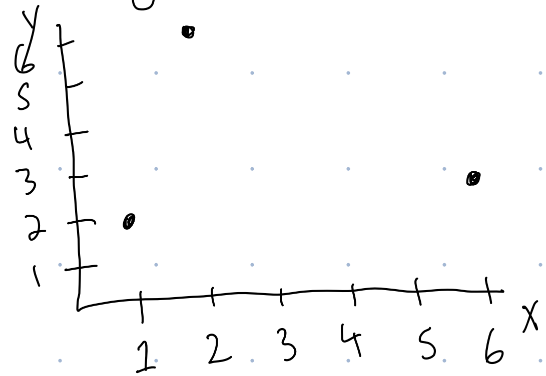
Closest Points Problem

Input: Array of 2-D points: P = | (1, 2) | (2, 6) | (6, 3) | ...



Output: Distance b/t 2 closest points

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

Assume:

x, y coordinats are unique for each pt

~~| (1, 2) | (1, 6) |~~

| (1, 2) | (2, 1) |

Applications:
- air traffic control
- robotics
- stereo imaging

Algorithms + Ethics

Algorithm is essentially a mathematical object.

But once it gets implemented for a particular task, has ethical implications

Ethical Matrix (O'Neil + Gunn)

Where travel + why?

Air Traffic Control Improvement

Harm? Benefit?

Choice to use? Are users informed enough to understand meaningfully take responsibility for use?

Unfair treatment of different groups?

Stakeholders	Well-Being	Autonomy	Justice
Airplane Passengers	Safety, Cheaper (?) Environment	A - transport Zoom	Equal access (rich/poor) (geographic)
Airline Cos	$, safer	No choice (international)	Difference by country, by region
Employees, ATC	Automation → No job Efficiency, less stress	No choice safer	Who is to blame if error
Civilians, Bystanders	Noise pollution, Make war easier, environment, help economy/tourists	No choice, opt out (vote)	(Un) Equal impact on environment, near airport more affected

Army			
People of color			
Non-binary folks			

Ethical matrix does not tell you what to do!

Tool for thinking about consequences, both + and −.

→ How can I mitigate negatives? → Modify → Aid

→

Closest Points 2D

Before designing a sophisticated algorithm, try to benchmark
- Want better than "Brute Force"
- Can't do better than 1D

Brute Force (check every pair)

- min ← ∞ $O(1)$
- for i ← 1 to n-1 $O(n)$ } $O(n^2)$
  - for j ← i+1 to n $O(n)$
    - if dist(p_i, p_j) < min, then min ← dist(p_i, p_j) $O(1)$
- return min

$O(n^2)$

Closest Pts 1D

5	9	0	20	17

- Sort(P) $O(n\log n)$ +
- min ← ∞ $O(1)$
- for i = 1 to n-1: $O(n)$
  - if dist(p_i, p_{i+1}) < min, then min ← dist(p_i, p_{i+1}) $O(1)$ $O(1)$
- return min

0	5	9	17	20

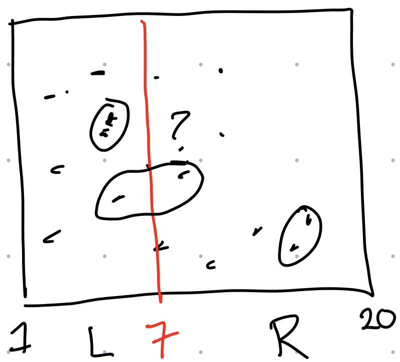
- Sort(P) $O(n\log n)$ n
- $O(n)$ } $O(n\log n)$

Ethical Matrix: Race? Include the right stakeholders / how many?
Weakness? When to use? What to do after fill out?

