Algoritmos y Estructuras de Datos

Guía Práctica 6
Complejidad

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ejercicio 1. Probar utilizando las definiciones que:

a) $n^2 - 4n - 2 = O(n^2)$.

b) Para todo $k \in \mathbb{N}$ y toda función $f : \mathbb{N} \to \mathbb{N}$, si $f \in O(n^k)$, entonces $f \in O(n^{k+1})$.

c) Si $f : \mathbb{N} \to \mathbb{N}$ es tal que $f \in O(\log n)$, entonces $f \in O(n)$.

Ejercicio 2. Determinar la verdad o falsedad de cada una de las siguientes afirmaciones. **Justificar**.

a) $2^n = O(1)$.

b) $n = O(n!)$.

c) $n! = O(n^n)$.

d) $2^n = O(n!)$.

e) Para todo $i, j \in \mathbb{N}$, $i \cdot n = O(j \cdot n)$.

f) Para todo $k \in \mathbb{N}$, $2^k = O(1)$.

g) $\log n = O(n)$.

h) $n! = O(2^n)$.

i) $n^5 + bn^3 \in \Theta(n^5) \iff b = 0$.

j) Para todo $k \in \mathbb{R}$ $n^k log(n) \in O(n^{k+1})$.

k) Para toda función $f : \mathbb{N} \to \mathbb{N}$, $f = O(f)$.

Ejercicio 3. ¿Qué significa, intuitivamente, $O(f) \subseteq O(g)$? ¿Qué se puede concluir acerca del crecimiento de f y g cuando, simultáneamente, tenemos $O(f) \subseteq O(g)$ y $O(g) \subseteq O(f)$?

Ejercicio 4. Determinar el orden de complejidad temporal de peor caso de los siguientes algoritmos, asumiendo que todas las operaciones sobre arreglos y matrices toman tiempo $O(1)$. La complejidad se debe calcular en función de una medida de los parámetros de entrada, por ejemplo, la cantidad de elementos en el caso de los arreglos y matrices y el valor en el caso de parámetros naturales.

a) Sumatoria, que calcula la sumatoria de un arreglo de enteros:

```
function Sumatoria(arreglo A)
    int i, total;
    total := 0;
    for i := 0 ... Long(A) - 1 do
        total := total + A[i];
    end for
end function
```

b) SumatoriaLenta, que calcula la sumatoria de n, definida como la suma de todos los enteros entre 1 y n, de forma poco eficiente:

```
function SumatoriaLenta(nat n)
    int i, total;
    total := 0;
    for i := 1 ... n do
        for j := 1 ... i do
            total := total + 1;
        end for
    end for
end function
```

c) ProductoMat, que dadas dos matrices A (de $p \times q$) y B (de $q \times r$) devuelve su producto AB (de $p \times r$):

```
function ProductoMat(matriz A, matriz B)
    int fil, col, val, colAFilB;
    matriz res(Filas(A), Columnas(B));
    for fil := 0 ... Filas(A) - 1 do
        for col := 0 ... Columnas(B) - 1 do
            val := 0;
            for colAFilB := 0 ... Columnas(A) - 1 do
                val := val + (A[fil][colAFilB] * B[colAFilB][col]);
            end for
            res[fil][col] := val;
        end for
    end for
    return res;
end function
```

d) InsertionSort, que ordena un arreglo pasado como parámetro:

```
function InsertionSort(arreglo A)
    int i, j, valor;
    for i := 0 ... Long(A) - 1 do
        valor := A[i];
        j := i - 1;
        while j ≥ 0 ∧ a[j] > valor do
            A[j+1] := A[j];
            j := j - 1;
        end while
        A[j+1] := valor;
    end for
end function
```

e) BúsquedaBinaria, que determina si un elemento se encuentra en un arreglo, que debe estar ordenado:

```
function BúsquedaBinaria(arreglo A, elem valor)
    int izq := 0, der := Long(A) - 1;
    while izq < der do
        int medio := (izq + der) / 2;
        if valor < A[medio] then
            der := medio;
        else
            izq := medio;
        end if
    end while
    return A[izq] = valor;
end function
```

f) AlgoritmoQueHaceAlgo:

```
function AlgoritmoQueHaceAlgo(arreglo A)
    int i := 1; int j := 1;
    int suma := 1; int count := 0;
    while i ≤ tam(A) do
        if i ≠ A[i] then
            count := count+1;
        end if
        j := 1;
        while j ≤ count do
            int k := 1;
            while k ≤ tam(A) do
                suma := suma + A[k];
                k := k * 2;
            end while
            j := j+1;
        end while
        i := i+1;
    end while
    return suma
```

end function

Ejercicio 5. Para cada una de las siguientes afirmaciones, decida si son verdaderas o falsas y justifique su decisión.

a) $O(n^2) \cap \Omega(n) = \Theta(n^2)$

b) $\Theta(n) \cup \Theta(n \log n) = \Omega(n \log n) \cap O(n)$

c) $f \in O(g) \iff O(f) \subseteq O(g)$

d) Si $f \in \Omega(g)$, entonces $O(f) \cap \Omega(g) = O(g) \cap \Omega(f)$

e) Si $f(n) < g(n)$ para todo n, entonces $\Theta(f)! = \Theta(g)$

f) Si $f \in O(g)$, entonces $f * g \in \Theta(g)$

Ejercicio 6. Para cada una de las siguientes afirmaciones, decida si son verdaderas o falsas y justifique su decisión.

a) $n + m = O(nm)$.

b) $n + m^5 = O(m^5)$.

c) $nm = O(n + m)$.

d) $m^5 = O(n + m^5)$.

Ejercicio 7. Sean $f, g : \mathbb{N} \to \mathbb{N}$, y supongamos que está definido el límite $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = \ell \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$. Probar que:

a) $0 < \ell < +\infty$ si y sólo si $f \in \Theta(g)$.

b) $\ell = +\infty$ si y sólo si $f \in \Omega(g)$ y $f \notin O(g)$.

c) $\ell = 0$ si y sólo si $f \in O(g)$ y $f \notin \Omega(g)$.

Recordar las definiciones de límite:

- $\lim_{n\to+\infty} a_n = \ell \in \mathbb{R}$ si $\forall\, \varepsilon > 0.\, \exists n_0 \in \mathbb{N}.\, \forall n > n_0.\, |a_n - \ell| < \varepsilon$.
- $\lim_{n\to+\infty} a_n = +\infty$ si $\forall\, M > 0.\, \exists n_0 \in \mathbb{N}.\, \forall n > n_0.\, a_n > M$.