CS 68: BIOINFORMATICS

Prof. Sara Mathieson
Swarthmore College
Spring 2018

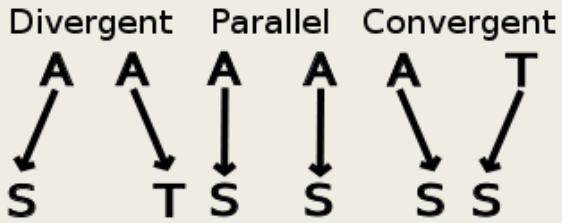
Outline: Mar 9

- Types of evolution
- Perfect phylogeny
- Gusfield’s algorithm
- Special case: root unknown

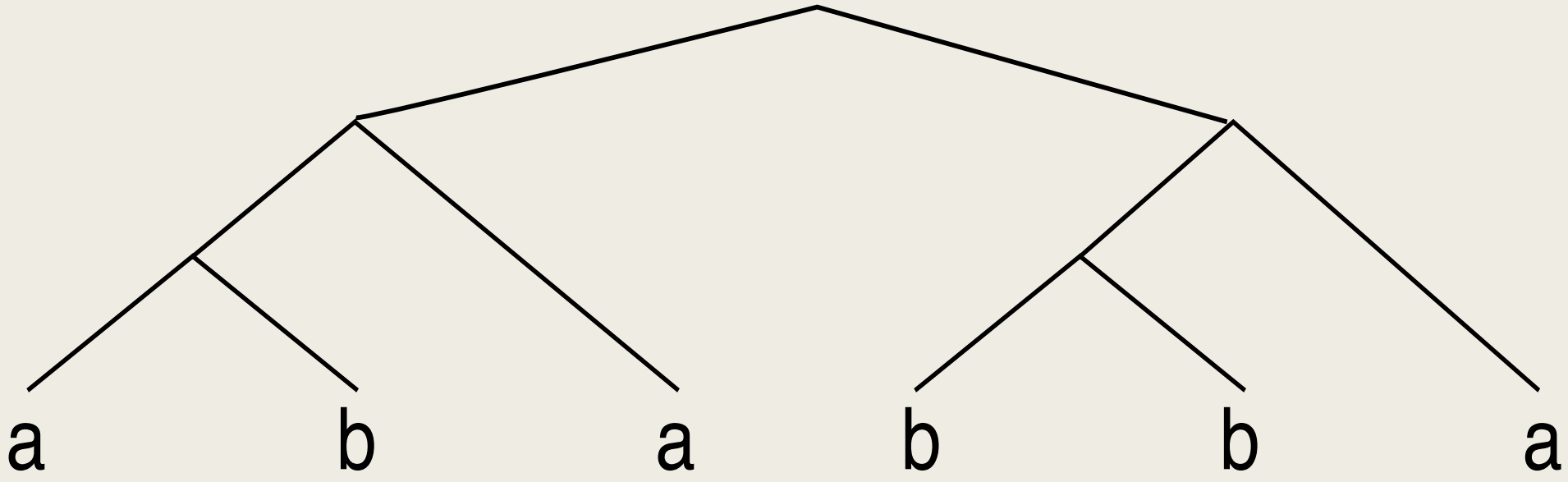
Types of evolution

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



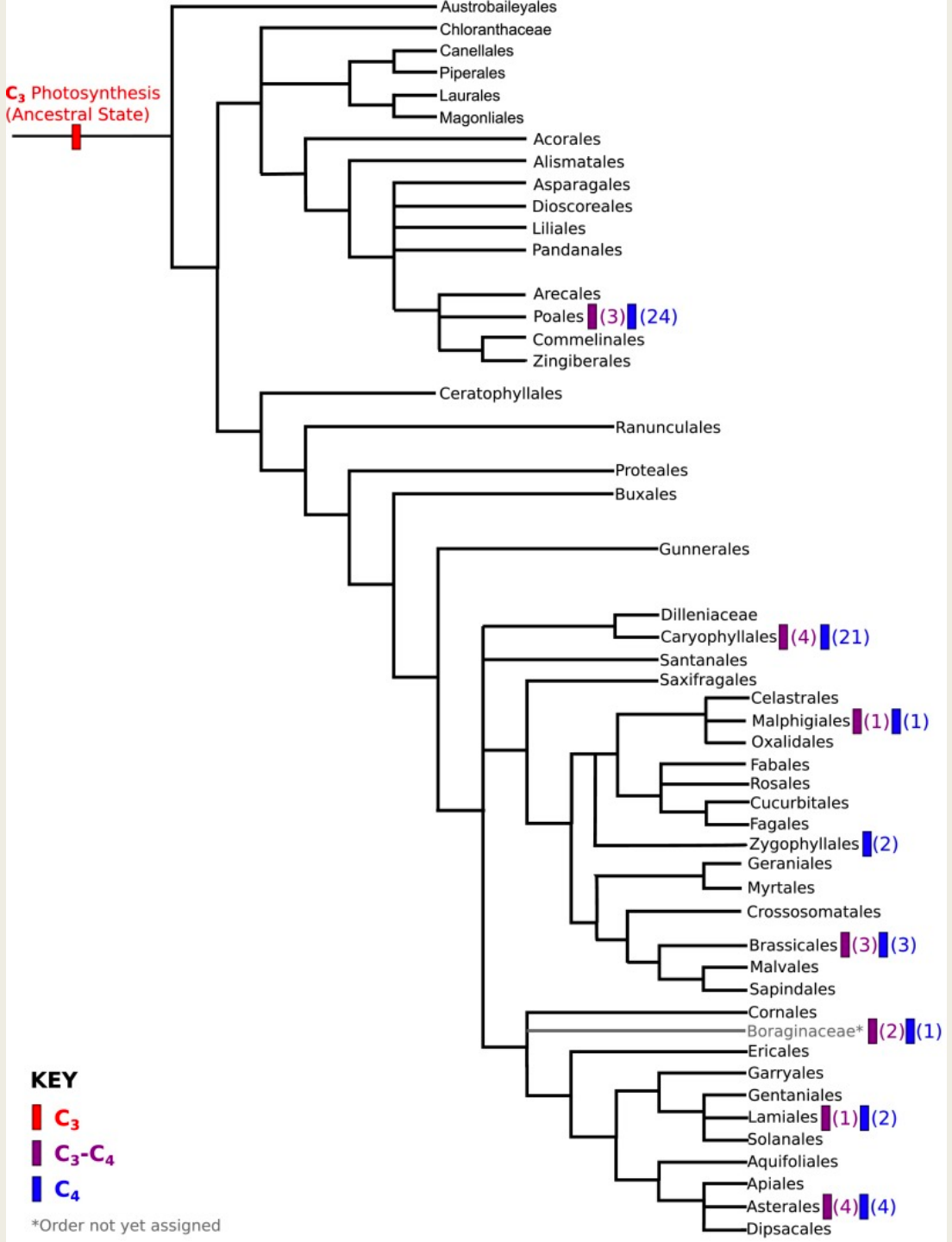
Examples of convergent evolution

- Flight evolved independently in bats and birds
- Opposable thumbs evolved independently in primates and pandas
- Blue eyes evolved independently in humans and lemurs

Images: wikipedia

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".



Perfect Phylogeny

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* ($O_i \cap O_j = \varnothing$), 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* ($O_i \cap O_j = \varnothing$), 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* ($O_i \cap O_j = \varnothing$), or
  - *One contains the other* ($O_j \subset O_i$ *or* $O_j \supset O_i$)

