Projet de Pratique de la Programmation et Projet

IUP GMI première année

Calcul d'enveloppes convexes suivant le paradigme « diviser pour régner »

1 L'enveloppe convexe

Notre objectif est de calculer l'*enveloppe convexe* d'un ensemble de points du plan. L'enveloppe convexe d'un ensemble de points est la frontière du plus petit domaine *convexe* contenant tous les points de l'ensemble. Un domaine *convexe* est un domaine tel que tout segment reliant deux points du domaine est entièrement inclus dans le domaine. L'enveloppe convexe d'un ensemble de points est un polygone dont les sommets sont des points de l'ensemble de départ. Tous les points de cet ensemble ne sont pas des sommets de l'enveloppe convexe. Un polygone est convexe si et seulement si tous ces angles sont saillants (inférieurs ou égaux à 180 degrés).

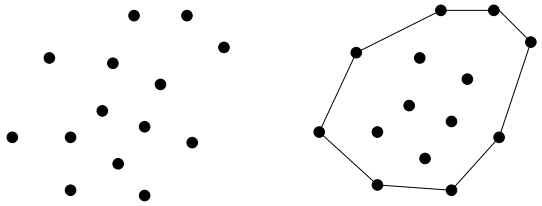

FIG. 1 – (a) un nuage de points dans le plan ; (b) l'enveloppe convexe du nuage de points.

2 L'algorithme

Il existe de nombreux algorithmes pour le calcul des enveloppes convexes. L'algorithme qui nous intéresse fait appel à une stratégie récursive de type *diviser pour régner* (en anglais *divide and conquer*). Si l'ensemble ne contient qu'un seul point (cas trivial) alors l'enveloppe convexe est précisément ce point. Ce cas trivial étant traité, on cherche à calculer l'enveloppe convexe pour un ensemble de N points. On suppose qu'il est possible de l'obtenir pour un ensemble de points deux fois plus petit. On divise donc l'ensemble des N points en deux sous-ensembles de taille comparable. On calcule l'enveloppe convexe de chacun des deux sous-ensembles de points puis on fusionne les deux enveloppes convexes obtenues. On obtient ainsi l'enveloppe convexe de l'ensemble initial. Cet algorithme est appliqué récursivement aux nuages de points jusqu'à ce qu'il ne reste plus que des cas triviaux à traiter.

2.1 Diviser en deux sous-ensembles

Pour tous les cas non triviaux, il faut diviser l'ensemble de points en deux sous-ensembles, gauche et droit, ayant le même nombre de points (à 1 point près) et tel qu'aucun point du sous-ensemble de gauche n'ait une abscisse strictement supérieure à celle d'un point du sous-ensemble de droite[1]. Réciproquement aucun point du sous-ensemble de droite ne doit avoir d'abscisse strictement inférieure à celle d'un point du sous-ensemble de gauche. Pour mettre en œuvre cette scission, il faut trier les points dans l'ordre croissant de leur

[1] Vous supposerez que deux points de l'ensemble de départ ne peuvent pas avoir la même abscisse.

abscisse. Pour un ensemble de départ de N points, les N/2 premiers points de l'ensemble trié appartiennent au sous-ensemble de gauche et les autres points appartiennent au sous-ensemble de droite. Cette division assure qu'aucun des points de l'un des sous-ensembles ne peut être à l'intérieur de l'enveloppe convexe de l'autre sous-ensemble.

2.2 L'algorithme de fusion

2.2.1 Principe

On suppose que l'enveloppe convexe de chacun des deux sous-ensembles a été calculée (cf. figure 2). Il nous reste à en déduire celle de l'ensemble initial, c'est-à-dire celle de la réunion des deux sous-ensembles. Pour

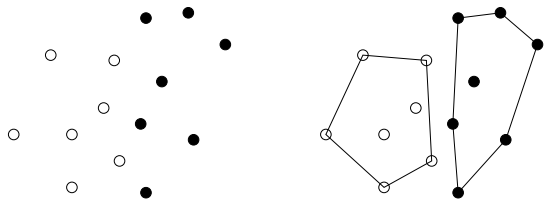

FIG. 2 – (a) Le nuage de points a été scindé en deux (points blancs à gauche et points noirs à droite) ; (b) L'enveloppe convexe de chacun des deux sous-ensembles a été calculée.

cela, il faut ajouter deux segments *tangents* permettant de relier les deux enveloppes convexes et éliminer les segments intérieurs de sorte que l'enveloppe résultant reste convexe et entoure tous les points de l'ensemble de départ (cf. figure 3).

