CS 364 COMPUTATIONAL BIOLOGY

Sara Mathieson
Haverford College

Outline

3 handouts!

- * Handout 10
- * NJ handout
- * midterm study guide

- Recap UPGMA algorithm
- Neighbor Joining algorithm (start)
- Begin: midterm review

On notecard: write topics that need the most review

Recap UPGMA algorithm

Recap questions: discuss with a partner

1) How do we define a tree metric?

2) True or False: every dissimilarity map is a tree metric.

3) How do we define an ultrametric? (both theoretically and intuitively)

4) True or False: every tree metric is an ultrametric.

5) What biological assumption(s) are we making when creating ultrametric trees like those produced by UPGMA?

6) What two biological factors might make ultrametric trees an unrealistic assumption?

Recap questions: discuss with a partner

1) How do we define a tree metric?

   A dissimilarity map δ is a tree metric if the sum of the edge weights between A and B is equal to $\delta(A,B)$.

2) True or False: every dissimilarity map is a tree metric.

3) How do we define an ultrametric? (both theoretically and intuitively)

4) True or False: every tree metric is an ultrametric.

5) What biological assumption(s) are we making when creating ultrametric trees like those produced by UPGMA?

6) What two biological factors might make ultrametric trees an unrealistic assumption?

Recap questions: discuss with a partner

1) How do we define a tree metric?

   A dissimilarity map δ is a tree metric if the sum of the edge weights between A and B is equal to $\delta(A,B)$.

2) True or False: every dissimilarity map is a tree metric. False!

3) How do we define an ultrametric? (both theoretically and intuitively)

4) True or False: every tree metric is an ultrametric.

5) What biological assumption(s) are we making when creating ultrametric trees like those produced by UPGMA?

6) What two biological factors might make ultrametric trees an unrealistic assumption?

Recap questions: discuss with a partner

1) How do we define a tree metric?

   A dissimilarity map δ is a tree metric if the sum of the edge weights between A and B is equal to $\delta(A,B)$.

2) True or False: every dissimilarity map is a tree metric. False!

3) How do we define an ultrametric? (both theoretically and intuitively)
   - The distance from the root to each leaf is the same.
   - 3-point condition: For all distinct A,B,C, $\delta(\text{A,B}) \leq \max\{\delta(\text{A,C}), \delta(\text{B,C})\}$
   - Intuitively this means out of these three distances, two are equal and one is less.

4) True or False: every tree metric is an ultrametric.

5) What biological assumption(s) are we making when creating ultrametric trees like those produced by UPGMA?

6) What two biological factors might make ultrametric trees an unrealistic assumption?

Recap questions: discuss with a partner

1) How do we define a tree metric?

   A dissimilarity map δ is a tree metric if the sum of the edge weights between A and B is equal to $\delta(A,B)$.

2) True or False: every dissimilarity map is a tree metric. False!

3) How do we define an ultrametric? (both theoretically and intuitively)
   - The distance from the root to each leaf is the same.
   - 3-point condition: For all distinct A,B,C, $\delta(A,B) \leq \max\{\delta(A,C), \delta(B,C)\}$
   - Intuitively this means out of these three distances, two are equal and one is less.

4) True or False: every tree metric is an ultrametric.

   False!

5) What biological assumption(s) are we making when creating ultrametric trees like those produced by UPGMA?

6) What two biological factors might make ultrametric trees an unrealistic assumption?

Recap questions: discuss with a partner

1) How do we define a tree metric?

   A dissimilarity map δ is a tree metric if the sum of the edge weights between A and B is equal to $\delta(A,B)$.

2) True or False: every dissimilarity map is a tree metric. False!

3) How do we define an ultrametric? (both theoretically and intuitively)
   - The distance from the root to each leaf is the same.
   - 3-point condition: For all distinct A,B,C, $\delta(A,B) \leq \max\{\delta(A,C), \delta(B,C)\}$
   - Intuitively this means out of these three distances, two are equal and one is less.

4) True or False: every tree metric is an ultrametric. False!

5) What biological assumption(s) are we making when creating ultrametric trees like those produced by UPGMA?

   We are assuming that evolution is proportional to time (i.e. the "molecular clock" assumption).

6) What two biological factors might make ultrametric trees an unrealistic assumption?

Recap questions: discuss with a partner

1) How do we define a tree metric?

   A dissimilarity map δ is a tree metric if the sum of the edge weights between A and B is equal to $\delta(A,B)$.

2) True or False: every dissimilarity map is a tree metric. False!

3) How do we define an ultrametric? (both theoretically and intuitively)
   - The distance from the root to each leaf is the same.
   - 3-point condition: For all distinct A,B,C, $\delta(\text{A,B}) \leq \max\{\delta(\text{A,C}), \delta(\text{B,C})\}$
   - Intuitively this means out of these three distances, two are equal and one is less.

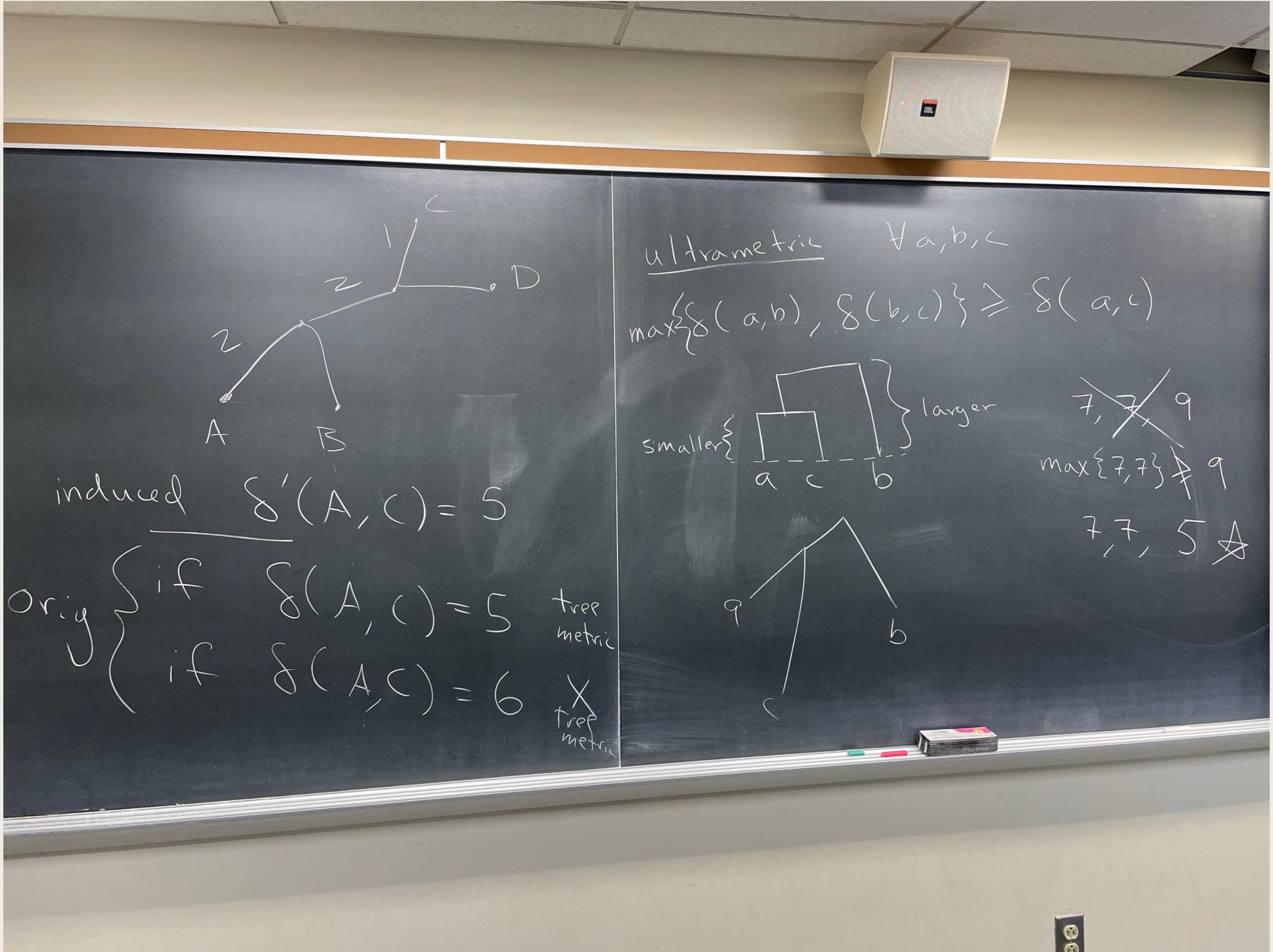
4) True or False: every tree metric is an ultrametric.

   False!

5) What biological assumption(s) are we making when creating ultrametric trees like those produced by UPGMA?

   We are assuming that evolution is proportional to time (i.e. the "molecular clock" assumption).

