



## PROYECTO 1

Docente: Cecilia Hernández

Integrantes: Linna Cao Merino

Bastián Gómez Jara

Jesús Gómez Zambrano

Concepción, abril de 2022

1. [0.5 puntos] Determine si las siguientes afirmaciones son verdaderas o falsas. En cada caso, muestre los valores de las constantes  $n_0$  y  $c$  que hacen cierta su afirmación.

a)  $\log(n!)$  es  $O(\log(n))$

Sabemos que:

$$n! > n \text{ , para todo } n > 1$$

Hipótesis se mantiene si se aplica en logaritmo, tal que:

$$\log(n!) > \log(n)$$

Si escribimos esta ecuación como definición de  $w$   
Tenemos que:

$$\log(n!) > \log(n) + c, \text{ con } c > 0$$

Así tenemos que  $\log(n!) \in \omega(\log(n))$

Como podemos ver  $w$  y  $O$  son del mismo orden,  
esto no puede ocurrir, ya que  $w$  es constante  
inferior estricta

∴ Podemos concluir que la afirmación es falsa  
 $\forall n > 1$  y  $c > 0$

b)  $2^{100} n^4$  es  $\omega(n^4)$

Por def:

$$2^{100} n^4 > g(n) \quad , \quad g(n) = n^4$$

$$\Rightarrow 2^{100} n^4 > n^4 \cdot c$$

Para  $n_0 > 0$ , simplificamos por  $n^4$

$$\Rightarrow c < 2^{100}$$

$\forall n_0 > 0$  y  $c < 2^{100}$ , la afirmación es Verdadera

c)  $(\sqrt{16})^{\log(n)}$  es  $\Theta(\sqrt{n})$

Simplificamos la expresión, tal que:

$$(\sqrt{16})^{\log(n)} = 4^{\log(n)} = 2^{2 \log(n)} = 2^{2 \log(n)} = n^2$$
$$\Rightarrow (\sqrt{16})^{\log(n)} = n^2$$

Ahora por def de  $\Theta$ , tenemos que:

$$g(n)C_1 \leq n^2 \leq g(n)C_2 \quad , \text{ para todo } n > 0$$

$$\text{Sea } g(n) = n^2$$

para los intervalos:

$C_1 n^2 \leq n^2$ , simplificamos por  $n^2$

$$\Rightarrow C_1 \leq 1$$

Para la cota superior

$$n^2 \leq C_2 n^2, \text{ simplificamos por } n^2$$

$$1 \leq C_2$$

$\therefore (\sqrt{n})^{\log(n)} \in \Theta(n^2)$  para  $n_0 > 0, C_1 \leq 1$   
 $C_2 \geq 1$ , por lo tanto la afirmación es falsa

d)  $\sqrt{n} - 3 \log(n^{10}) \in \Omega(n)$

Simplificamos la expresión, tal que:

$$\sqrt{n} - 3 \log(n^{10}) = \sqrt{n} - 30 \log(n)$$

si encontramos un  $O$  menor al  $\Omega$  propuesto  
la afirmación es falsa, por lo tanto basta mostrar  
el  $O$  de la expresión, tal que:

$$\sqrt{n} - 30 \log(n) \leq g(n)c$$

para  $n \geq 1, 30 \log(n) > 0$

$$\Rightarrow \sqrt{n} - 30 \log(n) \leq \sqrt{n}$$

Entonces  $\Rightarrow g(n) = \sqrt{n}$  con  $c > 0$

$$\sqrt{n} - 30 \log(n) \leq \sqrt{n}c$$

Por lo tanto  $\sqrt{n} - 3 \log(n^{10}) \in O(\sqrt{n})$

$\therefore \forall n_0 \geq 1, c > 0$  la cota superior de  $\sqrt{n} - 3 \log(n^{10})$  es  
 $O(\sqrt{n})$ , como  $O \subset \Omega$  la afirmación es falsa

$$e) n^2 / 4^{200} \in O(n^2)$$

Por def:

$$n^2 / 4^{200} < g(n)C \quad , \quad g(n) = n^2 \quad n > 0$$

$$\Rightarrow n^2 / 4^{200} < n^2 C, \text{ simplificamos por } n^2$$

$$\Rightarrow 1 / 4^{200} < C$$

•  $\forall n > 0$  y  $C > 1 / 4^{200}$ , la afirmación es verdadera

2. [0.5 puntos] Ordene de menor a mayor orden asintótico las siguientes funciones.

