Molecular Evolution

Outline

1. Evolutionary Tree Reconstruction
2. Two Hypotheses for Human Evolution
3. Did we evolve from Neanderthals?
4. Distance-Based Phylogeny
5. Neighbor Joining Algorithm
6. Additive Phylogeny
7. Least Squares Distance Phylogeny
8. UPGMA
9. Character-Based Phylogeny
10. Small Parsimony Problem
11. Fitch and Sankoff Algorithms

Outline

12. Large Parsimony Problem
13. Nearest Neighbor Interchange
14. Evolution of Human Repeats
15. Minimum Spanning Trees

Section 1: Evolutionary Tree Reconstruction

Early Evolutionary Studies

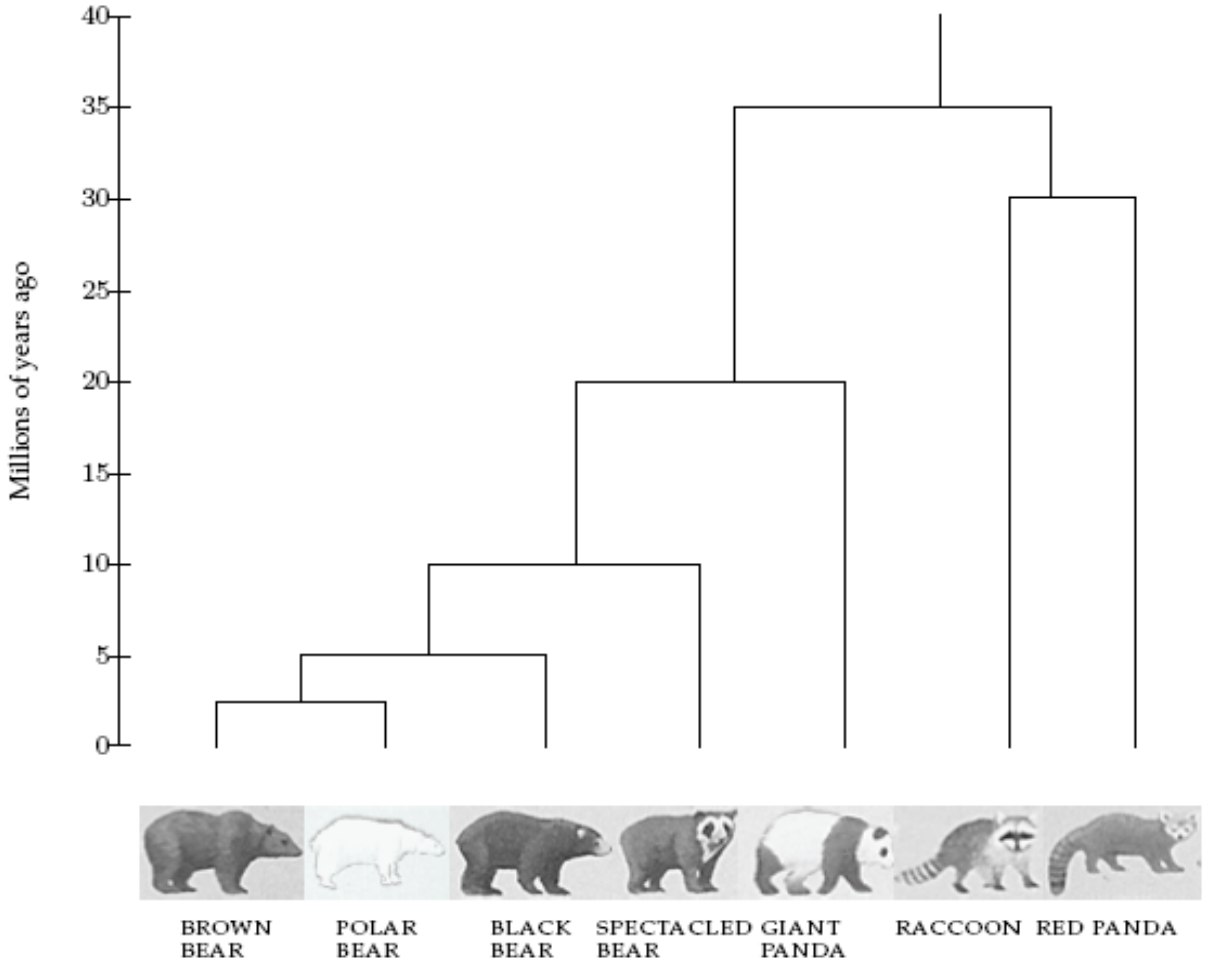
- Anatomical features were the dominant criteria used to derive evolutionary relationships between species since Darwin till early 1960s.

- The evolutionary relationships derived from these relatively subjective observations were often inconclusive. Some of them were later proven incorrect.

DNA Analysis: The Giant Panda Riddle

- For roughly 100 years, scientists were unable to figure out to which family the giant panda should belong.
- 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.

Emile Zuckerkandl

Linus Pauling

Gaylord Simpson vs. Emile Zuckerkandl

- "*From the point of view of hemoglobin structure, it appears that gorilla is just an abnormal human, or man an abnormal gorilla, and the two species form actually one continuous population.*"—Emile Zuckerkandl, Classification and Human Evolution, 1963

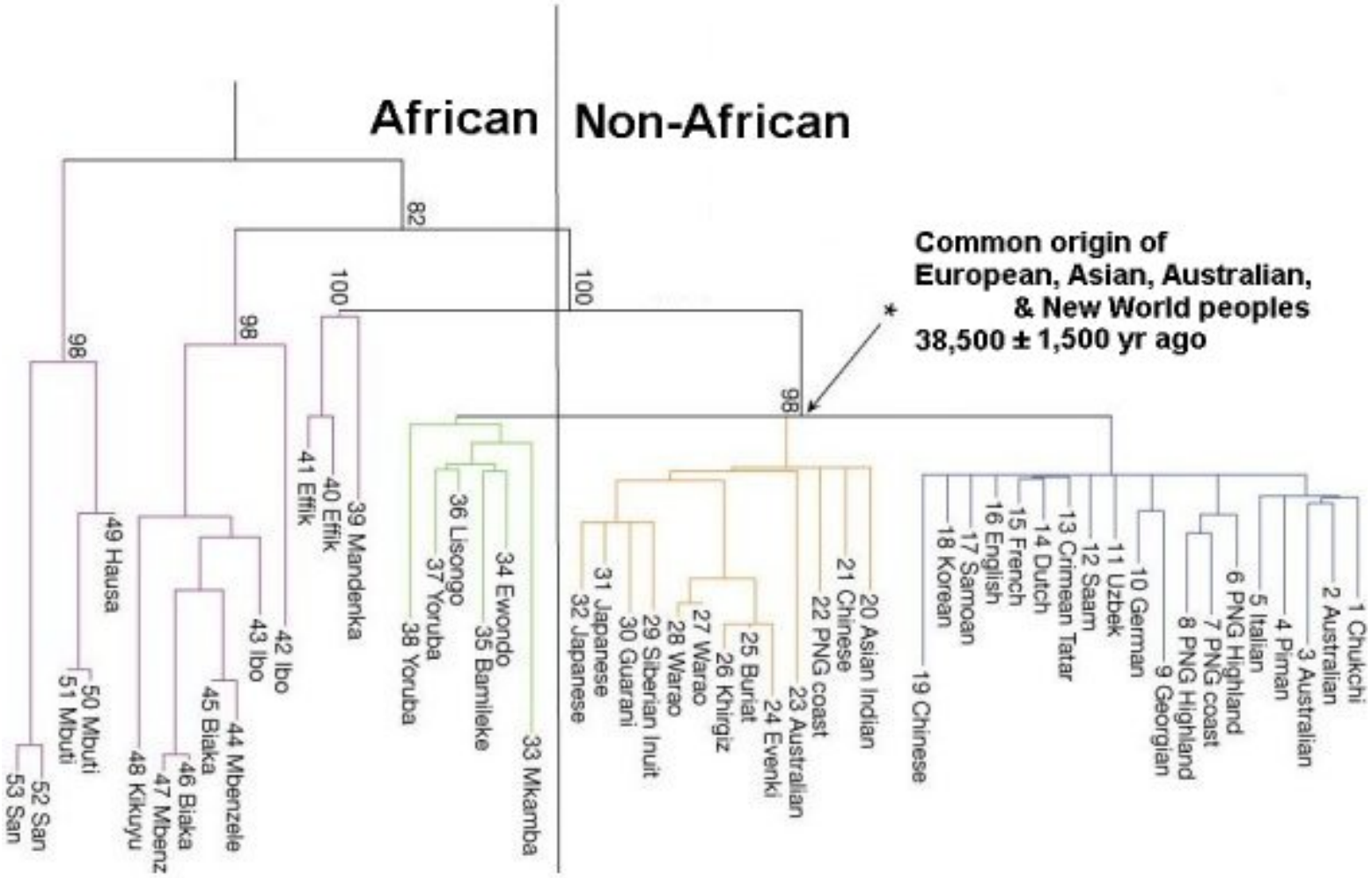
Emile Zuckerkandl

- "*From any point of view other than that properly specified, that is of course nonsense. What the comparison really indicates is that hemoglobin is a bad choice and has nothing to tell us about attributes, or indeed tells us a lie.*"—Gaylord Simpson, *Science*, 1964

Gaylord Simpson

Section 2: Two Hypotheses for Human Evolution

Human Evolutionary Tree



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

”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

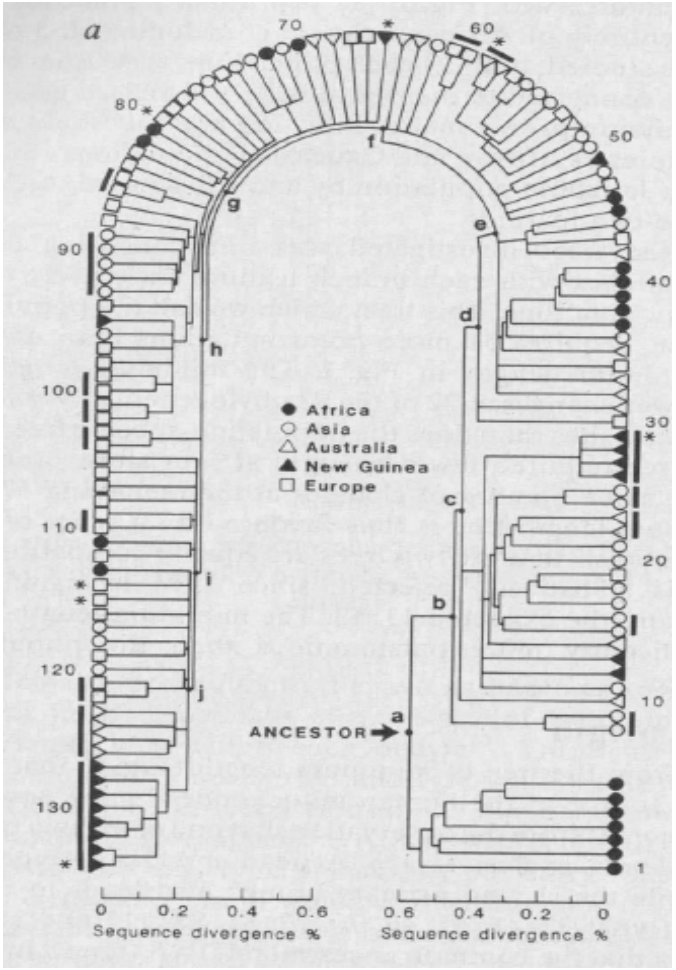
- Humans evolved in the last two million years as a single species. Independent appearance of modern traits in different areas.
- Humans migrated out of Africa mixing with other humanoids along the way.
- There is genetic continuity from Neanderthals to humans.

mtDNA Analysis Supports "Out of Africa"

- 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

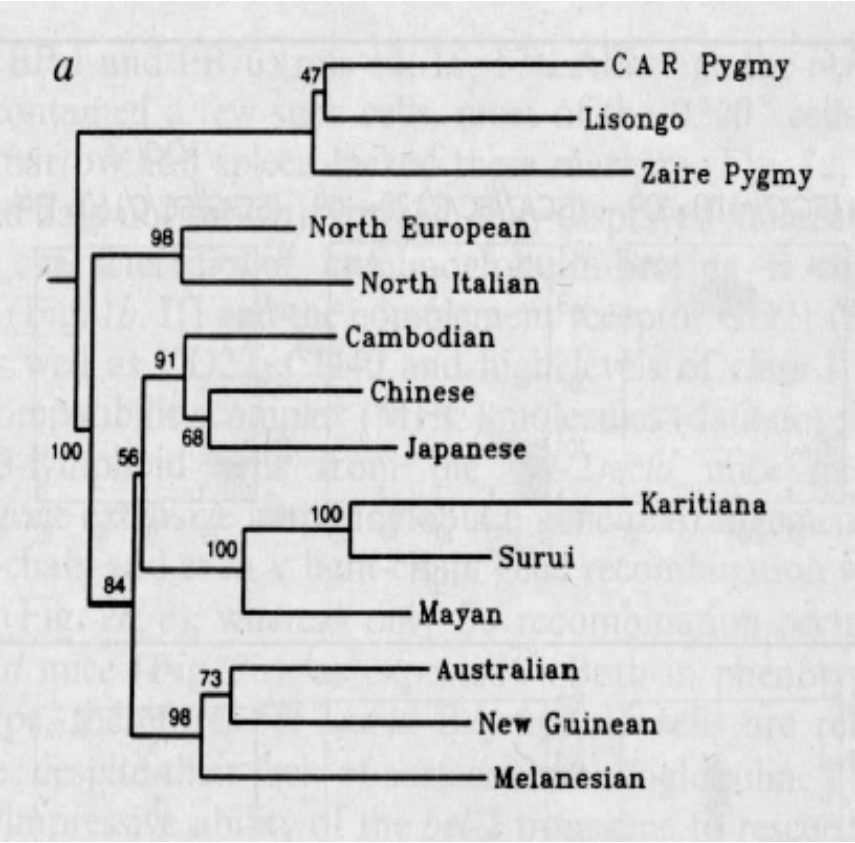
- The evolutionary tree separates one group of Africans from a group containing all five populations.



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: 5 Tribes

1. Yorubans
2. Western Pygmies
3. Eastern Pygmies
4. Hadza
5. Ikung

http://www.becominghuman.org

Section 3: Did We Evolve From Neanderthals?

Two Neanderthal Discoveries Across Europe

- The distance from Feldhofer to Mezmaiskaya is 2,500 km.
- Is there a connection between Neanderthals and modern Europeans?

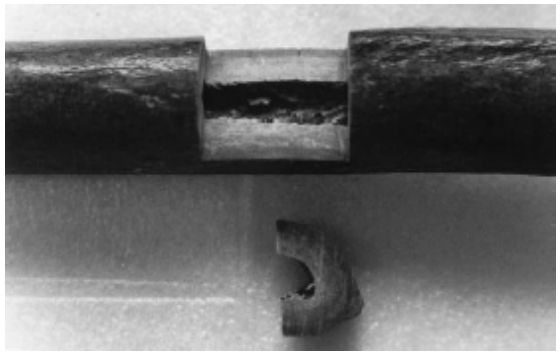
Multiregional Hypothesis?

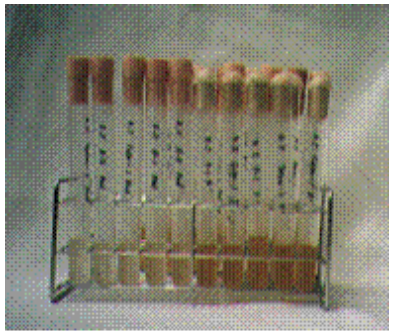
- May predict some genetic continuity from the Neanderthals to today's Europeans.

- Can explain the occurrence of varying regional characteristics.

Sequencing Neanderthal mtDNA

- mtDNA from the bone of Neanderthal is used because it is up to 1,000 times more abundant than nuclear DNA.
- DNA decays over time, and only a small amount of ancient DNA can be recovered (upper limit: 100,000 years).
  - **Quick Note**: This makes *Jurassic Park* impossible.
- PCR of mtDNA fragments are too short, implying that human DNA may have mixed in.





Silica Plug or Cotton Plug

Neanderthals vs Humans: Surprising Divergence

- Human vs. Neanderthal:
  - 22 substitutions and 6 indels in 357 bp region.

- Human vs. Human:
  - Only 8 substitutions.

Human

Neanderthal

Section 4: Distance-Based Phylogeny

Phylogenetic Analysis of HIV Virus

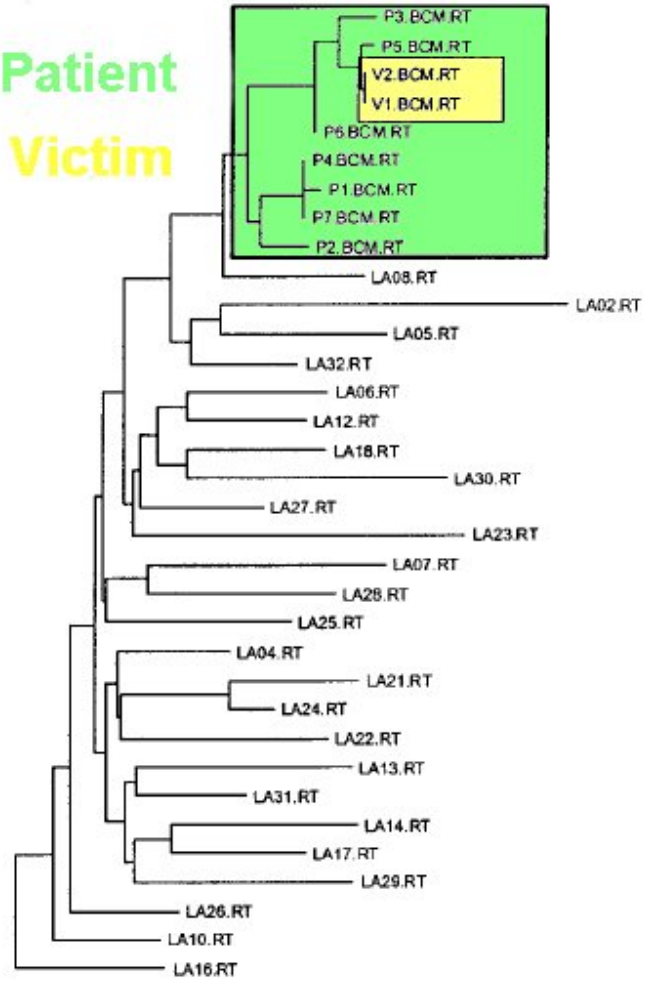
- Lafayette, Louisiana, 1994 – A woman claimed her ex-lover (who was a physician) injected her with HIV+ blood.
- Records show the physician had drawn blood from an HIV+ patient that day.
- But how can we prove that the blood from that specific HIV+ patient ended up in the woman?

Phylogenetic Analysis of HIV Virus

- HIV has a high mutation rate, which can be used to trace paths of transmission.
- Two people who got the virus from two different people will have very different HIV sequences.
- *Tree reconstruction* methods were used to track changes in HIV genes.

Phylogenetic Analysis of HIV Virus

- Took samples from the patient, the woman, and control HIV+ patients.
- In tree reconstruction, the woman's sequences were found to be evolved from the patient's sequences, indicating a close relationship between the two.
- Nesting of the victim's sequences within the patient's sequence indicated transmission from patient to victim.
- This was the first time phylogenetic analysis was used in a court case.



How Many Times Has Evolution Invented Wings?

