UGR

Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones
Grado en Ingeniería Informática

Asignatura: Algorítmica

Práctica 1

Autores:
Míriam Mengíbar Rodríguez
Néstor Rodríguez Vico
Ángel Píñar Rivas

Granada, 10 de marzo de 2016

Índice:

1. Introducción.

En primer lugar, hemos modificado los códigos proporcionados para que muestre una salida más cómoda para manejar los datos. Para ello, en la salida del programa se mostrará el tamaño y el tiempo de ejecución separado por un espacio. Hemos utilizado la macro adaptándola a los distintos algoritmos (cambiando nombre del ejecutable, número de elementos, etc). Así a la hora de generar los gráficos nos resultará más cómodo.

Cuando ejecutamos los algoritmos, a priori, podemos deducir, de entre los que tienen el mismo orden de eficiencia, cual de ellos será más o menos eficiente.

A continuación, hemos generado los gráficos a partir de los ficheros de datos con GNUPLOT. También nos hemos servido de esta herramienta para realizar los ajustes de la función generada por nuestros datos con su correspondiente orden de eficiencia y observar las constantes ocultas.
Para comprobar que efectivamente el algoritmo se corresponde con su orden de eficiencia teórico, hemos ilustrado los datos junto con la función ajustada en el mismo gráfico.

Para satisfacer nuestra curiosidad acerca de la veracidad del análisis, decidimos compilar con un nivel de optimización mayor, realizando sus correspondientes gráficos para el análisis, viendo así que con los mismos datos se pueden obtener tiempos de ejecución diferentes en la misma computadora.
Por otra parte, ajustamos los datos a funciones que no se corresponden con la función que representa el orden de eficiencia para observar la diferencia entre los datos representados y un orden de eficiencia que no se corresponde.

2. Cálculo de la eficiencia empírica.

-Algoritmos $O(n^2)$:

Tamaño (n)	Burbuja	Inserción	Selección
1000	0,00788237	0,00441383	0,00457589
2000	0,01477620	0,00879420	0,01316770
3000	0,02801130	0,01204690	0,02069200
4000	0,04618930	0,02119060	0,03255930
5000	0,06047170	0,02956570	0,03889970
6000	0,08848080	0,04596440	0,04936160
7000	0,12500800	0,06264950	0,06756980
8000	0,16721300	0,07533030	0,08717010
9000	0,21691100	0,11270800	0,11018200
10000	0,27312500	0,11644000	0,13574600
11000	0,33464900	0,14435700	0,16445500
12000	0,40321100	0,16859000	0,19597800
13000	0,47890200	0,19864800	0,23139500
14000	0,56141500	0,22464200	0,26773400
15000	0,65915500	0,26459000	0,30710300
16000	0,75242900	0,29581900	0,34791600
17000	0,85922700	0,33730200	0,40308700
18000	0,96170300	0,37362300	0,44099800
19000	1,06951000	0,41644200	0,49121700
20000	1,19724000	0,46979200	0,54556800
21000	1,33572000	0,50486400	0,59987800
22000	1,45902000	0,56656700	0,66003600
23000	1,61162000	0,62377600	0,72179000
24000	1,75526000	0,66247300	0,78529400
25000	1,91238000	0,74554500	0,85024900

-Algoritmos O(nlog(n)):

Tamaño (n)	Mergesort	Quicksort	Heapsort
1000	0,000348421	0,000089372	0,000139449
2000	0,000421394	0,000239261	0,000357171
3000	0,000816062	0,000443156	0,000473314
4000	0,000773576	0,000519613	0,000662576
5000	0,000970916	0,000693103	0,000824775
6000	0,001307020	0,000915989	0,000894614
7000	0,001375880	0,000895099	0,001187930
8000	0,002153790	0,001076360	0,001293050
9000	0,001889980	0,001317180	0,001451660
10000	0,001956840	0,001364110	0,001997820
11000	0,002386320	0,001460560	0,001932310
12000	0,002619400	0,001445240	0,002088500
13000	0,003113940	0,001630020	0,002345570
14000	0,002859210	0,001779090	0,002861440
15000	0,003485080	0,002017360	0,002560330
16000	0,003696280	0,002129060	0,003175340
17000	0,003391910	0,002152930	0,003348040
18000	0,004667890	0,002645580	0,003742030
19000	0,004830510	0,002547260	0,003487860
20000	0,004471570	0,003015840	0,003420030
21000	0,004459080	0,002782340	0,003611200
22000	0,004859170	0,003488290	0,004277830
23000	0,005095820	0,002900540	0,005270150
24000	0,006138640	0,003715630	0,004326780
25000	0,008681480	0,003245150	0,004548800

