Entertainment: CS 5 Green book to be made into feature-length movie starring George Clooney as the happy turtle.

New computer program predicts today's weather:
`IndexError: list index out of range`

Sports: CS 5 Green Prof runs to class in record time. Were steroids involved?

News in Brief

HMC CS Department to replace Apple computers with new Pumpkin brand computers. (p. 42)

Talking tree debuts in CS 5 Green!

CS 5 Green Today

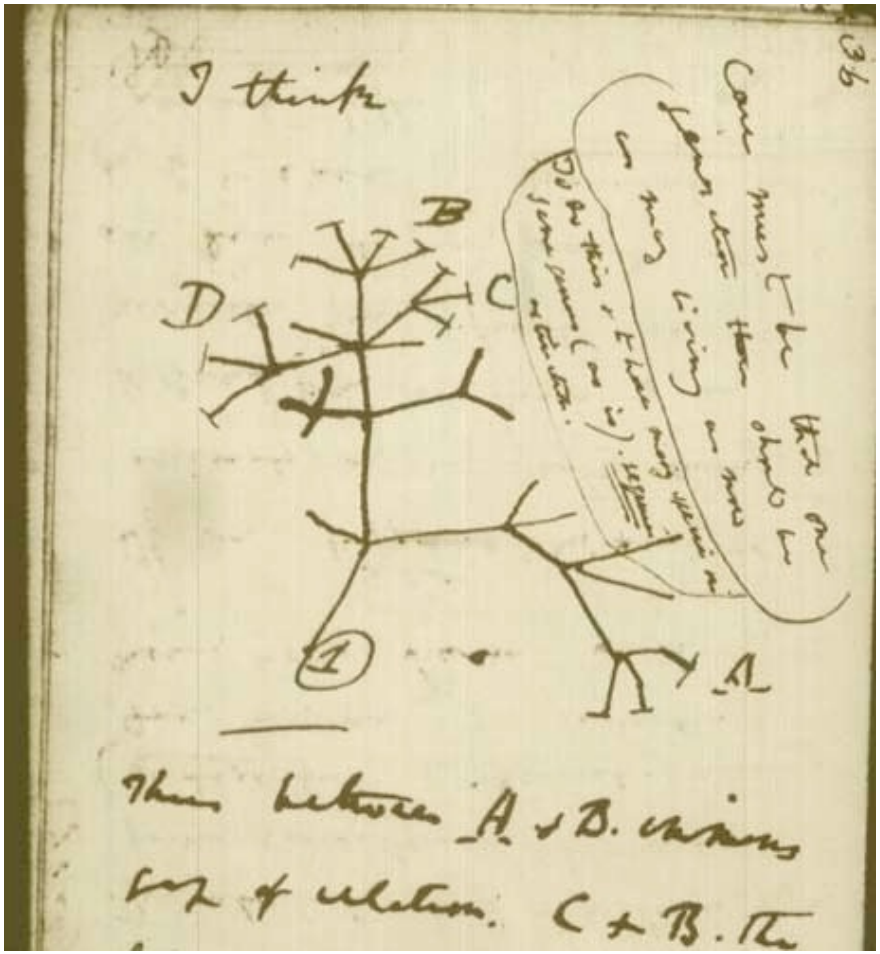
HMC CS 5 Green Professors discover new discovery

Claremont, CA: Researchers at Harvey Mudd College have made an extremely important new discovery said a spokesperson for the College. The discovery was evidently discovered while the researchers were trying to discover another discovery. "The professors discovered that their discovery had not been previously discovered, which is an important discovery in its own right," said the excited spokesperson. A number of prominent scientists also expressed their tremendous enthusiasm and said that they looked forward to reading what was actually discovered.

