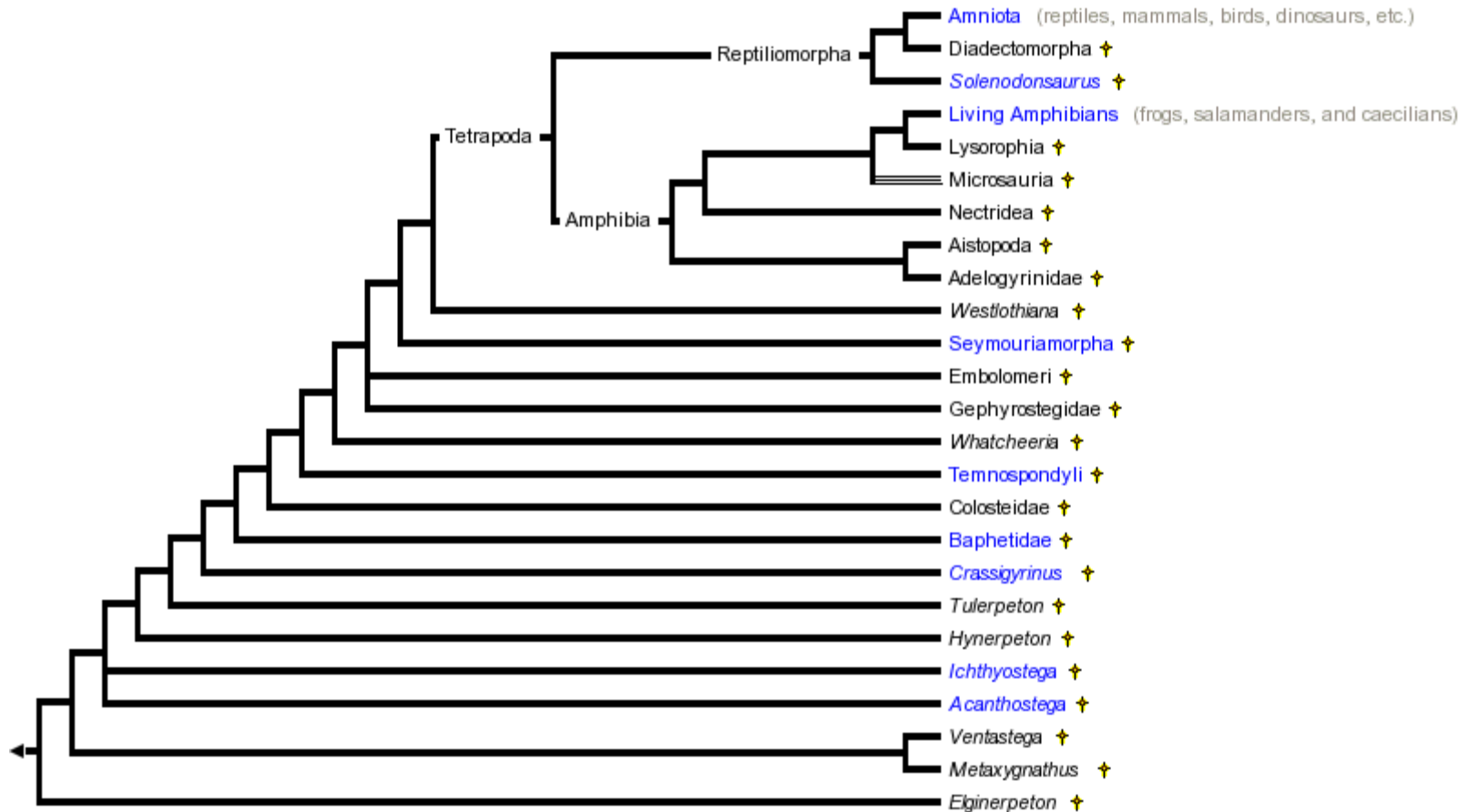


CMSC423: Bioinformatic Algorithms, Databases and Tools

