Feuille d'exercices n°18 de l'option d'informatique.

I L'ordre «radar»

Énoncé : M_1 et M_2 sont deux points d'un plan euclidien orienté, distincts de l'origine. On connaît leurs coordonnées (X_1, Y_2) et (X_2, Y_2) dans un repère orthonormé direct d'origine O. On fait décrire à une demi-droite d'origine O un tour complet, dans le sens trigonométrique, en partant de la demi-droite Oy' (d'angle polaire $-\pi/2$). Quel est le premier des deux points que cette demi-droite rencontrera?

Nous allons construire un algorithme qui permet de déterminer ce point. Remarquons d'abord que si les vecteurs $\overrightarrow{OM}_1$ et $\overrightarrow{OM}_2$ ont même direction et même sens, les points sont atteints en même temps, puisqu'ils sont alors sur une même demi-droite d'origine O. Ceci se produit ssi les deux conditions suivantes sont satisfaites:

* les deux vecteurs sont liés
* leur produit scalaire est positif

ce qui se traduit algébriquement par:

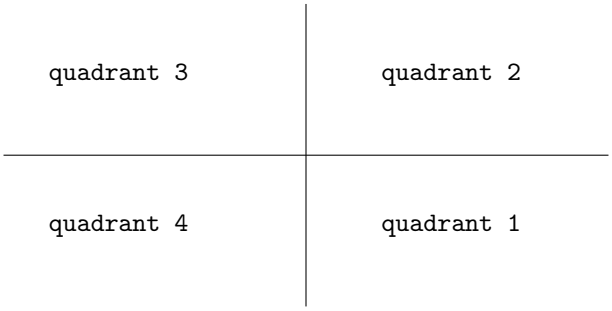
$$X_1.Y_2 - X_2.Y_1 = 0$$
$$X_1.X_2 + Y_1.Y_2 > 0$$

Ce cas de figure éliminé, voyons comment aborder le cas général. On ne peut se contenter de comparer des pentes, pour deux raisons: deux vecteurs opposés ont même pente; et certains vecteurs n'ont pas de pente (ceux qui dirigent l'axe $y'Oy$).

Nous allons décomposer le plan privé de O en quatre quadrants numérotés de 1 à 4, définis par les équations suivantes:

* quadrant 1 : $X \geqslant 0$ et $Y < 0$
* quadrant 2 : $X > 0$ et $Y \geqslant 0$
* quadrant 3 : $X \leqslant 0$ et $Y > 0$
* quadrant 4 : $X < 0$ et $Y \leqslant 0$

Ainsi le quadrant numéro k contient les extrémités des vecteurs d'origine O et d'angle polaire compris entre $(k-2)\pi/2$ inclus et $(k-1)\pi/2$ exclus. Il est clair que les quatre quadrants réalisent une partition du plan privé de l'origine O.



Manifestement, si M_1 et M_2 sont dans des quadrants distincts, le premier point rencontré est celui situé dans le quadrant de plus petit numéro. Supposons maintenant que les deux points sont situés dans un même quadrant.

Quelque soit le quadrant, la comparaison des pentes Y_1/X_1 et Y_2/X_2 suffit pour conclure: le point de plus faible pente est rencontré le premier. On peut toujours (au moins mathématiquement) calculer ces pentes car les abscisses sont non nulles.

En caml, on considère la notion de point définie par :

```
1  type point = {x : float; y : float};;
```

1. Écrire une fonction `quadrant` qui à partir d'un point M donné par ses coordonnées réelles X et Y retourne le numéro du quadrant contenant le point M.

```
1  #quadrant;;
2  - : point -> int = <fun>
   #quadrant {x=1.; y=(-.2.5)};;
4  - : int = 1
```

2. Écrire une fonction `OrdreRadar` qui, étant données les coordonnées des deux points M_1 et M_2, s'assure que ces points sont distincts, et applique l'algorithme précèdent, pour fournir l'une des trois réponses possibles:

* le point M_1 est rencontré avant le point M_2
* les points M_1 et M_2 sont rencontrés en même temps
* le point M_2 est rencontré avant le point M_1

est retourne respectivement les valeurs -1, 0 et +1 selon le cas.

