CS 68: BIOINFORMATICS

Prof. Sara Mathieson
Swarthmore College
Spring 2018

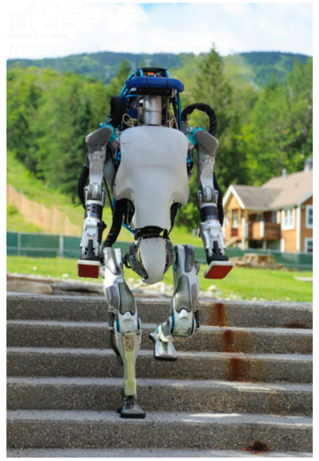
Outline: Apr 16

Remind me to hand back Lab 5

- Midterm 2 review session (feel free to jump in with random questions)
- Begin with tree algorithm overview + algorithm practice
- Population genetics overview
- Wed: HMM and PCA recaps

  - Lab 8 due TODAY
  - Office hours TODAY 3-5pm (let me know if you want to meet before/after)
  - Part 1 of Lab 8 worth more than Part 2

  - Make sure you have a study guide for Midterm 2
  - Can bring a handwritten “cheat-sheet”, front & back



TODAY!

Jennifer Barry
Talk: Arms on Legs
Monday, April 16 11:45am-12:45pm
SCI 199

Humanoid robots fit into our world. They can climb escalators, walk down supermarket aisles, use hand tools, and reach the top shelf. However, they must constantly exert energy and computational power just to stay balanced and upright. A humanoid robot can lean over to grab something that's almost out of reach, but if its mass shifts too far, it will fall. The robot may fit in the supermarket aisle but every time it lifts a foot, it has to figure out how to put it down without crashing into a shelf. Hardware with arms and legs is already complicated, but a humanoid also needs an articulated neck or multiple cameras, because there is no single camera placement that can simultaneously see the next step in a staircase and a vase on a high shelf. In this talk, we explore the joys and challenges of manipulation with humanoid robots.

Biography: Jennifer Barry graduated with High Honors from Swarthmore College in 2007 with a physics major and mathematics minor. She is currently a Senior Roboticist at Boston Dynamics where she works on manipulation with the next generation Atlas robot. Her Ph.D. work at MIT was on planning for complex manipulation problems, where she was co-advised by Leslie Kaelbling and Tomas Lozano-Perez.

Sequence variation vocabulary

- **Haplotype:** sequence from a single **chromosome** (i.e. one row from our input to perfect phylogeny or PCA). Humans are **diploid** (2 chromosomes)
- **SNP: single nucleotide polymorphism** (caused by a single mutation in the past)
- **Allele:** type of observed variation (**biallelic** could have an “A” allele and “a” allele, often we encode these as “0” and “1”)
- **Ancestral:** allele of the ancestor of all sequences in the sample
- **Derived:** allele that doesn’t match the ancestral (result of the mutation)

Relevant for projects

- **Reference** (REF in VCF files): allele of whoever we sequenced first
- **Alternate** (ALT in VCF files): allele(s) that don’t match the reference

Where does sequence variation come from?

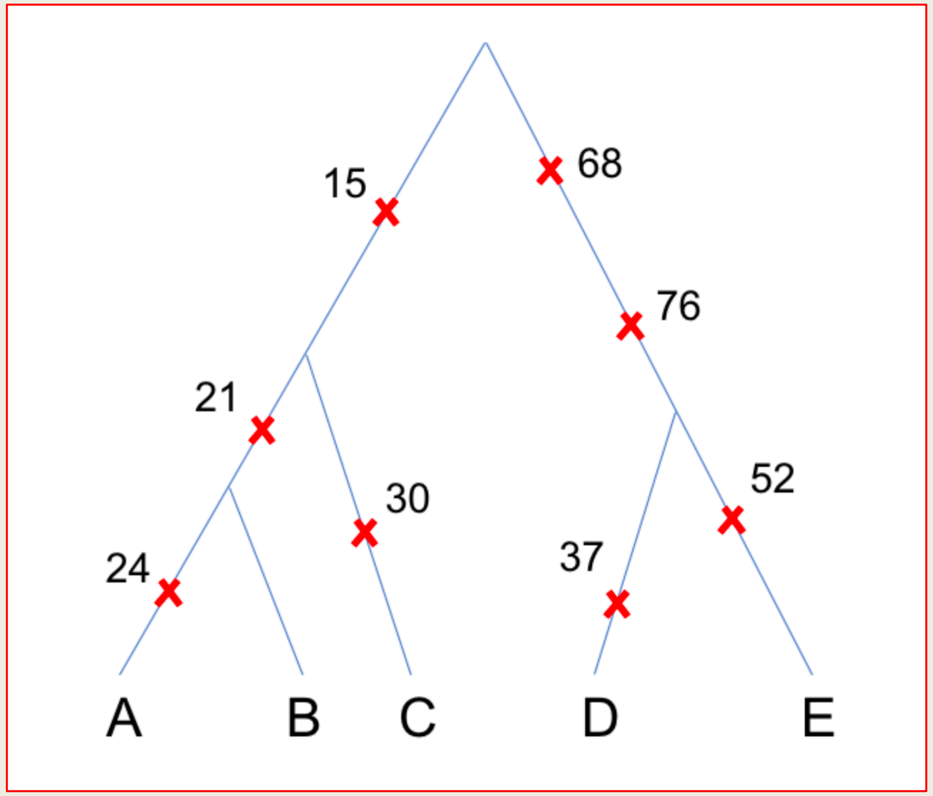
See Handout 10

Assumption:

- Ancestral state was all 0's

What actually happened:

8 mutations occurred in the past



Where does sequence variation come from?

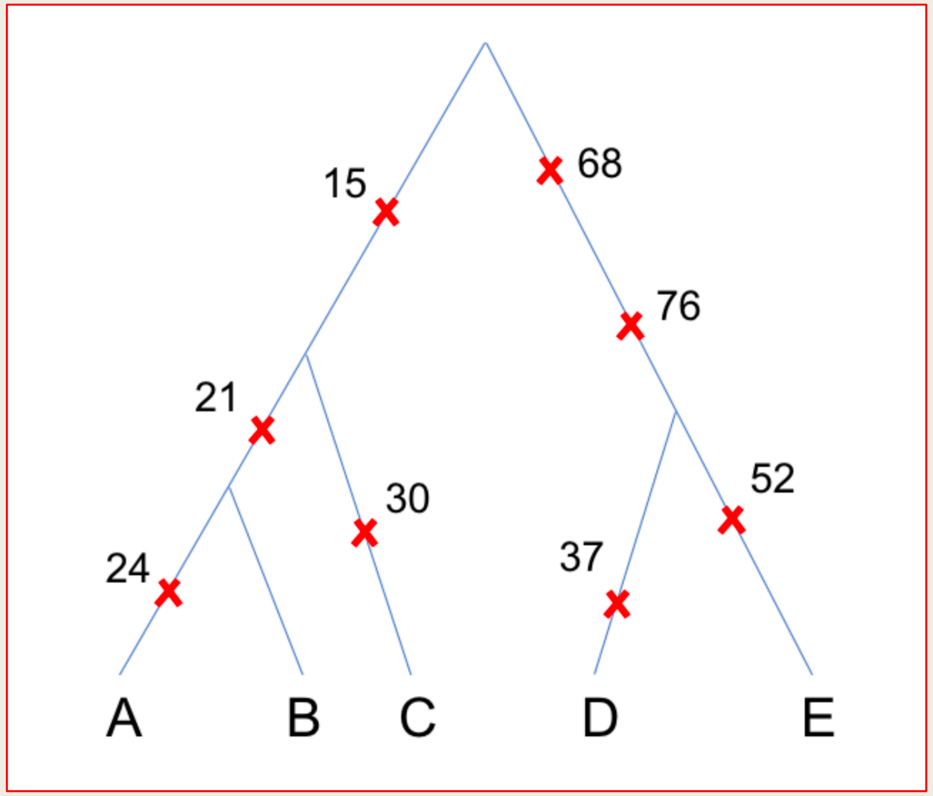
See Handout 10

Assumption:

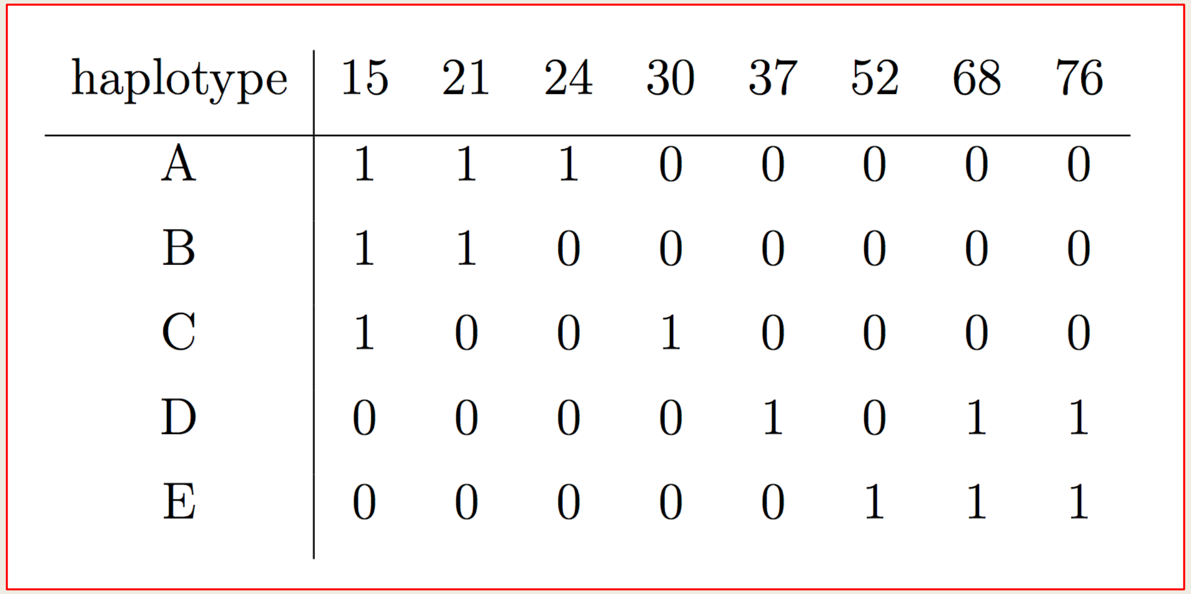
- Ancestral state was all 0's

What actually happened:

8 mutations occurred in the past



What we observe: 8 SNPs in this region



haplotype	15	21	24	30	37	52	68	76
A	1	1	1	0	0	0	0	0
B	1	1	0	0	0	0	0	0
C	1	0	0	1	0	0	0	0
D	0	0	0	0	1	0	1	1
E	0	0	0	0	0	1	1	1

Where does sequence variation come from?

