Práctica 1: Análisis de Eficiencia de Algoritmos

Torres de Hanoi

Para el estudio de los algoritmos con orden de eficiencia O(2^n) vamos a utilizar el de las Torres de Hanoi.
La diferencia principal con los algoritmos de otras eficiencias es que utilizamos tamaños de entrada muy pequeños, su eficiencia es peor.

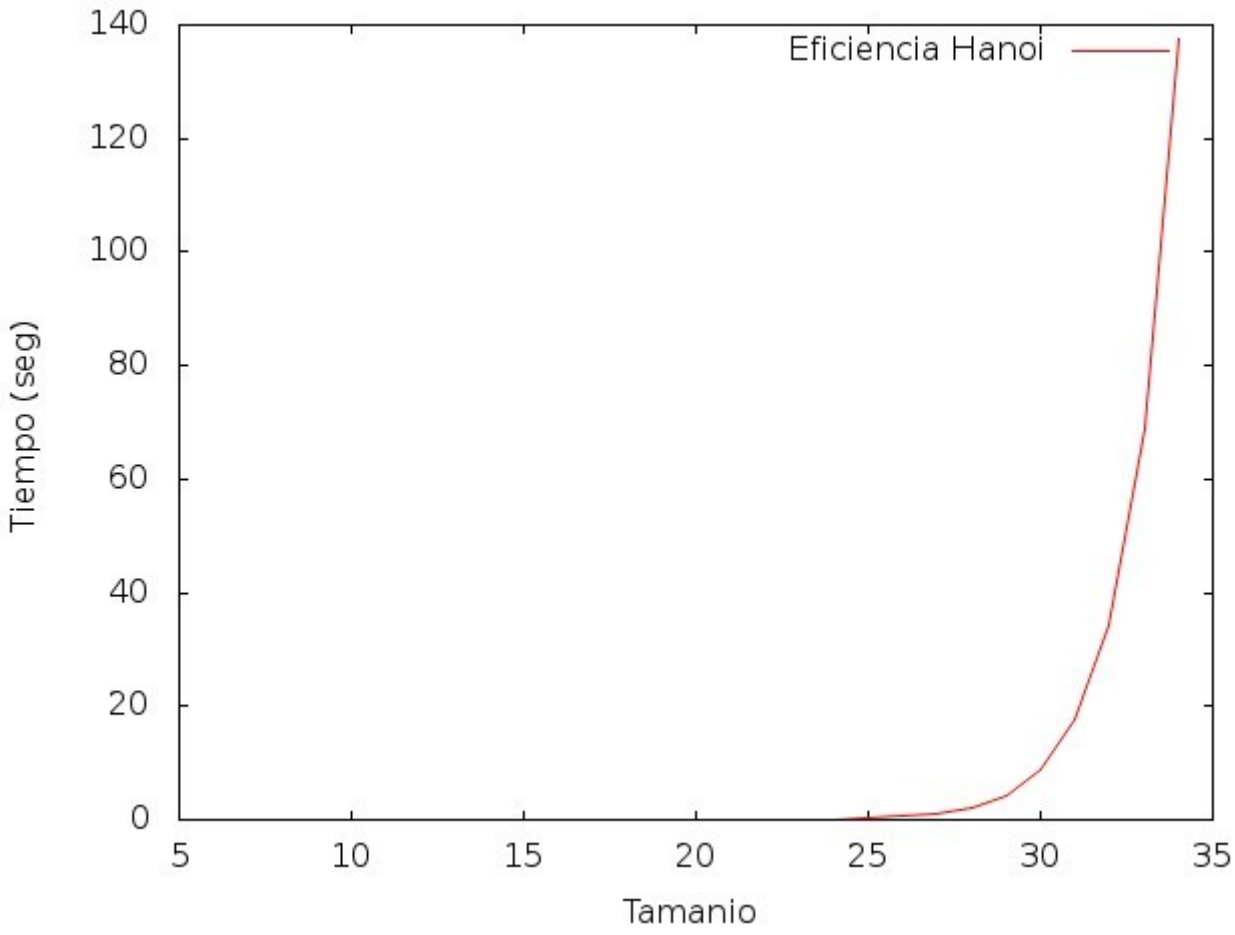
Calculo de la Eficiencia Empírica

Lo primero que hacemos es calcular la tabla con los tiempos en función del tamaño de las entradas, en este caso, se comienza desde una entrada muy pequeña (9), para terminar en 34.

