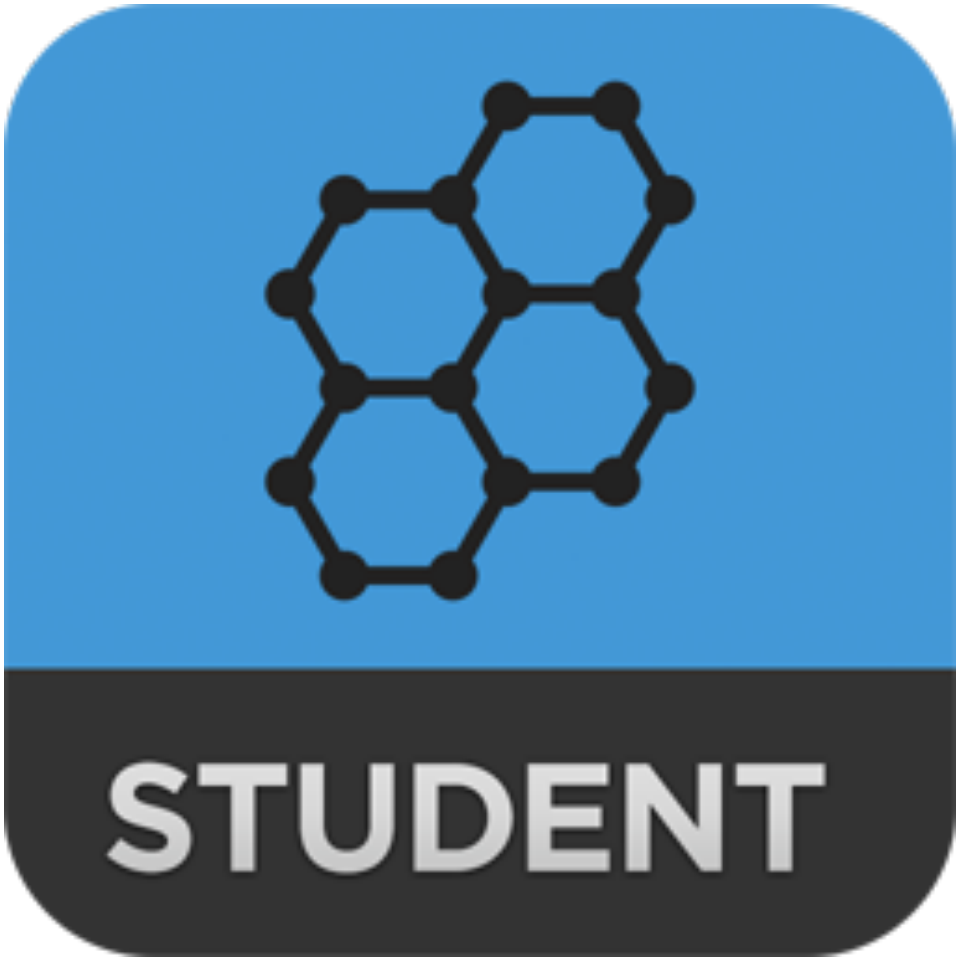
Trees

Chris Piech

CS 106B
Lecture 18
Feb 8, 2016

Socrative



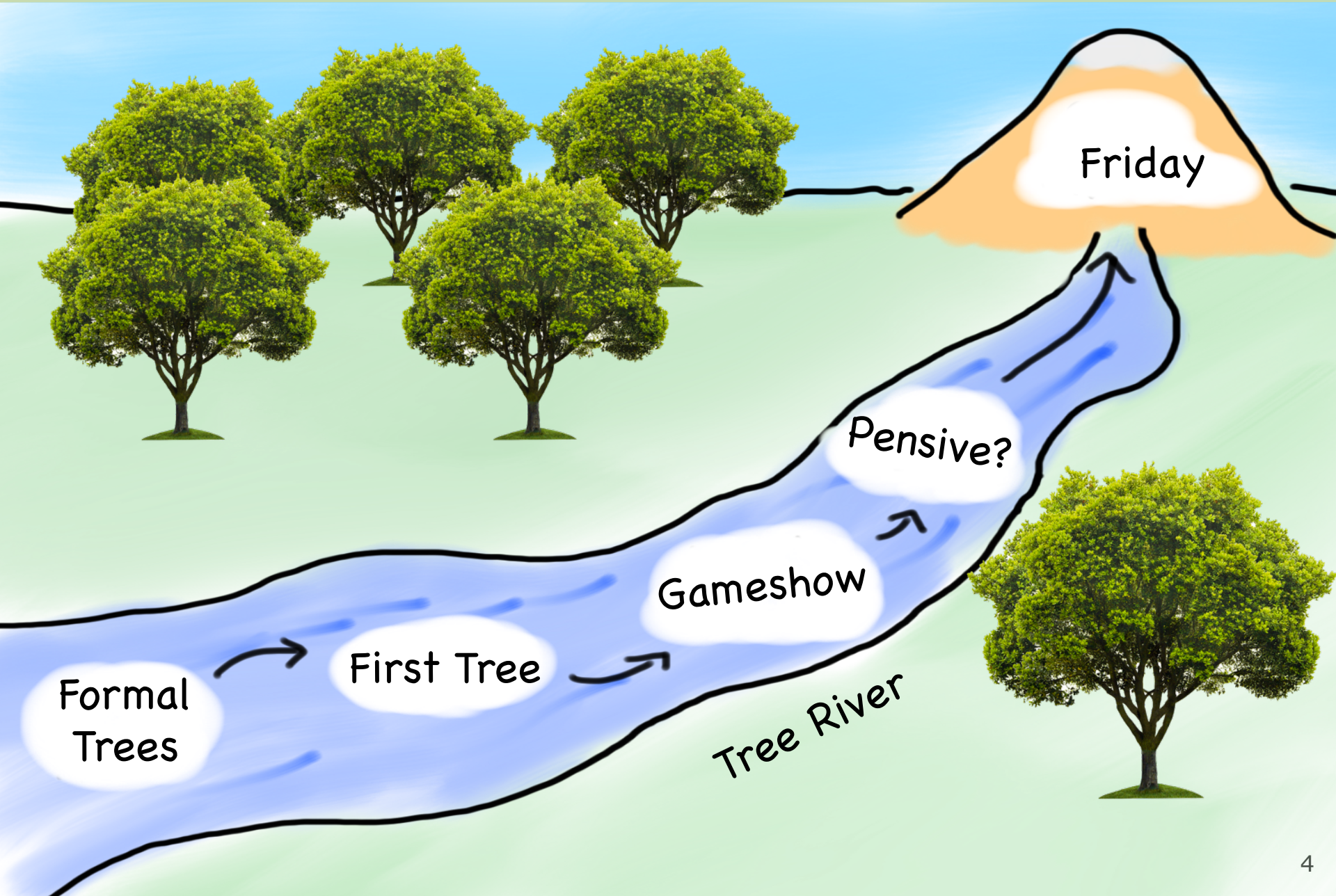
Room: **106BWIN16**

Today's Goal

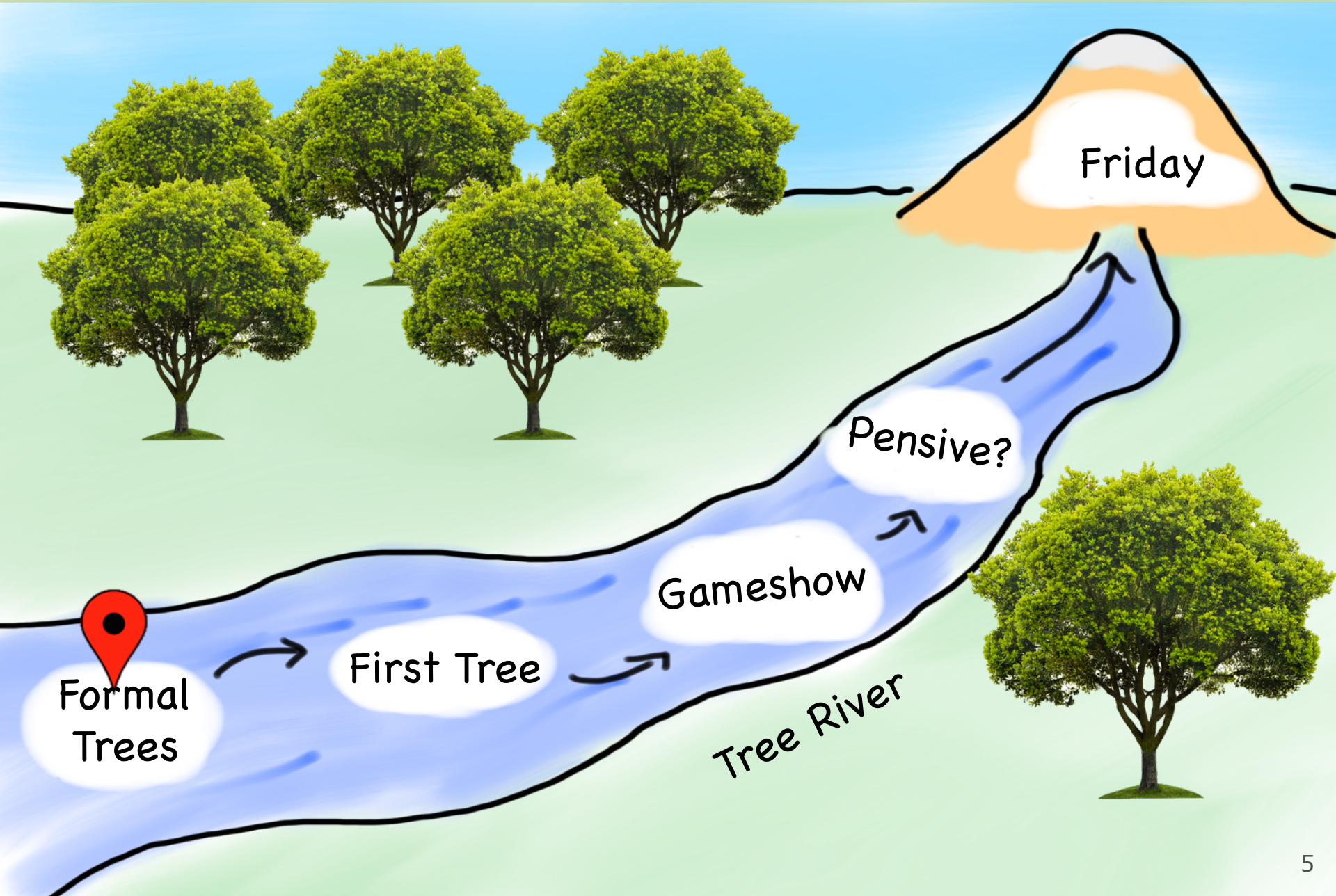
1. Be able to define a tree

2. Be able to traverse a tree

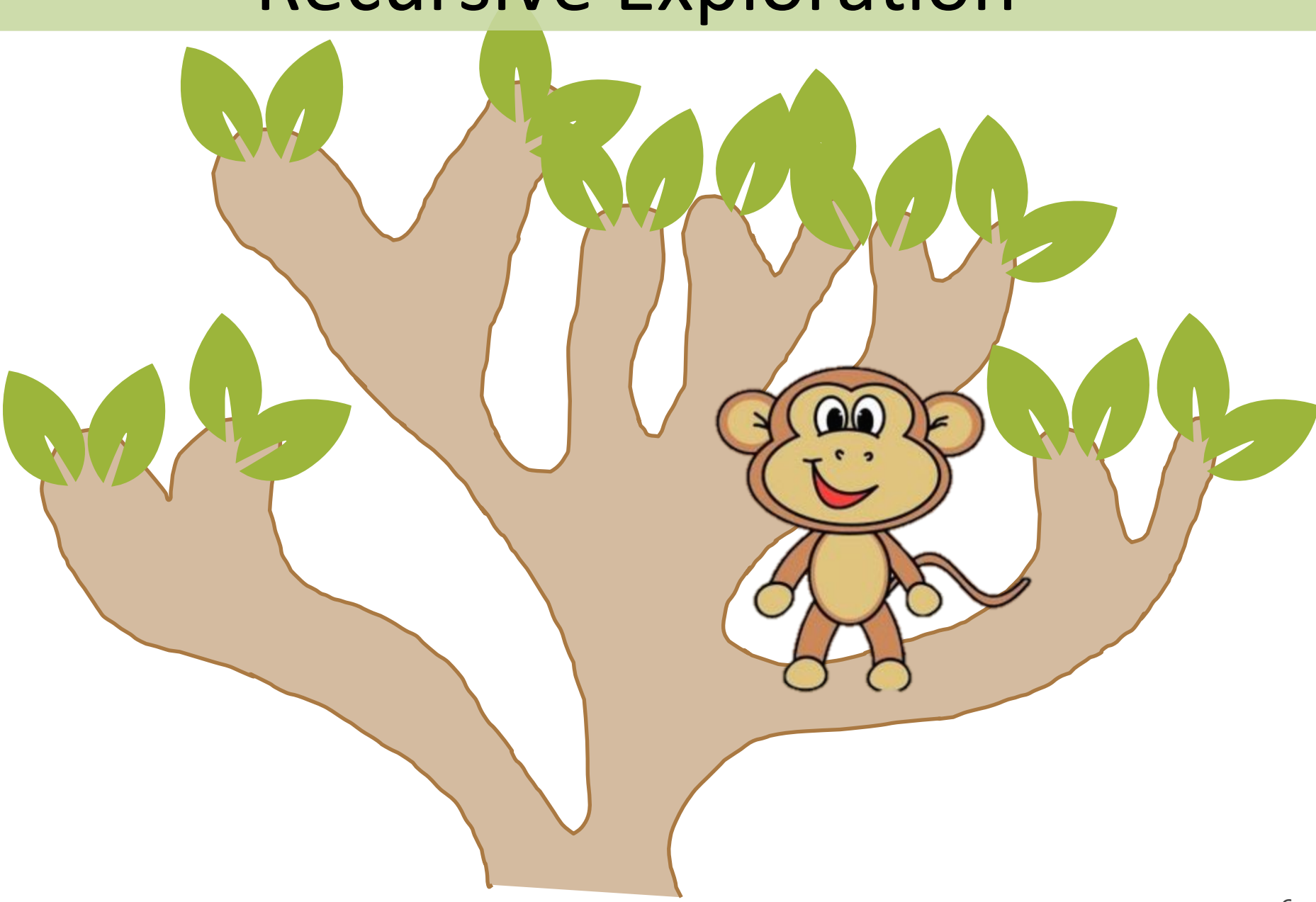
Today's Route



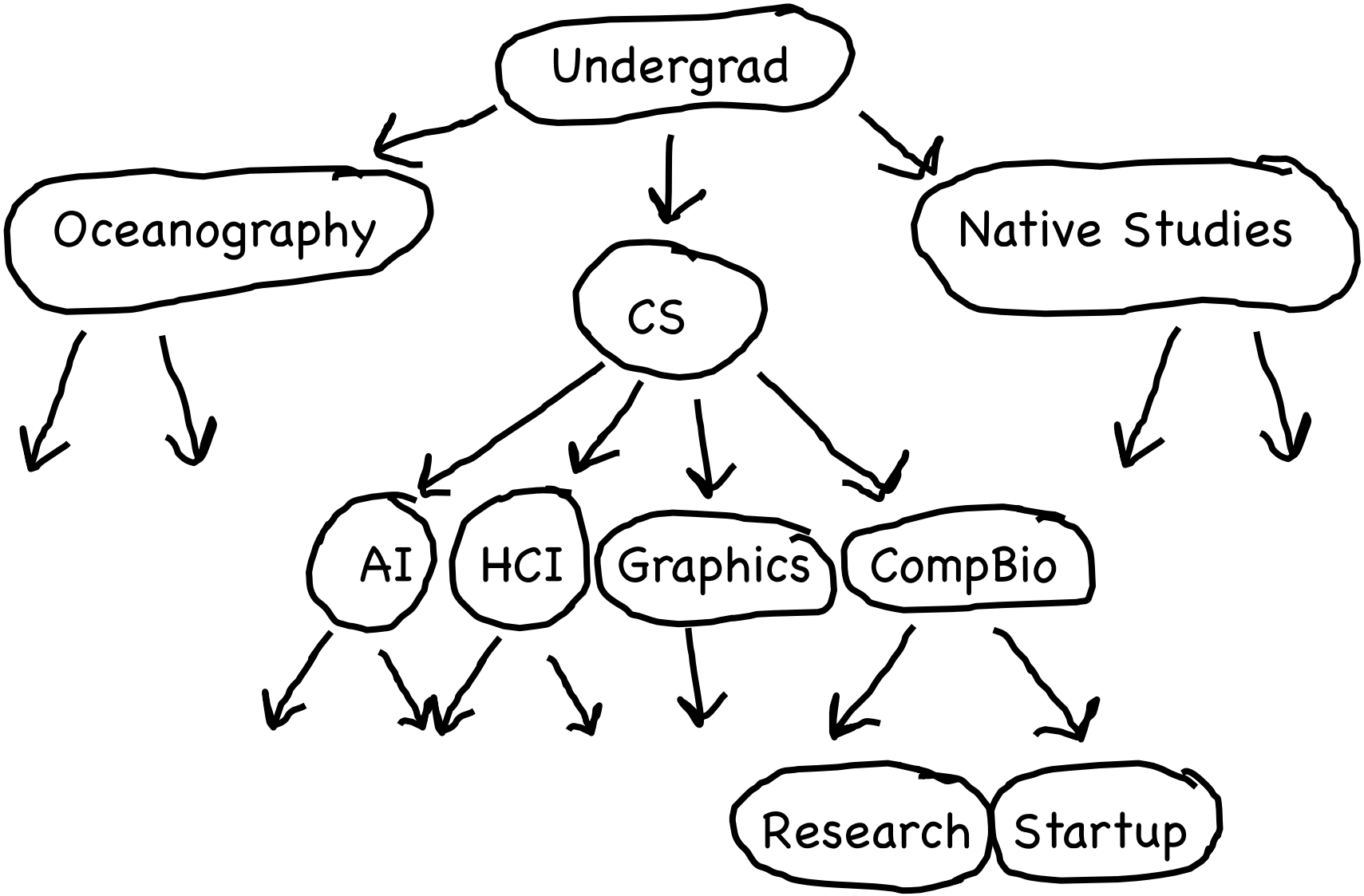
Today's Route



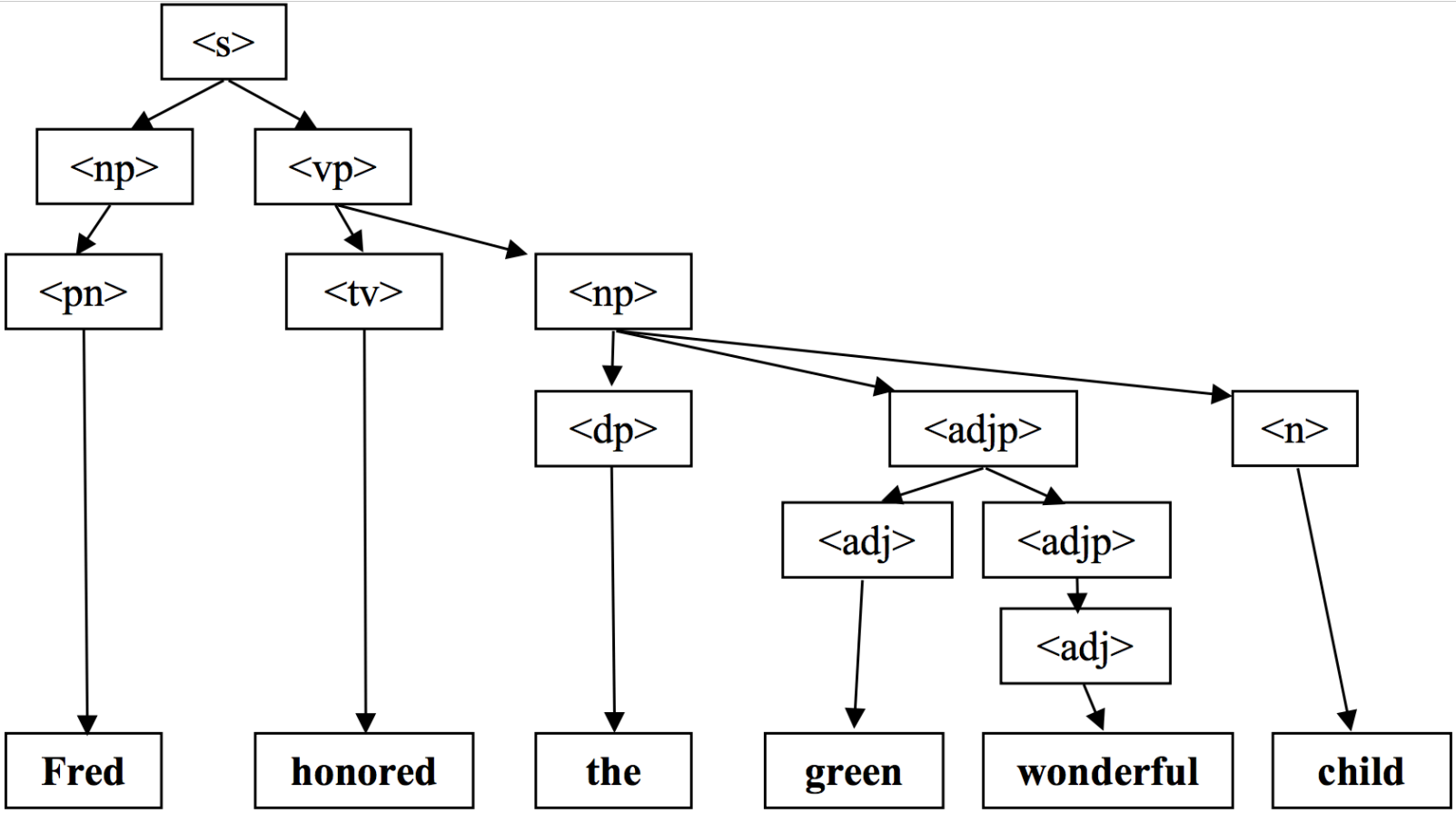
Recursive Exploration



Decision Trees

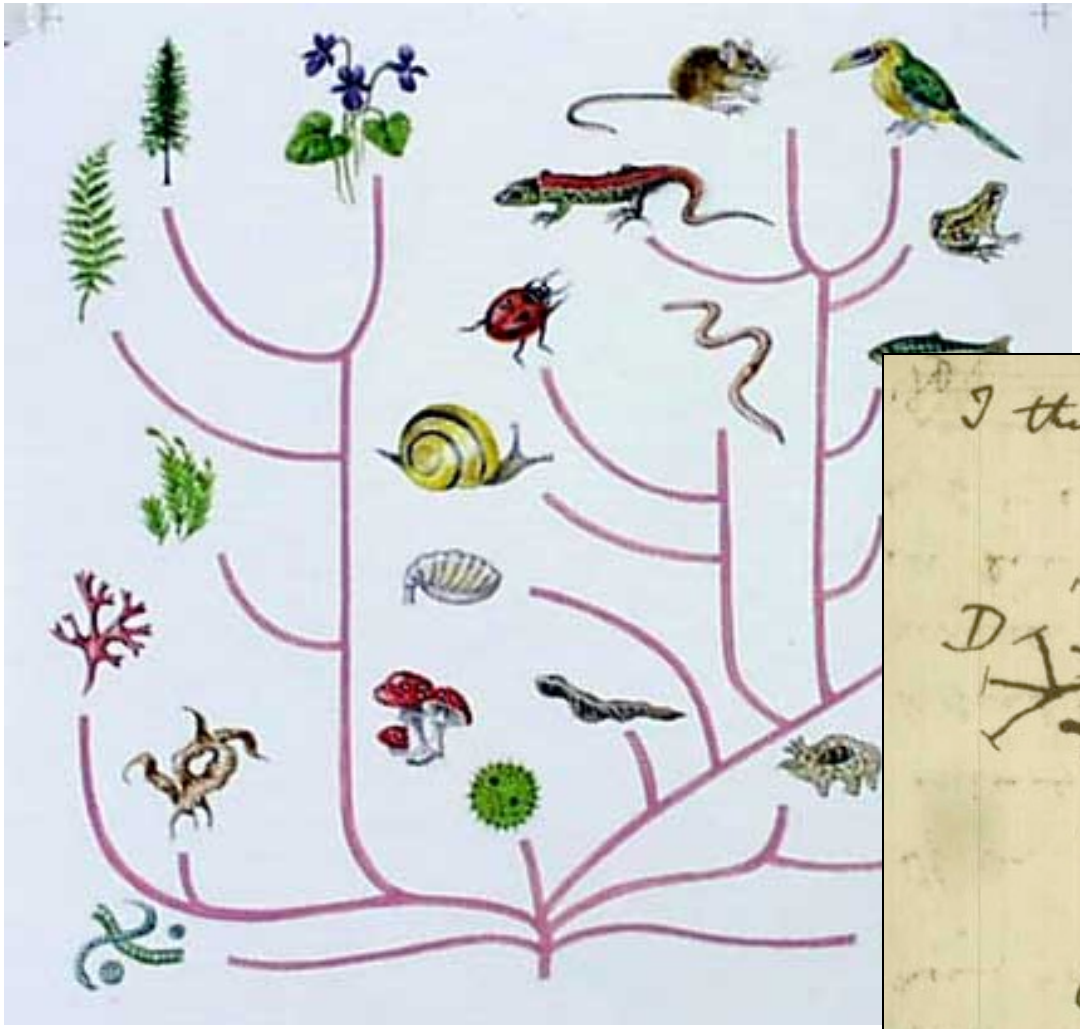


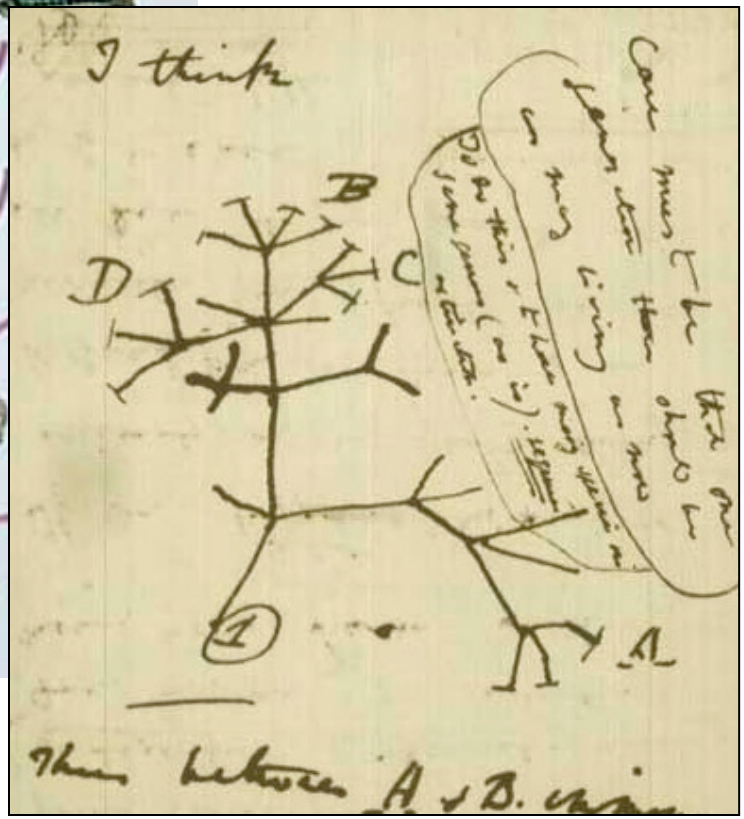
Syntax Tree



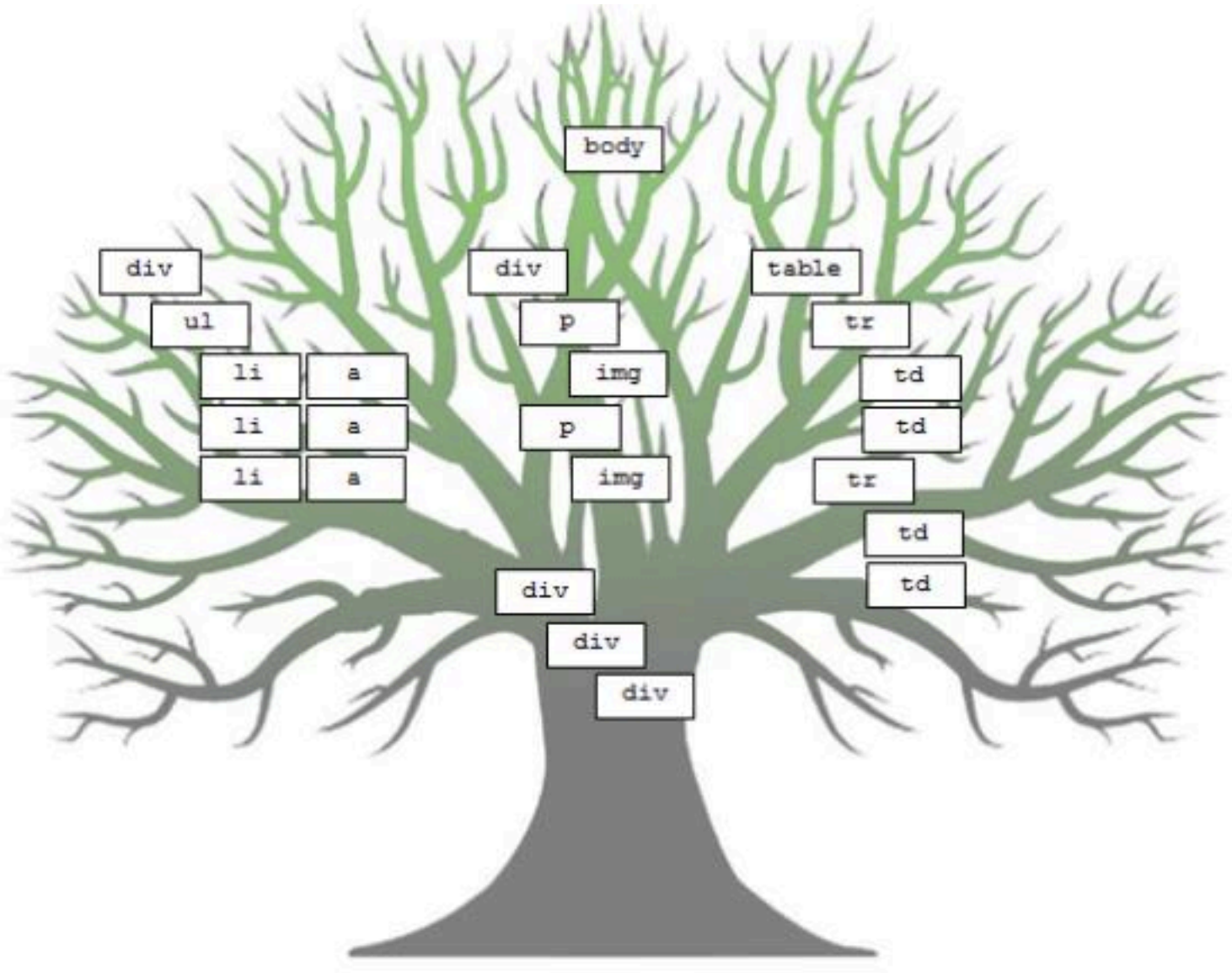
Random expansion from `sentence.txt` *grammar for symbol* `"<s>"`

