P1

Guión de Prácticas. Algorítmica.

PRÁCTICA 1: ANÁLISIS DE EFICIENCIA DE ALGORITMOS

PEDRO BEDMAR LÓPEZ

Índice:

1. Planteamiento del problema:

En esta práctica se pretende abordar la comparación de eficiencia entre algoritmos. Ya que en la parte teórica de la asignatura se está estudiando eficiencia teórica, en esta parte de prácticas nos centraremos en mayor medida en la empírica.

Para ello, vamos a comparar la eficiencia de una serie de algoritmos implementados en C++ utilizando entradas de diferentes tamaños. Calcularemos los tiempos de ejecución de cada uno de los algoritmos para cada tamaño y también seguiremos los demás pasos descritos en el índice, comprobando que la eficiencia empírica se ajusta a la teórica.

2. Tablas con eficiencia empírica de algunos algoritmos

Utilizando la implementación de los algoritmos proporcionada por los profesores, calcularemos el tiempo que tardan en ejecutarse gracias al reloj del sistema. Este reloj es capaz de capturar en una variable el momento temporal actual, y si realizamos una medición al inicio y otra al final de la ejecución del algoritmo y restamos la final menos la inicial, conseguimos la duración. Para transformarla a segundos la dividimos entre la constante CLOCKS_PER_SEC.

Hemos elegido los alogritmos ordenación por burbuja, mergesort, Floyd y el de las torres de Hanoi para calcular sus tiempos de ejecución. Para facilitar el trabajo, hemos compilado todos los ficheros .cpp donde se implementan los algoritmos y hemos utilizado la macro de terminal facilitada por los profesores para generar todos los datos de cada tabla con una sola sentencia.

```
cvi092058:Codigo pedrobedmar$ ./mimacro.sh
200000
400000
600000
800000
1000000
1200000
1400000
1600000
1800000
2000000
2200000
2400000
```

Burbuja (Eficiencia teórica $O(n^2)$)	
Tamaño del vector	Tiempo de ejecución
1000	0.001855
2000	0.007076
3000	0.017372
4000	0.034142
5000	0.052776
6000	0.08067
7000	0.114251
8000	0.151512
9000	0.192493
10000	0.2397
11000	0.29398
12000	0.349082
13000	0.403938
14000	0.474267
15000	0.540915
16000	0.630308
17000	0.704165
18000	0.78747
19000	0.885611
20000	1.01912
21000	1.06992
22000	1.21971
23000	1.32774
24000	1.42937
25000	1.53783

Mergesort (Eficiencia teórica O(n*log(n))	
Tamaño del vector	Tiempo de ejecución
200000	0.047781
400000	0.102986
600000	0.135755
800000	0.195189
1000000	0.216123
1200000	0.271799
1400000	0.335818
1600000	0.401036
1800000	0.385968
2000000	0.43459
2200000	0.502702
2400000	0.551731
2600000	0.602459
2800000	0.666982
3000000	0.744913
3200000	0.795849
3400000	0.742373
3600000	0.801243
3800000	0.842135
4000000	0.901344
4200000	0.964785
4400000	1.0168
4600000	1.07018
4800000	1.12943
5000000	1.19718

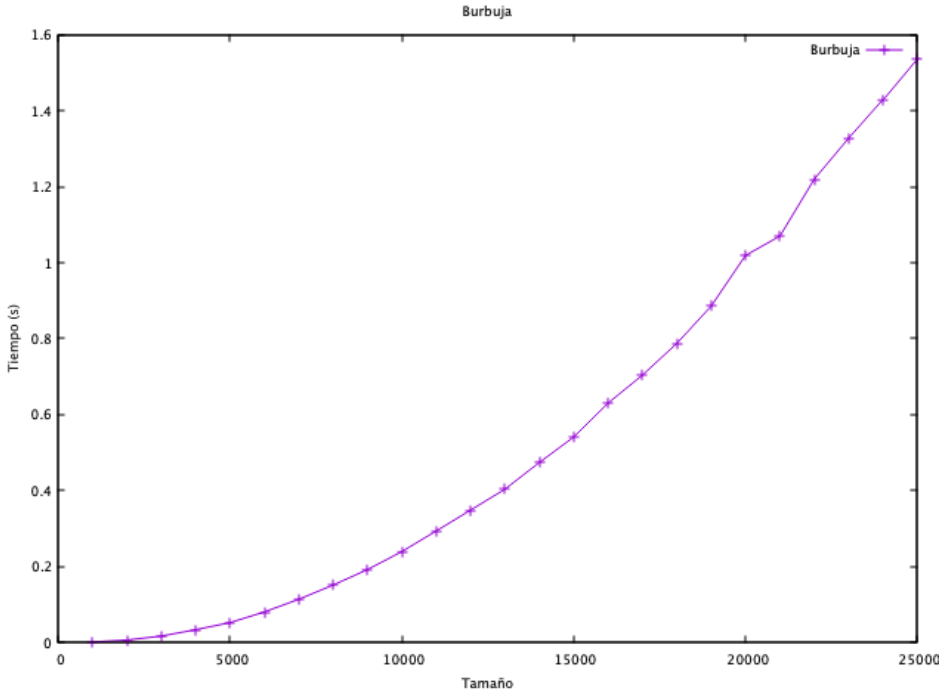
Floyd (Eficiencia teórica $O(n^3)$)	
Nodos del grafo	Tiempo de ejecución
30	0.000196
60	0.001219
90	0.004187
120	0.011034
150	0.020167
180	0.0389
210	0.054249
240	0.077448
270	0.11377
300	0.146427
330	0.197922
360	0.254823
390	0.313639
420	0.390967
450	0.478096
480	0.585113
510	0.690987
540	0.814934
570	0.966867
600	1.10688
630	1.29404
660	1.47337
690	1.68899
720	1.93522
750	2.15256

Torres de Hanoi (Eficiencia teórica $O(2^n)$)	
Número de discos	Tiempo de ejecución
10	8e-06
11	1.3e-05
12	1.9e-05
13	3.6e-05
14	9.1e-05
15	0.000144
16	0.00029
17	0.000609
18	0.001137
19	0.002271
20	0.004856
21	0.011218
22	0.022862
23	0.04167
24	0.081584
25	0.1604
26	0.303922
27	0.601902
28	1.18573
29	2.31288
30	4.6598
31	9.33049
32	18.387
33	36.2806
34	71.9401

