Algorítmica

Grado en Ingeniería Informática, Grupo E Curso 2011–2012

Problemas de análisis de la eficiencia

1. Demostrar las siguientes afirmaciones:
   a) $\forall\, k > 0,\ k \cdot f \in \mathrm{O}(f)$,
   b) Si $f \in \mathrm{O}(g)$ y $h \in \mathrm{O}(g)$ entonces $(f + h) \in \mathrm{O}(g)$ y, en particular, si $f \in \mathrm{O}(g)$ entonces $(f + g) \in \mathrm{O}(g)$,
   c) Si $f \in \mathrm{O}(g)$ y $g \in \mathrm{O}(h)$ entonces $f \in \mathrm{O}(h)$,
   d) $n^r \in \mathrm{O}(n^5)$ si $0 \leq r \leq 5$,
   e) $n^k \in \mathrm{O}(b^n),\ \forall\, b > 1$ y $k \geq 0$,
   f) $\log_b n \in \mathrm{O}(n^k),\ \forall\, b > 1, k > 0$,
   g) $\mathrm{máx}(n^3, 10n^2) \in \mathrm{O}(n^3)$,
   h) $\sum_{i=1}^{n} i^k \in \Theta(n^{k+1}),\ \forall\, k \in N$.
2. Considerar la siguientes funciones de n:
   i) $f_1(n) = n^2$
   ii) $f_2(n) = n^2 + 1000n$
   iii) $f_3(n) = \begin{cases} n, & \text{si } n \text{ es impar} \\ n^3, & \text{si } n \text{ es par} \end{cases}$
   iv) $f_4(n) = \begin{cases} n, & \text{si } n \leq 100 \\ n^3, & \text{si } n > 100 \end{cases}$

   Indicar para cada par distinto (i, j), si $f_i(n)$ es $\mathrm{O}(f_j(n))$ y si $f_i(n)$ es $\Omega(f_j(n)$.
3. Probar que las siguientes afirmaciones son ciertas:
   a) 17 es $\mathrm{O}(1)$.
   b) $n(n-1)/2$ es $\mathrm{O}(n^2)$.
   c) $\mathrm{máx}(n^3, 10n^2)$ es $\mathrm{O}(n^3)$.
4. Demostrar que si $f(n)$ es $\mathrm{O}(n^a)$ y $g(n)$ es $\mathrm{O}(n^b)$, entonces:
   a) $f(n)g(n)$ es $\mathrm{O}(n^{a+b})$
   b) $f(n) + g(n)$ es $\mathrm{O}(n^{\mathrm{máx}(a,b)})$

5. Existe un dicho popular entre los programadores veteranos: «la eficiencia es menos importante que la corrección de un programa», es decir, la velocidad con que se ejecute un programa es irrelevante si las respuestas que proporciona no son correctas. ¿Verdadero o falso?

6. Encontrar el entero k más pequeño tal que $f(n)$ es $\mathrm{O}(n^k)$ en los siguientes casos:
   a) $f(n) = 13n^2 + 4n - 73$
   b) $f(n) = 1/(n+1)$
   c) $f(n) = 1/(n-1)$
   d) $f(n) = (n-1)^3$
   e) $f(n) = (n^3 + 2n - 1)/(n+1)$
   f) $f(n) = \sqrt{n^2 - 1}$

7. Usando la notación $\mathrm{O}(\cdot)$, dar el peor caso de tiempo de ejecución de las siguientes funciones, considerando que el tamaño de las entradas viene dado por el valor de `n`:

```
1:   void ejemplo1(int n)
2:   {
3:     for (int i = 0; i < n; i++)
4:       for (int j = 0; j < n; j++)
5:         {
6:           C[i][j] = 0;
7:           for (int k = 0; k < n; k++)
8:            C[i][j] += A[j][k] * B[k][j];
9:         }
10:  }
```

```
1:   void ejemplo2(int n)
2:   {
3:     for (int i = 1; i < n; i++)
4:       for (int j = i+1; j <= n; j++)
5:         for (int k = 1; k <= j; k++)
6:           Global += k*i;
7:   }
```

```
1:   void ejemplo3(int n)
2:   {
3:     int x = 0;
4:     int y = 0;
5:     for (int i = 1; i <= n; i++)
```

```
6:        if (int i % 2 == 1)
7:          {
8:            for (int j = i; j <= n; j++)
9:              x++;
10:           for (int j = 0; j < i; j++)
11:             y++;
12:         }
13:   }
```

