CS 364 COMPUTATIONAL BIOLOGY

Sara Mathieson
Haverford College

Outline

- Parsimony
- Fitch’s Algorithm
- Sankoff’s Algorithm

Parsimony Problem

Ancestral State Reconstruction

- Given:
  - States at the leaves
  - Tree topology (and sometimes branch lengths)

- Find the ancestral states at each internal vertex

Characters: Heritable traits

	Wings	Flying	Swimming	Cute
	Present	Absent	Absent	Present
	Present	Present	Absent	Absent
	Absent	Absent	Present	Absent
	Absent	Absent	Absent	Present

Images: BBC

Characters: Heritable traits

	Wings	Flying	Swimming	Cute
	1	0	0	1
	1	1	0	0
	0	0	1	0
	0	0	0	1

Images: BBC

Characters: DNA sequence

ACTGAGATAGGGAGGGCGGA

ACTCAGATAGGGAGAGGAGA

ACTGAGAGAGGGAGAGCAGA

ACTGAGAGAGGGAGAGGAGA

Characters: DNA sequence

0001001011111001001 0

00011010111111010010

00011010111010010010

000100101001-0010010

RNA, Protein sequence etc…

Evolution in Great Apes

List characteristics or traits

- Orange beard
- Miniaturized hair
- Way of movement
- …

Orangutan (Bornean)

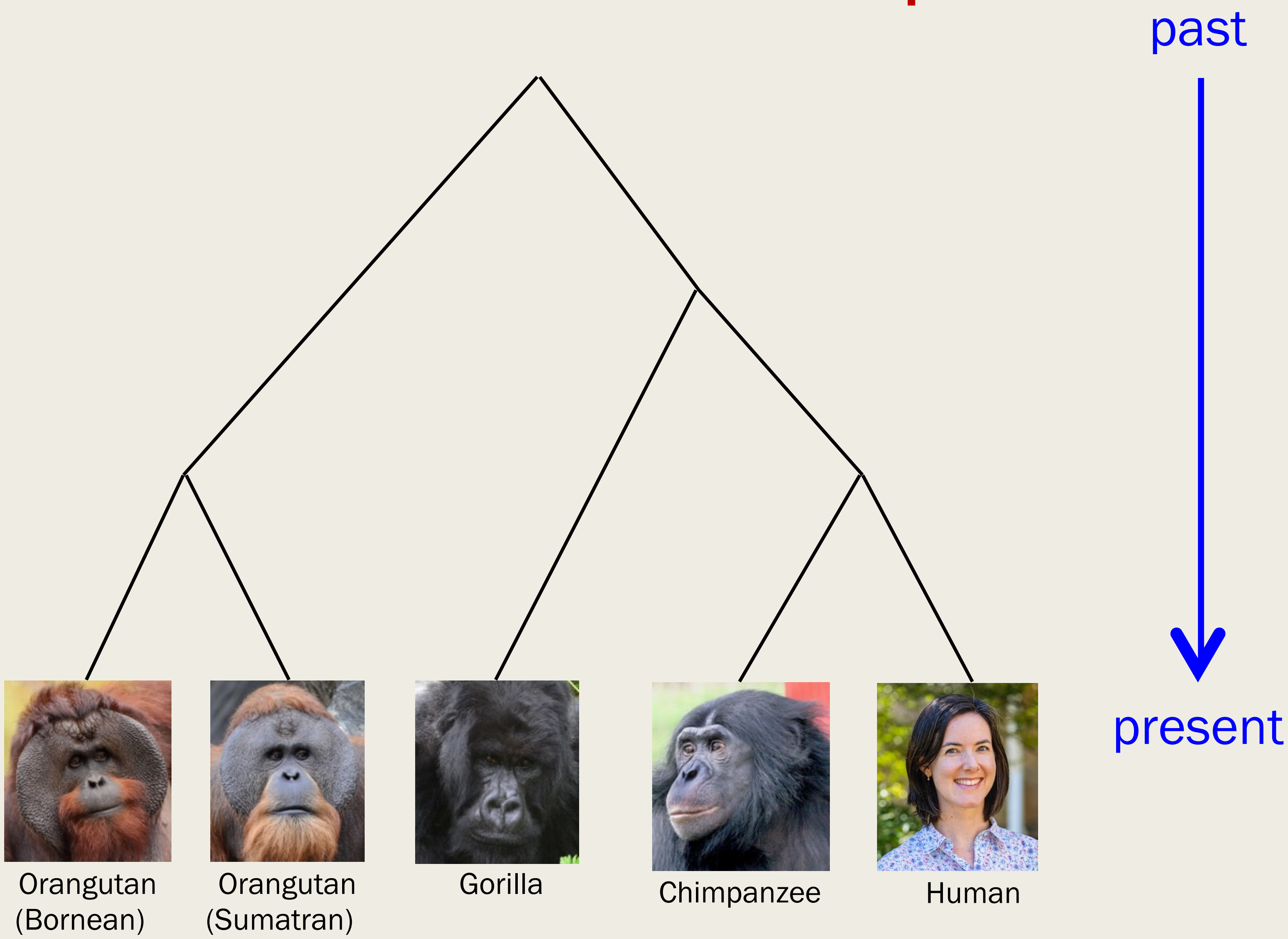
Orangutan (Sumatran)

