Parsimony

1: a. The quality of being careful with money or resources

b. The quality of being stingy

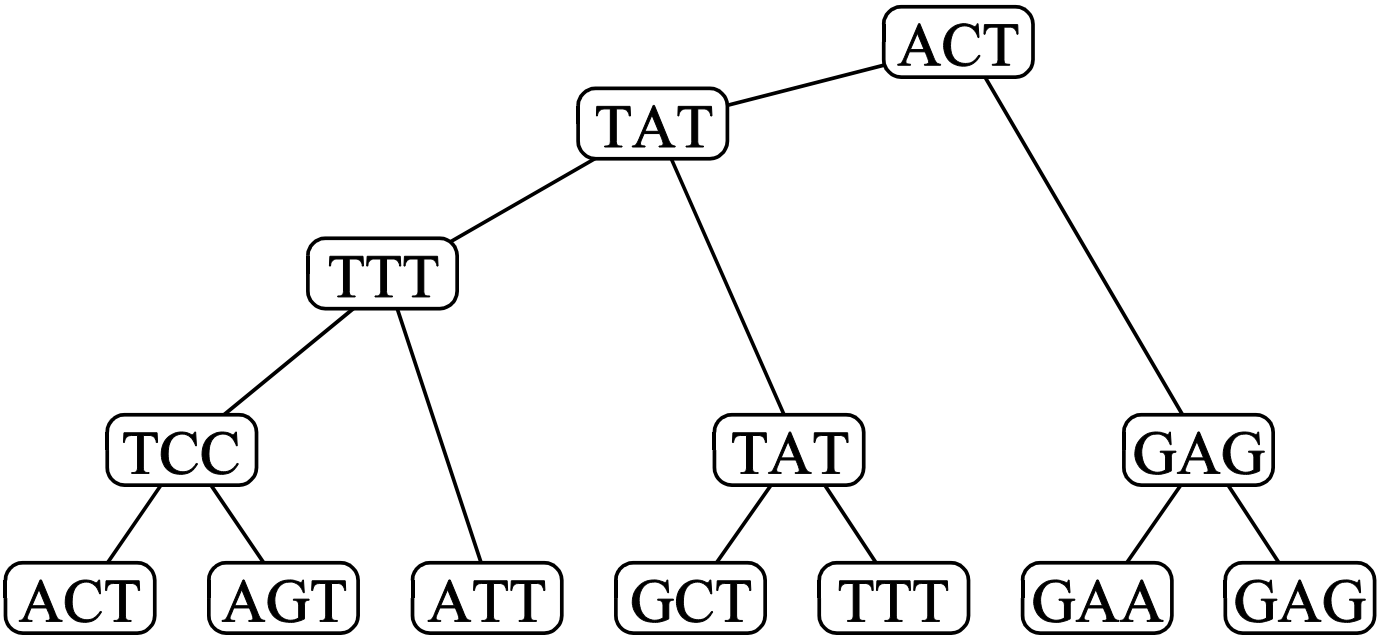
2: Economy in the use of means to an end; *especially*: Economy of explanation in conformity with Occam's razor

Tree parsimony:
Given a (rooted or unrooted) binary tree with sequences on the leaves, place sequences on the internal nodes so that the total number of mutations needed to explain the phylogeny is as small as possible.

For example...

Parsimony

The *parsimony* of a tree full of sequences is the sum of the numbers of mutations along each edge.

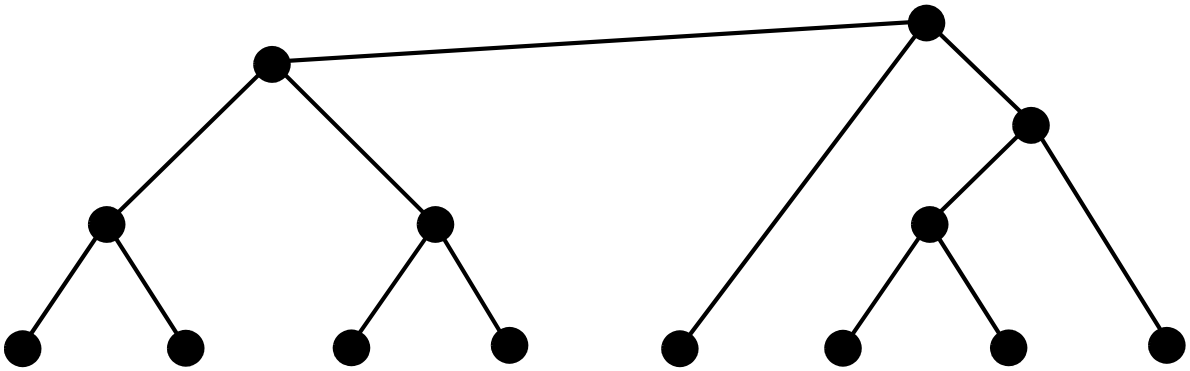


Parsimony = 18

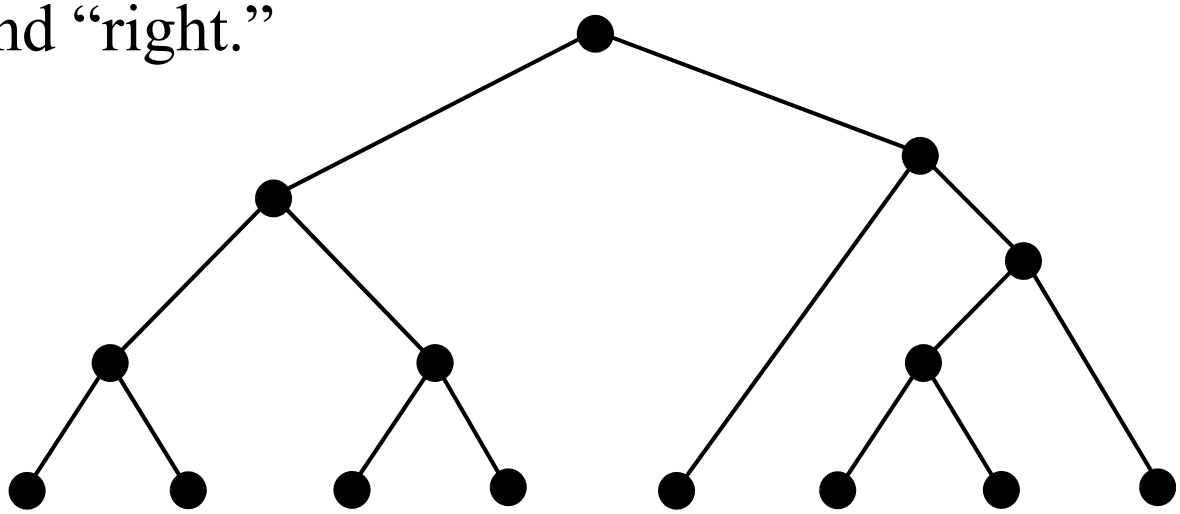
Incidentally, phylogenetic trees are binary trees, sometimes rooted, sometimes unrooted.

Binary Trees

An (unrooted) binary tree is a tree in which all nodes have degree 1 or 3.

