Solutions to Math 416 Homework due December 10, Chapter 30

December 10, 2004

Problem 30.2-8

We are given $a = (a_0, \ldots, a_{n-1})$ and a complex number z; we want $y = (y_0, \ldots, y_{n-1})$, where $y_k = \sum_{j=0}^{n-1} a_j z^{kj}$.

Following the hint, define a vector f by $f_j = a_j z^{j^2/2}$ and a vector g by $g_j = z^{-j^2/2}$. (Here, use either square root of z, but be consistent.) Then, from the definition of convolution,

$$\begin{aligned}
(f \otimes g)(k) &= \sum_j f_j g_{k-j} \\
&= \sum_j (a_j z^{j^2/2})(z^{-(k-j)^2/2}) \\
&= \sum_j (a_j z^{j^2/2})(z^{(-k^2+2jk-j^2)/2}) \\
&= \sum_j a_j (z^{(-k^2+2jk)/2}) \\
&= z^{-k^2/2} \sum_j a_j z^{jk},
\end{aligned}$$

so $y_k = z^{k^2/2}(f \otimes g)(k)$ is the chirp transform.

We can compute f and g from a and z in time $O(n\log(n))$ as follows. First, compute a square root $w = z^{1/2}$ of z. Next, compute $w^1, w^2, w^4, w^8, \ldots, w^n$ (*i.e.*, w^{2^ℓ}, by repeated squaring, in total time $O(\log(n))$). Finally, for each j, compute w^{j^2} by multiplying together $O(\log(n))$ appropriate powers of w, according to the binary expansion of j. We can then compute $a_j w^{j^2}$ and $w^{-j^2} = 1/w^{j^2}$ in constant time each.

Alternatively, to compute w^{j^2} for all these j's, compute $1, w, w^2, w^3, \ldots$, computing each w^j from w^{j-1} and w in constant time. Then compute $1, w^1, w^4, w^9, \ldots$, by computing $w^{j^2} = w^{(j-1)^2+2j-1} = w^{(j-1)^2} \cdot w^j \cdot w^{j-1}$ in constant time from $w^{(j-1)^2}$, w^j, and w^{j-1}. This takes time $O(n)$ instead of time $O(n\log(n))$.

Next, compute the convolution of f and g in time $O(n\log(n))$, using the FFT algorithm. Finally, multiply w^{k^2} by $(f \otimes g)(k)$ in constant time for each of n possible k's, for a total of time $O(n)$.

ADDITIONAL COMMENTARY: Note that if $|z|$ is bigger than around $1 + 1/n^2$, then $z^{j^2/2}$ is going to grow out of control. Depending on the a's, the result may be dominated by the largest few terms, so time much less than n suffices to get a good floating point represenation. If one really wants an exact representation (assuming z has terminating real and imaginary decimal expansions), then one needs precision around n^2 bits to store a number like 2^{n^2}. The resulting algorithm will take time at least n^3 in any reasonable model of computation and the output itself will be of size around n^2 bits. A similar statement holds if $|z|$ is less than around $1 - 1/n^2$. These problems go away if $|z| = 1$.

It follows that the DFT, for any n, can be reduced to a convolution. The main result of this section is that any convolution can be reduced to a DFT for n a power of 2. Also, convolution of length n can be reduced to convolution of length $n' > n$ by padding with zeros. It follows that the DFT for any n can be done in time $O(n\log(n))$, by reducing to the DFT of the next larger power of 2.

Problem 30-2

(a) The sum of two Toeplitz matrices is Toeplitz. (See part (b).) The product is not necessarily Toeplitz. For example,

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}.$$

(b) For convenience, define $b_i = a_{i,0} = a_{i+1,1}$ for $i = 0, 1, \ldots, n-1$. Also define $a_{i,0} = a_{1,1-i}$ for $i = -1, -2, \ldots, -n+1$. Thus we have

$$\left(\begin{array}{cccc|ccc} a_{-3,0} & a_{-2,0} & a_{-1,0} & a_{0,0} & a_{1,0} & a_{2,0} & a_{3,0} \\ \hline & & & & a_{1,1} & a_{2,1} & a_{3,1} \\ & & & & a_{1,2} & a_{1,1} & a_{2,1} \\ & & & & a_{1,3} & a_{1,2} & a_{1,1} \\ & & & & a_{1,4} & a_{1,3} & a_{1,2} \end{array}\right),$$

where the top row (which is not part of the matrix) is also $(a_{1,4}, a_{1,3}, a_{1,2}, a_{1,1} = a_{0,0}, a_{2,1}, a_{3,1}, a_{4,1})$. Observe that $a_{ij} = a_{k\ell}$ if $j - i = \ell - k$; this formula extends also to the top (0'th) row. It follows that $a_{j,k} = b_{k-j}$.

It then follows that one can represent a Toeplitz matrix by just the top row of b's ($2n-1$ numbers). The sum of two Toeplitz matrices is represented by the sum of the two corresponding top rows, which can be computed in time $O(n)$. (Since any top row leads to a Toeplitz matrix, it follows, in part (a), that the sum of two Toeplitz matrices is Toeplitz.)

(c) To multiply by a vector, it is convenient to index the vector backwards: $v = (v_{n-1}, v_{n-2}, \ldots, v_0)$. Also, we will multiply separately by the upper and lower triangle:

$$\begin{pmatrix} b_0 & b_1 & b_2 & b_3 \\ 0 & b_0 & b_1 & b_2 \\ 0 & 0 & b_0 & b_1 \\ 0 & 0 & 0 & b_0 \end{pmatrix} \cdot \begin{pmatrix} v_{n-1} \\ v_{n-2} \\ \vdots \\ v_1 \\ v_0 \end{pmatrix}$$

The resulting vector product is n of the $2n-1$ terms in the convolution of the $b_{i\geq 0}$ sequence and the v sequence. We can do this in time $O(n\log(n))$. Similarly, we can multiply the lower triangle by v by convolving the $b_{i<0}$ sequence with the v sequence. We then add the two vectors of length n.

(d) We can multiply a Toeplitz matrix by an arbitrary matrix M in time $n^2\log(n)$ by multiplying by each column of M separately. Some speedups are possible if M is also Toeplitz, but note that we need to output n^2 numbers.

ADDITIONAL COMMENTARY: Is somewhat more natural and elegant to define the convolution as $(f \otimes g)(k) = \sum_{0\leq j<n} f_j g_{k-j}$, where k and $k-j$ are taken modulo n. Thus there are only n elements in $(f \otimes g)$, not $2n-1$. This corresponds to multiplying polynomials modulo $x^n - 1$, rather than multiplying polynomials without modular reduction. This may be regarded as an alternative to padding with zeros. We then have the formula that the Fourier transform of $f \otimes g$ is the pointwise product of the Fourier transform of f and the Fourier transform of g.

This corresponds to the *circulant* variation of Toeplitz matrices, of the form

$$\begin{pmatrix} a & b & c & d \\ d & a & b & c \\ c & d & a & b \\ b & c & d & a \end{pmatrix}.$$

Note that a circulant matrix is a special kind of Toeplitz matrix. The sum of two circulant matrices is circulant and the product of two circulants is circulant, which can be checked easily. One can represent a circulant matrix by its top row (which is now part of the matrix). To multiply two circulants, take the Fourier transforms of each top row, multiply those together, then form a circulant from the result. This takes time $O(n\log(n))$.