

Lecture 11

Phylogenetic trees

Principles of Computational
Biology

Teresa Przytycka, PhD

Phylogenetic (evolutionary) Tree

- showing the evolutionary relationships among various biological species or other entities that are believed to have a common ancestor.
- Each node is called a taxonomic unit.
- Internal nodes are generally called hypothetical taxonomic units
- In a phylogenetic tree, each node with descendants represents the most recent common ancestor of the descendants, and the
- edge lengths (if present) correspond to time estimates.

Methods to construct phylogenetic trees

- Parsimony
- Distance matrix based
- Maximum likelihood

Parsimony methods

The preferred evolutionary tree is
the one that requires

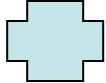
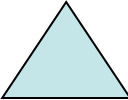
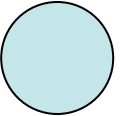
“the minimum net amount of evolution”

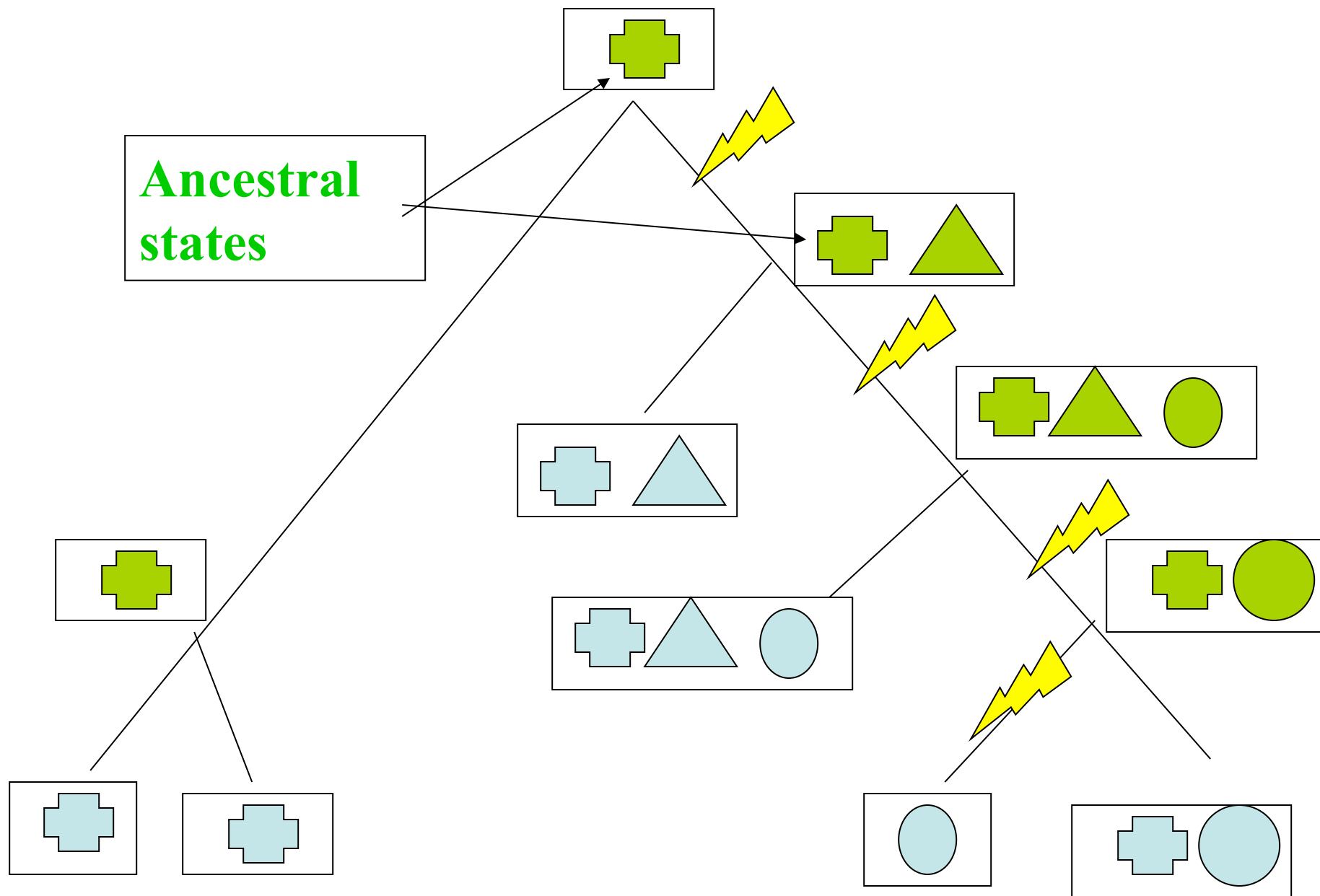
[Edwards and Cavalli-Sforza, 1963]

Assumption of character based parsimony

- Each taxa is described by a set of characters
- Each character can be in one of finite number of states
- In one step certain changes are allowed in character states
- Goal: find evolutionary tree that explains the states of the taxa with minimal number of changes

Example

			
Taxon1	Yes	Yes	No
Taxon 2	YES	Yes	Yes
Taxon 3	Yes	No	No
Taxon 4	Yes	No	No
Taxon 5	Yes	No	Yes
Taxon 6	No	No	Yes



Ancestral
states

4 Changes

Version parsimony models:

- Character states

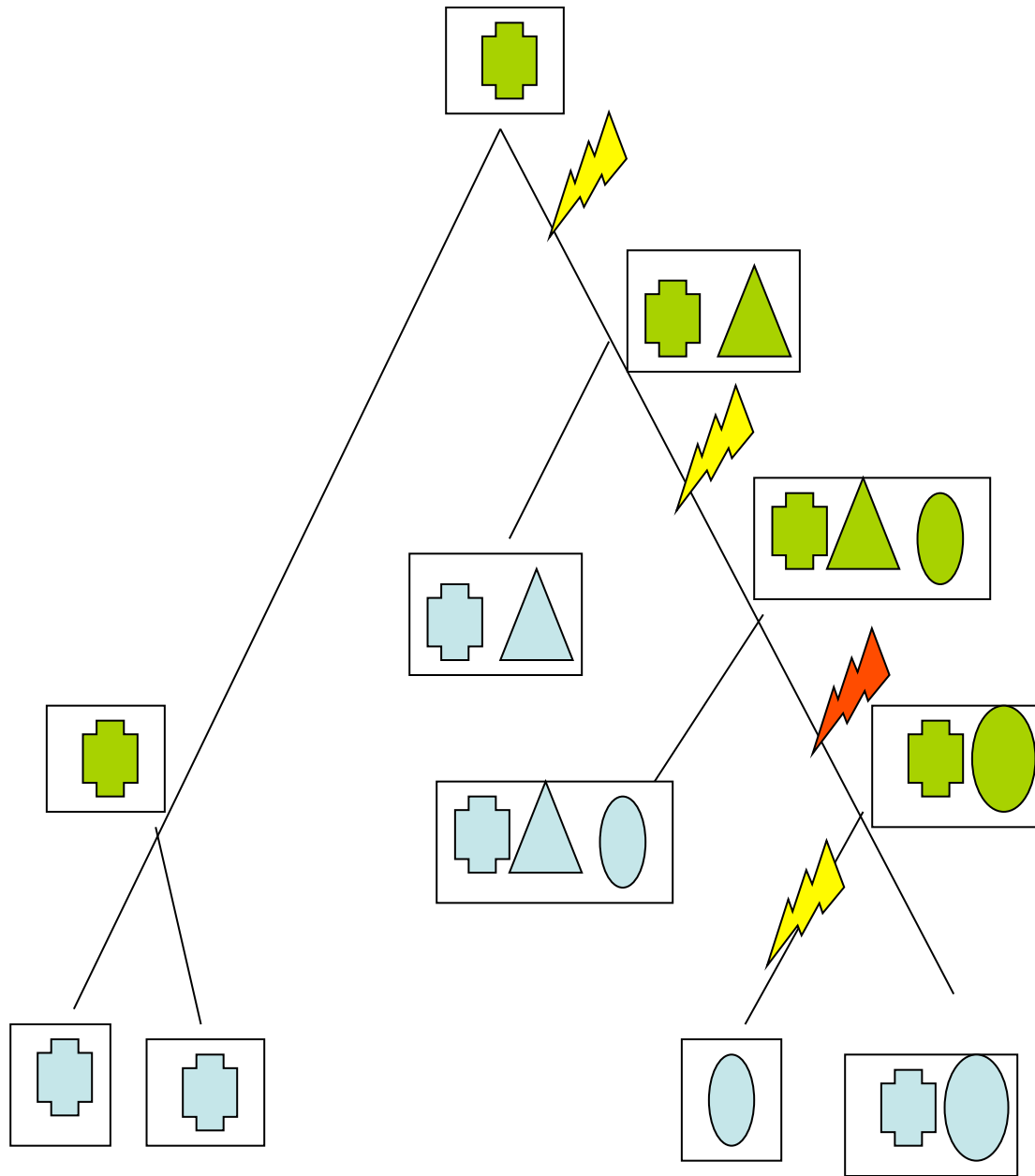
- Binary: states are 0 and 1 usually interpreted as presence or absence of an attribute (eg. character is a gene and can be present or absent in a genome)
- Multistate: Any number of states (Eg. Characters are position in a multiple sequence alignment and states are A,C,T,G).

- Type of changes:

- Characters are **ordered** (the changes have to happen in particular order or not).
- The changes are **reversible** or not.

Variants of parsimony

- **Fitch Parsimony** unordered, multistate characters with reversibility
- **Wagner Parsimony** ordered, multistate characters with reversibility
- **Dollo Parsimony** ordered, binary characters with reversibility but only one insertion allowed per character characters that are relatively hard to gain but easy to lose (like introns)
- **Camin-Sokal Parsimony**- no reversals, derived states arise once only
- (binary) **perfect phylogeny** – binary and non-reversible; each character changes at most once.

