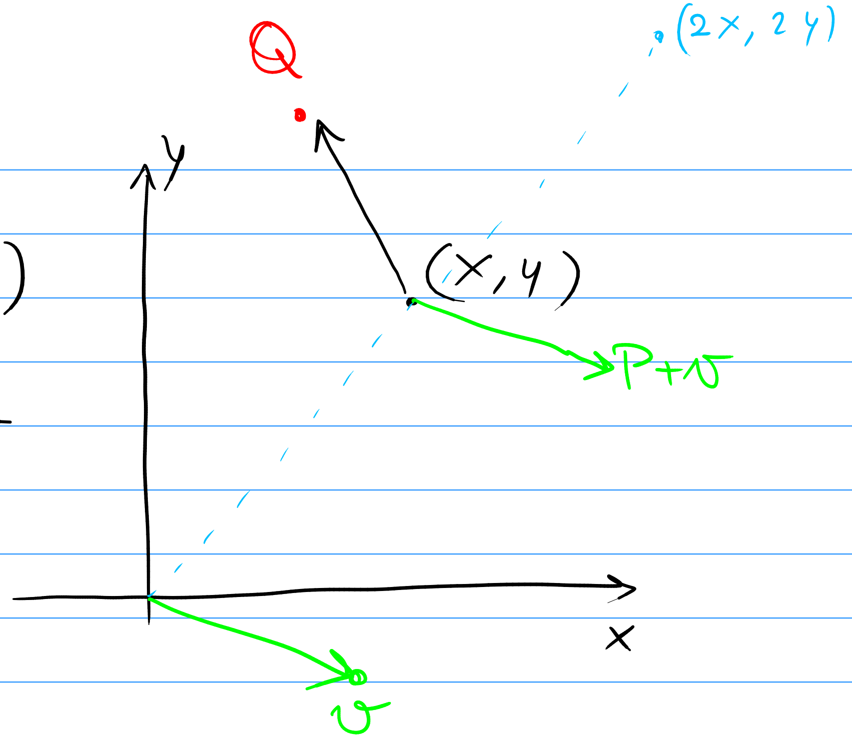
GEOMETRIA COMPUTAZIONALE.

Oggetto fondamentale: il punto P $(P.x, P.y)$

Identificato dalle sue coordinate cartesiane

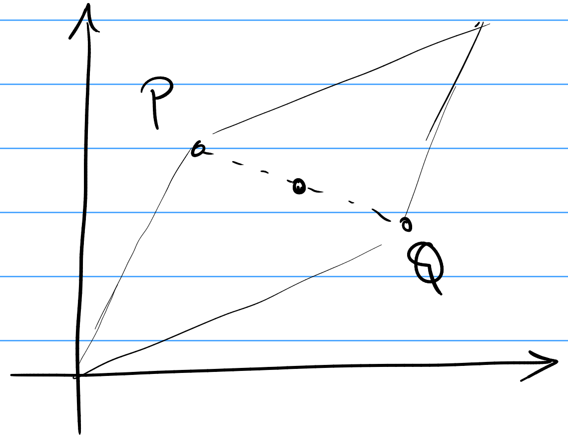


Riscalare un punto: $\lambda P = (\lambda P.x,\ \lambda P.y)$

Traslare un punto: $P+v = (P.x + v.x,\ P.y + v.y)$

Differenza tra due punti: $Q-P = (Q.x - P.x,\ Q.y - P.y)$

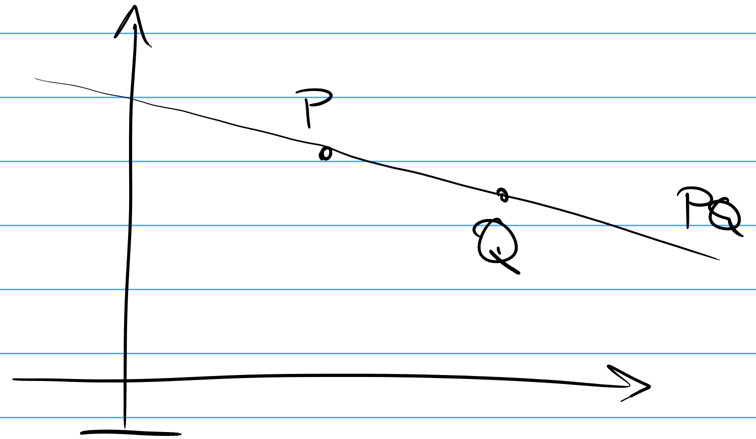
Trovare il punto medio tra P, Q: $\frac{P+Q}{2}$.



