Instituto Politécnico Nacional

Escuela Superior de Cómputo

Graficación de ordenes de complejidad

Unidad de aprendizaje: Analisis de algoritmos

Grupo: 3CM3

Alumno:
López Manríquez Ángel

M. en C.:
Franco Martinez Edgardo Adrian

27 de septiembre de 2018

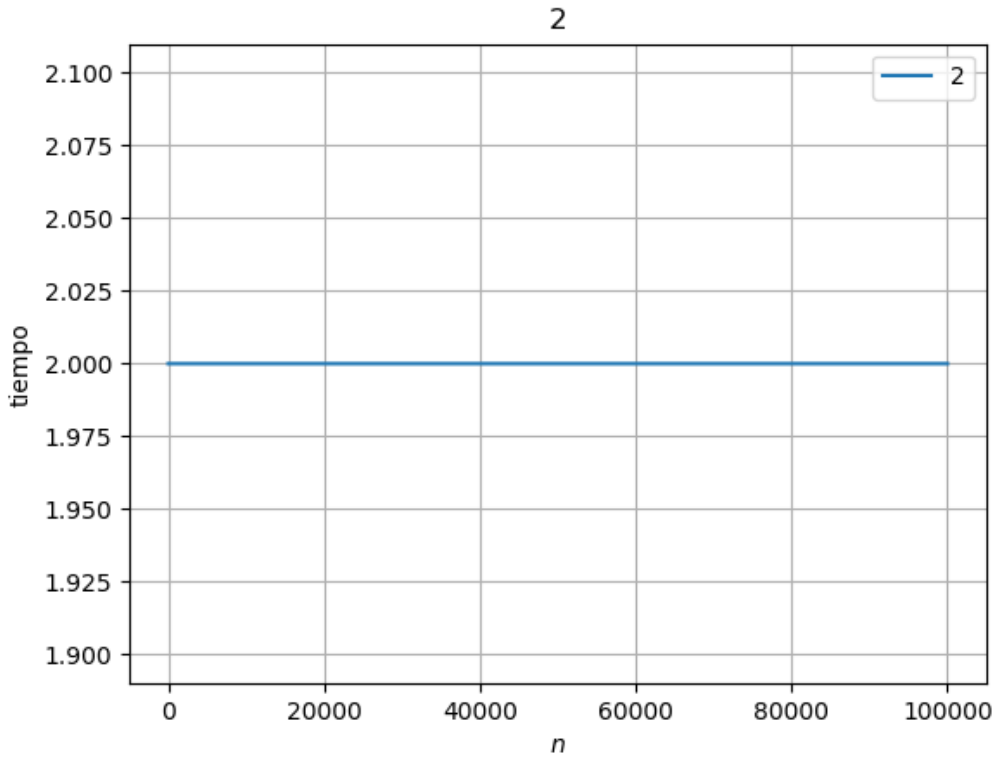
Índice

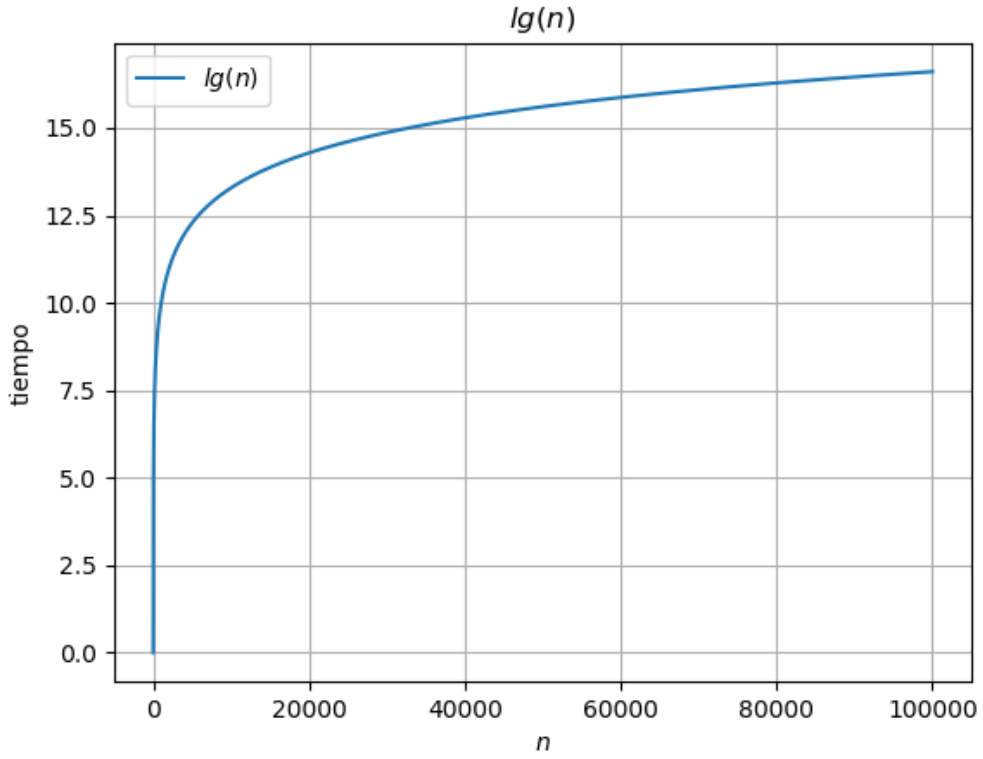
Graficaciónde ordenes de complejidad

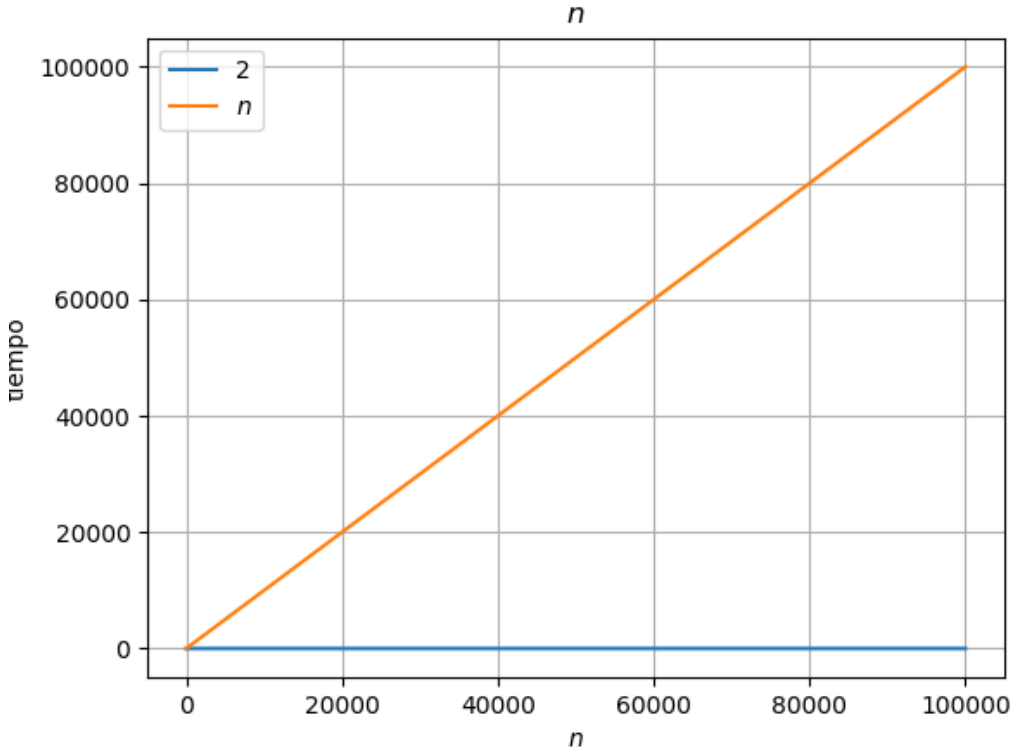
López Manríquez Ángel
3CM3

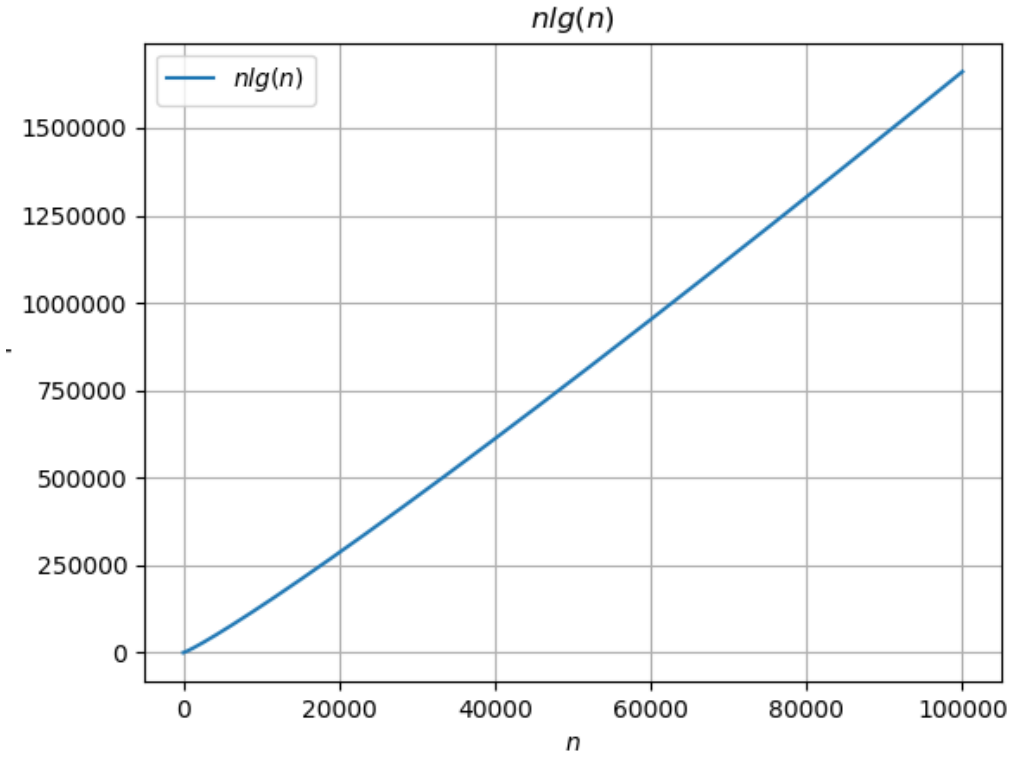
27 de septiembre de 2018

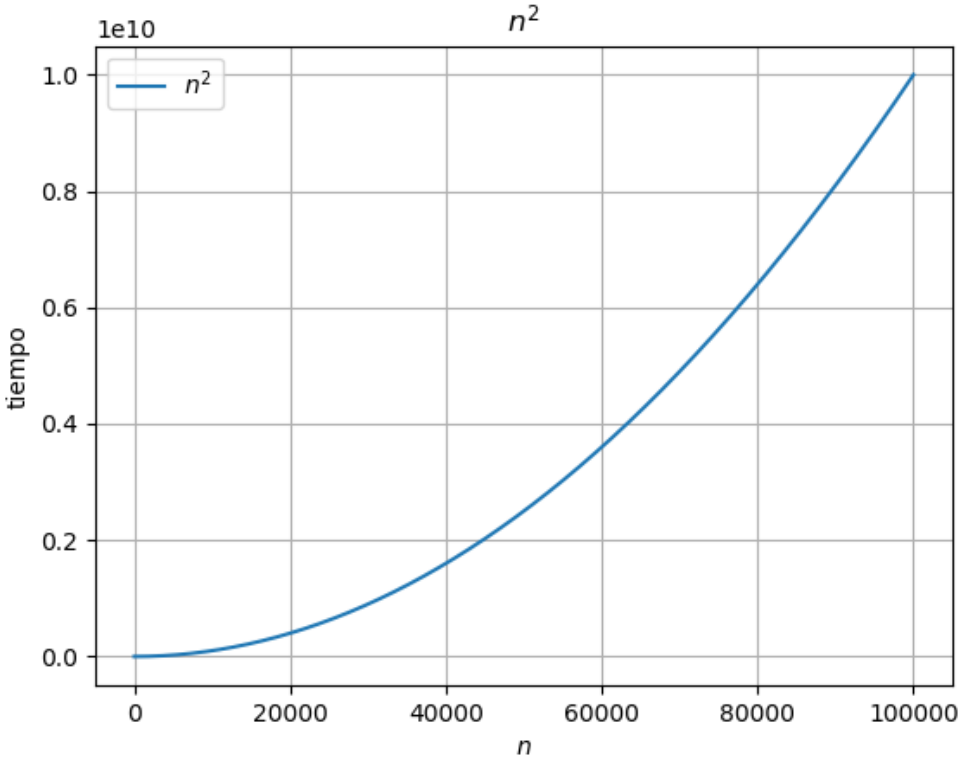
1. Graficas de manera separada

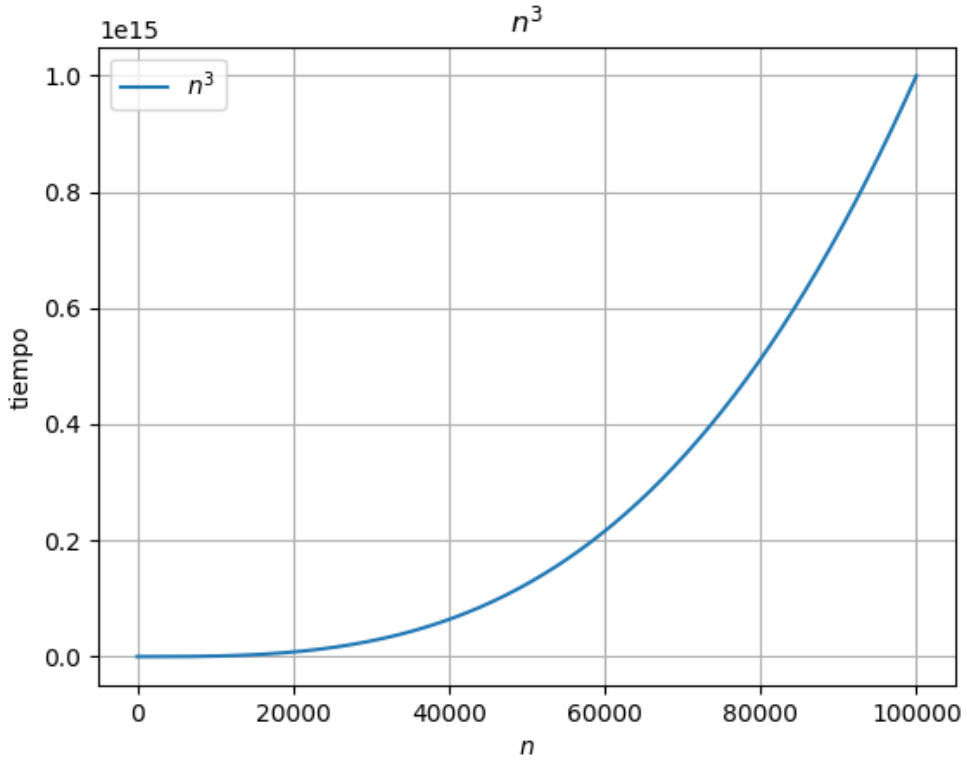


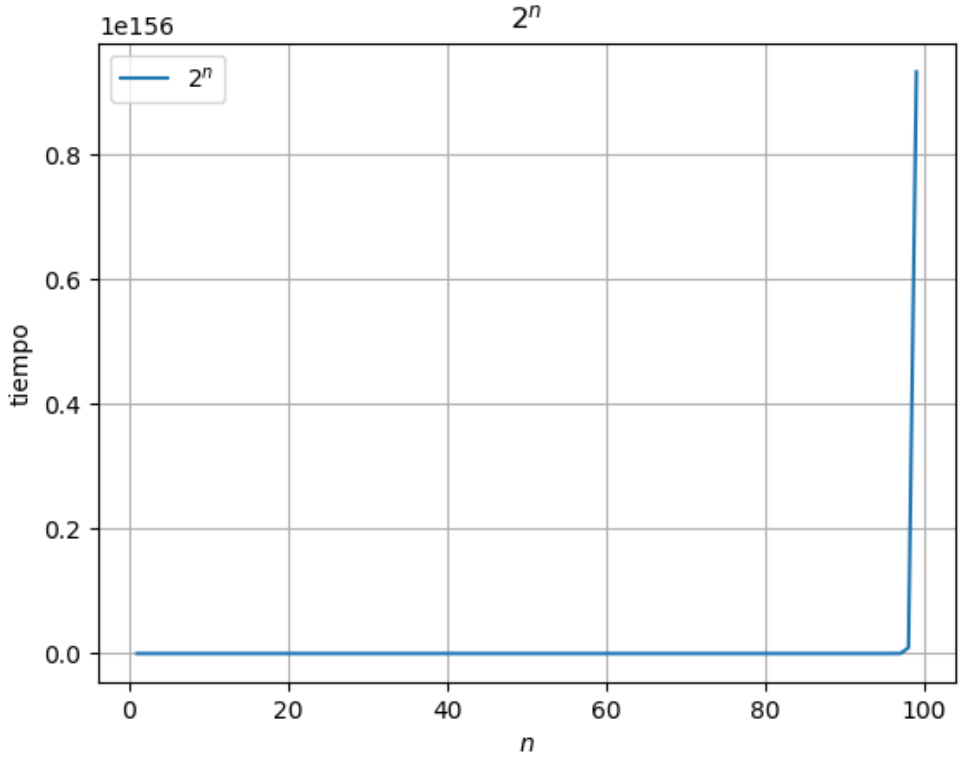

lg(n)
lg(n)
15.0
12.5
10.0
7.5
5.0
2.5
0.0
tiempo
0
20000
40000
60000
80000
100000
n

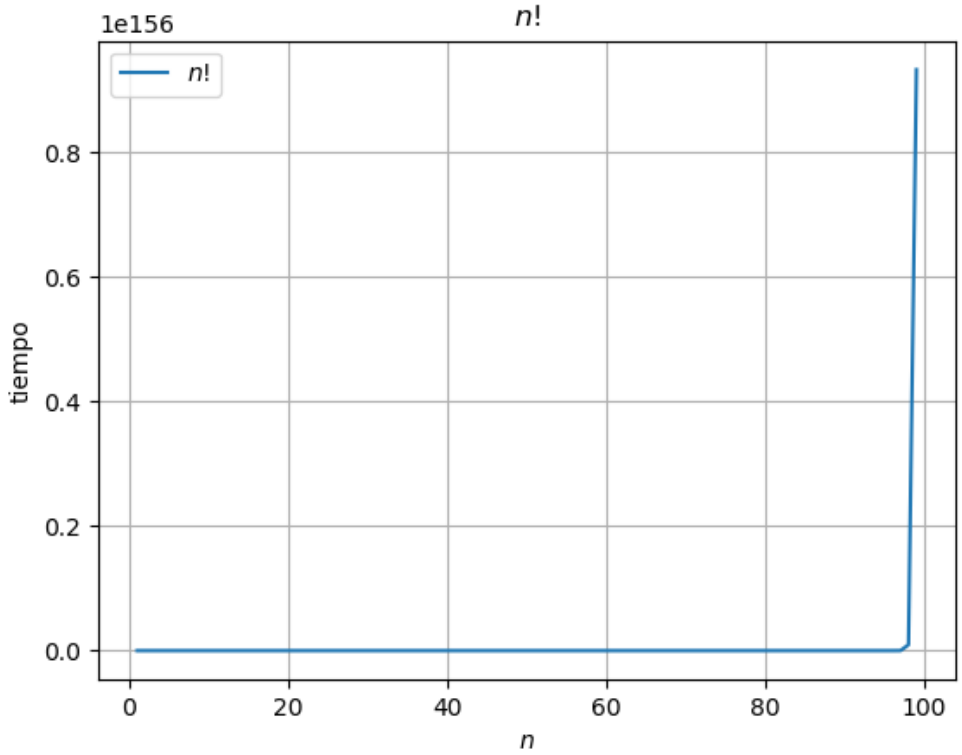

n
2
n
100000
80000
60000
40000
20000
0
