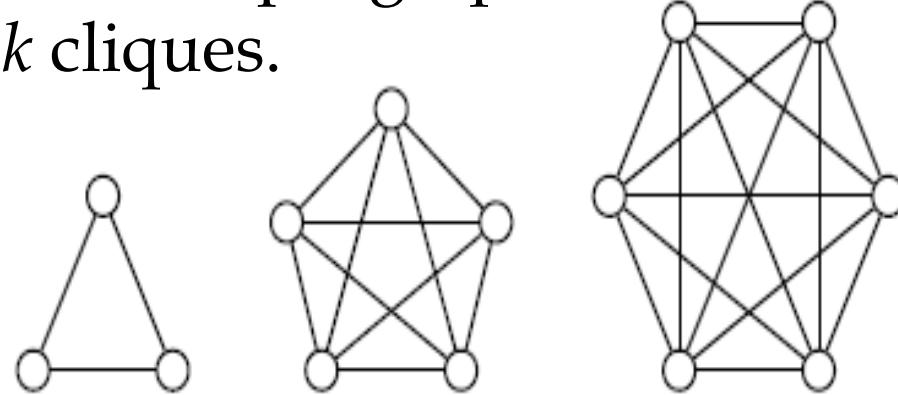


Lecture 20: Clustering and Evolution

Study Chapter 10.4 – 10.8

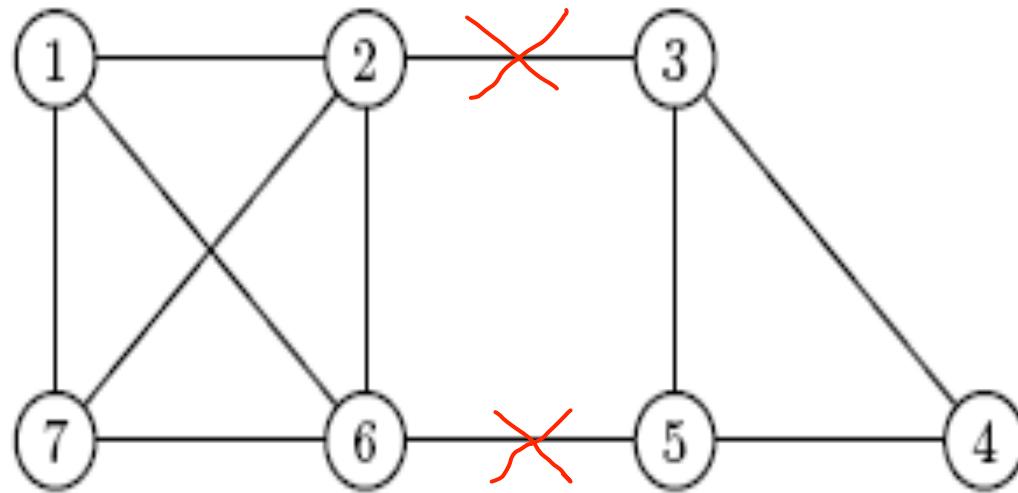
Clique Graphs

- A **clique** is a graph where every vertex is connected via an edge to every other vertex
- A **clique graph** is a graph where each connected component is a clique
- The concept of clustering is closely related to clique graphs. Every partition of n elements into k clusters can be represented as a clique graph on n vertices with k cliques.



Transforming Graphs into a Clique Graphs

- Clusters are maximal cliques (cliques not contained in any other complete subgraph)
 - 1,6,7 is a non-maximal clique.
- An arbitrary graph can be transformed into a clique graph by adding or removing edges



Corrupted Cliques Problem



Determine the smallest number of edges that need be added or removed to transform a graph to a clique graph

Input: A graph G

Output: The smallest number of edge additions and/or removals that transforms G into a clique graph



Distance Graphs



- One can turn a distance matrix into a distance graph
 - Genes or Species are vertices of the graph
 - Choose a distance threshold θ
 - If the distance between two vertices is below θ , draw an edge between them
 - The resulting graph may contain cliques
 - These cliques represent clusters of closely located data points!



Transforming Distance Graph into Clique Graph

The distance graph (threshold $\theta=7$) is transformed into a clique graph after removing the two highlighted edges

	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

(a) Distance matrix, d (distances shorter than 7 are shown in bold).

After transforming the distance graph into the clique graph, the dataset is partitioned into three clusters

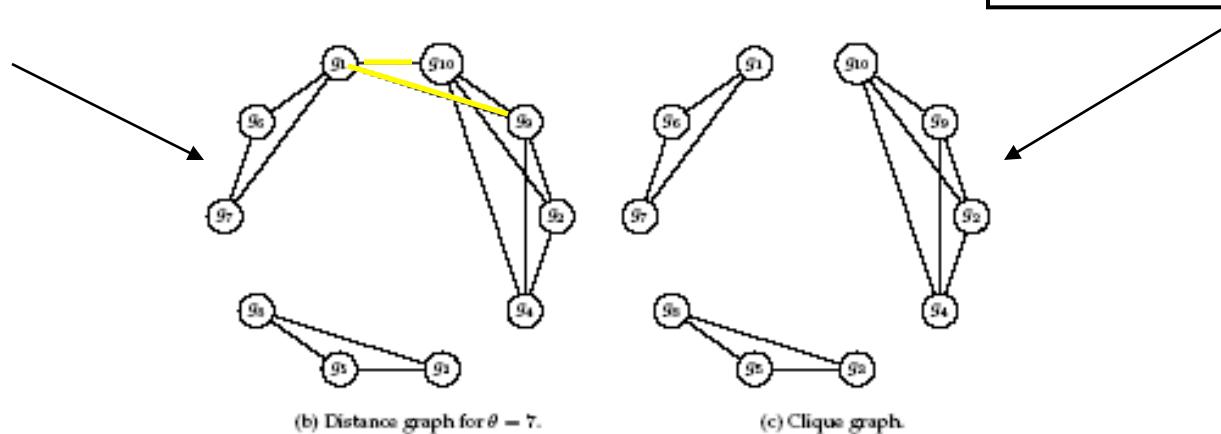


Figure 10.6 The distance graph (b) for $\theta = 7$ is not quite a clique graph. However, it can be transformed into a clique graph (c) by removing edges (g_1, g_{10}) and (g_1, g_9) .



