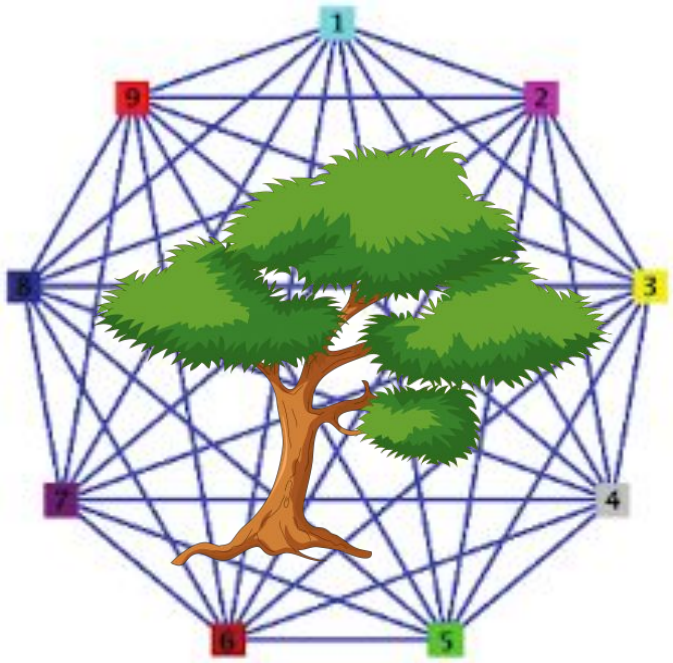
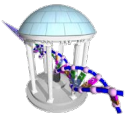
Comp 555 - BioAlgorithms - Spring 2022



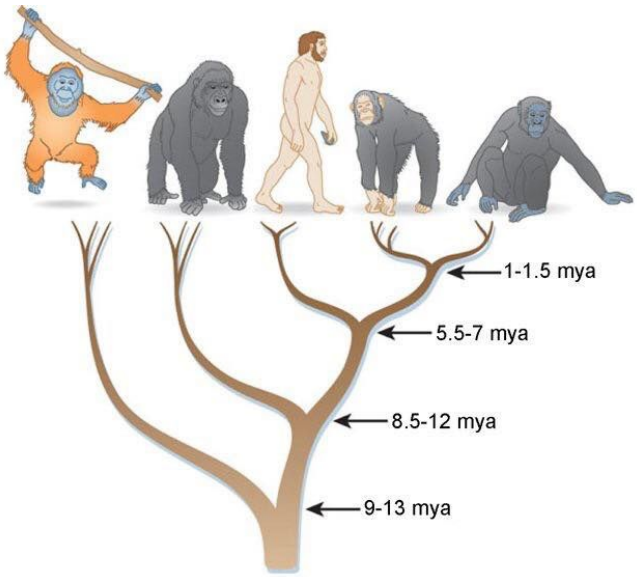
- PS #5 is graded
  PS #6 by Thursday
- Final Study session
  Thursday April 28
  (2:00-3:15)
- No office hrs on Thursday

Evolutionary Trees

Tools for Measuring Dissimilarity

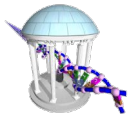


- We've developed many tools for comparing pairs of sequences this semester
  - Hamming Distance
  - Alignment scores
  - k-mer profiles
- Given a population of N sequences it is a simple matter to compare them pairwise



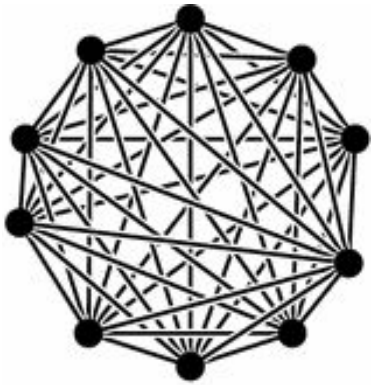
- Is it possible to ascertain their evolutionary relationships?

Distance Matrices

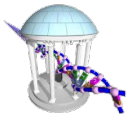


- One can easily construct a matrix filled with scores comparing the "Distances" between each pair of sequences
- Properties
  - Identity: $d_{i,i} = 0$
  - Symmetry: $d_{i,j} = d_{j,i}$
  - Triangle Inequality: $d_{i,j} \leq d_{i,k} + d_{k,j} \quad \forall j$
- Can be visualized as a graph. A clique graph with edges for each distance

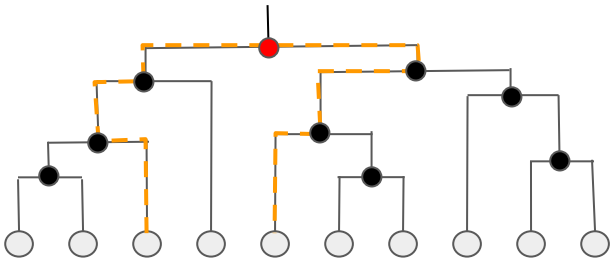
	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0



Phylogenetic Tree Problem

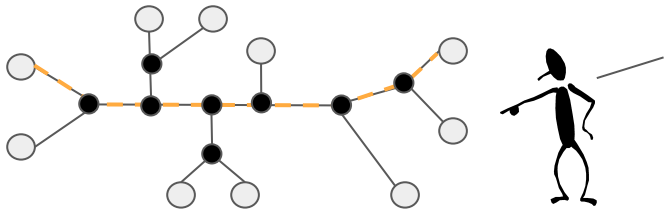


- Construct a tree graph that attempts to preserve all pairwise distances of the distance matrix
- The tree can be rooted



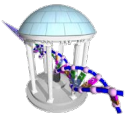
	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

