Travaux Dirigés d'algorithmique n°2

Cours d'algorithmique

—Licence CFA / Troisième Année—

Algorithmes numériques de base, complexités

Dans cette séance de travaux dirigés, nous étudions la complexité de plusieurs algorithmes numériques simples. Dans tout ce TD, on considère que l'affectation, la comparaison, la somme, l'accès au tableaux sont toutes des opérations qui prennent un temps constant.

▸ **Exercice 1. (Complexité).** Pour chaque programme de l'exercice 1 du TD n°1 dire quelle est la complexité. Peut-on retrouver cette complexité sans refaire le calcul ?

▸ **Exercice 2. (Somme des entrées d'un tableau).**

On considère l'algorithme suivant :

```
Entrée : un tableau de nombres T de taille n
Sortie : un nombre s
s = 0
pour i de 1 à n faire
    s = s + T[i]
retourner s
```

1. Que calcule l'algorithme ?
2. Réécrire la boucle «pour» en une boucle «tant que».
3. Pour chaque opération op parmis affectation, comparaison, somme et accès, donner en fonction de n le nombre de fois que op est exécutée. Qu'en pensez vous ?
4. Montrer que la complexité de cet algorithme est $\Theta(n)$. Peut-on faire mieux ?

▸ **Exercice 3. (Calcul du maximum d'un tableau).**

1. En s'inspirant de l'exercice précédent, écrire un algorithme qui prend en paramètre un entier n et un tableau de réels T de taille n, et qui retourne le maximum m des nombres du tableau et l'indice k d'une occurence de m dans le tableau (on s'autorise l'instruction `retourner a, b`).
2. Quel est la complexité si l'on compte comme opérations élémentaires les comparaisons de réels.
3. Quel est la complexité si l'on compte comme opérations élémentaires les affectations de réels.
4. Quel est la complexité si l'on compte comme opérations élémentaires les comparaisons et les affectations de réels.
5. Commenter.

▸ **Exercice 4. (Écriture d'un nombre en base 2).**

1. Décrire deux méthodes pour convertir un nombre n en base 2.
2. Écrire sous la forme d'un algorithme la plus simple des deux ; Quel est la complexité de cet algorithme ?

▸ **Exercice 5. (Calcul de a^n).**

1. Écrire un algorithme qui prend en paramètre un nombre a et un entier positif ou nul n et qui retourne a^n.
2. Quel est la complexité de cet algorithme.

On considère maintenant l'algorithme suivant

```
Entrée : un nombre a et un entier positif n
Sortie : p

n2  = n; a2i = a; res = 1
tant que n2 > 0 faire
    si n2 mod 2 == 1 alors
        res = res*a2i
    n2 = n2 / 2
    a2i = a2i*a2i
return res
```

3. Quel est le nombre d'étapes de boucle ?
4. Montrer que pour tout i, à la fin de la i-ème étape de boucle on a

$$\texttt{a2i} = \texttt{a}^{2^i} \qquad \text{et} \qquad \texttt{a}^{\texttt{n}} = \texttt{res} \times \texttt{a2i}^{\texttt{n2}}\,.$$

5. En déduire que l'algorithme retourne a^n.

▸ **Exercice 6. (Evaluation d'un polynôme dense).**

Un polynôme est une expression de la forme

$$P(x) = \sum_{i=0}^{d} c_i x^i\,.$$

On suppose de plus $c_d \neq 0$ (c'est-à-dire que d est le degré de P). On stocke un polynôme sous la forme d'une structure contenant deux variables : le degrée d et un tableaux $C[0..d]$ contenant les coefficients. On veut calculer la valeur de $P(x)$ pour une valeur de x donnée.

1. En utilisant la fonction puissance précédente écrire un algorithme qui répond au problème.
2. Quels sont les nombres d'opérations (additions, multiplications) effectués dans le calcul précédent ? Quel est sa complexité ?

Une première amélioration consiste à accumuler en même temps la valeur de $P(x)$ et les valeurs des puissances de x.

3. Écrire l'algorithme correspondant. Quels sont les nombres d'opérations effectués dans cette méthode ? Quel est sa complexité ?

En fait, on peut faire encore mieux en utilisant la méthode de Hörner. Elle provient de l'écriture suivante du polynôme :

$$P_n(x) = (\cdots((a_n x + a_{n-1})x + a_{n-2})x + \cdots + a_1)x + a_0$$

4. Mêmes questions ?
5. Comparer les trois méthodes.