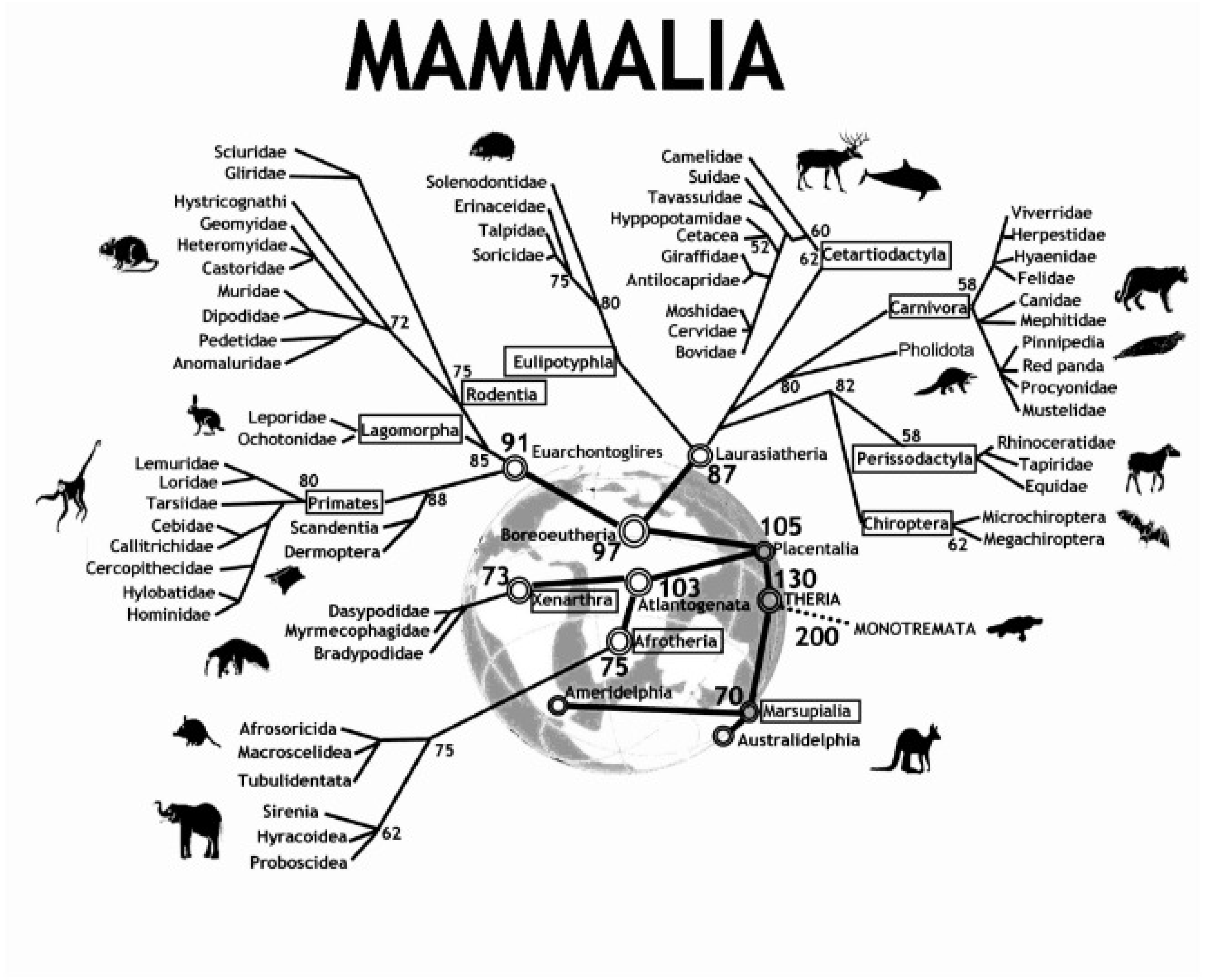
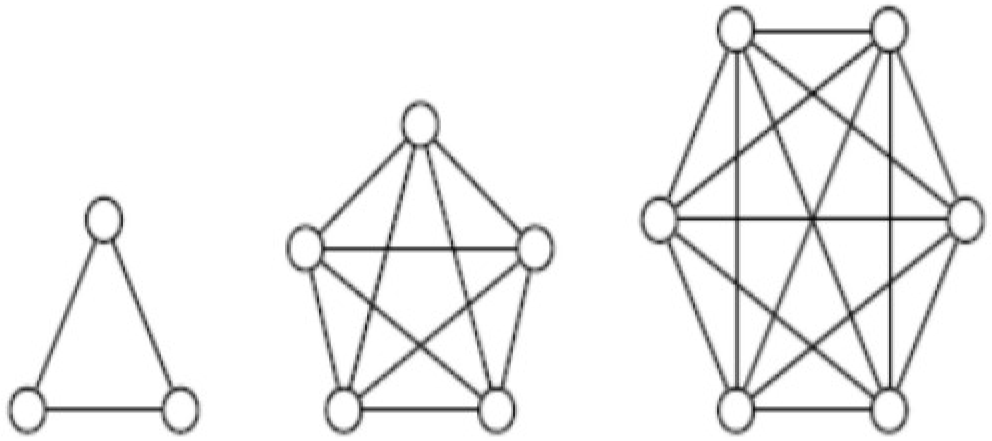
Clustering and Evolution



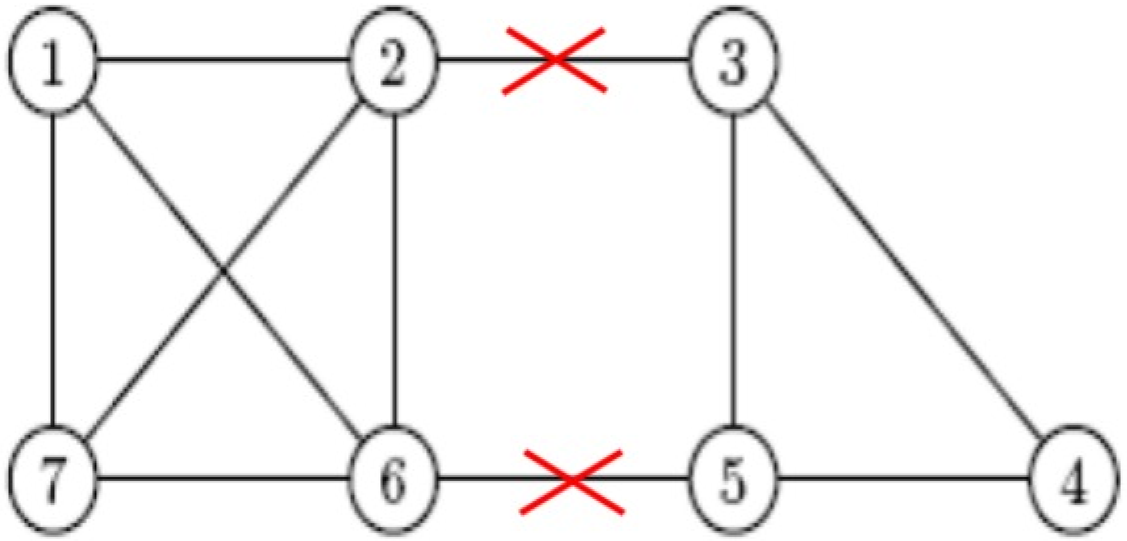
Clique Graphs

- 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.



Graph Transformations

- How to transform a given graph into a clique graph
- 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

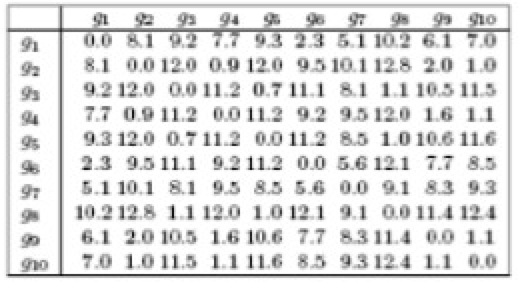
Distance Graphs

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 a Distance Graph into a Clique Graph

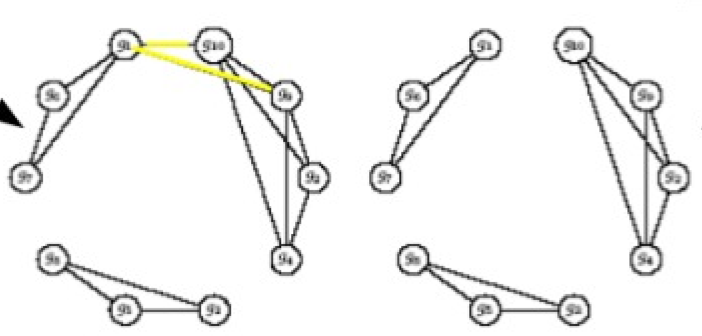
