Selection & Closest Pair

Undergraduate

welcome to ~~Graduate~~ Algorithms

fundamentalalgorithms.com

Last compiled January 12, 2026 at 12:45 Noon.

Fundamental Algorithms, CS381, Spring 2026

Instructors:

- *Mike Atallah* (`matallah@...`).
  Lectures: Tuesday and Thursday, 10:30–11:45AM, SMTH 108.
  Office hours: Thursday, after lecture, in LWSN 2116D.
- *Kent Quanrud* (`krq@...`).
  Lectures: Tuesday and Thursday, 1:30–2:45PM, BHEE 170.
  Office hours: Tuesday, after lecture, in LWSN 1211.

Graduate TAs:	Email (@purdue.edu)	Office hours (LWSN Commons)[1]
Kyle Asbury	asbury7	Thursday, 1:30–2:30PM
Arnav Burudgunte	aburudgu	TBD
Qi Ling	ling102	Thursday, 9:30–10:30AM
Ethan Mader	emader	Thursday, 4–5PM
Han Qin	qin184	Tuesday, 2:30–3:30PM
Haoyu Song	song522	Tuesday, 1:30–2:30PM
Borna Tavasoli	btavasol	Thursday, 6–7PM
Justin Zhang	zhan3554	Wednesday, 12:30–1:30PM, Thursday, 3:30–4:30PM

Undergraduate TAs:	Email (@purdue.edu)	Office hours (LWSN Commons)[1]

Please read the syllabus

- will ask for questions next class

Highlights

Awesome TA's

2 midterms, final

weekly homework

drop lowest 20% of HW scores

late/resubmit HW policy

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, $\boxed{56}$, 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

select(A[1..n], k) given array A[1..n] of comp. elements and rank k, return rank element

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

`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 $A[1..n]$.

Designing a recursive algorithm

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

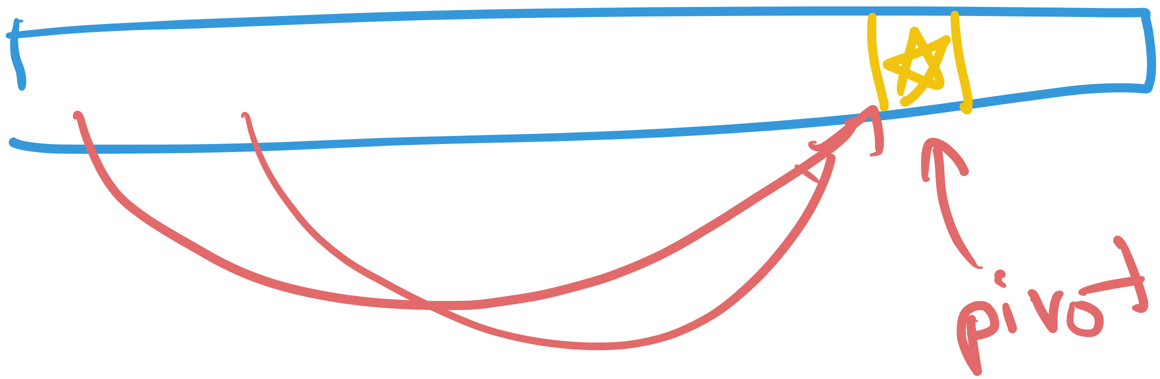
② Implement spec

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

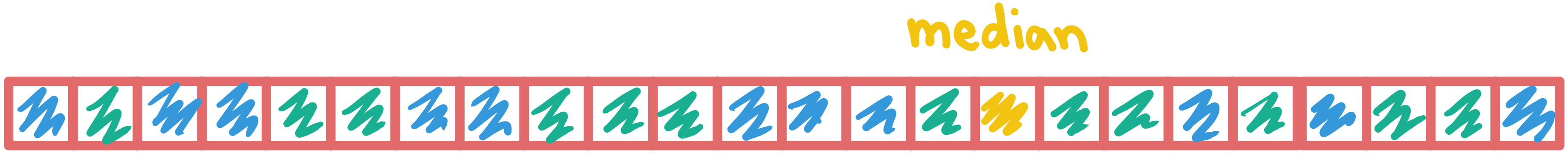
② Implement spec

thought experiment: suppose O(n)-time subroutine

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



1. find median



2. divide input

3. recurse on appropriate half

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

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

`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.

Running time

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

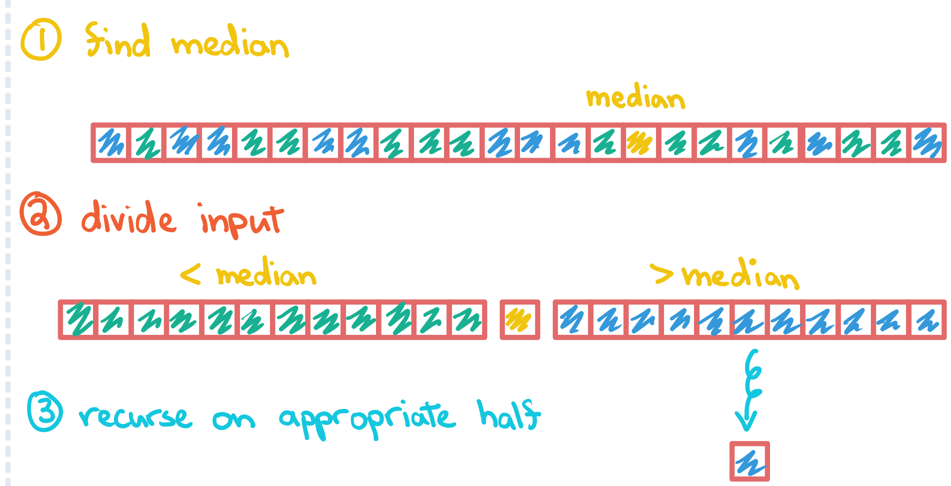
n	n
$n/2$	$n/2$
$n/4$	$n/4$
$n/8$	$n/8$
$\vdots$	
$+\ 1$	

$$O(n)$$

`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 $n \le 10$ 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.



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 $A[1..n]$.

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.)* */

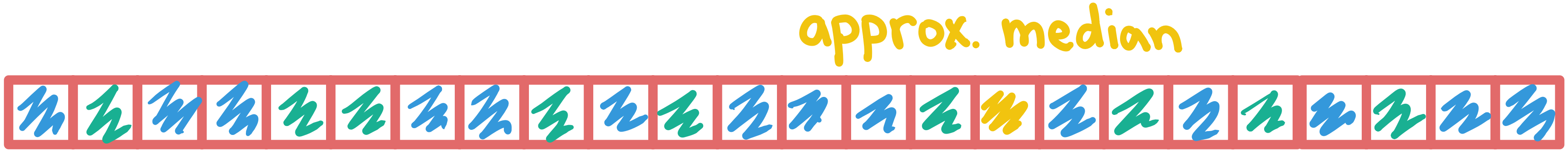
1. If $n \le 10$ then find the kth element by brute force.
2. Let $p =$ `median`(A). // *We assume a linear time subroutine for* `median`$(A[1..n])$.
3. If $k = \lceil n/2 \rceil$ then return p.
4. If $k < \lceil n/2 \rceil$ then return `select`(k, B), where $B[1..\lceil n/2 \rceil - 1]$ is an array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

O(n) time

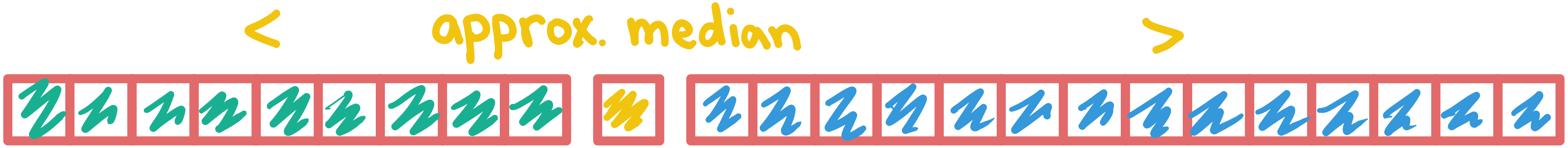
New thought experiment: suppose O(n)-time subroutine

`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)

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

`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 $n \le 10$ 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.