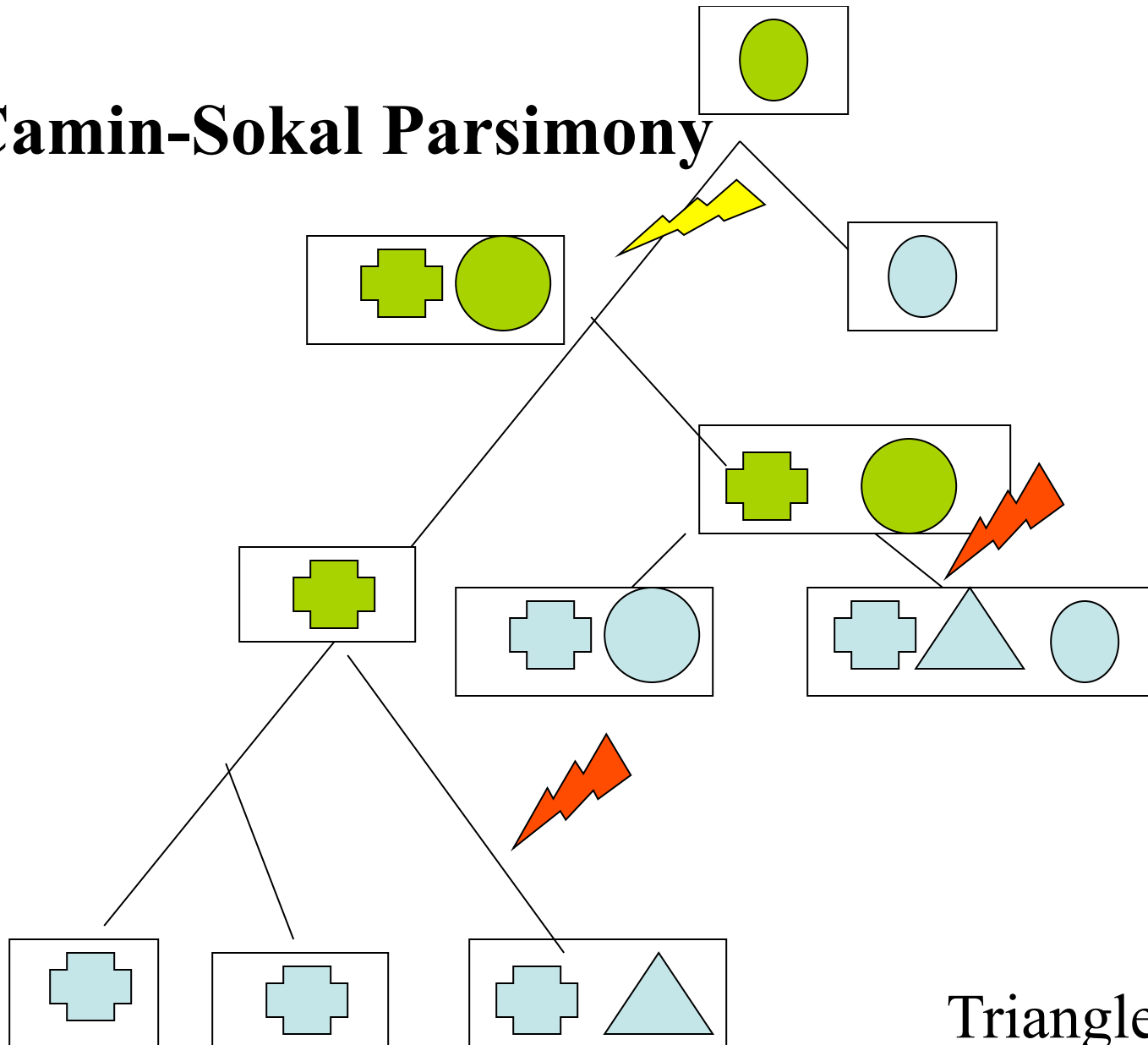


**Prefect – No
(triangle gained
and the lost)**

Dollo – Yes

**Camin-Sokal –
No (for the same
reason as perfect)**

Camin-Sokal Parsimony



3 Changes

Homoplasy

Having some states arise more than once is called homoplasy.

Example – triangle in the tree on the previous slide

Finding most parsimonious tree

- There are exponentially many trees with n nodes
- Finding most parsimonious tree is NP-complete (for most variants of parsimony models)
- Exception: Perfect phylogeny if exists can be found quickly. Problem – perfect phylogeny is too restrictive in practice.

Perfect phylogeny

- Each change can happen only once and is not reversible.
- Can be directed or not

Example: Consider binary characters. Each character corresponds to a gene.

0-gene absent

1-gene present

It make sense to assume directed changes only form 0 to 1.

The root has to be all zeros

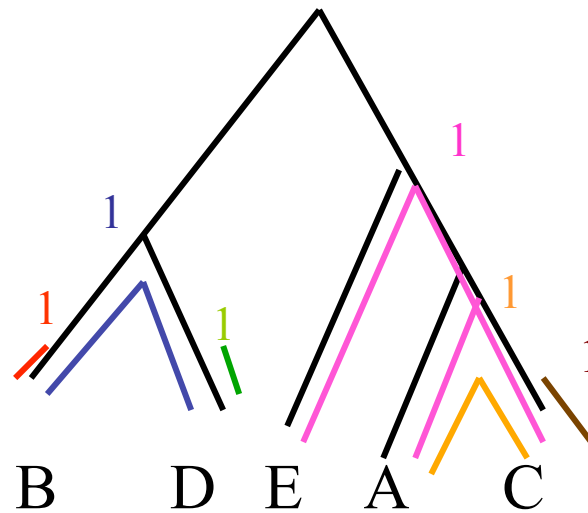
Perfect phylogeny

Example: characters = genes; 0 = absent ; 1 = present

Taxa: genomes (A,B,C,D,E)

genes

A	0	0	0	1	1	0
B	1	1	0	0	0	0
C	0	0	0	1	1	1
D	1	0	1	0	0	0
E	0	0	0	1	0	0



Perfect phylogeny tree

Goal: For a given **character state matrix** construct a tree topology that provides perfect phylogeny.

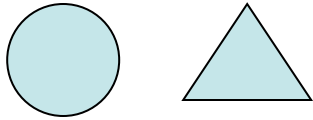
Does there exist perfect
parsimony tree for our example
with geometrical shapes?

There is a simple test

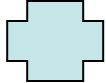
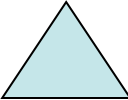
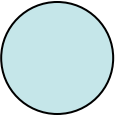
Character Compatibility

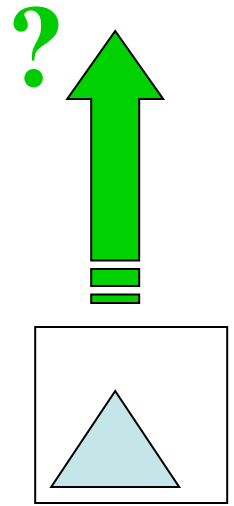
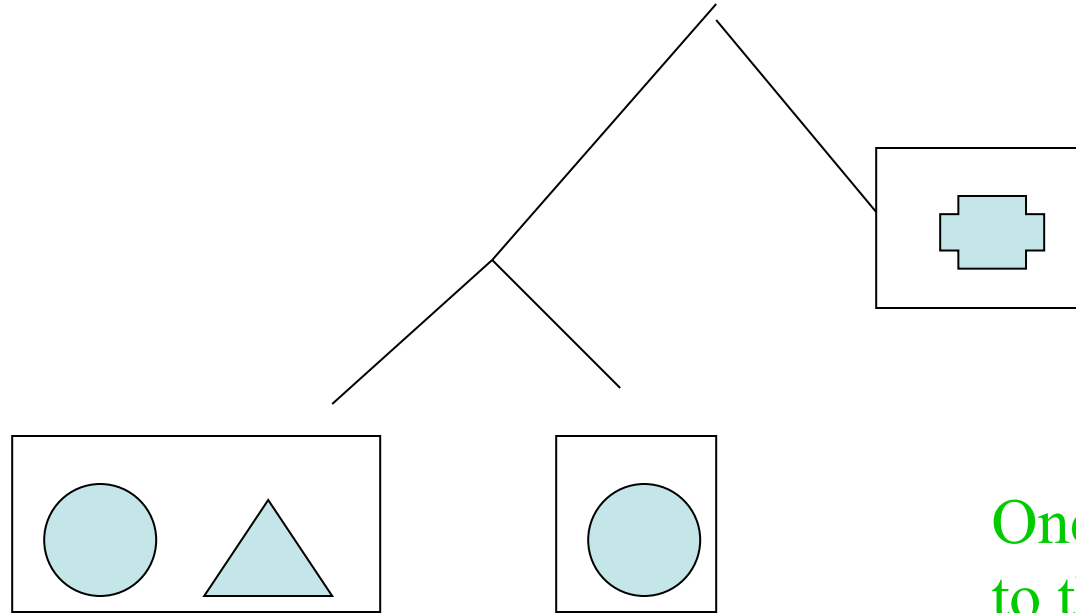
- Two characters A, B are **compatible** if there do not exist four taxa containing all four combinations as in the table
- Fact: **there exists perfect phylogeny if and only if and only if all pairs of characters are compatible**

	A	B
T1	1	1
T2	1	0
T3	0	1
T4	0	0



Are not compatible

			
Taxon1	Yes	Yes	No
Taxon 2	YES	Yes	Yes
Taxon 3	Yes	No	No
Taxon 4	Yes	No	No
Taxon 5	Yes	No	Yes
Taxon 6	No	No	Yes



One cannot add triangle to the tree so that no character changes its state twice:

If we add it to one of the left branches it will be inserted twice if to the right most – circle would have to be deleted (insertion and the deletion of the circle)

Ordered characters and perfect phylogeny

- Assume that we in the last common ancestor all characters had state 0.
- This assumption makes sense for many characters, for example genes.
- Then compatibility criterion is even simpler: **characters are compatible if and only if there do not exist three taxa containing combinations $(1,0)$, $(0,1)$, $(1,1)$**

Example

A 0 0 0 1 1 0
B 1 1 0 0 0 0
C 0 0 0 1 1 1
D 1 0 1 0 0 0
E 0 0 0 1 0 0

Under assumption that states are directed from 0 to 1: if i and j are two different genes then the set of species containing i is either disjoint with set of species containing j or one of these sets contains the other.