- Or unrooted

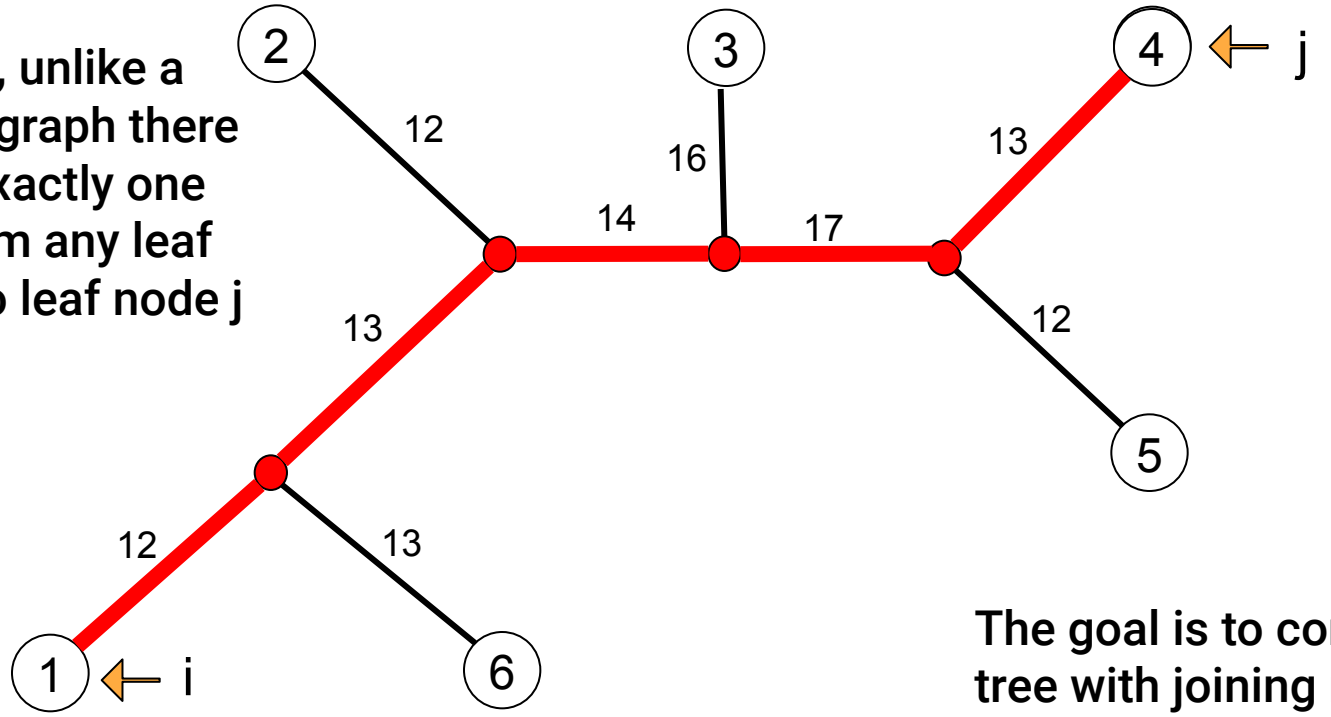


A common ancestor is inferred at the "joining nodes" of these trees. There is n-2 joining nodes in an unrooted tree and n-1 in a rooted one

Distance in Trees



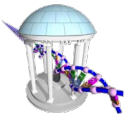
In a tree, unlike a general graph there exists exactly one path from any leaf node i to leaf node j



The goal is to construct a tree with joining nodes and edges such that all distances in the distance matrix are preserved, at least approximately.

$$d_{1,4} = 12 + 13 + 14 + 17 + 13 = 69$$

Fitting a Tree to a Distance Matrix



- D_{ij} may be defined as the "distance" between a gene in species *i* and species *j*, where the gene of interest is sequenced for all *n* species.

  D_{ij} – ***distance between i and j***

- Note the difference with

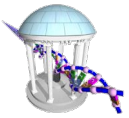
  $d_{ij}(T)$ – ***tree distance between i and j***

Lengths of path in an (***unknown***) tree *T*

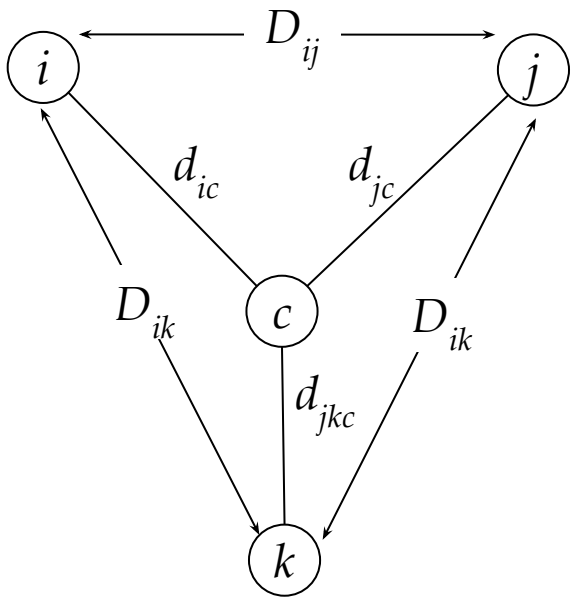
- Fitting means $D_{ij} \cong d_{ij}(T)$

Distance between species (***given***)

Simple Case a 3-leaved tree



- Tree reconstruction for any 3x3 matrix is straightforward
- We have 3 leaves *i, j, k* and a center vertex *c*



Observe:

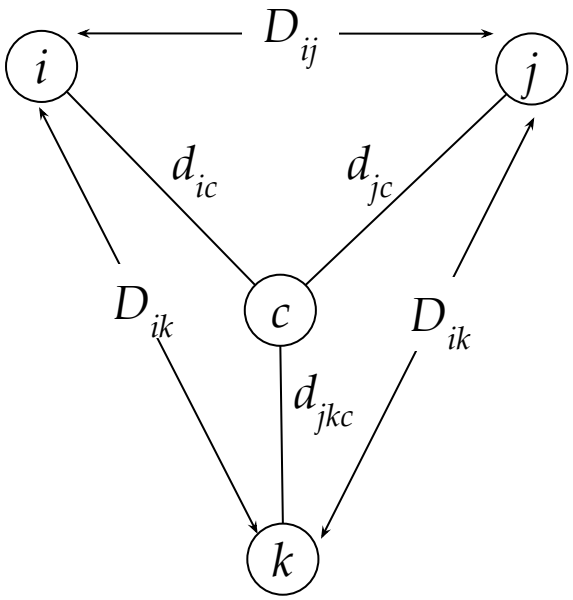
$$d_{ic} + d_{jc} = D_{ij}$$

$$d_{ic} + d_{kc} = D_{ik}$$

$$d_{jc} + d_{kc} = D_{jk}$$

3 linear equations with
3 unknowns (d_{ic}, d_{jc}, d_{kc}).

