# SELECTION

# SELECTION

Given n values, find the k-th smallest

in an array or linked list

rank = k

# SELECTION

Given n values, find the k-th smallest

(n values → in an array or linked list)

(k-th smallest → rank = k)

- easy if we sort first → $O(n\log n)$

# SELECTION

Given n values, find the k-th smallest

in an array or linked list (Given n values)

rank = k (k-th smallest)

- easy if we sort first → $O(n\log n)$
- easy and fast if $k = O(1)$ or $k = n - O(1)$ → $O(n)$

# SELECTION

Given $n$ values, find the k-th smallest

(Given $n$ values → in an array or linked list)

(k-th smallest → rank = $k$)

- easy if we sort first → $O(n \log n)$
- easy and fast if $k = O(1)$ or $k = n - O(1)$ → $O(n)$

harder as $k \to \frac{n}{2}$ (median)

# SELECTION - Given n values, find the k-th smallest

Simple algorithm:

# SELECTION - Given $n$ values, find the $k$-th smallest

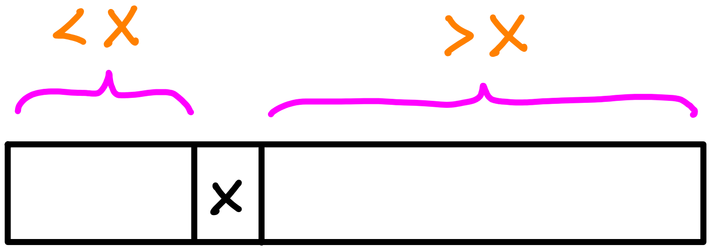
Simple algorithm:

- pick a random input value: $x$

# SELECTION - Given n values, find the k-th smallest

## Simple algorithm:

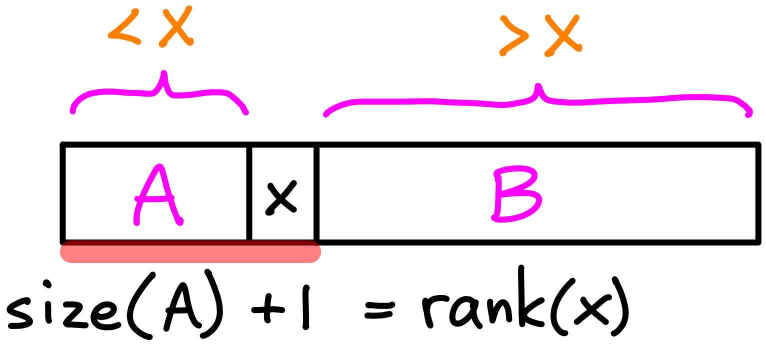
- pick a random input value: x
- use x as pivot, partition input →



# SELECTION - Given $n$ values, find the $k$-th smallest

## Simple algorithm:

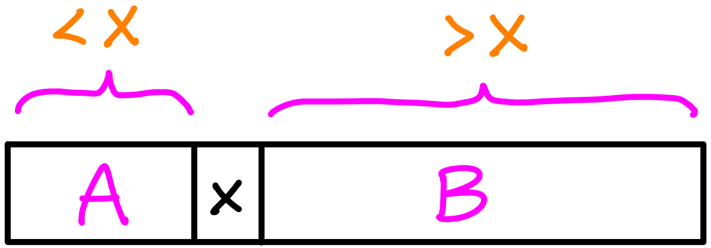
- pick a random input value: $x$
- use $x$ as pivot, partition input →



# SELECTION - Given n values, find the k-th smallest

## Simple algorithm:

- pick a random input value: x
- use x as pivot, partition input →



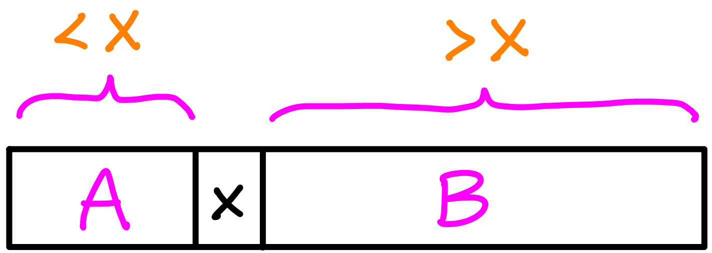

size(A) + 1 = rank(x)

- if rank(x) = k, return x

# SELECTION – Given $n$ values, find the $k$-th smallest

## Simple algorithm:

- pick a random input value: $x$
- use $x$ as pivot, partition input →



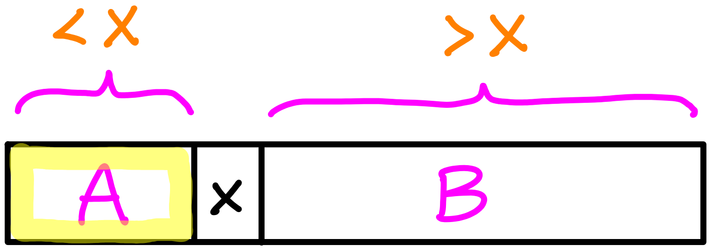

$$\text{size}(A) + 1 = \text{rank}(x)$$

- if $\text{rank}(x) = k$, return $x$
- if $k < \text{rank}(x)$ ...?

# SELECTION - Given n values, find the k-th smallest

## Simple algorithm:

- pick a random input value: x
- use x as pivot, partition input →
- ┌ if rank(x) = k, return x
  └ if k < rank(x), recurse on A

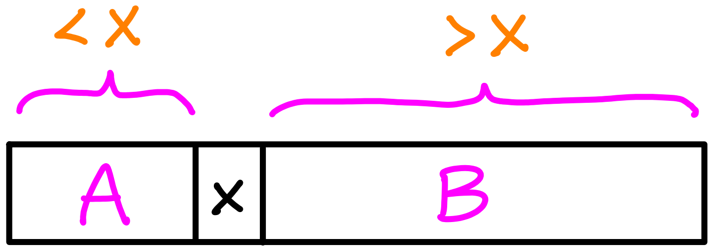



size(A) + 1 = rank(x)

# SELECTION – Given $n$ values, find the $k$-th smallest

## Simple algorithm:

- pick a random input value: $x$
- use $x$ as pivot, partition input →



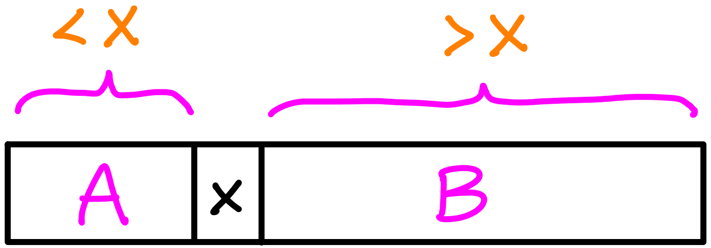

$\text{size}(A) + 1 = \text{rank}(x)$

- if $\text{rank}(x) = k$, return $x$
  - if $k < \text{rank}(x)$, recurse on A → find value with rank $k$

# SELECTION – Given n values, find the k-th smallest

## Simple algorithm:

- pick a random input value: $x$
- use $x$ as pivot, partition input →



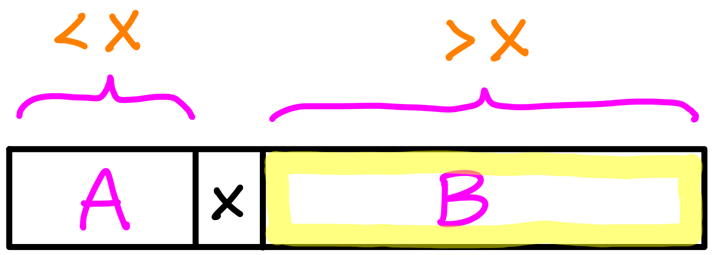

$size(A) + 1 = rank(x)$

- if $rank(x) = k$, return $x$
  - if $k < rank(x)$, recurse on A → find value with rank k
  - if $rank(x) < k$ ... ?

# SELECTION - Given $n$ values, find the $k$-th smallest

Simple algorithm:

- pick a random input value: $x$
- use $x$ as pivot, partition input →



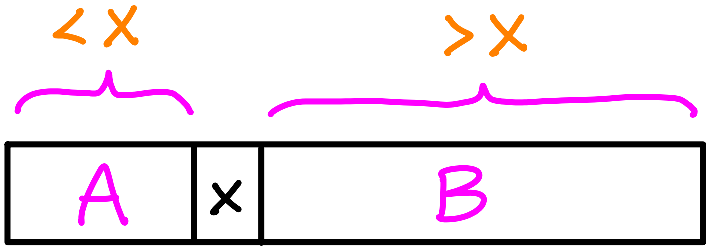

$\text{size}(A) + 1 = \text{rank}(x)$

- if $\text{rank}(x) = k$, return $x$
  - if $k < \text{rank}(x)$, recurse on A → find value with rank $k$
  - if $\text{rank}(x) < k$, recurse on B

# SELECTION – Given $n$ values, find the $k$-th smallest

## Simple algorithm:

- pick a random input value: $x$
- use $x$ as pivot, partition input →



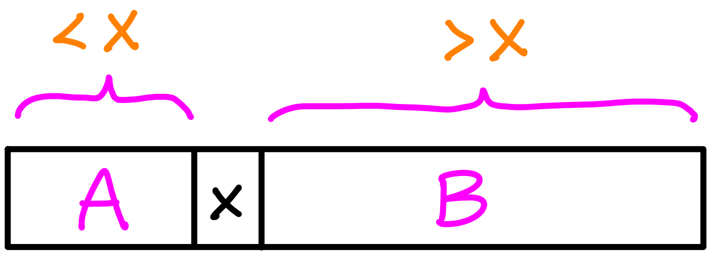

$$\text{size}(A) + 1 = \text{rank}(x)$$

- if $\text{rank}(x) = k$, return $x$
  - if $k < \text{rank}(x)$, recurse on $A$ → find value with rank $k$
  - if $\text{rank}(x) < k$, recurse on $B$ → find value with rank ...?

