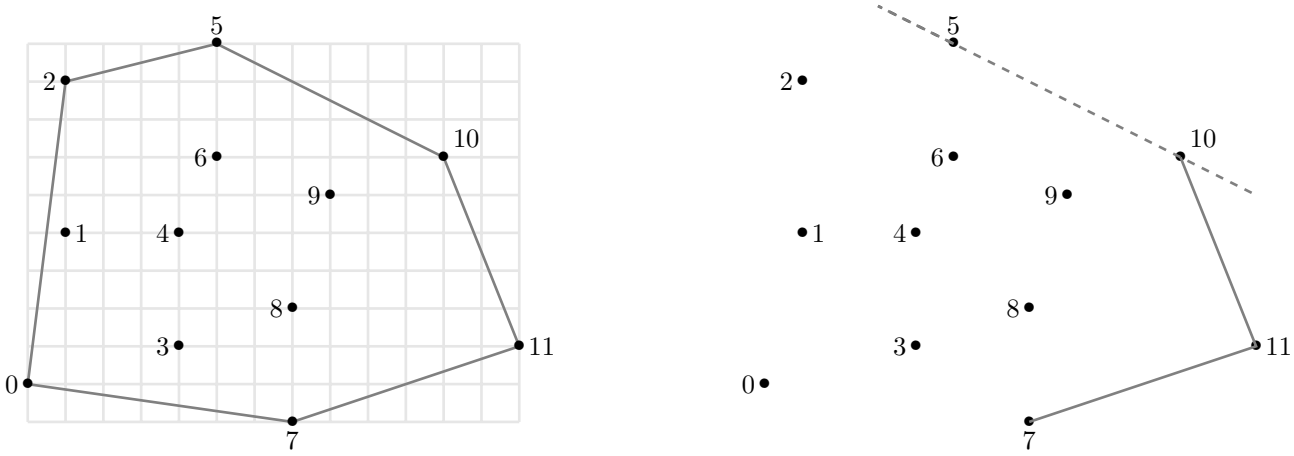
DM15

Ce sujet étudie un algorithme de détermination de l'*enveloppe convexe* d'un nuage de points dans le plan affine. Un ensemble $C \subset \mathbb{R}^2$ est dit *convexe* si et seulement si pour toute paire de points $p, q \in C$, le segment de droite $[p, q]$ est inclus dans C. L'enveloppe convexe d'un ensemble $P \subset \mathbb{R}^2$ est le plus petit convexe contenant P. Dans le cas où P est un ensemble fini, appelé *nuage de points*, le bord de cette enveloppe est un polygone convexe dont les sommets appartiennent à P, comme illustré sur la figure 1a.



(a) Nuage de points et bord de son enveloppe convexe.

(b) Mise à jour du paquet cadeau : insertion du point 5.

Figure 1 – Enveloppe convexe d'un nuage de points.

Dans toute la suite, on suppose que le nuage de points P est de taille $n \geqslant 3$ et qu'il ne contient pas 3 points distincts alignés. Ces hypothèses simplifient les calculs en ignorant les cas pathologiques, comme par exemple la présence de 3 points alignés sur le bord de l'enveloppe convexe. Les coordonnées des points, supposées entières, sont données dans une base orthonormée du plan, orientée dans le sens direct.
En OCaml, les points sont définis par le type `point`. Le nuage de points est stocké dans un tableau. On donne ci-dessous le codage du nuage de points de la figure 1a.

```
type point = {x : int; y : int};;
let tab_points = [|
  {x = 0; y = 0}; {x = 1; y = 4}; {x = 1; y = 8}; {x = 4; y = 1};
  {x = 4; y = 4}; {x = 5; y = 9}; {x = 5; y = 6}; {x = 7; y = - 1};
  {x = 7; y = 2}; {x = 8; y = 5}; {x = 11; y = 6}; {x = 13; y = 1}
|];;
```

L'accès à un élément du tableau se fait à l'aide de la notation pointée. Celui aux coordonnées des points se fait de la même façon, en utilisant les notations introduites par le type `point`.

```
#tab_points.(3);;
- : point = {x = 4; y = 1}
#tab_points.(3).x;;
- : int = 4
#tab_points.(3).y;;
- : int = 1
```

Préliminaires

On rappelle la définition de l'*ordre lexicographique* $<_{lex}$ sur des couples d'entiers.

$$(x_1, y_1) <_{lex} (x_2, y_2) \iff \begin{cases} x_1 < x_2 \\ \text{ou} \\ (x_1 = x_2) \text{ et } (y_1 < y_2) \end{cases}$$

Question 1. Écrire une fonction `lexico p1 p2` de signature `point -> point -> bool` qui prend en argument deux points `p1` et `p2` et qui renvoie `true` si $p_1 <_{lex} p_2$, `false` dans le cas contraire.

Question 2. Écrire une fonction `plus_bas_point p1 p2` de signature `point -> point -> point` qui renvoie le plus bas des points `p1` et `p2`, c'est-à-dire celui de plus petite ordonnée. En cas d'égalité, la fonction renvoie celui de plus petite abscisse.

