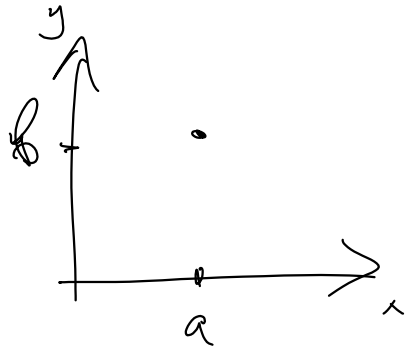
Computational Geometry

Algorithms for problems that involve geometric objects:

- points
- lines
- line segments
- planes (in 3-D & above)

Geometry in 2-D

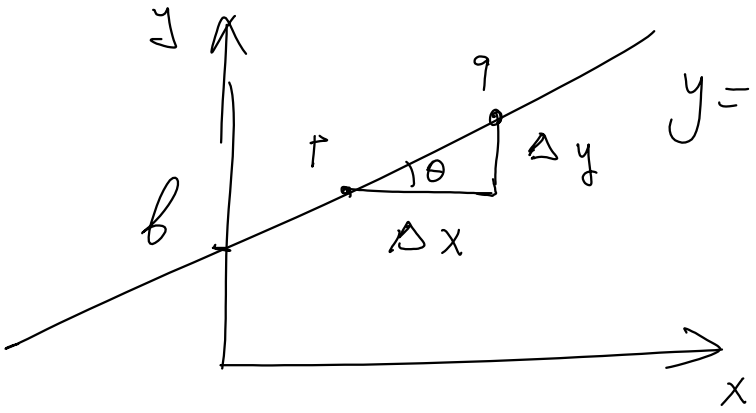
Points: $p = (a, b)$



lines: $ax + by + c = 0$

$y = mx + b$ ← non-vertical lines

m: slope b: y-intercept



$y = mx + b$

$$m = \frac{\Delta y}{\Delta x} = \tan\theta$$

$$\ell = (m, b)$$

Primal Space

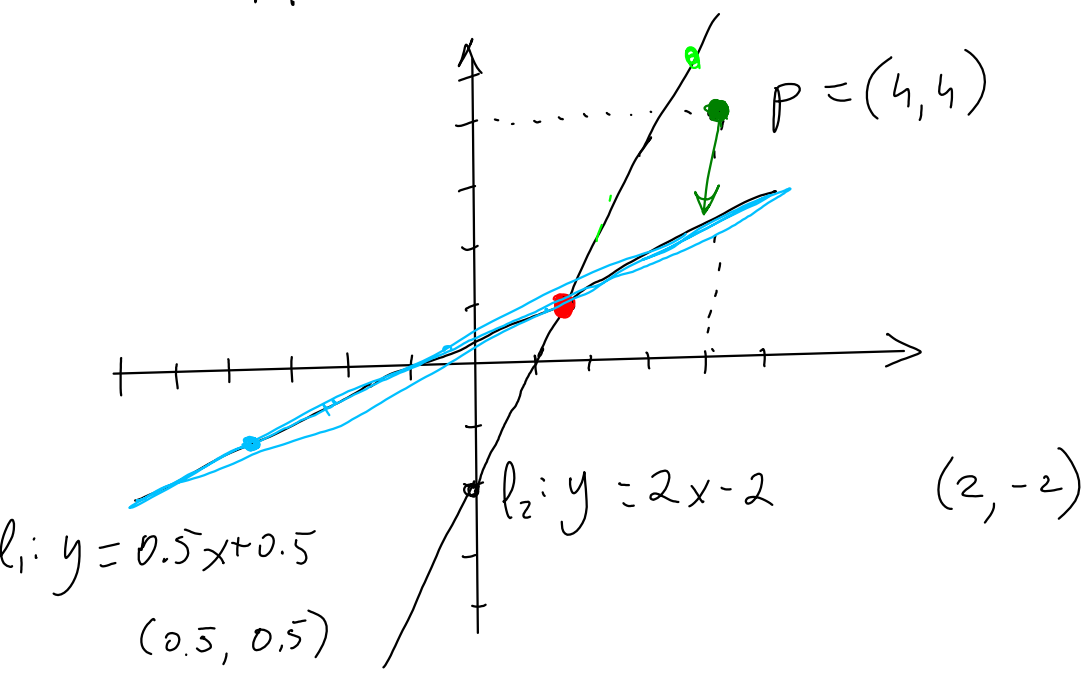
$p = (a, b)$ $\Longrightarrow$ Dual Space $p^* = (a, -b)$

