

Molecular Evolution

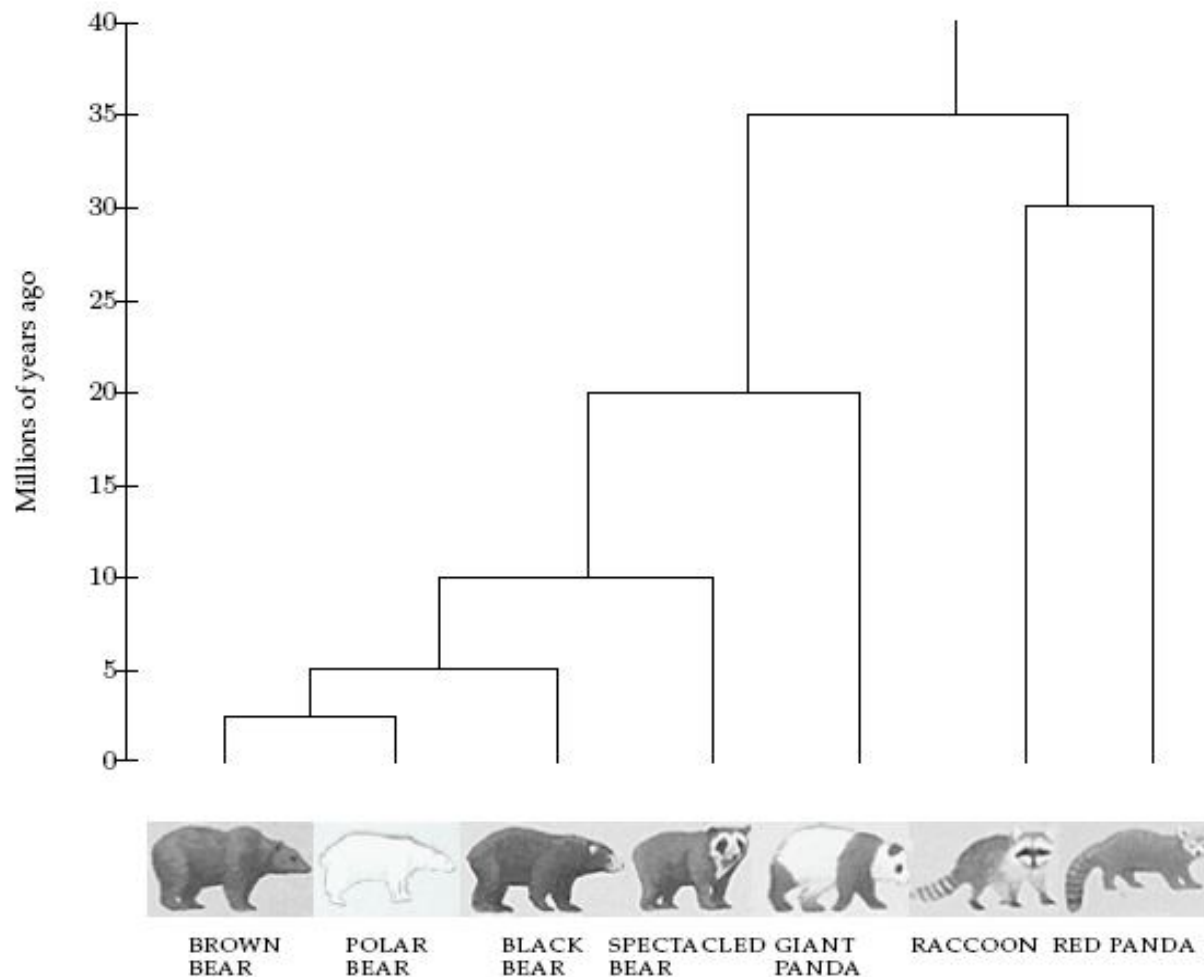
Early Evolutionary Studies

- Anatomical features were the dominant criteria used to derive evolutionary relationships between species since Darwin till early 1960s
- The evolutionary relationships derived from these relatively subjective observations were often inconclusive. Some of them were later proved incorrect

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.

