Selection & Closest Pair

Selection

Input: n comparable elem. $A[1 \sim n]$, $k \in [n]$

Goal: kth smallest element of $A[1..n]$

e.g. median $(k = \lceil n/2 \rceil)$ of

10, 56, 77, 91, 70, 24, 36, 27, 64, 50, 65

Obv. approach: sort A, look up kth smallest

$O(n \log n)$

Better?

1 Recursive specification

Designing a recursive algorithm

1. Recursive specification
   - declares input/output of algo
   - "contract" algo must fulfill
   - induction hypothesis for correctness
2. Implement spec
3. Prove correctness

   argue algo satisfies recursive spec
   for all n by induction on n

① Recursive specification

`select`$(k, A[1..n])$: Given an array $A[1..n]$ of comparable elements and an integer $k \in \{1, \ldots, n\}$, returns the kth smallest element out of $A[1..n]$.

① Recursive specification

`select`$(k, A[1..n])$: Given an array $A[1..n]$ of comparable elements and an integer $k \in \{1, \ldots, n\}$, returns the kth smallest element out of $A[1..n]$.

② Implement spec

Designing a recursive algorithm

① Recursive specification

- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness

argue algo satisfies recursive spec for all n by induction on n

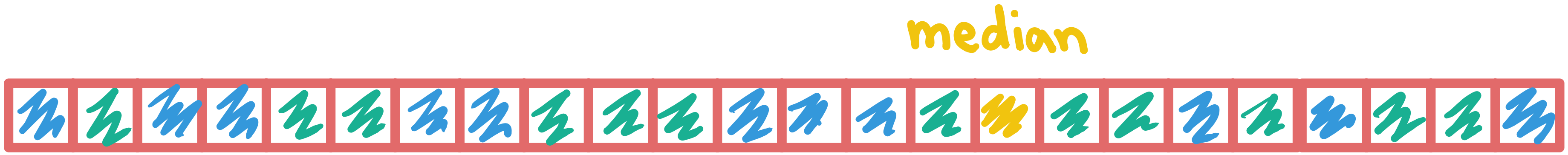
thought experiment: suppose O(n)-time subroutine

$$\texttt{median}(A[1..n]) = \texttt{select}(\lceil n/2 \rceil, A[1..n])$$
$$= \text{the } \lceil n/2 \rceil\text{th smallest element out of } A[1..n].$$

how to use median to find kth element?

"pivot"

① find median



② divide input

③ recurse on appropriate half

Designing a recursive algorithm
① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness
② Implement spec
③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

① Recursive specification

`select`$(k, A[1..n])$: Given an array $A[1..n]$ of comparable elements and an integer $k \in \{1, \ldots, n\}$, returns the kth smallest element out of $A[1..n]$.

② Implement spec

thought experiment: suppose $O(n)$-time subroutine

$$\texttt{median}(A[1..n]) = \texttt{select}(\lceil n/2 \rceil, A[1..n])$$
$$= \text{the } \lceil n/2 \rceil\text{th smallest element out of } A[1..n].$$

<u>`select`$(k, A[1..n])$</u>

/ `select`$(k, A[1..n])$ = the kth smallest element in $A[1..n]$. We assume the elements in A are distinct for simplicity. (Otherwise, break ties arbitrarily.) */*

1. If $n \le 10$ then find the kth element by brute force.
2. Let $p =$ `median`(A). *// We assume a linear time subroutine for `median`$(A[1..n])$.*
3. If $k = \lceil n/2 \rceil$ then return p.
4. If $k < \lceil n/2 \rceil$ then return `select`(k, B), where $B[1..\lceil n/2 \rceil - 1]$ is an array of the $\lceil n/2 \rceil - 1$ elements smaller than p.
5. If $k > \lceil n/2 \rceil$ then return `select`$(k - \lceil n/2 \rceil, B)$, where $B[1..n - \lceil n/2 \rceil]$ is an array of the $n - \lceil n/2 \rceil$ elements larger than p.

Running time

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

n — n

$n/2$ — $n/2$

$n/4$ — $n/4$

$\vdots$

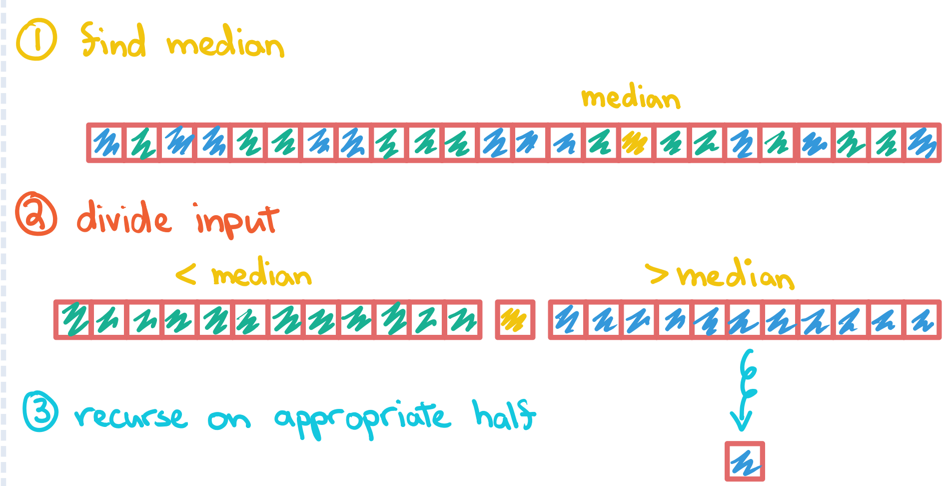
n

$$T(n) = O(n)$$

select$(k, A[1..n])$

/ select$(k, A[1..n])$ = the kth smallest element in $A[1..n]$. We assume the elements in A are distinct for simplicity. (Otherwise, break ties arbitrarily.) */*

1. If $n \leq 10$ then find the kth element by brute force.
2. Let $p = \text{median}(A)$. *// We assume a linear time subroutine for median$(A[1..n])$.*
3. If $k = \lceil n/2 \rceil$ then return p.
4. If $k < \lceil n/2 \rceil$ then return select(k, B), where $B[1..\lceil n/2 \rceil - 1]$ is an array of the $\lceil n/2 \rceil - 1$ elements smaller than p.
5. If $k > \lceil n/2 \rceil$ then return select$(k - \lceil n/2 \rceil, B)$, where $B[1..n - \lceil n/2 \rceil]$ is an array of the $n - \lceil n/2 \rceil$ elements larger than p.



① Recursive specification

`select`$(k, A[1..n])$: Given an array $A[1..n]$ of comparable elements and an integer $k \in \{1, \ldots, n\}$, returns the kth smallest element out of $A[1..n]$.

Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

② Implement spec

thought experiment: suppose O(n)-time subroutine

$$\texttt{median}(A[1..n]) = \texttt{select}(\lceil n/2 \rceil, A[1..n])$$
$$= \text{the } \lceil n/2 \rceil \text{th smallest element out of } A[1..n].$$

Problem: don't have O(n)-time `median`$(A[1..n])$

`select`$(k, A[1..n])$

/* `select`$(k, A[1..n])$ = *the kth smallest element in $A[1..n]$. We assume the elements in A are distinct for simplicity. (Otherwise, break ties arbitrarily.)* */

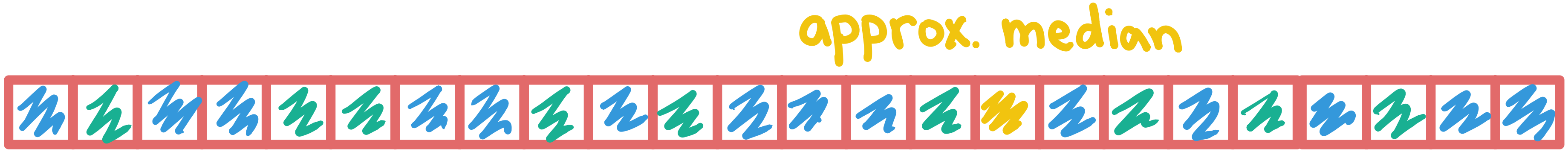
1. If $n \leq 10$ then find the kth element by brute force.
2. Let $p =$ `median`(A). // *We assume a linear time subroutine for* `median`$(A[1..n])$.
3. If $k = \lceil n/2 \rceil$ then return p.
4. If $k < \lceil n/2 \rceil$ then return `select`(k, B), where $B[1..\lceil n/2 \rceil - 1]$ is an array of the $\lceil n/2 \rceil - 1$ elements smaller than p.
5. If $k > \lceil n/2 \rceil$ then return `select`$(k - \lceil n/2 \rceil, B)$, where $B[1..n - \lceil n/2 \rceil]$ is an array of the $n - \lceil n/2 \rceil$ elements larger than p.

O(n) time

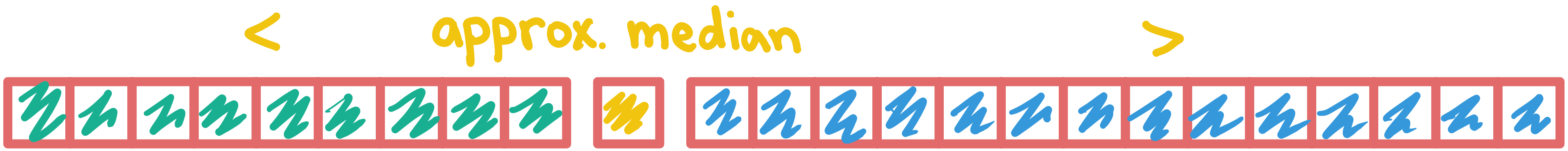
New thought experiment: suppose O(n)-time subroutine

`apx-median`$(A[1..n])$ returns an element $p \in A[1..n]$ with rank between $n/10$ and $9n/10$, in $O(n)$ time.