$y = ax - b$

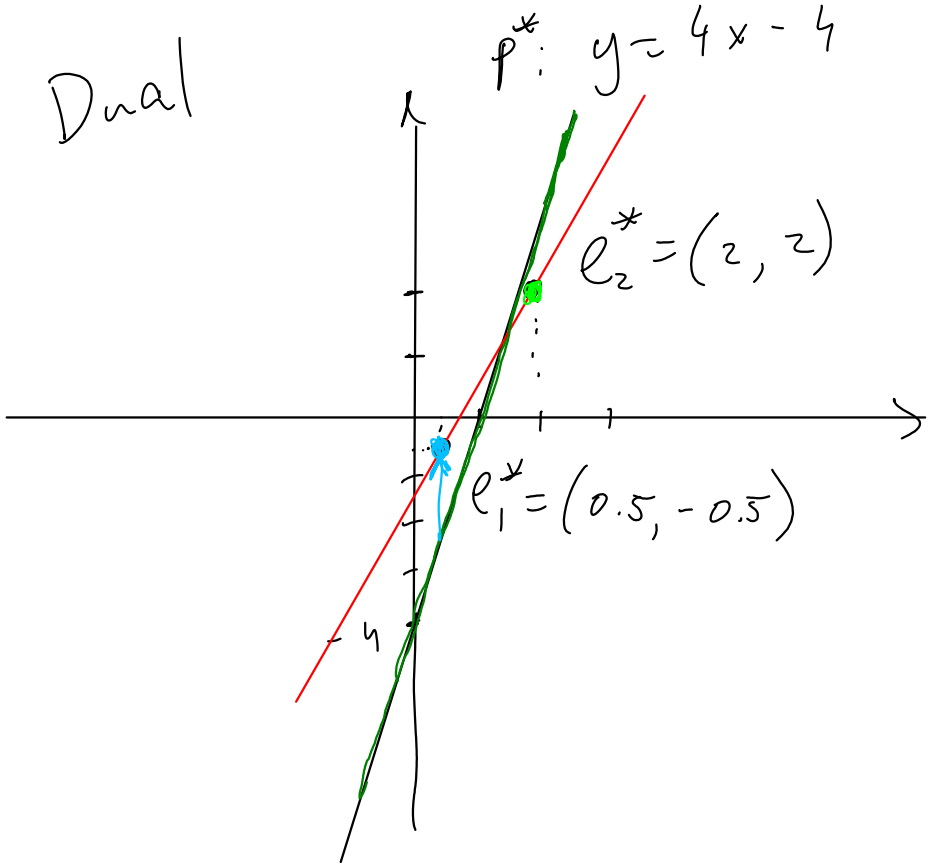
$\ell = (a, b)$ $\Longrightarrow$ $\ell^* = (a, -b)$

$y = ax + b$

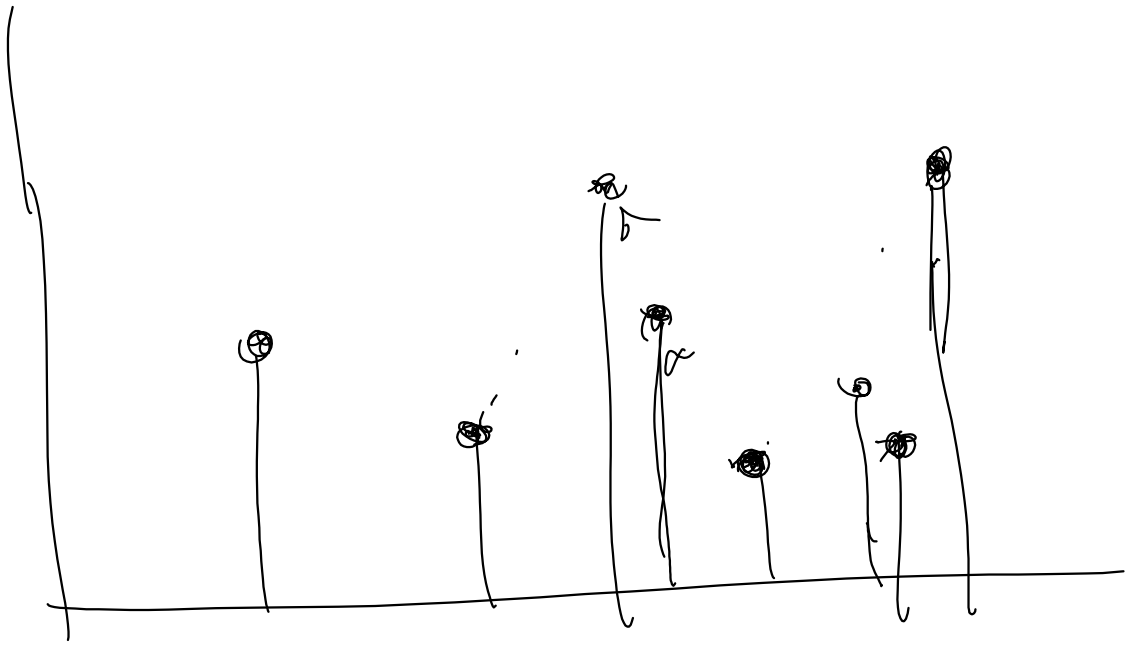
Primal



Dual



Given a point p & a line l in primal space, p lies above l iff p^* lies above l^* in dual space