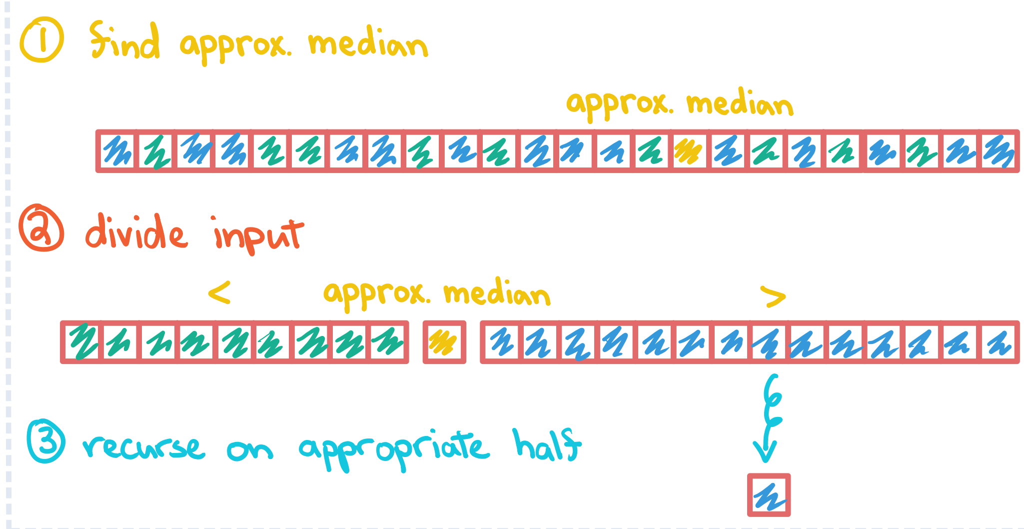
Running time

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 $n \leq 10$ 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.

$$T(n) = T(9n/10) + O(n)$$



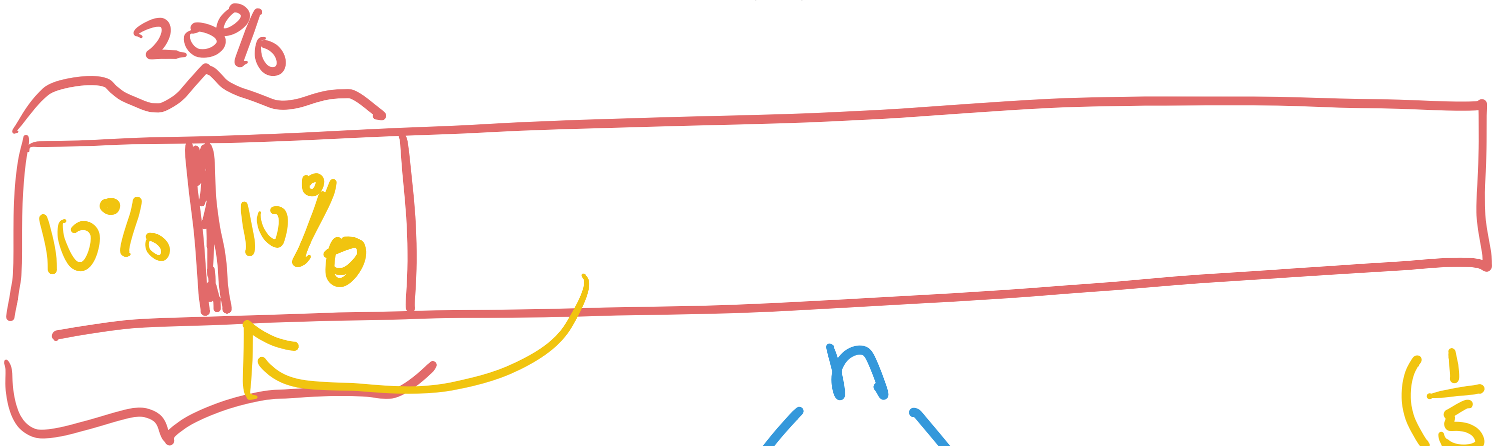
$$n$$
$$\frac{9}{10} n$$
$$\left(\frac{9}{10}\right)^2 n$$
$$\vdots$$
$$+ \underline{\hspace{6em}}$$

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

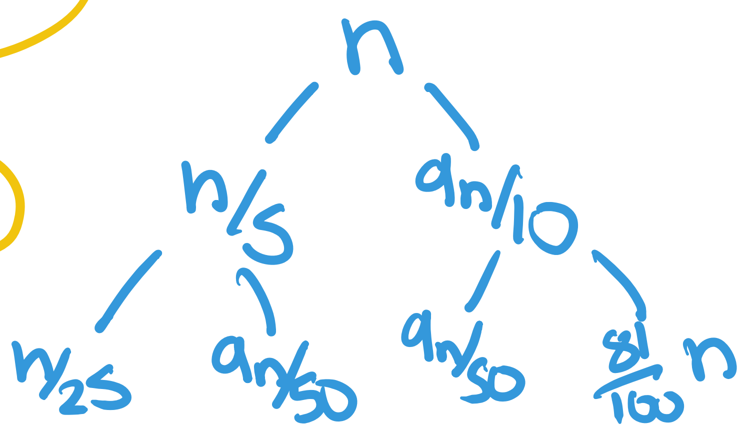
$$\underbrace{\tfrac{9}{10}}_{\text{constant} < 1}$$

Remains to implement

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.



median p



n

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

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

$\left(\frac{1}{5}+\frac{9}{10}\right)^3 n$

$$T(n) = T(n/5) + T(9n/10) + O(n)$$

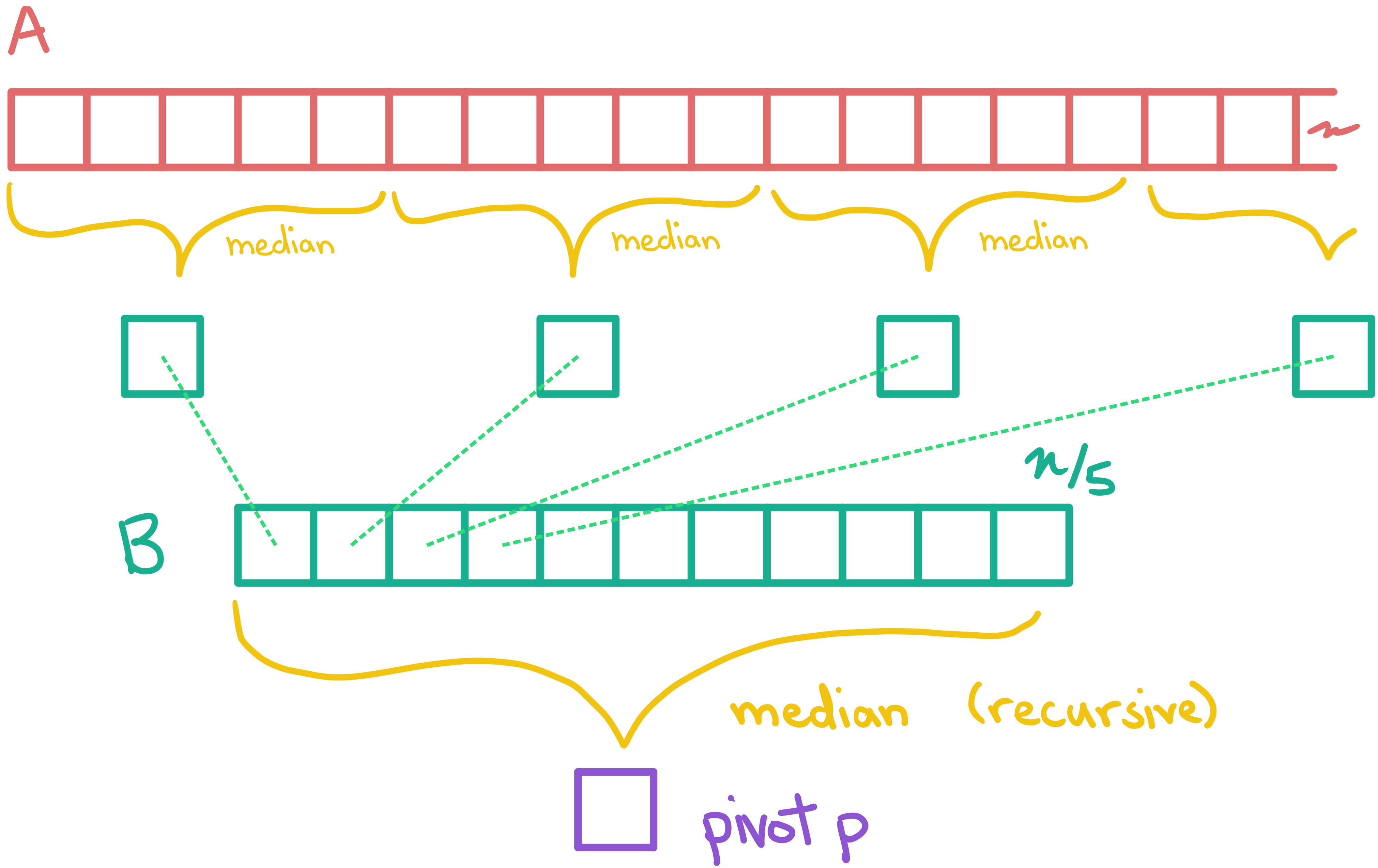
$$\sum_{i=0}^{\log_{10/9} n} \left(\frac{1}{5}+\frac{9}{10}\right)^i \cdot n = \left(\frac{11}{10}\right)^{\log_{10/9} n} \cdot n$$

