Character-based (parsimony based) phylogenies

Lecture 17

Character-based phylogeny problem

- 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

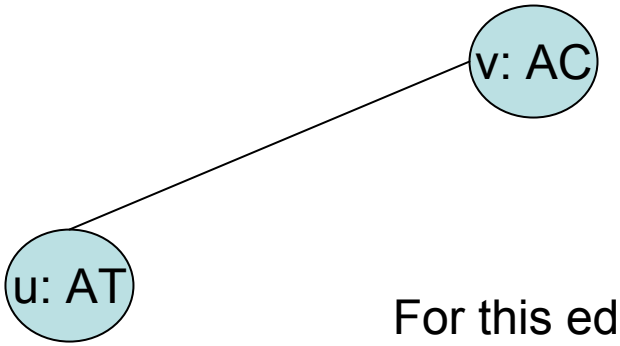
The parsimony score of the tree. Intuition

- 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

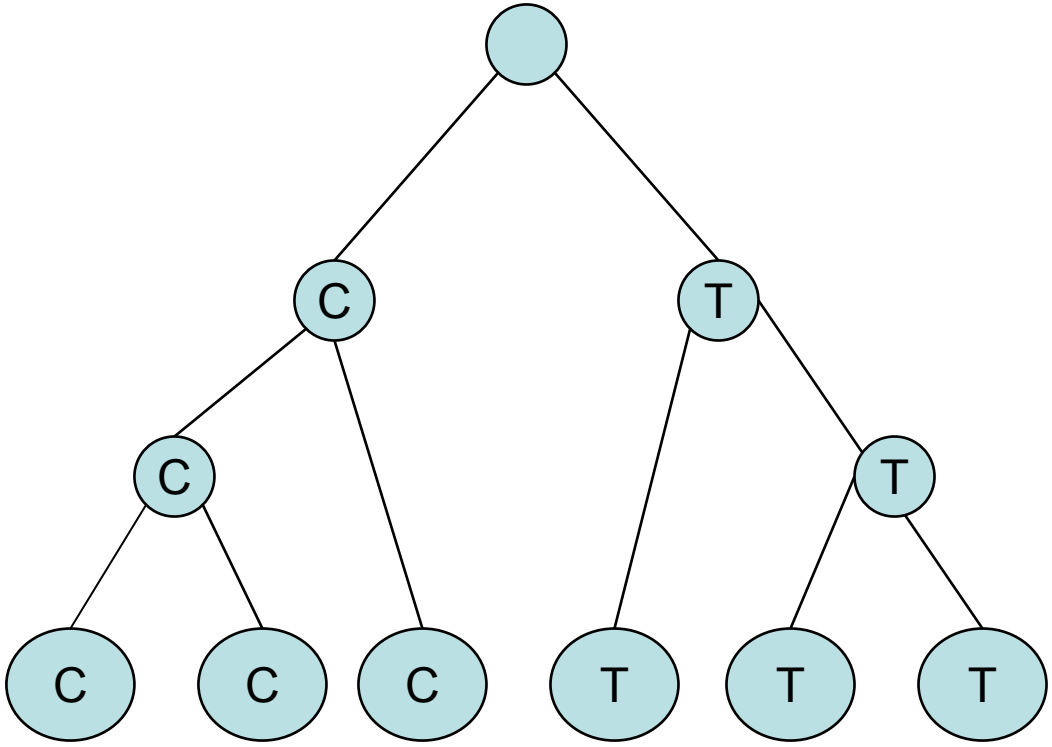
- Let V(T) be a set of vertices and E(T) be a set of edges of a given phylogenetic tree, and let the value of a character j in vertex v be vj.

