Facultad Politécnica – Ingeniería Informática
Algoritmos y Estructura de Datos III – 2018 – 1er Semestre – Sección TQ
Prof. Cristian Cappo

Análisis de algoritmos

Tarea #3 – A entregar vía EDUCA el Jueves 9/03/2018 hasta las 23:55 hs – TP: 70

Los documentos debe estar en formato PDF. En el caso de código, indique claramente el número de ejercicio y solo envíe los fuentes. Por cada ejercicio de código haga una subcarpeta separada. No incluya las clases de java en ningún paquete. Siempre debe colocar la identificación de los miembros del grupo y el identificador de grupo.

Ejercicio 1 (10p) (Código en Java)

Genere vectores de componentes aleatorios cuyos longitudes varíen varíen de 50000 a 1000000, de a 50000. Los valores posibles para cada elemento es de 10000 a 1000000. Luego realice N búsquedas aleatorias utilizando la estrategia de **búsqueda binaria** (esto es, busque un elemento aleatorio en el propio arreglo) y N búsquedas aleatorias usando una **búsqueda lineal**.

Genere la tabla parecida a la que se muestra abajo:

N	*T(n)* en ms	Búsqueda binaria			Búsqueda Lineal		
		T/N	T/(N $\log_2$ N)	T/N^2	T/N	T/(N $\log_2$ N)	T/N^2
50000							
100000							
150000							
..							

Observaciones:

- Para aplicar la búsqueda binaria el vector debe estar ordenado, para ello aplique el algoritmo de *quicksort* utilizando la mediana de tres para la elección del pivot (ver método en libro de WEISS). Para ello cree una clase separada denominada *Util*, donde deberá estar el código del algoritmo de ordenación. *El cálculo de tiempo no debe incluir la ordenación*.
- Su programa debe generar la tabla mencionada, es decir no debe solo transcribir los resultados en un documento. Utilice un formato legible para los decimales, utilice *printf()* para ello con las cadenas de formato adecuadas.
- A continuación un breve código de como calcular el tiempo de ejecución (en ms) en Java:

```
public class MedicionTiempo {
  public static void main ( String [] args) {
     final int MAX_ITER = 100000000;
     long t1, t2;
     t1 = System.currentTimeMillis();
     /* Medimos cuanto tiempo lleva una iteracion de 10000000 */
     for (int k=0; k < max_iter; k++) ;
     t2 = System.currentTimeMillis();
     System.out.println("Tiempo : " + (t2 - t1) + "ms");
  }
}
```

En un documento separado, explique de forma clara y precisa los resultados que obtuvo en sus experimentos. Debe explicar los valores obtenidos y las tendencias de cada columna (que corresponden a la Busqueda Binaria y la Búsqueda Lineal). Aquí debe colocar los resultados que obtuvo en su entorno de pruebas además indique que plataforma de pruebas utilizó (CPU, RAM, Sistema Operativo, versión de Java, etc.).

Ejercicio 2 (8p) (En Documento)

a) Encuentre el costo asintótico en términos de Θ de la siguiente recurrencia. Puede utilizar cualquier método conocido para encontrarlo (4p)

$$T(n)=\begin{cases} 1 & si \quad n=0 \\ 2\mathrm{T}(\frac{n}{2})+\frac{n}{\log n} & si \quad n>0 \end{cases}$$

b) Encuentre el orden de crecimiento en término de O (*O(g(n)*) de las siguientes sumas (utilice la función *g(n)* más simple y precisa posible).

a) $\sum_{i=0}^{n-1}(i^2+1)^2$ b) $\sum_{i=1}^{n}(i+1)\,2^{i-1}$ c) $\sum_{i=2}^{n-1}\log_2 i^2$ d) $\sum_{i=0}^{n-1}\sum_{j=0}^{i-1}(i+j)$

Ejercicio 3 (10p) (En Documento)

a) Dado este código en Java, Indique la ecuación de recurrencia (2p), resuélvala e indique el costo asintótico en términos de Θ (2p) .

```
int f ( int n ) {
    if n > 3  then  { return  f ( n -1 ) * f ( n - 3 ) ; }
    else return 3;
}
```

b) Considere el tradicional algoritmo de ordenación por selección (*SelectSort*) :