Heuristics for Corrupted Clique Problem



- Corrupted Cliques problem is NP-Hard, some heuristics exist to approximately solve it:
- **CAST** (Cluster Affinity Search Technique): a practical and fast algorithm:
 - **CAST** is based on the notion of genes *close* to cluster C or *distant* from cluster C
 - Distance between gene i and cluster C :

$d(i, C)$ = average distance between gene i and *all* genes in C

Gene i is *close* to cluster C if $d(i, C) < \theta$ and *distant* otherwise



CAST Algorithm

```
1. CAST( $S, G, \theta$ )
2.  $P \leftarrow \emptyset$ 
3. while  $S \neq \emptyset$ 
4.    $v \leftarrow$  vertex of maximal degree in the distance graph  $G$ 
5.    $C \leftarrow \{v\}$ 
6.   while a close gene  $i$  not in  $C$  or distant gene  $i$  in  $C$  exists
7.     Find the nearest close gene  $i$  not in  $C$  and add it to  $C$ 
8.     Remove the farthest distant gene  $i$  in  $C$ 
9.   Add cluster  $C$  to partition  $P$ 
10.   $S \leftarrow S \setminus C$ 
11.  Remove vertices of cluster  $C$  from the distance graph  $G$ 
12. return  $P$ 
```

S – *set of elements*, G – *distance graph*, θ – *distance threshold*



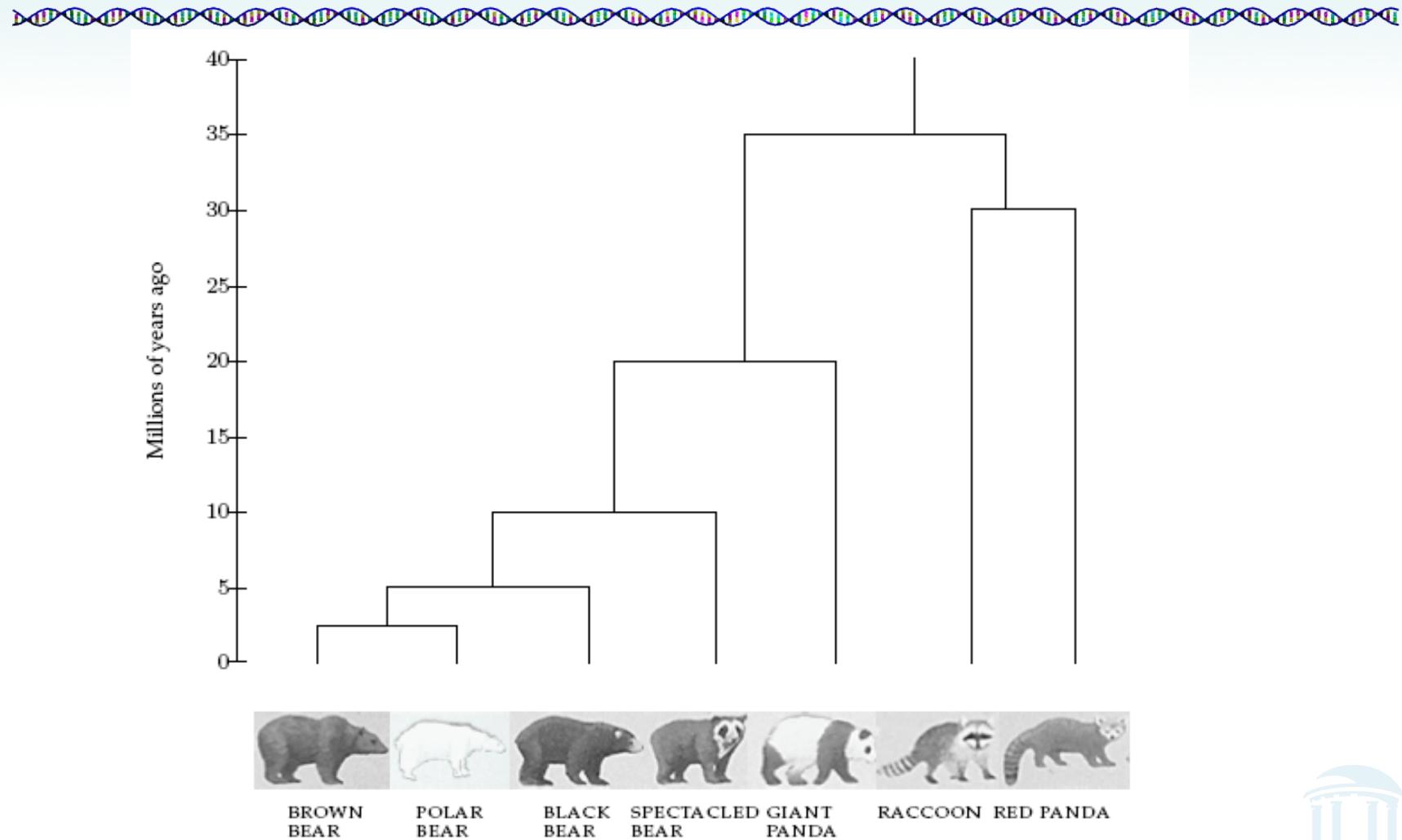
Evolution and DNA Analysis: the Giant Panda Riddle



- For roughly 100 years scientists were unable to figure out which family the giant panda belongs to
- Giant pandas look like bears but have features that are unusual for bears and typical for raccoons, e.g., they do not hibernate
- In 1985, Steven O'Brien and colleagues solved the giant panda classification problem using DNA sequences and algorithms



Evolutionary Tree of Bears and Raccoons



Evolutionary Trees: DNA-based Approach



- 40 years ago: Emile Zuckerkandl and Linus Pauling brought reconstructing evolutionary relationships with DNA into the spotlight
- In the first few years after Zuckerkandl and Pauling proposed using DNA for evolutionary studies, the possibility of reconstructing evolutionary trees by DNA analysis was hotly debated
- Now it is a dominant approach to study evolution.



