Université Savoie Mont Blanc,
UFR SceM
INFO601_CMI, L3 INFO

November 27, 2025

TD 5, Géométrie algorithmique

Exercice 1. Virage à gauche ou à droite

Soient deux segments consécutifs $[p, q]$ et $[q, r]$. Comment déterminer si on fait un virage à gauche ou à droite ou si on va tout droit ? Est-on obligé de déterminer l'angle ?

Exercice 2. Triangulation d'un polygone simple

Tout polygone simple peut être décomposé en union de triangles, i.e. *triangulé*. Il est possible d'exhiber une triangulation en temps linéaire (Tarjan 1991), mais l'algorithme est difficile. Un algorithme beaucoup plus facile à écrire est basé sur le "découpage d'oreille". Tout polygone simple contient au moins deux oreilles, c'est-à-dire deux segments consécutifs qui tournent à gauche (en supposant que le contour du polygone va dans le sens trigonométrique). On met un triangle à chacune des oreilles, on enlève les deux sommets au fond des oreilles et on relance l'algorithme. Ecrivez donc cet algorithme.

La sortie sera donc une liste de triplets de sommets correspondants aux triangles.

Exercice 3. Test de convexité d'un polygone

Un ensemble C est dit *convexe* ssi $\forall p, q \in C, [pq] \subset C$. On voit alors qu'un polygone simple et son intérieur forme un ensemble *convexe* ssi si les segments font toujours un virage à gauche.

Si on a une liste de points $(p_0, \ldots, p_{n-1})$, est-il suffisant de tester Orientation$(p_{i-1}, p_i, p_{i+1}) \geq 0$ pour tout sommet p_i ? Comment garantir que cela fonctionne ?

Exercice 4. Intersection entre deux droites

On se donne deux droites, l'une passant par les points a et b, l'autre par les points c et d. Comment calculer leur point d'intersection ?

Exercice 5. Intersections dans une soupe de segments

On se donne n segments dans le plan. Proposez un algorithme qui retourne la liste des intersections dans cette soupe de segments. Selon vous, quelle est la complexité minimale d'un tel algorithme ?

Exercice 6. Intersection rayon et segment

Ecrire une fonction HRayon-Intersecte-Segment qui retourne vrai si un rayon $[pq)$ coupe le segment $[r, s]$. Adaptez Intersection-Segments.

Exercice 7. Test intérieur dans un polygone

Un point p est à l'intérieur d'un polygone simple P ssi un rayon partant de lui intersecte un nombre impair de fois le contour. Utiliser la fonction précédente pour trouver un algorithme qui détermine si p est dans l'intérieur de P en temps linéaire.

Exercice 8. Calcul incrémental d'enveloppe convexe

On rappelle que vous disposez de la fonction Orientation($\underline{\mathbf{E}}p, q, r : Point$) qui vous retourne un nombre positif si r est à gauche de $\vec{pq}$, 0 si sur la ligne portant $\vec{pq}$, négatif sinon.

Un polygone sera représenté sous forme d'une séquence de points, i.e. un tableau de Point. On propose l'algorithme suivant pour calculer l'enveloppe convexe de n points quelconques, sous forme d'une file.

- on prend les 3 premiers pour créer un triangle orienté dans le sens trigonométrique. Cela crée un polygone convexe initial noté P
- puis on ajoute point par point les points restants à P en le mettant à jour pour qu'il reste convexe.

1. Ecrire l'action MetAJour-Cvx(**ES** F: File de Point, **E** P : Point) qui met à jour l'enveloppe convexe actuelle stockée sous forme de File avec le nouveau point P.

   Les fonctions pour manipuler les Files sont rappelées à la fin. Pour parcourir une file il suffit d'enlever un élément en tête de file et le remettre en queue de file jusqu'à tomber sur le même point.

