Licence informatique - L3 Année 2011/2012

**Conception d'algorithmes et applications (LI325)**

COURS 3

Résumé. Dans cette troisième séance entièrement consacrée aux applications du principe Diviser pour Régner, nous aborderons deux algorithmes pour trouver l'enveloppe convexe d'un ensemble fini de points du plan.

## 1. Diviser pour Régner pour la Géométrie Algorithmique

1.1. **Enveloppe convexe : QUICKHULL.** Par définition, un ensemble $C$ est dit *convexe* si et seulement si tout segment reliant deux points de $C$ est entièrement inclus dans $C$. Notre objectif est de calculer l'enveloppe convexe d'un ensemble $E$ de points du plan. C'est-à-dire le plus petit ensemble convexe contenant tous les points de $E$. On admettra que l'enveloppe convexe d'un ensemble fini $E$ de points est un polygone dont les sommets sont des points de $E$.

Soit $(\overrightarrow{AB})$ une droite orientée dans la direction donnée par le vecteur $\overrightarrow{AB}$ et $E$ un ensemble de points, on note $E_{(\overrightarrow{AB})}$, l'ensemble des points de $E$ situés à gauche de $(\overrightarrow{AB})$ union $\{A, B\}$. Un couple $(P, Q)$ de points de $E$ sera dit *E-antipodal* si et seulement si tous les autres points de $E$ sur la droite $(PQ)$ se situent sur le segment $[P, Q]$. Notre premier algorithme appelé QUICKHULL repose sur le principe suivant :
Soit $E$ un ensemble de points et un couple $(P, Q)$ $E$-antipodal tels que tous les points de $E$ sont à gauche de la droite orientée $(\overrightarrow{PQ})$. On construit un algorithme SEMIHULL qui prend en entrée $E$ et $(P, Q)$ et qui renvoie la liste des points de l'enveloppe convexe de $E$ rencontrés en allant de $P$ à $Q$ par le chemin qui passe par tous les points de l'enveloppe.

Pour ce faire, SEMIHULL divise l'ensemble $E$ en plusieurs sous-ensembles, de la manière suivante :

On détermine le point $H \in E$ tel que la distance de $H$ à $(PQ)$ soit maximum, et s'il y a plusieurs prétendants, $H$ sera celui dont l'angle $(\widehat{HPQ})$ est le plus ouvert.

Nous construisons maintenant les deux droites orientées, $(\overrightarrow{PH})$ et $(\overrightarrow{HQ})$. On résout récursivement SEMIHULL pour $(E_{(\overrightarrow{PH})}, (P, H))$ et $(E_{(\overrightarrow{HQ})}, (H, Q))$. Pour obtenir le résultat de SEMIHULL sur $(E, (P, Q))$, il suffit de concaténer le résultat de SEMIHULL sur $(E_{(\overrightarrow{PH})}, (P, H))$ avec celui de SEMIHULL sur $(E_{(\overrightarrow{HQ})}, (H, Q))$ en prenant soin de ne pas écrire deux fois le point $H$.

Nous avons donc le principe Diviser pour Régner suivant :
Diviser : On scinde l'ensemble de points $E$ en deux ensembles.
Régner : On résout le problème sur chacune des deux sous-séquences en utilisant récursivement SEMIHULL si celle-ci n'est pas réduite à deux éléments et en renvoyant directement l'enveloppe sinon.
Combiner : On concatène les deux sous-listes obtenues pour obtenir la liste demandée.

Nous pouvons maintenant donner un pseudo-code pour SEMIHULL. Soit $l$ une liste non vide, on note $Trunc(l)$ la liste $l$ privée de son premier élément. On note classiquement par $*$ la concaténation de listes. On suppose que $E$ contient au moins deux points.

**Algorithme 1** : SEMIHULL$(T, (P, Q))$

**Entrées** : Un tableau $T$ contenant les points de $E$ et un couple de point de $(P, Q)$ avec les hypothèses que tous les points de $E$ autres que $P$ et $Q$ sont à gauche de $(\overrightarrow{PQ})$ (En particulier, $(P, Q)$ est $E$-antipodal).