8. La definición tradicional de x^n, donde $x \in \mathbb{R}$ y $n \in \mathbb{N}$ proporciona un algoritmo lineal para calcular este valor. Proponga un algoritmo cuyo orden de eficiencia sea asintóticamente mejor y calcule su orden. Para facilitar la resolución, suponga primero que n es una potencia positiva de 2. Extienda después el algoritmo para cualquier valor entero positivo de n. (*Indicación:* El algoritmo debe ser $\mathrm{O}(\log(n))$).

9. Resolver las siguientes ecuaciones de recurrencia:

   a) $T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$

   b) $T(n) = 2 \cdot T(\sqrt{n}) + \log n,\ n \geq 4,\ T(2) = 1$

   c) $T(n) = 2 \cdot T(n/2) + \log n,\ n \geq 2,\ T(1) = 1$

   d) $T(n) = 5 \cdot T(n/2) + (n \cdot \log n)^2,\ n \geq 2,\ T(1) = 1$

10. Expresar, en notación $\mathrm{O}(\cdot)$, el orden que tendría un algoritmo cuyo tiempo de ejecución fuera $T(n)$:

    a) Si $T(n) = \log n!$

    b) Si $T(n) = n!$

11. Estimar el peor caso de tiempo de ejecución para la siguiente función. Emplear la notación $\mathrm{O}(\cdot)$.

```
1:   int recursiva(int n)
2:   {
3:     if (n <= 1)
4:       return 1;
5:     else
6:       return (recursiva(n - 1) + recursiva(n - 1));
7:   }
```

12. Dada la siguiente función:

```
1:   int E(int n)
2:   {
3:     if (n == 1)
4:       return n;
5:     else
6:       return (E(n/2) + 1);
7:   }
```

a) ¿Cuál es el valor que devuelve la función `E`?

b) Obtener una expresión para el peor caso de tiempo de ejecución de la función `E`.

13. Ordene las siguientes funciones de mayor a menor, según la rapidez con que crecen: 2^n, n, n^3, 4^n, $n/\log(n)$, $n^3/(n^2+1)$, $n!$, $\sqrt{n^3}$, n^n, 2^{2^n}, $\sin n$,

14. La siguiente función, `max(i, n)`, devuelve el mayor de entre los elementos del array `A` situados entre las posiciones `i` e `(i + n - 1)`. Por comodidad, se puede suponer que `n` es una potencia de 2.

```
1:   int max(int i, int n)
2:   {
3:     int m1, m2;
4:
5:     if (n == 0)
6:       return A[i];
7:     else
8:       {
9:         m1 = max(i, n/2);
10:        m2 = max(i + n/2, n/2);
11:        return (m1 < m2 ? m1 : m2);
12:      }
13:  }
```

Estudiar el peor caso de tiempo de ejecución en función de `n`, es decir, del número de elementos de cuales el mayor es encontrado.

15. Sean a y b enteros positivos. Suponga una función Q definida recursiva como sigue:

$$Q(a,b) = \begin{cases} 0, & \text{si } a < b \\ Q(a-b,b)+1, & \text{su } a \geq b \end{cases}$$

a) Encontrar el valor de $Q(2,3)$ y de $Q(14,3)$.

b) ¿Qué hace esta función? Encontrar $Q(5861,7)$.

16. Sea n un entero positivo. Suponga una función L definida recursivamente como sigue:

$$L(n) = \begin{cases} 0, & \text{si } n = 1 \\ L(\lceil \frac{n}{2} \rceil) + 1, & \text{si } n > 1 \end{cases}$$

    a) Encontrar $L(25)$.

    b) ¿Qué calcula esta función?

17. La búsqueda secuencial y la búsqueda binaria plantean un tiempo de equilibrio entre el tiempo de búsqueda y el de preprocesamiento. ¿Cuántas búsquedas binarias habría que hacer en un vector de n elementos para compensar el tiempo invertido en la ordenación?

