L6 5800

feb 4/7 2022

shelat

Matrix

multiplication

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

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

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

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

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

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

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

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

$$
\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix}
= \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}
$$

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

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

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

$$\Theta(n^3)$$

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

[Strassen]

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

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

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

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

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

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

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

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

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

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

[strassen]

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

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

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

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

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

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

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

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

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

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

[strassen]

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

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

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

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

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

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

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

$$M(n) = 7M(n/2) + 18n^2$$

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

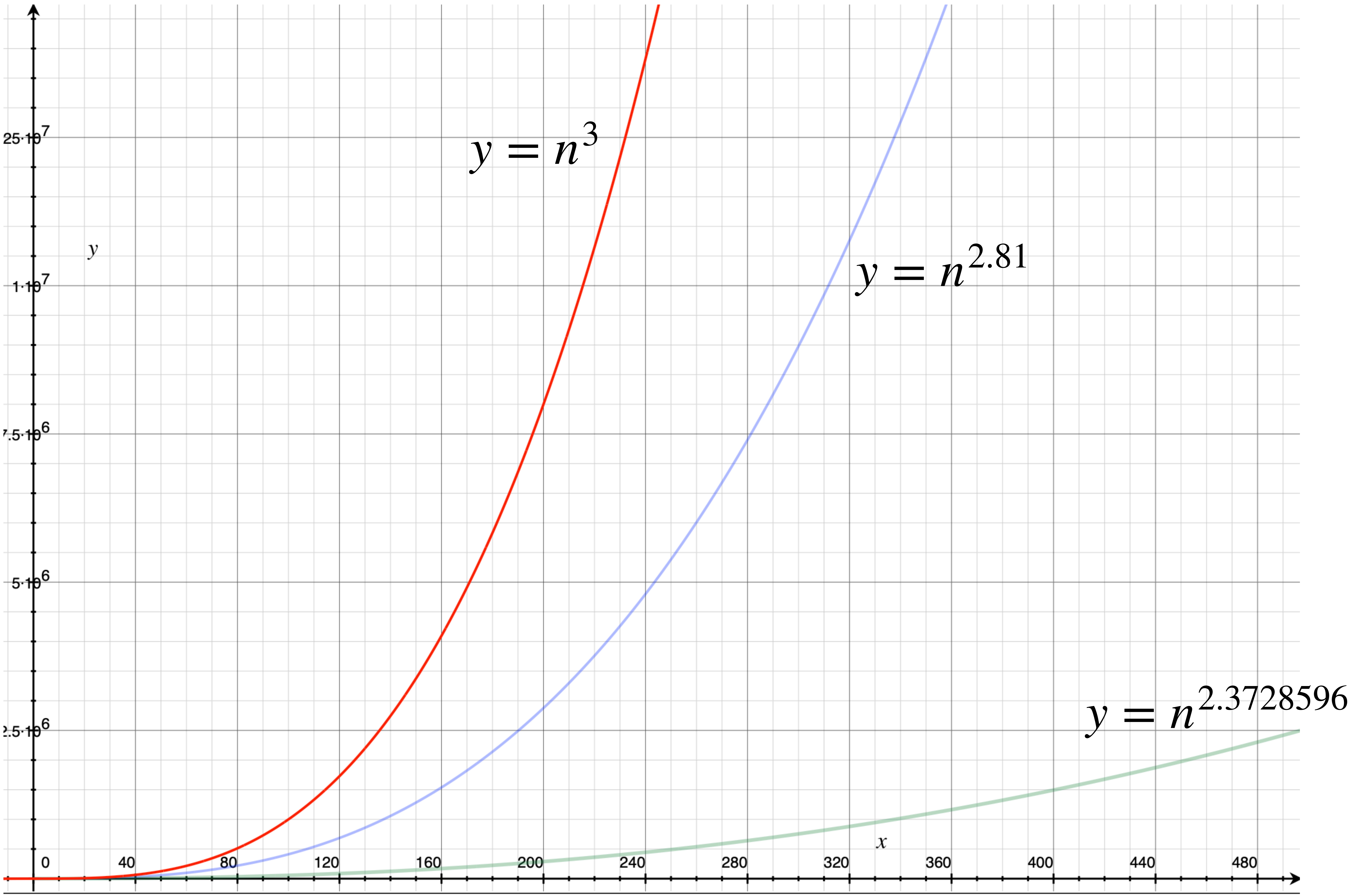
taking this idea further

3x3 matricies [Laderman'75] in 23 mults

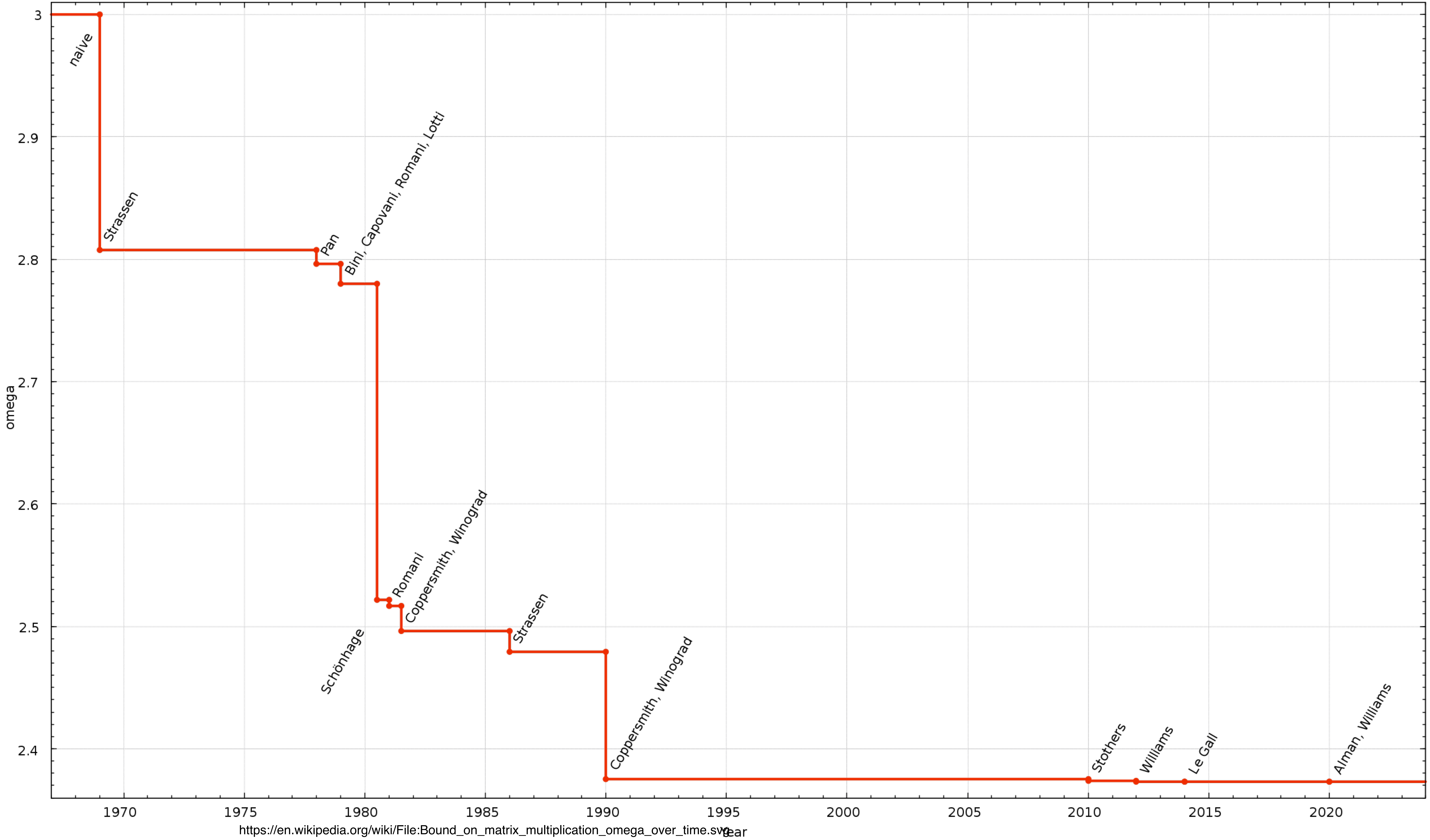
1978 victor pan method

70x70 matrix using 143640 mults

what is the recurrence:

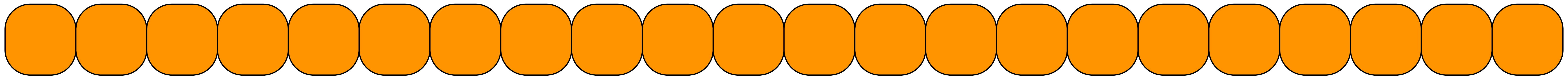


$y = n^3$
$y = n^{2.81}$
$y = n^{2.3728596}$
y
x
25·10^7
1·10^7
7.5·10^6
5·10^6
2.5·10^6
0
40
80
120
160
200
240
280
320
360
400
440
480

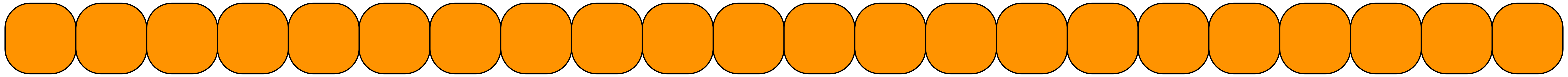


https://en.wikipedia.org/wiki/File:Bound_on_matrix_multiplication_omega_over_time.svg

MEDIAN



problem: given a list of **n** elements, find the element of rank **n**/2. (half are larger, half are smaller)

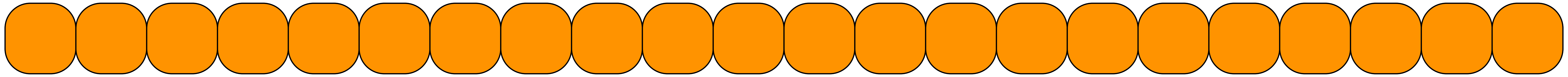


problem: given a list of **n** elements, find the element of rank n/2. (half are larger, half are smaller)

can generalize to **i**

first solution: sort and pluck.

$$O(n \log n)$$

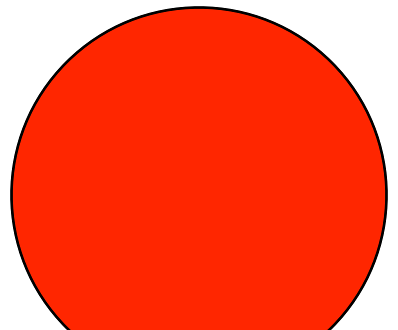


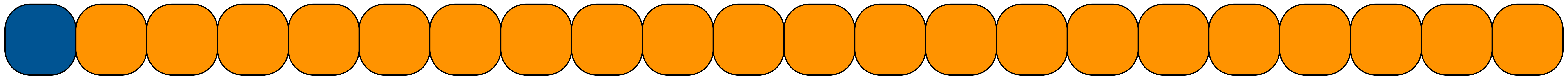
problem: given a list of **n** elements, find the element of rank **i**.

key insight:

we do not have to "fully" sort.

semi sort can suffice.

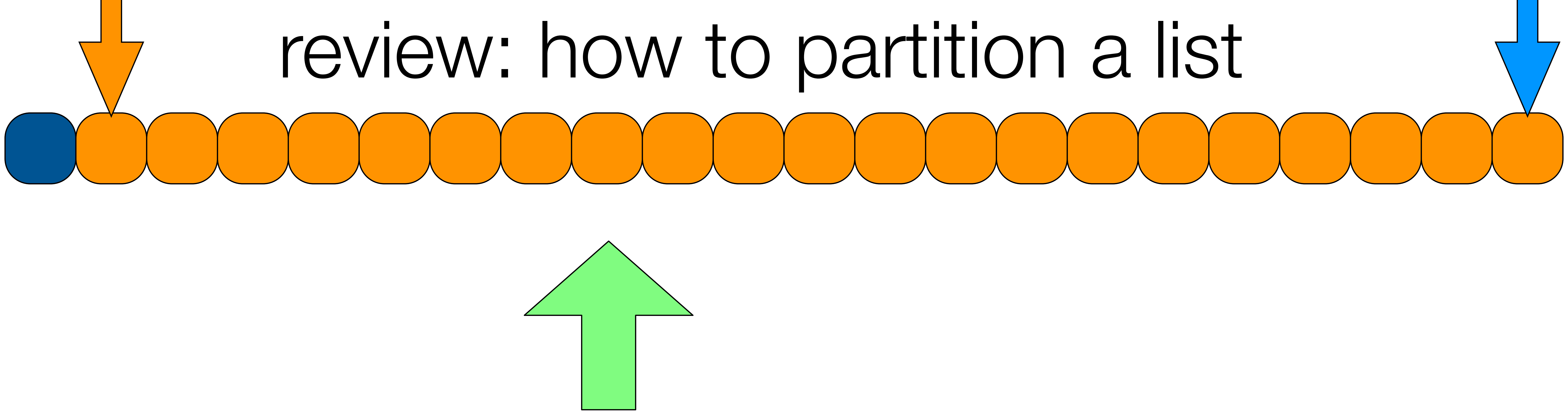




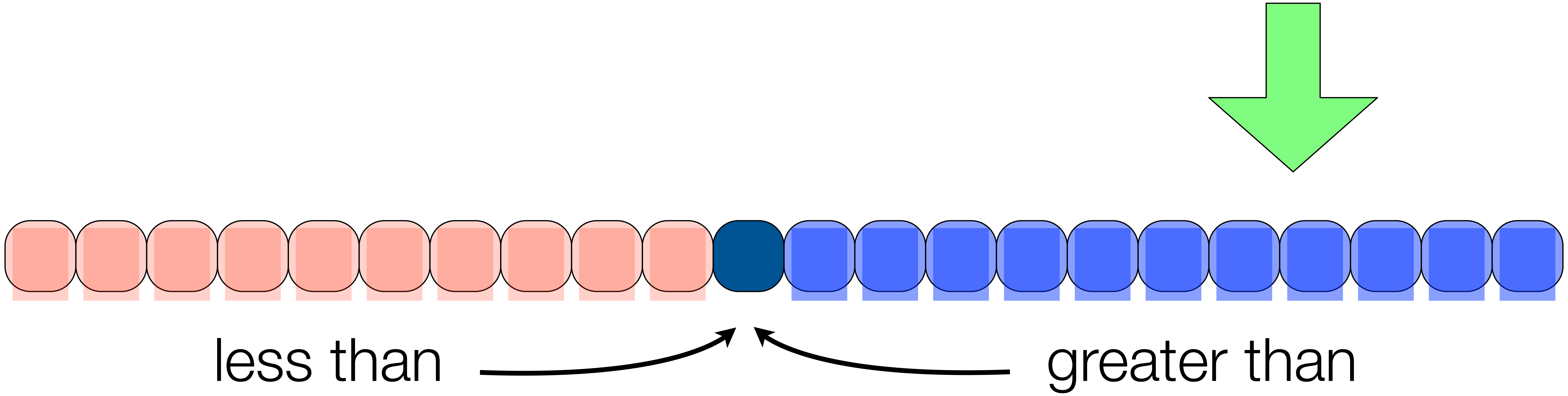
pick first element
partition list about this one
see where we stand

review: how to partition a list

review: how to partition a list



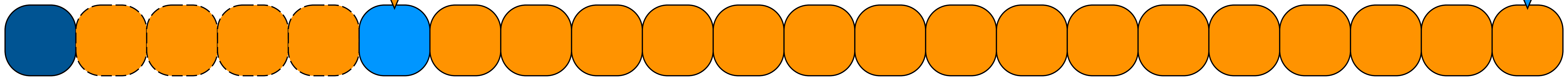
GOAL: start with THIS LIST and END with THAT LIST