E.g. visibility counting problem

point

$$\begin{pmatrix} a \\ b \end{pmatrix} \longleftrightarrow \begin{pmatrix} a \\ -b \end{pmatrix}$$

line

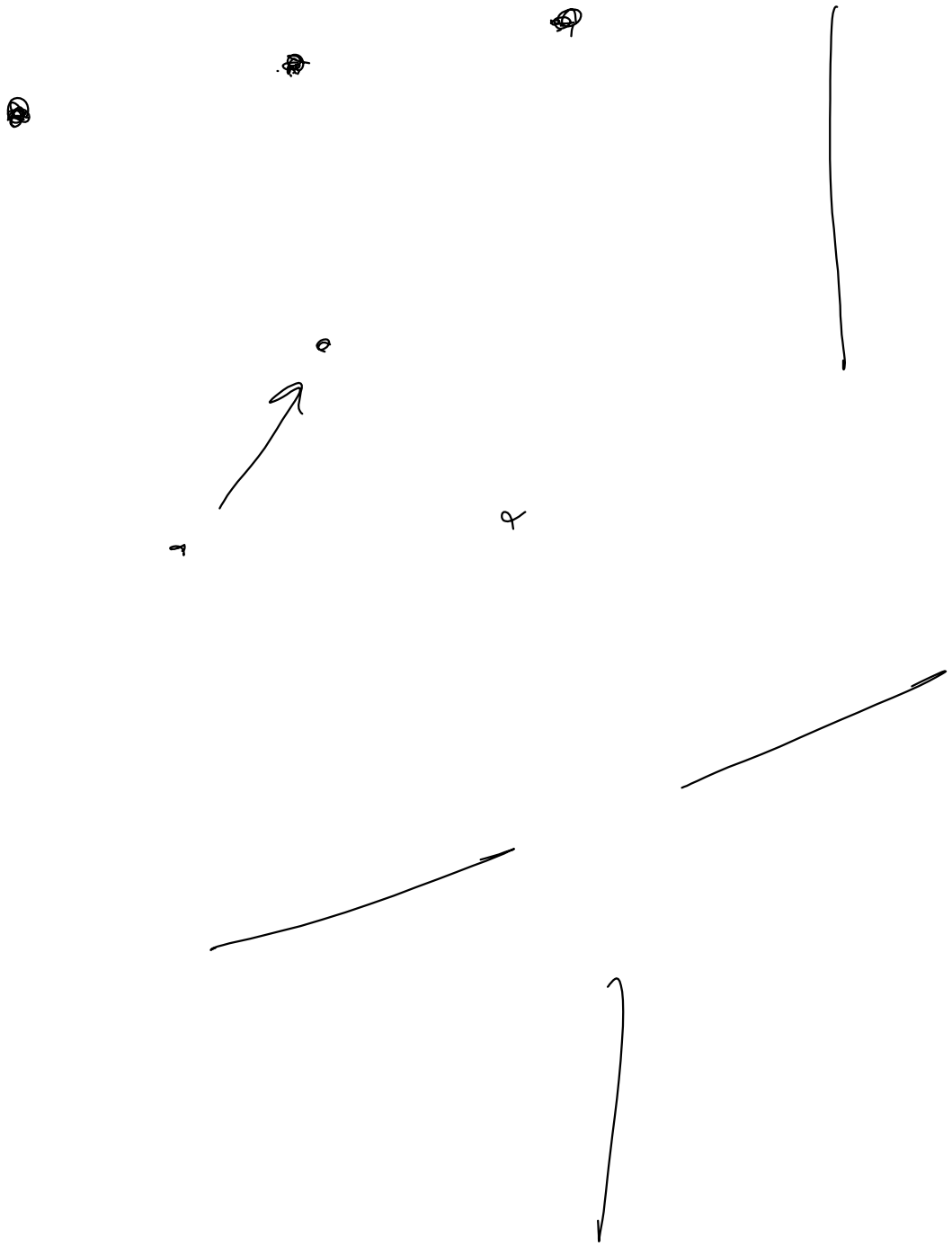
line

$$\begin{pmatrix} m \\ b \end{pmatrix} \longleftrightarrow \begin{pmatrix} m \\ -b \end{pmatrix}$$

