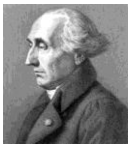
Algorithm Design and Analysis

Victor Adamchik — CS 15-451 Spring 2015

Lecture 3 — Jan 21, 2015 — Carnegie Mellon University

Fast Fourier Transform



Gauss (1777 – 1855) — Lagrange (1736 –1813) — Fourier (1768 –1830)

High Level Idea

To compute the product A(x)B(x) of polynomials

1) evaluate A(x) and B(x) at roots of unity, using the Vandermonde matrix — O(n log n)
2) multiply $A(x_k)B(x_k)$, — O(n)
3) then find the polynomial using Lagrange's interpolation via the Vandermonde matrix — O(n log n)

Computing Polynomials

Given a polynomial of degree n.

$$A(x) = \sum_{k=0}^{n} a_k x^k$$

What is the complexity of computing its value at a single point, $A(x_0)$?

Horner's Rule: **O(n)**

$$A(x) = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + a_n x)\ldots)$$

Computing Polynomials

So we compute the single value in linear time.

Therefore, it takes $O(n^2)$ to compute a polynomial of degree n at n points.

In the next slides we will develop a new method that gives O(n log n) runtime complexity.

Computing Polynomials

The key idea is to use the divide-and-conquer algorithm. We split a polynomial into two parts: with even and odd degree terms.

$A(x) = A_0(x^2) + x\ A_1(x^2)$

For example,

$1+2x+3x^2+4x^3+5x^4+6x^5=(1+3x^2+5x^4)+x(2+4x^2+6x^4)$

$A_0(x) = 1+3x+5x^2$ $A_1(x) = 2+4x+6x^2$

Observe, computing A(-x) takes O(1). Thus, our special points are half pos and half neg.

Worst-time Complexity

Let T(n) be the complexity of computing a degree-n polynomial at 2n points. Thus

$$T(n) = 2\,T(n/2) + O(n)$$

This solves to O(n log n).

The only problem is that the algorithm requires of having half positive and half negative points on each iteration.

Very special points

$A(x) = A_0(x^2) + x\ A_1(x^2)$

So, we need to find such a set of points that

1) half of points are negative and the second half is positive

2) this property holds after squaring (on each iteration)

Roots of Unity

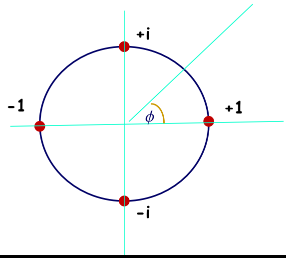
They are defined as solutions to $z^n = 1$.

The n-th roots of unity are points on the complex unit circle every $2\pi/n$ radians apart

$$z = e^{i\ 2\pi\ k/n}$$

k=0,1,…,n-1

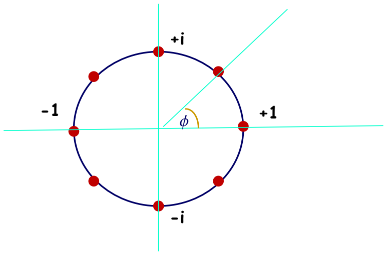
The angle $\phi=2\ \pi/n$



Roots of Unity

They are defined as solutions to $z^n = 1$.

Here is n = 8



Complex numbers on a unit circle are represented by $z = e^{i\phi} = \cos(\phi) + i \sin(\phi)$

Roots of Unity: n = 8

Let $w = \sqrt{i}$, then roots of $z^8 = 1$ can be written as

$$1, w, w^2, w^3, w^4, w^5, w^6, w^7$$

Since $i^2 = -1$, and thus $w^4 = -1$, they can also be written as

$$1, w, w^2, w^3, -1, -w, -w^2, -w^3$$

Let us take a half and square them

$$(1, w, w^2, w^3)^2 = (1, w^2, w^4, w^6) = (1, w^2, -1, -w^2)$$

Do it again

$$(1, w^2)^2 = (1, w^4) = (1, -1)$$

Computing Polynomials

Given a polynomial

$$A(x) = \sum_{k=0}^{n-1} a_k x^k$$

Our task to compute a polynomial <u>at n points</u>:

$$A(1), A(w), A(w^2), \ldots, A(w^{n-1})$$

where $w^n = 1$.