Se nos pide ordenar las siguientes funciones:

a)  $3^{4^{1000000}}$

b)  $n\sqrt[3]{n}$

c)  $3^{0.1n}$

d)  $n^2$

e)  $\log(n)^2 \log(\log(n))$

Primero identificamos sus órdenes asintóticos

a)  $3^{4^{1000000}} \rightarrow$  Al solo ser constantes es  $O(1)$

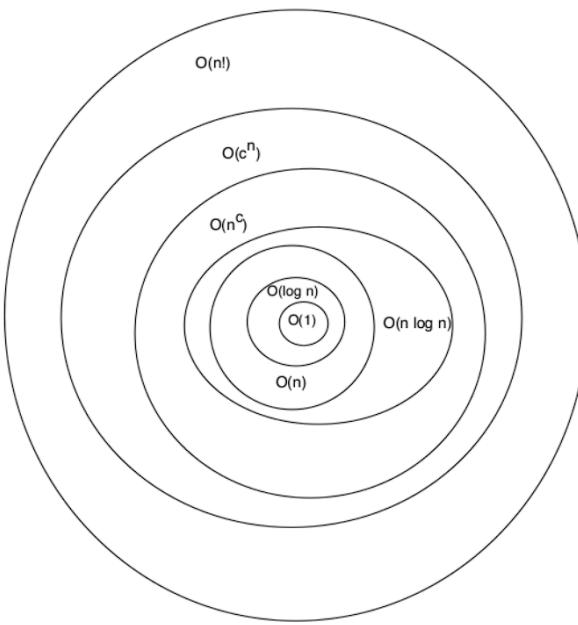
b)  $n\sqrt[3]{n} \Rightarrow$  Simplificamos tal que  $n\sqrt[3]{n} = n^{4/3} \in O(n^{4/3})$

c)  $3^{0.1n} \Rightarrow$  para  $n$  muy grandes podemos exhibirlo de manera  $3^n \in O(3^n)$

d)  $n^2 \Rightarrow$  Simplemente  $\in O(n^2)$

e)  $(\log(n))^{2 \log(n)} \Rightarrow E: (\log(n))^{\log(n^2)} \in O((\log(n))^{\log(n^2)})$

Usando el Diagrama de Benoit visto en clase:



Así podemos ordenar las funciones, tal que:

$$3^{100000} < n \sqrt[3]{n} < n^2 < 3^{o,n} < \log(n)^{2 \log(n)}$$

3. [0.5 puntos] Para cada una de las siguientes funciones  $f(n)$ , encuentre un  $c > 0$  y un  $n_0$  que muestre que  $f(n) \in O(g(n))$ . Explique por qué tales valores funcionan bien.

a)  $f(n) = 45n^3 + \sqrt{n} - 10$ ,  $g(n) = 15n^4$

Sabemos que para todos  $n \geq 1$

$$45n^3 \leq 45n^4$$

Y que:

$$\sqrt{n} \leq n^4$$

Ahora:

$$45n^3 + \sqrt{n} - 10 \leq 45n^4 + n^4 \leq 15n^4 \cdot c$$

$$\Rightarrow 45n^3 + \sqrt{n} - 10 \leq 46n^4 \leq 15n^4 \cdot c$$

utilizando la inequación de la derecha:

$$46n^4 \leq 15n^4 \cdot c / \frac{1}{n^4}$$

$$\Rightarrow 46 \leq 15 \cdot c \Rightarrow c \geq 46/15$$

si escogemos  $n_0 = 1$  y  $c = 46/15$   
tenemos que,

$$45(1)^3 + \sqrt{1} - 10 \leq 15 \cdot 46/15$$

$$\Rightarrow 45 + 1 - 10 \leq 15 \cdot 46/15$$

$$\Rightarrow 36 \leq 46$$

∴ Se cumple la condición

$$b) f(n) = 18 \log(n), g(n) = 30n + 2n \log(n)$$

Sabemos que, para todo  $n_0 \geq 1$

$$18 \log(n) \leq 18n \log(n) + 30n$$

Ahora decimos:

$$18 \log(n) \leq 18n \log(n) + 30n \leq C(30n + 2n \log(n))$$

Tomaremos la inferioridad de la derecha:

$$18n \log(n) + 30n \leq C(30n + 2n \log(n)) \quad / \cdot \frac{1}{30n + 2n \log(n)}$$

$$\frac{18n \log(n) + 30n}{30n + 2n \log(n)} \leq C$$

Para  $n_0 = 2$

$$\frac{18(2) \log(2) + 30(2)}{30(2) + 2(2) \log(2)} \leq C$$

$$\Rightarrow \frac{36 + 60}{60 + 4} \leq C \Rightarrow \frac{96}{64} \leq C \Rightarrow \frac{3}{2} \leq C$$