6) What two biological factors might make ultrametric trees an unrealistic assumption?
   - Mutation rates differ significantly across species.
   - Natural selection (both positive and negative) can change the tempo of evolution.



induced $\delta'(A,C) = 5$

Orig $\begin{cases} \text{if } \delta(A,C) = 5 & \text{tree metric} \\ \text{if } \delta(A,C) = 6 & \text{X tree metric} \end{cases}$

ultrametric $\forall a,b,c$

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

~~7, 7, 9~~

$\max\{7,7\} \ngeq 9$

7, 7, 5 ☆

Bonus questions

- In what scenarios is an ultrametric tree likely a GOOD assumption?

- What is the runtime of UPGMA in terms of the number of samples *n*?

Bonus questions

- In what scenarios is an ultrametric tree likely a GOOD assumption?

If our samples are from the same species or population, then there has likely been the same amount of evolution from the root to each leaf, so it is (usually) okay to assume time and evolution are proportional.

- What is the runtime of UPGMA in terms of the number of samples *n*?

Bonus questions

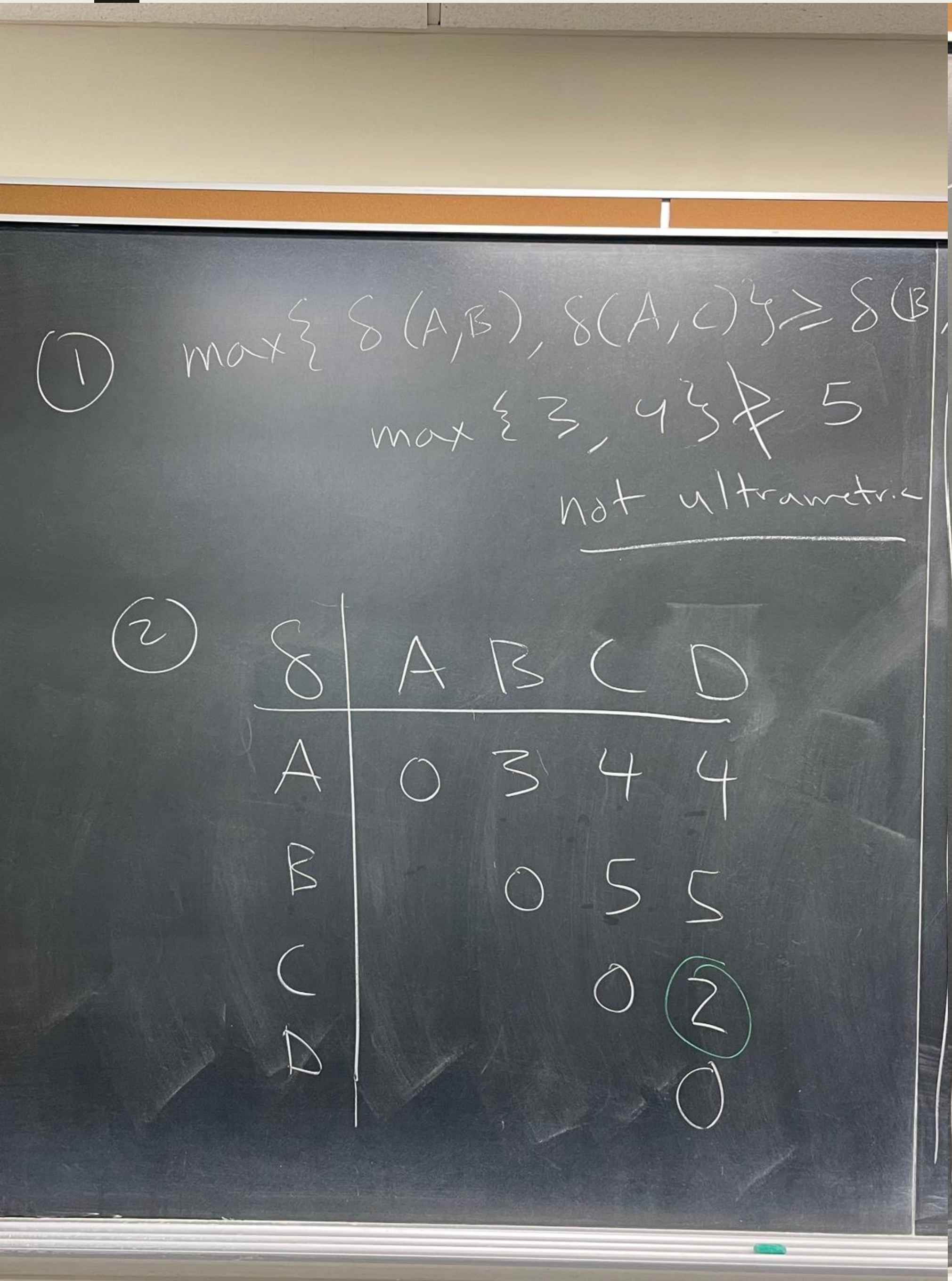
- In what scenarios is an ultrametric tree likely a GOOD assumption?

If our samples are from the same species or population, then there has likely been the same amount of evolution from the root to each leaf, so it is (usually) okay to assume time and evolution are proportional.

- What is the runtime of UPGMA in terms of the number of samples *n*?

During each iteration we must do O(n^2) work to compute the new matrix of distances. We merge two nodes each iteration, so we have O(n) iterations total. This gives us a runtime of O(n^3), which can be improved by reusing some distances from the previous iteration.

Handout 9, page 2



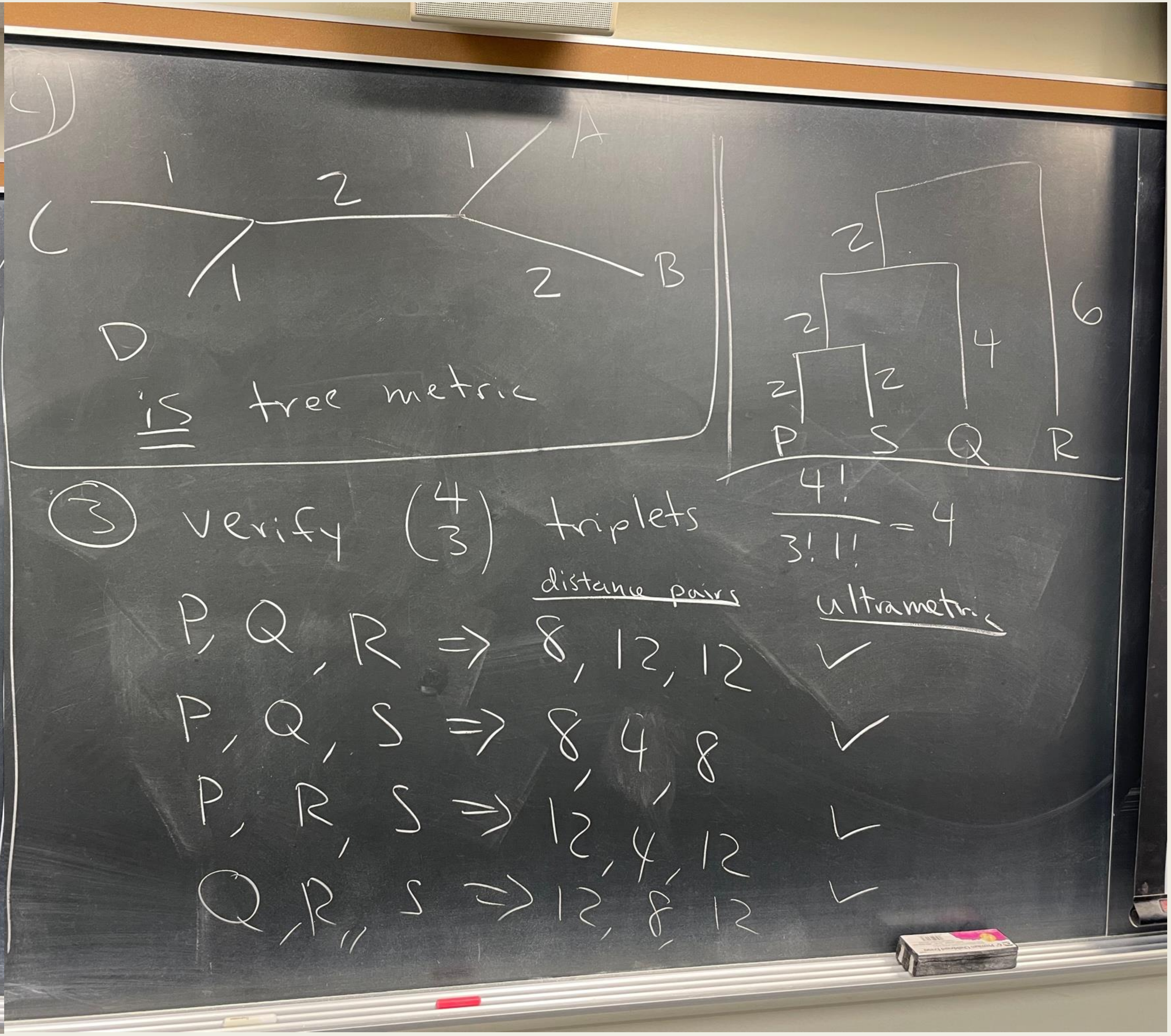
① $\max\{\delta(A,B), \delta(A,C)\} \geq \delta(B,$

$\max\{3, 4\} \ngeq 5$

not ultrametric

②

δ	A	B	C	D
A	0	3	4	4
B		0	5	5
C			0	2
D				0



is tree metric

③ verify $\binom{4}{3}$ triplets $\frac{4!}{3!\,1!} = 4$

	distance pairs	ultrametric
P, Q, R ⇒	8, 12, 12	✓
P, Q, S ⇒	8, 4, 8	✓
P, R, S ⇒	12, 4, 12	✓
Q, R, S ⇒	12, 8, 12	✓

Next phylogenetic tree algorithm: Neighbor-Joining (NJ)

Notes about UPGMA vs NJ

- NJ was first described in 1987 by Saitou and Nei. Their paper currently has 73,000 citations (an average of over 5 citations a day for the last 37 years!)