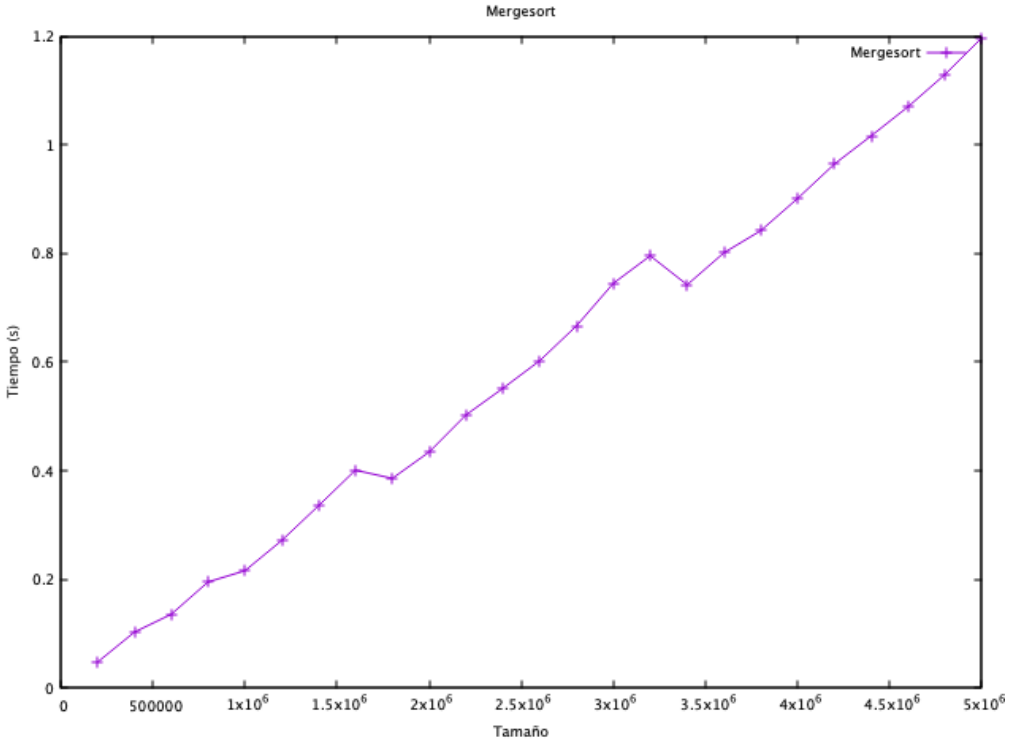
3. Gráficas resultantes

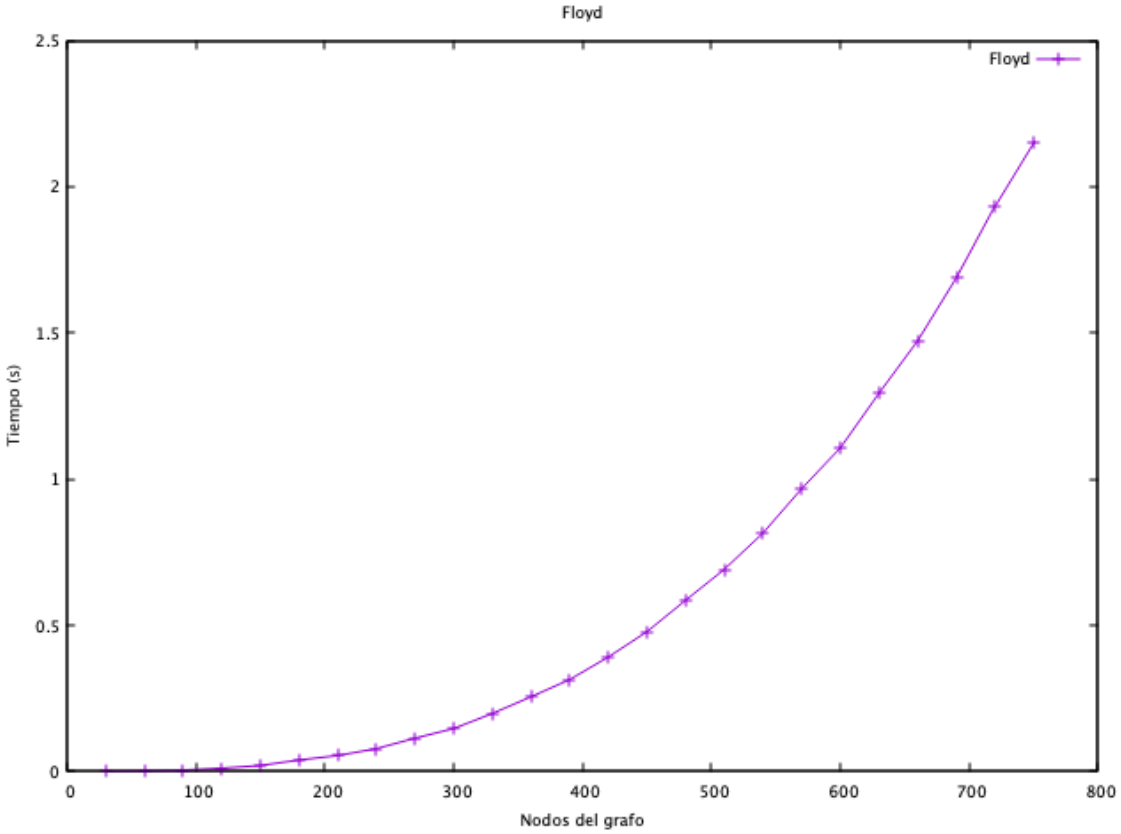
Para cada una de las tablas anteriores generamos el gráfico correspondiente utilizando gnuplot. Para generar las gráficas podemos utilizar:

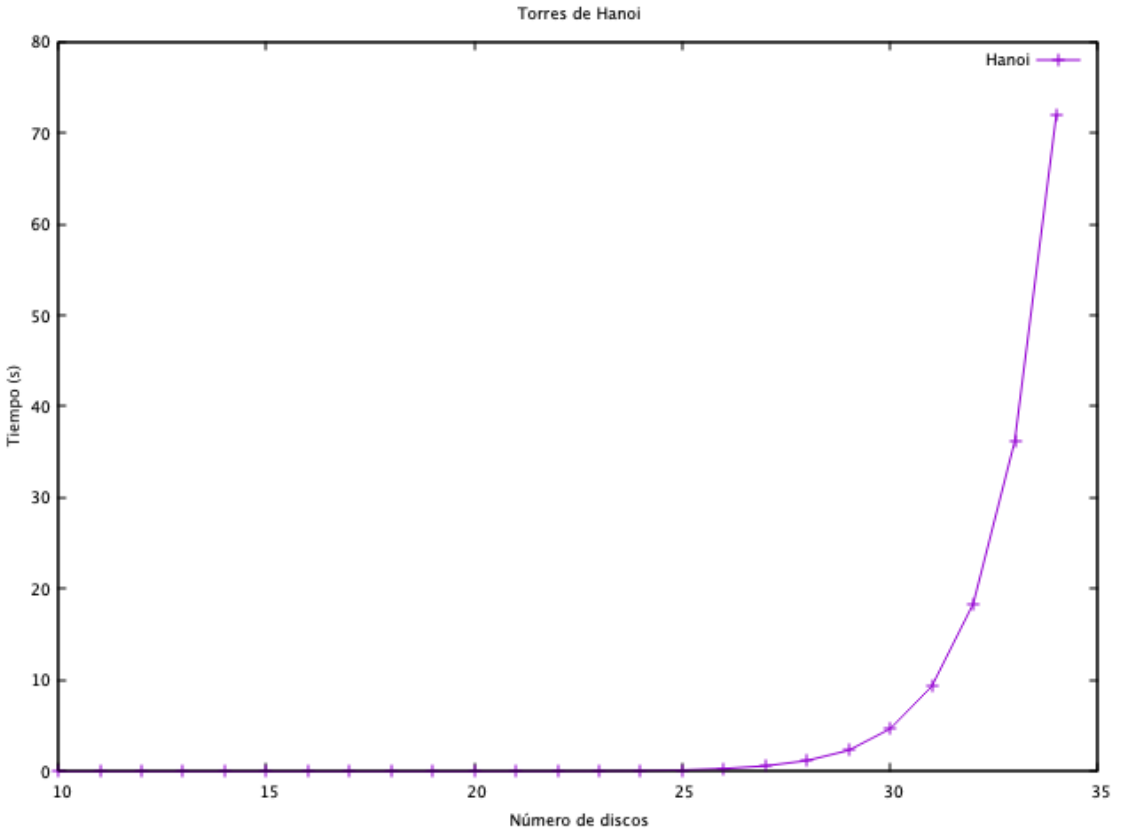
```
gnuplot> set title "Burbuja"
gnuplot> set ylabel "Tiempo (s)"
gnuplot> set xlabel "Tamaño"
gnuplot> plot "salida_burbuja.dat" w lp title "Burbuja"
```



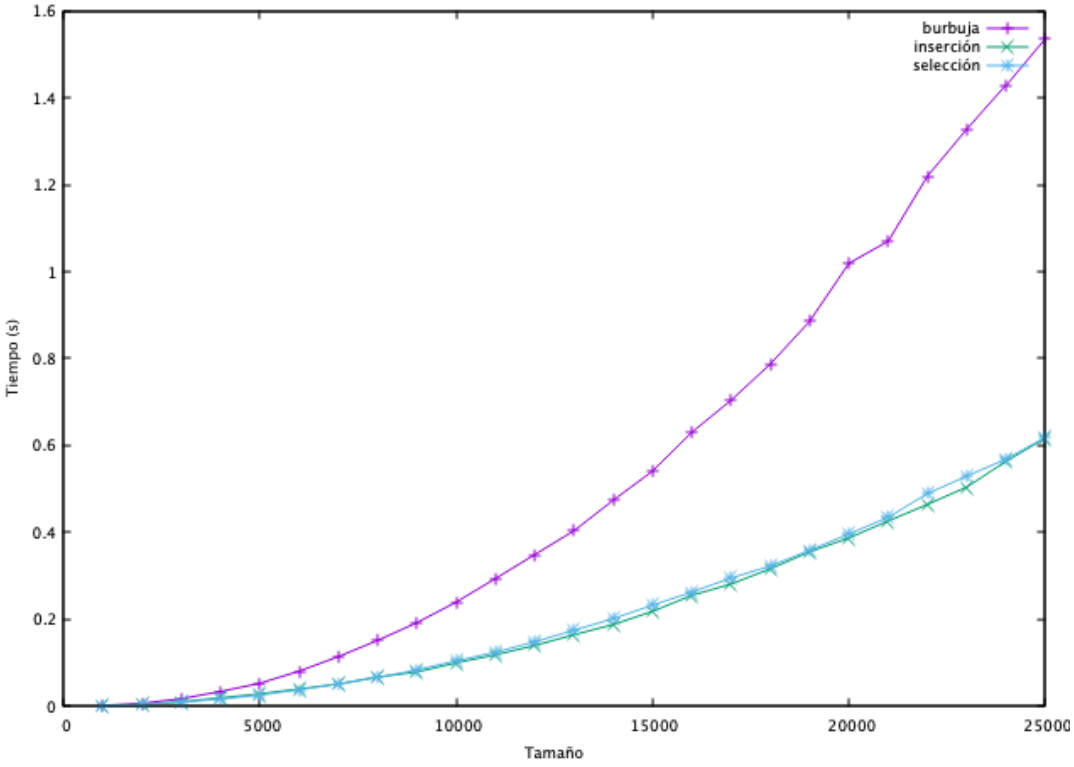
En el siguiente gráfico aparecen dos picos con valor diferente al que deberían tener: Puede deberse a que el planificador del sistema operativo retire la cpu al proceso actual, por lo que la duración de la ejecución aumenta.

