Universidad de Granada

ESCUELA TECNICA SUPERIOR INGENIEŔIA INFORMÁTICA Y TELECOMUNICACIONES

ANÁLISIS DE EFICIENCIA DE ALGORITMOS

Práctica 1

Daniel Monjas Miguélez

Indice

1. Cálculo de la eficiencia empírica.

En la primera parte de la práctica nos centramos en el cálculo de la eficiencia empírica de los algoritmos elegidos. Como se indicó en clase he utilizado el algoritmo de Inserción como representante de orden $O(n^2)$, el algoritmo Quicksort como representante de orden $O(n*log(n))$ y el algoritmo de Floyd con orden $O(n^3)$. Además para los algoritmos utilizados he realizado 10 ejecuciones para cada uno de los tamaños usados, de forma que el dato que aparecerá en las gráficas no será otro que la media de los diez tiempos de ejecución para cada tamaño.

1.1 Alogrimto de Inserción

Los tiempos de ejecución obtenidos para el algoritmo de inserción sin ningún tipo de optimización son (sólo se incluirán los valores de los tamaños, la media y cinco de las ejecuciones por cuestión de tamaño):

Tamaño	Media	Tiempos de Ejecución				
4000	0.013685	0.015815	0.013838	0.013877	0.013443	0.013081
8000	0.0524105	0.05373	0.053368	0.051914	0.052557	0.05229
12000	0.1178577	0.118764	0.118507	0.117751	0.118963	0.115943
16000	0.2093507	0.211666	0.207448	0.209642	0.210768	0.208424
20000	0.3272508	0.330689	0.326415	0.325613	0.327564	0.327145
24000	0.4726259	0.47489	0.47257	0.471294	0.475906	0.472859
28000	0.6425901	0.649738	0.641361	0.640661	0.645246	0.643897
32000	0.8407799	0.841205	0.837072	0.841635	0.837355	0.838963
36000	1.06255	1.06509	1.05846	1.06711	1.06503	1.06461
40000	1.309767	1.31332	1.30718	1.30386	1.3083	1.31149
44000	1.582734	1.58449	1.58847	1.58432	1.5789	1.58302
48000	1.89902	1.91424	1.89734	1.89071	1.9044	1.89834
52000	2.221734	2.21277	2.2298	2.23124	2.22487	2.21958
56000	2.57781	2.59069	2.57862	2.58105	2.56523	2.58213
60000	2.957033	2.96842	2.96099	2.95463	2.95565	2.95234
64000	3.37416	3.37833	3.37839	3.36595	3.3745	3.38828
68000	3.795755	3.84553	3.78194	3.79663	3.79244	3.7859
72000	4.339826	4.26068	4.27159	4.30665	4.26938	4.27734
76000	4.747083	4.74331	4.7399	4.72654	4.76194	4.77072
80000	5.261537	5.25542	5.27023	5.24186	5.26192	5.26386
84000	5.785753	5.79045	5.7972	5.77781	5.78846	5.80487
88000	6.413192	6.40834	6.38501	6.40125	6.37443	6.39647
92000	7.042922	7.05084	6.97721	6.99745	7.1643	7.31583
96000	7.657275	8.43018	7.67828	7.58662	7.56452	7.57438
100000	8.246308	8.26389	8.25684	8.20495	8.20974	8.23968

Figura 1: Tabla de Tiempos Inserción sin Optimizar

En la tabla se aprecian los tamaños usados, que van desde 4000 hasta 100000, con saltos de 4000 en 40000. También se aprecia los distintos tiempos obtenidos y la media de los mismo. Además de esta tabla se ha obtenido la siguiente gráfica:

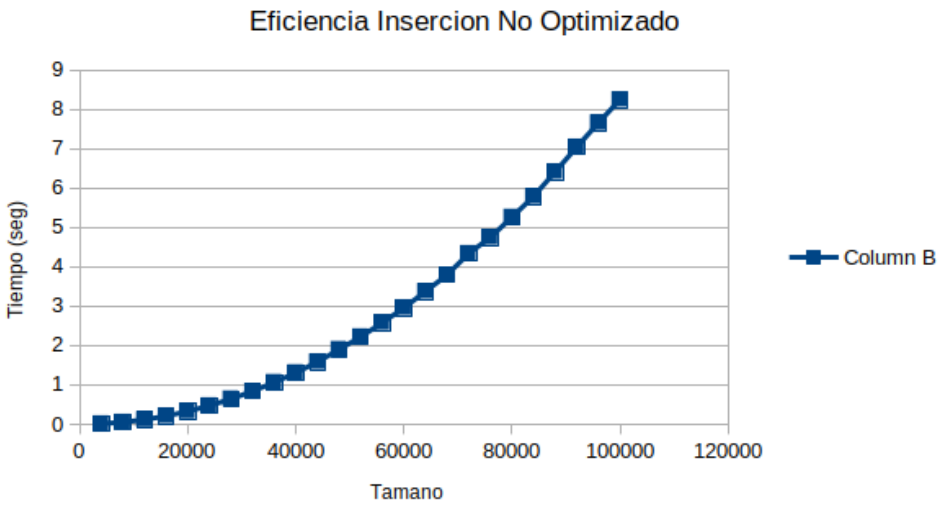


Figura 2: Grafica Insercion sin Modificar

En esta gráfica se puede apreciar que lo más probable es que la función de la gráfica se corresponda con una polinomial de grado 2, lo cual coincidiría con lo teórico, pues el algoritmo de inserción es de orden $O(n^2)$. Hemos comprobado lo anterior ajustando la gráfica dada por los datos de la tabla a una función tipo $f(x)=a*x^2+b*x+c$, dando lugar a la siguiente gráfica:

