

Selection & Closest Pair

Selection

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

Goal: k th 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 k th 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
for all n by induction on n spec

$\text{select}(k, A[1..n])$: Given an array $A[1..n]$ of comparable elements and an integer $k \in \{1, \dots, n\}$, returns the k th 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
for all n by induction on n spec

① Recursive specification

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

② Implement spec

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

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

how to use median to find k th element?

"pivot"

① find median

median



② divide input

< median

> median



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

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

② Implement spec

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

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

$\text{select}(k, A[1..n])$

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

1. If $n \leq 10$ then find the k th element by brute force.
2. Let $p = \text{median}(A)$. *// We assume a linear time subroutine for $\text{median}(A[1..n])$.*
3. If $k = \lceil n/2 \rceil$ then return p .
4. If $k < \lceil n/2 \rceil$ then return $\text{select}(k, B)$, 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 $\text{select}(k - \lceil n/2 \rceil, B)$, 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
 1
 $n/2$
 1
 $n/4$
 1
 1

work

n

$n/2$

$n/4$

$$\frac{1}{O(n)}$$

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

① Find median

median



② divide input

< median

> median



③ recurse on appropriate half



$\Rightarrow O(n)$ time

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

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

② Implement spec

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

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

Problem: don't have

$O(n)$ -time $\text{median}(A[1..n])$

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

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

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

$O(n)$ time

① find approx. median

approx. median



② divide input

< approx. median >



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

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

② Implement spec

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

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

$\text{select}(k, A[1..n])$

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

1. If $n \leq 10$ then find the k th element by brute force.
2. Let $p = \text{apx-median}(A)$. *// $O(n)$ 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 $\text{select}(k, A[1..\ell - 1])$, where $B[1..\ell - 1]$ is an array of the $\ell - 1$ elements smaller than p .
6. If $k > \ell$ then return $\text{select}(k - \ell, B)$, 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 k th element by brute force.
2. Let $p = \text{apx-median}(A)$. *// 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 .

① Find approx. median



② divide input



③ recurse on appropriate half



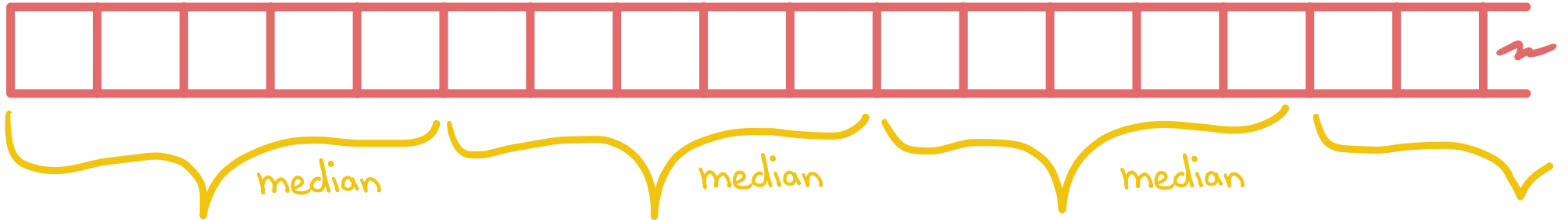
Remains to implement

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

"Median of Medians"

