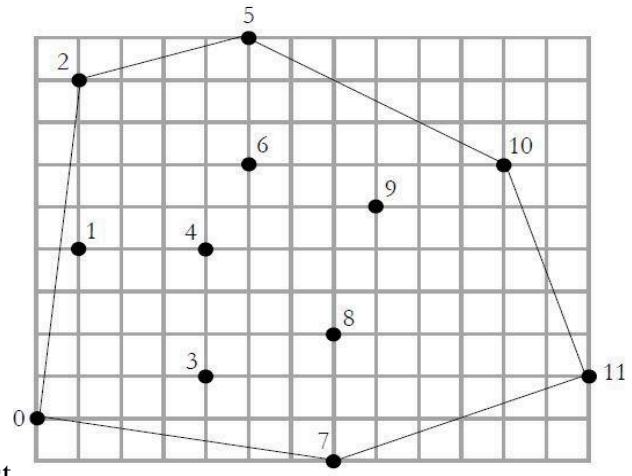
INSTITUT PRÉPARATOIRE AUX ÉTUDES D'INGÉNIEURS DE TUNIS
ANNÉE UNIVERSITAIRE 2017-2018

DS N°1– INFORMATIQUE

Durée : 1h.30
Documents : non autorisés

Exercice 1 : Liste

L'objectif de cet exercice est de calculer des enveloppes convexes de nuages de points dans le plan affine. On rappelle qu'un ensemble $E \subset \mathbb{R}^2$ est convexe si et seulement si pour toute paire de points$p, q \subset E$, le segment de droite [p, q] est inclus dans$E$. L'enveloppe convexe d'un ensemble $P \subset \mathbb{R}^2$, notée Conv(P), est le plus petit convexe contenant P. Dans le cas où P est un ensemble fini (appelé nuage de points), le bord de Conv(P) est un polygône convexe dont les sommets appartiennent à P.



L'idée est d'implémenter un algorithme de calcul du bord de l'enveloppe convexe d'un nuage de points P dans le plan affine dit "paquet cadeau". Ceci consiste à envelopper le nuage de points P progressivement en faisant pivoter une droite tout autour. Le nuage de points P est de taille n ≥ 3 et ne contient pas 3 points distincts alignés. Les coordonnées sont stockées dans une liste de listes L, comme dans l'exemple ci-dessous qui contient les coordonnées du nuage de points$p_j$de la figure ci-dessus:

L= [[0,1,1,4,4,5,5,7,7,8,11,13],[0,4,8,1,4,9,6,-1,2,5,6,1]]

Le premier élément de la liste contient les abscisses, tandis que le deuxième contient les ordonnées. Les abscisses sont ordonnées selon l'ordre croissant.

1. Ecrire une fonction **plusBas(L)** qui renvoie la position j du point le plus bas (c'est-à-dire de plus petite ordonnée) parmi les points du nuage P. En cas d'égalité, la fonction renvoie l'indice du point de plus petite abscisse parmi les points les plus bas.

Dans la suite, nous effectuons un test, celui de l'orientation. Etant donnés trois points p_i, $p_j$et $p_k$du nuage P, l'orientation est déterminée en calculant le déterminant à partir des

coordonnées des vecteurs $\vec{p_i p_j}$ et $\vec{p_i p_k}$.

2. Ecrire une fonction **orient(L,i,j,k)** qui renvoie le résultat (−1, 0 ou 1) du test d'orientation selon le signe du déterminant respectivement négatif, nul ou positif. Le calcul du déterminant est donnée par :

$$\det(\vec{p_i p_j}, \vec{p_i p_k}) = \left(x_j - x_i\right) \times \left(y_k - y_i\right) - \left(y_j - y_i\right) \times \left(x_k - x_i\right)$$

Un algorithme a été proposé par R. Jarvis, consiste à envelopper peu à peu le nuage de points P dans une sorte de paquet cadeau, qui à la fin du processus est exactement le bord de Conv(P).

Au départ, l'algorithme commence par le point le plus bas. A chaque itération et relativement à un point p_i, l'algorithme effectue une recherche du point suivant (p_j) dans l'enveloppe convexe. Le point suivant p_j est déterminé en faisant parcourir l'ensemble P de k points $k \in 1 \cdots n - 1$ tel que $p_j \prec p_k \Leftrightarrow orient(L, i, j, k) < 0$. L'idée consiste donc à chercher le point p_j en parcourant l'ensemble $P \backslash \{p_i\}$ et en partant de p_0 si p_i est différent de p_0 et partant de p_1 sinon.

Exemple : La démarche de calcul du prochain point de p_{10} de l'exemple de la figure ci-dessus est la suivante :

i	j	k	orient(L,i,j,k)
10	0	1	orient(L,10,0,1) < 0
10	1	2	orient(L,10,1,2) < 0
10	2	3	orient(L,10,2,3) > 0
10	2	4	orient(L,10,2,4) > 0
10	2	5	orient(L,10,2,5) < 0
10	**5**	6	**orient(L,10,5,6) > 0**
10	**5**	7	**orient(L,10,5,7) > 0**
10	**5**	8	**orient(L,10,5,8) > 0**
10	**5**	9	**orient(L,10,5,9) > 0**
10	**5**	10	**orient(L,10,5,10) = 0**
10	**5**	11	**orient(L,10,5,11) > 0**

Le prochain point est p_5 dans l'enveloppe Conv(P).

