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

Ensemble de points en positions convexes

1 Préliminaires

Dans ce TD, on manipulera des ensembles de points et des polygones convexes. Les points seront tous à coordonnées entières.

```
#type point == int*int;;
#type liste_de_points == point list;;
#type polygone == point list;;
```

1.1 Égalité de deux polygones

On dira que deux polygones sont égaux si leurs sommets ont les mêmes coordonnées et apparaissent dans le même ordre dans les listes (le premier élément de la liste du premier polygone correspond à un élément de la liste du second polygone, mais pas forcément au premier). Écrire une fonction `estEgal` de type `polygone -> polygone -> bool = <fun>` qui teste l'égalité des deux polygones.

1.2 Condition de non alignement

Écrire une fonction `aligne` de type `point -> point -> point -> bool = <fun>` qui étant donné trois points A, B et C retourne `true` si les trois points sont alignés et `false` sinon.

Écrire alors une fonction `sansAlignement` de type `liste_de_points -> true = <fun>` qui étant donné une liste de points k retourne `true` si cette liste ne contient aucun triplet de points alignés et `false` sinon.

1.3 convexité d'un polygone

On rappelle qu'un triangle `ABC` est *direct* si la mesure de l'angle $\widehat{\overrightarrow{AB}, \overrightarrow{AC}}$ (prise entre $-\pi$ et π) est positive. Le plan étant bien sûr orienté. La fonction :

```
1  let tourneAGauche (A:point) (B:point) (C:point) =
2    let xa = fst A and ya = snd A and xb = fst B and yb = snd B
3    and xc = fst C and yc = snd C in
4    let X1 = xb - xa and Y1 = yb - ya and X2 = xc -xa and Y2 = yc - ya in
5            X1*Y2 - X2*Y1 > 0;;
```

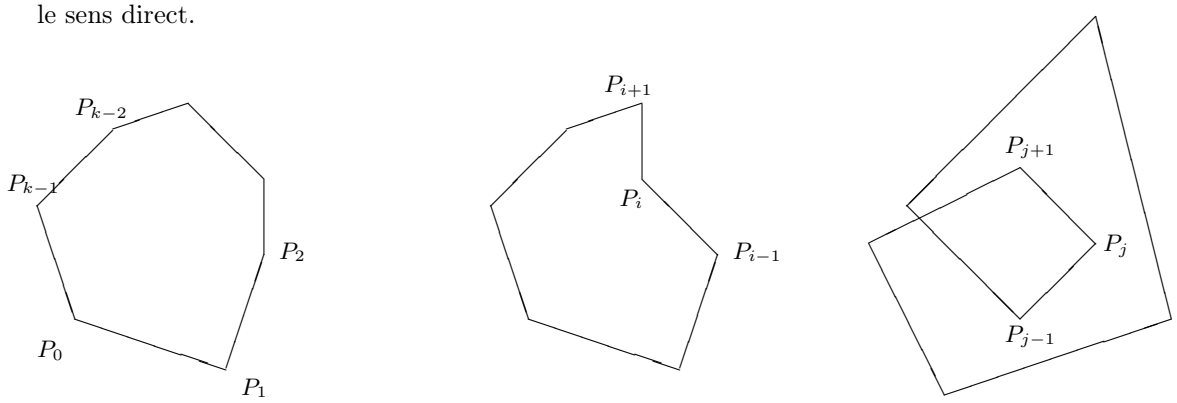
nous dit si les trois points A, B et C forment un triangle direct ou indirect (on supposera que trois points ne sont jamais alignés). Cette méthode peut donc également servir à savoir de quel coté de la droite orienté (AB) se trouve le point C. Si le triangle (ABC) est direct, on dira que le point C est *à gauche* de la droite (AB).

Soit $(P_0, \ldots, P_{k-1})$ une suite de points, où $\forall k \in [\![0, k-1]\!],\ P_i = (x_i, y_i)$. On admettra sans démonstration que le polygone $(P_0, \ldots, P_{k-1})$ est convexe si les deux conditions suivantes sont satisfaites (les indices sont bien sûr calculés modulo k) :

(1) pour tout i de $[\![0, k-1]\!]$, les triplets de points (P_i, P_{i+1}, P_{i+2}) sont des triangles directs,

(2) on a $x_{i+1} \geq x_i$ et $x_{i+1} > x_{i+2}$ pour une et une seule valeur de i.

