Laboratorio 8 Teoría de la Computación

Jorge Andrino
Joab Hernández

Octubre 2025

Problema 1

(a) Complejidad asintótica

Pseudocódigo	**Costo**
`for (i =` $\lfloor n/2 \rfloor$`; i` $\leq$ `n; i++)`	$\Theta(n)$
`for (j = 1; j` $\leq n - \lfloor n/2 \rfloor$`; j++)`	$\Theta(n)$
`for (k = 1; k` $\leq$ `n; k = 2k)`	$\Theta(\log n)$

Cuadro 1: Estructura de bucles y costo por nivel (Problema 1).

Explicaciones de cada término.

- Bucle en i: recorre de $\lfloor n/2 \rfloor$ a $n \Rightarrow \#i = n - \lfloor n/2 \rfloor + 1 = \Theta(n)$.
- Bucle en j: la condición $j + n/2 \leq n$ implica $j \leq n - \lfloor n/2 \rfloor \Rightarrow \#j = \Theta(n)$.
- Bucle en k: duplica k hasta $n \Rightarrow \#k = \lfloor \log_2 n \rfloor + 1 = \Theta(\log n)$.

Multiplicación total.

$$T(n) \;=\; \Theta(n) \cdot \Theta(n) \cdot \Theta(\log n) \;=\; \boxed{\Theta(n^2 \log n)}.$$

(b) Resultados experimentales

Tabla de tiempos

n	tiempo (s)	fuente
1	2.400000e-06	measured
10	5.599999e-06	measured
100	8.892000e-04	measured
1 000	1.200920e-01	measured
10 000	1.667895e+01	estimated
100 000	2.084903e+03	estimated
1 000 000	2.501884e+05	estimated

Cuadro 2: Tiempos experimentales del **Problema 1**.

