# Análisis de Algoritmos
# Tarea 02: Complejidad

**Profesora:** María de Luz Gasca Soto
**Ayudantes:** Rodrigo Fernando Velázquez Cruz
Teresa Becerril Torres
**Nombre**: Alvaro Ramirez Lopez
**N°. de Cuenta.**: 316276355
**Correo**: alvaro@ciencias.unam.mx

## 1. Problema 1

Sea $\Pi$ un problema. El desempeño computacional en el peor de los casos para $\Pi$ es $O(n^2)$ y también es $\Omega(n \log_2 n)$.

Sea $A$ un algoritmo que soluciona $\Pi$. ¿Cuales de las siguientes afirmaciones resultan consistentes con la información sobre $\Pi$?

**Justifique su respuesta**

a) $A$ tiene en el peor caso complejidad $O(n^3)$.
b) $A$ tiene en el peor caso complejidad $O(n)$.
c) $A$ tiene en el peor caso complejidad $\Theta(n \log n)$.
d) $A$ tiene en el peor caso complejidad $\Theta(n^2)$.

**Solución:**

a) $A$ tiene en el peor caso complejidad $O(n^3)$.

Esto es cierto, porque el peor caso descrito por el problema es $O(n^2)$, y se sabe que $n^2 \in O(n^3)$

$$\forall n \geq 1 \mid n^2 \leq 1 \cdot n^3$$

a) $A$ tiene en el peor caso complejidad $O(n)$.

Esto es falso, porque el peor caso descrito por el problema es $O(n^2)$, y como O es una cota superior, $n$ no esta en $O(n^2)$.

$$\forall n \geq 1 \mid n \leq n^2$$

a) $A$ tiene en el peor caso complejidad $\Theta(n \log n)$.

Esto es falso, porque el peor caso descrito por el problema es $O(n^2)$, y como $O$ es una cota superior, $n \log n$ no esta en $O(n^2)$, aunque se cumpla que $n \log n$ si pertenece a la cota inferior de $\Omega(n \log_2 n)$.

$$\Theta(n \log n) = \cancel{O(n \log n)} \cap \Omega(n \log n)$$

a) $A$ tiene en el peor caso complejidad $\Theta(n^2)$.

Igual esto es falso, dado a que se cumple que $n^2 \in O(n^2)$, pero no se cumple que $n^2 \in \Omega(n \log_2 n)$.

$$\Theta(n^2) = O(n^2) \cap \cancel{\Omega(n \log n)}$$

# 2. Problema 2

Supongamos que un algoritmo $A$ se ejecuta en el peor de los casos con tiempo $f(n)$ y el algoritmo $B$ toma tiempo $g(n)$, en el peor caso.

Responda las siguientes preguntas con **si, no** o **tal vez** y **justifica formalmente tu respuesta.**

¿Es $B$ mas rápido que $A$, para toda $n$ mayor que alguna $n_0$ ...

a) ... si $g(n) \in \Omega(f(n) \log n)$ ?
b) ... si $g(n) \in \Theta(f(n) \log n)$ ?

**Solución:**

a) **¿Es $B$ mas rápido que $A$, para toda $n$ mayor que alguna $n_0$ si $g(n) \in \Omega(f(n) \log n)$?**

Sí, $B$ es más rápido que $A$ para toda $n$ mayor que alguna $n_0$ si $g(n) \in \Omega(f(n) \log n)$.

**Justificación:**

Según la notación asintótica, $g(n) \in \Omega(f(n) \log n)$ implica que $g(n)$ crece al menos tan rápido como $f(n) \log n$ para valores suficientemente grandes de $n$.

Si $g(n) \in \Omega(f(n) \log n)$, significa que existe una constante positiva $c$ y un valor $n_1$ tal que para todos los $n$ mayores que $n_1$, se cumple que $g(n) \geq c \cdot f(n) \log n$.

Ahora, si consideramos $n$ mayor que $n_1$, podemos comparar los tiempos de ejecución en el peor caso:

- Para el algoritmo $A$: $T_{A(n)} \leq f(n)$
- Para el algoritmo $B$: $T_{B(n)} \leq g(n)$

Usando la relación de $g(n) \geq c \cdot f(n) \log n$, podemos escribir:

$$g(n) \geq c \cdot f(n) \log n$$
$$\Rightarrow T_B(n) \leq c \cdot f(n) \log n$$
$$\Rightarrow T_B(n) \leq f(n) \cdot (c \cdot \log n)$$

Dado que $c \cdot \log n$ crece más lentamente que $f(n)$, para valores suficientemente grandes de $n$, $B$ será más rápido que $A$.