Out of Africa Hypothesis



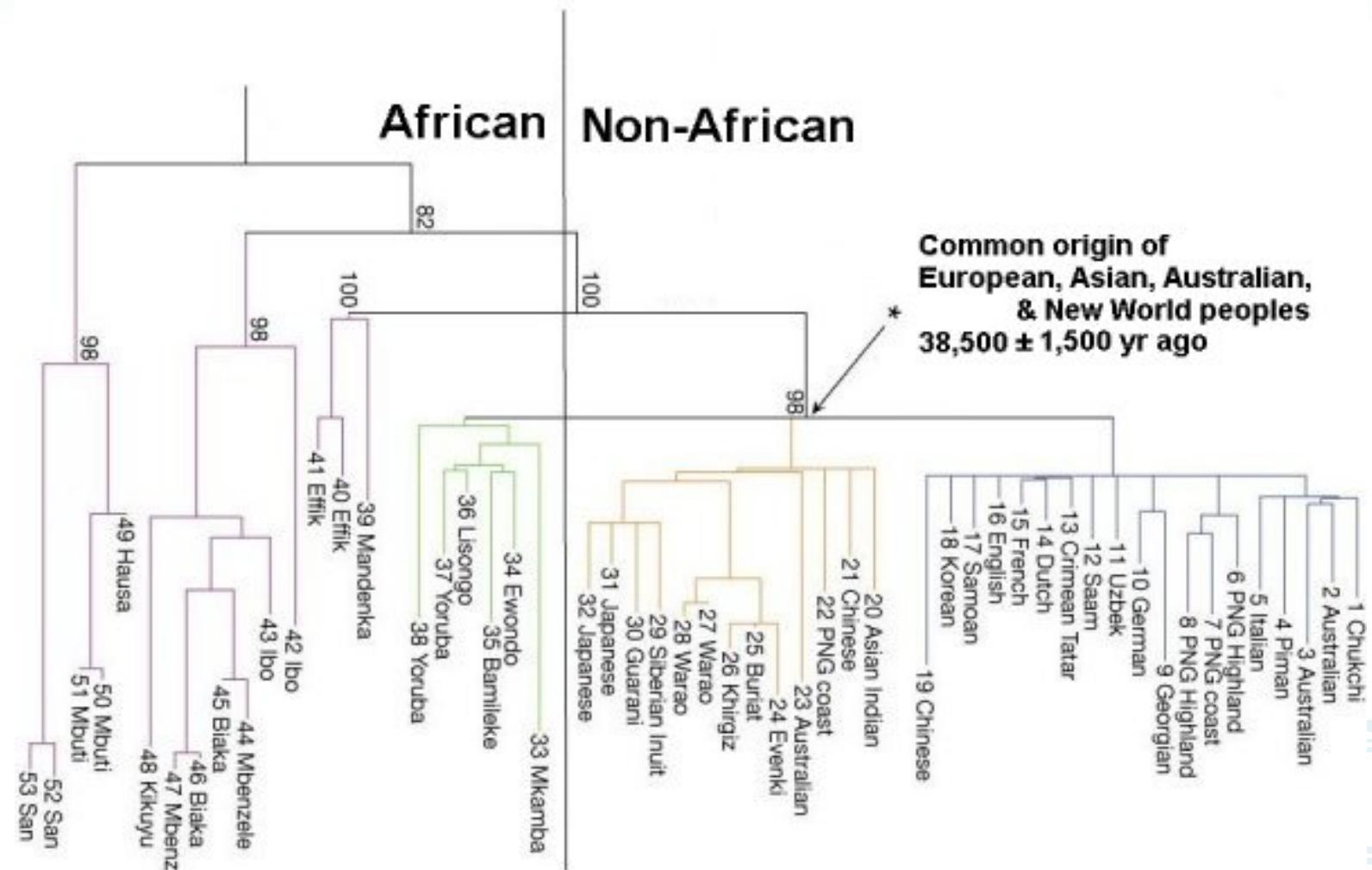
- Around the time the giant panda riddle was solved, a DNA-based reconstruction of the human evolutionary tree led to the **Out of Africa Hypothesis** that claims our most ancient ancestor lived in Africa roughly 200,000 years ago
- Largely based on mitochondrial DNA



Human Evolutionary Tree (cont'd)



http://www.mun.ca/biology/scarr/Out_of_Africa2.htm



The Origin of Humans: "Out of Africa" vs Multiregional Hypothesis



Out of Africa:

- Humans evolved in Africa ~150,000 years ago
- Humans migrated out of Africa, replacing other humanoids around the globe
- There is no direct descendence from Neanderthals

Multiregional:

- Humans evolved in the last two million years as a single species. Independent appearance of modern traits in different areas
- Humans migrated out of Africa mixing with other humanoids on the way
- There is a genetic continuity from Neanderthals to humans



mtDNA analysis supports “Out of Africa” Hypothesis



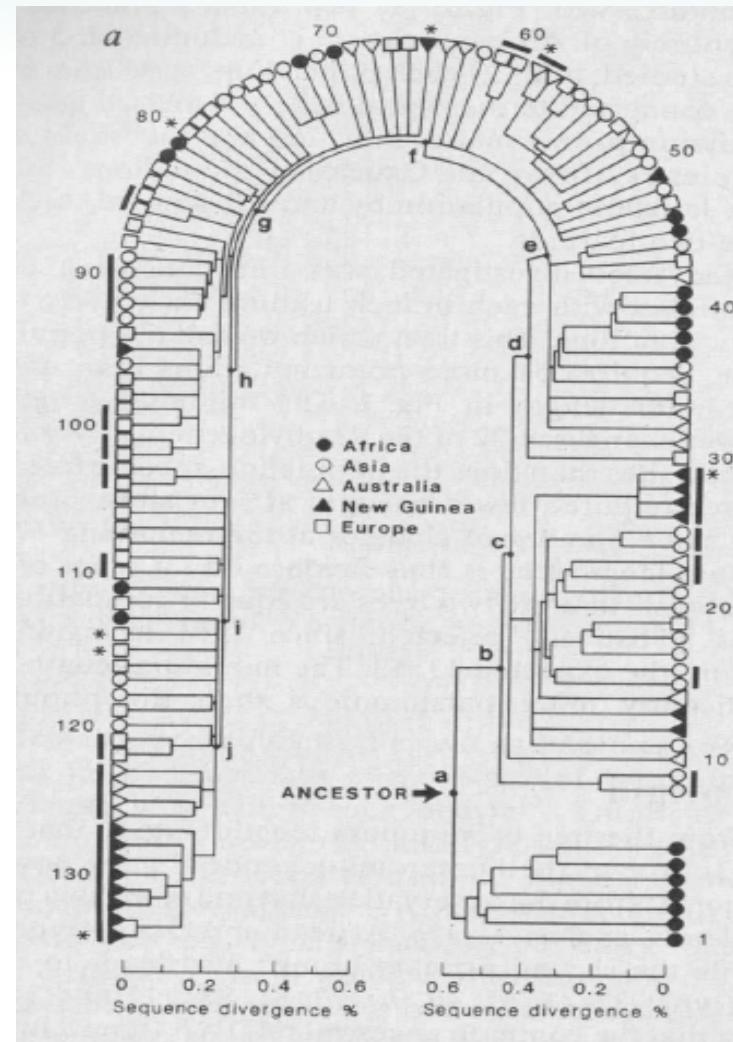
- African origin of humans inferred from:
 - African population was the most diverse (sub-populations had more time to diverge)
 - The evolutionary tree separated one group of Africans from a group containing all five populations.
 - Tree was rooted on branch between groups of greatest difference.