① find approx. median



② divide input



③ recurse on appropriate half
(somewhat unbalanced)

Designing a recursive algorithm

1. Recursive specification
   - declares input/output of algo
   - "contract" algo must fulfill
   - induction hypothesis for correctness
2. Implement spec
3. Prove correctness

   argue algo satisfies recursive spec for all n by induction on n

① Recursive specification

`select`$(k, A[1..n])$: Given an array $A[1..n]$ of comparable elements and an integer $k \in \{1, \dots, n\}$, returns the kth smallest element out of $A[1..n]$.

② Implement spec

thought experiment: suppose O(n)-time subroutine

`apx-median`$(A[1..n])$ returns an element $p \in A[1..n]$ with rank between $n/10$ and $9n/10$, in $O(n)$ time.

<u>`select`$(k, A[1..n])$</u>

/ `select`$(k, A[1..n])$ = the kth smallest element in $A[1..n]$. We assume the elements in A are distinct for simplicity. (Otherwise, break ties consistently.) */*

1. If $n \leq 10$ then find the kth element by brute force.
2. Let $p =$ `apx-median`(A). *// $O(n)$ time by assumption.*
3. Compare p to each element in $A[1..n]$ to identify its rank ℓ. *// $O(n)$.*
4. If $k = \ell$ then return p.
5. If $k < \ell$ then return `select`$(k, A[1..\ell - 1])$, where $B[1..\ell - 1]$ is an array of the $\ell - 1$ elements smaller than p.
6. If $k > \ell$ then return `select`$(k - \ell, B)$, where $B[1..n - \ell]$ is an array of the $n - \ell$ elements larger than p.

Running time

$$T(n) = T\left(\frac{9n}{10}\right) + O(n)$$

n — n

$\frac{9n}{10}$ — $\frac{9n}{10}$

$\left(\frac{9}{10}\right)^2 n$ $\left(\frac{9}{10}\right)^2 n$

$\left(\frac{9}{10}\right)^3 n$

$+ \vdots$

$$\sum_{i=0}^{\infty} \left(\frac{9}{10}\right)^i n$$

$\log_{10/9}(1/10)$

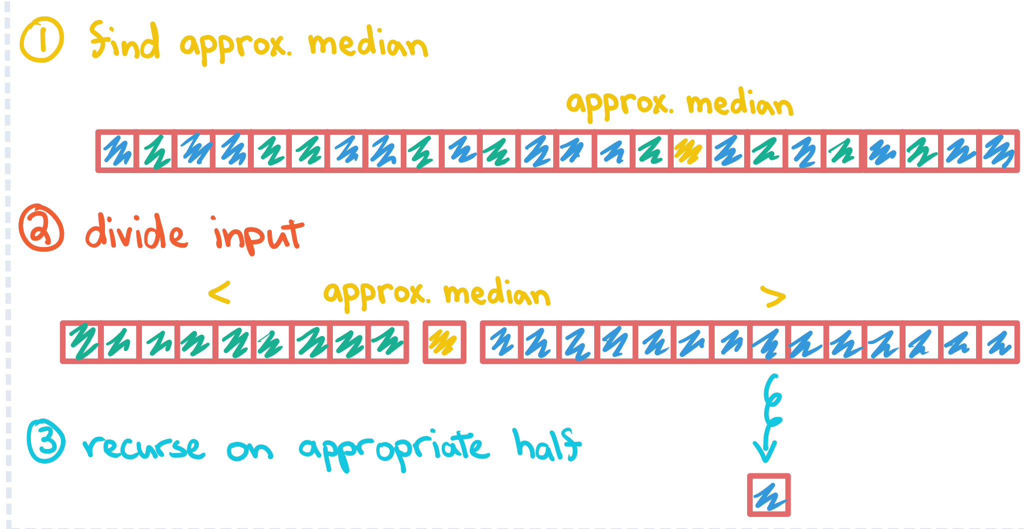
$$\sum_{i=0}^{\infty} (.99999)^i = O(1)$$

$$n\left(\frac{1}{10}\right)^{\log_{10/9} n} = n\left[\left(\frac{10}{9}\right)^{\log_{10/9} n}\right]$$

`select`$(k, A[1..n])$

/* `select`$(k, A[1..n])$ = *the kth smallest element in* $A[1..n]$. *We assume the elements in A are distinct for simplicity. (Otherwise, break ties consistently.)* */

1. If $n \le 10$ then find the kth element by brute force.
2. Let p = `apx-median`(A). // *$O(n)$ time by assumption.*
3. Compare p to each element in $A[1..n]$ to identify its rank ℓ. // *$O(n)$.*
4. If $k = \ell$ then return p.
5. If $k < \ell$ then return `select`$(k, A[1..\ell - 1])$, where $B[1..\ell - 1]$ is an array of the $\ell - 1$ elements smaller than p.
6. If $k > \ell$ then return `select`$(k - \ell, B)$, where $B[1..n - \ell]$ is an array of the $n - \ell$ elements larger than p.



$= n\cdot\left(n^{\log_{10/9}(11/10)}\right)$

$n\sum_{i=0}^{h}\left(\frac{11}{10}\right)^i = O\left(n\left(\frac{11}{10}\right)^h\right)$

$= n\left(\frac{11}{10}\right)^{\log_{10/9} n}$

Remains to implement

$h = \log_{10/9} n$

$\boxed{\texttt{apx-median}(A[1..n])}$ returns an element $p \in A[1..n]$ with rank between $n/10$ and $9n/10$, in $O(n)$ time.

Select$(A[1..n], n/10)$

n

$n/5 \qquad 9n/10$

$\frac{n}{25} \qquad \frac{9n}{50} \qquad \frac{9n}{50} \qquad \frac{81n}{100}$

h

n

$\left(\frac{1}{5}+\frac{9}{10}\right)n$

$\left(\frac{1}{5}+\frac{9}{10}\right)^2 n$

$\vdots$

$\left(\frac{1}{5}+\frac{9}{10}\right)^i n$

$T(n)$ = exact median on n

$S(n)$ = apx-median on n

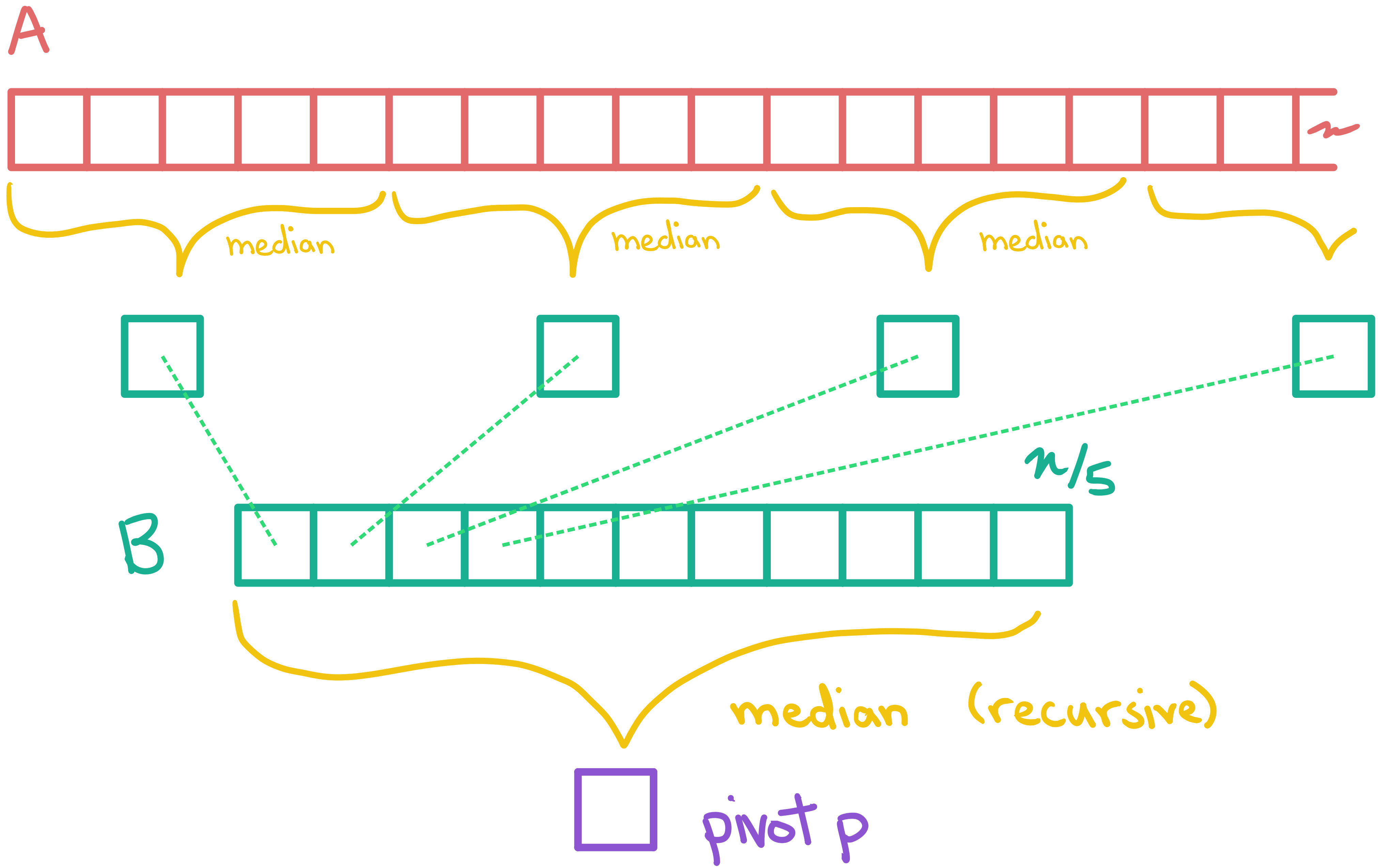
$S(n) = T\left(\frac{n}{5}\right) + O(n)$