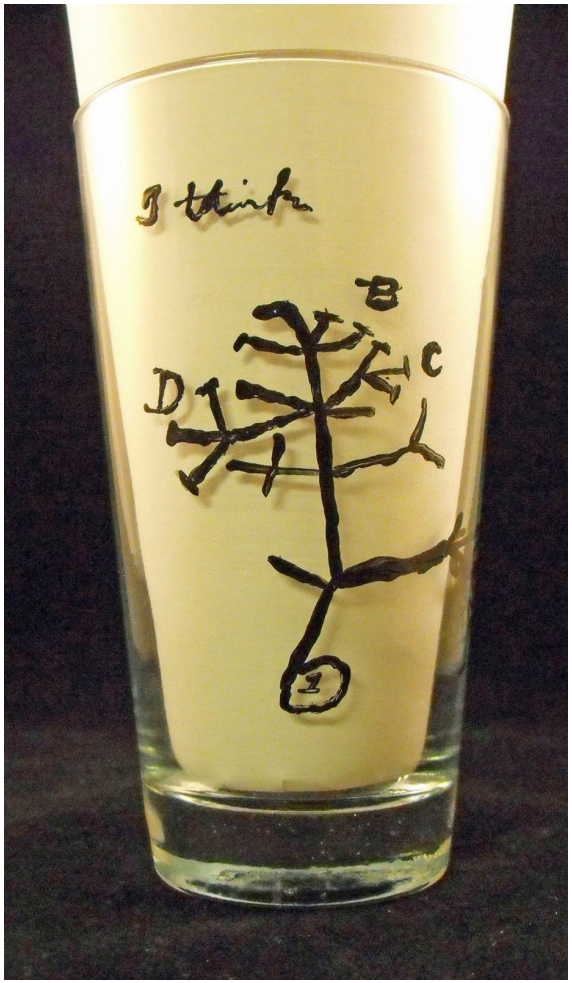
Learning Goals

- Review functions on trees
- Introduce a distance-based approach to phylogenetic tree reconstruction (UPGMA)

Phylogenetic Trees…

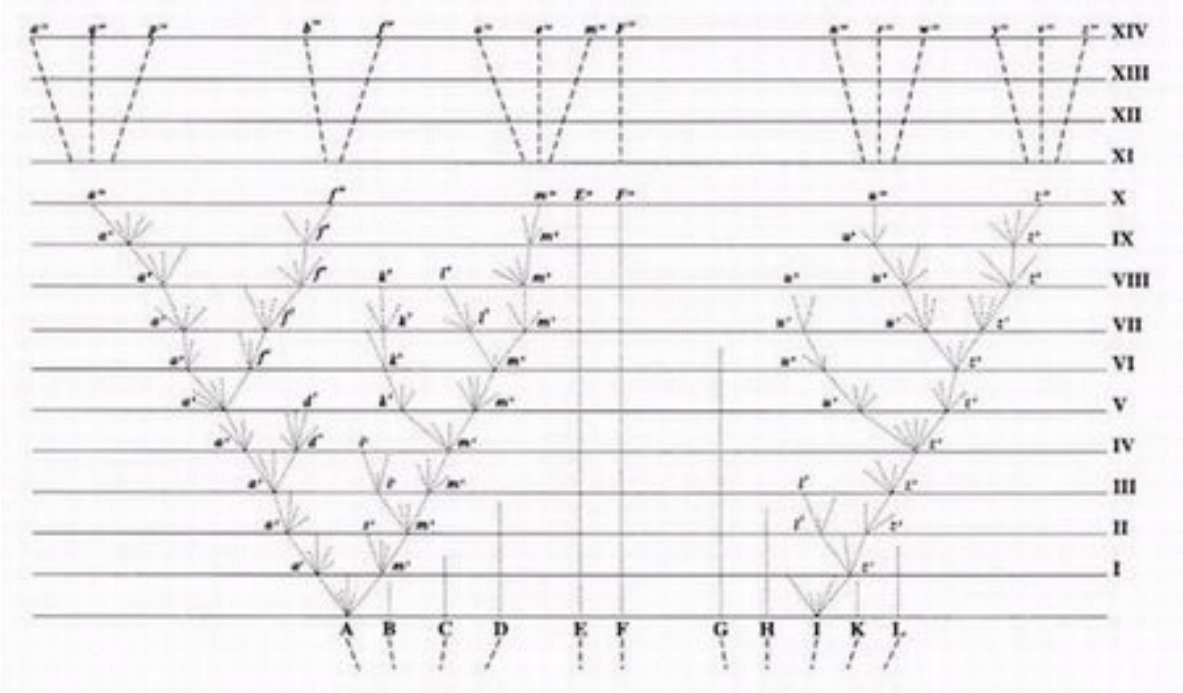


From Darwin's notebooks, 1837


I think
B
C
D
1

Really?

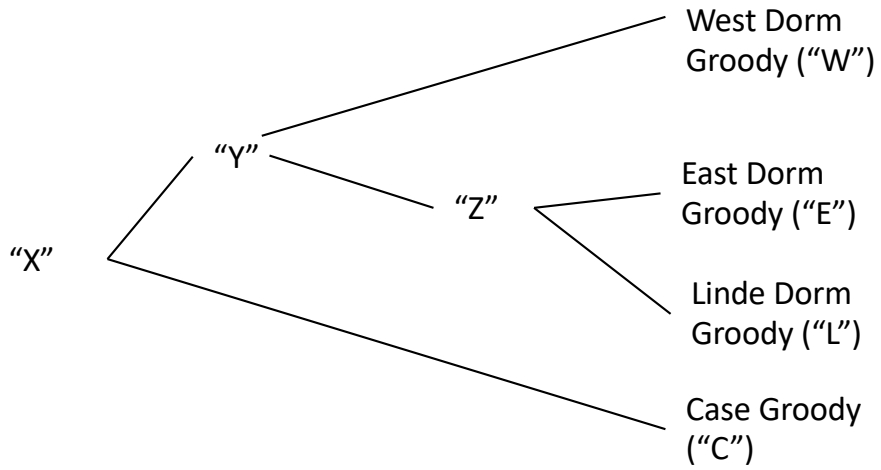
The only figure in *On the Origin of Species by Natural Selection* (Darwin, 1859)



