15-859(M): Randomized Algorithms — **Homework 5**
Date: Nov 15, 2004 — **Due:** Nov 29, 2004

Ground Rules

Please let us know, for each question, if you have seen the question before. And do prove claims that you make. This assignment should also be done individually: **no collaboration is allowed**. Also, we will be strict with the deadline on this one as well: **due in class on Monday Nov 29th.**

Questions

1. **(Small Inner Products.)** Show that there exist $N = 2^{c_\epsilon \times n}$ unit vectors $\{v_1, v_2, \ldots, v_N\}$ in $\frac{1}{\sqrt{n}}\{-1, 1\}^n$ such that the mutual inner products $|\langle v_i, v_j \rangle| \leq \epsilon$ for all $1 \leq i \neq j \leq N$. (Here c_ϵ is a constant that depends on ϵ, but not on n.)

2. **(Vertex Cover.)** Given a graph $G = (V, E)$, a *vertex cover* of G is a set of vertices $C \subseteq V$ such that each edge has at least one endpoint in C. Finding the vertex cover of the smallest cardinality is NP-complete.

   (a) Consider the following algorithm for Vertex Cover:

      i. Start with $C \leftarrow \emptyset$.

      ii. Pick an edge $\{u, v\}$ such that $\{u, v\} \cap C = \emptyset$. Add an arbitrary endpoint to C.

      iii. If C is a vertex cover, halt, else goto Step (ii).

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

   (b) Now suppose we randomize the algorithm thus: when we pick an edge $\{u, v\}$, we flip an unbiased coin to decide which endpoint to add to C. If k is the size of a smallest vertex cover, show that $E[|C|] \leq 2k$.

   (c) Suppose each vertex v had a weight $w(v)$, and the objective was to pick a set of *smallest weight*. Give an example to show that the above algorithms do not work for this problem. Now alter the algorithm thus: on picking an edge $\{u, v\}$, add u to the cover with probability $\frac{w(v)}{w(u)+w(v)}$. If W is the weight of a least-weight vertex cover, show that $E[w(C)] \leq 2W$.

3. **(Streaming and frequency moments.)** Given a stream of m numbers $a_1, a_2, \ldots, a_m$, with each $a_i \in \{1, 2, \cdots, n\}$, we would like to compute some statistics on this data.

   In particular, let $q_i = |\{j \mid a_j = i\}|$ be the *frequency* of item i, i.e., the number of times the number i appears in the stream. Then the k^{th} *frequency moment* F_k is defined as $F_k = \sum_i q_i^k$. In this question we will construct a randomized algorithm for approximating the second moment F_2 while processing each element only once, and using only $O(\log n \log m)$ bits of space.

   (a) Let $\vec{v} = (v_1, v_2, \ldots, v_n)$ be an n-bit vector with each v_i picked u.a.r. from $\{-1, 1\}$. Consider the random variable $X_v = (\vec{v} \cdot \vec{q})^2$, where $\vec{q}$ is the vector of frequencies. Prove that the expected value of X_v is equal to F_2.

   (b) Determine the variance of X_v.

   (c) Give an FPRAS for F_2 based on the above two parts. (Don't worry about space issues yet.)

   (d) Briefly (one or two lines) describe how to compute the random variable X_v, *given the vector* v, while using a workspace of only $O(\log m \log n)$ bits and a single pass over the stream.

(e) (Extra Credit) A naïve implementation of the algorithm requires us to store n bits of space for the vector v. How can one implement the algorithm for part (c) using storage only $O(\log n)$ bits per vector?

4. **(Random walks on spanning trees.)** Given a connected graph $G = (V, E)$ with $|V| = n$, our goal is to pick a random spanning tree of G. To do this, we construct a *directed* random walk on the space of all spanning trees.

First note the following property of random walks on any directed graph.

(a) Given a strongly-connected directed graph $H = (U, E')$ with the in-degree of every vertex equal to its out-degree, define the *degree of vertex* u as $d(u) = \text{in-degree}(u) = \text{out-degree}(u)$. Prove that a stationary distribution of a random walk on such a graph is given by $\pi^*(u) = d(u)/|E'|$.

Next we study random walks on *rooted* spanning trees. A rooted spanning tree is tuple (T, r), where T is a spanning tree of G and r is the *root* of T. Given the root, the *parent* $\mathsf{parent}(v)$ of any vertex $v \neq r$ is the *second* vertex on the unique path from v to r (the first vertex being v itself).

Consider the following Markov chain $\mathcal{M}$ on the *rooted spanning trees* of G. Starting from a rooted tree (T, r), pick a random neighbor of the root r in G (u.a.r.), say v. With probability $1/2$, stay at (T, r). Otherwise move to (T', v), where T' is the spanning tree obtained by removing the edge $(v, \mathsf{parent}(v))$ and adding the edge (r, v).

(b) Prove that $\mathcal{M}$ is ergodic (irreducible and aperiodic). Give an upper bound on its diameter.

(c) What is the stationary distribution π^* of $\mathcal{M}$?

(d) Suppose we sample from the stationary distribution π^* of $\mathcal{M}$: if we get (T, r), we just output the spanning tree T. What is the resulting probability distribution on *unrooted* spanning trees of G?

Finally we will use a coupling argument to prove that the above Markov chain mixes fast.

(e) Consider the following coupling (X, Y) for the chain $\mathcal{M}$. Let $X = (T_X, r_X)$ and $Y = (T_Y, r_Y)$.

- If the roots of X and Y are different (i.e., $r_X \neq r_Y$), then pick the next state for X and Y independently.
- If $r_X = r_Y = r$, then pick a neighbor of r u.a.r. and use this to obtain the next state in both X and Y.

Using this coupling, prove that the chain $\mathcal{M}$ mixes in time $\tau_{\mathcal{M}}(\varepsilon) \leq \mathcal{C}(G) O(\log 1/\varepsilon) + M_{\varepsilon/2}(G)$, where $\mathcal{C}(G)$ is the cover time of the natural random walk with self loops on the graph G, and $M_\varepsilon(G)$ is the ε-*meeting time* of G, defined as follows. For nodes $x, y \in V$, consider two independent natural random walks on G starting at x and y: t_{xy} is the least time such that

$$\Pr[\text{the two walks occupy the same node in } V \text{ at some time } t' \leq t_{xy}] \geq 1 - \varepsilon.$$

The meeting time of G is defined to be $M_\varepsilon(G) = \max_{x,y \in V} t_{xy}$.

(f) (Nothing to do here.) Note that we have related the mixing time $\tau_{\mathcal{M}}(\varepsilon)$ to two parameters that depend *only on the underlying graph* G. A theorem of Aldous shows that $M_\varepsilon(G) \leq 2\mathcal{C}(G) \log \frac{1}{\varepsilon}$, and hence $\tau_{\mathcal{M}}(\varepsilon) \leq O(\mathcal{C}(G) \log \frac{1}{\varepsilon})$. Of course, $\mathcal{C}(G) = O(n^3)$, and thus we have shown that $\mathcal{M}$ is rapidly mixing.