Blum, Floyd, Pratt,
Rivest, Tarjan

A



B

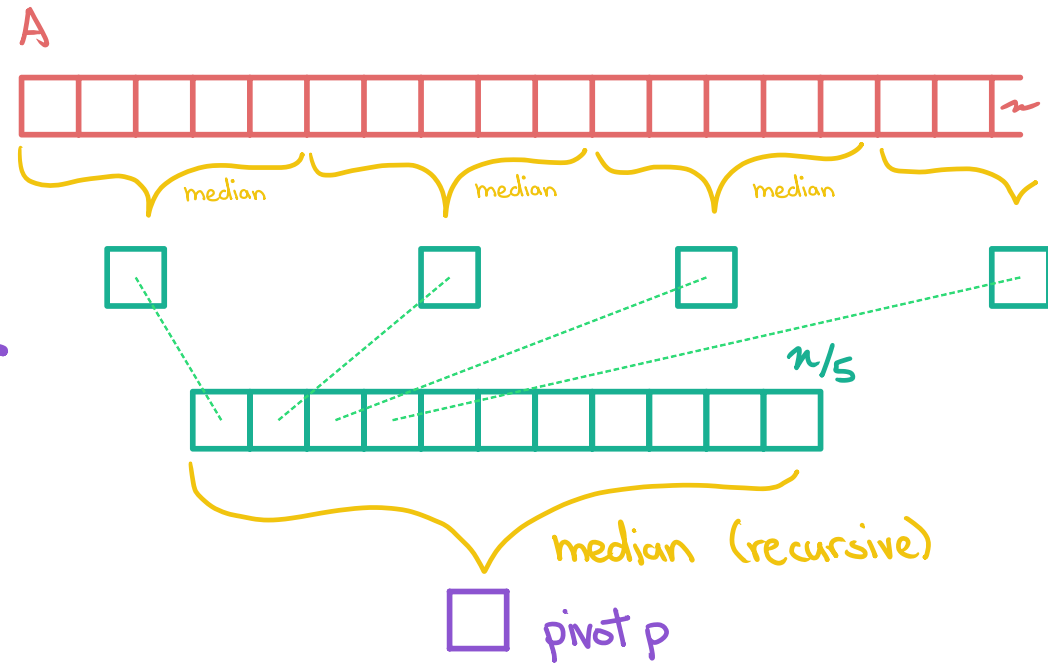


pivot p

claim:

M-of-M pivot p has rank

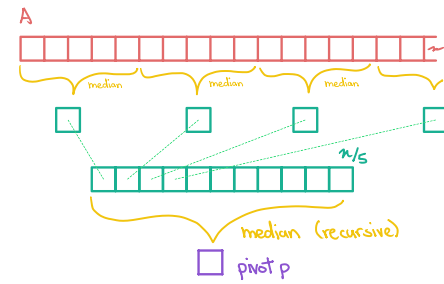
$$\frac{3n}{10} \leq \text{rank}(p) \leq \frac{7n}{10} + 5$$



claim:

M-of-M pivot p has rank

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



— B sorted —→

$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$						
$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$						

B

$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$
---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------

$n/5$

					$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$
					$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$	$\frac{n}{2}$

each column sorted



$\frac{n}{2}$ = def. smaller

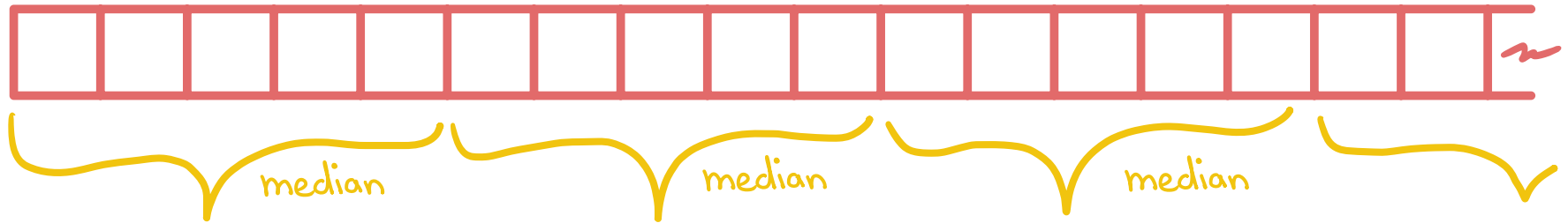
$\frac{n}{2}$ = def. bigger

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

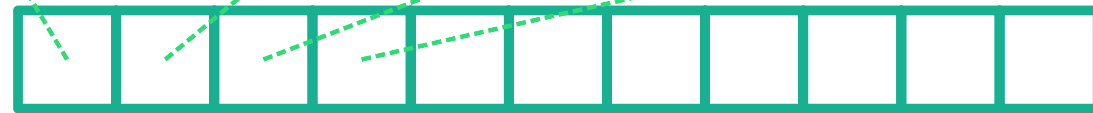
"Median of medians"

