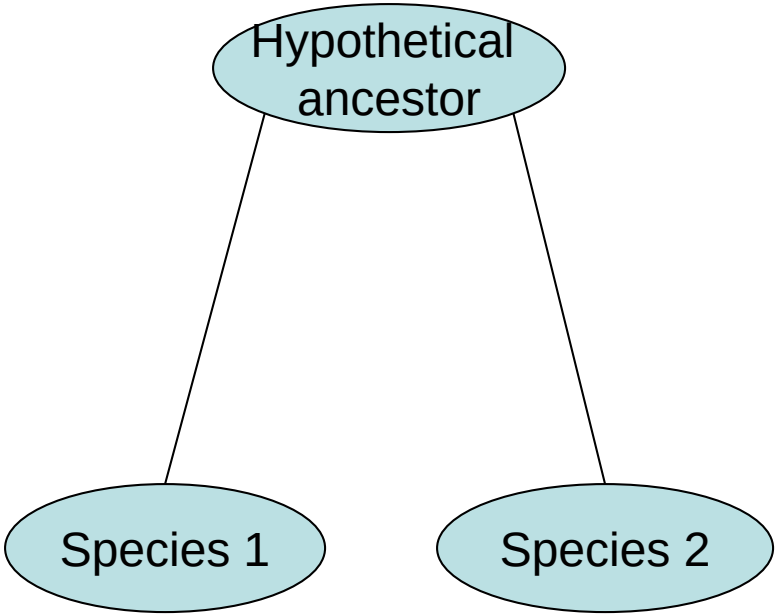
Parsimony and phylogenetic trees

Lecture 15

Phylogenetics

- Inferring phylogenies from a given data set



Two approaches to inferring phylogenies

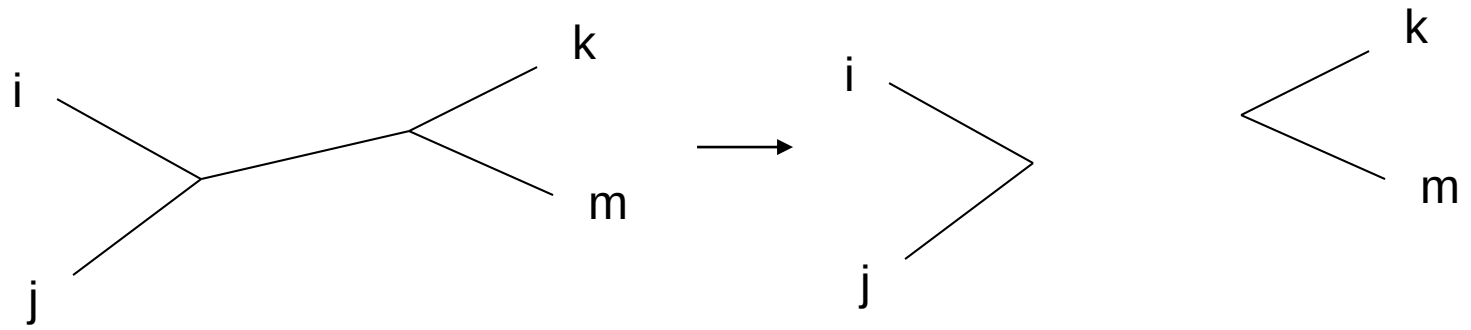
- Distance-based: for example paiwise edit distance, score of pairwise alignment etc.
- Character-based: examine each character separately in a given site of a biological sequence

Distance-based - recap

- Input: distance matrix of pairwise distances for N species
- Goal: find a tree consistent with the distance matrix. This means that the sum of edge lengths connecting each pair of leaves *ij* corresponds to a distance M_{ij}

Additivity of distances

- For any 4 objects, i,j,k,m, there are 3 different sums of 2 distances each:
- Dij+Dkm, Dik+Djm, Dim+Djk
- From these 3 sums, 2 should be equal and greater than the third – this will allow to group the smallest in between 3 into a separate subtree

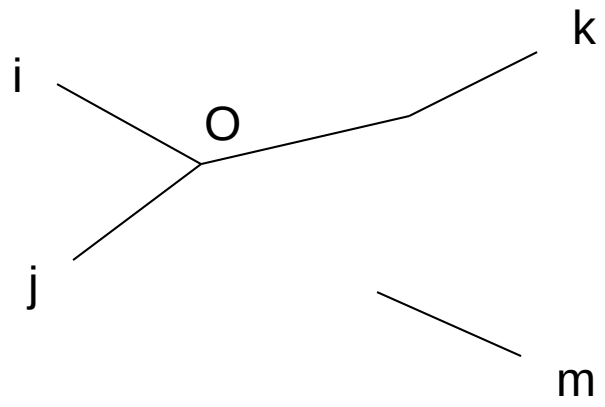


The distances are additive iff for any 4 objects there exits the following combination: Dij+Dkm<(Dim+Djk=Dik+Djm)

Why the distances have to be additive

For 3 objects, any set of distances is OK:

Let iO=a, jO=b, kO=c, then Dij=a+b, Dik=a+c, Djk=b+c

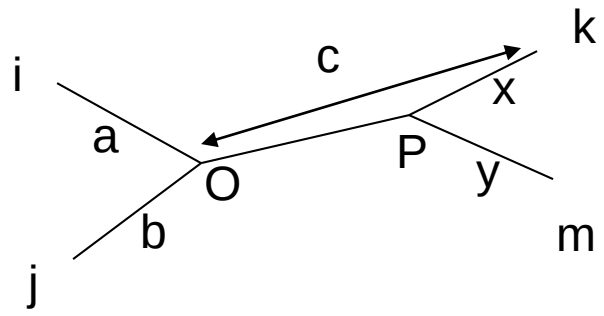


Why the distances have to be additive

Let iO=a, jO=b, kO=c

Dij=a+b, Dik=a+c, Djk=b+c

If we are adding the fourth object, m, to an arbitrary position P in the tree, and let mP=y, and kP=x, then OP=c-x



Then:

Dim+Djk=a+c-x+y + b+c=a+b+y+(2c-x)

Dik+Djm=a+c + b+c-x+y=a+b+y+(2c-x)

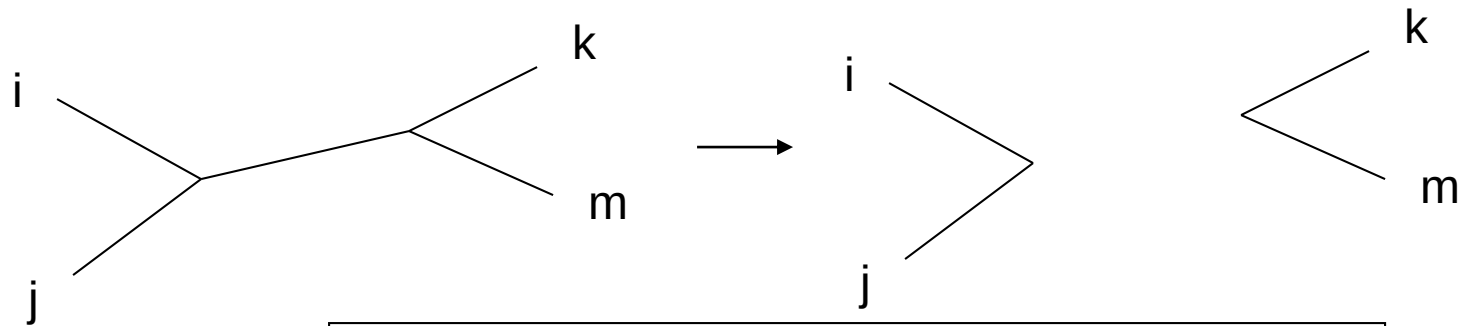
and

Dij+Dkm=a+b + x+y=a+b+y +x

2c-x>=x, since 2c>=2x

The rule of additivity

- The distances are additive if, for any 4 objects, i,j,k,m,



Dij+Dkm<(Dim+Djk=Dik+Djm)