tiempo
0
20000
40000
60000
80000
100000
n


nlg(n)
nlg(n)
1500000
1250000
1000000
750000
500000
250000
0
0
20000
40000
60000
80000
100000
n


n^2
1e10
n^2
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
0
20000
40000
60000
80000
100000
n


n^3
1e15
n^3
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
0
20000
40000
60000
80000
100000
n

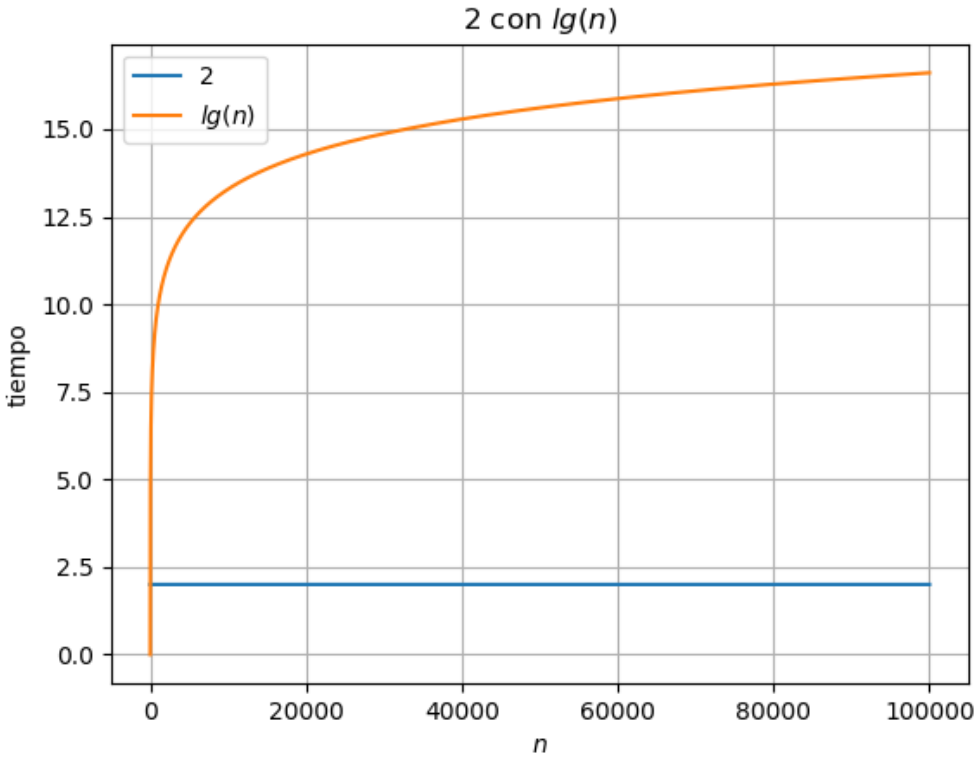

2^n
1e156
2^n
0.8
0.6
0.4
0.2
0.0
tiempo
0
20
40
60
80
100
n

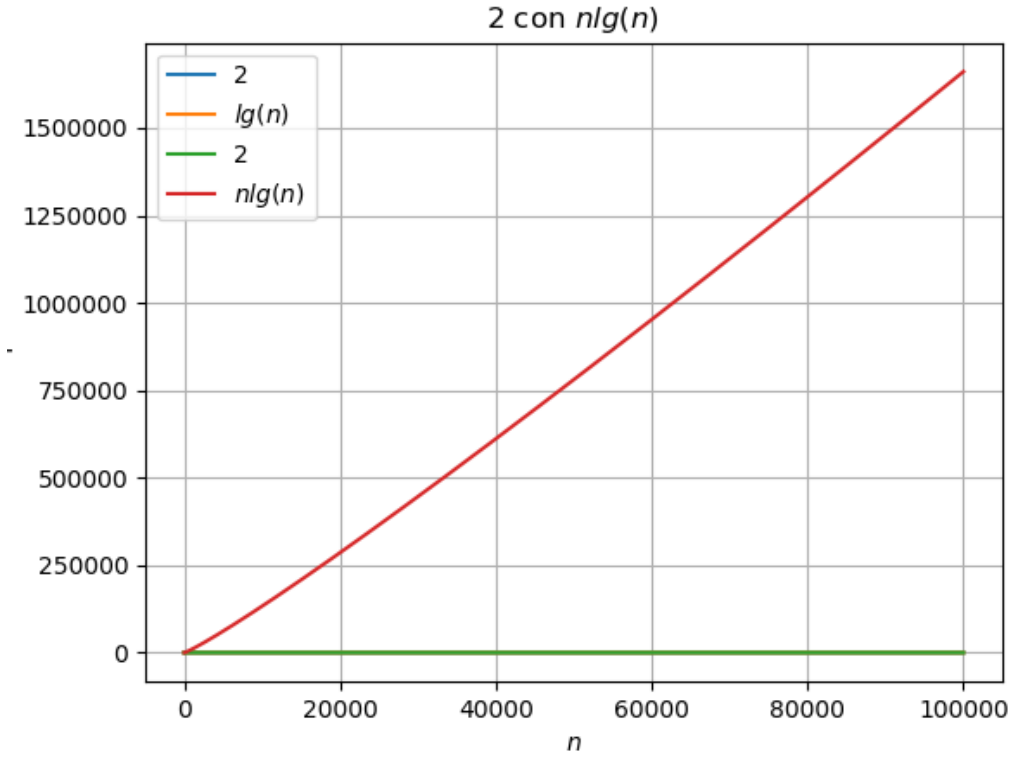


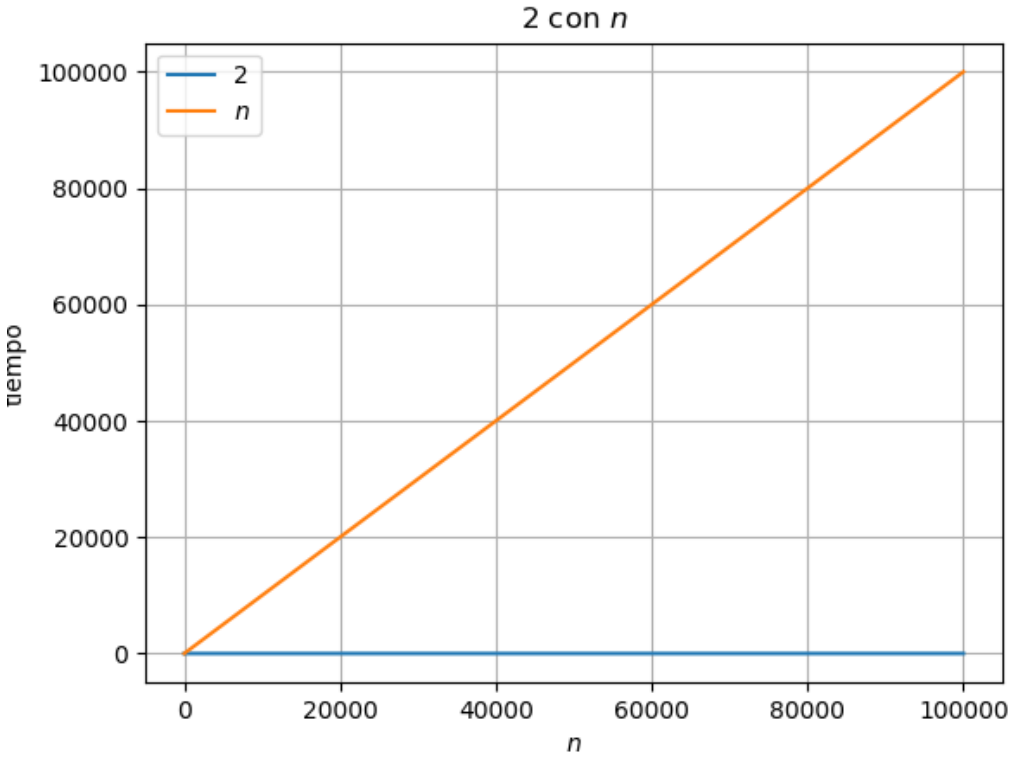
En las graficas anteriores podemos apreciar como los bosquejos de las graficas se ven gravemente o casi despreciables las variaciones a medida que crece n. Las complejidades perfectasa a simple vista pueden ser la constante, logaritmica y lineal, de ahi le siguen las cuadraticas, exponenciales y factoriales.

