



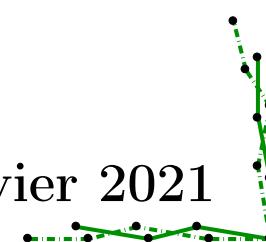
Conception et Pratique de l'Algorithmique

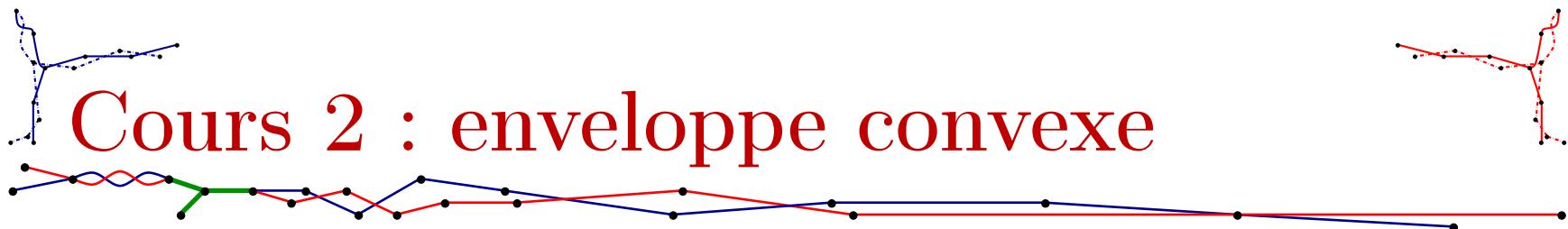
<http://www-apr.lip6.fr/~buixuan/cpaad2020>

Binh-Minh Bui-Xuan



PARIS, Janvier 2021





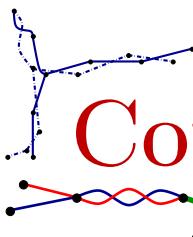
Cours 2 : enveloppe convexe

RAPPEL COURS + TME 1 :

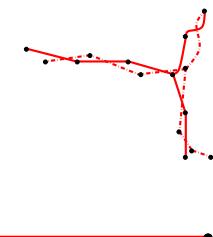
- collision : cas le plus simple est la collision des cercles
- problème CERCLEMIN : nuage de points \rightarrow cercle
- algorithme naïf : complexité $O(n^4)$
- techniques : incrémental (Ritter), précalcul (Akl-Toussaint)



1



Cours 2 : enveloppe convexe



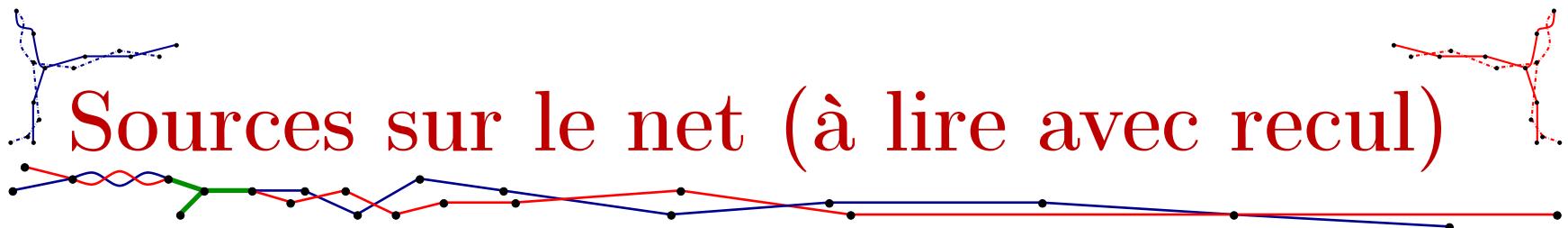
RAPPEL COURS + TME 1 :

- collision : cas le plus simple est la collision des cercles
- problème CERCLEMIN : nuage de points → cercle
- algorithme naïf : complexité $O(n^4)$
- techniques : incrémental (Ritter), précalcul (Akl-Toussaint)

AUJOURD'HUI :

- collision : cas de polygones convexes (esthétique !)
- problème ENVCONVEXE : nuage de points → polygone convexe
- algorithme naïf : complexité $O(n^3)$
- techniques : précalcul (pixel, Akl-Toussaint), décomposition
- algorithmes : Jarvis, Graham (+variants), Chan, QuickHull





Sources sur le net (à lire avec recul)

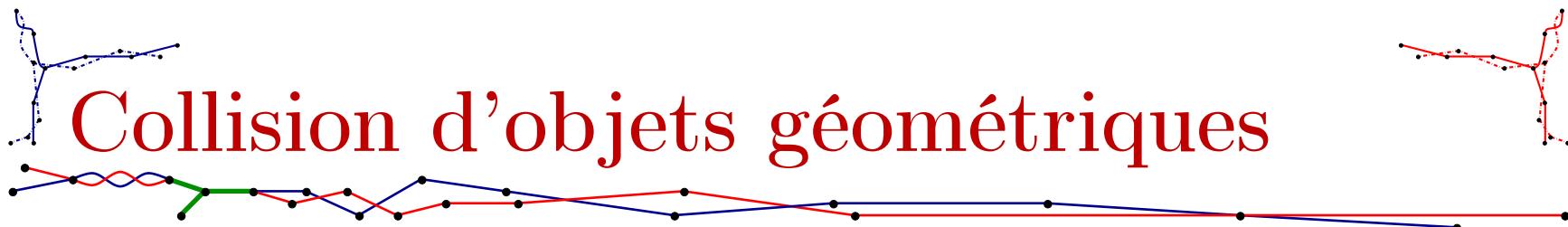
GAME DEVELOPMENT :

- ici: <https://gamedevelopment.tutsplus.com/tutorials/collision-detection-using-the-separating-axis-theorem--gamedev-169>

WIKIPEDIA :

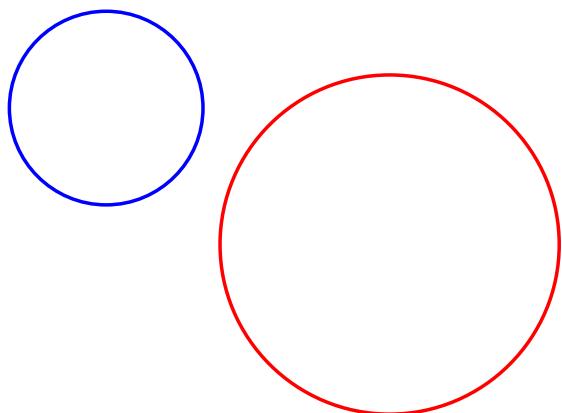
- http://en.wikipedia.org/wiki/Convex_hull_algorithms



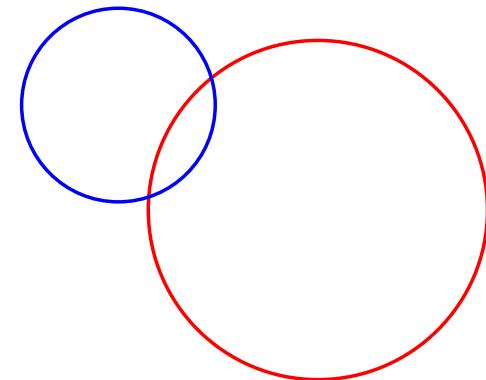


QUESTION : touché ?