**Sorties** : Une liste contenant les points de l'enveloppe convexe de $E$ rencontrés en allant de $P$ à $Q$ par le chemin qui passe par tous les points de l'enveloppe.

```
si T a deux éléments alors
    retourner (P, Q)
sinon
    H := T[LEPLUSLOIN(T, (P, Q))];
    T1 le tableau contenant les points P, H et les points de T (strictement) à gauche de (PH);
    T2 le tableau contenant les points Q, H et les points de T à (strictement) gauche de (HQ);
    retourner SEMIHULL(T1, (P, H)) *Trunc(SEMIHULL(T2, (H, Q)))
```

Nous décrivons ici un algorithme auxiliaire LEPLUSLOIN qui permet de trouver le point le plus éloigné de $PQ$ dans le cas où ce point est unique. Il existe des versions permettant d'obtenir s'il y a plusieurs candidats, celui dont l'angle $(\widehat{HPQ})$ est le plus ouvert.

**Algorithme 2** : LEPLUSLOIN$(T, (P, Q))$

**Entrées** : Un tableau $T$ de longueur $n$ contenant les points de $E$ et un couple de points de $(P, Q)$.

**Sorties** : la position dans le tableau $T$ du point situé le plus loin de la droite orientée $(\overrightarrow{PQ})$.

```
pos :=1;
M := (Q_x − P_x)(T[1]_y − P_y) − (Q_y − P_y)(T[1]_x − P_x);
pour i allant de 2 à n faire
    t := (Q_x − P_x)(T[i]_y − P_y) − (Q_y − P_y)(T[i]_x − P_x);
    si t > M alors
        M := t ; pos := i
retourner pos
```

Pour obtenir l'enveloppe convexe d'un ensemble quelconque $E$ (les éléments de $E$ sont dans un tableau $T_E$) il suffit de faire les opérations suivantes : Soit $P_G$ (resp. $P_D$) le point le plus à gauche de $E$ (resp. le plus à droite). Plus précisément, on prend $P_G$ (resp. $P_D$) le point le plus petit (resp. le plus grand) pour l'ordre léxicographique sur les coordonnées, cela évite les ambigüités. On construit $T^+$ (resp. $T^-$) le tableau contenant $P_G$, $P_D$ et tous les points à gauche (resp. à droite) de $(\overrightarrow{P_G P_D})$. On calcule SEMIHULL$(T^+, (P_G, P_D))$ et SEMIHULL$(T^-, (P_D, P_G))$. On a alors que :

QUICKHULL$(T_E)$ := Trunc(SEMIHULL$(T^+, (P_G, P_D))$)*Trunc(SEMIHULL$(T^-, (P_D, P_G))$).

### Analyse de l'algorithme SEMIHULL

*Preuve de Terminaison :* Par récurrence sur la longueur du tableau $T$. Si $T$ a deux éléments l'algorithme s'arrête immédiatement. Sinon, il appelle deux fois SEMIHULL sur des tableaux ayant au moins un élément de moins ($T_1$ (resp. $T_2$) ne contient pas $Q$ (resp. $P$)).

*Preuve de Validité :* Il y a plusieurs points à vérifier. Tout d'abord, il faut vérifier que les instances $(T, (P, Q))$ données à SEMIHULL à chaque étape de la récursion vérifient bien les hypothèses. Le tableau $T$ a bien au moins deux éléments car il contient $P$ et $Q$ qui sont distincts. Il