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



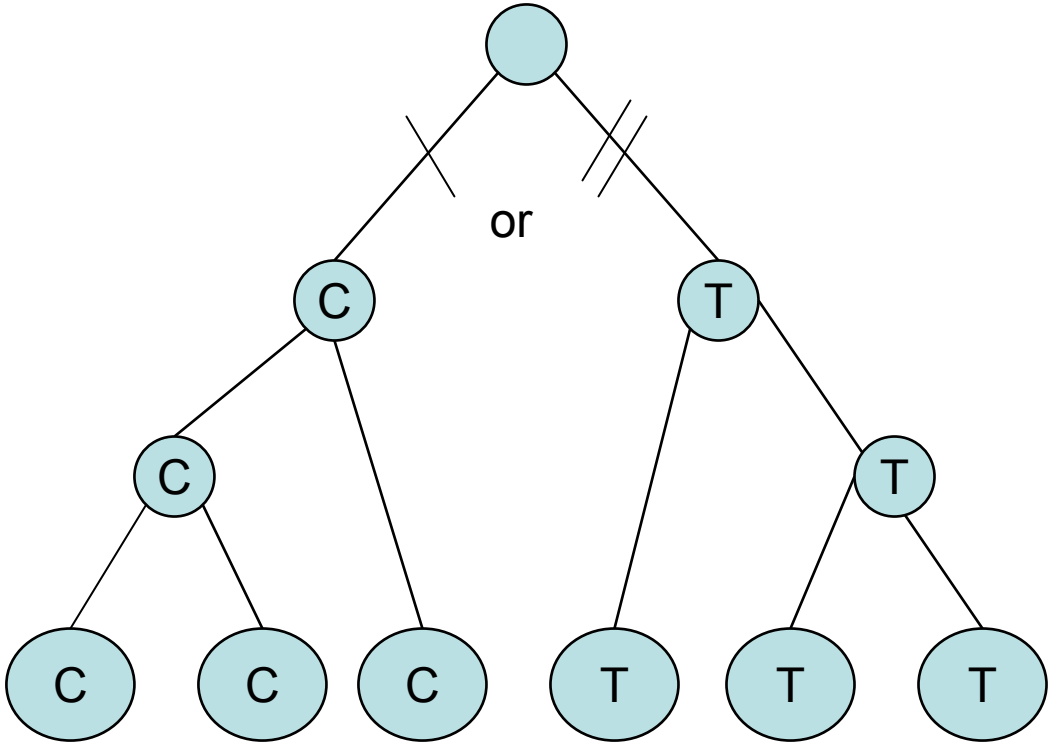
For this edge, we add 1 to the total parsimony score of the tree

Pscore example



What is Pscore of this tree?

Pscore example



Pscore =1

Character-based phylogeny

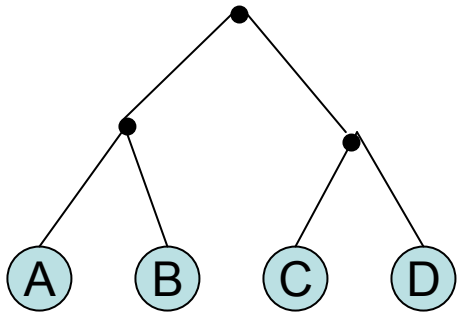
- Small parsimony problem: given a topology of a rooted phylogenetic tree and the character matrix C, find a labeling of ancestral sequences which implies the minimum parsimony score
- Large parsimony problem: given the character matrix C, build the tree with the minimum parsimony score (minimum number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

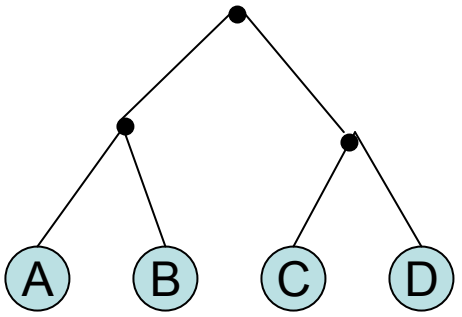
tree T

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

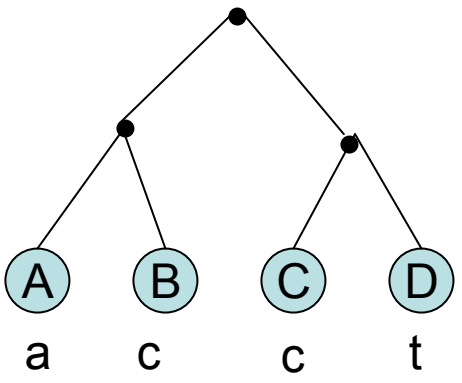
Each leaf node contains only 1 state for each character, and is initially marked with this character

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***:

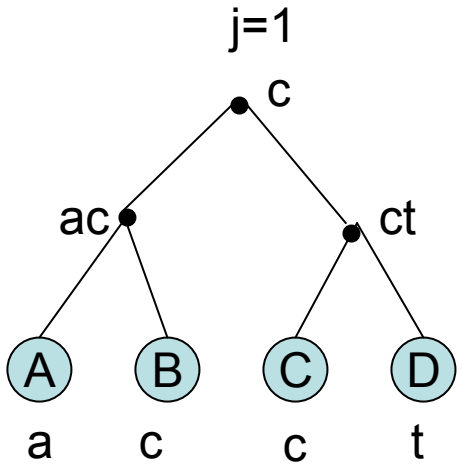
//initialize

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

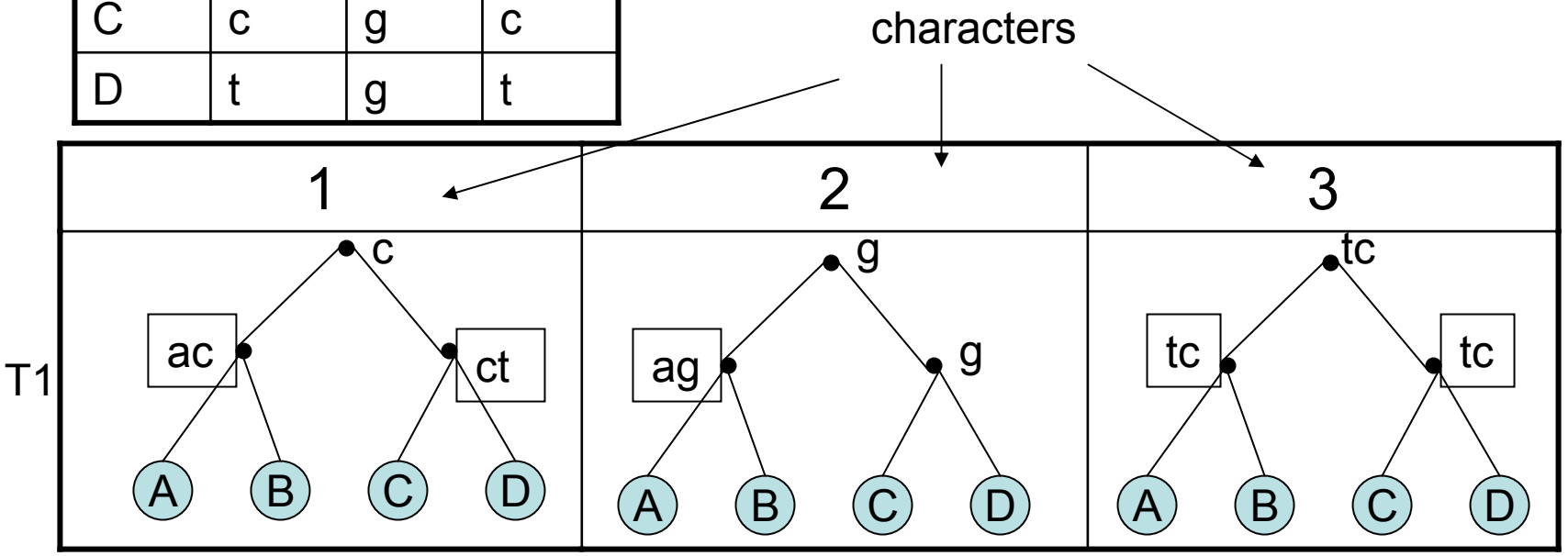
$Set_u \cup Set_w$

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

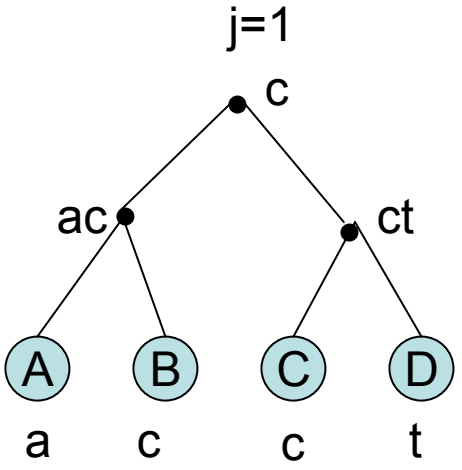
The Fitch algorithm phase I example

	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*

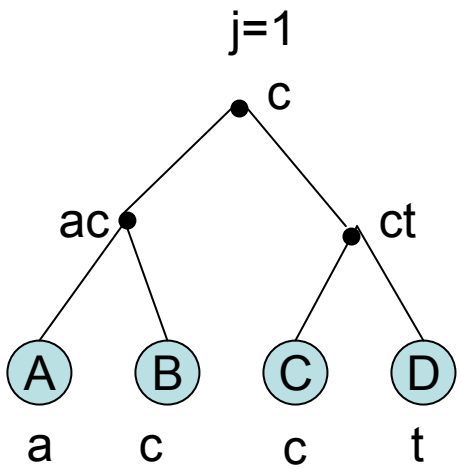
for each vertex *v* with parent *u*

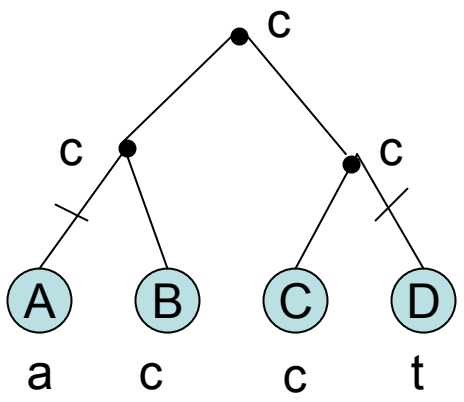
