L8

SEP 19 2013

abhi shelat

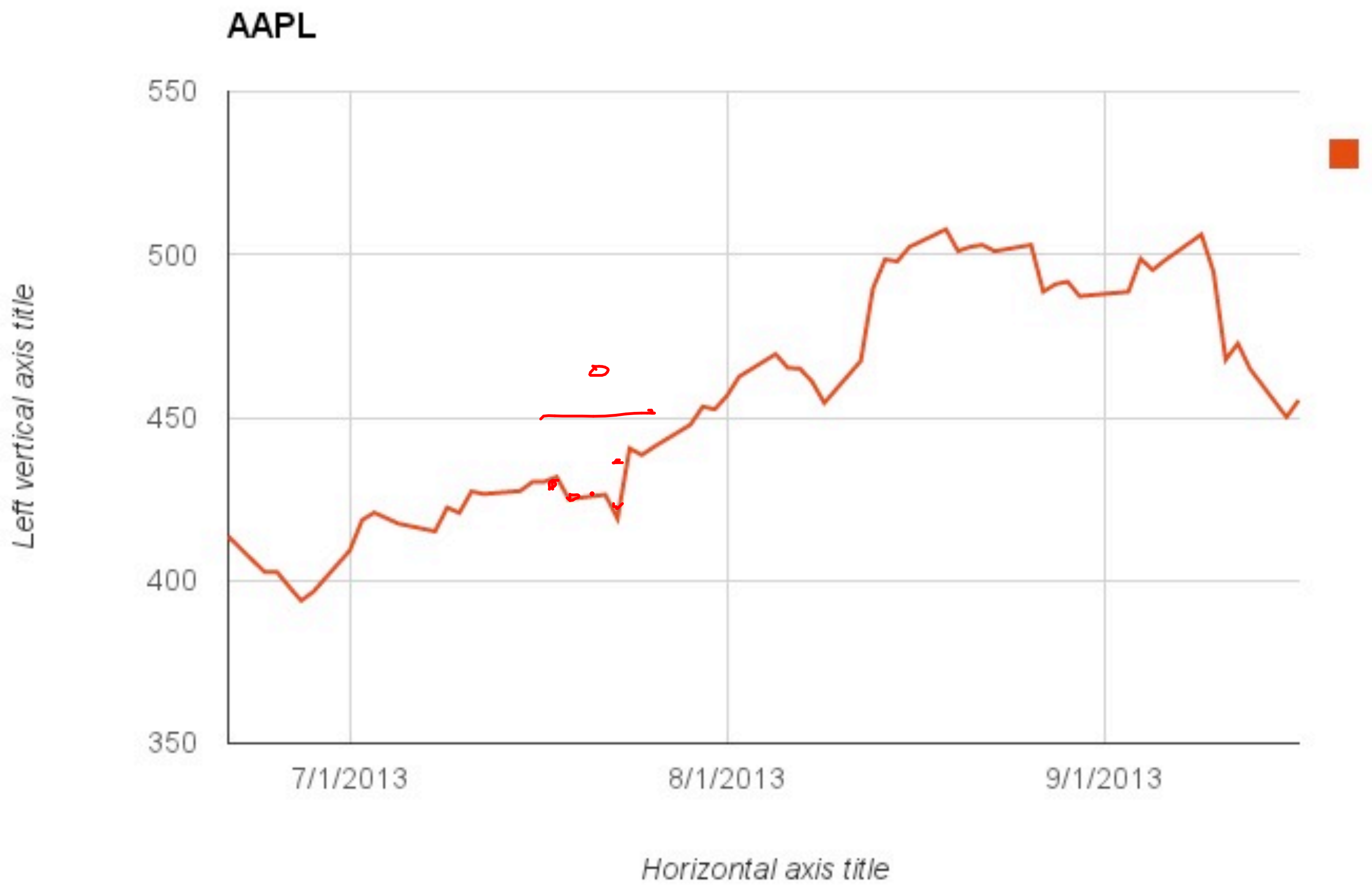
FFT, INTRO TO DP

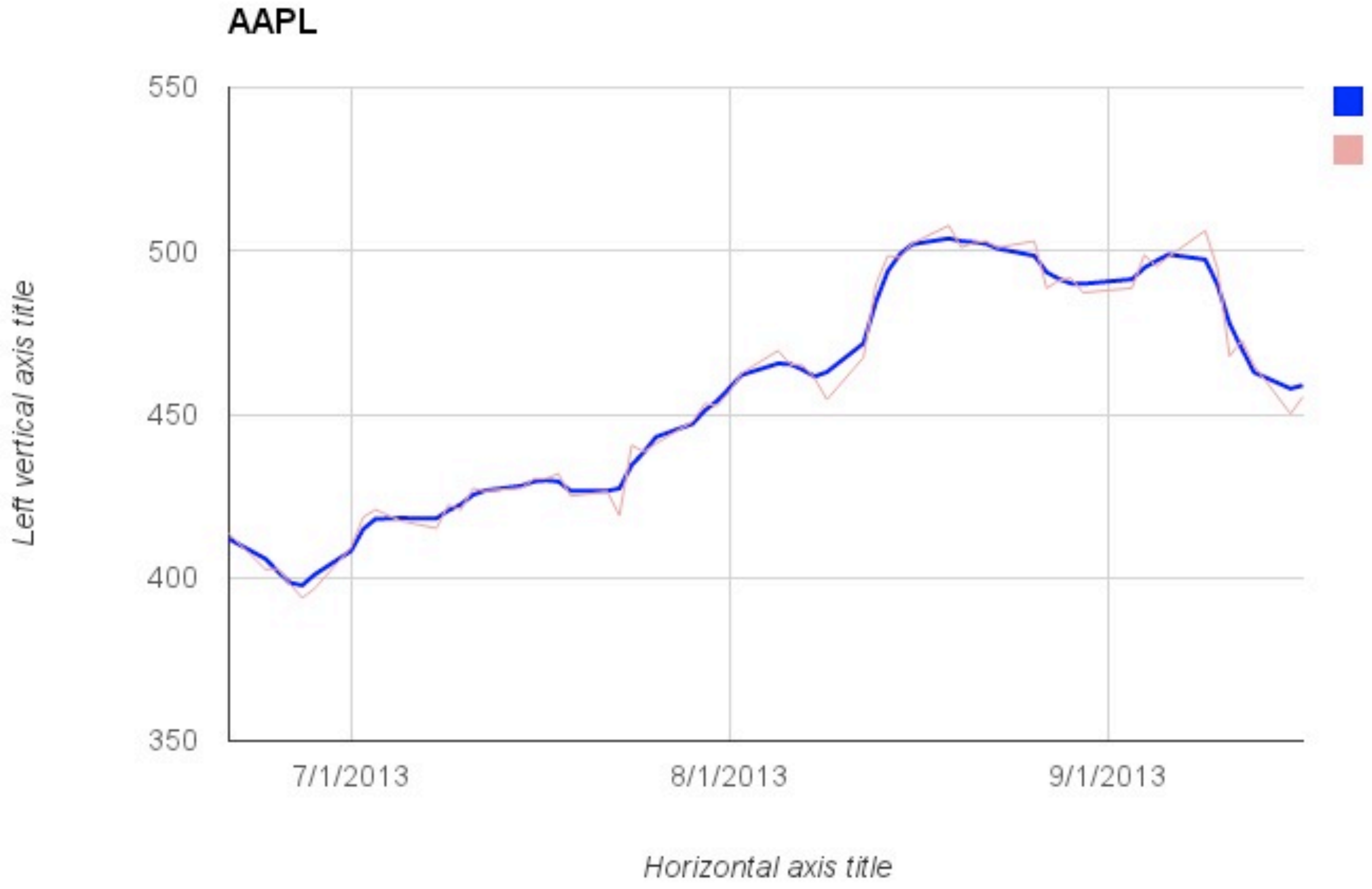
HW2: extension to SUNDAY NOON.

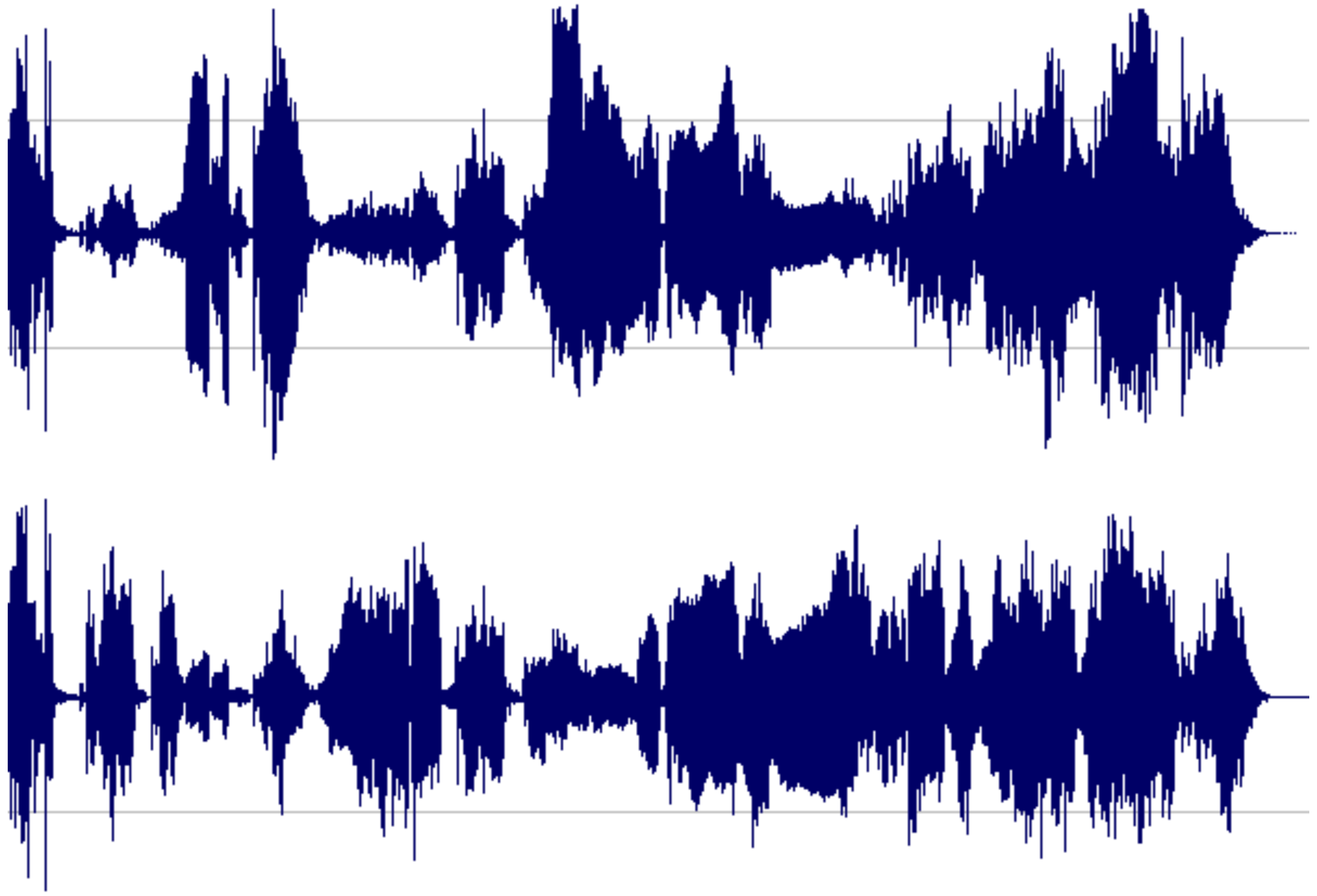
Fast Fourier Transform

© Jim Hatch Illustration / www.khulsey.com

- average every 5 data points into a new data point




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



big ideas:

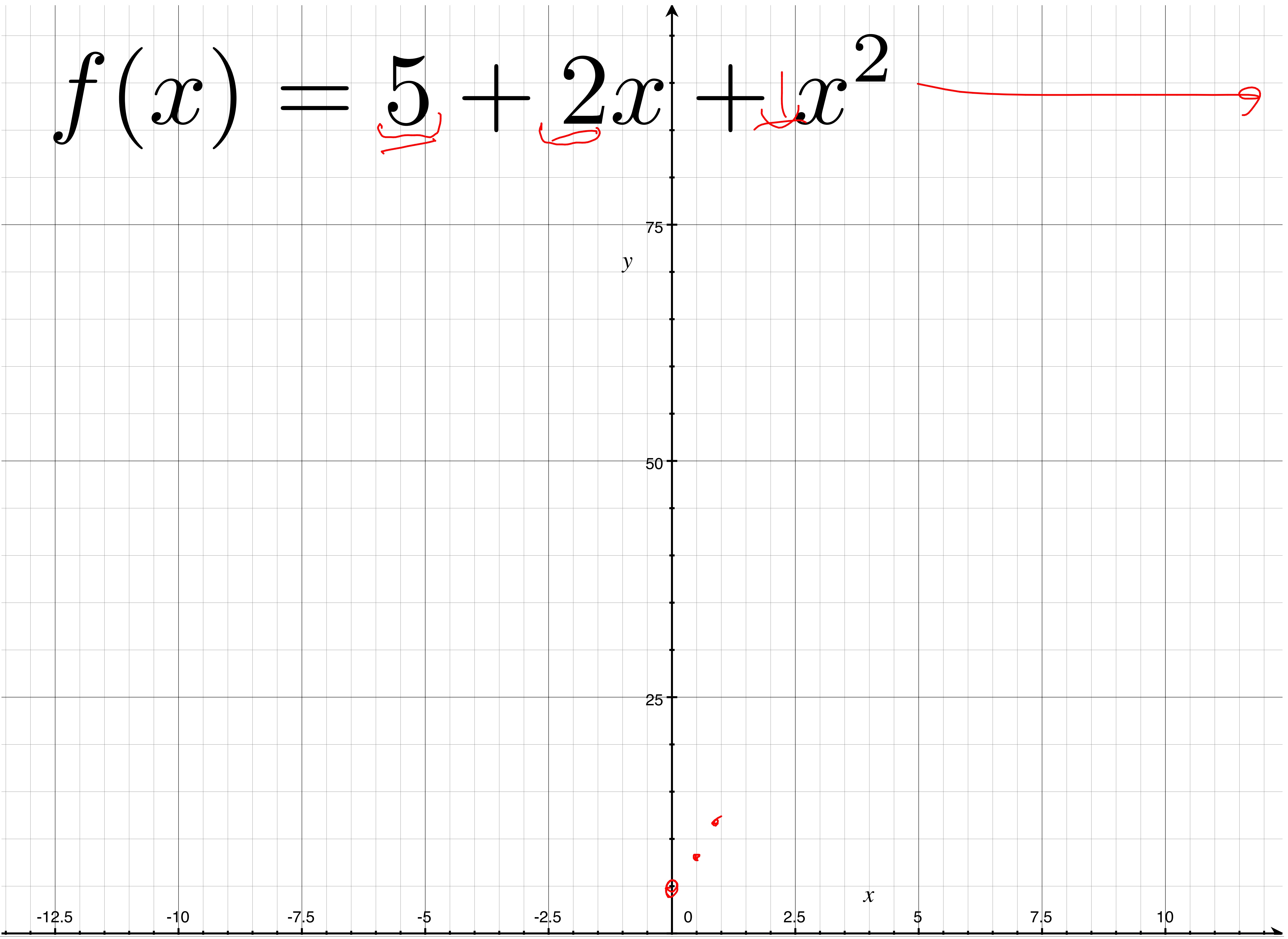
change representations of mathematical objects