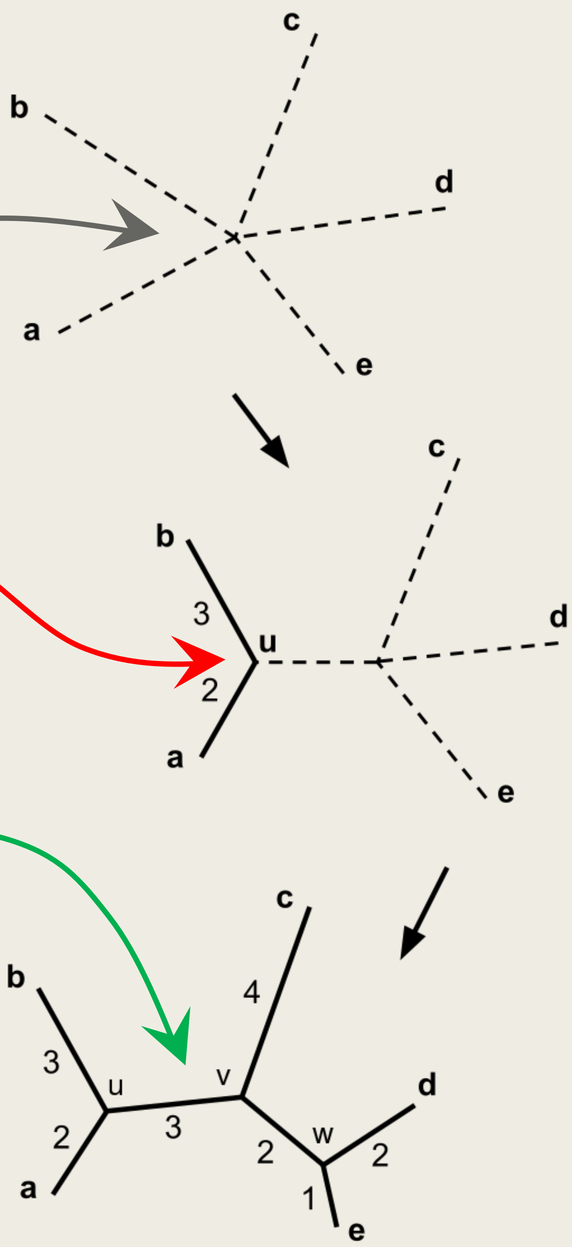
- Both UPGMA and NJ are greedy, polynomial-time clustering algorithms that produce edge weights as well as binary tree topologies.

- NJ creates unrooted trees (direction of evolution is not apparent on all branches), while UPGMA creates rooted trees.

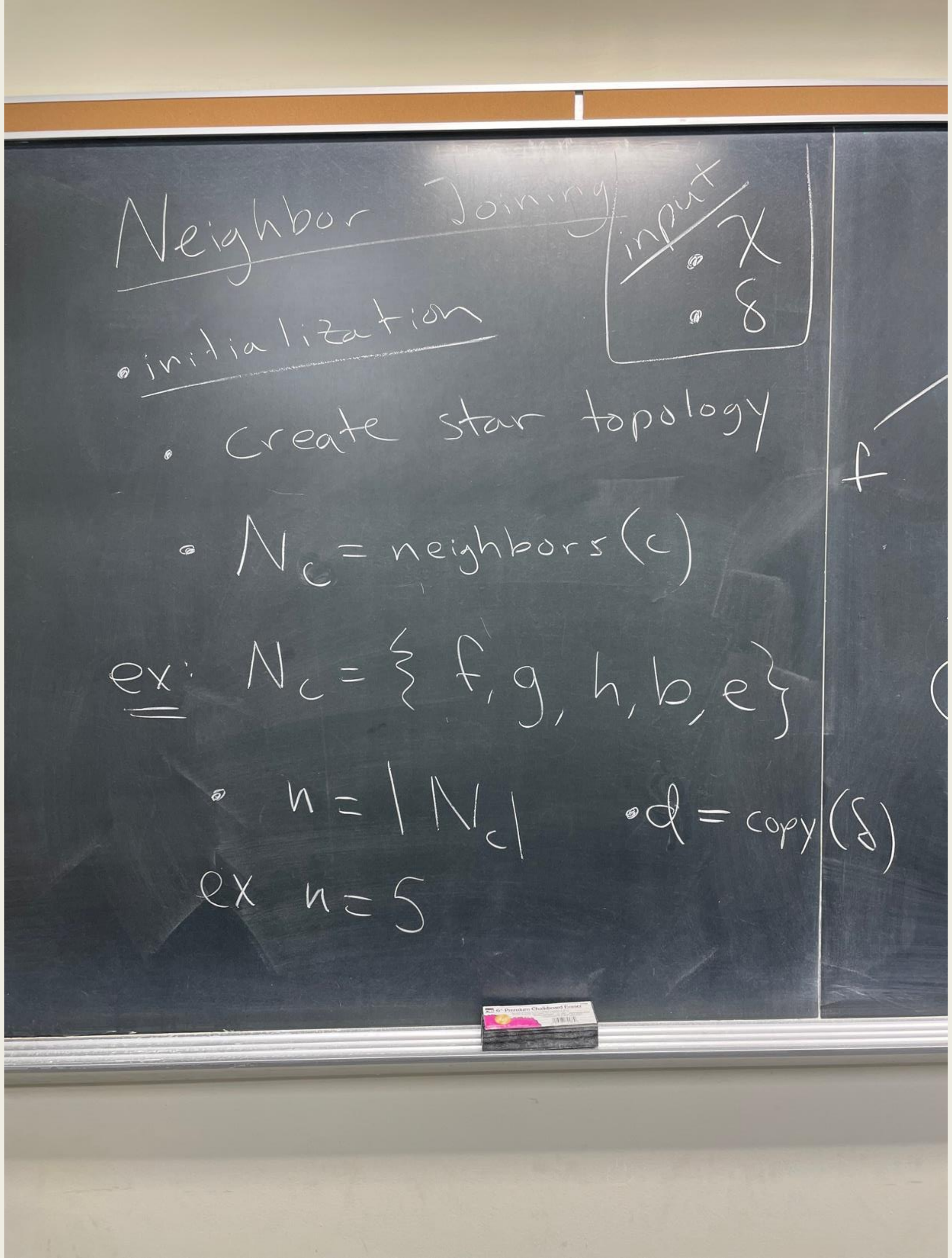
- NJ is much better for representing multi-species evolution and in general creates more realistic trees that better approximate the original dissimilarity map.

NJ at high level

- Start with a star tree
- At each stage, add another node that connects two other nodes. Chose this node to minimize some function of the distances.
- Repeat until we have a binary tree (i.e. every node has three edges)



So just like UPGMA, we join two taxa at each iteration, but instead of choosing the minimum entries in Δ, we choose the minimum entries in Q, a matrix which is a function of distances.



