Lycée La Prat's
Classe de PT

Mercredi 14 janvier 2015

Épreuve d'Informatique 2

Durée 2 h

L'usage des calculatrices est interdit.

La présentation, la lisibilité, l'orthographe, la qualité de la rédaction et la précision des raisonnements entreront pour une part importante dans l'appréciation des copies. Les algorithmes doivent être commentés.

Exercice 1 (Cours)

Écrire une fonction qui calcule une intégrale à l'aide de la méthode des trapèzes : la fonction `methode_trap` prend en arguments

- une fonction `f`, qui prend en argument un float (nombre à virgule flottante) et retourne un float.
- deux float `a` et `b` qui représentent les bornes de l'intervalle d'intégration.
- un entier `n` représentant le nombre de points (à 1 près) dans l'intervalle $[a, b]$.

La fonction retourne un float approximant $\int_a^b f(t)\,\mathrm{d}t$ à l'aide de la méthode des trapèzes.
On pourra faire un dessin (ce n'est pas obligatoire).

Exercice 2 (théorème du sandwich)

Le problème dit du sandwich s'énonce de la manière suivante : d ensembles $E_1, \ldots, E_d$ finis de points en dimension d peuvent toujours être séparé chacun en deux parties de cardinal au plus $\lfloor \mathrm{Card}\,(E_i)/2 \rfloor$ par un hyperplan de dimension $d-1$ (certains points peuvent être dans l'hyperplan). $\lfloor x \rfloor$ désigne la partie entière de x.
De manière concrète, 3 ensembles de points dans l'espace peuvent être séparés en deux parties quasi-égales chacun par un plan. De façon imagée, le théorème dit que l'on peut couper en quantités égales, d'un seul coup de couteau, le jambon, le fromage et le pain d'un sandwich.
De même un ensemble fini de points dans le plan peut être séparé en 4 à l'aide de deux droites.
Ce sujet porte sur la résolution algorithmique de ce problème et de problèmes connexes selon différentes méthodes.

Vous êtes vivement encouragé à réutiliser les fonctions définies dans les questions précédentes du problème. Même si vous n'avez pas su programmer la dite fonction.

Partie 1 (Grand et petit)

Dans cette partie nous supposerons donné une liste L contenant n nombres réels (représentés par des float). Nous utiliserons la liste de longueur 11 suivante comme exemple :

3	2	5	8	1	34	21	6	9	14	8

1) Écrire une fonction `caculeIndiceMax`, d'argument L, qui renvoie l'indice du plus grand élément de la liste L (s'il y a plusieurs exemplaires du plus grand élément, elle renvoie le plus petit indice).
Dans l'exemple, la fonction renverra 6, car la case 6 contient la valeur 34.

2) Écrire une fonction `nombrePlusPetit`, d'arguments L et val, qui renvoie le nombre d'éléments de la liste L dont la valeur est inférieure ou égale à val.
Sur l'exemple, avec $val = 5$, elle renverra 4.

Partie 2 (Un tri pour accélérer)

Nous allons maintenant calculer la médiane d'une liste.
Rappelons qu'une valeur médiane m d'un ensemble E de nombres est un élément de E tel que les deux ensembles $E_{<m}$ (les nombres de E strictement plus petits que m) et $E_{>m}$ (les nombres de E strictement plus grands que m) vérifient Card $E_{<m} \leqslant \lfloor n/2 \rfloor$ et Card $E_{>m} \leqslant \lfloor n/2 \rfloor$. Notez que le problème de la médiane est une reformulation du théorème du sandwich pris en dimension 1.

1) Écrire une fonction `mediane_naive` d'argument L et calculant une médiane de L, en parcourant tous les éléments m de L et en calculant pour chacun d'eux les valeurs de Card $E_{<m}$ et Card $E_{>m}$.

Estimer sa complexité dans le pire cas (ne pas hésiter à numéroter les lignes de vos programme, dans la marge, pour pouvoir y faire référence clairement).

