CS 761 Randomized Algorithms **Last Updated:** June 1, 2018
Homework 2 **Due Date: June 15 (11pm), 2018**

You are allowed to discuss with others but not allowed to use any references except the course notes and the books "Probability and Computing" and "Randomized Algorithms". Please list your collaborators for each question. This will not affect your marks. In any case, you must write your own solutions.

There are totally 55 marks, and the full mark is 50.

---

1. **k-wise independence**

   (10 marks) Suppose that we are given m vectors $v_1, \ldots, v_m \in \{0,1\}^l$ such that any k of the vectors are linearly independent modulo 2. Let $v_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,l})$ for $1 \leq i \leq m$. Let u be chosen uniformly at random from $\{0,1\}^l$. Let $X_i = (\sum_{j=1}^{l} v_{i,j} u_j) \mod 2$. Show that the X_i are uniform, k-wise independent random bits.

2. **Maximum load**

   (10 marks) We have shown that the maximum load when n items are hashed into n bins using a hash function chosen from a 2-universal family of hash functions is at most $\sqrt{2n}$ with probability at least $1/2$. Generalize this argument to k-universal hash functions. That is, find a value such that the probability that the maximum load is larger than that value is at most $1/2$. Then, find the smallest value of k such that the maximum load is at most $3 \ln n / \ln \ln n$ with probability at least $1/2$ when choosing a random hash function from a k-universal family.

3. **Finding approximate median in sublinear time**

   (10 marks) We would like to find an approximate median of n distinct numbers in sublinear time. To do so, we sample $m \ll n$ numbers, find the median c of these m numbers, and report c as the approximate median of the n numbers. Let the sorted list of the n numbers be $\{x_1, x_2, \ldots, x_n\}$ and so the true median is $x_{n/2}$. The approximate median is said to be a $\pm k$-approximation if $c \in [x_{\frac{n}{2}-k}, x_{\frac{n}{2}+k}]$. Suppose we want the algorithm to succeed to find a $\pm k$-approximation with probability at least 0.9999. What is the tradeoff between m and k? How large should we set m if we want $k \leq \epsilon n$ for some small constant ϵ? How large should we set m if we want $k \leq n^{1-\eta}$ for $\eta \leq 1/2$?

4. **Algebraic matching**

   (10 marks) In the following two sub-problems, you can assume that the determinant of a matrix where each entry is an element in $\mathbb{F}[x]$ of degree at most d can be computed in $O(dn^{\omega})$ field operations, where $\mathbb{F}[x]$ is the set of single variate polynomials with coefficients in a finite field $\mathbb{F}$ and $\omega \approx 2.37$ is the matrix multiplication exponent. Note that the determinant of such a matrix would be a single variate polynomial.

   (a) Given a bipartite graph where each edge is red or blue and a parameter k, determine if there is a perfect matching with exactly k red edges in $O(n^{\omega})$ field operations with high probability.

(b) Given a bipartite graph with a non-negative weight on each edge, determine the weight of a maximum weighted perfect matching in $O(Wn^{\omega})$ field operations with high probability, where W is the maximum weight of an edge.

5. **Network coding**

   (15 marks) Suppose $G = (V, E)$ is a directed acyclic graph and $s \in V$ is the only vertex with indegree zero. In this problem, we would like to design a fast (and distributed) algorithm to compute the edge connectivity from s to v for every $v \in V - s$ (i.e. the number of edge-disjoint directed paths from s to v).

   Consider the following "network coding" algorithm. Let $e_1, e_2, \ldots, e_d$ be the d out-going edges of s. Choose a finite field $\mathbb{F}$. Initially, we assign a d-dimensional unit vector $\vec{e_i}$ to each edge e_i, where $\vec{e_i}$ is the standard unit vector with one in the i-th position and zero otherwise. Then, we follow the topological ordering to process the vertices. When we process a vertex x, there is already a d-dimensional vector (where each entry is an element in $\mathbb{F}$) for each of its incoming edge. Now, for each outgoing edge of x, we compute a d-dimensional vector for it by taking a random linear combination of the incoming vectors in x (i.e. random coefficients from $\mathbb{F}$ and arithmetic over $\mathbb{F}$). We repeat this process until every edge in the graph has a d-dimensional vector. Finally, for each vertex v, we compute the rank of its incoming vectors, and return this value as the edge connectivity from s to v.

   Prove that this algorithm outputs the correct answers for all vertices $v \in V - s$ with high probability when $|\mathbb{F}| = \Theta(\text{poly}(|V|))$. Give a fast implementation and an upper bound on the total running time to compute the edge connectivity from s to all vertices $v \in V - s$.