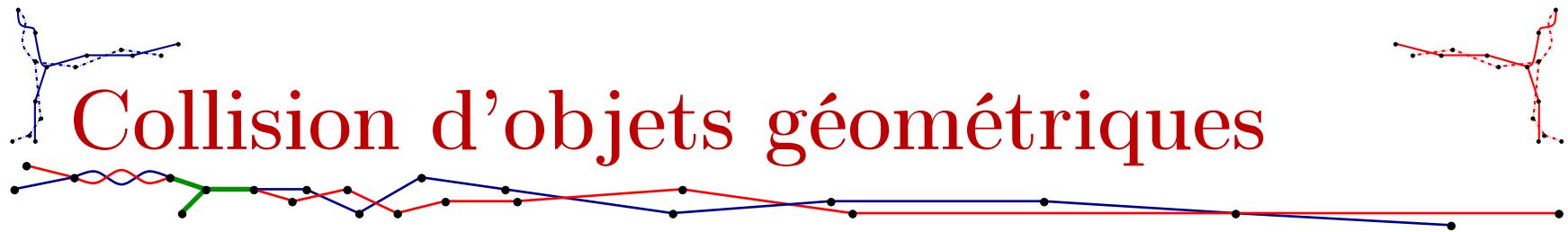
False



True

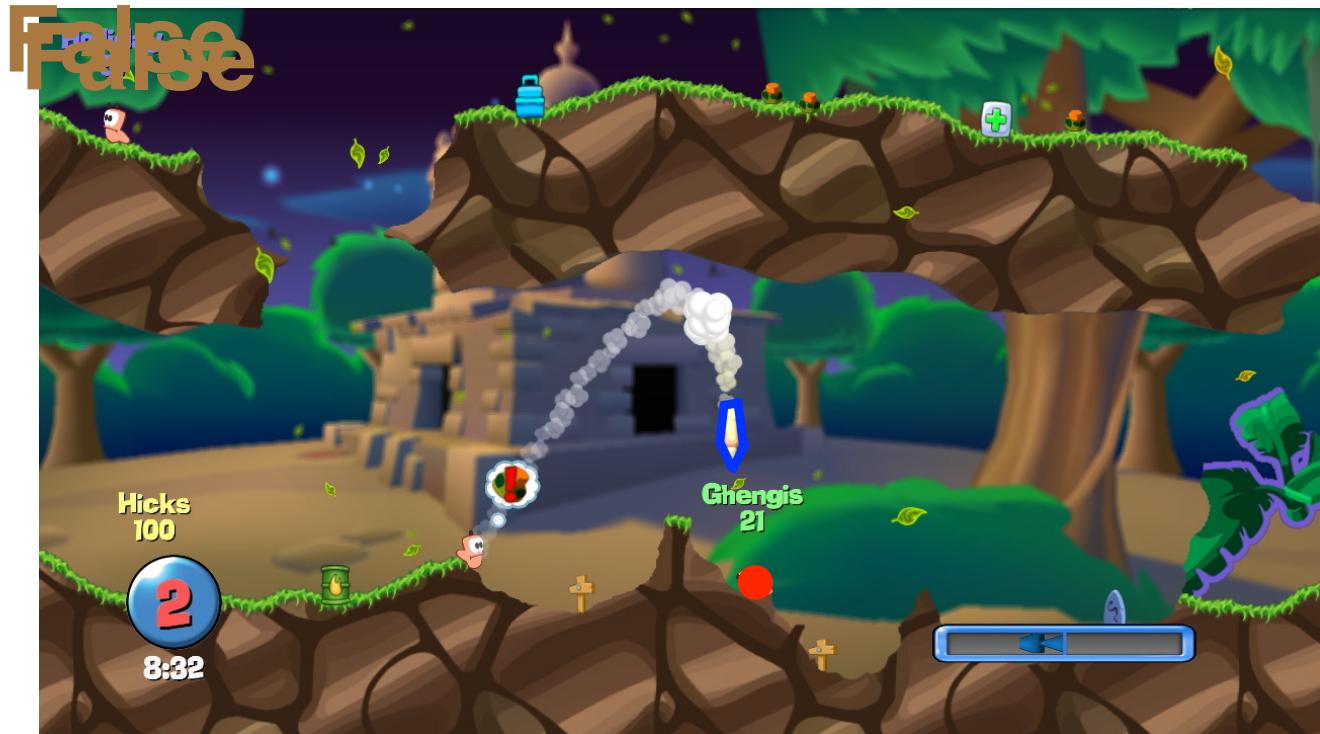


3

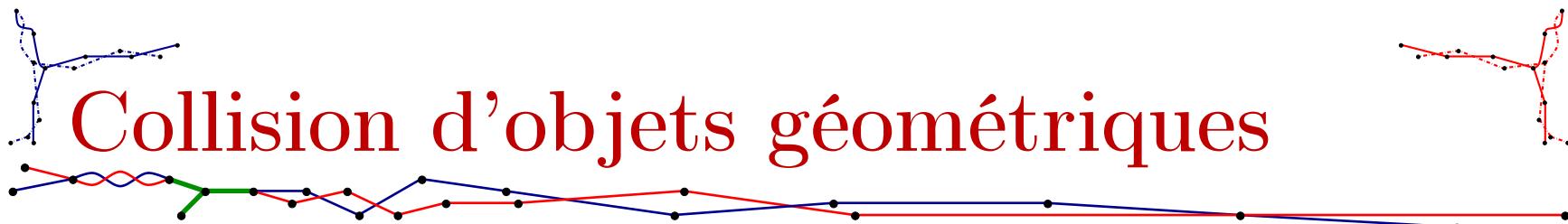


Collision d'objets géométriques

QUESTION : touché ?