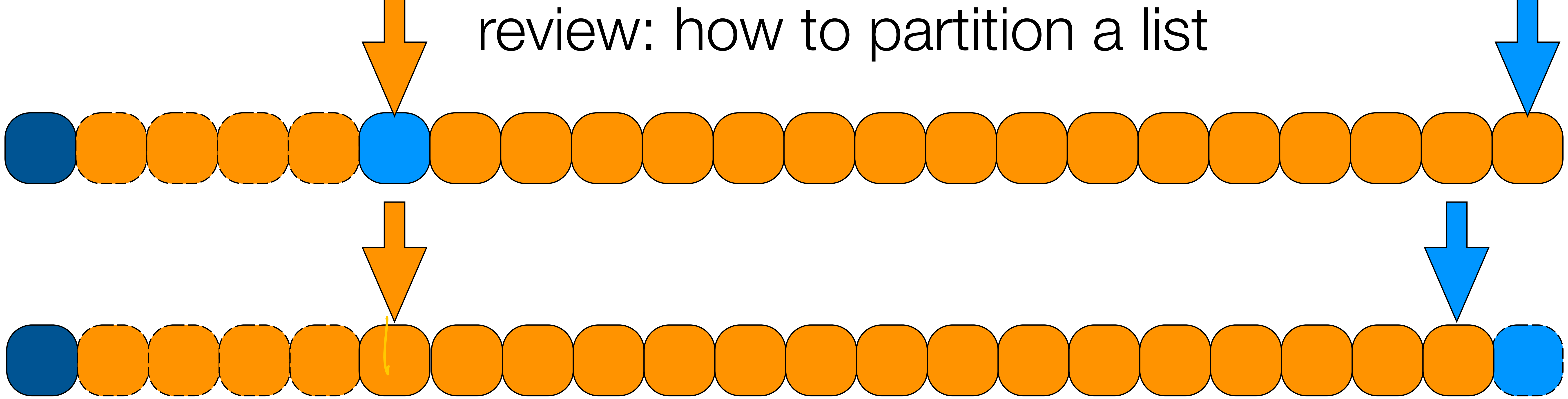
review: how to partition a list

review: how to partition a list

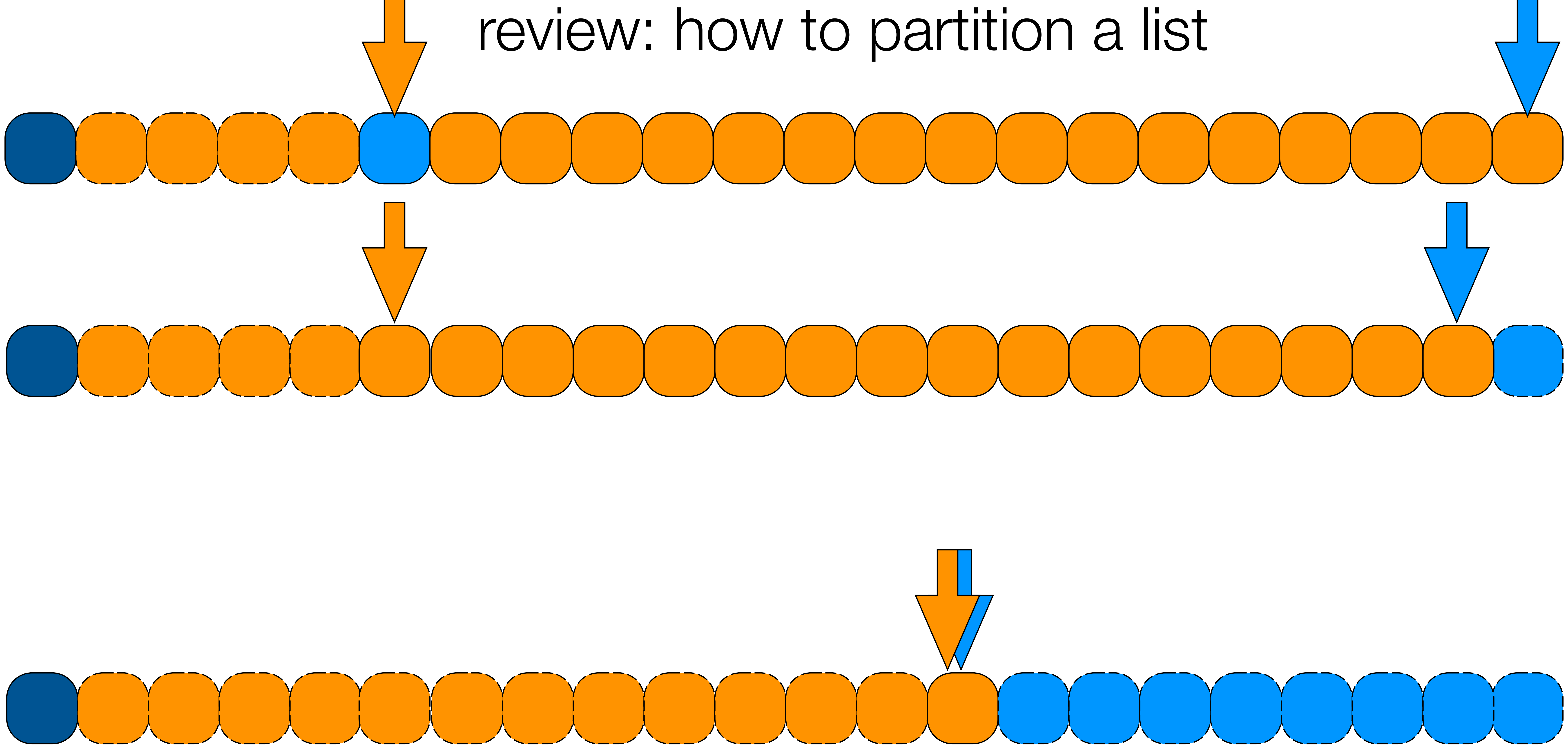
swap



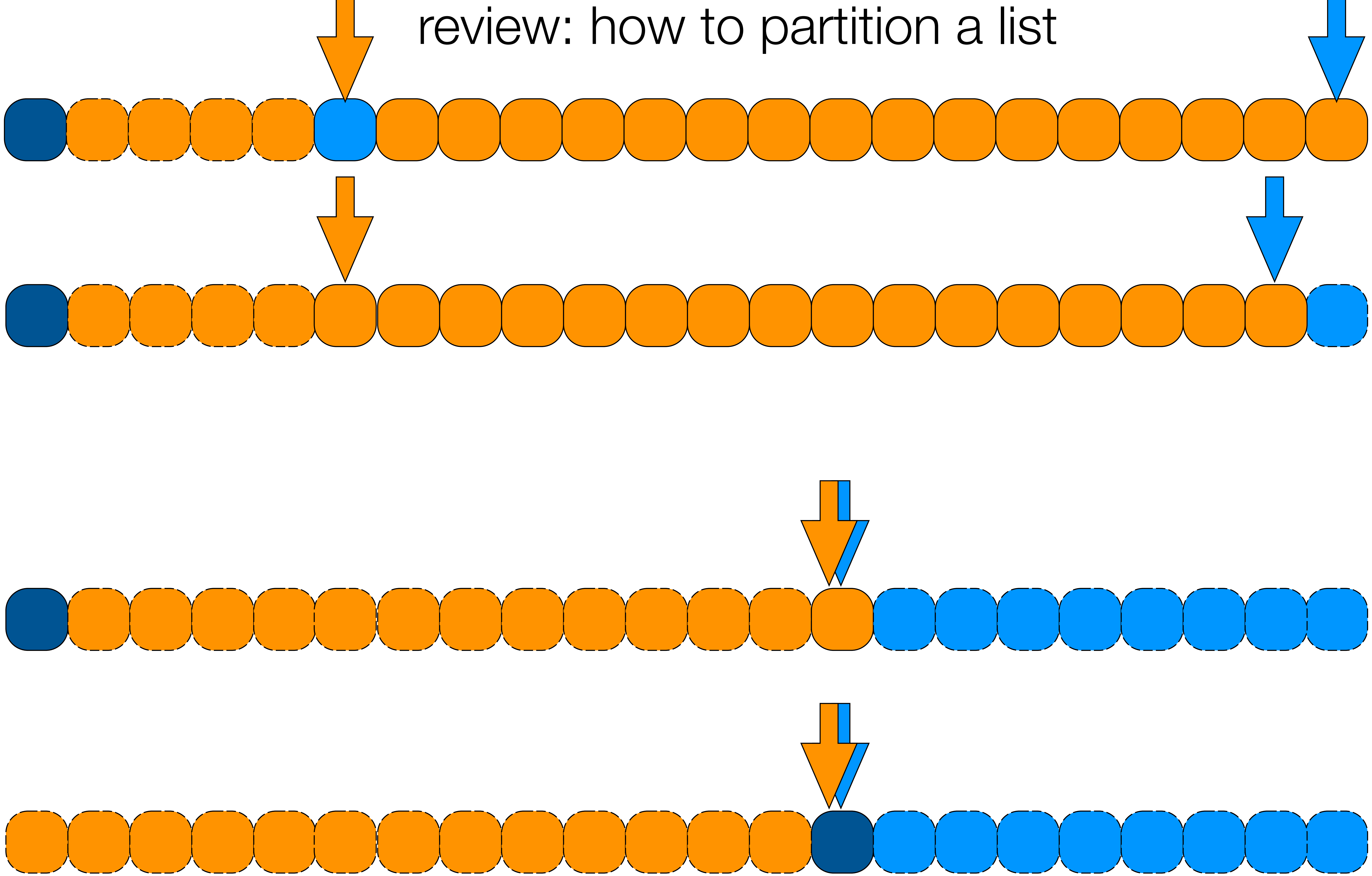
review: how to partition a list



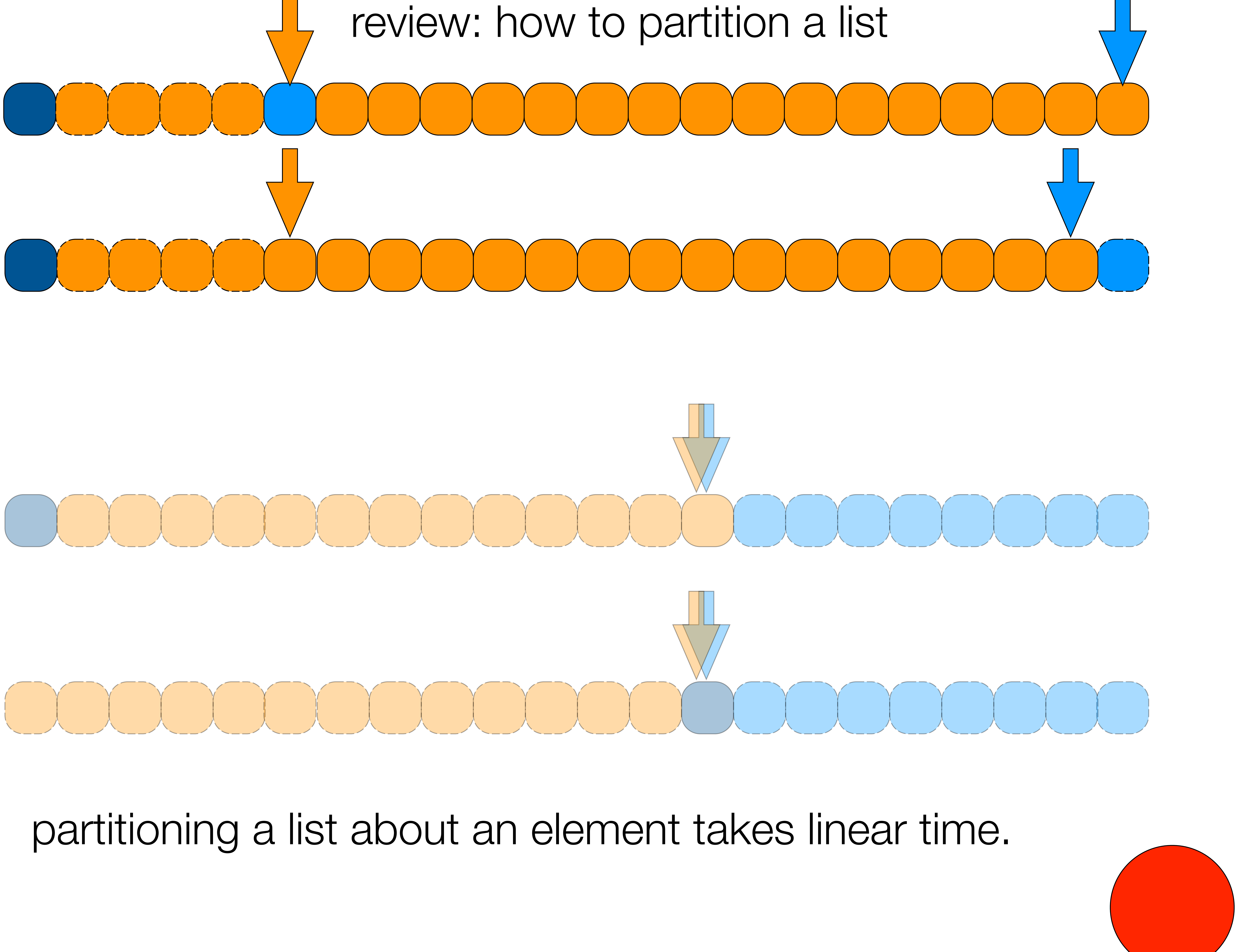
review: how to partition a list



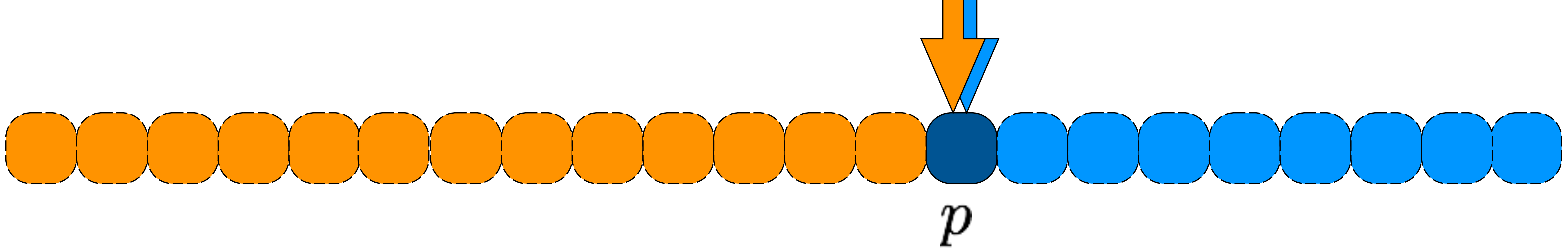
review: how to partition a list



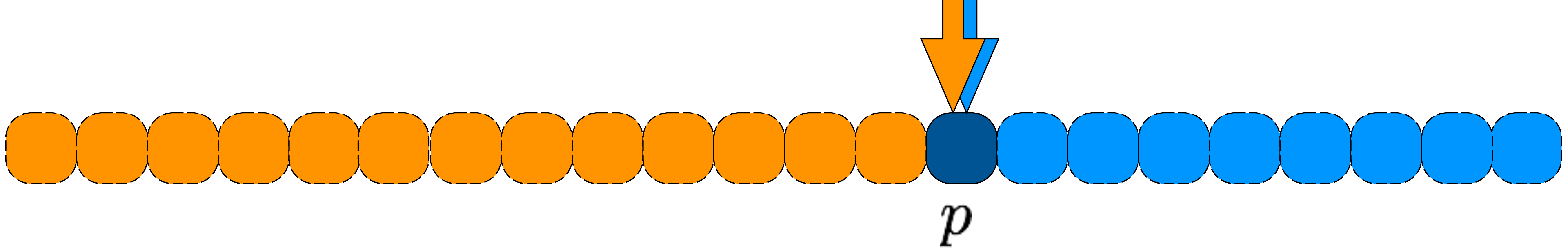
review: how to partition a list



partitioning a list about an element takes linear time.



select $(i, A[1, \ldots, n])$



select $(i, A[1, \ldots, n])$

handle base case of 1 element.

partition list about first element

if pivot p is position **i**, return pivot

else if pivot p is in position > **i** select $(i, A[1, \ldots, p-1])$

else select $((i-p-1), A[p+1, \ldots, n])$

select $(i, A[1, \ldots, n])$

Assume our partition always splits list into two eql parts

handle base case.

partition list about first element

if pivot is position **i**, return pivot

else if pivot is in position > i select $(i, A[1, \ldots, p-1])$

else select $((i-p-1), A[p+1, \ldots, n])$

select $(i, A[1, \ldots, n])$

Assume our partition always splits list into two eql parts

handle base case.

partition list about first element

if pivot is position **i**, return pivot

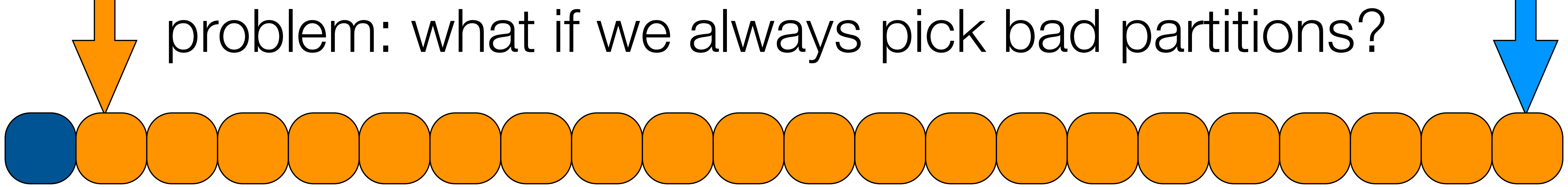
else if pivot is in position > i select $(i, A[1, \ldots, p-1])$

else select $((i-p-1), A[p+1, \ldots, n])$

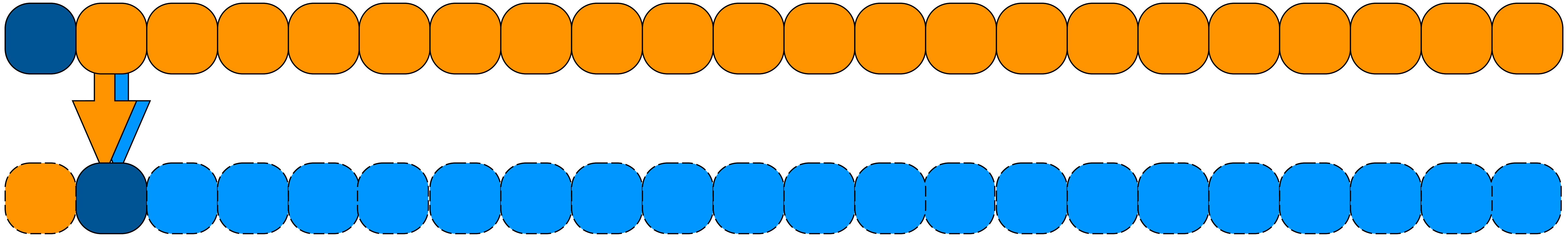
$$T(n) = T(n/2) + O(n)$$

$$\Theta(n)$$

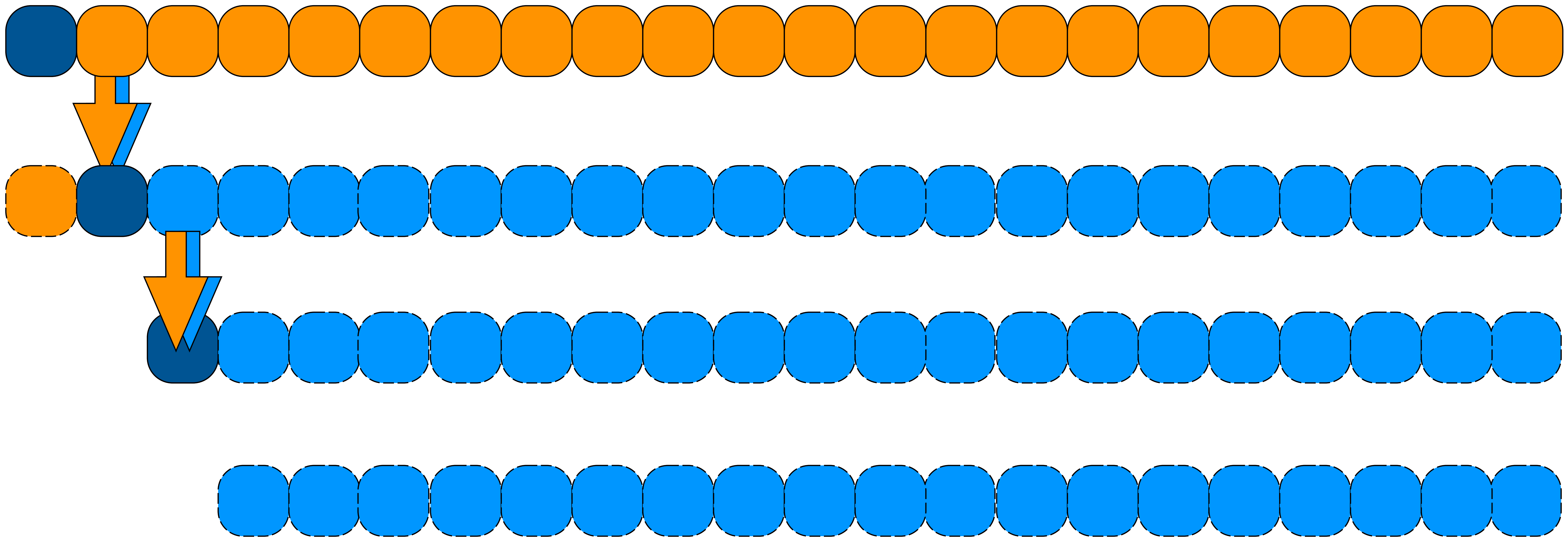
problem: what if we always pick bad partitions?

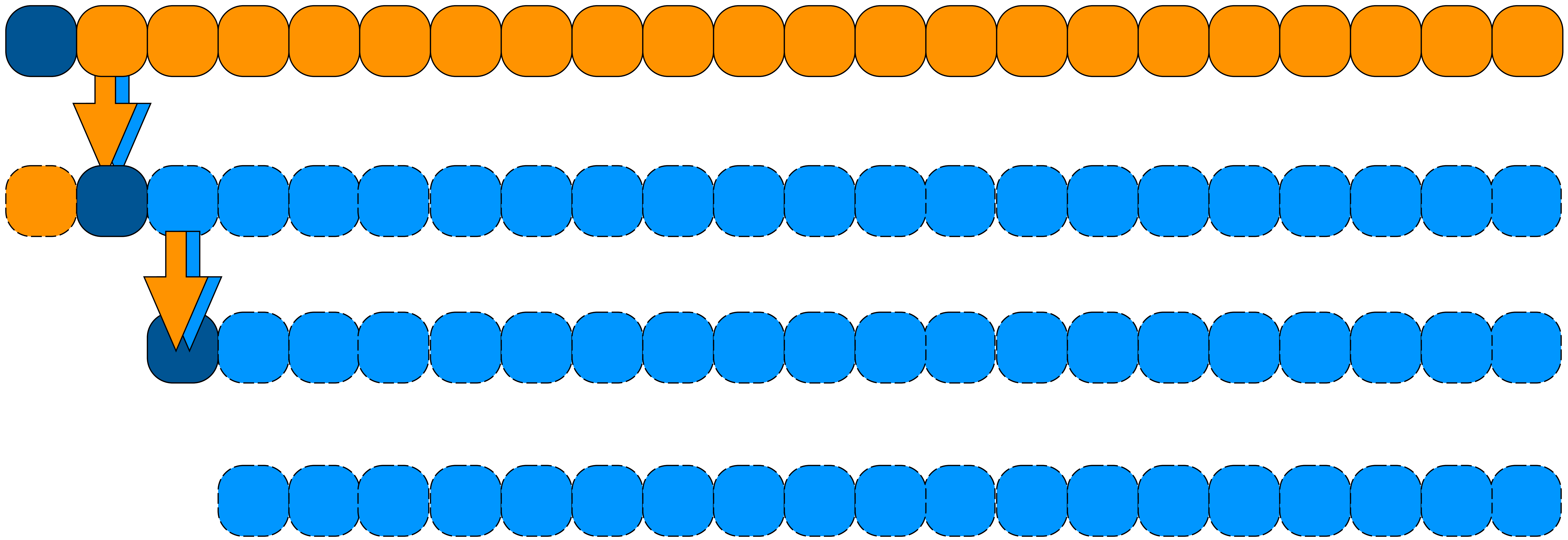


problem: what if we always pick bad partitions?



problem: what if we always pick bad partitions?





problem: what if we always pick bad partitions?

select $(i, A[1, \ldots, n])$

handle base case.
partition list about first element
if pivot is position **i**, return pivot
else if pivot is in position > i select $(i, A[1, \ldots, p-1])$
else select $((i-p-1), A[p+1, \ldots, n])$

select $(i, A[1, \ldots, n])$

handle base case.

partition list about first element

if pivot is position **i**, return pivot

else if pivot is in position > **i** select $(i, A[1, \ldots, p-1])$

else select $((i-p-1), A[p+1, \ldots, n])$

$$T(n) = T(n-1) + O(n)$$

$$\Theta(n^2)$$

Needed:

a good partition element

partition $(A[1, \ldots, n])$

Needed:

a good partition element

partition $(A[1, \ldots, n])$

produce an element where
30% smaller, 30% larger

solution: bootstrap

image: gucci

image: mark nason

image: d&g

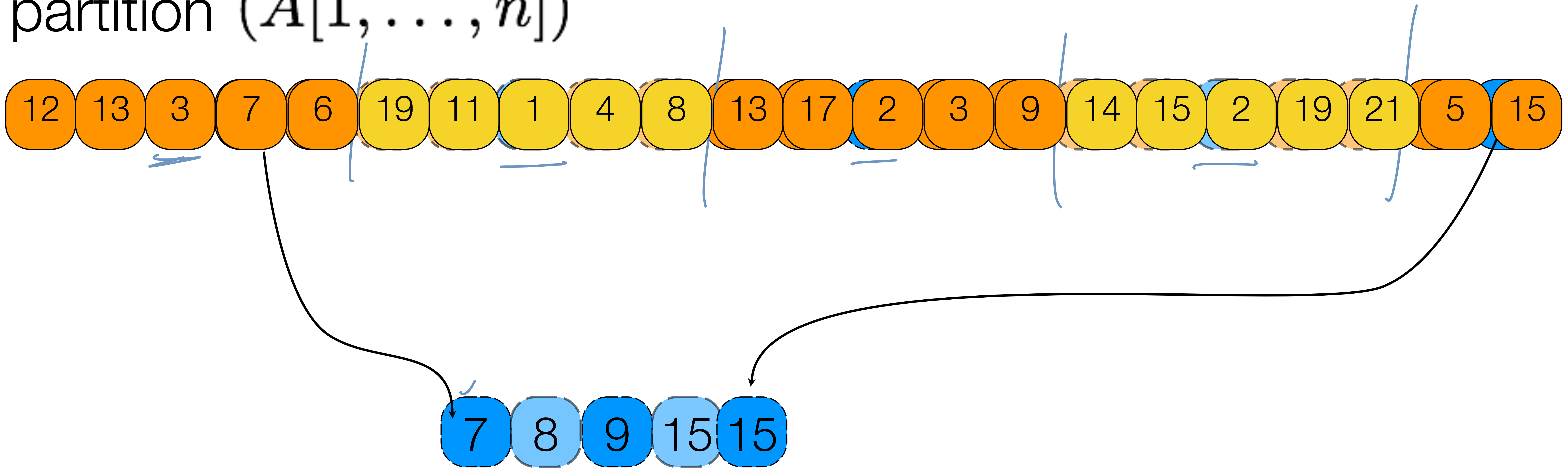
partition $(A[1, \ldots, n])$

12	13	3	7	6	19	11	1	4	8	13	17	2	3	9	14	15	2	19	21

partition $(A[1,\ldots,n])$

12	13	3	7	6	19	11	1	4	8	13	17	2	3	9	14	15	2	19	21	5	15

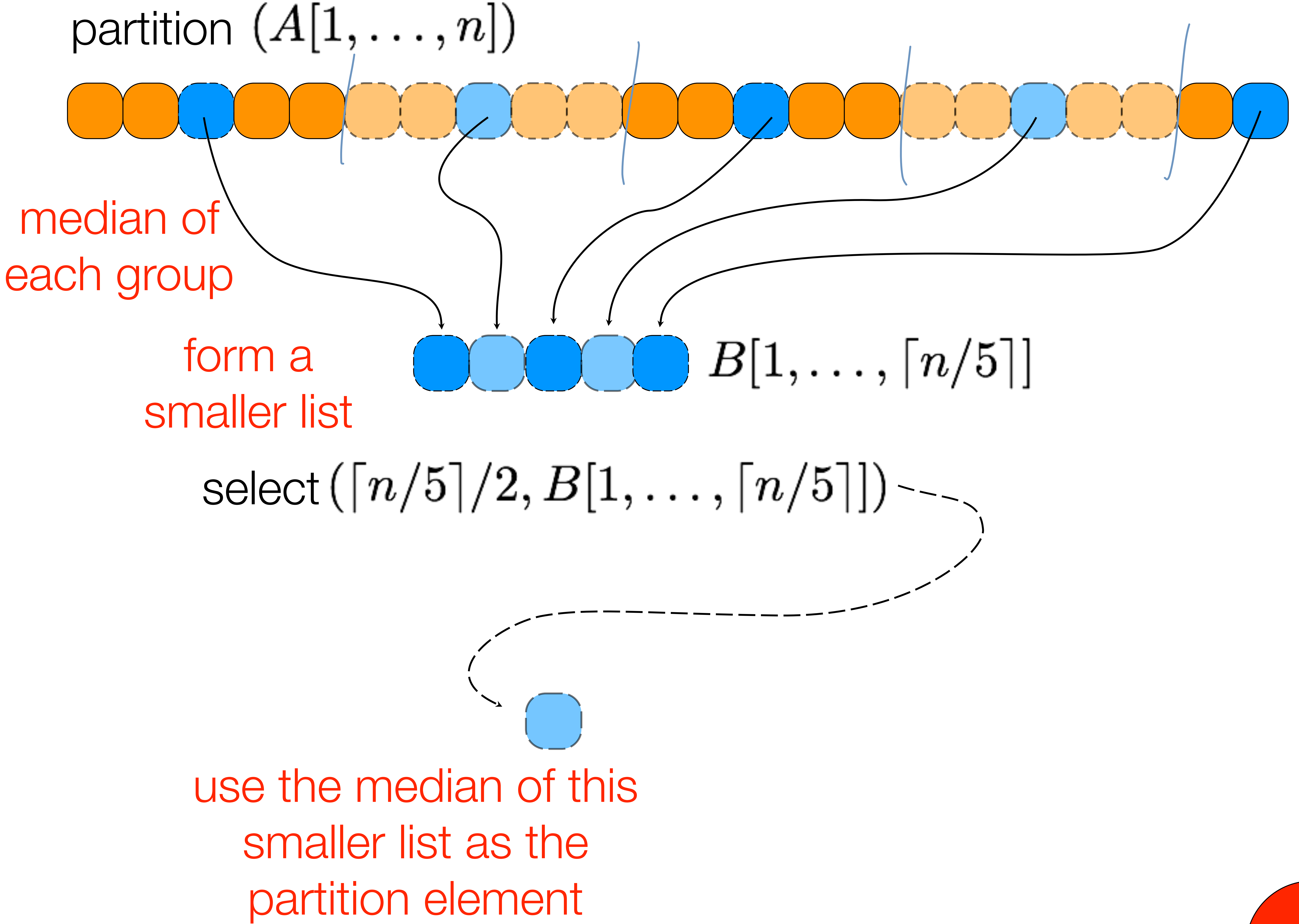
partition $(A[1,\ldots,n])$



divide list into groups of 5 elements

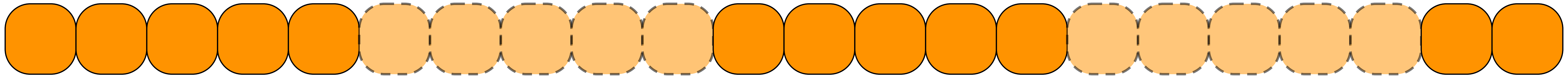
find median of each small list using brute force

