Introduction to Bioinformatics Algorithms
Homework 6

Saad Mneimneh
Computer Science
Hunter College of CUNY

Problem 1: Suffix trees
Describe how you can find in linear time the following, using a suffix tree data structure:
(a) a longest match between x and y

(b) a longest **unique** match between x and y if it exists

(c) a longuest repeat in x

(d) a longest **non-overlapping** repeat in x

Problem 2: Parsimony
The following dynamic programming algorithm solves the maximum parsimony problem for a general distance criterion and one character.

$$f_v(a) = 0, v \text{ is a leaf labeled } a$$

$$f_v(a) = \infty, v \text{ is a leaf not labeled } a$$

$$f_v(a) = \sum_{w \in \delta(v)} \min_{b \in A} [f_w(b) + d(a,b)]$$

$$M_w(a) = \{b : f_w(b) + d(a,b) \text{ is minimal}\}$$

We seek $\min_{a \in A} f_{root}(a)$ and M can be used for backtracking.

(a) What is the running time and space requirement for this algorithm? Assume $|A| = r$ (r states), we have n leaves (objects), and m characters.

(b) Adapt this algorithm to the special case when the distance is defined as (obtain better running time):

$$d(a,b) = \begin{cases} 0 & a = b \\ 1 & a \neq b \end{cases}$$

Problem 3: Perfect phylogeny (optional)
Consider binary characters and let 1_i be the set of objects that have state 1 for character i. Define 0_i similarly.

(a) Consider the following character state matrix:

	c_1	c_2
A	0	1
B	1	1
C	1	0

According to the condition stated in class, namely that 1_i and 1_j are either disjoint or one is a subset of the other, a perfect phylogeny does not exist. However, the colored graph produced by the state matrix is acyclic, implying that a perfect phylogeny does exist. Which interpretation is correct?

(b) Consider ordered undirected (the state tree is unrooted) non-binary characters. Show that for each character i, the state tree can be rooted in such a way that for each binary factor j, $|0_j| \geq |1_j|$. *Hint*: root the tree arbitrarily, then reverse edges that violate the condition, and argue that you would still have a tree.

Problem 4: Viterbi (optional)
A loaded die has probability 1/2 for 6. The probability of switching to a loaded die is 0.05. The probability of switching to a fair die is 0.1. We observe the following:

```
651166453132651245366646316366631623264552362666666625151631
```

Which parts of this sequence are generated by the loaded die? Use the Viterbi algorithm to figure this out. Use the log transformation in your implementation.