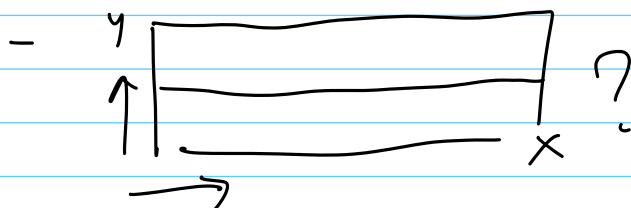# ECS 122A Lecture 4
Divide + Conquer

Closest pair of points in 2D

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 $\frac{y}{2}$, $\frac{n}{2}$ 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)
```
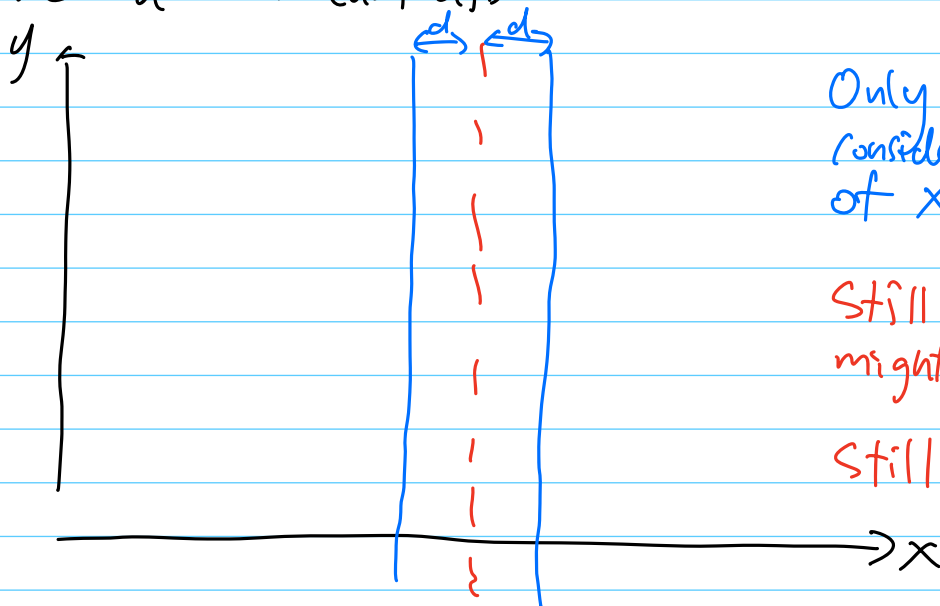
How to compute one left one right case?

Trivial: $O(n^2)$ ⇒ total runtime >> $O(n \log n)$

The alg is clever geometry. Good to see

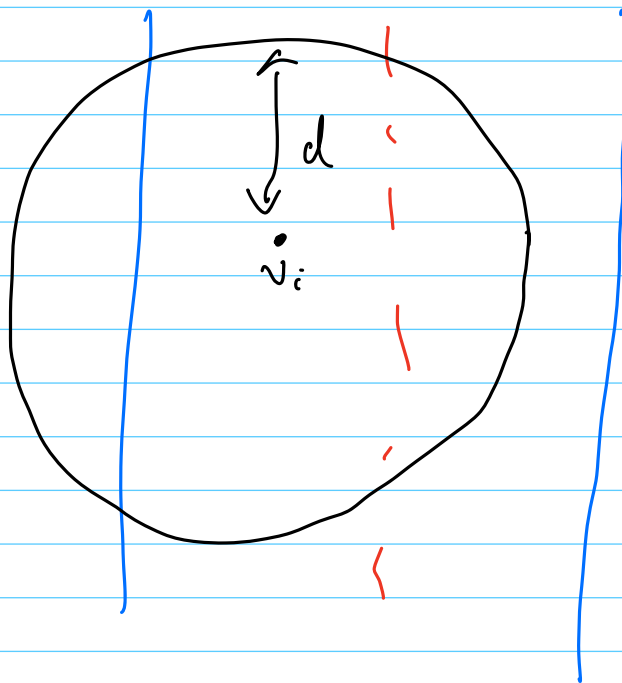Let $d = min(d_L, d_R)$

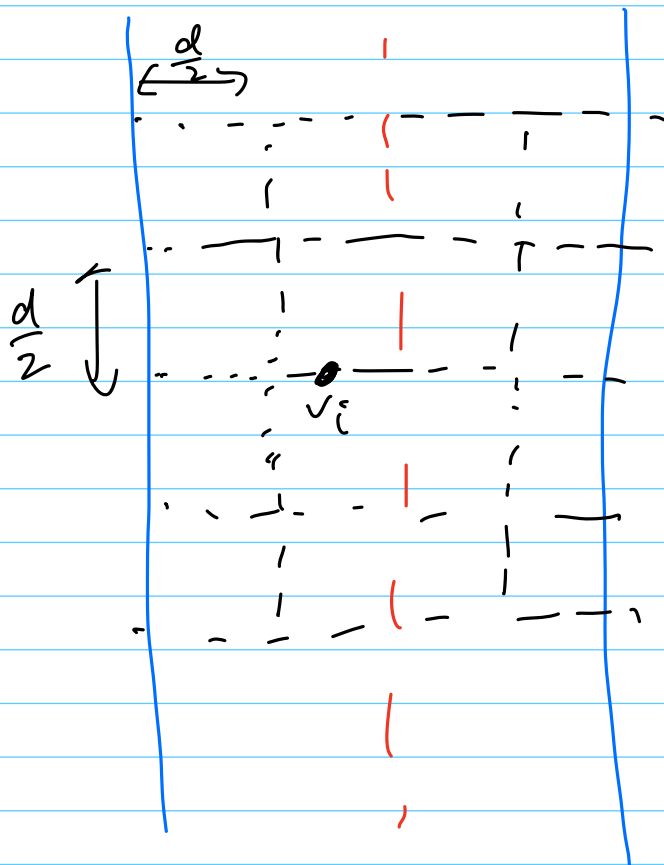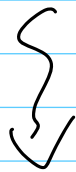Only need to consider strip of x-median ± d

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 algs 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 DfC structure

$$A = \begin{array}{|c|c|} \hline A_{11} & A_{12} \\ \hline A_{21} & A_{22} \\ \hline \end{array} \qquad B = \begin{array}{|c|c|} \hline B_{11} & B_{12} \\ \hline B_{21} & B_{22} \\ \hline \end{array}$$

$$A \qquad\qquad B$$

$$= \begin{array}{|c|c|} \hline \overset{C_{11}}{A_{11}B_{11}} & \overset{C_{12}}{A_{11}B_{12}} \\ + A_{12}B_{21} & + A_{12}B_{22} \\ \hline A_{21}B_{11} & A_{21}B_{12} \\ + A_{22}B_{21} & + A_{22}B_{22} \\ \hline \end{array} \qquad 8 \ \frac{n}{2} \times \frac{n}{2}$$

C_{21} (left) $\qquad C_{22}$ multiplications

$$C$$

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)

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

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

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

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

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

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

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

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

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

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

$$\uparrow$$

sufficiently large

$$\log_2 7 < \log_2 8 = 3$$

Fun exercise:

Given complex #s

$$a + bi \quad , \quad c + di$$

compute product using 3 real number
multiplications.