

Comparación de Órdenes de Crecimiento y Simulación de Costos

Javier Jhairt López Rojas

27 de enero de 2026

1. Introducción

La siguiente solución muestra una explicación exhaustiva de los comportamientos asintóticos de diferentes casos de comportamientos algorítmicos. El objetivo principal es crear un entendimiento comparativo de diversos escenarios hipotéticos de comportamiento computacional, de tal forma que se pueda implementar en escenarios reales.

2. Metodología

2.1. Supuestos

Los supuestos realizados para el desarrollo de esta práctica son los siguientes:

- Entrada de datos limitada: En un mundo real, algunos algoritmos con complejidad demasiado elevada una entrada de datos igual de masiva provoca daños en hardware, por lo que se decidió acotar dichos valores.
- Modelo de cómputo idealizado: Se asume que el hardware funciona de manera ideal, sin interrupciones, errores o fallos.

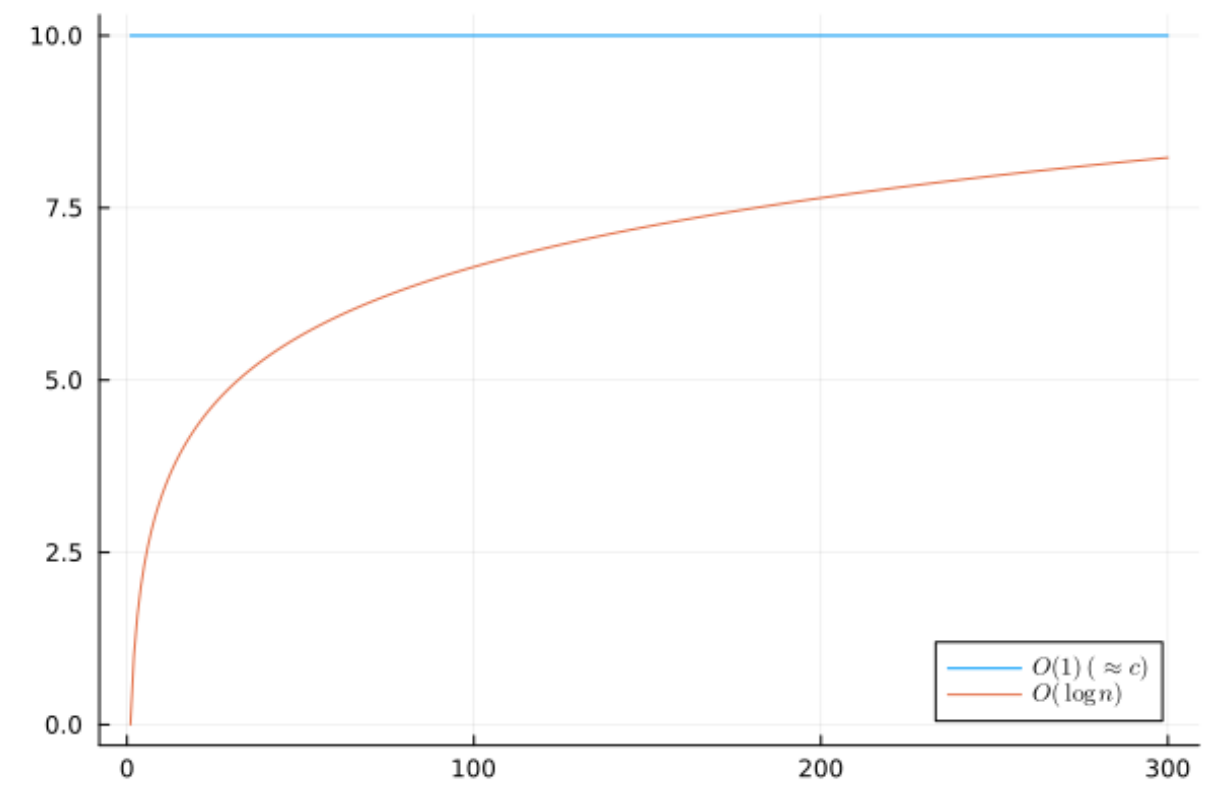
2.2. Herramientas y entorno

Los experimentos se realizaron en Julia (notebook Jupyter/Quarto). Se utilizaron funciones matemáticas y gráficas para comparar visualmente los crecimientos y generar una tabla de tiempos simulados.

