CS4800: Algorithms & Data
Jonathan Ullman

Lecture 6:

- Divide-and-Conquer: Inversions, Closest Pair
- Dynamic Programming Warmup (?)

Jan 26, 2018

Counting Inversions

Approximate Sortedness

- Which of these is “more sorted”?

8	3	28	11	17	42

11	3	8	17	28	42

- only one elem out of place
- 5/6 of elems are sorted

Counting Inversions

2	1	3	4

vs.

4	1	2	3

- A Measure of Sortedness: Number of Inversions
  - Inversion: a pair $i < j$ such that $A[i] > A[j]$ (a sorted list has no inversions)
  - "Kendall tau distance" / "Bubble sort distance"

8	3	28	11	17	42
1	2	3	4	5	6

Inversions:

(1,2)
(3,4)
(3,5)

$\boxed{3}$

11	3	8	17	28	42
1	2	3	4	5	6

Inversions:

(1,2)
(1,3)

$\boxed{2}$

Counting Inversions

- A Measure of Sortedness: Number of Inversions
  - Inversion: a pair $i < j$ such that $A[i] > A[j]$
  - “Kendall tau distance”

8	3	28	11	17	42

11	3	8	17	28	42

- Many Applications
  - Collaborative filtering / recommender systems
  - Social choice theory / voting theory

Counting Inversions: Basic Algorithm

8	3	28	17	11	42	2	35

For all i, j s.t. $i < j$:

check if $A[j] > A[i]$

Output # of inversions

# of possible pairs

$$\frac{n(n+1)}{2} = \Theta(n^2)$$

Counting Inversions: D&C

Idea: Sort and Count at the same time!

L, $c_L = 2$ R, $c_R = 3$

8	3	28	17	11	42	2	35

A

Step 1: Cut list in half

Step 2: $c_L \leftarrow$ CountInversions(L) $c_R \leftarrow$ CountInversions(R)

Three types of inversions:

1. $1 \le i < j \le \frac{n}{2}$ ("on the left") $\boxed{c_L}$
2. $\frac{n}{2}+1 \le i < j \le n$ ("on the right") $\boxed{c_R}$
3. $1 \le i \le \frac{n}{2}$ $\frac{n}{2}+1 \le j \le n$ ("in the middle") $\boxed{c_M}$

$T(n) = \Omega(n^2)$

compare $\frac{n}{2} \cdot \frac{n}{2}$ items $\frac{n^2}{4} = \Omega(n^2)$

Step 3: Try all inversions of type 3 and count them

Step 4: return $c_L + c_R + c_M$

Counting Inversions: D&C

L, $c_L = 2$ R, $c_R = 3$

8	3	28	17	11	42	2	35	A

↑ smallest elem on the right

Goal: Count inversions i, j s.t. $i \in L$, $j \in R$.

$\Updownarrow$

# of times an elem of R is smaller than an elem of L.

~~3~~	~~8~~	~~17~~	~~28~~	L_{sorted}

~~2~~	~~11~~	35	42	R_{sorted}

4	0	0	2	0	0	0	0	$c_M = 6$
2	3	8	11	17	28	35	42	

When we moved 2 into place, it was smaller than all elems in L that had not been placed.

Note: Be careful with ties.

Counting Inversions: D&C

8	3	28	17	11	42	2	35

Claim: Given two sorted lists (L, R) w/ # of inversions (c_L, c_R), we can produce one sorted list A and compute c_M in $O(n)$ time.

3	8	17	28

2	11	35	42

Note: Be careful with ties. Make sure to add L if there is a tie.