Si tomamos  $n_0 = 2$  y  $C = 3/2$ , tenemos que:

$$18 \log(2) \leq \frac{3}{2} (30 \log(4) + 4 \log(2))$$

$$18 \leq 36C \quad //$$

∴ se cumple la condición

4. [0.5 puntos] Resuelva las siguientes recurrencias

a)  $T(n) = 7T(n/2) + cn^2$

En este caso podemos aplicar el Teo. Maestro

$$a=7 \quad b=2 \quad f(n) = cn^2$$

Este caso, es un Caso 1 del Teo. Maestro y por lo tanto:

$$f(n) \in O(n^{\log_2(7-\epsilon)}) \text{, cuando } \epsilon=3$$

Así  $T(n)$  es  $\Theta(n^{\log_2(7)})$

b)  $T(n) = 16T(\sqrt{n}) + \log_2(n)$

Decimos que:  $k = \log_2(n) \Rightarrow n = 2^k$   
 $\Rightarrow T(2^k) = 16T(2^{k/2}) + \log_2(2^k)$

Ahora Decimos que:

$$S(k) = T(2^k) \Rightarrow S(k/2) = T(2^{k/2})$$

$$S(k) = 16S(k/2) + k$$

Por teo. Maestro, con Caso 1:

$$f(n) \in O(n^{\log_2(16-\epsilon)}), \text{ con } \epsilon=14$$

$$\Rightarrow S(k) \in \Theta(k^{\log_2(16)}) \Rightarrow S(k) \in \Theta(k^4)$$

$$\Rightarrow T(n) \in \Theta((\log_2(n))^4)$$

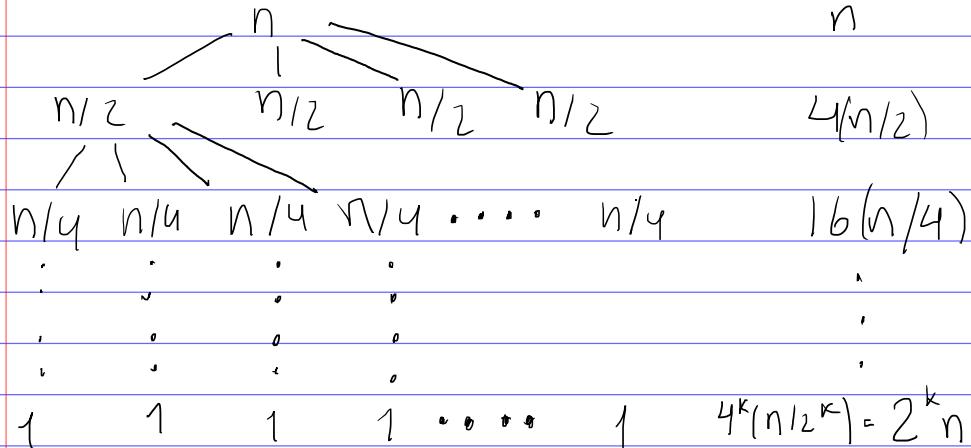
5. [0.5 puntos] Construya los árboles recursivos para las siguientes recurrencias y estime la solución de la recurrencia y luego demuestre por sustitución.

$$a) T(n) = 4T(\frac{n}{2}) + n$$

Construimos el Árbol recursivo:

Tamaño

Costo



Para  $T(n/2)$  llega a  $T(n)$  en el nivel  $k$ , entonces

$$1 = \frac{n}{2^k} \Leftrightarrow n = 2^k \Rightarrow k = \log(n)$$

Luego cada nivel aporta un costo, para el costo total  
Tenemos:

$k = \log(n)$ , para  $T(n)$ , entonces el costo para  $T(n)$  es:

$$\sum_{k=1}^{\log(n)} 2^k n = n \cdot \sum_{k=1}^{\log(n)} 2^k \cdot \left( \frac{2^{\log(n)+1} - 1}{2-1} \right) \cdot n$$

$$= n \cdot \left( 2 \cdot 2^{\log(n)} - 1 \right) = 2n^2 - n$$

Como las ramas son iguales podemos decir  
que por Aproximación  $T(n)$  es  $\Theta(2n^2 - n)$

Ahora queremos demostrar por sustitución que

$$T(n) = 4T(n/2) + n \in \Theta(2n^2 - n)$$

Primero vemos la cota superior, para esto suponemos:

$$T(n) \leq C(2n^2 - n), \text{ para todo } n_0 > 0$$

$$T(n) = 4T(n/2) + n \leq C(2n^2 - n)$$

$$\Rightarrow 4(C(2(n/2)^2 - n/2)) + n \leq C(2n^2 - n)$$

$$\Rightarrow 4(Cn^2/2 - nc/2) + n \leq 2n^2C - nc$$

$$\Rightarrow 2n^2C - 2nc + n \leq 2n^2C - nc / + (nc - 2n^2C)$$

$$\Rightarrow n - nc \leq 0 / \cdot \frac{1}{n}$$

$$\Rightarrow 1 - C \leq 0$$

$$\Rightarrow 1 \leq C \quad (\text{i})$$

Ahora vemos la cota inferior:

Hipótesis:  $T(n) \geq C(2n^2 - n)$ , para todo  $n_0 > 0$

$$T(n) = 4T(n/2) + n \geq C(2n^2 - n)$$

$$\Rightarrow 4(C(2(n/2)^2 - n/2)) + n \geq C(2n^2 - n)$$

$$\Rightarrow 4(Cn^2/2 - nc/2) + n \geq 2n^2C - nc$$

$$\Rightarrow 2nc^2 - 2nc + n \geq 2n^2C - nc / + nc - 2n^2C$$

$$\Rightarrow n - nc \geq 0 / \cdot \frac{1}{n}$$

$$\Rightarrow 1 - C \geq 0$$

$$\Rightarrow 1 \geq C \quad (\text{ii})$$

De (i) decimos que  $T(n) \in \Theta(2n^2 - n)$  para  $C \geq 1$

y de (ii) decimos que  $T(n) \in \Omega(2n^2 - n)$  para  $C \leq 1$

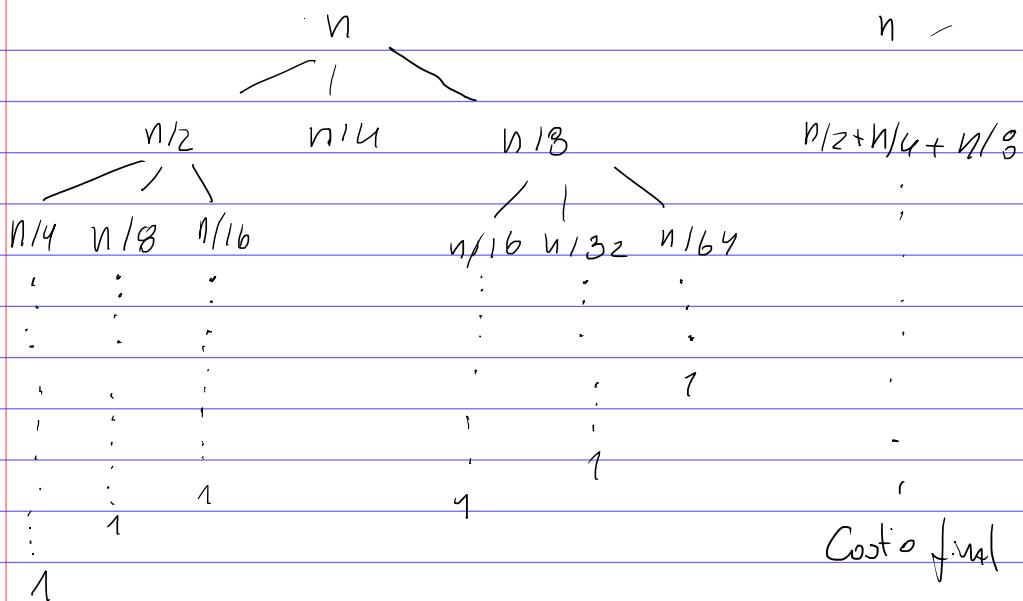
Entonces podemos decir que para  $C=1$

$$T(n) \in \Theta(2n^2 - n)$$

$$b) T(n) = T(n/2) + T(n/4) + n$$

Tamaño

Costo



Como para la aproximación solo nos interesa la cota inferior y superior, al sacarlos la rama donde solo hay recursividad ( $n/2$ ) y la rama donde solo hay recursividad ( $n/3$ ).

En la rama  $T(n/2)$  se llega al caso  $T(1)$  en el nivel  $k$  tal que:

$$T(1) = 1 = n/2^k \Rightarrow n = 2^k \Rightarrow k = \log_2(n)$$

Para la rama  $T(n/3)$  se llega al caso  $T(1)$  en el nivel  $k$  tal que:

$$T(1) = 1 = n/3^k \Rightarrow n = 3^k \Rightarrow k = \log_3(n)$$

Como  $\log_2(n) \geq \log_3(n)$ , para todo  $n > 0$

Así el costo Aproximado por nivel es  $n$

Entonces la cota superior sera  $n * \log_2(n) = \mathcal{O}(n \log(n))$

y la cota inferior sera  $n * \log_3(n) = \Omega(n \log(n))$

Ahora demostraremos por sustitución

Para la constante superior tenemos que:

Hipótesis:  $T(n) \leq (n \log(n))c$  para todo  $n \geq 0$

$$T(n) = T(n/2) + T(n/4) + T(n/8) + n \leq cn \log(n)$$

$$(c(n/2) \log(n/2)) + (c(n/4) \log(n/4)) + (c(n/8) \log(n/8)) + n$$

$$\Rightarrow \frac{c}{8} (4 \log(n/2) + 2 \log(n/4) + \log(n/8) + n) \leq cn \log(n)$$

$$\Rightarrow \frac{c}{8} (4 \log(n/2) + 2 \log(n/4) + \log(n/8) + 1) \leq c \log(n)$$

