EECS 336: Design and Analysis of Algorithms
Weekly Problem Set #2

Class Homepage: www.cs.northwestern.edu/~kao/eecs336-algorithms/index.htm

Posted on the Class Homepage: Tuesday, October 7, 2014.

Due Time: the start of class on Tuesday, October 14, 2014.

Policy for This Problem Set: Different problem sets may have different policies. This problem set is to be done by one student singly. To answer the questions in this problem set, you may consult your textbook, your lecture notes, the Internet, and any materials that you can find in libraries. You may also discuss solution ideas for these questions with the instructor or the teaching associates, but no one else. You may not copy answers from other people, including those from your fellow students or those posted on the Internet. If you copy all or portions of your answers from other people, you will receive 0 point for the entire problem set. If two students have identical or essentially identical answers but the original sources of the answers cannot be determined, both students will receive 0 point for the entire problem set.

Questions: There are 3 questions.

1. (40 points) This question modifies Exercise 15.4-6 on page 397. We define a *double-increasing sequence* to be a sequence of numbers $x_1, x_2, \ldots, x_p$ with $p \geq 2$ such that there exists an index k with $1 \leq k \leq p-1$ for which $x_1, x_2, \ldots, x_k$ is monotonically increasing and $x_{k+1}, x_2, \ldots, x_p$ is also monotonically increasing. For instance, the sequence 10, 19, 8, 12, the sequence 10, 19, 28, 33, and the sequence 10, 19, 28, 28 are double-increasing sequences but the sequence 10, 19, 8, 2 and the sequence 10, 19, 8, 8 are not.

   Give a little-o(n^2)-time algorithm to find the longest double-increasing subsequence of a sequence of n numbers. Prove the correctness of your algorithm and analyze the time complexity of your algorithm.

   Partial Credit: You will receive 25 points instead of 40 points if your algorithm has a time complexity of big-O(n^2) but not little-o(n^2).

2. (30 points) This question is a special case of the longest common subsequence problem in Section 15.4. Give an O(Kn)-time O(n)-space algorithm for the following algorithmic problem. Prove the correctness of your algorithm, and analyze the time complexity and space complexity of your algorithm.

   Input: a positive integer K and two sequences $X = x_1x_2x_3 \ldots x_n$ and $Y = y_1y_2y_3 \ldots y_n$ both of length n.

   Output: "Yes" if the length of the longest common subsequence of X and Y is at least n – K; "No" otherwise.

3. (30 points) Give an algorithm for the following algorithmic problem. Your algorithm should run in polynomial time. Prove the correctness of your algorithm and analyze the time complexity of your algorithm.

   Input: a positive integer K and an undirected graph G where each edge has a positive weight.

   Output: for each pair of vertices x and y, a shortest path among all paths between x and y that have exactly K edges.