gather all medians



partition $(A[1, \ldots, n])$
median of each group
form a smaller list
$B[1, \ldots, \lceil n/5 \rceil]$
select $(\lceil n/5 \rceil / 2, B[1, \ldots, \lceil n/5 \rceil])$
use the median of this smaller list as the partition element

partition $(A[1,\ldots,n])$



divide list into groups of 5 elements

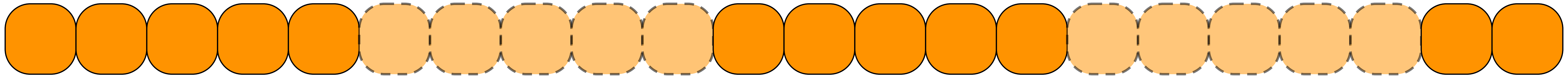
find median of each small list using brute force

gather all medians

call select(...) on this sublist to find median

return the result

partition $(A[1,\ldots,n])$



divide list into groups of 5 elements

find median of each small list

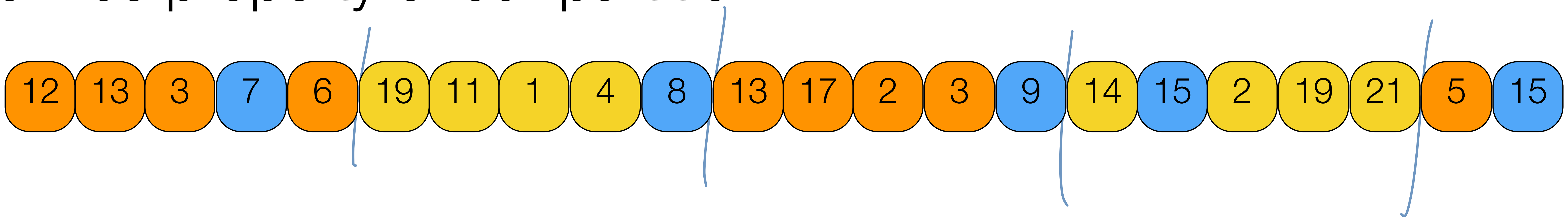
gather all medians

call select(...) on this sublist to find median

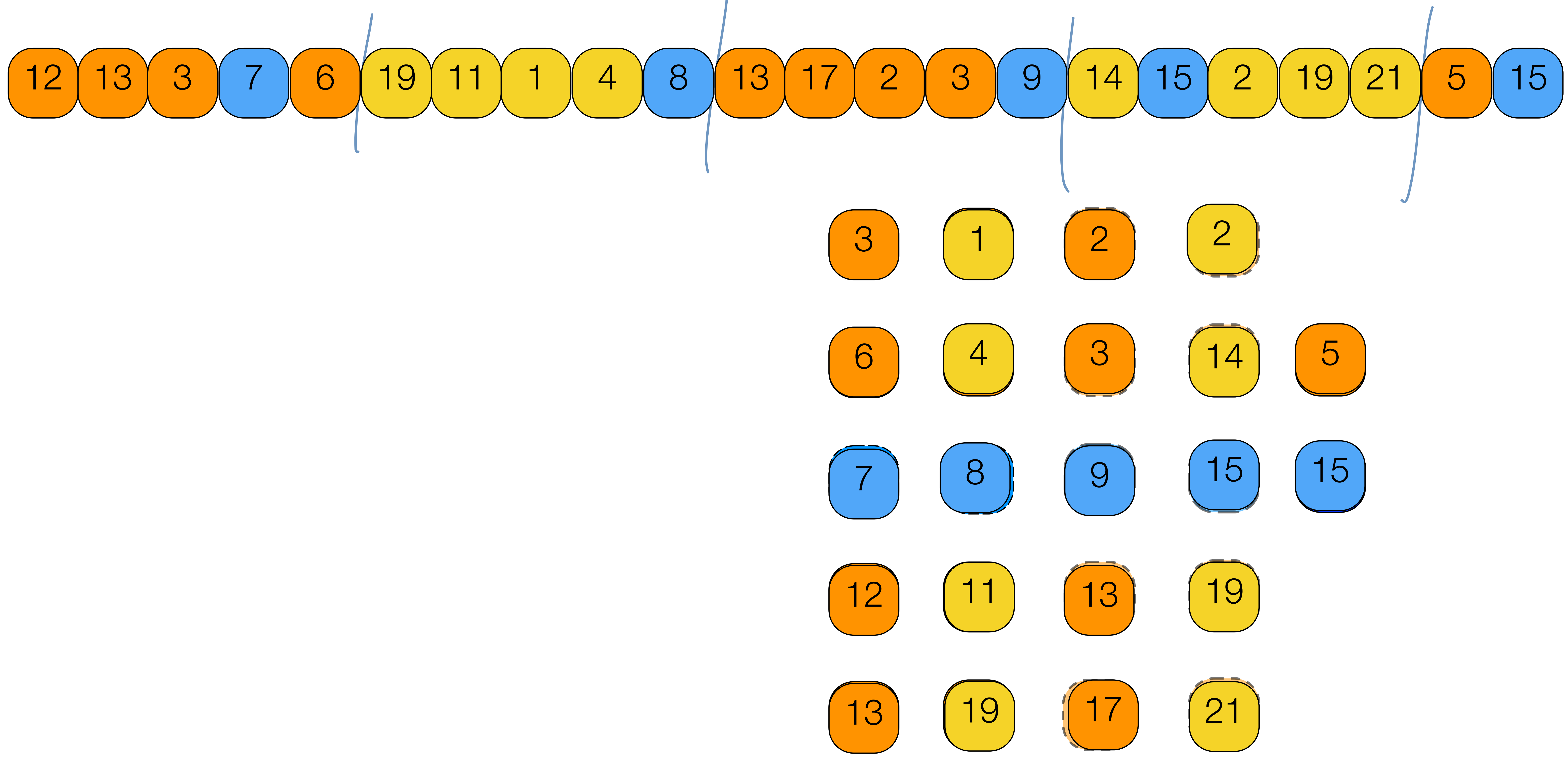
return the result

$$P(n) = S(\lceil n/5 \rceil) + O(n)$$

a nice property of our partition

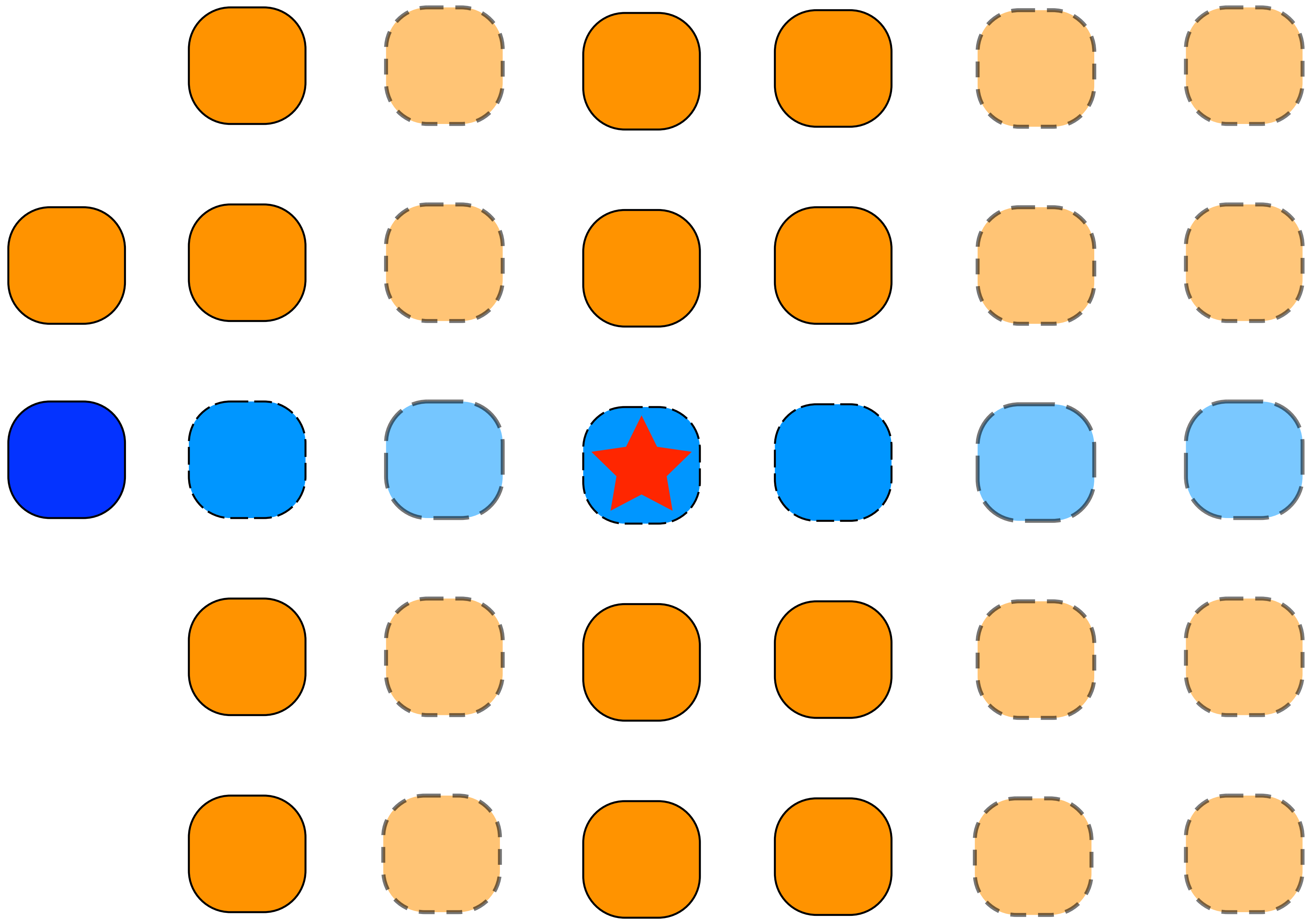


a nice property of our partition



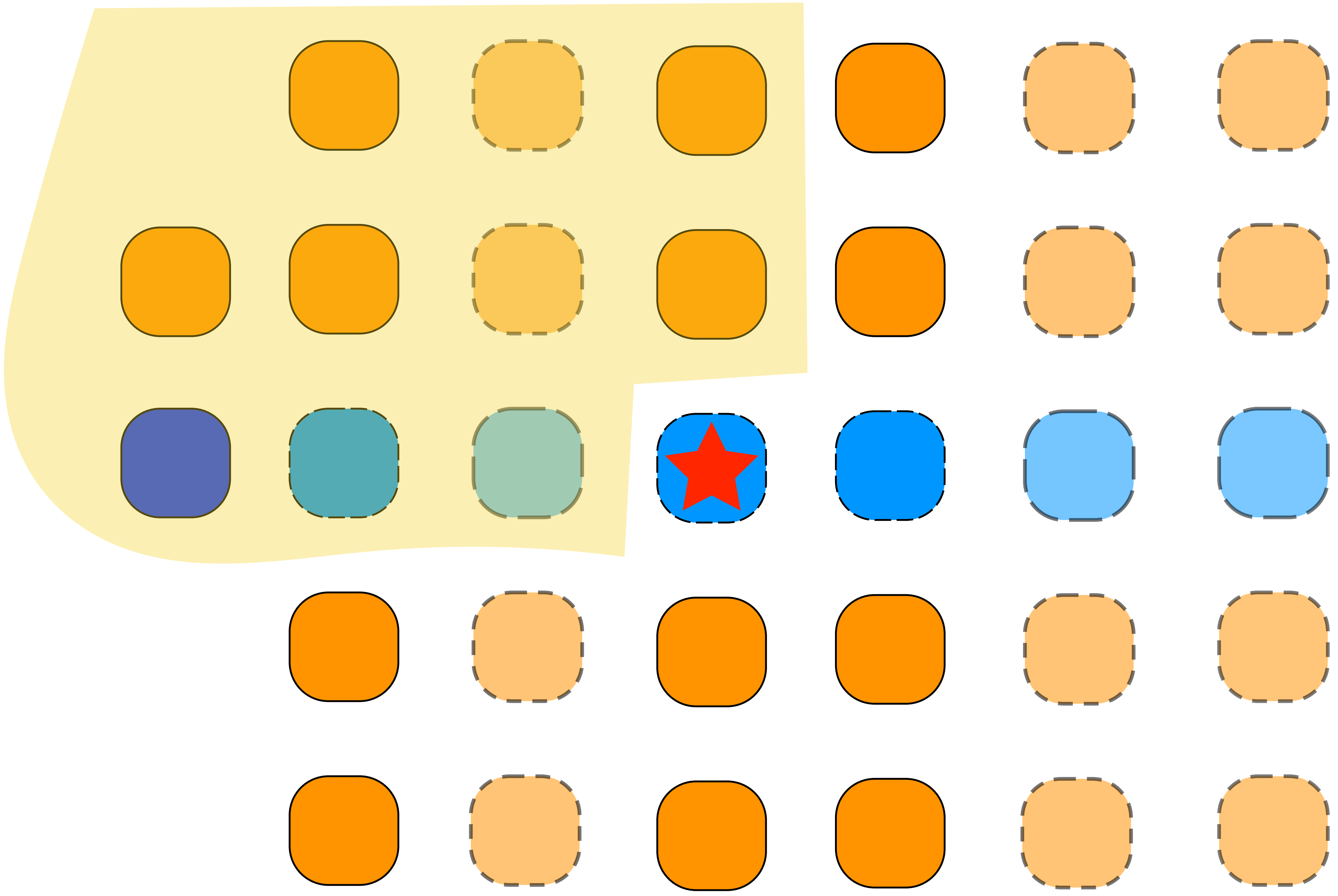
Imagine rearranging the elements by sorting each column and then also sorting the medians.

SWITCH TO A BIGGER EXAMPLE



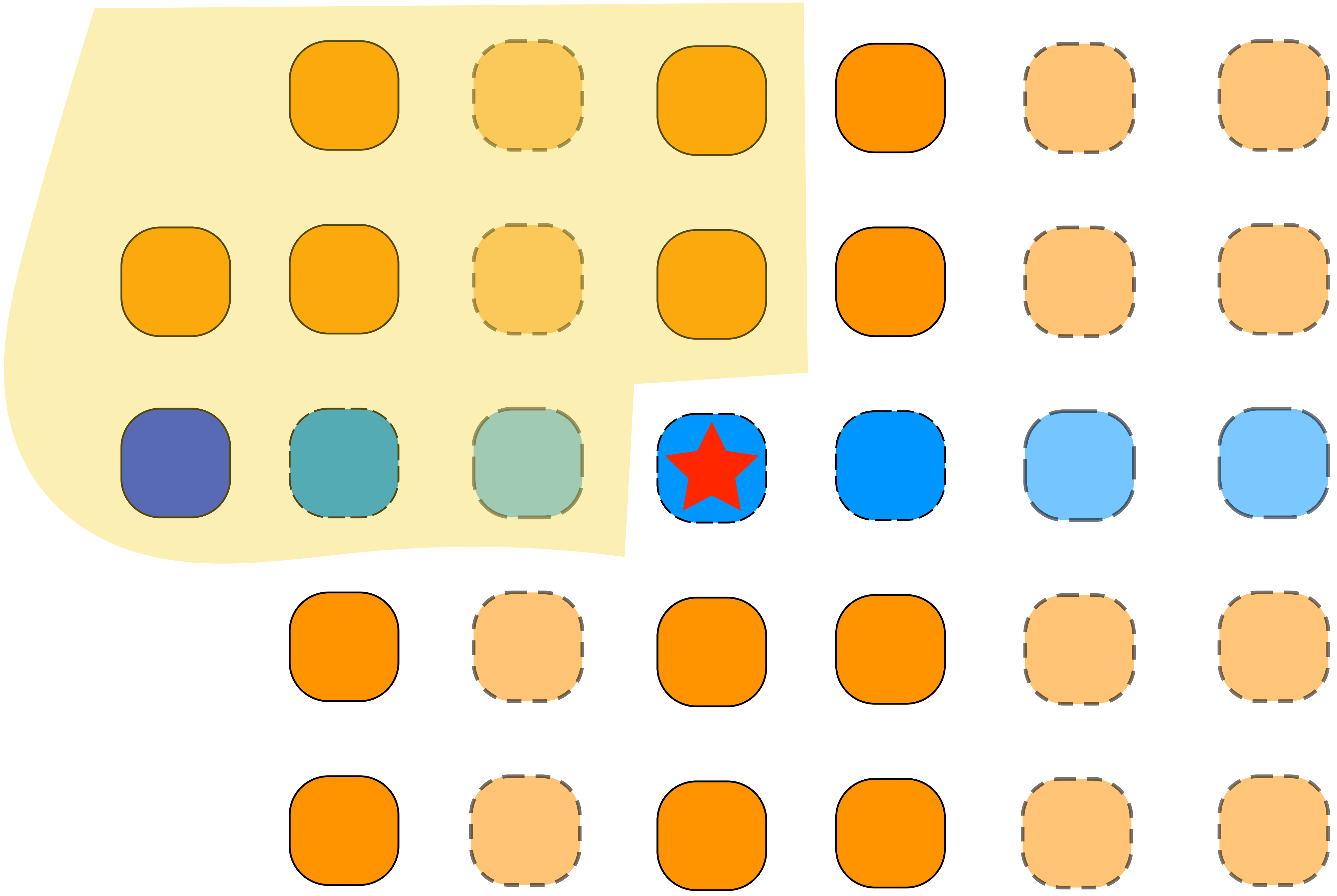
SWITCH TO A BIGGER EXAMPLE

These yellow elements are all smaller than the median. How many are there?



SWITCH TO A BIGGER EXAMPLE

These yellow elements are all smaller than the median. How many are there?



$$3\left(\left\lceil \frac{1}{2}\lceil n/5\rceil \right\rceil - 2\right)$$

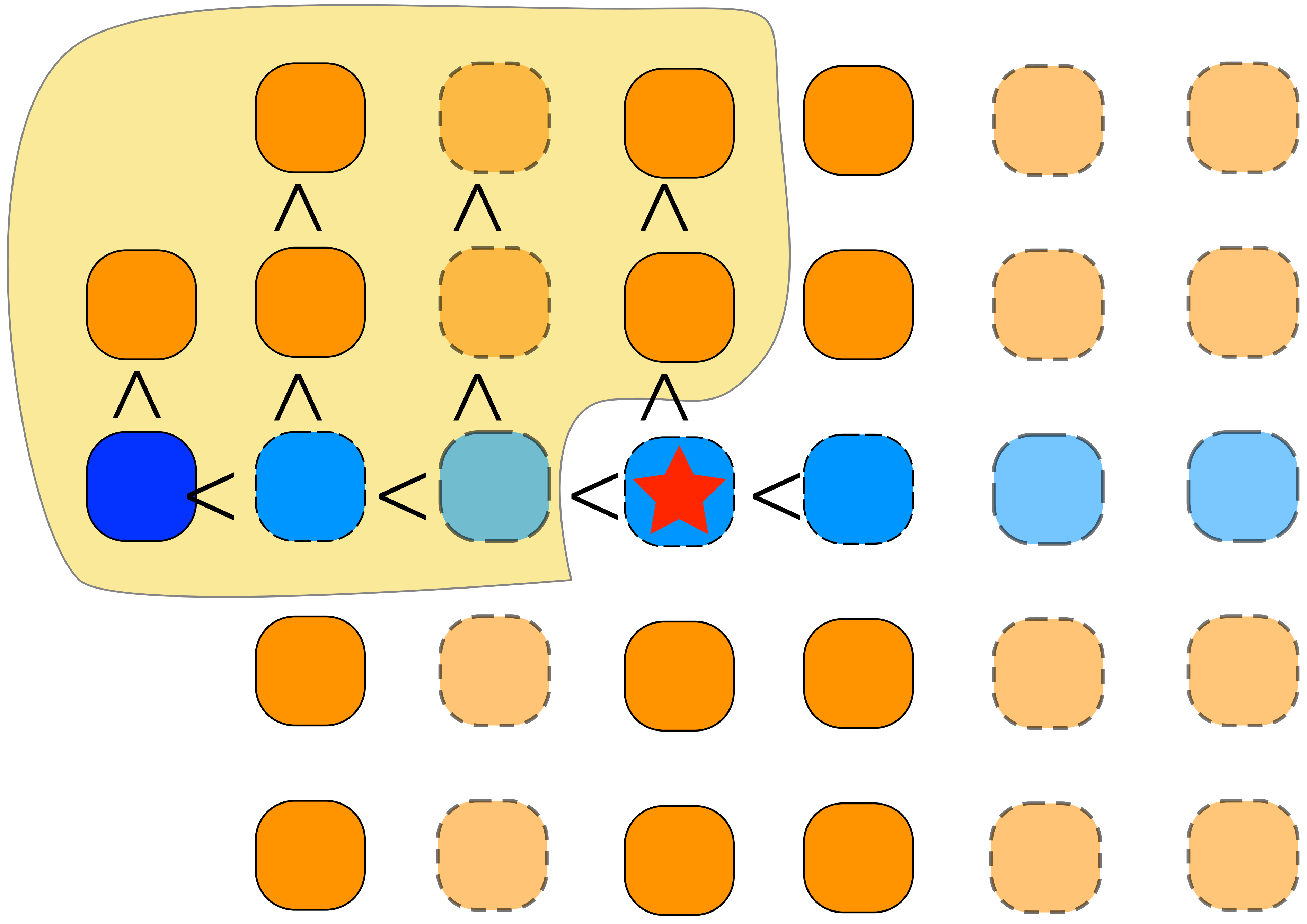
$$\geq \frac{3n}{10} - 6$$

There are $\lceil n/5 \rceil /2$ columns. Ignoring the first and last, each column has 3 elements in it that are smaller than the median.

