CSE 620: Project 4
Distance-Based Phylogenetic Tree Reconstruction
Due: Oct. 11, 2013

For this project you will be adding code to your proj4/PhyloTreeAlg.py file. There will be no unit testing code provided, but the file **project4/PhyloTree_util.py** provides a number of functions that may be useful to you in debugging.

You will also need a working copy of **PhyloTree.py**. Copy yours over from project 2 if you are confident it is working. If not, you may instead copy over (and rename) the **PhyloTree_sol.py** solution distributed in that project.

For both problems: your implementations are expected to by asymptotically efficient, but you will not be graded on speed past that constaint.

Problem 1: Implement the code for the UPGMA algorithm to reconstruct a phylogenetic tree from a distance matrix representing leaf distances for an ultrametric full binary tree.

UPGMA algorithm:

1. For each leaf label *i* create a node n_i and assign it a height of 0.
2. For each node n_i create a set $C_i = \{n_i\}$
3. While there is more than one node left:
    1. Find the *i,j* that minimizes *DIST(i,j).*
    2. Create a new node n_k and assigned it a height of *½Dist(i,j).*
    3. Let $C_k = C_i \cup C_j$.
    4. Assign n_i as a child of n_k with an edge whose length is the difference in the heights.
    5. Assign n_j as a child of n_k with an edge whose length is the difference in the heights.
    6. For each $z \neq i,j,k$, calculate: $DIST(k,z) = \frac{|C_i|DIST(C_i,C_z)+|C_j|D(C_j,C_z)}{|C_i|+|C_j|}$
    7. Remove n_i and n_j from the list of candidate nodes and add n_k.

Problem 2: Implement the code for the Neighbor-Joining (NJ) algorithm to reconstruct a phylogenetic tree from a distance matrix representing leaf distances for a full binary tree.

Note: while your output will conceptually be an unrooted binary tree, this is difficult to represent in a PhyloTree object geared towards the representation of a root tree. The results will be that you have a root containing three children, every other node has either two or zero children, and the "root" of the tree is an arbitrary node that has no relation to the actual phylogenetic root.

NJ algorithm:

1. Create a "star" graph: a graph with a node n_i for each leaf label *i*, and a node *v* (which does not match to any leaf) such that there is an edge (v,n_i) for each *i*.
2. While *v* is incident to more than three edges:
    1. For each node *x* adjacent to *v*: compute $r_x = \frac{\sum_{w \in S} DIST(x,S)}{|S|-2}$ where *S* is the set of all nodes adjacent to *v*.
    2. Find the *i,j* that minimize $M(i,j) = DIST(i,j) - (r_i + r_j)$
    3. Remove the edges from *i* to *v* and *j* to *v*, connect them each to a knew node *k*, and connect *k* to *v*.
    4. Calculate $DIST(i,k) = \frac{1}{2}(DIST(i,j) + r_i - r_j$
    5. Calculate $DIST(j,k) = DIST(i,j) - DIST(i,k)$
    6. For each node $z \neq i,j,k$: $\frac{1}{2}(DIST(k,z) = DIST(i,z) + DIST(j,z) - DIST(i,j))$
3. Calculate distance for the remaining three edges incident to *v*, using the formula $DIST(i,v) = \frac{1}{2}(DIST(i,j) + DIST(i,k) - DIST(i,j))$, where *i*, *j* and *k* are the remaining nodes adjacent to *v*.

Comment: The above description is the standard conceptual description. Depending on how you approach the implementation, the specific details may vary. (e.g. You might not find it necessary to actually maintain the star tree – which is fine.)