Oggetto Fondamentale: Retta

Una retta la identifichiamo con due punti. (≠ distinti)

(spesso rappresentata dall'equazione $y = ax + b$) ← male



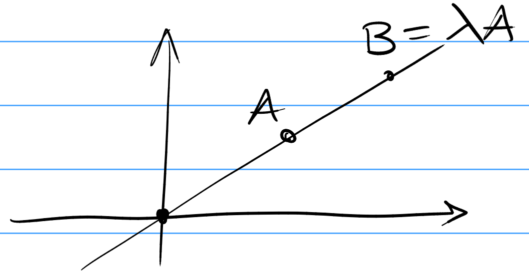
Domanda: Come capisco se il punto c sta sulla retta $\overline{AB}$.

Caso 1: $c = (0,0)$. $\overline{AB}$ contiene l'origine? Vero se $A = (0,0)$ o $B = (0,0)$.

Altrimenti serve che $B = \lambda A$.

$$\Updownarrow$$

$$\begin{cases} B.x = \lambda A.x \\ B.y = \lambda A.y \end{cases} \iff \begin{cases} \dfrac{B.x}{A.x} = \lambda \\ \dfrac{B.y}{A.y} = \lambda \end{cases} \iff \frac{B.x}{A.x} = \frac{B.y}{A.y} \iff \boxed{A.x \cdot B.y - A.y \cdot B.x = 0}$$



Definizione fondamentale: Prodotto vettore

$$A \wedge B := A.x \cdot B.y - A.y \cdot B.x .$$

Fatto: O, A, B allineati $\Leftrightarrow$ $A \wedge B = 0$

Fatto: A, B, C allineati $\Leftrightarrow$ traslo C in O $\quad A-C,\ B-C,\ \overset{O}{C - C}$ allineati

$$\Leftrightarrow (A-C) \wedge (B-C) = 0$$

nilpotente

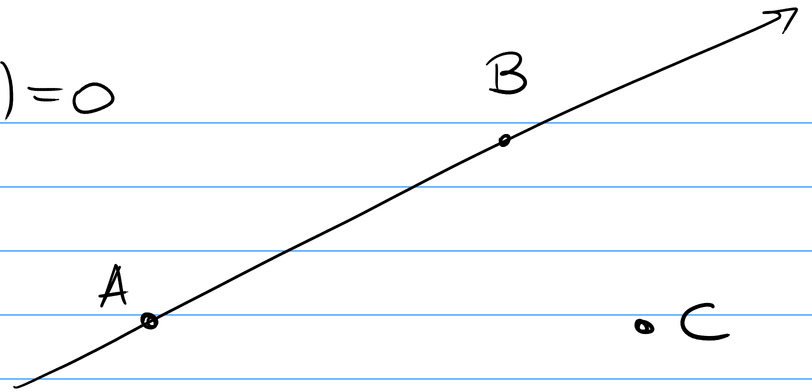
Proprietà facili: $A \wedge B = -B \wedge A$, $\quad A \wedge A = 0$, $\quad A \wedge (B+C) = (A \wedge B) + A \wedge C.$

Domanda fondamentale: Data una retta $\overrightarrow{AB}$ e un punto C,

il punto C sta sulla retta?

sta a dx della retta?

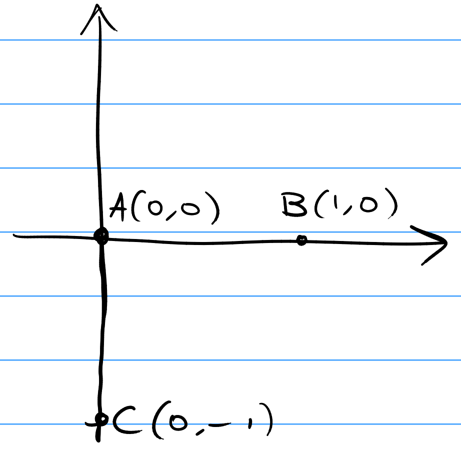
sta a sx della retta?