Animal Tree

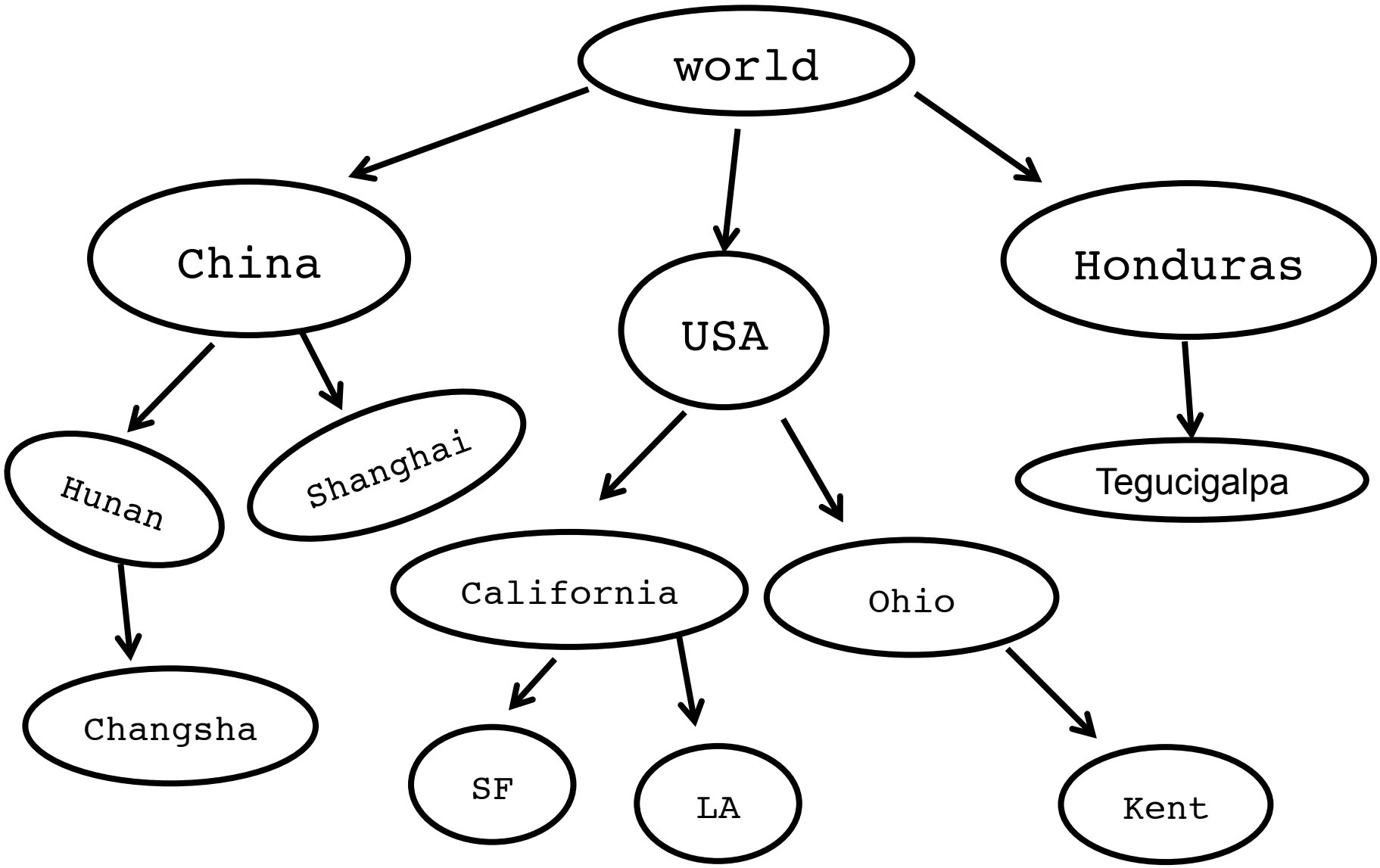




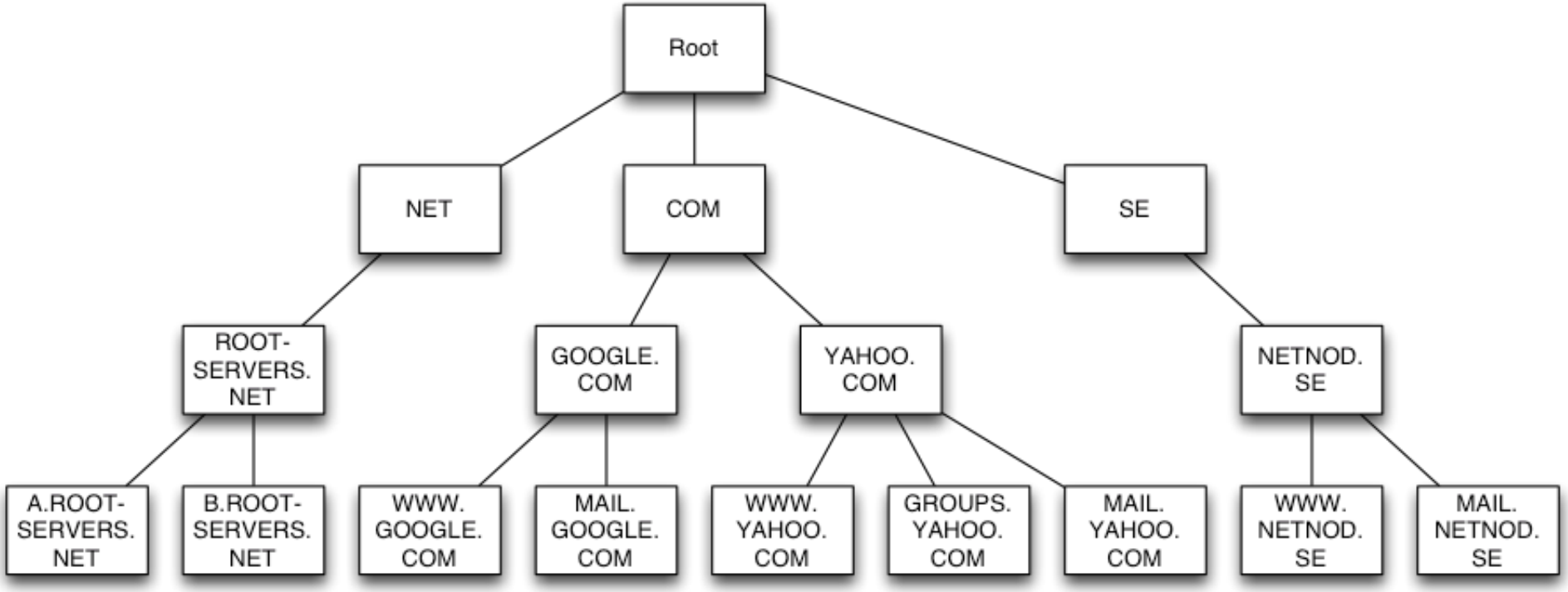
Websites


body
div
ul
li
a
li
a
li
a
div
p
img
p
img
div
div
div
table
tr
td
td
tr
td
td

Hierarchies

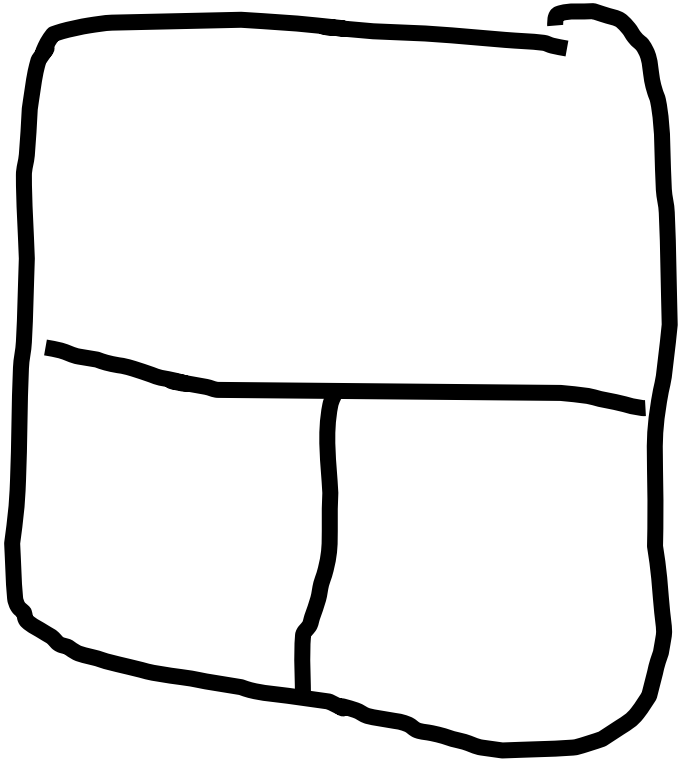


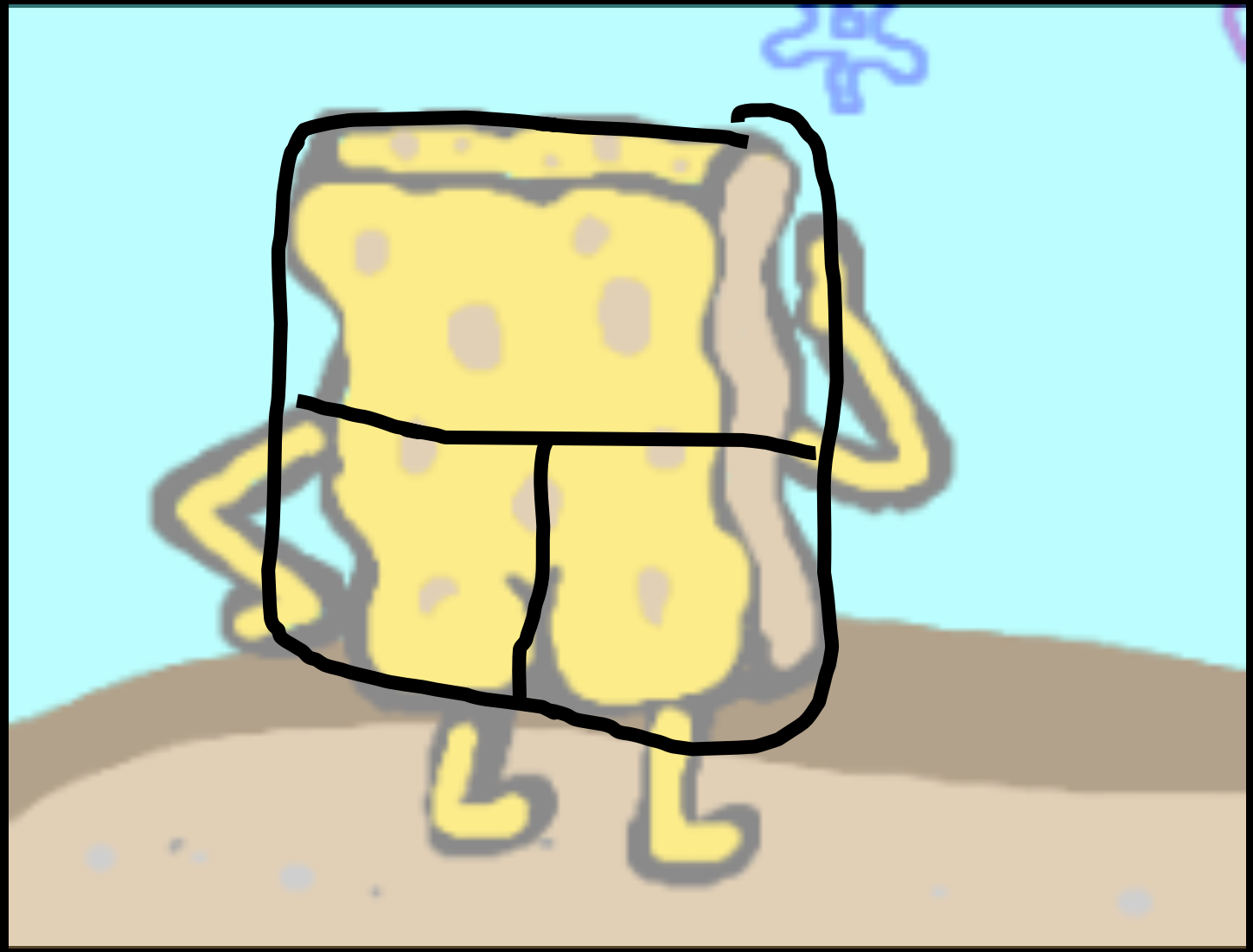
Internet Domains



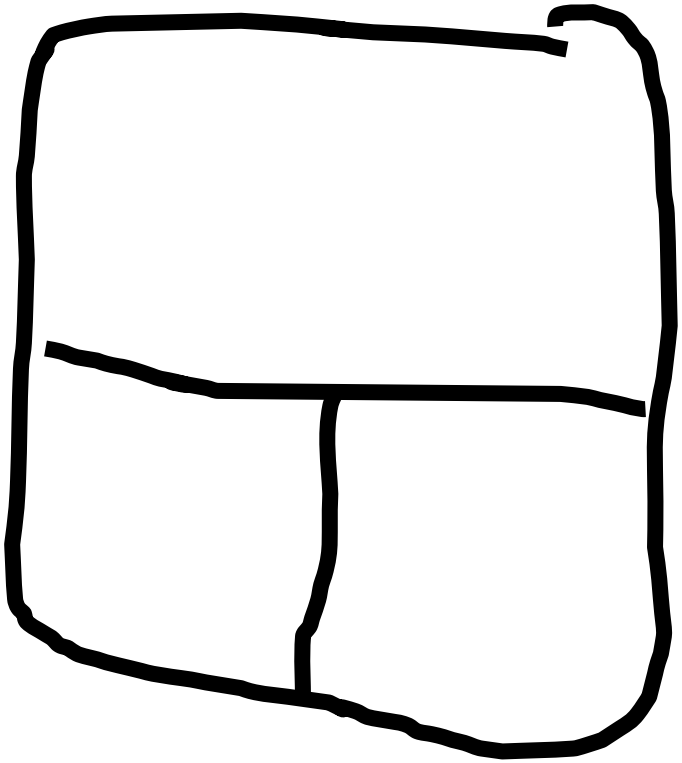
We have seen: Recursion on Trees

But How About Making a Tree Variable?

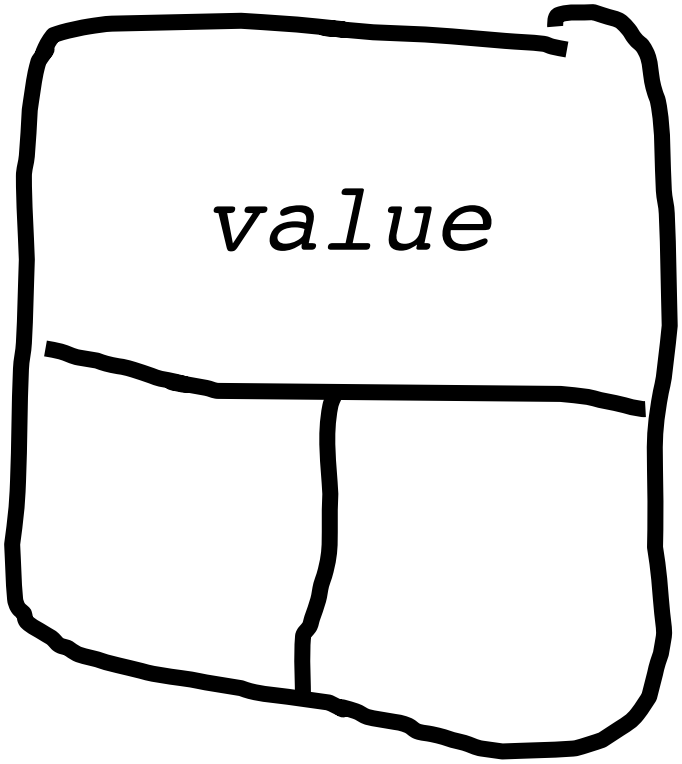




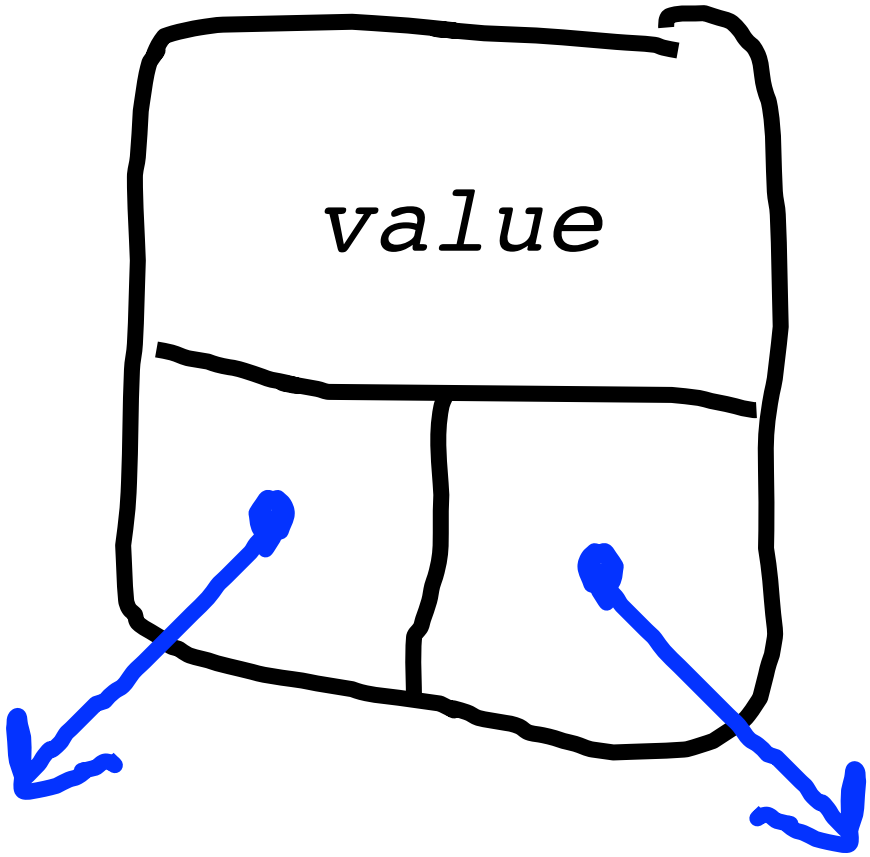
Binary Tree:

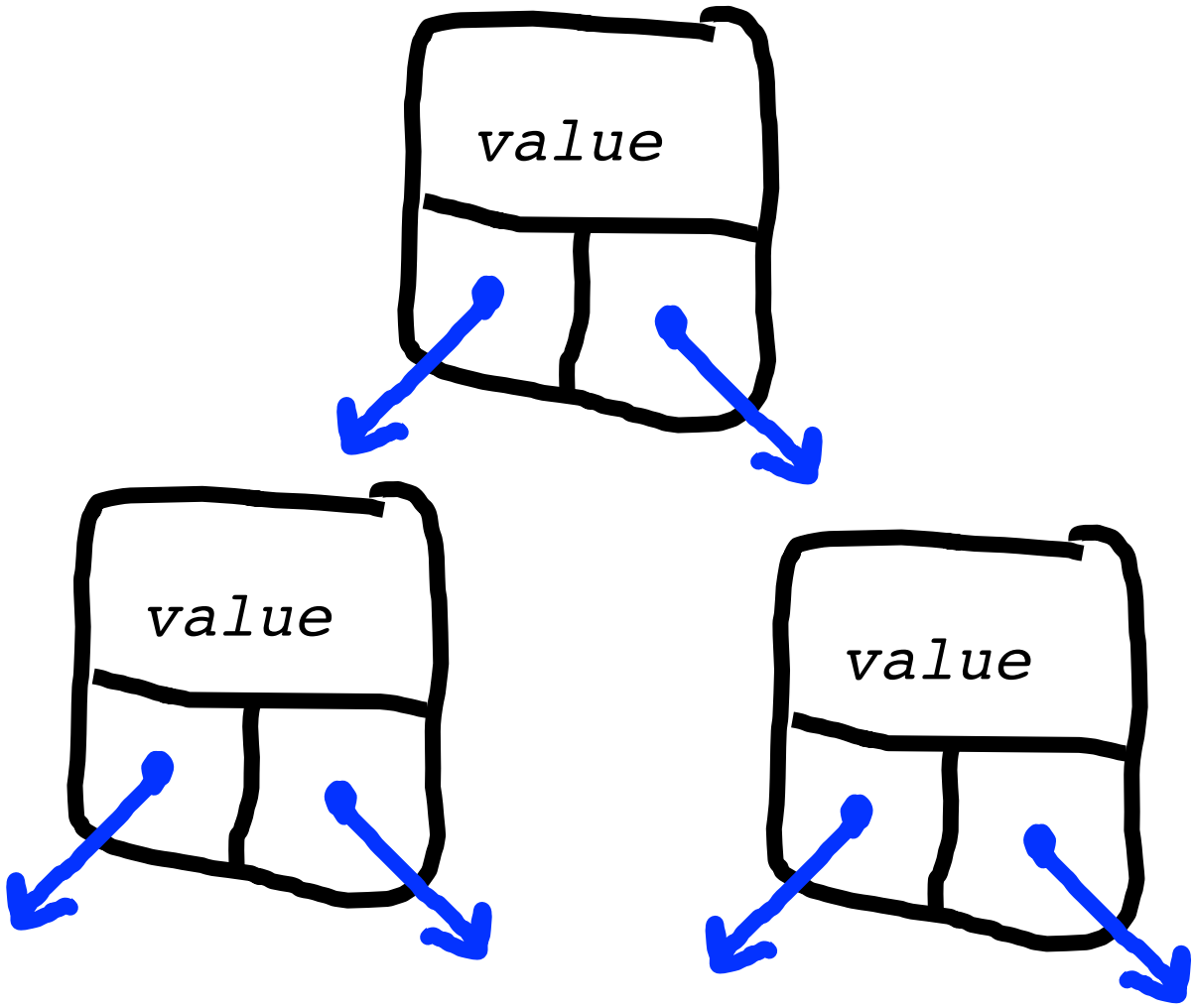


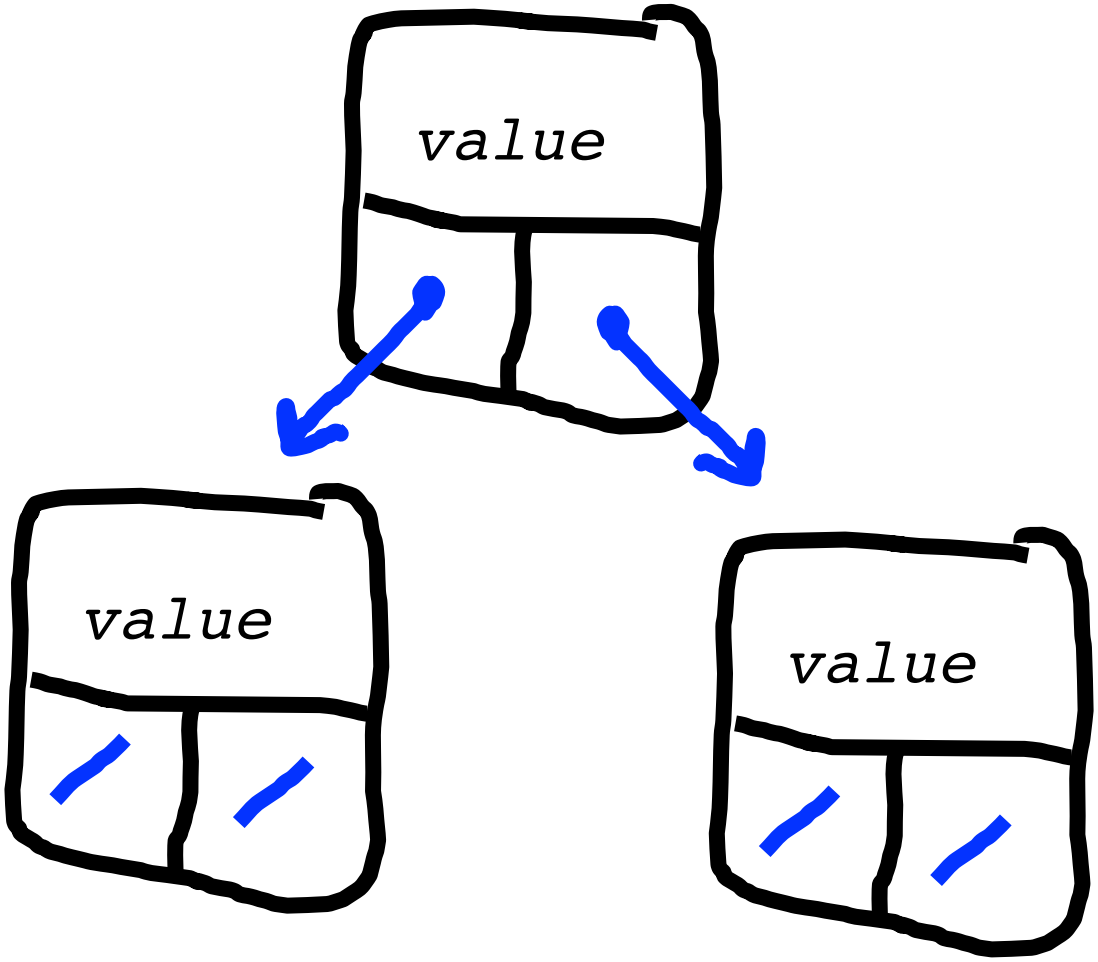
Binary Tree:



Binary Tree:



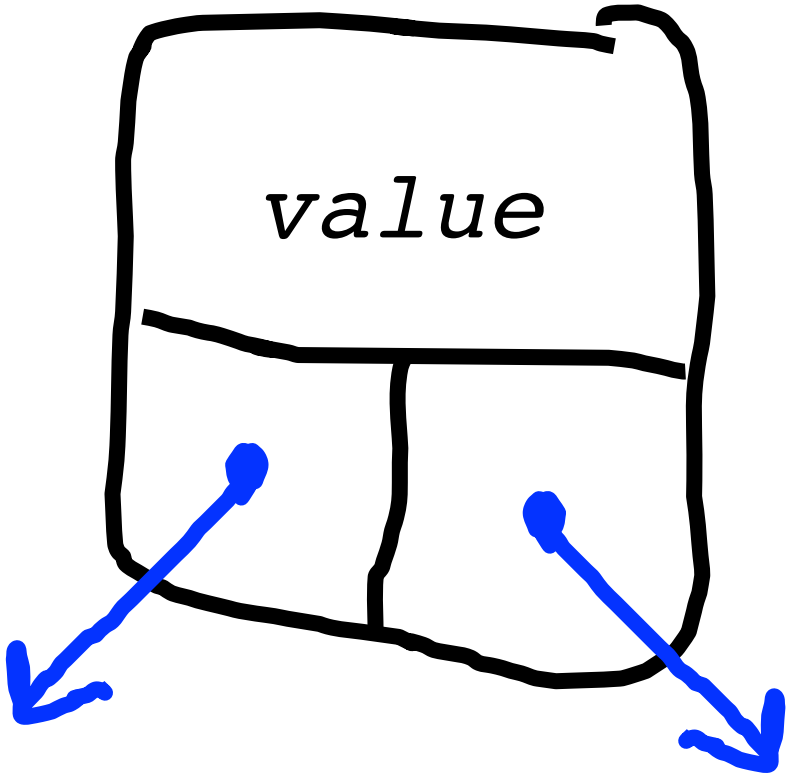

value
value
value


value
value
value

Most Important Slide

Binary Tree

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

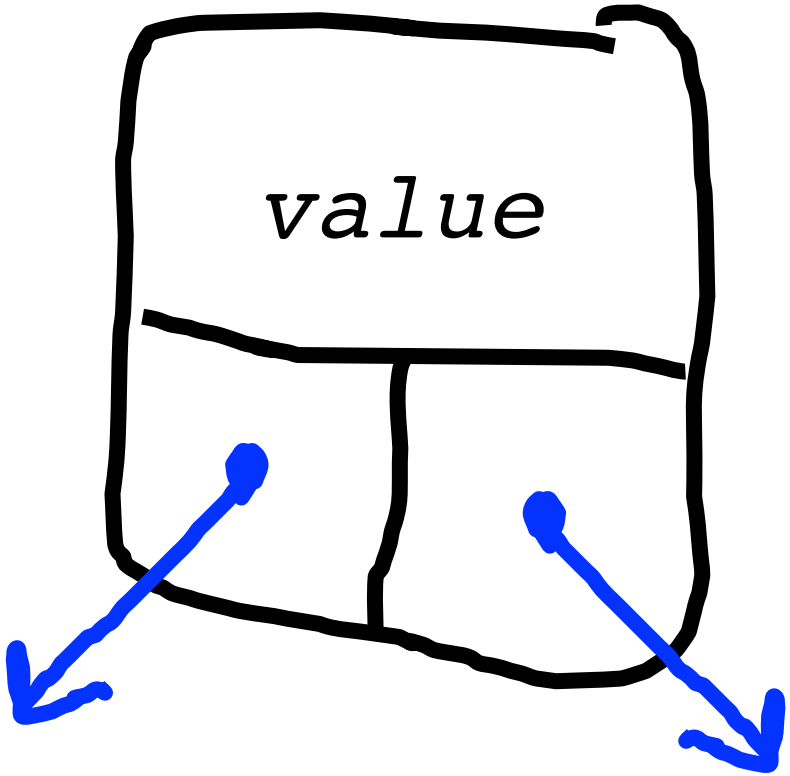


Does value have to be a string?

No

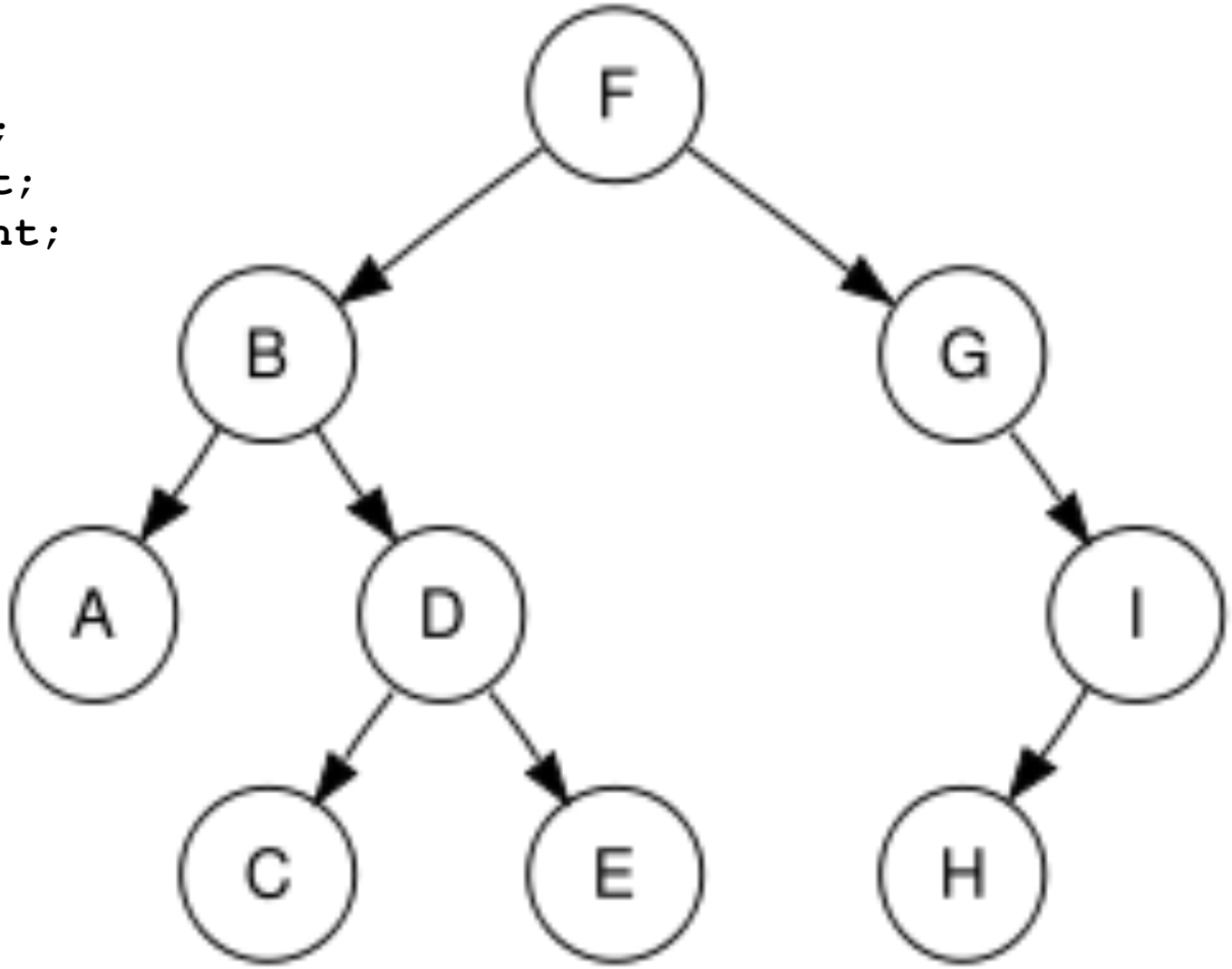
Binary Tree

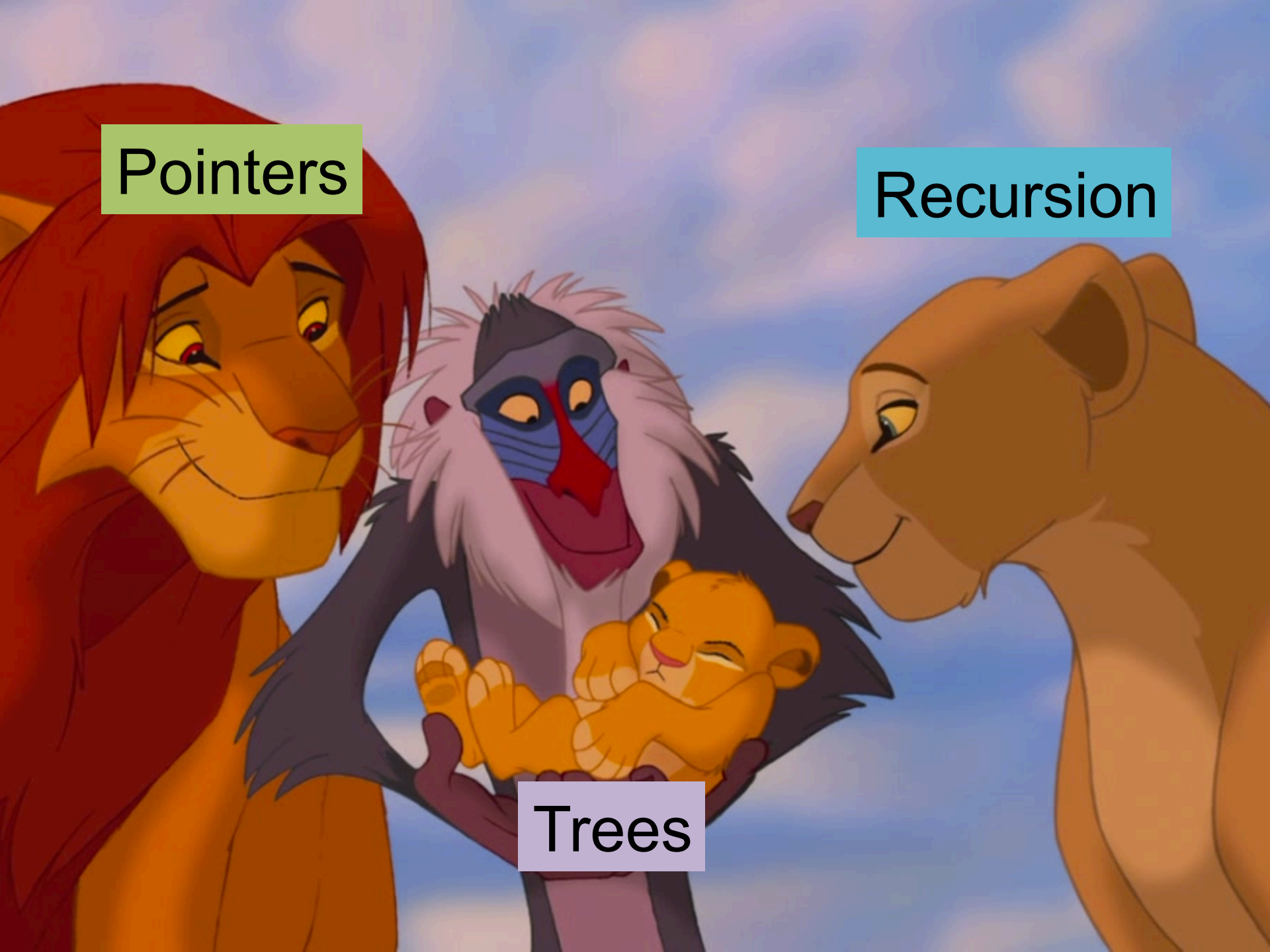
```
struct Tree {
  string value;
  Tree * left;
  Tree * right;
};
```



Example Binary Tree

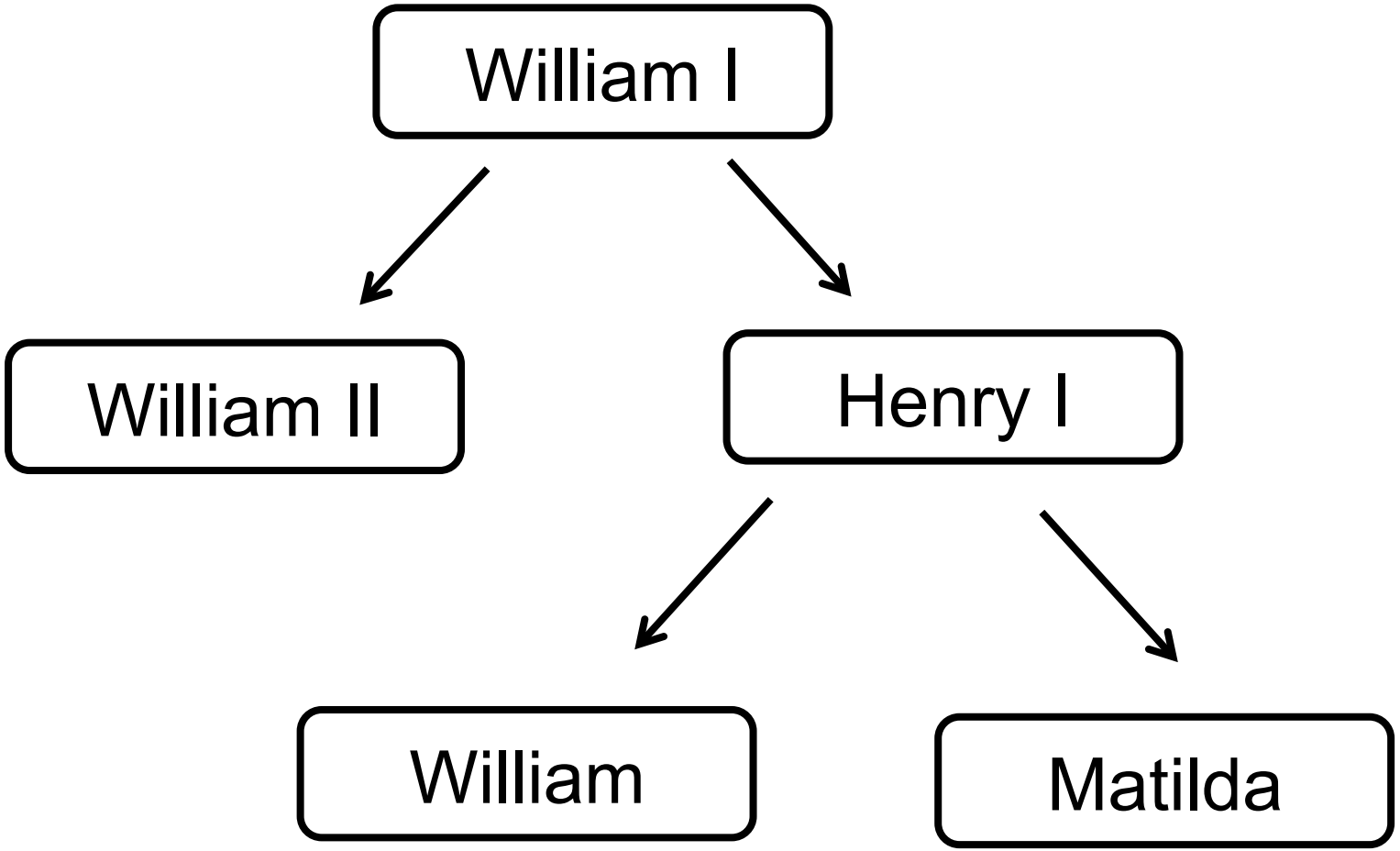
```
struct Tree {
    char value;
    Tree * left;
    Tree * right;
};
```



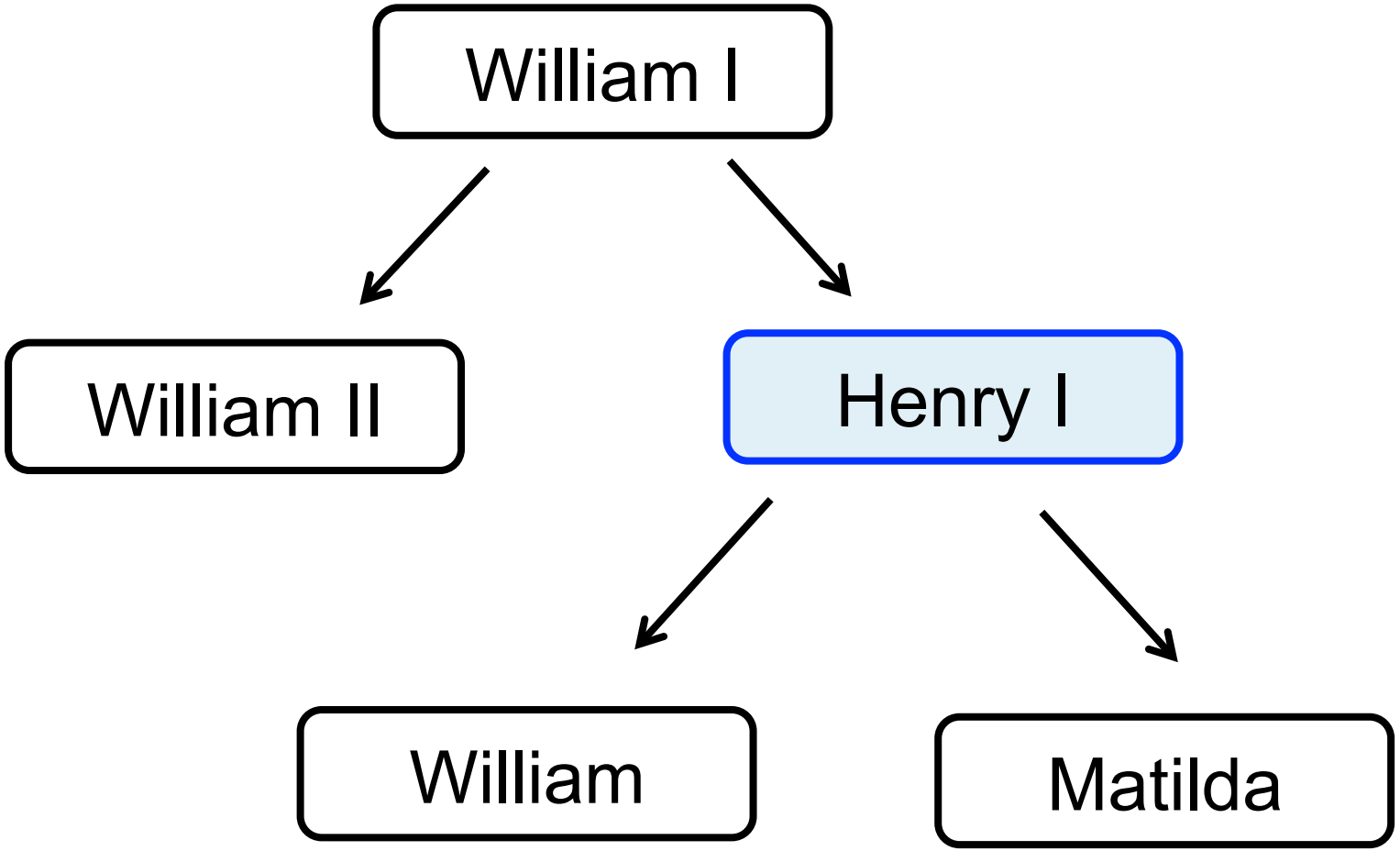

Pointers
Recursion
Trees

Terminology

Terminology

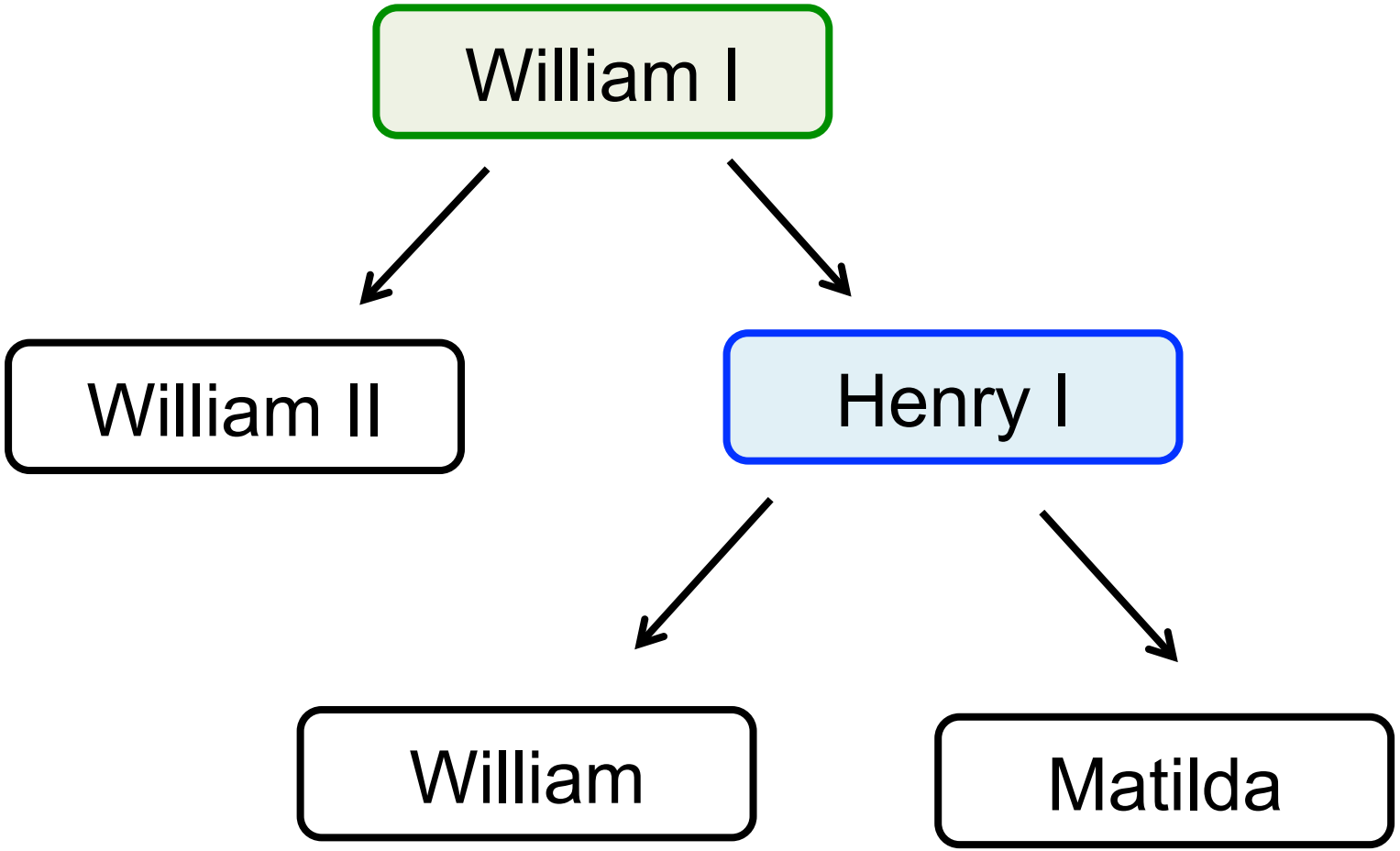


Terminology



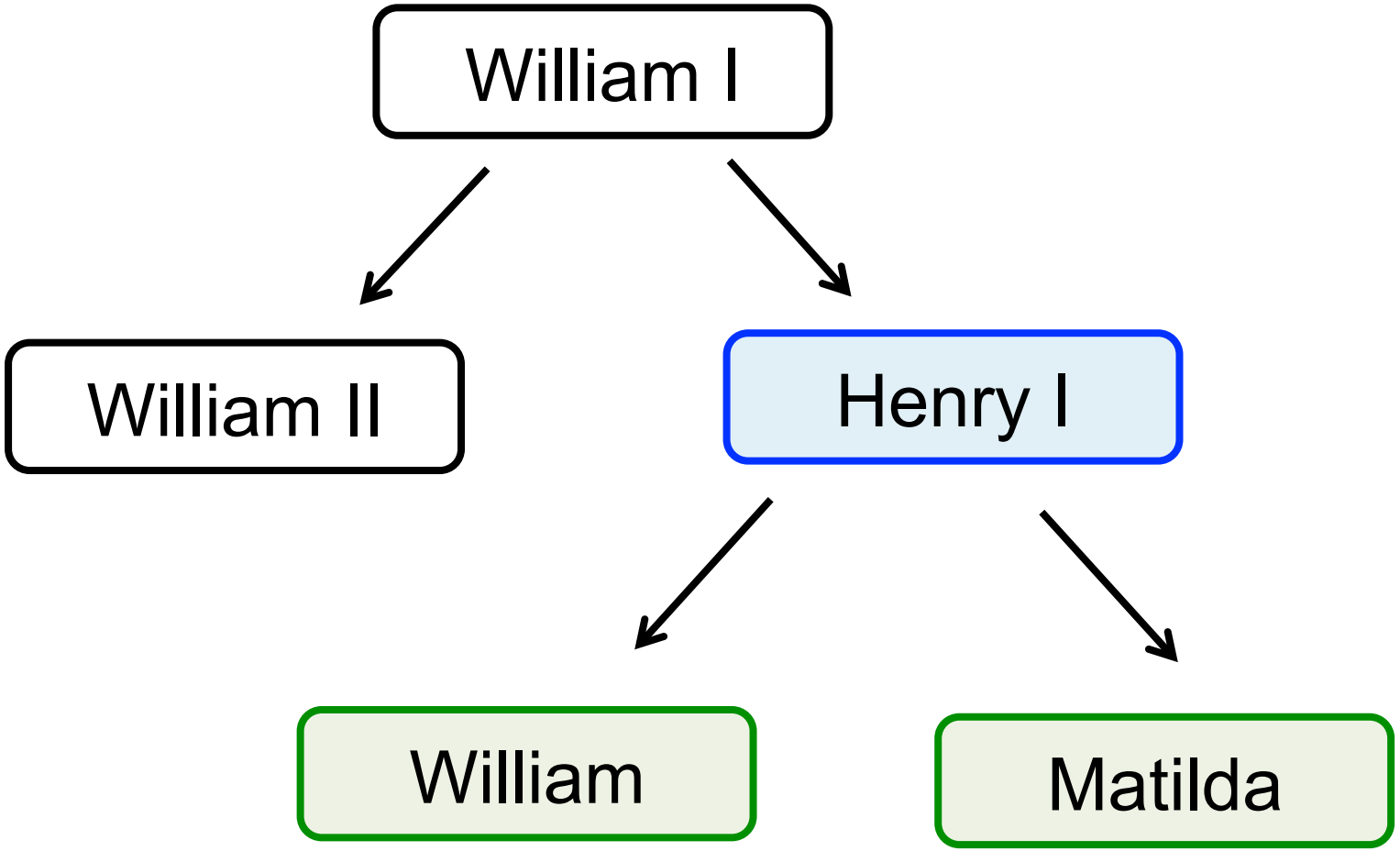
We call a single element a **“Node”** or **“Tree”**