Gorilla

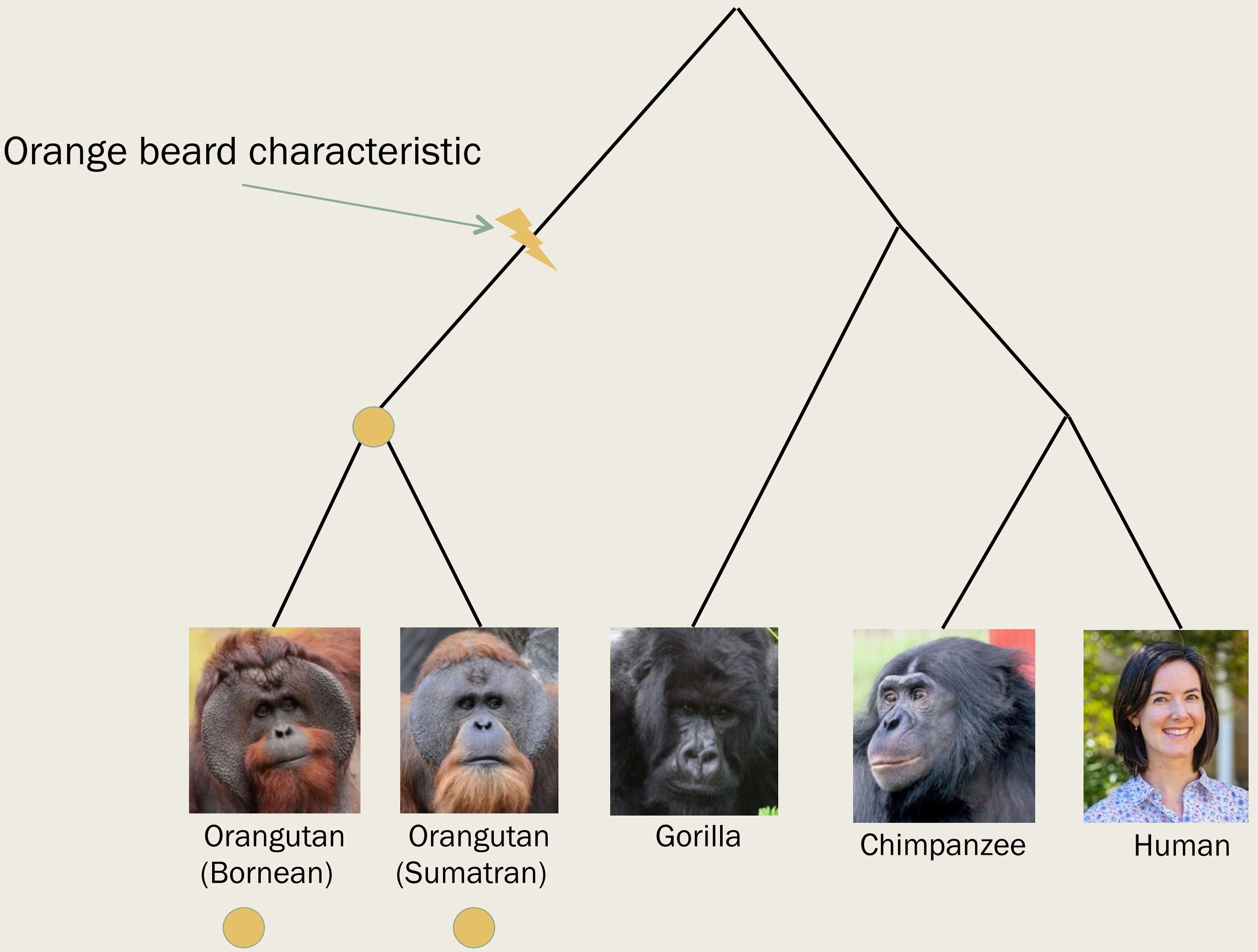
Chimpanzee

Human

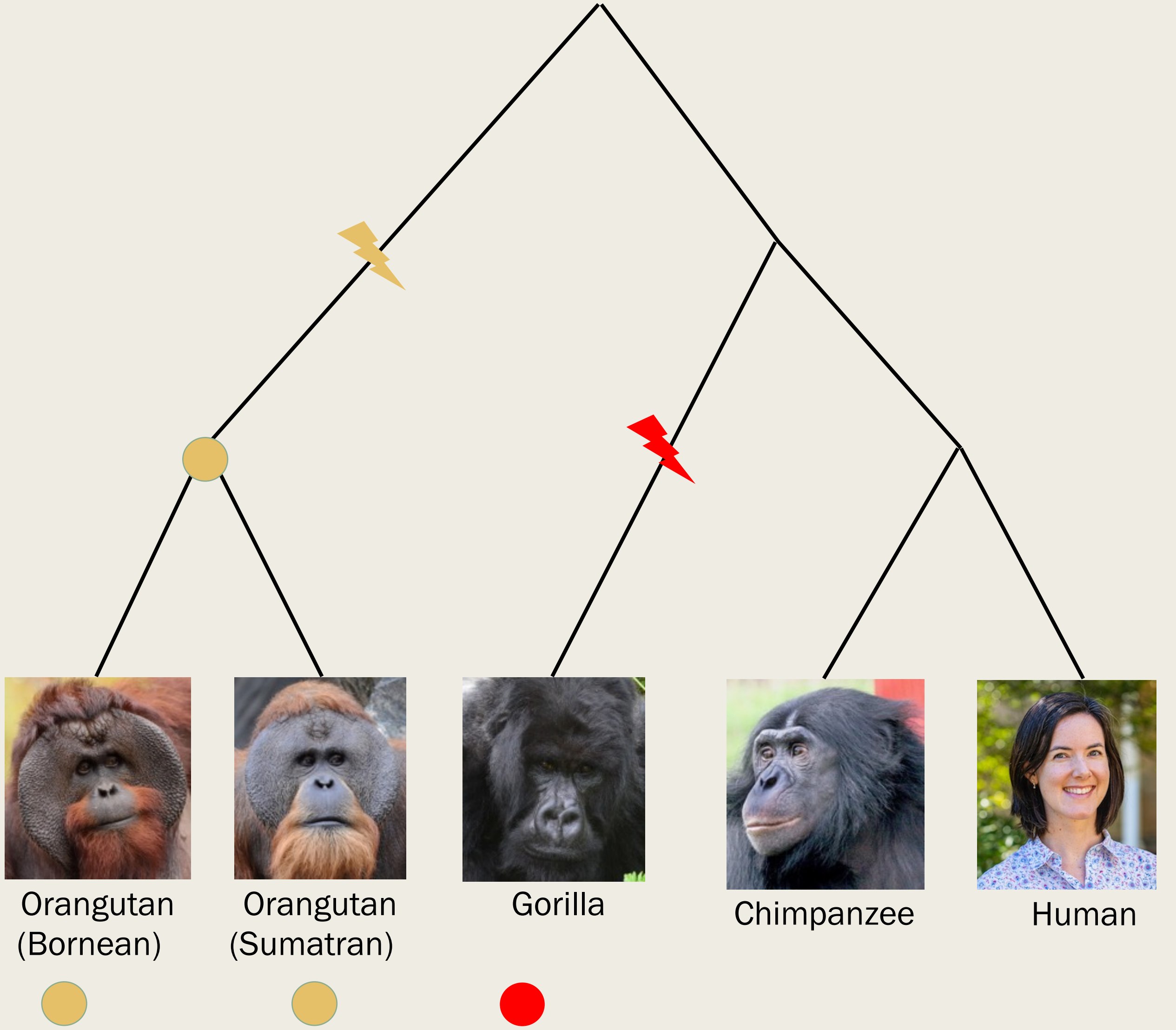
Evolution in Great Apes



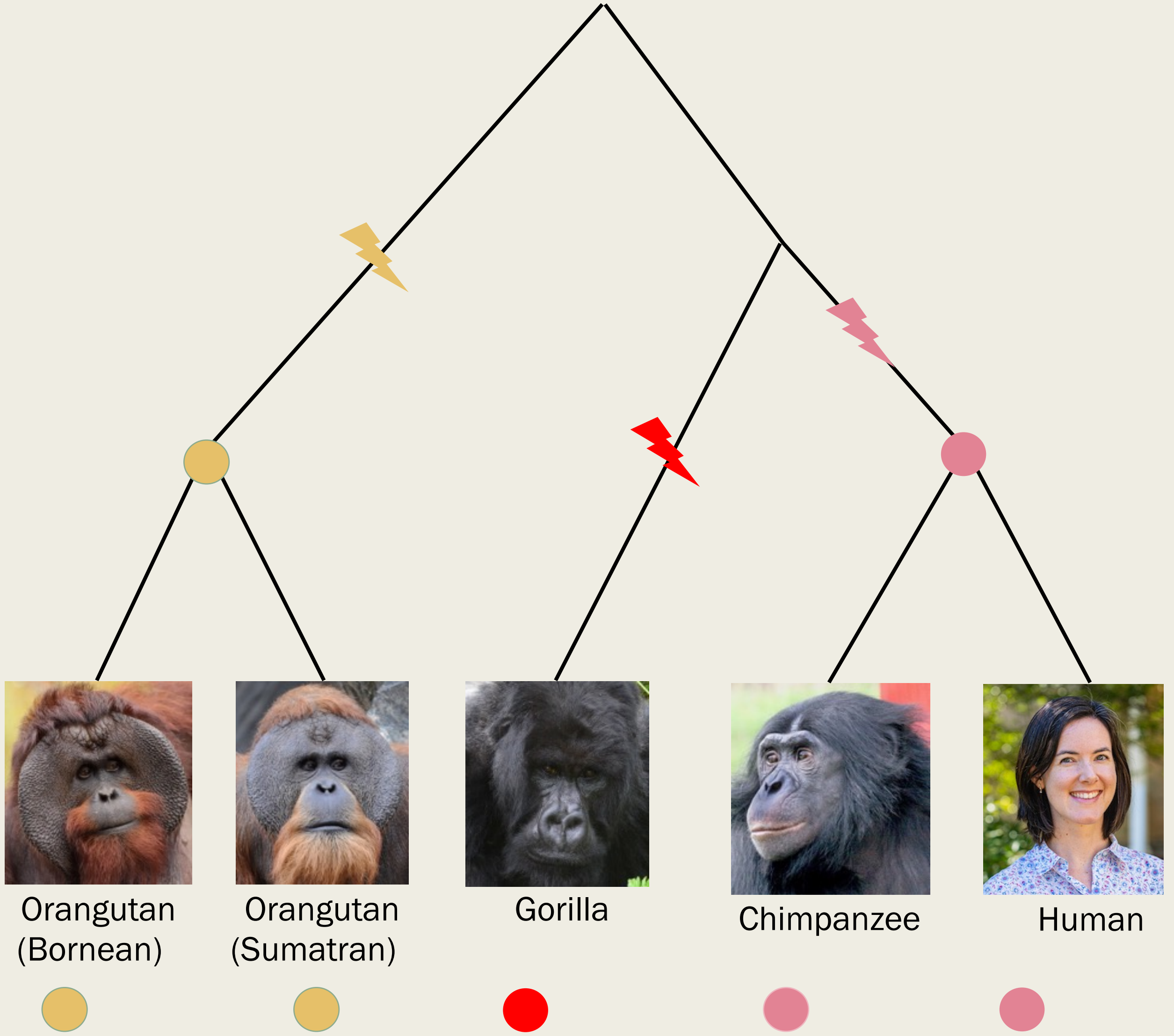
Evolution in Great Apes



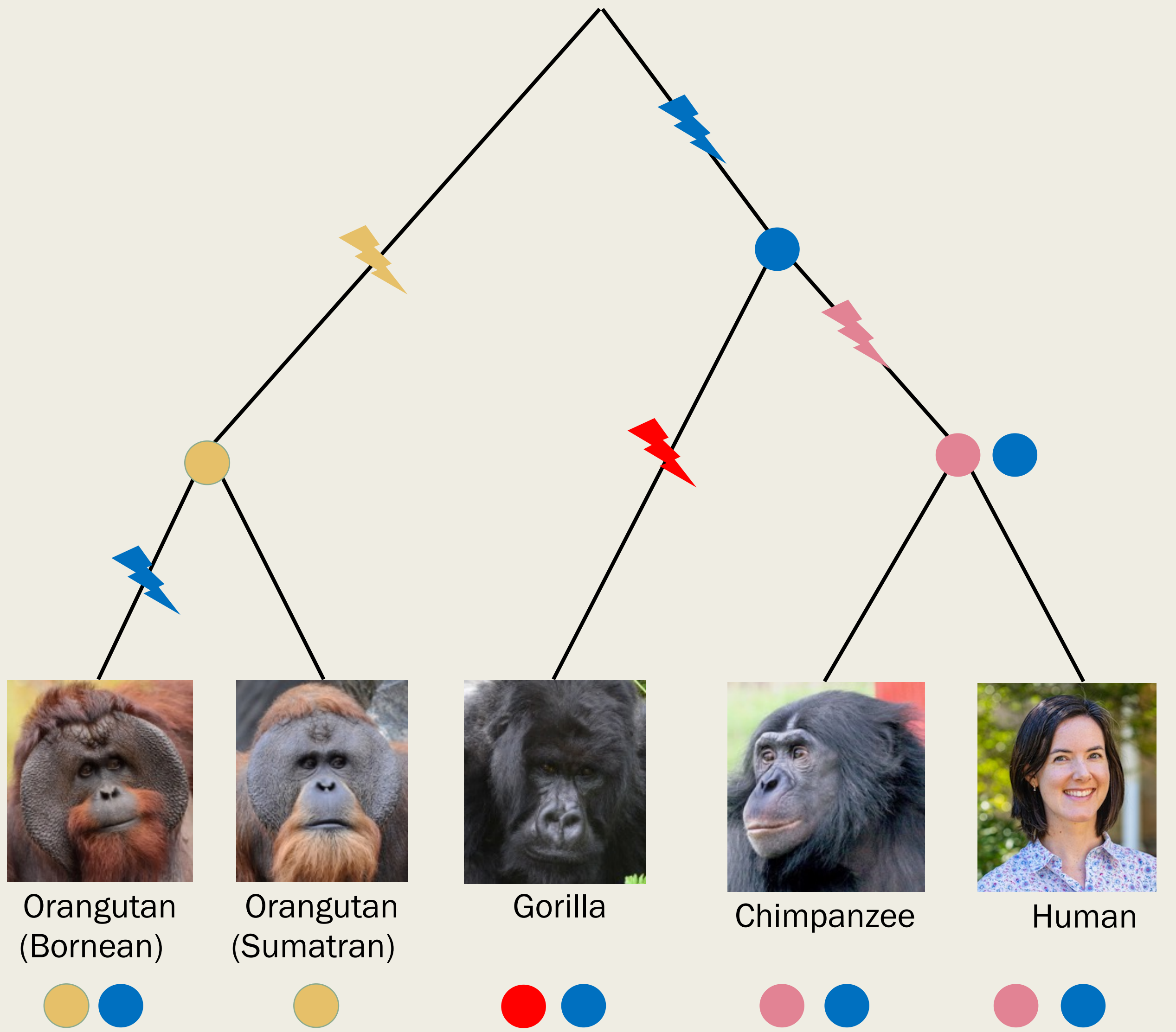
Evolution in Great Apes



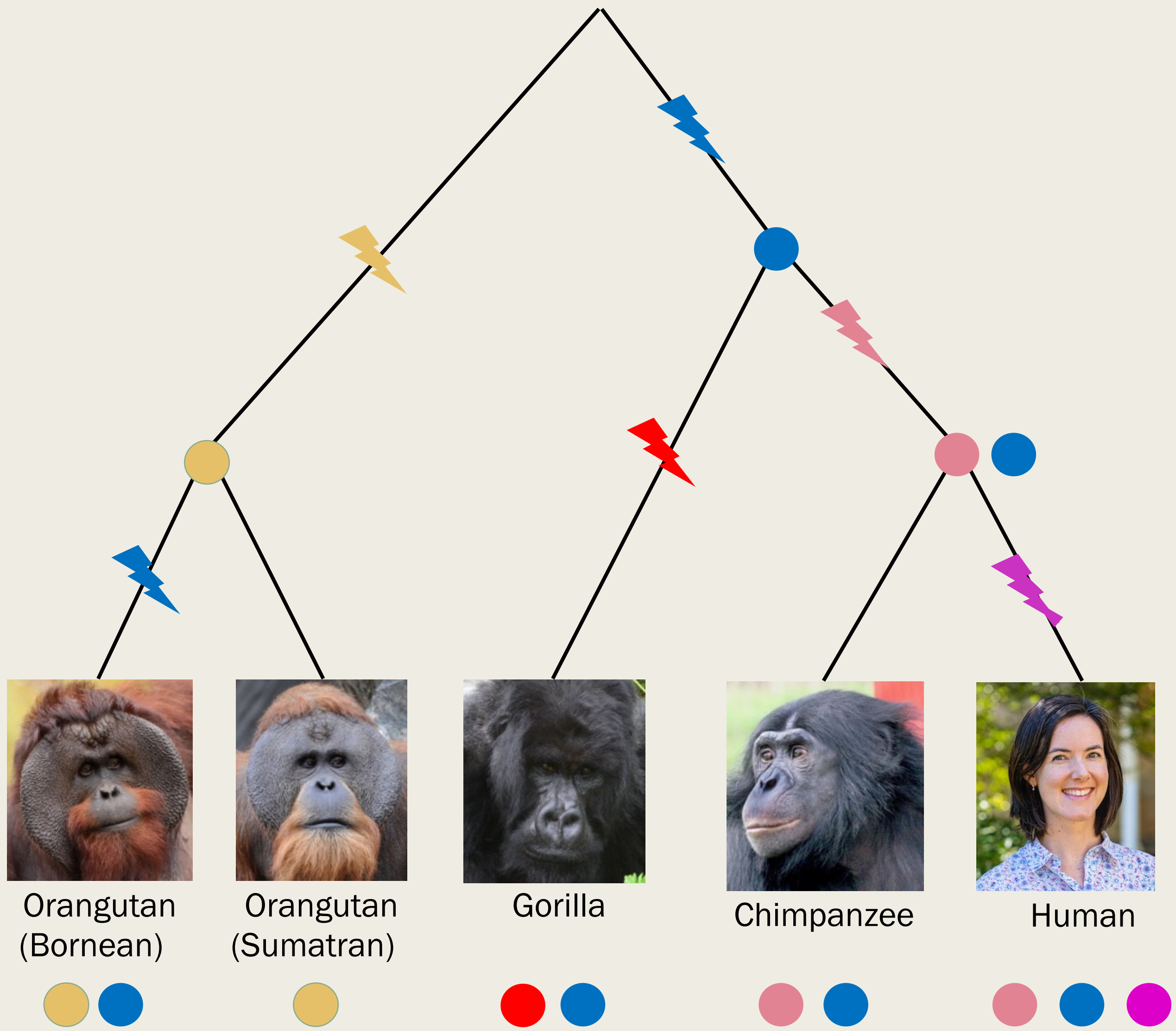
Evolution in Great Apes



Evolution in Great Apes



Evolution in Great Apes



Evolution in Great Apes

We do not observe the evolutionary (phylogenetic) tree.

Only the mutations/characters at the leaves

If we *knew* the shape of the tree, could we reconstruct the characters of the ancestors?

Orangutan (Bornean)

Orangutan (Sumatran)

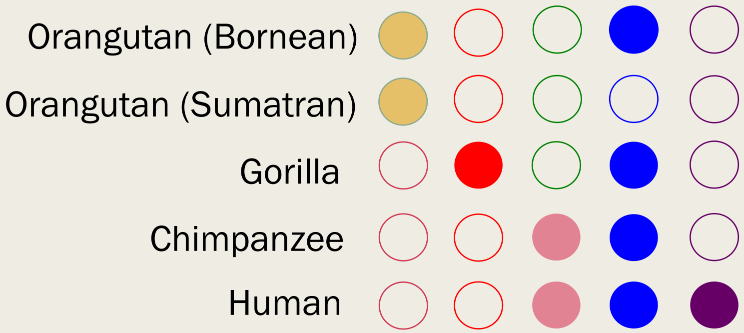
Gorilla

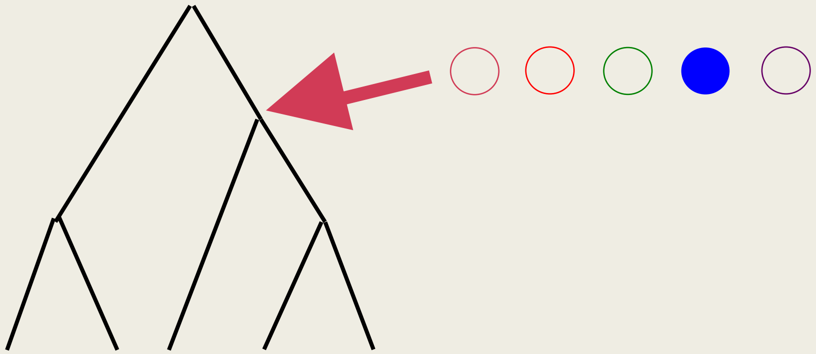
Chimpanzee

Human

GOAL

- <u>Input:</u>
  - *Presence/absence of characteristics (states) at leaves*
  - *Tree topology*



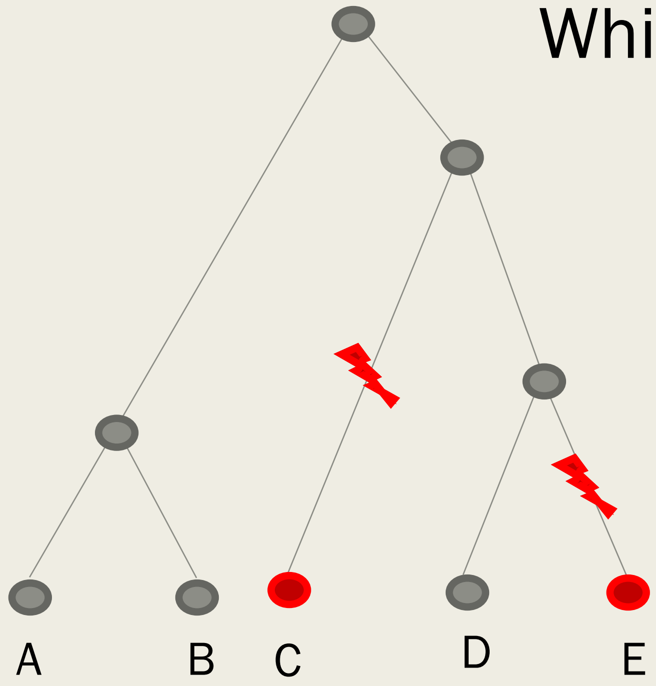


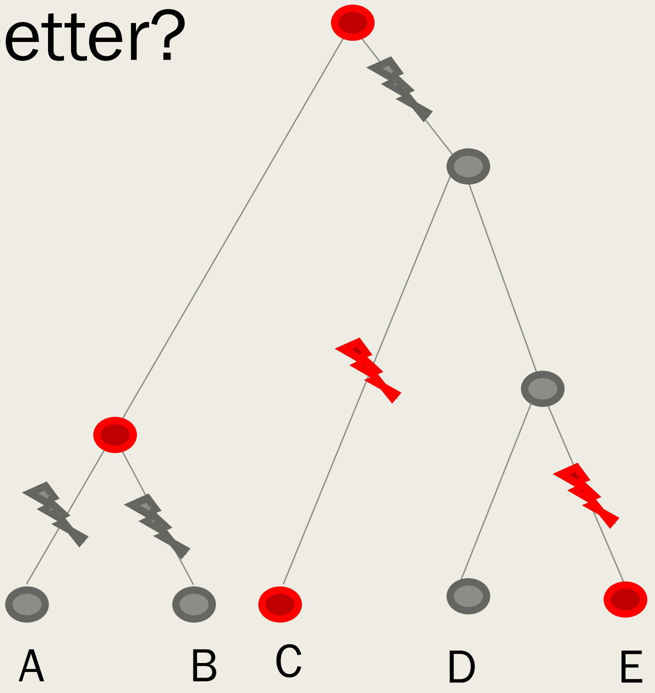
<u>Output:</u>

Ancestral states at each internal node (and root)

Parsimony

Which is better?





The parsimony principle: “Everything else being equal, the best explanation is the one that requires the fewest evolutionary changes”

Two parsimony problems

- The “Small” parsimony problem – given the character states at the leaves, and the topology of the tree, reconstruct the states at the ancestral nodes.

- The “Big” parsimony problem – given the character states at the leaves, reconstruct the tree, and the states at the ancestral nodes.

Two parsimony problems

- The “Small” parsimony problem – given the character states at the leaves, and the topology of the tree, reconstruct the states at the ancestral nodes.

  Fitch’s algorithm (Fitch, 1971)

  Sankoff’s algorithm (Sankoff, 1975)

The parsimony principle: “Everything else being equal, the best explanation is the one that requires the fewest evolutionary changes”

Fitch’s algorithm (unweighted parsimony)

Example of Fitch's algorithm in the literature

BIOINFORMATICS **ORIGINAL PAPER**

Vol. 29 no. 23 2013, pages 2987–2994
doi:10.1093/bioinformatics/btt527

Genome analysis Advance Access publication September 24, 2013

FPSAC: fast phylogenetic scaffolding of ancient contigs

Ashok Rajaraman[1,2], Eric Tannier[3,4] and Cedric Chauve[1,5,*]

[1]Department of Mathematics, Simon Fraser University, Burnaby (BC) V5A1S6, Canada, [2]International Graduate Training Center in Mathematical Biology, Pacific Institute for the Mathematical Sciences, Vancouver (BC), Canada, [3]INRIA Grenoble Rhône-Alpes, Montbonnot 38334, France, [4]Université de Lyon 1, Laboratoire de Biométrie et Biologie Évolutive, CNRS UMR5558 F-69622 Villeurbanne, France and [5]LaBRI, Université Bordeaux I, 33405 Talence, France

Associate Editor: Michael Brudno

Black death pandemic in Europe: 1347-1351
Killed 30-60% of Europe's population

In this paper, they used Fitch's algorithm to help reconstruct the genome of the ancient Black death pathogen

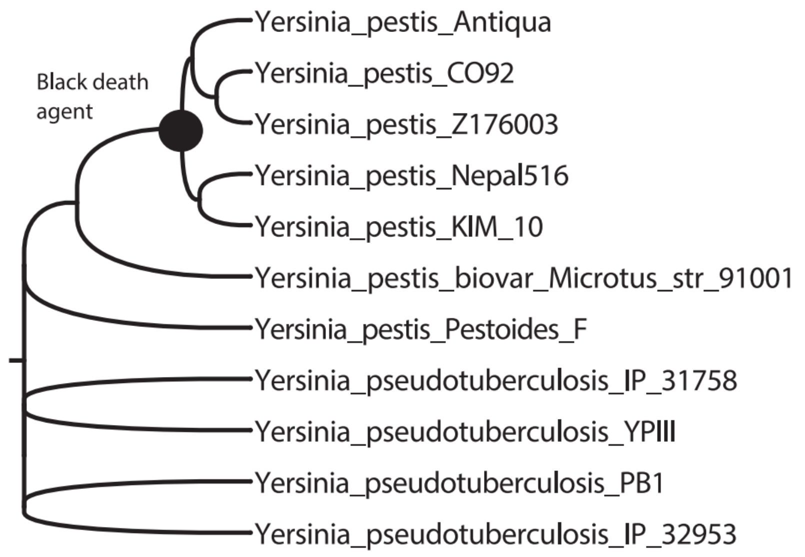
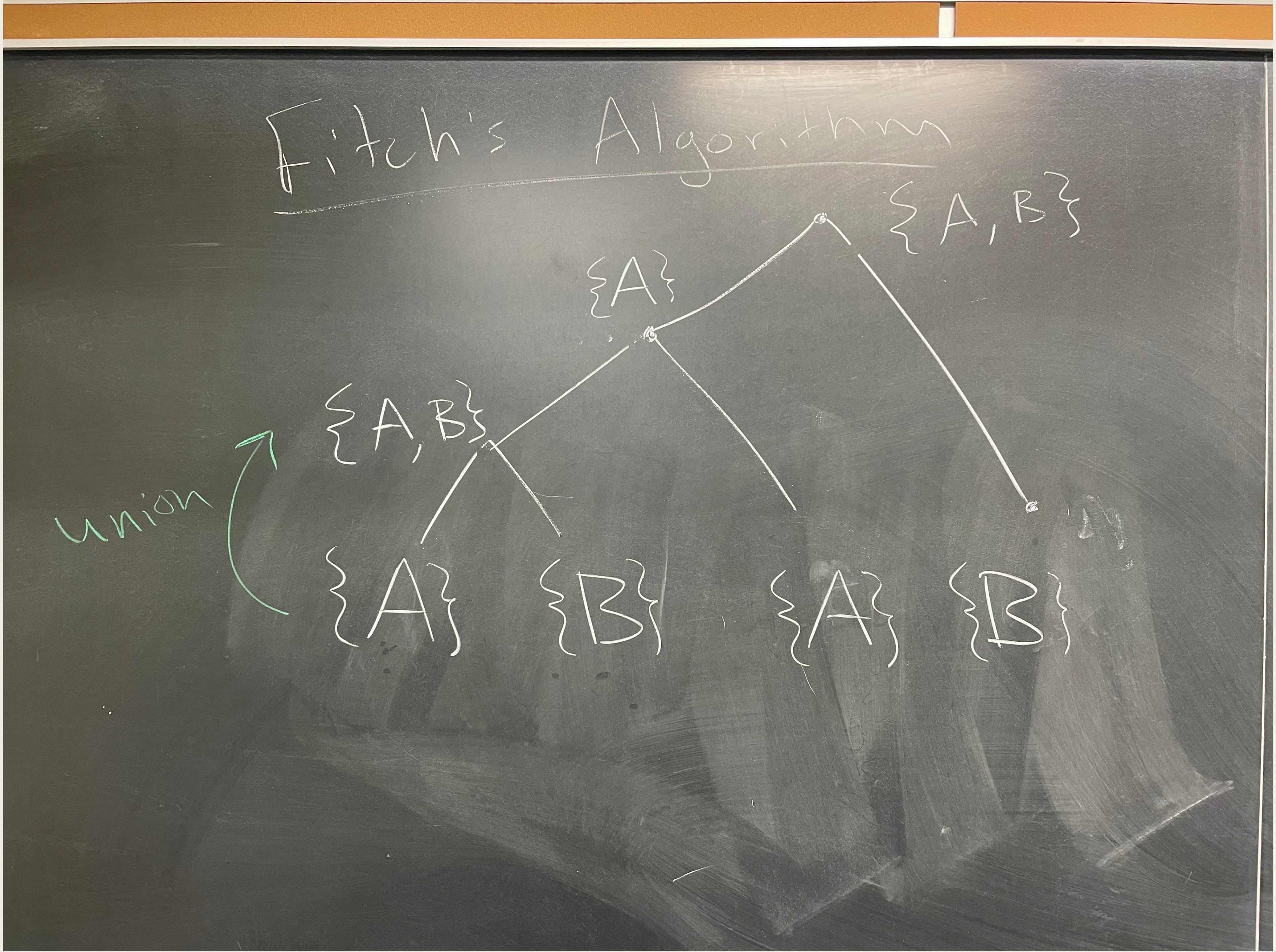


