Complexité des algorithmes
M1 MIAGE – TD 2

1 Évaluation de complexité avec deux paramètres

Déterminer la complexité des algorithmes suivants :

```
function Nabla1(m,n:integer)
begin
   i := 1;
   j := 1;
   Tant que (i <= m) et (j <= n) faire
       i := i+1
       j := j+1
   fin tant que

function Nabla2(m,n:integer)
begin
   i := 1;
   j := 1;
   Tant que (i <= m) ou (j <= n) faire
       i := i+1
       j := j+1
   fin tant que

function Nabla3(m,n:integer)
begin
   i := 1;
   j := 1;
   Tant que (j <= n) faire
       Si i <= m alors
           i := i+1
       Sinon
           j := j+1
       fin si
   fin tant que

function Nabla4(m,n:integer)
begin
   i := 1;
   j := 1;
   Tant que (j <= n) faire
       Si i <= m alors
           i := i+1
       Sinon
           j := j+1
           i := 1
       fin si
   fin tant que
```

2 Reconnaissance d'un langage

Donner un algorithme pour répondre, à partir d'un entier n et d'un vecteur de nombres parmi 0, 1 et 2, si oui ou non le vecteur constitue une suite de la forme n fois le chiffre 1, puis n fois le chiffre 2, puis le chiffre 0 (c'est à dire 1^n2^n0).

1. Donner sa complexité en fonction de n.
2. Pourquoi est-ce équivalent à la complexité en fonction de la taille de l'entrée ?

3 Évaluation de complexité

Calculer la complexité du programme ci-dessous en nombre d'instructions `inst` exécutées, en fonction de la valeur de `n`.

```
for i=1 to n-1 do
    for j= i+1 to n do
        for k= 1 to j do
            X[i,j,k] := inst(X[i,j,k] )
        end do
     end do
end do;
```

Comment pourrait-on savoir si la complexité moyenne est la même que la complexité pire-cas ?

4 Entier manquant

On dispose d'un tableau non trié $A[1 \dots n]$ qui contient tous les entiers de 0 à n, sauf un.

1. Donnez un algorithme qui trouve l'entier manquant en temps $O(n^2)$.
2. Donnez un algorithme qui trouve l'entier manquant en temps $O(n \log n)$
3. Donnez un algorithme qui trouve l'entier manquant en temps $O(n)$.
4. Donnez un algorithme qui trouve l'entier manquant en temps $O(n)$, sans tableau auxiliaire.

5 Évaluation de complexité

Expliquer précisément ce que réalisent les fonctions suivantes sur les entiers naturels :

1.
```
function R1(n:integer) return integer is
begin
    if (n <= 0) then
        return 1
    else
        return 2 * R1(n-1)
    end if
end R1.
```
2.
```
function R2(n:integer) return integer is
begin
    if (n <= 0) then
        return 1
    else
       return R2(n-1) + R2(n-1)
    end if
end  R2.
```

Calculer et comparer la complexité en nombre d'appels récursifs des fonctions R1 et R2. Expliquer pourquoi on peut néglier ainsi le nombre exact d'opérations et ne compter que les appels de fonction.

6 Évaluation de complexité

Expliquer précisément ce que définissent les fonctions suivantes sur les entiers naturels :

1.
```
function f1(a,n:integer) return integer is
begin
    if (n=0) then
        return 1
    else
        return a * f1(a,n-1)
    end if
end.
```

2.
```
function f2(a,n:integer) return integer is
x: integer
begin
    x:= 1;
    for i= 0 to n do
        x:= x*a
    end for;
    return x
end.
```

3.
```
function f3(a,n:integer) return integer is
begin
    if (n=0) then
        return 1
    else if (n mod 2=0) then
        return f3(a*a, n/2)
    else
        return a* f3(a*a, n/2)
    end if
end.
```

7 Produit matriciel

Considérons deux matrices carrées d'entiers d'ordre n. Notons A et B ces deux matrices. Le produit de A par B résulte en une matrice carrée C définie par :

$$C_{i,j} = \sum_{k=1}^{n} A_{i,k} \cdot B_{k,j}.$$

1. Donnez un algorithme qui calcule un tel produit de matrices. Nous supposerons que ces matrices sont représentées par un tableau à deux dimension.
2. Calculer la complexité de cet algorithme. Préciser si celle-ci est au pire des cas, au meilleur des cas, au cas moyen.
3. Adapter l'algorithme pour le cas où A est de dimension $m \times n$ et B de dimension $n \times p$. Quelle est alors la complexité de l'algorithme ?

8 Boucles imbriquées

Calculer la complexité du programme ci-dessous en nombre d'appel de la fonction `f` en fonction de n dans les deux cas suivants :

```
for i = 1 to n-1 do
    for j = i+1 to n do
        for k = 1 to j do
            t[i,j,k] := f(t[i,j,k])
        end do
    end do
end do
```

```
for i = 1 to n do
    for j = 1 to i do
        for k = j to i do
            t[i,j,k] := f(t[i,j,k])
        end do
    end do
end do
```

9 Calcul d'un nombre de Fibonacci

Leonardo Fibonacci (1175 – 1250) etait un mathématicien italien. En particulier, il a écrit plusieurs livres sur le calcul, la géométrie, la trigonométrie, ... La suite de Fibonacci apparait dans l'un de ses ouvrage : le *Liber Abaci*.

« Un homme met un couple de lapins dans un lieu isolé de tous les côtés par un mur. Combien de couples obtient-on en un an si chaque couple engendre tous les mois un nouveau couple à compter du troisième mois de son existence ? »

Si on note $\mathcal{F}_n$ le nombre de couples au début du n-ième mois, on obtient la récurrence :

$$\begin{cases} \mathcal{F}_1 = \mathcal{F}_2 = 1 \\ \mathcal{F}_{n+2} = \mathcal{F}_n + \mathcal{F}_{n+1} \end{cases}$$

- Effectuez *à la main* le calcul du huitième nombre de Fibonacci d'abord par l'algorithme de complexité exponentielle, puis de complexité linéaire et enfin par l'algorithme de complexité logarithmique.
- Ecrivez le pseudo-code complet (ou le programme Pascal/Ada/Java/Caml/C/C++/Python si vous préférez) de l'algorithme de complexité logarithmique pour calculer le n-ième nombre de Fibonacci.

10 Partition

On dispose d'un tableau non trié $A[1 \dots n]$ de n entiers. On souhaite décider s'il existe un sous-ensemble $I \subseteq \{1, 2, \dots, n\}$ tel que

$$\sum_{i \in I} A[i] \quad = \quad \sum_{i \in \{1,2,\dots,n\} \setminus I} A[i]$$

Proposez un algorithme pour résoudre ce problème. Quelle est sa complexité en temps ?