CloPts(P) (Divide + Conquer 2D Closest Pts)

Base Case: Later

Divide:



Sort by x

(1,5)	(2,2)	(4,0)	(10,20)	(12,3)	(20,6)

L R

midline: 7

Conquer:

$\delta_1 \leftarrow$ CloPts(L)

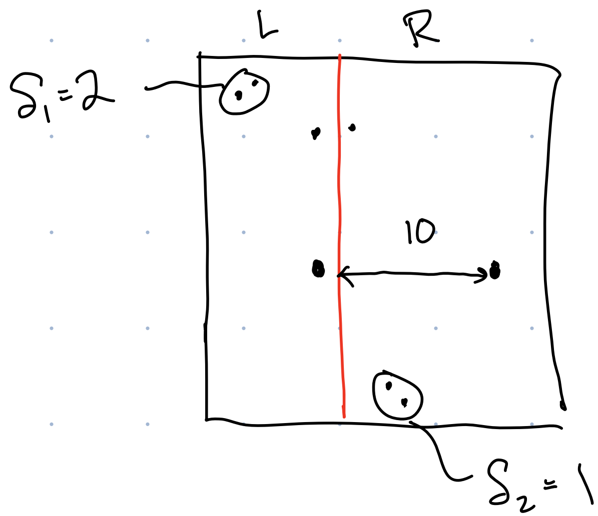
$\delta_2 \leftarrow$ CloPts(R)

$\delta \leftarrow \min(\delta_1, \delta_2)$

Combine

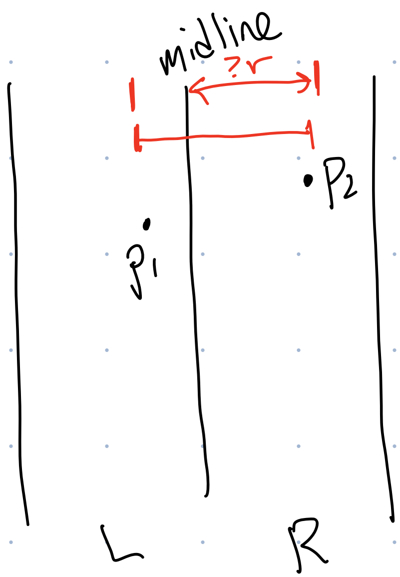
Combine:

Let's think about this:



Lemma: If p_2 in R has difference in x-coord. of $\boxed{\text{more than } r} = \delta$ from midline, then for any point $p_1 \in L$, $d(p_1, p_2) >$ $r = \delta$

Proof Sketch: (Basic ideas, but need more English in real proof)



$$d(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$(x_1 - x_2)^2 = \Big(\overset{>r}{\text{diff in x-coord from } x_2 \text{ to midline}} + \text{diff " " midline to } \underset{>0}{x_1}\Big)^2$$