Counting Inversions: D&C

MergeAndCount$(L[1 \dots \ell], R[1, \dots, r])$:

  Let $i, j, k \leftarrow 1$, $c \leftarrow 0$

  For $k = 1, \dots, \ell + r$ // Loop over elts

    If $i > \ell$: // L is empty

      $A[k] = R[j]$, $j \leftarrow j + 1$

    Elif $j > r$: // R is empty

      $A[k] = L[i]$, $i \leftarrow i + 1$

    Elif $L[i] < R[j]$: // L is smaller

      $A[k] = L[i]$, $i \leftarrow i + 1$

    Else: // R is smaller

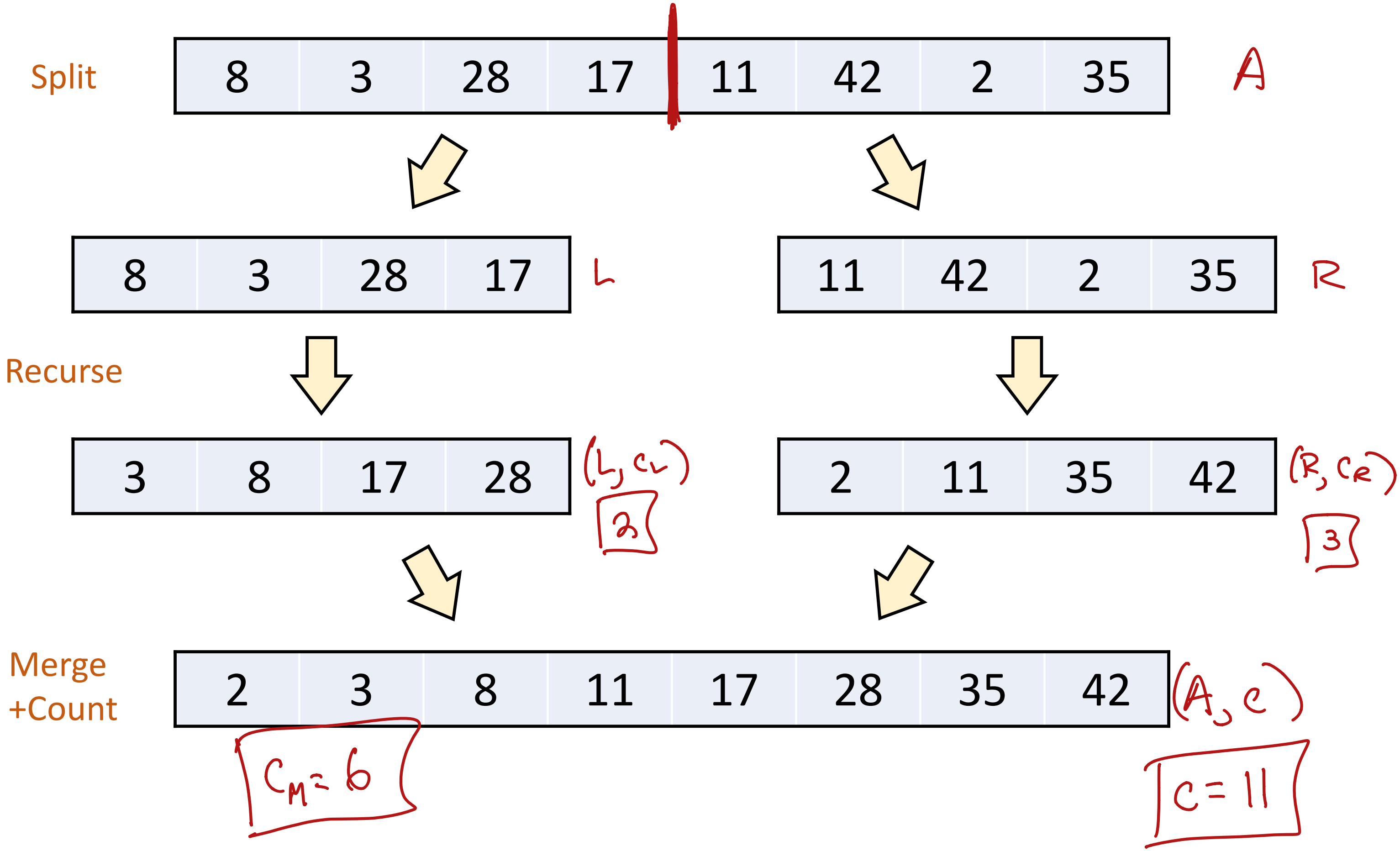
      $A[k] = R[j]$, $j \leftarrow j + 1$,