point

General Position

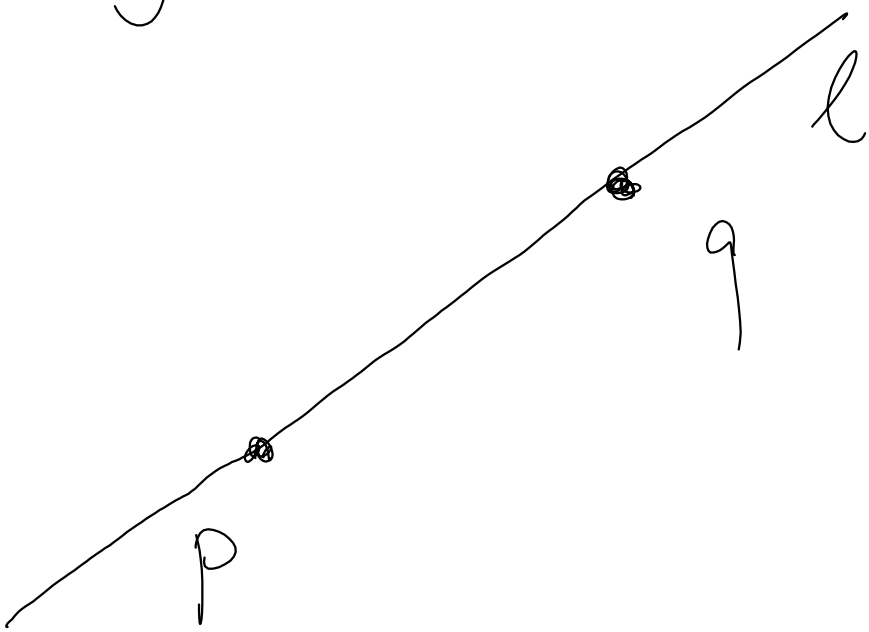
- No three points are co-linear



Given two points $p = (p_x, p_y)$

$q = (q_x, q_y)$.

They define a line



$\ell : y = mx + b$

$$\begin{cases} p_y = m p_x + b \\ q_y = m q_x + b \end{cases} \Rightarrow \begin{array}{l} b = p_y - m p_x \\ b = q_y - m q_x \end{array}$$

$$p_y - m p_x = q_y - m q_x$$

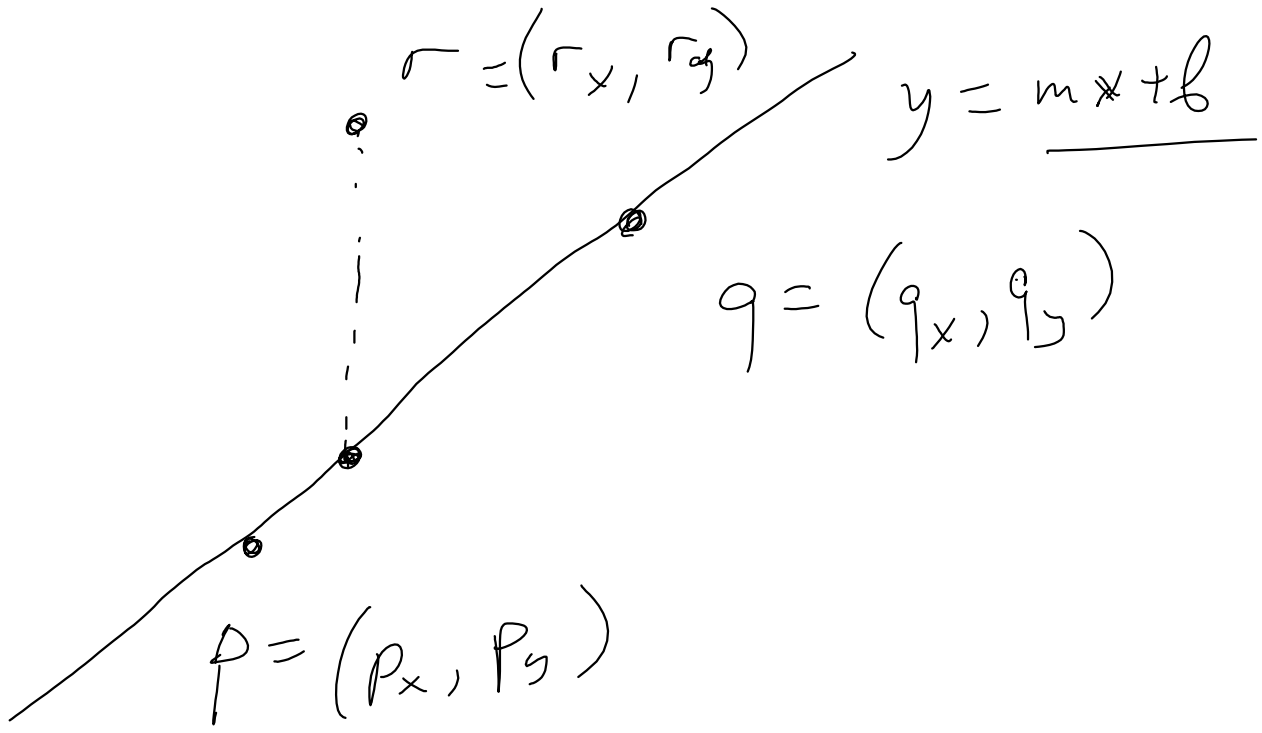
$$m(p_x - q_x) = p_y - q_y$$

$$m = \frac{p_y - q_y}{p_x - q_x}$$

$$b = p_y - \frac{p_y - q_y}{p_x - q_x} \cdot p_x$$

$$= \frac{\cancel{p_x p_y} - p_y q_x - \cancel{p_x p_y} + p_x q_y}{p_x - q_x}$$

$$= \frac{p_x q_y - p_y q_x}{p_x - q_x}$$



$$r_y \gtrless m \cdot r_x + b$$

$$\gtrless \frac{p_y - q_y}{p_x - q_x} \cdot r_x + \frac{p_x q_y - p_y q_x}{p_x - q_x}$$