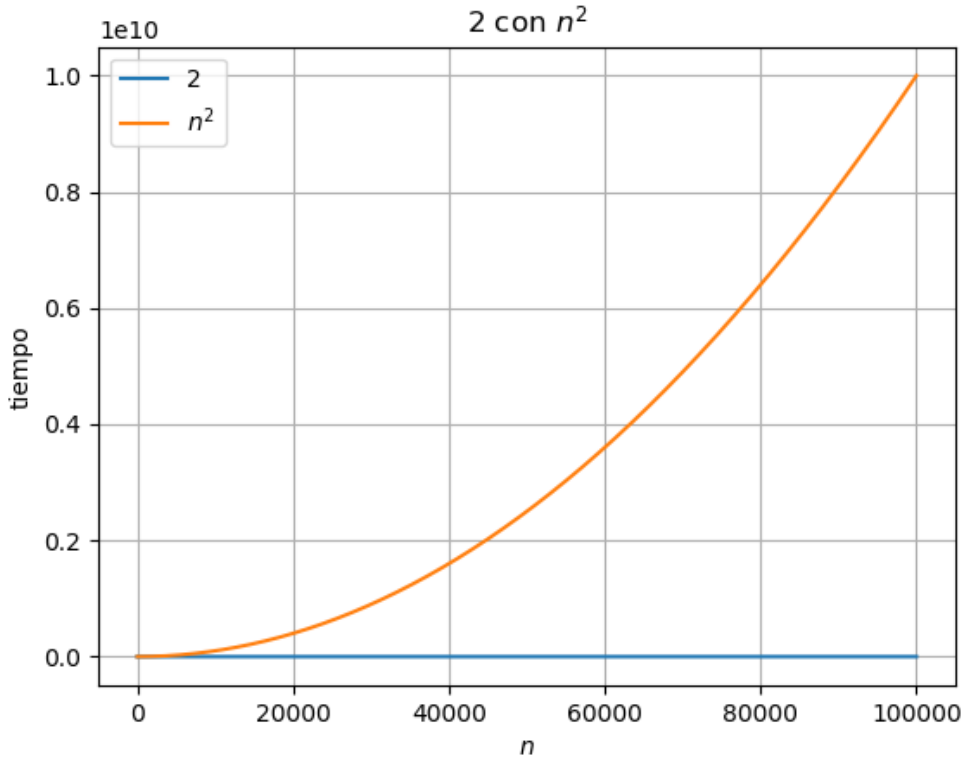
2. Graficas a pares

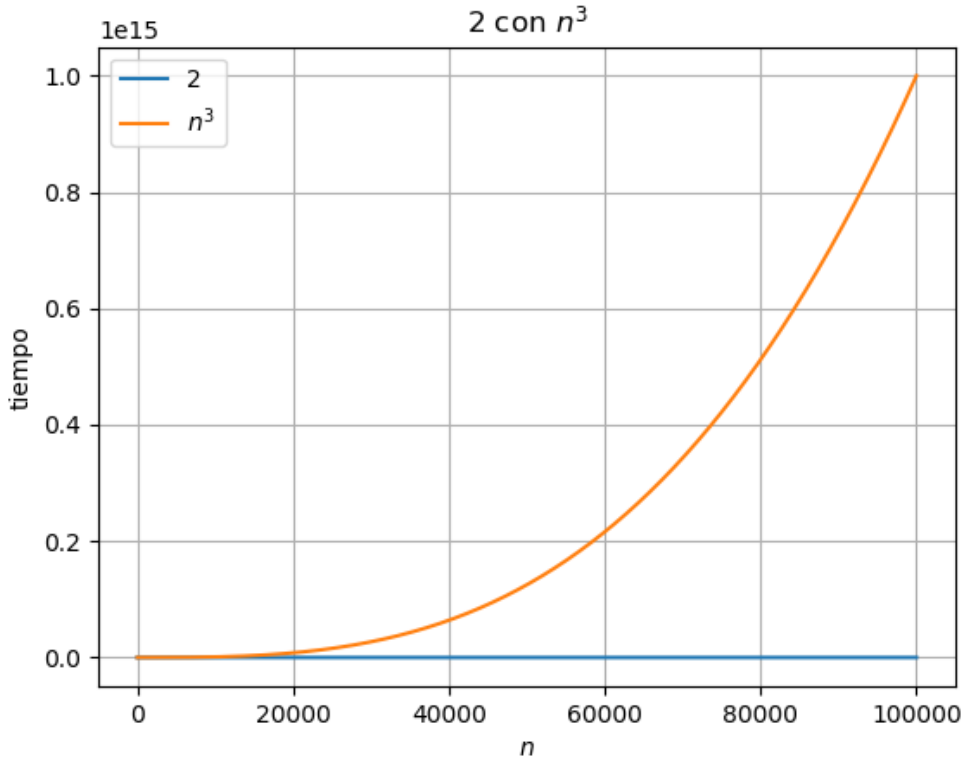
2.1. Complejidad constante

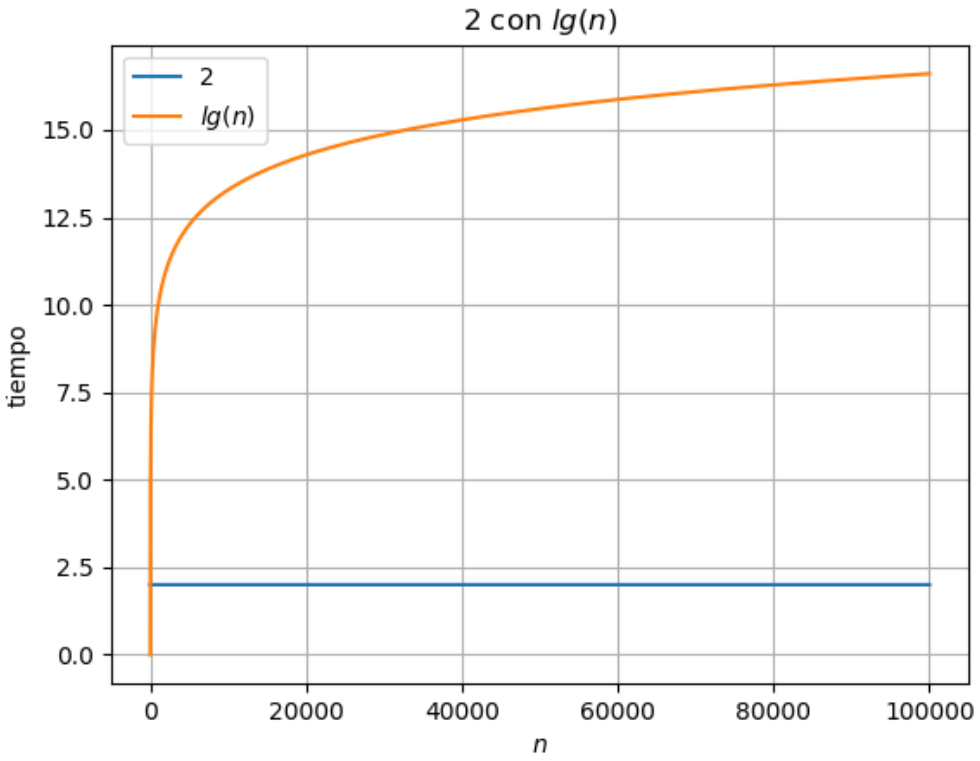



2 con nlg(n)
2
lg(n)
2
nlg(n)
1500000
1250000
1000000
750000
500000
250000
0
0
20000
40000
60000
80000
100000
n


2 con n
2
n
100000
80000
60000
40000
20000
0
tiempo
0
20000
40000
60000
80000
100000
n

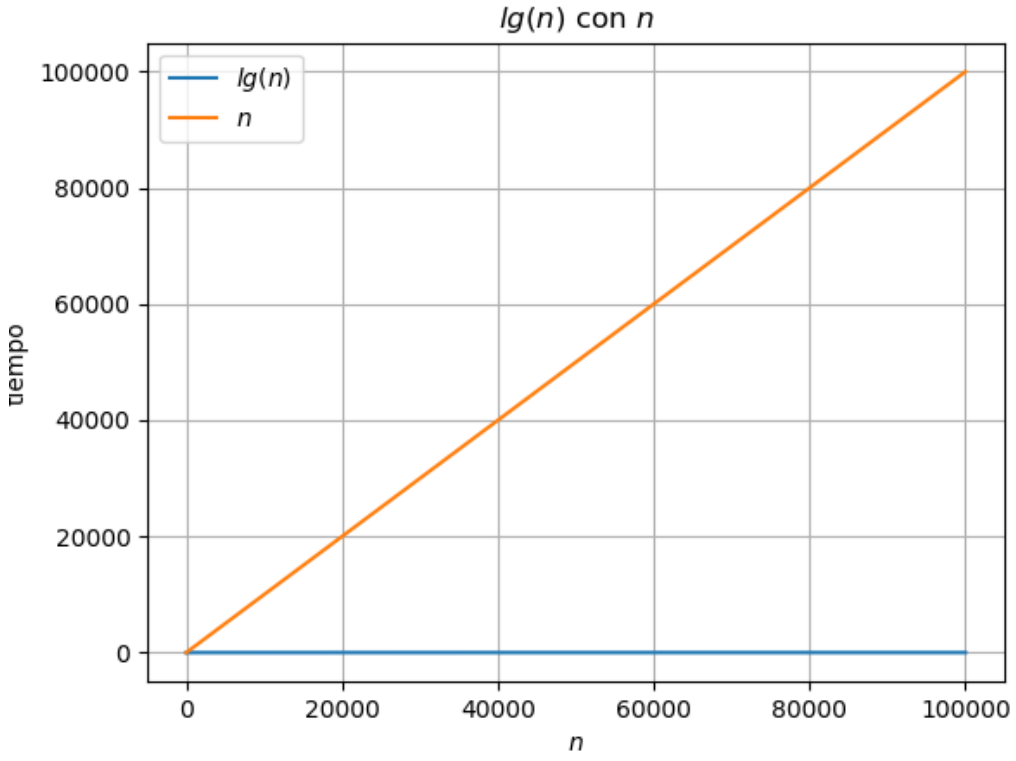

2 con n^2
1e10
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
2
n^2
0
20000
40000
60000
80000
100000
n

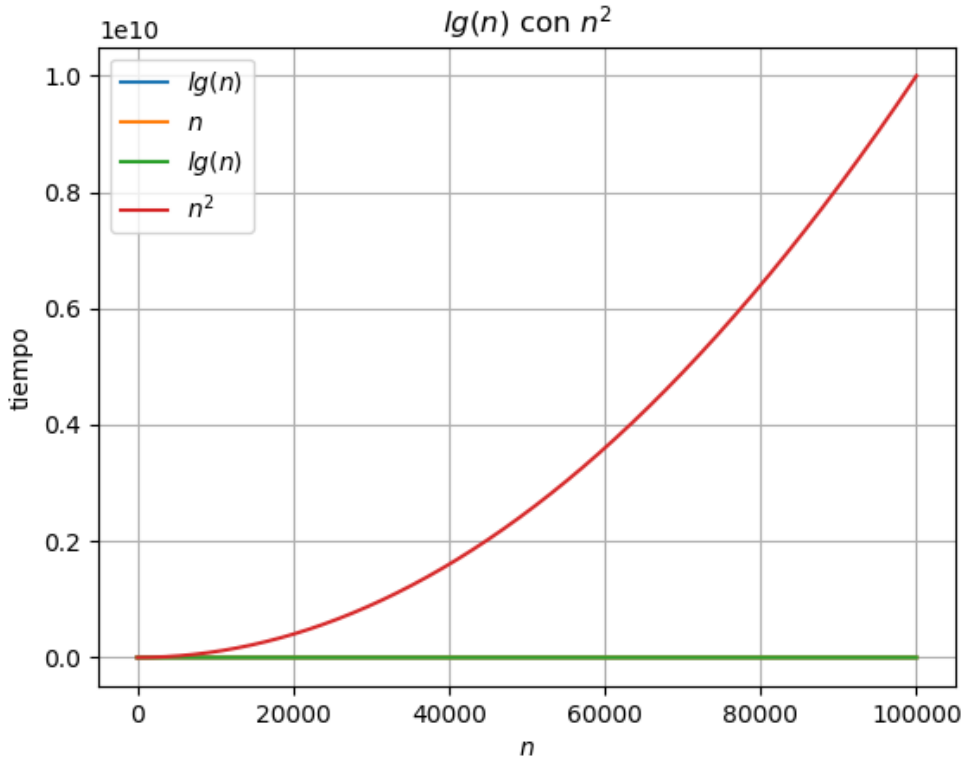

2 con n^3
1e15
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
2
n^3
0
20000
40000
60000
80000
100000
n

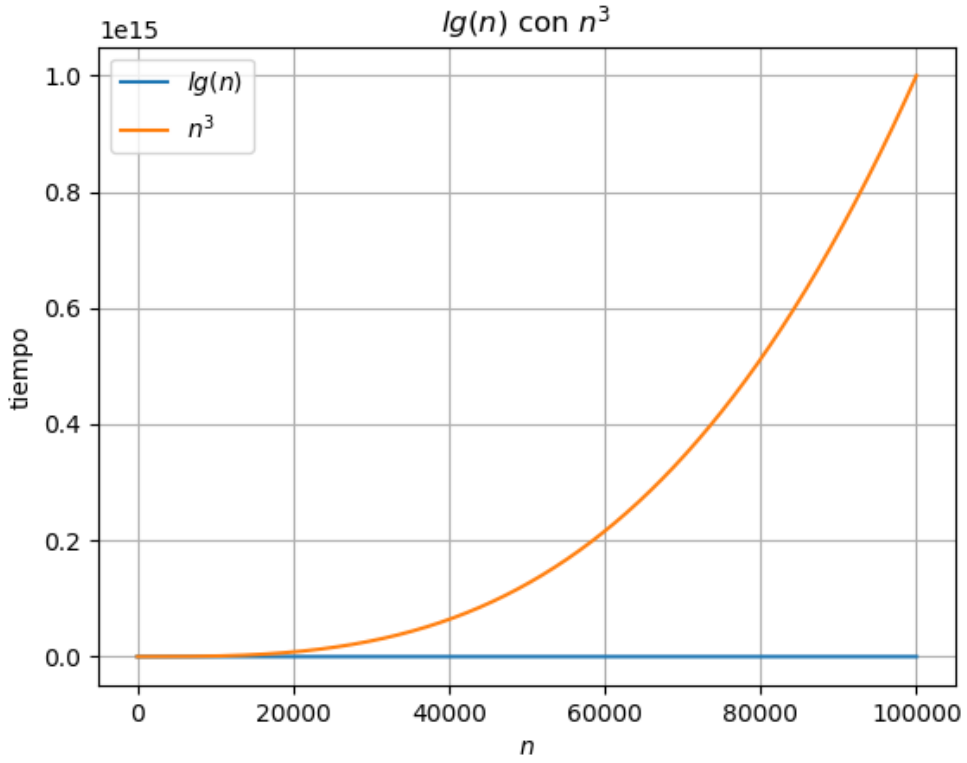


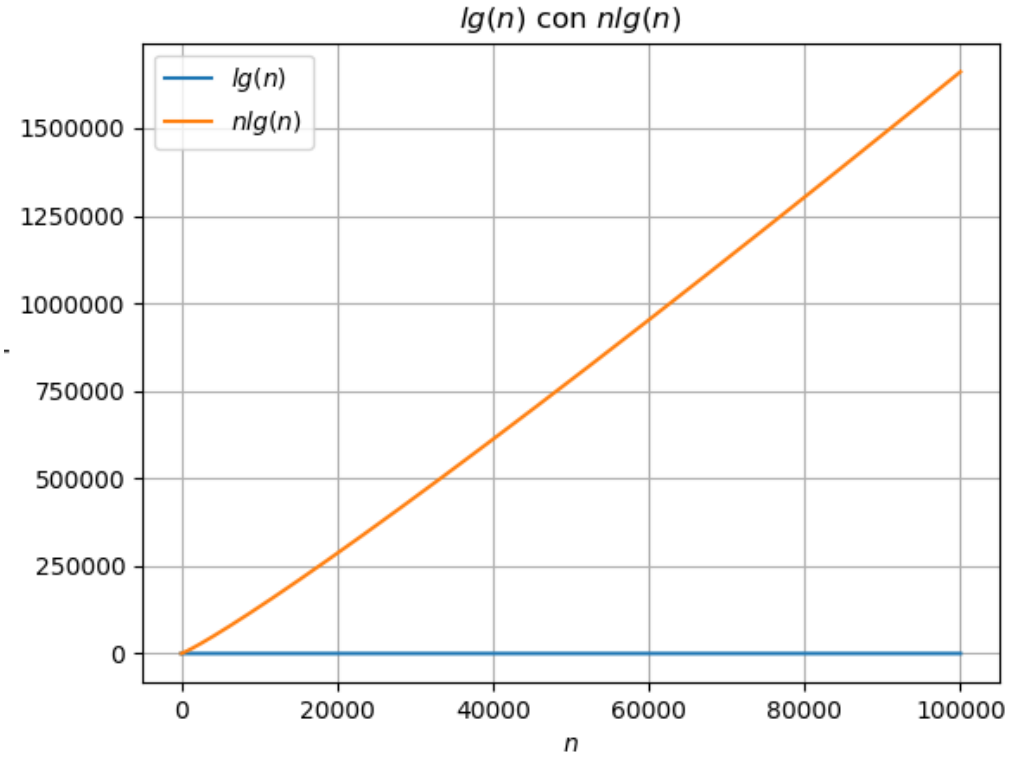
En las graficas anteriores pudimos observar lo que se vio en precalculo, la funcion constante nunca varia para cualquier tamaño de n, cosa que se nota bastante si la comparamos con una función muy costosa como el factorial.