Une seule astuce : pour éviter les divisions, on compare les produit $Y_1.X_2$ et $Y_2.X_1$, au lieu des quotients Y_1/X_1, et Y_2/X_2. Comme on s'est ramené au cas où les abscisses sont positives, ceci est licite.

II Enveloppe convexe d'un ensemble fini de points du plan

Soit A une partie non vide du plan affine. On appelle enveloppe convexe de A la plus petite partie convexe du plan contenant A. On peut aussi la définir comme l'intersection de toutes les parties convexes du plan qui contiennent A ou encore comme l'ensemble des barycentres de points de A affectés de coefficients tous positifs.

Si A est une partie bornée, il existe une méthode «mécanique» pour construire l'enveloppe convexe: imaginons un disque D contenant A, et posons sur le plan une ficelle faisant le tour de ce disque; rapprochons les deux extrémités libres et tirons; lorsque la ficelle et est tendue, elle délimite l'enveloppe convexe de A.

Une variante de cette méthode va nous permettre de construire un algorithme, valable lorsque A est une partie finie. Déterminons d'abord un point M_1 dont on est certain qu'il appartient à la frontière de l'enveloppe convexe (on peut par exemple, après avoir fixé un repère, choisir un point d'abscisse minimale). Attachons une ficelle à ce point, et choisissons comme sens de parcours le sens trigonométrique.

Imaginons maintenant que nous marchons parallèlement à l'axe Oy, dans le sens des y décroissants, jusqu'à ce que nous dépassions tous les points de A. Tendons la ficelle, et «rabattons» progressivement celle-ci vers la droite sur A : ceci revient à considérer une demi-droite d'origine M_1, dont on fait augmenter la pente en partant de $-\infty$.

Le premier point qui sera rencontré par la ficelle dans ce mouvement appartiendra à l'enveloppe convexe; ce point M_2 est tel que la pente de la droite M_1M_2 est la plus petite possible. Il se peut même que cette pente soit $-\infty$, si plusieurs points ont la même abscisse que M_1 : dans ce cas, on choisit celui qui a la plus petite ordonnée, ainsi c'est le plus éloigné de M_1, dans le sens de parcours.

Partant de M_2, on recommence, mais cette fois la ficelle est tendue dans le prolongement de la demi-droite que l'on vient de déterminer. On continue jusqu'au moment où l'on retrouve le point M_1 : la boucle est alors bouclée, et on a la liste des points $M_1M_2 \dots M_n$ de l'enveloppe de A.

À chaque fois que l'on trouve plusieurs points de contact, on choisit celui qui est plus éloigné du point de base; et, pour éviter une étape superflue, on choisit M_1 d'ordonnée maximale, parmi les points d'abscisse minimale. Pour mesurer l'éloignement, on peut utiliser la distance euclidienne, mais, comme les points sont alignés, on peut se contenter de prendre:

$$d(M_1, M_2) = |X_2 - X_1| + |Y_2 - Y_1|$$

où X_1, Y_1, X_2, Y_2 sont les coordonnées de M_1 et M_2. Cette quantité se calcule bien plus simplement, et permet elle aussi de classer les points, lorsqu'ils sont alignés.

Comme on ne peut pas manipuler $+\infty$ et $-\infty$ (ni même des nombres très grands), on compare les pentes de la façon suivante: soient M_1 le point de départ, M_2 et M_3 les deux points d'arrivée possibles; on devrait comparer les rapports:

$$\frac{Y_3 - Y_1}{X_3 - X_1} \text{ et } \frac{Y_2 - Y_1}{X_2 - X_1}$$

mais l'un ou l'autre peut ne pas être calculable. Quitte à changer à la fois le signe du numérateur et du dénominateur, on peut supposer que les deux fractions ont un dénominateur positif. On compare alors les produits:

$$(Y_3 - Y_1).(X_2 - X_1) \text{ et } (Y_2 - Y_1).(X_3 - X_1)$$

On détermine donc le point M_1, puis on fait le tour de l'enveloppe convexe: à la k-ième étape, on cherche le sommet M_{k+1} qui suit le sommet M_k trouvé précédemment. Lorsque ce point est M_1, c'est que l'on a terminé. Le programme dresse alors la liste des coordonnées des sommets de l'enveloppe convexe, puis dessine dans un même repère les points du nuage et l'enveloppe.