Neighbor Joining

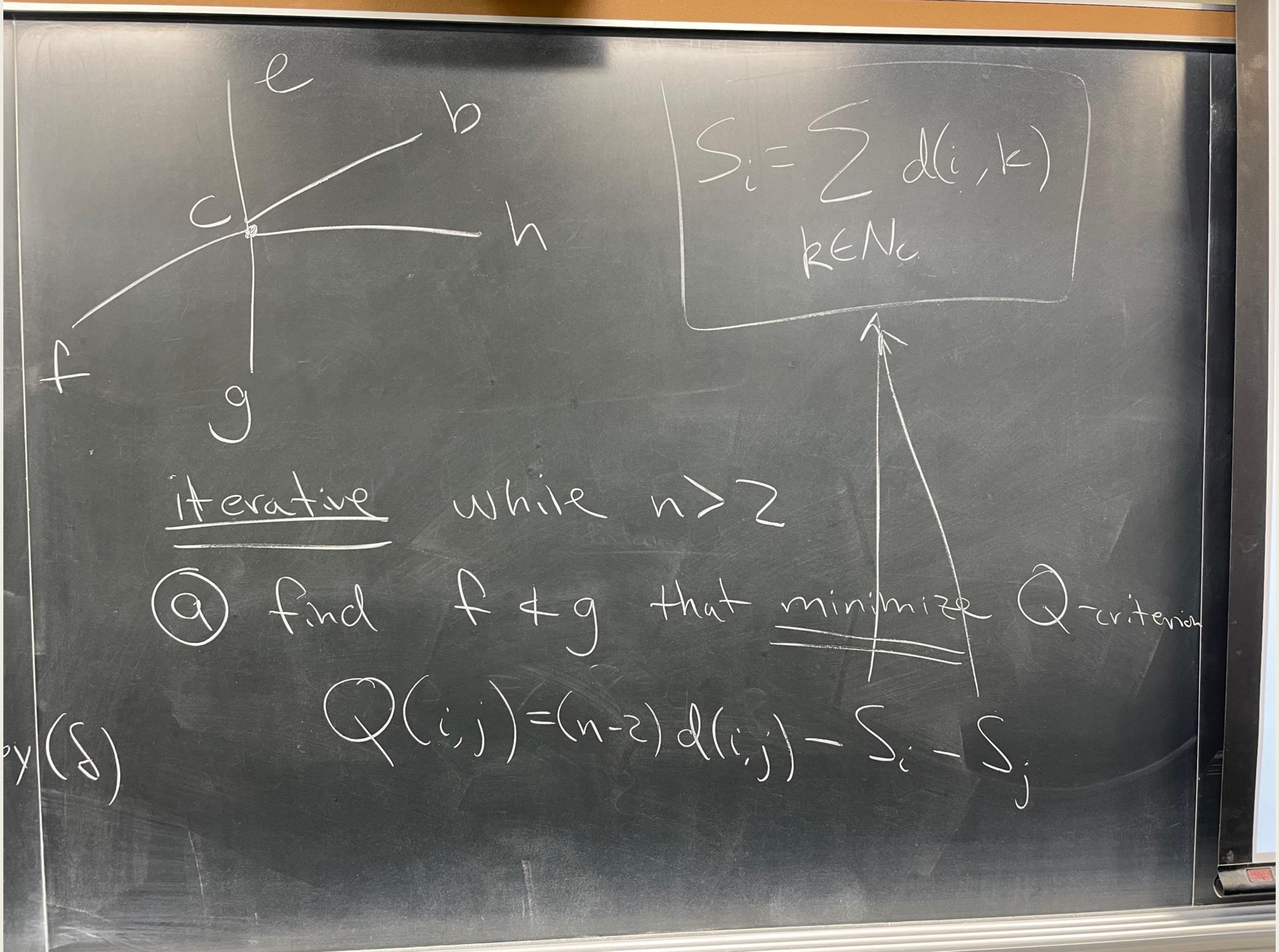
input
- χ
- δ

- initialization
  - create star topology
  - $N_c = \text{neighbors}(c)$

ex: $N_c = \{f, g, h, b, e\}$

- $n = |N_c|$
- $d = \text{copy}(\delta)$

ex $n = 5$



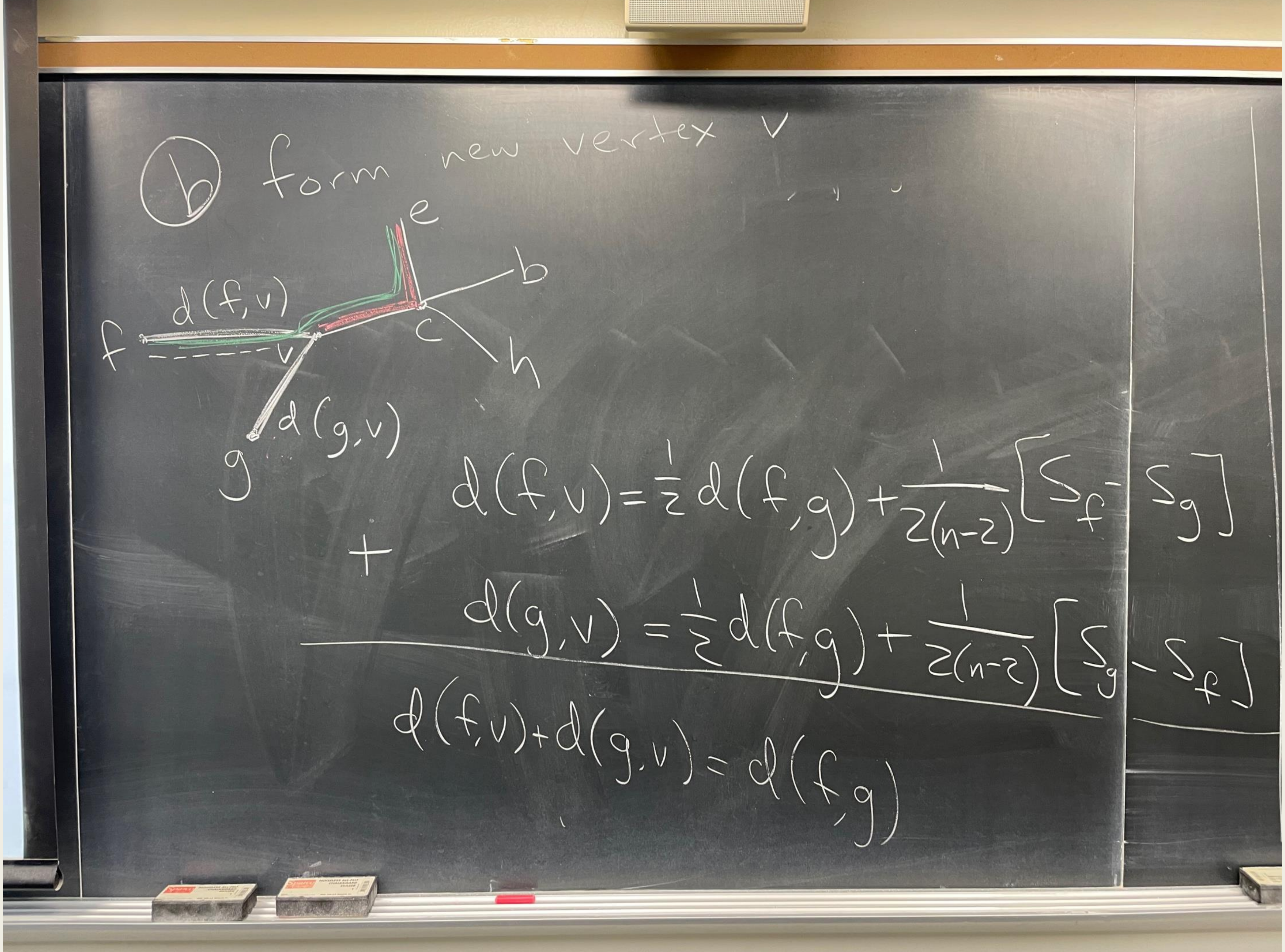
$$\boxed{S_i = \sum_{k \in N_c} d(i,k)}$$

<u>iterative</u> while $n > 2$

ⓐ find f & g that minimize Q-criterion

$$Q(i,j) = (n-2)\, d(i,j) - S_i - S_j$$

py(δ)

