Diseño de Algoritmos

Unidad 3: Algoritmos de «divide y vencerás»

Versión 2.1

1. Sea el siguiente algoritmo, que calcula la parte entera por defecto del logaritmo de $n \in \mathbb{N}$ en base $b \geq 2$:

$$
\begin{array}{l}
log(n, b) \rightarrow r \\
\text{si } n < b \\
\quad r \leftarrow 0 \\
\text{si no} \\
\quad r \leftarrow log(n \operatorname{div} b, b) + 1
\end{array}
$$

a) Identifique los elementos que lo determinan como perteneciente al esquema de «divide y vencerás». ¿Con qué subesquema de este se identifica?

b) Analice mediante ecuaciones de recurrencia el número de operaciones aritméticas que realiza cuando n es potencia de b.

c) El número exacto de operaciones aritméticas se obtiene inmediatamente si se presupone que el algoritmo es correcto. ¿Por qué?

2. Análogamente a la búsqueda binaria, puede desarrollarse un algoritmo de *búsqueda ternaria*, fijando el umbral al menos en $n_0 = 2$ y reduciendo sucesivamente la búsqueda en un vector de tamaño $n > n_0$ a otra en un subvector cuyo tamaño sea aproximadamente de un tercio de n.

a) Desarrolle esta idea escribiendo el algoritmo apropiado.

b) Haga un análisis completo cuando n es potencia de tres.

c) Suponga que los resultados del análisis son asintóticamente ciertos para todo n. Compárelos con los que se obtienen para la búsqueda binaria.

3. Dada una función $f : \mathbb{R} \rightarrow \mathbb{R}$, un intervalo $[a, b] \in \mathbb{R}^2$ y una precisión $p \in \mathbb{R}^+$:

a) Diseñe un algoritmo análogo a la búsqueda binaria que calcule un intervalo, cuya longitud no exceda a p, que contenga una raíz de f.

b) Analice el número de evaluaciones de la función f que realiza el algoritmo en función de $l = b - a$ y p. Para simplificar puede suponer que $l = p \cdot 2^k$, $k \in \mathbb{N}$.

Para ello, la función f se supondrá continua y estrictamente monótona en $[a, b]$ y tal que $f(a) \cdot f(b) < 0$. Bajo estas condiciones, el teorema de Bolzano garantiza la existencia de, exactamente, una raíz de la función en el intervalo abierto (a, b).

4. Sea $[v_1, \ldots, v_n] \in \mathbb{Z}^n$ tal que $v_1 < \cdots < v_n$.

a) Demuestre por inducción sobre n que:

$$\forall i, j \in [1, n]\ (i < j \implies v_i - i \leq v_j - j)$$

b) Empleando la propiedad anterior, diseñe un algoritmo que permita encontrar un índice $p \in [1, n]$ tal que $v_p = p$, si existe un índice tal, o bien devuelva 0 en caso de que no exista. El algoritmo deberá realizar en el peor de los casos $\Theta(\log n)$ operaciones.

5. En ciertos criptosistemas es necesario calcular números naturales de la forma a^n mód p. El siguiente algoritmo lo hace:

$$\begin{array}{l} potmod(a, n, p) \to r \\ r \leftarrow 1 \\ \text{desde } i \leftarrow 1 \text{ hasta } n \\ \qquad r \leftarrow (r \cdot a) \text{ mód } p \end{array}$$

A la operación $(r \cdot a)$ mód p se la denomina, por razones obvias, «producto modular». En estos criptosistemas a y n son menores que p, un número primo de gran tamaño. Además, es crucial que el algoritmo que realiza tal cálculo sea eficiente y, en concreto, que todo producto modular tenga siempre sus dos primeros operandos menores que p. En particular, un número lineal de productos modulares como el que efectúa el algoritmo anterior es inaceptable.

Teniendo en cuenta todo lo anterior:

a) Diseñe un nuevo algoritmo que suponga una mejora sustancial. Cuide especialmente de que ningún producto modular emplee operandos superiores a p. Obtenga una cota superior del número de productos modulares que realiza en función de n.

b) Estime el tiempo máximo de ejecución de una implementación en la que p se almacena empleando 256 bits y un producto modular de números de este tamaño se ejecuta en 0,1 µs. Compárelo con el que se obtendría, bajo las mismas condiciones, con una implementación del primer algoritmo.

6. Se desea calcular el producto escalar de dos vectores:

$$\vec{u} \cdot \vec{v} = \sum_{i=1}^{n} u_i \cdot v_i$$

a) Diseñe un algoritmo que emplee la técnica de equilibrado.

b) Demuestre su corrección.

c) Analícelo y compare el resultado obtenido con el del algoritmo iterativo trivial.

7. El siguiente algoritmo permite «invertir» un vector:

```
invierte(v, i, j) → v
n ← j − i + 1
si n > 1
   k ← i − 1 + n div 2
   invierte(v, i, k)
   invierte(v, i + j − k, j)
   desde l ← i hasta k
         v[l] ↔ v[l + j − k]
```

a) Identifique los elementos que lo determinan como perteneciente al esquema de «divide y vencerás». ¿Con qué subesquema de este se identifica?

b) Analice la eficiencia temporal del algoritmo. Para ello, especifique claramente qué operación elemental tomará como patrón y suponga que n es potencia de dos.

c) Obtenga otro algoritmo que sea más simple que este y además mejore sustancialmente su eficiencia.

8. Analice el número exacto de comparaciones del algoritmo de *ordenación por fusión* cuando $n = n_0 \cdot 2^k$ y se emplea como algoritmo base por debajo del umbral, n_0, el método de intercambio directo. Calcule el *umbral óptimo teórico* bajo estas condiciones.