Reconstructing a 3 Leaved Tree (cont'd)



$$d_{ic} + d_{jc} = D_{ij}$$

$$+ \quad d_{ic} + d_{kc} = D_{ik}$$

$$2d_{ic} + d_{jc} + d_{kc} = D_{ij} + D_{ik}$$

$$2d_{ic} + D_{jk} = D_{ij} + D_{ik}$$

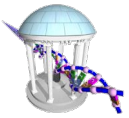
$$\boldsymbol{d_{ic} = (D_{ij} + D_{ik} - D_{jk})/2}$$

Similarly,

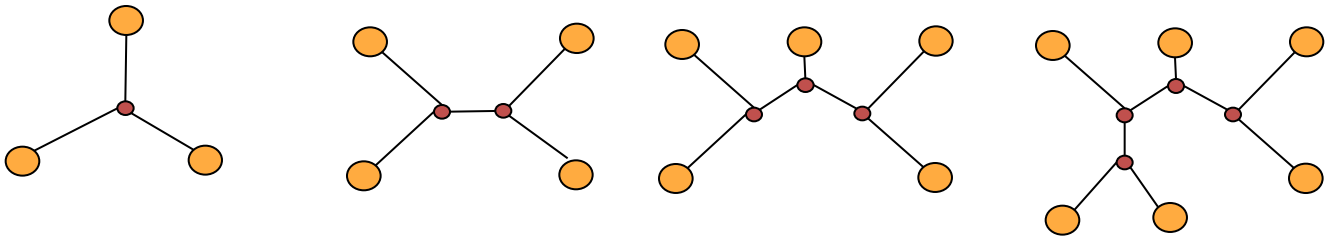
$$\boldsymbol{d_{jc} = (D_{ij} + D_{jk} - D_{ik})/2}$$

$$\boldsymbol{d_{kc} = (D_{ki} + D_{kj} - D_{ij})/2}$$

Trees with > 3 Leaves



- An unrooted tree with *n* leaves has *2n-3* edges*



- This means fitting a given tree to a distance matrix *D* requires solving a system of "n choose 2" or ½ x(x-1) equations with *2n-3* variables (over-specified)

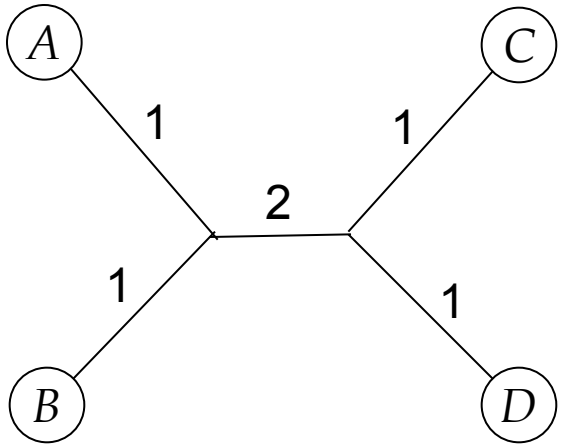
- This is not always possible to solve for *n* > 3 given arbitrary/noisy distances

* Assumes all internal nodes are of degree 3 (i.e. a node is arrived to along one edge and separates into 2 cases by mutation)

Additive Distance Matrices

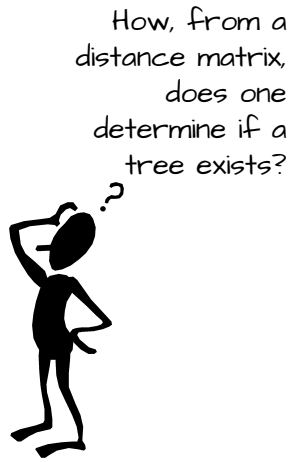
Definition: Matrix *D* is ***Additive*** if there exists a tree *T* with $d_{ij}(T) = D_{ij}$ *for all i,j*

	A	B	C	D
A	0	2	4	4
B	2	0	4	4
C	4	4	0	2
D	4	2	2	0

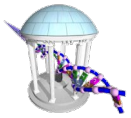


NON-ADDITIVE otherwise

	A	B	C	D
A	0	2	2	2
B	2	0	3	2
C	2	3	0	2
D	2	2	2	0



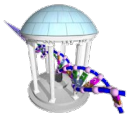
Distance Based Phylogeny Problem



- Goal: Reconstruct an evolutionary tree from a distance matrix
- Input: n x n distance matrix D_{ij}
- Output: weighted tree T with n leaves fitting D

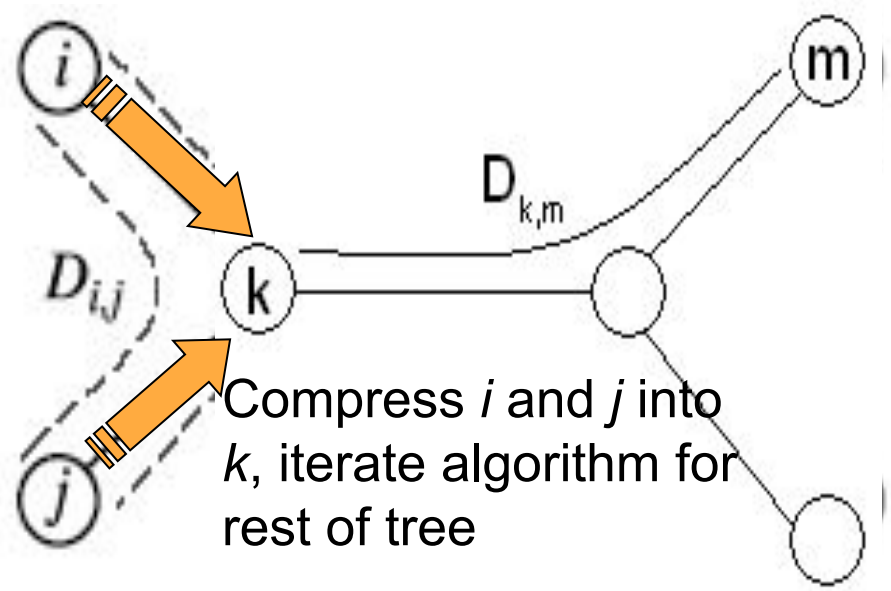
- If D is additive, this problem has a solution and there is a simple algorithm to solve it