a nice property of our partition

$$3\left(\left\lceil \frac{1}{2}\lceil n/5\rceil \right\rceil - 2\right)$$

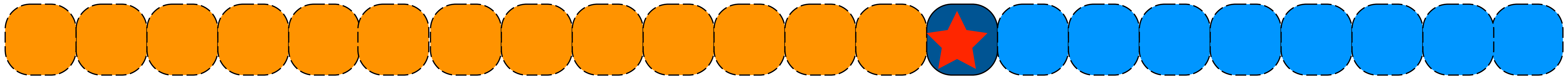
$$\geq \frac{3n}{10} - 6$$

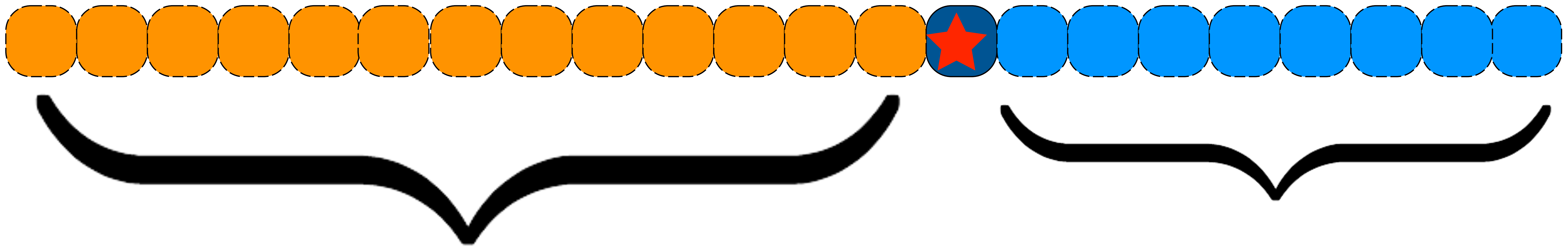


this implies there are
at most $\frac{7n}{10} + 6$ numbers

larger than ★
/smaller

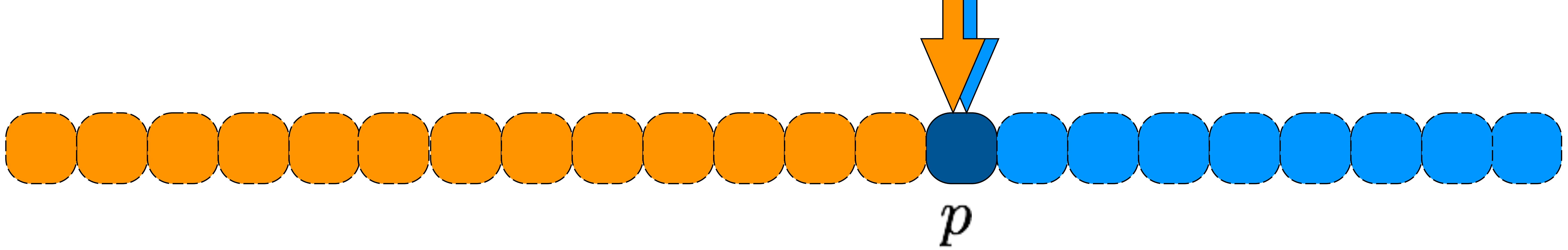
a nice property of our partition





$$\leq \frac{7n}{10} + 6 \qquad \leq \frac{7n}{10} + 6$$

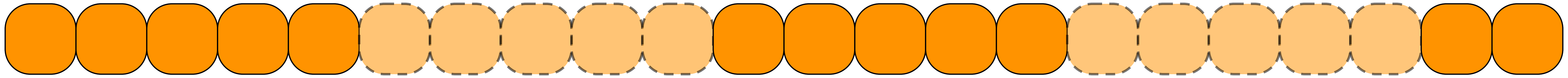
The median-of-medians is guaranteed to have a **linear fraction** of the input that is smaller and larger than it.



select $(i, A[1, \ldots, n])$

handle base case for small list
else pivot = FindPartitionValue(A,n)
partition list about pivot
if pivot is position **i**, return pivot
else if pivot is in position > **i** select $(i, A[1, \ldots, p-1])$
else select $((i-p-1), A[p+1, \ldots, n])$

FindPartition $(A[1, \ldots, n])$



divide list into groups of 5 elements

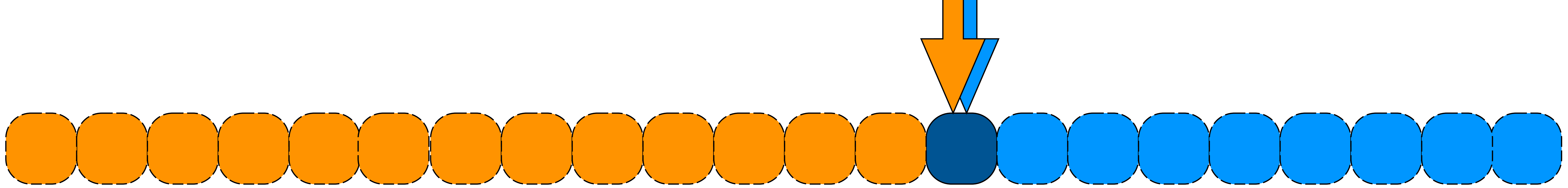
find median of each small list

gather all medians

call select(...) on this sublist to find median

return the result

$$P(n) = S(\lceil n/5 \rceil) + O(n)$$



p

select $(i, A[1, \ldots, n])$

handle base case for small list
else pivot = FindPartitionValue(A,n)
partition list about pivot
if pivot is position **i**, return pivot
else if pivot is in position > **i** select $(i, A[1, \ldots, p-1])$
else select $((i-p-1), A[p+1, \ldots, n])$

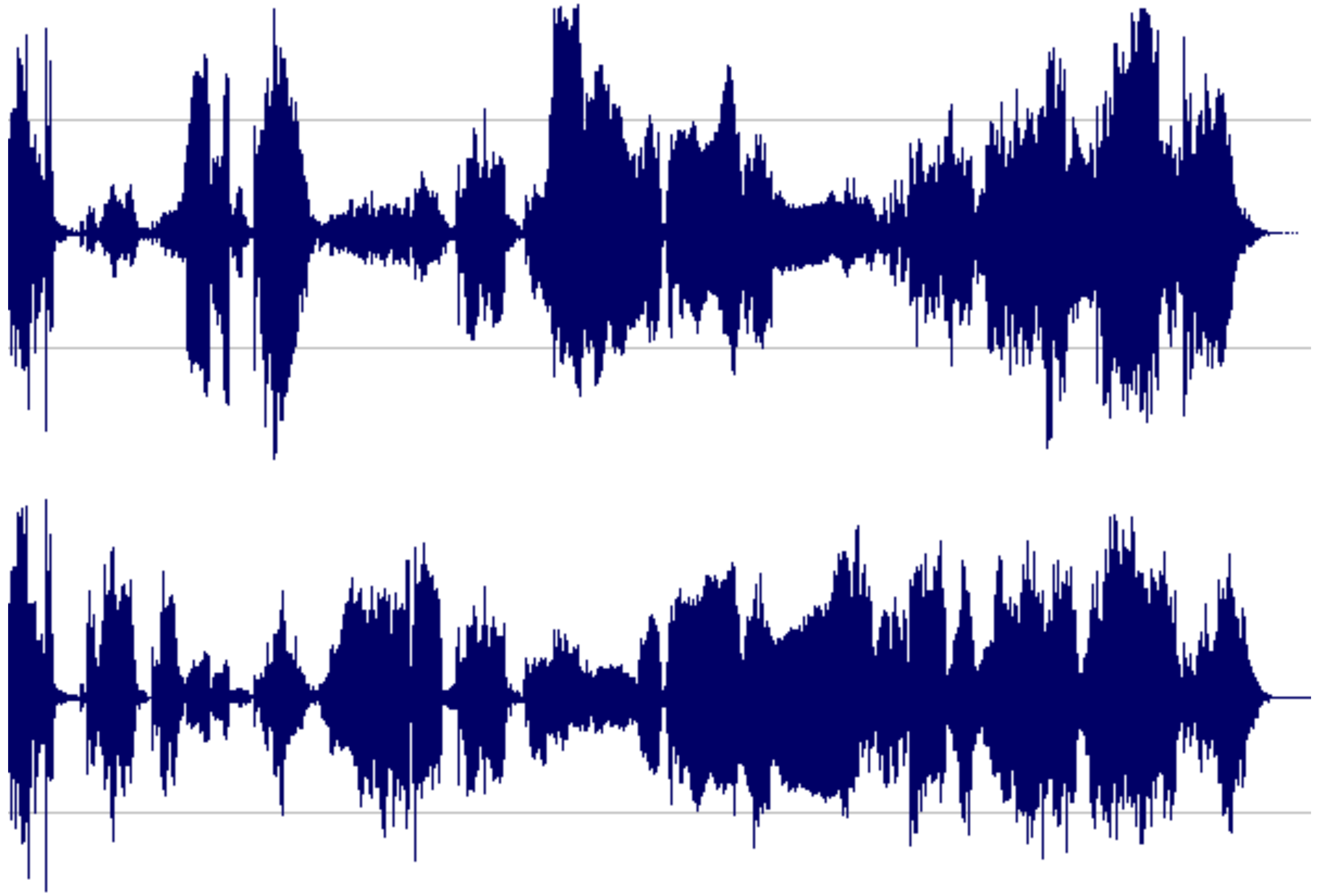
$$S(n) = S(\lceil n/5 \rceil) + \Theta(n) + S(\lceil 7n/10 + 6 \rceil)$$

$$\Theta(n)$$

You can use induction like in the homework problem.

How to get intuition for S(n)

Fast Fourier Transform



Fourier transforms are used in signals processing and EE.
We are going to present a CS interpretation of the technique.

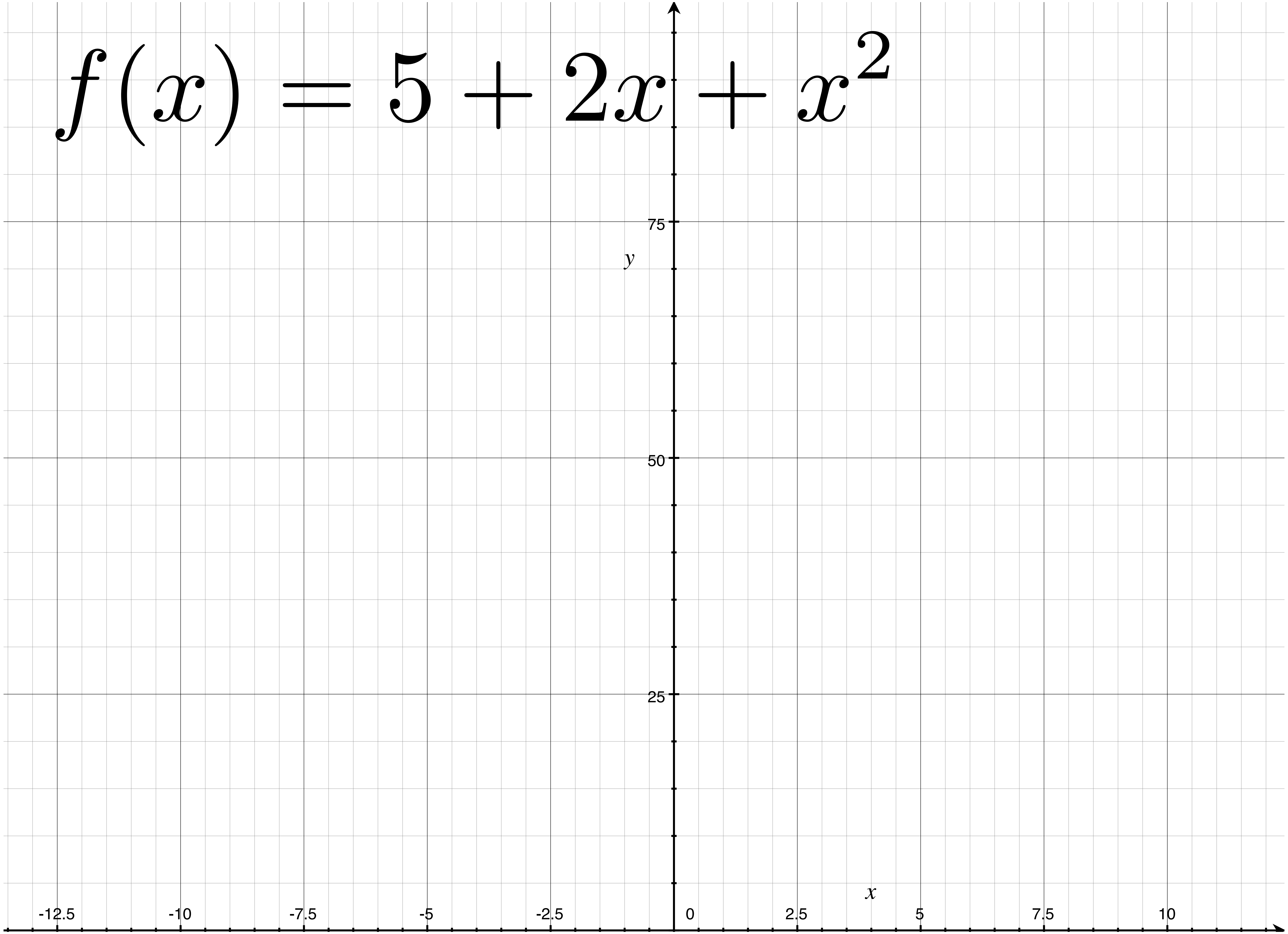
big ideas:

big ideas:

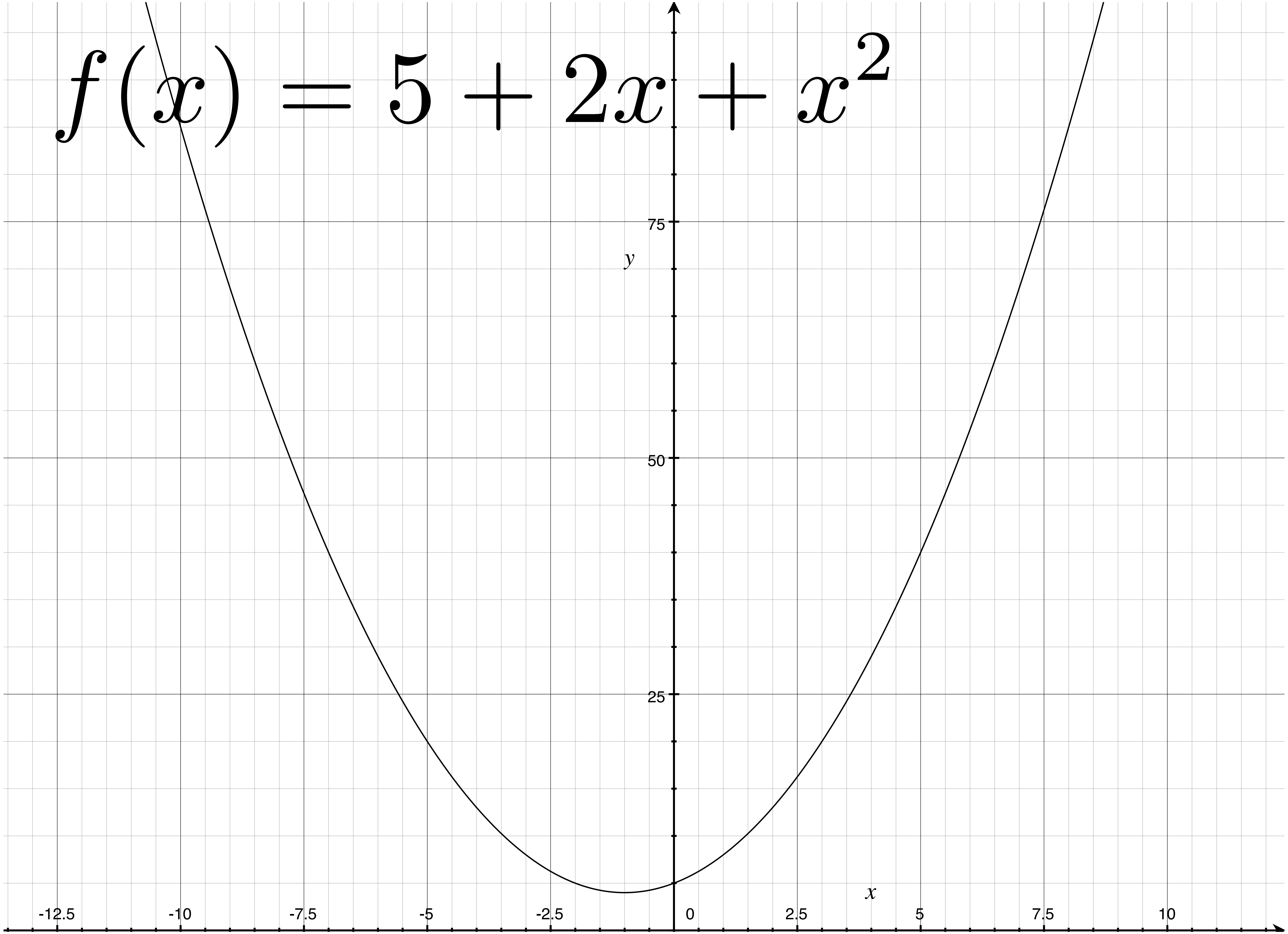
1. Changing representation from polynomial (coefficient form) into polynomial (point-wise form)

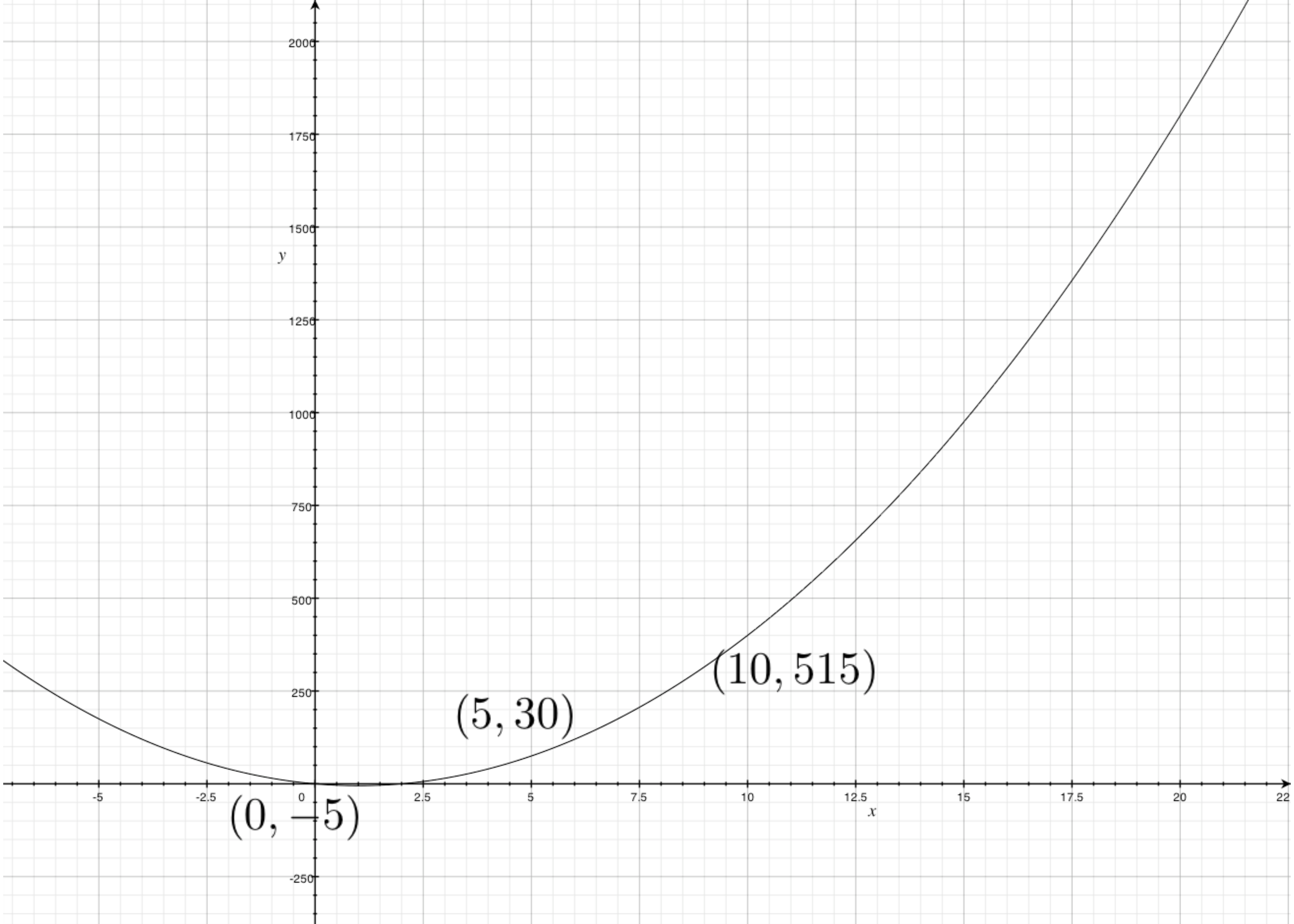
2. Clever divide and conquer

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



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





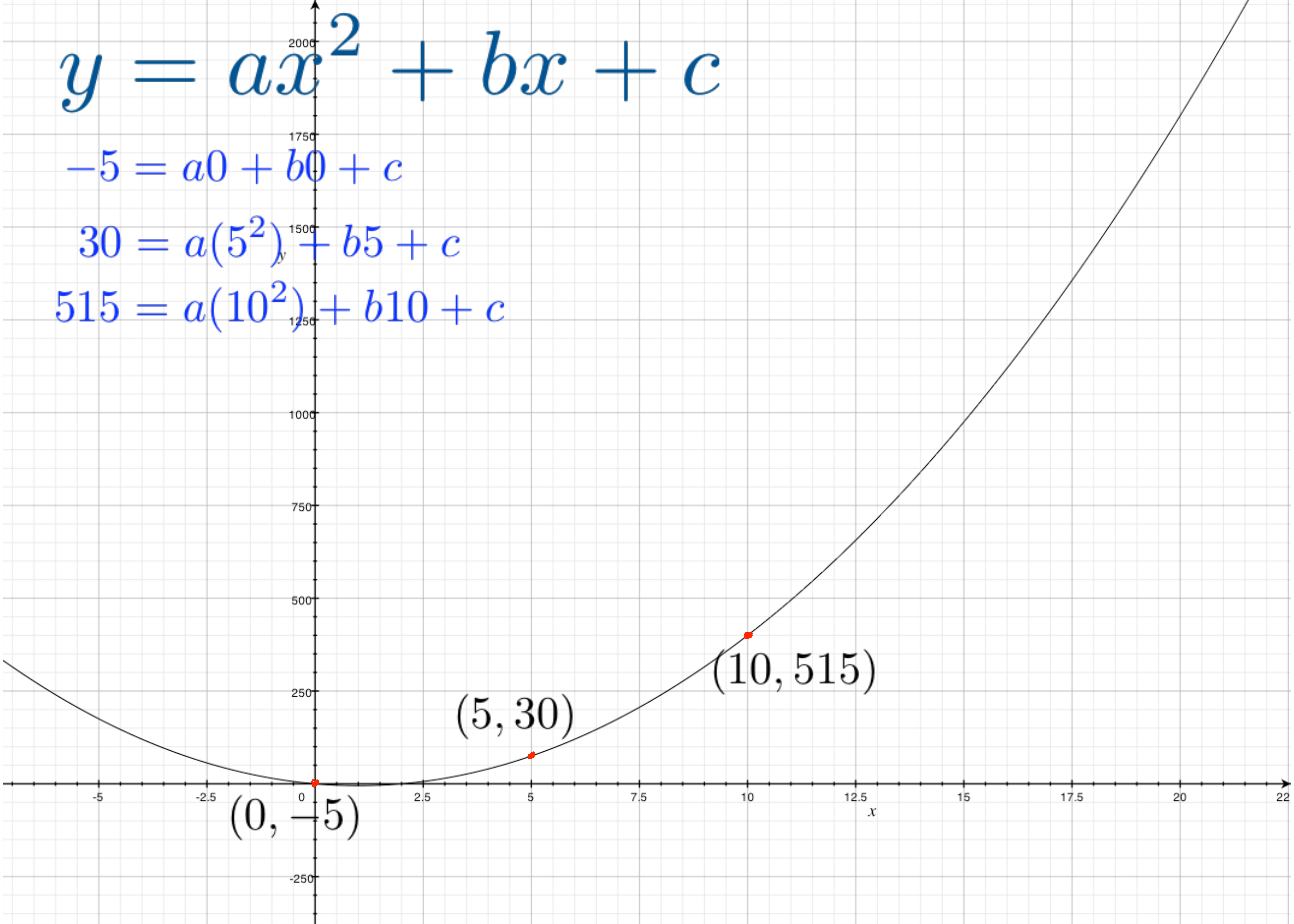
2000
1750
1500
y
1250
1000
750
500
250
-250
-5
-2.5
0
2.5
5
7.5
10
12.5
15
17.5
20
22
x
$(10, 515)$
$(5, 30)$
$(0, -5)$

