Algorithm Design and Analysis

Victor Adamchik — CS 15-451 — Spring 2014

Lecture 3 — Jan 17, 2014 — Carnegie Mellon University

# Fast Fourier Transform

Gauss
(1777 – 1855)

Lagrange
(1736 –1813)

Fourier
(1768 –1830)

## Outline

1) Legendre's Interpolation
2) Vandermonde Matrix
3) Roots of Unity
4) Polynomial Evaluation

## High Level Idea

To compute the polynomial (order n) product A(x)B(x),

1) evaluate A(x) and B(x) at some (2n+1) points $x_k$,
2) multiply $A(x_k)B(x_k)$,
3) then find the polynomial which passes through these points.

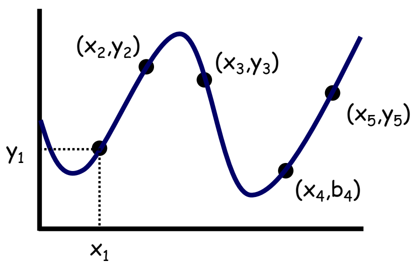
## The worst-case complexity

To compute the polynomial product A(x)B(x),

1) evaluate A(x) and B(x) at some points $x_k$,
   - what's the complexity of $A(x_k)$-?
   - what's the complexity of $A(x_k)$ at n points?
2) multiply $A(x_k)B(x_k)$,
3) then find the polynomial which passes through these points.
   - how can we do it? What is it complexity?

## Interpolation

Given a set of points




Find a (low-degree) polynomial which "fits the data"

## Interpolation

Given pairs $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$

Theorem:

There is exactly one polynomial P(x) of degree at most n such that $P(x_k) = y_k$ for all k = 0, …, n.

## Theorem Proof

There are two things to prove.

1. There is at *least* one polynomial of degree $\leq n$ passing through all n+1 data points.

2. There is at *most* one polynomial of degree $\leq n$ passing through all n+1 data points.

Let's prove #2 first.

## Proof #2: There is at *most* one polynomial

Suppose P(x) and Q(x) both do the trick.

Let $R(x) = P(x) - Q(x)$.

Since deg(P), deg(Q) $\leq n$ we must have deg(R) $\leq n$.

But $R(a_k) = b_k - b_k = 0$ for all $k = 0, \ldots, n$.

Thus R(x) has more roots (n+1) than its degree.

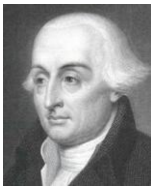
Thus, R(x) must be the 0 polynomial, i.e.,

$P(x)=Q(x)$.

## Proof #1

The method for constructing the polynomial is called Lagrange's Interpolation.

Discovered in 1795 by J.-L. Lagrange.



## Lagrange Interpolation

| | |
|---|---|
| $a_0$ | $b_0$ |
| $a_1$ | $b_1$ |
| $a_2$ | $b_2$ |
| … | … |
| $a_{n-1}$ | $b_{n-1}$ |
| $a_n$ | $b_n$ |

Want P(x) with degree $\leq n$ such that $P(a_k) = b_k \ \forall k$.

## Special Case

| | |
|---|---|
| $a_0$ | 1 |
| $a_1$ | 0 |
| $a_2$ | 0 |
| … | … |
| $a_{n-1}$ | 0 |
| $a_n$ | 0 |

Once we solve this special case, the general case is very easy.

## Special Case

| | |
|---|---|
| $a_0$ | 1 |
| $a_1$ | 0 |
| $a_2$ | 0 |
| … | … |
| $a_{n-1}$ | 0 |
| $a_n$ | 0 |

Let $Q(x) = (x-a_1)(x-a_2)\cdots(x-a_n)$

Degree is n. ✔

$Q(a_1) = Q(a_2) = \cdots = Q(a_n) = 0$. ✔

$Q(a_0) =$??

## Lagrange Interpolation

Numerator is a deg. n polynomial

Denominator is a nonzero field element

| | |
|---|---|
| $a_0$ | 1 |
| $a_1$ | 0 |
| $a_2$ | 0 |
| … | … |
| $a_{n-1}$ | 0 |
| $a_n$ | 0 |

$$S_0(x) = \frac{Q(x)}{Q(a_0)} = \frac{(x-a_1)\dots(x-a_n)}{(a_0-a_1)\dots(a_0-a_n)}$$

Call this the selector polynomial for $a_0$.

## Another special case

| | |
|---|---|
| $a_0$ | 0 |
| $a_1$ | 1 |
| $a_2$ | 0 |
| … | … |
| $a_{n-1}$ | 0 |
| $a_n$ | 0 |

$$S_1(x) = \frac{(x-a_0)(x-a_2)\dots(x-a_n)}{(a_1-a_0)(a_1-a_2)\dots(a_1-a_n)}$$

## Lagrange Interpolation

| | |
|---|---|
| $a_0$ | 0 |
| $a_1$ | 0 |
| $a_2$ | 0 |
| … | … |
| $a_{n-1}$ | 0 |
| $a_n$ | 1 |

$$S_n(x) = \frac{(x-a_0)\dots(x-a_{n-1})}{(a_n-a_0)\dots(a_n-a_{n-1})}$$

## Great! But what about this data?

| | |
|---|---|
| $a_0$ | $b_0$ |
| $a_1$ | $b_1$ |
| $a_2$ | $b_2$ |
| … | … |
| $a_{n-1}$ | $b_{n-1}$ |
| $a_n$ | $b_n$ |