-> some representations have efficiency benefits over others

$$f(x) = \underline{5} + \underline{2}x + \underline{1}x^2 \longrightarrow$$

polynomial. degree 2,
then 3 numbers represent
this polynomial

n-1 degree polynomial → n coefficients

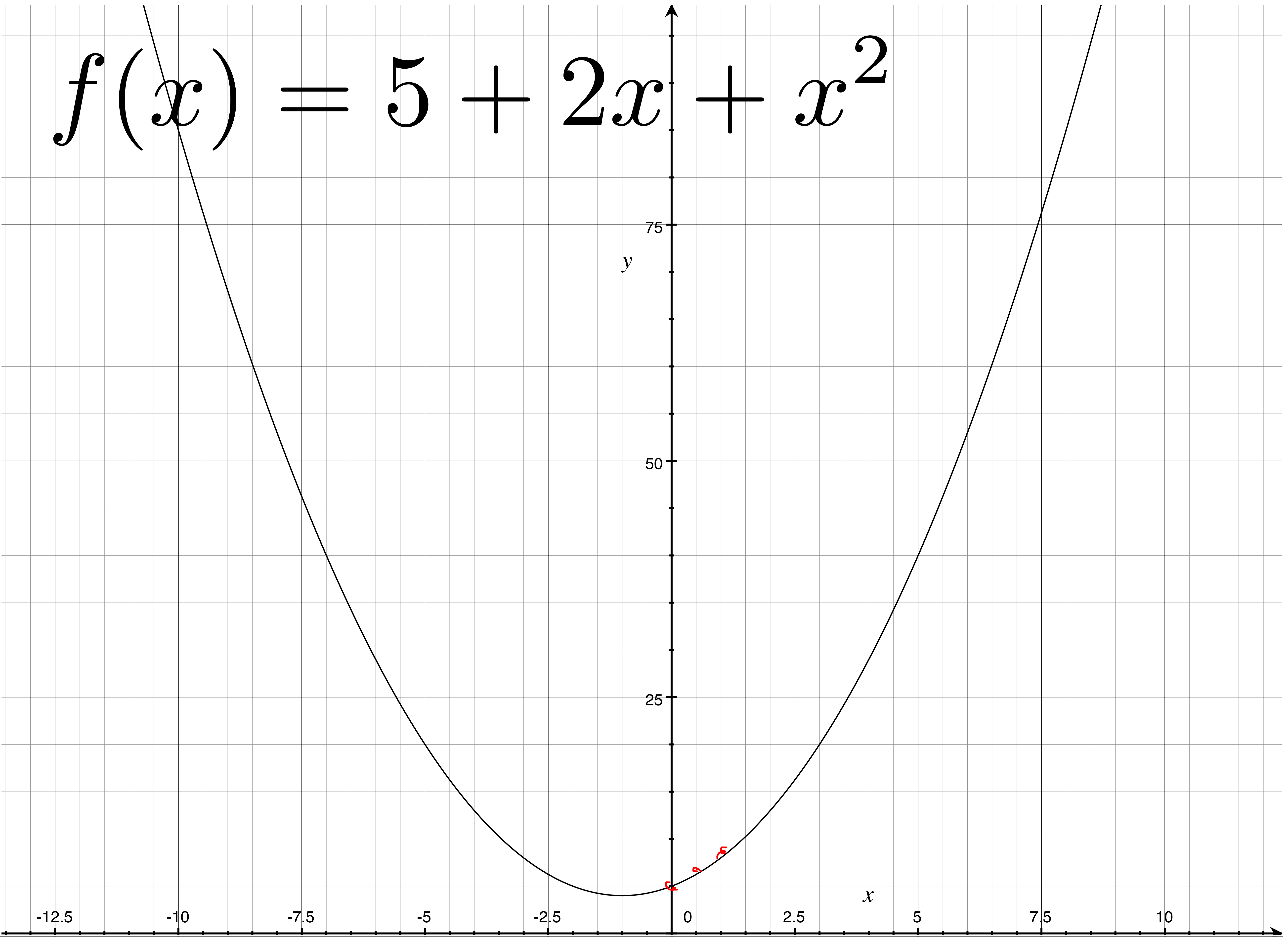


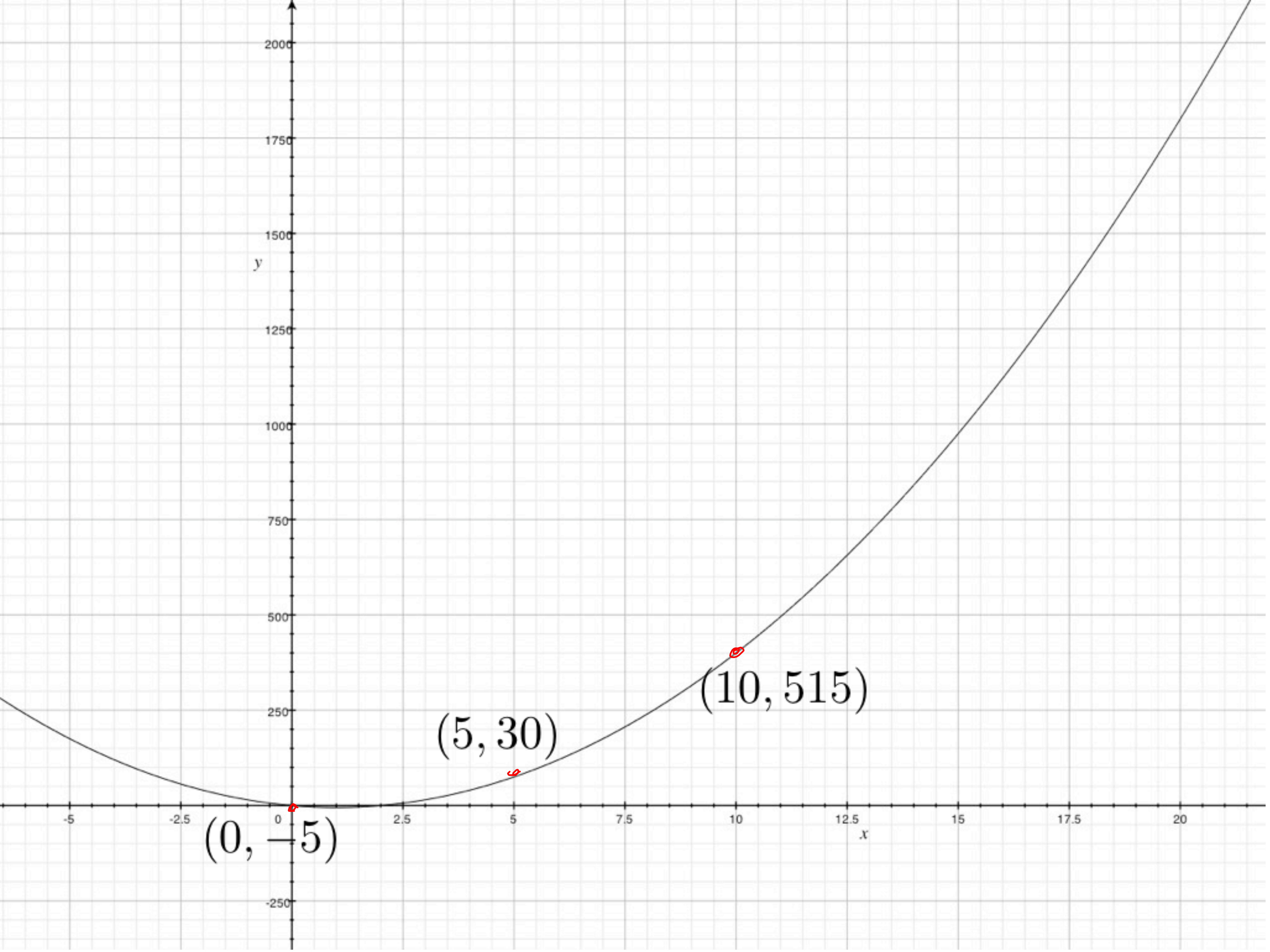
$x=0, \quad f(x)=5$

$x=1 \quad f(1) = 8$

$f(2) = 13$

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





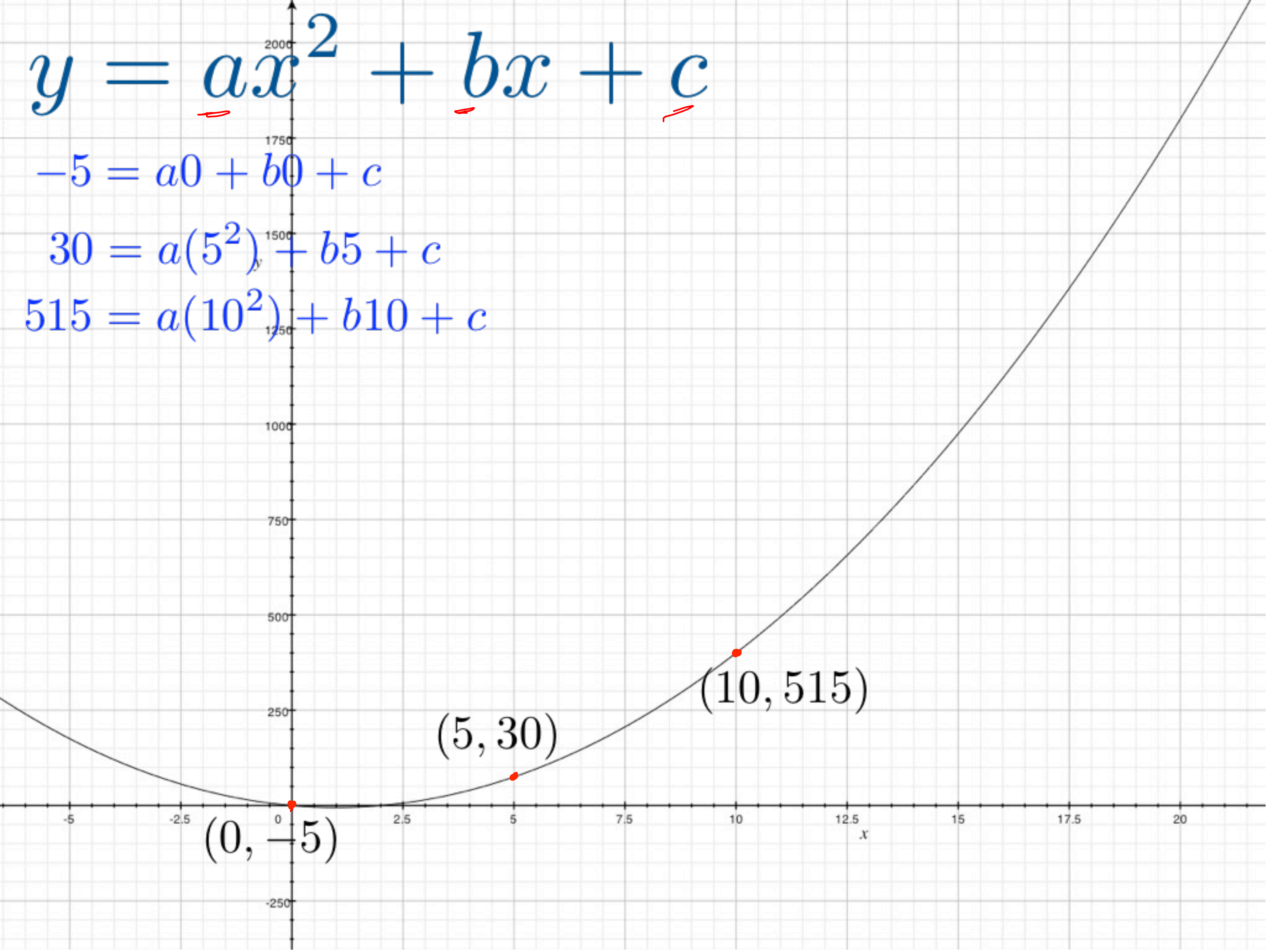
if I give you n points, do they define an $(n-1)$-degree polynomial??

$$y = \underline{a}x^2 + \underline{b}x + \underline{c}$$

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

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

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



$f(0) = -5 \Rightarrow$

$f(0) = -5$, then

$$\left.\begin{array}{l} -5 = \underline{a}\cdot 0 + \underline{b}\cdot 0 + \underline{c} \\ 30 = a(5)^2 + b(5) + c \\ 515 = a(10)^2 + b(10) + c \end{array}\right]$$

Solve linear system for a, b, c.

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$ ← coefficient form of the polynomial

$A(0)$ $A(1)$ ... $A(n-1)$ for n distinct points

↰ point-wise representation of a polynomial.

coeff.

degree $n-1$ polynomial

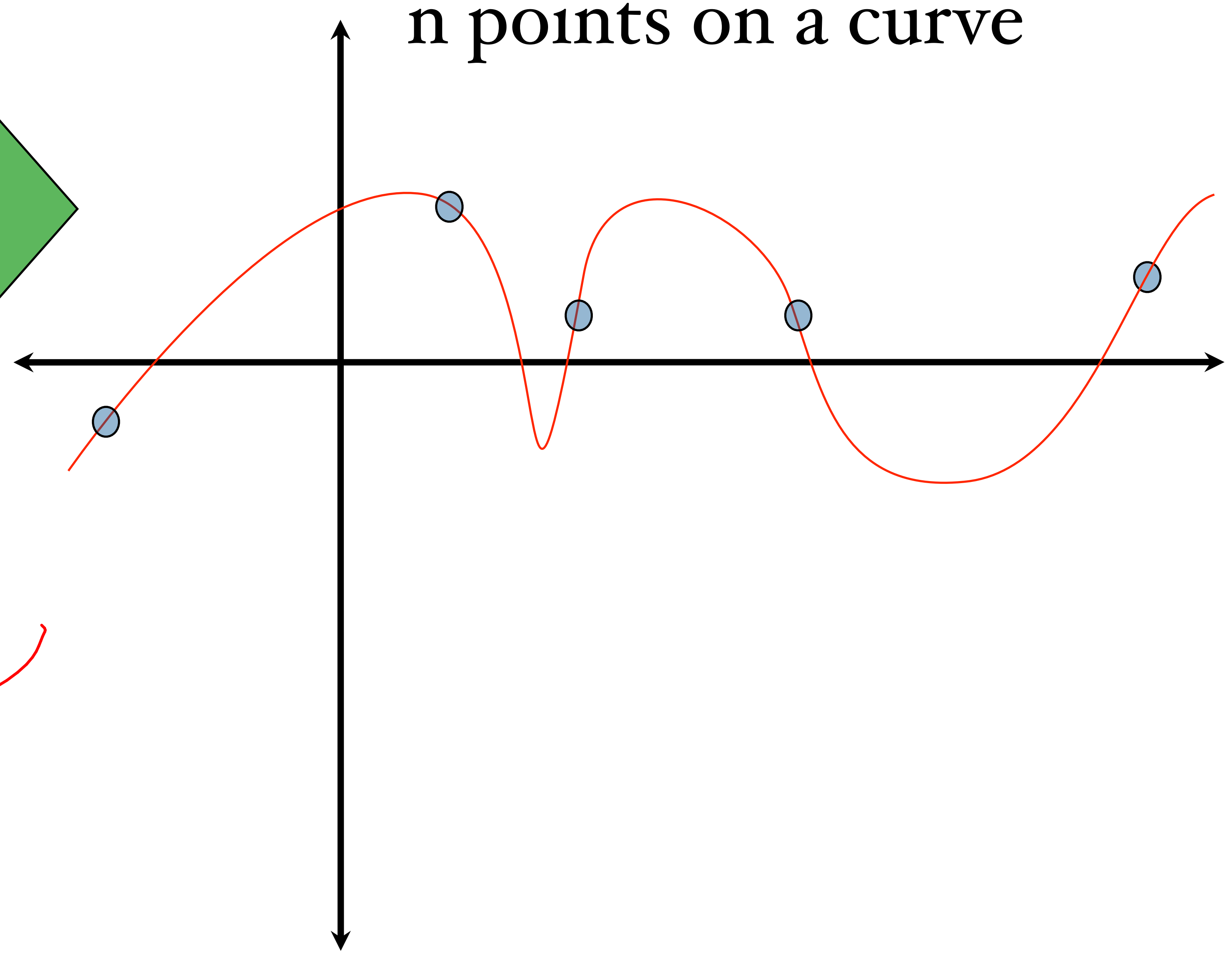
$$A(x)$$

FFT

IFFT

point-wise form

n points on a curve



FFT

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

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

output: point-wise representation

$A(w_0)$ $A(w_1)$ . . . $A(w_n)$ where $w_1 \ldots w_n$ are distinct points (specially chosen)