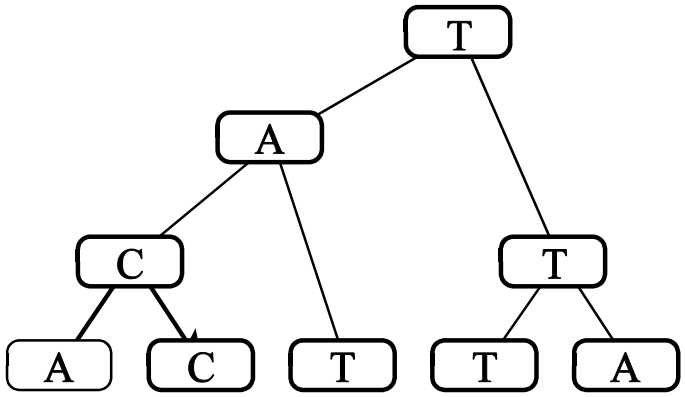


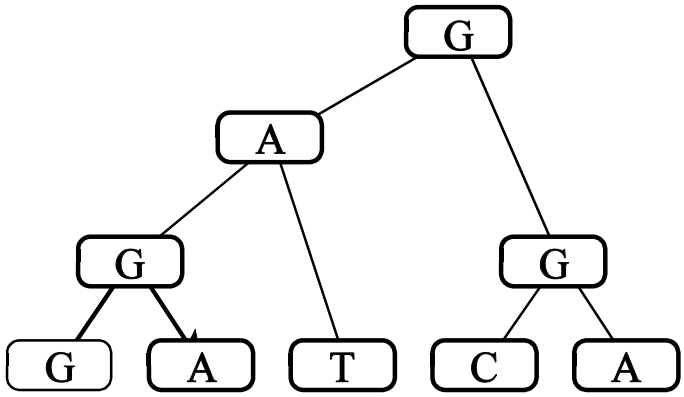
A rooted binary tree is a tree with exactly one vertex of degree 2 (the root) and all other vertices of degree 1 or 3. When a binary tree is rooted, each internal node can be thought of as having two "children" which are its two neighboring vertices further from the root than itself. They are called "left" and "right."

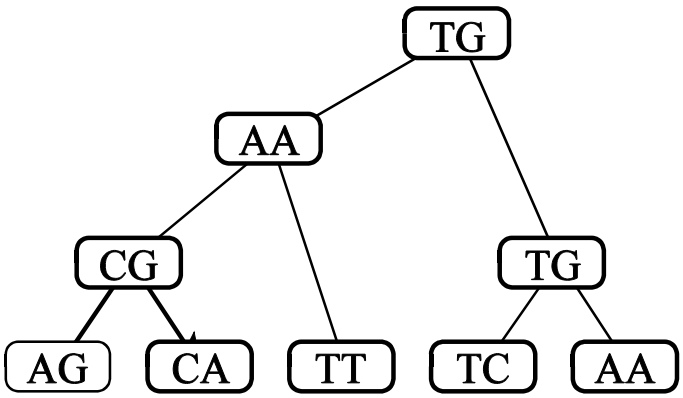


Your Turn

Compute the parsimony of each of these trees.