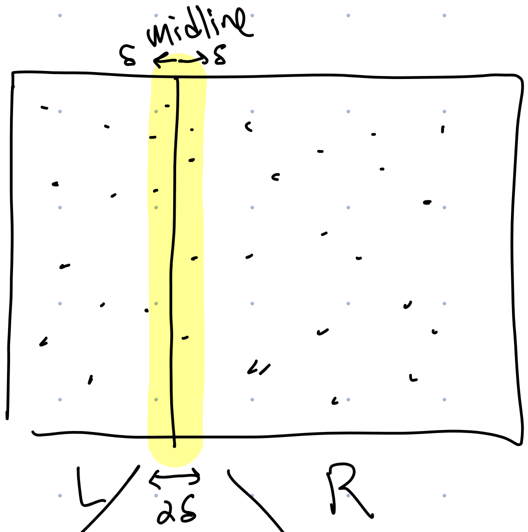
$$> r^2$$

$$(y_1 - y_2)^2 > 0$$

$$\Downarrow$$

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} > \sqrt{r^2}$$

$$d(p_1, p_2) > r$$



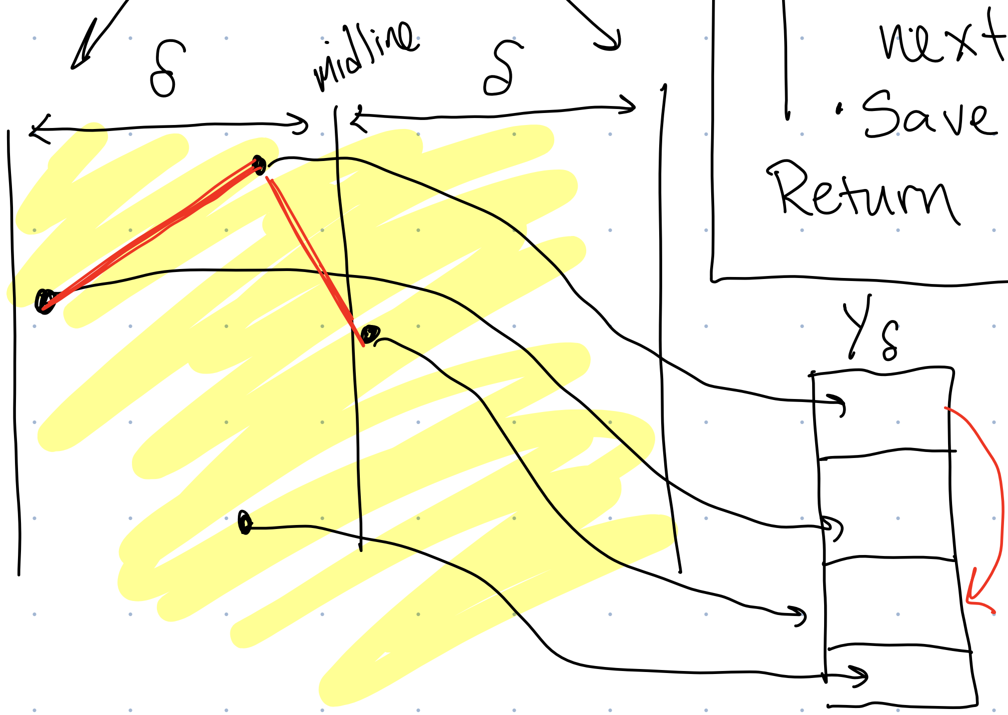
We want to check points in this region,
Looks like a line!

Combine Step

$Y_\delta \leftarrow$ y-sorted list of pts in P within δ of midline

For $p \in Y_\delta$
- ~~Check~~ check distance from p to next <u>2+</u> pts
- Save if smallest found

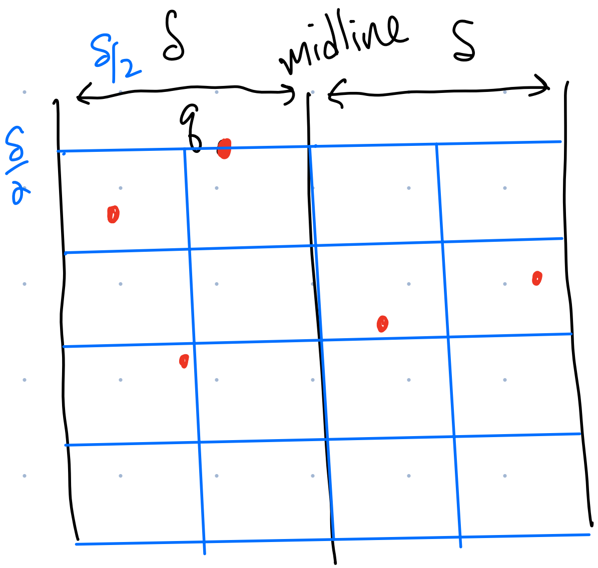
Return smallest distance found



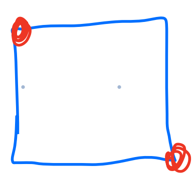
What goes here?