Terminology: Parent



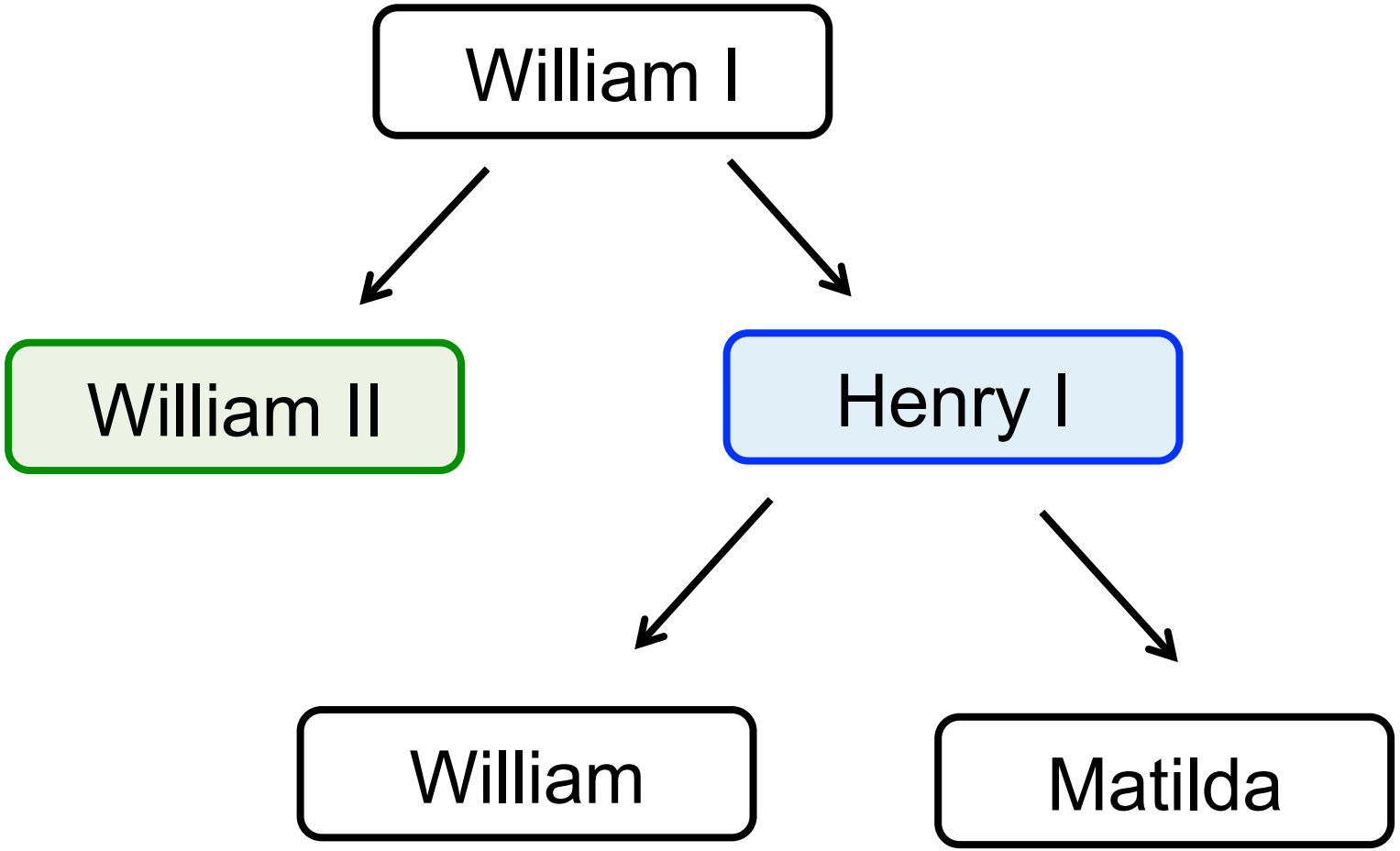
We call the Tree that points to us our **“Parent”**

Terminology: Children



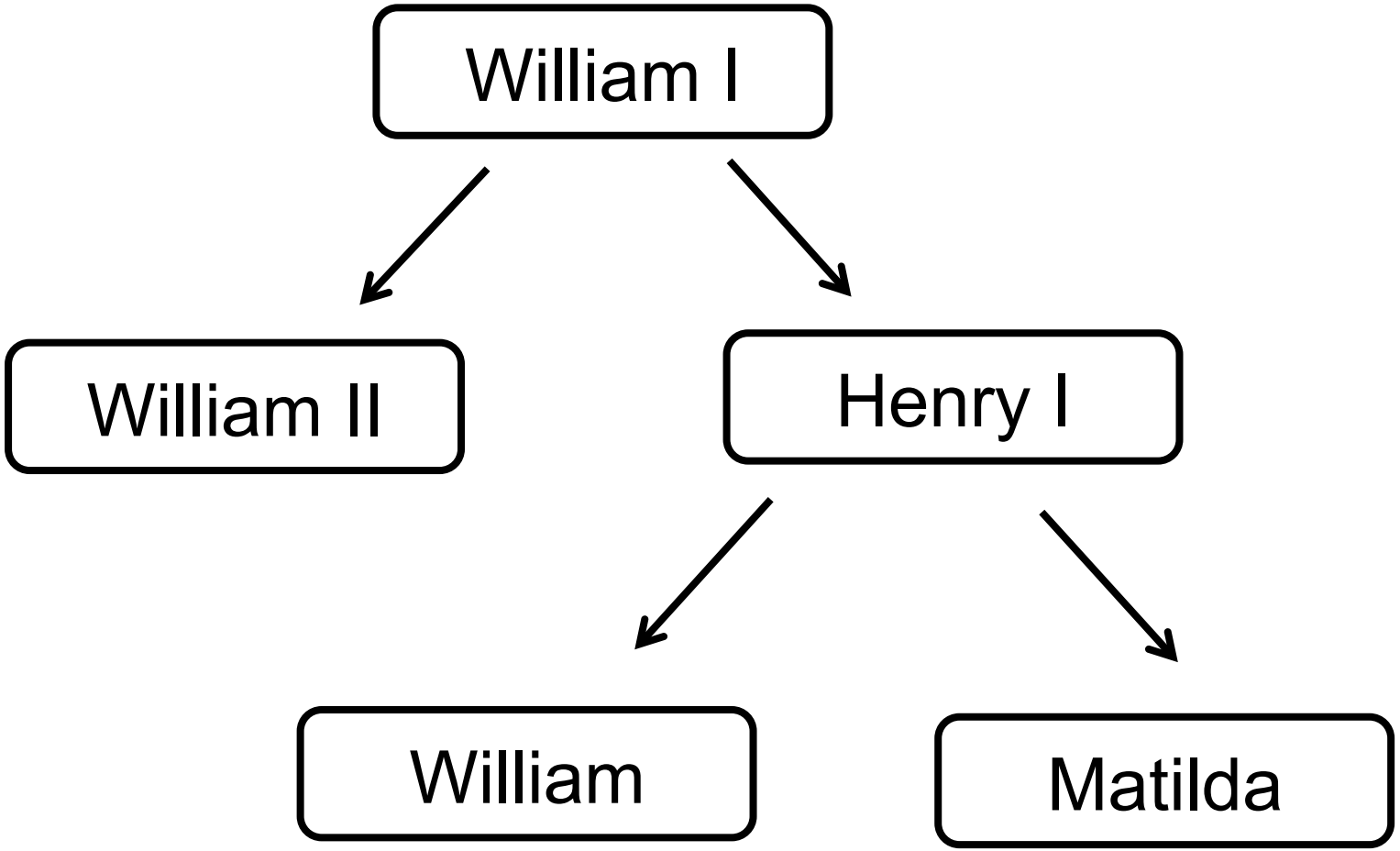
We call the nodes that we point to our **"Children"**

Terminology: Siblings

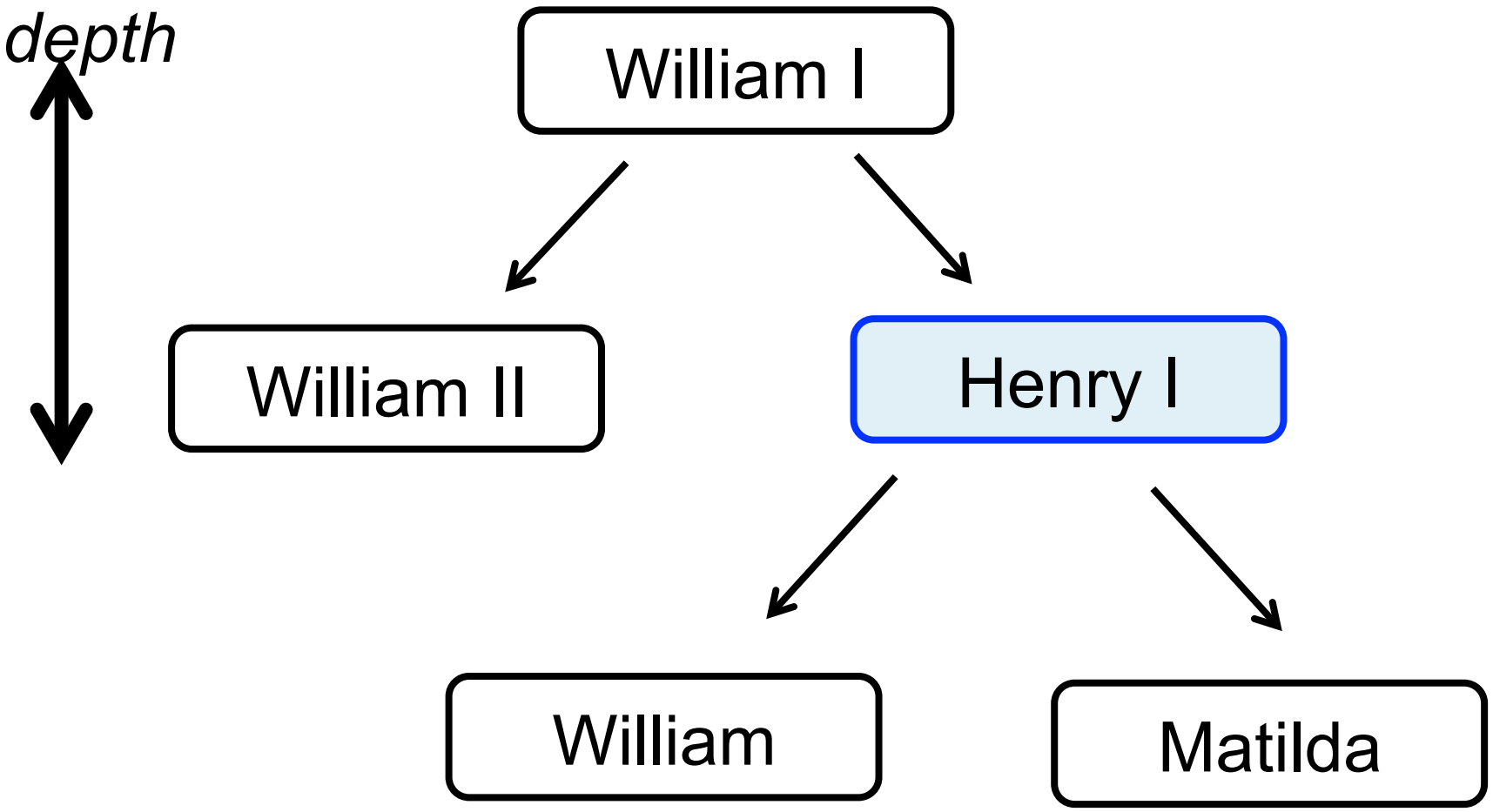


We call nodes that share the same parent **“Siblings”**

Terminology

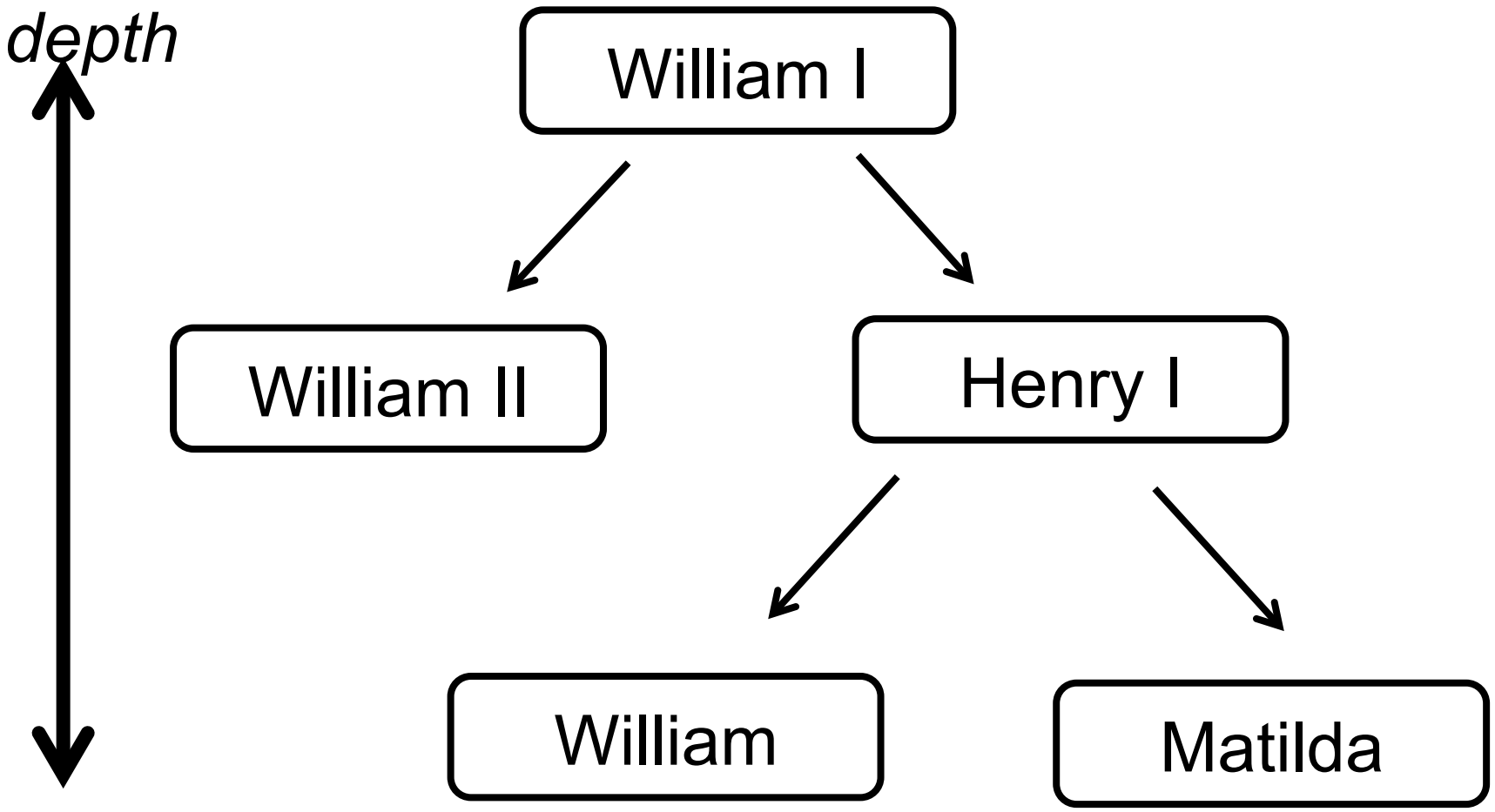


Terminology



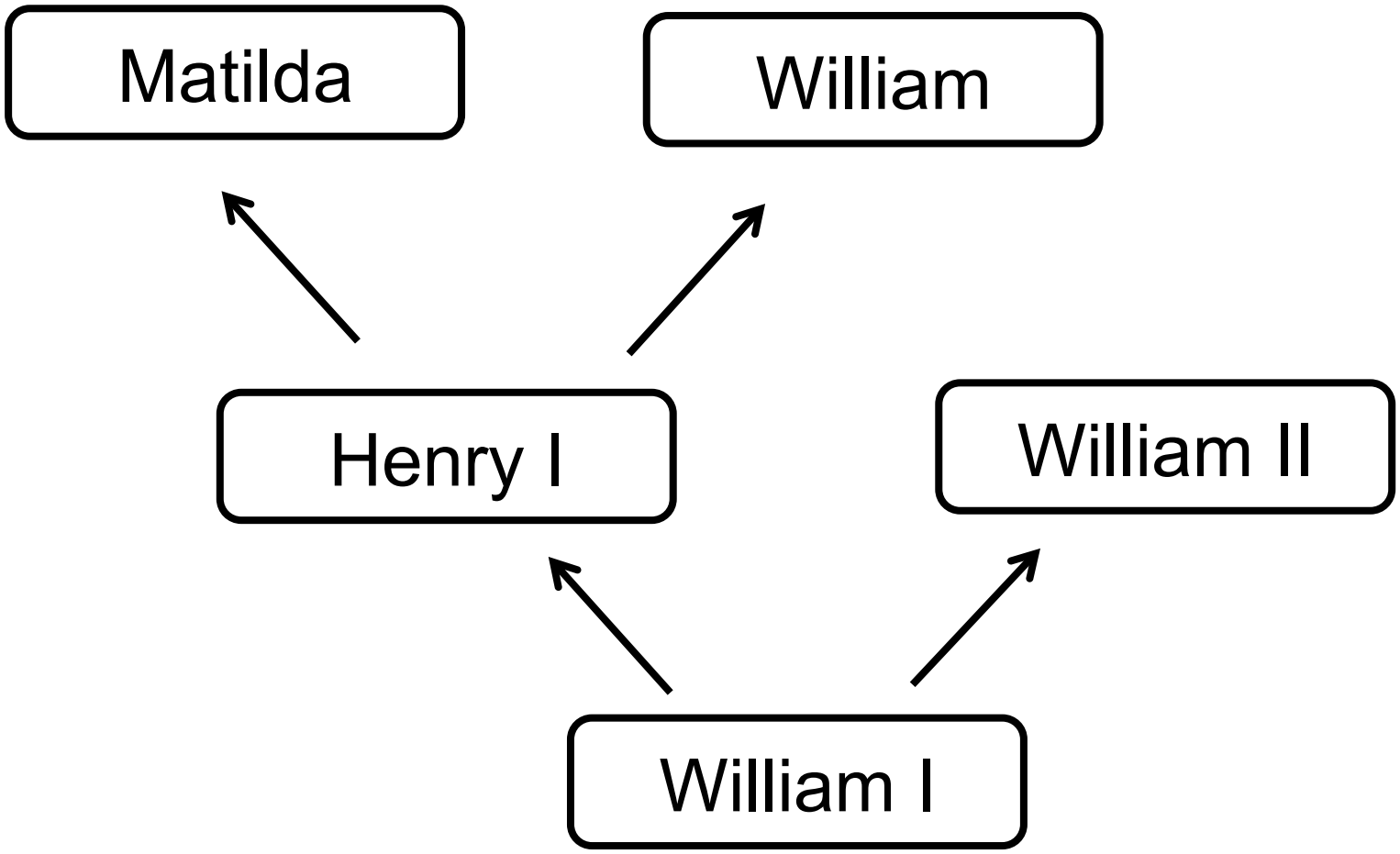
The depth of a node is the number of links from the root.

Terminology

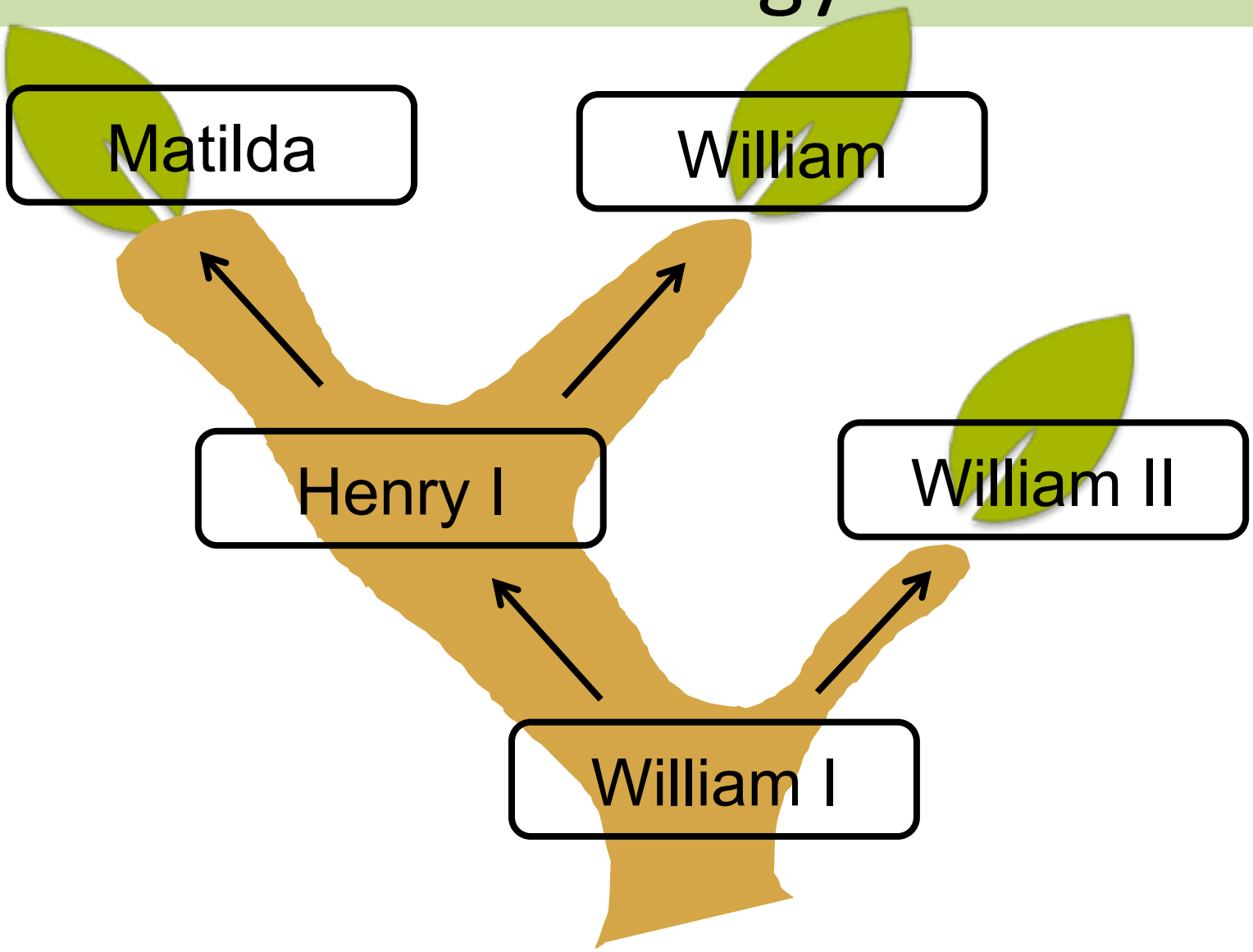


The depth of a tree is the maximum number of links from the root.