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

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

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

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



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

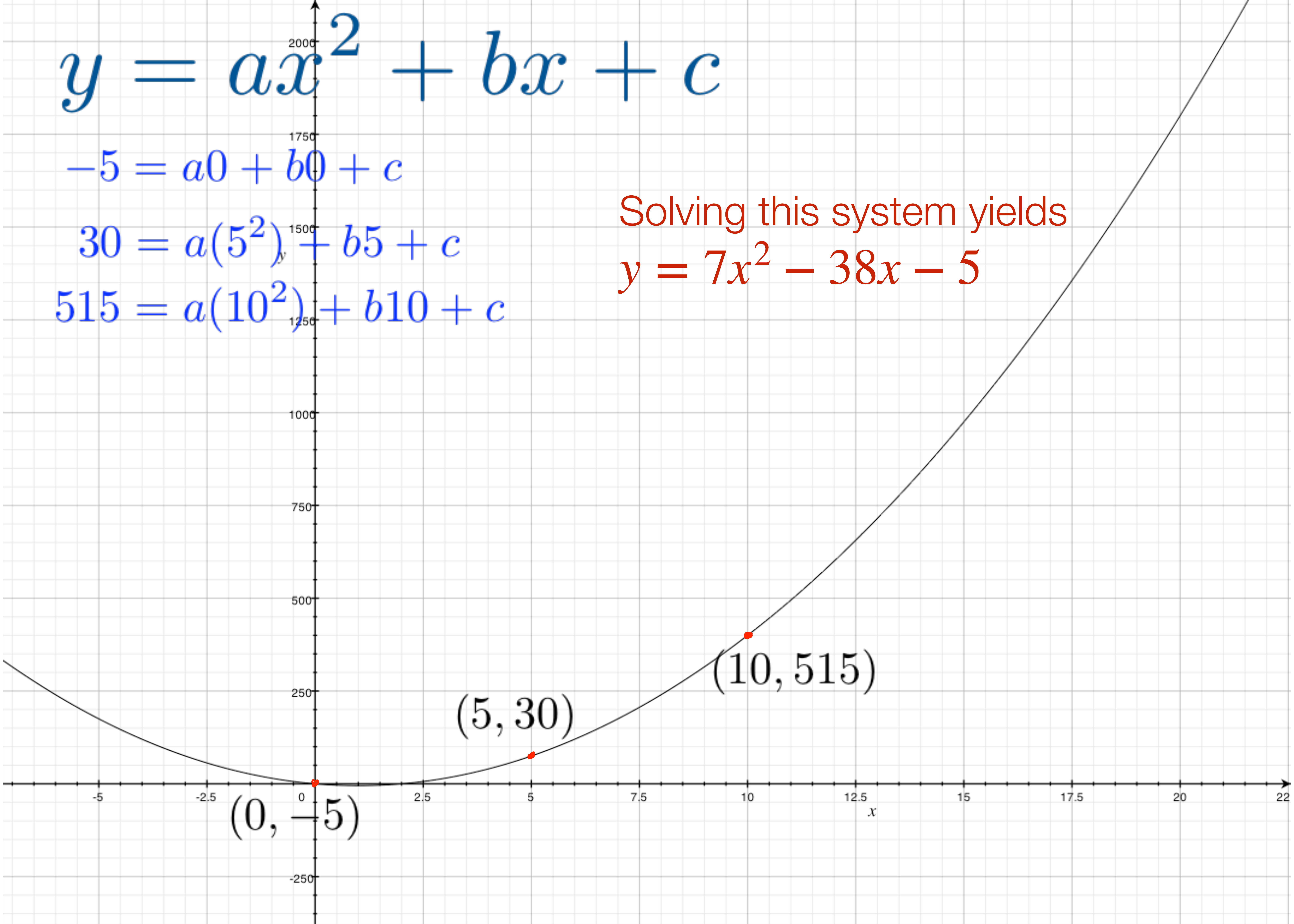
$$-5 = a0 + b0 + c$$

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

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

Solving this system yields

$$y = 7x^2 - 38x - 5$$

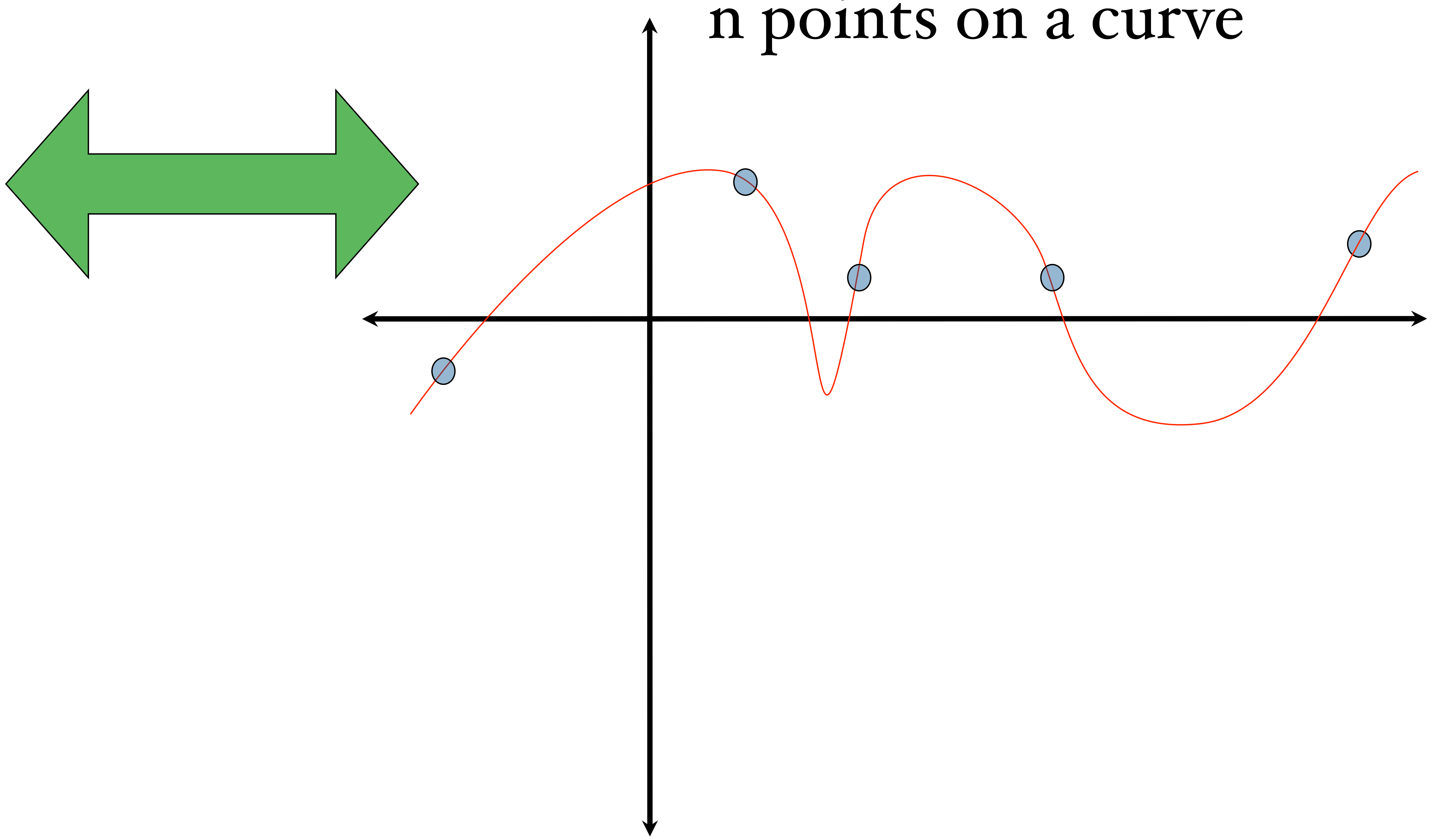


$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

This is a polynomial. Its standard representation is given by its coefficients.

degree $n-1$
polynomial

$$A(x)$$



Two ways to represent a polynomial.

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:

FFT

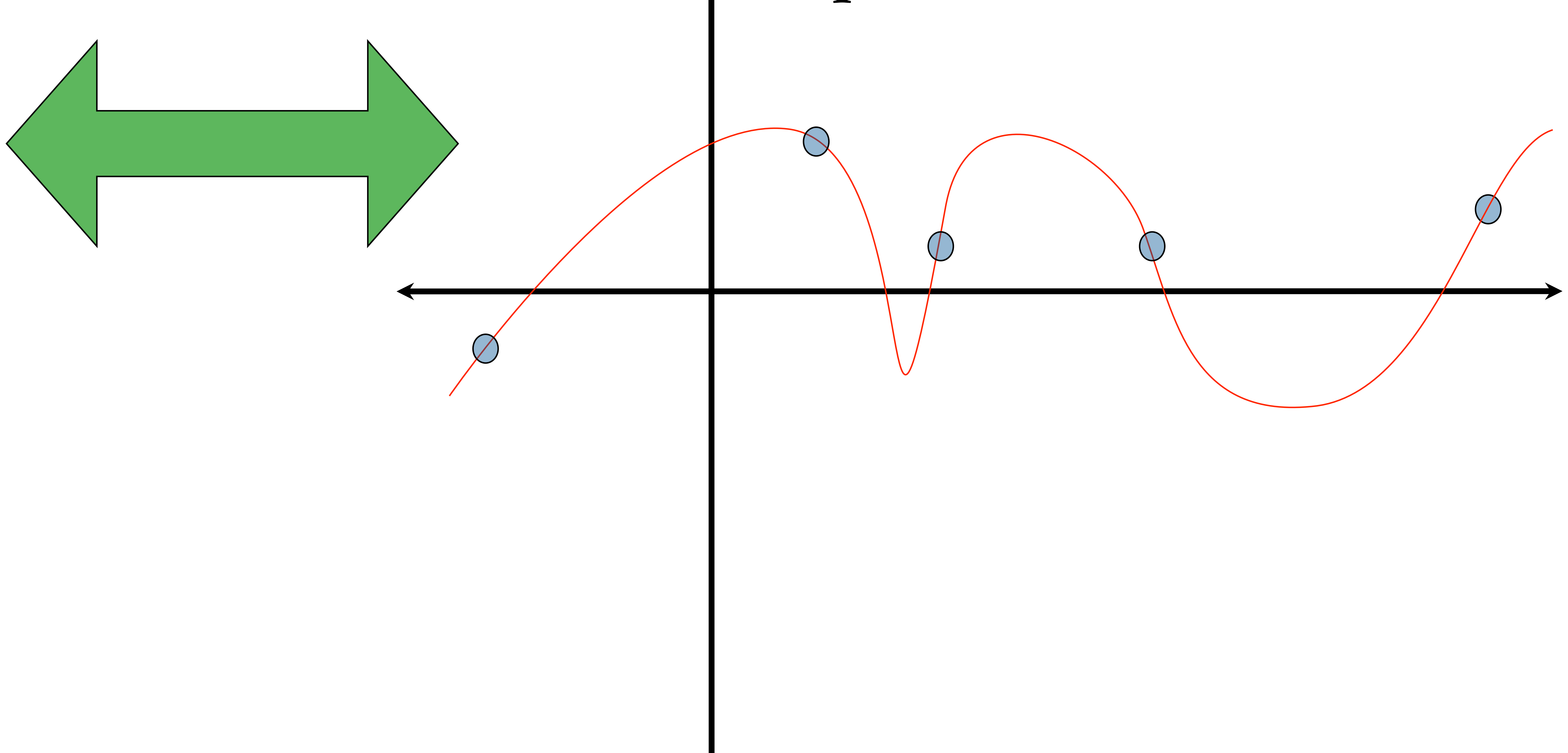
input: $a_0, a_1, a_2, \ldots, a_{n-1}$

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

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

n points on a curve

$$A(x)$$



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

Inverse FFT: Given n-points,

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

$$y_i = A(\omega_i)$$

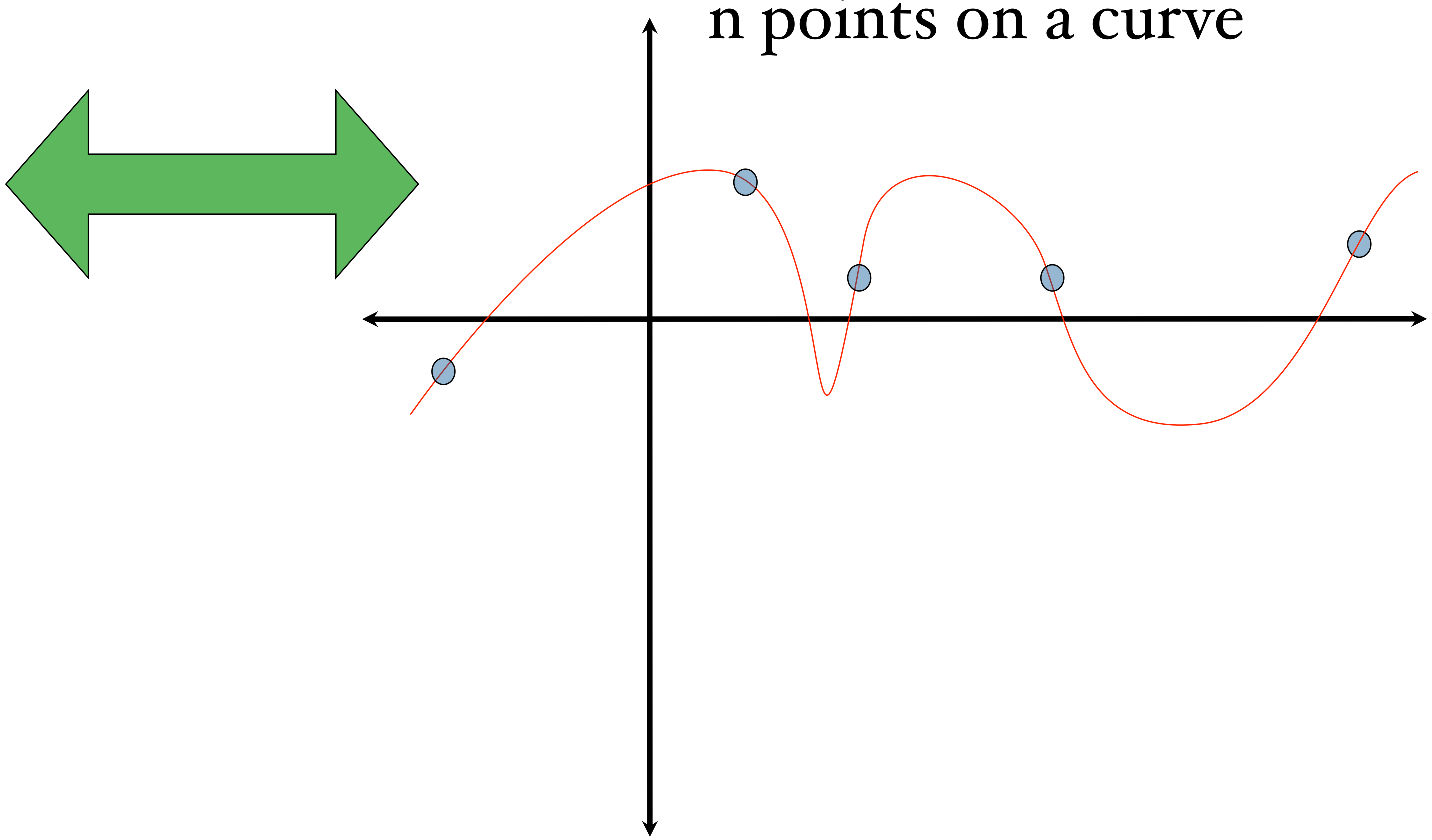
FFT

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.

$$A(x)$$



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

Brute force method to evaluate A at n points:

solve the large problem by

solving smaller problems
and combining solutions

$T(n)=$

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

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

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

$$+ a_1 x + a_3 x^3 + a_5 x^5 + \cdots + a_{n-1} x^{n-1}$$

$$A_e(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_n x^{(n-2)/2}$$

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

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

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

suppose we had already had eval of A_e, A_o on the values {4,9,16,25}

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

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

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

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

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)

What we need

We could compute

$$A(2)$$
$$A(-2)$$
$$A(3)$$
$$A(-3)$$
$$A(4)$$
$$A(-4)$$
$$A(5)$$
$$A(-5)$$

8, degree n

What we need

We could compute	If we had…
$A(2)$	$A_e(4), A_o(4)$
$A(-2)$	$A_e(9), A_o(9)$
$A(3)$	$A_e(16), A_o(16)$
$A(-3)$	$A_e(25), A_o(25)$
$A(4)$	
$A(-4)$	
$A(5)$	
$A(-5)$	
8, degree n	8 degree $n/2$

What we need

We could compute	If we had…	Which we could compute	If we had…
$A(2)$	$A_e(4), A_o(4)$		
$A(-2)$	$A_e(9), A_o(9)$		
$A(3)$	$A_e(16), A_o(16)$		
$A(-3)$	$A_e(25), A_o(25)$		
$A(4)$			
$A(-4)$			
$A(5)$			
$A(-5)$			
8, degree n	8 degree $n/2$		

What we need

We could compute	If we had…	Which we could compute	If we had…
$A(2)$	$A_e(4), A_o(4)$		
$A(-2)$	$A_e(9), A_o(9)$		$A_{ee}(16), A_{eo}(16), A_{oe}(16), A_{oo}(16)$
$A(3)$	$A_e(16), A_o(16)$		$A_{ee}(81), A_{eo}(81), A_{oe}(81), A_{oo}(81)$
$A(-3)$	$A_e(25), A_o(25)$		$A_{ee}(256), A_{eo}(256), A_{oe}(256), A_{oo}(256)$
$A(4)$			$A_{ee}(625), A_{eo}(625), A_{oe}(625), A_{oo}(625)$
$A(-4)$			
$A(5)$			
$A(-5)$			
8, degree n	8 degree $n/2$		16 degree $n/4$

What we need

We could compute	If we had…	Which we could compute	If we had…
$A(2)$	$A_e(4), A_o(4)$		$A_{ee}(16), A_{eo}(16), A_{oe}(16), A_{oo}(16)$
$A(-2)$	$A_e(9), A_o(9)$		$A_{ee}(81), A_{eo}(81), A_{oe}(81), A_{oo}(81)$
$A(3)$	$A_e(16), A_o(16)$		$A_{ee}(256), A_{eo}(256), A_{oe}(256), A_{oo}(256)$
$A(-3)$	$A_e(25), A_o(25)$		$A_{ee}(625), A_{eo}(625), A_{oe}(625), A_{oo}(625)$
$A(4)$			
$A(-4)$			
$A(5)$			
$A(-5)$			
8, degree n	8 degree $n/2$		16 degree $n/4$

We need a better way to pick the points.
The FFT uses the *roots of unity*.

Roots of unity

$$x^n = 1$$

should have n solutions

what are they?

Remember this?

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

$$x^n = 1$$

the n solutions are:

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

$$x^n = 1$$

the n solutions are:

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 n^th 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

What is this number?

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

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

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

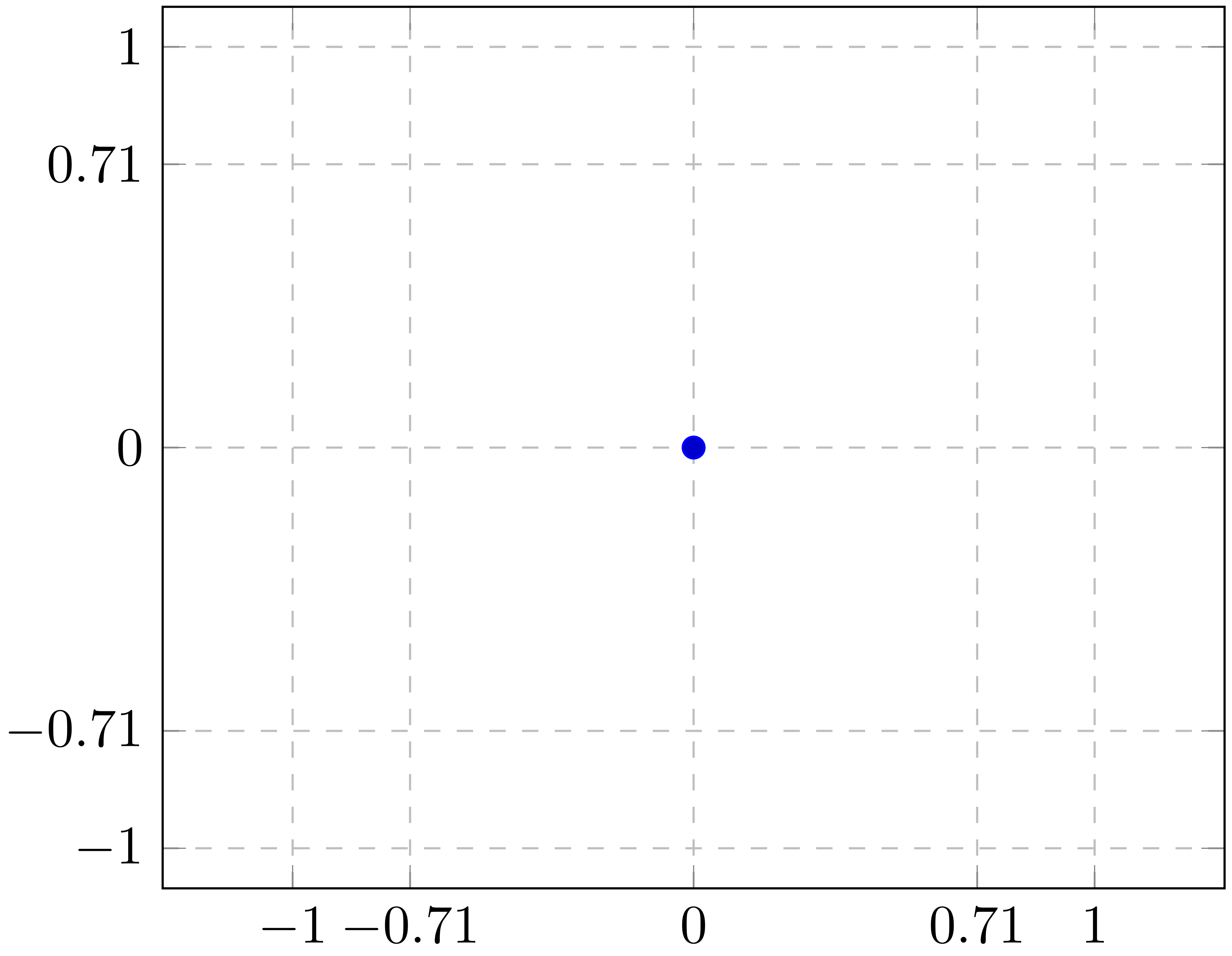
$e^{2\pi ij/n}$ = $\omega_{j,n}$ is an n[th] root of unity