Evolutionary Tree of Humans (mtDNA)



The evolutionary tree separates one group of Africans from a group containing all five populations.

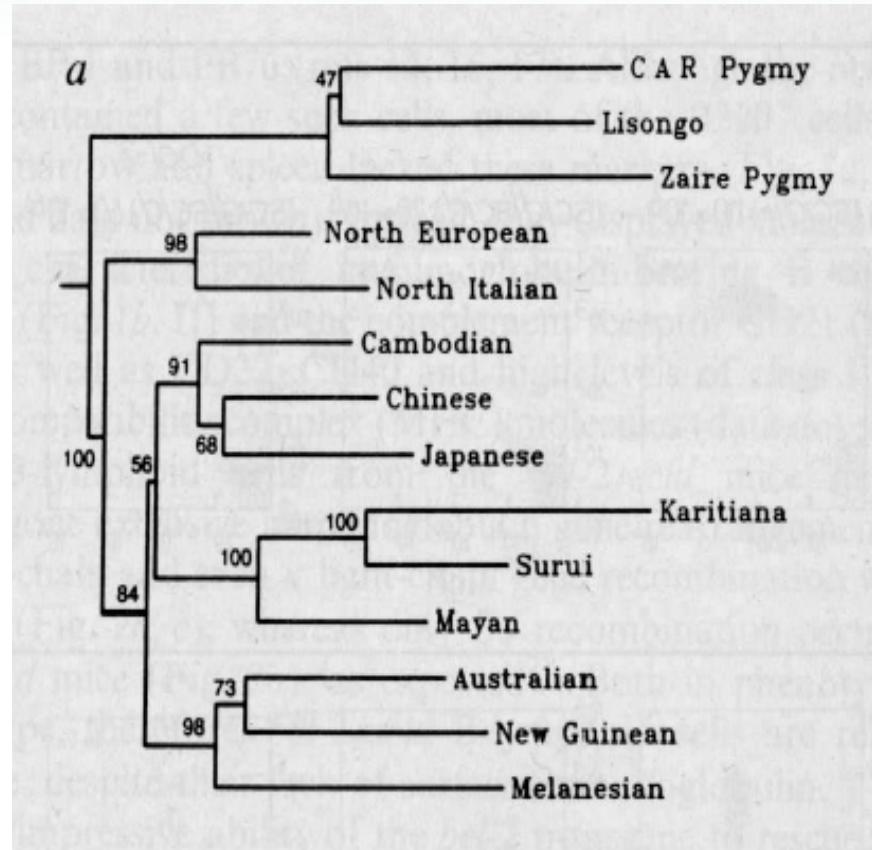


Vigilant, Stoneking, Harpending, Hawkes, and Wilson (1991)

Evolutionary Tree of Humans: (microsatellites)



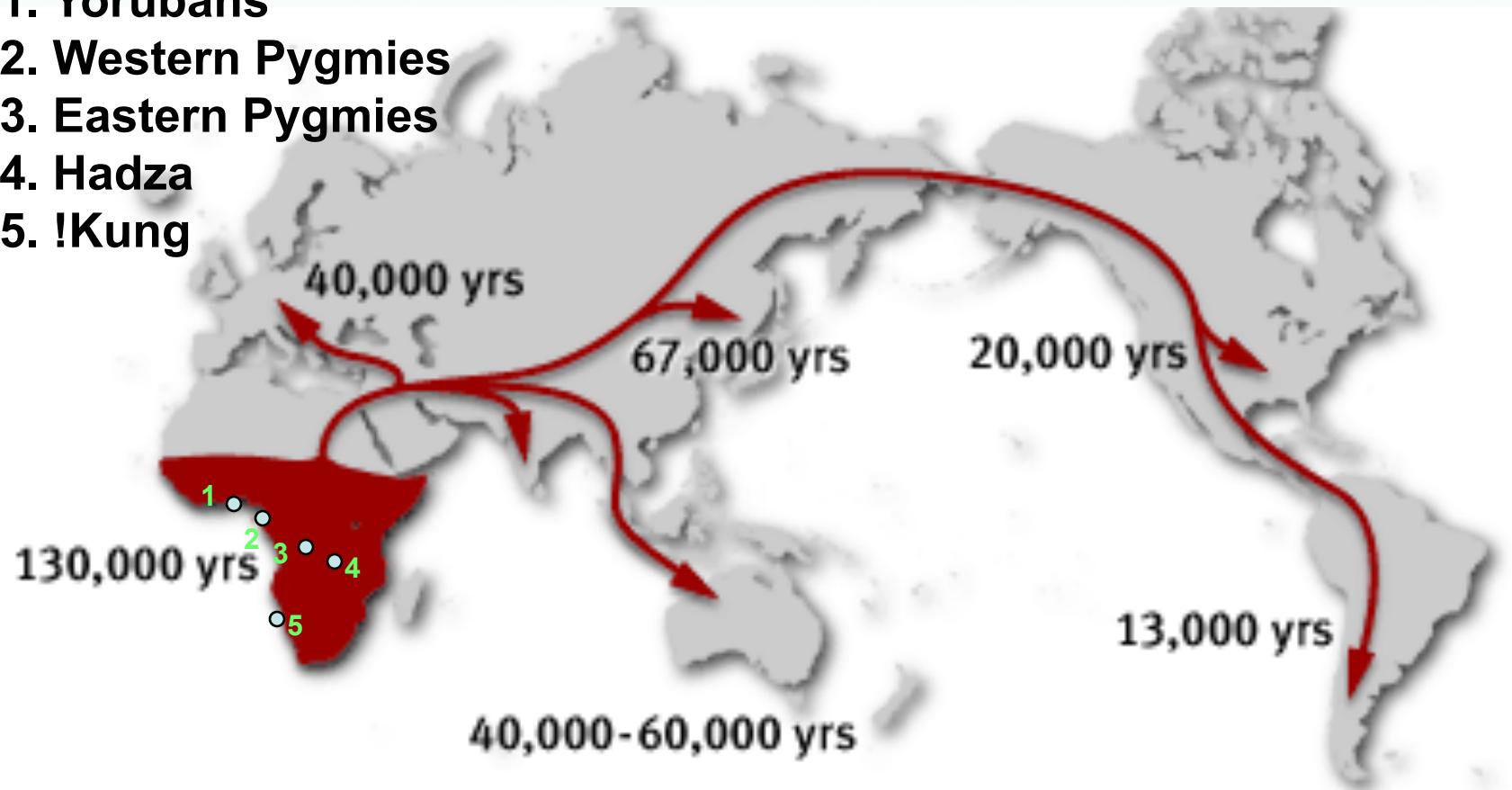
- Neighbor joining tree for 14 human populations genotyped with 30 microsatellite loci.



