CSci 421
Introduction to Algorithms

Homework Assignment 4
Due: Friday, Feb 2, 2001

Winter 2001 Handout 4

W. L. Ruzzo January 26, 2001

Reading Assignment:

By now you should have finished all of chapters 1–3 and 5. Also read:

- Material on dynamic programming: Stamp problem (lecture notes), 5.10, 6.8 (+ lecture notes).
- Material on Greedy Algorithms: Fractional knapsack (lecture notes), 6.6, 7.6.
- Next: Chapter 7, Graph Algorithms.

Homework:

1. Run the Knapsack algorithm (Fig 5.10, pg 110) on the sequence of weights $k_1 = 5$, $k_2 = 2$, $k_3 = 4$, $k_4 = 3$, and $k_5 = 6$], with knapsack capacity $K = 16$. Show a table like Fig 5.11 to summarize the computation.
2. Problem 5.19.
3. Run the string alignment algorithm given in lecture (similar to Fig 6.27, pg 158) on strings $S = tcatag$ and $T = tataag$. Build the cost matrix and traceback pointers as in the example given in lecture. Assume that aligning two identical letters gives a score of $+2$, whereas aligning a letter with a mismatched letter or a gap gives a score of -1.
4. You are given a binary tree with n leaves, with the ith leaf labeled by a letter S_i from a fixed alphabet Σ. You are also given a function c, which assigns a cost $c(S, S') \geq 0$ to each pair of letters S, S' in Σ. Assume $c(S, S') = c(S', S)$ and $c(S, S) = 0$ for all $S, S' \in \Sigma$. The problem is to give each internal node x of the tree a label $S_x \in \Sigma$ so as to minimize the total over all tree edges of the cost of that edge, where the cost of an edge whose end points are labeled U and V is $c(U, V)$. Give an efficient algorithm to solve this problem, and analyze its running time as a function of n and $|\Sigma|$. Assume that each evaluation of the cost function c costs $O(1)$ operations; e.g., via table lookup using a table built in to your algorithm. Hint: use dynamic programming. As we've seen before, you might need a stronger induction hypothesis; i.e., you'll calculate more at each internal node than just whether to put letter "S" there. For instance, it might help to know the total cost of the subtree rooted there, assuming that it's labeled "S".

   [Although it doesn't matter for purposes of this homework, this algorithm is sometimes used in estimating evolutionary trees. The S_i are letters at corresponding positions in DNA or protein sequences from n different species, the tree is their presumed evolutionary tree, and the goal is to try to reconstruct the corresponding letters in the ancestral sequences.]