reste à montrer que le couple $(P, Q)$ est $T$-antipodal. Initialement, cela est vrai car on commence avec $P_G$ et $P_D$ respectivement les points le plus à gauche et le plus à droite de $E$. Maintenant comme à chaque étape, le tableau $T$ ne contient pas de points à droite de $(\overrightarrow{PQ})$, il s'ensuit qu'il n'y a pas de point de $T_1$ situé avant $P$ sur $(\overrightarrow{PH})$ ni de point de $T_2$ situé après $Q$ sur $(\overrightarrow{HQ})$. De plus $H$ étant le point le plus loin de $(PQ)$, on obtient que $(P, H)$ et $(H, Q)$ sont antipodaux. Maintenant, vérifions que SEMIHULL renvoie bien l'enveloppe convexe des points de $T$. Par induction, si $T$ a deux éléments, l'algorithme renvoie la liste $(P, Q)$ qui est bien l'enveloppe convexe des deux points. Maintenant, si $T$ contient plus de deux points, on remarque que le point $H$ est un point de l'enveloppe convexe des points de $T$. En effet, si on trace une droite parallèle à $(PQ)$ et passant par $H$, il n'y a aucun point de $T$ au-dessus de cette droite ; et s'il y a d'autres points de $T$ sur cette droite, par notre choix, $H$ se situe à leur extrême gauche. Ainsi $H$ ne peut pas être exprimé comme combinaison convexe de deux autres points de $T$. Donc tous les points situés à l'intérieur du triangle $(PQH)$ ne sont pas dans l'enveloppe convexe des points de $T$. Donc les points de l'enveloppe convexe de $T$ ne sont autres que les points de l'enveloppe convexe de $T_1$ et de l'enveloppe convexe de $T_2$. De plus, on remarque qu'il suffit de concaténer ses deux listes en prenant soin de ne pas copier deux fois $H$, pour obtenir la liste voulue.

*Analyse de la Complexité* dans le pire des cas en nombre d'opérations élémentaires $(+, -, \times, *, <)$ : Soit $n$ le nombre de points de $T$. L'obtention à partir de $T$ de $T_1$ et $T_2$ est effectuée en $O(n)$ opérations (il faut vérifier pour chaque point une inégalité du type $ax + by + c > 0$). De même, pour trouver le point le plus éloigné $H$, on fait $O(n)$ opérations élémentaires ($n-1$ déterminants $2 \times 2$ et $n-1$ comparaisons). Puis, récursivement, on cherche les enveloppes convexes de $T_1$ et $T_2$. Comme pour QUICKSORT, le pire des cas arrive quand les points sont mal répartis entre $T_1$ et $T_2$, c'est-à-dire quand $T_1$ est vide et $T_2$ contient tous les points de $T$ sauf un (ou réciproquement). Or le lecteur remarquera que ce cas se produit à toutes les étapes de la récursion pour certaines instances. On obtient alors la récurrence $T(n) = T(n-1) + O(n)$. Et donc QUICKHULL fait dans le pire des cas $O(n^2)$ opérations élémentaires sur une instance de taille $n$. Néanmoins, comme QUICKSORT, l'algorithme QUICKHULL est un bon algorithme en moyenne. Ceci est très difficile à montrer, mais intuitivement en faisant l'hypothèse raisonnable qu'il existe un entier $b$ tel que $T_1$ et $T_2$ ont chacun un cardinal au plus égal à $\left|\frac{l(T)}{b}\right|$ et ainsi de suite à chaque niveau de récursion, alors l'algorithme fait $O(n \log n)$ opérations élémentaires pour trouver l'enveloppe convexe d'un ensemble de $n$ points.