$T(n) = \underline{S(n) + O(n)} + T\left(\frac{9n}{10}\right)$

$= T\left(\frac{n}{5}\right) + T\left(\frac{9n}{10}\right) + O(n)$

"Median of Medians"

Blum, Floyd, Pratt, Rivest, Tarjan

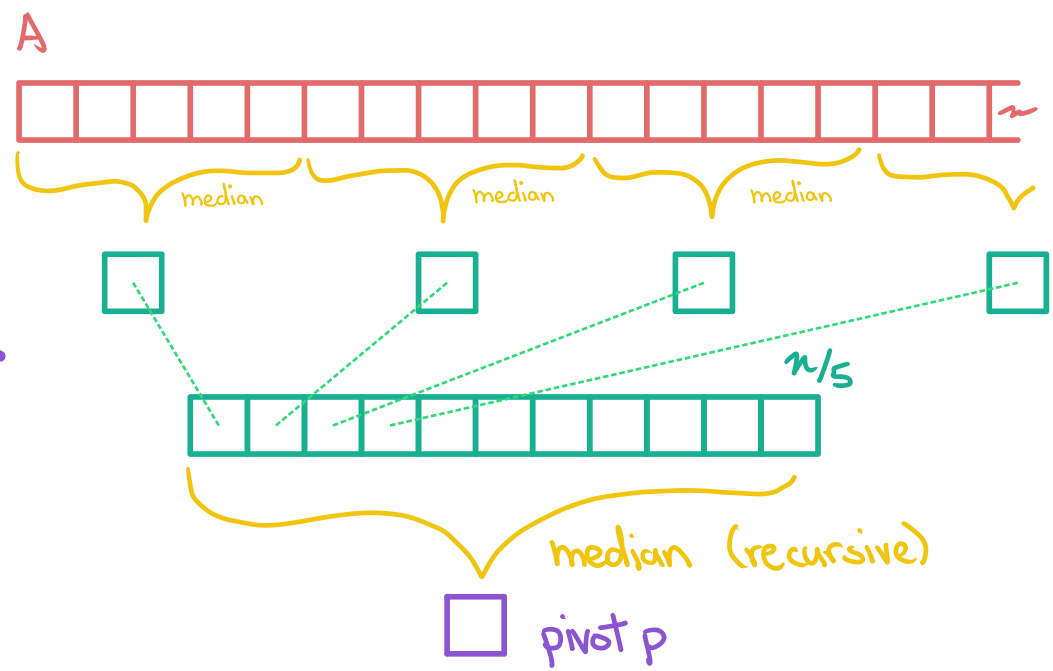


$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$$

claim:

M-of-M pivot p has rank

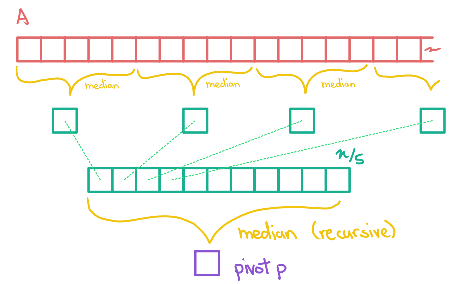
$$\frac{3n}{10} \le \text{rank}(p) \le \frac{7n}{10} + 5$$

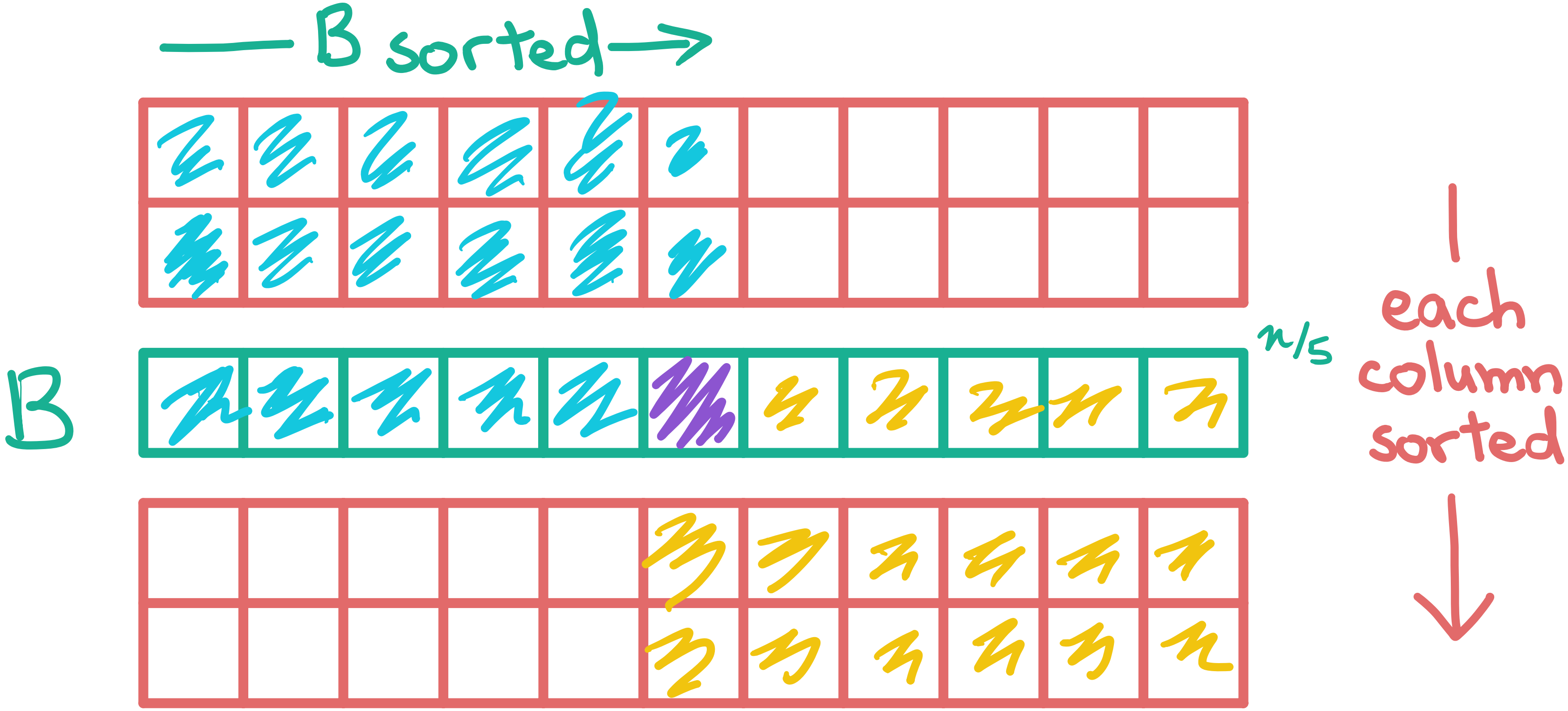


claim:

M-of-M pivot p has rank

$$\frac{3n}{10} \leq \text{rank}(p) \leq \frac{7n}{10} + 5$$





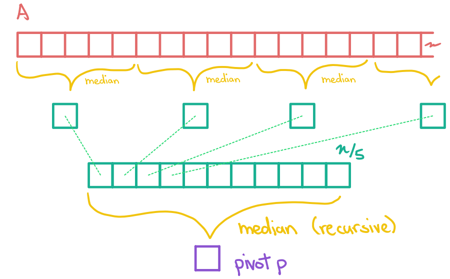
= def. smaller

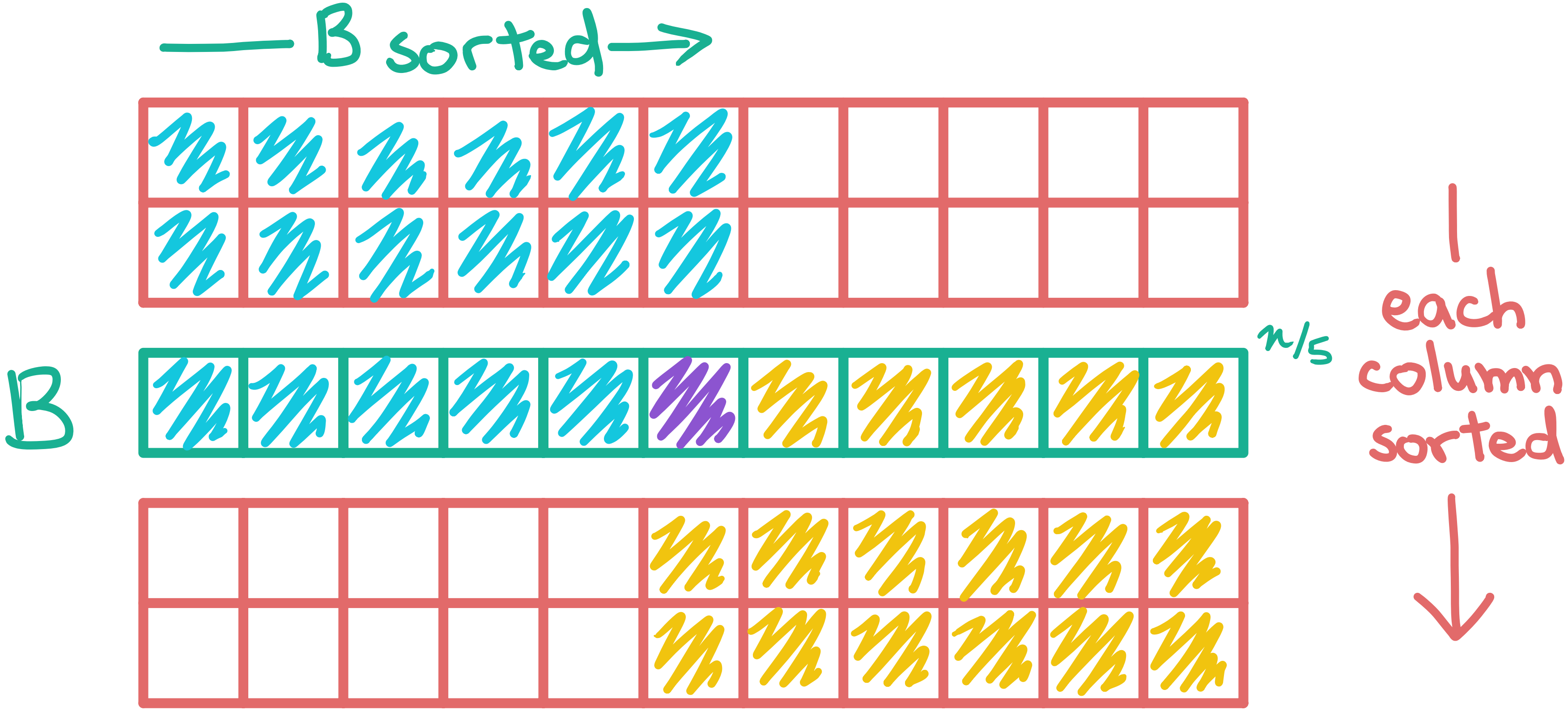
= def. bigger

claim:

M-of-M pivot p has rank

$$\frac{3n}{10} \le \text{rank}(p) \le \frac{7n}{10} + 5$$





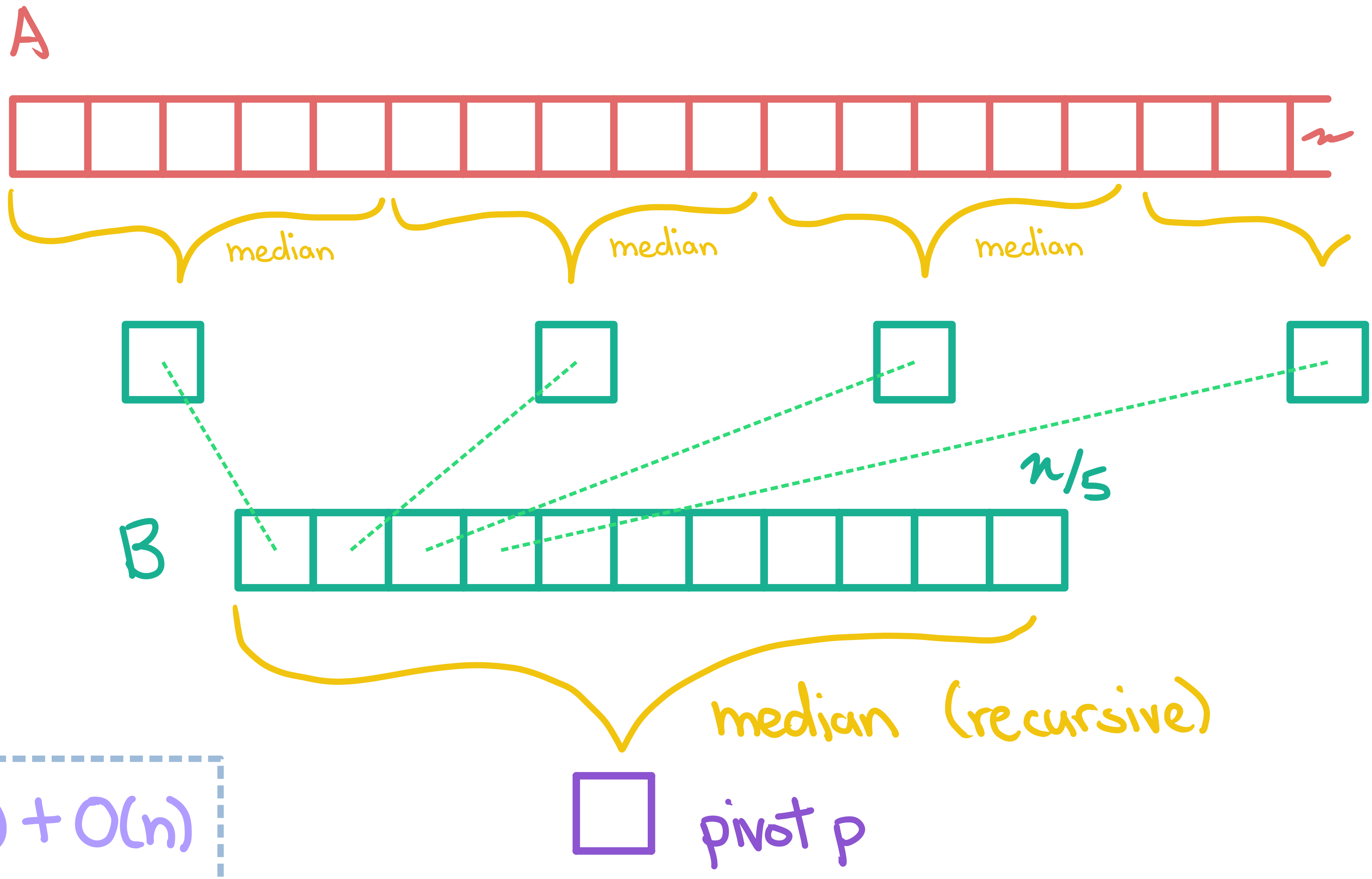
= def. smaller

= def. bigger

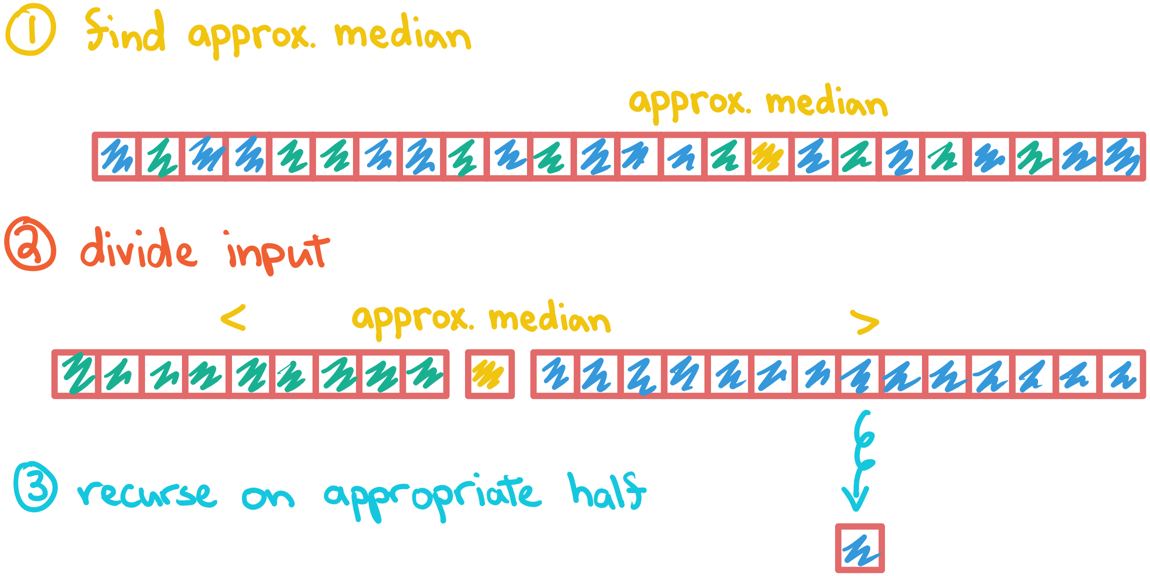
$$\frac{n}{5} \times \frac{1}{2} \times 3 = \frac{3n}{10}$$

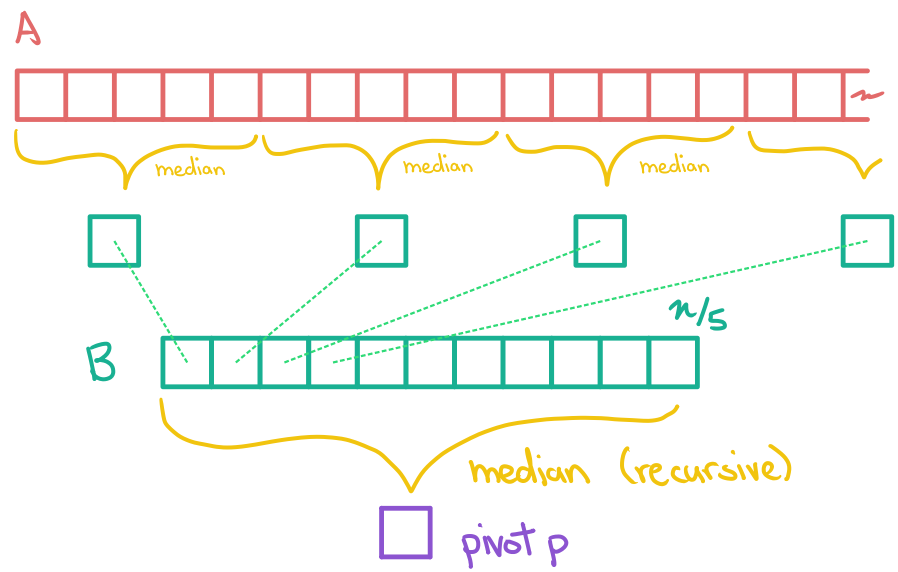
"Median of medians"

