Chapter 6. Geometric Algorithms

Topics include:

- Determine for a set of n line segments if 2 of them intersect.
- Determine whether a point is inside a polygon.
- Construct simple polygons
- Construct convex hull of n points: Graham's scan & Gift-wrapping alg.
- Find the closest pair of points in a set of n points

6.1. Properties of Line Segments

A point is a pair $p = (x, y) \in \mathbb{R}^2$

Let $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$ be 2 pts.

The line segment $\overline{p_1 p_2}$ is the set

$$\overline{p_1 p_2} = \{ \lambda p_1 + (1-\lambda) p_2 \mid 0 \le \lambda \le 1 \}$$

$$= \{ \text{Convex combinations of } p_1, p_2 \}$$

$\overrightarrow{p_1 p_2}$ denotes a directed segment

For $\overrightarrow{p_1p_2}$, if $p_1 = (0,0)$, we write $\overrightarrow{p_2}$

Some basic questions concerning line segments:

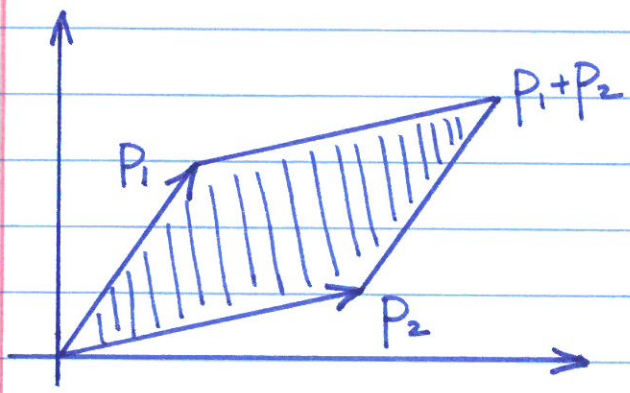
(1) Given $\overrightarrow{p_0p_1}$, $\overrightarrow{p_0p_2}$, is $\overrightarrow{p_0p_1}$ clockwise from $\overrightarrow{p_0p_2}$ w.r.t. p_0 ?

(2) Given $\overline{p_1p_2}$, $\overline{p_2p_3}$, do we make a left/right turn when traversing from p_1 to p_2 and then p_3 ?

(3) Do 2 line segments $\overline{p_1p_2}$ and $\overline{p_3p_4}$ intersect ?

Claim. The above questions (1), (2), and (3) can be answered in $O(1)$ time using only $+, -, *$ and comparisons.

Cross Product



$p_1 = (x_1, y_1)$ $p_2 = (x_2, y_2)$

$$p_1 \times p_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix}$$

$$= x_1y_2 - x_2y_1$$

$$= -p_2 \times p_1$$

(= Area of parallelogram spanned by $\overrightarrow{p_1}, \overrightarrow{p_2}$)

$p_1 \times p_2 > 0 \Rightarrow p_1$ is clockwise from p_2

$< 0 \Rightarrow p_1$ is counterclockwise fr. p_2

$= 0 \Rightarrow \vec{p_1}, \vec{p_2}$ are collinear (pointing same/opposite direction)

To determine whether $\vec{p_0p_1}$ is clockwise from $\vec{p_0p_2}$:

- Translate p_0 to origin $(0,0)$ by:

$p_1' = (x_1 - x_0, y_1 - y_0)$

$p_2' = (x_2 - x_0, y_2 - y_0)$

- If $p_1' \times p_2' > 0$, then $\vec{p_0p_1}$ is clockwise from $\vec{p_0p_2}$

To determine whether consecutive segments turn left/right:

Consider $\overline{p_0p_1}, \overline{p_1p_2}$

Left turn if $\vec{p_0p_2}$ is counterclockwise from $\vec{p_0p_1}$

To determine whether 2 segments intersect:

Def. The bounding box of a figure is the smallest rectangle containing the figure s.t. its sides are parallel to x- and y-axes.

A bounding box is represented by lower left and upper right points.

