**U.S.T.H.B - 06-12-2011**
**Département Informatique**
***N. Bensaou - C. Ighilaza***

**Complexité**
**Master M1- RSD**

| Nom | Prénom | Matricule |
|---|---|---|
| | | |

# Contrôle n°1 - durée 1h30 mn

Exercice 1.- Prouver l'expression suivante:

$$\sum_{i=1}^{n} \frac{i}{2^i} = 2 - \frac{n}{2^n} - \frac{2}{2^n}$$

Exercice 2.- Soit $A[1..N]$ un tableau de caractères. On veut vérifier si $A$ contient un mot palindrome ou non. si .

a- Écrire une fonction itérative qui retourne 1 si $A$ contient un palindrome et 0 sinon

b- Donner un invariant de boucle

c- Prouver la correction de cet algorithme. Donnez sa complexité.

d- Écrire une solution récursive pour le même problème. Donnez l'invariant de boucle.

Exercice 3.- a- Trouver les ordres de grandeur en notation "grand O" des fonctions suivantes:

$3n^3 + 2^{n-2}$; $4n^3 + 12$; $n^2 log(5n^4)$; $\frac{1}{2}n^2 - 10n - 60$; $\frac{1}{n}$

b- Classer ces fonction par ordre croissant de leur ordre de grandeur.

Exercice 4.- Soient $f, g, S, T$ des fonctions de $\mathcal{N} \longrightarrow \mathcal{N}$. On suppose que:

$$\begin{array}{rcl} S(n) & \in & O(f(n)) \\ T(n) & \in & O(g(n)) \end{array}$$

Montrer que $S(n)T(n) \in O(f(n)g(n))$.

**Master RSD (2011-2012)**

**Corrigé du contrôle N°1 du module Complexité**

***<u>Exercice 1 :</u>***

$$\sum_{i=1}^{n} \frac{i}{2^i} = 2 - \frac{n}{2^n} - \frac{2}{2^n}$$

On prouve l'expression par récurrence.

Pour n=1 on a

$\sum_{i=1}^{1} \frac{i}{2^i} = 2 - \frac{1}{2^1} - \frac{2}{2^1}$ ?

$$\sum_{i=1}^{1} \frac{i}{2^i} = \frac{1}{2^1}$$

$2 - \frac{1}{2^1} - \frac{2}{2^1} = 2 - \frac{1}{2} - 1 = 1 - \frac{1}{2} = \frac{2}{2} - \frac{1}{2} = \frac{1}{2}$

Donc l'expression est vraie pour n=1, on suppose qu'elle est vraie pour n et on montre qu'elle est vraie pour n+1.

$$\sum_{i=1}^{n+1} \frac{i}{2^i} = 2 - \frac{n+1}{2^{n+1}} - \frac{2}{2^{n+1}}$$

$$\sum_{i=1}^{n+1} \frac{i}{2^i} = \sum_{i=1}^{n} \frac{i}{2^i} + \frac{n+1}{2^{n+1}} = 2 - \frac{n}{2^n} - \frac{2}{2^n} + \frac{n+1}{2^{n+1}}$$

$$= 2 - \frac{2n}{2^{n+1}} - \frac{4}{2^{n+1}} + \frac{n+1}{2^{n+1}}$$

$$= 2 + \frac{-2n-4+n+1}{2^{n+1}}$$

$$= 2 + \frac{-n-3}{2^{n+1}}$$

$$= 2 + \frac{-n-2-1}{2^{n+1}}$$

$$= 2 + \frac{-(n+1)-2}{2^{n+1}}$$

$= 2 - \frac{n+1}{2^{n+1}} - \frac{2}{2^{n+1}}$ Ce qu'il fallait démontrer

Donc l'expression est Vraie.

***Exercice 2 :***

a- fonction PalindromeIter(A :tableau[1..n] de caractères ; n :entier) :entier ;

```
var i, j :entier ;
debut
    i← 1;  j←n;
    tantque (i<j) et (A[i]=A[j]) faire
       i←i+1;  j←j-1;
    fait;
    si (i>=j) alors retourner 1
    sinon  retourner 0 ;
    fsi ;
 fin ;
```

b- ***Invariant de boucle*** **:**

« A la fin de la $k^{ième}$ itération de la boucle ***tantque*** tous les caractères ***A[1 ..i]*** sont égaux à ***A[n..j]*** puis on incrémente **i** et on décrémente **j.** La fonction s'arrête si $i_{k+1} \geq j_{k+1}$ et retourne **1** ou bien elle s'arrête si $A[i_{k+1}] \neq A[j_{k+1}]$ et retourne **0** »

c- On montre la validité de l'invariant par récurrence sur i.

L'invariant est facilement vérifié pour la **1ère** itération on a i=1 et j=n si A[1]=A[n] alors on incrémente $i_2=i_1+1$ et on décrémente $j_2=j_1-1$, si i>=j alors la fonction se termine et le mot est un palindrome sinon si A[1] $\neq$ A[n] la fonction se termine et le mot n'est pas un palindrome.

On suppose que l'invariant est vrai à la fin de l'itération **i** et on montre qu'il est vrai, s'il y en a, pour l'itération **i+1**, c'est que $\forall\, k \in \{1..i\}$ et $l \in \{n..j\}$ A[$k$]=A[$l$]. Si $k_{i+1} < l_{i+1}$ et $A[k_{i+1}] = A[l_{i+1}]$ on incrémente $k$ et on décrémente $l$ sinon si $k_{i+1} \geq l_{i+1}$ alors la fonction se termine et le mot est un palindrome sinon si $A[k_{i+1}] \neq A[l_{i+1}]$ alors la fonction se termine et le mot n'est pas un palindrome. L'invariant à la fin de l'itération i+1 est donc vérifié.

Au pire des cas le mot est un palindrome et aura fait n/2 comparaisons, donc la complexité au pire des cas est de ***O(n/2)*** et dans le meilleur des cas la fonction s'arrête quand A[1] ≠ A[n] donc la complexité est en O(1).

d- fonction PalindromeRec(A :tableau[1..n] de caractères ; n, i, j :entier) :entier ;

```
debut                                        (1er appel i=1 et j=n)
    si (i>=j) alors retourner 1
    sinon si (A[i]≠A[j]) alors retourner 0
          sinon   retourner(PalindromeRec(A, n, i+1, j-1);
          fsi ;
     fsi ;
 fin ;
```

***<u>Invariant :</u>***

« A la fin de la $k^{ième}$ itération si i>=j la fonction se termine et le mot est un palindrome ou bien A[i] ≠ A[j] et la fonction se termine et le mot n'est pas un palindrome sinon on fait un appel récursif en incrémentant i et en décrémentant j »