Blum, Floyd, Pratt,
Rivest, Tarjan

A

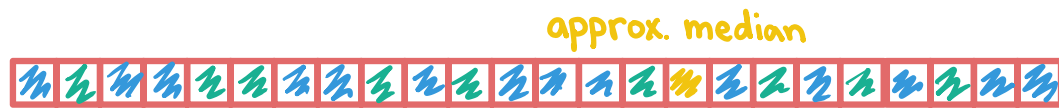


B



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

① Find approx. median



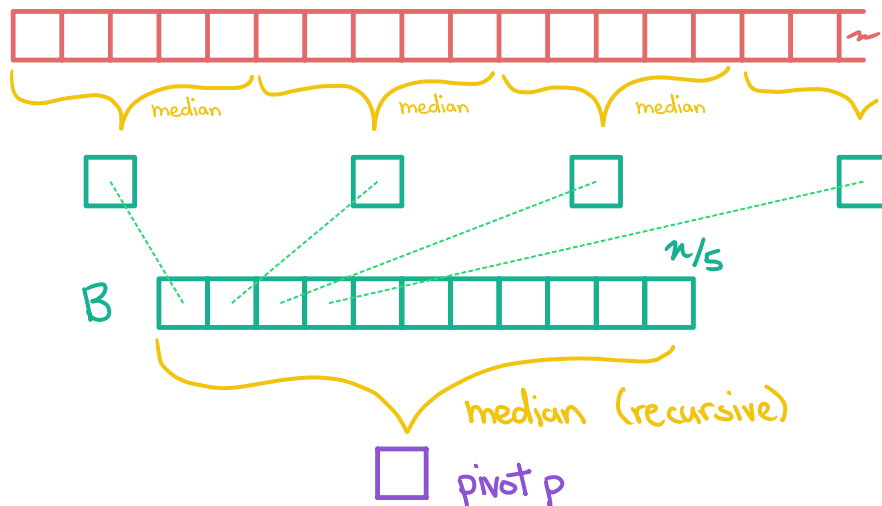
② divide input



③ recurse on appropriate half



A

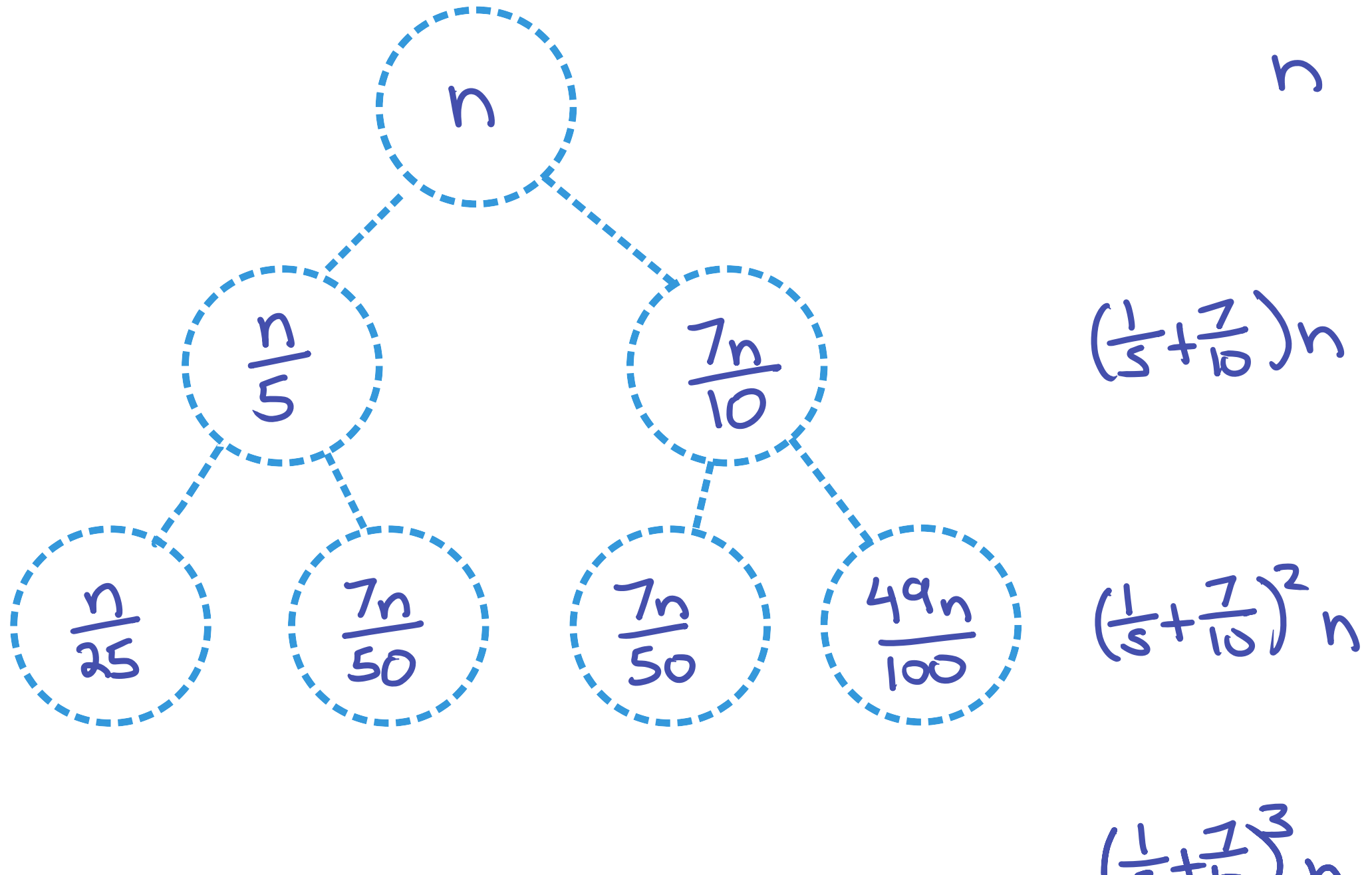


select($k, A[1..n]$)

/ select($k, A[1..n]$) = the k th 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 k th element by brute force.
2. Let $p = \text{median-of-medians}(A)$.