-Floyd O(n^3):

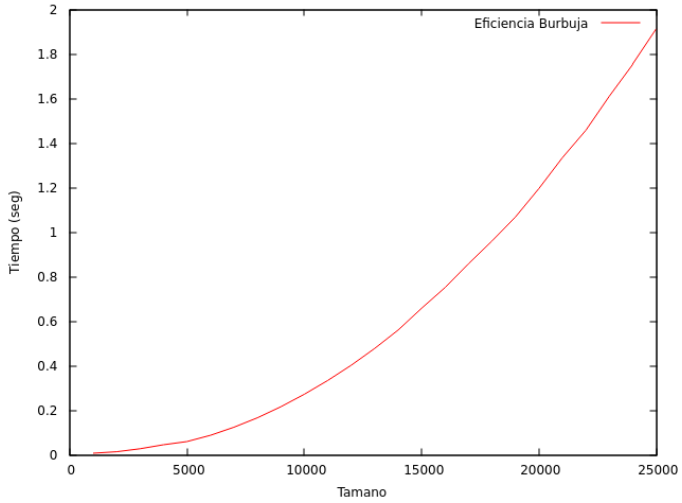
Tamaño (n)	Floyd
100	0,0153342
200	0,0602728
300	0,1906970
400	0,3963710
500	0,7552540
600	1,2863600
700	2,0585300
800	3,0471200
900	4,4482500
1000	6,0862000
1100	8,1648300
1200	10,8145000
1300	13,3560000
1400	16,5162000
1500	20,5134000
1600	24,6053000
1700	29,3076000
1800	34,7590000
1900	40,9219000
2000	47,5747000
2100	54,8748000
2200	63,1209000
2300	72,8652000
2400	81,7078000
2500	92,0205000

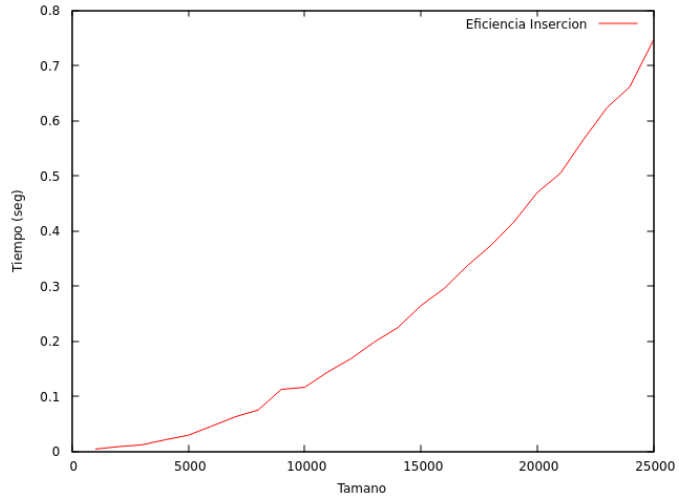
-Hanoi O(2^n):