# SELECTION – Given n values, find the k-th smallest

## Simple algorithm:

- pick a random input value: $x$
- use $x$ as pivot, partition input →



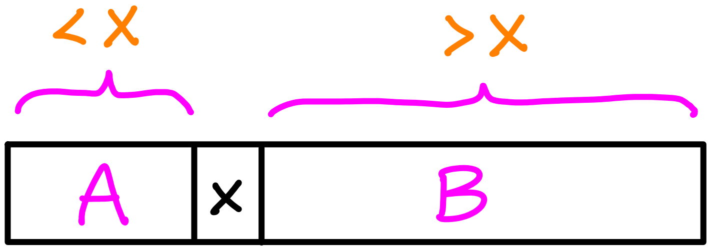

$\text{size}(A) + 1 = \text{rank}(x) = r$

- if $\text{rank}(x) = k$, return $x$
  - if $k < \text{rank}(x)$, recurse on A → find value with rank $k$
  - if $\text{rank}(x) < k$, recurse on B → find value with rank $k - r$

# SELECTION - Given $n$ values, find the $k$-th smallest

## Simple algorithm:

- pick a random input value: $x$
- use $x$ as pivot, partition input →




$\text{size}(A) + 1 = \text{rank}(x) = r$

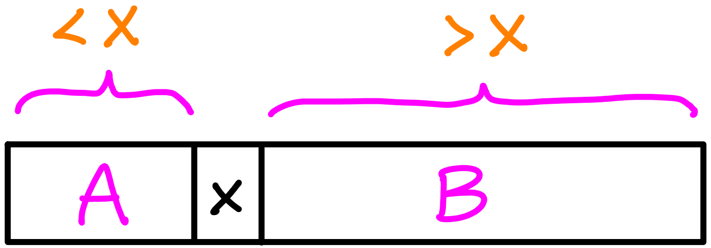
- if $\text{rank}(x) = k$, return $x$
  - if $k < \text{rank}(x)$, recurse on $A$ → find value with rank $k$
  - if $\text{rank}(x) < k$, recurse on $B$ → find value with rank $k - r$

Worst case : $T(n) = \Theta(\,?\,) + T(\,?\,)$

# SELECTION – Given n values, find the k-th smallest

## Simple algorithm:

- pick a random input value: $x$
- use $x$ as pivot, partition input → $\Theta(n)$




$\text{size}(A) + 1 = \text{rank}(x) = r$

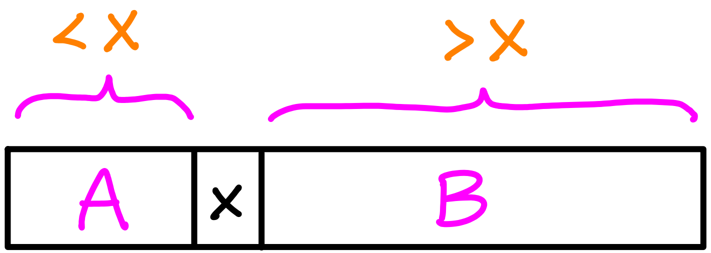
- if $\text{rank}(x) = k$, return $x$
  - if $k < \text{rank}(x)$, recurse on $A$ → find value with rank $k$
  - if $\text{rank}(x) < k$, recurse on $B$ → find value with rank $k - r$

Worst case : $T(n) = \Theta(n) + T(n-1) = ?$

# SELECTION – Given $n$ values, find the $k$-th smallest

**Simple algorithm:**

- pick a random input value: $x$
- use $x$ as pivot, partition input → $\Theta(n)$
- if $\text{rank}(x) = k$, return $x$
  - if $k < \text{rank}(x)$, recurse on A → find value with rank $k$
  - if $\text{rank}(x) < k$, recurse on B → find value with rank $k - r$



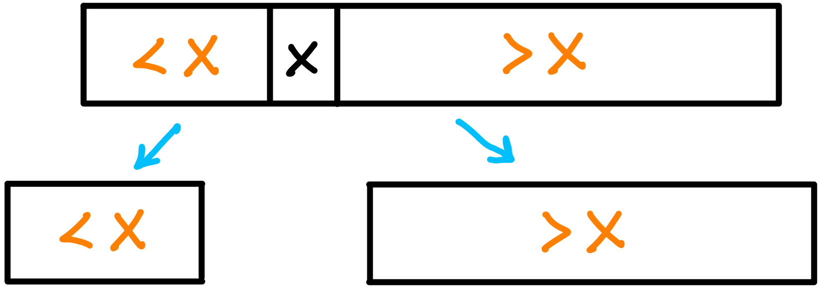

$\text{size}(A) + 1 = \text{rank}(x) = r$

**Worst case:** $T(n) = \Theta(n) + T(n-1) = O(n^2)$

# SELECTION

## summary

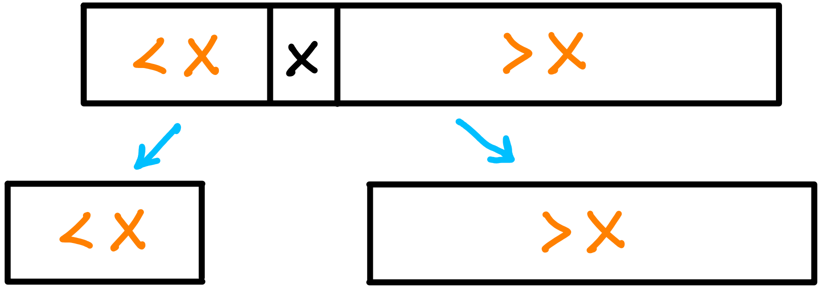
- pick a random input value: x
- use x as pivot, partition input →
- recurse left or right if necessary



# Deterministic SELECTION algorithm (1973)

↳ Blum, Floyd, Pratt, Rivest, Tarjan

- pick a ~~random~~ input value: x
- use x as pivot, partition input →
- recurse left or right if necessary

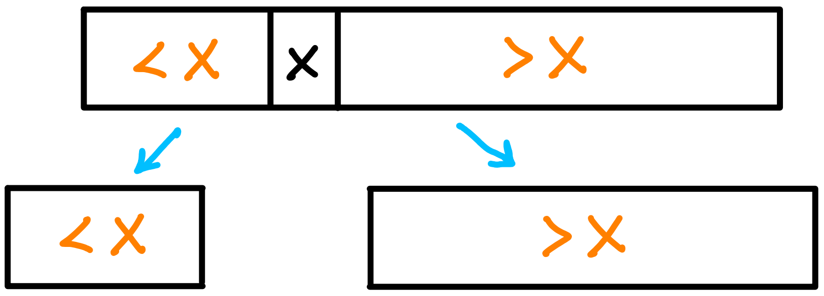


# Deterministic SELECTION algorithm (1973)

↳ Blum, Floyd, Pratt, Rivest, Tarjan

- do something more clever: use some other x
- get O(n) time

- pick a ~~random~~ input value: x
- use x as pivot, partition input →
- recurse left or right if necessary

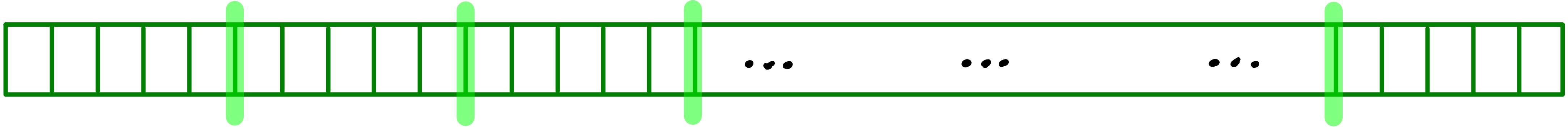


# Deterministic SELECTION algorithm

# Deterministic SELECTION algorithm

For now, assume n = multiple of 5

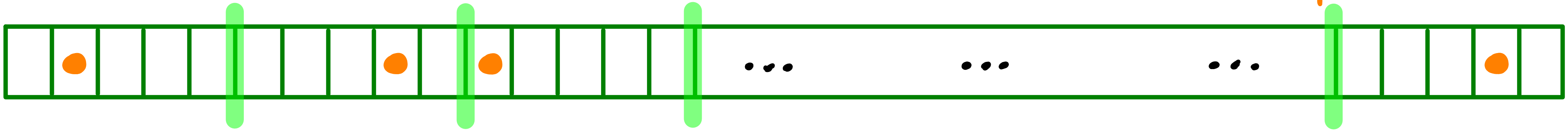
blocks of size 5



# Deterministic SELECTION algorithm

For now, assume n = multiple of 5

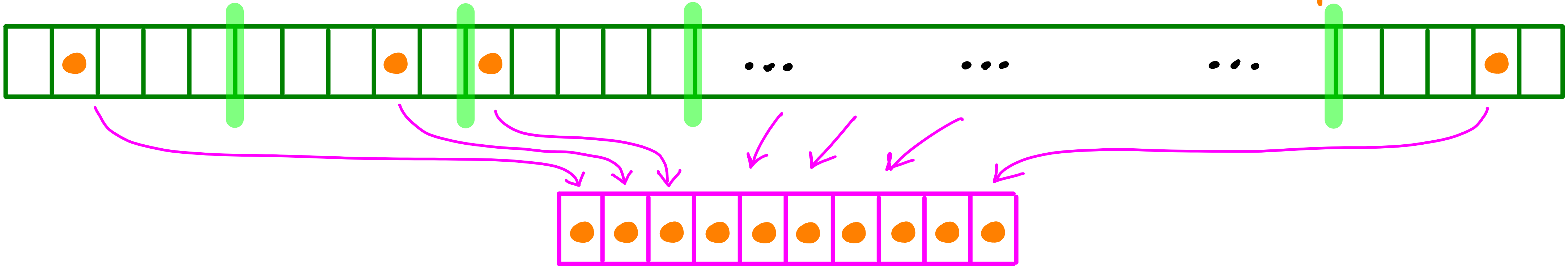
- for each block of size 5: find 3rd smallest value = representative



