ECS 122A Lecture 4

Divide + Conquer

Closest pair of points in 2D

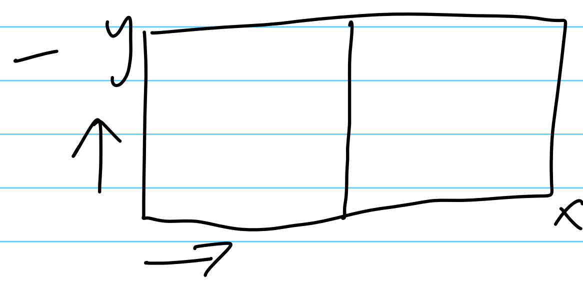
Trivial $O(n^2)$ alg

Given list of n points (x_i, y_i)

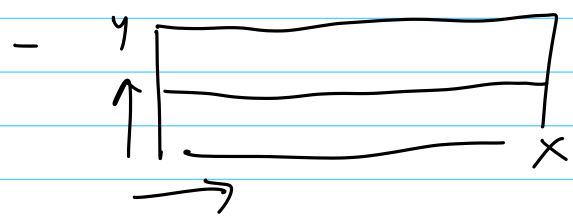
Find the pair closest to each other.

Want to improve to $O(n \log n)$ time.

How to divide?

- 

?

- 

?

- For $O(n \log n)$ time, want split into $\frac{n}{2}, \frac{n}{2}$ halves.

Split by median in x-coordinate (can do y too)

Observation (case analysis):

Closest pair either

- both in left half recursion

- both in right half
- One left one right ← needs work

Alg outline:

function closest (List A)
    Find median, split into left, right halves
    d_L = closest(left half)
    d_R = closest (right half)

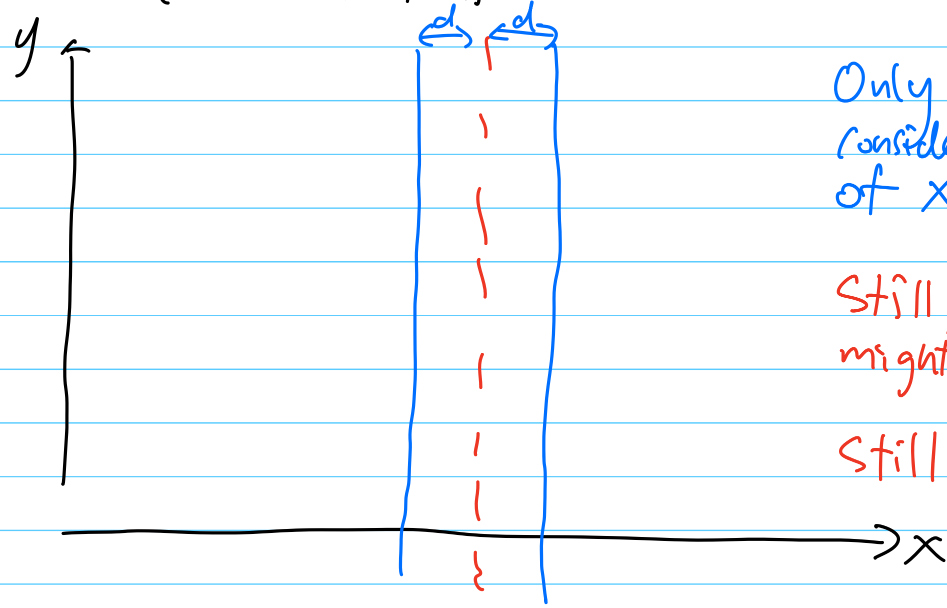
    // Compute one left one right case
        as d_{cross} in $O(n)$ time
return $\min(d_L, d_R, d_{cross})$

How to compute one left one right case?

Trivial : $O(n^2) \Rightarrow$ total runtime $\gg O(n \log n)$

The alg is clever geometry. Good to see

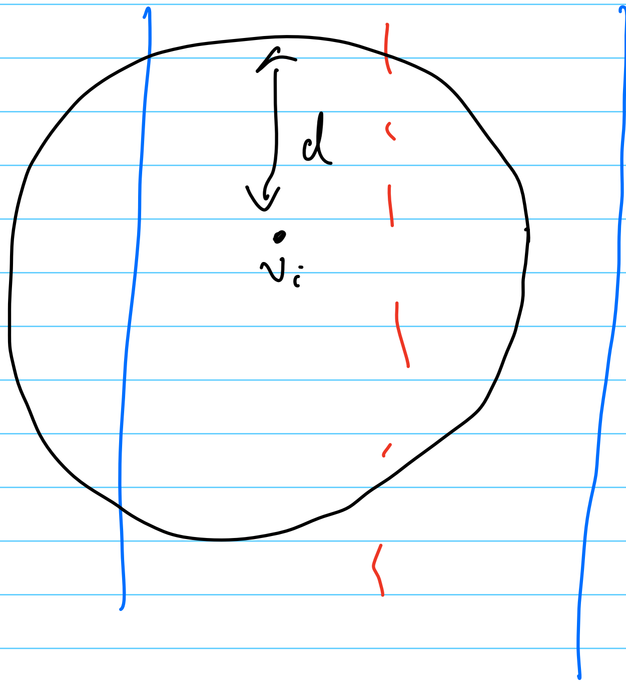
Let $d = \min(d_L, d_R)$



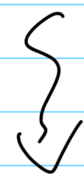
Only need to consider strip of x-median $\pm d$