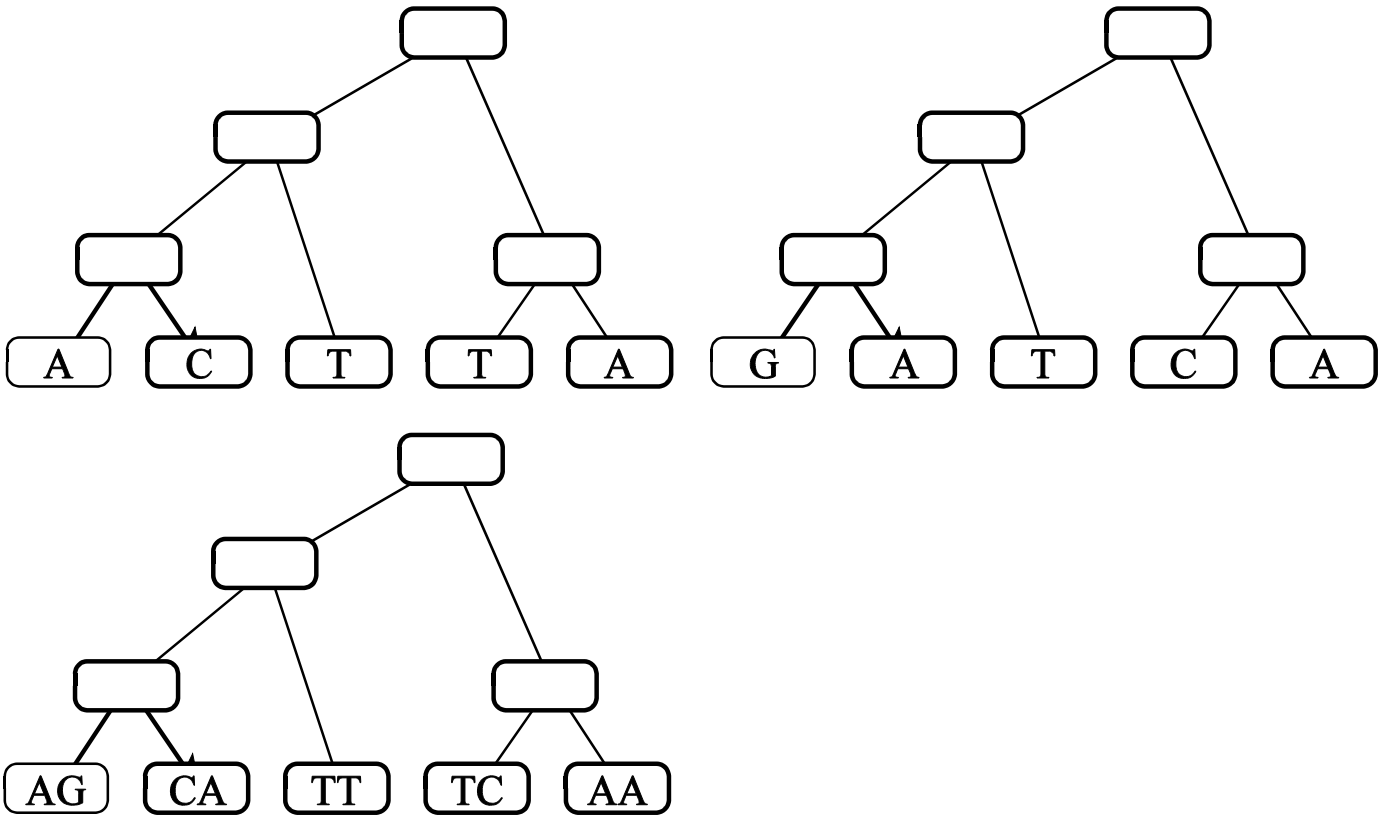






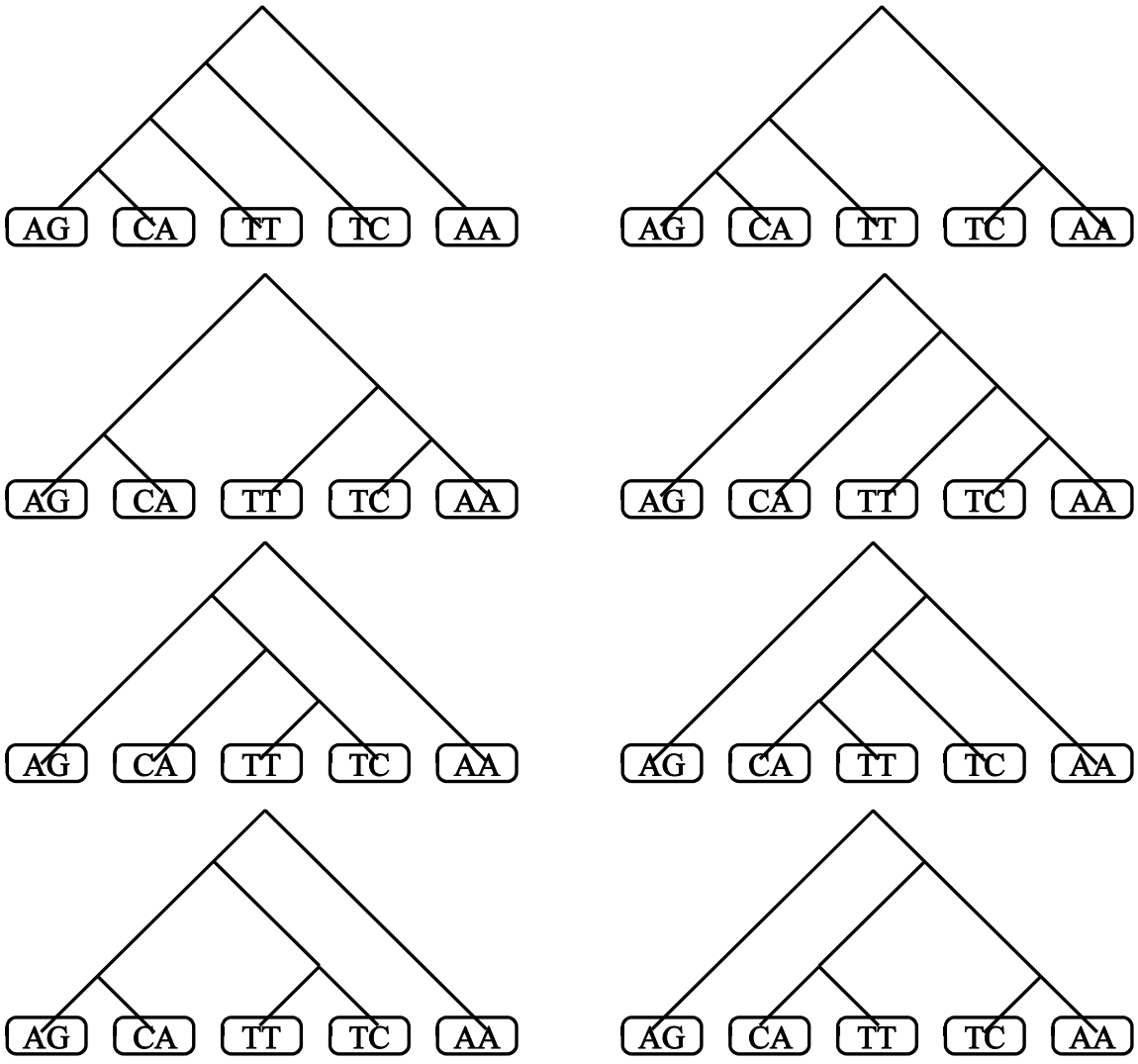
Optimal Parsimony

The way it usually works is as follows: We are given the sequences on the leaves of the tree (the nodes of degree 1) and we wish to select sequences for the internal nodes to minimize the total parsimony.



What Do We Do with This?

In a typical use of parsimony, we are given the sequences on the species alive today, and we wish to try different configurations of internal nodes (topologies) to see which one minimizes the parsimony.



Most Parsimonious Tree is NP-Complete

The problem of finding the most optimally parsimonious tree, including selecting the best topology, is *NP-Complete*.

This means that the problem of selecting the best topology and internal sequences is among that class of problems which nobody knows how to solve efficiently today, and for which there may exist no efficient algorithm.

So, what's hard?

- Finding the topology, or
- Finding the sequences to put on the internal nodes once a topology is selected?

Fitch's Algorithm

Stage I — Put sets on the nodes:

For each leaf node:

Construct a set on that leaf containing only the character on that node

For each internal node:

If the sets on the children have elements in common,

then the internal node is assigned the intersection of those two sets

If the sets on the children have no elements in common,

then the internal node is assigned the union of those two sets.

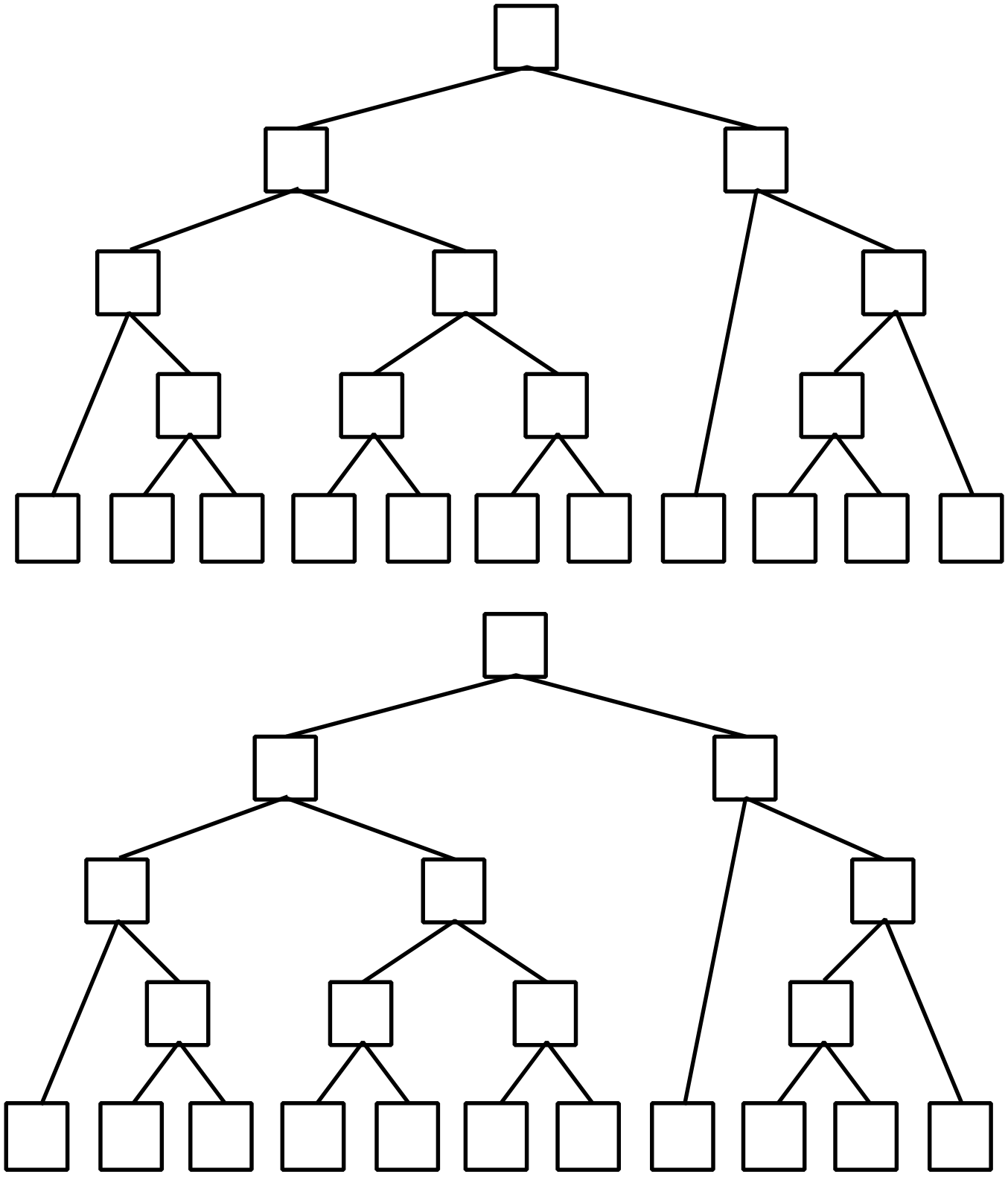
Stage II — Reading from the sets

Label the root with any element from its set.