From the 6th edition (1872): "The affinities of all the beings of the same class have sometimes been represented by a great tree. I believe this simile largely speaks the truth. The green and budding twigs may represent existing species; and those produced during former years may represent the long succession of extinct species."

More recursion on trees: `mrca`

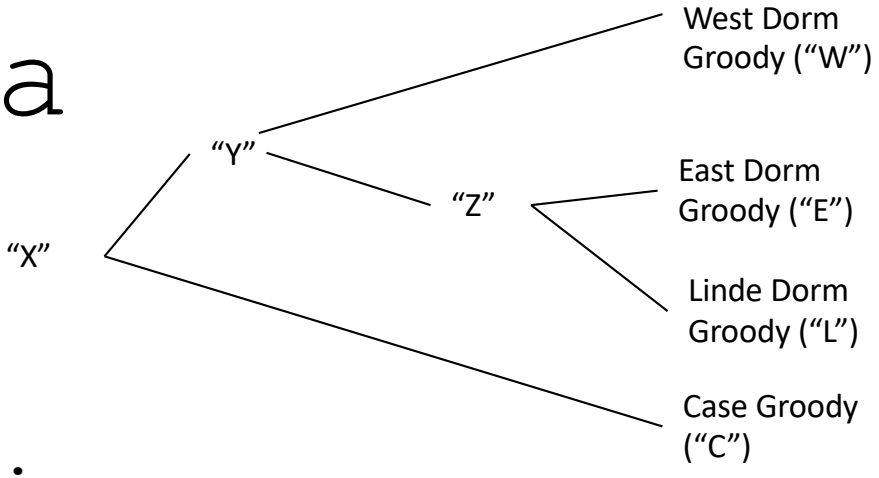
```
>>> find("L", groodies)
True
```




```
>>> mrca("L", "E", groodies) # use find as a helper!
'Z'
>>> mrca("W", "E", groodies)
'Y'
>>> mrca("W", "C", groodies)
'X'
>>> mrca("W", "Prof Bush", groodies)
None
```

You should use `find` here…

mrca



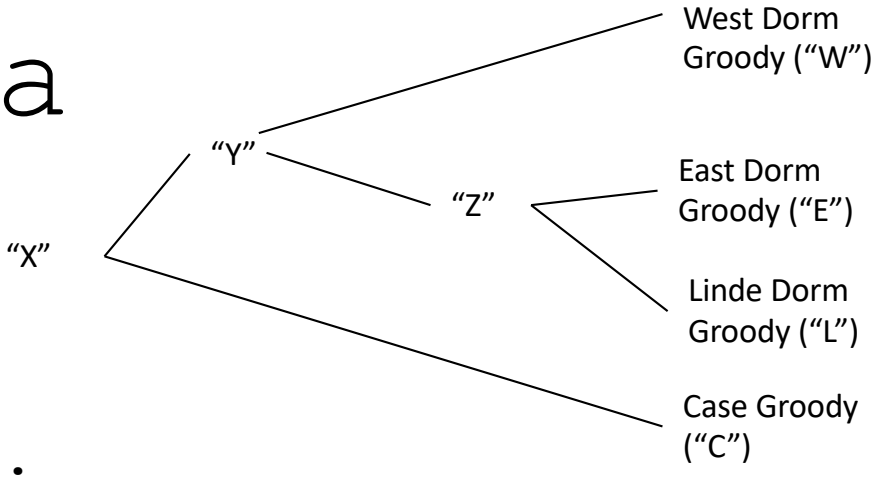
Notes

Q

```
def mrca(species1, species2, tree):
    """Return the name of the most recent commmon ancestor of
    species1 and species2. If there isn't one, return None."""
    root, left, right = tree
```