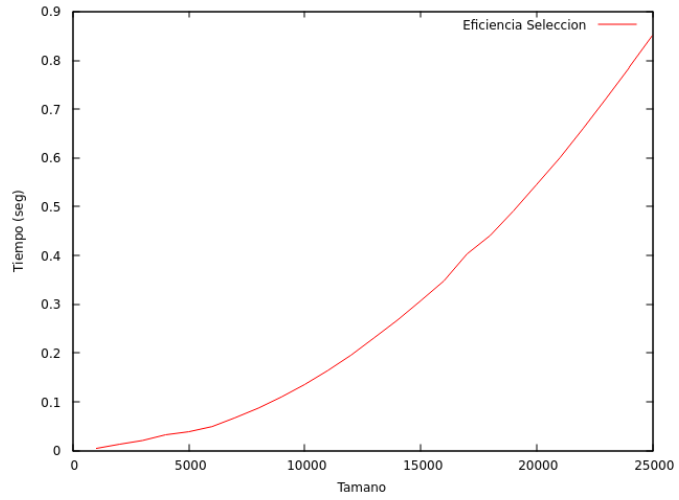
Tamaño (n)	Hanoi
11	0,000049388
12	0,000095221
13	0,000097116
14	0,000140576
15	0,000263059
16	0,000438135
17	0,001069090
18	0,002443660
19	0,004090040
20	0,008017000
21	0,015165200
22	0,028768300
23	0,056720900
24	0,114949000
25	0,223017000
26	0,446514000
27	0,886230000
28	1,760720000
29	3,525010000
30	6,974380000
31	13,93350000
32	27,74360000
33	55,38720000
34	110,7680000
35	222,0150000