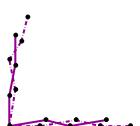
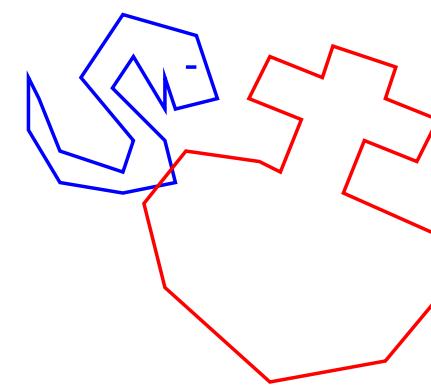
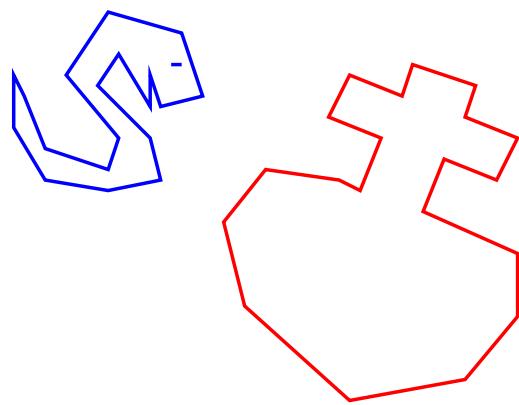


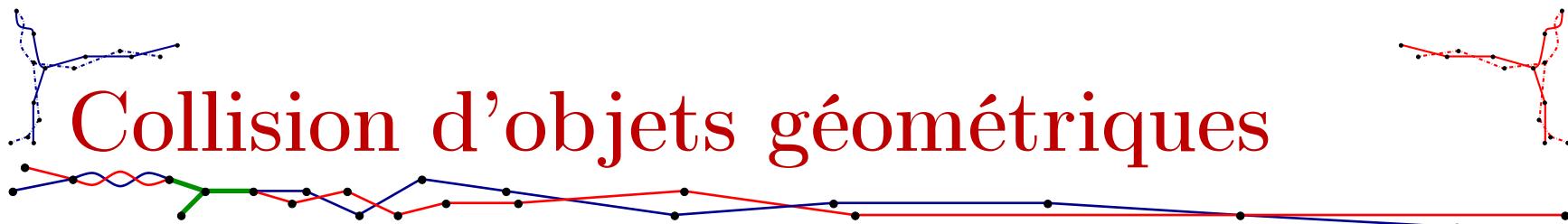
3



QUESTION : touché ?

??????

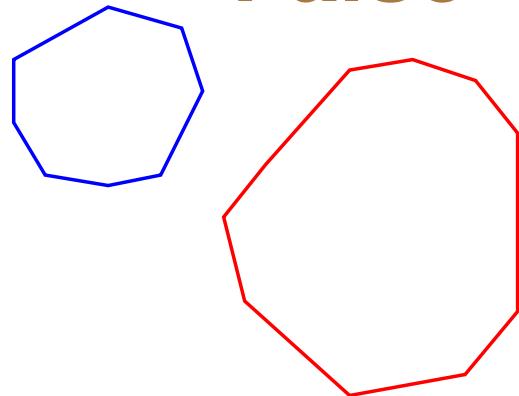




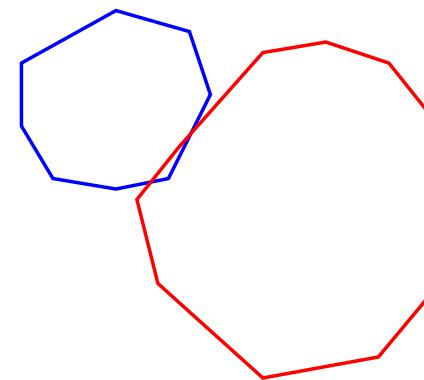
Collision d'objets géométriques

QUESTION : touché ?

