Análisis de Algoritmos 2024

Trabajo Práctico 2 - Notación Asintótica 2

Fecha de entrega: 6 de Septiembre de 2024

El objetivo de este trabajo es realizar los ejercicios propuestos de forma colaborativa. Para los ejercicios compartidos entre grupos, pueden compartir el desarrollo del ejercicio o realizar dos resoluciones diferentes.
En particular para este práctico se pide en algunos casos tener código funcionando, verifica tener una computadora que ejecute tu código si te tocó algún ejercicio de este tipo. Se pedirá ejecución en clase. Además completa la justificación de tu respuesta con gráficas de las funciones. Podés usar GeoGebra para esto.
Todos los ejercicios deben ser resueltos por todos los estudiantes, pero en este trabajo sólo deben ser publicados los que te tocaron según la asignación de la tabla siguiente. Dicha asignación corresponde a la presentación oral del TP. Si el día de la práctica hay varias resoluciones logradas, elegiremos 1 grupo para exponer ese ejercicio.
Por supuesto, si querés compartir la resolución de otros ejercicios en este proyecto Overleaf, también son bienvenidas pero no es obligatorio.

Nota: Colocá el nombre del/de los estudiante/s que resolvió/resolvieron el ejercicio al iniciar la resolución del ejercicio que te tocó.

Las respuestas deben quedar escritas en este mismo .tex, en Overleaf.

Grupos (Estudiantes por grupo)	**Ej. asignado**
Grupo 1 (Sthefany, Victoria, Paula, Adriano, Facundo)	1.3 — 2.5*a* — 3.1
Grupo 2 (Sebastián, Bautista, Antonio, Albany, Luis)	2.4.a— 2.5*b* — 3.2
Grupo 3 (Lucas, Valentina, Nicanor, Ignacio, Franco)	2.1 — 2.4.b — 3.3
Grupo 4 (Manuel, Jan, Ulises, Demian , Luciano)	2.2 — 2.4.c — 3.1
Grupo 6 (Roman, Ulises, Guillermo, Jamiro, Juan)	1.2 —2.6 — 3.2
Grupo 7 (Christopher, Tomás, Joaquín, Leonard, Facundo)	1.4 — 2.5*c* — 3.3
Grupo 8 (Facundo, Belén, Jeremías, Bruno, Kevin)	1.1 — 2.3 — 3.2

1. Repaso de O

1. Supongamos un algoritmo con un costo de $O(n)$ tarda 20 segundos en realizar un determinado procesamiento sobre una computadora de 3GHz. Responda:

a) ¿Cuánto se tardaría en hacer el mismo procesamiento en una máquina de 1 GHz?

b) ¿Como se calcularía el tiempo de ejecución si la cantidad de datos a procesar se duplica?

Ejercicio 1.1 - Grupo 8: Belén, Bruno, Facundo, Jeremias, Kevin

a) Sabiendo que el costo de un algoritmo de $f(n)$ tarda 20 segundos en una computadora de 3 GHz, entonces para calcular el tiempo en una computadora de 1 GHz, se tiene que:

- Una computadora de 1 GHz es 3 veces más lenta que una de 3 GHz
- Entonces el tiempo que se tardaria en procesar en una maquina de 1GHz es 3 veces mayor

$$\therefore \text{Tiempo en 1 GHz = 20 segundos x 3 = 60 segundos}$$

b)

2. ¿Qué tipo de crecimiento caracteriza mejor a estas funciones?

3. Dadas dos clases de complejidad $O(f(n))$ y $O(g(n))$, decimos que $O(f(n)) \subseteq O(g(n))$ si y sólo si para toda función $h(n) \in O(f(n))$ sucede que $h \in O(g(n))$

   a) ¿Qué significa que $O(f(n)) \subseteq O(g(n))$? ¿Qué se puede concluir cuando simultáneamente $O(f(n)) \subseteq O(g(n))$ y $O(f(n)) \supseteq O(g(n))$? Pruebe su afirmación por contradicción.

   b) Ordene y grafique utilizando $\subseteq$ las siguientes clases de complejidad:

   - $O(1)$
   - $O(\sqrt{x})$
   - $O(\sqrt{2})$
   - $O(log\ x)$
   - $O(log\ x!)$
   - $O(x+1)$
   - $O(1/x)$
   - $O(log\ x)$
   - $O(log\ log\ x)$
   - $O(2^x)$
   - $O(x)$
   - $O(x^x)$
   - $O((log\ x))$
   - $O(x!)$
   - $O(x\ log\ x)$

4. ¿Por qué hay un error en la siguiente expresión? $O(f(n)) - O(f(n)) = 0$ Además, ¿cómo debería ser realmente el miembro derecho de la igualdad anterior?