Using Neighboring Leaves

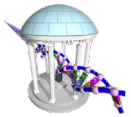


- Find ***neighboring leaves*** *i* and *j* with common parent *k*
- Remove the rows and columns of *i* and j
- Add a new row and column corresponding to *k*, where the distance from *k* to any other leaf *m* can be computed as:

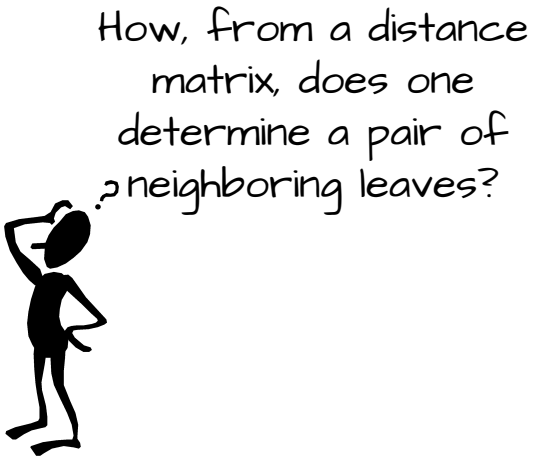
$$D_{km} = (D_{im} + D_{jm} - D_{ij})/2$$



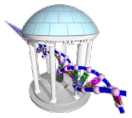
Finding Neighboring Leaves



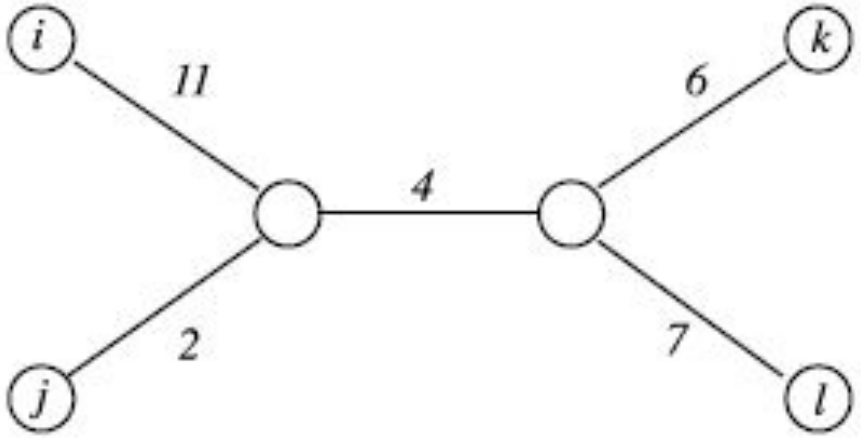
- Or solution assumes that we can easily find neighboring leaves given only distance values
- How might one approach this problem?
- It is not as easy as selecting a pair of closest leaves.



Neighbors might not be close

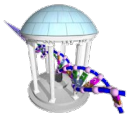


- Close leaves aren't necessarily neighbors
- *i* and *j* are neighbors, but (d_{ij} = 13) > (d_{jk} = 12)



- Finding a pair of neighboring leaves is a nontrivial problem! (we'll return to it later)

Neighbor Joining

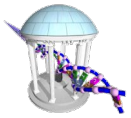


- In 1987 Naruya Saitou and Masatoshi Nei developed a neighbor joining approach for phylogenetic tree reconstruction

- **Finds a pair of leaves that are close to each other but far from other leaves:** implicitly finds a pair of neighboring leaves

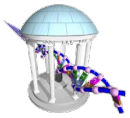
- Advantages: works well for additive and other non-additive matrices, it does not require a "molecular clock" assumption

Degenerate Triples

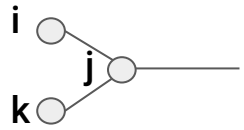


- A degenerate triple is a set of three distinct elements $1 \leq i,j,k \leq n$ where $D_{ij} + D_{jk} = D_{ik}$
- Called *degenerate* because it implies i, j, and k are collinear.
- Element *j* in a degenerate triple *i,j,k* lies on the evolutionary path from *i* to *k* (or is attached to this path by an edge of length 0).

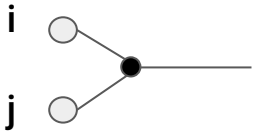
Looking for Degenerate Triples



- If distance matrix *D* **has** a degenerate triple *i,j,k* then *j* can be "removed" from *D* thus reducing the size of the problem.



- If distance matrix *D* **does not have** a degenerate triple *i,j,k, one can "create"* a degenerative triple in *D* by shortening all hanging or leaf edges in the tree.



Shortening Hanging Edges

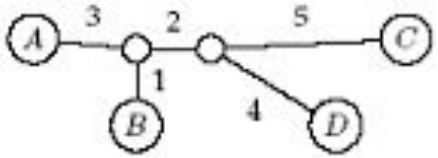
- Shorten all “hanging” edges (edges that connect leaves) until a degenerate triple is found

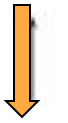
	A	B	C	D
A	0	4	10	9
B	4	0	8	7
C	10	8	0	9
D	9	7	9	0