Une méthode plus efficace serait de trier la liste par ordre croissant tout en prenant la cellule du milieu dans la liste triée. Cette méthode certes rapide requiert $O(n \ln n)$ opérations. Il existe une méthode optimale en temps linéaire $O(n)$ pour trouver la médiane d'un ensemble de n éléments. La suite de cette partie a pour but d'en proposer une implémentation.

2) **a)** Écrire une fonction `partition`, d'arguments `L` (liste) et `indicePivot` (un indice appartenant à $[\![0, n-1]\!]$, où L est de longueur n). Si $p = L[indicePivot]$, la fonction devra retourner un triplet constitué :

- Une liste en mettant en premier les éléments strictement plus petits que p, puis les éléments égaux à p, puis les éléments strictement plus grands que p ;
- L'indice du premier élément égal à p dans la nouvelle liste ;
- L'indice du dernier élément égal à p dans la nouvelle liste.

Exemple : triplet retournée par l'appel `partition(L,3)`

```
([3, 2, 5, 1, 6, 8, 8, 34, 21, 9, 14], 5, 6)
```

b) Notons `M, i_pivot1, i_pivot2 = partition(L, indicePivot)`.
Que représentent `i_pivot1` et `i_pivot2` ? (illustrer sur l'exemple précédent).

c) Comment récupérer, avec du découpage en tranche de M, la liste des éléments

- strictement plus petits que le pivot ?
- strictement plus grands que le pivot ?

3) Écrire une fonction `elementK` d'arguments L et k (entier) qui cherche le $k+1$-ième élément (i.e. l'élément d'indice k) dans la liste L. On essayera autant que possible de ne pas trier entièrement la liste *(50% des points si vous triez entièrement la liste. N'y passez pas trop de temps non plus)*.

4) Supposons que dans l'algorithme précédent nous voulions rechercher le premier élément mais qu'à chaque étape le pivot choisi est le plus grand élément, quel est un ordre de grandeur du nombre d'opérations réalisées par votre fonction ?

5) L'algorithme précédent ne semble donc pas améliorer le calcul de la médiane. Le problème vient du fait que le pivot choisi peut être mauvais c'est-à-dire qu'à chaque étape un seul élément de la liste a été éliminé. En fait, si l'on peut choisir un pivot p dans L tel qu'il y ait au moins $(b-a)/5$ éléments plus petits et $(b-a)/5$ plus grands alors on peut montrer que l'algorithme précédent fonctionne optimalement en temps $O(n)$.

Pour choisir un tel élément dans L, on réalise l'algorithme `choixPivot` suivant (illustré sur l'exemple) :

- On découpe la liste en paquets de 5 éléments plus éventuellement un paquet plus petit. On calcule la médiane de chaque paquet.

(3)	2	5	8	1	34	21	6	9	(14)	(8)

- S'il n'y a qu'un seul paquet, on renvoie sa médiane.
- Sinon, on rappelle la fonction `choixPivot` sur les éléments médians précédents. Ici sur la liste

3	14	8

Écrire la fonction `choixPivot` de paramètre L et qui renvoie la valeur du pivot à l'aide de l'algorithme ci-dessus.

Quelle est la nature de cette fonction ? (est-elle itérative ?)

Partie 3 (En dimension 2, avec des points)

Définition de $\lceil x \rceil$: Si $x \in \mathbb{R}_+$, $\lceil x \rceil$ est le plus petit entier supérieur ou égal à x. Il vérifie $\lceil x \rceil \in \mathbb{N}$ et $\lceil x \rceil - 1 < x \leqslant \lceil x \rceil$.

Dans la partie précédente, nous avons étudié le problème de la médiane en dimension 1. On supposera donc que vous disposez maintenant d'une fonction `indiceMedian` d'argument L qui calcule la médiane de la liste L et renvoie l'indice de cet élément dans L. Vous supposerez de plus que cette fonction ne modifie pas l'ordre des éléments de la liste de départ L.

