CS 581

Tandy Warnow

Today’s material (from Chapters 1-3)

- Newick strings
- Representing rooted trees using clades and rooted triplet trees
- Constructing a rooted tree from its set of clades using Hasse Diagrams
- Constructing a rooted tree from rooted triplet trees using Aho, Sagiv, Szymanski, and Ullman
- Constructing a rooted tree from rooted subtrees of any size

Today’s material (from Chapters 1-3)

- Newick strings
- Representing rooted trees using clades and rooted triplet trees
- Constructing a rooted tree from its set of clades using Hasse Diagrams
- Constructing a rooted tree from rooted triplet trees using Aho, Sagiv, Szymanski, and Ullman
- Constructing a rooted tree from rooted subtrees of any size

Newick representations

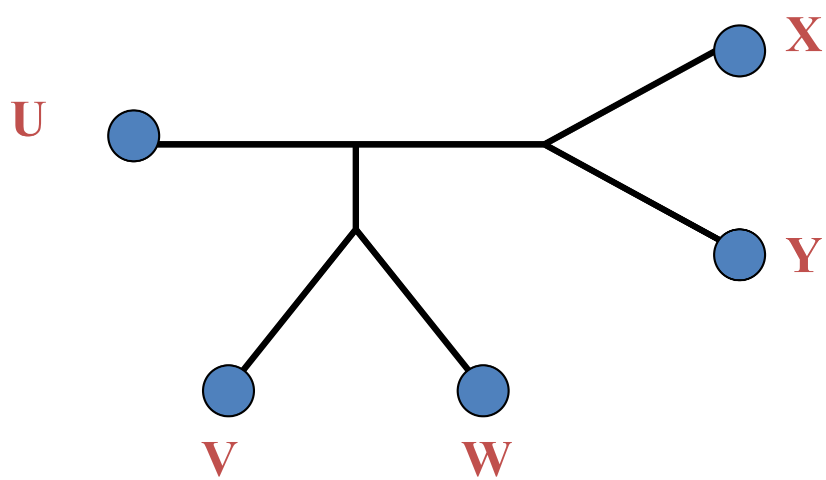
- For a rooted tree, we represent a graph with a string with the taxa, commas, and nested parentheses.
- For example, what tree is represented by (a,(b,(c,((d,e),(f,g))))))?
- How do we represent an unrooted tree? (Easy - root it somewhere, and write down the Newick representation of the rooted version.)

(U,((V,W),(X,Y)))

or

((X,Y),(U,(V,W))

or …



Rooted vs. unrooted

- Task: be able to move between rooted and unrooted representations of trees
- Task: be able to compare two trees and see if they are different or the same

Today's material (from Chapters 1-3)

- Newick strings
- Representing rooted trees using clades and rooted triplet trees
- Constructing a rooted tree from its set of clades using Hasse Diagrams
- Constructing a rooted tree from rooted triplet trees using Aho, Sagiv, Szymanski, and Ullman
- Constructing a rooted tree from rooted subtrees of any size

Clades

Definition: Let T be a rooted tree leaf-labelled by S, let v an internal node in T, and let $\mathbf{X_v}$ be the set of leaves in T below v. Let **Clades(T) = {$\mathbf{X_v}$: v in V(T)}**. Note: X_v is also called the "cluster" at node v, so this is sometimes called Clusters(T).

- Question: Given Clades(T), can we compute T?

Triplet Trees

Definition: Let T be a rooted tree leaf-labelled by S. A triplet tree is a rooted 3-leaf subtree of T, such as ((a,b),c). The set of all triplet trees of T is denoted Triplets(T).

- Question: Given Triplets(T), can we compute T?

Today’s material (from Chapters 1-3)

- Newick strings
- Representing rooted trees using clades and rooted triplet trees
- Constructing a rooted tree from its set of clades using Hasse Diagrams
- Constructing a rooted tree from rooted triplet trees using Aho, Sagiv, Szymanski, and Ullman
- Constructing a rooted tree from rooted subtrees of any size

Computing rooted trees from clades

- Partially order the set of clades by containment, add in the full set S, and compute the Hasse Diagram of the resultant poset (partially ordered subset)

Note: Hasse Diagrams and Partially Ordered Sets are explained in Appendix B in the textbook.

