CS 787: Advanced Algorithms	**Homework 2**
Out: 10/20/15	**Due: 11/03/15**

Guidelines

- This homework consists of 6 problems. Only the problems marked with an asterisk (two in all) will be graded. You are not required turn in solutions to the unmarked problems but you are highly encouraged to solve and write up your solutions to all of them. We will provide solutions for all the problems.
- Some of the problems are difficult, so please get started early. Late submissions do not get any credit.
- Please typeset your solutions.
- Homework should be done in pairs. Please write your names clearly on your homework.

Problems

1. Let X be a random variable that takes on n values: $x_1, \cdots, x_n$; and let p_i denote the probability that $X = x_i$. Note: $\sum_{i=1}^{i=n} p_i = 1$. Give an algorithm for instantiating X using unbiased coin flips. That is, your algorithm should take as input a string of random bits, and return a value for X such that the output is x_i with probability exactly p_i for each index i. How many coin flips does your algorithm need in expectation?

2. In the **Max-3-SAT** problem, we are given a 3-CNF formula, and our goal is to find an assignment for the variables that satisfies as many clauses as possible.

   (a) Consider a randomized algorithm that returns a uniformly random assignment for the variables. What approximation guarantee does this algorithm achieve?

   (b) Derandomize the above algorithm using the method of conditional expectations.

3. In this question we analyze a contention resolution system in a balls-and-bins framework. We have n resources (bins) and n jobs (balls); we allocate resources to jobs in rounds. In the first round, we throw all the n balls into n bins uniformly at random. After round $i \geq 1$, we remove every ball that occupies a bin by itself in round i, and in the next round, $i + 1$, we throw the remaining balls into the n bins. (One can imagine the lonely balls getting service, whereas none of the colliding ones receive service.) The process continues until no balls are left.

   Prove that this process runs for at most $c \log \log n$ rounds with high probability, where c is some constant. *(Hint: Keep track of the number of balls remaining after every round.)*

4. Recall that in the set cover problem our goal is to cover a given set of elements using a collection of subsets of the least total cost. In the **maximum coverage problem** our goal is complementary—we want to cover the most number of elements using a collection of subsets with total cost no larger than a given bound B.

   (a) Give a constant factor approximation for the maximum coverage problem. Try to obtain as good an approximation as possible.

   (b) Prove that if maximum coverage can be approximated within a factor of $1 - 1/e + \epsilon$ in polynomial time for some constant $\epsilon > 0$, then there exists a constant $\epsilon' > 0$ such that set cover can be approximated within a factor of $(1 - \epsilon') \log n$ in polynomial time. You may assume that you know the cost of the optimal solution for the set cover problem (but not the actual solution).

5. (*) In the **metric Traveling Salesman Problem** (a.k.a. metric TSP) we are given a complete undirected graph with lengths ℓ_e on edges. Lengths satisfy the "metric" property: for every triple of vertices u, v, w, we have:

$$\ell_{(u,w)} \leq \ell_{(u,v)} + \ell_{(v,w)}$$

In words, the direct edge from u to w is shorter than going from u to w via v. The goal is to find the shortest "tour" or cycle that visits every vertex in the graph. Consider the following greedy algorithm. We start at an arbitrary vertex u, and at each step, travel to the closest unvisited vertex; after visiting every vertex, we go back to u to complete the tour.

Show that the greedy algorithm for metric TSP is an $O(\log n)$-approximation, where n is the number of vertices. *(Hint: Argue that the kth least expensive edge in the tour output by the greedy algorithm has weight at most $OPT/(n-k+1)$; try $k = 1$ and $k = 2$ first.)*

6. (*) Recall that a vertex cover of a graph $G = (V, E)$ is a set of vertices $S \subset V$ such that each edge has at least one endpoint in S. We analyzed the following algorithm in class and showed that it obtains a 2-approximation for the Minimum Vertex Cover problem.

   i. Start with $S \leftarrow \emptyset$.
   ii. Pick an edge (u, v) such that $\{u, v\} \cap S = \emptyset$. Add both u and v to S.
   iii. If S is a vertex cover, halt, else go to Step (ii).

   (a) Consider changing step (ii) of the algorithm to the following: "Pick an edge (u, v) such that $\{u, v\} \cap S = \emptyset$. Add an arbitrary endpoint of the edge to S."

   Give an instance on which this algorithm may return a set which is $\Omega(n)$ times larger than the smallest vertex cover.

   (b) Next consider changing step (ii) to the following: "Pick an edge (u, v) such that $\{u, v\} \cap S = \emptyset$. Flip a coin and with probability $1/2$ add u to S, else add v to S."

   Show that this variant achieves a 2-approximation. *(Hint: Consider any vertex v in the optimal vertex cover, and let $N(v)$ be the set of the neighbors of v as well as v itself. What is $E[|S \cap N(v)|]$?)*

   (c) Suppose each vertex v has a weight $w(v)$, and the objective is to pick a set of smallest weight. Give an example to show that the above algorithms do not work for this problem.

   (d) Finally consider changing step (ii) so that upon picking edge (u, v), we add u to S with probability $\frac{w(v)}{w(u)+w(v)}$ and add v to S with probability $\frac{w(u)}{w(u)+w(v)}$. If W is the weight of the least-weight vertex cover, show that this variation obtains an expected weight $\mathrm{E}[\sum_{v \in S} w(v)] \leq 2W$.