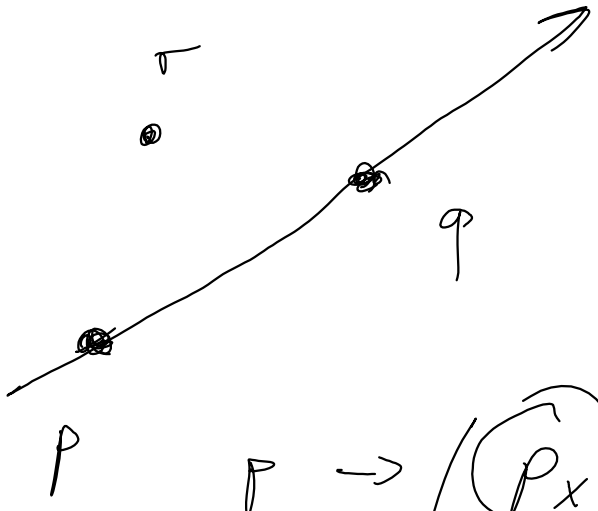
Robustness (rounding errors)

if $x == y \Rightarrow$ if $|x-y| \le \varepsilon$

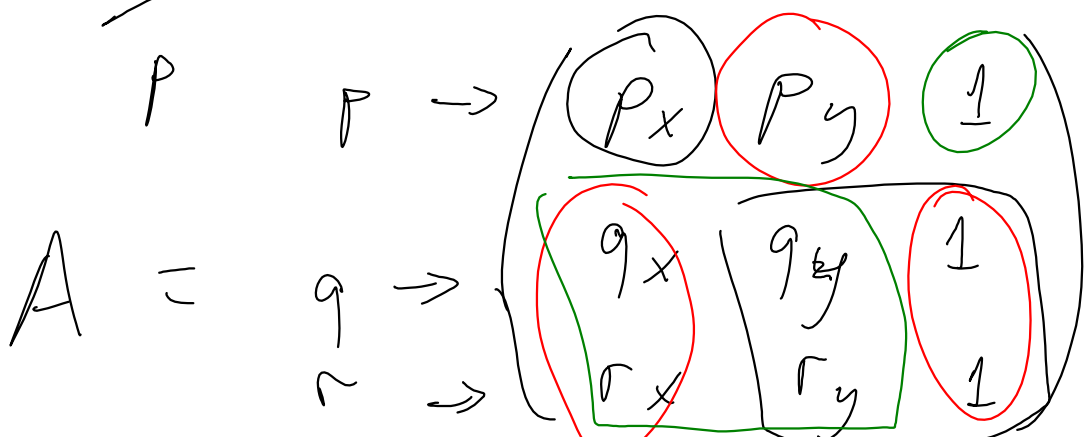
...

$$r_y(p_x - q_x) \gtrless (p_y - q_y) \cdot r_x + p_x q_y - p_y q_x$$

$\Leftrightarrow$ r is above / on / below $\overline{pq}$



$A =$



r is to the $\begin{matrix}\text{left}\\ \text{on}\\ \text{right}\end{matrix}$ of $\overrightarrow{pq}$

iff $|A| \begin{matrix}<\\ =\\ >\end{matrix} 0$

$$|A| = p_x \cdot \begin{vmatrix} q_y & 1 \\ r_y & 1 \end{vmatrix} - p_y \cdot \begin{vmatrix} q_x & 1 \\ r_x & 1 \end{vmatrix} + 1 \cdot \begin{vmatrix} q_x & q_y \\ r_x & r_y \end{vmatrix}$$

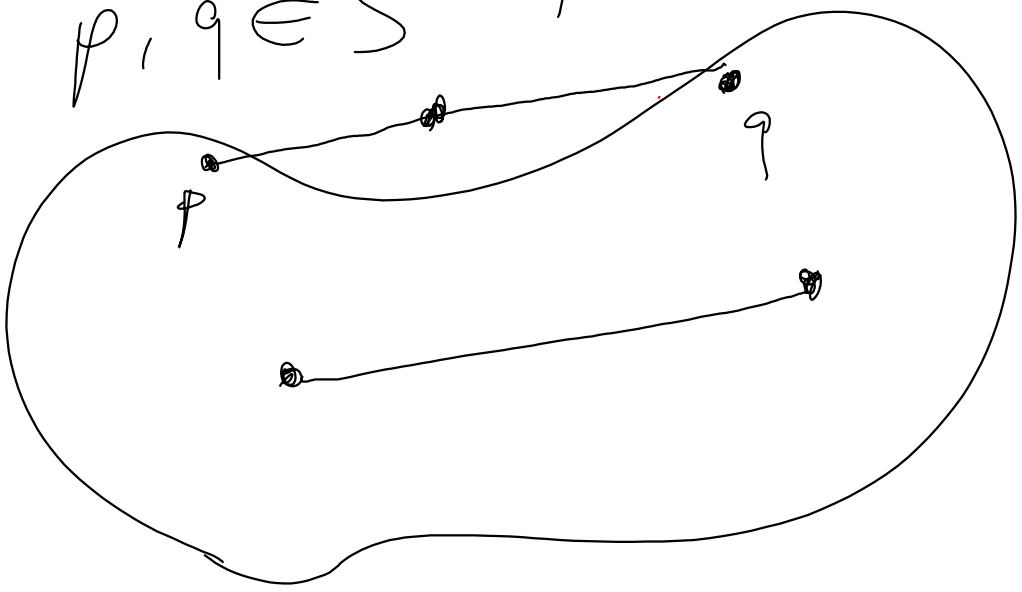
$$p_x(q_y - r_y) - p_y(q_x - r_x) + q_x r_y - q_y r_x$$