Blum, Floyd, Pratt, Rivest, Tarjan



$T(n/5) + O(n)$ time



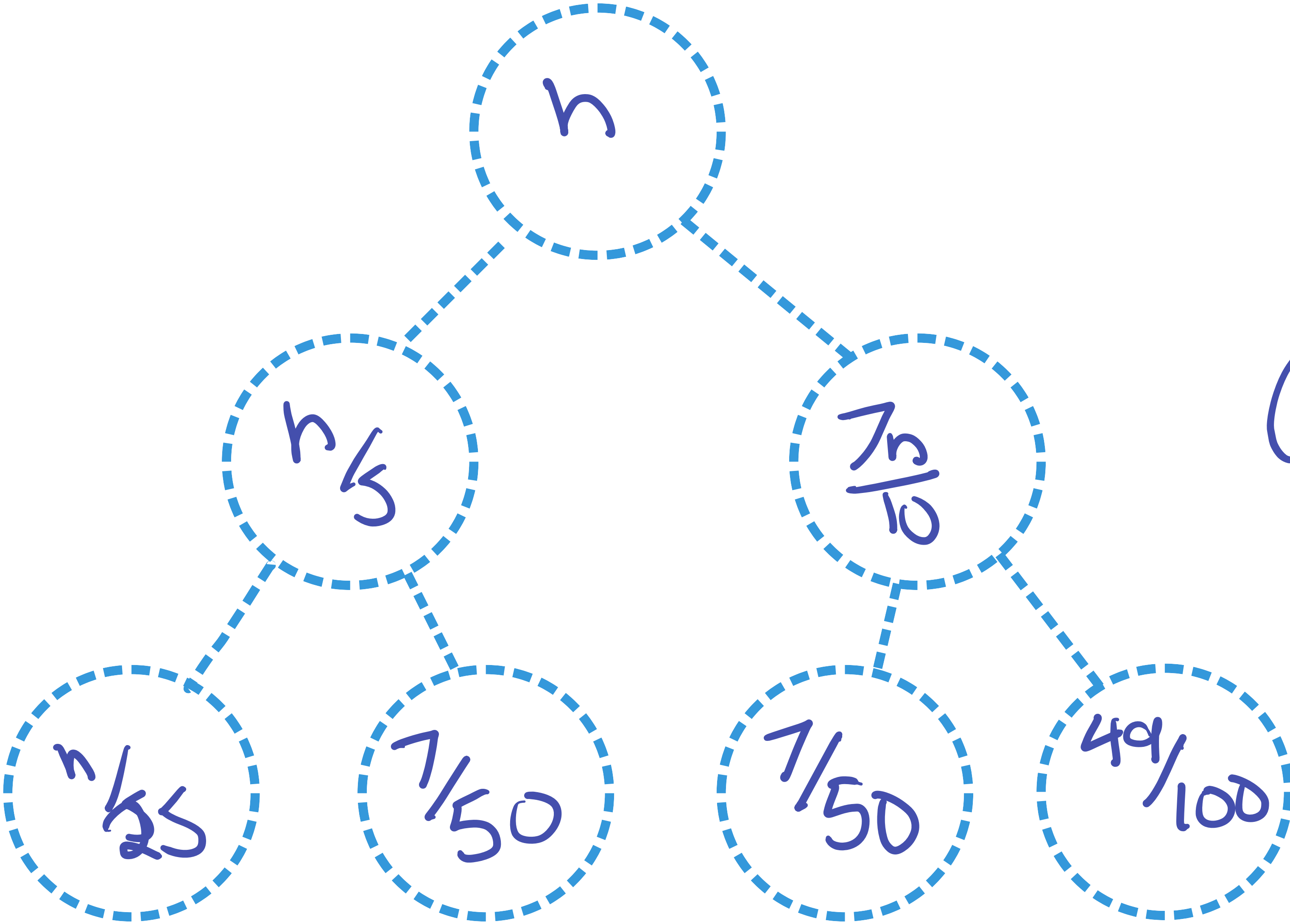


<u>`select`$(k, A[1..n])$</u>

/* `select`$(k, A[1..n])$ = *the kth smallest element in* $A[1..n]$. *We assume the elements in* A *are distinct for simplicity. (Otherwise, break ties consistently.)* */

1. If $n \leq 10$ then find the kth element by brute force.
2. Let $p =$ `median-of-medians`(A).
3. Compare p to each element in $A[1..n]$ to identify its rank ℓ.
4. If $k = \ell$ then return p.
5. If $k < \ell$ then return `select`(k, B), where $B[1..\ell - 1]$ is an array of the $\ell - 1$ elements smaller than p.
6. If $k > \ell$ then return `select`$(k - \ell, B)$, where $B[1..n - \ell]$ is an array of the $n - \ell$ elements larger than p.

$$T(n) = T(n/5) + O(n) + T\left(\frac{7n}{10}\right)$$



n

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

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

$$\sum \left(\frac{1}{5}+\frac{7}{10}\right)^i n = \sum \left(\frac{9}{10}\right)^i n$$

$$\cancel{\frac{1}{5}+\frac{7}{10}=}$$

$$= O(n)$$

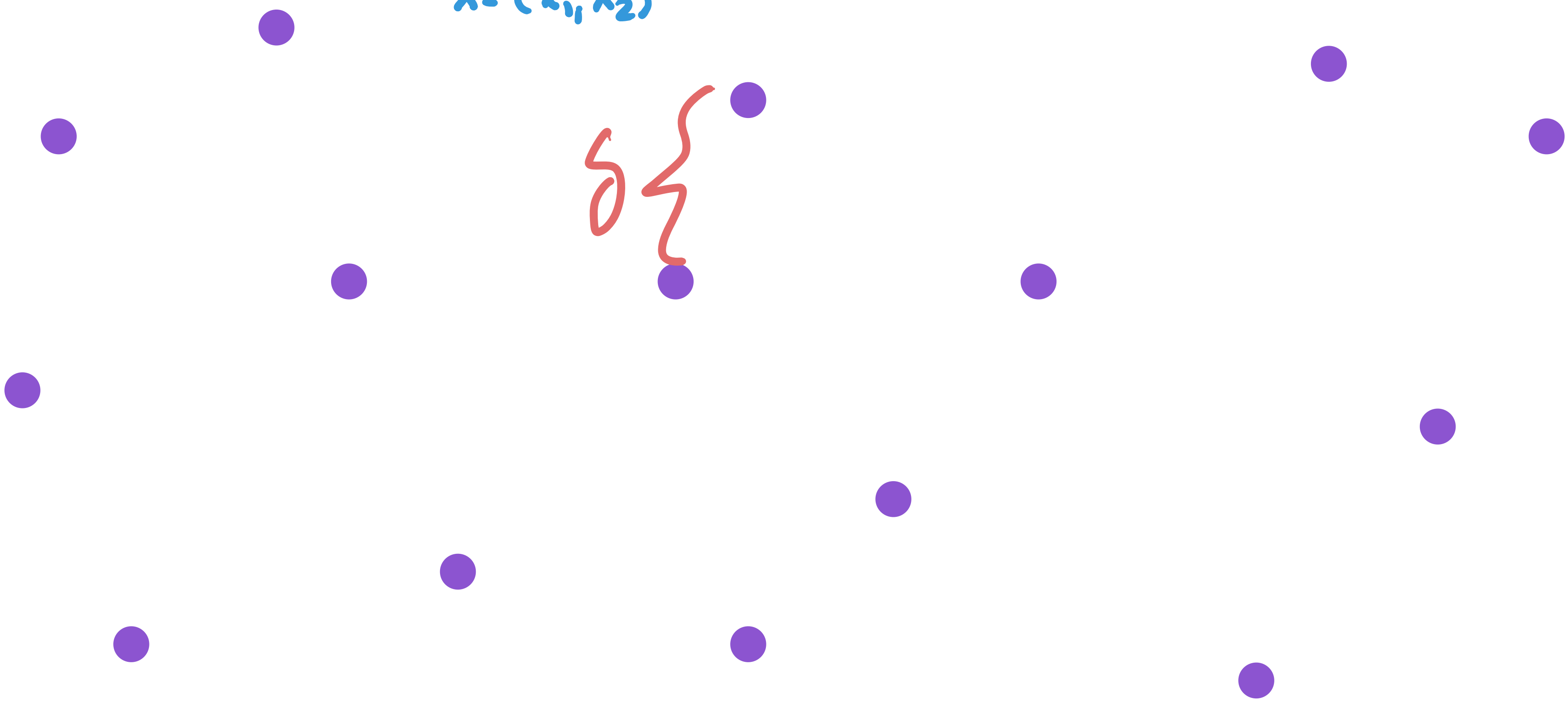
Minimum Euclidean distance

Input: n points in $\mathbb{R}^2$

Goal: min distance over all pairs

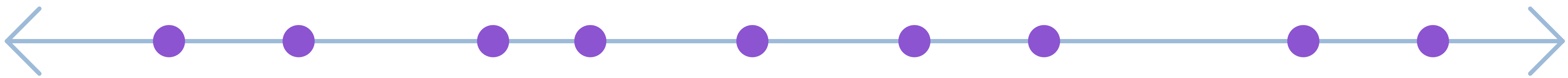
$\|x-y\| = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$ for $x, y \in \mathbb{R}^2$

$x = (x_1, x_2)$

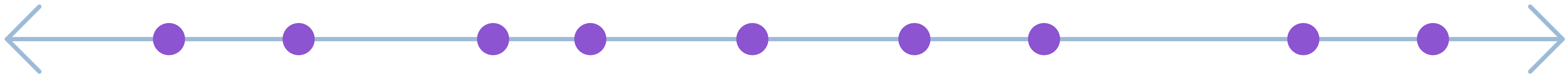


$O(n^2)$ obvious. Better?