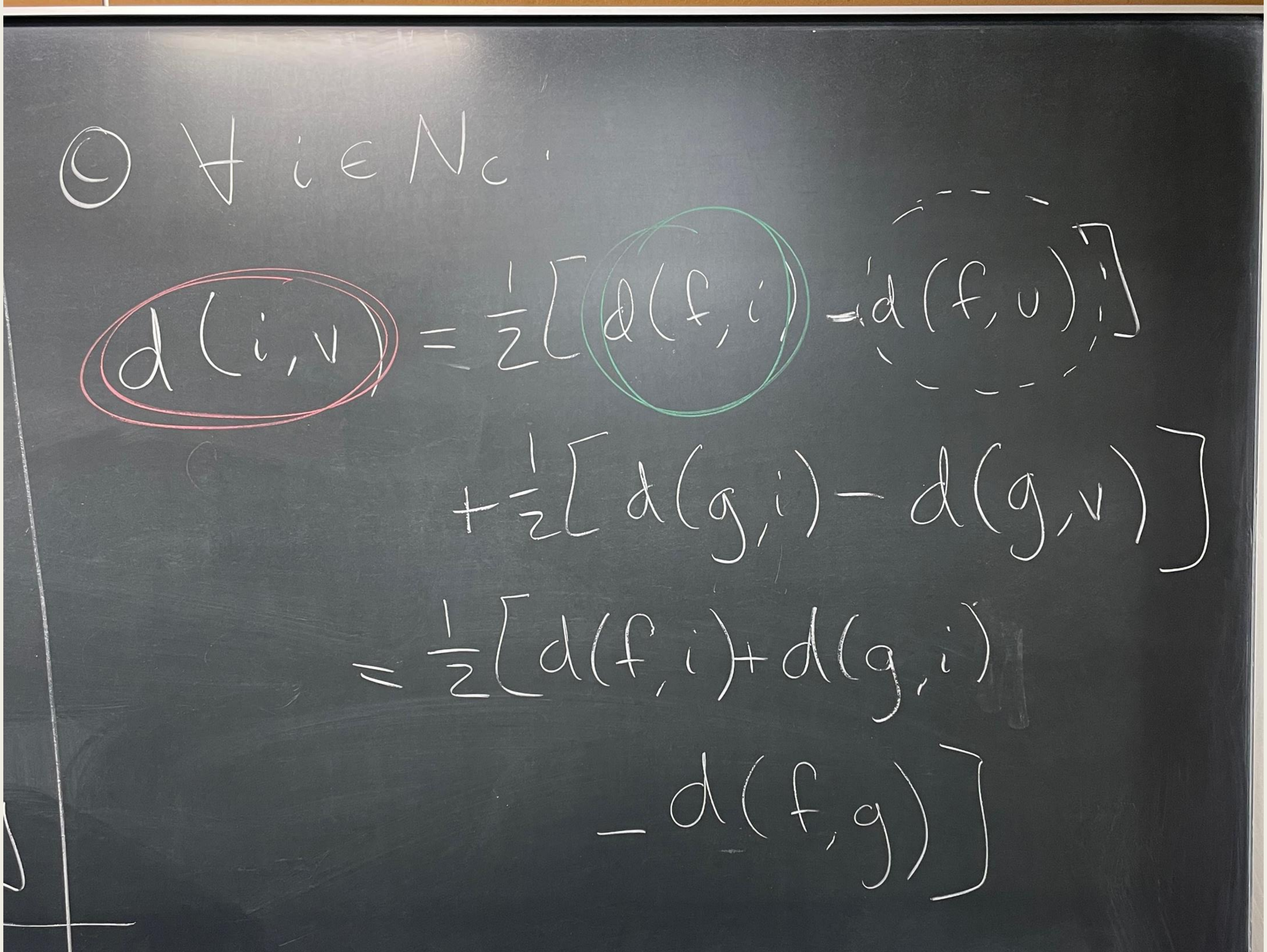


ⓑ form new vertex v

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

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

$$d(f,v) + d(g,v) = d(f,g)$$

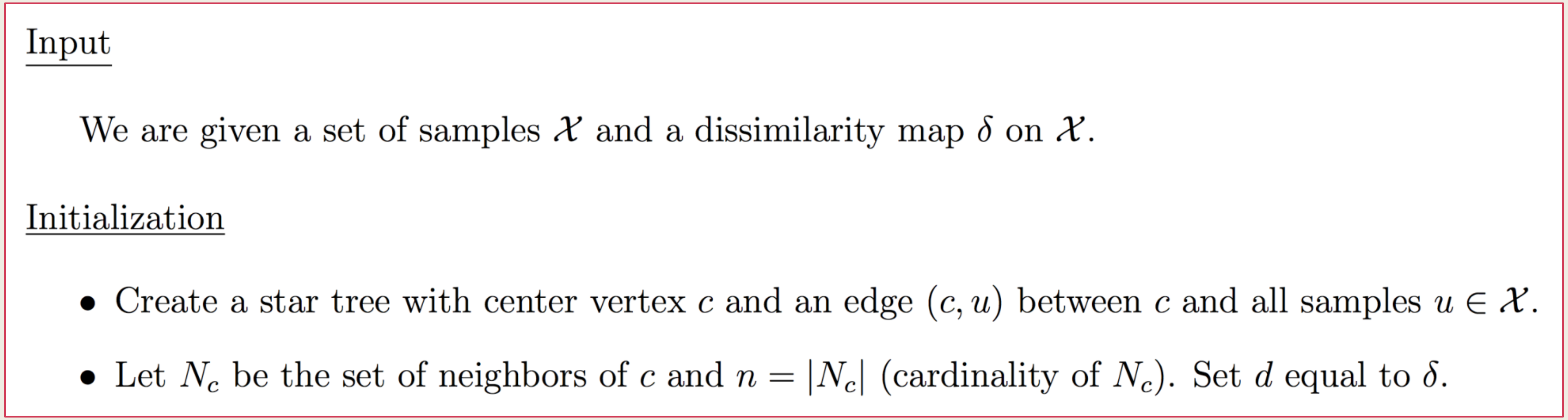


ⓒ $\forall\, i \in N_c$:

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

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

NJ initialization

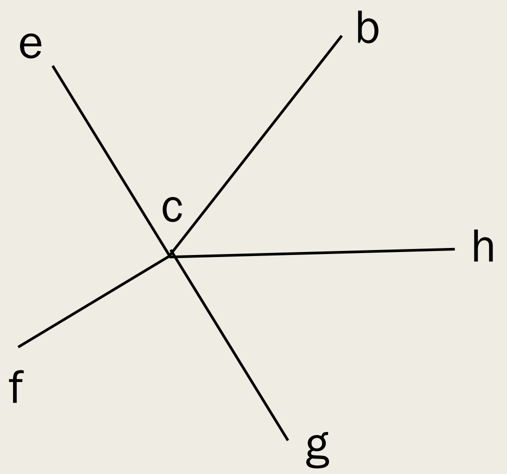


Input

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

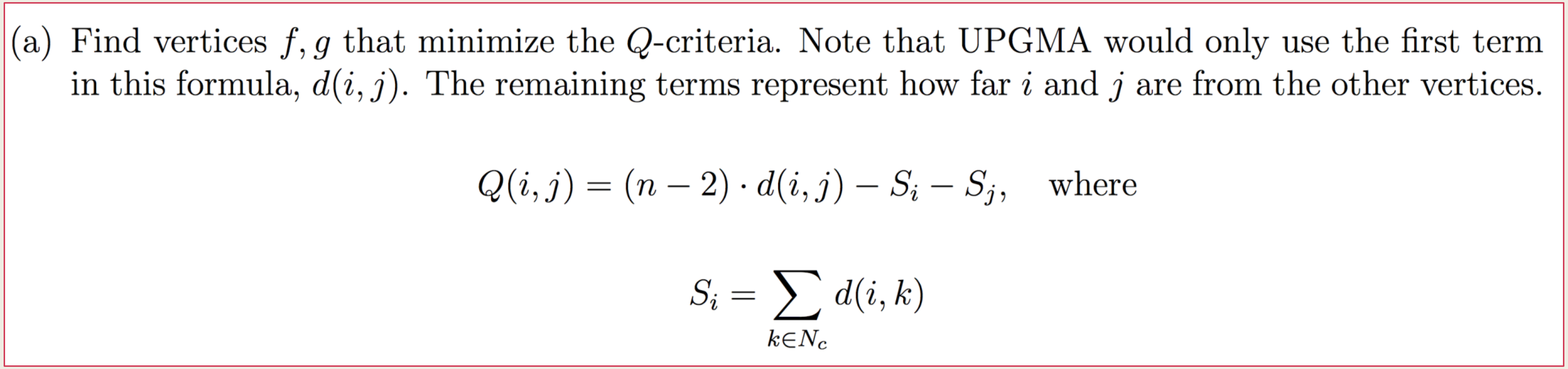
Initialization

- 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)

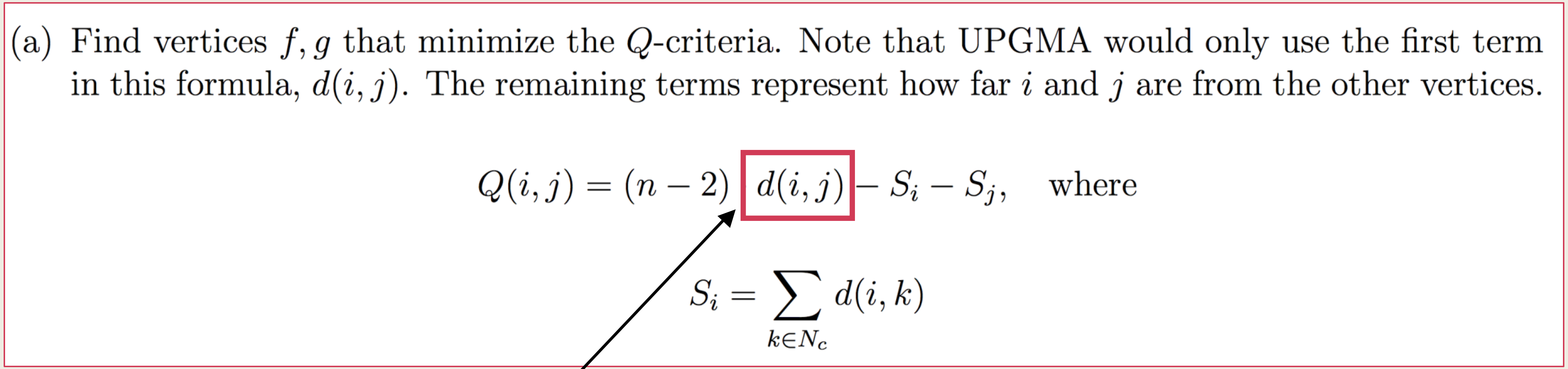


(a) Find vertices f, g that minimize the Q-criteria. Note that UPGMA would only use the first term in this formula, $d(i, j)$. The remaining terms represent how far i and j are from the other vertices.

