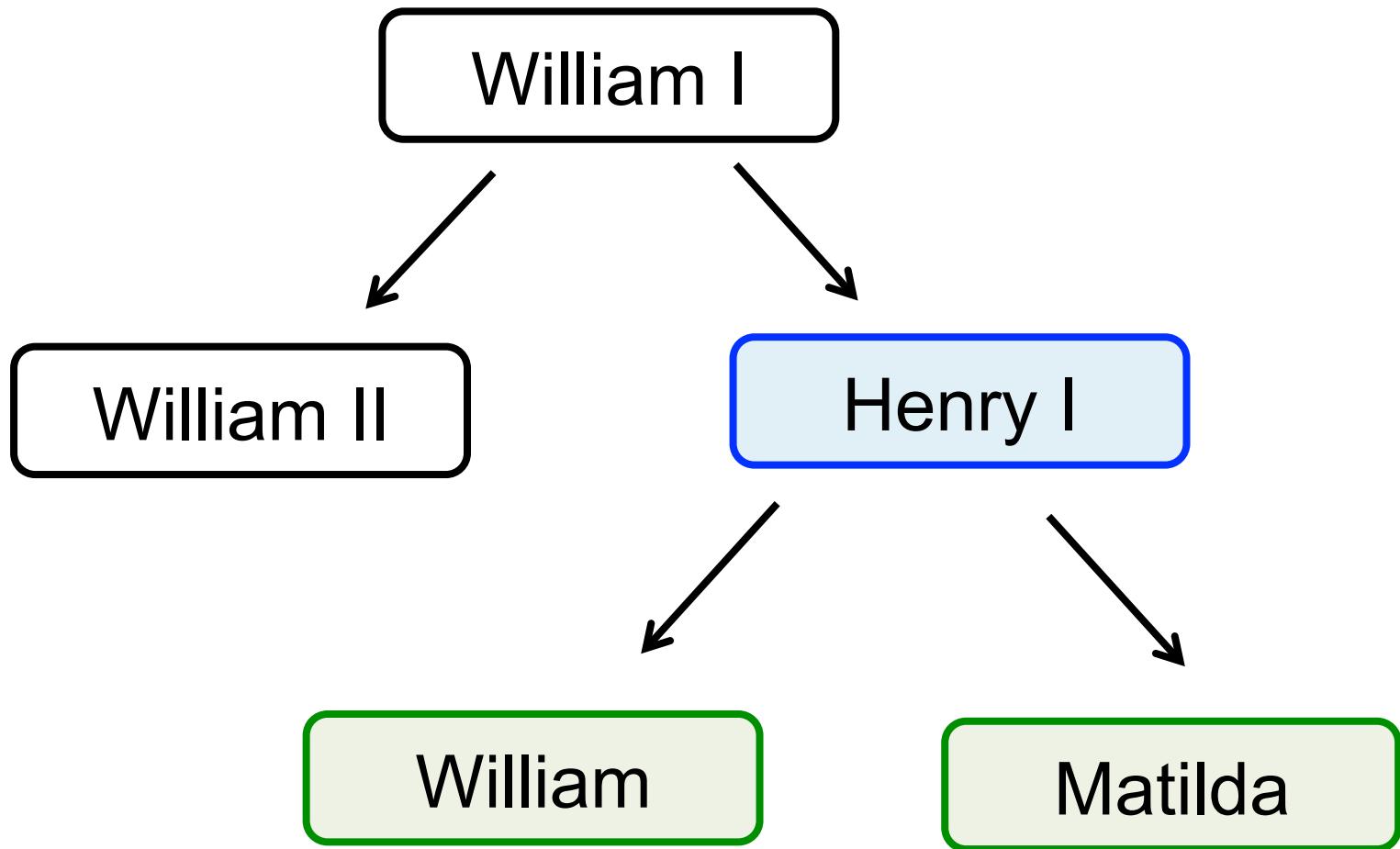
We call a single element a “**Node**” or “**Tree**”

Terminology: Parent



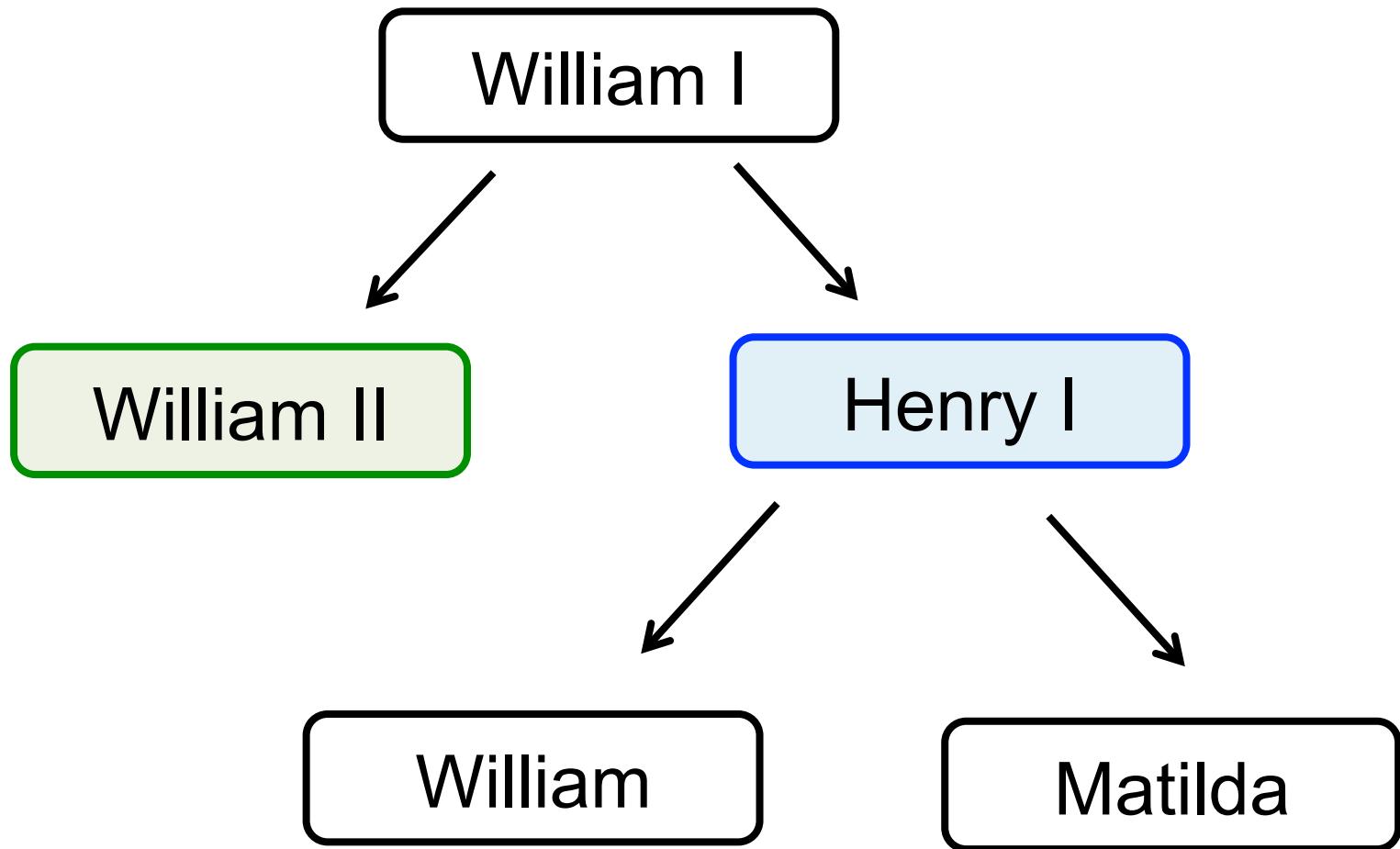
We call the Tree that points to us our “**Parent**”

Terminology: Children



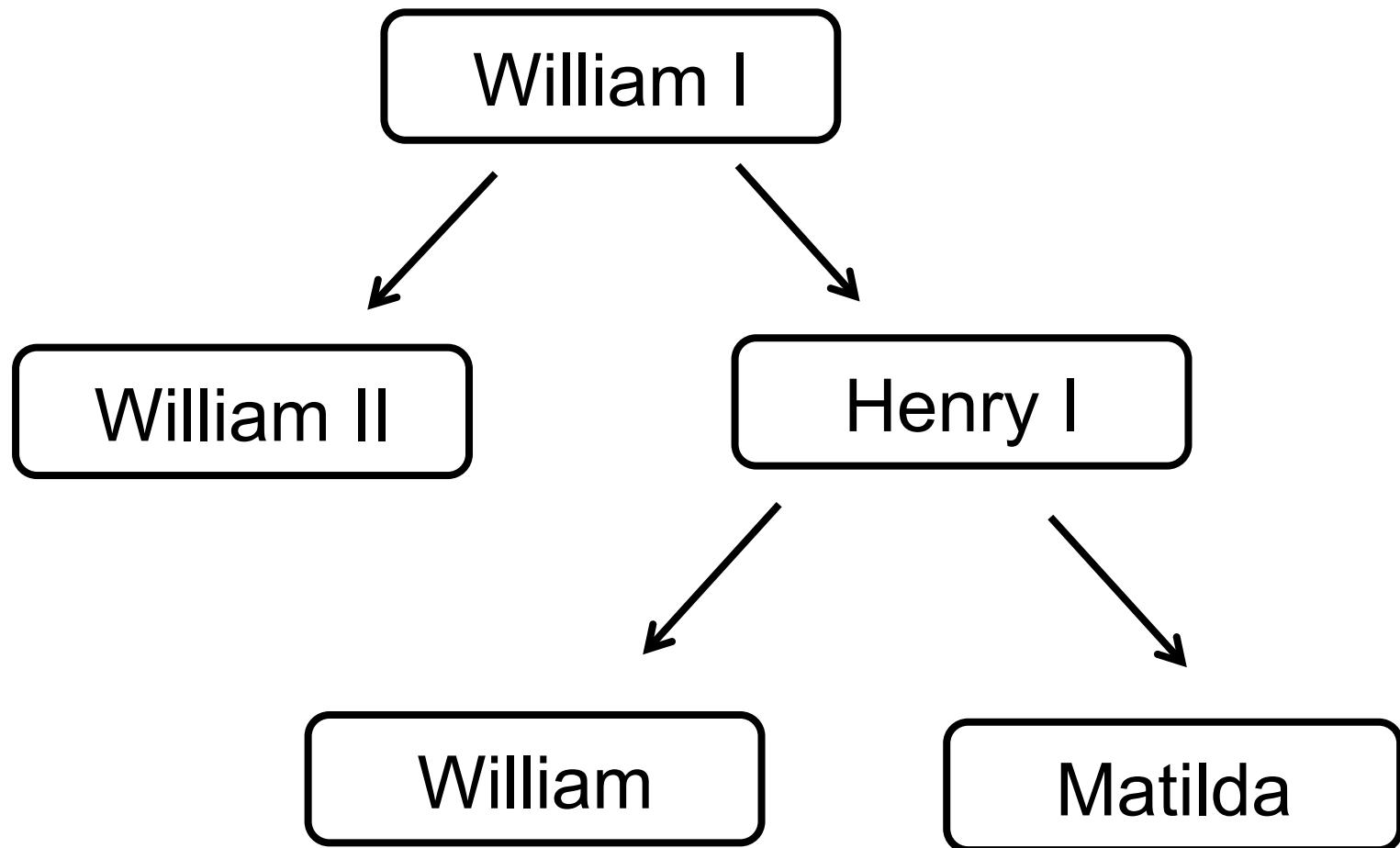
We call the nodes that we point to our “**Children**”

Terminology: Siblings

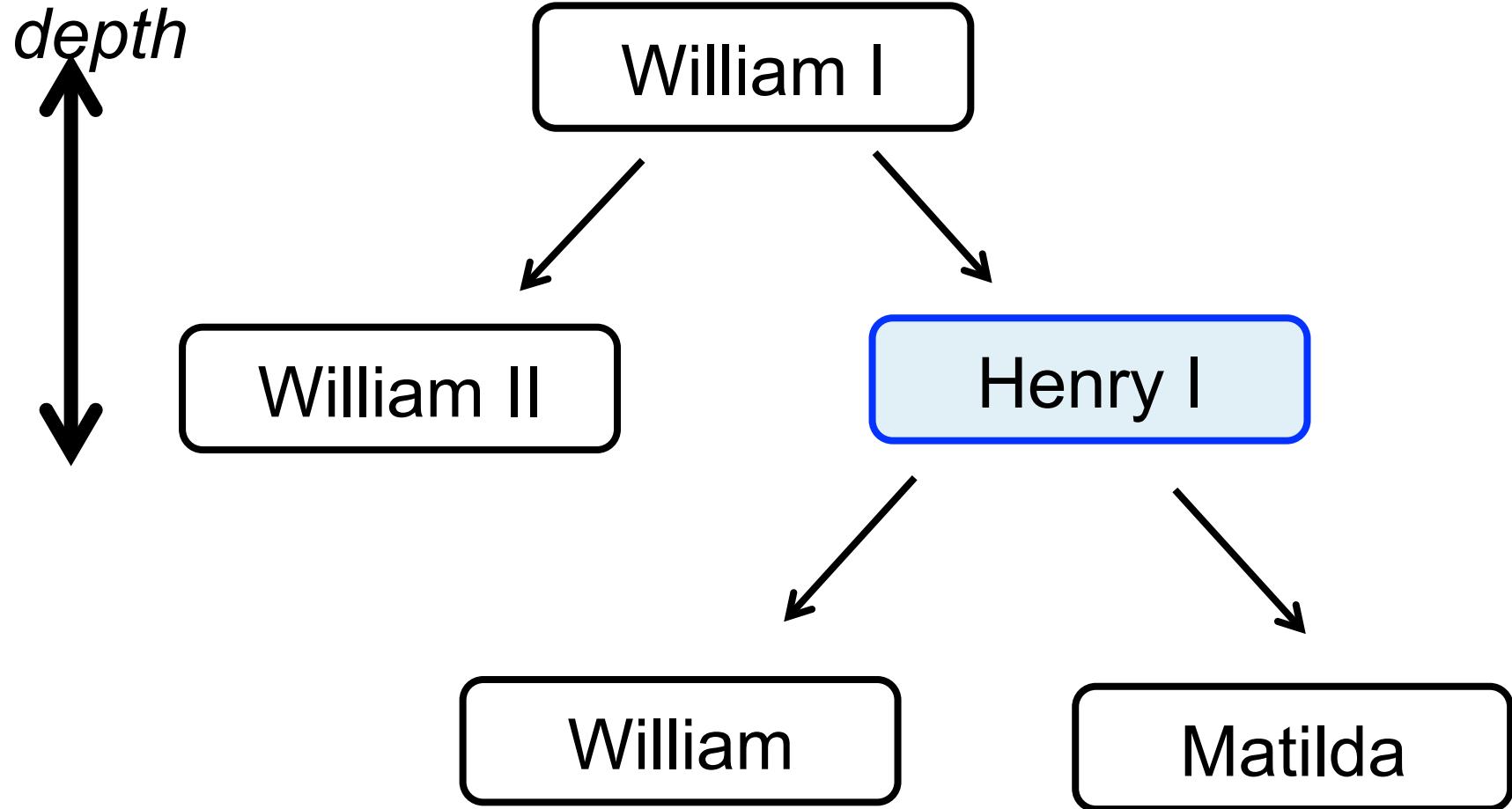


We call nodes that share the same parent “**Siblings**”