For all other nodes, after its parent has been labeled, if the node's set contains the character assigned to the parent, use that character.

Otherwise, select any character from the set on that node.

Fitch's Algorithm

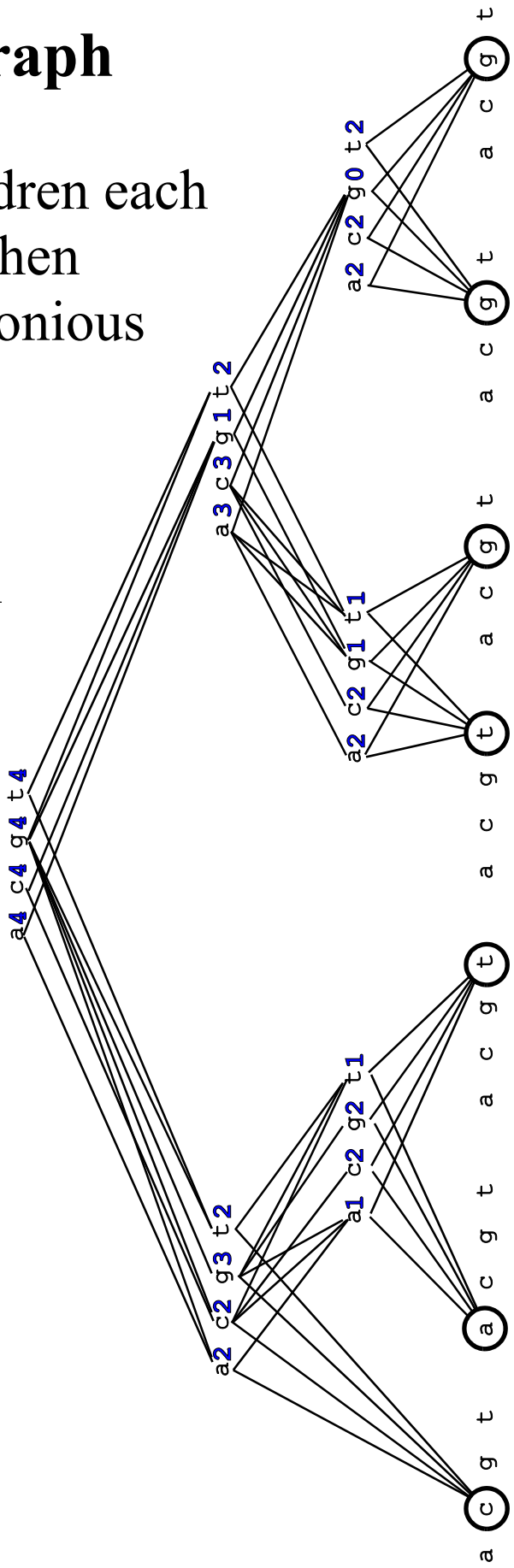


Parsimony Graph

This graph shows which children each internal node should select when building an optimally parsimonious tree.

In other words, any complete subtree of this graph gives an optimally parsimonious arrangement.

The numbers next to each base indicate the minimal cost to that point.



I Am a Node

My Name	S
Left Child	M
Right Child	N

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	T (data = c)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

Optimal Parsimony Algorithm

Given a rooted tree with a single nucleotide on each leaf, we wish to find all optimally parsimonious ways to put bases on the internal nodes, and the final optimal cost.

Algorithm:

1. *Set up:* Split each vertex into a set of 4 "base-vertices" labeled with the 4 bases, a, c, t, g.

2. *Leaves:* At each leaf there are four base-vertices. One base-vertex corresponds to the nucleotide on that leaf, the other three don't. Assign the base-vertex that corresponds to the nucleotide a cost of "0," and the three other base-vertices a cost of "∞."

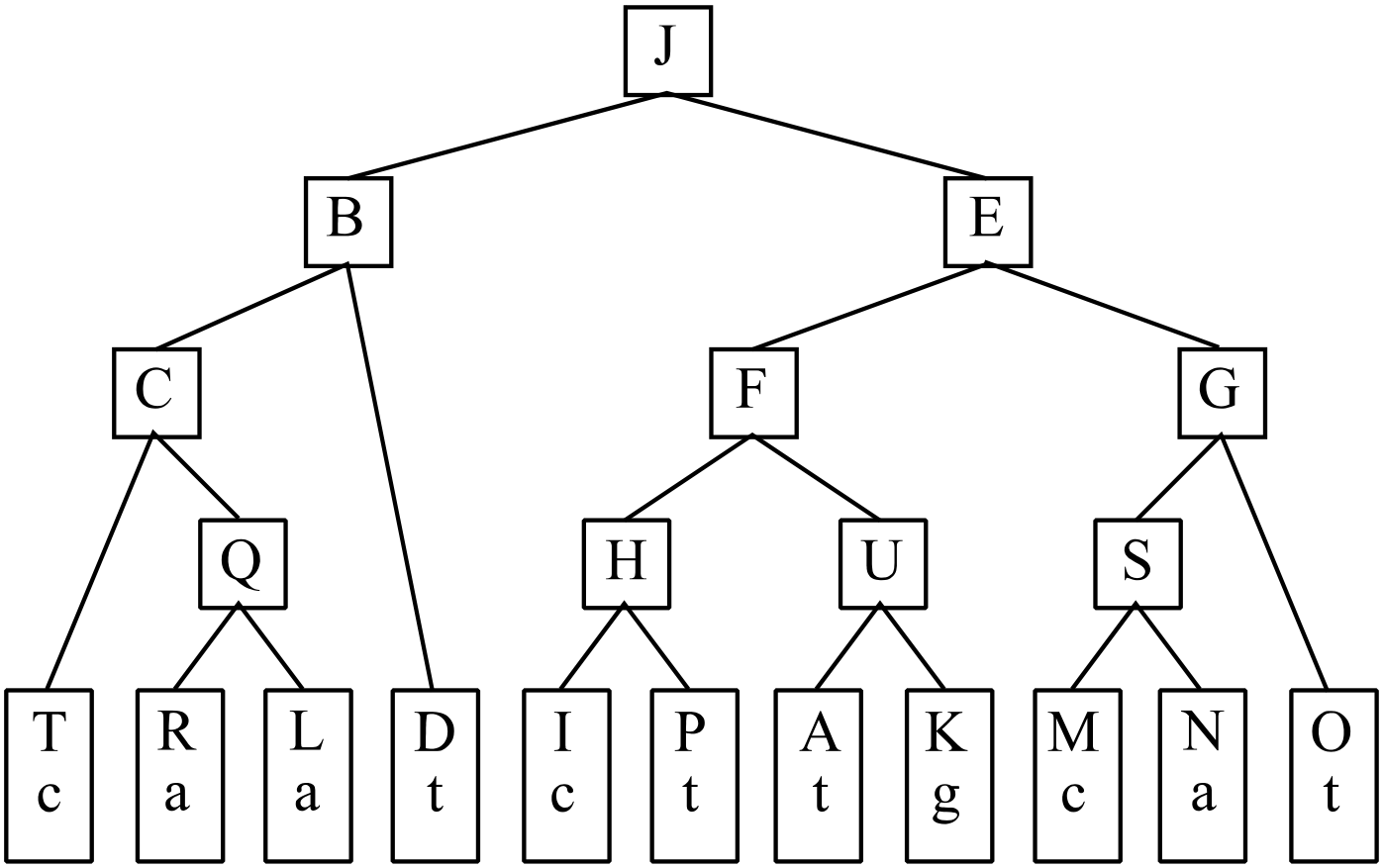
3. *Internal Nodes:* For each internal node:
   a. First makc sure its children have been assigned their costs.