Ex. A line segment $\overline{p_1 p_2}$ has bounding box $(\hat{p}_1, \hat{p}_2)$ with lower left point $\hat{p}_1$ and upper right point $\hat{p}_2$

where $\hat{x}_1 = \min(x_1, x_2)$ $\hat{y}_1 = \min(y_1, y_2)$

$\hat{x}_2 = \max(x_1, x_2)$ $\hat{y}_2 = \max(y_1, y_2)$ □

Fact. Two rectangles $(\hat{p}_1, \hat{p}_2)$ and $(\hat{p}_3, \hat{p}_4)$ intersect iff $\hat{x}_2 \geq \hat{x}_3$, $\hat{x}_4 \geq \hat{x}_1$,

$\hat{y}_2 \geq \hat{y}_3$, $\hat{y}_4 \geq \hat{y}_1$ □

Fact. If the bounding boxes of 2 line segments do not intersect, then neither do the segments. □

Def. A segment <u>straddles</u> a line if p_1 is on one side and p_2 is on the other side of the line.

Fact. Two line segments intersect iff

(1) Their bounding boxes intersect, and

(2) Each segment straddles line containing the other. ⊠

6.2. Is a point inside a polygon?

Def. A path is a sequence of points $(P_0, P_1, \dots, P_n)$ and line segments

$\overline{P_0P_1}, \overline{P_1P_2}, \dots, \overline{P_{n-1}P_n}$

It's closed if $P_n = P_0$

A polygon is a closed path

A polygon is simple if none of their segments intersect.

A polygon (simple) encloses a region in the plane: the inside of the poly.

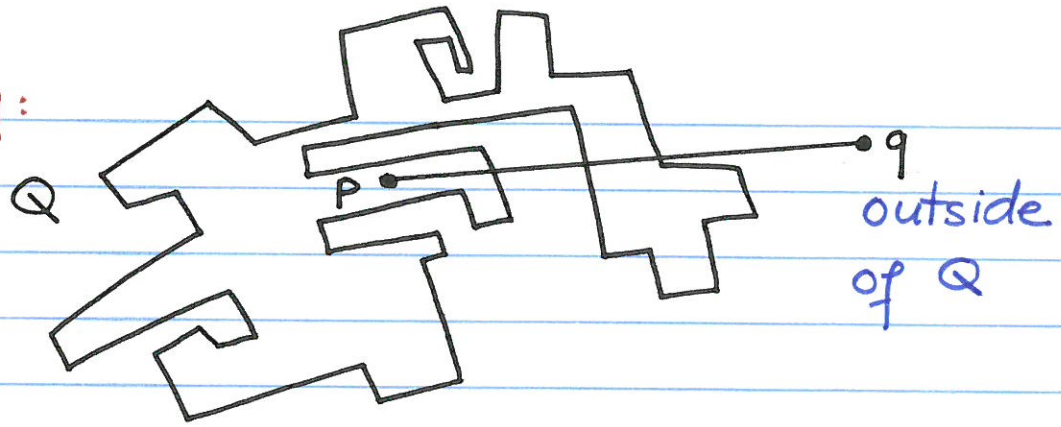
A simple polygon is a convex poly. if every line segment connecting 2 points inside the polygon lies entirely inside the polygon.

Consider the following problem:

Input. A simple polygon Q and a point p.

Question. Is p inside Q?

Idea:



Count the number of times $\overline{pq}$ intersect edges of Q:

if odd, p is inside

otherwise p is outside

INSIDE_OUTSIDE ($Q = \langle p_0, \ldots, p_n \rangle$, $p = \langle x, y \rangle$)