See Handout 10

Assumption:

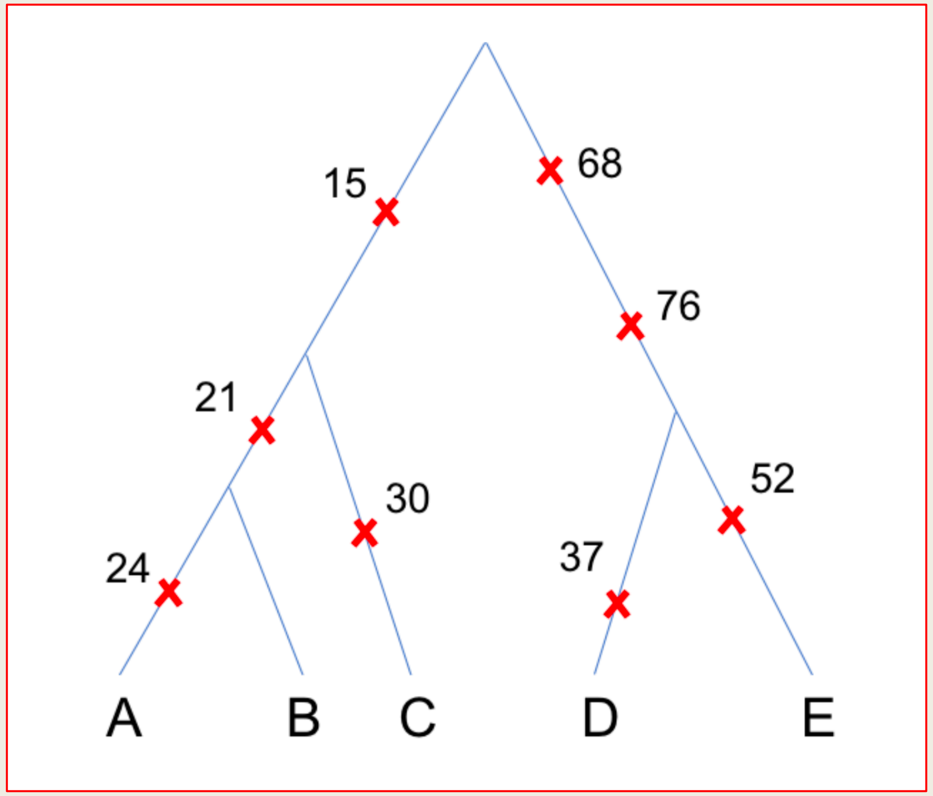
- Ancestral state was all 0's

What we observe: 8 SNPs in this region

haplotype	15	21	24	30	37	52	68	76
A	1	1	1	0	0	0	0	0
B	1	1	0	0	0	0	0	0
C	1	0	0	1	0	0	0	0
D	0	0	0	0	1	0	1	1
E	0	0	0	0	0	1	1	1

What actually happened:

8 mutations occurred in the past



δ	A	B	C	D	E
A	0	1	3	6	6
B		0	2	5	5
C			0	5	5
D				0	2
E					0

From the SNPs we can obtain a dissimilarity map! We could also get one through pairwise sequence alignment.

Where does sequence variation come from?

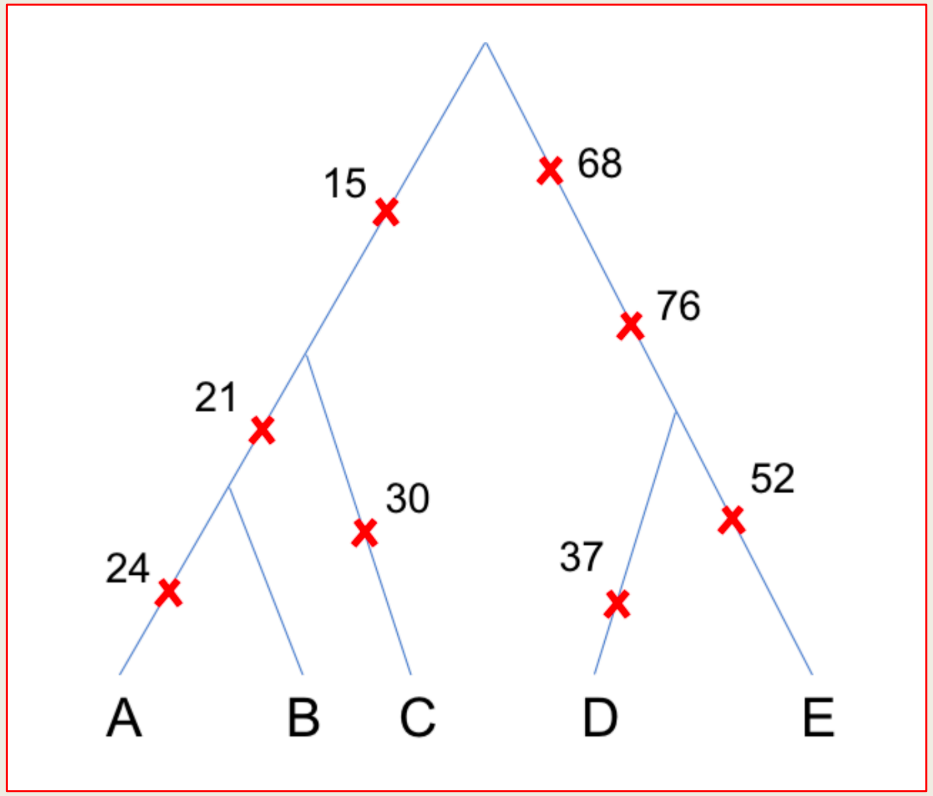
See Handout 10

Assumption:

- Ancestral state was all 0's

What actually happened:

8 mutations occurred in the past



What we observe: 8 SNPs in this region

haplotype	15	21	24	30	37	52	68	76
A	1	1	1	0	0	0	0	0
B	1	1	0	0	0	0	0	0
C	1	0	0	1	0	0	0	0
D	0	0	0	0	1	0	1	1
E	0	0	0	0	0	1	1	1

From the SNPs we can obtain a dissimilarity map! We could also get one through pairwise sequence alignment.

δ	A	B	C	D	E
A	0	1	3	6	6
B		0	2	5	5
C			0	$\boxed{5}$	5
D				0	2
E					0

Dissimilarity maps

- Record number of pairwise differences (which could be obtained from a pairwise sequence alignment)

A *dissimilarity map* δ is a function mapping pairs of samples from a set $\mathcal{X}$ to distances. It has the following two properties, but not necessarily the triangle inequality.

1. $\delta(x, x) = 0$
2. $\delta(x, y) = \delta(y, x)$

Example:

δ	A	B	C	D	E
A	0	1	3	6	6
B		0	2	5	5
C			0	5	5
D				0	2
E					0

Tree Algorithm Overview

Tree Algorithm Overview:

"Nothing in biology makes sense except in the light of evolution."
-Theodosius Dobzhansky

most of the second half relates to trees!

- Building phylogenetic trees: UPGMA & NJ

- Ancestral state reconstruction: Fitch, Sankoff

- Perfect phylogeny: naïve algorithm, Gusfield

- Coalescent events indirectly create a tree
- PCA can be interpreted as splits in a tree

Tree Algorithm Overview:

"Nothing in biology makes sense except in the light of evolution."
-Theodosius Dobzhansky

most of the second half relates to trees!

- Building phylogenetic trees: UPGMA & NJ

  Input: symmetric dissimilarity map for pairs of species (can be distantly related)
  Output: tree topology (structure) and branch lengths

- Ancestral state reconstruction: Fitch, Sankoff

- Perfect phylogeny: naïve algorithm, Gusfield

- Coalescent events indirectly create a tree
- PCA can be interpreted as splits in a tree

Tree Algorithm Overview:

"Nothing in biology makes sense except in the light of evolution."
-Theodosius Dobzhansky

most of the second half relates to trees!

- Building phylogenetic trees: UPGMA & NJ

  Input: symmetric dissimilarity map for pairs of species (can be distantly related)
  Output: tree topology (structure) and branch lengths

- Ancestral state reconstruction: Fitch, Sankoff

  Input: leaf labels of all species/samples for a single site/feature AND rooted tree topology
  Output: internal node labels (i.e. ancestral state) + total score of all mutations

- Perfect phylogeny: naïve algorithm, Gusfield

- Coalescent events indirectly create a tree
- PCA can be interpreted as splits in a tree

Tree Algorithm Overview:

"Nothing in biology makes sense except in the light of evolution."
-Theodosius Dobzhansky

most of the second half relates to trees!

- Building phylogenetic trees: UPGMA & NJ

  Input: symmetric dissimilarity map for pairs of species (can be distantly related)
  Output: tree topology (structure) and branch lengths

- Ancestral state reconstruction: Fitch, Sankoff

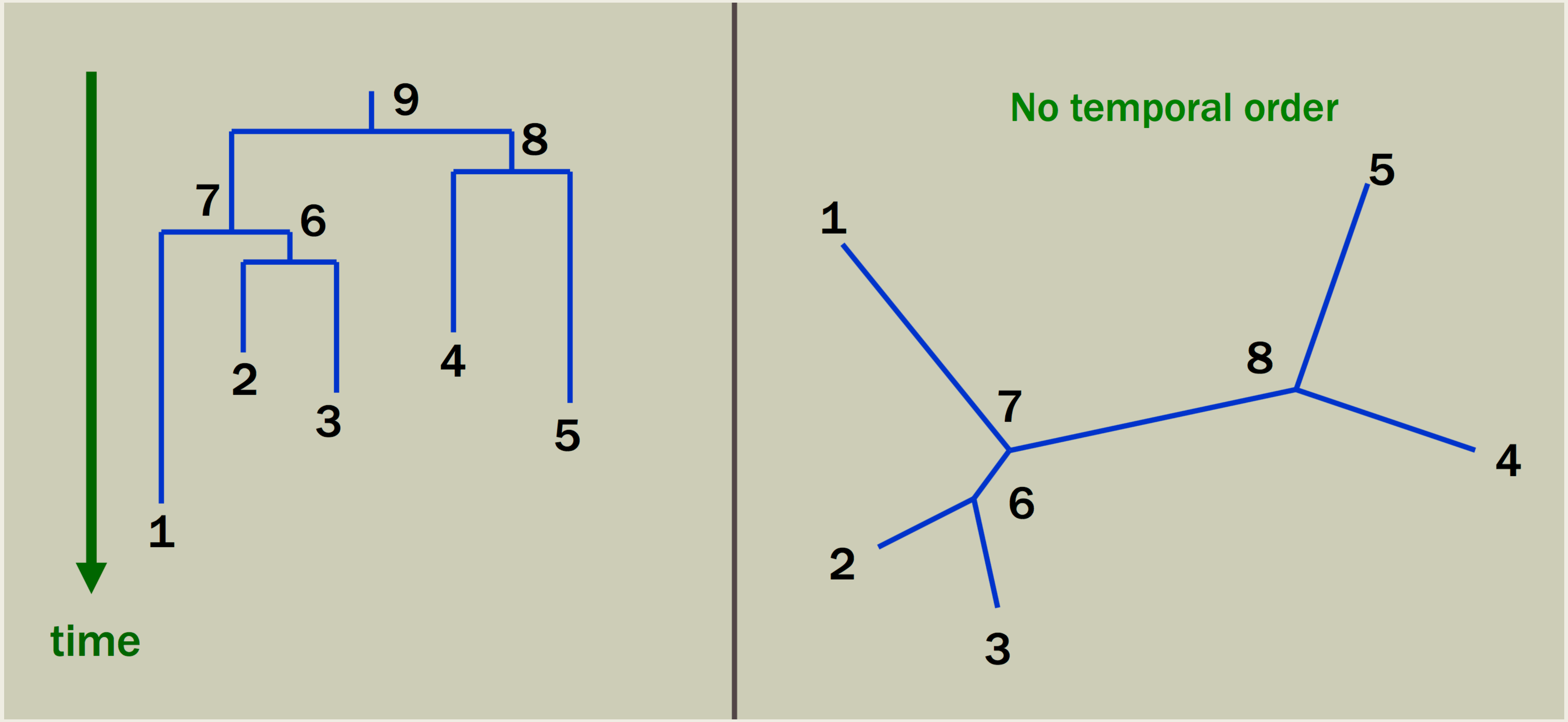
  Input: leaf labels of all species/samples for a single site/feature AND rooted tree topology
  Output: internal node labels (i.e. ancestral state) + total score of all mutations