if u_j € Set_v

$v_j=u_j$

else

v_j=any element from Set_v



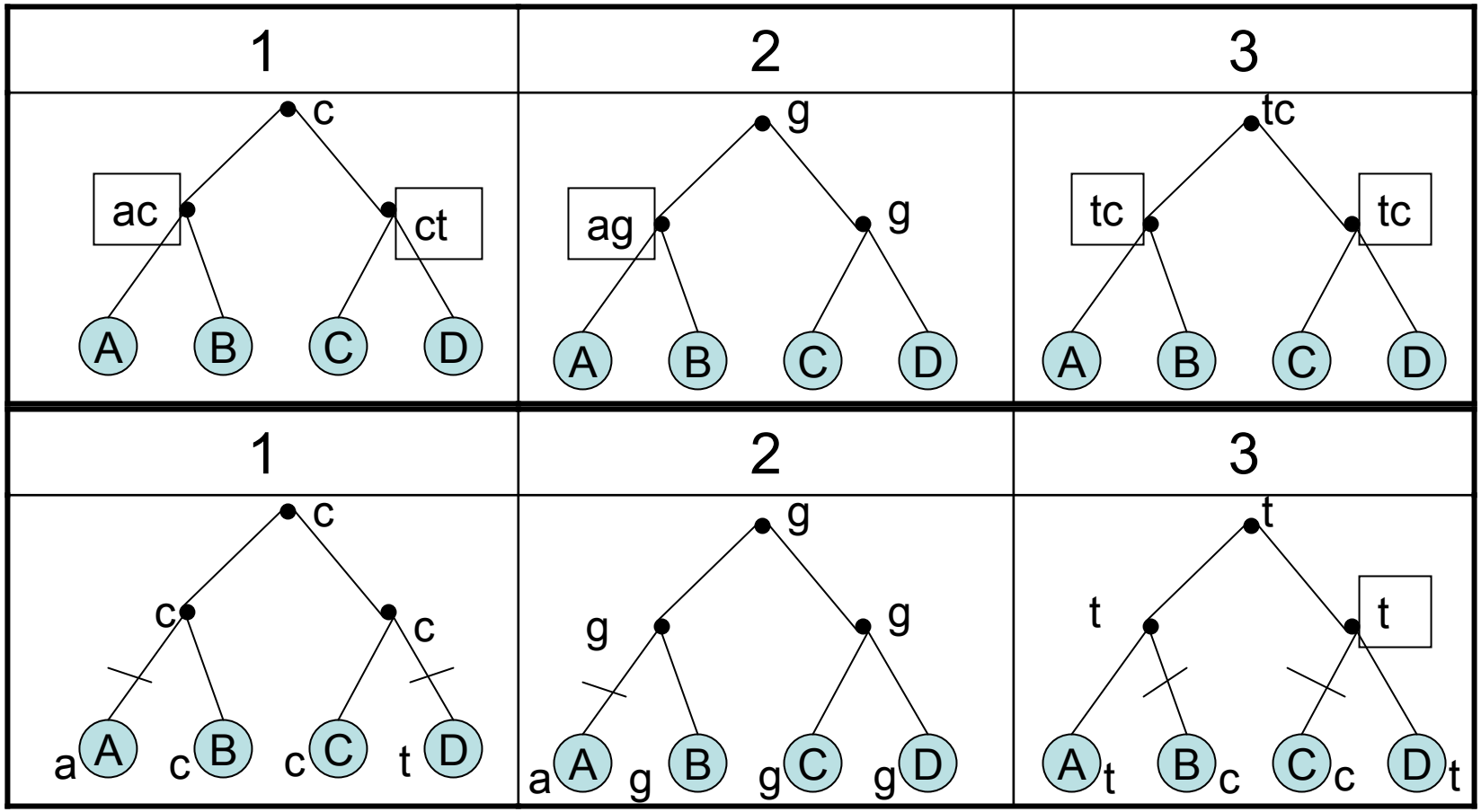


The parsimony score for the first character=2

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

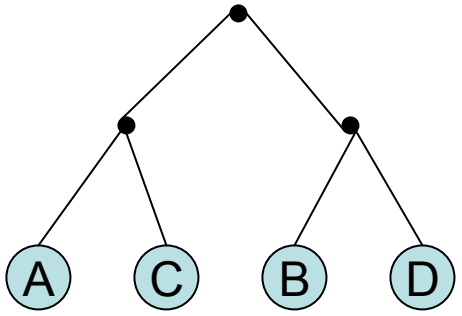
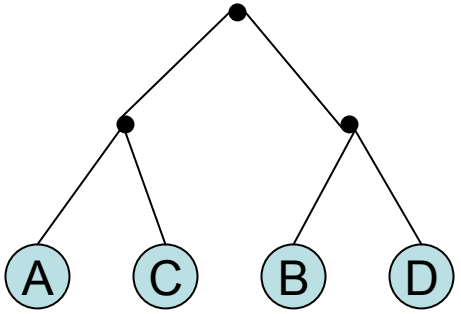
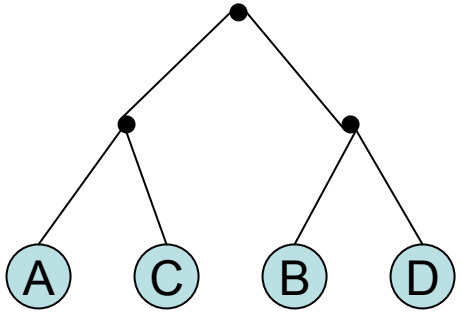
- Different, application-specific costs are assigned for each change of character from state to state
- This is more realistic, since substitutions happen with different probabilities