- Tabla Tamaño/Tiempo

9 4e-06
10 9e-06
11 6.6e-05
12 0.000128
13 0.000256
14 0.000508
15 0.00045
16 0.000918
17 0.001818
18 0.003605
19 0.007215
20 0.012728
21 0.019266
22 0.037212
23 0.070588
24 0.137573
25 0.271657
26 0.544326
27 1.08621
28 2.24842
29 4.40313
30 8.94462
31 17.5627
32 34.3644
33 68.8651
34 137.583

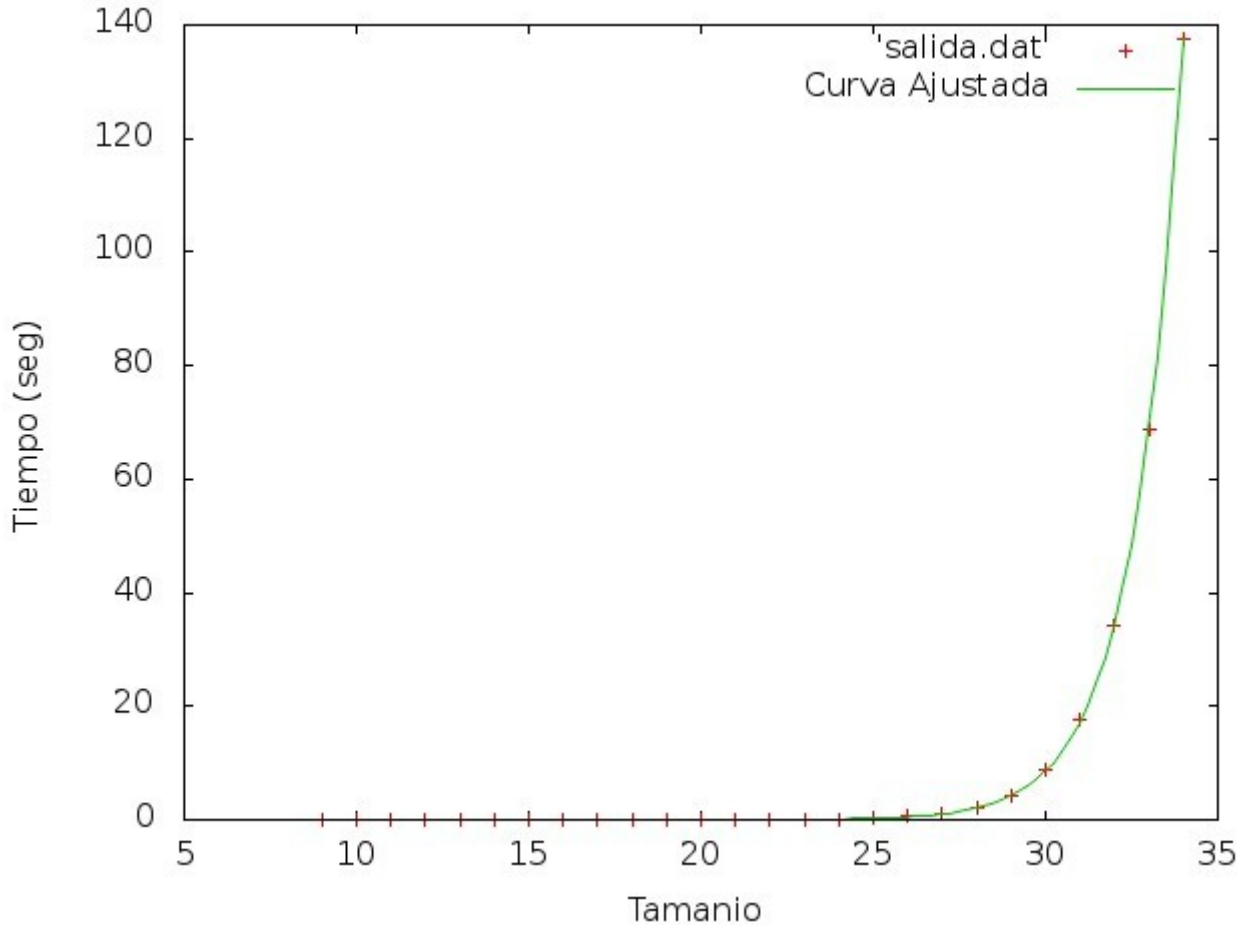
Una vez obtenida la tabla utilizamos gnuplot y generamos la siguiente gráfica:



Los tiempos han sido sacados de un pc con Intel(R) Core(TM) i7-2670QM CPU @ 2.20GHz y la siguiente orden de compilación:
g++ hanoi.cpp -o hanoi -std=gnu++0x

Calculo de la Eficiencia Híbrida

Para el siguiente estudio he utilizado **f(x) = (a0*2^x)** en gnuplot y con la gráfica que se genera podemos comprobar como se produce un ajuste prácticamente perfecto, esto nos dice que la información teórica y las pruebas empíricas coinciden.



Estudio adicional

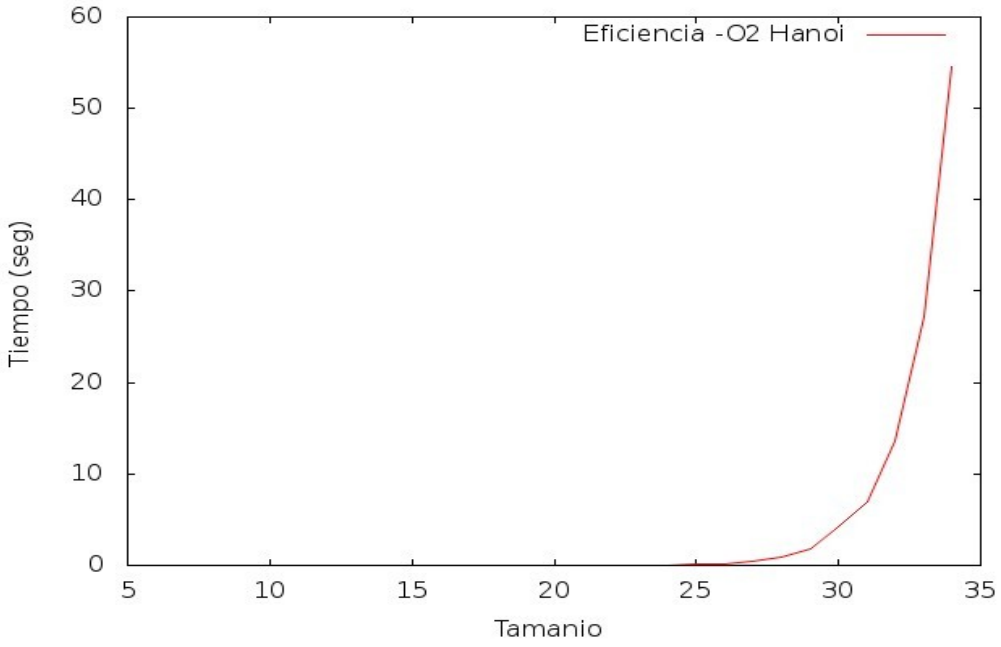
Ahora probamos con una opción de optimización compilación como es -O2 con la cual conseguimos mejorar los tiempos en función de las entradas, dando como resultado la siguiente tabla y gráfica:

-Tabla:

9 5e-06
10 8e-06
11 1.4e-05

12 2.8e-05
13 5.5e-05
14 0.000112
15 0.000295
16 0.000486
17 0.000883
18 0.001759
19 0.003283
20 0.005892
21 0.010951
22 0.019843
23 0.030015
24 0.057436
25 0.116318
26 0.215733
27 0.432737
28 0.88417
29 1.75423
30 4.19834
31 6.9042
32 13.6655
33 27.2631
34 54.5477

-Gráfica:



Como conclusión podemos ver que se consigue una mejora de ejecución de mas de la mitad de tiempo, fijándonos en la entrada mas costosa, vemos que sin optimización tardamos 137 segundos y para -O2 se queda en 54.