- Perfect phylogeny: naïve algorithm, Gusfield

  Input: matrix of data (samples on rows, sites on columns), same as PCA
  Output: yes/no on perfect phylogeny, Gusfield also builds the tree and returns the number of repeat mutations

- Coalescent events indirectly create a tree
- PCA can be interpreted as splits in a tree

Rooted vs. unrooted trees

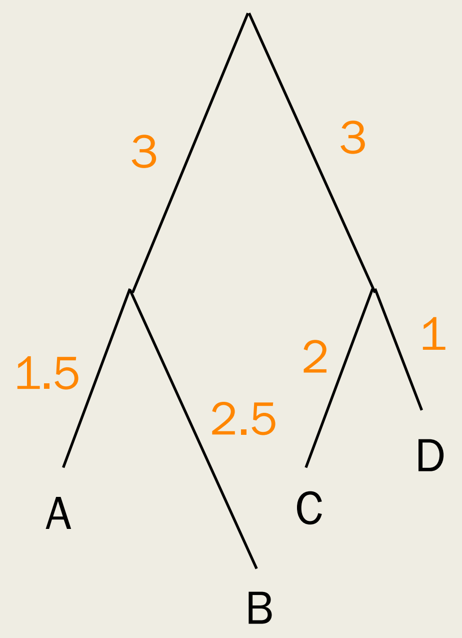


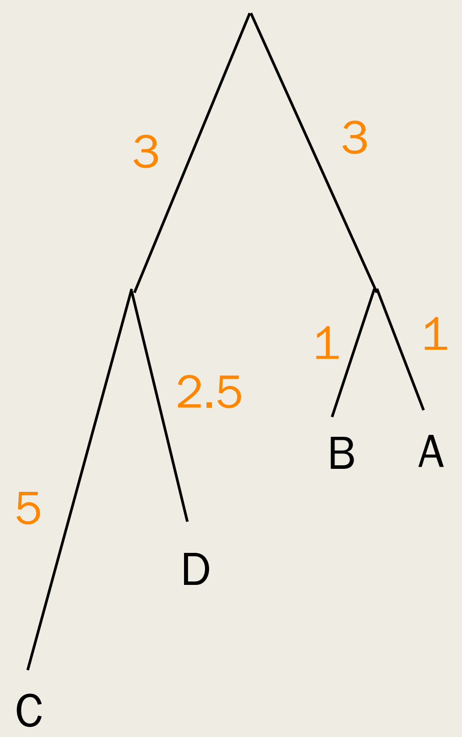
Note: neither of these are ultrametric trees

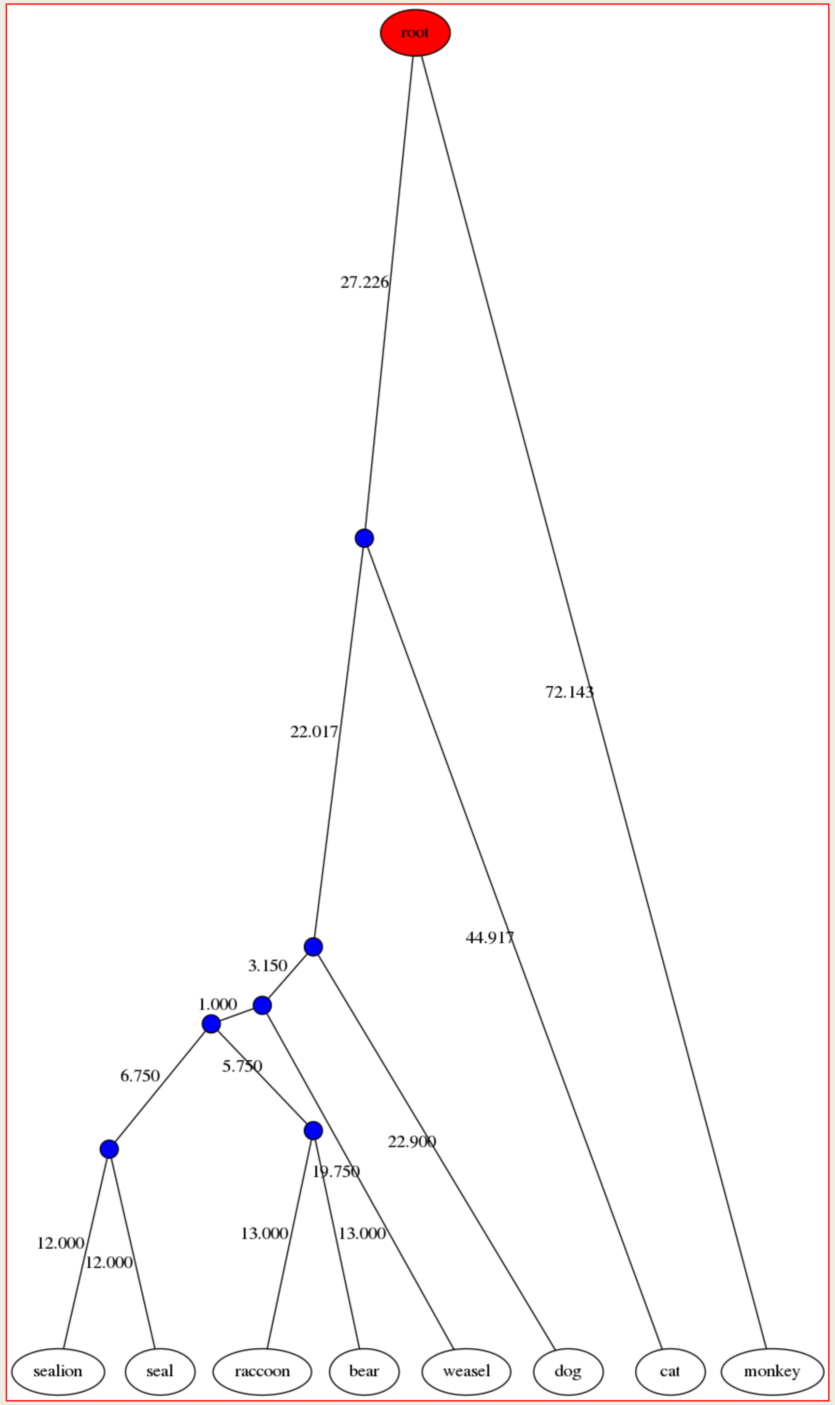
Credit: Ameet Soni

Topology and branch lengths

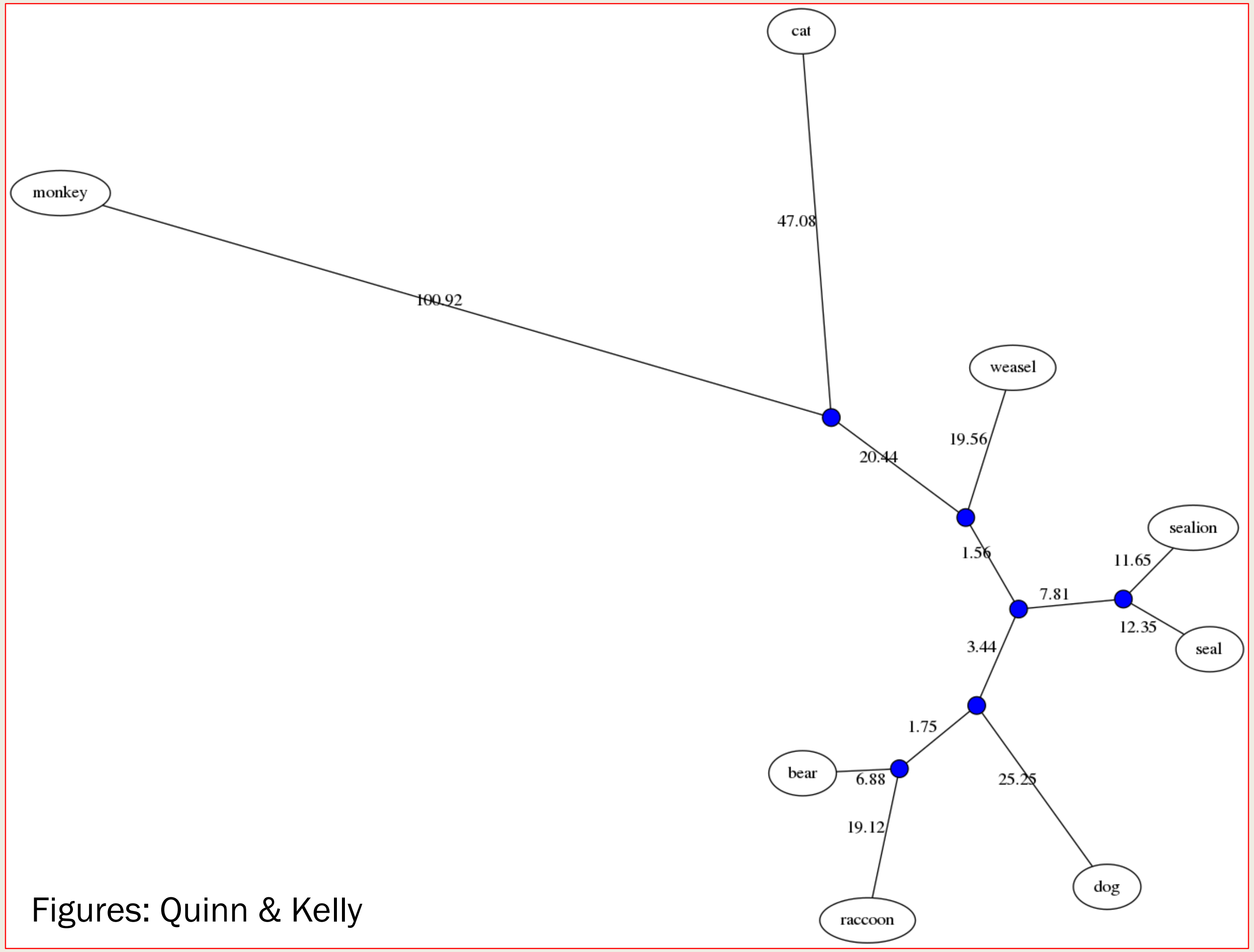
- The two trees below have different branch lengths but the same **topology**







Mammals dataset: there are topology differences!