The distance graph (threshold θ=7) is transformed into a clique graph after removing the two highlighted edges



	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



(b) Distance graph for $\theta = 7$. (c) Clique graph.

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)< θ and distant otherwise

CAST Algorithm

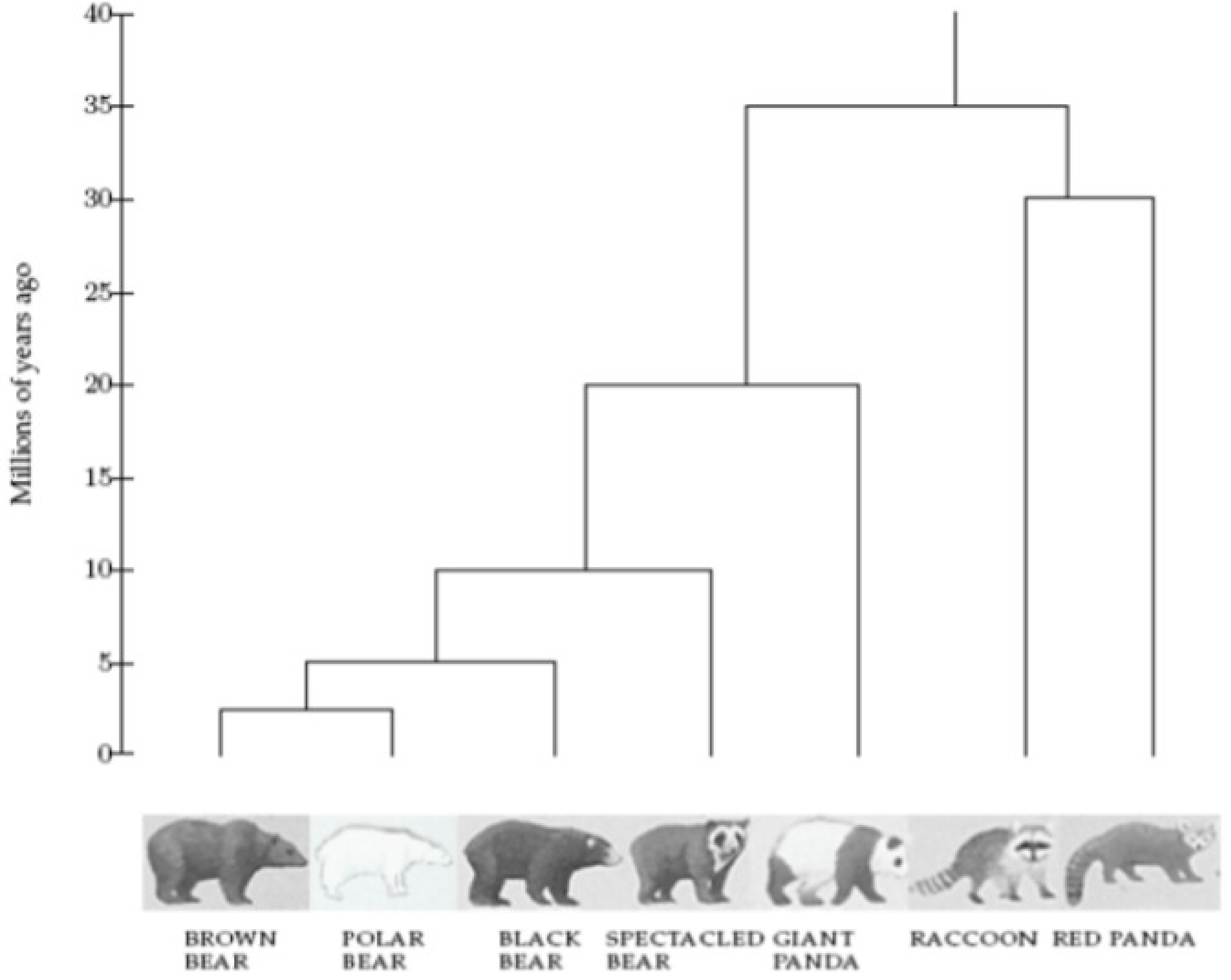
```
1. CAST(S, G, $\theta$)
2. $\mathbf{P} \leftarrow \emptyset$
3. while $\mathbf{S} \neq \emptyset$
4.    $v \leftarrow$ vertex of maximal degree in the distance graph $\mathbf{G}$