Fatto: C sta sulla retta $\Longleftrightarrow (A-C)\wedge(B-C)=0$

Assumiamo C stia a dx

Allora posso muovere i punti tenendo sempre C sulla dx in modo che alla fine



Osservazione: In ogni momento del movimento

$(A-C)\wedge(B-C)\neq 0 \Longrightarrow$ non può cambiare segno

All'inizio e alla fine il segno è lo stesso. Alla fine $(0,1)\wedge(1,1)=0\cdot 1-1\cdot 1=-1$

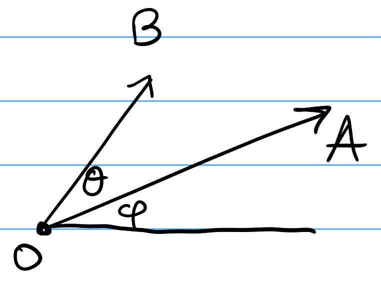
$\Longrightarrow \boxed{(A-C)\wedge(B-C)<0}$ all'inizio.

Se C è a sx $\Longrightarrow (A-C)\wedge(B-C)>0$.

OSSERVAZIONE CRUCIALE: $\mathrm{sgn}((A-C)\wedge(B-C))$ ci dice se C è a dx/sx della retta $\vec{AB}$.

Definizione "alternativa" del prodotto vettore: $\left(|P| = \sqrt{P.x^2 + P.y^2}\right)$

$$A = (|A|\cos\varphi,\ |A|\sin\varphi)$$
$$B = (|B|\cos(\theta+\varphi),\ |B|\sin(\theta+\varphi))$$

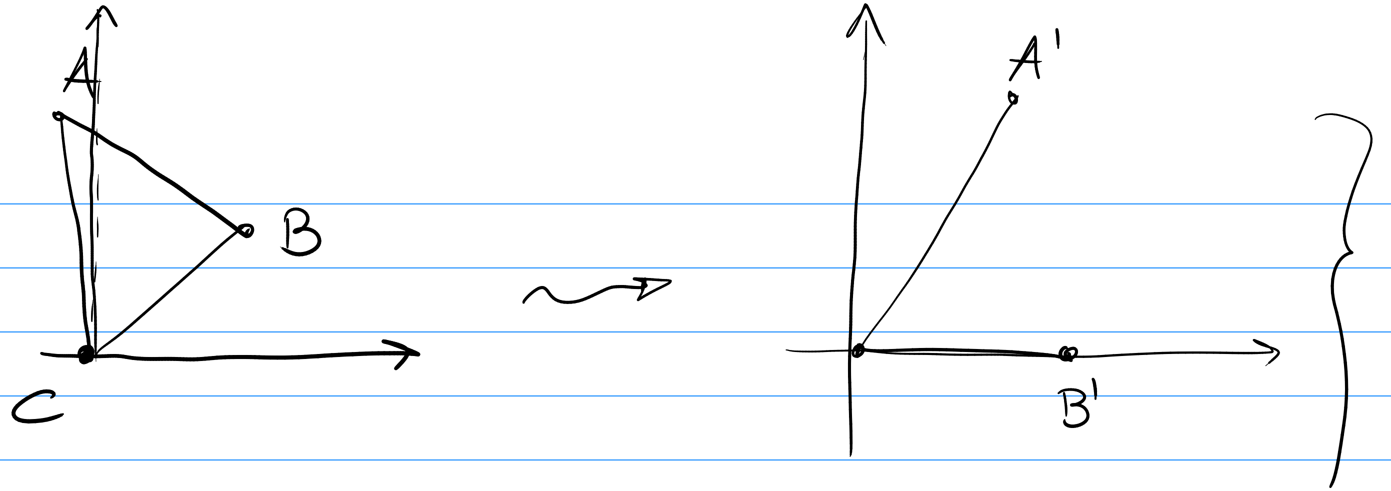


$$A\wedge B = |A|\cdot|B|\,(\cos\varphi\,\sin(\theta+\varphi) - \sin\varphi\,\cos(\theta+\varphi))$$
$$= \boxed{|A|\cdot|B|\,\sin(\theta).}$$

(quindi l'angolo tra A, B è $\mathrm{asin}\left(\frac{A\wedge B}{|A|\cdot|B|}\right)$.)

