TP Informatique commune : Révisions avant les concours

Les exercices proposés sont extraits de sujets de concours et constituent des exercices classiques à savoir refaire. Vous pouvez aussi faire les exercices proposés dans la feuille "Conseils pour les concours".

On importera les modules suivants avant de démarrer:

```
import numpy as np
import random
```

Fonctions sur les listes

1. Ecrire des fonctions de recherche du minimum/maximum d'une liste sans utiliser les fonctions `min` et `max`.
2. Écrire une fonction `smul(n,L)` à deux paramètres, un nombre et une liste de nombres, qui multiple chaque élément de la liste par le nombre et renvoie une nouvelle liste.
3. Écrire une fonction `vsom(L1,L2)` qui prend en paramètre deux listes de nombres de même longueur et qui renvoie une nouvelle liste constituée de la somme terme à terme de ces deux listes.
4. Écrire une fonction `vdif(L1,L2)` qui prend en paramètre deux listes de nombres de même longueur et qui renvoie une nouvelle liste constituée de la différence terme à terme de ces deux listes (la première moins la deuxième).
5. Écrire une fonction calculant la moyenne des éléments d'une liste.
6. Écrire une fonction `egal(L1,L2)` prenant en entrée deux listes de même longueur et qui renvoie un booléen indiquant si elles sont égales, sans utiliser le `==`. L'adapter pour qu'elle renvoie l'indice de la 1ère différence rencontrée dans le cas où les listes ne sont pas égales.
7. Écrire une fonction qui supprime les doublons dans une liste (quelconque) et donner sa complexité. En écrire une autre qui supprime les doublons dans une liste triée (on ne triera pas la liste dans la fonction) avec une meilleure complexité que précédemment.
8. Écrire une fonction `reduire(L)` qui prend en entrée une liste de nombres et la réduit de moitié en supprimant les éléments les plus grands.
9. Écrire une fonction `muter(L)` qui prend en entrée une liste et inverse aléatoirement deux de ses éléments (distincts).
10. Écrire une fonction `croiser(L1,L2)` qui prend en entrée deux listes L_1 et L_2 de même taille et qui renvoie la liste croisée constituée de la première moitié de la liste L_1 et de la deuxième moitié de la liste L_2.
11. Écrire une fonction `decomp_binaire(n)` prenant en entrée un entier n et qui renvoie la liste correspondant à sa décomposition binaire, par exemple `decomp_binaire(4)` renverra la liste `[1,0,0]`.
12. Écrire la fonction inverse `versEntier(L)` qui prend en entrée une liste de 0 et de 1 et renvoyant l'entier en base 10 correspondant.

13. Ecrire une fonction de recherche dichotomique d'un élément dans une liste. La fonction renverra un booléen indiquant si l'élément est présent ou non dans la liste (variante : renvoyer l'unique indice i tel que $L[i] \leqslant x < L[i+1]$). Donner sa complexité.

14. **Tris:** Écrire les fonctions de tris par insertion, tri fusion et tri rapide d'une liste et rappeler leurs complexités. Les tris par insertion et rapide devront être en place, c'est-à-dire modifier directement la liste donnée en entrée sans créer une nouvelle liste.

Fonctions sur les tableaux numpy

1. Écrire une fonction `nombre_zeros(t)` prenant en entrée un tableau de taille (n, n) et renvoyant le nombre d'éléments nuls dans ce tableau. Donner sa complexité.

2. Écrire une fonction `calculer_distances(t)` prenant en entrée un tableau de taille $(n, 2)$ représentant des coordonnées de points dans le plan et renvoyant un tableau de taille (n, n) dans lequel se trouvent les distances entre ces points.

3. Écrire une fonction `selectionner(t,imin,imax)` prenant en entrée un tableau de taille (n, p) (pas forcément carré) et deux indices $imin$ et $imax$ et renvoyant une liste contnenant les coordonnées des points du tableau dont les valeurs sont comprises (au sens large) entre $imin$ et $imax$.

Les fichiers

Des données se trouvent dans le répertoire de travail sous forme d'un fichier `donnees.txt`. On a représenté une donnée par ligne. Proposer une suite d'instructions permettant de créer à partir de ce fichier une liste de flottants `liste_donnees` contenant les différentes valeurs des données. On prendra garde à ne pas insérer dans la liste la première ligne du fichier.

Analyse numérique

1. Écrire des fonctions `methode_rectangles(f,a,b,n)` et `methode_trapezes(f,a,b,n)` qui calculent de manière approchée l'intégrale de f entre a et b en découpant l'intervalle $[a, b]$ en n sous-intervalles de même taille et en utilisant respectivement la méthode des rectangles et des trapèzes.

2. Écrire une fonction `dicho(f,a,b,eps)` qui applique la méthode de dichotomie à la fonction f sur l'intervalle $[a, b]$ (on supposera $f(a)$ et $f(b)$ de signes opposés) en s'arrêtant quand on a trouvé une valeur x telle que $|f(x)| < eps$. Variante : on prend en entrée n à la place de eps et on s'arrête après n étapes. Donner une estimation de l'erreur commise après n étapes, ainsi que le nombre d'étapes nécessaire pour que l'erreur soit $< \varepsilon$.

3. Écrire une fonction `newton(f,fp,x_0,n)` qui applique la méthode de Newton à la fonction f et sa dérivée f_p en partant du point x_0 en n étapes. Donner une condition suffisante sur f pour que la méthode converge.

4. Écrire une fonction `euler(F,x_0,T)` qui applique la méthode d'Euler au problème de Cauchy:

$$\begin{cases} x'(t) = F(x(t), t) \\ x(t_0) = x_0 \end{cases}$$

   La fonction prendra en entrée une fonction F, la condition initiale x_0 et le vecteur (tableau numpy) T des temps $[t_0, \ldots, t_n]$.