En resumen, si $g(n) \in \Omega(f(n) \log n)$ y si $n$ es mayor que algún valor $n_0$ (que podría ser el máximo entre $n_1$ y otro valor), entonces $B$ será más rápido que $A$ para esos valores de $n$.

b) **¿Es $B$ mas rápido que $A$, para toda $n$ mayor que alguna $n_0$ si $g(n) \in \Theta(f(n) \log n)$?**

Sí, $B$ es más rápido que $A$ para toda $n$ mayor que alguna $n_0$ si $g(n) \in \Theta(f(n) \log n)$.

Justificación:

Cuando $g(n) \in \Theta(f(n) \log n)$, esto implica que hay dos constantes positivas $c_1$ y $c_2$ tales que para valores suficientemente grandes de $n$, se cumple:

$$c_1 \cdot f(n) \log n \leq g(n) \leq c_2 \cdot f(n) \log n$$

Ahora, para comparar los tiempos de ejecución en el peor caso de los algoritmos $A$ y $B$:

Algoritmo $A$: $T_{A(n)} \leq f(n)$ Algoritmo $B$: $T_{B(n)} \leq g(n)$ Usando la relación $c_1 \cdot f(n) \log n \leq g(n)$, podemos escribir:

$$T_B(n) \leq c_2 \cdot f(n) \log n$$

Dado que $c_2 \cdot f(n) \log n$ crece más lentamente que $f(n)$, para valores suficientemente grandes de $n$, $B$ será más rápido que $A$.

En resumen, si $g(n) \in \Theta(f(n) \log n)$ y si $n$ es mayor que algún valor $n_0$ (que podría ser el valor donde la desigualdad de las constantes $c_1$ y $c_2$ comienza a mantenerse), entonces $B$ será más rápido que $A$ para esos valores de $n$.

# 3. Problema 3

Considera los siguientes ciclos anidados:

```
...
  i <-- n;
  while i > 0 do
  j <-- i;
  while not (j > n)
  <cuerpo del repeat> // requiere O(1)
  j <-- j * 2
  end_w
  i <-- i / 2
  end_w
...
```

a) Determina el desempeño computacional $T(n)$ de los ciclos anidados.
b) Si en el código anterior cambiamos la asignación

$i \leftarrow \frac{i}{2}$ por $i \leftarrow i$ div 2

¿Cual seria el desempeño computacional $T(n)$ del proceso? **Justifica**

Para facilitar las operaciones aritméticas, en ambos incisos, puedes suponer que $n$ es potencia de 2.

**Solución:**

**a) Determina el desempeño computacional $T(n)$ de los ciclos anidados.**

Dado que podemos suponer que n es una potencia de 2, la relación entre i y n será i = n. Además, sabemos que en cada iteración del bucle exterior (while i > 0), i se divide por 2 y j se duplica en cada iteración del bucle interno (while not (j > n)). Esto significa que ambos bucles se ejecutarán log_2(n) veces.

Dentro del bucle interno, el "cuerpo del repeat" requiere O(1) operaciones, lo que significa que su tiempo de ejecución es constante, independientemente del valor de n o i.

Entonces, el tiempo total de ejecución del código se puede calcular como el producto del número de iteraciones de ambos bucles y el tiempo de ejecución del "cuerpo del repeat" en cada iteración:

Tiempo total = $\log_2(n) \cdot \log_2(n) \cdot O(1) = O(\log^2(n))$

Por lo tanto, el desempeño computacional de este pseudocódigo es O(log^2(n)), lo que significa que el tiempo de ejecución aumenta de manera cuadrática en relación con el logaritmo de n.

**b) Si cambiamos la asignación de $i \leftarrow \frac{i}{2}$ por $i \leftarrow i$ div 2, ¿Cual seria el desempeño computacional $T(n)$ del proceso?**

T(n) seguiría siendo el mismo, si reemplazamos la asignación i = i/2 por i = i div 2, donde div representa la división entera (es decir, una división que redondea hacia abajo al número entero más cercano), el desempeño computacional del proceso se mantiene esencialmente igual, ya que la cantidad de iteraciones no cambia y el "cuerpo del repeat" sigue ejecutándose O(1) veces en cada iteración.

