FUNDAMENTOS DE ANÁLISIS DE ALGORITMOS

GRADO EN INGENIERÍA INFORMÁTICA. La Rábida 16 de **JUNIO** del 2011

ALUMNO/A:		Nº HOJAS:		NOTA:	

Instrucciones

- Leer bien antes de responder cada pregunta, **y sólo en las hojas que se entregan.**
- Escribir el nombre en cada hoja que se utilice para responder.
- Tiempo máximo: **120** minutos.

EJERCICIO 1	PUNTOS: 1

Responder Verdadero (V) o Falso (F), según corresponda a cada afirmación. Las respuestas Falsas deberán ser justificadas. Las respuestas falsas no justificadas serán consideradas incorrectas (0,25 puntos cada respuesta correcta).

(a) La única medida de eficiencia de un algoritmo es el tiempo de ejecución_____.

(b) En general, el tiempo de ejecución a priori de un algoritmo es exactamente igual al tiempo medido de manera empírica_________.

(c) En la definición de O, y Ω, la constante C que afecta a f(n) representa sólo la diferencia de las versiones de los compiladores a utilizar en los algoritmos____.

(d) En general un algoritmo tiene tres casos, mejor caso, peor caso y caso promedio______.

EJERCICIO 2	PUNTOS: 1

Indicar razonadamente la verdad o falsedad de las siguientes afirmaciones (0,25 ptos cada respuesta correcta):

(a) $2^{n+1} \in \Omega(2^n)$

(b) $(n+1)! \in \Omega(n!)$

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

(d) $3^n \in \Omega(2^n)$

EJERCICIO 3	PUNTOS: 2

Dado el siguiente algoritmo (1 punto cada respuesta correcta):

```
public long Suma SubsecuenciaMaxima (int vector[], int n)
{
int i, j, k;
/*1*/    int sumMax=0, posInicioSec=0, posFinSec=0, sumaActual;
/*2*/    for (i=0; i<n; i++)
/*3*/          for (j=i; j<n; j++)
/*4*/           {
/*5*/             sumaActual=0;
/*6*/             for (k=i; k<j; i++)
/*7*/                sumaActual = sumaActual + vector[k];
/*8*/              if  (sumaAcutal > sumaMax)
/*9*/               {
/*10*/                sumaMax= sumaActual;
/*11*/                posInicioSec= i;
/*12*/                posFinSec= j;
/*13*/              }
/*14*/          }
/*15*/   return sumMax;
}
```

(a) Calcular el orden de complejidad temporal del algoritmo para el peor caso indicando las reglas aplicadas de la función O para su cálculo.

(b) Corroborar el resultado anterior mediante conteo de operaciones elementales.

EJERCICIO 4	PUNTOS: 3

Resolver las siguientes ecuaciones de recurrencia (1 punto cada una):

(a) Determinar y resolver la ecuación de recurrencia para el siguiente algoritmo:

```
public void OrdenarVector (int vector[], int n)
{
int i, maxPos;
if (n> 1 )
  {
   maxPos=0;
   for (i=1;  i<n ;  i++)
       if (vector[i] > vector[maxPos])
         maxPos= i;
   if (maxPos != 0)
     {
      i= vector[0];
      vector[0]= vector[maxPos];
      vector[maxPos]= i;
     }
    OrdenarVector(vector, n-1);
  }
}
```

(b) $T(n)= 5\, T(n-1) - 8\, T(n-2) + 4\, T(n-3),\ n\geq 3,$ $T(0)=0,\ T(1)=1,\ T(2)=2$

(c) $T(n)= 4T(n/2) + n^2$

EJERCICIO 5	PUNTOS: 1

Escribir un algoritmo voraz para entregar billetes en un cajero automático que suministra la cantidad de billetes solicitada de forma que el número total de billetes sea mínimo. Se supone que el cajero dispone de suficientes billetes de todas las cantidades consideradas. Explicar el funcionamiento del algoritmo: cuál es el conjunto de candidatos, la función de selección, la función para añadir un elemento a la solución, el criterio de finalización, el criterio de coste, etc. Comprobar que es correcto para billetes de 10, 20 y 50 €, y no lo es si no existen billetes de 10 €.

EJERCICIO 6	PUNTOS: 2

Ordenar el siguiente vector utilizando Mergesort y Quicksort : $A = \{3, 41, 52, 26, 38, 57, 9, 49\}$.

Fórmulas

$$\sum_{i=0}^{n-1} a_i = ((a_i+a_{n-1})n)/2$$

$$\sum_{i=0}^{n-1} a_i = \sum_{i=0}^{n-1} a_0\, \Pi^i = a_0 \sum_{i=0}^{n-1} \Pi^i = a_0(\Pi^n-1)/(\Pi-1)$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$