Informatique PCSI
TP 10b : analyse d'un programme

Exercice 1

Déterminer le nombre d'opérations mathématiques effectuées par le programme suivant en fonction de n avec n entier strictement positif.

Quelle est la complexité de l'algorithme en fonction de n ?

```
c = 0
while n > 0:
    c = c + 1
    n = n // 10
```

Exercice 2

Voici deux programmes pour calculer 2^n avec n entier naturel :

```
def puissance2(n):
    p = 1
    for k in range(1, n+1):
        p = 2 * p
    return p

def puissance2(n):
    p = 1
    b = 2
    while n > 0:
        n, r = n // 2, n % 2
        p = p * b ** r
        b = b * b
    return p
```

1. Dans le premier programme, quel est le nombre exact d'affectations et de multiplications effectuées dans le corps de la boucle en fonction de n ? En déduire le niveau de complexité.
2. Prouver la terminaison et la correction du deuxième programme en considérant l'invariant de boucle : $p \times b^n$.

   Déterminer en fonction de n le nombre d'opérations élémentaires effectuées dans le corps de la boucle (compter les opérations mathématiques et les affectations).

   Prouver que le niveau de complexité de cet algorithme est en $\mathcal{O}(\log_2 n)$.

Exercice 3

L'objectif est de comparer deux algorithmes permettant d'évaluer la valeur de $P(x)$ pour une valeur de x donnée avec $P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$.

La liste des coefficients est notée $a = [a_0,\, a_1,\, \ldots,\, a_n]$; attention à l'ordre !

Le programme ci-dessous correspondant au premier algorithme.

```
def polynome(x, a):
    p = a[0]
    for i in range(1, len(a)):
        puis = 1
        for j in range (1, i+1):
            puis = puis * x
        p = p + a[i] * puis
    return p
```

1. Étudier la complexité de ce programme en fonction de n.
2. Le programme correspondant au deuxième algorithme, appelé *algorithme de Hörner*, est basé sur une écriture différente de $P(x) : P(x) = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + xa_n)\ldots))$. Donc ici, la liste des coefficients est parcourue dans l'ordre inverse ; la variable p est initialisée avec a_n et nous n'utilisons qu'une seule boucle dans laquelle est effectuée une multiplication par x et l'addition du coefficient précédent. Écrire le programme et déterminer sa complexité.

Le principal est de bien gérer l'indice de boucle pour l'algorithme de Hörner.