CS 364 COMPUTATIONAL BIOLOGY

Sara Mathieson
Haverford College

Outline

- Recap Sankoff's Algorithm
- Perfect Phylogeny problem
- Gusfield's Algorithm

Notes:
- Lab 6 posted, due Mon
- Partners required
- Final project info coming soon!

Recap Sankoff’s algorithm (weighted parsimony)

Ancestral state reconstruction via parsimony

- <u>Input:</u> rooted, binary phylogenetic tree and leave labels

- <u>Output:</u> internal vertex labels that minimize the parsimony score (weighted or unweighted)

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.

$$\begin{array}{c|cc} \sigma & a & b \\ \hline a & 0 & 2 \\ b & 1 & 0 \end{array}$$

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

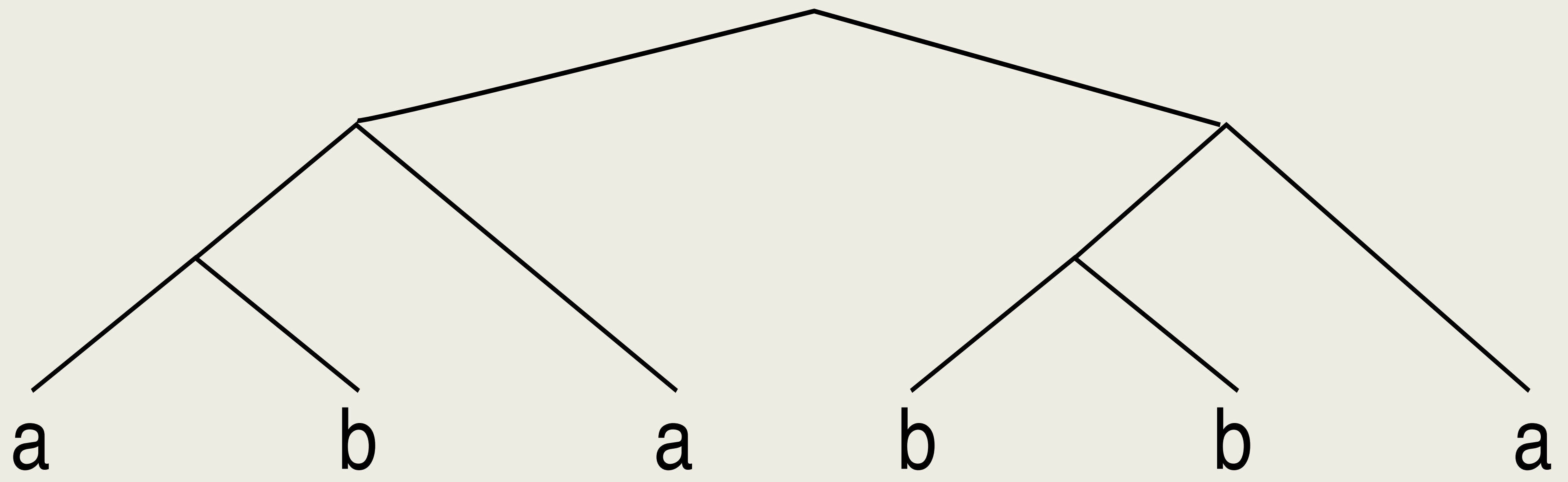
$$A_v(x) = \min_y \{A_{c_1}(y) + \sigma(x,y)\} + \min_z \{A_{c_2}(z) + \sigma(x,z)\}.$$

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.