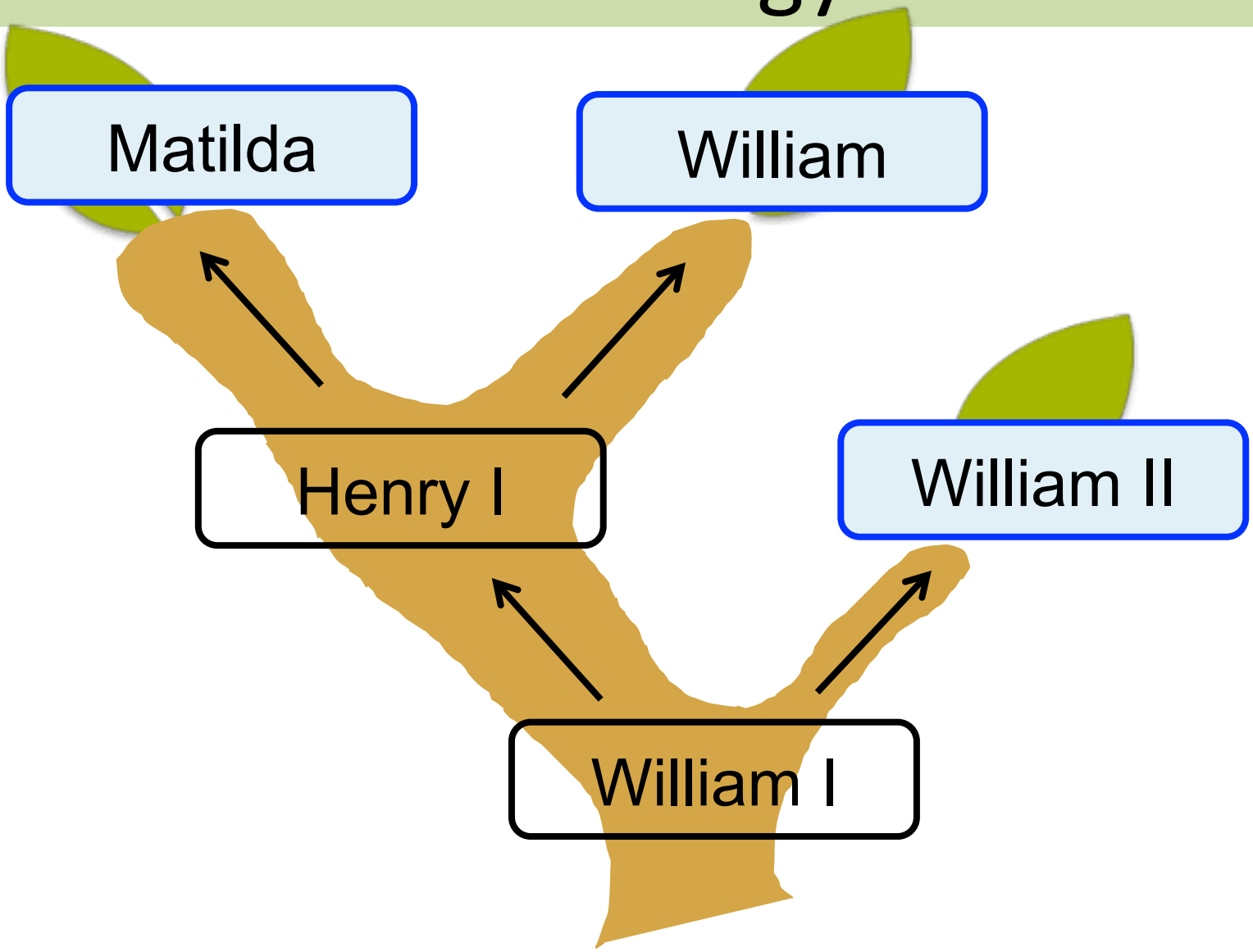
Terminology



Terminology

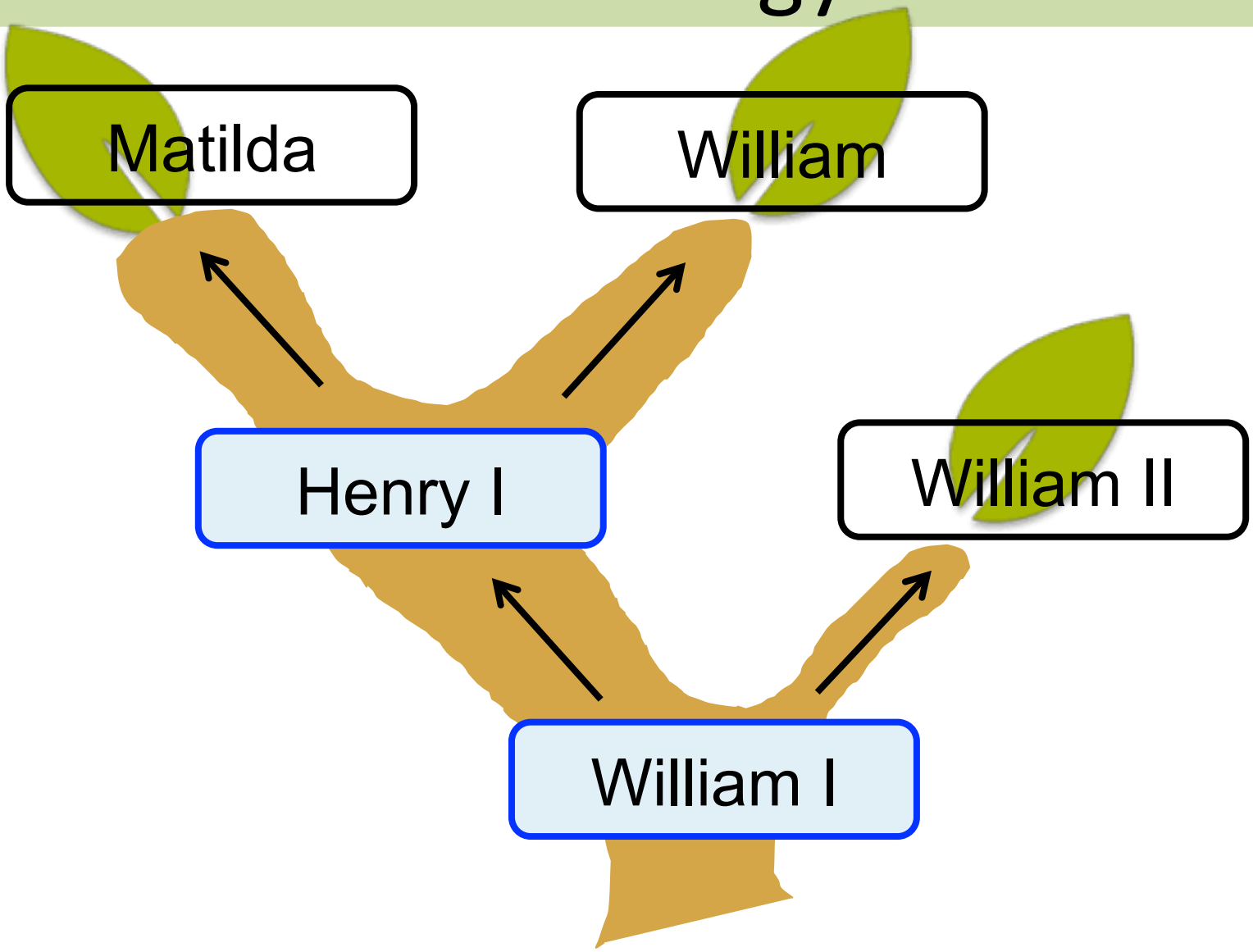


Terminology



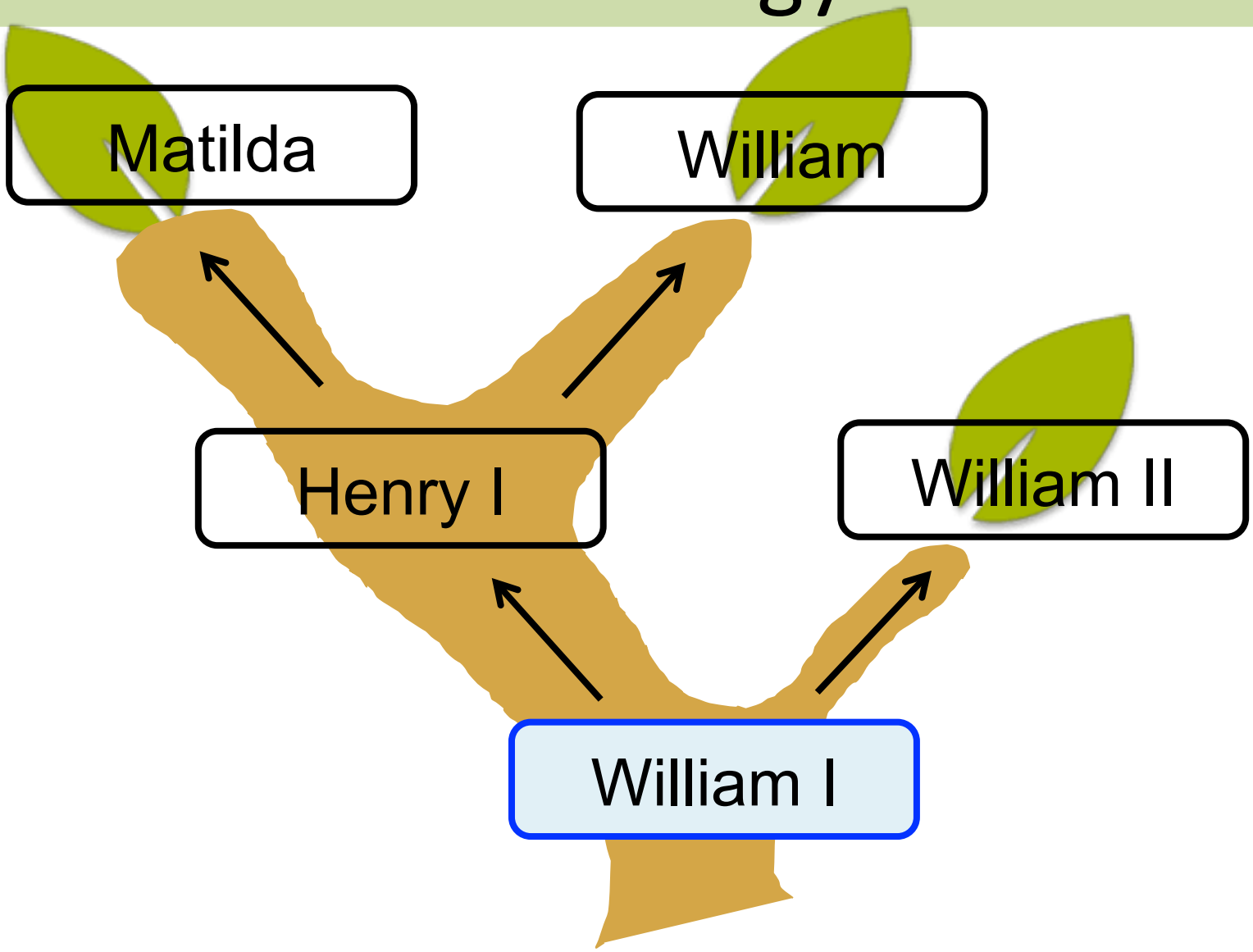
We call nodes that have no children **“Leaves”**

Terminology



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

Terminology



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

Why Binary?

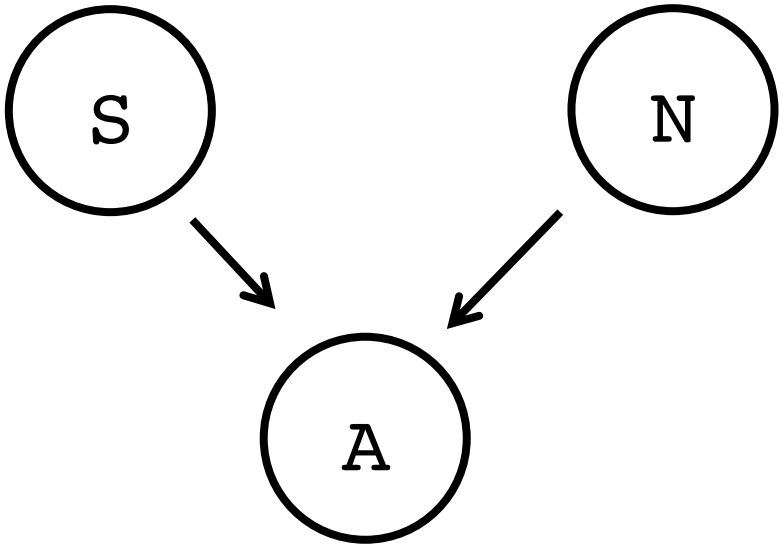
End Terminology

Tree Definition

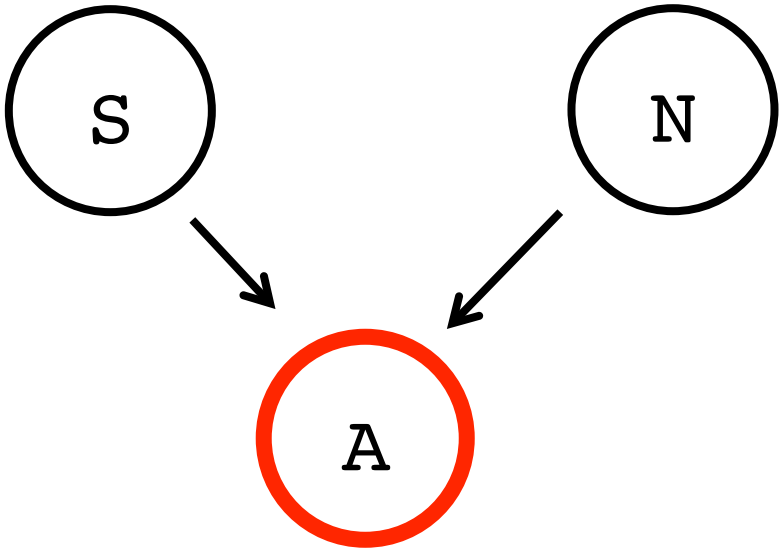
Only One Parent

No Cycles

Only One Parent

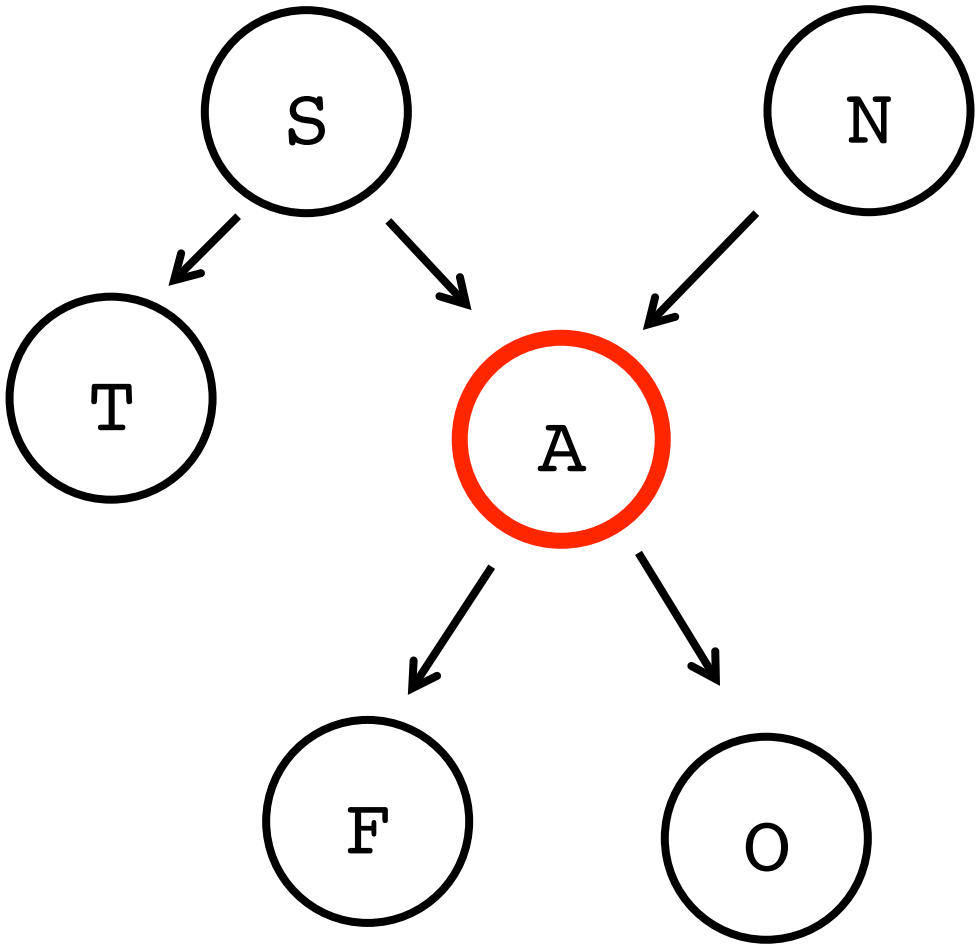


Only One Parent



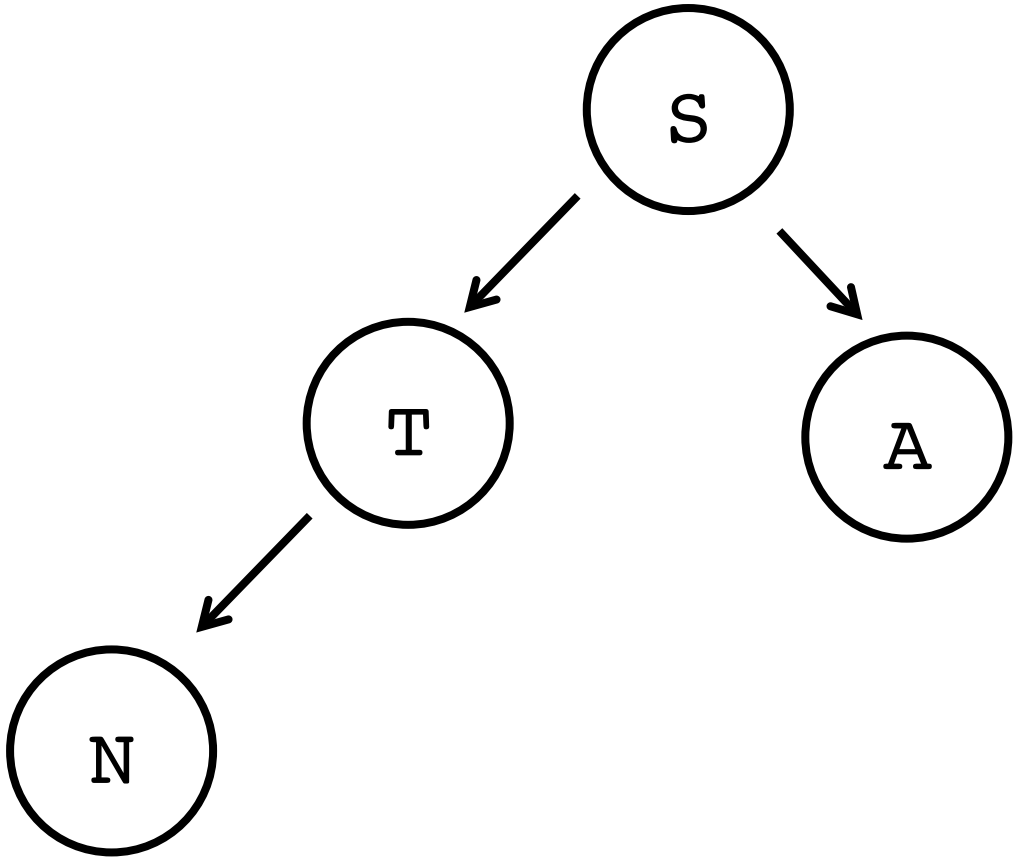
This is not a tree because the red node has two parents

Only One Parent

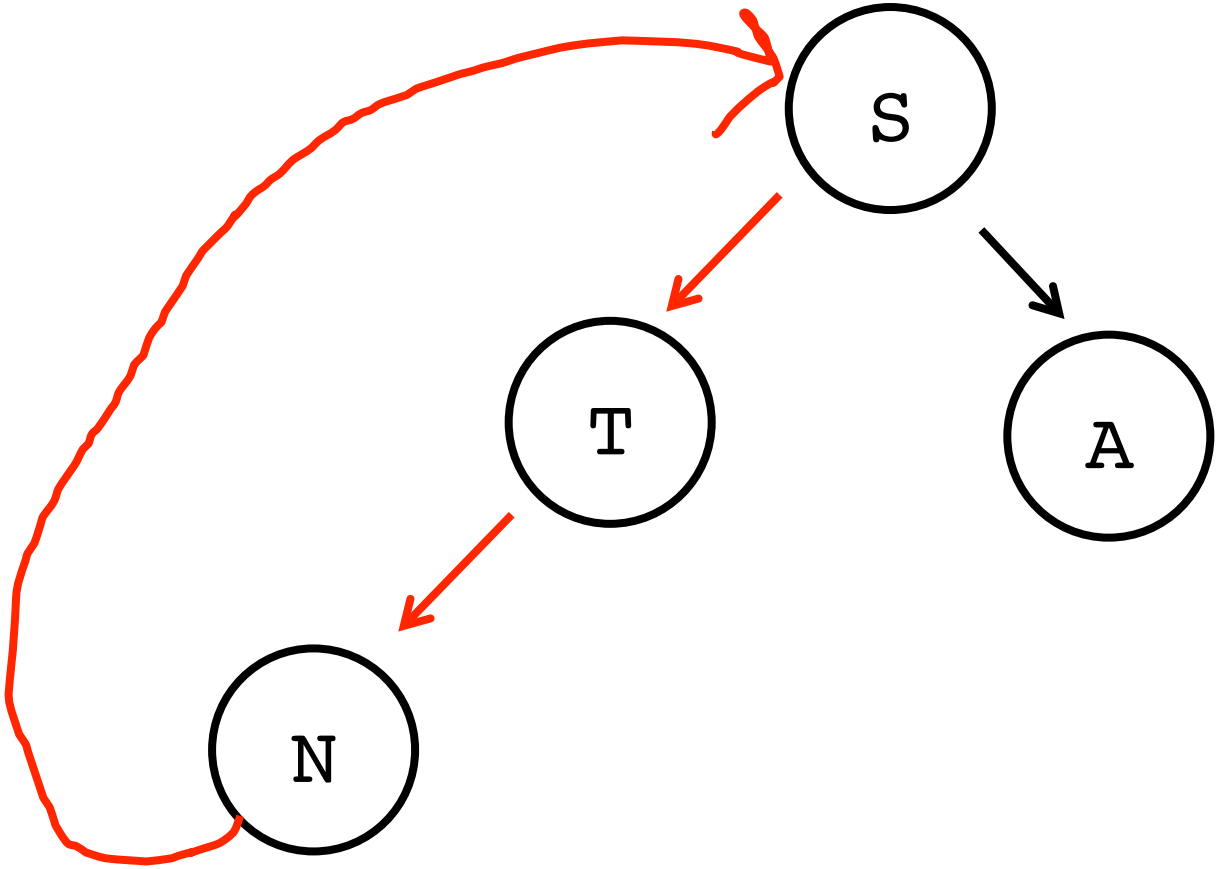


This is not a tree because the red node has two parents

No Cycles

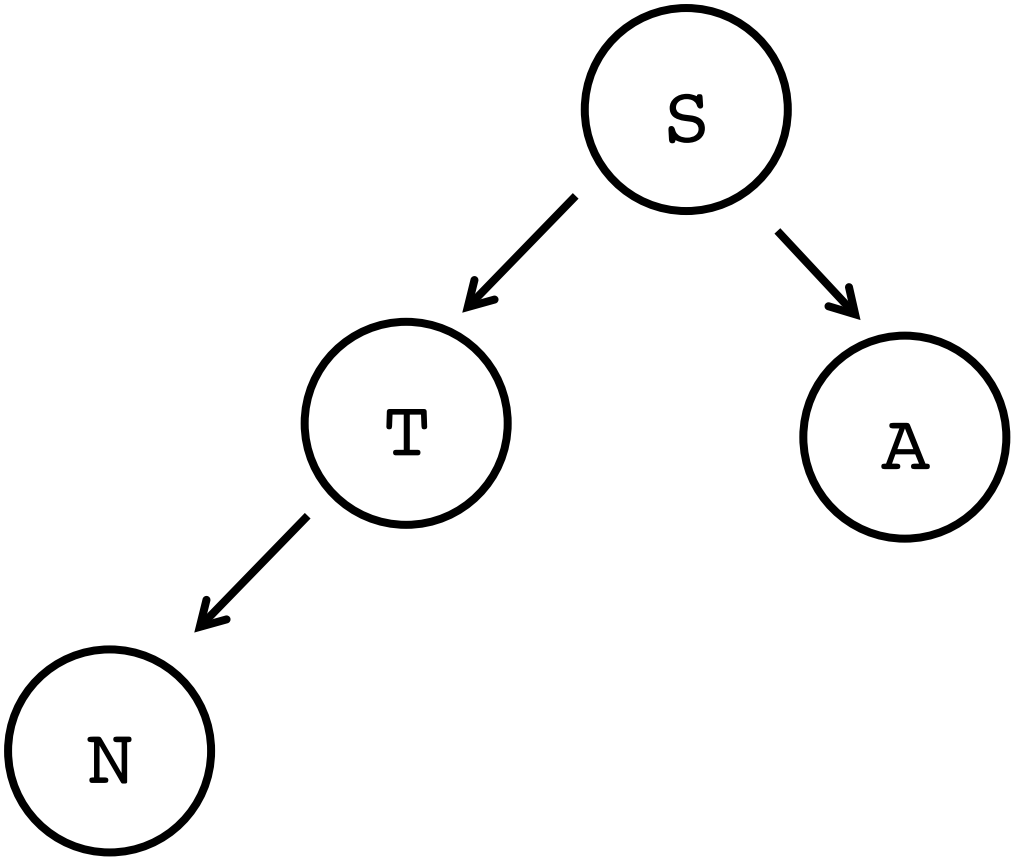


No Cycles

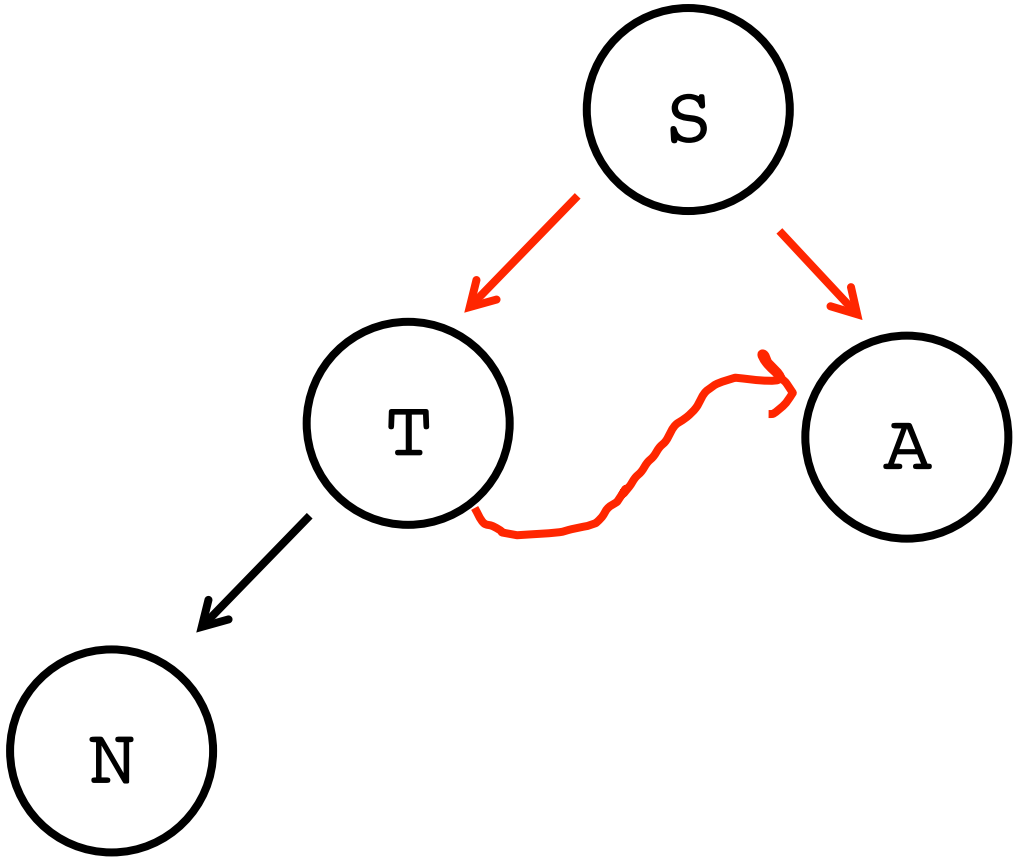


This is not a tree because the red edges make a cycle

No Cycles



No Cycles

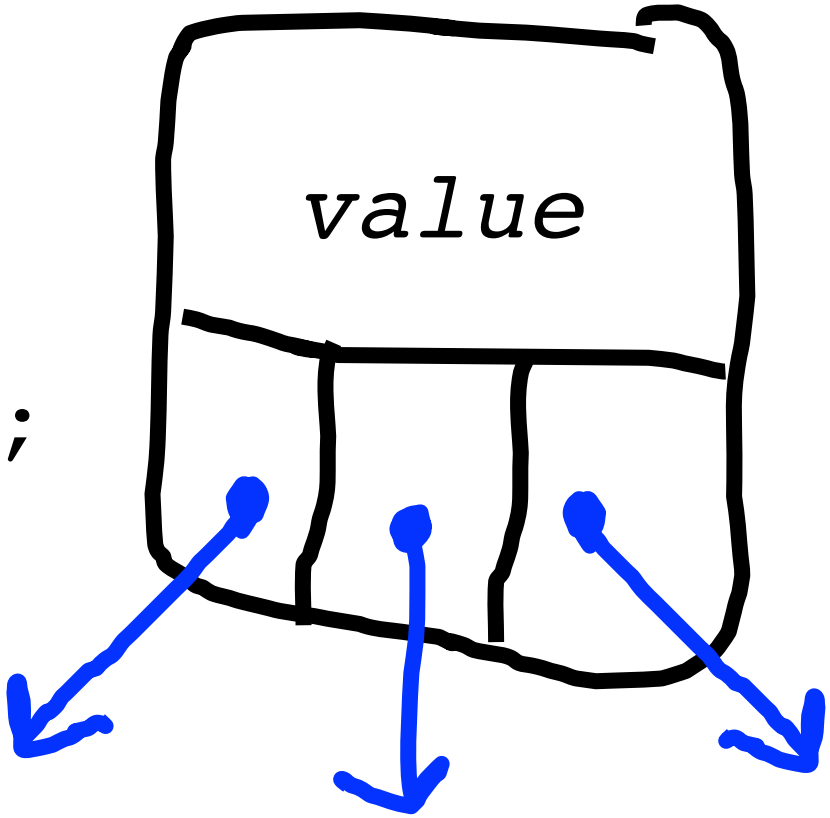


This is not a tree because the red edges make a cycle

Other Kinds of Trees

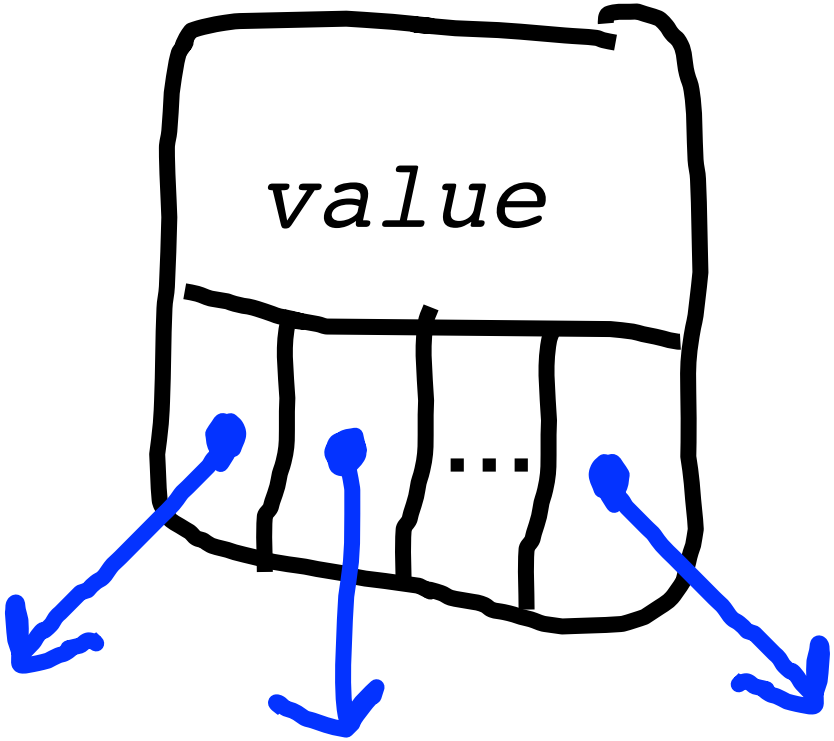
Trinary Tree

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



Tree

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



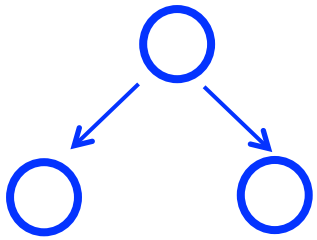
Structs or Classes

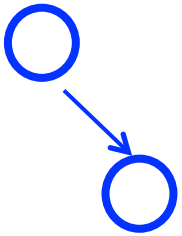
```
struct Tree {
    string value;
    Tree * left;
    Tree * right;
};
```

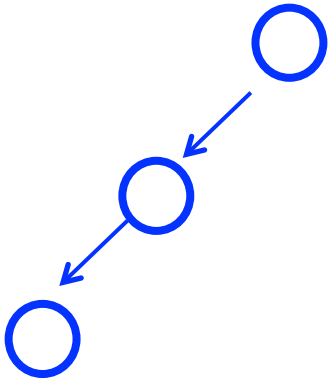
```
class Tree {
private:
    string value;
    Vector<Tree *> children;
};
```

How Many Valid Binary Trees?