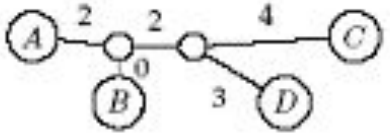
$\delta = 1$

	A	B	C	D
A	0	2	8	7
B	2	0	6	5
C	8	6	0	7
D	7	5	7	0

$i \leftarrow A$
$j \leftarrow B$
$k \leftarrow C$

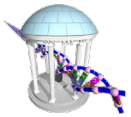






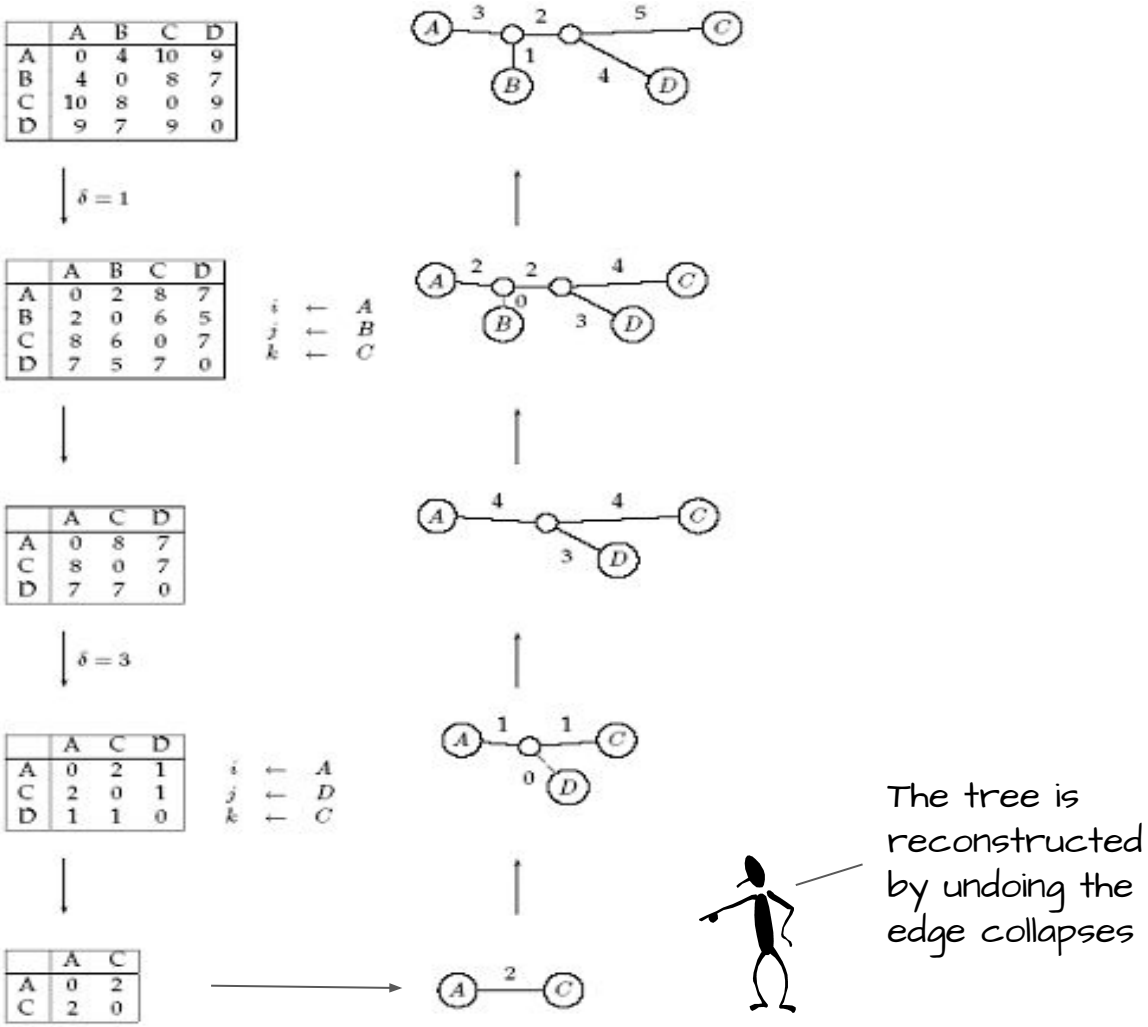
Now (A,B,D) and (A,B,C) are degenerate

Finding Degenerate Triples

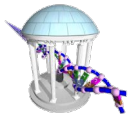


- If there is no degenerate triple, all hanging edges are reduced by the same amount δ, so that all pair-wise distances in the matrix are reduced by 2δ.
- Eventually this process collapses one of the leaves (when δ = length of shortest hanging edge), forming a degenerate triple *i,j,k* and reducing the size of the distance matrix *D*.
- The attachment point for *j* can be recovered in the reverse transformations by saving D_{ij} for each collapsed leaf.

Reconstructing Trees

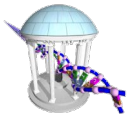


AdditivePhylogeny Algorithm



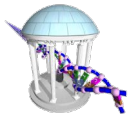
1. AdditivePhylogeny(D)
2. if D is a 2 x 2 matrix
3. T = tree of a single edge of length $D_{1,2}$
4. return T
5. if D is non-degenerate
6. δ = trimming parameter of matrix D
7. for all $1 \leq i \neq j \leq n$
8. $D_{ij} = D_{ij} - 2\delta$
9. else
10. $\delta = 0$

AdditivePhylogeny (cont'd)



1. Find a triple i, j, k in D such that $D_{ij} + D_{jk} = D_{ik}$
2. $x = D_{ij}$
3. Remove j^{th} row and j^{th} column from D
4. T = AdditivePhylogeny(D)
5. Add a new vertex v to T at distance x from i to k
6. Add j back to T by creating an edge (v,j) of length 0
7. for every leaf l in T
8. if distance from l to v in the tree $\neq D_{l,j}$
9. output "matrix is not additive"
10. return
11. Extend all "hanging" edges by length δ
12. return T

The Four Point Condition

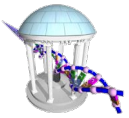


- AdditivePhylogeny Algorithm provides a way to check if distance matrix *D* is additive (i.e. it converges to a single 2 by 2 matrix)

- An even more efficient check for additivity is the "four-point condition", which can be tested before running AdditivePhylogeny()