Fig. 3. Phylogeny of the considered genomes from Bos *et al.* (2011)


Fitch's Algorithm
{A, B}
{A}
{A,B}
union
{A}
{B}
{A}
{B}

bottom-up phase

S_v = state set at node v

- $S_v = \{x\}$ if v is a leaf assigned base x
- internal vertices:

let c_1 & c_2 = children of v

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

intersection

union

empty set

top-down phase

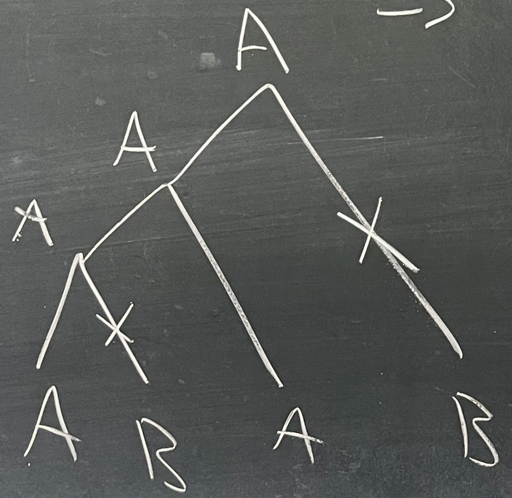
- assign root arbitrarily
- assign node c:

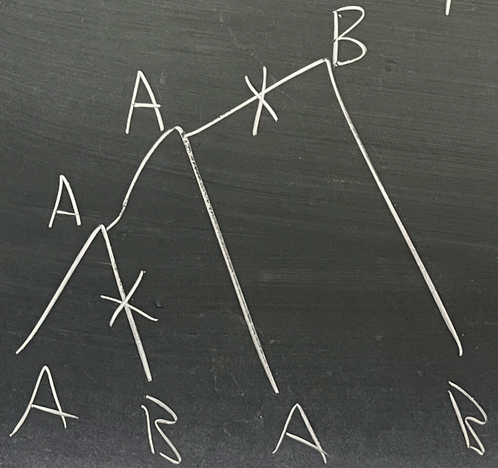
  if parent of c is x & $x \in S_c$

  $\Rightarrow$ then assign x to c

  otherwise

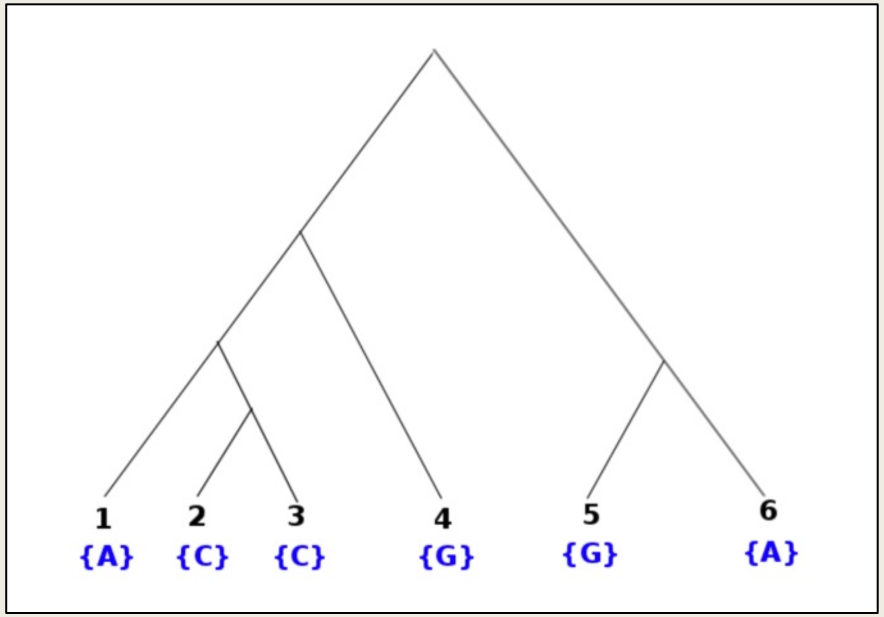
  $\Rightarrow$ choose arbitrarily.





$\boxed{\text{parsimony score} = 2}$

Fitch’s algorithm

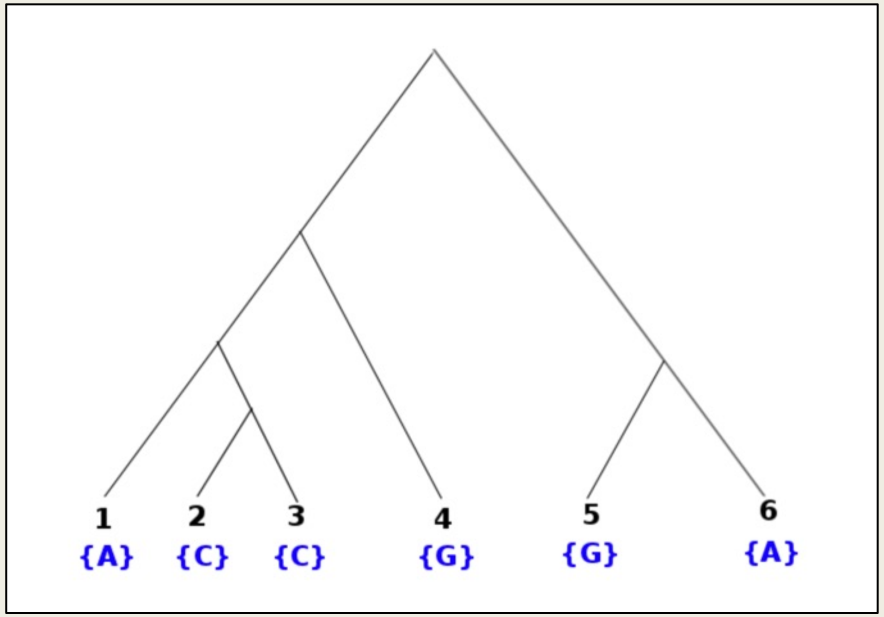


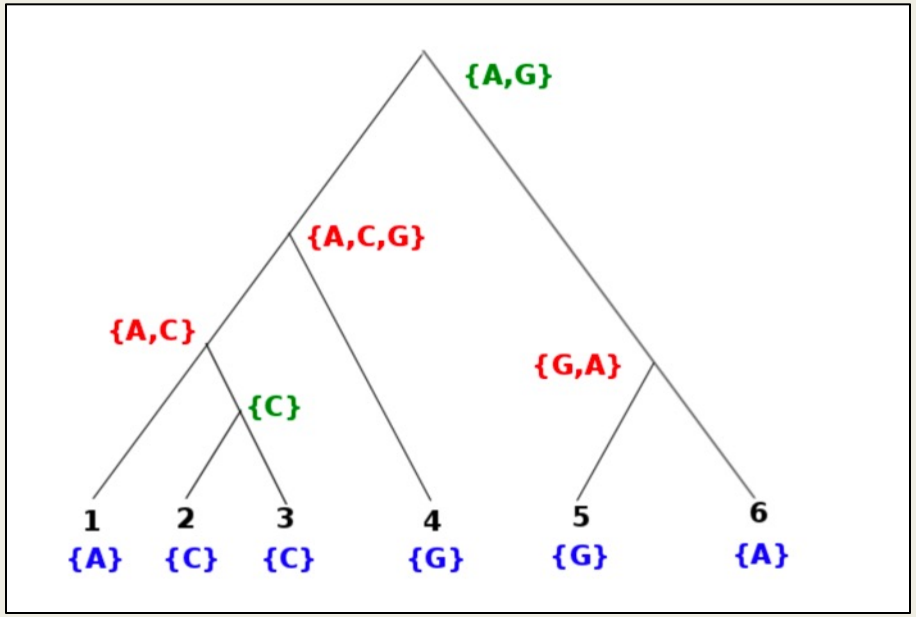
1. Traverse the tree from leaves to root, at each step assign to the parent node the set that is the intersection of the set of characters in the child nodes if it’s non empty, but the union if the intersection is empty.

2. Traverse the tree from root to leaves. Assign state of the root randomly from its set. At each node, if the parent state is in that node’s set, assign that. Otherwise, assign randomly.

Fitch’s algorithm

1. Traverse the tree from leaves to root, at each step assign to the parent node the intersection of the set of characters in the child nodes if it’s non empty, but the union if the intersection is empty.

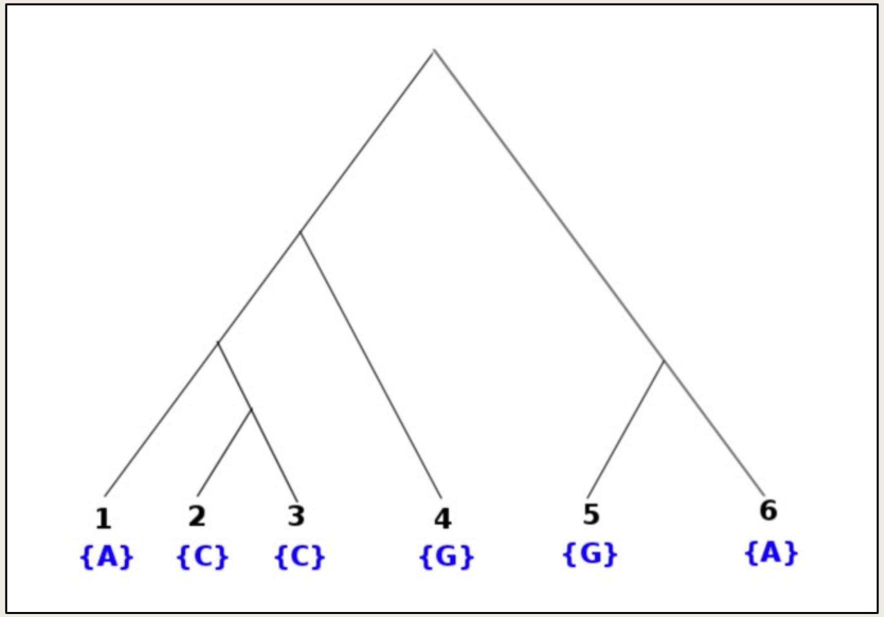


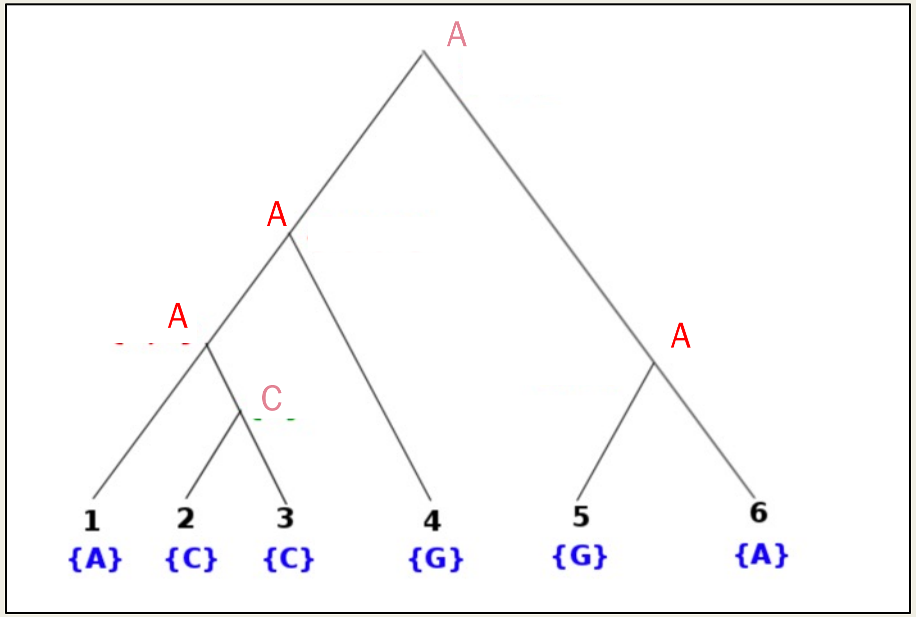


Intersection: no mutation
Union: mutation no matter what we choose

Fitch's algorithm

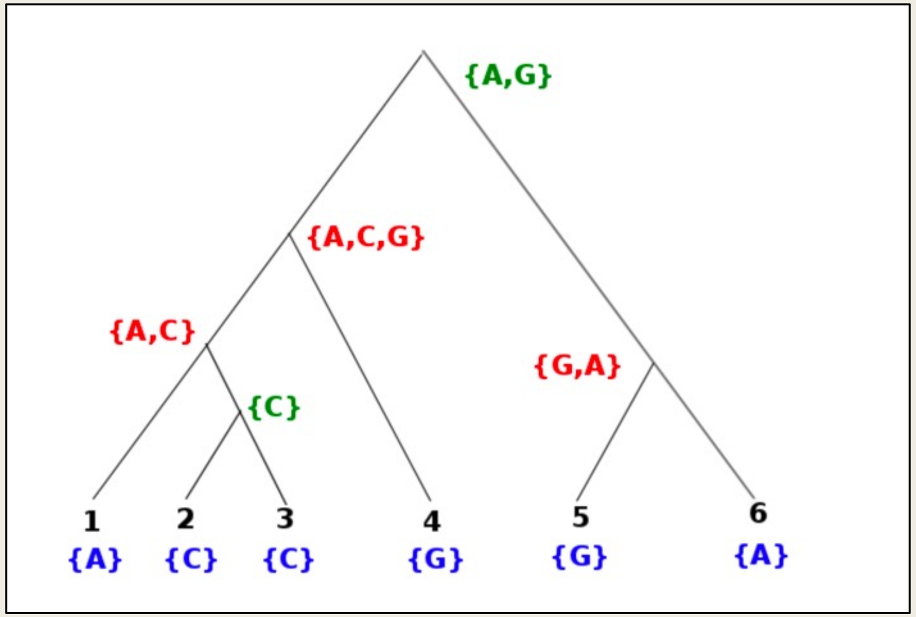
2. Traverse the tree from root to leaves. Assign state of the root randomly from its set. At each node, if the parent state is in that node's set, assign that. Otherwise, assign randomly.

