Centro de Investigación y de Estudios Avanzados del IPN

Departamento de Ingeniería Eléctrica
Sección de Computación

Programa de Maestría en Ingeniería Eléctrica
con Opción en Ciencias de la Computación

Análisis y Diseño de Algoritmos
Enero-Abril 2005

Lista de Ejercicios No. 1

1. Demostrar que 2^n es un $O(3^n)$, $n > 0$.

2. Demuestre que para cualesquiera constantes reales a y b, donde $b > 0$, $(n+a)^b = \Theta(n^b)$.

3. Demuestre que el tiempo de ejecución de un algoritmo es $\Theta(g(n))$ si y solamente si su tiempo de ejecución del peor caso es $O(g(n))$ y el tiempo de ejecución del mejor caso es $\Omega(g(n))$.

4. Sea P un problema. La complejidad en tiempo P es en el peor caso $O(n^2)$. Así también, la complejidad en tiempo P es en el peor caso $\Omega(n\log n)$. Sea A un algoritmo que resuelve a P. ¿Cuáles de las siguientes afirmaciones son consistentes con la información acerca de la complejidad de P? Justifique sus respuestas.

   a) A tiene complejidad en tiempo del peor caso $O(n^2)$.
   b) A tiene complejidad en tiempo del peor caso $O(n^{3/2})$.
   c) A tiene complejidad en tiempo del peor caso $O(n)$.
   d) A tiene complejidad en tiempo del peor caso $\Theta(n^2)$.
   e) A tiene complejidad en tiempo del peor caso $\Theta(n^3)$.

5. Determine cual es el valor de regreso de la siguiente función y determine una cota superior asintótica para el peor caso para el tiempo de ejecución del programa siguiente:

```
int PowersOfTwo(int n)
{
     int  i;
     while( ! ODD(n)  ) {
          n = n / 2;
          i = i + 1;
     }
     return i;
}
```

   La función ODD regresa un valor verdadero si n es impar, regresa un valor falso en caso contrario, y toma un tiempo $O(1)$.

6. Dar una regla para calcular el tiempo de ejecución de la instrucción ***case*** en Pascal.

7. Considere el procedimiento cuyo cuerpo es

```
sum := 0
for( i = 1; i < f(n); i++ )
     sum := sum + 1;
```

   Donde $f(n)$ es un llamado a la función f. Para cada uno de los casos siguientes proporciones un cota superior al tiempo de ejecución del procedimiento

   a. El tiempo de ejecución de $f(n)$ es $O(n)$, y el valor de $f(n)$ es $n!$
   b. El tiempo de ejecución de $f(n)$ es $O(n)$, y el valor de $f(n)$ es n
   c. El tiempo de ejecución de $f(n)$ es $O(n^2)$, y el valor de $f(n)$ es n
   d. El tiempo de ejecución de $f(n)$ es $O(1)$, y el valor de $f(n)$ es 0

8. Sea

$$p(n) = \sum_{i=0}^{d} a_i n^i$$

   donde $a_d > 0$ y sea k una constante. Demuestre las siguientes propiedades.

   a. Si $k \geq d$, entonces $p(n) = O(n^k)$.
   b. Si $k \leq d$, entonces $p(n) = \Omega(n^k)$.
   c. Si $k = d$, entonces $p(n) = \Theta(n^k)$.
   d. Si $k > d$, entonces $p(n) = o(n^k)$.
   e. Si $k < d$, entonces $p(n) = w(n^k)$.

9. ¿Cuál es el valor que regresa la siguiente función? Exprese su respuesta como una función de n. Dar el tiempo de ejecución del peor caso usando la notación O.

```
int P1(int n)
{
      int   r=0;
      for( i = 1; i <= n; i++ )
            for( j = 1; j <= i; j++ )
                  for( k = j; k <= i+j; k++ )
                        for( l = 1; l <= i+j-k; l++ )
                              r = r + 1;
      return r;
}
```