- The above property is **necessary and sufficient for perfect phylogeny** under 0 to 1 ordering
- Why works: associated with each character is a subtree. These subtrees have to be nested.

Simple test for perfect phylogeny

- Fact: there exists perfect phylogeny if and only if and only if all pairs of characters are compatible
- Special case: if we assume directed parsimony ($0 \rightarrow 1$ only) then characters are compatible if and only if there do not exist tree taxa containing combinations $(1,0), (0,1), (1,1)$
- Observe the last one is equivalent to non-overlapping criterion
- Optimal algorithm: Gusfield $O(nm)$:
 $n = \# \text{ taxa}; m = \# \text{ characters}$

Two version optimization problem:

Small parsimony: Tree is given and we want to find the labeling that minimizes #changes – there are good algorithms to do it.

Large parsimony: Find the tree that minimize number of evolutionary changes. For most models NP complete

One approach to large parsimony requires:

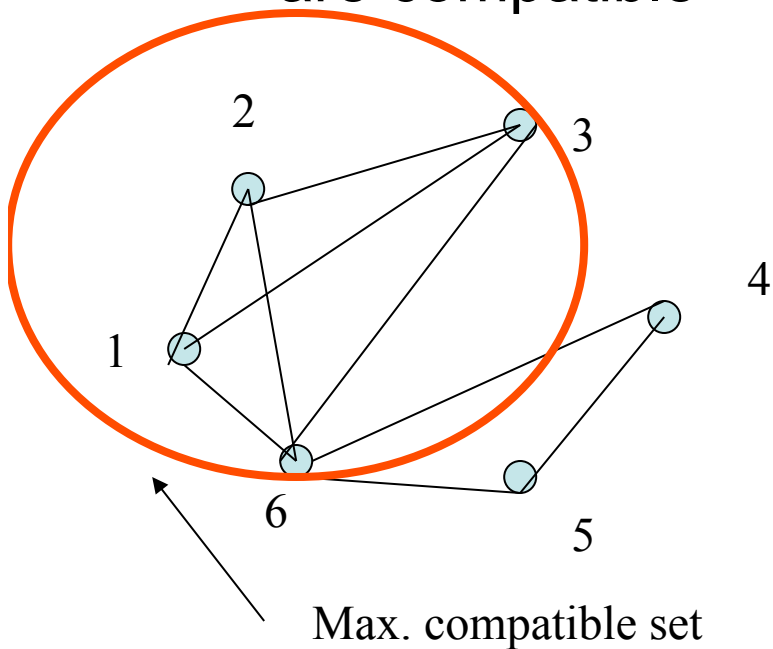
- generating all possible trees
- finding optimal labeling of internal nodes for each tree.

Fact 1: #tree topologies grows exponentially with #nodes

Fact 2: There may be many possible labels leading to the same score.

Clique method for large parsimony

- Consider the following graph:
 - nodes – characters;
 - edge if two characters are compatible



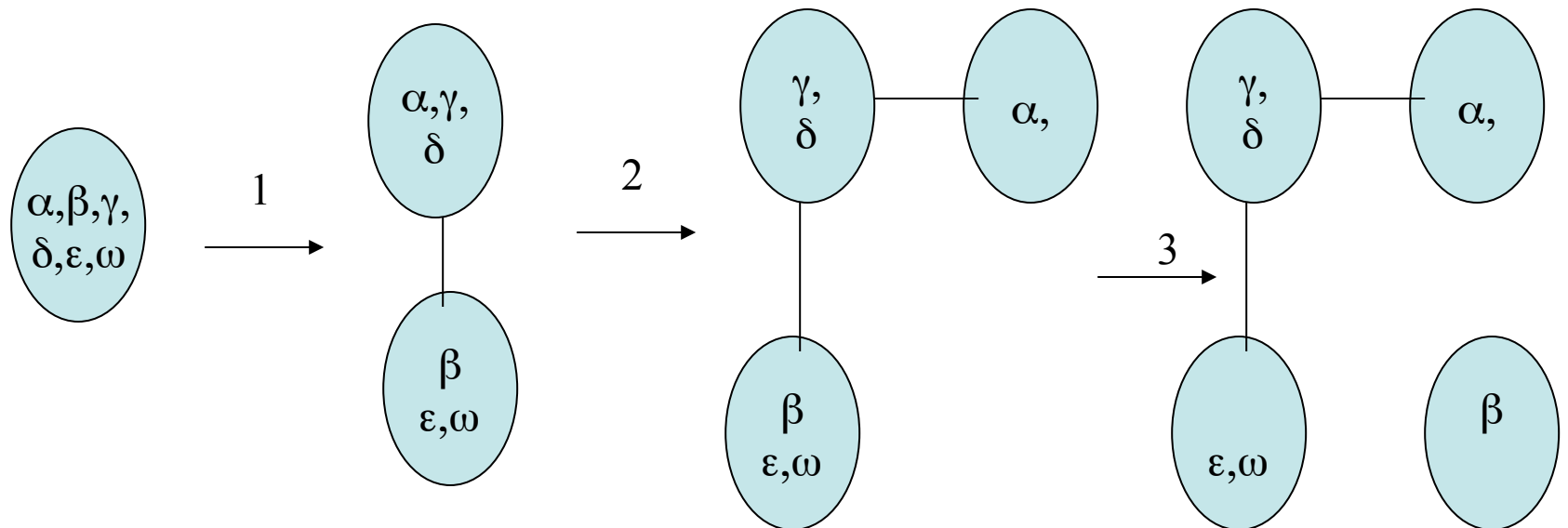
characters

	1	2	3	4	5	6
α	1	0	0	1	1	0
β	0	0	1	0	0	0
γ	1	1	0	0	0	0
δ	1	1	0	1	1	1
ϵ	0	0	1	1	1	0
ω	0	0	0	0	0	0

3,5 INCOMPATIBLE

Clique method (Meacham 1981) -

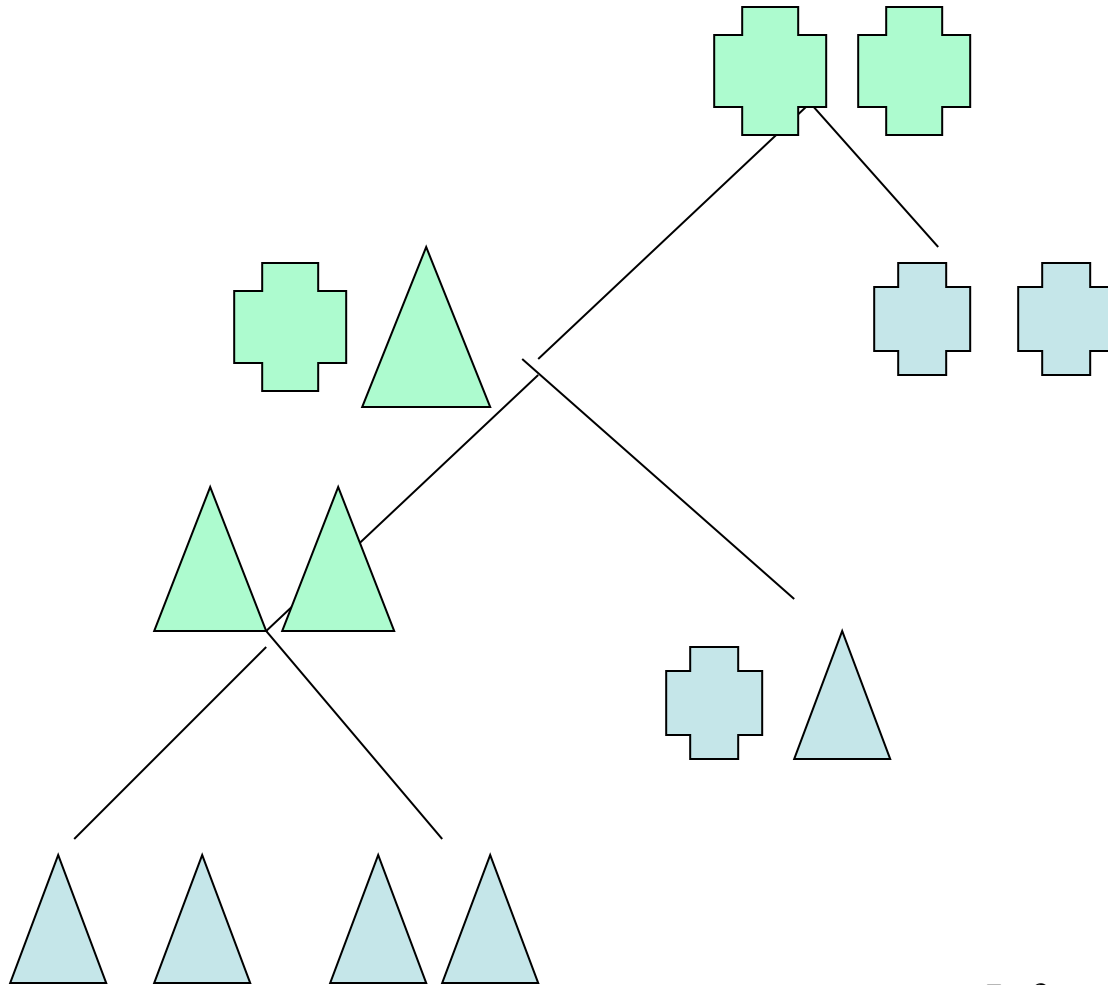
- Find maximal compatible clique (NP-complete problem)
- Each character defines a partition of the set into two subsets