Terminology

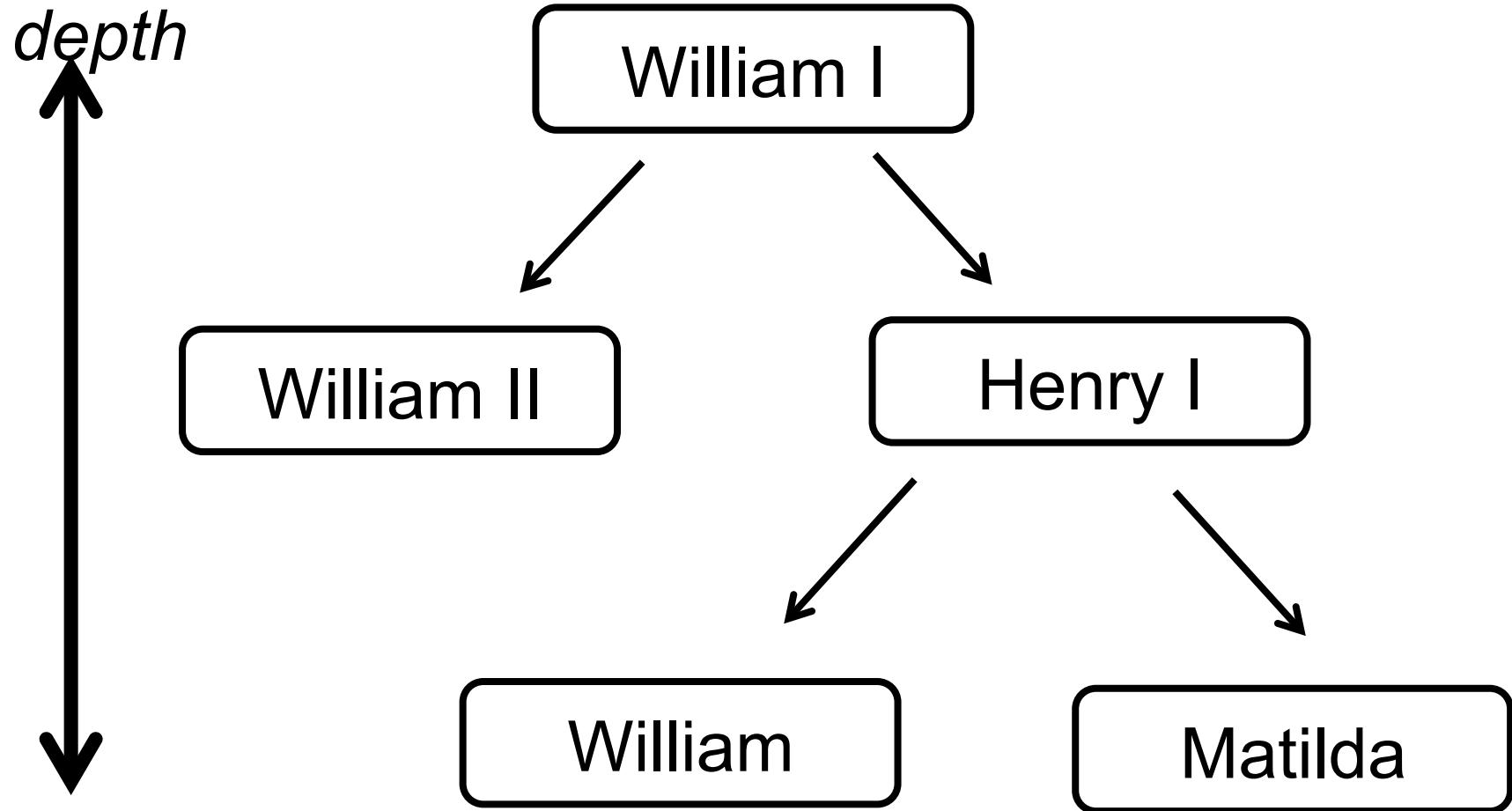


Terminology



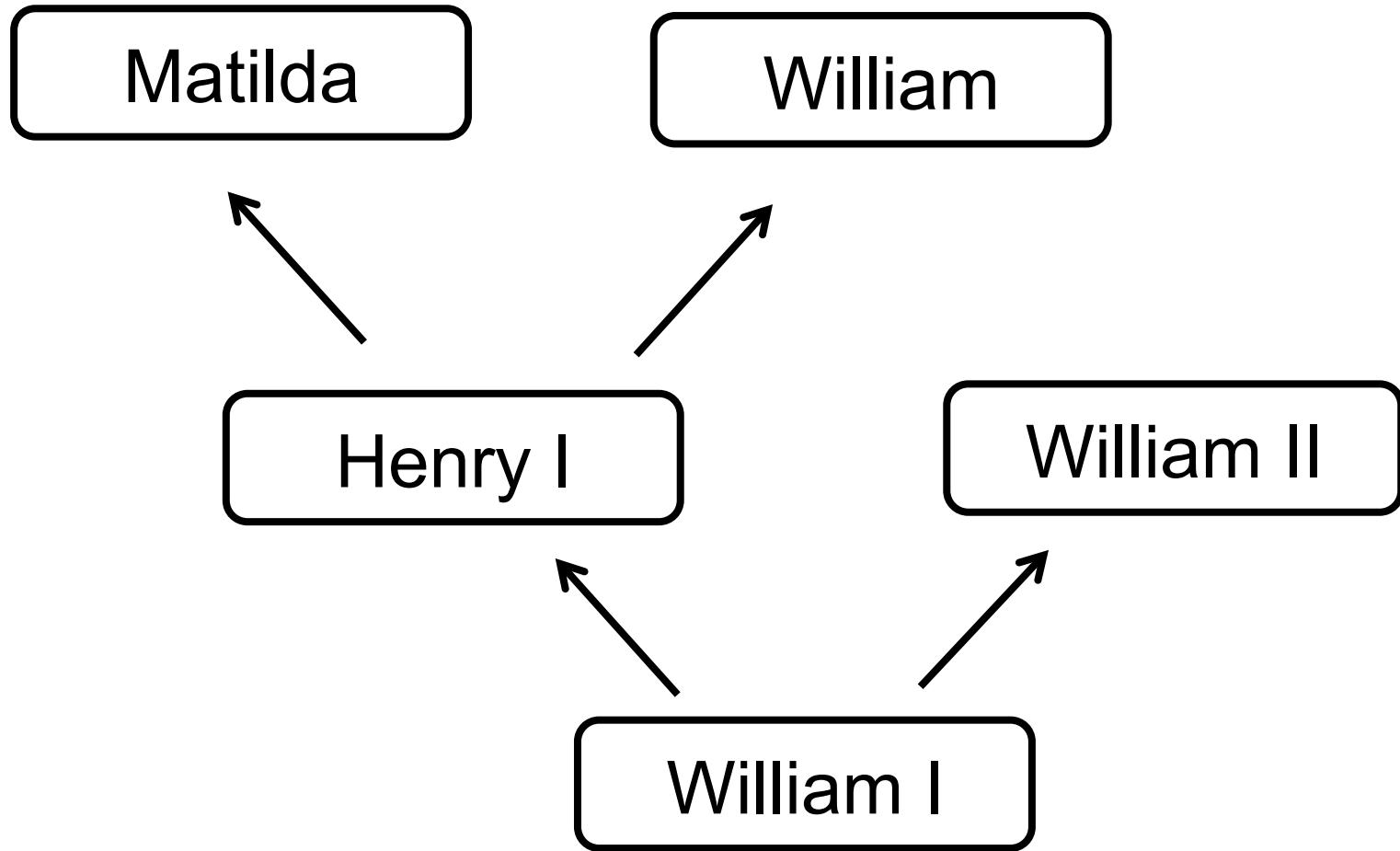
The depth of a node is the number of links from the root.

Terminology

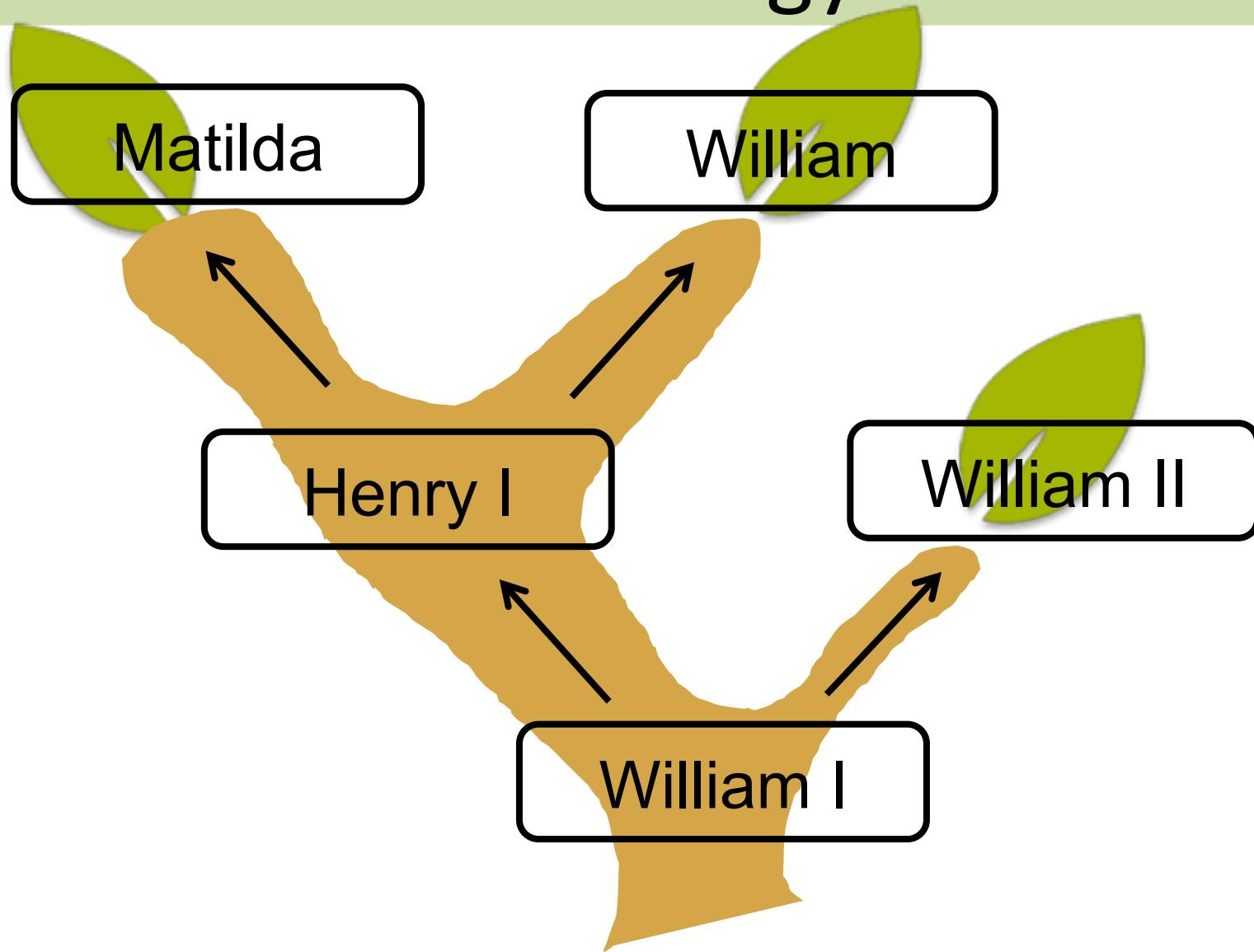


The depth of a tree is the maximum number of links from the root.

