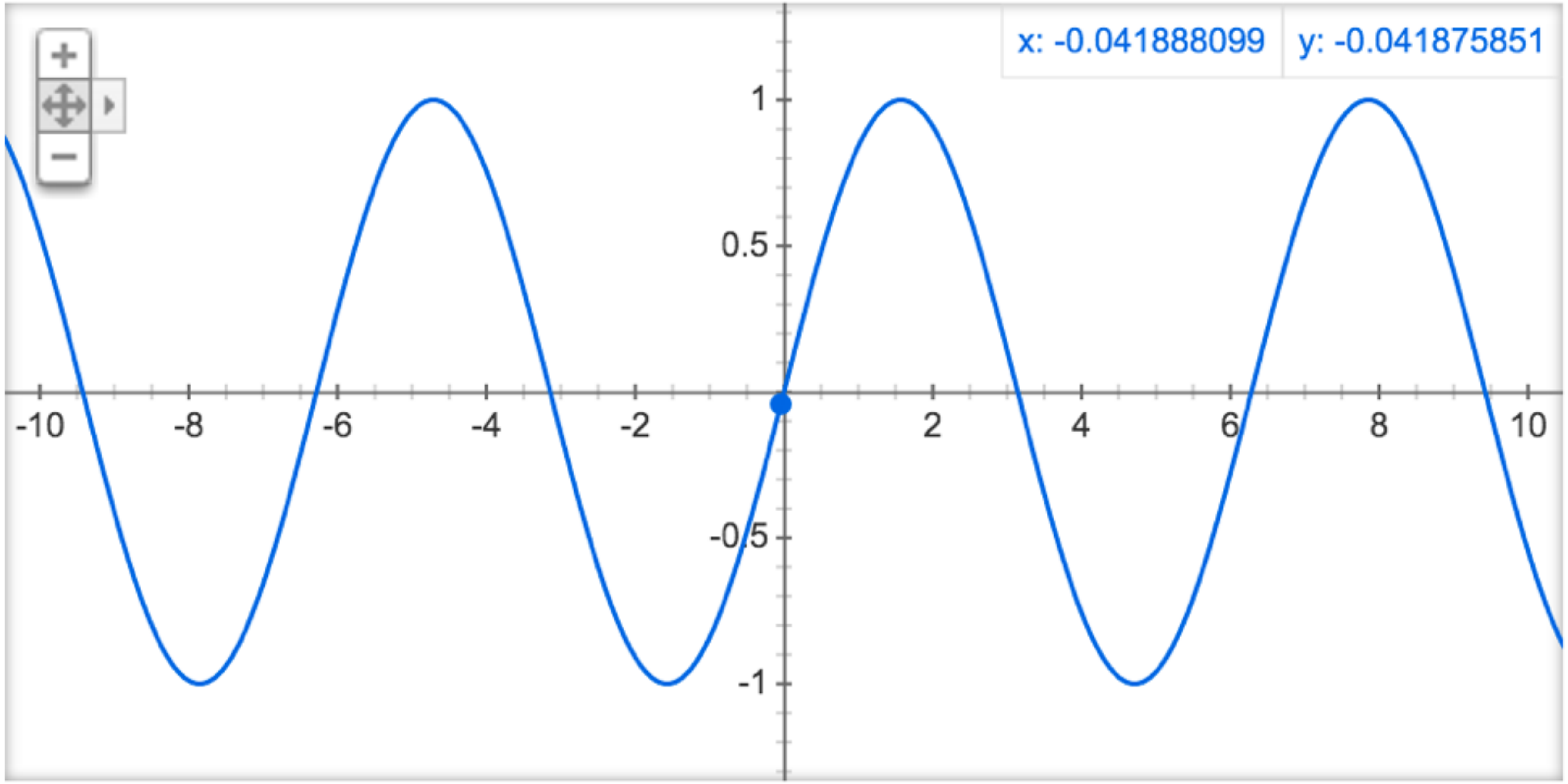
L7

feb 11 2016

abhi shelat

Matrix, FFT

Graph for sin(x)



$\cos\left(\frac{\pi}{4}\right) = \frac{1}{\sqrt{2}}$

$\sin\left(\frac{\pi}{4}\right) = \frac{1}{\sqrt{2}}$

$\cos\left(\frac{\pi}{2}\right) = 0$

$\sin\left(\frac{\pi}{2}\right) = 1$

userid:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \bigstar \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \quad =$$

Using the standard method, how many multiplications does it take to multiply two NxN matrices? ______________

$\cos(\pi/4) =$

$\cos(\pi/2) =$

$\sin(\pi/4) =$

$\sin(\pi/2) =$

Mergesort

Karatsuba

Closest pair

arbitrage

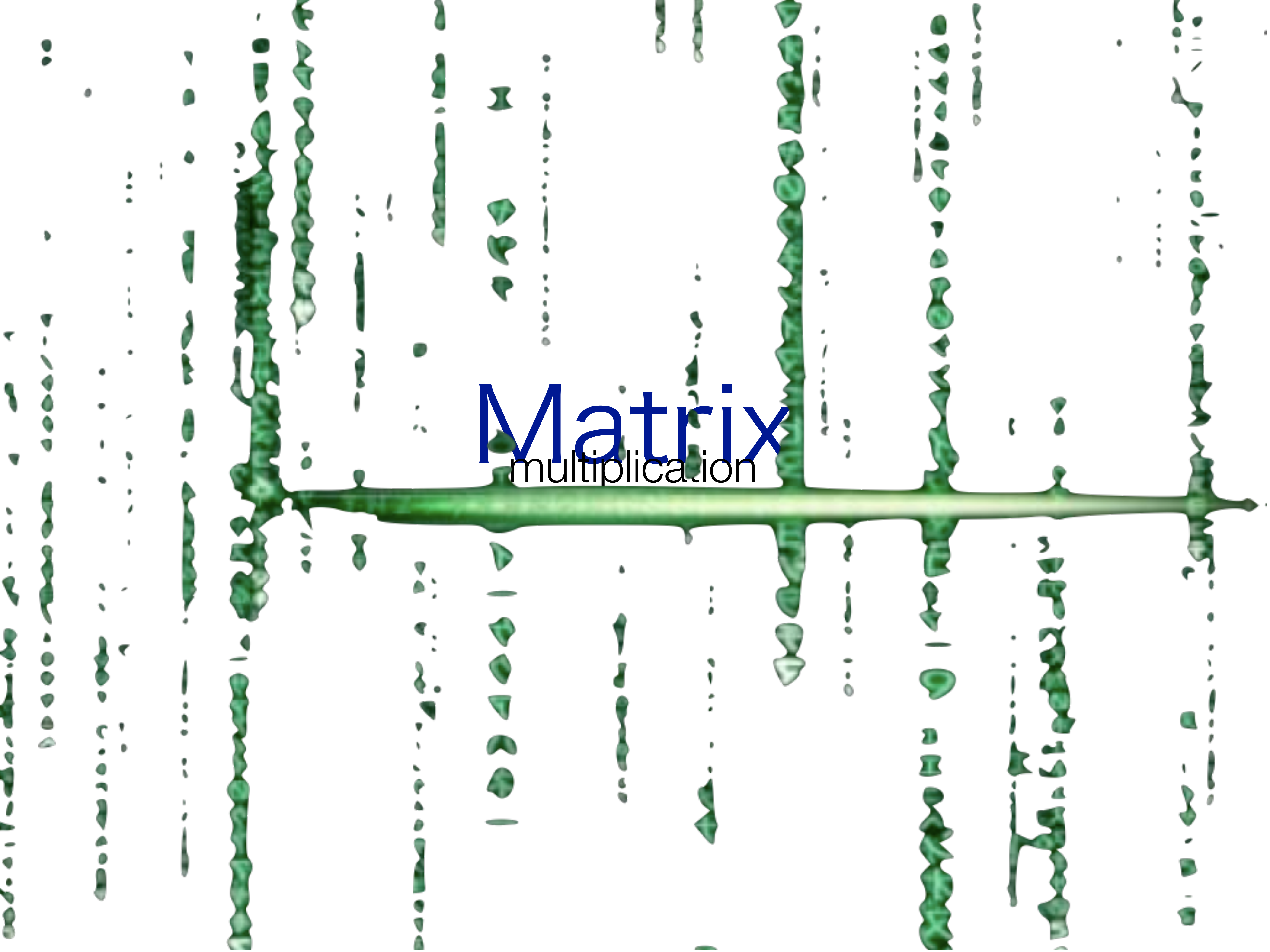
Matrix

FFT

MEDIAN

Matrix

multiplication



$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \star \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \left[\begin{matrix} \boxed{5} + 14 = 19 \\ \\ \\ \end{matrix} \right.$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \bigstar \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 5+14 & 6+16 \\ 15+28 & 18+32 \end{bmatrix}$$

$$= \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

$$\begin{array}{cc} & n \\ n & \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \end{array} \begin{array}{cc} & n \\ n & \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & & & \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix} \end{array} = \begin{array}{cc} & n \\ n & \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & & & \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix} \end{array}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & & & \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & & & \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix}$$

$$\underline{c_{i,j}} = \sum_{k=1}^{n} \underbrace{a_{i,k} \cdot b_{k,j}}_{}$$

$\Theta(n)$

matrix-mult: n^2 terms $\cdot$ n ops

$= \Theta(n^3)$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & & & \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & & & \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix}$$

Divide each matrix into 4 matrices that are $\frac{n}{2} \times \frac{n}{2}$.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \cdot \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} A \cdot E + B \cdot G & A \cdot F + B \cdot H \\ C \cdot E + D \cdot G & C \cdot F + D \cdot H \end{bmatrix}$$

where each is an $\frac{n}{2} \times \frac{n}{2}$ matrix.

$$T(n) = 8T\left(\frac{n}{2}\right) + \underbrace{4\left(\frac{n}{2}\right)^2}_{}$$

$\Theta(n^2)$

case 1 applies:

$$\Theta\left(n^{\log_2 8}\right) = \Theta\left(n^3\right).$$

$$
\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix}
$$

$$
= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}
$$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

$$T(n) = 8T(n/2) + \Theta(n^2)$$

$$\Theta(n^3)$$

$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

$P_1 + P_2 = AF - \cancel{AH} + \cancel{AH} + BH$

$= AF + BH$

[Strassen]

$P_3 + P_4 = CE + \cancel{D}E - \cancel{D}E + DG = CE + DG$

$$P_1 = A(F - H)$$

$$P_2 = (A + B)H$$

$$P_3 = (C + D)E$$

$$P_4 = D(G - E)$$

$$P_5 = (A + D)(E + H)$$

$$P_6 = (B - D)(G + H)$$

$$P_7 = (A - C)(E + F)$$

$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

$R = P_5 + P_4 - P_2 + P_6$ ✓ $S = P_1 + P_2$

$T = P_3 + P_4$ $U = P_5 + P_1 - P_3 - P_7$ ✓

[strassen]

$P_1 = A \cdot (F - H)$

$P_2 = (A + B) \cdot H$

$P_3 = (C + D) \cdot E$

$P_4 = D \cdot (G - E)$

$P_5 = (A + D) \cdot (E + H)$

$P_6 = (B - D) \cdot (G + H)$

$P_7 = (A - C) \cdot (E + F)$

$\boxed{AE} + \cancel{AH} + \cancel{DE} + \cancel{DH}$

$- \cancel{DE} + \cancel{DG}$

$- \cancel{AH} - \cancel{BH}$

$\boxed{+BG} - \cancel{DG} + \cancel{BH} - \cancel{DH}$

$T(n) = 7T\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2$

$= \Theta\left(n^{\log_2 7}\right)$

$\sim n^{2.805}$

$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

$$R = P_5 + P_4 - P_2 + P_6 \qquad S = P_1 + P_2$$

$$T = P_3 + P_4 \qquad U = P_5 + P_1 - P_3 - P_7$$

[strassen]

$$P_1 = A(F - H)$$

$$P_2 = (A + B)H$$

$$P_3 = (C + D)E$$

$$P_4 = D(G - E)$$

$$P_5 = (A + D)(E + H)$$

$$P_6 = (B - D)(G + H)$$

$$P_7 = (A - C)(E + F)$$

$$= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

$R = P_5 + P_4 - P_2 + P_6$ $S = P_1 + P_2$

$T = P_3 + P_4$ $U = P_5 + P_1 - P_3 - P_7$

[strassen]