      $c \leftarrow c + (\ell - i + 1)$ // Add inversions

  Return (A, c)

time
$O(1) \times$
$O(\ell + r)$
$= O(\ell + r)$

Counting Inversions: D&C



Counting Inversions: D&C

SortAndCount($A[1, \dots, n]$):

If $n = 1$, return $(A, 0)$ // Base case

$O(1)$

$\ell \leftarrow \left\lceil \frac{n}{2} \right\rceil$ // Split into two lists

$L \leftarrow A[1, \dots, \ell], R \leftarrow A[\ell + 1, n]$

$(L, c_L) \leftarrow \text{SortAndCount}(L)$ → $T(\frac{n}{2})$ // Recurse

$(R, c_R) \leftarrow \text{SortAndCount}(R)$ → $T(\frac{n}{2})$ // Recurse

$(A, c_M) \leftarrow \text{MergeAndCount}(L, R)$ → $O(n)$ // Merge

return $(A, c_L + c_R + c_M)$

$$T(n) = 2 \times T(\tfrac{n}{2}) + O(n) \qquad T(n) = \Theta(n \log n)$$

Ask the Audience!

- Suppose I wanted to output a list of all inversions (i.e. all pairs $i < j$ such that $A[i] > A[j]$), can our D&C algorithm be modified to output a list of all inversions in $O(n \log n)$ time?

No. There may actually be $\Omega(n^2)$ inversions so we cannot possibly write them down.

Counting Inversions: Fun Fact

- Improved to $O\left(n\sqrt{\log n}\right)$ in 2010
  - It is "not possible" to sort in less than $n \log n$ time
  - Counting inversions is easier than sorting

Closest Pair

Closest Pair

- General Problem: Given points $p_1, \ldots, p_n$, find the pair p_i, p_j minimizing $d(p_i, p_j)$
  - Foundational problem in computational geometry
  - Closely related to nearest-neighbor search, classification, and clustering problems

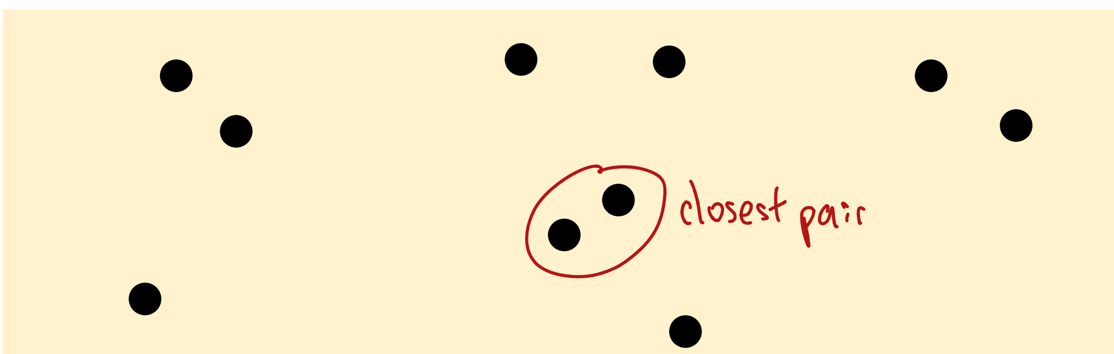
Closest Pair in 2D

Naïve Algorithm: $\Theta(n^2)$ time
(try all pairs)

- Special Case: Closest Pair in 2D
- Given $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n) \in \mathbb{R}^2$ find the pair p_i, p_j minimizing the Euclidean distance

Assume you can obtain exact distances in $O(1)$ time

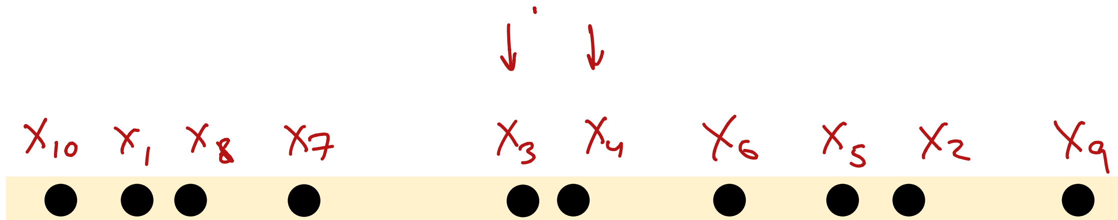
$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$



Assumption: all the x and y coordinates distinct

Ask the Audience!