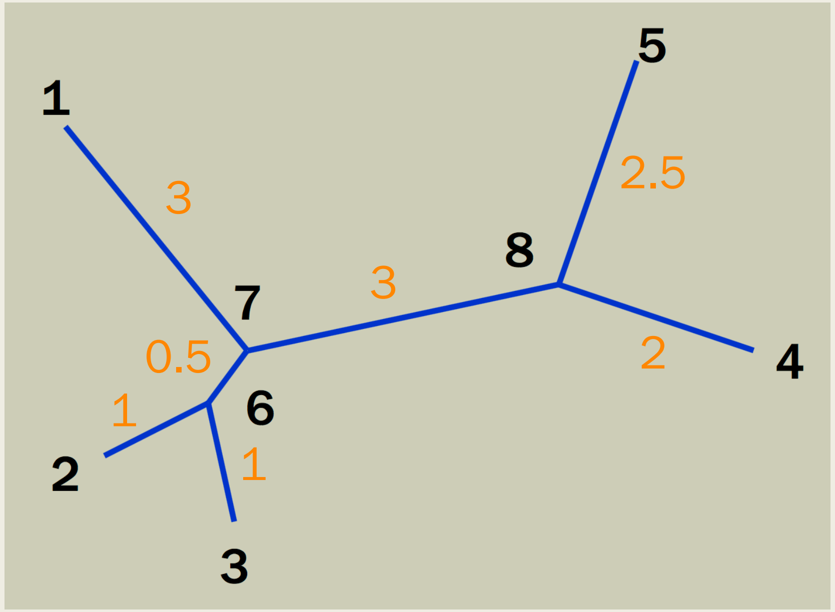
Figures: Quinn & Kelly

Tree metric

- Any method that produces a tree topology and edge weights **induces** a tree metric
- Both UPGMA and NJ produce tree metrics
- These may not match the input dissimilarity maps, but we would like them to be close

A dissimilarity map δ is a *tree metric* if $\exists$ a tree topology and edges weights such that $\forall\ x, y \in \mathcal{X}$,

$$\delta(x,y) = \sum \text{all edge weights in the path from } x \text{ to } y.$$

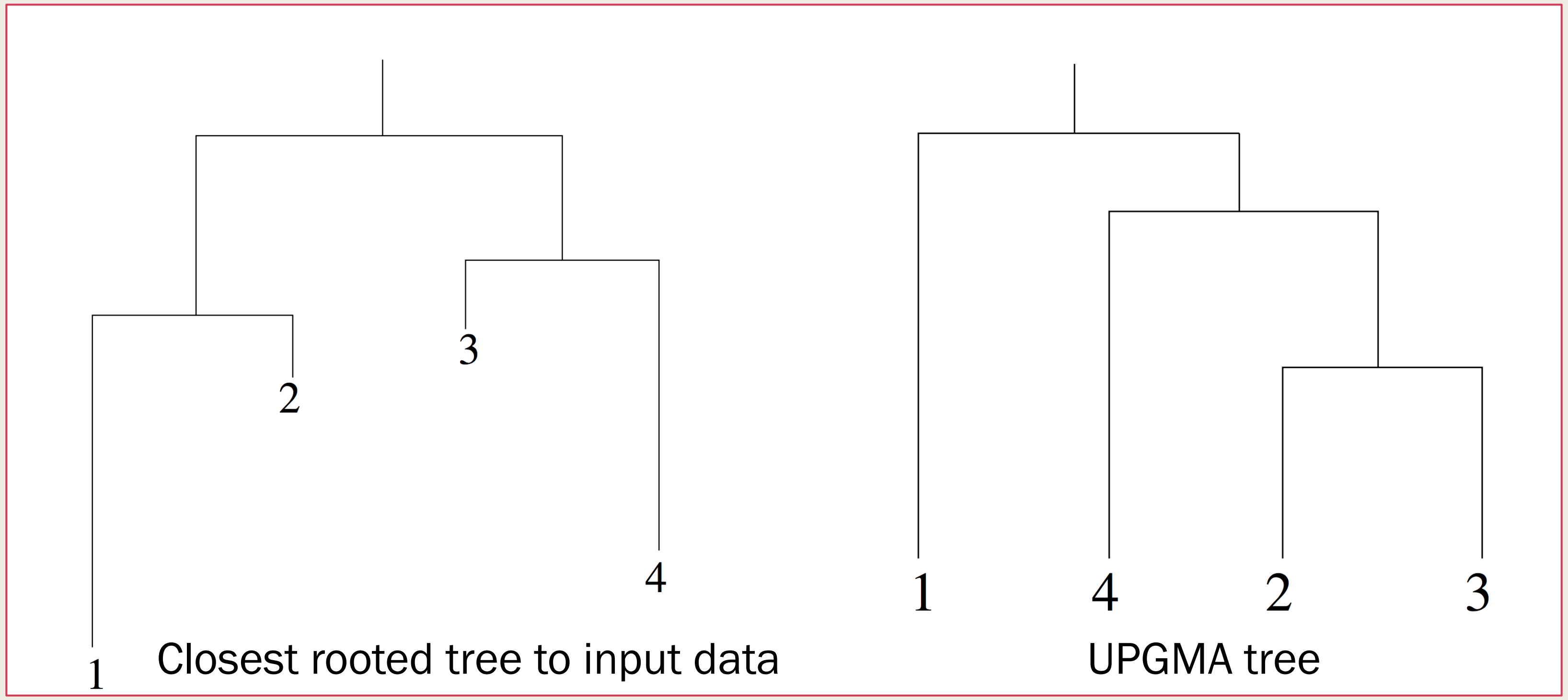


Examples of induced tree metric:

- delta(2,5) = 1+0.5+3+2.5 = 7
- delta(1,4) = 3+3+2 = 8

UPGMA

UPGMA produces rooted, ultrametric trees

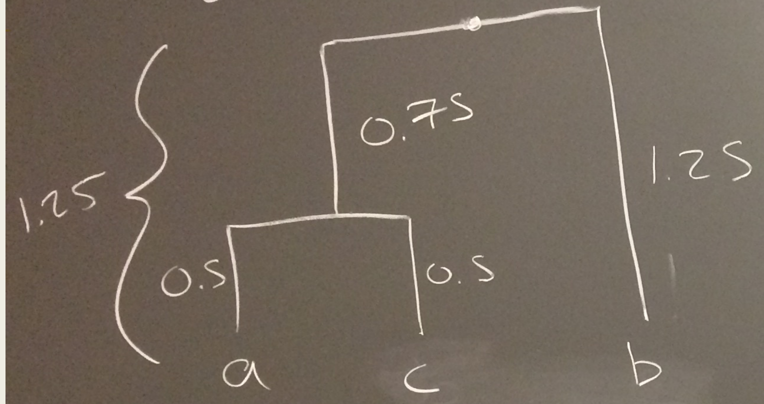


- Depending on the dataset, UPGMA can make unrealistic assumptions about the rate of evolution
- UPGMA assumes a constant rate of evolution on all branches ("molecular clock", which means that the number of mutations is directly proportional to time)
- If the molecular clock assumption holds, it can be helpful for estimating evolutionary events
- The flexibility of NJ is almost always more desirable, but UPGMA can be helpful for sequences from the same species or very closely related species

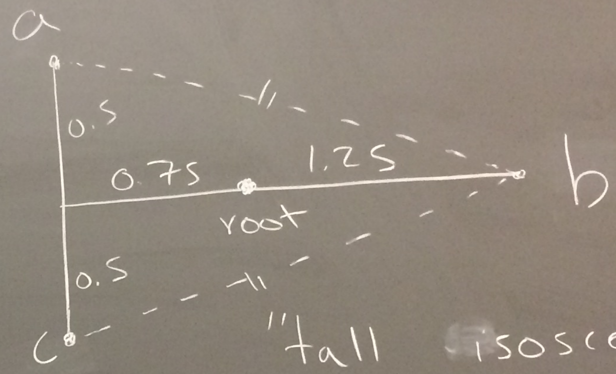
Another way to think about ultrametrics

ultrametric

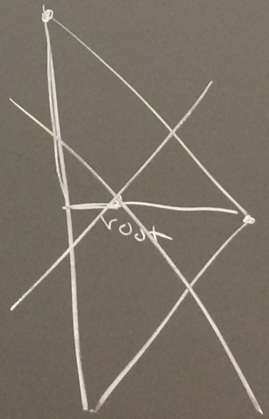
$$\delta(a,b) \le \max\{\delta(a,c), \delta(b,c)\}$$



$\rightarrow$



"tall isosceles triangle inequality"



UPGMA

UPGMA initialization:

1. Each sample $x \in \mathcal{X}$ starts in it's own cluster $C_x = \{x\}$
2. Set cluster distances $\Delta(C_i, C_j) = \delta(i, j)$ for all i, j

UPGMA update:

1. Find C_i and C_j (where $i \neq j$) that minimize $\Delta(C_i, C_j)$, and merge to create $C_{ij} = C_i \cup C_j$
2. Set the distances from C_{ij} to every other cluster C_k using the update rule:

$$\Delta(C_i \cup C_j, C_k) = \frac{|C_i|}{|C_i| + |C_j|}\Delta(C_i, C_k) + \frac{|C_j|}{|C_i| + |C_j|}\Delta(C_j, C_k)$$

3. Join C_i and C_j with interior vertex v; set the height of v equal to $\Delta(C_i, C_j)/2$

Neighbor-Joining (NJ)

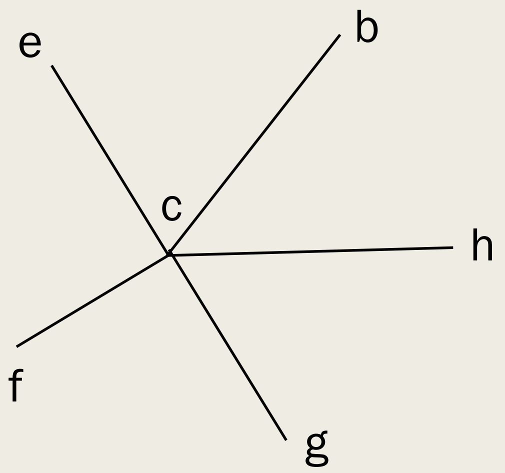
NJ initialization

Input

We are given a set of samples $\mathcal{X}$ and a dissimilarity map δ on $\mathcal{X}$.

Initialization

- Create a star tree with center vertex c and an edge (c, u) between c and all samples $u \in \mathcal{X}$.
- Let N_c be the set of neighbors of c and $n = |N_c|$ (cardinality of N_c). Set d equal to δ.



