Guión de Prácticas. Algorítmica.

Práctica 1: Análisis de eficiencia de algoritmos

Pedro Bedmar López

P1

Índice:

# Planteamiento del problema

# Tablas con eficiencia empírica de algunos algoritmos

# Gráficas resultantes

# Eficiencia híbrida de algunos algoritmos

# Eficiencia empírica y parámetros externos

# Conclusión

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.

1. 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.



|  |  |
| --- | --- |
| Burbuja (Eficiencia teórica O(n2)) | |
| 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 |

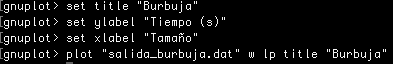
|  |  |
| --- | --- |
| Mergesort (Eficiencia teórica O(n\*log(n)) | |
| Tamaño del vector | Tiempo de ejecución |
| 200000 | 0.047781 |
| 400000 | 0.102986 |
| 600000 | 0.135755 |
| 800000 | 0.195189 |
| 1000000 | 0.216123 |
| 1200000 | 0.271799 |
| 1400000 | 0.335818 |
| 1600000 | 0.401036 |
| 1800000 | 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 |

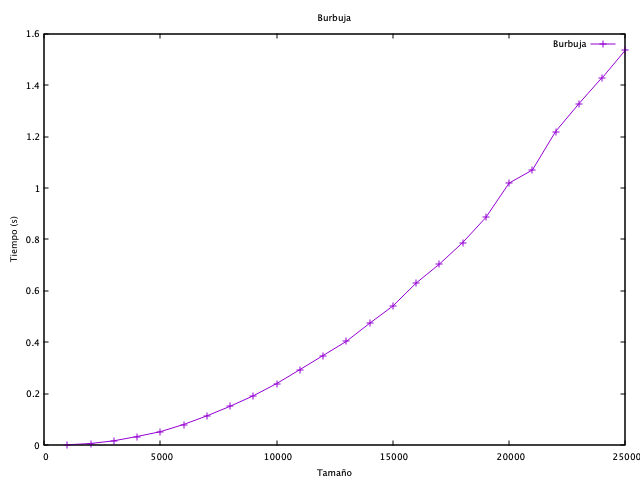
|  |  |
| --- | --- |
| Floyd (Eficiencia teórica O(n3)) | |
| Nodos del grafo | Tiempo de ejecución |
| 30 | 0.000196 |
| 60 | 0.001219 |
| 90 | 0.004187 |
| 120 | 0.011034 |
| 150 | 0.020167 |
| 180 | 0.0389 |
| 210 | 0.054249 |
| 240 | 0.077448 |
| 270 | 0.11377 |
| 300 | 0.146427 |
| 330 | 0.197922 |
| 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(2n)) | |
| 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 |

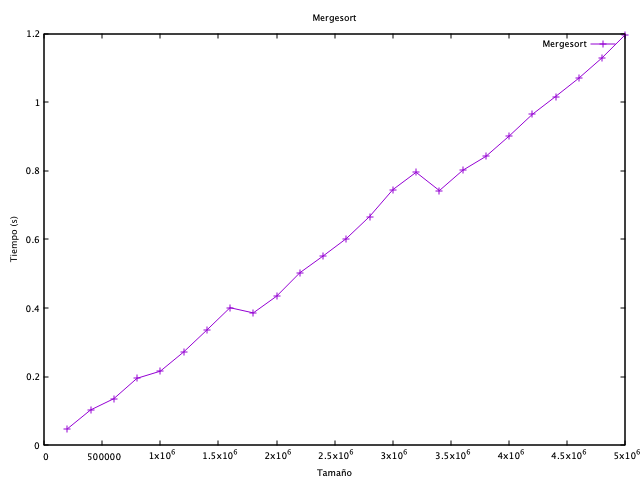
1. Gráficas resultantes

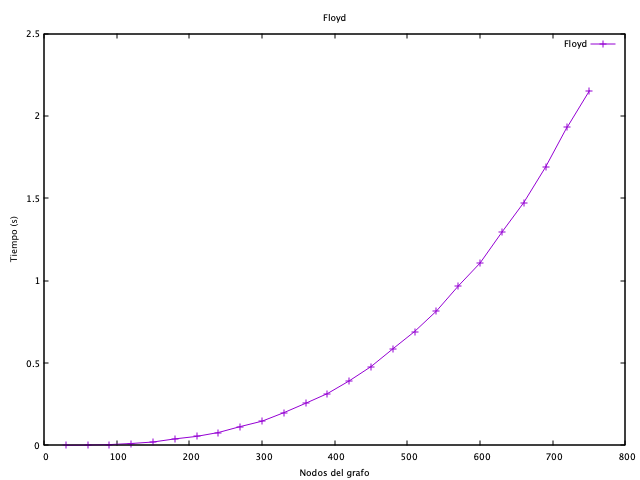
Para cada una de las tablas anteriores generamos el gráfico correspondiente utilizando gnuplot. Para generar las gráficas podemos utilizar:

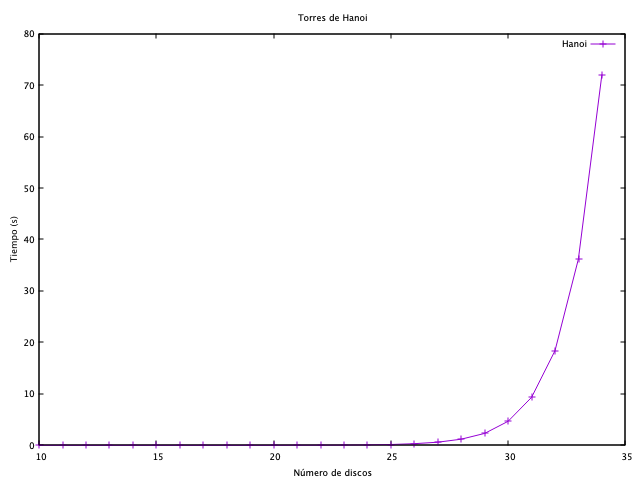




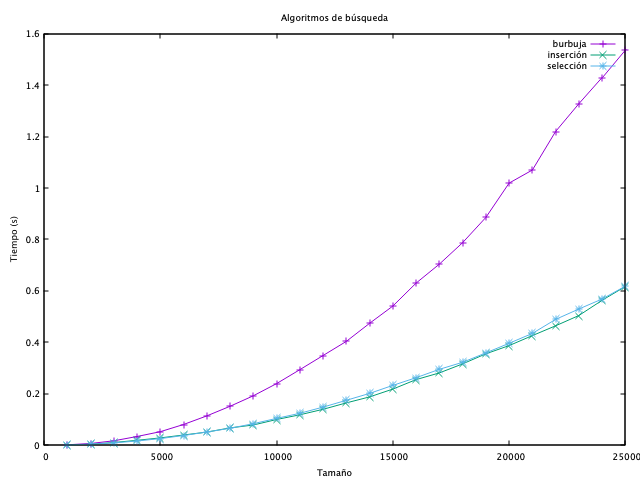
En el siguiente gráfico aparecen dos picos con valor diferente al que deberían tener: Puede deberse a que el planificador del sistema operativo retire la cpu al proceso actual, por lo que la duración de la ejecución aumenta.

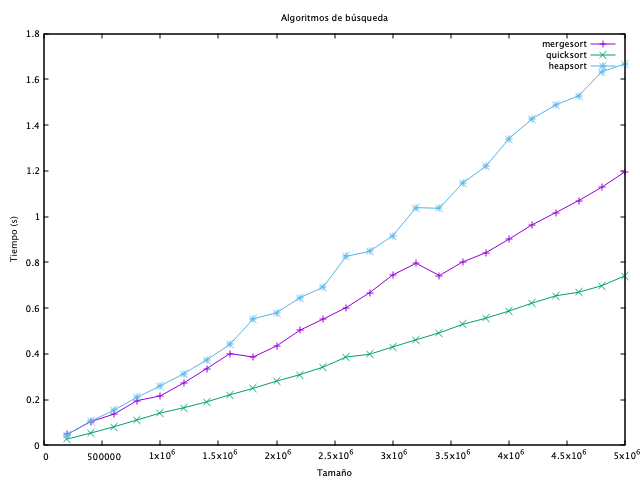






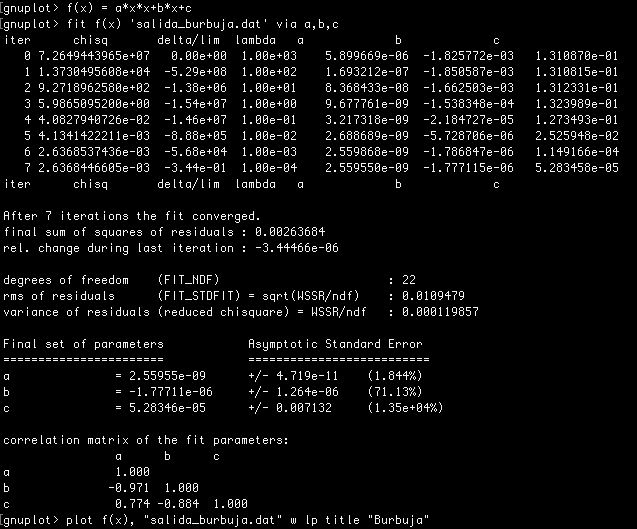
En los dos ejemplos inferiores se compara la eficiencia empírica de tres algoritmos con O(n2) y con O(n\*log(n)) respectivamente. Los diferentes tiempos en la ejecución de algoritmos con la misma eficiencia teórica se debe a las constantes ocultas que multiplican la ecuación de eficiencia. Por tanto, que tengan tiempos diferentes no significa que tengan eficiencias teóricas diferentes.