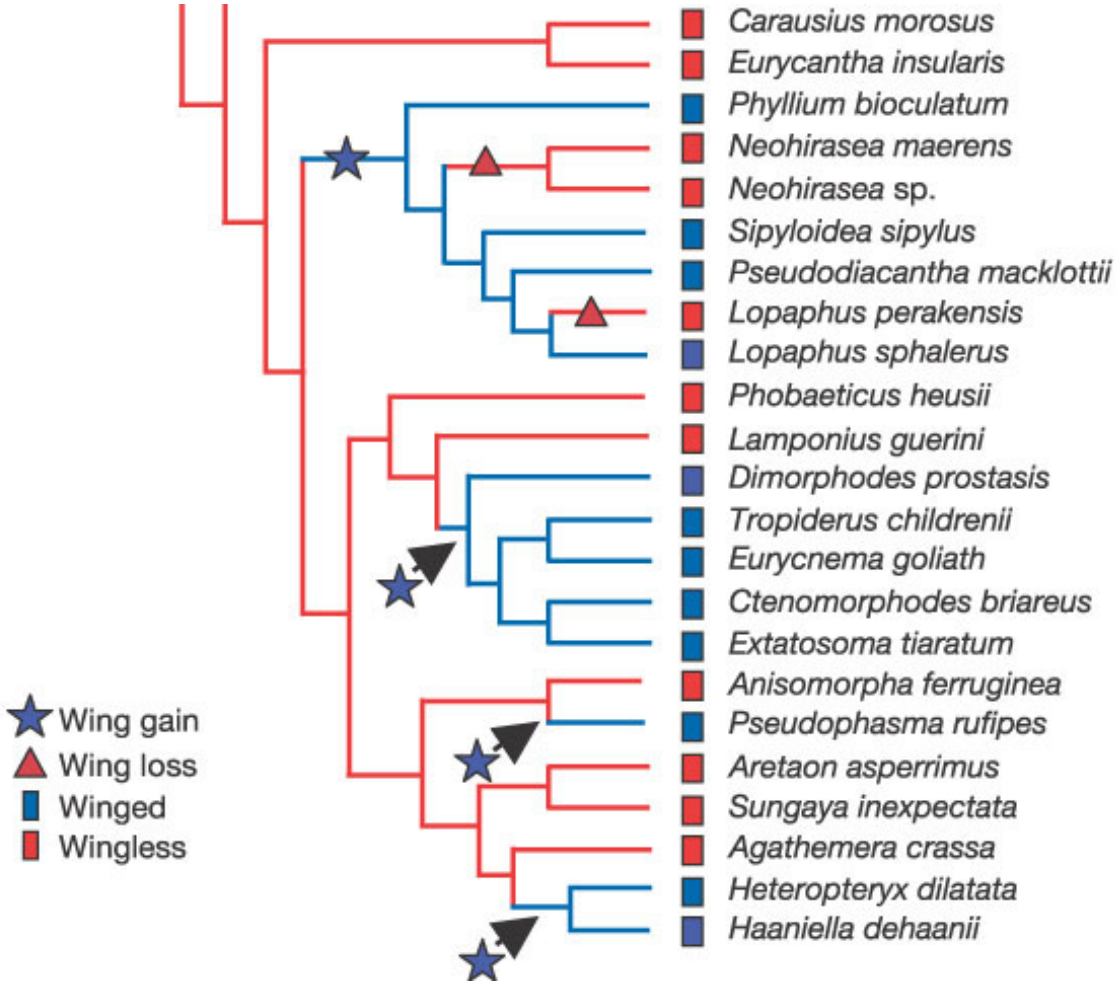
- Whiting et. al. (2003) looked at winged and wingless stick insects.

Reinventing Wings

- Previous studies had shown *winged* → *wingless* transitions.
- The *wingless* → *winged* transition is much more complicated (need to develop many new biochemical pathways).
- Biologists used multiple tree reconstruction techniques, all of which required re-evolution of wings.

Most Parsimonious Tree: Winged vs. Wingless

- The evolutionary tree is based on *both* DNA sequences and presence/absence of wings.
- Most parsimonious reconstruction gives a *wingless* ancestor for stick insects.
  - Therefore their wings evolved.



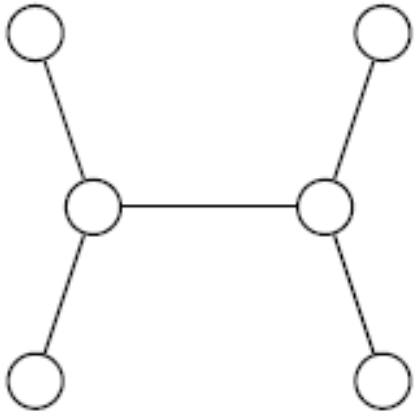
Evolutionary Trees

- **Evolutionary Tree**: A tree constructed from DNA sequences of several organisms.
  - Leaves represent existing species.
  - Internal vertices represent ancestors.
  - Edges represent evolutionary steps.
  - Root represents the oldest evolutionary ancestor of the existing species represented in the tree.

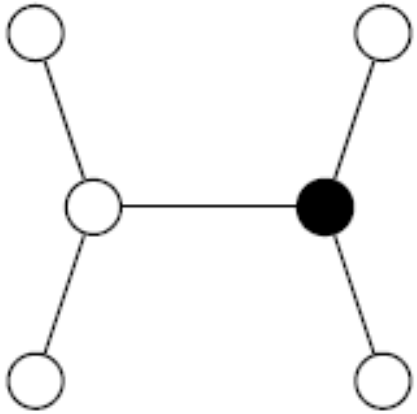
Rooted and Unrooted Trees

- In an **unrooted tree,** the position of the root ("oldest ancestor") is unknown.

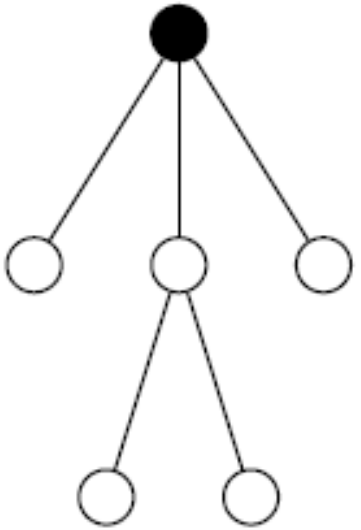
- Otherwise, the tree is **rooted** and can be oriented with the root at the top.



Unrooted Tree



Rooted Tree



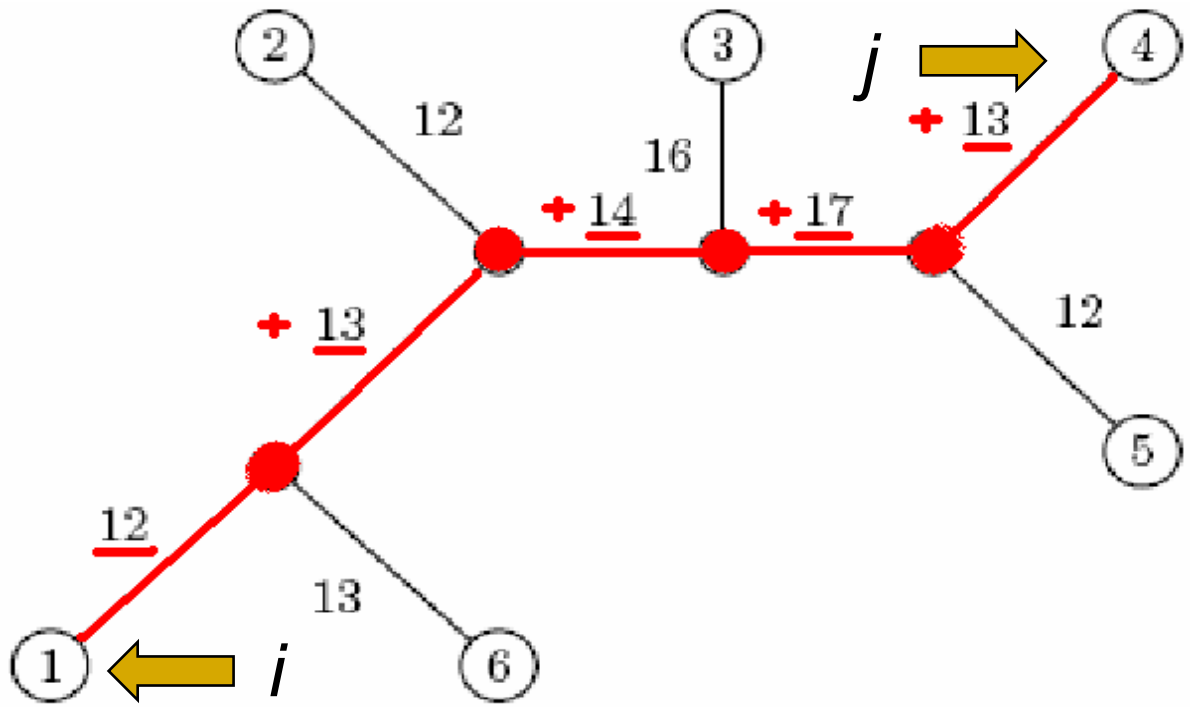
Root at the Top

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:
  - $d_{i,j}(T)$ = the length of a path between leaves i and j
  - = tree distance between i and j

Distance in Trees: Example



$$d_{1,4} = 12 + 13 + 14 + 17 + 12 = 68$$

Distance Matrix

- Given n species, we can compute the n x n distance matrix $D_{i,j}$.
- $D_{i,j}$ 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.
- Note the difference between $D_{i,j}$ and $d_{i,j}(T)$.

Fitting Distance Matrix

- Given n species, we can compute the n x n distance matrix $D_{i,j}$.

- The 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 $D_{i,j}$.

- Fitting means $\underbrace{D_{i,j}}_{\text{Edit distance between species (}known\text{)}} = \overbrace{d_{i,j}(T)}^{\text{Lengths of path in an (}unknown\text{) tree }T}$.

Reconstructing a 3-Leaved Tree

- Tree reconstruction for any 3x3 matrix is straightforward.

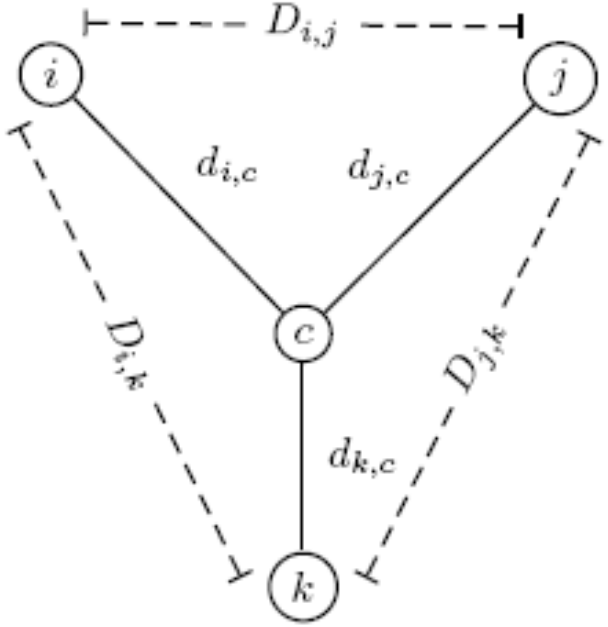
- We have 3 leaves i, j, k and a center vertex c.

- **Observe**:

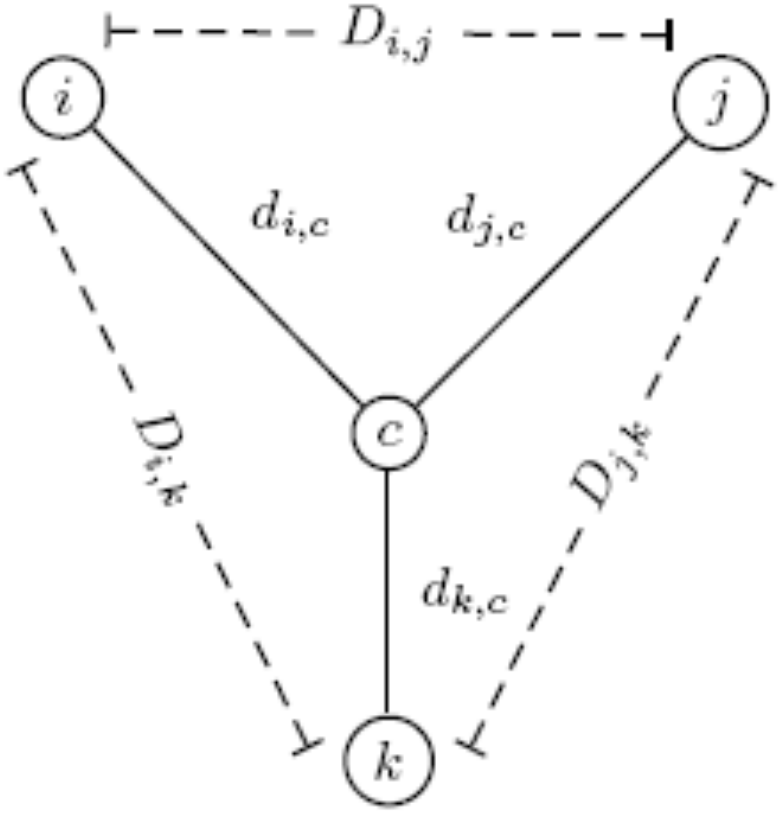
$$d_{i,c} + d_{j,c} = D_{i,j}$$

$$d_{i,c} + d_{k,c} = D_{i,k}$$

$$d_{j,c} + d_{k,c} = D_{j,k}$$



Reconstructing a 3-Leaved Tree



$$d_{i,c} + d_{j,c} = D_{i,j}$$

$$+ \quad [\; d_{i,c} + d_{k,c} = D_{i,k} \;]$$

$$2d_{i,c} + d_{j,c} + d_{k,c} = D_{i,j} + D_{i,k}$$

$$2d_{i,c} + D_{j,k} = D_{i,j} + D_{i,k}$$

$$d_{i,c} = \frac{D_{i,j} + D_{i,k} - D_{j,k}}{2}$$

$$\text{Similarly,} \quad d_{j,c} = \frac{D_{i,j} + D_{j,k} - D_{i,k}}{2}$$

$$\text{and} \quad d_{k,c} = \frac{D_{k,i} + D_{k,j} - D_{i,j}}{2}$$

Trees with > 3 Leaves

- A 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 $C(n, 2)$ equations with $2n - 3$ variables.
  - Recall:

$$C(n,2) = \binom{n}{2} = \frac{n!}{(n-2)! \cdot 2!} = \frac{n(n-1)}{2}$$

- This set of equations is not always possible to solve for $n > 3$, as it was in the $n = 3$ case.

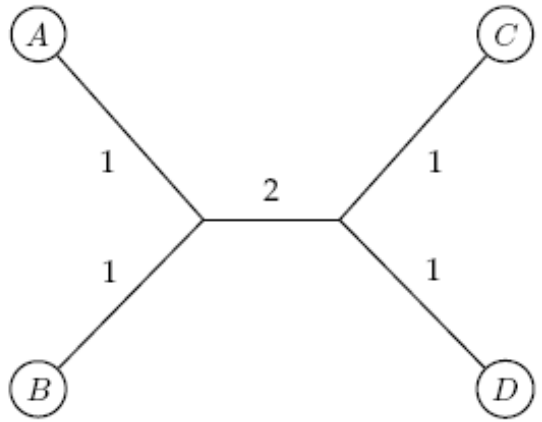
Additive Distance Matrices

- Matrix D is **additive** if there exists a tree T with $d_{i,j}(T) = D_{i,j}$

- D is **nonadditive** otherwise.

Additive Matrix

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



Nonadditive Matrix

	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

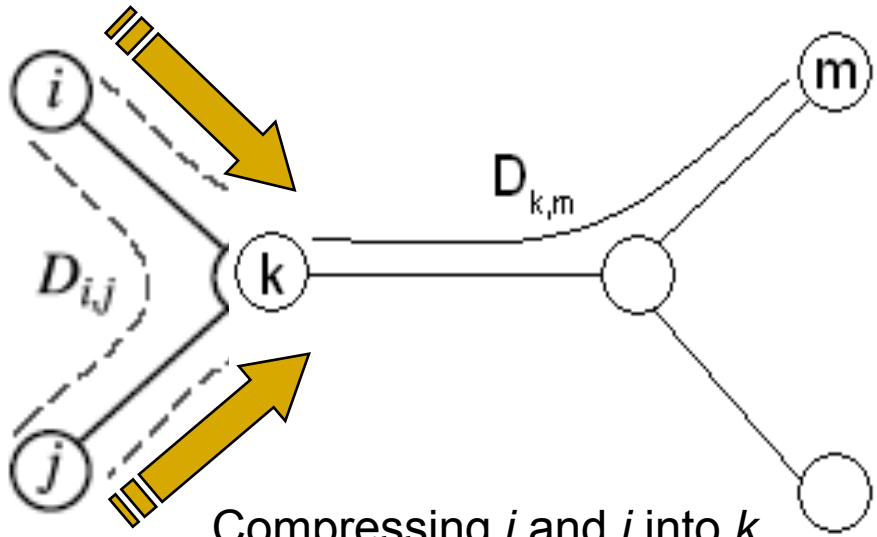
- Goal: Reconstruct an evolutionary tree from a distance matrix.
- Input: n x n distance matrix $D_{i,j}$.
- Output: Weighted tree T with n leaves fitting D.
- If D is additive, this problem has a solution and there is a simple algorithm to solve it.

Section 5: Neighbor-Joining Algorithm

Using Neighboring Leaves to Construct the Tree

1. Find neighboring leaves *i* and *j* with parent *k*.
2. Then *compress* leaves *i* and *j* into *k* as follows:
   - 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 recomputed by the equation below:

$$D_{k,m} = \frac{D_{i,m} + D_{j,m} - D_{i,j}}{2}$$

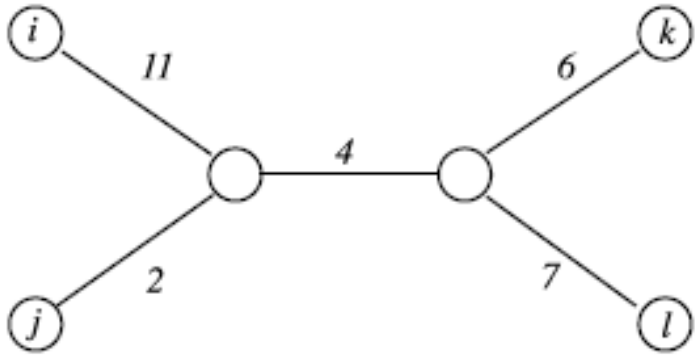


Compressing *i* and *j* into *k*

How Do We Find Neighboring Leaves?

- **Proposal**: To find neighboring leaves we simply select a pair of closest leaves in the tree.
  - **WRONG**

- Closest leaves in the tree aren't necessarily neighbors.
  - **Example**: i and j are neighbors, but $d_{i,j} = 13 > d_{j,k} = 12$



- Finding a pair of neighboring leaves is a nontrivial problem

Neighbor-Joining Algorithm

- **1987**: Naruya Saitou and Masatoshi Nei develop a neighbor-joining algorithm for phylogenetic tree reconstruction.

- **Idea**: 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 and does not have the flawed molecular clock assumption.

Naruya Saitou

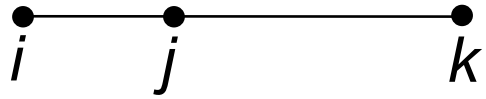
Masatoshi Nei

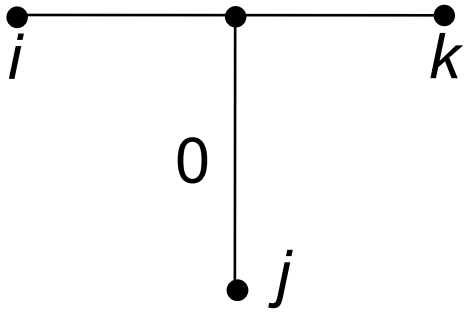
Section 6: Additive Phylogeny

Degenerate Triples

- A **degenerate triple** is a set of three distinct elements $1 \leq i, j, k \leq n$ for which $D_{i,j} + D_{j,k} = D_{i,k}$

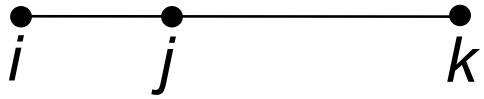
- Interpretation: Element j in a degenerate triple i, j, k lies on the evolutionary path from i to k (or is attached to this path by an edge of length 0).
  - **Example**: Illustrated below are the two possible examples.

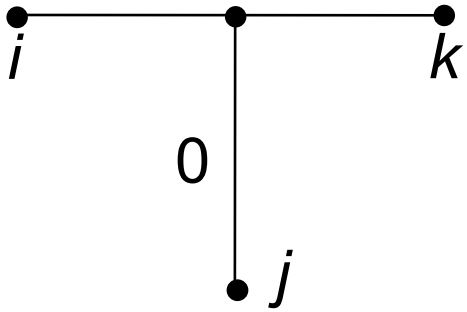




Looking for Degenerate Triples

- Why are degenerate triples of any interest?
- 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).