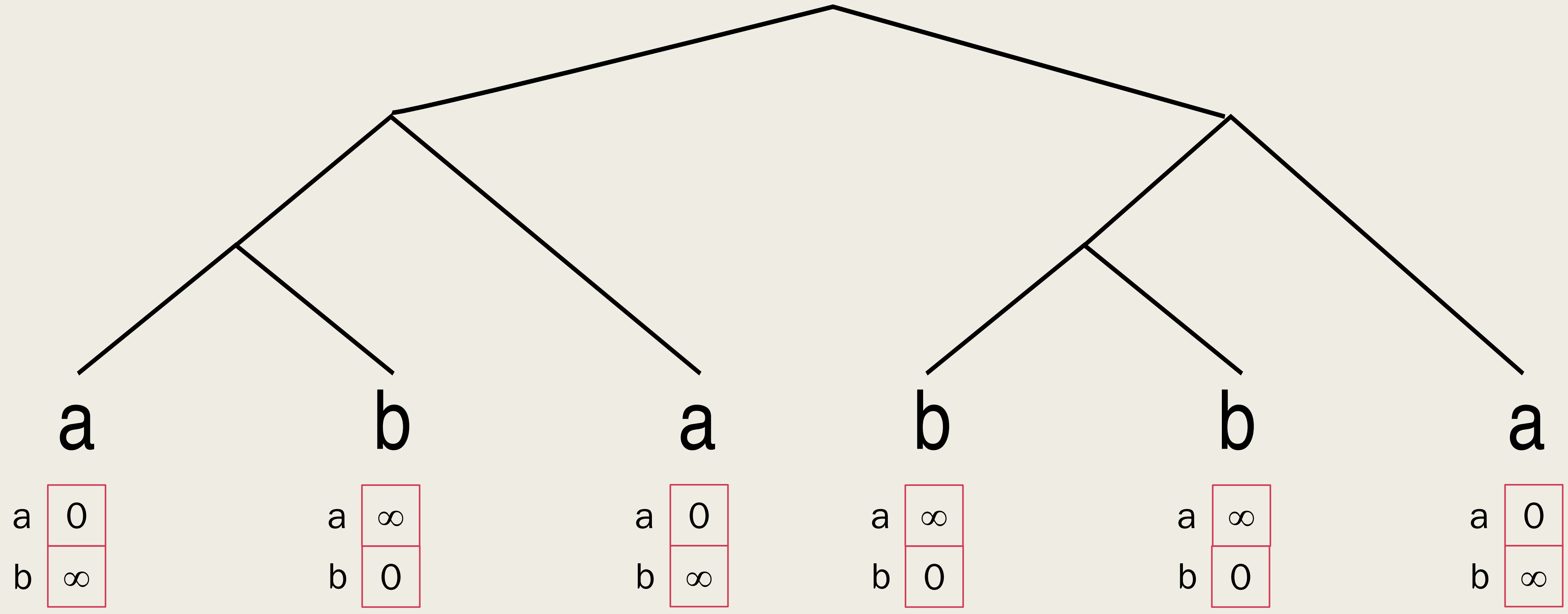
Handout 14 example

σ	a	b
a	0	2
b	1	0



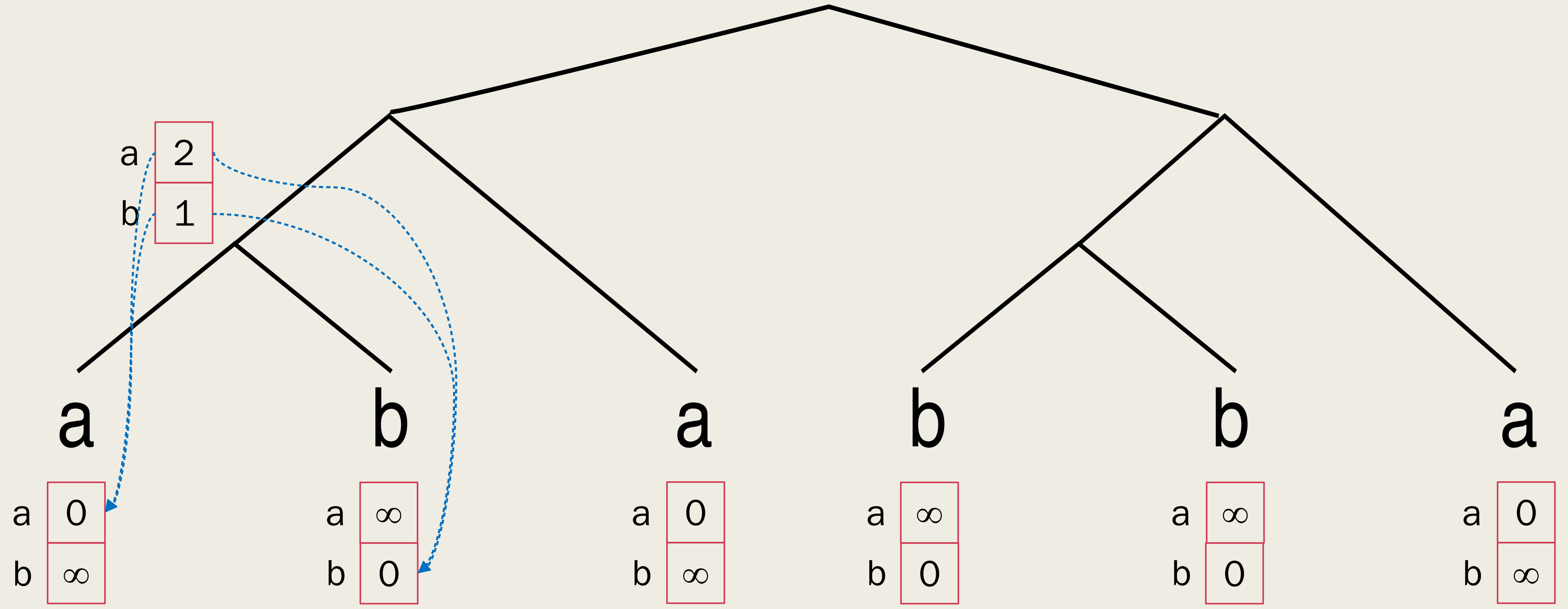
Handout 14 example

σ	a	b
a	0	2
b	1	0



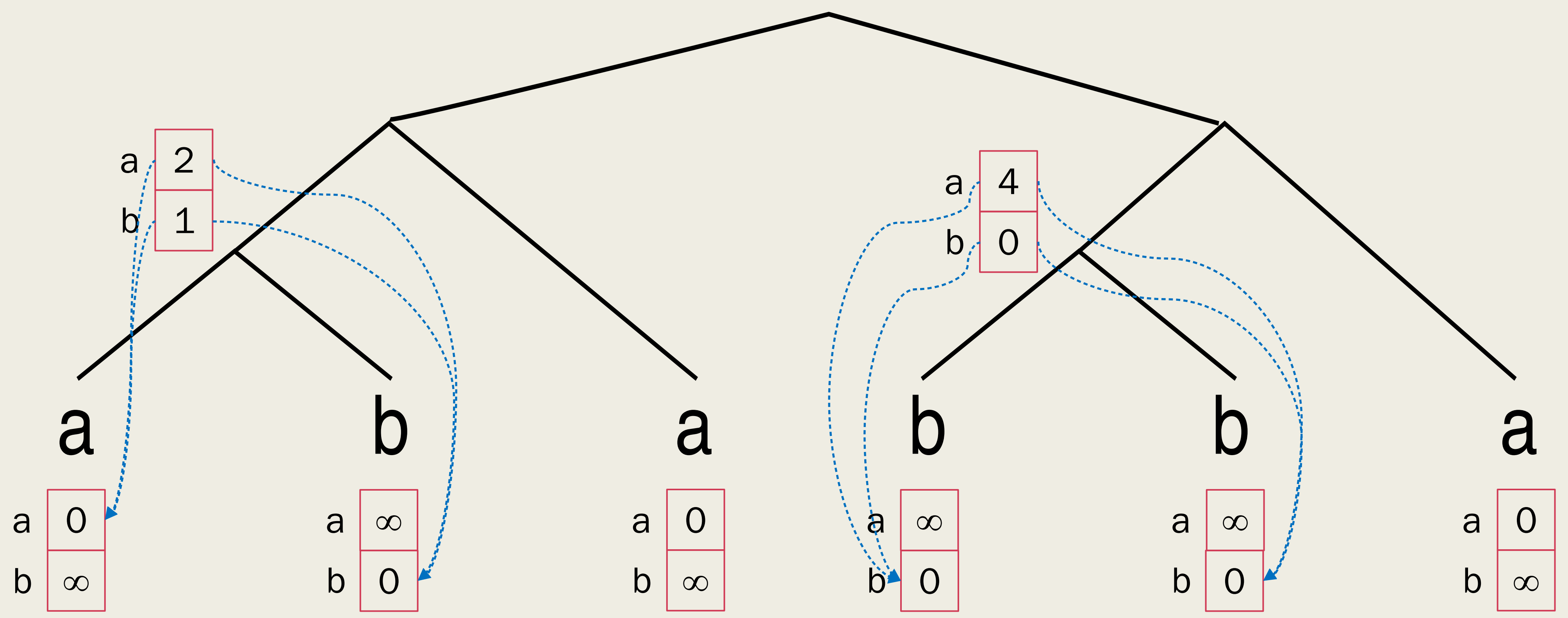
Handout 14 example

σ	a	b
a	0	2
b	1	0



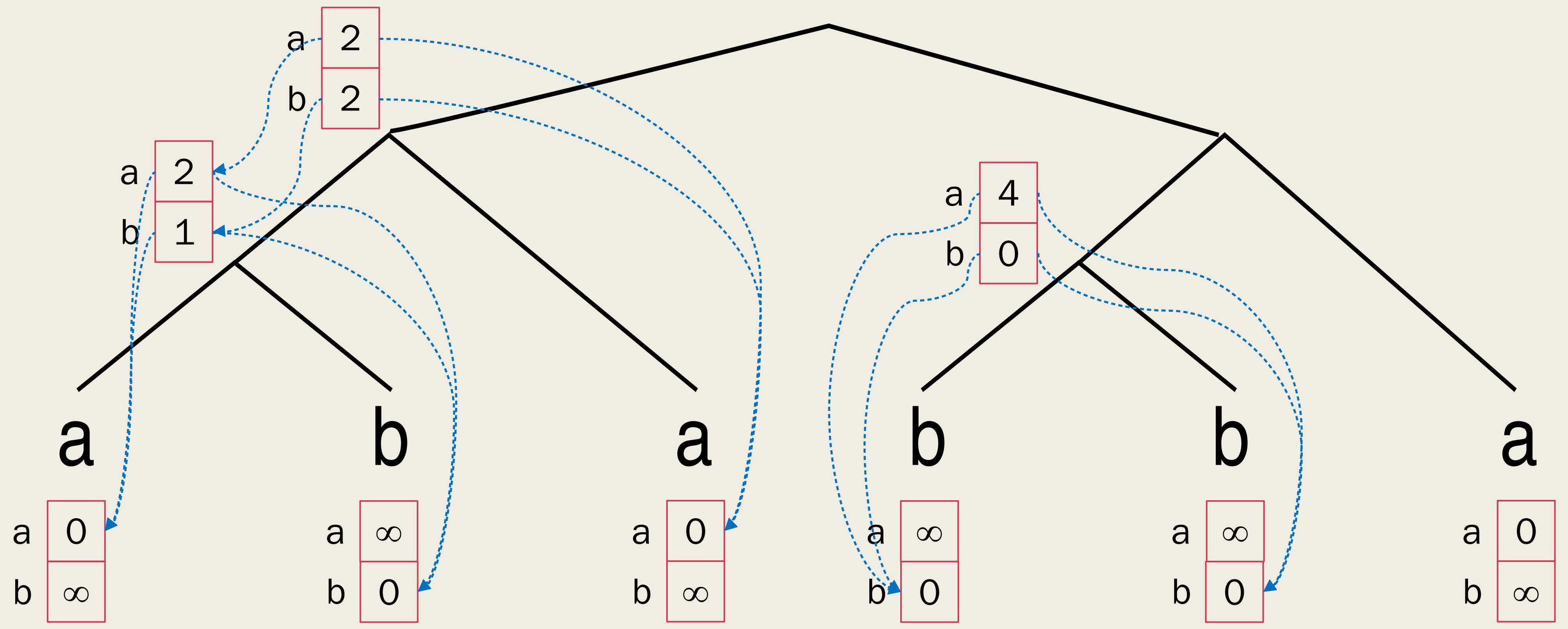
Handout 14 example

σ	a	b
a	0	2
b	1	0



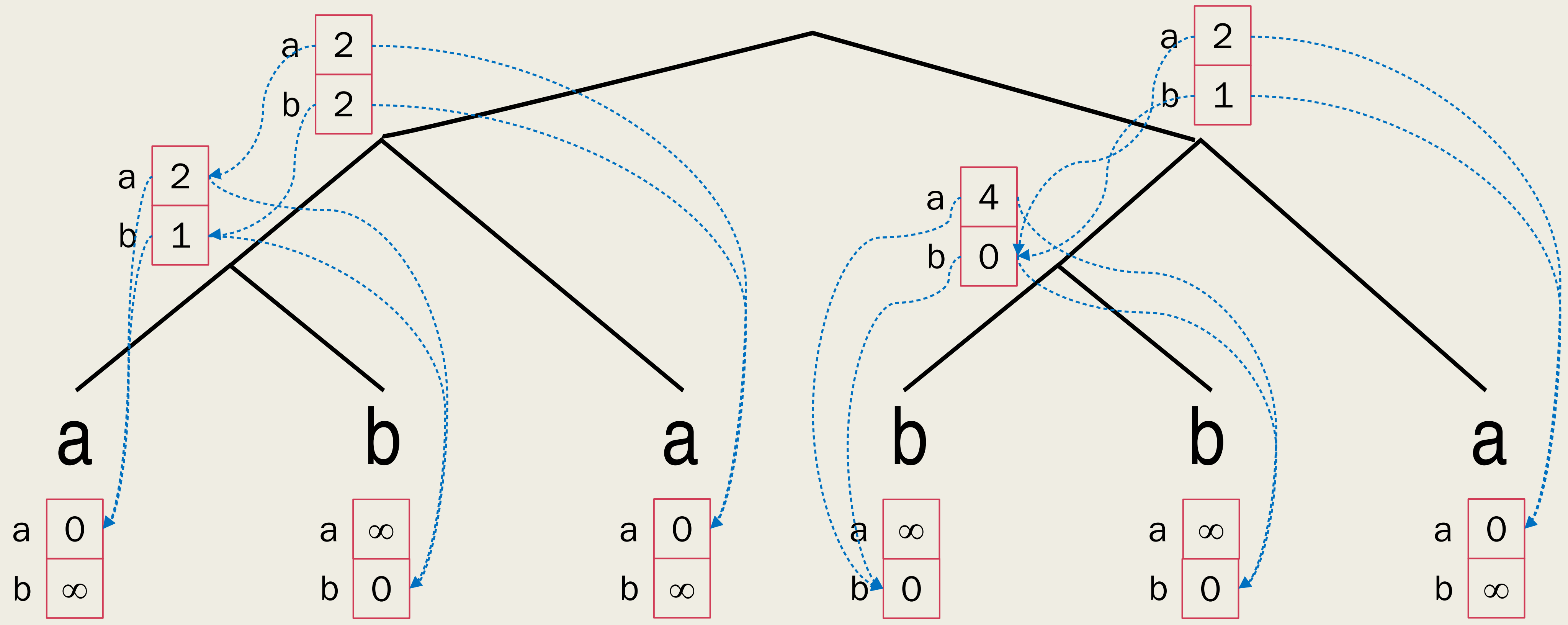
Handout 14 example

σ	a	b
a	0	2
b	1	0



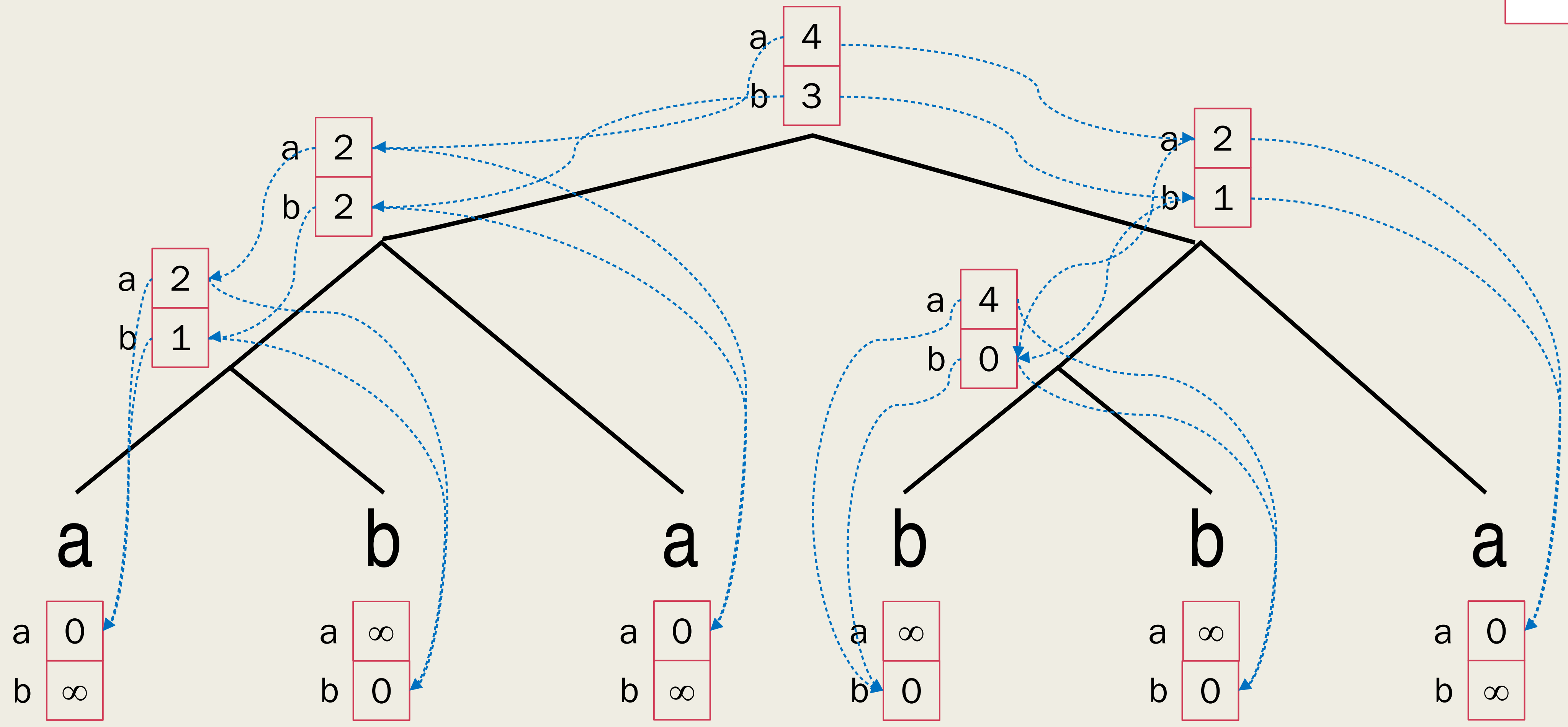
Handout 14 example

σ	a	b
a	0	2
b	1	0