$N_c = \{b, e, f, g, h\}, \quad |N_c| = 5$

NJ Iterative step (part a)

(a) Find vertices f, g that minimize the Q-criteria. Note that UPGMA would only use the first term in this formula, $d(i, j)$. The remaining terms represent how far i and j are from the other vertices.

$$Q(i,j) = (n-2) \cdot d(i,j) - S_i - S_j, \quad \text{where}$$

$$S_i = \sum_{k \in N_c} d(i,k)$$

NJ Iterative step (part a)

(a) Find vertices f, g that minimize the Q-criteria. Note that UPGMA would only use the first term in this formula, $d(i,j)$. The remaining terms represent how far i and j are from the other vertices.

$$Q(i,j) = (n-2) \cdot \boxed{d(i,j)} - S_i - S_j, \quad \text{where}$$

$$S_i = \sum_{k \in N_c} d(i,k)$$

UPGMA

NJ Iterative step (part a)

(a) Find vertices f, g that minimize the Q-criteria. Note that UPGMA would only use the first term in this formula, $d(i,j)$. The remaining terms represent how far i and j are from the other vertices.

$$Q(i,j) = (n-2) \cdot \boxed{d(i,j)} \boxed{- S_i - S_j,} \quad \text{where}$$

$$S_i = \sum_{k \in N_c} d(i,k)$$

UPGMA

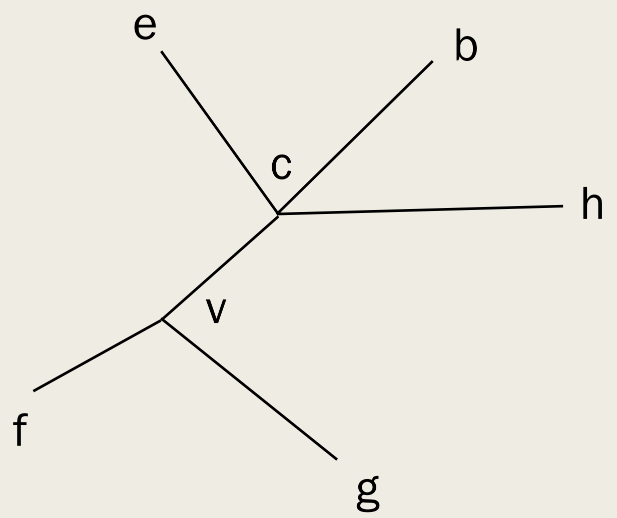
How far away *i* and *j* are from all the other vertices (further away means we'll join them earlier)

NJ Iterative step (part b)

(b) Join f and g at internal vertex v. Now N_c contains v but not f and g. Compute the new edges weights:

$$d(f,v) = \frac{1}{2}d(f,g) + \frac{1}{2(n-2)}[S_f - S_g]$$

$$d(g,v) = \frac{1}{2}d(f,g) + \frac{1}{2(n-2)}[S_g - S_f]$$

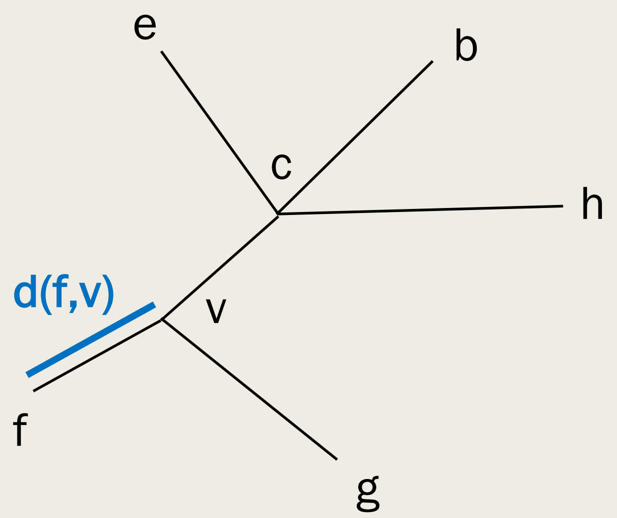


NJ Iterative step (part b)

(b) Join f and g at internal vertex v. Now N_c contains v but not f and g. Compute the new edges weights:

$$\boxed{d(f,v)} = \frac{1}{2}d(f,g) + \frac{1}{2(n-2)}[S_f - S_g]$$

$$d(g,v) = \frac{1}{2}d(f,g) + \frac{1}{2(n-2)}[S_g - S_f]$$

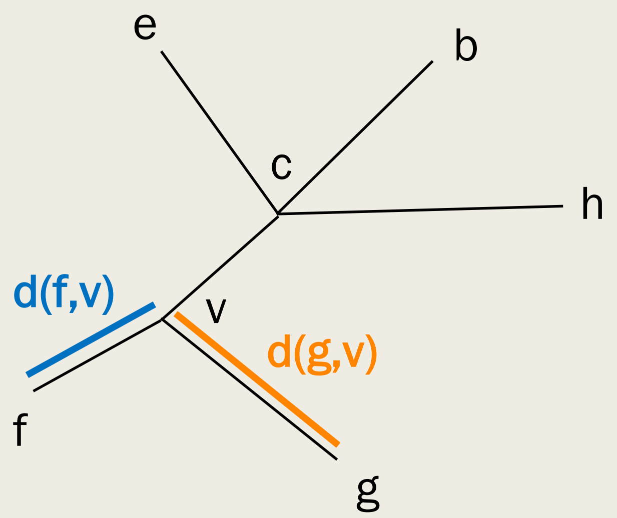


NJ Iterative step (part b)

(b) Join f and g at internal vertex v. Now N_c contains v but not f and g. Compute the new edges weights:

$$\boxed{d(f,v)} = \frac{1}{2}d(f,g) + \frac{1}{2(n-2)}[S_f - S_g]$$

$$\boxed{d(g,v)} = \frac{1}{2}d(f,g) + \frac{1}{2(n-2)}[S_g - S_f]$$



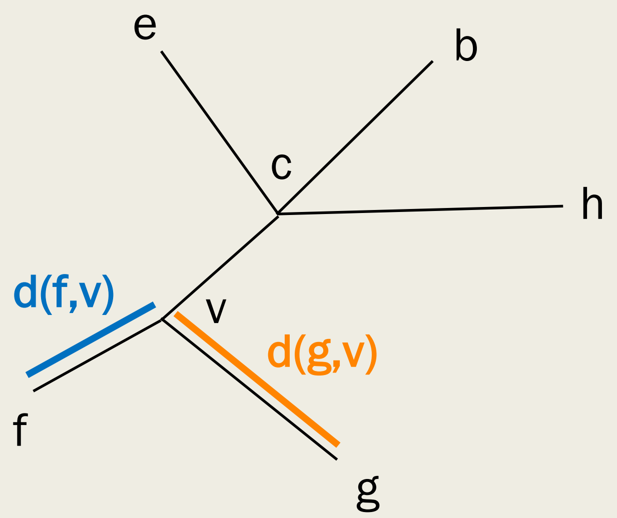
NJ Iterative step (part b)

UPGMA

(b) Join f and g at internal vertex v. Now N_c contains v but not f and g. Compute the new edges weights:

$$\boxed{d(f,v)} = \boxed{\frac{1}{2}d(f,g)} + \frac{1}{2(n-2)}[S_f - S_g]$$

$$\boxed{d(g,v)} = \boxed{\frac{1}{2}d(f,g)} + \frac{1}{2(n-2)}[S_g - S_f]$$



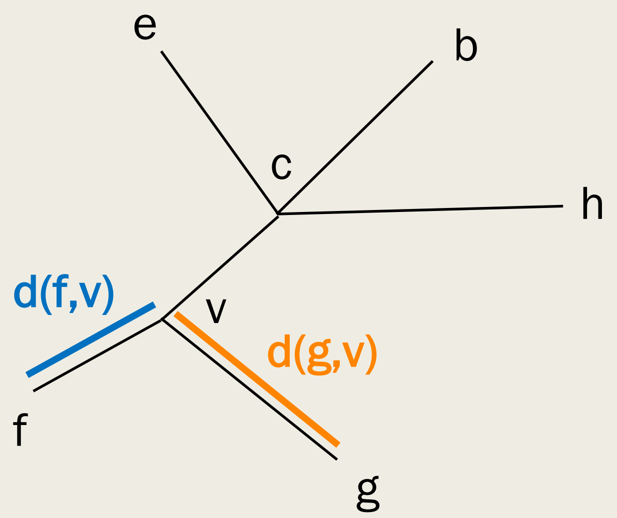
NJ Iterative step (part b)

UPGMA

(b) Join f and g at internal vertex v. Now N_c contains v but not f and g. Compute the new edges weights:

$$\boxed{d(f,v)} = \boxed{\frac{1}{2}d(f,g)} + \frac{1}{2(n-2)}\boxed{[S_f - S_g]}$$

$$\boxed{d(g,v)} = \boxed{\frac{1}{2}d(f,g)} + \frac{1}{2(n-2)}\boxed{[S_g - S_f]}$$



The *difference* between how far *f* and *g* are from other vertices. In this example *g* is on average further from other vertices, so d(g,v) > d(f,v)

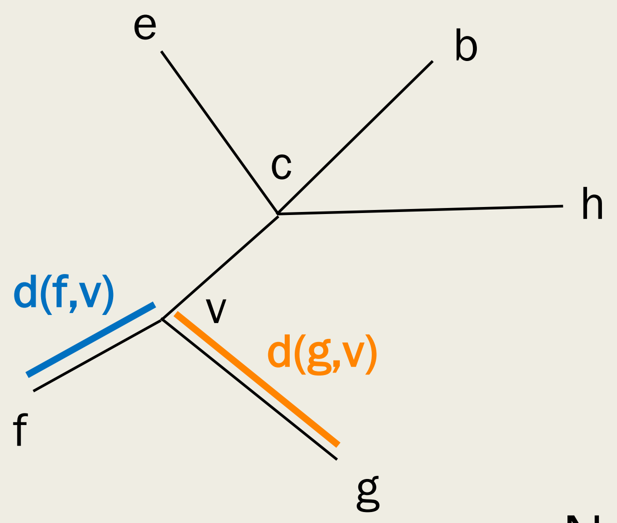
NJ Iterative step (part b)

UPGMA

(b) Join f and g at internal vertex v. Now N_c contains v but not f and g. Compute the new edges weights:

$$\boxed{d(f,v)} = \boxed{\frac{1}{2}d(f,g)} + \frac{1}{2(n-2)}\boxed{[S_f - S_g]}$$

$$\boxed{d(g,v)} = \boxed{\frac{1}{2}d(f,g)} + \frac{1}{2(n-2)}\boxed{[S_g - S_f]}$$



