Análisis y Diseño de Algoritmos
Complejidad
(2º de Grado Ing. Inf., Ing. Sw., Ing. Comp.)
E.T.S.I. INFORMÁTICA

Relación de Problemas de Complejidad

1. De las siguientes afirmaciones, indicar cuales son ciertas y cuales no:

   a) $n^2 \in O(n^3)$
   b) $n^3 \in O(n^2)$
   c) $2^{(n+1)} \in O(2^n)$
   d) $(n+1)! \in O(n!)$
   e) $f(n) \in O(n) \Rightarrow 2^{f(n)} \in O(2^n)$
   f) $3^n \in O(2^n)$
   g) $\log n \in O(n^{1/2})$
   h) $n^{1/2} \in O(\log n)$
   i) $n^2 \in \Omega(n^3)$
   j) $n^3 \in \Omega(n^2)$
   k) $2^{(n+1)} \in \Omega(2^n)$

2. Supongamos la función $T(n) = 2T(n-1) + 3^n(n+5)$, con $n \geq 1$. Se pide calcular el orden de complejidad resolviendo la ecuación de forma exacta para la condición inicial $T(0) = 0$.

3. Resolver las siguientes ecuaciones y dar su orden de complejidad:

   a) $T(n) = 3T(n–1) + 4T(n–2)$ si $n > 1; T(0) = 0; T(1) = 1$.
   b) $T(n) = 4T(n/2) + n^2$ si $n > 1, n$ potencia de $2; T(1) = 1;$.
   c) $T(n) = 2T(n/2) + n \log n$ si $n > 1, n$ potencia de $2; T(1) = 1$.
   d) $T(n) = 3T(n/2) + 5n + 3$ si $n > 1, n$ potencia de $2; T(1) = 1$.
   e) $T(n) = 2T(n/2) + \log n$ si $n > 1, n$ potencia de $2; T(1) = 1$.
   f) $T(n) = T(n–1) + 2T(n–2)–2T(n–3)$ si $n > 2; T(n) = 9n^2–15n + 106$ si $n = 0, 1, 2$.
   g) $T(n) = 2T(n/4) + n^{1/2}$ si $n \geq 4, n$ potencia de $4; T(1) = 1$.
   h) $T(n) = 4T(n/3) + n^2$ si $n \geq 3, n$ potencia de $3; T(1) = 1$.
   i) $T(n) = T(n-1) + 6T(n-2) + n$, $T(0) = 0, T(1) = 1$

4. Para cada una de las siguientes recurrencias determine su orden de complejidad usando el Teorema Maestro. Una vez realizado esto, encuentre las ecuaciones exactas para cada una de ellas (no es necesario calcular los valores de los coeficientes que dependen de las condiciones iniciales):

   a) $T(n) = 4T(n/2) + n^2$
   b) $T(n) = 2T(n/2) + n \log n$
   c) $T(n) = 3T(n/2) + 5n + 3$
   d) $T(n) = 2T(n/2) + \log n$
   e) $T(n) = 5T(n/2) + (n \log n)^2$
   f) $T(n) = 2T(\sqrt{n}) + \log n$

5. Dado un valor numérico x y un valor natural $n \geq 0$, queremos escribir un algoritmo para calcular x^n. Se pide:

   a) Diseñar un algoritmo para resolver el problema mediante un enfoque iterativo, e indicar su complejidad en términos del número de multiplicaciones ejecutadas.
   b) Diseñar un algoritmo alternativo siguiendo un enfoque de divide y vencerás, de manera que sea más eficiente que el enfoque anterior.
   c) Plantear y resolver (quizás mediante el Teorema Maestro) una recurrencia para el coste computacional del algoritmo de divide y vencerás.