mrca

Notes

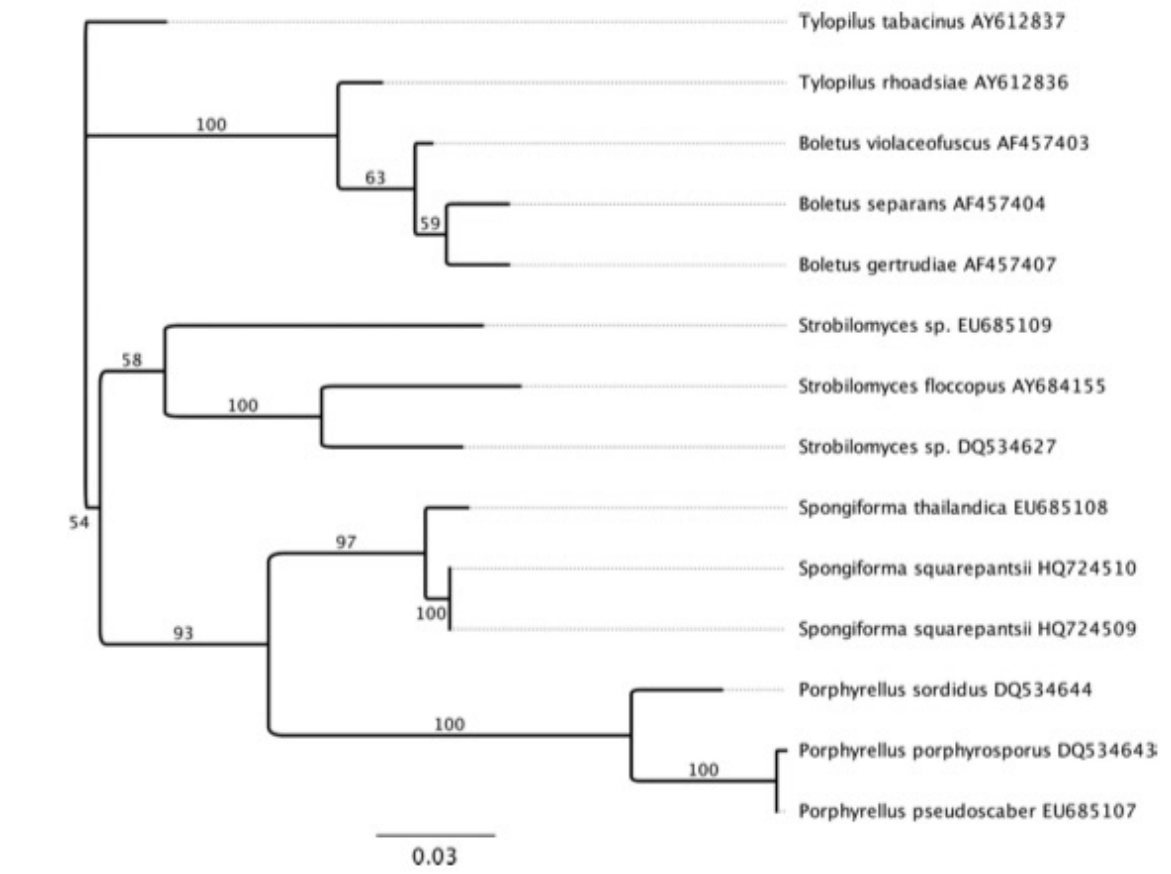



```
def mrca(species1, species2, tree):
    """Return the name of the most recent commmon ancestor of
    species1 and species2. If there isn't one, return None."""
    root, left, right = tree
    if left == ():
        return None
    elif (not find(species1, tree)) or (not find(species2, tree)):
        return None
    else:
        if find(species1, left) and find(species2, left):
            return mrca(species1, species2, left)
        elif find(species1, right) and find(species2, right):
            return mrca(species1,species2, right)
        else:
            return root
```