Optimal Parsimony Algorithm

b. There are 4 base-vertices at that node. For each one determine the minimal cost of using that base-vertex at that node, which is the sum of a left cost and a right cost, and assign that cost to that base-vertex
   i. The left cost is obtained by looking at the list of costs on the base-vertices of the left child, adding 1 to those base-vertices with different nucleotide labels, and taking the minimum. Draw an edge to the base vertices in the left child that yield this minimum cost.
   ii. Same thing for the right cost.

4. *The Answer:* The minimum parsimony is the minimum cost appearing at the root, and any tree contained in the graph using a minimal cost base-vertex at the root gives an optimally parsimonious reconstruction.

The Secret Tree



All Optimal Reconstructions

```
[[[c,[a,a]],t],[[[c,t],[t,g]],[[c,a],t]]]   6
(((ca(aaa))tt)t(((ctt)t(ttg))t((caa)tt)))
(((ca(aaa))tt)t(((ctt)t(ttg))t((cca)tt)))
(((ca(aaa))tt)t(((ctt)t(ttg))t((cta)tt)))
(((cc(aaa))tt)t(((ctt)t(ttg))t((caa)tt)))
(((cc(aaa))tt)t(((ctt)t(ttg))t((cca)tt)))
(((cc(aaa))tt)t(((ctt)t(ttg))t((cta)tt)))
(((ct(aaa))tt)t(((ctt)t(ttg))t((caa)tt)))
(((ct(aaa))tt)t(((ctt)t(ttg))t((cca)tt)))
(((ct(aaa))tt)t(((ctt)t(ttg))t((cta)tt)))
```

Counting Optimally Parsimonious Trees

Once we have constructed the parsimony graph, we can answer the question "How many optimal reconstructions are there."

Algorithm:

1. *Set up:* Construct the parsimony graph

2. *Leaves:* At each leaf there are four base-vertices. One base-vertex corresponds to the nucleotide on that leaf, the other three don't. Assign the base-vertex that corresponds to the nucleotide the number "1," and the three other base-vertices a cost of "0."

3. *Internal Nodes:* For each internal node:
   a. First makc sure its children have been assigned their numbers.

Counting Optimally Parsimonious Trees

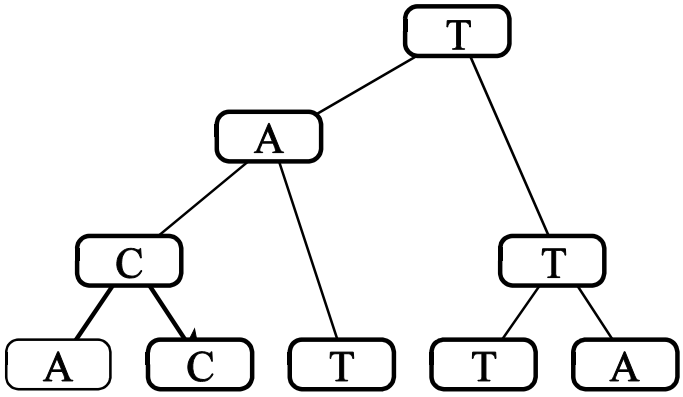
b. For each base-vertex add together the numbers on the base vertices in its left child to which it is connected, do the same for the right, and assign to this base-vertex the product of those sums.

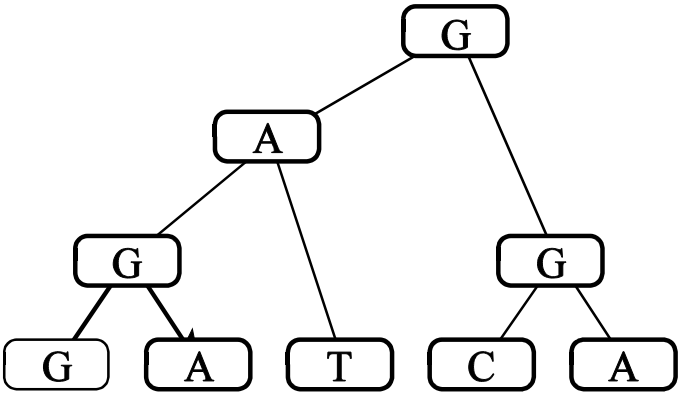
4. *The Answer:* The number of optimally parsimonious trees is the sum of the numbers on the minimal cost base-vertices at the root.

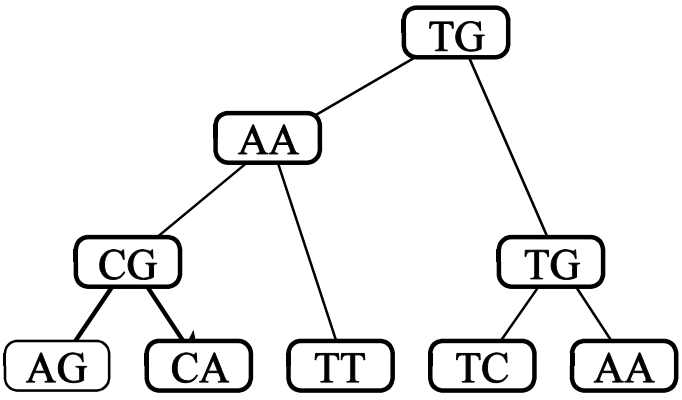
Handout #1 — Computing Parsimony

What is the parsimony of each of the following trees full of sequences:

Recall that the parsimony is the sum of the numbers of mutations across the edges.

