CS 577: Introduction to Algorithms	Homework 2
Out: 02/14/12	**Due: 02/21/12**

Ground Rules

Same as for homework 1.

Problems

1. **(4 points)** Problem 5.1 in the textbook (Pg. 246).

2. **(4 points.)** Problem 13.9 in the textbook (Pg. 788–789).

   Hint: Note that you don't want to accept the first few bids, because you don't get a chance to compare them to other bids, and the highest one is unlikely to be among the first few. Furthermore, you should use the information in the first few bids to later reject bids that you know are not the highest bid (e.g. those that are lower than some bid you rejected earlier). Note that it is possible to get a success probability better than $1/4$ but you only need to prove a $1/4$ for this homework.

3. **(4+3 points.)** Randomness can be very useful when we want to verify the correctness of a program. In this question we will see a basic example of how randomness helps in generating good test cases. Suppose we are given $n \times n$ matrices A, B, and C, and wish to check whether $AB = C$. One way to do this is to just multiply A and B out and compare the result to C. This would take $O(n^3)$ time[1]. We will now develop an $O(n^2)$ time test that has high accuracy.

   (a) Let r be an $n \times 1$ random vector, where each entry is independently 0 with probability $1/2$ and 1 with probability $1/2$. Prove that for two $n \times n$ matrices X and Y with $X \neq Y$, the probability that $Xr = Yr$ is at most $1/2$. Note that X and Y are fixed matrices, not random matrices.

   Hint: Consider the non-zero matrix $(X - Y)$ and prove that Pr$[(X - Y)r = 0]$ is less than $1/2$.

   (b) Use the above to design an $O(n^2)$ time randomized algorithm for checking whether $AB = C$, that returns the correct answer with probability at least $1/2$. More precisely, if $AB = C$ is correct, your algorithm should output "correct" with probability 1, and if it is not, then your algorithm should output "incorrect" with probability at least $1/2$.

4. **(Extra credit.)** In this question, you are given n points each picked independently uniformly at random from the two-dimensional unit square $[0, 1]^2$. We say that a point (x_1, y_1) dominates another one (x_2, y_2), if $x_1 > x_2$ and $y_1 > y_2$. A point is undominated if no other point dominates it. Compute the expected number of undominated points.

Note: Since a few people in class do not have the textbook yet, we have reproduced the questions from the book on the back of this page.

[1] Or $O(n^{2.81})$ time using fast matrix multiplication (Strassen's algorithm)