$P_1 = A(F - H)$

$P_2 = (A + B)H$

$P_3 = (C + D)E$

$P_4 = D(G - E)$

$P_5 = (A + D)(E + H)$

$P_6 = (B - D)(G + H)$

$P_7 = (A - C)(E + F)$

$$M(n) = 7M(n/2) + \underline{18n^2}$$

$$= \Theta(\underline{n^{\log_2 7}})$$

$2.^{805}$

taking this idea further

3x3 matricies [Laderman'75]

to use 23 matrix multiplication of size $\frac{n}{3} \times \frac{n}{3}$ + [ ] addition

$$T(n) = 23\,T\left(\frac{n}{3}\right) + \Theta(n^2)$$

$$= \Theta\left(n^{\log_3 23}\right) = n^{2.854}$$

If he had 21

$$n^{\log_3 21} = 2.771$$

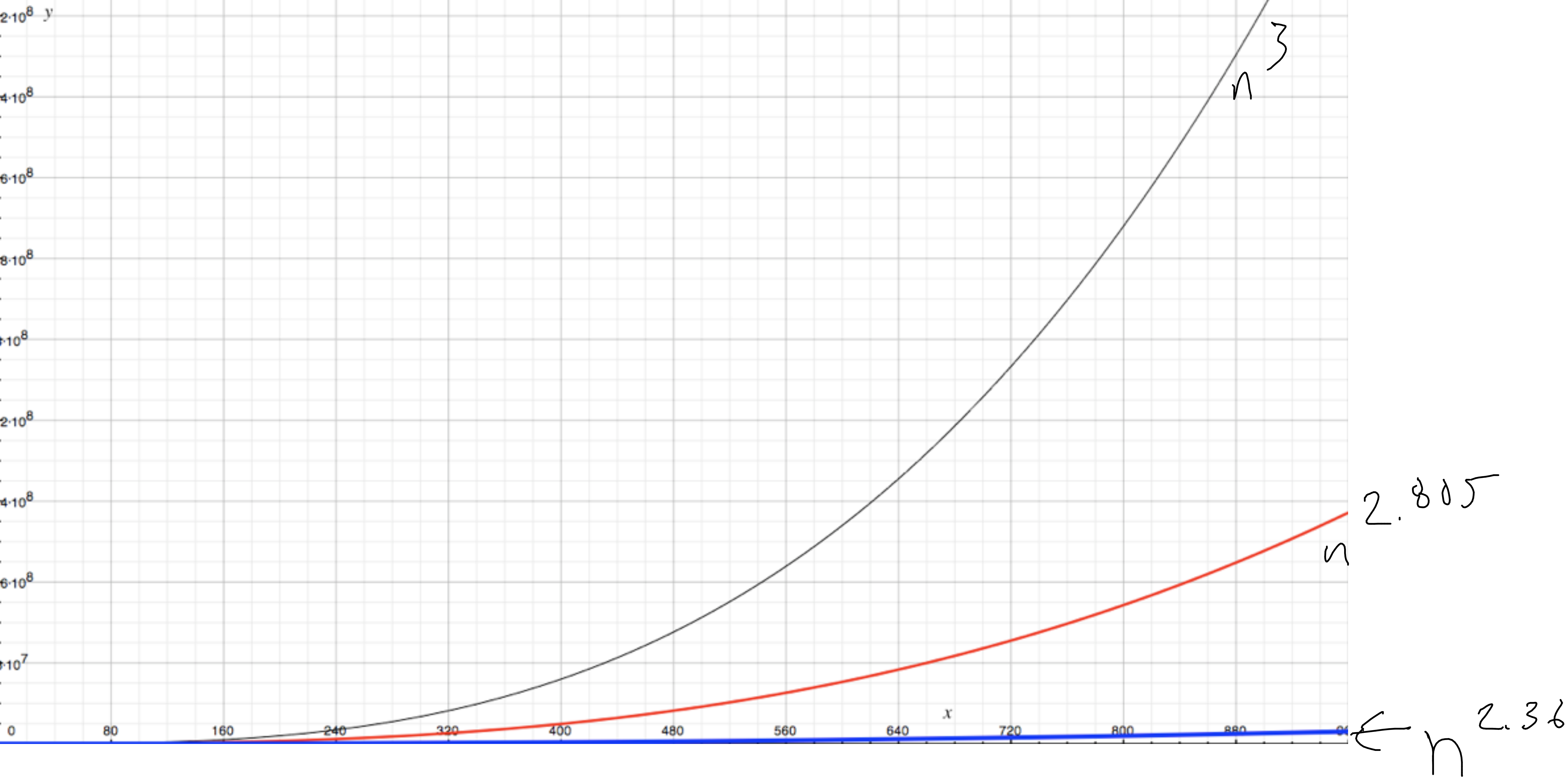
1978 victor pan method

70x70 matrix using 143640 mults

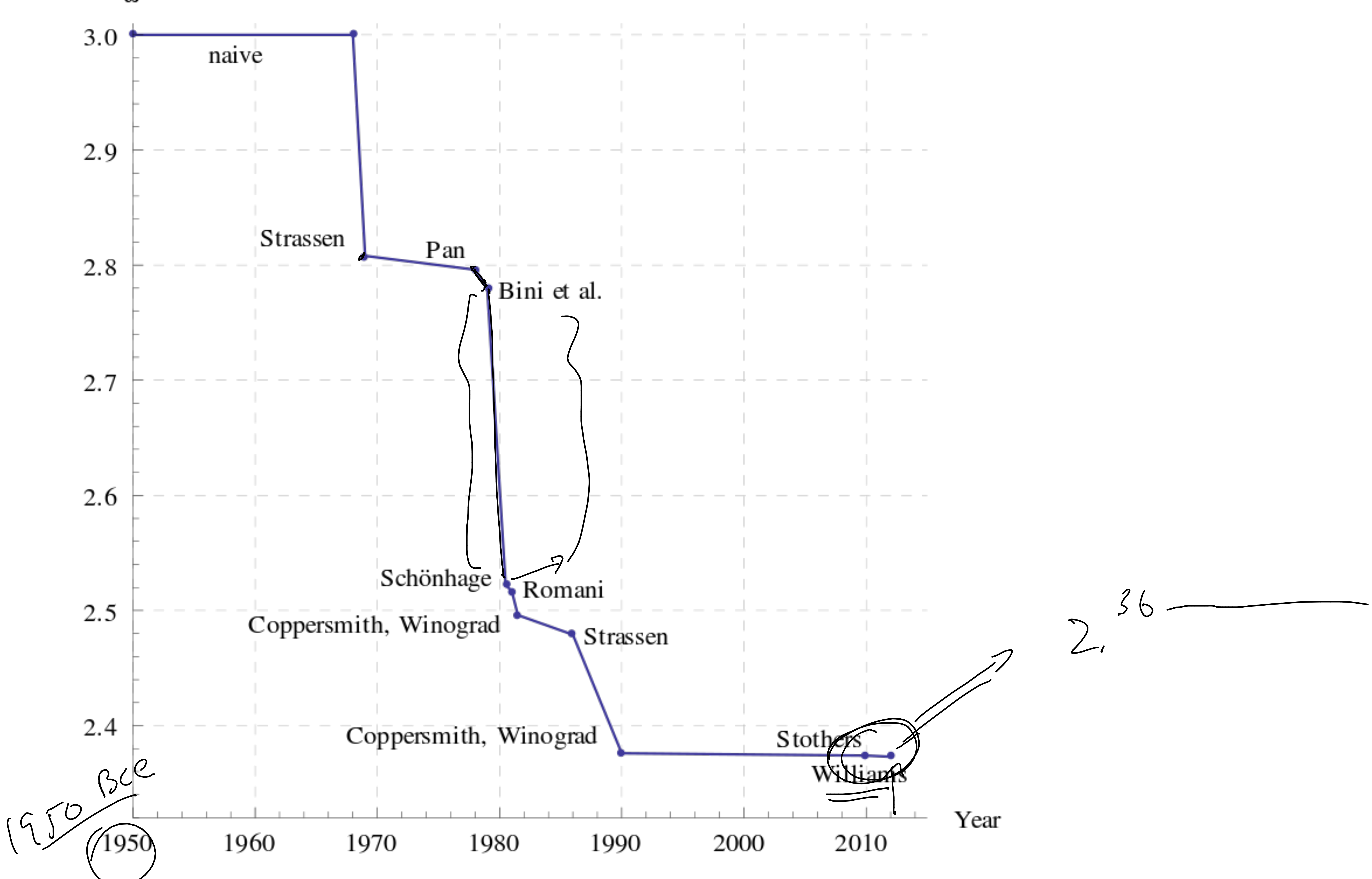
what is the recurrence:

$$T(n) = 143640\, T\left(\frac{n}{70}\right) + \Theta(n^2)$$

$$= \Theta\left(n^{\log_{70} 143640}\right) \sim n^{2.795}$$



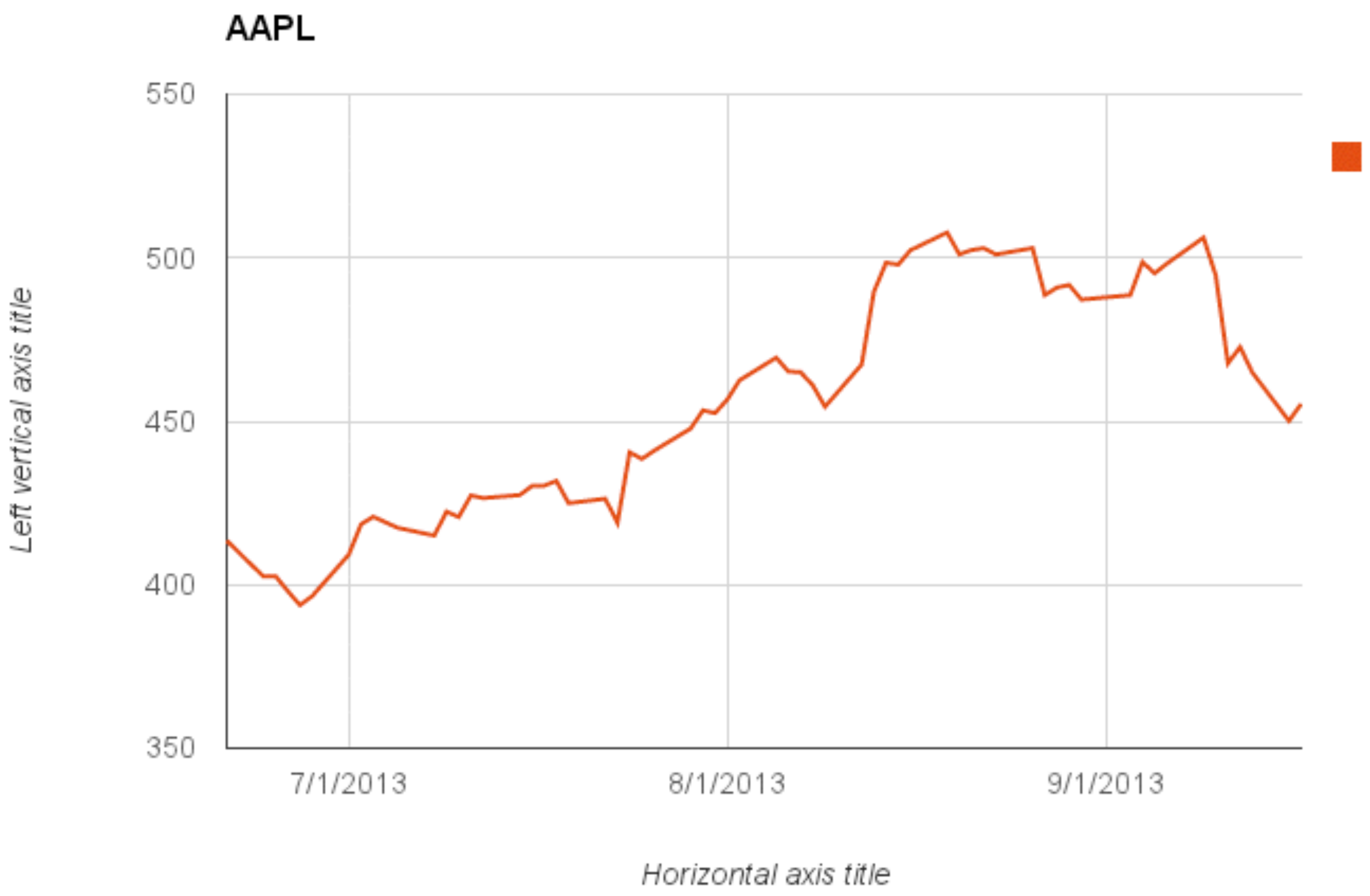
y
x
0
80
160
240
320
400
480
560
640
720
800
880
n^3
$n^{2.805}$
$n^{2.36}$



https://en.wikipedia.org/wiki/File:Bound_on_matrix_multiplication_omega_over_time.svg

