NOM :
Prénom :

Vendredi 30 Janvier 2026

Interrogation 6

Complexité

Correction

Exercice 1 :
Donner les syntaxes précises suivantes avec rédaction éventuelle :

1. Définition d'un variant de boucle.

Un variant de boucle est une variable de la boucle correspondant à une suite d'entier strictement décroissante avec le nombre de passage dans la boucle. La boucle s'arrêtant quand le variant est nul.

2. Définition d'un invariant de boucle.

Un invariant de boucle est une propriété qui est vraie après chaque passage dans la boucle. L'invariant de boucle permet de vérifier que la boucle est correcte.

3. Complexité d'une structure `if/else`.

Pour une structure :

```
if c :
    p
else :
    q
```

en notant $C(k)$ le nombre d'opérations de l'instruction `k`, le nombre d'opérations effectuées au maximum est $C(c)+\max(C(p),C(q))$.

4. Complexité d'une boucle `for`

Pour une boucle `for`, le nombre d'opérations effectuées est le nombre d'opérations sur corps de la boucle, multiplié par le nombre de passage dans la boucle. Donc pour une boucle

```
for k in range(a,b) :
    p
```

le nombre d'opérations effectuées est $(b-a+1)C(p)$, où $C(p)$ est le nombre d'opérations du bloc d'instructions `p`.

Exercice 2 :
On considère la fonction suivante :

```
def mystere(L:list) -> float :
    S=0
    for x in L :
        for y in L :
            S = S + x*y
    return(S)
```

1. Faire un dérouler de cette fonction pour `L=[1,3,7]`.
2. Déterminer la complexité de cet algorithme.
3. Proposer un autre code de la même fonction mais avec un meilleure complexité.
4. Étudier la correction de votre nouveau code.

1. On a le dérouler suivant :

Ligne	x	y	S	Retour
2			0	
3	1		0	
4	1	1	0	
5	1	1	1	
4	1	3	1	
5	1	3	4	
4	1	7	4	
5	1	7	11	
3	3	7	11	
4	3	1	11	
5	3	1	14	
4	3	3	14	
5	3	3	23	
4	3	7	23	
5	3	7	44	
4	7	1	44	
5	7	1	51	
4	7	3	51	
5	7	3	72	
4	7	7	72	
5	7	7	121	
6				121

2. Si n est la longueur de la liste `L`, la ligne 5 contenant 3 opérations, la boucle **`for`** de la ligne 4 contient alors $3n$ opérations, et donc la boucle **`for`** de la ligne 3 contient $3n^2$ opérations. Donc la fonction **`mystere`** effectue $4n^2+2$ opérations pour une liste de longueur n.

Donc **`mystere`** a une complexité linéaire $O(n^2)$.

3. Si L est une liste, l'algorithme calcule :

$$\sum_{x\in L}\sum_{y\in L} xy = \sum_{x\in L}\left(x\sum_{y\in L} y\right) = \left(\sum_{x\in L} x\right)^2.$$

Donc on peut proposer, comme simplification de code :

```
1 def mystereSimple(L:list) -> float :
2     S=0
3     for x in L :
4         S = S+x
5     return(S**2)
```

5. Soit $n\in\mathbb{N}$ la longueur de la liste et soit $k\in\mathbb{N}$ le nombre de passage dans le boucle **`for`** et S_k la valeur de la variable `S` à la fin du k-ème passage dans la boucle **`for`**. Donc S_0 (on est pas encore passé dans la boucle **`for`**). Si $\exists k\in\{0,\dots,n-1\}$ tel que $S_k=\sum_{j=0}^{k-1}L[j]$, alors $S_{k+1}=S_k+L[k]=\sum_{j=0}^{k}L[j]$. Donc, par récurrence, $\forall k\in\{0,\dots,n\}$, $S_k=\sum_{j=0}^{k-1}L[j]$.

Et donc, à la sortie de la boucle **`for`**, après le n-ème passage, l'algorithme renvoie $S_n^2=\left(\sum_{j=0}^{n-1}L[j]\right)^2$.