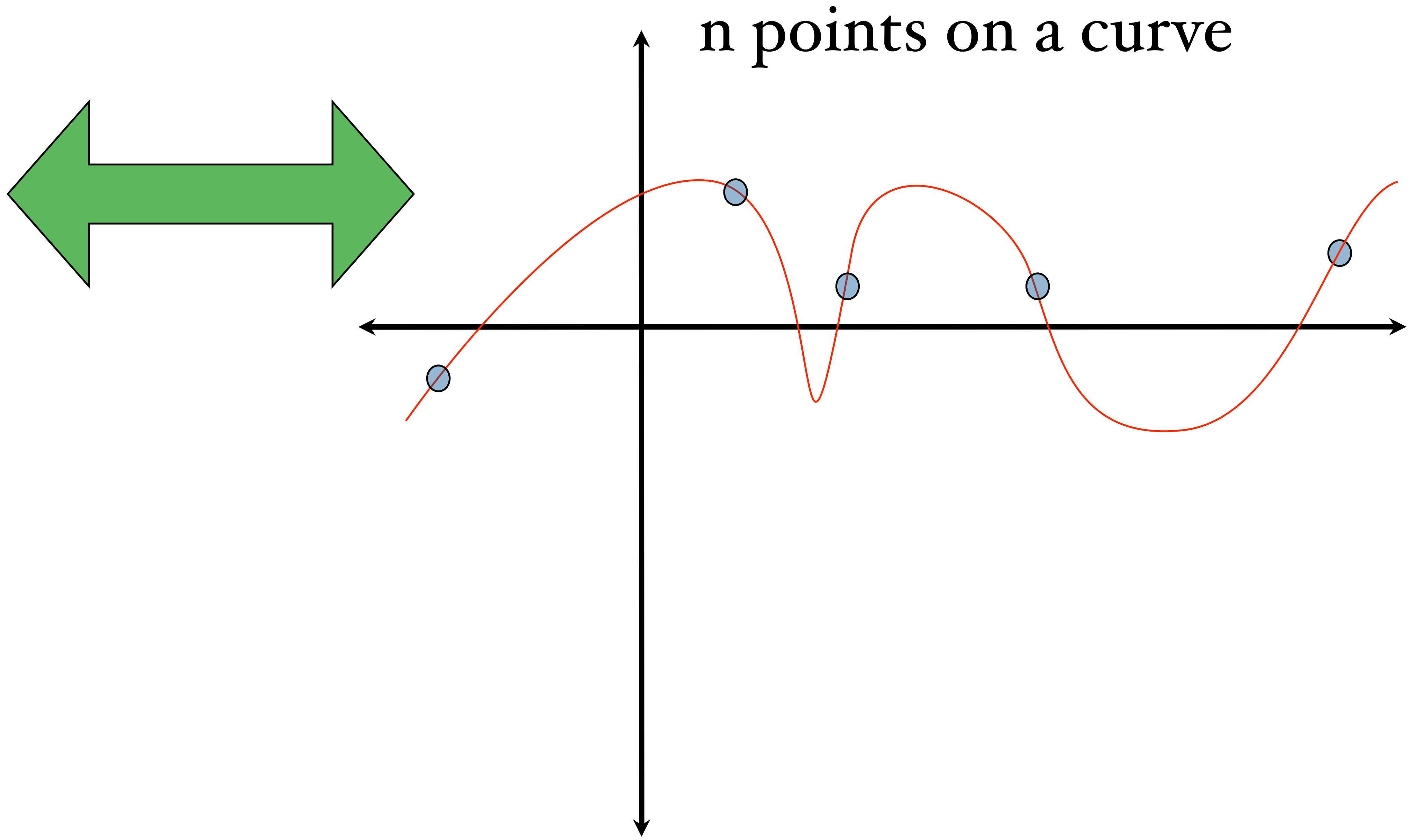
FFT

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,

$y_0 \quad y_1 \quad \dots \quad y_{n-1}$

produce the n- coefficients for the polynomial A s.t.

$$A(w_i) = y_i.$$

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

$$y_i = A(\omega_i)$$

$$\underline{A(x)} = \underline{a_0} + \underline{a_1}x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$$

Brute force method to evaluate A at n points:

$A(w_0) = \square$ will take $\underline{\Theta(n)}$ operations

$A(w_0) .. A(w_{n-1})$ takes $\Theta(n^2)$ operations.

polynomial interpolations (brute force) $\underline{\Theta(n^3)}$

or

$\underline{\Theta(n^2)}$

we seek an

$\underline{\Theta(n \log n)}$

solve the large problem by

solving smaller problems
and combining solutions

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

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

$\boxed{n - \text{power of } 2}$

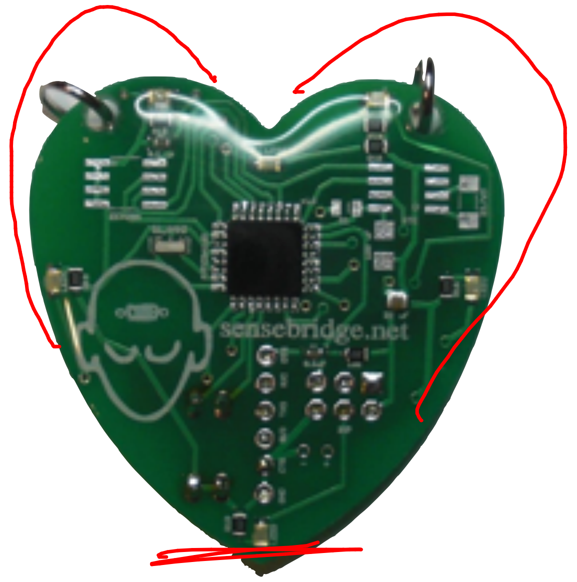
$$= \underline{a_0} + \underline{a_2 x^2} + \underline{a_4 x^4} + a_{n-2} x^{n-2}$$

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

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

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

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



$$\boxed{A(x) = A_e(x^2) + \underline{x} \cdot A_o(x^2)} \leftarrow$$

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

$$= a_0 + a_2x^2 + a_4x^4 + \cdots + a_{n-2}x^{n-2}$$

$$+a_1x + a_3x^3 + a_5x^5 + \cdots + a_{n-1}x^{n-1}$$

$$A_e(x) = a_0 + a_2x + a_4x^2 + \cdots + a_nx^{(n-2)/2}$$

$$A_o(x) = a_1 + a_3x + a_5x^2 + \cdots + a_{n-1}x^{(n-2)/2}$$

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

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

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

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

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

4 points

$\downarrow$

$\Theta(n)$ time

eval of A on 8 points

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

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

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

$\vdots$

$\vdots$

$A(5) =$ ______

$A(-5) =$ ______

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

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

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

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

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

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

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

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

Evaluates degree n poly on the nth roots of unity

```
E[ ] ← FFT($A_e$)                    $T(n/2)$
O[ ] ← FFT($A_o$)                    $T(n/2)$

return $A[w_0 \dots w_{n-1}]$ using the equation
    $A(\underline{w_i}) = A_e(\underline{w_i}^2) + \underline{w_1} \cdot A_o(\underline{w_i}^2)$   ← $O(n)$
```

Last remaining issue:

What points w_i do we use??

n^th -

roots of unity

$$x^n = 1$$

should have n solutions

what are they?

Remember this?

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

Euler's identity

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

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

① $$\left[\underbrace{e^{2\pi i j/n}}_{\text{this is a } n^{th} \text{ root of unity.}}\right]^n = \left[e^{(2\pi i/n)\cdot j}\right]^n = \left[\underline{e^{2\pi i}}\right]^j = 1^j = \underline{1}$$

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

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

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

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

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

What is this number?

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

from calculus
Taylor series
etc

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

What is this number?

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

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

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

Why is this true?

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

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

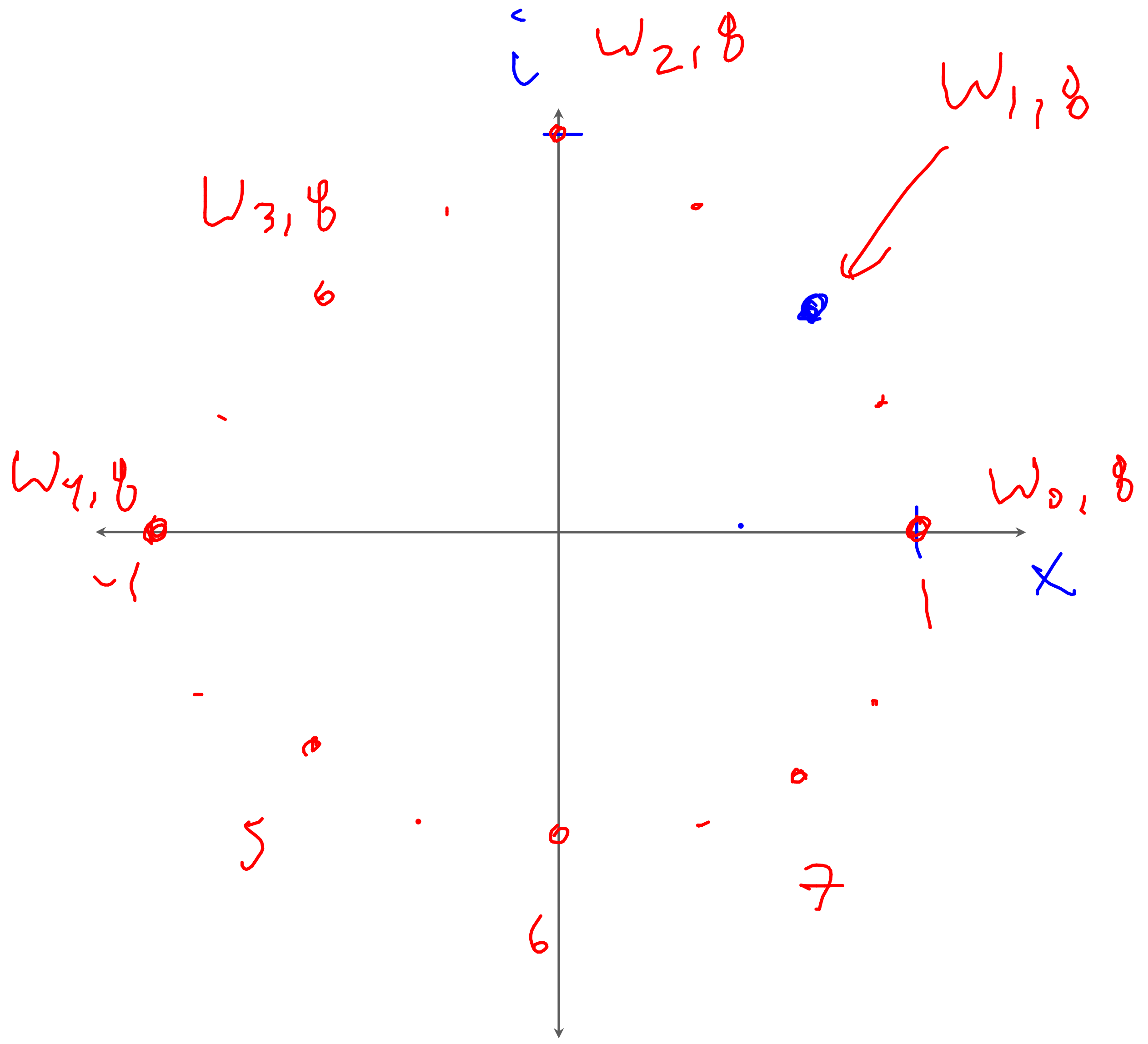
$$\omega_{0,n}, \omega_{2,n}, \dots, \omega_{n-1,n}$$

