

# Phylogeny

Slides:

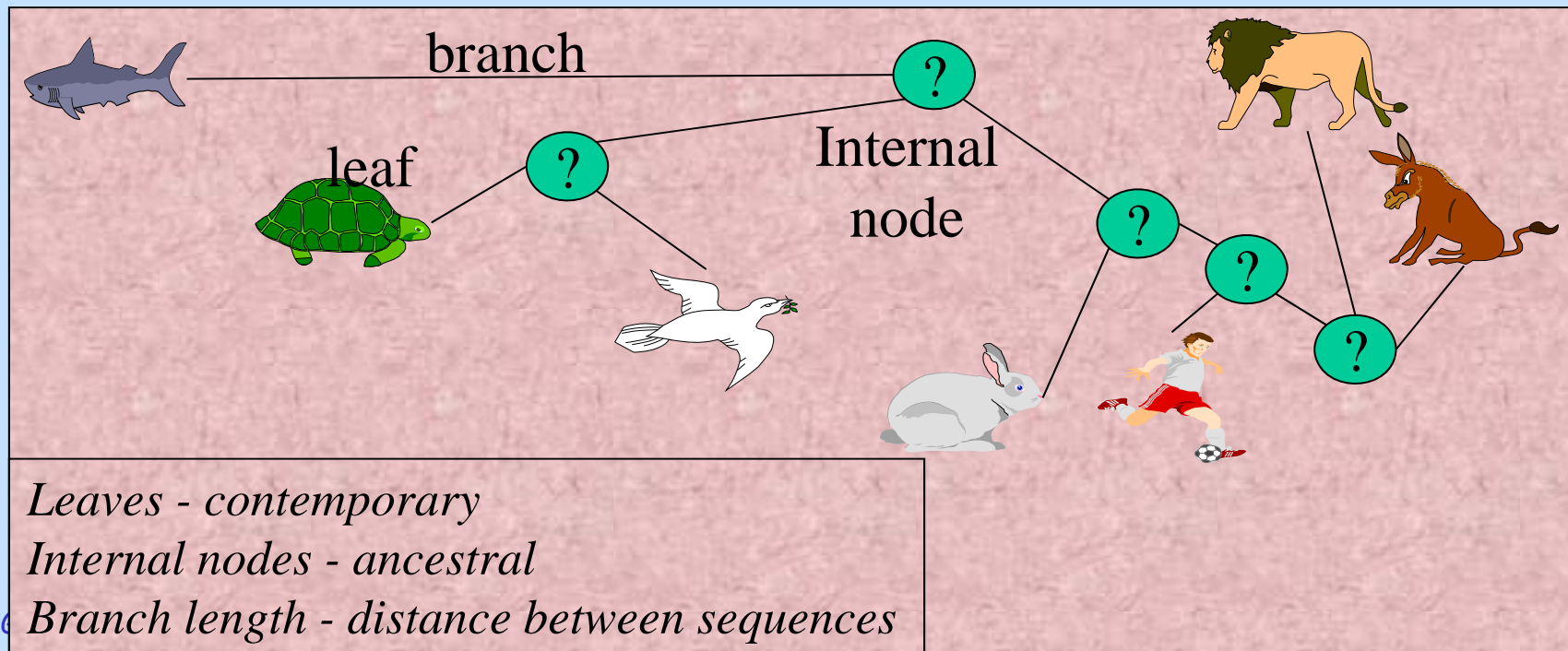
- Adi Akavia
- Nir Friedman's slides at HUJI (based on ALGMB 98)
- Anders Gorm Pedersen, Technical University of Denmark

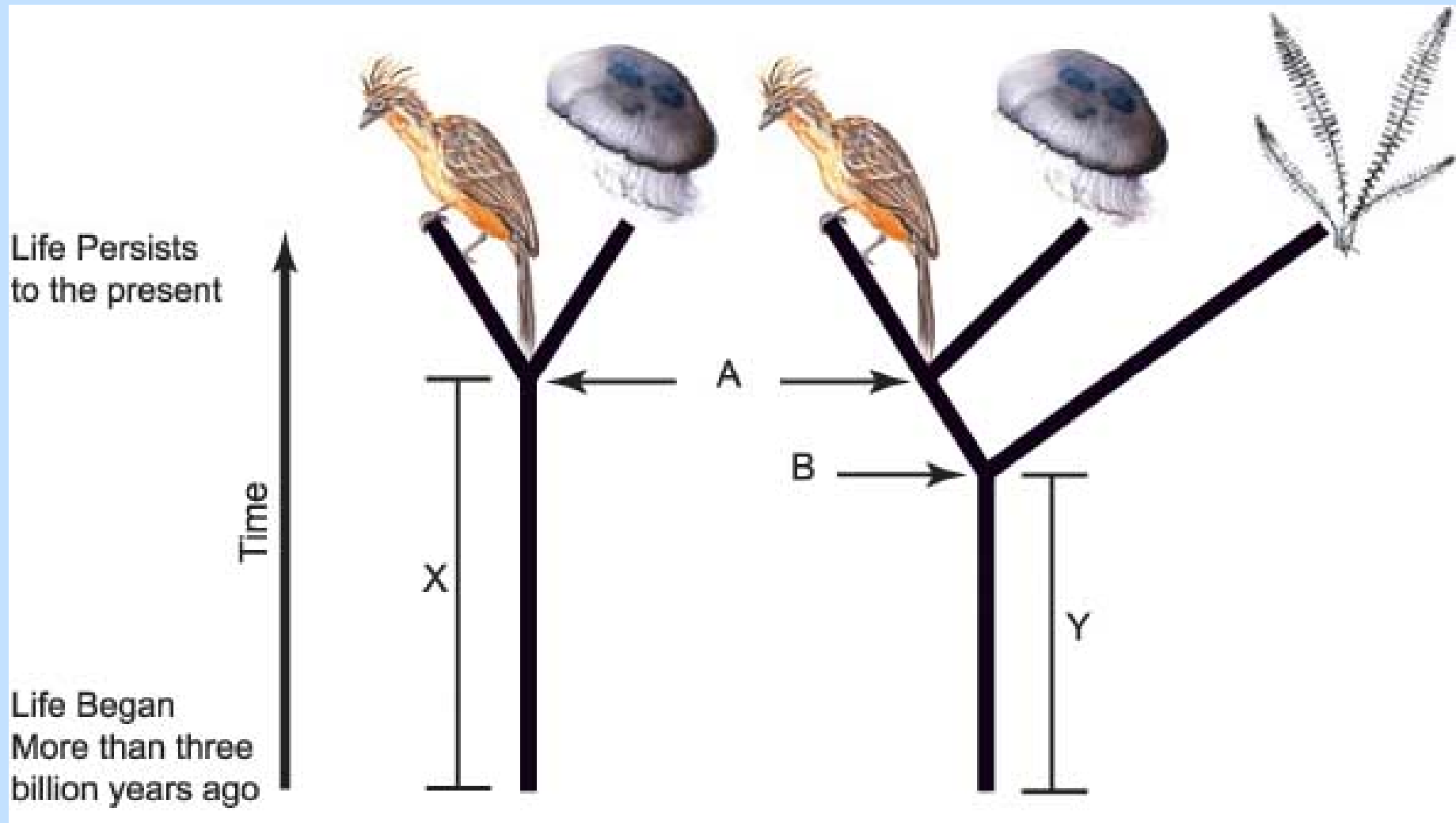
Sources: Joe Felsenstein "Inferring Phylogenies" (2004)

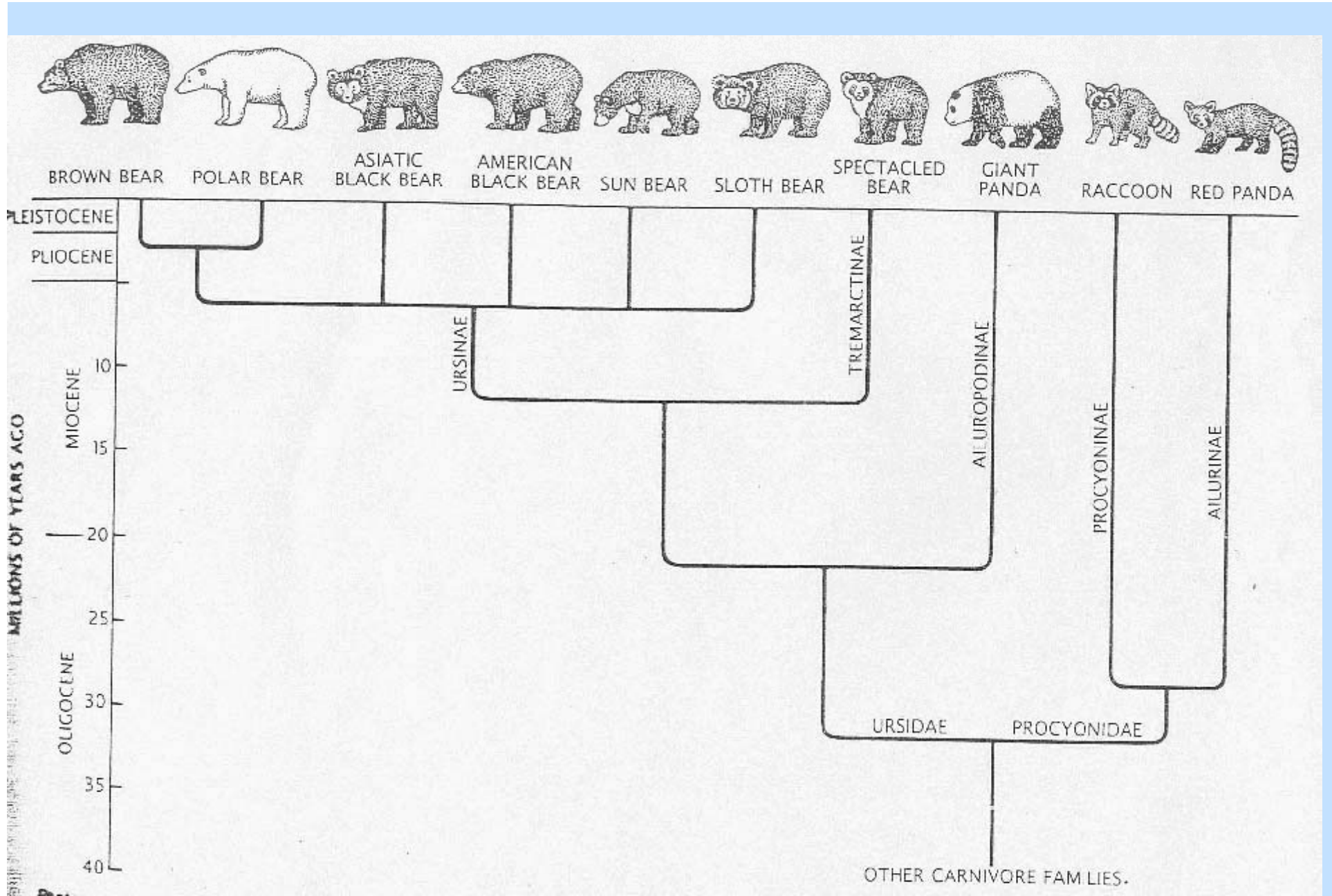


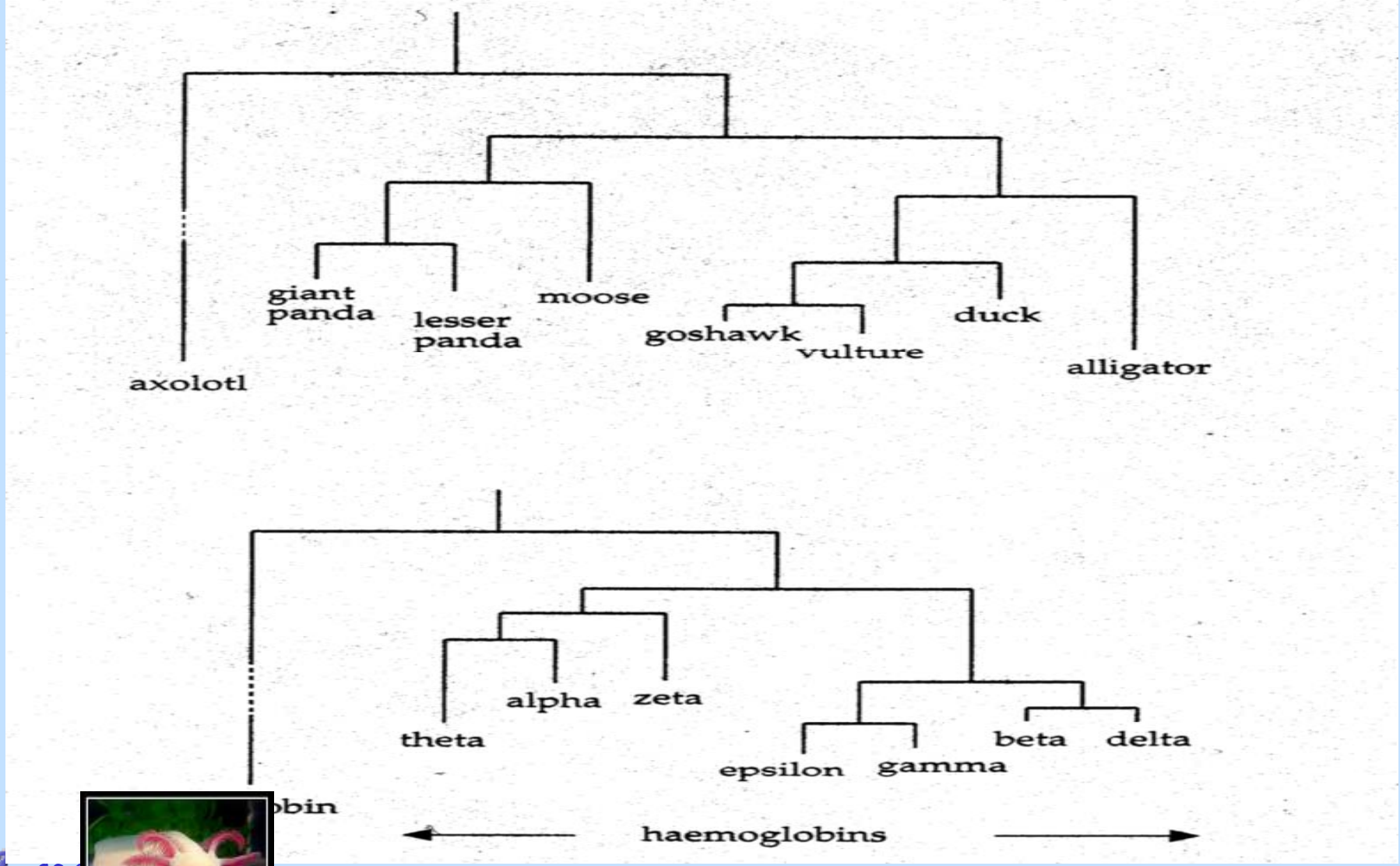
# Phylogeny

- **Phylogeny**: the ancestral relationship of a set of species.
- Represented by a **phylogenetic tree**