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

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

NJ Iterative step (part a)



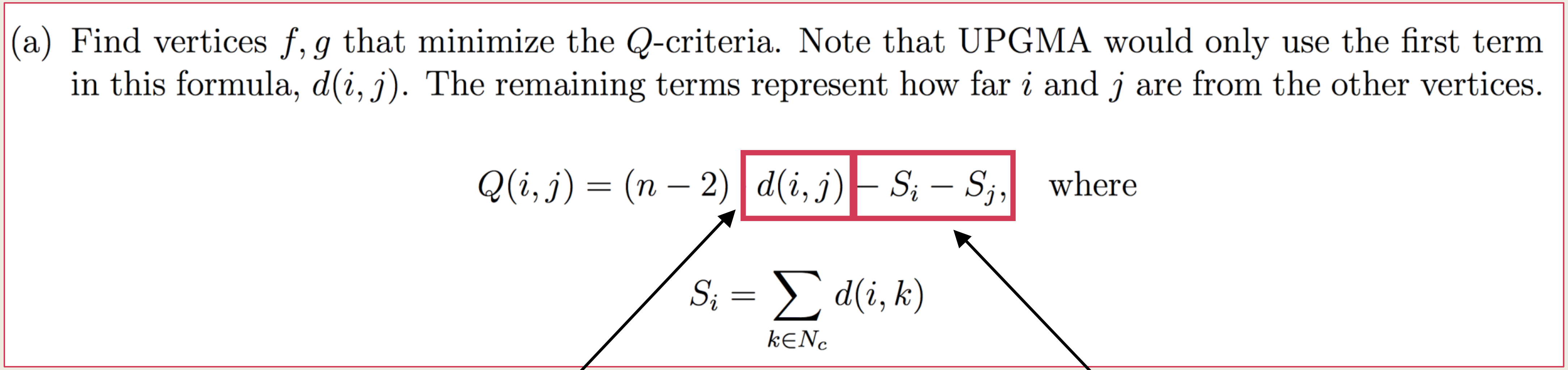
(a) Find vertices f, g that minimize the Q-criteria. Note that UPGMA would only use the first term in this formula, $d(i, j)$. The remaining terms represent how far i and j are from the other vertices.

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

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

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.

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

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

UPGMA

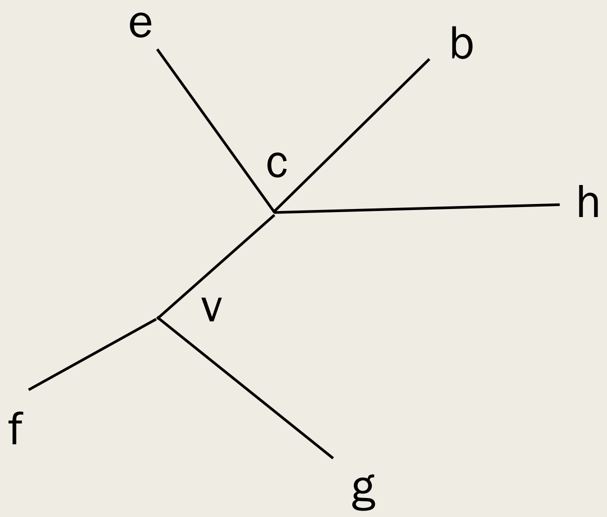
How far away *i* and *j* are from all the other vertices (further away means we'll join them earlier)

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:

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

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

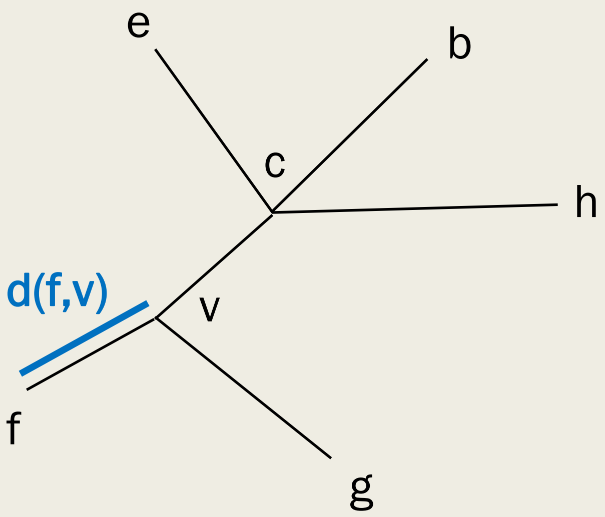


NJ Iterative step (part b)

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

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

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

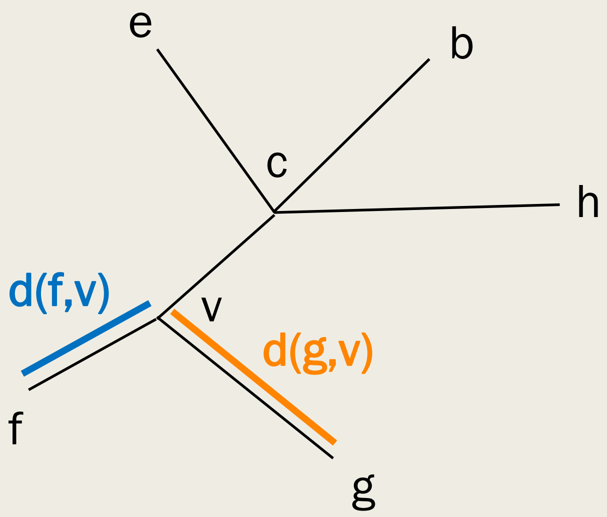


NJ Iterative step (part b)

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

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

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



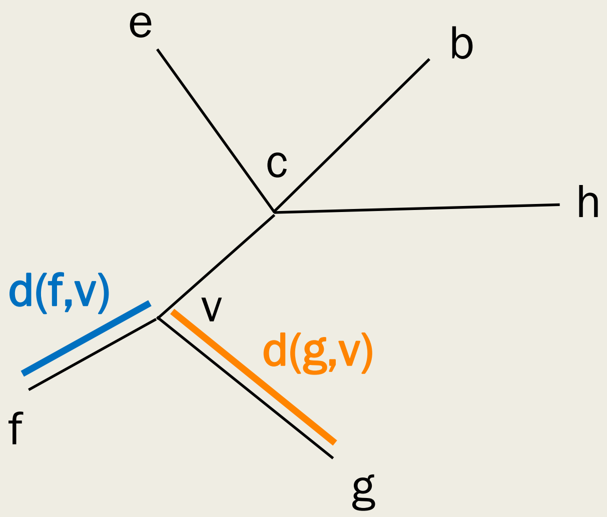
NJ Iterative step (part b)

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:

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

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



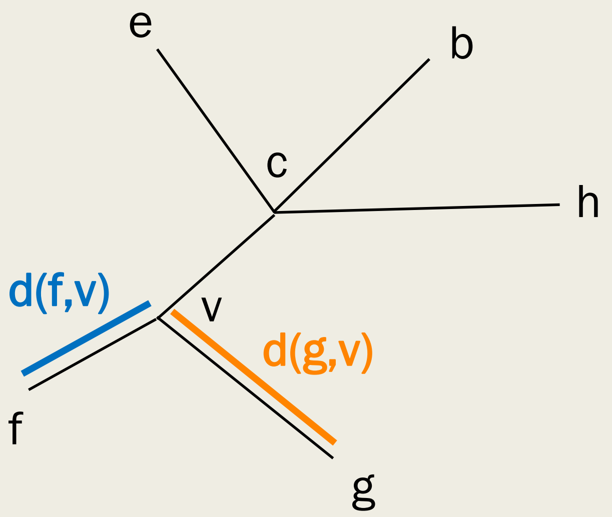
NJ Iterative step (part b)

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)

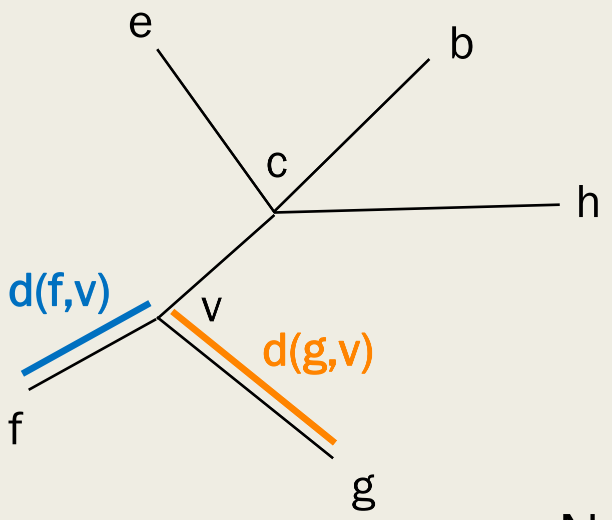
NJ Iterative step (part b)

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:

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

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



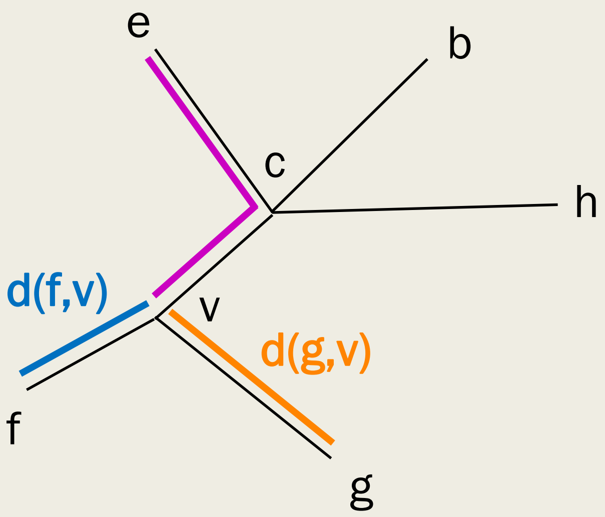
The *difference* between how far *f* and *g* are from other vertices. In this example *g* is on average further from other vertices, so d(g,v) > d(f,v)

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

NJ Iterative step (part c)

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

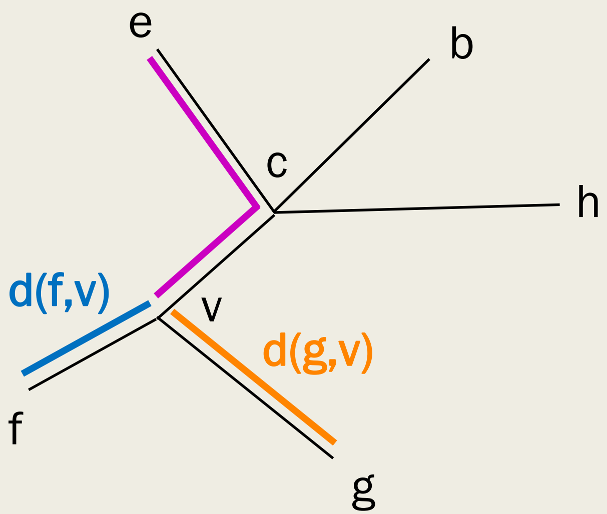
$$\boxed{d(i,v)} = \frac{1}{2}[d(f,i) - d(f,v)] + \frac{1}{2}[d(g,i) - d(g,v)]$$



NJ Iterative step (part c)

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

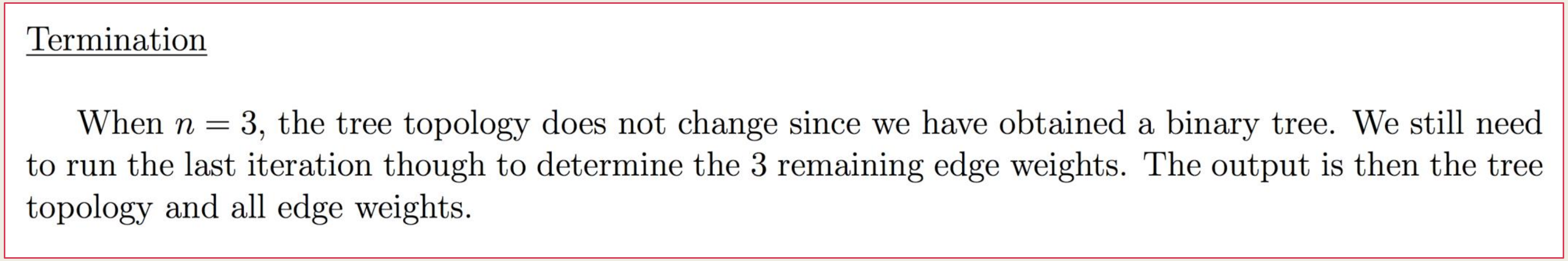
$$\boxed{d(i,v)} = \frac{1}{2}[d(f,i) - d(f,v)] + \frac{1}{2}[d(g,i) - d(g,v)]$$



Another way to write this:

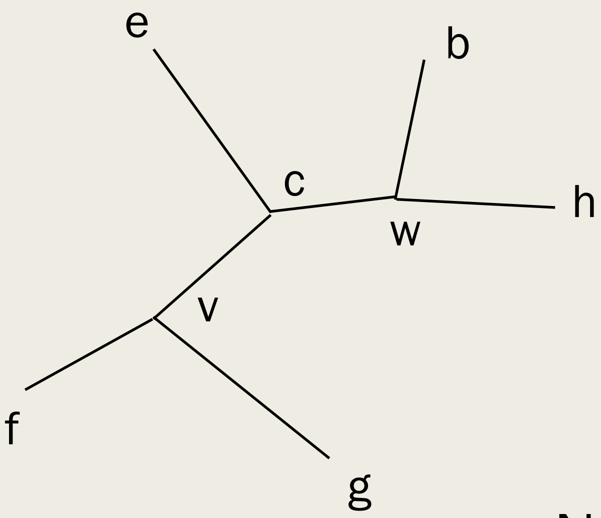
d(i,v) = ½[d(f,i) + d(g,i) – d(f,g)]

NJ Termination



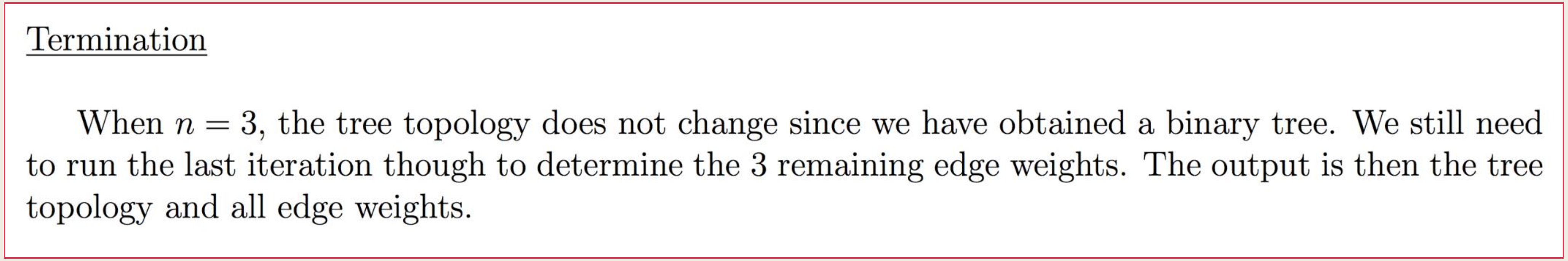
Termination

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.



