CS 122A HW 5, Due Friday Nov. 5, 2010

1. In the DP solution to the weighted interval problem the book explains how to do the traceback using an algorithm to select a set of intervals (see the book). However, in class I tried to explain how to leave pointers during the forward computation (when the OPT(j) values are determined) so that the solution (an optimal set of intervals) could be found just by following a path of pointers from cell n to cell 0 in the OPT table. However, one has to be careful in how you select the set of intervals from the path of pointers, and several ideas were mentioned in class. Things got a bit muddled in class. (If you didn't come to class you can catch it on reruns).

Figure out a correct way to set and use pointers in the forward computation, so that the optimal set of intervals can be determined from them. Be precise in how you use the pointers to determine the intervals, and justify your answer. In other words, write up that part of the lecture in a cleaner, more correct and less confusing way.

2. The question was raised in class as to why we need a traceback anyway. Can't we just select the intervals during the forward computation? In particular, at the moment we set $OPT(j)$ can't we choose to take interval j into the set of intervals if and only if $OPT(p(j)) + v(j) \geq OPT(j-1)$? Wouldn't this always give us an optimal set of intervals? If you say yes, give a proof, and if you say no, give an example to show why.

3. The secondary structure problem discussed in section 6.5 in the book seeks to find the secondary structure that maximizes the number of base pairs it contains, i.e. the number of base pairs that are in the matching. Review the definitions of "secondary structure" on page 274.

Now suppose we modify the problem so that you get a value of 1 when you match an A to a U, and you get a value of 3 when you pair a C and a G, and you get a value of 4 when you pair a C and a U (in this version of the problem you are allowed to pair a C and a U). All other aspects of the problem, and the definition of a secondary structure, are the same as before. The problem now is to find a secondary structure (with the modification that C can pair with U) that has the maximum total value.

Show how to modify the recurrences for the basic secondary structure problem to solve this new problem. Write out the pseudo-code to solve the recurrences by DP, and analyze the worst-case running time of this DP solution.

4. Suppose we are given a rooted tree T with n leaves and m non-leaf nodes. Each leaf is colored with one of $k < n$ given colors, so several leaves can have the same color. We need to color each interior node of T with one of the k given colors to *maximize* the number of edges whose (two) endpoints are colored the same color.

We can solve this with a DP algorithm that runs in $O(mk)$ time. Let $V(v, i)$ denote the optimal solution value when the problem is applied to the subtree rooted at node v, and v is required to be given color i. Let $V(v)$ denote the optimal solution value when the problem is applied to the subtree rooted at node v, and there is no restriction on which of the k colors v can be.

4a. Using that notation, develop recurrences for this problem, and explain the correctness of your recurrences.

4b. Expalin how the recurrences are are evaluated (solved) in an efficient DP way.

4c. Show that the time bound for your DP is $O(mk)$.