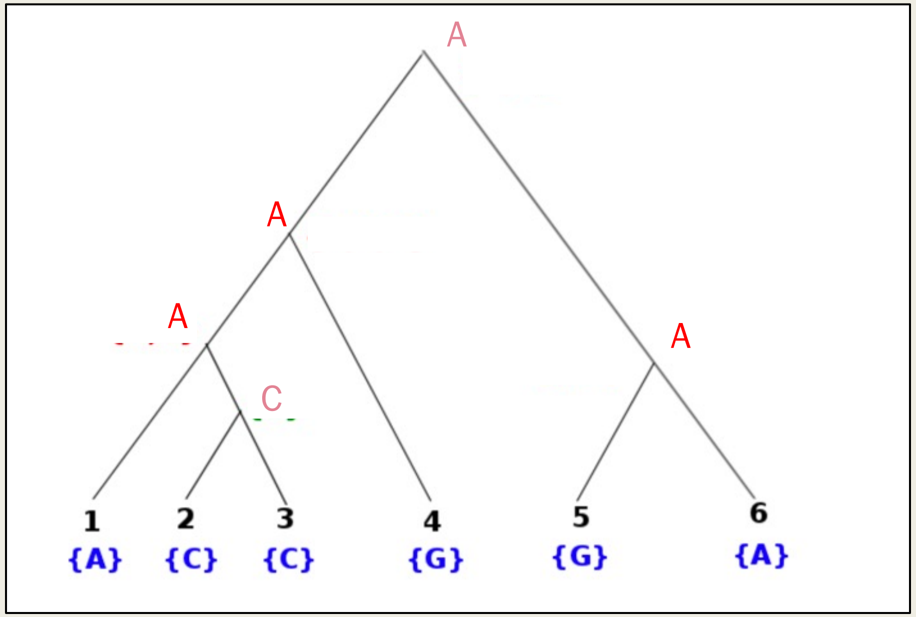




Fitch’s algorithm

The **parsimony score** is equal to the minimum number of mutations required



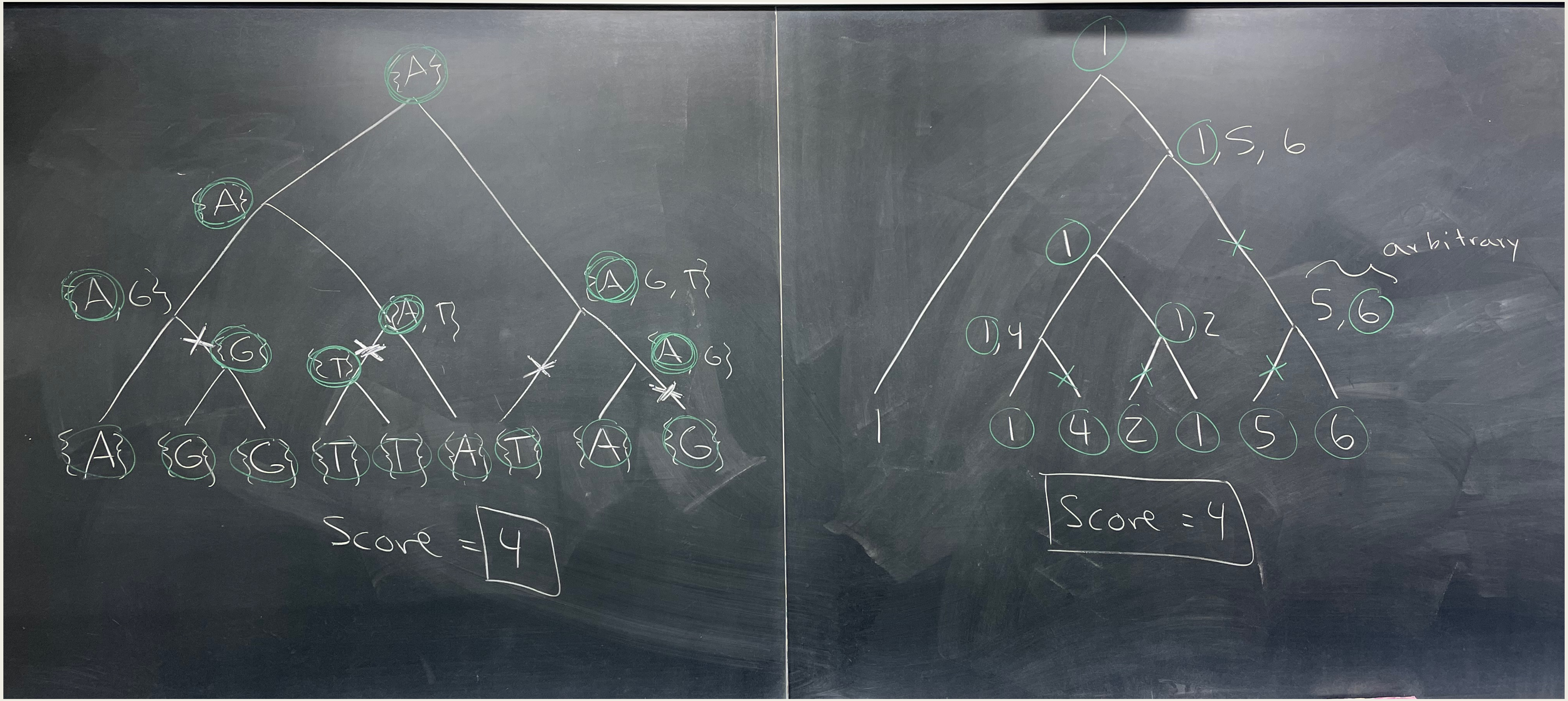


Fitch’s algorithm handout (Handout 14, page 1)

1. Traverse the tree from leaves to root, at each step assign to the parent node the set that is the intersection of the set of characters in the child nodes if the intersection is non-empty, but the union if the intersection is empty.

2. Traverse the tree from root to leaves. Assign state of the root randomly from its set. At each node, if the parent state is in that node’s set, assign that. Otherwise, assign randomly from the child node character set.

Fitch's algorithm handout (Handout 14, page 1)



Sankoff’s algorithm (weighted parsimony)

Sankoff's algorithm

Fitch's algorithm assumes that every possible change is equally [un]likely

Sankoff's (1975) is a generalization that allows different changes to have different costs. Keeps track of the cost of being in each state at each node.

E.g.

To

From

σ	a	b
a	0	2
b	1	0

a->b cost 2
b->a cost 1

Sankoff : weighted parsimony

input: rooted binary tree with labeled leaves & cost function σ

i.e. $\sigma(a,b)$ is cost of mutation $a \to b$

output: internal vertex labels that minimize weighted parsimony score

• in

a
c
g
t

parsimony

tree with

s & cost

cost of

b

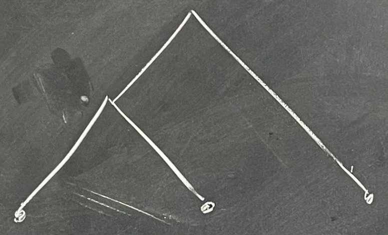
labels

weighted

ore

- initialization

leaf label characters assigned score of 0, other characters ∞

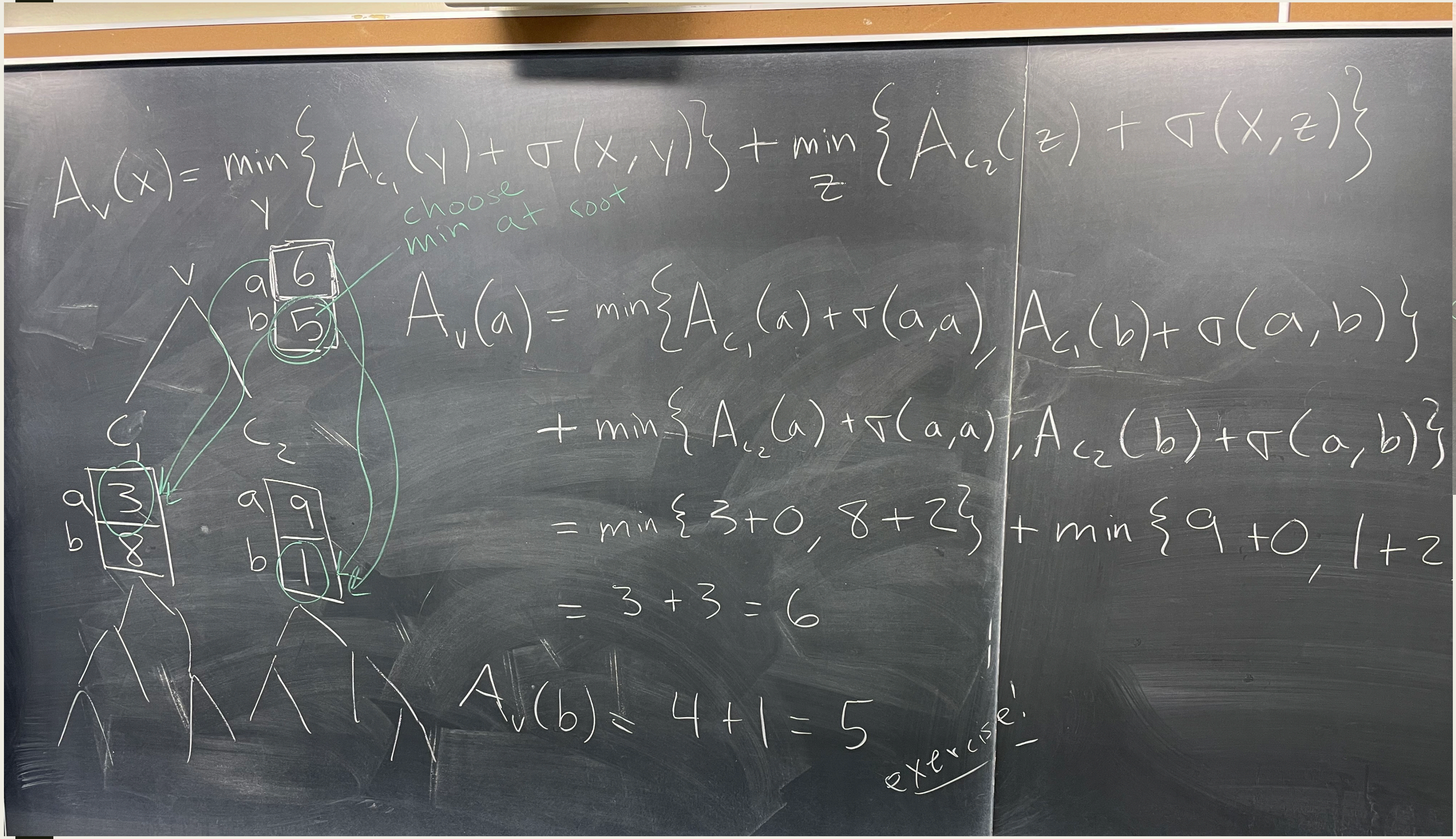


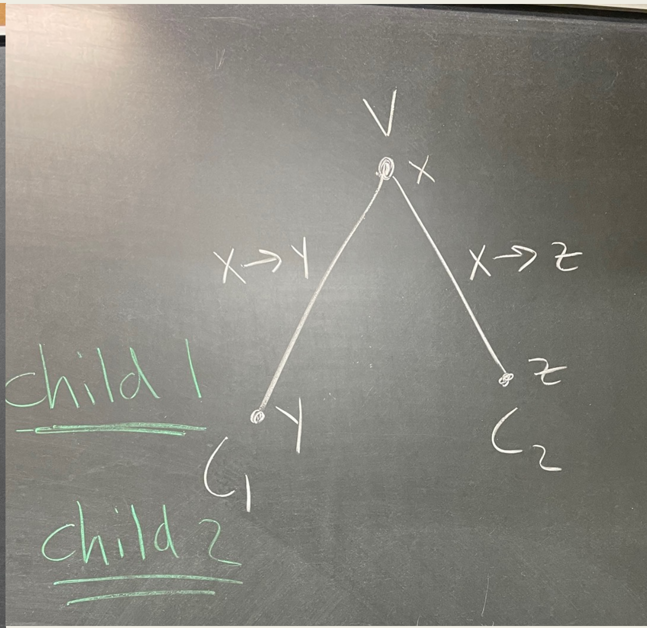
	c	g	t
a	∞	∞	∞
c	0	∞	∞
g	∞	0	∞
t	∞	∞	0

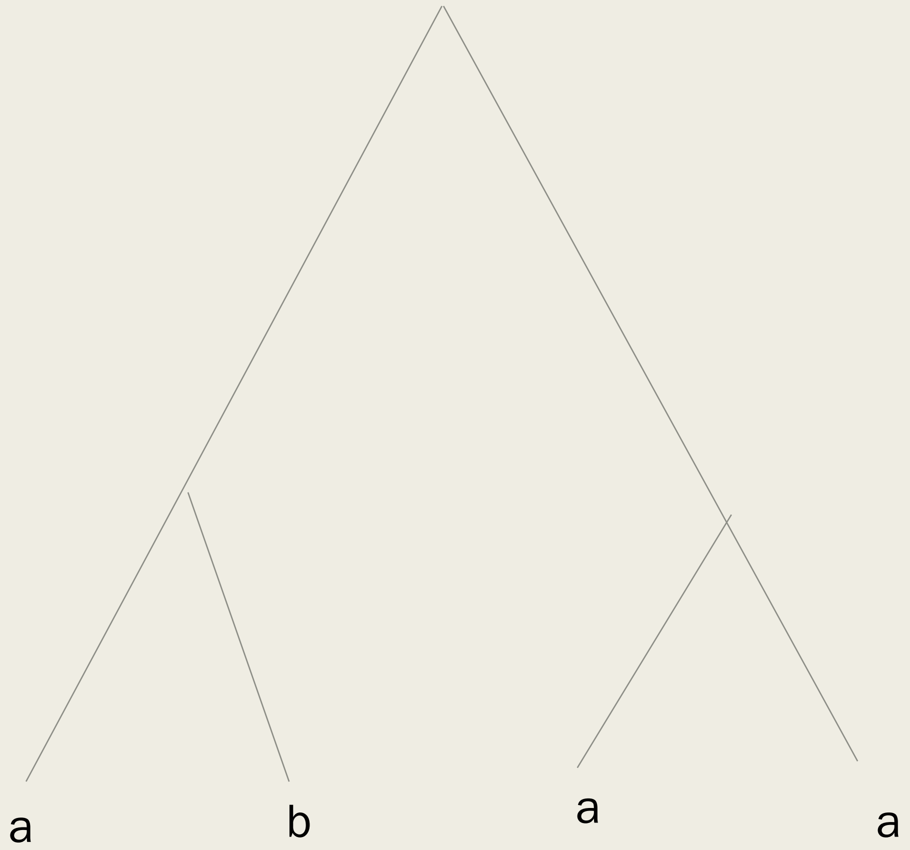
- iterative step

let $A_v(x) =$ optimal cost of subtree rooted at v, with label x

[c_1 & c_2 children of v]


$A_v(x) = \min_y \{A_{c_1}(y) + \sigma(x,y)\} + \min_z \{A_{c_2}(z) + \sigma(x,z)\}$
choose min at root
v
a 6
b 5
c_1 c_2
a 3
b 8
a 9
b 1
$A_v(a) = \min\{A_{c_1}(a) + \sigma(a,a), A_{c_1}(b) + \sigma(a,b)\}$
$+ \min\{A_{c_2}(a) + \sigma(a,a), A_{c_2}(b) + \sigma(a,b)\}$
$= \min\{3+0, 8+2\} + \min\{9+0, 1+2$
$= 3 + 3 = 6$
$A_v(b) = 4 + 1 = 5$
exercise!