3. Gráficos a partir de los datos obtenidos.

3.1 Algoritmos $O(n^2)$.

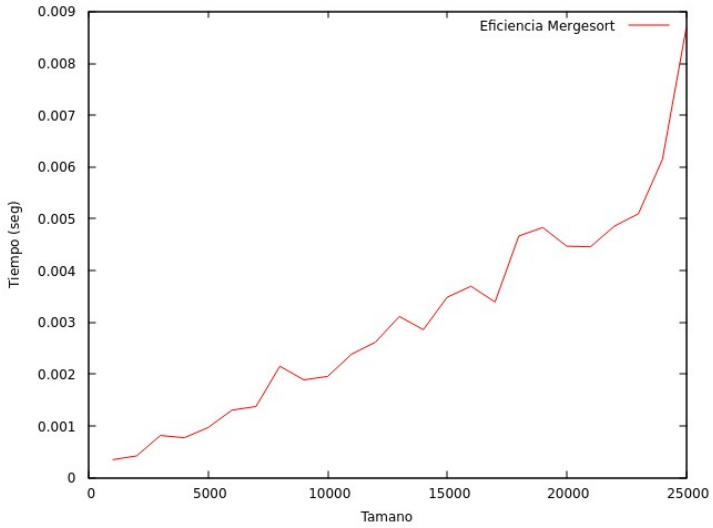


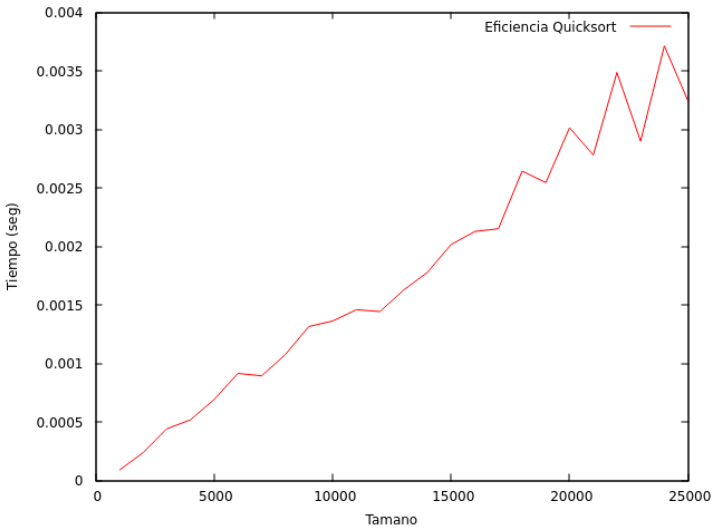


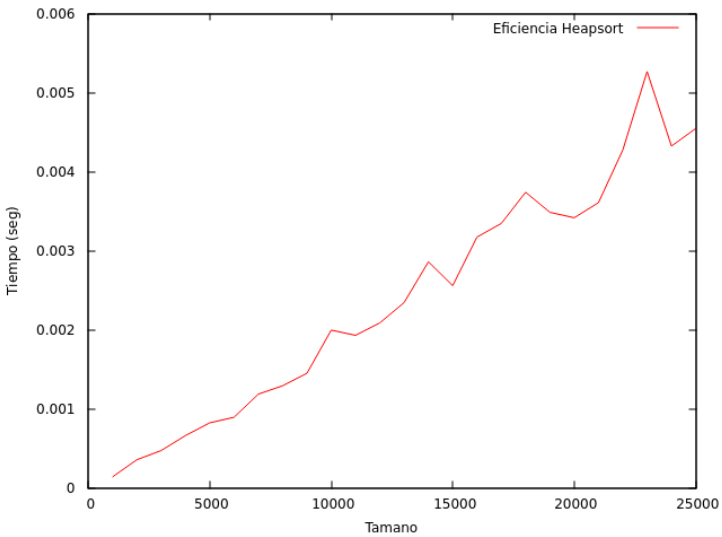


Como podemos observar en las gráficas, los algoritmos de inserción, selección y burbuja son algoritmos del mismo orden de eficiencia (O(n)). En particular, el algoritmo de burbuja es menos eficiente debido a la existencia de una constante oculta, que es mayor que la de los algoritmos de inserción y selección, por lo cual es más lento.

3.2. Algoritmos O(nlog(n)).

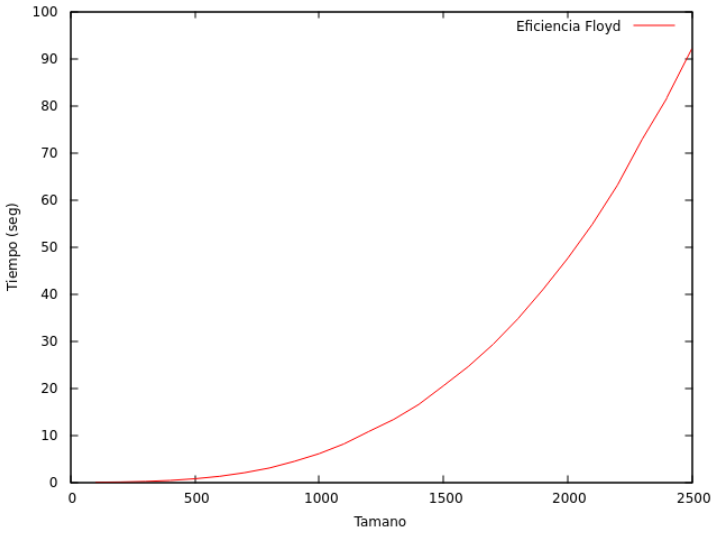






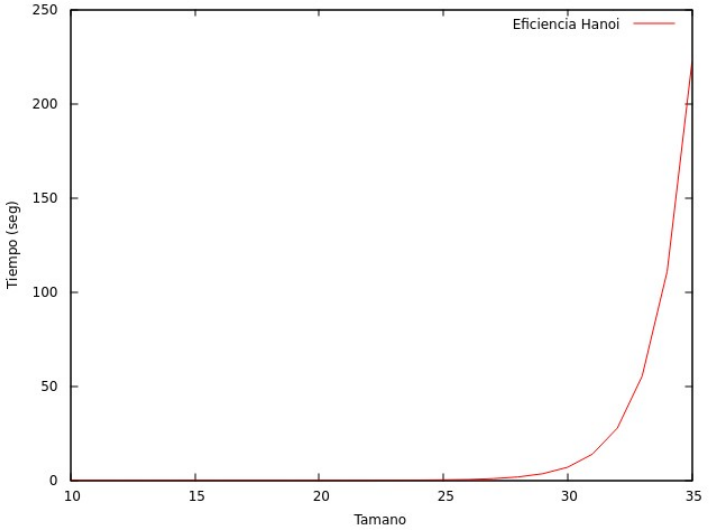
Como podemos observar en las gráficas, los algoritmos mergesort, heapsort y quicksort son algoritmos del mismo orden de eficiencia (O(nlog(n))). En particular, el algoritmo quicksort es el más eficiente debido a la existencia de una constante oculta, que es menor que la de los algoritmos mergesort y quicksort, por lo cual es más rápido.