Small parsimony

- Assumptions: the tree is known
- Goal: find the optimal labeling of the tree (optimal = minimizing cost under given parsimony assumption)

“Small” parsimony

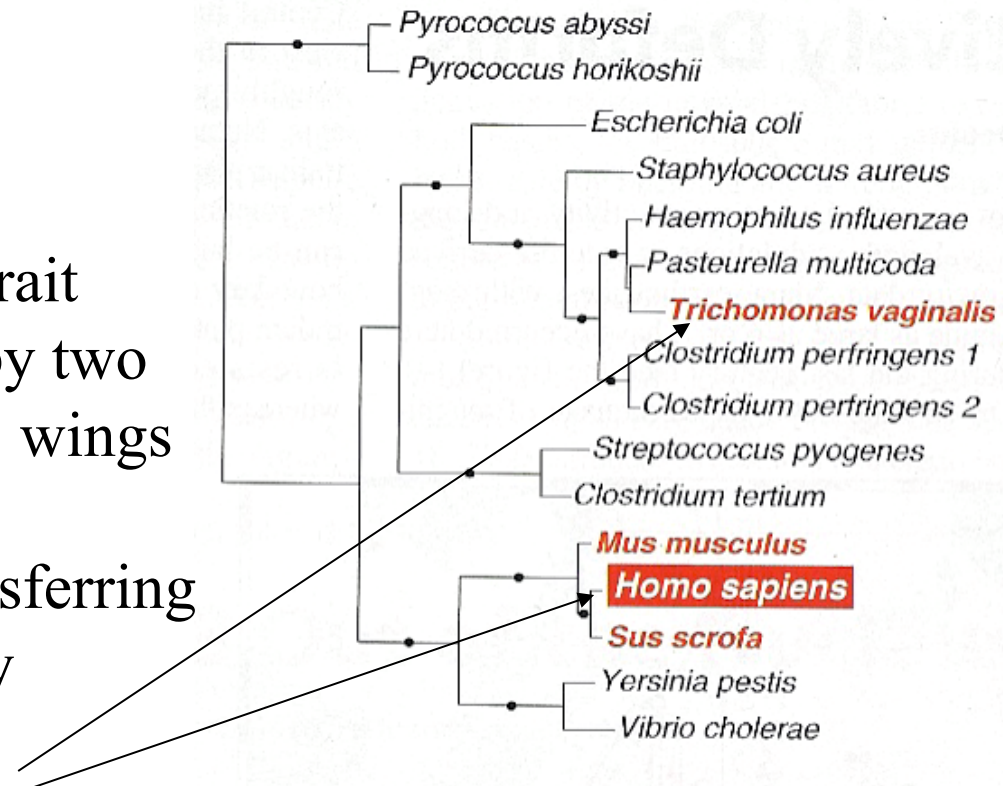


Infer nodes labels

Application of small parsimony problem

- errors in data
- loss of function
- convergent evolution (a trait developed independently by two evolutionary pathways e.g. wings in birds and bats)
- lateral gene transfer (transferring genes across species not by inheritance)

Red – gene encoding N-acetylneuraminidase lyase



Gene swapping among friends and neighbors. Phylogeny of the gene encoding N-acetylneuraminidase lyase (Ensembl ID IGL_M1_ctg1425_20). Phylogenetic relationships for this

From paper: Are There Bugs in Our Genome:
Anderson, Doolittle, Nesbo, Science 292 (2001) 1848-51

Dynamic programming algorithm for small parsimony problem

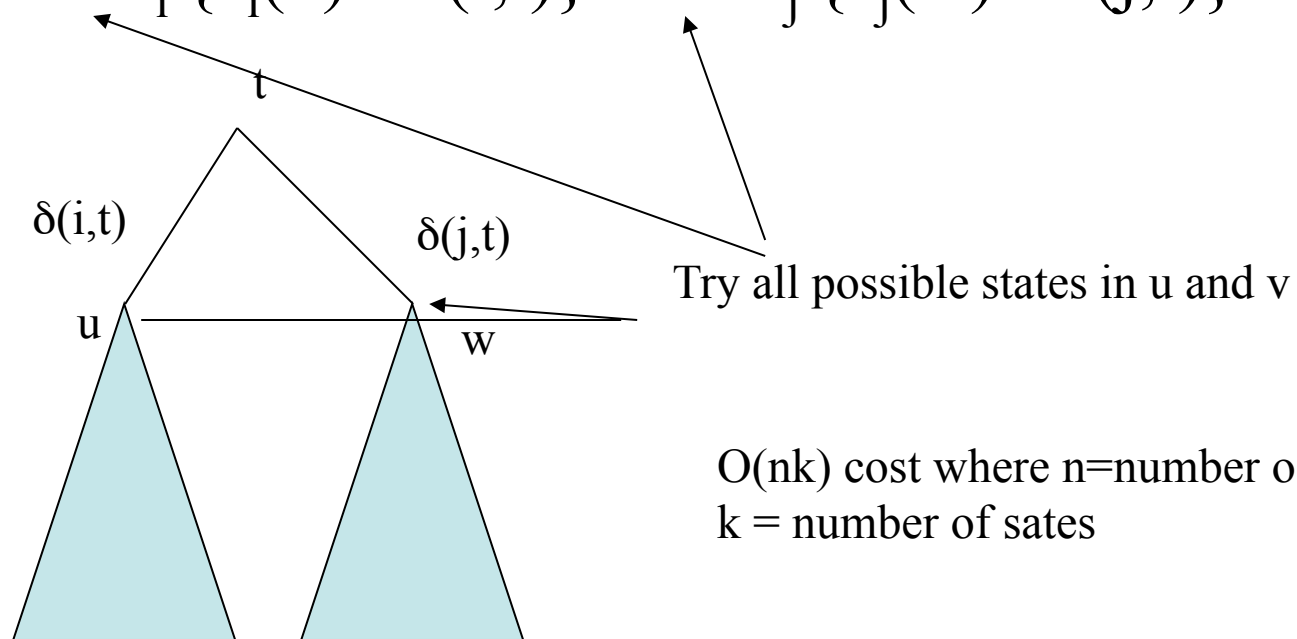
- Sankoff (1975) comes with the DP approach (Fitch provided an earlier non DP algorithm)
- Assumptions
 - one character with multiple states
 - The cost of change from state v to w is $\delta(v,w)$ (note that it is a generalization, so far we talk about cost of any change equal to 1)

DP algorithm continued

$s_t(v)$ = minimum parsimony cost for node v under assumption that the character state is t .

$s_t(v) = 0$ if v is a leaf. Otherwise let u, w be children of v

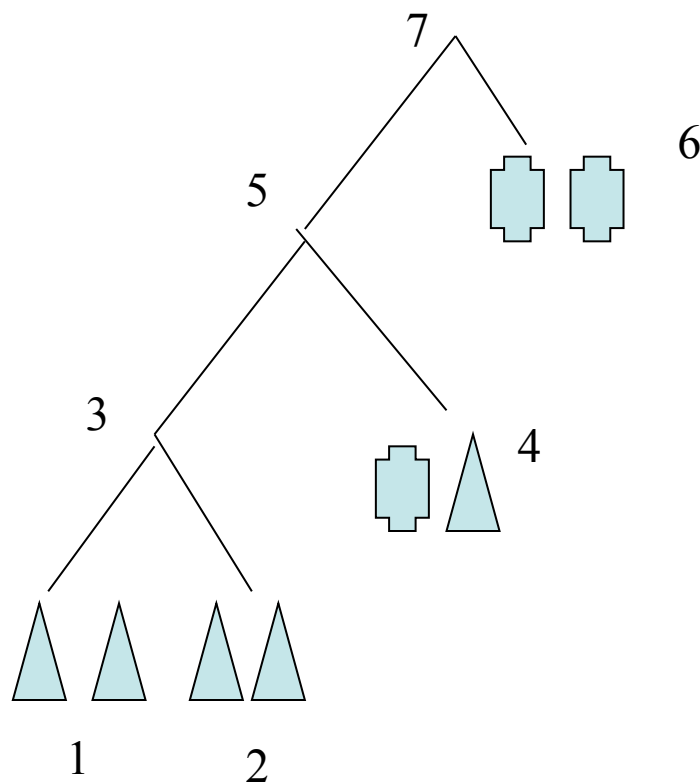
$$s_t(v) = \min_i \{s_i(u) + \delta(i,t)\} + \min_j \{s_j(w) + \delta(j,t)\}$$




$O(nk)$ cost where n = number of nodes
 k = number of states

Exercise

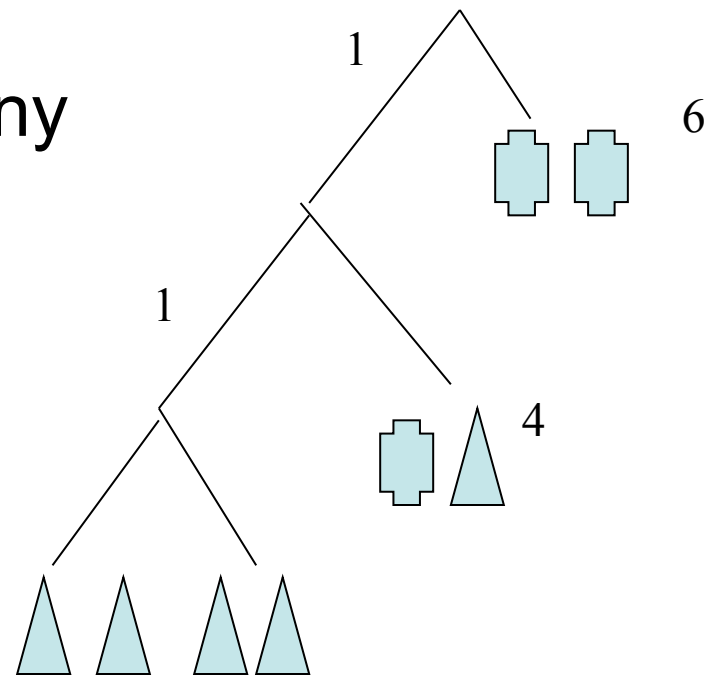
Left and right characters are independent,
We will compute the left.



