

SÉANCE 4 | TÉLÉCODE

GÉOMÉTRIE

ALGORITHMIQUE



PLAN

01

INTRODUCTION

02

CROSS PRODUCT

03

POSITION D'UN POINT PAR
RAPPORT À UNE DROITE

04

IMPLÉMENTATION EN PYTHON

05

INTERSECTION DE SEGMENTS

06

PROBLÈMES

1. INTRODUCTION

- Géométrie
 - Toujours travailler avec des **entiers** (sinon problèmes d'approximation)
 - Souvent en 2D
- Traite de différents sujets :
 - intersection de segments
 - aires d'un polygone
 - convexité (enveloppe convexe)



2. PRODUIT VECTORIEL

- 2 vecteurs $u(ux, uy, 0)$ et $v(vx, vy, 0)$:
 - $w = u \times v = (ux.vy - uy.vx).z$
 - $w = \|u\|.\|v\|.sin(u, v).z$
- Donc le signe de w donne l'angle (u, v) !
- A partir de $sin(u, v)$, on peut savoir de quel côté est situé un point par rapport à une droite



2. PRODUIT VECTORIEL

- Ne pas retenir les formules !
 - $w = u \times v = (ux.vy - uy.vx).z$
 - $w = \|u\|.\|v\|.sin(u, v).z$
- Savoir qu'on l'exprime de 2 manières (une via coordonnées et une autre avec l'angle) et que le produit vectoriel est anti-commutatif



3. POSITION D'UN POINT PAR RAPPORT À UNE DROITE

- Données : 3 points A, B, C et leurs coordonnées (x, y)
- Problème : déterminer si c est à gauche, à droite ou sur la droite (a, b)
- Idée : considérer l'**angle entre les vecteurs AB et BC**
- (NB : on retrouve cette idée dans la recherche de l'enveloppe convexe)



4. IMPLÉMENTATION EN PYTHON

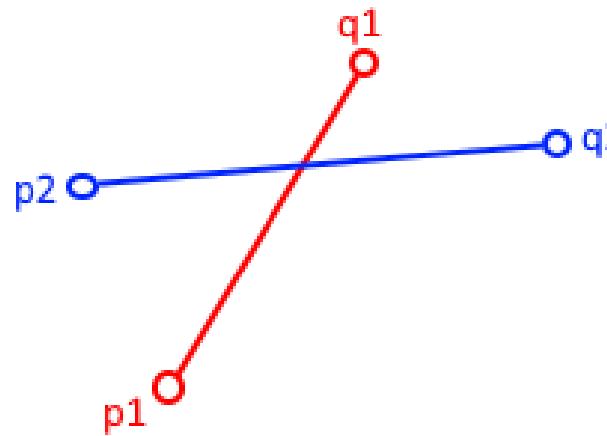
```
class Pt :  
  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y  
  
    def __sub__(self, p) :  
        return Pt(self.x-p.x, self.y-p.y)  
  
    def cross(self, p) :  
        return self.x * p.y - self.y * p.x  
  
    def position(a, b, c) :  
        # renvoie la position du point c par rapport à la droite (a, b)  
        # 1 si gauche, -1 si droite, 0 si colinéaire  
        u = b-a  
        v = c-b  
        return sgn(u.cross(v))
```

5. INTERSECTION DE SEGMENTS

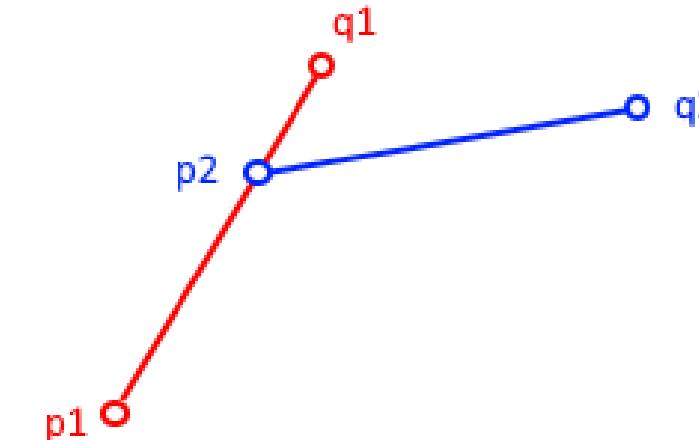
- Données : 4 points a, b, c, d (deux segments $[a, b]$ et $[c, d]$)
- Problème : déterminer si les segments $[a, b]$ et $[c, d]$ se croisent
- Idée :
 - regarder comment on pourrait caractériser l'intersection de segments avec l'algorithme précédent (faire des dessins)