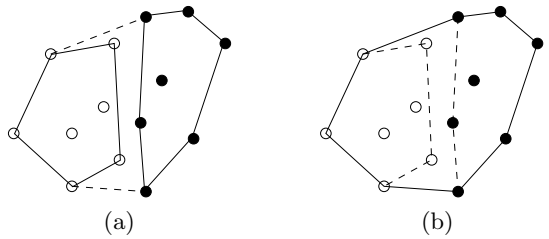

FIG. 3 – (a) Pour fusionner les deux enveloppes, il faut d'abord trouver les deux segments *tangents* reliant les deux enveloppes (en pointillés), et (b) éliminer les arêtes intérieures (en pointillés) qui n'appartiennent pas à l'enveloppe convexe globale.

2.2.2 Un point peut *voir* certains segments

Cette section est une définition qui se révélera utile dans la suite du raisonnement. On suppose qu'un point M situé à l'extérieur d'une enveloppe convexe peut *voir* un segment $[AB]$ de cette enveloppe convexe si et seulement si l'une des deux conditions suivantes est vérifiée :

- en allant de A vers B on parcourt l'enveloppe dans le sens direct (le sens contraire des aiguilles d'une montre) et un observateur se trouvant en A et regardant vers B a le point M à sa droite ;
- en allant de A vers B on parcourt l'enveloppe dans le sens indirect (le sens des aiguilles d'une montre) et un observateur se trouvant en A et regardant vers B a le point M à sa gauche.

De façon équivalente, le point $M(x, y)$ peut *voir* le segment $[AB]$ (avec $A = (x_A, y_A)$ et $B = (x_B, y_B)$) si et seulement si l'une des deux conditions suivantes est satisfaite :

- en allant de A vers B on parcourt l'enveloppe dans le sens direct ET $(y_A - y)(x_B - x) - (y_B - y)(x_A - x) > 0$;

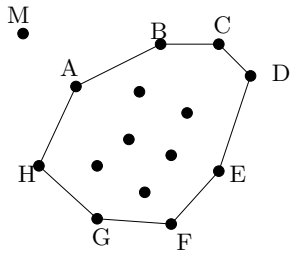

FIG. 4 – Le point M peut *voir* les segments $[HA]$, $[AB]$ et $[BC]$ de l'enveloppe mais ne peut pas *voir* les segments $[CD]$, $[DE]$, $[EF]$, $[FG]$ et $[GH]$.

– en allant de A vers B on parcourt l'enveloppe dans le sens indirect ET $(y_A - y)(x_B - x) - (y_B - y)(x_A - x) < 0$.

2.2.3 Trouver les segments *tangents*

On considère
– un point G qui prend comme valeur les positions successives des sommets de l'enveloppe convexe gauche ;
– un point D qui prend comme valeur les positions successives des sommets de l'enveloppe convexe droite.
Initialement le point G se trouve au point G_0 le sommet d'abscisse maximale dans l'enveloppe de gauche. Le point D se trouve au point D_0 d'abscisse minimale dans l'enveloppe de droite.

Le segment *tangent* supérieur
Le point G parcourt l'enveloppe de gauche dans le sens direct. $G.suiv\_dir$ est le sommet qui suit G dans ce sens. Le point D parcourt l'enveloppe de droite dans le sens indirect. $D.suiv\_indir$ est le sommet qui suit D dans ce sens. Voici l'algorithme permettant de trouver le segment tangent supérieur :

$G \leftarrow G_0$;
$D \leftarrow D_0$;
Faire
{
    trouvé $\leftarrow$ vrai ;
    TantQue((D n'est pas le seul point de l'ensemble de droite) && (G voit le segment $[DD.suiv\_indir]$))
        $D \leftarrow D.suiv\_indir$; trouvé $\leftarrow$ faux ;
    TantQue((G n'est pas le seul point de l'ensemble de gauche) && (D voit le segment $[GG.suiv\_dir]$))
        $G \leftarrow G.suiv\_dir$; trouvé $\leftarrow$ faux ;
}
TantQue(trouvé == faux) ;

à la fin du déroulement de cet algorithme, le segment $[GD]$ sera le segment tangent supérieur.