$$n^{1+\log_{10/9} 11/10}$$

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

"Median of medians"

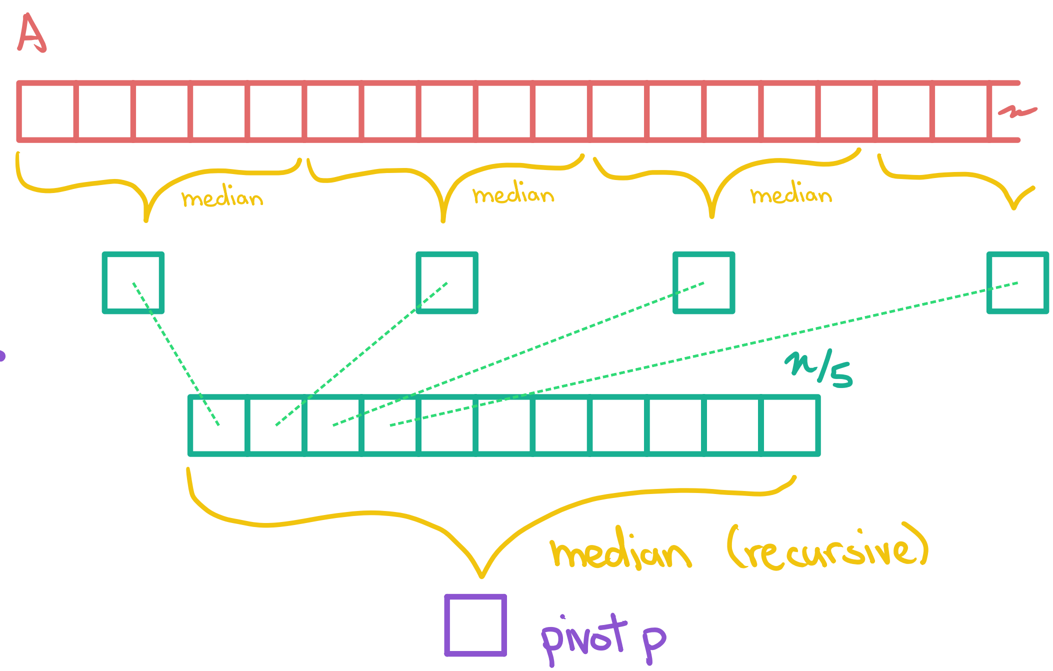
Blum, Floyd, Pratt,
Rivest, Tarjan 72



claim:

M-of-M pivot p has rank

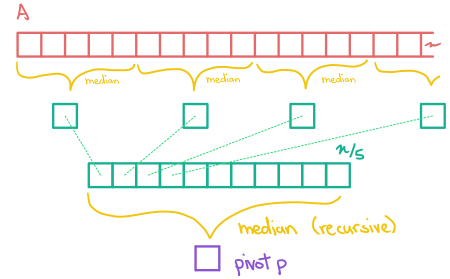
$$\frac{3n}{10} \leq \text{rank}(p) \leq \frac{7n}{10} + 5$$



claim:

M-of-M pivot p has rank

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



= def. smaller

= def. bigger

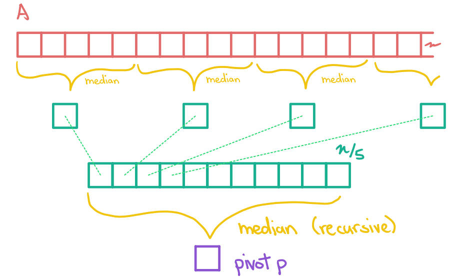
$$\frac{3}{10} n - 4$$

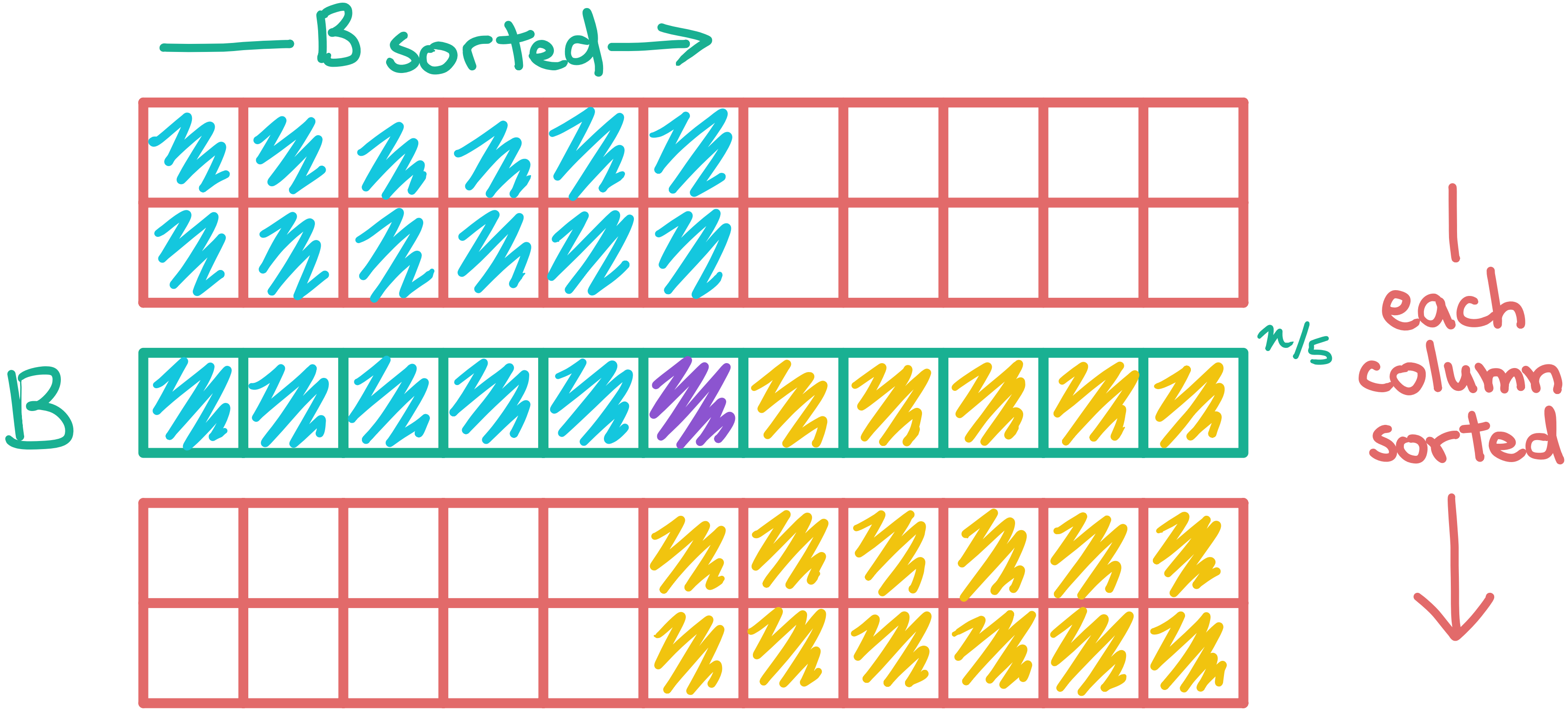
$$\frac{3}{5} \cdot \frac{1}{2} = \frac{3}{10} n$$

claim:

M-of-M pivot p has rank

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





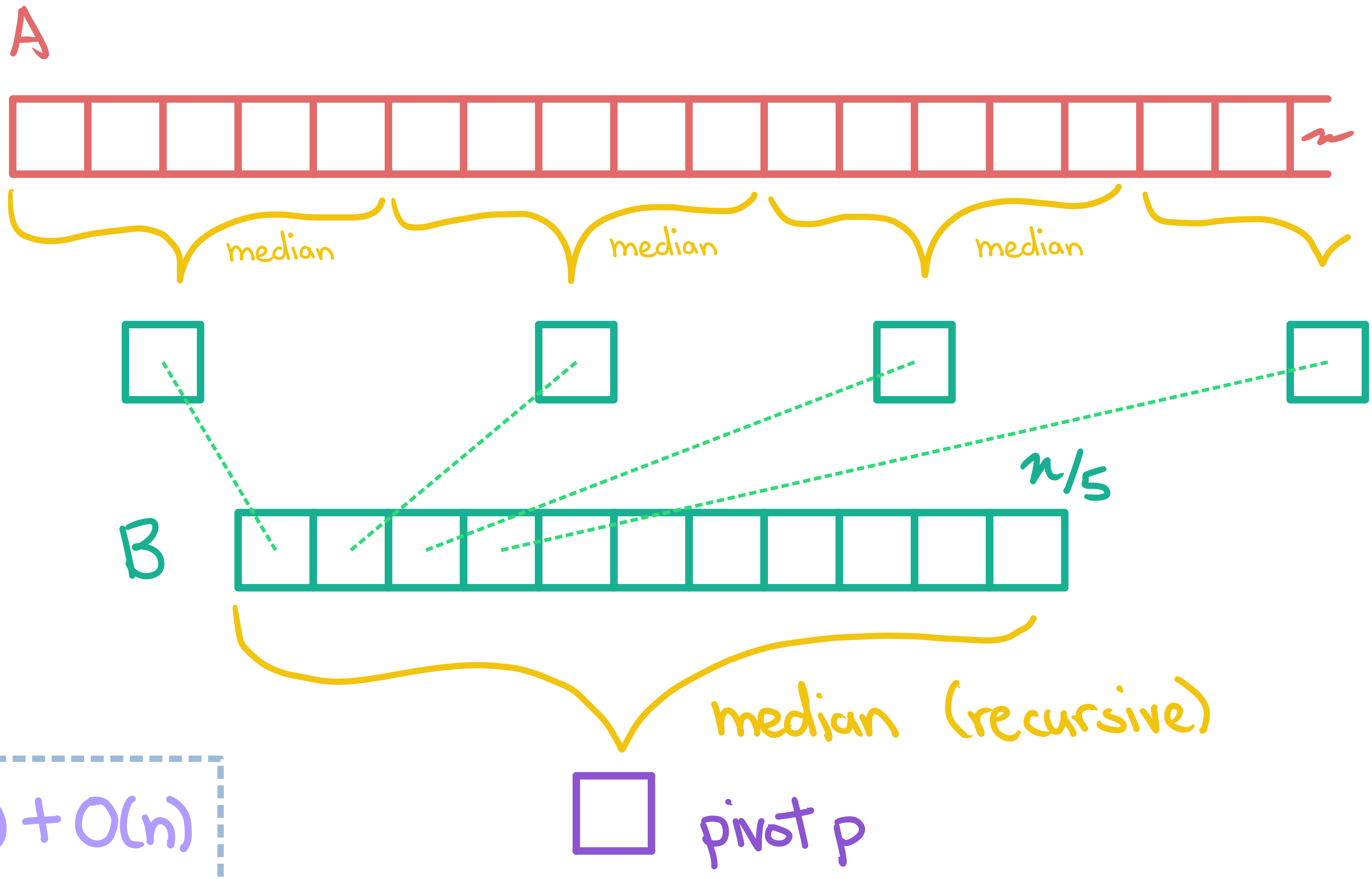
= def. smaller

= def. bigger

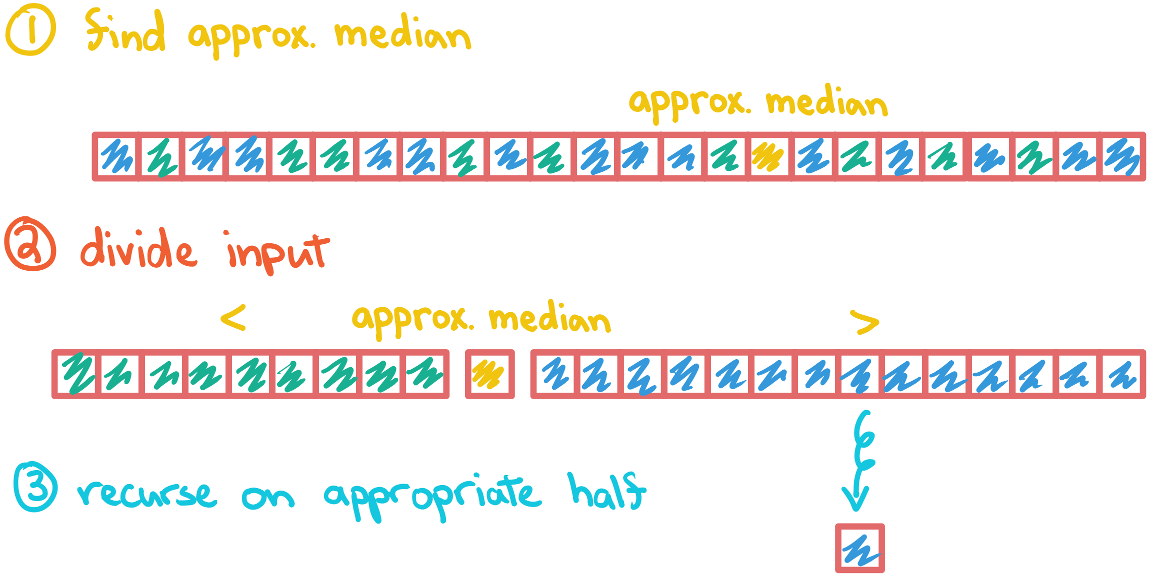
$$\frac{n}{5} \times \frac{1}{2} \times 3 = \frac{3n}{10}$$