Warmup: 1 dimension (all y-coordinates = 0)



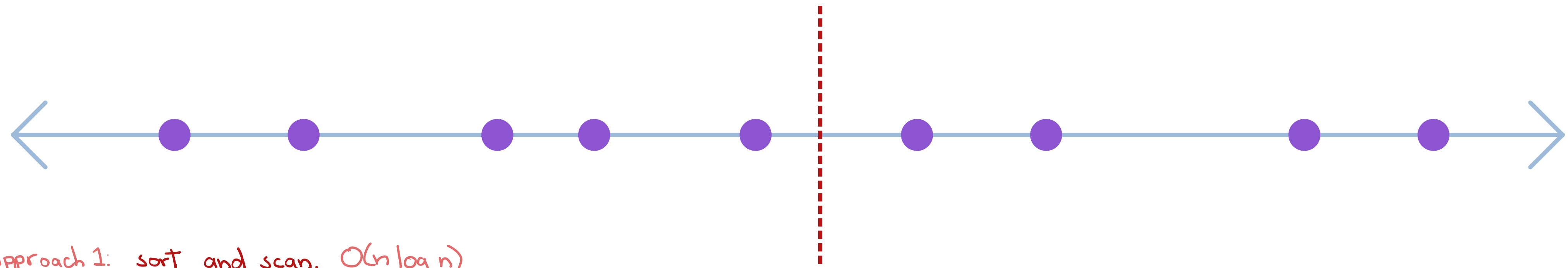
Warmup: 1 dimension (all y-coordinates = 0)



Approach 1: sort and scan. $O(n \log n)$

(hard to generalize to $\mathbb{R}^2$)

Warmup: 1 dimension (all y-coordinates = 0)



Approach 1: sort and scan. $O(n \log n)$

(hard to generalize to $\mathbb{R}^2$)

Approach 2.

Split at median, recurse

Compare w/ min pair "across" split

$O(n \log n)$ (like merge sort)

seems to generalize better

Generalizing to $\mathbb{R}^2$

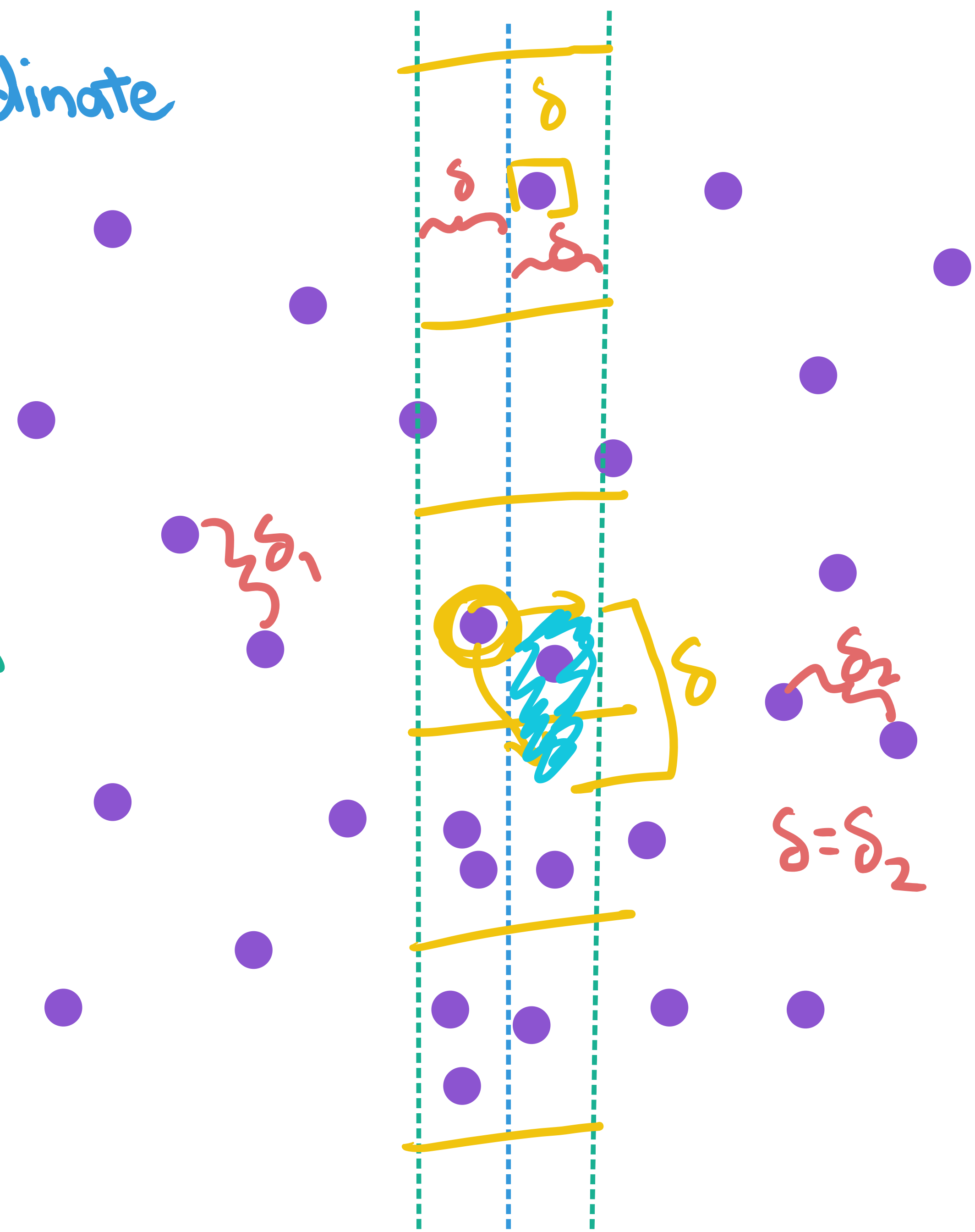
Approach 2.

Split at median, recurse

Compare w/ min pair "across" split

1. split at median x-coordinate
2. recurse on each half

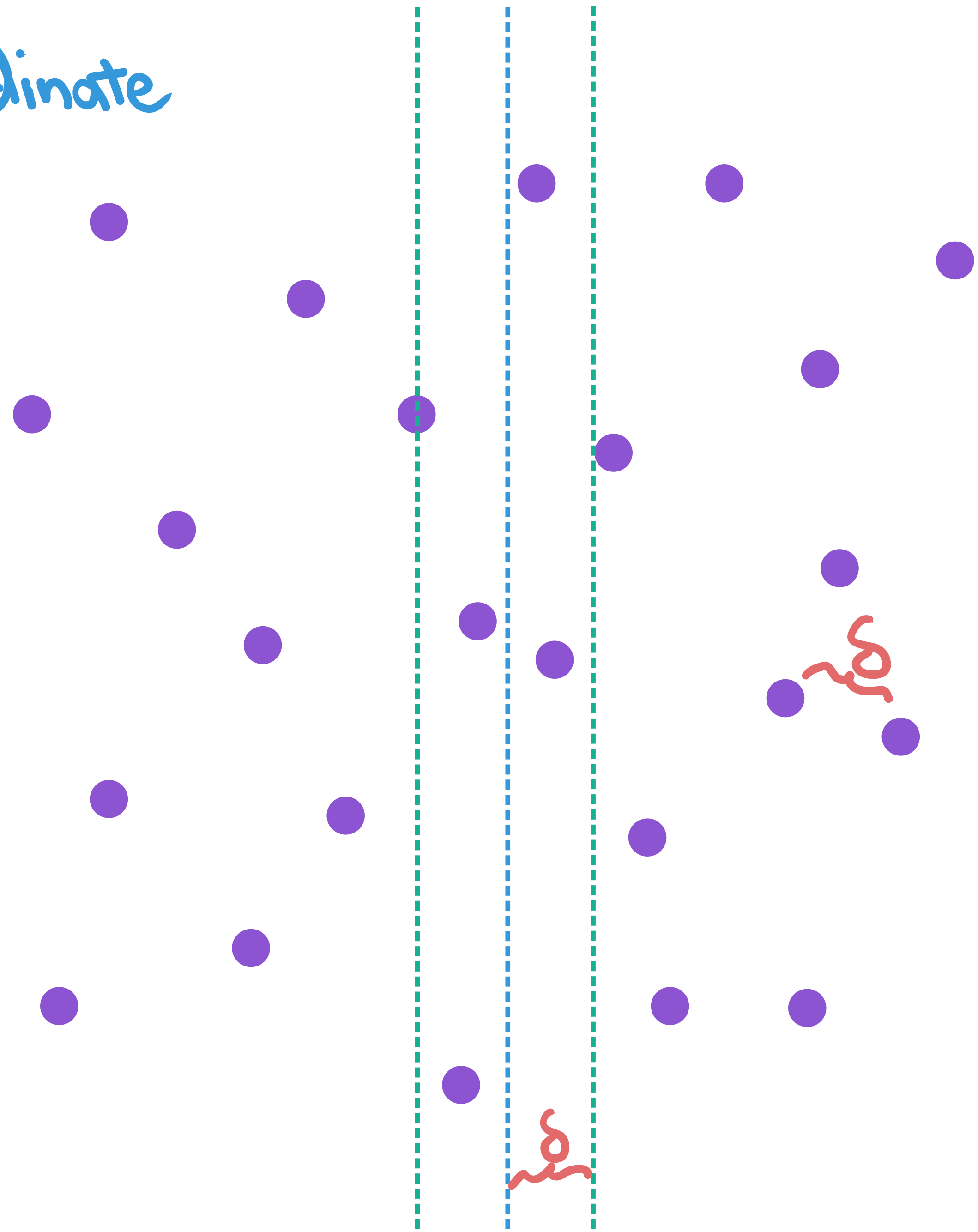
   let δ = min distance from the two halves
3. look at "vertical band" of width 2δ around median
4. compute min distance inside band



