L5

shelat

16f-4800

sep 23 2016

Matrix Mult, Median, FFT

merge-sort (A, p, r)
    if $p < r$
        $q \leftarrow \lfloor (p+r)/2 \rfloor$
        merge-sort (A, p, q)
        merge-sort $(A, q+1, r)$
        merge(A, p, q, r)

```
MERGE(A[1..n], m):
  i ← 1;  j ← m + 1
  for k ← 1 to n
      if j > n
          B[k] ← A[i];  i ← i + 1
      else if i > m
          B[k] ← A[j];  j ← j + 1
      else if A[i] < A[j]
          B[k] ← A[i];  i ← i + 1
      else
          B[k] ← A[j];  j ← j + 1
  for k ← 1 to n
      A[k] ← B[k]
```

jeff erickson

Karatsuba(ab, cd)

Base case: return b*d if inputs are 1-digit

ac = Karatsuba(a,c)

bd = Karatsuba(b,d)

t = Karatsuba((a+b),(c+d)) $\left(\frac{n}{2}+1\right)$

$3T(n/2) + 2n$

mid = t - ac - bd $4n$

RETURN ac*100^2 + mid*100 + bd $3n$

Closest(P,SX,SY)

Let q be the middle-element of SX

Divide P into Left, Right according to q

delta,r,j = MIN(Closest(Left, LX, LY) Closest(Right, RX, RY))

Mohawk = { Scan SY, add pts that are delta from q.x }

For each point x in Mohawk (in order):

Compute distance to its next 15 neighbors

Update delta,r,j if any pair (x,y) is < delta

Return (delta,r,j)

Can be reduced to 7!

```
arbit+(A[1...n])
 base case if |A|<=2, …
 (lg,minl,maxl) = arbit(left(A))
 (rg,minr,maxr) = arbit(right(A))
   return max{maxr-minl,lg,rg},
          min{minl, minr},
          max{maxl, maxr}
```

(3) New examples

(1) MATRIX MULT

(2) MEDIAN

(3) FFT

Matrix
multiplication

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \star \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1\cdot 5 + 2\cdot 7 & 1\cdot 6 + 2\cdot 8 \\ & 3\cdot 6 + 4\cdot 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

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

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

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

- C_{11} takes n operations (mults & adds)
- there are n^2 elements in the answer
- So the naive alg requires $n \cdot n^2 = \Theta(n^3)$ op.

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

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

A $\quad n/2 \quad$ B $\qquad\qquad$ E $\qquad$ F

$$
\left[\begin{array}{cc|cc}
a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
\hline
\vdots & & & \\
a_{n,1} & a_{n,2} & \cdots & a_{n,n}
\end{array}\right]
\left[\begin{array}{cc|cc}
b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\
b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\
\hline
\vdots & & & \\
b_{n,1} & b_{n,2} & \cdots & b_{n,n}
\end{array}\right]
=
\begin{bmatrix}
c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\
c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\
\vdots & & & \\
c_{n,1} & c_{n,2} & \cdots & c_{n,n}
\end{bmatrix}
$$

C $\qquad$ D $\qquad\qquad$ G $\qquad$ H

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

$\frac{n}{2} \times \frac{n}{2}$ sized matrix

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

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

AE = $\frac{n}{2} \times \frac{n}{2}$ matrix

$T(n) = 8T(\frac{n}{2}) + \Theta(n^2)$ by case I, we have

$T(n) = \Theta(n^3)$

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

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

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

$$\Theta(n^3)$$

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

$= AF + BH$

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

$= AF + BH$

[Strassen]

$P_1 = A(F - H)$

$P_2 = (A + B)H$

$P_3 = (C + D)E$

$P_4 = D(G - E)$

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

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

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

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

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

$R = P_5 + P_4 - P_2 + P_6$

$S = P_1 + P_2$

$T = P_3 + P_4$

$U = P_5 + P_1 - P_3 - P_7$

[strassen]

$P_1 = A(F - H)$

$P_2 = (A + B)H$

$P_3 = (C + D)E$

$P_4 = D(G - E)$

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

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

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

$P_5 + P_4 - P_2 + P_6:$

$$\begin{aligned} & AE + \cancel{DE} + \cancel{AH} + \cancel{DH} \\ & \cancel{DG} - \cancel{DE} \\ & \qquad - \cancel{AH} - \cancel{BH} \\ & BG - \cancel{DG} + \cancel{BH} - \cancel{DH} \end{aligned}$$

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

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

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

[strassen]

$P_1 = A(F - H)$

$P_2 = (A + B)H$

$P_3 = (C + D)E$

$P_4 = D(G - E)$

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

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

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

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

by case I, we have

$T(n) = \Theta(n^{\log_2 7}) \approx \Theta(n^{2.807})$

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

$R = P_5 + P_4 - P_2 + P_6$

$S = P_1 + P_2$

$T = P_3 + P_4$

$U = P_5 + P_1 - P_3 - P_7$

[strassen]

$P_1 = A(F - H)$

$P_2 = (A + B)H$

$P_3 = (C + D)E$

$P_4 = D(G - E)$

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

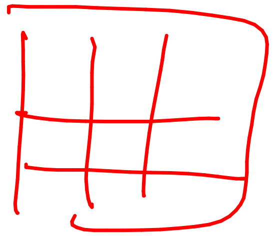
$P_6 = (B - D)(G + H)$

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

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

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

taking this idea further



3x3 matricies [Laderman'75] using 23 operations.

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

by case 1 =

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

goal, if one could do it in 21, $n^{\log_3 21} \sim n^{2.77}$

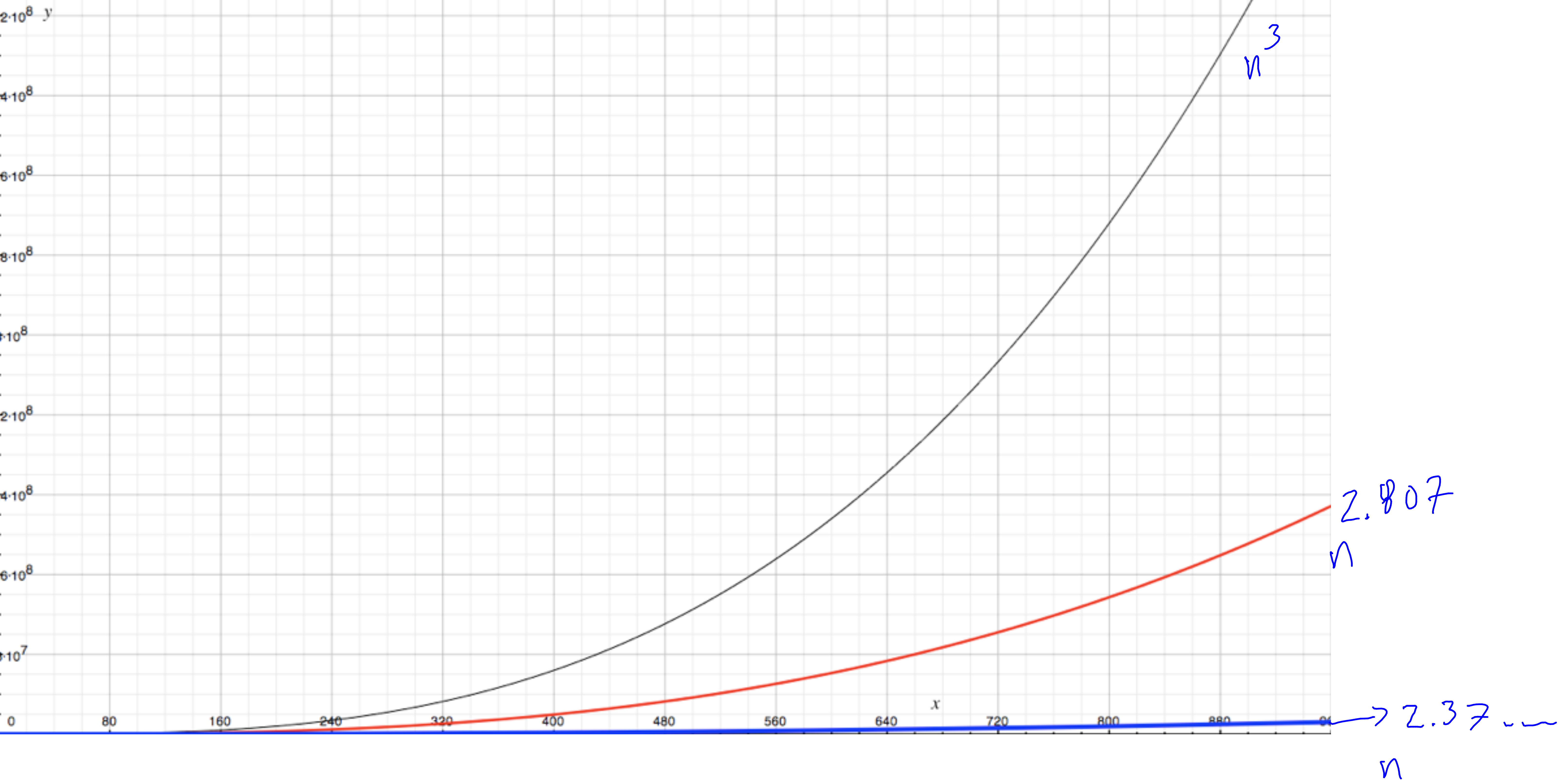
1978 victor pan method

70x70 matrix using 143640 mults

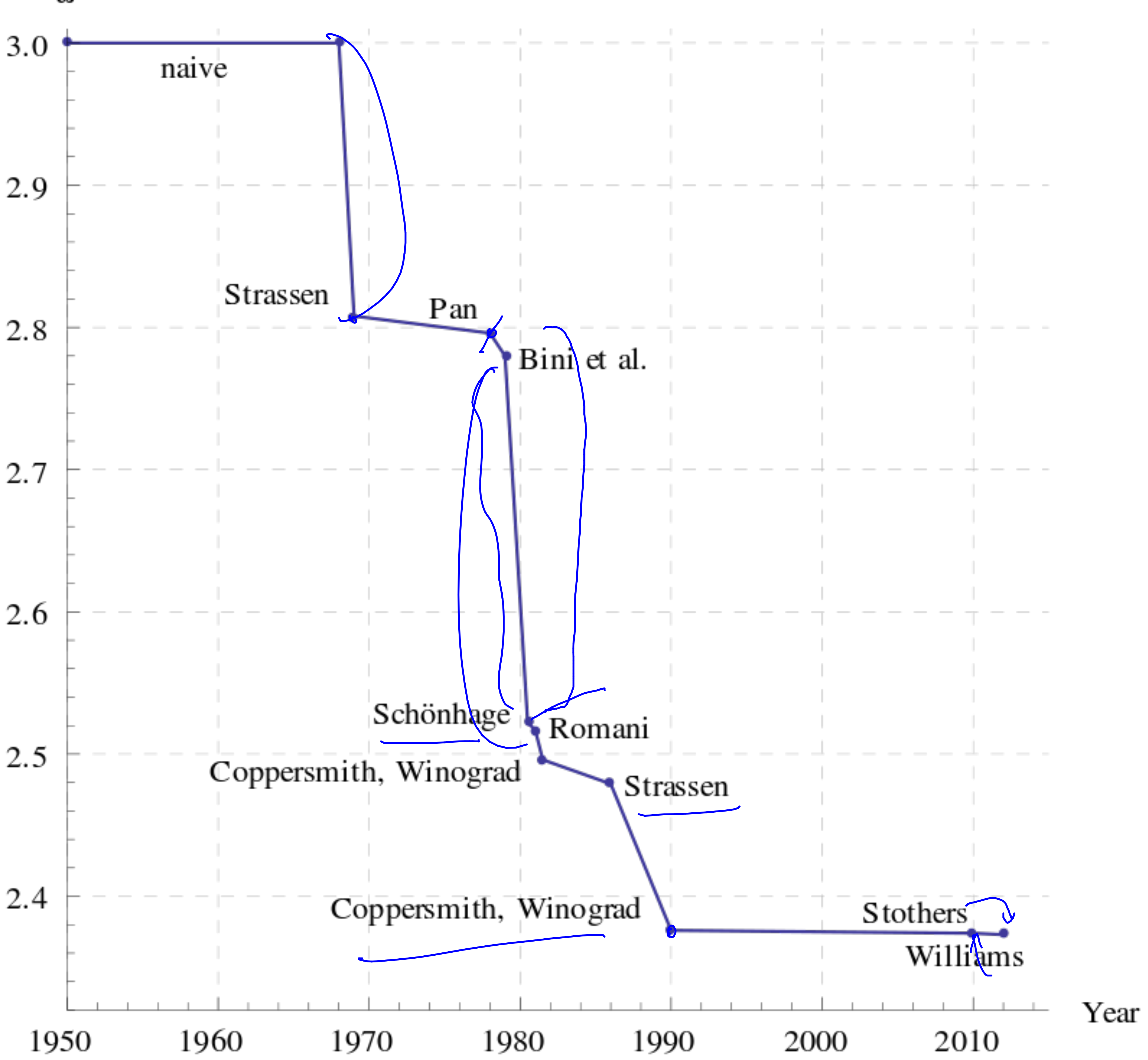
what is the recurrence:

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

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



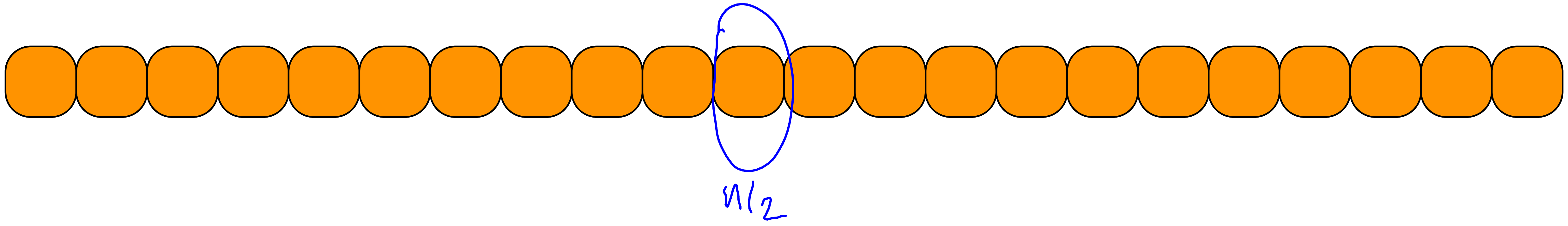
y
x
0
80
160
240
320
400
480
560
640
720
800
880
n^3
$n^{2.807}$
$n^{2.37\ldots}$



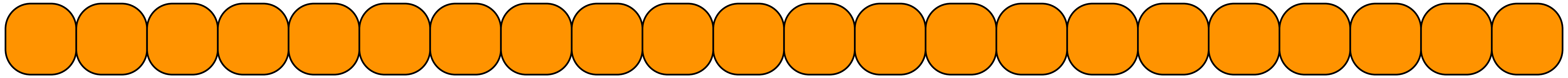
$n = 2$

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)

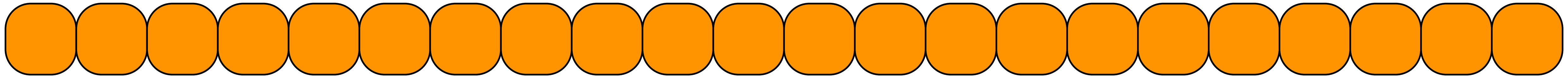


problem: given a list of **n** elements, find the element of rank n/2. (half are larger, half are smaller)

can generalize to i

first solution: sort and pluck.

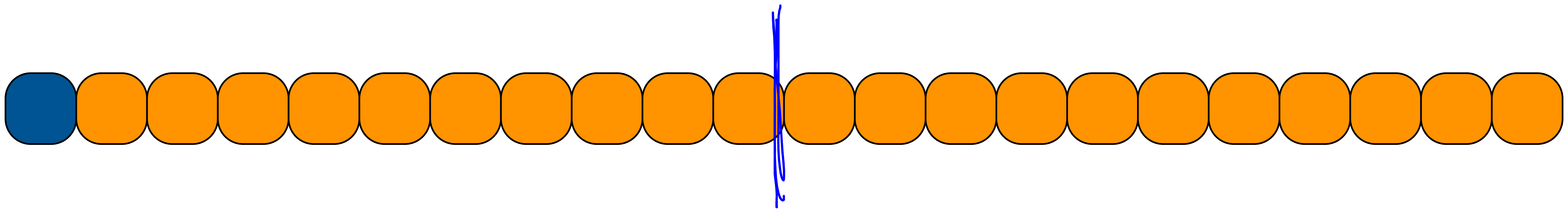
$$O(n \log n)$$



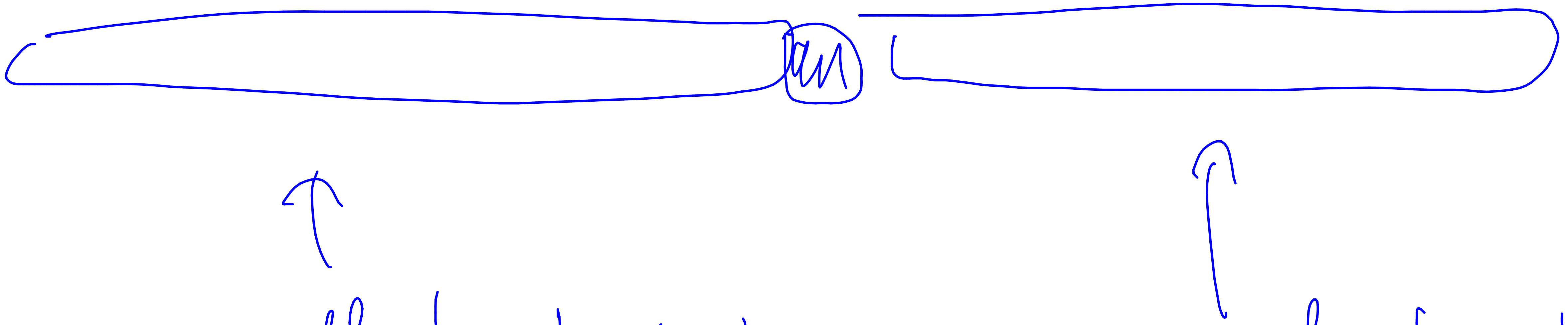
problem: given a list of **n** elements, find the element of rank **i**.

key insight:

we do not have to "fully" sort.
semi sort can suffice.



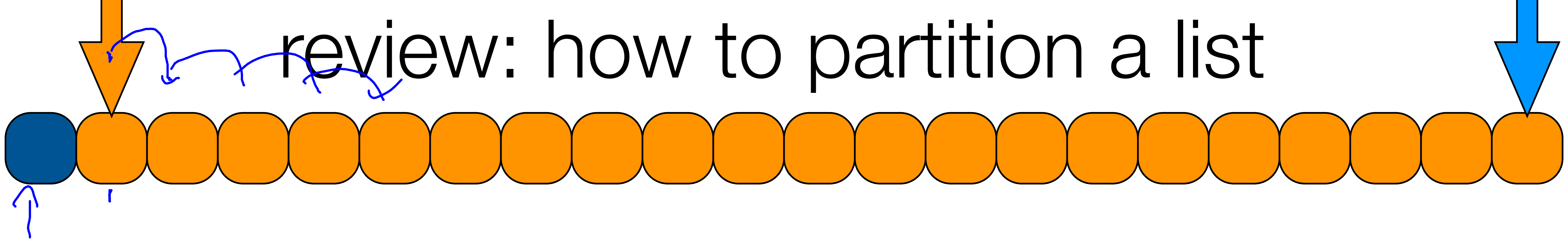
pick first element
partition list about this one
see where we stand



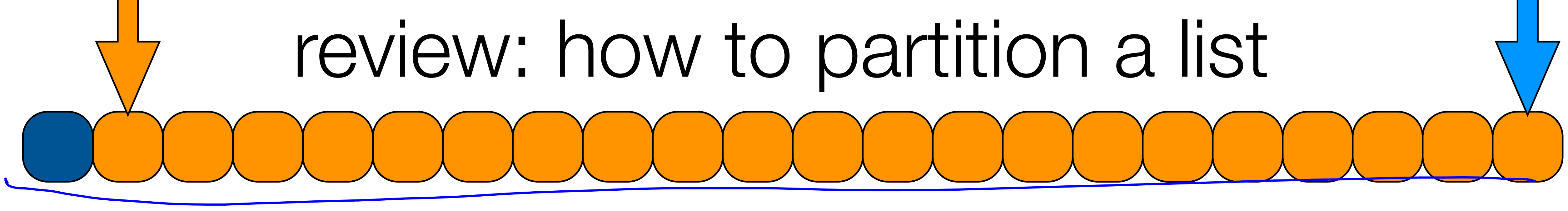
all elements that are smaller than blue

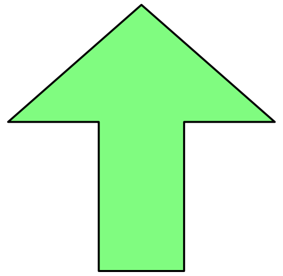
all elements that are larger than blue

review: how to partition a list

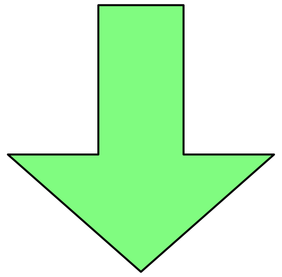


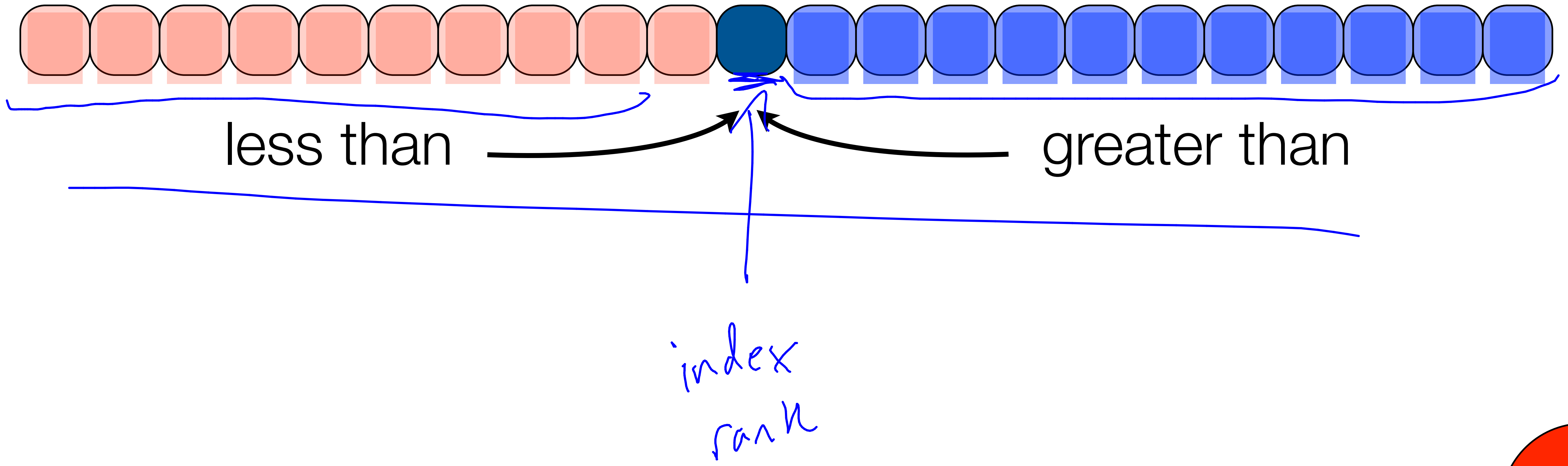
review: how to partition a list





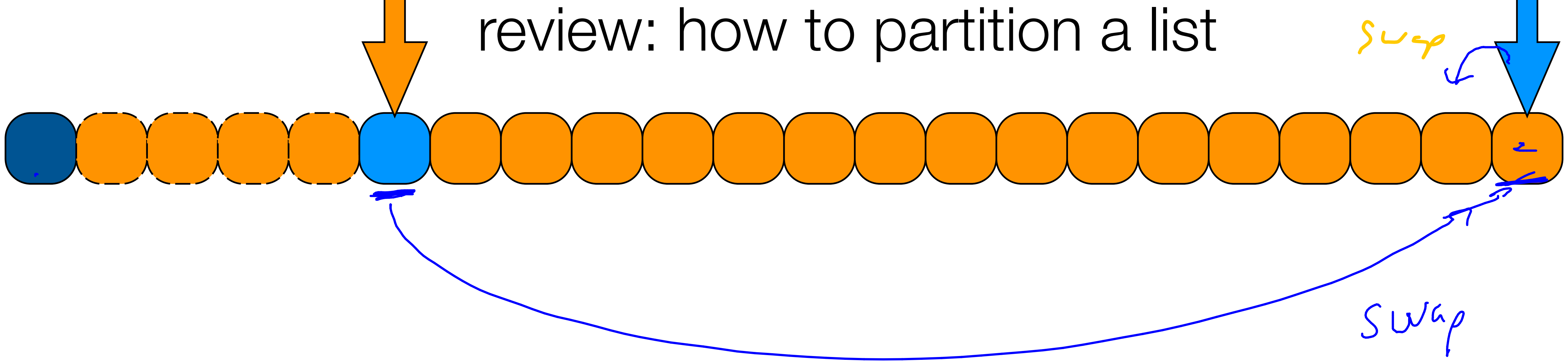
GOAL: start with THIS LIST and END with THAT LIST