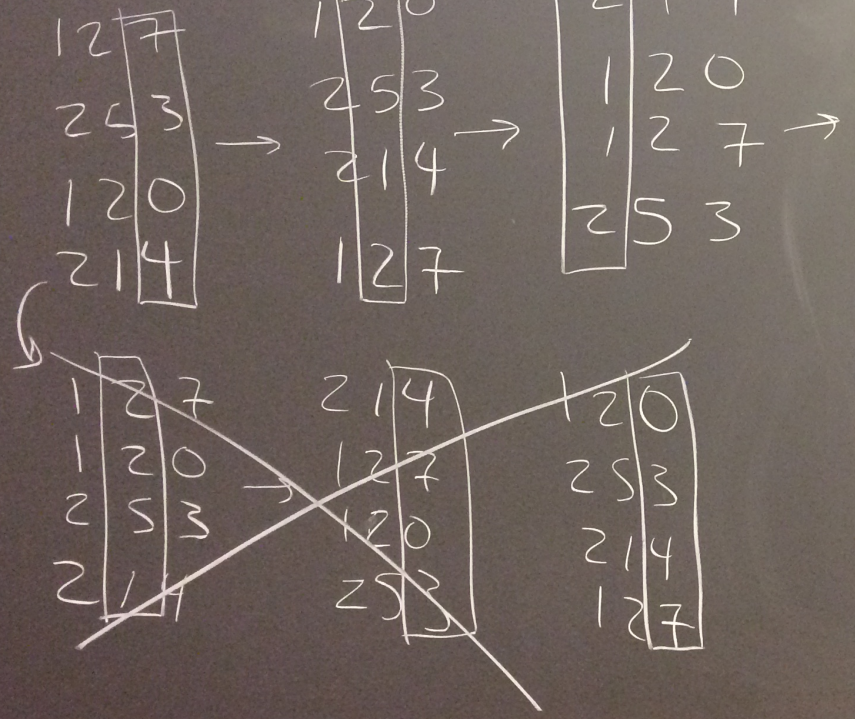
- If $O_i \supset O_j$, then character *i* occurred more anciently than character *j*

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

Radix Sort

Step 1 radix sort columns in descending order, viewing each column as a binary #.

ex



Radix Sort

- k = # possible digits.
  - $k = 10$ for decimal
  - $k = 2$ for binary
- sort into k bins, from least to most significant digit.