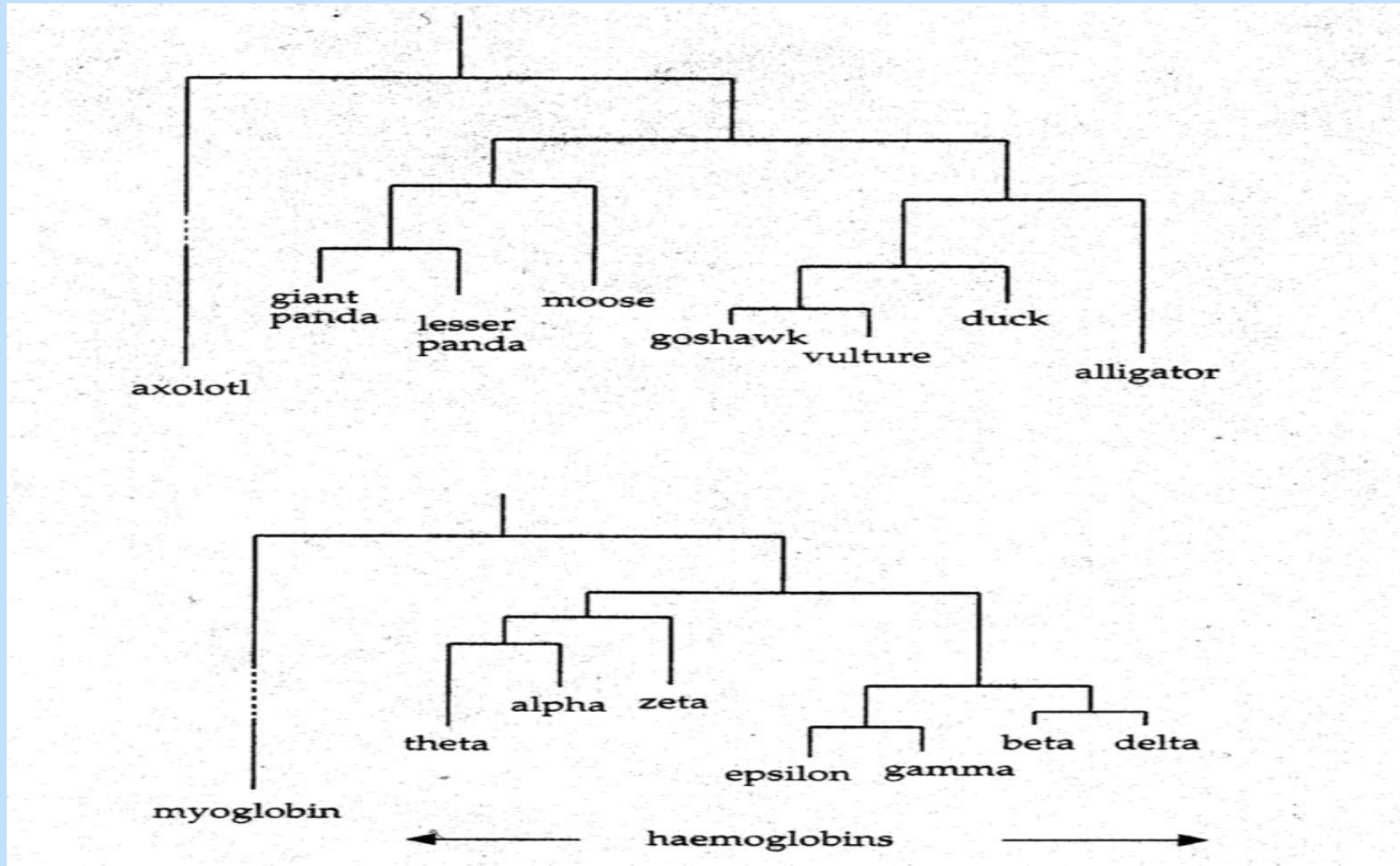






CGC



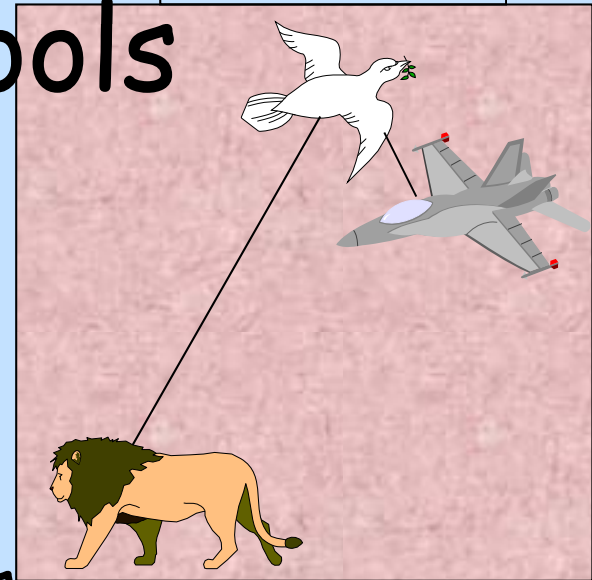


# Phylogeny schools

“classical”

## Classical vs. Modern:

- Classical - morphological characters
- Modern - molecular sequences.



## Difficulties:

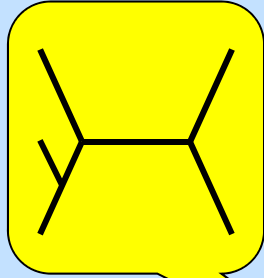
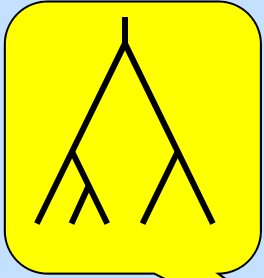
- Gene duplication
- Orthologs vs. paralogs

### Sequence Homology Caused By:

- **Orthologs** - speciation,
- **Paralogs** - duplication
- **Xenologs** - horizontal (e.g., by virus)



# Trees and Models



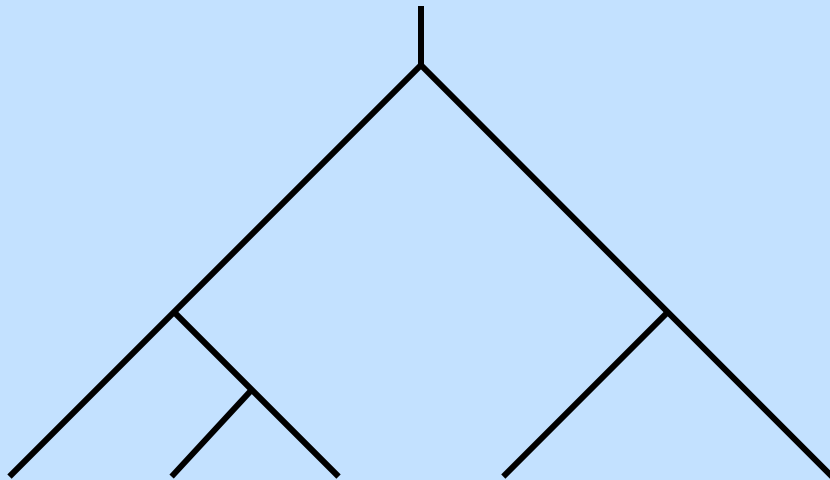
- rooted / unrooted
- topology / distance
- binary / general

“molecular  
clock”



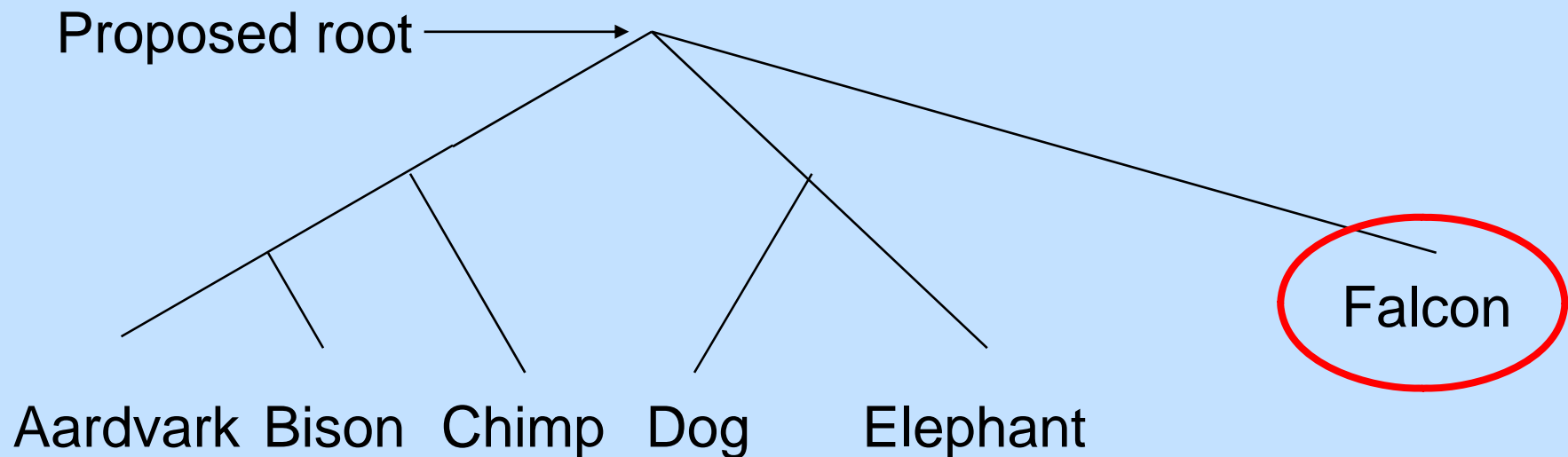
# To root or not to root?

- **Unrooted** tree represents phylogeny without the root node: no directionality



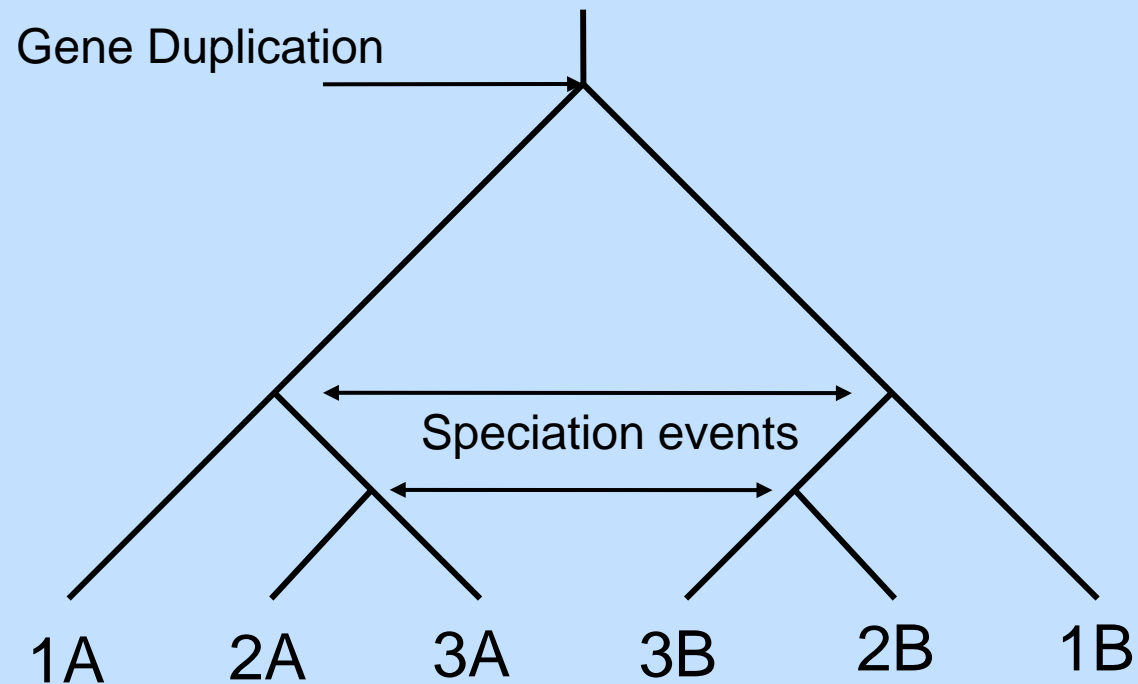
# Rooting an Unrooted Tree

- We can estimate the position of the root by introducing an **outgroup**:
  - a species that is definitely most distant from all the species of interest



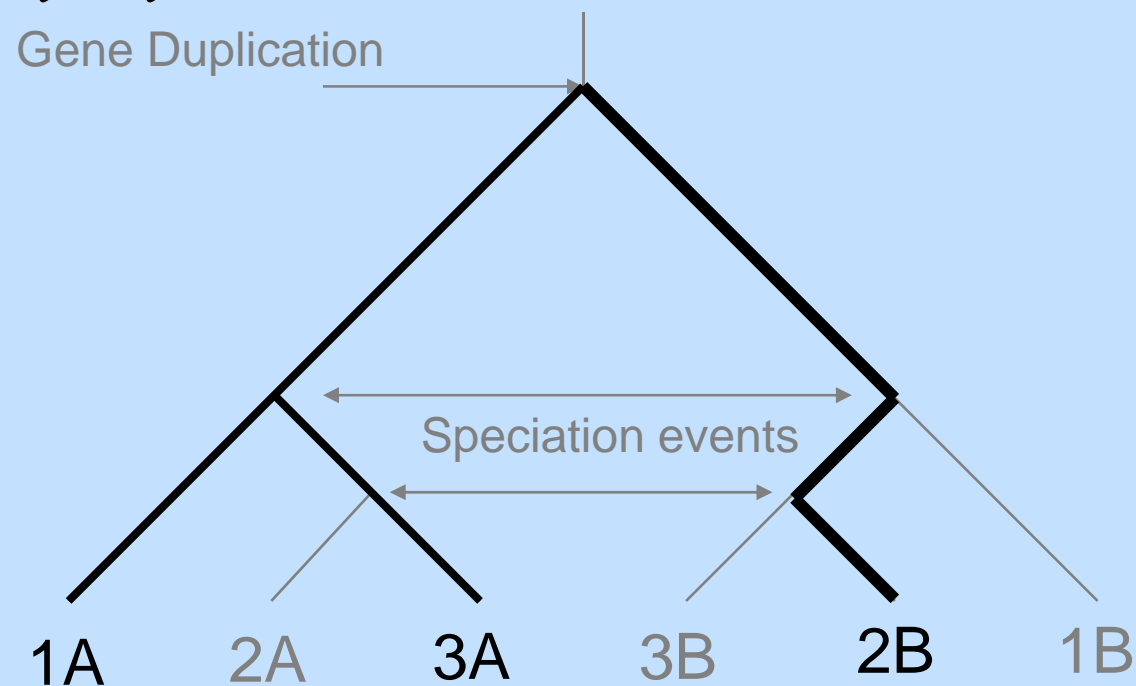
# Dangers of Paralogs

- Right species distance:  $(1,(2,3))$



# Dangers of Paralogs

- Right species distance:  $(1,(2,3))$
- If we only consider 1A, 2B, and 3A:  $((1,3),2)$



