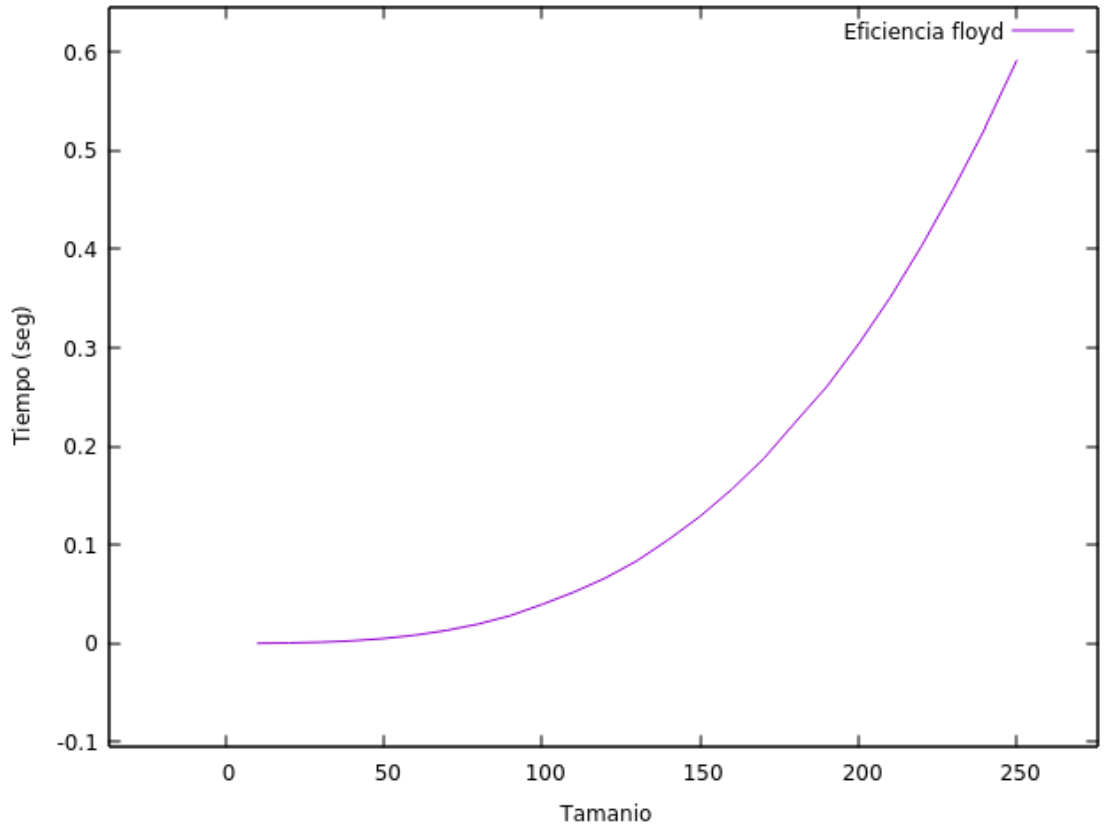
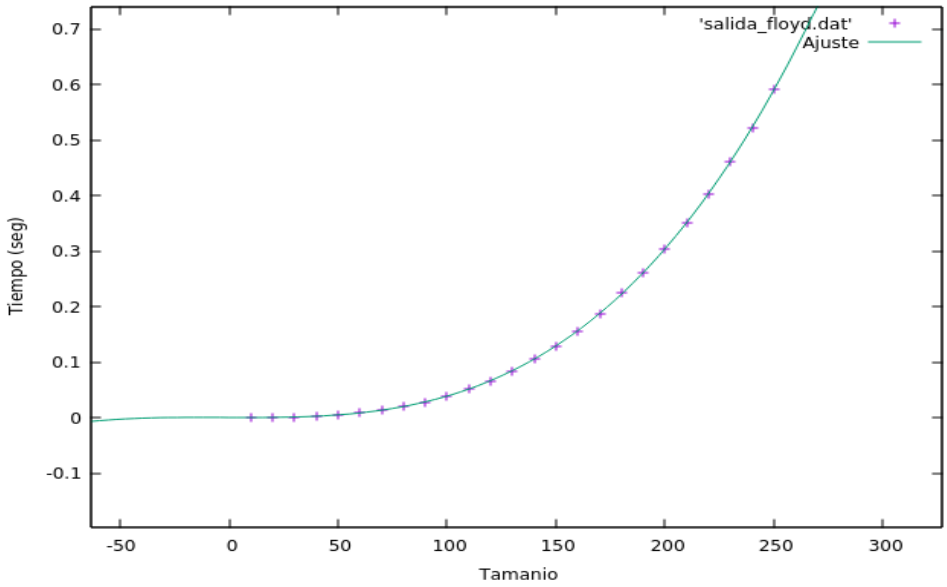
1. Calcule la eficiencia empírica, siguiendo las indicaciones de la sección 3. Defina adecuadamente los tamaños de entrada de forma tal que se generen al menos 25 datos. Incluya en la memoria tablas diferentes para los algoritmos de distinto orden de eficiencia (una con los algoritmos de orden O(n 2), otra con los O(n log n), otra con O(n 3) y otra con O(2n)).

Para el cálculo empírico de los algoritmos, nos basamos en la recolección de los tiempos para distintos tamaños de entrada. Por ejemplo, para la ordenación, el tamaño corresponde con el número de elementos a ordenar del vector. En este caso, se han planteado a través de un script, 25 tamaños distintos para cada algoritmo, adaptados a su eficiencia para evitar ejecuciones muy largas. A continuación, una recopilación de capturas de la representación en gráficas de cada uno:

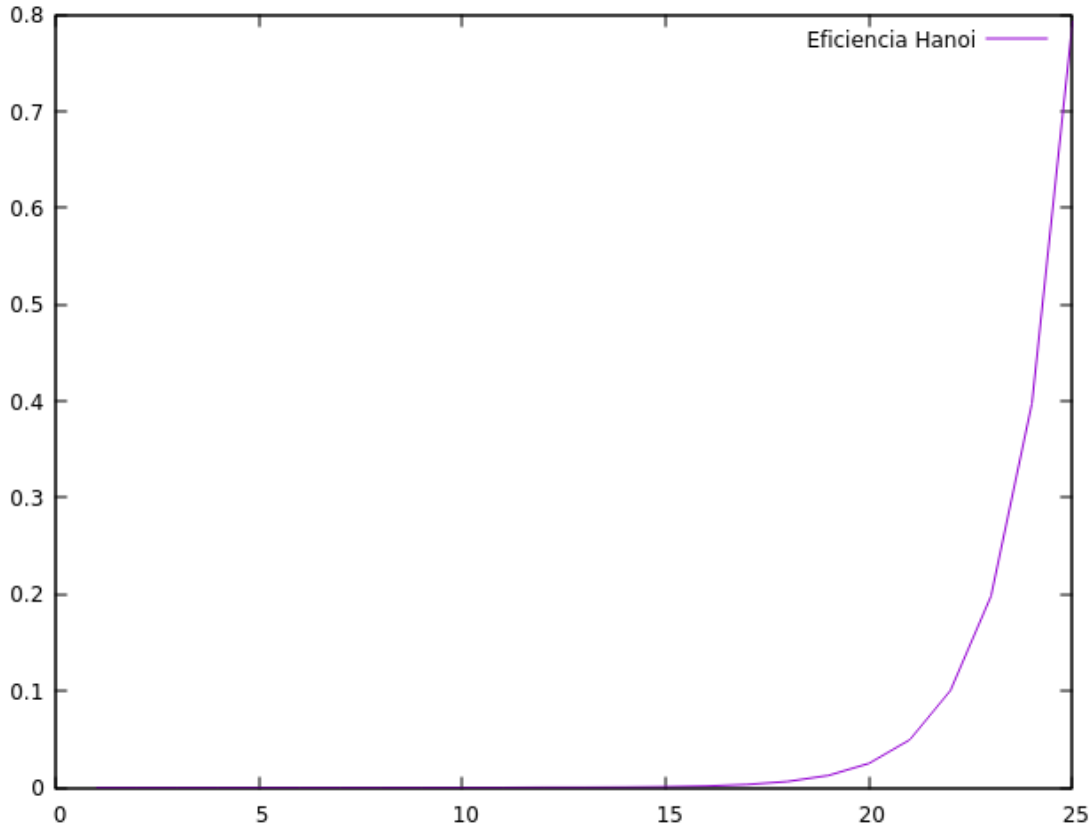
ALGORITMO DE FLOYD: Para tamaños de 10 a 250, con un incremento de 10 en 10, esta es la representación de la eficiencia de Floyd.



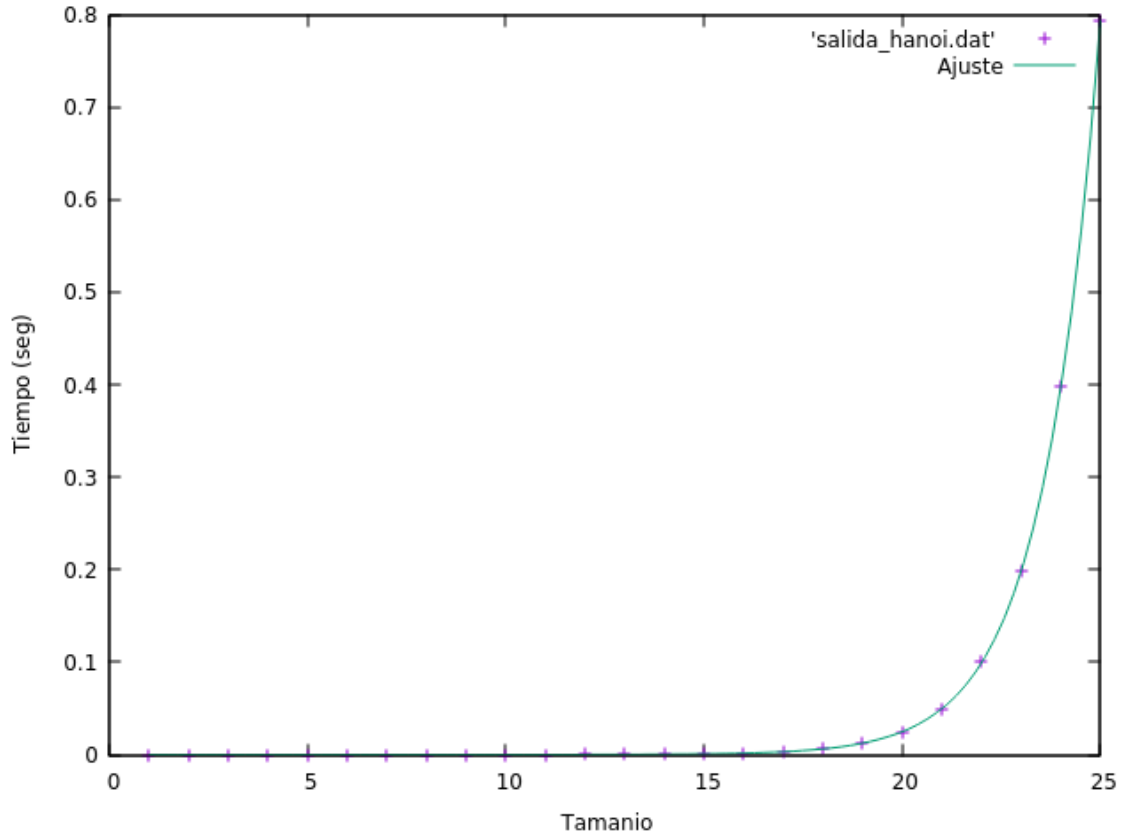
ALGORITMO DE FLOYD COMPARADO CON SU TEÓRICA: Tras el ajuste con su eficiencia teórica (n^3), esta es la comparación entre lo que debería tardar en ejecutar para cada tamaño, y lo que tarda realmente.