- An overall algorithm is similar to the Fitch algorithm (you can read it in your textbook)

Large Parsimony problem for 4x3 matrix

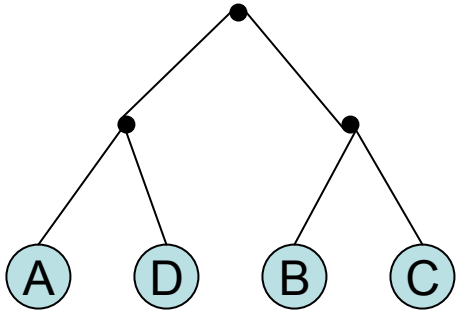
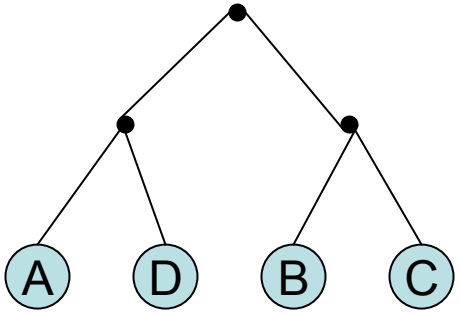
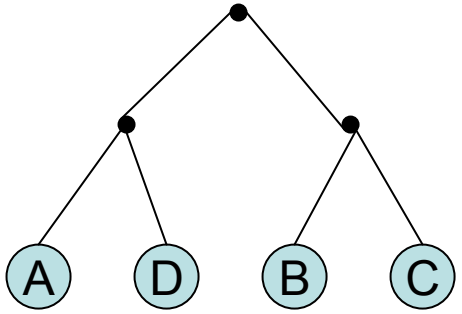
	1	2	3
A	a	a	t
B	c	g	c
C	c	g	c
D	t	g	t

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*

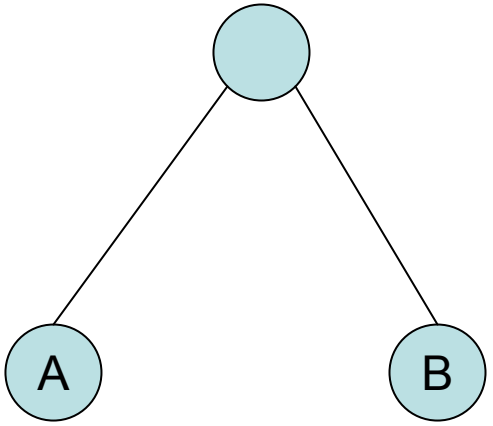
M_i – label of species *i*. All labels are distinct

