



Análisis y diseño de Algoritmos

Guía de estudio

IIA/LCD

1. Notación asintótica

1. Supón que las funciones que se listan a continuación, indican el tiempo que requiere algún algoritmo, para resolver un problema de tamaño n . Elige el término o términos dominantes y determina a qué orden pertenece usando la notación O .

$f(n)$	Término dominante	$O(\dots)$
$5 + 0,001n^3 + 0,025n$		
$500n + 100n^{1,5} + 50n \log_{10} n$		
$0,3n + 5n^{1,5} + 2,5n^{1,75}$		
$n^2 \log_2 n + n(\log_2 n)^2$		
$n \log_3 n + n \log_2 n$		
$100n^2 + 0,01n^2$		
$0,01n + 100n^2$		
$100n \log_3 n + n^3 + 100n$		

2. Determina para cada una de las siguientes expresiones si es cierta o falsa. Justifica tu respuesta.

- $3n^2 + 10n \log n = O(n \log n)$
- $3n^2 + 10n \log n = \Omega(n^2)$
- $3n^2 + 10n \log n = \Theta(n^2)$
- $n \log n + n/2 = O(n)$
- $\sqrt{n} + \log n = O(n)$
- $\sqrt{n} + \log n = \Theta(\log n)$
- $\sqrt{n} + \log n = \Theta(n)$
- $2\sqrt{n} + \log n = \Theta(\sqrt{n})$
- $\sqrt{n} + \log n = \Omega(1)$
- $\sqrt{n} + \log n = \Omega(\log n)$
- $\sqrt{n} + \log n = \Omega(n)$

3. Ordena ascendentemente la siguiente lista de funciones, considerando la notación O :

$$\begin{aligned}f(n) &= c \text{ para cualquier constante } c \geq 0 \\f(n) &= n^2 \\f(n) &= n^2 + n + 1 \\f(n) &= 2^n \\f(n) &= n \\f(n) &= n \log n + 1 \\f(n) &= n + c; \text{ para cualquier constante } c \geq 0 \\f(n) &= n^n \\f(n) &= \sqrt{n} \\f(n) &= n^{1.3} \\f(n) &= \log n \\f(n) &= n! \\f(n) &= n^3 + 3\end{aligned}$$

4. Escribe el pseudocódigo para cada uno de los siguientes algoritmos de ordenamiento:

- a) burbuja
- b) insertion sort
- c) merge sort
- d) quick sort

5. Para cada uno de los algoritmos de ordenamiento del inciso anterior calcula el tiempo computacional en el peor de los casos.

2. Divide y vencerás

1. Resuelve cada una de las siguientes relaciones de recurrencia, usando un árbol de recursión, si es posible.

- a) $T(n) = 2T(n/3) + 1$
- b) $T(n) = 5T(n/4) + n$
- c) $T(n) = 7T(n/7) + n$
- d) $T(n) = 9T(n/3) + n^2$
- e) $T(n) = T(n-1) + n^c$, donde $c \geq 1$ es una constante
- f) $T(n) = T(n-1) + c^n$
- g) $T(n) = 4T(n/2) + n$
- h) $T(n) = 4T(n/2) + n^2$
- i) $T(n) = 4T(n/2) + n^3$

2. Imagina que tienes que elegir uno de los siguientes tres algoritmos:

- El algoritmo A resuelve un problema de tamaño n , dividiéndolo en cuatro subproblemas de la mitad del tamaño original, recursivamente soluciona cada subproblema y después combina las soluciones en un tiempo $O(1)$.

- El algoritmo B resuelve un problema de tamaño n , soluciona recursivamente cinco subproblemas de tamaño $n/2$ y combina las soluciones en un tiempo $O(n)$.
- El algoritmo C resuelve un problema de tamaño n , dividiéndolo en tres subproblemas de tamaño $n/2$ recursivamente soluciona cada subproblema y después combina las soluciones en un tiempo $O(n^2)$.

¿Cuál es la complejidad computacional de cada algoritmo (usa la notación O)? ¿Cuál algoritmo escogerías? Justifica tu respuesta.