$S_t(v)$	1	2	3	4	5	6	7
							
							

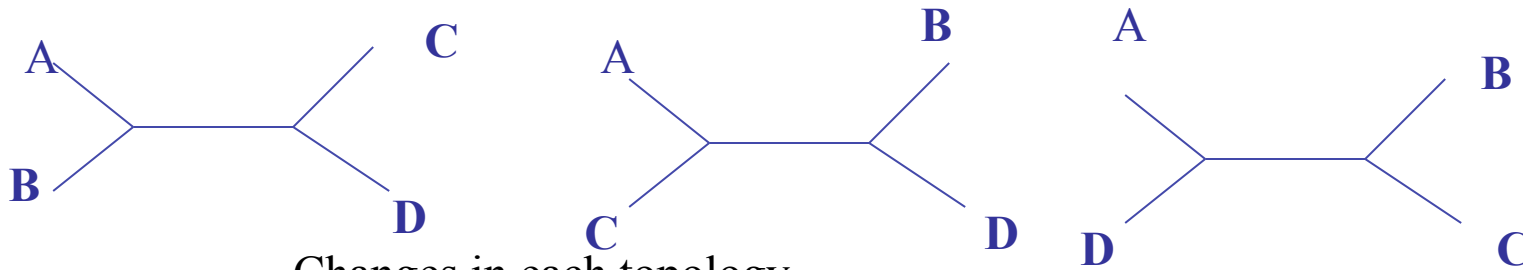
Branch lengths

- Numbers that indicate the number of changes in each branch
- Problem – there may be many most parsimonious trees
- Method 1: Average over all most parsimonious trees.
- Still a problem – the branch lengths are frequently underestimated



Character patterns and parsimony

- Assume 2 state characters (0/1) and four taxa A,B,C,D
- The possible topologies are:



A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1

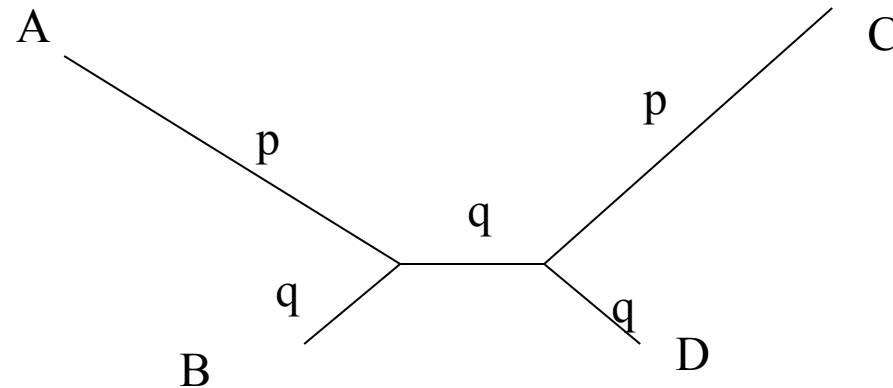
Changes in each topology

0, 0, 0
1, 1, 1
1, 1, 1
1, 2, 2

Informative character
(helps to decide the tree topology)

Informative characters: $xyyy$, $xyxy$, $xyyx$

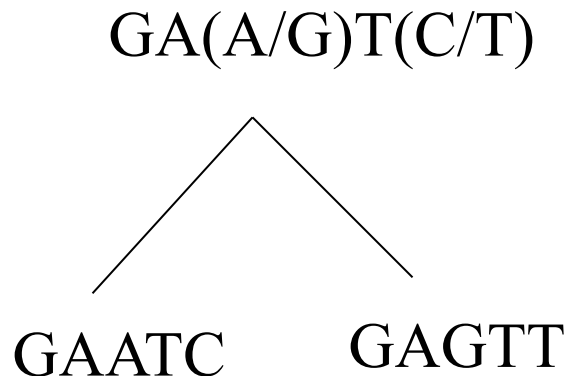
Inconsistency



- Let p , q character change probability
- Consider the three informative patterns $xyyy$, $xyxy$, $xyyx$
- The tree selected by the parsimony depends which pattern has the highest fraction;
- If $q(1-q) < p^2$ then the most frequent pattern is $xyxy$ leading to incorrect tree.

Distance based methods

- When two sequences are similar they are likely to originate from the same ancestor
- Sequence similarity can approximate evolutionary distances



Distance Method

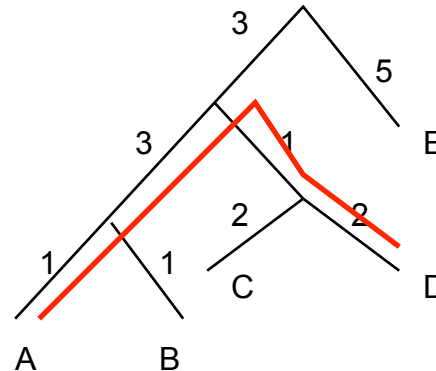
- Assume that for any pair of species we have an estimation of evolutionary distance between them
 - eg. alignment score
- Goal: construct a tree which best approximates these distance

Tree from distance matrix

M

	A	B	C	D	E
A	0	2	7	7	12
B	2	0	7	7	12
C	7	7	0	4	11
D	7	7	4	0	11
E	12	12	11	11	0

T



length of the path from A to D = 1+3+1+2=7

Consider weighted trees: $w(e)$ = weight of edge e

Recall: In a tree there is a **unique** path between any two nodes.

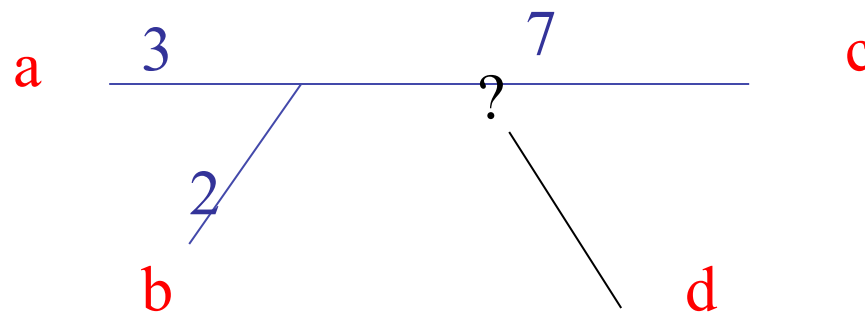
Let e_1, e_2, \dots, e_k be the edges of the path connecting u and v then the **distance between u and v in the tree** is:

$$d(u,v) = w(e_1) + w(e_2) + \dots + w(e_k)$$

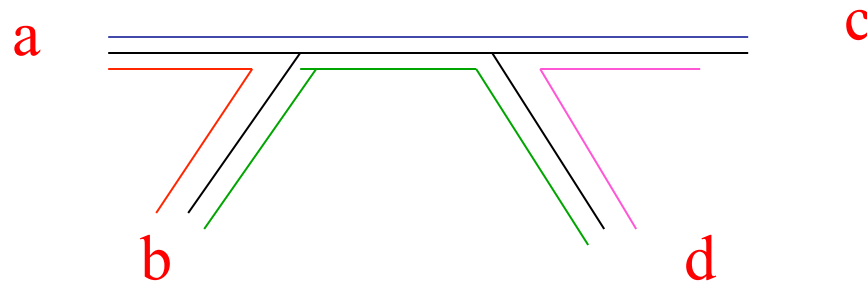
Can one always represent a distance matrix as a weighted tree?

a	0	10	5	10
b	10	0	9	5
c	5	9	0	8
d	10	5	8	0
	a	b	c	d

There is no way to add d to the tree and preserve the distances



Quadrangle inequality



$$d(a,c) + d(b,d) = d(a,d) + d(b,c) \geq d(a,b) + d(d,c)$$

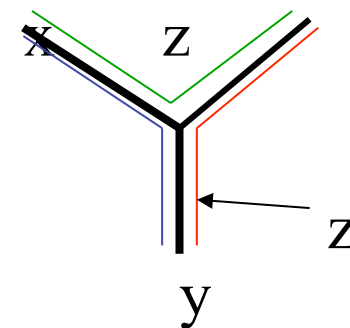
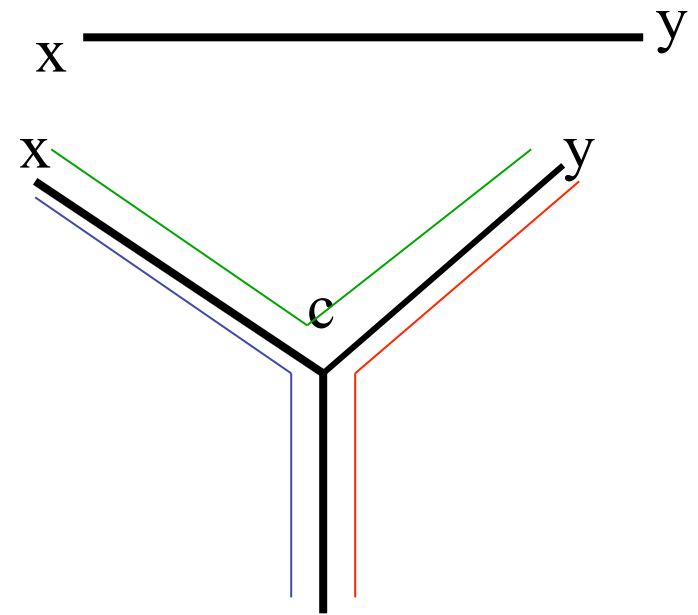
- Matrix that satisfies quadrangle inequality (called also the four point condition) for every four taxa is called **additive**.
- Theorem: Distance matrix can be represented **precisely** as a weighted tree if and only if it is additive.

