T.D. 1

Exercice 1

Montrer que les assertions suivantes sont exactes:

(a) $n(n-1)$ est en $O(n^2)$;

(b) $\max(n^3, 10n^2)$ est en $O(n^3)$;

(c) Si $p(x)$ est un polynôme quelconque de degré k, à coefficients entiers positifs, alors $p(n)$ est en $\Theta(n^k)$.

Exercice 2

Soient $f(x)$ et $g(x)$ des fonctions positives asymptotiquement. En s'aidant de la définition de base de la notion Θ, montrer que: $f(n) + g(n) = \Theta(\max(f(n), g(n)))$.

Exercice 3

Soient $f(x)$ et $g(x)$ des fonctions positives asymptotiquement. On suppose que $S(n) \in O(f(n))$ et $T(n) \in O(g(n))$. Montrer que:

(a) si $f(n) \in O(g(n))$ alors $S(n) + T(n) \in O(g(n))$;

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

Exercice 4

Peut-on écrire:

(a) $2^{n+1} \in O(2^n)$?

(b) $2^{2n} \in O(2^n)$?

Exercice 5

Soient $f(x)$ et $g(x)$ des fonctions asymptotiquement strictement positives, prouver ou démentir les affirmations suivantes:

(a) $f(n) = O(g(n))$ implique que $g(n) = O(f(n))$;

(b) $f(n) + g(n) = \Theta(\min(f(n), g(n)))$;

(c) $f(n) = O(g(n))$ implique que $2^{f(n)} = O(2^{g(n)})$.

Exercice 6

En utilisant la notation O, donner, en fonction de n, la complexité dans le pire des cas des algorithmes suivants:

(a) Algorithme $prodmat$:

```
for i in range (n):
    for j in range (n):
        C [i] [j] = 0
        for k in range (n):
            C [i] [j] = C [i] [j] + A [i] [k] * B [k] [j]
```

(b) Algorithme $mystere$:

```
l = 0
for i in range (n - 1):
    for j in range (i + 1, n + 1):
        for k in range (j):
            l = l + 1
```

(c) Algorithme $oddidonc$:

```
x = 0
y = 0
for i in range (n):
    if (i % 2) == 1:
        for j in range (n):
            x = x + 1
        for j in range (n):
            y = y + 1
```

(d) Fonction $recursive$:

```
def recursive (n):
    if n == 0:
        return 2
    return recursive (n - 1) + recursive (n - 1)
```

(e) Si on remplace l'instruction "$return\ recursive(n-1) + recursive(n-1)$" par "$return\ 2 * recursive(n-1)$", la complexité de la fonction $recursive$ est-elle modifiée?

Exercice 7

(a) Ecrire un algorithme itératif qui prenne en entrée un entier n et retourne le factoriel de cet entier et prouver cet algorithme.

(b) Déterminer la complexité de cet algorithme.

(c) Ecrire un algorithme récursif qui calcule $n!$ puis prouver cet algorithme.

(d) Déterminer la complexité de cet algorithme récursif.

Exercice 8

Donner une fonction Min qui, étant donné un tableau T de n valeurs entières, retourne la plus petite de ces valeurs. Donner la complexité, en fonction de n, de cette fonction.