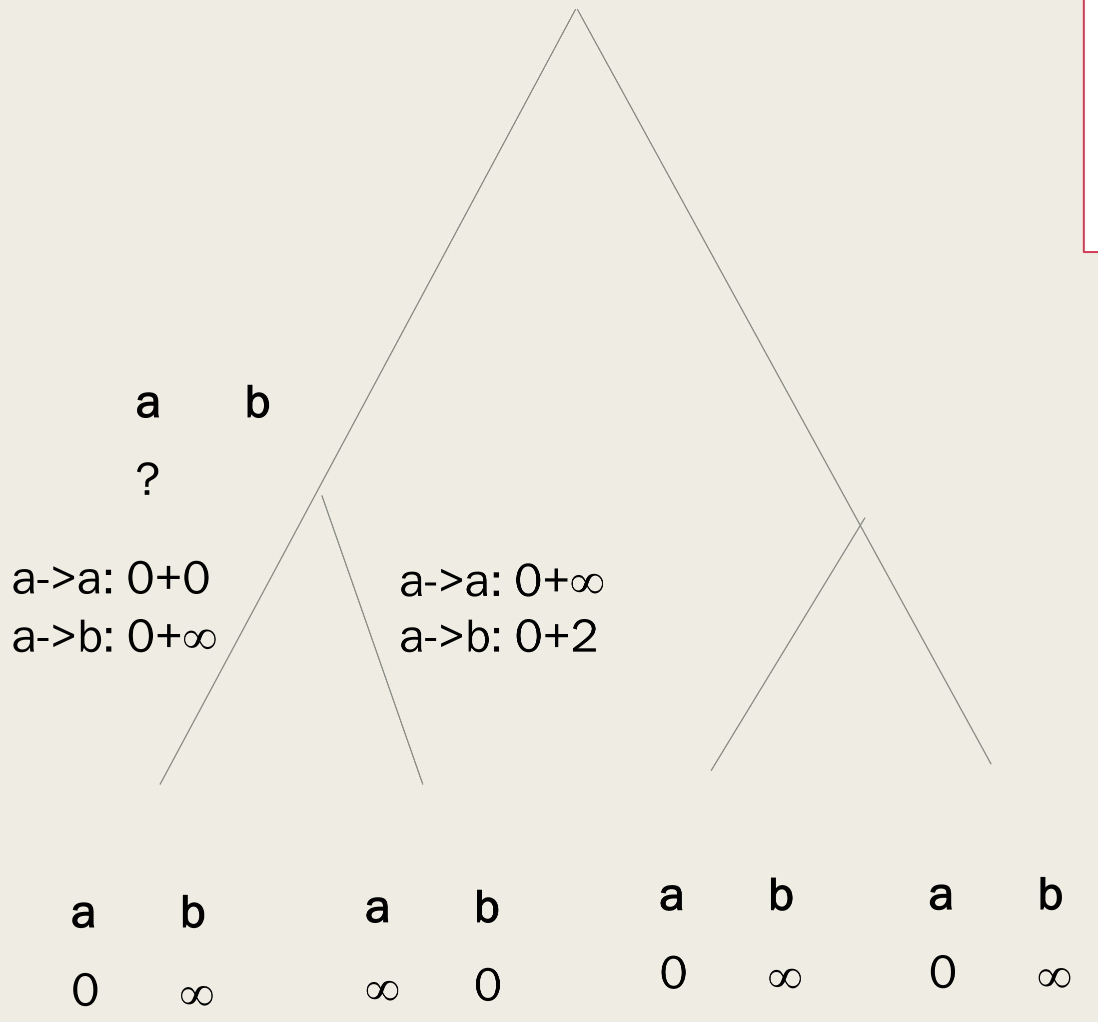

v
x
$x \to y$
$x \to z$
child 1
y
z
c_1
c_2
child 2

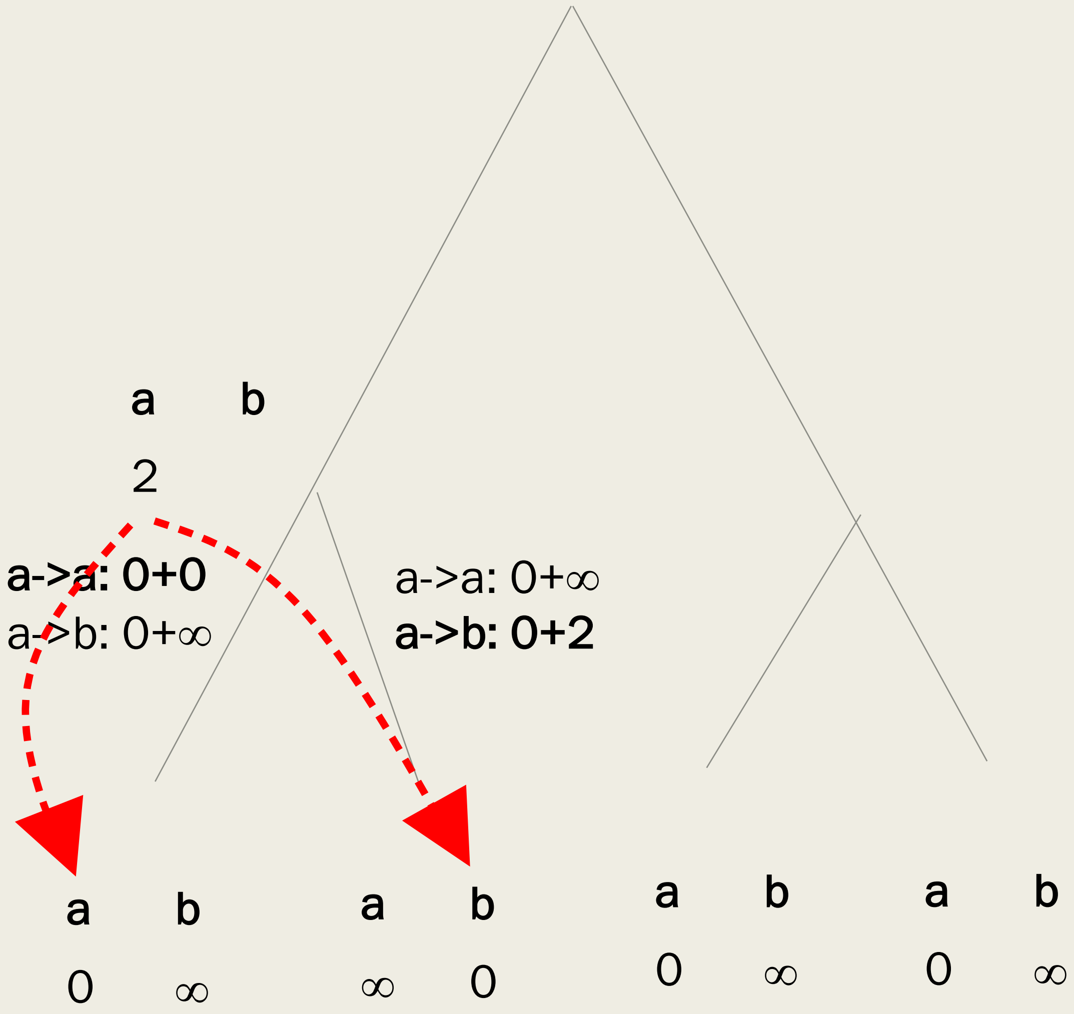


σ	a	b
a	0	2
b	1	0

σ	a	b
a	0	2
b	1	0

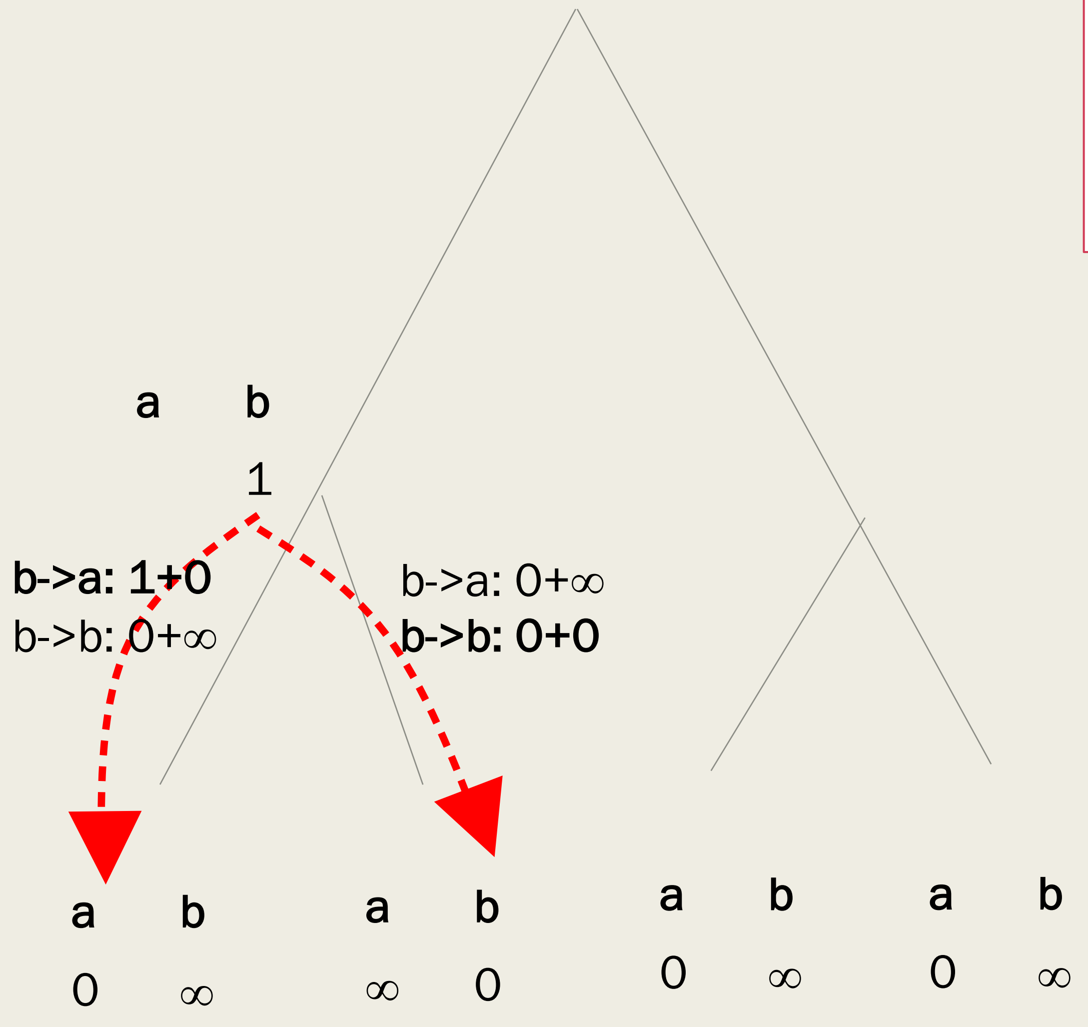
σ	a	b
a	0	2
b	1	0



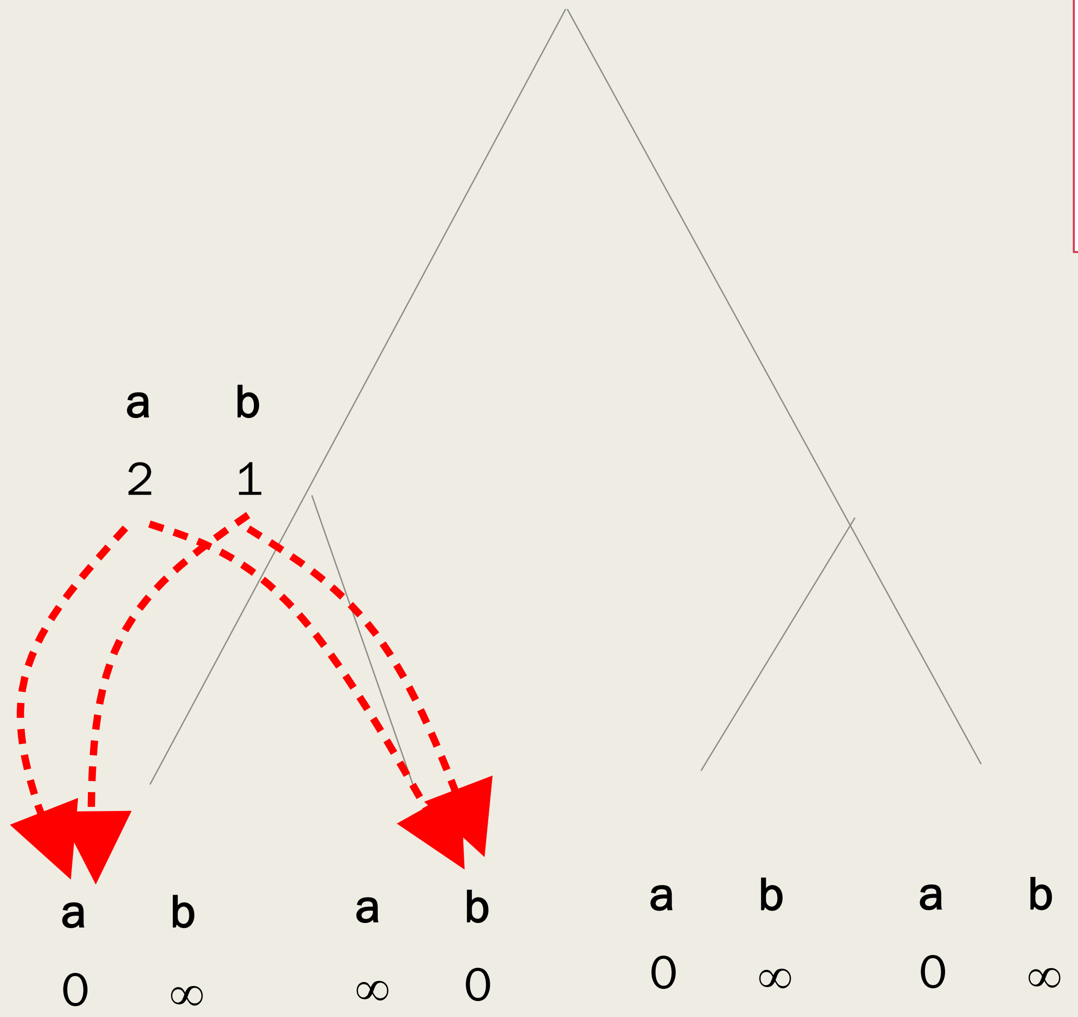


σ	a	b
a	0	2
b	1	0

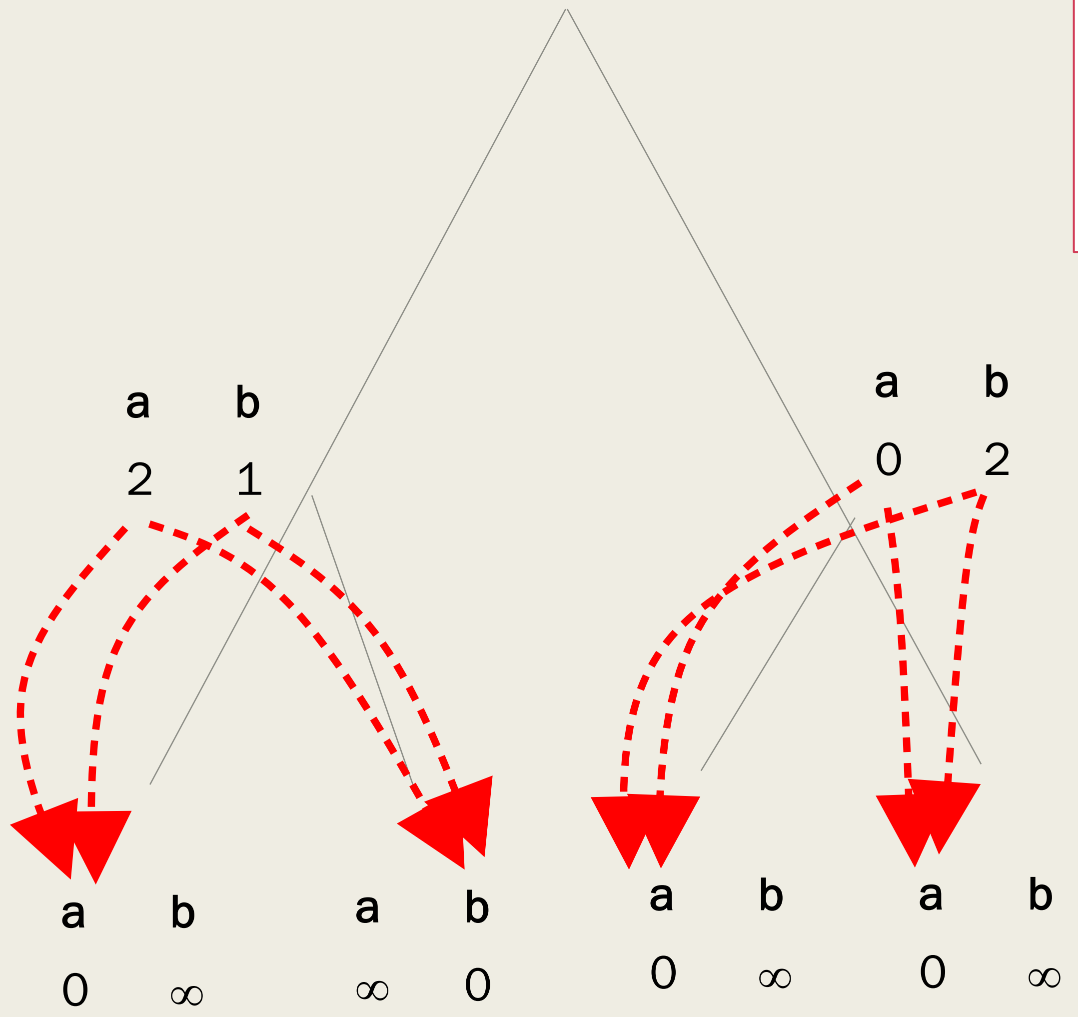
σ	a	b
a	0	2
b	1	0

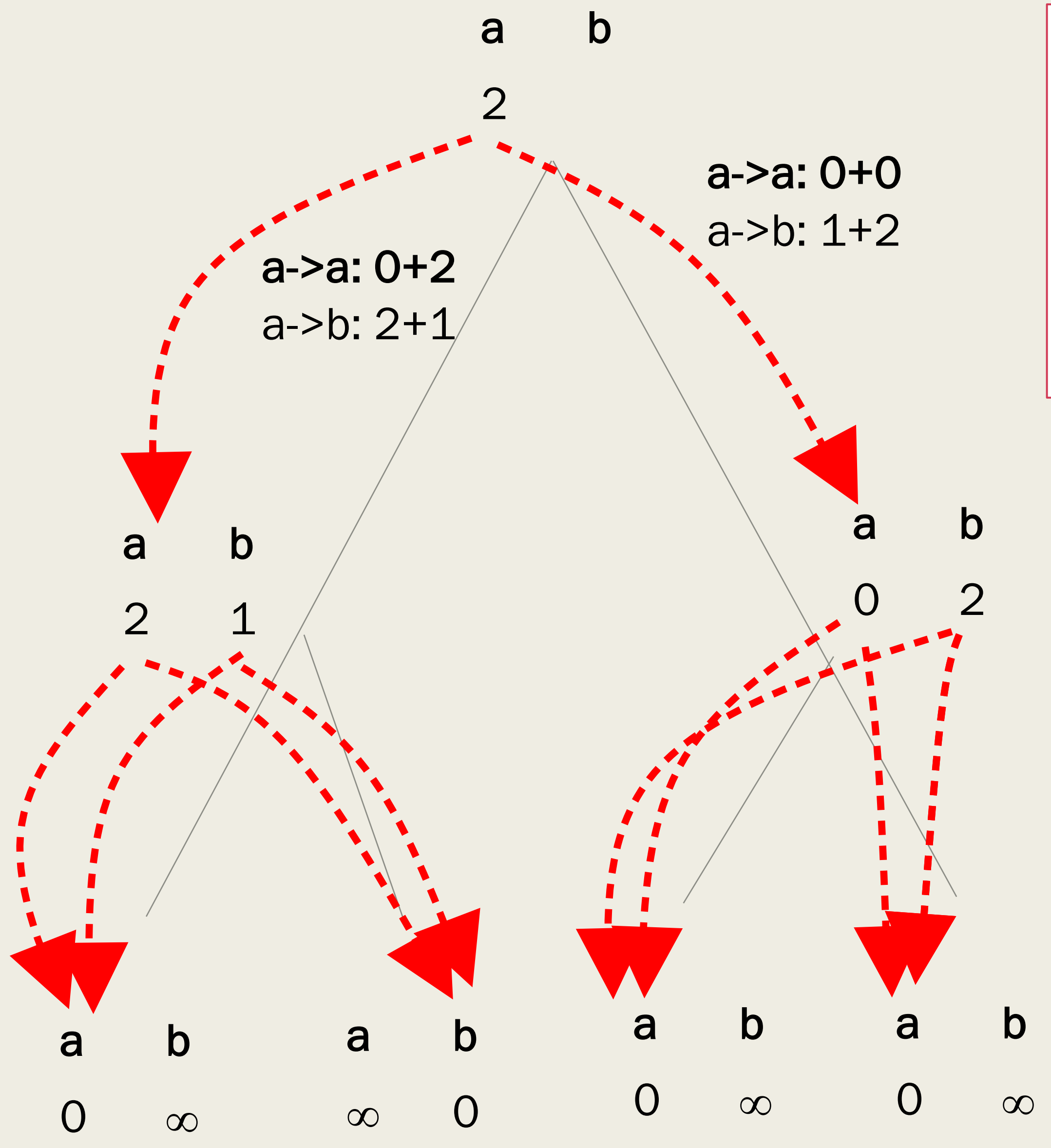


σ	a	b
a	0	2
b	1	0

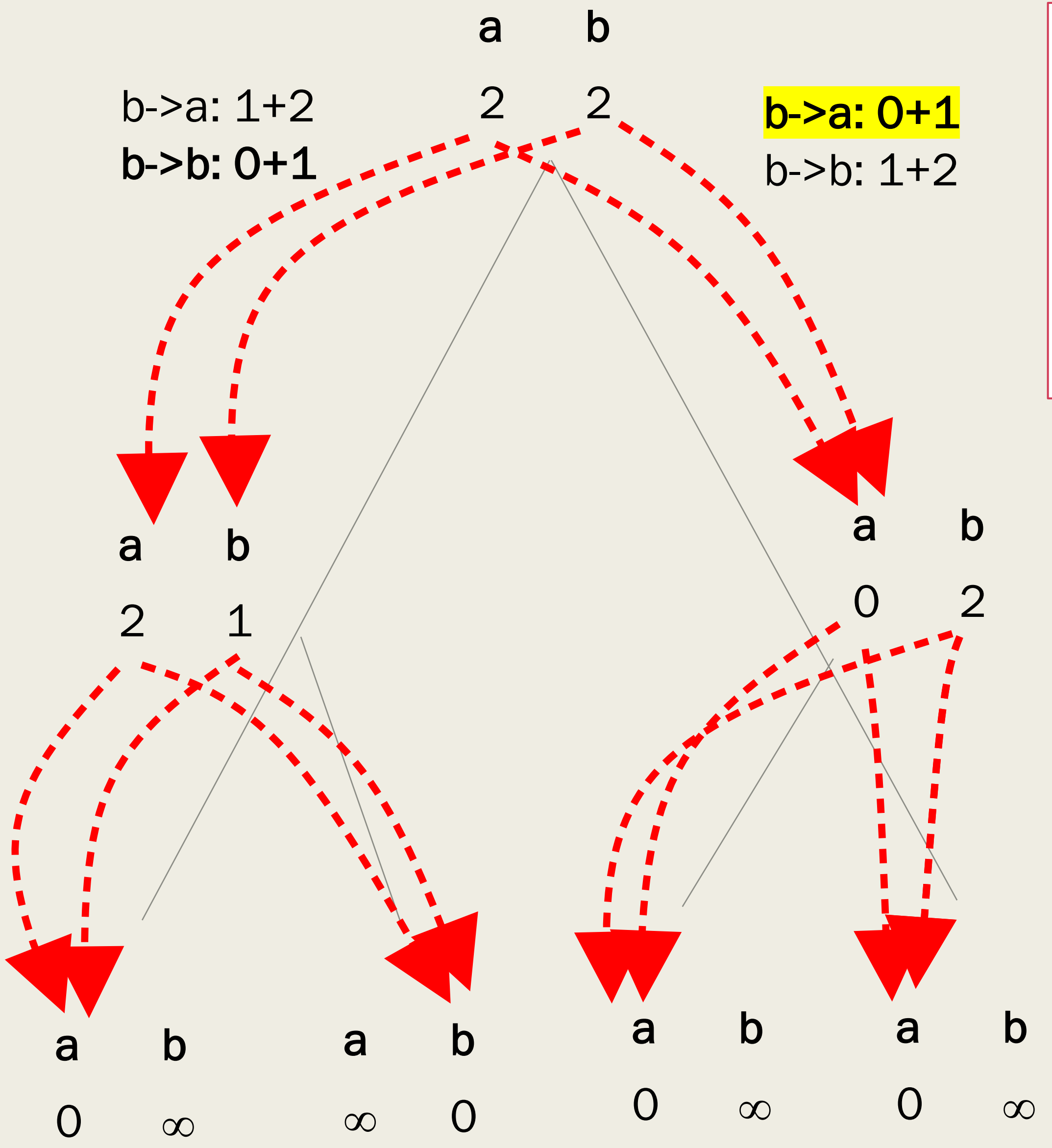


σ	a	b
a	0	2
b	1	0