If the distances are not additive, we CANNOT construct a phylogenetic tree

UPGMA algorithm - summary

- Initialization:
  - Create N clusters, 1 species per cluster
  - Set the size of each cluster to 1
  - Create leaf for each cluster
- Iteration (until only 1 cluster left)
  - Find Ci and Cj with min dCiCj
  - Create a new cluster C(ij) which has n(ij)=ni+nj members
  - Connect Ci and Cj through a new parent node and set the distance from this new parent node to the leaf node of each cluster to ½ dCiCj
  - Delete columns and rows that correspond to amalgamed clusters i and j
  - Add a column and a row for a new cluster
  - Compute distances from a new cluster C(ij) to all remaining clusters:

$$dC(ij)Ck=\Sigma_{\text{all } x\in C(ij),\ \text{all } y\in Ck}\ dXY/(n(ij)*nk)$$

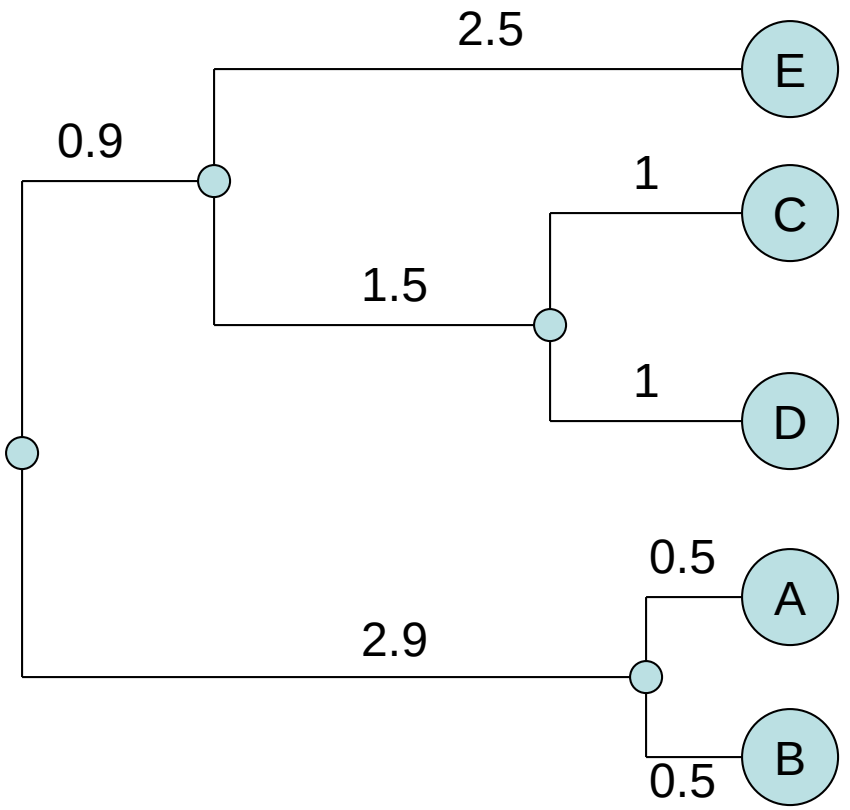
What was the goal?

- Input: distance matrix of pairwise distances for N species
- Goal: find a tree consistent with the distance matrix. This means that **the sum of edge lengths connecting each pair of leaves ij corresponds to a distance Mij**

UPGMA tree – consistent with M?

	A	B	C	D	E
A	0				
B	1	0			
C	4	3	0		
D	8	7	2	0	
E	10	9	4	6	0

	A	B	C	D	E
A	0				
B	1	0			
C	6.8	6.8	0		
D	6.8	6.8	2	0	
E	6.8	6.8	5	5	0



This was caused by averaging distances between elements of the clusters

This would not happen if the molecular clock had constant speed over all branches of the tree

Less ambitious goal

- Find the tree which predicts the set of distances as closely as possible

$$SSQ(T)=\Sigma_{i \text{ from } 1 \text{ to } N}\Sigma_{j\neq i}w_{ij}(d_{ij}-TreeD_{ij})$$

dij – input distance (value Mij in the distance matrix)

wij – weight which intuitively quantifies the accuracy of distances

Tree-Dij – distance between leaf i and leaf j in the tree (sum of edge lengths)

The least squares method for fitting the function to the experimental curve

Distance-based phylogeny

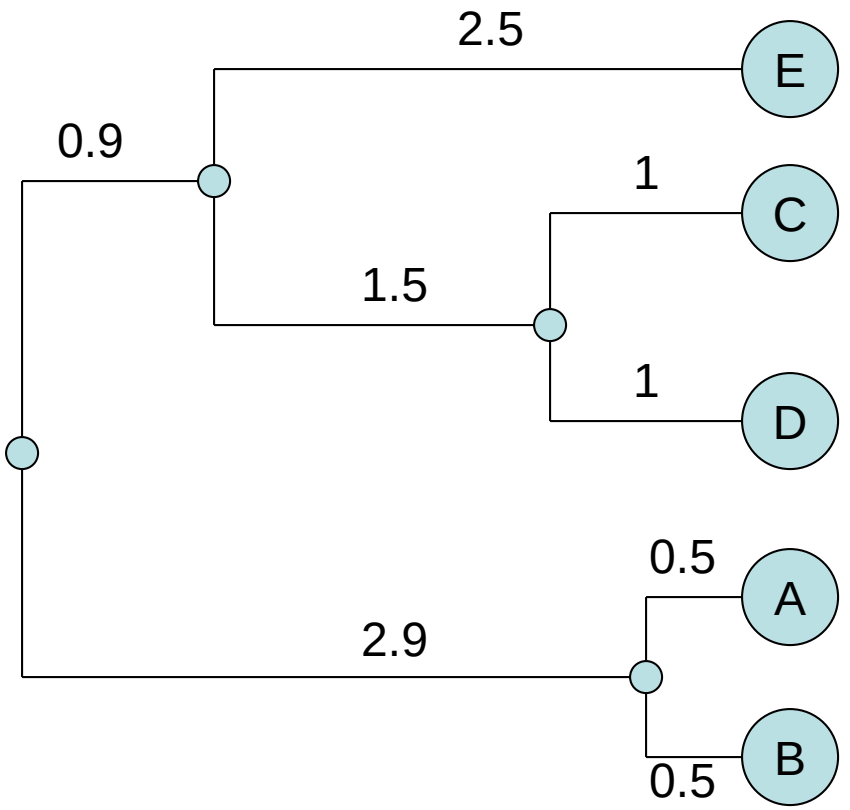
$$SSQ(T)=\Sigma_{i \text{ from } 1 \text{ to } N}\Sigma_{j\neq i} w_{ij}(d_{ij}-TreeD_{ij})$$

- Small problem: the tree is given, minimize the above expression
- Large problem: build a tree - which minimizes SSQ - from scratch (NP-complete)

Can we improve this tree?