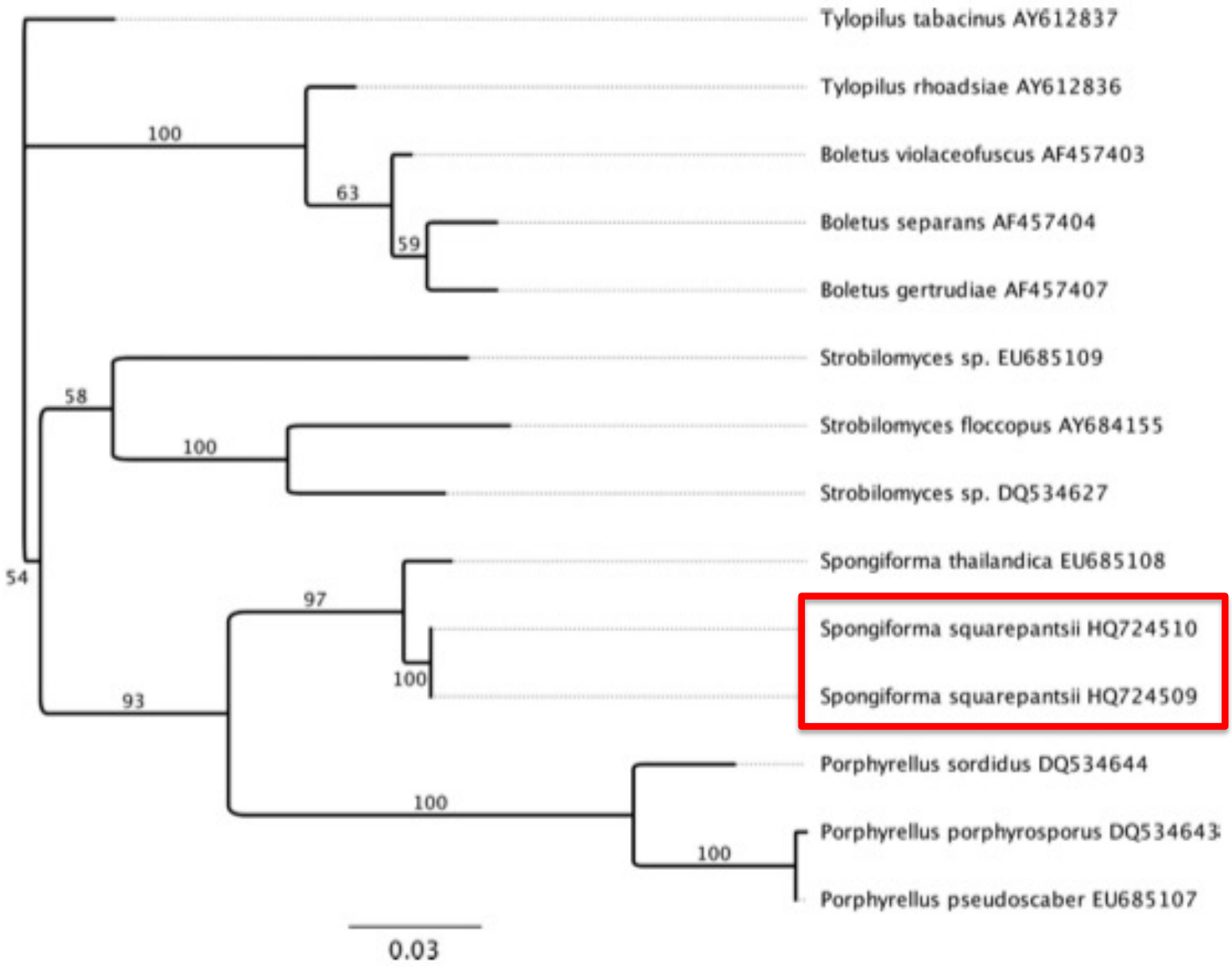
Phylogenetic Reconstruction

Input: DNA or protein sequences for each species

Output: Species tree



D. Desjardin, K. Peay, T. Bruns, *Mycologia* 103(5), 2011


Tylopilus tabacinus AY612837
Tylopilus rhoadsiae AY612836
Boletus violaceofuscus AF457403
Boletus separans AF457404
Boletus gertrudiae AF457407
Strobilomyces sp. EU685109
Strobilomyces floccopus AY684155
Strobilomyces sp. DQ534627
Spongiforma thailandica EU685108
Spongiforma squarepantsii HQ724510
Spongiforma squarepantsii HQ724509
Porphyrellus sordidus DQ534644
Porphyrellus porphyrosporus DQ534643
Porphyrellus pseudoscaber EU685107
100
63
59
58
100
54
97
100
93
100
100
0.03