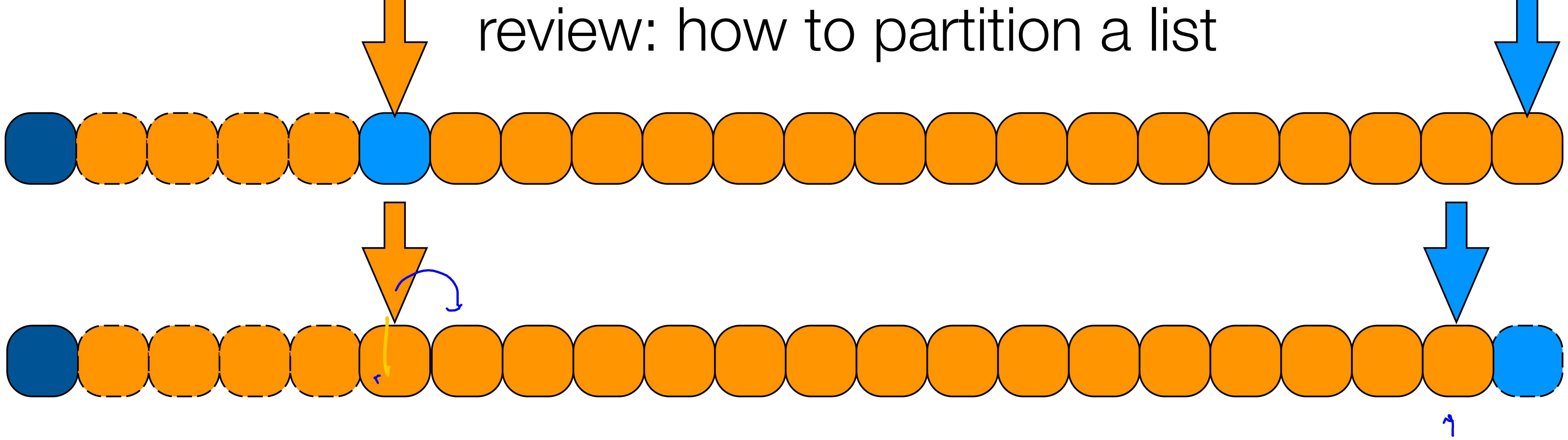


review: how to partition a list

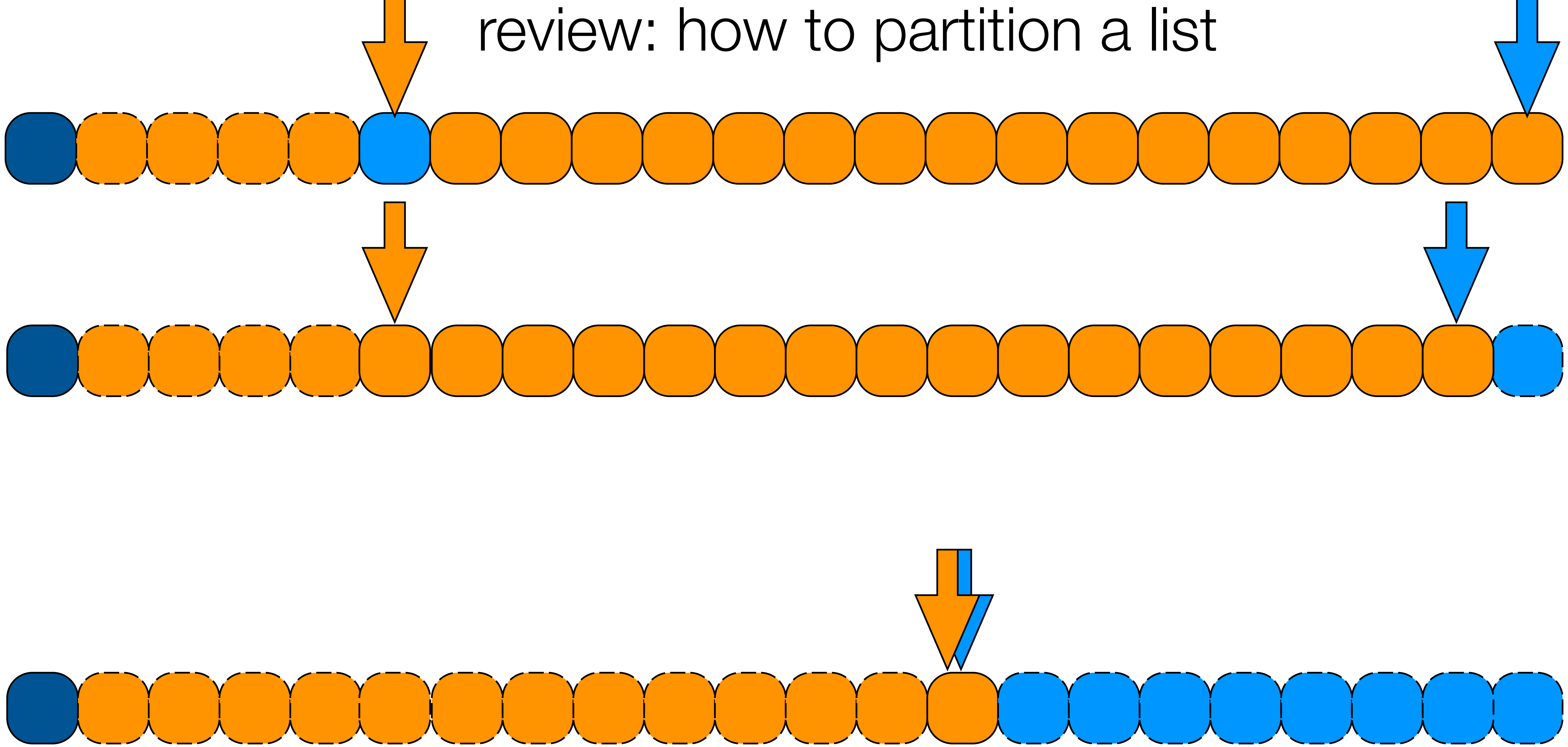
review: how to partition a list



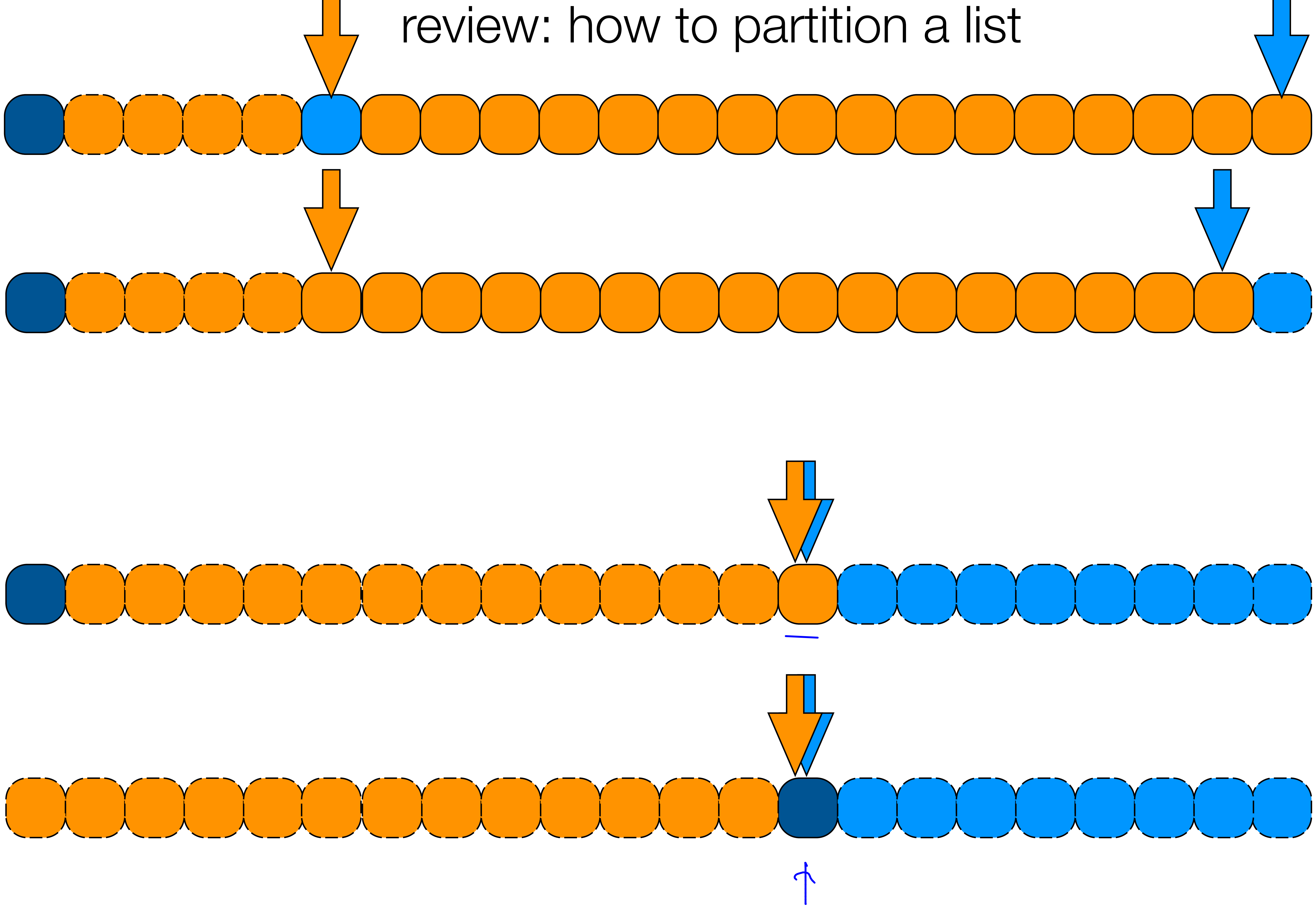
review: how to partition a list



review: how to partition a list

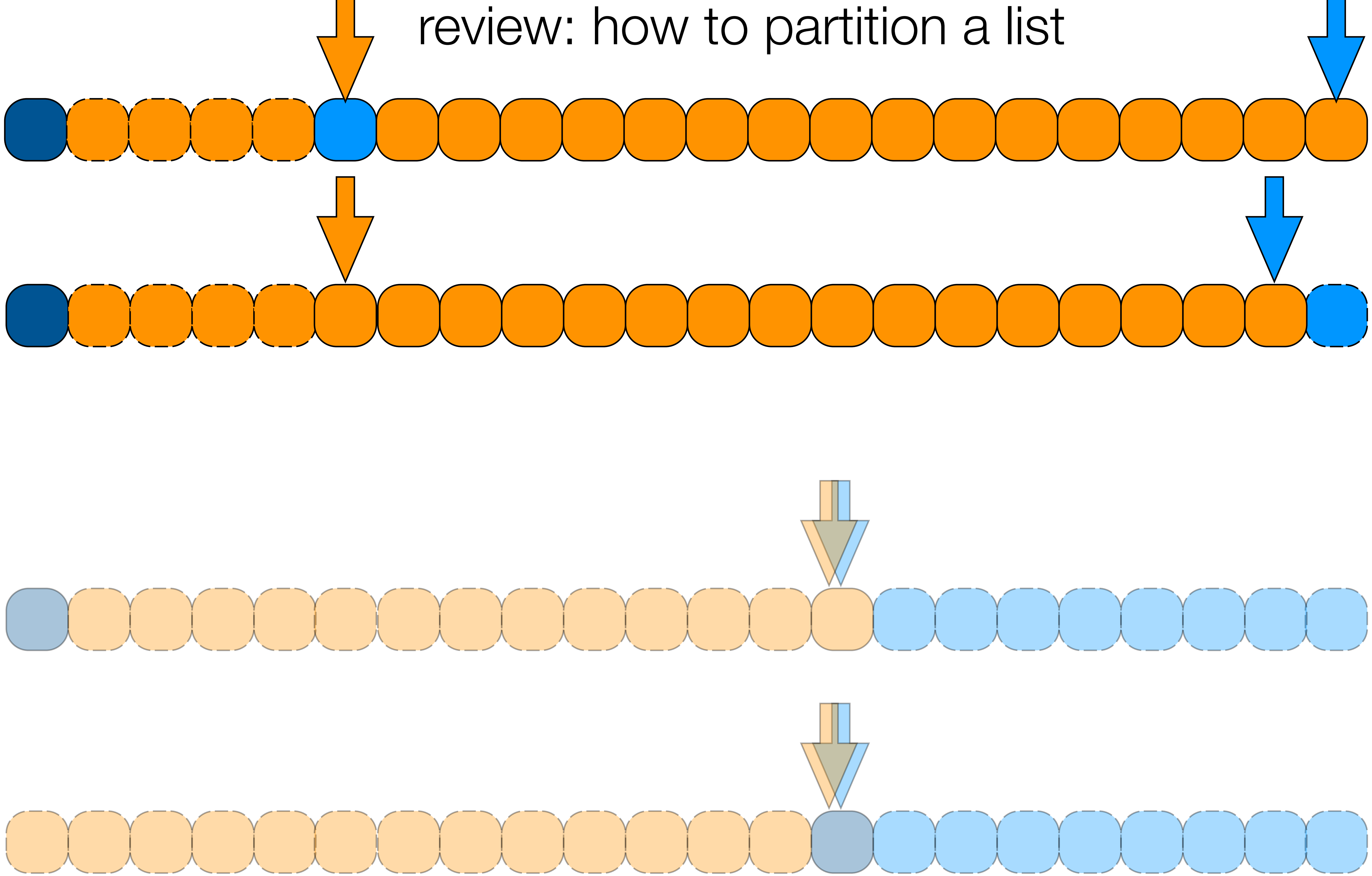


review: how to partition a list

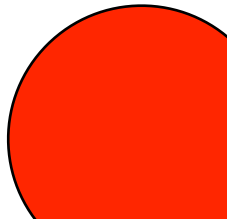


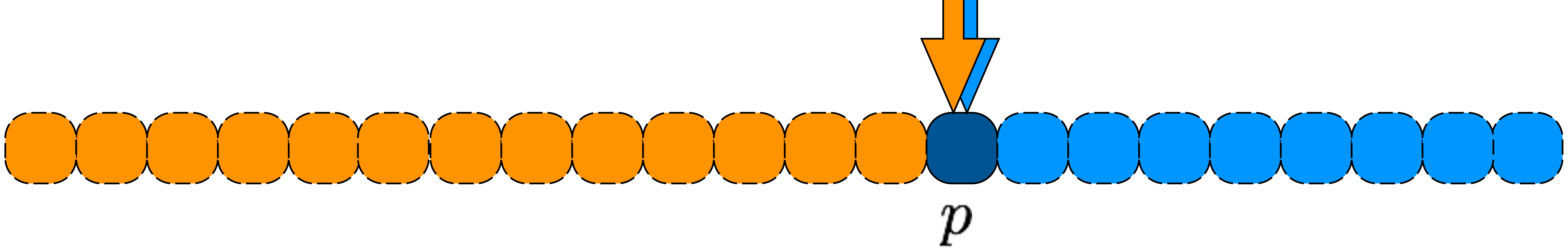
$\Theta(n)$

review: how to partition a list

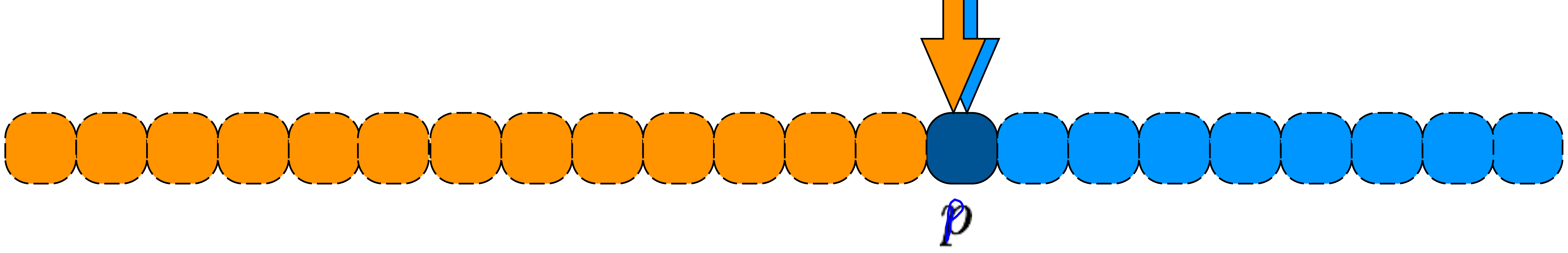


partitioning a list about an element takes linear time.





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



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

handle base case.

partition list about first element p

if pivot p is position i, return pivot $\Big\} \to \Theta(n)$

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

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

right

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

case 3, we get

$S(n) = \Theta(n)$

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

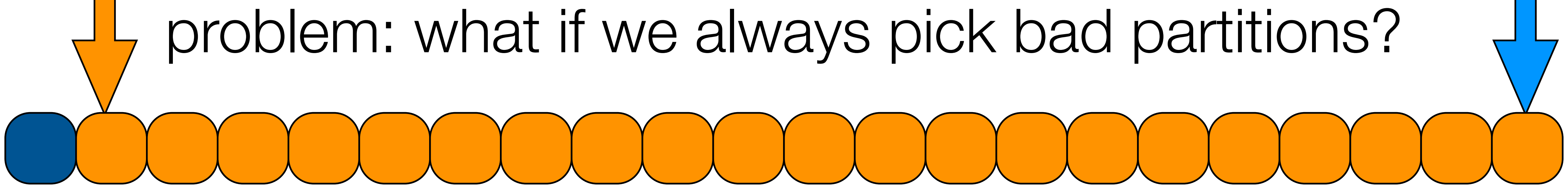
else if pivot is in position > **i** select $(i, A[1,\ldots,p-1])$

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

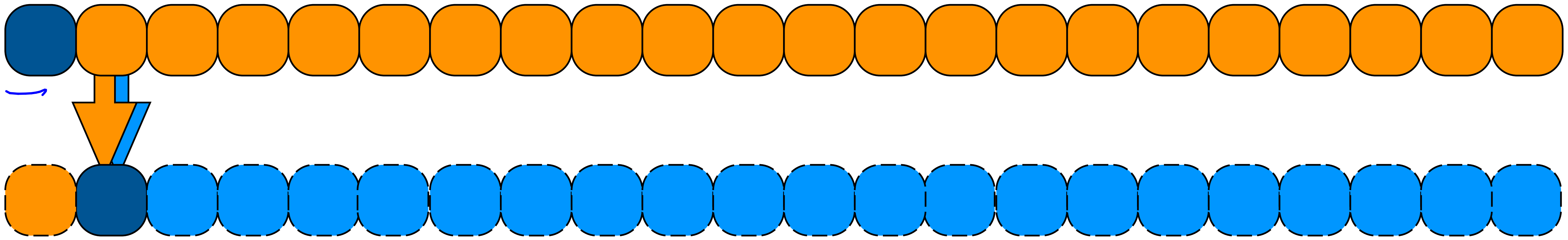
$$T(n) = T(n/2) + O(n)$$

$$\Theta(n)$$

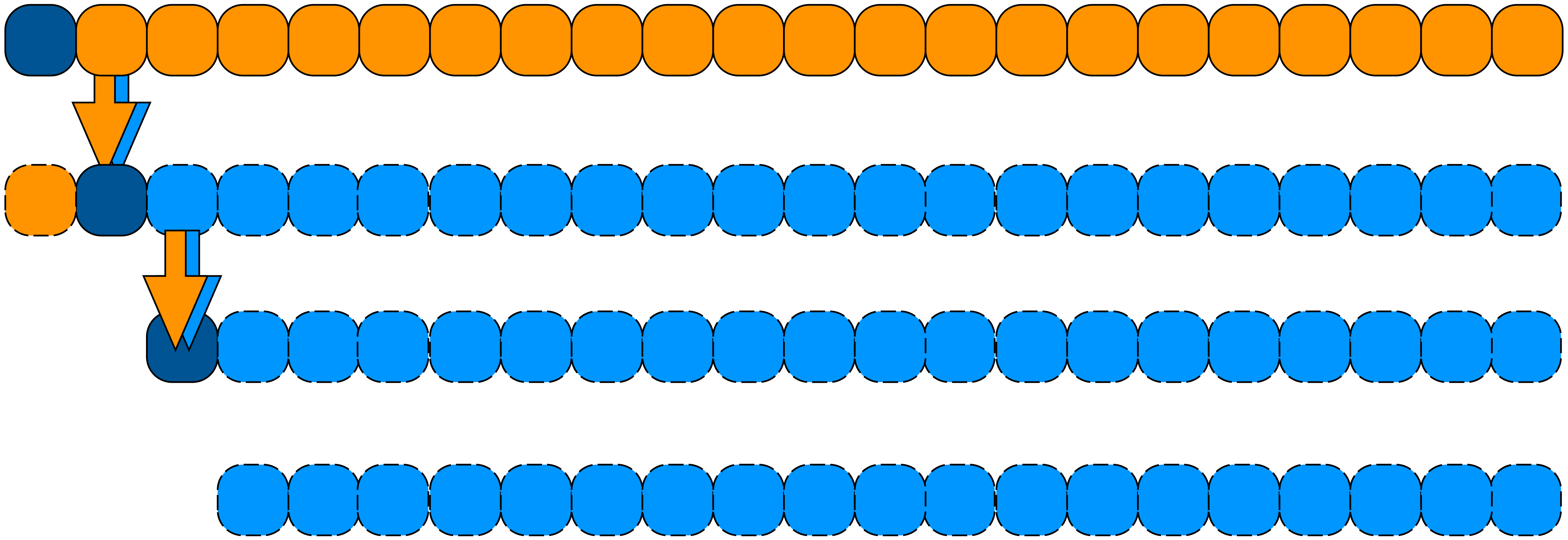
problem: what if we always pick bad partitions?