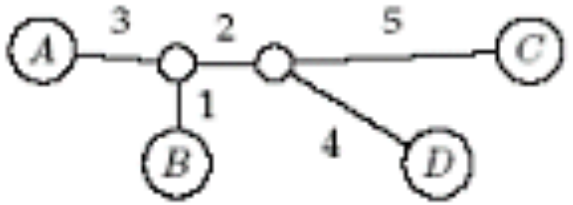




Shortening Hanging Edges

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

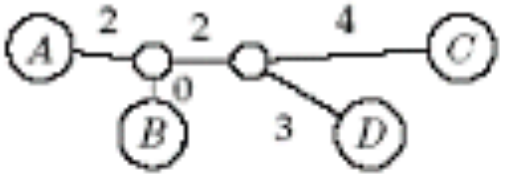
	A	B	C	D
A	0	4	10	9
B	4	0	8	7
C	10	8	0	9
D	9	7	9	0



$\delta = 1$

	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

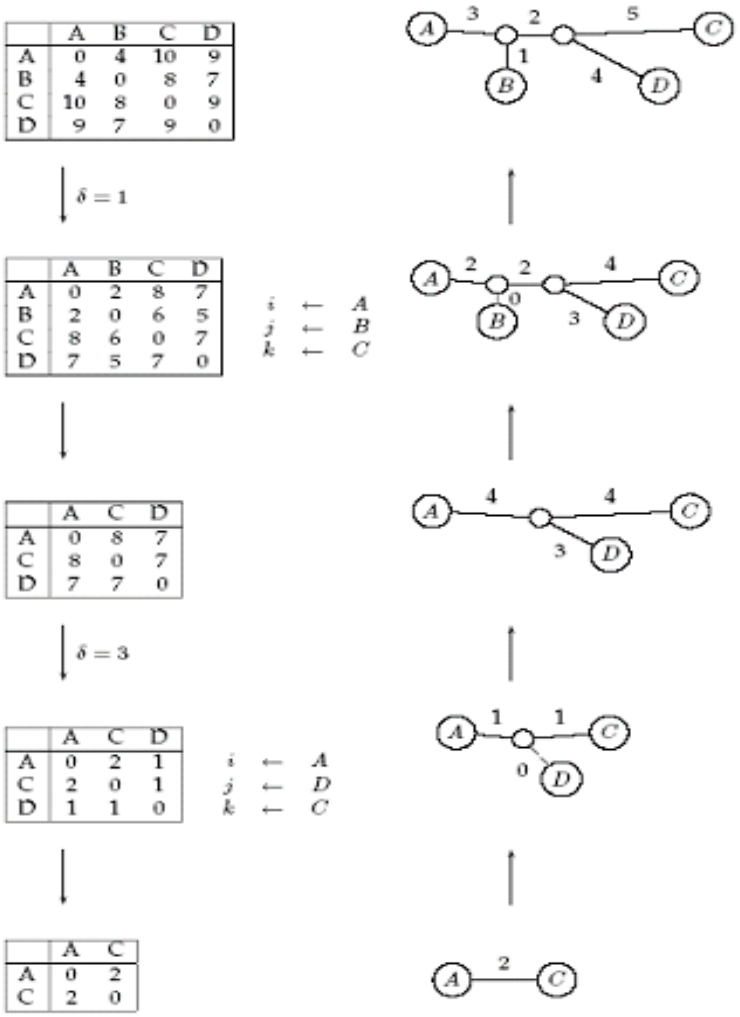
- If there is no degenerate triple, the length of all hanging edges is reduced by the same amount δ, so that all pair-wise distances in the matrix are reduced by 2δ.

- 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 $D_{i,j}$ for each collapsed leaf.

Reconstructing Trees for Additive Matrices

- At right is an example of constructing a tree for an additive matrix.
- The first stage is to continually find degenerate triples and compress the matrix until we only have two vertices.
- We then iteratively draw the tree by reversing this process.



AdditivePhylogeny Algorithm Part I

```
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 ≤ i ≠ j ≤ n
8.              D_ij = D_ij - 2δ
9.      else
10.         δ = 0
```

AdditivePhylogeny Algorithm Part II

11. Find a triple i, j, k in D such that $D_{ij} + D_{jk} = D_{ik}$
12. $x = D_{ij}$
13. Remove j^{th} row and j^{th} column from D
14. T = AdditivePhylogeny(D)
15. Add a new vertex v to T at distance x from i to k
16. Add j back to T by creating an edge (v,j) of length 0
17. **for** every leaf l in T
18. **if** distance from l to v in the tree $\neq D_{l,j}$
19. output "matrix is not additive"
20. **return**
21. Extend all "hanging" edges by length δ
22. **return** T

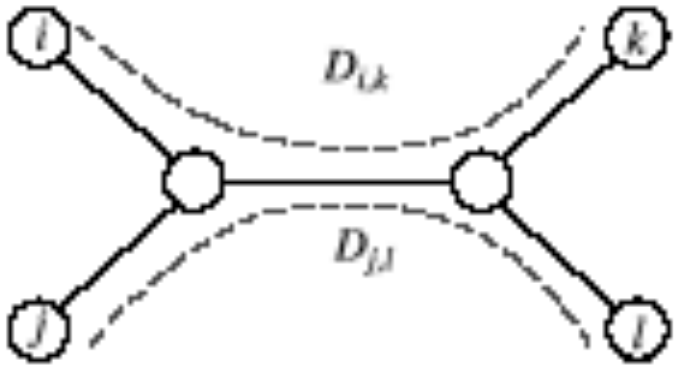
The Four Point Condition

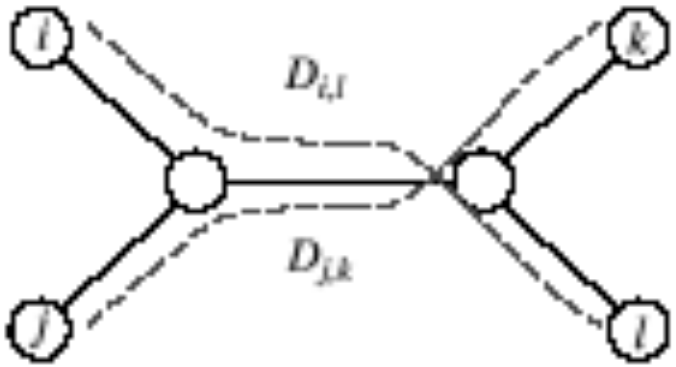
- AdditivePhylogeny provides a way to check if our distance matrix D is additive

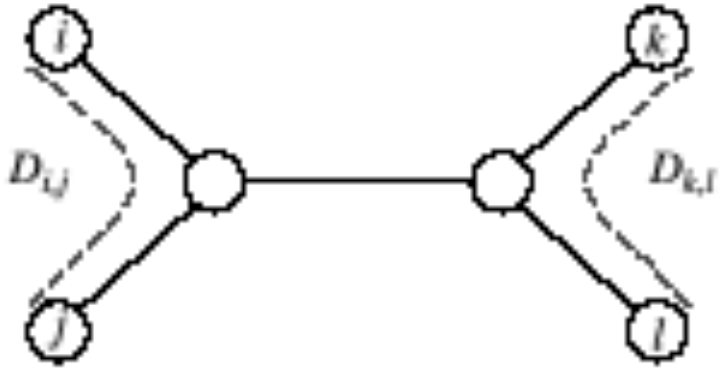
- An even more efficient check for additivity of our matrix is the "four-point condition."

The Four Point Condition: Motivating Example

Compute:

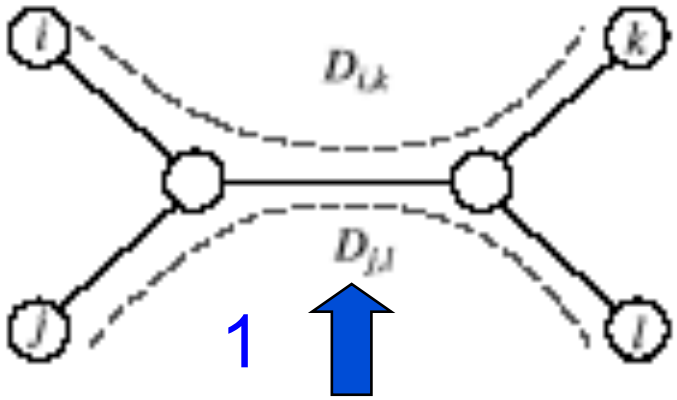


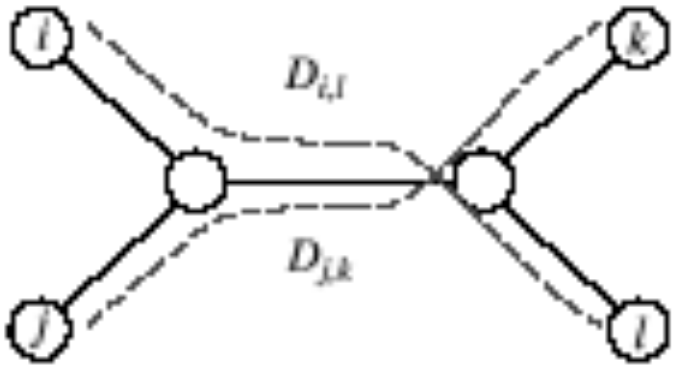


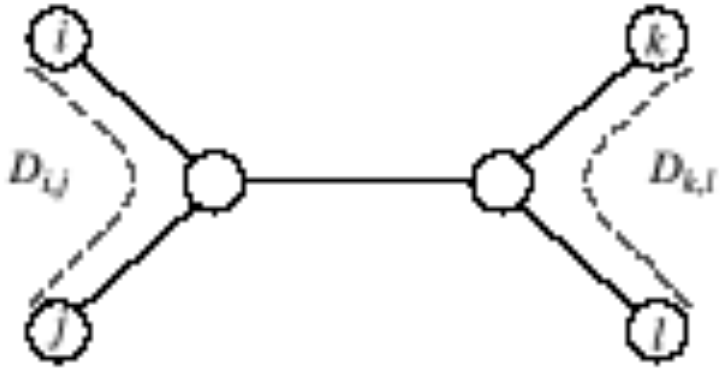


The Four Point Condition: Motivating Example

Compute: 1. $D_{i,k} + D_{j,l}$,

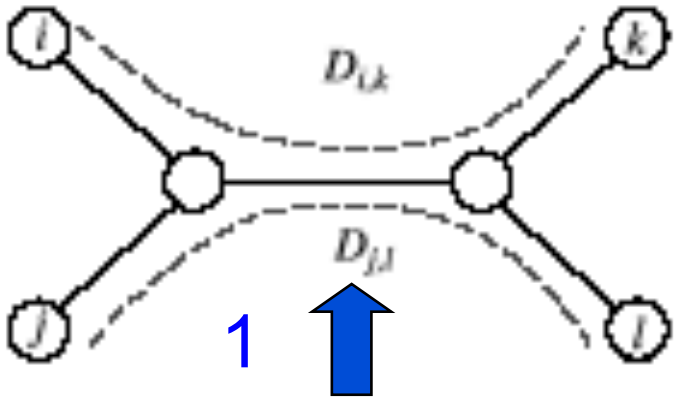


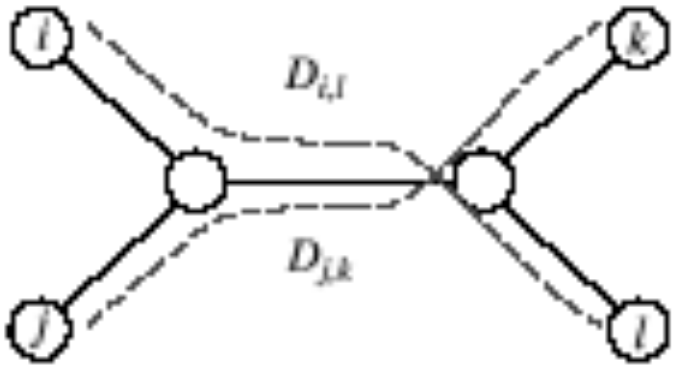


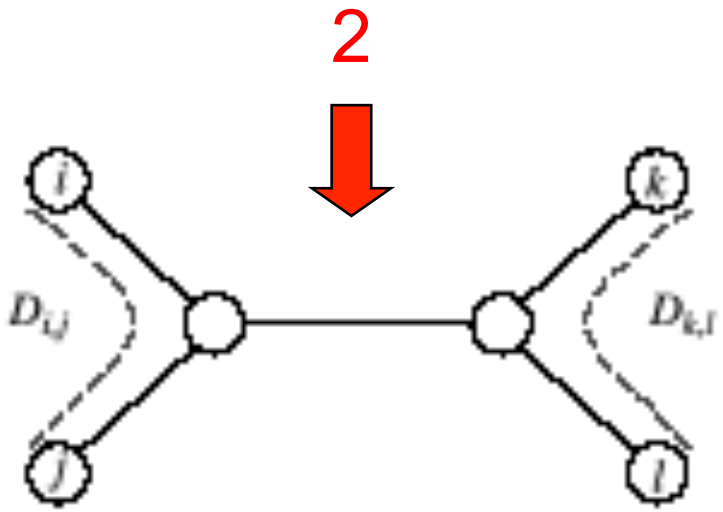


The Four Point Condition: Motivating Example

Compute: 1. $D_{i,k} + D_{j,l}$, 2. $D_{i,j} + D_{k,l}$,

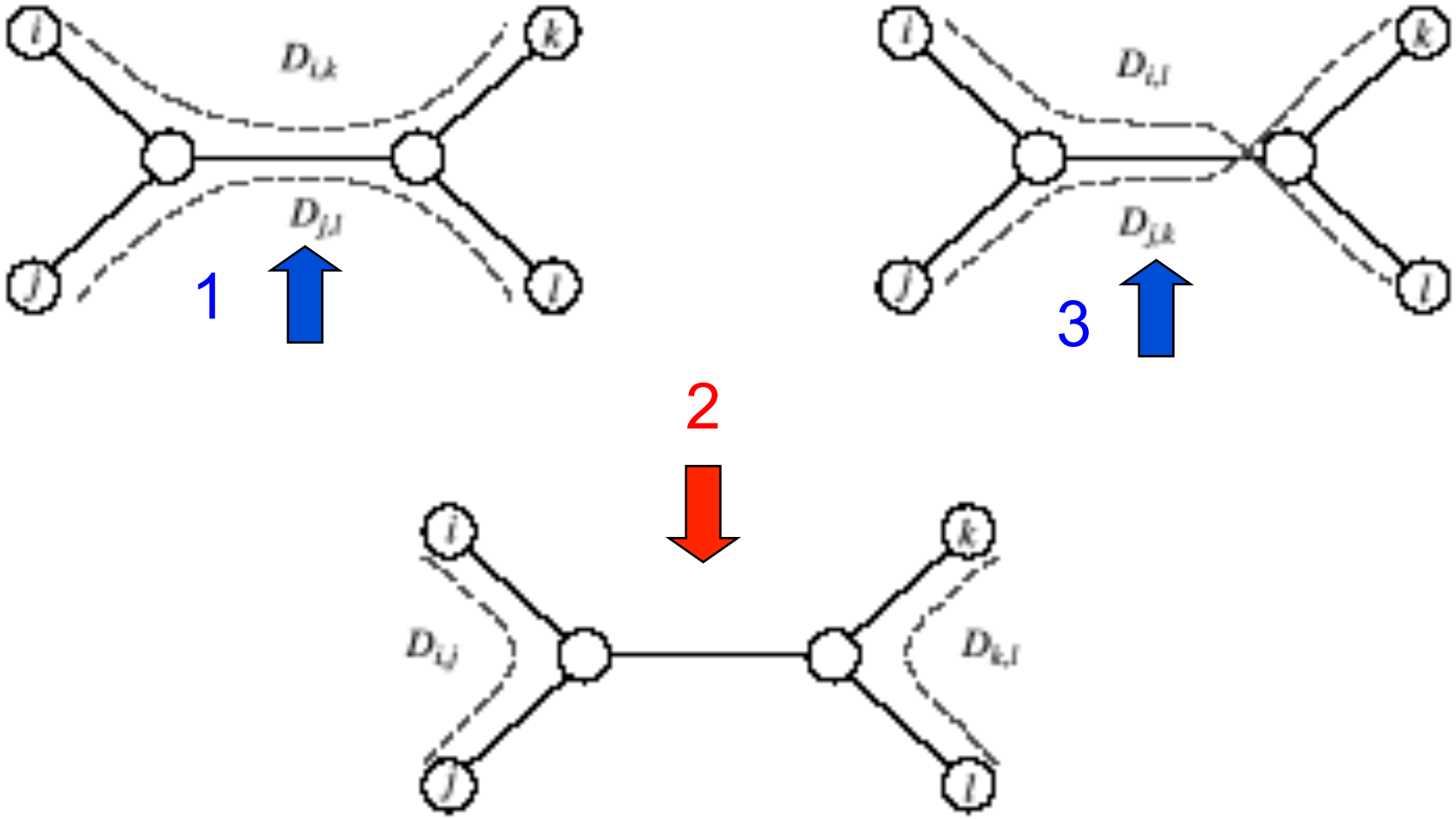






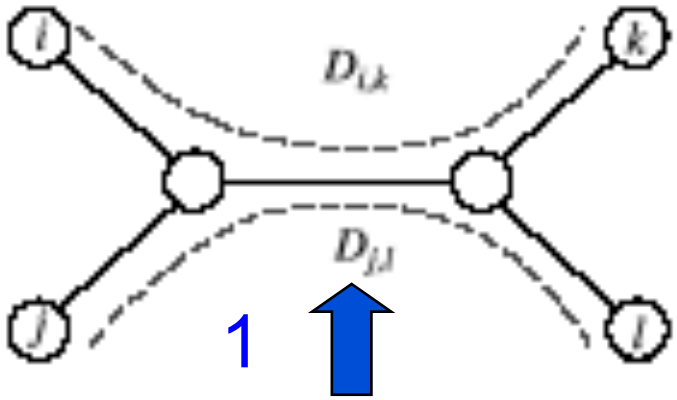
The Four Point Condition: Motivating Example

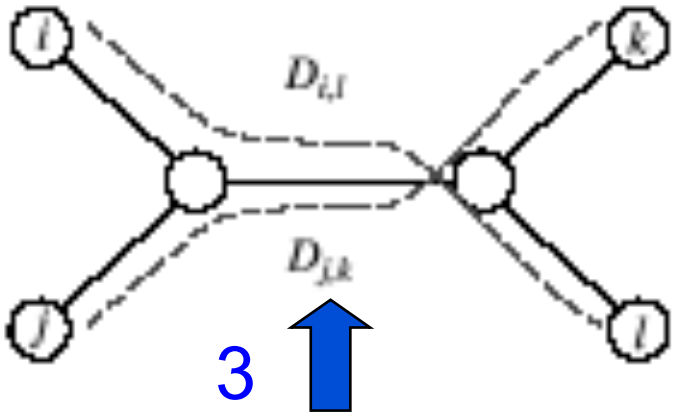
Compute: 1. $D_{i,k} + D_{j,l}$, 2. $D_{i,j} + D_{k,l}$, 3. $D_{i,l} + D_{j,k}$



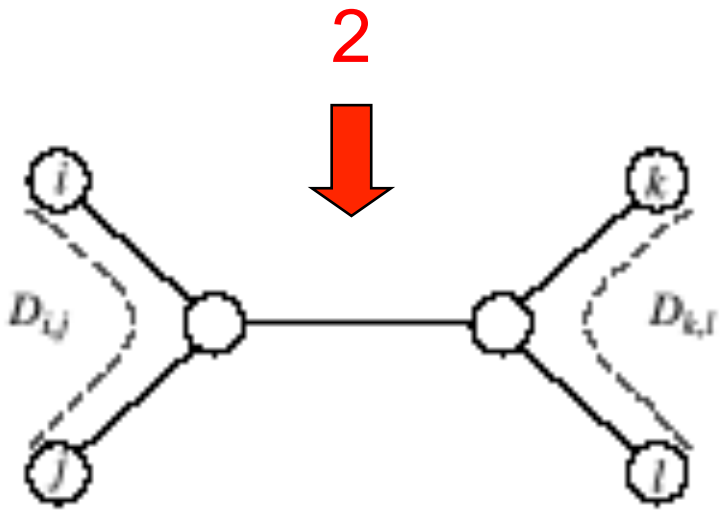
The Four Point Condition: Motivating Example

Compute: 1. $D_{i,k} + D_{j,l}$, 2. $D_{i,j} + D_{k,l}$, 3. $D_{i,l} + D_{j,k}$



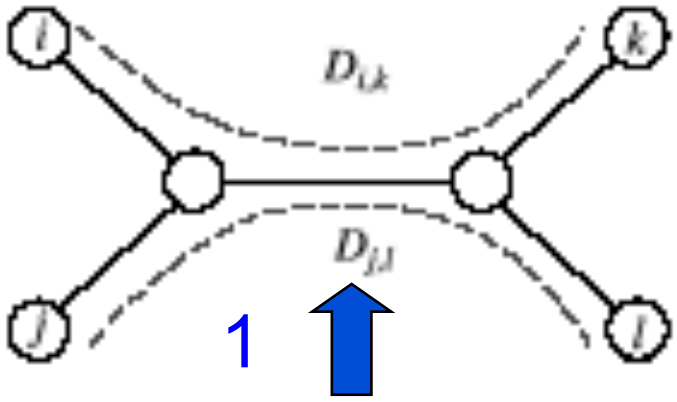


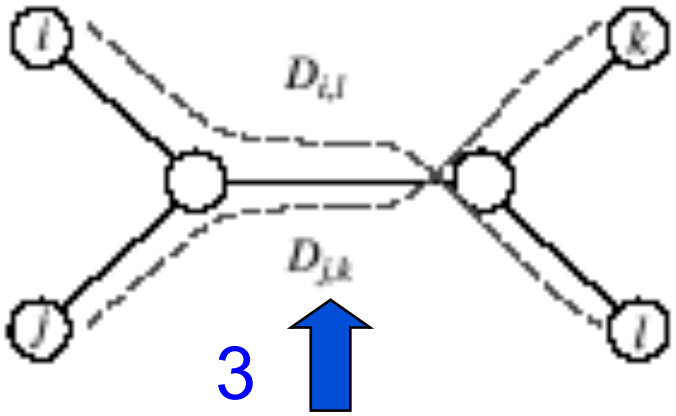
1 and **3** represent the **same number**: **the length of all edges + the middle edge (it is counted twice)**



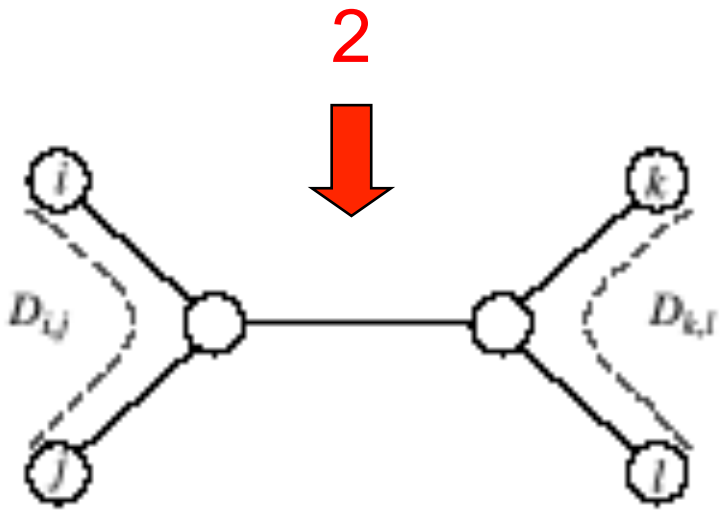
The Four Point Condition: Motivating Example

Compute: 1. $D_{i,k} + D_{j,l}$, 2. $D_{i,j} + D_{k,l}$, 3. $D_{i,l} + D_{j,k}$





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



2 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 the following sums are the same, with the third sum smaller than these first two.
  - $D_{i,j} + D_{k,l}$
  - $D_{i,k} + D_{j,l}$
  - $D_{i,l} + D_{j,k}$

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

Section 7: Least-Squares Distance Phylogeny

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:

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

$$\text{Squared Error} = \sum_{i=1}^{n} \sum_{j=1}^{n} \left(d_{i,j}(T) - D_{i,j} \right)^2$$

- **Least Squares Distance Phylogeny Problem**: Find the best approximation tree T for a non-additive matrix D.
  - This problem is *NP*-Hard.