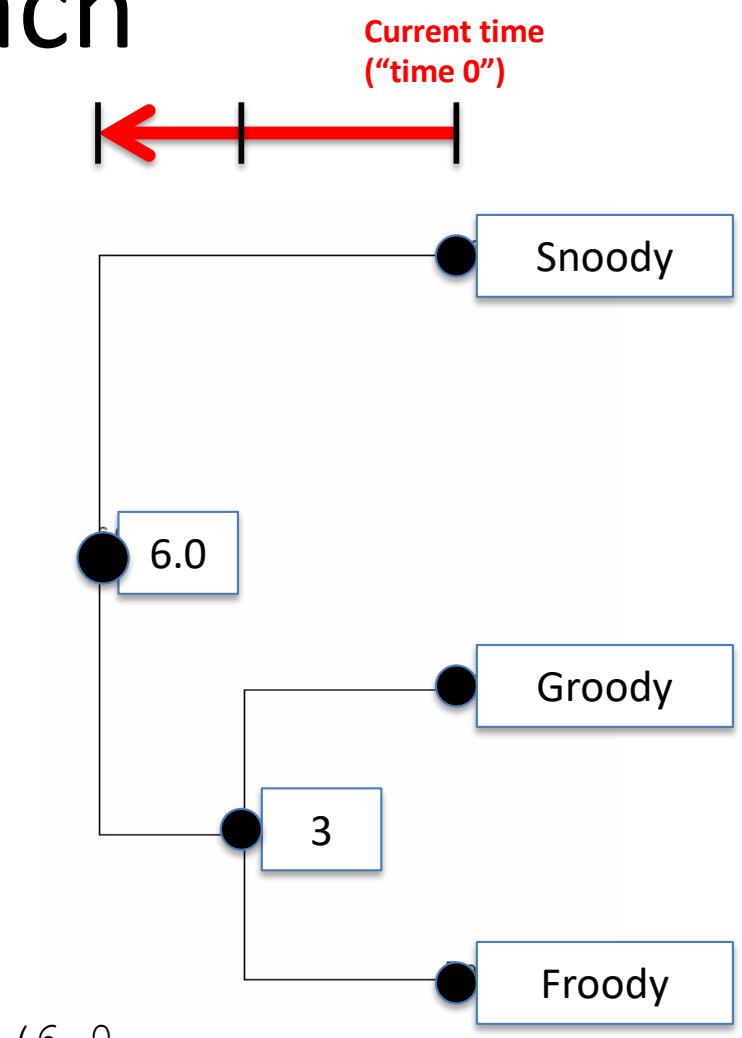
Distance-Based Approach

```
Groody CATCAACCAGTGACCAGTATAGGACGCCC
Froody CAACACTCAGTGACAAGTCTAGCACGCCC
Snoody AATCGCCCGGCGTCAGGCATAGCTAGCGC
```

	G	F	S
G	0	6	12
F	6	0	12
S	12	12	0

Distance-Based Approach

	G	F	S
G	0	6	12
F	6	0	12
S	12	12	0




```
(6.0,
    ('Snoody', (), ()),
    (3.0,
        ('Groody', (), ()),
        ('Froody', (), ())
    )
)
```

Unweighted Pair Group Method with Arithmetic Mean (UPGMA) Algorithm

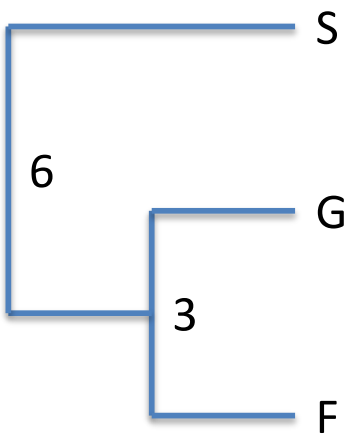
Groody
Froody
Snoody

	G	F	S
G	0	6	12
F	6	0	12
S	12	12	0

Unweighted Pair Group Method with Arithmetic Mean (UPGMA) Algorithm

Groody
Froody
Snoody

	G	F	S
G	0	6	12
F	6	0	12
S	12	12	0



	(G,F)	S
(G,F)	0	12
S	12	0

$D_1((G,F),S) = (D_0(G,S) + D_0(F,S)) / 2$
$= (12 + 12) / 2 = 12$

Worksheet

Try This One…

This algorithm's "bark" is worse than it's bite.

```
Aoody
Boody
Coody
Doody
Eoody
```

	A	B	C	D	E
A	0				
B	4	0			
C	16	16	0		
D	16	16	8	0	
E	16	16	8	6	0

The matrix is symmetric, so we just need to keep the bottom or top half!

Worksheet

Try This One…

D_0

	A	B	C	D	E
A	0				
B	4	0			
C	16	16	0		
D	16	16	8	0	
E	16	16	8	6	0

D_1

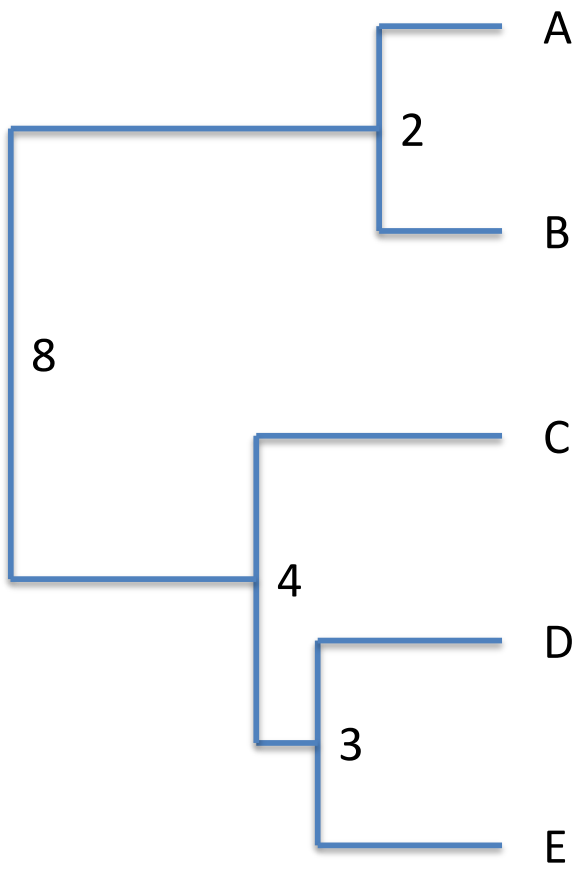
	(A,B)	C	D	E
(A,B)	0			
C	16	0		
D	16	8	0	
E	16	8	6	0