9. Sean A y B dos matrices cuadradas de dimensión n, siendo n una potencia de dos. Una forma de obtener el producto consiste en descomponer cada una de las dos matrices en cuatro submatrices cuadradas de dimensión $n/2$:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

El problema se reduce así, a calcular 8 productos y 4 adiciones de matrices cuadradas de dimensión $n/2$. Para calcular estos productos puede volverse a utilizar este mismo método, y así sucesivamente hasta obtener matrices de, por ejemplo, dimensión 2, que pueden multiplicarse directamente con operaciones escalares.

a) Demuestre que el número de operaciones escalares que realiza este algoritmo no es distinto del que se obtiene con el algoritmo clásico de multiplicación de matrices.

b) En 1969, *Strassen* demostró que tras cada descomposición sólo son necesarios 7 productos en vez de 8. En concreto demostró que bastan 7 productos y 18 operaciones entre adiciones y sustracciones. El algoritmo resultante lleva su nombre; compare el número de operaciones escalares que realiza con el de los anteriores.

10. Del análisis del algoritmo de *Strassen* de multiplicación de matrices que emplea como algoritmo base el método clásico, se obtiene la siguiente ecuación de recurrencia:

$$t(n) = \begin{cases} n^2(2n-1), & n \leq n_0 \\ 7t\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2, & n > n_0 \end{cases}$$

Esta expresa el número de operaciones escalares que realiza el algoritmo para multiplicar dos matrices $n \times n$, con $n = n_0 \cdot 2^k$.

Calcule su valor exacto en función de n_0 y su orden asintótico, sujetos a las condiciones indicadas. ¿Depende el orden del valor del umbral?

11. Para el cálculo de los términos de la solución de una *ecuación de recurrencia lineal y homogénea* se puede representar la ecuación de la siguiente manera:

$$\forall n \geq k \; f_n = a_k f_{n-1} + \cdots + a_2 f_{n-k+1} + a_1 f_{n-k}$$
$$f_0 = c_1, \ldots, f_{k-2} = c_{k-1}, f_{k-1} = c_k$$

donde $a_1, \ldots, a_{k-1}, a_k$ y $c_1, \ldots, c_{k-1}, c_k$ son elementos conocidos de un anillo $\mathscr{A} = \langle A, +, \cdot, -, 0, 1 \rangle$. Esto mismo puede representarse en forma matricial:

$$\forall n \geq k \begin{bmatrix} f_n \\ f_{n-1} \\ \vdots \\ f_{n-k+1} \end{bmatrix} = \begin{bmatrix} a_k & \cdots & a_2 & a_1 \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} f_{n-1} \\ f_{n-2} \\ \vdots \\ f_{n-k} \end{bmatrix}$$
$$\begin{bmatrix} f_{k-1} \\ f_{k-2} \\ \vdots \\ f_0 \end{bmatrix} = \begin{bmatrix} c_k \\ c_{k-1} \\ \vdots \\ c_1 \end{bmatrix}$$

o, más concisamente:

$$\forall n \geq k \; \vec{x}_n = A \cdot \vec{x}_{n-1}$$
$$\vec{x}_{k-1} = \vec{c}$$

a) Diseñe un algoritmo que reciba A, $\vec{c}$, k y n, y calcule f_n en un número de operaciones de $\mathscr{A}$ perteneciente, en el peor de los casos, a $\Theta(\log n)$.

b) Partiendo de esta idea diseñe un algoritmo específico para la sucesión de *Fibonacci*. Compárelo con el obtenido a partir de la propia definición recursiva de la sucesión.

12. Dada la siguiente versión del algoritmo de ordenación rápida, realice el análisis detallado del *número de intercambios* que se producen en el caso promedio. Para ello, considere las hipótesis simplificadoras que estime convenientes.

```
ordenación-rápida(v, i, j) → v
si i < j
   p ← pivote(v, i, j)
   ordenación-rápida(v, i, p − 1)
   ordenación-rápida(v, p + 1, j)
```

```
pivote(v, i, j) → (v, p)
(p, x) ← (i, v[i])
desde k ← i + 1 hasta j
      si v[k] ≤ x
         p ← p + 1
         v[p] ↔ v[k]
v[i] ↔ v[p]
```

13. Analice el consumo de espacio, en el peor caso, del algoritmo del ejercicio anterior. Proponga una mejora significativa.

14. Dados n números enteros, $x_1, \ldots, x_n$, se desea encontrar el único *polinomio mónico* de grado n que interpola a la *función nula* en dichos valores. Para ello se dispone de:

a) Un algoritmo que permite multiplicar dos polinomios de coeficientes enteros de grado k en $O(k \log k)$ operaciones elementales.

b) Un algoritmo que permite multiplicar un polinomio de grado k por otro de grado 1, ambos de coeficientes enteros, en $O(k)$ operaciones elementales.

Diseñe un algoritmo que resuelva este problema en un número de operaciones elementales perteneciente en el peor de los casos a $O(n \log^2 n)$.

15. Sea un algoritmo de «divide y vencerás» cuya eficiencia temporal viene definida por la siguiente ecuación de recurrencia:

$$t(n) = \begin{cases} d, & n = n_0 \\ a\,t\left(\frac{n}{b}\right) + cn^k, & n > n_0 \end{cases}$$

donde $a, c, d \in \mathbb{R}^+$, $n_0 \in \mathbb{N}^*$ y $b, k \in \mathbb{N}$, con $b \geq 2$ y $n = n_0 \cdot b^l$.

Bajo estas condiciones calcule:

a) La forma explícita de $t(n)$.

b) Su orden.

Tenga en cuenta que se pueden presentar distintas soluciones en función de los valores de los parámetros.

16. Analice algunos de los algoritmos de «divide y vencerás» que aparecen en los ejercicios anteriores a la luz del resultado del ejercicio anterior.