6. Calcular mediante expansión de recurrencias el tiempo de ejecución y determinar la complejidad en los casos mejor y peor del siguiente algoritmo:

```
public static int recursiva (int n){
    if ( n <= 1 )
         return 1;
    else
         return recursiva (n-1)+recursiva(n-1);
};
```

7. Dado un array A de n numeros enteros distintos que se encuentra ordenado, y un número entero x, diseñar un algoritmo que determine (devuelva un booleano) si existen dos elementos de A cuya suma sea exactamente x.

   a) ¿Puedes proporcionar la complejidad exacta de tu algoritmo?
   b) Existe una solución sencilla que es de orden lineal. ¿La has encontrado?

Problemas complementarios

1. Demostrar las siguientes inclusiones estrictas: $O(1) \subset O(\log n) \subset O(n) \subset O(n \log n) \subset O(n^2) \subset O(n^3) \subset O(n^k) \subset O(2^n) \subset O(n!)$.

2. Demostrar que si $f \in O(g) \Leftrightarrow g \in \Omega(f)$.

3. Dar un ejemplo de funciones f y g tales que $f \in O(g)$ pero que $f \notin \Omega(g)$.

4. Demostrar que $\forall a, b > 1$ se tiene que $\log_a n \in \Theta(\log_b n)$.

5. Considérense las siguientes funciones de n:

   $f_1(n) = n^2$;

   $f_2(n) = n^2 + 1000n$;

   $$f_3(n) = \begin{cases} n & \text{si } n \text{ es impar} \\ n^3 & \text{si } n \text{ es par} \end{cases};$$

   $$f_4(n) = \begin{cases} n & \text{si } n <= 100 \\ n^3 & \text{si } n > 100 \end{cases};$$

   Para cada posible valor de i, j indicar si $f_i \in O(f_j)$ y si $f_i \in \Omega(f_j)$.

6. Suponiendo que $T_1 \in O(f)$ y que $T_2 \in O(f)$, indicar cuáles de las siguientes afirmaciones son ciertas:

   a) $T_1 + T_2 \in O(f)$.
   b) $T_1 – T_2 \in O(f)$.
   c) $T_1/T_2 \in O(1)$.
   d) $T_1 \in O(T_2)$.

7. Encontrar dos funciones $f(n)$ y $g(n)$ tales que $f \notin O(g)$ y $g \notin O(f)$.

8. Consideremos los algoritmos que obtienen el recorrido en inorden y la altura de un árbol binario, implementar y determinar la complejidad de ambos algoritmos.

9. Consideremos un algoritmo recursivo que calcula la suma de los dígitos de un numero natural, implementar el algoritmo y calcular su tiempo de ejecución.

10. Calcular para el siguiente algoritmo que corresponde al algoritmo de ordenación por selección el tiempo de ejecución y determinar la complejidad en los casos mejor y peor:

```
static void intercambia (int[] a, int i, int j) {
 int temp;
 temp = a[i];
 a[i] =a [j];
 a[j] = temp;
}

static int posMinimo (int[] a, int prim, int ult) {
 int k;
 int pmin = prim;
 for ( k = prim+1; k <= ult; k++ )
      if ( a[k] <a[pmin] )
        pmin = k;
 return pmin;
}

public static void seleccion (int[] a, int prim, int ult) {
 int i;

 for ( i = prim; i <ult; i++ ) {
     intercambia (a, i, posMinimo (a, i, ult));
 }
}
```

11. Calcular el tiempo de ejecución y determinar la complejidad de los casos mejor y peor del siguiente algoritmo:

```
public static void algoritmo (int n) {
 int x = 1;
 int i, j, k;
 for ( i = 1; i <= n; i++ )
     for ( j = i+1; j <= n; j++ )
         for ( k = i + j - 1; k <= n; k++ )
             x = x+ 2;
}
```

12. Ordenar las siguientes funciones en orden creciente de complejidad: $(n-2)!, 4\ln(n+100)^{10}, 2^{2n}, 10^{-6}n^5 + 9n^3, (\ln n)^2, \sqrt[3]{n}, 5^n, n + 10^{10}$.