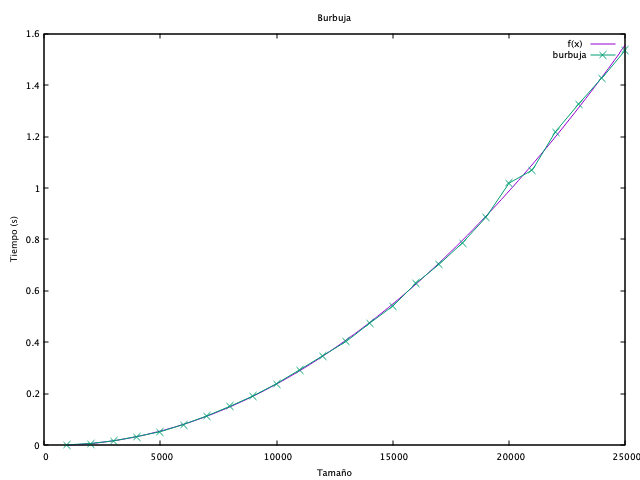


# Eficiencia híbrida de algunos algoritmos

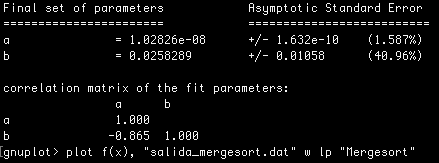
Como la eficiencia de la ordenación por burbuja es O(n2), pedimos a gnuplot que nos ajuste los datos a una curva cuadrática f(x) = a\*x\*x+b\*x+c, dejando a su elección los parámetros a, b y c. Para realizarlo aplicamos los siguientes comandos:

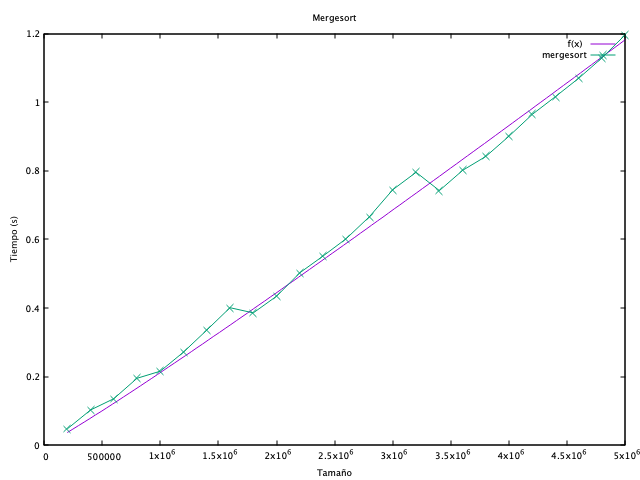


El resultado es el siguiente, con una muy buena aproximación, confirmando los datos mostrados en la matriz de correlación.

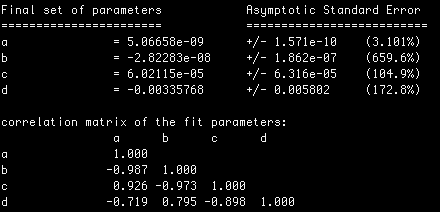


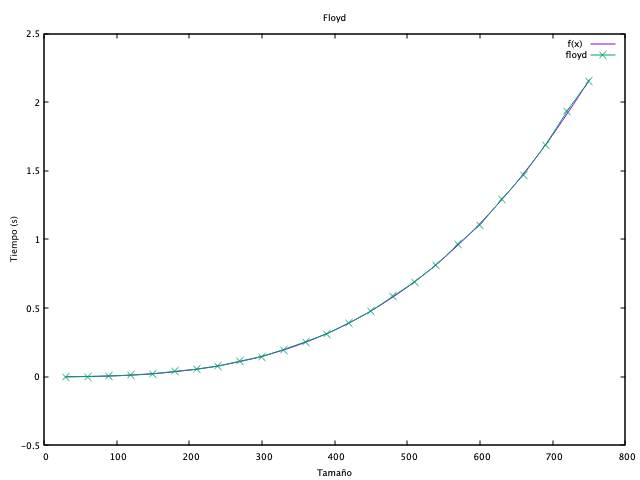
Ahora, con mergesort (O(n\*log(n)), la ajustamos a una función f(x) = a\*x\*log(x) + b.