$$|A| = \begin{vmatrix} (p_x - r_x) & (p_y - r_y) \\ (q_x - r_x) & (q_y - r_y) \end{vmatrix}$$

$$= (p_x - r_x)(q_y - r_y) - (p_y - r_y)(q_x - r_x)$$

↑ ↑

Defn: An object S is convex iff for every pair of points $p, q \in S$ the segment $\overline{pq} \in S$



⇕

$\alpha p + (1-\alpha) q \in S$ for all $0 \le \alpha \le 1$

Given points $\vec{p}$ & $\vec{q}$

the line segment $\overline{pq}$

is a set of all points

defined by $\alpha\vec{p} + (1-\alpha)\vec{q}$

for all $0 \leq \alpha \leq 1$

$$p = \begin{pmatrix} p_x \\ p_y \end{pmatrix} \qquad q = \begin{pmatrix} p_y \\ q_y \end{pmatrix}$$

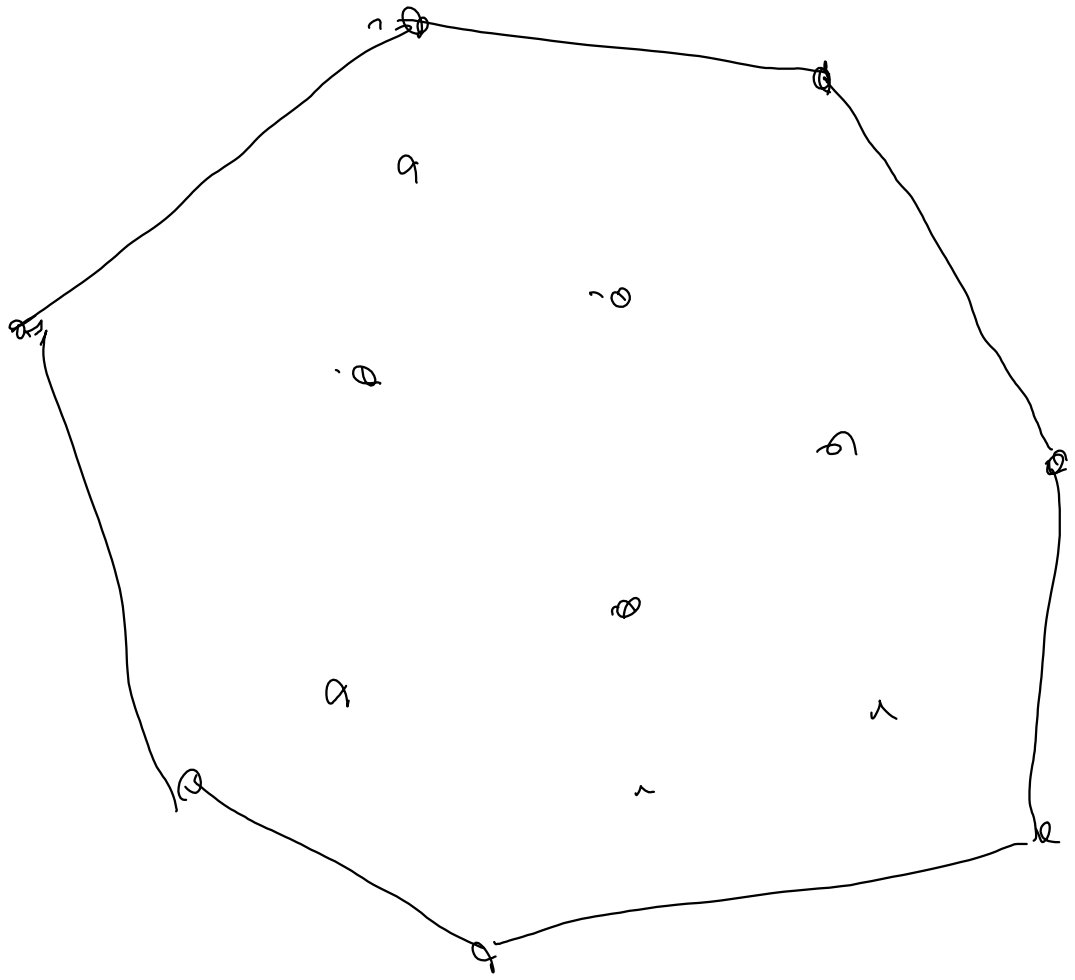
$$\alpha\vec{p} + (1-\alpha)\vec{q} =$$

$$\begin{pmatrix} \alpha p_x + (1-\alpha) q_x \\ \alpha p_y + (1-\alpha) q_y \end{pmatrix}$$