18. Sean a y b enteros no negativos. Sea la función MCD, definida como sigue:

$$\mathrm{MCD}(a,b) = \begin{cases} \mathrm{MCD}(b,a), & \text{si } a < b \\ a, & \text{si } b = 0 \\ \mathrm{MCD}(b, \quad \mathrm{mód}\ (a,b)), & \text{en otro caso} \end{cases}$$

    a) Encontrar $\mathrm{MCD}(6,15)$, $\mathrm{MCD}(20,28)$ y $\mathrm{MCD}(540,168)$.

    b) ¿Qué hace esta función?

19. Construir una función que sume los elementos de un array de números enteros recursivamente.

20. Utilizando una función recursiva, calcular:

$$\binom{n}{m} = \begin{cases} 1, & \text{si } n = 1 \text{ ó } m = n \\ \binom{n-1}{m-1} + \binom{n-1}{m}, & \text{en otro caso} \end{cases}$$

    Se supone que $0 \leq m \leq n$ y $n \geq 1$.

    ¿Cuál es la eficiencia de esta función? ¿Podrías hacerlo mejor?

21. Resolver las fórmulas de recurrencia siguientes:

    a) $T(n) = T(n-1) + n$, $n \geq 2$ y $T(1) = 1$.

    b) $T(n) = T(n/2) + 1$, $n \geq 2$, $T(1) = 0$ y n potencia de 2.

    c) $T(n) = T(n/2) + n$, $n \geq 2$, $T(1) = 0$ y n potencia de 2.

    d) $T(n) = 2T(n/2) + n$, $n \geq 2$, $T(1) = 0$ y n potencia de 2.

    e) $T(n) = 2T(n/2) + 1$, $n \geq 2$, $T(1) = 0$ y n potencia de 2.

    f) $T(n) = 2T(n/2) + n^2$, $n \geq 2$, $T(1) = 0$ y n potencia de 2.

22. Determinar el orden de ejecución de un algoritmo cuyo tiempo de ejecución es:

$$T(n) = T(n/2) \cdot T^2(n/4)$$

23. Si $T(n)$ es el tiempo de ejecución de cierto algoritmo, determinar su orden a partir de la siguiente recurrencia:

$$T(n) = 3 \cdot T(n/2) - 2 \cdot T(n/4) + \log n \quad ; \quad T(2) = 3,\, T(1) = 3$$

24. Demostrar que para cualesquiera funciones f y $g : \mathbb{N} \to \mathbb{R}^+$, se verifica:
    a) $\lim_{n\to\infty} \frac{f(n)}{g(n)} \in \mathbb{R}^+ \Rightarrow f(n) \in \Theta(g(n))$
    b) $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) \in \mathrm{O}(g(n))$ pero $f(n) \notin \Theta(g(n))$
    c) $\Theta((f(n))^2) = (\Theta(f(n)))^2$

25. Sean $f(n)$ y $g(n)$ dos funciones asintóticamente no negativas. Demostrar la veracidad o falsedad de cada una de las siguientes afirmaciones:
    a) $\max(f(n), g(n))$ es $\mathrm{O}(f(n) + g(n))$
    b) $\max(f(n), g(n))$ es $\Omega(f(n) + g(n))$

26. El tiempo de ejecución de un algoritmo A está descrito por la recurrencia:

$$T(n) = 7 \cdot T(n/2) + n^2$$

    Otro algoritmo A' tiene un tiempo de ejecución dado por:

$$T'(n) = a \cdot T'(n/4) + n^2$$

    ¿Cuál es el mayor valor de la constante a que hace al algoritmo A' asintóticamente más rápido que A?

27. Resolver exactamente las siguientes ecuaciones de recurrencia:
    a) $T(n) = 5 \cdot T(n-1) + 6 \cdot T(n-2) + 4 \cdot 3^n$, con $T(0) = 0$ y $T(1) = 36$
    b) $T(n) = n \cdot T^2(n/2)$, con $T(1) = 8$; siendo n potencia de 2

28. Resolver exactamente la siguiente ecuación de recurrencia:

$$T(n) = 2 \cdot T(\sqrt{n}) + \log_2 \log_2 n, \text{ para } n \geq 4 \text{ y } T(2) = 1$$

29. Resolver exactamente la siguiente recurrencia, siendo n una potencia de 2:

$$T(n) = \frac{2^n \cdot T^2(n/2)}{T(n/4)}, \quad T(1) = 1,\ T(2) = 2$$