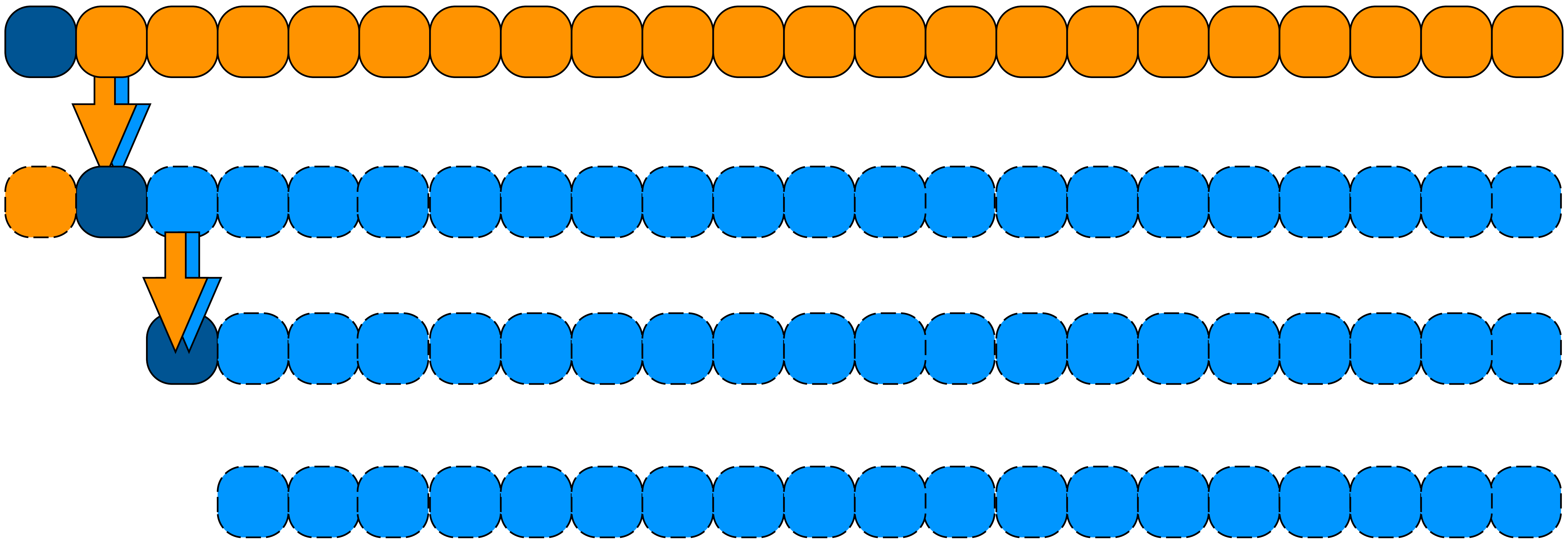


problem: what if we always pick bad partitions?



problem: what if we always pick bad partitions?





problem: what if we always pick bad partitions?

$$S(n) = S(n-1) + \Theta(n) \leadsto \Theta(n^2)$$

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

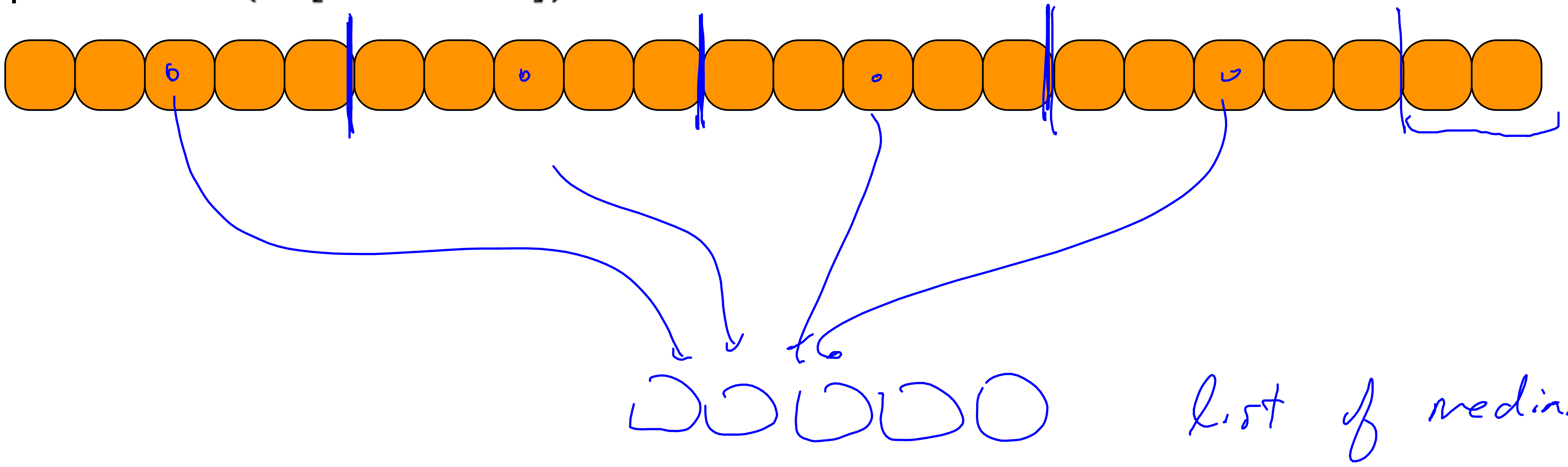
solution: bootstrap

image: gucci

image: mark nason

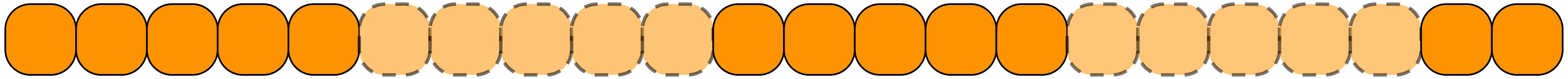
image: d&g

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

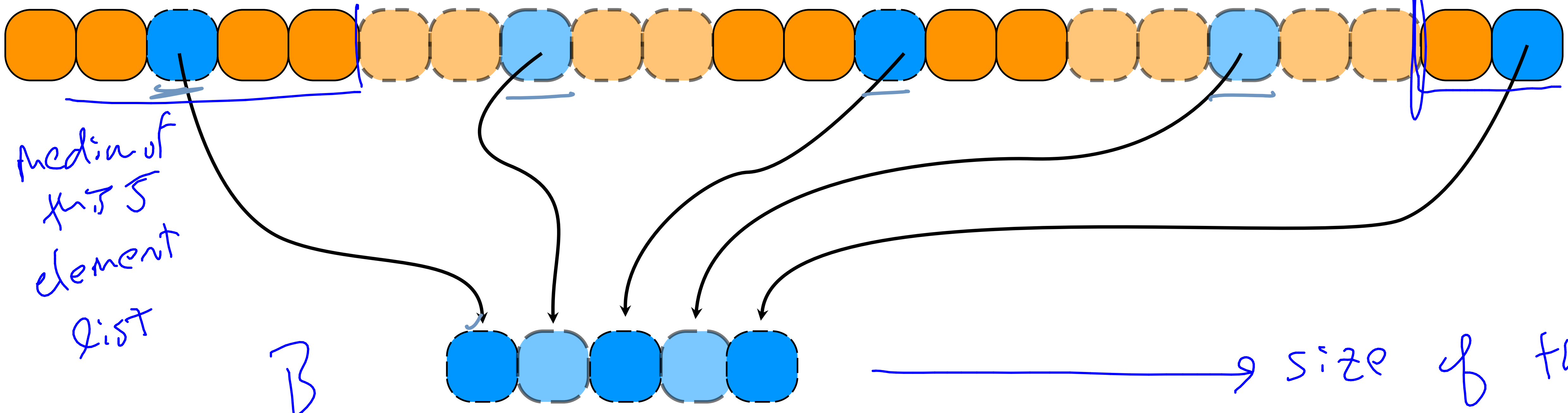


list of medians from each group.

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



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



median of this 5 element list

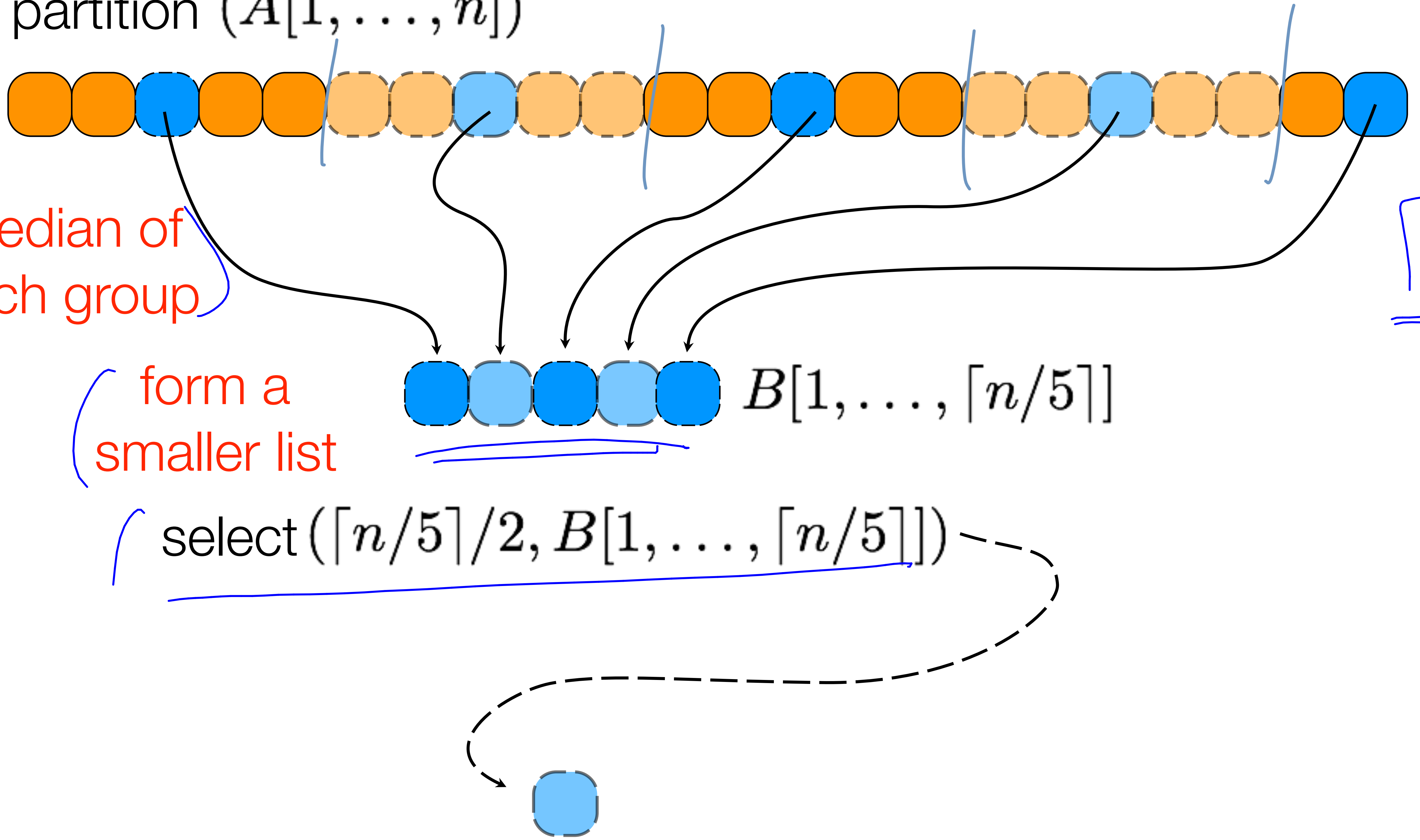
B

size of this list is $\left\lceil \frac{n}{5} \right\rceil$

find median of this list

$\text{Select}\left(\left\lceil \frac{n/5}{2} \right\rceil, B\right)$

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



median of each group

$\lceil \frac{n}{5} \rceil$ $\Theta(1)$

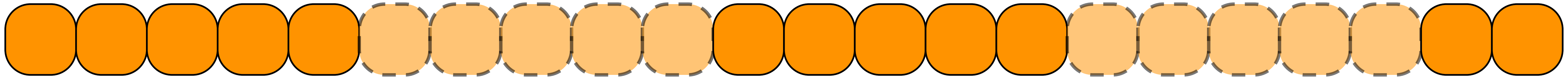
form a smaller list

$B[1, \ldots, \lceil n/5 \rceil]$

select $(\lceil n/5 \rceil / 2, B[1, \ldots, \lceil n/5 \rceil])$

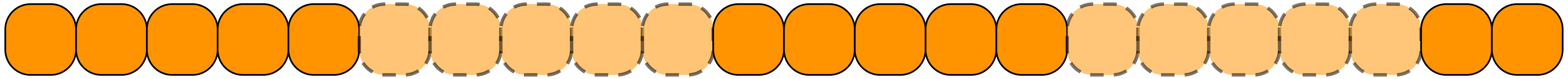
use the median of this smaller list as the partition element

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



1.
2.
3.
4.
5.

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



divide list into groups of 5 elements
find median of each small list
gather all medians

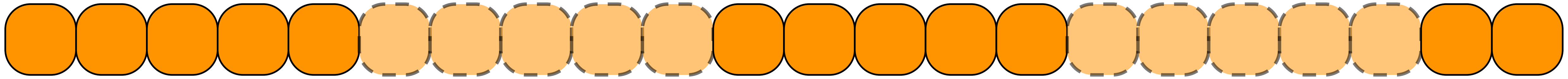
$\lceil \frac{n}{5} \rceil \cdot \Theta(1)$