2. Orden Exacto: Θ

1. ¿Qué significa que el tiempo de ejecución de un algoritmo está *en el orden exacto de* $f(n)$?
2. Demostrar que:
   a) $T(n) = 5 \cdot 2^n + n^2$ está en el orden exacto de 2^n .
   b) $T(n) = n^3 + 9 * n^2 * log(n)$ está en el orden exacto de n^3.
3. Indicar, respondiendo si/no, para cada par de expresiones *(A, B)* de la siguiente tabla, si *B* es O, Ω o Θ de *A*. Suponer que $k \geq 1, \epsilon > 0$ y $c > 1$ son constantes.

A	B	$O(B)$	$\Omega(B)$	$\Theta(B)$
$log^k n$	n^e			
n^k	c^n			
$\sqrt{n}$	$n^{sin\ n}$			
2^n	$2^{n/2}$			
$n^{log\ c}$	$c^{log\ n}$			
$log\ n$	$lg\ n^n$			

Cuadro 0.1: Completar con *si* o *no* según *A* sea de *B*

A	B	$O(B)$	$\Omega(B)$	$\Theta(B)$
$log^k n$	n^e			
n^k	c^n			
$\sqrt{n}$	$n^{sin\ n}$			
2^n	$2^{n/2}$			
$n^{log\ c}$	$c^{log\ n}$			
$log\ n$	$lg\ n^n$			

4. Demostrar las siguientes propiedades:
   a) (transitiva) $f(n) \in \Omega(g(n))\ y\ g(n) \in \Omega(h(n)) \Rightarrow f(n) \in \Omega(h(n))$
   b) (simetría)$f(n) \in \Theta(g(n)) \Leftrightarrow g(n) \in \Theta(f(n))$

Resolución - Grupo 3: Margni, Villarroel, Fernandez, Mendiberri, Fabris

Para probar la propiedad de simetría la hacermos en dos pasos:

- ($\Rightarrow$) Si $f(n) \in \Theta(g(n))$ entonces $g(n) \in \Theta(f(n))$ **(I)**
- ($\Leftarrow$) Si $g(n) \in \Theta(f(n))$ entonces $f(n) \in \Theta(g(n))$ **(II)**

Empezaremos probando (I)

Asumimos a $f(n) \in \Theta(g(n))$ como verdadera y lo reescribimos utilizando la definición de orden exacto

- $f(n) \in \Theta(g(n)) \Leftrightarrow \exists c_0, d_0 \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0 : c_0 \cdot g(n) \leq f(n) \leq d_0 \cdot g(n)$ **(1)**

Luego queremos llegar a que se cumpla que $g(n) \in \Theta(f(n))$, es decir

- $g(n) \in \Theta(f(n)) \Leftrightarrow \exists c_1, d_1 \in \mathbb{R}^+, \exists n_1 \in \mathbb{N}, \forall n \geq n_1 : c_1 \cdot f(n) \leq g(n) \leq d_1 \cdot f(n)$ **(2)**

En (1) por un lado tenemos que

- $c_0 \cdot g(n) \leq f(n)$
- $g(n) \leq \frac{1}{c_0} \cdot f(n)$
- $g(n) \leq d_1 \cdot f(n)$ **(3)** (Con $d_1 = \frac{1}{c_0} \in \mathbb{R}^+$)

Y por otro lado, en (1)

- $f(n) \leq d_0 \cdot g(n)$
- $\frac{1}{d_0} \cdot f(n) \leq g(n)$
- $c_1 \cdot f(n) \leq g(n)$ **(4)** (Con $c_1 = \frac{1}{d_0} \in \mathbb{R}^+$)

$\therefore$ Usando (3) y (4) llegamos a que $c_1 \cdot f(n) \leq g(n) \leq d_1 \cdot f(n)$ y así queda probado que (2) se cumple (es decir, $g(n) \in \Theta(f(n))$)
Por consecuencia (I) es verdadero