D_2

	(A,B)	C	(D,E)
(A,B)	0		
C	16	0	
(D,E)	16	8	0

D_3

	(A,B)	(C,(D,E))
(A,B)	0	
(C,(D,E))	16	0



DEMO!

I wood love to see this in action!

Implementing UPGMA

```
1 Groody
2 Froody
3 Snoody
```

	1	2	3
1	0	6	12
2	6	0	12
3	12	12	0

Let's see this in the provided `mitoData.py` file

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())
groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1):0,   (N1, N2):6,   (N1, N3):12,
                (N2, N1):6,   (N2, N2):0,   (N2, N3):12,
                (N3, N1):12,  (N3, N2):12,  (N3, N3):0}
```

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())
groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1):0,  (N1, N2):6,  (N1, N3):12,
                (N2, N1):6,  (N2, N2):0,  (N2, N3):12,
                (N3, N1):12, (N3, N2):12, (N3, N3):0}
```

1. For all *pairs* of trees in the `groodies_lst`, find the closest pair

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())
groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1):0,  (N1, N2):6,  (N1, N3):12,
                (N2, N1):6,  (N2, N2):0,  (N2, N3):12,
                (N3, N1):12, (N3, N2):12, (N3, N3):0}
```

1. For all *pairs* of trees in the `groodies_lst`, find the closest pair

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
```

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())
groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1):0,   (N1, N2):6,   (N1, N3):12,
                (N2, N1):6,   (N2, N2):0,   (N2, N3):12,
                (N3, N1):12,  (N3, N2):12,  (N3, N3):0}
```

1. For all *pairs* of trees in the `groodies_lst`, find the closest pair

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
```

2. Remove those from the `groodies_lst`

```
groody_lst.remove(N1)
groody_lst.remove(N2)
```

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())
groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1):0,   (N1, N2):6,   (N1, N3):12,
                (N2, N1):6,   (N2, N2):0,   (N2, N3):12,
                (N3, N1):12,  (N3, N2):12,  (N3, N3):0}
```

1. For all *pairs* of trees in the `groodies_lst`, find the closest pair

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
```

2. Remove those from the `groodies_lst`

3. Make a new tree by joining these two trees…

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())
groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1):0,   (N1, N2):6,   (N1, N3):12,
                (N2, N1):6,   (N2, N2):0,   (N2, N3):12,
                (N3, N1):12,  (N3, N2):12,  (N3, N3):0}
```

1. For all *pairs* of trees in the `groodies_lst`, find the closest pair

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
```

2. Remove those from the `groodies_lst`

3. Make a new tree by joining these two trees…

```
new_tree = (3.0, N1, N2)
         = (3.0, ("Groody", (), ()), ("Froody", (), ()))
```

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())                    N1                              N2
groodies_lst = [N3, (3.0, ("Groody", (), ()), ("Froody", (), ()))]
groodies_mat = {(N1, N1):0,   (N1, N2):6,   (N1, N3):12,
                (N2, N1):6,   (N2, N2):0,   (N2, N3):12,
                (N3, N1):12,  (N3, N2):12,  (N3, N3):0}
```

1. For all *pairs* of trees in the `groodies_lst`, find the closest pair

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
```

2. Remove those from the `groodies_lst`

3. Make a new tree by joining these two trees…

```
new_tree = (3.0, N1, N2)
         = (3.0, ("Groody", (), ()), ("Froody", (), ()))
```

4. Add this new tree to the `groodies_lst`

```
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())                    N1                          N2
groodies_lst = [N3, (3.0, ("Groody", (), ()), ("Froody", (), ()))]
groodies_mat = {(N1, N1):0,   (N1, N2):6,   (N1, N3):12,
                (N2, N1):6,   (N2, N2):0,   (N2, N3):12,
                (N3, N1):12, (N3, N2):12, (N3, N3):0}
```

5. Update the distance matrix…

```
groodies_mat[(N3, new_tree)] = …
groodies_mat[(new_tree, N3)] = …
```

We are still holding on to `N1` and `N2`,
even though we have removed them from the `groodies_lst`!