# Deterministic SELECTION algorithm

For now, assume n = multiple of 5

- for each block of size 5: find 3rd smallest value = representative

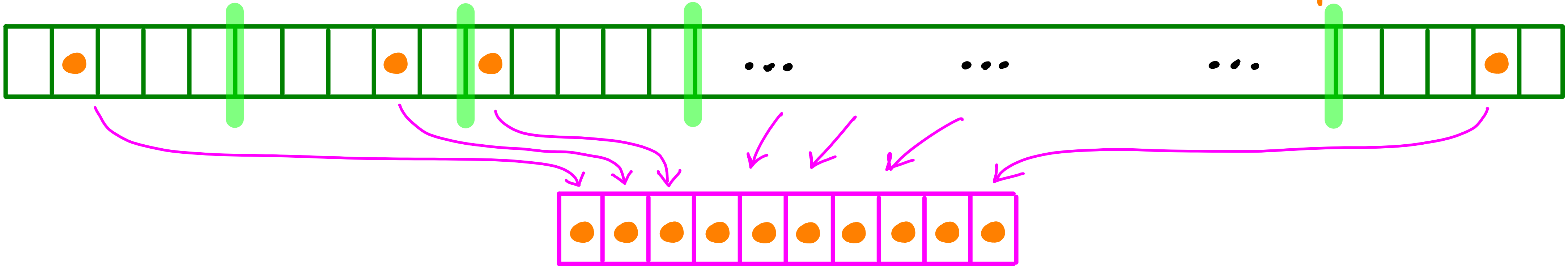


- copy representatives to new list

# Deterministic SELECTION algorithm

For now, assume n = multiple of 5

- for each block of size 5: find 3rd smallest value = representative

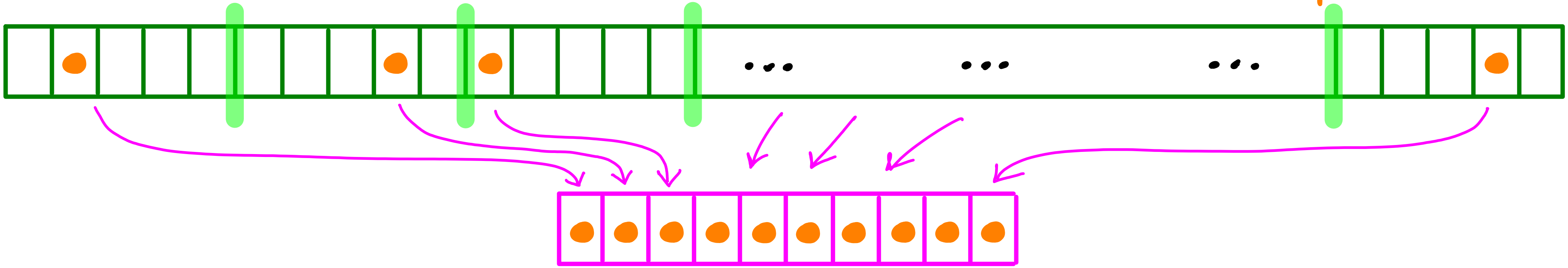



- copy representatives to new list
- find median of representatives → x

# Deterministic SELECTION algorithm

For now, assume n = multiple of 5

- for each block of size 5: find 3rd smallest value = representative



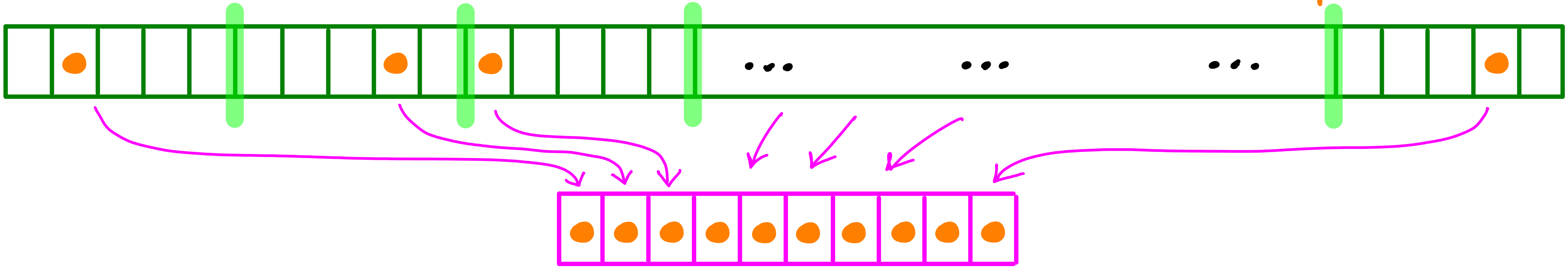

- copy representatives to new list
- find median of representatives → x

How?

# Deterministic SELECTION algorithm

For now, assume n = multiple of 5

- for each block of size 5: find 3rd smallest value = representative



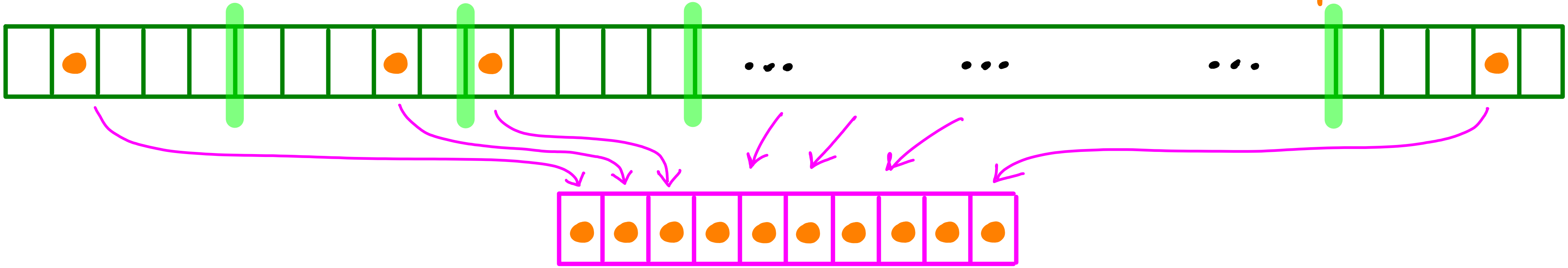
- copy representatives to new list
- find median of representatives → x

How? Use the algorithm (recurse)

# Deterministic SELECTION algorithm

For now, assume n = multiple of 5

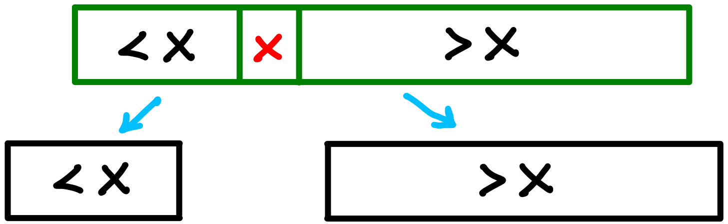
- for each block of size 5: find 3rd smallest value = representative



- copy representatives to new list
- find median of representatives → x

How? Use the algorithm (recurse)

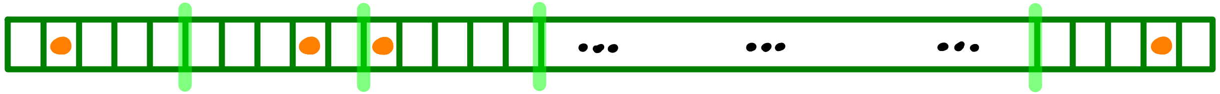
- use x as pivot, partition input




- recurse left or right if necessary

$T(n) = ?$

Time to run Selection on n elements

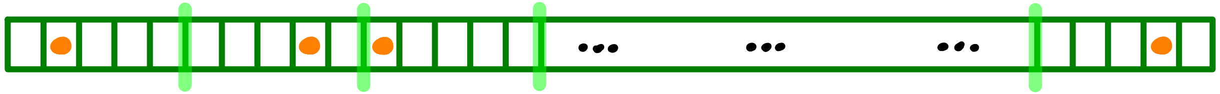



$T(n) = ?$

Time to run Selection on n elements

- for each block of size 5: find representative = ?

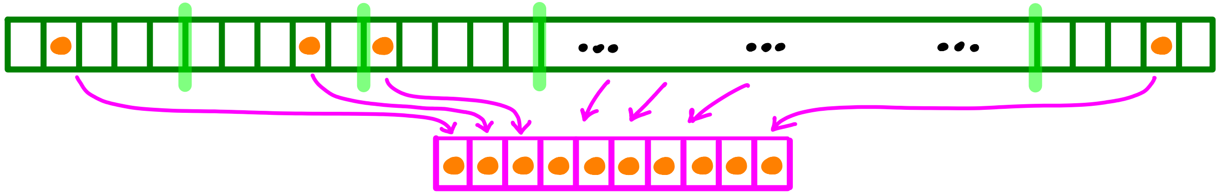
$T(n) = ?$



Time to run Selection on n elements

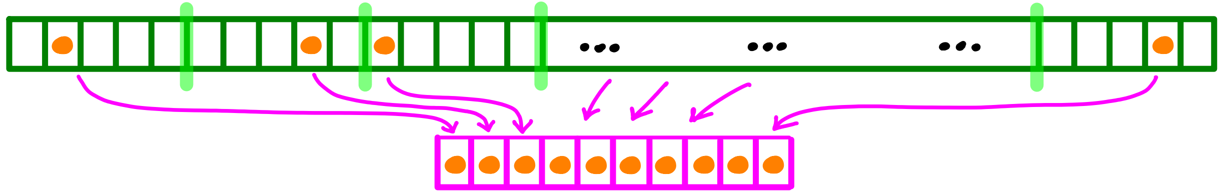
- for each block of size 5: find representative $= \frac{n}{5} \cdot \Theta(1) = \Theta(n)$

$T(n) = ?$




Time to run Selection on n elements

- for each block of size 5: find representative $= \frac{n}{5} \cdot \Theta(1) = \Theta(n)$
- copy representatives to new list $= \Theta(n)$



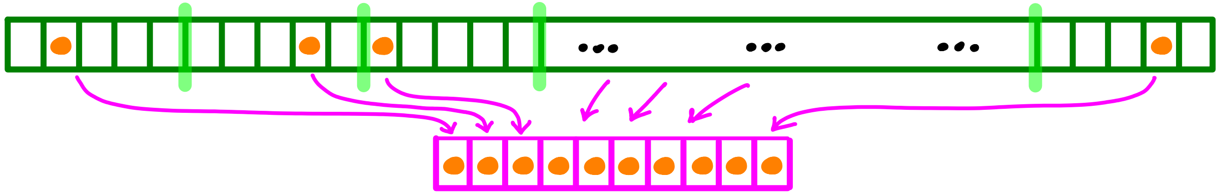
$T(n) = ?$

Time to run Selection on n elements

- for each block of size 5: find representative $= \frac{n}{5} \cdot \Theta(1) = \Theta(n)$
- copy representatives to new list $= \Theta(n)$
- find median of representatives $\rightarrow x$ ... ?