Lets compute $\omega_{\underset{j}{1},\underset{n}{8}} = e^{2\pi i(1/8)} = \cos\left(2\pi/8\right) + i\cdot\sin\left(2\pi/8\right)$

$$= \frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}$$

$$\frac{\sqrt{2}}{2} = \frac{1}{\sqrt{2}}$$

Compute all 8 roots of unity



$W_{0,8} =$

$W_{1,8} = \frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}$

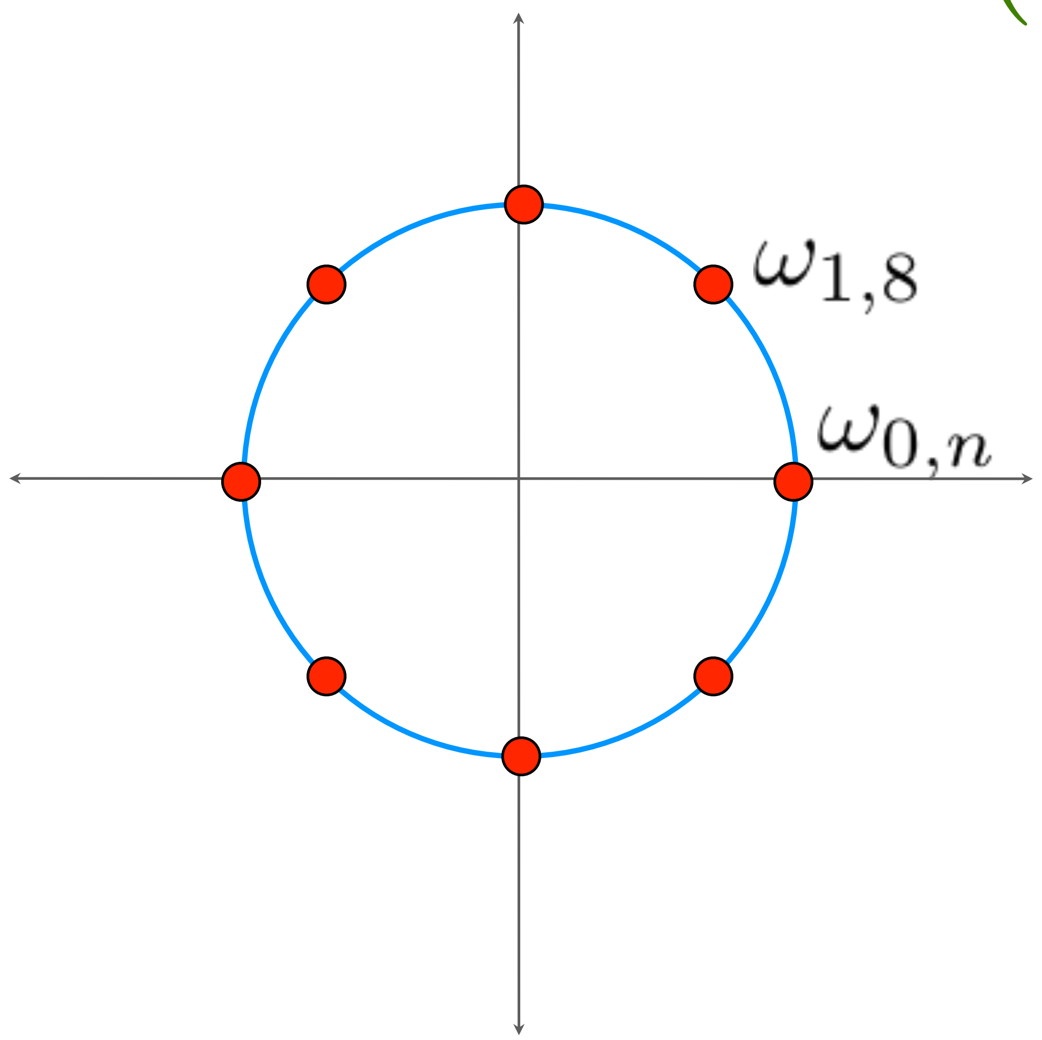
Then graph them

roots of unity

$$x^n = 1$$

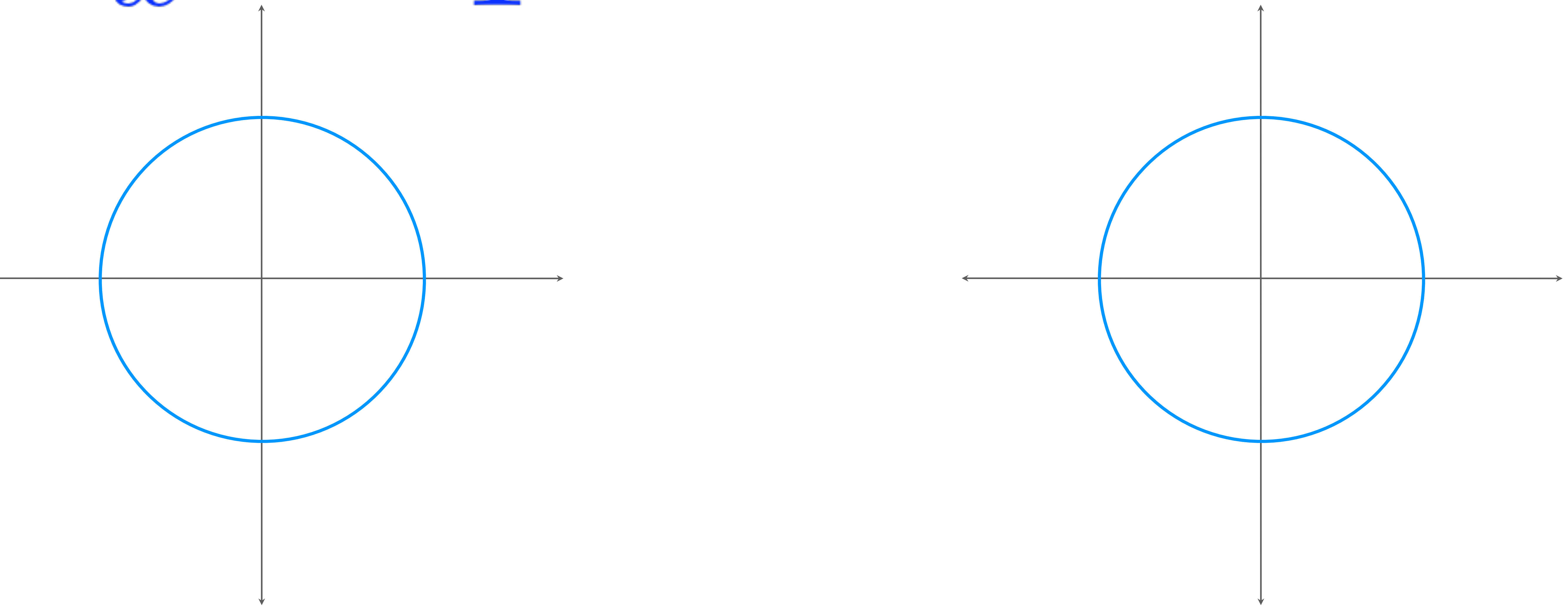
should have n solutions

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



squaring the nth roots of unity

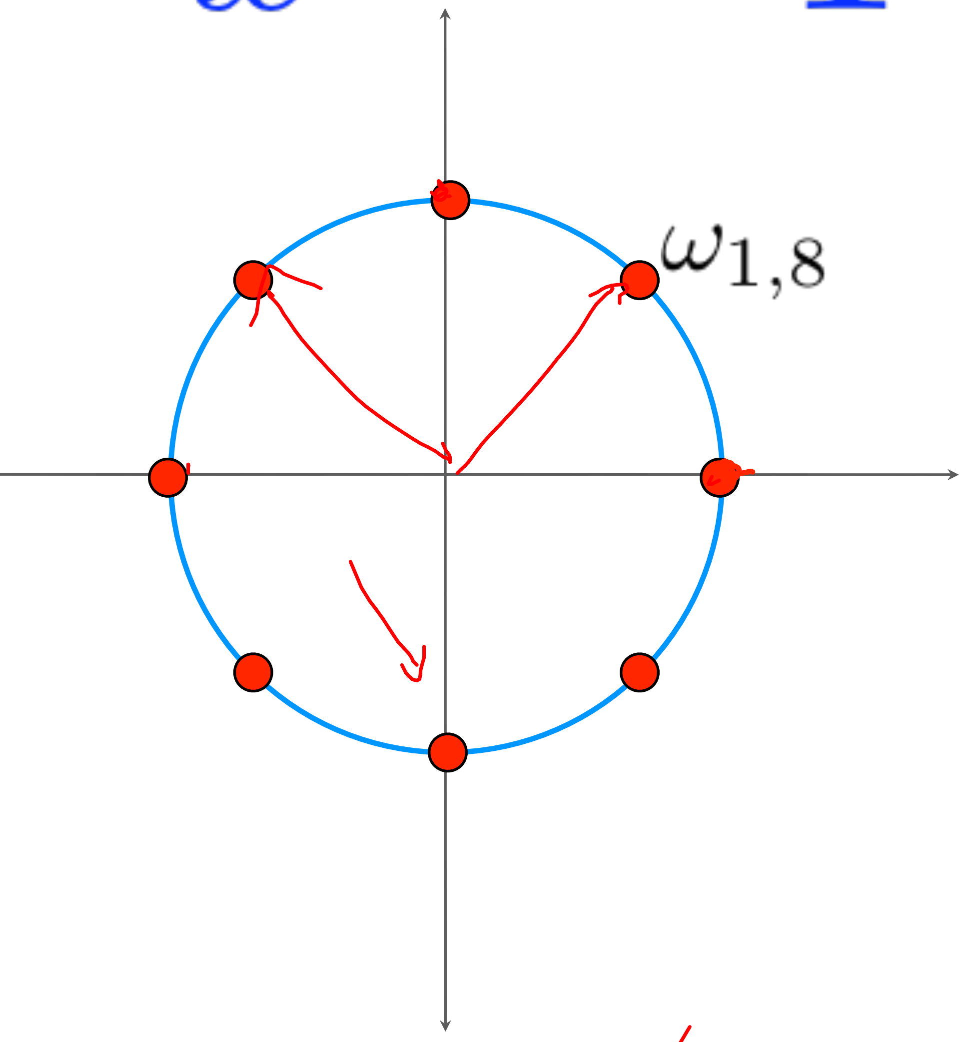
$$x^n = 1$$

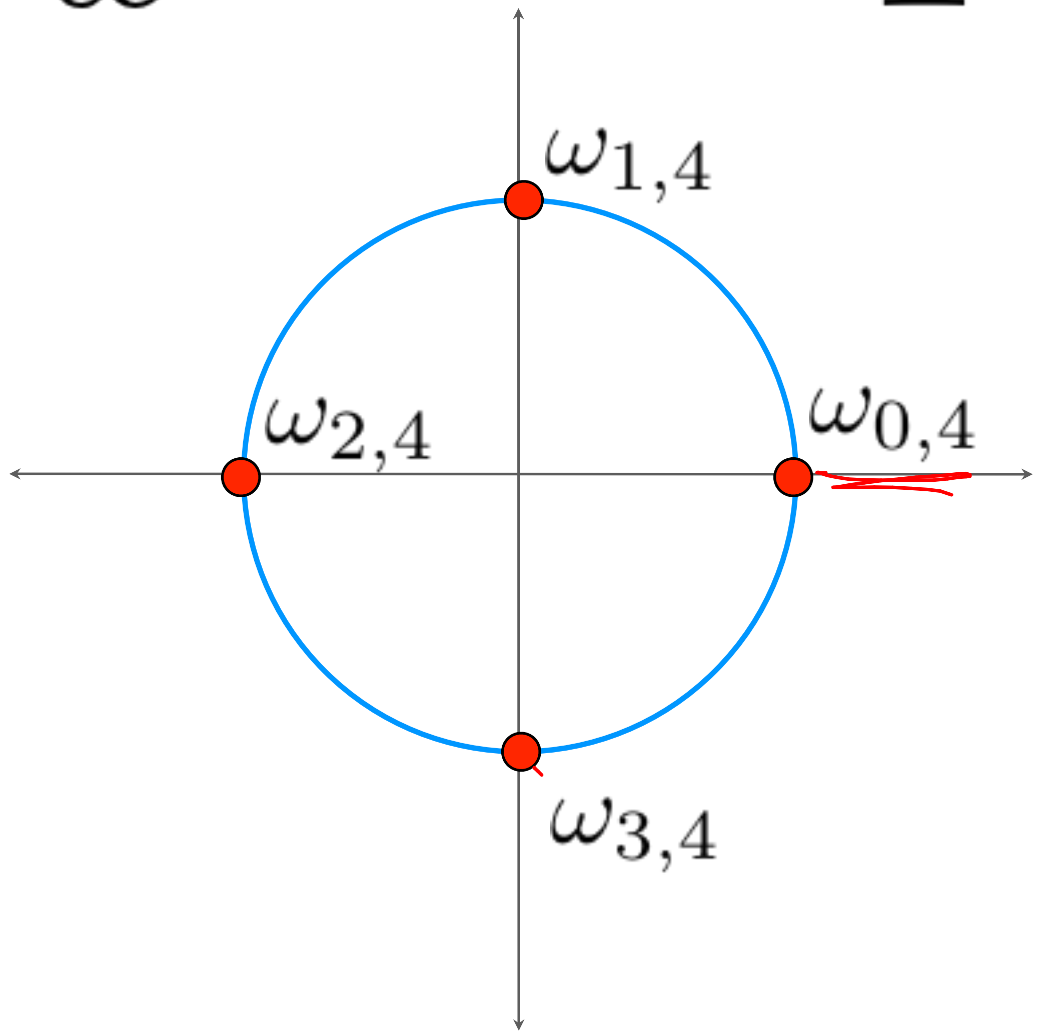


squaring the nth roots of unity

$$x^{n=8} = 1 \qquad\qquad x^{n/2} = 1$$

$i^2 = -1$





→ produce the set of $(n/2)$th roots of unity

$$(\omega_{0,8})^2 = 1^2 = 1$$

$$(\omega_{1,8})^2 = \underline{\left(\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right)^2} = \cancel{\frac{1}{2}} + \frac{\left(\frac{i}{2}\right)\cdot 2}{i} + \cancel{\frac{i^2}{2}} = i = \omega_{1,4}$$

$$(\omega_{2,8})^2 = (i)^2 = -1 = \omega_{2,4}$$

Fact: squaring an nth root produces an n/2th root

example: ~~$\omega_{1,8}$~~ $=$