Lemma: Only need to look at next 7 pts in Y_s

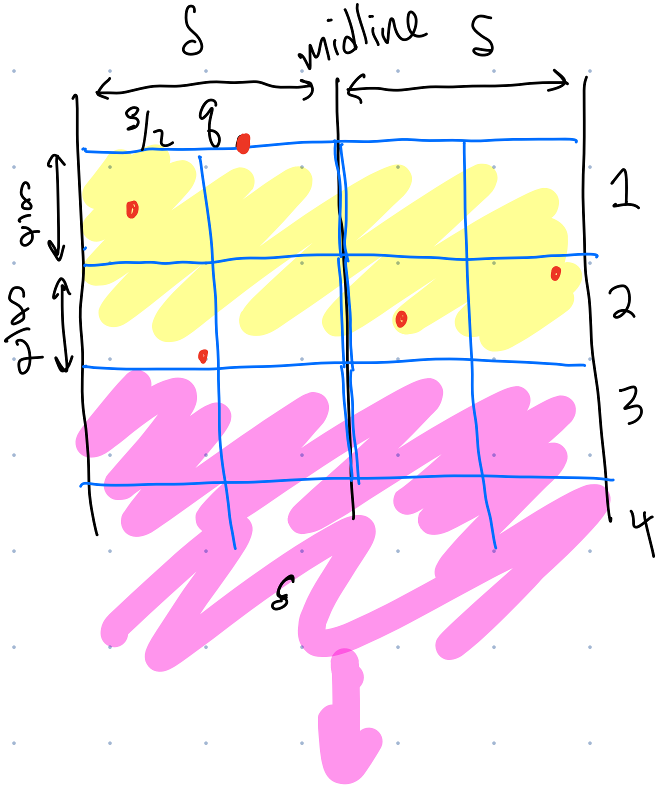
Pf: Imagine dividing region w/in δ of midline into $\frac{\delta}{2} \times \frac{\delta}{2}$ squares starting at current pt (q). Each square can contain at most one point. To see this, for contradiction, suppose there are 2 pts in a square. The pts have largest distance when on opp. corners. In that case, their distance is



$$\sqrt{\left(\frac{\delta}{2}\right)^2 + \left(\frac{\delta}{2}\right)^2} = \sqrt{\frac{\delta^2}{4} + \frac{\delta^2}{4}} = \sqrt{\frac{\delta^2}{2}} = \frac{\delta}{\sqrt{2}} < \delta$$



Each box is in L or R, so any 2 pts in a box must have distance $\geq \delta$, a contradiction



Pts in rows 3+ have distance at least δ from q, b/c diff in y-coordinate is at least δ.

So only 1st 2 rows are relevant

There are at most 7 pts other than q in 1st 2 rows, b/c only 8 box.

CloPts(P) (Divide + Conquer 2D Closest Pts)

Base Case: later

Divide: Sort by x

$$\underbrace{(1,5) \mid (2,2) \mid (4,0)}_{L} \;\Big|_{7}\; \underbrace{(10\ 20) \mid (12,3) \mid (20,6)}_{R}$$

Divide into L, R by midline

Conquer:

$\delta_1 \leftarrow \text{CloPt}(L)$

$\delta_2 \leftarrow \text{CloPt}(R)$

$\delta \leftarrow \min(\delta_1, \delta_2)$

Combine:

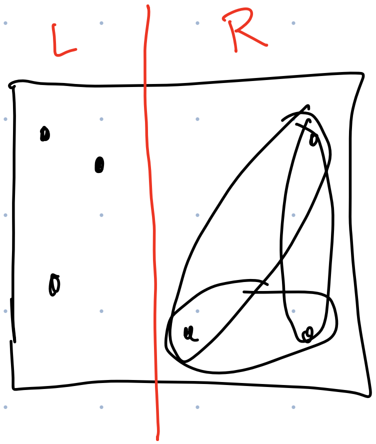
$Y_\delta \leftarrow$ pts w/in δ of midline, sorted by y-coordinate

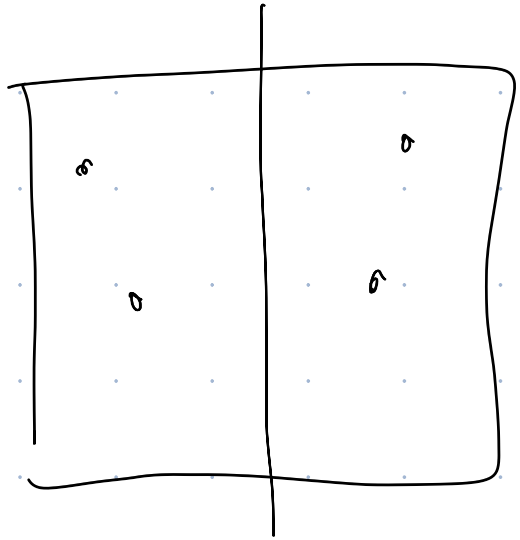
```
for p_i ∈ Y_δ:
    for j ← i+1 to i+7:
        if d(p_i, p_j) < δ, then δ ← d(p_i, p_j)
return δ
```

Base Case!