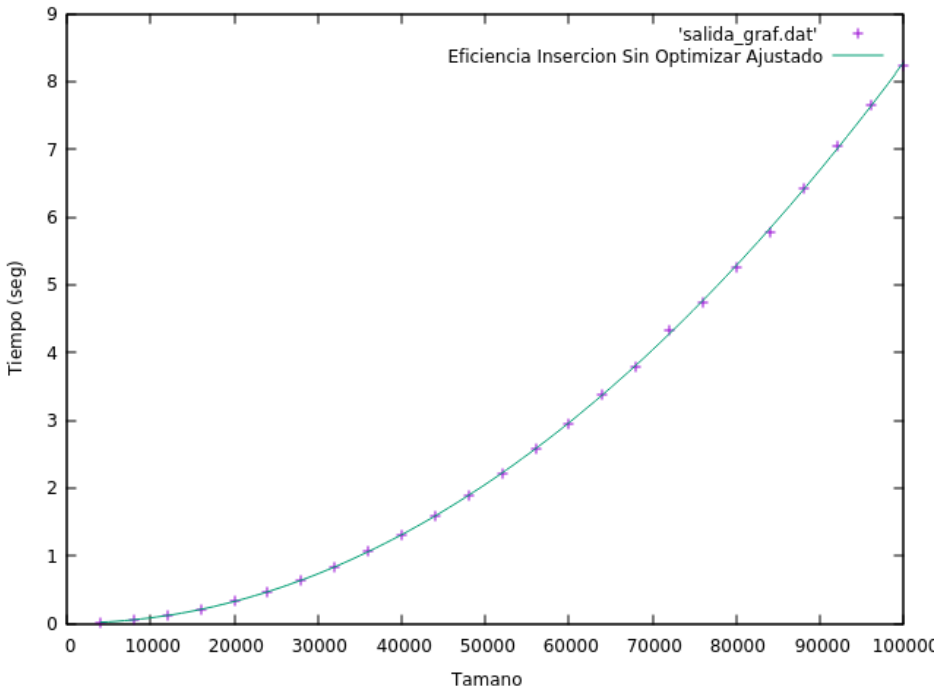


Figura 3: Grafica Inserción Ajustada

Vemos que la función f(x) se ajusta perfectamente a la gráfica, es más hemos calculado el coeficiente de determinación que es R^2=0.99993022198, con lo que el ajuste es muy bueno. También vemos cuales son los valores a, b y c que han asignado a f(x) para que se ajuste tan bien a la tabla.

```
gnuplot> set xlabel "Tamano"
gnuplot> set ylabel "Tiempo (seg)"
gnuplot> f(x)=a*x*x+b*x+c
gnuplot> fit f(x) 'salida_graf.dat' via a,b,c
iter      chisq       delta/lim  lambda   a             b             c
   0 5.5134663935e+20   0.00e+00  2.71e+09    1.000000e+00   1.000000e+00   1.000000e+00
   1 9.5454756094e+16  -5.77e+08  2.71e+08    1.314580e-02   9.999879e-01   1.000000e+00
   2 7.2111803321e+09  -1.32e+12  2.71e+07   -1.050620e-05   9.999877e-01   1.000000e+00
   3 5.5145766910e+09  -3.08e+04  2.71e+06   -1.226028e-05   9.999802e-01   1.000000e+00
   4 5.5063125865e+09  -1.50e+02  2.71e+05   -1.225109e-05   9.992306e-01   1.000000e+00
   5 4.7646681199e+09  -1.56e+04  2.71e+04   -1.139613e-05   9.295020e-01   9.999972e-01
   6 6.5925312135e+07  -7.13e+06  2.71e+03   -1.339484e-06   1.093002e-01   9.999651e-01
   7 1.1917307567e+02  -5.53e+10  2.71e+02   -6.316579e-10   1.058348e-04   9.999605e-01
   8 2.3277258897e+00  -5.02e+06  2.71e+01    1.153112e-09  -3.972737e-05   9.999287e-01
   9 2.3129461085e+00  -6.39e+02  2.71e+00    1.152122e-09  -3.960515e-05   9.967606e-01
  10 1.3311690974e+00  -7.38e+04  2.71e-01    1.075309e-09  -3.020329e-05   7.568594e-01
  11 1.2623779213e-02  -1.04e+07  2.71e-02    8.429462e-10  -1.762050e-06   3.114315e-02
  12 1.1416073152e-02  -1.06e+04  2.71e-03    8.357001e-10  -8.751231e-07   8.512027e-03
  13 1.1416073035e-02  -1.03e-03  2.71e-04    8.356978e-10  -8.748465e-07   8.504968e-03
iter      chisq       delta/lim  lambda   a             b             c

After 13 iterations the fit converged.
final sum of squares of residuals : 0.0114161
rel. change during last iteration : -1.02878e-08

degrees of freedom    (FIT_NDF)                        : 22
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.0227796
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.000518912

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 8.35698e-10      +/- 6.137e-12    (0.7344%)
b               = -8.74846e-07     +/- 6.575e-07    (75.16%)
c               = 0.00850497       +/- 0.01484      (174.5%)

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 'salida_graf.dat', f(x) title 'Eficiencia Insercion Sin Optimizar Ajustado'
```

Figura 4: Coeficientes de la Función de Ajuste

1.2 Algoritmo Quicksort

En la siguiente tabla se representan los tiempos de ejecución obtenidos para el algoritmo Quicksort sin ningún tipo de optimización. Al igual que en la caso anterior (y de ahora en adelante), se mostrarán solo 5 de los 10 tiempos tomados, la media, y el tamaño utilizado:

Tamaño	Media	Tiempo de Ejecución				
400000	0.0508621	0.051201	0.05123	0.050238	0.051067	0.051255
800000	0.1058222	0.107062	0.105696	0.105546	0.10531	0.105819
1200000	0.1630979	0.163904	0.162651	0.162765	0.162735	0.164464
1600000	0.2216881	0.222367	0.221551	0.22063	0.223117	0.222002
2000000	0.2806205	0.281896	0.282628	0.281075	0.280467	0.279893
2400000	0.3413742	0.344997	0.33886	0.339023	0.341332	0.340674
2800000	0.4020371	0.403773	0.402863	0.399725	0.401768	0.402133
3200000	0.4616384	0.462488	0.462309	0.463125	0.463924	0.460121
3600000	0.5233502	0.523504	0.523067	0.524208	0.522849	0.527506
4000000	0.5844746	0.582532	0.587133	0.583305	0.582032	0.584077
4400000	0.6477136	0.649141	0.645432	0.643978	0.646053	0.648229
4800000	0.7086967	0.71242	0.704391	0.708838	0.710199	0.708963
5200000	0.7710292	0.772775	0.775582	0.782437	0.768278	0.770085
5600000	0.8331474	0.834521	0.833506	0.831788	0.833757	0.835976
6000000	0.8954168	0.890643	0.908687	0.898604	0.895908	0.893727
6400000	0.9596896	0.96046	0.957486	0.966524	0.953888	0.954979
6800000	1.027132	1.03523	1.01762	1.02512	1.01999	1.02664
7200000	1.087501	1.08917	1.0836	1.08264	1.09091	1.08712
7600000	1.15403	1.15844	1.1579	1.15769	1.14484	1.15229
8000000	1.217409	1.21217	1.21521	1.23486	1.21188	1.21974
8400000	1.303428	1.2884	1.27163	1.28666	1.27627	1.28689
8800000	1.401386	1.37355	1.46359	1.37086	1.37182	1.39113
9200000	1.438487	1.45407	1.42963	1.45688	1.46618	1.43318
9600000	1.481707	1.46952	1.48289	1.48084	1.48689	1.48826
10000000	1.577475	1.56326	1.57141	1.61895	1.56007	1.62896

Figura 5: Tabla de Tiempos de Ejecución Quicksort

En esta tabla se muestran los tiempos obtenidos por el algoritmo Quicksort para tamaños desde 400000 hasta 10000000, realizando saltos de tamaño de 400000. Debido a que el orden del algoritmo Quicksort es O(n*log(n)) se aprecia que en un tiempo muy pequeño se ha realizado la ordenación de un vector con un tamaño mucho mayor que con el algoritmo de inserción. Para representar gráficamente lo anterior se ha utilizado la gráfica dando lugar al siguiente gráfico:

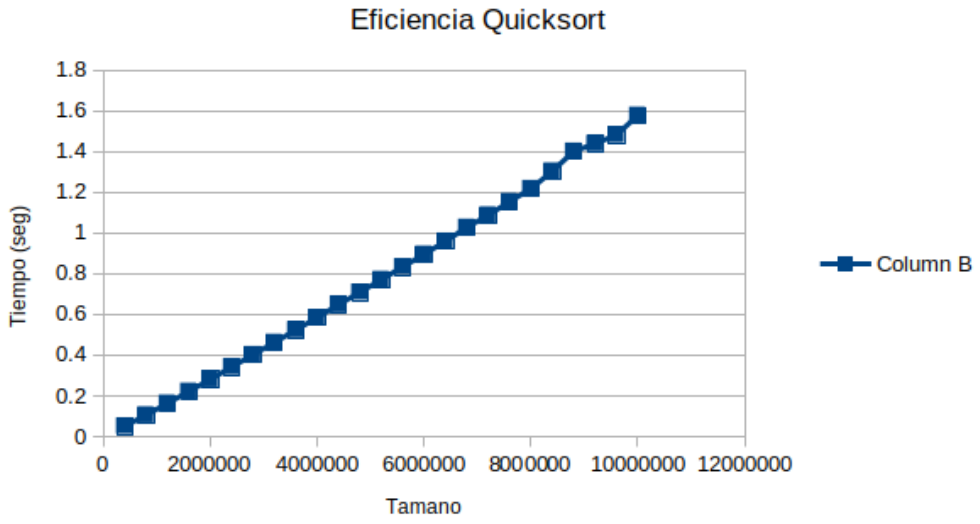


Figura 6: Grafica Quicksort

Si bien este algoritmo se debería asemejar a una función de tipo f(x)=a*x*log(x)+b, se parece más bien a una función lineal. Veremos como esto se verifica con las dos siguientes graficas:

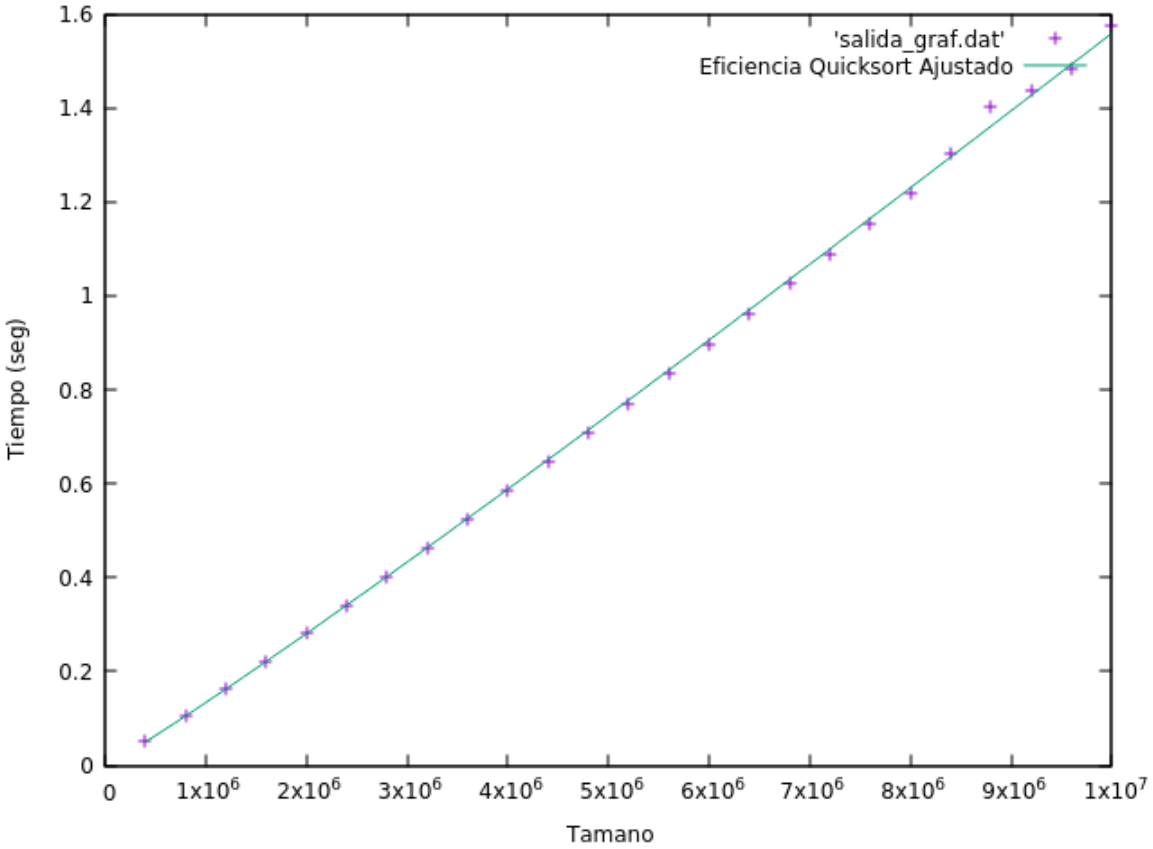


Figura 7: Quicksort Ajuste Logarítimico

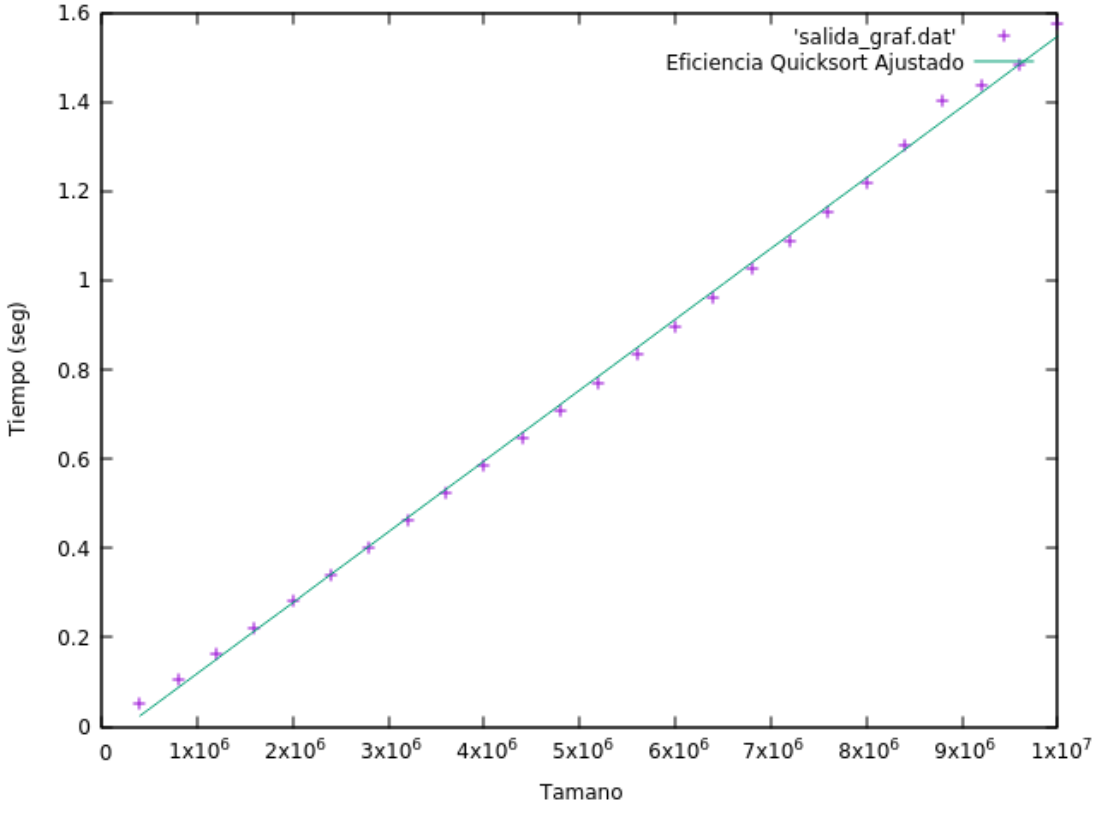


Figura 8: Quicksort con Ajuste Lineal

Se aprecia a primera vista que el segundo ajuste es mucho mejor que el primero. Esto además te lo verifica el coeficiente de determinación que es $R^2 = 0.998691$ en la segunda gráfica frente al $R^2 = 0.8372575$ de la primera gráfica. Luego si bien teóricamente el algoritmo Quicksort es O(n*log(n)) en este caso se asemeja más a una función lineal. En la siguiente imagen se muestra los valores a,b de la función con forma f(x)=a*x+b:

```
gnuplot> fit f(x) 'salida_graf.dat' via a,b
iter      chisq       delta/lim  lambda   a             b
   0 8.8399998972e+14   0.00e+00  4.20e+06    1.000000e+00   1.000000e+00
   1 3.3986927709e+11  -2.60e+08  4.20e+05    1.960785e-02   9.999999e-01
   2 1.3595702170e+04  -2.50e+12  4.20e+04    3.926600e-06   9.999999e-01
   3 6.3673863575e+00  -2.13e+08  4.20e+03    5.823085e-09   9.999998e-01
   4 6.3673820707e+00  -6.73e-02  4.20e+02    5.815295e-09   9.999995e-01
iter      chisq       delta/lim  lambda   a             b

After 4 iterations the fit converged.
final sum of squares of residuals : 6.36738
rel. change during last iteration : -6.73244e-07

degrees of freedom    (FIT_NDF)                        : 23
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.526158
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.276843

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 5.81529e-09      +/- 3.648e-08    (627.4%)
b               = 1                +/- 0.2169       (21.69%)

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

Figura 9: Coeficientes Función de Ajuste Lineal

Para demostrarlo gráficamente, la siguiente, gráfica es una comparación entre quicksort e inserción:

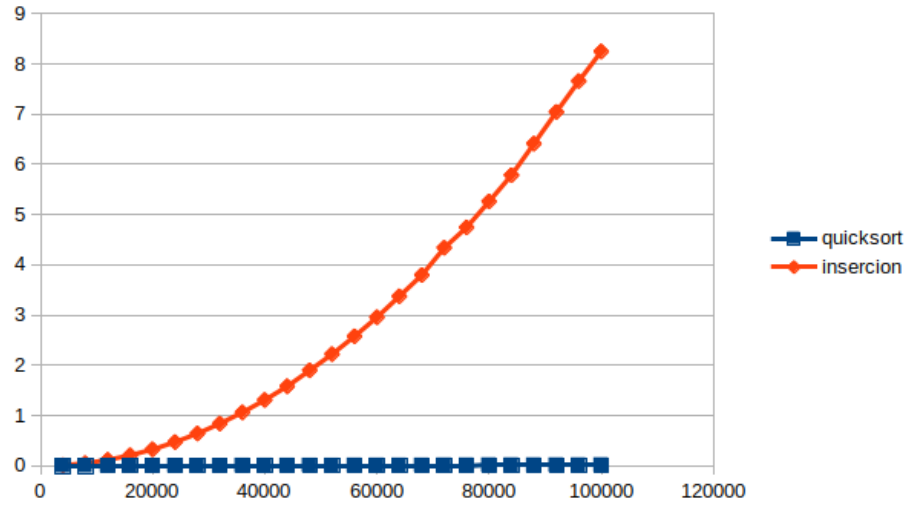


Figura 10: Comparacion Algoritmos de Ordenacion

Probando con tamaños más pequeños de los utilizados en la gráfica, he comprobado que a partir de un tamaño de 200 aproximadamente, el algoritmo quicksort es mejor que el de inserción. En tamaños menores hay casos en los que los tiempos de ejecución de inserción es mejor que el de quicksort

1.3 Algoritmo de Floyd

El último algoritmo es un representante de la clase $O(n^3)$ y sus tiempos de ejecución , sin optimización, son:

Tamaño	Media	Tiempos				
80	0.004327	0.005704	0.004873	0.005511	0.006238	0.004424
160	0.0185739	0.019629	0.019057	0.017998	0.018159	0.017992
240	0.060981	0.061361	0.059941	0.061681	0.065261	0.062678
320	0.1415652	0.145741	0.141809	0.142107	0.141659	0.140599
400	0.274456	0.277841	0.273709	0.274037	0.27411	0.274412
480	0.4719649	0.474434	0.470982	0.471363	0.471886	0.472767
560	0.7473421	0.750663	0.746781	0.747668	0.747155	0.747899
640	1.113593	1.11615	1.11319	1.11278	1.11362	1.11421
720	1.585595	1.58826	1.58462	1.585	1.5857	1.58518
800	2.170823	2.17323	2.16939	2.17003	2.17012	2.17094
880	2.892699	2.89065	2.88743	2.88805	2.88461	2.88537
960	3.741426	3.74748	3.74289	3.73925	3.74049	3.73979
1040	4.758461	4.75776	4.75375	4.75828	4.76471	4.75871
1120	6.049679	5.96812	5.9466	5.96737	6.38089	6.20308
1200	7.352899	7.31443	7.30758	7.30171	7.31062	7.35342
1280	8.877567	8.87694	8.8732	8.87373	8.87892	8.87681
1360	10.71341	10.6568	10.6727	10.6644	10.6552	10.7186
1440	12.72497	12.6714	12.6927	12.6659	13.1991	12.6676
1520	14.9355	14.9334	14.9442	14.9442	14.9365	14.9321
1600	17.6516	17.4291	18.756	17.9642	17.7647	17.4722
1680	20.18855	20.194	20.1857	20.1967	20.1932	20.1952
1760	23.25965	23.2612	23.2608	23.2665	23.2511	23.2605
1840	26.59292	26.6015	26.5873	26.5962	26.5801	26.5852
1920	30.44521	30.2508	30.9234	30.3608	30.2704	30.6373
2000	34.42601	34.855	34.1415	34.146	34.7026	34.167

Figura 11: Tabla de Tiempos de Ejecución Algoritmo de Floyd

Se puede observar que el algoritmo de Floyd es bastante más lento, que los visto anteriormente, dejando claro que es menos eficiencte, como se puede observar en la siguiente gráfica:

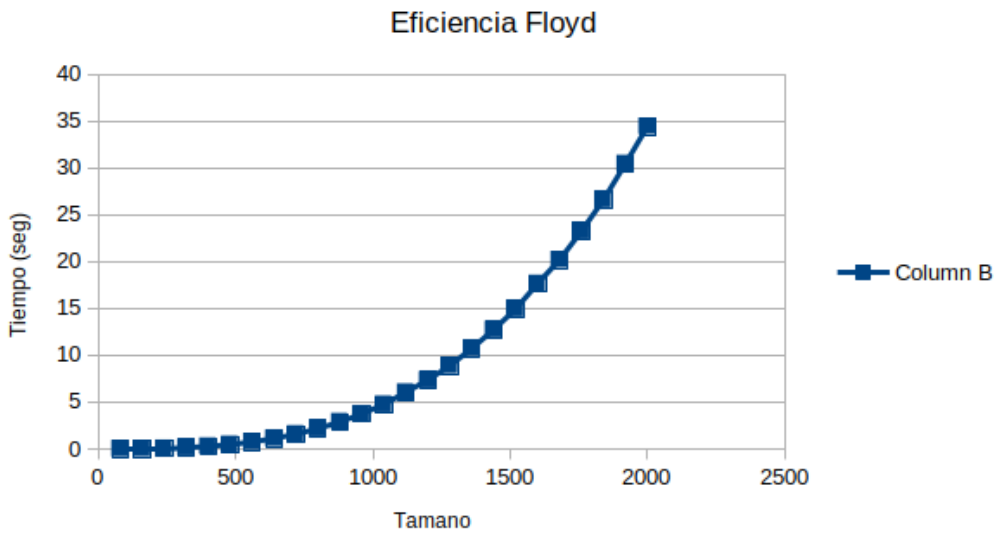
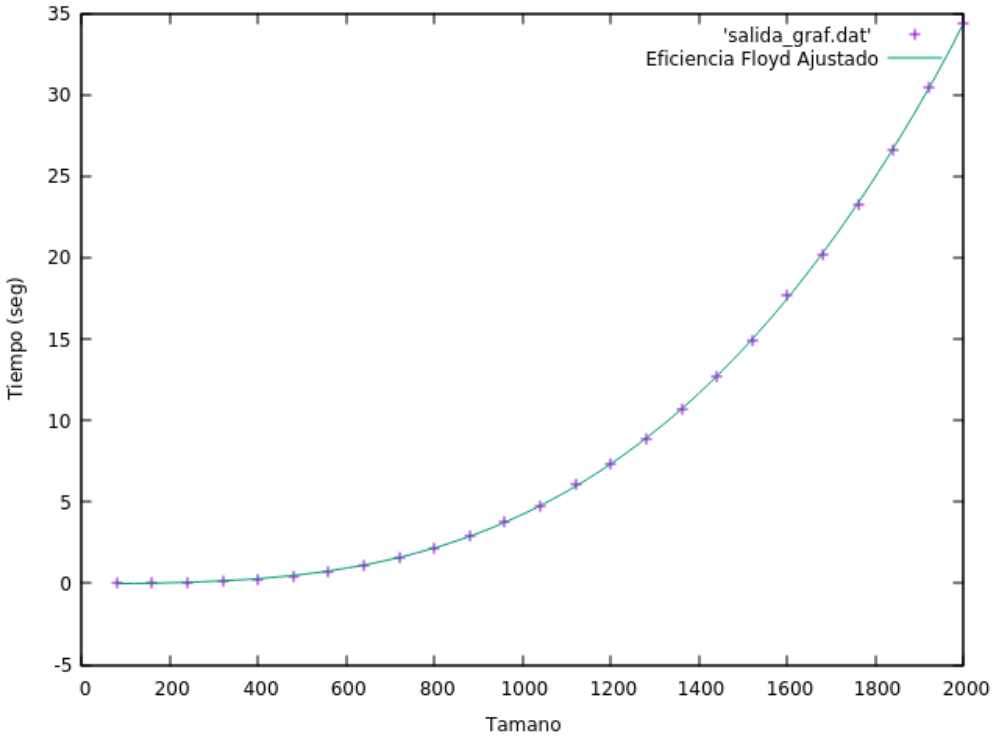


Figura 12: Grafica Algoritmo de Floyd

Además, hemos comprobado a ver si coincide la eficiencia teórica con la empírica, donde hemos obtenido que si ajustamos la nube de puntos obtenidas con el algoritmo de Floyd a una función de la forma $f(x)=a*x^3+b*x^2+c*x+d$, obtenemos lo siguiente:



Donde se aprecia que la función se ajusta perfectamente a la nube de puntos, es más, se obtiene un coeficiente de determinación de $R^2=0.9999725$, lo cual es un ajuste muy bueno.

Con el algoritmo de Floyd concluimos el cálculo, de la eficiencia empírica, así como de la híbrida de los algoritmos que he escogido para esta práctica.