call select(...) on this sublist to find median $S(\lceil \frac{n}{5} \rceil)$

return the result

$$P(n) = S(\lceil \tfrac{n}{5} \rceil) + \Theta(n)$$

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



divide list into groups of 5 elements

find median of each small list

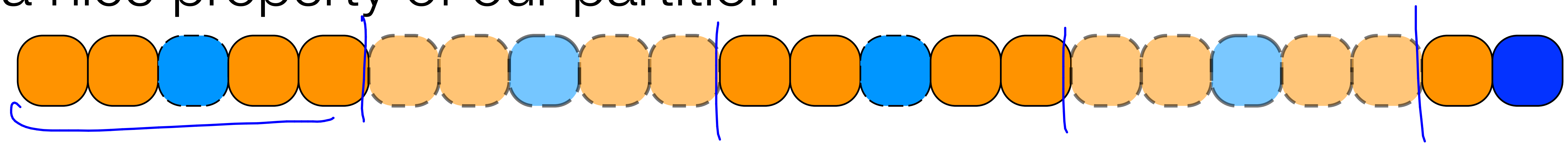
gather all medians

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



0

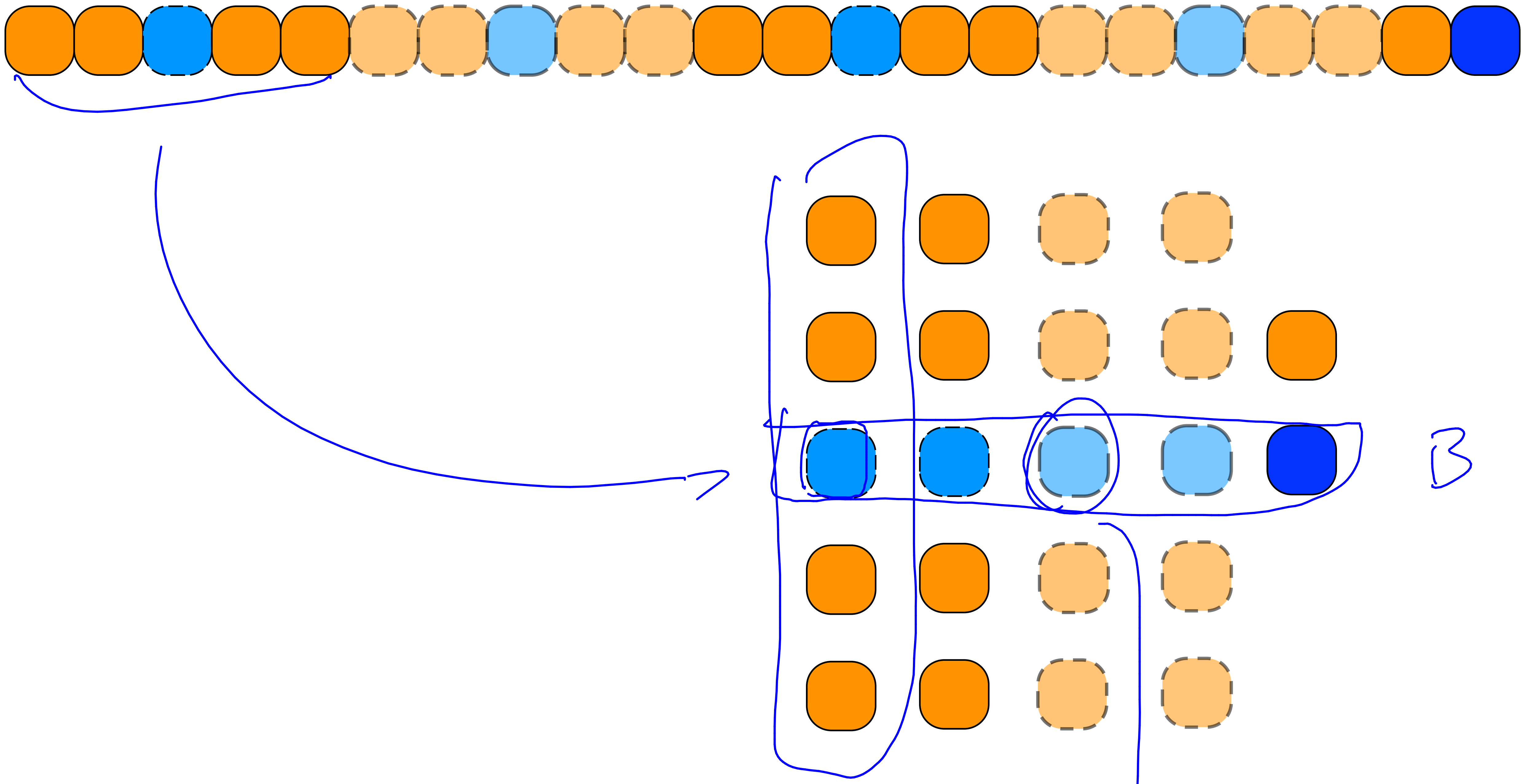
2

0

2

0

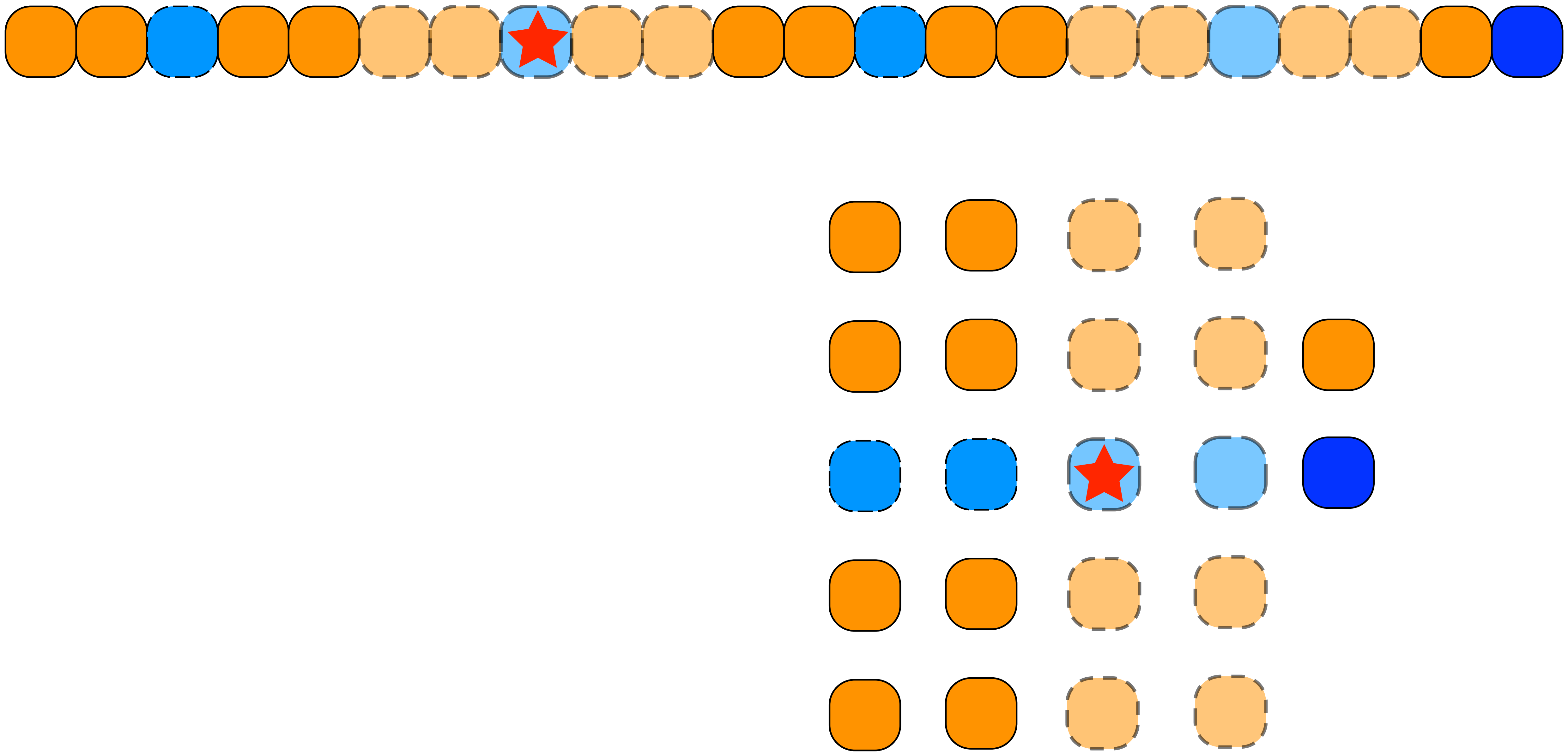
a nice property of our partition



sort each column

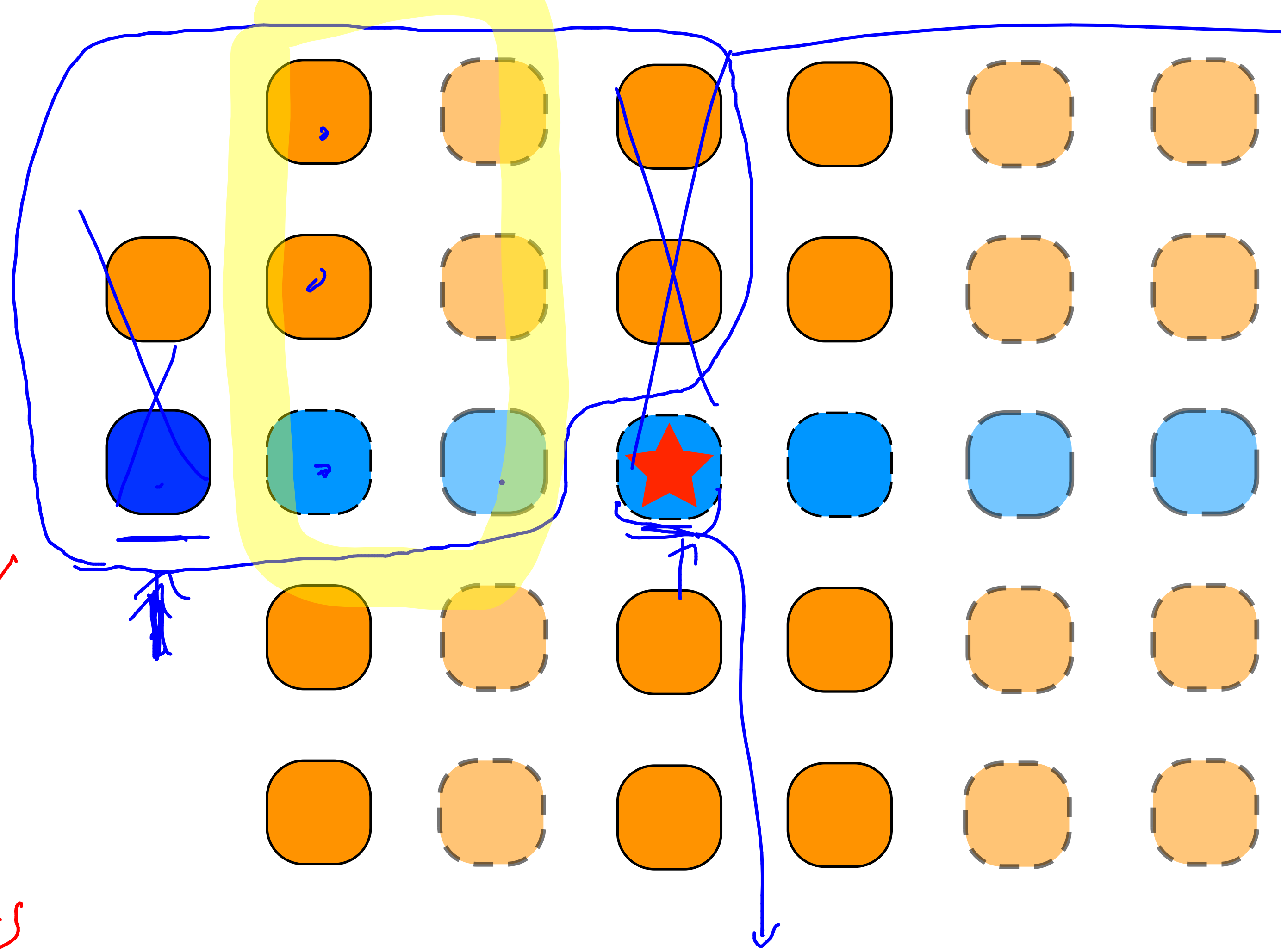
median of medians that ? eventually returns

a nice property of our partition



SWITCH TO A BIGGER EXAMPLE

$$3\left(\frac{\frac{n}{5}}{2} - 2\right)$$

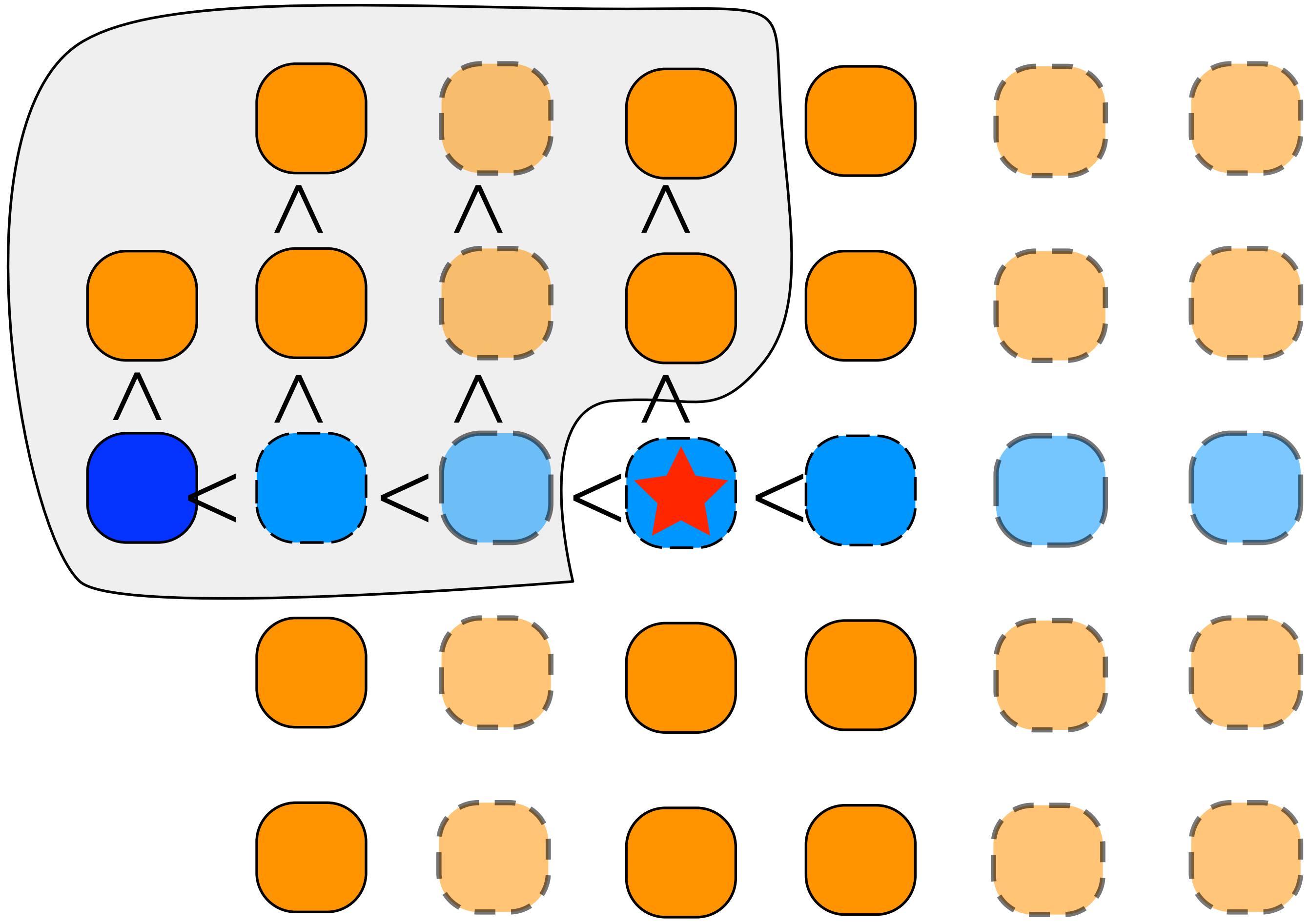


smaller than our partition element

this box contains at least many elements

med of med that we use as the partition element.

a nice property of our partition

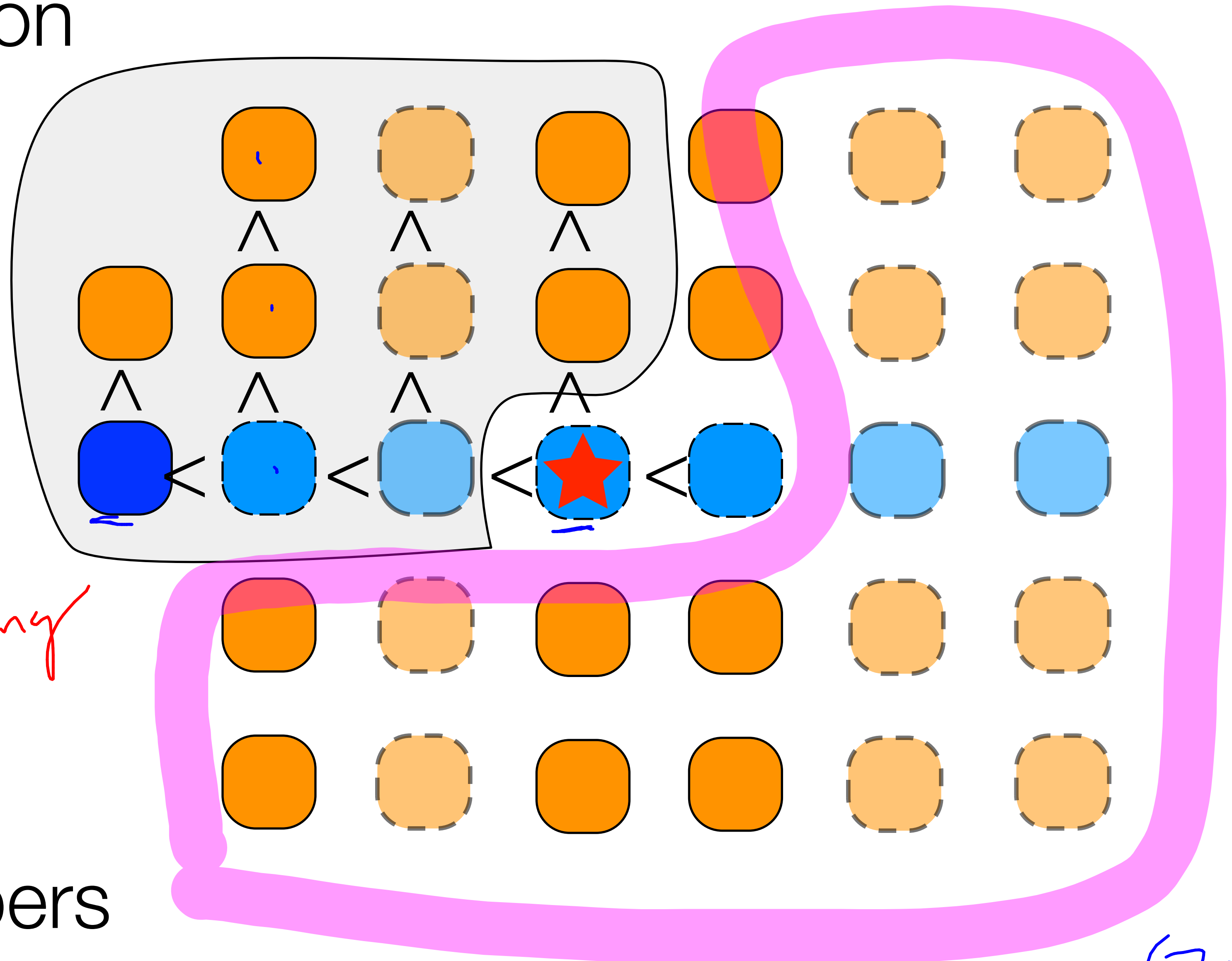


a nice property of our partition

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

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

box has this many



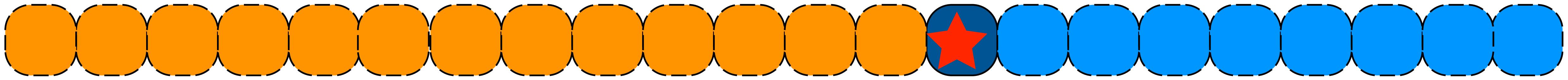
this implies there are at most $\frac{7n}{10}+6$ numbers larger than ★ /smaller

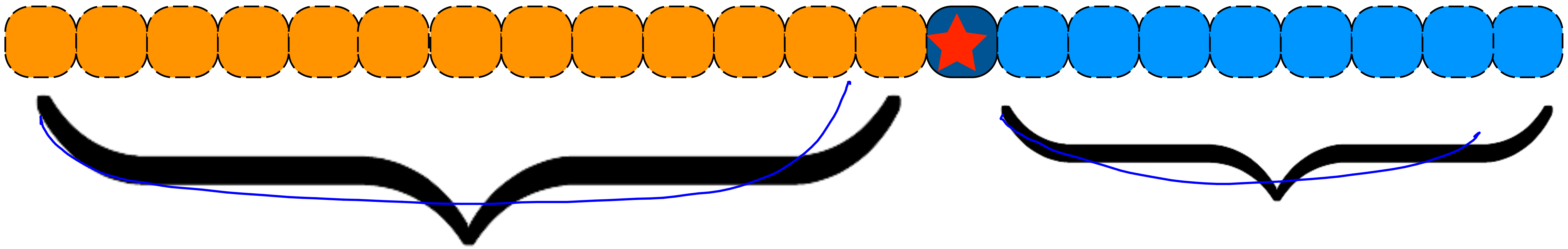
can have at most $\left(\frac{7n}{10}+6\right)$ elements

$$n - \left(\frac{3n}{10}-6\right)$$

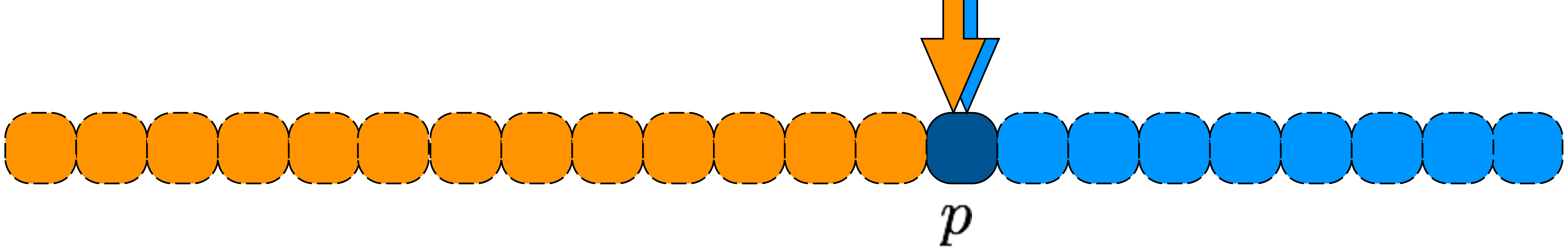
$$= \frac{7n}{10}+6$$

a nice property of our partition

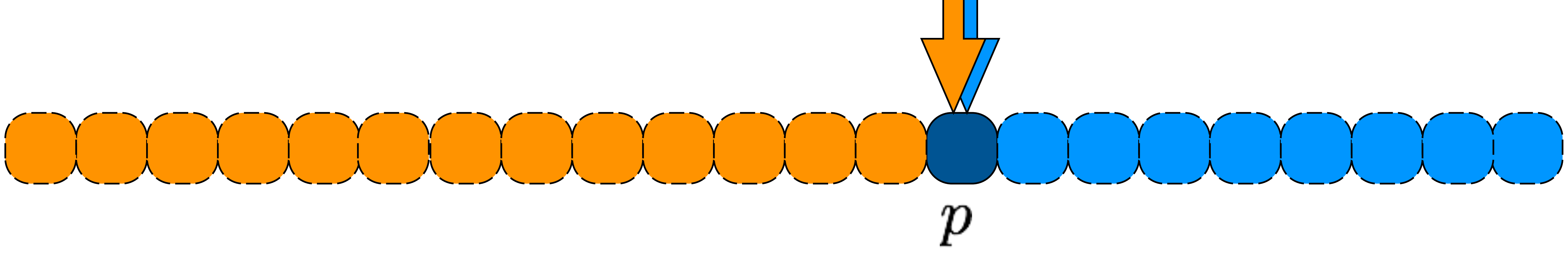


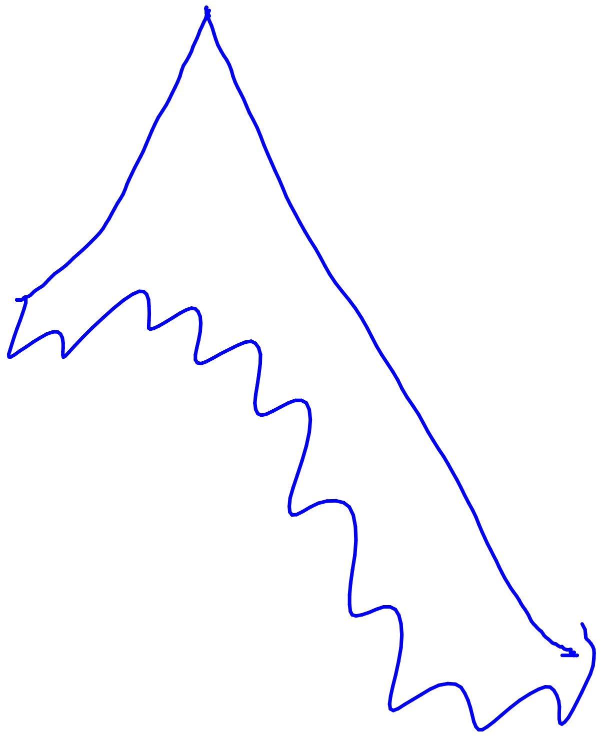


$$\le \frac{7n}{10} + 6 \qquad\qquad \le \frac{7n}{10} + 6$$



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





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

handle base case for small list

else pivot = FindPartitionValue(A,n) $\longrightarrow P(n)$

partition list about pivot $\longrightarrow \Theta(n)$

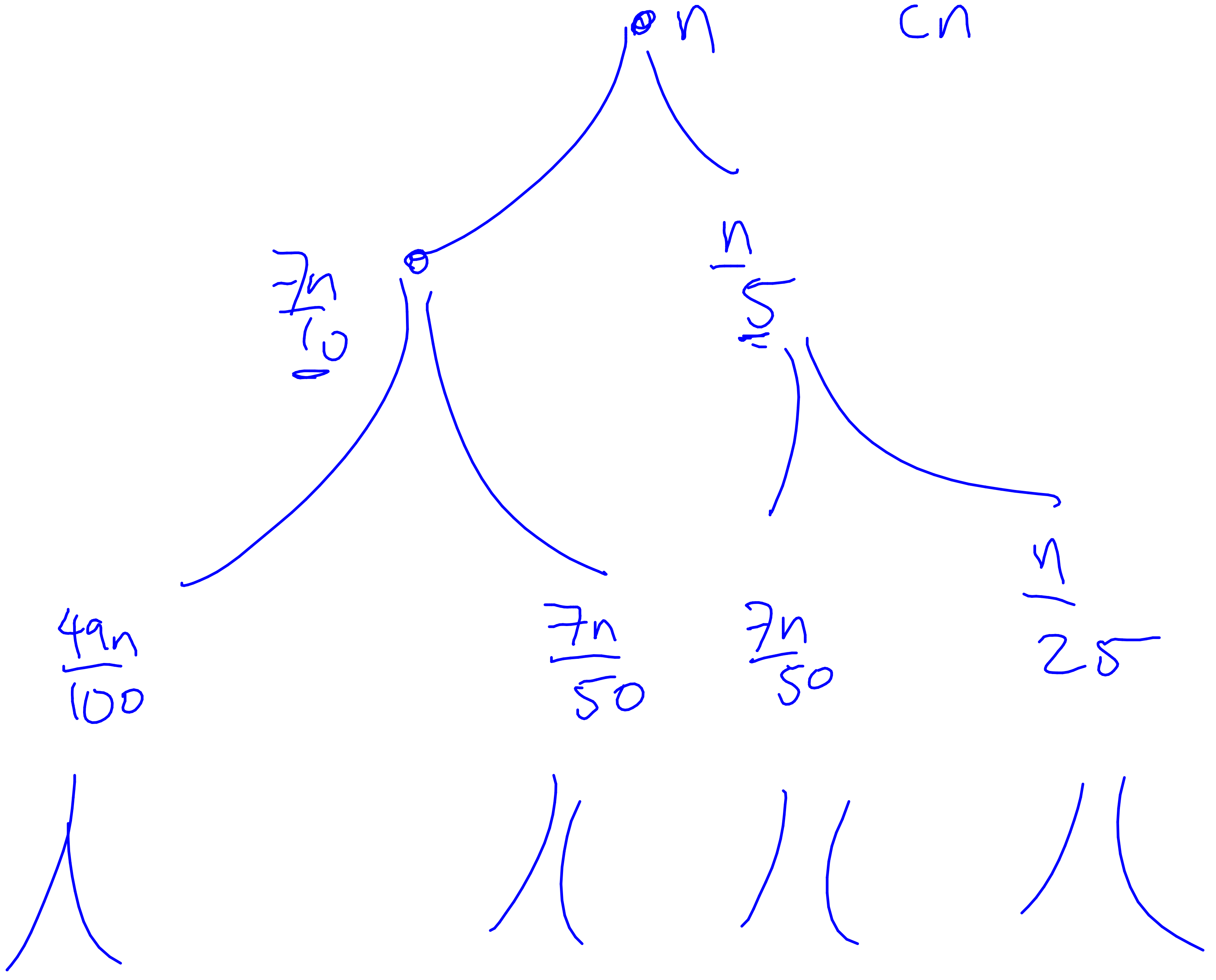
if pivot is position **i**, return pivot