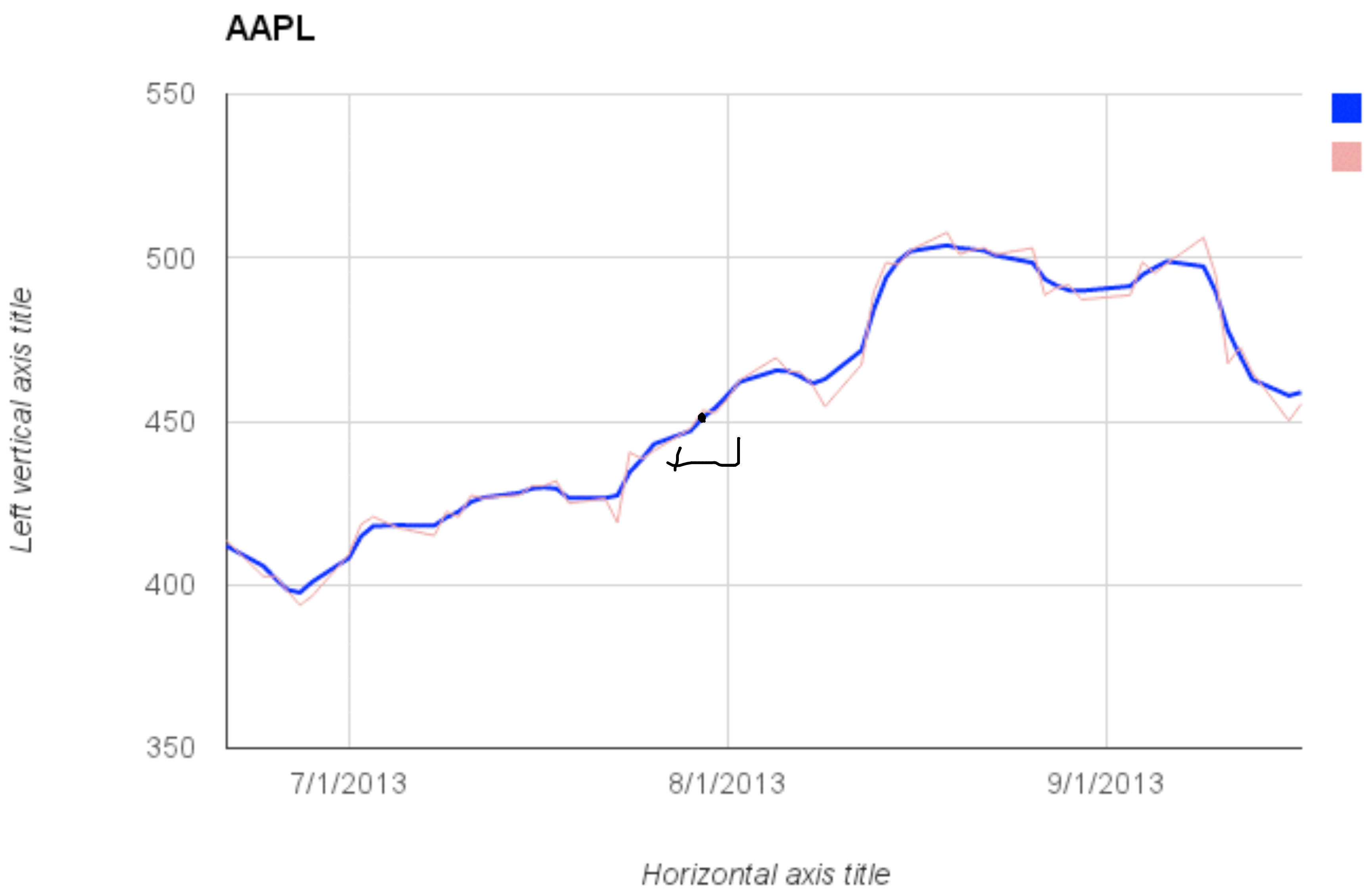
Fast Fourier Transform

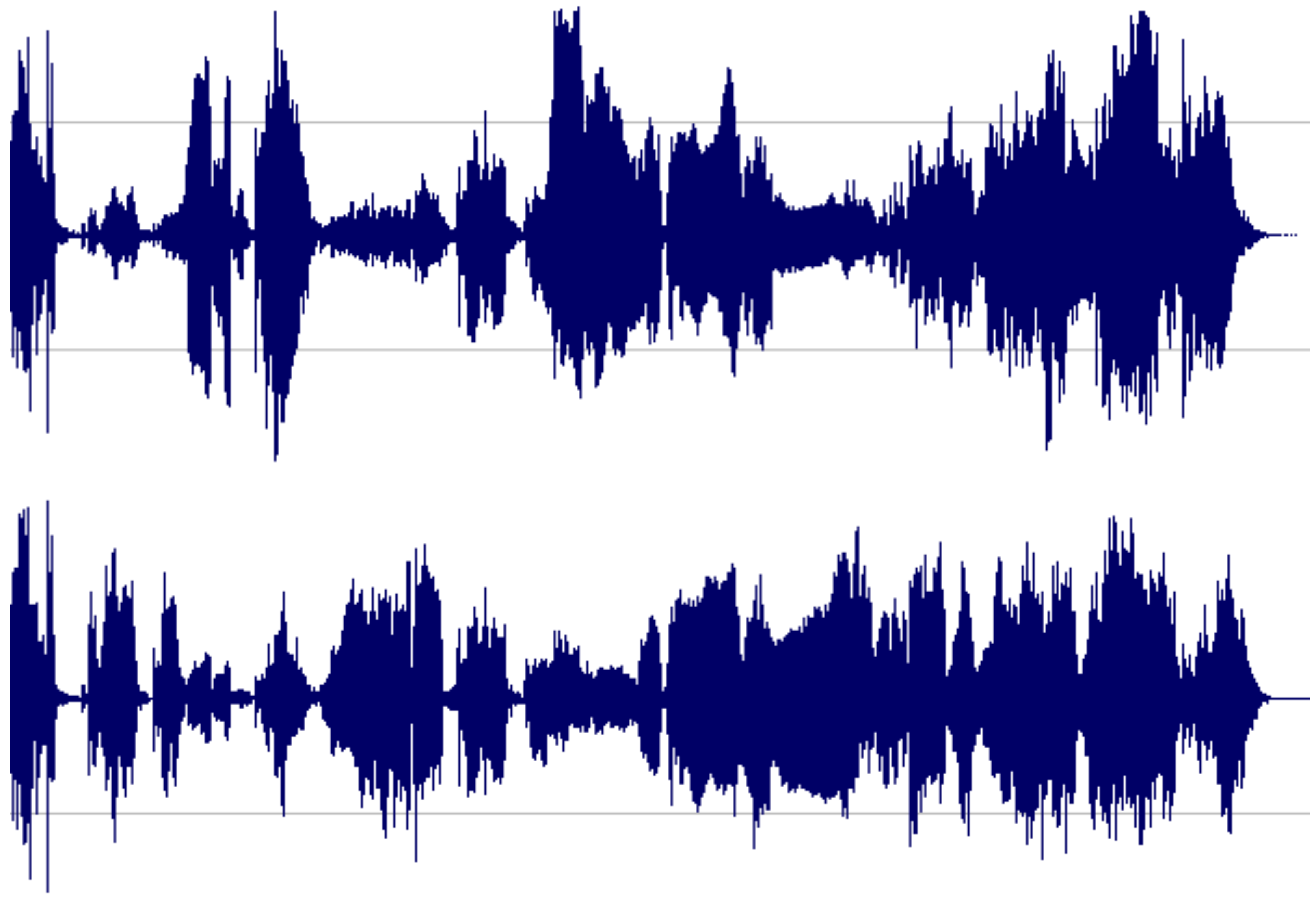
FF-1



AAPL
550
500
450
400
350
7/1/2013
8/1/2013
9/1/2013
Left vertical axis title
Horizontal axis title

FFT


AAPL
550
500
450
400
350
7/1/2013
8/1/2013
9/1/2013
Horizontal axis title
Left vertical axis title