2.2. complejidad logaritmica



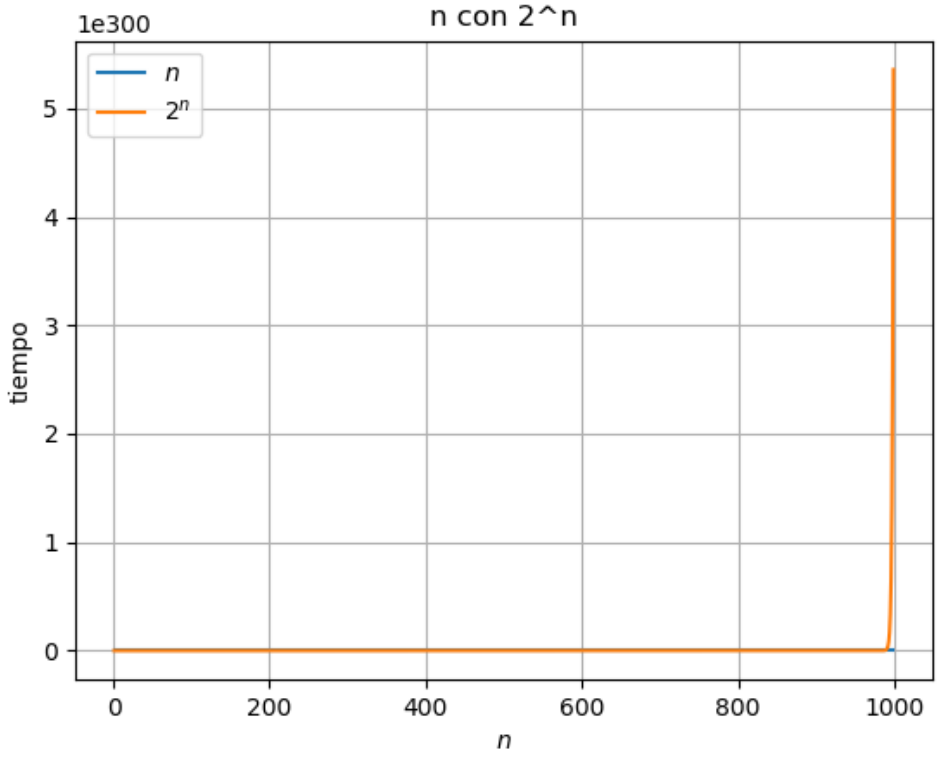


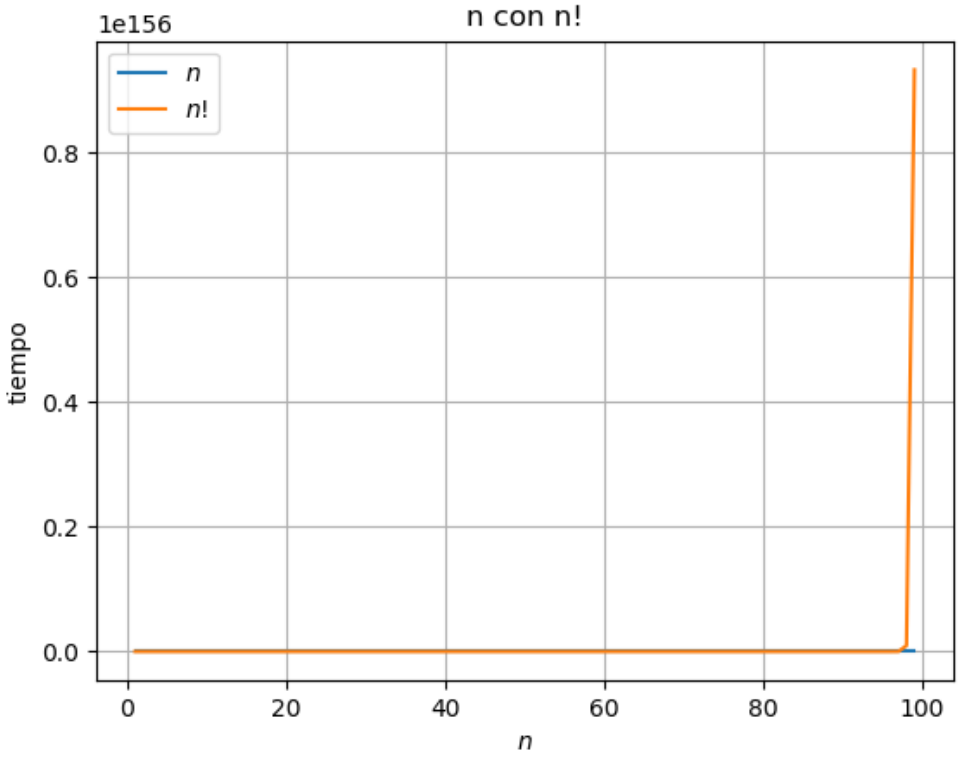

$lg(n)$ con n^3
1e15
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
$lg(n)$
n^3
0
20000
40000
60000
80000
100000
n

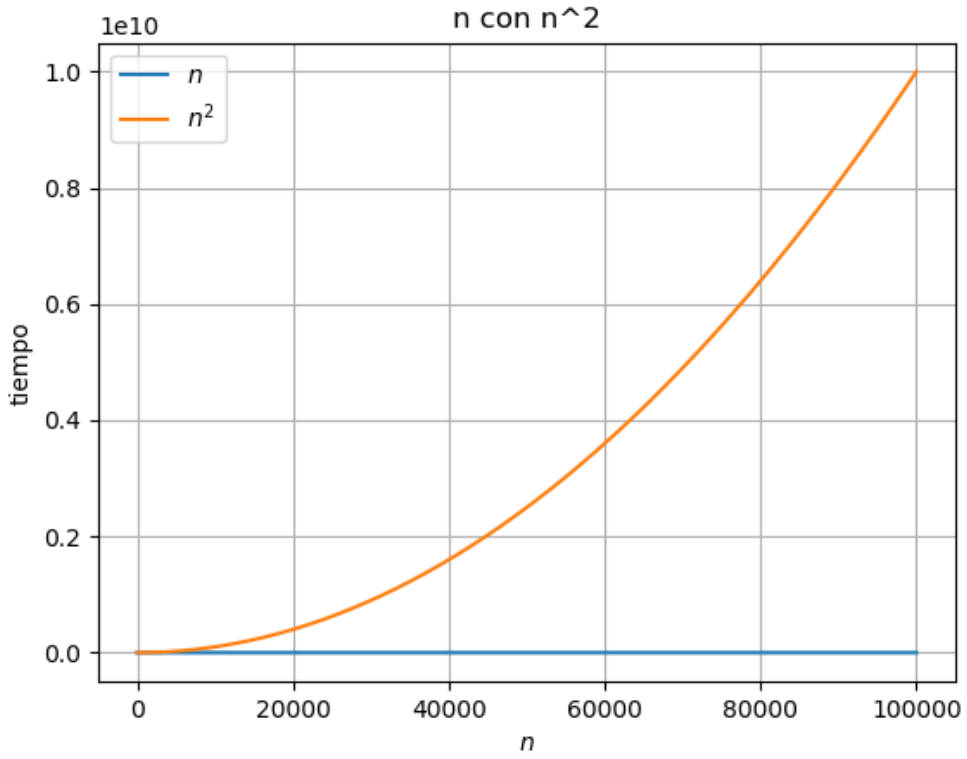

$lg(n)$ con $nlg(n)$
1500000
1250000
1000000
750000
500000
250000
0
$lg(n)$
$nlg(n)$
0
20000
40000
60000
80000
100000
n

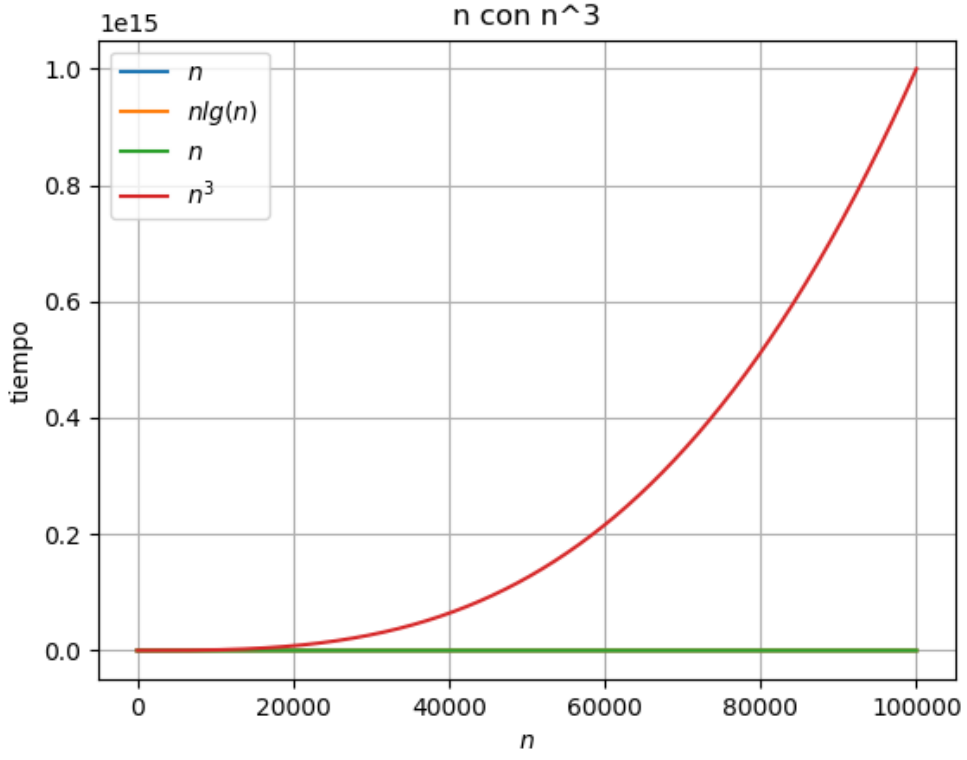
Se aprecia que para rangos muy grandes de n la funcion complejidad $\log n$ asemeja a una constante, en conclusion esta misma no sufre mucho cambio a medida de que $n \to \infty$.