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 $\text{select}(k, B)$, where $B[1..\ell - 1]$ is an array of the $\ell - 1$ elements smaller than p .
6. If $k > \ell$ then return $\text{select}(k - \ell, B)$, where $B[1..n - \ell]$ is an array of the $n - \ell$ elements larger than p .

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



$\cos n \sqrt{n}$

{

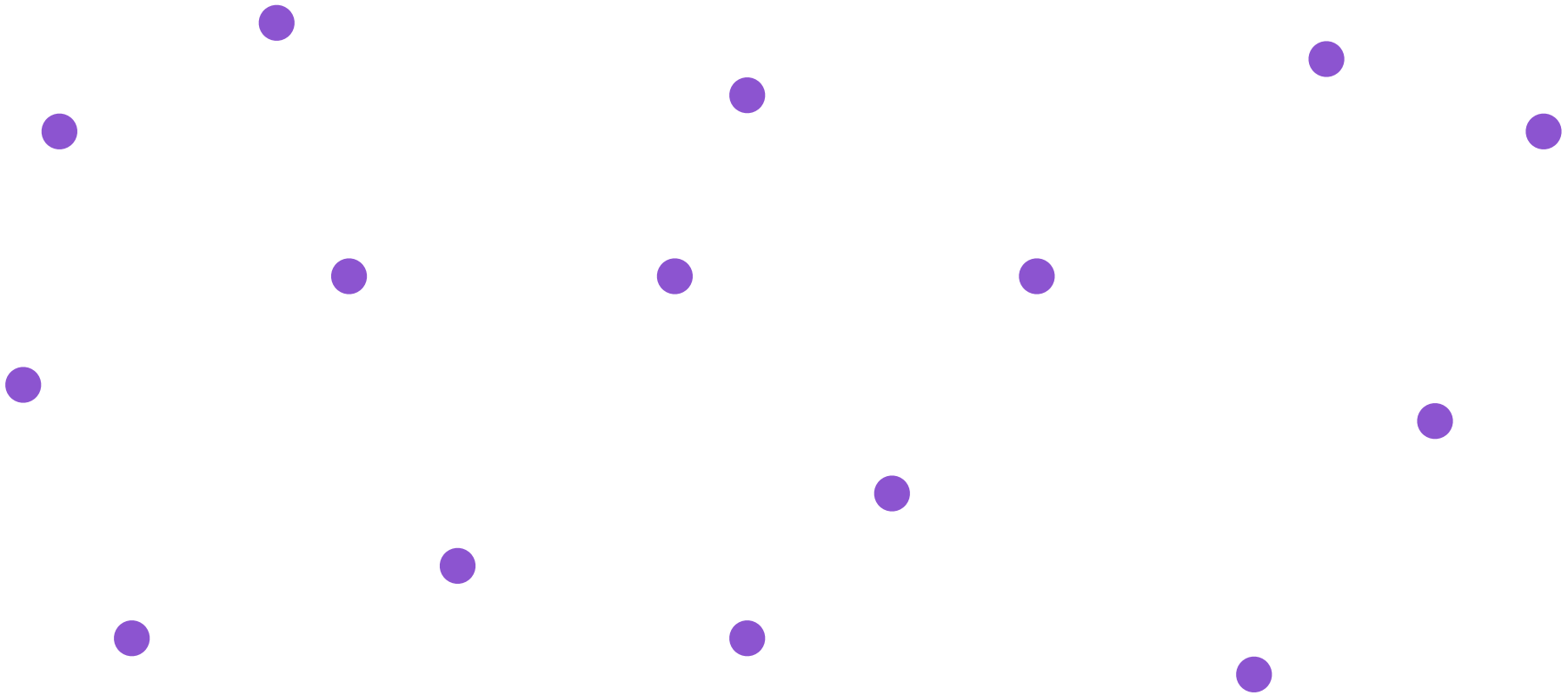
$$\sum_{i=0}^{\infty} \left(\frac{1}{5} + \frac{7}{10}\right)^i n = O(n)$$