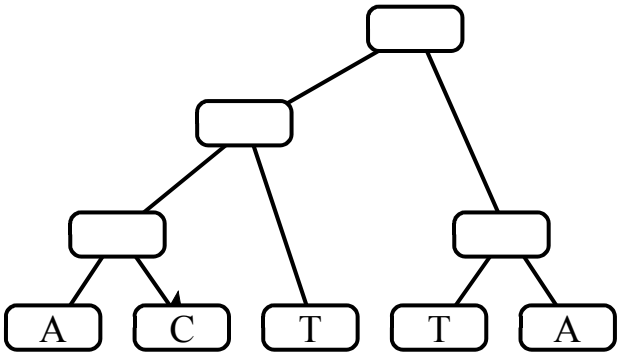


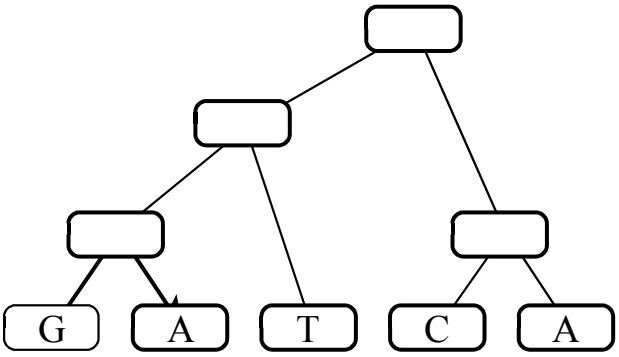


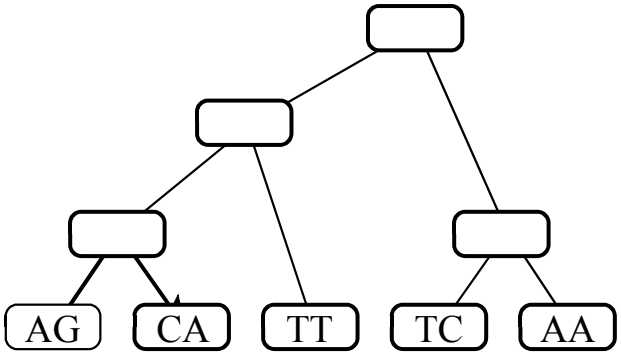


Handout #2 — Optimally parsimonious reconstructions

The leaves of these trees represent species that are alive today. The internal nodes represent hypothetical ancestral species, which may be identical to the species that we see today.