5.    $\mathbf{C} \leftarrow \{v\}$
6.    while *a close gene i not in **C*** or *distant gene i in* **C** exists
7.       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 of the Giant Panda

- 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



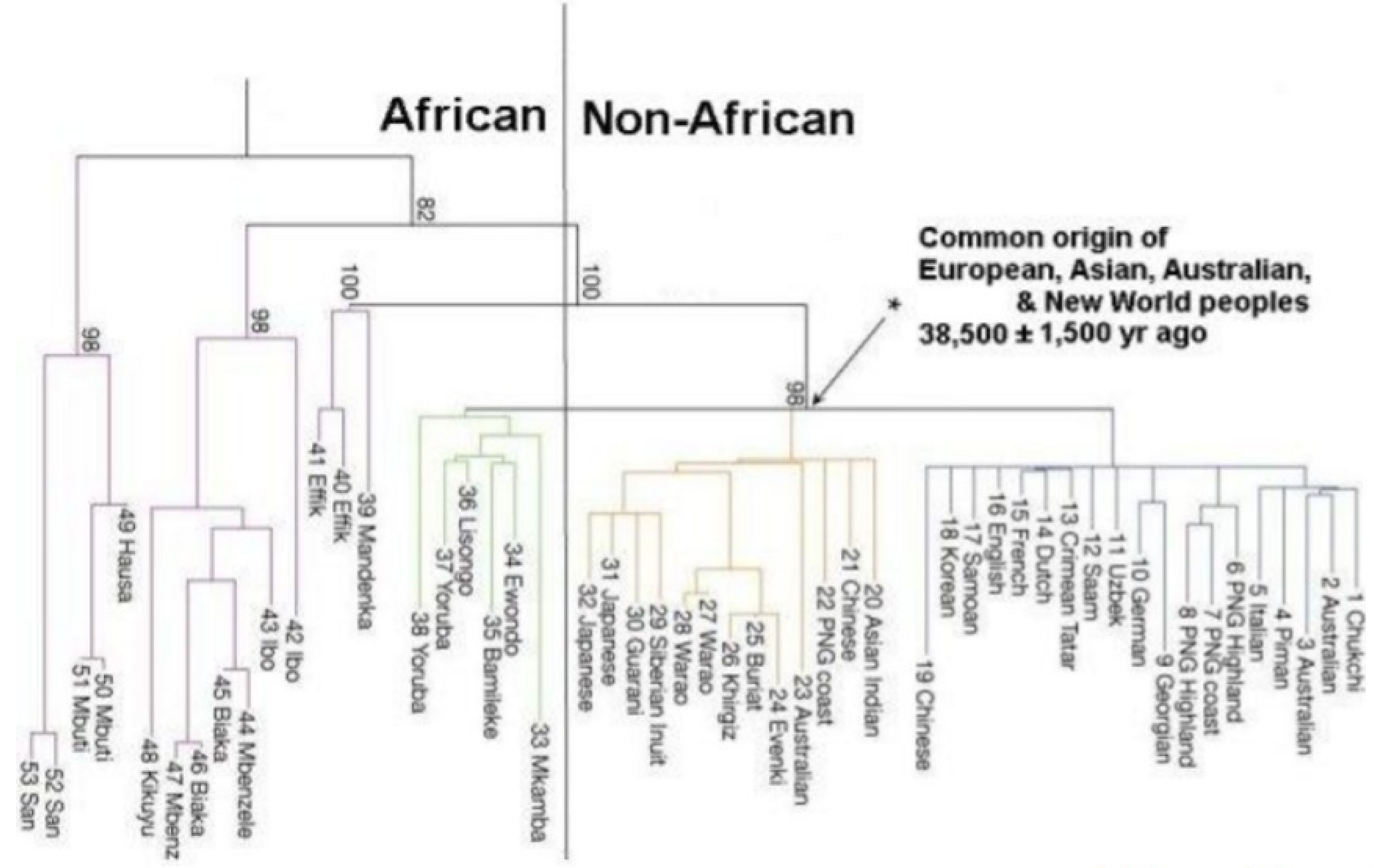
Evolutionary Trees: DNA-based Approach

- 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



First Previous Next Notebook

"Out of Africa" vs Multiregional Hypothesis

Out of Africa:

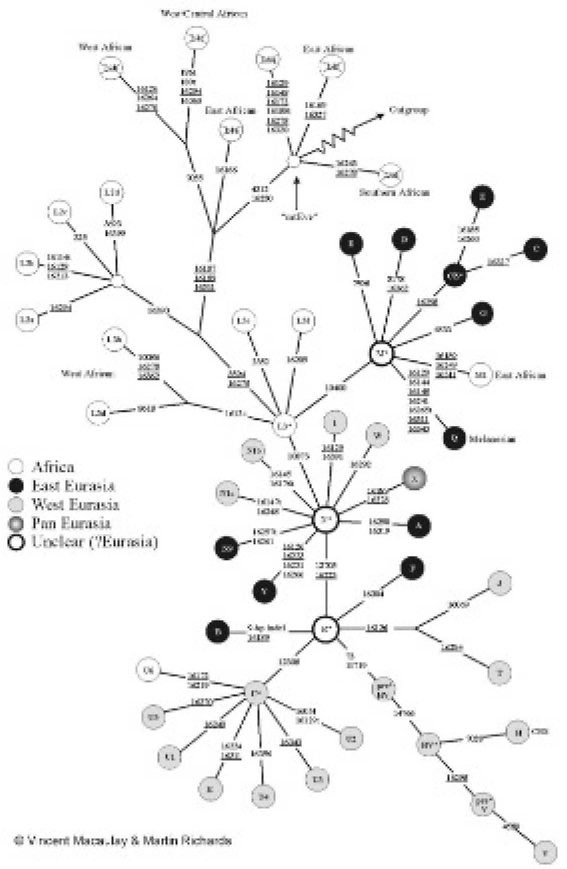
- Humans evolved in Africa ~150,000 years ago