Implementing UPGMA

```
findClosestPair( speciesList, Distances ):
   """Takes a list of species trees and the distance dictionary
      as input and returns a tuple (X, Y) where X and Y are in the
      list and have the minimum distance between any two items in the list."""

updateDist( speciesList, Distances, newTree):
   """Takes a list of species trees, the distance dictionary, and a newTree
      that was just formed by merging two trees found by findClosestPair.
      Those two trees can be found by looking inside newTree.  Those two trees
      are removed from the distance dictionary and the newTree is added to the
      dictionary."""

upgma( speciesList, Distances):
    """Returns the phylogenetic tree constructed by the UPGMA algorithm."""
```

findClosestPair: 7 lines
updateDist: 8 lines
upgma: 12 lines

Updating the distance dictionary: a subtlety

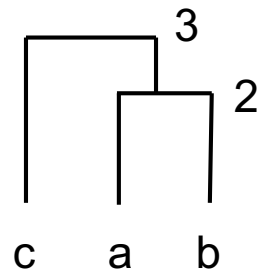
	a	**b**	**c**	**d**
a	0			
b	4	0		
c	6	6	0	
d	12	12	9	0

→

	a,b	**c**	**d**
a,b	0		
c	6	0	
d	12	9	0

→

	a,b,c	**d**
a,b,c	0	
d	???	0



Updating the distance dictionary: a subtlety

	a	b	c	d
a	0			
b	4	0		
c	6	6	0	
d	12	12	9	0

→

	a,b	c	d
a,b	0		
c	6	0	
d	12	9	0

→

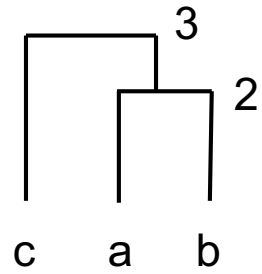
	a,b,c	d
a,b,c	0	
d	11	0

When we calculate distances between nodes with different numbers of leaves, we should weight by the number of leaves.

2 leaves 1 leaf = 3 total

a,b **c**

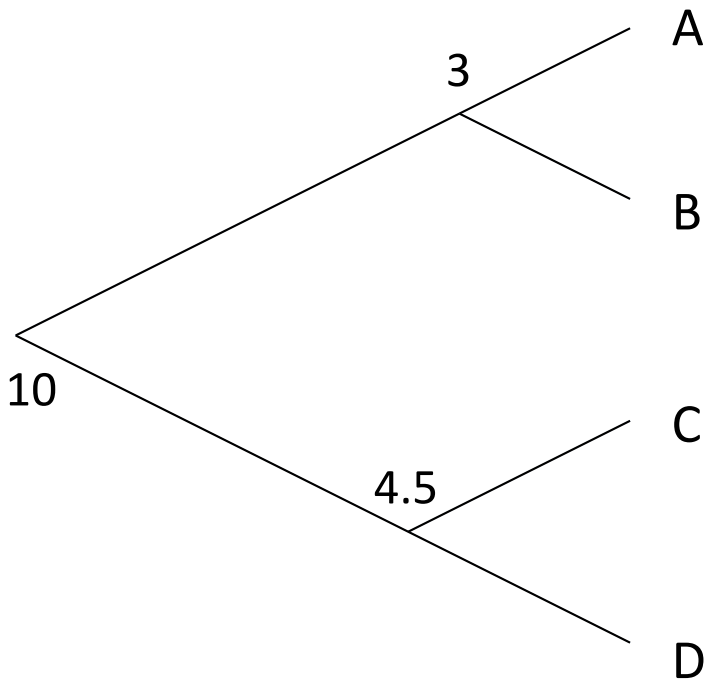
12 × 2/3 + 9 × 1/3 = 11




```
T1 = (a,b)
T2 = c
T3 = d
```

```
new_tree = ("Anc", T1, T2)
dist(new_tree, T3) = dist(T1, T3) x leaf_count(T1)/leaf_count(newTree) +
                     dist(T2, T3) x leaf_count(T2)/leaf_count(newTree)
```

Inferring time…



The `scale` function you write in lab will be useful…

Rhea
(South America)

Ostrich
(Africa)

TRIASSIC
200 million years ago

https://en.wikipedia.org/wiki/Rhea_(bird)
https://en.wikipedia.org/wiki/Ostrich
https://en.wikipedia.org/wiki/Gondwana

UPGMA assumes a molecular clock

	G	F	S
G	0	6	12
F	6	0	12
S	12	12	0

A note on biogeography/migrations