TD02 Algorithmique Avancée
Notations asymptotiques et complexité algorithmique

Exercice 1.

Montrer que la relation O(.) « est de l'ordre de » est une relation réflexive, transitive, mais pas symétrique.

Exercice 2.

Trouver l'erreur

a) $O(n) = O(n^3 + (n - n^3)) = O(\max\{n^3, n - n^3\}) = O(n^3)$
b) $O(n^2) = O(n + n + \dots + n) = O(\max\{n, \dots, n\}) = O(n)$

Exercice 3.

Le but de cet exercice est de tester votre compréhension de la notion de complexité dans le pire des cas.

1) Si je prouve que la complexité dans le pire des cas d'un algorithme est en $O(n^2)$, est-il possible qu'il soit en O(n) sur certaines données ?
2) Si je prouve que la complexité dans le pire des cas d'un algorithme est en $O(n^2)$, est-il possible qu'il soit en O(n) sur toutes les données ?
3) Si je prouve que la complexité dans le pire des cas d'un algorithme est en $\Theta(n^2)$, est-il possible qu'il soit en O(n) sur certaines données ?
4) Si je prouve que la complexité dans le pire des cas d'un algorithme est en $\Theta(n^2)$, est-il possible qu'il soit en O(n) sur toutes les données ?

Exercice 4.

Sur une échelle croissante, classer les fonctions suivantes selon leur comportement asymptotique : c'est-à-dire $g(n)$ suit $f(n)$ si $f(n) = O(g(n))$.

$f_1(n) = 2n$	$f_2(n) = 2^n$	$f_3(n) = \log(n)$	$f_4(n) = \frac{n^3}{3}$
$f_5(n) = n!$	$f_6(n) = \log(n)^2$	$f_7(n) = n^n$	$f_8(n) = n^2$
$f_9(n) = n + \log(n)$	$f_{10}(n) = \sqrt{n}$	$f_{11}(n) = \log(n^2)$	$f_{12}(n) = e^n$
$f_{13}(n) = n$	$f_{14}(n) = \sqrt{\log(n)}$	$f_{15}(n) = 2^{\log_2(n)}$	$f_{16}(n) = n\log(n)$

Exercice 5.

Parmi les assertions suivantes, lesquelles sont vraies ?

a) $3^n \in O(2^n)$
b) $2^n \in O(3^n)$
c) $2^n \in O(n!)$
d) $n! \in O(2^n)$
e) $n! \in O(2^{n\log(n)})$
f) $5n \in O(2n)$
g) $n^2 \in O(10^{-5} n^3)$
h) $25n^4 - 19n^3 + 13n^2 \in O(n^4)$
f) $2^{n+100} \in O(2^n)$
j) $k \in O(1)$ (k est une constante)

Exercice 6.

Calculer la complexité de l'algorithme suivant :

```
// a,i,j,k,x entiers
a = 1 ;
Pour i = 1 à x faire
        k = 0 ;
        Pour j = 1 à 2a faire
                k = k+1 ;
        finpour
        a = k ;
finpour
```

Exercice 7.

Déterminer la complexité des algorithmes suivants par rapport au nombre d'itérations effectuées où m et n sont deux entiers positifs : (i,j,n,m entiers)

Algorithme A

i=1 ; j=1 ;

Tantque (i≤m) et (j≤n) **faire** i=i+1; j=j+1 **fintantque**

Algorithme B

i=1; j=1;

Tantque (i≤m) ou (j≤n) **faire** i=i+1 ; j=j+1 **fintantque**

Algorithme C

i=1 ; j=1 ;

Tantque (j≤n) **faire Si** (i≤m) **alors** i=i+1 **sinon** j=j+1 **finsi fintantque**

Algorithme D

i=1; j=1;

Tantque (j≤n) **faire Si** (i≤m) **alors** i=i+1 **sinon** j=j+1; i=1 **finsi fintantque**