Examen Final Febrero2014.pdf

Anthorio

Análisis y diseño de algoritmos

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

E. T. S. de Ingeniería Informática — Curso: 2013–14

Análisis y Diseño de Algoritmos — Examen final: 30 de Enero

Alumno: _____

Grado/Grupo: _____

Especificación y Verificación

1.1 Especifica:

a) **[1p]** una función $numVeces$ que dado un array a y elemento v de su tipo base, determine el número de veces que v aparece en a:
int numVeces($Tipo[]\ a, Tipo\ v$)

b) **[2p]** una función $esPermutacion$ que compruebe si dos arrays del mismo tipo y de la misma longitud contienen los mismos elementos permutados:

boolean esPermutacion ($Tipo[]\ a, Tipo[]\ b$)

Nota: en los arrays puede haber elementos repetidos.

1.2 **[7p]** Verificar formalmente el siguiente código:

$\{Q \equiv (k \geq 0) \wedge k\ es\ par\}$
`int` $x = 1$;
`int` $y = 2^k$;
`while` ($x < y$) {
    $x = x * 2$;
    $y = y/2$;
}
$\{R \equiv x = 2^{k/2}\}$

Complejidad, Divide y Vencerás

2.1 **[1.5p]** Resolver la siguiente ecuación en recurrencia:

$$T(n) = \begin{cases} 0 & n = 0 \\ 2T(n-1) + n + 2^n & n > 0 \end{cases}$$

2.2 **[7p]** Dado un número natural n, diseña un algoritmo basado en la técnica *Divide y Vencerás*, de complejidad $\mathcal{O}(log\ n)$, que compruebe si n es el producto de tres números naturales consecutivos. Por ejemplo, $210 = 5 * 6 * 7$, mientras que 12 no puede escribirse como producto de tres naturales consecutivos.

2.3 **[1.5p]** Para el algoritmo diseñado, escribe la ecuación de recurrencia y resuélvela mediante el Teorema Maestro (que debe dar como resultado complejidad logarítmica).

Programación dinámica y algoritmos voraces

Dada una secuencia $L_1, \cdots, L_n$ de n listas **ordenadas** con $l_1, \cdots, l_n$ elementos, respectivamente, se pretende mezclarlas **por pares** hasta obtener una única lista ordenada realizando **el número mínimo de movimientos de datos**. Teniendo en cuenta que para mezclar de forma ordenada dos listas ordenadas L y L' de longitudes l y l' hay que hacer $l + l'$ movimientos de datos, la forma en la que se **asocian** las listas para mezclarse determina la eficiencia del proceso.
Por ejemplo, supón que hay tres listas L_1, L_2 y L_3 con 30, 20 y 10 elementos respectivamente. Suponiendo que la mezcla ordenada de dos listas L, L' se representa como (LL'), la mezcla de las listas L_1, L_2 y L_3 puede puede realizarse de dos formas distintas:

1. $((L_1L_2)L_3)$, es decir, primero mezclamos L_1 con L_2 (50 movimientos) y, a continuación, mezclamos el resultado con L_3 (60 movimientos), lo que da un total de 110 movimientos
2. $(L1(L_2L_3))$, es decir, primero mezclamos L_2 con L_3 (30 movimientos) y, a continuación, mezclamos L_1 con el resultado (60 movimientos), lo que da un total de 90 movimientos

Nota: Observa que en el proceso de mezcla, el orden de las listas no se modifica. Lo relevante es cómo se colocan los paréntesis.
Utilizando programación dinámica,

1. **[5p]** encuentra la ecuación de recurrencia para el número mínimo de movimientos
2. **[5p]** implementa un algoritmo de programación dinámica bottom-up que implemente la recurrencia del apartado anterior.

Bactracking y Ramificación y Poda

Supón te dedicas a organizar bodas. La pareja cuya boda estás organizando en este momento te da la lista con sus n invitados. Además de la lista, la pareja también te proporciona información sobre cómo es la relación entre los distintos invitados. Esta información puede representarse mediante un grafo $G(V, E)$ en el que cada vértice es un invitado, y una arista $(u, v) \in E$ significa que u y v se llevan muy mal. Tu trabajo consiste en distribuir los invitados en mesas (de un máximo K de comensales, que no tienen que estar completamente ocupadas) de forma que ningún par de invitados que se llevan mal compartan mesa.

1. **[6p]** Dado un número máximo de mesas M, diseña, utilizando la técnica de vuelta atrás, un algoritmo que encuentre una distribución correcta de invitados en las mesas, o devuelva una indicación de que no existe ninguna distribución.
2. **[4p]** Modifica el algoritmo anterior para que se encuentre la distribución que utiliza el menor número de mesas.