3. Figuras y comparación de los órdenes de crecimiento

3.1. $O(1)$ vs $O(\log n)$

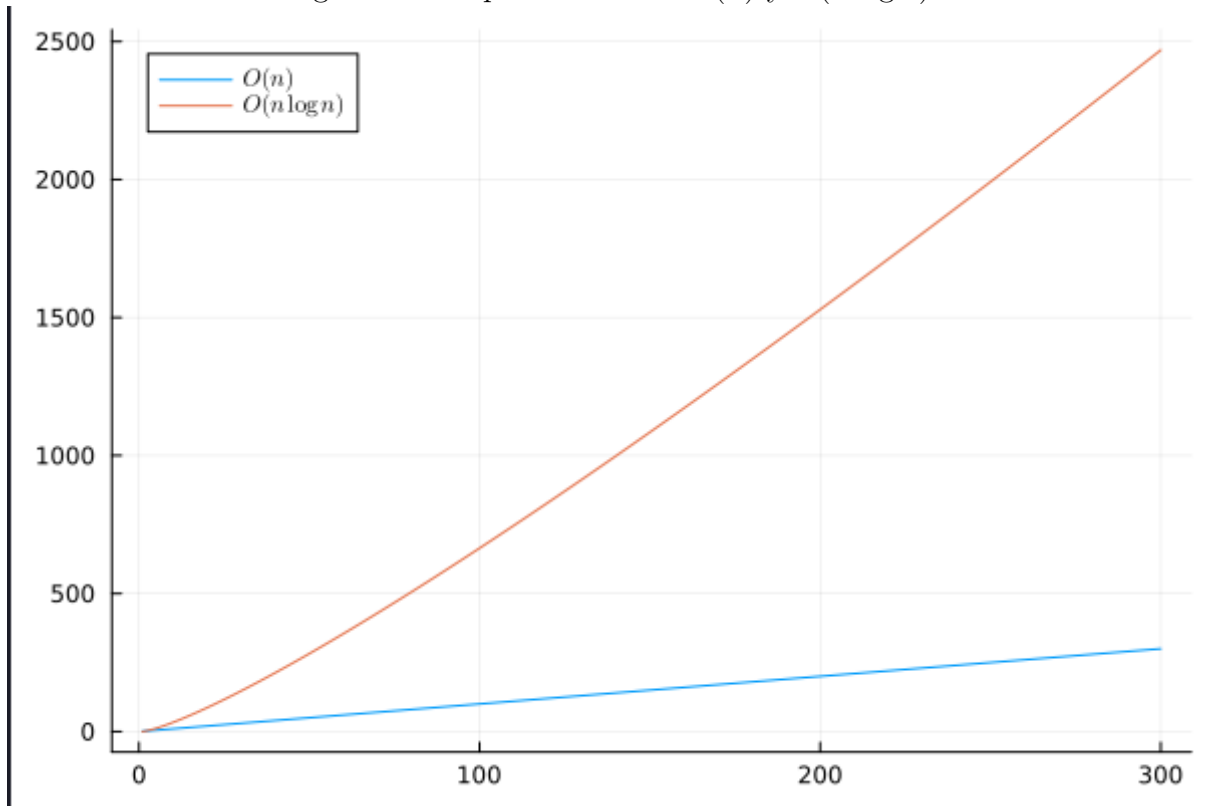
Figura 1: Comparación entre $O(1)$ y $O(\log n)$ en un rango seleccionado de n .



Discusión: El rango usado para esta comparación fue de $[0-300]$. Podemos notar que a medida que n crece el logaritmo cada vez más converge a un comportamiento más estable dado que la función logaritmo tiene la propiedad de llevar números muy grandes a cantidades más suaves.