Gráfica

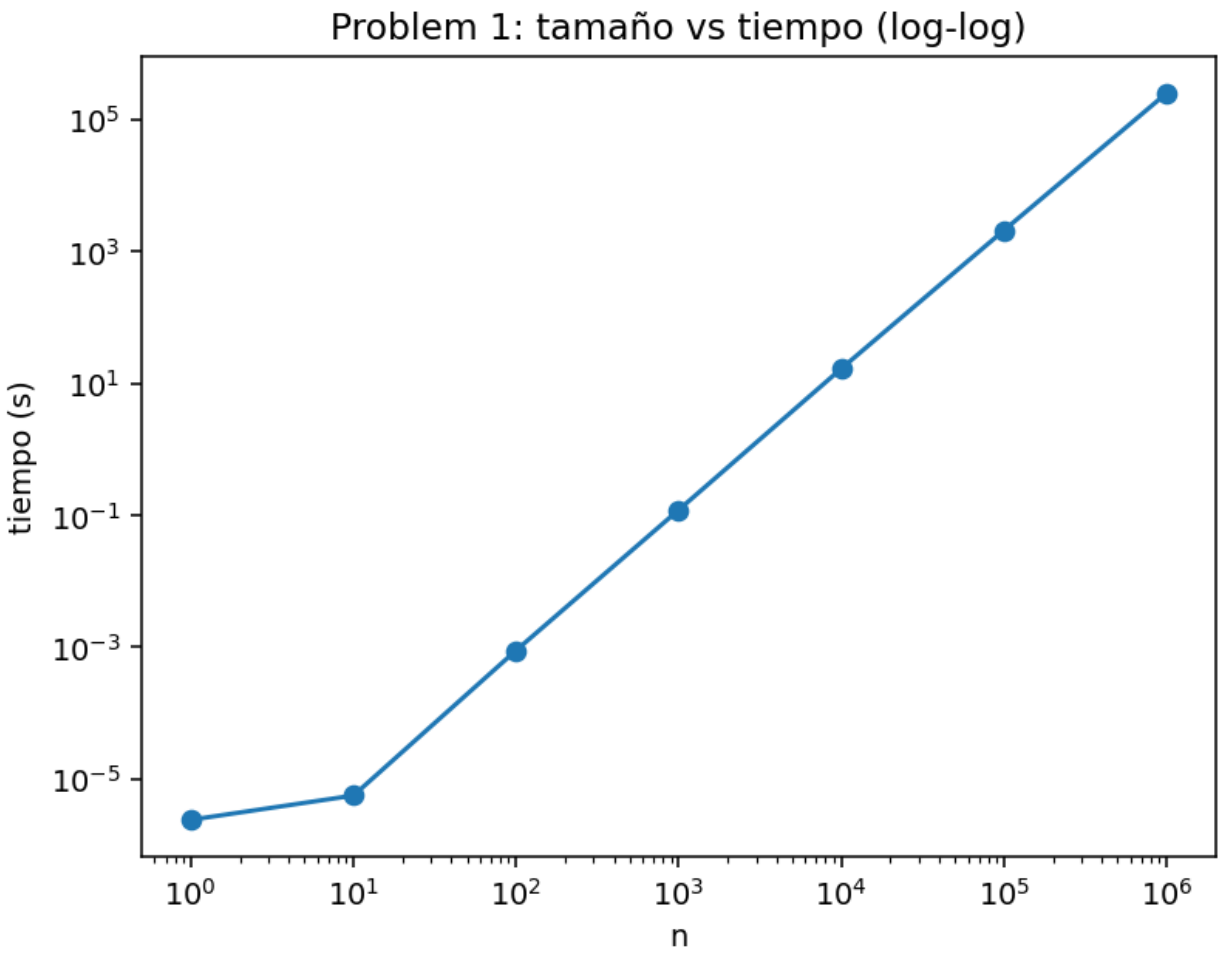

Figura 1: Curva n vs. tiempo (log–log) del Problema 1.

Problema 2

(a) Complejidad asintótica

Pseudocódigo	**Costo**
`if (n ≤ 1) return;`	$\Theta(1)$
`for (i = 1; i ≤ n; i++) {`	$\Theta(n)$
`  for (j = 1; j ≤ n; j++) { printf(...); break; }`	$\Theta(1)$ por cada i
`}`	

Cuadro 3: Estructura y costo por nivel (Problema 2).

Explicaciones de cada término.

- La rama $n \leq 1$ es constante.

- Para cada i, el bucle interno ejecuta una vez y se interrumpe con `break` $\Rightarrow$ trabajo constante por iteración externa.

Multiplicación total.

$$T(n) \;=\; \Theta(n) \cdot \Theta(1) \;=\; \boxed{\Theta(n)}.$$

(b) Resultados experimentales

Tabla de tiempos

n	tiempo (s)	fuente
1	6.999981e-07	measured
10	1.400000e-06	measured
100	5.400001e-06	measured
1 000	7.280000e-05	measured
10 000	9.189000e-04	measured
100 000	7.195600e-03	measured
1 000 000	6.978500e-02	measured

Cuadro 4: Tiempos experimentales del **Problema 2**.