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

Lets compute $\omega_{1,8}$

Compute all 8 roots of unity

ω_0 ω_1 ω_2 ω_3 ω_4 ω_5 ω_6 ω_7



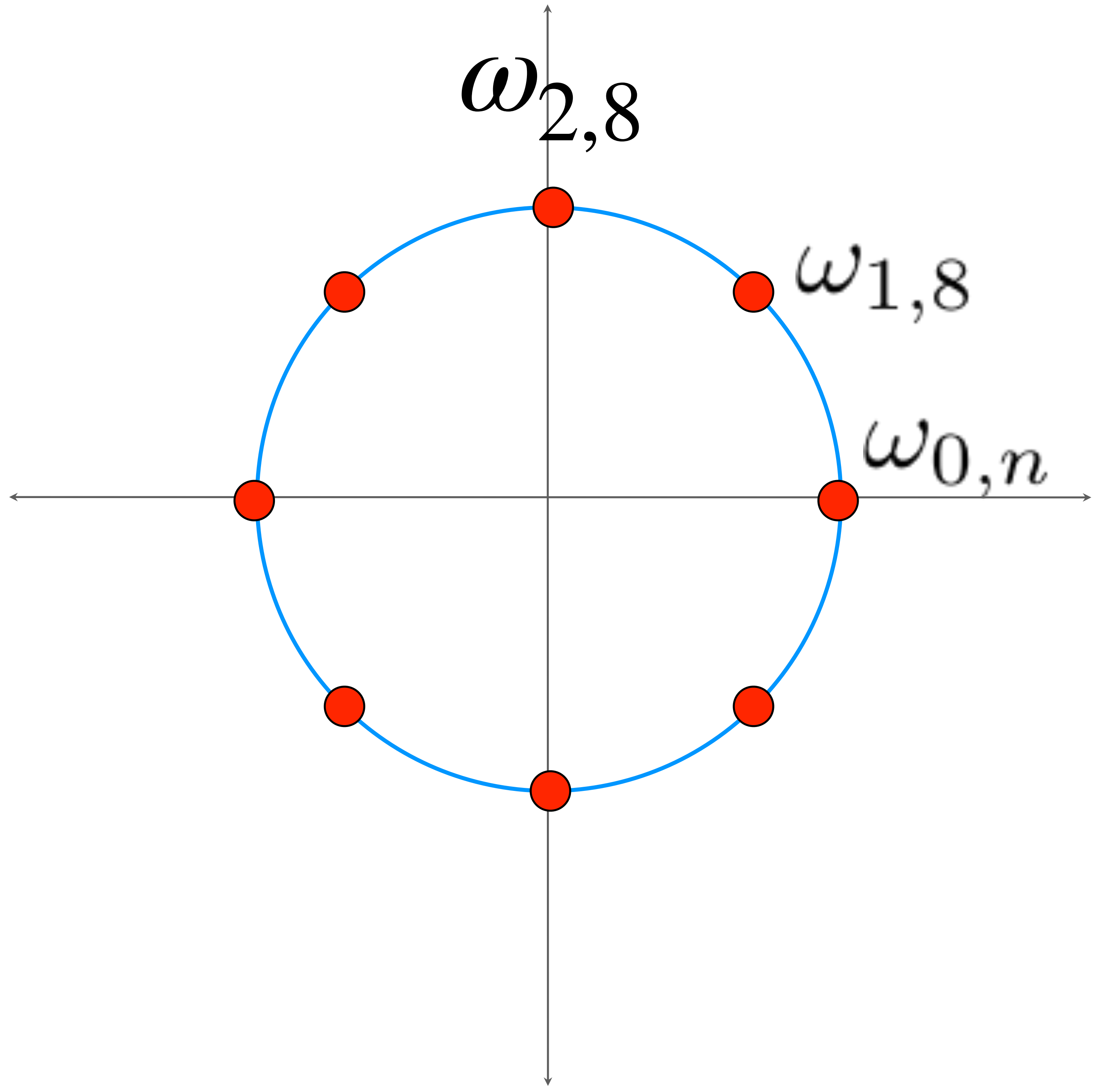
Then graph them

roots of unity

$$x^n = 1$$

should have n solutions

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

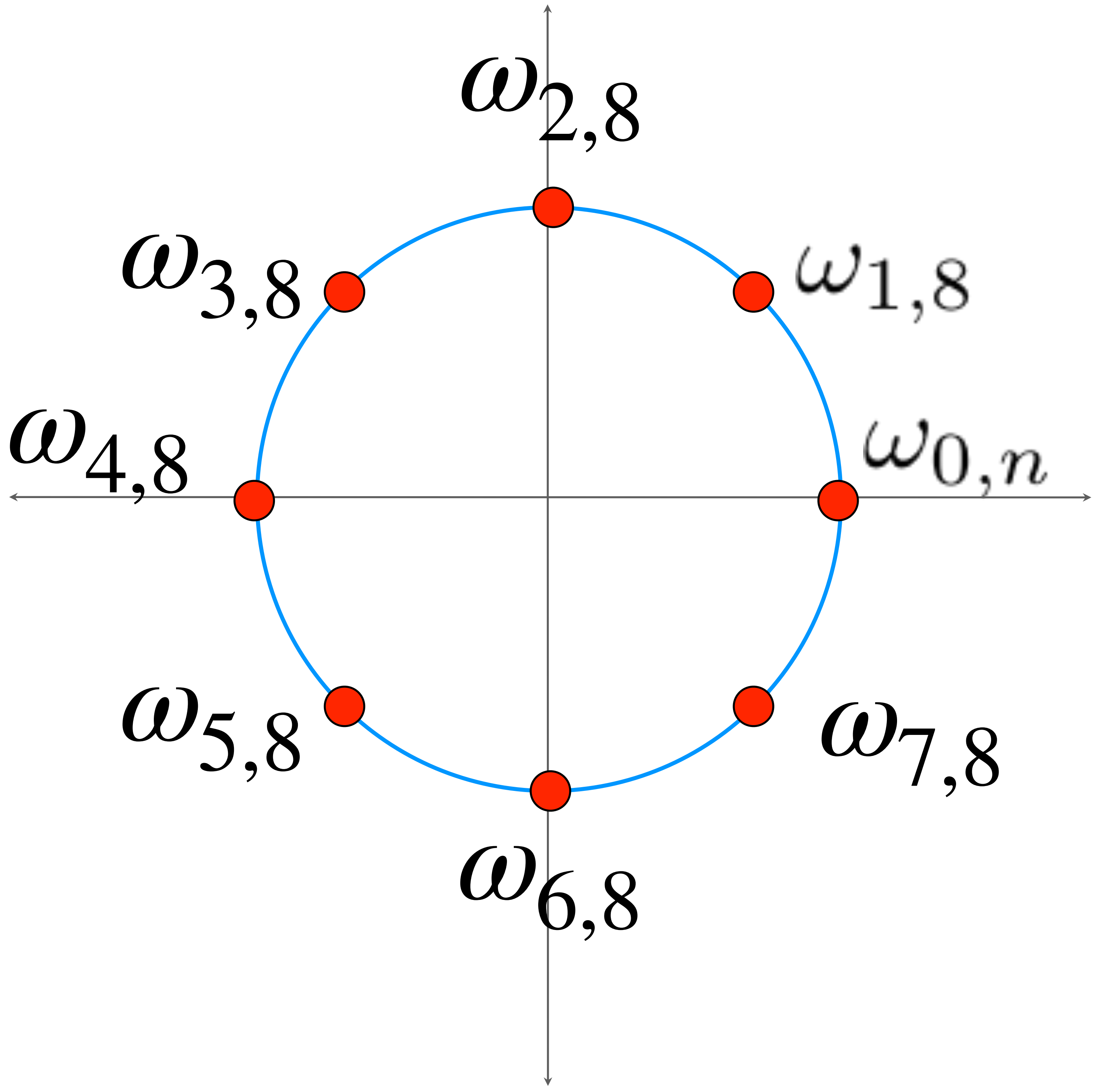


roots of unity

$$x^n = 1$$

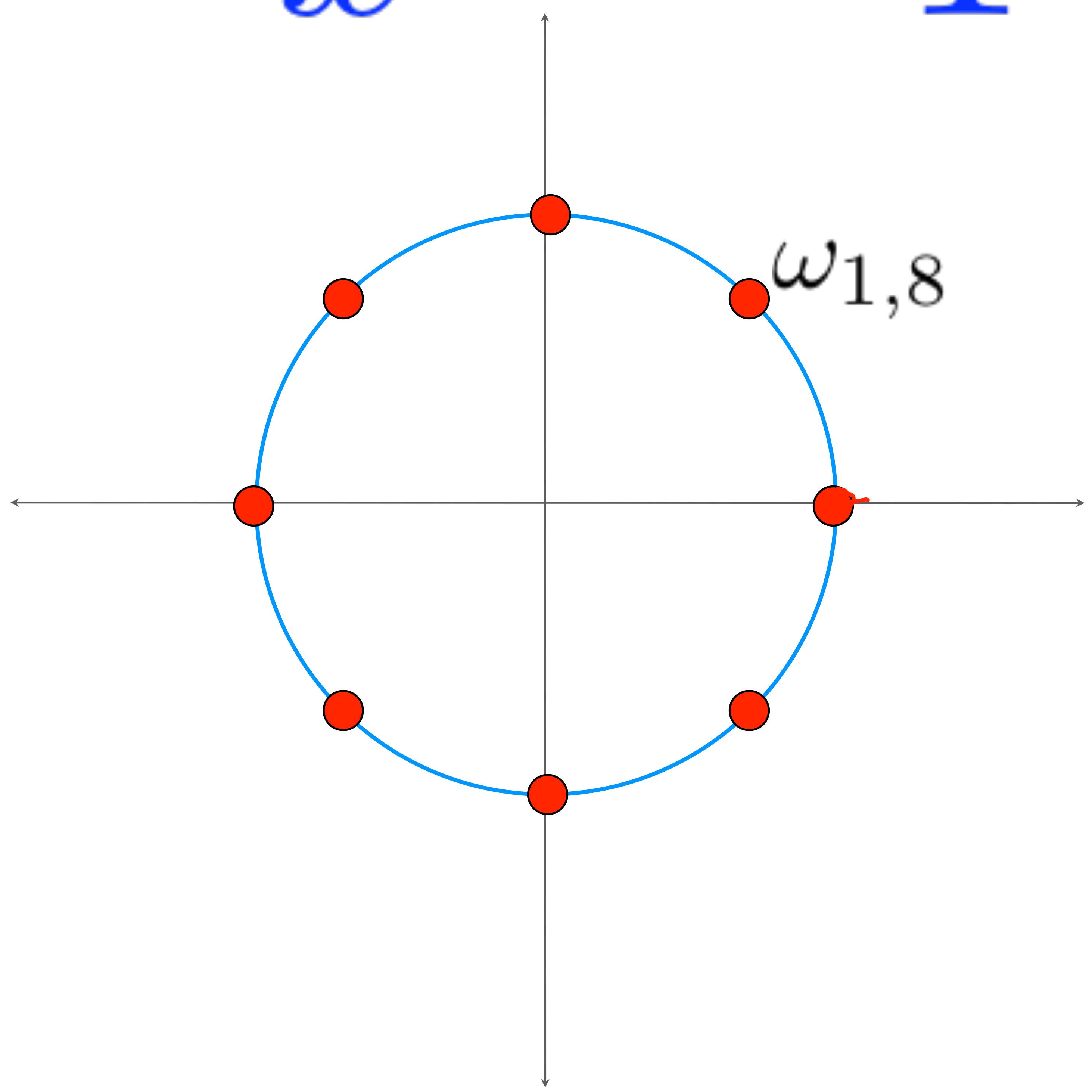
should have n solutions

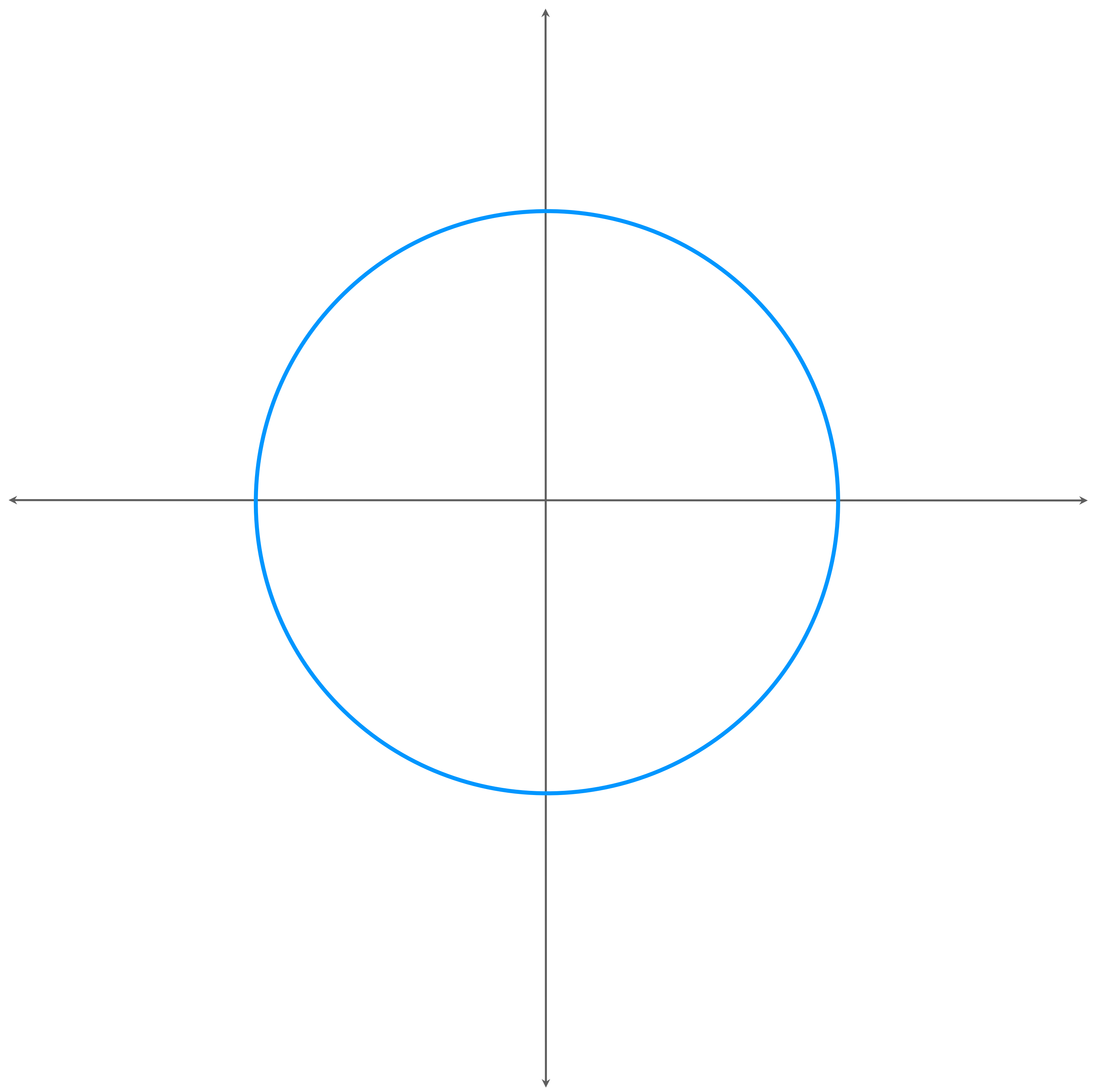
$$e^{2\pi i j/n} = \cos(2\pi j/n) + i \sin(2\pi j/n)$$



Squaring the nth roots of unity

$$x^n = 1$$





ω_0	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7
1	$\frac{1}{\sqrt{2}}+\frac{i}{\sqrt{2}}$	i	$-\frac{1}{\sqrt{2}}+\frac{i}{\sqrt{2}}$	-1	$-\frac{1}{\sqrt{2}}-\frac{i}{\sqrt{2}}$	$-i$	$\frac{1}{\sqrt{2}}-\frac{i}{\sqrt{2}}$

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

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

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

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

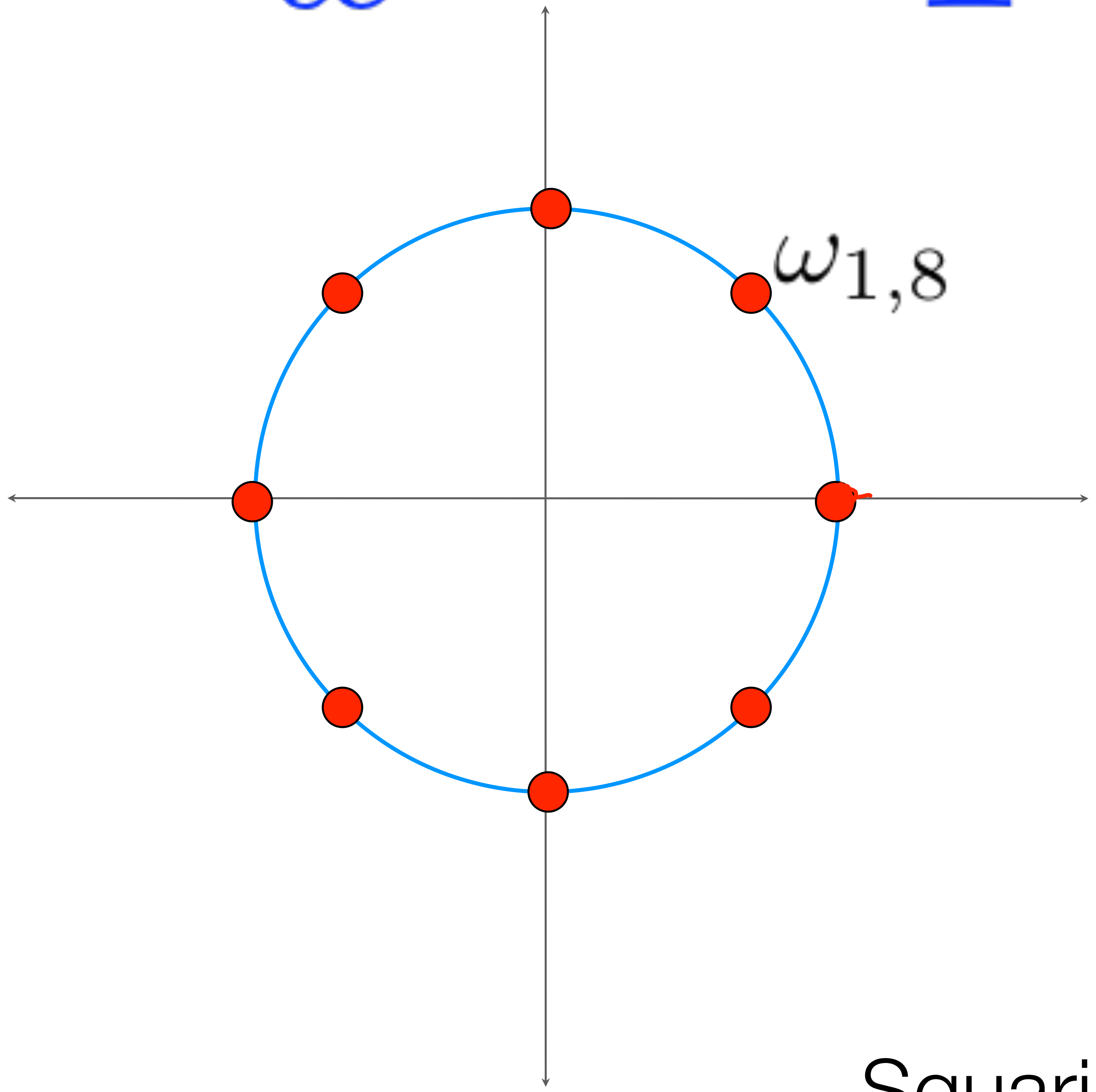
$$\omega_{1,8}^2 = \left(\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right)^2 = \left(\frac{1}{\sqrt{2}}\right)^2 + 2\left(\frac{1}{\sqrt{2}}\frac{i}{\sqrt{2}}\right) + \left(\frac{i}{\sqrt{2}}\right)^2$$

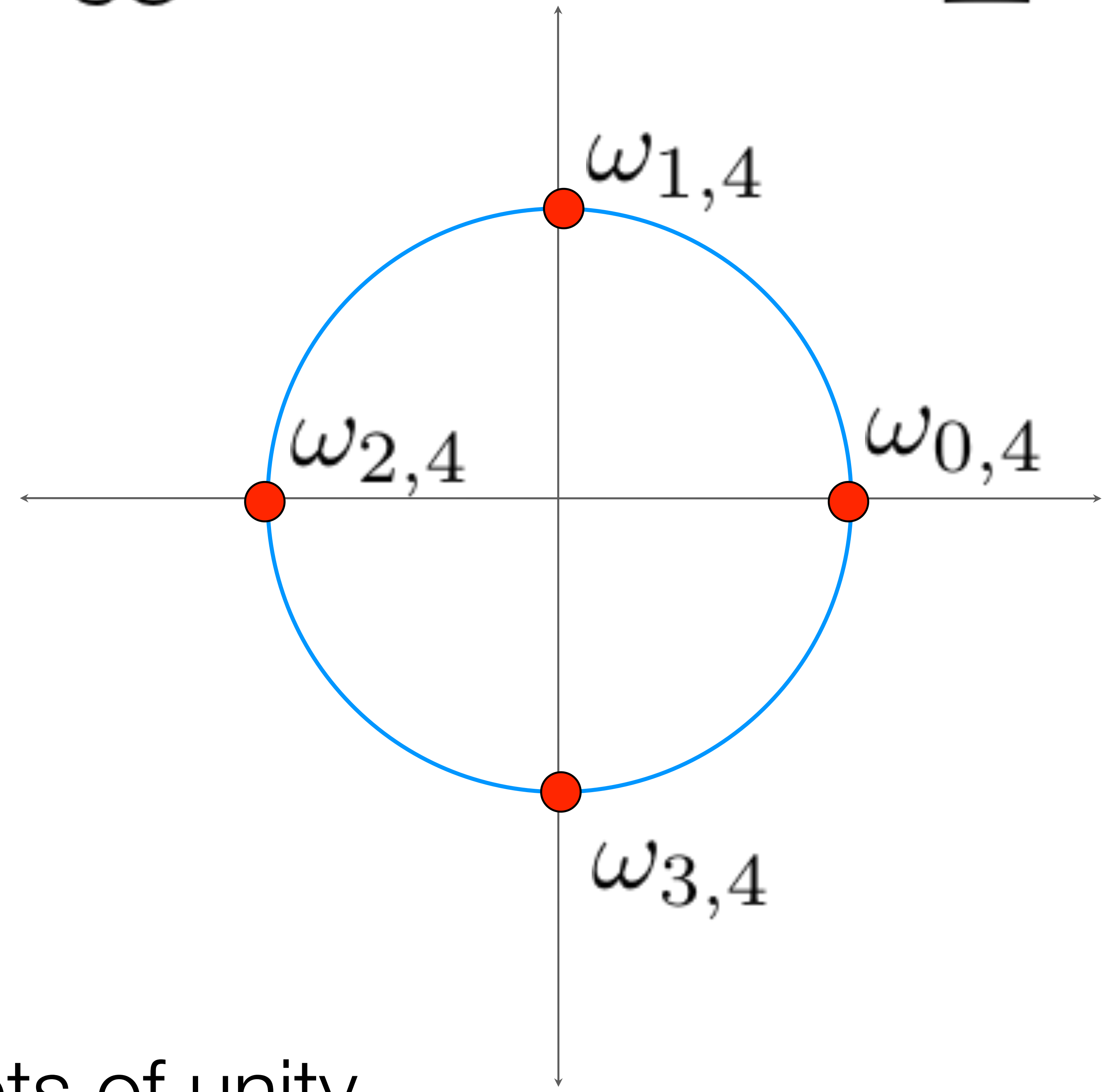
$$= 1/2 + i - 1/2$$

$$= i$$

Squaring the nth roots of unity

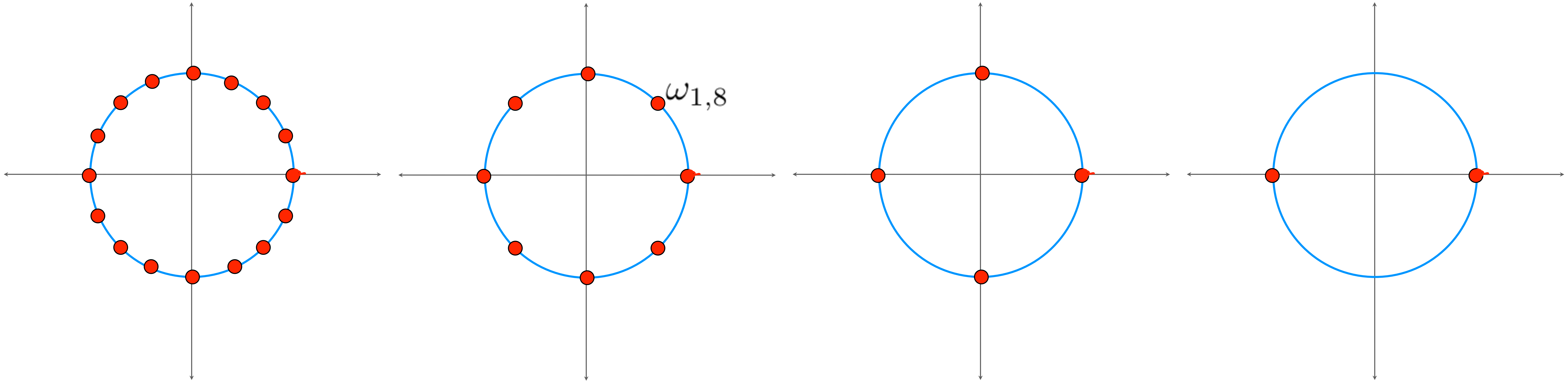
$x^n = 1$ $\qquad\qquad$ $x^{n/2} = 1$





Squaring all of the nth roots of unity
produces the n/2th roots of unity

If n=16



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

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,...,n])

Evaluates degree **n** poly on the **nth roots of unity**

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

Evaluates degree **n** poly on the **nth roots of unity**

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

For 1..n, 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.