10. ¿Cuál es una cota superior asíntótica justa al tiempo de ejecución de los siguientes fragmentos de programa?

```
a) sum = 0;
   for( i = 0; i < n; i++ )
         for( j = 0; j < n; j++ )
               for( k = 0; k < n; k++ )
                     sum++;
b) sum = 0;
   for( i = 0; i < n; i++ )
         for( j = 0; j < n; j++ )
               for( k = j; k < n; k++ )
                     sum++;

c) sum = 0;
   for( i = 0; i < n; i++ )
         for( j = 0; j < i; j++ )
               for( k = 0; k < n; k++ )
                     sum++;

d) sum = 0;
   for( i = 0; i < n; i++ )
         for( j = 0; j < i; j++ )
               for( k = j; k < n; k++ )
               sum++;
e) sum = 0;
   for( i = 0; i < n; i++ )
         for( j = 0; j < i; j++ )
               for( k = 0; k < i; k++ )
               sum++;
```

11. Suponga una variante del quicksort en el cual siempre se elige como pivote el primer elemento del subarreglo que va a ser ordenado. ¿Qué modificaciones al algoritmo dado en clase se tendrían que hacer para evitar ciclos infinitos cuando hay secuencias de elementos iguales? ¿Cuál sería el orden de complejidad del algoritmo modificado?

12. Modificar el quicksort para encontrar el k-ésimo elemento más pequeño de un arreglo desordenado de elementos.

13. Representar el peor y el mejor caso para los siguientes métodos de ordenamiento con el siguiente conjunto de datos de entrada S = { 25, 6, 9, 13, 26, 4, 2, 17 } y calcular el número de comparaciones y movimientos que se realizan en cada caso.

    a) QuickSort
    b) HeapSort

14. Supongamos que tenemos una recurrencia de la forma

    $T(1) = a$
    $T(n) = T(n\text{-}1) + g(n)$, para $n > 1$

    Dar una cota superior para $T(n)$ cuando $g(n)$ es de la forma

    a. n^2

b. $n^2 + 3n$
c. $n^{3/2}$
d. $n\log n$

15. Supongamos que tenemos una recurrencia de la forma

$T(1) = a$
$T(n) = cT(n/d) + bn^k$, para n un potencia positiva de d

Expandiendo iterativamente $T(n/d^i)$ para $i = 1, 2, \ldots$ Muestre que

a. si $c > d^k$, entonces, $T(n)$ es un $O(n^{\log_d c})$
b. si $c = d^k$, entonces, $T(n)$ es un $O(n^k \log n)$
c. si $c < d^k$, entonces, $T(n)$ es un $O(n^k)$

16. Demuestre que usando únicamente comparaciones, no puede existir un algoritmo que resuelva el problema de ordenamiento en un tiempo ***promedio*** menor a $O(n \log n)$. Sugerencia: Analice la longitud promedio de las ramas de los árboles de decisión en donde un árbol con n nodos tiene i nodos en la rama izquierda y n-i nodos en la rama derecha; tendrá que variar i para todas las posibilidades.

17. Supongamos que estamos comparando dos programas en la misma máquina de los algoritmos de ordenamiento por inserción y por mezcla (mergesort). Para entradas de tamaño n, el programa con el algoritmo de inserción toma un tiempo $8n^2$, mientras que el programa con el algoritmo de mezcla toma $64n\log n$ pasos. ¿Para cuales valores de n el algoritmo de inserción es mejor que el algoritmo de mezcla?. ¿Cómo podría mejorar el algoritmo de mezcla para que sea más rápido sobre las entradas de tamaño pequeño?

18. El método de ordenamiento por inserción se puede expresar recursivamente como sigue. Para ordenar A[1..n], se ordena recursivamente A[1..n-1] y se inserta A[n] en el arreglo ordenado A[1..n-1]. Escriba una recurrencia para el tiempo de ejecución de la versión recursiva del ordenamiento por inserción y resuélvala.

19. Demostrar que $\log (n!)$ es $\Theta(n \log n)$.

20. Determine, por iteración, una cota superior asintótica de la recurrencia

$T(n) = 3T(n/2) + n$

21. Use el teorema maestro para dar cotas asintóticas de las siguientes recurrencias.

a) $T(n) = 4T(n/2) + n$
b) $T(n) = 4T(n/2) + n^2$
c) $T(n) = 4T(n/2) + n^3$