

## TD 2

## Exercices d'application

## Savoirs et compétences :

- Alg – C15 : Récursivité : avantages et inconvénients.

## Exercice 1 – Fonction mystère

D'après ressources de C. Lambert. On donne la fonction suivante :

```
■ Python
def mystere(L):
    """
    Ceci est la fonction mystère, saurez-vous trouver
    son but ?
    Entrée :
        * L(list) : liste de nombres entiers ou réels
    Sortie :
        * ???
    """
    n = len(L)
    if n==0 :
        return (None)
    elif n==1 :
        return (L[0])
    else :
        x = mystere(L[0:n-1])
        if x<=L[-1] :
            return (x)
        else :
            return (L[-1])
```

**Question 1** Sans coder la fonction, déterminer le résultat de l'instruction `print(mystere([14, 20, 3, 16]))` ? Vous pourrez représenter de façon graphique l'empilement et le dépilement de la pile d'exécution.

**Question 2** D'après vous quel est le but de cette fonction ?

**Question 3** Programmer la fonction et tester l'instruction précédente. Sur plusieurs exemples, vérifiez la conjecture faite à la question précédente.

**Question 4** Question subsidiaire. – Montrer que la propriété suivante est une propriété d'invariance :  
 $\mathcal{P}$  : l'algorithme retourne le plus petit élément de la liste de taille  $k$ , s'il existe.



- Il faudra montrer que l'algorithme se termine au moyen d'un variant de boucle.
- Il faudra montrer que  $\mathcal{P}$  est une propriété d'invariance.

## Exercice 2 – Palindrome...

D'après ressources de C. Lambert. On souhaite réaliser une fonction *miroir* dont le but est de retourner le «miroir» d'une chaîne de caractères. Par exemple le résultat de *miroir*("miroir") serait "riorim".

**Question 1** Programmer la fonction *miroir\_it* permettant de répondre au problème de manière itérative.

**Question 2** Programmer la fonction *miroir\_rec* permettant de répondre au problème de manière récursive.

**Question 3** Que renvoie la fonction si la chaîne de caractère est "Eh ! ça va la vache" ?

**Question 4** Évaluer la complexité algorithmique de chacune des deux fonctions.

### Exercice 3 – Calcul de $n!$

On rappelle la définition de  $n!$  :

$$\forall n \in \mathbb{N} : \begin{cases} n! = \prod_{k=1}^n k & \text{si } n \geq 1 \\ n! = 1 & \text{si } n = 0 \end{cases}$$

**R** Pour vérifier vos résultats, vous pouvez utiliser la fonction disponible dans la bibliothèque `math` :

```
>>> from math import factorial
>>> print(factorial(4))
24
```

**Question 1** Définir la fonction `fact_it` permettant de calculer  $n!$  de façon itérative.

**Question 2** Donner alors la complexité algorithmique de votre algorithme.

**Question 3** Définir la fonction `fact_rec` permettant de calculer  $n!$  de façon récursive.

**Question 4** Évaluer  $1010!$  dans les trois cas (itératif, récursif et fonction native de Python). Expliquer ce qu'il se passe ?

### Exercice 4 – Suite de Fibonacci

D'après ressources de C. Lambert.

On définit la suite de Fibonacci de la façon suivante :

$$\forall n \in \mathbb{N}, \begin{cases} u_0 = 0, u_1 = 1 \\ u_{n+2} = u_n + u_{n+1} \end{cases}$$

**Question 1** Définir la fonction `fibonacci_it` permettant de calculer  $u_n$  par une méthode itérative. Évaluer la complexité algorithmique de l'algorithme.

**Question 2** Définir la fonction `fibonacci_rec` permettant de calculer  $u_n$  par une méthode récursive « intuitive ». Évaluer la complexité algorithmique de l'algorithme.

**Question 3** Observer comment passer du couple  $(u_n, u_{n+1})$  au couple  $(u_{n+1}, u_{n+2})$ . En déduire une autre méthode récursive pour calculer le  $n^{\text{e}}$  terme de la suite de Fibonacci. Évaluer la complexité algorithmique de l'algorithme.

### Exercice 5 – Calcul de déterminant

D'après ressources de David Prévost – UPSTI

On souhaite calculer le déterminant d'une matrice (carrée)  $A$  par la formule de développement sur la première ligne (ou sur la première colonne) mais pas une autre (par exemple si on le faisait sur la troisième, la fonction ne conviendrait pas pour un déterminant  $2 \times 2$ ).

**Question 1** Écrire une fonction `extraire(A, lig, col)` qui permet de fabriquer la matrice extraite de  $A$  en rayant la ligne `lig` et la colonne `col` (dans l'optique d'obtenir son déterminant).

**Question 2** Écrire une fonction récursive `determinant(A)` retournant le déterminant de  $A$ .

**Question 3** Tester l'algorithme sur une matrice (de taille au moins  $3,3$ ) et comparer le résultat à ce que retourne la fonction `det` du module `linalg` de la bibliothèque `numpy`.

### Exercice 6 – Bubble bobble

D'après ressources de Jean-Pierre Becirspahic

<http://info-llg.fr/>.

On suppose disposer d'une fonction `circle([x, y], r)` qui trace à l'écran un cercle de centre  $(x; y)$  de rayon  $r$ .

**Question** Définir deux fonctions récursives permettant de tracer les dessins présentés figure suivante (chaque cercle est de rayon moitié moindre qu'à la génération précédente).