# Type of Data

- **Distance-based**
  - Input: matrix of distances between species
  - Distance can be
    - fraction of residues they disagree on,
    - alignment score between them,
    - ...
- **Character-based**
  - Examine each character (e.g., residue) separately



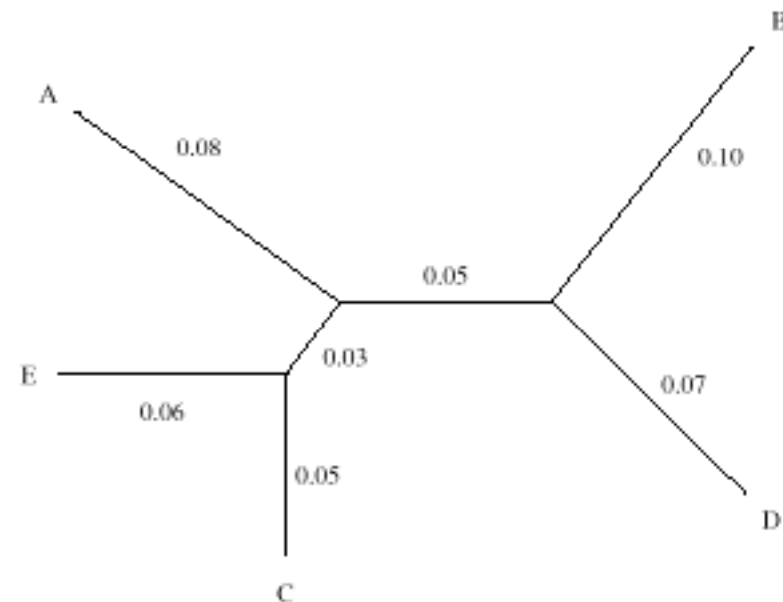
# Distance Based Methods



# Tree based distances

- $D(i,j)$ =sum of arc lengths on the path  $i \rightarrow j$

species	A	B	C	D	E
A	0	0.23	0.16	0.20	0.17
B	0.23	0	0.23	0.17	0.24
C	0.16	0.23	0	0.15	0.11
D	0.20	0.17	0.15	0	0.21
E	0.17	0.24	0.11	0.21	0



# The Problem

**Input:** matrix **d** of distances between species

**Goal:** Find a tree with leaves=chars and edge distances, matching d best.

**Quality measure:** sum of squares:

$$SSQ(T) = \sum_i \sum_{j \neq i} w_{ij} (d_{ij} - t_{ij})^2$$

$t_{ij}$ : distance in the tree

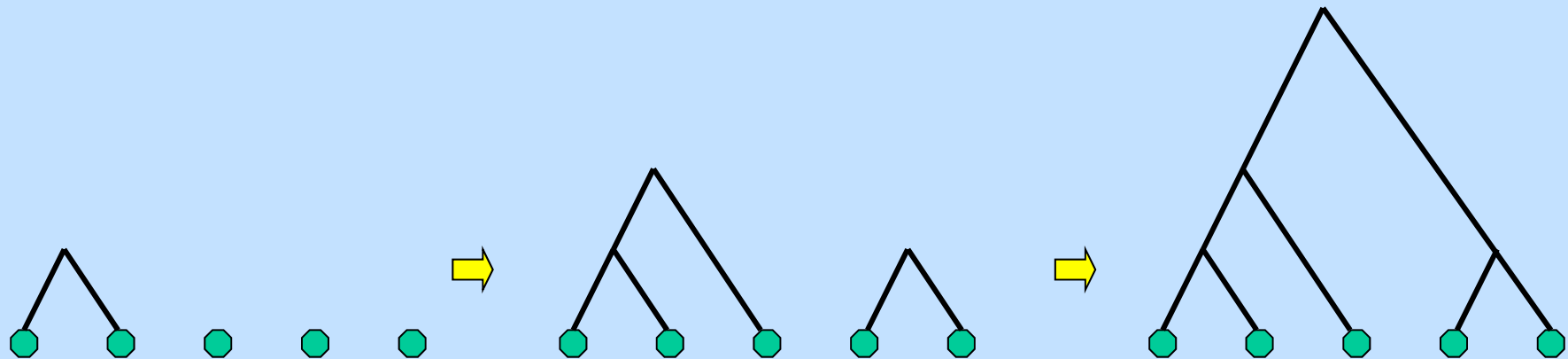
$w_{ij}$ : pair weighting. (1)  $\equiv 1$  (2)  $1/d_{ij}$  (3)  $1/d_{ij}^2$



# UPGMA Clustering (Sokal & Michener 1958)

(Unweighted pair-group method with arithmetic mean)

- Approach: Form a tree; closer species according to input distances should be closer in the tree
- Build the tree bottom up, each time merging two smaller trees
- All leaves are at same distance from the root



# Efficiency lemma

- Main idea: gradually form **clusters**: sets of species
- Repeatedly identify two clusters and merge them.
- For clusters  $C_i, C_j$ , define the distance between them to be the average dist betw their members:

$$d(C_i, C_j) = \frac{1}{|C_i| |C_j|} \sum_{p \in C_i} \sum_{q \in C_j} d(p, q)$$

- Lemma: If  $C_k$  is formed by merging  $C_i$  and  $C_j$  then for every other cluster  $C_l$

$$d(C_k, C_l) = (|C_i| * d(C_i, C_l) + |C_j| * d(C_j, C_l)) / (|C_i| + |C_j|)$$



→ Can efficiently update distances between clusters.

# UPGMA algorithm

**Initialize:** each node is a cluster  $C_i = \{i\}$ .  $d(C_i, C_j) = d(i, j)$  set  $\text{height}(i) = 0 \forall i$

**Iterate:**

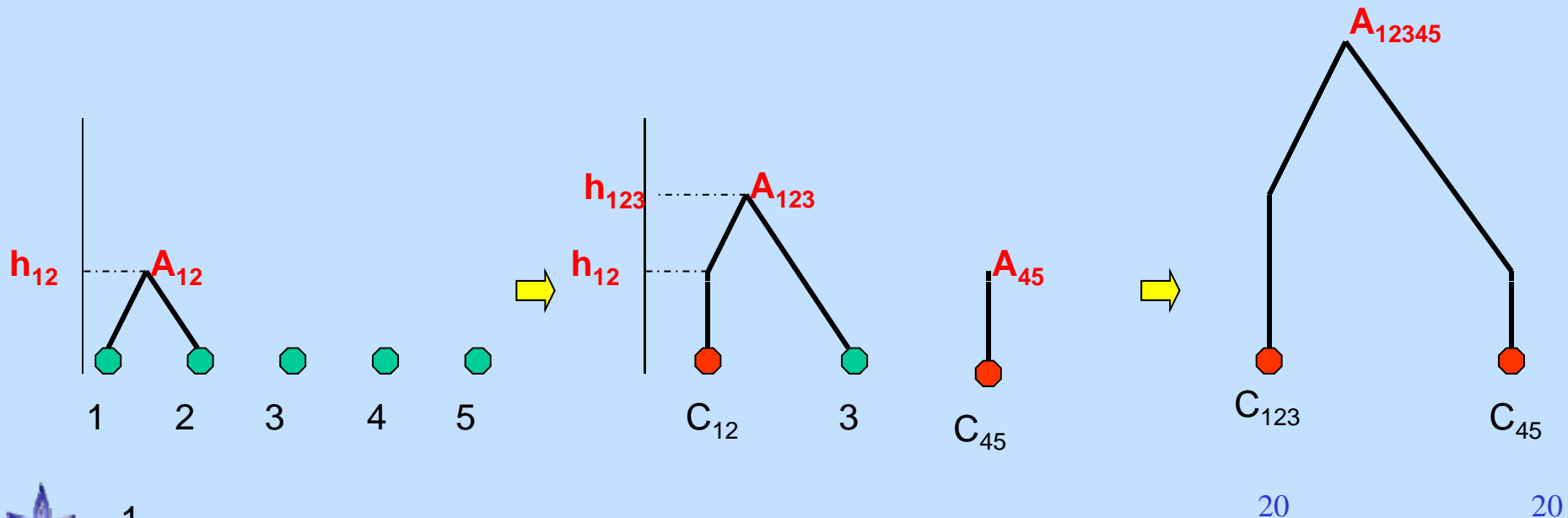
- Find  $C_i, C_j$  with smallest  $d(C_i, C_j)$
- Introduce a new cluster node  $C_k$  that replaces  $C_i$  and  $C_j$   
//  $C_k$  represents all the leaves in clusters  $C_i$  and  $C_j$
- Introduce a new tree node  $A_{ij}$  with  $\text{height}(A_{ij}) = d(C_i, C_j)/2$   
//  $d(C_i, C_j)$  is the average dist among leaves of  $C_i$  and  $C_j$
- Connect the corresponding tree nodes  $C_i, C_j$  to  $A_{ij}$  with  
 $\text{length}(C_i, A_{ij}) = \text{height}(A_{ij}) - \text{height}(C_i)$   
 $\text{length}(C_j, A_{ij}) = \text{height}(A_{ij}) - \text{height}(C_j)$
- For all other  $C_l$ :  
 $d(C_k, C_l) = (|C_i| * d(C_i, C_l) + |C_j| * d(C_j, C_l)) / (|C_i| + |C_j|)$   
// dist to any old cluster is the ave dist between its leaves and leaves in  $C_i, C_j$

Time: Naïve:  $O(n^3)$ ; Can show  $O(n^2 \log n)$  (ex.) and  $O(n^2)$  (harder ex.)

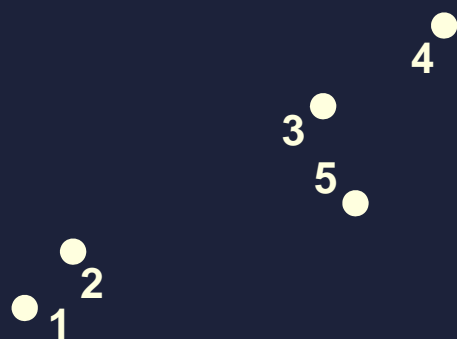


# UPGMA alg (2)

- A nodes represent the groups of nodes that they replaced, and maintain the average dist of the set from other leaf nodes/clusters



# UPGMA Algorithm



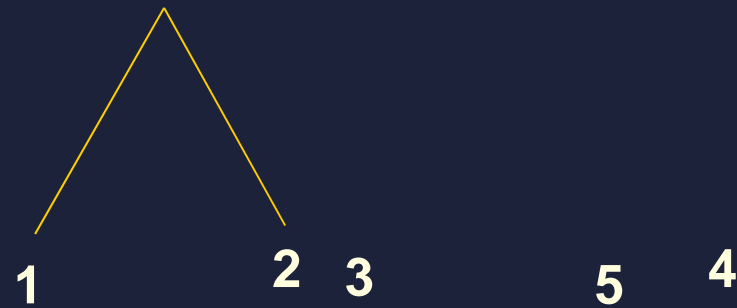
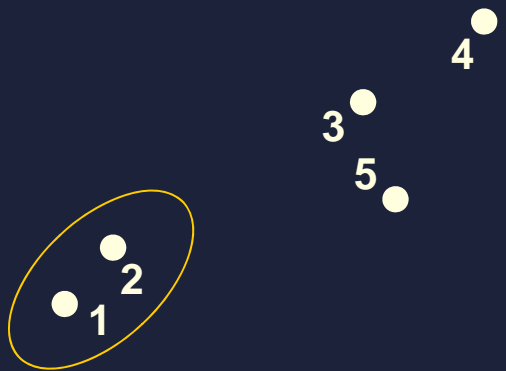
1

2 3

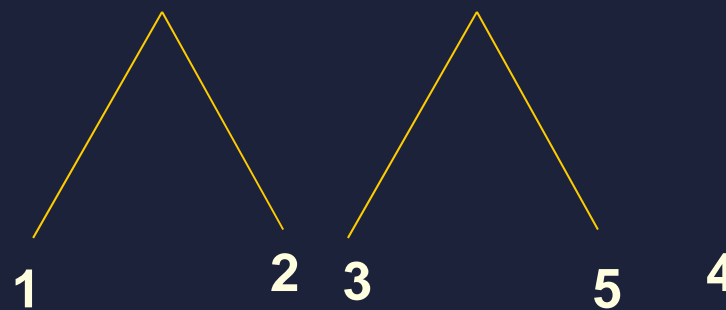
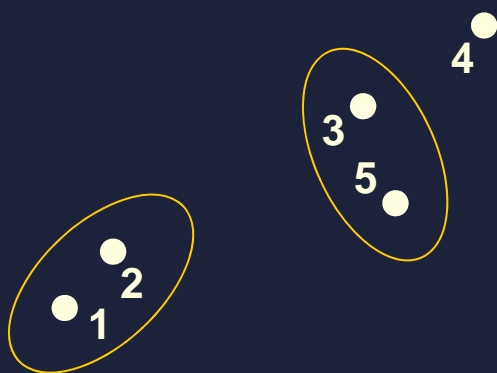
5