Ces conditions sont illustrées par la figure 1. En utilisant la fonction `tourneAGauche`, écrire une fonction `estConvexe` du type `polygone -> bool = <fun>` qui teste si un polygone est convexe et orienté dans le sens direct.



convexe non convexe non convexe

FIG. 1 – *Test de convexité d'un polygone*

1.4 Appartenance à un polygone convexe

Écrire une fonction `estDans` de type `polygone -> point -> bool = <fun>` qui teste si un point P est à l'intérieur d'un polygone *poly* supposé convexe.

1.5 Découpage d'un polygone convexe

Écrire une fonction `decoupe` de type `polygone -> point -> point -> polygone = <fun>` telle `decoupe poly debut fin` raccourcit le polygone `poly` pour qu'il aille du sommet `debut` au sommet `fin`.

2 Recherche d'un pentagone vide

On admettra sans démonstration le résultat suivant, dû à Harboth

Étant donné un ensemble S d'au moins 10 points du plan, alors il existe un pentagone convexe ayant pour sommets 5 points de S et dont l'intérieur ne contient aucun point de S.

Le but de cette partie est de construire, à partir d'un hexagone convexe ayant pour sommets 6 points de S (dont l'intérieur peut éventuellement contenir des points de S), un pentagone convexe dont l'intérieur ne contient aucun point de S.

2.1 Sélection des points dans un hexagone

Écrire une fonction `selection` de type `polygone -> liste_de_points -> liste_de_points = <fun>` qui étant donné un polygone p et une liste de points s retourne la liste des points de s situés à l'intérieur du polygone p.

Dans la suite du problème, l'ensemble des points de S situés à l'intérieur d'un polygone sera simplement appelé *l'intérieur du polygone*.

2.2 Un seul point dans un hexagone

Étant donné un hexagone convexe h dont l'intérieur contient un seul point A, il s'agit de construire, en testant de quel coté se trouve A par rapport à une diagonale de h, un pentagone convexe ayant A et 4 sommets de h comme sommets.

Écrire une fonction `pentagone1` de type `polygone -> point -> polygone = <fun>` qui prend en paramètre un hexagone h et un point A et retourne un tel pentagone convexe.

2.3 Deux points dans un hexagone

Étant donné un hexagone convexe h contenant exactement deux points A et B, on veut maintenant construire, en testant de quel coté se trouve les sommets de h par rapport à la droite (AB), un pentagone d'intérieur vide ayant A, B et 3 sommets de h comme sommets. Écrire une fonction `pentagone2` de type `polygone -> point -> point -> polygone = <fun>` qui prend en paramètre un hexagone `h` et deux points A et B et retourne un tel pentagone convexe.

2.4 Plus de deux points dans un hexagone

On se donne à présent un hexagone convexe h contenant un ensemble K (ayant au moins deux points), et deux points A et B formant une arête de l'enveloppe convexe de K (ce qui signifie que tous les points de K, sauf A et B, sont à gauche de la droite orienté (AB)). On se propose de construire, en testant de quel coté se trouvent les sommets de h par rapport à la droite (AB), soit un pentagone d'intérieur vide ayant A, B et trois sommets de h comme sommets, soit un hexagone convexe ayant A, B et quatre sommets de h comme sommets et dont l'intérieur contient strictement moins de points que l'intérieur de h.

Écrire une fonction `pentagonePlus` de type

```
polygone -> point -> point -> liste_de_points -> polygone = <fun>
```

qui retourne soit un pentagone convexe d'intérieur vide, soit un hexagone convexe d'intérieur plus petit que h. Les arguments sont un polygone h, deux points A et B et une liste de points k.

Écrire une fonction `minimum` de type `liste_de_points -> point = <fun>` qui étant donné une liste de points k retourne le point d'abscisse minimale de k.

Écrire une fonction *pointSuivant* de type `point -> liste_de_points ->  liste_de_points = <fun>` étant donné point A sur l'enveloppe convexe des points de la liste k retourne le point suivant sur l'enveloppe convexe de la liste k.

On observera que si B est le point qui suit a sur l'enveloppe convexe, alors pour tout $c \notin \{A, B\}$, le triangle ABC tourne à gauche.

2.5 Assemblage

Tous les ingrédients étant réunis, écrire une fonction `pentagoneVide` de type

```
polygone -> liste_de_points -> polygone = <fun>
```

qui étant donné un hexagone convexe h et une liste de points k contenant les sommets de h retourne un pentagone convexe d'intérieur vide.