False

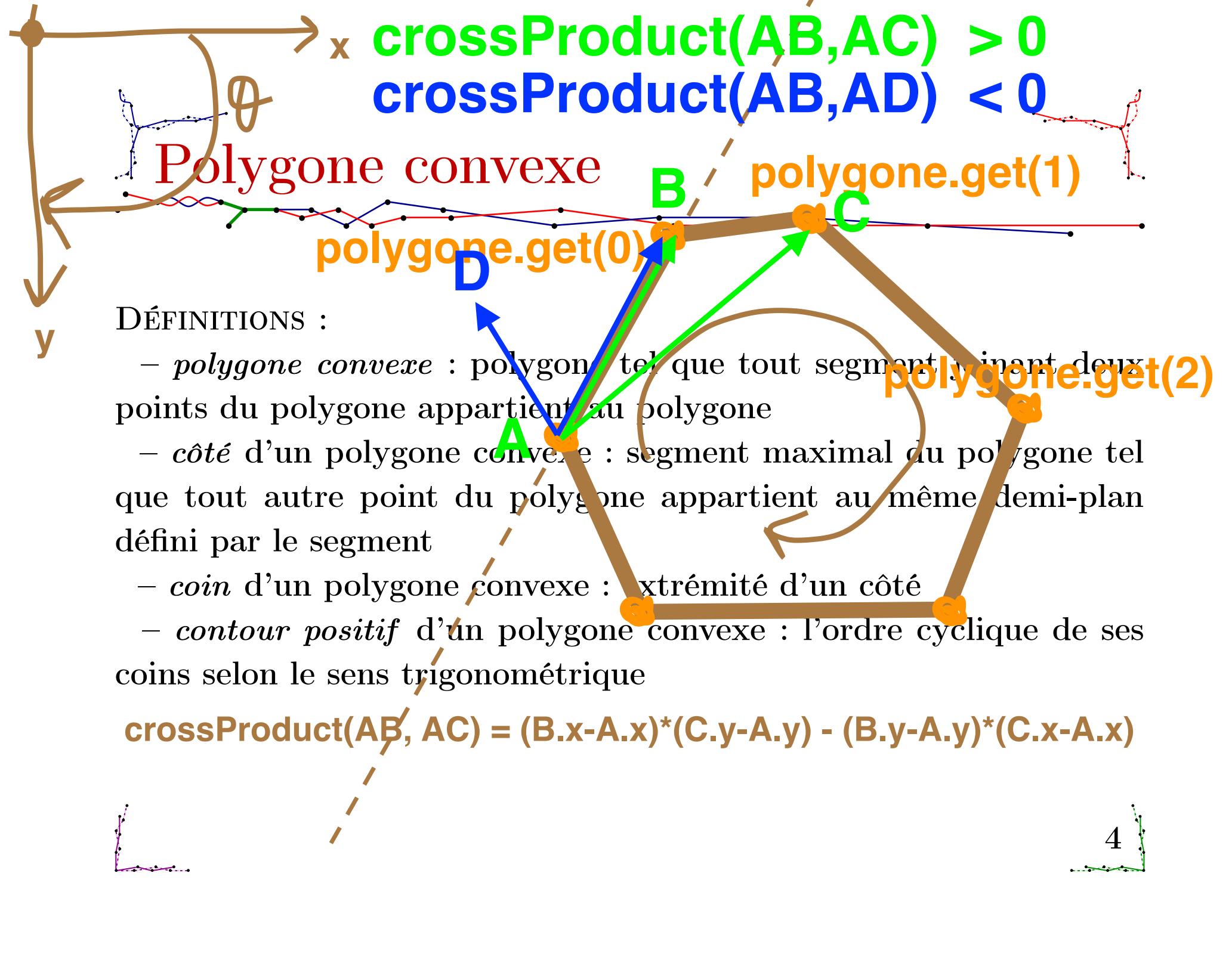


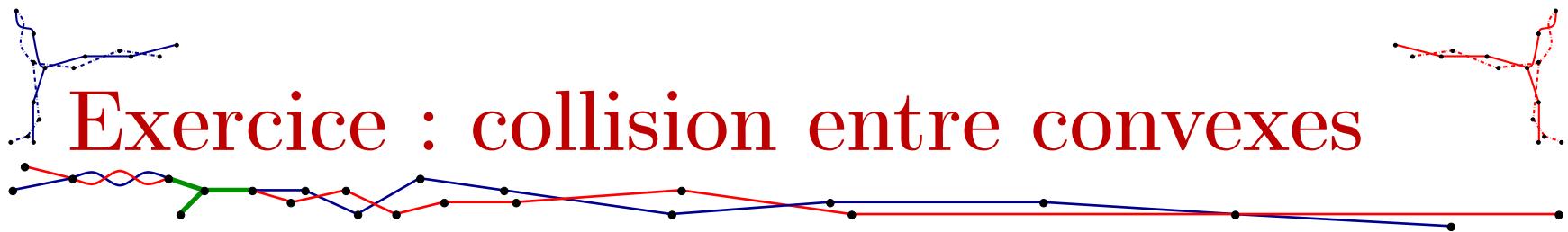
True!



3







Exercice : collision entre convexes

Separating axis theorem

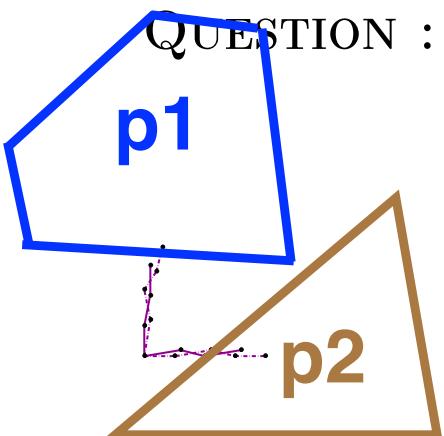
EXERCICE : Soient deux polygones convexes p_1 et p_2 . Montrer qu'il n'y a pas de collision entre p_1 et p_2 si et seulement si une des deux conditions suivantes est vérifiée :

condition 1

– il existe un côté de p_1 tel que tout coin de p_2 est “de l'autre côté” des coins de p_1 dans les demi-plans définis par ce côté **propRouge**

condition 2

– il existe un côté de p_2 tel que tout coin de p_1 est “de l'autre côté” des coins de p_2 dans les demi-plans définis par ce côté



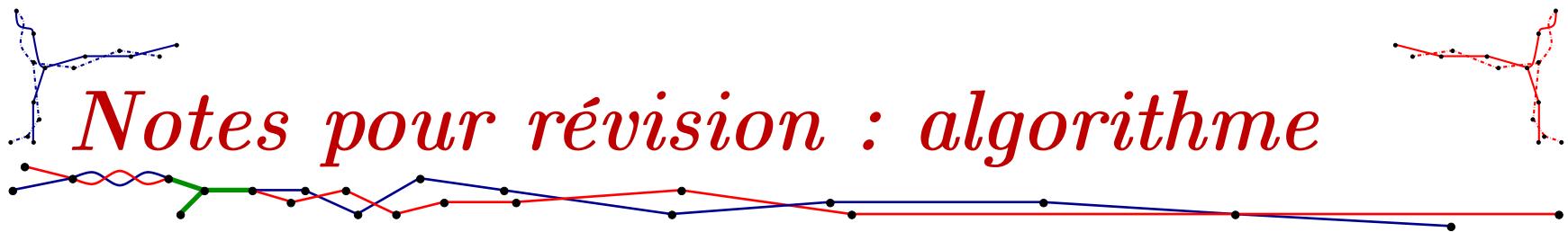
QUESTION : implantation ?

```
public boolean estCollision(ArrayList<Point> p1,
                           ArrayList<Point> p2) {
    boolean trouvé = false;
    for (Côté côté : liste_des_côtés(p1)) {
        boolean propRouge = true;
        for (Coin coin : liste_des_coins(p2)) {
            if (du_même_côté(coin, liste_des_coins(p1))) {
                propRouge = false;
                break;
            }
        }
        if (propRouge) {
            trouvé = true;
            break;
        }
    }
    if (trouvé) return false;
}

//code pour condition 2
//return false (pas de Collision)
//sinon, return true (Collision)
```

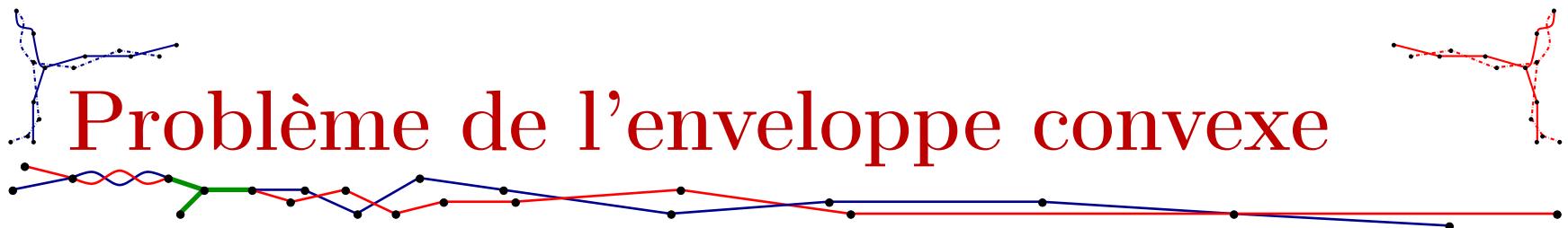
```
public boolean estCollision(ArrayList<Point> p1,
                           ArrayList<Point> p2) {
    for (Côté côté : liste_des_côtés(p1)) {
        boolean propRouge = true;
        for (Coin coin : liste_des_coins(p2)) {
            if (du_même_côté(coin, liste_des_coins(p1))) {
                propRouge = false;
                break;
            }
        }
        if (propRouge) return false;
    }
    //code pour condition 2
    //return false (pas de Collision)
    //sinon, return true (Collision)
}
```





1. pour tout côté pq de p1
2. r \leftarrow un coin de p1 distinct de p et de q
3. côtéSéparant \leftarrow true
4. pour tout coin s de p2
5. si s est du même côté que r par rapport à (pq) alors
6. côtéSéparant \leftarrow false
7. si côtéSéparant = true alors retourner pas_de_collision
8. pour tout côté pq de p2
9. r \leftarrow un coin de p2 distinct de p et de q
10. côtéSéparant \leftarrow true
11. pour tout coin s de p1
12. si s est du même côté que r par rapport à (pq) alors
13. côtéSéparant \leftarrow false
14. si côtéSéparant = true alors retourner pas_de_collision
15. retourner collision





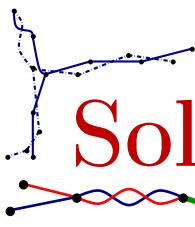
Problème de l'enveloppe convexe

IN : Points, une liste de coordonnées de points en 2D

OUT : Enveloppe, une liste d'éléments de Points représentant le contour positif de l'enveloppe convexe de Points