Let $p_i = \langle x_i, y_i \rangle$, $i = 0, \ldots, n$

Choose $\hat{y}$ s.t. $\hat{y} > \text{Max}\{y, y_0, \ldots, y_n\}$

Let $q = \langle x, \hat{y} \rangle$

Let δ be # of times $\overline{pq}$ intersects edges of Q

if δ is odd then p is inside Q

else p is outside Q

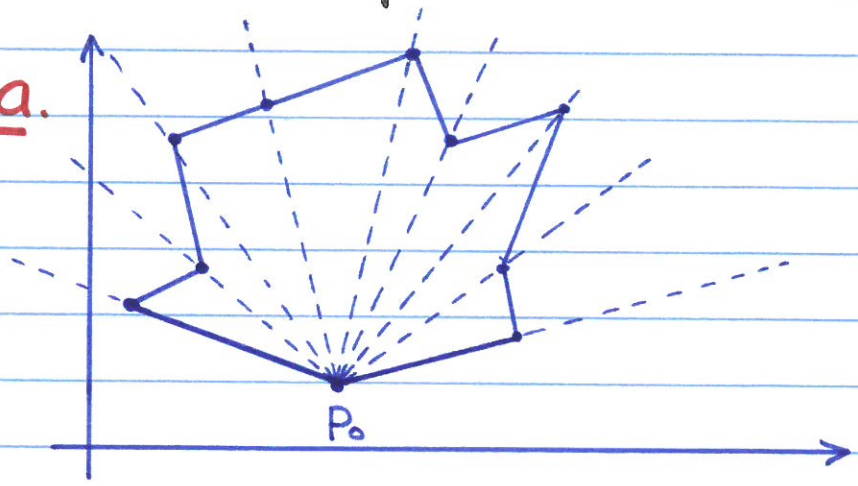
Complexity $O(n)$ time.

6.3. Constructing Simple Polygons.

Input. A set $\{p_0, p_1, \ldots, p_{n-1}\}$ of n points

Output. A simple polygon connecting these points.

Idea.



SIMPLE-POLYGON $(\langle p_0, p_1, \ldots, p_{n-1} \rangle)$

- Let p_0 be the point with least y-coord (if several exist, choose one with least x-coord.)
- For $i = 1$ to $n-1$ do
  compute slope s_i of $\overline{p_0 p_i}$
- Sort s_i in nondecreasing order (i.e. counterclockwise) starting with positive slopes (if same slope, sort by distances)
- $\langle p_0, p_{i_1}, \ldots, p_{i_{n-1}} \rangle$ is the simple polygon

6.4. Determining whether any pair of segments intersect

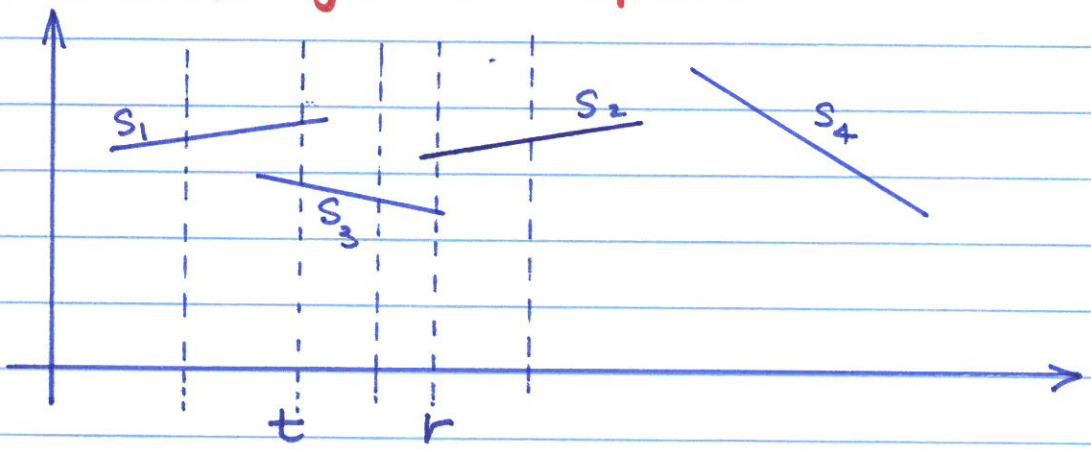
Input. n line segments

Question. $\exists\ s_i, s_j : s_i, s_j$ intersect ?

Assumptions: (1) No vertical line segments.

(2) No 3 segments intersect at 1 point.