Example

FFT(4, 1, 3, 2, 2, 3, 1, 4)

What does this function compute?

Example

FFT(4, 1, 3, 2, 2, 3, 1, 4)

What does this function compute?

It evaluates $4 + 1x + 3x^2 + 2x^3 + 2x^4 + 3x^5 + 1x^6 + 4x^7$

on the 8th roots of unity, which are

Example

FFT(4, 1, 3, 2, 2, 3, 1, 4)

What does this function compute?

It evaluates $4 + 1x + 3x^2 + 2x^3 + 2x^4 + 3x^5 + 1x^6 + 4x^7$

on the 8th roots of unity, which are

ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8
1	$\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}$	i	$\frac{-1}{\sqrt{2}} + \frac{i}{\sqrt{2}}$	-1	$\frac{-1}{\sqrt{2}} + \frac{-i}{\sqrt{2}}$	$-i$	$\frac{1}{\sqrt{2}} + \frac{-i}{\sqrt{2}}$

$$A(x) = 4 + 1x + 3x^2 + 2x^3 + 2x^4 + 3x^5 + 1x^6 + 4x^7$$

FFT on $A(x) = 4 + 1x + 3x^2 + 2x^3 + 2x^4 + 3x^5 + 1x^6 + 4x^7$

$$A_e(x) = 4 + 3x + 2x^2 + 1x^3$$

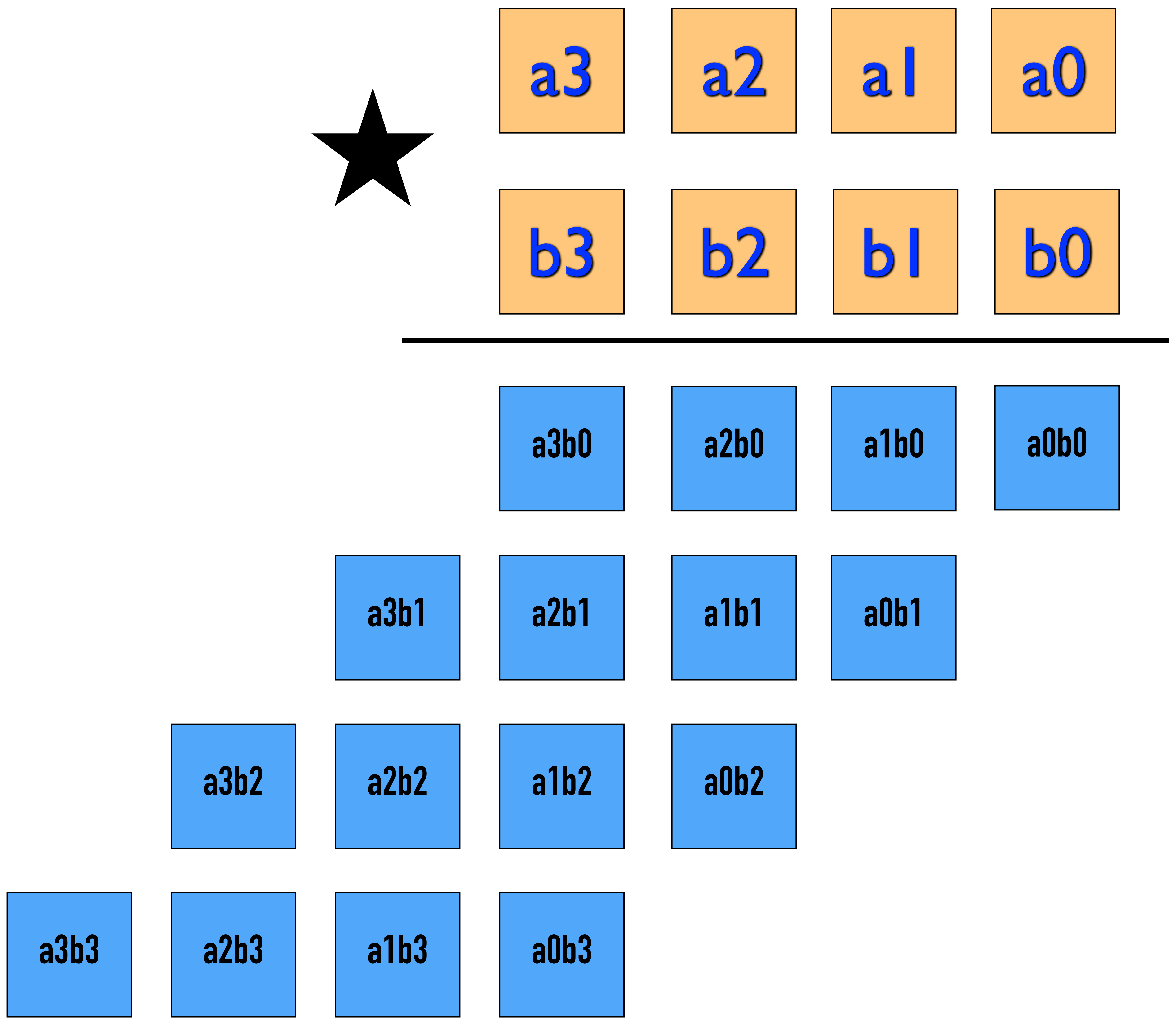
$$A_o(x) = 1 + 2x + 3x^2 + 4x^3$$

$$\mathrm{FFT}(A_e) \overset{\text{returns}}{=} \left\{ \begin{array}{cccc} 1 & i & -1 & -i \\ 10 & 2+2i & 2 & 2-2i \end{array} \right\}$$

4th roots of unity are $\{1, i, -1, -i\}$

$$\mathrm{FFT}(A_o) \overset{\text{returns}}{=} \left\{ \begin{array}{cccc} 1 & i & -1 & -i \\ 10 & -2-2i & -2 & -2+2i \end{array} \right\}$$

What can you do with the FFT?

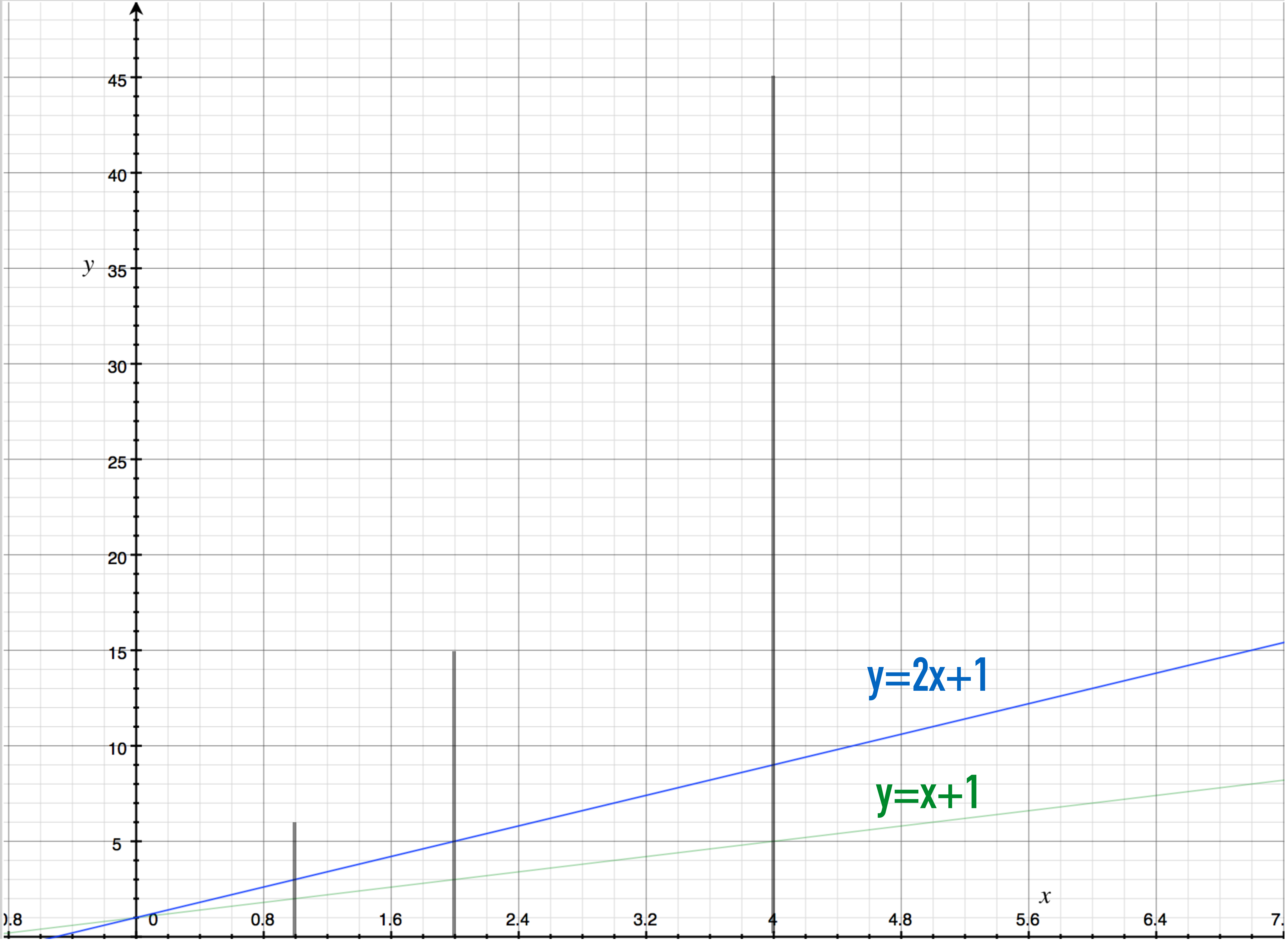

a3 a2 a1 a0
b3 b2 b1 b0
a3b0 a2b0 a1b0 a0b0
a3b1 a2b1 a1b1 a0b1
a3b2 a2b2 a1b2 a0b2
a3b3 a2b3 a1b3 a0b3

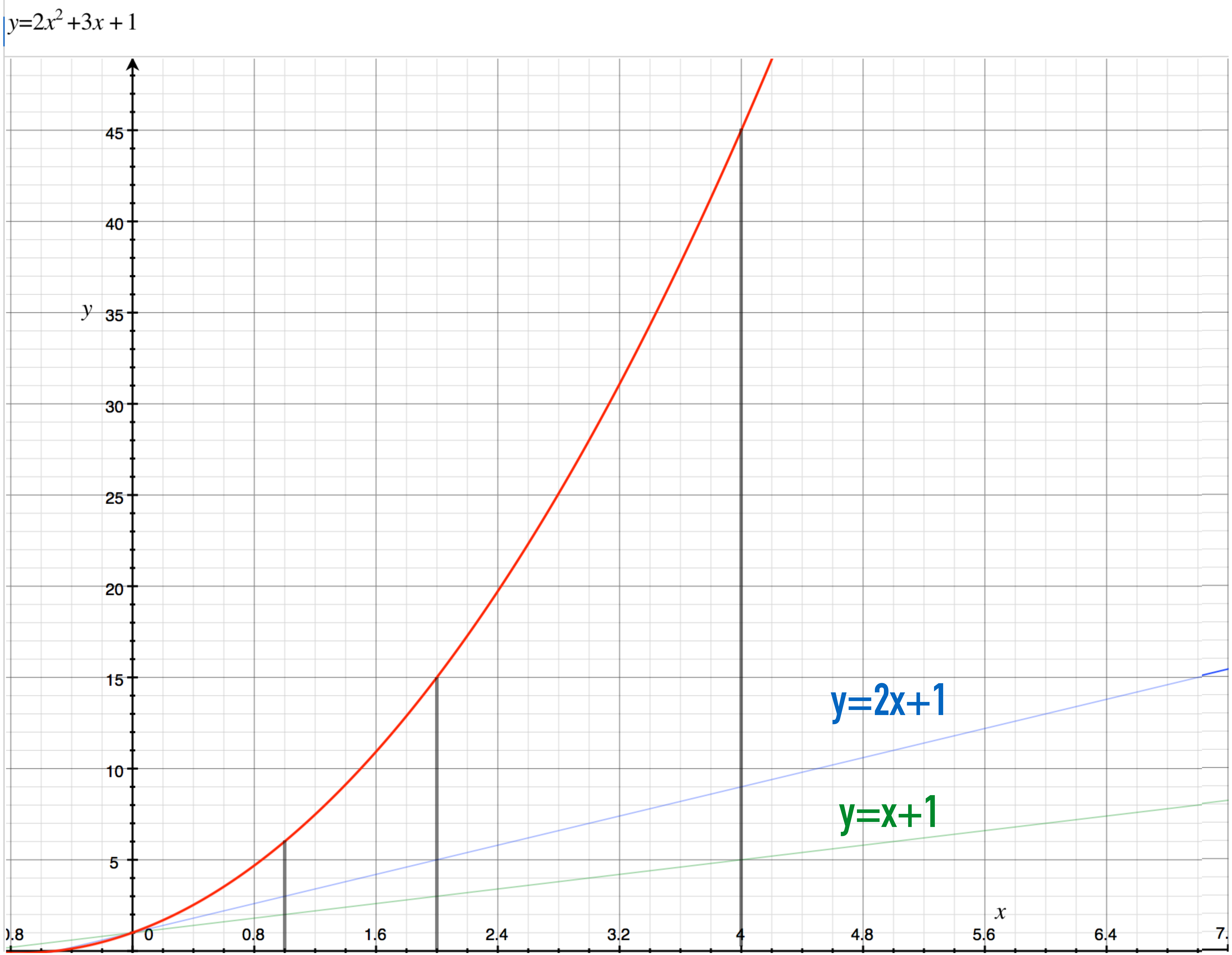
$$A(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

$$B(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

$$C(x) = \begin{array}{l} a_3b_3x^6 + \\ (a_3b_2 + a_2b_3)x^5 + \\ (a_3b_1 + a_2b_2 + a_1b_3)x^4 + \\ (a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3)x^3 + \\ (a_2b_0 + a_1b_1 + a_0b_2)x^2 + \\ (a_1b_0 + a_0b_1)x + \\ a_0b_0 \end{array}$$

$y=2x+1$





y=2x^2+3x+1
y=2x+1
y=x+1
y
x
45
40
35
30
25
20
15
10
5
0
0.8
1.6
2.4
3.2
4
4.8
5.6
6.4

a_3 a_2 a_1 a_0 ★ b_3 b_2 b_1 b_0

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7$$

$$B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7$$

a_3 a_2 a_1 a_0 ★ b_3 b_2 b_1 b_0

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7$$

$$B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7$$

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

a_3 a_2 a_1 a_0 ★ b_3 b_2 b_1 b_0

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7$$

$$B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7$$

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

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

a_3 a_2 a_1 a_0 ★ b_3 b_2 b_1 b_0

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7$$

$$B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7$$

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

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

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

a3 a2 a1 a0 ★ b3 b2 b1 b0

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7$$

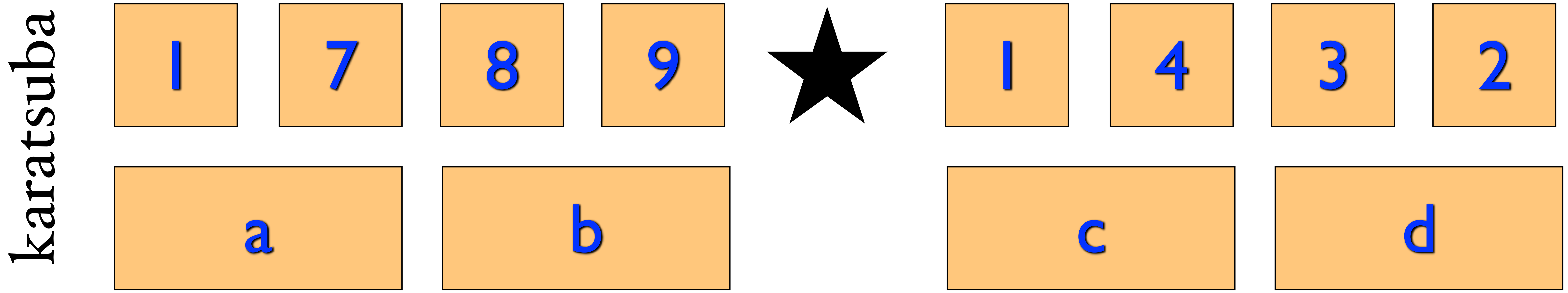
$$B(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + 0x^4 + 0x^5 + 0x^6 + 0x^7$$

$A(\omega_0)$	$A(\omega_1)$	$A(\omega_2)$	....	$A(\omega_7)$	**FFT**
$B(\omega_0)$	$B(\omega_1)$	$B(\omega_2)$	....	$B(\omega_7)$	**FFT**
$C(\omega_0)$	$C(\omega_1)$	$C(\omega_2)$	....	$C(\omega_7)$	

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

IFFT

application to mult



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

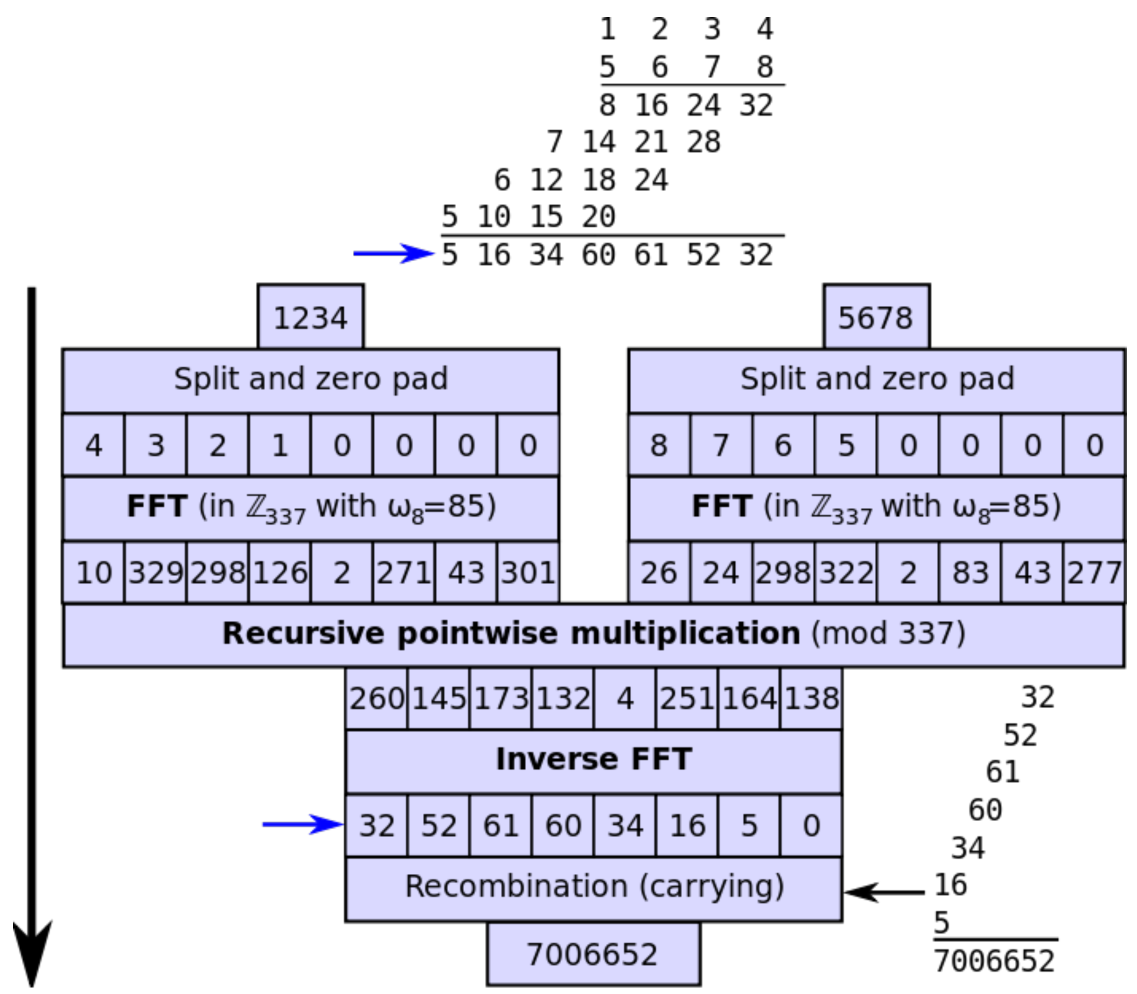
application to mult

karatsuba

1	7	8	9	★	1	4	3	2
a		b			c		d	

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

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



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

Multiplying n-bit numbers

Schönhage–Strassen ‘71	$O(n \log n \log \log n)$
Fürer ‘07	$O(n \log n 4^{\log^*(n)})$
Harvey-van der Hoeven ‘20	$O(n \log n)$