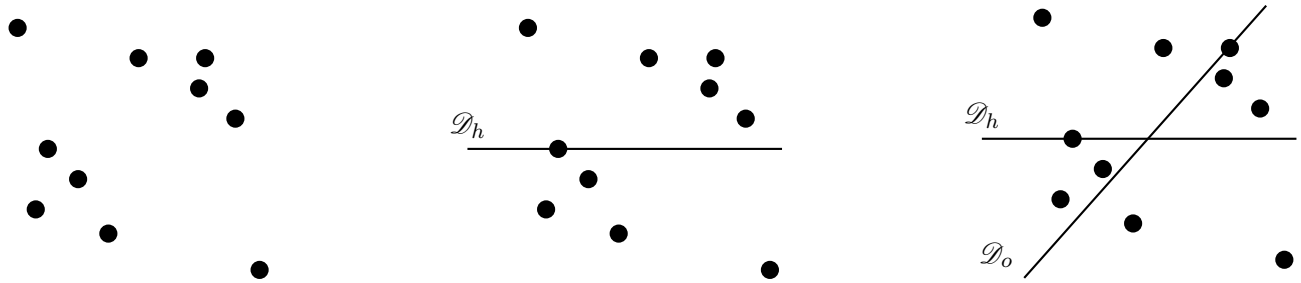
Dans cette partie, nous généralisons l'algorithme de manière à trouver deux droites dans le plan séparant un ensemble de n points en 4 parties de cardinal plus petit que $\lceil n/4 \rceil$.

Soit E un ensemble de n points tel qu'il n'existe pas trois points alignés.

Nous allons chercher deux droites $\mathscr{D}_h$ et $\mathscr{D}_o$ séparant cet ensemble de points en 4 parties comme le montre la troisième figure ci-dessous.

En effet, dans cette figure chaque partie est composée d'exactement 2 points, les points situés sur les droites n'étant pas pris en compte.

Nous supposerons donnés dans cette partie deux liste `Lx` et `Ly` de taille n contenant les coordonnées des n points. Ainsi le point i a comme abscisse `Lx[i]` et comme ordonnée `Ly[i]`.



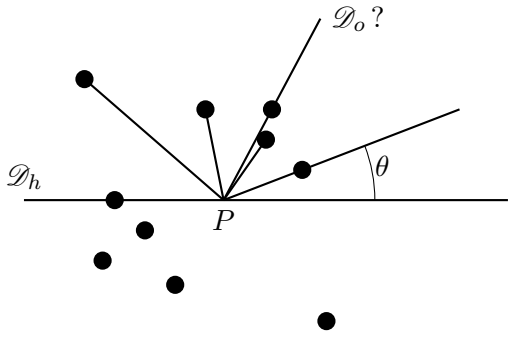
1) La première étape est de séparer les points en deux ensembles de même cardinal.

On effectuera cette séparation par une droite horizontale $\mathscr{D}_h$, qui passe par un point d'ordonnée médiane.

Écrire une fonction `coupeY(Lx, Ly)` qui renvoie l'ordonnée d'une droite horizontale séparant les points en deux parties de cardinal au plus $\lfloor n/2 \rfloor$.

2) La seconde droite $\mathscr{D}_o$, oblique, est plus difficile à trouver. Nous allons partir du point P d'intersection entre $\mathscr{D}_h$ et $\mathscr{D}_o$.

Soit P un point sur la droite horizontale $\mathscr{D}_h$ précédente de coordonnées (x, y). On veut vérifier si ce point peut être le point d'intersection des deux droites $\mathscr{D}_h$ et $\mathscr{D}_o$. Nous allons trouver dans un premier temps l'angle entre $\mathscr{D}_h$ et $\mathscr{D}_o$ en utilisant le fait que $\mathscr{D}_o$ doit séparer en deux parties de cardinal proche les points au dessus de $\mathscr{D}_h$. Ensuite nous allons vérifier si la droite $\mathscr{D}_o$ ainsi devinée sépare les points en dessous de $\mathscr{D}_h$ en deux parties presque égales.