- Even More Special Case: Closest Pair in 1D
- Given $x_1, \ldots, x_n \in \mathbb{R}$ find the pair x_i, x_j minimizing the distance $|x_i - x_j|$
- Find an $O(n \log n)$ time algorithm



Alg I: • Sort the points

• Consider all pairs A[i], A[i+1] in the sorted list

$O(n \log n)$

n-1 choices

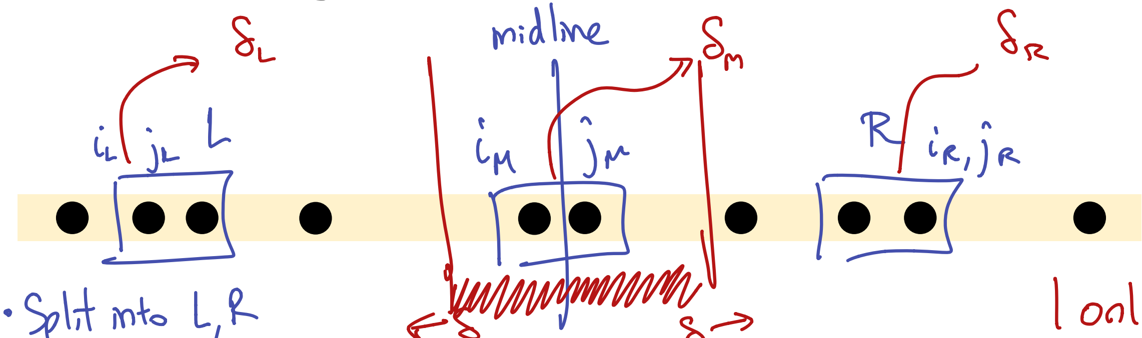
(Fact: Closest pair i,j is in consecutive places in the sorted order.)

Ask the Audience!

- Even More Special Case: Closest Pair in 1D
- Given $x_1, \dots, x_n \in \mathbb{R}$ find the pair x_i, x_j minimizing the distance $|x_i - x_j|$

Let $\delta = \min\{\delta_L, \delta_R\}$

- Find an $O(n \log n)$ time algorithm



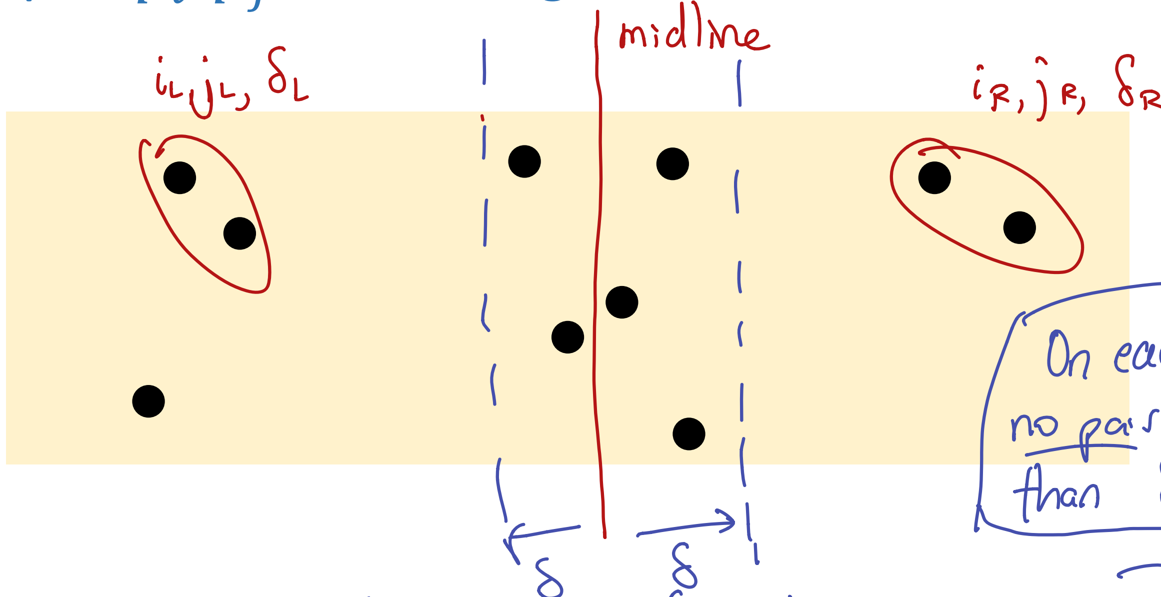
Alg II:
- Split into L, R
- Find (i_L, j_L), (i_R, j_R) recursively
- Consider rightmost $\hat{i}_M$ in L, leftmost point $\hat{j}_M$ in R
- Choose the best of δ_L, δ_R