DÉFINITION : l'enveloppe convexe d'un ensemble de points est le plus petit polygone convexe contenant ces points





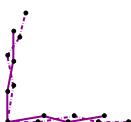
Solution naïve

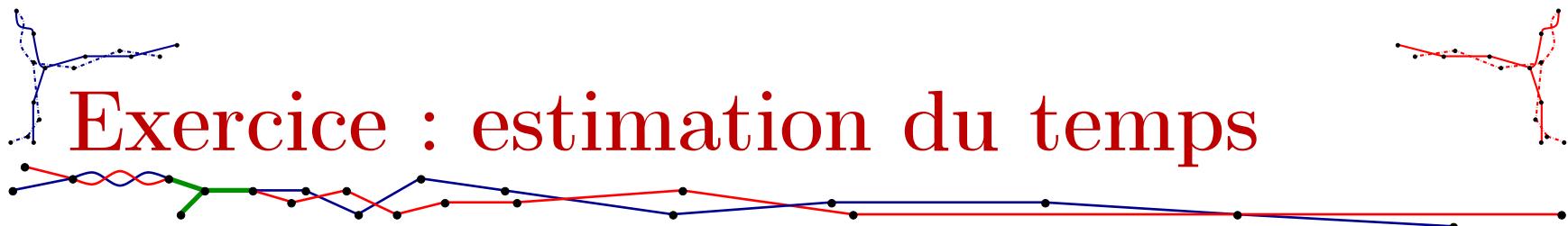


PRINCIPE : pour toute paire de points dans Points, vérifier s'elle forme un côté de l'enveloppe convexe de Points en testant si tout autre point de Points sont “du même côté” des demi-plans définis par le segment passant par la paire de points

QUESTION : complexité ? implantation ?

Notes pour révision : voir TME2.



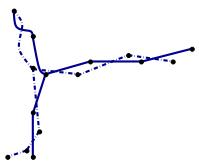


Exercice : estimation du temps

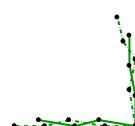
QUESTION : un ordinateur de l'ordre du Giga-Hertz exécutant un algorithme en $O(n^3)$, avec $n = 10000$, quel est le temps d'exécution attendu (en ordre de grandeur) ?

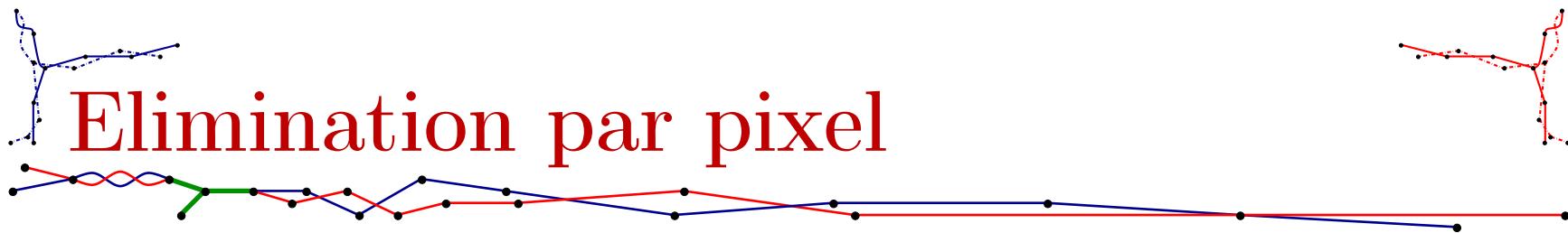
- algorithme en $O(n \log n)$?
- algorithme en $O(n \log h)$, avec $h \ll n$?
- algorithme en $O(n + n' \log h)$, avec $n' \ll n$ et $h \ll n$?





Enveloppe convexe : précalculs





Question: comment calculer les points rouges?

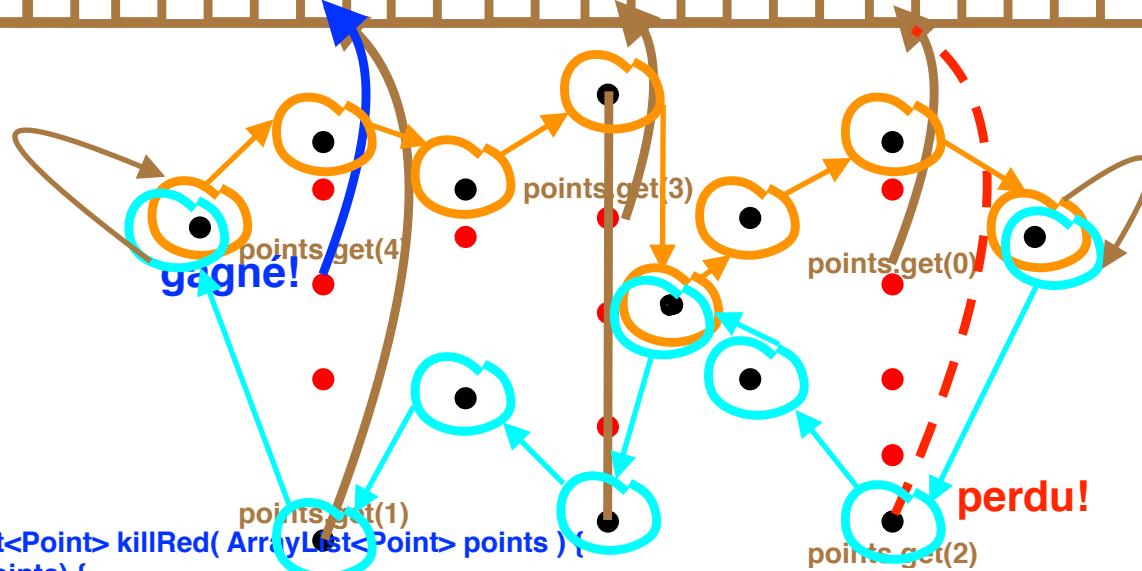
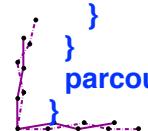
IDÉE : si trois points de Points ont le même abscisse, alors au plus deux de ces points appartiennent à l'enveloppe convexe de Points

