

CSE 421

Algorithms

Lecture 16

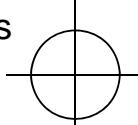
Fast Fourier Transform

Slides from 2006. A different slide deck will be used for the 2009 lecture

FFT, Convolution and Polynomial Multiplication

- FFT: $O(n \log n)$ algorithm
 - Evaluate a polynomial of degree n at n points in $O(n \log n)$ time
- Polynomial Multiplication: $O(n \log n)$ time

Complex Analysis



- Polar coordinates: $r e^{\theta i}$
- $e^{\theta i} = \cos \theta + i \sin \theta$
- a is an n^{th} root of unity if $a^n = 1$
- Square roots of unity: $+1, -1$
- Fourth roots of unity: $+1, -1, i, -i$
 - Eighth roots of unity: $+1, -1, i, -i, \beta + i\beta, \beta - i\beta, -\beta + i\beta, -\beta - i\beta$ where $\beta = \sqrt{2}$

$$e^{2\pi ki/n}$$

- $e^{2\pi i} = 1$
- $e^{\pi i} = -1$
- n^{th} roots of unity: $e^{2\pi ki/n}$ for $k = 0 \dots n-1$
- Notation: $\omega_{k,n} = e^{2\pi ki/n}$
- Interesting fact:

$$1 + \omega_{k,n} + \omega_{k,n}^2 + \omega_{k,n}^3 + \dots + \omega_{k,n}^{n-1} = 0$$
 for $k \neq 0$

FFT Overview

- Polynomial interpolation
 - Given $n+1$ points (x_i, y_i) , there is a unique polynomial P of degree at most n which satisfies $P(x_i) = y_i$

Polynomial Multiplication

$n-1$ degree polynomials

$$A(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}, \\ B(x) = b_0 + b_1 x + b_2 x^2 + \dots + b_{n-1} x^{n-1}$$

p_1, p_2, \dots, p_{2n}

$$A(p_1), A(p_2), \dots, A(p_{2n}) \\ B(p_1), B(p_2), \dots, B(p_{2n})$$

$$C(x) = A(x)B(x)$$

$$C(x) = c_0 + c_1 x + c_2 x^2 + \dots + c_{2n-2} x^{2n-2}$$

$$C(p_1) = A(p_1)B(p_1)$$

FFT

- Polynomial $A(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1}$
- Compute $A(\omega_{j,n})$ for $j = 0, \dots, n-1$
- For simplicity, n is a power of 2

Useful trick

$$A(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_{n-1}x^{n-1}$$

$$A_{\text{even}}(x) = a_0 + a_2x + a_4x^2 + \dots + a_{n-2}x^{(n-2)/2}$$

$$A_{\text{odd}}(x) = a_1 + a_3x + a_5x^2 + \dots + a_{n-1}x^{(n-2)/2}$$

$$\text{Show: } A(x) = A_{\text{even}}(x^2) + x A_{\text{odd}}(x^2)$$

Lemma: $\omega_{j,2n}^2 = \omega_{j,n}$

Squares of $2n^{\text{th}}$ roots of unity are n^{th} roots of unity

FFT Algorithm

```
// Evaluate the  $2n-1^{\text{th}}$  degree polynomial A at
//  $\omega_{0,2n}, \omega_{1,2n}, \omega_{2,2n}, \dots, \omega_{2n-1,2n}$ 
FFT(A, 2n)
```

 Recursively compute FFT(A_{even}, n)

 Recursively compute FFT(A_{odd}, n)

 for $j = 0$ to $2n-1$

$$A(\omega_{j,2n}) = A_{\text{even}}(\omega_{j,2n}^2) + \omega_{j,2n} A_{\text{odd}}(\omega_{j,2n}^2)$$

Polynomial Multiplication

- $n-1^{\text{th}}$ degree polynomials A and B
- Evaluate A and B at $\omega_{0,2n}, \omega_{1,2n}, \dots, \omega_{2n-1,2n}$
- Compute C($\omega_{j,2n}$) for $j = 0$ to $2n - 1$
- We know the value of a $2n-2^{\text{th}}$ degree polynomial at $2n$ points – this determines a unique polynomial, we just need to determine the coefficients

Now the magic happens . . .

$$C(x) = c_0 + c_1x + c_2x^2 + \dots + c_{2n-1}x^{2n-1}$$

(we want to compute the c_i 's)

Let $d_j = C(\omega_{j,2n})$

$$D(x) = d_0 + d_1x + d_2x^2 + \dots + d_{2n-1}x^{2n-1}$$

Evaluate D(x) at the $2n^{\text{th}}$ roots of unity

$$D(\omega_{j,2n}) = [\text{see text for details}] = 2nc_{2n-j}$$

Polynomial Interpolation

- Build polynomial from the values of C at the $2n^{\text{th}}$ roots of unity
- Evaluate this polynomial at the $2n^{\text{th}}$ roots of unity