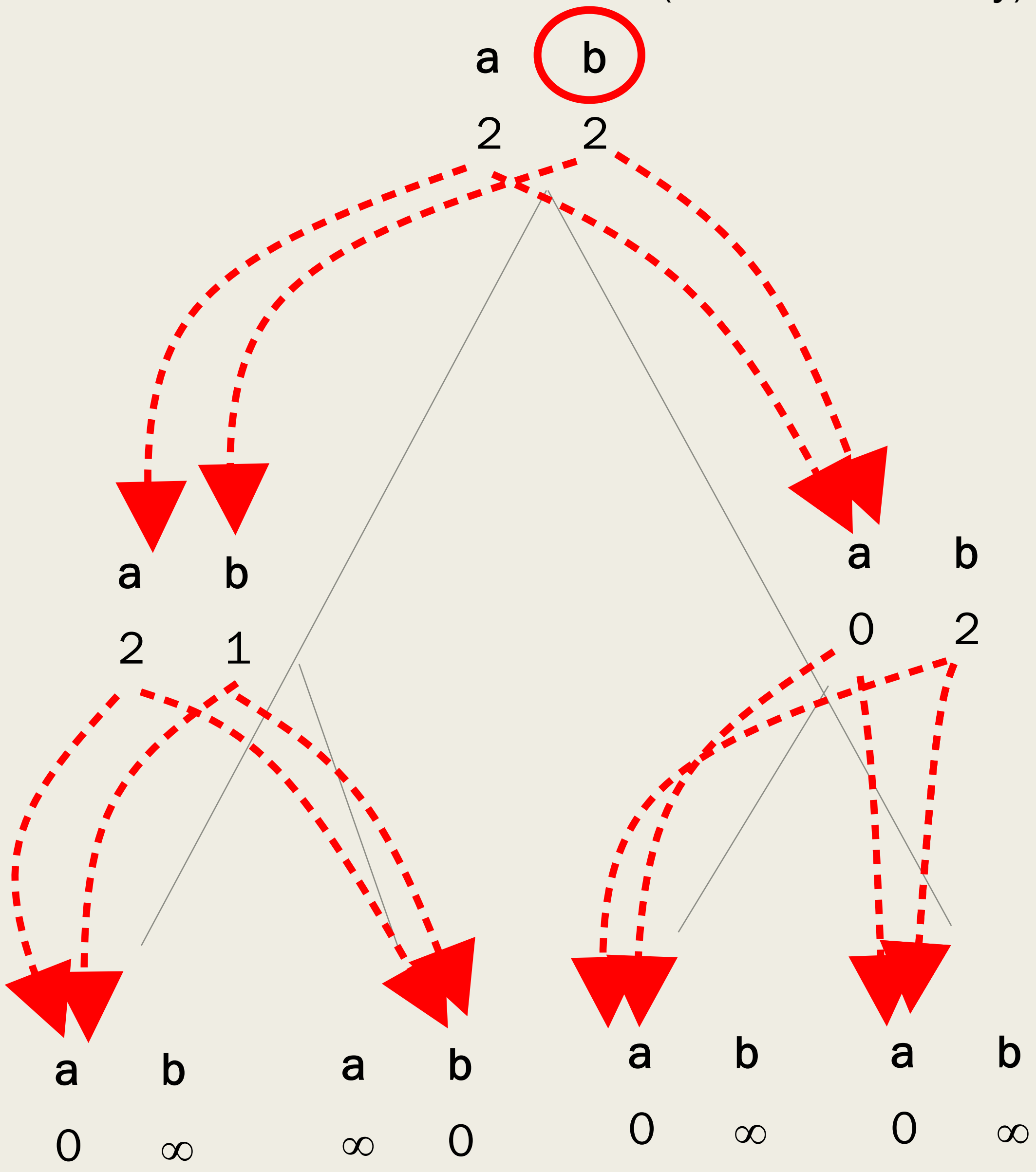
σ	a	b
a	0	2
b	1	0



σ	a	b
a	0	2
b	1	0

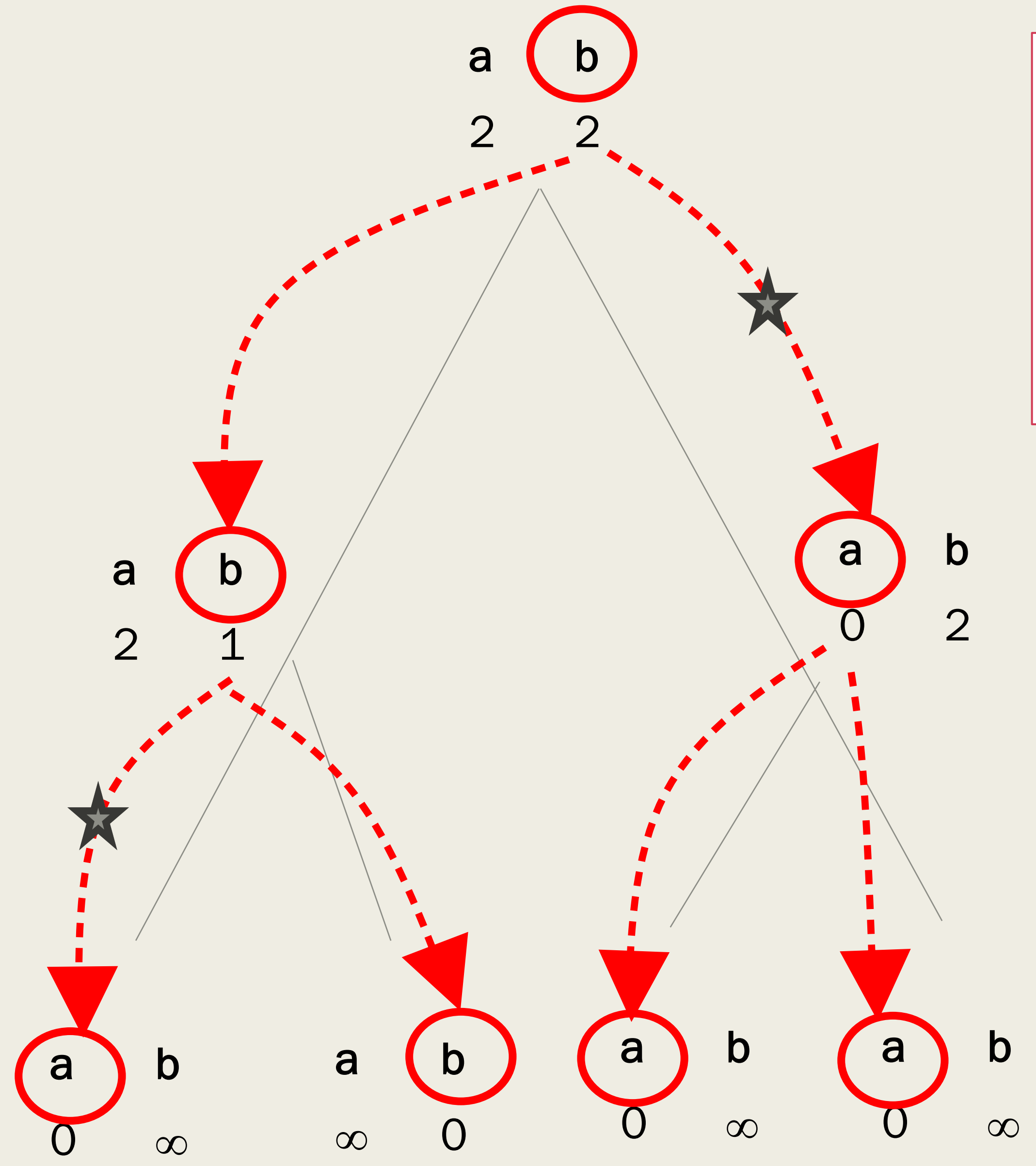
Corrected!

(choose arbitrarily)



σ	a	b
a	0	2
b	1	0

Corrected!



σ	a	b
a	0	2
b	1	0

★ mutation

Corrected!

Sankoff's algorithm handout

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

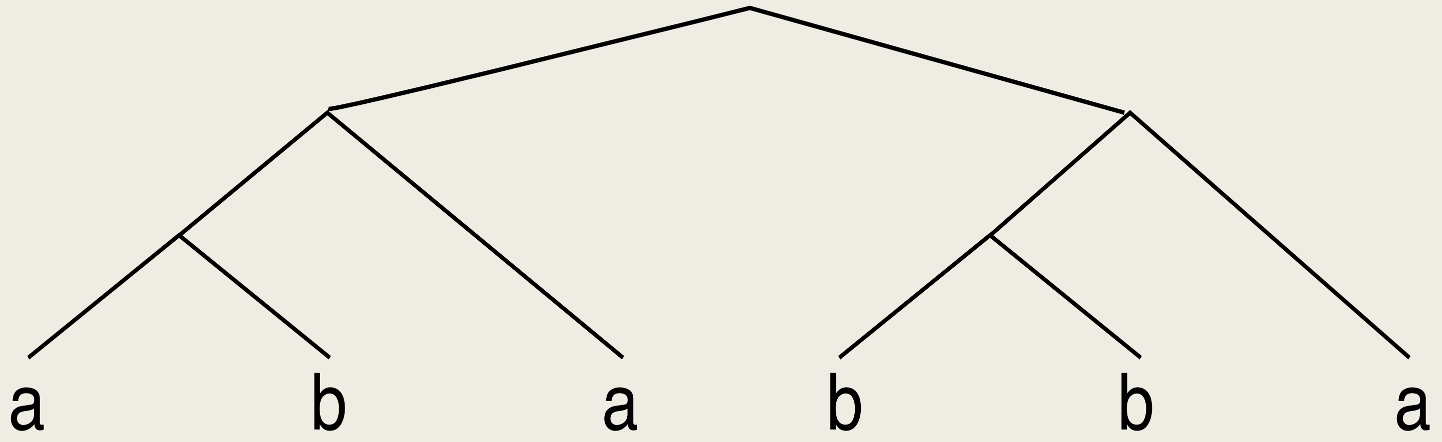
$$A_v(x) = \min_y\{A_{c_1}(y) + \sigma(x,y)\} + \min_z\{A_{c_2}(z) + \sigma(x,z)\}.$$

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.