The Sweeping Technique



Segments can be ordered w.r.t. an imaginary vertical sweep line

w.r.t. sweep line t: s_1 is above s_3

$$s_1 \geq_t s_3$$

Similarly, $s_2 >_r s_3$

s_4 is incomparable with s_1, s_2, s_3 w.r.t. $\geq_r$, $>_t$

Fact. Given any sweep line x, the relation $\gtrsim_x$ is a total order on those segments that intersect x.

Observation. When sweeping line x from left to right:

- A segment *enters* the ordering $\gtrsim_x$ when its left endpoint is encountered
- It *leaves* $\gtrsim_x$ when its right endpoint is encountered by x.

Question. What happens when the sweep line passes through intersection of 2 segments?

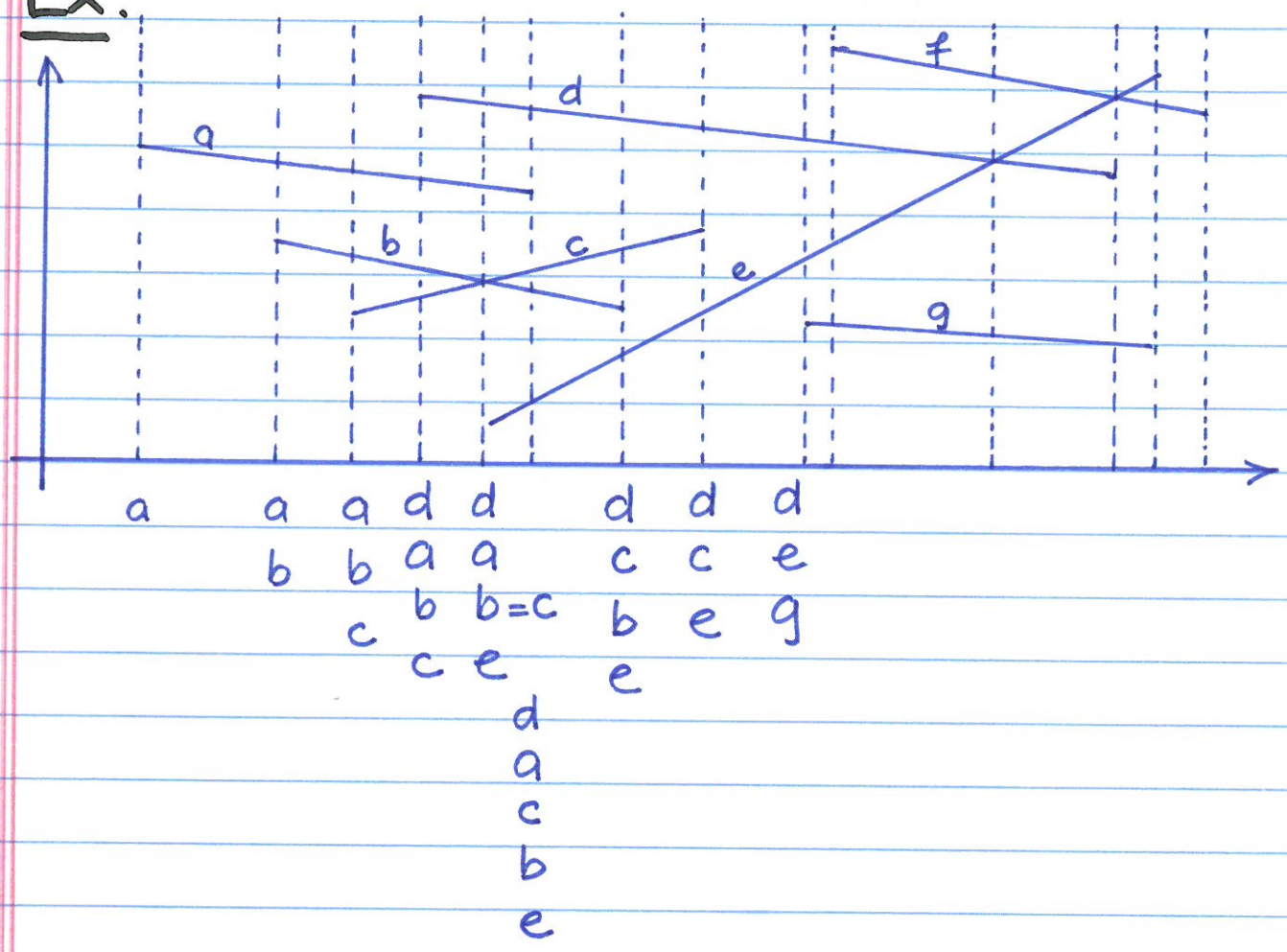
$\Rightarrow$ Their positions in the order change! (reverse)