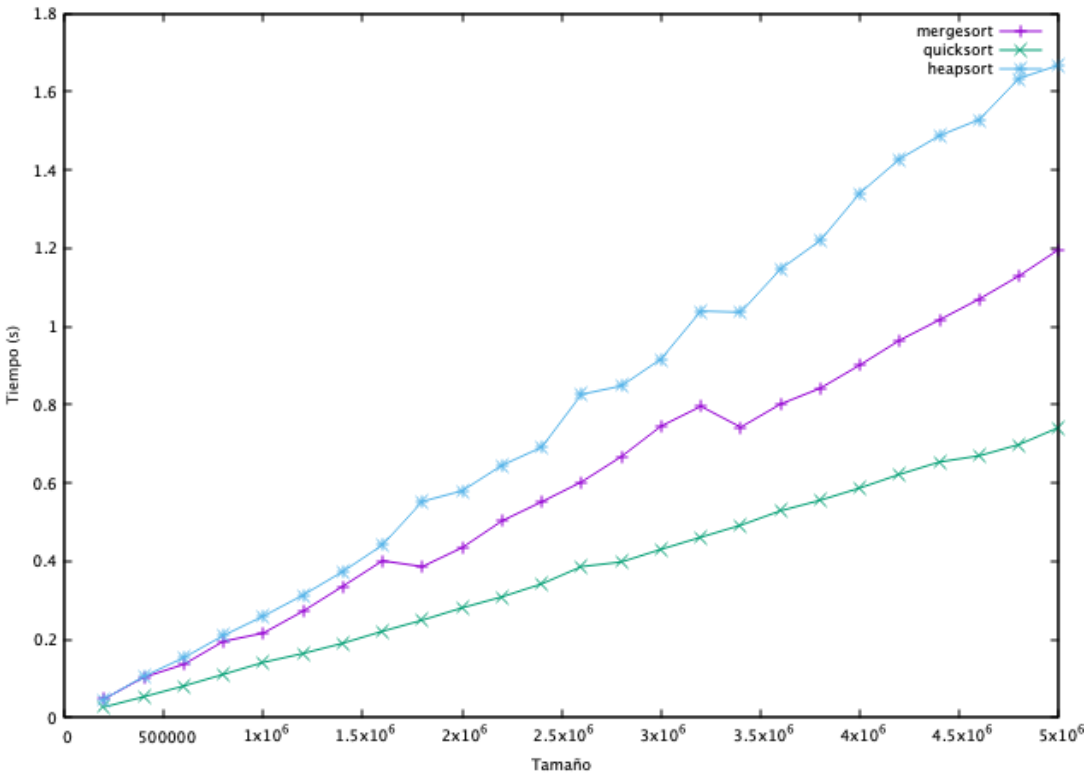



Floyd
Floyd
Tiempo (s)
Nodos del grafo
0
0.5
1
1.5
2
2.5
0
100
200
300
400
500
600
700
800


Torres de Hanoi
Hanoi
Tiempo (s)
Número de discos
0
10
20
30
40
50
60
70
80
10
15
20
25
30
35

En los dos ejemplos inferiores se compara la eficiencia empírica de tres algoritmos con $O(n^2)$ y con $O(n*log(n))$ respectivamente. Los diferentes tiempos en la ejecución de algoritmos con la misma eficiencia teórica se debe a las constantes ocultas que multiplican la ecuación de eficiencia. Por tanto, que tengan tiempos diferentes no significa que tengan eficiencias teóricas diferentes.



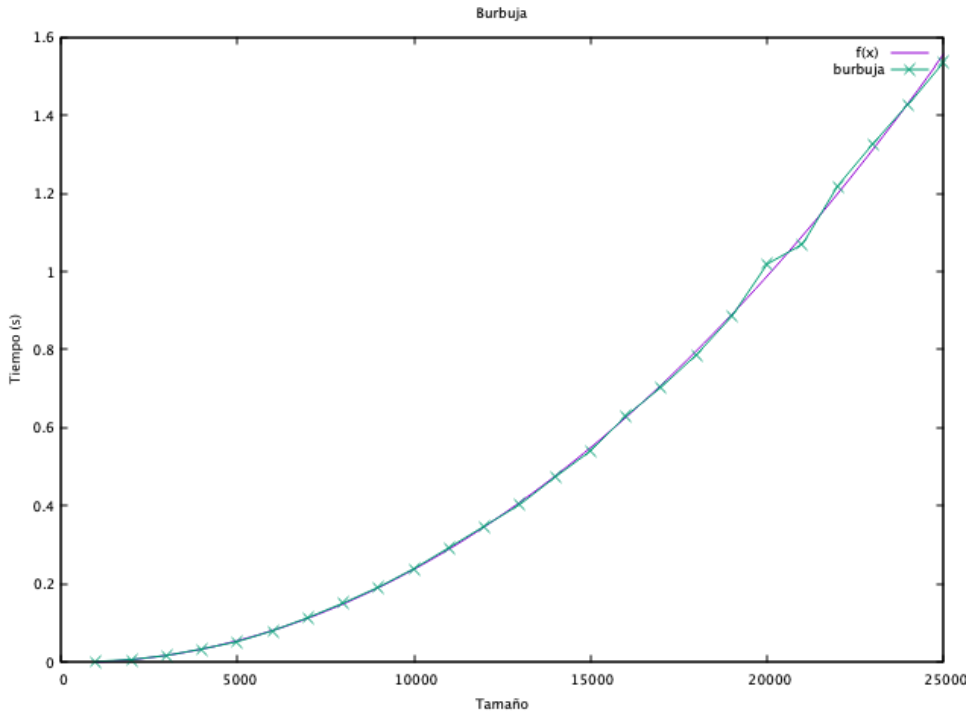


4. Eficiencia híbrida de algunos algoritmos

Como la eficiencia de la ordenación por burbuja es $O(n^2)$, pedimos a gnuplot que nos ajuste los datos a una curva cuadrática f(x) = a*x*x+b*x+c, dejando a su elección los parámetros a, b y c. Para realizarlo aplicamos los siguientes comandos:

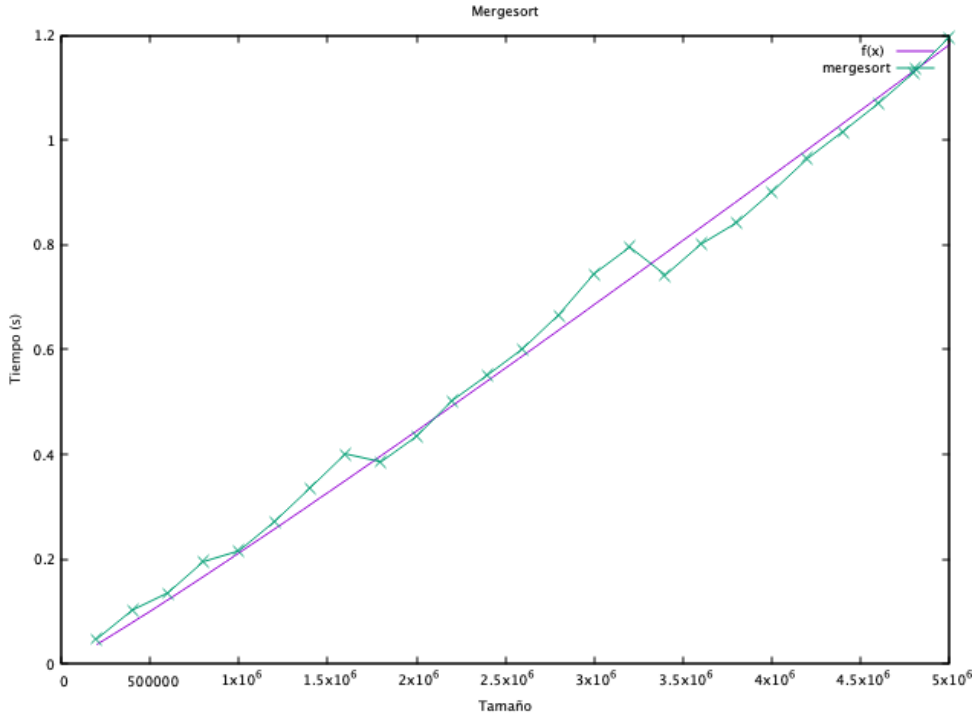
```
gnuplot> f(x) = a*x*x+b*x+c
gnuplot> fit f(x) 'salida_burbuja.dat' via a,b,c
iter      chisq       delta/lim  lambda   a             b             c
   0 7.2649443965e+07   0.00e+00  1.00e+03    5.899669e-06  -1.825772e-03   1.310870e-01
   1 1.3730495608e+04  -5.29e+08  1.00e+02    1.693212e-07  -1.850587e-03   1.310815e-01
   2 9.2718962580e+02  -1.38e+06  1.00e+01    8.368433e-08  -1.662503e-03   1.312331e-01
   3 5.9865095200e+00  -1.54e+07  1.00e+00    9.677761e-09  -1.538348e-04   1.323989e-01
   4 4.0827940726e-02  -1.46e+07  1.00e-01    3.217318e-09  -2.184727e-05   1.273493e-01
   5 4.1341422211e-03  -8.88e+05  1.00e-02    2.688689e-09  -5.728706e-06   2.525948e-02
   6 2.6368537436e-03  -5.68e+04  1.00e-03    2.559868e-09  -1.786847e-06   1.149166e-04
   7 2.6368446605e-03  -3.44e-01  1.00e-04    2.559550e-09  -1.777115e-06   5.283458e-05
iter      chisq       delta/lim  lambda   a             b             c

After 7 iterations the fit converged.
final sum of squares of residuals : 0.00263684
rel. change during last iteration : -3.44466e-06

degrees of freedom    (FIT_NDF)                        : 22
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.0109479
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.000119857

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 2.55955e-09      +/- 4.719e-11    (1.844%)
b               = -1.77711e-06     +/- 1.264e-06    (71.13%)
c               = 5.28346e-05      +/- 0.007132     (1.35e+04%)

correlation matrix of the fit parameters:
                a      b      c
a               1.000
b              -0.971  1.000
c               0.774 -0.884  1.000
gnuplot> plot f(x), "salida_burbuja.dat" w lp title "Burbuja"
```

El resultado es el siguiente, con una muy buena aproximación, confirmando los datos mostrados en la matriz de correlación.



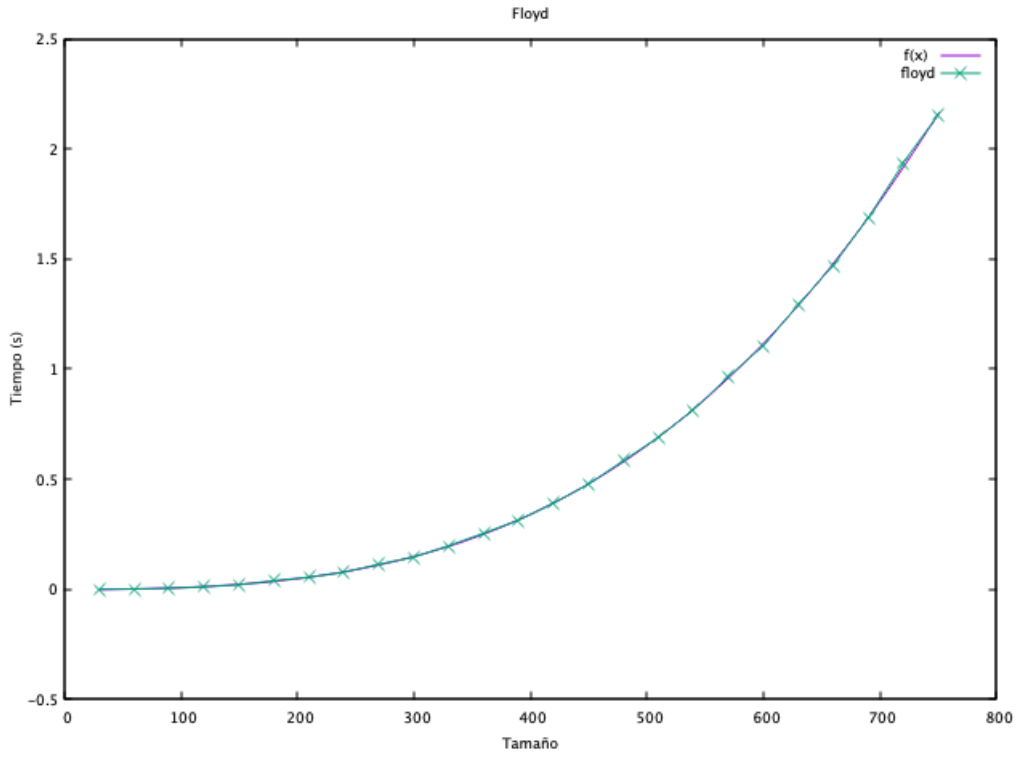
Ahora, con mergesort (O(n*log(n)), la ajustamos a una función f(x) = a*x*log(x) + b.

```
Final set of parameters            Asymptotic Standard Error
======================             ==========================
a               = 1.02826e-08      +/- 1.632e-10    (1.587%)
b               = 0.0258289        +/- 0.01058      (40.96%)

correlation matrix of the fit parameters:
                a      b
a               1.000
b              -0.865  1.000
```