big ideas: FFT

① change of representation of a polynomial

coefficient form $\longleftrightarrow$ point-wise form

② Divide & Conquer strategy

$$f(x) = 5 + 2x + 1x^2$$

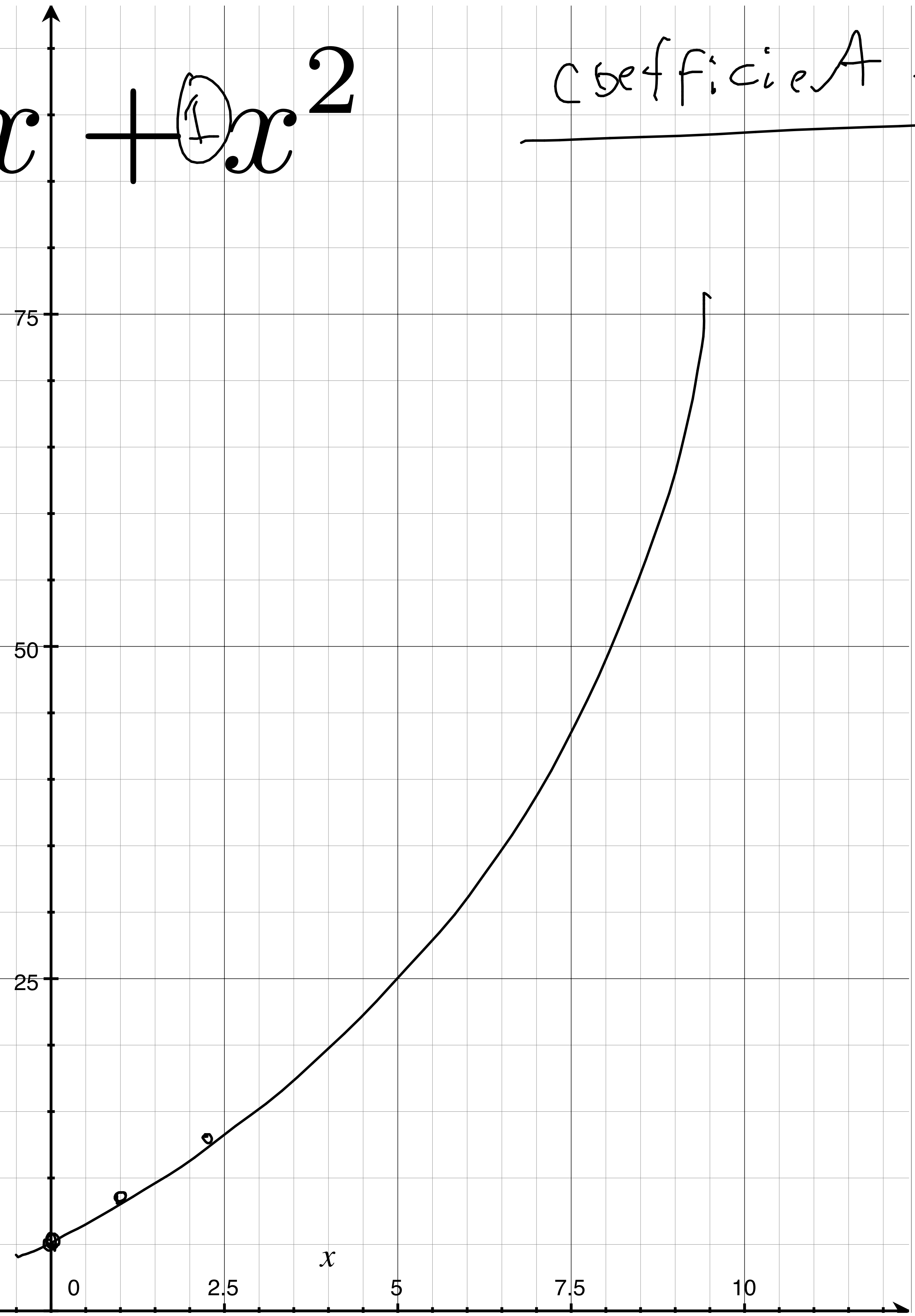
coefficient-form

$f(0) = 5$

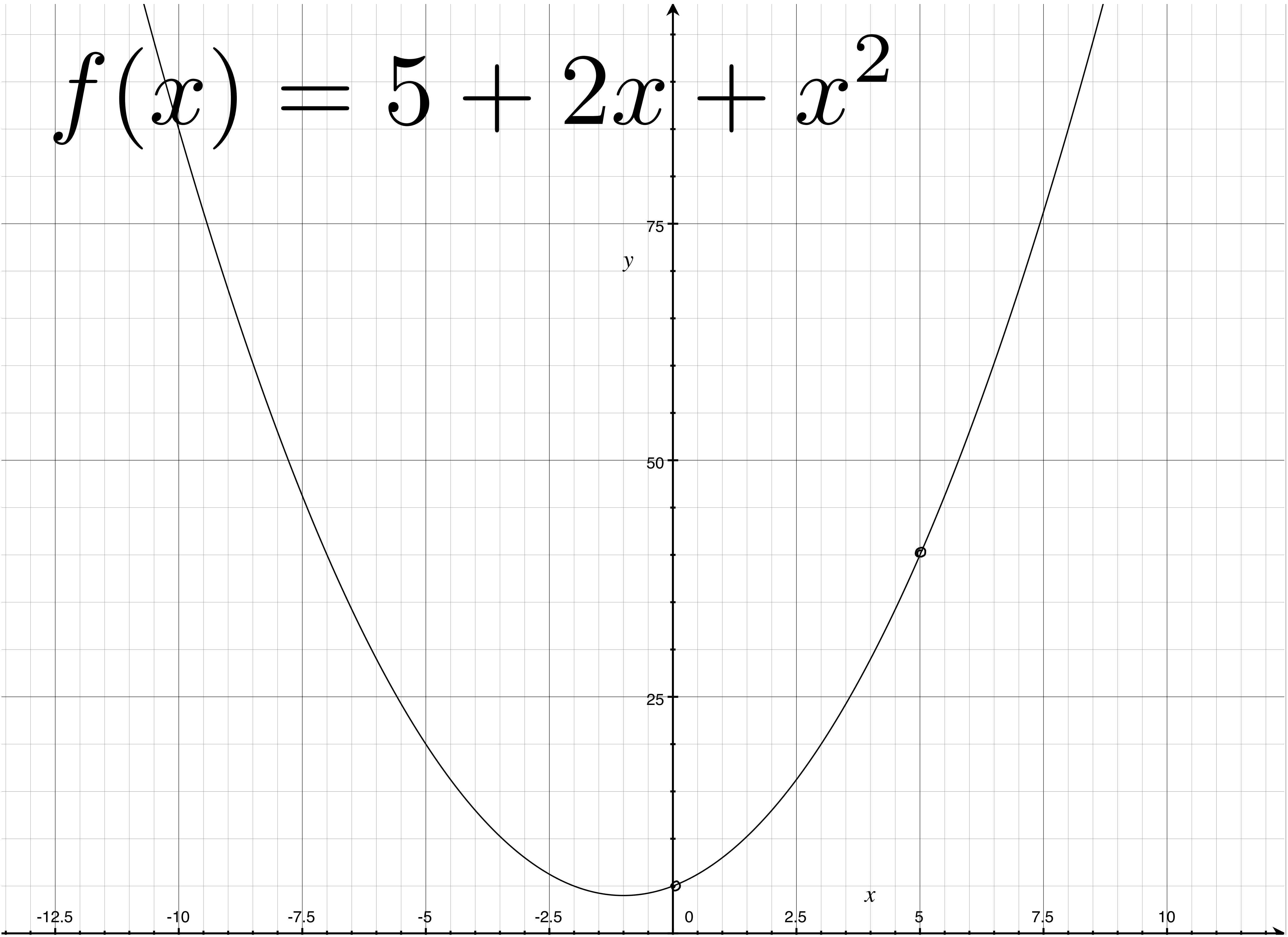
$f(1) = 8$

$f(2) = 13$

point-wise form



$$f(x) = 5 + 2x + x^2$$

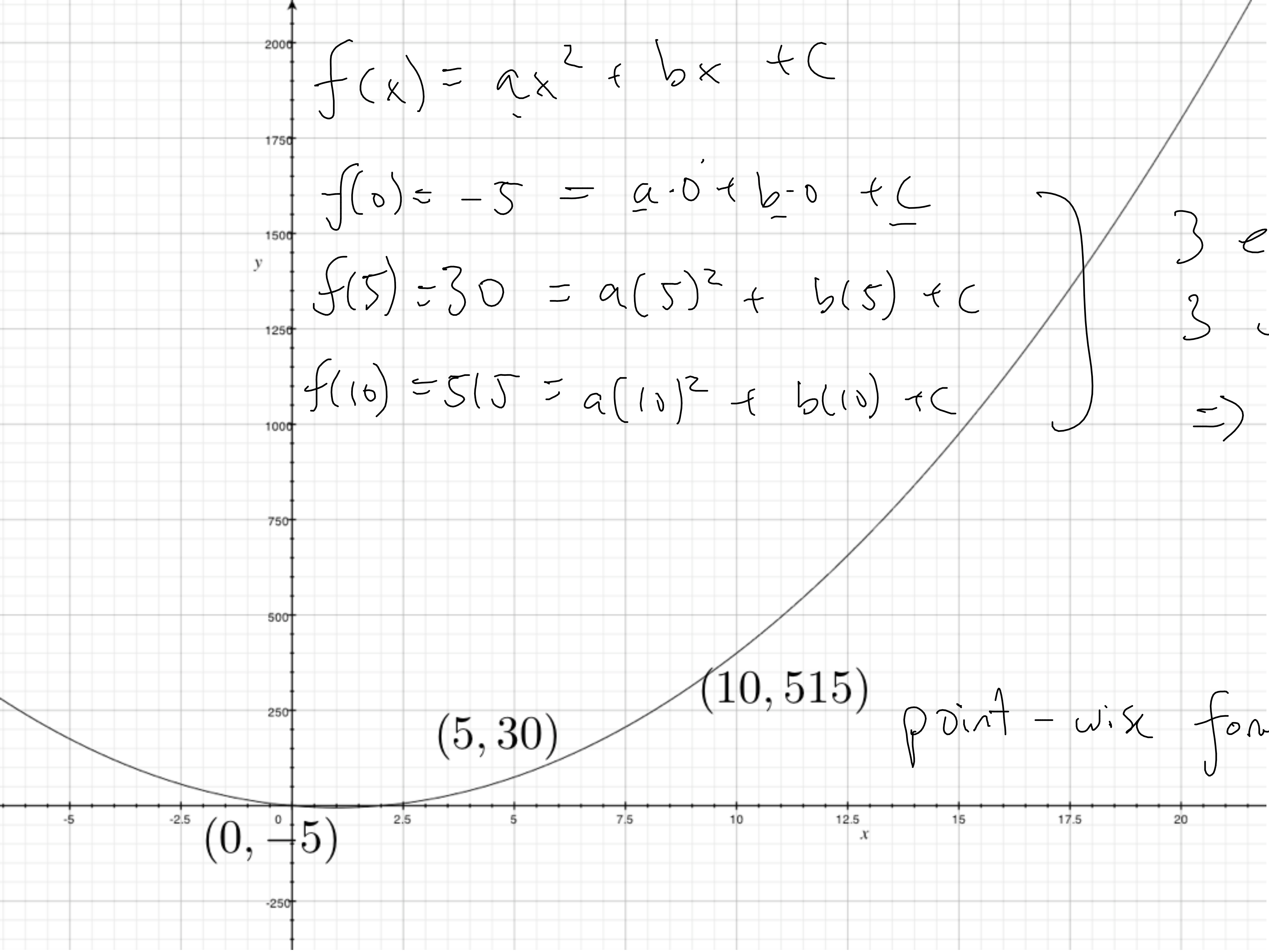


$$f(x) = ax^2 + bx + c$$

$$f(0) = -5 = a\cdot 0 + b\cdot 0 + c$$
$$f(5) = 30 = a(5)^2 + b(5) + c$$
$$f(10) = 515 = a(10)^2 + b(10) + c$$

3 equations
3 unknowns
$\Rightarrow$ we can solve this system

point-wise form

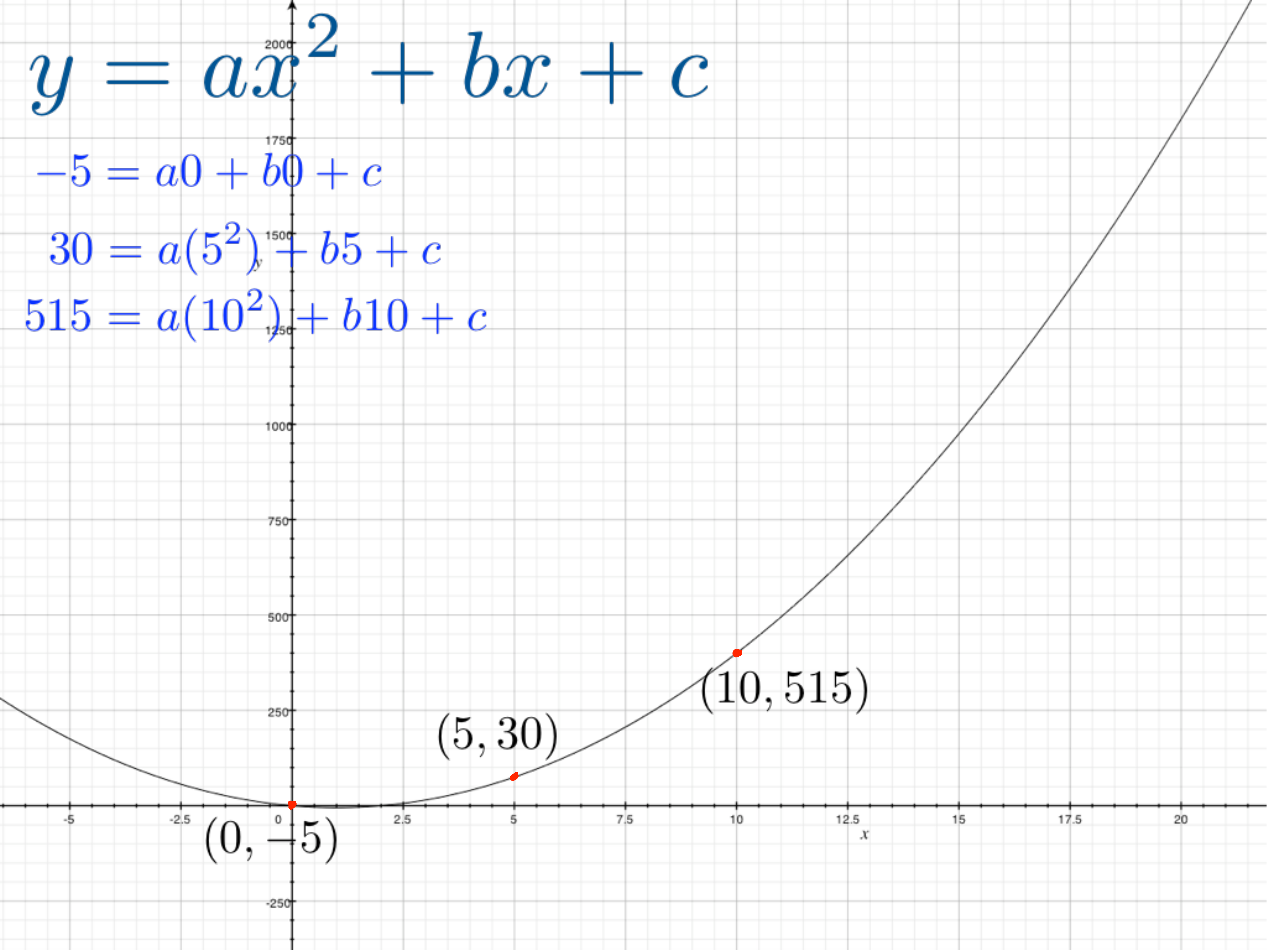


$$y = ax^2 + bx + c$$

$$-5 = a0 + b0 + c$$

$$30 = a(5^2) + b5 + c$$

$$515 = a(10^2) + b10 + c$$



$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

how efficient is the transformation ??