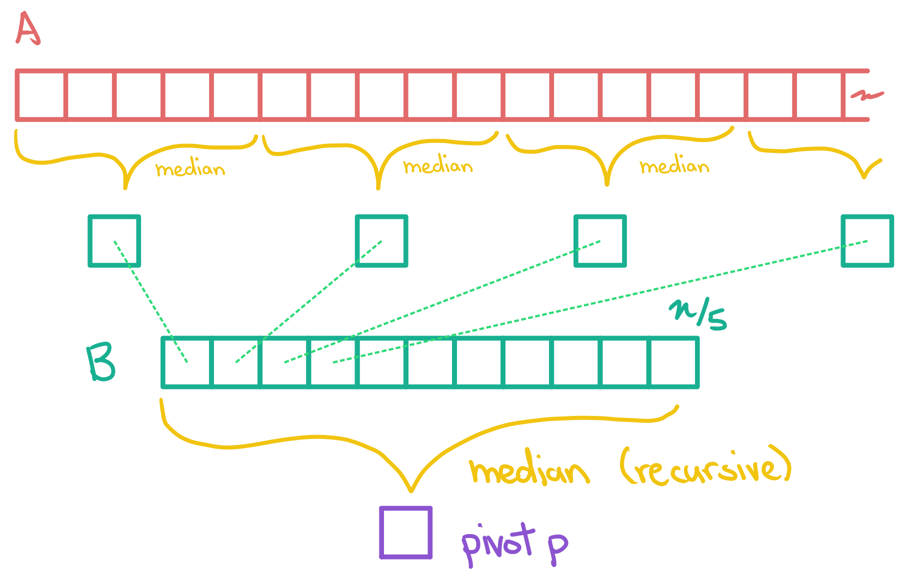
"Median of medians"

Blum, Floyd, Pratt, Rivest, Tarjan



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



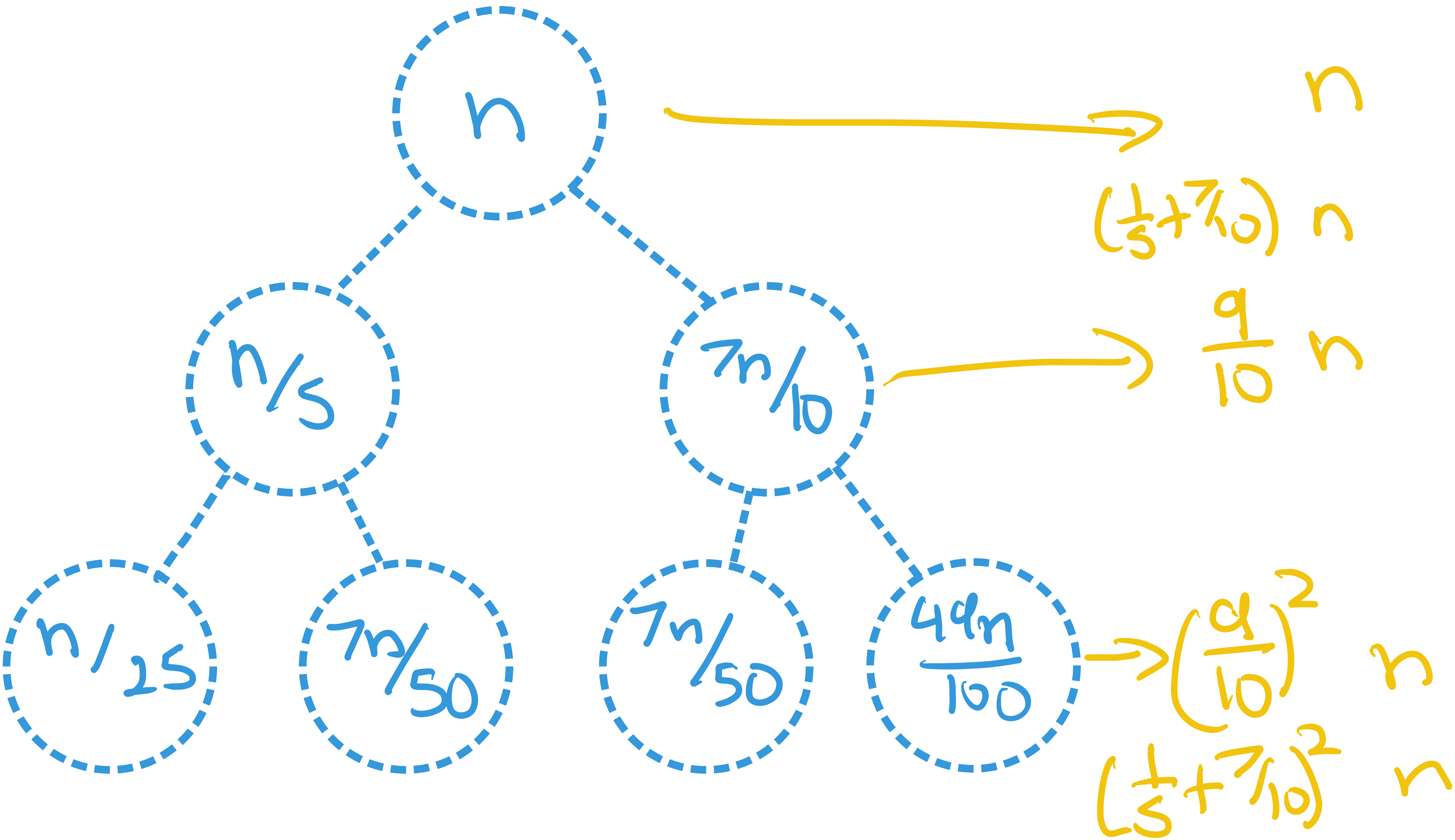


select(k, $A[1..n]$)

/* 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.)* */

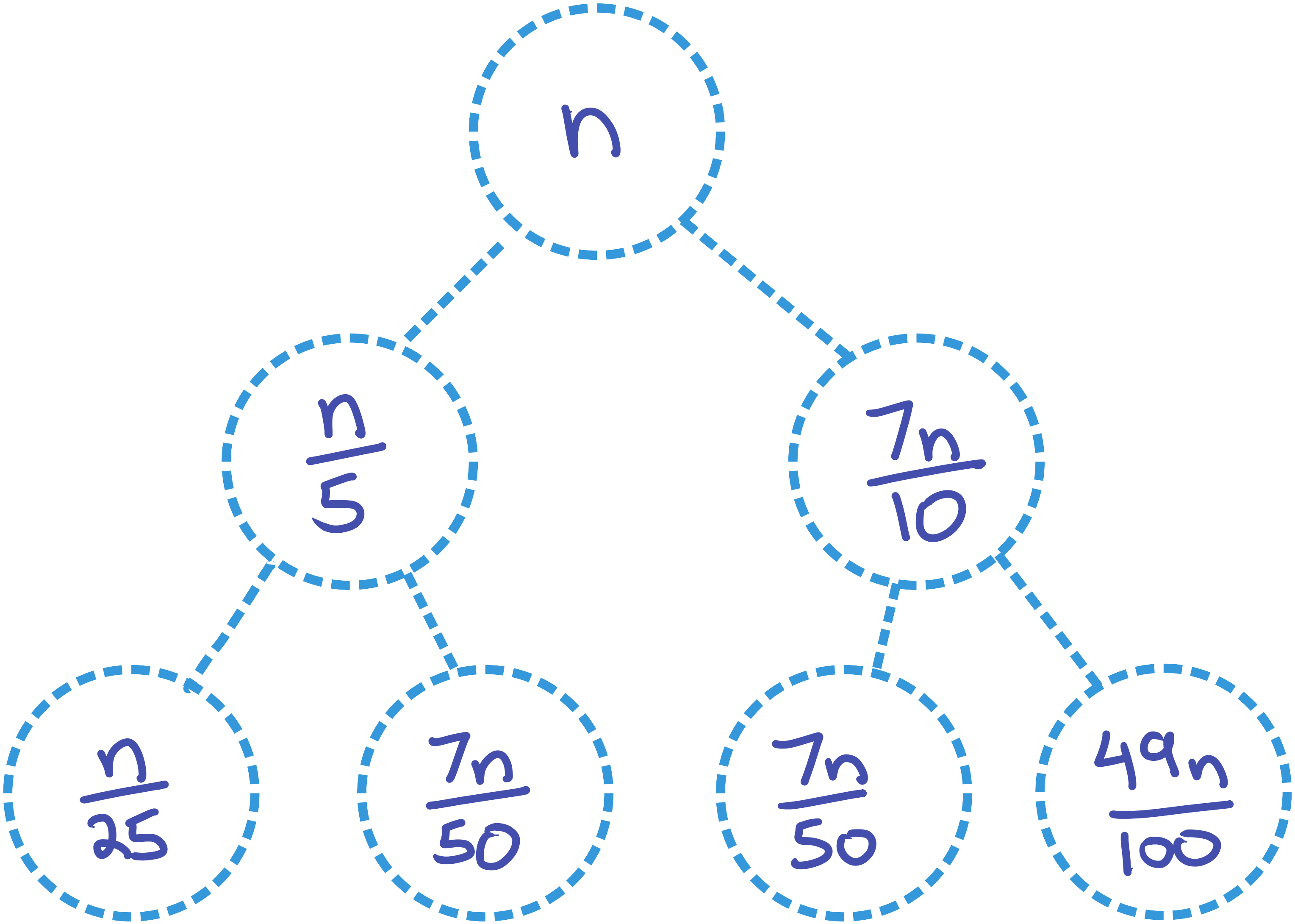
1. If $n \leq 10$ then find the kth element by brute force.
2. Let $p =$ median-of-medians(A).
3. Compare p to each element in $A[1..n]$ to identify its rank ℓ.
4. If $k = \ell$ then return p.
5. If $k < \ell$ then return select(k, B), where $B[1..\ell - 1]$ is an array of the $\ell - 1$ elements smaller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