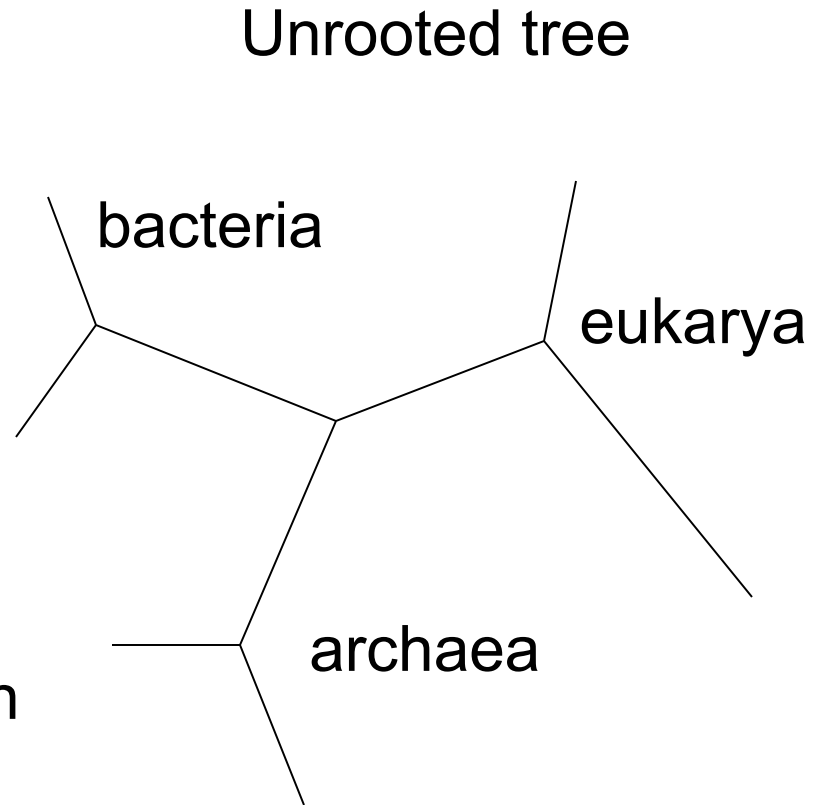
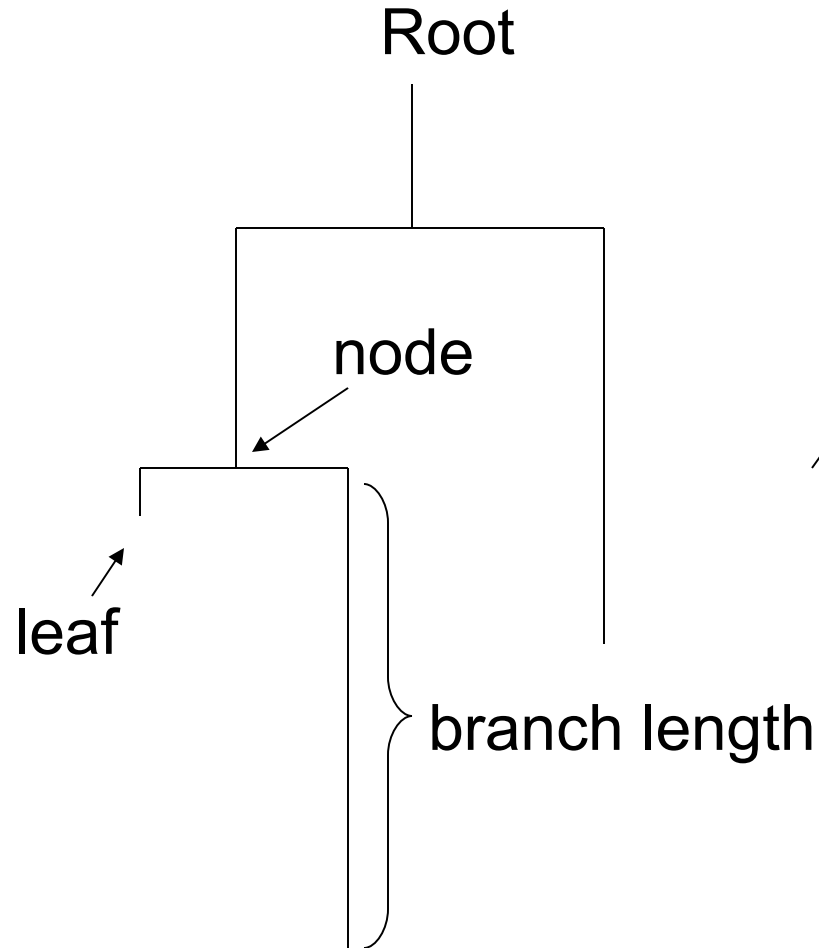
Phylogenetic trees