3.3. Floyd O(n^3).



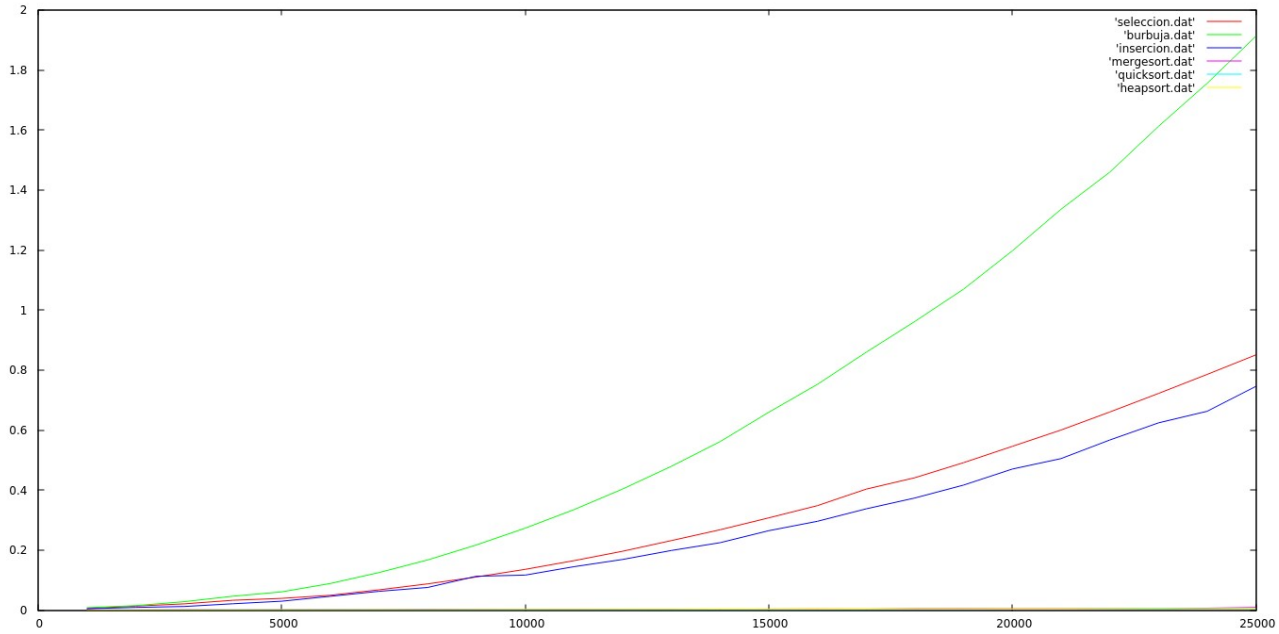
Podemos ver que para valores pequeños, el tiempo empleado es poco, pero conforme aumenta el tamaño (el valor de n) el tiempo empleado en la ejecución del algoritmo Floyd aumenta de forma exagerada, siendo un algoritmo de O(n^3), a partir de valores cercanos a 100.

3.4. Hanoi O(2^n).



Podemos ver claramente como para valores pequeños, el tiempo empleado es diminuto, pero conforme aumenta el tamaño (el valor de n) el tiempo empleado en la ejecución del algoritmo Hanoi aumenta de forma potencial, creciendo muy rápidamente a partir de un tamaño de 30.