4

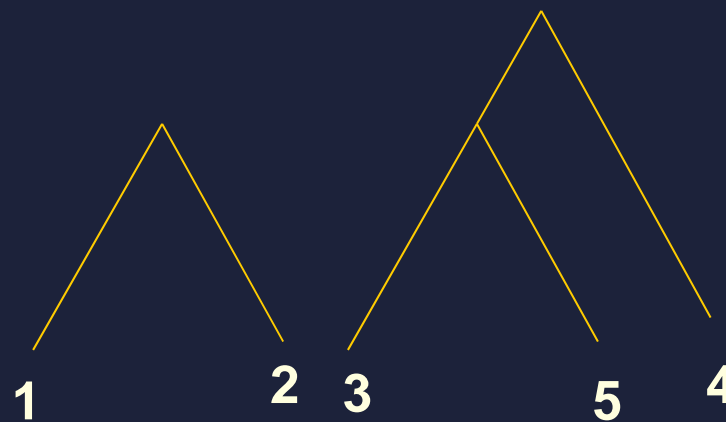
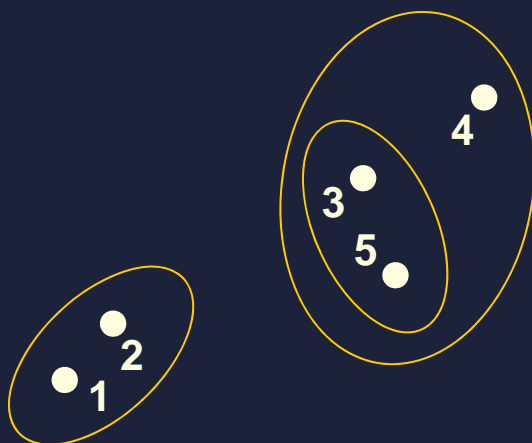
# UPGMA Algorithm



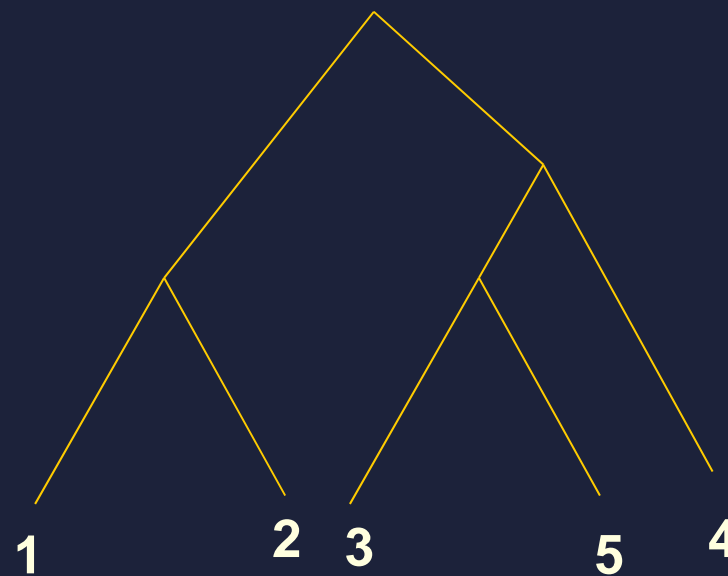
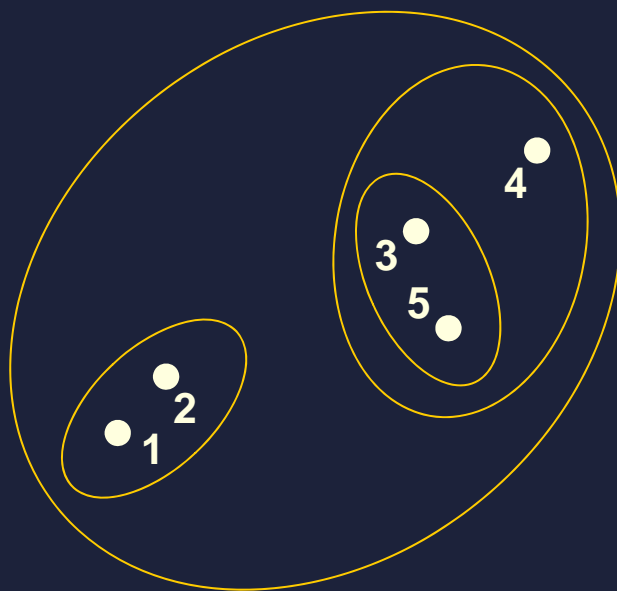
# UPGMA Algorithm



# UPGMA Algorithm

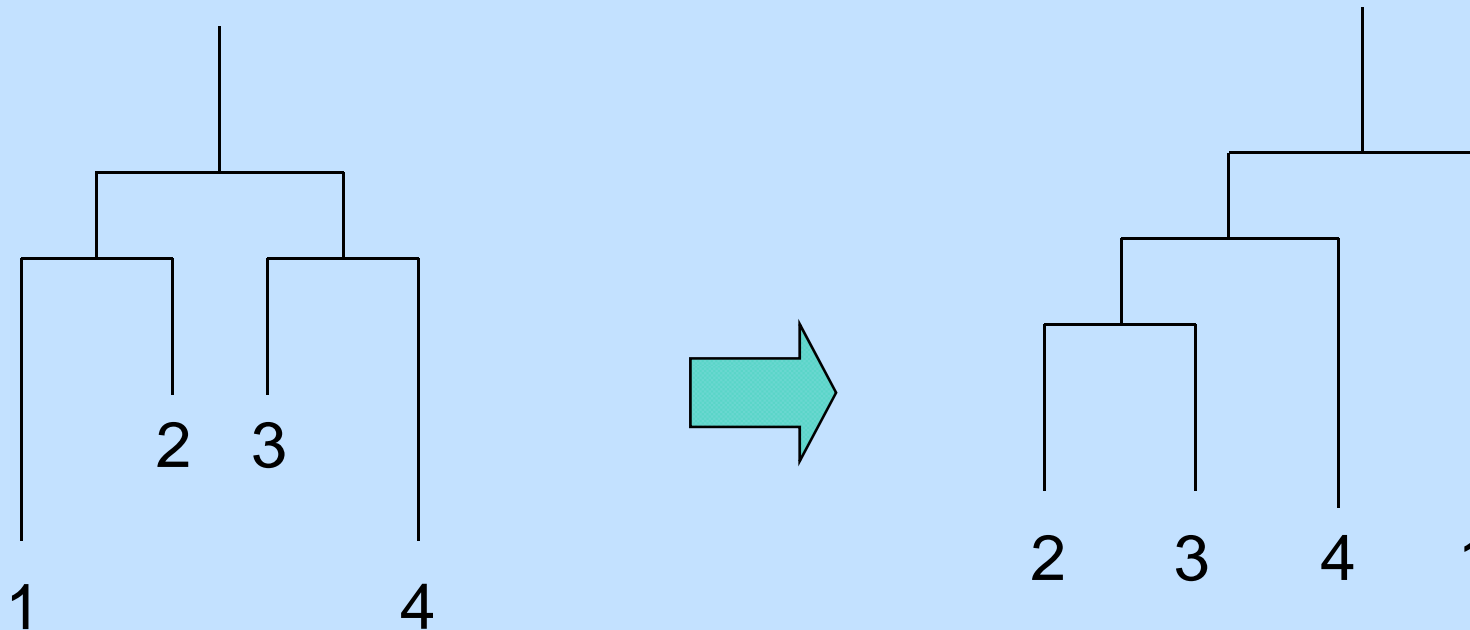


# UPGMA Algorithm



# Molecular Clock

- UPGMA implicitly assumes the tree is **ultrametric**: equal leaf-root distances => common uniform clock

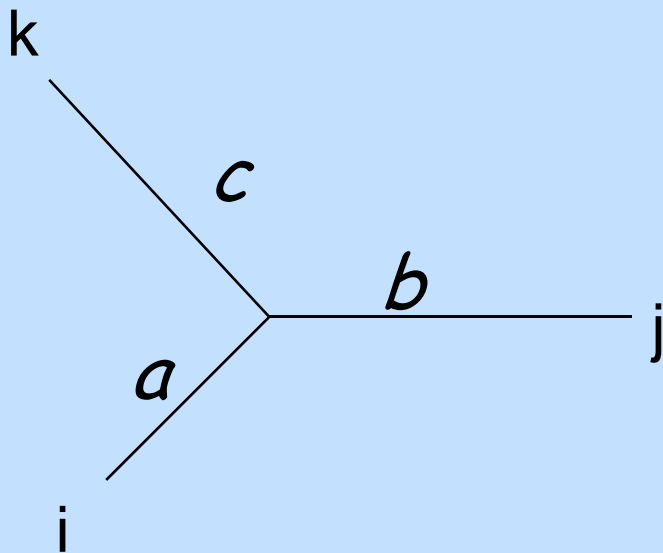


- Works reasonably well for nearby species



# Additivity

- A weaker additivity assumption:
  - In "real" trees, distances between species are the sum of distances between intermediate nodes



$$d(i, j) = a + b$$

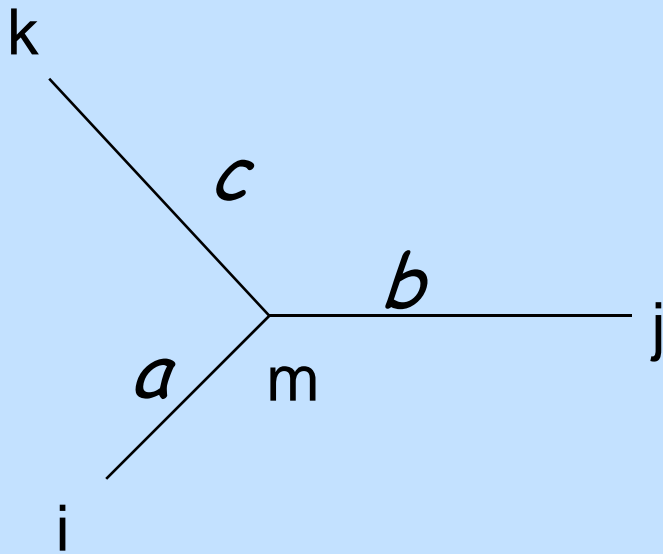
$$d(i, k) = a + c$$

$$d(j, k) = b + c$$



# Consequences of Additivity

- Suppose input distances are additive
- For any three leaves



$$d(i, j) = a + b$$

$$d(i, k) = a + c$$

$$d(j, k) = b + c$$

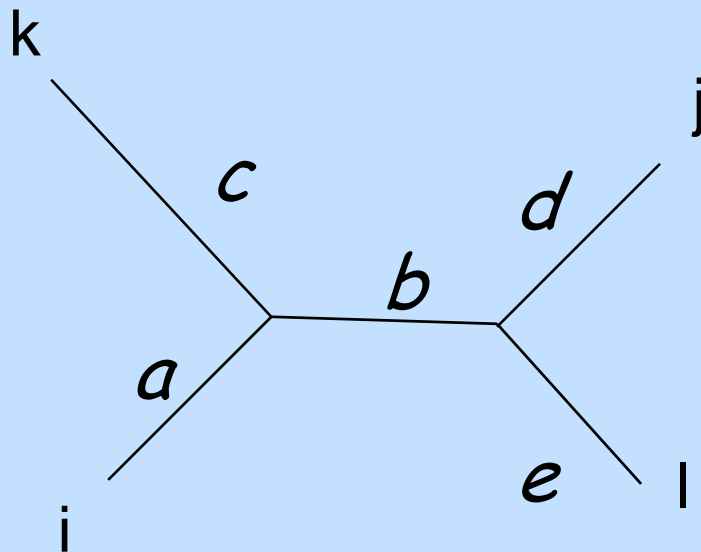
- Thus

$$d(m, k) = \frac{1}{2}(d(i, k) + d(j, k) - d(i, j))$$



# Consequences of Additivity II

- Suppose input distances are additive
- For any four leaves
- Induced tree wt:  $W$