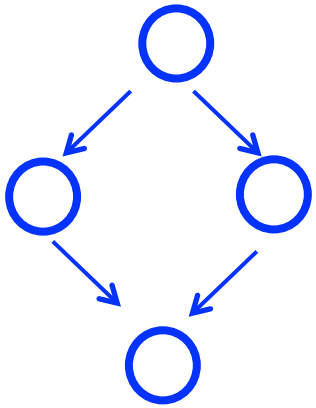
A) 3 **B) 4** **C) 5** **D) 6** **E) 7**

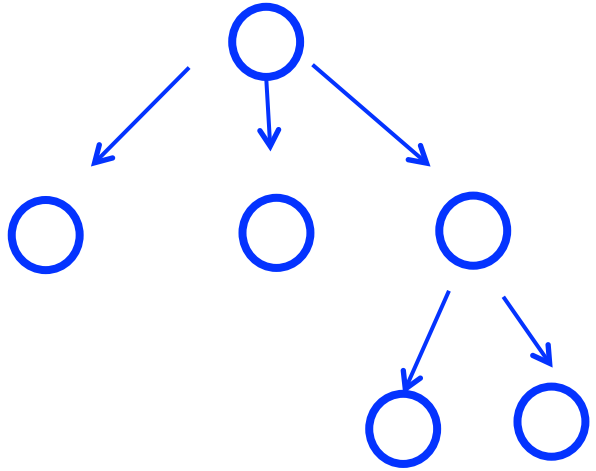






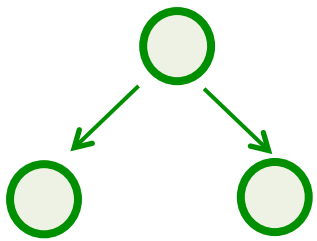
NULL

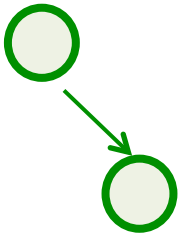


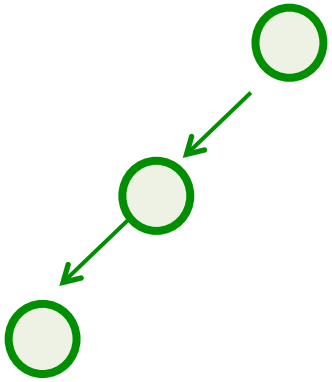


How Many Valid Binary Trees?

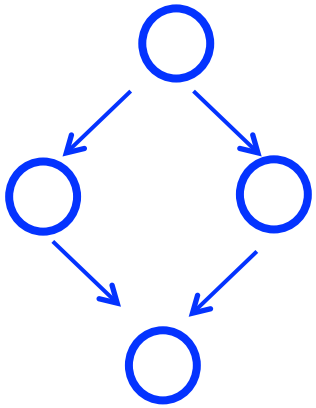
A) 3 **B) 4** **C) 5** **D) 6** **E) 7**

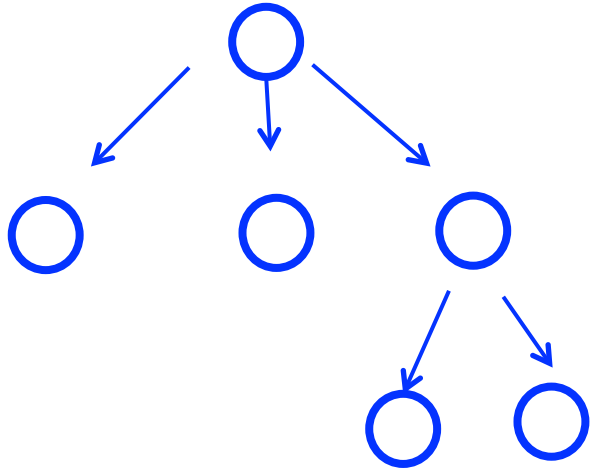






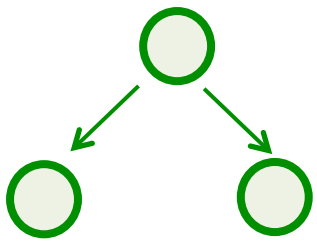
NULL

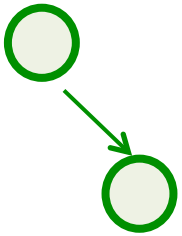


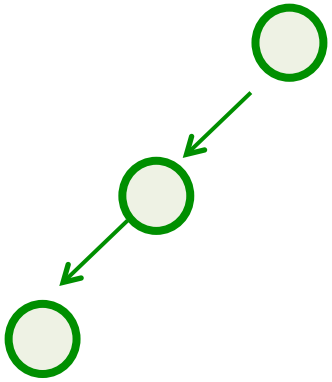


How Many Valid Binary Trees?

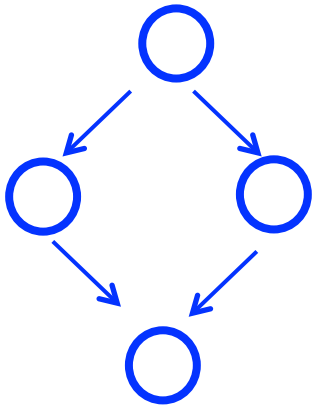
A) 3 **B) 4** **C) 5** **D) 6** **E) 7**

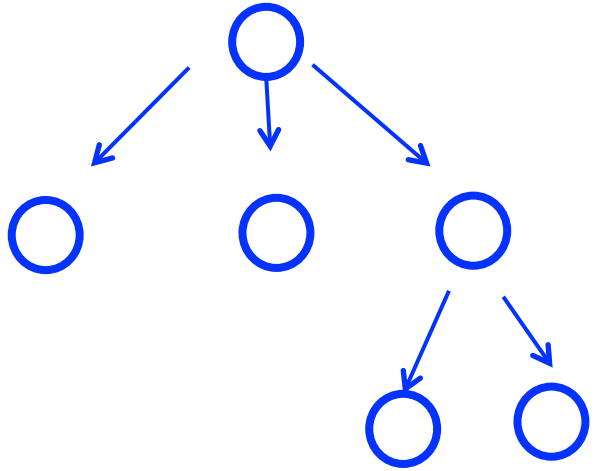






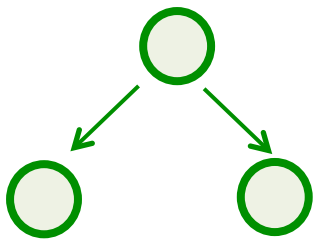
NULL

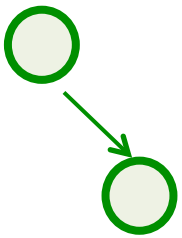


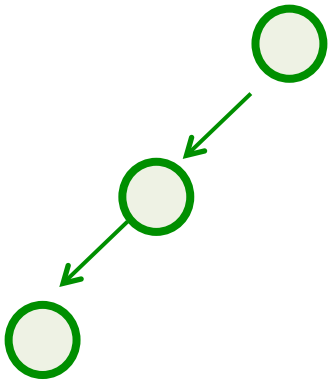


How Many Valid Binary Trees?

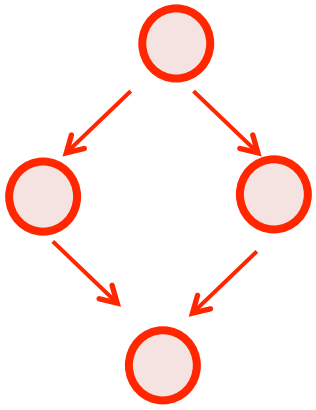
A) 3 **B) 4** **C) 5** **D) 6** **E) 7**

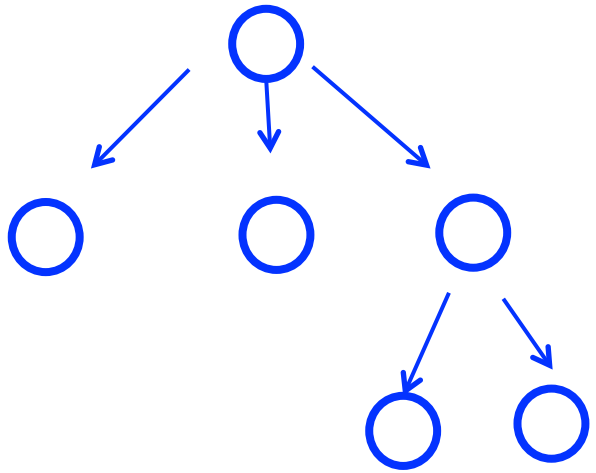






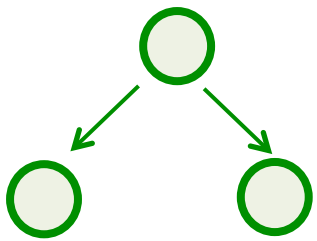
NULL

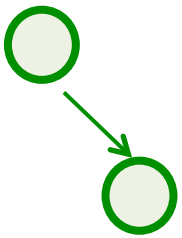


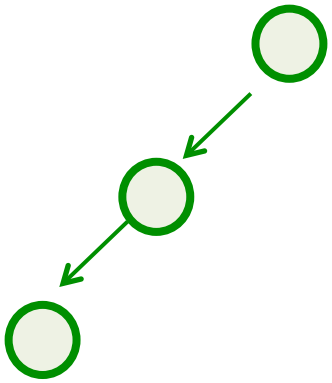


How Many Valid Binary Trees?

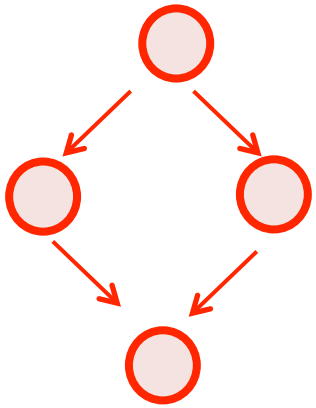
A) 3 **B) 4** **C) 5** **D) 6** **E) 7**

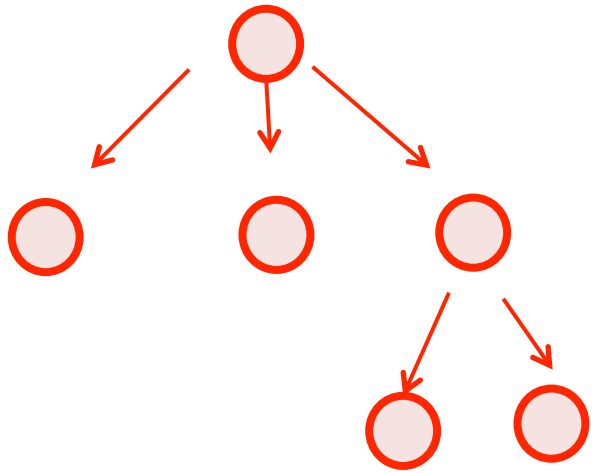






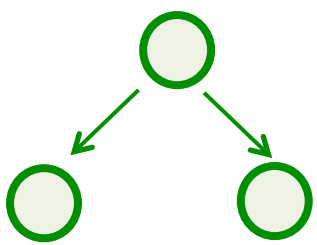
NULL

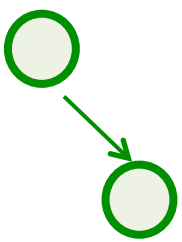


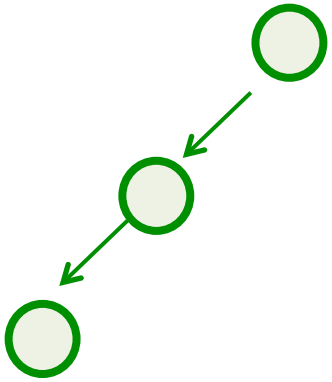


How Many Valid Binary Trees?

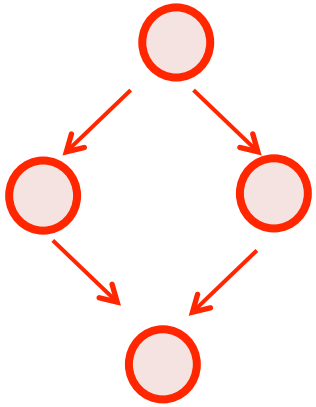
A) 3 **B) 4** **C) 5** **D) 6** **E) 7**