Por propiedades de:

$$\log(n/2) = \log(n) - \log(2) = \log(n) - 1$$

$$\log(n/4) = \log(n) - \log(4) = \log(n) - 2$$

$$\log(n/8) = \log(n) - \log(8) = \log(n) - 3$$

$$\Rightarrow \frac{c}{8} (4 \log(n) - 4 + 2 \log(n) - 4 + \log(n) - 3) + 1 \leq c \log(n)$$

$$\Rightarrow \frac{c}{8} (7 \log(n) - 11) + 1 \leq c \log(n)$$

$$\Rightarrow \frac{7}{8} c \log(n) - \frac{11c}{8} + 1 \leq c \log(n)$$

$$\Rightarrow 1 - 11c/8 \leq \frac{1}{8} c \log(n) / .8$$

$$\Rightarrow 8 - 11c \leq c \log(n)$$

$$\Rightarrow 8 \leq c \log(n) + 11c$$

Como  $\log(n)$  es positivo para todo  $n \geq 1$ , es suficiente decir que  $c \geq 8/11$  para que se cumpla

Para la cota inferior:

Hipótesis:  $T(n) \geq (n \log(n))_C$  para todos  $n > 0$

$$T(n) = T(n/2) + T(n/4) + T(n/8) + n \geq cn \log(n)$$

$$(c(n/2) \log(n/2)) + (c(n/4) \log(n/4)) + (c(n/8) \log(n/8)) + n$$

$$\Rightarrow \frac{c}{8} (4 \log(n/2) + 2 \log(n/4) + \log(n/8) + n) \geq cn \log(n)$$

$$\Rightarrow \frac{c}{8} (4 \log(n/2) + 2 \log(n/4) + \log(n/8)) + 1 \geq c \log(n)$$

Por propiedades:

$$\log(n/2) = \log(n) - \log(2) = \log(n) - 1$$

$$\log(n/4) = \log(n) - \log(4) = \log(n) - 2$$

$$\log(n/8) = \log(n) - \log(8) = \log(n) - 3$$

$$\Rightarrow \frac{c}{8} (4 \log(n) - 4 + 2 \log(n) - 4 + \log(n) - 3) + 1 \geq c \log(n)$$

$$\Rightarrow \frac{c}{8} (7 \log(n) - 11) + 1 \geq c \log(n)$$

$$\Rightarrow \frac{7}{8} c \log(n) - \frac{11}{8} c + 1 \geq c \log(n)$$

$$\Rightarrow 1 - 11c/8 \geq \frac{1}{8} c \log(n) / \cdot 8$$

$$\Rightarrow 8 - 11c \geq c \log(n)$$

$$\Rightarrow 8 \geq c \log(n) + 11c$$

Como  $\log(n)$  es positivo para todo  $n \geq 1$ , tenemos que hacer

que sea negativo, para esto basta decir que  
 $c \leq -1$  para que se cumpla

6. [0.5 puntos] Use substitución para demostrar las siguientes recurrencias. Note que debe aplicar las definiciones asintóticas.

$$a) T(n) = T(n-1) + \log(n) \text{ su solución es } T(n) = \Theta(n \log(n))$$

Para demostrar esto debemos demostrar con superior y cota inferior:

Para la cota superior:

$$\text{Hipótesis: } T(n) \leq c(n \log(n)) + d \log(n), \forall n_0 \geq 1$$

$$T(n) = T(n-1) + \log(n) \leq c(n \log(n)) + d \log(n)$$

$$\Rightarrow c(n-1) \log(n-1) + d \log(n-1) + \log(n) \leq c(n \log(n)) + d \log(n) / -d \log(n)$$

$$\Rightarrow c(n-1) \log(n-1) + d \log(n-1) + (1-d) \log(n) \leq c(n \log(n))$$

Si escogemos  $d=1$ , tenemos:

$$\Rightarrow c(n-1) \log(n-1) + \cancel{\log(n-1)} + \cancel{d \log(n)} \leq c(n \log(n))$$

$$\Rightarrow c(n \log(n-1)) + (1-c) \log(n) \leq c(n \log(n))$$

Como  $\log(n)$  es una función creciente y positiva para todo  $n \geq 1$ , podemos escoger un  $c \geq 1$  que cumple con la desigualdad.