- Humans migrated out of Africa, replacing other humanoids around the globe
- There is no direct descendence from Neanderthals

Multiregional:

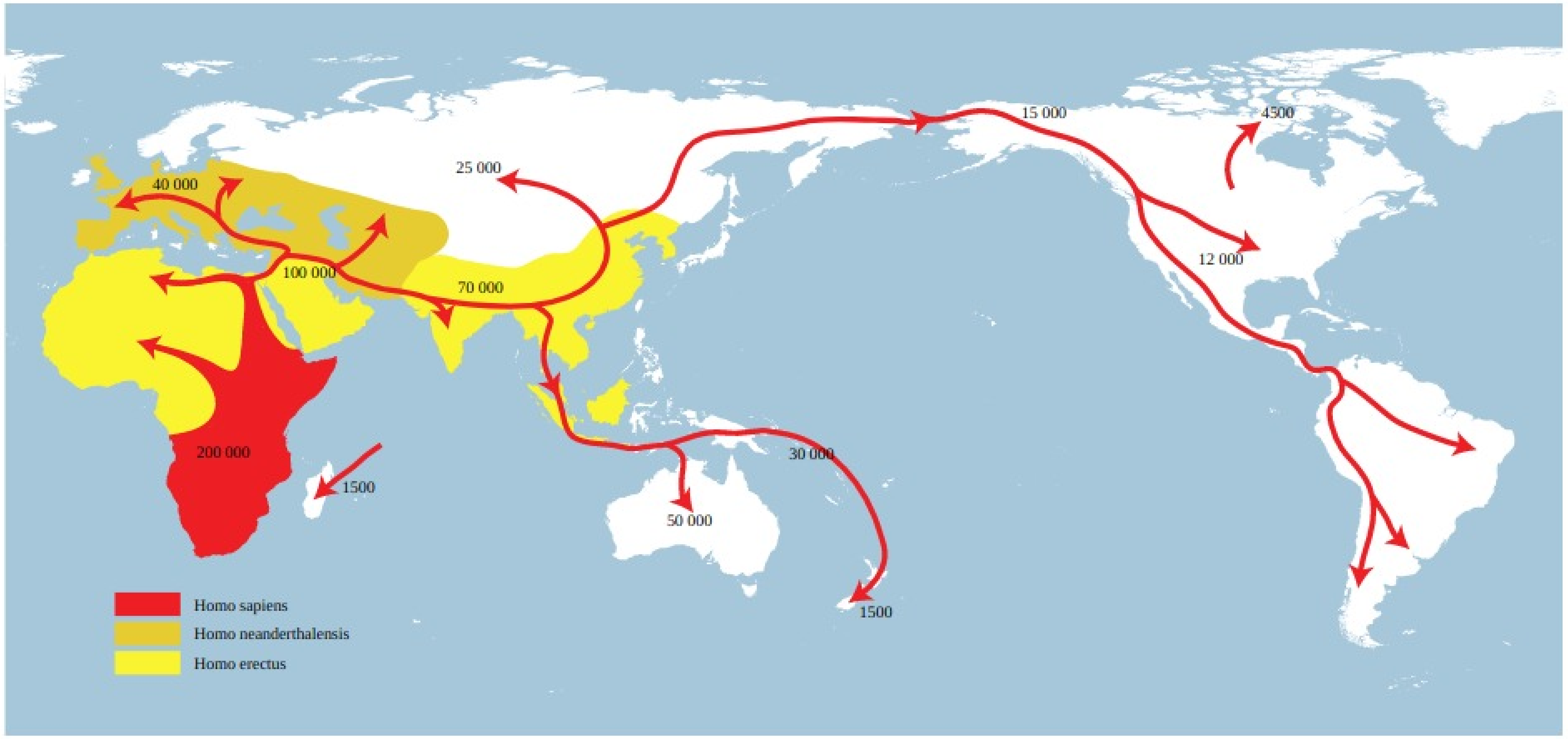
- Humans evolved in the last two million years as a single species. Independent appearance of modern traits in different areas
- Humans migrated out of Africa mixing with other humanoids on the way
- There is a genetic continuity from Neanderthals to humans

mtDNA Analysis

- Supports the "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.