We can write these computations in a matrix form! And thus compute all of then at once!

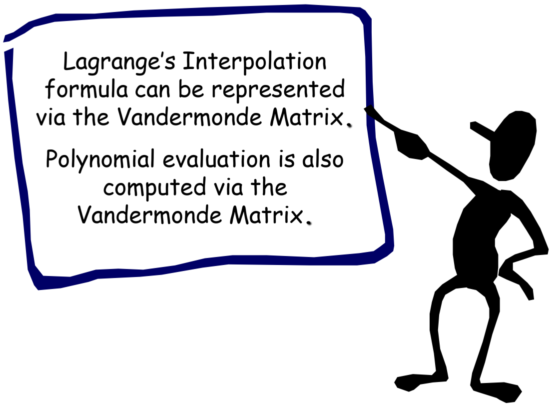
Computing Polynomials

$$A(x) = \sum_{k=0}^{n-1} a_k x^k$$

$$\begin{pmatrix} A(1) \\ A(w) \\ \ldots \\ A(w^{n-1}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & w & w^2 & \ldots & w^n \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & w^{n-1} & w^{2n-2} & \ldots & w^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \ldots \\ a_{n-1} \end{pmatrix}$$

This is the Vandermonde matrix (with $x_0=1$).

We will prove that the above matrix multiplication can be done in $O(n \log n)$



Primitive Roots of Unity

Definition: A complex number w is called a n-th primitive root of unity if

1) $w^n = 1$

2) $w^p \neq 1$, for p = 1, .2, …, n-1

Roots of Unity

Claim 1: Let w be a primitive root of $z^n = 1$ then

$$\sum_{k=0}^{n-1} w^k = 0$$

since $w^n=1$

Proof. Multiply it by w

$$w\sum_{k=0}^{n-1} w^k = w(1+w+\ldots+w^{n-1}) =$$

$$= w + w^2 + \ldots + w^{n-1} + w^n = \sum_{k=0}^{n-1} w^k$$

Roots of Unity

Claim 2: Let w be a primitive root of $z^n = 1$ and p = 1, …, n-1 then

$$\sum_{k=0}^{n-1} w^{kp} = 0$$

Proof.

$$\sum_{k=0}^{n-1} w^{kp} = \sum_{k=0}^{n-1} \left(w^p\right)^k = \frac{\left(w^p\right)^n - 1}{w^p - 1} = \frac{1^p - 1}{w^p - 1} = 0$$

Modular Arithmetic

Consider a set of powers of 2

1,2,4,8,16,32,64,128

modulo 17

1,2,4,8,-1,-2,-4,-8

Square and then do mod 17 again

{1,2,4,8}^2 = {1, 4, 16, 64} ={1,4,-1,-4}

Computing Polynomials

$$A(x)=\sum_{k=0}^{n-1} a_k x^k$$

$$\begin{pmatrix} A(1) \\ A(w) \\ \dots \\ A(w^{n-1}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w & w^2 & \dots & w^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & w^{n-1} & w^{2(n-1)} & \dots & w^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_{n-1} \end{pmatrix}$$

Consider n = 4 (intuition)

Computing Polynomials, n = 4

swap

V_4 V_2

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & i & -i \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -i & i \end{pmatrix} \begin{pmatrix} a_0 \\ a_2 \\ a_1 \\ a_3 \end{pmatrix} \begin{matrix} \text{even} \\ \\ \text{odd} \end{matrix}$$

$$= \begin{pmatrix} V_2 & \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} V_2 \\ V_2 & -\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} V_2 \end{pmatrix}$$

Computing Polynomials

In general case, that can be written as a block matrix

$$V_{2n} = \begin{pmatrix} V_n & D_n V_n \\ V_n & -D_n V_n \end{pmatrix} \begin{pmatrix} a_{even} \\ a_{odd} \end{pmatrix} = \begin{pmatrix} V_n a_{even} + D_n V_n a_{odd} \\ V_n a_{even} - D_n V_n a_{odd} \end{pmatrix}$$

where D_n is a diagonal matrix

$$D_n = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & w & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & w^{n-1} \end{pmatrix}.$$

