Estudiante: Eduardo Andrés Chavarría Rey Carnét: 2013012779

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 (…) |
|  |  | O () |
|  |  | O () |
|  |  | O () |
|  |  | O () |
|  |  | O () |
|  |  | O () |
|  |  | O () |
|  |  | O () |
|  |  | O () |
|  | n(log2 n)2 | O (n(log n)2) |
|  |  | O () |
|  |  | 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?

Necesitamos saber T(5000)=c \* 5000^2, más sin embargo no conocemos el valor de la constante c.   
 Por lo tanto usaremos los datos Z=100 y T(Z)=1ms, lo que implica que T(100)=1ms.  
**Sabemos que (conclusiones):**



**Lo que implica que:**

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?

Necesitamos saber el valor de T(n), para n=100.000, para T(n)=c\*n y para T(n)=c\*n^3.

**Sabemos que (conclusiones):**

10=T (1000)=c\*1000 y 10=T (1000)=c\*(1000) ^3

* 10=c\*1000 y 10= c\*1,000,000,000
* c=10/1000 y c=10/1,000,000,000
* c= 1 / 100 y c= 1/100,000,000

**Lo que implica que:**

* T(100,000)= 1/100\*100,000 y T(100,000)= 1/100,000,000 \* 100,000^3
* T(100,000)= 100,000/100 y T(100,000)= 100,000^3/100,000,000
* T(100,000)=1000 y T(100,000)= 10,000,000

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 .

**Sabemos que (conclusiones) para :**

* 100ms = TA(104) = Ca \*104 \*log10(104)
* 100 = Ca \*104 \*log10(104)
* = Ca \*10,000
* = Ca \*10,000
* Ca=
* Ca=
* Ca=

**Y además sabemos (concluimos) para**

* 500ms =
* 500=

**Partiendo de n=0, logramos identificar que:**

* ≥

**Gracias a los despejes anteriores sabemos que:**

* ≥

Punto a partir del cual empieza a desempeñarse mejor.

Con lo que concluimos, que para un tiempo promedio de procesamiento de , la mejor elección sería usar el paquete de software 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++ ) { 2 n

for ( j = 0; j < n; j++ ) { 2n

a [ j ] = randomValue ( i ); 2\*C

}

goodSort( a ) ; n\*log (n)

}

O(n)= 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, así 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.

Definir Bucketsort (lista archivo) :

Casillas = Matriz de 1000x1000 1

Para i=0 hasta i=tamanio del archivo { 2n (

Casillas[archivo[i]%1000 ] = archivo 4 )

}

Imprimir (Casillas) n

1+2n (4)+n = 9n+1

O(n)= n

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).

Nickname: Lobezno

Nombre: Eduardo

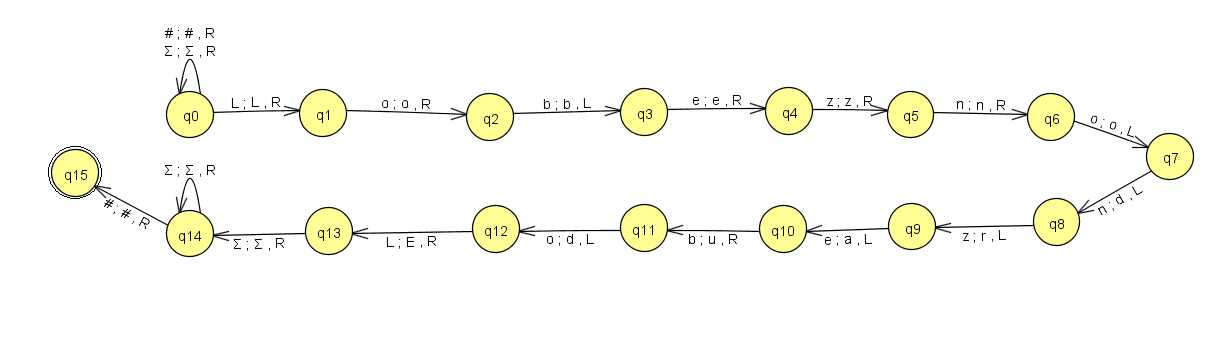
El conjunto de valores que usará esta máquina de Turing, será el equivalente a los del alfabeto en español, el espacio (\_) y el carácter numeral (#).

#: Indica el inicio y el final de la cinta.

Σ: Cualquier letra del alfabeto y el espacio.

R: En la 3° columna equivale a “->”

L: En la 3° columna equivale a “<-”



Entrada:  
 #Esta\_es\_la\_tarea\_de\_Lobezno\_de\_análisis#

Salida:

#Esta\_es\_la\_tarea\_de\_Eduardo\_de\_análisis#