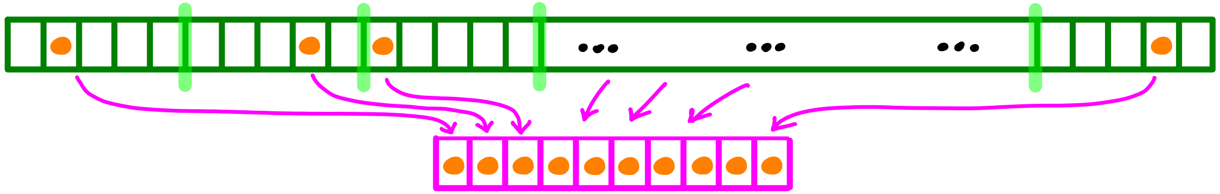
$T(n) = ?$

Time to run Selection on n elements



- for each block of size 5: find representative $= \frac{n}{5} \cdot \Theta(1) = \Theta(n)$
- copy representatives to new list $= \Theta(n)$
- find median of representatives $\rightarrow x$ (recurse)

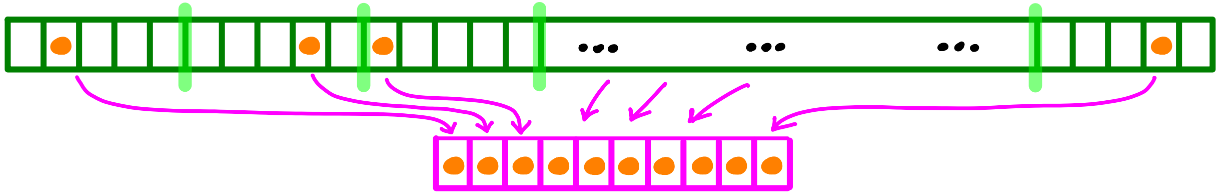
$T(n) = ?$



Time to run Selection on n elements

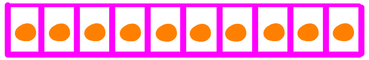
- for each block of size 5: find representative $= \frac{n}{5} \cdot \Theta(1) = \Theta(n)$
- copy representatives to new list $= \Theta(n)$
- find median of representatives $\rightarrow x$ (recurse)

$T(n) = ?$



Time to run Selection on n elements

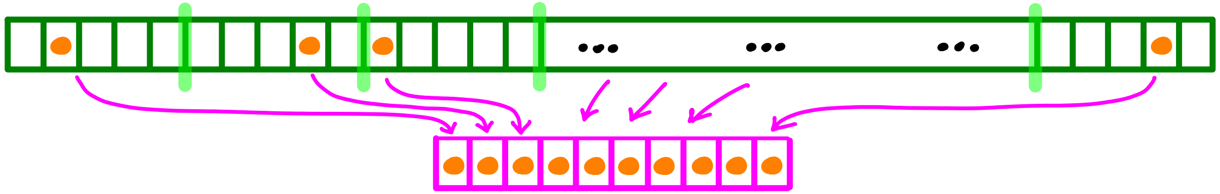
- for each block of size 5: find representative $= \frac{n}{5} \cdot \Theta(1) = \Theta(n)$
- copy representatives to new list $= \Theta(n)$



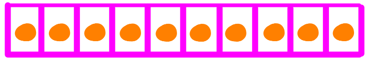
- find median of representatives $\rightarrow x$ (recurse) $= T(\frac{n}{5})$

$T(n) = ?$

Time to run Selection on $n$ elements

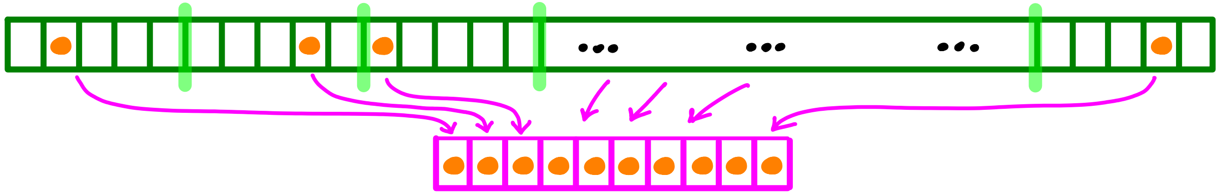


- for each block of size 5: find representative $= \frac{n}{5} \cdot \Theta(1) = \Theta(n)$
- copy representatives to new list $= \Theta(n)$



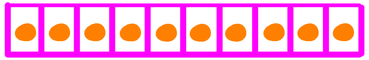
- find median of representatives $\rightarrow x$ (recurse) $= T(\frac{n}{5})$
- use $x$ as pivot, partition input $= \Theta(n)$

$T(n) = ?$

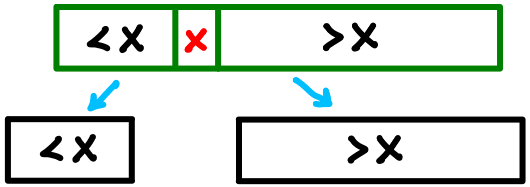


Time to run Selection on $n$ elements

- for each block of size 5: find representative $= \frac{n}{5} \cdot \Theta(1) = \Theta(n)$
- copy representatives to new list $= \Theta(n)$



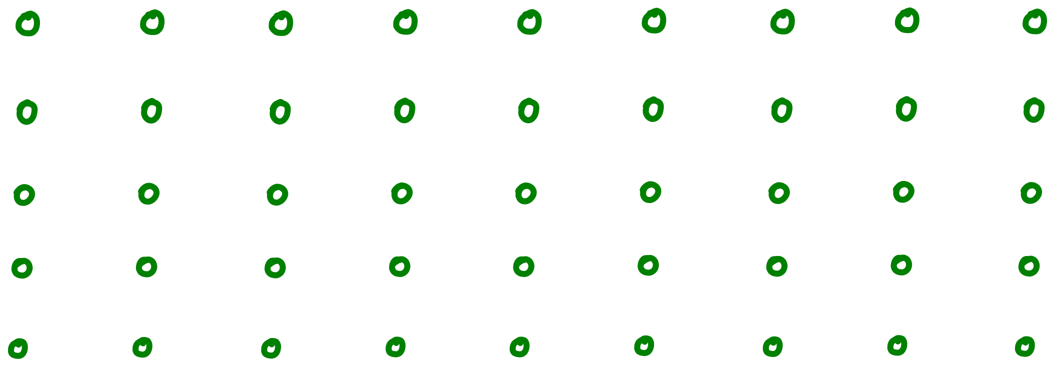
- find median of representatives $\rightarrow x$ (recurse) $= T\left(\frac{n}{5}\right)$
- use $x$ as pivot, partition input $= \Theta(n)$