Brute force

$A(1) = $ ... (compute)

$A(2) = $ ...

$\vdots$

$A(n) = $ ...

how much time would simple evaluation require ??

$\Theta(n^2)$

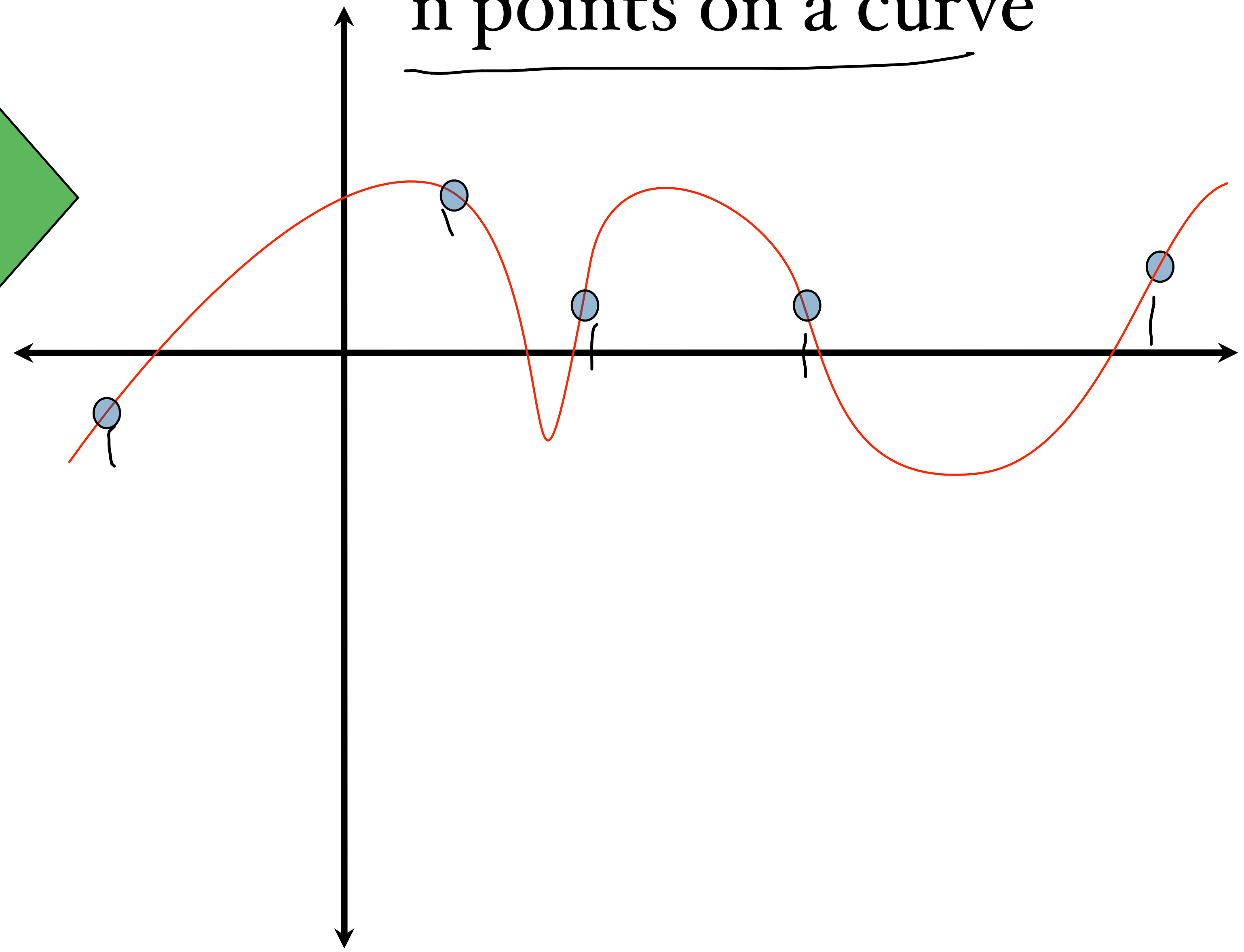
degree $n-1$ polynomial

$$A(x)$$

coefficient form

point-wise form

n points on a curve



FFT

input: $a_0, a_1, a_2, \ldots, a_{n-1}$

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

output: $A(w_0), A(w_1), A(w_2) \ldots A(w_n)$ n distinct points

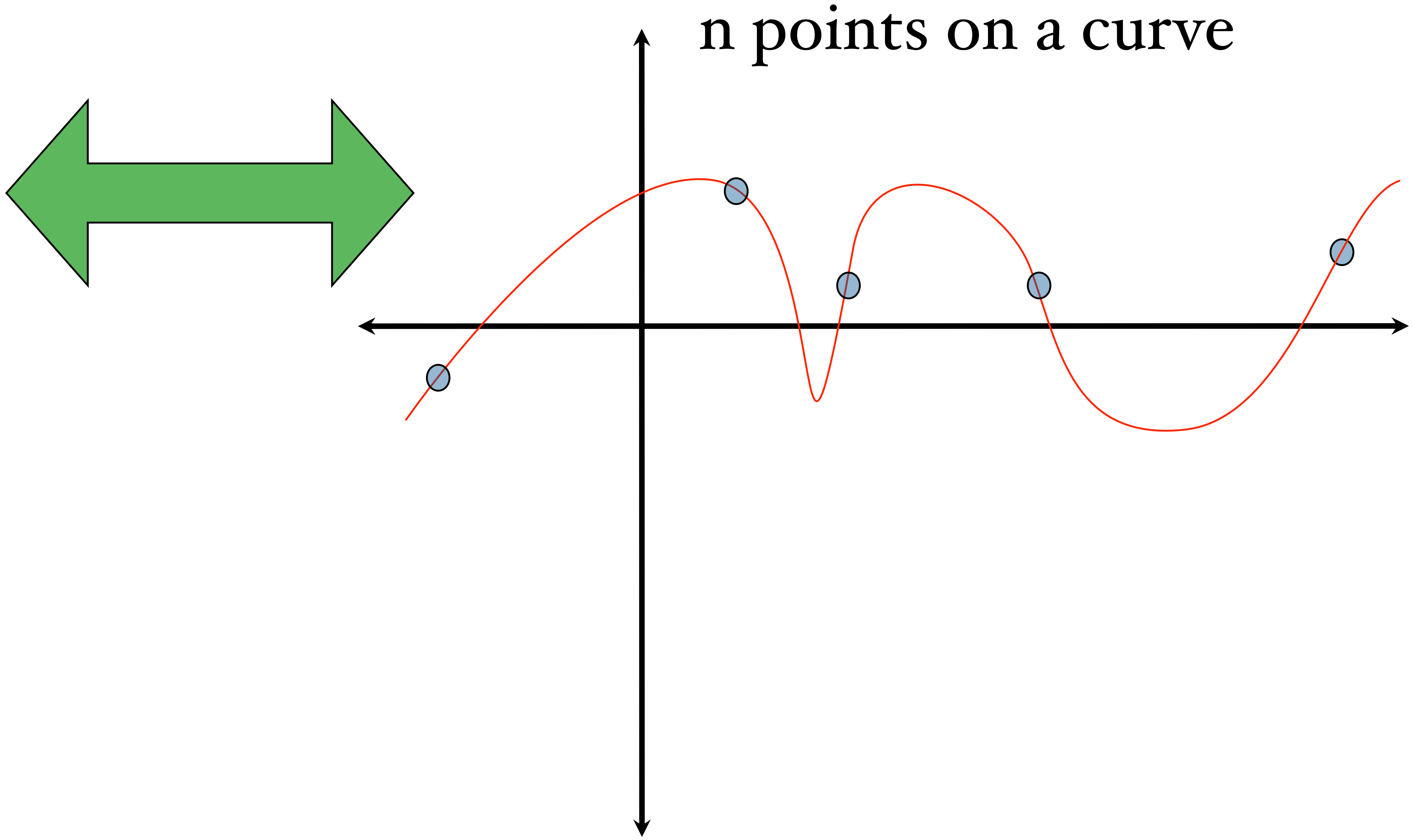
FFT

input: $a_0, a_1, a_2, \ldots, a_{n-1}$

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

output: evaluate polynomial A at (any) n different points.

$A(x)$



Later, we shall see that the same ideas for FFT can be used to implement Inverse-FFT.

Inverse FFT: Given n-points, → coefficient form.

(Same technique)

Later, we shall see that the same ideas for FFT can be used to implement Inverse-FFT.

Inverse FFT: Given n-points,

$$y_0, y_1, \cdots, y_{n-1}$$

find a degree n polynomial A such that

$$y_i = A(\omega_i)$$

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

Brute force method to evaluate A at n points:

solve the large problem by

solving smaller problems
and combining solutions

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

look for a solution that follows this recurrence.

