# CS 681 Fall 2001 – Problem Set 5

**Due in class Monday, November 19**

**(1)** Consider a very simple on-line auction system that works as follows. There are $n$ *bidding agents*; agent $i$ has a bid $b_i$, which is a positive natural number. We will assume that all bids $b_i$ are distinct from one another. The bidding agents appear in an order chosen uniformly at random, each proposes its bid $b_i$ in turn, and at all times the system maintains a variable $b^*$ equal to the highest bid seen so far. (Initially $b^*$ is set to 0.)

What is the expected number of times that $b^*$ is updated when this process is executed, as a function of the parameters in the problem?

**Example:** Suppose $b_1 = 20$, $b_2 = 25$, and $b_3 = 10$, and the bidders arrive in the order $1, 3, 2$. Then $b^*$ is updated for 1 and 2, but not for 3.

**(2)** A problem that arises in the mapping of genomes is to order a collection of markers on a chromosome subject to "between-ness" constraints — certain markers are known (or at least believed) to lie between certain pairs of others.

As an abstraction of this problem, suppose we are given a set of $n$ markers $m_1, m_2, \ldots, m_n$; our goal is to produce a linear ordering of these markers. We are also given $C$ constraints, each represented by an ordered triple $(m_i, m_j, m_k)$. Such a constraint says that in the ordering we produce, $m_j$ must come *between* $m_i$ and $m_k$. (The constraint does not specify which of $m_i$ or $m_k$ should come first, only that $m_j$ should be in the middle.)

Now, it is not always possible to satisfy all constraints simultaneously, so we wish to produce an ordering that satisfies as many as possible. Unfortunately, deciding whether there is an ordering that satisfies at least $C'$ of the $C$ constraints is an NP-complete problem (you don't have to prove this.)

Give a constant $\varepsilon > 0$ (independent of $n$) and an algorithm with the following property. If it is possible to satisfy $C^*$ of the constraints, then the algorithm produces an ordering of markers satisfying at least $\varepsilon C^*$ of the constraints. You can provide either a deterministic algorithm running in polynomial time, or a randomized algorithm; for a randomized algorithm, you should give a polynomial bound $p(n)$ such that after $p(n)$ steps, the algorithm will have produced an ordering with the desired property with probability at least $1 - \frac{1}{n}$.