Minimum Euclidean distance

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

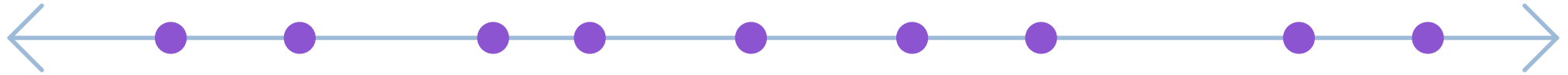
Goal: min distance over all pairs

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

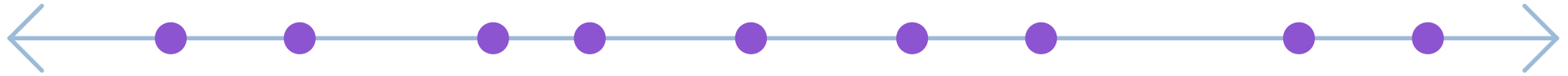


$O(n^2)$ obvious. Better?

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

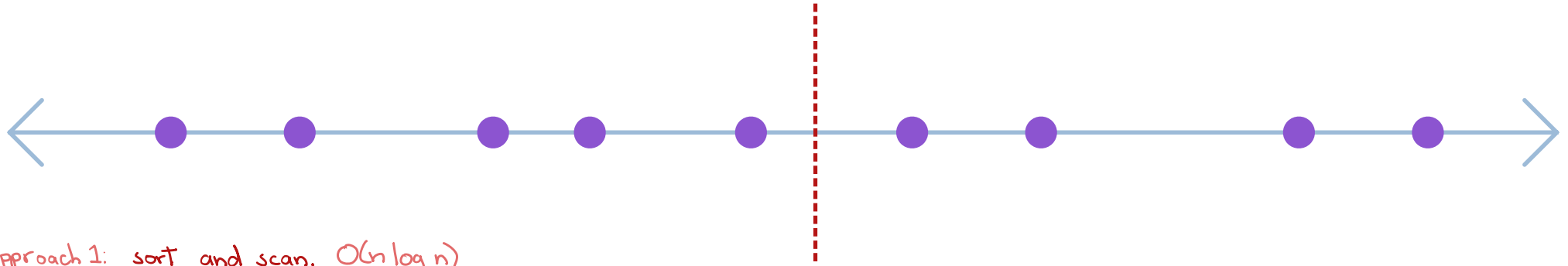


Warmup: 1 dimension (all y-coordinates = 0)