I only need to consider $\hat{i}_M, \hat{j}_M$ at distance $\leq \delta$ from midline.

Closest Pair in 2D

$\delta = \min\{\delta_L, \delta_R$

- Given $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n) \in \mathbb{R}^2$ find the pair p_i, p_j minimizing the Euclidean distance

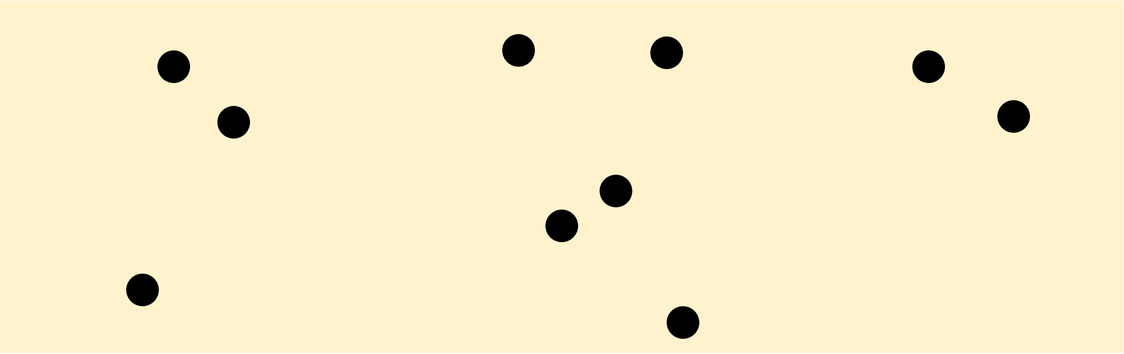


On each side, no pair is close than δ.

The only candidates I consider for closest pair are in the band around the midline.] Key Idea

Closest Pair in 2D

- Given $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n) \in \mathbb{R}^2$ find the pair p_i, p_j minimizing the Euclidean distance



- Key Idea: Only need to consider a band of width δ on each side around the midline

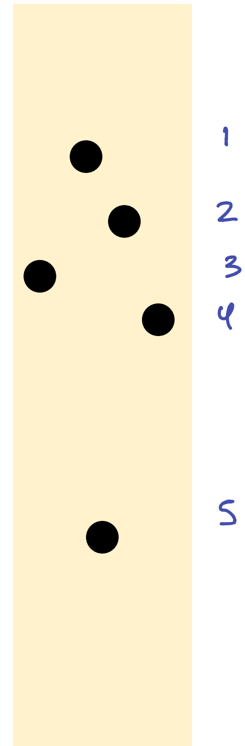
Closest Pair in 2D

1D: Travis Barker Mohawk

• Sort by y-coord and look at adjacent pairs.

2D: Mr. T Mohawk

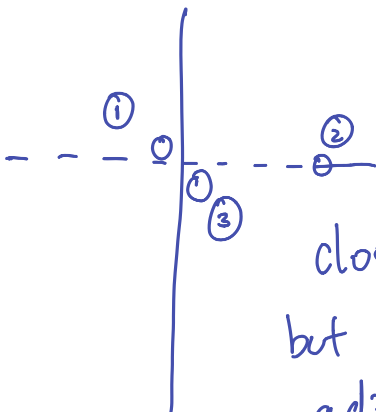
Idea: Sort by y, look at adjacent pairs.



Ask the Audience!