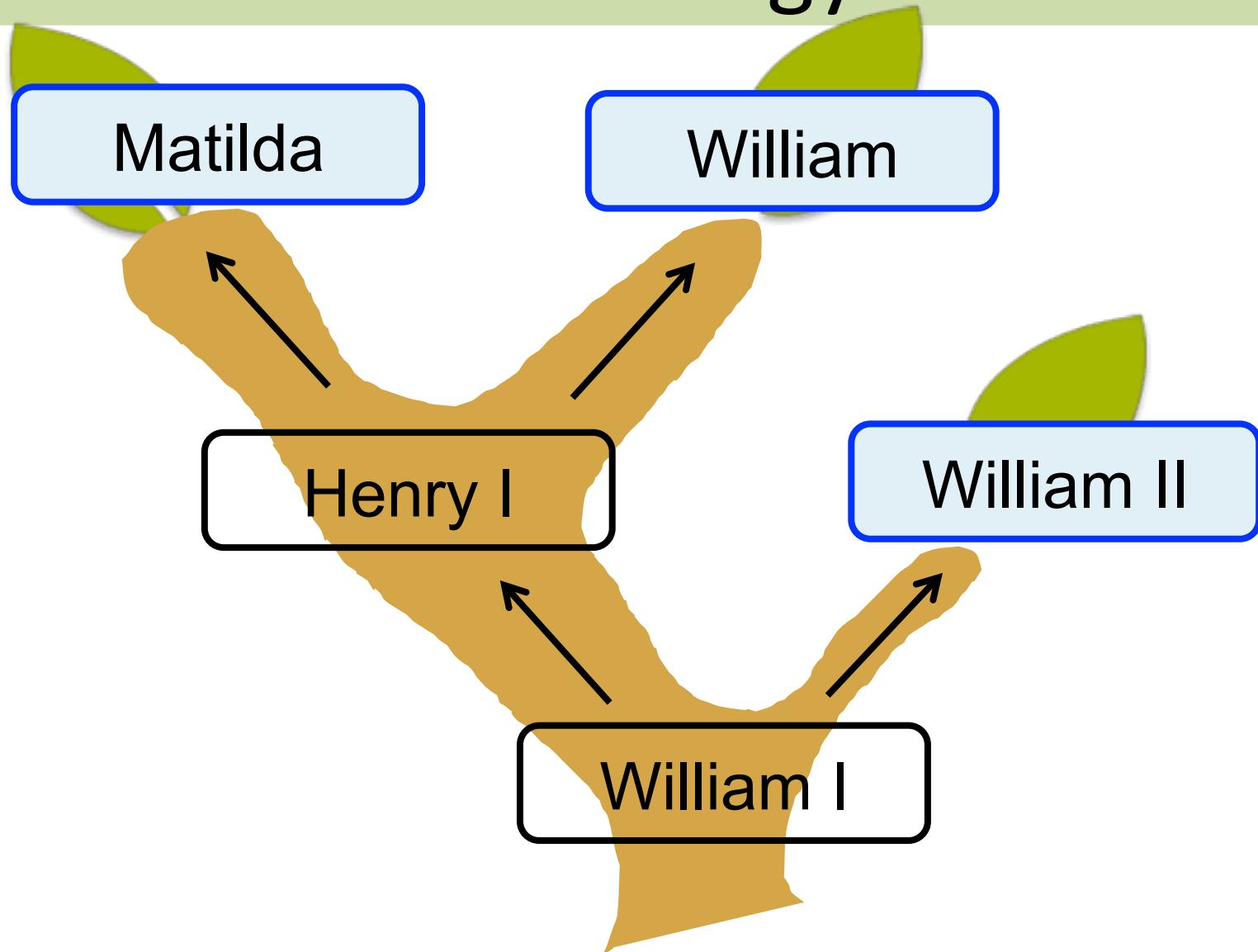
Terminology



Terminology

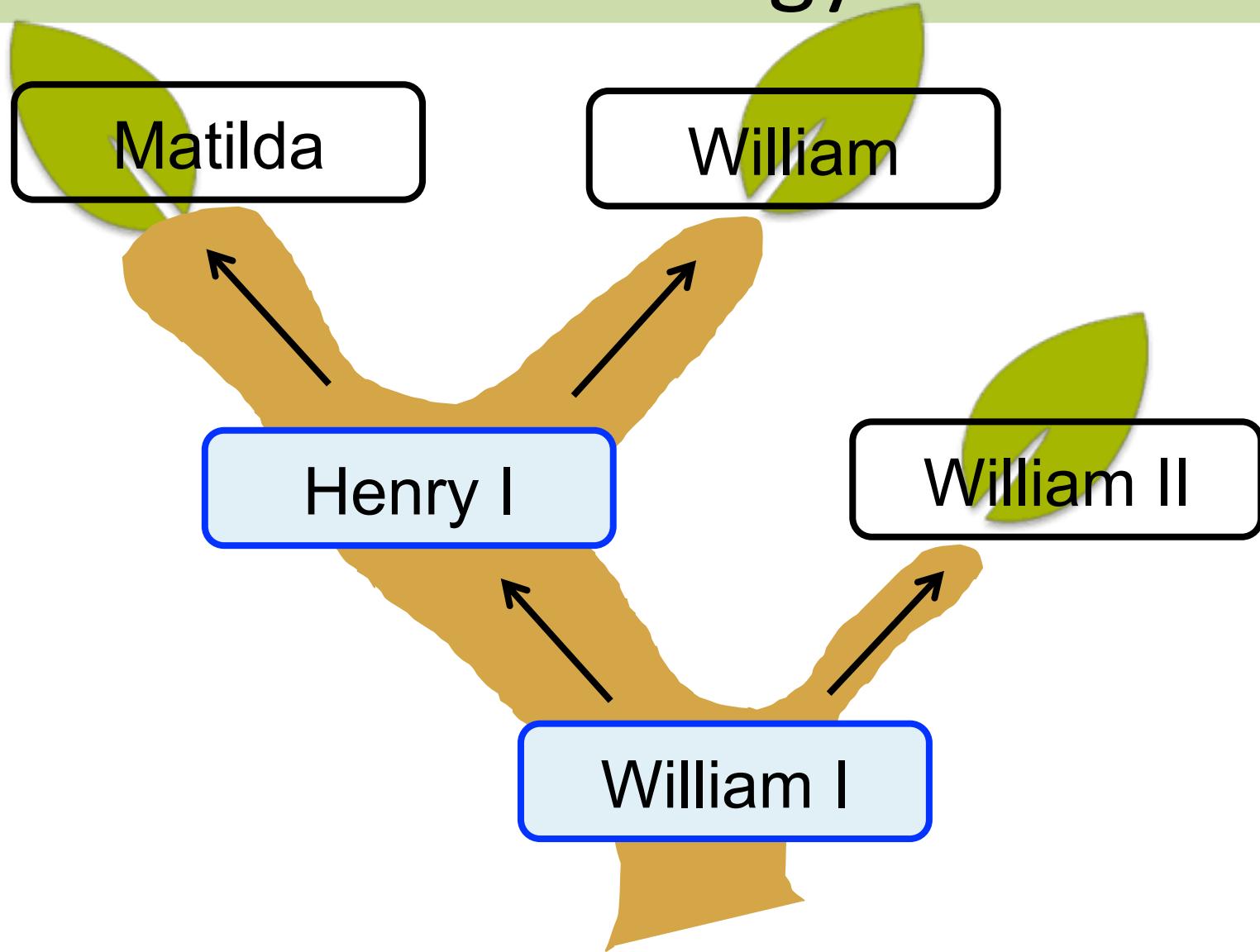


Terminology



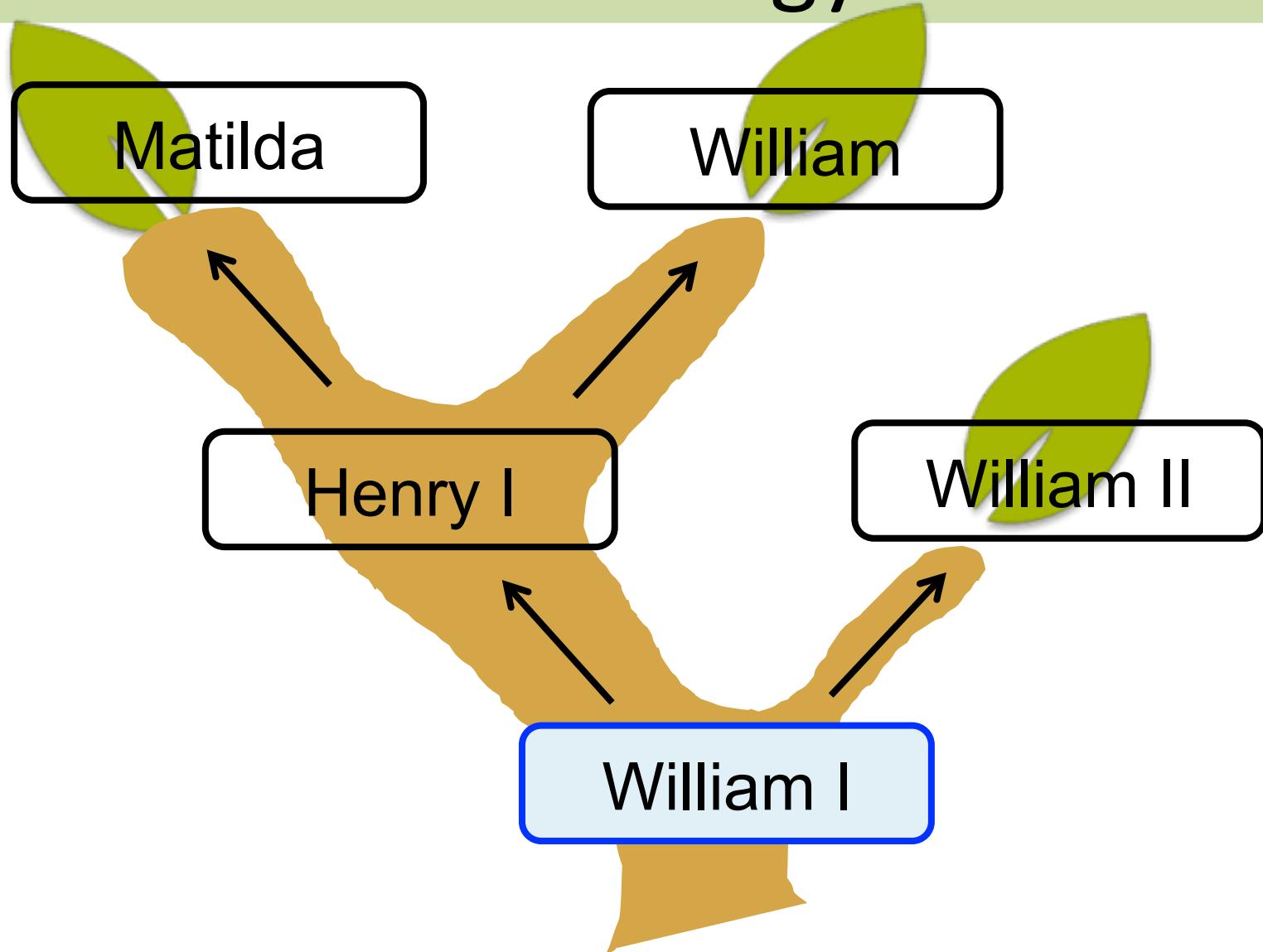
We call nodes that have no children “**Leaves**”

Terminology



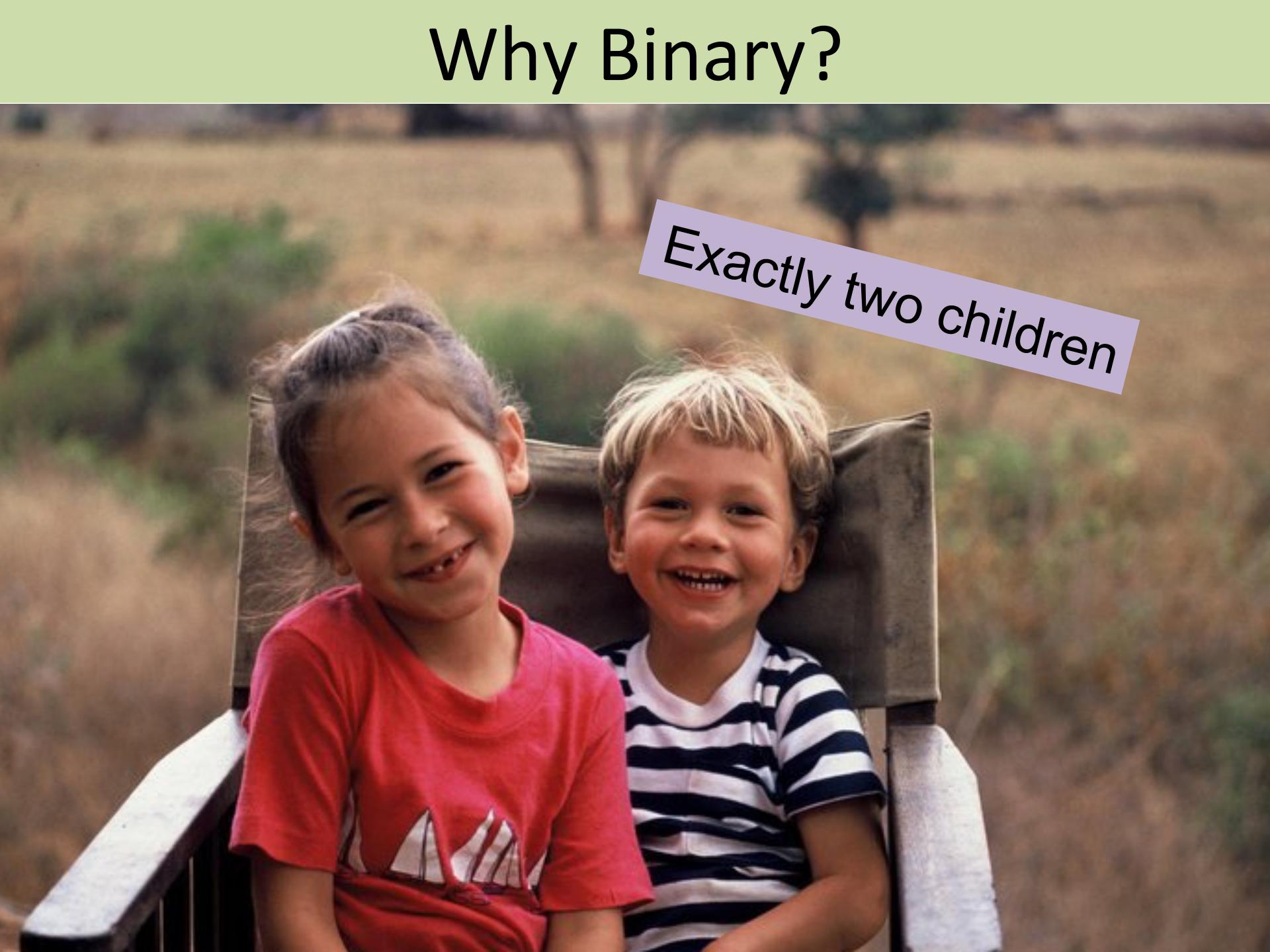
We call nodes that have children “**Inner Nodes**”

Terminology



We call the node without parents the “**Root**”

Why Binary?

A photograph of two young children, a girl and a boy, sitting side-by-side on a wooden bench outdoors. They are both smiling. The girl has dark hair tied back and is wearing a red t-shirt with a white graphic. The boy has blonde hair and is wearing a white and blue striped t-shirt. The background is a blurred landscape of green and brown fields.

Exactly two children

End Terminology

Tree Definition

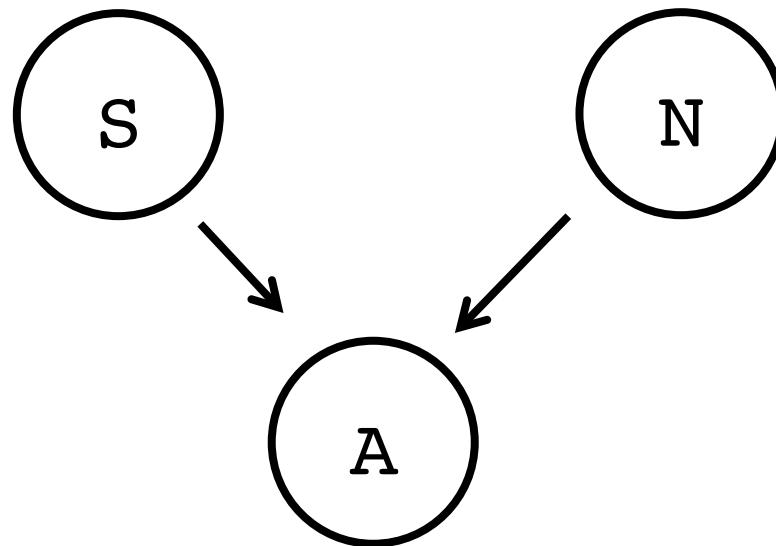


Only One Parent

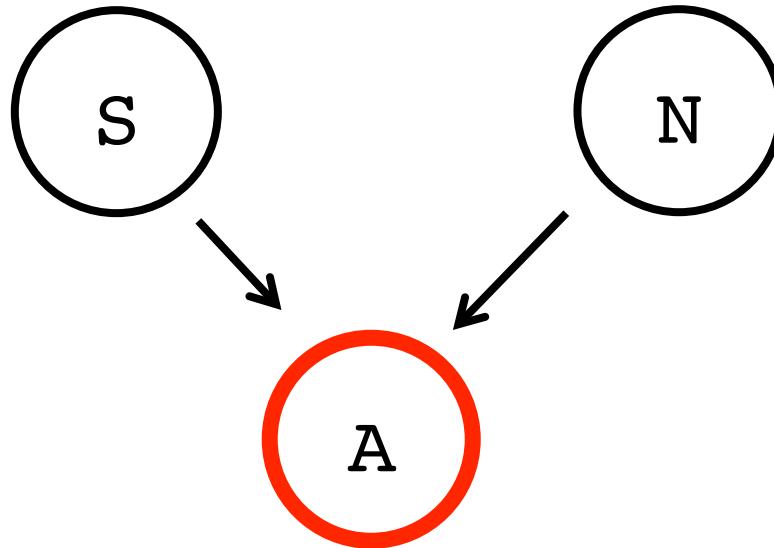


No Cycles

Only One Parent

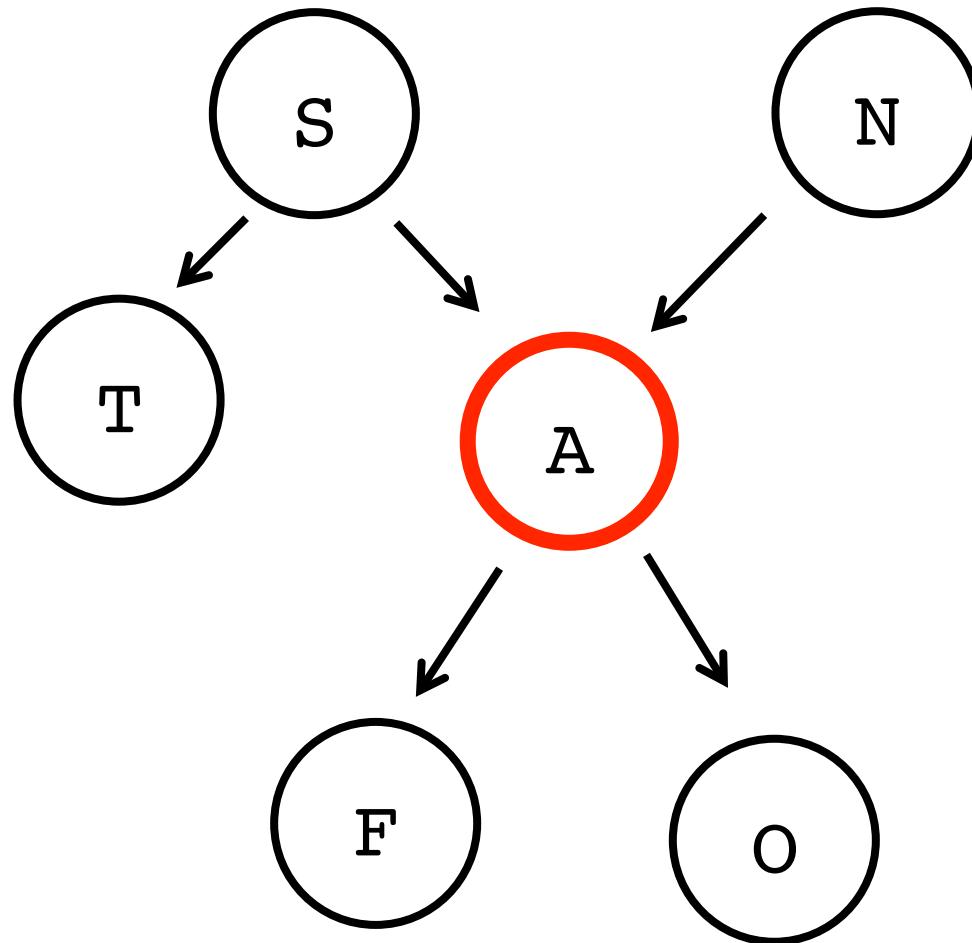


Only One Parent



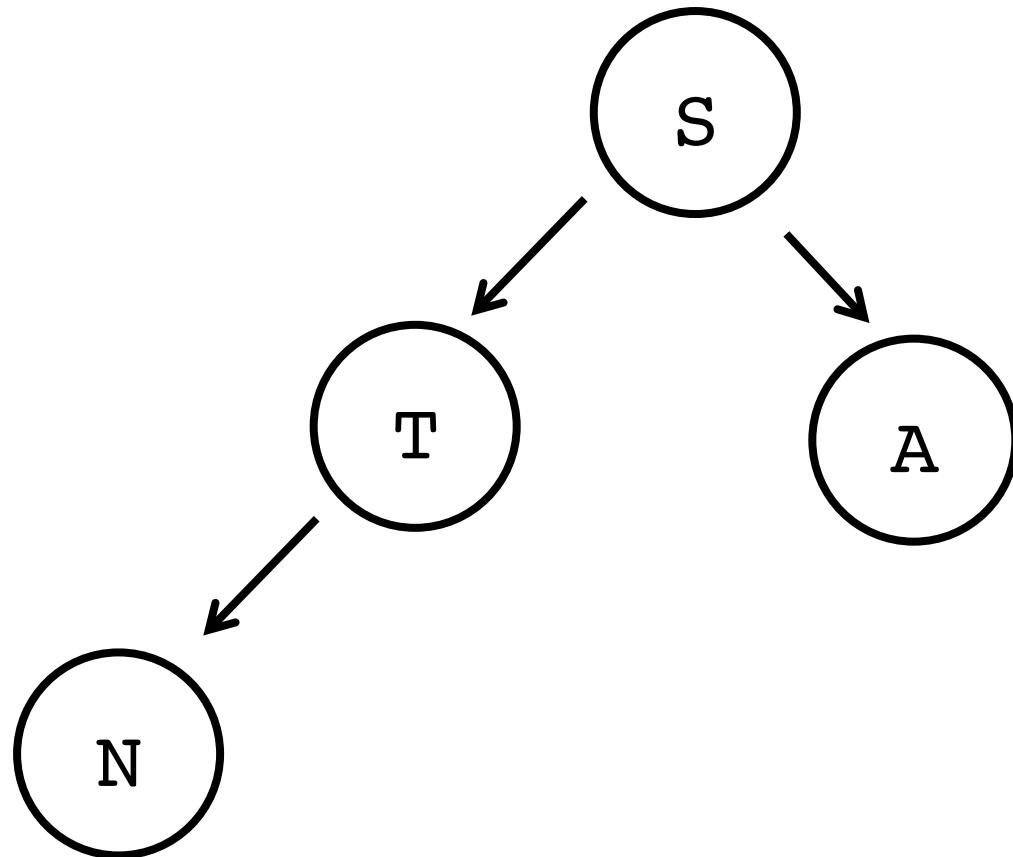
This is not a tree because the red node has two parents

