CS 224 HW 4 Due Nov. 29 (This gives you a lot of time but don't wait to start it. Some of the problems are easy and some are not.)

1. Suppose that binary matrix M is represented by a sparse-matrix data structure: for each taxon p, the characters that p possesses are represented in a linked list. That is, the 1s in the row for p are given by a linked list. Let l be the total length of these n lists. Give an $O(l)$- time algorithm to determine if M has a perfect phylogeny, and to construct one if possible.

2. The classic *consecutive ones problem* problem is the following: The input to the consecutive ones problem is a binary matrix, and the goal is to permute the columns of the matrix so that in each row, all ones become located in a block of consecutive entries. The analogous consecutive-ones property *for columns* requires that all the ones in every column be in a contiguous block. It is known that *if* M has a perfect phylogeny (with all zero ancestral sequence at the root) then the rows of M can be permuted so that every column has the consecutive-ones property. Prove this.

   The consecutive-ones property allows a visually nicer presentation of the matrix. An $O(n^2m)$ algorithm to reorder the rows was given in the literature. Linear time, but fairly complex, algorithms are known for creating, when possible, the consecutive-ones property in general, but for matrices with a perfect phylogeny, there is a much simpler linear time method. Describe it.

3. In the binary character perfect phylogeny problem, it was assumed that each character is in state zero at the root of the tree. This assumption is often too strong. However, once choices for the root states are made, the states can be relabeled so that the root states are all zero, and the resulting matrix can be tested to see if it has a phylogenetic tree. Of course, if M has a perfect phylogeny, then it has one where the root (ancestor node) is labeled with any chosen row of M. But it may not seem natural to place one of the input objects at the root of the tree. A more natural choice for the root state of a character j is the *majority* state: assign the root state of character j to be 1 if and only if $|O_j| \geq n/2$. Show that if there is any choice of root states that leads to a perfect phylogeny, then the majority choice for each character will also lead to one.

4. The following problem was recently discussed in an active exchange on bionet.molbio.evolution, an Internet newsgroup concerned with evolution: How can one tell if two binary trees in New Hampshire format are the same tree? The discussion on the net did not fully resolve the issue (in computer science worst-case terms) although one cubic-time (or more) method was suggested, and a very practical, but *exponential* worst case, method was also suggested. A simple linear-time method is possible.

   Let T be a binary tree where each leaf has a unique label. A New Hampshire encoding of T is a string that describes T. It can be obtained by a depth- first traversal of T as follows: the string is accumulated left to right with all symbols appended at the right end; the first time a non-leaf node is visited, append a left parenthesis to the growing string; when a leaf is visited, append its label; the first time the traversal backs up to a node, append a comma; and when the traversal backs up to a node for the second time, append a right parenthesis. For example, the tree shown in Figure 1 has New Hampshire code of *((D,B),(E,(C,A)))*. Note that while the code uniquely identifies the tree, the code is not unique because of choices allowed during the depth-first traversal. Stated another way, even if T is kept in a data structure that specifies a particular left and a right child for each non-leaf node, the choice of which child is left and which is right may differ in any two separate copies of T. Therefore even if the traversal was required to be an in-order traversal (left child visited before the right child), two different strings could be created for the two different copies of T. This gives rise to the problem addressed on the net, of determining if two New Hampshire codes, which are not identical strings, actually specify the same tree.

   One approach is to pick a "canonical" New Hampshire encoding with the property that any tree can be encoded in only one way. If an arbitrary New Hampshire encoding for the tree can be efficiently converted to the canonical one, then the problem of comparing two codes is solved by converting each to its respective canonical code and then checking for equality of the codes. An obvious canonical encoding is the "lexicographic-least" one. For example, the code above would be *(((A,C),E),(B,D))*.

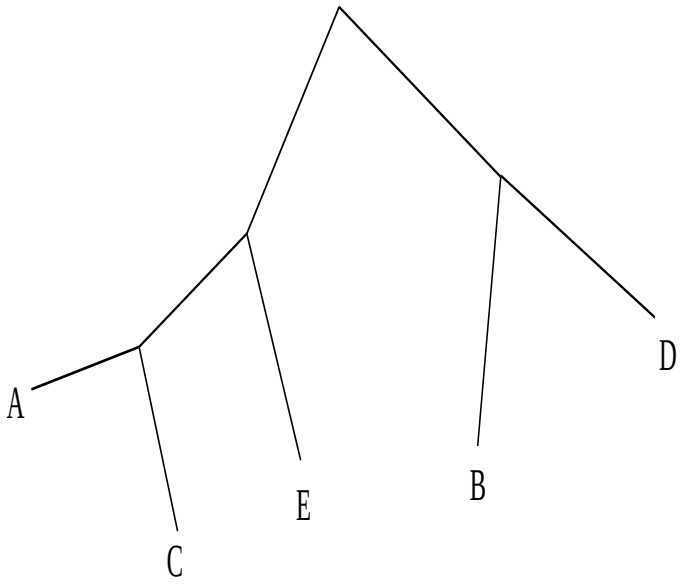

Figure 1: Tree for NH code.

Give a precise definition of "lexicographic-least" so that each tree has a unique lexicographic-least New Hampshire encoding. Then give a simple linear-time algorithm that converts any New Hampshire code to its lexicographic-least one. The fact that the labels are distinct is helpful.

5. Do exercise 5.6.4 on p. 99 of the notes on Split Networks.

6. I think the last paragraph on p. 100 in the proof that a Buneman graph is a Split Network, is unneeded. That is, the proof would work just fine if we removed that paragraph entirely. Do you agree? Justify your answer.

7. Do exercise 5.7.3 on p. 103

8. Do Exercise 7.1.5 (The convex hull algorithm computes a Buneman graph)

9. Write up the BFS algorithm to find a Buneman graph that Fumei outlined in class Nov. 10. If you missed that class, ask about it. Then comparing it to the convex hull algorithm in the book, discuss

advantages or disadvantages for either algorithm in comparison to the other? For example, which might be more efficient in practice or worst case, or simpler to program.

10. Median graph problem - there will be one more problem that will be posted later. If not posted soon, it will be part of the next homework.