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

$$= a_0 + a_2 x^2 + \cdots a_4 x^4 + \cdots + a_{n-2} x^{n-2}$$ ①

$$a_1 x + a_3 x^3 + \cdots + a_{n-1} x^{n-1}$$

$$A_e(y) = a_0 + a_2 y + a_4 y^2 + a_6 y^3 + \cdots + a_{n-2} y^{n-2/2}$$

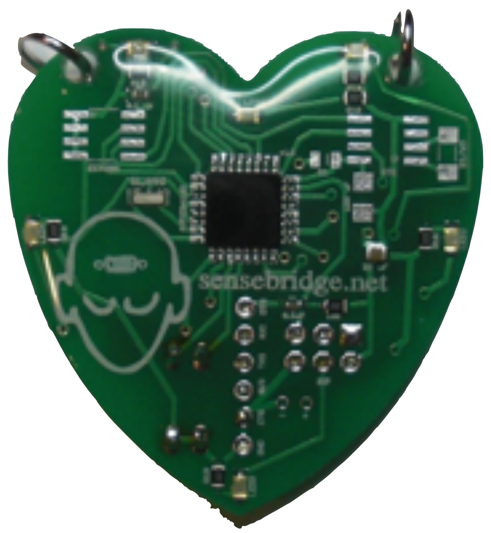
$$A_o(y) = a_1 + a_3 y + a_5 y^2 + \cdots + a_{n-1} y^{n-2/2}$$

both degree $\frac{n-2}{2}$

② polynomials

$\frac{n}{2} - 1$

③ $A(x) = A_e(x^2) + x \cdot A_o(x^2)$



$$
\begin{aligned}
A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \\
= a_0 + a_2 x^2 + a_4 x^4 + \cdots + a_{n-2} x^{n-2} \\
+ a_1 x + a_3 x^3 + a_5 x^5 + \cdots + a_{n-1} x^{n-1}
\end{aligned}
$$

$$
\boxed{
\begin{aligned}
A_e(x) &= a_0 + a_2 x + a_4 x^2 + \cdots + a_n x^{(n-2)/2} \\
A_o(x) &= a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{(n-2)/2}
\end{aligned}
}
$$

$$A(x) = A_e(x^2) + x A_o(x^2)$$

$$A(x) = A_e(x^2) + xA_o(x^2)$$

suppose we had already had eval of Ae,Ao on {4,9,16,25}

$$\boxed{\begin{array}{ll} A_e(4) & A_0(4) \\ A_e(9) & A_0(9) \\ A_e(16) & A_0(16) \\ A_e(25) & A_0(25) \end{array}}$$

$$A(2) = A_e(4) + 2 \cdot A_o(4)$$

$$A(-2) = A_e(4) - 2A_o(4)$$

$$A(x) = A_e(x^2) + xA_o(x^2)$$

suppose we had already had eval of Ae,Ao on {4,9,16,25}

$A_e(4)$ $A_0(4)$
$A_e(9)$ $A_0(9)$
$A_e(16)$ $A_0(16)$
$A_e(25)$ $A_0(25)$

given evaluations of Ae, Ao on 4 points

Then we could compute 8 terms:

$$A(2) = A_e(4) + 2A_o(4)$$

$$A(-2) = A_e(4) + (-2)A_o(4)$$

$$A(3) = A_e(9) + 3A_o(9)$$

$$A(-3) = A_e(9) + (-3)A_o(9)$$

...A(4), A(-4), A(5), A(-5)

FFT(f=a[1,...,n])

Evaluates degree n poly on the n^{th} roots of unity

- $E \leftarrow FFT(A_e)$ // $E[1 \dots n/2]$
- $O \leftarrow FFT(A_o)$ // $O[1 \dots n/2]$

then compute

—> $A(x) = A_e(x^2) \pm x \cdot A_o(x^2)$ for n points

$T(n) = 2T(\frac{n}{2}) + \Theta(n)$

Last remaining issue:

which points are we going to use??

Roots of unity

$$x^n = 1$$

should have n solutions

what are they?

Remember this?

$$e^{2\pi i} = 1$$

Euler

$$x^n = 1$$

$$e^{2\pi i} = 1$$

the n solutions are:

consider $\left\{1, e^{2\pi i/n}, e^{2\pi i2/n}, e^{2\pi i3/n}, \ldots, e^{2\pi i(n-1)/n}\right\}$ n of them

$$\left\{ e^{2\pi i (j/n)} \right\}_{j=0 \ldots n-1}$$

$$\left[e^{2\pi i (j/n)} \right]^n = \left(e^{2\pi i} \right)^j = 1^j = 1$$

$$x^n = 1$$

the n solutions are:

consider $e^{2\pi i j/n}$ for j=0,1,2,3,...,n-1

$$\left[e^{(2\pi i/n)j}\right]^n = \left[e^{(2\pi i/n)n}\right]^j = \left[e^{2\pi i}\right]^j = 1^j$$

$e^{2\pi i(j/n)}$ = $\omega_{j,n}$ is an n^{th} root of unity

$$\omega_{0,n}, \omega_{2,n}, \cdots, \omega_{n-1,n}$$

What is this number?

$e^{2\pi i j/n}$ = $\omega_{j,n}$ is an n[th] root of unity

$$e^{ix} = \cos(x) + i \cdot \sin(x)$$

(Taylor expansion)

Taylor series expansion

of a function f around point a

$$f(y) = f(a) + \frac{f'(a)}{1!}(y-a) + \frac{f''(a)}{2!}(y-a)^2 + \frac{f'''(a)}{3!}(y-a)^2 +$$

$$e^x =$$

around 0

What is this number?

$e^{2\pi ij/n}$ = $\omega_{j,n}$ is an n^{th} root of unity

$$\underline{e}^{ix} = \underline{\cos(x)} + \underline{i\sin(x)}$$

$$\underline{\underline{e^{2\pi ij/n}}} = \underline{\underline{\cos(2\pi j/n)}} + i\sin(2\pi j/n)$$

$e^{2\pi i j/n} = \omega_{j,n}$ is an nth root of unity

$$\omega_{0,n}, \omega_{2,n}, \ldots, \omega_{n-1,n}$$

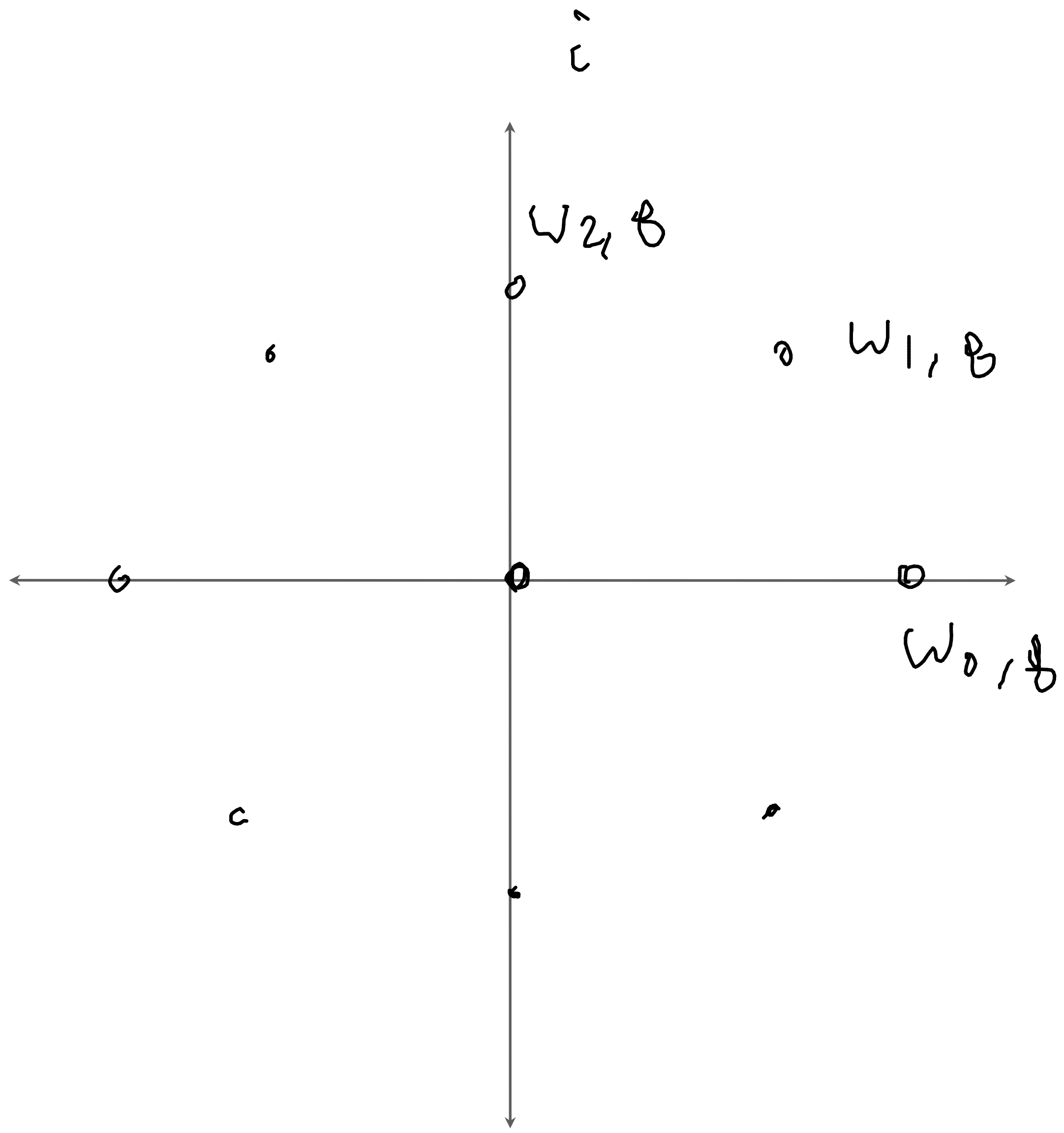
Lets compute $\omega_{1,8}$

$$\omega_{1,8} = \cos\left(2\pi\left(1/8\right)\right) + i\sin\left(2\pi\left(1/8\right)\right)$$

$$= \cos\left(\frac{\pi}{4}\right) + i\sin\left(\pi/4\right) = \boxed{\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}}$$