Only One Parent

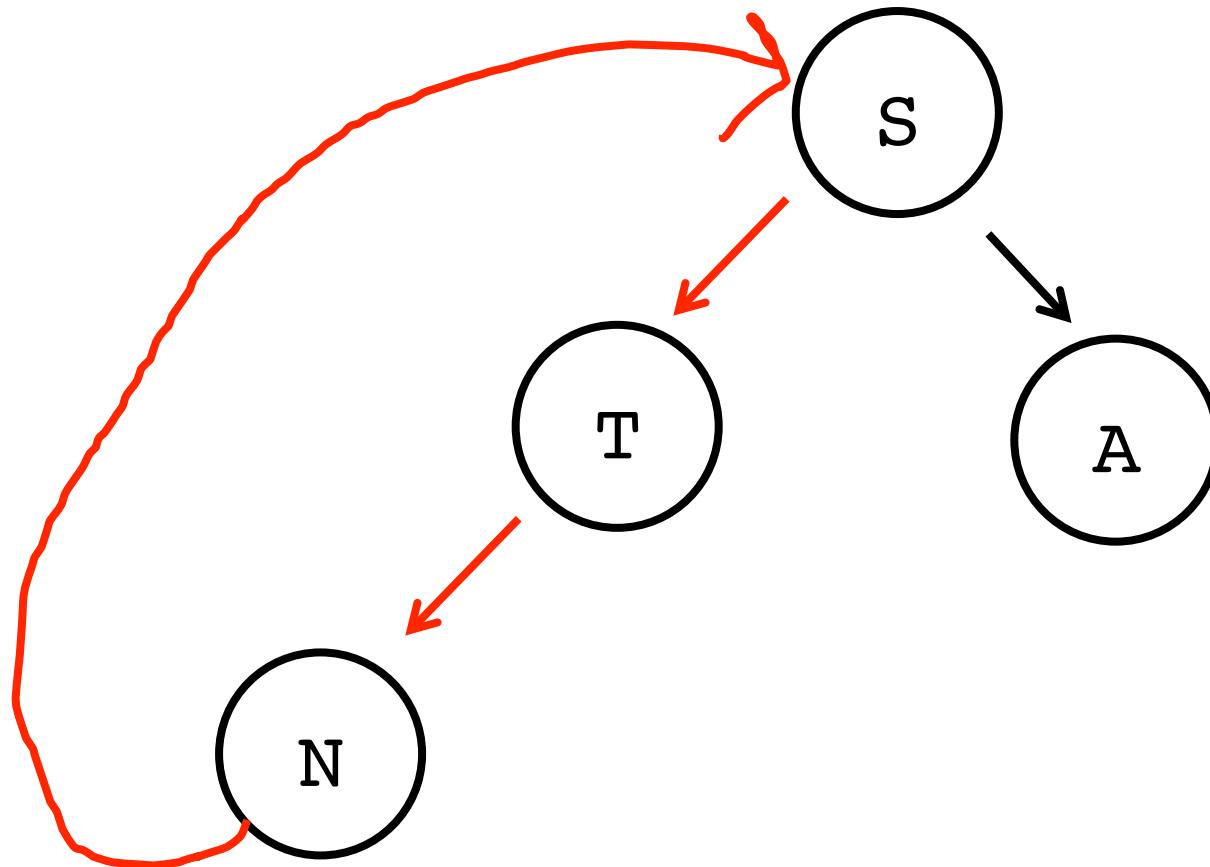


This is not a tree because the red node has two parents

No Cycles

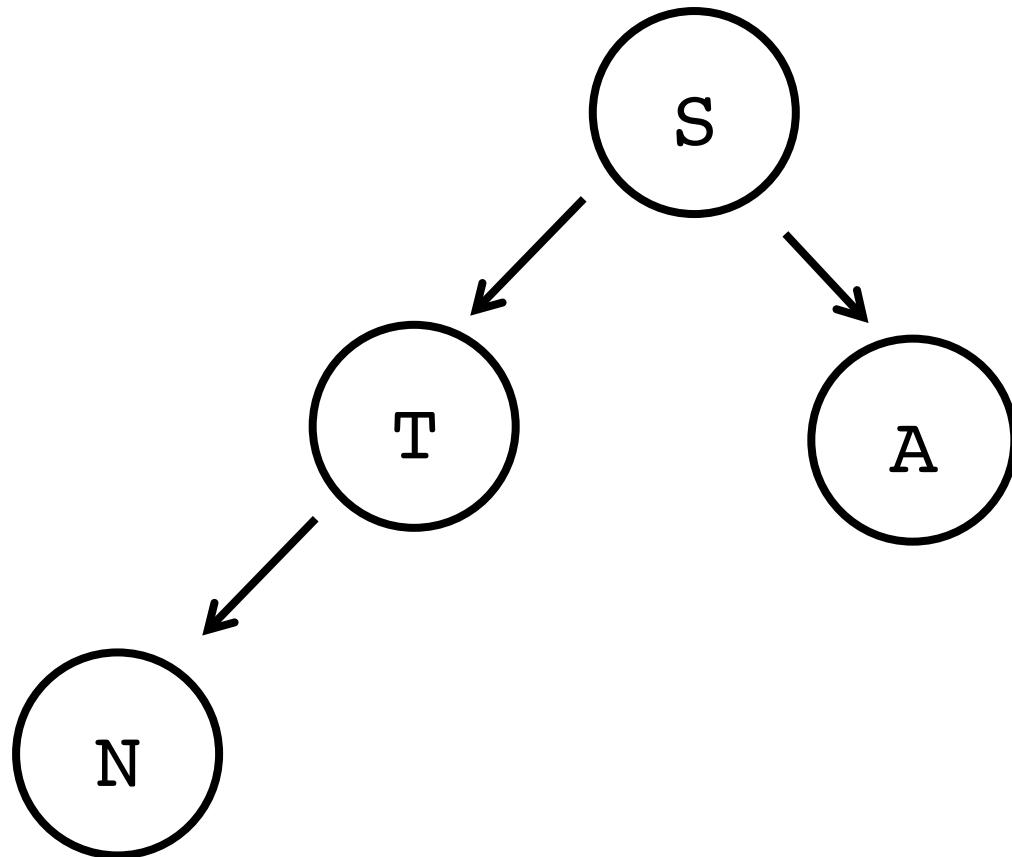


No Cycles

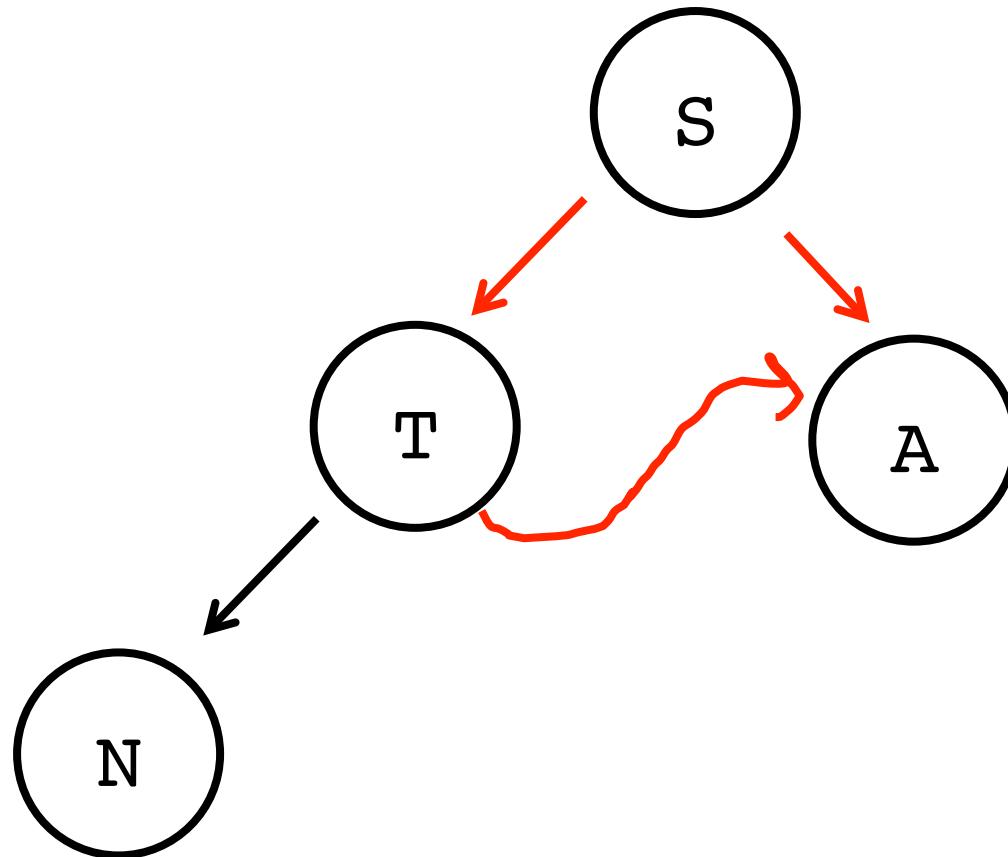


This is not a tree because the red edges make a cycle

No Cycles



No Cycles



This is not a tree because the red edges make a cycle

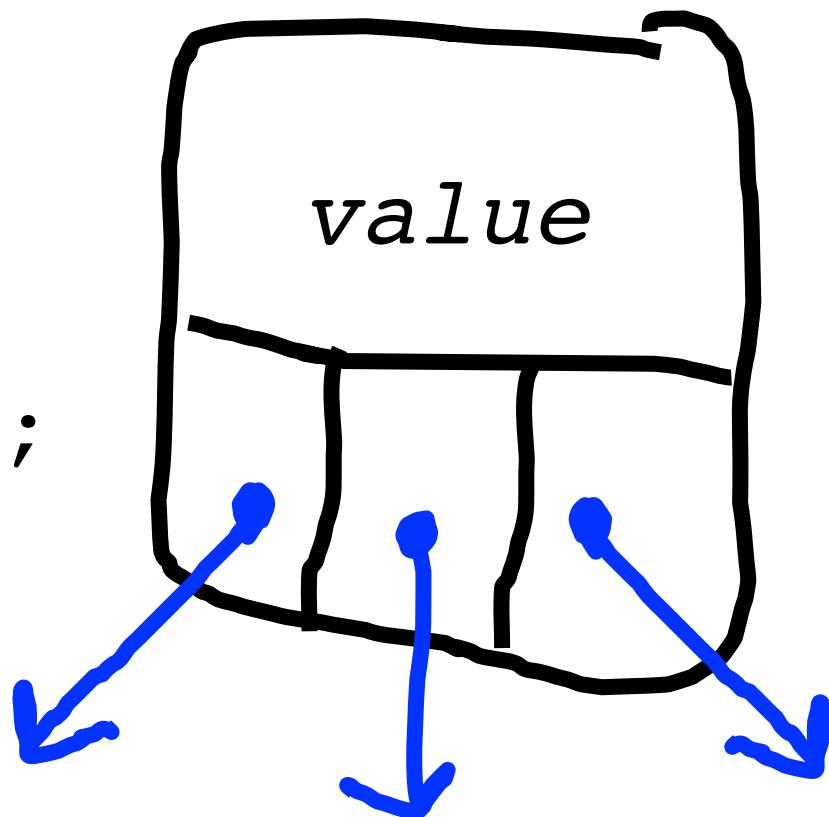
Other Kinds of Trees



One of my favorite trees

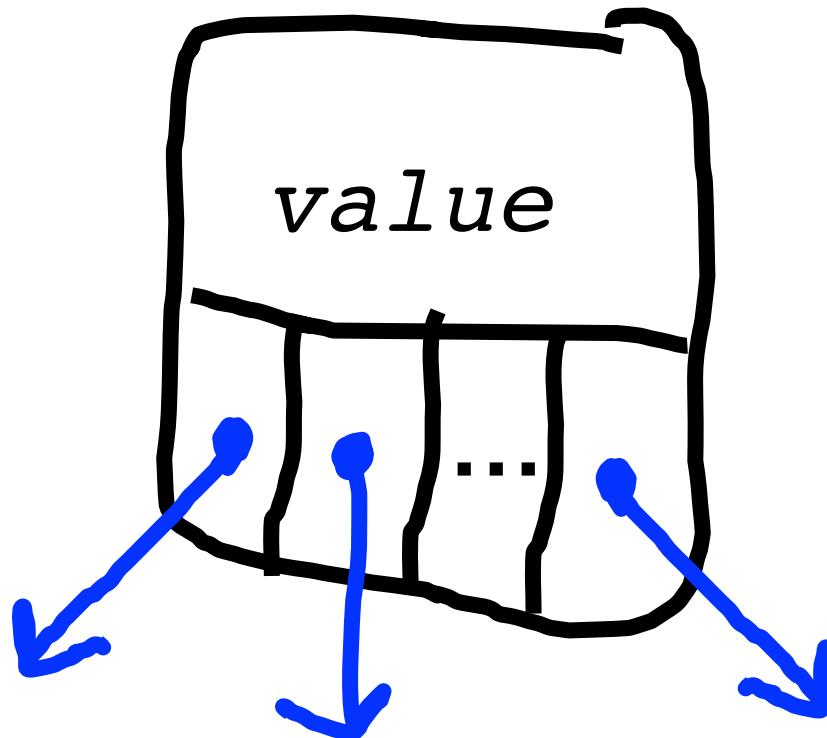
Trinary Tree

```
struct Tree {  
    string value;  
    Tree * left;  
    Tree * middle;  
    Tree * right;  
};
```



Tree

```
struct Tree {  
    string value;  
    Vector<Tree *> children;  
};
```



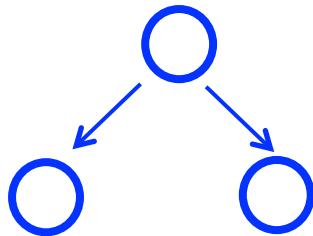
Structs or Classes

```
struct Tree {  
    string value;  
    Tree * left;  
    Tree * right;  
};
```

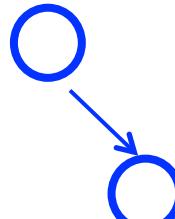
```
class Tree {  
private:  
    string value;  
    Vector<Tree *> children;  
};
```

How Many Valid Binary Trees?

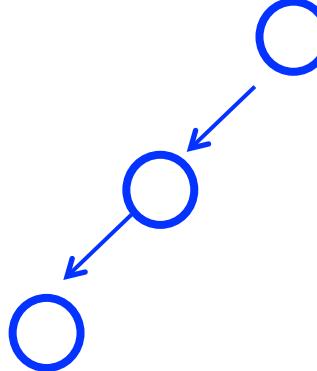
A) 3



B) 4



C) 5

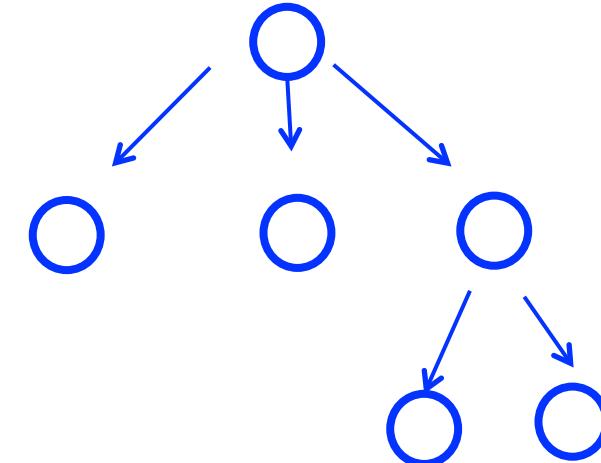
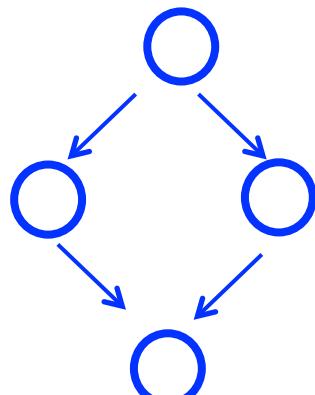


D) 6



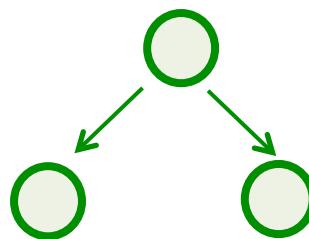
E) 7

NULL

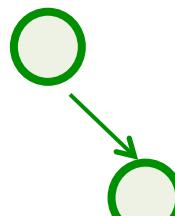


How Many Valid Binary Trees?

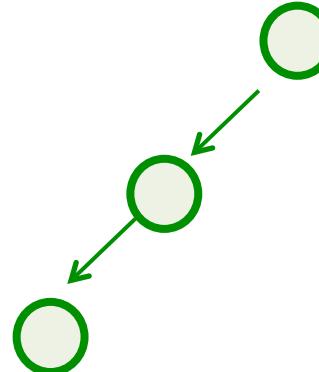
A) 3



B) 4



C) 5

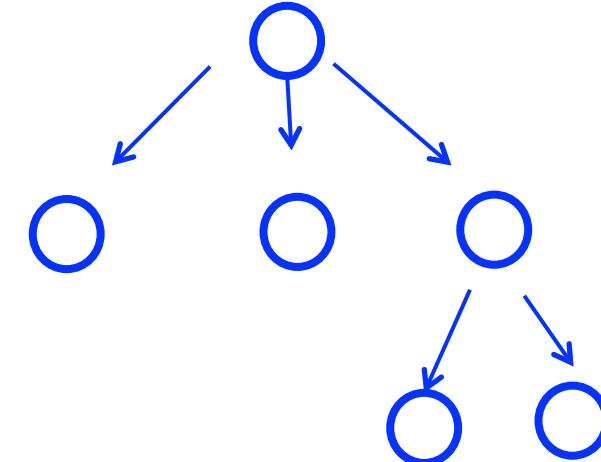
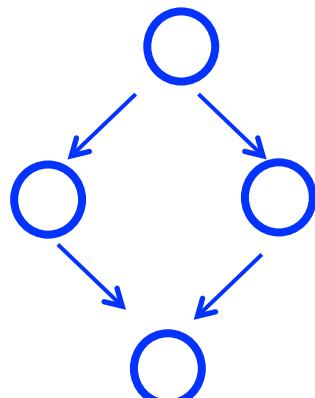


D) 6



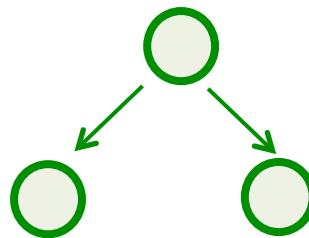
E) 7

NULL

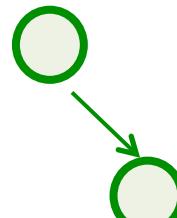


How Many Valid Binary Trees?