Proof

$$A(x) = \sum_{k=0}^{n-1} a_k x^k$$

$$\begin{pmatrix} A(1) \\ A(w) \\ \ldots \\ A(w^{n-1}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & w & w^2 & \ldots & w^{n-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & w^{n-1} & w^{2(n-1)} & \ldots & w^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \ldots \\ a_{n-1} \end{pmatrix}$$

Consider j-th row

$$A(w^j) = \sum_{k=0}^{n-1} w^{jk} a_k \quad = \sum_{k \text{ is even}} + \sum_{k \text{ is odd}}$$

Computing Polynomials

$$A(w^j) = \sum_{k=0}^{n/2-1} w^{j2k} a_{2k} + \sum_{k=0}^{n/2-1} w^{j(2k+1)} a_{2k+1}$$

Let w_n denote a root of $z^n = 1$.

Since $w_n^2 = w_{n/2}$, (it follows from $(z^2)^{n/2} = z^n$

$$A(w^j) = \sum_{k=0}^{n/2-1} w_{n/2}^{jk} a_{2k} + w_n^j \sum_{k=0}^{n/2-1} w_{n/2}^{jk} a_{2k+1} = F_1(j) + w_n^j F_2(j)$$

here F_j is a n/2 size problem.

Computing Polynomials

$$A(w^j) = \sum_{k=0}^{n/2-1} w_{n/2}^{jk} a_{2k} + w_n^j \sum_{k=0}^{n/2-1} w_{n/2}^{jk} a_{2k+1}$$

Let us compute $A(w^{j+n/2})$

$$A(w^{j+n/2}) = \sum_{k=0}^{n/2-1} w_{n/2}^{(j+n/2)k} a_{2k} + w_n^{j+n/2} \sum_{k=0}^{n/2-1} w_{n/2}^{(j+n/2)k} a_{2k+1}$$

Observe $w_{n/2}^{j+n/2} = w_{n/2}^j$ and $w_n^{n/2} = -1$ for even n

Periodic property

Symmetry property

Computing Polynomials

$$A(w^j) = F_1(j) + w_n^j F_2(j),\ j = 0, 1, \ldots, n/2-1$$

$$A(w^{j+n/2}) = F_1(j) - w_n^j F_2(j),\ j = 0, 1, \ldots, n/2-1$$

This outlines the divide and conquer algorithm.

Therefore, V.a can be computed in O(n log n)

Computing Polynomials

```
FFT(A, m, w) {
if (m==1) return vector (a_0)
else {
  A_even = (a_0, a_2, ..., a_{m-2})
  A_odd = (a_1, a_3, ..., a_{m-1})
  F_even = FFT(A_even, m/2, w^2)
  F_odd = FFT(A_odd, m/2, w^2)
  x = 1
  for (j=0; j < m/2; ++j) {
    F[j] = F_even[j] + x*F_odd[j]
    F[j+m/2] = F_even[j] - x*F_odd[j]
    x = x * w
  }
return F }
```

High Level Idea

To compute the product A(x)B(x) of polynomials

1) evaluate A(x) and B(x) at roots of unity, using the Vandermonde matrix — O(n log n)
2) multiply $A(x_k)B(x_k)$, — O(n)
3) then find the polynomial using Lagrange's interpolation via the inverse Vandermonde matrix — O(n log n)

Complexity of Interpolation

$$\begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-1} \end{pmatrix}^{-1} \begin{pmatrix} y_0 \\ y_1 \\ \dots \\ y_{n-1} \end{pmatrix}$$

We know that the complexity of interpolation depends on how fast can we inverse the Vandermonde matrix.

We will show that this step is also O(n log n)

Note, each x_k is a primitive root of unity

Inverse Vandermonde

Theorem.

$$V^{-1}(w) = \frac{1}{n} V\left(\frac{1}{w}\right)$$

where $w^n = 1$.

Inverse Vandermonde

$$V(w) = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w & w^2 & \dots & w^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & w^{n-1} & w^{2(n-1)} & \dots & w^{(n-1)(n-1)} \end{pmatrix}$$

Let V* be V where w -> 1/w.