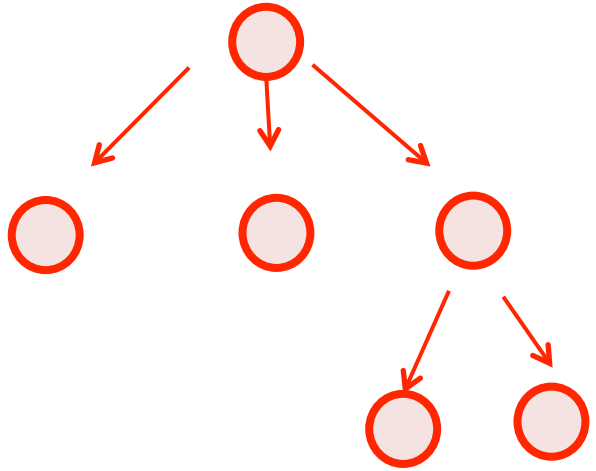




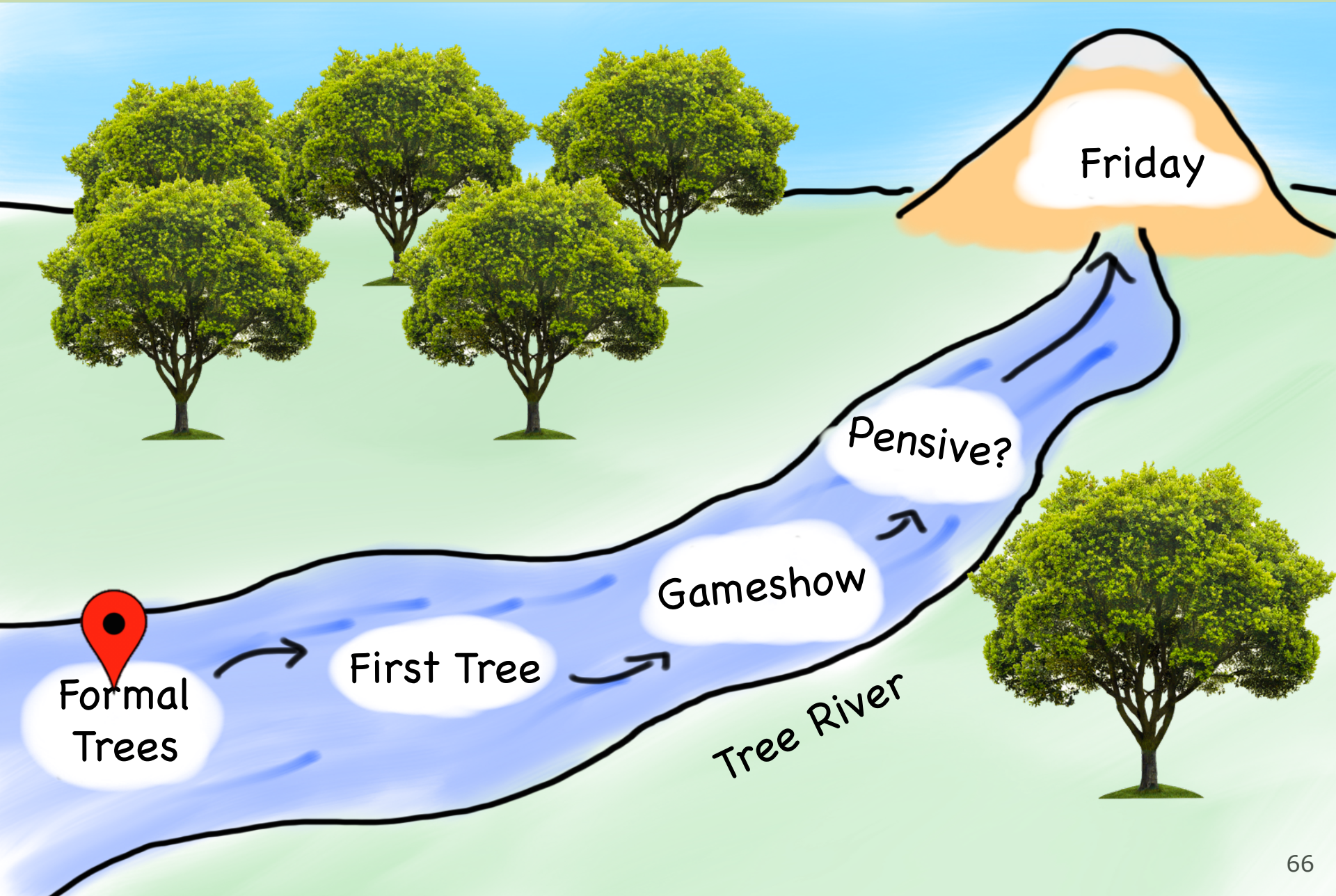


NULL

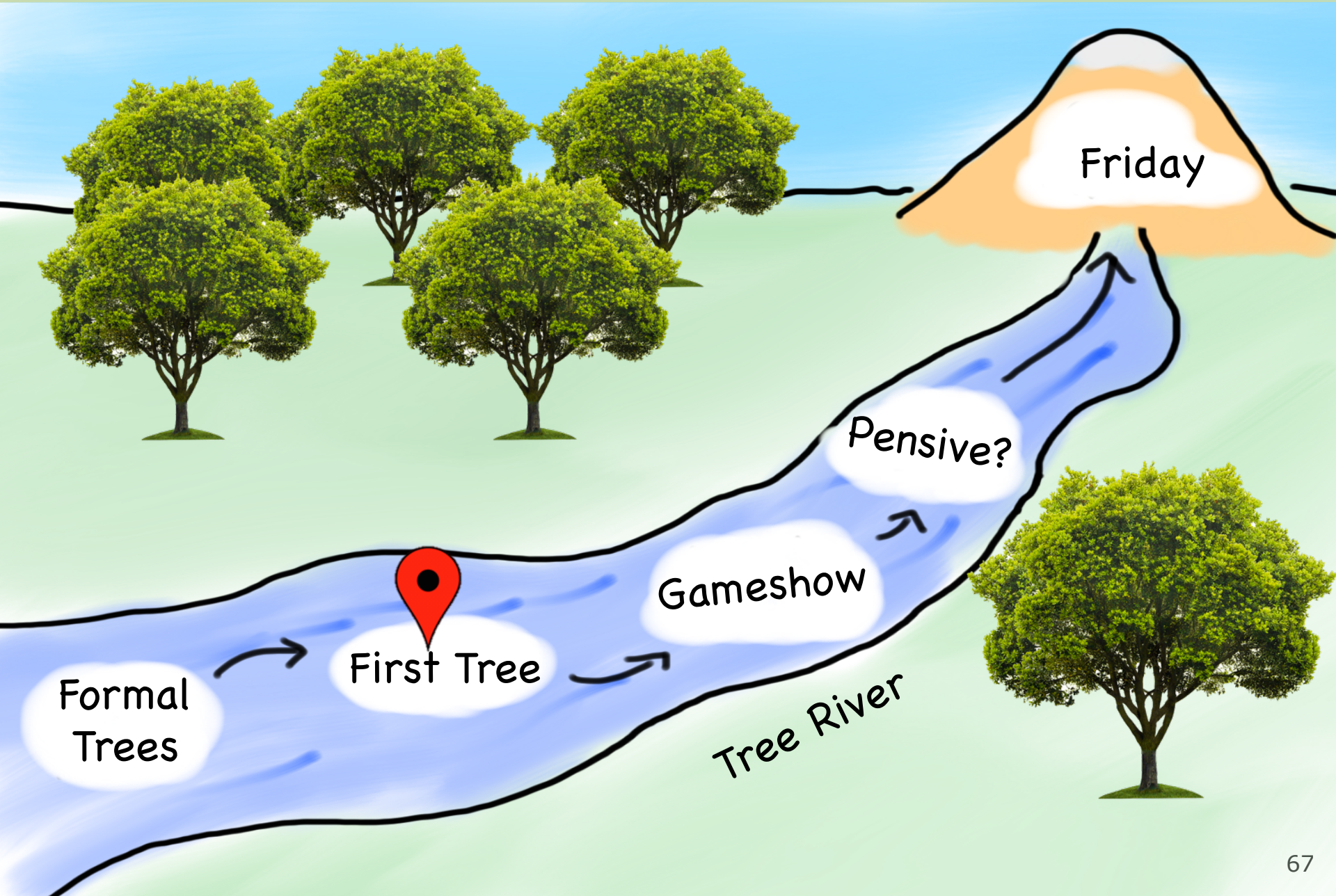




Today's Route

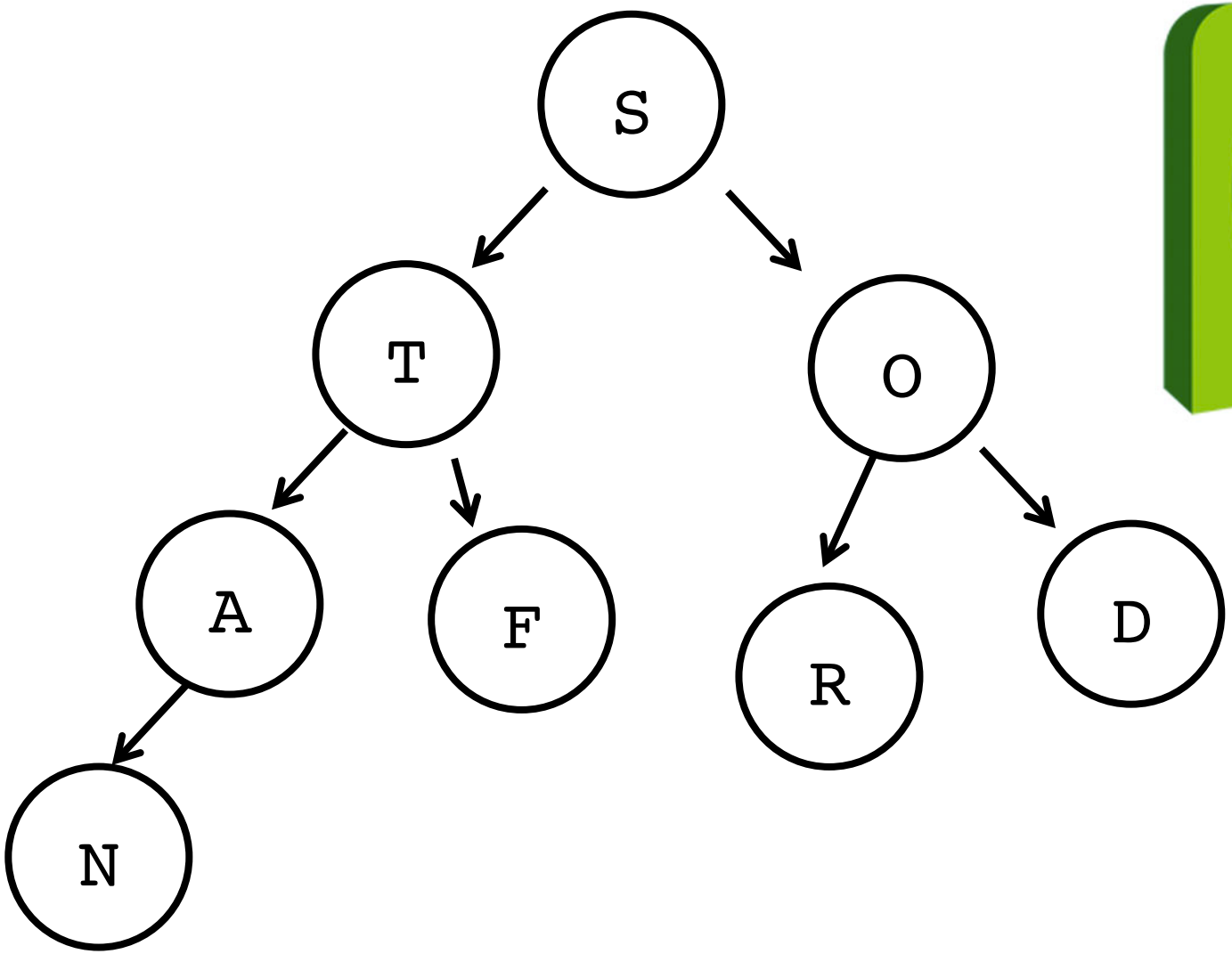


Today's Route

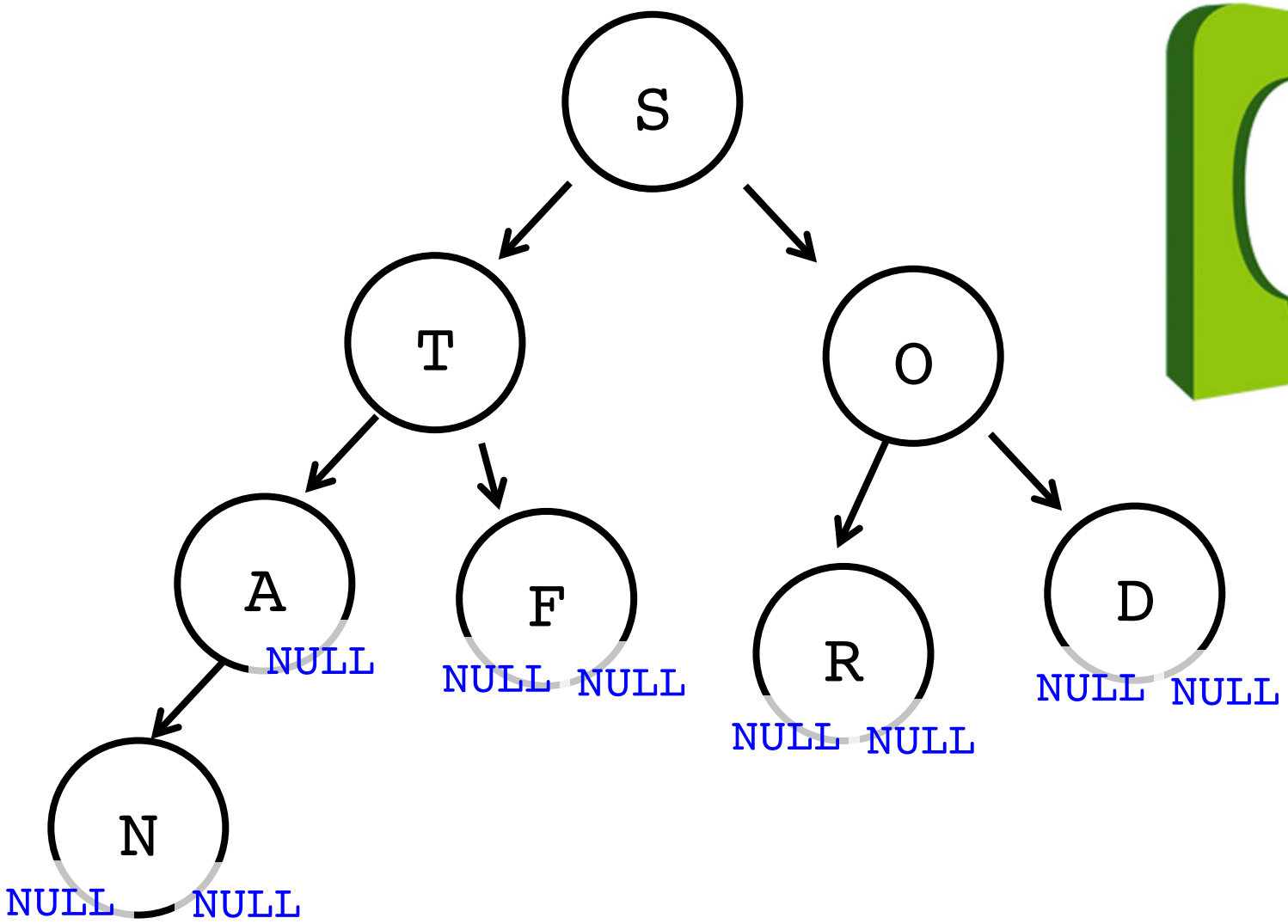



Qt

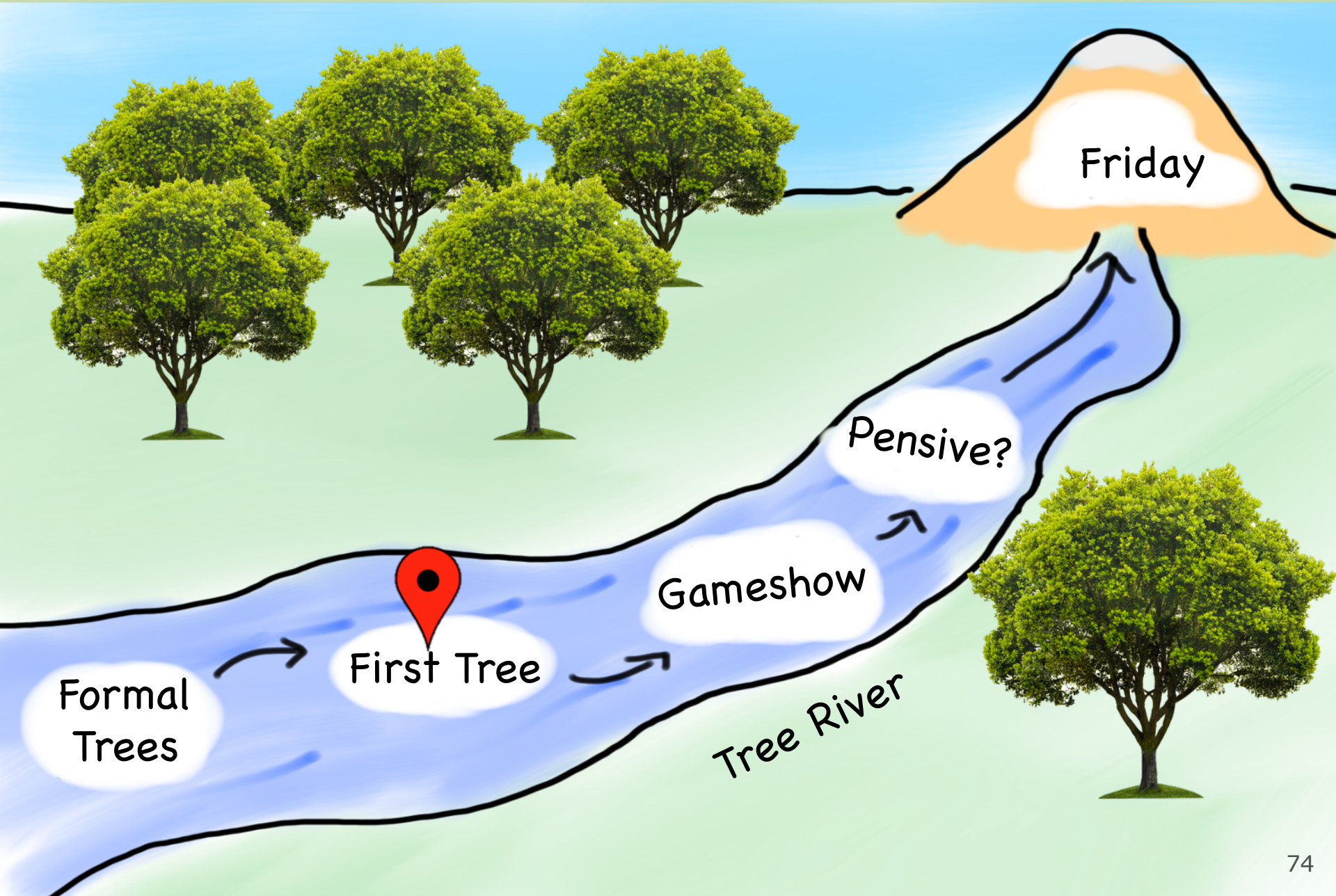
Cout This Tree



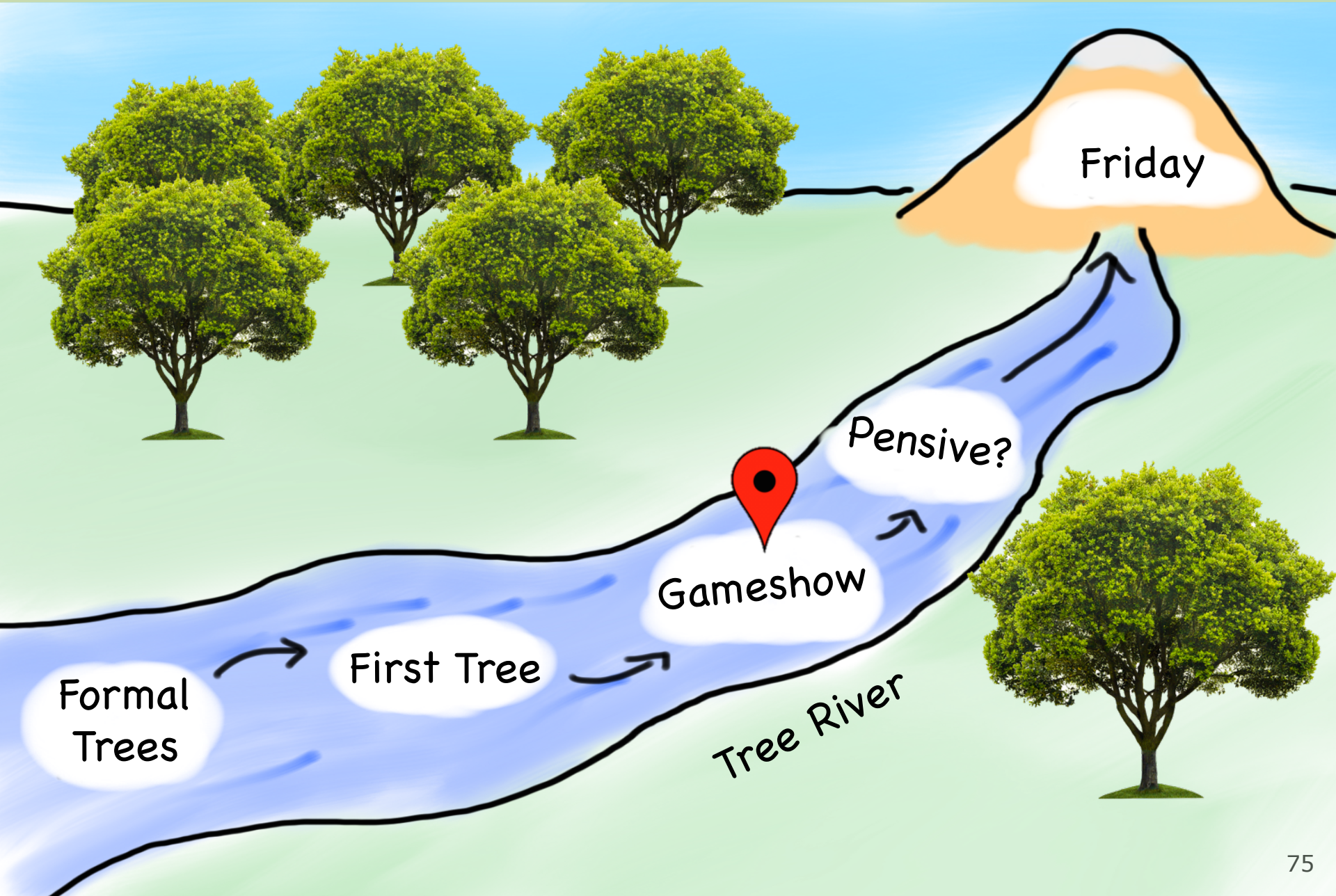
NULL Children



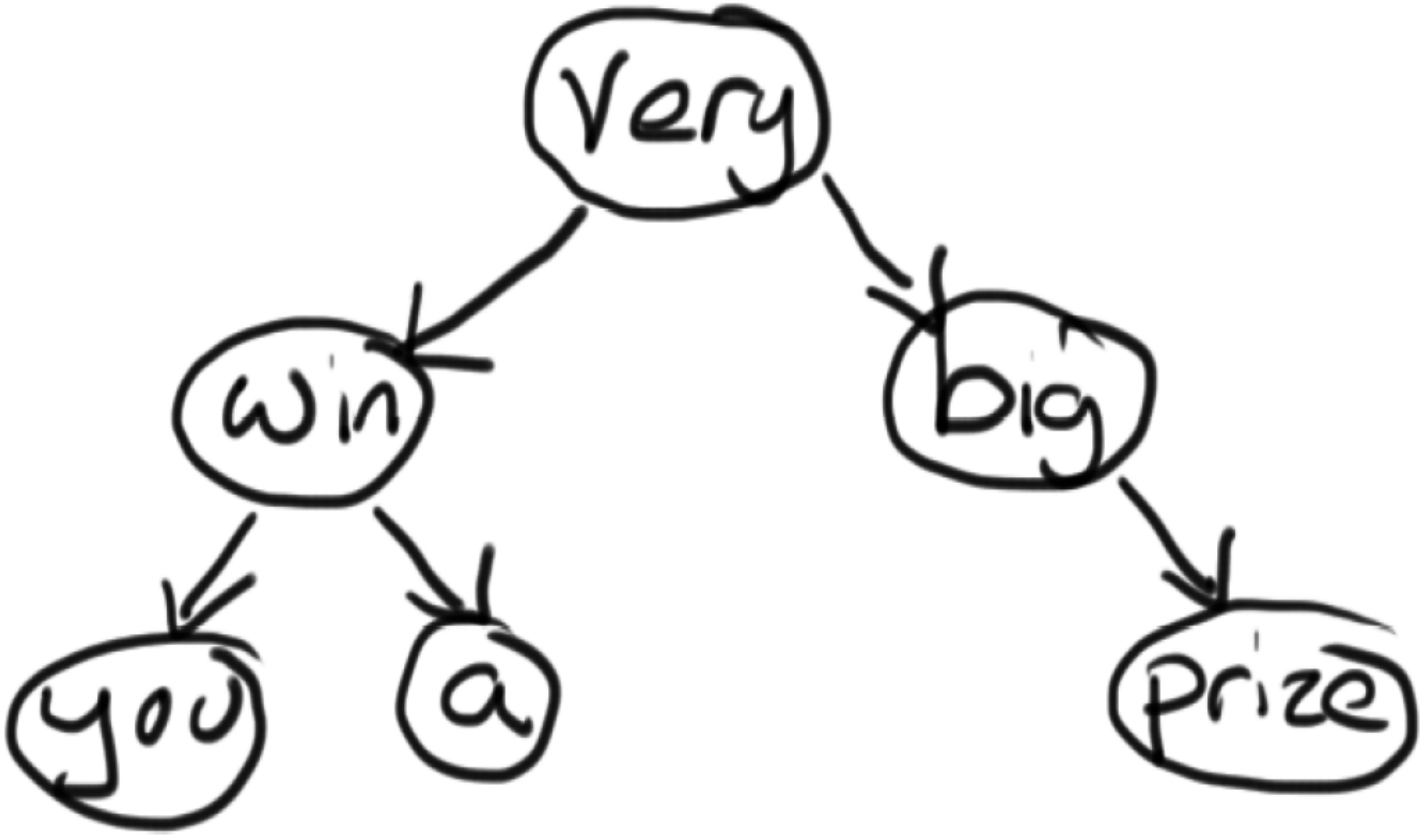
Today's Route



Today's Route



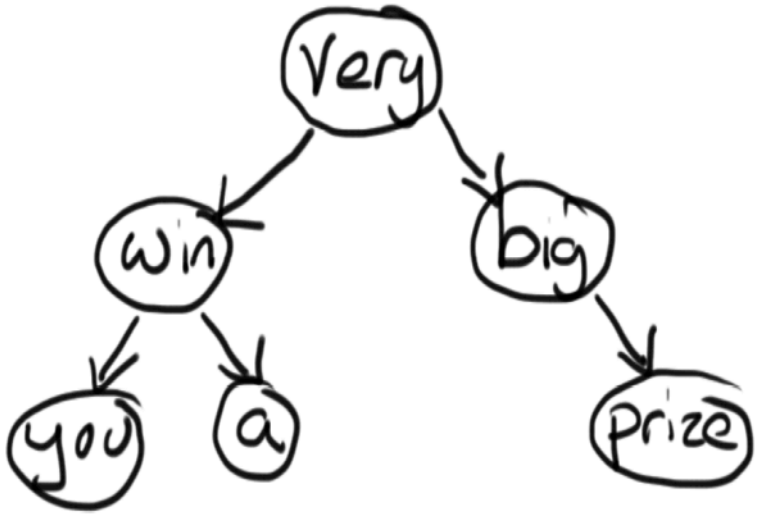
Game Show Tree



Game Show Tree

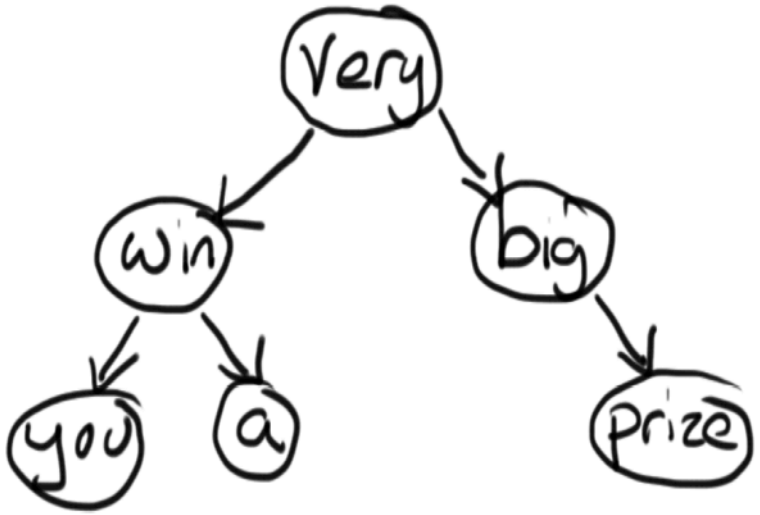
Game Show Tree

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



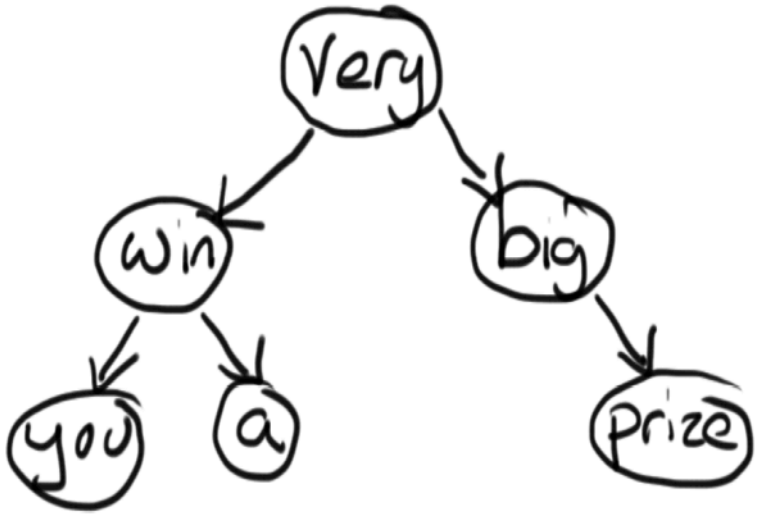
Game Show Tree

```
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;
}
```

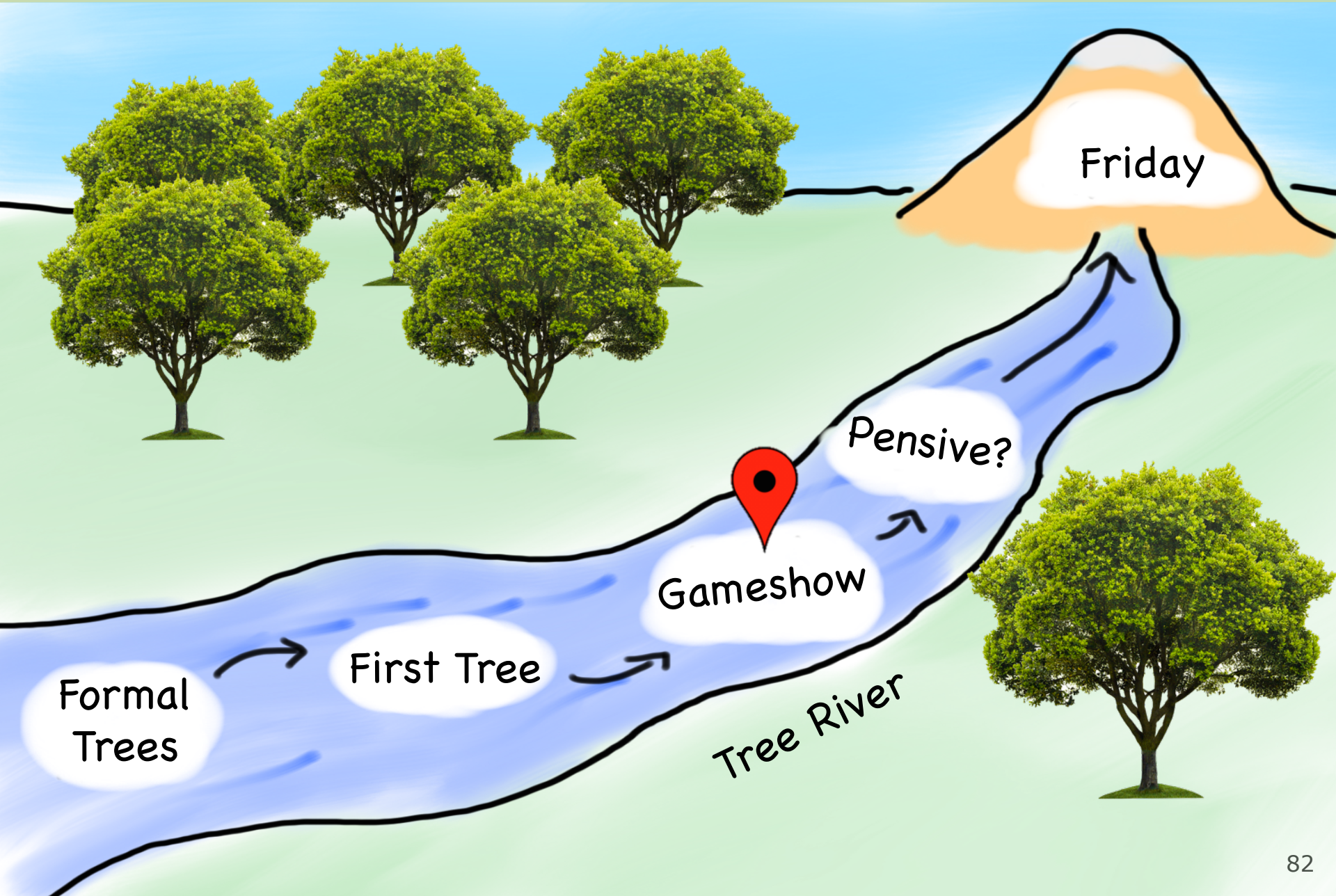


Game Show Tree

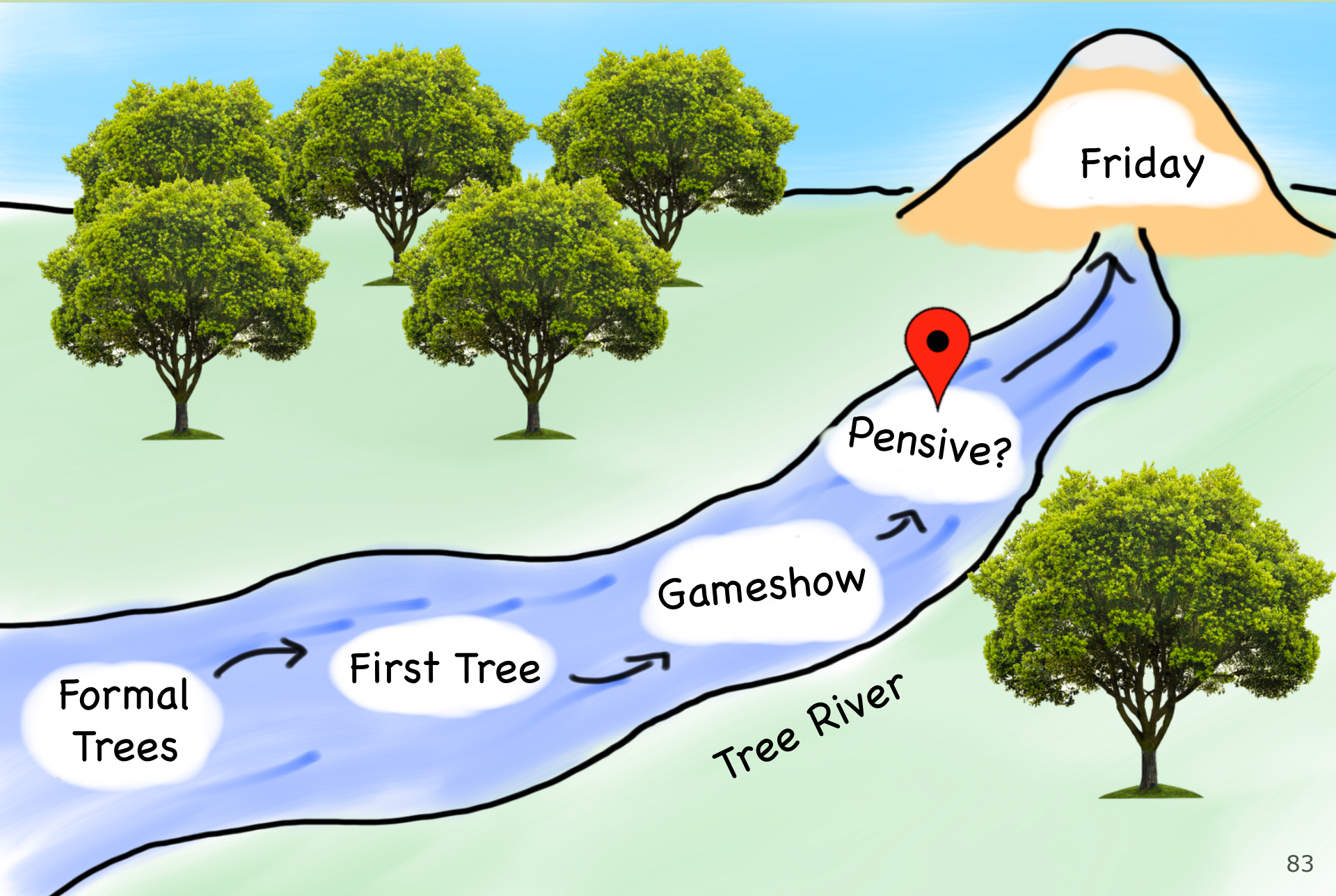
```
void doorOne(Tree * tree) {
    if(tree == NULL) return;
    cout<<tree->value<<" ";
    doorOne(tree->left);
    doorOne(tree->right);
}

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<<" ";
}
```