$$T(n) = T(n/5) + T(7n/10) + O(n)$$



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

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



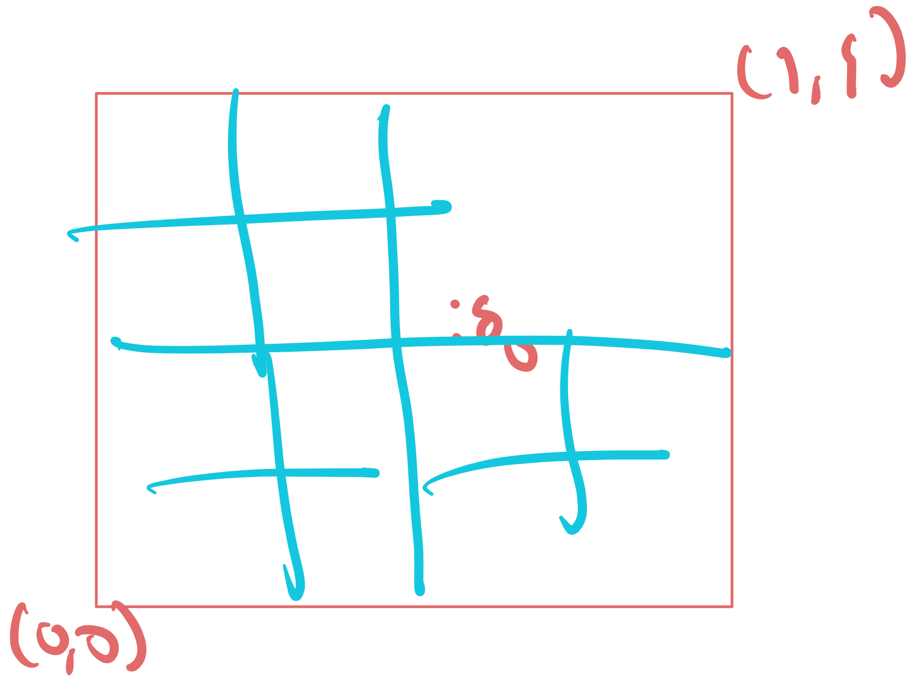
n

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

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

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

$$\sum_{i=0}^{\infty}\left(\frac{1}{5}+\frac{7}{10}\right)^{i} n = O(n)$$



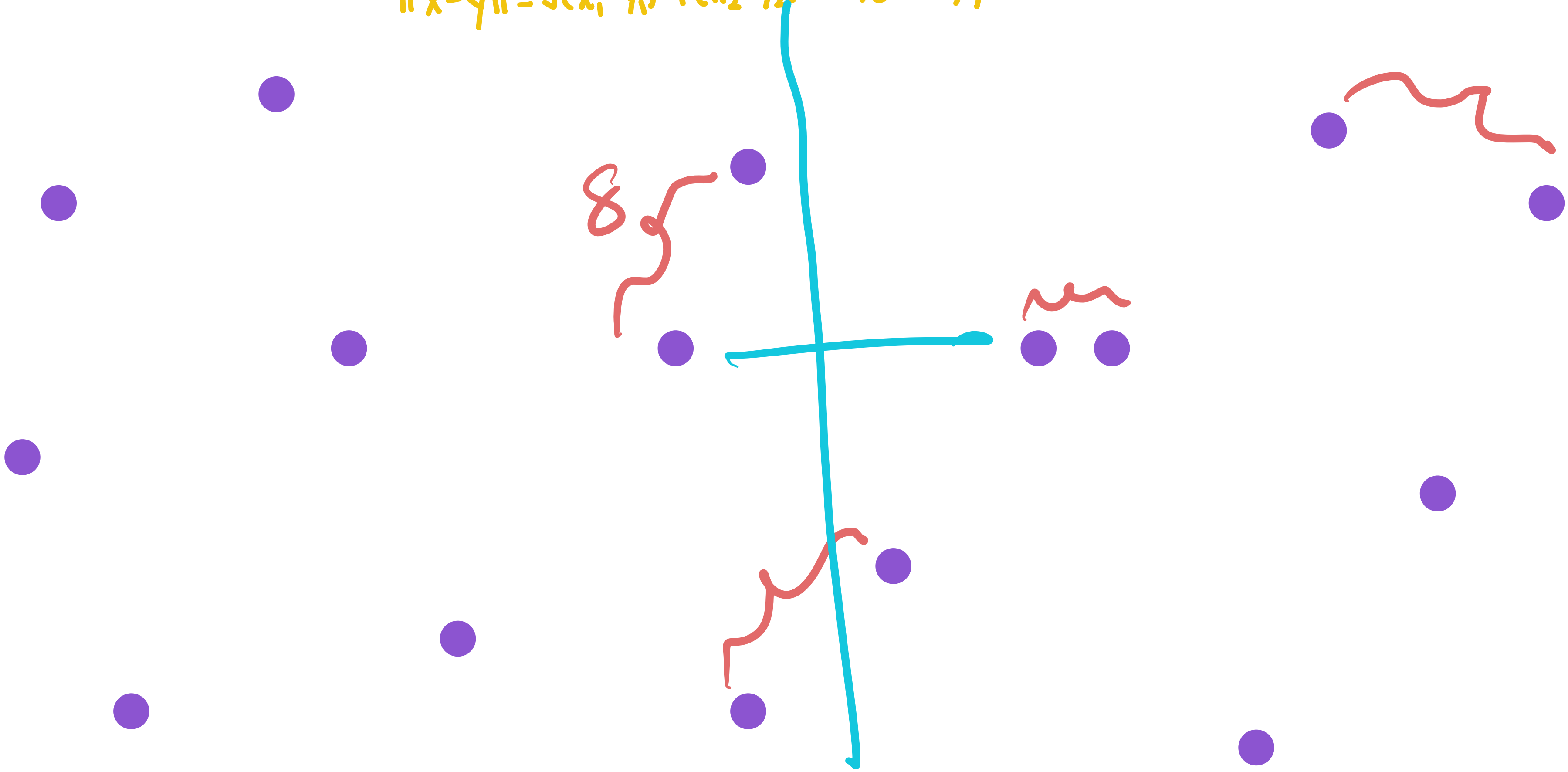
$\log(1/80)$

Minimum Euclidean distance

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

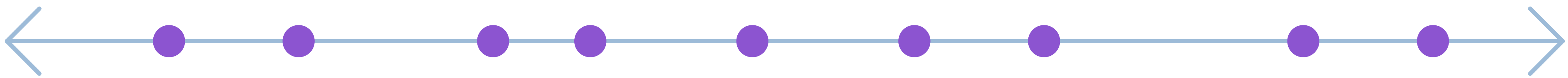
Goal: min distance over all pairs

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

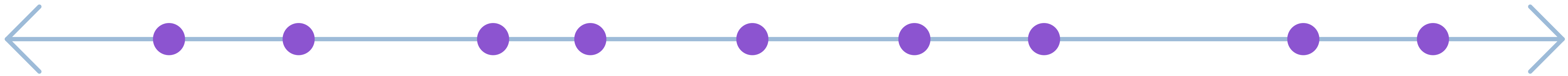


$O(n^2)$ obvious. Better?