We maintain 2 sets of data:

(1) Sweep line status: current ordering of segments

(2) Halting positions: where ordering changes.

Ex.



Sweep line status is a total order T.

To maintain T we need these operations:

- INSERT (T, s) : insert segment s to T
- DELETE (T, s) : remove s from T
- ABOVE (T, s) : return segment in T immediately above s
- BELOW (T, s) : return segment in T immediately below s.

Remark. Using red-black trees each of these operations can be performed in $O(\lg n)$ time, where n = # of input segments

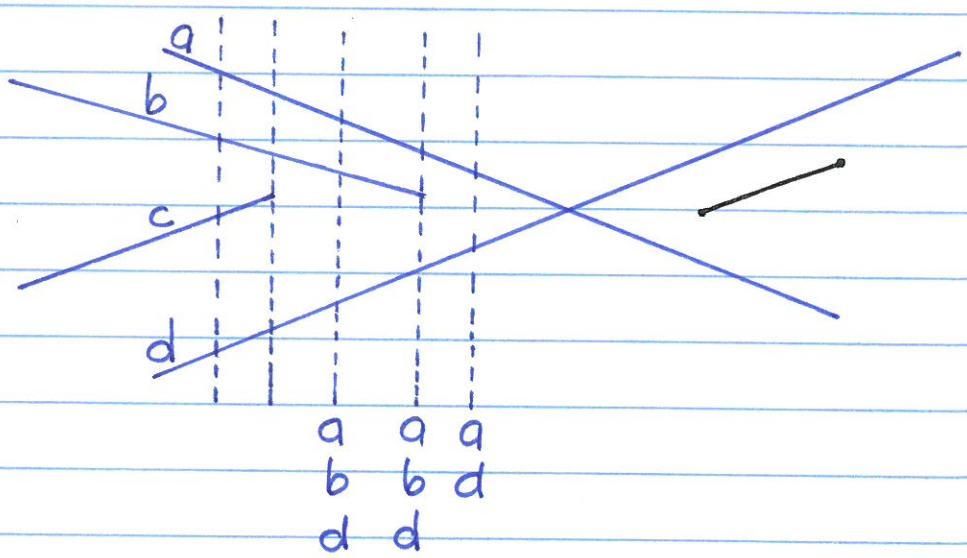
SEGMENT_INTERSECTION $(\langle s_1, \ldots, s_n \rangle)$

- $T = \phi$
- Sort endpoints of s_i's by x-coordin.
  (Ties broken by putting points with smaller y-coord. first)
- for each point p on the above list do
  - if p is left endpoint of segment s
    then INSERT(T, s)
    - if $\exists$ u = ABOVE(T,s) and s,u intersect, or
      $\exists$ v = BELOW(T,s) and s,v intersect
      then return true
  - if p is right endpoint of segment s
    then if $\exists$ u = ABOVE(T,s) and v = BELOW(T,s) s.t u,v intersect
    then return true

DELETE(T, s)

- return **false**

Observation. If 2 segments intersect, they are consecutive in T at some point:



Complexity: $O(n \lg n)$

6.5. Computing Convex Hull

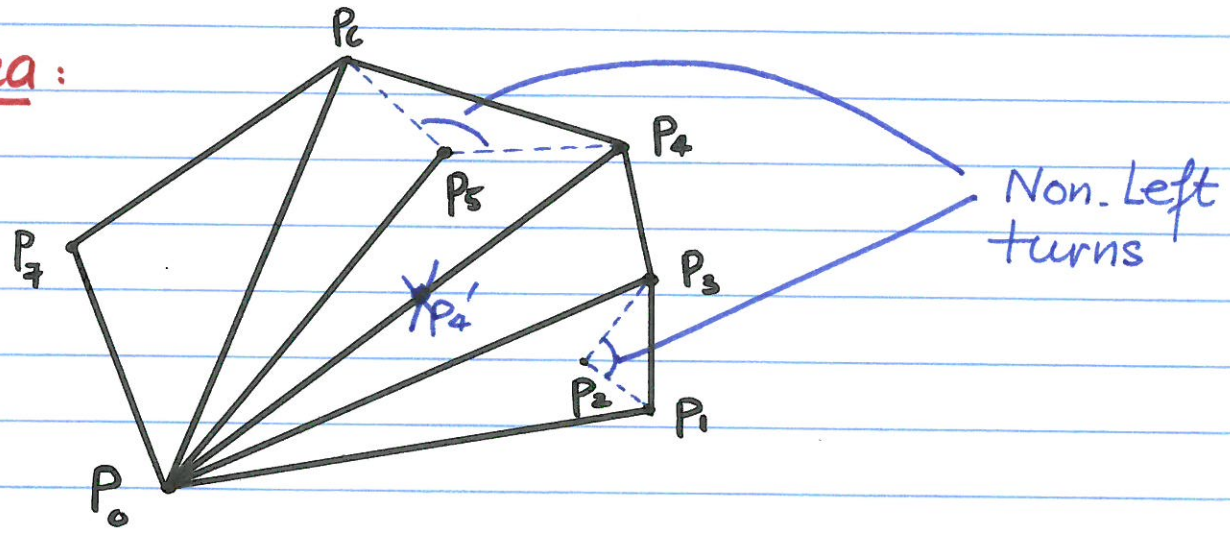
Def. The convex hull of a set S of points is the smallest convex polygon P for which each point in S is
- either on boundary of P
- or it's inside P

We write: $CH(S)$

We present 2 methods:
- Graham's scan
- Gift-wrapping algor.

Graham's Scan.

Idea:



GRAHAM-SCAN ($S = \langle p_0, p_1, \ldots, p_{n-1} \rangle$)

. Let $p_0 \in S$ be with smallest y-coord.
  /* If several, choose p_0 with least x-coord. */
. Let $\langle p_1, p_2, \ldots, p_{n-1} \rangle$ be remaining points sorted by slopes in counterclockwise order relative to p_0
  /* If there are several points with same slope, then remove all but the farthest point from p_0 */
. Push p_0, p_1, p_2 on stack
  /* with p_2 on top of stack */
. for $i = 3$ to $n-1$ do
    while traversing $\overline{uv}$, $\overline{vp_i}$ makes a non. left turn, where
      u = RIGHT_BELOW_TOP(S)
      v = TOP(S)
      do POP(S)
    PUSH(p_i, S)
. return S

Complexity : $O(n \lg n)$ time

Gift-wrapping algor.

Gift-WRAPPING ($S = \langle p_0, p_1, \ldots, p_{n-1} \rangle$)

- Let $p \in S$ be point with largest x-coord.
  /* If several exist, choose one with largest y-coord. */
- $P = \{p\}$
- **repeat**
  - Let q be point $\notin P$ s.t. slope of q relative to p is minimal (counter-clockwise)
  - Add q to P
  - $P = q$

  until no such point q can be found
- **return** P

Complexity : If $CH(S)$ has h vertices, then gift-wrapping takes $O(n \cdot h)$ steps.

6.6. Closest Pair of Points.

Input. A set $Q = \{p_1, \ldots, p_n\}$ of n points

Output. $1 \le i < j \le n$:

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

is minimal.

The Divide-and-Conquer approach

We want to find closest pair of $P \subseteq Q$

Let array X be P sorted by x-coord.

array Y be P sorted by y-coord.