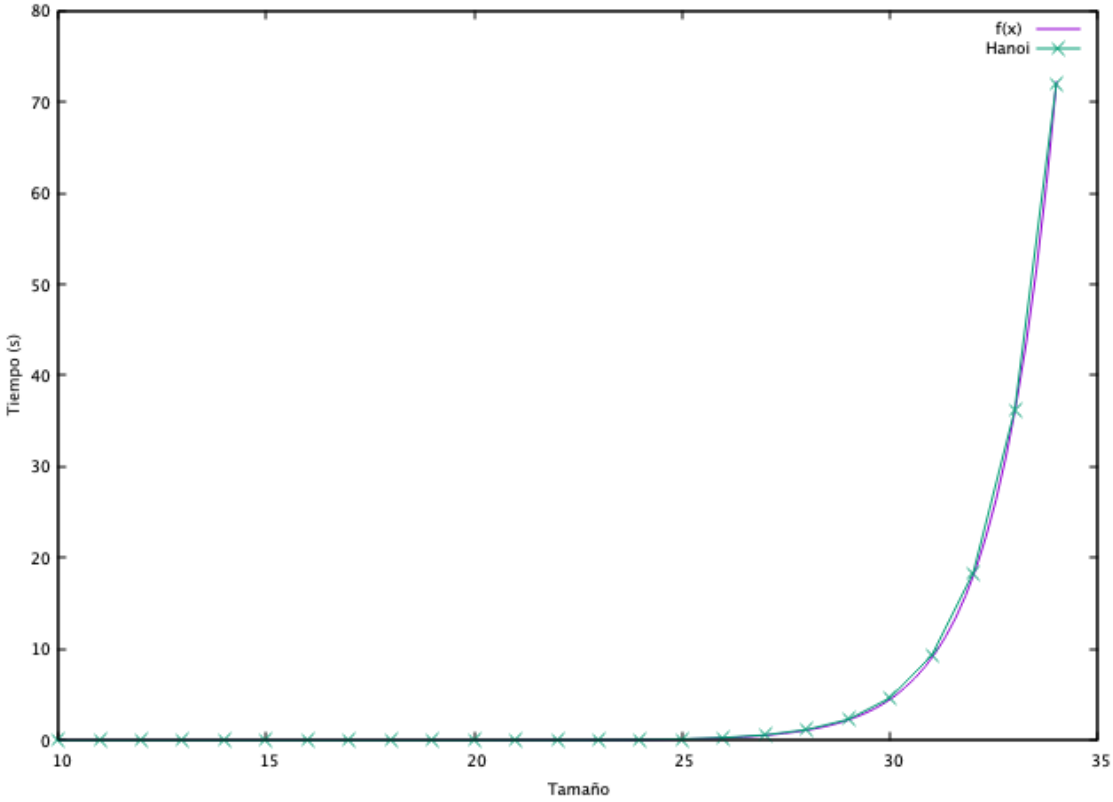
Para Floyd ($O(n^3)$), ajustamos a f(x) = a*x*x*x+b*x*x+c*x+d.

```
Final set of parameters            Asymptotic Standard Error
======================             ==========================
a               = 5.06658e-09      +/- 1.571e-10    (3.101%)
b               = -2.82283e-08     +/- 1.862e-07    (659.6%)
c               = 6.02115e-05      +/- 6.316e-05    (104.9%)
d               = -0.00335768      +/- 0.005802     (172.8%)

correlation matrix of the fit parameters:
                a      b      c      d
a               1.000
b              -0.987  1.000
c               0.926 -0.973  1.000
d              -0.719  0.795 -0.898  1.000
```



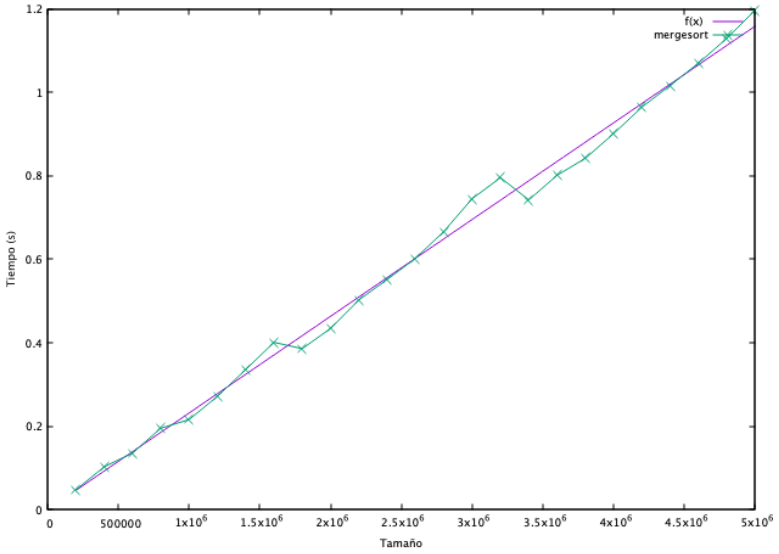
Para las torres de Hanoi, $f(x) = a*2^n+b$:

```
Final set of parameters            Asymptotic Standard Error
======================             ========================
a               = 4.20107e-09      +/- 6.503e-12    (0.1548%)
b               = -2.82282e-08     +/- 0.0258       (9.14e+07%)

correlation matrix of the fit parameters:
                a      b
a               1.000
b              -0.346  1.000
```



