Enveloppe convexe, algorithme de Graham

Vincent Picard, enseignant CPGE au Lycée Claude Fauriel

La géométrie algorithmique est le domaine de l'informatique qui s'intéresse comme son nom l'indique à la résolution informatique de problèmes de géométrie. Nous présentons ici un des problèmes les plus connus de ce domaine : le calcul de l'enveloppe convexe d'un ensemble de points. Il existe différents algorithmes permettant de construire cette enveloppe, nous nous intéresseront à celui de Graham publié en 1972.

1 Structure de pile modifiable

L'algorithme reposera sur l'utilisation d'une pile dont on est capable de lire le premier élément (sommet) mais également le second élément. On se propose dans un premier temps de programmer cette structure.

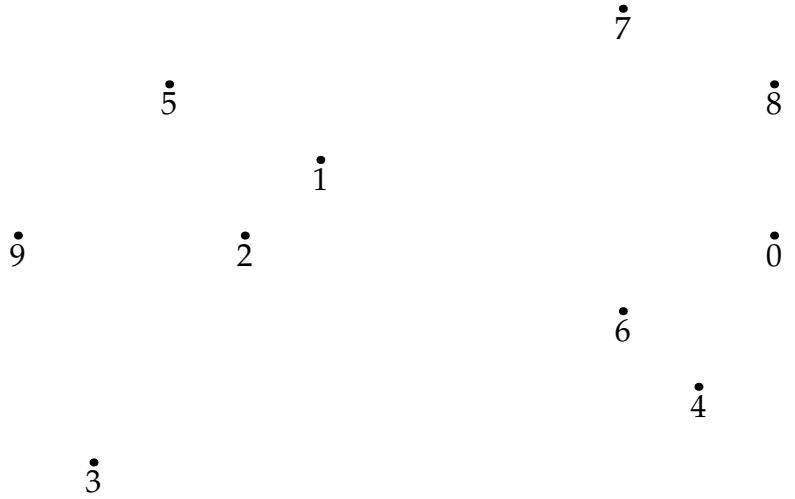
```
(* Nos piles seront modifiables *)
type 'a pile = 'a list ref;;

(* Créé une nouvelle pile *)
val creer_pile : unit -> 'a pile

(* Teste si la pile est vide *)
val pile_vide : 'a pile -> bool

(* Insère un élément dans la pile *)
val inserer : 'a -> 'a pile -> unit

(* Supprime l'élément en sommet de pile *)
val enlever : 'a pile -> unit

(* Retourne la valeur du sommet de la pile, sans changer la pile *)
val premier : 'a pile -> 'a

(* Retourne la valeur du deuxieme element dans la pile, sans la changer *)
val deuxieme : 'a pile -> 'a

(* Construit une liste contenant les éléments de la pile dans
 * l'ordre en commençant par le sommet de pile
 *)
val listepile : 'a pile -> 'a list
```

Question 1

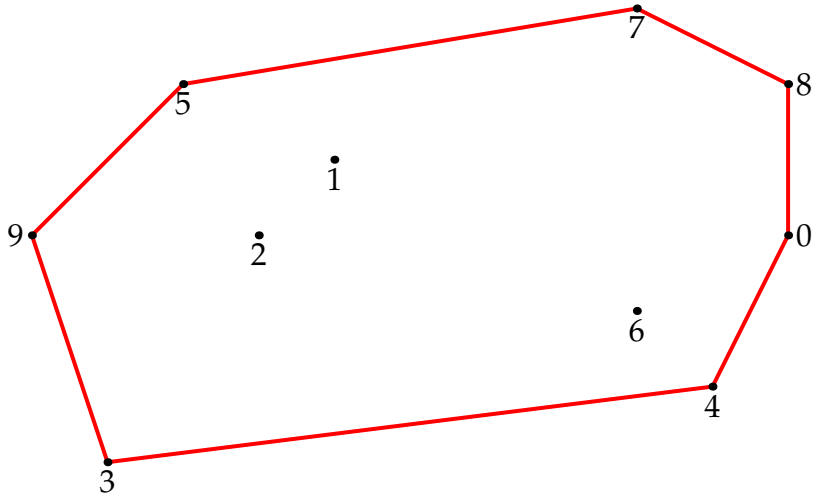
Programmer les fonctions de la structure de pile.

2 Algorithme de Graham

On considère un ensemble de points du plan.



On souhaite déterminer l'enveloppe convexe de cet ensemble de points. L'enveloppe convexe est un polygone convexe qui correspond à la frontière de la plus petite partie convexe contenant l'ensemble des points.