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

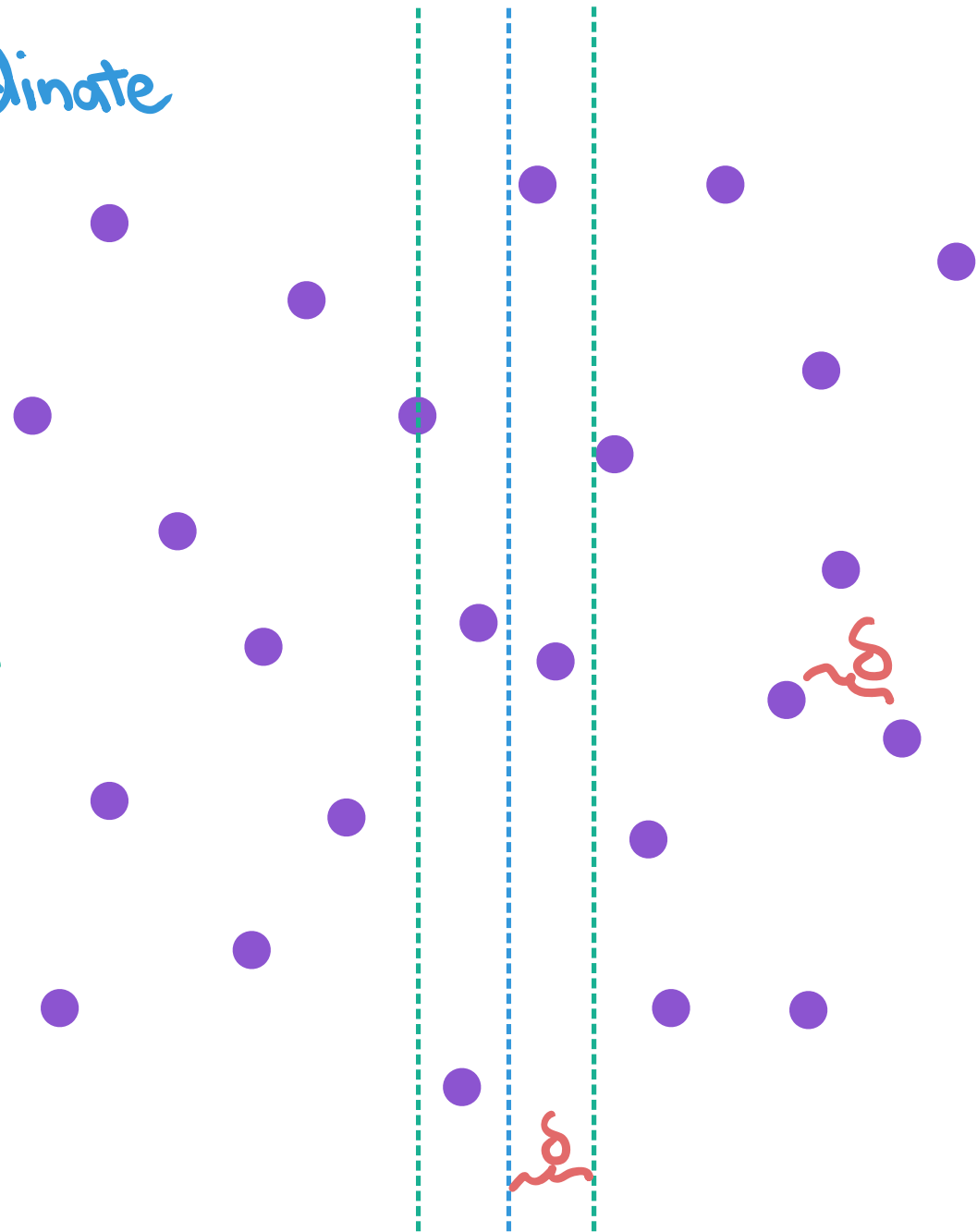
Compare w/ min pair "across" split

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



1. split at median x -coordinate

2. recurse on each half,

let δ = min dist. from two halves

Approach 2.

Split at median, recurse

Compare w/ min pair "across" split

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

4. compute min distance inside band

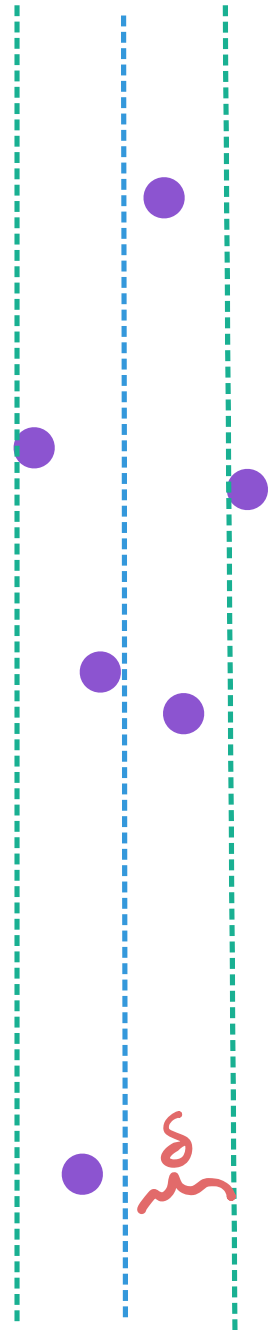
Observations

- 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

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.

