Jeffrey Camareno Fonseca Carne: 2013099863

Tarea
Análisis de algoritmos IC3002.40 2014
Prof. Mauricio Rojas

1. Asuma que cada una de las expresiones de abajo da el tiempo de procesamiento T(n) que toma un algoritmo para resolver un problema de tamaño n.
   Seleccione el término dominante que tenga el mayor crecimiento en n y especifique la complejidad O más baja para cada algoritmo.

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

2. Un algoritmo cuadrático tiene un tiempo de procesamiento T(n) = c * n^2 y tarda T(Z) segundos para procesar Z elementos de datos. Cuanto tiempo tardaría en procesar n = 5000 elementos de datos, asumiendo que Z = 100 y que T(Z) = 1ms?

T(z)= c * n^2=1ms
T(100)=c * 10000=1ms
C=1/10000

T(5000)=(1/10000) * $(5000)^2$ = 2500s = 2,500ms

Por lo tanto para procesar 5000 elementos de datos tardaría 2,500 mili segundos

3. Un algoritmo con una complejidad de tiempo O(f(n)) y un tiempo de procesamiento T(n) = c f(n) donde f(n) es una función conocida de n, tarda 10 segundos para procesar 1000 elementos de datos. Cuanto tiempo tomara procesar 100000 elementos de datos si f(n) = n y si f(n) = n^3?

T(n) = c f(n) = 10 sec
T(1000) = c f(1000) = 10

C = 10 / f(1000) ms

Entonces:

Con f(n) = n

T(100 000) = 10 / f(1000) * f(100 000)
T(100 000) = 10 / 1000 * 100 000 = 1000 ms

Con f(n) = n^3

T(100 000) = 10 / f(1000) * f(100 000) ^{3}
T(100 000) = 10 / $(1000)^3$ * (100 000) ^{3} = 10 000 000 ms

4. Se tienen dos paquetes de software A y B de complejidad O(n log n) y O(n) respectivamente. Y se tiene que

$$T_A(n) = C_a n \log_{10} n \text{ y}$$
$$T_B(n) = C_b n$$

expresan el tiempo en milisegundos de cada paquete.

Durante una prueba, el tiempo promedio de procesamiento de $n = 10^4$ elementos de datos con los paquetes A y B es de 100ms y 500ms respectivamente. Establezca las condiciones en las cuales uno de los paquetes empieza a desempeñarse mejor que el otro y recomiendo la mejor opción si se van a procesar $n = 10^9$

$T_A(10^4) = C_A * 10^4 \log 10^4 = 100ms$
$C_A = 100 / 10^4 \log(10^4) = 1/400$

$T_B(10^4) = CB * 10^4 = 500ms$
$C_B = 5 / 100 = 0{,}05$

$T_A(10^9) = 1/400 * 10^9 \log 10^9 = 22{,}500000$

$T_B(10^9) = 0{,}005 * 10^9 = 50{,}000000$

Por lo tanto es mejor T_A

5. Asuma que el arreglo a contiene n valores, que el método randomValue toma un numero constante c de pasos computacionales para producir cada valor de salida, y que el método goodSort toma n log n pasos computacionales para ordenar un arreglo. Determine la complejidad O para el siguiente fragmento de código:

```
for ( i = 0; i < n; i++ ) {
   for ( j = 0; j < n; j++ ) {
     a [ j ] = randomValue ( i );
   }
   goodSort( a ) ;
}
```

En el primer ciclo es n y además el good sort es n log n, por lo tanto la complejidad O es $O(n^2 \log n)$

6. Se le pide clasificar un archivo que contiene enteros entre 0 y 999999. No puede utilizar un millón de casillas, asi que en su lugar decide utilizar mil casillas numeradas desde 0 a 999 (recordar el ordenamiento por casillas o buckets comentado en clase). Comienza la clasificación colocando cada entero en las casillas correspondiente a sus tres primeras cifras. A continuación utiliza mil veces la ordenación por inserción para ordenar el contenido de cada casilla por separado. Y por último se vacían las casillas por orden para obtener una secuencia completamente ordenada.
Haga el pseudocódigo del algoritmo y realice el análisis del tiempo de ejecución. Compare con los tiempos esperados para ejecutar el ordenamiento solo usando el algoritmo de ordenamiento por inserción.

7. Cree una máquina de Turing que reemplace su Nick por su nombre. (Puede utilizar solo las primeras 3 letras del Nick y las primeras 5 del nombre).

= delimitador inicial
& = delimitador final

#	J	E	F	0	0	&