Compute V*.V   Each element of the product is

$$\{1, w^{-i}, w^{-2i}, \dots, w^{-i(n-1)}\} . \{1, w^{j}, w^{2j}, \dots, w^{j(n-1)}\}$$

$$= 1 + w^{j-i} + w^{2(j-i)} + \dots + w^{(n-1)(j-i)}$$

Inverse Vandermonde

$$(V^*.V)(i,j) = 1 + w^{j-i} + w^{2(j-i)} + \dots + w^{(n-1)(j-i)}$$

Recall

$$1 + x + x^2 + \dots + x^{n-1} = \sum_{k=0}^{n-1} x^k = \frac{1 - x^n}{1 - x}$$

It follows, that

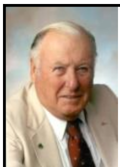
$$(V^*.V)(i,i) = n$$

$$(V^*.V)(i, j \neq i) = 0$$

V*.V = n I

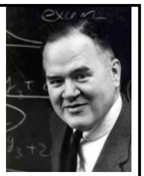
V(1/w).V(w) = n I

FFT History



Cooley and Tukey's paper 1965

It was known to Gauss, 1805.



Tukey derived the basic reduction while in a meeting of President Kennedy's Science Advisory Committee for off-shore detection of nuclear tests in the Soviet Union.

The idea was to analyze time series obtained from seismometers. Other possible applications to national security included the long-range acoustic detection of nuclear submarines.

High Level Idea

To compute the product A(x)B(x) of polynomials

1) evaluate A(x) and B(x) at roots of unity, using the Vandermonde matrix — O(n log n)
2) multiply $A(x_k)B(x_k)$, — O(n)
3) then find the polynomial using Lagrange's interpolation via the Vandermonde matrix — O(n log n)

Polynomial multiplication

$$a_0, a_1, \ldots, a_{n-1}$$
$$b_0, b_1, \ldots, b_{n-1}$$

FFT

$$A(1), A(w), A(w^2), \ldots, A(w^{2n-1})$$
$$B(1), B(w), B(w^2), \ldots, B(w^{2n-1})$$

Point-value multiplication

$$A(1)B(1), A(w)B(w), \ldots, A(w^{2n-1})B(w^{2n-1})$$

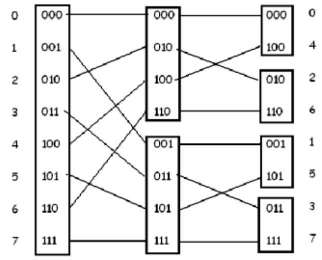
Inverse FFT

$$C(x) = \sum_{k=0}^{2n-2} c_k x^k$$

FFT in place

Let n = 8

The right column is a bit reversal!.



The recursive algorithm can simply call on the left and right halves, rather than on the odd and even indices.

All DSP processors include a hardware bit reversal capability

Discrete Fourier Transform

DFT converts a set of sample points into another set ordered by frequencies. It reveals periodicities in input data.

A DFT of $\{a_0, a_1, \ldots, a_{n-1}\}$ is defined by

$$b_j = \sum_{k=0}^{n-1} a_k w^{kj}$$

where $w^n = 1$. In a matrix form V.a = b

FFT is an algorithm for computing DFT.

Convolution

The convolution of two vectors a_k and b_k is a third vector $c = a \otimes b$ which represents an overlap between the two vectors.

$$c_j = \sum_{k=0}^{n-1} a_k b_{j-k} \qquad c_j = \sum_{k=-\infty}^{\infty} a_k b_{j-k}$$

The Convolution Theorem says that the DFT of a convolution of two vectors is the point-wise product of the DFT of the two vectors

$$DFT(a \oplus b) = DFT(a)\,DFT(b)$$

Convolution

$$DFT(a \oplus b) = DFT(a)\, DFT(b)$$

$$a \oplus b = DFT^{-1}(DFT(a)\, DFT(b))$$

It follows, using FFT we can compute convolution in O(n log n).

Note that inverse DFT is just a regular DFT with w replaced by w^{-1}.

Polynomial multiplication

$$A(x) = \sum_{k=0}^{n-1} a_k x^k \qquad B(x) = \sum_{k=0}^{n-1} b_k x^k$$

$$A(x)B(x) = \sum_{k=0}^{2n-2} c_k x^k \qquad c_k = \sum_{j=0}^{k} a_j b_{k-j}$$

this is just a convolution of two vectors a and b

Finite Fields (mod prime p)

Consider a set of powers of 2

1,2,4,8,16,32,64,128

modulo p=17

How do we find such prime p?