Def'n: (Convex Hull)

Given a set of points P in $\mathbb{R}^2$ a convex hull

- is a minimal convex set that contains P
- is an intersection of all convex sets that contain P.

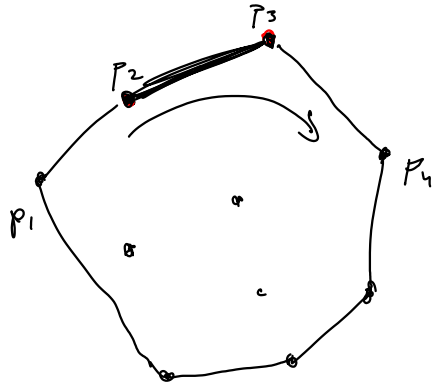


Computing convex hull:

Input: n points in $\mathbb{R}^2$

output: set of points on the boundary of convex hull in clockwise order

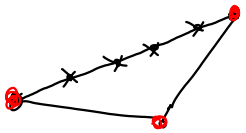
Convex Hull : Maximal Intersection of all half planes that contains all points



Convex Hull:

Input : set of points

Output : set of points on the convex hull in clockwise direction



Trivial alg:

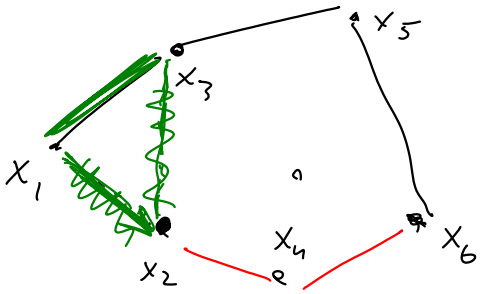
$O(n^3)$ { For every pair of points p_i, p_j : ← $O(n^2)$
→ if half planes H_{ij} or H_{ji} contain all $O(n)$ points, then include $\overline{p_i, p_j}$ into Convex Hull

$O(n^2)$ { Sort points on the Convex Hull in clockwise order

$(n-2) + (n-3) + (n-4) \ldots\ 1 = O(n^2)$

Convex Hull in $O(n \log n)$

using plane sweep technique



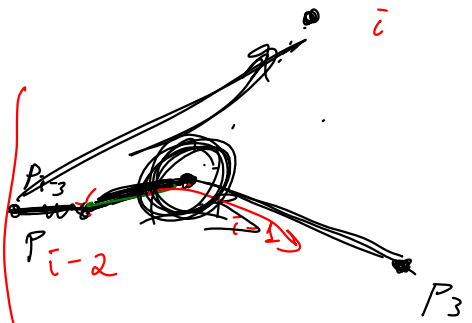
sort points by x-coord.

process these sorted points in this order

Upper Convex Hull :

→ Sort points by x-coordinate $(p_1, p_2, \ldots, p_n)$ (nlogn)

Add $\overrightarrow{p_1 p_2}$ to "convex Hull"



for $i = 3$ to n

$p = p_i$

~~if~~ while p lies to the left of $\overrightarrow{p_{i-2}\, p_{i-1}}$ then

remove p_{i-1} & add p_i

$S.push(p_1)$; $S.push(p_2)$

$\overrightarrow{p_{i-2}\, p_i}$

for $i = 3$ to n ≥

$p = p_i$

$p' = S.pop()$

$p'' = S.pop()$

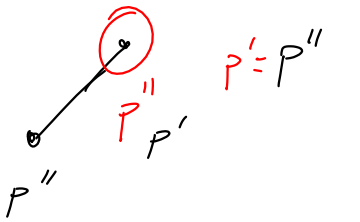
O(1)