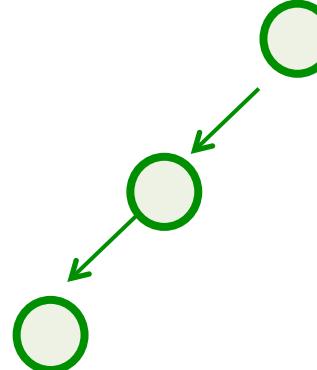
A) 3



B) 4



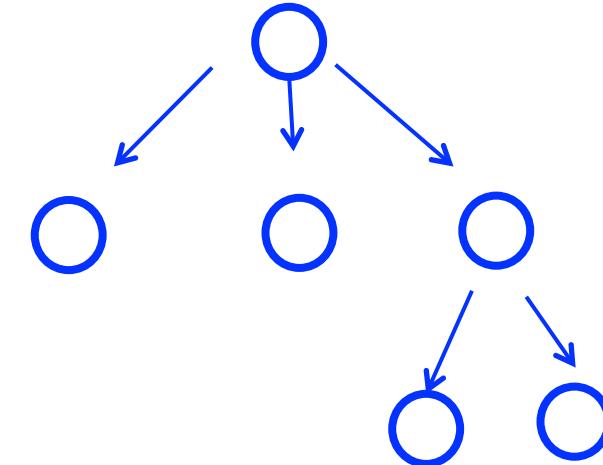
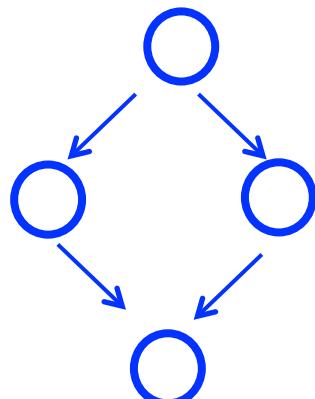
C) 5



D) 6

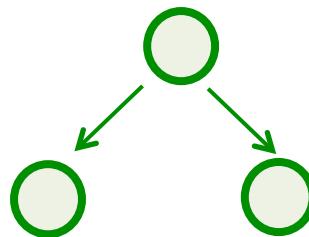


NULL

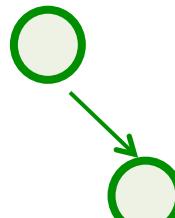


How Many Valid Binary Trees?

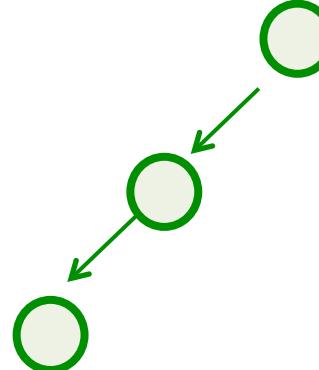
A) 3



B) 4



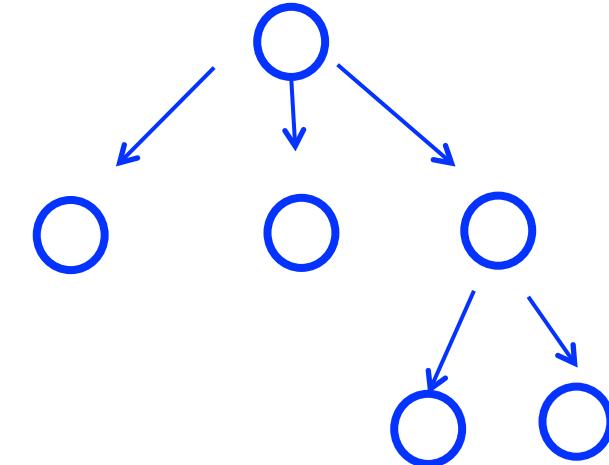
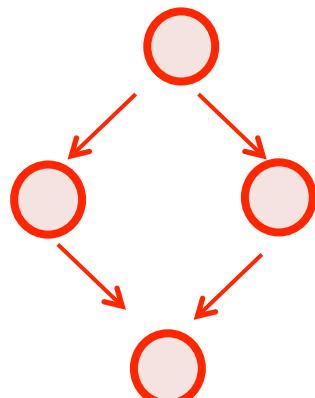
C) 5



D) 6

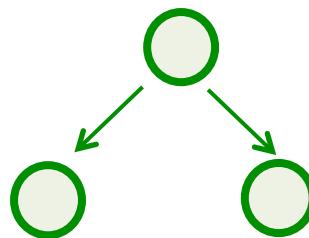


NULL

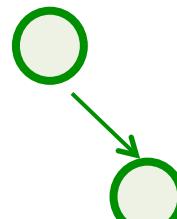


How Many Valid Binary Trees?

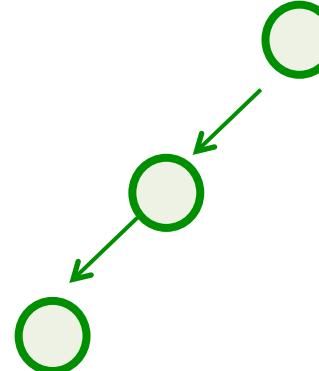
A) 3



B) 4



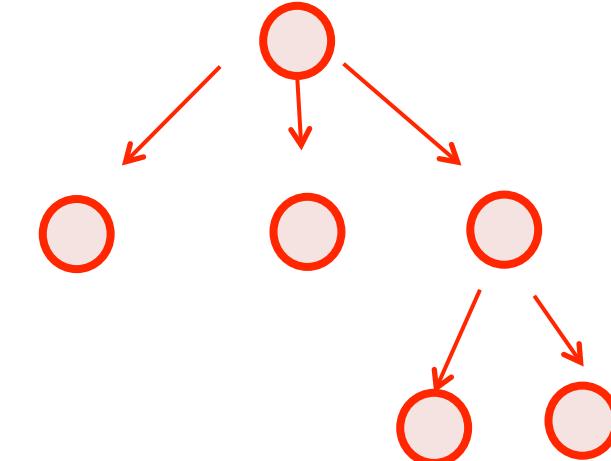
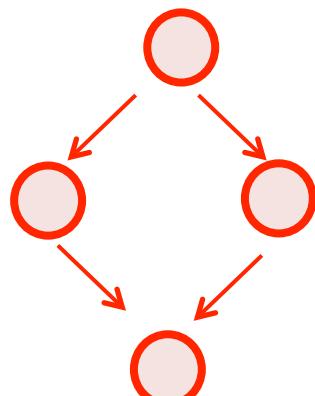
C) 5



D) 6

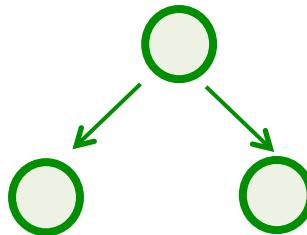


NULL

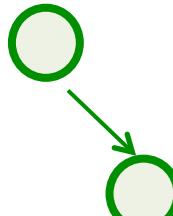


How Many Valid Binary Trees?

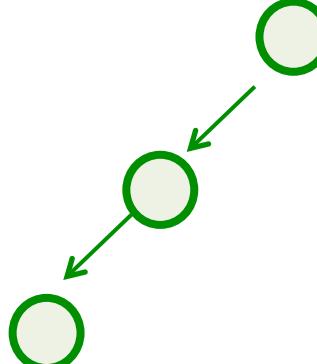
A) 3



B) 4



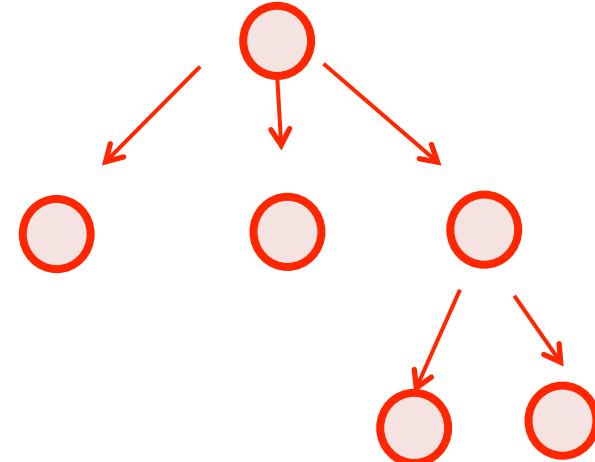
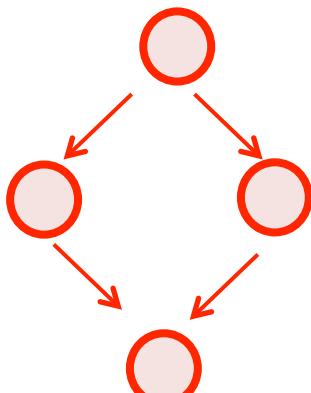
C) 5