Section 8: UPGMA

UPGMA

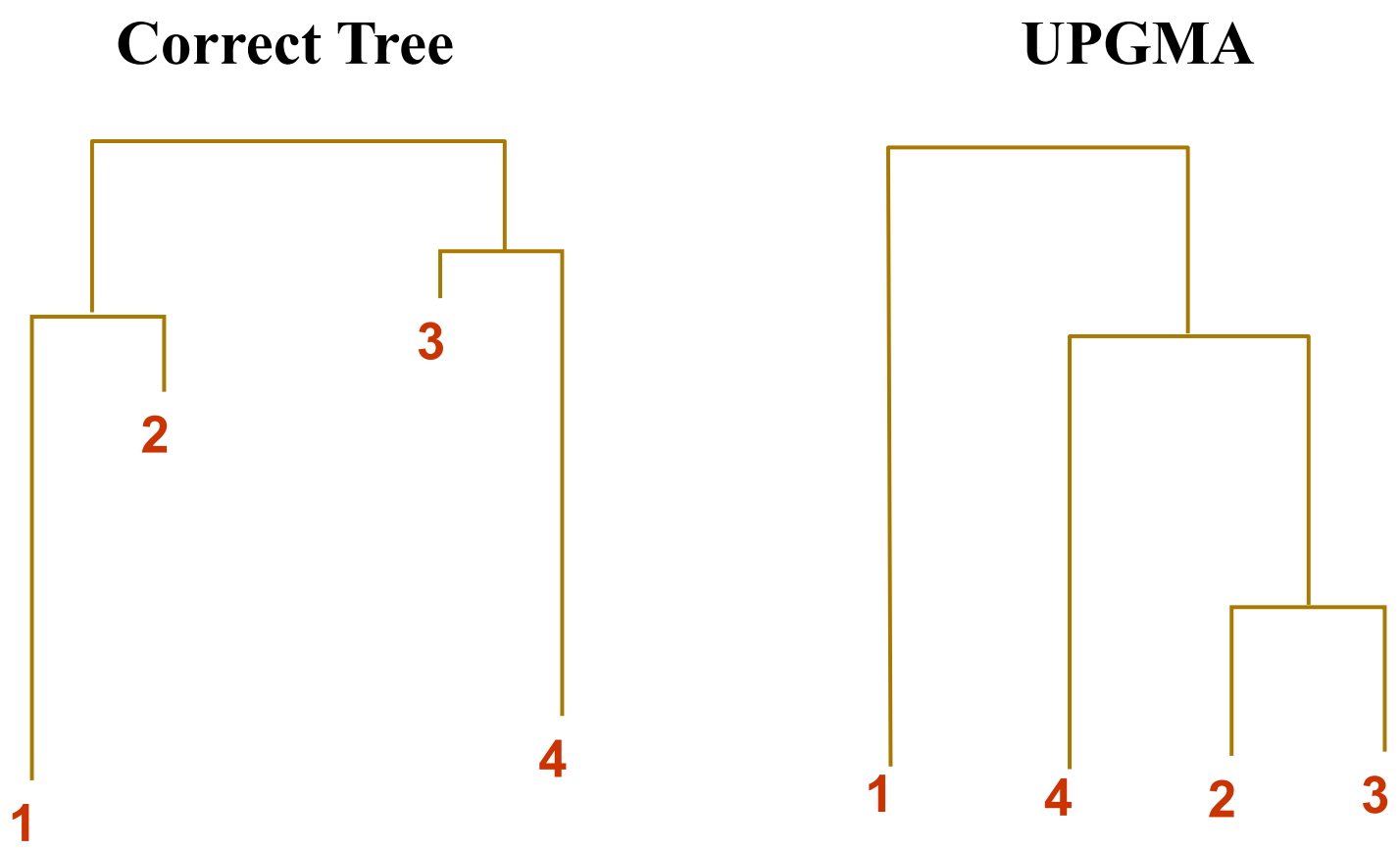
- **UPGMA** is a clustering algorithm that:
  - Computes the distance between clusters using average pairwise distance.
  - Assigns a height to every vertex in the tree, effectively assuming the presence of a *molecular clock* and "dating" every vertex.

UPGMA's Weakness

- UPGMA produces an **ultrametric tree**: The distance from the root to any leaf is the same.

- The reason for this is that UPGMA assumes a constant molecular clock.
  - All species represented by the leaves in the tree are assumed to accumulate mutations (and thus evolve) at the same rate.
  - This is a major pitfall of UPGMA.

UPGMA's Weakness: Example



Clustering in UPGMA

- Given two disjoint clusters of sequences C_i and C_j,

$$d_{i,j} = \frac{1}{|C_i| \cdot |C_j|} \sum_{p \in C_i} \sum_{q \in C_j} d_{p,q}$$

- Note that if C_k is the union of C_i and C_j, then the distance from C_k to another cluster C_l is:

$$d_{k,l} = \frac{d_{i,l}|C_i| + d_{j,l}|C_j|}{|C_i| + |C_j|}$$

UPGMA Algorithm

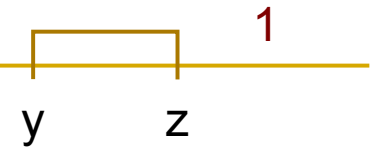
1. **Initialization**:
   - Assign each x_i to its own cluster C_i.
   - Define one leaf per sequence, each at height 0.
2. **Iteration**:
   - Find two clusters C_i and C_j such that $d_{i,j}$ is minimal.
   - Let $C_k = C_i \cup C_j$.
   - Add vertex connecting C_i to C_j and place it at height $d_{i,j}/2$.
   - Delete C_i and C_j.
3. **Termination**:
   - When a single cluster remains.

UPGMA Algorithm: Example

	v	w	x	y	z
v	0	6	8	8	8
w		0	8	8	8
x			0	4	4
y				0	2
z					0

UPGMA Algorithm: Example

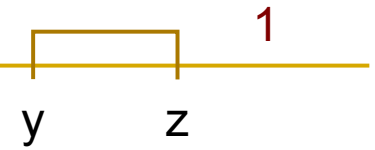
	v	w	x	y	z
v	0	6	8	8	8
w		0	8	8	8
x			0	4	4
y				0	2
z					0



UPGMA Algorithm: Example

	v	w	x	y	z
v	0	6	8	8	8
w		0	8	8	8
x			0	4	4
y				0	2
z					0

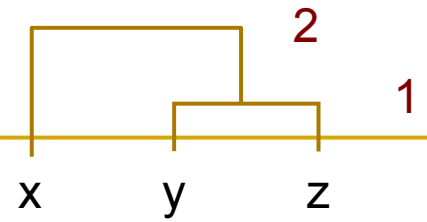
	v	w	x	yz
v	0	6	8	8
w		0	8	8
x			0	4
yz				0



UPGMA Algorithm: Example

	v	w	x	y	z
v	0	6	8	8	8
w		0	8	8	8
x			0	4	4
y				0	2
z					0

	v	w	x	yz
v	0	6	8	8
w		0	8	8
x			0	4
yz				0

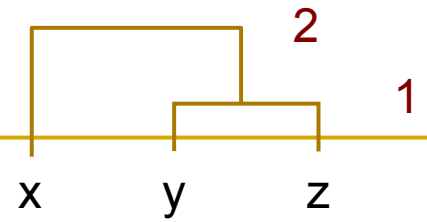


UPGMA Algorithm: Example

	v	w	x	y	z
v	0	6	8	8	8
w		0	8	8	8
x			0	4	4
y				0	2
z					0

	v	w	xyz
v	0	6	8
w		0	8
xyz			0

	v	w	x	yz
v	0	6	8	8
w		0	8	8
x			0	4
yz				0

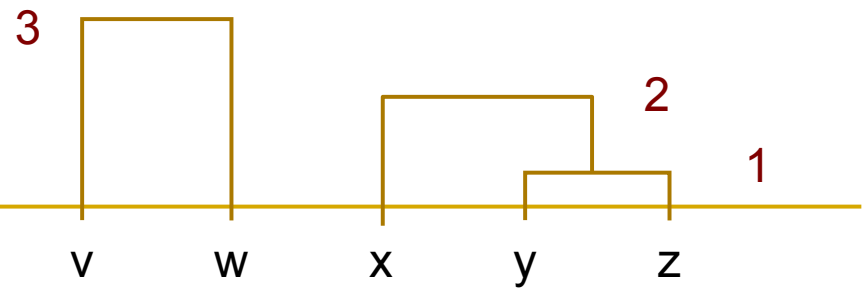


UPGMA Algorithm: Example

	v	w	x	y	z
v	0	6	8	8	8
w		0	8	8	8
x			0	4	4
y				0	2
z					0

	v	w	xyz
v	0	6	8
w		0	8
xyz			0

	v	w	x	yz
v	0	6	8	8
w		0	8	8
x			0	4
yz				0



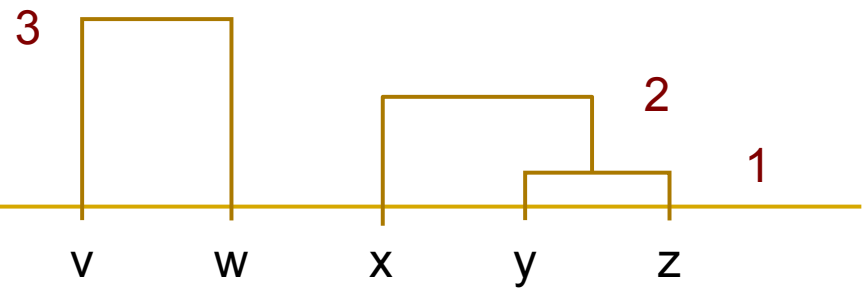
UPGMA Algorithm: Example

	v	w	x	y	z
v	0	6	8	8	8
w		0	8	8	8
x			0	4	4
y				0	2
z					0

	v	w	xyz
v	0	6	8
w		0	8
xyz			0

	vw	xyz
vw	0	8
xyz		0

	v	w	x	yz
v	0	6	8	8
w		0	8	8
x			0	4
yz				0



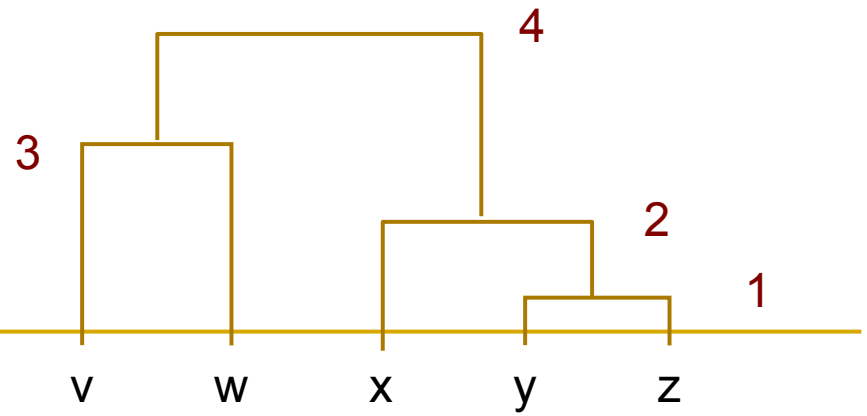
UPGMA Algorithm: Example

	v	w	x	y	z
v	0	6	8	8	8
w		0	8	8	8
x			0	4	4
y				0	2
z					0

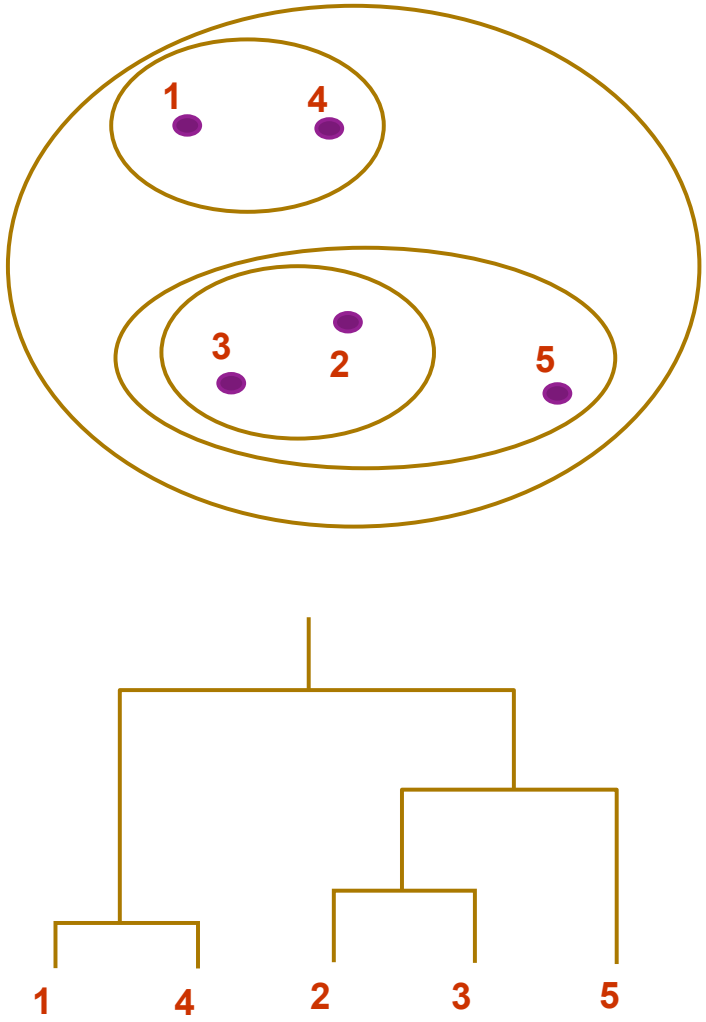
	v	w	xyz
v	0	6	8
w		0	8
xyz			0

	vw	xyz
vw	0	8
xyz		0

	v	w	x	yz
v	0	6	8	8
w		0	8	8
x			0	4
yz				0



UPGMA Algorithm: Illustration



Section 9: Character-Based Phylogeny

Alignment Matrix vs. Distance Matrix

- Sequence a gene of *m* nucleotides in *n* species to generate an *n* x *m* alignment matrix.

- We can see that we can transform a given alignment matrix into a distance matrix, simply by considering edit/Hamming distance.

- However, we *cannot* transform a given matrix into a unique alignment matrix, since information is lost when transforming an alignment matrix into a distance matrix.

Character-Based Tree Reconstruction

- Better technique: **Character-based reconstruction algorithms** use the *n* x *m* alignment matrix.
  - (*n* = # of species, *m* = # of characters)
  - Use alignment matrix directly instead of using distance matrix.

- Goal: Determine what character strings at internal nodes would best explain the character strings for the *n* observed species.

Character-Based Tree Reconstruction

- Characters may be nucleotides, where A, G, C, T are **states** of this character. Other characters may be the # of eyes or legs or the shape of a beak or a fin.

- By setting the length of an edge in the tree to the Hamming distance, we may define the **parsimony score** of the tree as the sum of the lengths (weights) of the edges.

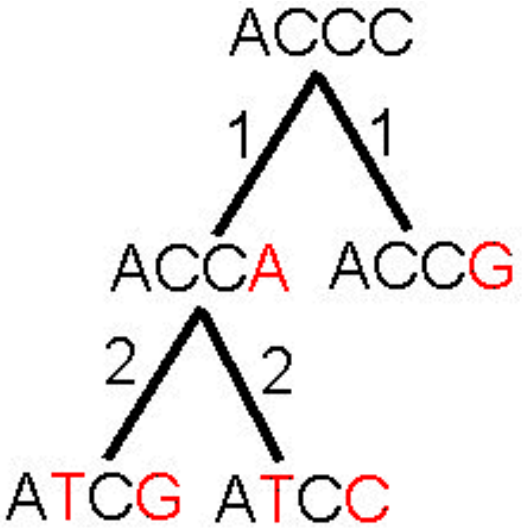
Section 10: Small Parsimony Problem

Parsimony Approach to Tree Reconstruction

- **Parsimony**: Applies Occam's razor principle: even though there are many possible trees explaining evolutionary relationships, we identify the simplest explanation for the data.
  - In laymen's terms, Occam's razor might be compared to **KISS**: **K**eep **I**t **S**imple **S**tupid
  - Parsimony therefore assumes observed character differences resulted from the *fewest* possible mutations.
  - Seeks the tree that yields lowest possible parsimony score: Sum of cost of all mutations found in the tree.

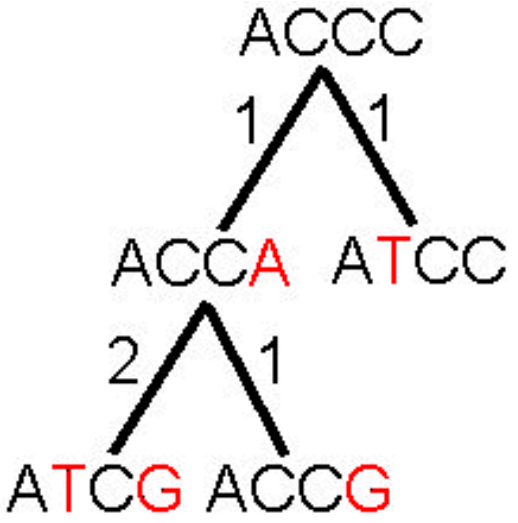
Parsimony and Tree Reconstruction

Less Parsimonious



Score = 6

More Parsimonious



Score = 5

Small Parsimony Problem

- Input: Tree *T* with each leaf labeled by an *m*-character string.
- Output: Labeling of the internal vertices of *T* minimizing the parsimony score.
- We can assume that every leaf is labeled by a single character, because the characters in the string are independent.
- **Note**: To avoid confusion, we reiterate that a "most parsimonious" labeling will provide the *smallest* parsimony score.