Evolutionary Trees

How are these trees built from DNA sequences?

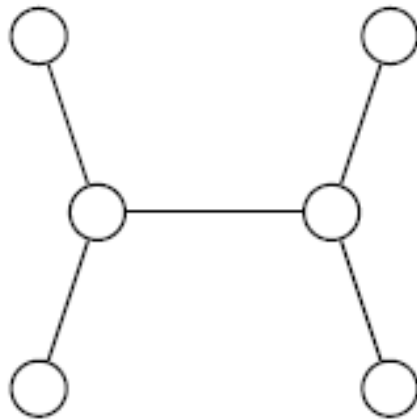
Evolutionary Trees

How are these trees built from DNA sequences?

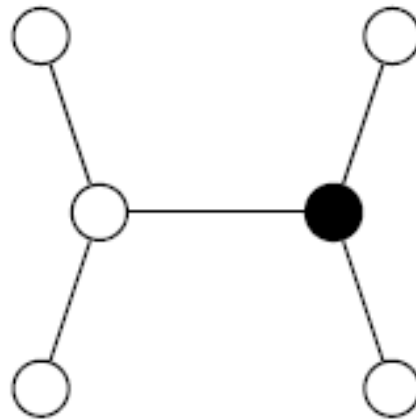
- leaves represent existing species
- internal vertices represent ancestors
- root represents the oldest evolutionary ancestor

Rooted and Unrooted Trees

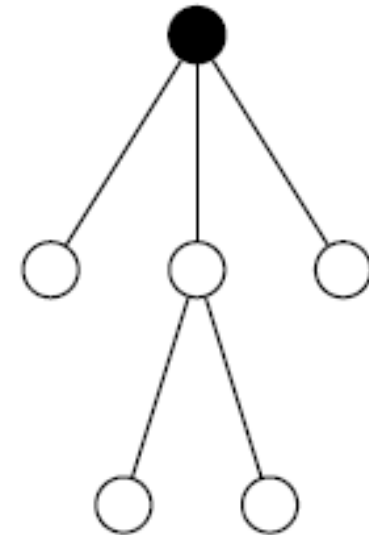
In the unrooted tree the position of the root (“oldest ancestor”) is unknown. Otherwise, they are like rooted trees