Di conseguenza il prodotto vettore non cambia se ruoto entrambi i punti

Area di un triangolo:



$$\text{Area}(A\overset{\triangle}{B}C) = \text{Area}\left((A-C)\overset{\triangle}{(B-C)}(0)\right)$$

$$= \text{Area}(A'B'O) = \pm\frac{B'.x \cdot A'.y}{2} = \pm\frac{B' \wedge A'}{2} = \pm\frac{(B-C)\wedge(A-C)}{2}$$

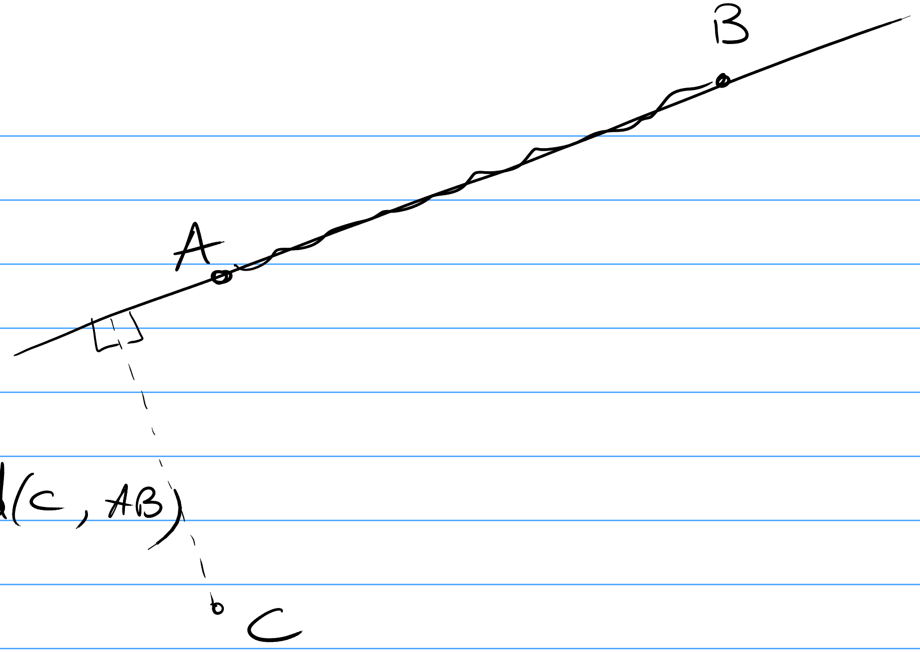
$\Rightarrow$ L'area di $A\overset{\triangle}{B}C$ è $\left|\frac{(B-C)\wedge(A-C)}{2}\right|$. (uguale scambiando A, B / B, C / C, A cambiando ordine)

Osservazione: Se non metto il valore assoluto trovo l'area segnata

$+A$ se A, B, C sono in ordine orario

$-A$ se A, B, C // // antiorario.

Distanza da una retta:



$$\underbrace{\text{Area}(A\hat{B}C) = \frac{|B-A| \cdot d(C,AB)}{2}}$$

So tutto tranne $d(C,AB) \Rightarrow$

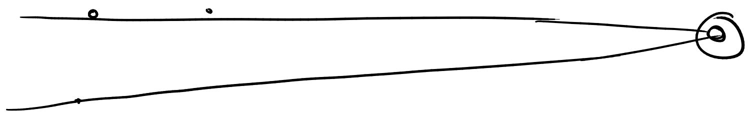
ho una formula per $d(C,AB)$

Remark: L'area di un triangolo a coordinate intere è $\frac{\text{intero}}{2}$.

In generale la proprietà bella del prodotto vettore è che è un intero se i pt sono a coordinate intere.

Se uso come i floating point (in C++ float, double)

Si incappa subito in problemi di precisione.



Obiettivo: Restare sempre negli interi.

Remark: Se "non si hanno cerchi", tutti i conti restano nei razionali

ma potrebbero esplodere i numeri coinvolti.

$$\frac{10^8}{10^9+1} + \frac{10^8+1}{10^9+2} = \frac{10^?}{10^{18}\dots}$$

Area di un poligono:

Area $(P_0 P_1 \dots P_{n-1}) =$

$$\frac{1}{2}\left| P_0 \wedge P_1 + P_1 \wedge P_2 + P_2 \wedge P_3 + \dots + P_{n-2} \wedge P_{n-1} + P_{n-1} \wedge P_0 \right|$$