1.2. **Enveloppe convexe : FUSIONHULL.** Le principe de cet algorithme est le suivant : on commence par trier en ordre croissant suivant leur abscisse les points de l'ensemble dont on cherche l'enveloppe convexe. Nous avons donc une séquence ordonnée de $n$ points $(P_1, ..., P_n)$. On scinde cette séquence en deux sous-séquences $(P_1, ..., P_k)$ et $(P_{k+1}, ..., P_n)$ de longueurs respectives $\lceil \frac{n}{2} \rceil$ et $\lfloor \frac{n}{2} \rfloor$. On fabrique récursivement les enveloppes convexes des deux sous-séquences (si la séquence comporte trois points ou moins, on résout directement). Enfin on fusionne les deux enveloppes convexes obtenues. Cette dernière opération nécessite quelqu'attention si on veut obtenir un algorithme efficace. Pour l'accomplir on construit un "pont supérieur" et un "pont inférieur" entre les deux enveloppes convexes. On obtient le pont supérieur de la façon suivante : Soient $G$ le point le plus à droite de l'enveloppe de gauche et $D$ le point le plus à gauche de l'enveloppe de droite et on note $\Delta$ la droite passant par $G$ et $D$ ; si le point qui suit $G$ en parcourant l'enveloppe convexe de gauche dans le sens direct (ou trigonométrique) est au dessus de $\Delta$ on remplace $G$ par celui-ci ; de même si le point qui précède $D$ sur l'enveloppe convexe de droite parcourue dans le sens direct est au dessus de $\Delta$ on remplace $D$ par celui-ci (son prédecesseur) ; on itère ces deux actions autant que possible. A la fin, la droite $\Delta$ fournit le pont supérieur. En termes imagés, on fait monter la droite $\Delta$ le plus haut possible. On procède symétriquement pour le pont inférieur.

Plus schématiquement, on a donc :

Diviser : On scinde la séquence initiale en deux sous-séquences $(P_1, ..., P_k)$ et $(P_{k+1}, ..., P_n)$ de longueur respective $\lceil \frac{n}{2} \rceil$ et $\lfloor \frac{n}{2} \rfloor$.

Régner : On résout par appel récursif le problème sur les sous-séquences $(P_1, ..., P_k)$ et $(P_{k+1}, ..., P_n)$

si elles contiennent plus de 3 éléments, sinon le problème se résout trivialement.
COMBINER : On crée les ponts supérieurs et inférieurs entre les enveloppes convexes respectives des deux sous-séquences et on les fusionne à l'aide des deux ponts.

Soit $M$ un point situé à l'extérieur d'un domaine convexe et soit $[AB]$ un segment situé sur le bord de ce domaine. On dira que $M$ peut *voir* le segment $[AB]$ si et seulement si le triangle $AMB$ n'intersecte pas le domaine convexe ailleurs qu'en $[AB]$. Supposons que $B$ soit le successeur de $A$ quand l'on tourne autour de l'enveloppe convexe dans le sens direct. On observe alors que $M$ voit le segment [AB] si et seulement si $M$ est à droite de la droite orientée $(\overrightarrow{AB})$. :

Une enveloppe convexe $C$ sera représentée en machine par une liste circulaire doublement chaînée contenant les points de $C$ dans l'ordre de leur apparition quand on parcourt $C$ dans le sens direct. On dispose des deux primitives *prec* et *succ* qui permettent étant donnés une liste et un point $P$ de cette liste, d'obtenir le point qui se trouve avant et respectivement après $P$. De plus, on dispose de EXTGAUCHE($l$) et EXTDROIT($l$) qui permettent de se positionner sur le point d'abscisse minimale, respectivement le point d'abscisse maximale de $l$.
Commençons par donner le pseudo-code de PONTSUP.

**Algorithme 3** : PONTSUP($L_G$, $L_D$)

**Entrées** : Une liste circulaire doublement chaînée $L_G$ représentant l'enveloppe convexe à gauche et une liste circulaire doublement chaînée $L_D$ représentant l'enveloppe convexe à droite.
**Sorties** : Le couple de points qui appartient au pont supérieur.

```
1  G := EXTDROIT(L_G);
2  D := EXTGAUCHE(L_D);
3  trouvé := faux;
4  tant que trouvé := faux faire
5      trouvé := vrai;
6      tant que (D voit le segment [G, succ(G)]) faire
7          G := succ(G);
8          trouvé := faux;
9      tant que (G voit le segment [prec(D), D]) faire
10         D := prec(D);
11         trouvé := faux
12 retourner (G, D)
```

### Analyse de l'algorithme PONTSUP

*Preuve de Terminaison :*
Considérons une droite verticale $V$ qui sépare les deux polygones convexes (quitte si les deux polygones convexes ont des points de même abscisse à décaler de $\epsilon$ vers la gauche le polygone convexe de gauche). On remarque qu'à chaque fois que le point $G$ ou $D$ est réaffecté, la nouvelle droite $(GD)$ coupe $V$ en un point d'ordonnée plus grande qu'avant. Or ce point d'intersection appartient à l'ensemble des points $X$, intersection de $V$ avec une doite passant par un point du polygone convexe de gauche et un point du polygone convexe de droite. Cet ensemble est fini. Donc l'algorithme se termine.