1	20
1	27
2	14
2	53

Sorted

Radix sort columns high to low

Handout 18:
Example 1

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

Radix sort columns high to low

Handout 18:
Example 1

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

Radix sort columns high to low

Handout 18:
Example 1

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

Radix sort columns high to low

Handout 18:
Example 1

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

Radix sort columns high to low

Handout 18:
Example 1

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

Radix sort columns high to low

Handout 18:
Example 1

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

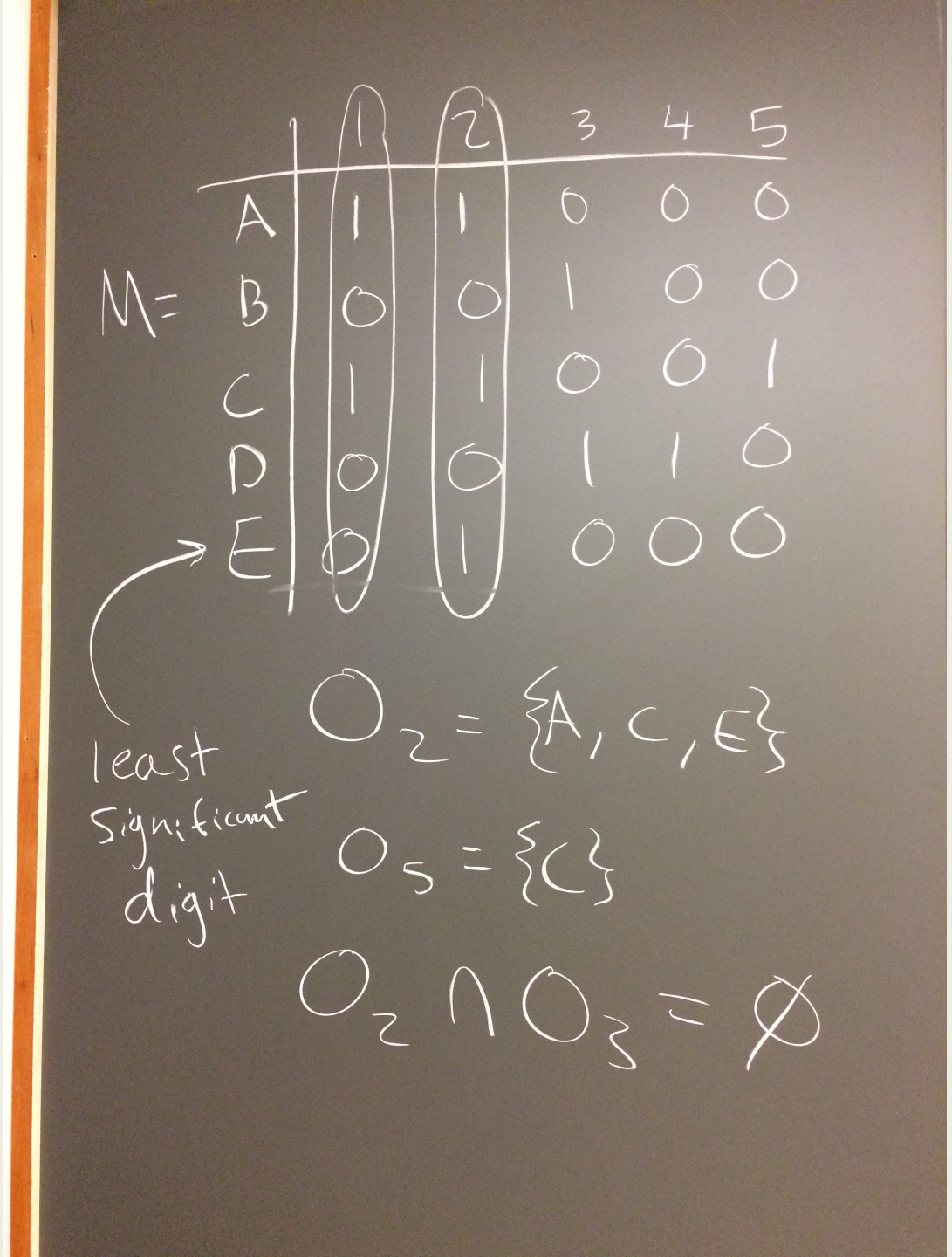
2	1	5	3	4
1	1	0	0	0
0	0	0	1	0
1	1	1	0	0
0	0	0	1	1
1	0	0	0	0