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

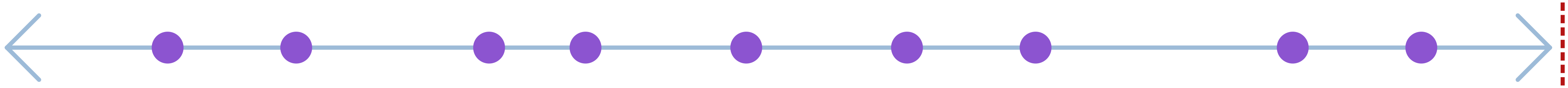


Warmup: 1 dimension (all y-coordinates = 0)



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

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

Generalizing to $\mathbb{R}^2$

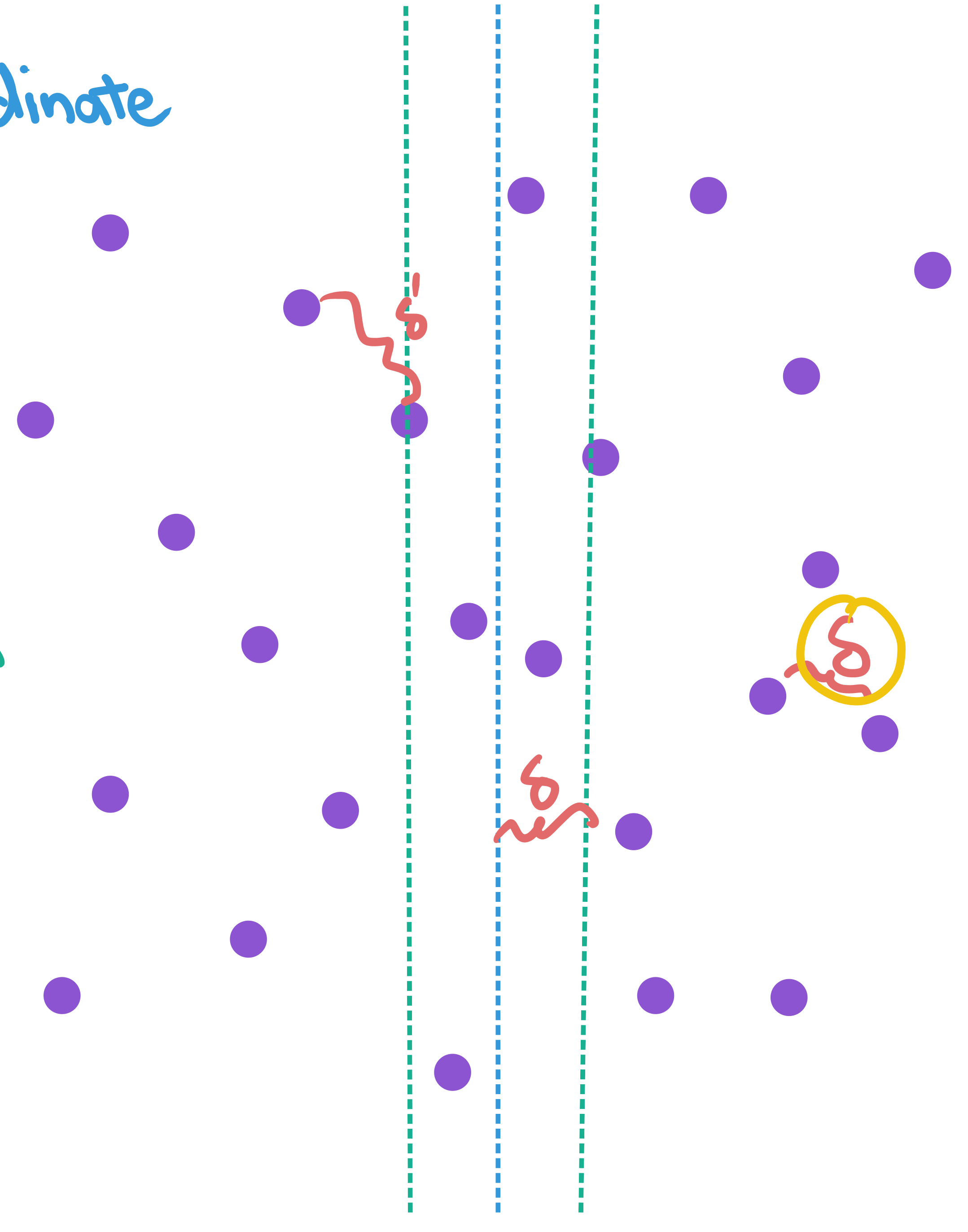
Approach 2.

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

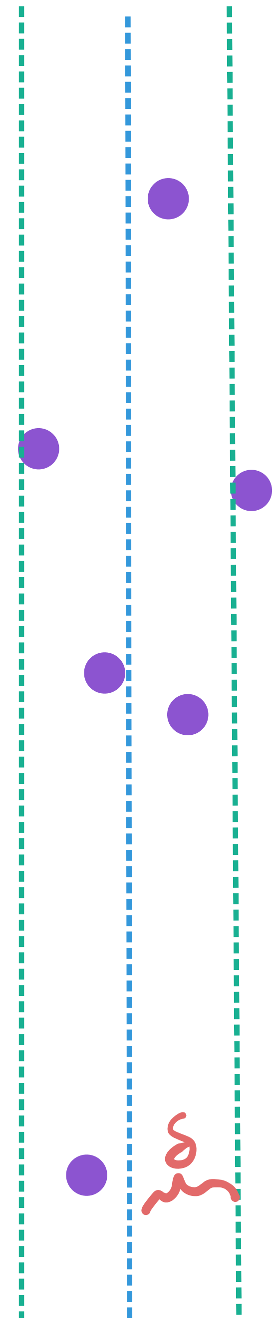


Approach 2.

Split at median, recurse

Compare w/ min pair "across" split

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

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

<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 $x = a$ */*

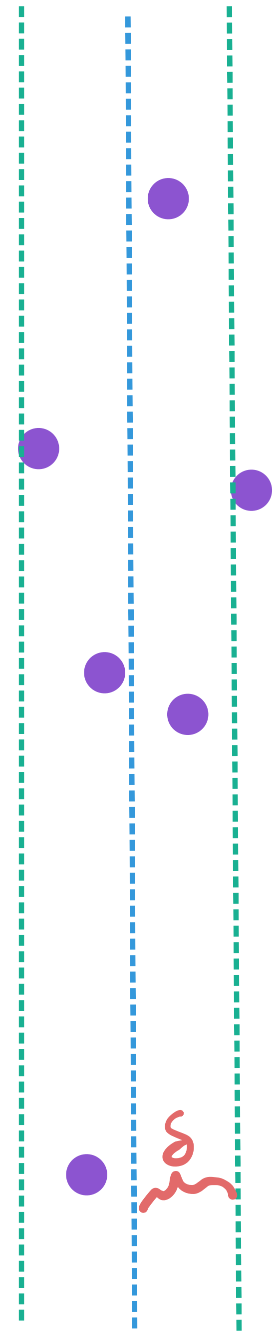
4. Let $P_1 = \{(x, y) \in P : x < a\}$ and $P_2 = \{(x, y) \in P : x \geq a\}$ *// $O(n)$*

/ 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 $\min\{\delta, \delta_3\}$.

