# CIS 667 and CIS 467H: Homework 2(Due: Feb 24, 2005)

**Problem 1.** (20 points)

a. What is the largest $k$ such that if you can multiply $3 \times 3$ matrices using $k$ multiplications (not assuming commutativity of multiplication), then you can multiply $n \times n$ matrices in time $o(n^{\lg 7})$? What would the running time of this algorithm be?

b. V. Pan has discovered a way of multiplying $68 \times 68$ matrices using 132464 multiplications, $70 \times 70$ matrices using 143640 multiplications, $72 \times 72$ matrices using 155424 multiplications. Which method yields the best asymtotic running time when used in a divide and conquer matrix multiplication algorithm? How does it compare to Strassen's algorithm?

**Problem 2.** (20 points)

You are given six polynomials $f_1, \ldots, f_6$ of degrees $1, 3, 2, 3, 4, 5$ respectively. We are interested in finding the product $f = f_1 f_2 f_3 f_4 f_5 f_6$ by performing pairwise multiplications. Assume that the cost of multiplying two polynomials of degree $a$ and $b$ is $a+b$, i.e. it is proportional to the space required to store the product which is a polynomial of degree $a+b$. Find a schedule for multiplying the six polynomials that is of the lowest possible total cost (total cost is the sum of the costs of all multiplications performed to determine $f$) for this non-traditional definition of a cost function.

Example. For three polynomials $g_1, g_2, g_3$ of degrees $1, 2, 3$ respectively, you first compute $g_2 g_3$ and then multiply the result by $g_1$. The cost of the first multiplication is $5$ $(2+3)$ and the cost of the second multiplication is 6 since you multiply the result, a degree 5 polynomial, to a degree one polynomial. Total cost is 11.

**Problem 3.** (20 points)

a. Let $M(n)$ be the time to multiply two $n \times n$ matrices and let $S(n)$ be the time to square an $n \times n$ matrix. Show that multiplying and squaring have essentially the same complexity: i.e. an $M(n)$ matrix multiplication algorithm implies an $O(M(n))$ squaring algorithm and and an $S(n)$ squaring algorithms implies an $O(S(n))$ matrix multiplication algorithm.

b. Show that interpolation can be done in $O(n^2)$ time. Hint: Read Exercise 30.1-5 of CLRS on page 830 (or Exercise 32.1-4 of CLR on page 783), and think of the implications of Problem 2, HW1.

**Problem 4.** (20 points)

Go to page 844 of CLRS. Problem 30-1 was partly solved in class; part (a) in the context of complex numbers, and part (c) through the Karatsuba-Ofman algorithm. Do part (b).

**Problem 5.** (20 points)

Suppose that we insert $n$ keys into a hash table of size $m$ using open addressing and uniform hashing. Let $p(n,m)$ be the probability that no collisions occur. Show that $p(n,m) \leq exp(-n(n-1)/(2m))$. Argue that when $n$ exceeds $\sqrt{m}$, the probability of avoiding collisions goes rapidly to zero.

**Hint:** Use $exp(x) \geq 1 + x$ for any real $x$. Note that $exp(x) = e^x$.

**Option 1.** (100 points)

(a) Implement Karatsuba-Ofman for polynomials of degree bound $n$, or arbitrarily long $n$-bit integers. Collect timing information compared to the ordinary algorithm for $n = 32, 128, 512, 1024$. (50 points).

(b) Implement Strassen's algorithm for $n \times n$ matrices (i) where $n$ is a power of 2, (ii) $n$ is not a power of two. Run experimental results for $n = 256$, $n = 512$ and $n = 1024$ and collect timing information. (50 points)

All programs to be sent to `alg667@cs.njit.edu` or `alg467@cs.njit.edu`.

■