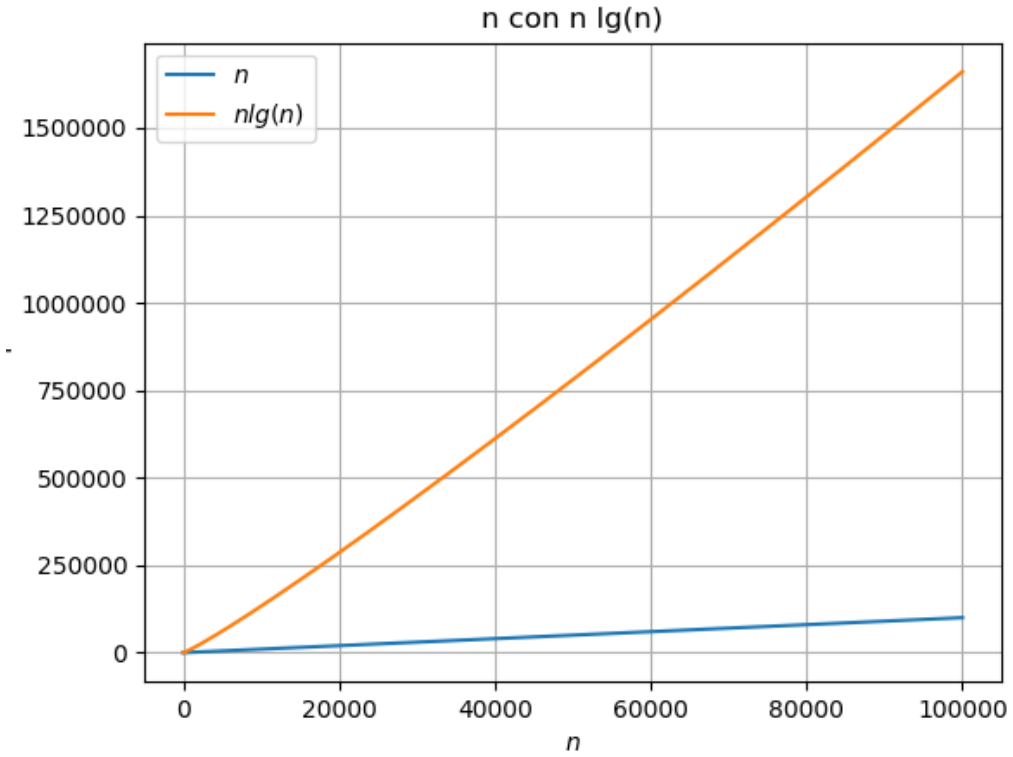
2.3. Complejidad lineal





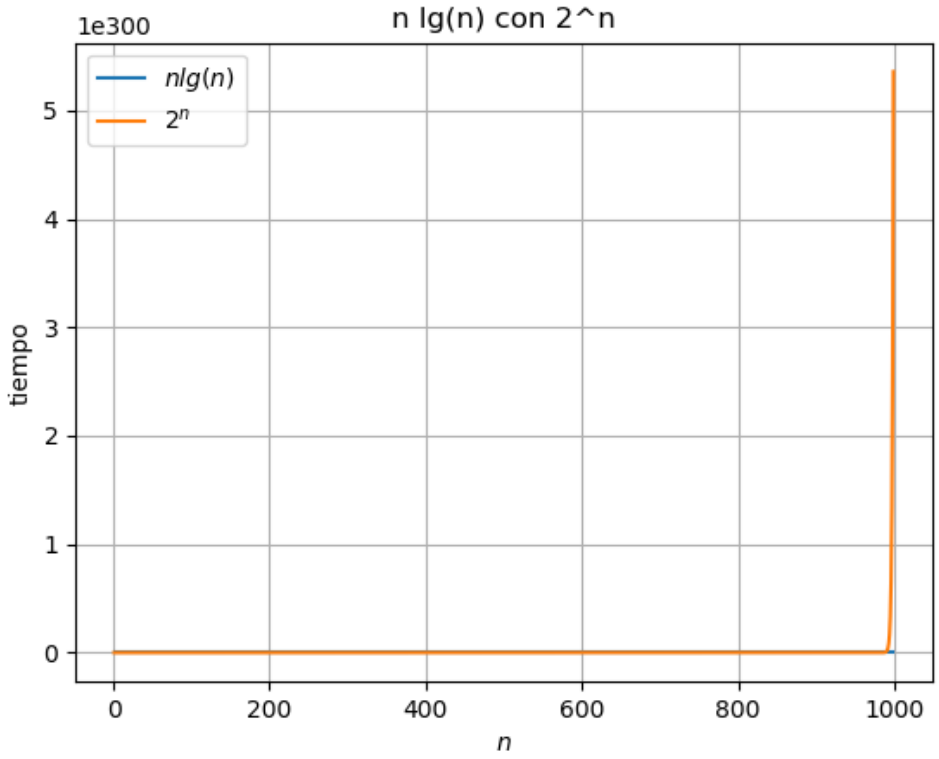

n con n^2
1e10
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
n
n^2
0
20000
40000
60000
80000
100000
n

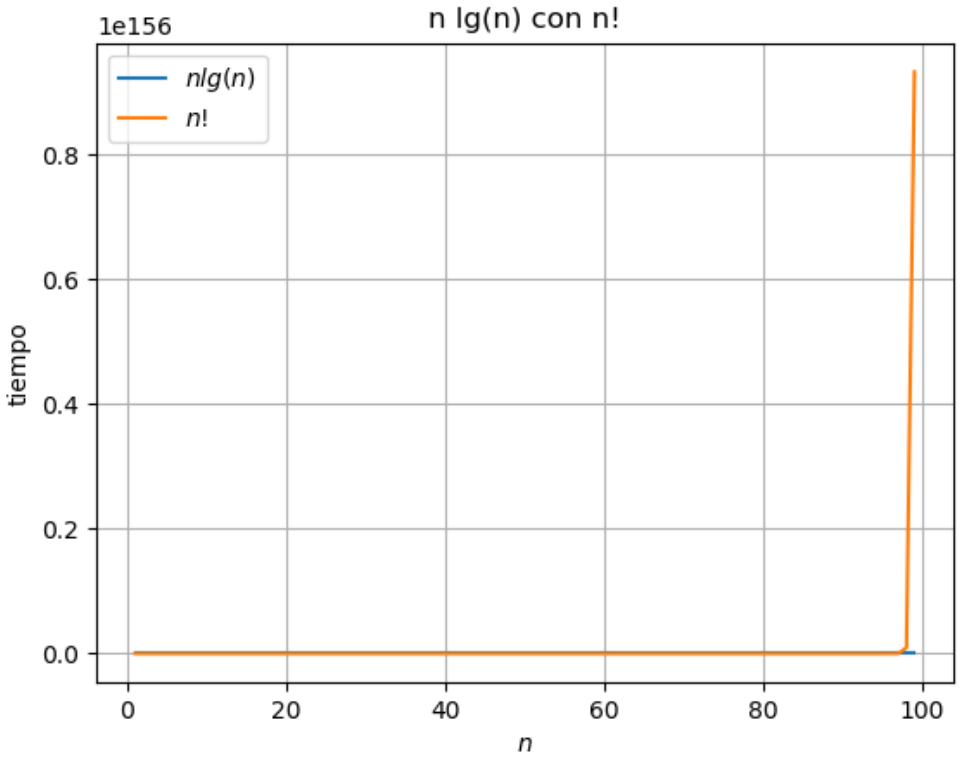

n con n^3
1e15
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
n
nlg(n)
n
n^3
0
20000
40000
60000
80000
100000
n

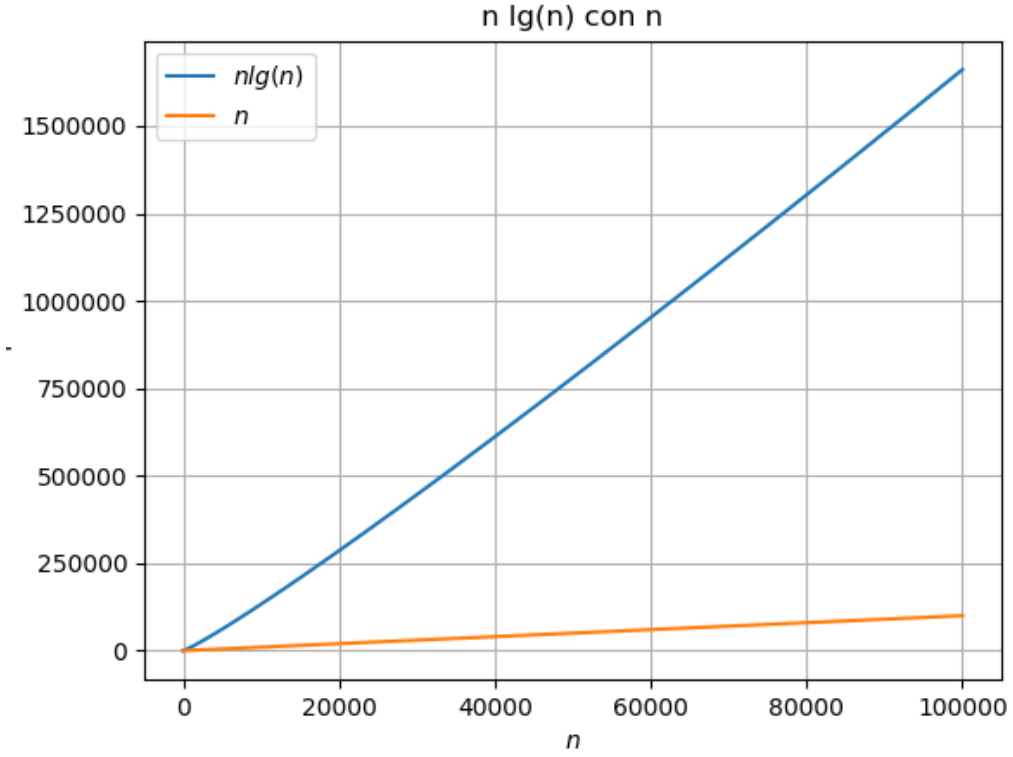


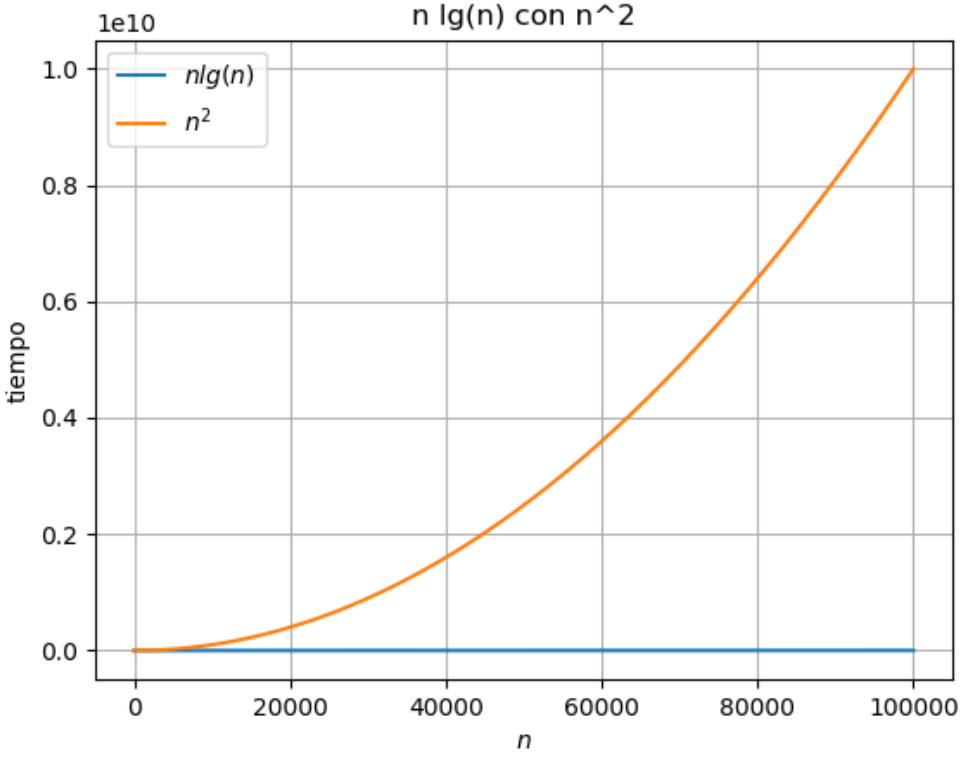
Definitivamente habria que preferirse un algoritmo de complejidad $O(n)$ a comparacion de las exponenciales ya que la diferencia en abismal para valores grandisimos.

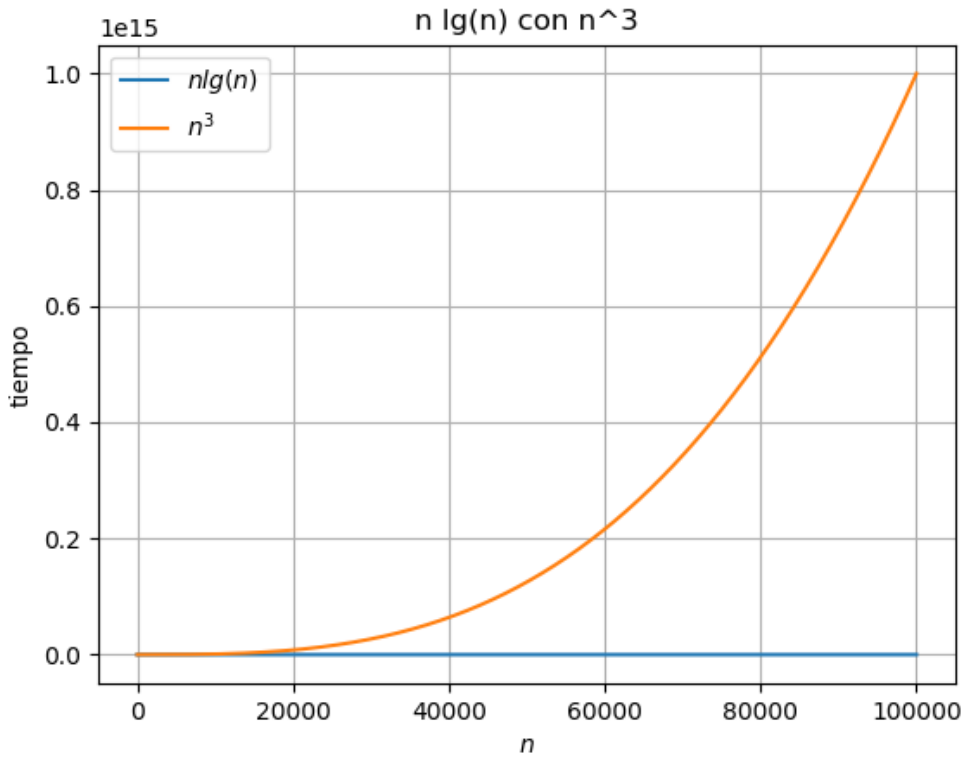
2.4. Complejidad $n \log n$





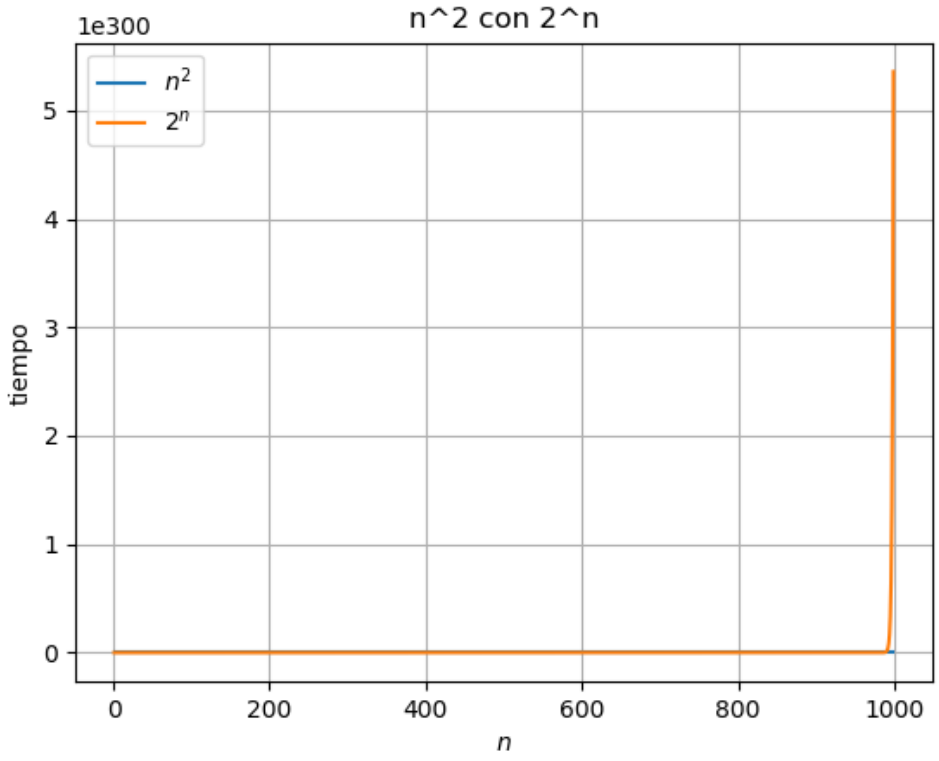

n lg(n) con n
nlg(n)
n
1500000
1250000
1000000
750000
500000
250000
0
0
20000
40000
60000
80000
100000
n