Phylogenetic trees – how evolution works

- <http://www.tolweb.org/tree/> - the tree of life



Anatomy of a tree



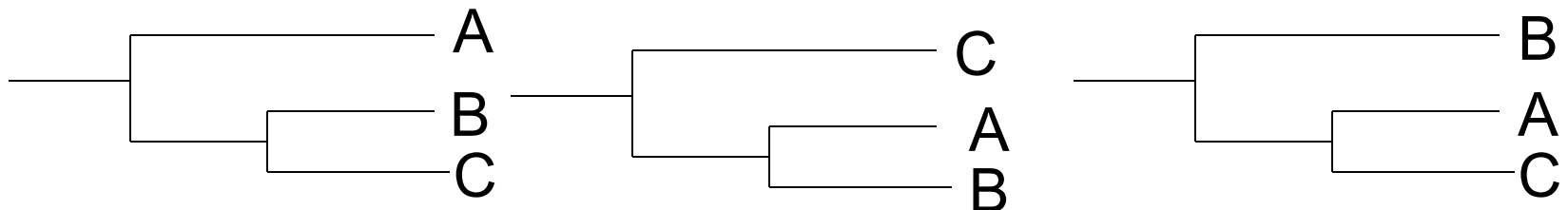
Phylogenetic trees are usually binary (though they don't have to)

Phylogeny questions

- Given several organisms & a set of features (usually sequence, but also morphological: wing shape/color...)
- A. Given a phylogenetic tree – figure out what the ancestors looked like (what are the features of internal nodes)



- B. Find the phylogenetic tree that best describes the common evolutionary heritage of the organisms



Phylogeny questions

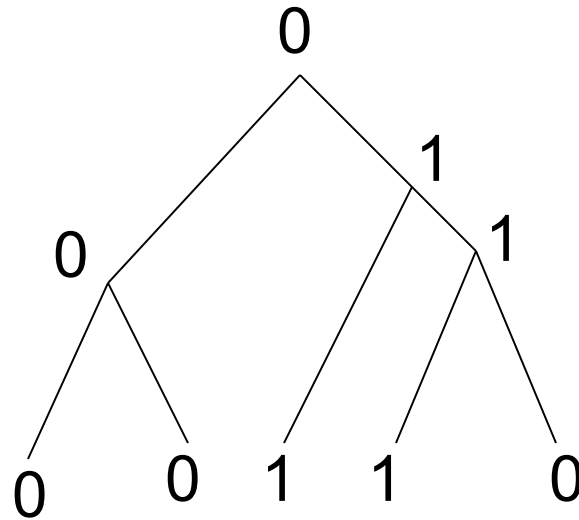
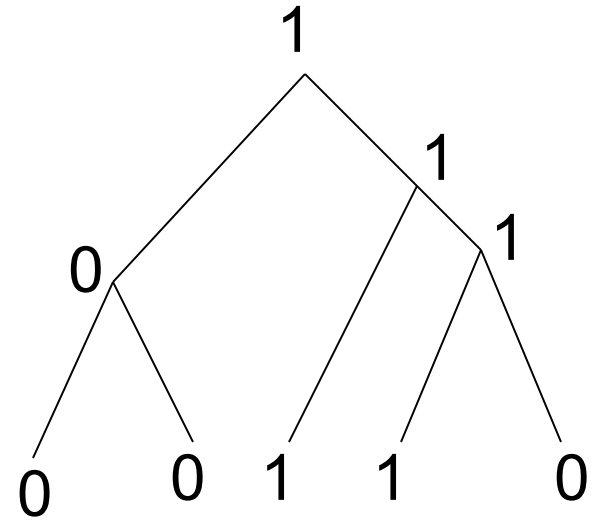
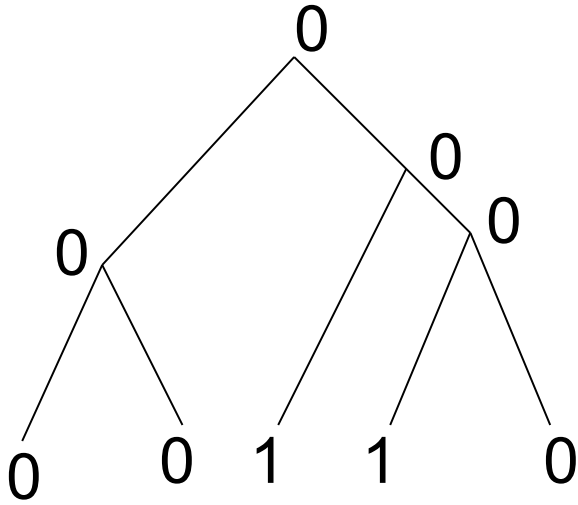
- A. Easy-ish – can be done with dynamic programming
- B. Hard – Many possible trees