	A	B	C	D	E
A	0				
B	1	0			
C	4	3	0		
D	8	7	2	0	
E	10	9	4	6	0

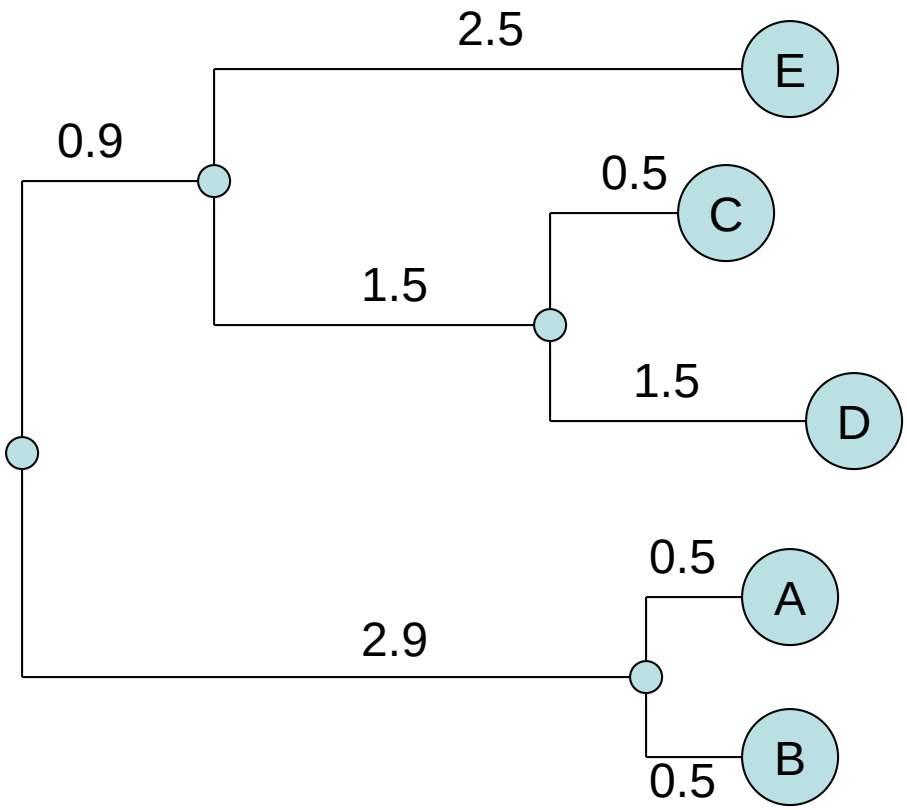
	A	B	C	D	E
A	0				
B	1	0			
C	6.8	6.8	0		
D	6.8	6.8	2	0	
E	6.8	6.8	5	5	0



Small problem has a solution

	A	B	C	D	E
A	0				
B	1	0			
C	4	3	0		
D	8	7	2	0	
E	10	9	4	6	0

	A	B	C	D	E
A	0				
B	1	0			
C	6.8	6.8	0		
D	6.8	6.8	2	0	
E	6.8	6.8	4.5	5.5	0



We can optimize the distances in the tree as much as we want, by redistributing the length between sibling leaves

Ultrametric trees and UPGMA

- The tree of slide 15 is clocklike, ultrametric: the total length of the path from a given internal node to each leaf is the same.
- The assumption: the molecular clock of mutations ticks with a constant pace
- UPGMA reconstructs the tree based on this molecular clock assumption, that is why a new node is always created at the same distance from all the leaves

When the tree reflects reality

$$SSQ(T)=\Sigma_{i \text{ from 1 to } N}\Sigma_{j\neq i}w_{ij}(d_{ij}-TreeD_{ij})$$

- If the solution to SSQ(T)=0, and there was a molecular clock with constant pace, then UPGMA guarantees to find an optimal solution, if not:
  - It can find a good enough solution, but the correctness of the tree topology is not guaranteed
  - Use *the neighbor-joining* algorithm to check the correctness of the tree topology. This algorithm relies on additivity but does not require the distances to be ultrametric

Test for ultrametric condition

- We can predict whether the reconstruction of the real tree is likely to be correct by testing our distances for *ultrametric condition*:

The distance matrix is *ultrametric* if for any triplet of sequences, Xi, Xj, Xk, the distances dij, dik, djk are either all equal or two are equal and the remaining one is smaller

Thus, if distances were derived from a real tree with a molecular clock, the distance matrix has to be ultrametric

Ultrametric and non-ultrametric distance matrices

	A	B	C	D
A	0			
B	1	0		
C	4	2	0	
D	8	7	5	0

dAB=1, dAC=4, dBC=2

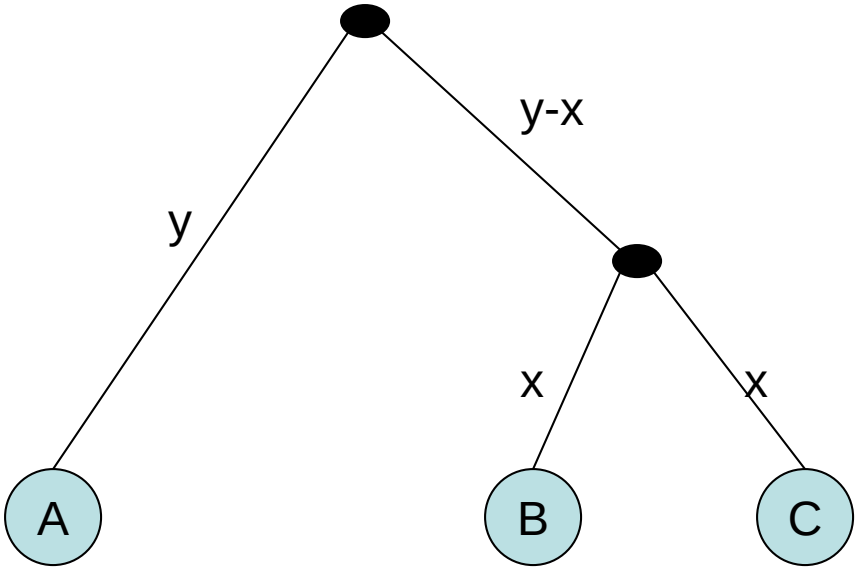
Non-ultrametric matrix

	A	B	C	D
A	0			
B	4	0		
C	2	4	0	
D	8	8	8	0

Ultrametric matrix

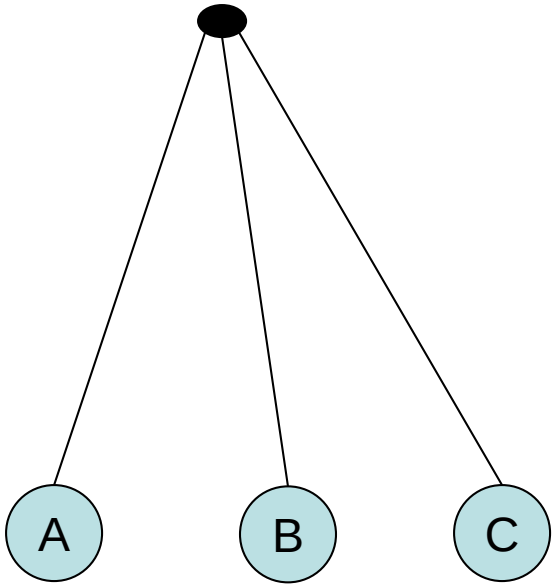
Ultrametric trees

AB=AC>BC



or

AB=AC=BC



AB=y+y-x+x=2y

AC=2y

BC=2x, x<=y, since y-x.=0 (no negative edge lengths)

The rule for ultrametric trees:

2 out of 3 distances have a tie, and are >= the third distance

End of part 1: distance-based phylogenies

Part 2. Character-based (parsimony based) phylogenies

Parsimony

- In science, *parsimony* is preference for the least complex explanation. This is regarded as good when judging hypotheses.
- Occam's razor also states the "principle of parsimony“: *entia non sunt multiplicanda praeter necessitatem*, is the principle that "entities must not be multiplied beyond necessity": the simplest explanation or strategy tends to be the best one
- Under maximum parsimony, **the preferred phylogenetic tree is the tree that requires the smallest number of evolutionary changes**.