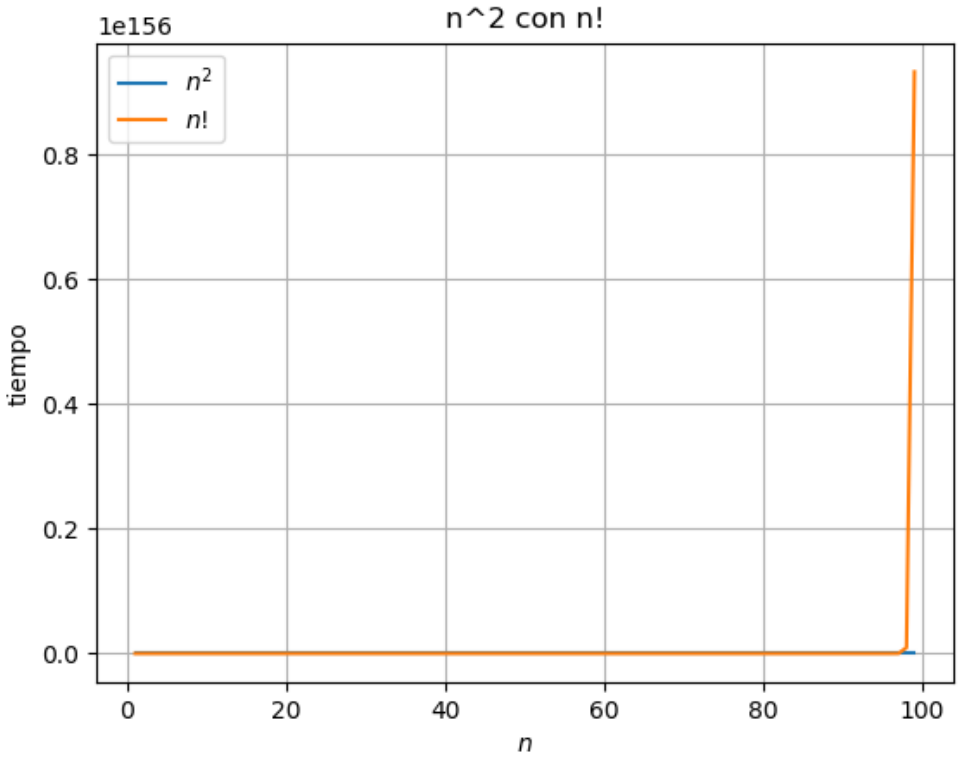

n lg(n) con n^2
1e10
nlg(n)
n^2
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
0
20000
40000
60000
80000
100000
n

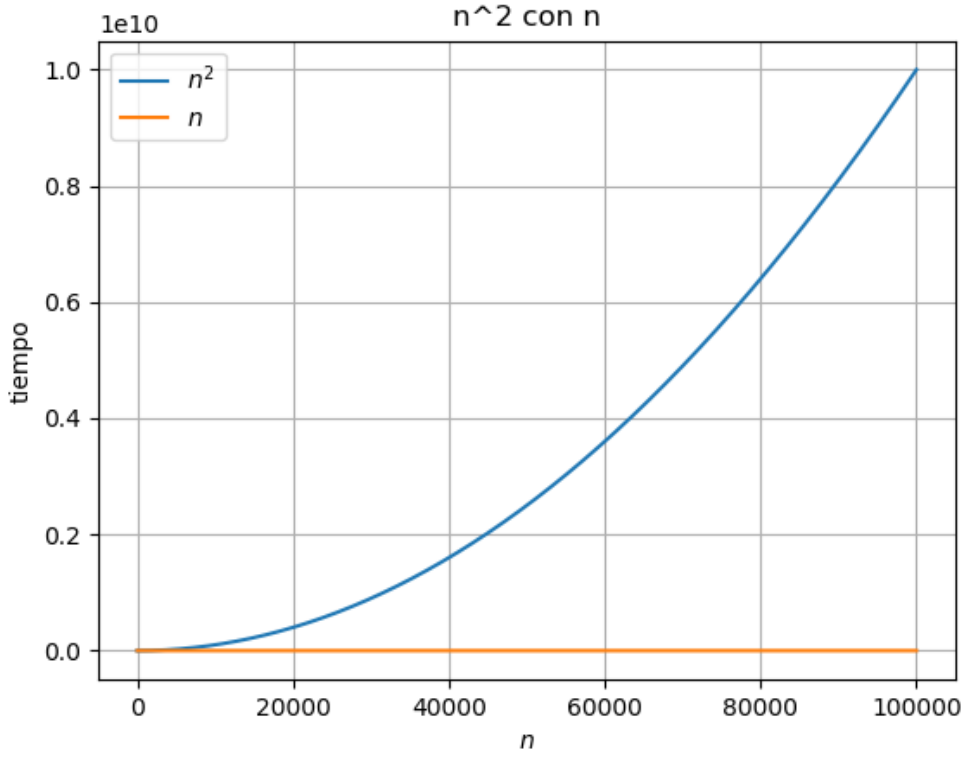


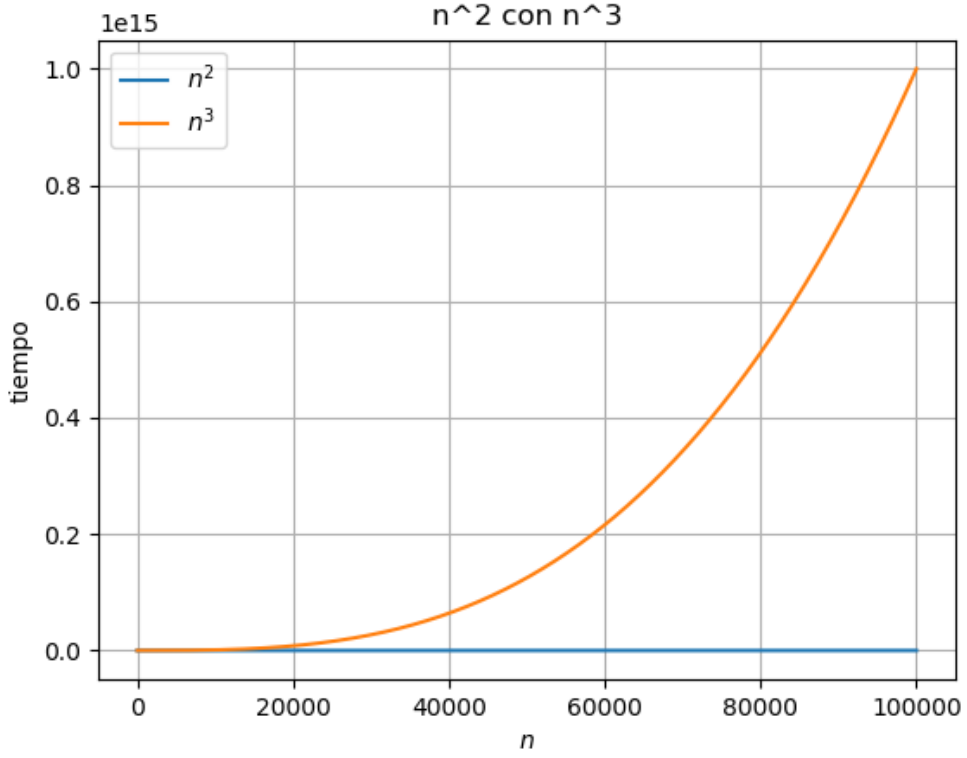
La unica grafica que toma valores mayores para una n grandes el $\log n$, cosa bastante razonable pues $\log n > 1 \iff n > 1$, la tercera mejor complejidad discutida en este ejercicio (claro, si considermos que la complejidad constante para una constante k pequeña). Esta complejidad es una de las que mas se presenta como optimizacion a algoritmos y la verdad es que se justifica el porque en las graficas.

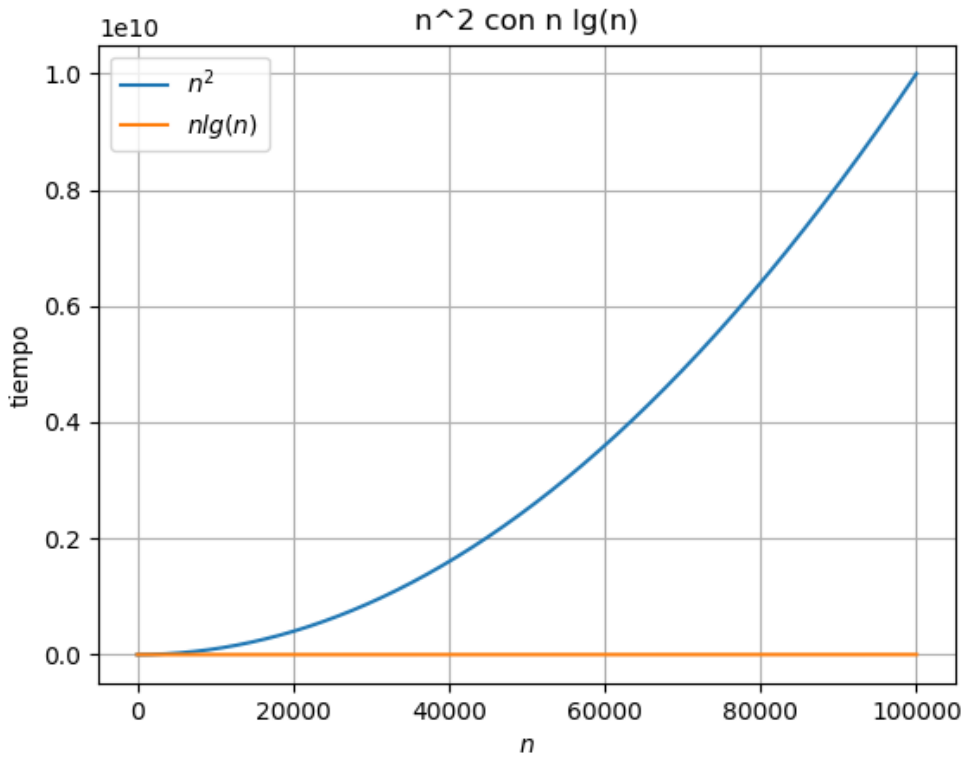
2.5. Complejidad cuadratica





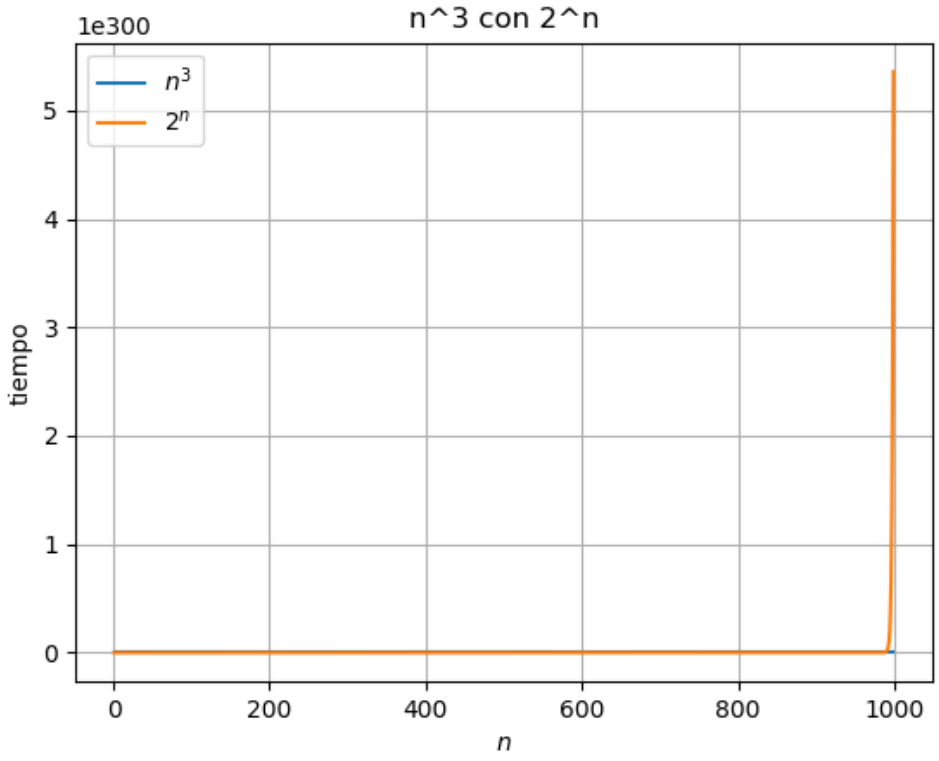

n^2 con n
1e10
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
n^2
n
0
20000
40000
60000
80000
100000
n

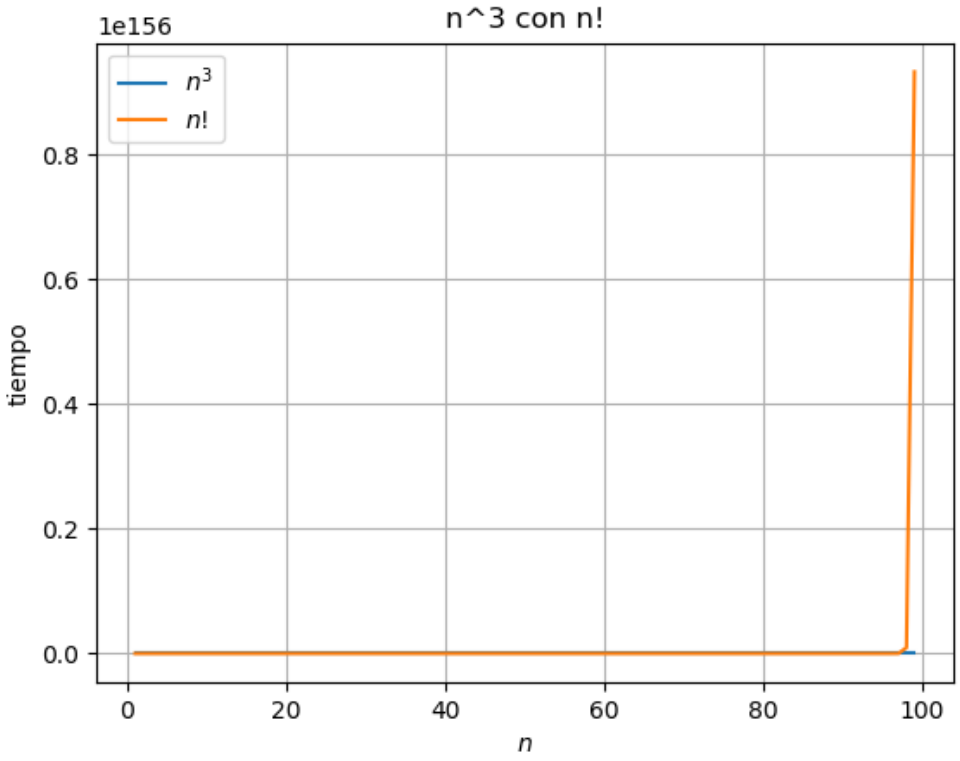

n^2 con n^3
1e15
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
n^2
n^3
0
20000
40000
60000
80000
100000
n