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

The problem is NP-hard

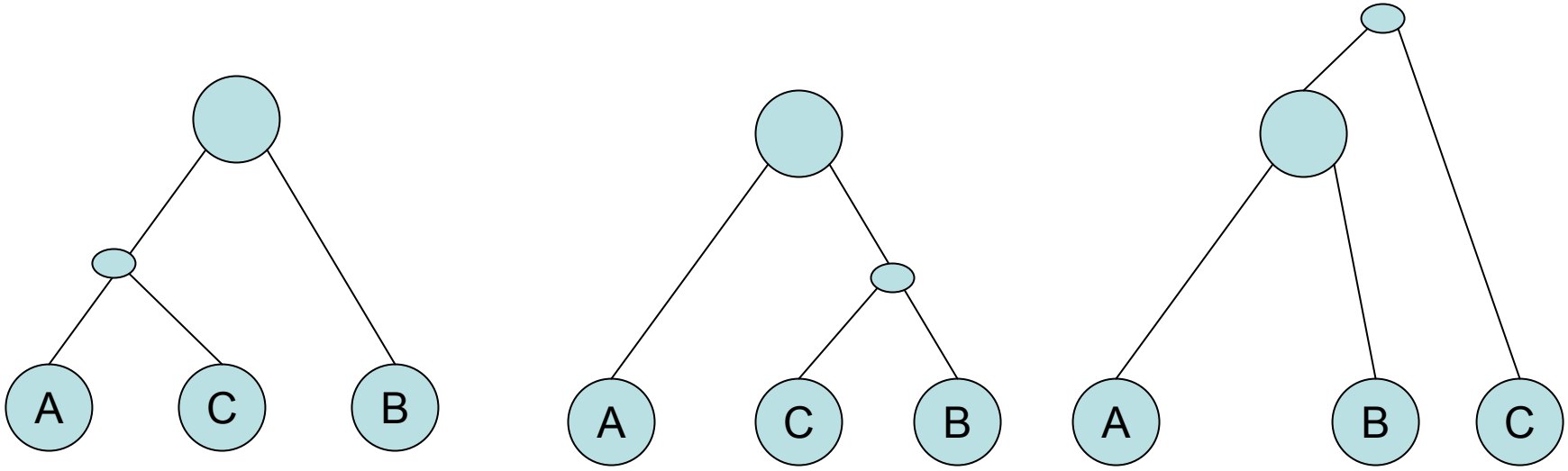
How many different trees

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



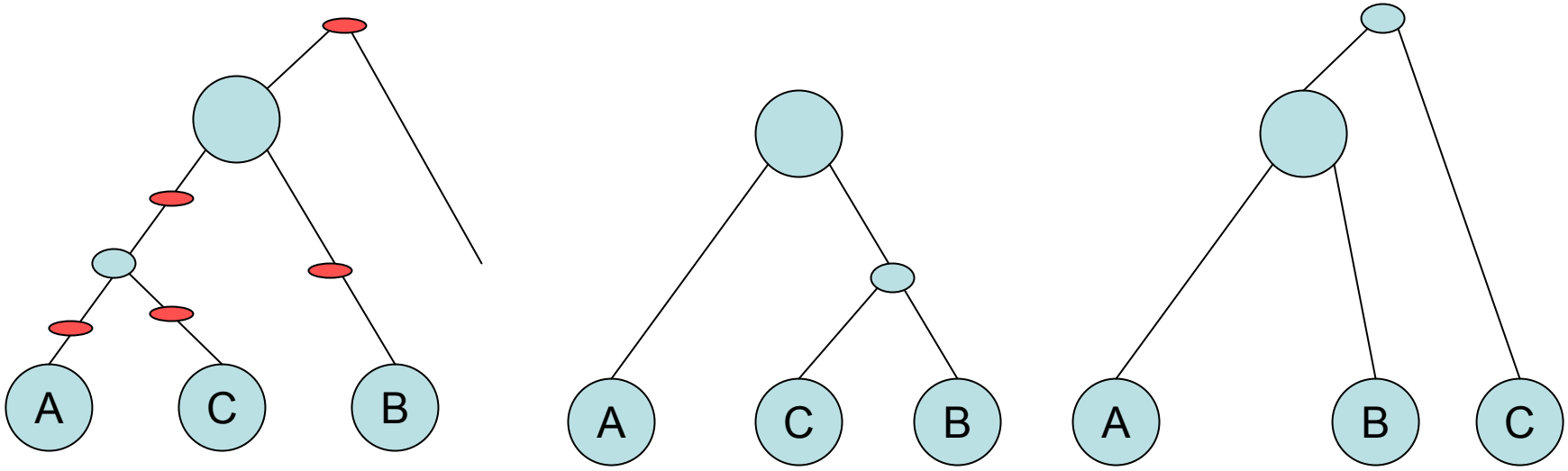
How many different trees

- 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

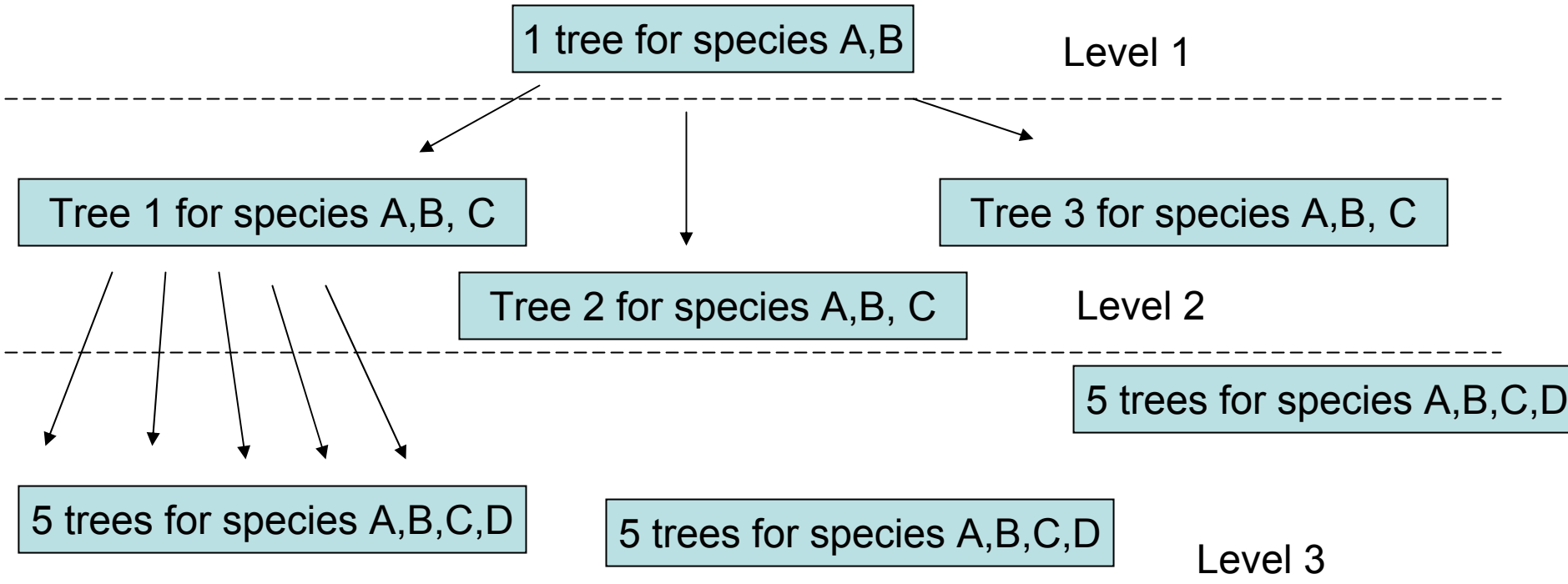
The Branch-and-Bound technique

- 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*

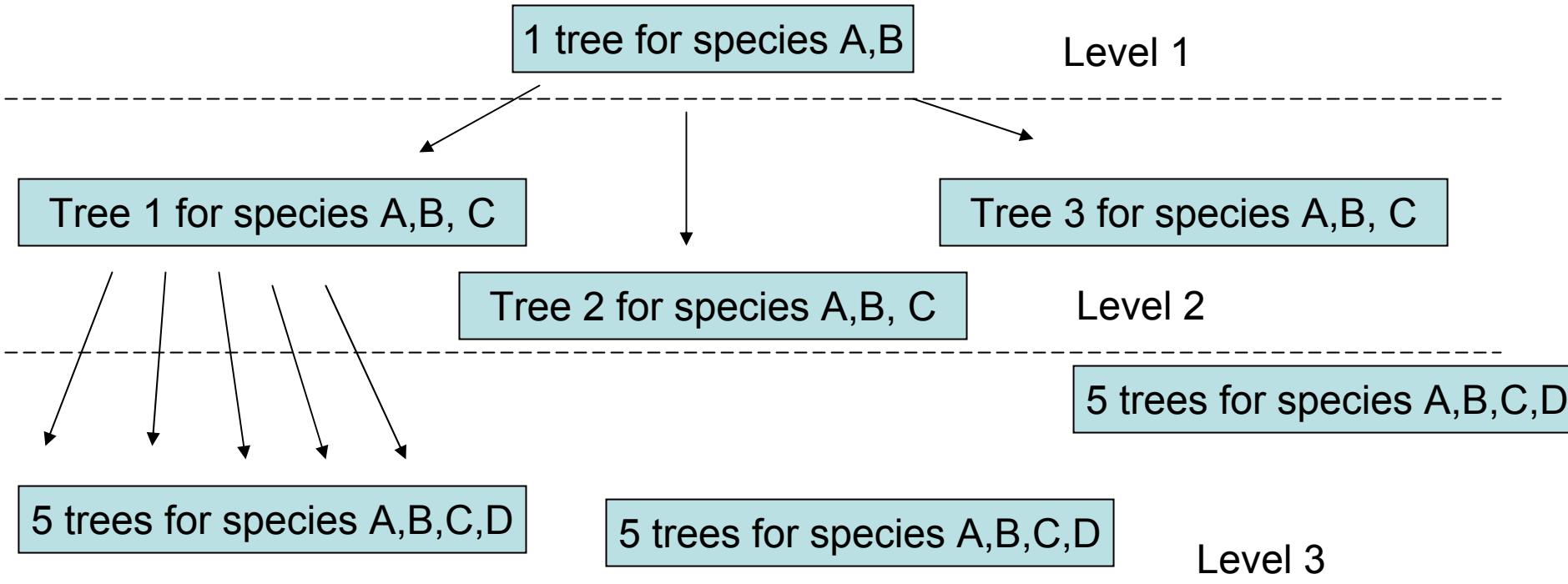