Constructing the tree representing an additive matrix (one of several methods)

1. Start form 2-leaf tree a,b where a,b are any two elements
2. For $i = 3$ to n (iteratively add vertices)
 1. Take any vertex z not yet in the tree and consider 2 vertices x,y that are in the tree and compute
$$d(z,c) = (d(z,x) + d(z,y) - d(x,y)) / 2$$
$$d(x,c) = (d(x,z) + d(x,y) - d(y,z)) / 2$$
 2. From step 1 we know position of c and the length of brunch (c,z).

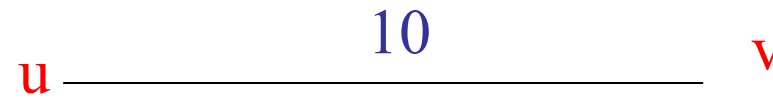
If c did not hit exactly a brunching point add c and z

else take as y any node from sub-tree that brunches at c and repeat steps 1,2.



Example

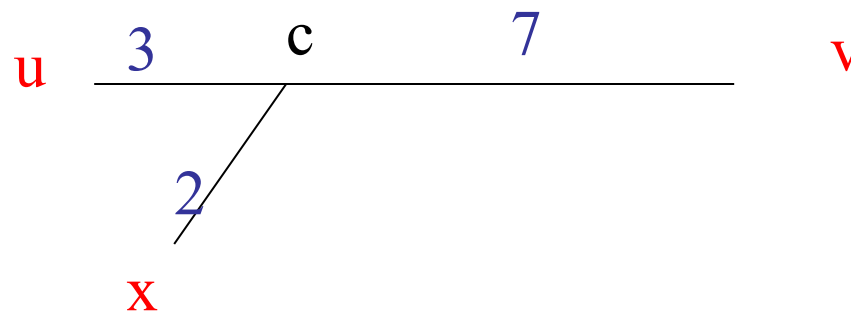
u	0	10	5	9
v	10	0	9	5
x	5	9	0	9
y	9	5	9	0
	u	v	x	y



Adding x:

$$d(x,c) = (d(u,x) + d(v,x) - d(u,v))/2 = (5+9-10)/2 = 2$$

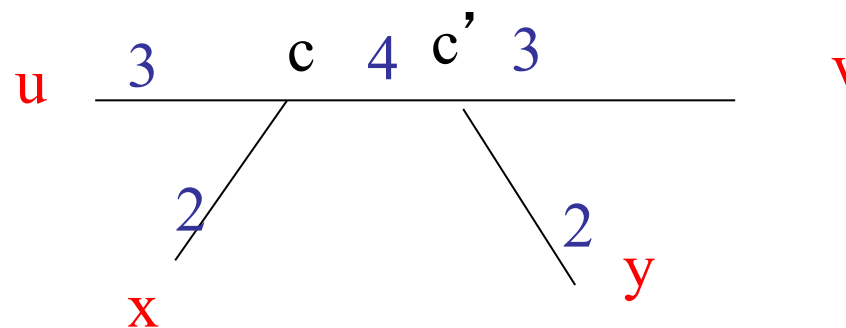
$$d(u,c) = (d(u,x) + d(u,v) - d(x,v))/2 = (5+10-9)/2 = 3$$



Adding y:

$$d(y,c') = (d(u,y) + d(v,y) - d(u,w))/2 = (5+9-10)/2 = 2$$

$$d(u,c') = (d(u,y) + d(u,v) - d(y,v))/2 = (10+9-5)/2 = 7$$



Real matrices are almost never additive

- Finding a tree that minimizes the error
Optimizing the error is hard
- Heuristics:
 - Unweighted Pair Group Method with Arithmetic Mean (UPGMA)
 - Neighborhood Joining (NJ)

Hierarchical Clustering

- **Clustering problem:** Group items (e.g. genes) with similar properties (e.g. expression pattern, sequence similarity) so that
 - The clusters are **homogenous** (the items in each cluster are highly similar, as measured by the given property – sequence similarity, expression pattern)
 - The clusters are well **separated** (the items in different clusters are different)
- **Hierarchical clustering** Many clusters have natural sub-clusters which are often easier to identify e.g. cuts are sub-cluster of carnivore sub-cluster of mammals
Organize the elements into a tree rather than forming explicit partitioning

The basic algorithm

Input: distance array d ; cluster to cluster **distance** function

Initialize:

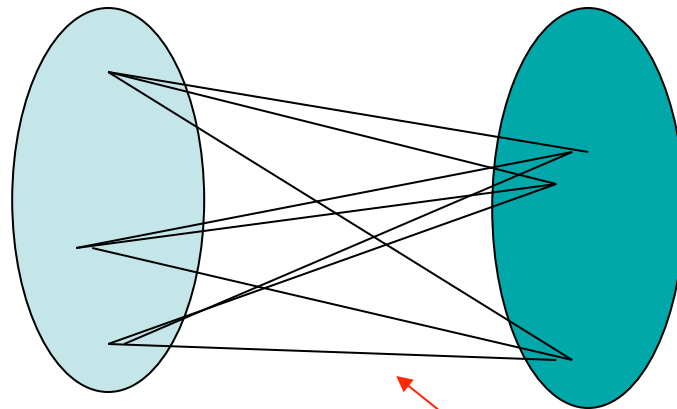
1. Put every element in one-element cluster
2. Initialize a forest T of one-node trees (each tree corresponds to one cluster)

while there is more than one cluster

1. Find two closest clusters C_1 and C_2 and merge them into C
2. Compute **distance** from C to all other clusters
3. Add new vertex corresponding to C to the forest T and make nodes corresponding to C_1 , C_2 children of this node.
4. Remove from d columns corresponding to C_1, C_2
5. Add to d column corresponding to C

A distance function

- $d_{\text{ave}}(C_1, C_2) = 1/(|C_1||C_2|) \sum_{x \in C_1, y \in C_2} d(x, y)$



Average over all distances

Example (on blackboard)

d

	A	B	C	D	E
A	0	2	7	7	12
B	2	0	7	7	12
C	7	7	0	4	11
D	7	7	4	0	11
E	12	12	11	11	0

Unweighted Pair Group Method with Arithmetic Mean (UPGMA)

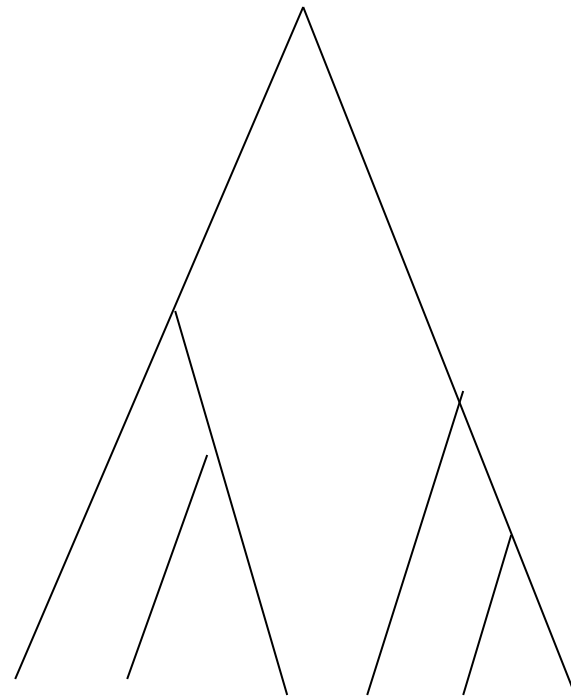
- Idea:
 - Combine hierarchical clustering with a method to put weights on the edges
 - Distance function used:

$$d_{\text{ave}}(C_1, C_2) = 1/(|C_1||C_2|) \sum_{\substack{x \text{ in } C_1 \\ y \text{ in } C_2}} d(x, y)$$

- We need to come up with a method of computing brunch lengths

Ultrametric trees

- The distance from any internal node C to any of its leaves is constant and equal to $h(C)$
- For each node (v) we keep variable h – height of the node in the tree. $h(v) = 0$ for all leaves.



UPGMA algorithm

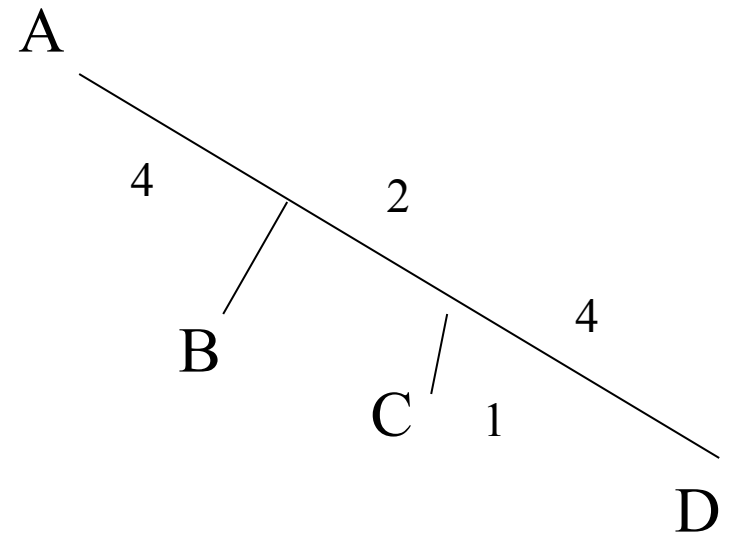
Initialization (as in hierarchical clustering); $h(v) = 0$
while there is more than one cluster

1. Find two closest clusters C_1 and C_2 and merge them into C
2. Compute d_{ave} from C to all other clusters
3. Add new vertex corresponding to C to the forest T and make nodes corresponding to C_1 , C_2 children of this node.
4. Remove from d columns corresponding to C_1, C_2
5. Add to d column corresponding to C
6. $h(C) = d(c_1, c_2) / 2$
7. Assign length $h(C) - h(C_1)$ to edge (C_1, C)
8. Assign length $h(C) - h(C_2)$ to edge (C_2, C)

Neighbor Joining

A	0	5	7	10
B	5	0	4	7
C	7	4	0	5
D	10	7	5	0

- Idea:
 - Construct tree by iteratively combing first nodes that are neighbors in the tree
- Trick: Figuring out a pair of neighboring vertices takes a trick – the closest pair want always do:
- B and C are the closest but are NOT neighbors.