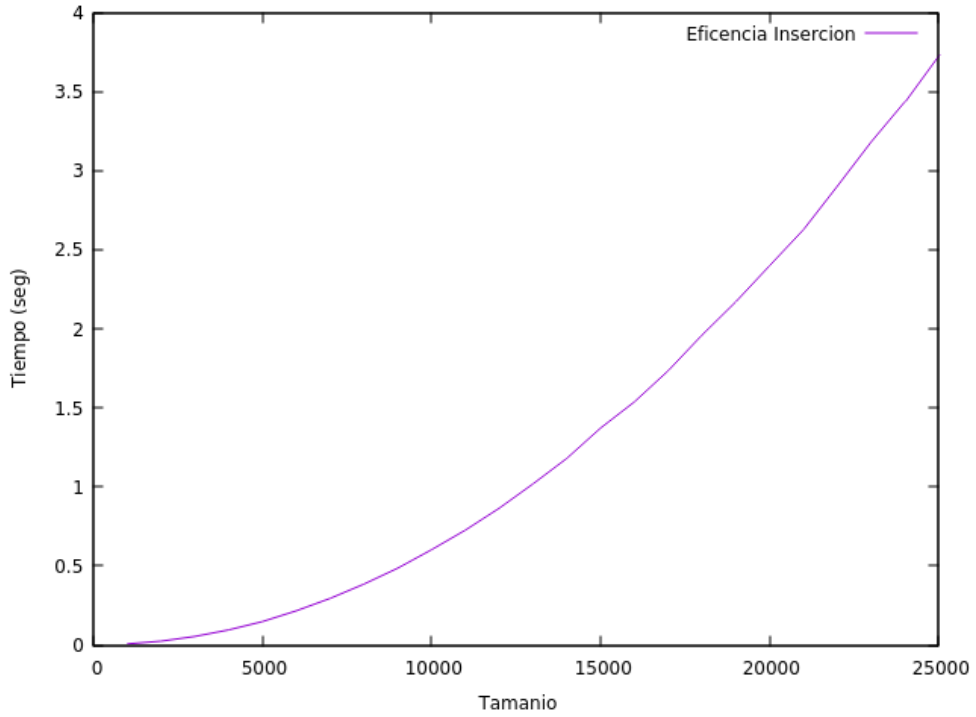
ALGORITMO DE HANOI: Al tener una eficiencia teórica que aumenta tan rápido, los tamaños debían ser más pequeños, partiendo de 0 y terminando en 25.



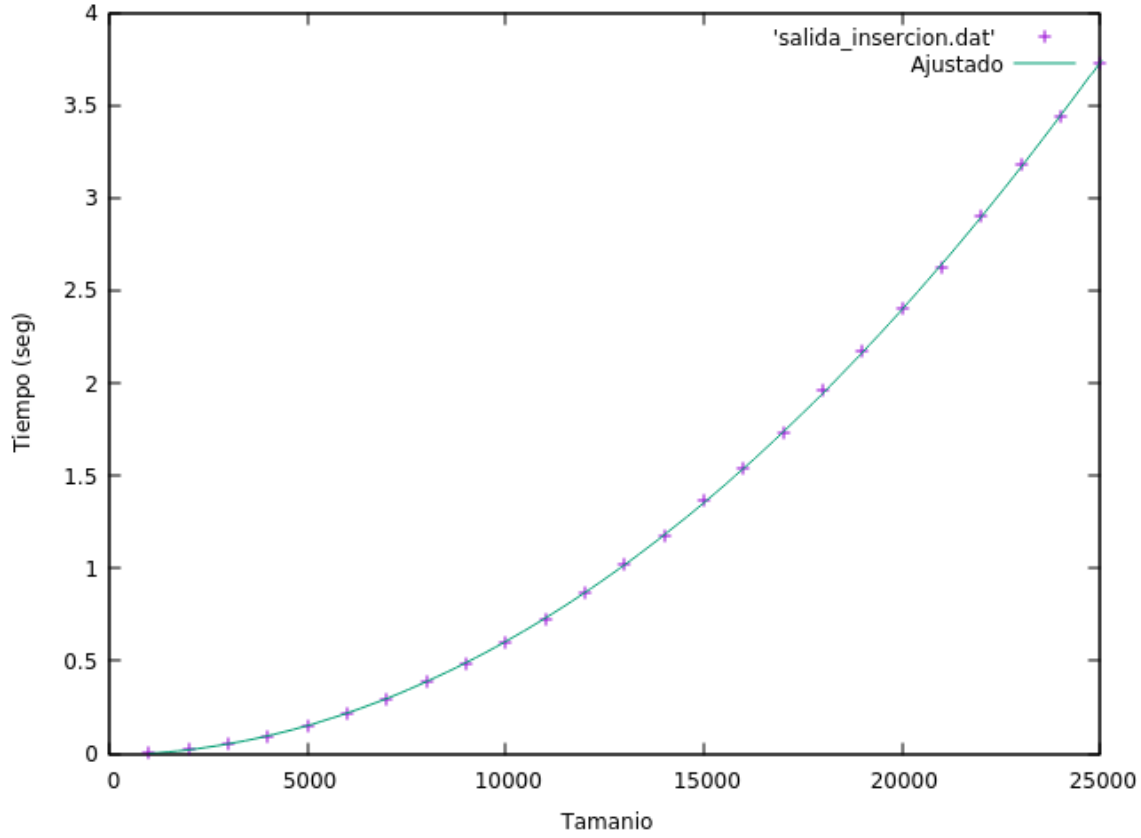
ALGORITMO DE HANOI COMPARADO CON SU TEÓRICA: Su teórica consiste en 2^n, y el ajuste entre mis resultados y esta, queda tal que así.



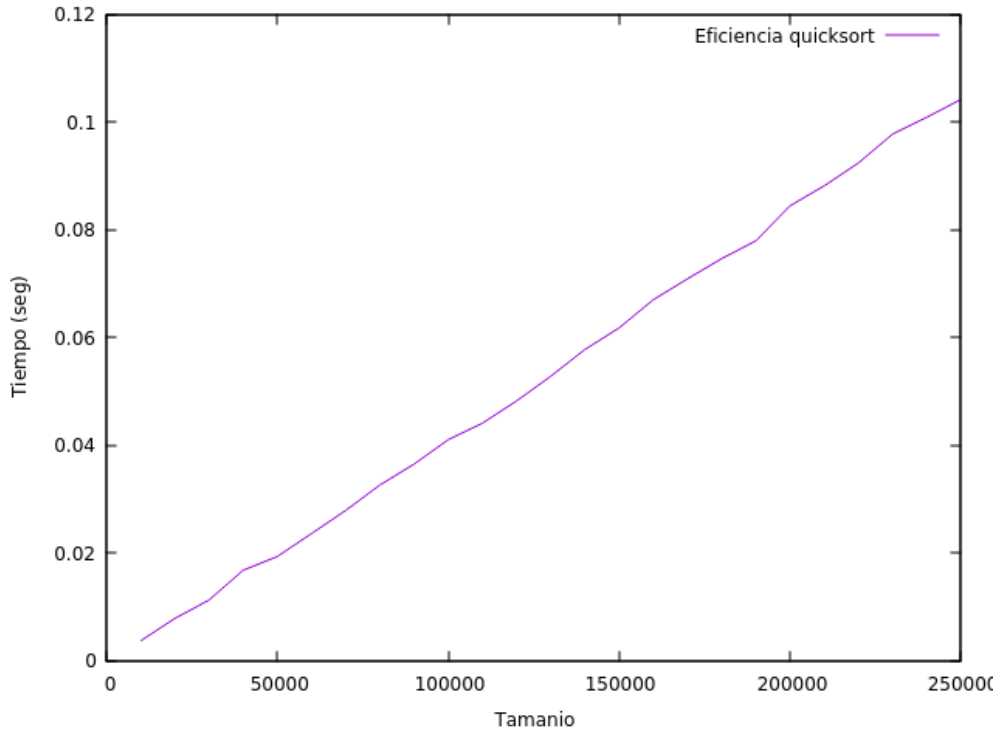
ALGORITMO DE INSERCIÓN: Este es el algoritmo de ordenación con eficiencia n^2 elegido por mí para esta práctica. Su eficiencia permite tamaños más grandes y por ello van desde 1000 hasta 25000 elementos en el vector a ordenar.



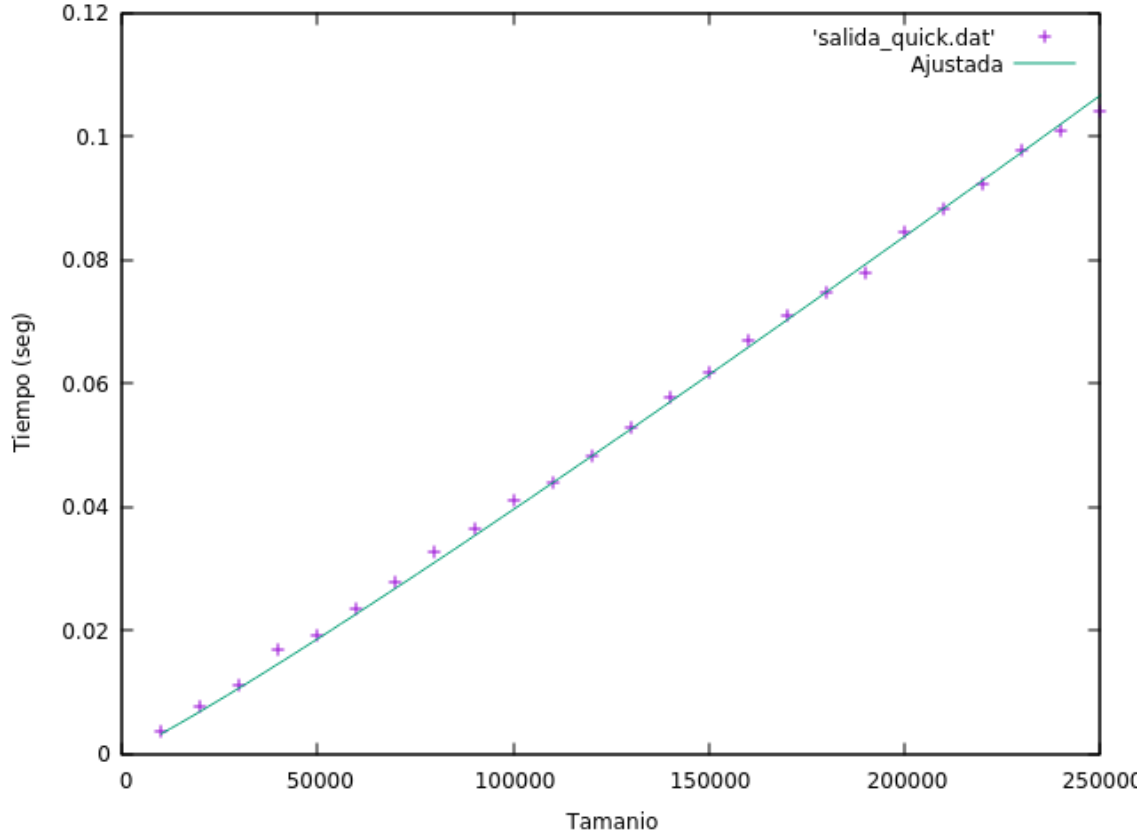
ALGORITMO DE INSERCIÓN COMPARADA CON SU TEÓRICA: Tras el ajuste, este es el resultado, con pequeños desvíos en tamaños grande, pero por lo general realmente acertada



ALGORITMO DE QUICKSORT: El otro algoritmo de ordenación seleccionado ha sido este, con una eficiencia n*log(n). Al ser la mejor eficiencia de todas, sus tamaños parten desde 10000 terminando en 250000.