3.2. $O(n)$ vs $O(n \log n)$

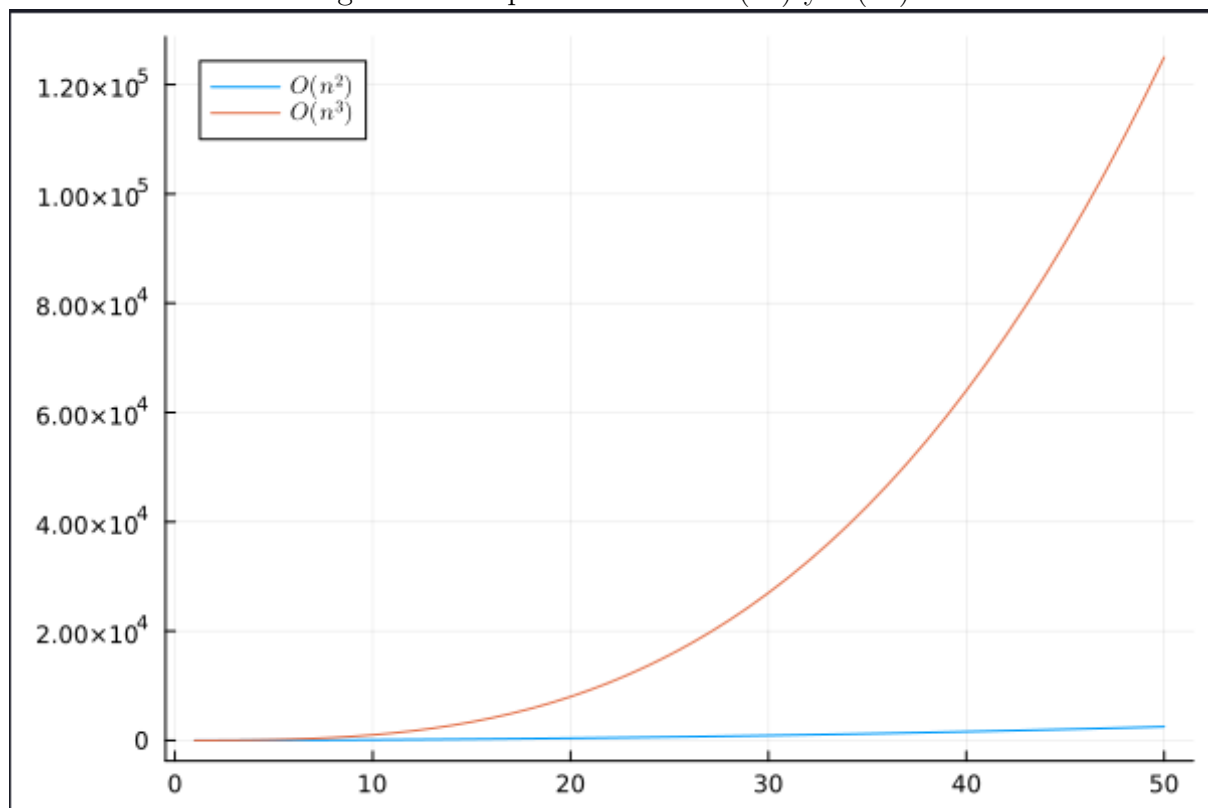
Figura 2: Comparación entre $O(n)$ y $O(n \log n)$.



Discusión: el rango usado fue de $[0-300]$. Se puede apreciar que a medida que n crece, el comportamiento asintótico de $n \log n$ es mucho más inmenso que un comportamiento lineal, esto porque a diferencia de la función \log , tiene un factor n que lo potencia a tener un mayor crecimiento.

