TP 11 : complexité

PTSI 2024/2025

POINT COURS : LISTE ALÉATOIRES AVEC NUMPY

On rappelle qu'on peut générer une liste aléatoire d'entier à l'aide du module `numpy`. Par exemple :

```
import numpy as np
alea = np.random.randint(-10,100,12)
```

va générer une liste de 12 entiers aléatoirement prix entre -10 et 100 exclu.

Exercice 1 : *Méthode de* HORNER

Niveau : 1

Dans cet exercice on considère des fonctions polynômiales $f_n : x \mapsto \sum_{k=0}^{n} a_k x^k$ où $n \in \mathbb{N}$ et la famille $(a_k)_{k\in[\![0\ ;\ n]\!]}$ est une famille de nombre réels. On considérera que f_n est de degré n.

1. a. Écrire une fonction `evalpoly` qui prend en entrée une liste des coefficients $(a_k)_{k\in[\![0\ ;\ n]\!]}$ d'une fonction polynômiale et un flottant `x` et qui renvoie la valeur de $f_n(x)$.
   b. Estimer la complexité de cette fonction en fonction de n.
2. On propose maintenant la réécriture suivante d'une évaluation de f_n en $x \in \mathbb{R}$:

$$f_n(x) = a_0 + x f_{n-1}(x)$$

ou $f_{n-1} : x \mapsto \sum_{k=1}^{n} a_k x^{k-1}$

   a. Justifier la formule précédente.
   b. Écrire une fonction Python `horner` qui prend en entrée une liste des coefficients $(a_k)_{k\in[\![0\ ;\ n]\!]}$ et un flottant pour la valeur de x qui renvoie la valeur de $f_n(x)$ calculée récursivement à partir de la formule précédente.
   c. Pour $n \in \mathbb{N}$ le degré d'une fonction polynomiale f_n, on note $C(n)$ la complexité temporelle du calcul de $f_n(x)$ par la fonction précédente.
      i. Quelle relation pouvez-vous écrire entre $C(n+1)$ et $C(n)$?
      ii. Estimer la complexité de votre fonction `horner` et la comparer avec celle de la première fonction.

Exercice 2 : *Tri fusion*

Niveau : 4

On va programmer dans cet exercice un algorithme de tri dont la complexité est beaucoup plus faible que celle d'un tri type insertion ou sélection.

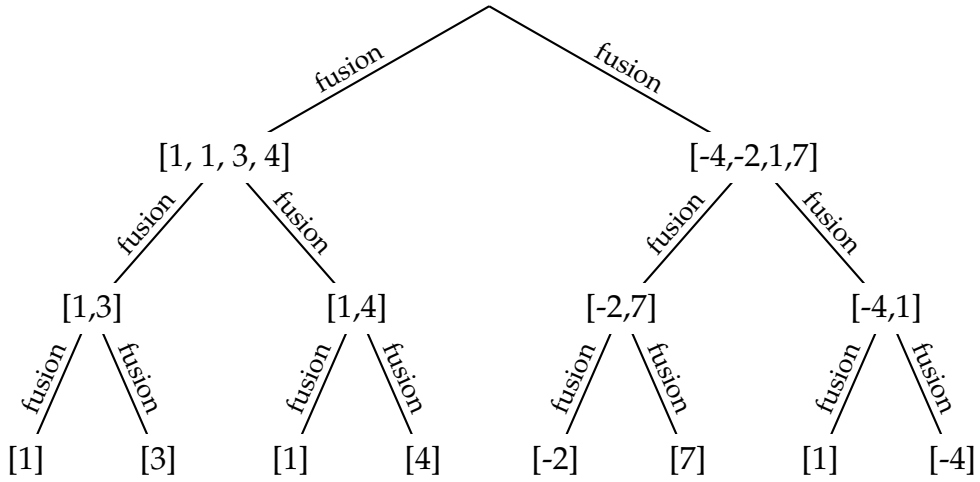
1. Programmer une fonction python que vous nommerez `fusion` et qui prendra en entrée deux listes de nombres triées par ordre croissant et retournera une liste triée par ordre croissant et formée des éléments des deux listes

Pour simplifier, on explique L'algorithme pour une liste de longueur 2^n où n est un entier naturel :

- Au rang 0, on considère la que la liste est formé de 2^n sous-listes de longueur 1. On les fusionne 2 à 2 pour obtenir des listes triées de longueur 2 qu'on réunit dans une nouvelle liste.
- Au rang 1, on considère que la liste est formé de 2^{n-1} sous-listes de longueur 2 qui sont triées . On les fusionne 2 à 2 pour obtenir des listes triées de longueur 4 qu'on réunit dans une nouvelle liste.
- Au rang 2 ...
  On donne ci-dessous une représentation sous forme d'arbre de l'algorithme pour trier la liste :

  `L = [1, 3, 1, 4, -2, 7, 1, -4]`

  L'algorithme consiste donc à ne pas trier directement la liste de départ, mais à la subdiviser en sous-liste jusqu'à obtenir des sous-listes de longueur 1 ... faciles à trier, puis à fusionner les listes triées.



2. Faire tourner vous-même sur feuille cet algorithme pour la liste

   `L = [2, 3, 1, -2.7, -4, 2, 6, -3.8]`

3. Écrire une fonction `fusion` qui prend en entrée deux listes de nombres **supposées triées par ordre croissant** et qui renvoie une liste contenant tous les éléments des deux listes (répétés autant de fois qu'ils sont présents) mais dans l'ordre croissant.
4. Écrire une fonction récursive `tri_fusion` qui met en œuvre l'algorithme précédent pour une liste de longueur quelconque. Autrement dit une fonction qui prendra entrée une liste notée `liste` et renverra une version triée de cette liste.
5. Effectuer des tests de temps d'exécution comme dans le cas précédent mais en prenant des longueurs de liste `lg` prenant les valeurs 2^k pour $k \in [\![7\ ;\ 12]\!]$ et commenter.

Exercice 3 : *Taxicab*

Niveau : 2

1. Écrire une fonction Python qui étant donné un entier naturel N renvoie la liste des couples d'entiers (a, b) avec $a < b$ tels que $a^3 + b^3 = N$.
2. Évaluer la complexité temporelle de votre fonction en fonction de N.
3. À l'aide de votre fonction déterminer le nombre dit "taxicab(2)" qui est le premier nombre entier qui est la somme de deux cubes d'entiers d'au moins deux manières différentes.
4. Essayer de déterminer taxicab(3) avec la méthode précédente .....