$$w_{3,8} = \left(-\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right)^2 = \frac{1}{2} - \frac{2i}{2} + \frac{i^2}{2} = \underline{\underline{-i}}$$ ✓

Fact: squaring an nth root produces an n/2th root

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

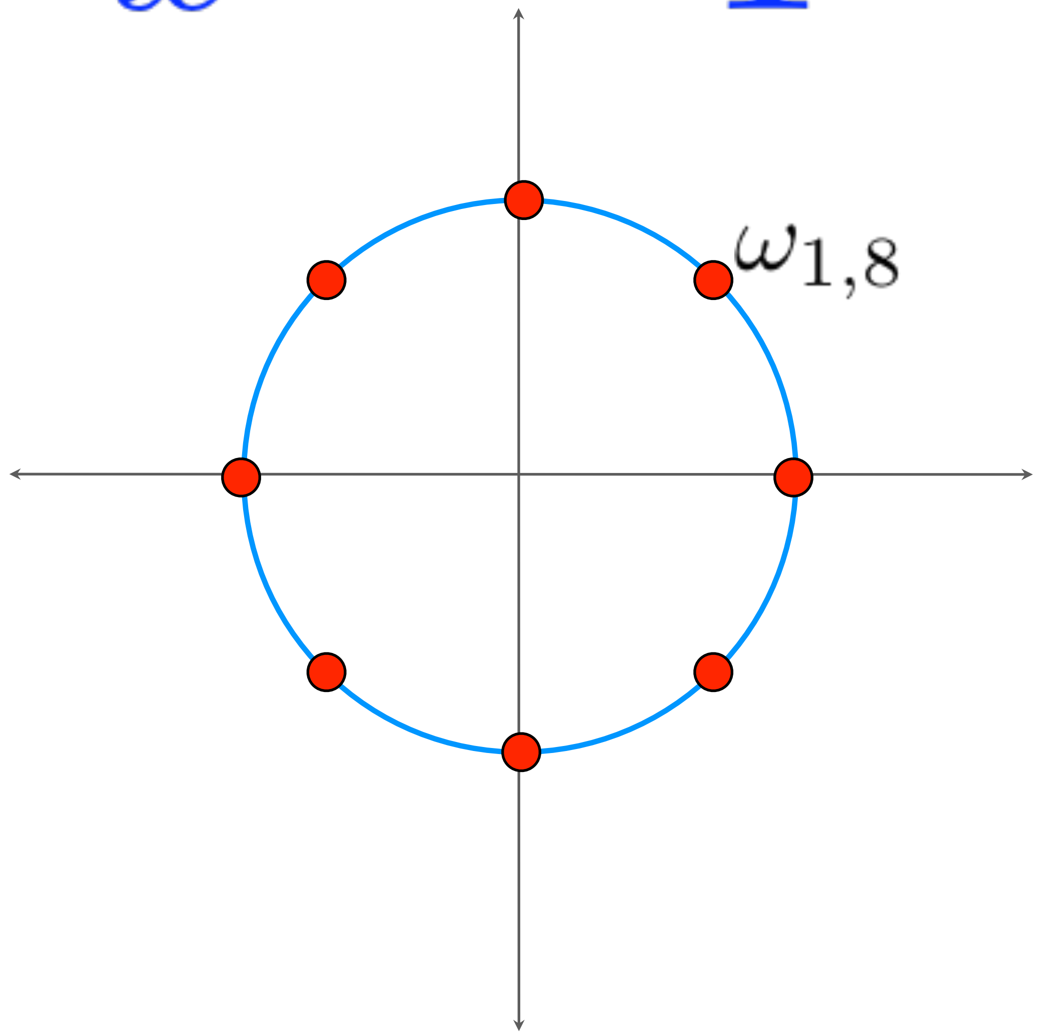
$$\omega_{1,8}^2 = \left(\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right)^2 = \left(\frac{1}{\sqrt{2}}\right)^2 + 2\left(\frac{1}{\sqrt{2}}\frac{i}{\sqrt{2}}\right) + \left(\frac{i}{\sqrt{2}}\right)^2$$

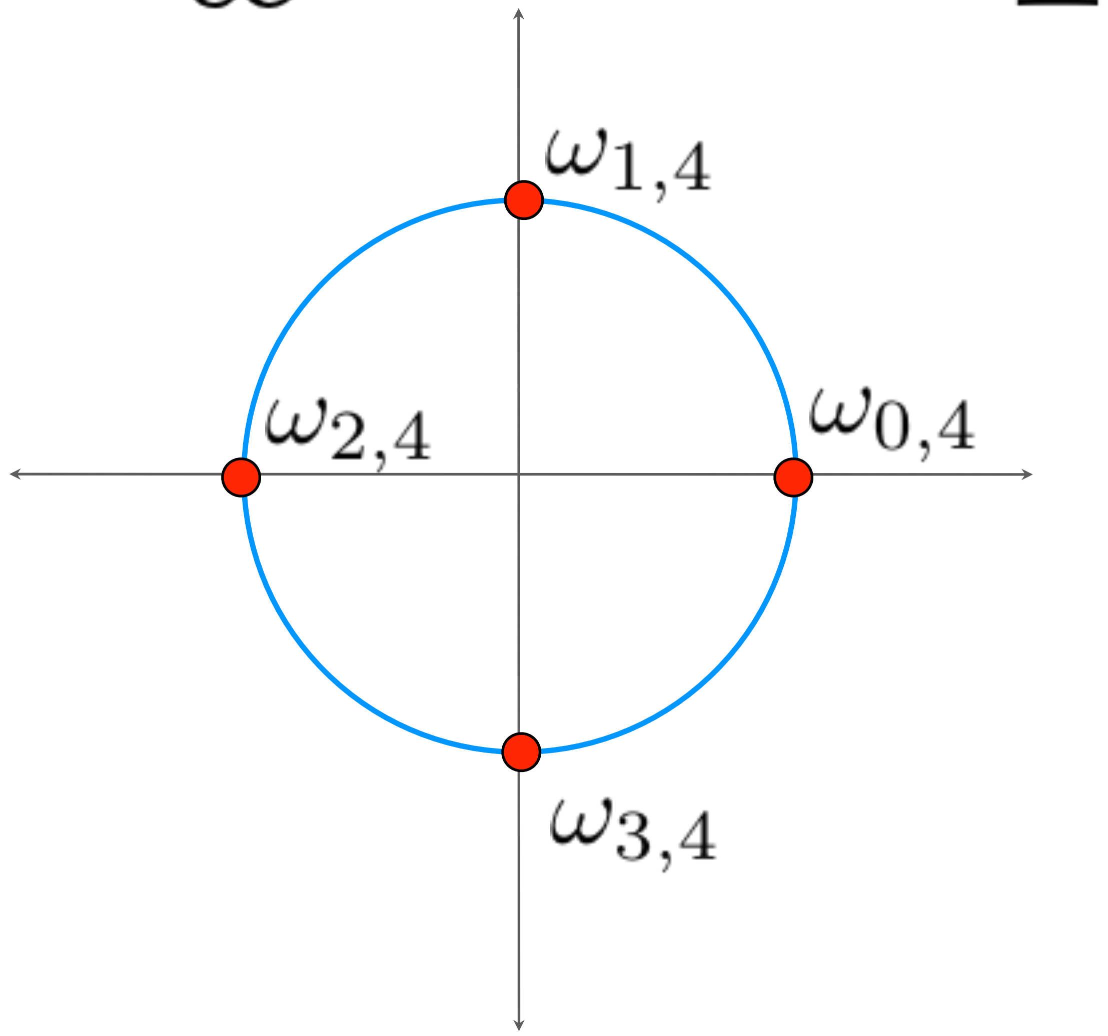
$$= 1/2 + i - 1/2$$

$$= i$$

roots of unity

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





FACT: squaring an nth root
results in an n/2th root

$$A(\underset{\uparrow}{x}) = A_e(x^2) + xA_o(x^2)$$

evaluate at a root of unity

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

evaluate at a root of unity

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

nth root
of unity

n/2th root
of unity

n/2th root
of unity

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

Evaluates degree n poly on the nth roots of unity

$FFT(A_e) \longrightarrow$ evaluation of A_e on the $(\frac{n}{2})^{th}$ roots of unity