Dado que los bucles exteriores e interiores todavía se ejecutan log_2(n) veces cada uno, y el "cuerpo del repeat" sigue requiriendo O(1) operaciones en cada iteración, el tiempo total de ejecución seguirá siendo:

$T(n) = \log_2(n) * \log_2(n) * O(1) = O(\log^2(n))$

Por lo tanto, el desempeño computacional del proceso no cambia, sigue siendo O(log^2(n)) incluso con la asignación modificada.

# 4. Problema 4

Proporciona un algoritmo (código) cuyo desempeño computacional sea $\Theta(n^3 \log n)$. Debes usar operaciones básicas, **no** debes usar procesos.

**Justifica** formalmente que tu algoritmo alcanza el tiempo pedido.

**Solución:**

```
...
  i <-- n;
  j <-- n;
  k <-- n;
  t <-- 9;
  arr[] <-- length(arr)=10;
  while i > 0 do
    while j > 0 do
      while k > 0 do
        arr[t] <-- t mod 2;
        t <-- t div 2;
      end_w
      j <-- j / 2
    end_w
    i <-- i / 2
  end_w
...
```

**Justificación**

El algoritmo descrito anteriormente tiene un tiempo de ejecución en el peor caso de $T(n) = \Theta(n^3 \log n)$. Para que esto sea cierto, se debe cumplir que: T(n) pertenece a $O(n^3 \log n) \cap \Omega(n^3 \log n)$.

**Primero vamos a tratar el caso para $O(n^3 \log n)$:**

Aquí tenemos 3 ciclos while que van desde $n$ hasta 1, y en cada uno de ellos se realizan operaciones de tiempo constante. Por lo tanto, el tiempo de ejecución en el peor caso es de tiempo $O(n^3)$.

Ahora, en la parte mas interna del ciclo while, se realiza una operación que divide a la mitad el valor de $t$ en cada iteración. Esto significa que la operación interna que hace el arreglo se ejecutará $\log n$ veces. Por lo tanto, el tiempo de ejecución en el peor caso es de tiempo $O(\log n)$ solo para esa parte del código.

Por lo tanto, si tomamos el tiempo de ejecución de los ciclos while con el tiempo de ejecución del arreglo nos dice que el algoritmo si pertenece a $O(n^3 \log n)$.

**Ahora vamos a tratar el caso para $\Omega(n^3 \log n)$:**

Aquí tenemos 3 ciclos while que van desde $n$ hasta 1 como lo vimos anteriormente, entonces podemos determinar que el tiempo de ejecución pertenece a $\Omega(n^3)$, ya que por lo menos toma $n^3$ de tiempo de ejecución.

Ahora, en la parte mas interna del ciclo while, se realiza una operación que divide a la mitad el valor de $t$ en cada iteración. Para que pertenezca a $\Omega(\log n)$ el tiempo de ejecución del arreglo debe ser mayor o igual a $\log n$, entonces si pertenece a $\Omega(\log n)$ y por lo tanto pertenece $\Omega(n^3 \log n)$.

Como podemos observar, el algoritmo cumple con las condiciones para pertenecer a $O(n^3 \log n)$ y $\Omega(n^3 \log n)$, por lo tanto, el algoritmo pertenece a $\Theta(n^3 \log n)$.

# 5. Problema 5

Considera las siguientes funciones de complejidad:

$n^2 \log 3^n$

$n^{2.75}$

$3^2 \cdot n!$

$\frac{n^2}{\log n}$

$(3n^2 - 2n + 8)^4$

$(7n - 1)!$

$4^n$

$2^{2n} \cdot n^{2n}$

$n^{\log n}$

$6n^3 - 3n^2 + 2^{n+2}$

Usando la definición formal de $O, \Omega, \Theta, o$ y $\omega$ asi como las relaciones estar contenido, $\subset$, y ser igual, $=$, **ordenar** las funciones de complejidad dadas en términos de $O, \Theta$ y $\Omega$.