$$\omega_{0,8} = 1$$

Compute all 8 roots of unity



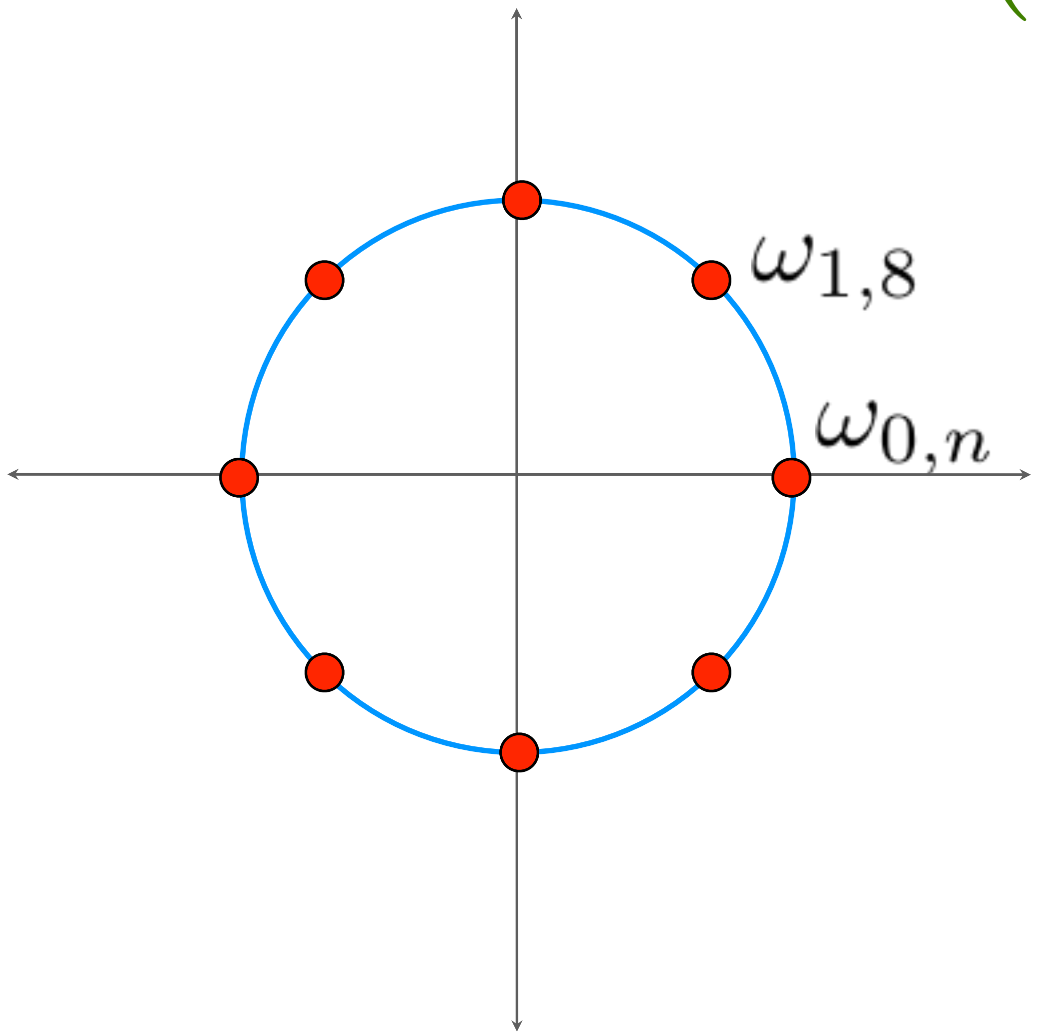
Then graph them

roots of unity

$$x^n = 1$$

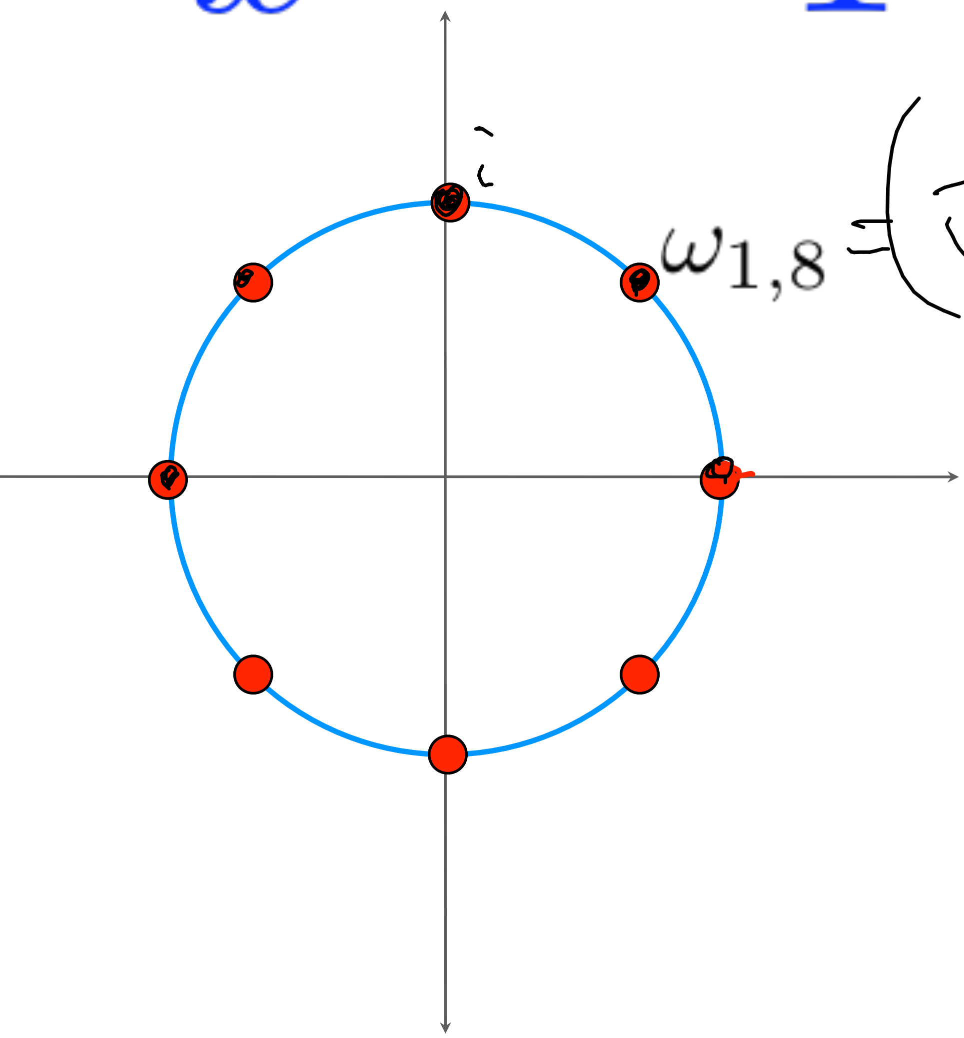
should have n solutions

$$e^{2\pi i j/n} = \cos(2\pi j/n) + i\sin(2\pi j/n)$$



squaring the $\underline{n}^{th}$ roots of unity

$$x^n = 1$$



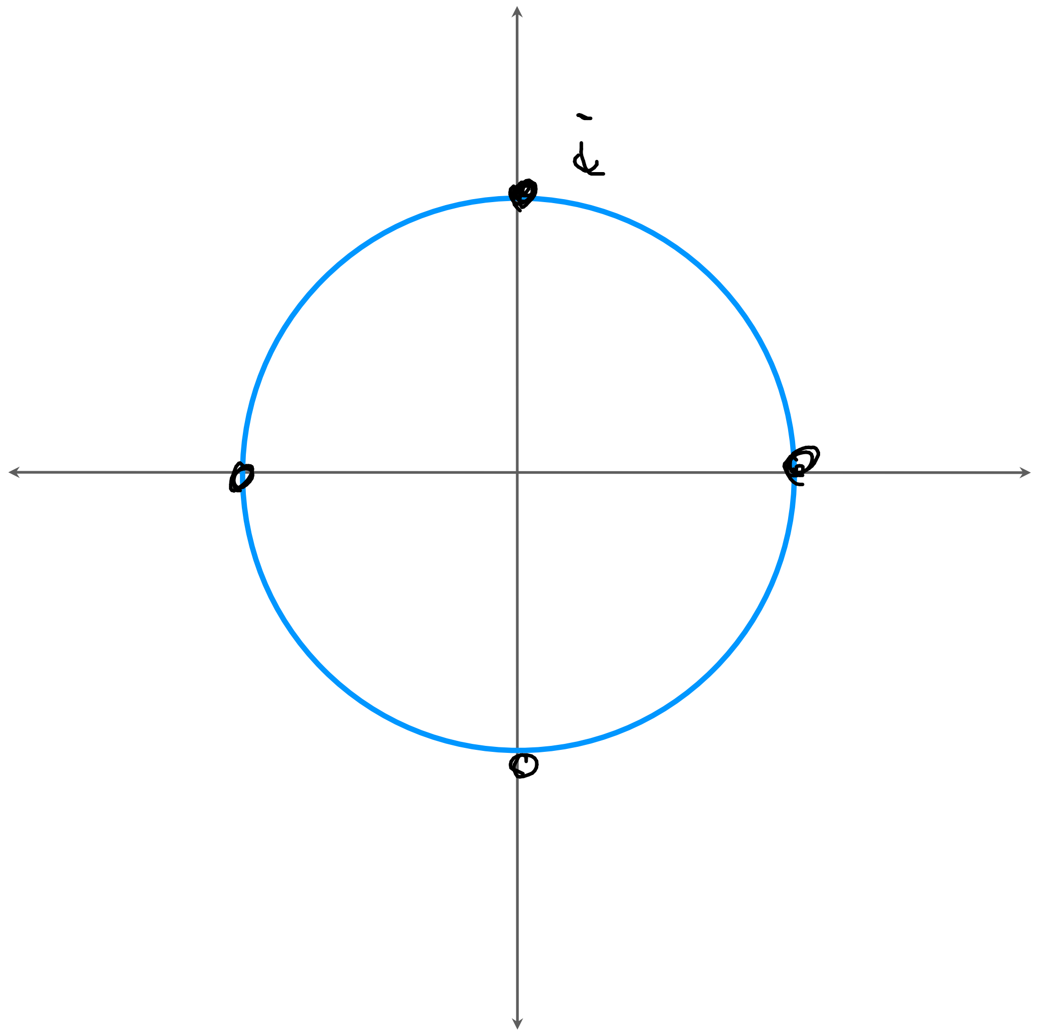
$$\omega_{1,8} = \left(\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right)^2 =$$