*Preuve de Validité :* Evident par construction.

*Analyse de la Complexité* en nombre d'opérations élémentaires $(+, *, /, -, <)$ :
On remarque que la boucle tant que de la ligne 4 ne peut être itérée plus de $n + m$ fois où $n$ et $m$ sont respectivement les cardinaux des ensembles convexes gauche et droit. En effet, une fois un point réaffecté, l'ancien point peut être supprimé car on est sûr que le pont ne passe pas par lui. Donc à chaque itération, on peut supprimer un point. Il n'y a donc pas plus de $n + m$ itérations. Maintenant, à chaque itération, on fait un nombre constant d'opérations élémentaires. Pour vérifier que $A$ voit $[BC]$, on teste la valeur d'un déterminant $2 \times 2$. Donc l'algorithme PONTSUP fait $O(n + m)$ opérations pour construire le pont supérieur entre deux polygones convexes.

Maintenant, il est aisé de décrire un algorithme de type Diviser pour Régner à partir de PONTSUP et PONTINF, en voici le pseudo-code :

**Algorithme 4** : FUSIONHULL($T$)

**Entrées** : Un tableau $T$ de taille $n$ contenant les points de $E$.
**Sorties** : Une liste doublement chaînée circulaire contenant les points de l'enveloppe convexe $C$ de $T$ dans l'ordre de leur apparition quand on parcourt $C$ dans le sens direct.

```
1  si n ≤ 3 alors
2  |  retourner la liste circulaire associée à T
3  sinon
4  |  trier les points de T par abscisse croissante;
5  |  Soit T1 le tableau contenant les ⌈n/2⌉ premiers éléments de T;
6  |  Soit T2 le tableau contenant les ⌊n/2⌋ éléments suivants de T;
7  |  l1 := FUSIONHULL(T1);
8  |  l2 := FUSIONHULL(T2);
9  |  (P, Q) := PONTSUP(l1, l2);
10 |  (R, S) := PONTINF(l1, l2);
11 |  Copier dans une liste l les élements de la liste l1 compris entre P à R inclus, puis les
   |  éléments de la liste l2 de S à Q et boucler la liste en liant Q et P;
12 |  retourner l
```

**Analyse de l'algorithme FUSIONHULL**

*Preuve de Terminaison :* Par induction sur la taille du tableau, si le tableau a moins de 3 éléments, l'algorithme est clairement fini. Supposons que l'algorithme FUSIONHULL se termine pour des instances de taille plus petites que $n$, alors pour des instances de taille $n$, comme FUSIONHULL appelle récursivement FUSIONHULL sur les instances strictement plus petites par hypothèse d'induction, il se termine.

*Preuve de Validité :* Par induction sur la taille du tableau, si $n \leq 3$, c'est trivial. Supposons que pour toute instance de taille $n < n_0$, l'algorithme FUSIONHULL renvoie la liste des éléments de l'enveloppe convexe, alors la fusion des deux enveloppes grâce au pont inférieur et au pont supérieur assure que FUSIONHULL renvoie la liste demandée.

*Analyse de la Complexité :* En utilisant le fait que PONTSUP et PONTINF donnent les couples en $O(n)$, on a la formule de récurrence suivante pour $\mathcal{T}$ le nombre d'opérations élémentaires faites par FUSIONHULL sur une instance de taille $n$ : $\mathcal{T}(n) = \mathcal{T}(\lceil n/2 \rceil) + \mathcal{T}(\lfloor n/2 \rfloor) + O(n)$ et donc par le théorème maître, FUSIONHULL fait $O(n \log n)$ opérations pour donner l'enveloppe convexe d'un ensemble de cardinal $n$.