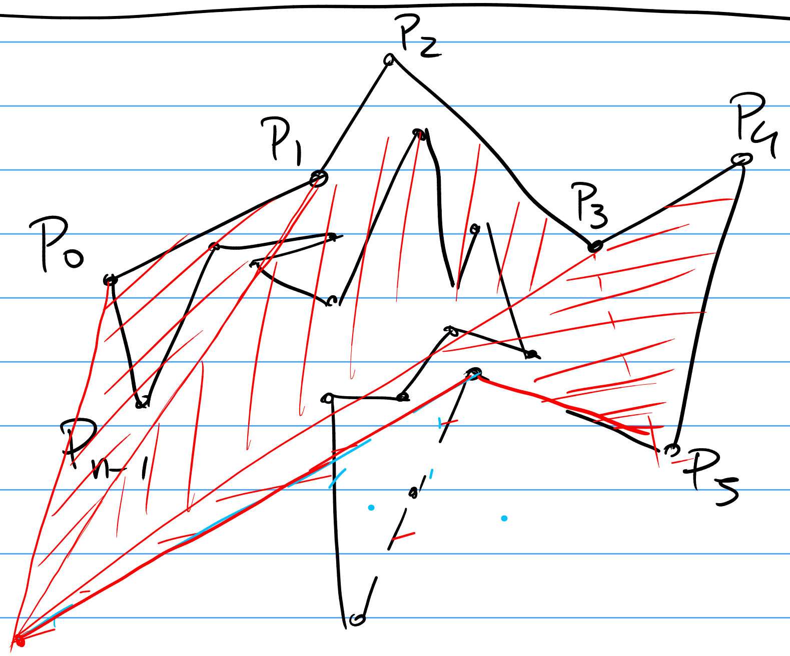
pollice mano dx

indice mano dx

il segno è

+ se il palmo è in alto

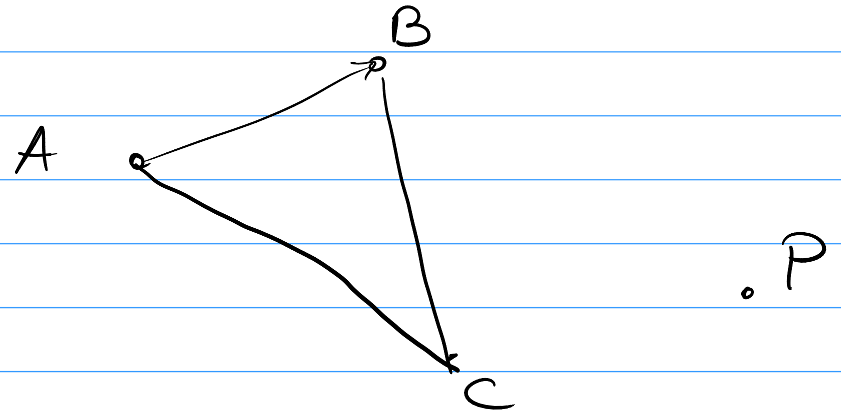
− se il palmo è in basso



Capire se un punto e' dentro o fuori da un poligono.

Caso facile: Poligono e' triangolo

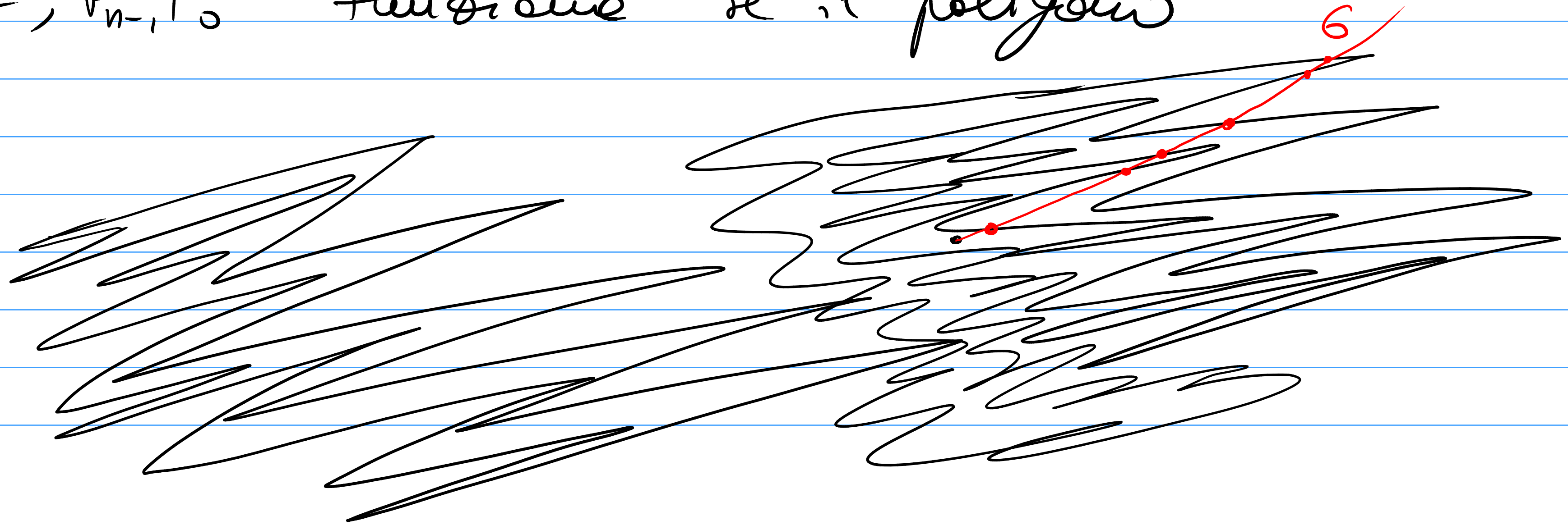
Controllare se P sta dalla stessa parte di AB, BC, CA.