Así decimos que  $T(n) = \Theta(n \log(n))$ , para todo  $n_0 > 1$ ,  $c \geq 1$  y  $d=1$  (i)

• Agora para a cota inferior temos que:

$$\text{Hipótese } T(n) \geq C(n \log(n)) - d \log(n-1), \forall n \geq 2$$

$$T(n) = T(n-1) + \log(n) \geq C(n \log(n)) - d \log(n-1)$$

$$\Rightarrow C(n-1) \log(n-1) - d \log(n-2) + \log(n) \geq C(n \log(n)) - d \log(n-1) + d \log(n-1)$$

$$\Rightarrow Cn \log(n-1) + (d-1) \log(n-1) - d \log(n-2) + \log(n) \geq Cn \log(n)$$

$$b) T(n) = T(n-2) + 2\log(n) \text{ su soluciones } T(n) = O(n \log n)$$

$$\text{Hipótesis: } T(n) \leq c(n \log(n)) + d \log(n), \forall n \geq 1$$

$$T(n) = T(n-2) + 2\log(n) \leq c(n \log(n)) + d \log(n)$$

$$\Rightarrow (c(n-2)\log(n-2)) + d\log(n-2) + 2\log(n) \leq cn\log(n) + d\log(n) - d\log(n)$$

$$\Rightarrow (c(n-2)\log(n-2)) + d\log(n-2) + (2-d)\log(n) \leq cn\log(n)$$

Tomamos  $c=1$ ,  $d=2$ , tenemos que:

$$\Rightarrow (c(n-2)\log(n-2)) + 2\log(n-2) + \cancel{o(\log^0(n))} \leq cn\log(n)$$

$$\Rightarrow c(n\log(n-2) + 2\log(n-2) + 2\log(n-2)) \leq cn\log(n)$$

$$\Rightarrow cn\log(n-2) + (2-c)\log(n-2) \leq cn\log(n)$$

Como  $\log(n)$  es una función creciente y positiva para todo  $n \geq 2$ , podemos escoger un  $c \geq 2$  que cumple con la desigualdad.

Así decimos que  $T(n) = O(n \log n)$ , para  $n \geq 2$ ,  $c \geq 1$  y  $d = 2$ .

7. [0.5 punto] Proporcione un análisis asintótico de peor caso en notación  $O()$  para el tiempo de ejecución de los siguientes fragmentos de programa.

(a)

```
for( int i = 1; i <= n; i += 2 ) {
    for( int j1 = 0; j1 < i; j1++ ) {
        for( int k = 0; k < n; k += 2 ) {
            f(); // O(n)
        }
        for( int j2 = 1; j2 < n; j2 += 2 ) {
            g(); // O(1)
        }
    }
}
```

$$i \rightarrow n/2 \quad j_1 \rightarrow \sum_{i=1}^{n/2} \sum_{j_1=0}^i j_1 \quad K \rightarrow n/2 \cdot j_1$$

$$\Rightarrow j_1 = \sum_{i=1}^{n/2} \sum_{j_1=0}^i j_1 = \sum_{i=1}^{n/2} \frac{i(i+1)}{2} = \sum_{i=1}^{n/2} \frac{i^2}{2} + \sum_{i=1}^{n/2} \frac{i}{2}$$

$$\sum_{i=1}^{n/2} \frac{i}{2} = \frac{n/2(n/2+1)}{2} = \frac{\frac{n^2}{4} + \frac{n}{2}}{2} = \frac{n^2}{8} + \frac{n}{4} = \frac{n^2 + 2n}{8}$$

$$\sum_{i=1}^{n/2} \frac{i^2}{2} = \frac{n/2((n/2)+1)(n+1)}{6} = \frac{n^3 + 3n^2 + 2n}{24}$$

$$j_1 = \frac{n^2 + 2n}{8} + \frac{n^3 + 3n^2 + 2n}{24} = \frac{n^3 + 6n^2 + 8n}{24}$$

$$K = j_1 \cdot n/2 : \frac{n^3 + 6n^2 + 8n}{24} \cdot n/2 = \frac{n^4 + 6n^3 + 8n^2}{48}$$

Ahora la complejidad de estos 3 ciclos sobre  $f()$  es:

$$K \cdot n = \frac{n^4 + 6n^3 + 8n^2}{48} \cdot n = \frac{n^5 + 6n^4 + 8n^3}{48} \Rightarrow O(n^5)$$

$j \rightarrow \log(n) \cdot i$

$$\Rightarrow j = n/2 \log(n) \approx \frac{n \log(n)}{2}$$

Ahora la complejidad para estos 2 ciclos sobre  $g(i)$  es:

$$j \cdot i = \frac{n \log(n)}{2} \cdot i = \frac{n \log(n)}{2} \Rightarrow O(n \log(n))$$

finalmente la complejidad del fragmento de programa

c)

$$O(n^5) + O(n \log(n)) = O(n^5)$$

(b)

```
for ( i=1; i < n; i += 2 ) {
    for ( j = n; j > 0; j /= 2 ) {
        for ( k = j; k > 0; k /= 2 ) {
            sum += (i + j * k); \(\rightarrow O(n)\)
        }
    }
}
```