$$\frac{(2n-3)!}{2^{n-2}(n-2)!} \quad \text{rooted trees with } n \text{ leaves}$$

Scoring a tree – Sankoff's algorithm

- Assumption – we try to minimize # of state changes from root to leaves – Parsimony approach
- Small parsimony
 - given a tree where leaves are labeled with m-character strings
 - find labels at internal nodes s.t. # of state transitions is minimized
- Weighted small parsimony
 - same as parsimony except that state transitions are assigned weights
 - minimize the overall weight of the tree

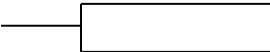
Example



Sankoff's algorithm

- At each node v in the tree store $s(v,t)$ – best parsimony score for subtree rooted at v if character stored at v is t
- Traverse the tree in post-order and update $s(v,t)$ as follows
 - assume node v has children u and w
 - $s(v,t) = \min_i \{s(u,i) + \text{score}(i,t)\} + \min_j \{s(w,j) + \text{score}(j,t)\}$
- Character at root will be the one that maximizes $s(\text{root}, t)$
- Note – this solves the weighted version. For unweighted set $\text{score}(i,i) = 0$, $\text{score}(i,j) = 1$ for any i,j

Trees as clustering

- Start with a distance matrix – distance (e.g. alignment distance) between any two sequences (leaves)
 - Intuitively – want to cluster together the most similar sequences
 - UPGMA – Unweighted Pair Group Method using Arithmetic averages
 - Build pairwise distance matrix (e.g. from a multiple alignment)
 - Pick pair of sequences that are closest to each other and cluster them – create internal node that has the sequences as children
 - Repeat, including newly created internal nodes in the distance matrix
- 
- Key element – must be able to quickly compute distance between clusters (internal nodes) – weighted distance

$$D(cl_1, cl_2) = \frac{1}{|cl_1| + |cl_2|} \sum_{p \in cl_1, q \in cl_2} D(p, q)$$

Trees as clustering

- Note that UPGMA does not estimate branch lengths – they are all assumed equal
- Neighbor-joining
 - distance between two sequences is not sufficient – must also know how each sequence compares to every other sequence
 - $\text{NJdist}(i,j) = D(i,j) - (r_i + r_j)$ $-r_i, r_j$ correction factors

$$r_i = \frac{1}{m-2} \sum_k D(i,k)$$

Neighbor joining

- Pick two nodes with $NJdist(i,j)$ minimal
 - Create parent k s.t.
 - $D(k, m) = 0.5 (D(i,m) + D(j,m) - D(i,j))$ for every other node m
 - $D(i, k) = 0.5 (D(i,j) + r_i - r_j) - \text{length of branch between } i \text{ \& } k$
 - $D(j, k) = 0.5 (D(i,j) + r_j - r_i) - \text{length of branch between } j \text{ \& } k$

Trees as clustering

- Note that both UPGMA and NJ assume distance matrix is additive: $D(i,j) + D(j,k) = D(i,k)$ - usually not true but close
- Also, NJ can be proven to build the optimal tree!
- But, simple alignment distance is not a good metric