He aquí el gráfico con todos los algoritmos de ordenación:



4. Calculo de la eficiencia híbrida.

4.1. Algoritmos O(n^2).

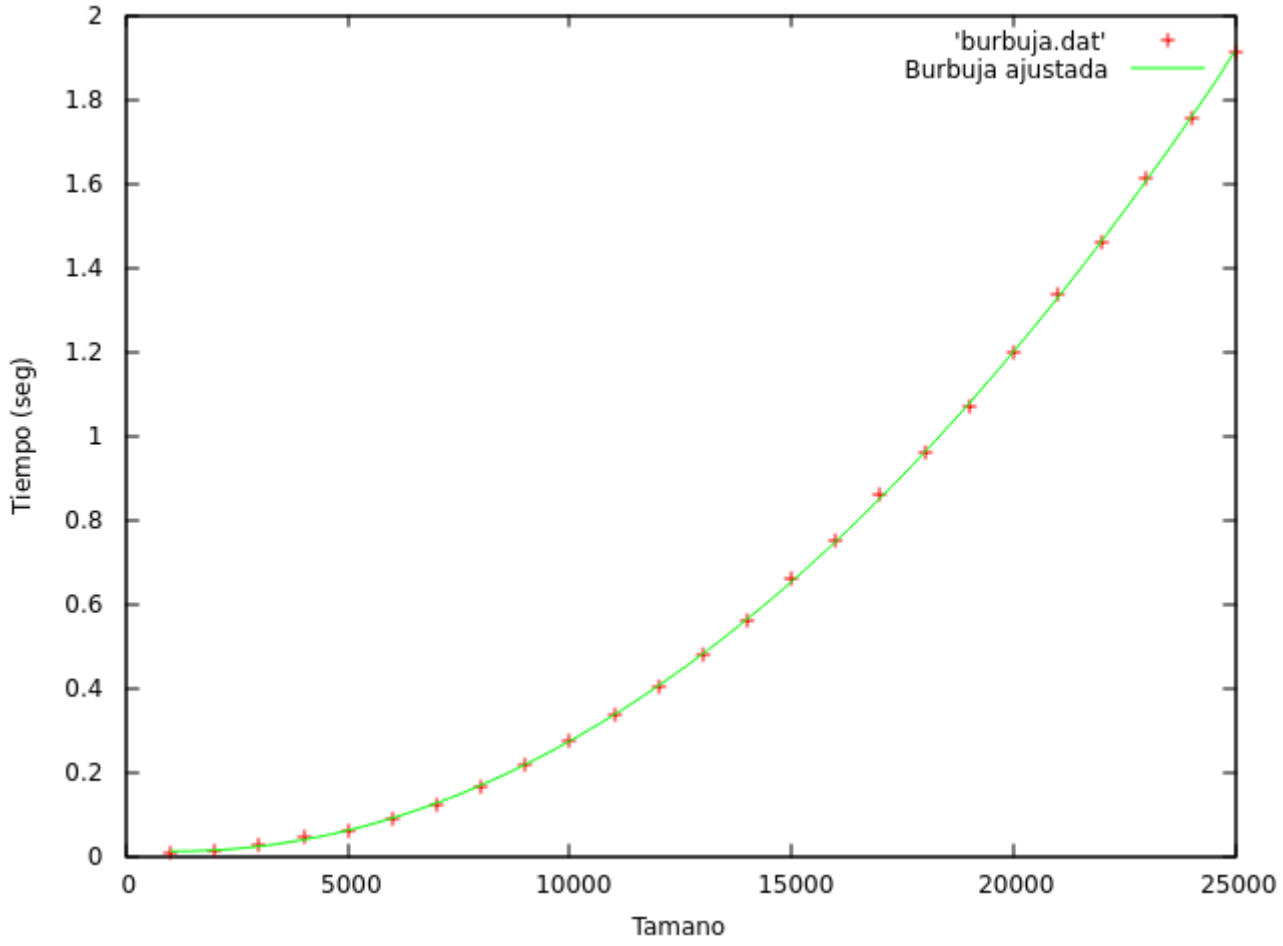
Para los algoritmos O(n^2) hemos definido una función como sigue:
f(x) = a0*x*x+a1*x+a2.
Una vez hemos definido esa función, hemos hecho un ajuste como sigue:
fit f(x) 'xxxxxx.dat' via a0,a1,a2
donde xxxxxx indica el nombre del fichero que contiene los datos generados por el algoritmo (tamaño y su correspondiente tiempo de ejecución).
Una vez se ha hecho el ajuste, hemos pintado la función f(x) junto con los datos obtenidos de forma empírica para poder ver las diferencias entre el cálculo teórico y el cálculo empírico.