- recurse left or right if necessary $= T(?)$

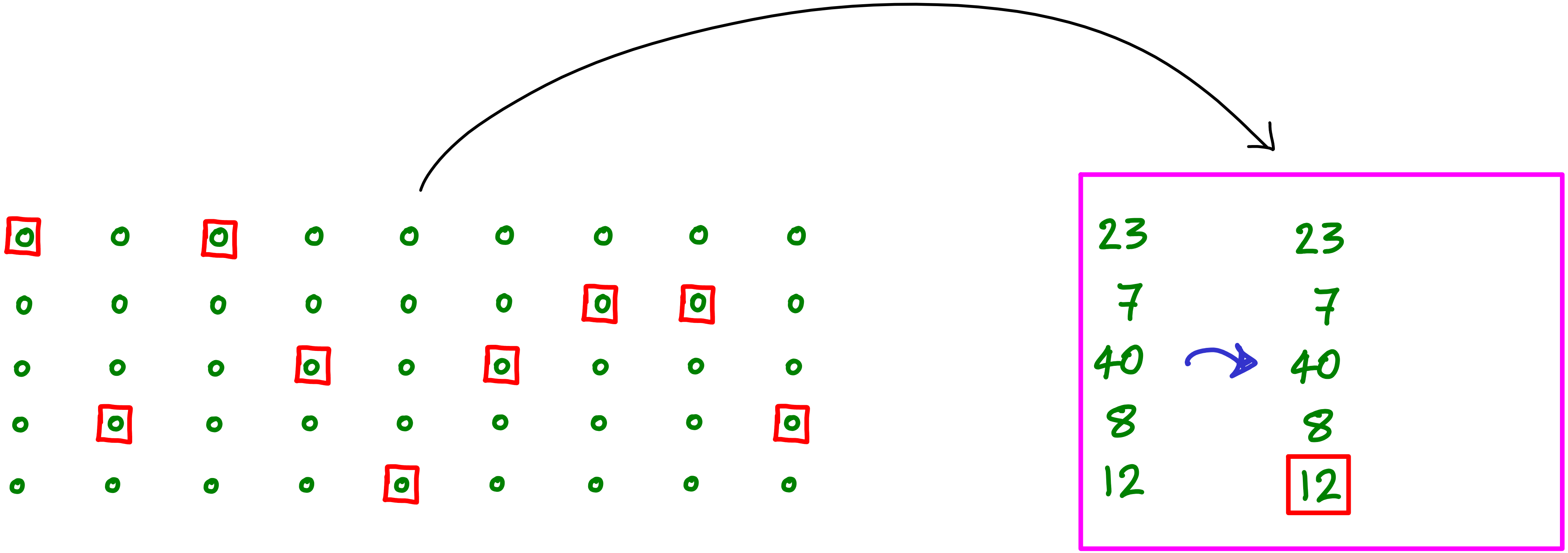
# VISUALIZATION - not part of algorithm

- blocks of size 5



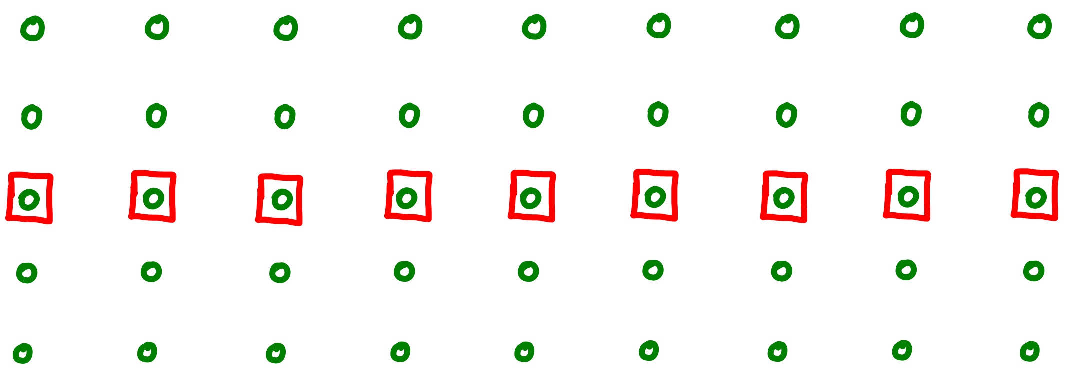
# VISUALIZATION – not part of algorithm

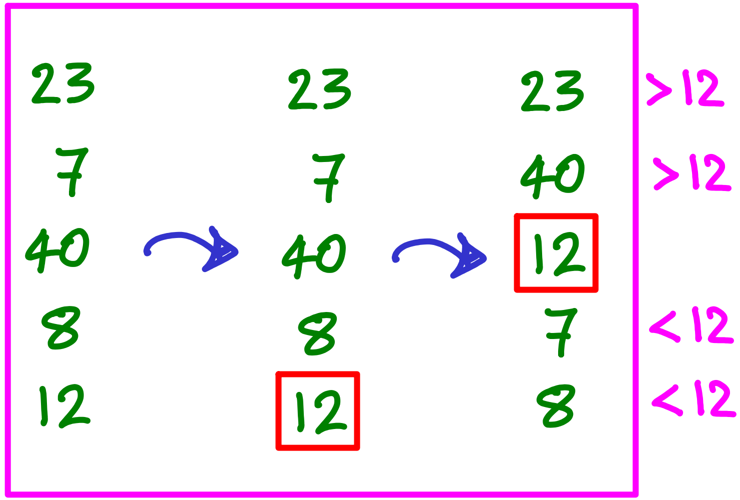
- blocks of size 5
- find representatives



# VISUALIZATION - not part of algorithm

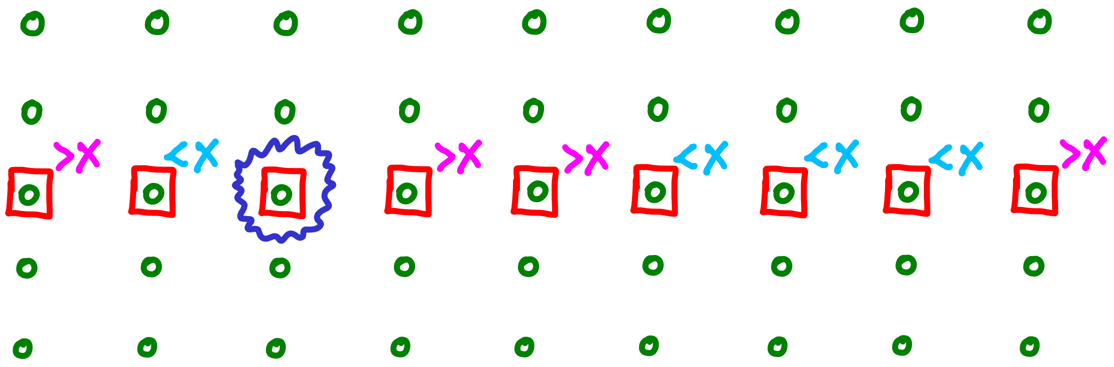
- blocks of size 5
- find representatives





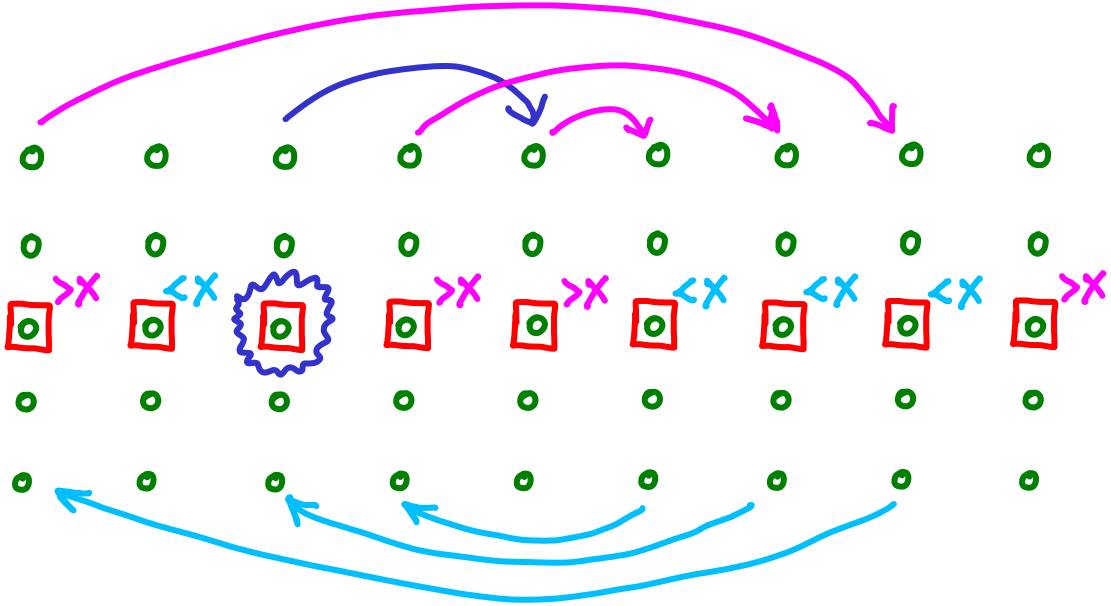
# VISUALIZATION - not part of algorithm

- blocks of size 5
- find representatives
- find median of representatives → x



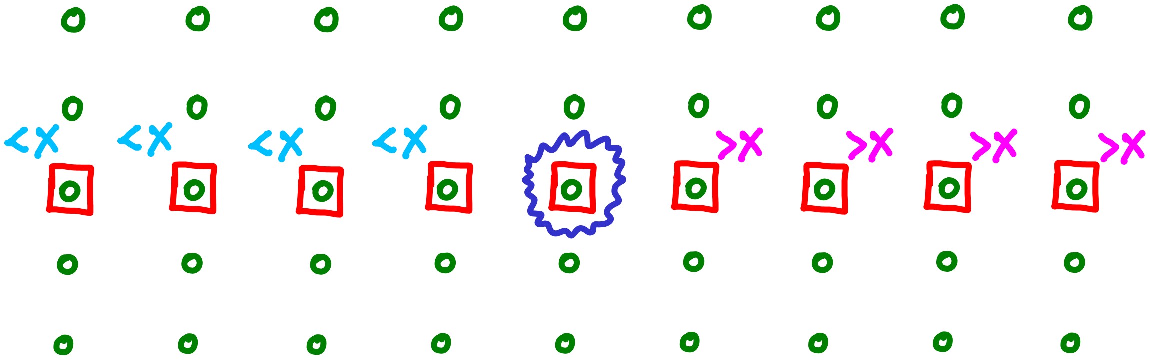
# VISUALIZATION – not part of algorithm

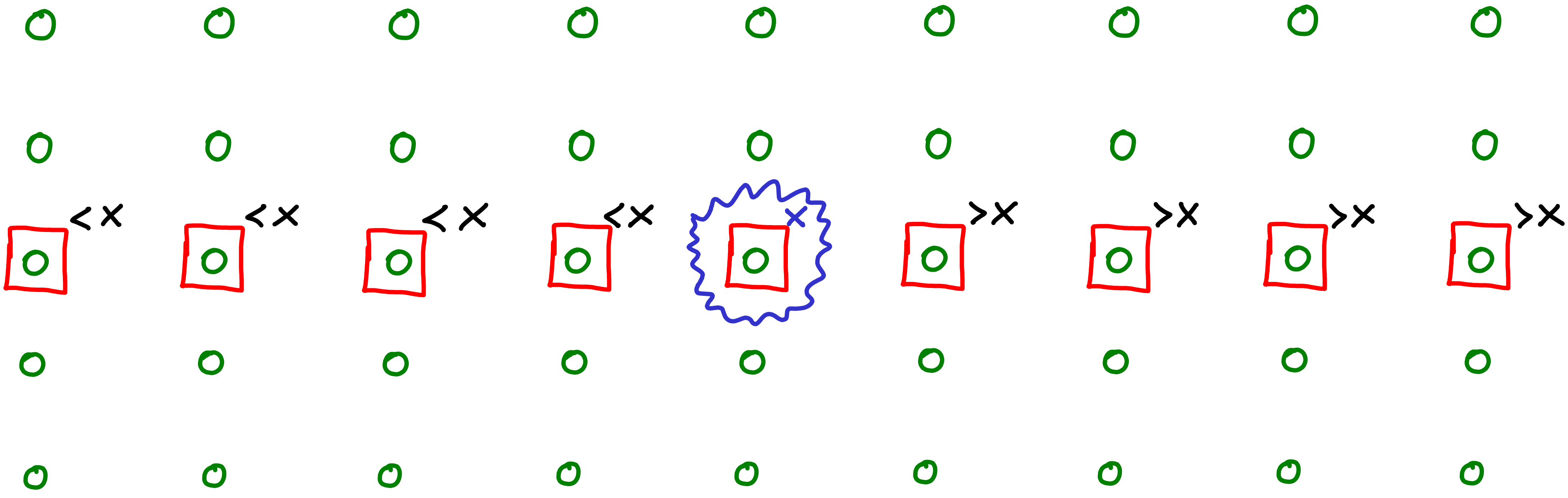
- blocks of size 5
- find representatives
- find median of representatives → x

