

Nombre de boucles

Déterminer la complexité des algorithmes suivants :

```
1. x = 0
   for i in range(n) :
       for j in range(n) :
           x += 1
```

```
2. x = 0
   for i in range(n) :
       for j in range(i) :
           x += 1
```

```
3. i, x = n, 0
   while i > 1 :
       x += 1
       i //= 2
```

```
4. def f(n) :
    if n == 0 :
        return 0
    else :
        x = 0
        for i in range(n) :
            x += 1
        return x + f(n//2)
```

Code à complexité imposée

- Écrire une fonction, de complexité linéaire, `maxi1(L:list)->tuple` qui renvoie le plus grand élément d'une liste d'entiers ainsi que la liste de ses positions.
- Écrire une fonction, de complexité linéaire, `maxi2(L:list)->tuple` qui renvoie le deuxième plus grand élément de la liste ainsi que la liste de ses positions ; on supposera que la liste comporte au moins deux éléments différents.

Liste ou dictionnaire

On veut coder une fonction qui renvoie l'intersection de deux listes L1 et L2 (la liste des éléments présents dans L1 et L2).

- Évaluer la complexité des deux codes suivants en fonction de $n = \max(\text{len}(L1), \text{len}(L2))$.

```
def inter(L1,L2) :
    L = []
    for x in L1 :
        if x in L2 :
            L.append(x)
    return L
```

```
def Inter(L1,L2) :
    d = {x : True for x in L2}
    L = []
    for x in L1 :
        if x in d :
            L.append(x)
    return L
```

- Écrire une fonction de complexité linéaire `SansDoublon(L:list)->list` qui renvoie une copie de la liste L, dans laquelle chaque élément n'apparaît qu'une seule fois.

Améliorer la complexité

On souhaite déterminer la longueur de la plus grande composante croissante d'une liste L d'entiers, c'est-à-dire la longueur maximale d'une sous-liste de L croissante (au sens large). La liste $L=[1,1,2,1,3]$ possède deux composantes croissantes $[1,1,2]$ et $[1,3]$, la plus longue est donc de longueur 3. On considère les fonctions suivantes :

```
def CC(L:list ,x:int) :
    j = x
    while j < len(L)-1 and L[j]<=L[j+1] :
        j += 1
    return j+1
```

```
def PGCC(L:list) :
    i,M = 0,CC(L,0)
    for x in range(len(L)) :
        if CC(L,x) > M :
            i,M = x,CC(L,x)
    return M
```

- Quel est le rôle de la fonction CC ?
- Évaluer la complexité dans le pire des cas de CC en fonction de la longueur n de L.
- En déduire la complexité de la fonction PGCC, en fonction de la longueur n de L
- Écrire une fonction `PGCC2(L:list)->int` qui renvoie aussi la longueur de la plus grande composante croissante de L dont la complexité sera linéaire.