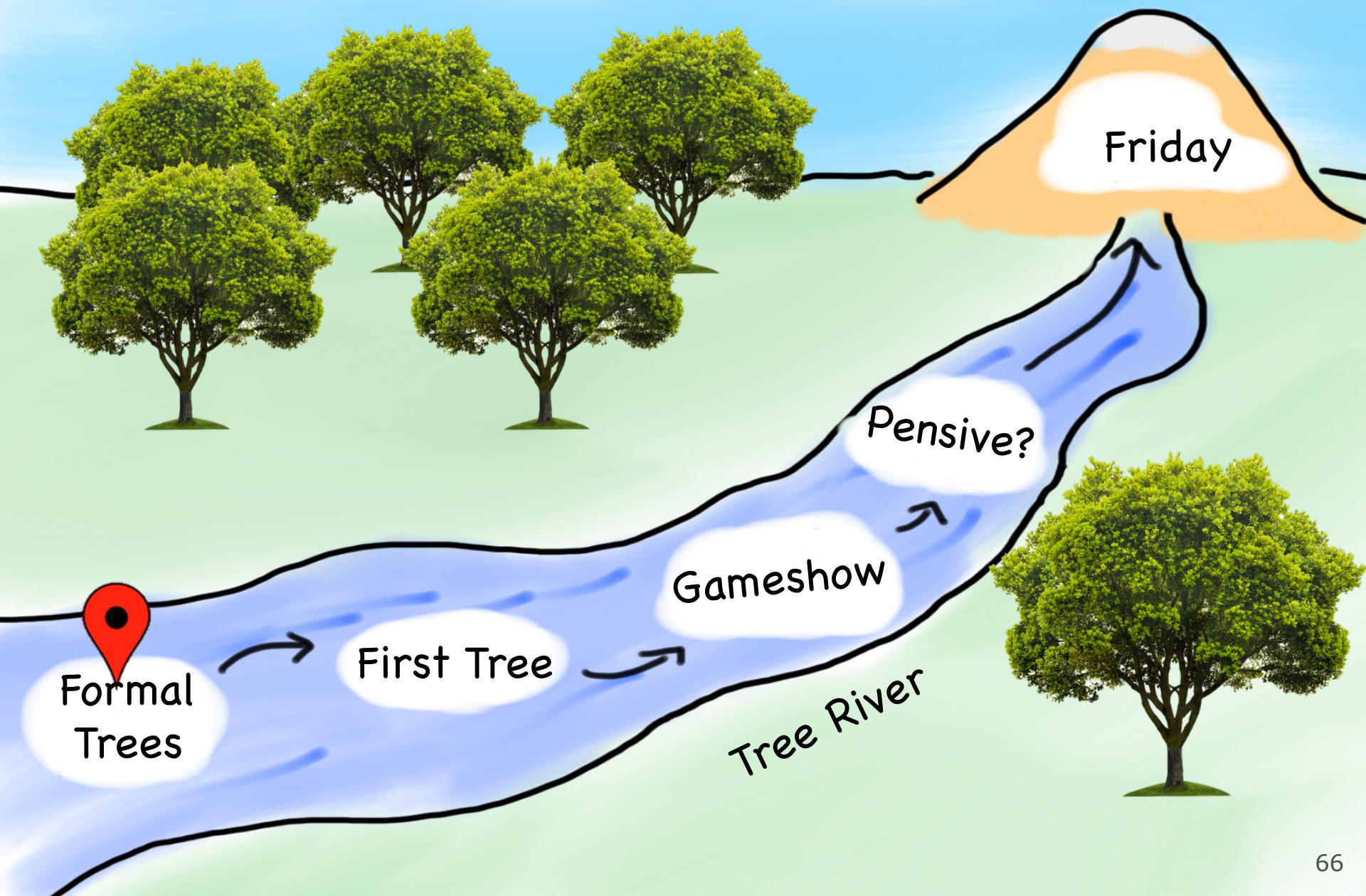
D) 6



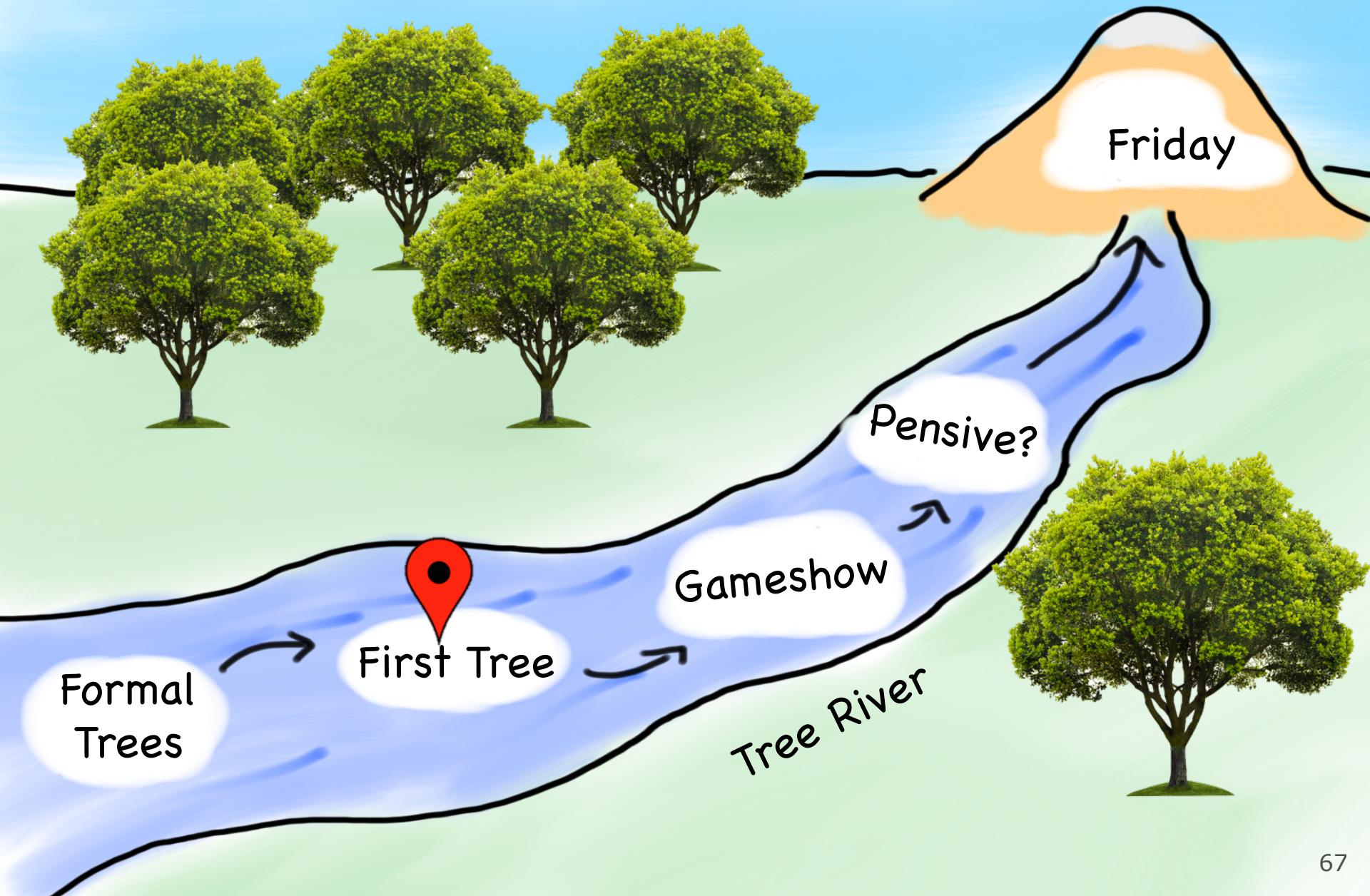
NULL



Today's Route



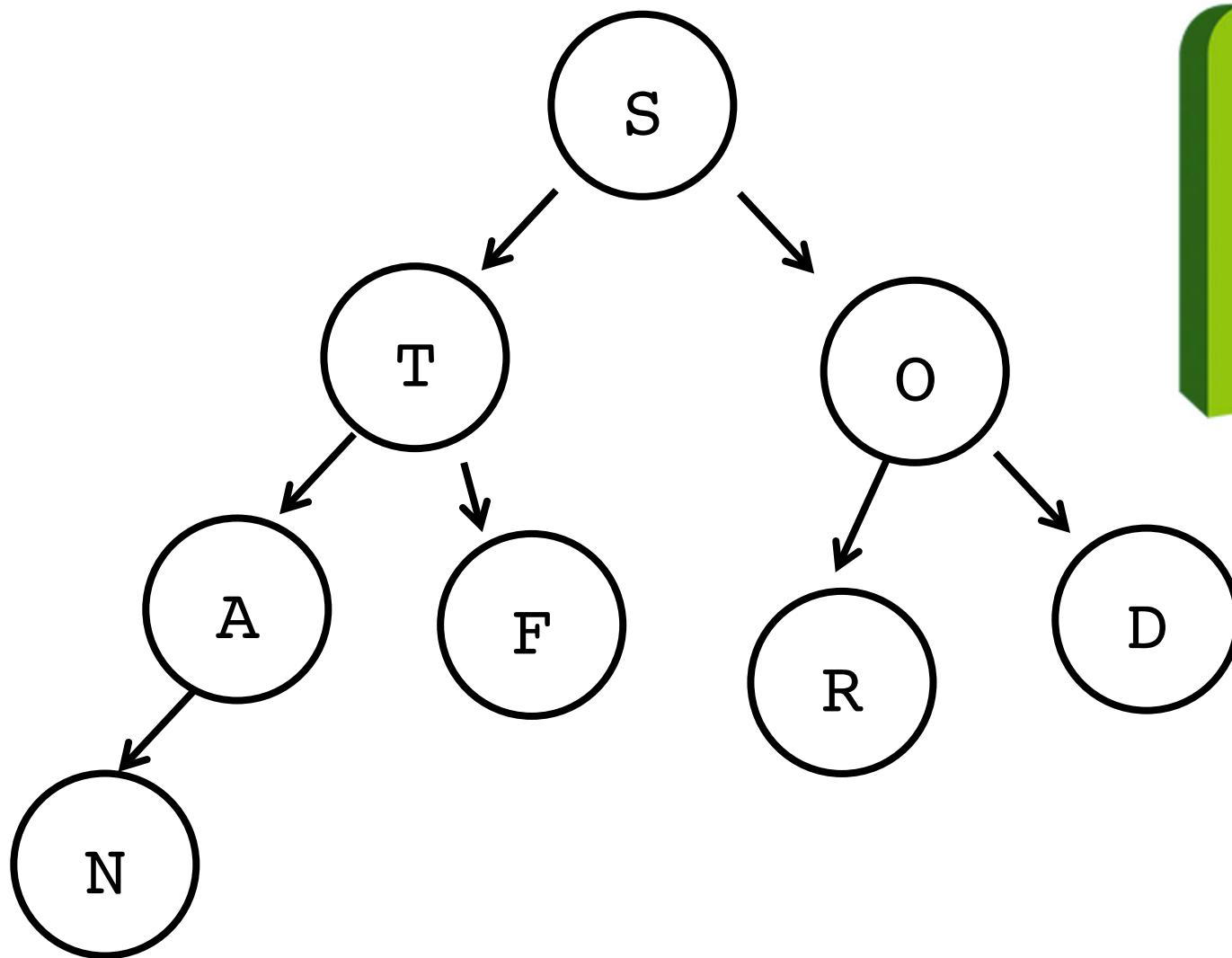
Today's Route



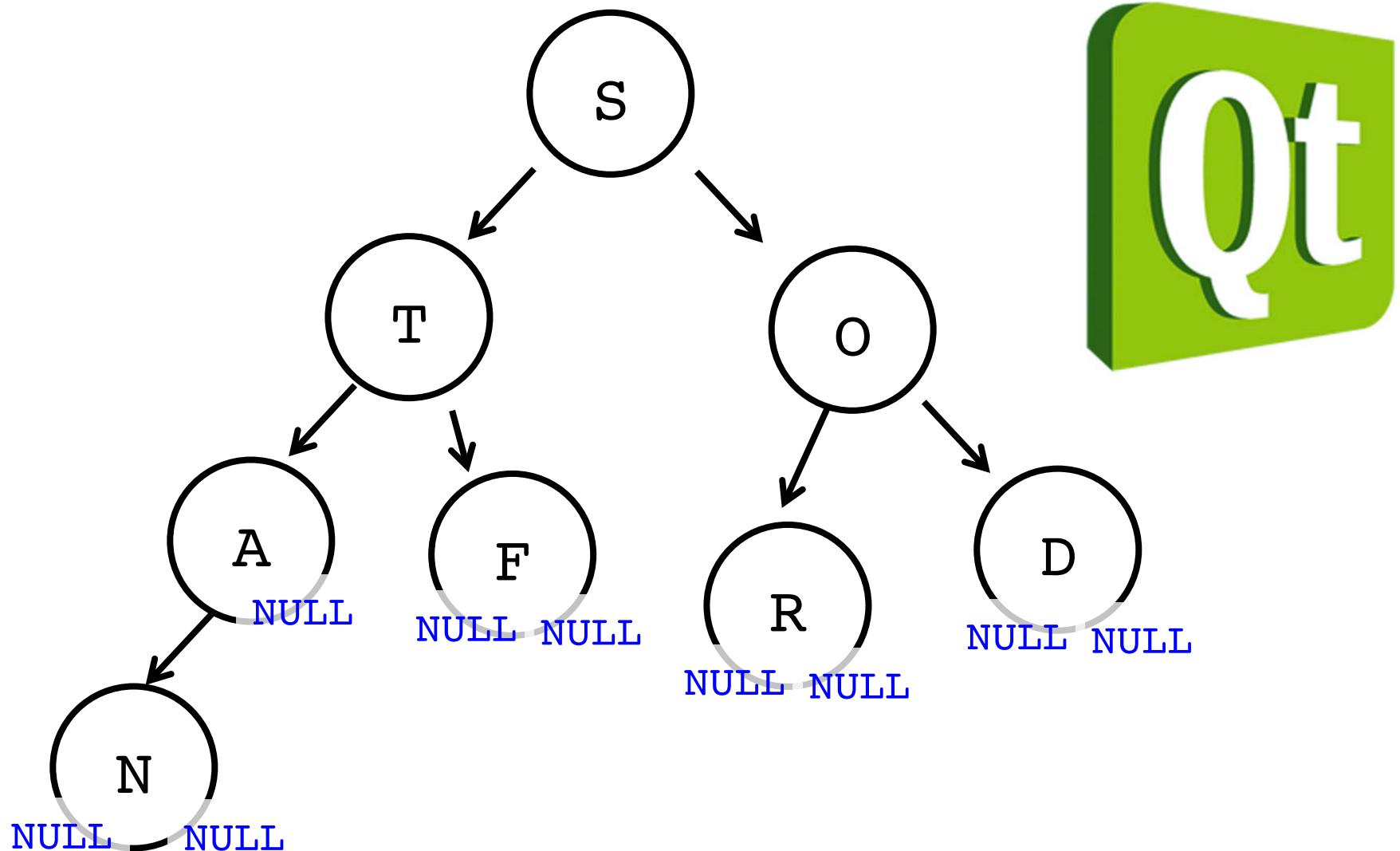




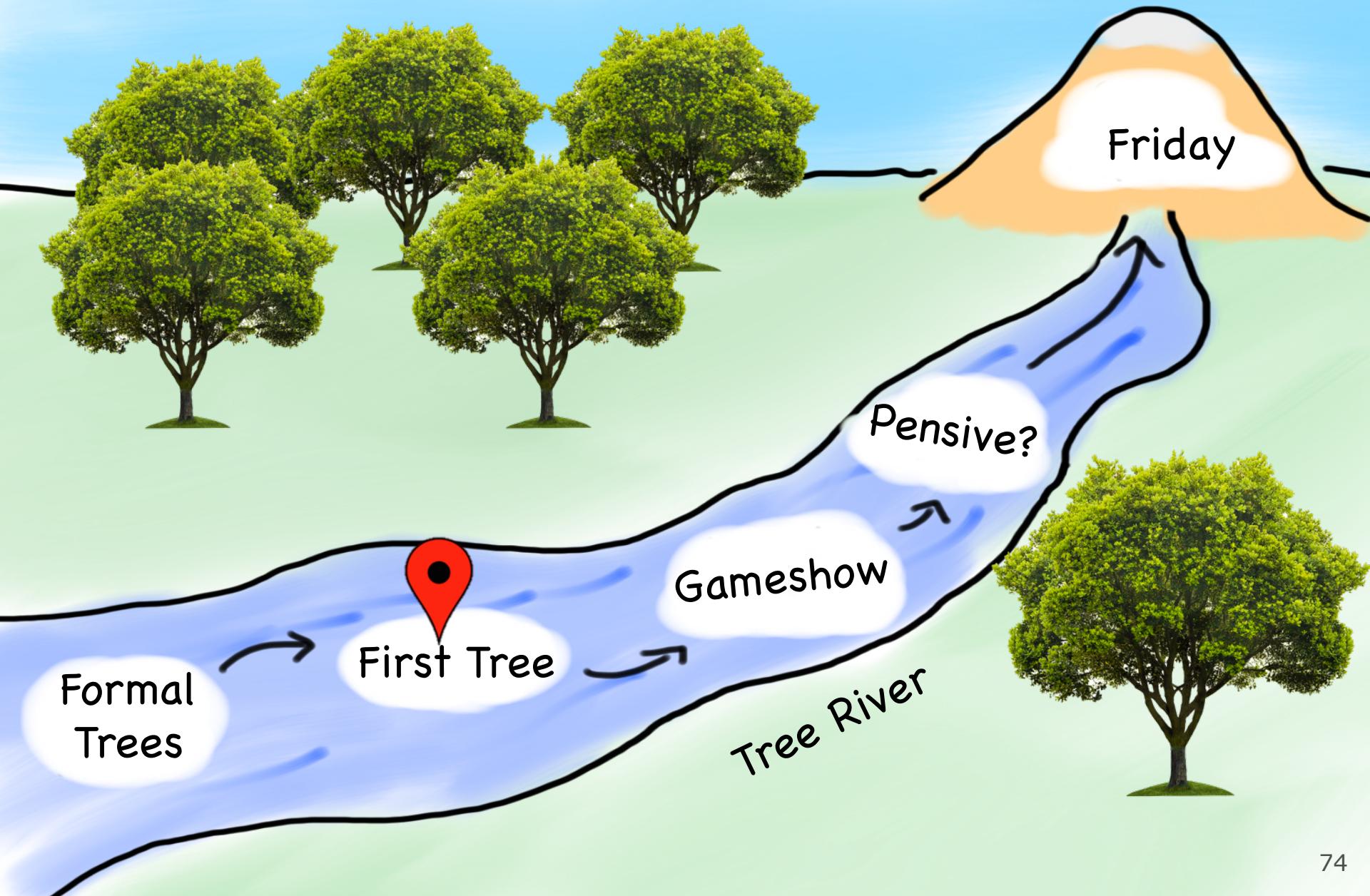
Cout This Tree



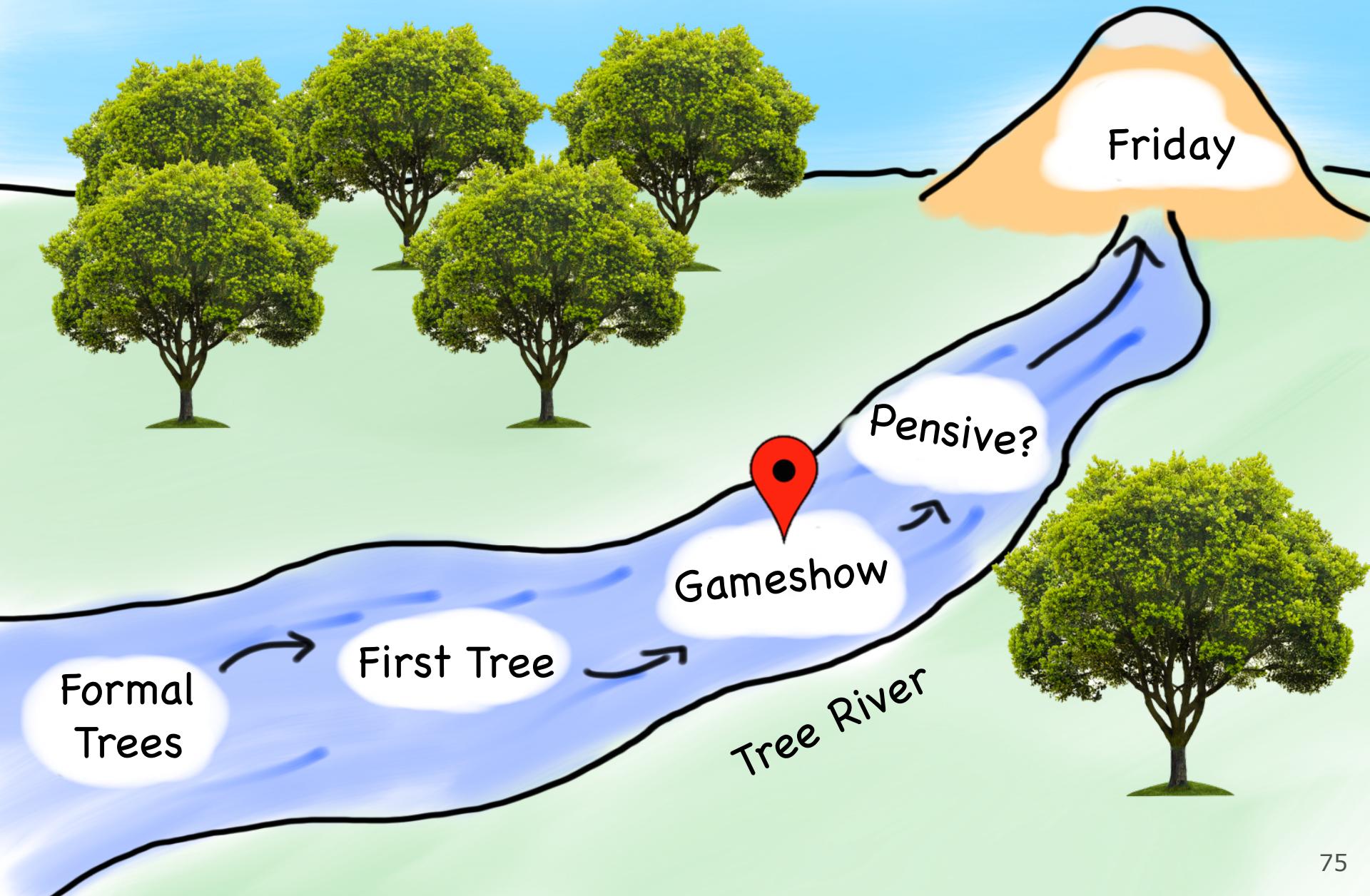
NULL Children



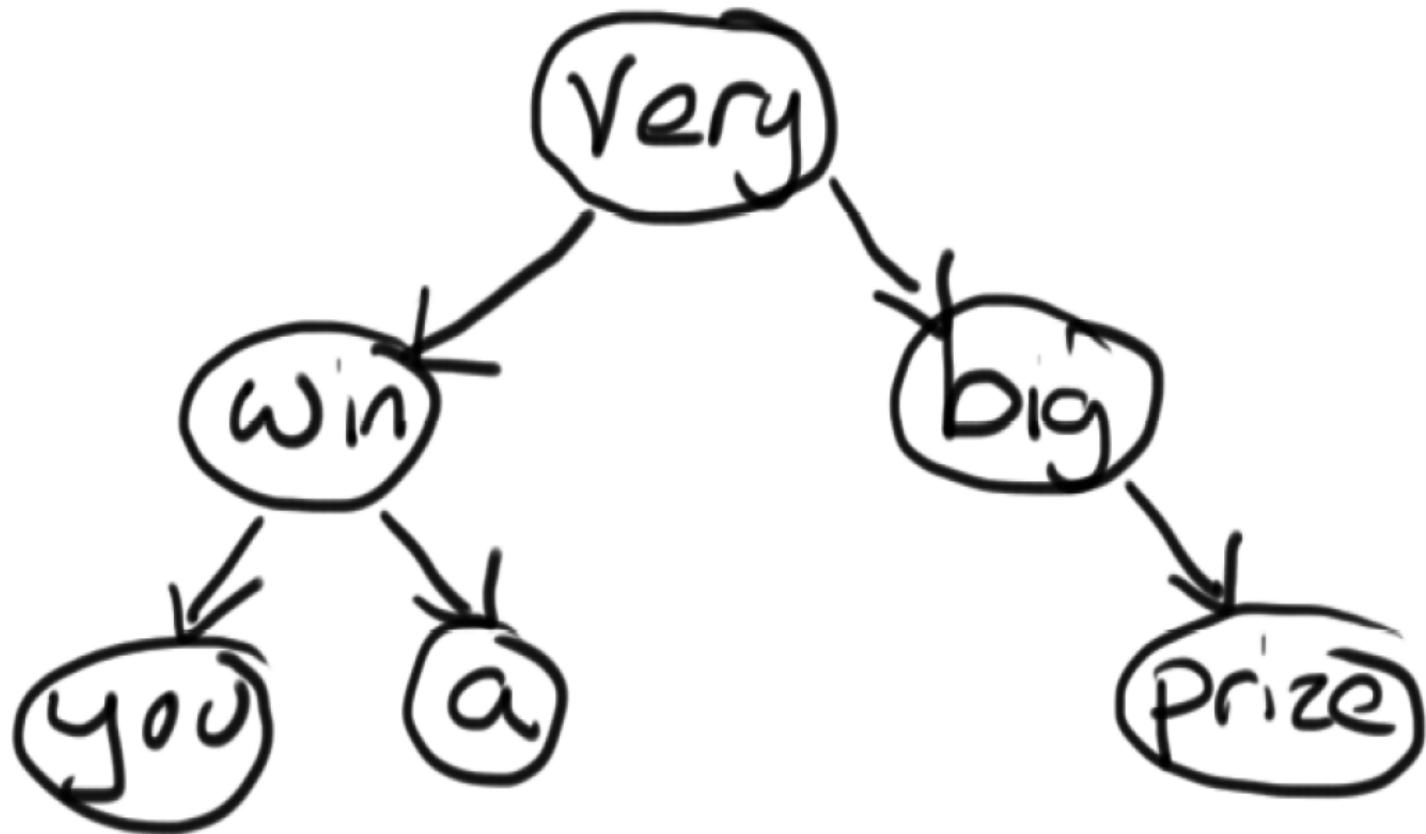
Today's Route



Today's Route



Game Show Tree



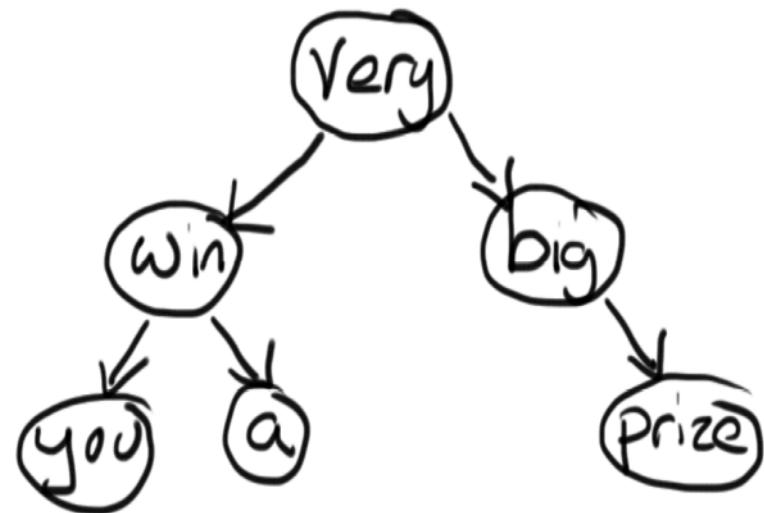
Game Show Tree

I VOLUNTEER

AS TRIBUTE

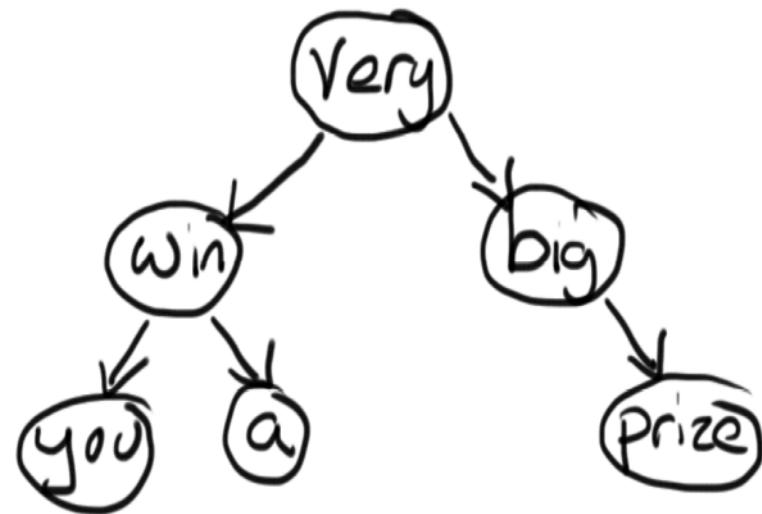
Game Show Tree

```
struct Tree {  
    string value;  
    Tree * left;  
    Tree * right;  
};
```



Game Show Tree

```
int main() {
    introduction();
    Tree * tree = initTree();
    int choice = getUserChoice();
