FUNDAMENTOS DE ANÁLISIS DE ALGORITMOS

GRADO EN INGENIERÍA INFORMÁTICA. La Rábida 15 de septiembre de 2016. **Tiempo máximo: 120 minutos.**

ALUMNO/A		Nº HOJAS		NOTA	

EJERCICIO 1	PUNTOS: 2,5

- Usar las relaciones $\subset$ y **=** para ordenar los órdenes de complejidad, O, $\boldsymbol{\Omega}$, y $\boldsymbol{\Theta}$, de las siguientes funciones:

$n^2, n,\ n^3, n * \log n,\ (n+3)^2,\ n * \sqrt{n},\ T(n)$, siendo $\boldsymbol{T(n)}$ la función:

$$\boldsymbol{T(n)} = \begin{cases} \mathbf{1} & \boldsymbol{si\ n \leq 1} \\ \mathbf{4} * \boldsymbol{T}\left(\frac{\boldsymbol{n}}{\mathbf{3}}\right) + \boldsymbol{n} & \boldsymbol{si\ n} > 1 \end{cases}$$

NOTAS: $\log_3 4 = 1.26184$

$$X^{\log_a Y} = Y^{\log_a X}$$
$$\sqrt{n} = n^{\frac{1}{2}} = n^{0.5}$$

EJERCICIO 2	PUNTOS: 2,5

- El algoritmo de ordenación QuickSort puede ser implementado por:

```
QuickSort (A, p, r)  /* Ordena un vector A desde p hasta r */
        if (p < r) {
                q =Partition (A, p, r)
                /* El vector A[p..r] se particiona en dos subvectores A[p..q] y A[q+1..r], de forma que los elementos de A[p..q] son menores o iguales que los de A[q+1..r] (según elemento pivote) */
                QuickSort (A, p, q)
                QuickSort (A, q+1, r)
                }
```

- El algoritmo utiliza para su implementación la función **Partition**.

```
int función Partition (A:vector;, primero,ultimo:int)
    piv= A[primero]; i= primero-1;j= ultimo+1;
    mientras j ≥ i hacer
            mientras A[j] ≥ piv hacer
               j = j - 1;
            fmientras
            mientras A[i] ≤ piv hacer
               i = i + 1;
            fmientras
            si i < j entonces  /* A[i] ↔ A[j] */
                    temp: int; temp= A[j]; A[j]=A[i]; A[i]=temp;
            fsi
    fmientras
    return j; /* retorna el índice para la división (partición) */
ffuncion Partition;
```

Se pide:

a. (1 punto). Calcular la complejidad del algoritmo propuesto para el caso **mejor** por el método de la **ecuación característica**.
b. (0,5 puntos). Calcular la complejidad del algoritmo propuesto para el caso **mejor** por el Teorema maestro.
c. (1 punto). Calcular la complejidad del algoritmo propuesto para el caso **peor** por el método de la **ecuación característica** y compararla con la del caso mejor.

FUNDAMENTOS DE ANÁLISIS DE ALGORITMOS

GRADO EN INGENIERÍA INFORMÁTICA**.** La Rábida 15 de septiembre de 2016. **Tiempo máximo: 120 minutos.**

NOTA: El Teorema maestro aplicado a **T(n) = aT(n/b) + Θ (n^{k}long^{p}n)** es:

$$T(n) \in \begin{cases} O(n^{\log_b a}) & \text{si } a > b^k \\ O(n^k \cdot \log^{p+1} n) & \text{si } a = b^k \\ O(n^k \cdot \log^p n) & \text{si } a < b^k \end{cases}$$

EJERCICIO 3	PUNTOS: 3

Resolver las siguientes ecuaciones de recurrencia y calcular el orden temporal (1 punto cada apartado):

1. **(1 punto)**

```
function total(n:positivo)
      if n=1 then 1 else total(n-1) + 2 * parcial(n-1)
```

- siendo

```
function parcial (m:positivo)
      if m=1 then 1 else 2 * parcial(m-1)
```

2. **(1 punto)**

```
function total(n,m:positivo)
      if n=1 then m else    m + total (n-1, 2 * m)
```

3. **(1 punto)**

$$T(n) = \begin{cases} a & \text{si n=1} \\ 2T\left(\frac{n}{4}\right) + \lg n & \text{si n>1} \end{cases}$$

EJERCICIO 4	PUNTOS: 2

Tenemos que ejecutar un conjunto de n tareas, cada una de las cuales requiere un tiempo unitario. En un instante T=1, 2, ... podemos ejecutar únicamente una tarea. La tarea i produce unos beneficios b_i ($b_i > 0$) sólo en el caso en el que sea ejecutada en un instante anterior o igual a d_i.

- Diseñar un algoritmo **voraz** para resolver el problema aunque no se garantice la solución óptima que nos permita seleccionar el conjunto de tareas a realizar de forma que nos aseguremos que tenemos el **mayor beneficio posible**. Detallar :

a. (1,5 puntos) Las estructuras y/o variables necesarias para representar la información del problema y el método voraz utilizado (El procedimiento o función que implemente el algoritmo). Es necesario marcar en el seudocódigo propuesto a que corresponde cada parte en el esquema general de un algoritmo voraz. Si hay más de un criterio posible elegir uno razonadamente y discutir los otros. Indicar razonadamente el orden de dicho algoritmo.

b. (0,5 puntos) Aplicar el algoritmo implementado en el apartado anterior a la siguiente instancia:

i	1	2	3	4
b_i	50	10	15	30
d_i	2	1	2	1