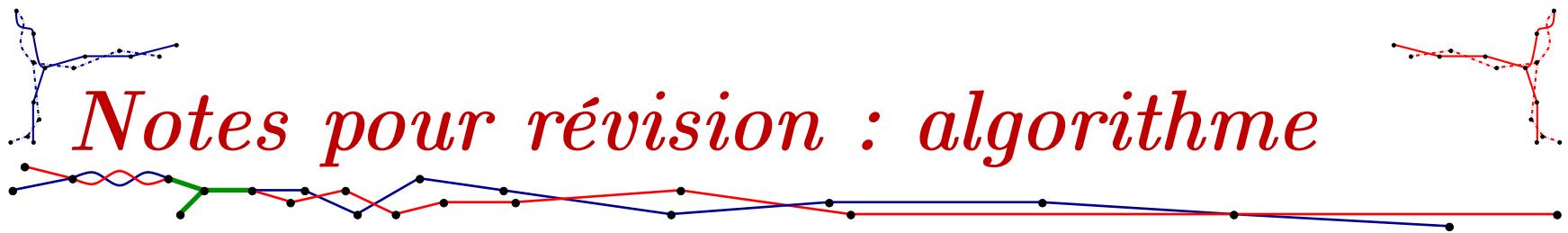
tableau ymin



tableau ymax

```
public ArrayList<Point> killRed( ArrayList<Point> points ) {
    for (Point p: points) {
        if ( ?????panier????? ) {
            mettre_p_dans_panier
        }
    }
    parcourir_le_panier
}
```





Notes pour révision : algorithme

0. $xMax = 0$; for (Point p: points) if ($p.x > xMax$) $xMax=p.x$;
1. $ymin \leftarrow$ tableau de points de taille assez grande ~~$xMax+1$~~
2. pour tout p dans Points
3. si $ymin[p.x] = \text{NULL}$ ou si $ymin[p.x].y > p.y$
4. $ymin[p.x] \leftarrow p$
5. similairement avec $ymax$
6. parcourir $ymin$ et $ymax$ et retourner les éléments non NULL

Complexité avec $n = \text{points.size}()$:

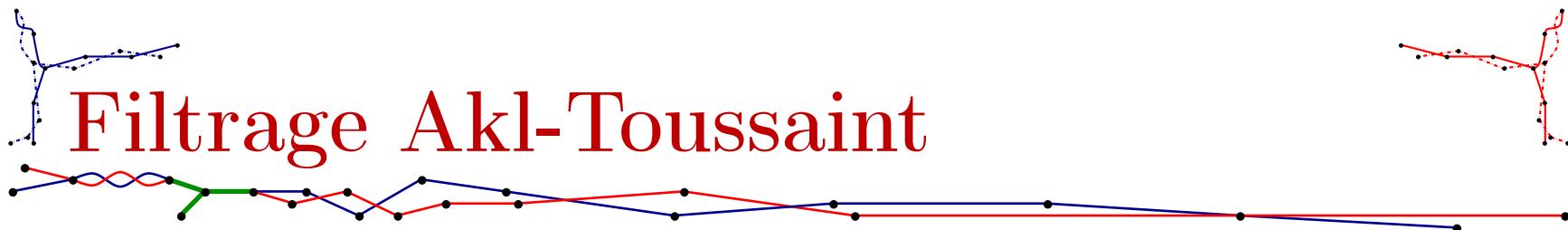
ligne 0 $\rightarrow O(n)$
ligne 1 $\rightarrow O(1)$
lignes 2-4 $\rightarrow O(n)$
ligne 5 $\rightarrow O(n)$, idem que lignes 2-4
ligne 6 $\rightarrow O(xMax)$



TOTAL:

$$\begin{aligned} & O(n) + O(1) + O(xMax) \\ & = \\ & O(n+xMax) \\ & = \\ & O(\max(n,xMax)) \end{aligned}$$

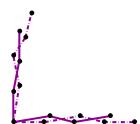
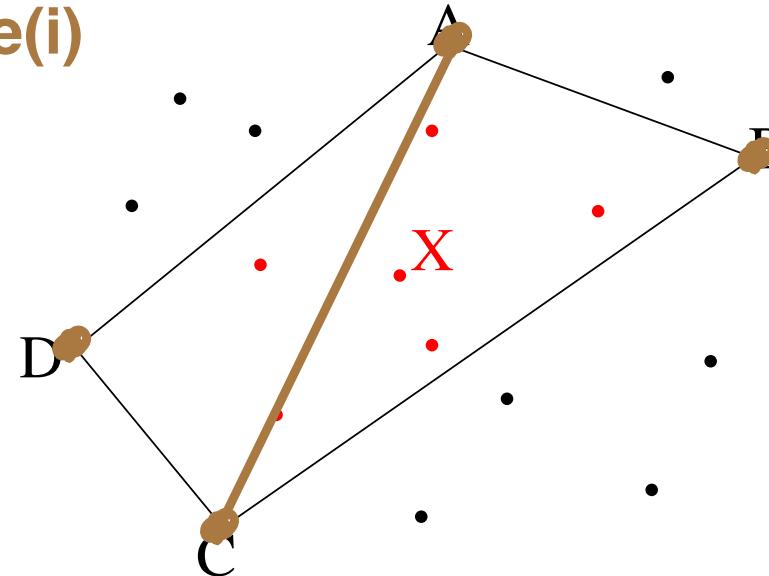




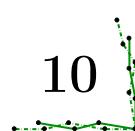
Filtrage Akl-Toussaint

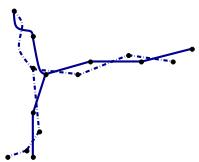
IDÉE : soient A, B, C, D les points les plus au Nord, Est, Sud, Ouest parmi Points, alors aucun point appartenant strictement au quadrilatère ABCD est sur l'enveloppe convexe de Points