Osservazione: Il trick di guardare che stia dalla stessa parte di P_0P_1, P_1P_2, P_2P_3, ---, $P_{n-1}P_0$ funziona se il poligono e' convesso.

Poligono generico:

Osservazione: Il problema può essere non banale anche per un umano.



Trick: Disegno una semiretta che parte dal punto P e conto le intersezioni con il poligono.

Se # e' pari ⟹ sono fuori

Se # e' dispari ⟹ sono dentro.

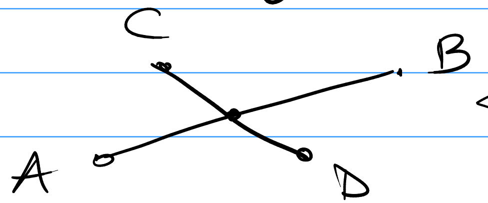
ACHTUNG: La semiretta NON deve contenere vertici.

Come posso trovare una semiretta che non contiene vertici?

Scelgo Q a caso e grande e considero il segmento $\overline{PQ}$.

Resta da capire se $\overline{PQ}$ interseca $\overline{P_i P_{i+1}}$.

Sottoproblema generico: Dati due segmenti AB, CD: si intersecano?



⟺ $\begin{cases} \text{C e D da parti opposte di AB.} \\ \text{A e B da parti opposte di CD} \end{cases}$

OSSERVAZIONE GENERALE: Spesso i problemi di geometria sono noiosi da scrivere perché bisogna gestire i casi "degeneri" (allineamenti, punti che coincidono, parallelismi, punti con coordinate uguali ...).

↗ Tengo conto di tutti i casi

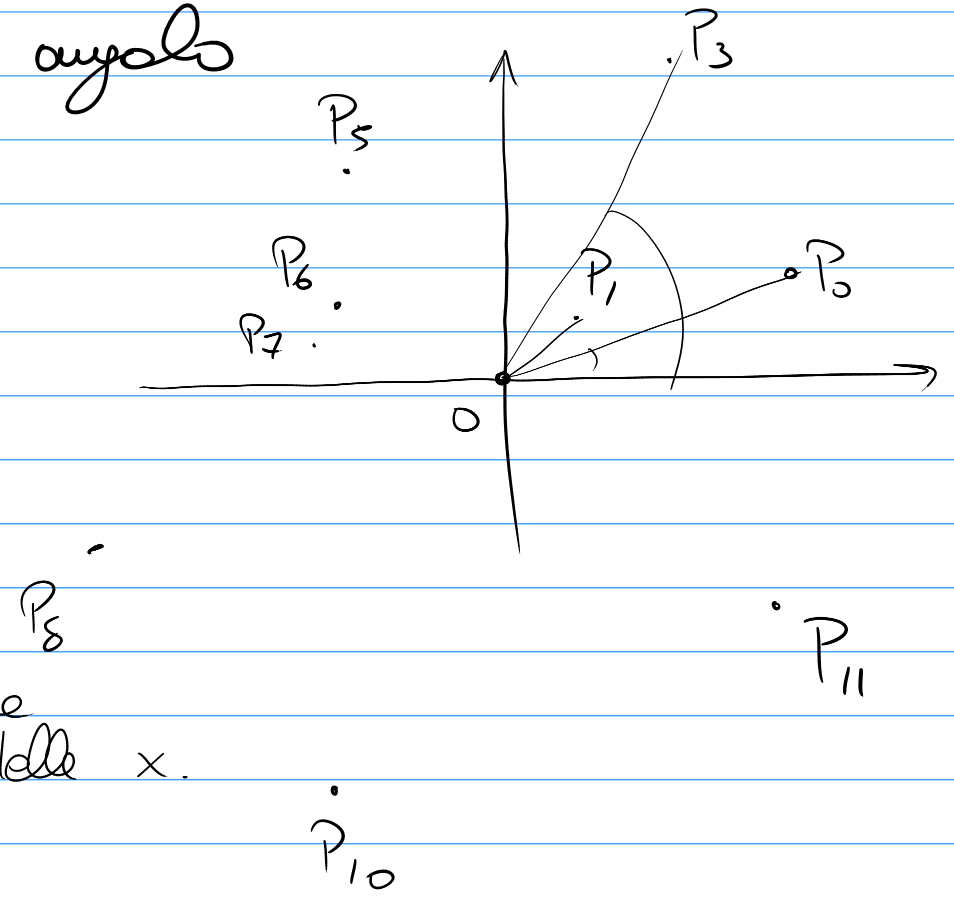
↘ Perturbo il problema per ottenere genericità.