3. Analiza el siguiente pseudocódigo, supón que la función g tiene una complejidad $O(n)$. ¿Cuál es la relación de recurrencia? Resuelve la relación de recurrencia, usando un árbol de recursión.

```
function f(n)
  if n > 1:
    f(n/3);
    f(n/3);
    f(n/3);
    g(n);
```

4. A continuación se muestran dos versiones del teorema maestro:

Teorema 1 Si $T(n) = aT(n/b) + O(n^d)$, para algunas constantes $a > 0, b > 1$ y $d \geq 0$ entonces

$$T(n) = \begin{cases} O(n^d) & \text{si } d > \log_b a \\ O(n^d \log n) & \text{si } d = \log_b a \\ O(n^{\log_b a}) & \text{si } d < \log_b a \end{cases}$$

Teorema 2 Sean $a \geq 1$ y $b > 1$ constantes, $f(n)$ una función y si $T(n) = aT(n/b) + f(n)$, entonces $T(n)$ tiene las siguientes cotas asintóticas:

- Si $f(n) = O(n^{\log_b a - \epsilon})$ para alguna constante $\epsilon > 0$, entonces $T(n) = \Theta(n^{\log_b a})$
- Si $f(n) = \Theta(n^{\log_b a})$, entonces $T(n) = \Theta(n^{\log_b a} \log n)$
- Si $f(n) = \Omega(n^{\log_b a + \epsilon})$ para alguna constante $\epsilon > 0$ y si $af(n/b) \leq cf(n)$, para alguna constante $c < 1$ y n suficientemente grande, entonces $T(n) = \Theta(f(n))$.

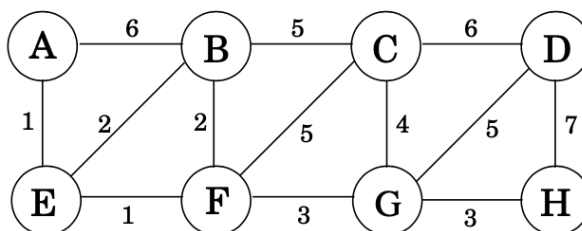
Responde las siguientes preguntas en tu tableta o cuaderno:

- Explica en tus propias palabras cuáles son las diferencias entre ambas versiones del teorema maestro
- Resuelve la relación de recurrencia $T(n) = 9T(n/3) + n$ usando cada una de las versiones del teorema maestro. Incluye todos los detalles de tus cálculos. Al aplicar el Teorema 2, especifica el valor de ϵ y de $f(n)$
- Resuelve la relación de recurrencia $T(n) = T(2n/3) + 1$ usando cada una de las versiones del teorema maestro. Incluye todos los detalles de tus cálculos. Al aplicar el Teorema 2, especifica el valor de ϵ y de $f(n)$
- Resuelve la relación de recurrencia $T(n) = 3T(n/4) + n \lg n$ usando cada una de las versiones del teorema maestro. Incluye todos los detalles de tus cálculos. En este caso al aplicar el teorema 2, hay que mostrar que se cumple que $af(n/b) \leq cf(n)$.

3. Algoritmos ávidos

Lee cuidadosamente cada uno de los siguientes problemas, encuentra una solución usando una heurística voraz y escribe el pseudocódigo correspondiente. Explica mediante un ejemplo cómo funciona.

1. Imagina que vas a viajar en automóvil con tus amigos hacia el norte del país, partiendo de la ciudad de México. Cuando inician el viaje, el tanque está lleno, y alcanzan a recorrer k kilómetros. Imagina que poseen un mapa que muestra las distancias entre las gasolineras a lo largo de la ruta. Sean $d_1 < d_2 < \dots < d_n$, las distancias para cada gasolinera, donde d_i indica la distancia desde la CDMX a la gasolinera en cuestión. Puedes suponer que la distancia entre gasolineras es lo más de k kilómetros. Tu objetivo es hacer el menor número de paradas a lo largo de la ruta. Diseña un algoritmo eficiente para determinar en cuáles gasolineras detenerte a cargar gasolina. Analiza la complejidad de tu algoritmo en términos de n , es decir, en términos del número de gasolineras.
2. Imagina que deseas n centavos en cambio, usando el menor número de monedas con denominaciones de 1, 10 y 25 centavos. Si se utiliza la estrategia ávida, ¿es posible obtener una solución óptima? Justifica por qué la estrategia ávida funciona o da un contraejemplo, para el cual la solución no sea óptima.
3. Dados los símbolos A, B, C, D, E, F, G y H con probabilidades $1/30, 1/30, 1/30, 2/30, 3/30, 5/30, 5/30, 12/30$, haz lo siguiente:
 - a) Construye el árbol correspondiente para determinar la codificación de Huffman.
 - b) Muestra la codificación de Huffman, es decir, para cada uno de los 8 símbolos da la codificación correspondiente.
4. Observa el siguiente grafo