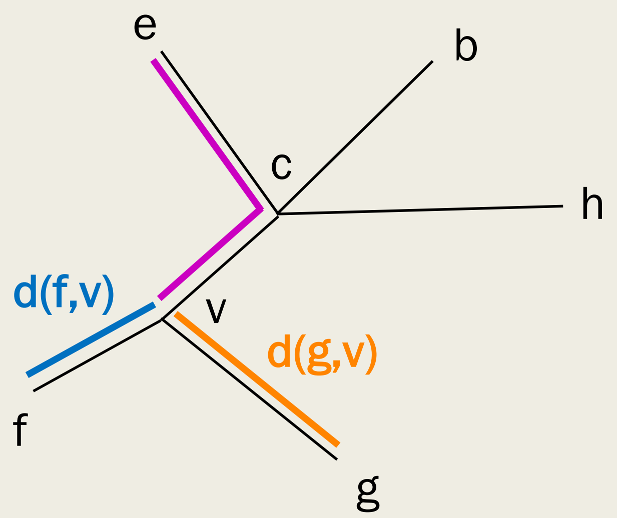
The *difference* between how far *f* and *g* are from other vertices. In this example *g* is on average further from other vertices, so d(g,v) > d(f,v)

$N_c = \{b, e, h, v\}, \quad |N_c| = 4$

NJ Iterative step (part c)

(c) Compute the distances from v to all remaining vertices $i \in N_c$:

$$\boxed{d(i,v)} = \frac{1}{2}[d(f,i) - d(f,v)] + \frac{1}{2}[d(g,i) - d(g,v)]$$



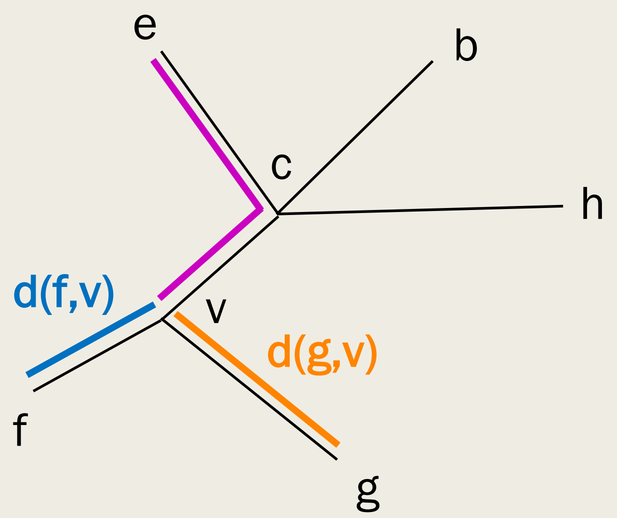
NJ Iterative step (part c)

(c) Compute the distances from v to all remaining vertices $i \in N_c$:

$$\boxed{d(i,v)} = \frac{1}{2}[d(f,i) - d(f,v)] + \frac{1}{2}[d(g,i) - d(g,v)]$$

Another way to write this:

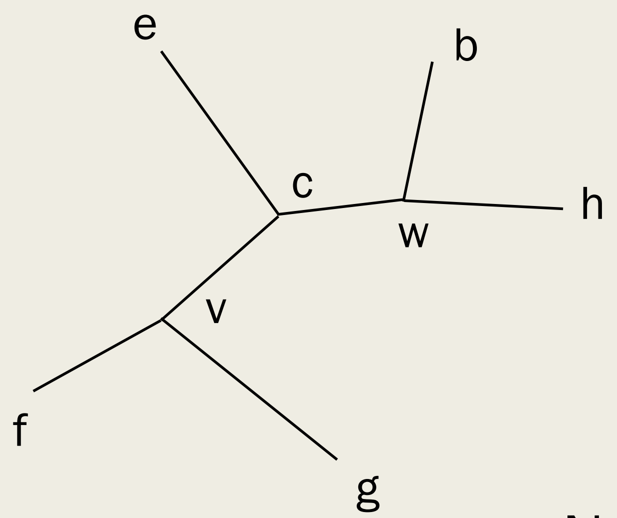
$$d(i,v) = \tfrac{1}{2}[d(f,i) + d(g,i) - d(f,g)]$$



NJ Termination

Termination

When $n = 3$, the tree topology does not change since we have obtained a binary tree. We still need to run the last iteration though to determine the 3 remaining edge weights. The output is then the tree topology and all edge weights.

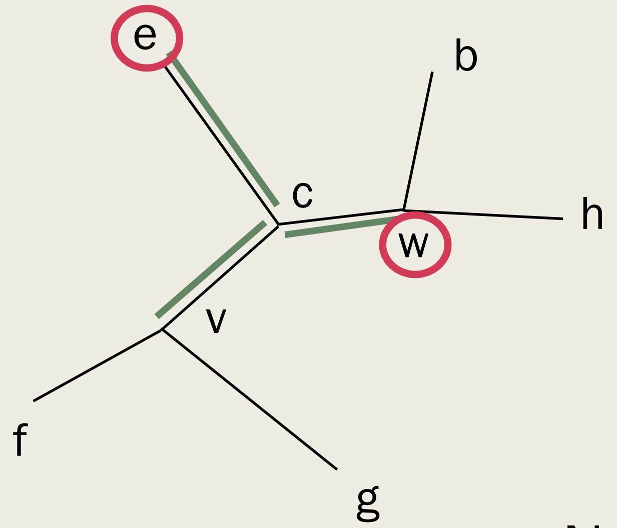


$N_c = \{e, v, w\}, \quad |N_c| = 3$

NJ Termination

Termination

When $n = 3$, the tree topology does not change since we have obtained a binary tree. We still need to run the last iteration though to determine the 3 remaining edge weights. The output is then the tree topology and all edge weights.



We could "merge" *e* and *w* at *c*, then we would find d(e,c) and d(w,c) in step (b) and find d(v,c) in step (c)

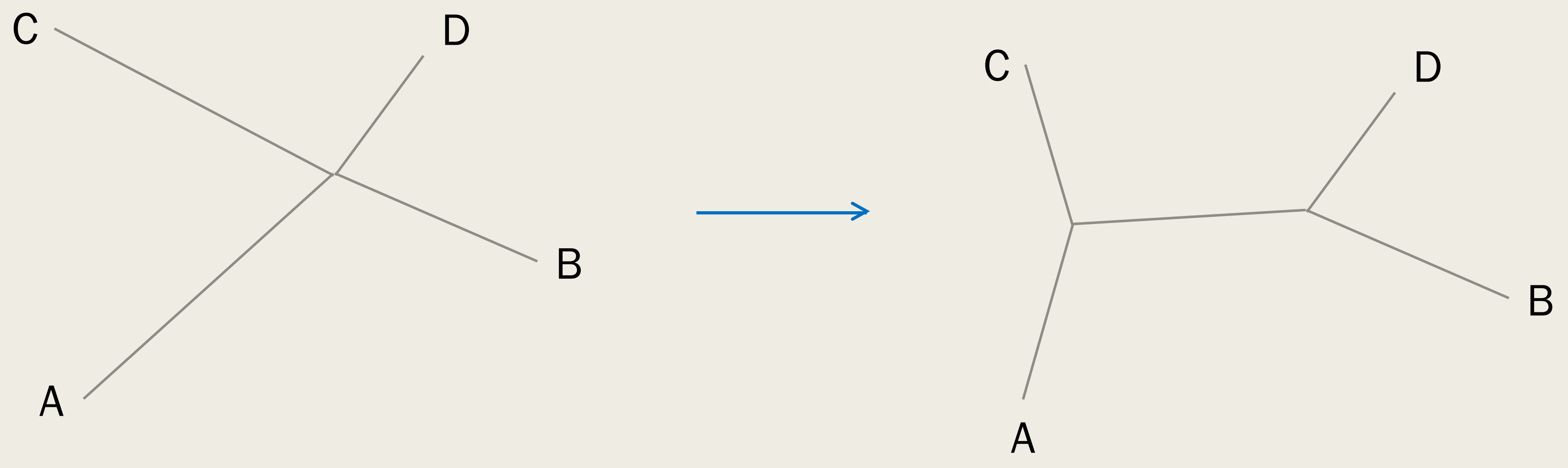
$N_c = \{e, v, w\}, \quad |N_c| = 3$

Q-criteria intuition

- Goal: we want the smallest tree that adequately explains the observed patterns of evolution (called BME: Balanced Minimum Evolution)

- Q-criteria minimizes the “whole tree length”, which is the average of all the different ways we could walk around the tree

- The idea is that we want to merge nodes that are far away, so we don’t have to “walk” to each of them separately, we can use the path to their merged vertex

