Assignment 1.

Post date: Sept. 16, 2014.
Due date: Sept. 23, 2014 by noon time. Hand in printed paper at assignment box at 4th floor of MC across from the tutorial centre.

Question 1. (10 marks)

Deal with the "many liars" problem (see course notes) when the total number of people is odd. Explain your modification to the algorithm explained in the class, and argue that modification is sufficient to solve the problem.

Question 2. (10 marks)

Design an efficient dynamic programming algorithm to compute the longest increasing subsequence. Details follow:

Longest Increasing Subsequence (LIS): Given a sequence of numbers $a_1, a_2, \dots, a_n$. An increasing subsequence of the input sequence is a subsequence $a_{i_1}, a_{i_2}, \dots, a_{i_k}$ such that $i_1 < i_2 < \cdots < i_k$, $a_{i_1} < a_{i_2} < \cdots < a_{i_k}$. The problem asks for the longest increasing subsequence.

(a) Let $D[i]$ be the length of the LIS that ends **at** a_i (including a_i). Derive a recurrence relation to compute $D[i]$ from $D[k]$ for $k < i$. Prove the correctness of the recurrence relation.

(b) Based on the above recurrence relation, design an $O(n^2)$ dynamic programming algorithm for computing LIS. A pseudo code must be provided, and a proof of time complexity should be included. The pseudo code needs to include the initialization and computation of the DP table, as well as the backtracking steps (not just claiming that a standard backtracking will work).

Note: An $O(n \log n)$ time algorithm exists for the LIS problem. Essentially, that algorithm maintains an auxiliary array that helps to speed up the recurrence relation calculation in (a). That auxiliary array needs to be dynamically updated during the dynamic programming, which makes the proof a little tedious to write.