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

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

$$S(n) = S\left(\frac{7n}{10} + 6\right) + P(n) + \Theta(n) = S\left(\frac{7n}{10} + 6\right) + S\left(\frac{n}{5}\right) + \Theta(n)$$

$$P(n) = S\left(\frac{n}{5}\right) + \Theta(n)$$



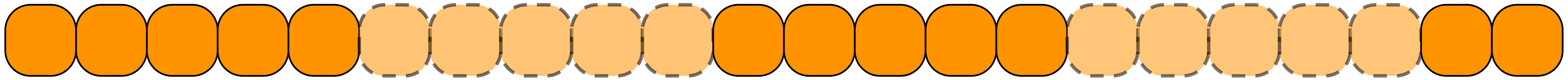
$$\underline{n}$$

$$\underbrace{\frac{7n}{10} + \frac{n}{5}}$$

$$\left(\frac{7}{10} + \frac{1}{5}\right)^2 \cdot n$$

$$\left(\frac{7}{10} + \frac{1}{5}\right)^3 \cdot n$$

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



divide list into groups of 5 elements

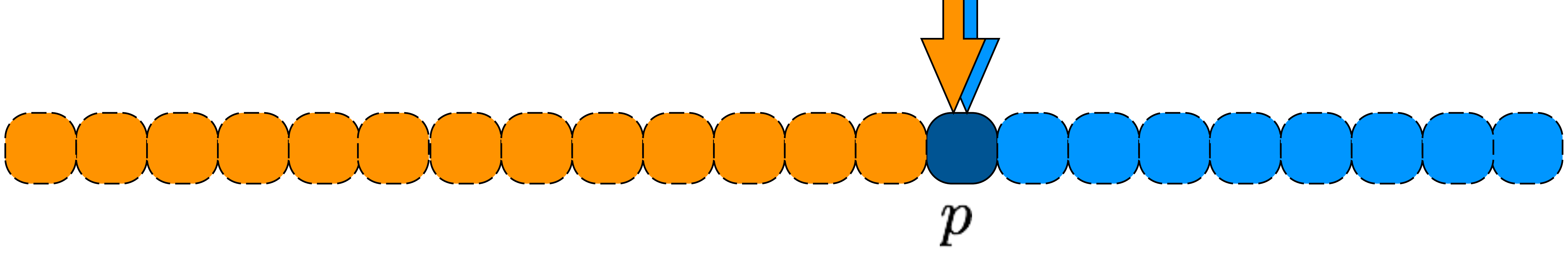
find median of each small list

gather all medians

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

return the result

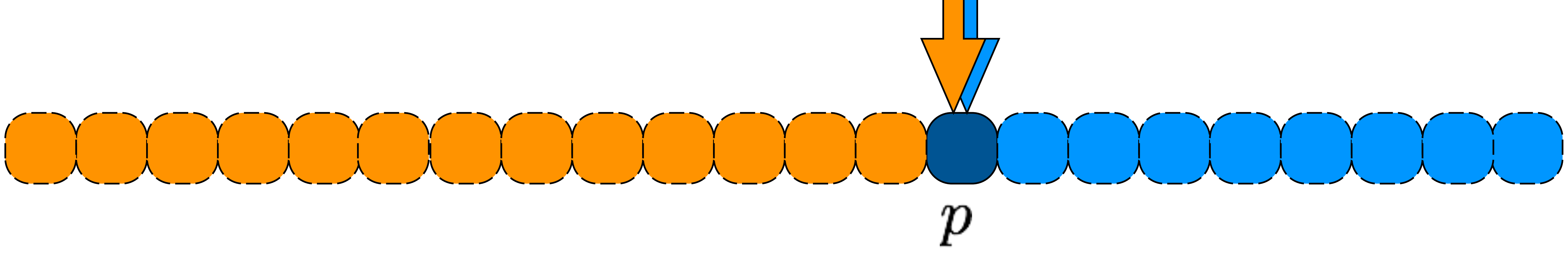
$$P(n) = S(\lceil n/5 \rceil) + O(n)$$



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

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



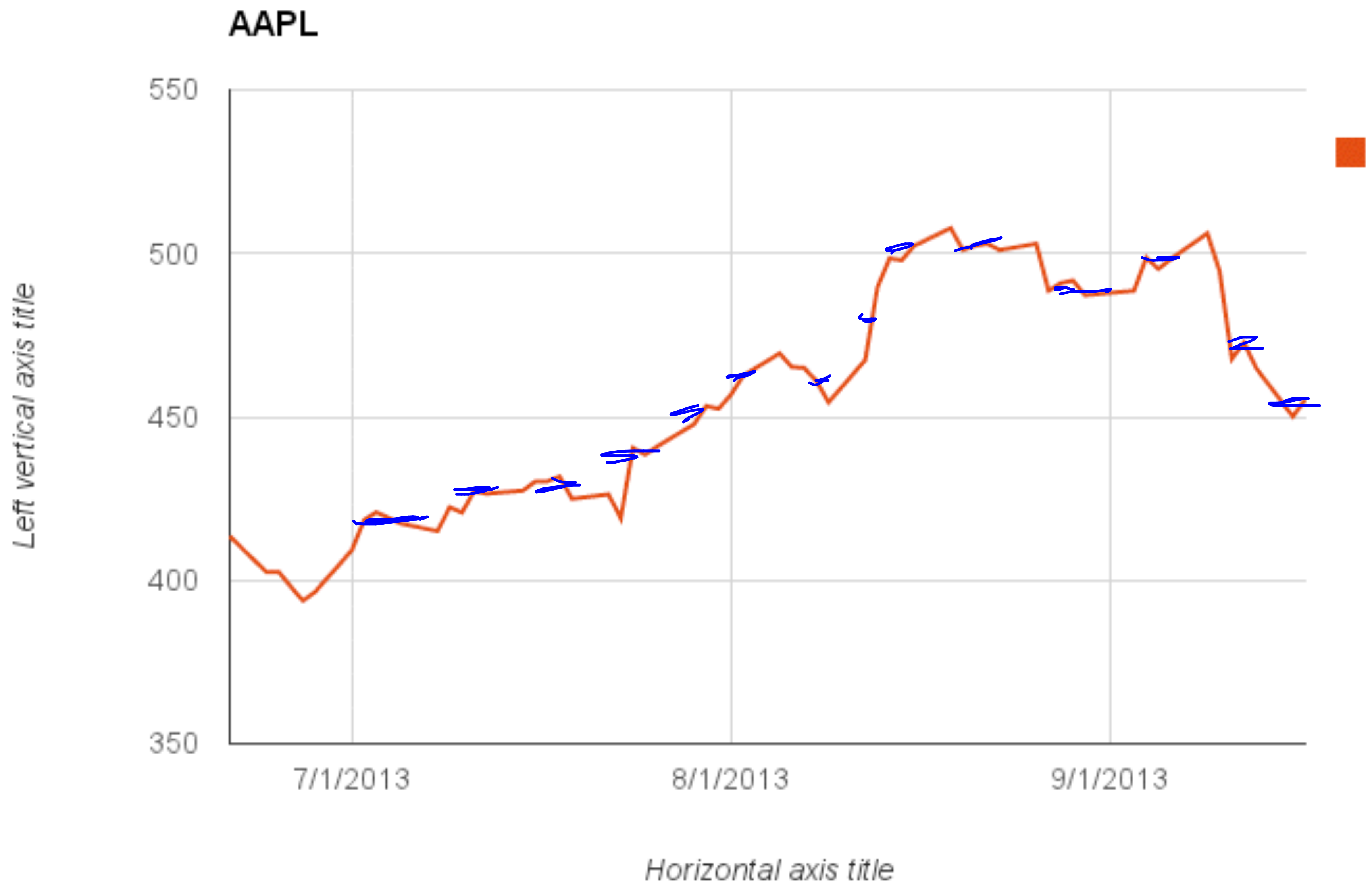
select $(i, A[1, \ldots, n])$

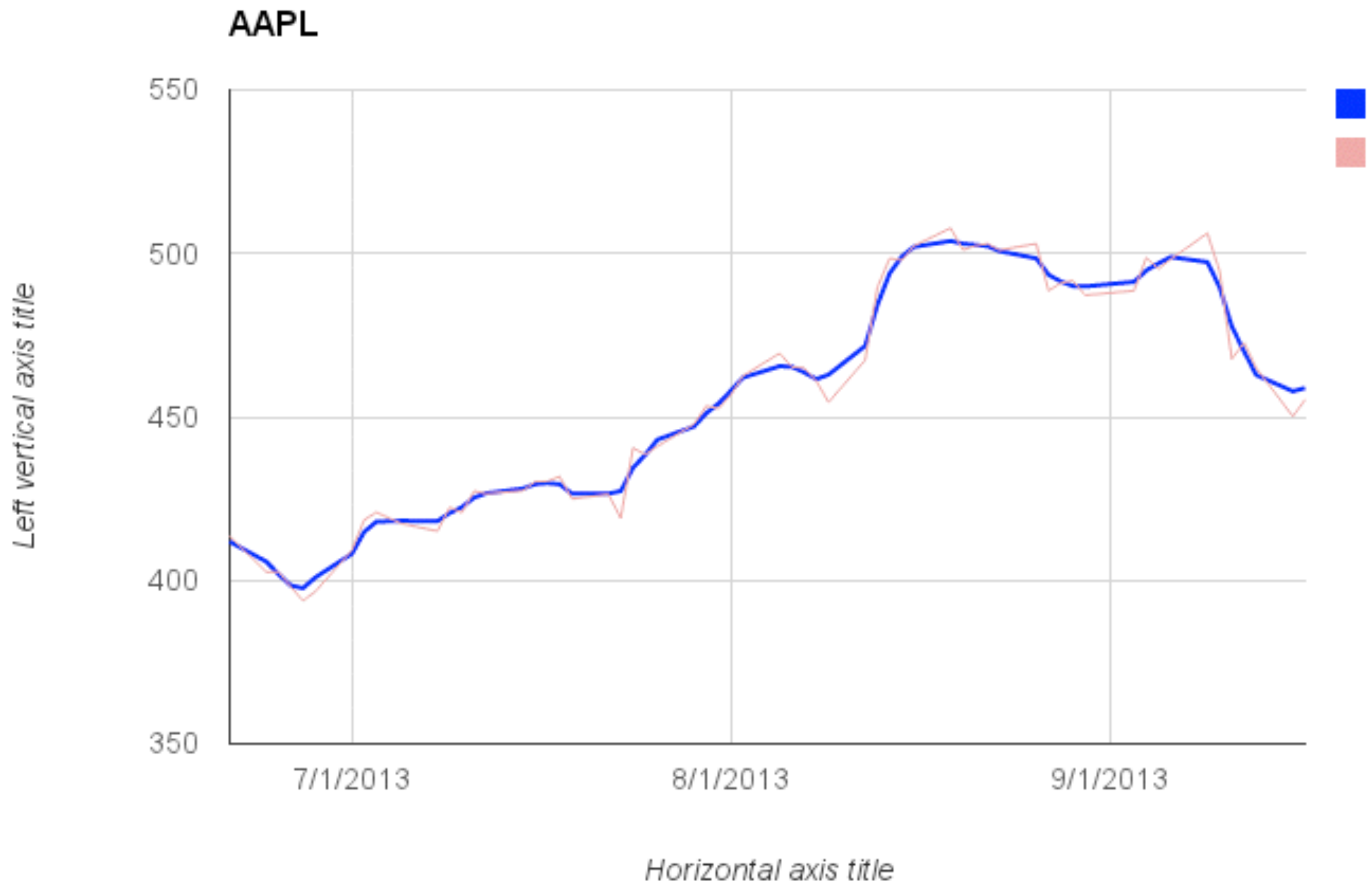
handle base case for small list
else pivot = FindPartitionValue(A,n)
partition list about pivot
if pivot is position **i**, return pivot
else if pivot is in position > **i**   select $(i, A[1, \ldots, p-1])$
else select $((i-p-1), A[p+1, \ldots, n])$

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

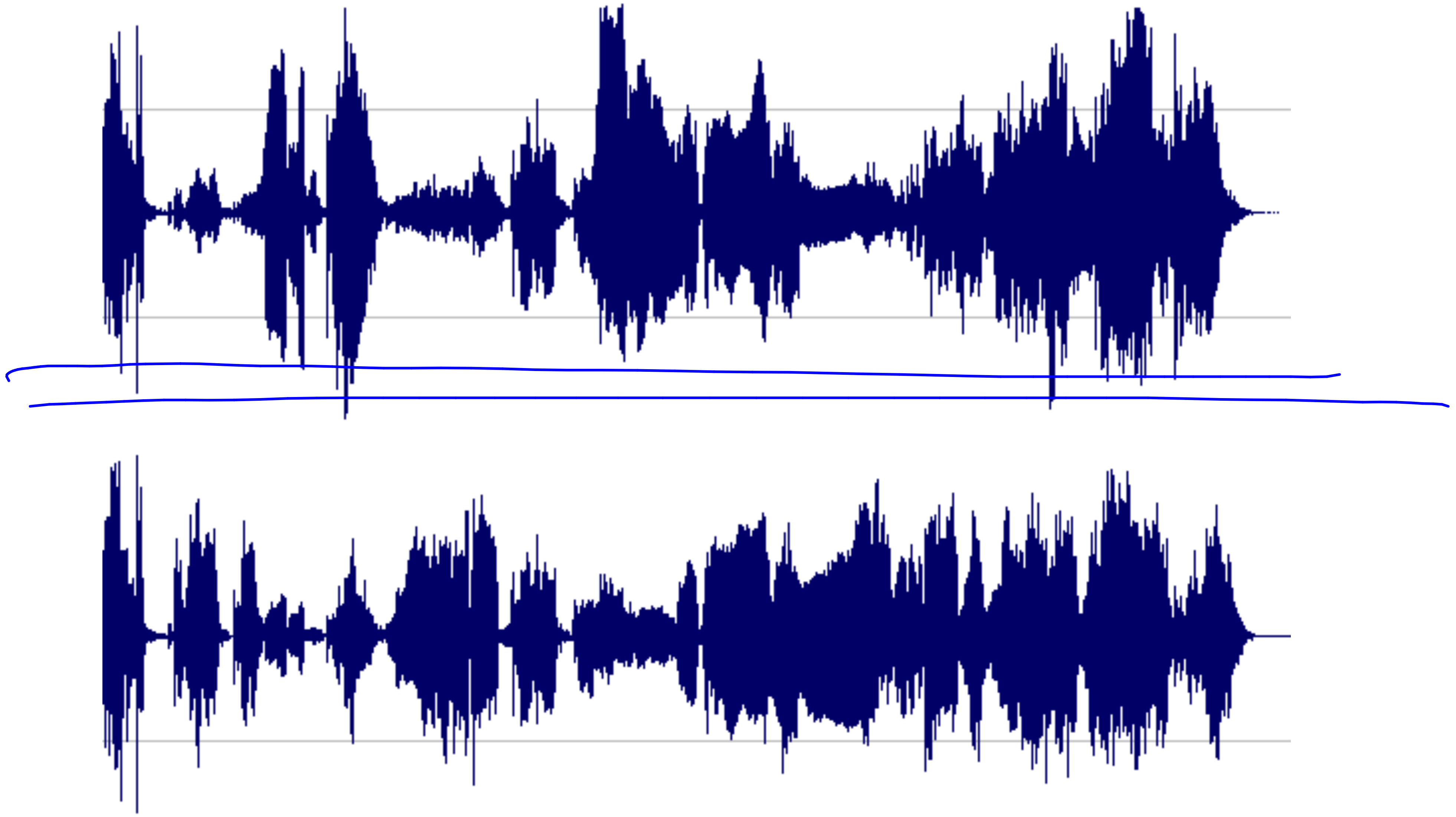
$$\Theta(n)$$

Fast Fourier Transform


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



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



big ideas:

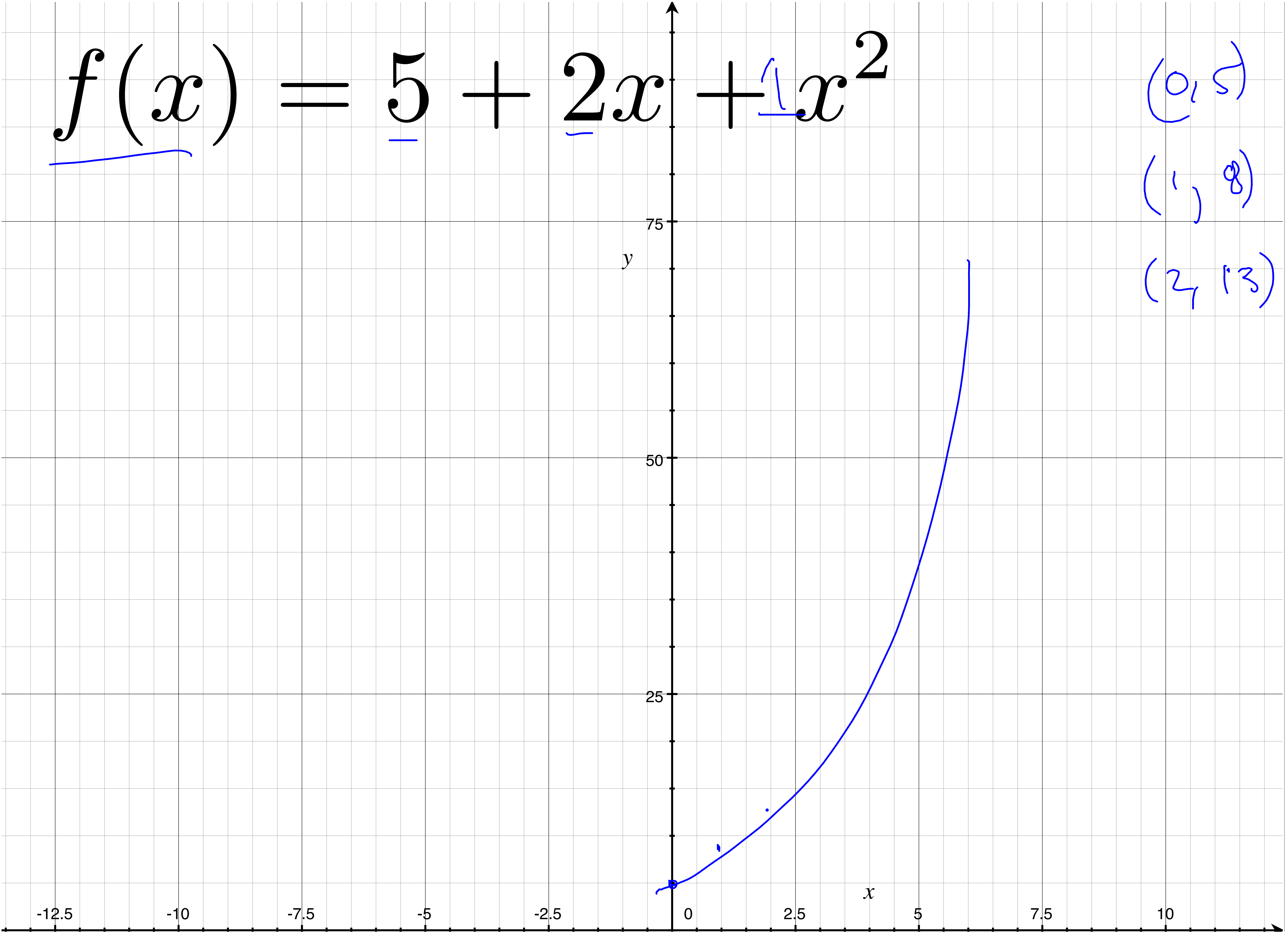
"change of representation
using divide & conquer"

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

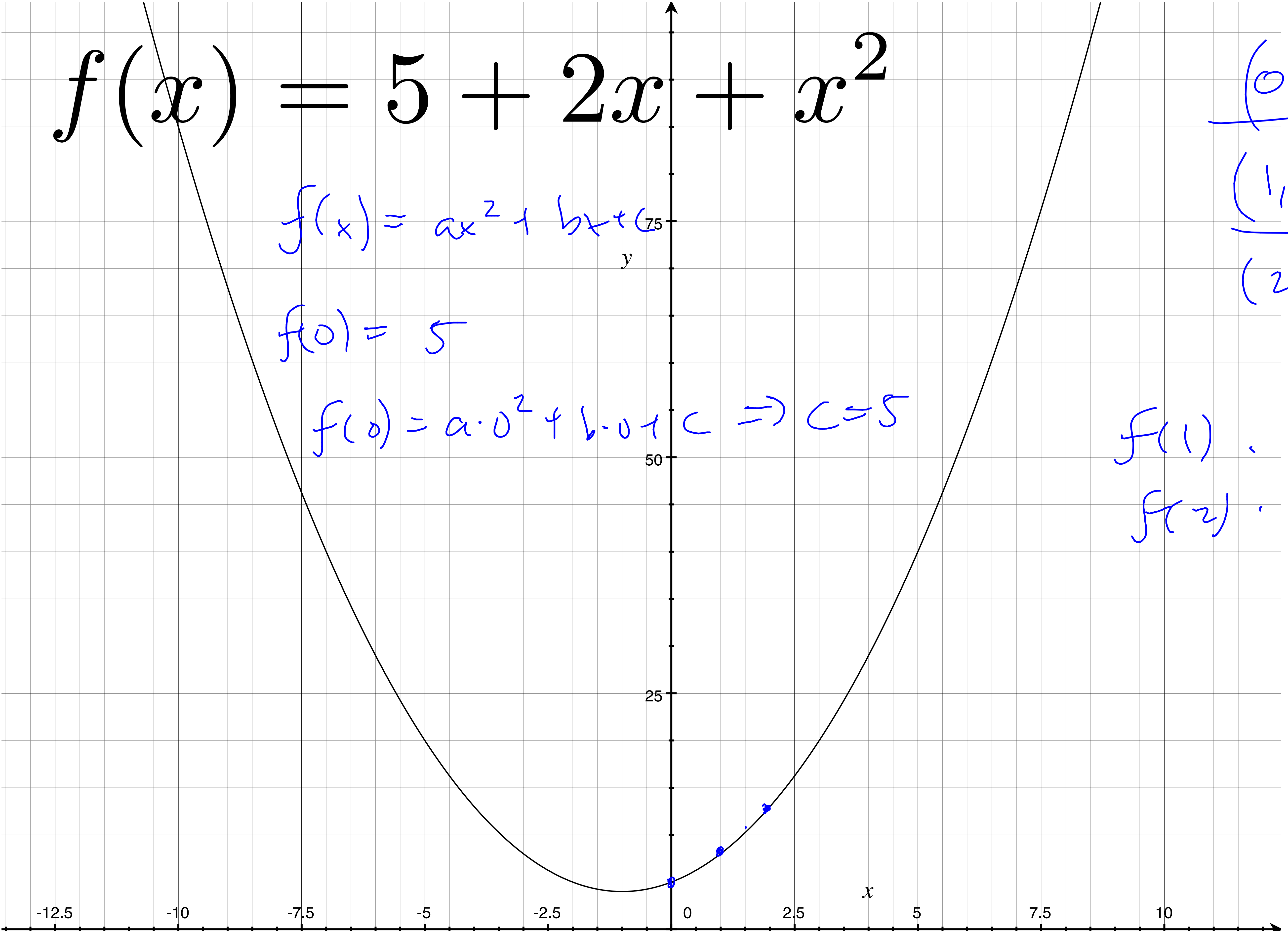
(0, 5)

(1, 8)

(2, 13)



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



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

$f(0) = 5$

$f(0) = a \cdot 0^2 + b \cdot 0 + c \Rightarrow c = 5$

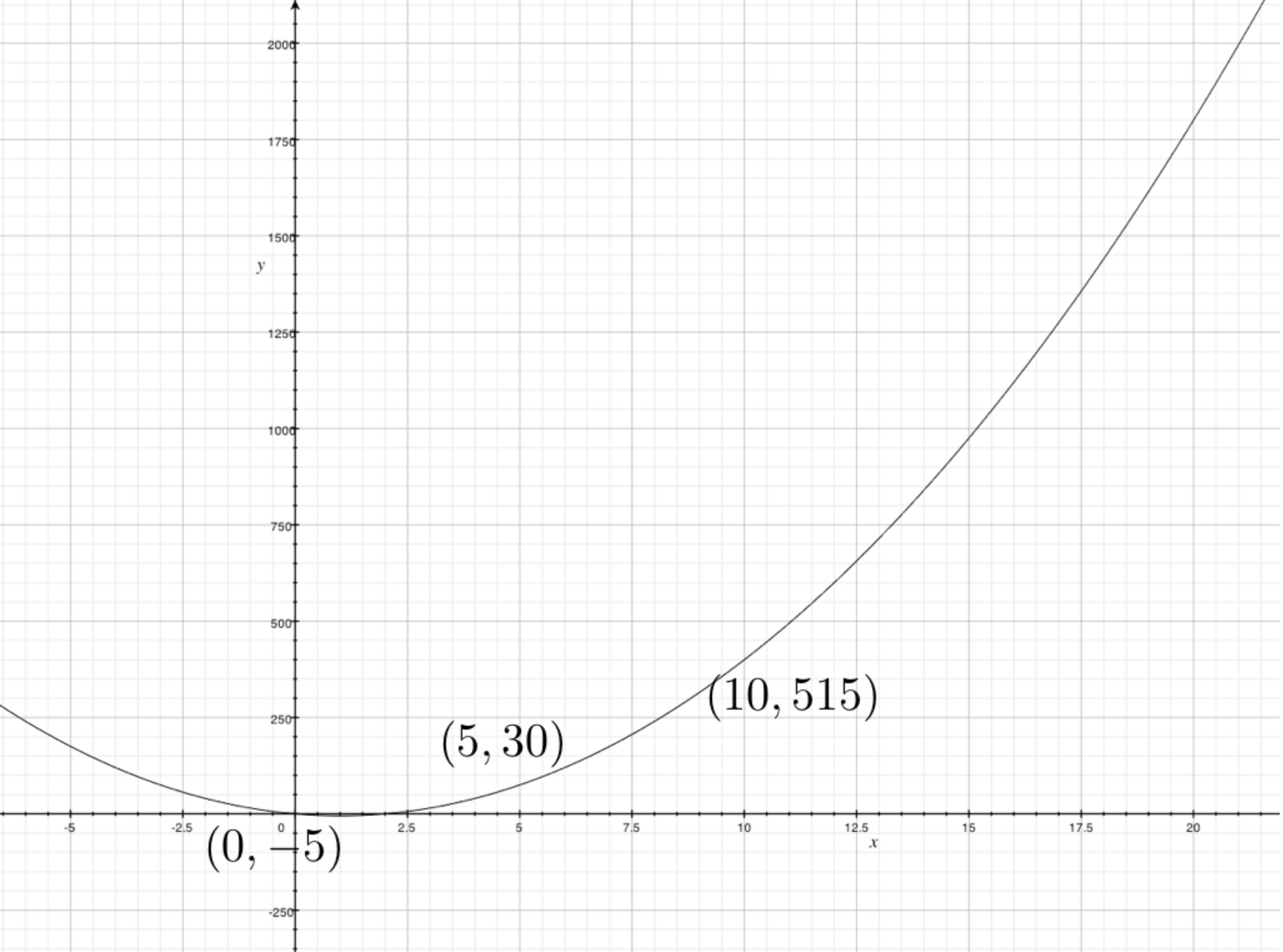
(0,5)

(1,8)

(2,13)

$f(1)$.