2. Comparación Optimización/Sin Optimización

En esta sección vamos a haceruna comparación de como mejoran los algoritmos anteriores si al compilarlos añadimos algún tipo de optimización. En este caso hemos compilado utilizando la opción -O2, y en todos los algoritmos hemos obtenido una mejora significativa. Debido a que ya hemos observado las gráficas y las tablas en el apartado 1, en este apartado sólo mostraremos la nueva tabla de tiempos de ejecución, una gráfica del algoritmo optimizado y una gráfica comparando el algoritmo optimizado y sin optimizar.

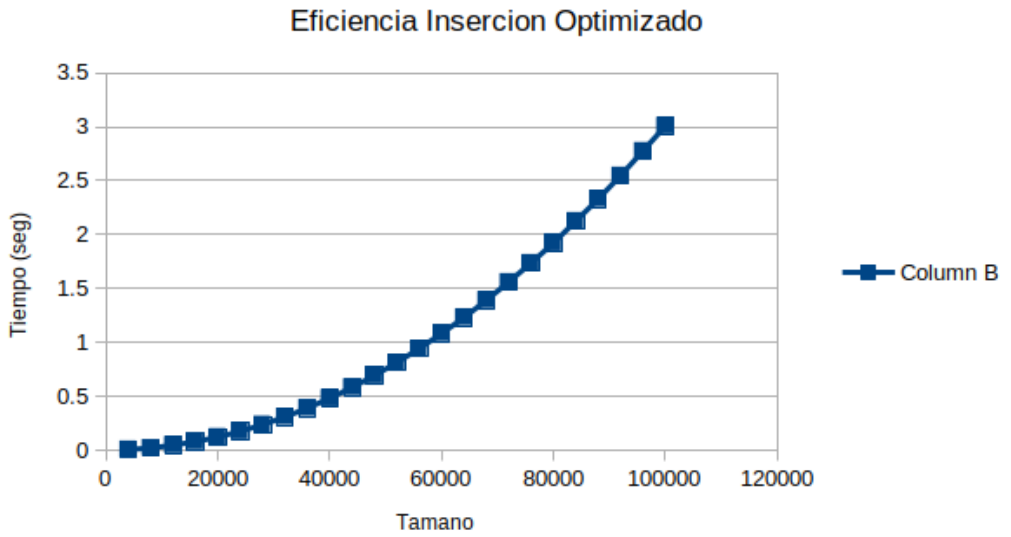
2.1 Algoritmo de Inserción

La tabla de tiempos de ejecución del algoritmo de inserción mejorado con optimización -O2 es:

Tamaño	Media	Tiempos				
4000	0.0050064	0.005298	0.005115	0.004917	0.004896	0.005096
8000	0.0195348	0.020007	0.020037	0.019191	0.01903	0.019081
12000	0.0458243	0.044823	0.045732	0.048051	0.047744	0.048832
16000	0.0779805	0.085124	0.078553	0.077444	0.076677	0.076971
20000	0.1206774	0.121951	0.121629	0.120618	0.120264	0.1202
24000	0.1738309	0.175032	0.174709	0.175145	0.172145	0.173631
28000	0.2360644	0.235206	0.234928	0.236302	0.236616	0.236128
32000	0.3078526	0.308834	0.307006	0.305862	0.308152	0.309005
36000	0.3895774	0.388735	0.388318	0.390521	0.390086	0.390404
40000	0.4813142	0.477641	0.482678	0.477953	0.479889	0.481613
44000	0.5815536	0.584673	0.582843	0.582098	0.579216	0.581874
48000	0.6920452	0.689955	0.694992	0.692414	0.691727	0.690779
52000	0.8128955	0.813049	0.811034	0.808964	0.811945	0.816168
56000	0.9417769	0.943795	0.943476	0.941344	0.939226	0.946878
60000	1.082384	1.08148	1.08415	1.0811	1.07723	1.08527
64000	1.229125	1.23279	1.22998	1.22813	1.2234	1.22613
68000	1.39041	1.39568	1.3816	1.38895	1.38836	1.38907
72000	1.558992	1.56141	1.55463	1.55615	1.55921	1.56229
76000	1.737133	1.73665	1.73661	1.73629	1.73663	1.73608
80000	1.921798	1.92054	1.92423	1.92532	1.91431	1.92261
84000	2.125755	2.12224	2.13015	2.12202	2.12073	2.12906
88000	2.330134	2.32985	2.32568	2.32727	2.33667	2.32881
92000	2.546825	2.55394	2.56668	2.54733	2.54885	2.53788
96000	2.775406	2.77905	2.7736	2.76229	2.76652	2.76967
100000	3.007505	3.02452	3.00519	2.98664	3.01231	3.00579

Figura 13: Tabla de Tiempos de Ejecución Algoritmo de Ordenación Optimizado

A partir de la tabla anterior se obtiene la siguiente gráfica:



Esta al igual que en el caso sin optimizar se ajusta a una función de la forma $f(x)=a*x^2+b*x+c$, dando lugar la siguiente gráfica:

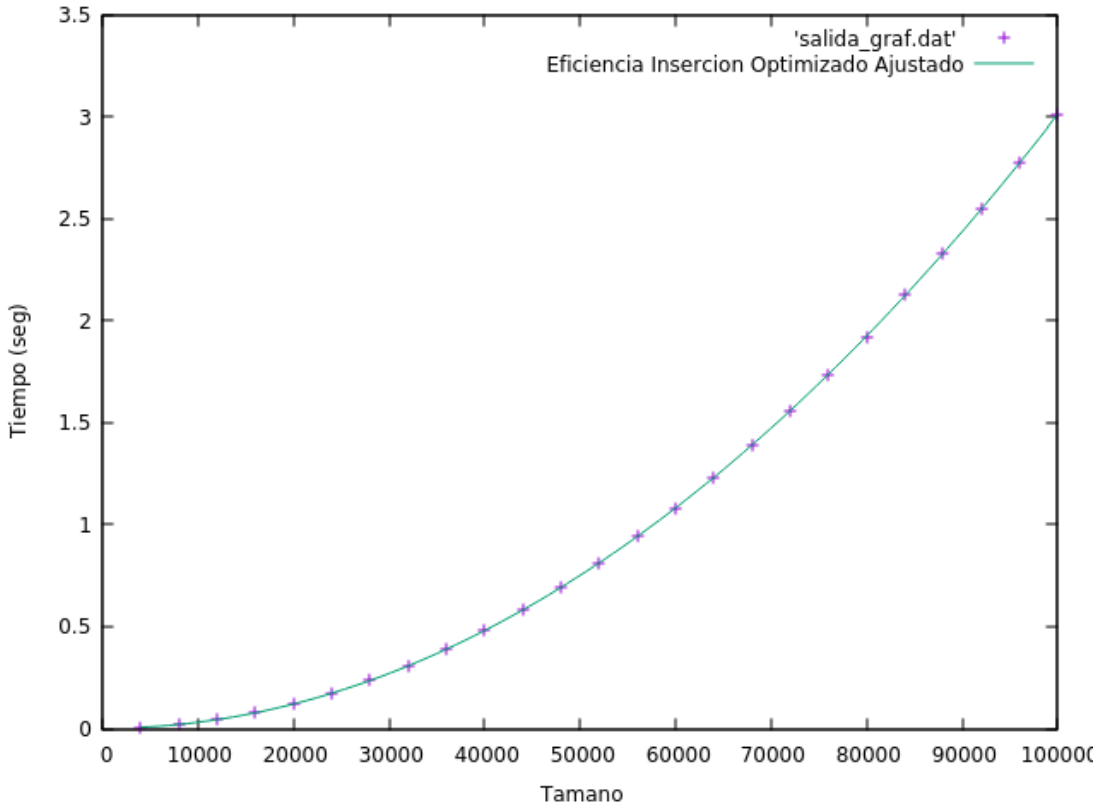


Figura 14: Algoritmo de Inserción Optimizado Ajustado

En el ajuste de la imagen anterior se puede observar un coeficiente de determinación de $R^2=0.999998$. Por último agregaremos una gráfica comparando el algoritmo sin optimizar y optimizado.

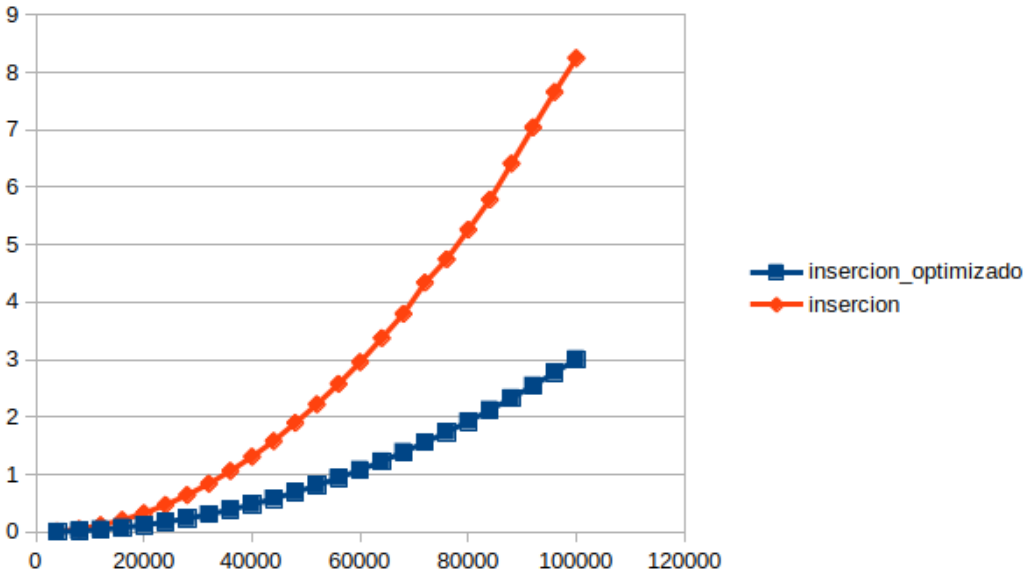


Figura 15: Comparacion Insercion Optimizada y sin Optimizar

2.2 *Algoritmo Quicksort Optimizado*

La tabla de tiempos de ejecución del quicksort optimizado es:

Tamaño	Media	Tiempos				
4000	0.0050064	0.005298	0.005115	0.004917	0.004896	0.005096
8000	0.0195348	0.020007	0.020037	0.019191	0.01903	0.019081
12000	0.0458243	0.044823	0.045732	0.048051	0.047744	0.048832
16000	0.0779805	0.085124	0.078553	0.077444	0.076677	0.076971
20000	0.1206774	0.121951	0.121629	0.120618	0.120264	0.1202
24000	0.1738309	0.175032	0.174709	0.175145	0.172145	0.173631
28000	0.2360644	0.235206	0.234928	0.236302	0.236616	0.236128
32000	0.3078526	0.308834	0.307006	0.305862	0.308152	0.309005
36000	0.3895774	0.388735	0.388318	0.390521	0.390086	0.390404
40000	0.4813142	0.477641	0.482678	0.477953	0.479889	0.481613
44000	0.5815536	0.584673	0.582843	0.582098	0.579216	0.581874
48000	0.6920452	0.689955	0.694992	0.692414	0.691727	0.690779
52000	0.8128955	0.813049	0.811034	0.808964	0.811945	0.816168
56000	0.9417769	0.943795	0.943476	0.941344	0.939226	0.946878
60000	1.082384	1.08148	1.08415	1.0811	1.07723	1.08527
64000	1.229125	1.23279	1.22998	1.22813	1.2234	1.22613
68000	1.39041	1.39568	1.3816	1.38895	1.38836	1.38907
72000	1.558992	1.56141	1.55463	1.55615	1.55921	1.56229
76000	1.737133	1.73665	1.73661	1.73629	1.73663	1.73608
80000	1.921798	1.92054	1.92423	1.92532	1.91431	1.92261
84000	2.125755	2.12224	2.13015	2.12202	2.12073	2.12906
88000	2.330134	2.32985	2.32568	2.32727	2.33667	2.32881
92000	2.546825	2.55394	2.56668	2.54733	2.54885	2.53788
96000	2.775406	2.77905	2.7736	2.76229	2.76652	2.76967
100000	3.007505	3.02452	3.00519	2.98664	3.01231	3.00579

Figura 16: Tabla Tiempos de Ejecución Quicksort Optimizado

A continuación se muestra la gráfica obtenida de la tabla anterior:

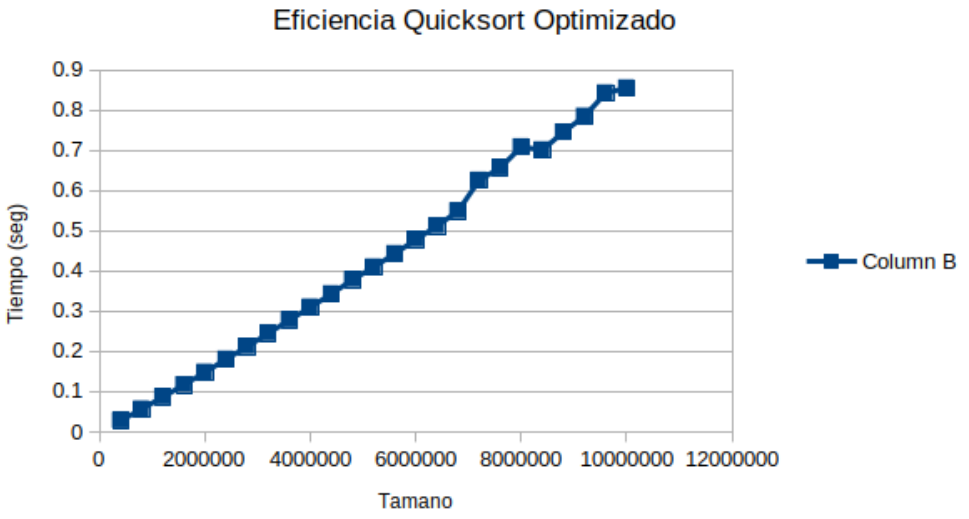


Figura 17: Quicksort Optimizado