    suspense();
    switch (choice) {
        case 1: doorOne(tree); break;
        case 2: doorTwo(tree); break;
        case 3: doorThree(tree); break;
    }
    return 0;
}
```

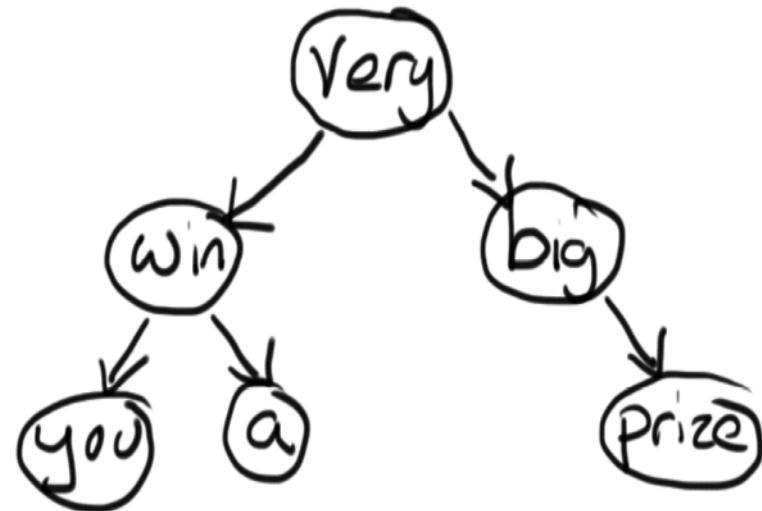


Game Show Tree

```
void doorOne(Tree * tree) {  
    if(tree == NULL) return;  
    cout<<tree->value<<" "  
    doorOne(tree->left);  
    doorOne(tree->right);  
}
```

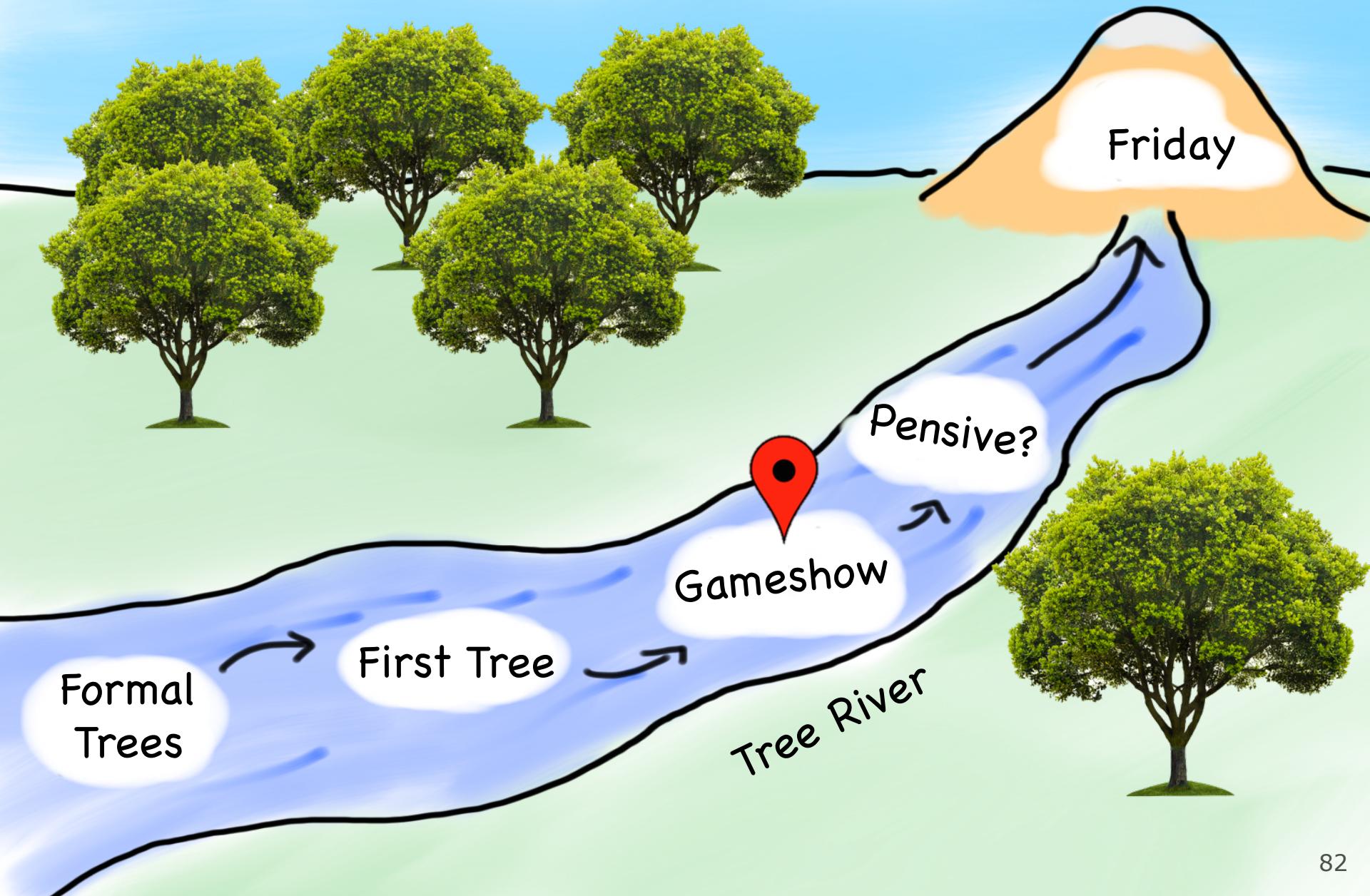
```
void doorTwo(Tree * tree) {  
    if(tree == NULL) return;  
    doorTwo(tree->left);  
    cout<<tree->value<<" "  
    doorTwo(tree->right);  
}
```

```
Void doorThree(Tree * tree) {  
    if(tree == NULL) return;  
    doorThree(tree->left);  
    doorThree(tree->right);  
    cout<<tree->value<<" "  
}
```