Evaluating the tree: small parsimony problem

- Under the maximum parsimony, we seek the tree which has the fewest total number of mutational events along its branches
- Finding the most parsimonous tree is the NP-complete problem

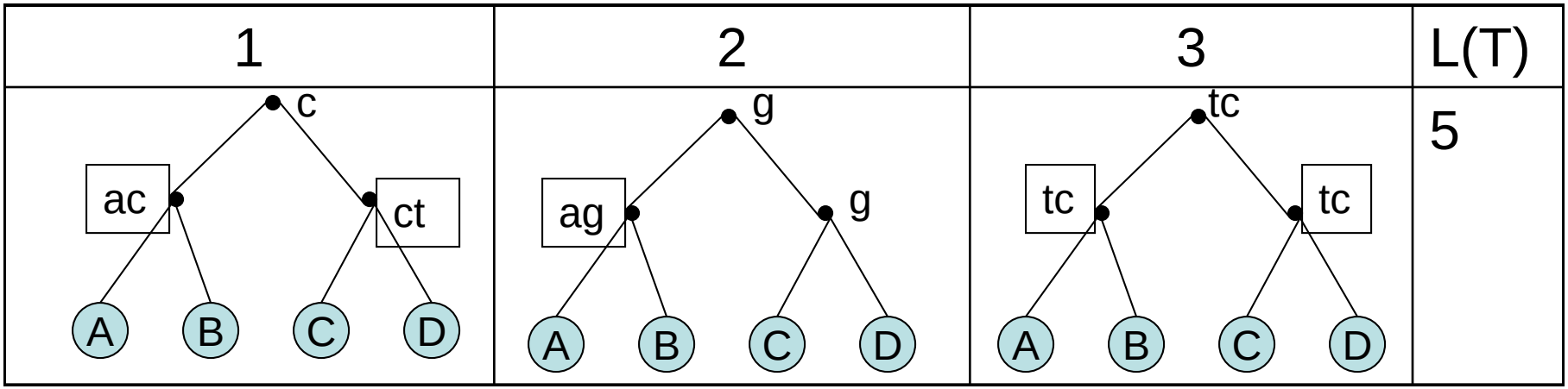
The Fitch algorithm for evaluating the tree given the multiple alignment

	1	2	3
A	a	a	t
B	c	g	c
C	c	g	c
D	t	g	t

Input: multiple alignment and the suggested tree topology

Output: the “length” of the tree – the total number of mutational events which corresponds to a given tree

T1



Perform DFS for each position and label each parent node by an intersection of its children. If this intersection is empty, label parent by the union of its children

In case of an empty intersection, L(T) will be increased by 1

Character-based phylogeny problem with parsimony score

- Input:
  - a set of *N* species
  - a set of *M* characters for each species
  - The input is generally presented as an *NxM* matrix $\boldsymbol{C}$, where each entry $\boldsymbol{C}_{ij}$ represents the value of character *j* for species *i*
  - In addition, the weight matrix may be supplied to weight the score of transition between different characters

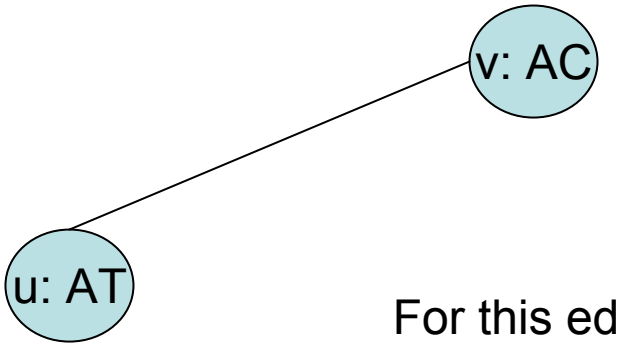
The parsimony score of the tree. Intuition

- The most parsimonous explanation of a given phylogeny is the tree with an overall minimum number of changes along its branches
- The logic is the basic philosophy of Ockham's razor – finding the simplest explanation that works
- The score is the total number of times the value of some character changes along some edge

The parsimony score of the tree. Definition

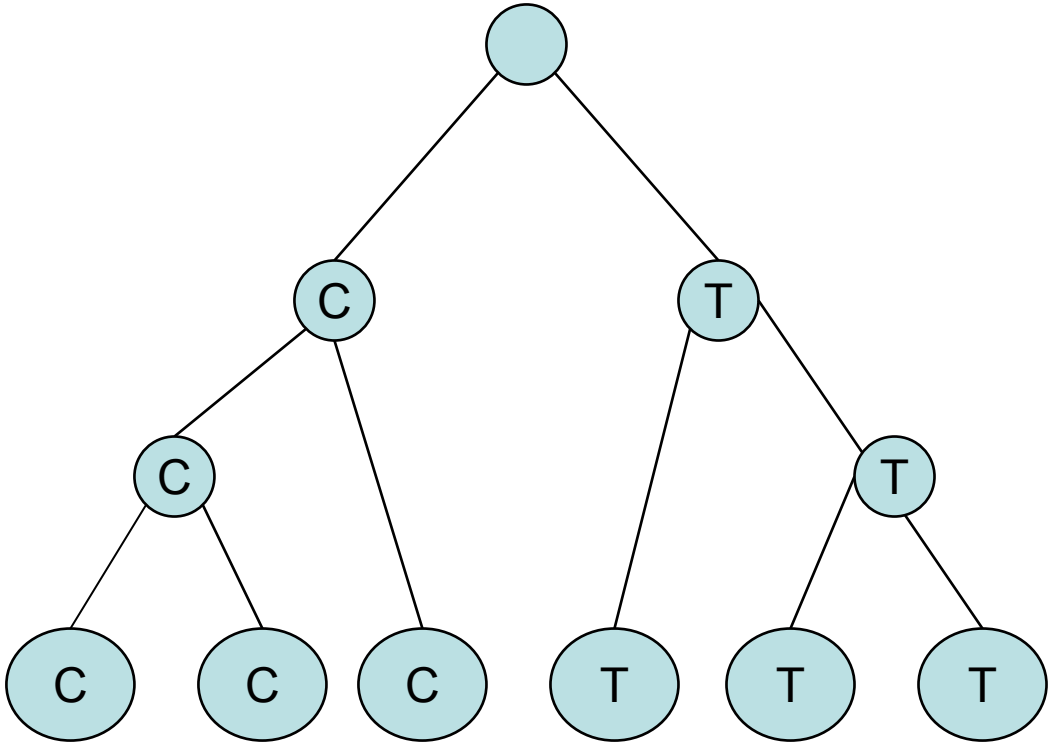
- Let V(T) be a set of vertices and E(T) be a set of edges of a given phylogenetic tree, and let the value of a character j in vertex v be vj.

$$PScore(T)=\Sigma_{(\text{for each } v,u \in E(T))} |\{j: vj \neq uj\}|$$



For this edge, we add 1 to the total parsimony score of the tree

Pscore example



What is Pscore of this tree?