**Solución:**

**Funciones ordenadas en términos de $0(f(n))$, es de la forma $g(n) \leq c \cdot f(n)$**

Las funciones que pertenecen a $O(n^{\log n})$ son:

$n^{\log n}$

Las funciones que pertenecen a $O(n)$ son:

$n^{\log n}$

$4^n$

$2^{2n} \cdot n^{2n}$

Las funciones que pertenecen a $O(n^2)$ son:

$n^{\log n}$

$4^n$

$2^{2n} \cdot n^{2n}$

$n^2 \log 3^n$

$n^{2.75}$

$\frac{n^2}{\log n}$

Las funciones que pertenecen a $O(n^3)$ son:

$n^{\log n}$

$4^n$

$2^{2n} \cdot n^{2n}$

$n^2 \log 3^n$

$n^{2.75}$

$\frac{n^2}{\log n}$

$6n^3 - 3n^2 + 2^{n+2}$

Las funciones que pertenecen a $O(n^8)$ son:

$n^{\log n}$

$4^n$

$2^{2n} \cdot n^{2n}$

$n^2 \log 3^n$

$n^{2.75}$

$\frac{n^2}{\log n}$

$6n^3 - 3n^2 + 2^{n+2}$

$(3n^2 - 2n + 8)^4$

Las funciones que pertenecen a $O(n!)$ son:

$n^{\log n}$

$4^n$

$2^{2n} \cdot n^{2n}$

$n^2 \log 3^n$

$n^{2.75}$

$\frac{n^2}{\log n}$

$6n^3 - 3n^2 + 2^{n+2}$

$(3n^2 - 2n + 8)^4$

$3^2 \cdot n!$

$(7n - 1)!$

Estas funciones están contenidas de la siguiente forma

$O(n^{\log^n}) \subset O(n) \subset O(n^2) \subset O(n^3) \subset O(n^8) \subset O(n!)$.

**Funciones ordenadas en términos de $\Omega$, es de la forma $g(n) \geq c \cdot f(n)$**

Las funciones que pertenecen a $\Omega(n!)$ son:

$n^{\log n}$

$4^n$

$2^{2n} \cdot n^{2n}$

$n^2 \log 3^n$

$n^{2.75}$

$\frac{n^2}{\log n}$

$6n^3 - 3n^2 + 2^{n+2}$

$(3n^2 - 2n + 8)^4$

$3^2 \cdot n!$

$(7n - 1)!$

Las funciones que pertenecen a $\Omega(n^8)$ son:

$n^{\log n}$

$4^n$

$2^{2n} \cdot n^{2n}$

$n^2 \log 3^n$

$n^{2.75}$

$\frac{n^2}{\log n}$

$6n^3 - 3n^2 + 2^{n+2}$

$(3n^2 - 2n + 8)^4$

Las funciones que pertenecen a $\Omega(n^3)$ son:

$n^{\log n}$

$4^n$

$2^{2n} \cdot n^{2n}$

$n^2 \log 3^n$

$n^{2.75}$

$\frac{n^2}{\log n}$

$6n^3 - 3n^2 + 2^{n+2}$

Las funciones que pertenecen a $\Omega(n^2)$ son:

$n^{\log n}$

$4^n$

$2^{2n} \cdot n^{2n}$

$n^2 \log 3^n$

$n^{2.75}$

$\frac{n^2}{\log n}$

Las funciones que pertenecen a $\Omega(n)$ son:

$n^{\log n}$

$4^n$

$2^{2n} \cdot n^{2n}$

Las funciones que pertenecen a $\Omega(n^{\log n})$ son:

$n^{\log n}$

Estas funciones están contenidas de la siguiente forma

$\Omega(n^{\log^n}) \subset \Omega(n) \subset \Omega(n^2) \subset \Omega(n^3) \subset \Omega(n^8) \subset \Omega(n!)$.

**Funciones ordenadas en términos de Θ, es de la forma $O(f(n)) \cap \Omega(f(n))$**

Las funciones que pertenecen a $O(n^{\log n})$ son:

$n^{\log n}$

Las funciones que pertenecen a $O(n)$ son:

$4^n$

$2^{2n} \cdot n^{2n}$

Las funciones que pertenecen a $O(n^2)$ son:

$n^2 \log 3^n$

$n^{2.75}$

$\frac{n^2}{\log n}$

Las funciones que pertenecen a $O(n^3)$ son:

$6n^3 - 3n^2 + 2^{n+2}$

Las funciones que pertenecen a $O(n^8)$ son:

$(3n^2 - 2n + 8)^4$

Las funciones que pertenecen a $O(n!)$ son:

$3^2 \cdot n!$

$(7n - 1)!$

# 6. Problema 6

Opcional. Usando la definición de $O$ y $\Omega$, para los siguientes incisos, demuestra formalmente, si $g(n) \in O(f(n))$ o si $g(n) \in \Omega(f(n))$.

| - | $g(n)$ | $f(n)$ |
|---|---|---|
| a) | $2^n$ | $5^{\log n}$ |
| b) | $\frac{n^2}{\log n}$ | $n(\log n)^2$ |
| c) | $\log^3(n)$ | $n^{0.5}$ |
| d) | $n!$ | $2^n$ |
| e) | $3n$ | $2^n$ |
| f) | $\log 3^n$ | $\log 2^n$ |

**Solución:**

**Inciso a)**

**P.D** $2^n \in O(5^{\log n})$

**Demostración**

Queremos encontrar $c$ y $n_0$ tal que $2^n \leq c \cdot 5^{\log n}$ para todo $n \geq n_0$.

