Partiel

Les documents, calculatrices et téléphones portables ne sont pas autorisés.
Vous pouvez écrire vos algorithmes en Python ou en pseudo-code. Dans le second cas, vous devrez préciser si certains de vos choix s'écartent des standards de Python (indice de début de liste à 1 au lieu de 0, par exemple). Quel que soit le langage choisi, il vous est demandé de faire bien attention à l'indentation. Enfin, toute réponse devra être justifiée.

****

Exercice 1 : Nombre d'occurences d'un élément dans une liste de listes

Le but de cet exercice est d'écrire une procédure comptant de nombre d'occurences d'un élément dans une liste de listes. Pour chacune des fonctions à écrire, on pourra utiliser les fonctions écrites dans les questions précédentes (même si la question n'a pas été traitée).

(a) Écrire une fonction occ(L,x) prenant en entrée une liste L et retournant le nombre d'occurences de x dans L (par exemple, occ($[1,3,3,5,3,8,3]$, 3) doit renvoyer 4).

(b) Écrire une fonction somme(L) prenant en entrée une liste d'entiers L et retournant la somme des éléments de L.

(c) Écrire une fonction listeocc (LL,x) prenant en entrée une liste de listes d'entiers $LL = [L_0, L_1, \ldots, L_k]$ et un entier x et retournant la liste $[n_0, n_1, \ldots, n_k]$ où n_i est le nombre d'occurences de x dans L_i.

(d) Écrire une fonction totalocc(LL,x) prenant en entrée une liste de listes d'entiers et retournant le nombre total d'occurences de x dans les sous-listes de LL.

Exercice 2 : Fonction de Mc Carthy

La fonction de McCarthy est la fonction $f : \mathbb{N} \to \mathbb{N}$ définie par la récurrence suivante :

$$f(n) = \begin{cases} n - 10 \text{ si } n > 100 \\ f(f(n+11)) \text{ sinon} \end{cases}$$

Écrire une fonction mc(n) prenant en entrée un entier naturel n et retournant $f(n)$.

Exercice 3 : Recherche des plus grands éléments d'une liste

Le but de cet exercice est de calculer la liste ordonnée des k plus grands éléments d'une liste d'entiers L (dont on notera n la longueur). On souhaite obtenir une méthode qui fait peu de comparaisons entre deux éléments de la liste lorsque k est petit.

(a) Si on commence par trier la liste L par une méthode de tri avancée puis qu'on extrait les k plus grands éléments de la liste triée, combien de comparaisons fait-on ?

(b) Écrire une fonction récursive recmaxi(L) retournant l'indice d'un élément maximum de la liste L (non vide) basée sur le principe suivant : le plus grand élément d'une liste L de n entiers est soit le premier élément de L, soit le plus grand élément de la sous-liste $L[1:]$ de L.

(c) En utilisant la fonction précédente, écrire une fonction maxi(L,k) prenant en entrées une liste d'entiers L et un entier $k > 0$ et retournant la liste ordonnée par ordre croissant des k plus grands éléments de L.

(d) Quel est le nombre de comparaisons entre deux éléments de la liste qu'effectue votre algorithme maxi(L,k) ?

(e) Expliquer comment faire une fonction maxi(L,k) effectuant $O(n \log k)$ comparaisons entre deux éléments de L (il est demandé d'expliquer l'idée mais pas d'écrire la procédure).

Exercice 4 : Tri bizarre

L'algorithme suivant prétend trier les éléments d'une liste de longueur paire.

```
def TriBizarre(L):
    n=len(L)
    for k in range(n/2-1):
        for i in range (n/2-1):
            if L[2*i]>L[2*i+2]:
                L[2*i],L[2*i+2]=L[2*i+2],L[2*i]
            if L[2*i+1]>L[2*i+3]:
                L[2*i+1],L[2*i+3]=L[2*i+3],L[2*i+1]
```

TriBizarre.py

Si vous pensez que l'algorithme ci-dessus trie correctement, expliquez pourquoi. Sinon, donnez un exemple de liste de longueur paire qui n'est pas correctement triée par cet algorithme puis expliquez ce que fait celui-ci.

Exercice 5 : Calcul de déterminant

On rappelle que le déterminant d'une matrice $M = (m_{i,j}) \in \mathcal{M}_n(\mathbb{R})$ peut se calculer en développant le long d'une colonne : pour $1 \leq k \leq n$ fixé,

$$\det(M) = \sum_{i=1}^{n} (-1)^{i+k} m_{i,k} \det(\widetilde{M}_{i,k})$$

où $\widetilde{M}_{i,j}$ est la matrice obtenue de M en supprimant la i-ème ligne et la j-ième colonne.

(a) Écrire une fonction récursive det(M,k) qui calcule le déterminant d'une matrice M (représentée comme la liste de ses lignes) en développant le long de la k-ième colonne.

(b) Quelle est la complexité de l'algorithme ?

(c) Développer le long de la colonne k est intéressant si elle contient beaucoup de 0 : pourquoi ? Améliorer la fonction de la question (a) en tenant compte de cette observation.