Generalizing to $\mathbb{R}^2$

Approach 2.
Split at median, recurse
Compare w/ min pair "across" split

1. split at median x-coordinate
2. recurse on each half
   let δ = min distance from the two halves
3. look at "vertical band" of width 2δ around median
4. compute min distance inside band



1. split at median x-coordinate

2. recurse on each half,
   let δ = min dist. from two halves

3. look at "vertical band" of width 2δ around median line

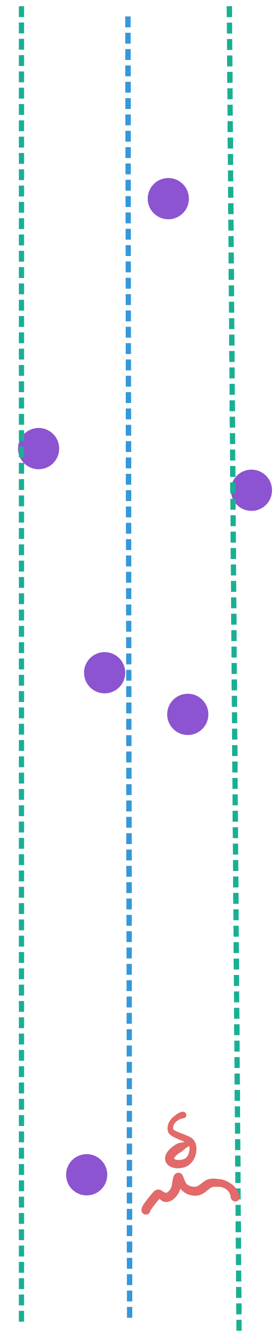
4. compute min distance inside band

Approach 2.
Split at median, recurse
Compare w/ min pair "across" split

Observations

- very narrow
- points on left side have min distance $\geq \delta$
- points on right side have min distance $\geq \delta$

Intuitively: not dense,
fewer comparisons needed

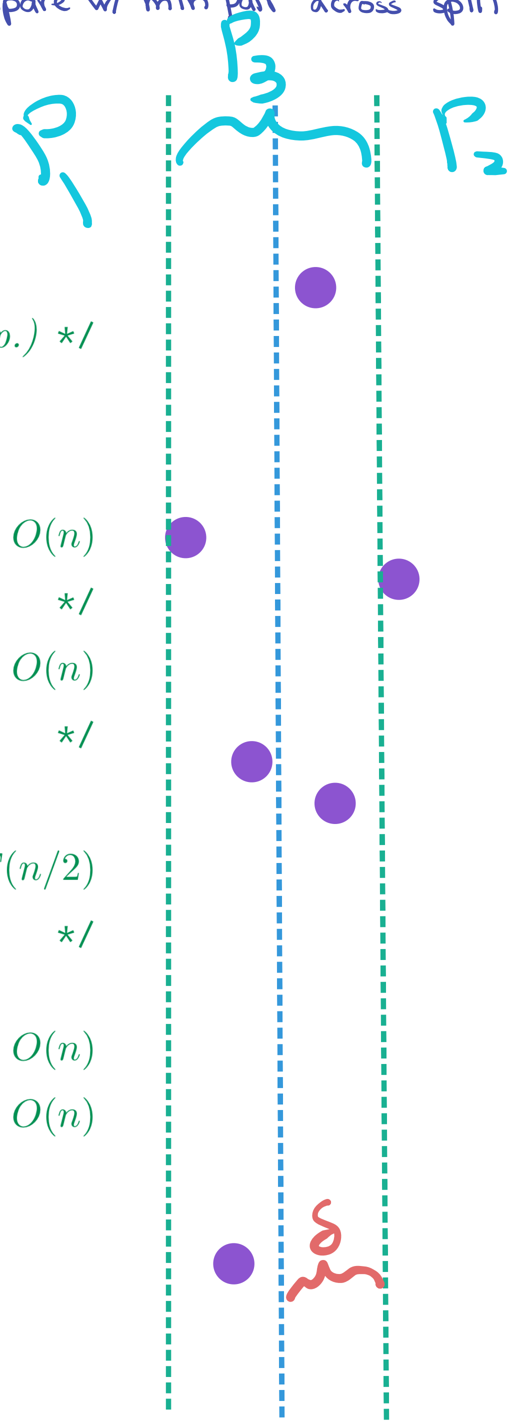


1. split at median x-coordinate
2. recurse on each half,
   let δ = min dist. from two halves
3. look at "vertical band" of width 2δ around median
4. compute min distance inside band

Observations
- very narrow
- points on left side have min distance $\geq \delta$
- points on right side have min distance $\geq \delta$

Intuitively: not dense,
fewer comparisons needed

Approach 2.
Split at median, recurse
Compare w/ min pair "across" split



min-distance(P)

/* *We assume P is sorted by y-coordinate. (Otherwise sort P once in a preprocessing step.)* */

1. If $n \leq 1$ then return $+\infty$.
2. If $n = 2$ then return the distance between the two points in P.
3. Let a be the median of the x-coordinates. // $O(n)$

/* *Divide P along the vertical line $x = a$* */

4. Let $P_1 = \{(x, y) \in P : x < a\}$ and $P_2 = \{(x, y) \in P : x \geq a\}$ // $O(n)$

/* *Find the minimum distances within each half P_1 and P_2 recursively.* */

5. Let δ_1 = min-distance(P_1), δ_2 = min-distance(P_2), and $\delta = \min\{\delta_1, \delta_2\}$. // $2T(n/2)$

/* *It remains to find the minimum distance across P_1 and P_2.* */

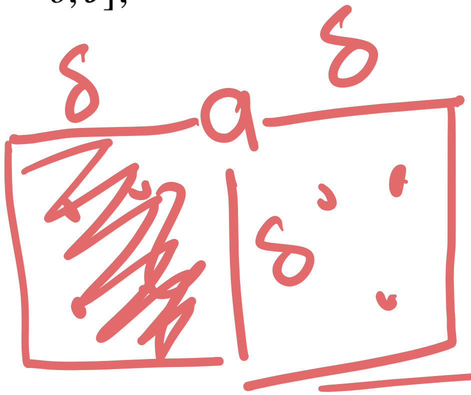
6. Let $P_3 = \{(x, y) \in P : |x - a| < \delta\}$ and let $p_1, \ldots, p_k$ list P_3 in decreasing order of y-coordinate. // $O(n)$
7. Let δ_3 be the minimum of $\|p_i - p_j\|$ over all i, j with $i < j < i + 10$. // $O(n)$
8. Return $\min\{\delta, \delta_3\}$.

(a = median x-coordinate)

Lemma 11.3. *Let $b \in \mathbb{R}$. There are at most 10 points in P with coordinate in the rectangle $[a-\delta, a+\delta] \times [b-\delta, b]$,*

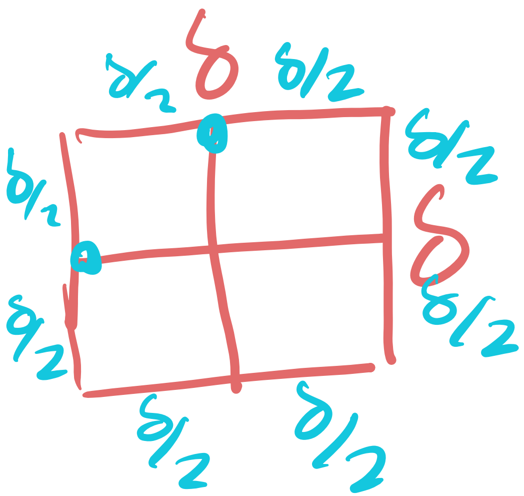
Observations
- very narrow
- points on left side have min distance $\geq \delta$
- points on right side have min distance $\geq \delta$

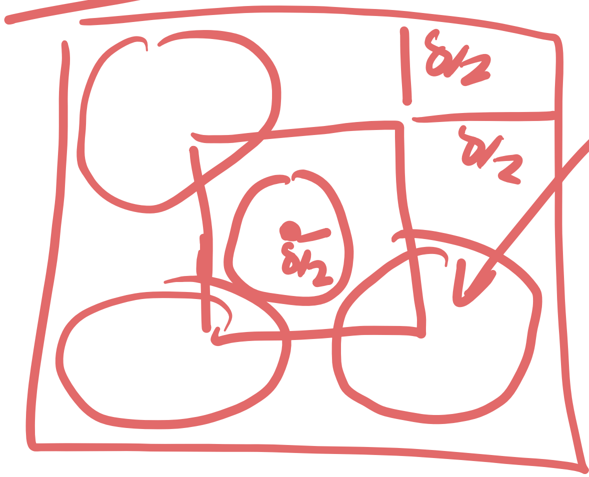
Intuitively: not dense, fewer comparisons needed