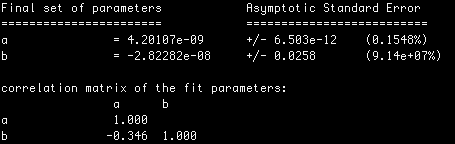


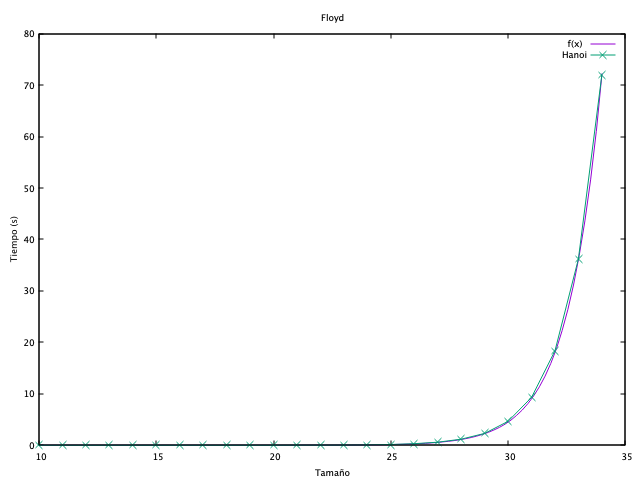
Para Floyd (O(n3)), ajustamos a f(x) = a\*x\*x\*x+b\*x\*x+c\*x+d.



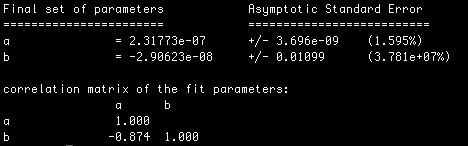


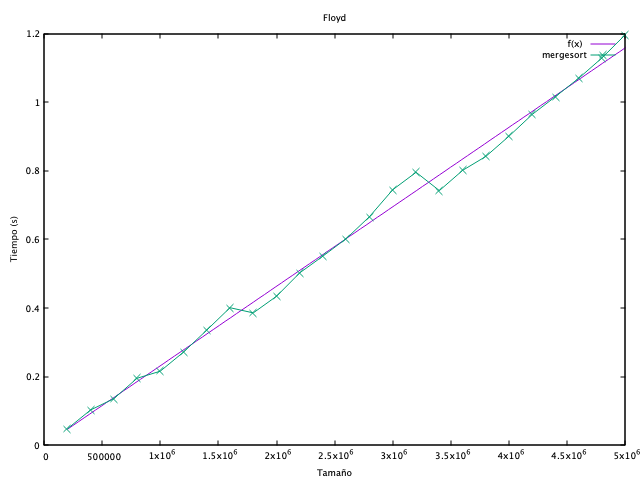
Para las torres de Hanoi, f(x) = a\*2n+b:



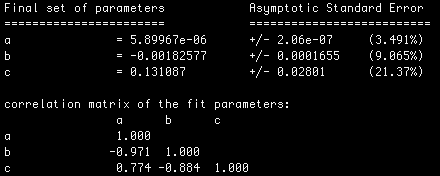


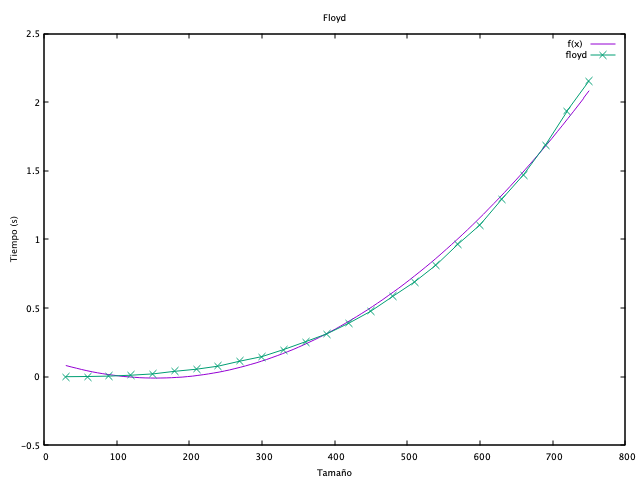
Si decidimos aplicar un ajuste diferente a mergesort, por ejemplo, y utilizamos un ajuste lineal, el ajuste es menos preciso:





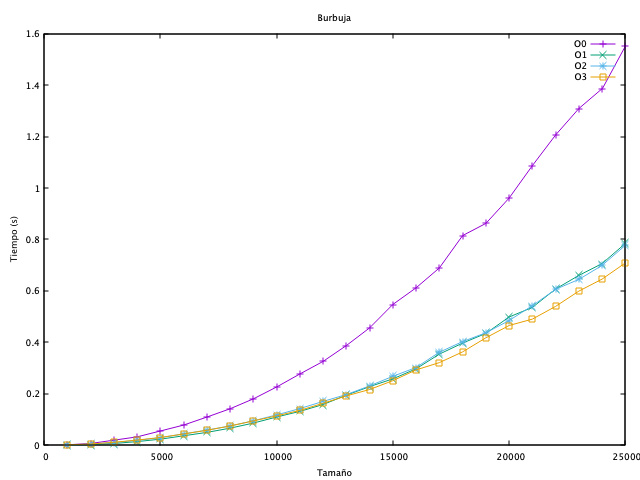
En el caso de floyd con un ajuste cuadrático tampoco coincide exactamente:

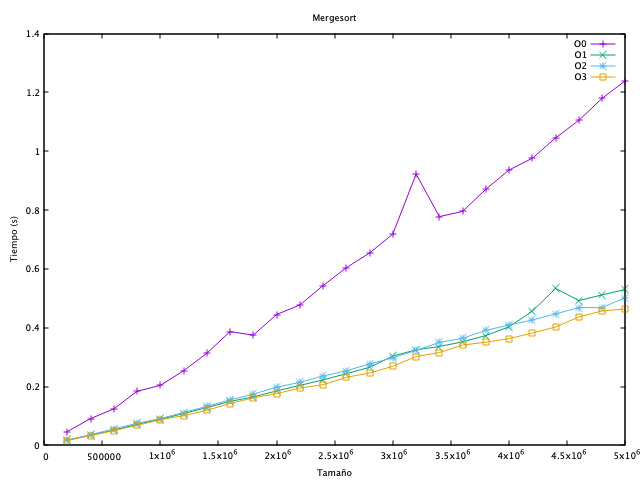


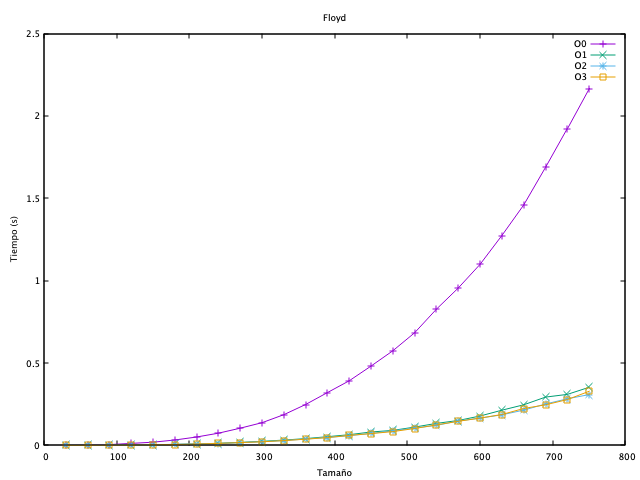


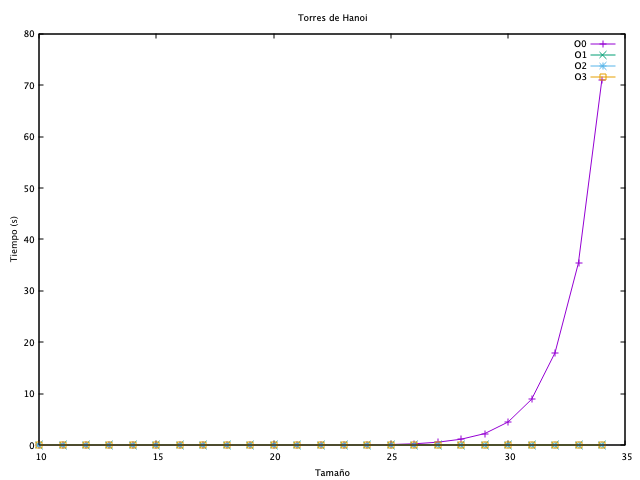
1. Eficiencia empírica y parámetros externos:

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.