ALGORITMO DE QUICKSORT COMPARADA CON SU TEÓRICA: De todas, es la que menos se ajusta y aun así la aproximación es certera.



TABLAS CLASIFICADAS POR EFICIENCIA

Eficiencia $O(n^2)$

Tamaño	burbuja	inserción	selección
1000	0,009326	0,006717	0,0039843
2000	0,036214	0,024055	0,0161128
3000	0,081513	0,05369	0,035522
4000	0,143469	0,094589	0,065098
5000	0,22451	0,148125	0,098737
6000	0,322415	0,215842	0,14236
7000	0,448451	0,294876	0,197098
8000	0,584871	0,384413	0,257235
9000	0,743189	0,485826	0,334225
10000	0,907302	0,602935	0,396813
11000	1,10214	0,726386	0,480536
12000	1,31375	0,864664	0,573581
13000	1,54938	1,01861	0,680197
14000	1,77838	1,17975	0,783223
15000	2,05705	1,37162	0,896243
16000	2,33831	1,53765	1,0198
17000	2,64727	1,73426	1,15057
18000	3,00253	1,96032	1,28965
19000	3,29181	2,17137	1,43709
20000	3,70991	2,39982	1,59251
21000	4,0858	2,6275	1,76388
22000	4,46812	2,90237	1,9482
23000	4,81371	3,18144	2,11209
24000	5,17747	3,43809	2,29516
25000	5,62167	3,72998	2,49193

Tabla que recopila la información de los tiempos de cada algoritmo de n^2. Como vemos en comparación, la que mejores tiempos tiene es selección, y el que peor, la burbuja.

Eficiencia O(n·log(n))			
tamaño	quicksort	mergesort	heapsort
10000	0,003625	0,00665079	0,004824
20000	0,007771	0,014437	0,008362
30000	0,011211	0,018691	0,013449
40000	0,016733	0,029801	0,018422
50000	0,01928	0,04314	0,023023
60000	0,023538	0,039113	0,029465
70000	0,027832	0,049835	0,03519
80000	0,032545	0,061842	0,039763
90000	0,036439	0,074306	0,046071
100000	0,040993	0,08895	0,050819
110000	0,044036	0,070985	0,056759
120000	0,048172	0,081647	0,063013
130000	0,052775	0,09109	0,068089
140000	0,057742	0,102959	0,076176
150000	0,061734	0,114649	0,079362
160000	0,066984	0,126993	0,088945

170000	0,070871	0,139966	0,090666
180000	0,074599	0,153146	0,099272
190000	0,077911	0,167807	0,107028
200000	0,084415	0,181705	0,109911
210000	0,088137	0,138456	0,131347
220000	0,092379	0,148041	0,12323
230000	0,097764	0,158239	0,129261
240000	0,100855	0,168314	0,147278
250000	0,104177	0,179208	0,144524

Tabla que recopila la información de los tiempos del resto de algoritmos de ordenación, en este caso con eficiencia de n*log(n). Observando la tabla, se ve perfectamente que a pesar de tener tamaños mayores a los 3 primeros, los tiempos son muy reducidos en comparación.

Eficiencia $O(n^3)$	
Tamaño	floyd
10	4,50E-05
20	0,000311
30	0,001085
40	0,002529
50	0,004823
60	0,00831
70	0,013163
80	0,019669
90	0,02811
100	0,03936
110	0,051813
120	0,066292
130	0,083675
140	0,105342
150	0,129086
160	0,156446
170	0,187267
180	0,224309
190	0,260606
200	0,303713
210	0,350943
220	0,403632
230	0,461212
240	0,52302
250	0,590928

En este caso, el algoritmo de Floyd es el único de eficiencia n^3, por lo que no se puede comparar el tiempo con más algoritmos similares.

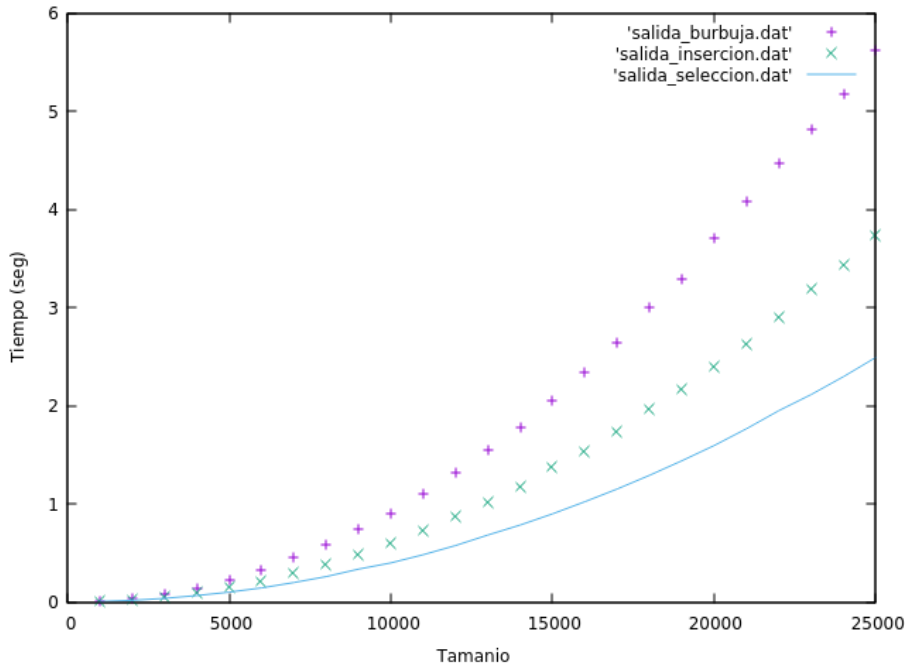
Eficiencia O(2^n)	
Tamaño	hanoi

1	7,00E-06
2	4,00E-06
3	4,00E-06
4	6,00E-06
5	6,00E-06
6	7,00E-06
7	8,00E-06
8	1,00E-05
9	1,70E-05
10	2,70E-05
11	5,80E-05
12	0,000103
13	0,000203
14	0,000382
15	0,000775
16	0,001563
17	0,003168
18	0,006212
19	0,012572
20	0,024988
21	0,049553
22	0,100354
23	0,198337
24	0,397377
25	0,792694

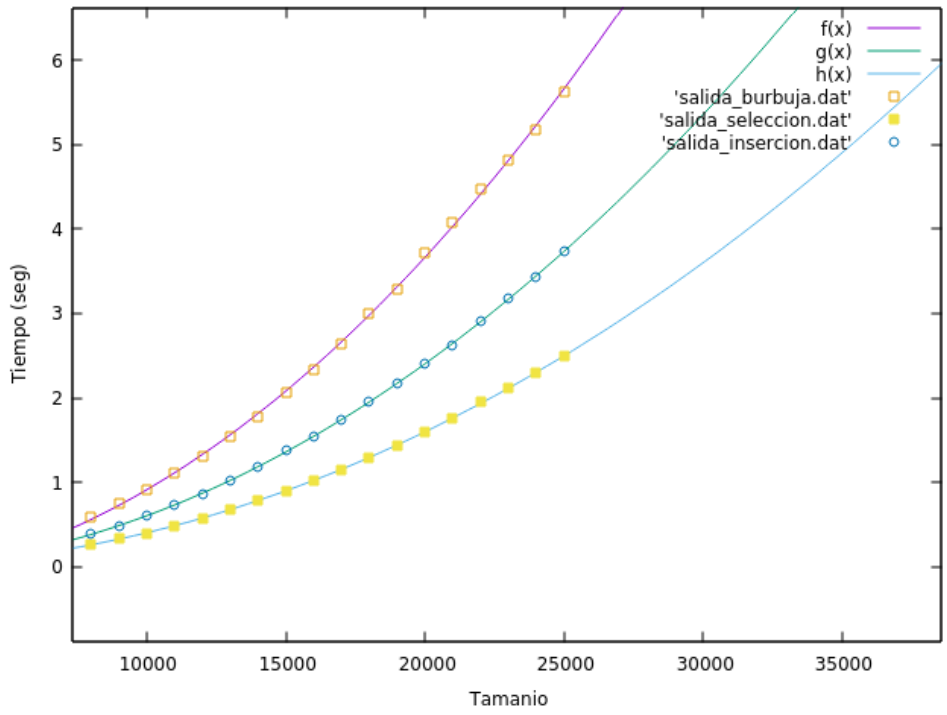
Por otra parte, el algoritmo de Hanoi es el único de eficiencia 2^n, pero se observa que para tamaños minúsculos en comparación con los anteriores, su tiempo es similar a estos.

2. Con cada una de las tablas anteriores, genere un gráfico comparando los tiempos de los algoritmos. Indique claramente el significado de cada serie. Para los algoritmos que realizan la misma tarea (los de ordenación), incluya también una gráfica con todos ellos, para poder apreciar las diferencias en rendimiento de algoritmos con diferente orden de eficiencia.