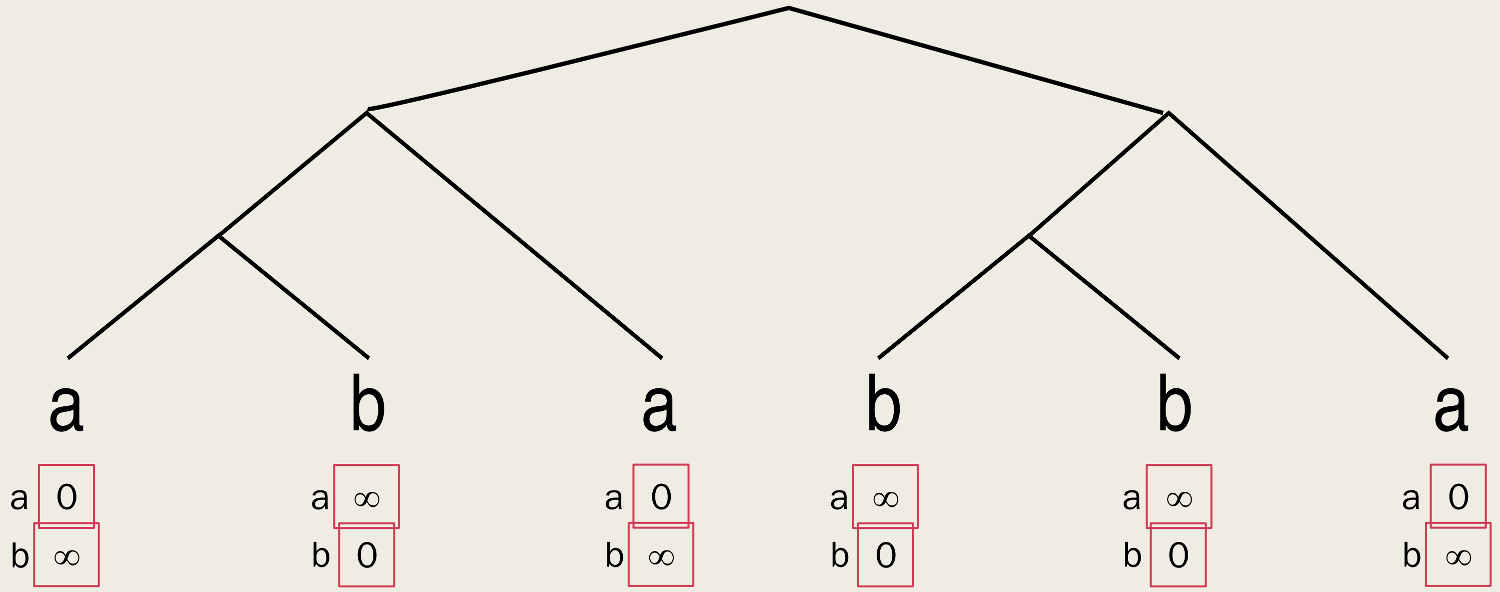
Sankoff algorithm

σ	a	b
a	0	2
b	1	0



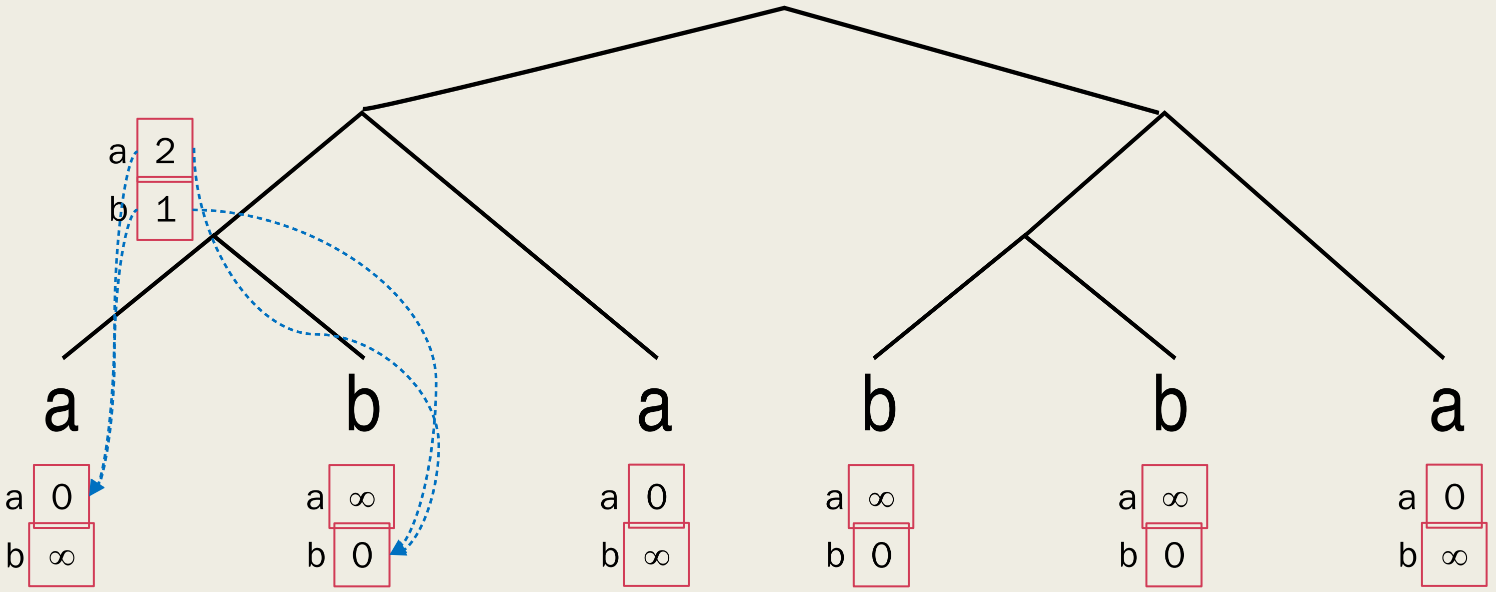
Sankoff algorithm

σ	a	b
a	0	2
b	1	0



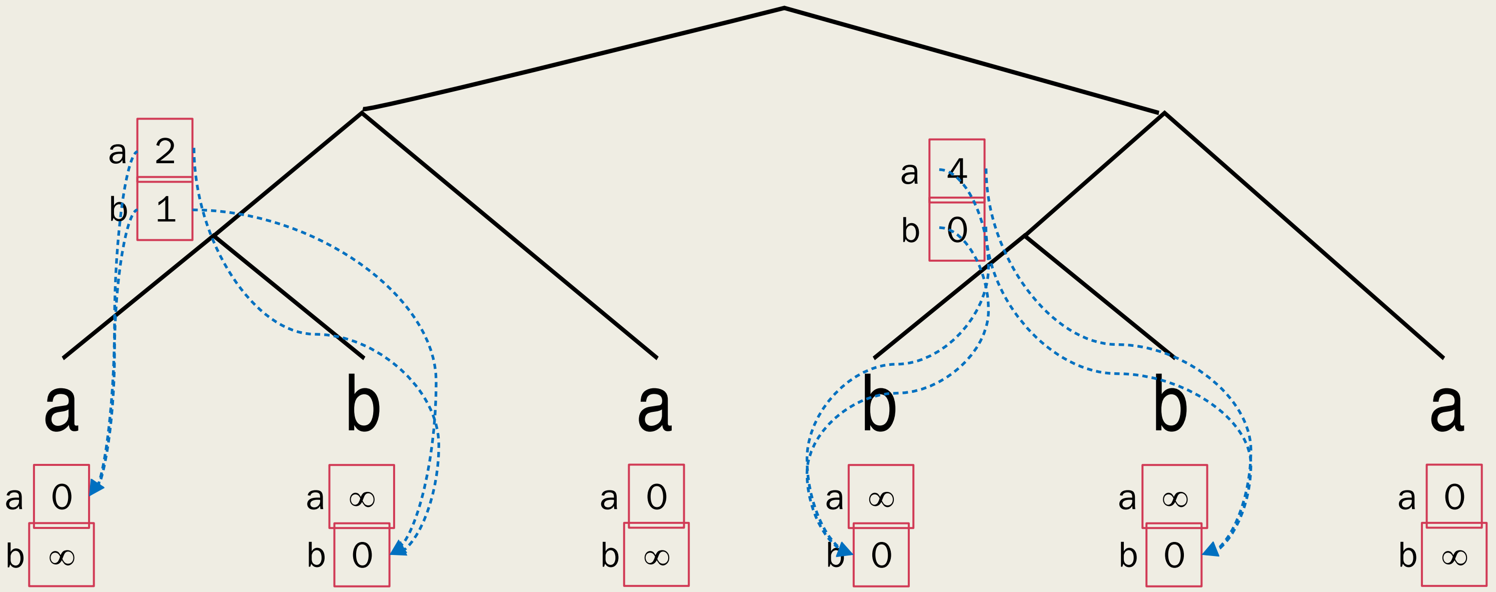
Sankoff algorithm

σ	a	b
a	0	2
b	1	0



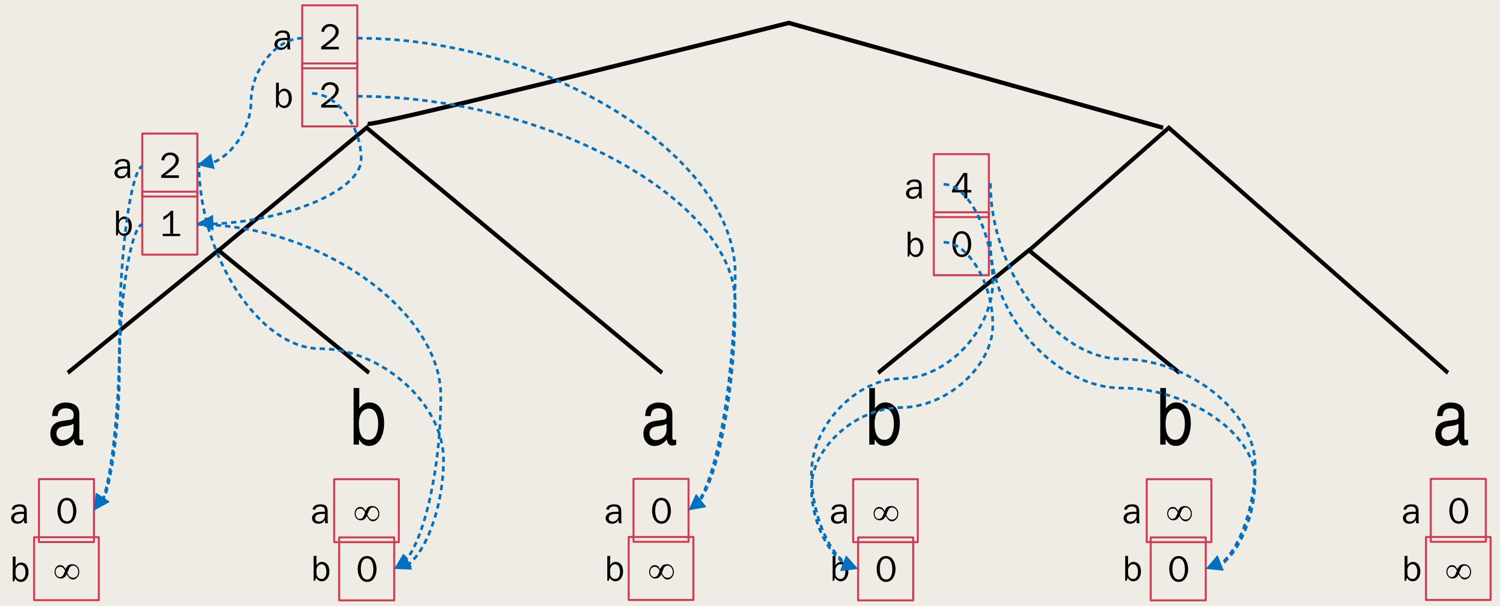
Sankoff algorithm

σ	a	b
a	0	2
b	1	0



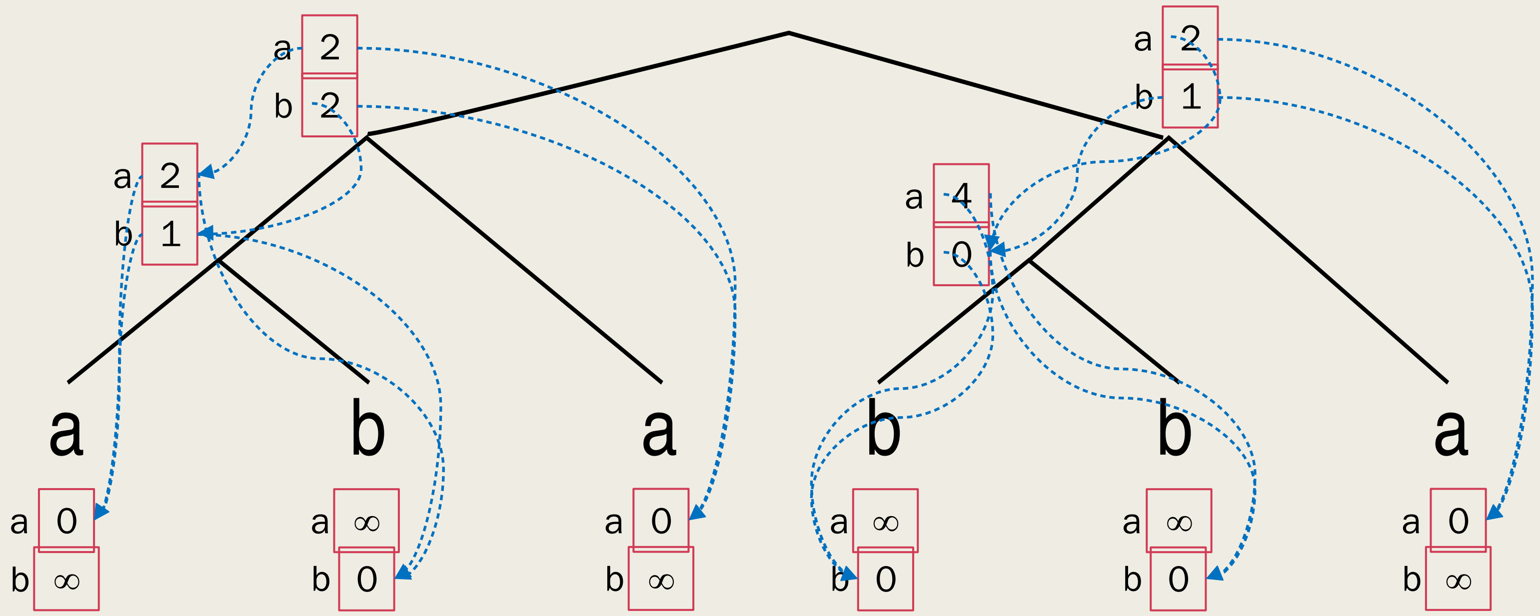
Sankoff algorithm

σ	a	b
a	0	2
b	1	0



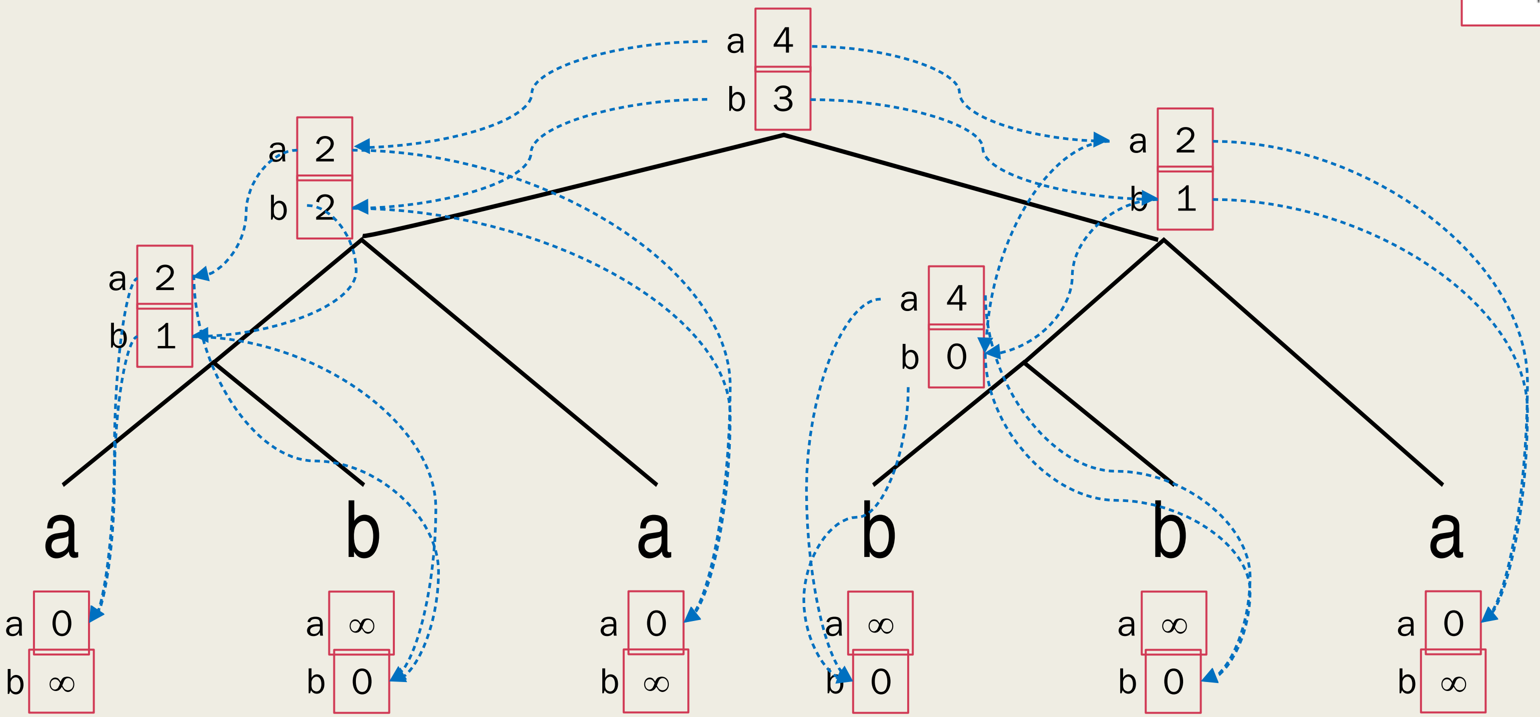
Sankoff algorithm

σ	a	b
a	0	2
b	1	0



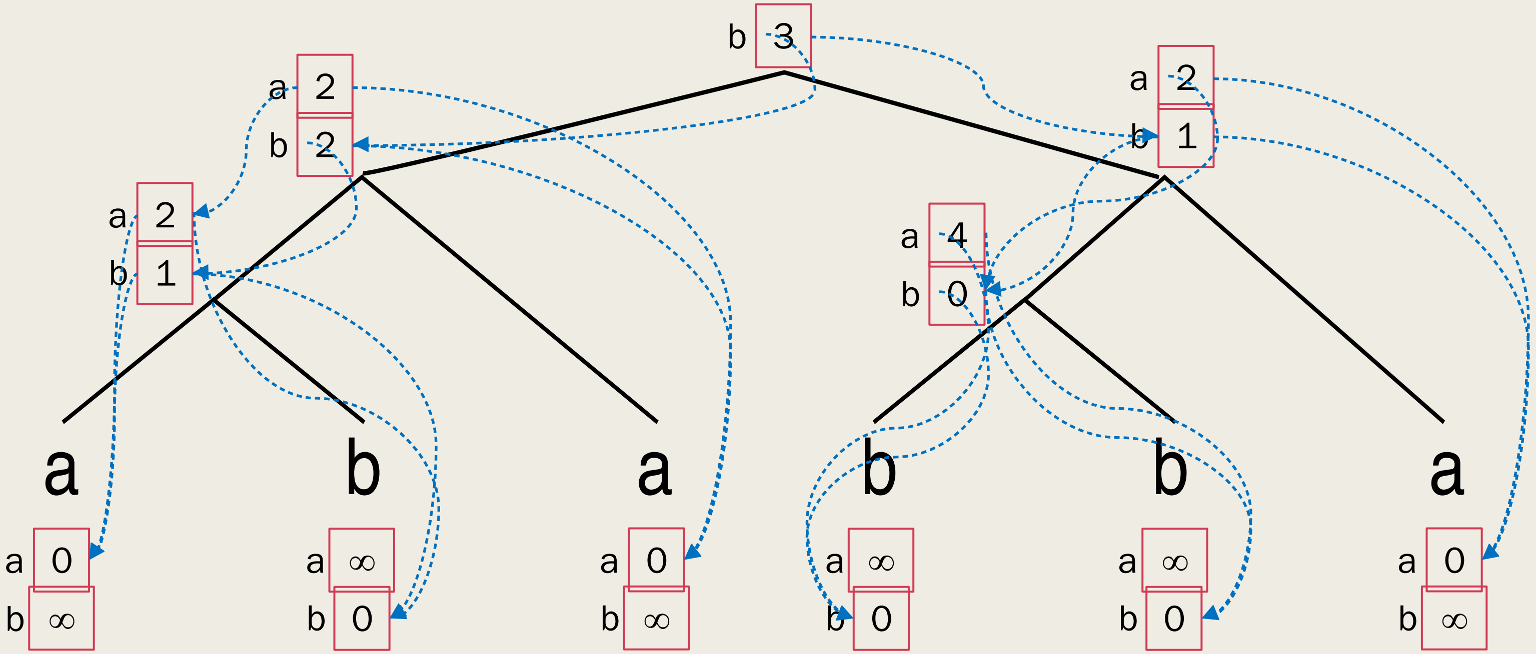
Sankoff algorithm