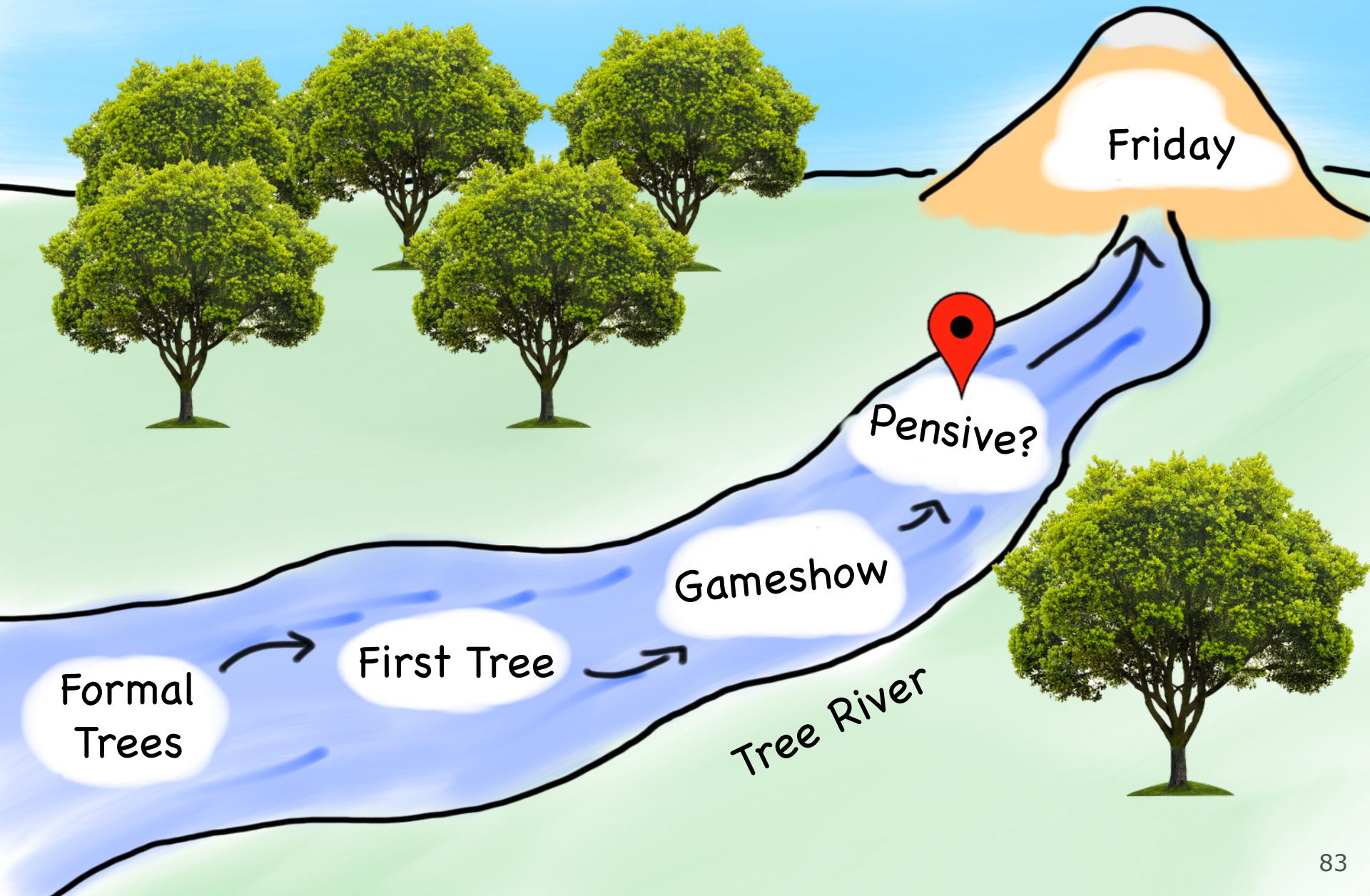




Today's Route



Today's Route



DUMBLEDORE IS SO EPIC

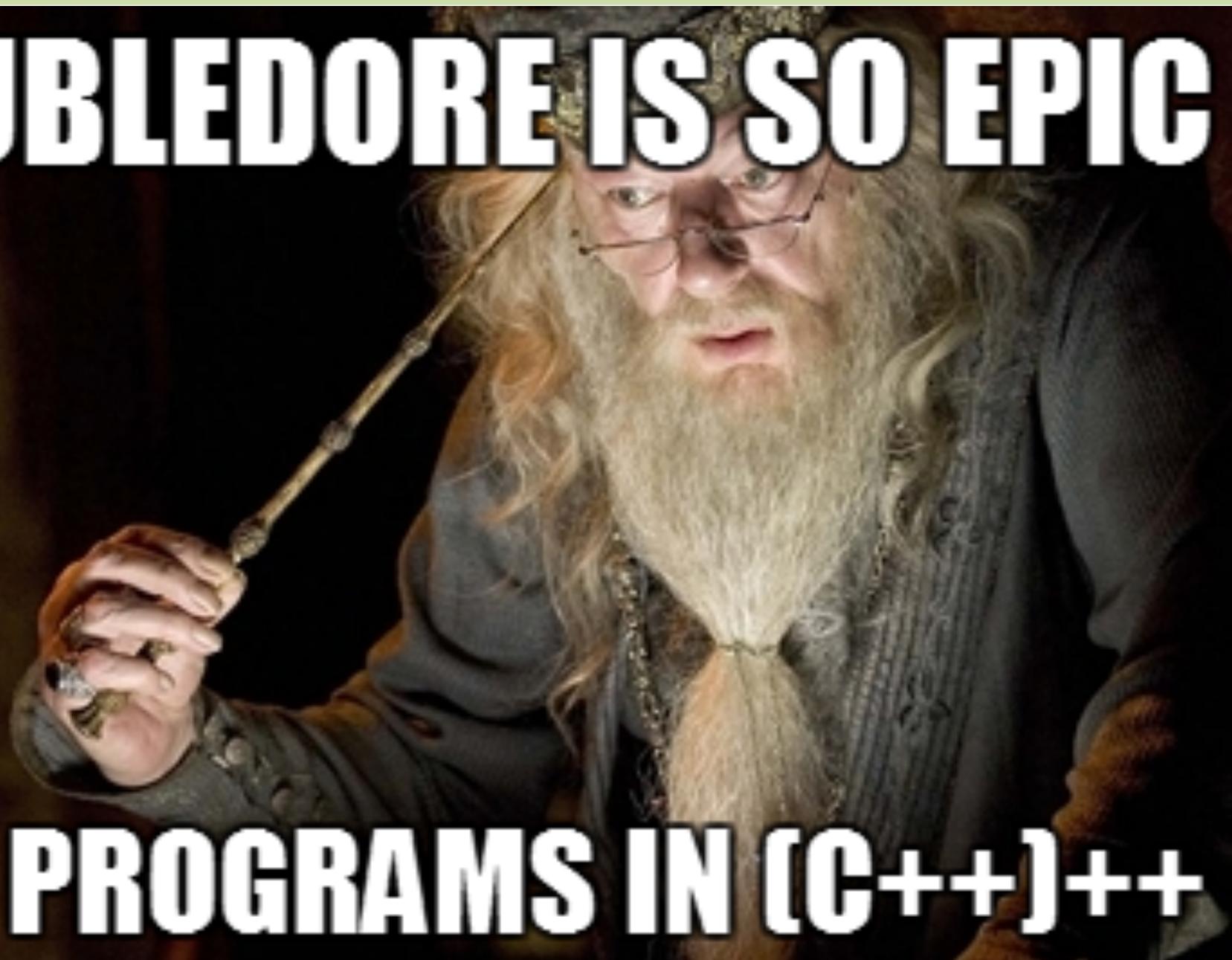


DUMBLEDORE IS SO EPIC



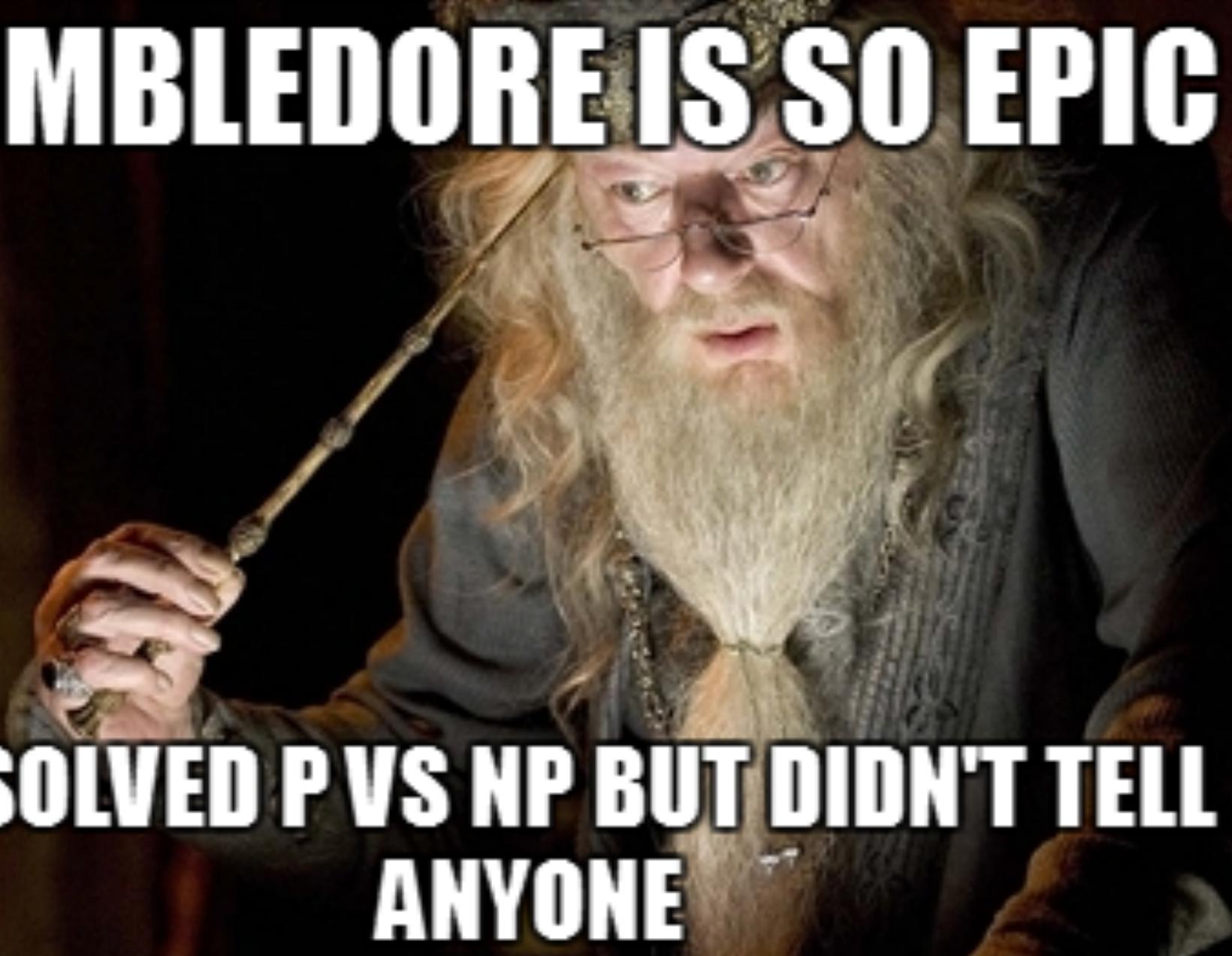
HE BEAT WATSON AT JEOPARDY

DUMBLEDORE IS SO EPIC



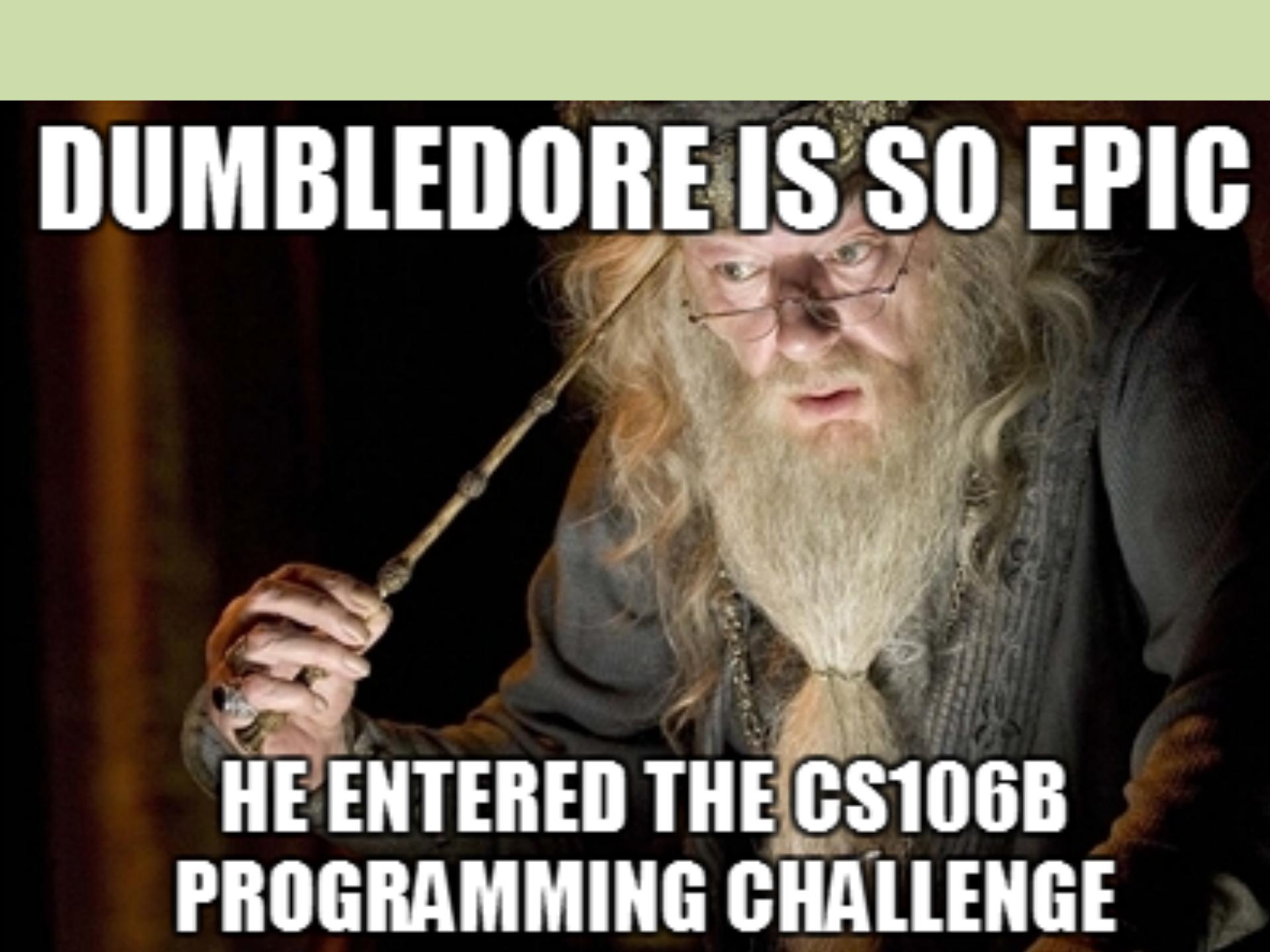
HE PROGRAMS IN (C++)++

DUMBLEDORE IS SO EPIC



**HE SOLVED P VS NP BUT DIDN'T TELL
ANYONE**

DUMBLEDORE IS SO EPIC



HE ENTERED THE CS106B
PROGRAMMING CHALLENGE

How come Dumbledore knows everything?

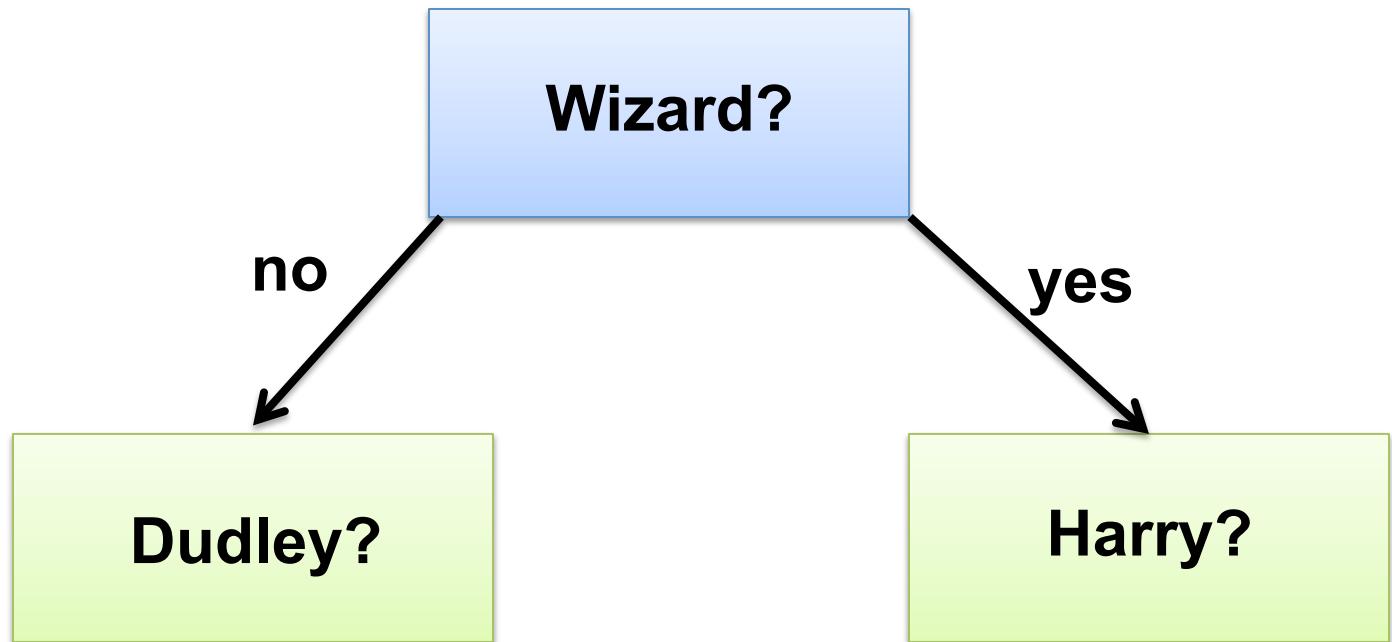
Pensive



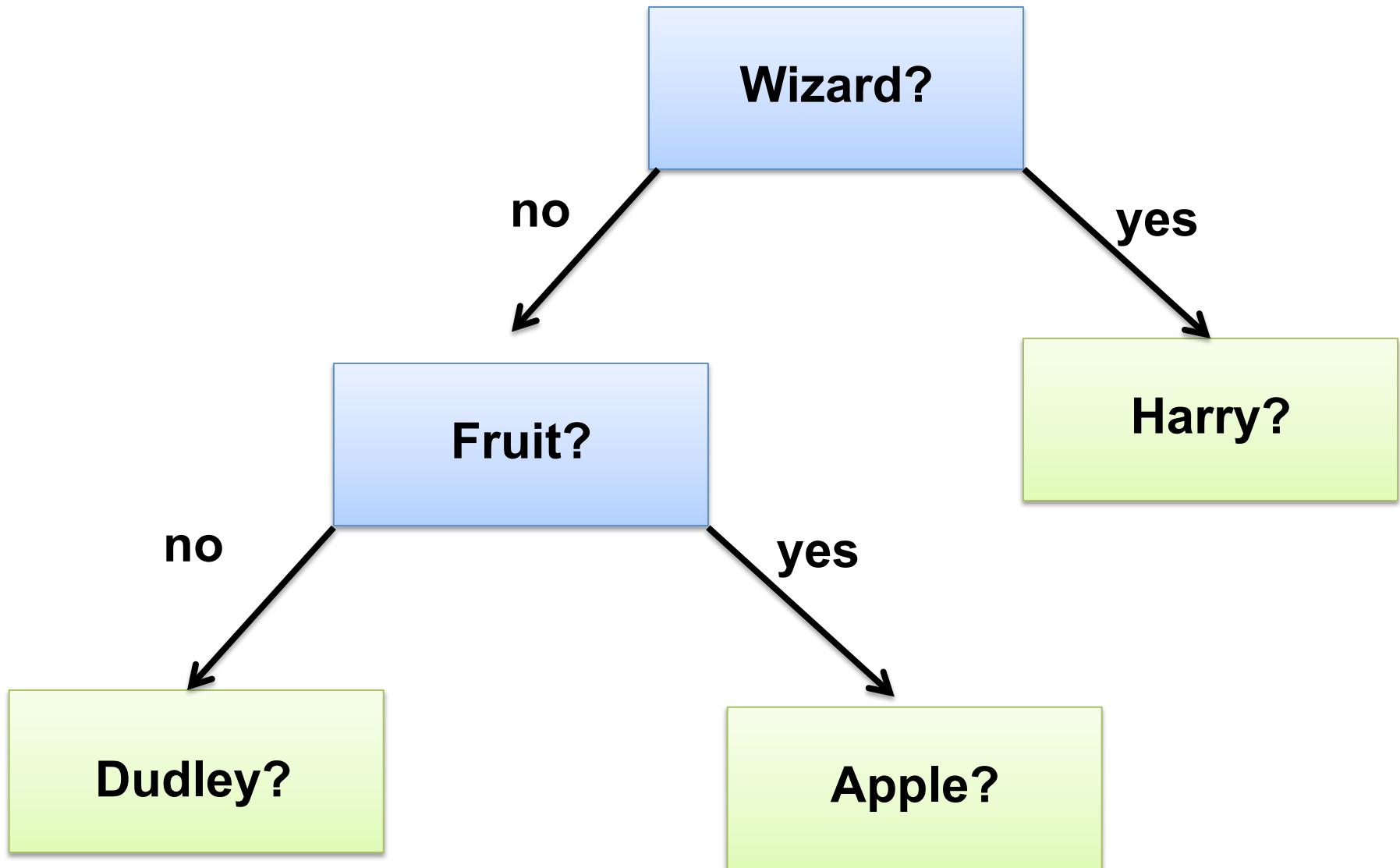
Pensive Demo



Pensive



Pensive



"Do, or do not.

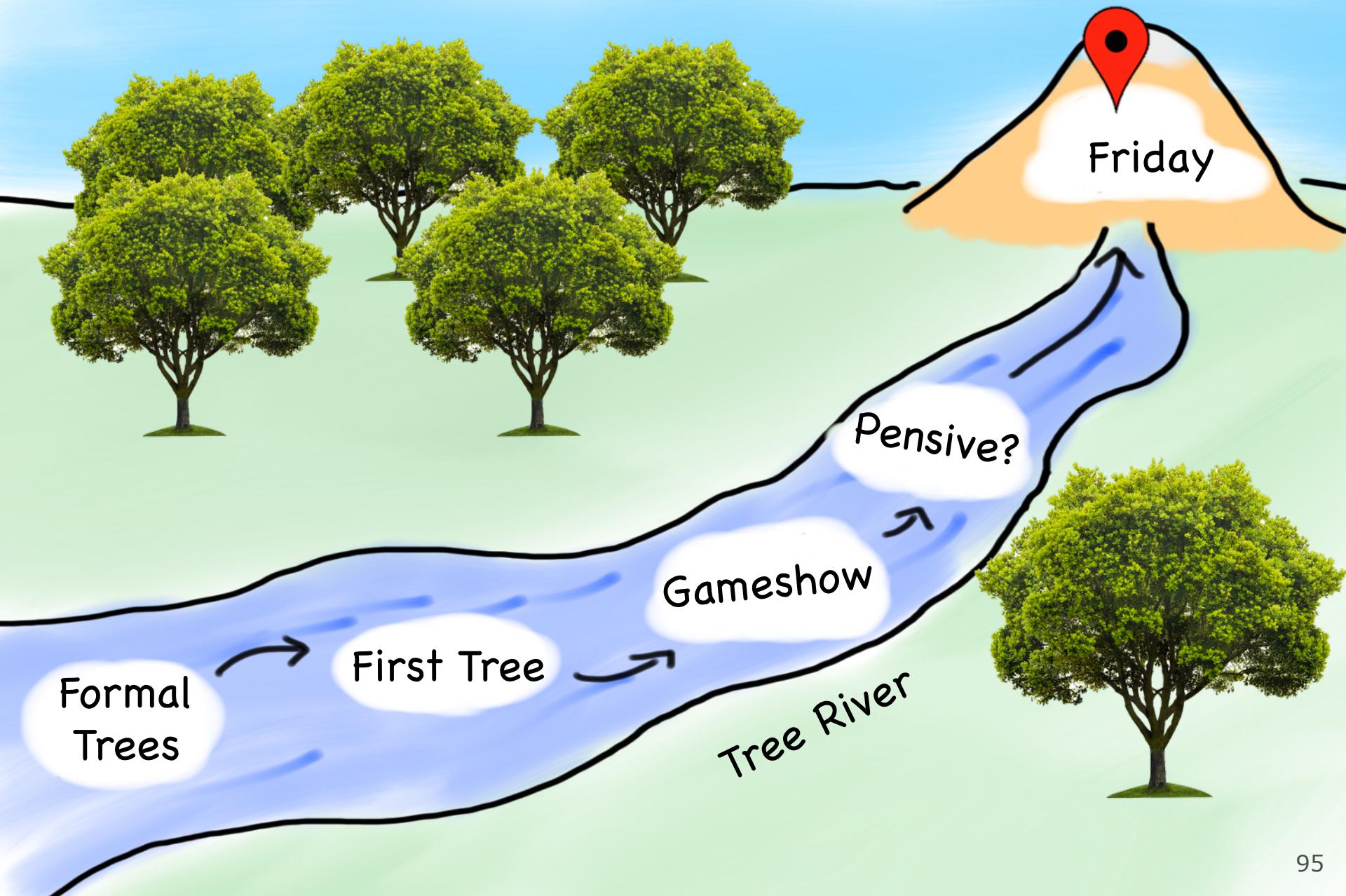
There is no try."

-Dumbledore

* actually Yoda. But what ever



Today's Route



Today's Goal

1. Be able to define a tree
2. Be able to traverse a tree