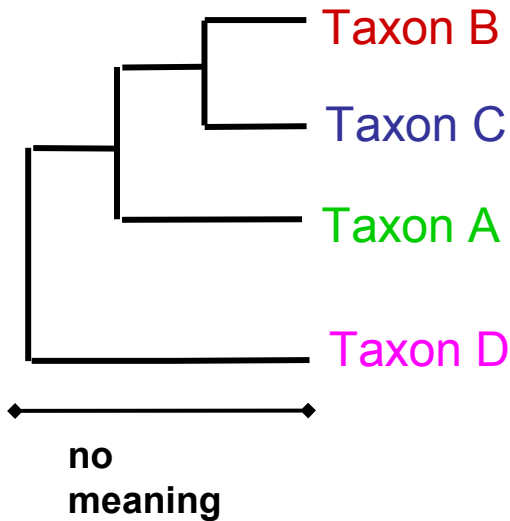
Maximum likelihood

- For every branch $S \rightarrow T$ of length t , compute $P(T|S,t)$ – likelihood that sequence S could have evolved in time t into sequence T
- Find tree that maximizes the likelihood
- Note that likelihood of a tree can be computed with an algorithm similar to Sankoffs
- However, no simple way to find a tree given the sequences – most approaches use heuristic search techniques
- Often, start with NJ tree – then "tweak" it to improve likelihood