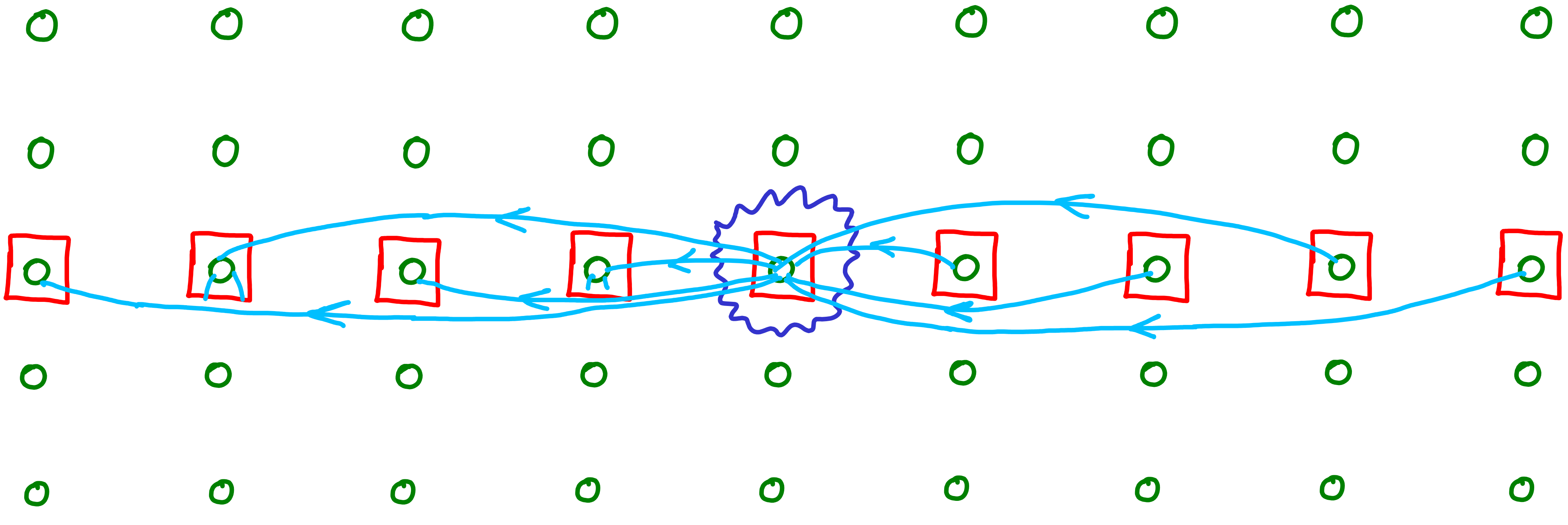


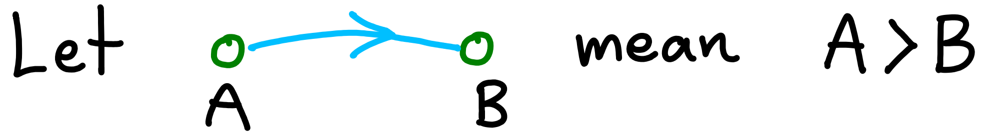
# VISUALIZATION – not part of algorithm

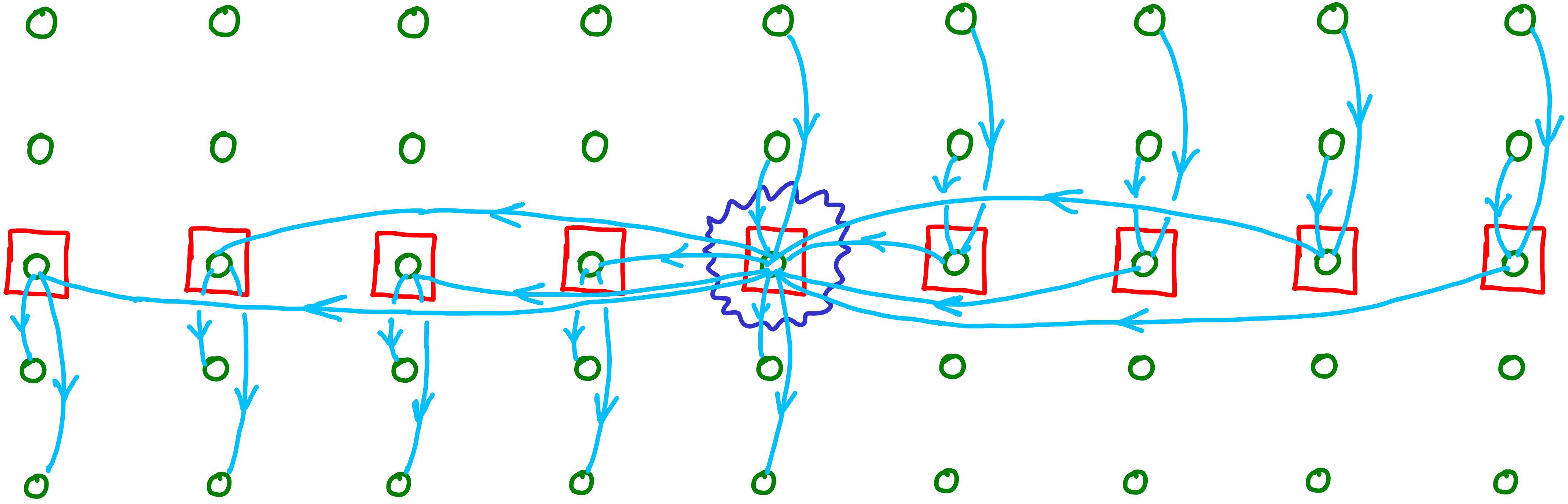
- blocks of size 5
- find representatives
- find median of representatives → x

