# Computabilidad y Complejidad

## Práctica 8

1) Determinar para cada función $t(n)$ en la siguiente tabla, cuál es el mayor tamaño $n$ de una instancia de un problema que puede ser resuelto en cada uno de los tiempos indicados en las columnas de la tabla, suponiendo que el algoritmo para resolverlo utiliza $t(n)$ microsegundos.

| $t(n)$ | 1 seg. | 1 min. | 1 hora | 1 día | 1 mes | 1 a no | 1 siglo |
|---|---|---|---|---|---|---|---|
| $log_2(n)$ | | | | | | | |
| $\sqrt{n}$ | | | | | | | |
| $n$ | | | | | | | |
| $n \times log_2(n)$ | | | | | | | |
| $n^2$ | | | | | | | |
| $2^n$ | | | | | | | |
| $n!$ | | | | | | | |

2) Si el tiempo de ejecución en el mejor caso de un algoritmo, $t_m(n)$, es tal que $t_m(n) \in \Omega(f(n))$ y el tiempo de ejecución en el peor caso de un algoritmo, $t_p(n)$, es tal que $t_p(n) \in O(f(n))$, ¿Se puede afirmar que el tiempo de ejecución del algoritmo es $\Theta(f(n))$?

3) Un algoritmo tarda 1 segundo en procesar 1000 items en una máquina determinada. ¿Cuánto tiempo tomará procesar 10000 items si se sabe que el tiempo de ejecución del algoritmo es $n^2$? ¿y si se sabe que es $n \times log_2 n$? ¿Qué se estaría asumiendo en todos los casos?

4) Un algoritmo toma $n^2$ días y otro $n^3$ segundos para resolver una instancia de tamaño $n$ de un problema. Mostrar que el segundo algoritmo superará en tiempo al primero solamente en instancias que requieran más de 20 millones de años para ser resueltas.

5) ¿Cuáles y cuántas serían las operaciones elementales necesarias para multiplicar dos enteros $n$ y $m$ por medio del algoritmo enseñado en la escuela primaria? ¿Esta cantidad depende de la entrada? Justifique.

6) Dar el tiempo de ejecución en función de $n$ de los siguientes algoritmos y una $f(n)$ tal que el tiempo de ejecución pertenezca a $\Theta(f(n))$. Determine si cada algoritmo o partes del mismo tiene casos de análisis (peor, mejor, etc.).

a)

```
p ← 0
for i ← 1 to n do
  for j ← 1 to n^2 do
    for k ← 1 to n^3 do
      p ← p + 1
```

b)

```
p ← 0
for i ← 1 to n do
  for j ← 1 to i do
    for k ← 1 to n do
      p ← p + 1
```

7) Definir y analizar el tiempo de ejecución de la multiplicación de una matriz triangular inferior por una matriz completa (en la que todos sus elementos pueden ser diferentes de 0). ¿Qué formas tiene de hallar $t(n)$? ¿De cuántas formas podría encontrar la pertenencia a O() o a $\Theta$() del algoritmo?

8) ¿Encuentra algún inconveniente para analizar las iteraciones while y repeat como recurrencias?

9) Considerar las matrices $A, B, C \in \mathbb{R}^{(n \times n)}$, y la notación tal que $X_{i,j}$, con $1 \leq i, j \leq 2$ y $X$ cualquiera de las matrices $A, B$ o $C$, identifica una de las cuatro submatrices de orden $n/2$.

a) Dar el orden del tiempo de ejecución del algoritmo D&C que se describe con las ecuaciones

$$\begin{aligned} C_{1,1} &= A_{1,1} \times B_{1,1} + A_{1,2} \times B_{2,1} \\ C_{1,2} &= A_{1,1} \times B_{1,2} + A_{1,2} \times B_{2,2} \\ C_{2,1} &= A_{2,1} \times B_{1,1} + A_{2,2} \times B_{2,1} \\ C_{2,2} &= A_{2,1} \times B_{1,2} + A_{2,2} \times B_{2,2} \end{aligned}$$

¿Sería necesario definir algo más?

b) Buscar el algoritmo de Strassen, dar su definición en función de las submatrices $X_{i,j}$ anteriores y dar su orden de tiempo de ejecución.

c) Comparar los dos algoritmos de multiplicación de matrices. ¿Los dos son algoritmos D&C? ¿Alguno de los dos es "mejor" que el otro en cuanto a tiempo de ejecución?