COMP 3711 Design and Analysis of Algorithms
Fall 2015
Assignment 3

1. Give an $O(n^2)$-time dynamic programming algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers (i.e, each successive number in the subsequence is greater than or equal to its predecessor). For example, if the input sequence is $\langle 5, 24, 8, 17, 12, 45\rangle$, the output should be either $\langle 5, 8, 12, 45\rangle$ or $\langle 5, 8, 17, 45\rangle$.

2. Given a set $S = \{a_1, a_2, \cdots, a_n\}$ of n positive integers and a positive integer W, design an $O(nW)$-time dynamic programming algorithm to decide whether there is a subset $K \subseteq S$ such that the sum of all numbers in K is W. For example, if $S = \{1, 4, 7, 3, 5\}$ and $W = 11$, then the answer is "yes" since K can be $\{4, 7\}$. (Your algorithm doesn't have to output the subset K.) If no such subset exists, your algorithm should return "no".

3. A palindrome is a string that reads the same backward or forward. Design a dynamic programming algorithm that takes a string $x[1..n]$ and returns the longest palindromic subsequence. Its running time should be $O(n^2)$.

   For example, for the string `ACGTGTCAAAATCG`, its longest palindromic subsequence is `GCAAAACG`.

   Note that a "smart" solution to this problem (actually given by some students in previous years) is to simply find the longest common subsequence of the string and its reverse. This is unfortunately not correct. For example, for the string `ACBAC`, the longest common subsequence of `ACBAC` and its reverse `CABCA` is `ABC`, but it is not palindromic.

4. This problem is a simplified version of the "Spread of Influence" problem studied by Kempe, Kleinberg, and Tardos. We use a directed graph $G = (V, E)$ to model a social network, where each vertex represents an individual. Each vertex v is associated with a threshold $t(v)$, and each edge (u, v) has a weight $w(u, v)$, which means that u has an influence of $w(u, v)$ on v. We initially set a given vertex r to be *active* (say, adopting a new product). Then r starts to spread his/her influence to its neighbors. When the total influence a vertex v receives from all its incoming neighbors is greater than or equal to its threshold $t(v)$, v also becomes active. Your job is to design an algorithm that, given an starting vertex r, computes the number of vertices that are eventually active. For full credits, your algorithm should run in $O(V + E)$ time. All edge weights and thresholds are nonnegative.

   Please check out

   `http://course.cse.ust.hk/comp3711/homework/Spread-of-Influence.ppt`

   for some background and an example.