$$d(i, j) + d(k, l) = W + b$$

$$d(i, l) + d(j, k) = W + b$$

$$d(i, k) + d(j, l) = W - b$$

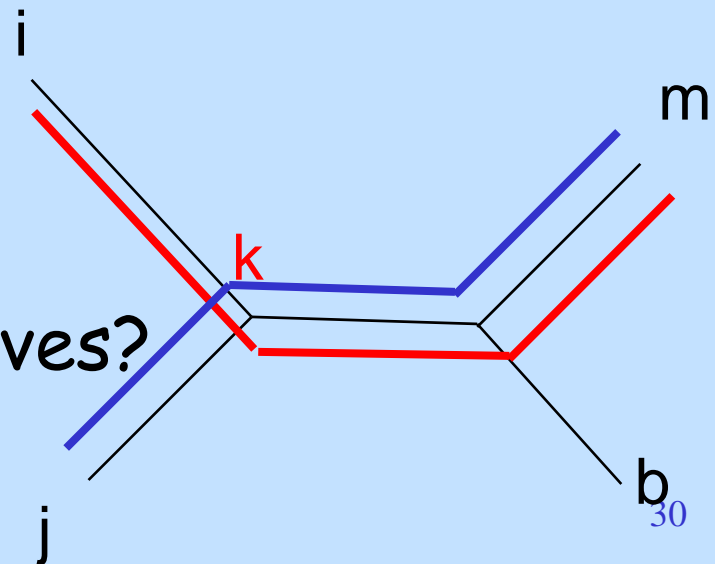
- Largest sum must appear **twice**



# Consequences of Additivity III

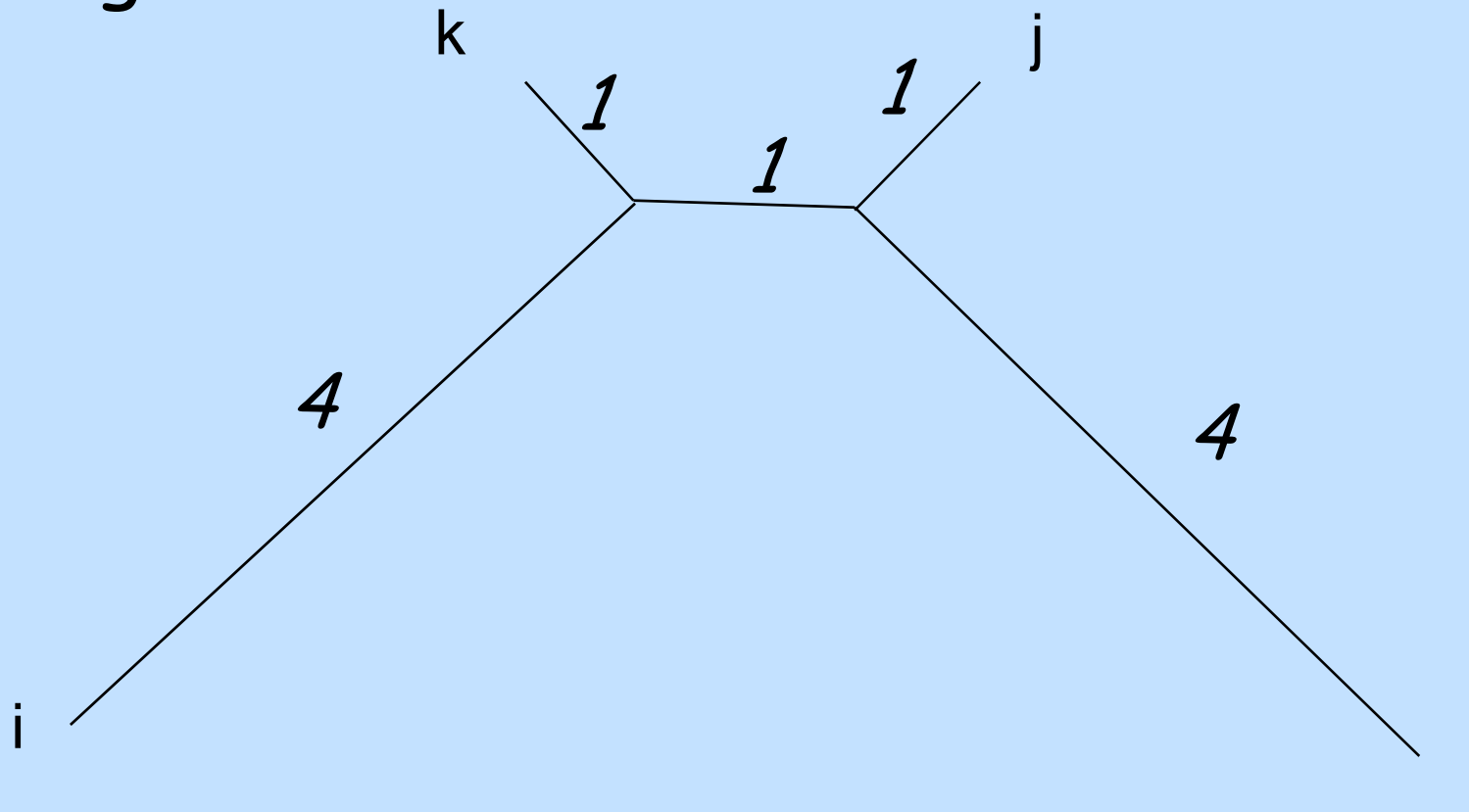
- If we can identify neighbor leaves, then can use pairwise distances to reconstruct the tree:
- Remove neighbors  $i, j$  from the leaf nodes
- Add  $k$
- Set  $d_{km} = (d_{im} + d_{jm} - d_{ij})/2$

But can we find neighbor leaves?



# Closest pairs may not be neighbors!

- Closest pair:  $kj$  even though they are not neighbors



# Neighbor Joining (Saitou-Nei '87)

- Let  $D(i, j) = d(i, j) - (r_i + r_j)$   
where

$$r_i = \frac{1}{|L| - 2} \sum_k d(i, k)$$

**Theorem:** if  $D(i, j)$  is minimal (among all pairs of leaves), then  $i$  and  $j$  are neighbors in the tree



# Neighbor Joining

$$r_i = \frac{1}{|L|-2} \sum_k d(i, k)$$
$$D(i, j) = d(i, j) - (r_i + r_j)$$

- Set  $L$  to contain all leaves

## Iteration:

- Choose  $i, j$  such that  $D(i, j)$  is minimal
- Create new node  $k$ , and set

$$d(i, k) = (d(i, j) + r_i - r_j) / 2$$

$$d(j, k) = (d(i, j) + r_j - r_i) / 2$$

$$d(k, m) = (d(i, m) + d(j, m) - d(i, j)) / 2$$

Time:  
 $O(n^3)$

- remove  $i, j$  from  $L$ , and add  $k$

**Termination:** if  $|L| = 2$ , connect two remaining nodes

**Thm:** Opt tree guaranteed if distances match a tree



# Neighbor Joining Algorithm

	A	B	C	D
A	-	17	21	27
B		-	12	18
C			-	14
D				-

# Neighbor Joining Algorithm

	A	B	C	D
A	-	17	21	27
B		-	12	18
C			-	14
D				-

i	$u_i$
A	$(17+21+27)/2=32.5$
B	$(17+12+18)/2=23.5$
C	$(21+12+14)/2=23.5$
D	$(27+18+14)/2=29.5$

# Neighbor Joining Algorithm

	A	B	C	D
A	-	17	21	27
B		-	12	18
C			-	14
D				-

i	$u_i$
A	$(17+21+27)/2=32.5$
B	$(17+12+18)/2=23.5$
C	$(21+12+14)/2=23.5$
D	$(27+18+14)/2=29.5$

	A	B	C	D
A	-	-39	-35	-35
B		-	-35	-35
C			-	-39
D				-

$$D_{ij} - u_i - u_j$$

# Neighbor Joining Algorithm

	A	B	C	D
A	-	17	21	27
B		-	12	18
C			-	14
D				-

i	$u_i$
A	$(17+21+27)/2=32.5$
B	$(17+12+18)/2=23.5$
C	$(21+12+14)/2=23.5$
D	$(27+18+14)/2=29.5$

	A	B	C	D
A	-	-39	-35	-35
B		-	-35	-35
C			-	-39
D				-

$$D_{ij} - u_i - u_j$$

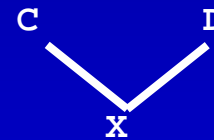
# Neighbor Joining Algorithm

	A	B	C	D
A	-	17	21	27
B		-	12	18
C			-	14
D				-

i	$u_i$
A	$(17+21+27)/2=32.5$
B	$(17+12+18)/2=23.5$
C	$(21+12+14)/2=23.5$
D	$(27+18+14)/2=29.5$

	A	B	C	D
A	-	-39	-35	-35
B		-	-35	-35
C			-	-39
D				-

$$D_{ij} - u_i - u_j$$



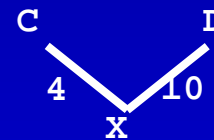
# Neighbor Joining Algorithm

	A	B	C	D
A	-	17	21	27
B		-	12	18
C			-	14
D				-

i	$u_i$
A	$(17+21+27)/2=32.5$
B	$(17+12+18)/2=23.5$
C	$(21+12+14)/2=23.5$
D	$(27+18+14)/2=29.5$

	A	B	C	D
A	-	-39	-35	-35
B		-	-35	-35
C			-	-39
D				-

$$D_{ij} - u_i - u_j$$

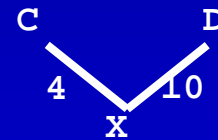


$$v_C = 0.5 \times 14 + 0.5 \times (23.5 - 29.5) = 4$$

$$v_D = 0.5 \times 14 + 0.5 \times (29.5 - 23.5) = 10$$

# Neighbor Joining Algorithm

	A	B	C	D	X
A	-	17	21	27	
B		-	12	18	
C			-	14	
D				-	
X					-

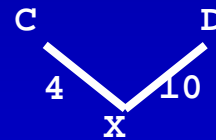


# Neighbor Joining Algorithm

	A	B	C	D	X
A	-	17	21	27	
B		-	12	18	
C			-	14	
D				-	
X					-

$$\begin{aligned}
 D_{XA} &= (D_{CA} + D_{DA} - D_{CD})/2 \\
 &= (21 + 27 - 14)/2 \\
 &= 17
 \end{aligned}$$

$$\begin{aligned}
 D_{XB} &= (D_{CB} + D_{DB} - D_{CD})/2 \\
 &= (12 + 18 - 14)/2 \\
 &= 8
 \end{aligned}$$

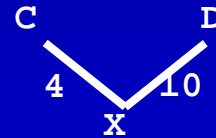


# Neighbor Joining Algorithm

	A	B	C	D	X
A	-	17	21	27	17
B		-	12	18	8
C			-	14	
D				-	
X					-

$$\begin{aligned}
 D_{XA} &= (D_{CA} + D_{DA} - D_{CD})/2 \\
 &= (21 + 27 - 14)/2 \\
 &= 17
 \end{aligned}$$

$$\begin{aligned}
 D_{XB} &= (D_{CB} + D_{DB} - D_{CD})/2 \\
 &= (12 + 18 - 14)/2 \\
 &= 8
 \end{aligned}$$

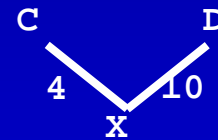


# Neighbor Joining Algorithm

	A	B	X
A	-	17	17
B		-	8
X			-

$$\begin{aligned}
 D_{XA} &= (D_{CA} + D_{DA} - D_{CD})/2 \\
 &= (21 + 27 - 14)/2 \\
 &= 17
 \end{aligned}$$

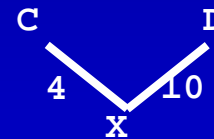
$$\begin{aligned}
 D_{XB} &= (D_{CB} + D_{DB} - D_{CD})/2 \\
 &= (12 + 18 - 14)/2 \\
 &= 8
 \end{aligned}$$



# Neighbor Joining Algorithm

	A	B	X
A	-	17	17
B		-	8
X			-

i	$u_i$
A	$(17+17)/1 = 34$
B	$(17+8)/1 = 25$
X	$(17+8)/1 = 25$



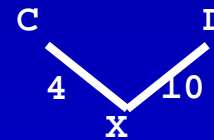
# Neighbor Joining Algorithm

	A	B	X
A	-	17	17
B		-	8
X			-

i	$u_i$
A	$(17+17)/1 = 34$
B	$(17+8)/1 = 25$
X	$(17+8)/1 = 25$

	A	B	X
A	-	-42	-28
B		-	-28
X			-

$$D_{ij} - u_i - u_j$$



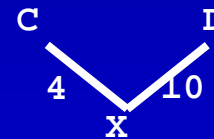
# Neighbor Joining Algorithm

	A	B	X
A	-	17	17
B		-	8
X			-

i	$u_i$
A	$(17+17)/1 = 34$
B	$(17+8)/1 = 25$
X	$(17+8)/1 = 25$

	A	B	X
A	-	-42	-28
B		-	-28
X			-

$$D_{ij} - u_i - u_j$$



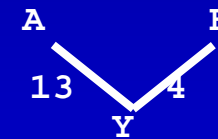
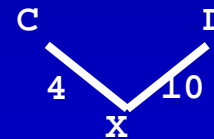
# Neighbor Joining Algorithm

	A	B	X
A	-	17	17
B		-	8
X			-

i	$u_i$
A	$(17+17)/1 = 34$
B	$(17+8)/1 = 25$
X	$(17+8)/1 = 25$

	A	B	X
A	-	-42	-28
B		-	-28
X			-

$$D_{ij} - u_i - u_j$$

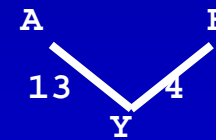
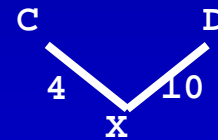


$$v_A = 0.5 \times 17 + 0.5 \times (34 - 25) = 13$$

$$v_D = 0.5 \times 17 + 0.5 \times (25 - 34) = 4$$

# Neighbor Joining Algorithm

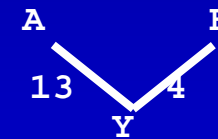
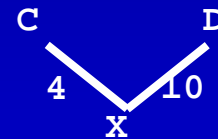
	A	B	X	Y
A	-	17	17	
B		-	8	
X			-	
Y				-



# Neighbor Joining Algorithm

	A	B	X	Y
A	-	17	17	
B		-	8	
X			-	4
Y				-

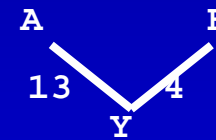
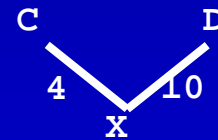
$$\begin{aligned}
 D_{YX} &= (D_{AX} + D_{BX} - D_{AB})/2 \\
 &= (17 + 8 - 17)/2 \\
 &= 4
 \end{aligned}$$



# Neighbor Joining Algorithm

	X	Y
X	-	4
Y		-

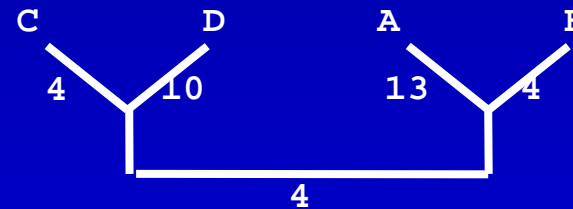
$$\begin{aligned}
 D_{YX} &= (D_{AX} + D_{BX} - D_{AB}) / 2 \\
 &= (17 + 8 - 17) / 2 \\
 &= 4
 \end{aligned}$$



# Neighbor Joining Algorithm

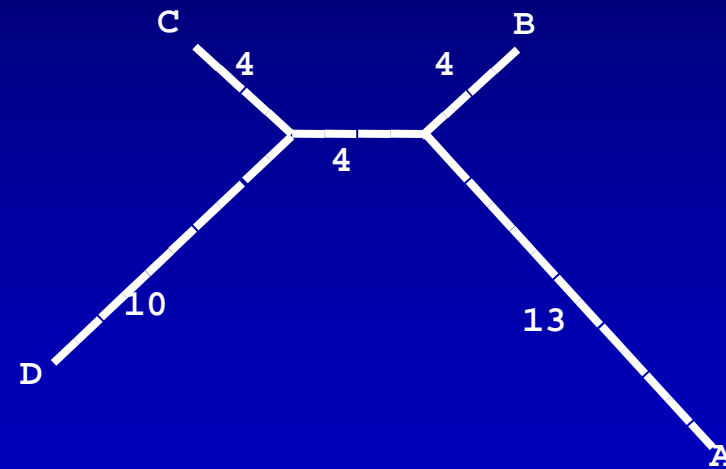
	X	Y
X	-	4
Y		-

$$\begin{aligned}
 D_{YX} &= (D_{AX} + D_{BX} - D_{AB})/2 \\
 &= (17 + 8 - 17)/2 \\
 &= 4
 \end{aligned}$$



# Neighbor Joining Algorithm

	A	B	C	D
A	-	17	21	27
B		-	12	18
C			-	14
D				-



# Naruya Saitou

- Division of Population Genetics, National Institute of Genetics & Department of Genetics, Graduate University for Advanced Studies (Sokendai) Mishima, 411-8540, Japan



# Probabilistic approaches



# Likelihood of a Tree

- Given:
  - $n$  aligned sequences  $M = X^1, \dots, X^n$
  - A tree  $T$ , leaves labeled with  $X^1, \dots, X^n$
- **reconstruction** :
  - labeling of internal nodes
  - branch lengths
- Goal: Find optimal reconstruction  $t^*$  : One maximizing the likelihood  $P(M/T, t^*)$