On représentera, sans nuire à la généralité, un point par un couple d'entiers. L'utilisation des entiers plutôt que les flottants évite les considérations d'approximations du calcul flottant. Une instance du problème sera représentée sous forme d'une liste de points dont on cherche à calculer l'enveloppe convexe. La solution sera aussi fournie sous forme d'une liste de points correpsondant au tracé rouge ci-dessus.

```
type point = int * int
```

Le problème donné en exemple correspond à:

```
let exemple =
  let z0 = (10, 3) in
  let z1 = (4, 4) in
  let z2 = (3, 3) in
  let z3 = (1, 0) in
  let z4 = (9, 1) in
  let z5 = (2, 5) in
  let z6 = (8, 2) in
  let z7 = (8, 6) in
  let z8 = (10, 5) in
  let z9 = (0, 3) in
    [z0; z1; z2; z3; z4; z5; z6; z7; z8; z9]
;;
```

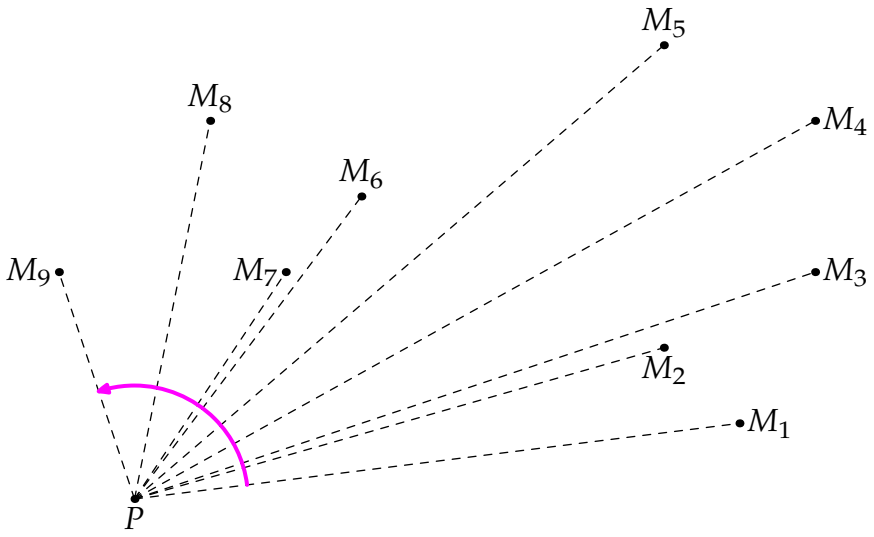
2.1 Détermination du pivot

L'algorithme de Graham commence par déterminer un point pivot noté P qui sera le point le plus bas. S'il existe plusieurs points les plus bas, on prendra celui le plus à gauche. Dans notre exemple le pivot correspondra donc au point numéroté 3.

> **Question 2**
>
> Écrire une fonction `trouve_pivot : point list -> (point * point list)`. Cette fonction retournera un couple `(pivot, reste)` où `pivot` est le point pivot et `reste` est la liste des autres points.

2.2 Tri des points

Les autres points $\{M_i,\ i = 1 \dots 9\}$ seront triés par ordre croissant de l'angle que forme PM_i avec l'horizontale. On obtient alors la figure suivante :



Le calcul et la comparaison des angles ne sont pas aisés, surtout s'ils nécessitent l'utilisation des flottants. Pour se faciliter les choses nous allons utiliser le déterminant.

Question 3

Écrire une fonction `vect : point -> point -> (int * int)` telle que `vect a b` retourne le vecteur $\overrightarrow{ab}$. Un vecteur sera représenté par le couple de ses coordonnées dans la base canonique. Écrire ensuite une fonction `det : (int * int) -> (int * int) -> int` calculant le déterminant de deux vecteurs par rapport à la base canonique.

On remarque alors, pour l'ordre proposé, qu'un point A est situé avant un point B ($A \preceq B$) si et seulement si $\det\left(\overrightarrow{PA}, \overrightarrow{PB}\right) \geq 0$.

