Tarea

Estudiante Kevin Méndez Arce 201129313

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(…) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

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?

N = 5000

Z = 100

T(Z) = 1ms

T(Z) = c\*n^2

1 = c\*(100^2)

C = 1/10000

T(n) = c\*n^2

T(n) = (1/10000)\*(5000)^2

T(n) = 2500 ms

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) = 10

f(1000)

10 = c\*f(1000)

C = 10/f(1000)

To mando f(n) = n

T(n) = c\*100000

T(n) = (10/f(1000)\*f(100000))

T(n) = (1/100\*100000)

T(n) = 1000 seg

Tomando f(n) = n^3

T(n) = (10/f(1000)\*f(100000))

T(n) = (1/100000000\*1000000000000000)

T(n) = 10000000 seg

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

y

expresan el tiempo en milisegundos de cada paquete.

Durante una prueba, el tiempo promedio de procesamiento de 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

Para cualquier valor que se encuentre antes de se recomienda utilizar la función , ya que el tiempo es menor. Después de esta cifre se recomienda utilizar

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++ ) { 🡪3n+1

for ( j = 0; j < n; j++ ) {🡪 3n+1

a [ j ] = randomValue ( i ); 2c

}

goodSort( a ) ; 🡪 n 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 pseudo código 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.

For (i <= n) For que recorre la lista. Valor = n

For(j<=n/10) For encargado de crear mis 1000 casillas 2n+1. Valor = 3n/1000+1

If (list[i]/1000 == j) Valor 3n/10+1

For (k < n/1000) For encargado de hacer el desglose de la casilla. Valor 3n/10+1

If (n%1000 =k) 3n+

Listinsert[j][k] = n%1000 Se inserta en casilla

Para el presente código se utiliza un O(n3)

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