(a) Unrooted tree



(b) Rooted tree



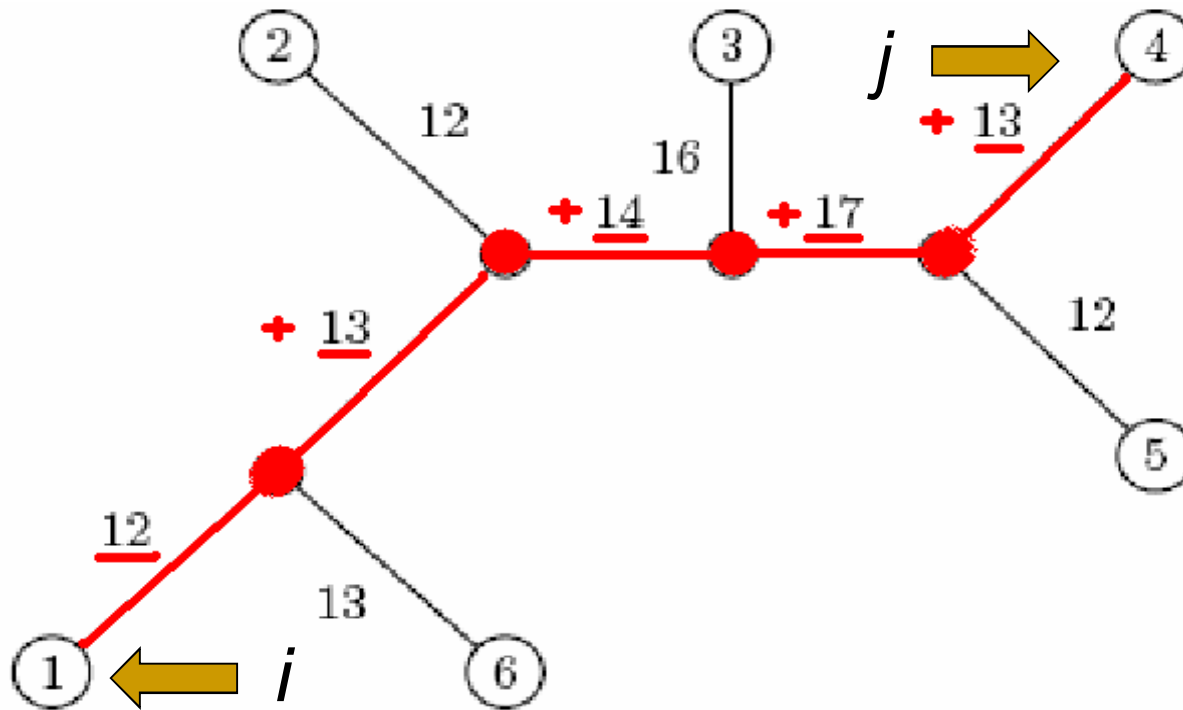
(c) The same rooted tree

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 $d_{ij}(T)$ - the length of a path between leaves i and j

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

Distance in Trees: an Example



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

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

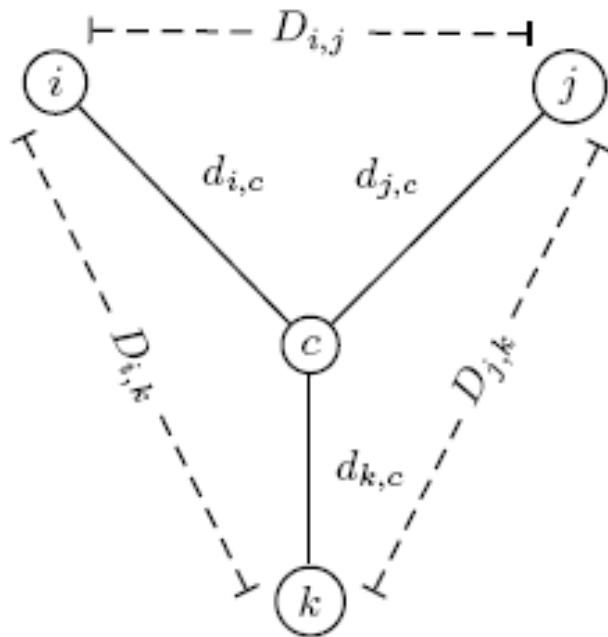
- Given n species, we can compute the $n \times n$ **distance matrix** D_{ij}
- Evolution of these genes is described by a tree that ***we don't know***.
- We need an algorithm to construct a tree that best ***fits*** the distance matrix D_{ij}

Fitting Distance Matrix

- Fitting means $\underbrace{D_{ij}}_{\text{Edit distance between species (*known*)}} = \underbrace{d_{ij}(T)}_{\text{Lengths of path in an (*unknown*) tree } T}$