A continuación mostramos que aunque teóricamente la nube de puntos se debería ajustar a una función de tipo f(x)=a*x*log(x)+b, realmente se ajusta mejor a una función de tipo g(x)=a*x+b, como se verá en las siguientes dos gráficas.

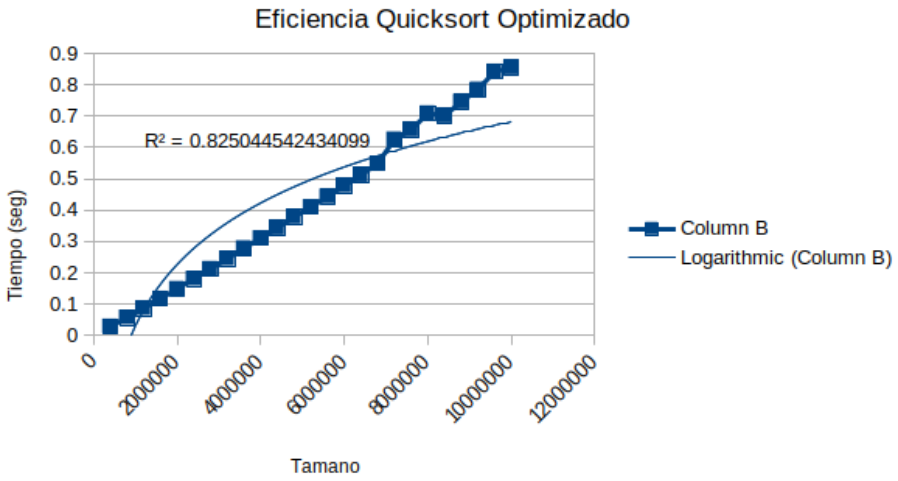


Figura 18: Quicksort Optimizado Ajuste Logarítmico

Como se puede apreciar, el ajuste no es muy bueno, es más el coeficiente de determinación es inferior incluso a 0.9. A continuación veremos el ajuste lineal.

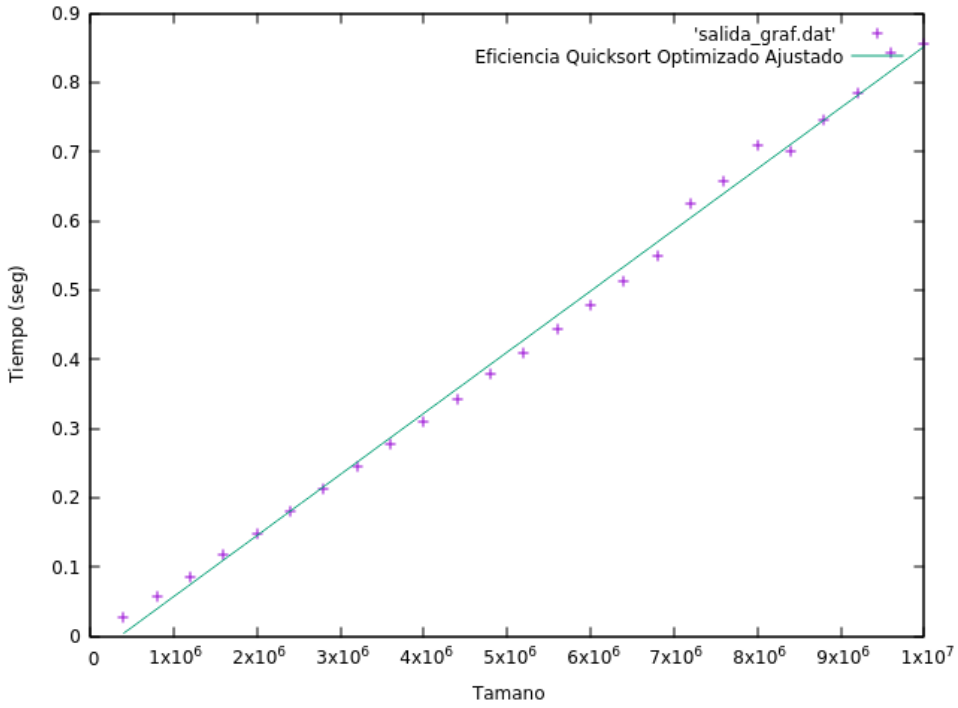


Figura 19: Quicksort Optimizado Ajuste Lineal

A simple vista se puede observar que es bastante mejor el ajuste lineal, pero de todas formas veamos que el coeficiente de determinación es R^2=0.99605, bastante mejor que para el ajuste logarítmico. Por último veamos la comparación entre con y sin optimización.

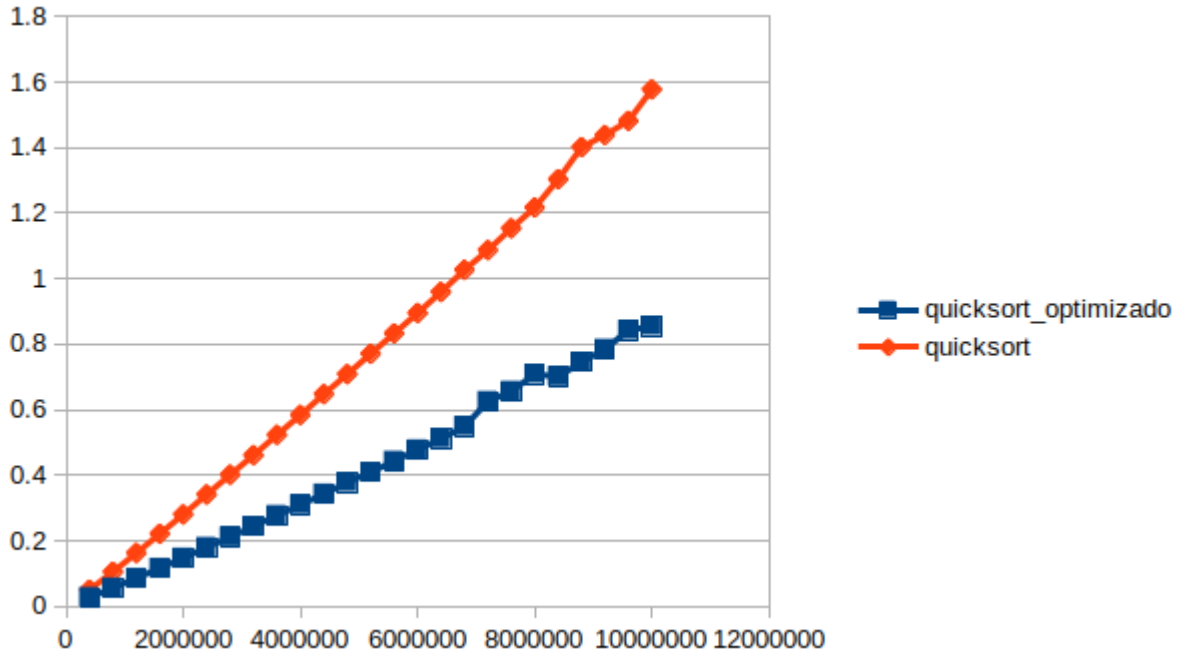


Figura 20: Comparacion Quicksort Optimizado y sin Optimizar

2.3 Algoritmo de Floyd

Por último, la tabla de tiempos de ejecución para el algoritmo de Floyd optimizado es:

Tamano	Media	Tiempos				
400000	0.0279277	0.037979	0.027013	0.026196	0.02651	0.026126
800000	0.0567158	0.057427	0.055798	0.055555	0.057626	0.059184
1200000	0.0863764	0.08894	0.085938	0.08718	0.086305	0.085804
1600000	0.1169306	0.117968	0.11558	0.117739	0.116929	0.117085
2000000	0.1482118	0.147444	0.148486	0.145032	0.149436	0.148116
2400000	0.1805992	0.18084	0.179494	0.179951	0.180664	0.178772
2800000	0.2119585	0.211784	0.207915	0.211778	0.215205	0.214695
3200000	0.2449142	0.249171	0.241139	0.244087	0.241748	0.245249
3600000	0.2778591	0.281198	0.2739	0.280397	0.277672	0.273611
4000000	0.3101874	0.31283	0.309055	0.310888	0.307849	0.30904
4400000	0.3432249	0.347272	0.342962	0.342777	0.342371	0.345666
4800000	0.3783026	0.380726	0.379658	0.38108	0.378845	0.375752
5200000	0.4098437	0.407523	0.406646	0.406932	0.4091	0.412408
5600000	0.4432609	0.442179	0.442871	0.444028	0.445217	0.438056
6000000	0.4784558	0.483766	0.476086	0.478146	0.477816	0.475279
6400000	0.5120641	0.510057	0.516818	0.51431	0.508522	0.51052
6800000	0.5491683	0.55262	0.547649	0.548386	0.551379	0.549995
7200000	0.6257125	0.580083	0.578319	0.601841	0.682057	0.701774
7600000	0.6566211	0.656802	0.691795	0.64781	0.639406	0.65322
8000000	0.7089885	0.666185	0.699843	0.714707	0.762255	0.707996