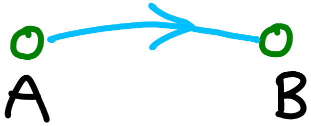



<x
<x
<x
<x
x
>x
>x
>x
>x

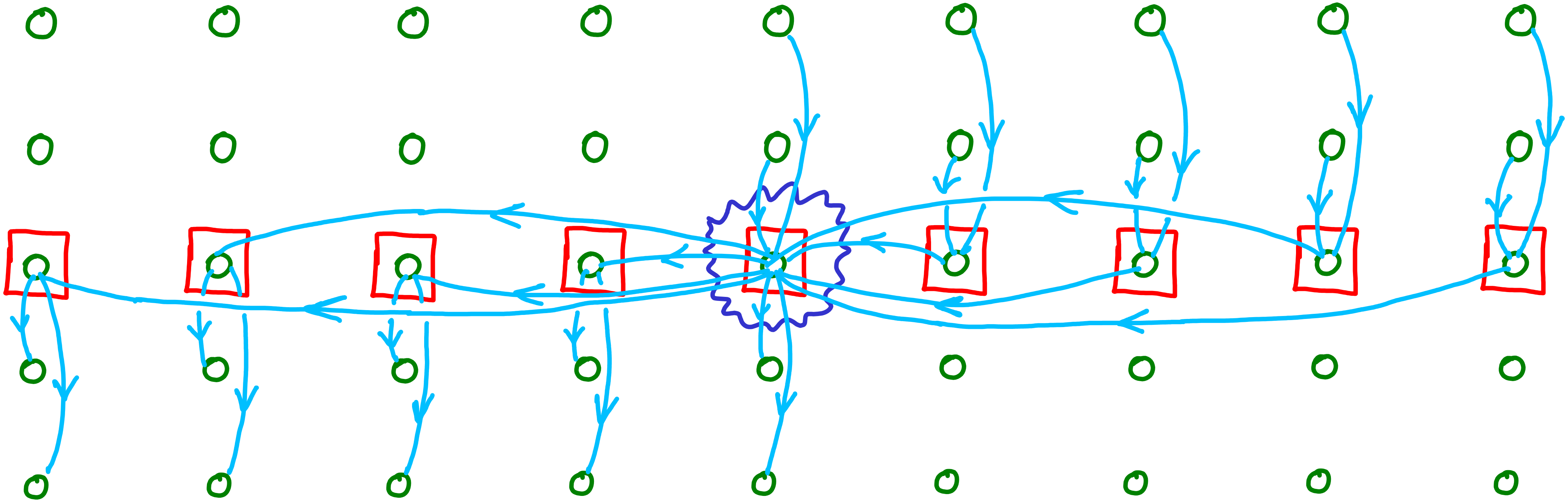


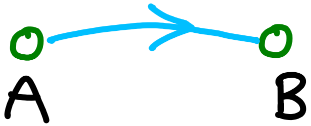

Let
A
B
mean A > B



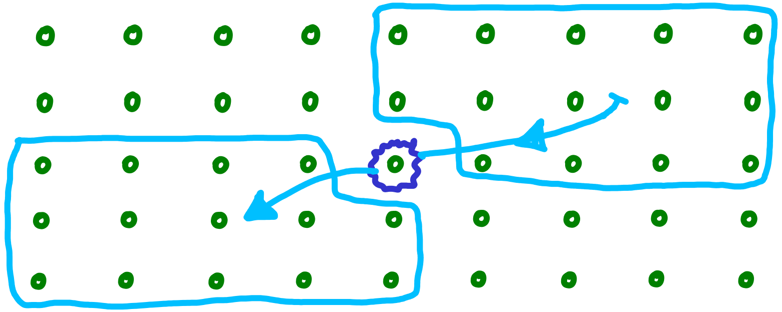
Let 

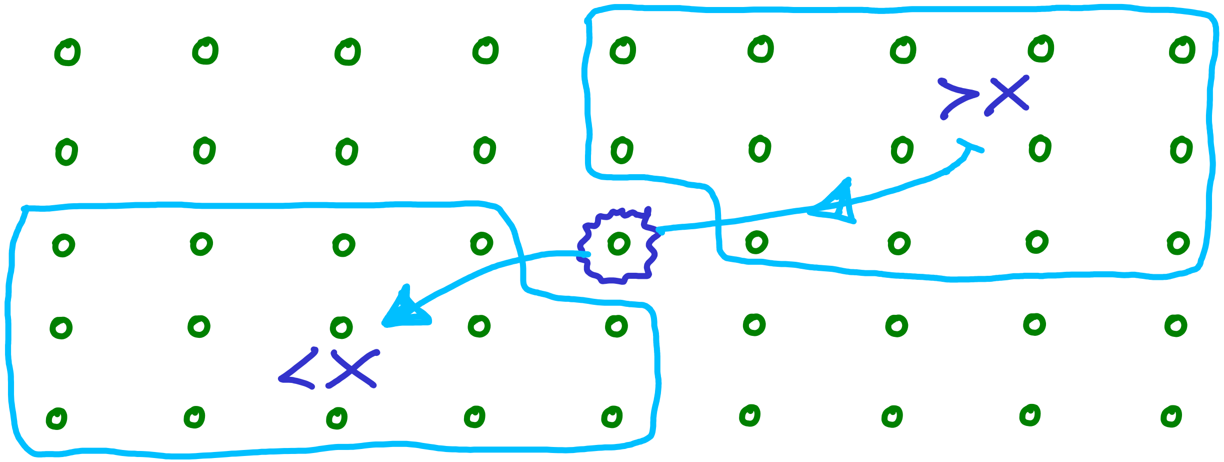
mean $A > B$



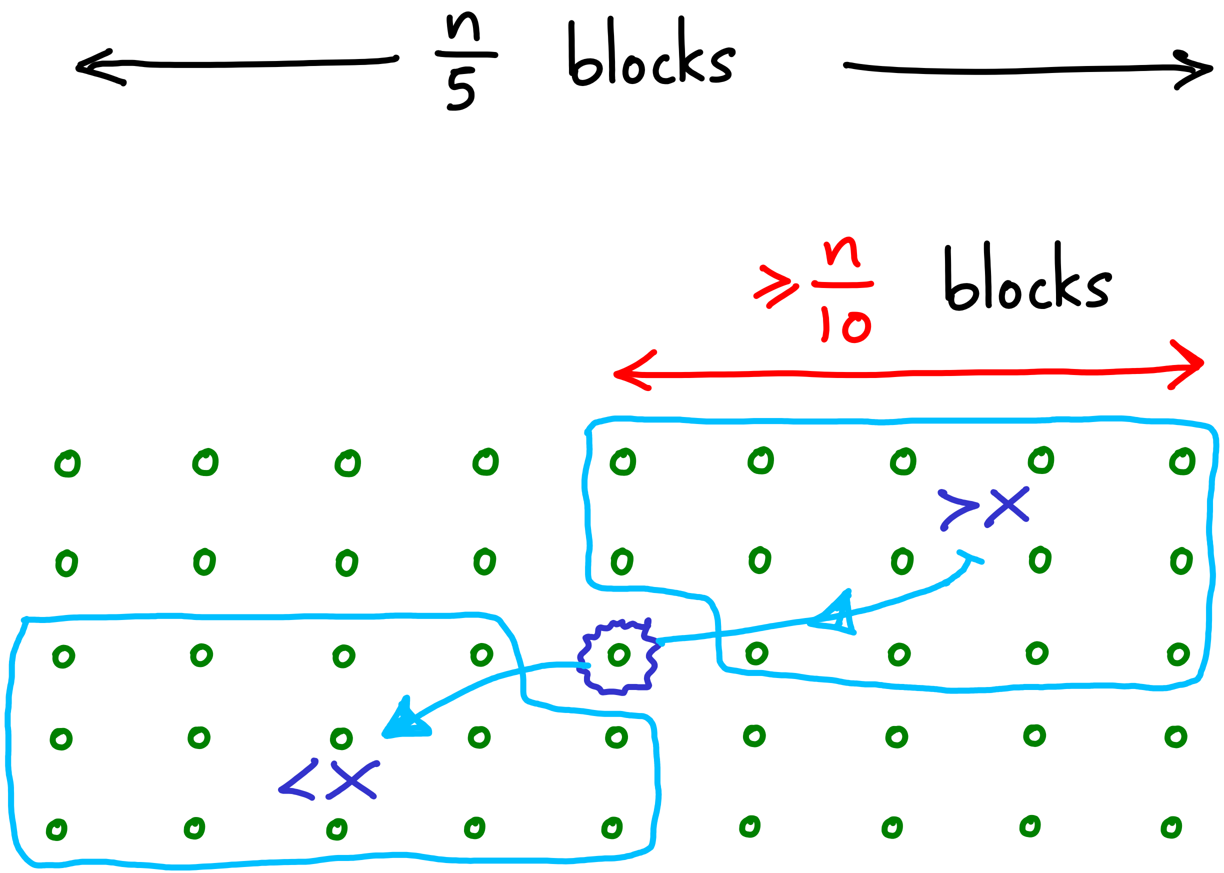
Let 

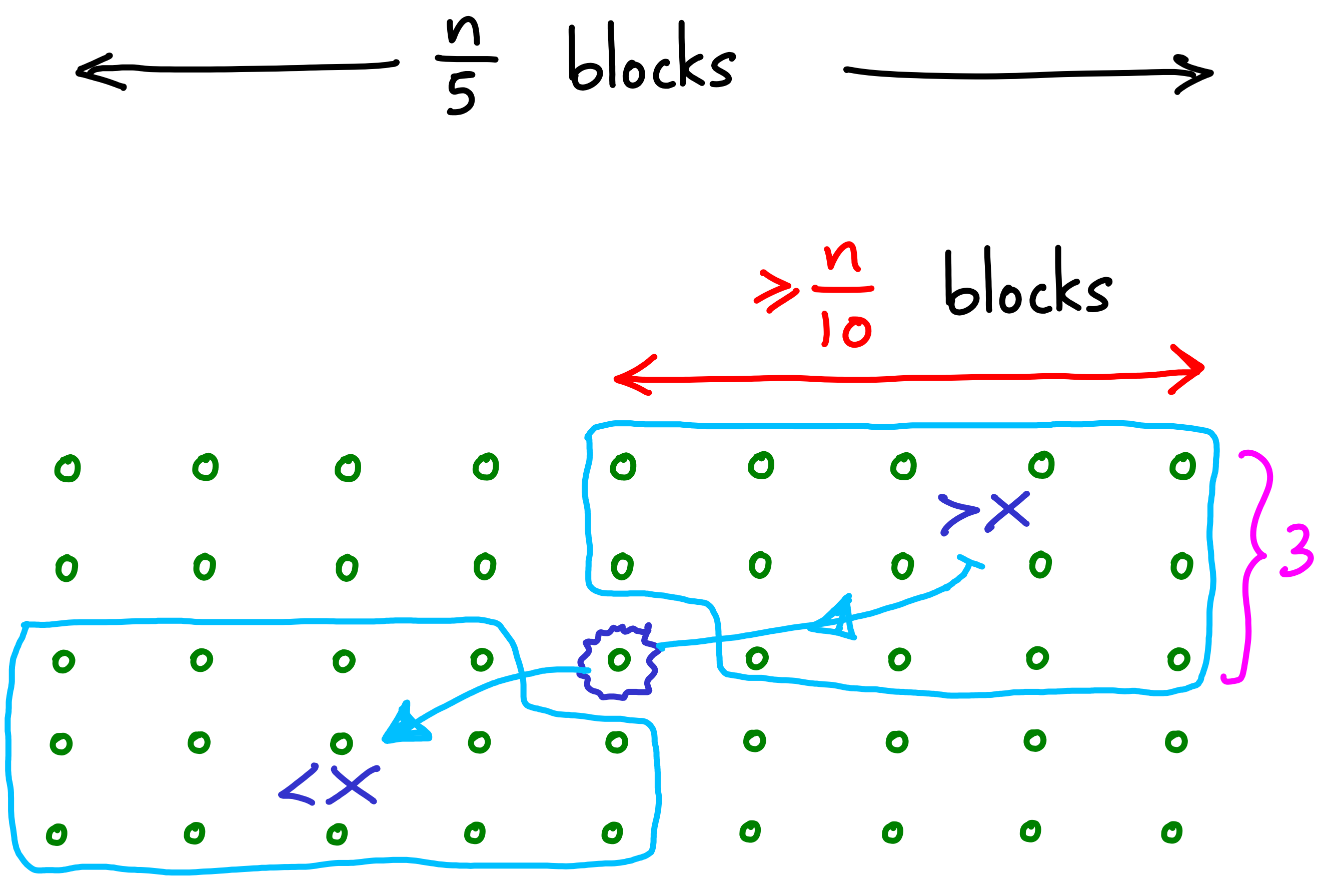
mean $A > B$





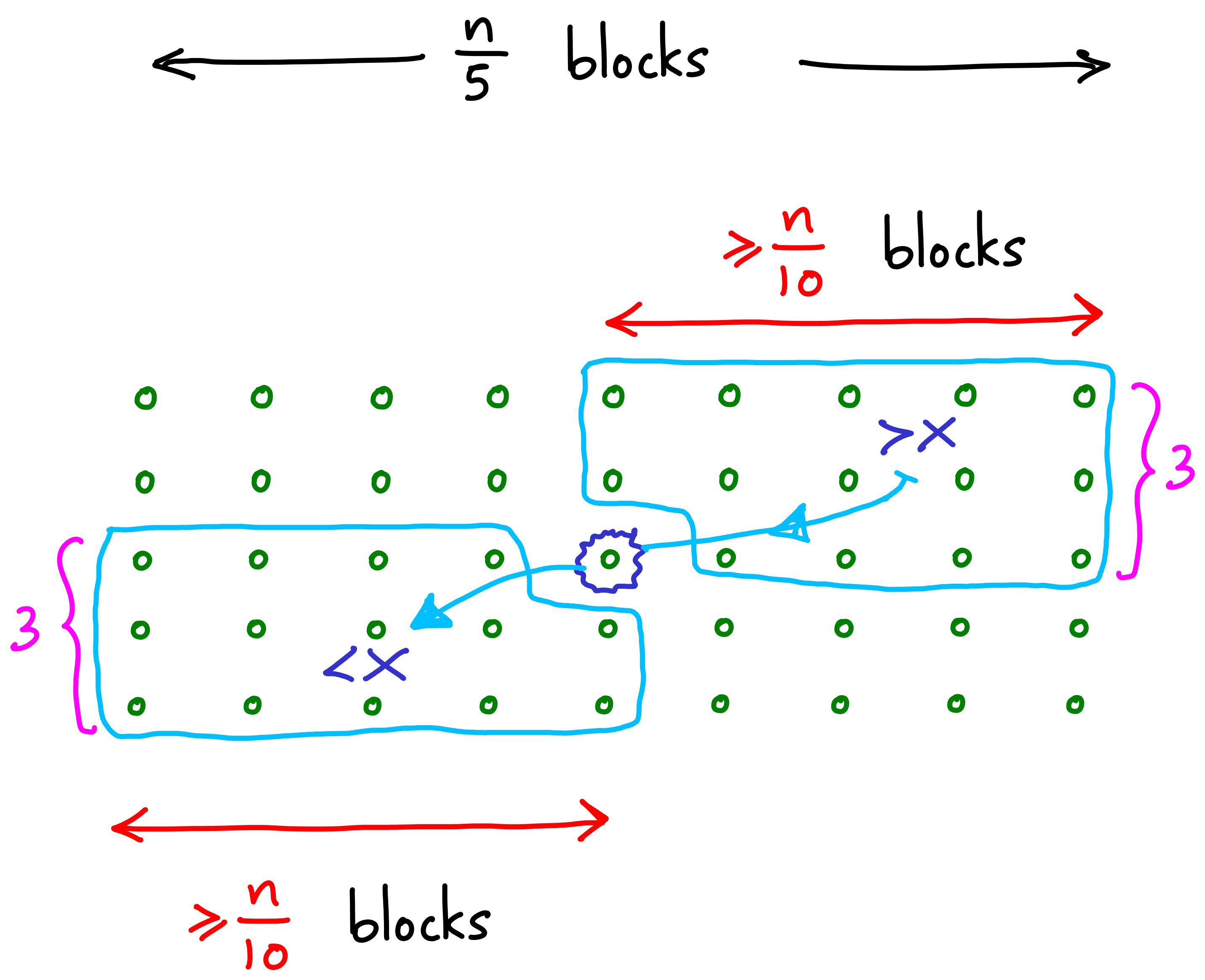
>x
<x


$\frac{n}{5}$ blocks
$\geq \frac{n}{10}$ blocks
$>x$
$<x$




#elements > x

$$\geq 3 \cdot \frac{n}{10}$$

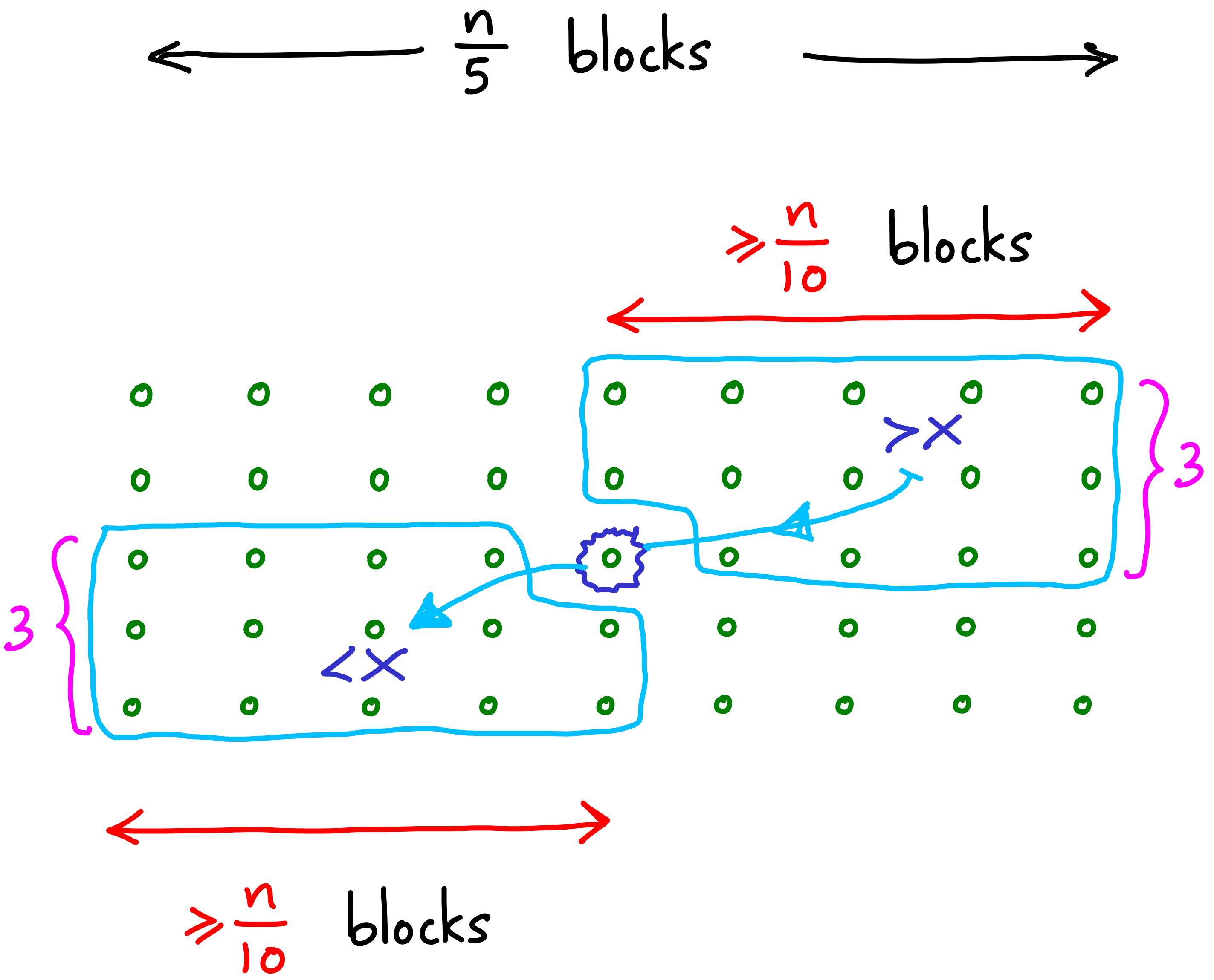



$\#\text{elements} > x$

$\geq 3 \cdot \frac{n}{10}$

$\#\text{elements} < x$

$\geq 3 \cdot \frac{n}{10}$




$\#$elements $> x$

$\geq 3 \cdot \frac{n}{10}$

$\#$elements $< x$

$\geq 3 \cdot \frac{n}{10}$

After partitioning, we recurse on at most $\frac{7n}{10}$ elements

- After partitioning, we recurse on at most $\frac{7n}{10}$ elements
- If $n$ is not a multiple of 5, we might need to recurse on a few more elements.

- After partitioning, we recurse on at most $\frac{7n}{10}$ elements
- If $n$ is not a multiple of 5,

  we might need to recurse on a few more elements.

  (or spend extra $O(n)$ work dealing with this)

- After partitioning, we recurse on at most $\frac{7n}{10}$ elements
- If $n$ is not a multiple of 5,

  we might need to recurse on a few more elements.

  (or spend extra $O(n)$ work dealing with this)

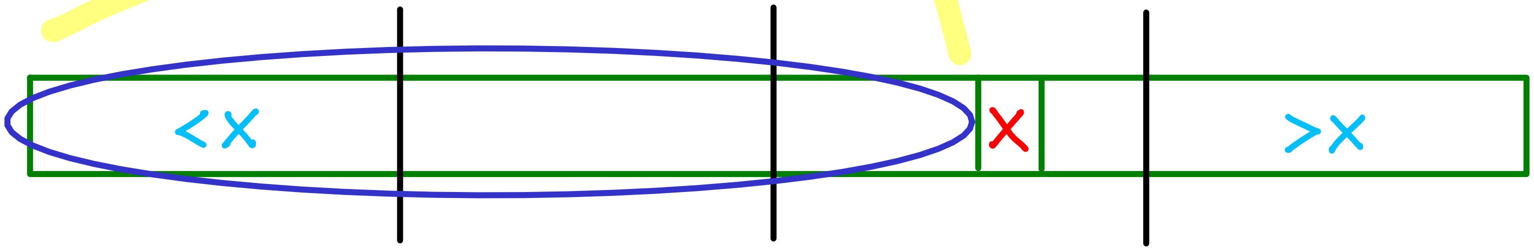
- For large enough $n$ ($n > 50$), we recurse on at most $\frac{3n}{4}$ elements.

- After partitioning, we recurse on at most $\frac{7n}{10}$ elements
- If $n$ is not a multiple of 5,

  we might need to recurse on a few more elements.