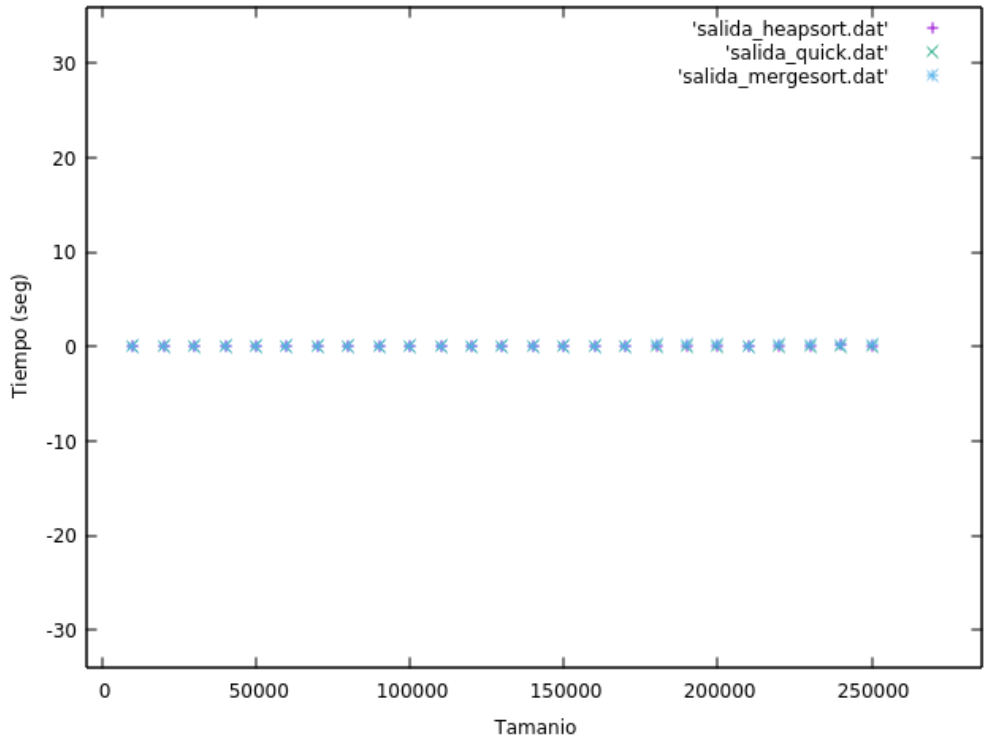
COMPARACIÓN DE ALGORITMOS CON EFICIENCIA N^2: Resultados de los algoritmos con eficiencia n^2 para comparar sus tiempos. Se observa lo mismo que en la tabla, pero de forma más visual. La burbuja, para los mismos tamaños, tarda más tiempo que el resto.



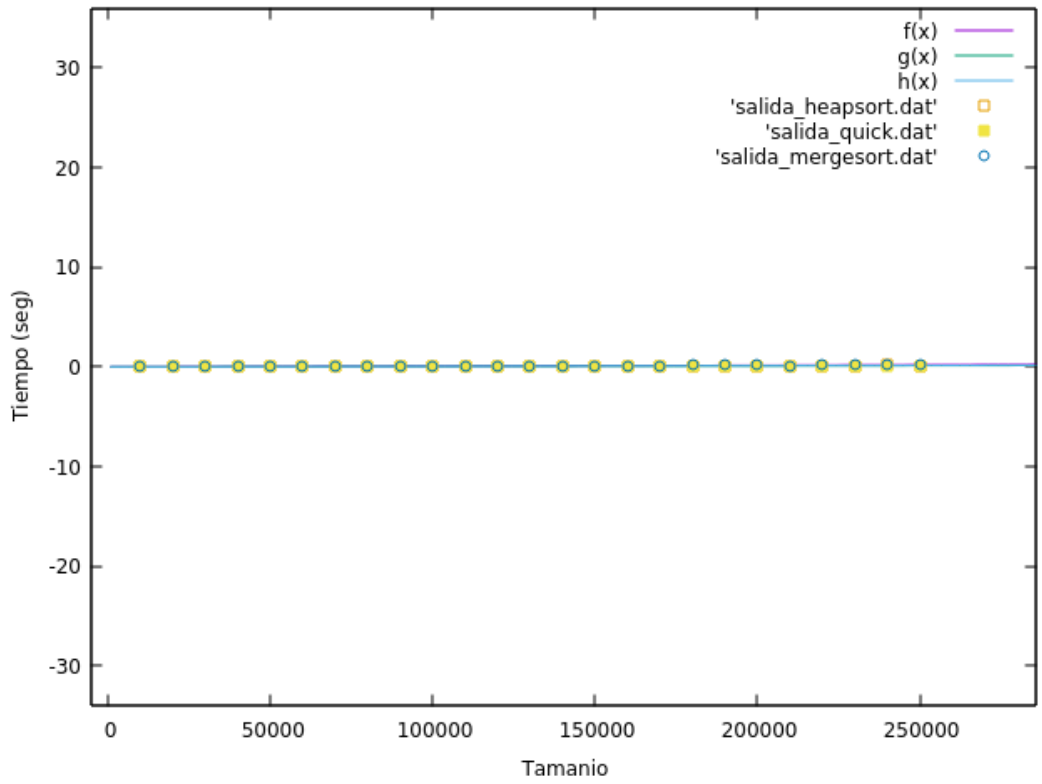
COMPARACIÓN DE ALGORITMOS CON EFICIENCIA N^2 CON SUS RESPECTIVAS TEÓRICAS: Esta gráfica viene a expresar exactamente lo mismo que la anterior, pero con sus ajustes. Vemos cómo teniendo ecuaciones teóricas idénticas, el ajuste de las constantes ocultas las aproxima bastante al resultado de cada una.



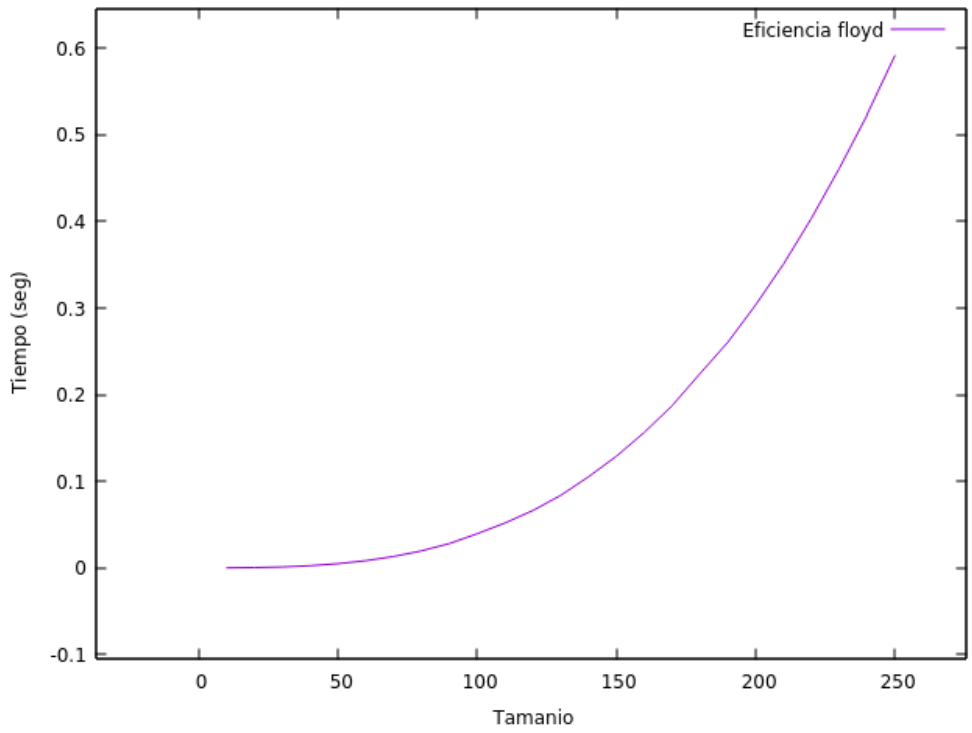
COMPARACIÓN DE ALGORITMOS CON EFICIENCIA N*LOG(N): Gráfica conjunta de todos los algoritmos que tienen eficiencia n*log(n). Como se observa, la eficiencia es similar en todos y el crecimiento es minúsculo. Por ello, es la mejor eficiencia.



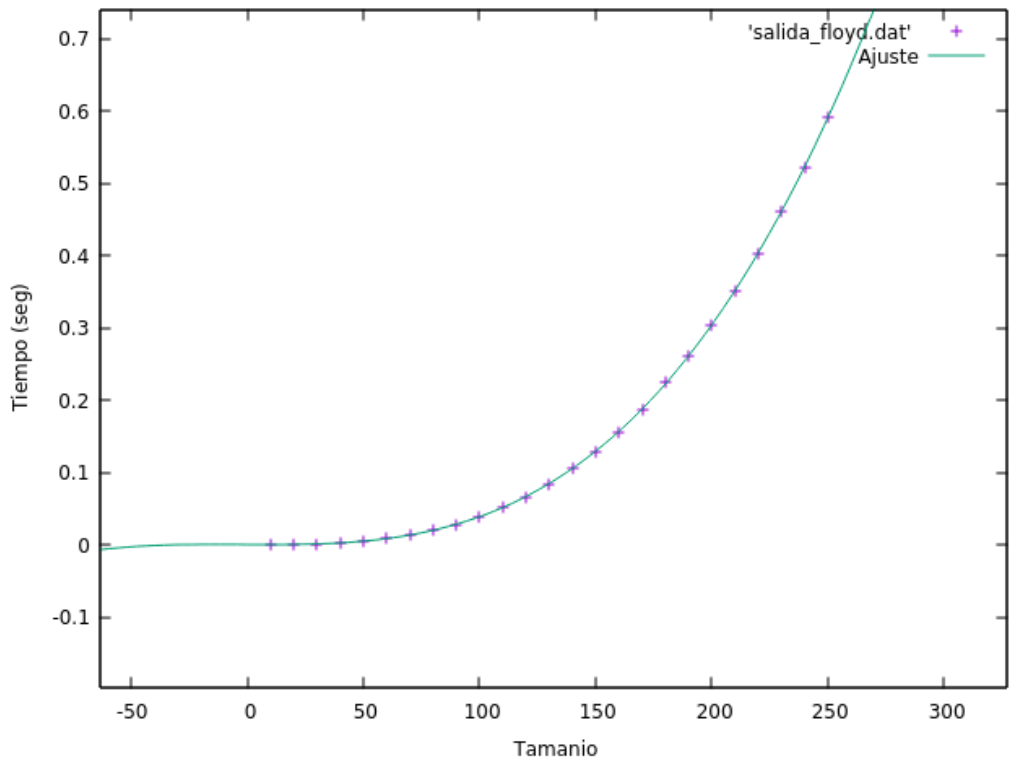
COMPARACIÓN DE ALGORITMOS CON EFICIENCIA N*LOG(N) CON SUS RESPECTIVAS TEÓRICAS: Comparación similar a la anterior, pero con el añadido de tener las teóricas ajustadas, que casi también coinciden entre ellas.