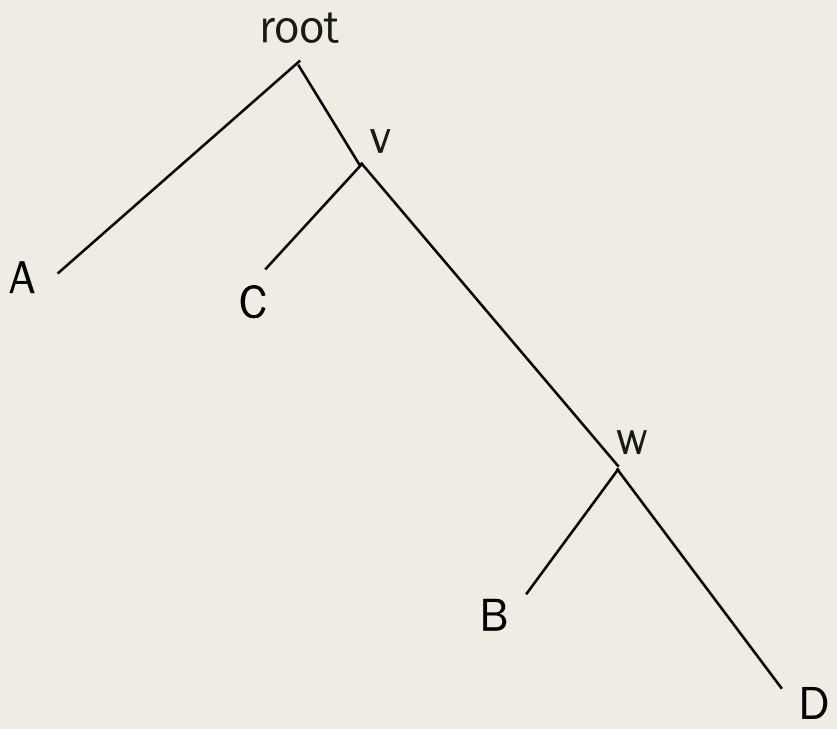


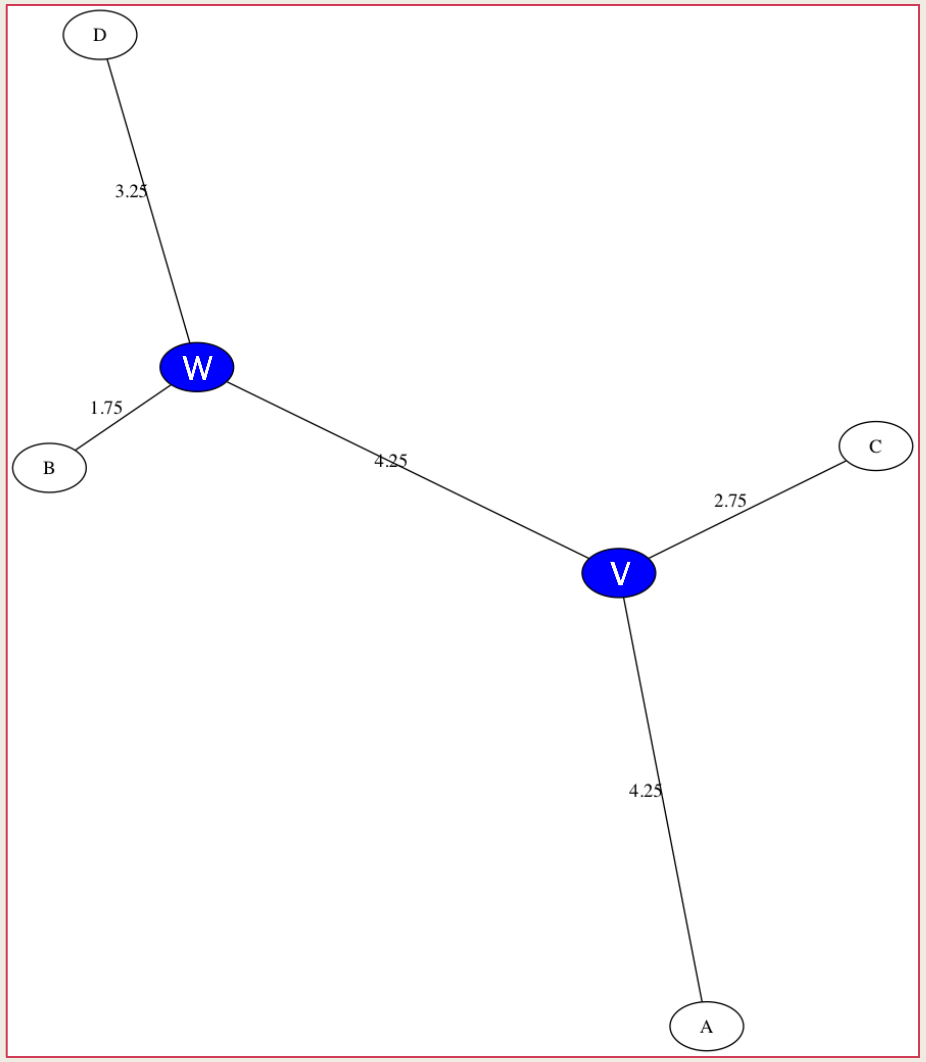
How to root a NJ tree?

- Method 1: use an ***outgroup***
- An outgroup is a species or sample that is more distantly related to all the other samples ("ingroup") than any pair of ingroup samples

How to root a NJ tree?

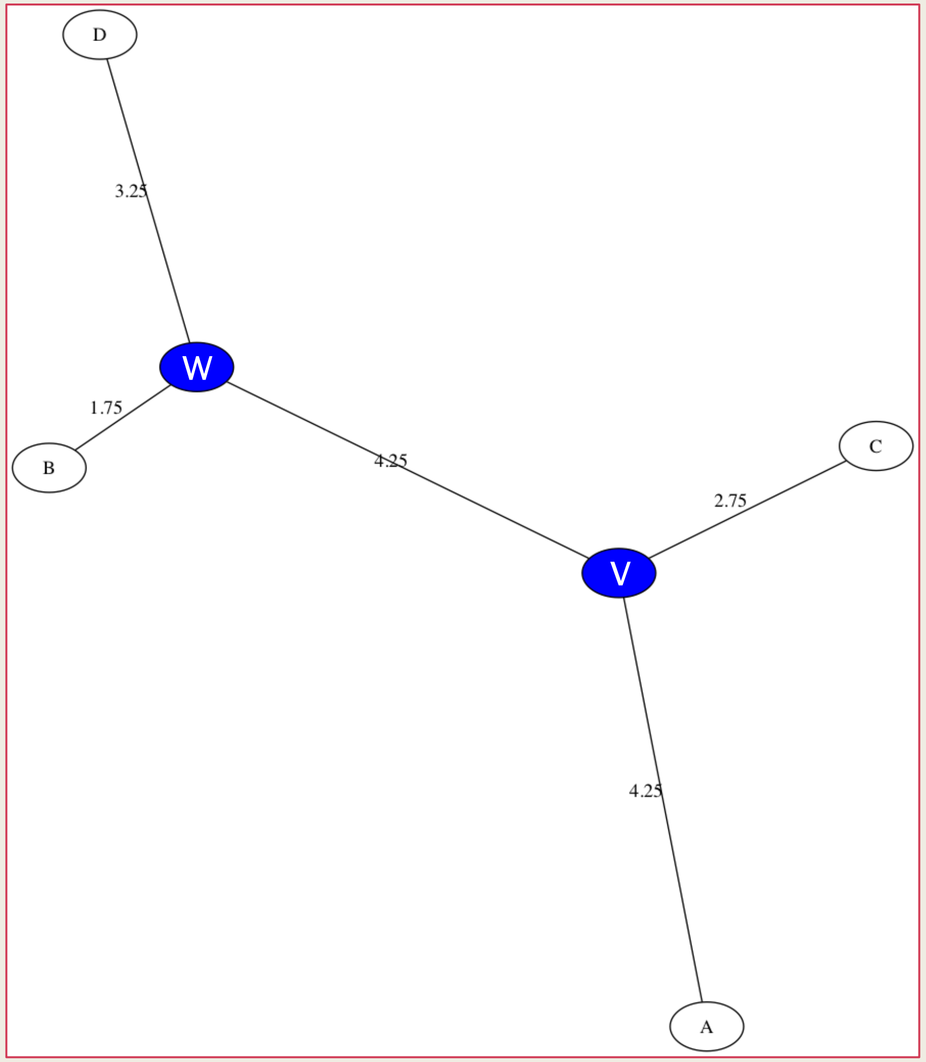
- Method 1: use an ***outgroup***
- An outgroup is a species or sample that is more distantly related to all the other samples ("ingroup") than any pair of ingroup samples
- For example, if we knew that A is an outgroup to ingroup {B,C,D}, we could root the NJ like this:





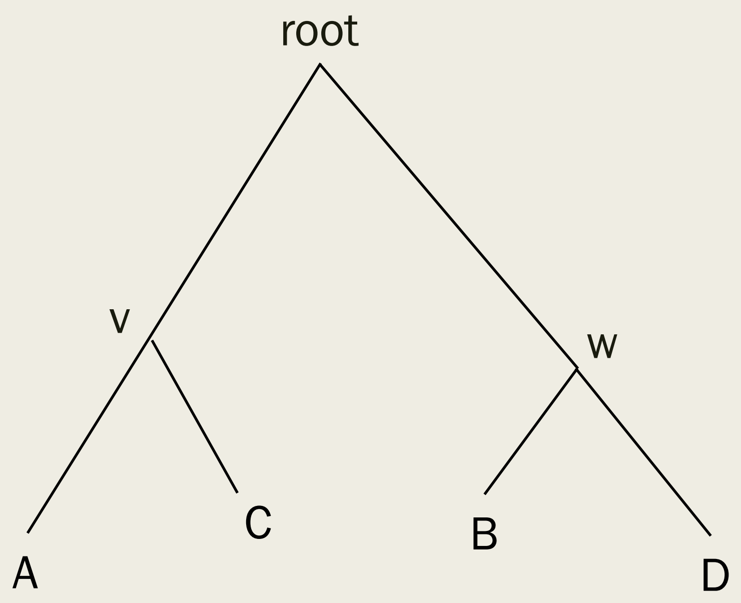
How to root a NJ tree?

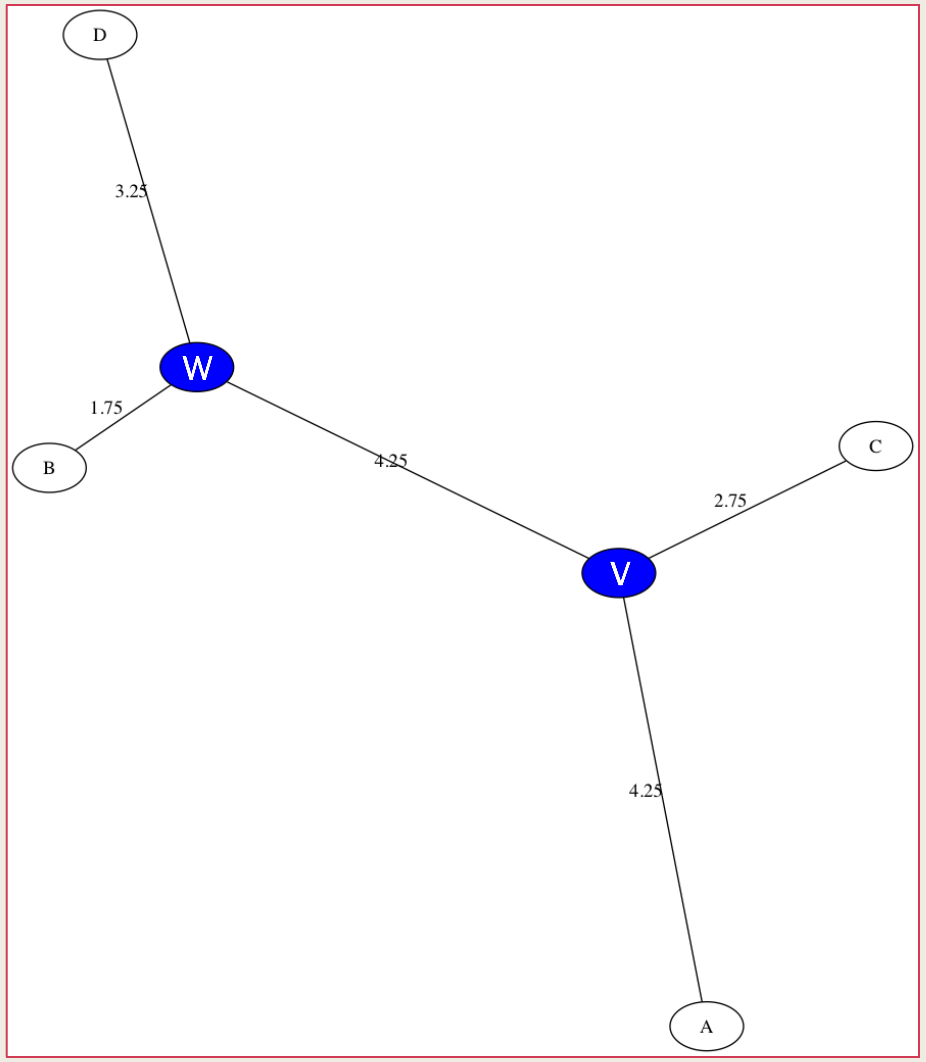
- Method 2: divide the longest path between leaves by 2
- *Assumption: molecular clock more or less valid*



How to root a NJ tree?