3.3. $O(n^2)$ vs $O(n^3)$

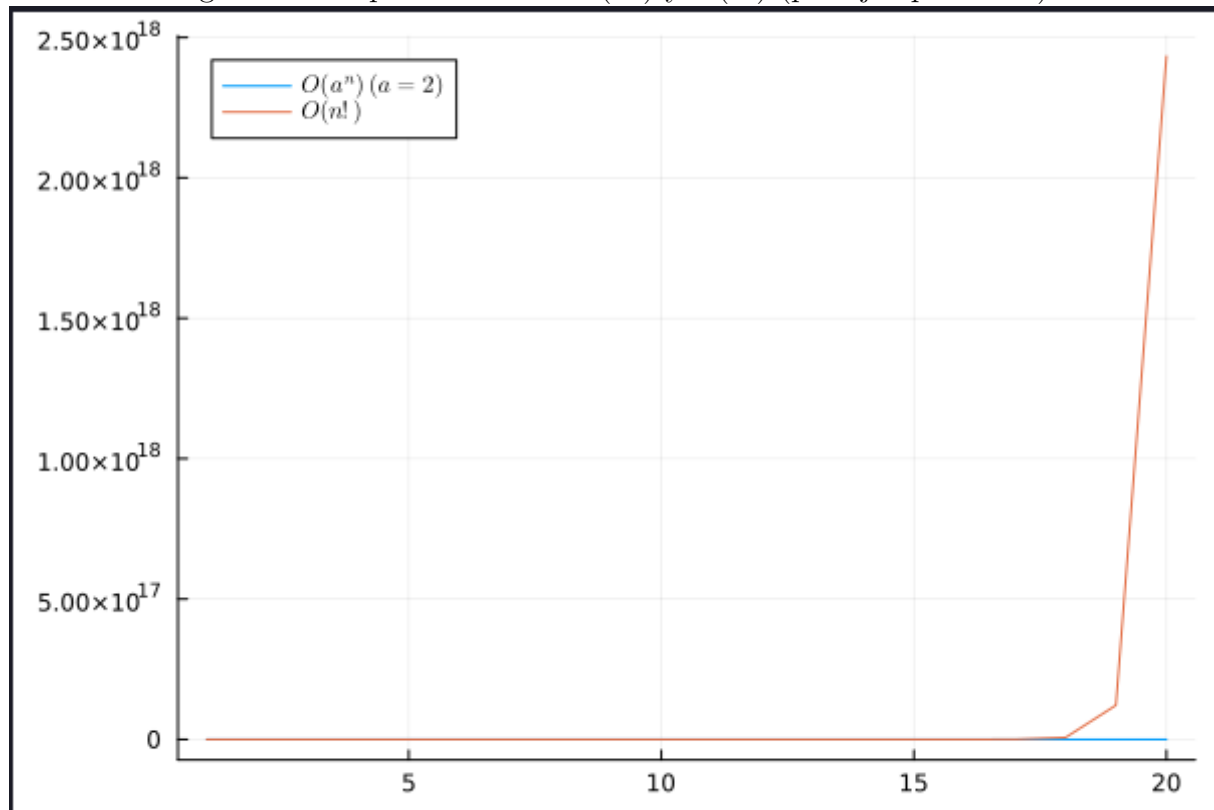
Figura 3: Comparación entre $O(n^2)$ y $O(n^3)$.



Discusión: Es interesante observar que la función n^2 puede verse “menos curvo” si n^3 domina el eje y , esto debido a que n^3 crece mucho más rápido que n^2 .

3.4. $O(a^n)$ vs $O(n!)$

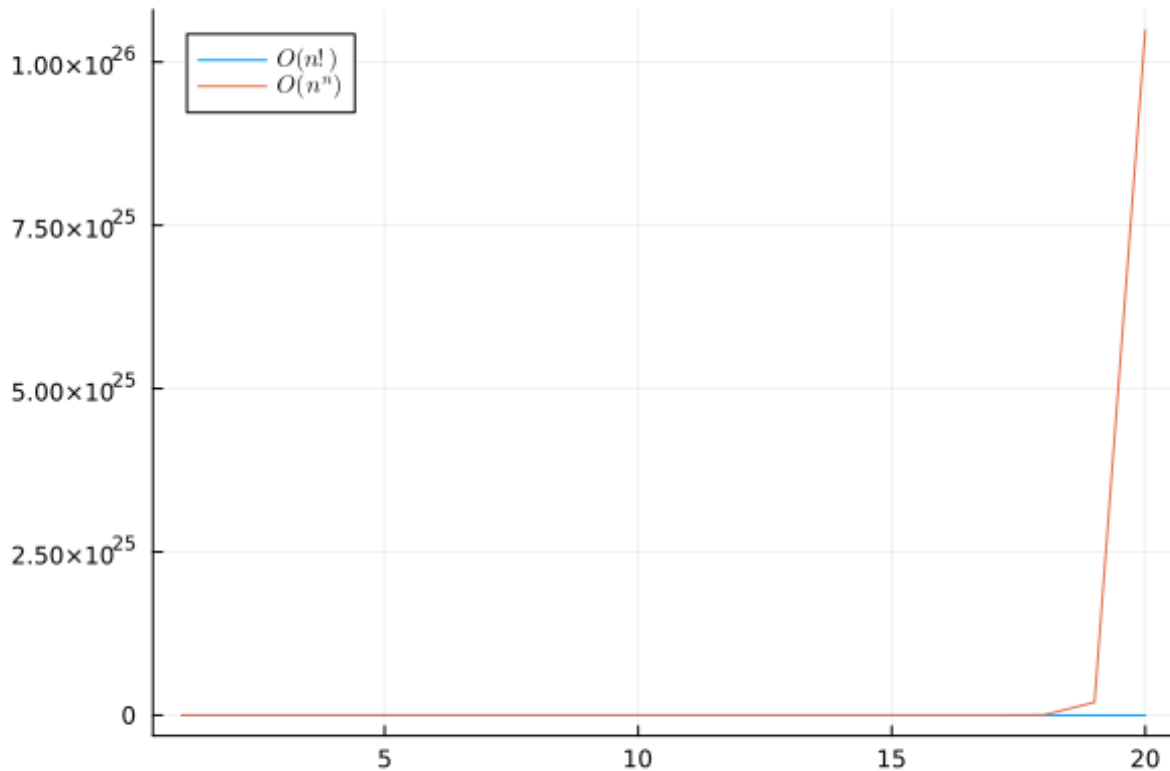
Figura 4: Comparación entre $O(a^n)$ y $O(n!)$ (por ejemplo $a = 2$).