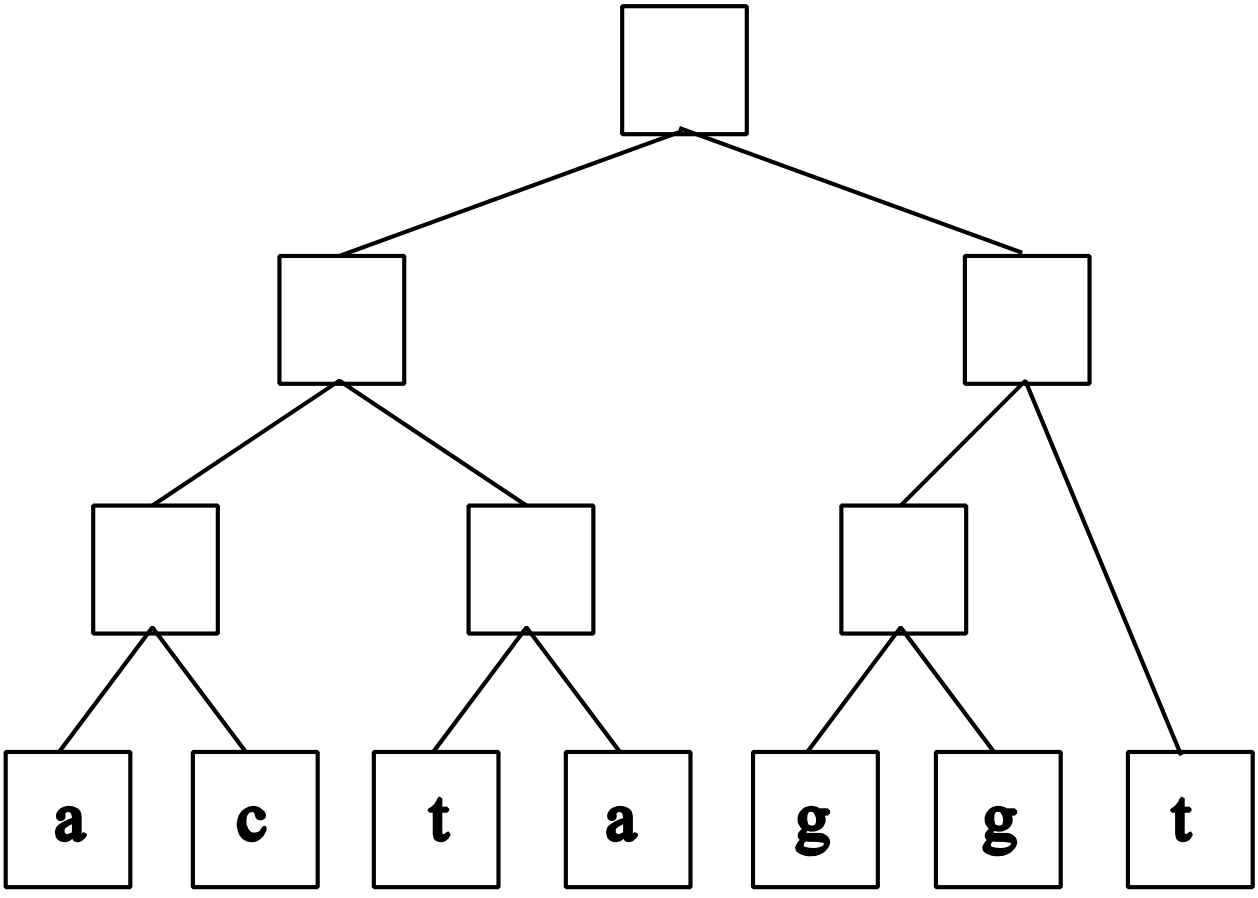




Handout #3 — Fitch's Algorithm for Optimal Parsimony

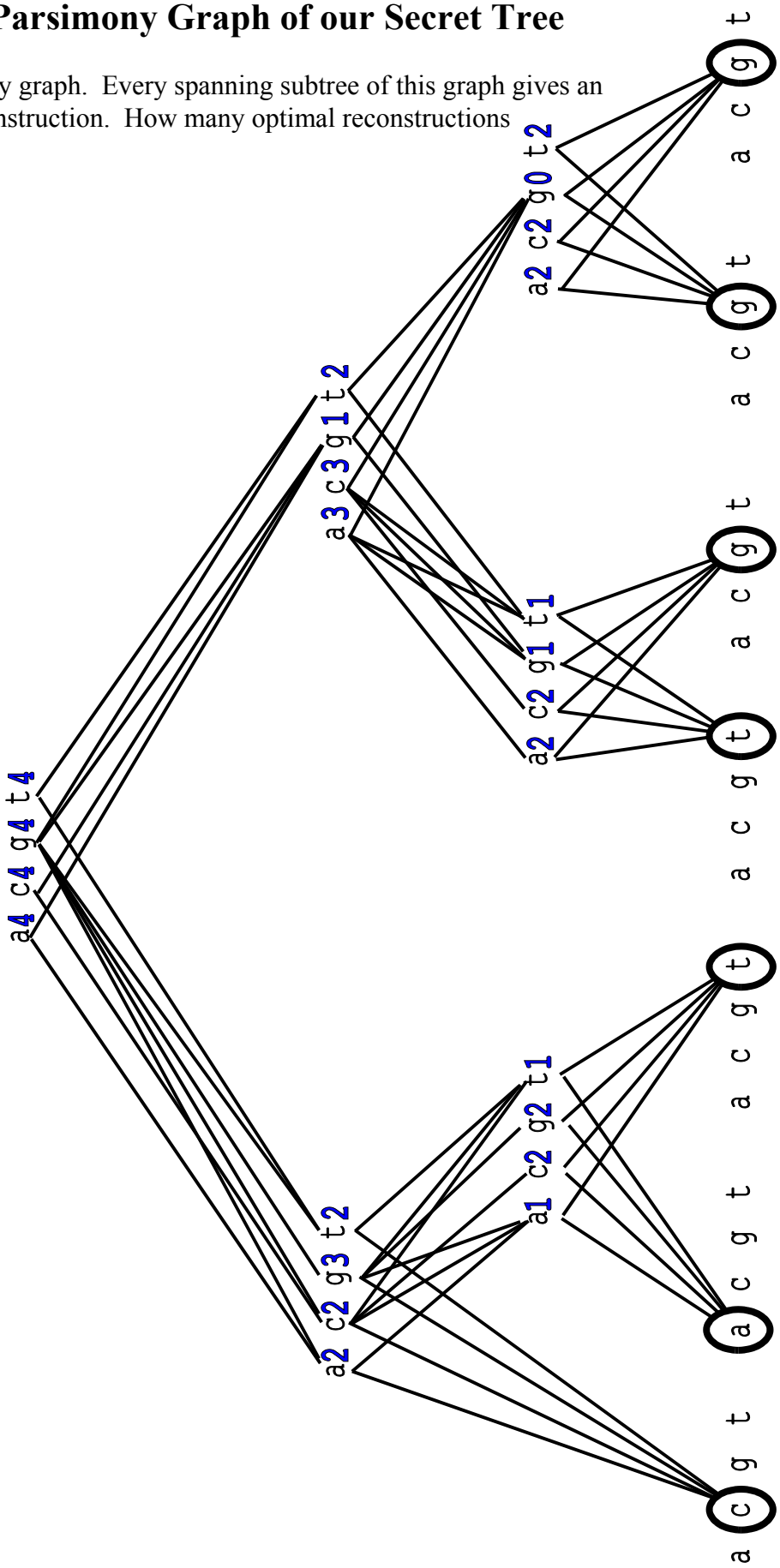
Use Fitch's algorithm to construct optimally parsimonious ancestral characters for the tree below, with the given characters on the leaves.

Fitch's Algorithm:
- For each leaf node:
  - Construct a set on that leaf containing only the character on that node
- For each internal node:
  - If the sets on the children have elements in common,
    - then the internal node is assigned the intersection of those two sets
  - If the sets on the children have no elements in common,
    - then the internal node is assigned the union of those two sets.



Handout #4 — The Parsimony Graph of our Secret Tree

Here is the optimal parsimony graph. Every spanning subtree of this graph gives an optimally parsimonious reconstruction. How many optimal reconstructions are there?



I Am a Node

My Name	J
Left Child	B
Right Child	E

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	B
Left Child	C
Right Child	D

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	E
Left Child	F
Right Child	G

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	C
Left Child	T
Right Child	Q

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	F
Left Child	H
Right Child	U

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	G
Left Child	S
Right Child	O

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	Q
Left Child	R
Right Child	L

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	H
Left Child	I
Right Child	P

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	U
Left Child	A
Right Child	K

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	S
Left Child	M
Right Child	N

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	T (data = c)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	R (data = a)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	L (data = a)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	D (data = t)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	I (data = c)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	P (data = t)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	A (data = t)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	K (data = g)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	M (data = c)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	N (data = a)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

I Am a Node

My Name	O (data = t)
Left Child	I am a leaf
Right Child	I am a leaf

My minimal costs			
a	c	g	t

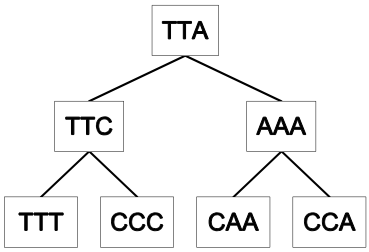
Optimal Children							
Left	Right	Left	Right	Left	Right	Left	Right

My numbers of subtrees			
a	c	g	t

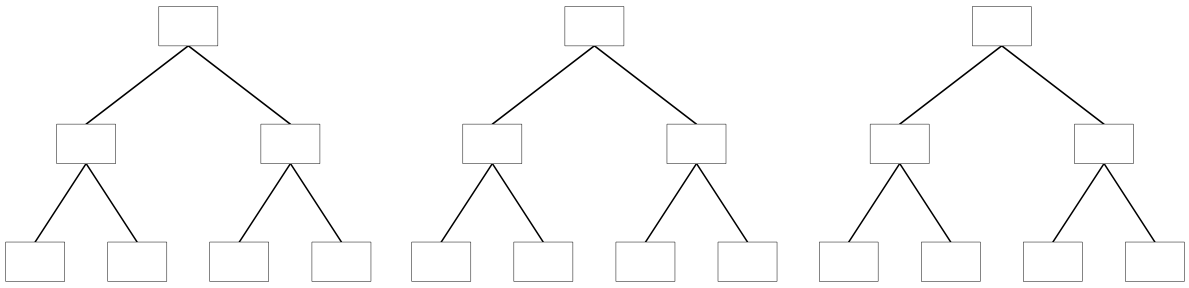
Exercises — Parsimony

Quick Concepts:

1. What is the total parsimony score of the tree shown to the right?



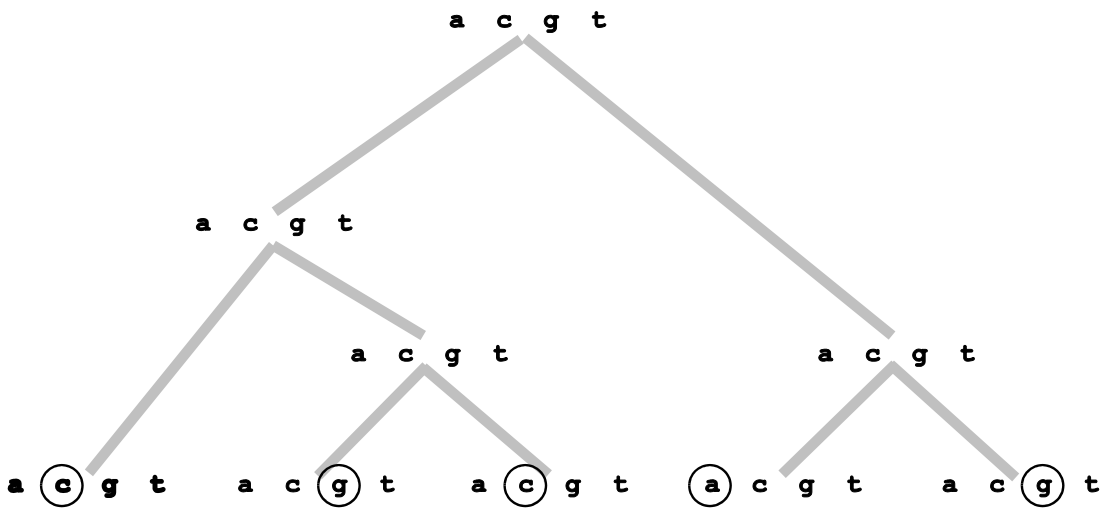
2. If we consider the bases in each position separately we get three trees, which you can fill in below. Which of them is individually optimally parsimonious?