Parsimony problems

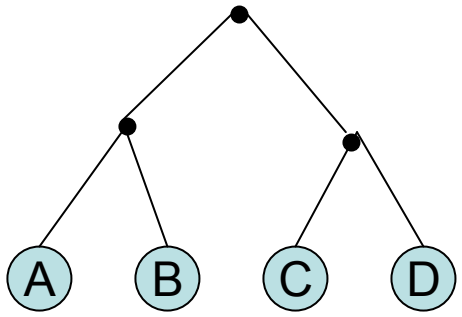
- Small parsimony problem: given a topology of a rooted phylogenetic tree and the character matrix C, find a labeling of ancestral sequences which implies the minimum parsimony score
- Large parsimony problem: given the character matrix C, build the tree with the minimum parsimony score (minimum number of character changes along its edges) – NP-hard

Assumptions

- The character changes are mutually independent
- After 2 species diverged, they continue to evolve separately
- Each character split is a 2-way split – bifurcating (binary) tree

The Fitch algorithm for the small parsimony problem

	1	2	3
A	a	a	t
B	c	g	c
C	c	g	c
D	t	g	t



Input:

matrix C (multiple alignment)

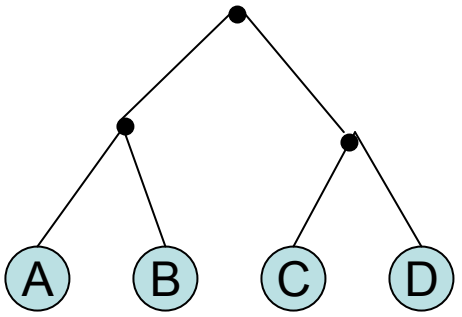
tree T

Output:

Labeling of ancestral nodes which minimizes the parsimony score of the tree

The Fitch algorithm. Phase I

	1	2	3
A	a	a	t
B	c	g	c
C	c	g	c
D	t	g	t



Compute the sets of all possible character states for each internal node, based on the states of its children

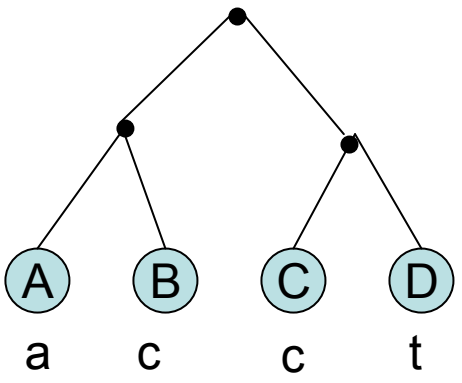
Each leaf node contains only 1 state for each character, and is initially marked with this character

Perform post-order traversal of the tree (each node is evaluated only after all its children have been evaluated) and for each node *v* compute the candidate character set

The Fitch algorithm. Phase I

	1	2	3
A	a	a	t
B	c	g	c
C	c	g	c
D	t	g	t

j=1



Phase I.

For each column *j* of matrix ***C***:

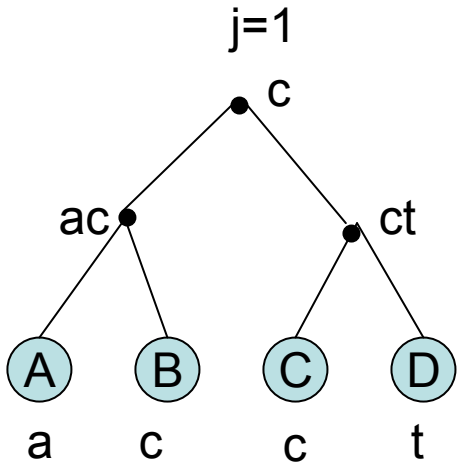
//initialize

for each leaf node *v* of species *i*:

$Set_v = \boldsymbol{C}_{ij}$

The Fitch algorithm. Phase I

	1	2	3
A	a	a	t
B	c	g	c
C	c	g	c
D	t	g	t



Phase I.

For each column *j* of matrix ***C***:

//initialize

for each leaf node *v*:

$Set_v = \boldsymbol{C}_{ij}$

perform post-order traversal of *T*

for each internal node *v*

with children *u* and *w*:

if $Set_u \cap Set_w \neq \emptyset$

$Set_v = Set_u \cap Set_w$

else

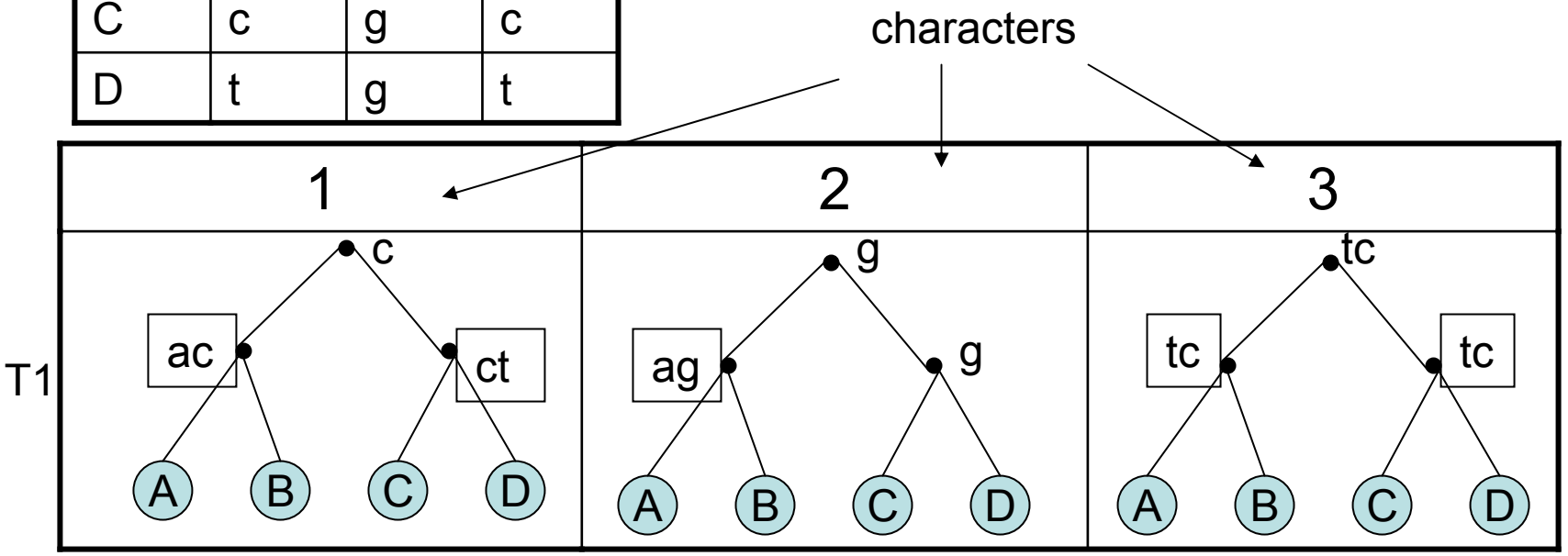
$Set_u \cup Set_w$

The Fitch algorithm. Phase I intuition

- If there is a state which fits both children, we take it as their common ancestor. If there is no such state (the intersection is empty), we take as the candidates the sets of its children, since this is better than taking any other set which does not occur in any of the children

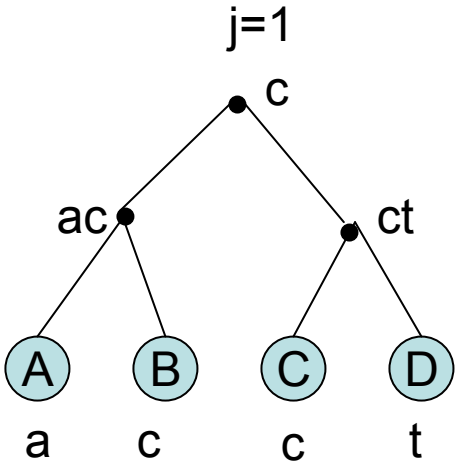
The Fitch algorithm phase I example

	1	2	3
A	a	a	t
B	c	g	c
C	c	g	c
D	t	g	t



The Fitch algorithm. Phase II

	1	2	3
A	a	a	t
B	c	g	c
C	c	g	c
D	t	g	t



We determine value v_j to assign to each internal node, which we choose from the candidate set of characters

We perform pre-order traversal (each child is evaluated after its parent has been evaluated)