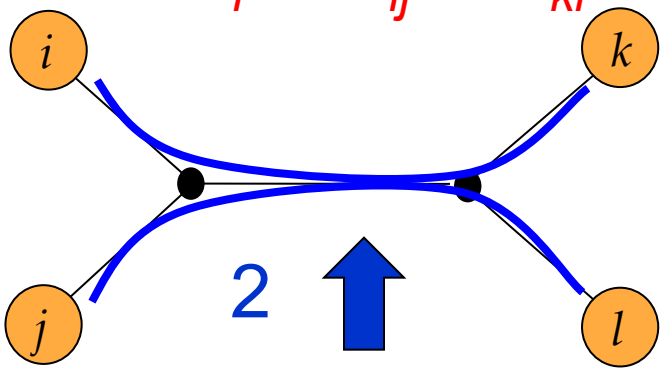
- Let $1 \leq i,j,k,l \leq n$ be four distinct leaves in a tree

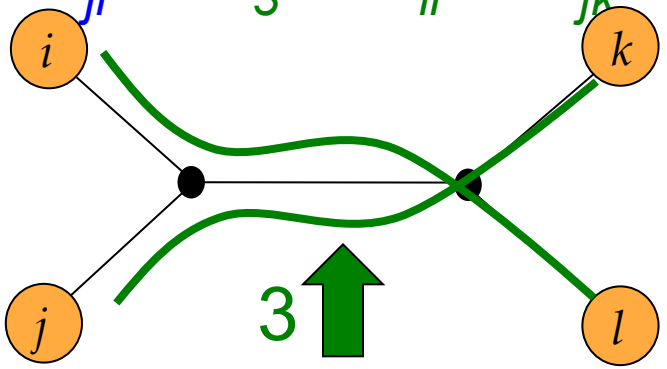
The Four Point Condition (cont'd)



Given 6 distances (D_{ij}, D_{ik}, D_{il}, D_{jk}, D_{jl}, D_{kl}):

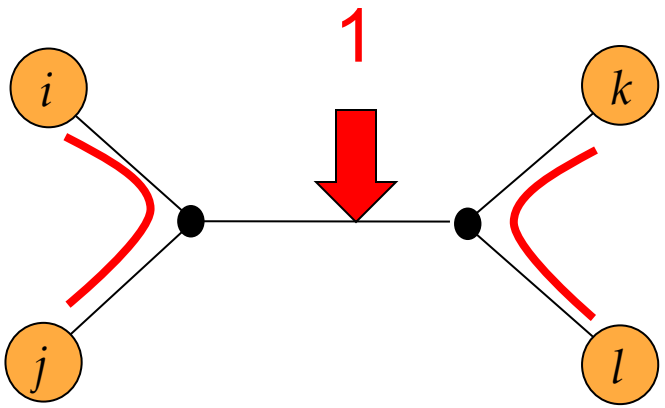
1. $t_1 = D_{ij} + D_{kl}$, 2. $t_2 = D_{ik} + D_{jl}$, 3. $t_3 = D_{il} + D_{jk}$





2 and **3** give the **same sum**: $t_2 == t_3$

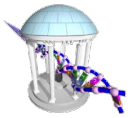
The length of all edges + the middle edge twice



And, $t_1 < t_2, t_3$

The length of all edges – the middle edge

The Four Point Condition: Theorem

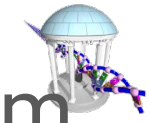


- The four point condition for the quartet *i,j,k,l* is satisfied if two of these sums are the same, with the third sum smaller than these first two. How many tests?

$$_nC_4 = n!/ (4! (n-4!)) = n(n-1)(n-2)(n-4)/24$$

- ***Theorem*** : An *n* x *n* matrix *D* is additive if and only if the four point condition holds for ***every*** quartet $1 \leq i,j,k,l \leq n$

Least-Squares Distance Phylogeny Problem



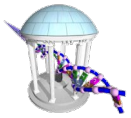
- If the distance matrix D is NOT additive, then we look for a tree T that approximates D the best:

$$\textbf{\textit{Squared Error}}: \quad \sum_{i,j} (d_{ij}(T) - D_{ij})^2$$

- Squared Error is a measure of the quality of the fit between distance matrix and the tree: we want to minimize it.

- **Least Squares Distance Phylogeny Problem**: finding the best approximation tree T for a non-additive matrix D.

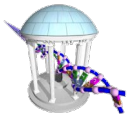
UPGMA



- Unweighted Pair Group Method with Arithmetic Mean (UPGMA)

- UPGMA is a hierarchical clustering algorithm:
  - assigns the distance between clusters to be the average pairwise distance
  - assigns a *height* to every vertex in the tree, that is midway between the cluster distances