Weighted Small Parsimony Problem

- A more general version of Small Parsimony Problem.
- Input includes a k x k scoring matrix describing the cost of transformation of each of k states into another one.
- For Small Parsimony problem, the scoring matrix is simply based on Hamming distance, which we can recall is given by:

$$d_H(v,w) = \begin{cases} 0 & \text{if } v = w \\ 1 & \text{otherwise} \end{cases}$$

Scoring Matrices: Example

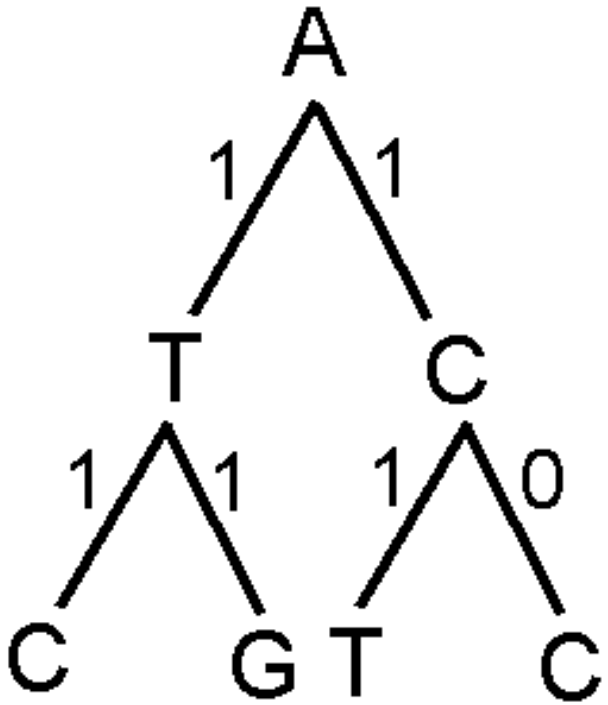
Small Parsimony Problem

	A	T	G	C
A	0	1	1	1
T	1	0	1	1
G	1	1	0	1
C	1	1	1	0

Weighted Parsimony Problem

	A	T	G	C
A	0	3	4	9
T	3	0	2	4
G	4	2	0	4
C	9	4	4	0

Unweighted vs. Weighted

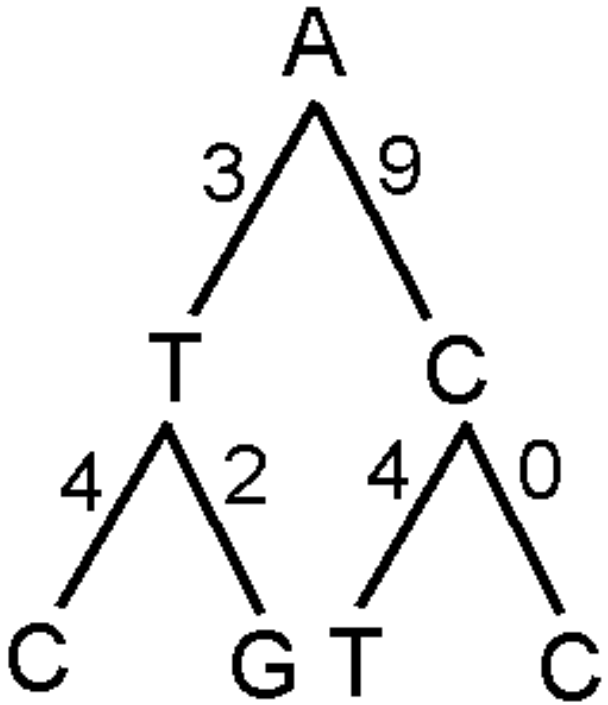


Small Parsimony Scoring Matrix

	A	T	G	C
A	0	1	1	1
T	1	0	1	1
G	1	1	0	1
C	1	1	1	0

Small Parsimony Score: 5

Unweighted vs. Weighted



Weighted Parsimony Scoring Matrix

	A	T	G	C
A	0	3	4	9
T	3	0	2	4
G	4	2	0	4
C	9	4	4	0

Weighted Parsimony Score: 22

Weighted Small Parsimony Problem

- Input: Tree T with each leaf labeled by elements of a k-letter alphabet and a k x k scoring matrix ($\delta_{i,j}$)

- Output: Labeling of internal vertices of T minimizing the weighted parsimony score.

Section 11: Fitch and Sankoff Algorithms

Sankoff's Algorithm: Dynamic Programming

- Calculate and keep track of a score for every possible label at each vertex.
  - $s_t(v)$ = minimum parsimony score of the subtree rooted at vertex v if v has character t.

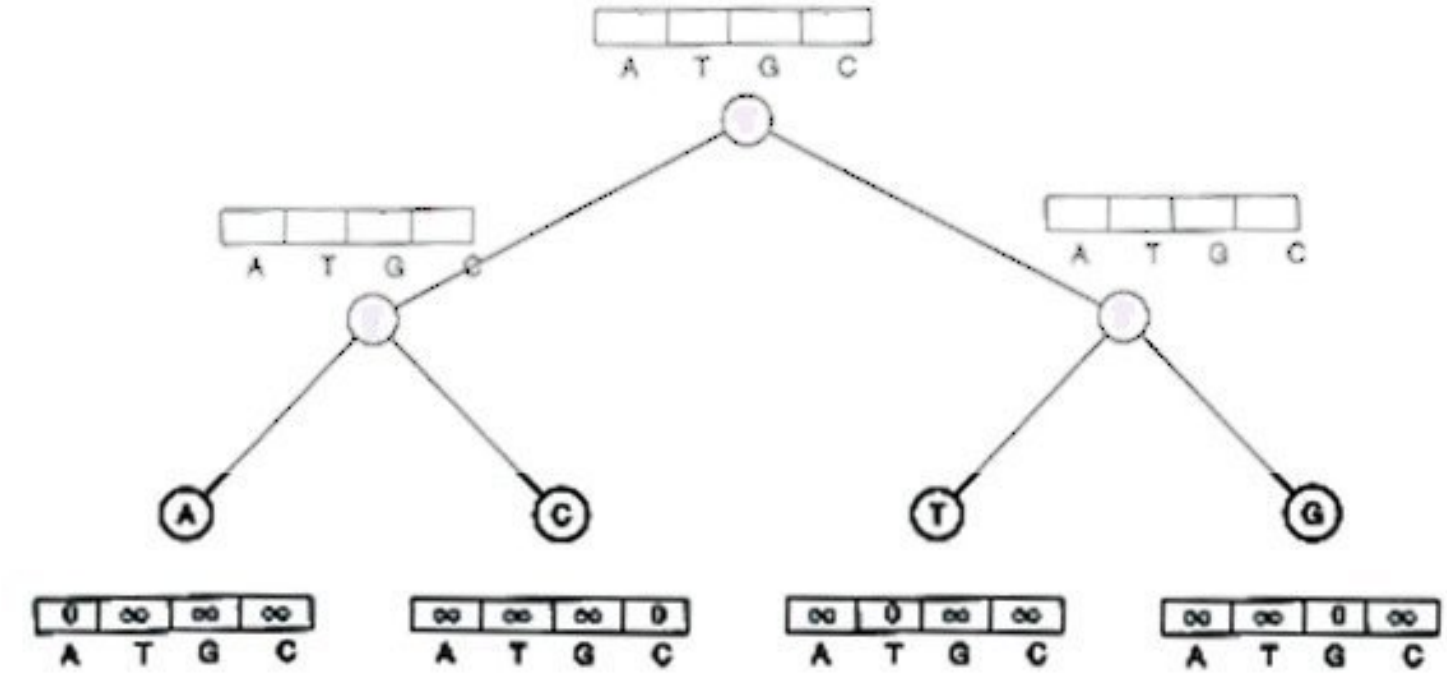
- The score at each vertex is based on scores of its children:

$$s_t(\text{parent}) = \min_i \{ s_i(\text{left child}) + \delta_{i,t} \} + \min_j \{ s_j(\text{right child}) + \delta_{j,t} \}$$

- Therefore, the parsimony score of the entire tree will be the score at the root.

Sankoff Algorithm

- Begin at leaves:
  - If leaf has the character in question, score is 0.
  - Else, score is ∞.



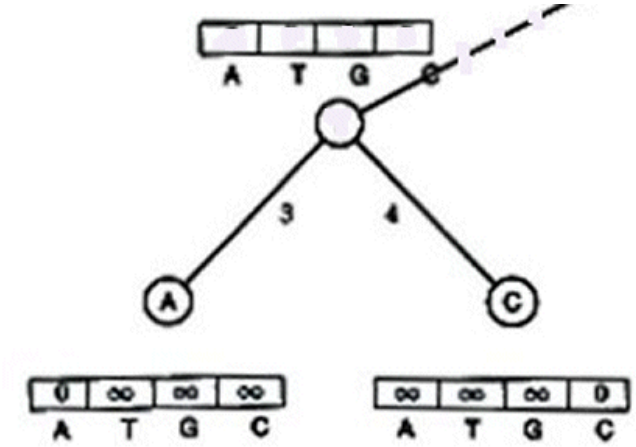
Sankoff Algorithm

- Next, apply the dynamic programming formulas along with the table to move up a level:

δ	A	T	G	C
A	0	3	4	9
T	3	0	2	4
G	4	2	0	4
C	9	4	4	0

$$s_t(v) = \min_i \{ s_i(u) + \delta_{i,t} \} + \min_j \{ s_j(w) + \delta_{j,t} \}$$

$$s_A(v) = 0 + \min_j \{ s_j(w) + \delta_{j,A} \}$$



	$s_i(u)$	$\delta_{i,A}$	sum
A	0	0	0
T	∞	3	∞
G	∞	4	∞
C	∞	9	∞

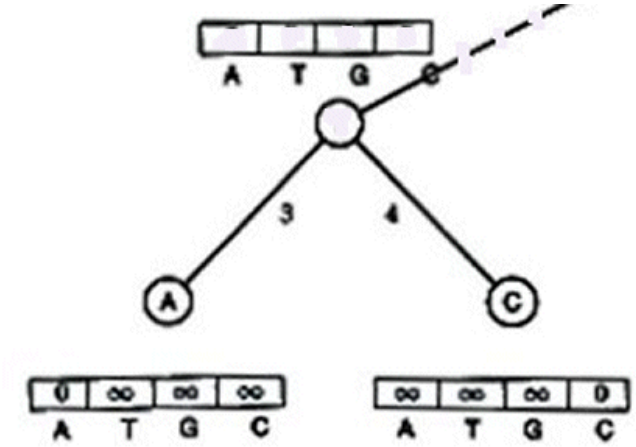
Sankoff Algorithm

- Next, apply the dynamic programming formulas along with the table to move up a level:

δ	A	T	G	C
A	0	3	4	9
T	3	0	2	4
G	4	2	0	4
C	9	4	4	0

$$s_t(v) = \min_i \{ s_i(u) + \delta_{i,t} \} + \min_j \{ s_j(w) + \delta_{j,t} \}$$

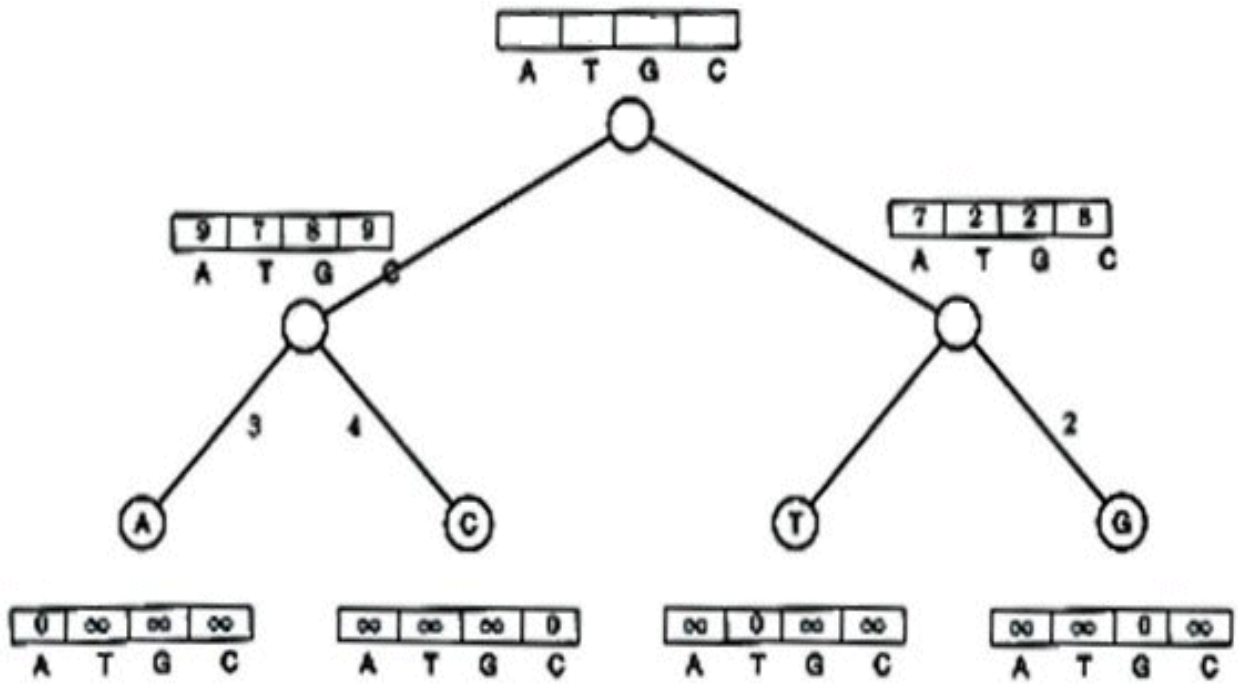
$$s_A(v) = 0 + 9 = 9$$



	$s_i(u)$	$\delta_{i,A}$	sum
A	0	0	0
T	∞	3	∞
G	∞	4	∞
C	∞	9	∞

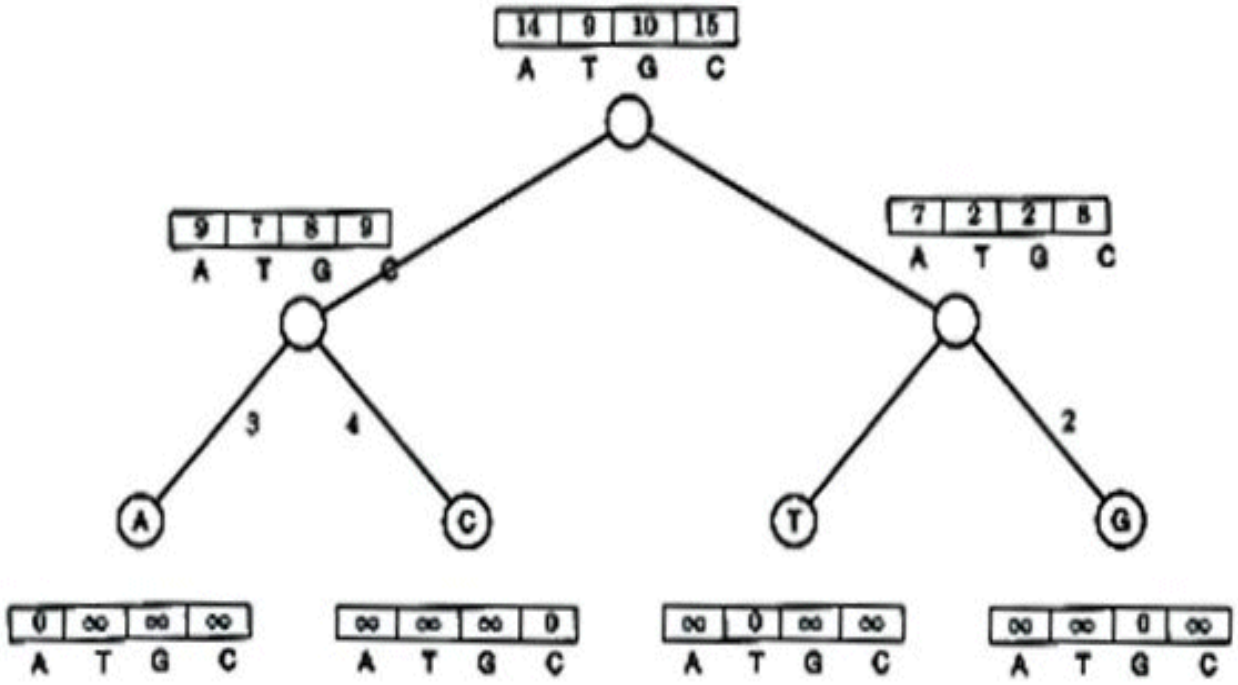
Sankoff Algorithm

- Repeat for right subtree:



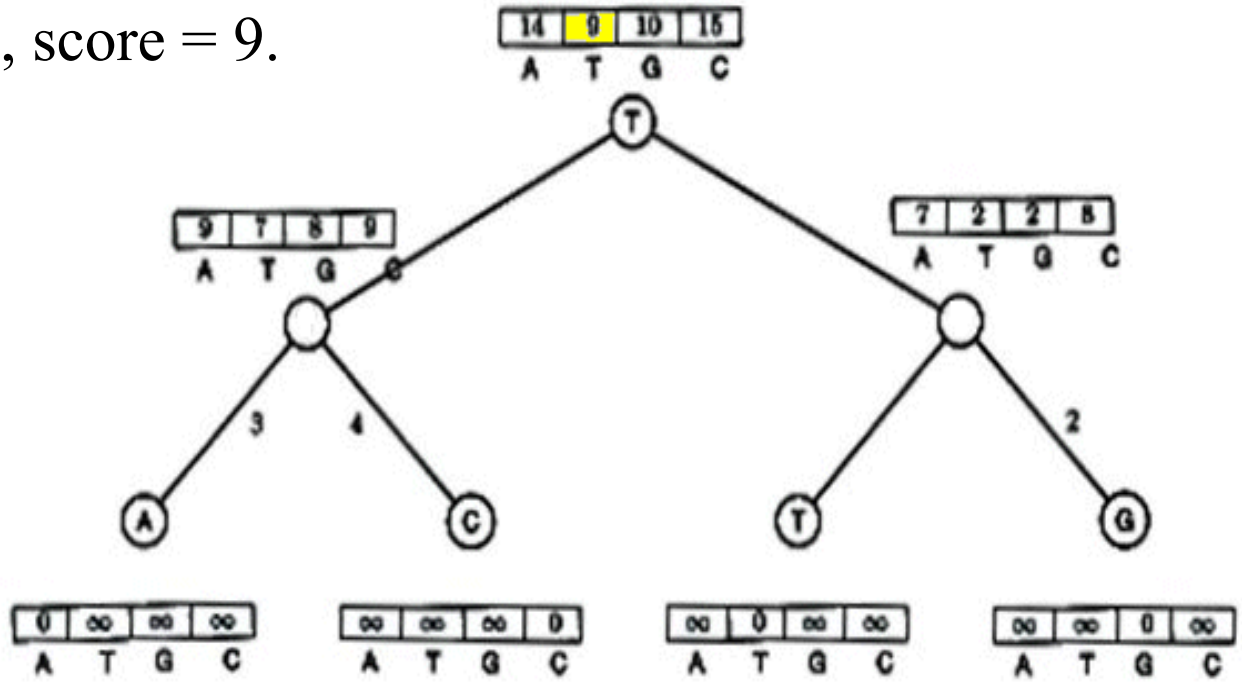
Sankoff Algorithm

- Repeat for right subtree:
- Repeat for root:



Sankoff Algorithm

- Repeat for right subtree:
- Repeat for root:
- Smallest score at root is minimum weighted parsimony score.
  - In this case, score = 9.



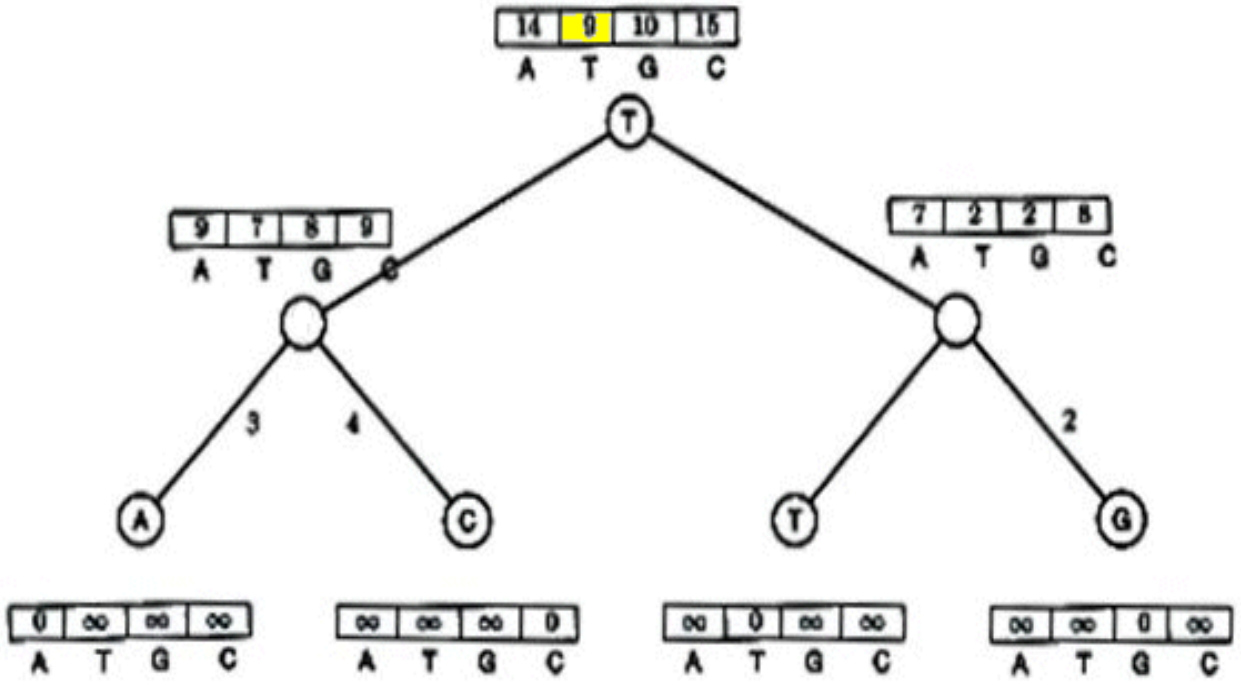
Sankoff Algorithm: Traveling down the Tree

- The scores at the root vertex have been computed by going up the tree.