I pity the fool who gets this wrong!

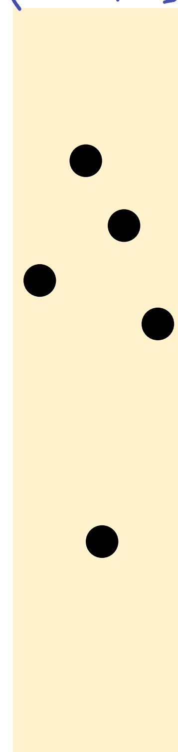
- Suppose I look at the points in the mohawk and sort by the y-coord.
- True/False: The closest pair in the mohawk is **adjacent** in the sorted list?



closest pair is 1, 3

but those are not adjacent

$\delta \leftarrow \quad \rightarrow \delta$



No pair in the left/right half is closer than δ

Closest Pair in 2D

Very cool, key claim

- Claim: Let $(x_1, y_1), \ldots, (x_m, y_m)$ be the points in the mohawk, sorted by y-coordinate and let p_k, p_ℓ be the closest pair, then $|k - \ell| \leq 15$

There are $\leq 15m$ such pairs where $m \leq n$ is the number of points in the mohawk.

Closest Pair in 2D

Sort all points by x, S_x
Sort all points by y, S_y

Closest(P, SX, SY)

Let q be the middle-element of SX

Divide P into Left, Right according to q. Scan to get LY, RY.

Split by x-coord

delta,r,j = MIN($\underbrace{\text{Closest(Left, LX, LY)}}_{i_L, j_L, \delta_L}$ $\underbrace{\text{Closest(Right, RX, RY)}}_{i_R, j_R, \delta_R}$)

$\delta = \min\{\delta_L, \delta_R\}$

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

Shave the mohawk
Sorted by y coord

For each point x in Mohawk (in order):

Compute distance to its next 15 neighbors

only check ≤ 15 other points

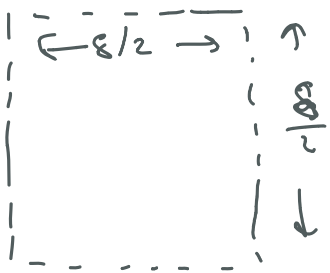
Update delta,r,j if any pair (x,y) is < delta

Return (delta,r,j)

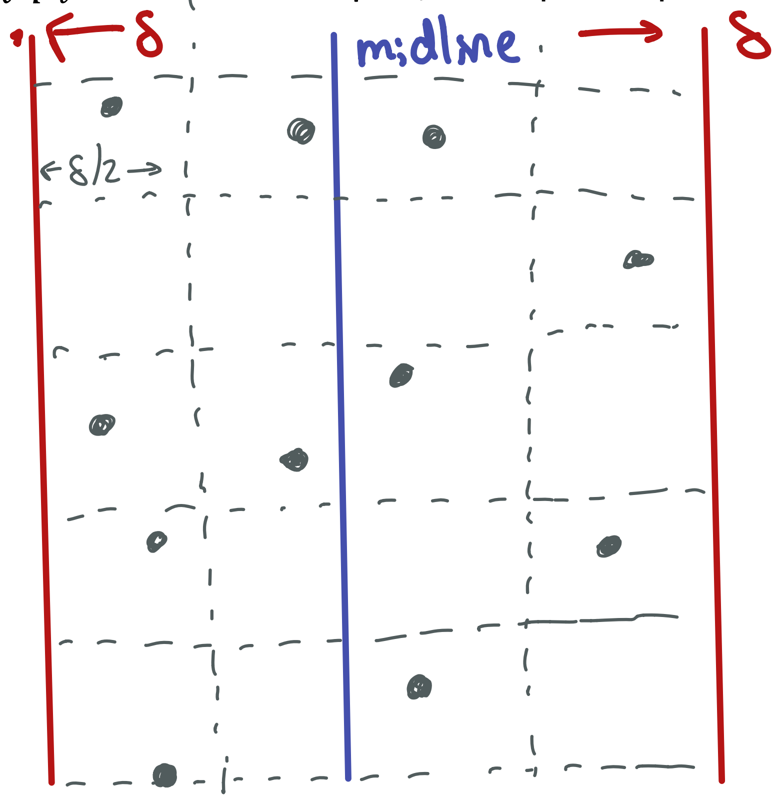
Best among (i_L, j_L), (i_R, j_R), Mohawk.