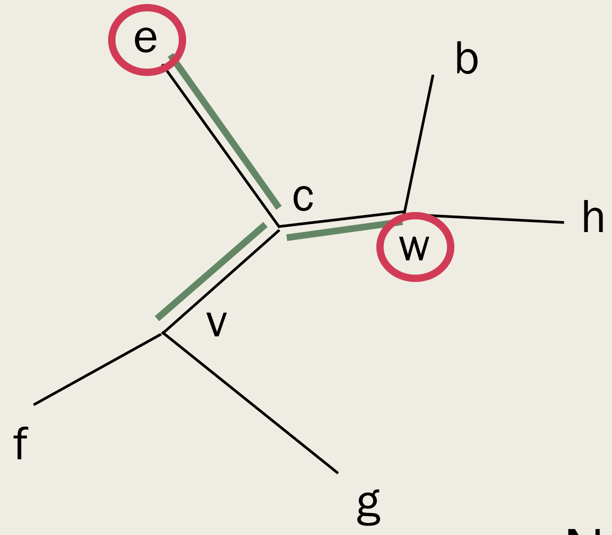
$N_c = \{e, v, w\}, \quad |N_c| = 3$

NJ Termination



Termination

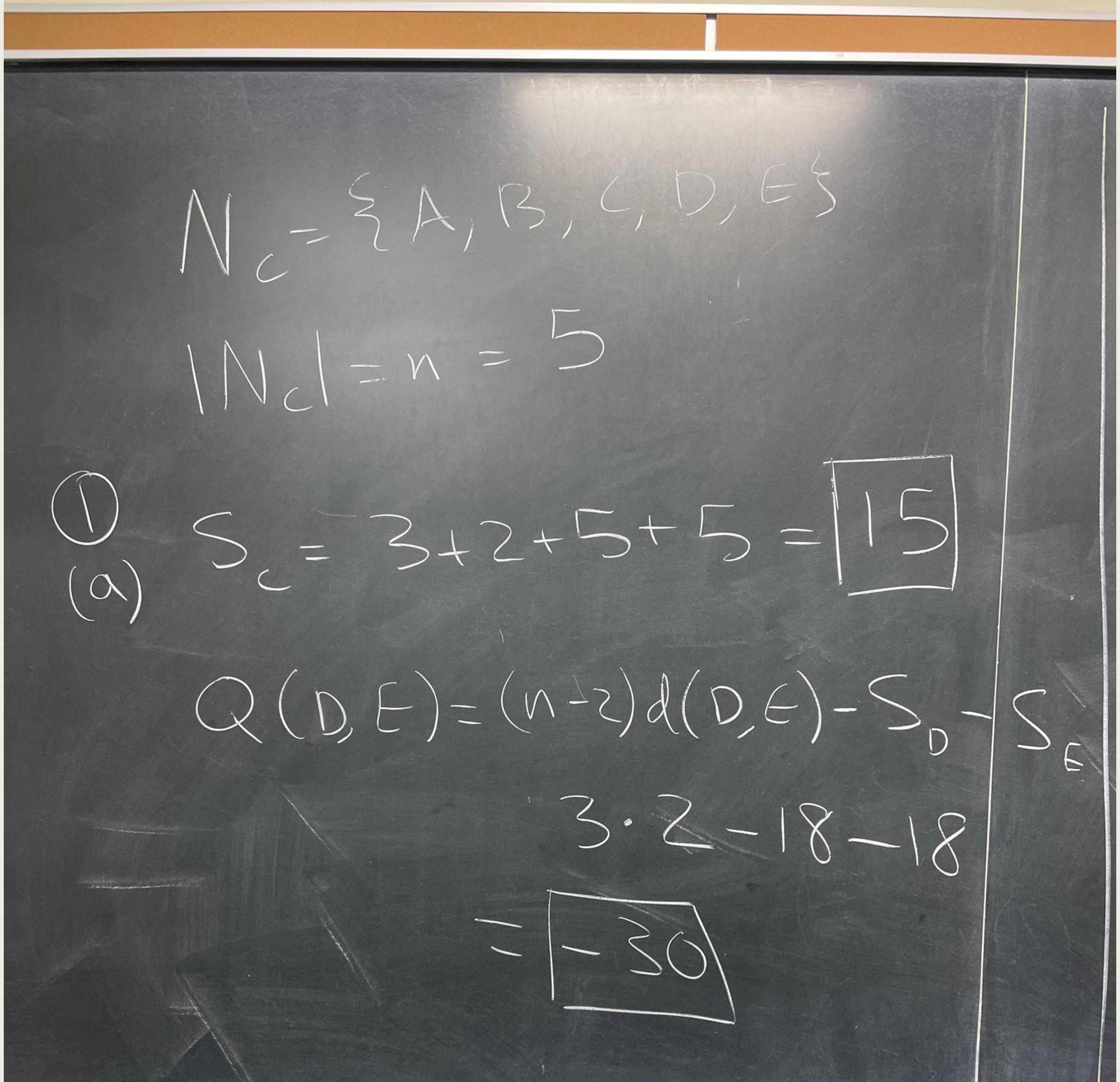
When $n = 3$, the tree topology does not change since we have obtained a binary tree. We still need to run the last iteration though to determine the 3 remaining edge weights. The output is then the tree topology and all edge weights.



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)

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

Handout 10



$$N_c = \{A, B, C, D, E\}$$

$$|N_c| = n = 5$$

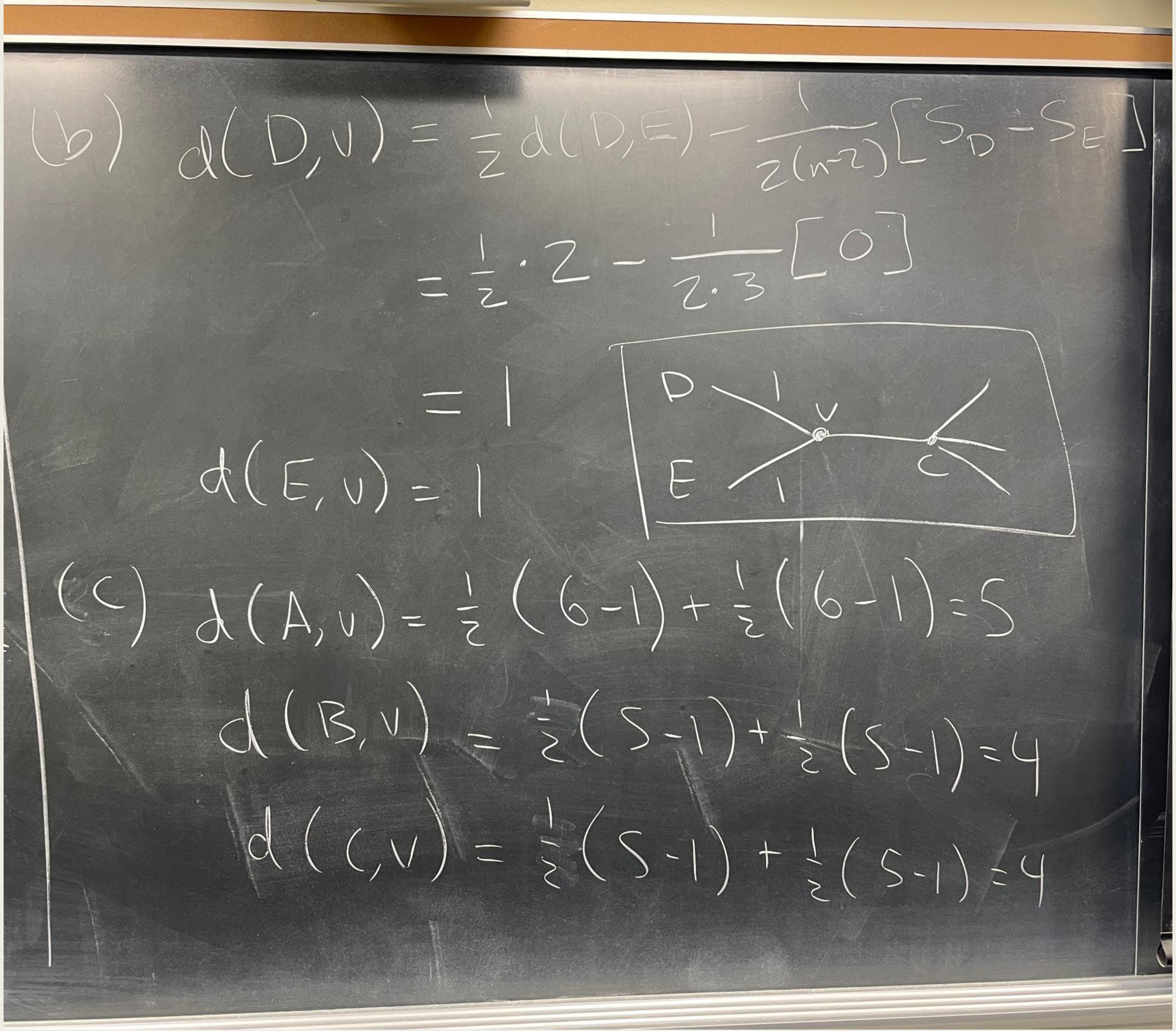
① (a)

$$S_c = 3+2+5+5 = \boxed{15}$$

$$Q(D,E) = (n-2)d(D,E) - S_D - S_E$$

$$3\cdot 2 - 18 - 18$$

$$= \boxed{-30}$$



(b) $d(D,v) = \frac{1}{2} d(D,E) - \frac{1}{2(n-2)}[S_D - S_E]$

$= \frac{1}{2} \cdot 2 - \frac{1}{2 \cdot 3}[0]$

$= 1$

$d(E,v) = 1$

(c) $d(A,v) = \frac{1}{2}(6-1) + \frac{1}{2}(6-1) = 5$

$d(B,v) = \frac{1}{2}(5-1) + \frac{1}{2}(5-1) = 4$

$d(C,v) = \frac{1}{2}(5-1) + \frac{1}{2}(5-1) = 4$

Begin: midterm 1 review

Topics for Midterm 1

1) String search
2) BWT and Read Mapping
3) Genome Assembly
4) Pairwise Sequence Alignment
5) Multiple Sequence Alignment and Phylogenetics

(2) BWT and Read Mapping

- Input: previously assembled reference sequence and millions-billions of reads from a new individual of the same species

- Output: the location(s) where each read maps (+ where the mismatches are)

- Pairwise sequence alignment is too slow

- What is the runtime of constructing the BWT and FM-Index? After that, what is the runtime of pattern matching? (see Lab 2)

(3) Genome Assembly

- Often the first step in studying the genetics of a new species
- Input: millions-billions of reads (used to be “long” reads, now are “short”)
- Output: contigs (ideally long and accurate, making up as much of the original genome as possible)
- Overlap graph assembly (Overlap Layout Consensus: OLC). Accurate but very slow
- De Bruijn graph (DBG) assembly. Fast but sometimes not as accurate
- What are the runtimes of these assembly algorithms in terms of *n, m, R*?

(4) Pairwise Sequence Alignment

- Used for studying the relationship between homologous sequences (often genes or regions from different species)
- Could be run after assembling two very different species
- Could be run on repetitive but diverged regions from the same individual
- We are giving up runtime by allowing gaps and mismatches
- Input: two sequences *x* and *y*, typically of similar length but not always. We also need a substitution matrix and gap penalty
- Output: optimal alignment(s) between *x* and *y*, AND an alignment score (higher is more similar, negative is usually not biologically meaningful)
- Two dynamic programming variations: global sequence alignment (align entire *x* with entire *y*) and local alignment (align highly similar regions in *x* and *y*)

How to study

- Go over all slides and readings => create study sheet (handwritten)

- Redo all handouts and questions/problems during class
  - *Including runtime*

- Come to office hours and lab next week to ask questions! (and/or Piazza)

On notecard: write topics that need the most review