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

PIERRICK GAUDRY, ALEXANDER KRUPPA, AND PAUL ZIMMERMANN

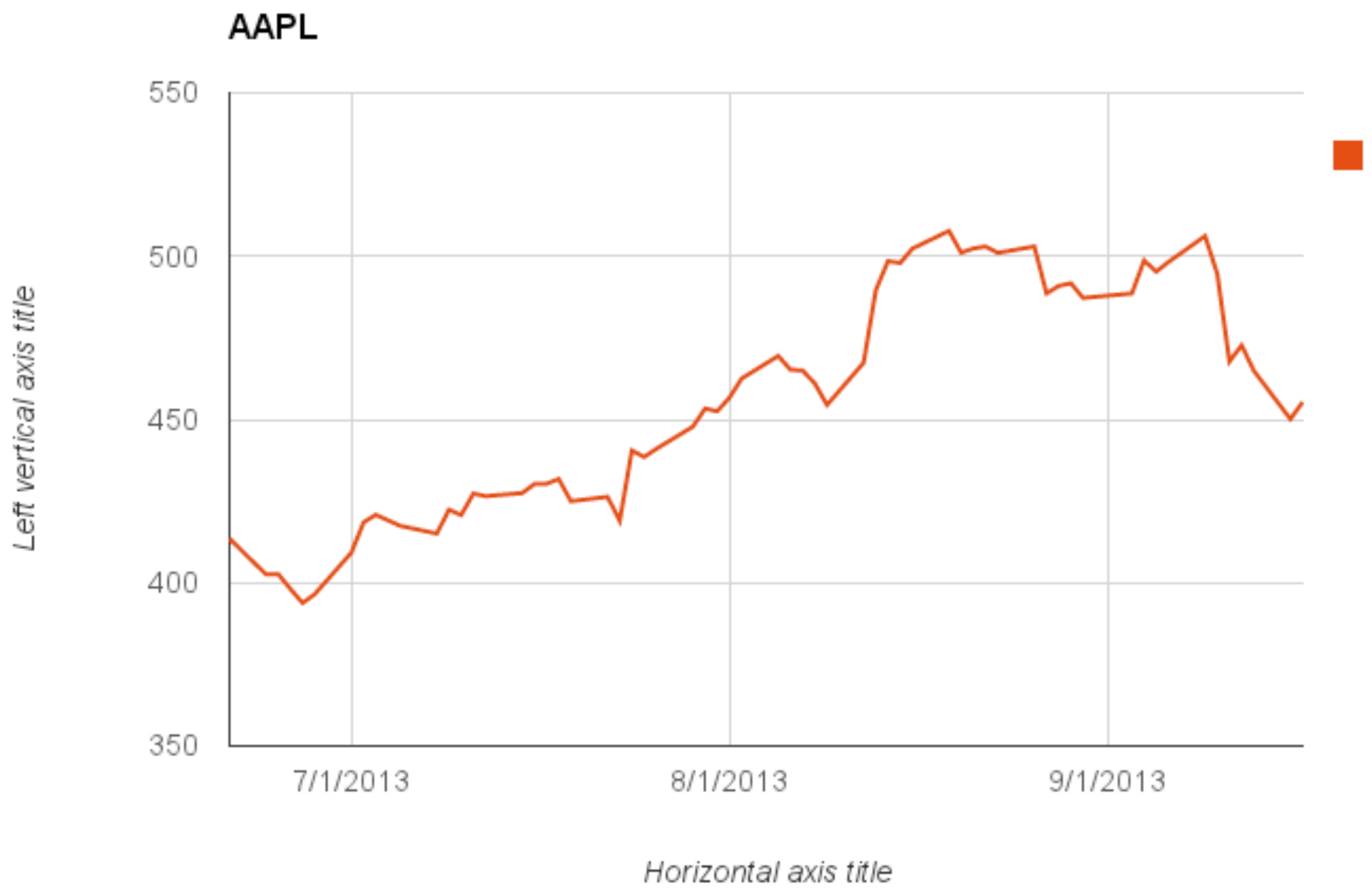
Abstract. Schönhage-Strassen's algorithm is one of the best known algorithms for multiplying large integers. Implementing it efficiently is of utmost importance, since many other algorithms rely on it as a subroutine. We present here an improved implementation, based on the one distributed within the 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

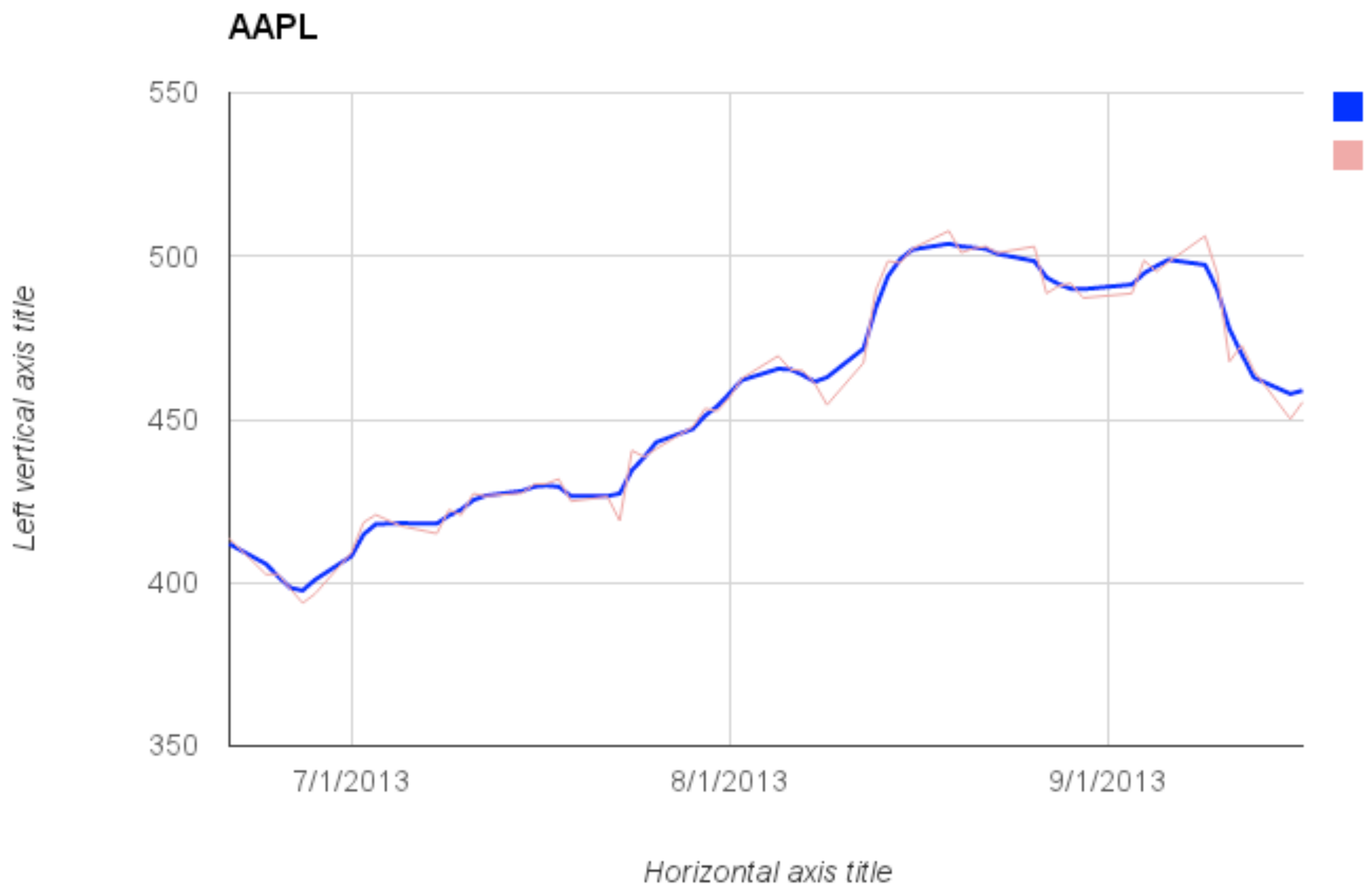
Since Schönhage and Strassen have shown in 1971 how to multiply two N-bit integers in $O(N \log N \log \log N)$ time [21], several authors showed how to reduce other operations — inverse, division, square root, gcd, base conversion, elementary functions — to multiplication, possibly with $\log N$ multiplicative 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]).

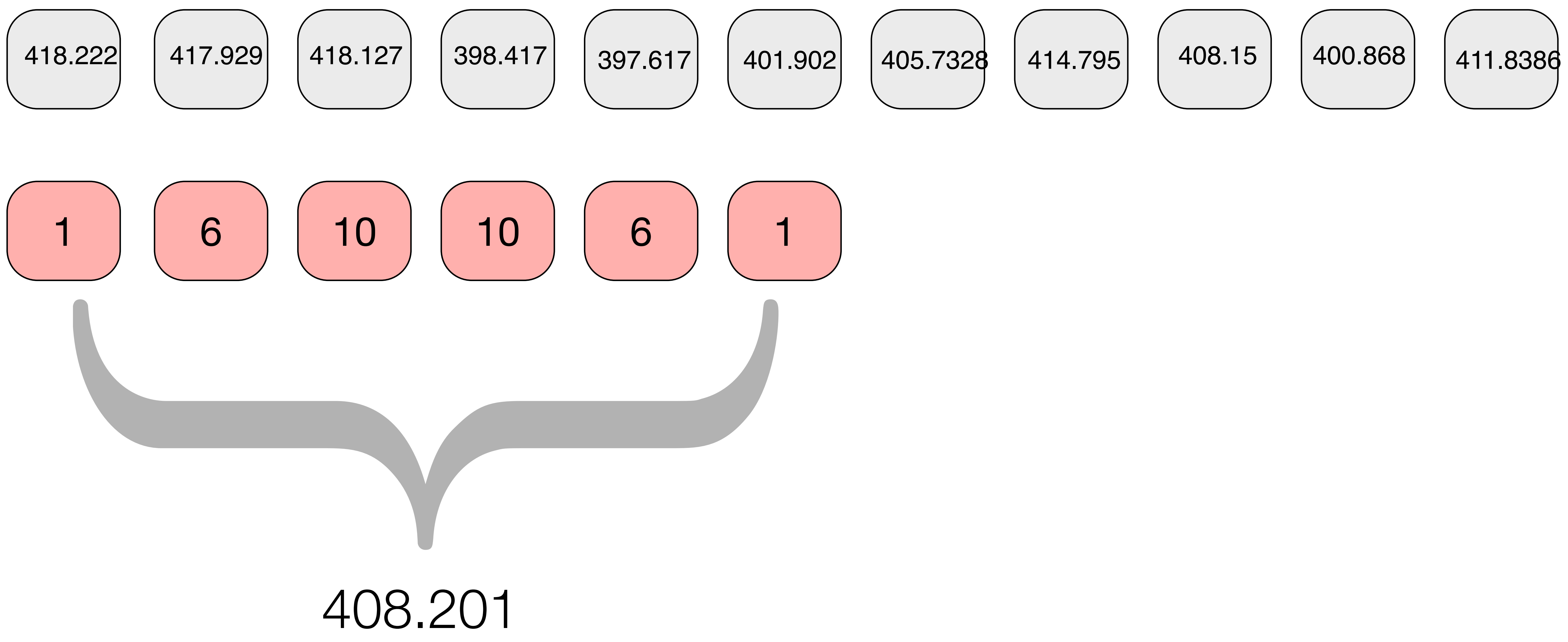
Strangely, much less effort was made for decreasing the implicit constant in $M(N)$ itself, although any gain on that constant will give a similar gain on all multiplication-based operations. Some authors reported on implementations of large integer arithmetic for specific hardware or as part of a number-theoretic 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

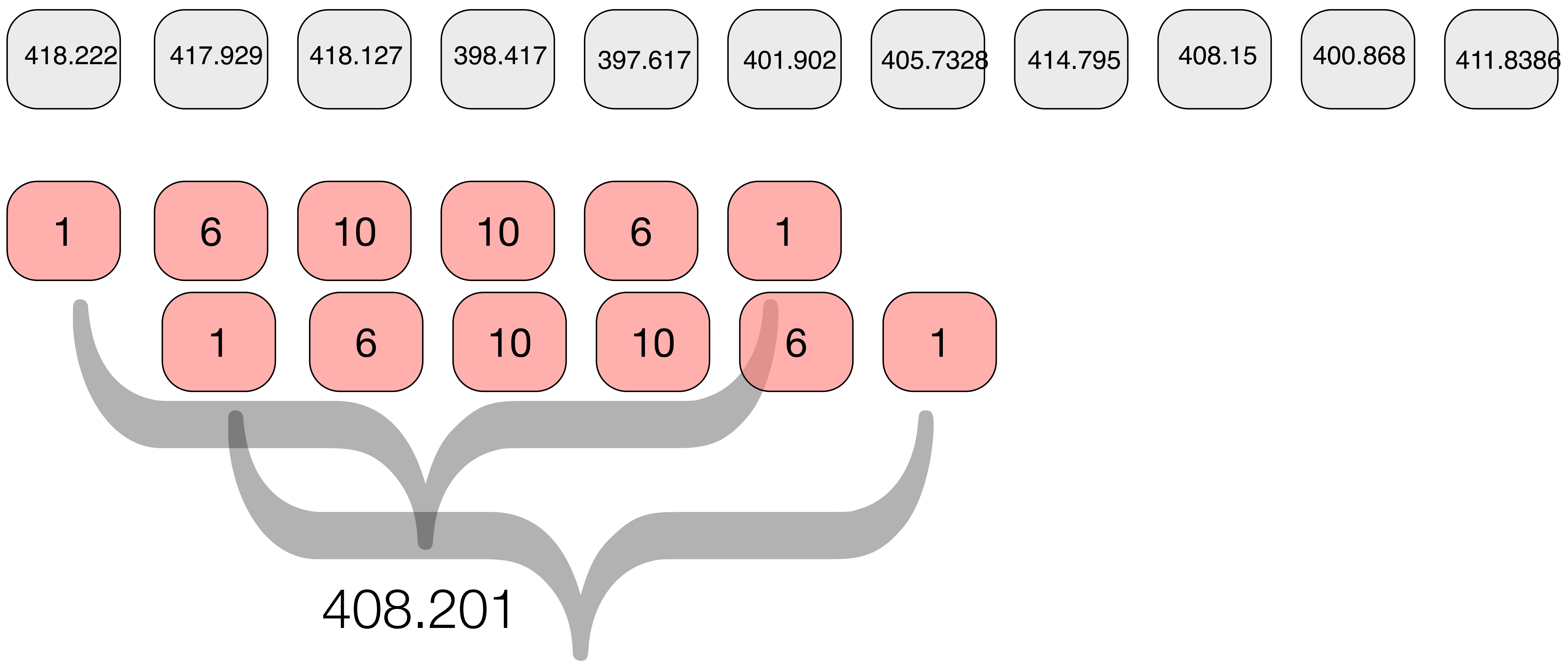


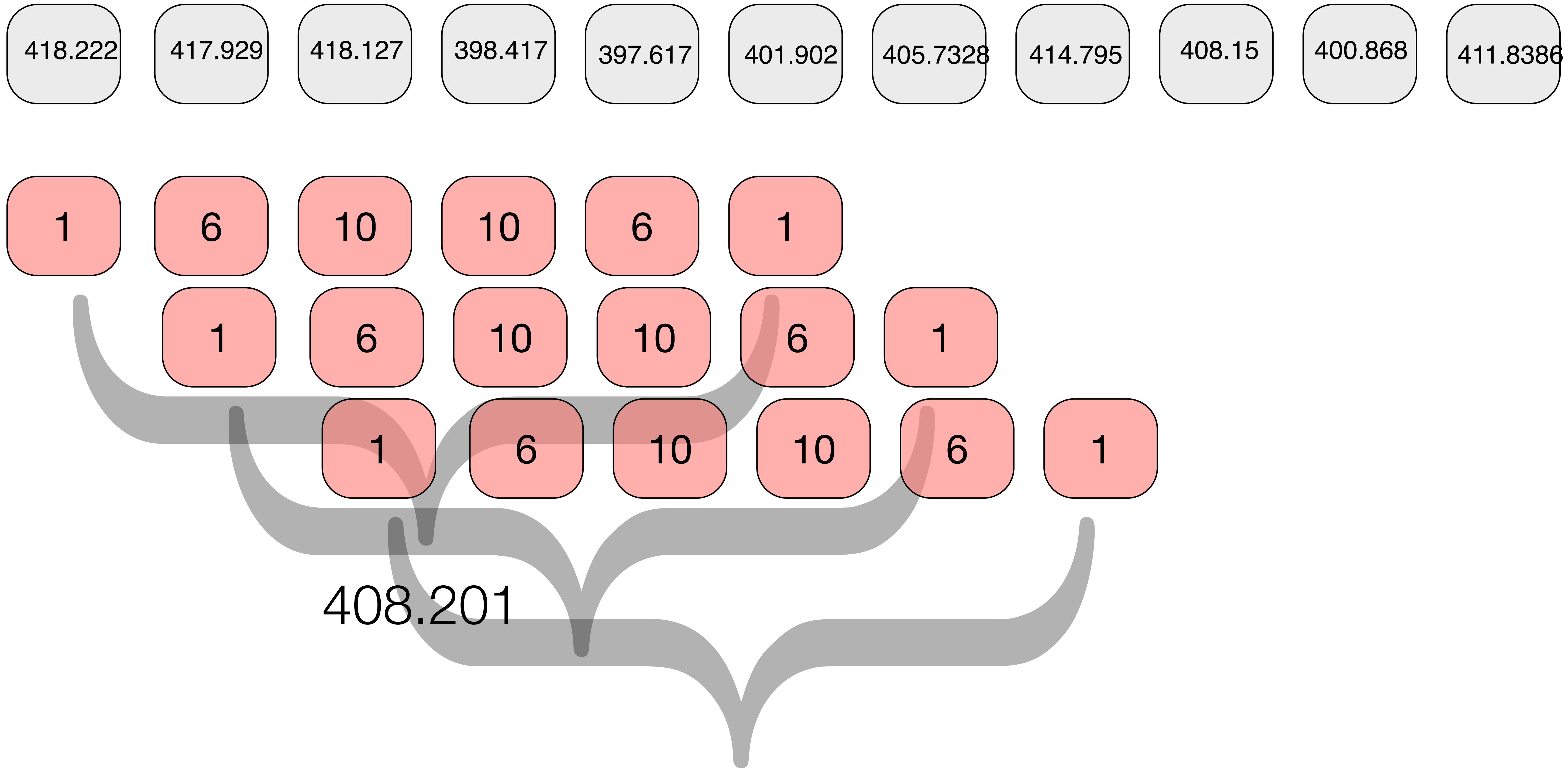
Applications of FFT

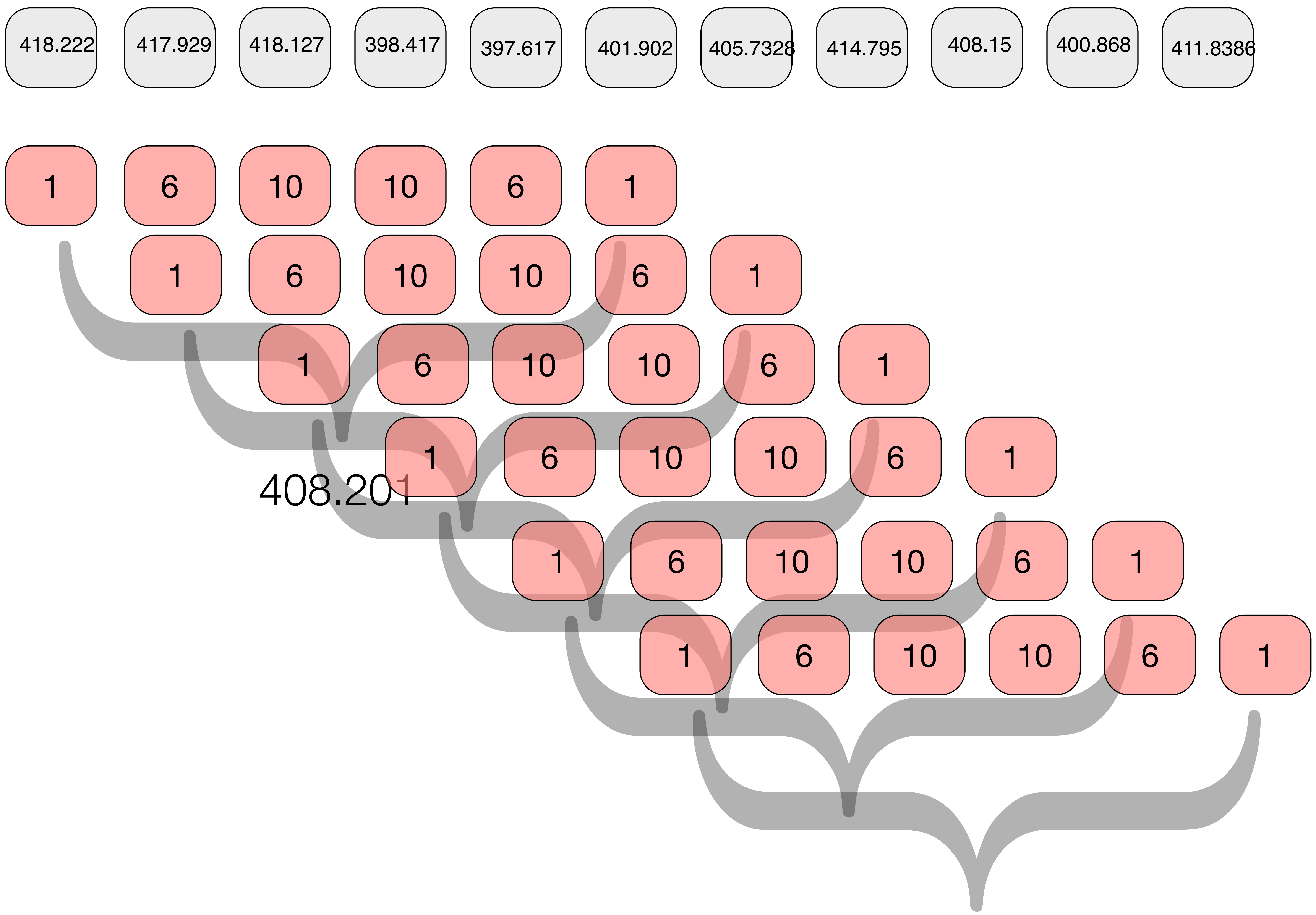




418.222
417.929
418.127
398.417
397.617
401.902
405.7328
414.795
408.15
400.868
411.8386
1
6
10
10
6
1
408.201


418.222
417.929
418.127
398.417
397.617
401.902
405.7328
414.795
408.15
400.868
411.8386
1
6
10
10
6
1
1
6
10
10
6
1
408.201


418.222
417.929
418.127
398.417
397.617
401.902
405.7328
414.795
408.15
400.868
411.8386
1
6
10
10
6
1
1
6
10
10
6
1
1
6
10
10
6
1
408.201


418.222
417.929
418.127
398.417
397.617
401.902
405.7328
414.795
408.15
400.868
411.8386
1 6 10 10 6 1
1 6 10 10 6 1
1 6 10 10 6 1
1 6 10 10 6 1
408.201
1 6 10 10 6 1
1 6 10 10 6 1

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.