$f(2)$.

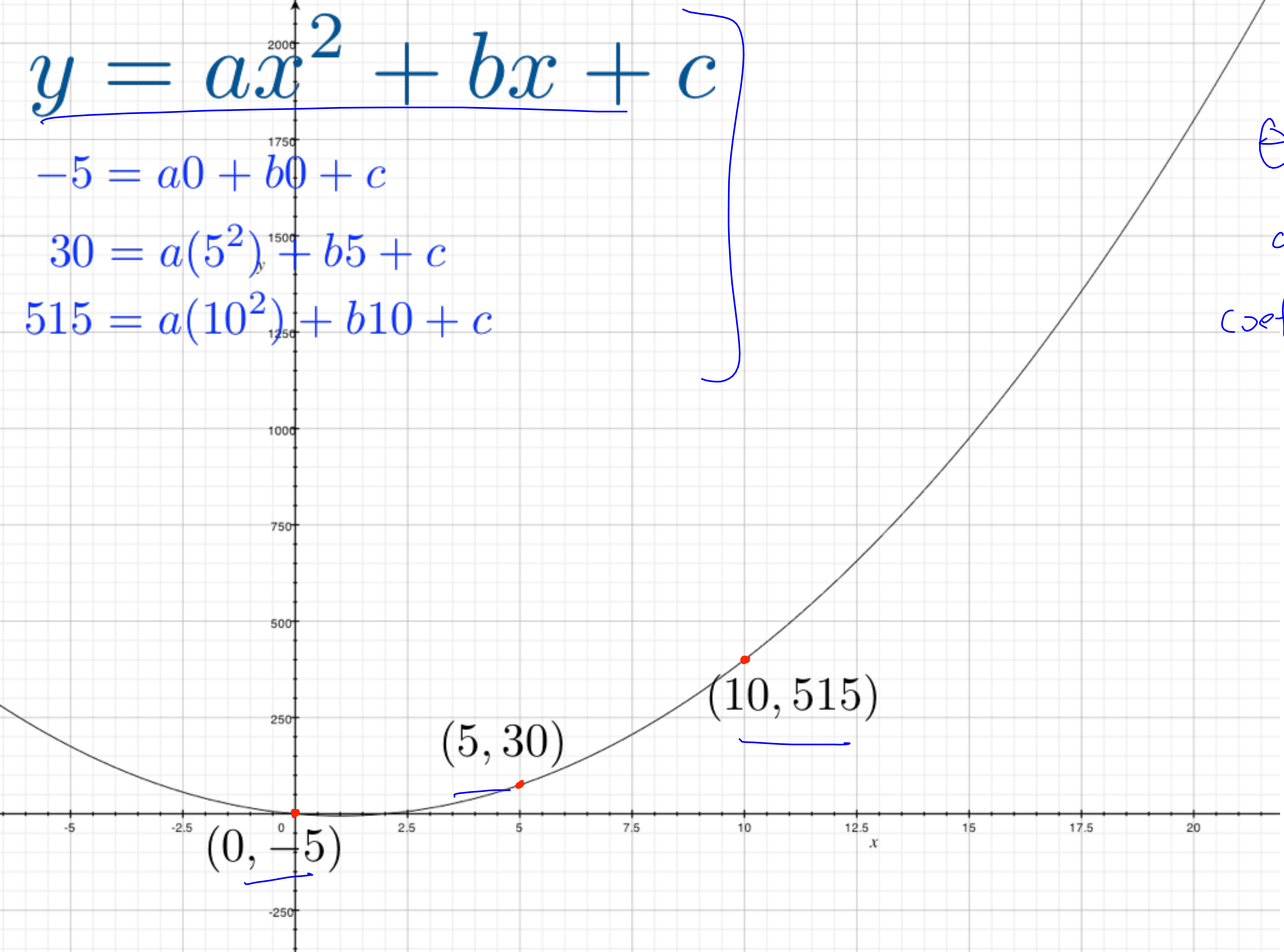

2000
1750
1500
1250
1000
750
500
250
-250
y
-5
-2.5
0
2.5
5
7.5
10
12.5
15
17.5
20
x
(10, 515)
(5, 30)
(0, −5)

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

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

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

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



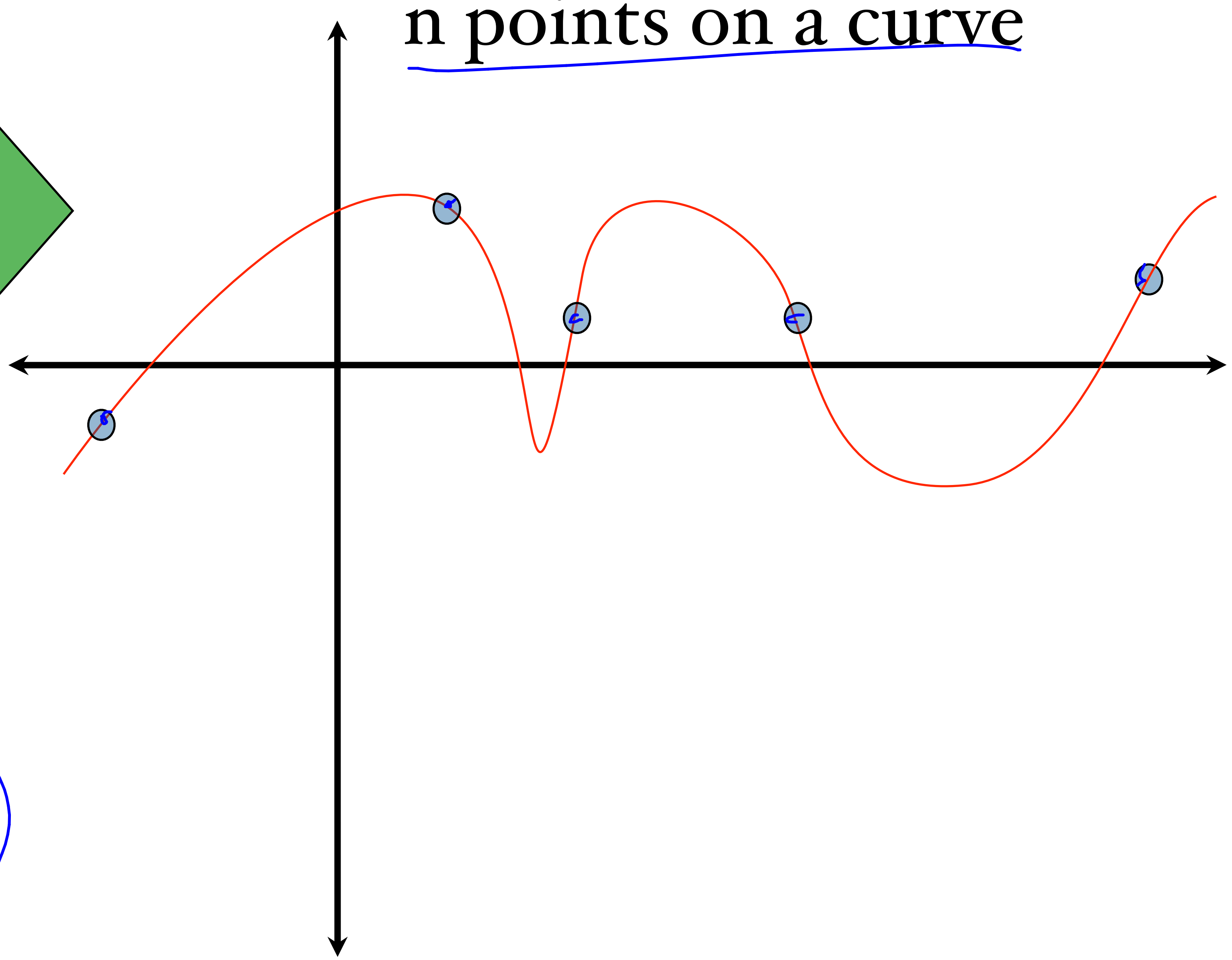
$\Theta(n^2)$ time to change from coeff → point rep.

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

degree $n-1$ polynomial

$A(x)$

n points on a curve



naively takes $\Theta(n^2)$

the FFT does it in a clever way in $\Theta(n \log n)$ time.

FFT

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

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

output:

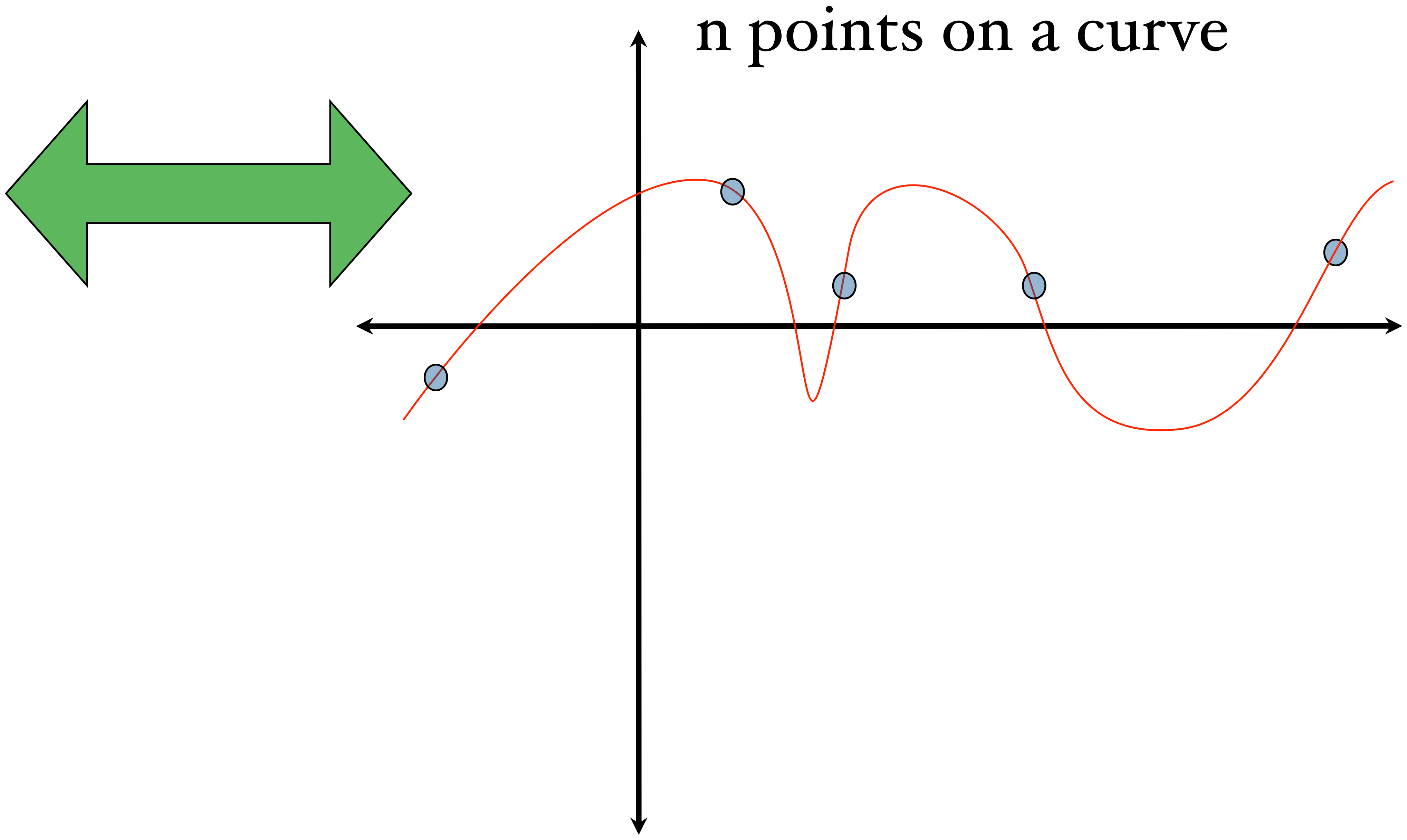
FFT

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.

$A(x)$



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

Inverse FFT: Given n-points,

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:

$\Theta(n^2)$ time.

solve the large problem by

solving smaller problems
and combining solutions

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

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

$$A_e(\cancel{x}\,y) = a_0 + a_2 y + a_4 y^2 + \ldots\ a_{n-2}\, y^{n/2-1} \qquad \tfrac{n}{2} \text{ terms}$$

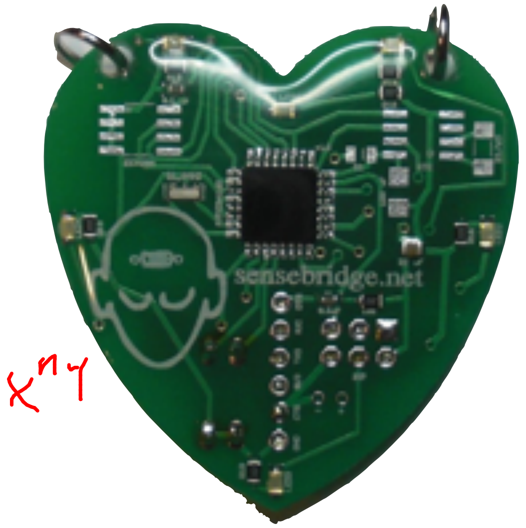
$$A_o(\cancel{x}\,y) = a_1 + a_3 y + a_5 y^2 + \ldots\ a_{n-1}\, y^{n/2-1} \qquad \tfrac{n}{2} \text{ terms}$$

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

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

$$+\ x\,(a_1 + a_3 x^2 + a_5 x^4 + \ldots + a_{n-1} x^{n-2})$$

$$= a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1}$$



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

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

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

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

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

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

degree $\frac{n}{2}$

n pts

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

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

$A(3) = \ldots$

$A(-3) = \ldots$

$A(4) = \ldots$

$A(-4) = \ldots$

$A(5) = \ldots$

$A(-5) = \ldots$

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

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

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

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

Last remaining issue:

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

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

$$x^n = 1$$

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

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

What is this number?

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

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

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

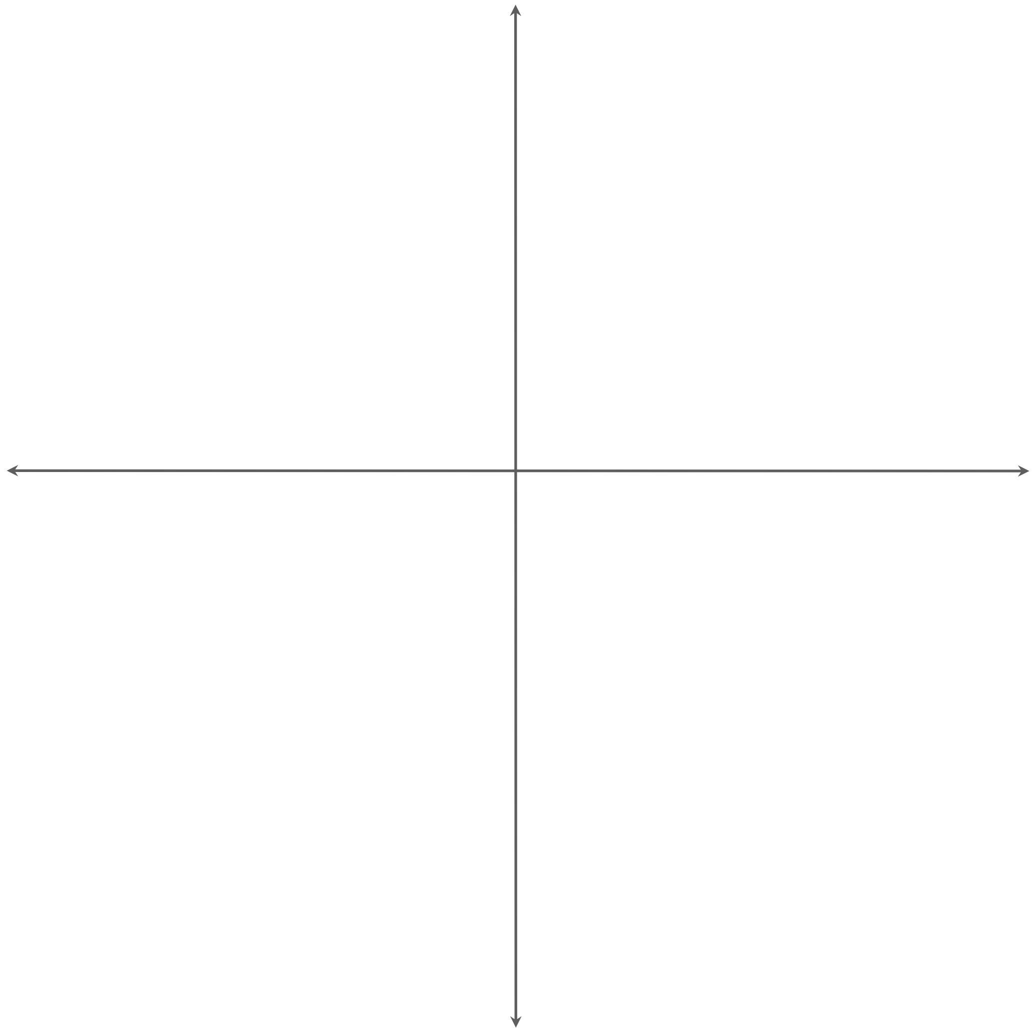
$$e^{2\pi ij/n} = \cos(2\pi j/n) + i\sin(2\pi j/n)$$

$e^{2\pi ij/n}$ = $\omega_{j,n}$ 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