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



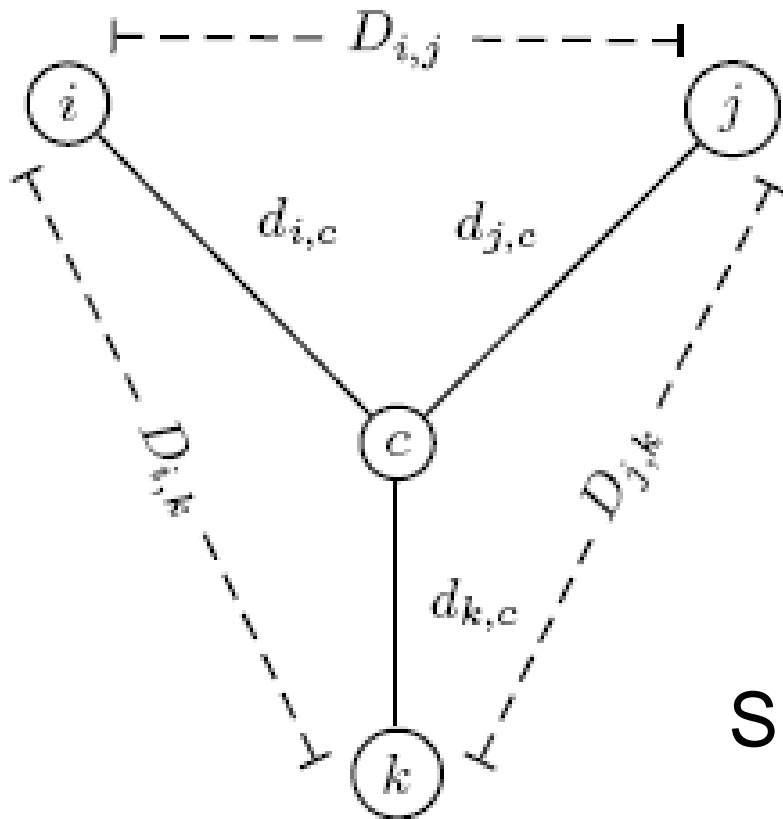
Observe:

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

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

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

Reconstructing a 3 Leaved Tree (cont'd)



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

$$2d_{ic} + \underbrace{d_{jc} + d_{kc}}_{D_{jk}} = D_{ij} + D_{ik}$$

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

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

Similarly,


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

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

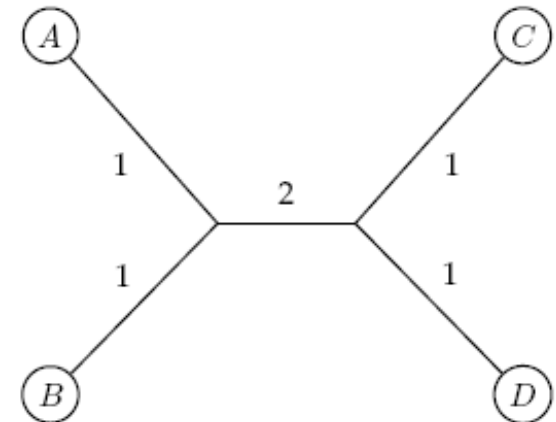
Trees with > 3 Leaves

- An unrooted binary 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” equations with $2n-3$ variables
- This is not always possible to solve for $n > 3$

Additive Distance Matrices

Matrix D is  ADDITIVE if there exists a tree T with $d_{ij}(T) = D_{ij}$

	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

?