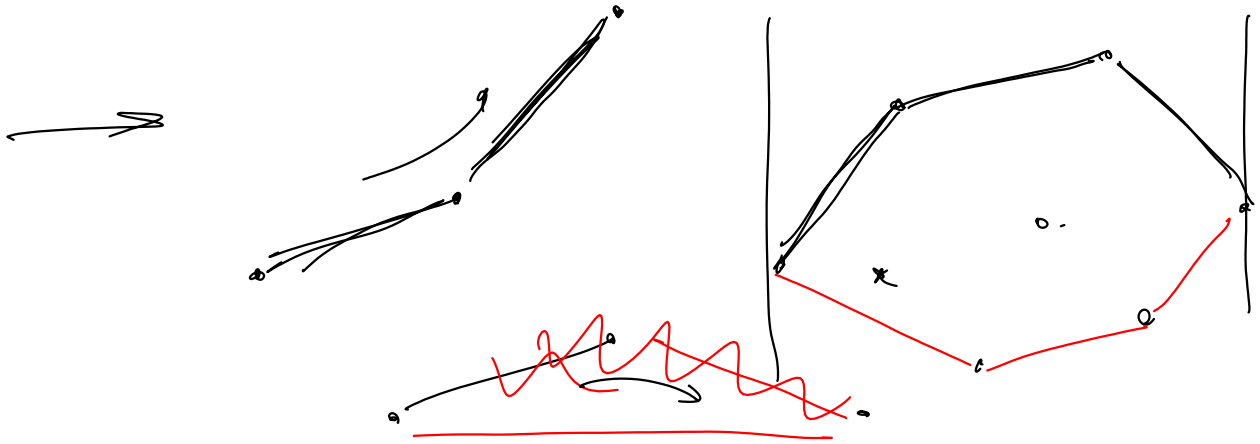
{ while p lies to the left of $\overrightarrow{p''p'}$

$p' = p''$

$p'' = S.pop()$ ≥

$S.push(p'')$; $S.push(p')$; $S.push(p)$





Loop Invariant Prior to i^{th} iteration, $= i+1$
Stack S contains all points of the upper convex Hull of points $P_1 \ldots P_{i-1}$

$P_{n+1-1} = P_n$

Termination: Stack S will contain all points of Upper convex Hull of points $P_1 \ldots P_n$

Maintenance:

Lower bound proof Convex Hull takes $\Omega(n \log n)$ time

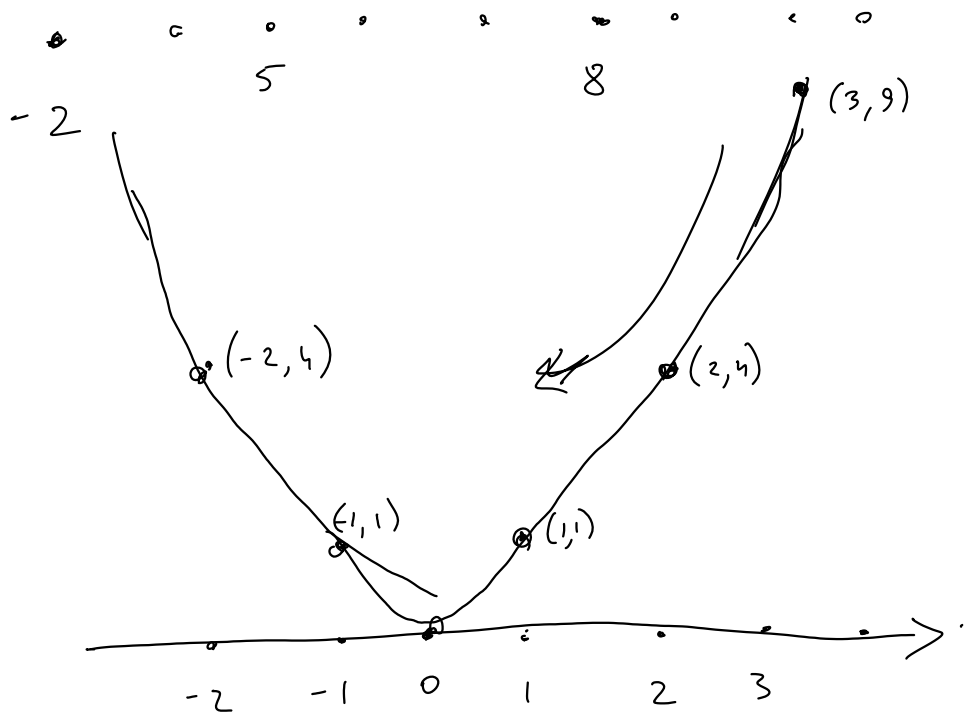
CH-sort (A[1 ... n])

```
for i = 1 to n
    P[i] = new Point( A[i], A[i].A[i] )
c = lower CH( P[1 ... n] )
```

for $i = n$ down to 1
    print $C[i].x$

$(5, 5^2)$

$(-2, (-2)^2)$

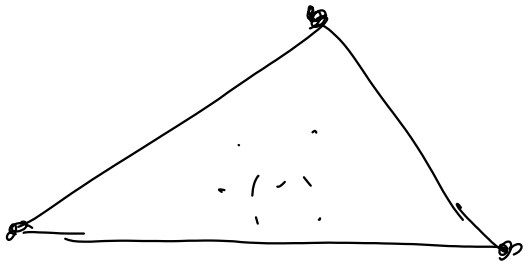


This proof is an example of reduction of sorting to convex hull

Best Convex Hull algorithm runs in

$O(n \log h)$ time,

where h is # of points on convex Hull.



output-sensitive algorithms