Ahora probaremos (II)

La prueba de esto es exactamente idéntica a la que realizamos en (I), con la única salvedad de que tenemos que reemplazar cada aparición de $f(n)$ por $g(n)$ y viceversa

$\therefore$ (II) es verdadero

Luego como (I) y (II) se cumplen, queda probada la propiedad de simetría
Es decir, $f(n) \in \Theta(g(n)) \Leftrightarrow g(n) \in \Theta(f(n))$

c) (regla de dualidad) $f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$

5. Sean P_1 y P_2 dos programas cuyos tiempos de ejecución son $T_1(n)$ y $T_2(n)$ respectivamente donde n es el tamaño de la entrada. Determine para los siguientes casos en qué condiciones P_2 se ejecuta más rápido que P_1:

a) $T_1(n) = 2n^2 \qquad T_2(n) = 1000n$

b) $T_1(n) = 3n^4 \qquad T_2(n) = 3n^3$

c) $T_1(n) = 126n^2\ T_2(n) = 12n^4$

6. Considere las siguientes funciones $f : \mathbb{N}^0 ß \mathbb{R}^+$:

$f_1(n) = 2 * n^5 - 16 * n^3$

$f_2(n) = 2 * n^5$

$f_3(n) = \{ \begin{matrix} 2 * n^3 & si\ n\ es\ par \\ 5 * n^4 & si\ n\ es\ impar \end{matrix}$

Muestre que $f_1(n) \in O(f_2(n))$, y que $f_3(n)$ es $\Omega(n^4)$.

3. Algoritmia en Algoritmos iterativos

1. Considere el siguiente algoritmo para encontrar la distancia más corta entre dos elementos del arreglo de números:

```
1  MODULO distMin (ARREGLO ENTERO a, ENTERO n) RETORNA ENTERO
2  //Entrada: arreglo de n numeros
3  //Salida: Minima distancia entre dos elementos cualesquiera del arreglo.
4      dmin ← ∞
5      PARA i ← 0 HASTA n-1 HACER
6          PARA j ← 0 HASTA n-1 HACER
7              if (i! = j y a[i] − a[j]) < dmin ENTONCES
8                  dmin ← |a[i] − a[j]|
9      RETORNA dmin
10 FIN ALGORITMO
```

a) Realiza tantas mejoras como consideres necesario al pseudocódigo anterior, tanto de eficiencia como de sintaxis y semántica del algoritmo. Por ejemplo, intenta minimizar el número de comparaciones entre elementos.

b) Analizar el tiempo de ejecución teórico de la versión original y de la versión mejorada y verificarlo empíricamente.

c) Responda a las siguientes preguntas

(i) Exprese el tiempo teórico en notación $\Theta(f(n))$ ¿Por qué en este caso es mejor utilizar Θ que O?

(ii) A partir de sus pruebas empíricas, podría decir cuál es el valor n_0 para el cual la definición de Θ ocurre

2. Implementar un algoritmo tal que dado un número entero n (con $n < 10^{18}$) compute la parte entera de la raíz cuadrada de n, usando solo operaciones primitivas ($+, -, *, /, =, ==, <, >$) y cualquier estructura de control.

- Cualquier solución lograda con funciones predefinidas tales como **import Math.sqrt()** o similares, **no** es válida.
- La solución en $O(\sqrt{n})$ es lenta dado el tamaño de entrada. Tratar de resolver el problema en tiempo $log(n)$
- Muestre, a partir de calcular el tiempo teórico $T(n)$ que su solución es de orden logarítmico.

3. Implementar un algoritmo para encontrar todos los elementos comunes entre dos listas ordenadas de números. Por ejemplo:

```
//
para las listas
l1 = [ 2, 5, 5, 5 ]
l2 = [ 2, 2, 3, 5, 5, 7 ]
//
la salida deberia ser
elementosComunes = [ 2, 5, 5 ]
```

a) ¿Cuál es el máximo número de comparaciones que tu algoritmo realiza si las longitudes de las listas son m y n respectivamente?

b) Analizar asintóticamente.

c) Hacer un programa que pruebe empíricamente que dadas dos listas diga cuántas comparaciones se realizan.