Finding Neighbors

- Let $u(C) = 1/(\#\text{clusters}-2)\sum_{\text{all clusters } C'} d(C, C')$
- Find a pair C_1, C_2 that minimizes
$$f(C_1, C_2) = d(C_1, C_2) - (u(C_1) + u(C_2))$$
- Motivation: keep $d(C_1, C_2)$ small while $(u(C_1) + u(C_2))$ large

Finding Neighbors

- Let $u(C) = 1/(\#clusters-2) \sum_{\text{all clusters } C'} d(C, C')$

- Find a pair $C_1 C_2$ that minimizes

$$f(C_1, C_2) = d(C_1, C_2) - (u(C_1) + u(C_2))$$

- For the data from example:

$$u(C_A) = u(C_D) = 1/2(5+7+10) = 11$$

$$u(C_B) = u(C_C) = 1/2(5+4+7) = 8$$

$$f(C_A, C_B) = 5 - 11 - 8 = -14$$

$$f(C_B, C_C) = 4 - 8 - 8 = -12$$

NJ algorithm

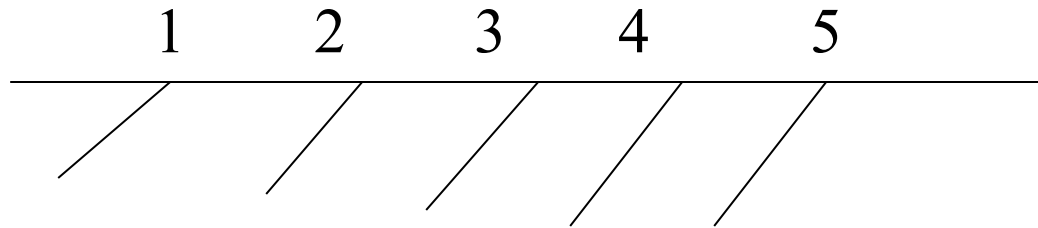
Initialization (as in hierarchical clustering); $h(v) = 0$

while there is more than one cluster

1. Find clusters C_1 and C_2 minimizing $f(C_1, C_2)$ and merge them into C
2. Compute for all C^* : $d(C, C^*) = (d(C_1, C^*) + d(C_2, C^*)) / 2$
3. Add new vertex corresponding to C to the forest T **and connect** it to C_1, C_2
4. Remove from d columns corresponding to C_1, C_2
5. Add to d column corresponding to C
6. Assign length $\frac{1}{2}(d(C_1, C_2) + u(C_1) - u(C_2))$ to edge C_1, C
7. Assign length $\frac{1}{2}(d(C_1, C_2) + u(C_2) - u(C_1))$ to edge C_2, C

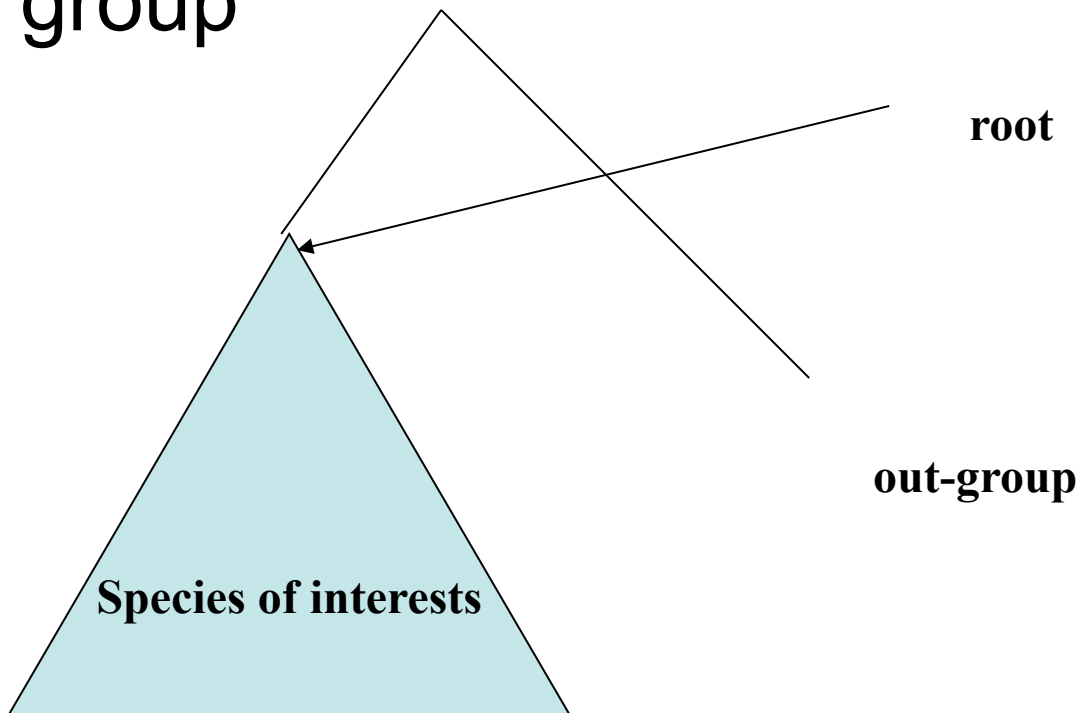
NJ tree is not rooted

The order of construction of internal nodes of NJ does not suggest an ancestral relation:



Rooting a tree

- Choose one distant organism as an out-group

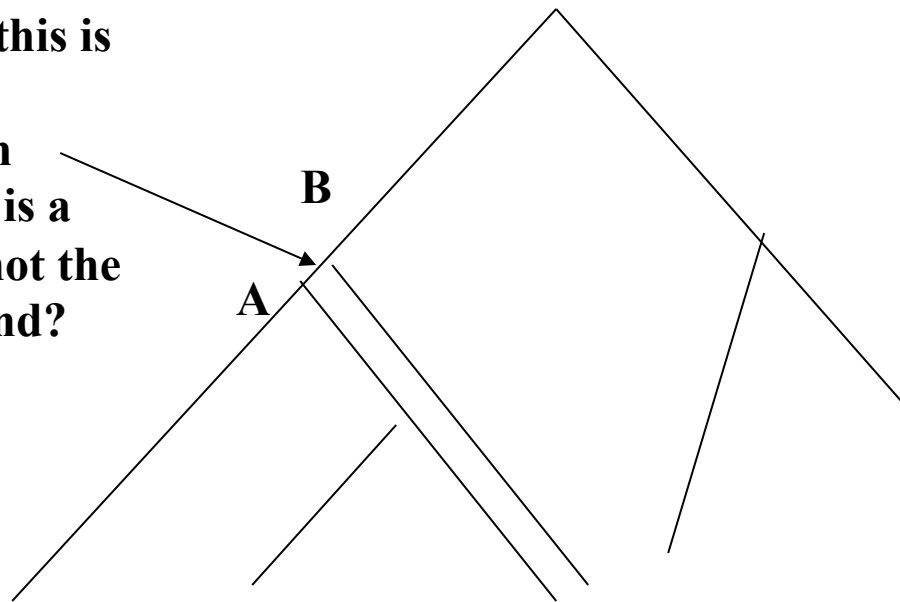


Bootstrapping

- Estimating confidence in the tree topology

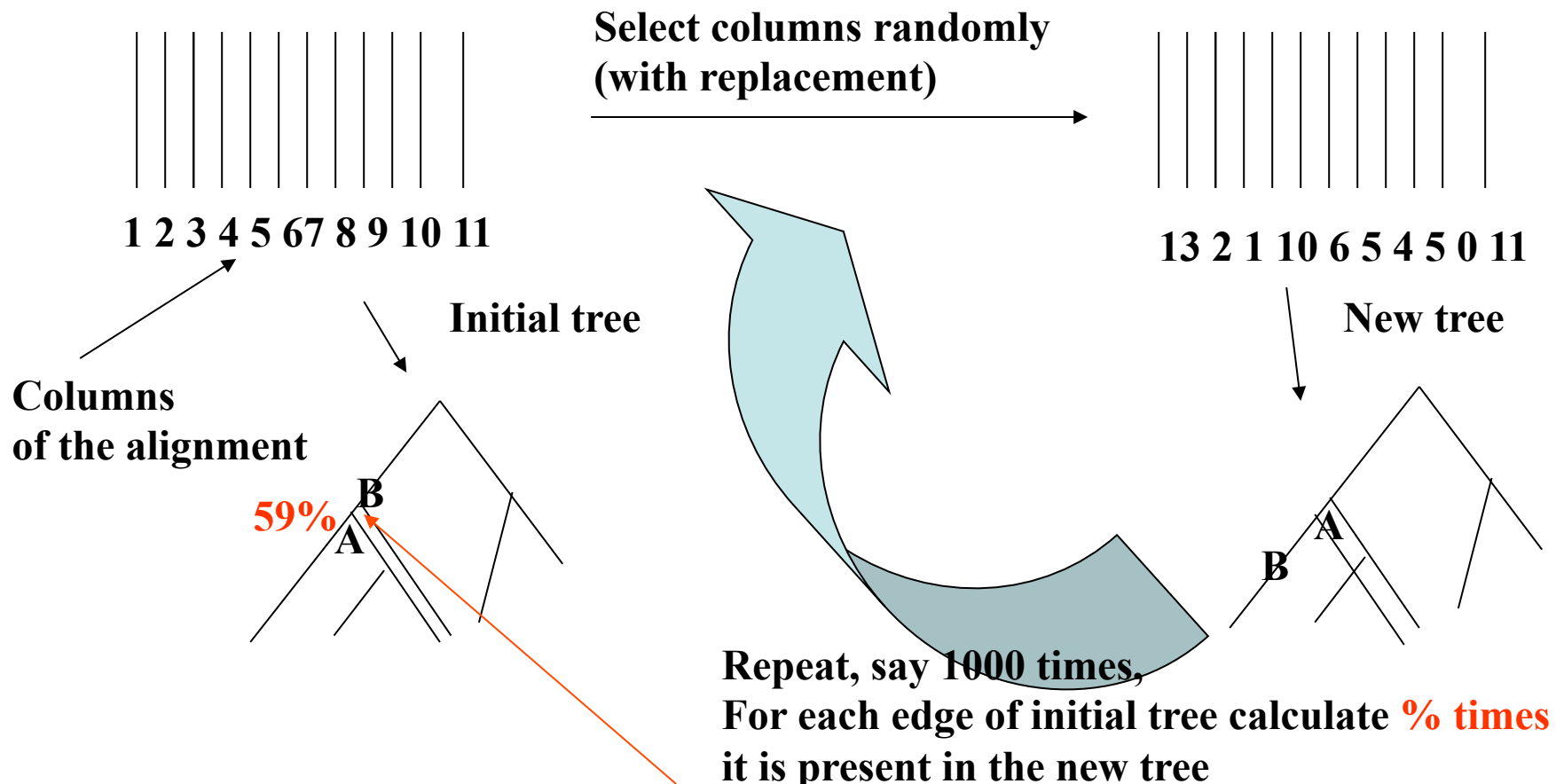
• Are we sure if this is correct?

• Is there enough evidence that A is a successor of B not the other way around?



Bootstrapping, continued

- Assume that the tree is build form multiple sequence alignment



Summary

- Assume you have multiple alignment of length N . Let T be the NJ tree build from this alignment
- Repeat, say 1000 times the following process:
 - Select randomly with replacement N columns of the alignment to produce a randomized alignment
 - Build the tree for this randomized alignment
- For each edge of T report % time it was present in a tree build from randomized alignment . This is called the **bootstrap value**.
- Trusted edges: 80% or better bootstrap.

Maximum Likelihood Method

- Given is a multiple sequence alignment and probabilistic model of for substitutions (like PAM model) **find the tree** which has the highest probability of generating the data.
- Simplifying assumptions:
 - Positions involved independently
 - After species diverged they evolve independently.

Formally:

- Find the tree T such that assuming evolution model M

$\Pr[\text{Data} | T, M]$ is maximized

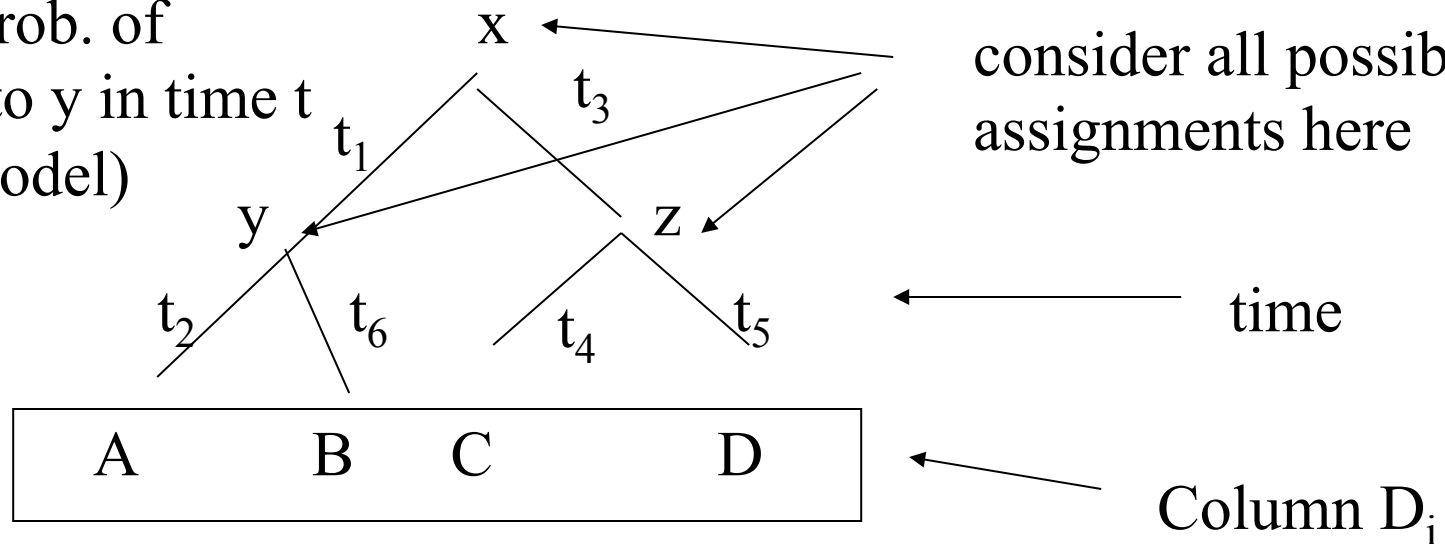
- From the independence of symbols:

$$\Pr[\text{Data} | T, M] = \prod_i \Pr[D_i | T, M]$$

Where the product is taken over all characters i
and D_i value of the character i is over all taxa

Computing $\Pr[D_i | T, M]$

$p(x,y,t)$ = prob. of mutation x to y in time t
(from the model)



$\Pr[D_i | T, M]$

$$= \sum_x \sum_y \sum_z$$

$$p(x)p(x,y,t_1)p(y,A,t_2)p(y,B,t_6)p(x,z,t_3)p(z,C,t_4)p(z,D,t_5)$$

Discovering the tree of life

- “Tree of life” – evolutionary tree of all organisms
- Construction: choose a gene universally present in all organisms; good examples: small rRNA subunit, mitochondrial sequences.
- Things to keep in mind while constructing tree of life from sequence distances:
 - Lateral (or horizontal) gene transfer
 - Gene duplication: genome may contain similar genes that may evolve using different pathways. Phylogeny tree need to be derived based on **orthologous** genes.

Where we go with it...

- We now how to compute for given column and given tree $\Pr[D_i | T, M]$
- Sum up over all columns to get
 $\Pr[\text{Data} | T, M]$

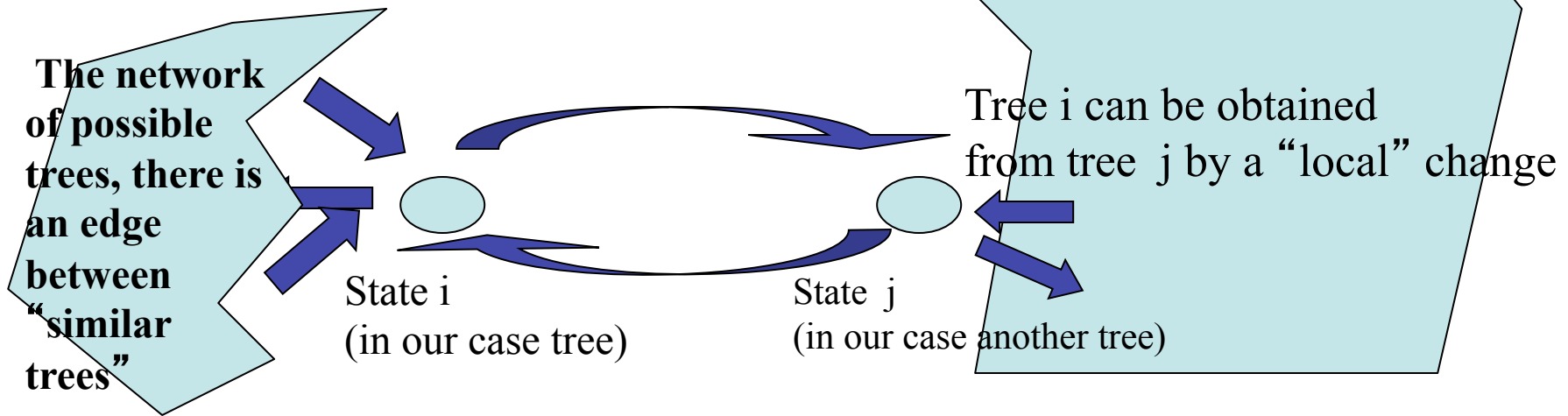
Now, explore the space of possible trees

Problem:

- Bad news: the space of all possible trees is HUGE

Various heuristic approaches are used.

Metropolis algorithm: Random Walk in Energy Space



Goal design transition probabilities so that the probability of arriving at state j is

$$P(j) = q(j) / Z$$

(typically, $q(S) = e^{-E(S)/kT}$, where E – energy)

Z – partition function = sum over all states S of

terms $q(s)$. Z cannot be computed analytically since the space is too large

temperature
const.

Monte Carlo, Metropolis Algorithm

- At each state i choose uniformly at random one of neighboring conformations j .
- Compute $p(i,j) = \min\left(1, q(i)/q(j)\right)$
- With probability $p(i,j)$ move to state j .
- Iterate

MrBayes

- Program developed by J.Huelsensbeck & F.Ronquist.
- Assumption:
- $q(i) = \text{Pr}[D | T_i, M]$
Prior probabilities: All trees are equally likely.
- Proportion of time a given tree is visited approximates posterior probabilities.

Most Popular Phylogeny Software

- PAUP
- PHYLIP