Devoir à la Maison n°2

Numérotation en base b

Rappel des consignes :

Le langage utilisé doit être Python. Il est conseillé de tester les fonctions sur ordinateur.

Les programmes et fonctions demandés doivent être manuscrits, les impressions de listings ne sont pas acceptées.

Il est inutile au sein d'une fonction de vérifier que les variables d'entrée satisfont les conditions nécessaires pour que le calcul soit défini.

Les tests sont parfois explicitement demandés, sinon il n'est pas nécessaire de les faire figurer sur la copie.

Introduction.

La liste des chiffres de 3072 en base 10 est [3,0,7,2]. Ceci signifie que :

$$3072 = 3 \times 10^3 + 0 \times 10^2 + 7 \times 10^1 + 2 \times 10^0$$

Plus généralement, si n est un entier naturel, alors la liste de ses chiffres en base 10 est $[a_p, a_{p-1}, \ldots, a_0]$ où p est un entier naturel, les a_k sont des éléments de $\{0, 1, \ldots, 9\}$ (*i.e.,* les a_k sont des chiffres) et :

$$n = \sum_{k=0}^{p} a_k 10^k$$

Encore plus généralement, si n est un entier naturel alors la liste de ses chiffres en base b est $[a_p, a_{p-1}, \ldots, a_0]$ où p est un entier naturel, les a_k sont des éléments de $\{0, 1, \ldots, b-1\}$ (*i.e.,* des chiffres en base b) et :

$$n = \sum_{k=0}^{p} a_k b^k \qquad (*)$$

Par exemple en base $b = 2$ on ne dispose que de deux chiffres : 0 et 1. Les chiffres de 19 en base 2 sont [1,0,0,1,1] car :

$$19 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Le but de ce devoir est de programmer et étudier des algorithmes permettant d'obtenir un nombre à partir de la liste de ses chiffres en base b, ou d'obtenir la liste des chiffres d'un nombre en base b.

Partie A. Des chiffres au nombre

On commence par écrire une fonction `Nombre` qui reçoit une liste de chiffres L et qui renvoie l'entier n dont les chiffres sont les éléments de L.

La base b sera spécifiée en argument optionnel, égal à 10 par défaut.

Ceci signifie que cet argument n'est pas obligatoirement précisé, et s'il ne l'est pas alors il sera égal à 10.

```
def Nombre(L,b=10):          # b est un argument optionnel
    ...
```

Exemple :

```
print(Nombre([3,0,7,2]))      # doit renvoyer 3072
print(Nombre([1,0,0,1,1],2)) # doit renvoyer 19
```

1. Définir la fonction `Nombre` en Python en utilisant la formule $(*)$.

On rappelle que le coût en temps du calcul de x^k est de $k-1$ opérations.

Le type de complexité peut être logarithmique, linéaire, etc.

2. Quel est le coût en temps de la fonction `Nombre` ?
Quel est son type de complexité ?

On souhaite améliorer le coût de la fonction `Nombre`.

Pour toute suite $(x_0, x_1, \ldots, x_p)$ de nombres on définit par récurrence la suite $(u_i)_{0\leqslant i\leqslant p+1}$:

$$u_0 = 0 \qquad \text{et} \qquad \forall i = 0 \ldots p \quad u_{i+1} = bu_i + x_i$$

3. Démontrer par récurrence finie que pour tout $i = 0 \ldots p+1$: $u_i = \sum_{j=0}^{i-1} b^{i-1-j} x_j$.

En déduire que si $(x_0, \ldots, x_p) = (a_p, \ldots, a_0)$ alors u_{p+1} est le nombre n dont les chiffres en base b sont $a_p, \ldots, a_0$.

4. Écrire une nouvelle fonction `Nombre` utilisant cet algorithme.

5. Quel est le coût en temps de cette dernière fonction ?
Quel est son type de complexité ?

Partie B. Du nombre aux chiffres

On considère l'algorithme suivant :

1 **Données d'entrée :** n et b entiers naturels avec $b \geqslant 2$
2 **Définir :** L reçoit la liste vide, m reçoit n
3 **Tant que** $m > 0$ **itérer :**
4 **Définir :** r reçoit le reste de la division euclidienne de m par b
5 m reçoit le quotient de la division euclidienne de m par b
6 **Ajouter :** r au début de la liste L
7 **Donnée de sortie :** L

6. Programmer cet algorithme en Python, en tant que fonction `Chiffres(n,b=10)`.

On pourra tester

```
print(Chiffres(3072))
print(Chiffres(19,2))
```

Pour tout entier k on note r_k et m_k les valeurs contenues par les variables `r` et `m` à l'issue de la boucle de cet algorithme.

On précise que r_0 n'est pas défini alors que m_0 est la valeur de `m` avant la première itération.

7. Exprimer m_{k+1} en fonction de m_k en utilisant la fonction partie entière.
En déduire que la suite (m_k) est strictement décroissante, puis que la boucle de l'algorithme n'est pas infinie.

On note dorénavant q le nombre d'itérations de la boucle.

8. Écrire pour tout $k = 0 \dots q-1$ une relation entre m_k, m_{k+1}, r_{k+1} et b.

On définit pour tout $k = 0 \dots q$: $v_k = b^k m_k + \sum_{i=1}^{k} r_i b^{i-1}$.

9. Démontrer que v_k est un invariant de boucle, *i.e.,* qu'il ne dépend pas de k.

10. Expliciter v_q et démontrer qu'il est égal à n.
Justifier que la fonction renvoie $[r_q, \dots, r_1]$, que tous les r_k sont des entiers compris entre 0 et $b-1$, puis que l'algorithme renvoie la liste des chiffres de n en base b.

11. Démontrer que r_q est non-nul, puis que $b^{q-1} \leqslant n \leqslant b^q - 1$.
En déduire que $q = \left\lfloor \frac{\ln n}{\ln b} \right\rfloor + 1$.

12. Exprimer le coût en temps de l'algorithme en fonction de q puis de n.
Quel est son type de complexité ?