Esempio: L'appartenenza ad un poligono è invariante per applicazioni lineari

$$(x,y) \mapsto (5x+10y,\ 21x+300y)$$

Esempio: Quando devo scegliere un punto ⇒ spesso conviene sceglierlo grande a caso.

SORT BY ANGLE: Vi dò n punti (distinti da O) e vi chiedo di ordinarli per angolo

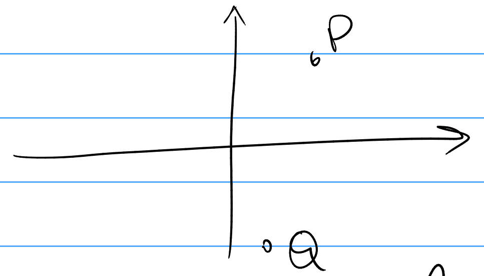


Modo 1: Calcolo l'angolo $\operatorname{atan2}(y, x) = \theta$.

Problema: Questo è lento e impreciso.

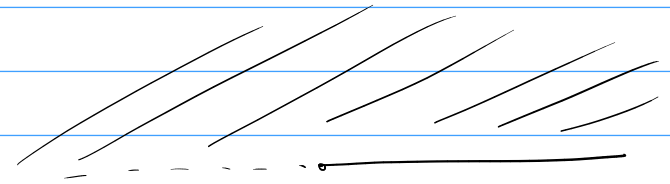
Modo 2: Uso sort e mi riduco al problema con due punti P, Q.

Caso 1: P e Q sono uno sopra e l'altro sotto l'asse delle x.

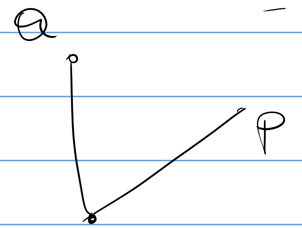


Nel caso, quello che sta sopra viene prima.

Più precisamente, P sta sopra $(P.y > 0)$ or $(P.y == 0$ and $P.x > 0)$



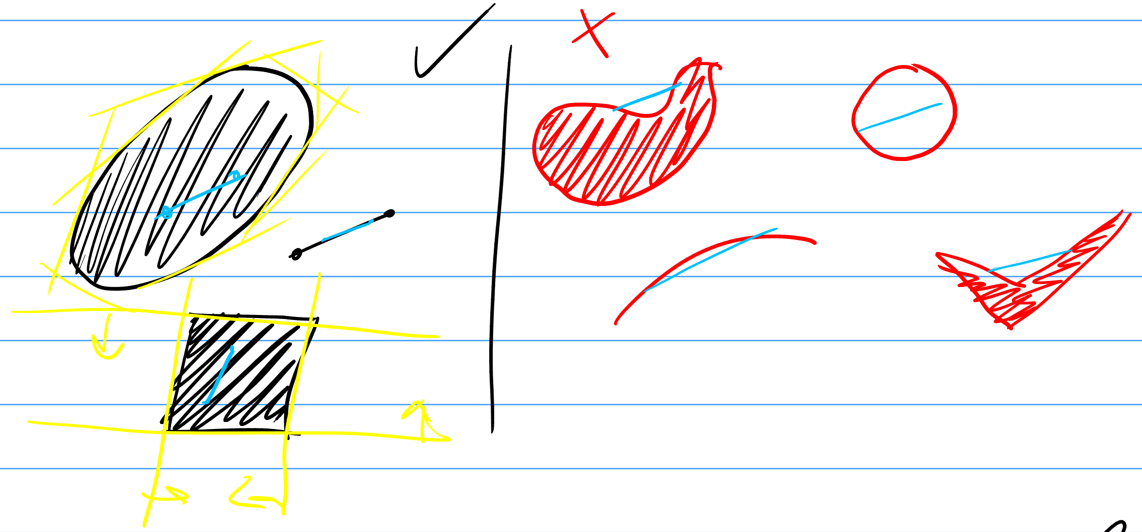
<u>Caso 2</u>: P, Q stanno dalla stessa parte.



