FUNDAMENTOS DE ANÁLISIS DE ALGORITMOS

GRADO EN INGENIERÍA INFORMÁTICA. 18 de junio de 2018. **Tiempo máximo: 120 minutos.**

ALUMNO/A		Nº HOJAS		NOTA	

EJERCICIO 1 PUNTOS: 1

Demostrar por las definiciones de notación asintótica si son verdaderas o falsas las siguientes afirmaciones:

a) $n^2 \in O((n+1)^2)$ (0,25 ptos)

b) $512n^2 + 5n \in \Theta(n^2)$ (0,25 ptos)

c) $(n+1)! \in \Omega(4(n!))$ (0,25 ptos)

d) $(n+1)! \in O(n!)$ (0,25 ptos)

EJERCICIO 2 PUNTOS: 2

Dado el esquema del algoritmo de ordenación QuickSort:

```
QuickSort (A, izq, der)  /* Ordena un vector A desde izq hasta der */
      if (izq < der) {
         piv=mediana (izq, der);
         div =partition (A, piv, izq, der);
          /* El vector A[izq..der] se particiona en dos subvectores A[izq..div] y A[div+1..der], de forma que los
             elementos de A[izq..div] son menores o iguales que los de A[div+1..der] (según elemento pivote) */
         QuickSort (A, izq, div);
         QuickSort (A, div+1, der);
         }
```

Donde, con "***mediana***" se obtiene la mediana de los elementos del array A entre las posiciones izq y der (el elemento que ocuparía la posición central si estuvieran ordenados), y "partition" es el procedimiento de particionar pero usando ***piv*** como pivote, con lo que el problema se divide en dos subproblemas de igual tamaño. Si el tiempo de ejecución del procedimiento "***mediana***" es $t_{med}(n)=20n$, y el de "partition" es $t_{par}(n)=n$:

a) (0,75 puntos) Calcular la complejidad del algoritmo propuesto por el método de la **ecuación característica**.

b) (0,25 puntos) Calcular la complejidad del algoritmo propuesto por el Teorema maestro.

c) (0,5 puntos) Calcular la complejidad del algoritmo propuesto por expansión de recurrencia.

d) (0,5 puntos) Si el método de la **Burbuja** tiene un tiempo de ejecución de n^2, justificar para qué valores de la entrada es preferible esta versión del `QuickSort` al método de la `Burbuja`.

NOTAS:

- Suma de los valores de la progresión geométrica $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$
- El Teorema maestro aplicado a $T(n) = aT(n/b) + \Theta(n^k \log^p n)$ es:

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

FUNDAMENTOS DE ANÁLISIS DE ALGORITMOS

GRADO EN INGENIERÍA INFORMÁTICA**.** 18 de junio de 2018. **Tiempo máximo: 120 minutos.**

EJERCICIO 3	PUNTOS: 2

- Usar las relaciones $\subset$ y **=** para ordenar los órdenes de complejidad, **O**, **Ω**, y **Θ**, de las siguientes funciones:

$$n^2,\ logn,\ 2^n, n,\ log^2 n,\ 5*2^n+n^2, 17,\ \sqrt{n},\ \sqrt{n}log^2 n,\ \left(\frac{1}{3}\right)^n,\ \frac{n}{logn},\ \left(\frac{3}{2}\right)^n, loglogn,\ T(n)$$

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 4	PUNTOS: 3

Un número capicúa está formado por una cadena de $\boldsymbol{n}$ dígitos, $\boldsymbol{C}$: array[1.. $\boldsymbol{n}$] de $0..9$, donde cada $\boldsymbol{C(i) = C(n-i+1)}$. Si consideramos que los números de $\boldsymbol{n}$ dígitos pueden tener los 10 dígitos de 0 a 9 con la misma probabilidad en cada una de sus posiciones (incluido números con 0 al principio), Se pide:

a) (1,5 puntos). Un **algoritmo iterativo** llamado $\boldsymbol{CAPit}$ que determine si un número $\boldsymbol{C}$ es capicúa.

Calcular el tiempo de ejecución para $\boldsymbol{CAPit(C, 1, n)}$ en el caso peor y en el caso medio, suponiendo equiprobabilidad de todas las entradas y siendo {0,1,2,3,4,5,6,7,8,9} el alfabeto que forma los números.

b) (1,5 puntos).Un **algoritmo recursivo** llamado $\boldsymbol{CAPrc}$ que determine si un número $\boldsymbol{C}$ es capicúa.

Calcular el tiempo de ejecución para $\boldsymbol{CAPrc(C, 1, n)}$ en el caso peor y en el caso medio, suponiendo equiprobabilidad de todas las entradas y siendo $\{0,1,2,3,4,5,6,7,8,9\}$ el alfabeto que forma los números.

NOTA:

- Para calcular la eficiencia temporal considerar como operación característica el número de comparaciones entre componentes del número (por ej. $C(i) = C(n-i+1)$ ó $C(i) \neq C(j)$), siendo $n = j - i + 1$ el tamaño de la cadena).

EJERCICIO 5	PUNTOS: 2

Consideremos el problema de la mochila modificado en el que tenemos:

- **n** objetos, cada uno con un peso ($\mathbf{p_i}$) y un valor o beneficio ($\mathbf{b_i}$)
- Una mochila en la que podemos meter objetos, con una capacidad de peso máximo **M**.
- Cada objeto puede **meterse** dentro de la mochila, **no** meterse, o meterse la **mitad** del objeto obteniendo la mitad del beneficio.
- **Objetivo:** llenar la mochila con esos objetos, maximizando la suma de los beneficios (valores) transportados, y respetando la limitación de capacidad máxima **M**.
- Se supondrá que los objetos **se pueden partir en la mitad** ($x_i = \mathbf{0}$, $x_i = \frac{1}{2}$, $x_i = \mathbf{1}$).

a) (1 punto). Diseñar un algoritmo **voraz** para resolver el problema aunque no se garantice la solución óptima. Es necesario marcar en el código propuesto a que corresponde cada parte en el esquema general de un algoritmo voraz (criterio, candidatos, función……). Si hay más de un criterio posible elegir uno razonadamente y discutir los otros. Comprobar si el algoritmo garantiza la solución óptima en este caso (la demostración se puede hacer con un contraejemplo).

b) (0,5 puntos). Calcular su tiempo de ejecución.

c) (0,5 puntos). Aplicar el algoritmo para n = 2; M = 5; p = (8, 5); b = (10, 6)