**Si `points.get(i)` est à l'intérieur du triangle ABC (ou ACD)
Alors `points.remove(i)`**



10

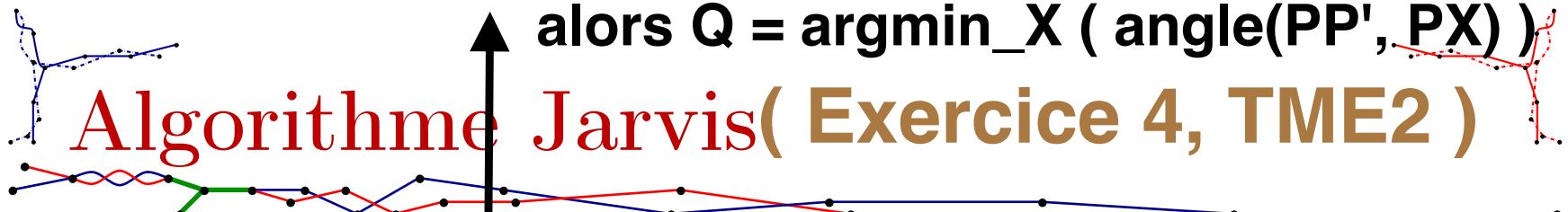




Enveloppe convexe : algorithmes

Notes pour révision : voir TME2

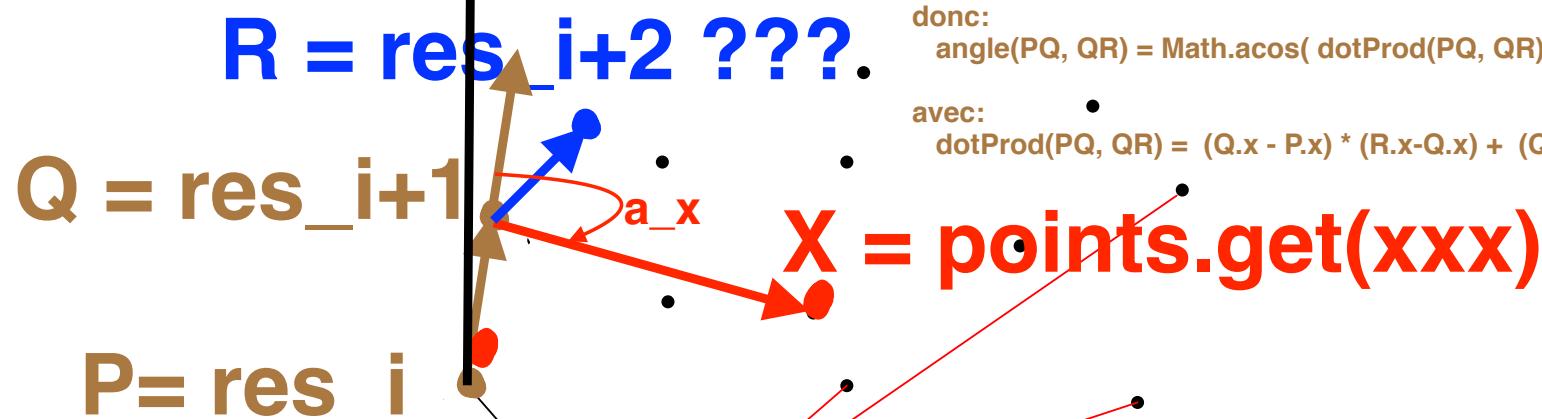




Supposons que `res_i` et `res_{i+1}` sont déjà calculés

IDÉE : si PQ est un côté de l'enveloppe convexe de Points, et R le point de Points tel que l'angle (\vec{PQ}, \vec{QR}) est minimum, alors QR est un côté de l'enveloppe convexe de Points

$$\text{dotProd}(PQ, QR) = |PQ| * |QR| * \cos(\text{angle}(PQ, QR))$$

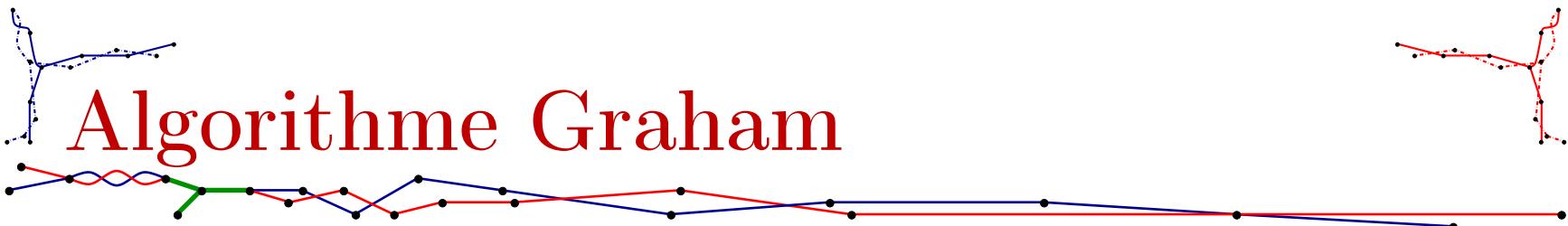


SuperPropriété:

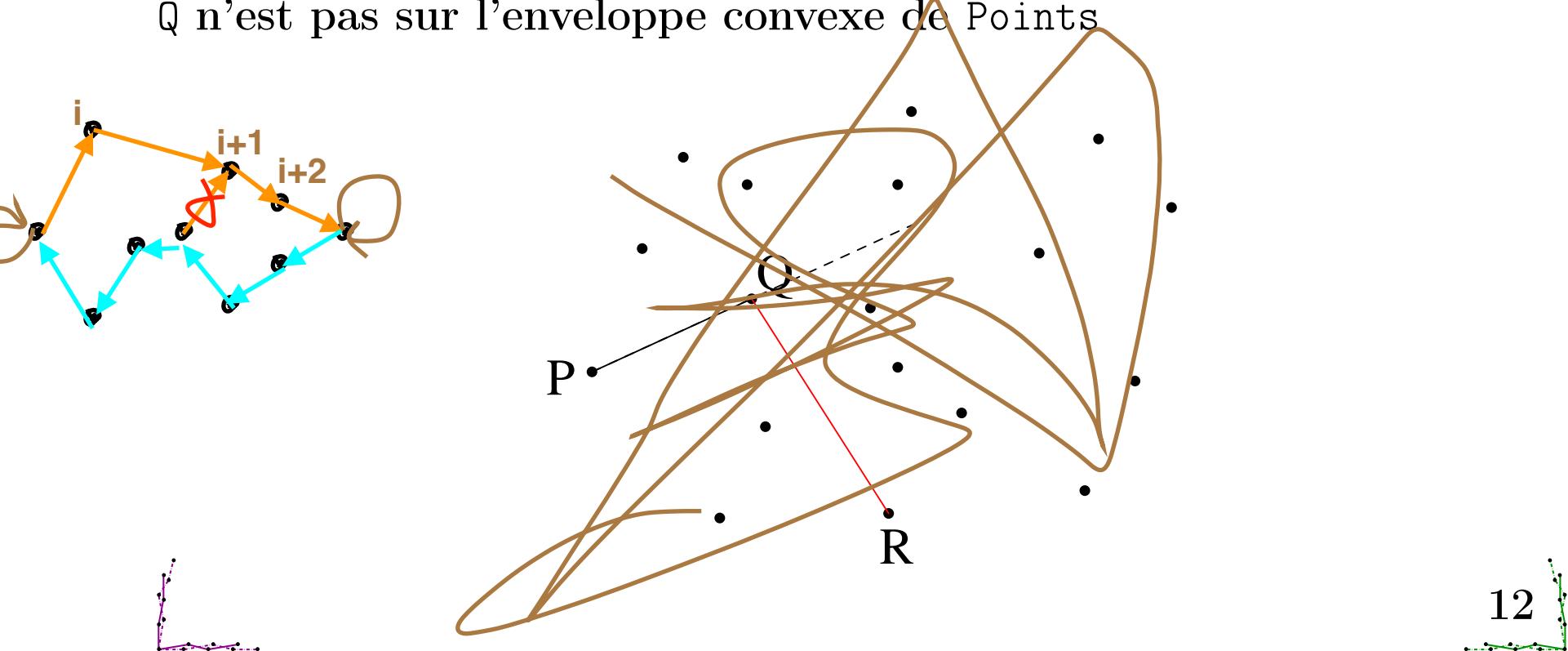
$$\text{angle}(PQ, QR) = \min_X \text{angle}(PQ, QX)$$

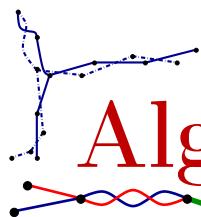
c.à.d.