Humanoid Migrations Out of Africa

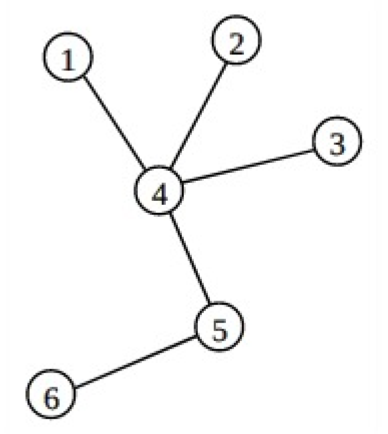


Evolutionary Trees

- How do you construct trees from DNA sequences?
  - leaves represent existing species
  - internal vertices represent ancestors
  - root represents the oldest evolutionary ancestor

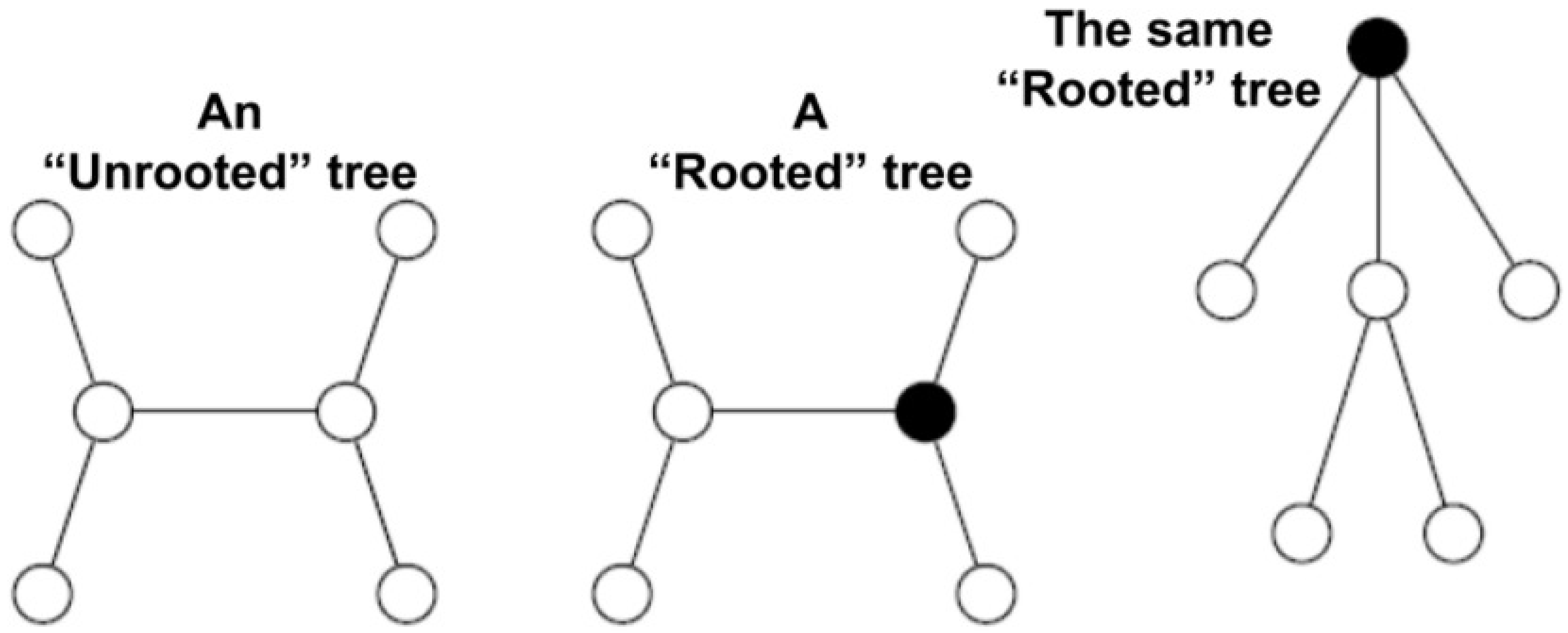
Trees

- Trees are a special case of a graph
- A connected tree with N-nodes has exactly N-1 edges
- There exists exactly one path from any node *i* to any other node *j* in a tree
- A tree contains no cycles
- The leafs of a tree have degree 1
- Interior nodes have degree > 1



Rooted and Unrooted Trees

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

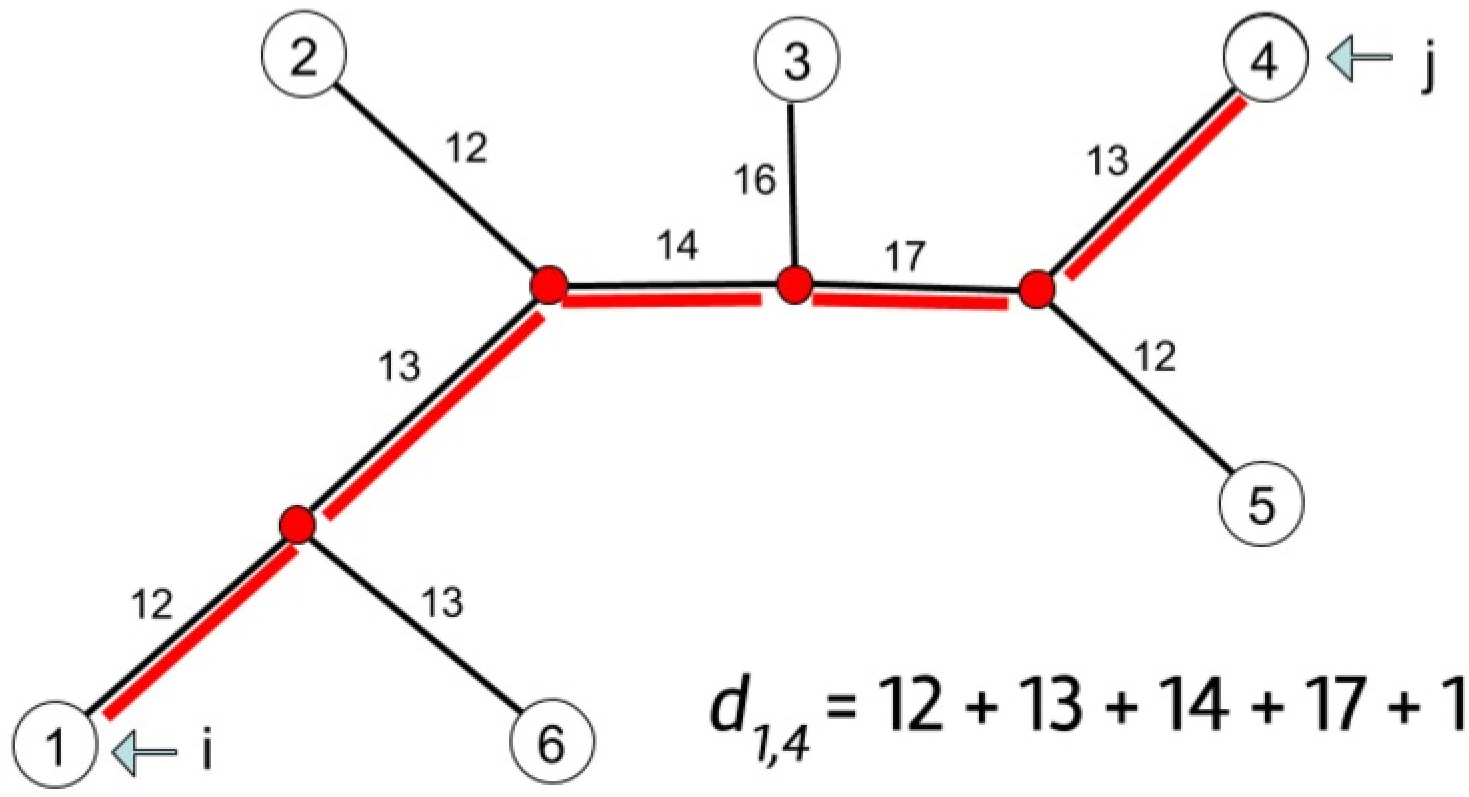


Distance in Trees

- 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

Example Tree Distance



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