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

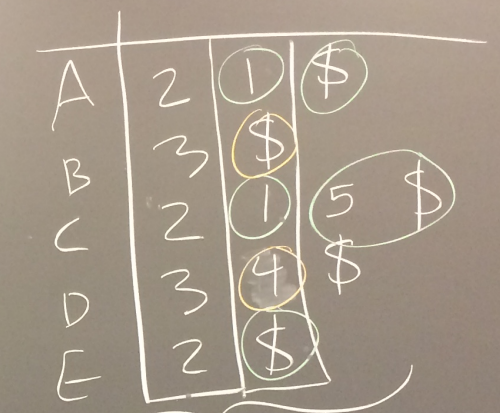
Gusfield’s Algorithm

Handout 18:
Example 1



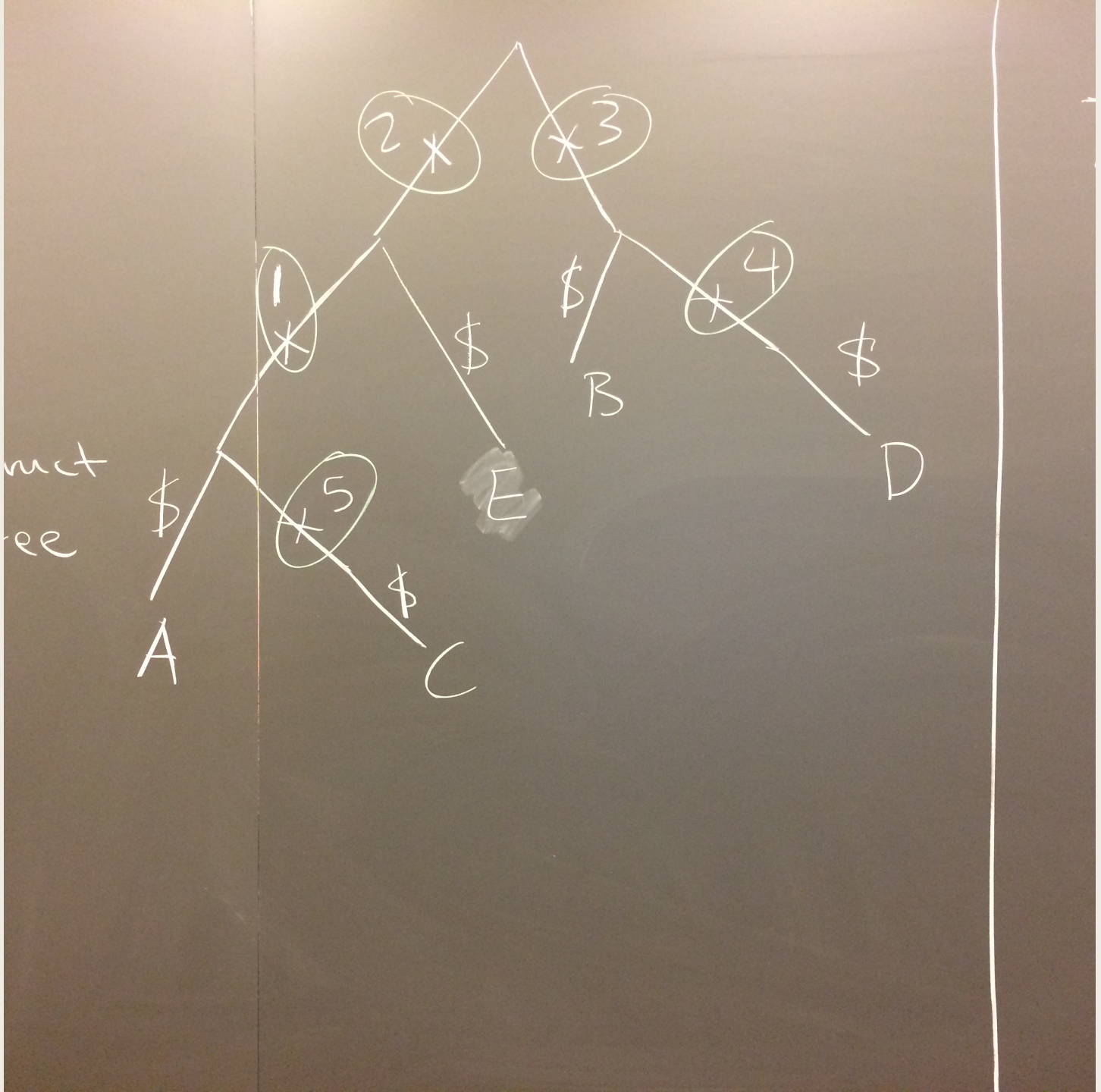
$M' =$

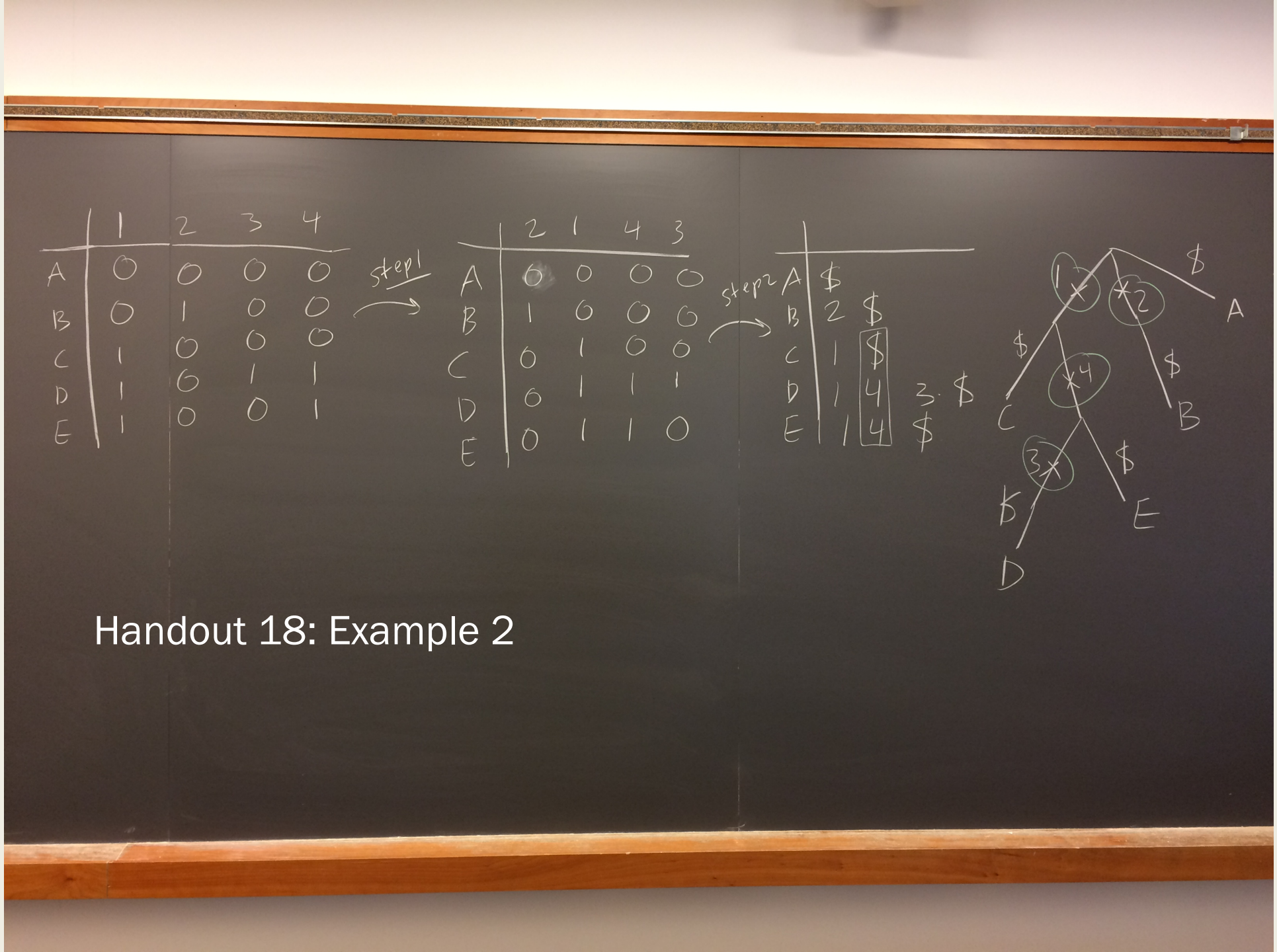
	2	1	3	5	4
A	1	1	0	0	0
B	0	0	1	0	0
C	1	1	0	1	0
D	0	0	1	0	1
E	1	0	0	0	0



Step 2 → for every row, ~~right~~ write out which mutations occur, followed by a $

Step 3 use rows to construct a keyword tree

