# FUNDAMENTOS DE ANÁLISIS DE ALGORITMOS

GRADO EN INGENIERÍA INFORMÁTICA**.** La Rábida 10 de junio del 2013

| ALUMNO/A | | Nº HOJAS | | NOTA | |
|---|---|---|---|---|---|

- **Tiempo máximo: 120 minutos.**

| EJERCICIO 1 | PUNTOS: 1 |
|---|---|

Responder brevemente las siguientes cuestiones justificando las respuestas (0,5 puntos cada respuesta correcta).

**(a)** Dos algoritmos, A y B, resuelven un problema mediante las funciones TA(n)=100n y TB(n)=$2n^2$, respectivamente. ¿Cuál deberíamos usar? ¿Cuándo uno de ellos, y cuál, es el doble de rápido y, cuándo 20 veces más rápido?

**(b)** Usando la definición de notación asintótica $\Theta$ demostrar que $512n^2 + 5n \in \Theta(n^2)$.

| EJERCICIO 2 | PUNTOS: 1 |
|---|---|

Usando las definiciones de notación asintótica y corroborándolo con la regla del límite, demostrar si son verdaderas o falsas las siguientes afirmaciones: (0,25 ptos cada respuesta correcta)

**(a)** $(n+1)! \in O(3(n!))$

**(b)** $n^2 \in \Omega((n+1)^2)$

**(c)** $f(n) \in O(n) \Rightarrow 2^{f(n)} \in O(2^n)$

**(d)** $(n+1)! \in \Omega(n!)$

| EJERCICIO 3 | PUNTOS: 2 |
|---|---|

Estudiar la complejidad del algoritmo de ordenación por Selección modificado (de forma que se intercambien los elementos únicamente si son distintos) para el caso medio. Decidir si es rentable o no la modificación.

- El procedimiento Selección_Modificado puede ser implementado como sigue:

```
procedimiento  SelectionModif (a:vector; primero,ultimo: int);
    para i=primero hasta ultimo-1 hacer
            posmin = PosMinimo(a,i,ultimo);
            si a[i] ≠ a[posmin] entonces
                Intercambia(a, i, posmin);
            fsi;
    fpara
fprocedimiento SelectionModif
```

- En el algoritmo anterior se utiliza una función que calcula la posición del elemento mínimo de un subvector :

```
Int función PosMinimo (a:vector;primero,ultimo:int);
/* devuelve la posición del mínimo elemento de a[primero..ultimo] */
    pmin=primero;
    para i=primero+1 hasta ultimo hacer
            si a[i] < a[pmin] entonces
                    pmin = i
            fsi;
    fpara
    return pmin;
ffuncion PosMinimo;
```

- También se utiliza el procedimiento ***Intercambia*** para intercambiar dos elementos de un vector:

```
función Intercambia (a:vector ; i , j :int );
/* intercambia a[i]  con a[j]  */
    aux = a[i] ;
    a[i] = a[j] ;
    a[j] = aux;
ffuncion Intercambia;
```

| EJERCICIO 4 | PUNTOS: 2 |
|---|---|

Se realiza una variante de los números de Fibonacci que denominaremos "**Nacci**" cuya ecuación recurrente es:

$$\textbf{Nacci}(n) = \begin{cases} 1 & \text{Si } n = 1 \\ 3 & \text{Si } n = 2 \\ 3/2\ \textbf{Nacci}(n-1) + \textbf{Nacci}(n-2) & \text{En otro caso} \end{cases}$$

a) Escribe dos posibles implementaciones para el cálculo del n-ésimo número de **Nacci** usando:
   1. Un algoritmo de divide y vencerás y
   2. Un procedimiento que devuelva el resultado de forma directa, mediante una simple operación aritmética.

b) Obtener del orden de complejidad de los dos algoritmos del apartado anterior. Comparar los órdenes de complejidad obtenidos, estableciendo una relación de orden entre los mismos.

| EJERCICIO 5 | PUNTOS: 1 |
|---|---|

Para resolver cierto problema se dispone de un algoritmo trivial cuyo tiempo de ejecución t(n) (para problemas de tamaño n) es cuadrático ($t(n)\in\Theta(n^2)$). Se ha encontrado una estrategia que consigue una reducción del orden al aplicar divide y vencerás con 3 subproblemas de tamaño **n**/2, y tiempo de dividir y combinar en O(**n**).

1. Calcular la eficiencia para el algoritmo Divide y Vencerás
2. Diseñar otro algoritmo de divide y vencerás con un orden mejor desarrollando otra descomposición recursiva. Especificar por lo menos dos tipos de descomposiciones (es decir, el tamaño de los subproblemas y el número de estos) y el orden de complejidad que se obtendría con las mismas. Considerar que la división del problema y combinación son siempre O(**n**), y que no puede existir ninguna descomposición en **a** o menos subproblemas si el tamaño de estos es de **n/a**.

| EJERCICIO 6 | PUNTOS: 1 |
|---|---|

Resolver la siguiente ecuación de recurrencia:

$$2T(n)= 6T(n/2) + 4T(n/4) + 2n, \quad \text{con} \quad T(1), T(2)=1$$

| EJERCICIO 7 | PUNTOS: 2 |
|---|---|

**a)** Especificar el algoritmo de ordenación Mergesort.

**b)** Realizar la traza para ordenar el siguiente vector utilizando Mergesort $A = \{9, 1, 3, 5, 0, 4, 2, 6, 8, 7\}$.

## Fórmulas

$$\sum_{i=0}^{n-1} a_i = \left(\frac{(a_0 + a_{n-1})n}{2}\right)$$

$$\mathbf{t(n)} \in \begin{cases} O(n^{\log_b a}) & \text{Si } a > b^p \\ O(n^p \cdot \log n) & \text{Si } a = b^p \\ O(n^p) & \text{Si } a < b^p \end{cases}$$