Area $4\delta^2 = (2\delta)^2$

Area $\frac{\pi}{4}\delta^2$

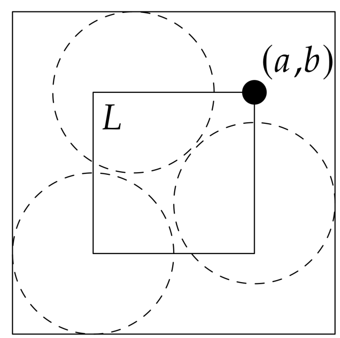


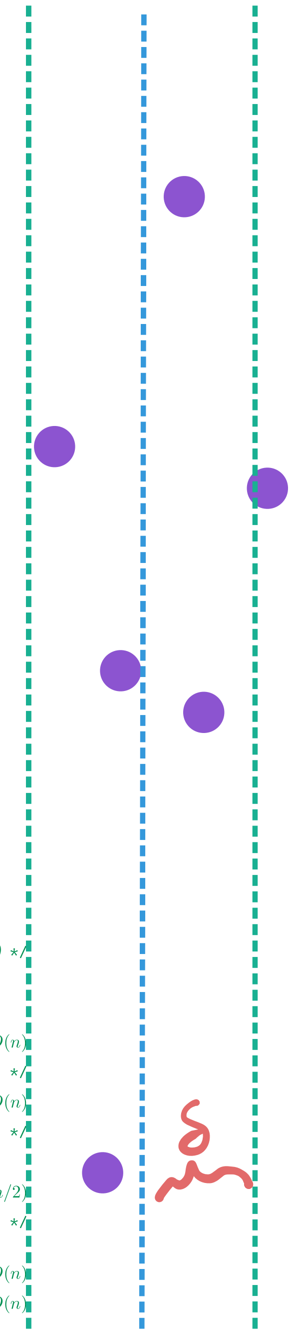


$$\sqrt{\left(\frac{\delta}{2}\right)^2 + \left(\frac{\delta}{2}\right)^2}$$

$$= \sqrt{\frac{1}{2}\delta^2} < \delta$$

$L = [a-\delta, a) \times [b-\delta, b]$ and $R = [a, a+\delta] \times [b-\delta, b]$





min-distance(P)

/* *We assume P is sorted by y-coordinate. (Otherwise sort P once in a preprocessing step.)* */

1. If $n \leq 1$ then return $+\infty$.
2. If $n = 2$ then return the distance between the two points in P.
3. Let a be the median of the x-coordinates. // $O(n)$

/* *Divide P along the vertical line $x = a$* */

4. Let $P_1 = \{(x,y) \in P : x < a\}$ and $P_2 = \{(x,y) \in P : x \geq a\}$ // $O(n)$

/* *Find the minimum distances within each half P_1 and P_2 recursively.* */

5. Let $\delta_1 =$ min-distance(P_1), $\delta_2 =$ min-distance(P_2), and $\delta = \min\{\delta_1, \delta_2\}$. // $2T(n/2)$

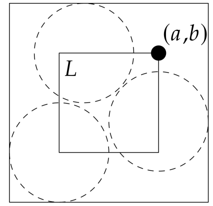
/* *It remains to find the minimum distance across P_1 and P_2.* */

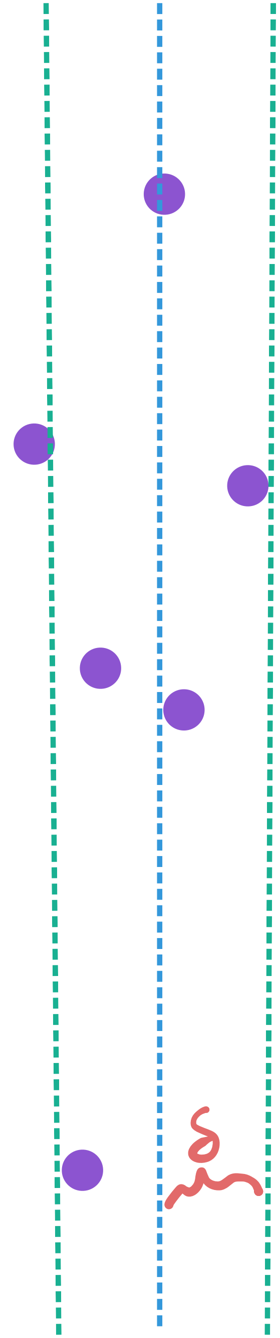
6. Let $P_3 = \{(x,y) \in P : |x - a| < \delta\}$ and let $p_1, \ldots, p_k$ list P_3 in decreasing order of y-coordinate. // $O(n)$
7. Let δ_3 be the minimum of $\|p_i - p_j\|$ over all i, j with $i < j < i + 10$. // $O(n)$
8. Return $\min\{\delta, \delta_3\}$.

Lemma 11.3. *Let $b \in \mathbb{R}$. There are at most 10 points in P with coordinate in the rectangle $[a-\delta, a+\delta] \times [b-\delta, b]$,*

Observations
- very narrow
- points on left side have min distance $\geq \delta$
- points on right side have min distance $\geq \delta$

Intuitively: not dense, fewer comparisons needed





min-distance(P)

/* *We assume P is sorted by y-coordinate. (Otherwise sort P once in a preprocessing step.)* */

1. If $n \leq 1$ then return $+\infty$.
2. If $n = 2$ then return the distance between the two points in P.
3. Let a be the median of the x-coordinates. // $O(n)$

/* *Divide P along the vertical line $x = a$* */

4. Let $P_1 = \{(x, y) \in P : x < a\}$ and $P_2 = \{(x, y) \in P : x \geq a\}$ // $O(n)$

/* *Find the minimum distances within each half P_1 and P_2 recursively.* */

5. Let $\delta_1 =$ min-distance(P_1), $\delta_2 =$ min-distance(P_2), and $\delta = \min\{\delta_1, \delta_2\}$. // $2T(n/2)$

/* *It remains to find the minimum distance across P_1 and P_2.* */

6. Let $P_3 = \{(x, y) \in P : |x - a| < \delta\}$ and let $p_1, \ldots, p_k$ list P_3 in decreasing order of y-coordinate. // $O(n)$
7. Let δ_3 be the minimum of $\|p_i - p_j\|$ over all i, j with $i < j < i + 10$. // $O(n)$
8. Return $\min\{\delta, \delta_3\}$.

Running time

Observations
- very narrow
- points on left side have min distance $\geq \delta$
- points on right side have min distance $\geq \delta$

Intuitively: not dense, fewer comparisons needed

<u>`min-distance`(P)</u>

/ We assume P is sorted by y-coordinate. (Otherwise sort P once in a preprocessing step.) */*

1. If $n \leq 1$ then return $+\infty$.
2. If $n = 2$ then return the distance between the two points in P.
3. Let a be the median of the x-coordinates. // $O(n)$

/ Divide P along the vertical line $x = a$ */*

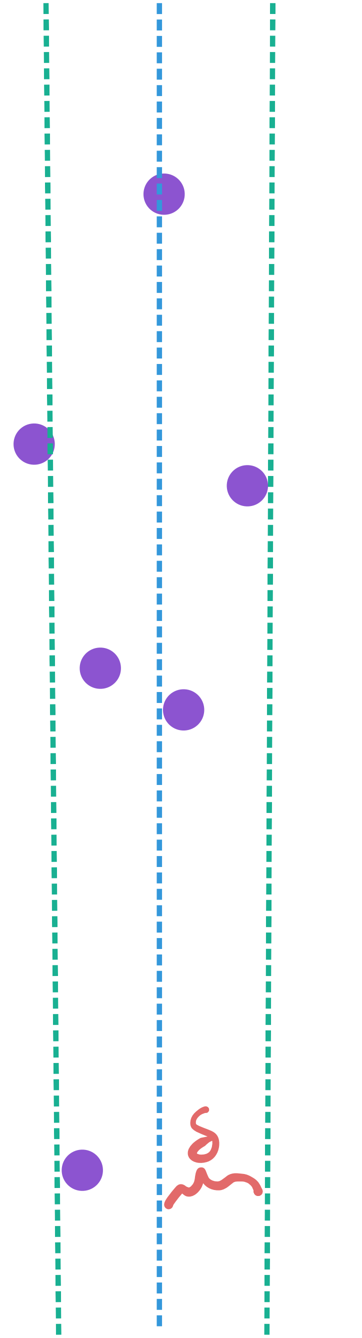
4. Let $P_1 = \{(x, y) \in P : x < a\}$ and $P_2 = \{(x, y) \in P : x \geq a\}$ // $O(n)$

/ Find the minimum distances within each half P_1 and P_2 recursively. */*

5. Let $\delta_1 =$ `min-distance`(P_1), $\delta_2 =$ `min-distance`(P_2), and $\delta = \min\{\delta_1, \delta_2\}$. // $2T(n/2)$

/ It remains to find the minimum distance across P_1 and P_2. */*

6. Let $P_3 = \{(x, y) \in P : |x - a| < \delta\}$ and let $p_1, \ldots, p_k$ list P_3 in decreasing order of y-coordinate. // $O(n)$
7. Let δ_3 be the minimum of $\|p_i - p_j\|$ over all i, j with $i < j < i + 10$. // $O(n)$
8. Return $\min\{\delta, \delta_3\}$.



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

Multiplying n-bit integers

Karatsuba

Input: $x, y \in \{0,1\}^n$

Output: product $(xy) \in \{0,1\}^{2n}$

$$
\begin{array}{r}
1234 \\
\times \; 5678 \\
\hline
9872 \\
8638 \\
+ \qquad\qquad \\
\hline
\end{array}
$$

$O(n^2)$ time

Better?