Question 3. Écrire une fonction `plus_bas tab` de signature `point array -> point` qui prend en paramètre un tableau de points `tab` et qui renvoie le point le plus bas.

Étant donnés trois points p_i, p_j, p_k du nuage P, distincts ou non, l'*orientation* du triplet (p_i, p_j, p_k) est définie par le signe du déterminant de la matrice 2×2 formée par les coordonnées des vecteurs $\overrightarrow{p_i p_j}$ et $\overrightarrow{p_i p_k}$, comme illustré sur la figure 2. On appelle *test d'orientation* l'opération qui renvoie $+1$ si la séquence (p_i, p_j, p_k) est orientée positivement, -1 si elle orientée négativement et 0 si les trois points sont alignés.

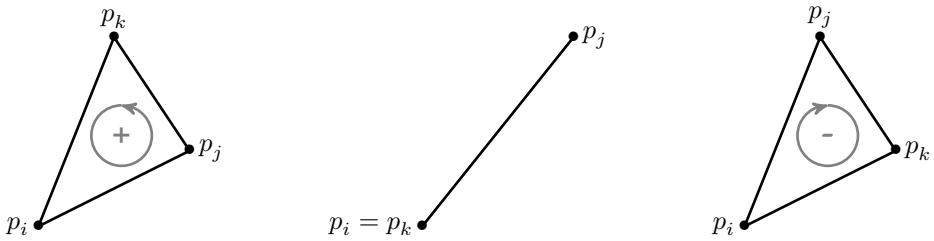


Figure 2 – Test d'orientation sur la séquence (p_i, p_j, p_k) : positif à gauche, nul au centre, négatif à droite.

Question 4. Avec la numération des points de la figure 1a, donner le résultat du test d'orientation pour les choix de triplets de points suivants : $(0, 3, 4)$ et $(8, 9, 10)$.

Question 5. Écrire une fonction `orient p1 p2 p3` de signature `point -> point -> point -> int` qui prend en paramètres trois points `p1`, `p2`, `p3` et qui renvoie -1, 0 ou $+1$ selon le résultat du test d'orientation sur la séquence (p_1, p_2, p_3).

Algorithme du paquet cadeau

Proposé par R. Javis en 1973, l'*algorithme du paquet cadeau* consiste à envelopper progressivement le nuage de points P en faisant pivoter une droite tout autour. Partant du point le plus bas du nuage P, chaque étape de la procédure sélectionne le prochain point du nuage P à insérer. À la fin du processus, on obtient exactement l'enveloppe convexe de P. Sa mise en œuvre est illustrée sur la figures 1b.
La détermination de proche en proche des points de l'enveloppe suit l'algorithme suivant. Notons p_i le dernier point inséré dans le paquet cadeau. Par exemple, $i = 10$ dans l'exemple de la figure $1b$. Considérons la relation binaire d'ordre total, notée $\preccurlyeq$, définie sur l'ensemble $P \setminus \{p_i\}$ par :

$$p_j \preccurlyeq p_k \iff \texttt{orient pi pj pk} \leqslant 0$$

Le prochain point à insérer (point 5 sur la figure 1b) est l'élément maximum pour la relation d'ordre $\preccurlyeq$. Il peut se calculer en temps linéaire par une simple itération sur les points de $P \setminus \{p_i\}$.

Question 6. Justifier brièvement que $\preccurlyeq$ soit une relation d'ordre total sur l'ensemble $P \setminus \{p_i\}$, c'est-à-dire :

- (réflexivité) pour tout $j \neq i$, $p_j \preccurlyeq p_j$,
- (antisymétrie) pour tous $j, k \neq i$, $p_j \preccurlyeq p_k$ et $p_k \preccurlyeq p_j$ implique $j = k$,
- (transitivité) pour tous $j, k, l \neq i$, $p_j \preccurlyeq p_k$ et $p_k \preccurlyeq p_l$ implique $p_j \preccurlyeq p_l$,
- (totalité) pour tous $j, k \neq i$, $p_j \preccurlyeq p_k$ ou $p_k \preccurlyeq p_j$.

Question 7. Écrire une fonction `prochain p1 tab` de signature `point -> point array -> point` qui prend en paramètre un point `p1` et un tableau `tab` et qui renvoie le point de `tab` à insérer dans le paquet cadeau, qui suit `p1`. Le temps d'exécution de la fonction doit être majoré par une constante fois la taille de `tab`.

En combinant les fonctions `prochain` et `plus_bas`, il est possible de déterminer l'enveloppe convexe de P. On commence par insérer le point p_i d'ordonnée la plus basse puis on itère le processus de mise à jour du paquet cadeau jusqu'à ce que le prochain point soit de nouveau p_i. À ce moment-là, on renvoie le paquet cadeau comme résultat sans insérer p_i une seconde fois.

Question 8. Écrire une fonction `convex tab` de signature `point array -> point list ref` qui prend en paramètre le tableau `tab` représentant le nuage P et qui renvoie une référence de liste contenant les points des sommets du bord de l'enveloppe convexe de P, sans doublon.

Question 9. Quelle est une majoration du temps d'exécution de votre fonction ?