FUNDAMENTOS DE ANÁLISIS DE ALGORITMOS

GRADO EN INGENIERÍA INFORMÁTICA. La Rábida 20 de junio del 2012

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) Sabiendo que los órdenes de eficiencia son clases de equivalencia y, dadas las funciones T1(n)= $2n^3 + 4n^2 + 5$ y T2(n)= $n^3 - 4n$,
¿Pertenecen a la misma clase?, en caso afirmativo ¿cuál es la función representante?

EJERCICIO 2	PUNTOS: 1

(0,25 ptos cada respuesta correcta):

(a) Demostrar que si $\mathbf{f(n) \in O(n)}$ entonces $\mathbf{(f(n))^2 \in O(n^2)}$. ¿Se puede generalizar el resultado? En caso afirmativo, enunciarlo.

(b) Indicar razonadamente la verdad o falsedad de las siguiente afirmación $\boldsymbol{f(n) \in O(n) \Rightarrow 2^{f(n)} \in O(2^n)}$

¿Cuáles de las siguientes afirmaciones son verdaderas? Demostrar las respuestas usando la definición de O(f(n)).

(c) $\mathbf{\log n \in O(n)}$

(d) $\mathbf{n \in O(\log n)}$

EJERCICIO 3	PUNTOS: 1

Escribir un algoritmo voraz para planificar un evento cultural que consiste en ***n*** conferencias que se celebrarán en 10 aulas diferentes. Cada conferencia se celebra una sola vez y se conoce el aula donde se celebra, la hora de comienzo y su duración. El único objetivo es asistir al máximo número posible de conferencias.

EJERCICIO 4	PUNTOS: 2

Ordenar el siguiente vector utilizando Mergesort y Quicksort : $A = \{9, 1, 3, 5, 0, 4, 2, 6, 8, 7\}$.

EJERCICIO 5	PUNTOS: 3

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

(a) $T(n)= 5\,T(n-1) + 6\,T(n-2) + 4 * 3^n$, $n \geq 2$, $T(0)=0$, $T(1)=36$

(b) $T(n)= 2T(n/2) + n\log n$

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

```
int funcion ternarysearch(v:vector;primero,ultimo,clave:int)
  tercio ← 0, dostercios ← 0;
  si primero = ultimo  entonces
       si clave = v[primero] entonces
          devolver primero;
       sino
          devolver -1;
       fsi
  fsi
  si ultimo-primero = 1 entonces
   si clave = v[primero]
          devolver primero;
   sino
          si clave = v[ultimo] entonces
                devolver ultimo;
          sino
                devolver -1;
          fsi
    fsi
  fsi
  tercio ←  ((ultimo - primero + 1) / 3) + primero;
  dostercios ←  (ultimo -tercio) + primero;
  si clave = v[tercio] entonces
   devolver tercio;
  sino
       si clave < v[tercio] entonces
          devolver ternarysearch (v, primero, tercio-1, clave);
       sino
          si clave = v[dostercios] entonces
                devolver dostercios;
          sino
                si clave < v[dostercios] entonces
                      devolver ternarysearch (v, tercio+1, dostercios-1, clave);
                sino
                      devolver ternarysearch (v, dostercios+1, ultimo, clave);
                fsi
          fsi
       fsi
  fsi
ffuncion ternarySearch
```

FUNDAMENTOS DE ANÁLISIS DE ALGORITMOS

GRADO EN INGENIERÍA INFORMÁTICA. La Rábida 20 de junio del 2012

EJERCICIO 6	PUNTOS: 2

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

```
procedimiento  InsercionBinaria(a:vector; primero,ultimo: int);
        para i:=primero+1 hasta ultimo hacer
                x:=a[i]; k:=Posicion(a,primero,i-1,x);
                para j:=i-1 hasta k+1 inc -1 hacer
                        a[j+1]:=a[j]
                fpara;
                a[k]:=x
        fpara
fprocedimiento InsercionBinaria;


Int función Posicion(a:vector;primero,ultimo,clave:int);
        mientras (primero<=ultimo) hacer
                mitad:=(primero+ultimo) / 2;
                si clave=a[mitad] entonces
                        return mitad
                sino
                        si clave<a[mitad] entonces
                                ultimo:=mitad-1
                        sino
                                primero:=mitad+1
                        fsi
                fsi;
        fmientras
        return mitad
ffuncion Posicion;
```

(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.

Fórmulas

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