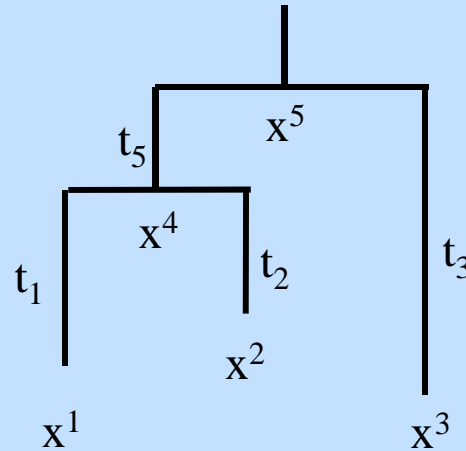


# Likelihood (2)

- We need a model for computing  $P(M/T, t^*)$
- Assumptions:
  - Each character is independent
  - The branching is a **Markov process**:
    - The probability that a node  $x$  has a specific label is only a function of the parent node  $y$  and the branch length  $t$  between them.
- The probabilities  $P(x/y, t)$  are known



# Example



$$P(x^1, \dots, x^5 | T, t^*) =$$

$$P(x^1 | x^4, t_1)P(x^2 | x^4, t_2)P(x^3 | x^5, t_3)P(x^4 | x^5, t_4)P(x^5)$$



# Calculating the Likelihood

Assume that the branch lengths  $t_{uv}$  are known.

$$P(M | T, t^*) = \prod_{\text{character } j} \left\{ \sum_{\text{reconstruction } R} P(M_{\cdot j}, R | T, t^*) \right\}$$
$$= \prod_{\text{character } j} \left\{ \sum_{\text{reconstruction } R} \left( P(\text{root}) \prod_{\text{edge } u \rightarrow v} P_{u \rightarrow v}(t_{uv}) \right) \right\}$$

Independence  
of sites

Markov property  
independence of  
each branch



# Additional Assumed Properties

Additivity:

$$\sum_b P(a | b, t) P(b | c, s) = P(a | c, s + t)$$

Reversibility (symmetry):

$$q_b P(a | b, t) = q_a P(b | a, t)$$

Allows one to freely move the root

Provable under broad and reasonable  
assumptions



# Jukes-Cantor Model (J-K '69)

- Assumptions:
  - Each base in sequence has equal chance of changing
  - Changes to other 3 bases with equal probability
- Characteristics
  - Each base appears with equal frequency in DNA
  - The quantity  $r$  is the rate of change
  - **NOTE:**  $r$  is not the rate of mutation, it is the rate of substitution event of a nucleotide throughout a population.
  - During each infinitesimal time  $\Delta t$  a substitution occurs with probability  $3r\Delta t$



# Jukes-Cantor Model

- $P_{A(t)}$  : prob of A in the character at time t
- Discrete case: Prob. change  $A \rightarrow B$  in one time unit is r
  - $P_{A(t+1)} = (1-3r) P_{A(t)} + r (1- P_{A(t)})$
  - $P_{A(t+1)} - P_{A(t)} = -3r P_{A(t)} + r (1- P_{A(t)})$
- Continuous case:
  - $\Delta P_{A(t)} = -3r P_{A(t)} + r (1- P_{A(t)}) = -4r P_{A(t)} + r$
- Solution to the differential equation:

$$P_{A(t)} = \frac{1}{4}(1 + 3e^{-4rt})$$



- prob. that the nucleotide remains unchanged over  $t$  time units:

$$P_{\text{same}} = \frac{1}{4} + \frac{3}{4}e^{-4rt}$$

- Probability of specific change:

$$P_{A \rightarrow B} = \frac{1}{4} - \frac{1}{4}e^{-4rt}$$

- Probability of change:

$$P_{\text{change}} = \frac{3}{4} - \frac{3}{4}e^{-4rt}$$

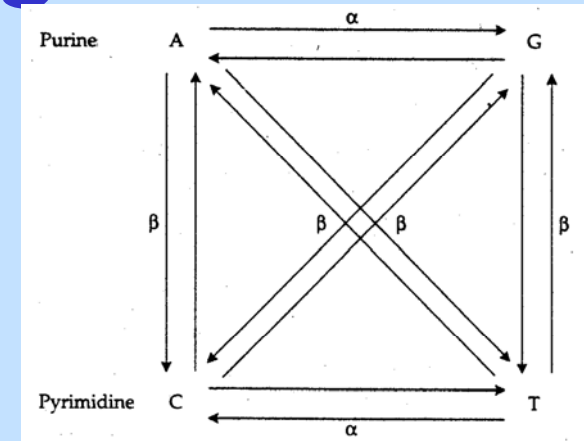
- Note: For

$$t \rightarrow \infty \quad P_{\text{change}} \rightarrow \frac{3}{4}$$



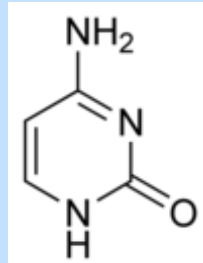
# Other Models

- Kimura's 2-parameter model:
  - A,G - purines; C,T - pyrimidines
  - Two different rates

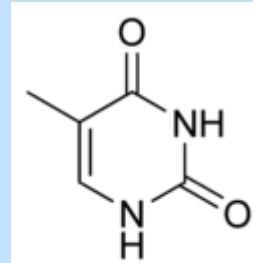


- purine-purine or pyrimidine-pyrimidine (transitions)
- purine-pyrimidine or pyrimidine-purine (transversions)

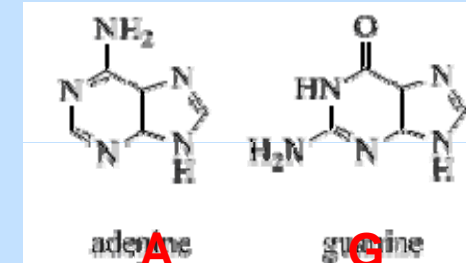
- Felsenstein '84 and Yano, Hasegawa & Kishino '85 extend the Kimura model to asymmetric base frequencies.



C



T



A

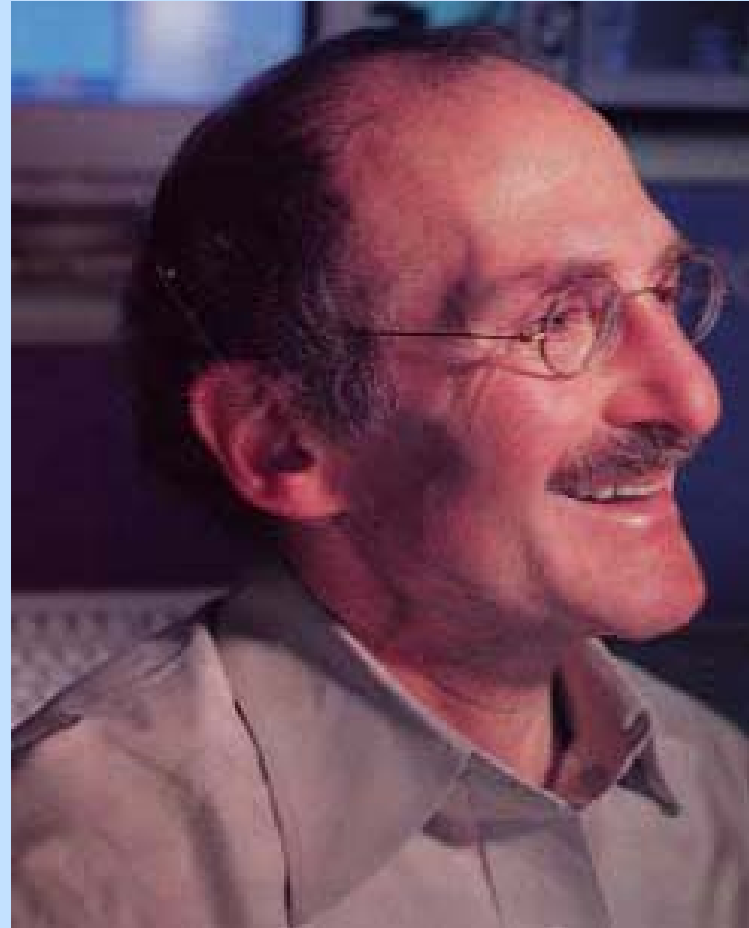
G



# Charles Cantor

Boston University  
Professor, Biomedical  
Engineering  
Professor of Pharmacology,  
School of Medicine  
Center for Advanced  
Biotechnology  
Ph.D., Biophysical Chemistry,  
University of California,  
Berkeley

Dr. Cantor is currently Chief  
Scientific Officer at  
Sequenom, Inc in San  
Diego, California.



# Efficient Likelihood Calculation (Felsenstein '73)

Use dynamic prog. similar to parsimony

Need  $S_j(a, v) = \Pr(\text{subtree rooted in } v \mid v_j = a)$

## Initialization:

For each leaf  $v$  set  $S_j(a, v) = 1$  if  $v$  is labeled by  $a$ , otherwise  
 $S_j(a, v) = 0$

## Recursion:

Traverse the tree in postorder: for each node  $v$  with children  $u$  and  $w$ , for each state  $x$

$$S_j(x, v) = \left( \sum_y S_j(y, u) p_{x \rightarrow y}(t_{vu}) \right) \left( \sum_y S_j(y, w) p_{x \rightarrow y}(t_{vw}) \right)$$

**Final Soln:**  $L = \prod_j \left( \sum_x S_j(x, \text{root}) P(x) \right)$

**Complexit**

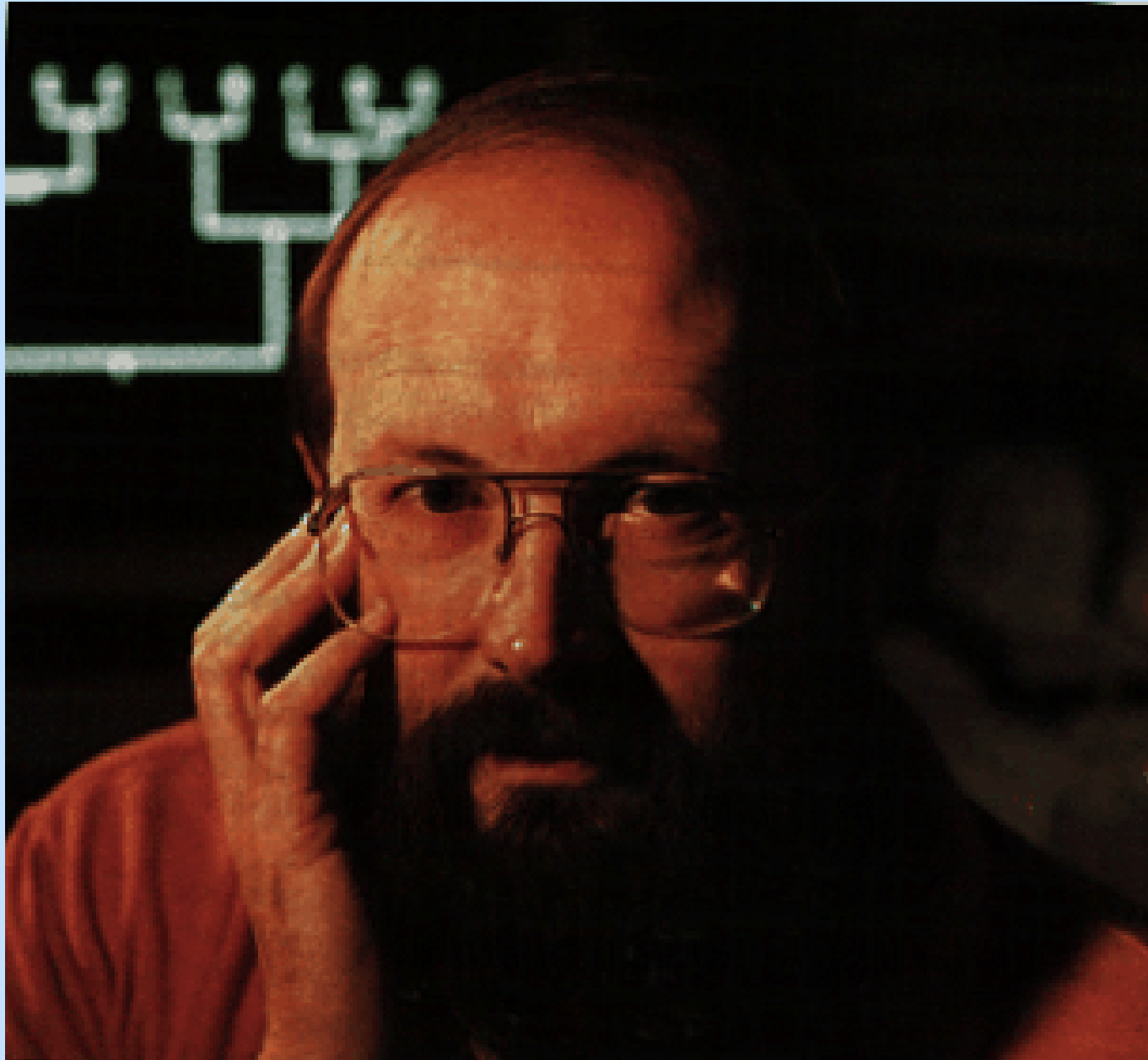
**y:**

**$O(nmk^2)$**

**n species, m**

**chars, k**





# Finding Optimal Branch lengths

- Optimizing the length of a **single** root-child branch **r-v** can be done using standard optimization techniques

$$\log L = \sum_{j=1}^m \log \left( \sum_{x,y} p(x) \cdot S''_j(x,r) \cdot p_{x \rightarrow y}(t_{rv}) \cdot S_j(y,v) \right)$$

- Under the symmetry assumption, each node can be made (temporarily) the root
- To optimize all the branch lengths: repeatedly optimize one branch at a time (and pray)
  - No guaranteed convergence, but often works



# Character Based Methods



# Inferring a Phylogenetic Tree

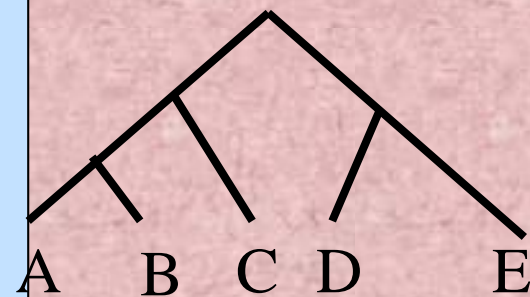
## Generic problem: Optimal Phylogenetic Tree:

- Input:
  - $n$  species,
  - set of characters,
  - for each species, the state of each of the characters.
  - (parameters)
- Goal: find a fully-labeled phylogenetic tree that best explains the data. (maximizes a target function).