$FFT(A_o) \longrightarrow$ " " A_o " " " " " "

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

for all $i = 0 \ldots n-1$, these squares will be $(\frac{n}{2})^{th}$ roots of unity, and so these recursive calls will have the answers

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

```
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:
```

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

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

```
Return n points.
```

application to mult

karatsuba

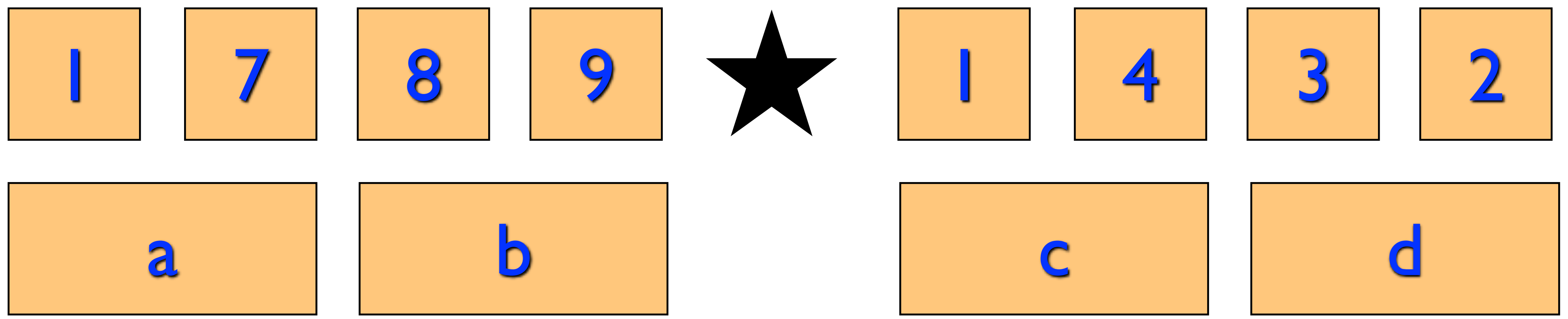
1 7 8 9 ★ 1 4 3 2

a b c d

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

application to mult

karatsuba



1 7 8 9 ★ 1 4 3 2

a b c d

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

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

4 1 0 2

a3 a2 a1 a0 ★ b3 b2 b1 b0

$A(10) = 4102$

$A(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + 0x^4 + \dots + 0 \cdot a_7 \cdot x^7$