The Fitch algorithm. Phase II

	1	2	3
A	a	a	t
B	c	g	c
C	c	g	c
D	t	g	t

For each character *j*

perform pre-order traversal of *T*

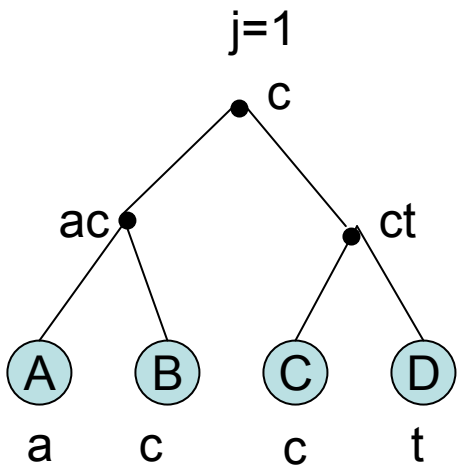
for each vertex *v* with parent *u*

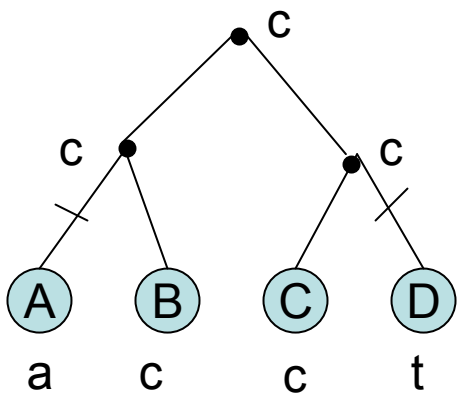
if u_j € Set_v

$v_j=u_j$

else

v_j=any element from Set_v



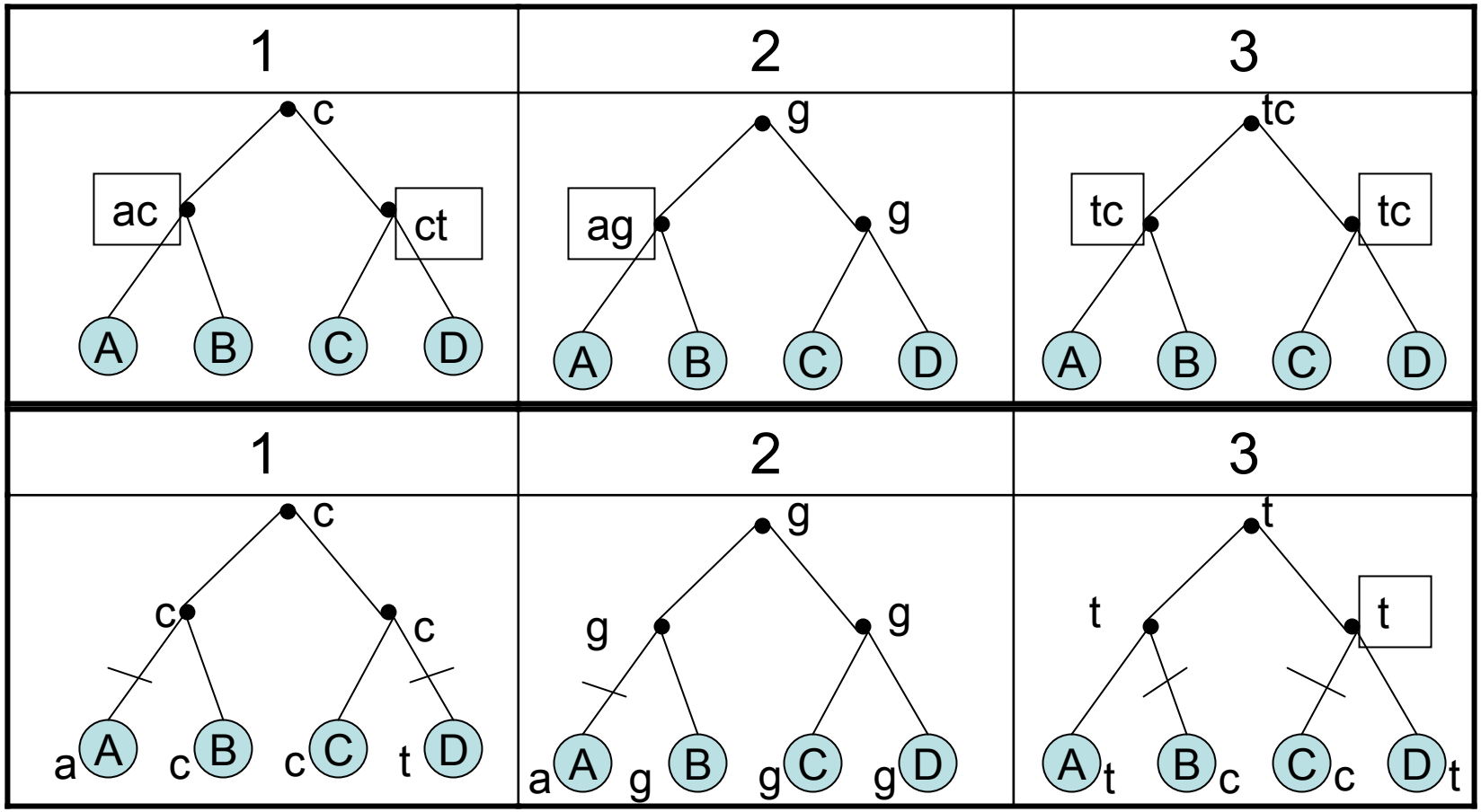


The parsimony score for the first character=2

The Fitch algorithm phase II example

	1	2	3
A	a	a	t
B	c	g	c
C	c	g	c
D	t	g	t

The total parsimony score of this tree with this optimal labeling is 5



The Fitch algorithm. Time complexity

- If there are k possible values of a character, then for a single character we perform at most $2k$ operations, and with $2N$ total nodes in the tree, there is $O(Nk)$ work for a single character
- $O(NMk)$ for all M characters

The weighted parsimony. The algorithm by Sankoff

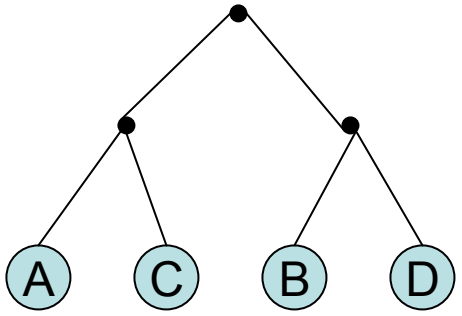
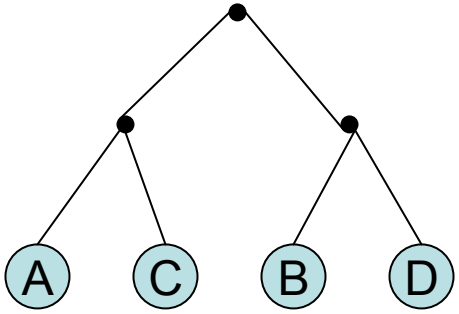
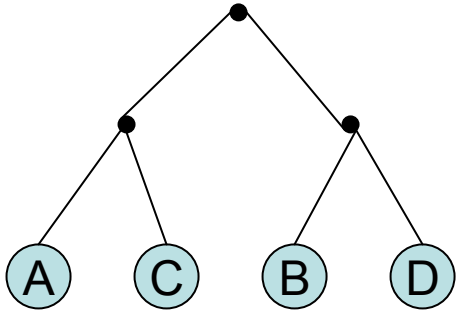
- Different, application-specific costs are assigned for each change of character from state to state
- This is more realistic, since substitutions happen with different probabilities
- An overall algorithm is similar to the Fitch algorithm (you can read it in your textbook)