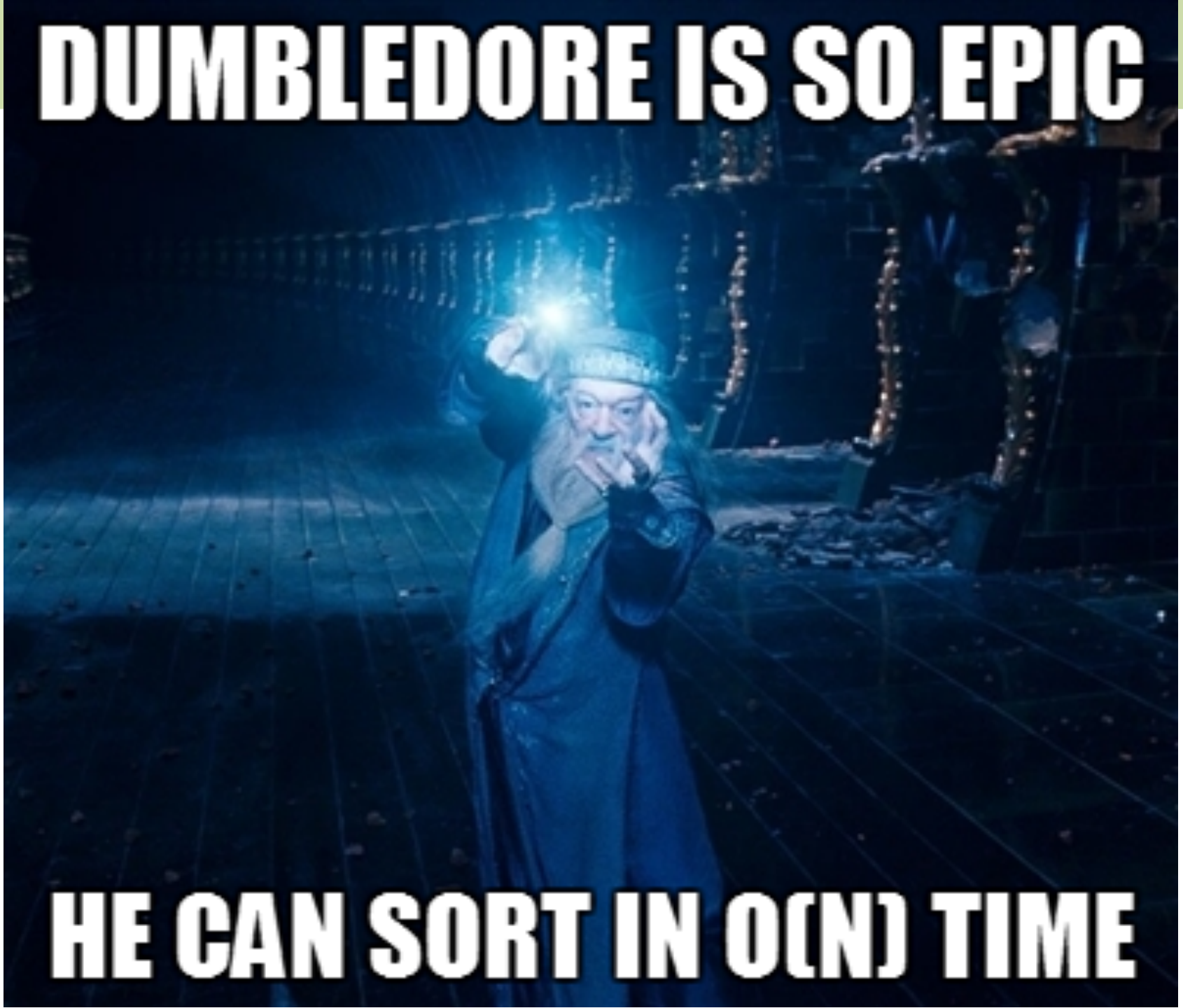


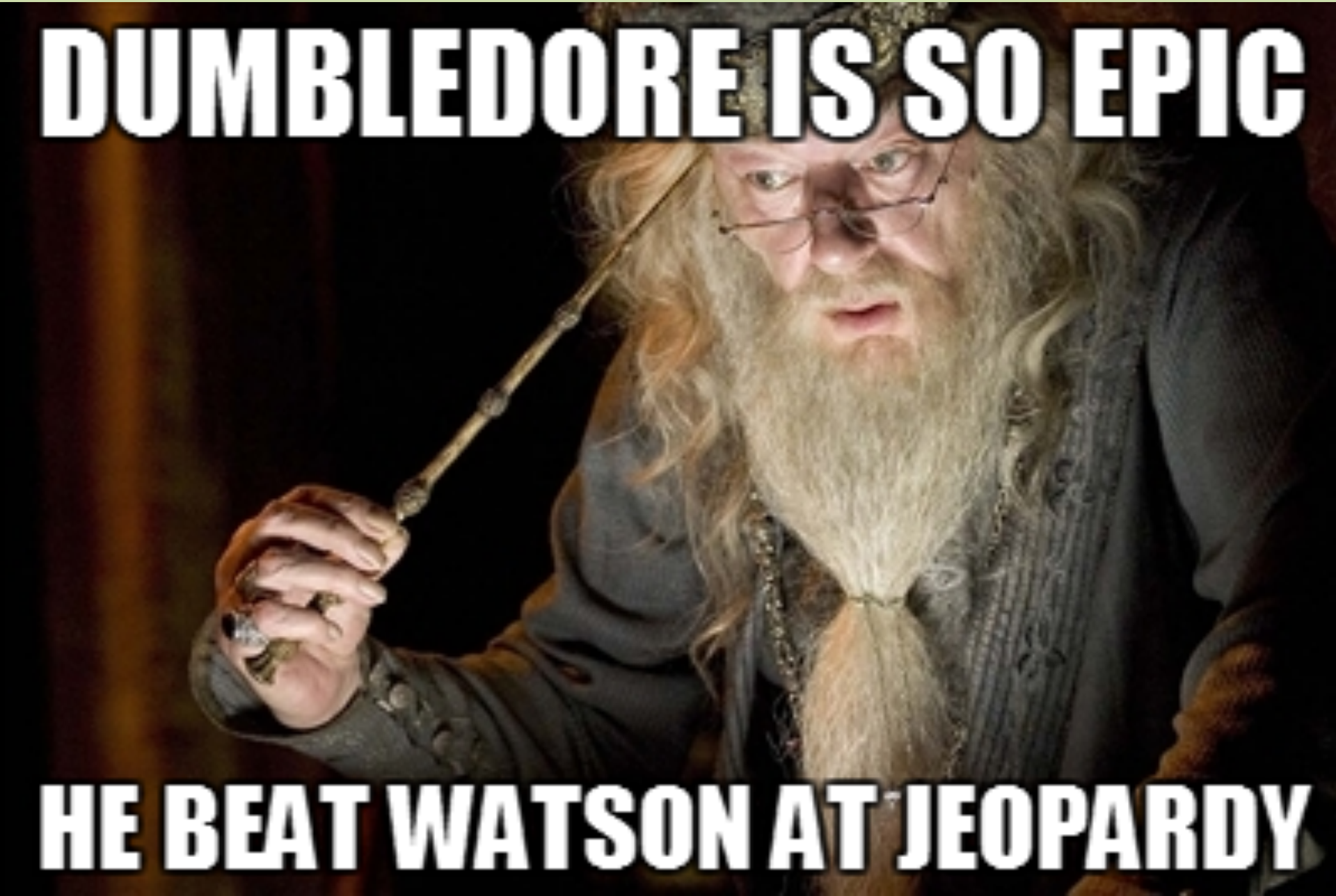
Today's Route



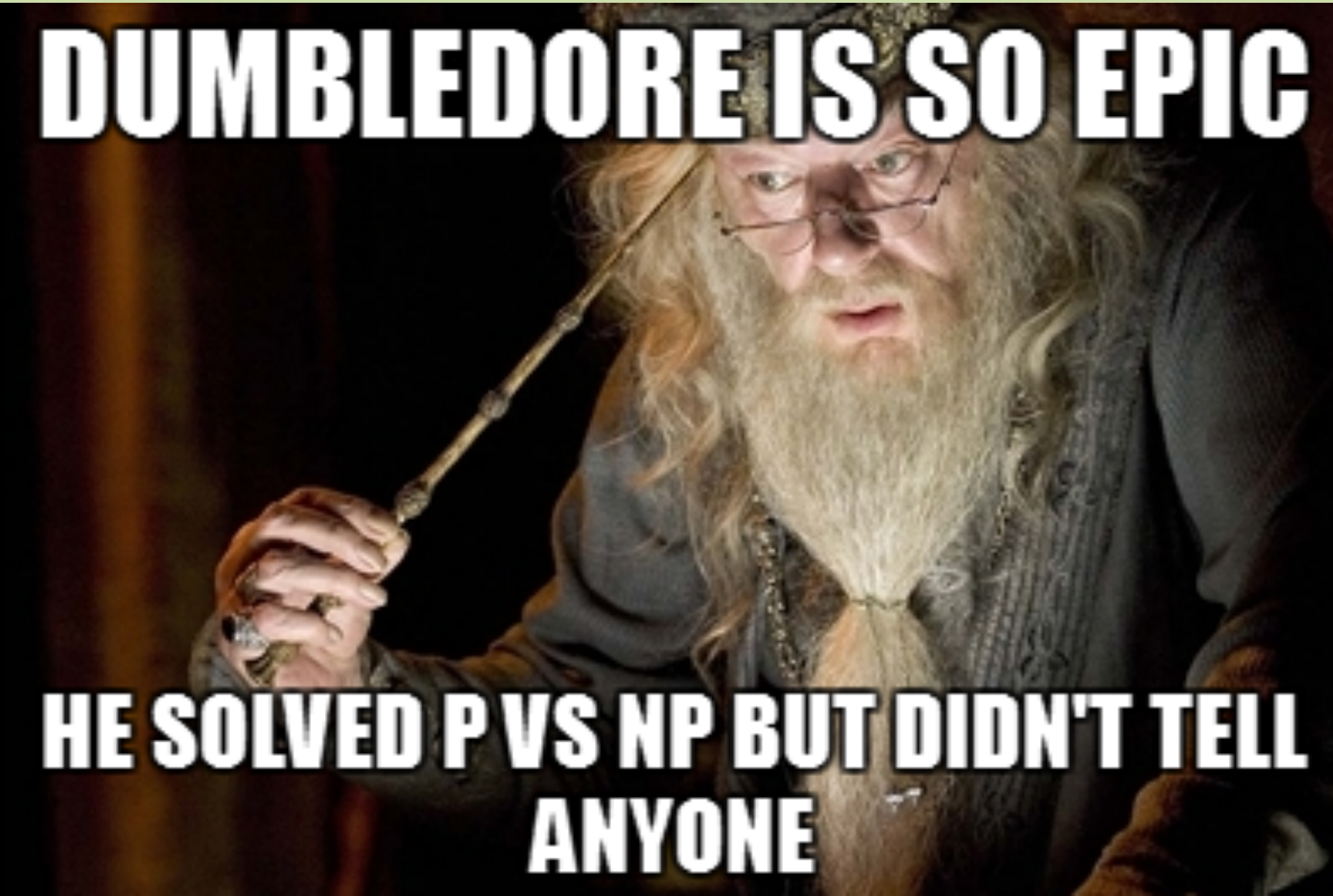
Today's Route

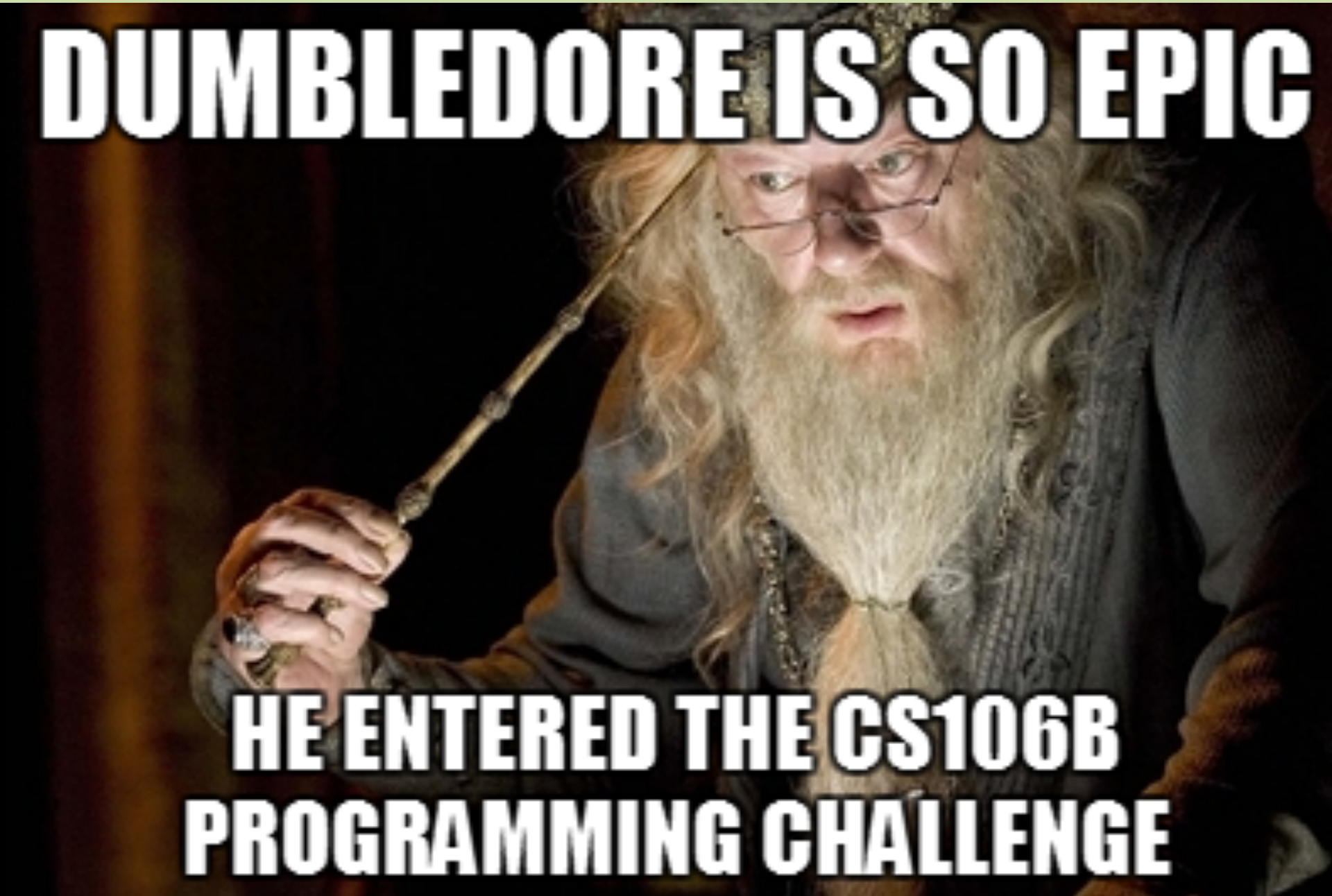



DUMBLEDORE IS SO EPIC
HE CAN SORT IN O(N) TIME


DUMBLEDORE IS SO EPIC
HE BEAT WATSON AT JEOPARDY


DUBLEDORE IS SO EPIC
HE PROGRAMS IN (C++)++


DUMBLEDORE IS SO EPIC
HE SOLVED P VS NP BUT DIDN'T TELL ANYONE

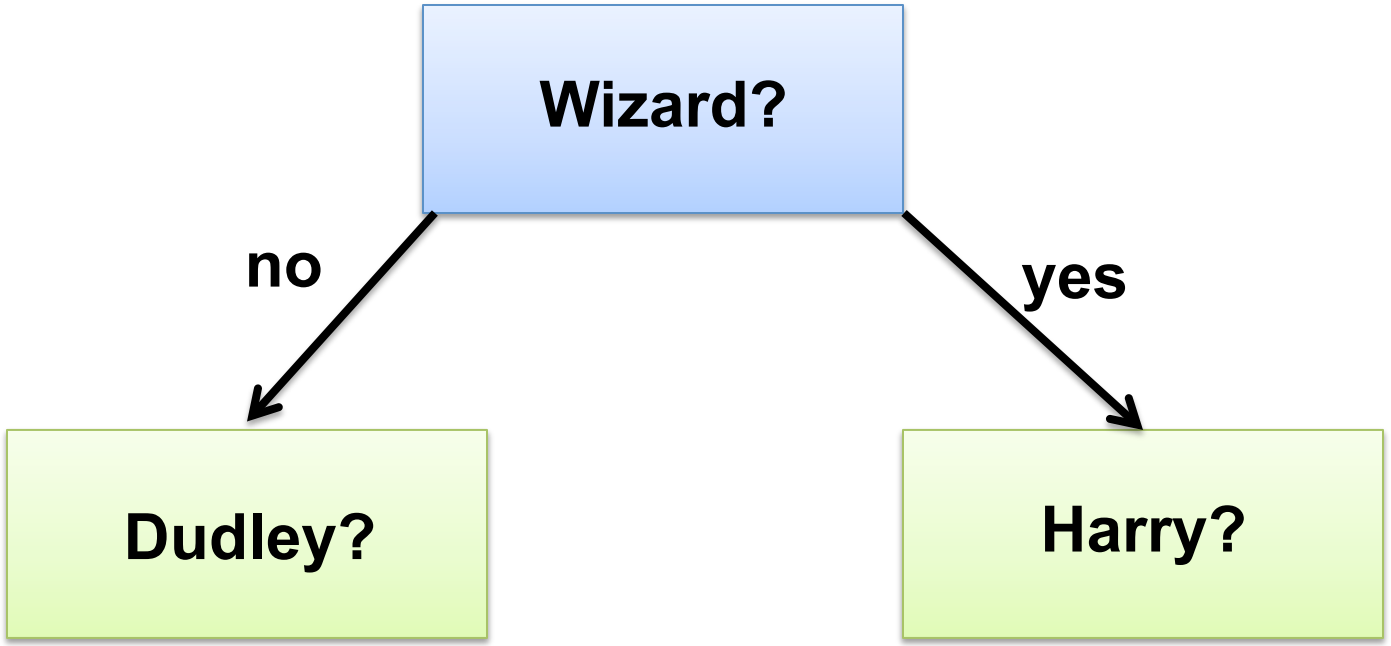

DUMBLEDORE IS SO EPIC
HE ENTERED THE CS106B PROGRAMMING CHALLENGE

How come Dumbledore knows everything?

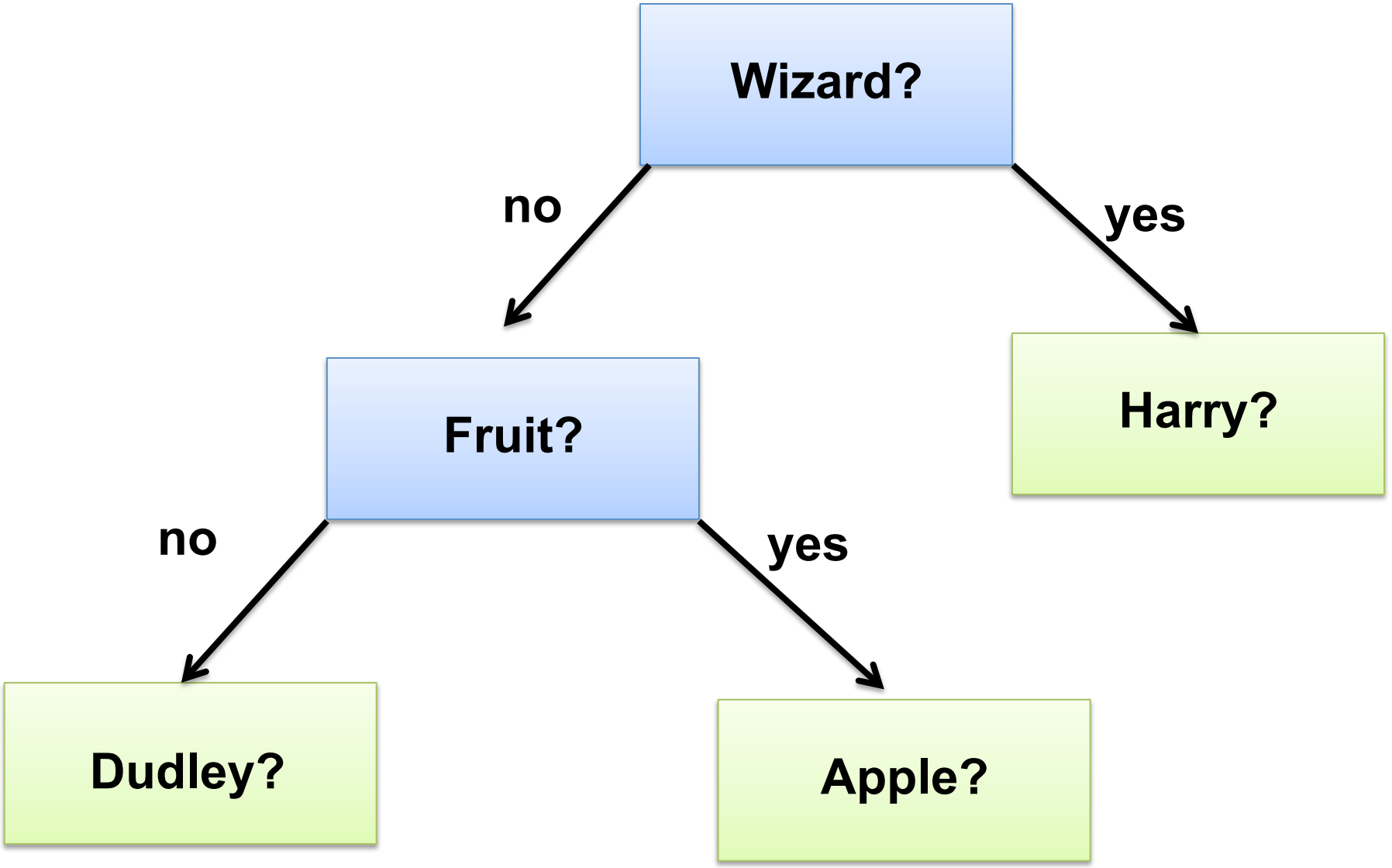
Pensive

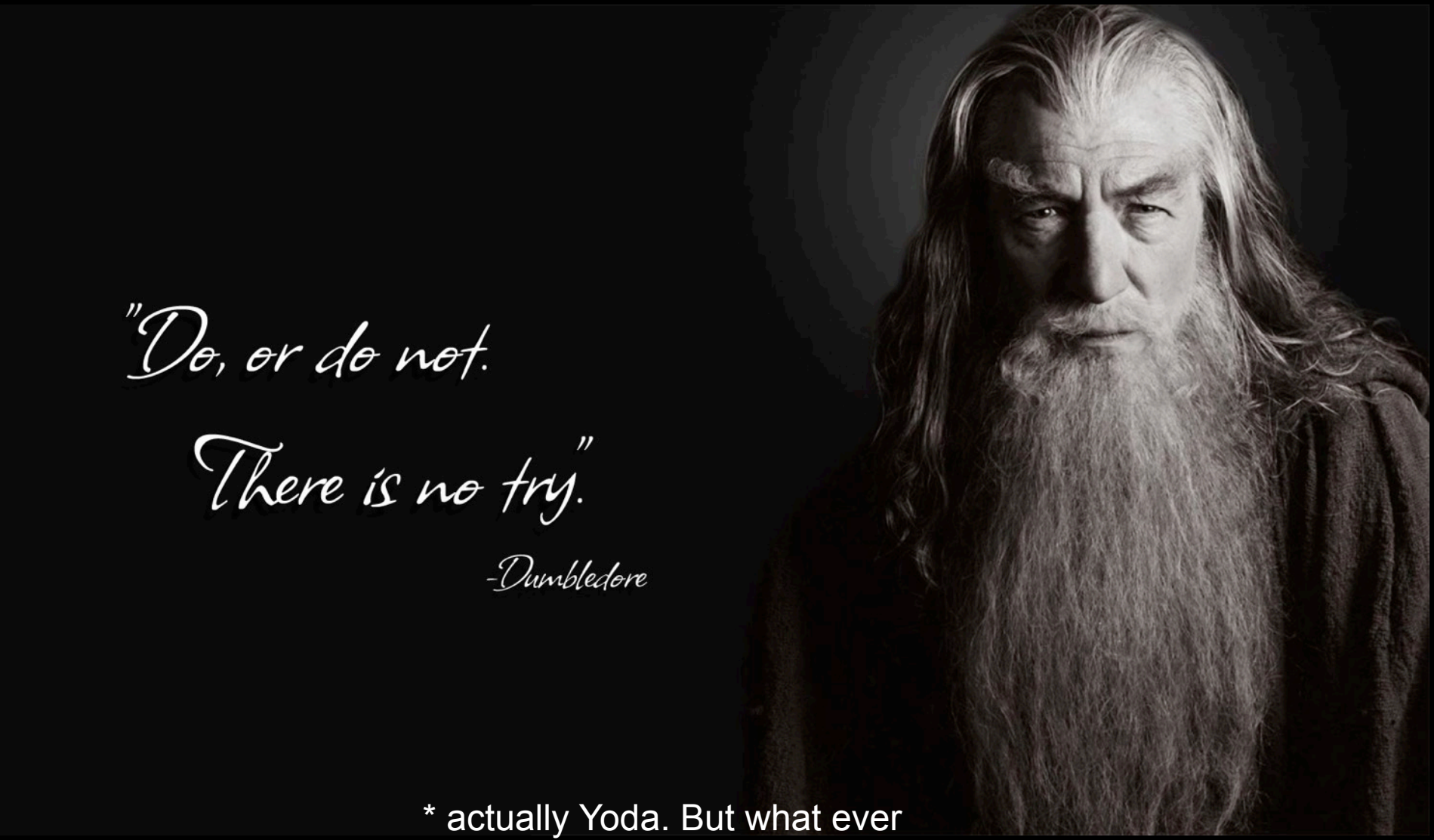
Pensive Demo

Pensive



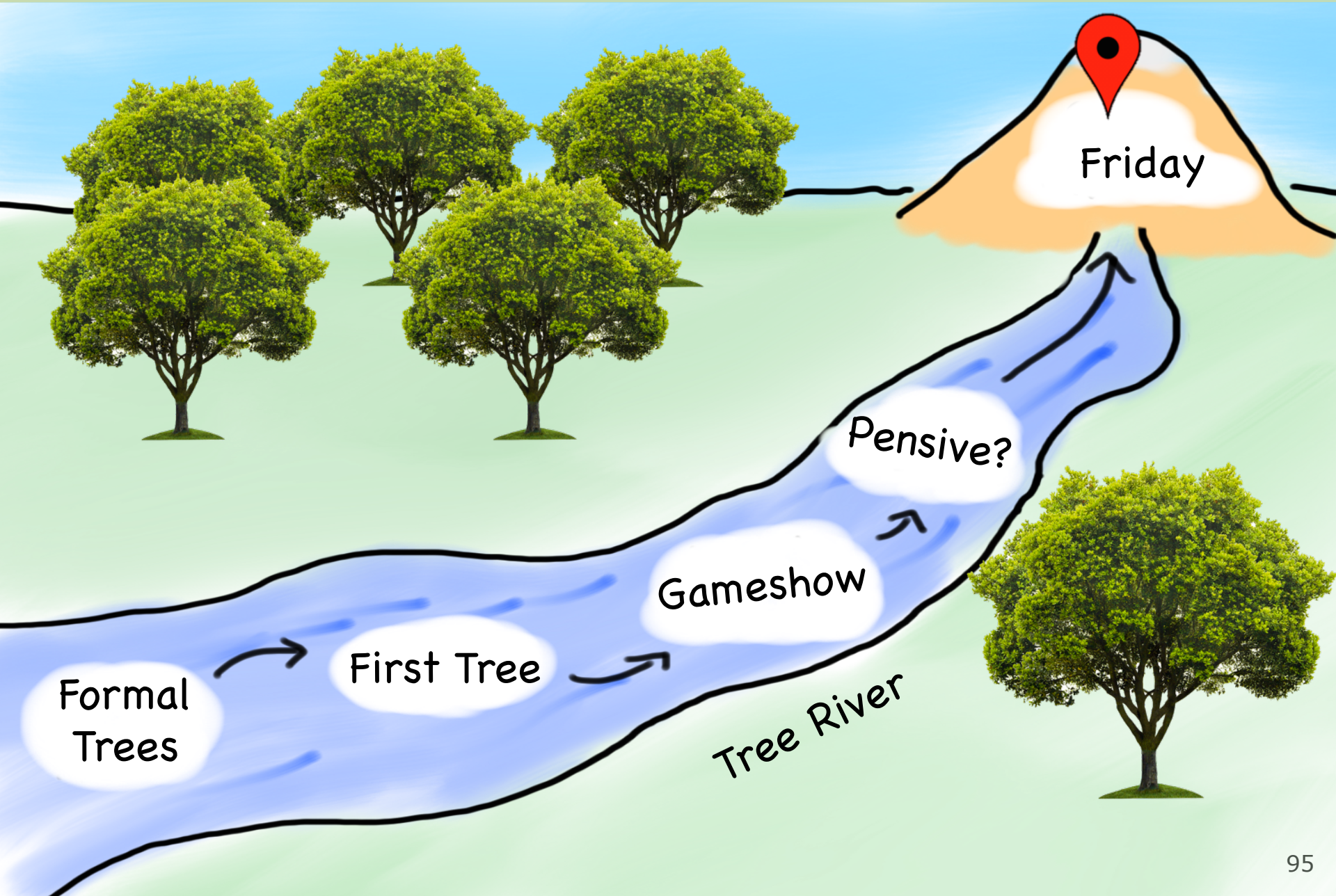
Pensive





* 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