3. Écrire une fonction `genere_pts` qui génère aléatoirement une liste de points dont les coordonnées (x, y) vérifieront $0 \leqslant x \leqslant$ `size_x()` et $0 \leqslant y \leqslant$ `size_y()` .

```
#genere_pts 5;;
- : point list =
 [{x = 600.0; y = 400.0}; {x = 600.0; y = 400.0}; {x = 600.0; y = 400.0};
```

4. Écrire une fonction `affiche_pts` qui affiche une liste de points dans la fenêtre graphique. On supposera que les coordonnées (x, y) des points de la liste vérifient $0 \leqslant x \leqslant$ `size_x()` et $0 \leqslant y \leqslant$ `size_y()`. Pour plus de lisibilité, on pourra représenter les points par des disques de rayon 2 pixels.

5. Écrire une fonction `point_depart` qui à partir d'une liste de points, retourne le point M_1 décrit dans l'algorithme précédent.

```
#point_depart [{x=200.;y=200.};{x= 100.;y=200.};{x= 100.;y = 150.};
                   {x= 100.;y= 100.};{x=200.;y= 100.}];;
- : point = {x = 100.0; y = 200.0}
```

6. Écrire une fonction `point_suivant` qui à partir d'un point `M`, d'une direction `dir` et d'une liste de points, retourne le premier point rencontré dans le balayage radar centré au point M à partir de la direction `dir`.

```
#point_suivant {x = 100.; y = 100.} {x=100.;y=0.}
   [{x=200.;y=200.};{x= 100.;y=200.};{x= 100.;y = 150.}; {x= 100.;y= 100.};{x=200.;y= 100.}];;
- : point = {x = 200.0; y = 200.0}
```

7. Écrire une fonction `envconvexe` qui à partir d'un ensemble fini A de points donnés par une liste de points retourne la liste des points représentant l'enveloppe de l'ensemble A.
8. Écrire une fonction `dessine_polygone` qui dessine le polygone définie par une liste de points.
9. Écrire une fonction qui représente une liste de points et son enveloppe convexe dans la fenêtre graphique.

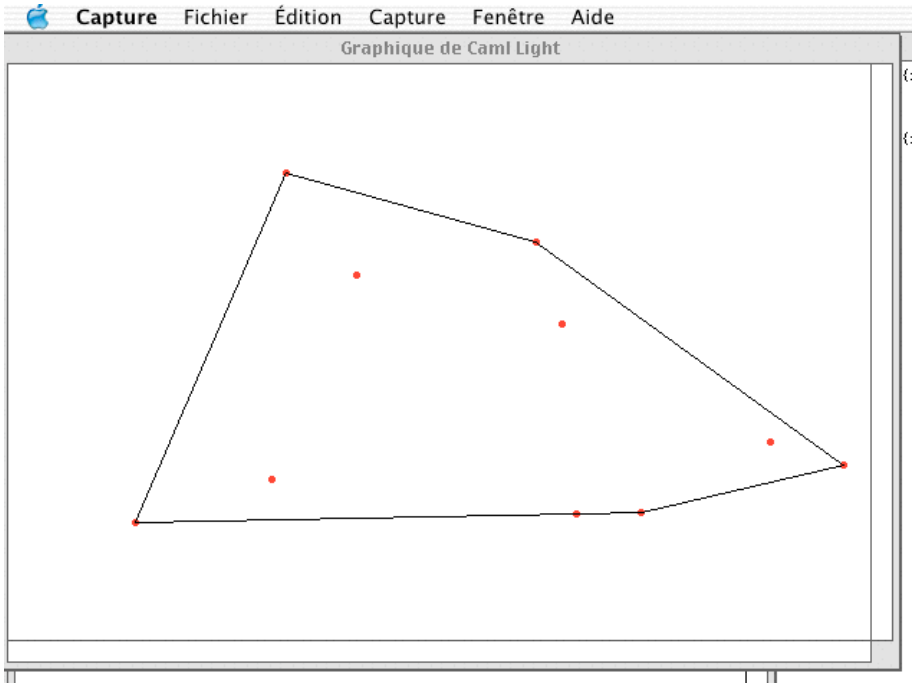


FIG. 1 – *Exemple*

Référence bibliographique : «l'informatique en SUP et en SPÉ» par Bruno Petazzoni édition ellipses.