Tomemos el logaritmo en base 5 en ambos lados:

$\left(\log_5 2^n \leq \log_5\left(c \cdot 5^{\log n}\right)\right)$

$\left(n \cdot \log_5 2 \leq \log_5 c + \log_5 5^{\log n}\right)$

$(n \cdot \log_5 2 \leq \log_5 c + \log n \cdot \log_5 5)$ (usando la propiedad $\log_a b^c = c \cdot \log_a b$)

$(n \cdot \log_5 2 \leq \log_5 c + \log n)$ (ya que $\log_5 5 = 1$)

A partir de aquí, notamos que el término $\log_5 c$ es una constante, por lo que podemos escribir:

$(n \cdot \log_5 2 \leq C + \log n)$, donde $(C = \log_5 c)$.

Ahora, podemos ver que si elegimos $c = 2^{\log_5 2}$ y $n_0 = 1$, entonces $2^n \leq c \cdot 5^{\log n}$ para todo $n \geq n_0$, ya que $n \cdot \log_5 2 \leq C + \log n$ será cierto. Por lo tanto, podemos decir que $2^n \in \Omega\left(5^{\log n}\right)$.

**P.D** $2^n \in \Omega\left(5^{\log n}\right)$

**Demostración**

Queremos encontrar $c$ y $n_0$ tal que $2^n \geq c \cdot 5^{\log n}$ para todo $n \geq n_0$.

Tomemos el logaritmo en base 5 en ambos lados:

$\log_5 2^n \geq \log_5\left(c \cdot 5^{\log n}\right)$

$n \cdot \log_5 2 \geq \log_5 c + \log_5\left(5^{\log n}\right)$

$n \cdot \log_5 2 \geq \log_5 c + \log n \cdot \log_5 5$ (usando la propiedad $(\log_a b^c = c \cdot \log_a b)$

$n \cdot \log_5 2 \geq \log_5 c + \log n)$ (ya que $\log_5 5 = 1$)

A partir de aquí, notamos que el término $\log_5 c$ es una constante, por lo que podemos escribir:

$n \cdot \log_5 2 \geq C + \log n$, donde $C = \log_5 c$.

Ahora, podemos ver que si elegimos $c = 1$ y $n_0 = 1$, entonces $2^n \geq c \cdot 5^{\log n}$ para todo $n \geq n_0$, ya que $n \cdot \log_5 2 \geq C + \log n$ será cierto. Por lo tanto, podemos decir que $2^n \in \Omega\left(5^{\log n}\right)$.

En resumen, hemos demostrado que $2^n \in O\left(5^{\log n}\right)$ y $2^n \in \Omega\left(5^{\log n}\right)$.

**Inciso d)**

**P.D** $n! \in O(2^n)$

**Demostración**

Queremos encontrar $c$ y $n_0$ tal que $n! \leq c \cdot 2^n$ para todo $n \geq n_0$.

Dado que $n!$ y $2^n$ son funciones factoriales y exponenciales respectivamente, podemos observar que $n!$ siempre será menor que $2^n$ a partir de algún punto. Esto se debe a que $2^n$ crece exponencialmente más rápido que n!.

Podemos tomar $c = 1$ y n_0 = 4 para demostrar esto:

Para $n = 4$, tenemos $4! = 24$ y $2^4 = 16$, lo cual cumple $4! \leq 2^n$.

Para cualquier $n$ mayor o igual a 4, $2^n$ seguirá creciendo exponencialmente más rápido que $n!$, y, por lo tanto, $n!$ estará acotado superiormente por $2^n$.

**P.D** $n! \in \Omega(2^n)$

**Demostración**

Queremos encontrar $c$ y $n_0$ tal que $n! \geq c \cdot 2^n$ para todo $n \geq n_0$.

Dado que $n!$ es una función factorial que crece más rápido que $2^n$, podemos tomar $c = 1$ y $n_0 = 0$ para demostrar esto.