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

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

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

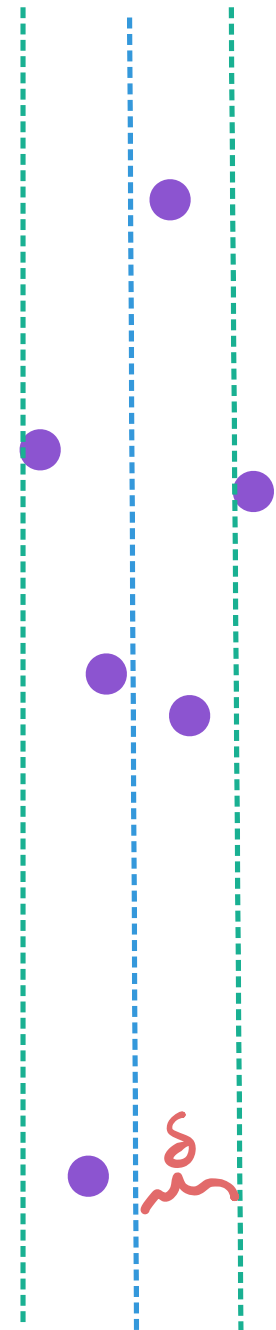
5. Let $\delta_1 = \text{min-distance}(P_1)$, $\delta_2 = \text{min-distance}(P_2)$, and $\delta = \min\{\delta_1, \delta_2\}$.

/ 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 $p_1, \dots, p_k$ list P_3 in decreasing order of y -coordinate.

7. Let δ_3 be the minimum of $\|p_i - p_j\|$ over all i, j with $i < j < i + 10$.

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



(a = median x-coordinate)

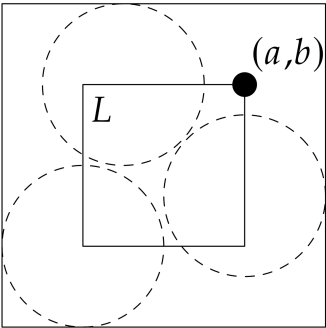
Lemma 11.3. *Let $b \in \mathbb{R}$. There are at most 10 points in P with coordinate in the rectangle $[a - \delta, a + \delta] \times [b - \delta, b]$,*

Observations

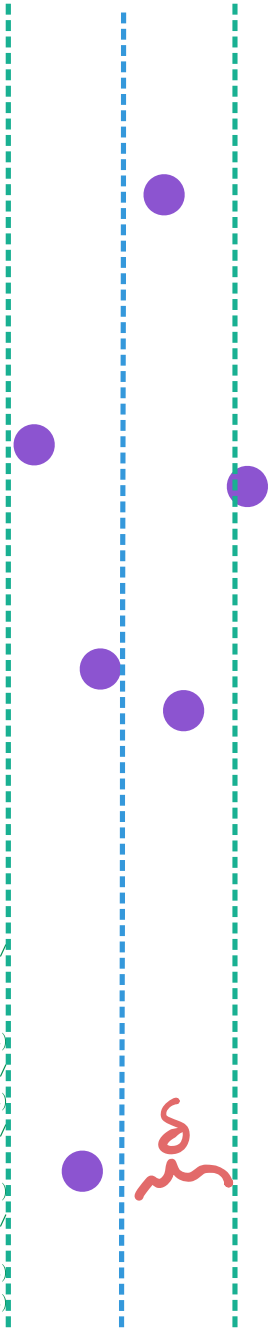
- very narrow
- points on left side have min distance $\geq \delta$
- points on right side have min distance $\geq \delta$

Intuitively: not dense, fewer comparisons needed

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



```
min-distance(P)
/* We assume P is sorted by y-coordinate. (Otherwise sort P once in a preprocessing step.) */
1. If n ≤ 1 then return +∞.
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 P1 = {(x,y) ∈ P : x < a} and P2 = {(x,y) ∈ P : x ≥ a} // O(n)
/* Find the minimum distances within each half P1 and P2 recursively. */
5. Let δ1 = min-distance(P1), δ2 = min-distance(P2), and δ = min{δ1, δ2}. // 2T(n/2)
/* It remains to find the minimum distance across P1 and P2. */
6. Let P3 = {(x,y) ∈ P : |x - a| < δ} and let p1, ..., pk list P3 in decreasing order of y-coordinate. // O(n)
7. Let δ3 be the minimum of ||pi - pj|| over all i, j with i < j < i + 10. // O(n)
8. Return min{δ, δ3}.
```