3. What internal sequences achieve optimal parsimony for the tree in problem 1?

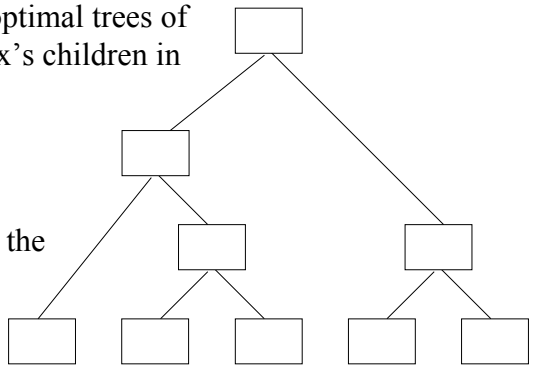
Presentation Problems:

4. Use Fitch's algorithm to find an optimally parsimonious reconstruction of the characters on the internal nodes of the tree given below. (For this problem, ignore the "a c g t" labels.)



5. Use the template above to find the optimal parsimony graph for the set of circled bases. What is the optimal parsimony? The shaded lines indicate the tree structure.

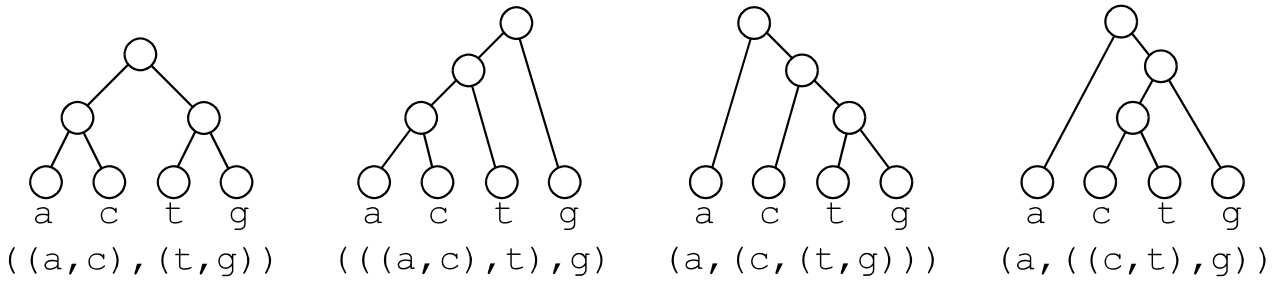
6. Count the number of optimal parsimony trees for the previous problem. The slickest solution assigns to each base-vertex the number of optimal trees of which it is the top, and does so by looking at that base-vertex's children in the parsimony graph.

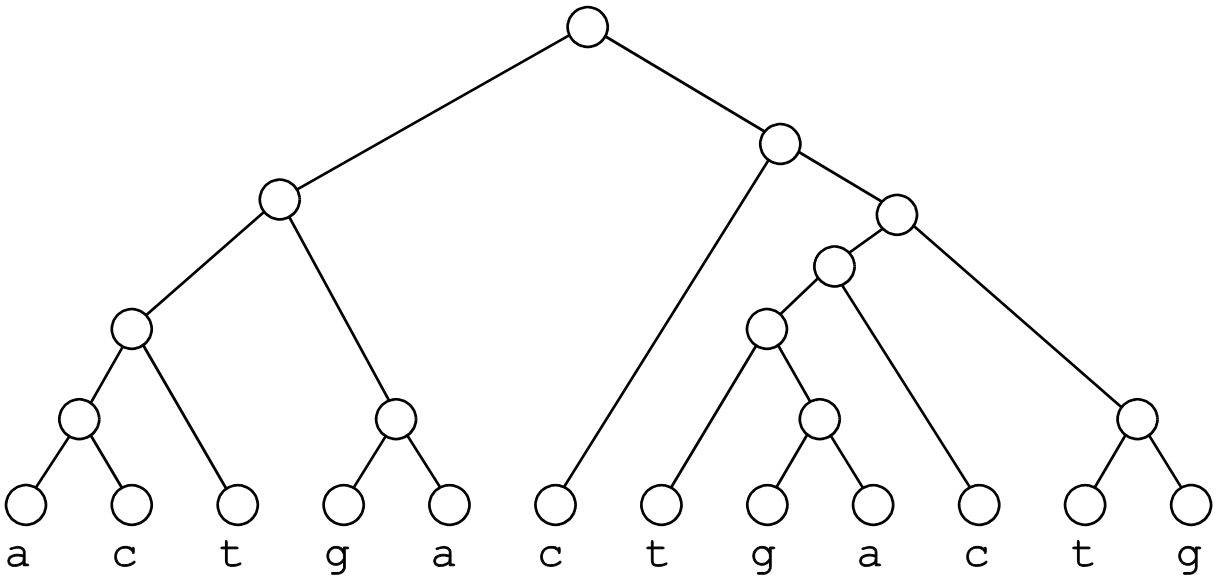


7. Draw the optimal trees in the previous two problems.

8. Place nucleotides in the leaves of the tree to the right so that the optimal parsimony is as large as possible. Prove that it is impossible to force the optimal parsimony to be larger.

9. Software tools often require the user to input trees in a text format. For example, the four trees shown below each have the text encoding indicated. Generating this text encoding is not difficult, but it can be tedious without a good trick. Fortunately, there is a good trick. Start at the root of the tree and walk around the tree counter-clockwise. When you depart from the root for the first time, say "Left Parenthesis." When you return to the root at the end of your trip, say "Right Parenthesis." Each leaf will be encountered once on your trip, and each internal node will be encountered three times. The first time you encounter an internal node... and so on and so on and so on... Fill in the missing details. Then work in pairs to write down the parenthesized expression for the large tree below. One person should walk around the tree and call out the parenthesization while the other person writes it down.





10. There is a one-to-one correspondence between the commas in the text representation of a tree and that tree's internal nodes. So, for example, you could be asked (and are being asked) if the tree: `(((acc)c(aac))gg)g(ta(aaa))` was optimally parsimonious given its leaves, but with freedom to assign characters to its internal nodes.

11. A given set of nucleotides on the leaves of a tree may yield many optimally parsimonious reconstructions. Selecting a "most representative" one then becomes something of an art. One possibility is to pay attention to the differences across the edges (0 or 1 on each edge for each optimal tree) and take the average over all optimal trees for each edge. For example, in problem 7 you found all optimal trees for the given set of bases on the leaves, and in that case if you average these differences for the left child edge of the root you get 1/3. Same for the right child. What are the other six averages?

12. Suppose I give you a tree with 10 leaves, with characters assigned to each leaf, and the internal nodes empty. Then you use Fitch's algorithm to assign characters to the internal nodes in an optimally parsimonious way.
    a. Is it possible for there to be only one optimal tree under Fitch's algorithm?
    b. Suppose a rooted, binary tree has 10 leaves, all of which are labeled "a," except for two of them, which are labeled "c." Prove or disprove that Fitch's algorithm must give at least two optimal reconstructions for that tree.