Facultad de Informática

Informática de Gestión

Estructuras de Datos y de la Información

Examen Final (Primer Parcial), Junio 2002

Ejercicio 1 (2.5 puntos) Se sabe que el tiempo $T(n)$ que consume en el caso peor un algoritmo recursivo obedece a la siguiente recurrencia:

$$\begin{array}{lcl} T(1) & = & 1 \\ T(n) & = & 2 * T(n/2) + n * \log_2(n) \text{ si } n > 1 \end{array}$$

Calcula en forma explícita una función $f(n)$ tal que $T(n) \in O(f(n))$, suponiendo que n es una potencia de 2, es decir $n = 2^m$ para un cierto $m \geq 0$.

NOTA: En el siguiente ejercicio se supondrá definido un tipo *vect* de vectores de enteros de la siguiente manera: *tipo vect* = **vector** [1..1000] **de** *entero*

Ejercicio 2 (5 puntos) Dado un vector de enteros y un entero n, la acción de *positivizar* el vector consiste en sustituir en las n primeras posiciones del vector todas las apariciones de números negativos (< 0) por 0, dejando sin modificar el resto de las n posiciones (las que contienen números ≥ 0).

(a) [2 puntos] Especificar un procedimiento *positivizar* que positivice (las n primeras posiciones de) un vector v de enteros.

(b) [3 puntos] Se ha escrito el siguiente fragmento de código como cuerpo del procedimiento *positivizar*:

$j := 0$;
mientras $j < n$ **hacer**
  si $v[j] < 0$ **hacer** $v := asig(v, j, 0)$ **fsi** ;
  $j := j + 1$
fmientras

Verificar si es correcto respecto a la especificación proporcionada. Si encuentras algún error corrígelo y verifica la versión corregida.

Recordar que $asig(v, i, e)$ es el vector que resulta de asignar el valor de e a la posición i del vector v:

$$asig(v, i, e)[j] = \begin{cases} e \text{ si } j = i \\ v[j] \text{ si } j \neq i \end{cases}$$

Ejercicio 3 (2.5 puntos) Se desea definir una función que calcule, en forma codificada, el número de veces que aparecen las cifras 0 y 1 en un número binario x dado. Si m es el número de veces que aparece la cifra 0 en x y n es el número de veces que aparece la cifra 1 en x, entonces la función devolverá $2^m * 3^n$.

Escribe una función recursiva no final *CeroUno*, que tome un número natural x formado exclusivamente por ceros y unos, y devuelva la codificación anteriormente descrita.

Por ejemplo, si $x = 10100$, entonces $m = 3$, $n = 2$ y la función devuelve $2^3 * 3^2 = 72$.