3. Ecrire une fonction **prochainPoint(L,i)**, qui à partir de l'indice *i* du point inséré en dernier dans le paquet cadeau, renvoie l'indice du prochain point à insérer.

4. Ecrire une fonction **convJarvis(L)** qui renvoie une nouvelle liste L2 contenant les indices des sommets du bord de Conv(P) en commençant par le poin le plus bas.

Exemple : Sur le nuage de la figure ci-dessus, le résultat sera la liste L2 = [7, 11, 10, 5, 2, 0].

Exercice 2 : Pile

L'objectif de cet exercice est d'étudier les manipulations de cartes, d'apparences magiques, proposées par Gilbreath en 1958.

En utilisant les fonctions de base des piles suivantes :

– taille_pile(p), qui renvoie la taille de la pile p,
– creer_pile(), qui renvoie une pile vide,
– est_vide(p), qui renvoie True ou False suivant que p est vide ou non,
– empiler(x,p), qui insère x au sommet de p,
– depiler(p) : qui supprime l'élément au sommet de p et renvoie sa valeur.

N.B : random.randint(a, b) où a et b sont des entiers, renvoie des entiers aléatoires compris entre a et b inclus.

Ecrire en Python les fonctions suivantes :

1. **inverser(p)** : permet d'inverser les éléments d'une pile
2. **couper(p)** : permet de couper une pile en deux piles. La coupure est aléatoire. La seconde pile présente les éléments dans l'ordre inverse. On évitera de renvoyer une pile vide. Exemple : couper([1,2,3,4,5,6,7]) donne comme résultat : ([1,2,3],[7,6,5,4])

3. **melange(p1,p2)** : qui à partir de deux piles, renvoie une pile ***p*** formée du mélange suivant :
   – tant qu'aucune pile n'est vide, on dépile aléatoirement **p1**ou **p2** et on empile l'élement sur **p**;
   – si l'une des piles d'entrée est vide, on dépile les éléments restants de l'autre pile sur **p** ;
   – on inverse **p**.

Exemple : melange([1,2,3,4],[5,6,7]) donne comme résultat : [1,5,2,3,6,4,7]

4. **TestGil()** : qui permet de :
   - Créer une pile formée de *n* répétitions de **k** cartes de 1 à **k**.
   - Couper puis mélanger cette pile selon les fonctions précédentes.

Exemple : [1,2,3,1,2,3,1,2,3,1,2,3] donne comme résultat : [3,2,1,3,2,1,3,2,1,1,3,2]

Bon travail

Un professeur d'école décide d'instaurer un système de points à la participation des étudiants dans un examen contenant *p* questions. A chaque étudiant, est associée une pile qui mémorise les différentes réponses.

-Si la réponse est correcte, alors un feu vert est empilé sur sa pile,
-Si la réponse est approximative alors un feu orangé est empilé et,
-Si la réponse est incorrecte alors un feu rouge est empilé.

Chaque étudiant a **0** point initialement, puis les points sont comptés de la façon suivante:

- Si la réponse est correcte, le nombre de points est incrémenté de 10 points,
- Si la réponse est approximative, le nombre de points est incrémenté de 5 points,
- Si la réponse est incorrecte, le nombre de points est incrémenté de 1 point.

On suppose qu'on a `n` étudiants numérotés de 1 à `n`, les réponses sont codées par des lettres :`'V'`pour le vert, `'R'`pour le rouge et `'O'`pour orange et que les étudiants ainsi que leurs piles de réponses sont stockés dans un dictionnaire `d` tels que les clés sont les numéros des étudiants et les valeurs sont les piles de réponses.

Exemple : d={1:['V','V','O','R',....], 2:['R','V','O','R',...], …, n:['O','V','O','R',...]}

Travail demandé : En utilisant les fonctions suivantes :

– creer_pile(), qui renvoie une pile vide,
– est_vide(p), qui renvoie True ou False suivant que p est vide ou non,
– empiler(x,p), qui insère x au sommet de p,
– depiler(p) : qui supprime l'élément au sommet de p et renvoie sa valeur.

Ecrire en Python les fonctions suivantes :

1. **Nombre** : qui saisit et retourne le nombre des étudiants enseignés par le professeur.
2. **SaisiRep**: qui, pour chaque étudiant, saisit sa pile de réponses et retourne le dictionnaire **d**.
3. **AfficheRep**: qui, à partir du dictionnaire **d**, affiche pour chaque étudiant ses réponses ('correcte', 'approximative' ou 'incorrecte').
4. **Resultat**: qui, à partir d'une pile de réponse **p**, calcule et retourne la somme de points cumulés.
5. **Resultats** : qui, à partir du dictionnaire **d**, retourne le dictionnaire **d1** contenant pour chaque étudiant la somme de points cumulés suivant sa pile de réponses.

Exemple **:** d1={1:30,2:75,...,n:5}

Bon travail