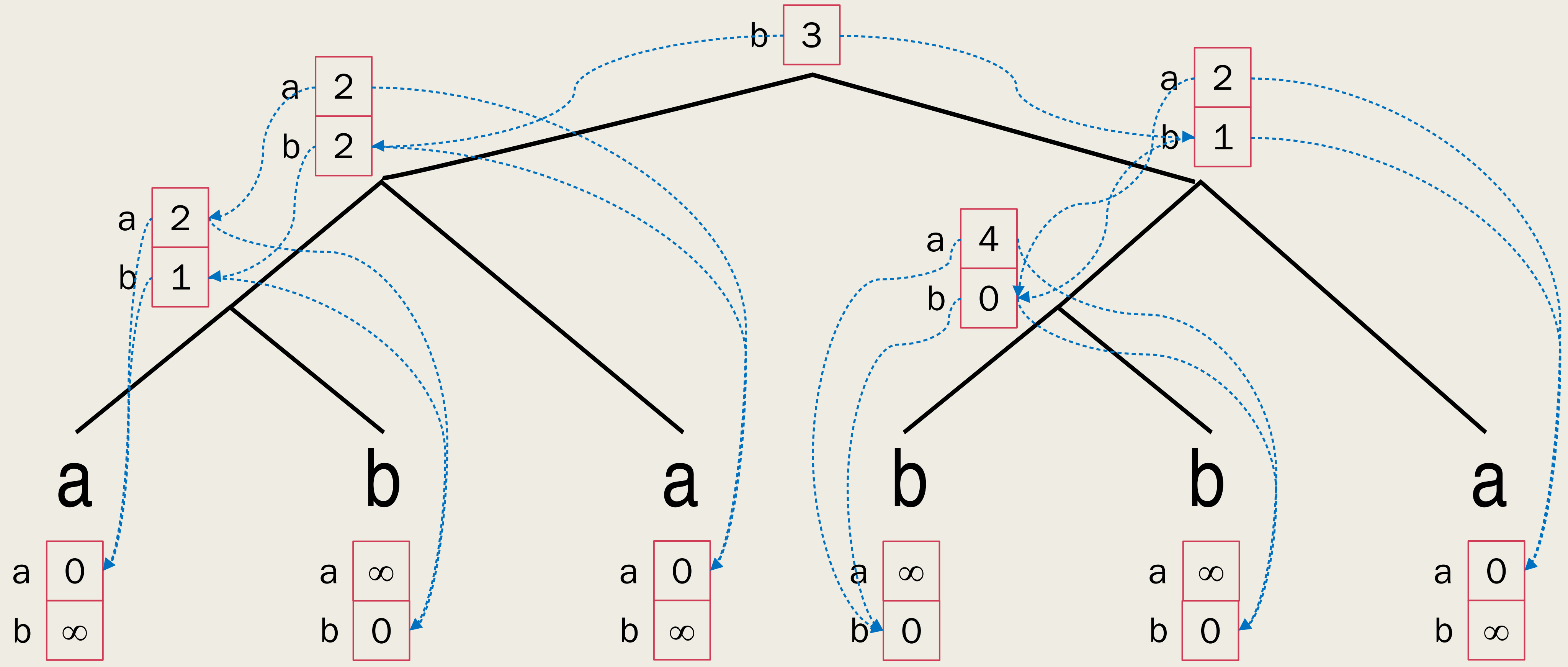
Handout 14 example

σ	a	b
a	0	2
b	1	0



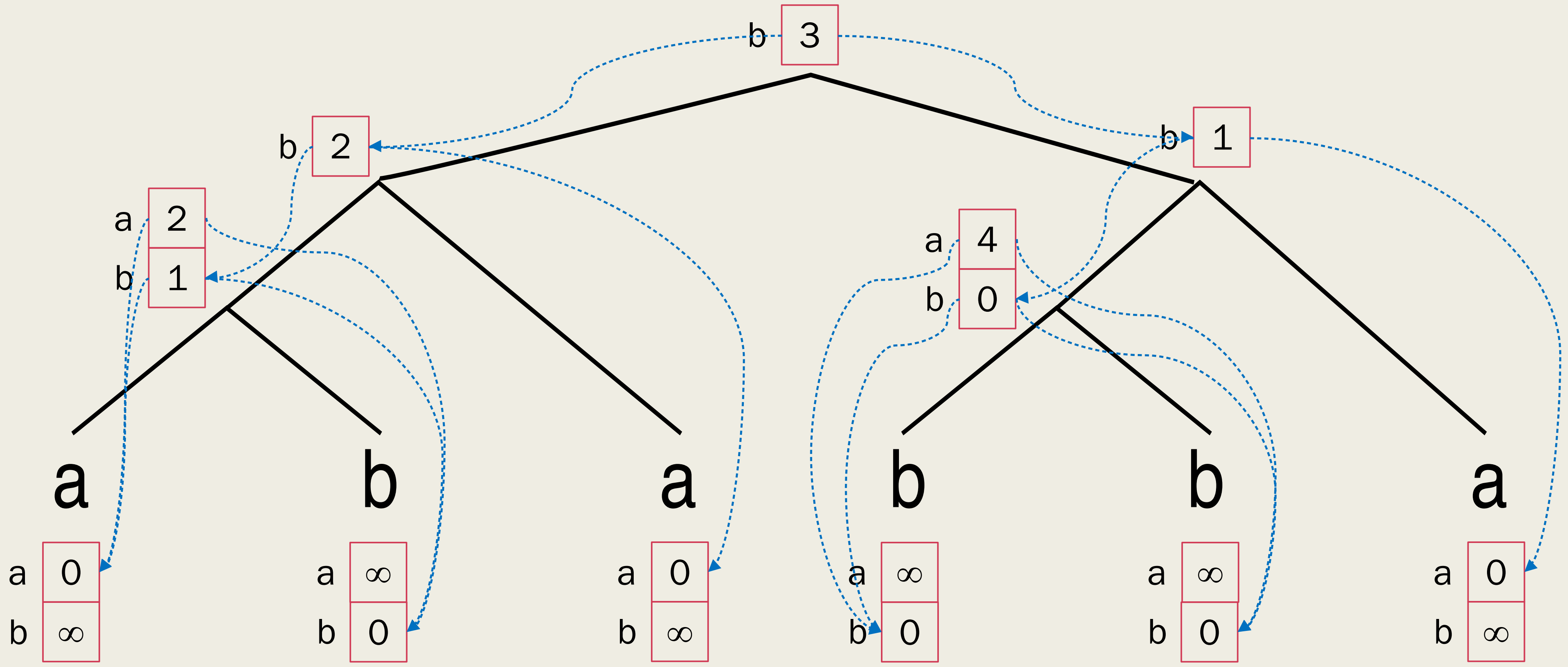
Handout 14 example

σ	a	b
a	0	2
b	1	0



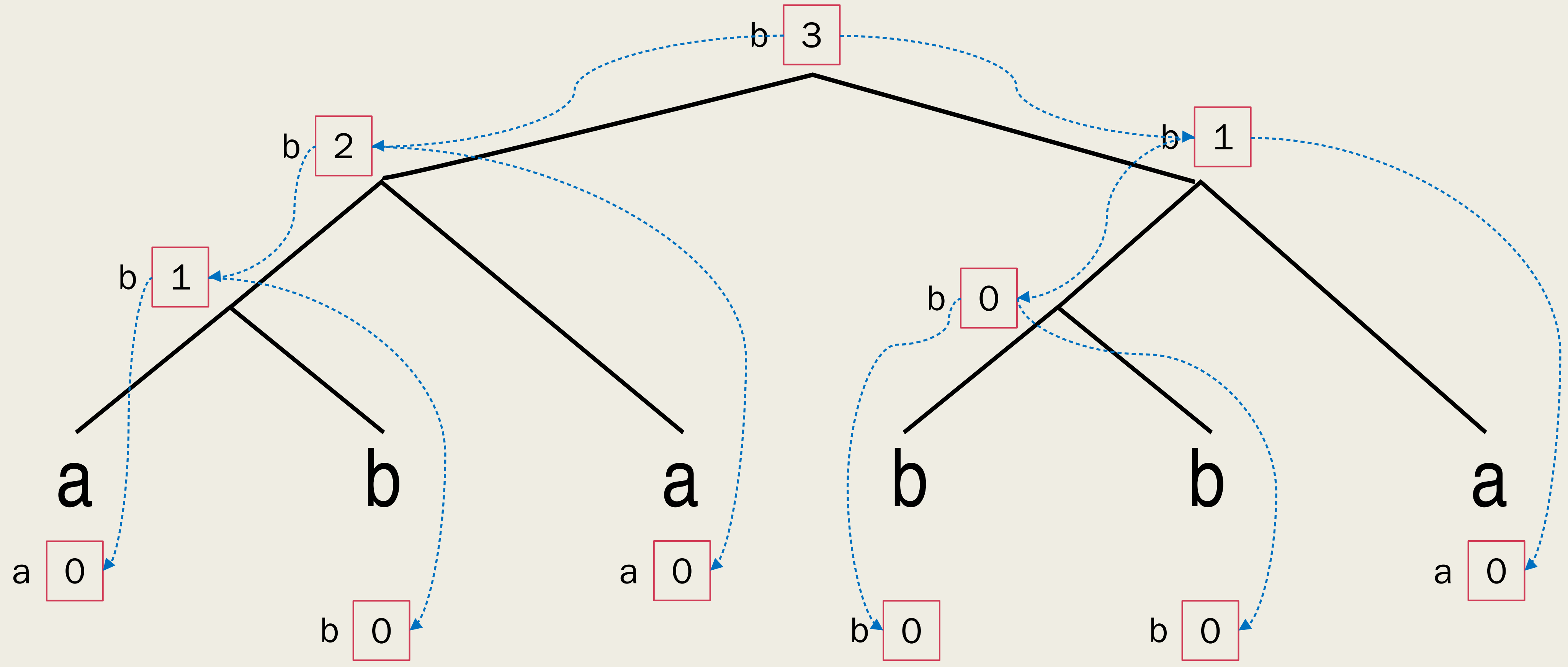
Handout 14 example

σ	a	b
a	0	2
b	1	0



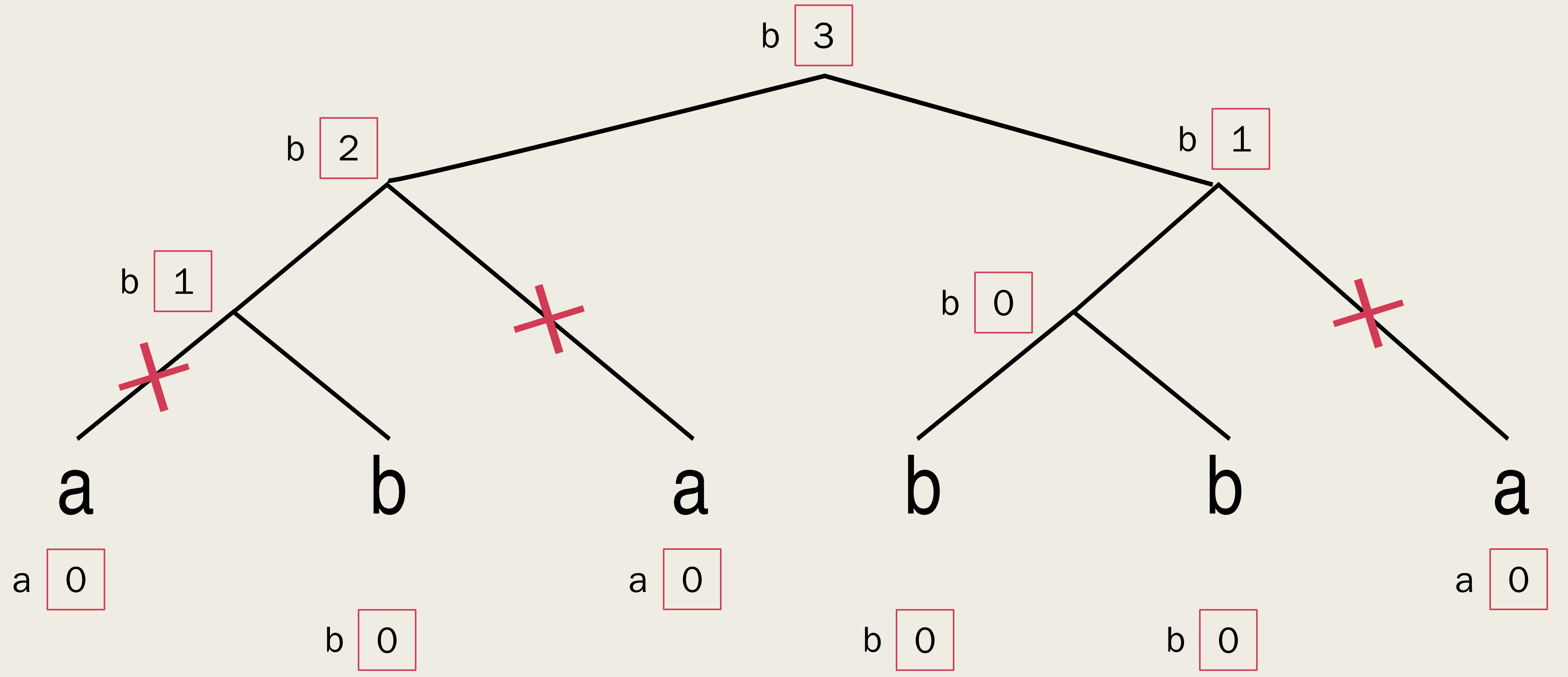
Handout 14 example

σ	a	b
a	0	2
b	1	0

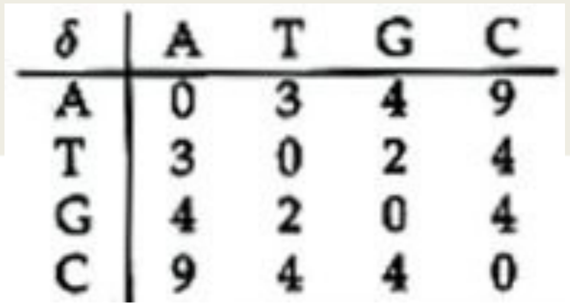


Handout 14 example

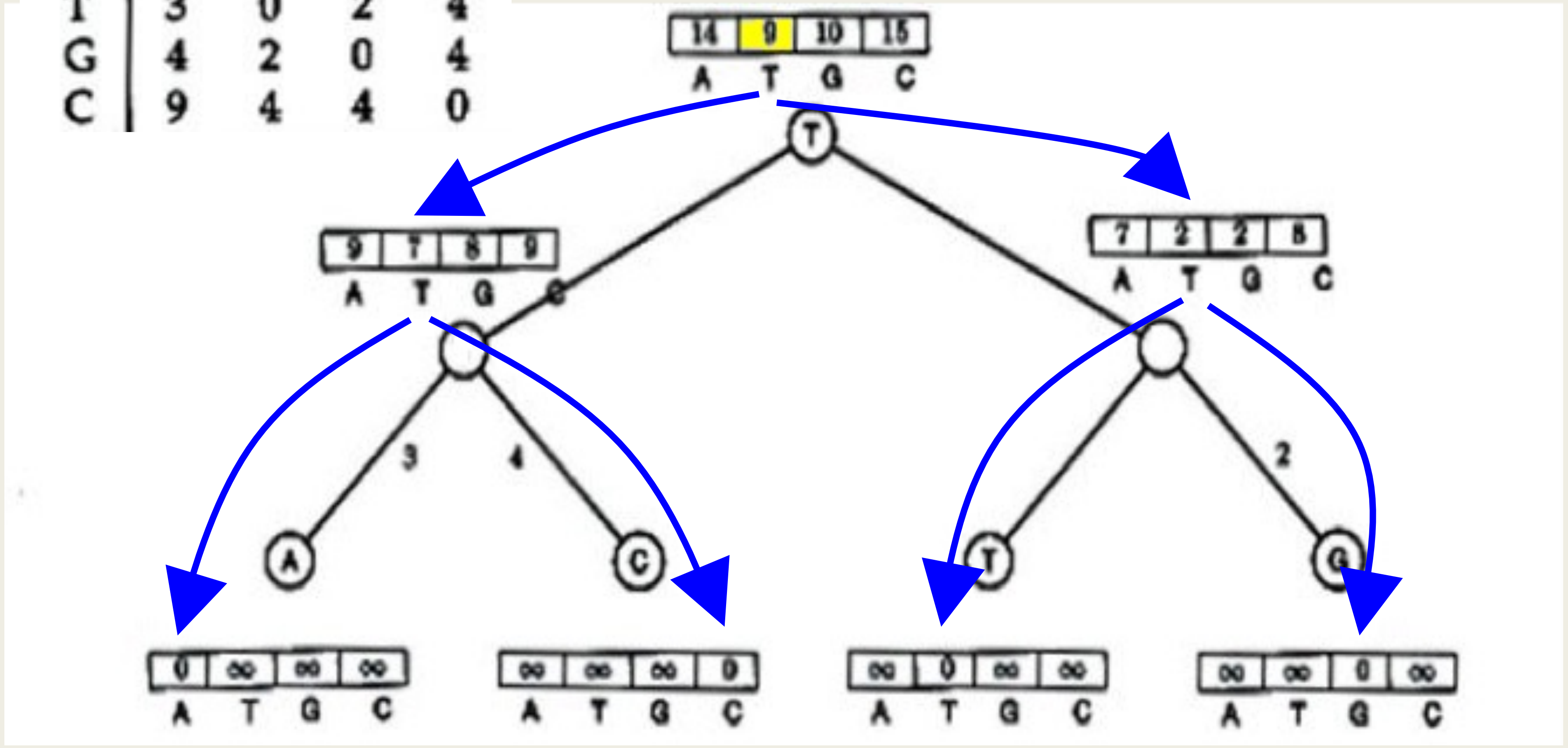
σ	a	b
a	0	2
b	1	0



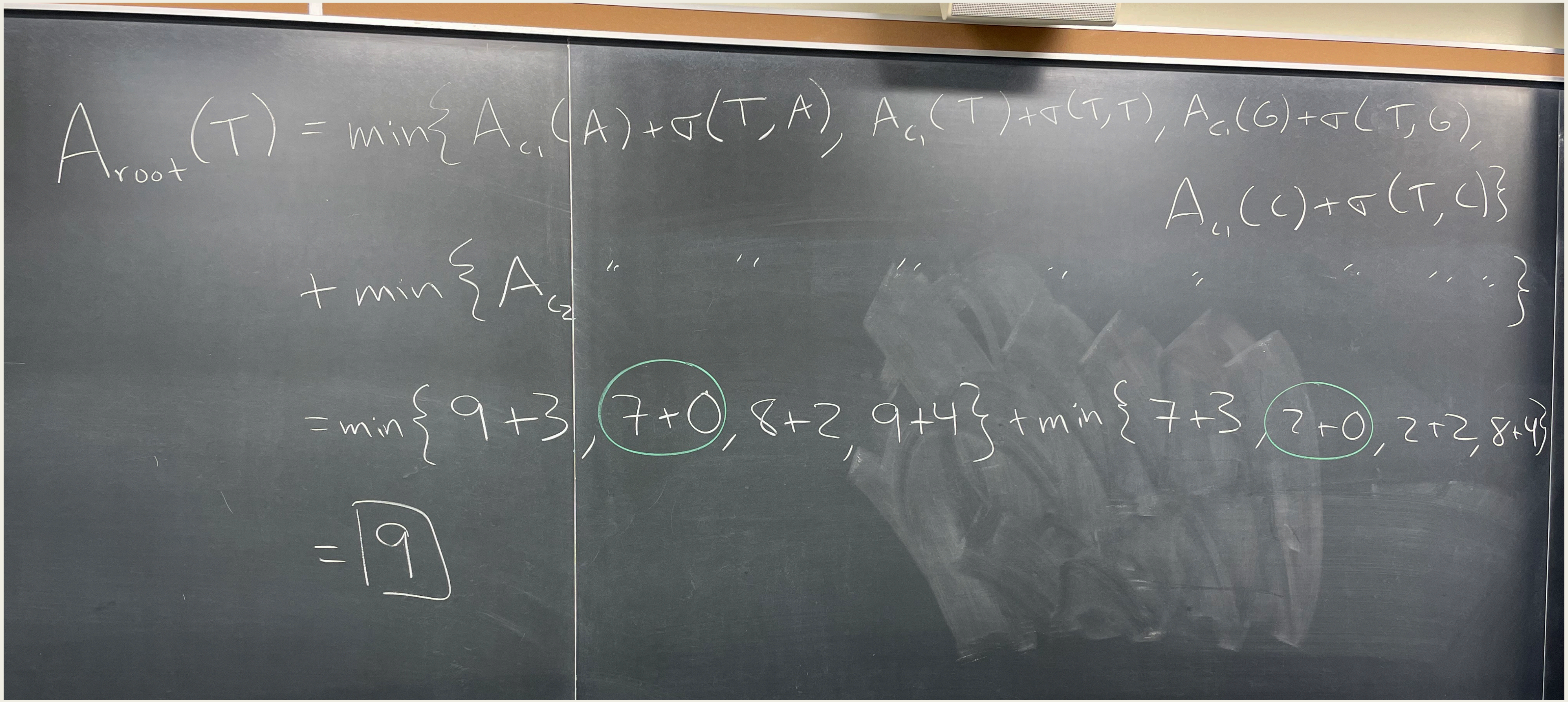
Handout 14 (second example)



