6.895 Randomness and Computation	February 21, 2006
Homework 1	
Lecturer: Ronitt Rubinfeld	*Due Date: March 6,2006*

Homework guidelines: You may work with other students, as long as (1) they have not yet solved the problem, (2) you write down the names of all other students with which you discussed the problem, and (3) you write up the solution on your own. No points will be deducted, no matter how many people you talk to, as long as you are honest. If you already knew the answer to one of the problems (call these "famous" problems), then let me know that in your solution writeup – it will not affect your score, but will help me in the future. It's ok to look up famous sums and inequalities that help you to solve the problem, but don't look up an entire solution.

The following problems are just for your own edification, and are not to be turned in:

1. (This one is intended to make sure you understand Chernoff bounds). Show that given any Self-Corrector A that runs in time $T(n)$ on inputs of size n with probability of error 1/3, one can convert it into a Self-Corrector that runs in time $O(T(n)\log 1/\beta)$ with probability of error at most β.

2. Given a coin with probability p of getting "heads", give a procedure for simulating one toss of a fair coin ($p = 1/2$). The procedure should run in expected time that is polynomial in $\frac{1}{p} + \frac{1}{1-p}$.

The following problems are to be turned in. Turn in your solution to *each* problem on a separate sheet of paper, with your name on each one.

1. You are given an approximation scheme $\mathcal{A}$ for f such that $Pr[\frac{f(x)}{1+\epsilon} \le \mathcal{A}(x) \le f(x)(1+\epsilon)] \ge 3/4$, and $\mathcal{A}$ runs in time polynomial in $1/\epsilon, |x|$. Construct an approximation scheme $\mathcal{B}$ for f such that $Pr[\frac{f(x)}{1+\epsilon} \le \mathcal{B}(x) \le f(x)(1+\epsilon)] \ge 1-\delta$, and $\mathcal{B}$ runs in time polynomial in $\frac{1}{\epsilon}, |x|, \log\frac{1}{\delta}$.

   Can your running time be made to be polynomial in $\log\frac{1}{\epsilon}$?

2. You are given $n \times n$ matrices A, B, C whose elements are from $\mathcal{Z}_2$ (integers mod 2). Show a (randomized) algorithm running in $O(n^2)$ time which verifies $A \cdot B = C$. The algorithm should always output "pass" if $A \cdot B = C$ and should output "fail" with probability at least 3/4 if $A \cdot B \ne C$. Assume the field operations $+, \times, -$ can be done in $O(1)$ steps.

3. A 3-SAT formula takes the "and" of a set of clauses, where each clause takes the "or" of a set of literals (each literal is a variable, or the negation of a variable). Show that for any 3-SAT formula in which every clause contains literals corresponding to 3 distinct variables, there is an assignment that satisfies at least 7/8 of the clauses.

4. Denote the complete graph on n nodes by K_n. Let $R(t)$ be the minimal n such that for any two-coloring of the edges of K_n, there is a subset of the vertices of K_n, of size t, such that all edges between vertices in this subset are the same color.

   Show that if $\binom{n}{t}2^{1-\binom{t}{2}} < 1$ then $R(t) > n$. (i.e., show that if $\binom{n}{t}2^{1-\binom{t}{2}} < 1$, then there is a coloring such that there is no subset of vertices of size t such that the edges joining these vertices are all one color).

5. To actually find a perfect matching, we can reuse the algorithm given in class as a subroutine by removing one edge at a time from the graph (together with both endpoints) and testing whether the resulting graph still has a perfect matching. If so, include the edge in the matching, otherwise, remove the edge (but not the endpoints) and try another edge. After at most a number of stages equal to the number of edges in the original graph, we have constructed a matching. However, this method is inherently sequential. We will now try to give a good parallel algorithm. Let's try to use the algorithm from class as a subroutine, but find a way to make all the subroutine calls at the same time.

   - First, let's prove a key lemma: Let $S_1, \ldots, S_k$ be subsets of a set S such that $|S| = m$. For all $i \neq j$, $S_i \neq S_j$. Let each element $x \in S$ have a weight w_x chosen independently and uniformly at random from the set $\{0, 1, \ldots, 2m-1\}$. Then show that

     $$Pr[\exists \text{ a unique set } S_i \text{ of minimum weight}] \geq \frac{1}{2}$$

     A hint is given on the course website.
   - Let B be the matrix formed by replacing x_{ij} with $2^{w_{ij}}$ in the Tutte-Edmonds matrix.[1] Use the lemma above to show that the following algorithm, which can be implemented to run in polylog time on polynomially many processors, outputs a perfect matching:
     - Calculate w, the largest power of 2 that divides $det(B)$
     - For each edge (i, j) in parallel do:
       * compute $t_{ij} = det(B_{ij})\frac{2^{w_{ij}}}{2^w}$ where B_{ij} is the (i, j) minor of B
       * place (i, j) in M iff t_{ij} is an odd integer
     - if M is a perfect matching then output M else fail.

[1]We referred to this as the Tutte matrix in class, and the Edmonds matrix in the notes