Closest Pair in 2D

- Claim: Let $(x_1, y_1), \ldots, (x_m, y_m)$ be the points in the mohawk, sorted by y-coordinate and let p_k, p_ℓ be the closest pair, then $|k - \ell| \leq 15$



$\delta/2 \times \delta/2$ cubby holes

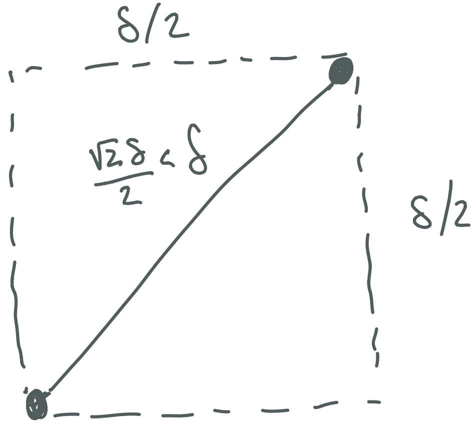


Closest Pair in 2D

- Claim: Let $(x_1, y_1), \ldots, (x_m, y_m)$ be the points in the mohawk, sorted by y-coordinate and let p_k, p_ℓ be the closest pair, then $|k - \ell| \leq 15$

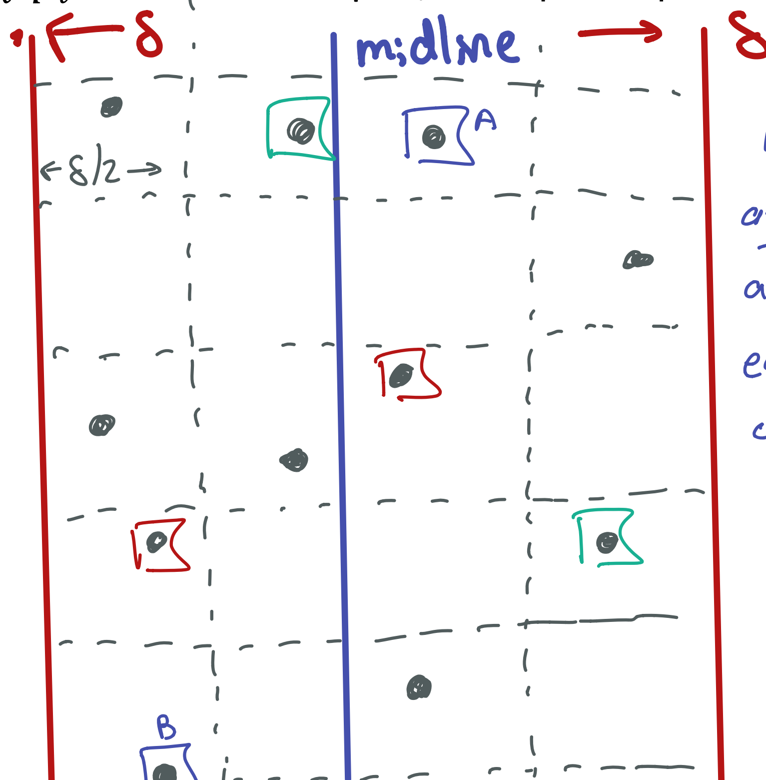
Fact ①: At most one point in each cubby hole

- Because no pair on the same side of the midline have distance $< \delta$.



Closest Pair in 2D

- Claim: Let $(x_1, y_1), \dots, (x_m, y_m)$ be the points in the mohawk, sorted by y-coordinate and let p_k, p_ℓ be the closest pair, then $|k - \ell| \le 15$



Fact ②: Any pair closer than δ can only be separated by 2 rows of cubbies

A, B are at least $\frac{3\delta}{2}$ apart because each row of cubbies is δ/2.

Fact ③: If A, B are closest, then only the 15 points in cubbies in the rows btw can be btw them in sorted order.