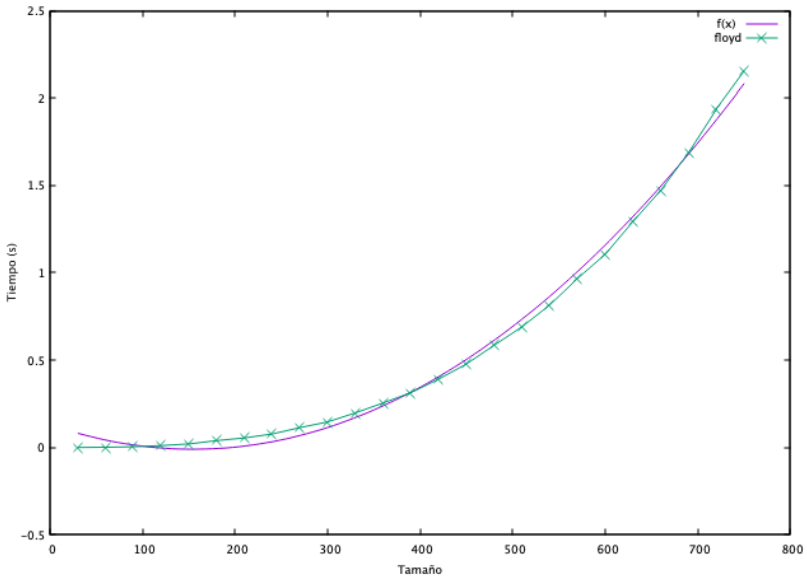
Si decidimos aplicar un ajuste diferente a mergesort, por ejemplo, y utilizamos un ajuste lineal, el ajuste es menos preciso:

```
Final set of parameters            Asymptotic Standard Error
======================             ==========================
a               = 2.31773e-07      +/- 3.696e-09    (1.595%)
b               = -2.90623e-08     +/- 0.01099      (3.781e+07%)

correlation matrix of the fit parameters:
                a      b
a               1.000
b              -0.874  1.000
```



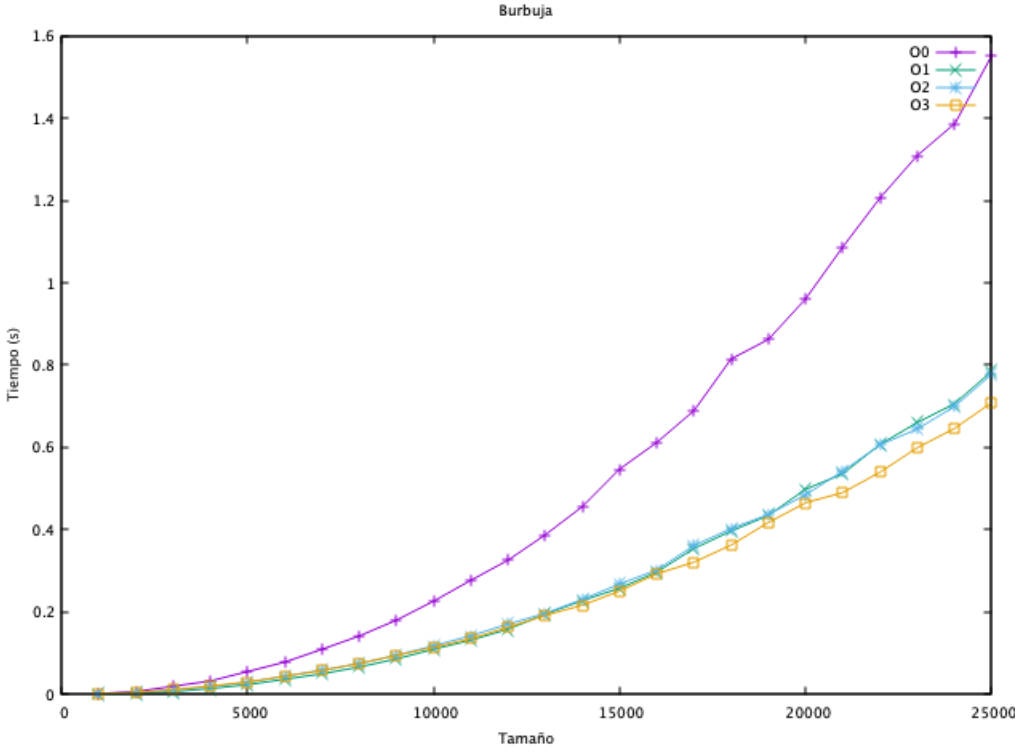
En el caso de floyd con un ajuste cuadrático tampoco coincide exactamente:

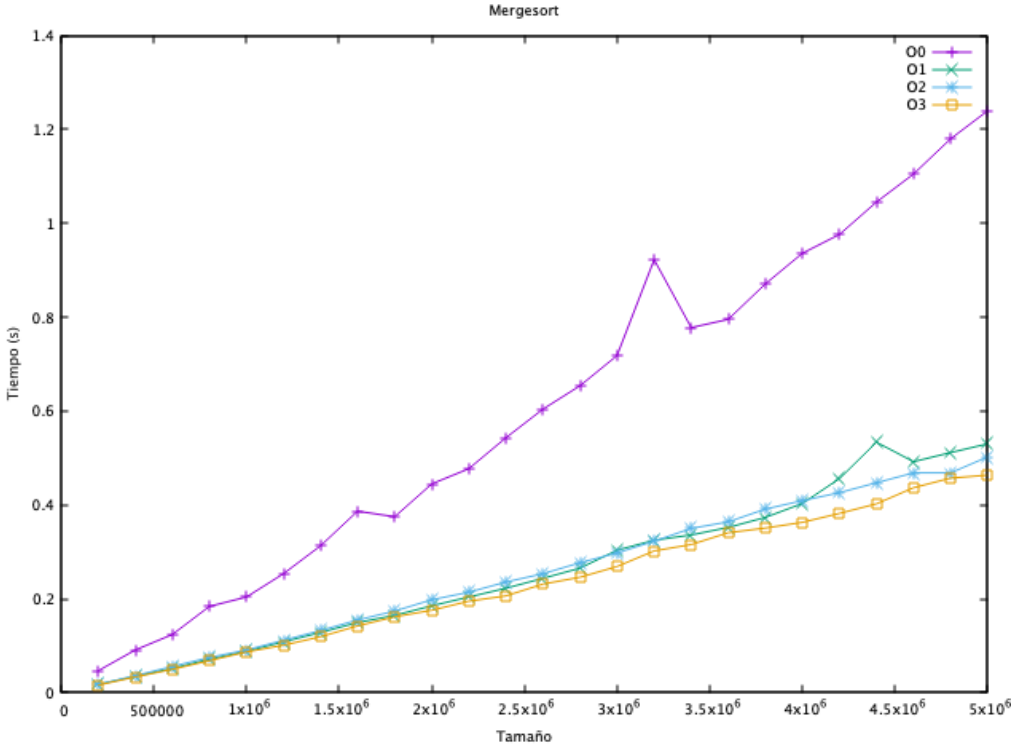
```
Final set of parameters            Asymptotic Standard Error
======================             ==========================
a               = 5.89967e-06      +/- 2.06e-07     (3.491%)
b               = -0.00182577      +/- 0.0001655    (9.065%)
c               = 0.131087         +/- 0.02801      (21.37%)

correlation matrix of the fit parameters:
                a      b      c
a               1.000
b              -0.971  1.000
c               0.774 -0.884  1.000
```

