

TD n° 6

1 Récursivité et opérations numériques classiques

Exercice 1. Produit récursif

Écrivez une fonction récursive **produit**(*a*, *b*) où *a* et *b* sont deux entiers positifs. Cette fonction retourne le produit de *a* et de *b* et se basera sur l'égalité suivante :

$$a \times b = a + a \times (b - 1)$$

Vous en donnerez une version non terminale et une version terminale.

Exercice 2. Puissance récursive

Écrivez une fonction récursive **puissance**(*x*, *n*) où *x* est un float et *n* un entier positif. Cette fonction retourne le résultat de x^n se basera sur l'égalité suivante :

$$x^n = x \times x^{n-1}$$

Vous en donnerez une version non terminale et une version terminale.

Exercice 3. Somme récursive

Écrivez une fonction récursive **somme**(*liste*) où *liste* est une liste de nombres. Cette fonction retourne la somme de tous les termes de la liste.

Vous en donnerez une version non terminale et une version terminale.

2 Récursivité et suites numériques récurrentes

Exercice 1. Suite arithmético-géométrique

Dans cet exercice on cherche à calculer, pour *a* et *b* nombres réels quelconques, et *n* entier positif, le terme u_n de la suite récurrente définie par :

$$u_n = \begin{cases} u_0, & \text{si } n = 0 \\ a \times u_{n-1} + b & \text{si } n \geq 1 \end{cases}$$

1. Ecrivez une fonction itérative **terme_de_rang_iter**(*n*, *u*₀, *a*, *b*) qui calcule, de façon **itératif** (en utilisant une boucle *for*), le terme de rang (d'indice) *n* de la suite.
2. Donnez une version récursive non terminale **terme_de_rang_rec1**(*n*, *u*₀, *a*, *b*);
3. Donnez enfin une version récursive terminale **terme_de_rang_rec2**(*n*, *u*₀, *a*, *b*).

3 Récursivité et listes

Exercice 1. Maximum récursif d'une liste, v1

Écrivez une fonction récursive *max_rec(liste)* qui retourne le maximum des valeurs de la liste (supposée non vide). Vous en donnerez une version **non terminale**.

Vous utiliserez la propriété suivante : le maximum d'une liste est la valeur maximum entre le premier élément de la liste et le maximum de la liste amputée de son premier élément.

Exercice 2. Maximum récursif d'une liste, v2

Écrivez une fonction récursive *max_dicho_rec(liste)* qui retourne le maximum des valeurs de la liste (supposée non vide). Vous en donnerez une version **non terminale**.

Vous utiliserez la propriété suivante : le maximum d'une liste est la valeur maximum entre le maximum de la liste contenant la première moitié des éléments, et le maximum de la liste contenant le reste des éléments.

Exercice 3. Comparatif

Quelle est, à votre avis, des deux fonctions précédentes, celle qui nécessite le moins d'appels récursifs ? Vous pourrez comparer ce qui se passe pour différentes valeurs de l'effectif initial n , par exemple $n = 8$ ou $n = 32$.

Exercice 4. liste triée ?

Écrivez une fonction récursive *est_croissante(liste)* qui retourne True si la liste (supposée numérique) est triée par ordre croissant. Vous en donnerez une version **non terminale**.

Vous pourrez utiliser la propriété suivante : une liste est croissante si elle contient au plus un élément, ou si son premier élément est inférieur ou égal au deuxième et si la liste privée de son premier élément est croissante.

Exercice 5. recherche linéaire dans une liste quelconque

Écrivez une fonction récursive *indice_v1(liste, x)* qui retourne le premier indice i tel que *liste[i]* soit égal à x . On suppose que la liste est numérique, mais pas forcément triée. Si x n'est pas présent dans la liste, la fonction retourne -1. Vous donnerez une version **non terminale**.

4 Récursivité et chaînes de caractères

Exercice 1. Palindrome

Un palindrome est une chaîne de caractères dont les lectures de gauche à droite et de droite à gauche donnent le même résultat.

- **kayak** ou **abba** sont des palindromes, ainsi que **aa** ou **a**, ou même **la chaîne vide** ;
- Par contre, **abca**, **baobab** ou **abcda** n'en sont pas.

Écrivez une fonction récursive *est_un_palindrome(mot)* qui retourne un booléen qui dit si, oui ou non, la chaîne de caractère *mot* est un palindrome.

Pour cela :

- déterminez les cas d'arrêt que vous utiliserez ;
- déterminez l'instruction récursive qui contient l'appel récursif.

Exercice 2. *Count* récursif

Dans cet exercice on cherche à écrire un équivalent récursif de la méthode *count* des chaînes de caractères. Pour résoudre cet exercice, vous n'avez pas le droit d'utiliser la méthode *count*.

Écrivez une fonction récursive *nb_occurrences(mot, lettre)* qui retourne le nombre de fois où apparaît la lettre dans le mot.

Vous prendrez le temps, comme toujours, de préciser le(s) cas d'arrêt et l'instruction récursive.

Peut-on proposer, comme pour le maximum, une version dichotomique ?

Exercice 3. Minuscules d'un mot

Écrivez une fonction *nb_minuscules(mot)* qui compte le nombre de minuscules dans le mot passé en paramètre. Mêmes consignes pour le(s) cas d'arrêt et l'instruction récursive. Peut-on proposer une version dichotomique ?

Exercice 4. Indice minimum

Écrivez une fonction *indice_minimum(mot, lettre)* qui retourne le plus petit indice *i* pour lequel *mot[i] = lettre*. Si *lettre* n'est pas dans *mot*, la fonction retournera l'indice qui suit le dernier caractère, soit *len(mot)*.

Par exemple :

- *indice_minimum("eabcdef", "e")* retourne 0 ;
- *indice_minimum("eabcdef", "a")* retourne 1 ;
- *indice_minimum("eabcdef", "f")* retourne 6 ;
- *indice_minimum("abcbd", "b")* retourne 1 ;
- *indice_minimum("eabcdf", "g")* retourne 6 ;
- *indice_minimum("f", "g")* retourne 1 ;
- *indice_minimum("", "f")* retourne 0 ;

Exercice 5. Indice maximum

Écrivez une fonction *indice_maximum(mot, lettre)* qui retourne le plus grand indice *i* pour lequel *mot[i] = lettre*. Si *lettre* n'est pas dans *mot*, la fonction retournera l'indice qui précède le premier caractère, soit -1.

Par exemple :

- *indice_maximum("eabcdef", "e")* retourne 5 ;
- *indice_maximum("eabcdef", "a")* retourne 1 ;
- *indice_maximum("eabcdef", "f")* retourne 6 ;
- *indice_maximum("eabcdf", "g")* retourne -1 ;
- *indice_maximum("f", "g")* retourne -1 ;
- *indice_maximum("", "f")* retourne -1 ;

5 Complément sur la recherche dans une liste

Exercice 1. recherche dichotomique dans une liste triée

Écrivez une fonction récursive *indice_v2(liste, x)* qui retourne un indice **i** tel que **liste[i]** soit égal à **x**. On suppose que la liste est numérique et triée par ordre croissant. Si **x** n'est pas présent dans la liste, la fonction retourne **-1**.

La recherche devra utiliser le principe de la dichotomie : à chaque appel récursif, la recherche devra être poursuivie dans une moitié de la zone de recherche.

Vous donnerez une version **non terminale**.