UPGMA's Weakness

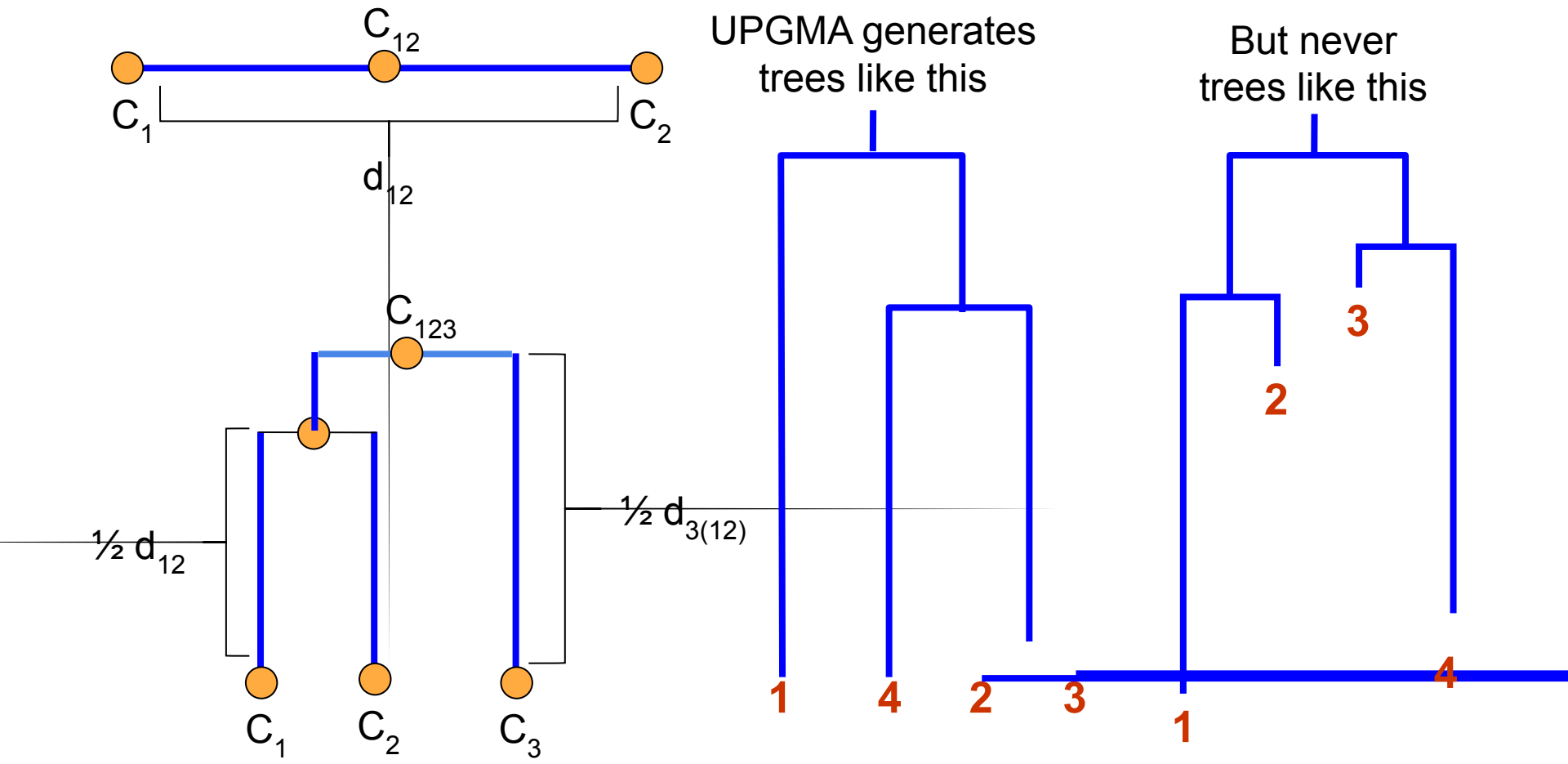


- The algorithm produces an *ultrametric rooted* tree : the distance from the root to every leaf is the same

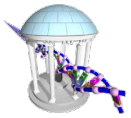
- UPGMA models a constant molecular clock:
  - all species represented by the leaves in the tree
  - assumed to coexist at t=0 and to have accumulated mutations (and thus evolve) at the same rate.

- In reality the assumptions of UPGMA are seldom true, but they are frequently approximately true.

“Balanced” Cluster Merging



Clustering in UPGMA



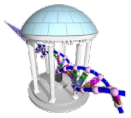
Given two disjoint clusters C_i, C_j of sequences,

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{\substack{p \in C_i \\ q \in C_j}} d_{pq}$$

Note that if $C_k = C_i \cup C_j$, then the distance to another cluster C_l is:

$$d_{kl} = \frac{d_{il}|C_i| + d_{jl}|C_j|}{|C_i| + |C_j|}$$

UPGMA Algorithm



Initialization:

Assign each x_i to its own cluster C_i

Define one leaf per sequence, each at height 0

Iteration:

Find two clusters C_i and C_j such that d_{ij} is min

Let $C_k = C_i \cup C_j$

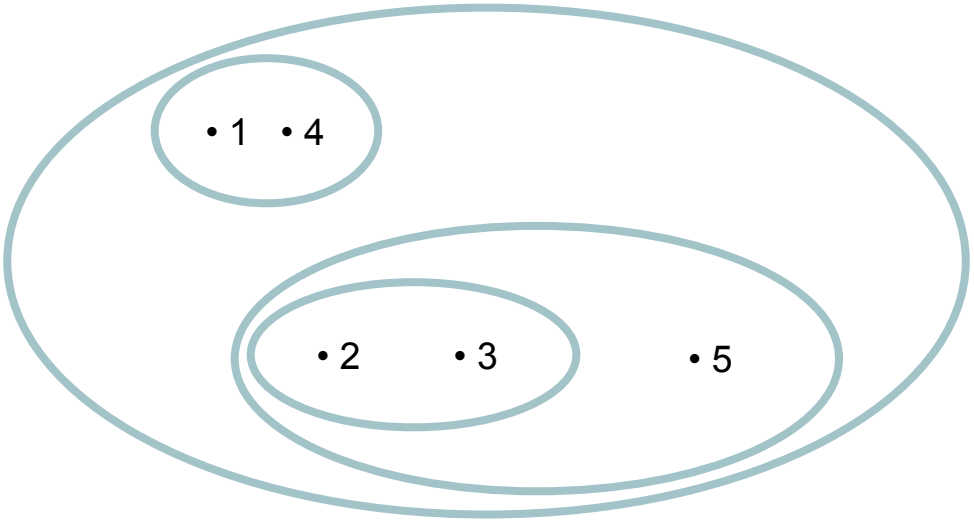
Add a vertex connecting C_i, C_j *and* place it at height $d_{ij}/2$

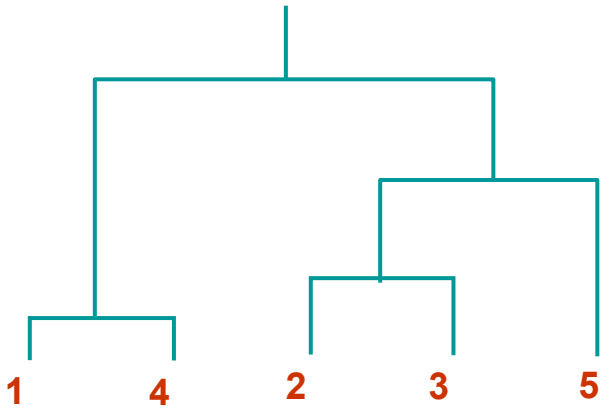
Delete C_i *and* C_j

Termination:

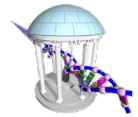
When a single cluster remains

UPGMA Algorithm (cont'd)