- After the scores at root vertex are computed, the Sankoff algorithm moves down the tree and assigns each vertex with an optimal character.

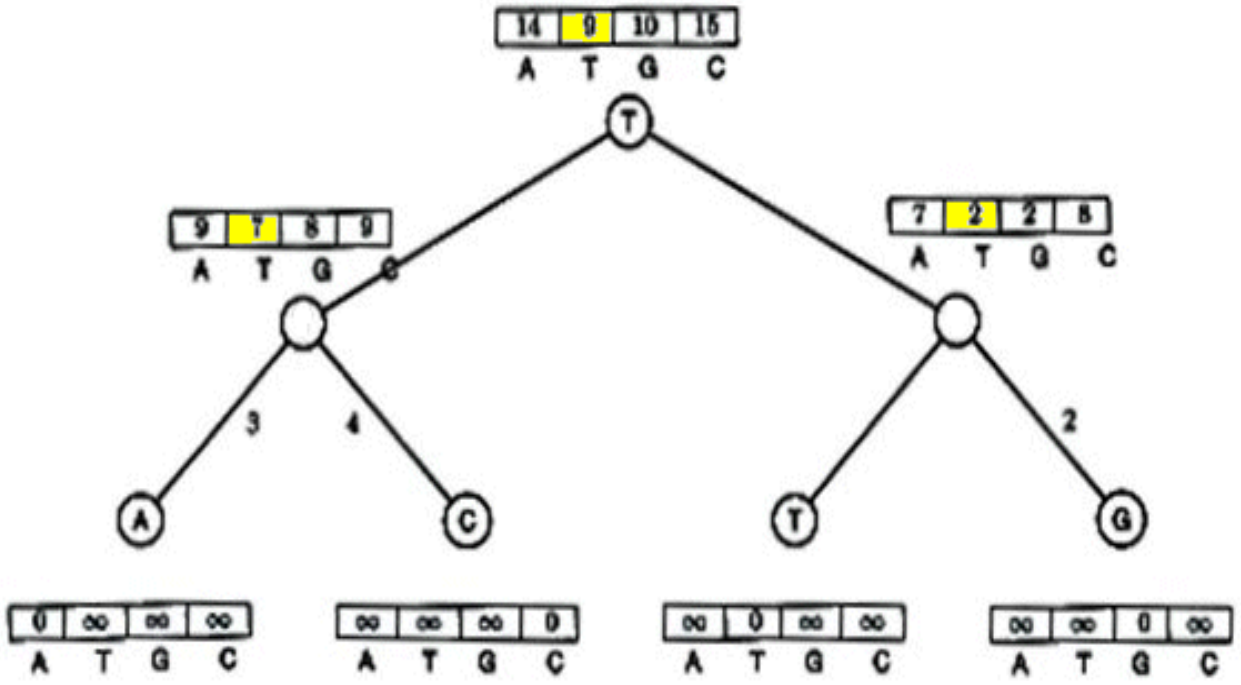
Sankoff Algorithm

- 9 is derived from 7 + 2.



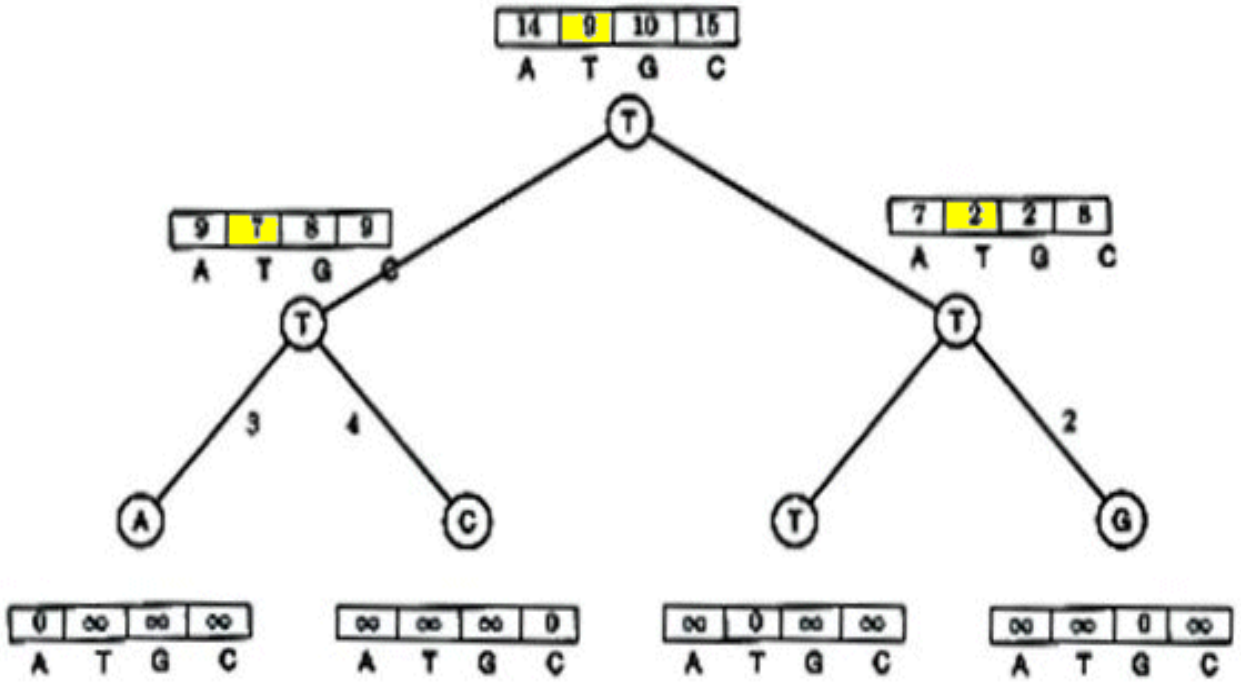
Sankoff Algorithm

- 9 is derived from 7 + 2
  - So the left child is T and the right child is T.



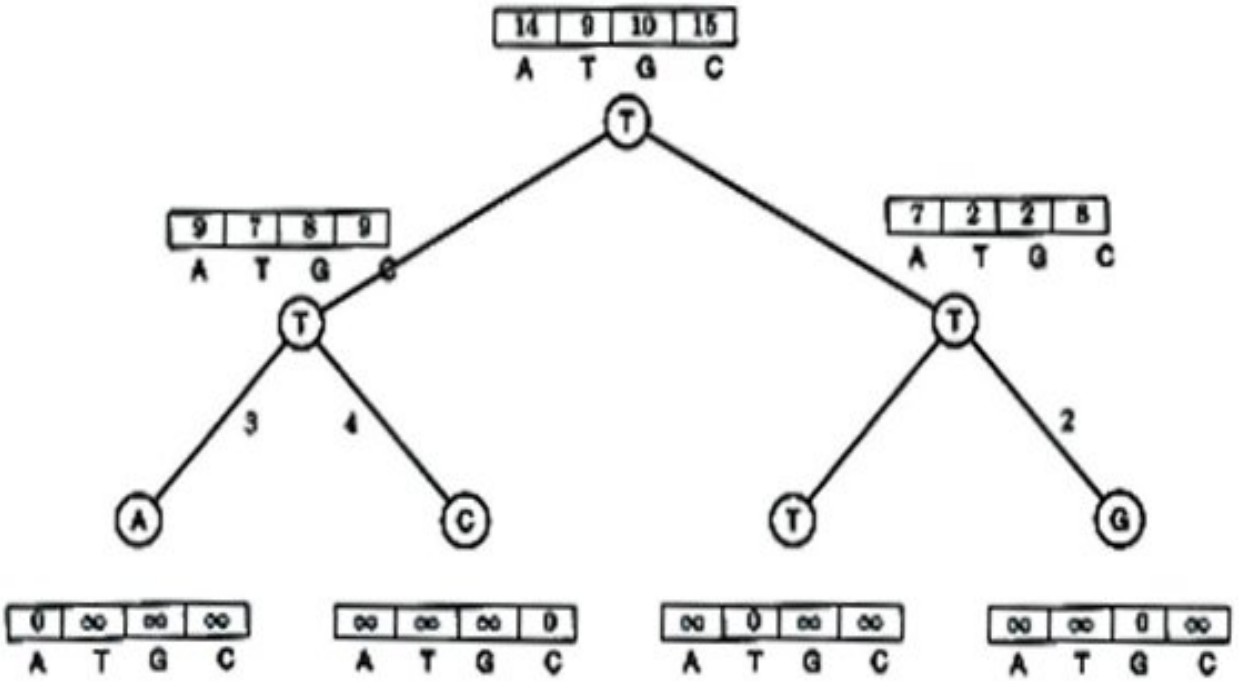
Sankoff Algorithm

- 9 is derived from 7 + 2
  - So the left child is T and the right child is T.



Sankoff Algorithm

- 9 is derived from 7 + 2
  - So the left child is T and the right child is T.
- And the tree is thus labeled…



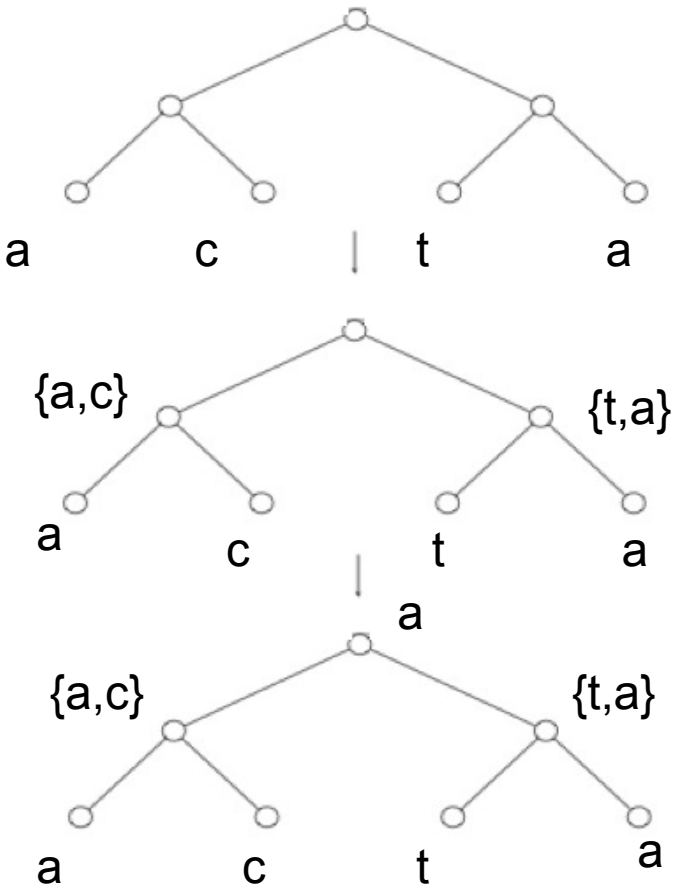
Fitch's Algorithm

- Solves Small Parsimony problem.

- Assigns a set of letters to every vertex in the tree.
  - Each leaf will be labeled with its observed character.

- As we move up the tree, for each parent vertex:
  - If its two children's sets of characters overlap, the parent's set of characters is the set of characters common to both children.
  - Otherwise, the parent's set of characters is the combined set of its children's characters.

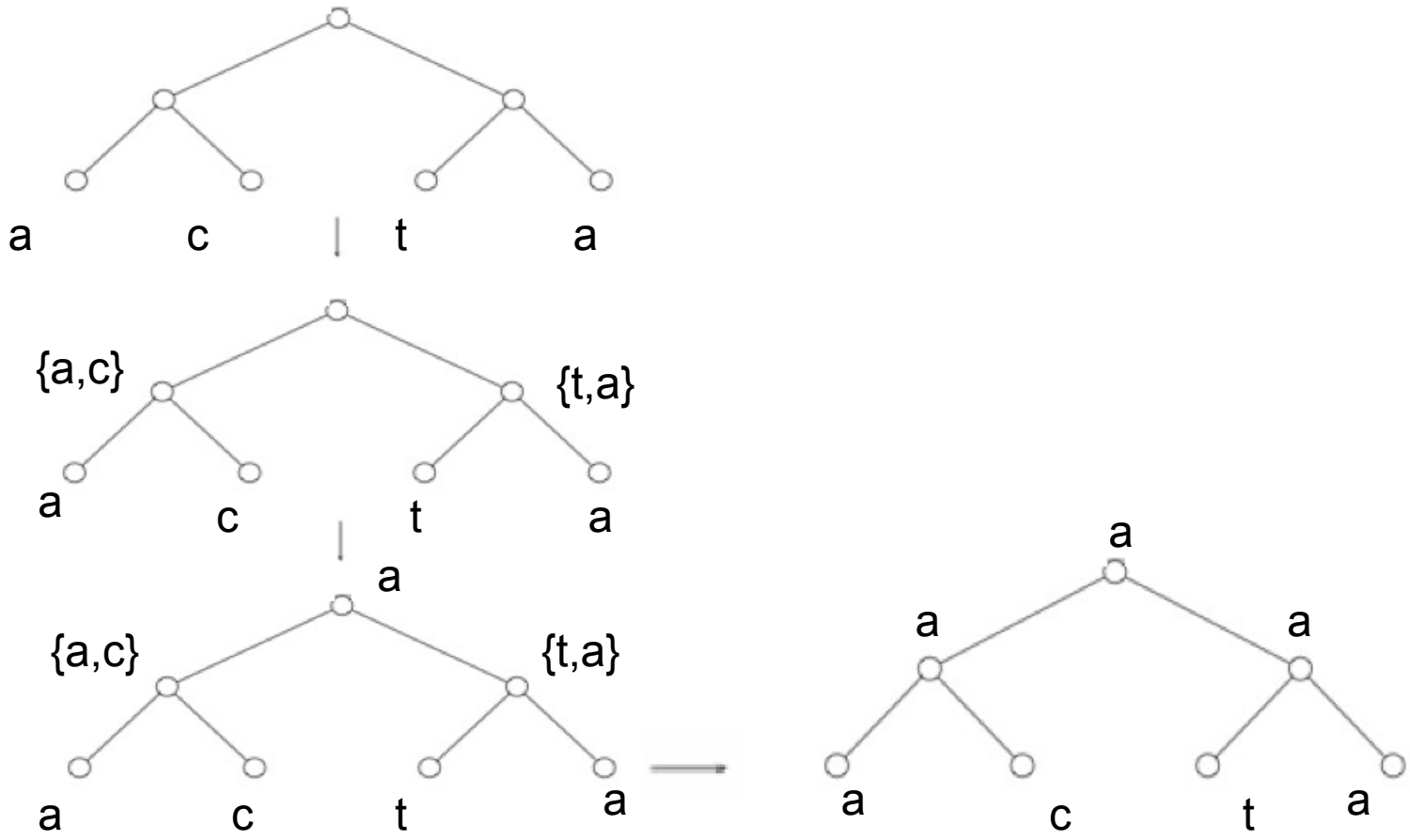
Fitch's Algorithm: Example



Fitch's Algorithm

- Next assign **labels** to each vertex, traversing the tree from root to leaves.
  - Assign root a label arbitrarily from its set of letters.
  - For all other vertices, if its parent's label is in its set of characters, assign to it its parent's label.
  - Otherwise, choose an arbitrary character from its set as its label.

Fitch's Algorithm: Example

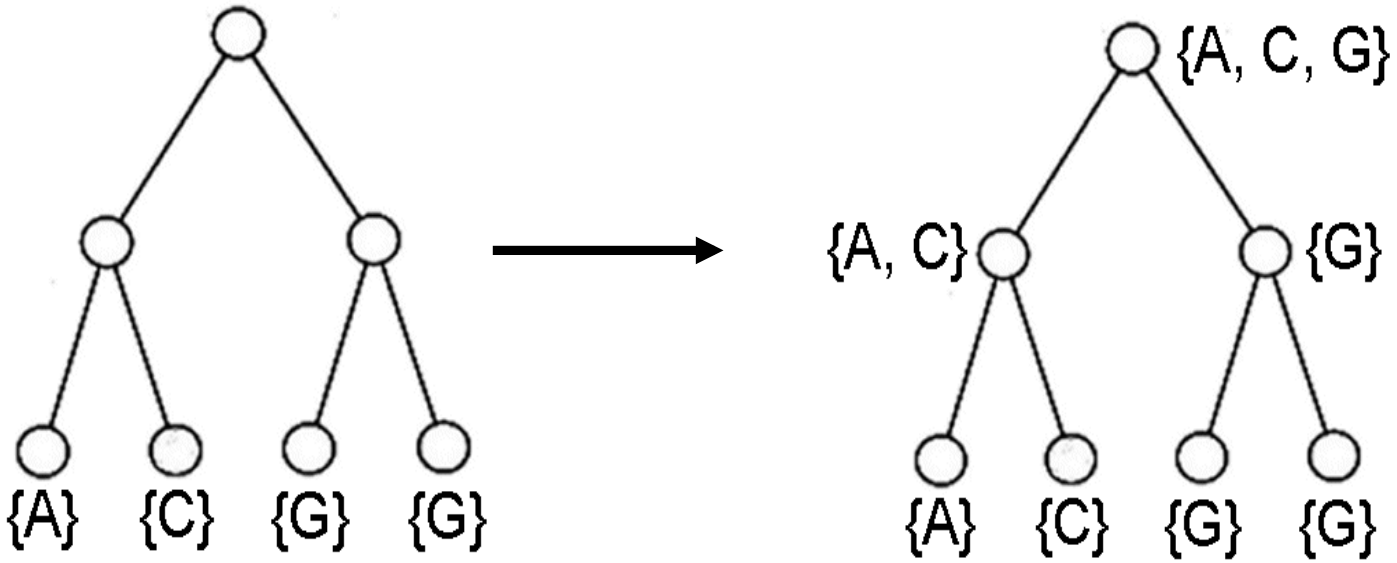


Fitch vs. Sankoff

- Both have O(nk) runtime
- Are they actually different algorithms?
- Let's compare …

Fitch

- As seen previously:



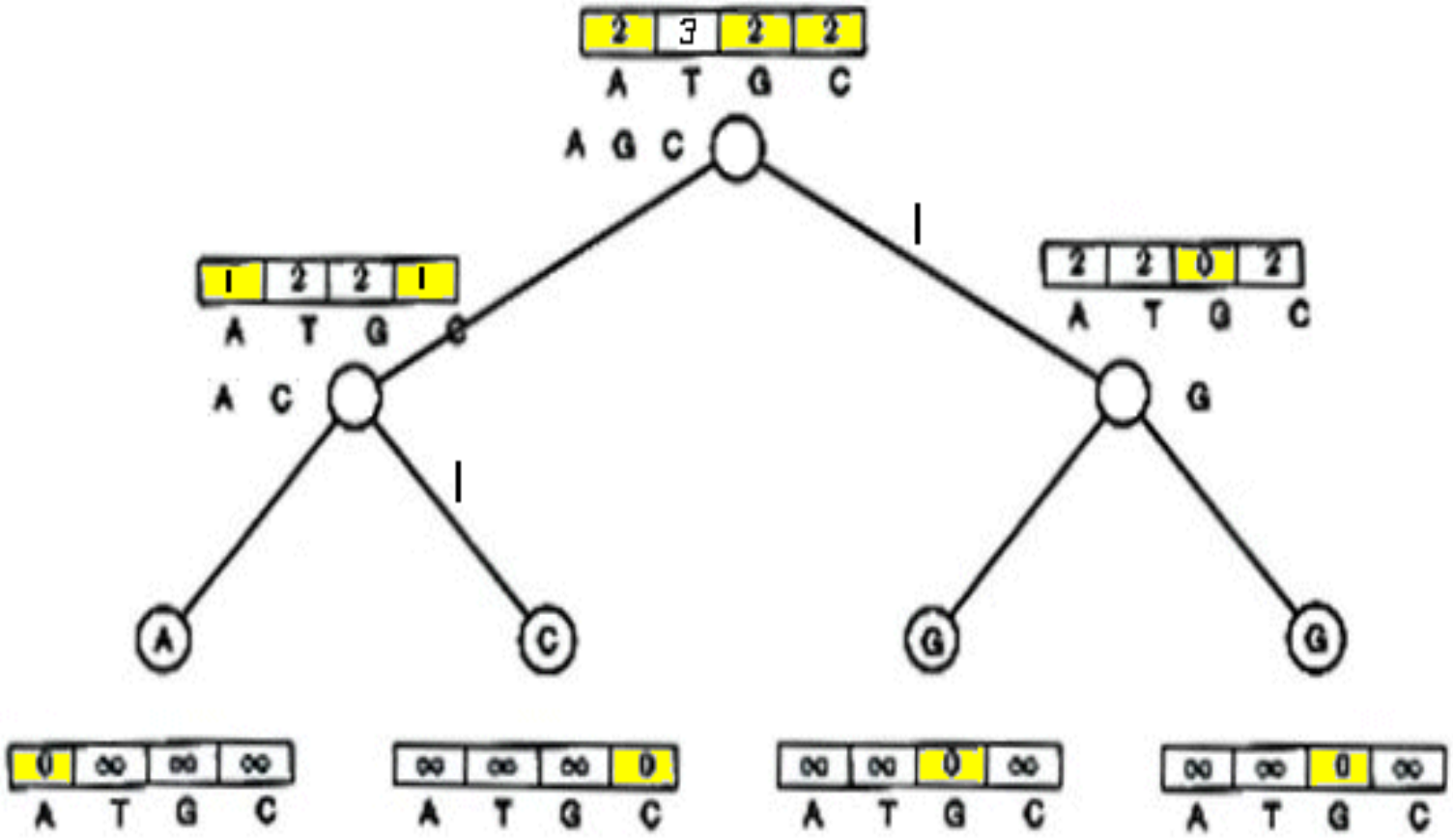
Comparison of Fitch and Sankoff

- The scoring matrix for the Fitch algorithm is merely:

	A	T	G	C
A	0	1	1	1
T	1	0	1	1
G	1	1	0	1
C	1	1	1	0

- So let's do the same problem using Sankoff algorithm and this scoring matrix.

