Homework 2
Due at 11:59:59pm, Wednesday, October 1, 2008

• For this and all future homeworks, groups of up to three students may submit a single, common solution. Please neatly print (or typeset) the full name, NetID, and alias (if any) of every group member on the first page of your submission.

• We will use the following point breakdown to grade dynamic programming algorithms: 60% for a correct recurrence (including base cases), 20% for correct running time analysis of the memoized recurrence, 10% for correctly transforming the memoized recursion into an iterative algorithm.

• A greedy algorithm must be accompanied by a proof of correctness in order to receive any credit.

1. (a) Let $X[1..m]$ and $Y[1..n]$ be two arbitrary arrays of numbers. A common supersequence of $X$ and $Y$ is another sequence that contains both $X$ and $Y$ as subsequences. Describe and analyze an efficient algorithm to compute the function $scs(X, Y)$, which gives the length of the shortest common supersequence of $X$ and $Y$.

(b) Call a sequence $X[1..n]$ of numbers oscillating if $X[i] < X[i+1]$ for all even $i$, and $X[i] > X[i+1]$ for all odd $i$. Describe and analyze an efficient algorithm to compute the function $los(X)$, which gives the length of the longest oscillating subsequence of an arbitrary array $X$ of integers.

(c) Call a sequence $X[1..n]$ of numbers accelerating if $2 \cdot X[i] < X[i-1] + X[i+1]$ for all $i$. Describe and analyze an efficient algorithm to compute the function $lxs(X)$, which gives the length of the longest accelerating subsequence of an arbitrary array $X$ of integers.

2. A palindrome is a string that reads the same forwards and backwards, like x, pop, noon, redivider, or amanaplanacatahama yakayamahatacanalpanama. Any string can be broken into sequence of palindromes. For example, the string bubbbaseesabanana (‘Bubba sees a banana.’) can be broken into palindromes in several different ways; for example:

   
   bub + baseesab + anana  
   b + u + bb + a + sees + aba + nan + a  
   b + u + bb + a + sees + a + b + anana  
   b + u + b + b + a + s + e + e + s + a + b + a + n + a + n + a

Describe and analyze an efficient algorithm to find the smallest number of palindromes that make up a given input string. For example, given the input string bubbbaseesabanana, your algorithm would return the integer 3.

3. Describe and analyze an algorithm to solve the traveling salesman problem in $O(2^n \text{poly}(n))$ time. Given an undirected $n$-vertex graph $G$ with weighted edges, your algorithm should return the weight of the lightest Hamiltonian cycle in $G$, or $\infty$ if $G$ has no Hamiltonian cycles. [Hint: The obvious recursive algorithm takes $O(n!)$ time.]
4. Ribonucleic acid (RNA) molecules are long chains of millions of nucleotides or bases of four different types: adenine (A), cytosine (C), guanine (G), and uracil (U). The sequence of an RNA molecule is a string \( b[1..n] \), where each character \( b[i] \in \{A, C, G, U\} \) corresponds to a base. In addition to the chemical bonds between adjacent bases in the sequence, hydrogen bonds can form between certain pairs of bases. The set of bonded base pairs is called the secondary structure of the RNA molecule.

We say that two base pairs \((i, j)\) and \((i', j')\) with \(i < j\) and \(i' < j'\) overlap if \(i < i' < j < j'\) or \(i' < i < j' < j\). In practice, most base pairs are non-overlapping. Overlapping base pairs create so-called pseudoknots in the secondary structure, which are essential for some RNA functions, but are more difficult to predict.

Suppose we want to predict the best possible secondary structure for a given RNA sequence. We will adopt a drastically simplified model of secondary structure:

- Each base can be paired with at most one other base.
- Only A-U pairs and C-G pairs can bond.
- Pairs of the form \((i, i + 1)\) and \((i, i + 2)\) cannot bond.
- Overlapping base pairs cannot bond.

The last restriction allows us to visualize RNA secondary structure as a sort of fat tree.

![Example RNA secondary structure with 21 base pairs, indicated by heavy red lines. Gaps are indicated by dotted curves. This structure has score \(2^2 + 2^2 + 8^2 + 1^2 + 7^2 + 4^2 + 7^2 = 187\).](image)

(a) Describe and analyze an algorithm that computes the maximum possible number of base pairs in a secondary structure for a given RNA sequence.

(b) A gap in a secondary structure is a maximal substring of unpaired bases. Large gaps lead to chemical instabilities, so secondary structures with smaller gaps are more likely. To account for this preference, let's define the score of a secondary structure to be the sum of the squares of the gap lengths. Describe and analyze an algorithm that computes the minimum possible score of a secondary structure for a given RNA sequence.

---

1This score function has absolutely no connection to reality; I just made it up. Real RNA structure prediction requires much more complicated scoring functions.
5. A subtree of a (rooted, ordered) binary tree $T$ consists of a node and all its descendants. Design and analyze an efficient algorithm to compute the largest common subtree of two given binary trees $T_1$ and $T_2$; this is the largest subtree of $T_1$ that is isomorphic to a subtree in $T_2$. The contents of the nodes are irrelevant; we are only interested in matching the underlying combinatorial structure.

Two binary trees, with their largest common subtree emphasized

*6. [Extra credit] Let $D[1..n]$ be an array of digits, each an integer between 0 and 9. A digital subsequence of $D$ is a sequence of positive integers composed in the usual way from disjoint substrings of $D$. For example, 3, 4, 5, 6, 23, 38, 62, 64, 83, 279 is an increasing digital subsequence of the first several digits of $\pi$:

$$\begin{array}{cccccccccccc}
3 & 1 & 4 & 1 & 5 & 9 & 6 & 2 & 3 & 4 & 3 & 8 & 4 & 6 & 2 & 6 & 4 & 3 & 3 & 8 & 3 & 2 & 7 & 9
\end{array}$$

The length of a digital subsequence is the number of integers it contains, not the number of digits; the previous example has length 10.

Describe and analyze an efficient algorithm to compute the longest increasing digital subsequence of $D$. [Hint: Be careful about your computational assumptions. How long does it take to compare two $k$-digit numbers?]