Teoría de la Computación 2024

Lab 08

06.noviembre.2024

1. Suponga que cada una de las expresiones siguientes da el tiempo de procesamiento $T(n)$ gastado por un algoritmo para resolver un problema de tamaño n. Seleccione el(los) términos dominante(s) que tienen el aumento más pronunciado en n y especificar el Big-Oh de menor complejidad para cada algoritmo.

Expresión	Términos dominantes	$O(\cdot)$
$5 + 0.001n^3 + 0.025n$		
$500n + 100n^{1.5} + 50n \log_{10} n$		
$0.3n + 5n^{1.5} + 2.5n^{1.75}$		
$n^2 \log_2 n + n(\log_2 n)^2$		
$n \log_3 n + n \log_2 n$		
$3 \log_8 n + \log_2 \log_2 \log_2 n$		
$100n + 0.01n^2$		
$2^n + n^{0.5} + 0.5n^{1.25}$		
$0.01^n \log_2 n + n(\log_2 n)^2$		
$100n \log_3 n + n^3 + 100n$		
$0.003 \log_4 n + \log_2 \log_2 n$		

2. Calcular cuál es el tiempo promedio que requiere un computador para hacer una operación aritmética de tipo suma entre dos cantidades.
   Sugerencia, hacer una simulación de sumar dos enteros a y b, y repetirla una cantidad grande N de veces. Sumar el tiempo que tarda estas repeticiones y luego calcular el tiempo promedio.

   a) ¿Hay diferencias al repetir varias veces este experimento?. Repita el experimento $k = 10^5$ veces, y muestre una distribución de los tiempos promedio. Analice sus resultados.

   b) ¿Hay diferencias al incrementar el tamaño de las cantidades a y b. Pruebe con números de tamaños, 5 dígitos, 10 dígitos, 15 dígitos, y compare los tiempos de ejecución.

   c) ¿Hay diferencias según el tipo de dato? Experimente con números enteros, y luego repita los experimentos con variables de tipo float, y compare las diferencias entre tiempos de ejecución.

   d) ¿Hay diferencias según la operación aritmética? Sustituya el cálculo de $a + b$ por a/b, y repita los experimentos con esta otra operación. Compare las diferencias entre tiempos de ejecución.

3. Muestre que en el algoritmo de reducción gaussiana (para resolver sistemas de ecuaciones):
   Algoritmo: (Eliminación Gaussiana).
   Inputs: $A \in \mathbb{R}^{n \times n}$, Outputs: $L \in \mathbb{R}^{n \times n}$, $U \in \mathbb{R}^{n \times n}$.

   Initialise $U = A$, $L = I$ (la matriz identidad de $n \times n$)
   For k = 1 to n-1:
       For i = k + 1 to n:
           $L_{i,k} = 1 \ / \ U_{i,k}$
           For j = k to n:
               $U_{i,j} = U_{i,j} - L_{i,j} U_{i,j}$.

   hay exactamente $n^3 + n^2 - 2n$ operaciones aritméticas (sumas, restas, productos, cocientes, y asignaciones), de modo que el tiempo de ejecución es de orden $O(n^3)$.

4. Proporcione un análisis del tiempo de ejecución (notación Big-Oh) para cada uno de los siguientes 4 fragmentos de programa. Tenga en cuenta que el tiempo de ejecución corresponde aquí al número de veces que se ejecuta la operación suma. *sqrt* es la función que devuelve la raíz cuadrada de un número dado.

   a)
```
suma = 0
For i in range(0, sqrt(n)/2):
    suma += 1
    For j in range(0, sqrt(n)/4):
        suma += 1
        For k in range(0, 8+j):
            suma += 1
```

   b)
```
suma = 0
For i in range(0, sqrt(n)/2):
    For j in range(i, 8+i):
        For k in range(j, 8+j):
            suma += 1
```

   c)
```
suma = 0
For i in range(0, 2*n):
    For j in range(0, i*i):
        For k in range(0, j):
            If (j % i == 1):
            suma += 1
```

5. Se debe elegir entre uno de los dos paquetes de software, A ó B, para procesar colecciones de datos, que contienen cada uno hasta 10^9 registros. El tiempo promedio de procesamiento del software A es $T_A(n) = 0.001n$ ms y el tiempo promedio de procesamiento del software B es $T_B(n) = 50\sqrt{n}$ ms.
   ¿Cuál algoritmo tiene un mejor rendimiento en el sentido Big-Oh?
   ¿Cuál algoritmo es mejor para una entrada de tamaño $n = 10^4$?
   ¿A partir de qué valor de n es mejor el desempeño del algoritmo B?