Sankoff


2 3 2 2
A T G C
A G C
1
1 2 2 1
A T G
A C
1
2 2 0 2
A T G C
G
A
C
G
G
0 ∞ ∞ ∞
A T G C
∞ ∞ ∞ 0
A T G C
∞ ∞ 0 ∞
A T G C
∞ ∞ 0 ∞
A T G C

Sankoff vs. Fitch

- For Sankoff algorithm, character t is optimal for vertex v if $s_t(v) = \min_{1<i<k} s_i(v)$.
  - Denote the set of optimal letters at vertex v as $S(v)$.
    - If S(left child) and S(right child) overlap, S(parent) is the intersection.
    - Else it's the union of S(left child) and S(right child)
    - This is the same as the Fitch recurrence.
  - Therefore the two algorithms are *identical*.

Section 12: Large Parsimony Problem

Large Parsimony Problem

- Input: An n x m matrix M describing n species, each represented by an m-character string.

- Output: A tree T with n leaves labeled by the n rows of matrix M, and a labeling of the internal vertices such that the parsimony score is minimized over all possible trees and all possible labelings of internal vertices.

- So in the large parsimony problem, we have to *construct* the tree ourselves. It is therefore (at least superficially) more difficult.

Large Parsimony Problem

- The Possible search space is huge, especially as n increases.
  - The number of possible rooted trees with n leaves:

$$\frac{(2n-3)!}{2^{n-2}(n-2)!}$$

  - The number of possible unrooted trees with n leaves:

$$\frac{(2n-5)!}{2^{n-3}(n-3)!}$$

- In fact, the large parsimony problem is *NP*-Complete.
  - Exhaustive search is only possible with small n (< 10).

- Hence, branch and bound methods or heuristics are used.

Section 13: Nearest Neighbor Interchange

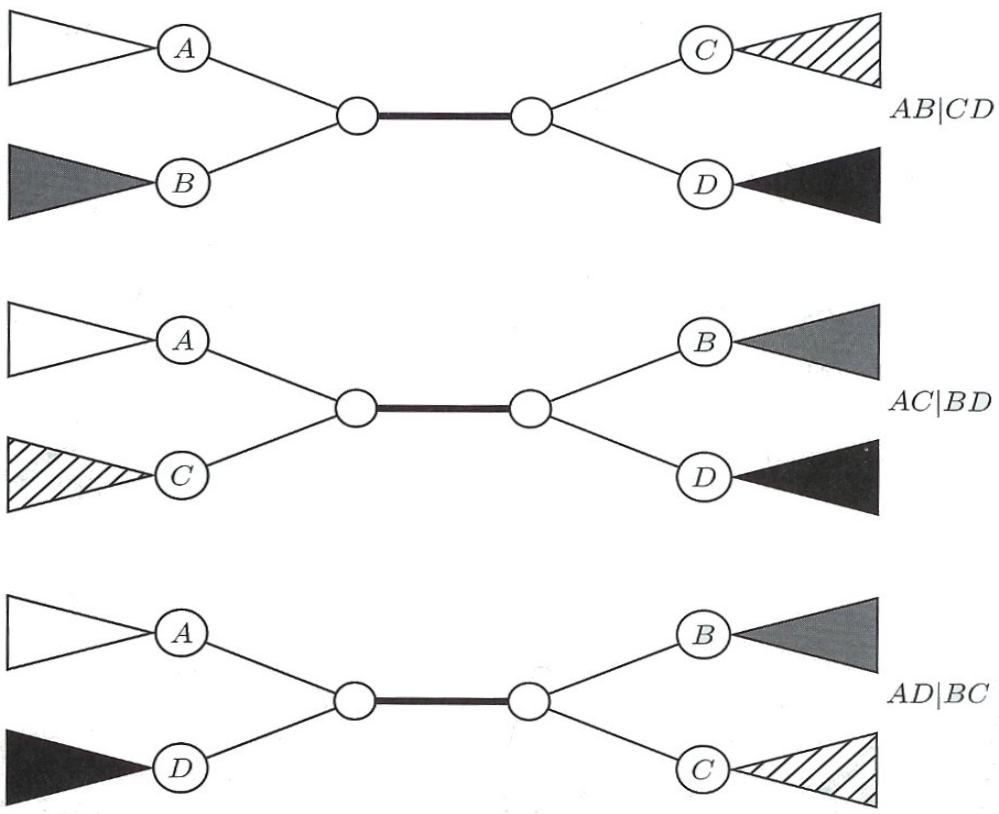
Nearest Neighbor Interchange: Greedy Algorithm

- **Nearest Neighbor Interchange**: A branch-swapping algorithm.
- Only evaluates a *subset* of all possible trees.
- Defines a **neighbor** of a tree as one reachable by a *nearest neighbor interchange*:
  - A rearrangement of the four subtrees defined by one internal edge.
  - Only three different possible such arrangements per edge.

Neighbors: Example

- The trees on the right are all nearest neighbors to each other.

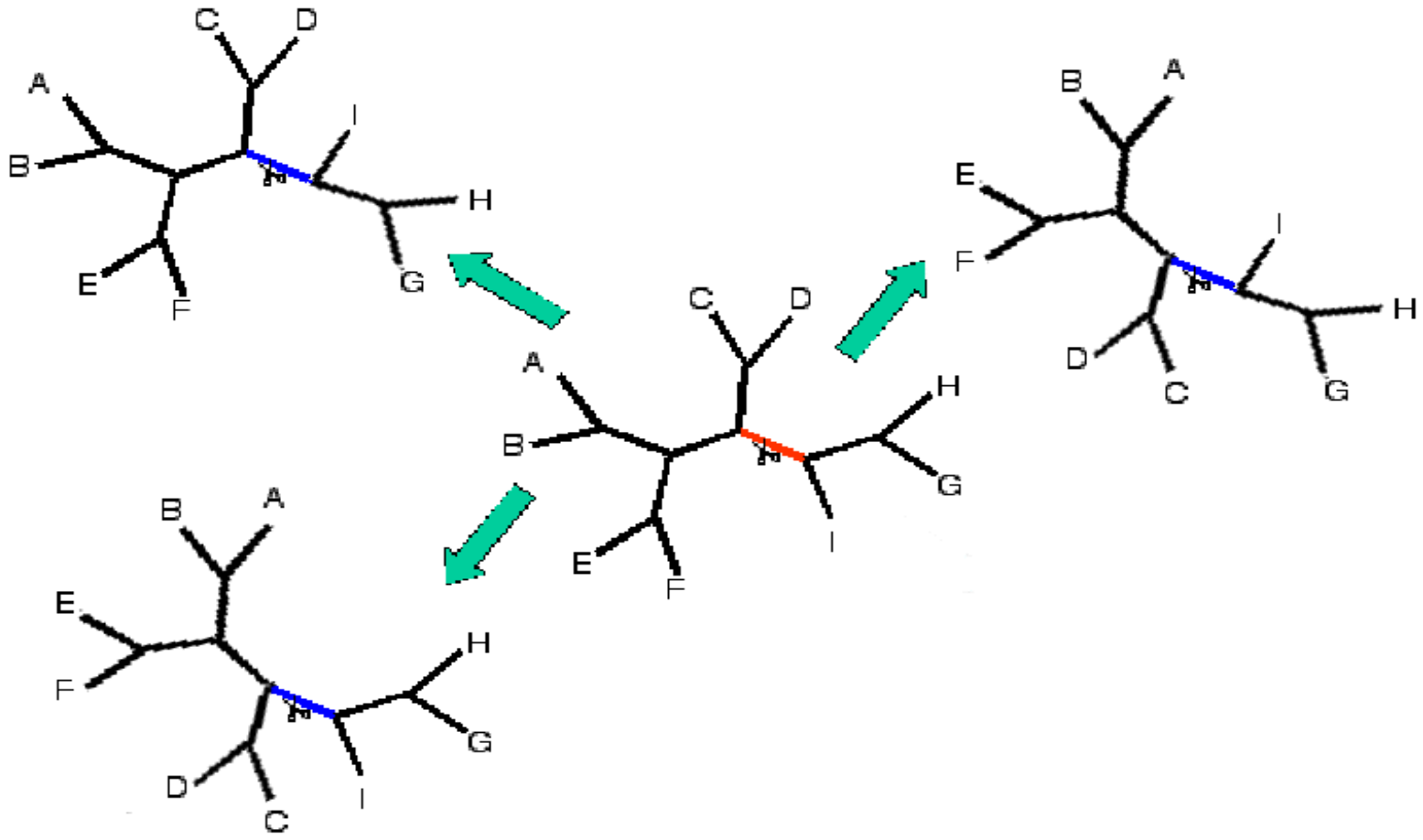
- Note that the structure of the subtrees beneath A, B, C, and D is unimportant as long as they are preserved.



Nearest Neighbor Interchange: Outline

- Start with an arbitrary tree and check its neighbors.
- Move to a neighbor if it provides the best improvement in parsimony score.
- Note that there is no way of knowing if the result is the *most* parsimonious tree.
- Furthermore, we could get stuck in a local optimum.

Nearest Neighbor Interchange: Example

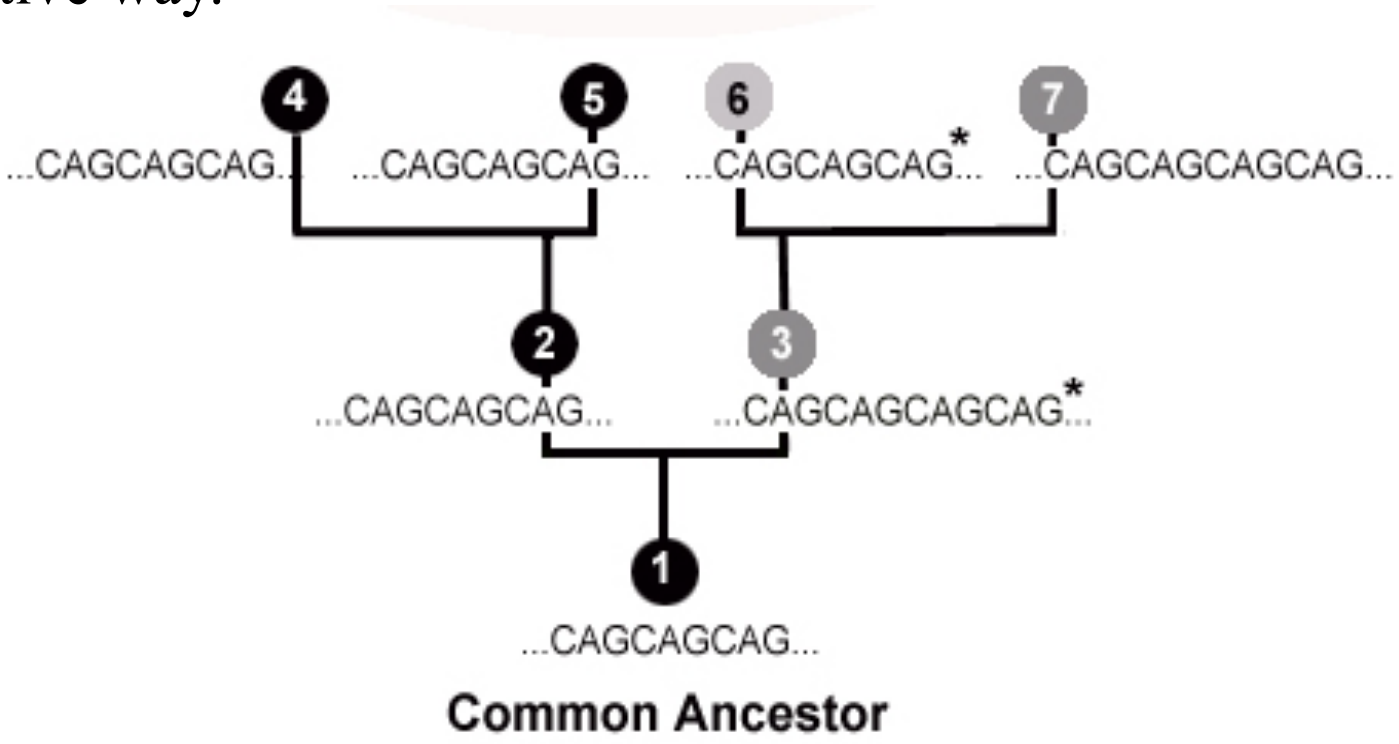


Homoplasy

- Given:
  1. CAGCAGCAG
  2. CAGCAGCAG
  3. CAGCAGCAGCAG
  4. CAGCAGCAG
  5. CAGCAGCAG
  6. CAGCAGCAG
  7. CAGCAGCAGCAG
- Most would group 1, 2, 4, 5, and 6 as having evolved from a common ancestor, with a single mutation leading to the presence of 3 and 7.

Homoplasy

- But what if this were the real tree?
- Parsimony would actually group these species in a highly nonintuitive way.



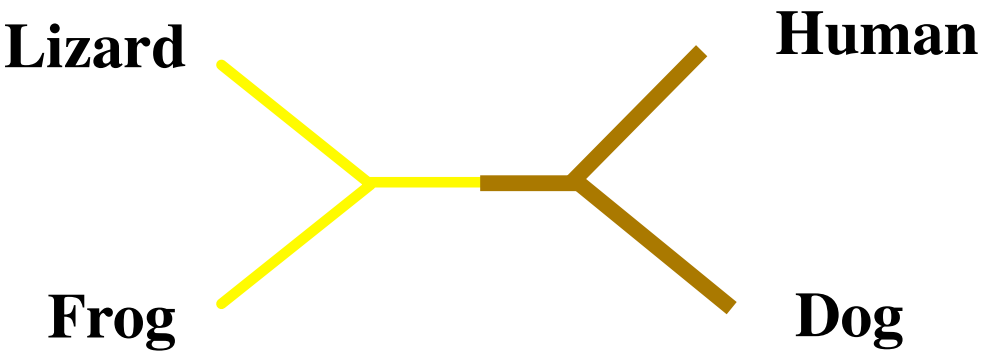
Homoplasy

- **Homoplasy**: Independent (or parallel) evolution of same/ similar characters.

- Parsimony seeks to *minimizes* homoplasy, so if homoplasy is common, parsimony may give incorrect results, as our example seemed to indicate.

Contradicting Characters

- An evolutionary tree is more likely to be correct when it is supported by multiple characters, as seen below.



MAMMALIA

Hair
Single bone in lower jaw
Lactation
etc.

- Note that in this case, tails are homoplasic. However, there is much more evidence that humans and dogs are closer than dogs and lizards…it makes more sense that humans lost tails.

Problems with Parsimony

- It is important to keep in mind that reliance on only one method for phylogenetic analysis provides an incomplete evolutionary picture.

- When different methods (parsimony, distance-based, etc.) all give the same result, it becomes much more likely that the result obtained is in fact correct.

Section 14: Evolution of Human Repeats

Alu Repeats

- **Alu repeats** (about 300 bp long) comprise the most common repeats in human genome (about 300 bp long).

- About 1 million Alu elements make up 10% of the human genome.

- Alu repeats are **retrotransposons**: They don't code for protein but copy themselves into RNA and then back to DNA via reverse transcriptase.

- Alu elements have been called "selfish" because their only function seems to be to make more copies of themselves.

What Makes Alu Elements Important?

- Alu elements began to replicate 60 million years ago. Therefore, their evolution can be used as a fossil record of primate and human history.

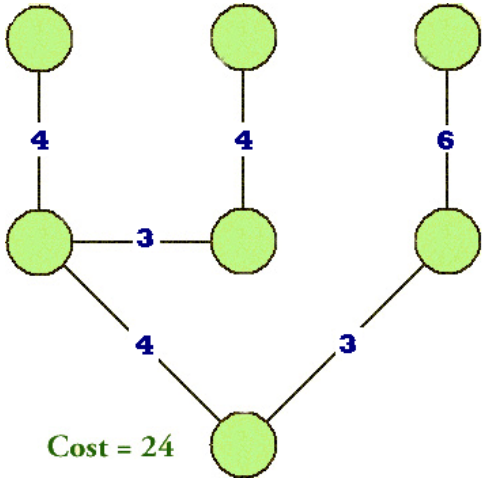
- Alu insertions are sometimes disruptive and can result in genetic disorders. Alu mediated recombination was linked to cancer.

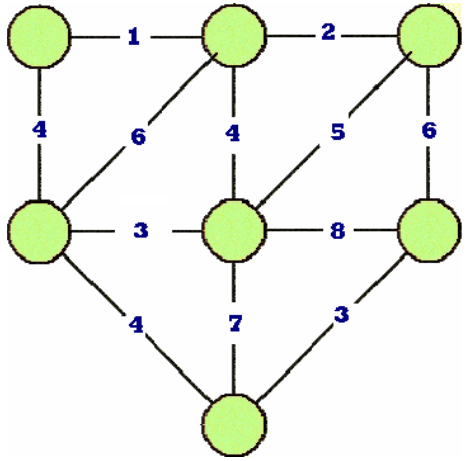
- Alu insertions can be used to determine genetic distances between human populations and human migratory history.

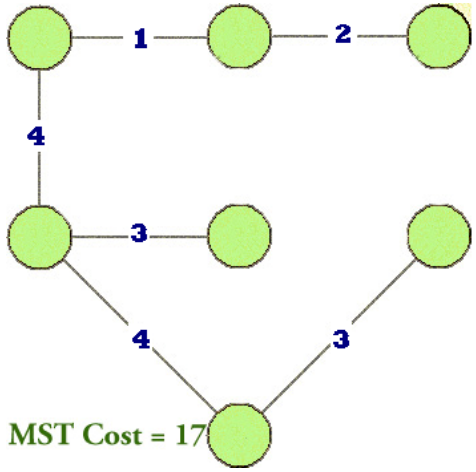
Section 15: Minimum Spanning Trees

What Is a Minimum Spanning Tree?

- A **Minimum Spanning Tree** of a graph:
  - Connects all the vertices in the graph into a tree *and*
  - Minimizes the sum of the edge weights in the tree among all spanning edges.

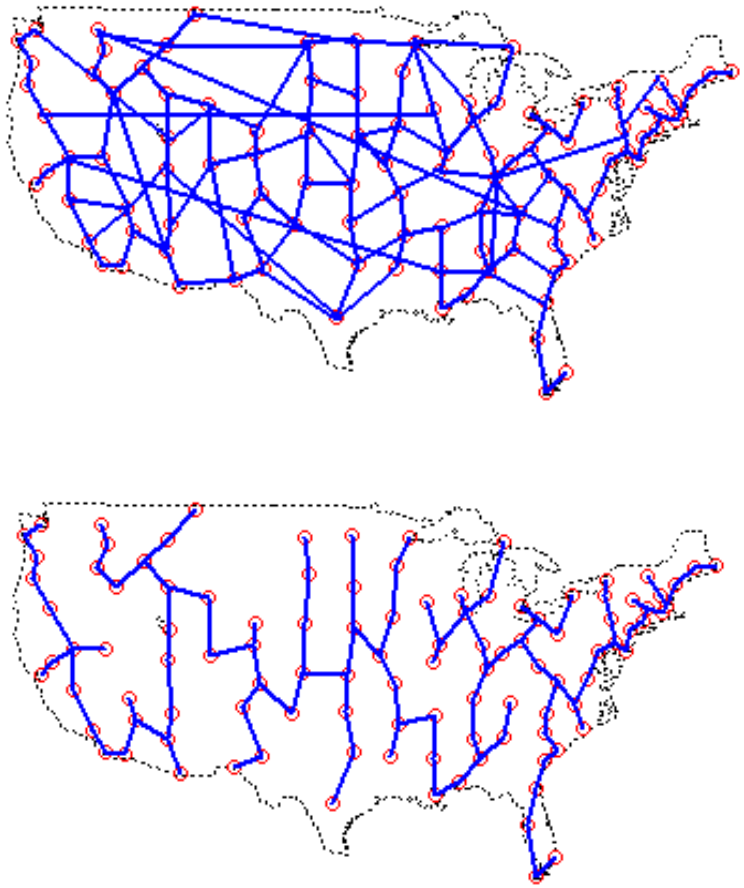






Minimum Spanning Trees

- The first algorithm for finding a MST was developed in 1926 by Otakar Borůvka.
  - Its purpose was to minimize the cost of electrical coverage in Bohemia by connecting all cities using as little electrical wire as possible to cut costs.
- We will use MST to study evolution of Alu repeats.