Human Migration Out of Africa



1. Yorubans
2. Western Pygmies
3. Eastern Pygmies
4. Hadza
5. !Kung



Evolutionary Trees



How are these trees built from DNA sequences?

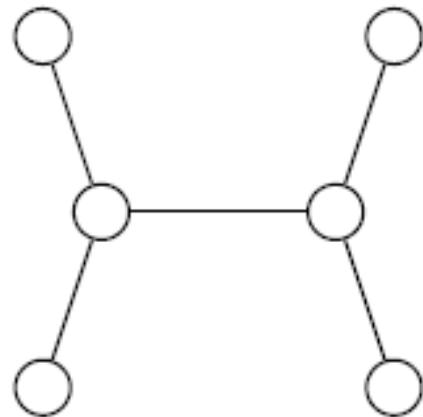
- leaves represent existing species
- internal vertices represent ancestors
- root represents the oldest evolutionary ancestor



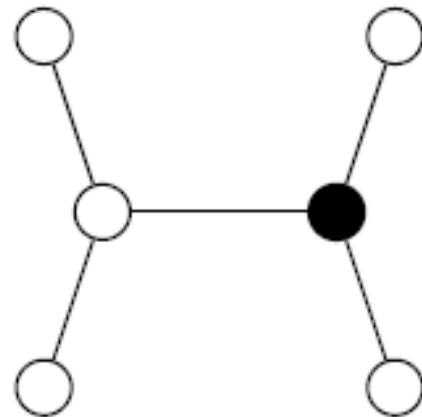
Rooted and Unrooted Trees



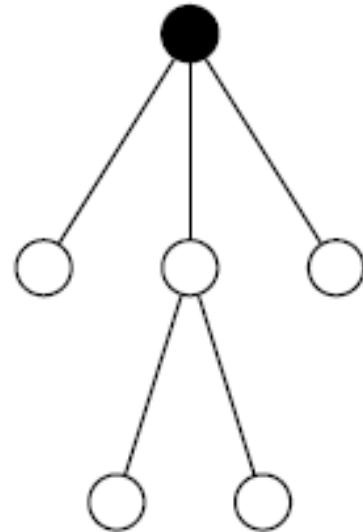
In the unrooted tree the position of the root (“oldest ancestor”) is unknown. Otherwise, they are like rooted trees