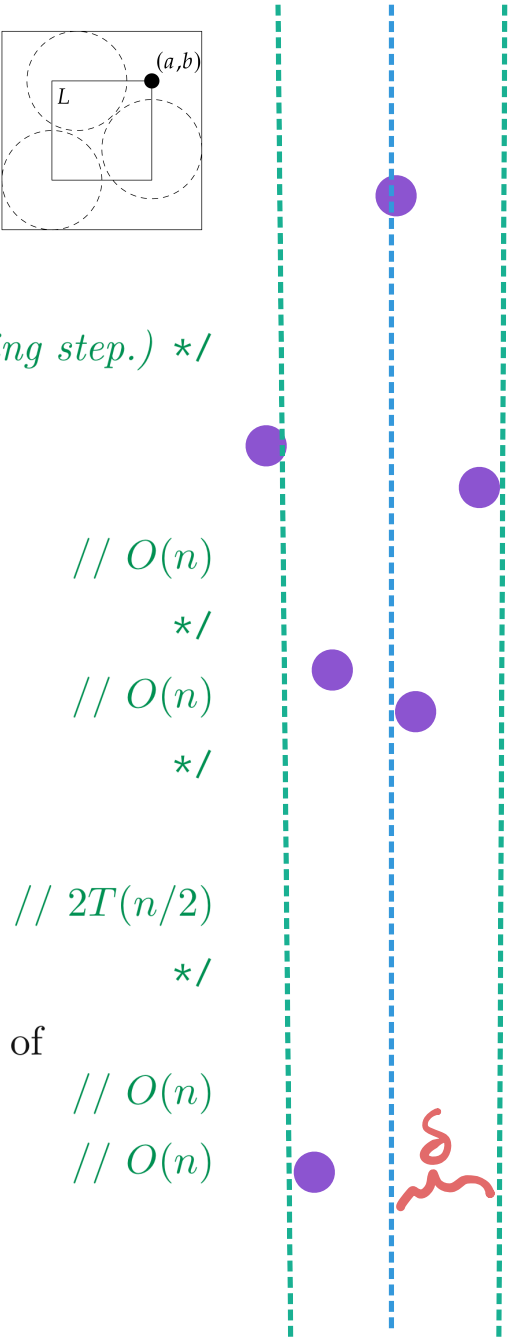


Lemma 11.3. Let $b \in \mathbb{R}$. There are at most 10 points in P with coordinate in the rectangle $[a - \delta, a + \delta] \times [b - \delta, b]$,

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 $\delta_1 = \text{min-distance}(P_1)$, $\delta_2 = \text{min-distance}(P_2)$, 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 $p_1, \dots, p_k$ 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

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 $\delta_1 = \text{min-distance}(P_1)$, $\delta_2 = \text{min-distance}(P_2)$, 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 $p_1, \dots, p_k$ list P_3 in decreasing order of y -coordinate.

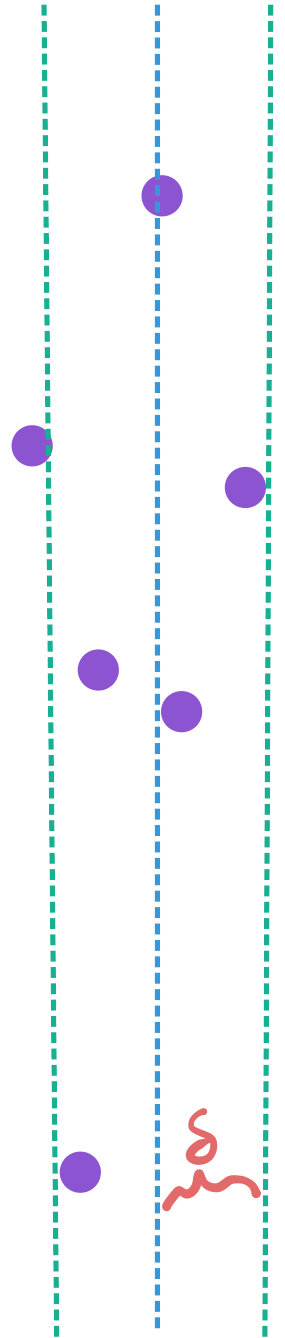
// $O(n)$

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

$$\begin{array}{r} 1234 \\ \times 5678 \\ \hline 9872 \\ 8638 \\ + \quad \swarrow \\ \hline \end{array}$$

$O(n^2)$ time

Better?