Algorithms – handout 9

Fast Fourier Transform

January 6, 2000

There will be no lecture notes this week. Next week a lecture is planned as usual, though check your e-mail for messages to see if its cancelled due to miluim

A degree $n-1$ polynomial can be represented by its n coefficients, or by its values in n points. The latter representation is a consequence of the fact that a degree n polynomial has at most n roots. Evaluation is easier with coefficient representation (in particular, using Horner's rule), whereas polynomial multiplication is easier using the point-value representation.

The discrete Fourier transform DFT is the evaluation of the polynomial at the n nth roots of unity (e.g., in the complex plane). The inverse DFT is the interpolation of the polynomial from its point-values at the n roots of unity.

The FFT algorithm performs DFT in $O(n \log n)$ algebraic operations. It uses symmetries associated with the n roots of unity. Let n be a power of 2, ω_n be a principle nth root of unity and ω_n^i for $0 \le i \le n-1$ be the nth roots of unity. Then we have that $(\omega_n^i)^j$ is an nth root of unity for every j, $(\omega_n^i)^2$ is an $(n/2)$th root of unity, $\omega_n^{n/2} = -1$, and $\sum_{j=0}^{n-1} (\omega_n^i)^j = 0$ for i not divisible by n.

FFT can be used in order to do polynomial multiplication in $O(n \log n)$ algebraic operations.

1. Pad with sufficiently many 0 coefficients.
2. Use FFT to go from coefficient representation to point-value representation.
3. Multiply pointwise.
4. Apply FFT to compute the inverse Fourier transform.

FFT can be used in order to speed up various other computations. For example, a polynomial can be interpolated from its roots in time $O(n(\log n))^2$.

FFT is among the nontrivial algorithms that are most widely used in practice, because of the importance of the Fourier transform in various areas of engineering.

Homework:

For polynomials with integer coefficients, it is possible to avoid working with complex numbers. If the coefficients of the two input polynomials are not too large, and one can argue that the coefficients of the product polynomial are at most $n/2$ bits long, then one can perform all operations modulo $2^{n/2}+1$ (where n itself is a power of 2). Show that in this case one can use 2 as a principle nth root of unity when performing DFT and inverse DFT using the FFT algorithm.