(a) Unrooted tree



(b) Rooted tree



(c) The
rooted tree



Distances in Trees

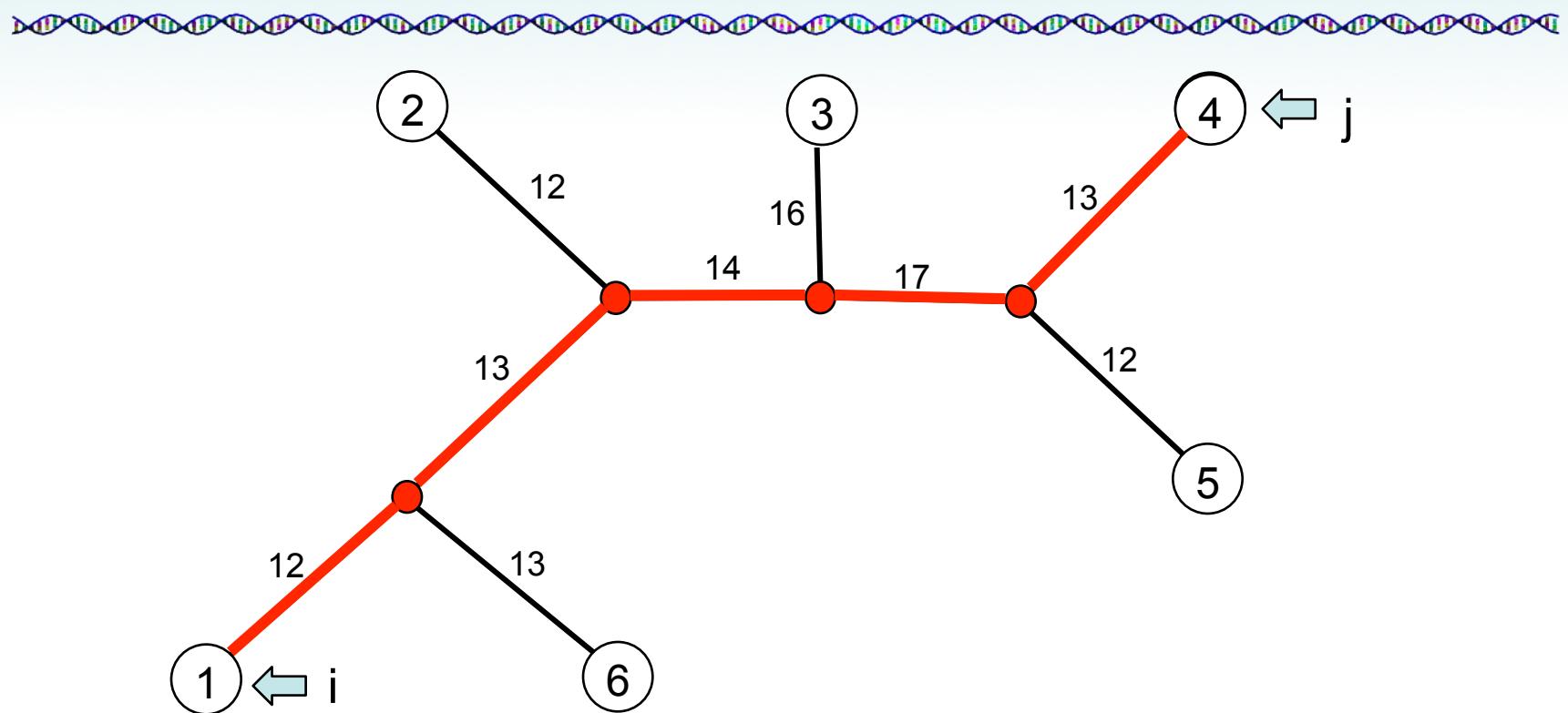


- Edges may have weights reflecting:
 - Number of mutations on evolutionary path from one species to another
 - Time estimate for evolution of one species into another
- In a tree T , we often compute

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



Distance in Trees



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



Distance Matrix



- Given n species, we can compute the $n \times n$ *distance matrix* D_{ij}
- D_{ij} may be defined as the edit distance between a gene in species i and species j , where the gene of interest is sequenced for all n species.

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



Edit Distance vs. Tree Distance



- Given n species, we can compute the $n \times n$ *distance matrix* D_{ij}
- D_{ij} may be defined as the edit distance between a gene in species i and species j , where the gene of interest is sequenced for all n species.
 D_{ij} – *edit distance between i and j*
- Note the difference with
 $d_{ij}(T)$ – *tree distance between i and j*



Fitting Distance Matrix



- Given n species, we can compute the $n \times n$ *distance matrix* D_{ij}
- Evolution of these genes is described by a tree that *we don't know*.
- We need an algorithm to construct a tree that best *fits* the distance matrix D_{ij}



Fitting Distance Matrix



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

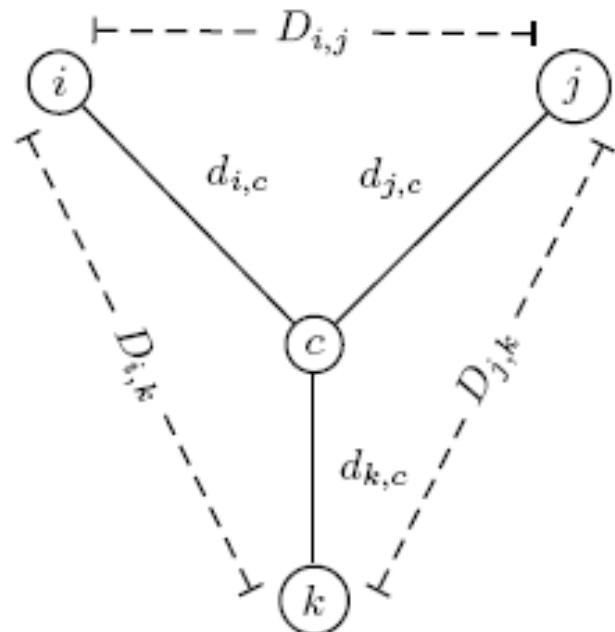
- Fitting means $D_{ij} = \underbrace{d_{ij}(T)}_{\text{Edit distance between species (*known*)}}$



Reconstructing 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:

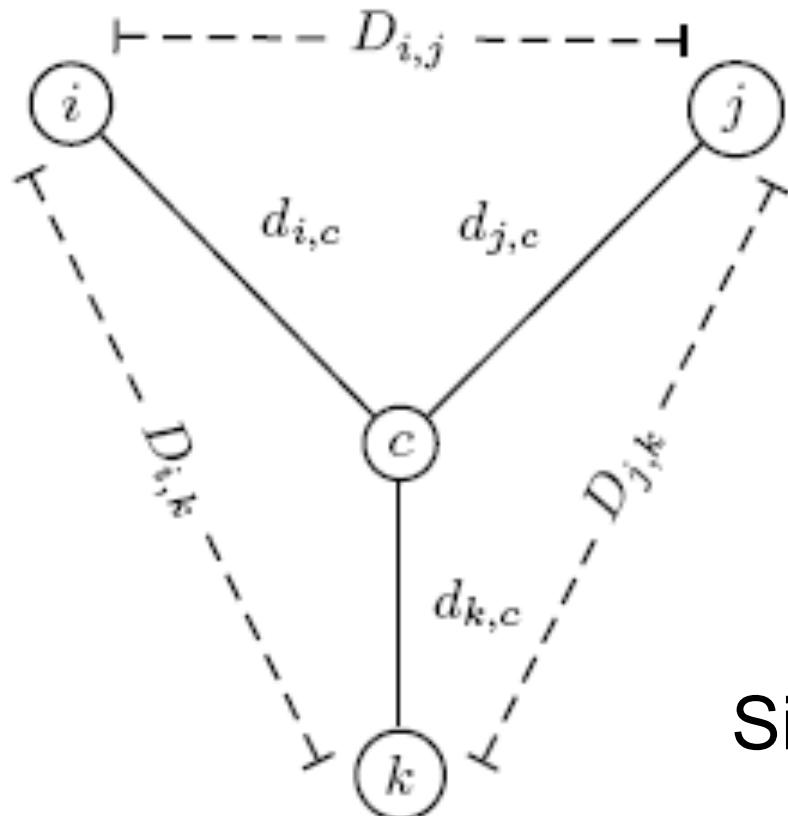
$$d_{ic} + d_{jc} = D_{ij}$$

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

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



Reconstructing a 3 Leaved Tree (cont'd)



$$\begin{aligned}
 d_{ic} + d_{jc} &= D_{ij} \\
 + \frac{d_{ic} + d_{kc}}{2} &= D_{ik} \\
 2d_{ic} + d_{jc} + d_{kc} &= D_{ij} + D_{ik} \\
 2d_{ic} + D_{jk} &= D_{ij} + D_{ik}
 \end{aligned}$$