### Assumptions:

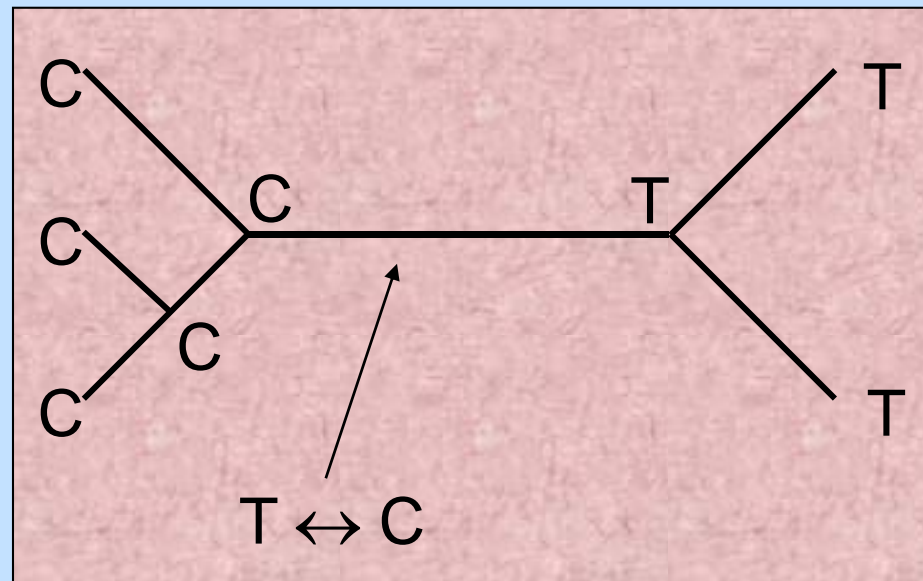
- characters - mutually independent
- species evolve independently

```
A: CAGGTA
B: CAGACA
C: CGGGTA
D: TGC ACT
E: TGCGTA
```



# A Simple Example

- Five species, three have 'C' and two 'T' at a specified position.
- A minimal tree has one evolutionary change:



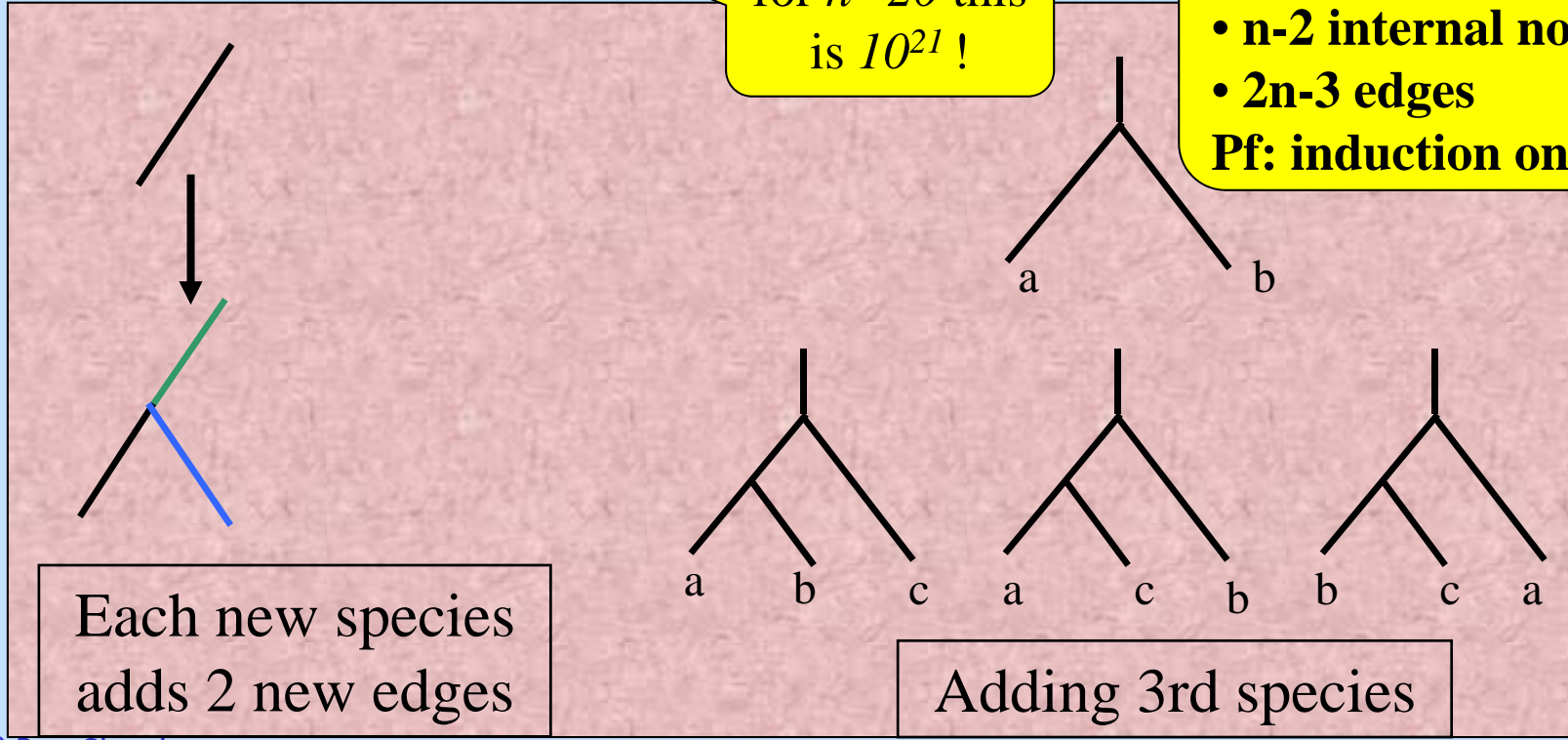
# Inferring a Phylogenetic Tree

## Naive Solution - Enumeration:

- No. of non-isomorphic, labeled, binary, unrooted trees, containing  $n$  leaves:  $(2n - 5)!! = \prod_{i=3 \dots n} (2i - 5)$
- Rooted:  $x(2n - 3)$

for  $n=20$  this is  $10^{21}$  !

**$n$  leaves  $\rightarrow$**   
 •  $n-2$  internal nodes  
 •  $2n-3$  edges  
**Pf: induction on  $n$**



# Parsimony

- **Goal**: explain data with min. no. of evolutionary changes (“mutations”, or mismatches)
- **Parsimony**:  $S(T) \equiv \sum_j \sum_{\{v,u\} \in E(T)} |\{j: v_j \neq u_j\}|$
- **“Small parsimony problem”**:
  - Input: leaf sequences + a leaf-labeled tree T
  - Goal: Find ancestral sequences implying minimum no. of changes (*most parsimonious*)
- **“Large parsimony problem”**:
  - Input: leaf sequences
  - Goal: Find a most parsimonious tree (topology, leaf labeling and internal seqs.)



# Algorithm for the Small Parsimony Problem (Fitch '71)

- Initialization: scan  $T$  in post-order, assign:

- leaf vertex  $m$ :  $S_m = \{\text{state at node } m\}$

- internal node  $m$  with children  $l, r$ :

$$S_m = \begin{cases} S_l \cup S_r & \text{if } S_l \cap S_r = \emptyset \quad (i) \\ S_l \cap S_r & \text{o/w} \quad (ii) \end{cases}$$

cost = no. of case (i) events.

- Solution Construction: scan tree in preorder, choose:

- for the root choose  $x \in S_{\text{root}}$

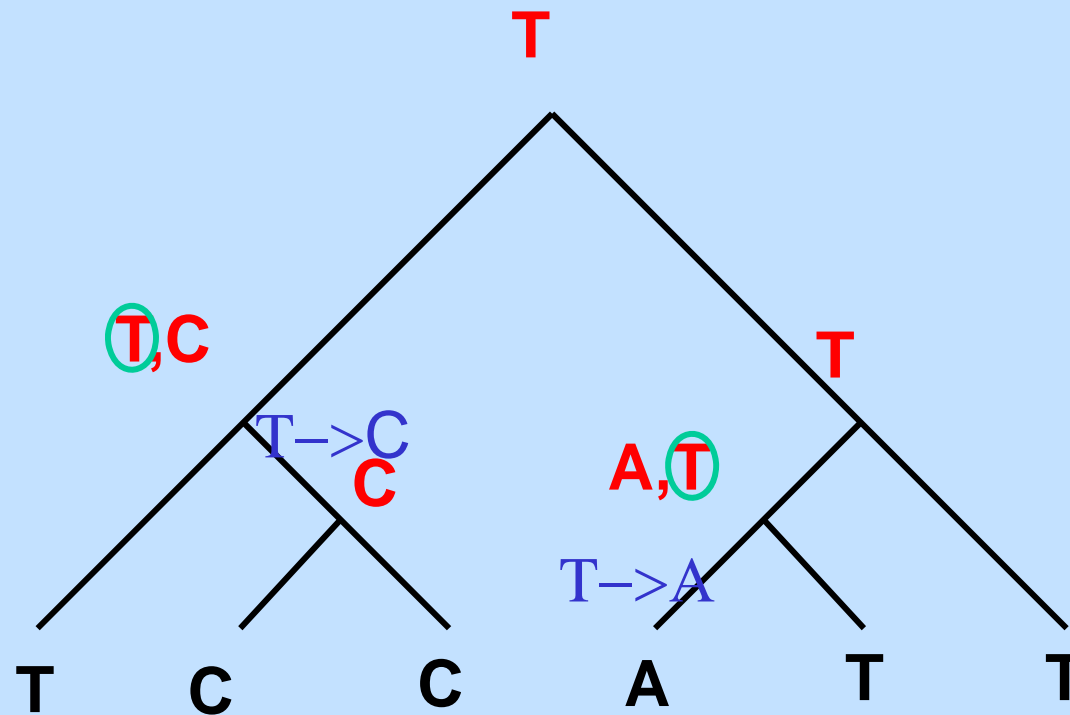
- at node  $m$  with child  $k$ :

time:  $O(n \cdot k)$   
( $k = \# \text{states } s$ )

*pick same state as  $k$  if possible; o/w - pick arbitrarily*

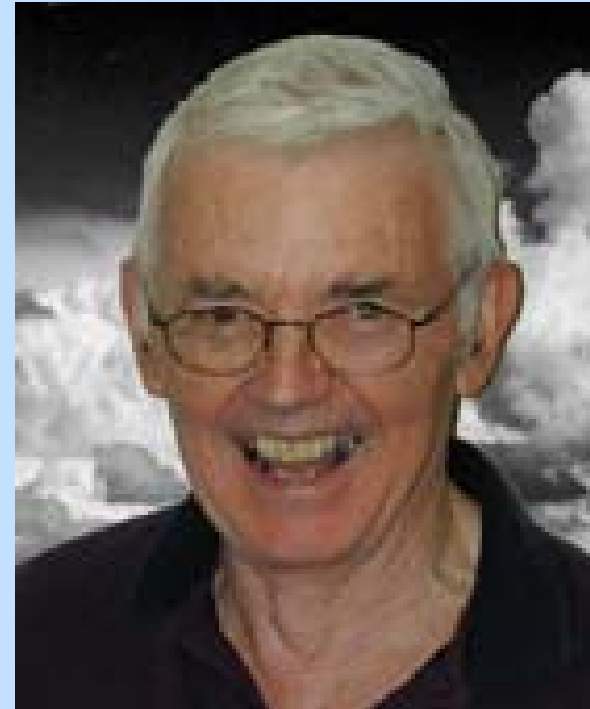


# Fitch's Alg



# Walter Fitch

- For more than 30 years, **Walter Fitch** has been recognized as one of the most influential evolutionary biologists in the world, and he has been credited as establishing a new scientific field: molecular phylogenetics. He has been elected to membership in the National Academy of Sciences, the American Academy of Arts and Sciences and the American Philosophical Society. He co-founded and was the first president of the Society for Molecular Biology, which established the annually awarded Fitch Prize. Additionally, he was a founding editor of *Molecular Biology and Evolution*.
- Fitch is at the **University of California, Irvine**, Dept. of Ecology and Evolutionary Biology



# Weighted Parsimony

$k$  states  $S_1, \dots, S_k$

$C_{ij}$  = cost of changing from state  $i$  to  $j$

Algorithm (Sankoff-Cedergren '88):

- Need:  $S_k(x)$  - best cost for the subtree rooted at  $x$  if state at  $x$  is  $k$

- For leaf  $x$ ,

$$S_k(x) = \begin{cases} 0 & \text{if state of } x \text{ is } k \\ \infty & \text{o/w} \end{cases}$$

- Scan  $T$  in postorder. At node  $a$  with children  $l, r$

- $S_k(a) = \min_m (S_m(l) + C_{mk}) + \min_m (S_m(r) + C_{mk})$

- $\text{Opt} = \min_m (S_m(\text{root}))$

time:  $O(n \cdot k^2)$



# David Sankoff



Over the past 30 years, Sankoff formulated and contributed to many of the fundamental problems in computational biology.

In **sequence comparison**, he introduced the **quadratic version of the Needleman-Wunsch algorithm**, developed the **first statistical test for alignments**, initiated the study of the limit behavior of random sequences with Vaclav Chvatal and described the **multiple alignment problem**, based on minimum evolution over a phylogenetic tree. In the study of **RNA secondary structure**, he developed algorithms based on general energy functions for **multiple loops** and for **simultaneous folding and alignment**, and performed the earliest studies of **parametric folding** and **automated phylogenetic filtering**.

Sankoff and Robert Cedergren collaborated on the first studies of the **evolution of the genetic code** based on tRNA sequences. His contributions to **phylogenetics** include early models for **horizontal transfer**, a general approach for optimizing the nodes of a given tree, a method for rapid bootstrap calculations, a generalization of the **nearest neighbor interchange** heuristic, various constraint, consensus and supertree problems, the computational complexity of several phylogeny problems with William Day, and a general technique for **phylogenetic invariants** with Vincent Ferretti. Over the last fifteen years he has focused on the evolution of genomes as the result of **chromosomal rearrangement** processes. Here he introduced the computational analysis of **genomic edit distances**, including parametric versions, the distribution of gene numbers in conserved segments in a random model with Joseph Nadeau, phylogeny based on **gene order** with Mathieu Blanchette and David Bryant, generalizations to include multi-gene families, including algorithms for analyzing **genome duplication** and hybridization with Nadia El-Mabrouk, and the statistical analysis of gene clusters with Dannie Durand. Sankoff is also well known in **linguistics** for his methods of studying grammatical variation and change in speech communities, the quantification of discourse analysis and production models of bilingual speech.