$$\hookrightarrow \log(n) \quad j \rightarrow \log(n) \cdot i = (\log(n))^2$$

$$k \rightarrow i \cdot j \cdot k = \log(n) \sum_{n=1}^{\log(n)+1} n = \log(n) \left[ \frac{\log(n)(\log(n)+1)}{2} \right]$$

$$\Rightarrow \log(n) \left[ \frac{(\log(n))^2 + (\log(n))}{2} \right] \cdot \left[ \frac{(\log(n))^3 + (\log(n))^2}{2} \right]$$

$$\Rightarrow \frac{(\log(n))^3 + (\log(n))^2}{2} = O((\log(n))^3)$$

Finalmente podemos decir que la complejidad de este fragmento de código es:

$$O((\log(n))^3) \cdot O(n) = O((\log(n))^3)$$

(c)

```
int F(int n) {
    for (i = 0; i < n; i += 2) {
        procesar(i); // O(n^2)
    }
    if (n <= 0)
        return 1; → O(1)
    else
        return F(n-5);
}
```

Tenemos que

$$i \rightarrow n/2$$

Así la complejidad del ciclo for

es de:

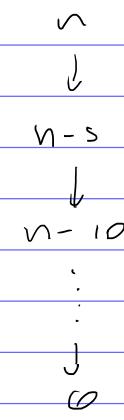
$$O(n^2) \cdot n/2 = O(n^3)$$

Ahora podemos hacer una ección de recursividad, tal que:

$$T(n) = T(n-5) + O(n^3)$$

Luego construimos un árbol de la forma:

Tamaño



Costo

$$O(n^3)$$

$$O(n^3)$$

$$O(n^3)$$

$$O_n \xrightarrow{\text{solo } n \neq 0} \text{costo } n=0$$

Sabemos que el costo sera:

$$(k-1) O(n^3) + O(1), \text{ para } k \text{ niveles}$$

$$\text{sabemos que } (n - 5(k)) = 0 \Rightarrow n = 5k \Rightarrow k = \frac{n}{5}$$

Ahora, podemos resolver el costo:

$$(\frac{n}{5} - 1) O(n^3) + O(1) = O(n^4) - O(n^3) + O(1)$$

Así, podemos decir que la complejidad del código fuente es aproximadamente  $O(n^4)$ .

(d)

```
int G(int x, int y){  
    int m = 1; → O(1)  
    if(y == 0){ → O(1)  
        return 1;  
    }  
    while(y > 0){ → O(log(y))  
        if(y & 1){  
            m *= x;  
        } → O(1)  
        x *= x; → O(1)  
        y = y >> 1; → O(1)  
    }  
    return m;  
}
```

El "if( $y \& 1$ )" en el peor caso se ejecutara  
log $y$  veces, ya que es formas limitado por el  
while ( $y > 0$ )

Asi, el fragmento de código tendría un complejidad

$$5 \log(y) + 1 = O(\log(y))$$

8. [0.5 punto] El siguiente código obtiene el año para el número de días a partir del año 1980. Analice el siguiente segmento de código y determine si es correcto o no.

siguiente segmento de código

```
int ObtieneA(int dias){  
    int a = 1980;  
    while( dias > 365){  
        if(agnobisiesto(a)){  
            if(dias > 366){  
                dias = dias - 366;  
                a = a + 1;  
            }  
        } else {  
            dias = dias - 365;  
            a = a + 1;  
        }  
    }  
    return a;  
}
```

En caso de que la función esté incorrecta, debería haber un contraejemplo que demuestre que está incorrecta, entonces probemos:

Tomamos un input: 366

Primero tenemos que saber que 1980 sí es un año bisiesto, por lo que el método Agnobisiesto dará true

• Primero definimos  $a = 1980$

• Entramos al ciclo while, ya que nuestro input es mayor a 365

• Entramos a la condición if de Agnobisiesto

• No se entra a la condición if, por que 366 no es mayor estricto que 365

• Como no hay un else se itera nuevamente el while manteniendo el valor de a

• Daure un loop infinito dentro del while

Como pudimos analizar el caso de input 366 es un contraejemplo a la correctitud de la función. Por ende la función Obtiene A no es correcta

9. [0.5 punto] Determine que computa la función  $G$ , y si es correcta o no. Determine además su complejidad asintótica.

```
int G(a,b)
    x=a, y=b, z = 0
    while( x > 0)
        if (x & 1)
            z = z + y
        x = x>>1
        y = y<<1
    return z
```

Luego de analizar el código dado, llegamos a la conclusión que la función multiplica  $a \cdot b$  y retorna la multiplicación en la variable  $z$ .