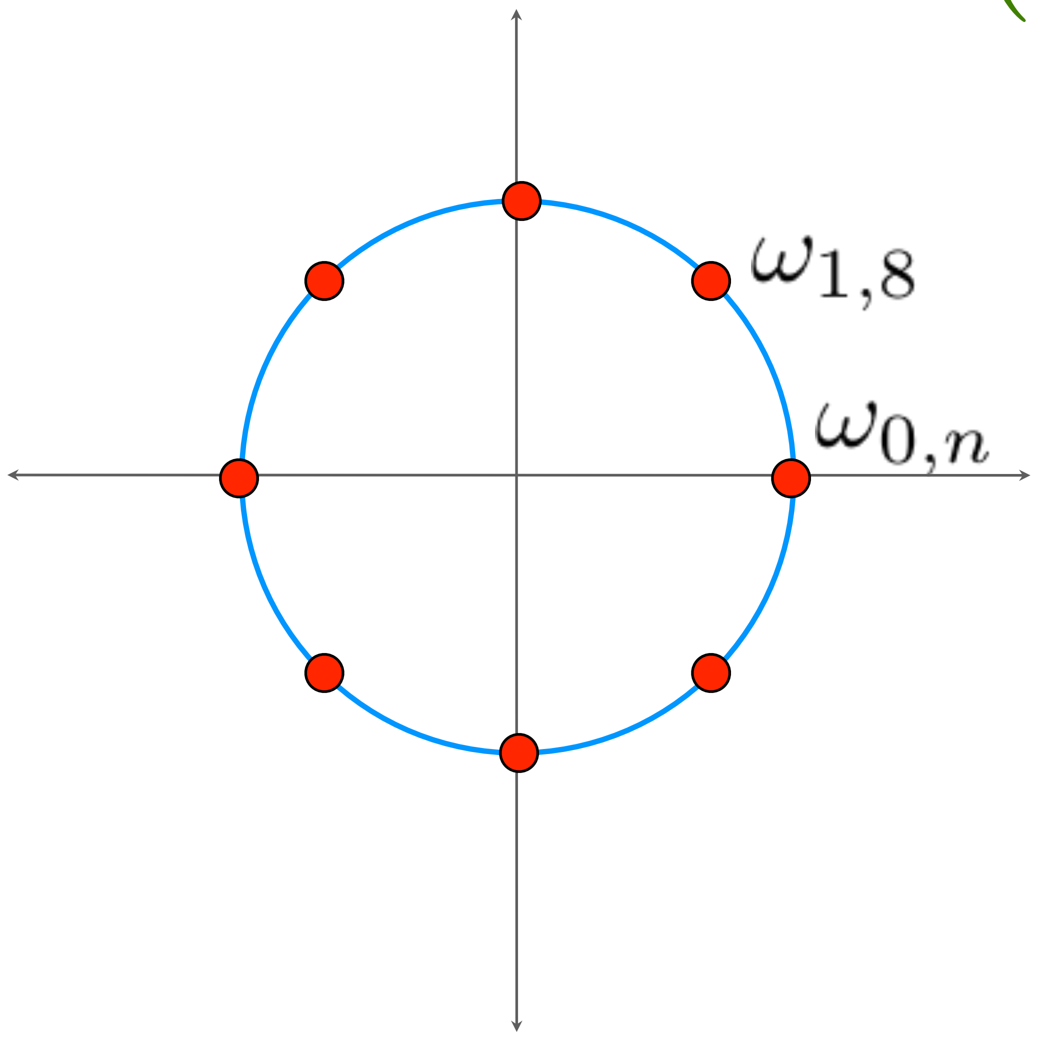
Then graph them

roots of unity

$$x^n = 1$$

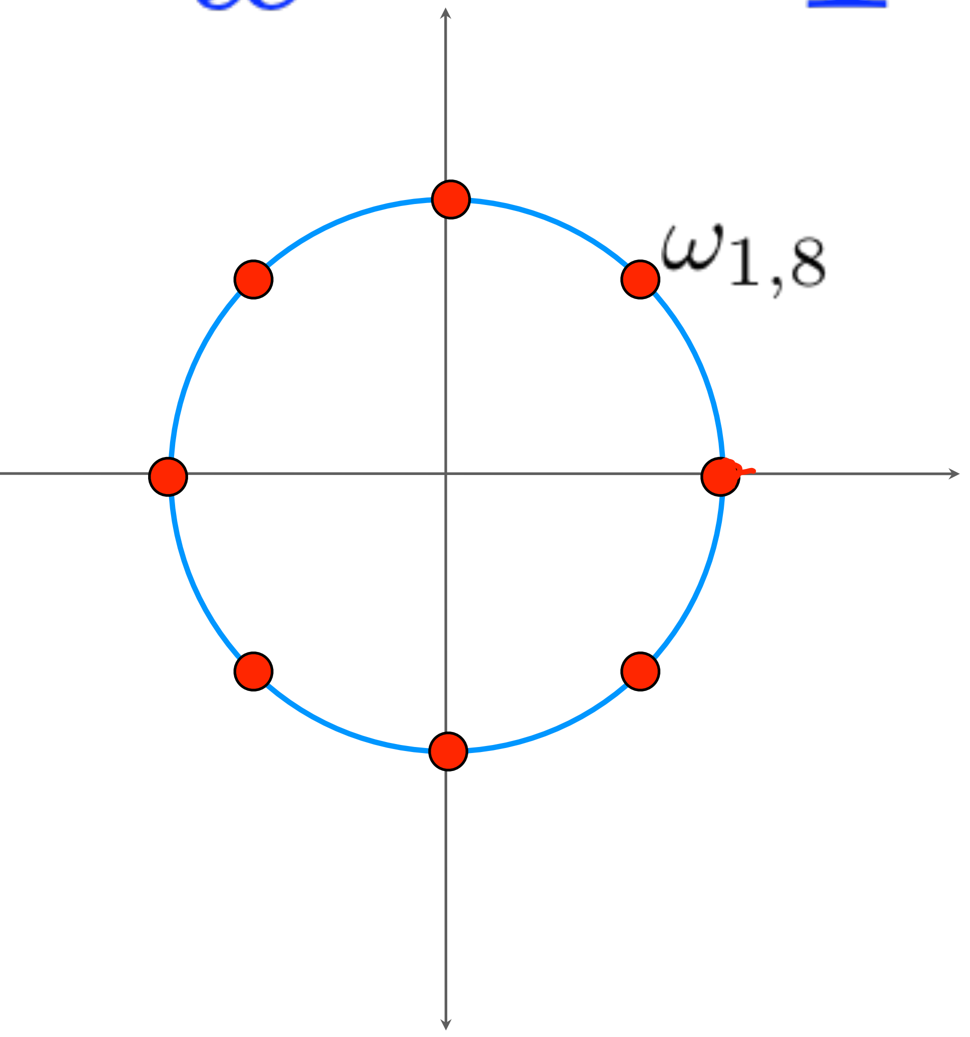
should have n solutions

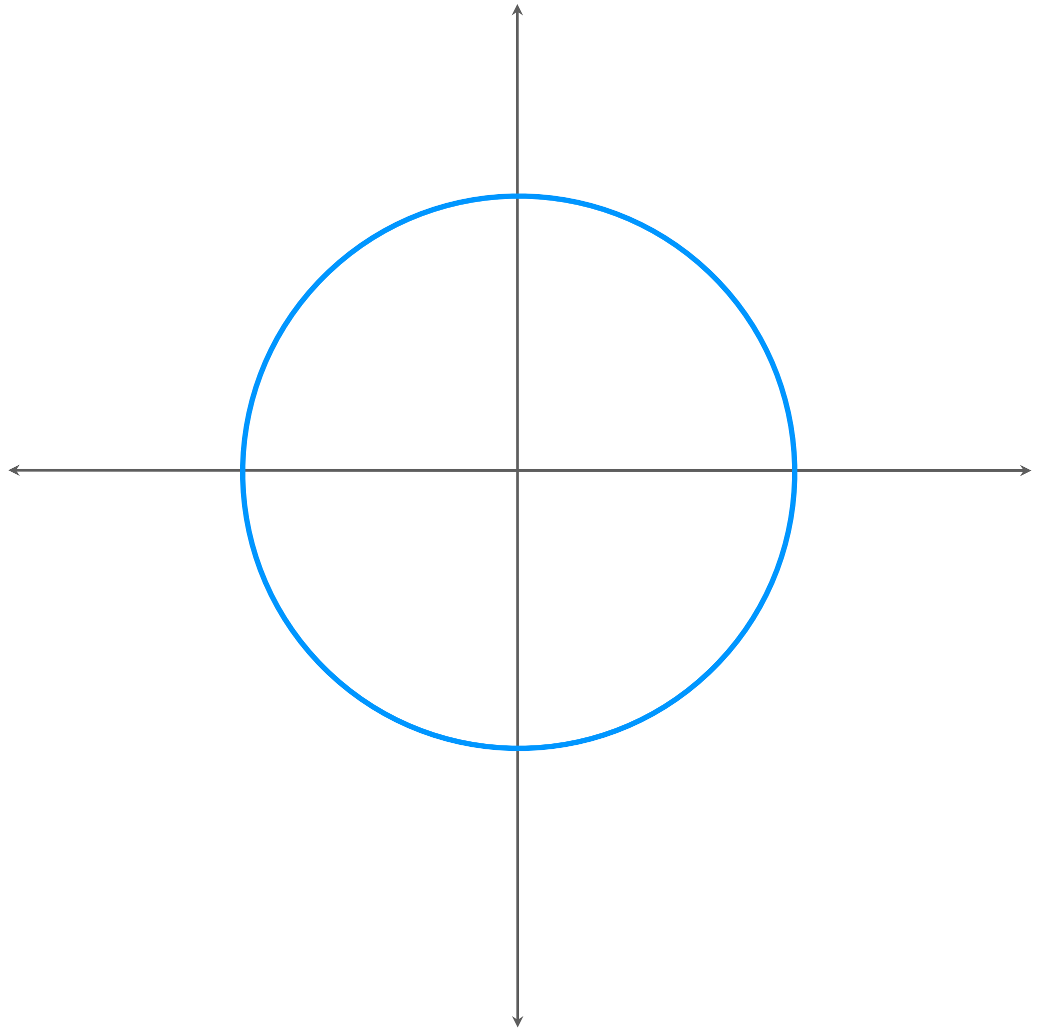
$$e^{2\pi ij/n} = \cos(2\pi j/n) + i\sin(2\pi j/n)$$



squaring the nth roots of unity

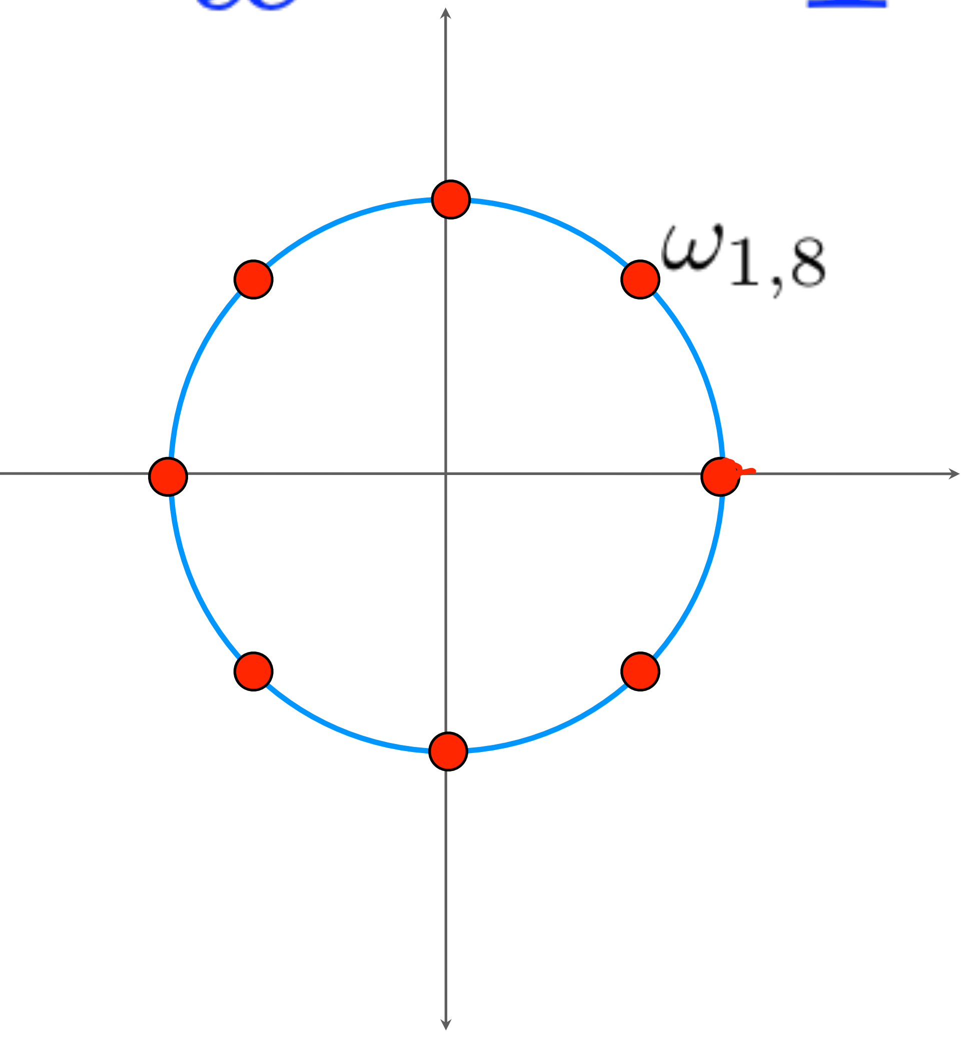
$$x^n = 1$$



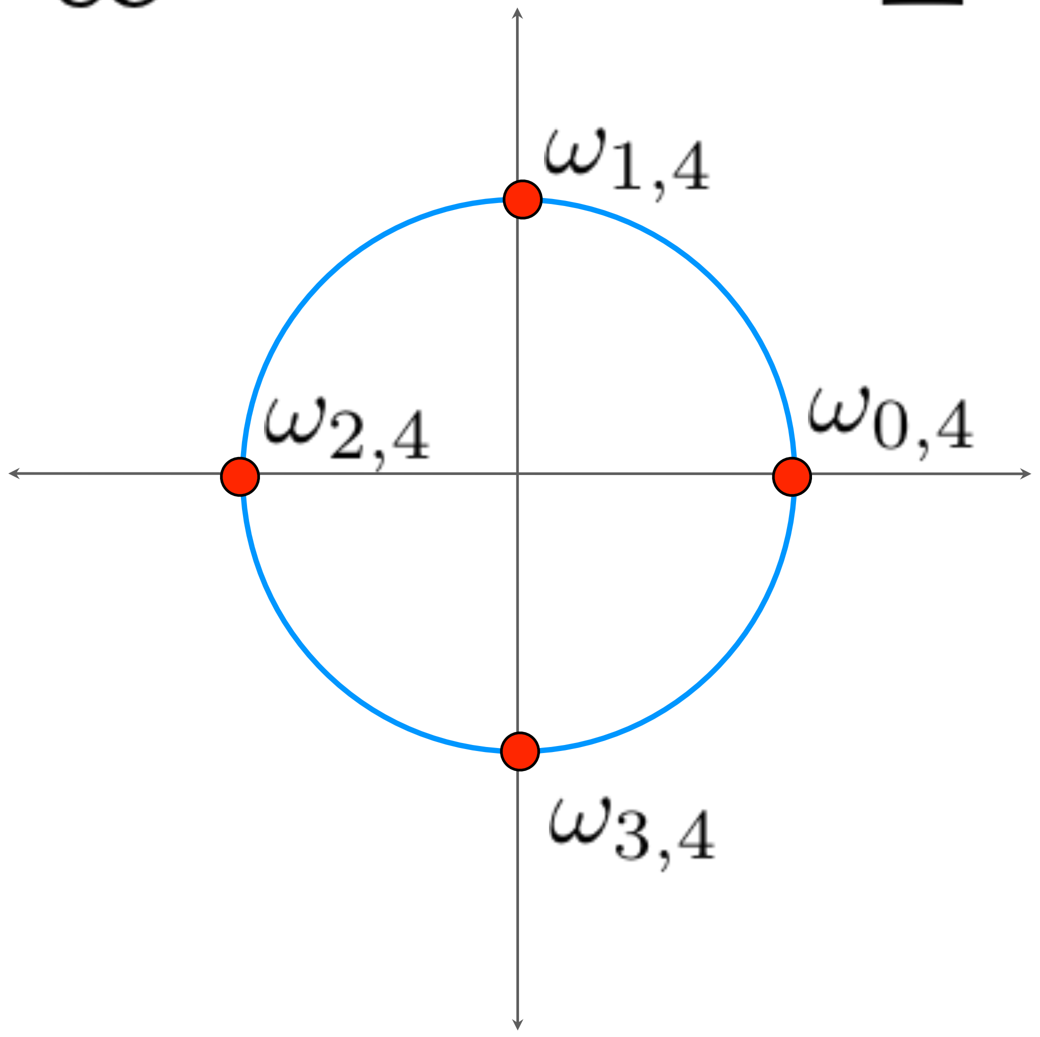


squaring the nth roots of unity

$$x^n = 1$$



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



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$^{\text{th}}$ root produces an n/2$^{\text{th}}$ root.

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

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

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

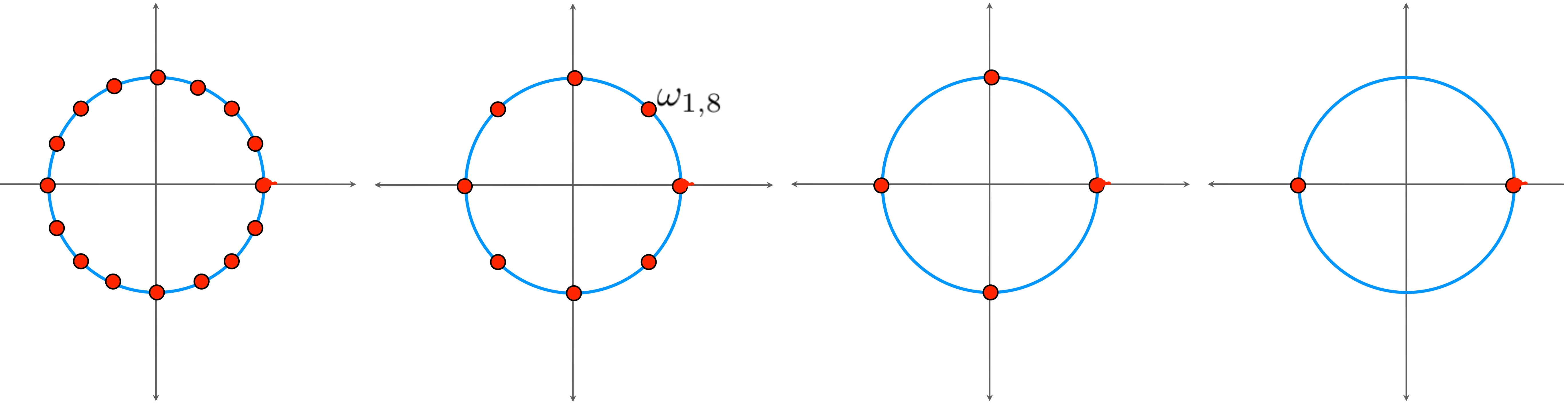
$1 \quad e^{2\pi i(1/(n/2))} \quad e^{2\pi i(2/(n/2))} \quad e^{2\pi i(3/(n/2))} \qquad 1$

$$e^{2\pi i((n/2)+1/(n/2))}$$

$$= e^{2\pi i(1+1/(n/2))}$$

$$= 1 \cdot e^{2\pi i(1/(n/2))}$$

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 n^{th} roots of unity

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

Base case if n<=2

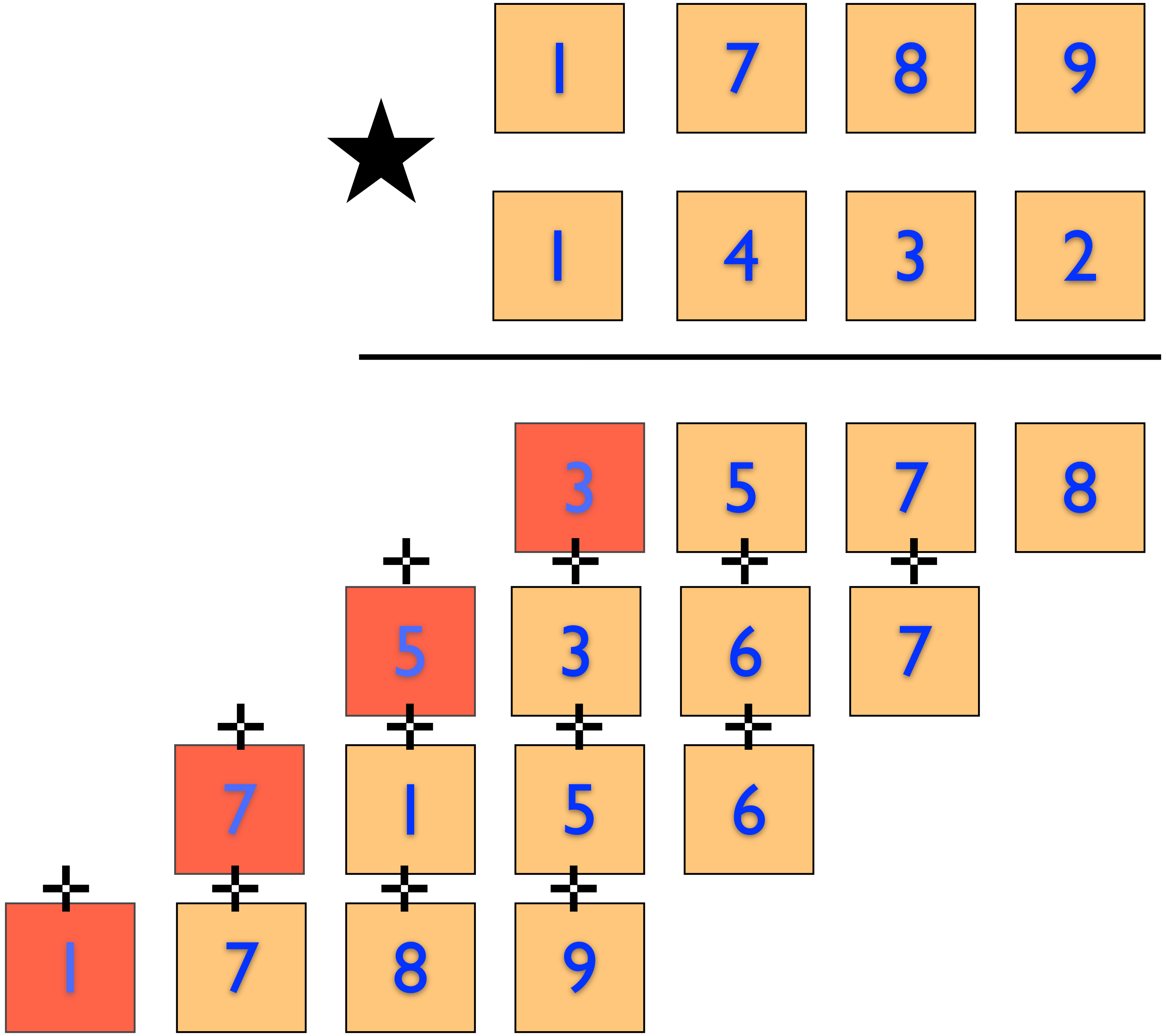
E[...] <- FFT(A_e) // eval Ae on n/2 roots of unity
O[...] <- FFT(A_o) // eval Ao on n/2 roots of unity

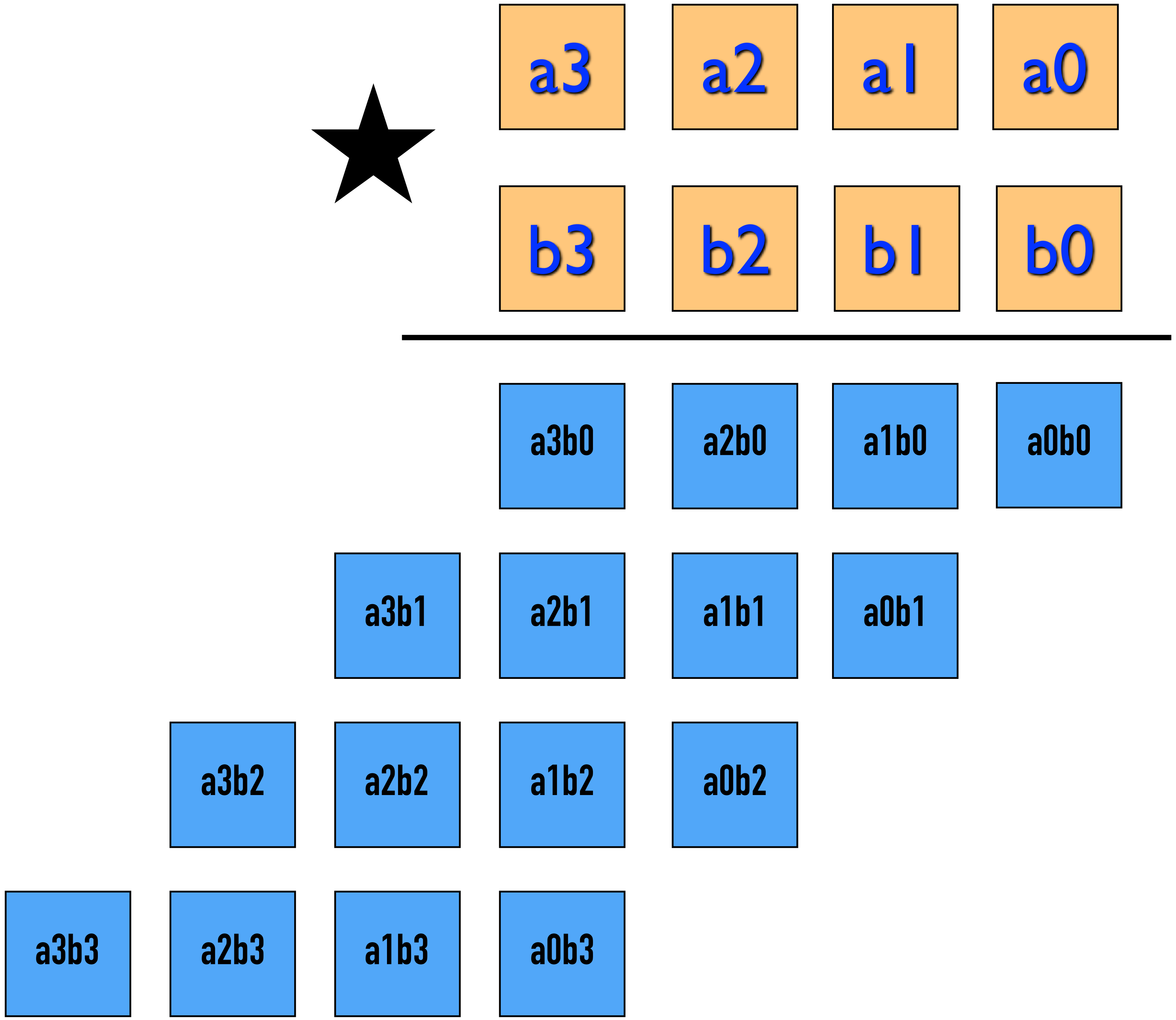
combine results using equation:

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

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

Return n points.


