Algorithms in the Real World (15-853), Fall 04
Assignment #4

Due December 1. Do 5 of the 6 problems.

Problem 1: 10pt
Recall that in the preprocessing step of the KMP algorithm for string matching, we compute values $l(i)$ for a string S. In the following we assume sequences are indexed starting at 1. $l(i)$ is defined to be 1 plus the length of the longest suffix of $S[2..i-1]$ that matches a prefix of S. We define $l(1) = 0$.
For example, for the string `abcadabd`, l is given by $(0, 1, 1, 1, 2, 1, 2, 3)$.
The following is incomplete code for computing $l(i)$. The variable i tracks the current position for which we are computing $l(i)$. j tracks the position of the character that we match against $i-1$. Fill in the two missing lines.

```
l[1] = 0;
j = 0;
for (i = 2; i <= n; i++) {
   while ((j > 0) && (s[i-1] != s[j]))
      j = _________ ;
   j = j + 1;
   l[i] = _________ ;
}
```

Argue (briefly and cleanly) that the total number of iterations of the while loop across all iterations of the for loop is bounded by n. The runtime of the routine is hence $O(n)$.

Problem 2: 10pt
Given two strings S_1 and S_2 and a text T, you want to find whether there is an occurrence of S_1 and S_2 interwoven in T, possibly with spaces. For example, the strings `abac` and `bbc` occur interwoven in `cabcbabcca`. Give an efficient algorithm for this problem (i.e. one that is polynomial in the size of the inputs).

Problem 3: 10pt
Consider the following gap model – each insertion or deletion costs a unit. However, if there are more than k consecutive insertions, or k consecutive deletions, they cost only k units. Give an algorithm that finds the minimum edit distance under this cost model in time $O(nm)$. Note that the time should not depend on k. (Do not worry about space efficiency).

Problem 4: 10pt
The *shortest superstring problem* is the problem of finding the shortest string that contains all given strings as its substrings. Formally, given a set of m strings $S_1, \cdots, S_m$, find the shortest string T such that each S_i is a substring of T.
Reduce this problem to a Traveling Salesman Problem. The reduction needs to take polynomial time (in $n = \sum_{i=1}^{m} |S_i|$). For extra credit prove that the shortest superstring problem is NP-hard.

Problem 5: 10pt
The main heuristic behind FAST and BLAST is that of finding small exact matches and using them to compose larger approximate matches. This heuristic is partially justified by the following lemma.

Lemma 1 *Suppose that $a_1, \ldots, a_t$ and $b_1, \ldots, b_t$ can be aligned with at most l mismatches (i.e., $a'_i \neq b'_i$), then for $k \leq \lfloor \frac{t}{l+1} \rfloor$, $a_1, \ldots, a_t$ and $b_1, \ldots, b_t$ share at least $t - (l+1)k + 1$ k-tuples (possibly overlapping).*

Prove the lemma, and show how it can be used to rapidly search all sequences in a database which can be aligned with a given query.

Problem 6: 10pt
Describe how Hirshberg's linear space algorithm can be used for local alignment if all you care about is the one alignment with maximum score.