Allora P viene prima di Q $\iff$ Q è a sx della retta $\overrightarrow{OP}$. ← lo so fare col prodotto vettore

Convex - Hull:

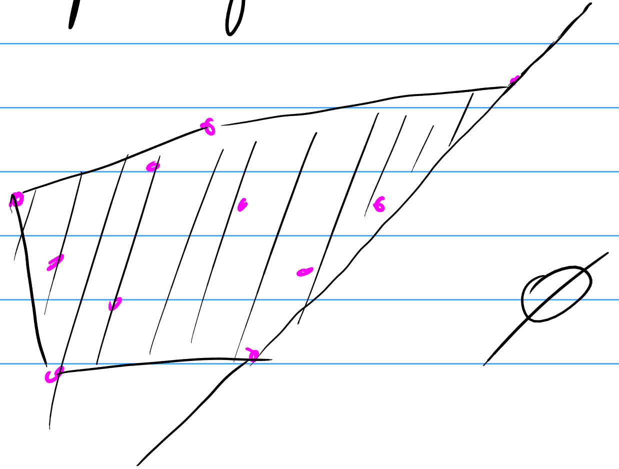
Un insieme è convesso se quando contiene P, Q $\Rightarrow$ contiene anche il segmento che li unisce



Remark: Un insieme convesso si scrive come intersezione (infinite) di semipiani

Il convex-hull di n-punti è il più piccolo insieme convesso che li contiene.

Esempio:



Remark. Il convex-hull è un poligono con vertici negli n punti.

Come trovo il convex-hull?

<u>Lentamente è facile:</u> Per ogni coppia di punti iniziali PQ, guardo se gli altri punti stanno dalla stessa parte, nel caso PQ è un lato del convex-hull
$O(n^3)$

<u>Andrew's monotone chain algorithm $O(n \log(n))$.</u>

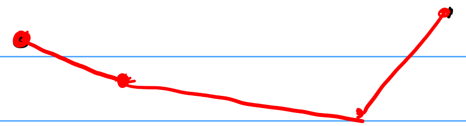
Costruire la parte di sotto (e sopra) del convex-hull aggiungendo punti con x che aumenta.

<u>Step 0:</u> Ordiniamo gli n punti per x. (e y poi)

<u>Step 1:</u> Inizialmente nel convex-hull ci sta solo il primo punto

<u>Step 2:</u> Aggiungete un punto alla volta e aggiornate il "sotto convex-hull".

Step 3: Alla fine avrò il sotto convex-hull.



Stessa storia per il sopra convex-hull } l'unione (ignorando i punti estremi) e' il convex-hull.

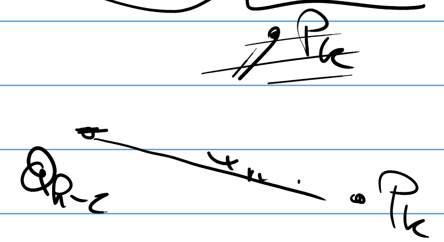
In ogni momento ho processato $P_0, P_1, \dots, P_{k-1}$

E il sotto convex-hull e' $Q_0, Q_1, \dots, Q_{h-1}$

Arriva P_k → lo metto nel sotto convex-hull $Q_0, Q_1, \dots, Q_{h-2}, Q_{h-1}, P_k$

Ma P_k deve stare a sx di $Q_{h-2} Q_{h-1}$.

Altrimenti elimino Q_{h-1}, e ripeto :)



Complessità: $O(n \log(n))$ dovuta al sort iniziale.

Sweep-line method: Processare i punti rispetto alla coordinata x, come se una retta parallela all'asse y si stesse muovendo da sx a dx.

Esempio: Calcolare l'area dell'unione di n rettangoli (in $n \log(n)$)

Calcolare l'area dell'unione di n triangoli (in $O(n^2)$)

Trovare la coppia di punti a distanza minima.