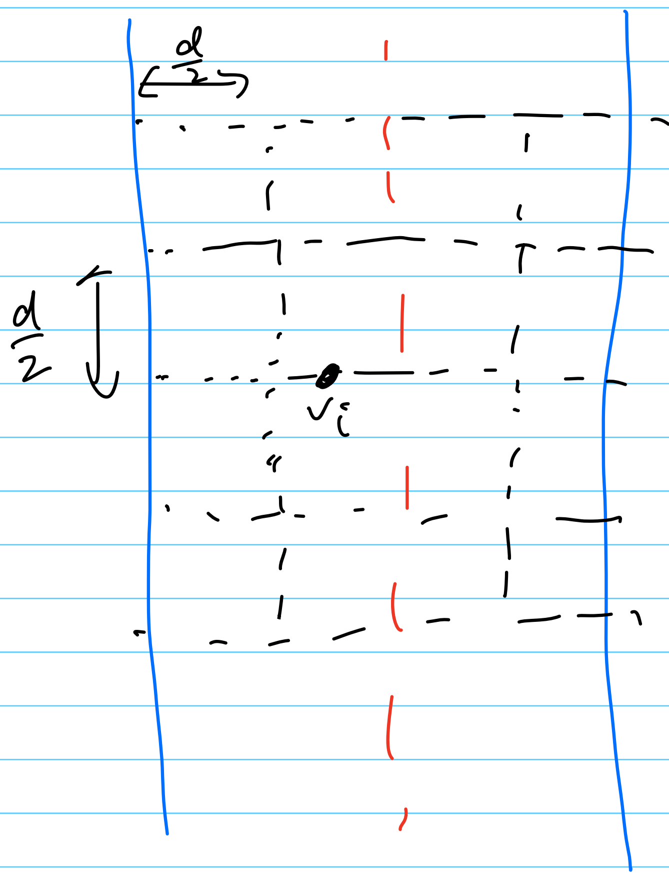
Still, every point might be in strip

Still $O(n^2)$...



Only need to consider radius-d ball around v_i (the intersection w/ strip)





This grid covers the radius-d circle around v_i (part within strip)

At most 1 point in each square since square has diameter/diagonal $d/\sqrt{2}$

Only need to consider 15 closest points in y-coordinate in strip

Actually only 7 points, if we sweep from small to large y.

function closest(List A)
  Find median, split into left, right halves
  d_L = closest(left half)
  d_R = closest(right half)

  Sort points in A in y-coordinate, new list A' → filter to strip of $\min(d_L, d_R)$

  for each v_i in A', compare w/ next 7 points in A'. ⇒ compute min dist d_{cross}

return $\min(d_L, d_R, d_{cross})$.

O(n log n) by sort, though cleverer alg for O(n)

O(n log n)

Runtime? $O(n \log^2 n)$ ... not quite.

If only we can sort in $O(n)$ time...

Observation: closest has exactly the same recursive structure as mergesort!

☆ Do both at the same time

(compute closest, merge in x-coord & y-coord)

Exercise to figure out details.

Strassen's Algorithm

Given 2 $n \times n$ matrices A, B, compute
$$C = AB$$

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \qquad B = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}$$

$c_{ij} = \langle a_i, b_j \rangle$ ~$O(n)$ time

Naive computation is $O(n^3)$

Matrix multiplication inherently has D&C structure

$$\underset{A}{\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}} \qquad \underset{B}{\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}}$$

$$= \begin{array}{ccc} & \begin{array}{cc} C_{11} & \qquad C_{12} \end{array} & \\ C_{21} & \underset{C}{\begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{22}B_{12} + A_{22}B_{22} \end{bmatrix}} & C_{22} \end{array}$$

8 $\frac{n}{2} \times \frac{n}{2}$ multiplications

Runtime recurrence:

$$T(n) \approx 8\,T\left(\frac{n}{2}\right) + c_1 \cdot n^2$$

Can prove that $T(n) \geq \Omega(n^3)$

This doesn't help

Strassen's Observation: (Magic)

Do 7 clever multiplication instead (plus more $O(n^2)$ arithmetic)

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_2 = (A_{21} + A_{22})\,B_{11}$$

$$M_3 = A_{11}\,(B_{12} - B_{22})$$

$$M_4 = A_{22}\,(B_{21} - B_{11})$$

$$M_5 = (A_{11} + A_{12})\,B_{22}$$

$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$

$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{bmatrix}$$

$$T(n) \le 7\,T\left(\frac{n}{2}\right) + cn^2$$

$\Rightarrow$ Can show that $T(n) \le c' n^{\log_2 7} - cn^2$

$\uparrow$ sufficiently large

$\log_2 7 < \log_2 8 = 3$

Fun exercise:

Given complex #s

$a+bi$, $c+di$

compute product using 3 real number multiplications.