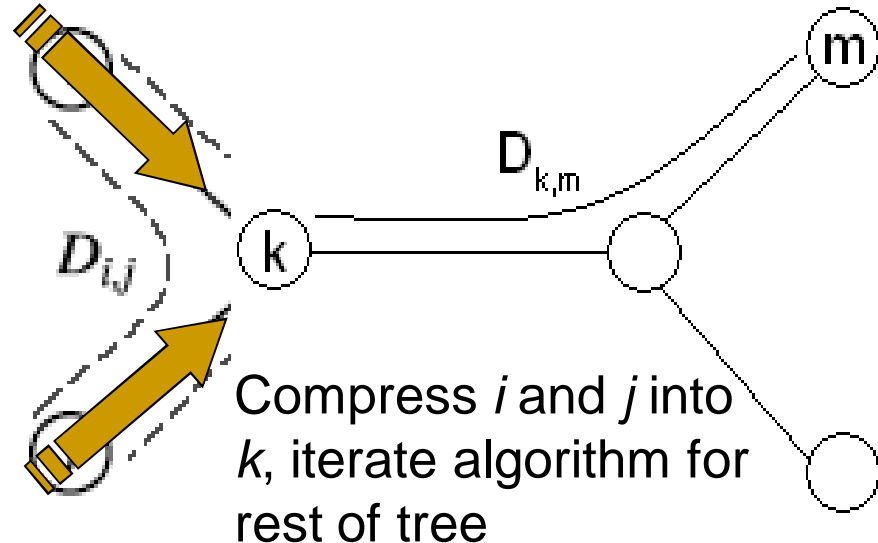
Distance Based Phylogeny Problem

- 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

Using Neighboring Leaves to Construct the Tree

- Find **neighboring leaves** i and j with 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

- To find neighboring leaves we simply select a pair of closest leaves.

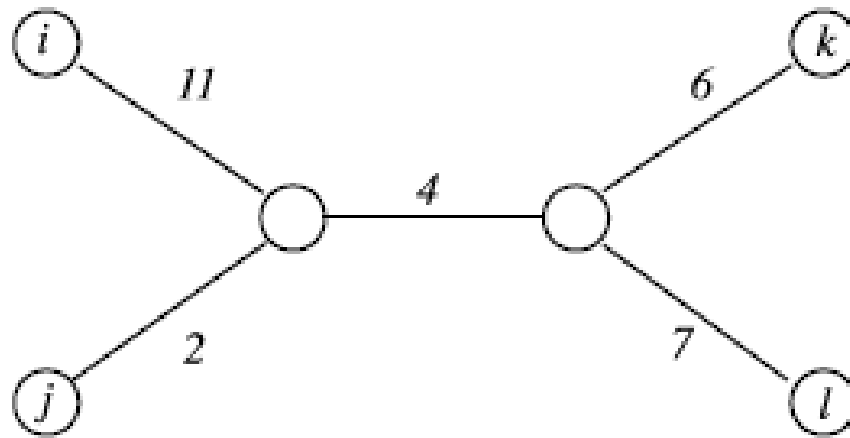
Finding Neighboring Leaves

- To find neighboring leaves we simply select a pair of closest leaves.

WRONG

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!

Neighbor Joining Algorithm

- In 1987 Naruya Saitou and Masatoshi Nei developed a neighbor joining algorithm for phylogenetic tree reconstruction
- **Find 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.

Degenerate Triples

- A degenerate triple is a set of three distinct elements $1 \leq i, j, k \leq n$, where $D_{ij} + D_{jk} = D_{ik}$
- 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 edges (in the tree).

Shortening Hanging Edges to Produce Degenerate Triples

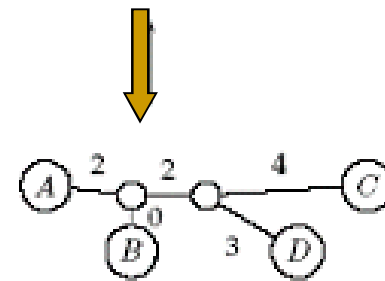
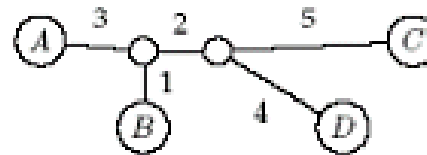
- Shorten all “hanging” edges (edges that connect leaves) until a degenerate triple is found

	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

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



Finding Degenerate Triples

- If there is no degenerate triple, all hanging edges are reduced by the same amount δ , so that all pairwise 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

$\delta = 1$

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

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

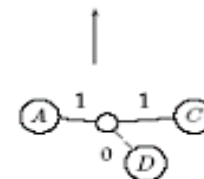
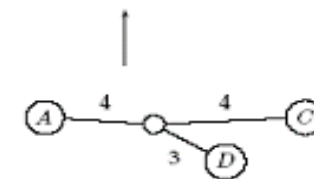
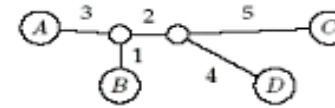
	A	C	D
A	0	8	7
C	8	0	7
D	7	7	0