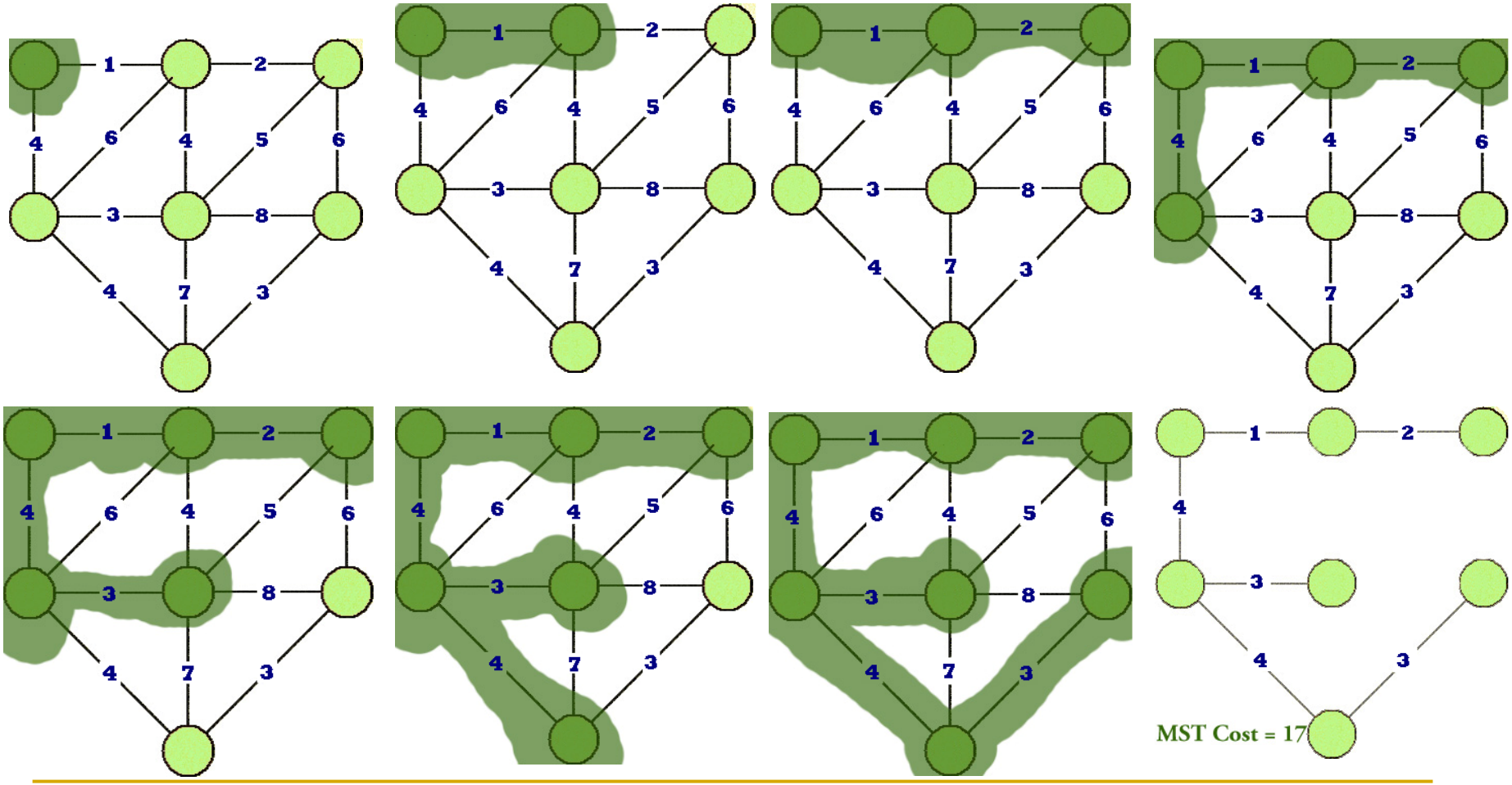


How Can We Find an MST?

- **Prim's Algorithm** (greedy):
  - Start from a tree *T* with a single vertex.
  - At each step, add the shortest edge connecting a vertex in *T* to a vertex not in *T*, growing the tree *T*.
    - This is repeated until every vertex is in *T*.

- Prim algorithm gives an absolute minimum and can be implemented in O(*m* log *m*) time (where *m* = # of edges).

Prim's Algorithm: Example

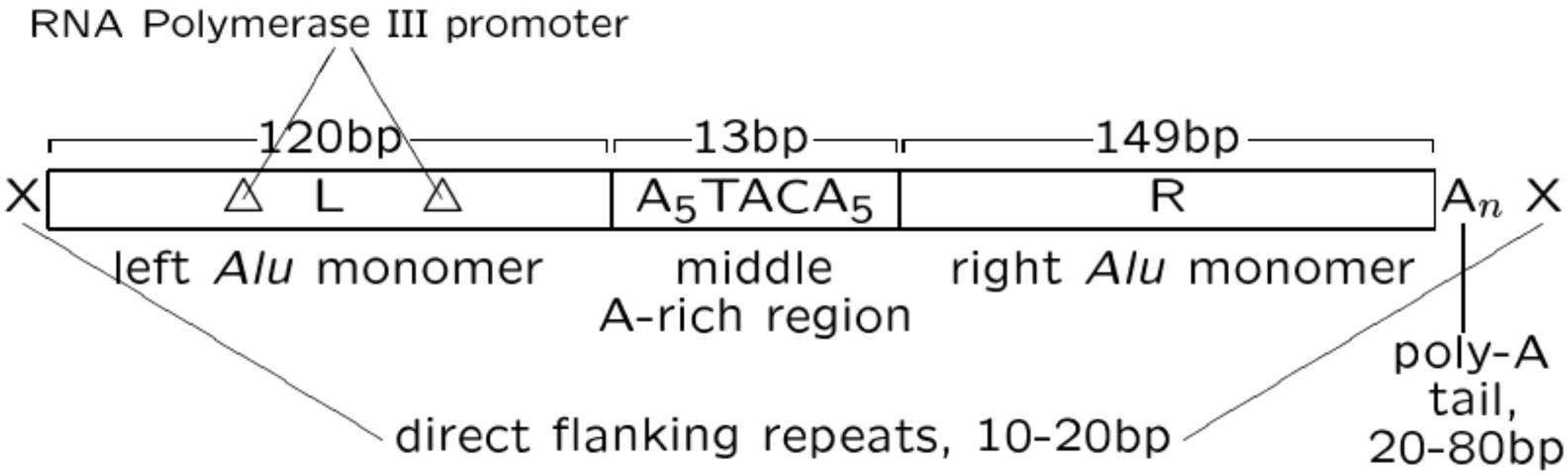


Why Prim Must Construct an MST

- Proof:
  - Let *e* be any edge that Prim algorithm chose to connect two sets of nodes. Suppose that Prim's algorithm is flawed and it is cheaper to connect the two sets of nodes via some other edge *f*.
  - Notice that since Prim algorithm selected edge e we know that cost(*e*) < cost(*f*)
  - By connecting the two sets via edge f, the cost of connecting the two vertices has gone up by exactly cost(*f*) – cost(*e*).
  - Therefore, we have a contradiction: edge *e* does not belong in the MST yet it cannot be formed without using edge *e*.

An Alu Element

- SINEs are flanked by short direct repeat sequences and are transcribed by RNA Polymerase III.



Alu Subfamilies

- We illustrate Alu subfamilies with a 40 bp sample segment of AluJb, AluSx, AluY and AluYa5 subfamily consensus sequences:

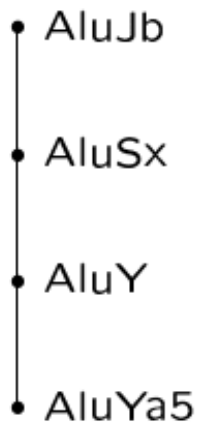
```
AluJb    ...G.........................A.............-..
AluSx    TGGCCAACATGGTGAAACCCCGTCTCTACTAAAAATACAAAAA-TT
AluY     ....T....C.....................................A..
AluYa5   C...T..A.C.....................................A..
```

- Early analyses identified 4 to 6 Alu subfamilies.

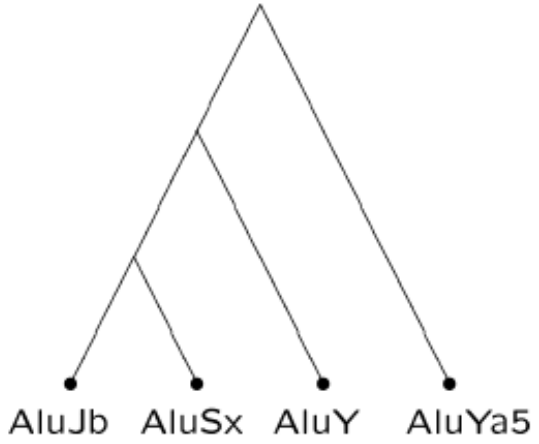
The Biological Story: Alu Evolution

- What do these Alu subfamilies tell us about Alu evolution?

```
AluJb    ...G.........................A.............-..
AluSx    TGGCCAACATGGTGAAACCCCGTCTCTACTAAAAATACAAAAA-TT
AluY     ....T....C.....................................A..
AluYa5   C...T..A.C.....................................A..
```



Evolutionary Tree

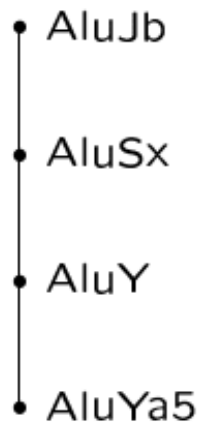


Phylogenetic Tree

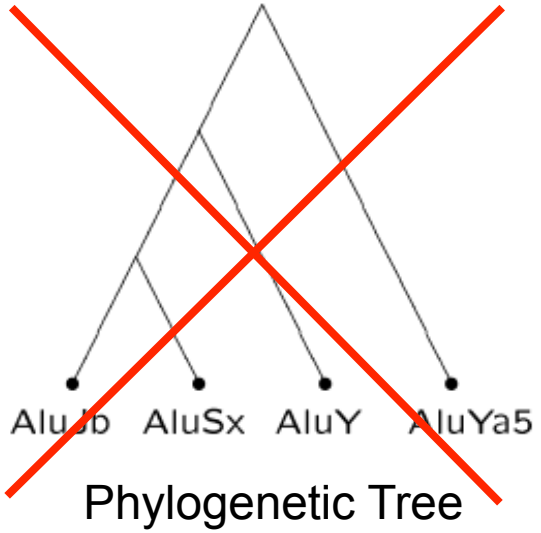
The Biological Story: Alu Evolution

- What do Alu subfamilies tell us about Alu evolution?

```
AluJb    ...G.........................A.............-..
AluSx    TGGCCAACATGGTGAAACCCCGTCTCTACTAAAAATACAAAAA-TT
AluY     ....T....C.....................................A..
AluYa5   C...T..A.C.....................................A..
```



Evolutionary Tree



Phylogenetic Tree

Alu Evolution: The Master Alu Theory

- Shen et al. (1991) conjectured that all Alu repeat elements have retroposed from a "single master gene."

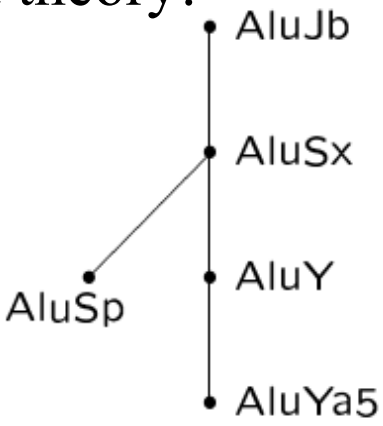
- **Conjecture**: 4 subfamilies implies linear evolution of 1 master gene.

Evolutionary Tree Genome

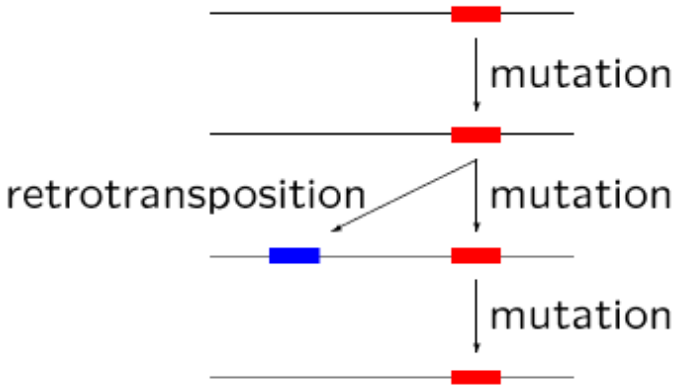
Alu Evolution: Alu Master Theory Proven Wrong

- Jurka and Milosavljevic (1991) identified additional subfamilies which do not fit the linear pattern of evolution.

- The AluSp and AluY subfamily lineages must have been produced by distinct master genes—this disproves the master Alu theory!



Evolutionary Tree

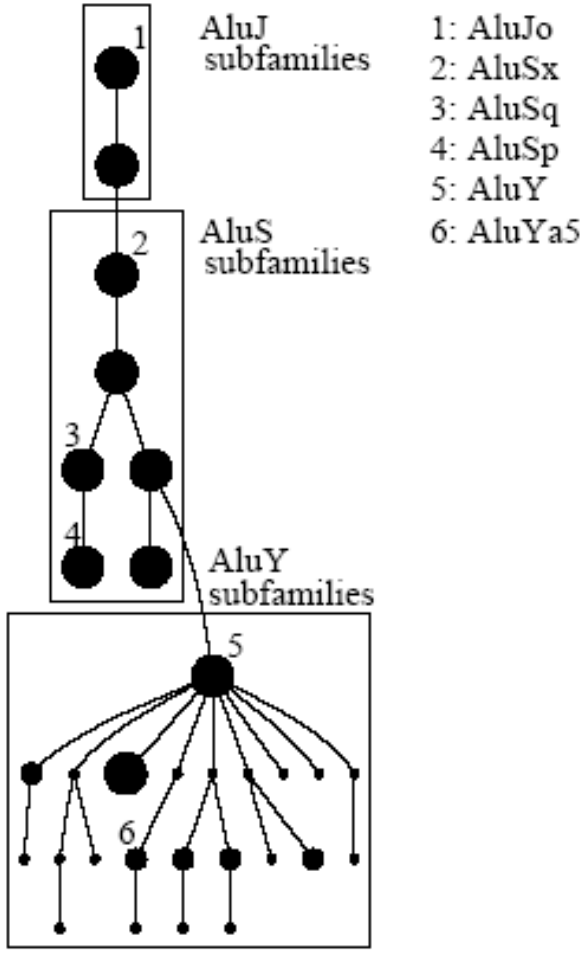


Genome

MST As An Evolutionary Tree

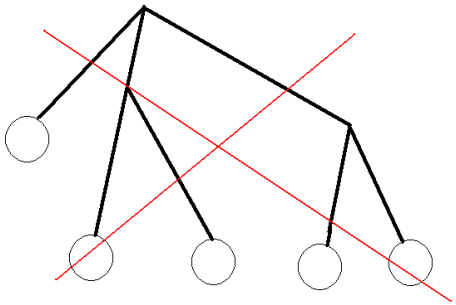
- At right: The evolutionary tree of the 31 Repbase Update subfamilies, defined as their Minimum Spanning Tree (Kruskal 1956).

- 14 leaves in this tree → at least 14 Alu source elements.

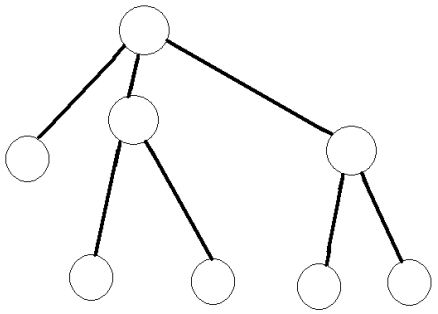


Alu Evolution: MST vs. Phylogenetic Tree

- A timeline of Alu subfamily evolution would be useful.
  - **Problem**: building a traditional phylogenetic tree with Alu subfamilies will not describe Alu evolution accurately.



- Why?
  - When constructing a typical phylogenetic tree, the input is made up of only leaf nodes.
  - Alu subfamilies may be *either* internal or external nodes of the evolutionary tree.
  - This is because Alu subfamilies that created new Alu subfamilies are still present.



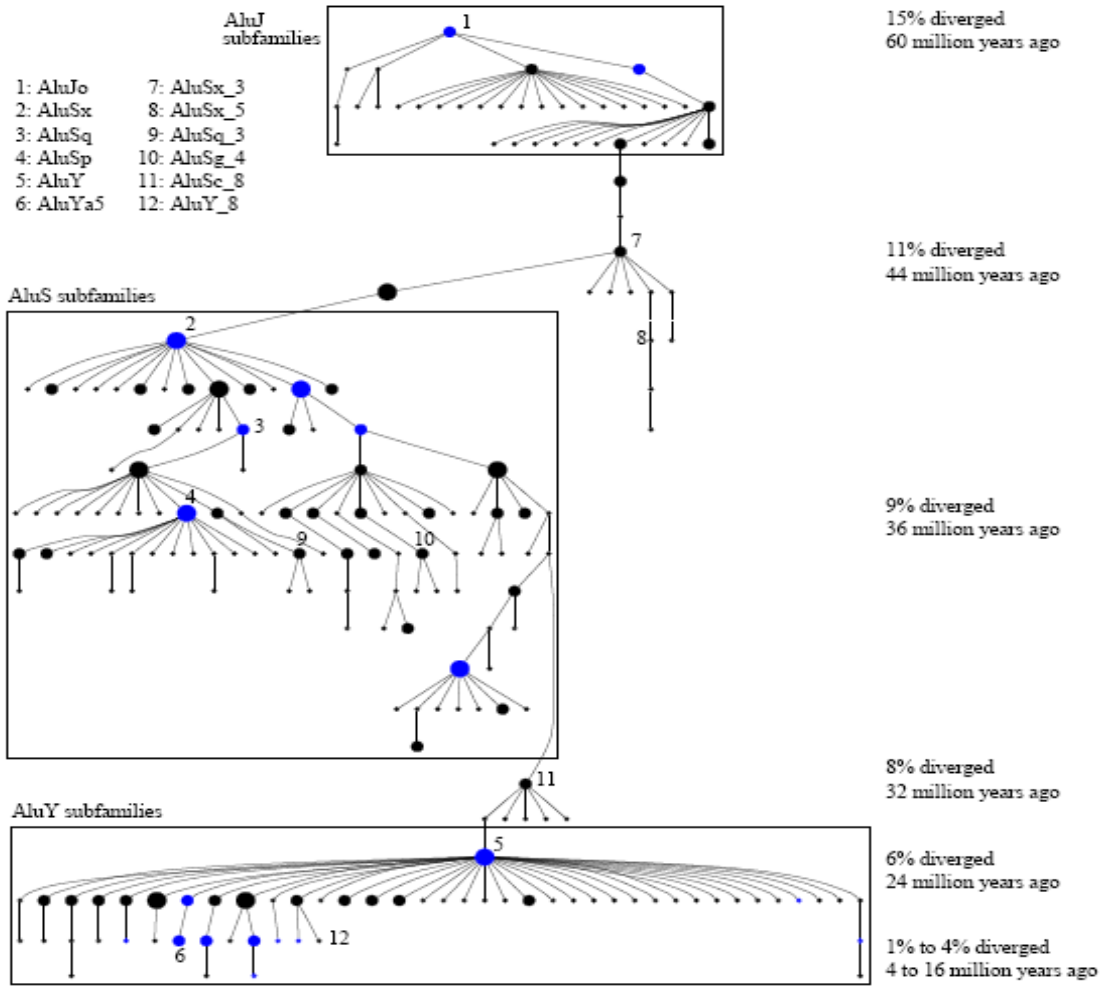
Constructing MST for Alu Evolution

- Building an evolutionary tree using an MST:
  - Define the *length* between two subfamilies as the Hamming distance between their sequences.
  - Place the subfamily with highest average divergence from its consensus sequence (the oldest subfamily) as the root.
  - It takes ~4 million years for 1% of sequence divergence between subfamilies to emerge.
  - This allows for the creation of a timeline of Alu evolution.

Constructing MST for Alu Evolution

- Why is the MST useful as an evolutionary tree?
  - The less the Hamming distance (edge weight) between two subfamilies, the more likely that they are directly related.
  - An MST represents a way for Alu subfamilies to have evolved minimizing the sum of all the edge weights (total Hamming distance between all Alu subfamilies), which makes it the most parsimonious way for the evolution of the subfamilies to have occurred.

MST As An Evolutionary Tree



References

- http://www.math.tau.ac.il/~rshamir/ge/02/scribes/lec01.pdf
- Serafim Batzoglou (UPGMA slides) http://www.stanford.edu/class/cs262/Slides