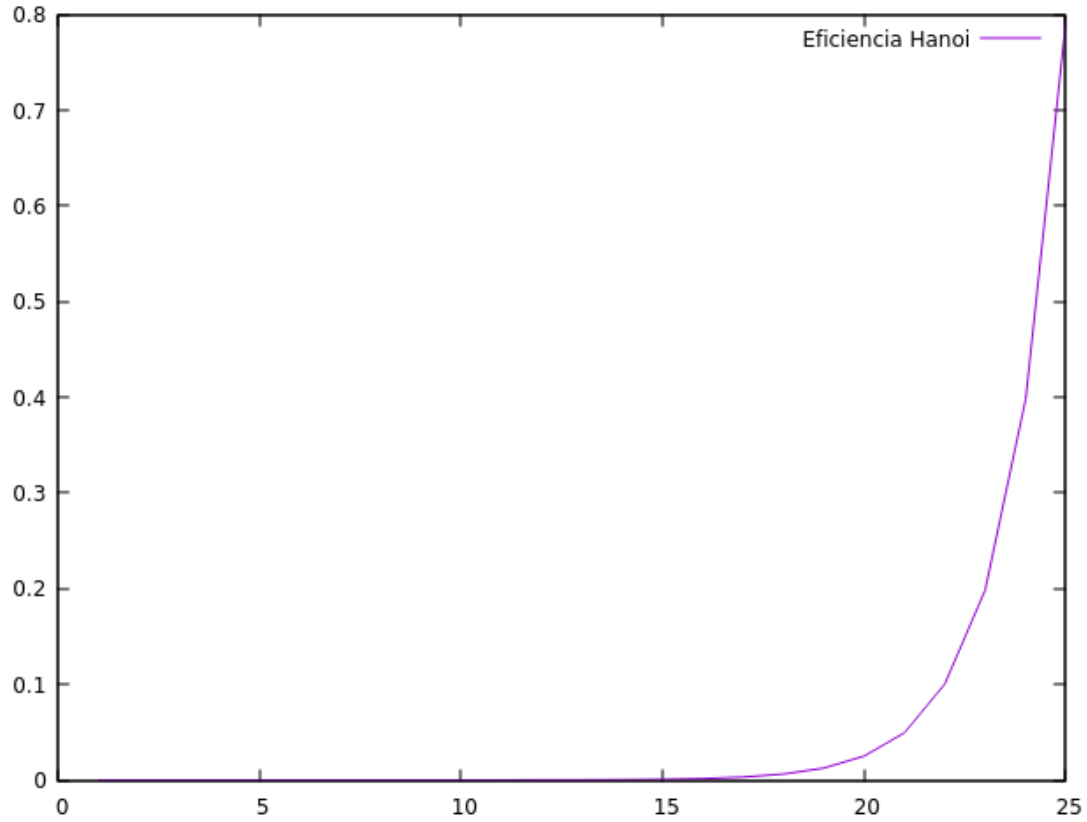
COMPARACIÓN DE ALGORITMOS CON EFICIENCIA N^3: Al solo haber un algoritmo de eficiencia n^3, la representación de esa tabla es similar a la representación ya vista de los tiempos por individual.



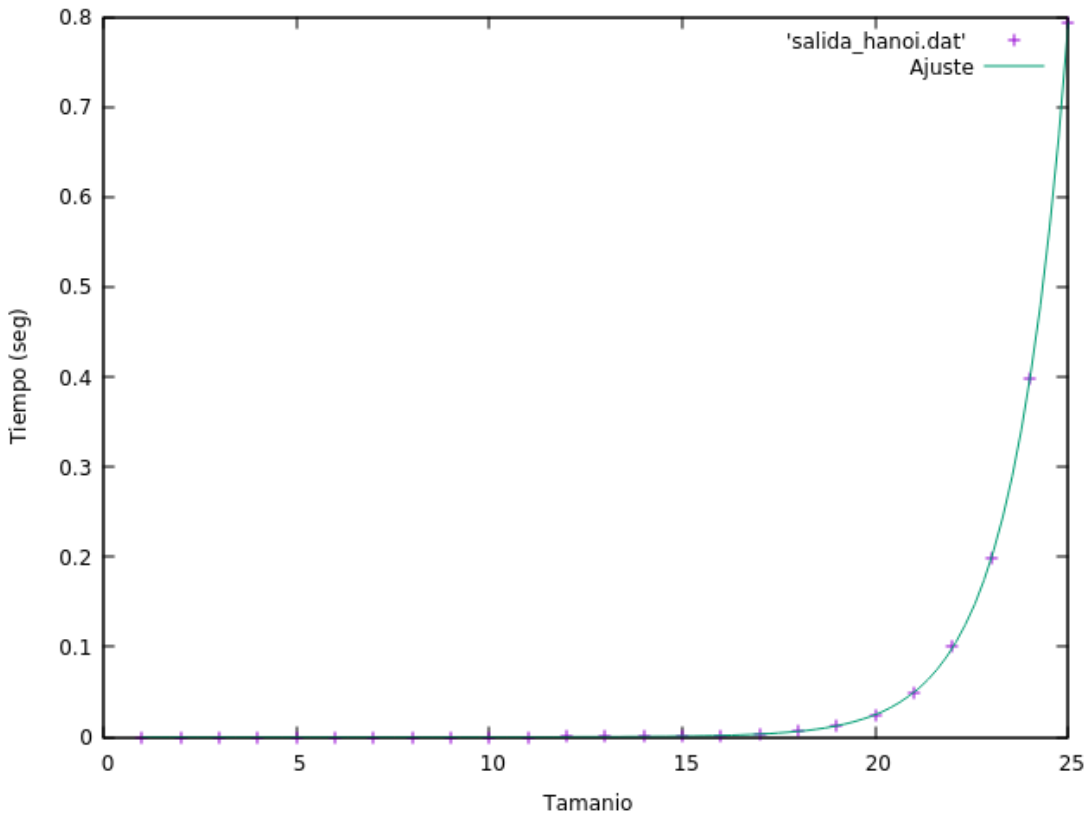
COMPARACIÓN DE ALGORITMOS CON EFICIENCIA N^3 CON SUS RESPECTIVAS TEÓRICAS:



COMPARACIÓN DE ALGORITMOS CON EFICIENCIA 2^N: Mismo caso que con el algoritmo de Floyd, la misma representación que la representación individual ya vista.

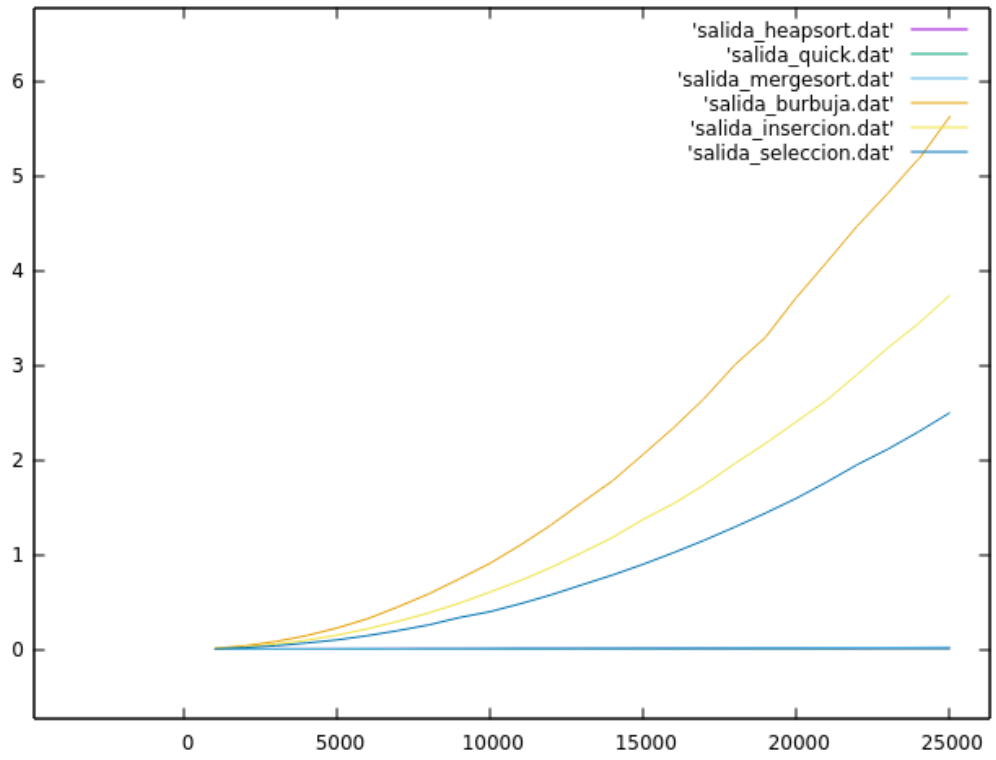


COMPARACIÓN DE ALGORITMOS CON EFICIENCIA 2^N CON SUS RESPECTIVA TEÓRICA:

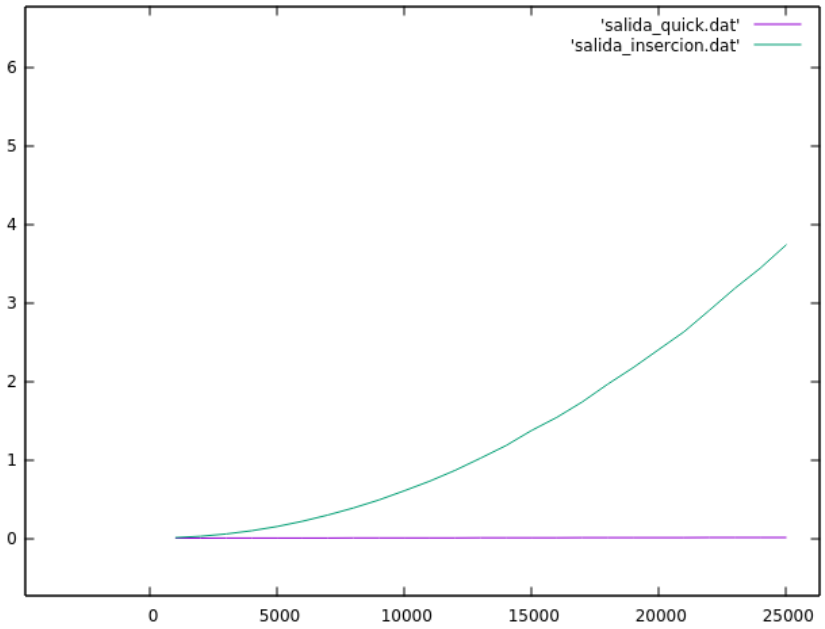


COMPARACIÓN DE TODOS ALGORITMOS DE ORDENACION: En esta gráfica vemos la comparación de todos los algoritmos de ordenación, tanto los que tienen eficiencia n^2 (selección, inserción y burbuja) como los que tienen eficiencia n*log(n) (mergesort, quicksort,

heapsort). Tras ponerlas todas juntas, se ve más claro que el mejor tiempo de las de n^2 es ya bastante más lento que el peor de los tiempo de n*log(n)



COMPARACIÓN DE LOS ALGORITMOS DE ORDENACION SELECCIONADOS POR MI: En esta imagen concreta simplemente se ve la comparación de los dos algoritmos seleccionados por mi (inserción y quicksort)