$\delta = 3$

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

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

	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)

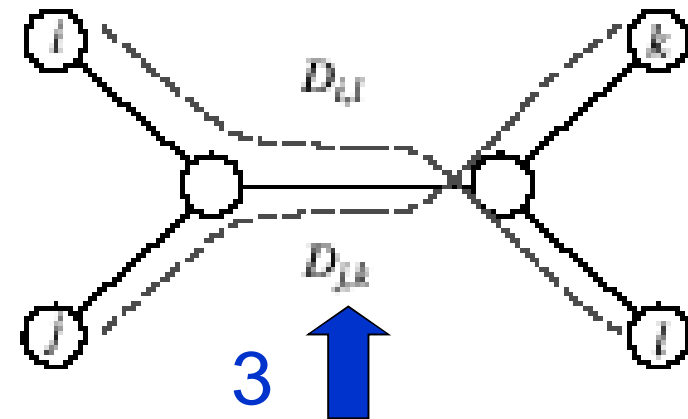
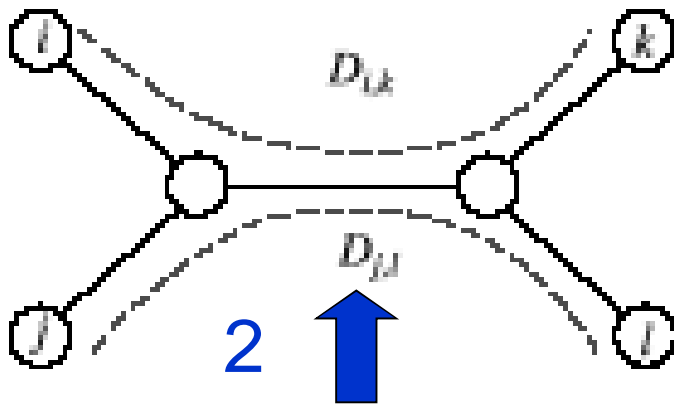
1. Find a triple i, j, k in D such that $D_{ij} + D_{jk} = D_{ik}$
2. $x = D_{ij}$
3. Remove j^{th} row and j^{th} column from D
4. $T = \text{AdditivePhylogeny}(D)$
5. Add a new vertex v to T at distance x from i to k
6. Add j back to T by creating an edge (v, j) of length 0
7. **for** every leaf l 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

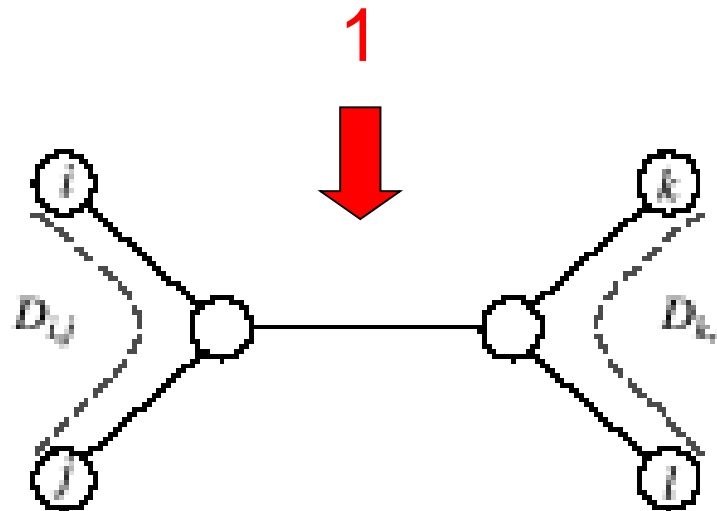
- AdditivePhylogeny provides a way to check if distance matrix D is additive
- **An even more efficient additivity check is the “four-point condition”**
- Let $1 \leq i, j, k, l \leq n$ be four distinct leaves in a tree

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