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



Handout 14 (second example)



$$A_{root}(T) = \min\{A_{c_1}(A)+\sigma(T,A),\ A_{c_1}(T)+\sigma(T,T),\ A_{c_1}(G)+\sigma(T,G),\ A_{c_1}(C)+\sigma(T,C)\}$$

$$+\min\{A_{c_2} \text{ '' '' '' '' '' '' '' ''}\}$$

$$=\min\{9+3,\ 7+0,\ 8+2,\ 9+4\}+\min\{7+3,\ 2+0,\ 2+2,\ 8+4\}$$

$$=\boxed{9}$$

Runtime of Small Parsimony

Suppose there are:

n leaves (samples/taxa)

k possible states (i.e. 4 for DNA)

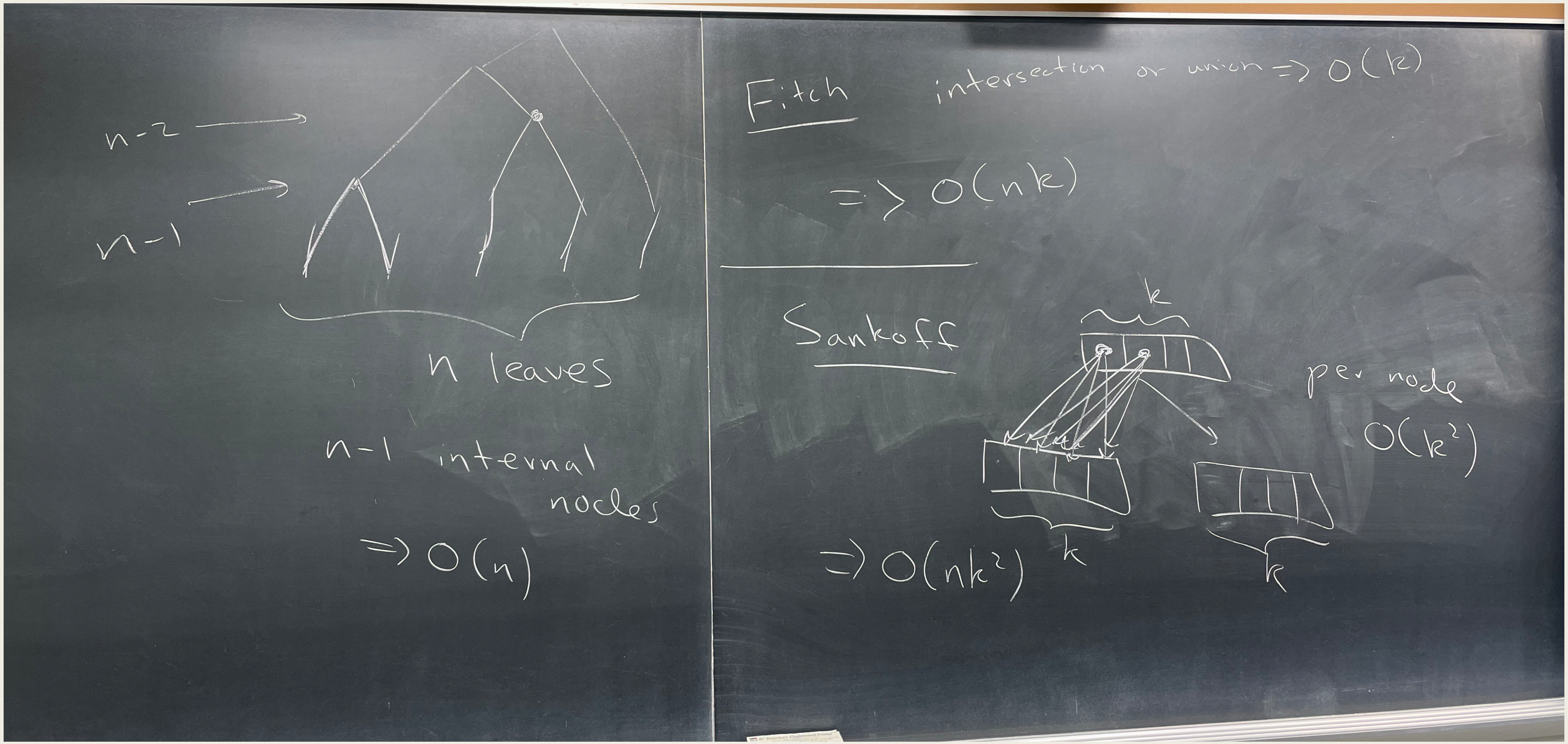
What is the complexity of Fitch's algorithm?

$O(nk)$

What is the complexity of Sankoff's algorithm?

$O(nk^2)$

Runtime of Small Parsimony



Perfect Phylogeny

Introduction

- With Fitch and Sankoff we were only looking at a single site

- When we have multiple sites, an important question is whether or not there exists a phylogeny that is "consistent" with all the sites

- By consistent we often mean that a mutation at a given site only occurs once

- If we can construct a phylogeny where each mutation only occurs once, this is called a *perfect phylogeny*

- We will study one algorithm for constructing a perfect phylogeny (or getting close if one does not exist), called *Gusfield's algorithm* (~1991)

Solving the big parsimony problem

Given a set of *m* characters, can we reconstruct the most parsimonious tree (i.e. the tree with the lowest parsimony score)?

Possible algorithm: go through every possible tree, compute the parsimony score for each character using Fitch or Sankoff, then pick the tree(s) with the lowest total score.

Problem: there are a LOT of trees:

3 taxa = 3 trees, 5 taxa = 105 trees, 10 taxa = 34,459,425 …

In fact this problem is NP-hard

Example 1

	paired fins	jaws	large dermal bones	fin rays	lungs	rasping tongue
	1	2	3	4	5	6
lamprey	0	0	0	0	0	1
shark	1	1	0	1	0	0
salmon	1	1	1	1	0	0
lizard	1	1	1	0	1	0

Input

Images: Getty, State Symbols USA, OBP, NWF

Data from “The Perfect Phylogeny Problem” by David Fernandez-Baca

Example 1

n=4 species

m=6 characters

	paired fins	jaws	large dermal bones	fin rays	lungs	rasping tongue
	1	2	3	4	5	6
lamprey	0	0	0	0	0	1
shark	1	1	0	1	0	0
salmon	1	1	1	1	0	0
lizard	1	1	1	0	1	0

Input

Data from “The Perfect Phylogeny Problem” by David Fernandez-Baca

Example 1

n=4 species

m=6 characters

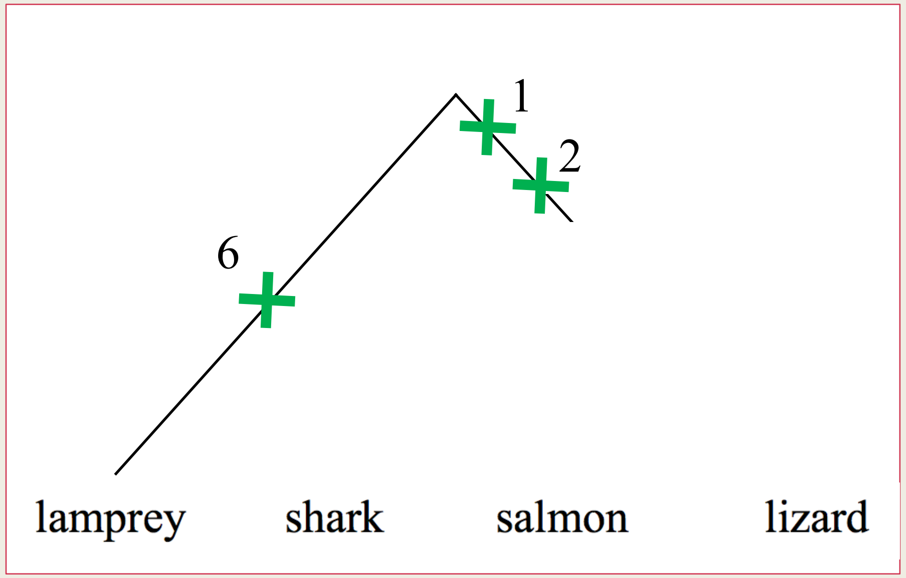
	paired fins	jaws	large dermal bones	fin rays	lungs	rasping tongue
	1	2	3	4	5	6
lamprey	0	0	0	0	0	1
shark	1	1	0	1	0	0
salmon	1	1	1	1	0	0
lizard	1	1	1	0	1	0

Input

Data from "The Perfect Phylogeny Problem" by David Fernandez-Baca

Example 1

	1 paired fins	2 jaws	3 large dermal bones	4 fin rays	5 lungs	6 rasping tongue
lamprey	0	0	0	0	0	1
shark	1	1	0	1	0	0
salmon	1	1	1	1	0	0
lizard	1	1	1	0	1	0



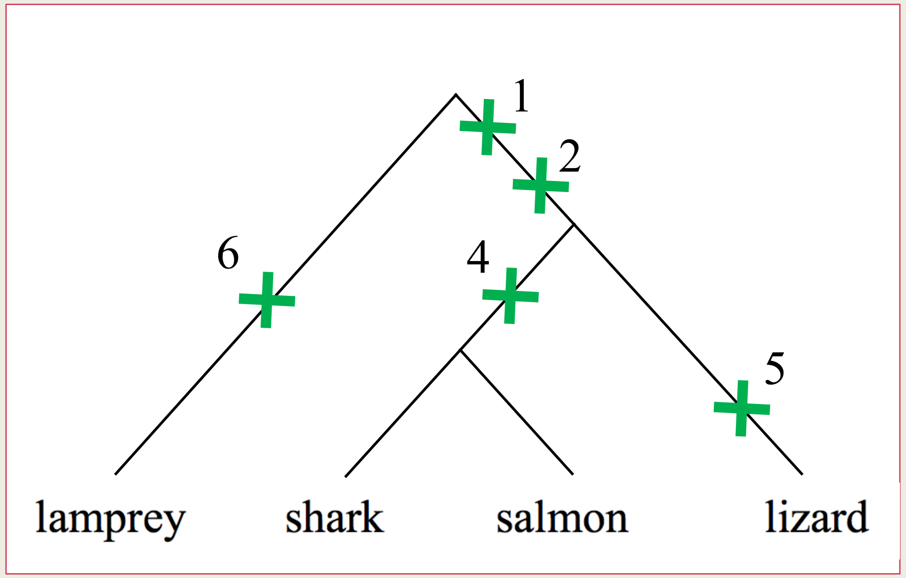
Input

Output

Data from “The Perfect Phylogeny Problem” by David Fernandez-Baca

Example 1

	paired fins	jaws	large dermal bones	fin rays	lungs	rasping tongue
	1	2	3	4	5	6
lamprey	0	0	0	0	0	1
shark	1	1	0	1	0	0
salmon	1	1	1	1	0	0
lizard	1	1	1	0	1	0



Input

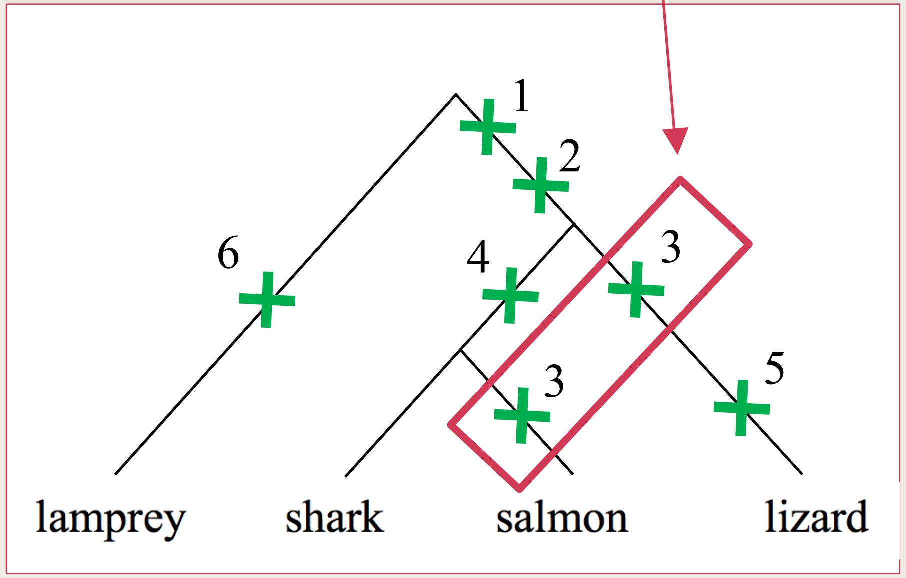
Output

Data from "The Perfect Phylogeny Problem" by David Fernandez-Baca

Example 1

	1 paired fins	2 jaws	3 large dermal bones	4 fin rays	5 lungs	6 rasping tongue
lamprey	0	0	0	0	0	1
shark	1	1	0	1	0	0
salmon	1	1	1	1	0	0
lizard	1	1	1	0	1	0

NOT a *perfect phylogeny*, character 3 evolved twice!