   Un ou des dessins peuvent aider.

```
// On cherche à construire une file F tels que 3 points consécutifs de F
// ''tournent à gauche''.
Action MetAJour-Cvx( ES F: File de Point, E P : Point )

Var premier, prec, cur : Point
    agauche : bool
premier <- Défile( F )
prec    <- premier
Enfile( F, premier )
// On cherche si P est complètement à l'intérieur de F
agauche <- Orientation( prec, Tete( F ), P ) > 0
Tant que Tete( F ) != premier et agauche Faire
    prec <- Defile( F )
    Enfile( F, prec )
    agauche <- Orientation( prec, Tete( F ), P ) > 0
Fin Tant que
// Si non, alors P est dans l'enveloppe convexe
// et on élimine les points couverts par P.
Si non agauche Alors
    Enfile( F, P )
    prec <- Defile( F )
    Tant que Orientation( P, prec, Tete( F ) ) <= 0 Faire
        prec <- Defile( F )
    Fin Tant que
Fin Si
// Il pourrait rester des points mal placés juste avant
prec <- Defile( F )
Tant que Tete( F ) != P Faire
    Enfile( F, prec )
    cur <- Défile( F )
    Si Orientation( prec, cur, Tete(F) ) > 0 Alors
       Enfile( F, cur )
    Fin Si
Fin Tant que
```

2. Ecrire la fonction Enveloppe-Cvx(S F: File de Point, **E** T : tableau de Point, **E** n:entier) qui créé le triangle initial et appelle MetAJour-Cvx sur les autres points.

```
Action Enveloppe-Cvx( S F: File de Point, E T : tableau de Point, E n : entier )
Var i : entier

Enfile( F, T[ 0 ] )
Si Orientation( T[ 0 ], T[ 1 ], T[ 2 ] ) > 0 Alors
     Enfile( F, T[ 1 ] )
     Enfile( F, T[ 2 ] )
Sinon
     Enfile( F, T[ 2 ] )
     Enfile( F, T[ 1 ] )
Fin Si
Pour i de 3 à n Faire
    MetAJour-Cvx( F, T[ i ] )
Fin Pour
```

3. Quelle est la complexité en pire cas de votre fonction ENVELOPPE-CVX selon n ?

```
Il est clair que la fonction MetAJour-Cvx à une complexité linéaire
en la taille de la file F.
A l'étape i de l'algorithme Enveloppe-Cvx, on note f_i cette taille.
On obtient un temps total T_n de l'ordre de :
T_n = cste + O(f_3) + O(f_4) + ... + O(f_n)
Il est clair que f_i <= n. Donc T_n = cste + O( (n-3)*n ) = O(n^2)
```

Exercice 9. Point appartient à un ensemble de points

En utilisant une structure de proximité représentant n points, proposez un algorithme qui retourne vrai si un point donné p est un de ces n points. Quelle est la complexité en pire cas de cet algorithme ?

Exercice 10. Point à distance ϵ d'un ensemble de n points

En utilisant une structure de proximité représentant n points, proposez un fonction qui retourne un point à distance inférieure à ϵ d'un point donné p, ou le point $INVALID$ sinon.

Exercice 11. Plus proches points dans un ensemble de n points

Avec une structure de proximité, déterminez la paire de points les plus proches parmi n points.

Exercice 12. Proximité avec des segments

Comment tester la proximité avec des segments ? Même si cette façon n'est pas optimale, on peut aussi utiliser des kD-tree. L'idée est de placer sur chaque segment suffisament de points. Ensuite, on ne fait qu'un test approximatif pour savoir si l'on est à côté d'un des points du segment. La petite difficulté est de ne pas mettre trop de points. En général, si on veut être sûr de détecter que l'on est à distance ϵ d'un segment, il s'agit de bien choisir le rayon δ des boules placés sur le segment de manière à : (1) ne pas avoir trop de points, (2) garantir que si un des points est à distance inférieur à δ alors la distance au segment est inférieure à ϵ.

Expliquez pourquoi un écart entre les points inférieur à 2ϵ et $\delta = \sqrt{2}\epsilon$ est un bon choix.

On note que l'aire de l'ensemble des points "distance à un segment de longueur l inférieure à ϵ" est de $\pi\epsilon^2 + 2l\epsilon$. Or l'aire de l'ensemble des points "à distance inférieure à $\sqrt{2}\epsilon$ des points répartis comme indiqué sur le segment de longueur l" est de $3\pi\epsilon^2 + l(1 + \frac{\pi}{2})\epsilon$. Le ratio d'aire est de $1,28$ pour des longs segments. On va donc surtout tester les bons segments.