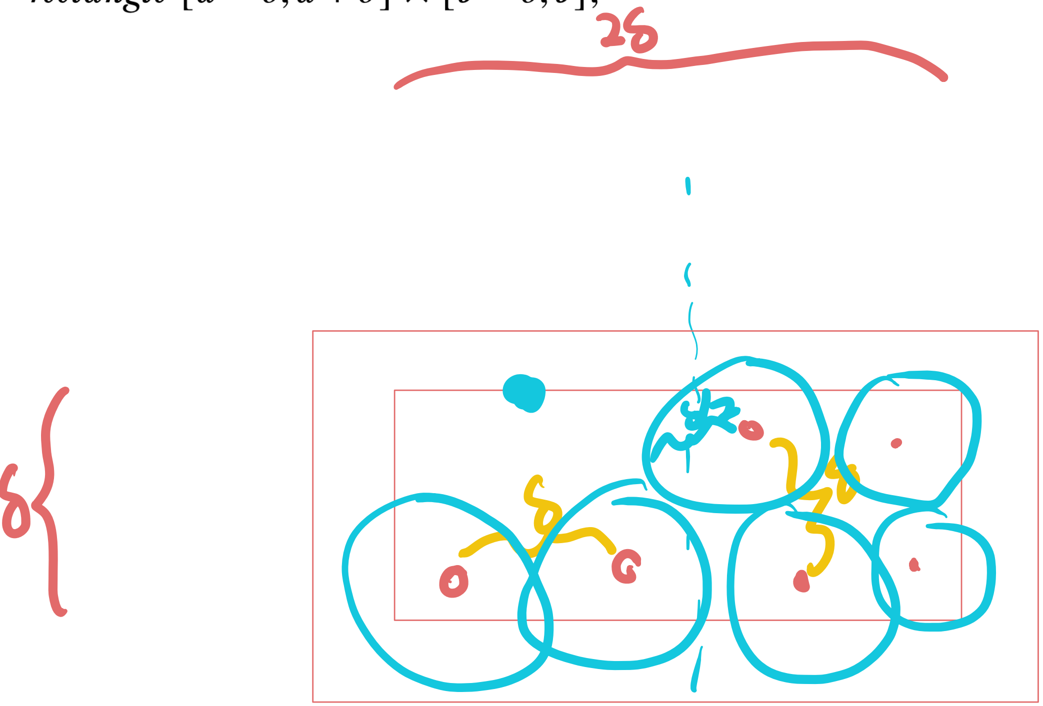


(a = median x-coordinate)

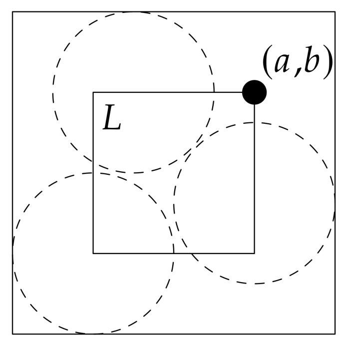
Lemma 11.3. *Let $b \in \mathbb{R}$. There are at most 10 points in P with coordinate in the rectangle $[a-\delta, a+\delta] \times [b-\delta, b]$,*

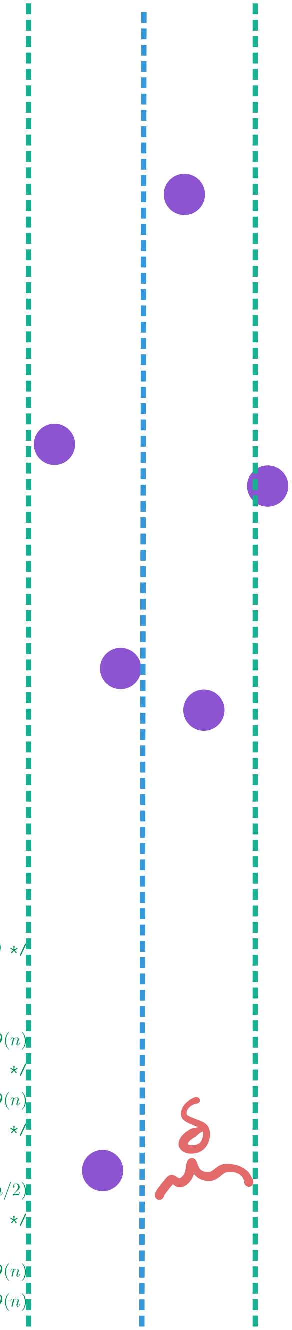
Observations
- very narrow
- points on left side have min distance $\geq \delta$
- points on right side have min distance $\geq \delta$

Intuitively: not dense, fewer comparisons needed



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





<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 $\delta = \min\{\delta_1, \delta_2\}$. // $2T(n/2)$

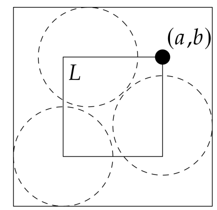
/* *It remains to find the minimum distance across P_1 and P_2.* */

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 $\min\{\delta, \delta_3\}$.

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

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



<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 $x = a$* */

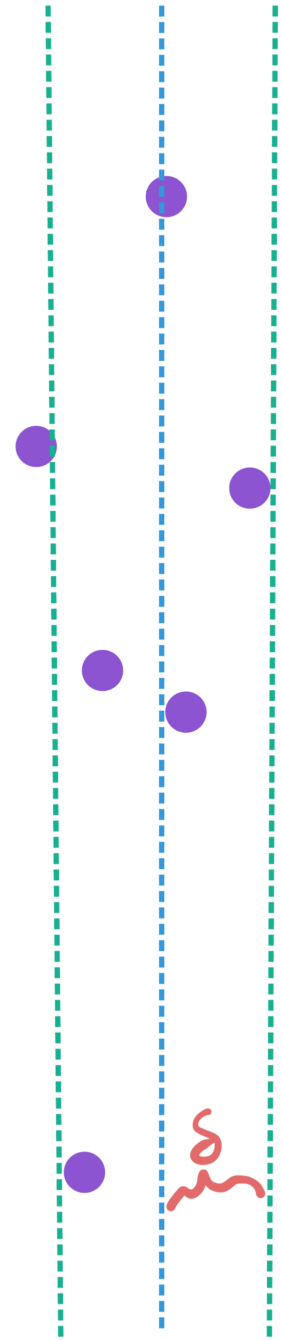
4. Let $P_1 = \{(x,y) \in P : x < a\}$ and $P_2 = \{(x,y) \in P : x \geq a\}$ // $O(n)$

/* *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 $\min\{\delta, \delta_3\}$.



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 $x = a$ */*

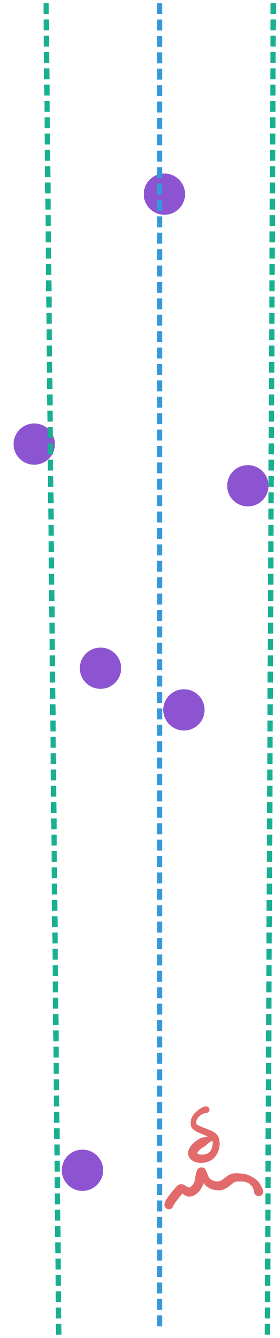
4. Let $P_1 = \{(x, y) \in P : x < a\}$ and $P_2 = \{(x, y) \in P : x \geq a\}$ // $O(n)$

/ 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 $(xy) \in \{0,1\}^{2n}$

$$\begin{array}{r} 1234 \\ \times\ 5678 \\ \hline \end{array}$$