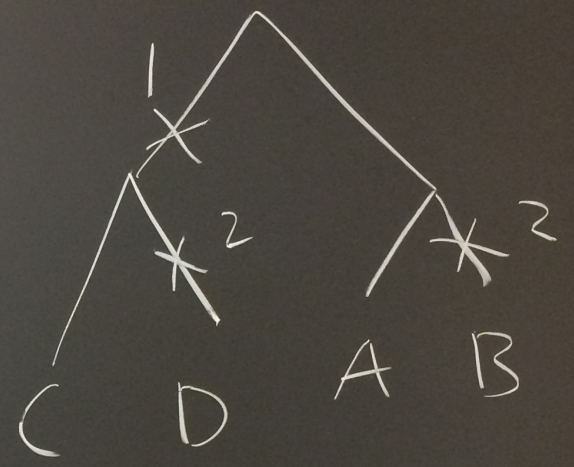

2
3
1
4
5
$
$
$
$
$
A
B
C
D
E



Handout 18: Example 2

extra

	1	2
A	0	0
B	0	1
C	1	0
D	1	1



no perfect phylogeny

$\Rightarrow$ "4 gamete test"

Don't know the root

- choose pseudoroot as the most common haplotype or sample

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

pseudoroot →

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

idea: flip all 0's & 1's if the pseudoroot was 1

→ run algorithm as normal.