Distance Matrix

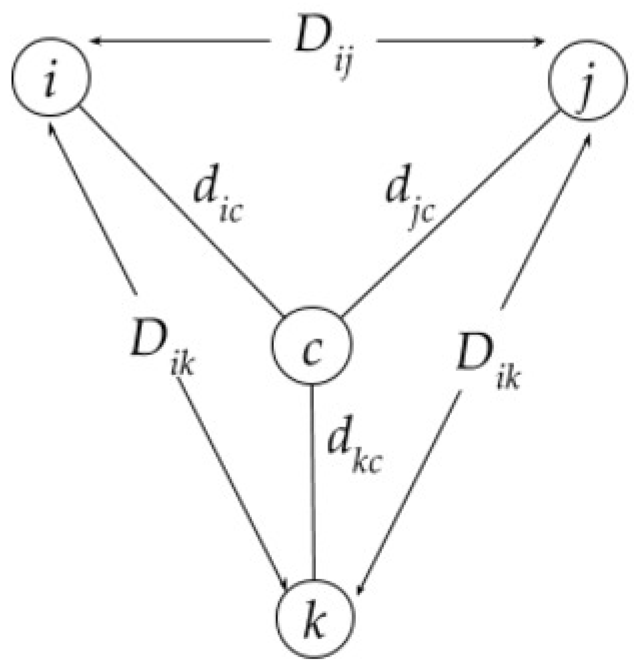
- Given *n* species, we can compute the *n x n* distance matrix **D**
- D_{ij} represents the distance between species *i* and species *j*
- There are many *measures* of distance
    - D_{ij} might be the edit distance between a gene in species i and species j
    - D_{ij} might be the number to reversals to match the gene order
    - D_{ij} might be *Tree distance*
- Gnereal Distance Matrix properties
    - $D_{ii} = 0$
    - $D_{ij} = D_{ji}$
    - $D_{ij} \leq D_{ik} + D_{kj}$

Evolutionary Trees and Distance Matrices

- The problem with evoltionary tree reconstruction is that we *observe only the leaf nodes*
- The ancestors (interior nodes) are inacessable to us
- *The problem:* Given only pairwise distances from leaf nodes of a tree, how do we infer distances to hidden ancestors.