Input

Output

Data from "The Perfect Phylogeny Problem" by David Fernandez-Baca

The perfect phylogeny problem

- If we can construct a phylogenetic tree where each mutation only occurs once, this is called a *perfect phylogeny*

- One algorithm for constructing a perfect phylogeny (or getting close if one does not exist) is called *Gusfield's algorithm* (1991)

- Notation:
  - *n species or samples*
  - *m sites in the genome or traits/characteristics*

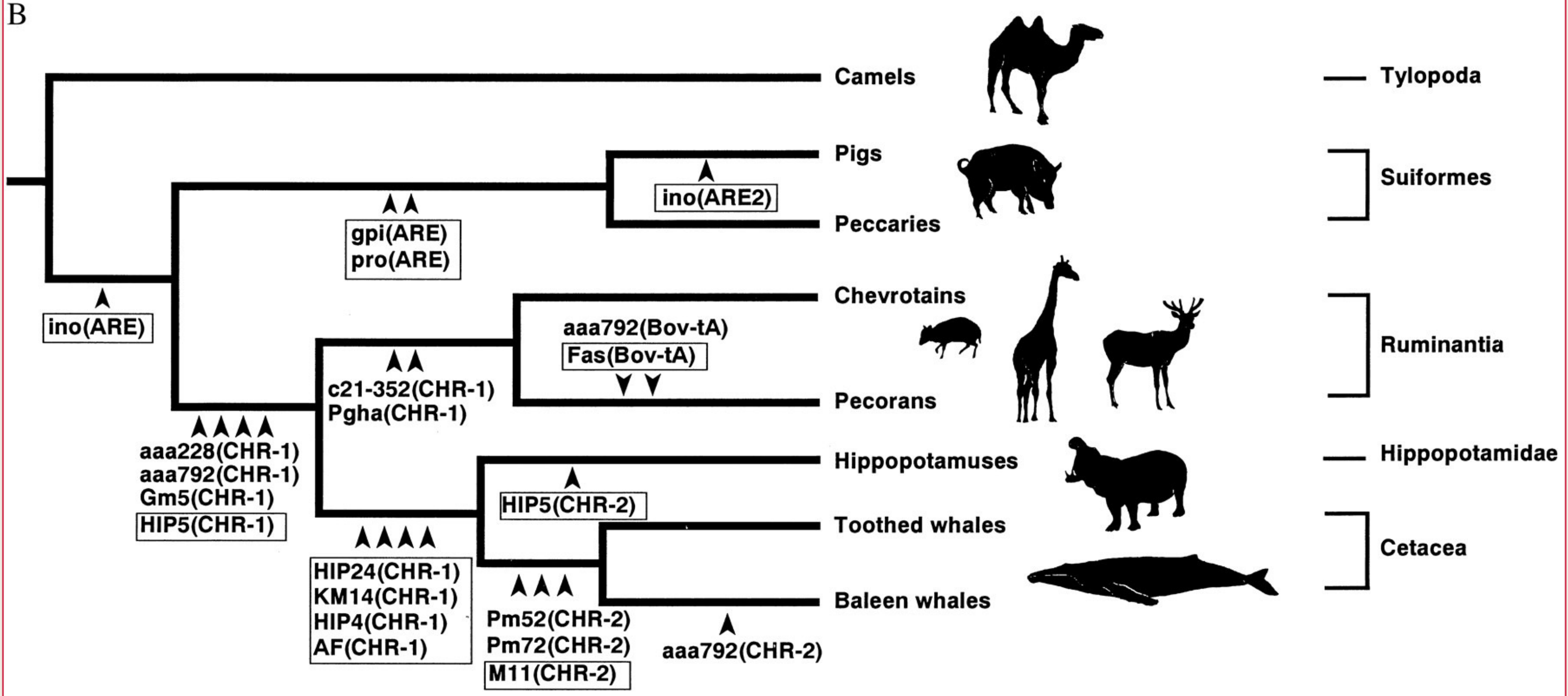
Perfect Phylogeny example

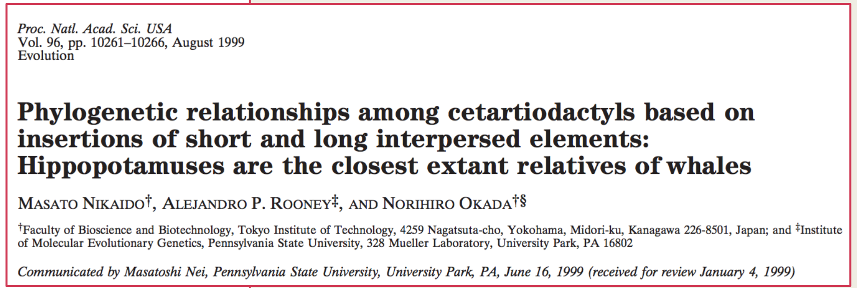
A

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Camel	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	0	0	0
Pig	0	0	0	?	0	0	0	0	?	0	0	0	?	?	0	0	?	1	1	1
Peccary	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	1	1
Chevrotain	?	0	?	?	?	?	?	?	?	1	0	?	?	?	1	1	0	?	0	0
Deer	0	0	0	0	0	0	0	1	?	1	1	1	1	1	1	?	1	1	0	0
Graffe	?	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
Sheep	0	0	0	0	0	?	0	1	1	1	1	1	1	1	1	1	1	1	0	0
Cow	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
Hippo	0	?	0	1	1	1	1	0	1	1	0	1	1	0	0	0	?	1	0	0
Humpback	1	1	1	1	1	1	1	0	1	1	0	1	1	0	0	0	?	?	0	0
Beaked	1	1	1	1	1	1	1	0	?	1	0	1	1	0	0	0	?	1	0	0

1	PM52	11	aaa792(Bov tA)
2	PM72	12	Gm5
3	M11	13	HIP5(CHR-1)
4	HIP24	14	HIP5(Bov A)
5	KM14	15	c21-352
6	HIP4	16	pgha
7	AF(CHR-1)	17	Fas
8	AF(MER)	18	INO
9	aaa228	19	pgi
10	aaa792(CHR-1)	20	pro

B




Proc. Natl. Acad. Sci. USA
Vol. 96, pp. 10261–10266, August 1999
Evolution

Phylogenetic relationships among cetartiodactyls based on insertions of short and long interpersed elements: Hippopotamuses are the closest extant relatives of whales

MASATO NIKAIDO[†], ALEJANDRO P. ROONEY[‡], AND NORIHIRO OKADA[†§]

[†]Faculty of Bioscience and Biotechnology, Tokyo Institute of Technology, 4259 Nagatsuta-cho, Yokohama, Midori-ku, Kanagawa 226-8501, Japan; and [‡]Institute of Molecular Evolutionary Genetics, Pennsylvania State University, 328 Mueller Laboratory, University Park, PA 16802

Communicated by Masatoshi Nei, Pennsylvania State University, University Park, PA, June 16, 1999 (received for review January 4, 1999)

Gusfield’s Algorithm

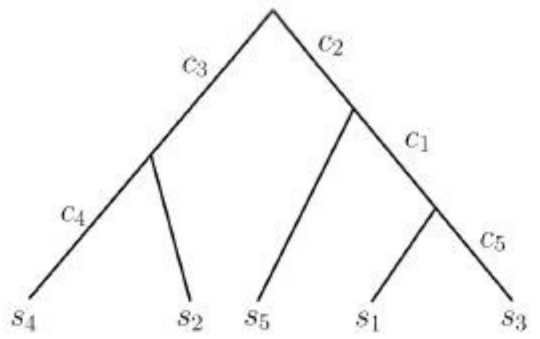
Perfect phylogeny

- Each mutation happens exactly once
- Mutations can never revert (you can only go 0->1, not back)
- Biologically; every character is absent in the ancestor, and evolves exactly once

m characters

n Taxa

$$M_1' = \begin{array}{c|ccccc} & c_2 & c_1 & c_3 & c_5 & c_4 \\ \hline s_1 & 1 & 1 & 0 & 0 & 0 \\ s_2 & 0 & 0 & 1 & 0 & 0 \\ s_3 & 1 & 1 & 0 & 1 & 0 \\ s_4 & 0 & 0 & 1 & 0 & 1 \\ s_5 & 1 & 0 & 0 & 0 & 0 \\ \hline \end{array} \longrightarrow \mathcal{T}_1 =$$



Gusfield's algorithm

1) Sort the columns high-to-low, treating them as binary numbers

	C1	C2	C3	C4	C5
S1	0	0	1	1	0
S2	1	0	0	0	0
S3	0	1	1	1	0
S4	1	0	0	0	1
S5	0	0	1	0	0

	C3	C4	C1	C2	C5
S1	1	1	0	0	0
S2	0	0	1	0	0
S3	1	1	0	1	0
S4	0	0	1	0	1
S5	1	0	0	0	0

Note 10101 (21) > 10100 (20) > 01010 (10) > 00100 (4) > 00010 (2)

Gusfield’s algorithm

2) Write out mutation number strings (with terminating $)

	C3	C4	C1	C2	C5
S1	1	1	0	0	0
S2	0	0	1	0	0
S3	1	1	0	1	0
S4	0	0	1	0	1
S5	1	0	0	0	0

S1	34$
S2	1$
S3	342$
S4	15$
S5	3$

Gusfield's algorithm

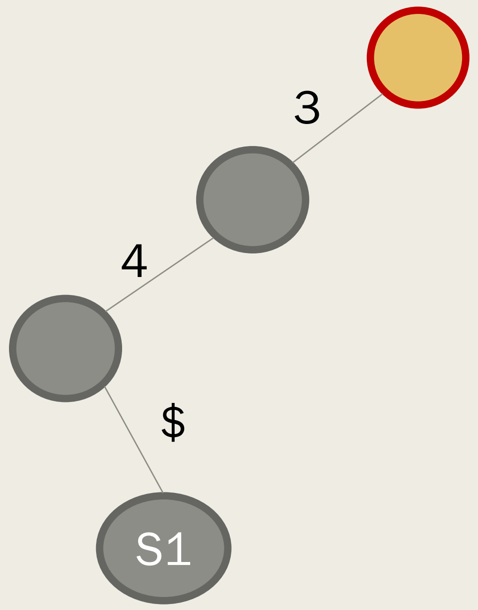
3) Build a tree – just like a suffix tree with taxa as positions!