Large Parsimony problem for 4x3 matrix

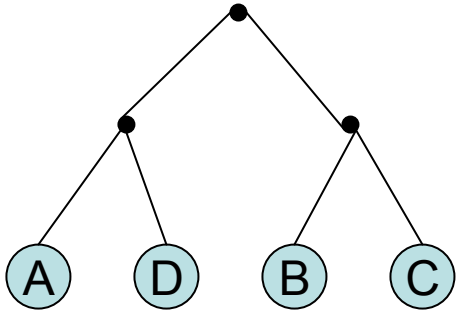
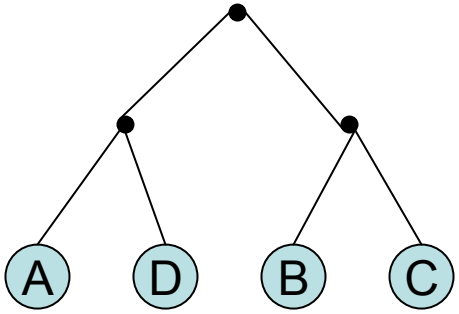
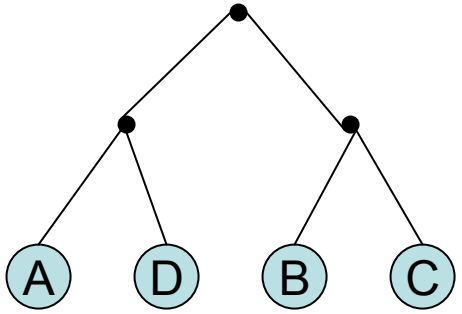
	1	2	3
A	a	a	t
B	c	g	c
C	c	g	c
D	t	g	t

All possible trees need to be evaluated in order to find the most parsimonous tree

T2

1	2	3	L(T)
			?

T3

1	2	3	L(T)
			?

Find the most parsimonous tree for this example: T1, T2 or T3 ?

The large parsimony problem

- Input: A matrix ***C*** describing *M* characters for a set of *N* species

M_{ij} – state of the *j*-th character for species *i*

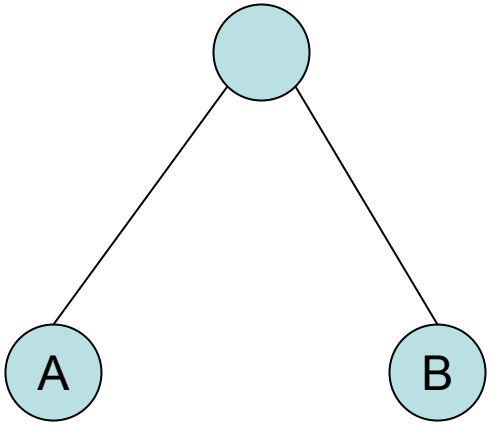
M_i – label of species *i*. All labels are distinct

Goal: find the most parsimonous tree: topology, leaf labeling and labels for internal nodes

The problem is NP-hard

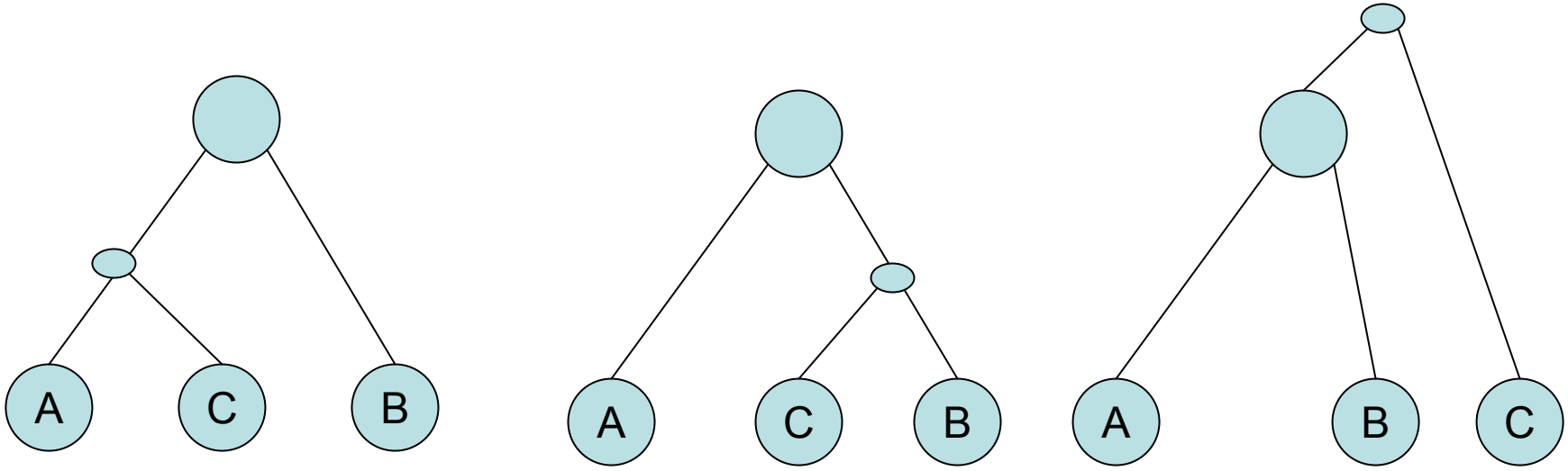
How many different trees

- For 2 species (N=2) only 1 possible tree



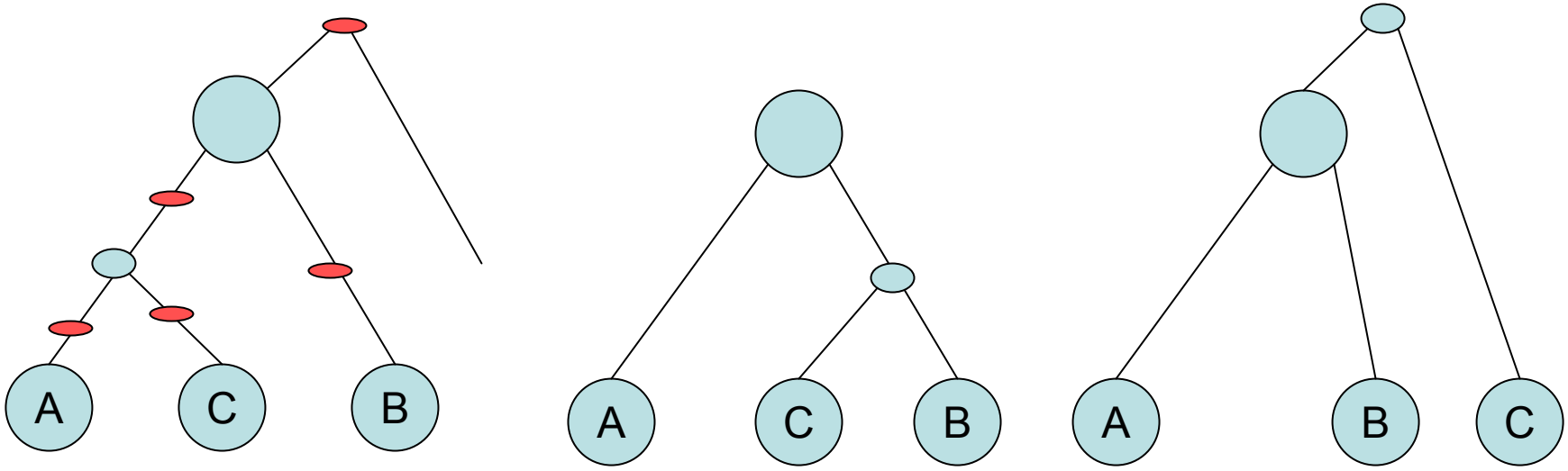
How many different trees

- For 3 species (N=3): a new leaf can be added by splitting any of 2+1 branches



How many different trees

- For 4 species (N=4): for each of these 3 trees, a new leaf can be added by splitting any of 4+1 branches