Discusión: Debido a que ambos crecen de manera muy abrupta, se decidió usar un rango de $[0-20]$ para poder apreciar el comportamiento de ambas funciones.

3.5. $O(n!)$ vs $O(n^n)$

Figura 5: Comparación entre $O(n!)$ y $O(n^n)$.



Discusión: La función n^n crece demasiado rápido incluso con valores pequeños.

4. Análisis y simulación de costo en formato de tabla

4.1. Tabla de tiempos simulados

Se calcula el número de operaciones para distintos tamaños de entrada n y se convierte a tiempo suponiendo 1 ns por operación.

Cuadro 1: Simulación de costos computacionales (1 operación = 1 ns).

Orden	n	Operaciones (aprox.)	Tiempo (aprox.)
$O(1)$	10	1	1 ns
	100	1	1 ns
	1000	1	1 ns

Orden	n	Operaciones (aprox.)	Tiempo (aprox.)
	10000	1	1 ns
$O(\log n)$	10	3.32	3.32 ns
	100	6.64	6.64 ns
	1000	9.97	9.97 ns
	10000	13.29	13.29 ns
$O(\sqrt{n})$	10	3.16	3.16 ns
	100	10.0	10.0 ns
	1000	31.62	31.62 ns
	10000	100.0	100.0 ns
$O(n^2)$	10	100	100 ns
	100	10 000	10 μ s
	1000	1 000 000	1 ms
	10000	1.00×10^8	0.1 s
$O(2^n)$	10	1024	1.02 μ s
	20	1.05×10^6	1.05 ms
	30	1.07×10^9	1.07 s
	50	1.13×10^{15}	\approx 13.03 días
$O(n!)$	5	120	120 ns
	10	3.63×10^6	3.63 ms
	20	2.43×10^{18}	\approx 77.15 años
	50	MUY GRANDE	MUY GRANDE
$O(n^n)$	5	3125	3.13 μ s
	10	1.00×10^{10}	10 s
	20	1.05×10^{26}	\approx 3.33×10^9 años
	50	MUY GRANDE	MUY GRANDE
$O(n^{(n^n)})$	2	16	16 ns
	3	7.63×10^{12}	\approx 2.12 h
	4	MUY GRANDE	MUY GRANDE
	5	MUY GRANDE	MUY GRANDE

5. Conclusiones

Dados los resultados de las pruebas desarrolladas en esta practica, podemos observar distintos escenarios que nos pueden ayudar a detectar cuando un algoritmo es mas eficiente que otro con condiciones dadas, de esta forma podemos concluir que un factor importante para observar cambios en el comportamiento de un algoritmo es la entrada de datos, ya que en algunos intervalos finos se pueden observar comportamientos muy diferentes con respecto a su mismo comportamiento con numeros mas largos.

6. Referencias

- Gómez Fuentes, M. C., & Cervantes Ojeda, J. (2014). Análisis de la complejidad algorítmica. En Introducción al Análisis y al Diseño de Algoritmos (pp. 13-22). Universidad Autónoma Metropolitana.