$$\frac{1}{2} + \frac{2i}{2} + \frac{i^2}{2}$$

$$\frac{1}{2} + i - \frac{1}{2}$$

$$\frac{i}{2} = i$$

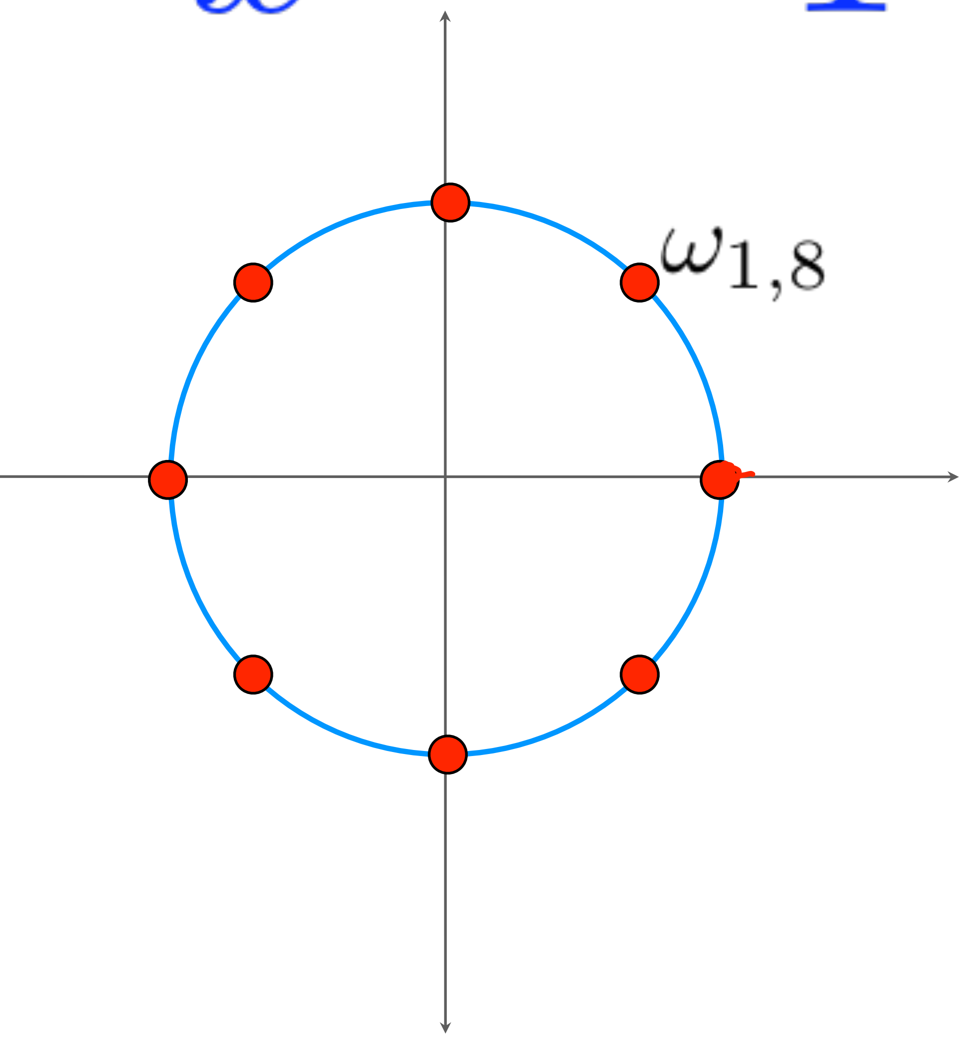
8th roots of unity



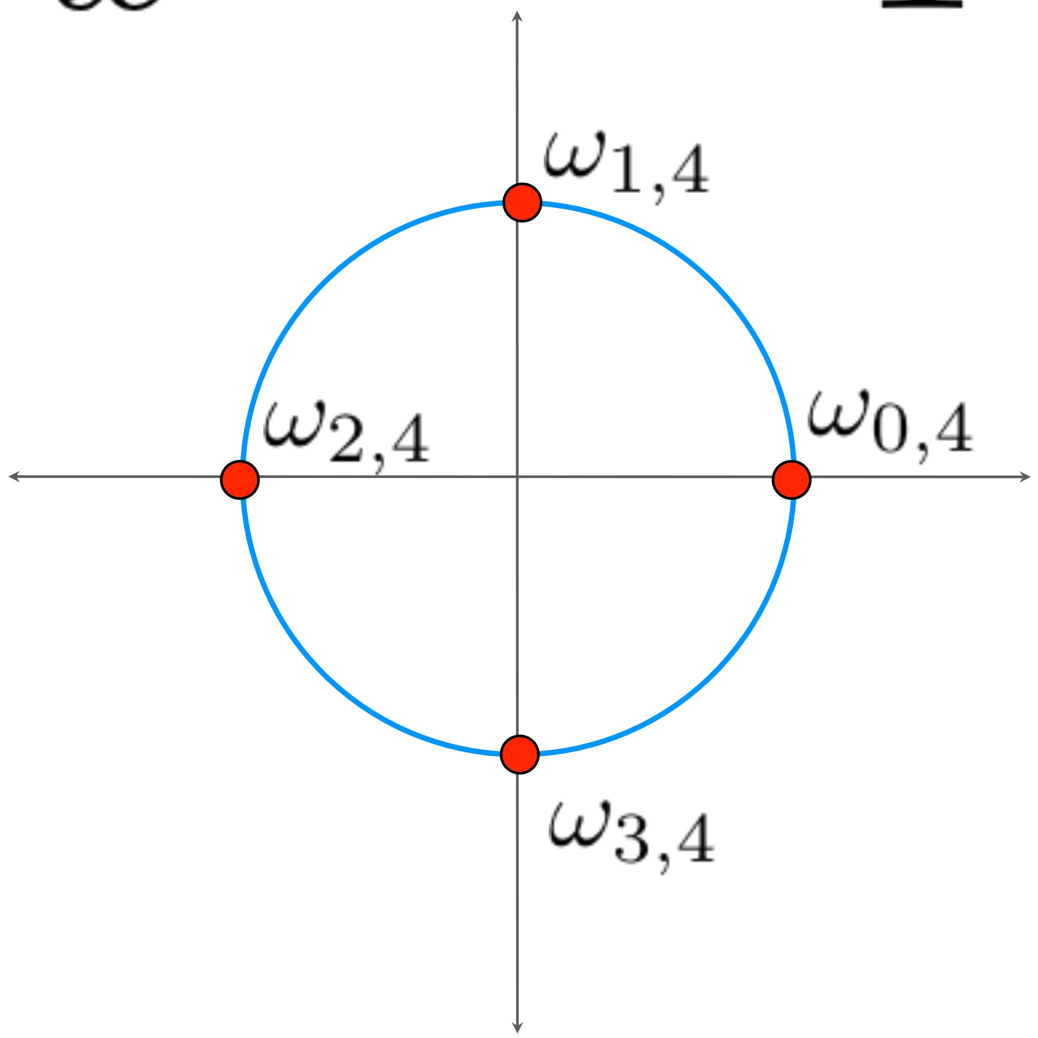
n/2 th roots of unity

squaring the nth roots of unity

$$x^n = 1$$



$$x^{n/2} = 1$$



Thm: Squaring an nth root produces an n/2th root.

example: $$\omega_{1,8} = \left(\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right)$$

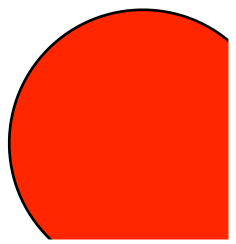
$$\omega_{1,8}^2 = \left(\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right)^2 = \left(\frac{1}{\sqrt{2}}\right)^2 + 2\left(\frac{1}{\sqrt{2}}\frac{i}{\sqrt{2}}\right) + \left(\frac{i}{\sqrt{2}}\right)^2$$

$$= 1/2 + i - 1/2$$

$$= i$$

Thm: Squaring an n^{th} root produces an $n/2^{th}$ root.

$$\left\{1, e^{2\pi i(1/n)}, e^{2\pi i(2/n)}, e^{2\pi i(3/n)}, \ldots, e^{2\pi i(n/2)/n}, e^{2\pi i(n/2+1)/n}, \ldots, e^{2\pi i(n-1)/n}\right\}$$



$$A(x) = A_e(x^2) + xA_o(x^2)$$

evaluate at a root of unity

$$A(x) = A_e(x^2) + xA_o(x^2)$$

evaluate at a root of unity

$$A(\omega_{i,n}) = A_e(\omega_{i,n}^2) + \omega_{i,n}A_o(\omega_{i,n}^2)$$

n^{th} root of unity

$n/2^{th}$ root of unity

$n/2^{th}$ root of unity

$FFT(f = a\{1, \ldots, n\})$

Evaluates degree n poly on the n^{th} roots of unity

FFT(f=a[1,...,n])

Base case if n<=2

E[...] <- FFT(A_e) // eval Ae on n/2 roots of unity

O[...] <- FFT(A_o) // eval Ao on n/2 roots of unity

combine results using equation:

$$A(\omega_{i,n}) = A_e(\omega_{i,n}^2) + \omega_{i,n} A_o(\omega_{i,n}^2)$$

$$A(\omega_{i,n}) = A_e(\omega_{i \bmod n/2, \frac{n}{2}}) + \omega_{i,n} A_o(\omega_{i \bmod n/2, \frac{n}{2}})$$

Return n points.

application to mult

karatsuba

1	7	8	9	★	1	4	3	2
a		b			c		d	

$$\Theta(n^{\log_2 3})$$

application to mult

karatsuba

1 7 8 9 ★ 1 4 3 2

a b c d

$$T(n) = 3T(n/2) + 6O(n)$$

$$\Theta(n^{\log_2 3})$$

a_3 a_2 a_1 a_0 $\bigstar$ b_3 b_2 b_1 b_0

a_3 a_2 a_1 a_0 ★ b_3 b_2 b_1 b_0

$$A(x) = a_0 + a_1 x + a2x^2 + a_3 x^3 + 0x^4 + \cdots + 0x^7$$

$$B(x) = b_0 + b_1 x + b2x^2 + b_3 x^3 + 0x^4 + \cdots + 0x^7$$

$A(\omega_0)$ $A(\omega_1)$ $A(\omega_2)$ .... $A(\omega_7)$

$B(\omega_0)$ $B(\omega_1)$ $B(\omega_2)$ .... $B(\omega_7)$

$C(\omega_0)$ $C(\omega_1)$ $C(\omega_2)$ .... $C(\omega_7)$

$$C(x) = c_0 + c_1 x + c_2 x^2 + \cdots c_7 x^7$$

a_3 a_2 a_1 a_0 ★ b_3 b_2 b_1 b_0

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + 0x^7$$

$$B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + \cdots + 0x^7$$

$A(\omega_1)$	$B(\omega_1)$	$C(\omega_1)$
$A(\omega_8)$	$B(\omega_8)$	$C(\omega_8)$

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

Multiplying n-bit numbers

A GMP-BASED IMPLEMENTATION OF SCHÖNHAGE-STRASSEN'S LARGE INTEGER MULTIPLICATION ALGORITHM

PIERRICK GAUDRY, ALEXANDER KRUPPA, AND PAUL ZIMMERMANN

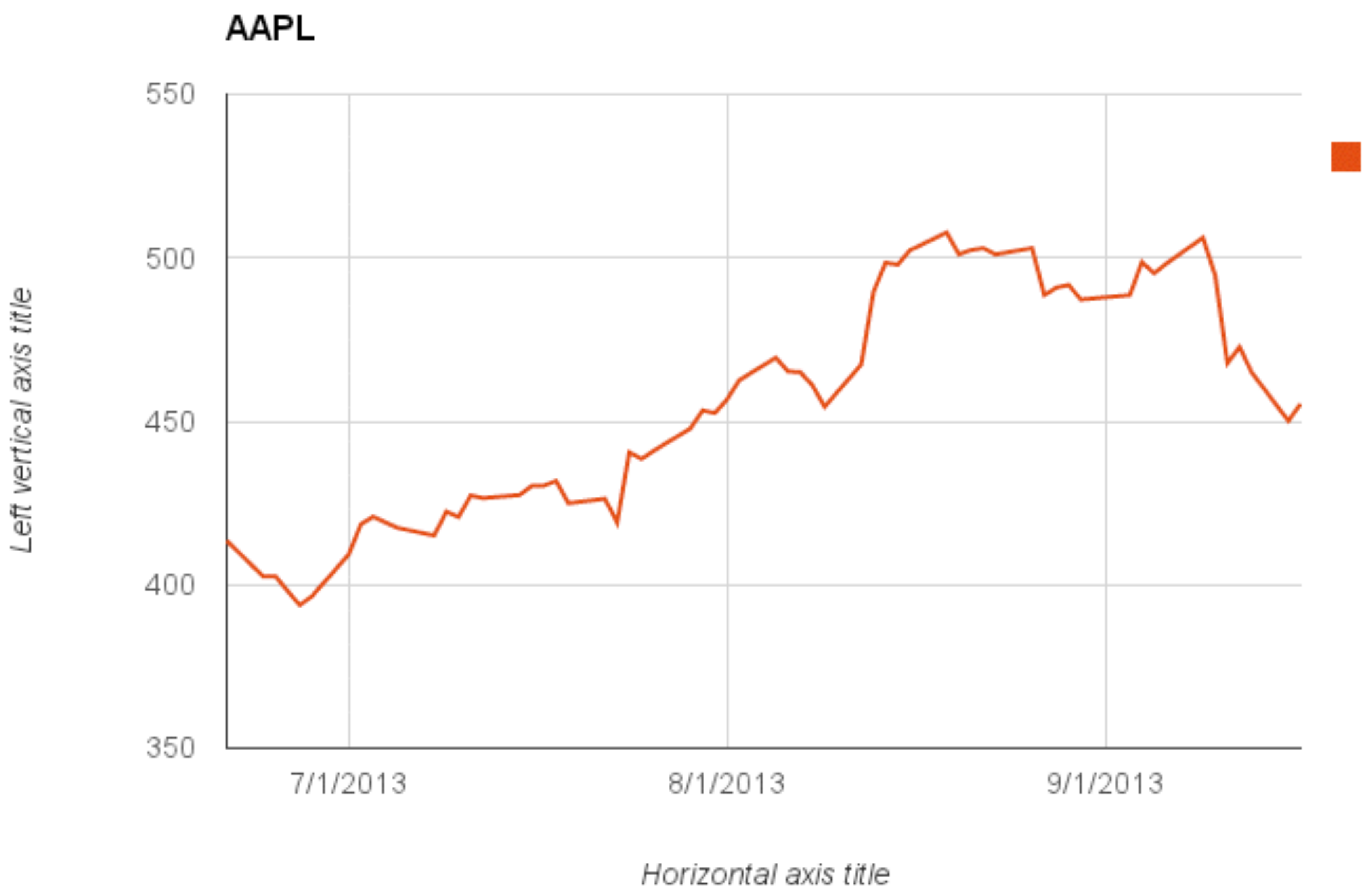
Abstract. Schönhage-Strassen's algorithm is one of the best known algorithms for multiplying large integers. Implementing it efficiently is of utmost importance, since many other algorithms rely on it as a subroutine. We present here an improved implementation, based on the one distributed within the GMP library. The following ideas and techniques were used or tried: faster arithmetic modulo $2^n + 1$, improved cache locality, Mersenne transforms, Chinese Remainder Reconstruction, the $\sqrt{2}$ trick, Harley's and Granlund's tricks, improved tuning. We also discuss some ideas we plan to try in the future.

Introduction

Since Schönhage and Strassen have shown in 1971 how to multiply two N-bit integers in $O(N \log N \log\log N)$ time [21], several authors showed how to reduce other operations — inverse, division, square root, gcd, base conversion, elementary functions — to multiplication, possibly with $\log N$ multiplicative factors [5, 8, 17, 18, 20, 23]. It has now become common practice to express complexities in terms of the cost $M(N)$ to multiply two N-bit numbers, and many researchers tried hard to get the best possible constants in front of $M(N)$ for the above-mentioned operations (see for example [6, 16]).

Strangely, much less effort was made for decreasing the implicit constant in $M(N)$ itself, although any gain on that constant will give a similar gain on all multiplication-based operations. Some authors reported on implementations of large integer arithmetic for specific hardware or as part of a number-theoretic project [2, 10]. In this article we concentrate on the question of an optimized implementation of Schönhage-Strassen's algorithm on a classical workstation.

Applications of FFT



Applications of FFT

String matching with *

ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCC
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC
CTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGG
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCC
CTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAG
TTTAATTACAGACCTGAA

Looking for all occurrences of

GGC*GAG*C*GC

where I don't care what the * symbol is.