- a) Usa el algoritmo de Kruskal para encontrar el árbol de recubrimiento mínimo.
- b) ¿Cuál es el costo de dicho árbol de recubrimiento mínimo?
- c) Usa el algoritmo de Prim para encontrar el árbol de recubrimiento mínimo.

4. Programación dinámica

1. Considera la secuencia $(A_n)_{n>0} = (1, 1, 3, 4, 8, 11, 21, 29, 55, \dots)$ definida como sigue:

$$\begin{aligned} A_1 &= A_2 = 1 \\ A_n &= B_{n-1} + A_{n-2} \quad n > 2 \\ B_1 &= B_2 = 2 \\ B_n &= A_{n-1} + B_{n-2} \quad n > 2 \end{aligned}$$

- a) Escribe el pseudocódigo de una solución recursiva y muestra que la complejidad de tal algoritmo es exponencial.
- b) Analiza la solución anterior y encuentra un algoritmo más eficiente.
2. Imagina que tienes una tarea con diferentes secciones etiquetadas de la A a la G. Cada sección tiene un puntaje y un tiempo aproximado para resolverla, tal y como se muestra en la siguiente tabla. Si tienes 15 horas para hacer la tarea, ¿qué secciones escogerías para obtener el mayor puntaje posible?

	A	B	C	D	E	F	G
puntaje	7	9	5	12	14	6	12
tiempo	3	4	2	6	7	3	5

- a) Usa programación dinámica para elegir las secciones que te permitan obtener el mayor puntaje. Lista las secciones que elegirías resolver en 15 horas.
- b) ¿Cuál es el máximo puntaje que se puede obtener?
3. Un estudiante de posgrado tiene 5 horas para resolver un examen con 4 problemas. Cada problema tiene cierto puntaje y toma cierto tiempo (horas) resolverlo. A continuación se muestra una tabla con los valores:

	1	2	3	4
puntaje	3	4	5	6
tiempo	2	3	4	5

- a) Escribe la función de optimización que te permite describir la solución óptima.
- b) Escribe el pseudocódigo de una solución recursiva.
- c) Puesto que la solución recursiva no es óptima, escribe una solución que utilice memoización.
- d) ¿Cuál sería la solución óptima para el estudiante de posgrado?
4. Si se tienen monedas de 1, 4, 5 y 10 centavos y se desea dar 8 centavos en cambio, ¿cuál sería la solución si se aplica el algoritmo ávido? ¿Es la solución óptima? Usa programación dinámica para construir la solución óptima.
5. Dadas las dimensiones de un conjunto de matrices $d = \{30, 35, 15, 5, 10, 20, 25\}$. Completa la tabla m y la tabla k , usando programación dinámica. ¿Cuál es la secuencia de multiplicación de menor costo?

$m :$

1	0		7825	9375	11875	
2		0	2,625	4,375	7,125	10,500
3			0	750		
4				0	1000	3500
5					0	
6						0

$k :$

1	0		1	3	3	
2	0	0	2	3	3	3
3	0	0	0	3		
4	0	0	0	0	4	5
5	0	0	0	0	0	
6	0	0	0	0	0	0

6. ¿Cómo se define la clase de problemas P? ¿Cómo se define la clase de problemas NP?
7. ¿Qué es un problema de decisión?
8. ¿Qué características debe tener un problema para pertenecer a la clase NP?

9. Para cada uno de los siguientes problemas, indica a qué clase pertenece P o NP y explica por qué

- Coloreado de un grafo
- SAT y k-SAT
- Primalidad
- Problema del camino hamiltoniano
- Problema de la cobertura de vértices