8400000	0.7011328	0.736161	0.71449	0.692916	0.694812	0.692148
8800000	0.7459186	0.736577	0.753796	0.73767	0.723682	0.729334
9200000	0.7850787	0.786282	0.755482	0.754774	0.775034	0.783102
9600000	0.8427454	0.822877	0.804928	0.836871	0.869988	0.863325
10000000	0.8552449	0.854954	0.884008	0.851278	0.917249	0.857945

Figura 21: Tabla Algoritmo de Floyd Optimizado

De esta tabla se deducirá la siguiente gráfica:

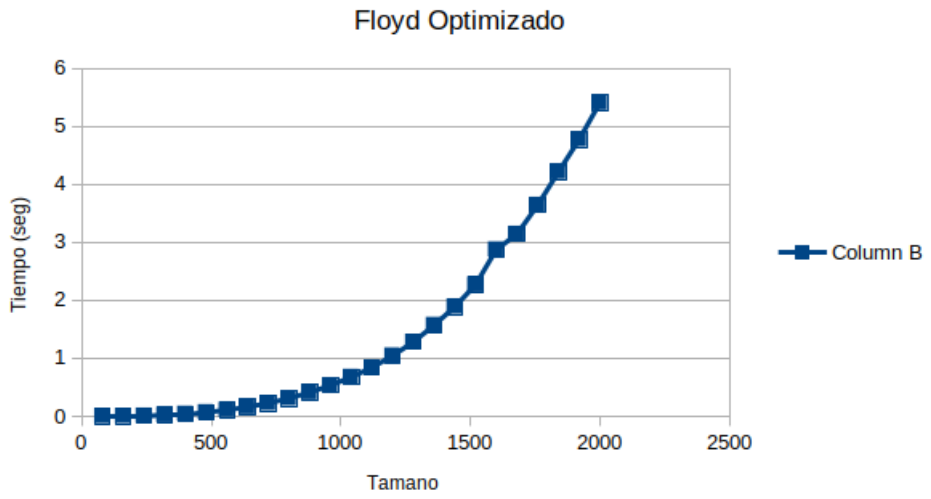


Figura 22: Grafica Floyd Optimizado

Ahora teniendo en cuenta el rendimiento teórico ($O(n^3)$), intentaremos ajustarla a una función de la forma $f(x)=a*x^3+b*x^2+c*x+d$, dando lugar a la siguiente gráfica:

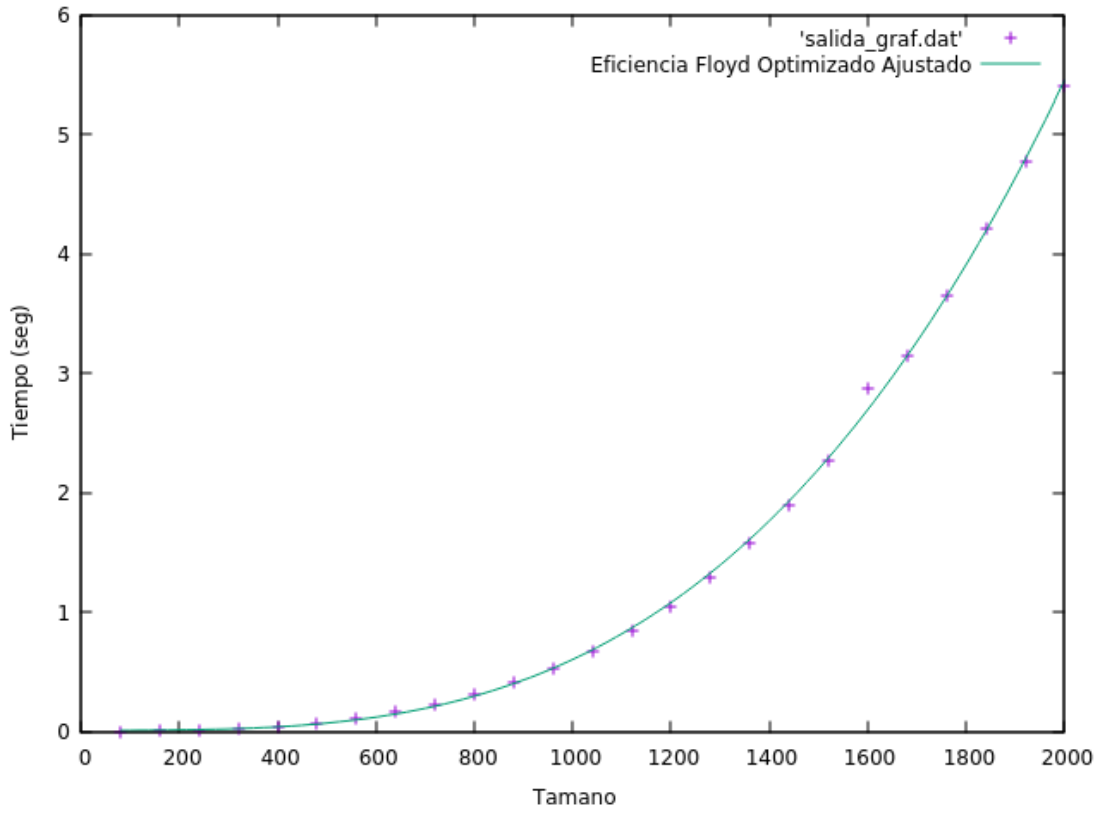


Figura 23: Floyd Optimizado Ajustado

Viendo que se trata de un muy buen ajuste, es más, el coeficiente de determinación es R^2=0.9993531. Por último veamos una gráfica comparativa entre con y sin optmiziación.

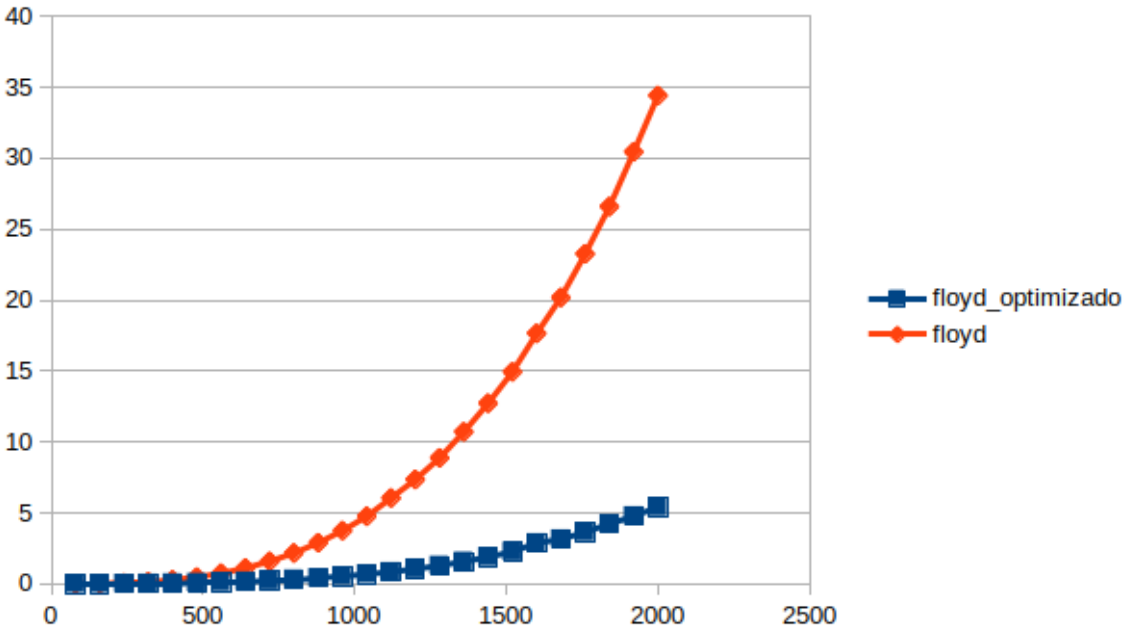


Figura 24: Comparacion Algoritmo de Floyd con y sin Optimizacion

La increíble mejora con optimización respecto a sin optimización es evidente.

3. Conclusiones

Como conclusiones a esta práctica he obtenido las siguientes:

1. Hay que ser cuidadoso con los algoritmos que se eligen, pues si encargarmos a un algoritmo cuya eficiencia es, por ejemplo, $O(n^3)$ un problema de gran tamaño, puede que no sea capaz de resolverlo en un tiempo aceptable. He podido observar de primera mano la gran diferencia que hay entre un algoritmo como Quicksort, que ordena 10000000 de elementos en un segundo y medio, frente a algoritmos como el de inserción que tarda ya ocho segundos para tamaños mucho menores, como son los 100000 elementos utilizados.
2. Otro punto a tener en cuenta es la increíble diferencia que se puede llegar a alcanzar en un mismo algortimo al compilar con y sin optimización. Este aspecto se ha hecho especialmente visible en el algoritmo de Floyd, que de tardar 35 segundo para procesar 2000 elementos ha pasado a apenas 5 segundo, dando lugar a una mejora considerable.
3. También se puede deducir que aunque la eficiencia teórica diga que un algortimo es de un determinado orden, luego en la práctica puede darse que no lo sea realmente, como ha ocurrido con el algoritmo Quicksort, que se ajustaba mejor a una función linal, a pesar de tratarse un algoritmo de orden O(n*log(n))