1 7 8 9
★ 1 4 3 2
3 5 7 8
+ + + +
5 3 6 7
+ + + +
7 1 5 6
+ + + +
1 7 8 9

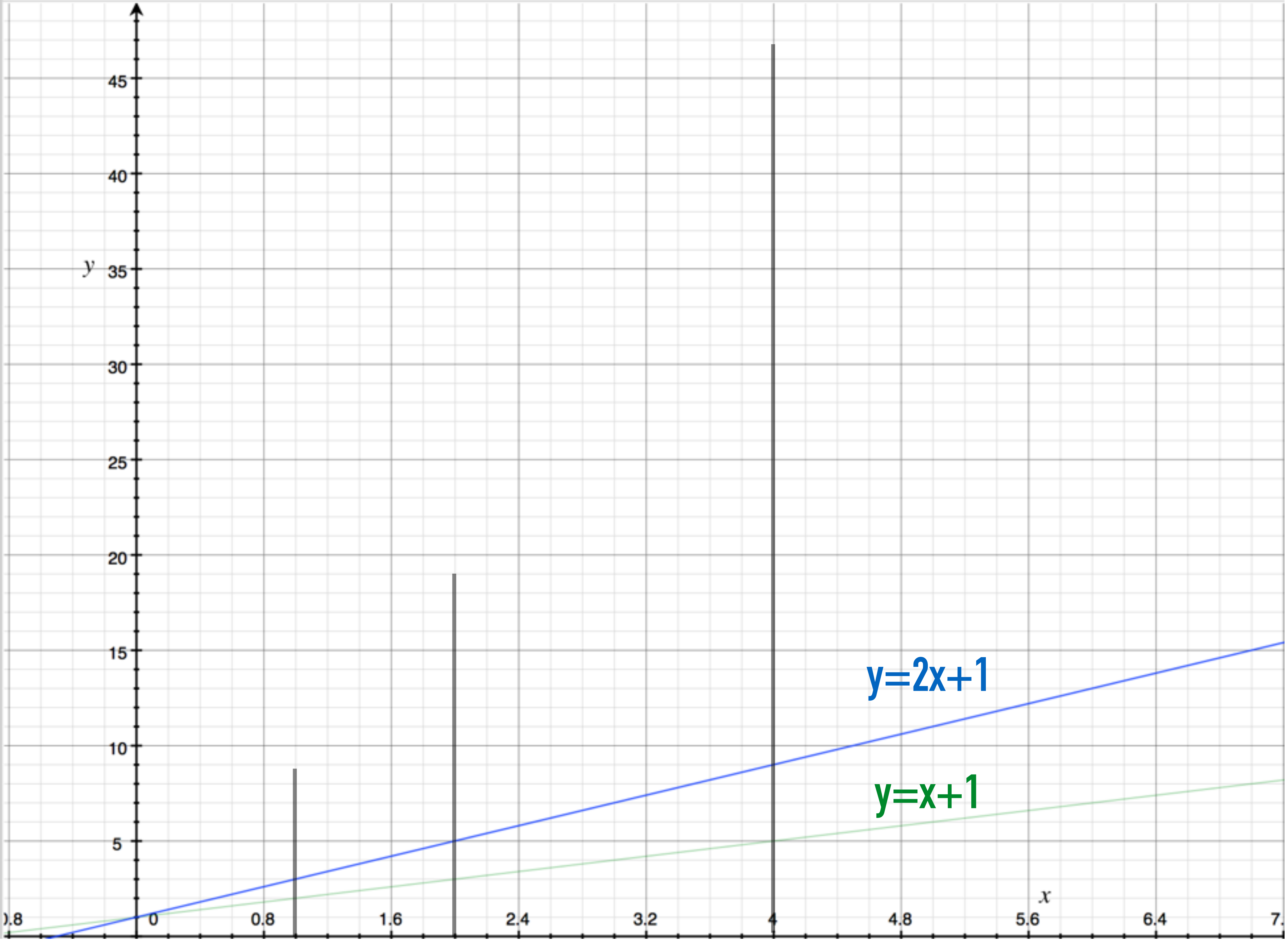

a3
a2
a1
a0
b3
b2
b1
b0
a3b0
a2b0
a1b0
a0b0
a3b1
a2b1
a1b1
a0b1
a3b2
a2b2
a1b2
a0b2
a3b3
a2b3
a1b3
a0b3

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

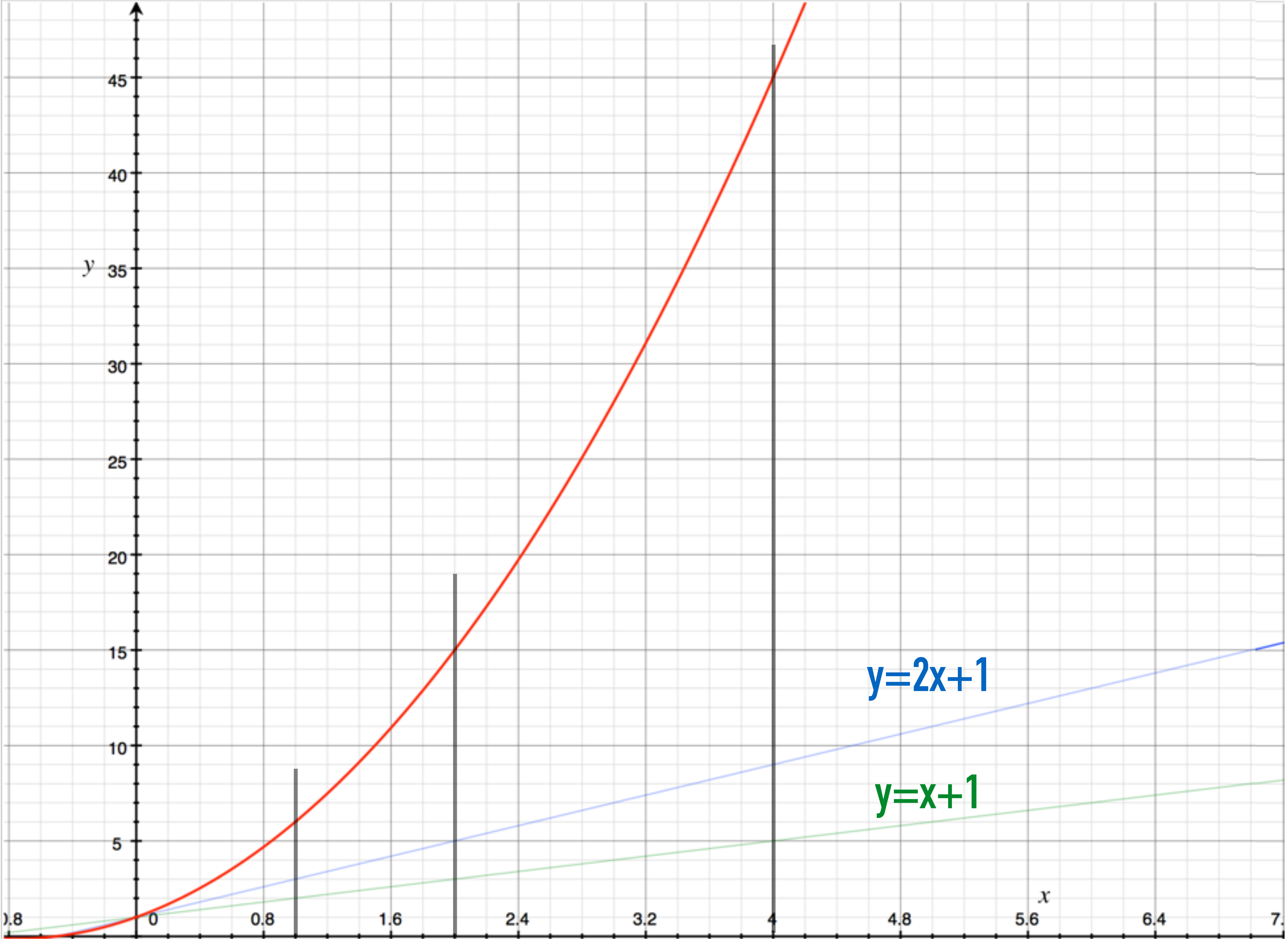
$$B(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

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

$y=2x+1$



$y = 2x^2 + 3x + 1$



a_3 a_2 a_1 a_0 $\star$ 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 $\star$ 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**

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

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**

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

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

application to mult

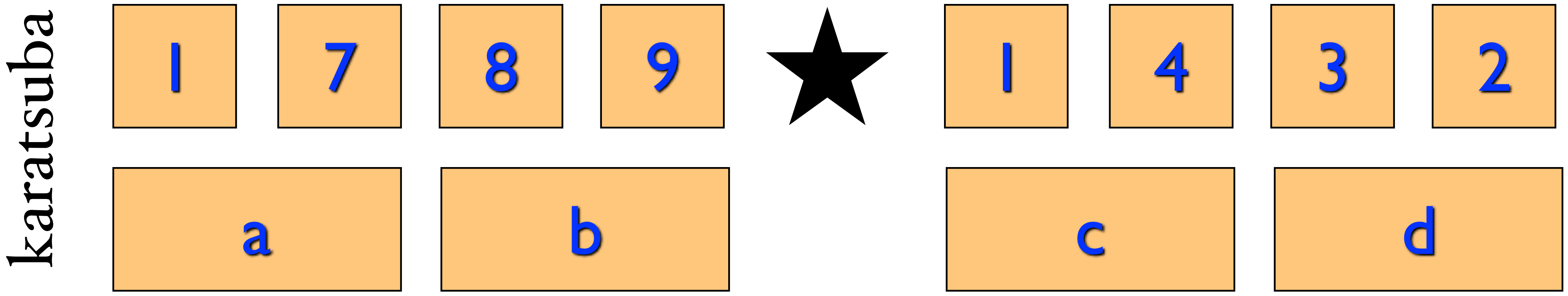
karatsuba

1 7 8 9 ★ 1 4 3 2

a b ★ c d

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

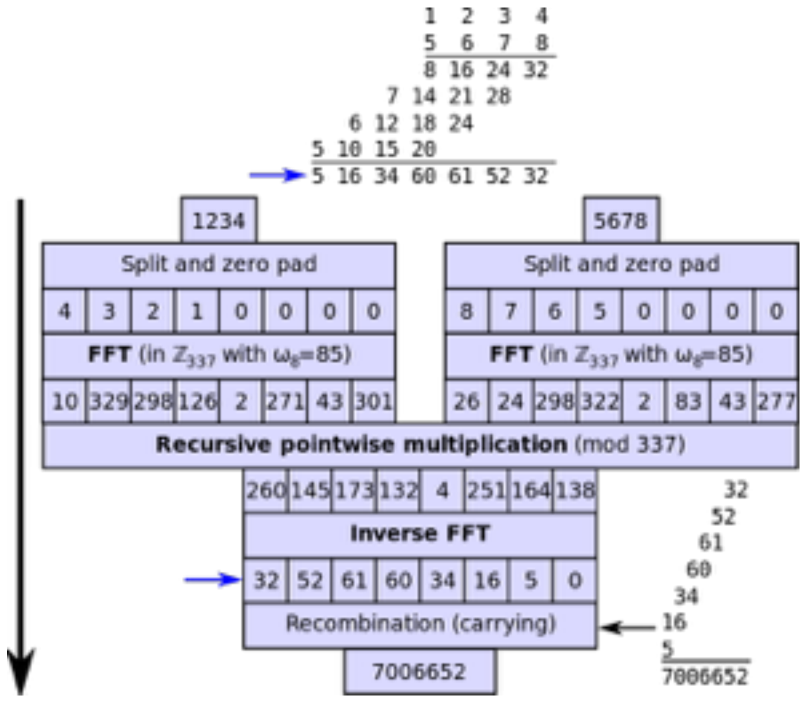
application to mult



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

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

Multiplying n-bit numbers



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

Schönhage–Strassen ‘71	$O(n \log n \log \log n)$
Fürer ‘07	$O(n \log(n) 2^{\log^*(n)})$

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

PIERRICK GAUDRY, ALEXANDER KRUPPA, AND PAUL ZIMMERMANN

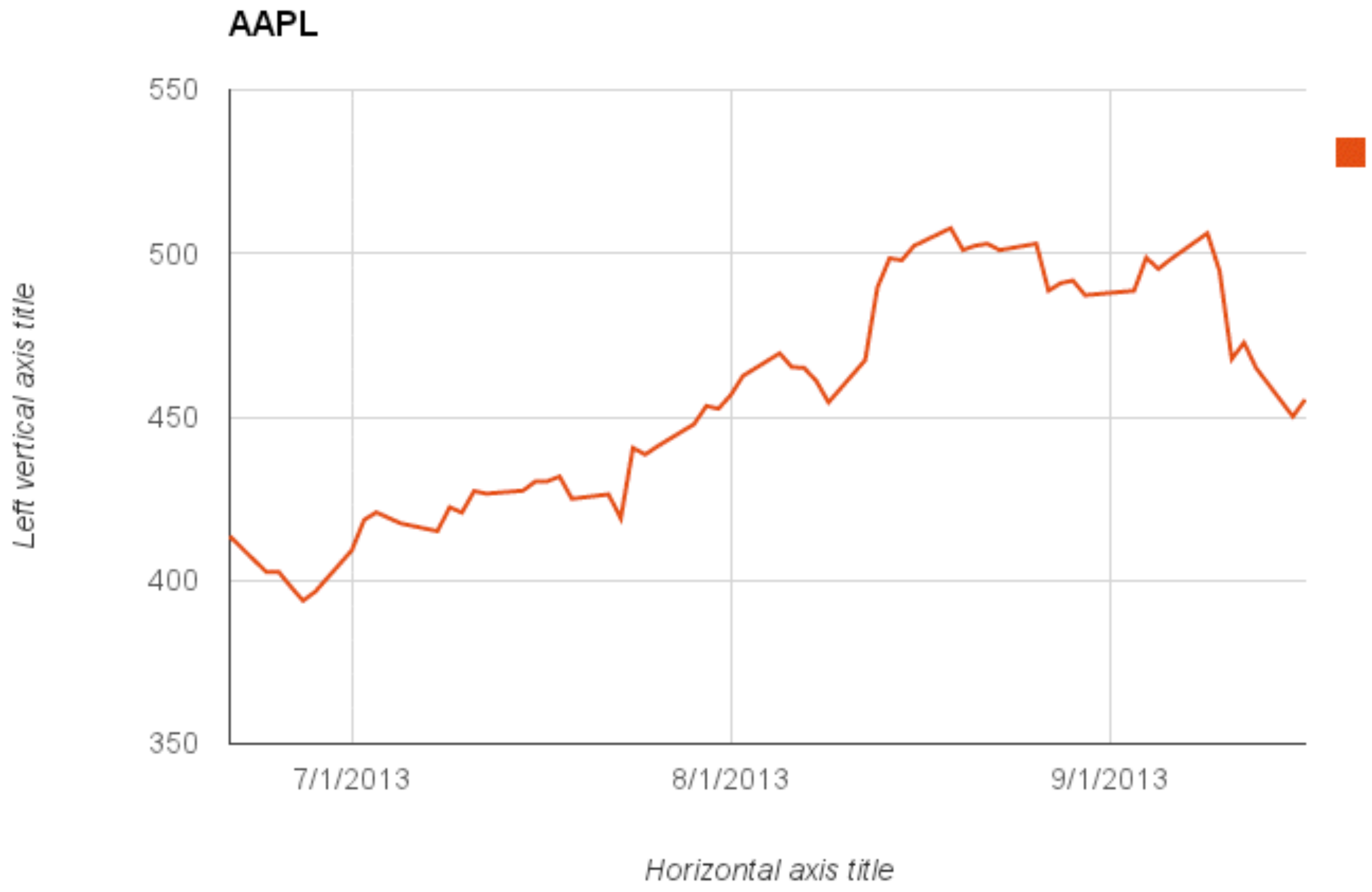
Abstract. Schönhage-Strassen's algorithm is one of the best known algorithms for multiplying large integers. Implementing it efficiently is of utmost importance, since many other algorithms rely on it as a subroutine. We present here an improved implementation, based on the one distributed within the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

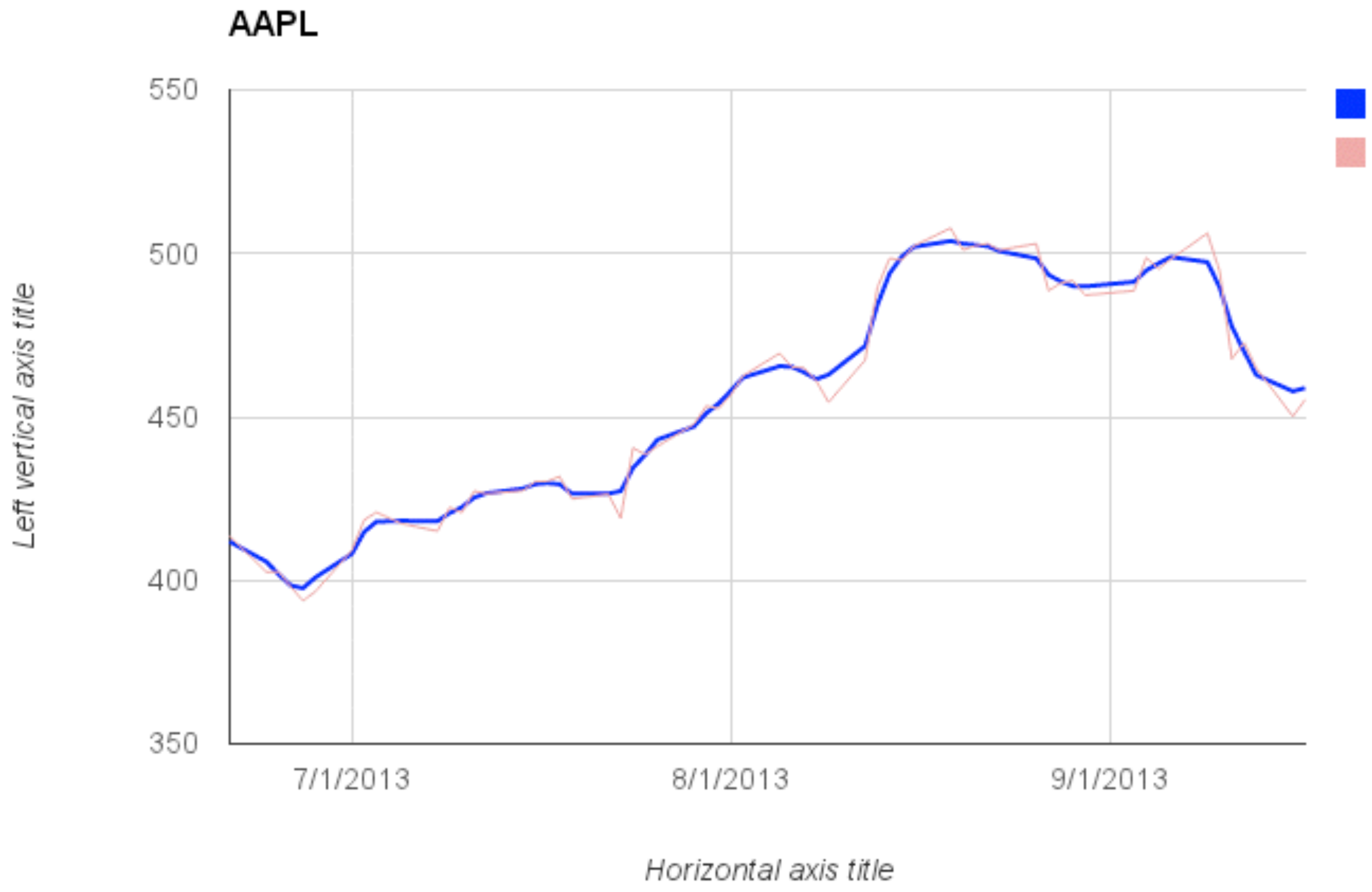
Since Schönhage and Strassen have shown in 1971 how to multiply two N-bit integers in $O(N \log N \log\log N)$ time [21], several authors showed how to reduce other operations — inverse, division, square root, gcd, base conversion, elementary functions — to multiplication, possibly with $\log N$ multiplicative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

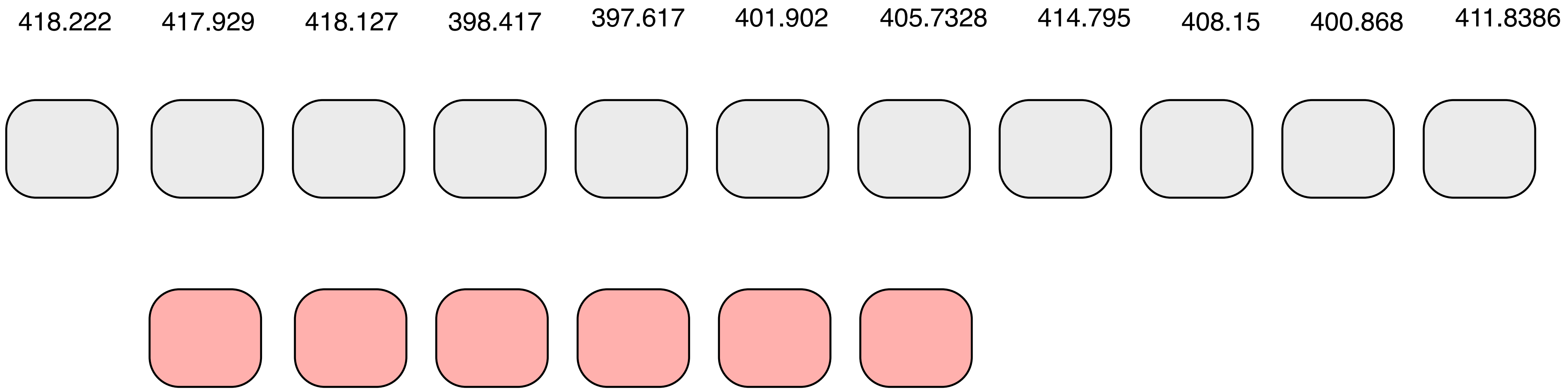
Strangely, much less effort was made for decreasing the implicit constant in $M(N)$ itself, although any gain on that constant will give a similar gain on all multiplication-based operations. Some authors reported on implementations of large integer arithmetic for specific hardware or as part of a number-theoretic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

String matching with *

ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCC
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC
CTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGG
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCCGCGCCGGGACAGAATGCC
CTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAG
TTTAATTACAGACCTGAA

Looking for all occurrences of

GGC*GAG*C*GC

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