- Escriba el algoritmo en Java, 2*p*
- Encontrar la T(n) del algoritmo que escribió teniendo en cuenta el peor caso. 2p
- Calcular el costo asintótico en notación O, Ω y Θ para el peor caso. 2*P*

Ejercicio 4 (5p) (En Documento)

a) Demuestre utilizando la definición formal de Ω (*no por límite*) que $\frac{1}{2}\log^2 n = \Omega(\log n)$ *(3p)*

b) Indique por cada expresión si es falsa o verdadera. Si es falsa fundamente. *(2p)*

b.1) Para cualquier constante *x, y > 1* tenemos que $n^x = O(y^n)$ _____

b.2) $n^3 + 4n\log n + 5n = n^3 + \Theta(n\log n)$ _____

Ejercicio 5 (10p) (En Documento)

Resuelva las siguientes recurrencias, por el método que usted conozca. Justificar las respuestas. Asumir que ***T(n)*** es constante para una ***n*** suficientemente pequeña, por ejemplo *T(n)=cte si n==1.*

a) $T(n) = T(\frac{9n}{10}) + n$

b) $T(n) = T(n-1) + \log n$

c) $T(n) = 4T(\frac{n}{2}) + n^2 lg\, n$

d) $T(n) = T(n-1) + \frac{1}{n}$

Ejercicio 6 (10p) (En Documento y en código)

Suponga que tiene el siguiente método Java:

```
public static boolean contiene ( int [] [] m, int valor ) {
    int N = m.length;
    for ( int f = 0; f < N; f++ )
        for ( int c = 0; c < N; c++ )
            if ( m[f][c] == valor )
                return true;
    return false;
}
```

1	2	3	4
3	4	5	6
5	6	7	8
9	10	11	12

m : representa una matriz de N filas y N columnas, donde en cada fila los elementos se encuentran ordenados de forma creciente y en cada columna los elementos también se encuentran ordenados en forma creciente (como el ejemplo de arriba). Retorna *true* si *valor* se encuentra en *m* o *false* en caso contrario.

a) ¿Cuál es el tiempo de ejecución (*T(N) = ..*) de este método considerando el peor caso?. Indique las cotas: *O*, *Θ*, *Ω* *(5p)*
b) Suponga que toma 4 segundos la ejecución en una matriz de 100x100. ¿Cuánto tiempo aproximadamente tomaría la ejecución en una matriz de 400x400? *(2p)*
c) Reescriba el algoritmo, en Java, de forma que ahora tenga una mejor cota superior en el peor caso. Fundamente. *(3p)*

Ejercicio 7 (5p) (En Documento)

Encuentre la forma cerrada de la siguiente expresión y luego demuestre por inducción matemática la correctitud de dicha forma cerrada.

$$\sum_{i=1}^{n} \frac{1}{i(i+1)}$$

Ejercicio 8 (12p) (En Documento)

Por cada par de funciones *f(n)* y *g(n)* dada en la tabla de abajo, escribir Θ si *f(n)* = Θ*(g(n))* , escribir O si *f(n)* = O*(g(n)),* escribir Ω si *f(n)* = Ω*(g(n))* o escribir X si ninguna relación se cumple.

Θ, O, Ω o X	*f(n)*	*g(n)*
	$n-100$	$n-200$
	$n^{1/2}$	$n^{2/3}$
	$100n+\log n$	$n+(\log n)^2$
	$10\log n$	$\log(n^2)$
	$(\log n)^{\log n}$	$n/\log n$
	$\sqrt{n}$	$(\log n)^3$
	$n\,2^n$	3^n
	$n!$	$(n+1)!$
	$n\log n$	$10n\log 10n$
	$n^{1.01}$	$(\log n)^{10}$
	$n^{1/2}$	$5^{\log_2 n}$
	$n^2/\log n$	$n(\log n)^2$