What size set of pts should trigger base case?

A) 0 B) ≤ 1 C) ≤ 2 $\boxed{\text{D) } \leq 3}$





CloPts(P) (Divide + Conquer 2D Closest Pts)

$O(1)$ **Base Case**: If $|P| \le 3$, then do brute force

Divide: Sort by x $O(n \log n)$

$$\underbrace{(1,5)\ \big|\ (2,2)\ \big|\ (4,0)}_{L}\ \Big|_{7}\ \underbrace{(10,20)\ \big|\ (12,3)\ \big|\ (20,6)}_{R}$$

Divide into L, R by midline $O(n)$

Conquer:

$\delta_1 = \text{CloPt}(L)$ $T\left(\frac{n}{2}\right)$

$\delta_2 = \text{CloPt}(R)$ $T\left(\frac{n}{2}\right)$

$\delta = \min(\delta_1, \delta_2)$ $O(1)$

$n = |P|$

$$T(n) = \begin{cases} O(1) & \text{if } n \le 3 \\ 2T\left(\frac{n}{2}\right) + O(n \log n) & \text{else} \end{cases}$$

$a = 2 \quad b = 2 \quad d = 1.01 \quad a \ ? \ b^d$

Combine:

$Y_\delta \leftarrow$ pts w/in δ of midline, sorted by y-coordinate $O(n \log n)$

for $p_i \in Y_\delta$: $O(n)$

  for $j \leftarrow i+1$ to $i+7$: $O(1)$

    if $d(p_i, p_j) < \delta$, then $\delta \leftarrow d(p_i, p_j)$ $O(1)$

return δ $O(1)$

Problem: Sorting at each recursive step takes too long.

PreSort(P)

$X \leftarrow P$ sorted by x

$Y \leftarrow P$ sorted by y

return X, Y

$O(n\log n)$

CloPts(X, Y)

1 If $|X| \le 3$, do Brute Force $O(1)$

2. Divide into X_L, Y_L, X_R, Y_R $O(n)$

3. $\delta = \min\{\text{CloPts}(X_L, Y_L), \text{CloPts}(X_R, Y_R)\}$ $2T(\frac{n}{2})$

4 Create Y_δ $O(n)$

5. For $p_i \in Y_\delta$: $O(n)$

   For $j \leftarrow i+1$ to $i+7$ $O(1)$

      if $\text{dist}(p_i, p_j) < \delta$, $\delta \leftarrow \text{dist}(p_i, p_j)$

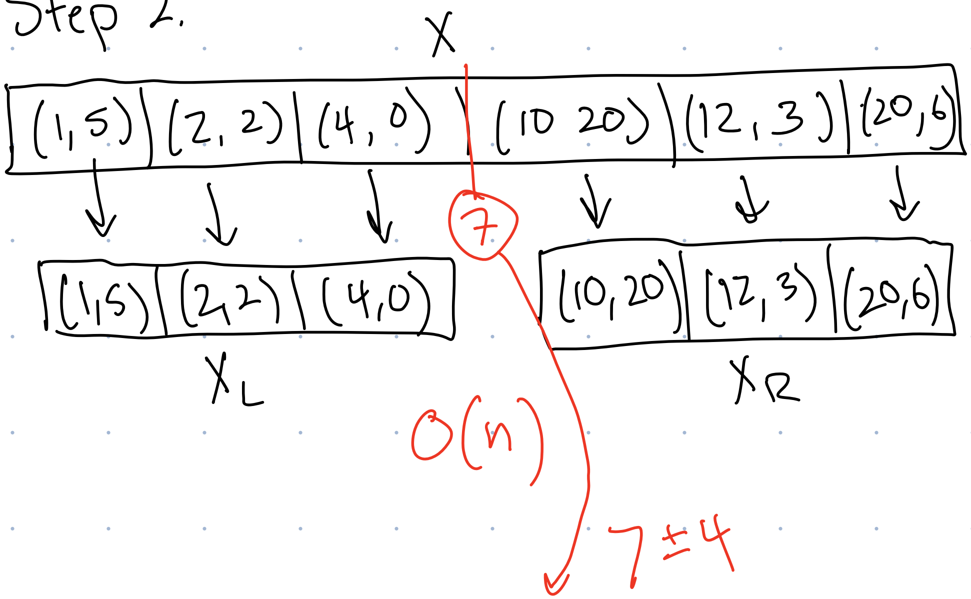
6 Return δ.