Para cualquier valor de $n$, $n!$ siempre será mayor o igual que $2^n$. Por lo tanto, $n! \in \Omega(2^n)$.

En resumen, hemos demostrado que $n!$ está acotado superior y inferiormente por 2^n. Esto significa que $n!$ y $2^n$ tienen una relación asintótica, pero la relación es más fuerte en el sentido de $\Omega$, ya que $n!$ crece más rápido que $2^n$.

**Inciso e)**

**P.D** $3n \in O(2^n)$

**Demostración**

Ahora, veamos si podemos demostrar alguna de estas relaciones para $3n$ y $2^n$.

Primero, consideremos la relación $3n \in O(2^n)$:

Queremos encontrar $c$ y $n_0$ tal que $3n \leq c \cdot 2^n$ para todo $n \geq n_0$.

Tomemos el logaritmo en base 2 en ambos lados de la desigualdad:

$\log_2 3n \leq \log_2(c \cdot 2^n)$

$\log_2 3n \leq \log_2 c + \log_2 2^n$

$\log_2 3n \leq \log_2 c + n$

A partir de aquí, notamos que el término $\log_2 c$ es una constante, por lo que podemos escribir:

$\log_2 3n \leq C + n$, donde $C = \log_2 c$.

Ahora, si elegimos $c = 1$ y $n_0 = 1$, entonces tendremos:

$\log_2 3n \leq 1 + n$

Esto significa que para cualquier $n \geq n_0 = 1$, $3n$ estará acotado superiormente por $2^n$. Por lo tanto, $3n \in O(2^n)$.

**P.D** $3n \in \Omega(2^n)$

**Demostración**

Queremos encontrar $c$ y $n_0$ tal que $3n \geq c \cdot 2^n$ para todo $n \geq n_0$.

Si tomamos $c = 1$ y $n_0 = 1$, tendremos:

$(3n \geq 1 \cdot 2^n)$

Esto es cierto para cualquier $n \geq n_0 = 1$, ya que $3n$ crece al menos tan rápido como $2^n$.

En resumen, hemos demostrado que $3n$ está acotado superiormente por $2^n$ (es decir, $3n \in O(2^n)$) y que 3n está acotado inferiormente por $2^n$ (es decir, $3n \in \Omega(2^n)$).

**Inciso f)**

**P.D** $\log 3^n \in O(\log(2^n))$

**Demostración**

Queremos encontrar $c$ y $n_0$ tal que $\log 3^n \leq c \cdot \log 2^n$ para todo $n \geq n_0$.

Utilicemos las propiedades de los logaritmos para simplificar la expresión:

$\log 3^n = n \cdot \log 3$

$\log 2^n = n \cdot \log 2$

La desigualdad se convierte en:

$(n \cdot \log 3 \leq c \cdot n \cdot \log 2)$

Podemos simplificar $n$ en ambos lados:

$\log 3 \leq c \cdot \log 2$

Ahora, podemos ver que esta desigualdad es cierta para cualquier $c$ positivo, ya que $\log 3$ y $\log 2$ son constantes positivas. Por lo tanto, no importa qué valores elijamos para $c$ y $n_0$, siempre será cierto que $(\log 3^n \leq c \cdot \log 2^n)$ para todo $n \geq n_0$.

**P.D** $\log 3^n \in \Omega(\log 2^n)$

**Demostración**

Queremos encontrar $c$ y $n_0$ tal que $\log 3^n \geq c \cdot \log 2^n$ para todo $n \geq n_0$.

Utilicemos las mismas simplificaciones que hicimos antes:

$\log 3^n = n \cdot \log 3$

$\log 2^n = n \cdot \log 2$

La desigualdad se convierte en:

$n \cdot \log 3 \geq c \cdot n \cdot \log 2$

Podemos simplificar $n$ en ambos lados:

$\log 3 \geq c \cdot \log 2$

Al igual que en el caso anterior, esta desigualdad es cierta para cualquier $c$ positivo, ya que $\log 3$ y $\log 2$ son constantes positivas.

En resumen, hemos demostrado que $\log 3^n$ está acotado superiormente por $\log 2^n$ (es decir, $\log 3^n \in O(\log 2^n)$) y que $\log 3^n$ está acotado inferiormente por $\log 2^n$ (es decir, $\log 3^n \in \Omega(\log 2^n)$). Esto significa que $\log 3^n$ y $\log 2^n$ tienen una relación de crecimiento logarítmico similar en términos asintóticos, y podemos decir que son asintóticamente equivalentes.