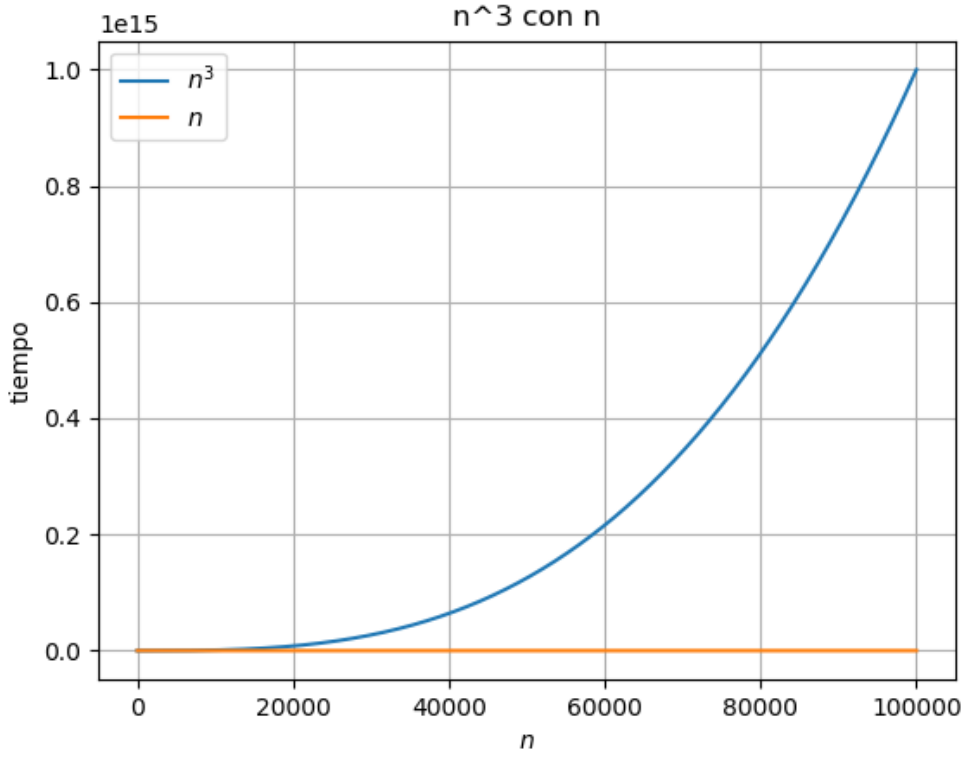


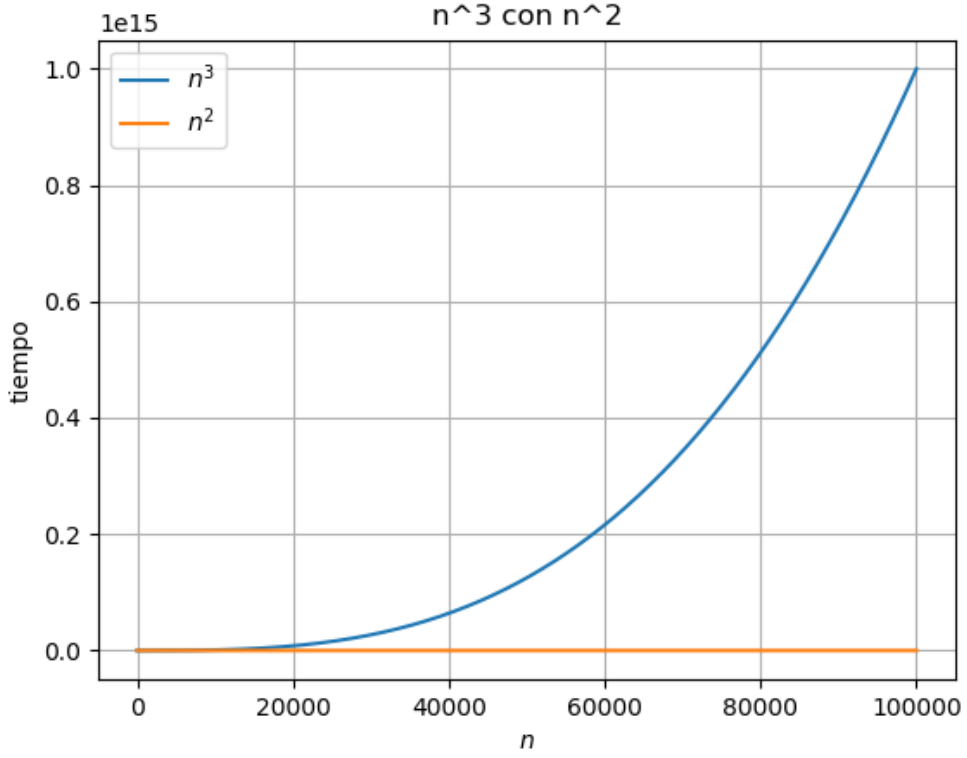
La unica grafica con valores mas grandes en el codominio a esta es n^3 por obvias razones. Es interesante ver uno de los costos mas usados en programacion pues siempre nos presentamos con un lazo for donde estas caracteristicas de graficas ocurren aunque varias veces se pueden omitir, muchas veces nos enfocamos en que metodos o funciones previamente implementadas hagan su cometido despreciando la complejidad de estos casos muy frecuentes en estructuras de datos lineales.

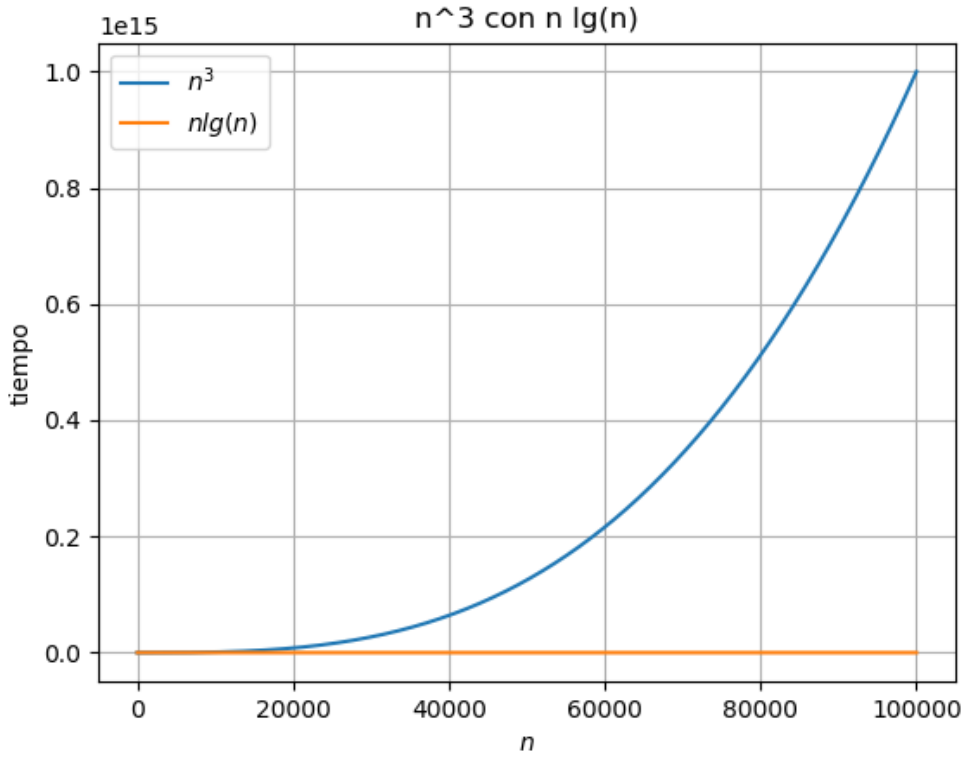
2.6. Complejidad cubica





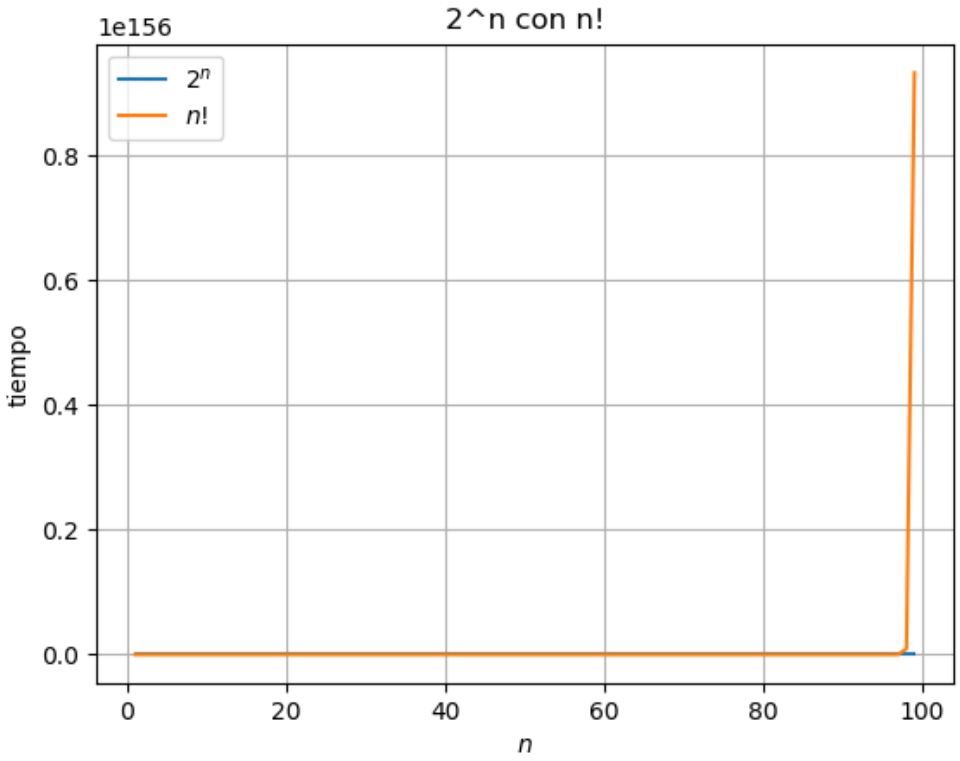

n^3 con n
1e15
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
n^3
n
0
20000
40000
60000
80000
100000
n

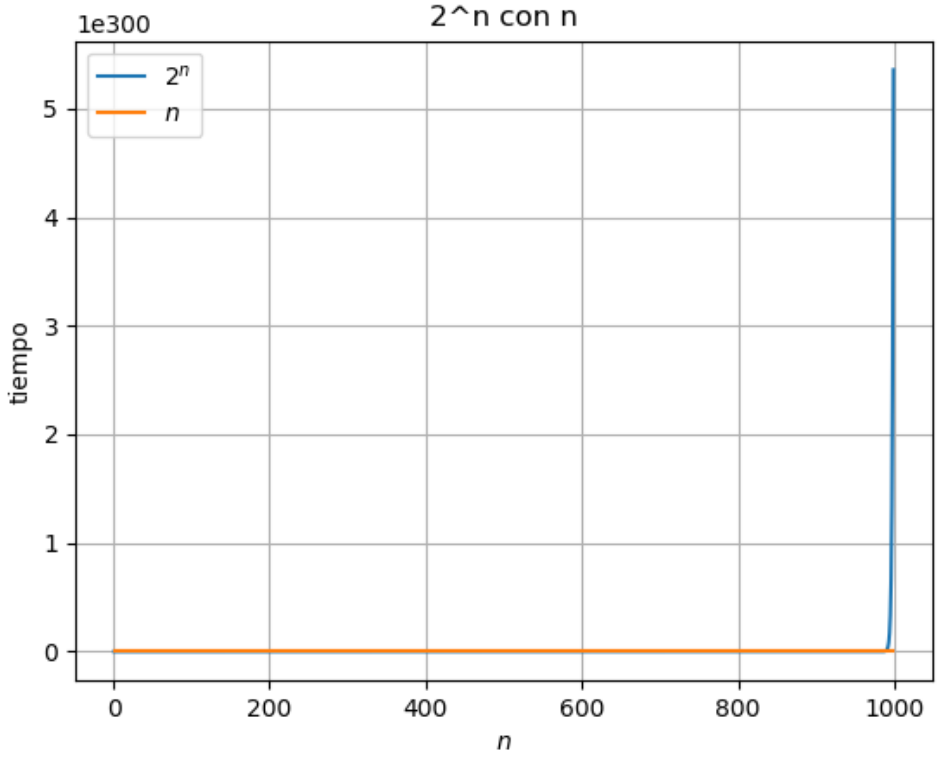

n^3 con n^2
1e15
1.0
0.8
0.6
0.4
0.2
0.0
tiempo
n^3
n^2
0
20000
40000
60000
80000
100000
n