S1	34$
S2	1$
S3	342$
S4	15$
S5	3$

Gusfield's algorithm

3) Build a tree from root to leaves, with mutations on the edges

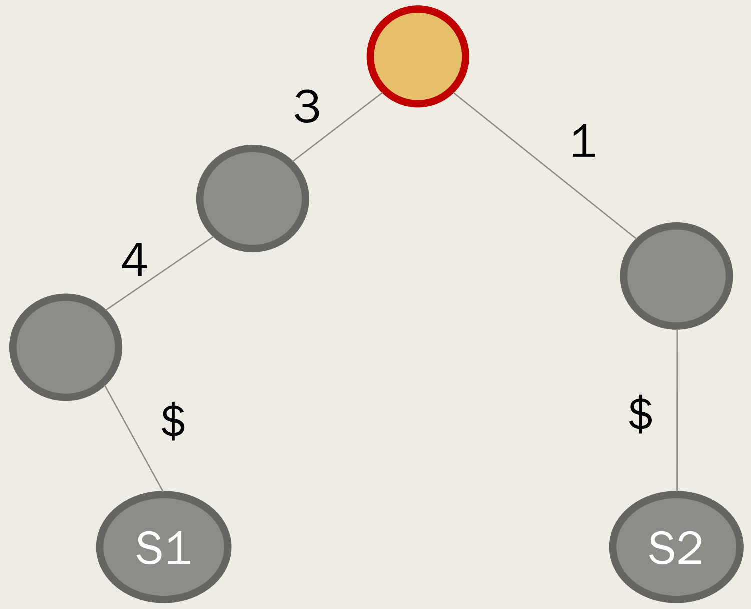
S1	34$
S2	1$
S3	342$
S4	15$
S5	3$



Gusfield’s algorithm

3) Build a tree from root to leaves, with mutations on the edges

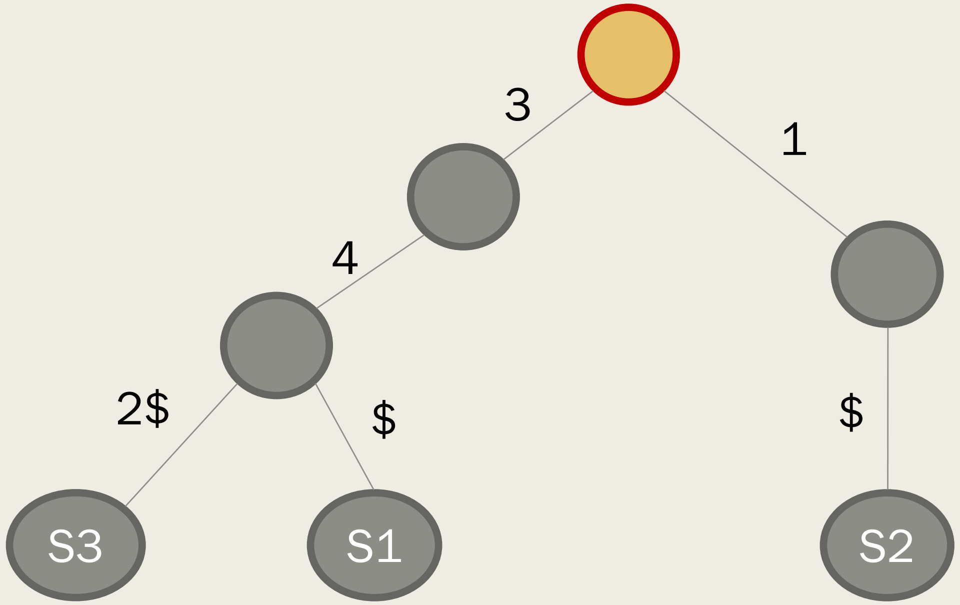
S1	34$
S2	1$
S3	342$
S4	15$
S5	3$



Gusfield's algorithm

3) Build a tree from root to leaves, with mutations on the edges

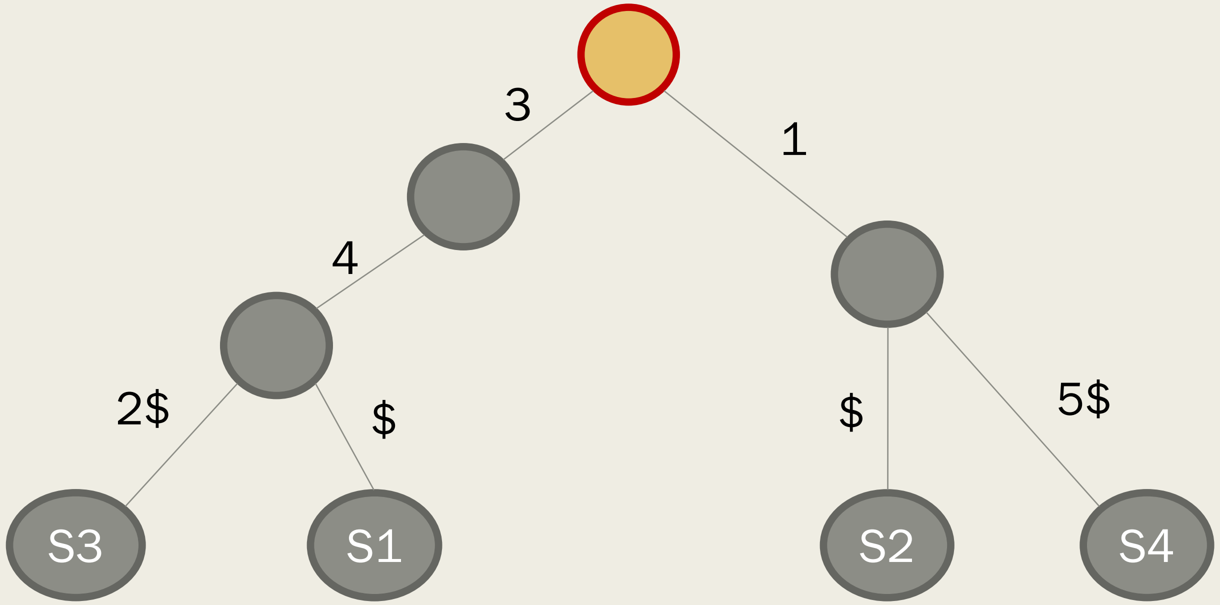
S1	34$
S2	1$
S3	342$
S4	15$
S5	3$



Gusfield’s algorithm

3) Build a tree from root to leaves, with mutations on the edges

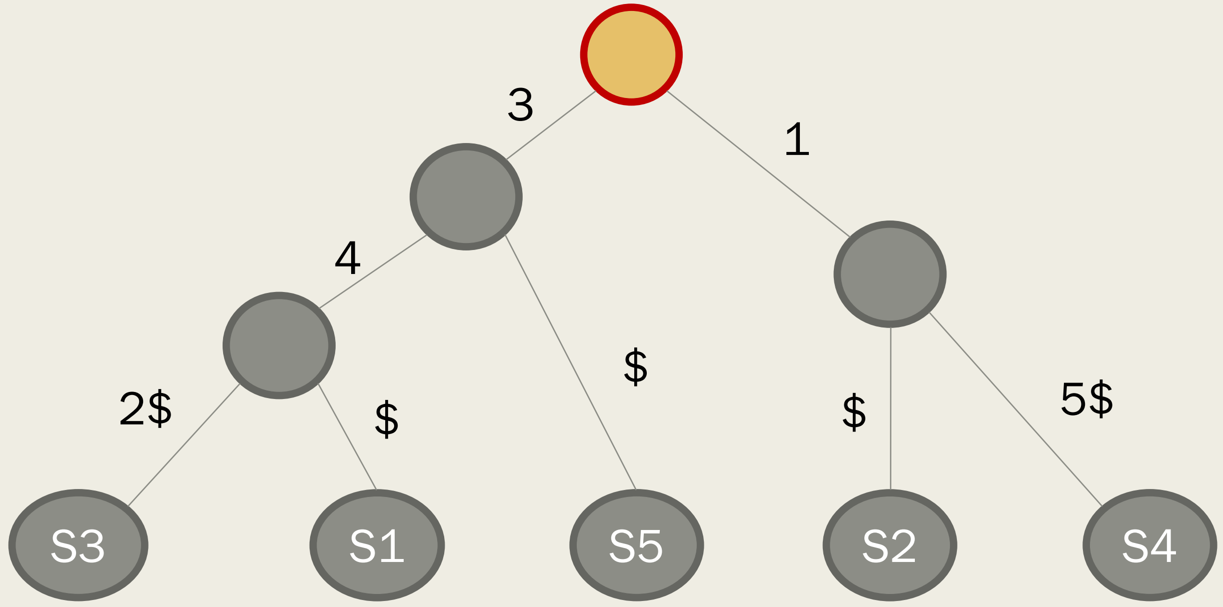
S1	34$
S2	1$
S3	342$
S4	15$
S5	3$



Gusfield's algorithm

3) Build a tree from root to leaves, with mutations on the edges

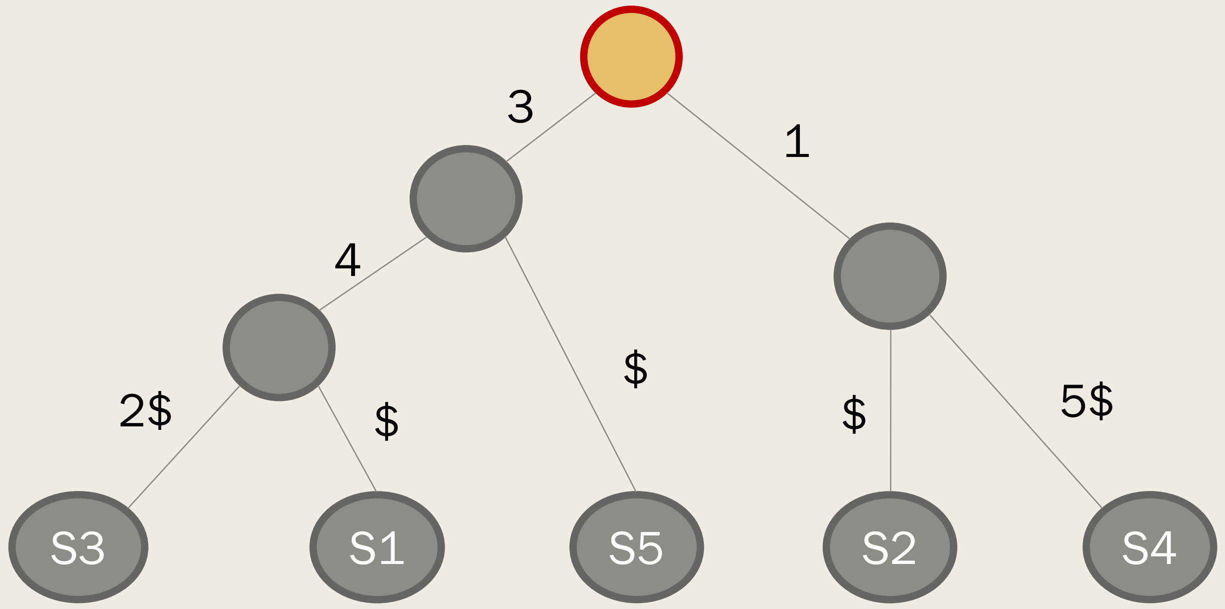
S1	34$
S2	1$
S3	342$
S4	15$
S5	3$



Gusfield's algorithm

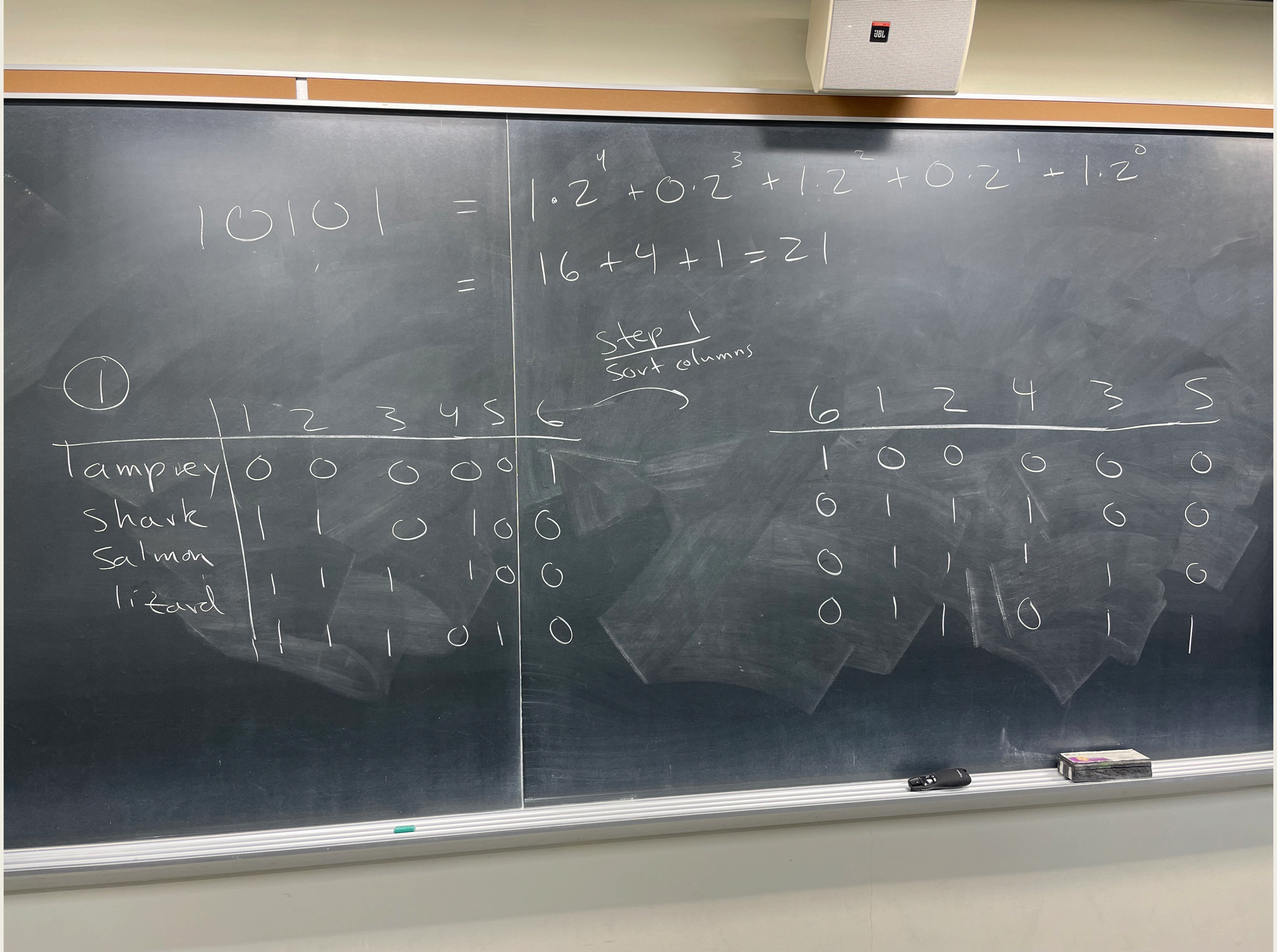
3) Build a tree from root to leaves, with mutations on the edges

S1	34$
S2	1$
S3	342$
S4	15$
S5	3$



Note: Does not have to be a binary tree (in what situation?)

Board example



$$10101 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$= 16 + 4 + 1 = 21$$

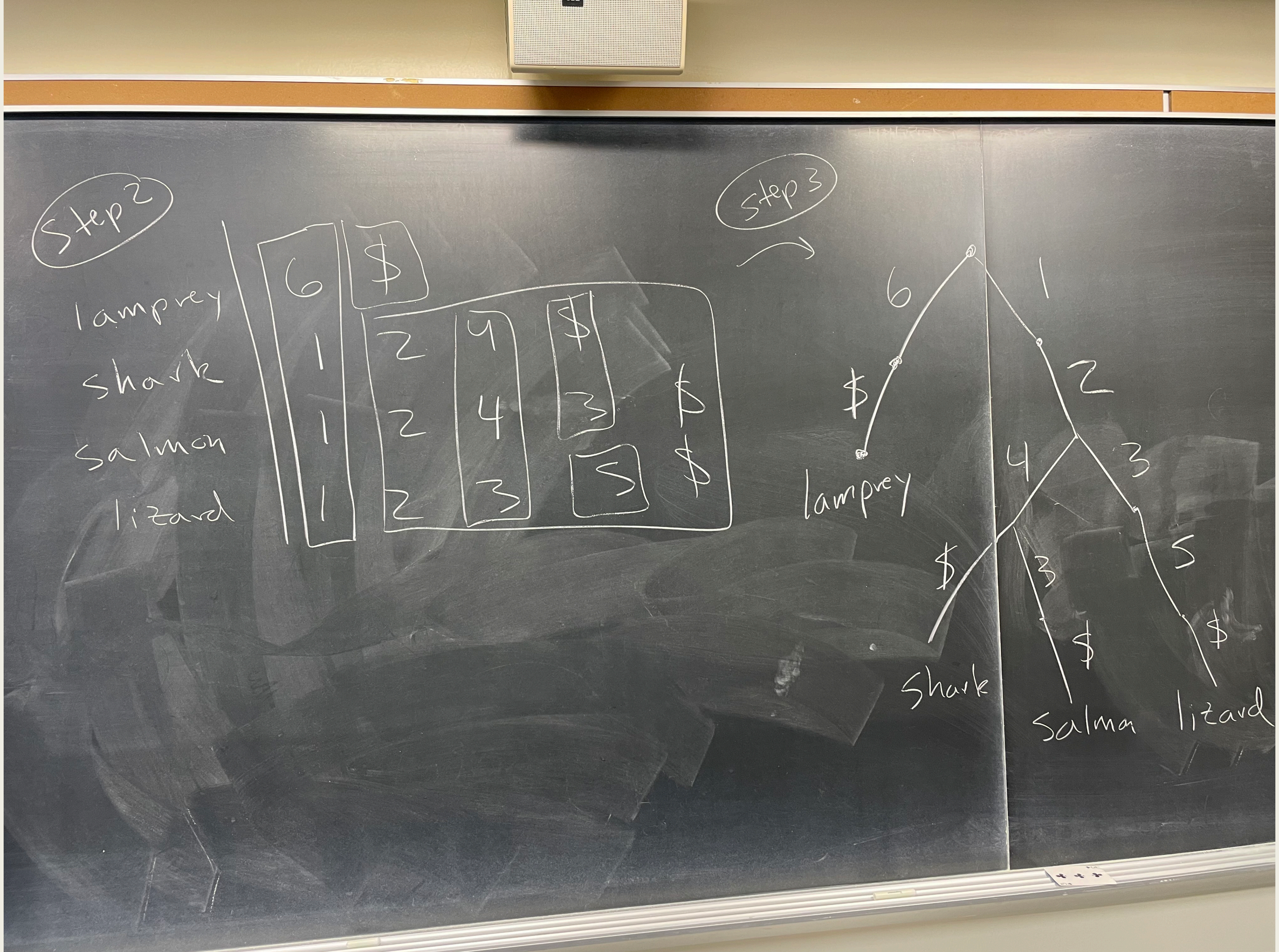
①

	1	2	3	4	5	6
lamprey	0	0	0	0	0	1
shark	1	1	0	1	0	0
Salmon	1	1	1	1	0	0
lizard	1	1	1	0	1	0