A simple case

- Tree reconstruction of a common ancestor from 3 leaf nodes
- We have 3 leaves i, j, k and want a tree with a common *center* vertex c
- So first coumpute all pairwise distances, $d_{i,j}$, $d_{i,k}$, and $d_{j,k}$.
- Then use them to infer $d_{i,c}$, $d_{j,c}$, and $d_{k,c}$



Observe:

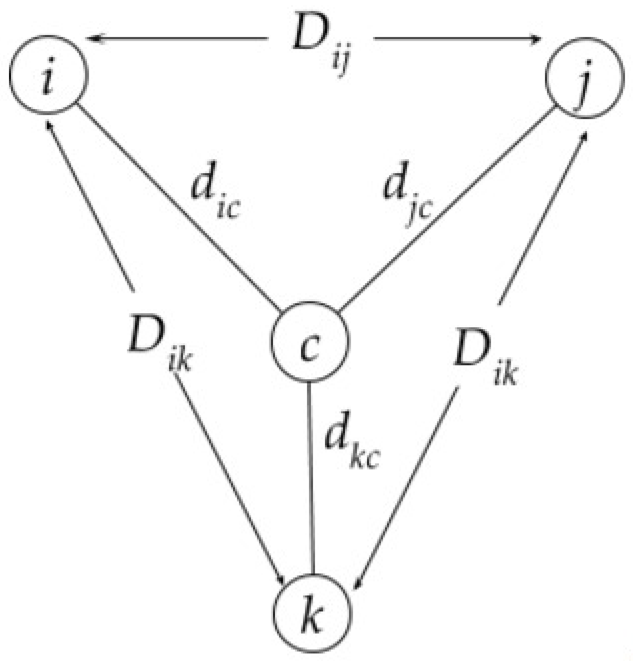
$$d_{ic} + d_{jc} = D_{ij}$$

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

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

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

Solution for 3-leave tree



$$\begin{aligned} &(d_{ic} + d_{jc} = D_{ij}) \\ +\,&(d_{ic} + d_{kc} = D_{ik}) \\ 2d_{ic} &+ d_{jc} + d_{kc} = D_{ij} + D_{ik} \end{aligned}$$

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

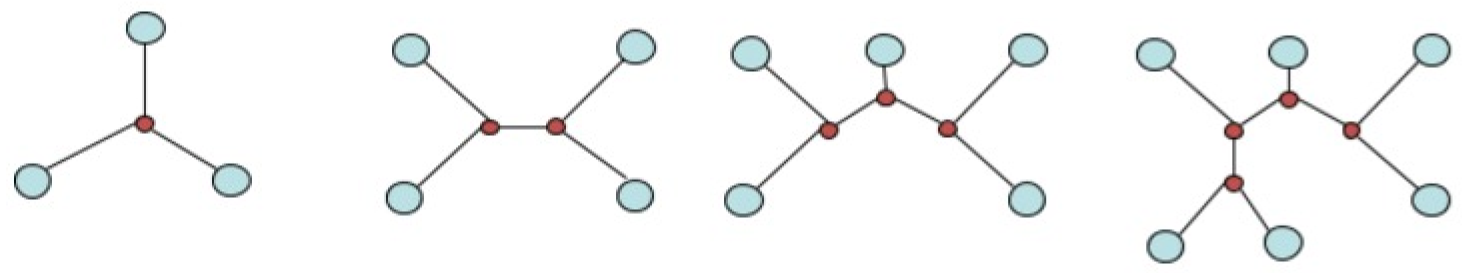
$$d_{ic} = \frac{D_{ij} + D_{ik} - D_{jk}}{2}$$

Similarly:

$$d_{jc} = \frac{D_{ij} + D_{jk} - D_{ik}}{2},\ d_{kc} = \frac{D_{ik} + D_{jk} - D_{ij}}{2},$$

Trees with more than 3 Leaves

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



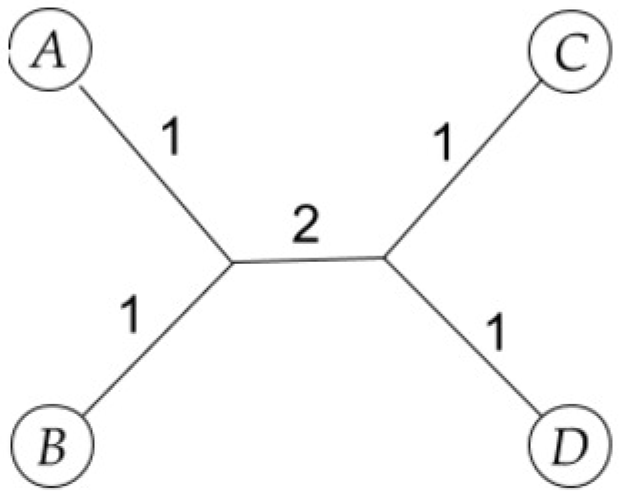
- This means fitting a given tree to a distance matrix **D** requires solving a system of “n choose 2” or ½ x(x-1) equations with 2n-3 variables
- This is not always possible to solve for n > 3 given arbitrary/noisy distances

* 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

- Given a tree, it is straightforward compute its distance matrix, **D**
- Definition: Matrix D is *Additive* if there exists a tree T with $d_{ij}(T) = D_{ij}$ for all i,j

δ	**A**	**B**	**C**	**D**
A	0	2	4	4
B	2	0	4	4
C	4	4	0	2
D	4	2	2	0



Given only a distance matrix

- If given only a distance matrix, does there exist a tree?