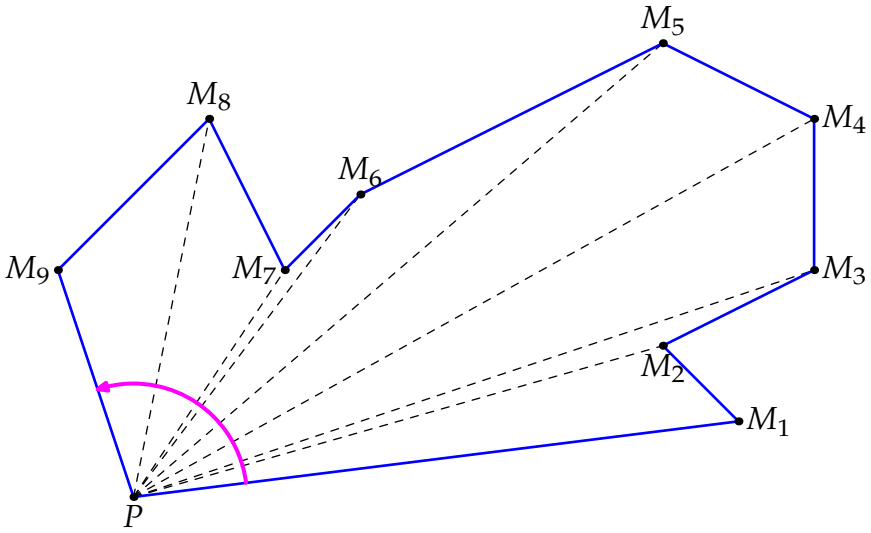
Question 4

Écrire une fonction `tri_fusion : point -> point list -> point list` telle que `tri_fusion p reste` trie les points de la liste `reste` selon l'ordre $\preceq$, en utilisant le tri fusion. L'argument `p` correspond au point pivot.

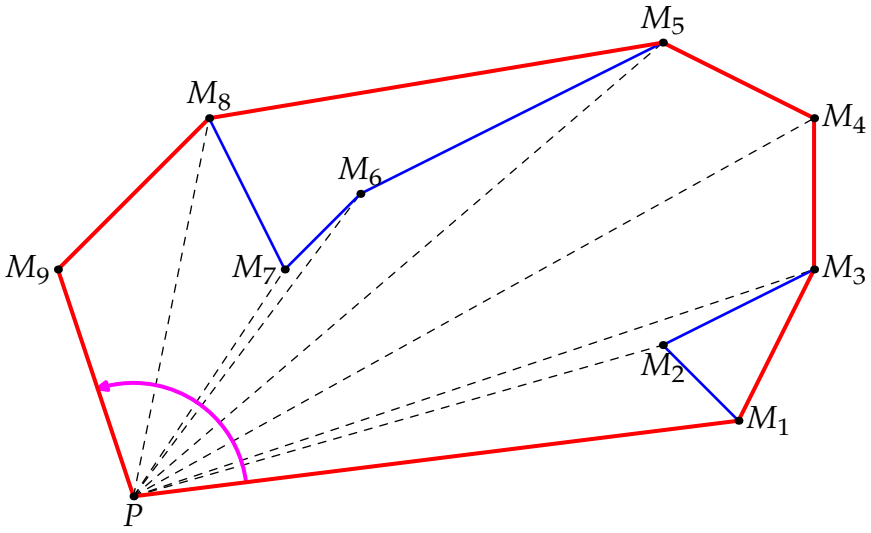
Vérifiez vos fonctions sur l'exemple proposé.

2.3 Algorithme

Si on relie les points $P, M_1, M_2, \ldots, M_8, M_9, P$, on n'obtient pas encore l'enveloppe convexe :



L'idée de l'algorithme de Graham est de détecter les points à enlever dans ce tracé bleu pour obtenir l'enveloppe convexe. Pour cela on remarque que dans le parcours bleu, à chaque point où l'on tourne vers la droite (par exemple M_2), ce point n'appartient pas à l'enveloppe convexe. On élimine alors ces points jusqu'à ce qu'il n'y ait plus que des virages vers la gauche. On obtient alors l'enveloppe convexe.



On constate dans l'exemple que pour détecter que M_6 n'est pas dans l'enveloppe convexe, il faut d'abord avoir enlevé M_7 pour remarquer un virage à droite (virage M_5, M_6, M_8). Ainsi quand on élimine un point, il faut vérifier si cela ne créé pas un virage à droite pour le point précédent. Cela donne l'idée d'utiliser une pile.

Pour tester si les points ABC forment un virage vers la gauche ou la droite, il suffit d'éudier le signe du déterminant $\det\left(\overrightarrow{AB}, \overrightarrow{BC}\right)$.

On décrit l'algorithme de Graham ainsi :

1. Déterminer le pivot P.
2. Trier les points restants $M_1, \dots, M_{n-1}$.
3. Créer une pile vide *pile*.
4. Insérer P puis M_1 dans la *pile*
5. Pour i allant de 2 à $n-1$:
   a. Soit B le sommet de pile et A l'élément d'en-dessous.
   b. Si ABM_i est un virage à gauche strict : inserer M_i dans la *pile*
   c. Si ABM_i est un virage à droite : enlever B de la *pile*, recommencer l'itération de la boucle avec le même i

Question 5

Écrire une fonction `graham : point list -> point list` calculant l'enveloppe convexe d'un ensemble de points. Cette fonction effectuera le parcours de la liste des points non-pivot (étape 5) à l'aide d'une fonction auxiliaire récursive prenant en argument la liste des points restants à considérer (pas de tableau).

Question 6

Quelle est la complexité asymptotique de l'algorithme de Graham dans le pire cas en fonction du nombre de points n ?