$B(x) = b_0 + b_1x + b_2x^2 + \dots + 0 \cdot b_7 \cdot x^7$

FFT(A): $A(w_0) \dots A(w_7) \rightarrow$ FFT $\Theta(n \log n)$

FFT(B): $B(w_0) \dots B(w_7)$ $\Theta(n \log n)$

$\rightarrow C(w_0) \dots C(w_7) \rightarrow \tilde{\Theta}(n)$

IFFT $C(x) = A(x)B(x)$ $\Theta(n \log n)$

$C(x) = c_0 + c_1x + c_2x^2 + \dots c_7x^7$

$C(10) = A(10) \cdot B(10)$

$=$

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

$\Theta(n \cdot \log \cdot \log \log n)$ – Schonhage-Strassen

$\Theta(n \cdot \log(n) \cdot \boxed{2^{\log^*(n)}})$ – Fürer

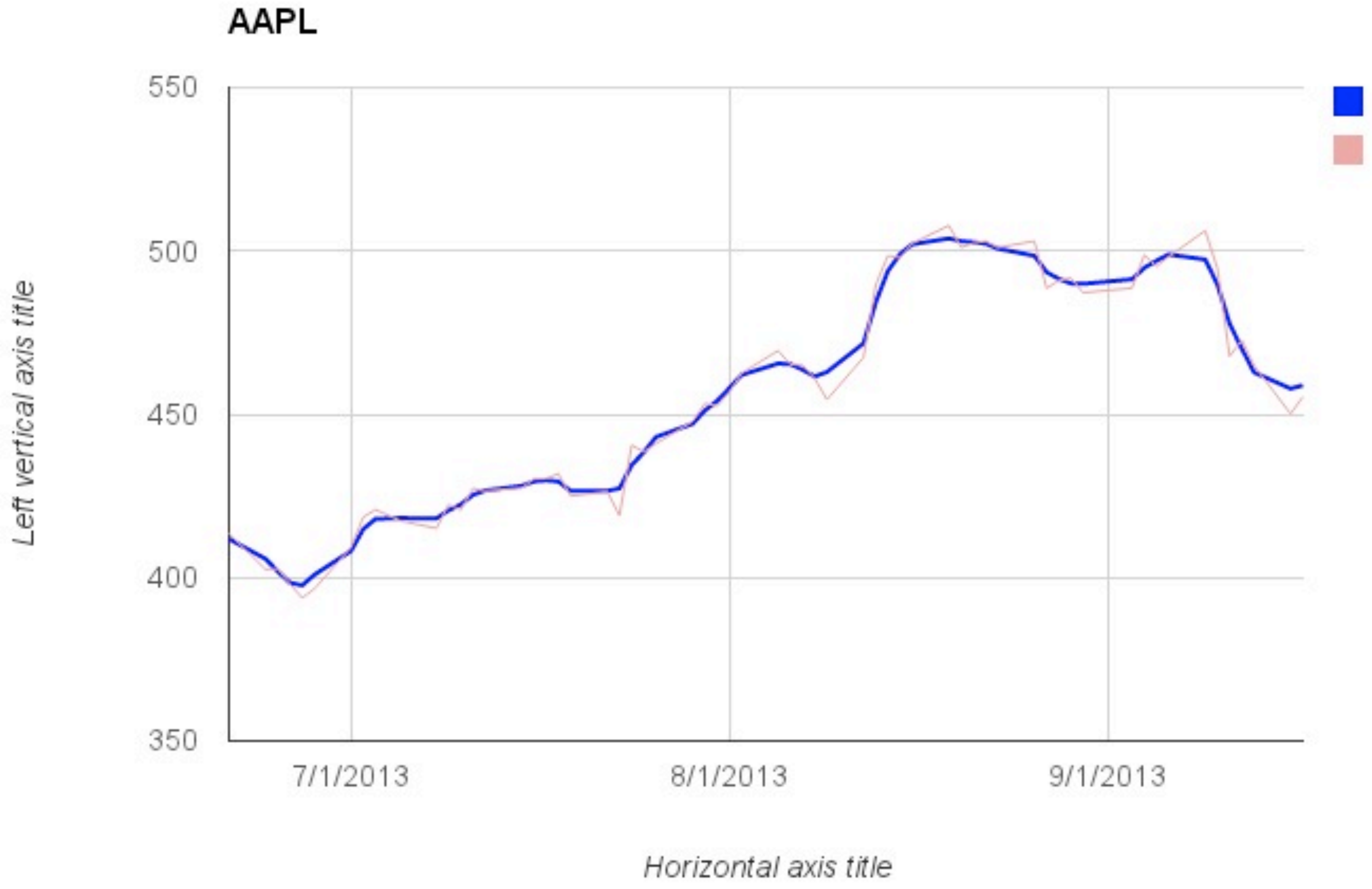
↑ $\underline{2006}$

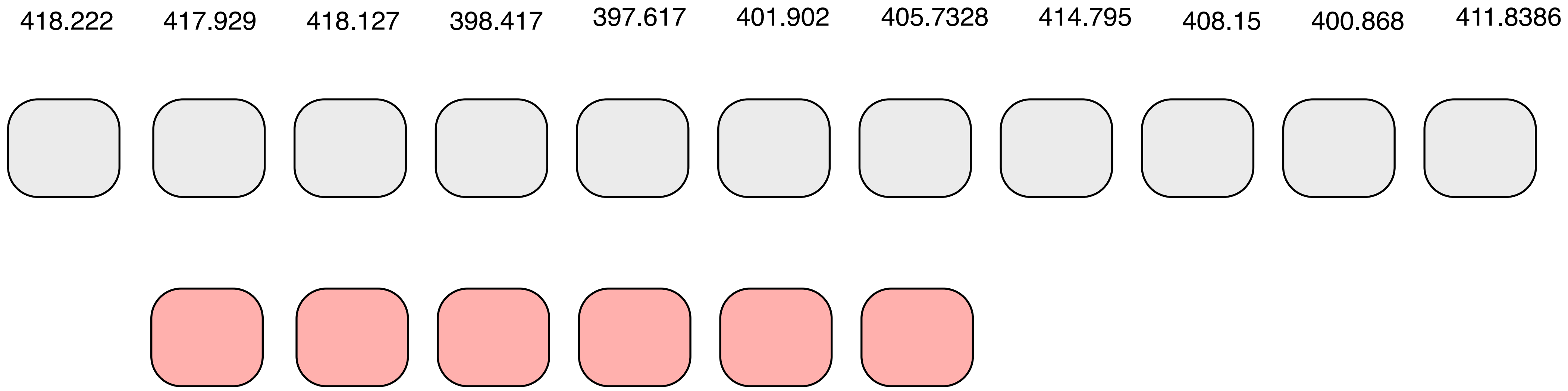
$\rightarrow \underline{32}$

$\log^*(2^{65530}) = 5$

Applications of FFT

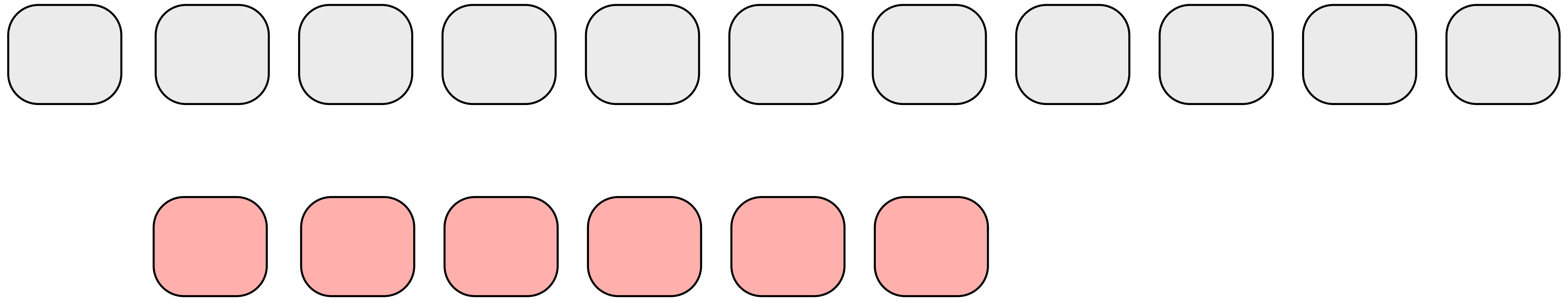
Applications of FFT

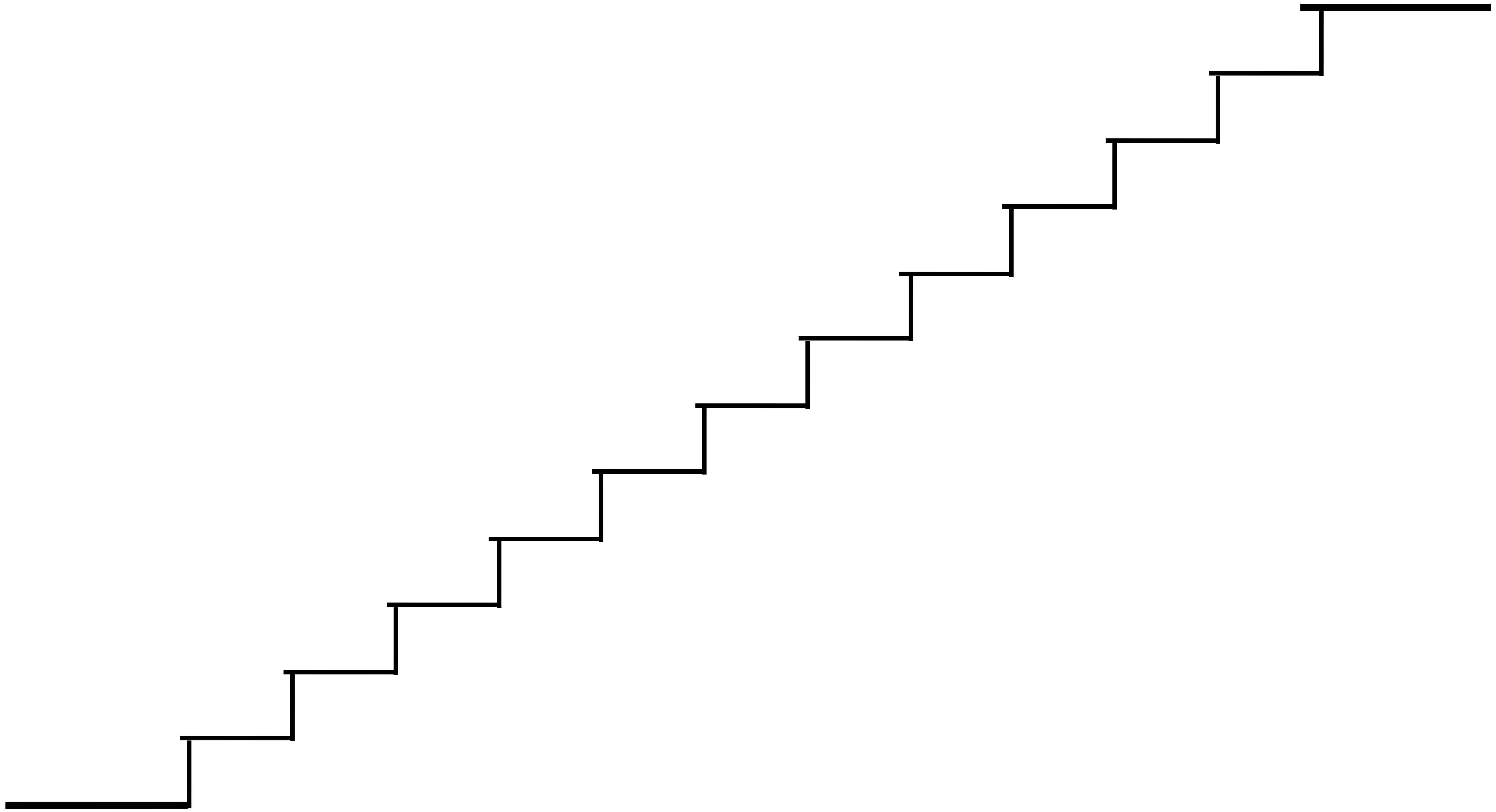




418.222 417.929 418.127 398.417 397.617 401.902 405.7328 414.795 408.15 400.868 411.8386

418.222 417.929 418.127 398.417 397.617 401.902 405.7328 414.795 408.15 400.868 411.8386





```
Stairs(n)
    if n<=1 return 1
    return Stairs(n-1) + Stairs(n-2)
```

```
Stairs(n)  if n<=1 return 1
           ret Stairs(n-1) + Stairs(n-2)
```

Stairs(5)

Stairs(4) Stairs(3)

Stairs(3) Stairs(2) Stairs(2) Stairs(1)

Stairs(2) Stairs(1) Stairs(1) Stairs(0) Stairs(1) Stairs(0)

```
    initialize memory M

  Stairs(n)
    if n<=1 then return 1
    if n is in M, return M[n]
    answer = Stairs(i-1)+ Stairs(i-2)
    M[n] = answer
    return answer
```

```
Stairs(n)
    if n<=1 then return 1
    if n is in M, return M[n]
    answer = Stairs(i-1)+ Stairs(i-2)
    M[n] = answer
    return answer
```

Stairs(5)

Stairs(n)

```
stair[0]=1
stair[1]=1
for i=2 to n
  stair[i] = stair[i–1]+stair[i–2]
return stair[i]
```

```
    initialize memory M

Stairs(n)
```

```
Stairs(n)
    if n<=1 then return 1
    if n is in M, return M[n]
    answer = Stairs(i−1)+ Stairs(i−2)
    M[n] = answer
    return answer
```

Stairs(5)

Stairs(n)

```
stair[0]=1
stair[1]=1
for i=2 to n
  stair[i] = stair[i−1]+stair[i−2]
return stair[i]
```

Dynamic Programming

two big ideas

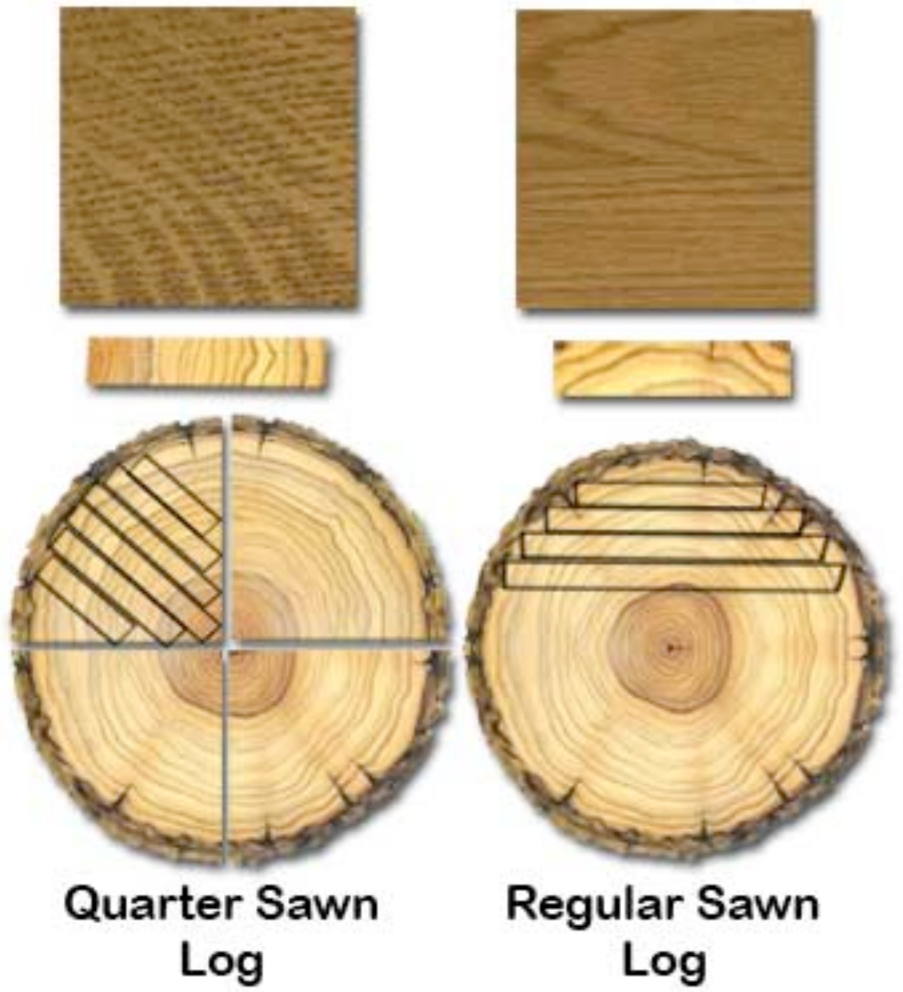
two big ideas

recursive structure

+

memoizing

wood cutting



http://www.amishhandcraftedheirlooms.com/quarter-sawn-oak.htm

http://snlm.files.wordpress.com/2008/08/bill-wakefield-and-carl-fie.gif

Spot price for lumber

Spot price for lumber

1”	2”	3”	4”	5”	6”	7”	8”

Log cutter dilemna

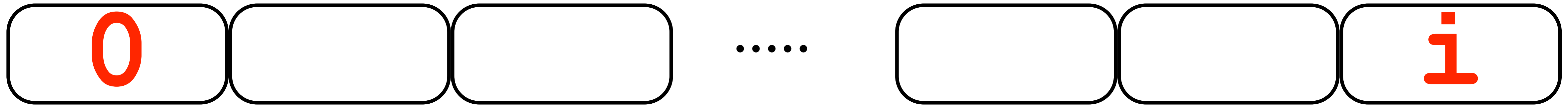
input to the problem: $n, (p_1, \ldots, p_n)$

goal:

Observation

Solution equation

Approach



```
BestLogs($n, (p_1, \ldots, p_n)$)
    if n<=0 return 0
```

```
BestLogs($n, (p_1, \ldots, p_n)$)
    if n<=0 return 0
    for i=1 to n
     Best[i] = $\max_{k=1 \ldots i} \{p_k + \text{Best}[i-k]\}$
```

The actual cuts?

```
BestLogs($n, (p_1, \ldots, p_n)$)
    if n<=0 return 0
    for i=1 to n
     Best[i] = $\max_{k=1\ldots i}\{p_k + \mathrm{Best}[i-k]\}$
```