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

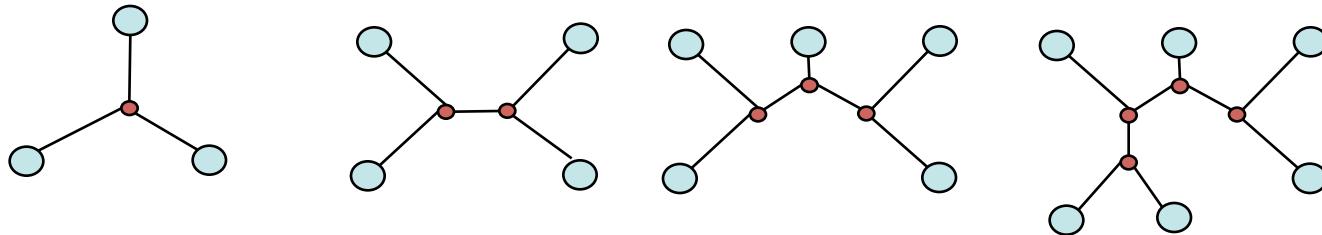
Similarly,

$$\begin{aligned}
 d_{jc} &= (D_{ij} + D_{jk} - D_{ik})/2 \\
 d_{kc} &= (D_{ki} + D_{kj} - D_{ij})/2
 \end{aligned}$$

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 $\frac{1}{2} \times (x-1)$ equations with $2n-3$ variables (over-specified)
- This is not always possible to solve for $n > 3$ given arbitrary/noisy distances

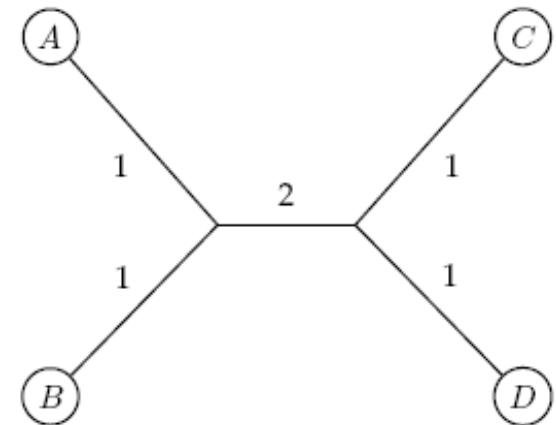
* Assumes all internal node are of degree 3
(i.e. all mutations separate into 2 cases)



Additive Distance Matrices

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	4	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



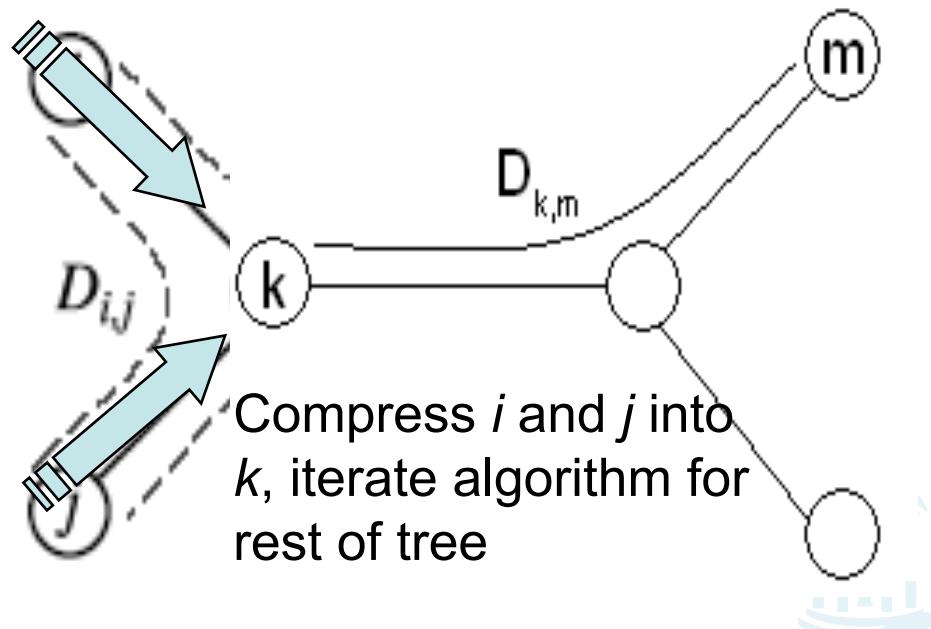
- Goal: Reconstruct an evolutionary tree from a distance matrix
- Input: $n \times n$ distance matrix D_{ij}
- Output: weighted tree T with n leaves fitting D
- If D is additive, this problem has a solution and there is a simple algorithm to solve it



Use Neighboring Leaves to Construct the Tree

- 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



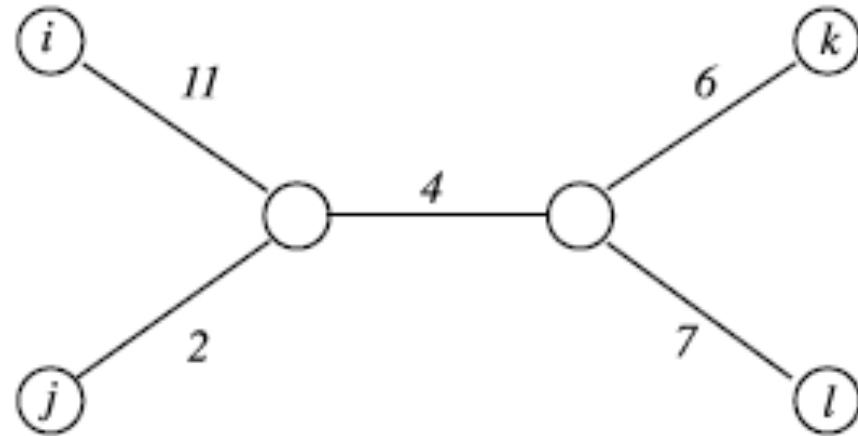
- Our solution assumes that we can easily find neighboring leaves given only distance values
- How might one approach this problem?
- It is not as easy as selecting a pair of closest leaves.