The Branch-and-Bound technique

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



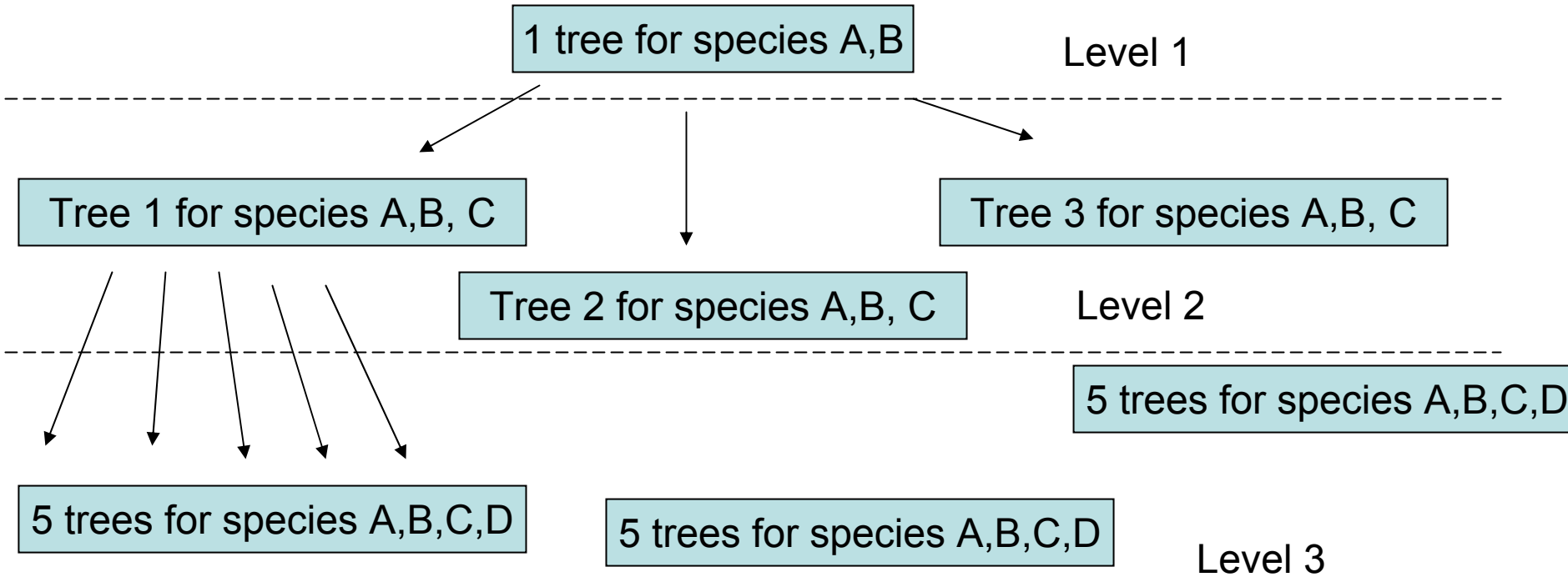
The Branch-and-Bound technique

- There are 2*k*-1 places to add a new species to an existing tree, thus each node at level *k* branches into 2*k*-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



Character compatibility and perfect phylogeny

- Compatibility of characters in the perfect phylogeny problem (see lecture 14) is a special case of parsimony.

- Reminder: a set of characters is compatible with the tree iff there is a labeling of the internal nodes s.t. the total number of character changes is exactly *k*-1, where *k* is the number of possible states for each character.

- This was demonstrated on the example of a character with only 2 states (binary). In this case, if the characters are compatible, the perfect phylogenetic tree can be built, with the parsimony score 1 for each character (each character changes its state from 0 to 1 only once)