1*3*5*…(2N-3) possible trees (2n-3)!! – exponential number of different trees

Solution for the large parsimony problem

- A brute-force solution: enumerate all possible trees, compute the PScore of each tree, and choose the tree with a minimum score
- Optimization over the search space:
  - Branch-and-bound

The Branch-and-Bound technique

- The search is presented as the leaves of the search tree. Each node in this tree corresponds to some variant of a possible phylogenetic tree
- In order to apply the B&B technique, the score of each search node should be ***monotonous***, i. e. the score of each node is >= the score of any of its ancestors
- In this case, the algorithm guarantees to find the best tree, but it does not guarantee that the search will be faster than the exponential time
- Performs quite good in practice

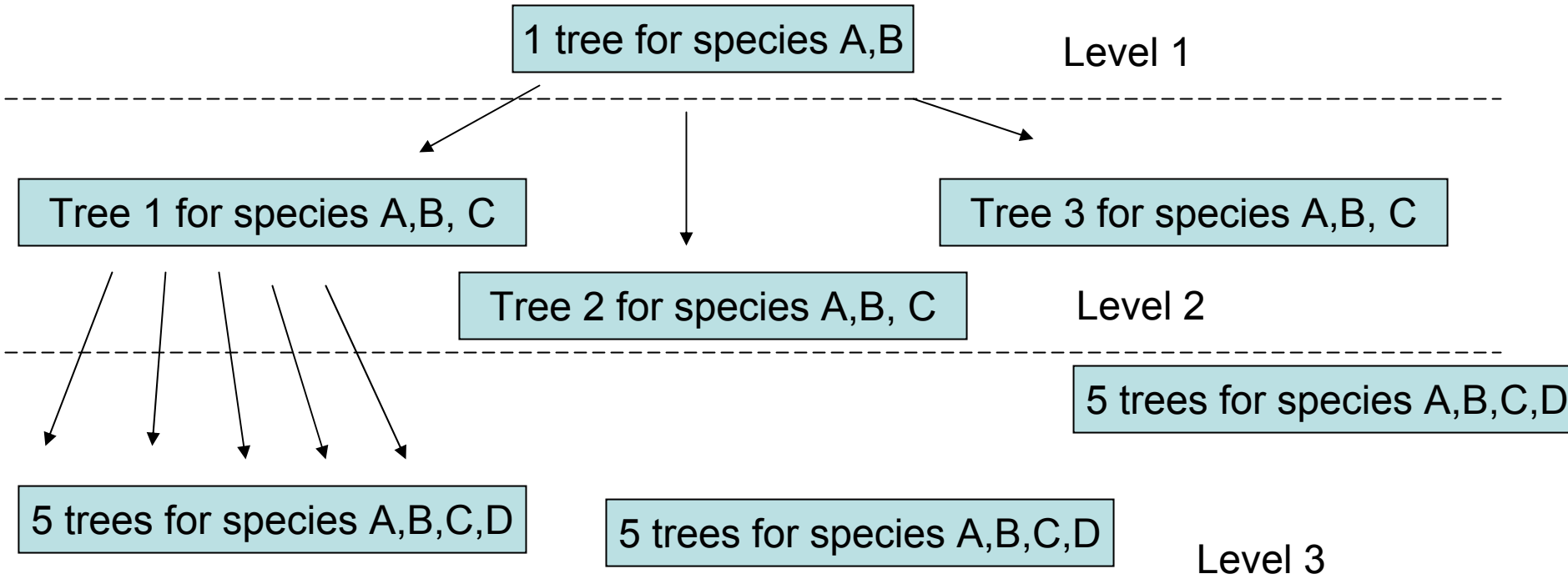
The Branch-and-Bound technique

- The search tree is traversed in order, and the score of the best leaf found so far is kept as a bound *B*.

- Whenever a node is reached with the score >*B*, the search tree is pruned at this node, i.e. its subtree is not searched, since it is guaranteed that any leaf in its subtree cannot have score <=*B*

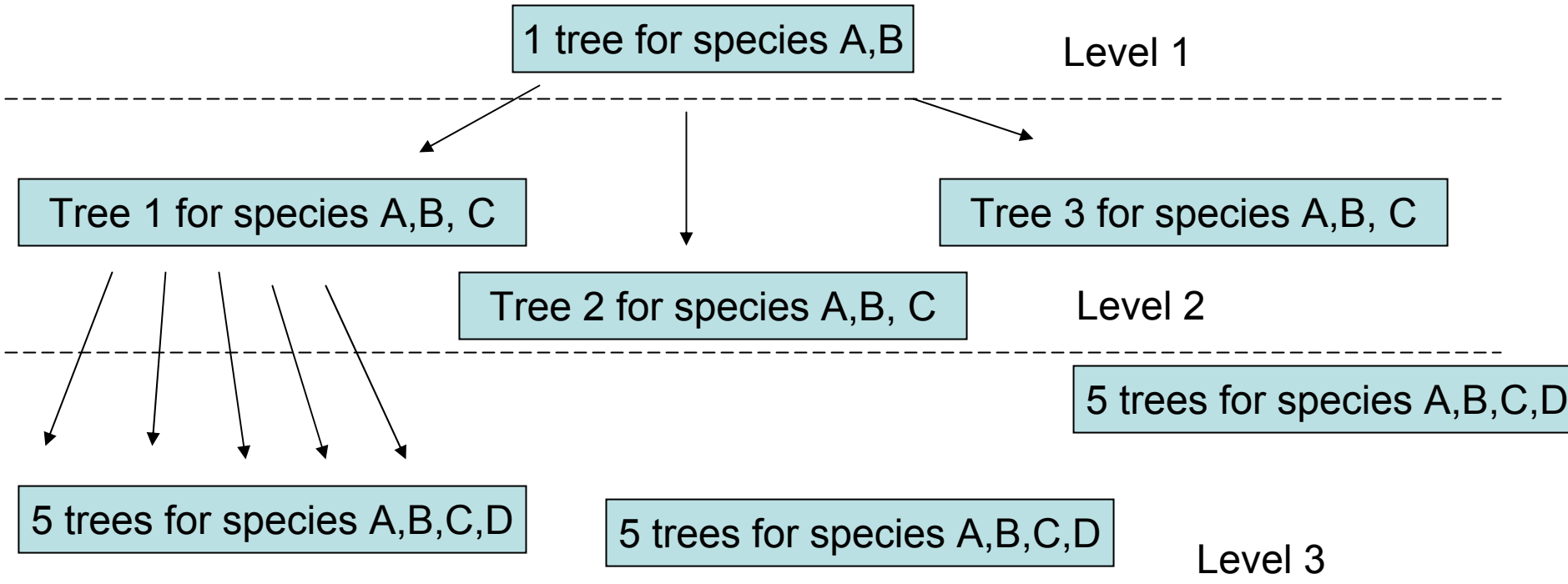
The Branch-and-Bound technique

- In level *k* of the tree we have nodes representing all possible phylogenetic trees for the first *k* species



The Branch-and-Bound technique

- There are $2k$-1 places to add a new species to an existing tree, thus each node at level k branches into $2k$-1 children.
- The requirement for monotonicity is satisfied, since adding a new node cannot reduce the PScore of the tree



The Branch-and-Bound technique

- The first tree for all *N* species is determined by finding a local minimum using one of the optimization techniques. This local minimum in many cases will be also a global minimum.
- The PScore of this tree is used as the bound on the rest of the search nodes, most of which are pruned and not expanded