Analyse de complexité

EXERCICE 1 – ÉVALUONS LA COMPLEXITÉ DE QUELQUES ALGORITHMES

Déterminer la complexité des algorithmes suivants :

```
tab = tableau de taille n
tab[0] = 1
tab[1] = 1
tab[2] = 1
Pour i allant de 3 à n-1
Faire   somme = 0
        Pour j allant de i-3 à i-1
        Faire somme = somme + tab[j]
        tab[i] = somme
```

```
cpt = 0
Pour i allant de 1 à n
Faire   Pour j allant de 1 à n*n*n*n
        Faire   Pour k allant de 1 à racine_carree(n)
                Faire cpt = (cpt + 1) modulo 100
```

```
Pour i allant de 1 à n
Faire   cpt = 0
        k = i
        Tant que k > 0
        Faire   cpt = cpt + (k modulo 2)
                k = k // 2
        Afficher "Nombre :", i, "Somme des bits :",cpt
```

```
tab = tableau de taille n
tab[0] = 1
tab[1] = 1
tab[2] = 1
Pour i allant de 3 à n-1
Faire   somme = 0
        Pour j allant de 0 à i-1
        Faire somme = somme + tab[j]
        tab[i] = somme
```

```
Fonction Affichage_ZigZag(tab : tableau)
        Affiche_Bout(tab,VRAI)

Fonction Affiche_Bout(tab : tableau, on_affiche_minimum : booléen)
        Si longueur(tab) > 0
        Alors   Si (on_affiche_minimum)
                Alors m = minimum de tab
                Sinon m = maximum de tab
                Afficher m
                Supprimer m de tab
                Affiche_Bout(tab,non(on_affiche_minimum))
```

```
Fonction Puissance2modulo100(n : entier positif)
        Si n == 0
        Alors Renvoyer 1
        Sinon Renvoyer (Puissance2modulo100(n-1)+Puissance2modulo100(n-1)) modulo 100
```

EXERCICE 2 – DES FONCTIONS MYSTÈRE CLASSIQUES !

Q1. Qu'effectue la fonction "mystère" suivante ?

```
Fonction mystere ( x : flottant , n : entier )
        res = 1
        Pour i allant de 1 à n
        Faire   res = res * x
        Renvoyer res
```

Q2. Quelle est la complexité (en temps) de cette fonction `mystere` ?

Q3. Quelle est sa complexité en espace ?

Considérons maintenant la fonction suivante :

```
Fonction mystere2 ( x : flottant , n : entier )
        b = tableau de 0 et de 1 codant n en binaire
        res = x
        Pour i allant de 1 à (taille de b - 1)
        Faire   Si b[i] == 0
                Alors res = res * res
                Sinon res = res * res * x
        Renvoyer res
```

Pour la première ligne, on supposera que `b[0]` est le bit de poids fort (donc toujours égal à 1), et `b[taille de b - 1]` le bit de poids faible. Par exemple, si `n` vaut `18`, alors `b` sera le tableau `[1,0,0,1,0]`.

Q4. Calculez `mystere2(2.0,11)`.

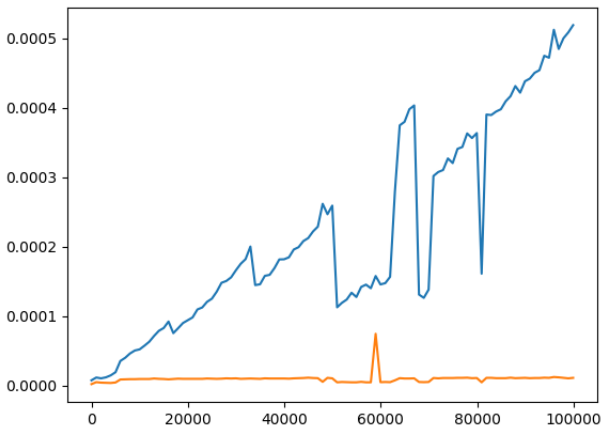
Q5. Quelle est la complexité en espace de `mystere2` ?

Q6. Prouvez que `mystere2` calcule la même chose que `mystere`. Indication : Il faut prouver par récurrence sur i qu'au bout de la i-ième itération, `res` vaut x^m où m est le nombre obtenu en convertissant en binaire le sous-tableau `b[0..i]`.

J'ai codé `mystere2` en python :

```python
def mystere2(x,n):
    b = list(map(int,bin(n)[2:]))
    res = x
    for i in range(1,len(b)):
        if b[i] == 0:
            res = res * res
        else:
            res = res * res * x
    return res
```

et ai testé d'une part le temps d'exécution de `mystere2(2,n)`, et d'autre part le temps d'exécution de `mystere2(2.0,n)` en fonction de n. Voici les deux courbes que j'ai obtenues :



Q7. Pouvez-vous deviner à quelle courbe correspond le temps d'exécution de `mystere2(2,n)`, et celui de `mystere2(2.0,n)` ? A votre avis, pourquoi une telle différence entre les deux courbes ?

EXERCICE 3 – RECHERCHE D'UN MOTIF DANS UN TEXTE (COMPLEXITÉ À PLUSIEURS VARIABLES ?)

Q1. Écrire (de manière naïve !) une fonction qui prend en paramètre deux chaînes de caractère `texte` et `motif` et qui renvoie `VRAI` si `motif` est bien une sous-chaîne de `texte` et `FAUX` dans le cas contraire.

Q2. Exprimez la complexité en temps de votre fonction en fonction de la longueur de `texte`, notée `n`, et la longueur de `motif`, notée `m`.

Attention ! La complexité finale doit bien dépendre de m.

Q3. Quelle est la complexité asymptotique en temps d'un appel de votre fonction lorsque :

1. le motif fait 10 lettres ? ($m = 10$)
2. le motif fait 3 lettres de moins que le texte ? ($m = n - 3$)
3. le motif a à peu près moitié moins de lettres que le texte ? ($m = n/2$)

EXERCICE 4 – DEUXIÈME MINIMUM

Toutes les fonctions suivantes sont censées renvoyer le deuxième plus petit élément d'un tableau (si le plus petit élément du tableau apparaît plus qu'une fois, alors il renvoie ce plus petit élément).

```
Fonction deuxieme_min_1 ( tab : tableau d'entiers )
        m = minimum(tab)
        Supprimer m de tab
        m = minimum(tab)
        Renvoyer m
```

```
Fonction deuxieme_min_2 ( tab : tableau d'entiers )
        Trier tab
        Renvoyer ?????
```

```
Fonction deuxieme_min_3 ( tab : tableau d'entiers )
        n = longueur(tab)
        indice_minimum = 0
        Pour i allant de 1 jusqu'à n-1
        Faire   Si (tab[i] < tab[indice_minimum])
                Alors   indice_minimum = i

        Si indice_minimum == 0
        Alors   indice_deuxieme_minimum = 0
        Sinon   indice_deuxieme_minimum = 1

        Pour j allant de 1 jusqu'à n-1
        Faire   Si (tab[j] < tab[indice_deuxieme_minimum]) et ?????
                Alors   indice_deuxieme_minimum = j

        Renvoyer tab[indice_deuxieme_minimum]
```

Q1. Remplacez les ????? de sorte que les fonctions renvoient bien le deuxième plus élément.

Q2. Quelle est la complexité asymptotique en temps de ses fonctions ?

Q3. Qu'auriez-vous écrit comme fonction ?