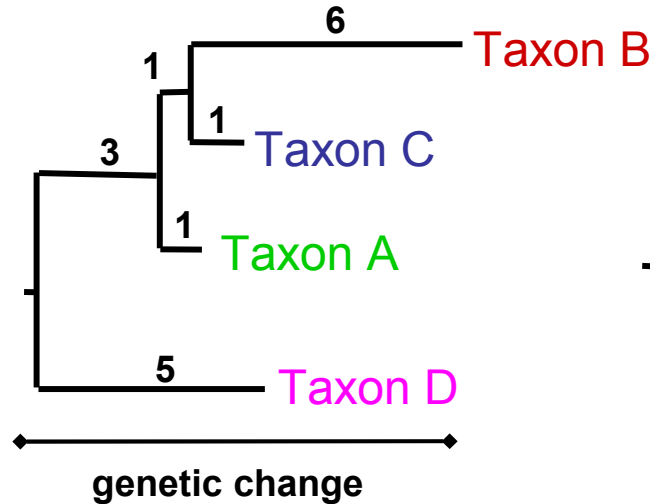
Tree analysis & display

Three types of trees

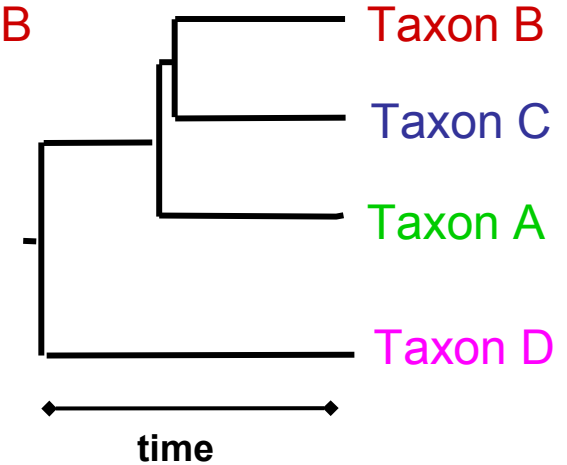
Cladogram



Phylogram

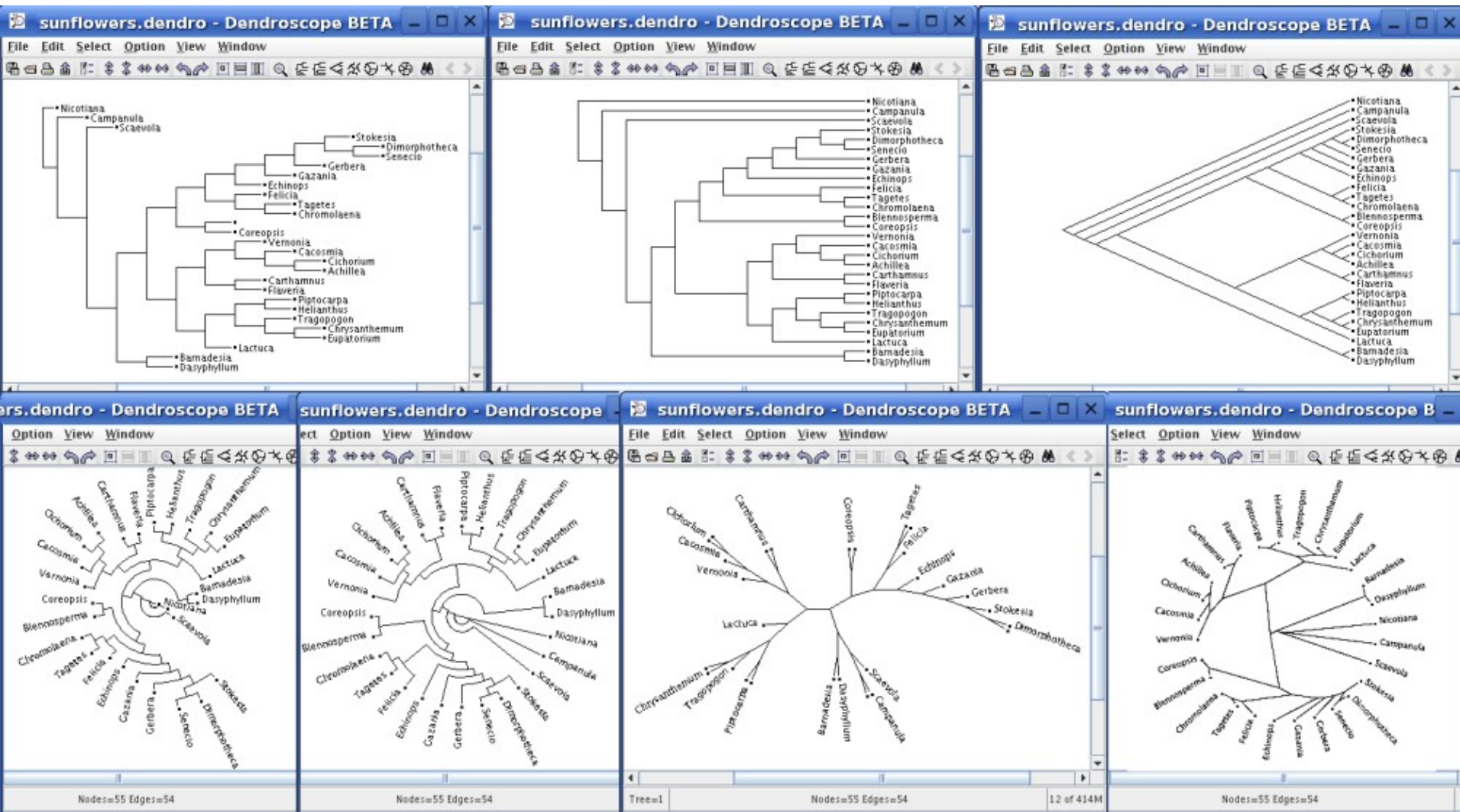


Ultrametric tree



All show the same evolutionary relationships, or branching orders, between the taxa.

Different tree views



Drawing trees

- Trees are easy to draw – just need to figure out how much space the leaves will take
- Step 1 – calculate how much space each node will take (how many leaves from current node)
- Step 2 – spread out the nodes according to # of leaves
- Many ways of optimizing: e.g. width, area
- For large trees
 - 3D displays (there's more room in 3D)
 - interactive displays (expand contract nodes as needed)

Analysis example

- Build multiple alignment (e.g. Muscle, ClustalW)
- Clean up alignment
 - manual editing
 - filters (pre-defined structure information)
- Build tree
 - PAUP – parsimony & others
 - Phylip – maximum likelihood
 - Tree-Puzzle –maximum likelihood
 - etc... (many packages)
- Integrated system – ARB
 - www.arb-home.de

Questions

- Why do you need a multiple alignment for phylogeny?
- What is the running time of the neighbor-joining algorithm, given k sequences of length L ?
- What is the parsimony score of the following tree, and what are the labels at internal nodes?