Finding Neighboring Leaves



- Closest 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 Algorithm



- In 1987 Naruya Saitou and Masatoshi Nei developed a neighbor joining algorithm 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 have the flawed 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

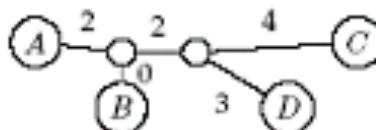
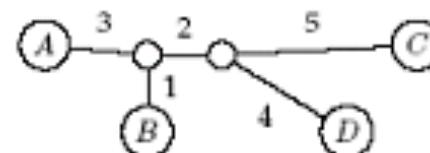
- Shorten all “hanging” edges (edges that connect leaves) until a degenerate triple is found

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

$$\downarrow \delta = 1$$

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

$$\begin{array}{l} i \leftarrow A \\ j \leftarrow B \\ k \leftarrow C \end{array}$$



Now (A,B,D)
are degenerate



Finding Degenerate Triples



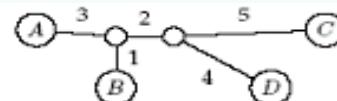
- If there is no degenerate triple, all hanging edges are reduced by the same amount δ , so that all pair-wise distances in the matrix are reduced by 2δ .
- Eventually this process collapses one of the leaves (when $\delta = \text{length of shortest hanging edge}$), forming a degenerate triple i,j,k 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 for Additive Distance Matrices



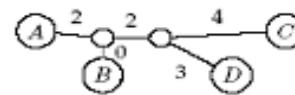
	A	B	C	D
A	0	4	10	9
B	4	0	8	7
C	10	8	0	9
D	9	7	9	0



$\downarrow \delta = 1$

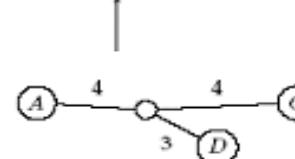
	A	B	C	D
A	0	2	8	7
B	2	0	6	5
C	8	6	0	7
D	7	5	7	0

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



\downarrow

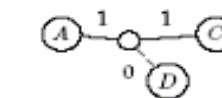
	A	C	D
A	0	8	7
C	8	0	7
D	7	7	0



$\downarrow \delta = 3$

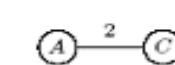
	A	C	D
A	0	2	1
C	2	0	1
D	1	1	0

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



\downarrow

	A	C
A	0	2
C	2	0



AdditivePhylogeny Algorithm



1. **AdditivePhylogeny(D)**
2. **if** D is a 2×2 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. $T = \text{AdditivePhylogeny}(D)$
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 $\neq D_{lj}$
 9. output "matrix is not additive"
10. return
11. Extend all "hanging" edges by length δ
12. return T



The Four Point Condition

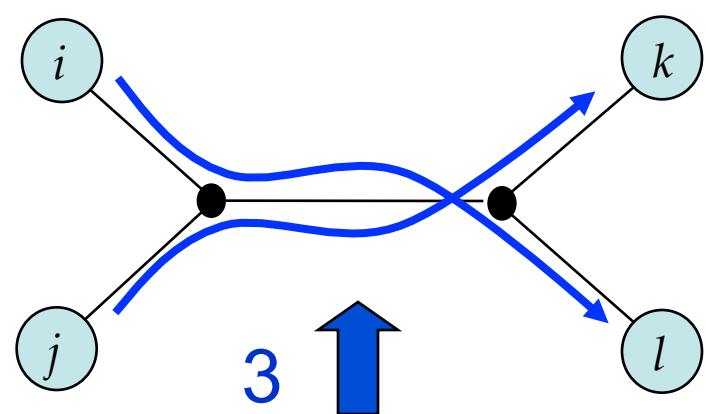
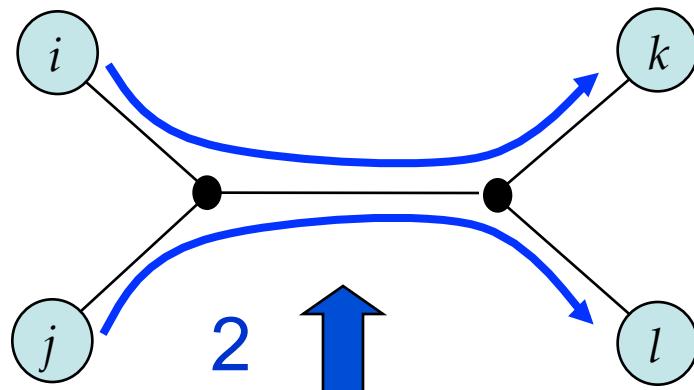


- AdditivePhylogeny Algorithm provides a way to check if distance matrix D is additive (i.e. it does not converge to a single 2 by 2 matrix)
- **An even more efficient additivity check is the “four-point condition”**
- Let $1 \leq i, j, k, l \leq n$ be four distinct leaves in a tree

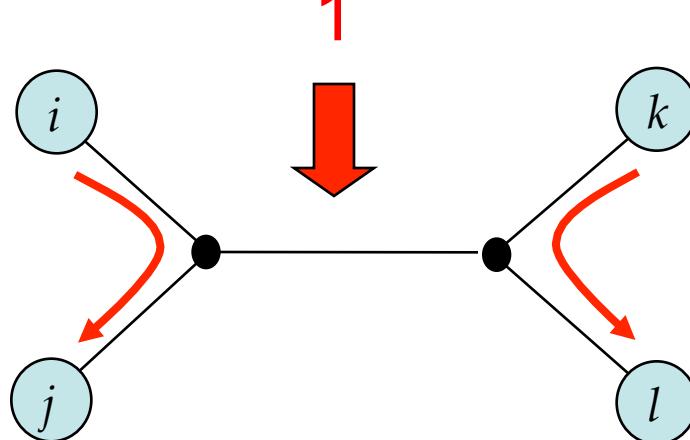


The Four Point Condition (cont'd)

Compute: 1. $D_{ij} + D_{kl}$, 2. $D_{ik} + D_{jl}$, 3. $D_{il} + D_{jk}$



2 and 3 represent the same number: the length of all edges + the middle edge (it is counted twice)



1 represents a smaller number: 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 $n \times n$ matrix D is additive if and only if the four point condition holds for *every* quartet $1 \leq i,j,k,l \leq n$



Next Time



- How to create trees if the matrices are not additive