σ	a	b
a	0	2
b	1	0



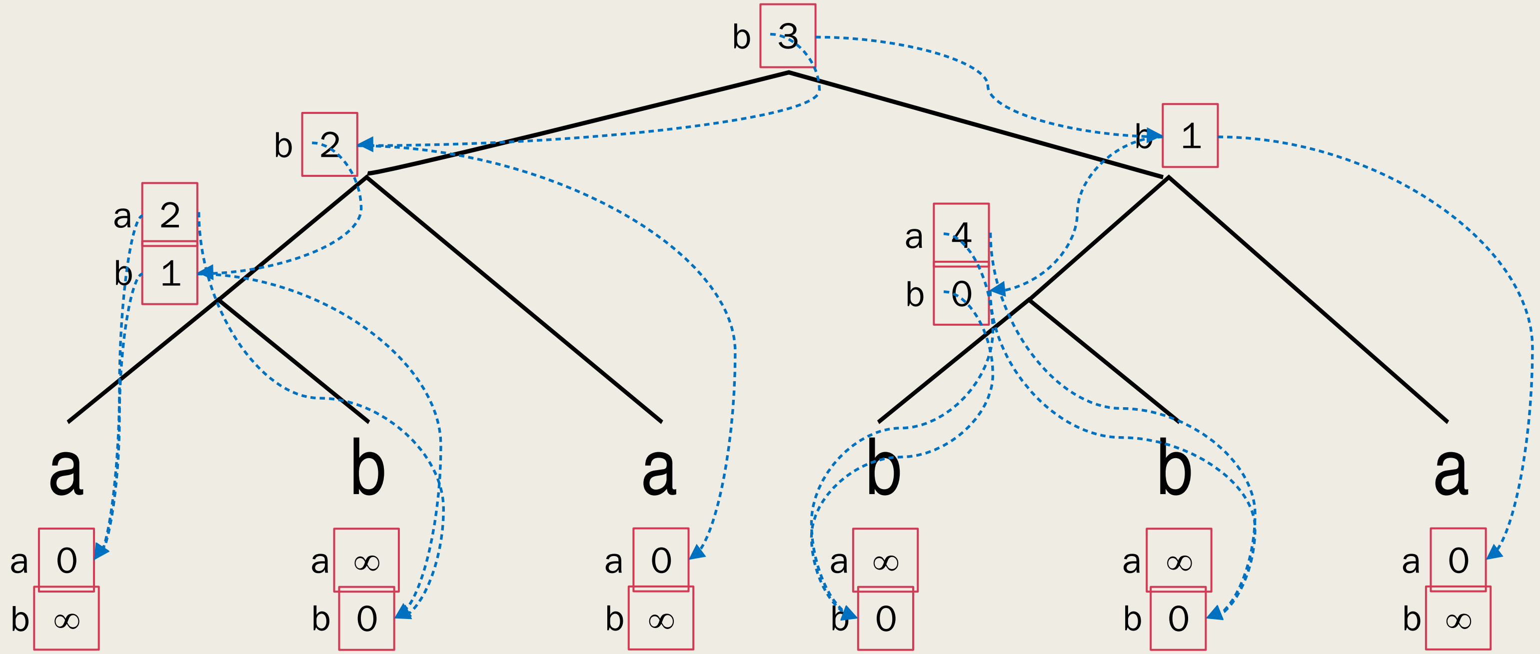
Sankoff algorithm

σ	a	b
a	0	2
b	1	0



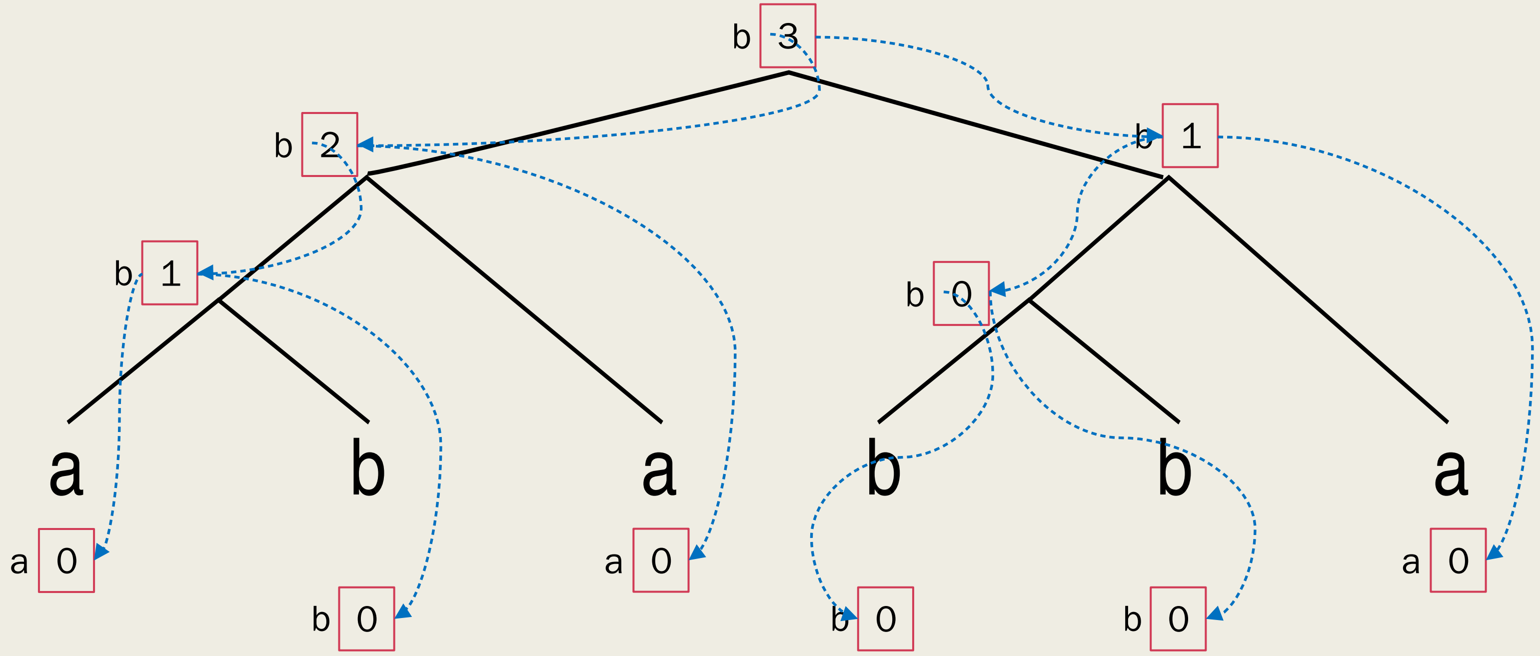
Sankoff algorithm

σ	a	b
a	0	2
b	1	0



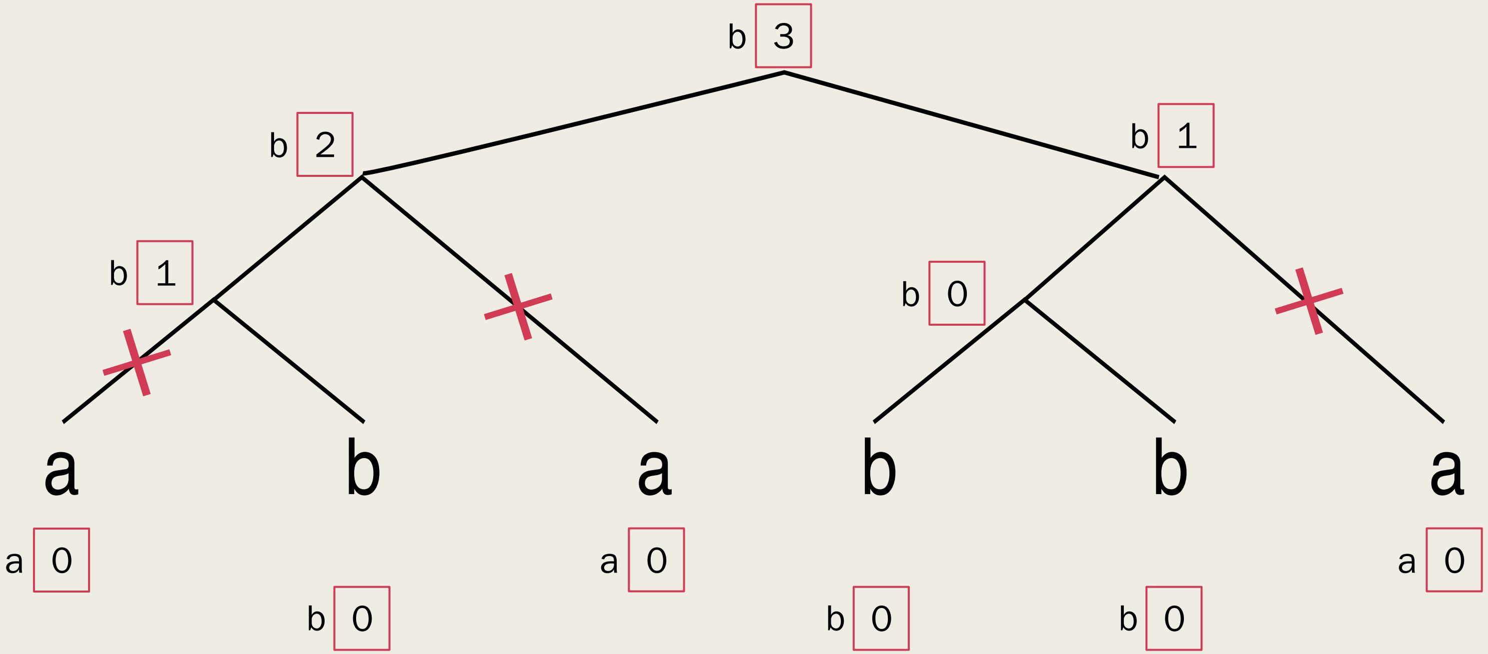
Sankoff algorithm

σ	a	b
a	0	2
b	1	0

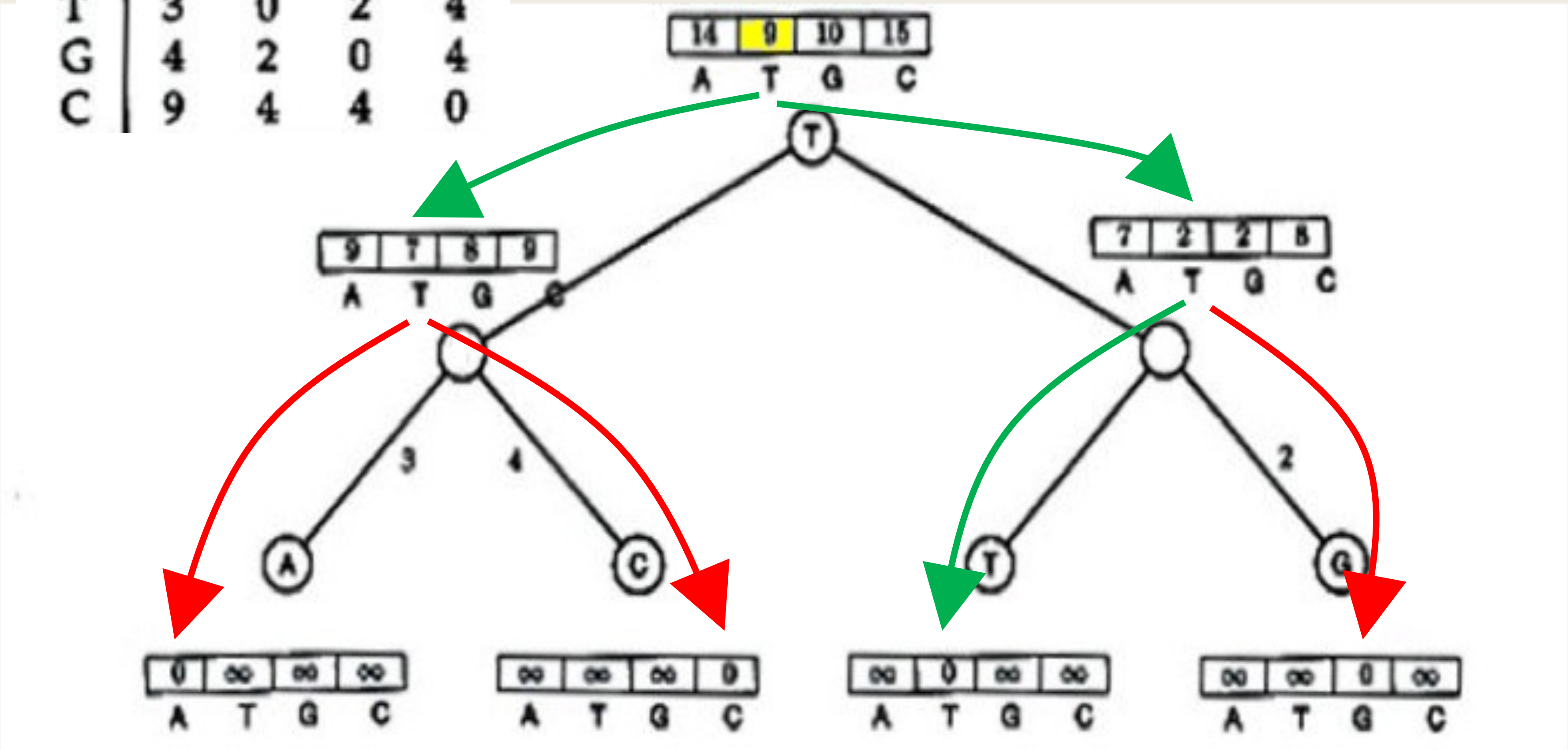


Sankoff algorithm

σ	a	b
a	0	2
b	1	0



δ	A	T	G	C
A	0	3	4	9
T	3	0	2	4
G	4	2	0	4
C	9	4	4	0



Runtime

Suppose there are N leaves and K states

What is the complexity of Fitch's algorithm?

What is the complexity of Sankoff's algorithm?