Licence Informatique ISTIC
Module ALG *2015-2016*

Méthodes algorithmiques
TD 0

Cette séance de travaux dirigés est consacrée à la notion de complexité, à l'étude du comportement asymptotique des fonctions élémentaires, et à la correction et la complexité de plusieurs algorithmes simples. Pour montrer qu'un algorithme est correct, on écrit une propriété qui est conservée à chaque étape de boucle que l'on appelle **invariant de boucle**.

Exercice 1 (Nombres d'opérations de programmes élémentaires)
Pour chacun des programmes suivants, dire en fonction de n quel est le nombre d'opérations `op` que le programme effectue.

1.
```
pour i de 1 à n faire op
```

2.
```
pour i de 0 à n faire op
```

3.
```
pour i de 1 à n-1 faire op
```

4.
```
pour i de 1 à n faire
     op; op
```

5.
```
pour i de 1 à n faire
    pour j de 1 à n faire op
```

6.
```
pour i de 1 à n faire
    pour j de 1 à n faire
        pour k de 1 à n faire op
```

7.
```
pour i de 1 à n faire
    op; op
    pour j de 1 à n faire
        op
        pour k de 1 à n faire
            op
```

8.
```
pour i de 1 à n faire
    pour j de 1 à i faire
        op
```

9.
```
tant que n >= 0 faire
    op
    n := n / 2
```

Exercice 2 (Manipulation des ordres de grandeurs)
Dans cet exercice f, g, h sont des fonctions positives.

1. Montrer que si $f \in O(g)$ et $g \in \Theta(h)$ alors $f \in O(h)$.
2. Montrer que si $f \in \Theta(g)$ et $g \in O(h)$ alors $f \in O(h)$.
3. Montrer par un contre-exemple que si $f \in \Theta(g)$ et $g \in O(h)$ alors f n'est pas nécessairement dans $\Theta(h)$.

Exercice 3 (Somme des entrées d'un tableau)
On considère l'algorithme suivant :

```
Entrée : un tableau de nombres A de taille n
Sortie : un nombre s
1. s = 0
2. i = 0
3. tant que i < n faire
4.    s = s + A[i]
5.    i = i + 1
6. retourner s
```

On veut montrer que l'algorithme calcule $\sum_{i=0}^{n-1} \mathtt{A}[i]$.

1. On considère une étape de boucle. On note s et s' les valeurs de la variable `s` avant et après l'étape de la boucle, et i et i' les valeurs de la variable `i` avant et après l'étape de la boucle. Montrer que si $s = \sum_{j=0}^{i-1} \mathtt{A}[j]$ alors $s' = \sum_{j=0}^{i'-1} \mathtt{A}[j]$.

On appelle l'énoncé "à la fin d'une étape de boucle, les variables `s` et `i` vérifient l'égalité $\mathtt{s} = \sum_{j=0}^{\mathtt{i}-1} \mathtt{A}[j]$" un **invariant de boucle**.

2. Montrer que avant la boucle l'invariant est vérifié.
3. En déduire que à la fin de la boucle $s = \sum_{j=0}^{n-1} \mathtt{A}[j]$.
4. Montrer que la complexité de cet algorithme est $\Theta(n)$.

Exercice 4 (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 d'entiers `A` de taille n, et qui retourne le maximum m des entiers du tableau et l'indice k d'une occurence de m dans le tableau.
2. Donner un énoncé qui caractérise le fait que m et k sont corrects. Un tel énoncé est appelé une **spécification**[1].
3. À l'aide de la spécification, écrire un invariant de boucle et montrer qu'il est correct, c'est-à-dire qu'il est vrai au début de la boucle et qu'il est conservé à chaque étape de boucle.
4. En déduire, que l'algorithme fonctionne.
5. Quelle est la complexité si l'on prend comme opérations élémentaires les comparaisons.
6. Quelle est la complexité si l'on prend comme opérations élémentaires les affectations.
7. Quelle est la complexité si l'on prend comme opérations élémentaires les comparaisons et les affectations.

[1] C'est le contrat que l'utilisateur de l'algorithme passe avec le développeur.

Exercice 5 (Écriture d'un nombre en base 2)

1. Décrire deux méthodes pour convertir un nombre n en base 2.

On considère l'algorithme suivant :

```
Entrée : un entier n et un tableau A alloué.
Sortie : un tableau A et un entier i
1. m  = n
2. i = 0
3. tant que m > 0 faire
4.    A[i] = m mod 2
5.    m   = m div 2
6.    i    = i + 1
7. retourner A, i
```

2. Montrer que pour tout i, à la fin de la i-ème étape de boucle $n = m \times 2^i + \sum_{j=0}^{i-1} \mathtt{A}[j] \times 2^j$.

3. En déduire qu'à la fin `A` contient l'écriture de n en base 2 càd. $n = \sum_{j=0}^{i-1} \mathtt{A}[j] \times 2^j$.

Exercice 6 (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. À l'aide d'un invariant de boucle montrer que l'algorithme est correct.

3. Quelle est la complexité de cet algorithme ?

On considère maintenant l'algorithme suivant :

```
Entrée : un nombre a et un entier positif n
Sortie : un nombre res
1. m  = n
2. b = a
3. res = 1
4. tant que m > 0 faire
5.    si m mod 2 == 1 alors
6.        res = res*b
7.    m = m div  2
8.    b = b*b
9. return res
```

4. Quel est le nombre d'étapes de boucle ?

5. Montrer que à la fin de chaque étape de boucle, on a

$$\mathtt{a}^{\mathtt{n}} = \mathtt{res} \times \mathtt{b}^{\mathtt{m}} .$$

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

L'exercice suivant est à rendre pour la prochaine séance de TD.

Exercice 7 (Calcul de $n!$)
On considère l'algorithme suivant :

```
Entrée: un entier n>0
Sortie: la factorielle n!
1. res = 1
2. m = n
3. tant que m>0 faire
4.   res = res*m
5.   m  = m - 1
6. retourner res
```

1. Montrer que l'invariant
$$\texttt{res} = \prod_{i=\texttt{m}+1}^{n} i \tag{1}$$
est vrai au départ de la boucle et conservé ensuite.
2. En déduire que le programme retourne $n!$ (on rappelle que $n! = \prod_{i=1}^{n} i$).
3. Quelle est la complexité de cet algorithme ?