If $|P| \le 3$, find closest pair directly

Divide Step Let $|P| = m$

Find a vertical line ℓ that bisects P into subsets P_L, P_R s.t.

$$|P_L| = \lceil \frac{m}{2} \rceil \;, \; |P_R| = \lfloor \frac{m}{2} \rfloor$$

All points in P_R (P_L) are to the right (left) or on line ℓ

Let X_L (X_R), Y_L (Y_R) be arrays associated with P_L (P_R)

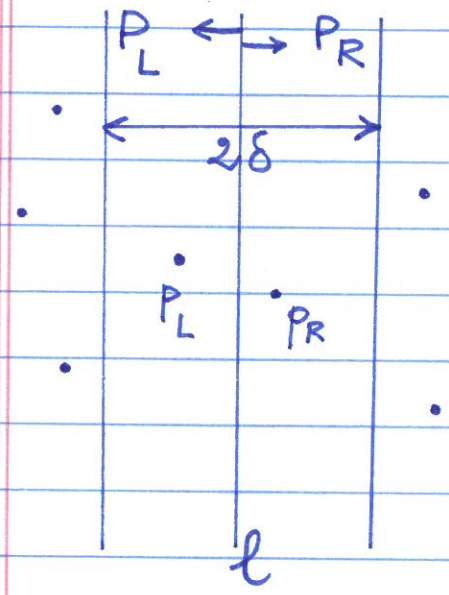
Conquer Step

Recursively obtain distance δ_L (δ_R) from P_L (P_R)

Let $\delta = \text{Min}(\delta_L, \delta_R)$

A problem. The closest pair may lie in the 2δ-wide vertical strip centered at line ℓ.

Question. How to find out if this occurs?



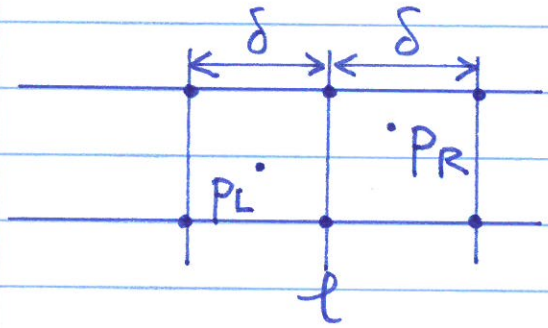
- Let P' be set of points within vertical strip
- Let Y' be P' sorted according to y-coord.
- For each point $p \in Y'$ find points $p' \in Y'$ within distance δ from p

Question. Given p, how many points in Y' following p do we need to check?

Claim 1. For each $p \in Y'$ we need only check 7 points following p in Y' to see if $d(p, p') < \delta$.

Consider a closest pair $(p_L, p_r) \in P_L \times P_R$ s.t. $\delta' = d(p_L, p_R) < \delta$.

Clearly, p_L, p_R are within a $\delta \times 2\delta$ rectangle centered at line ℓ:



Claim 2. At most 8 pts of P can reside within the $\delta \times 2\delta$ rectangle.

There can be at most 4 pts of P_L in the left square since they are at distance $\geq \delta$ from each other.

Similarly there can be at most 4 pts of P_R in the right square.

$\Rightarrow$ Claim 1.

CLOSEST-PAIR ($\langle p_1, p_2, \ldots, p_n \rangle = Q$)

- Sort Q by x-coord. to obtain X
- Sort Q by y-coord. to obtain Y
- Using X divide Q into P_L, P_R
  - Obtain X_L, X_R from X by cutting X at middle
  - Obtain Y_L, Y_R by "unmerging" Y
- Recursively compute closest pairs & shortest distances δ_L, δ_R in P_L, P_R
- Let $\delta = \min(\delta_L, \delta_R)$
- Let P' be points of Q within 2δ-wide strip centered at line ℓ
- Compute Y' for P' by "unmerging" Y
- for each $p \in P'$ in increasing order do
  - search for closest pair among p and the next 7 neighbors in Y'
  - Update if new closest pair found

Complexity:

Since "unmerging" takes $O(n)$ time, the recurrence is

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

$$\Rightarrow T(n) = O(n \lg n)$$