# Large Parsimony Problem

Input:  $n \times m$  matrix  $M$ :

- $M_{ij}$  = state of  $j^{\text{th}}$  character of species  $i$ .
- $M_{i\cdot}$  = label of  $i$  (all labels are distinct)

Goal:

Construct a phylogenetic tree  $T$  with  $n$  leaves and a label for each node, s.t.

- 1-1 correspondence of leaves and labels
- cost of tree is minimum.
  
- NP-hard



# Branch & Bound (Hendy-Penny '89)

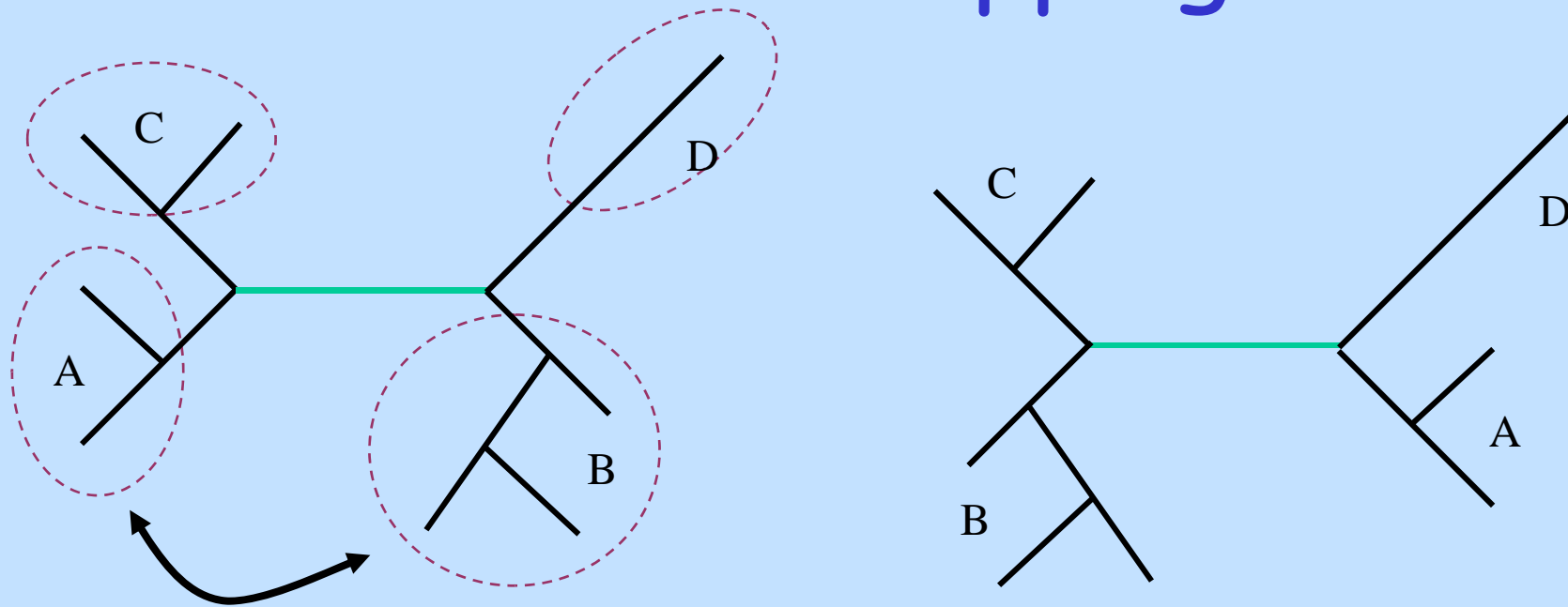
- enumerate all unrooted trees with increasing no. of leaves

Note: cost of tree with all leaves  $\geq$  cost of subtree with some leaves pruned (and same labeling)  
=> If cost of subtree  $\geq$  best cost for full tree so far, then: can prune (ignore) all refinements of the subtree.

enumeration & pruning can be done in  $O(1)$  time per visited subtree.



# Branch swapping



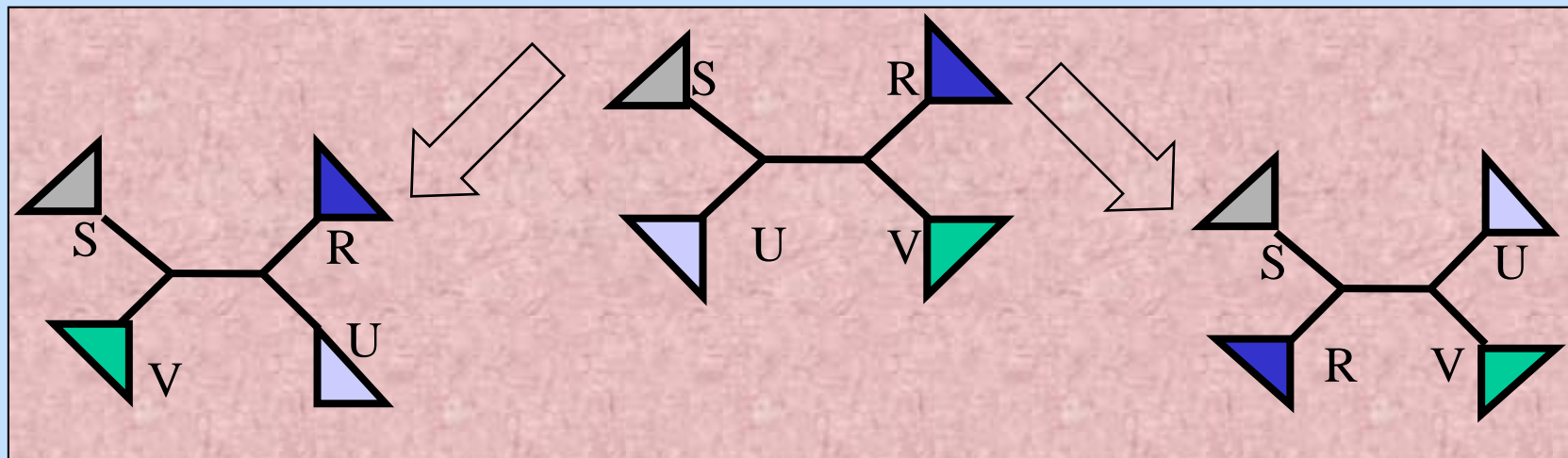
Each internal edge defines 4 sub-trees:

Can swap two such non-adjacent sub-trees



# Nearest Neighbor Interchanges

- handles  $n$ -labeled trees
- $T$  and  $T'$  are **neighbors** if one can get  $T'$  by following operation on  $T$ :



**SV|RU**

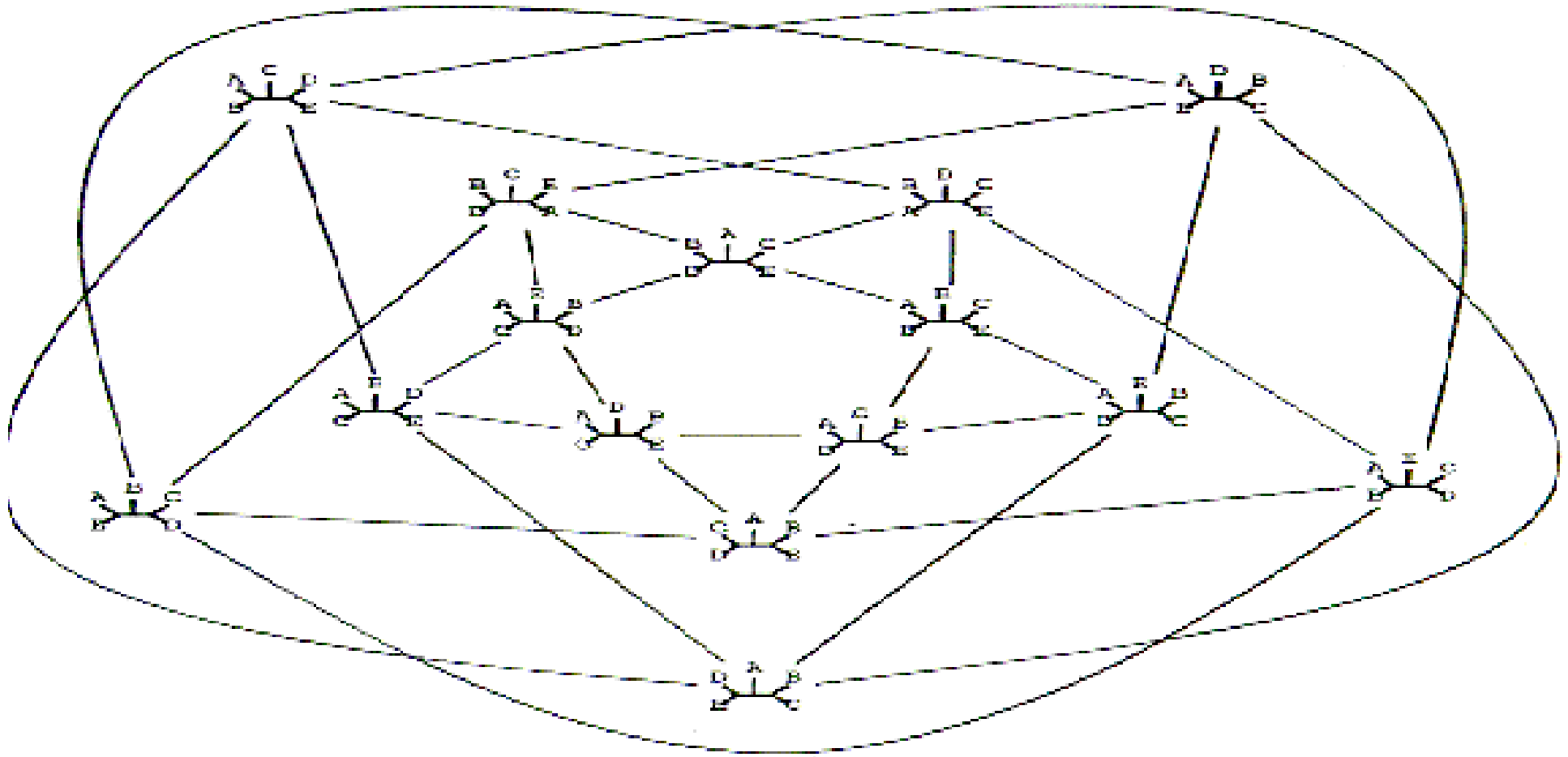
**SU|RV**

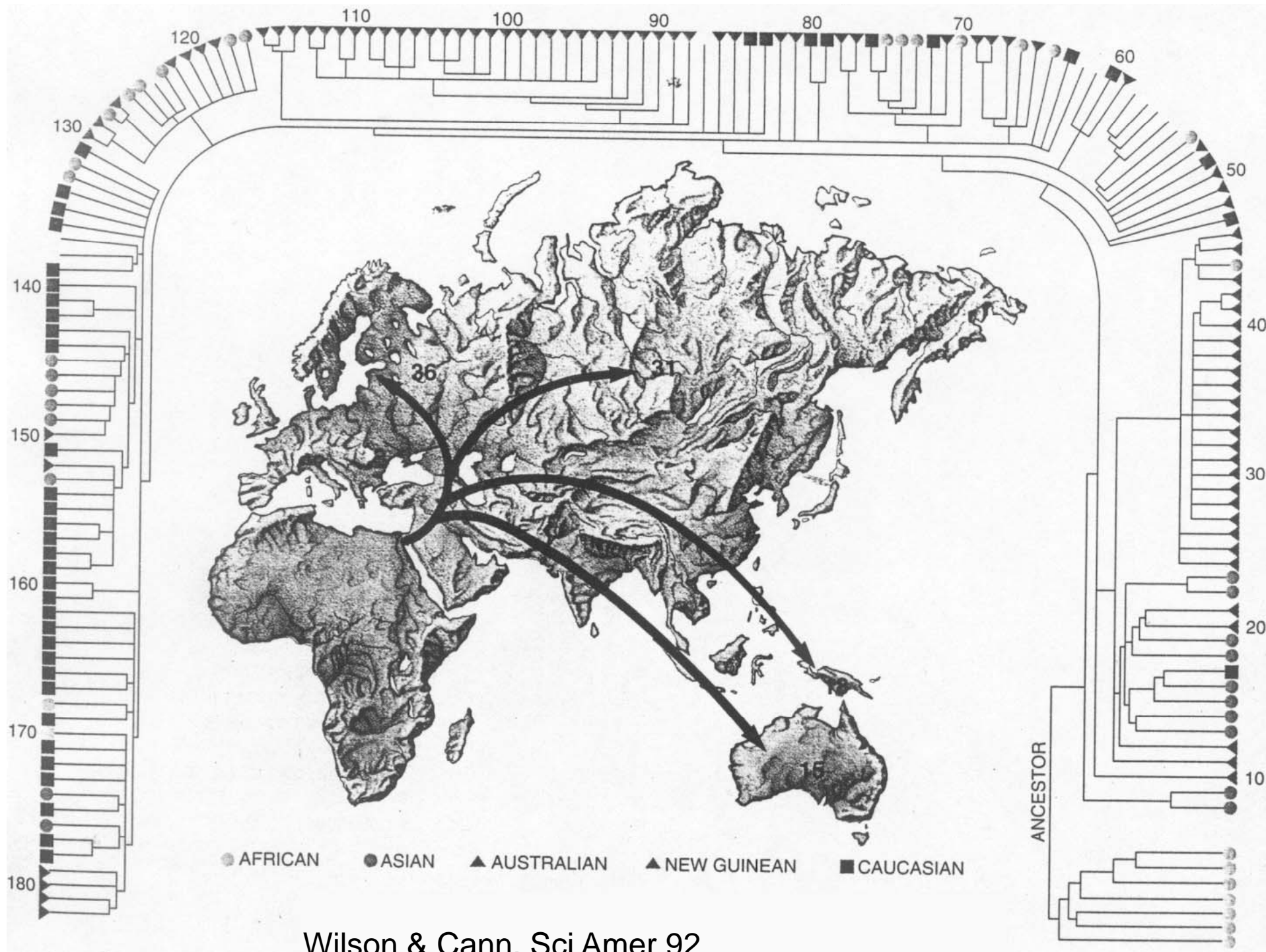
**SR|UV**

- Use the neighborhood structure on the set of solutions (all trees) via *hill climbing*, *annealing*, other heuristics...



# NN interchange graph





Wilson & Cann. Sci Amer 92

FIN