Le segment *tangent* inférieur
Pour trouver le segment tangent inférieur, on repart de G_0 et de D_0, mais cette fois-ci on parcourt le polygone de gauche dans le sens indirect et le polygone de droite dans le sens direct.

3 Le travail demandé

Vous devrez concevoir et écrire un programme capable de générer aléatoirement un ensemble de points dans le plan et d'en calculer l'enveloppe convexe. Cette enveloppe convexe devra prendre la forme d'une

liste circulaire doublement chaînée liant ensemble un sous-ensemble des points générés au départ. Un point du plan sera représenté par deux entiers. Vous trouverez à l'url `http://icps.u-strasbg.fr/~vivien/Enseignement/PPP-2001-2002/Dessin2D.tgz` un programme qui, étant donné un ensemble de points, produit un fichier image où ces points sont soit représentés seuls, soit liés les uns aux autres par des segments de droite. Vous pourrez utiliser ce programme pour vérifier le comportement de vos algorithmes.

Le travail demandé sera effectué obligatoirement en binômes.

Votre travail sera évalué au vu :
- du travail de conception ;
- de votre programme lui-même ;
- d'une démonstration de votre programme sur machine.

3.1 La conception

L'étape de la conception consistera à analyser ce problème et à décomposer l'algorithme en tâches distinctes, et chaque tâche en sous-tâches et ainsi de suite jusqu'à ce que les tâches à effectuer soient triviales. Il faudra alors affecter chaque tâche à une fonction dont vous fournirez le prototype. Le produit de cette étape de conception sera :
- un ou plusieurs fichiers entête indiquant les structures de données que vous comptez utiliser et les prototypes des fonctions qui vous permettront de construire votre programme ;
- un graphe indiquant les liens de dépendance entre vos fonctions ;
- la répartition des fonctions entre les deux membres du binôme ;
- le calendrier ou l'ordre dans lequel seront écrites les fonctions et qui permettra au programme d'être à tout moment testable.

Les fichiers entête, les graphes, la répartition et le calendrier devront être rendu à Arash Habibi sur support papier avant le **mardi 30 avril 2002 à 8h00** du matin (date et heure à laquelle vous aurez le corrigé de cette première étape). Les travaux rendus après cette date et cette heure ne seront pas pris en compte.

3.2 Le programme

Votre programme et tout ce qui est nécessaire pour sa compilation (fichiers sources, fichiers entêtes, makefile, etc.) devra être placé dans un seul répertoire dont le nom sera composé du nom des deux membres du binôme. Ce fichier devra être archivé, compressé (format `tgz`) et envoyé en attaché mail à Arash Habibi à l'adresse `habibi@ada.u-strasbg.fr` avant le **jeudi 23 mai 2002 à minuit**, la date d'arrivée du mail sur la machine *ada* faisant foi. En cas d'avarie sur la machine *ada*, vous pourrez envoyer vos travaux à l'adresse `habibi@dpt-info.u-strasbg.fr`.

3.3 La démonstration

Les démonstrations auront lieu pendant la séance de TP du vendredi 24 mai 2002. Lors du passage, vous n'utiliserez pas votre compte sur ada, mais un compte fourni par l'examinateur, et dans lequel il aura copié le travail que vous lui aurez envoyé par mail. C'est ce programme que vous devrez compiler et faire fonctionner. L'examinateur pourra vous demander de calculer l'enveloppe convexe d'un ensemble plus ou moins grand de points. Enfin, il pourra également vous demander de commenter le code source lui-même en expliquant le fonctionnement et l'intérêt de certaines parties.

3.4 Remarque générale

Ce projet est un projet long. Donc personne ne vous en voudra de ne pas avoir tout fini. Par contre on vous en voudra d'avoir *presque* tout fini mais de ne rien pouvoir présenter. Un des objectifs de ce projet est de vous exercer au test systématique et à la sauvegarde après chaque étape, avant de passer à l'étape suivante. Ceci vous permettra de garder votre programme dans un état testable et présentable à chaque instant.