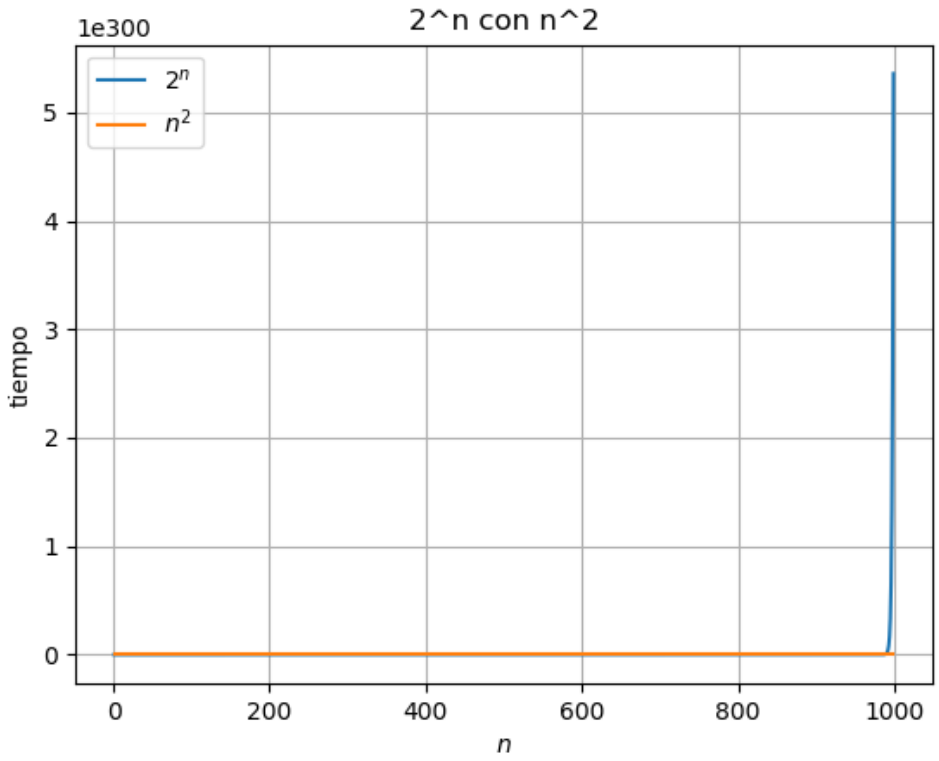


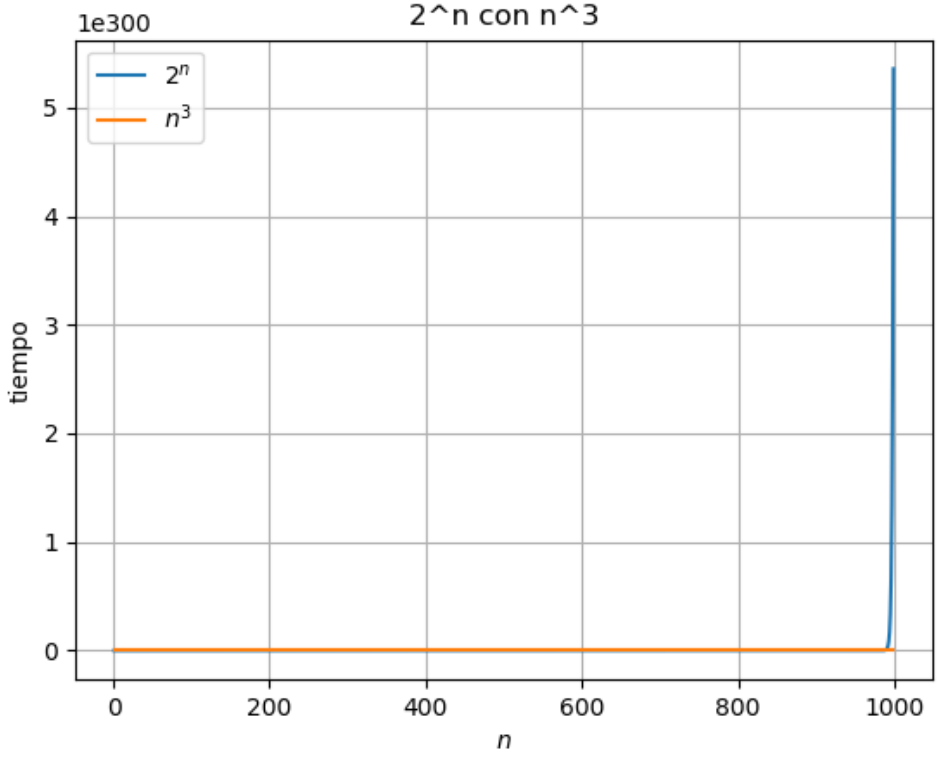
Funcion peor que n^2 como ya se discutio anteriormente, se deberia de evitar a cualquier costo a menos que sea inherente del problema como en la multiplicacion de matrices donde el tamaño del problema esta relacionado con las dimensiones de la matriz.

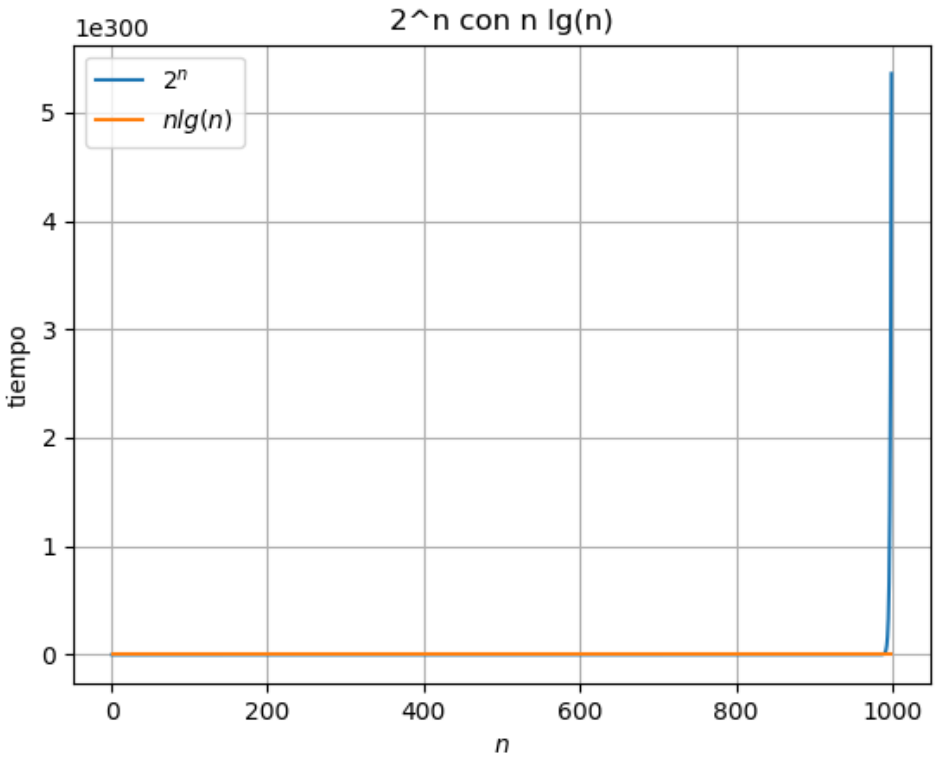
2.7. Complejidad exponencial (con $c = 2$)





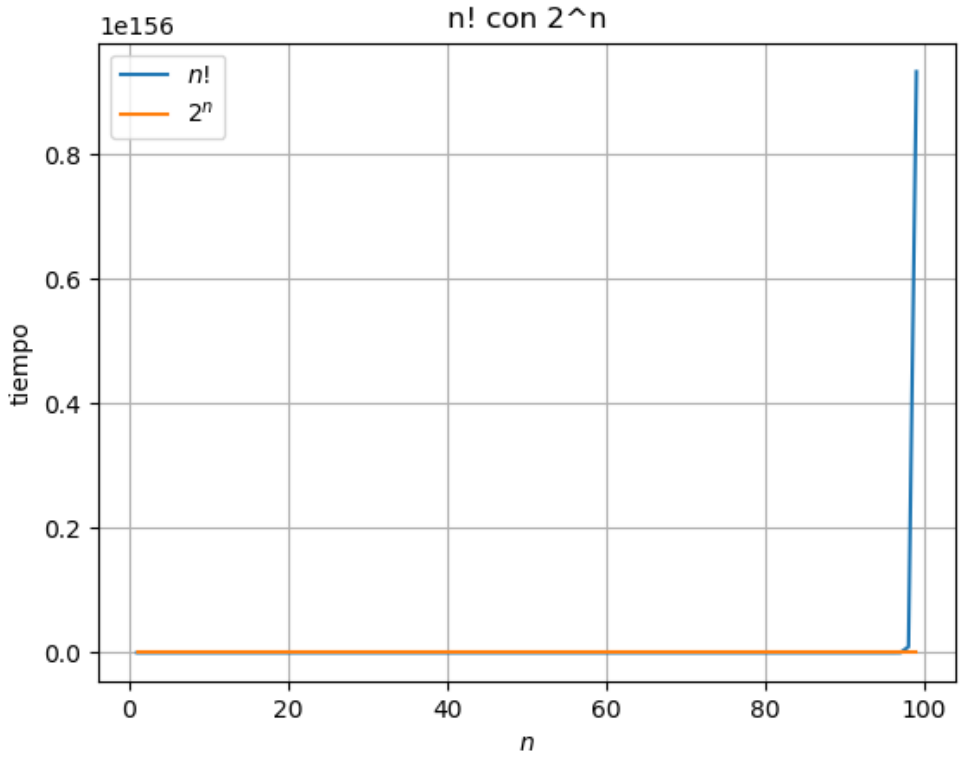

2^n con n^2
1e300
2^n
n^2
tiempo
n
0
200
400
600
800
1000
0
1
2
3
4
5

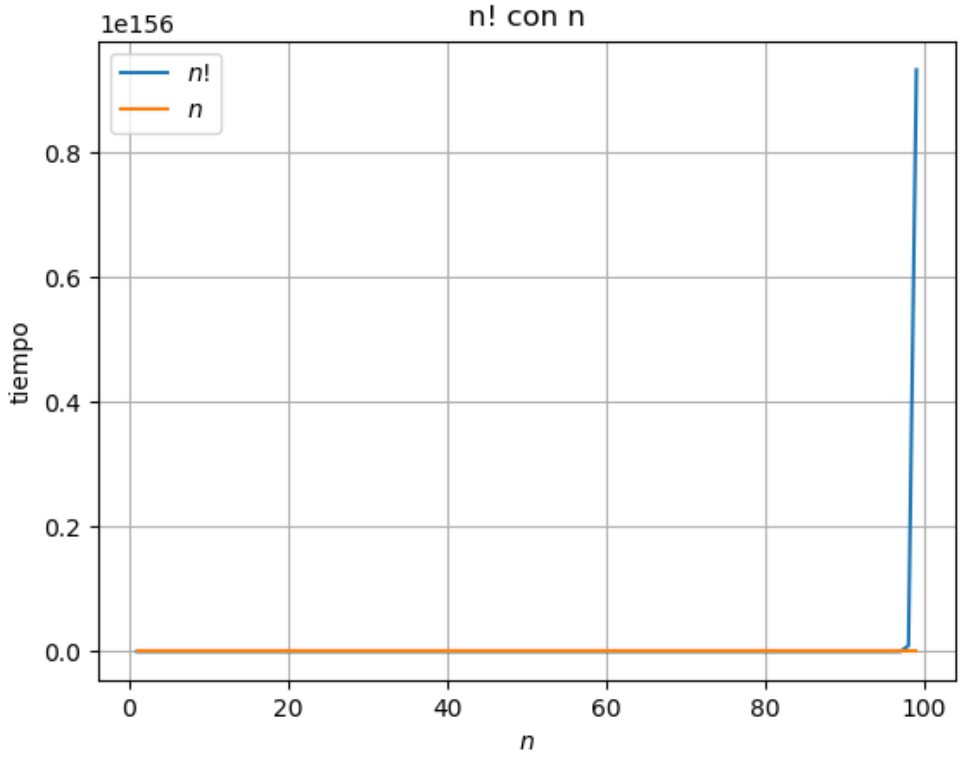

2^n con n^3
1e300
2^n
n^3
tiempo
n
0
200
400
600
800
1000
0
1
2
3
4
5

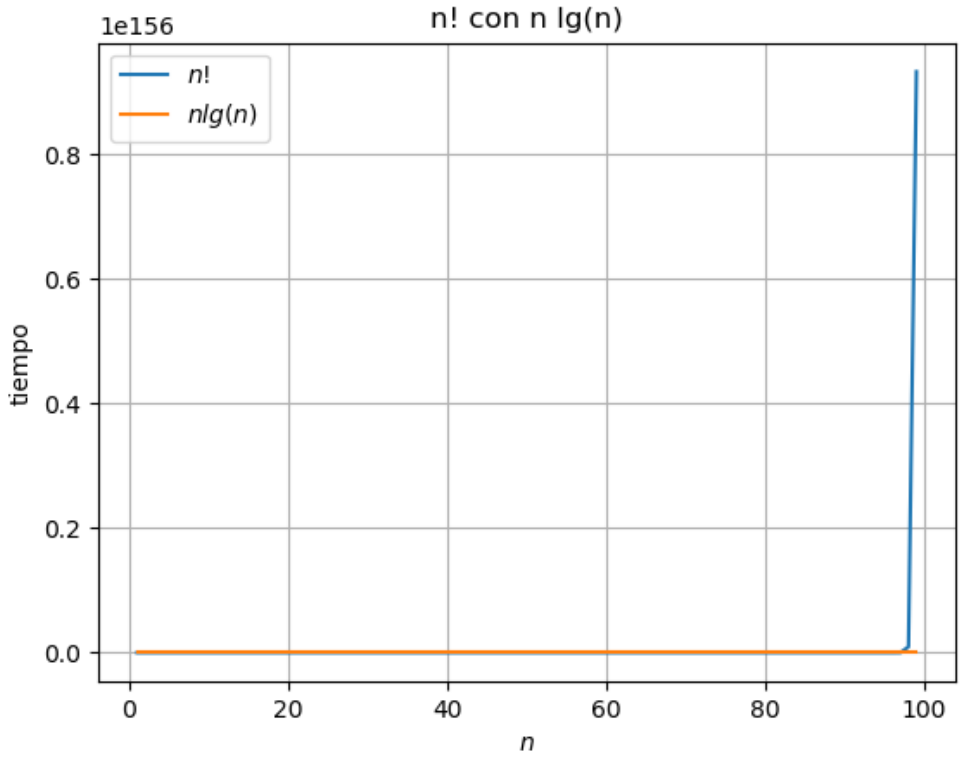


Aqui se escogio $c = 2$ pues este numero se presenta muy frecuentemente en los algoritmos, no es de extrañar pues los bits estan compuestos por dos valores, de cualquier forma, y asi ver que como se comportan varios algoritmos crecientes exponencialmente con esta base. A partir de aqui estamos entrando en terreno muy costoso pues las demas complejidades se ven opacadas por estas dos ultimas.

2.8. Complejidad factorial







Una de las peores complejidades de todas, en general si tu tamaño del problema es muy grande con frecuencas muy altas se deberia de omitir las implementaciones de algoritmos con este tipo de complejidad a menos que no se tenga de otra. Por lo general algorimos de complejidad $O(n) = n!$ se presentan en algorimos de backtracking. Se aprecia como absorbe por completo a todos los demas con n pequeña, tanto asi que fue el no se pudo graficar con las $1 < n < 100000$ solicitadas en conjunto con la exponencial, creame que intente hacer la grafica con estos valores por muy descabellado que parezca pero no obtuvoe buenos resultados, el graficador se desbordaba para $n = 0, 1, 2, ..., 10000$. Intente obtener 100000! de multiples formas pero siempre terminaba con mi computadora muerta, si se puede escoger otra implementacion que esta definitivamente se deberia de tomar.