Concrètement on considère les demi-droites partant de $P = (x, y)$ et joignant les k points de E au dessus strictement de $\mathscr{D}_h$ comme indiqué sur le schéma ci-dessus. On calcule ensuite les angles θ entre

$\mathscr{D}_h$ et ces demi-droites. Remarquez alors que toute demi-droite d'angle médian partage en deux parties de cardinal $\leqslant \lfloor k/2 \rfloor$ les points au dessus de $\mathscr{D}_h$.

Nous supposerons donnée une fonction `angle(P,M)` qui calcule et renvoie l'angle formé par une demi-droite horizontale partant de P et allant vers la droite et le segment $[PM]$, où P et M sont des couples de coordonnées. La valeur retournée sera comprise dans l'intervalle $[0, 2\pi[$ (on suppose que NumPy est chargé sous l'alias np).

Écrire une fonction `demiDroiteMedianeSup(Lx, Ly, P)` qui calcule et renvoie un angle médian entre $\mathscr{D}_h$ et les segments reliant $P = (x, y)$ avec les points de E dont l'ordonnée est supérieure à y.

3) Pour un point P donné, nous avons déterminé l'angle que doit prendre $\mathscr{D}_o$ pour couper les points au dessus de $\mathscr{D}_h$ en 2 parties de taille au plus moitié. Il faut maintenant vérifier que cette droite $\mathscr{D}_o$ coupe aussi les points en dessous de $\mathscr{D}_h$ en 2 parties quasi-égales. Il suffit de vérifier que l'angle de $\mathscr{D}_o$ sépare les angles formés entre P et les ℓ points en dessous de $\mathscr{D}_h$ en deux parties de cardinal $\leqslant \lceil \ell/2 \rceil$.

Écrire une fonction `verifieAngleSecondeDroite(Lx, Ly, P, theta)` qui calcule les angles formés entre $\mathscr{D}_h$ et les points strictement au dessous de $\mathscr{D}_h$. Votre fonction devra renvoyer :

- $\rightarrow$ 0 si theta est une médiane des ℓ angles.
- $\rightarrow$ -1 si le nombre d'angles strictement plus petits que theta est $> \lceil \ell/2 \rceil$
- $\rightarrow$ 1 si le nombre d'angles strictement plus grands que theta est $> \lceil \ell/2 \rceil$.

4) Si $xmin$ est l'abscisse minimale des points de E et $xmax$ l'abscisse maximale, alors l'intersection entre $\mathscr{D}_h$ et $\mathscr{D}_o$ a une abscisse entre $xmin$ et $xmax$. Nous allons donc rechercher cette abscisse en utilisant l'algorithme suivant :

1. Trouver $\mathscr{D}_h$ d'ordonnée y.
2. Soit $\alpha = xmin$ et $\beta = xmax$.
3. On calcule P le point au milieu de (α, y) et (β, y).
4. On calcule l'angle possible de la droite $\mathscr{D}_o$ par la fonction `demiDroiteMedianeSup`.
5. Soit d la valeur donnée par la fonction `verifieAngleSecondeDroite` avec l'angle trouvé précédemment. Si $d = 0$ alors on a trouvé $\mathscr{D}_o$. Si $d = -1$ on recommence à partir de l'étape 3 en prenant l'abscisse de P à la place de β. Si $d = 1$ on recommence mais en prenant l'abscisse de P à la place de α.

Écrire la fonction `secondeMediane(Lx, Ly, y)` qui à partir d'un ensemble E de points donnés par leurs coordonnées et de l'ordonnée de la droite $\mathscr{D}_h$ calcule et renvoie un couple contenant dans la première case l'abscisse x du point d'intersection de $\mathscr{D}_h$ et de $\mathscr{D}_o$ ainsi que l'angle de $\mathscr{D}_o$ dans la seconde case.

FIN DE L'ÉPREUVE