CISC 621 Algorithms, Homework due Tue, May 17, 2011 (accepted until Fri, May 20)

1. We have seen that edit distance, the minimal number of inserts, deletions, substitutions to convert string S to string T, can be found in O(mn) time, where m is length of S, n is length of T. We have also seen that the longest common substring of S and T can be found in O(mn) time.

   The longest approximately matching substring problem is LAMS(S, T, k), to find the longest substrings S' of S and T' of T such that the edit distance of S' and T' is at most k. For convenience we will stipulate that S' and T' be of the same length. Recall that a substring is a contiguous subsequence (adjacent elements) in the parent string. Furthermore, to simplify this exercise, and we will will only allow edit steps which are substitutions (replacement of a letter in S' corresponding letter in T'). Thus the edit distance between two strings S' and T' of the same length is the number of positions in which they differ.

   (a) Design and explain an algorithm to solve the LAMS(S,T,k) problem in O(mnk) time, where m = size(S) and n = size(T)..

   (b) LAMS is relevant to genomics, where long approximately matching substrings represent shared genes. Here not only the longest is of interest. Design and explain an algorithm on (S, T, m, k) to find all (i,j) such that the substrings S[i]..S[i+m-1] and T[j]..T[j+m-1] have edit distance at most k. Incidentally, cases of interest have the lengths of S and T in the millions, m in the thousands, and k small (tens).

2. Longest bimodal subsequence: Give a sequence of numbers S of length n, find the longest subsequence that is bimodal. It suffices to return the *length* of a longest bimodal subseqence. Bimodal means that the subsequence first increases then decreases (or vice versa), but it reverses direction only once. Thus (1,2,3,4,3,2) is bimodal, as is (4,3,2,1,2,5), but (2,3,4,3,4) is not. The longest bimodal subsequence of that last one has length 4, and is (2,3,4,3) or (2,3,4,4). Design and explain an efficient algorithm to find a longest bimodal subsequence.

   Hint: Longest increasing subsequence (LIS) has the following recursive solution which leads to a dynamic programming solution using O(n) space and O(n^2) time.

   int LIS(S) n = size(S); return $\max_{j=1..n} LIS(S, j)$;

   int LIS-end(S, j) // return length of longest increasing subsequence ending at j-th position. if (j == 1) return 1; // find max you can get by LIS-end to i followed by single step to j. ans = 1; for (int i = 1; i ¡ j; ++i) if (S[i] ¡= S[j]) ans = max(ans, 1+LIS-end(S,i)); return ans;

3. Extra credit problem: We have studied two algorithms dealing with sets of points in the (x,y) plane, namely convex hull and nearest pair of points.

This exercise is to find a *farthest* pair of points between two sets of points in the plane. Farthest(S, T) takes sequences of points S and T. It returns the maximum of dist(S[i], T[j]), for i in 1..size(S) and j in 1..size(T). Find an algorithm for Farthest(S,T) that runs in O(n log(n)) time, for n = size(S) + size(T).

4. Matrix hole problem is discarded.

5. 3SAT variants. Do 2 of the 3 parts.

   (a) Consider 3SAT-2l, the version of 3SAT in which each literal occurs at most 2 times. Thus the input is a conjunction of n clauses, each clause of length 3. There are 3n occurences of literals. Each literal occurs at most twice in the input. Thus if x is a variable, x is at most 2 of the 3n literal occurences and $\bar{x}$ is at most 2 of the 3n literal occurences. Saying that again: A literal is either a variable or it's negation, so we are considering the input to be a conjunction of clauses (of 3 literals each) in which a variable x occurs at most twice without negation and at most twice with negation (for a total of at most 4 times for the variable). For example $(x, y, z), (\bar{x}, w, u), (\bar{x}, \bar{w}, z)$ is a valid input since the literals $\bar{x}$ and u occur twice and all other literals occur once. Show that 3SAT-2l is NP-complete.

       Hint: use a reduction in which each instance of a variable is replaced by a new variable and clauses are added to link the new variables to behave as if they where instances of the original variable. Thus the reduction might be somewhat in the spirit of the reduction of SAT to 3SAT (3-CNF satisfiability, page 1082).

   (b) Consider 3SAT-1l, the version of 3SAT in which each literal appears at most once. (Thus each variable might occur twice, once negated and once not.) Example input: $(x, y, \bar{z}, (\bar{x}, \bar{w}, z)$. Solve 3SAT-1l in polynomial time.

   (c) Consider 3SAT-e3v, the version of 3SAT in which each *variable* appears exactly three times and each clause has exactly three literals. (For a given variable, it might be once negated, twice not-negated, or vice versa, or all 3 times negated or all 3 times not negated.) Example input: $(x, y, z), (x, \bar{y}, w), (\bar{x}, w, z), (z, w, \bar{y})$. Solve 3SAT-e3v in polynomial time.

6. Binary search trees and Trieps.

   (a) Draw a binary tree of maximum height such that it's nodes have average depth of exactly 3. The depth of the root is 0.

   (b) Problem 13-4, parts a, b, c, d only.