Gráfica

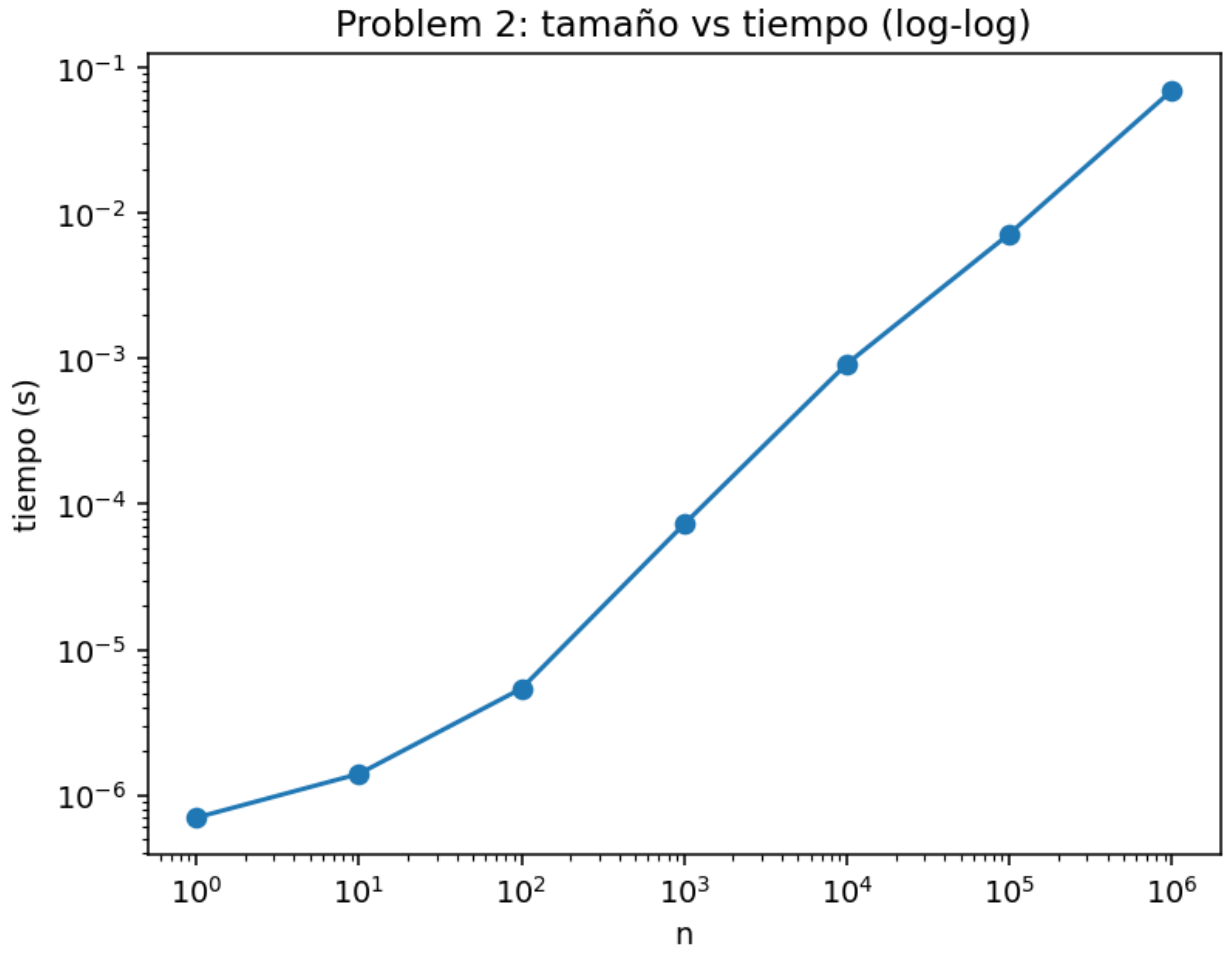

Figura 2: Curva n vs. tiempo (log–log) del Problema 2.

Problema 3

(a) Complejidad asintótica

Pseudocódigo	**Costo**
`for (i = 1; i ≤ ⌊n/3⌋; i++)`	$\Theta(n)$
`  for (j = 1; j ≤ n; j += 4)`	$\Theta(n)$

Cuadro 5: Estructura y costo por nivel (Problema 3).

Explicaciones de cada término.

- Bucle externo: $\lfloor n/3 \rfloor = \Theta(n)$.
- Bucle interno con paso 4: $\lceil n/4 \rceil = \Theta(n)$.

Multiplicación total.

$$T(n) \;=\; \Theta(n) \cdot \Theta(n) \;=\; \boxed{\Theta(n^2)}.$$

(b) Resultados experimentales

Tabla de tiempos

n	tiempo (s)	fuente
1	7.999988e-07	measured
10	1.400000e-06	measured
100	3.560000e-05	measured
1 000	3.998200e-03	measured
10 000	4.273643e-01	estimated
100 000	4.273927e+01	estimated
1 000 000	4.273930e+03	estimated

Cuadro 6: Tiempos experimentales del **Problema 3**.