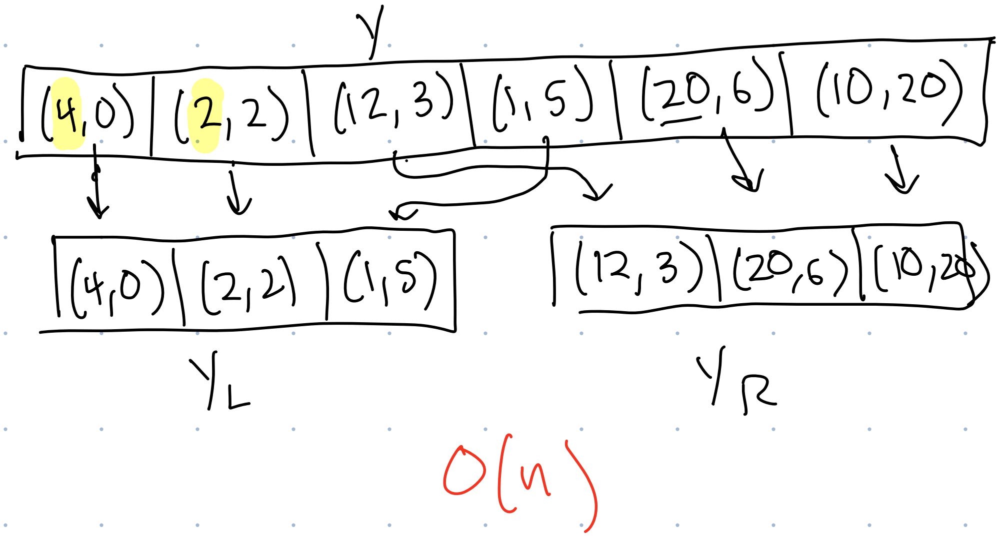
$$\boxed{T(n) = \begin{cases} O(1) & \text{if } n \le 3 \\ 2T(\frac{n}{2}) + O(n) & \text{else} \end{cases}}$$

$$O(n\log n) + O(n\log n) = O(n\log n)$$

Dividing Arrays Efficiently

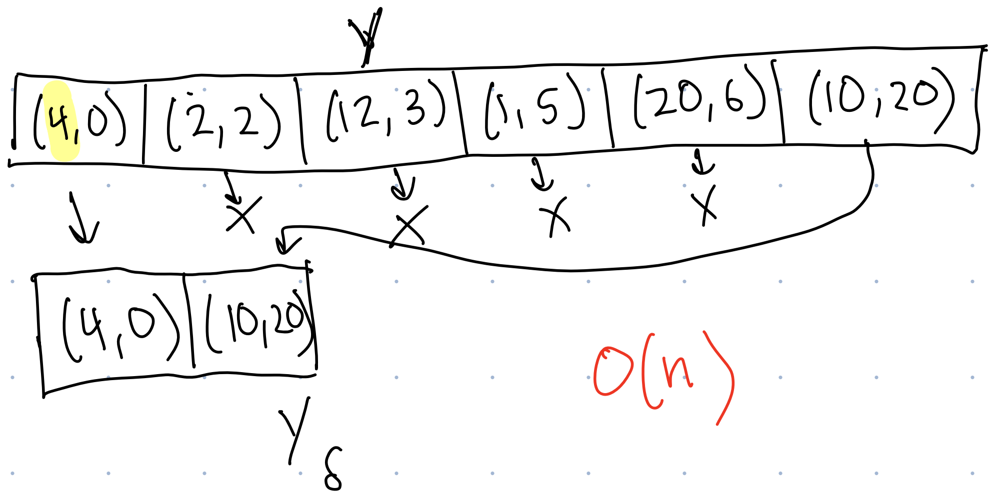
Step 2.





midline is 7

Step 4 : $\delta = 4$ Want x from 3 to 11



- Where did we use our assumption that $x, \cancel{y}$ points are unique?

- You can do Prg. Assign. 1!