- Method 2: divide the longest path between leaves by 2
- *Assumption: molecular clock more or less valid*
- Longest path:
- A -> v -> w -> D = 4.25+4.25+3.25 = 11.75





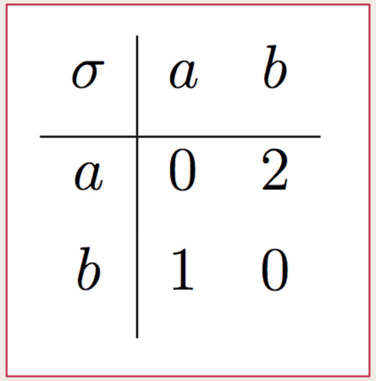
Ancestral State Reconstruction

Ancestral state reconstruction via parsimony

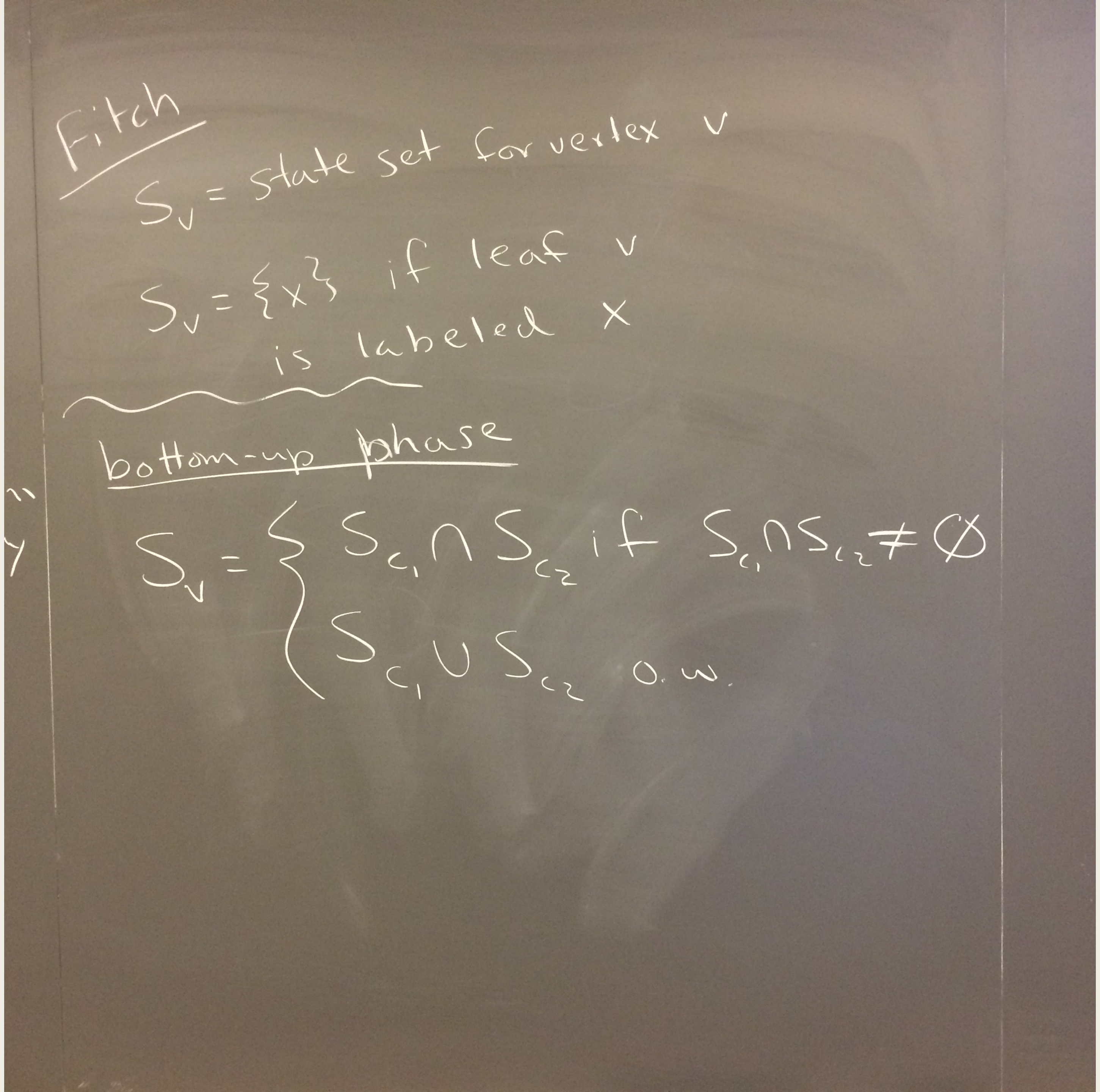
- **Input:** rooted, binary phylogenetic tree and leave labels

- **Output:** internal vertex labels that minimize the parsimony score (weighted or unweighted)

- For Sankoff we need a mutational scoring matrix (example with characters *a*,*b*), which does not have to be symmetric. Row is the “before” state, column is the “after” state.



σ	a	b
a	0	2
b	1	0



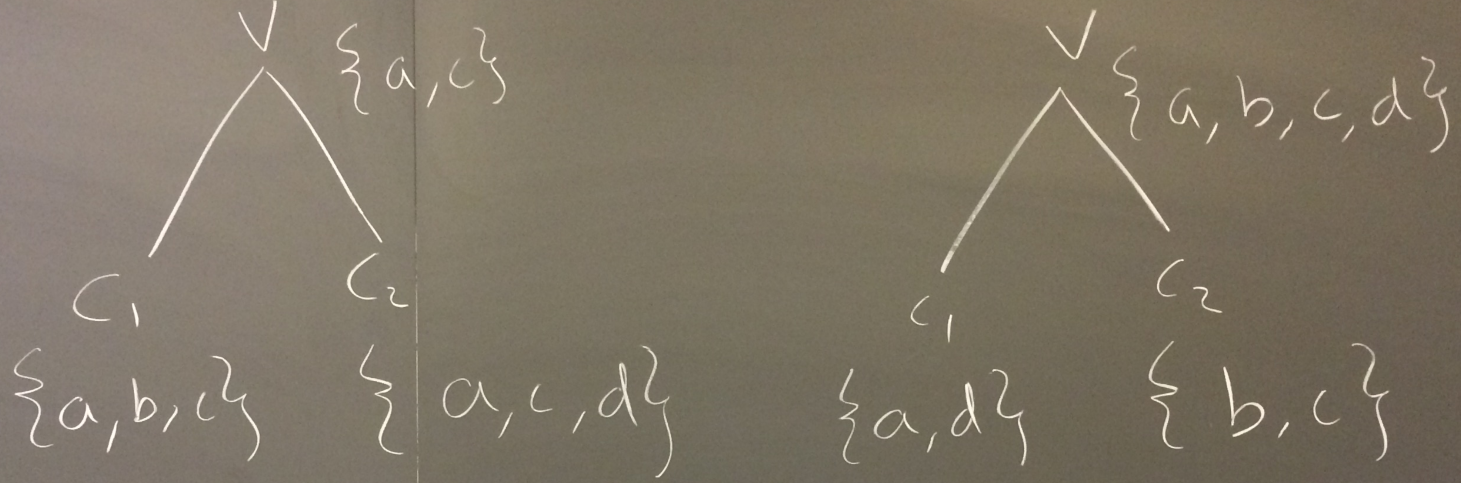
Fitch

S_v = state set for vertex v

$S_v = \{x\}$ if leaf v is labeled x

bottom-up ~~p~~phase

$$S_v = \begin{cases} S_{c_1} \cap S_{c_2} & \text{if } S_{c_1} \cap S_{c_2} \neq \emptyset \\ S_{c_1} \cup S_{c_2} & \text{o.w.} \end{cases}$$



$$\boxed{e_k(0) = \frac{E_k(0)}{E_k(0)+E_k(1)}}$$

$$1 - e_k(0) = e_k(1)$$

Recap Fitch

Recap Sankoff's algorithm

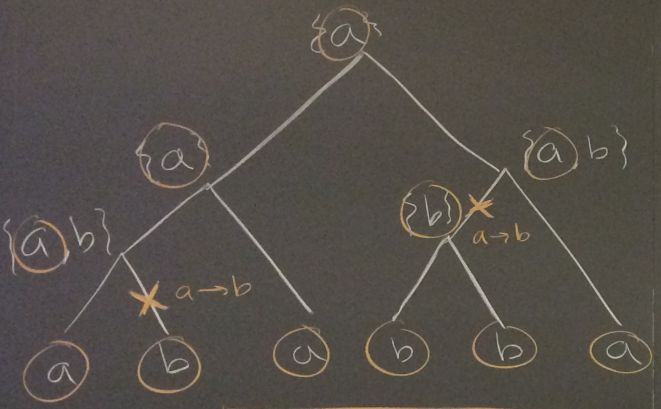
Initialization: Let $A_v(x)$ be the minimum parsimony score of assigning character x to vertex v. To begin $A_{\text{leaf}}(x) = 0$ if the leaf is assigned character x, and ∞ otherwise. This prevents us from ever tracing back to a non-assigned leaf label.

Bottom-up recursive step: Let c_1 and c_2 be the two children of vertex v. For all x in our character state set, let

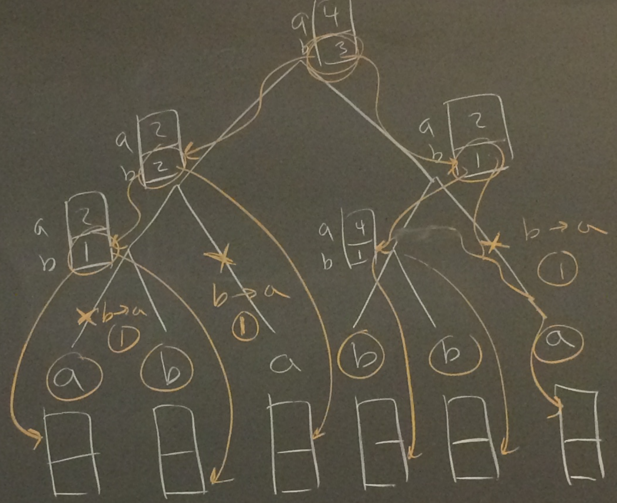
$$A_v(x) = \min_y \{A_{c_1}(y) + \sigma(x,y)\} + \min_z \{A_{c_2}(z) + \sigma(x,z)\}.$$

Keep track of a back-pointer to the minimum y and z.

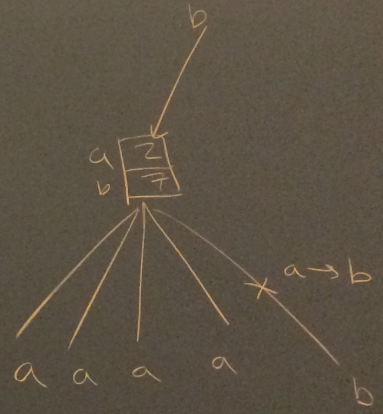
Top-down traceback: Choose root state x such that $A_{\text{root}}(x)$ is the minimum. Follow back-pointers to find the assigned state of every internal vertex.



2 mutation score



3 mutation score


b
a 2
b 7
a→b
a a a a
b