Gráfica

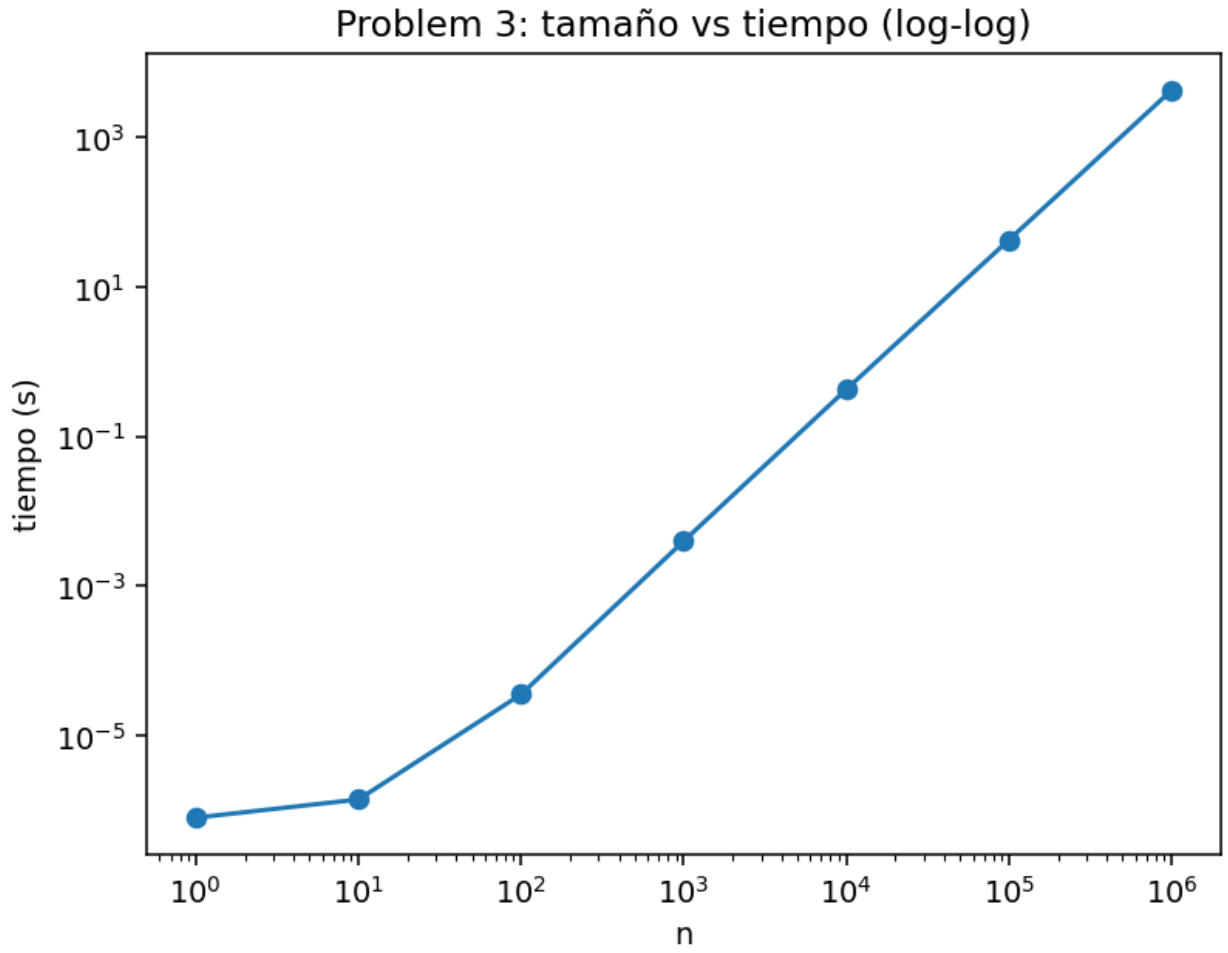

Figura 3: Curva n vs. tiempo (log–log) del Problema 3.