Step 1

Sort columns

6	1	2	4	3	5
1	0	0	0	0	0
0	1	1	1	0	0
0	1	1	1	1	0
0	1	1	0	1	1

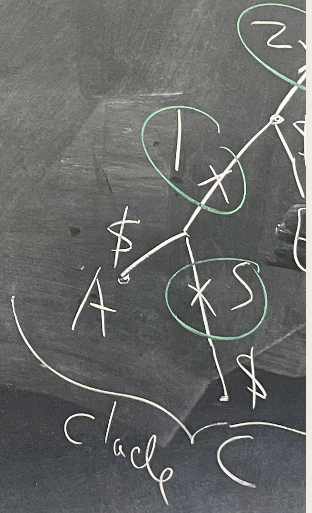


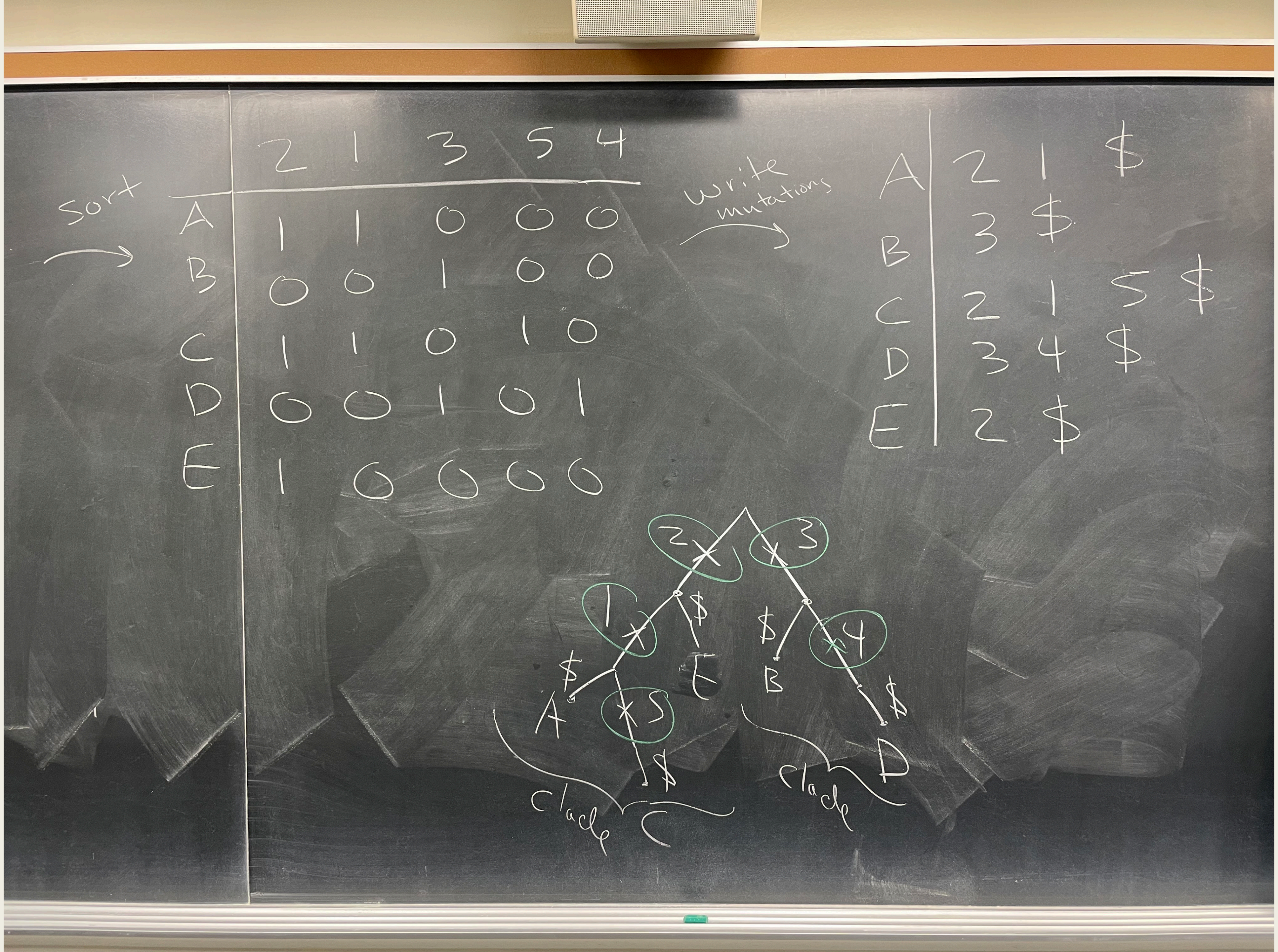
Step 2

lamprey	6	$			
shark	1	2	4	$	
salmon	1	2	4	3	$
lizard	1	2	3	5	$

②

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline A & 1 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 1 & 0 & 0 & 1 \\ D & 0 & 0 & 1 & 1 & 0 \\ E & 0 & 1 & 0 & 0 & 0 \end{array} \xrightarrow{\text{Sort}} \begin{array}{c|ccccc} & 2 & 1 & 3 & 5 & 4 \\ \hline A & 1 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 1 & 0 & 1 & 0 \\ D & 0 & 0 & 1 & 0 & 1 \\ E & 1 & 0 & 0 & 0 & 0 \end{array}$$



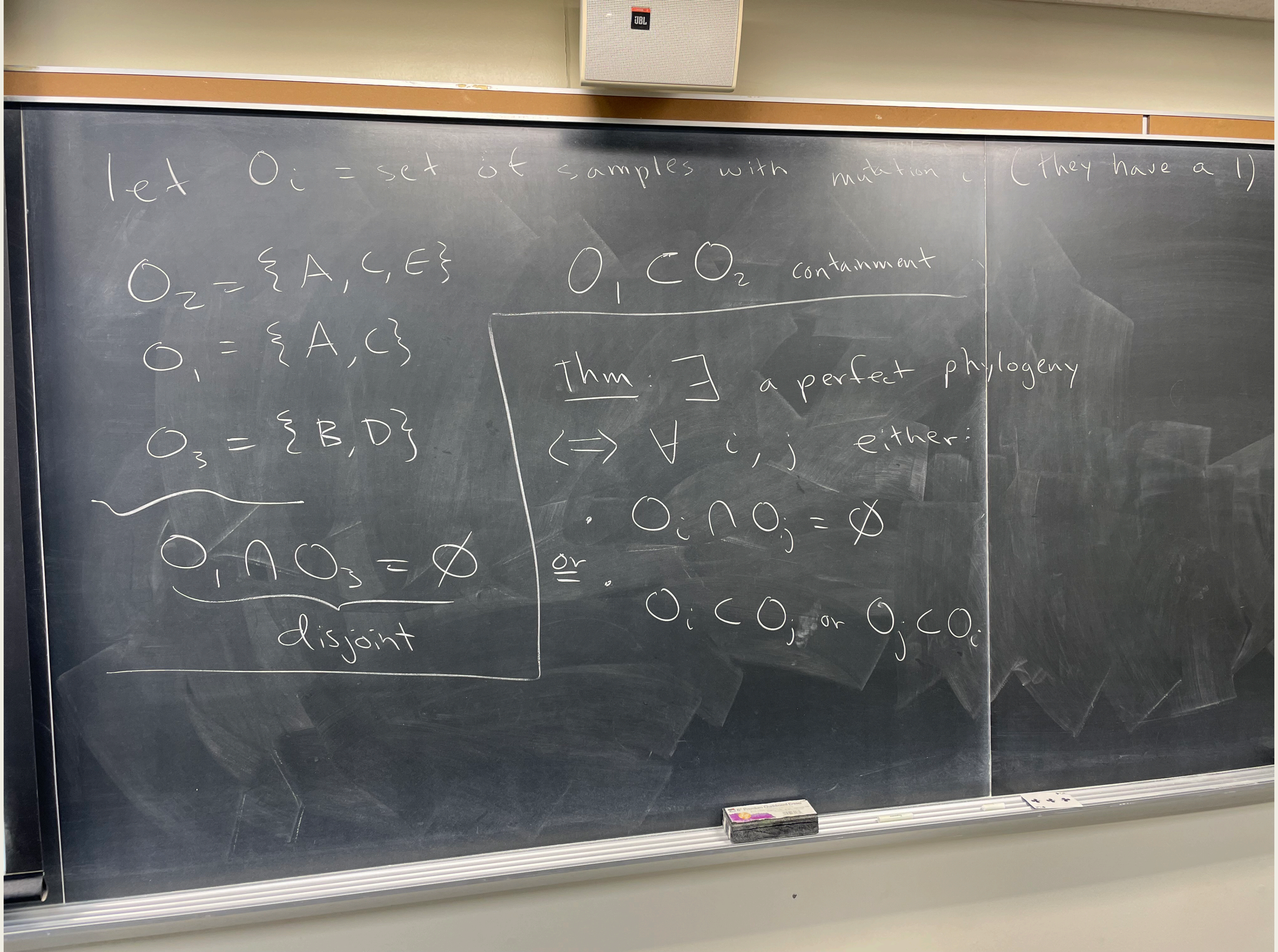


Sort →

	2	1	3	5	4
A	1	1	0	0	0
B	0	0	1	0	0
C	1	1	0	1	0
D	0	0	1	0	1
E	1	0	0	0	0

write mutations →

A	2	1	$	
B	3	$		
C	2	1	5	$
D	3	4	$	
E	2	$		



let O_i = set of samples with mutation i (they have a 1)

$O_2 = \{A, C, E\}$

$O_1 = \{A, C\}$

$O_3 = \{B, D\}$

$\underbrace{O_1 \cap O_3 = \emptyset}_{\text{disjoint}}$

$O_1 \subset O_2$ containment

<u>Thm</u>: $\exists$ a perfect phylogeny

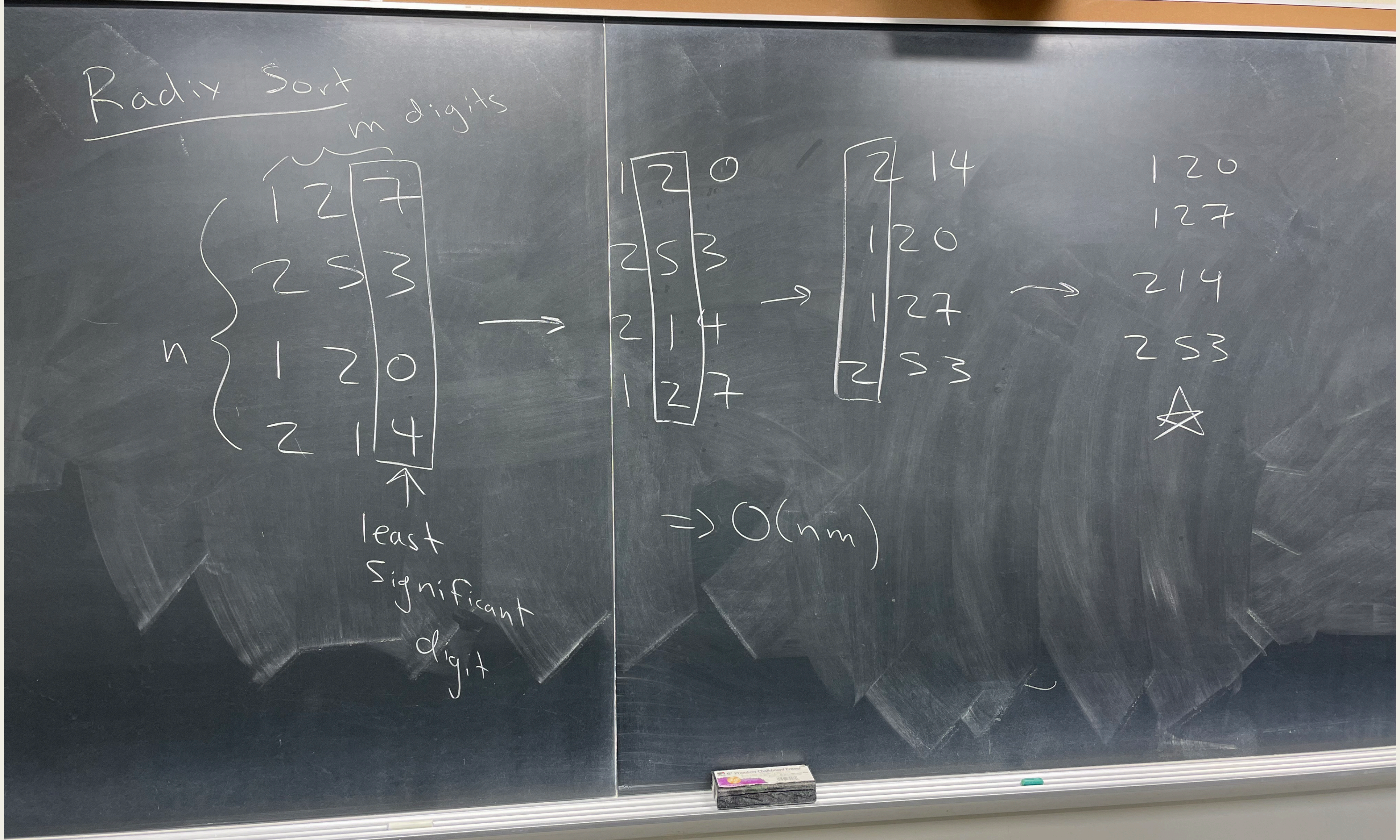
$\Longleftrightarrow \forall\ i, j$ either:

- $O_i \cap O_j = \emptyset$

<u>or</u>

- $O_i \subset O_j$ or $O_j \subset O_i$

Radix Sort



Radix Sort

m digits

$$n\left\{\begin{array}{c} 127 \\ 253 \\ 120 \\ 214 \end{array}\right. \longrightarrow \begin{array}{c} 120 \\ 253 \\ 214 \\ 127 \end{array} \longrightarrow \begin{array}{c} 214 \\ 120 \\ 127 \\ 253 \end{array} \longrightarrow \begin{array}{c} 120 \\ 127 \\ 214 \\ 253 \end{array}$$

★

least significant digit

$\Rightarrow O(nm)$

Radix sort columns high to low

Handout 15:
Example 2

1	2	3	4	5
1	1	0	0	0
0	0	1	0	0
1	1	0	0	1
0	0	1	1	0
0	1	0	0	0

2	1	3	4	5
1	1	0	0	0
0	0	1	0	0
1	1	0	0	1
0	0	1	1	0
1	0	0	0	0

3	4	2	1	5
0	0	1	1	0
1	0	0	0	0
0	0	1	1	1
1	1	0	0	0
0	0	1	0	0

2	1	5	3	4
1	1	0	0	0
0	0	0	1	0
1	1	1	0	0
0	0	0	1	1
1	0	0	0	0

3	2	1	5	4
0	1	1	0	0
1	0	0	0	0
0	1	1	1	0
1	0	0	0	1
0	1	0	0	0

2	1	3	5	4
1	1	0	0	0
0	0	1	0	0
1	1	0	1	0
0	0	1	0	1
1	0	0	0	0

Recap perfect phylogeny

- If we can construct a phylogeny where each mutation only occurs once (i.e. no convergence evolution), this is called a *perfect phylogeny*

- We will study one algorithm for constructing a perfect phylogeny (or getting close if one does not exist), called *Gusfield's algorithm* (~1991)