  (or spend extra $O(n)$ work dealing with this)

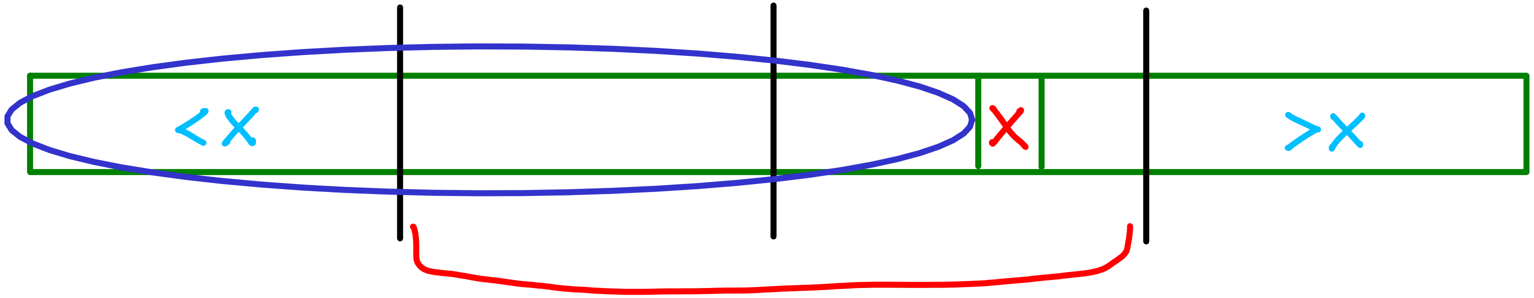
- For large enough $n$ ($n > 50$), we recurse on at most $\frac{3n}{4}$ elements.



- After partitioning, we recurse on at most $\frac{7n}{10}$ elements
- If $n$ is not a multiple of 5,
  we might need to recurse on a few more elements.
  (or spend extra $O(n)$ work dealing with this)

- For large enough $n$ ($n > 50$), we recurse on at most $\frac{3n}{4}$ elements.

  ↳ $x$ has rank between $\frac{n}{4}$ and $\frac{3n}{4}$



$$T(n) \leq \Theta(n) + T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right)$$

1) find representatives
2) copy to new list

3) find $x$

4) partition using $x$

5) recurse left or right

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + dn$$

assume $T(k) \leq ck$ for all $k < n$

$$T(n) \le T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + dn$$

assume $T(k) \le ck$ for all $k < n$

$$T(n) \le c \cdot \frac{n}{5} + c \cdot \frac{3n}{4} + dn$$

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + dn$$

assume $T(k) \leq ck$ for all $k < n$

$$T(n) \leq c \cdot \frac{n}{5} + c \cdot \frac{3n}{4} + dn$$

$$= \frac{19}{20} cn + dn$$

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + dn$$

assume $T(k) \leq ck$ for all $k < n$

$$T(n) \leq c \cdot \frac{n}{5} + c \cdot \frac{3n}{4} + dn$$

$$= \frac{19}{20} cn + dn$$

$$= cn - \left(\frac{cn}{20} - dn\right)$$

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + dn$$

assume $T(k) \leq ck$ for all $k < n$

$$T(n) \leq c \cdot \frac{n}{5} + c \cdot \frac{3n}{4} + dn$$

$$= \frac{19}{20} cn + dn$$

$$= cn - \left(\frac{cn}{20} - dn\right)$$

$$\leq cn$$ ...if $c = 20d$