Algorithmic Combinatorics
Shagnik Das

Wi 2016–17
Tibor Szabó

Exercise Sheet 13

Due date: 14:00, Feb 7th, by the end of the lecture.
Late submissions will be forgotten, but not forgiven.

You should try to solve all of the exercises below, and submit two solutions to be graded — each problem is worth 10 points. We encourage you to submit in pairs, but please remember to indicate the author of each individual solution.

Exercise 1 You use a matrix multiplication service of questionable repute to compute the product of two $n \times n$ matrices, A and B, and receive a matrix C, together with a hefty bill. To check that the answer is correct, you run the randomised verification algorithm, multiplying both C and AB by a random vector $\vec{x} \in \{0,1\}^n$.

(a) How many times do you have to run the algorithm to have at least 95% confidence in the outcome?

Suppose you run the algorithm, and find that $C\vec{x} \neq AB\vec{x}$. This proves the *existence* of a mistake. However, in order to get your money back, you need to explicitly *find* a mistake; that is, find some i and j such that $C_{ij} \neq (AB)_{ij}$.

(b) How many more arithmetic operations will this take?

[Hint at `http://discretemath.imp.fu-berlin.de/DMII-2016-17/hints/S13.html`.]

Exercise 2 Let $k \geq 1$ be some integer, and let A and B be two $2^k \times 2^k$ matrices. We wish to efficiently compute $C = AB$. We express these in terms of $2^{k-1} \times 2^{k-1}$ submatrices:

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}, B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}, \text{ and } C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}.$$

We now define some new matrices:

$$M_1 = (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2}), \quad M_2 = (A_{2,1} + A_{2,2})B_{1,1}, \quad M_3 = A_{1,1}(B_{1,2} - B_{2,2}),$$
$$M_4 = A_{2,2}(B_{2,1} - B_{1,1}), \quad M_5 = (A_{1,1} + A_{1,2})B_{2,2}, \quad M_6 = (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2}),$$
$$\text{and} \quad M_7 = (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2}).$$

(a) Verify the identities $C_{1,1} = M_1 + M_4 - M_5 + M_7$, $C_{1,2} = M_3 + M_5$, $C_{2,1} = M_2 + M_4$, and $C_{2,2} = M_1 - M_2 + M_3 + M_6$.

(b) One can reuse these identities to calculate the products in the definition of the matrices M_i, leading to a recursive algorithm for computing the product $C = AB$. Estimate the running time (in terms of the number of arithmetic operations) of this algorithm.

(c) For general integers $n \geq 1$, how can this algorithm be applied to $n \times n$ matrices?

Exercise 3 Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a non-zero polynomial, and let d_i be the degree of f in the variable x_i; that is, d_i is the maximum power of x_i appearing in f. Let $S \subseteq \mathbb{F}$ be a finite set with $|S| \geq \max_i d_i$.

(a) Prove that f can have at most $|S|^n - \prod_{i=1}^{n} (|S| - d_i)$ zeroes in S^n.

(b) For any given values $d_1, \ldots, d_n \in \mathbb{N} \cup \{0\}$, and a set $S \subseteq \mathbb{F}$ with $|S| \geq \max_i d_i$, construct a polynomial whose degree in x_i is at most d_i, $1 \leq i \leq n$, for which the bound in (a) is tight.

[Hint at `http://discretemath.imp.fu-berlin.de/DMII-2016-17/hints/S13.html`.]

Exercise 4 In lecture we saw a randomised algorithm for determining if there is a perfect matching in a bipartite graph with n vertices in each part.

(a) Using this algorithm, explain how one can find a perfect matching, if it exists, in such a graph.

(b) If it takes $O(n^\omega)$ operations to find the determinant of an $n \times n$ matrix, how many operations does your matching-finding algorithm require?

Exercise 5 Suppose we have some univariate polynomial $p \in \mathbb{F}[x]$ of degree at most d that we only have oracle access to, so that for any input $y \in \mathbb{F}$, we are told the value $p(y)$.

(a) Explain how we can determine the coefficient of x^k with at most $d + 1$ oracle queries.

Suppose now we have a bipartite graph $G = (U \cup V, E)$ with $|U| = |V| = n$, and suppose further that the edges E are coloured red or blue.

(b) Using part (a), explain how we can extend the randomised perfect matching testing algorithm to test whether or not there is a perfect matching of G with exactly k red edges.

Exercise 6 This exercise will show you how to extend our randomised perfect matching algorithm for bipartite graphs to general graphs. Let $G = ([n], E)$ be a graph on n vertices. As before, we introduce a new variable x_{ij} for each edge $\{i, j\} \in E$. The *Tutte matrix* A of the graph G is defined as $A = (a_{ij})_{i,j\in[n]}$, where

$$a_{ij} = \begin{cases} +x_{ij} & \text{if } i < j \text{ and } \{i, j\} \in E, \\ -x_{ji} & \text{if } i > j \text{ and } \{i, j\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

(a) Find both the Tutte matrix A and its determinant $\det(A)$ when $G = K_3$ and $G = C_4$.

(b) Show $\det(A)$ is not the zero polynomial if G has a perfect matching.

We can think of a permutation $\pi \in S_n$ in terms of its cycle structure[1], which allows us to define $\operatorname{sgn}(\pi) = (-1)^{\#\text{ even cycles in }\pi}$. We then have $\det(A) = \sum_{\pi\in S_n} \operatorname{sgn}(\pi) \prod_{i\in[n]} a_{i\pi(i)}$.

(c) If we think of isolated edges as cycles of length two, show that any nonzero monomial in this expansion of $\det(A)$ corresponds to a partition of $[n]$ into vertex-disjoint cycles in G.

(d) By reversing the direction of an odd cycle, show that if $\det(A)$ is not the zero polynomial, then there is some partition of $[n]$ into vertex-disjoint cycles in G, all of which have even length.

(e) Deduce that $\det(A)$ is not the zero polynomial if and only if G has a perfect matching, and give a randomised algorithm for testing for the existence of a perfect matching in G.

[1]For example, if $n = 6$ and $\pi(1) = 2$, $\pi(2) = 5$, $\pi(3) = 4$, $\pi(4) = 3$, $\pi(5) = 1$ and $\pi(6) = 6$, then π has cycle structure (1 2 5) (3 4) (6).