- Key assumption: ancestral state is all zeros (we will see how to relax this)

Recap perfect phylogeny

- If we can construct a phylogeny where each mutation only occurs once (i.e. no convergence evolution), this is called a *perfect phylogeny*

- We will study one algorithm for constructing a perfect phylogeny (or getting close if one does not exist), called *Gusfield's algorithm* (~1991)

- Key assumption: ancestral state is all zeros (we will see how to relax this)

- Notation:
  - *entire matrix of characters is often called M*
  - *O_i is the set of samples with character i*
  - *n samples (taxa) and m characters (sites or traits/characteristics)*

Observations so far…

- Theorem: there exists a perfect phylogeny if and only if for all pairs of characters *i*,*j*, either:
  - O_i *and* O_j *are disjoint* ($O_i \cap O_j = \emptyset$), or
  - *One contains the other* ($O_j \subset O_i$ *or* $O_j \supset O_i$)

Observations so far…

- Theorem: there exists a perfect phylogeny if and only if for all pairs of characters *i,j*, either:
  - *O_i and O_j are disjoint ($O_i \cap O_j = \emptyset$)*, or
  - *One contains the other ($O_j \subset O_i$ or $O_j \supset O_i$)*

- If $O_i \supset O_j$, then character *i* occurred more anciently than character *j*

Observations so far…

- Theorem: there exists a perfect phylogeny if and only if for all pairs of characters *i,j*, either:
  - *O_i and O_j are disjoint ($O_i \cap O_j = \emptyset$)*, or
  - *One contains the other ($O_j \subset O_i$ or $O_j \supset O_i$)*

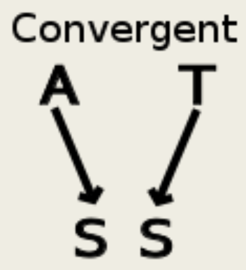
- If $O_i \supset O_j$, then character *i* occurred more anciently than character *j*

- If column *i* > column *j* as binary numbers, then either
  - *$O_i \cap O_j = \emptyset$ (disjoint), or*
  - *$O_i \supset O_j$ (i contains j)*

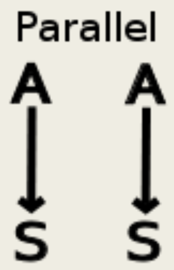
Thinking deeper about parsimony…

Types of evolution

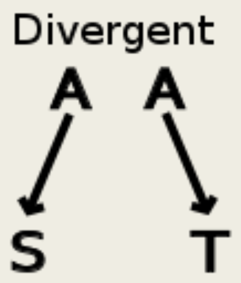
- Convergent evolution: distantly related species that develop the same characteristic (often abbreviated character) independently



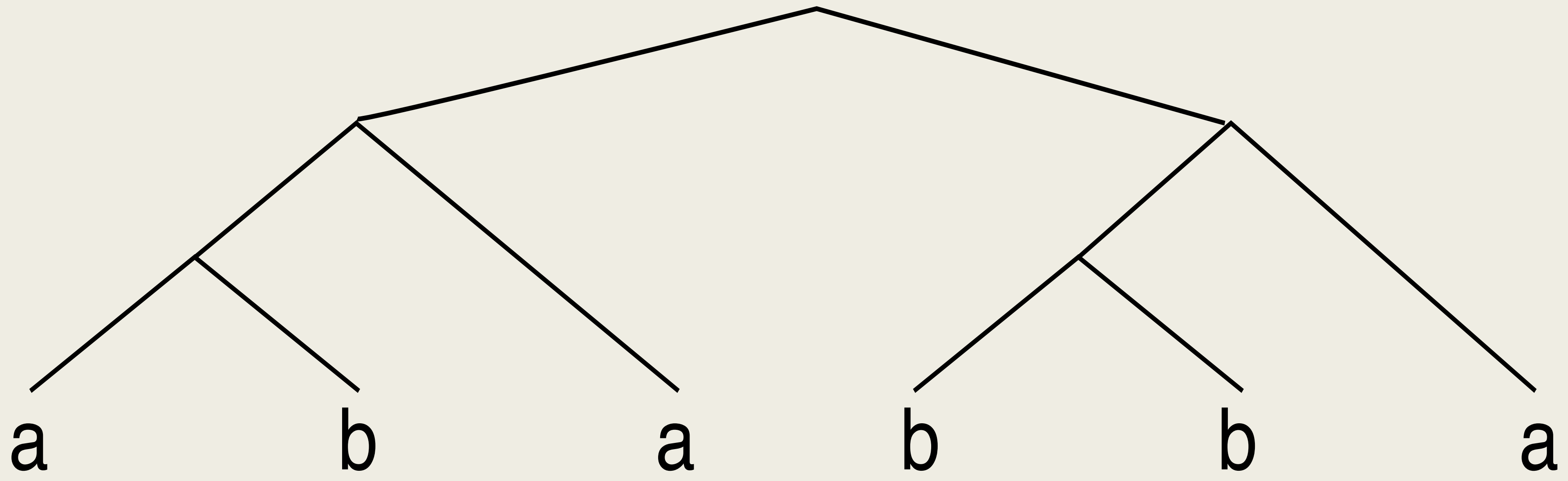
- Parallel evolution: similar species that independently evolve similar characters in parallel



- Divergent evolution: similar species that develop different characters over time



Character *a* could have evolved three times or *b* could have evolved twice



Examples of convergent evolution

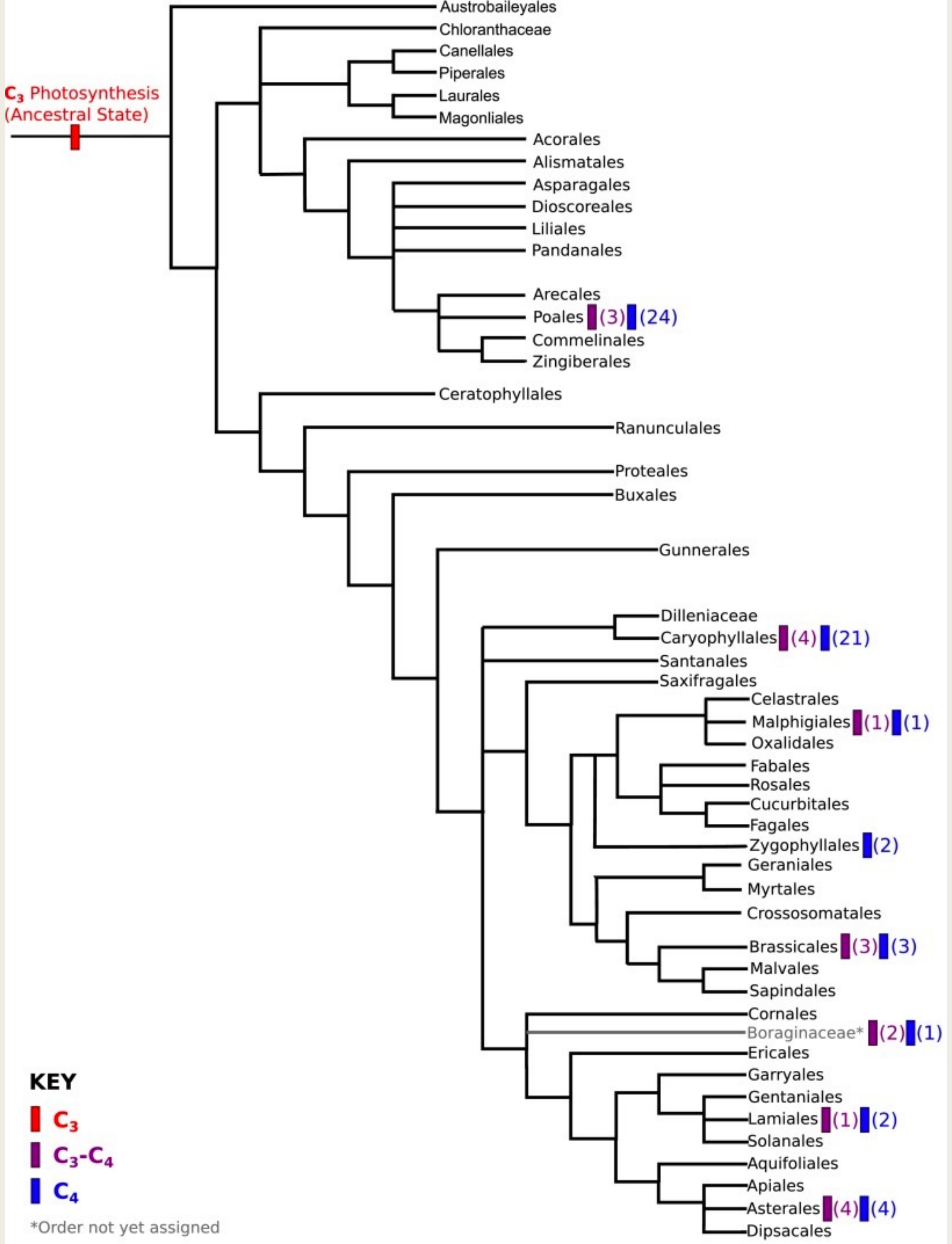
- Flight in bats and birds
- Opposable thumbs in primates and pandas
- Blue eyes in humans and lemurs $\longrightarrow$
- C_4 photosynthesis in many plants

Images: wikipedia

It happens but it is **rare**! Want a tree that *minimizes evolution*

Example of convergent evolution: C_4 photosynthesis in plants

Williams, Johnston, Covshoff, Hibberd (2013). "Phenotypic landscape inference reveals multiple evolutionary paths to C_4 photosynthesis".



Problems with parsimony

- Impractical (except for special cases – exact solution is NP-complete).

- Scales linearly with number of characters – going to be a problem for genomic data!

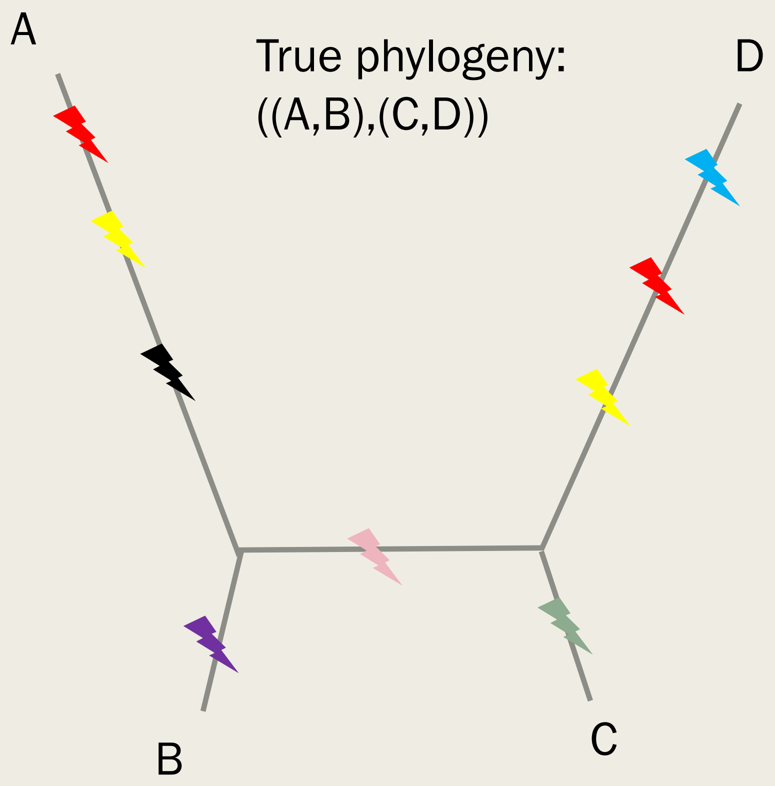
- Treats all characters the same – but some characters are more important than others

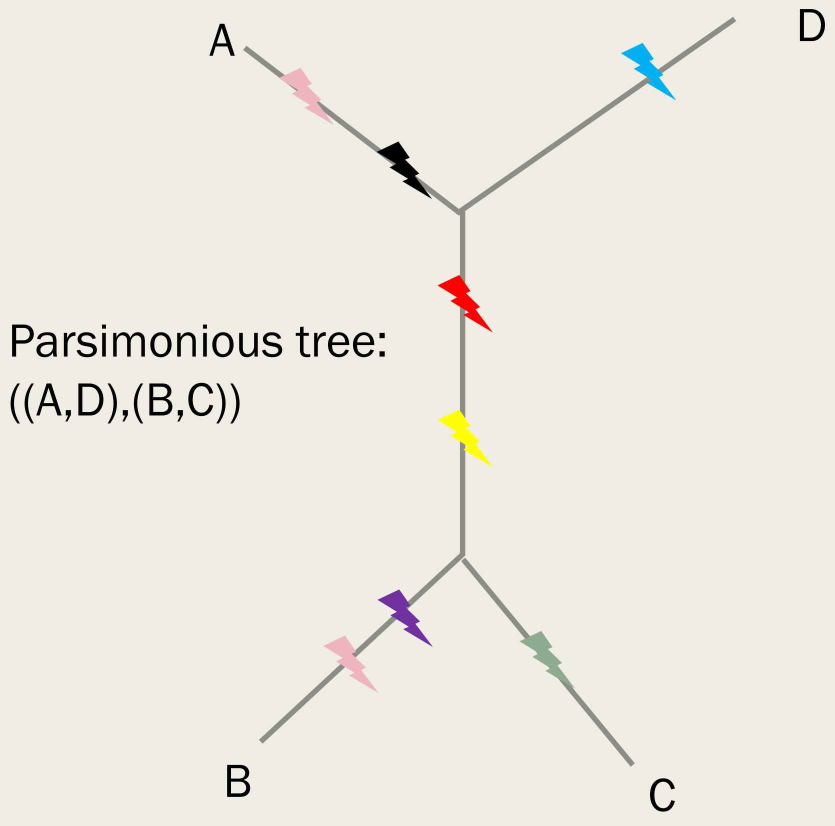
- Assumes convergent evolution is rare and that all mutations are equally likely

- Can be inconsistent – converges to the wrong answer when you have lots of data (long branch attraction)

Long branch attraction

If mutations happen at random, then long branches in the tree will tend to have more mutations -> they will look more similar -> they will be “attracted” to each other.





Gusfield's algorithm

- What is the runtime?
  - Sorting *m* characters
  - Building the tree

- Why does it work?

Gusfield’s algorithm

- Guaranteed to correctly reconstruct a perfect phylogeny if it exists
- If it does not exist, Gusfield will give you something “close” [we do not define here what “close” means]
- Can we tell if a perfect phylogeny exists just by looking at the character matrix?