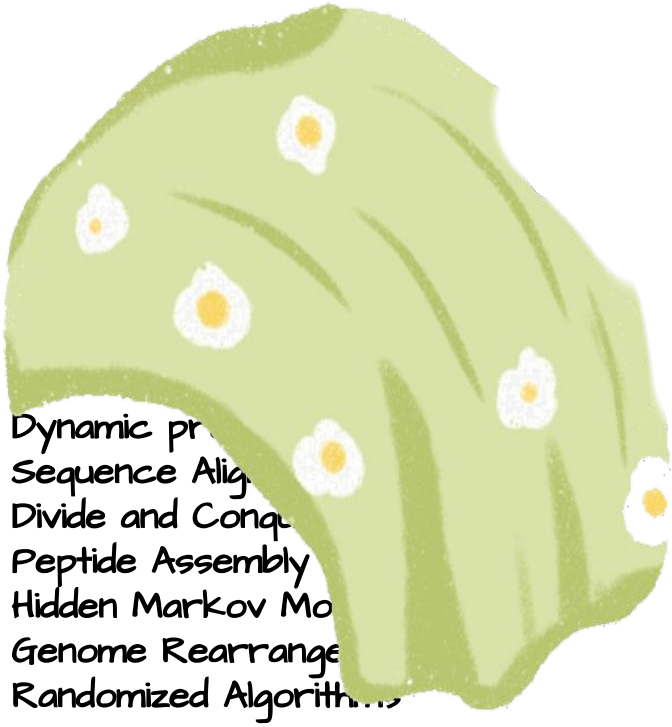




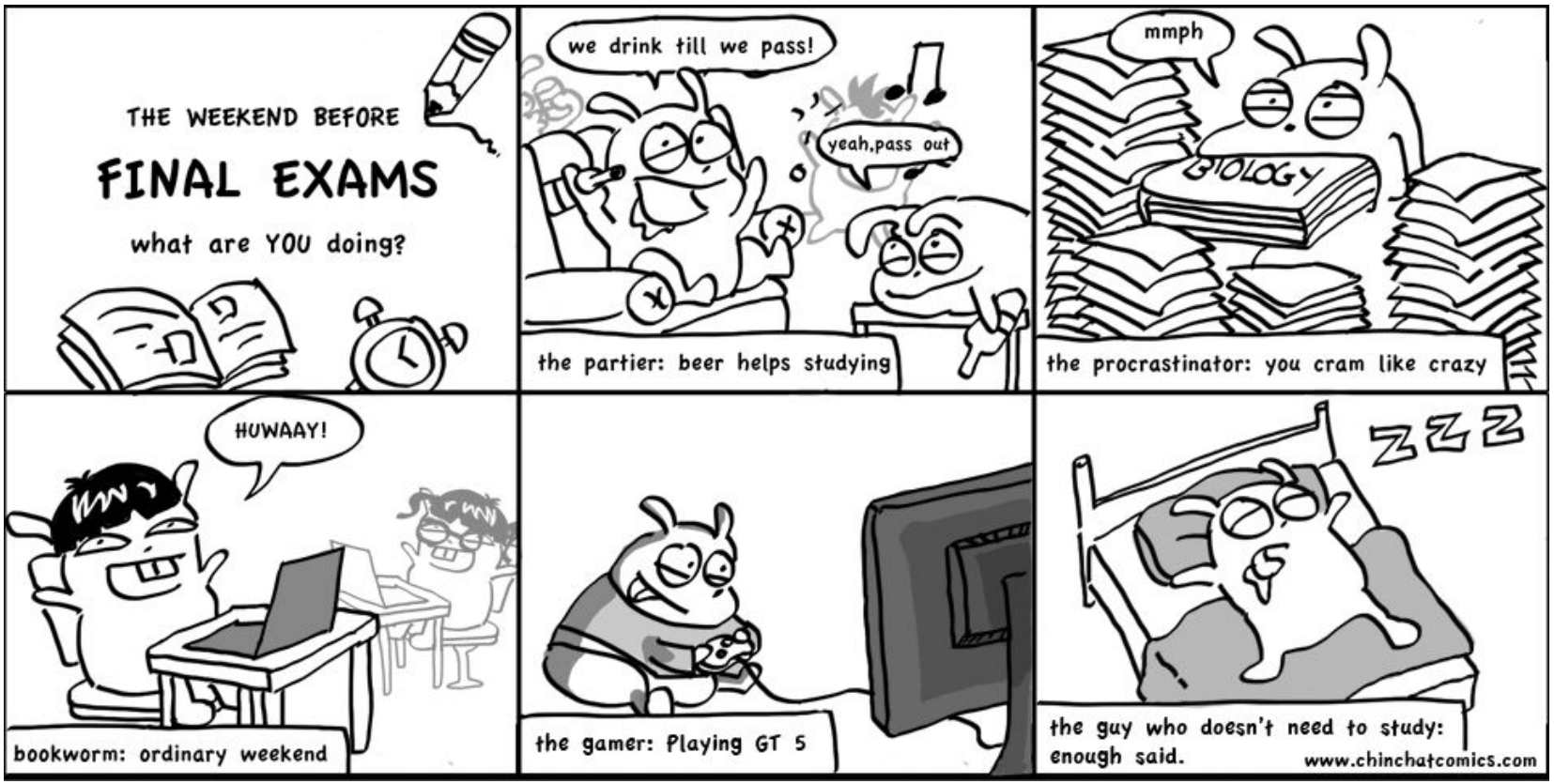
What will be covered on Final



- The Final will focus on material since the midterm, but since the course builds upon topics you may be asked to draw upon knowledge from the first half of the course
- Exam will be designed for 2 hrs (~15 questions), but you will have the entire 3 hrs to complete it.
- Make sure that your ONYEN and PID are correctly filled in on your exam's signature cell!
- Don't leave questions unanswered.

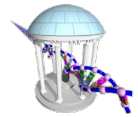


Studying for the Exam



- Lecture slides
- Lecture notebooks
- Problem sets

It's Over



- Final Thursday, 5/5
  - noon - 3pm
  - Be sure to sign into zoom
  - Open book, open notes, open internet, online
  - Will focus on material since midterm
  - Final Study session:
    - Thursday 4/28, 2:00-3:15pm