- If not, Matrix D is *Non-Additive*
- But, what is the closest tree?

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

How, from a distance matrix, does one determine if a tree exists?

Distance Based Phylogeny Problem

- **Goal:** Reconstruct an evolutionary tree from a distance matrix
- **Input:** n x n distance matrix **D**
- **Output:** A tree, T, with edge weights and n leaves fitting **D**

If we know that **D** is additive, this problem has a solution and there is a simple algorithm to solve it

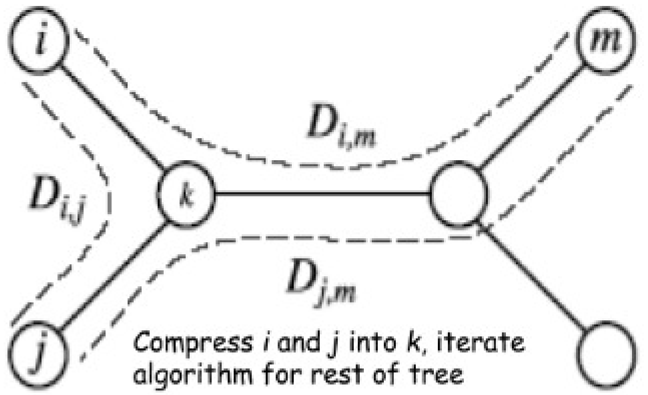
Key Idea: Merge neighbors

- 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:

$$d_{im} = \frac{D_{jm} + D_{km} - D_{jk}}{2}$$

$$d_{jm} = \frac{D_{im} + D_{km} - D_{ik}}{2}$$

$$d_{km} = \frac{D_{im} + D_{jm} - D_{ij}}{2}$$



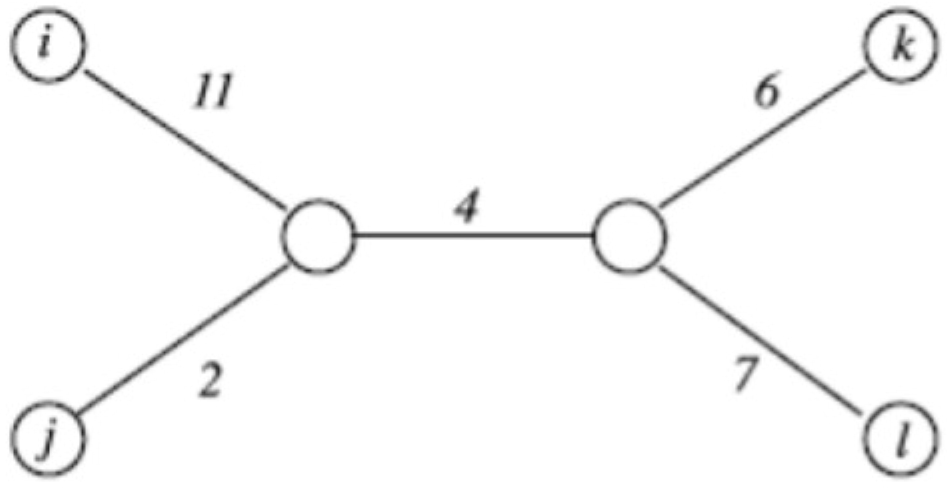
i
m
$D_{i,m}$
$D_{i,j}$
k
$D_{j,m}$
j
Compress i and j into k, iterate algorithm for rest of tree

How to find Neighboring Leaves?

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

Greedy might be wrong

- 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 nontrivial! (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).

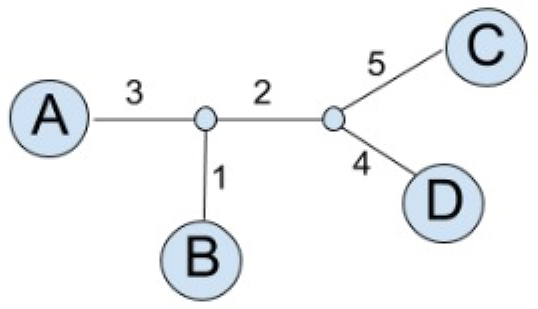
Looking for Degenerate Triples

- 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 eventually "create" a degenerative triple in *D* by shortening all hanging or leaf edges in the tree.

Shortening Hanging Edges

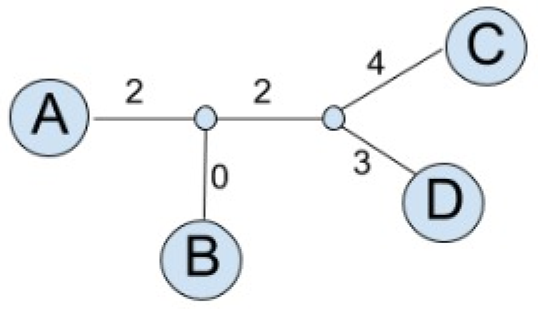
- **Approach:** Shorten all "hanging" edges (edges that connect leaves) until a degenerate triple is found

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



- Shorten all leaf edges by 1 (reduces distances by 2, Why?)
- $d_{AC} = d_{AB} + d_{BC}$ and $d_{AD} = d_{AB} + d_{BD}$ (i.e. degenerate triples)

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



Next Time

- We'll take these insights and derive an algorthim for constructing a tree from a distance matrix
- How do we determine if a given distance matrix is additive?
- If it is not additive, can we construct an *approximate* tree?