Ejercicio 4

1. Búsqueda Lineal (Linear Search)

El algoritmo recorre secuencialmente una lista de tamaño n comparando cada elemento con el valor buscado.

$$T(n) = \begin{cases} 1, & \text{si el elemento está en la primera posición (mejor caso)} \\ \dfrac{n}{2}, & \text{si el elemento está en una posición intermedia (caso promedio)} \\ n, & \text{si el elemento está al final o no está en la lista (peor caso)} \end{cases}$$

Complejidad asintótica:

$$\text{Mejor caso: } T_{\text{mejor}}(n) = O(1)$$
$$\text{Caso promedio: } T_{\text{prom}}(n) = O(n)$$
$$\text{Peor caso: } T_{\text{peor}}(n) = O(n)$$

2. Búsqueda Binaria (Binary Search)

Requiere que la lista esté ordenada. En cada paso compara el valor buscado con el elemento central y descarta la mitad restante.

$$T(n) = \begin{cases} 1, & \text{si el elemento está en la posición central (mejor caso)} \\ \log_2 n, & \text{número promedio de comparaciones (caso promedio)} \\ \log_2 n, & \text{siempre divide el espacio a la mitad (peor caso)} \end{cases}$$

Complejidad asintótica:

$$\text{Mejor caso: } T_{\text{mejor}}(n) = O(1)$$
$$\text{Caso promedio: } T_{\text{prom}}(n) = O(\log n)$$
$$\text{Peor caso: } T_{\text{peor}}(n) = O(\log n)$$

3. Quick Sort

Es un algoritmo de ordenamiento por división y conquista que selecciona un pivote, divide el arreglo en dos subarreglos y los ordena recursivamente.

$$T(n) = \begin{cases} n \log n, & \text{si el pivote divide uniformemente el arreglo (mejor caso)} \\ n \log n, & \text{promedio de divisiones equilibradas (caso promedio)} \\ n^2, & \text{si el pivote siempre es el menor o el mayor elemento (peor caso)} \end{cases}$$

Complejidad asintótica:

$$\text{Mejor caso: } T_{\text{mejor}}(n) = O(n \log n)$$
$$\text{Caso promedio: } T_{\text{prom}}(n) = O(n \log n)$$
$$\text{Peor caso: } T_{\text{peor}}(n) = O(n^2)$$

Ejercicio 5

1. Si $f(n) = \Theta(g(n))$ y $g(n) = \Theta(h(n))$, entonces $h(n) = \Theta(f(n))$.

   El enunciado sería verdadero. Por definición, $f(n) = \Theta(g(n))$ implica que existen constantes $c_1, c_2 > 0$ y n_0 tales que

   $$c_1 g(n) \leq f(n) \leq c_2 g(n), \quad \forall n \geq n_0.$$

   De igual forma, $g(n) = \Theta(h(n))$ implica que existen $d_1, d_2 > 0$ y n_1 tales que

   $$d_1 h(n) \leq g(n) \leq d_2 h(n), \quad \forall n \geq n_1.$$

   Componiendo ambas desigualdades se obtiene:

   $$(c_1 d_1) h(n) \leq f(n) \leq (c_2 d_2) h(n), \quad \forall n \geq \text{máx}(n_0, n_1),$$

   lo cual demuestra que $f(n) = \Theta(h(n))$, y por simetría, $h(n) = \Theta(f(n))$.

2. Si $f(n) = O(g(n))$ y $g(n) = O(h(n))$, entonces $h(n) = \Omega(f(n))$.

   El enunciado sería verdadero. Dado que O es transitiva: de $f = O(g)$ y $g = O(h)$ se sigue $f = O(h)$. Y ya que $f = O(h)$ es equivalente a $h = \Omega(f)$, se cumple lo que dice el enunciado.

3. El enunciado sería falso. En cada iteración interna, se crea una nueva tupla de longitud $k = j - i$, y tanto la creación como el cálculo del hash de la tupla cuestan $O(k)$. Por tanto, el costo total es:

   $$\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} (j - i) = \sum_{k=1}^{n-1} k(n - k) = \frac{(n-1)n(n+1)}{6} = \Theta(n^3).$$

   El tiempo de creación de la tupla inicial `atupla` es $O(n)$, pero no afecta el orden de crecimiento. Por lo tanto, $f(n) = \Theta(n^3)$.

Repositorio:

https://github.com/her21105/lab8_jorgeJoab_teoriacomputacion