$$R = \operatorname{argmin}_X \text{angle}(PQ, QX)$$



IDÉE : si P , Q et R sont trois points de Points d'abscisse croissant, et R appartient au “mauvais côté” des demi-plans définis par PQ , alors Q n'est pas sur l'enveloppe convexe de Points





Algorithme QuickHull



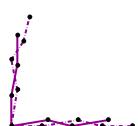
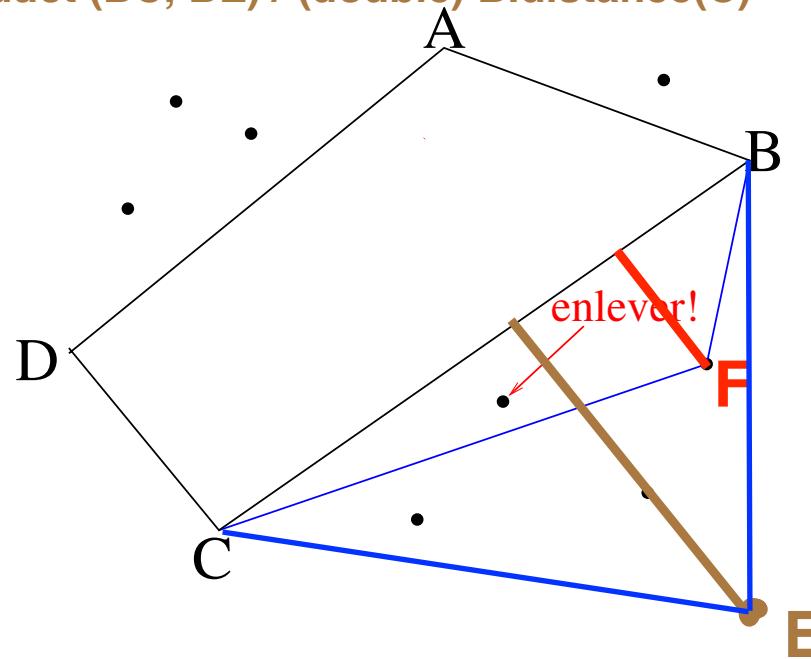
Question: comment déterminer E tel que E appartient à l'enveloppe convexe?

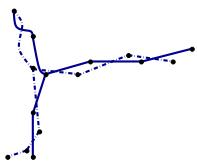
IDÉE : Akl-Toussaint est génial, comment l'étendre ?

$$E = \text{argmax}_X (\text{distance}(BC, X))$$

où:

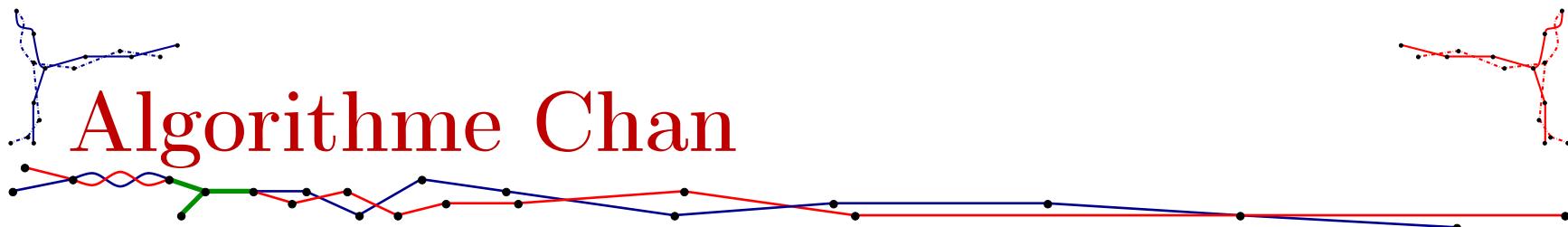
$$\text{distance}(BC, X) = \text{crossProduct}(BC, BE) / (\text{double}) B.\text{distance}(C)$$



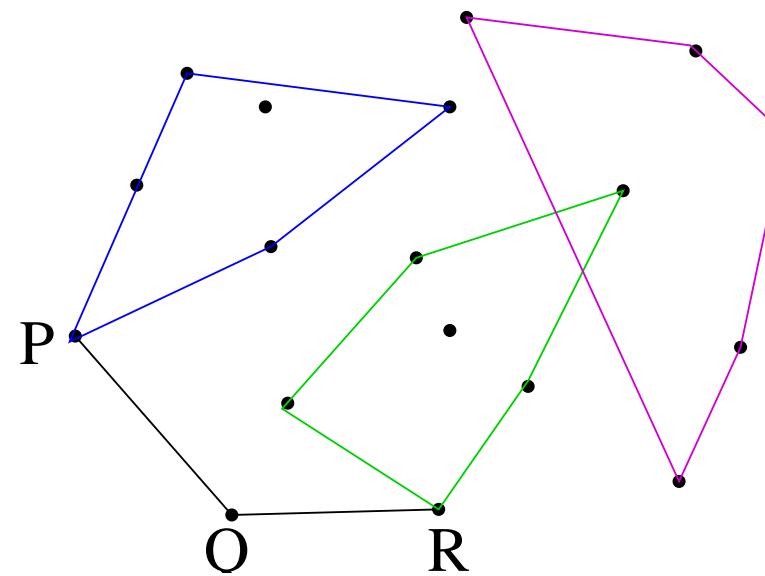


Peut on faire mieux ?



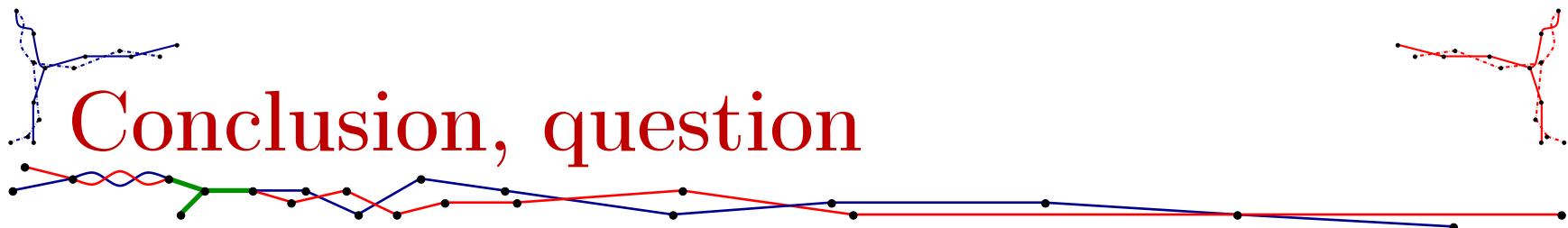


IDÉE : Jarvis + décomposition



14





Conclusion, question

CONCLUSION :

- algorithme naïf en $O(n^3)$
- technique incrémental (Jarvis)
- technique de précalcul (pixel, Akl-Toussaint, QuickHull)
- technique d'élagage (Graham)
- technique ultime : décomposition (Chan)

QUESTION :

- implantation ? (voir TME)