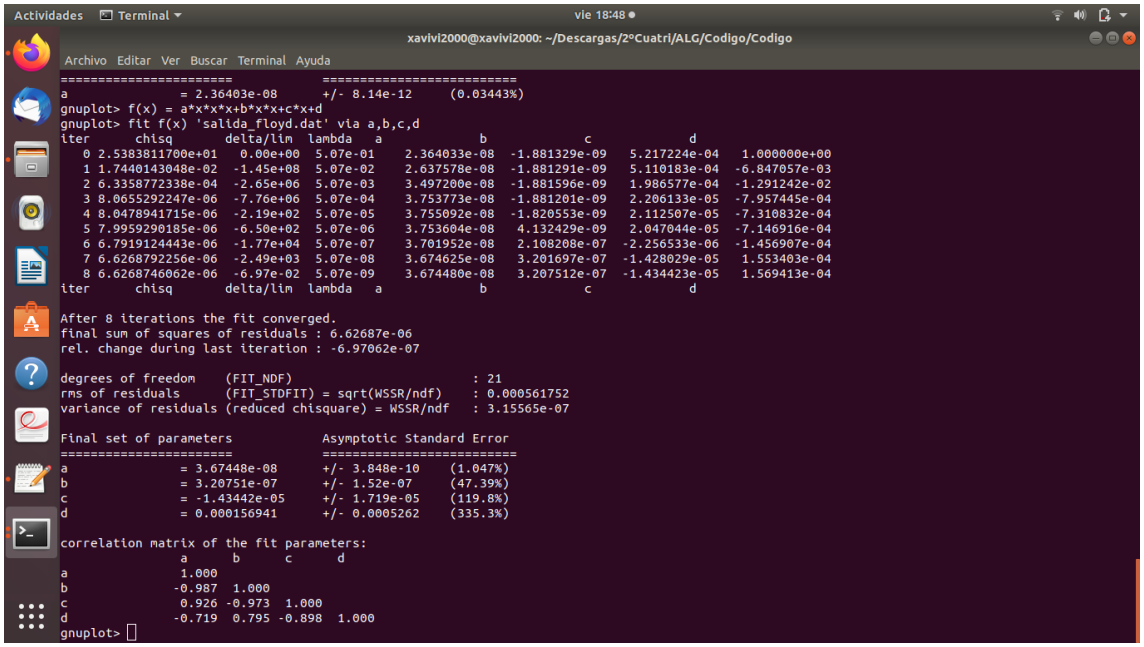


3. Calcule también la eficiencia hıbrida de todos los algoritmos, siguiendo las pautas indicadas en la sección 4. Pruebe también con otros ajustes que no se correspondan con la eficiencia teórica (ajuste lineal, cuadrático, etc.) y compruebe la variación en la calidad del ajuste.

El cálculo teórico nos da una expresión general, pero asociada a cada término de esta expresión aparece una constante de valor desconocido. Por ello, se ajusta la función a un conjunto de puntos usando regresión por mínimos cuadrados. Conociendo las constantes de la función, solo necesitas meter un valor a X para obtener el tiempo en el momento que quieras. En las imágenes posteriores se verá que con el uso de gnuplot se han obtenido los valores constantes ocultos.

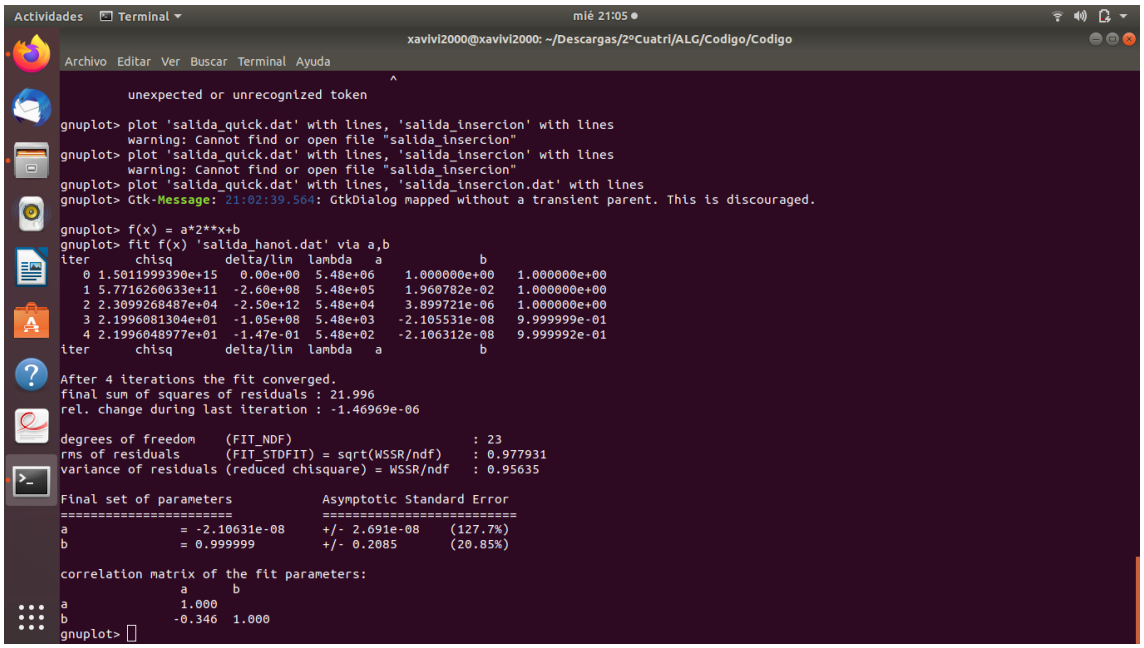
PARAMETROS DEL ALGORITMO AJUSTADO DE FLOYD: Con una teórica de n^3, se ajusta la función f(x) = a*x^3+b*x^2+c*x+d, obteniendo para ella, los siguientes valores:

a -> 3.67e-8 b-> 3.2075e-07 c-> -1.4344e5 d->0.0001569



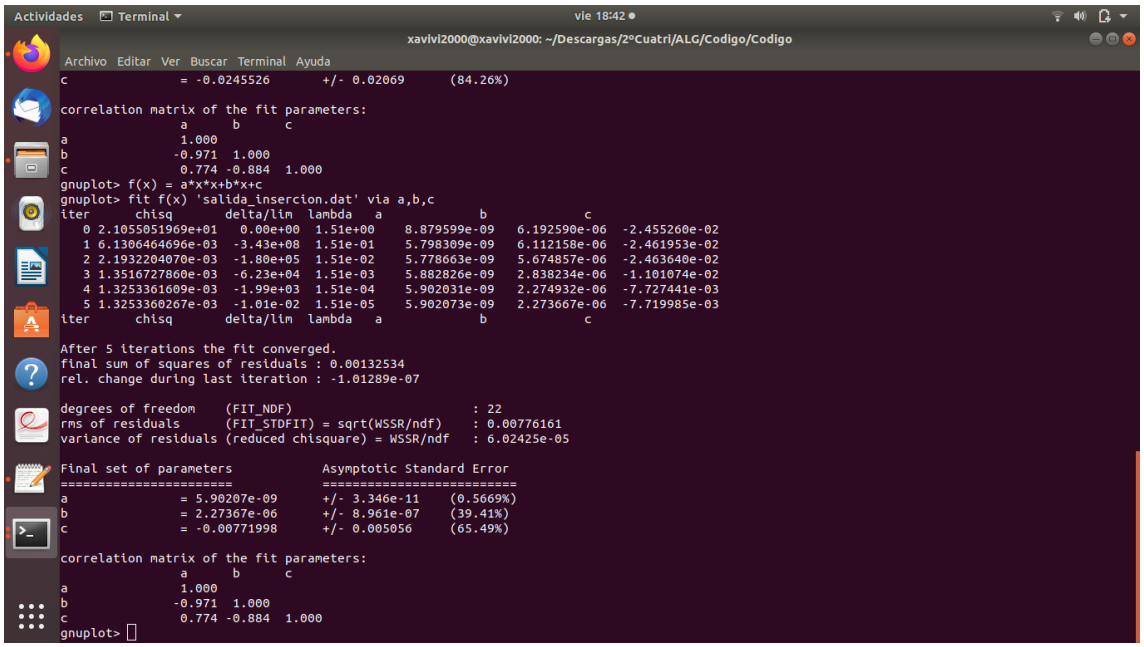
PARAMETROS DEL ALGORITMO AJUSTADO DE HANOI: Con una teórica de 2^n, se ajusta la función f(x) = a*2^x+b , obteniendo para ella, los siguientes valores:

a -> -2.10631e-8 b-> 0.9999999



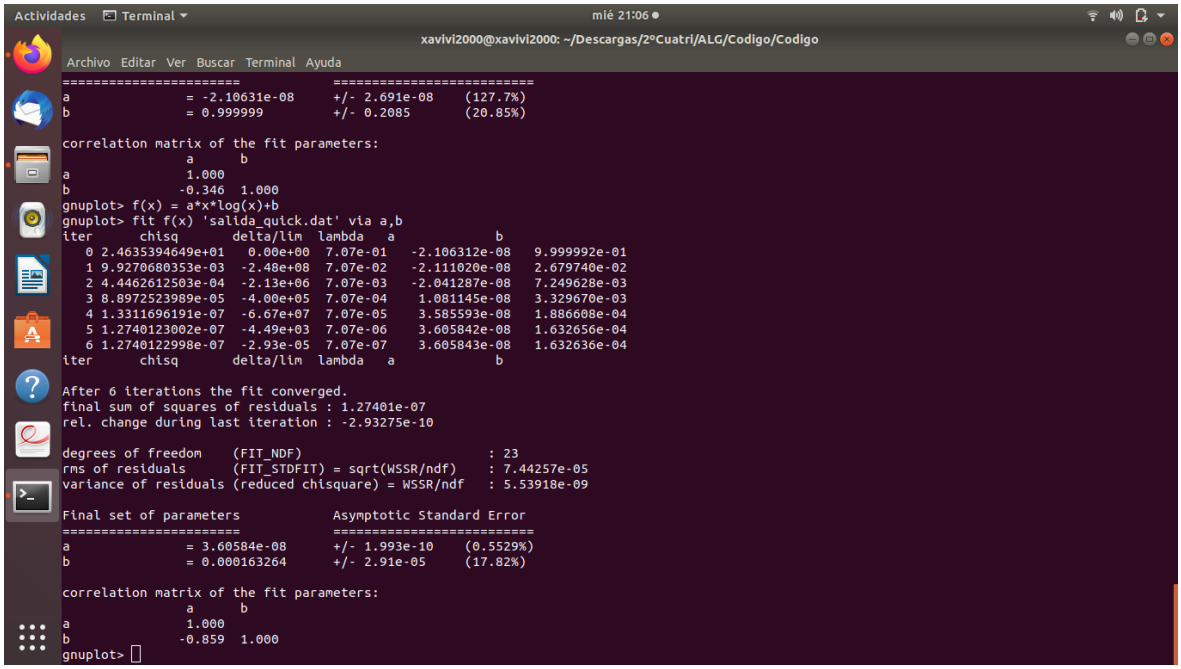
PARAMETROS DEL ALGORITMO AJUSTADO DE INSERCION: Con una teórica de n^2, se ajusta la función f(x) = a*x^2+b*x+c, obteniendo para ella, los siguientes valores:

a -> 5.90207e-09 b-> 2.27367e-6 c-> -0.00771998



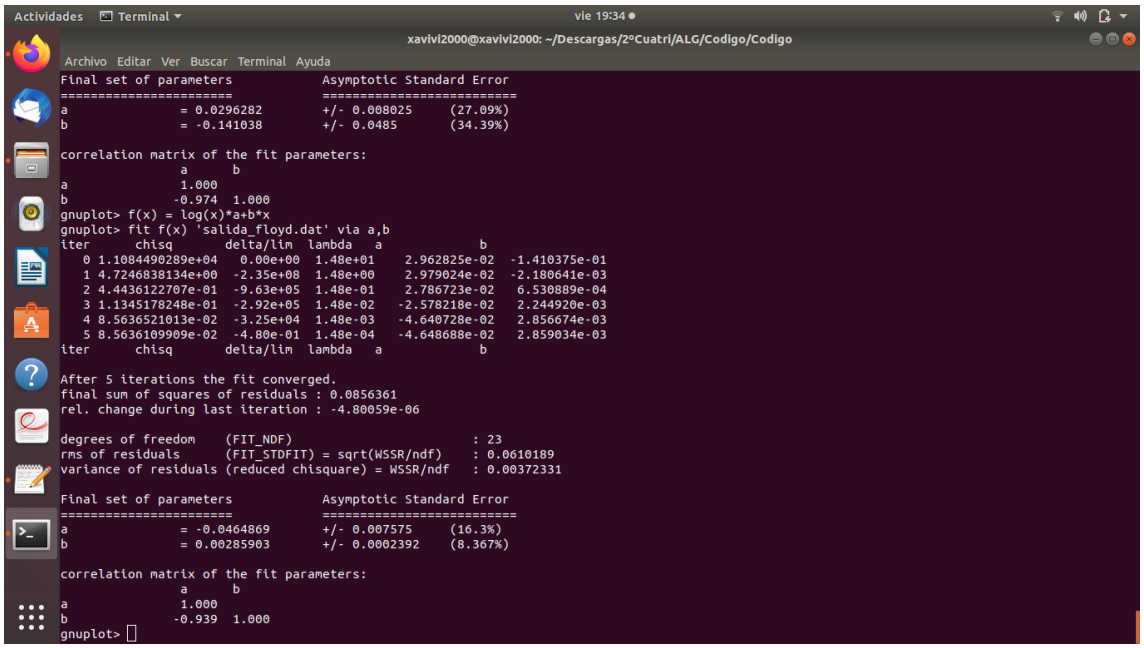
PARAMETROS DEL ALGORITMO AJUSTADO DE QUICKSORT: Con una teórica de n*log(n), se ajusta la función f(x) = a*x*log(x) + b, obteniendo para ella, los siguientes valores:

a -> 3.60584e-8 b-> 0.000163264

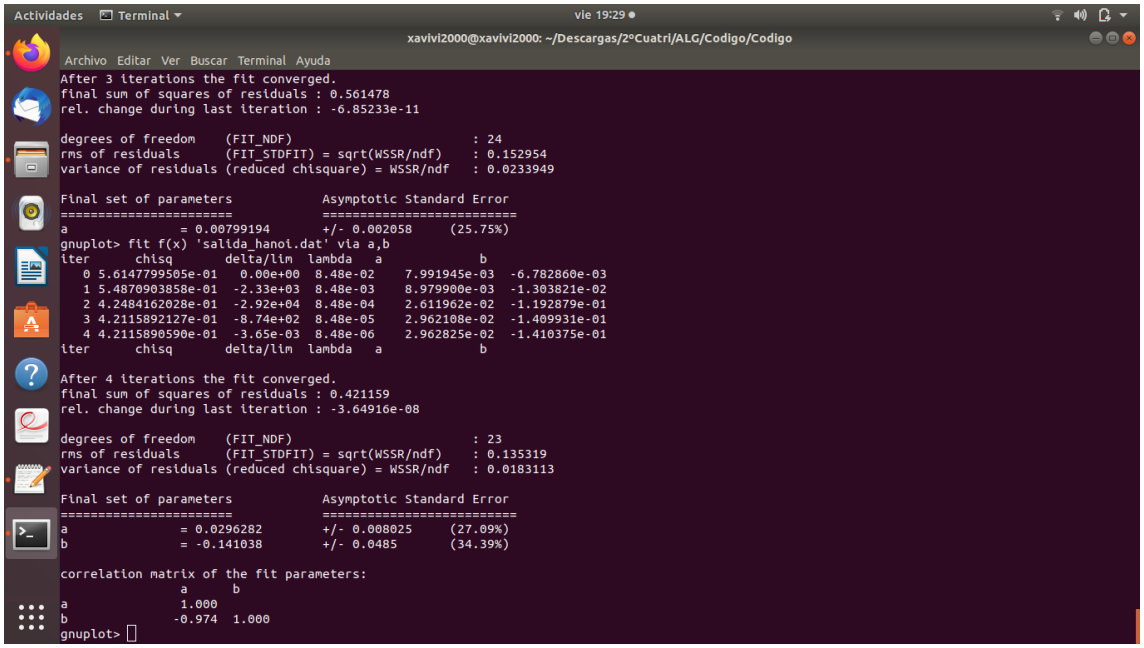


Ahora vienen 4 imágenes, una por cada algoritmo anterior, pero con un ajuste incorrecto para ver en la matriz de correlación la comparación de su precisión.

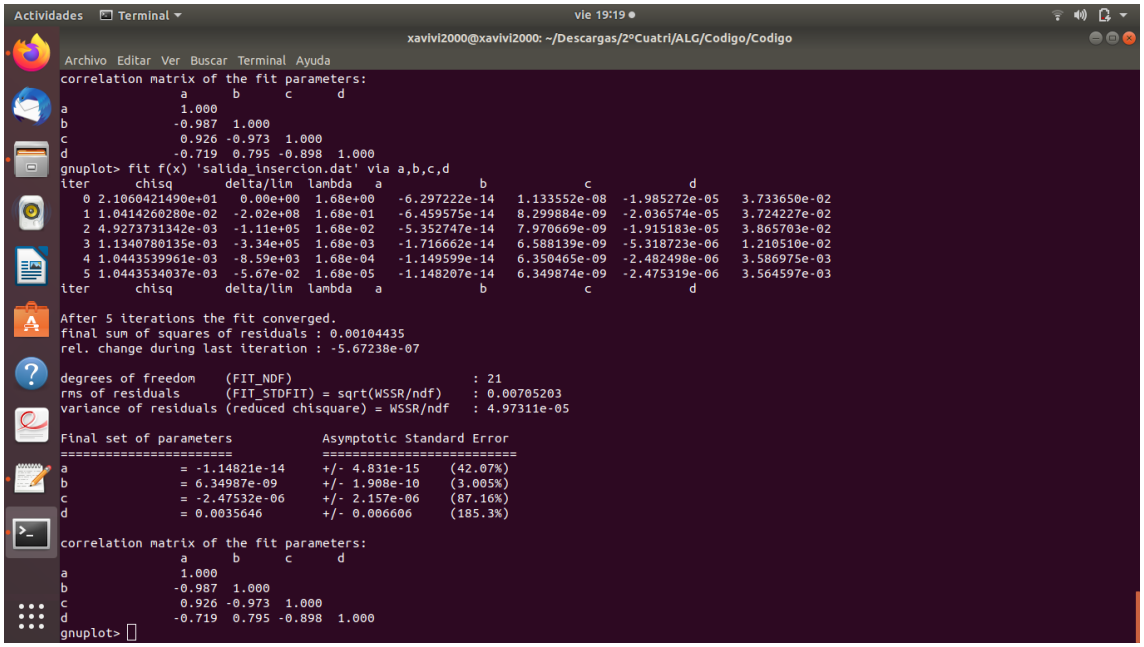
MAL AJUSTE DE FLOYD (CON 0(N)): En este caso ajusto la función f(x) = x * b + a * log(x) a un algoritmo de n^3



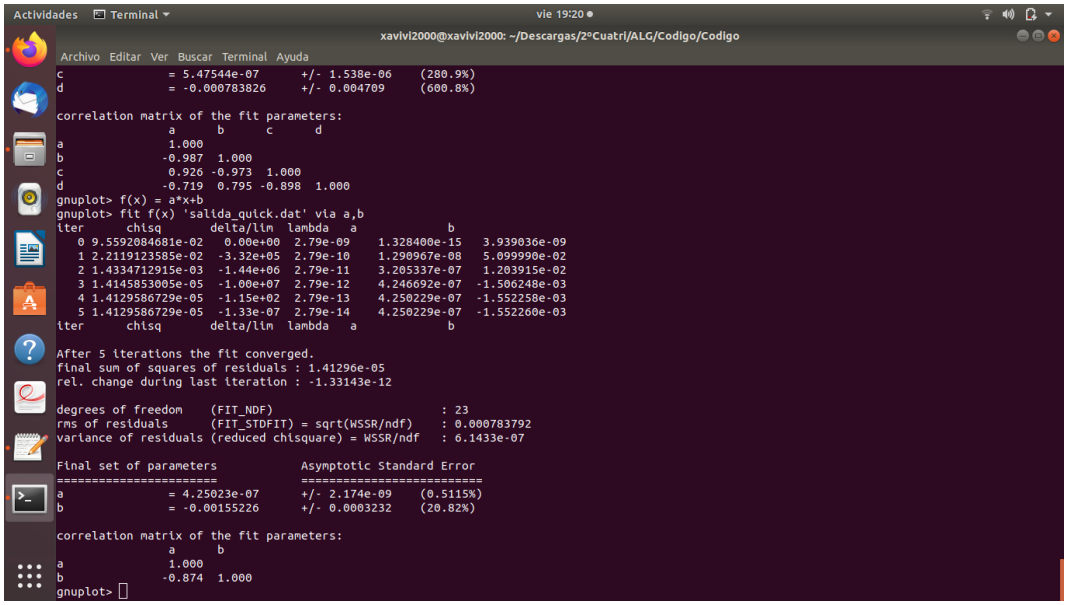
MAL AJUSTE DE HANOI (CON 0(N)): f(x) = x * b + a * log(x) pero con un algoritmo de 2^n



MAL AJUSTE DE INSERCION (CON 0(N^3)): Algoritmo de n^2 ajustado con uno de n^3



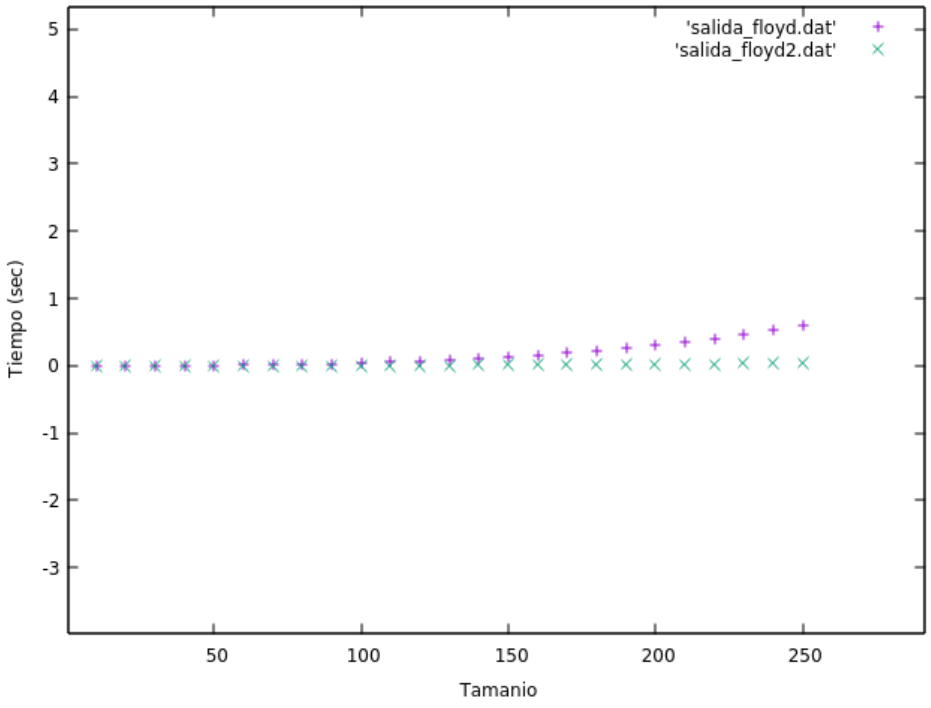
MAL AJUSTE DE QUICKSORT (CON 0(N)): Algoritmo de n^3 ajustado con una función igual a a*x+b