$P(x) = b_0 S_0(x) + \dots + b_n S_n(x)$

This formula is called Lanrange's Interpolation

## Lagrange Interpolation

Given pairs $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$

There is a unique polynomial to fit these points:

$$y(x) = \sum_{j=0}^{n} y_j \prod_{\substack{k=0 \\ k\neq j}}^{n} \frac{x-x_k}{x_j-x_k}$$

## Example

Given two polynomials

$A(x) = 1 + x + x^2$ $B(x) = 1 + 2x + 3x^2$

Compute their values at x = -2, -1, 0, 1, 2

{A(-2), A(-1), A(0), A(1), A(2)} = {3, 1, 1, 3, 7}

{B(-2), B(-1), B(0), B(1), B(2)} = {9, 2, 1, 6, 17}

Pointwise multiplication:

{C(-2), C(-1), C(0), C(1), C(2)} = {27, 2, 1, 18, 119}

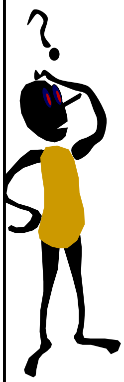
## Example

Points to fit

(-2,27), (-1,2), (0, 1), (1, 18), (2, 119)

This yields $$y(x) = \sum_{j=0}^{n} y_j \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}$$

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



$$y(x) = \sum_{j=0}^{n} y_j \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}$$

What is the runtime complexity of Lagrange's interpolation?

$O(n^3)$, if we expand

## Matrix Form

Consider a case of two points $\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$

$$y = \frac{y_1 x_0 - y_0 x_1}{x_0 - x_1} + \frac{y_0 - y_1}{x_0 - x_1} x = a_0 + a_1 x$$

This could be written in a matrix form

$$\begin{pmatrix} 1 & x_0 \\ 1 & x_1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix}$$

that defines $a_0$ and $a_1$.

## The Vandermonde Matrix

The Lagrange formula defines a polynomial $A(x) = \sum_{k=0}^{n} a_k x^k$

where coefficients $a_k$ can be found by solving this

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{pmatrix}$$

or in short V.a = y Thus, a = $V^{-1}$.y

## The Vandermonde Matrix

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix}$$

In order to inverse the matrix V, we have to prove that it's nonsingular.

## Determinant of the Vandermonde Matrix

$$\det \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix} = \prod_{k=0}^{n} \prod_{j=0}^{k-1} (x_k - x_j)$$

Since the n + 1 points are distinct, the determinant can't be zero.

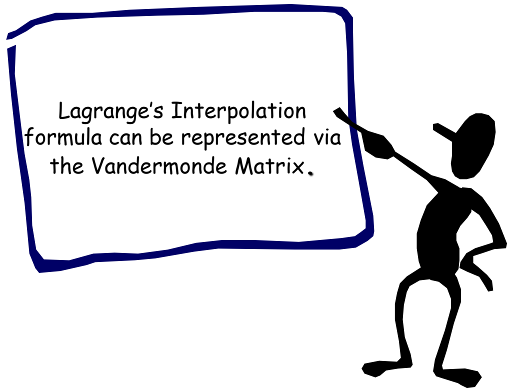
The proof (by induction on n) is left as an exercise to a student ☺

## Complexity of Interpolation

$$\begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix}^{-1} \begin{pmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{pmatrix}$$

It follows that the complexity of interpolation depends on how fast can we inverse the Vandermonde matrix.

The success depends on the values of $x_k$, k=0, …,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)

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

## 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 has 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,

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

## Worst-time Complexity

Let T(n) be the complexity of computing a degree-n polynomial at 2n+1 points. Thus

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

This solves to O(n log n).

Great! 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

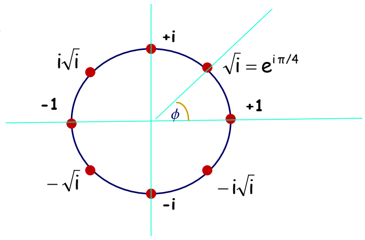
1) half of points are negative and the second half is positive

2) this property holds after squaring

## 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 of degree n.

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

Our task to compute

$A(1), A(w), A(w^2),\ldots$

where $w^{n+1} = 1$.

We can write these computations in a matrix form!!!

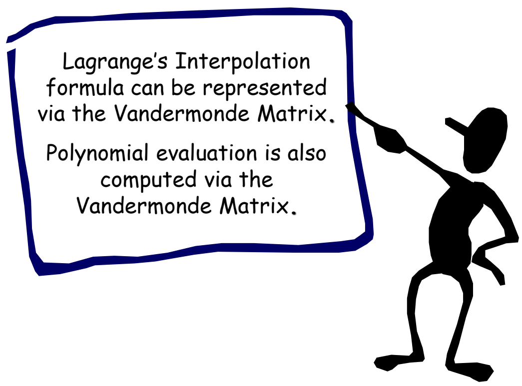
## Computing Polynomials

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

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

This is the Vandermonde matrix.

We will prove that this matrix multiplication can be done in O(n log n)



## High Level Idea

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

1) evaluate A(x) and B(x) at (2n+1) 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)