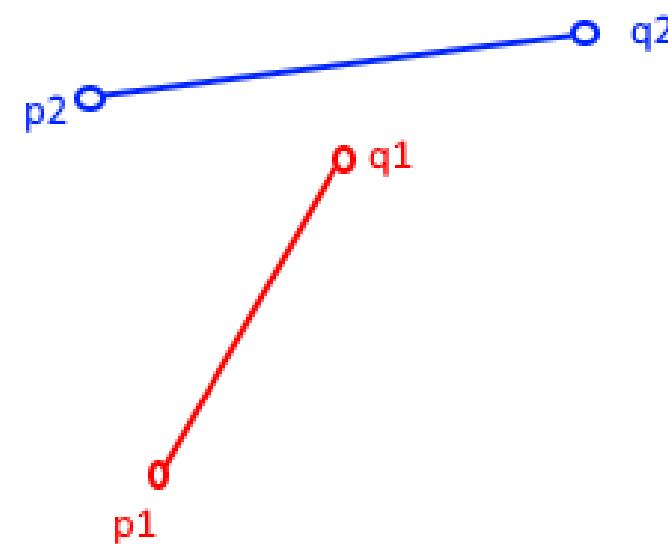
5. INTERSECTION DE SEGMENTS



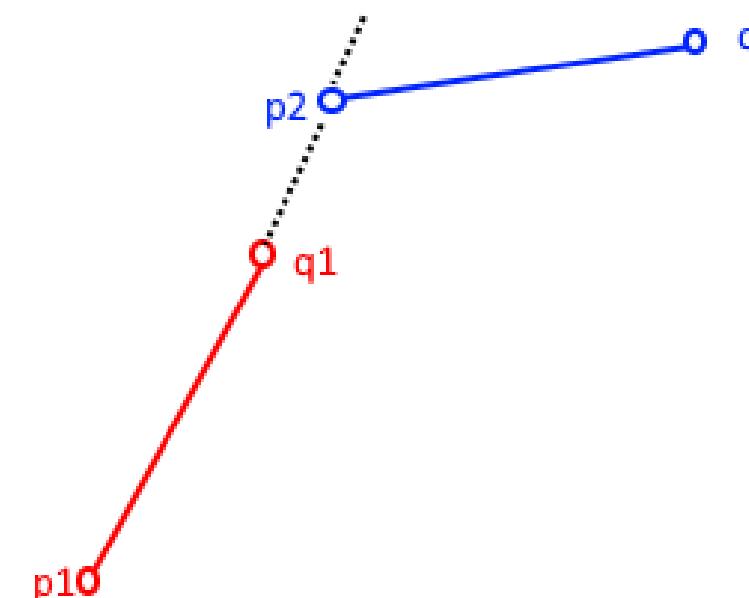
Example : Orientations of (p_1, q_1, p_2) and (p_1, q_1, q_2) are different. Orientations of (p_2, q_2, p_1) and (p_2, q_2, q_1) also different.



Example: Orientations of (p_1, q_1, p_2) and (p_1, q_1, q_2) are different. Orientations of (p_2, q_2, p_1) and (p_2, q_2, q_1) also different



Example: Orientations of (p_1, q_1, p_2) and (p_1, q_1, q_2) are different. Orientations of (p_2, q_2, p_1) and (p_2, q_2, q_1) are same



Example: Orientations of (p_1, q_1, p_2) and (p_1, q_1, q_2) are different. Orientations of (p_2, q_2, p_1) and (p_2, q_2, q_1) are same.

5. INTERSECTION DE SEGMENTS

- On considère $[a, b]$
 - il faut: $\text{orientation}(c) \neq \text{orientation}(d)$
- On considère $[c, d]$
 - il faut : $\text{orientation}(a) \neq \text{orientation}(b)$
- Il reste un cas embêtants à gérer :
 - a, b, c, d colinéaires
 - Considérer alors l'ordre des points



6. PROBLÈMES

CSES :

1. (géometrie) Point location test
2. (géometrie) Line segment intersection
3. (géométrie) Polygon area

Codeforces :

Black circles : 2002/C

Weird sum : 1648/A

Shawarma tent : 1271/C