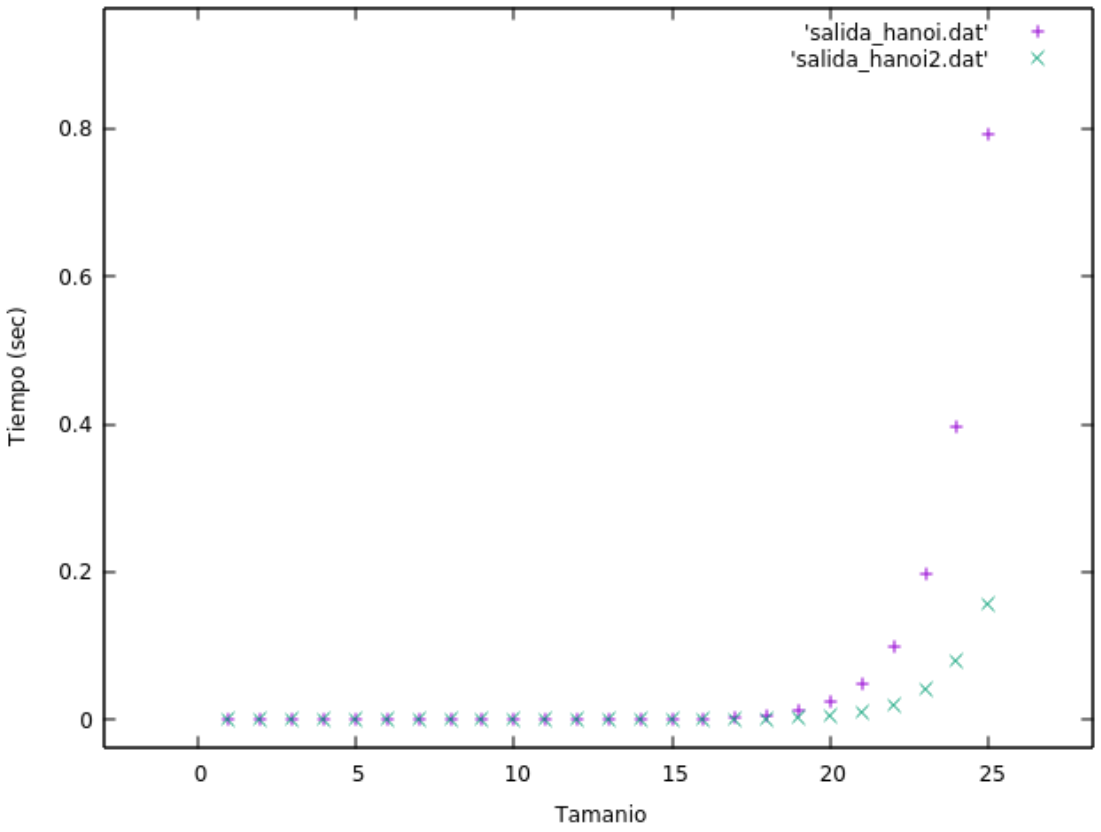
4. Otro aspecto interesante a analizar mediante este tipo de estudio es la variación de la eficiencia empírica en función de parámetros externos tales como: las opciones de compilación utilizada (con/sin optimización), el ordenador donde se realizan las pruebas, el sistema operativo, etc. Sugiera algún estudio de este tipo, consulte con el profesor de prácticas y llévelo a cabo.

Compilar con optimización es una forma de minimizar ciertos atributos de un programa para aumentar eficiencia y rendimiento. De esta forma, he comparado los tiempos de la ejecución normal de los programas que incluyen los algoritmos con los mismo compilados de una forma optimizada. Aquí está la diferencia.

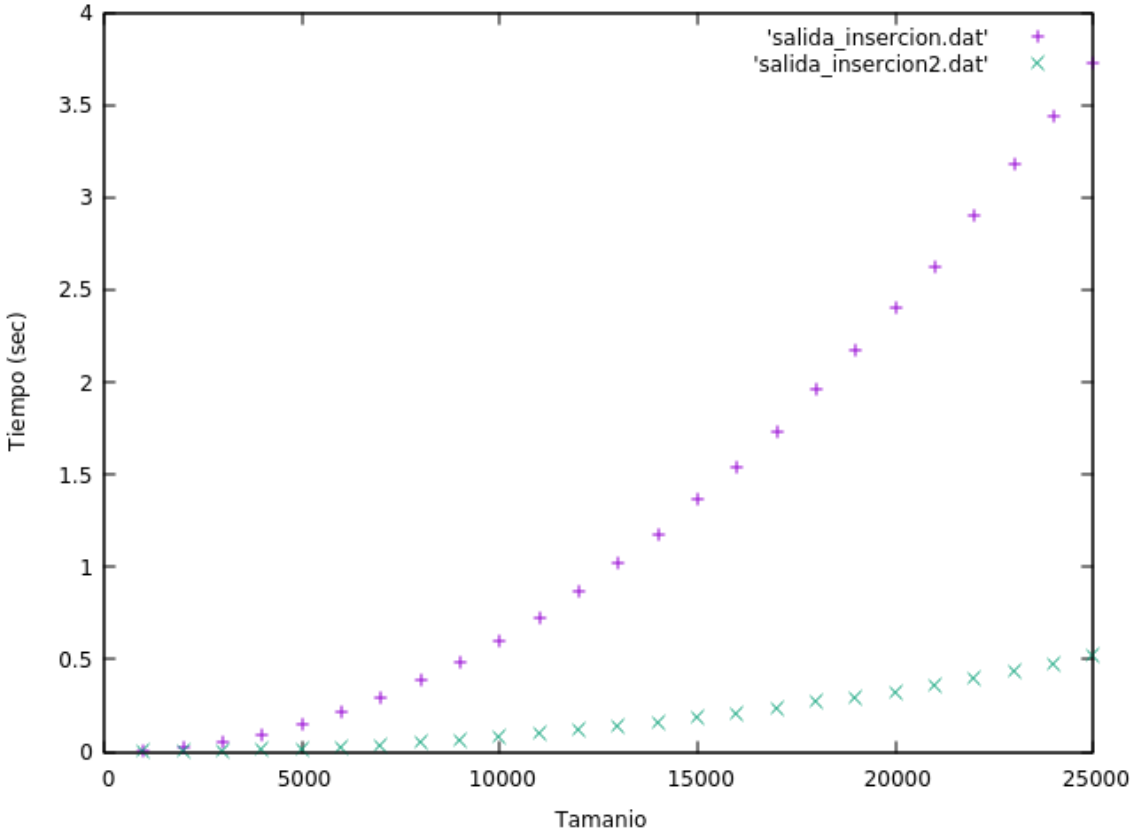
Floyd compilado con optimización (-01)



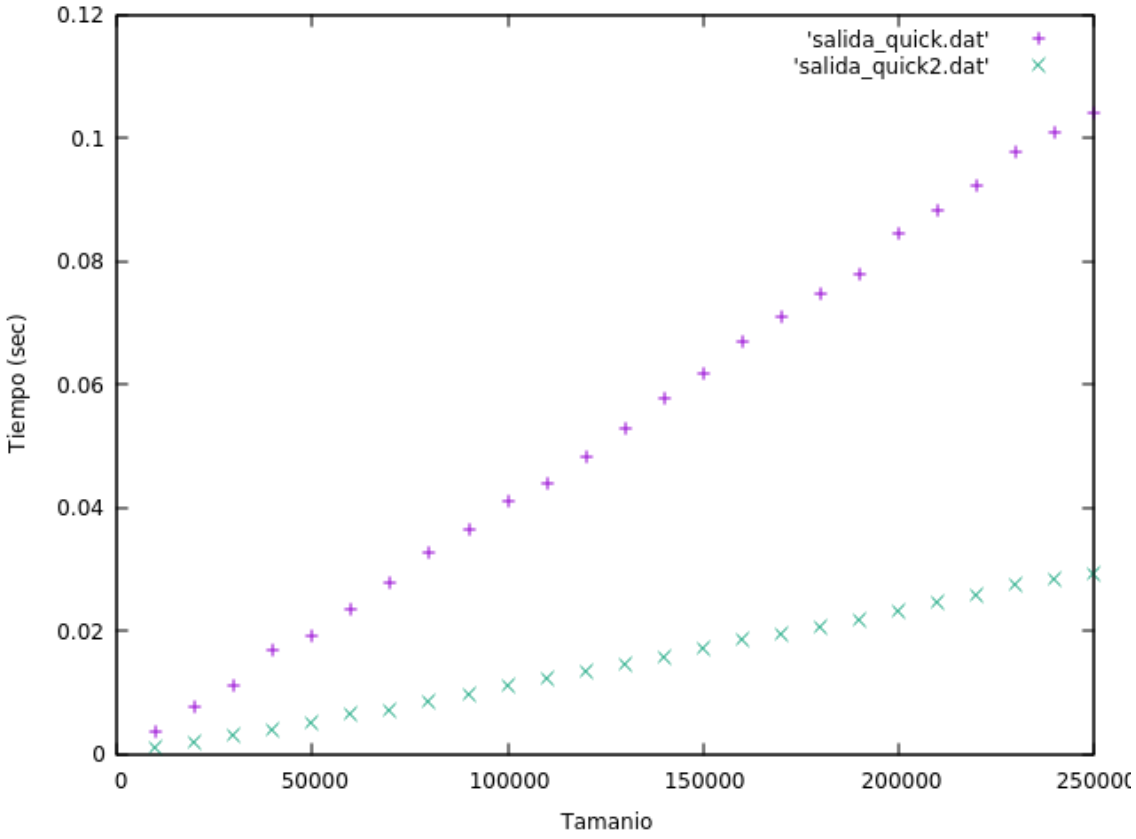
Hanoi compilado con optimización (-03)



Inserción compilado con optimización (-01)



Quicksort compilado con optimización (-02)



Como conclusión, vemos que elijas la optimización que elijas, el resultado es mucho mas eficiente.

Javier Ramírez Pulido