Primero escogemos un loop invariante:

- El loop invariante es:

$$Z + (x \cdot y) = (a \cdot b)$$

Cuando  $x$  es impar, tenemos que:

$$Z_k = Z_{k-1} + y_{k-1}$$

Ahora, cuando  $x$  es par:

$$Z_k = Z_{k-1}$$

↑

Demostramos la inicialización

$$Z=0 ; x=a ; y=b$$

$$\Rightarrow 0 + (a \cdot b) = a \cdot b \Rightarrow a \cdot b = a \cdot b \text{, es correcto}$$

Demos la forma Matemática:

$$x_k = \frac{(x_{k-1}) - 1}{2} ; y_k = (y_{k-1}) 2$$

$$\Rightarrow z_k + (x_k \cdot y_k) = a \cdot b$$

$$\Rightarrow z_{k-1} + y_{k-1} + \left[ \left( \frac{(x_{k-1}) - 1}{2} \right) \cdot 2 y_{k-1} \right] = a \cdot b$$

$$\Rightarrow z_{k-1} + y_{k-1} + (x_{k-1} \cdot y_{k-1}) - y_{k-1} = a \cdot b$$

$$\Rightarrow z_{k-1} + (x_{k-1} \cdot y_{k-1}) = a \cdot b // \text{ Se cumple para la}$$

iteración anterior

Demos la finalización

Cuando  $x_k = p$

$$\Rightarrow z_k + (x_k \cdot y_k) = a \cdot b$$

$$\Rightarrow z_k + (0 \cdot y_k) = a \cdot b$$

$\Rightarrow z_k = a \cdot b // \text{ se cumple que } z \text{ finalmente}$   
 $\text{es la multiplicación de } a \text{ por } b$

• Como cumple con los 3 requisitos anteriores,  
podemos decir que la función es correcta

### • Análisis asintótico

Para el ciclo while de la función  $G$ , este se repite  
log( $a$ ) veces y el "if ( $x & 1$ )" se itera una vez.  
log( $a$ ) veces, todo lo demás tendría una  
complejidad de  $O(1)$ , por ende tenemos una  
complejidad de

$$5 \log(a) + 3 = O(\log(a))$$

10. [1.5 puntos] Considere un arreglo de enteros  $A$  que contiene números enteros positivos distintos.
- Asuma que  $A$  contiene elementos en el rango  $[a..n]$  donde  $1 \leq a \leq n$ . Proporcione un algoritmo que dada una entrada que incluye solo la suma total de los elementos en  $A$ , el valor  $n$  y el valor  $a$ , el algoritmo entrega como salida: verdadero si falta un elemento, o hay un elemento repetido en  $A$ , o falso si no es así.
  - Ahora asuma que  $A$  contiene  $n$  elementos en el rango  $[1..n+1]$  y no contiene un elemento. Proporcione el mejor algoritmo para encontrar el elemento faltante si  $A$  está ordenado, obtenga su recursividad y obtenga su solución.
  - Asumiendo caso (b). Proporcione un algoritmo en  $O(n)$  para el caso en que el arreglo  $A$  no esté ordenado. Puede usar espacio extra si lo desea.
  - Implemente los algoritmos y realice una evaluación experimental.

a)  $\text{Vocol A(int sum, int n, int a)}$

```

a = a - 1;
int aux1 = (n(n+1))/2;
int aux2 = (a(a+1))/2;
if (sum != aux1 - aux2) {
    cout << "verdadero" << endl;
} else {
    cout << "falso" << endl;
}
}

```

b)  $\text{Int B (int *arr, int n)}$

```

for (int i=0; i < n-1; i++) {
    if (arr[i] != i+1)
        return i+1;
}

```

```

return -1;
}

```

Si analizamos el código podemos decir que su recurrencia es:

$$T(n) = T(n-1) + O(n)$$

Resolvemos por arbol Recursivo

Tamaño	Costo
n	1
n-1	1
n-2	1
:	:
1	1

$$\Rightarrow T(n-1) \text{ llega a } T(1) \text{ en el nivel } k \Rightarrow n-k=1 \Rightarrow k=n-1$$

• Costo total:  $\sum_{i=1}^n 1 = n-1$

Como solo tiene una rama entonces la recurrencia es  $\Theta(n)$

c) int C (int \*Avv, int n) {  
    int sum = 0, total;  
    total = ((n+1)\*(n+2))/2;  
    for (int i = 0; i < n; i++)  
        sum += Avv[i];  
    }  
    return total - sum;

d)

n;	Tiempo A;	Tiempo B;	Tiempo C
1000;	240;	4474;	3436
2000;	67;	8737;	6518
3000;	60;	13052;	9726
4000;	58;	22016;	12966
5000;	95;	21724;	16278
6000;	58;	25998;	19395
7000;	65;	35078;	22797
8000;	64;	34731;	26087
9000;	63;	38962;	29323
10000;	64;	43316;	32414