Tree construction from clades

Questions:

- Accuracy?
- Running time?
- But, *how are we to compute clades?*

Clade compatibility

- Definition: Let T be a rooted tree leaf-labelled by S, v an internal node in T, and X_v the leaves in T below v. Let Clades(T)={X_v: v in V(T)}.
- Theorem: Let X be a set of subsets of S. Then there exists a tree T such that X = Clades(T) if and only if for all A, B in X, either A and B are disjoint, or one contains the other.

Proof of the theorem

- One direction is easy
- The other direction is a proof by construction!

Today's material (from Chapters 1-3)

- Newick strings
- Representing rooted trees using clades and rooted triplet trees
- Constructing a rooted tree from its set of clades using Hasse Diagrams
- Constructing a rooted tree from rooted triplet trees using Aho, Sagiv, Szymanski, and Ullman
- Constructing a rooted tree from rooted subtrees of any size

Rooted Tree Compatibility

- Input: Set X of rooted trees, not all on the same set of leaves.
- Output: Tree T (if it exists) that agrees with all the trees in X, and otherwise “Fail”

This problem is solvable in polynomial time.

Proof: the Aho, Sagiv, Szymanski, and Ullman (ASSU) algorithm!

ASSU algorithm

Given set X of k triplet trees on n species:

- If n>1, then construct graph with each species one of the vertices, and edges (a,b) for triplets ab|c.
- If the graph has a single component, reject (the set is not compatible); else recurse on each component, and return tree formed by making the rooted trees on the components each a subtree off the root of the returned tree.

Why does it work?

If the set X of triplet trees is compatible,

- Then there is a rooted tree T with at least two subtrees off the root, T_1 and T_2.
- Any two leaves a,b in different subtrees off the root cannot be in a triplet ab|c.
- Hence the graph formed for the set of triplet trees cannot be connected.
- Therefore the graph formed for the set of triplet trees must have at least two components.
- This argument applies recursively to every subset of X.
- Hence the algorithm returns a tree on which all the triplet trees agree.

If the set X of triplet trees is not compatible, it is not hard to show that the algorithm will detect this (proof by induction on the number of taxa).

Today's material (from Chapters 1-3)

- Newick strings
- Representing rooted trees using clades and rooted triplet trees
- Constructing a rooted tree from its set of clades using Hasse Diagrams
- Constructing a rooted tree from rooted triplet trees using Aho, Sagiv, Szymanski, and Ullman
- Constructing a rooted tree from rooted subtrees of any size

Compatibility of rooted trees

- Suppose the input is a set X of rooted trees (not necessarily triplet trees).

Compatibility of rooted trees

- Suppose the input is a set X of rooted trees (not necessarily triplet trees).
- Can we use ASSU to determine if X is compatible, and to compute a compatibility supertree for X?

Compatibility of rooted trees

- Suppose the input is a set X of rooted trees (not necessarily triplet trees).
- Can we use ASSU to determine if X is compatible, and to compute a compatibility supertree for X?
- Solution: YES, just encode each rooted tree in X by its set of rooted triplet trees (or some subset of these that suffices to define each tree in X), and then run ASSU.

Summary (so far)

- We have seen how to construct a rooted tree from its set of clades or triplet trees.
- We have seen how to test compatibility of a set of clades or rooted trees.

Summary (so far)

- We have seen how to construct a rooted tree from its set of clades or triplet trees.
- We have seen how to test compatibility of a set of clades or rooted trees.

Can we use these ideas to design divide-and-conquer methods to construct large rooted trees?

Summary (so far)

- We have seen how to construct a rooted tree from its set of clades or triplet trees.
- We have seen how to test compatibility of a set of clades or rooted trees.

What can we do to construct unrooted trees?