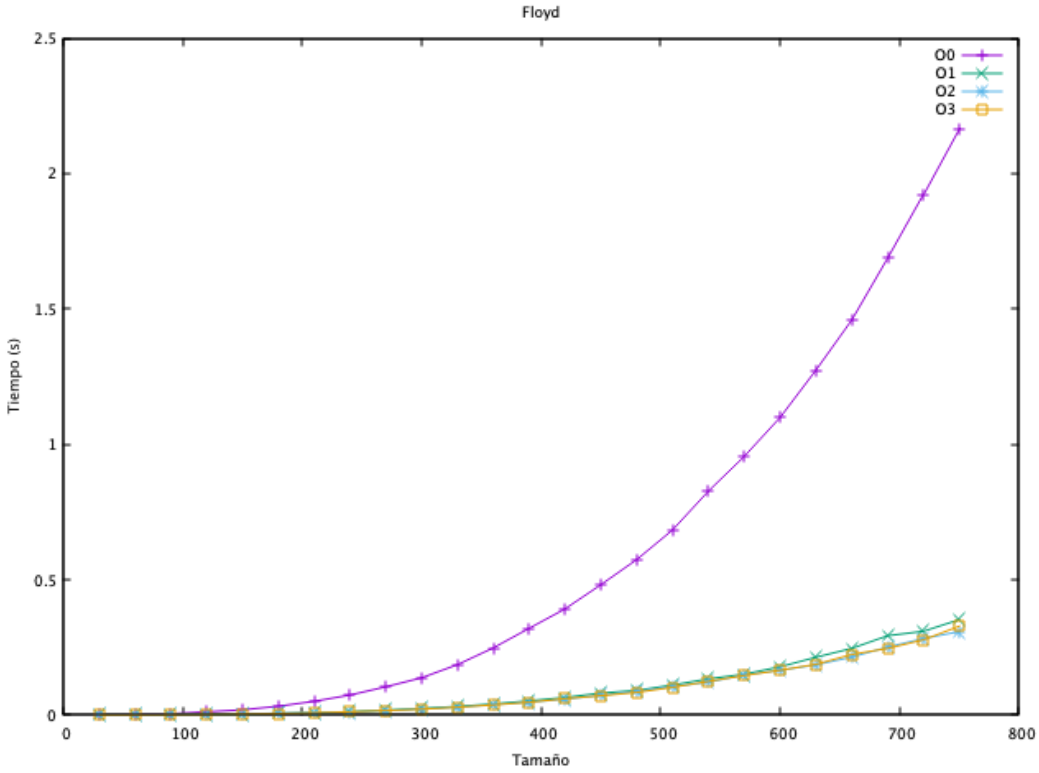


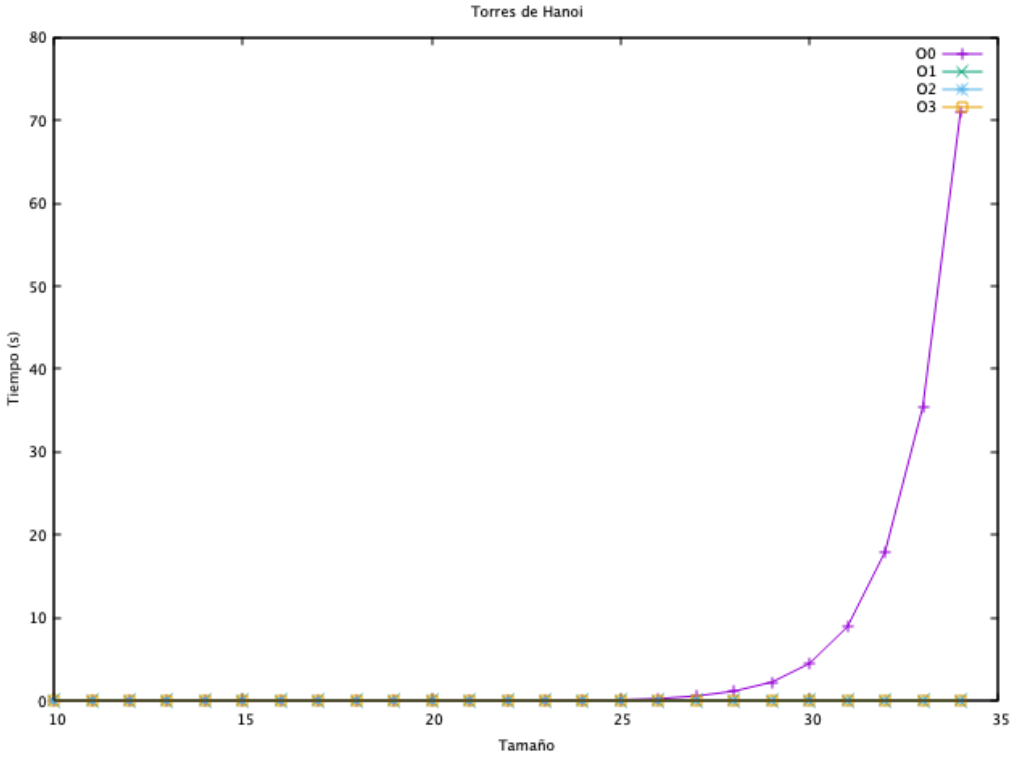
5. Eficiencia empírica y parámetros externos:

La eficiencia empírica como su nombre indica se calcula con una base experimental. Y la situación donde se realiza el experimento influye en una gran medida en el resultado final. Vamos a demostrarlo mostrando como la optimización en la compilación puede afectar: Con el compilador g++, vamos a aplicar los distintos flags de eficiencia -O0, -O1, -O2 y -O3 para analizar los resultados con cada uno.









Podemos observar como la utilización de optimizaciones de compilación puede dar lugar a grandes diferencias en el tiempo de ejecución.

6. Conclusión

En la práctica hemos demostrado como la eficiencia empírica se ajusta en gran medida a la eficiencia teórica. También hemos observado como factores del propio sistema donde se ejecutan los algoritmos afectan a esta eficiencia empírica, en lo que llamamos las constantes ocultas, y que aplicando optimizaciones en el compilador se pueden reducir ampliamente los tiempos de ejecución.