CS 466/666 Algorithm Design and Analysis **Last Updated:** June 24, 2019
Homework 3 **Due Date: July 5 (11pm), 2019**

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 60 marks. The full mark is 50. This homework is counted 8% of the course.

---

1. **Algebraic matching**

   (20 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.

2. **Network coding**

   (20 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 computed (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 prove an upper bound on the total running time to compute the edge connectivity from s to all vertices $v \in V - s$. You can assume that the rank of a $d \times k$ matrix for $k \leq d$ can be computed in $O(dk^{\omega-1})$ field operations where $\omega \approx 2.37$ is the matrix multiplication exponent.

3. **Matrix Rank**

(20 marks) Given a matrix $A \in \mathbb{F}^{m \times n}$ where $m \leq n$ and $\mathbb{F}$ is large enough finite field, we would like to compute the rank of A. Our approach is to "compress" the matrix A into an $m \times m$ matrix B with the same rank efficiently (i.e. a fast reduction to the square case).

(a) Consider the following compression algorithm. Each column of B is a random linear combination of all the columns of A, i.e. $B_i = r_1^{(i)} A_1 + \cdots + r_n^{(i)} A_n$ for $1 \leq i \leq m$ where A_i, B_i denote the i-th column of A, B and $r_j^{(i)}$ for $1 \leq i \leq m$ and $1 \leq j \leq n$ is an independent uniform random element in $\mathbb{F}$. Prove that $\text{rank}(B) = \text{rank}(A)$ with probability at least $1 - 1/m$ if $|\mathbb{F}| \geq m^2$. State the time complexity of this compression algorithm.

(b) Design a faster randomized compression algorithm. Describe the algorithm, prove its correctness (with high probability), and bound its time complexity. You can assume that the field size $|\mathbb{F}|$ is large enough, say $|\mathbb{F}| \geq n^4$.

(Hint: Use some "efficient" objects in the notes.)