-Burbuja.

Las constantes ocultas obtenidas mediante el ajuste son:

a0	= 3.34101e-09	+/- 1.807e-11	(0.5408%)
a1	= -7.65998e-06	+/- 4.839e-07	(6.317%)
a2	= 0.0163979	+/- 0.00273	(16.65%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:

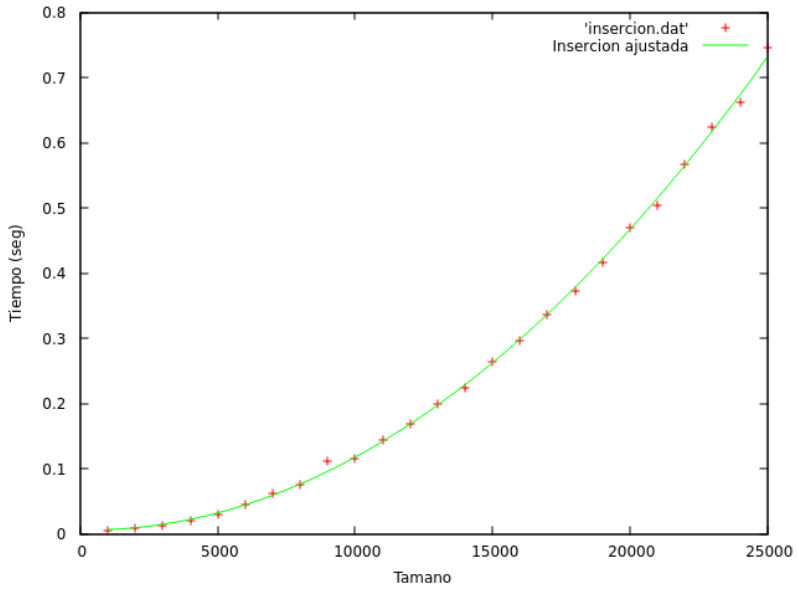


-Inserción.

Las constantes ocultas obtenidas mediante el ajuste son:

a0	= 1.18943e-09	+/- 2.727e-11	(2.292%)
a1	= -7.51316e-07	+/- 7.303e-07	(97.2%)
a2	= 0.00634759	+/- 0.004121	(64.91%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:

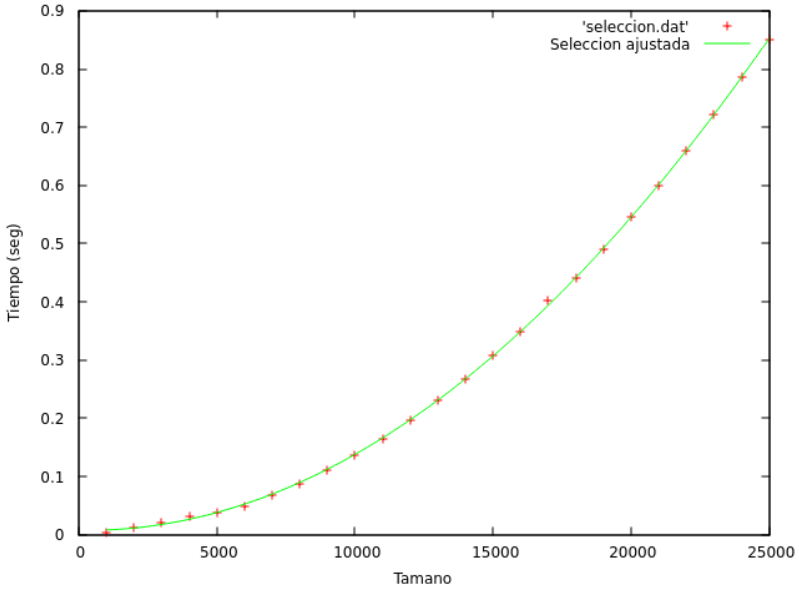


-Selección.

Las constantes ocultas obtenidas mediante el ajuste son:

a0	= 1.38517e-09	+/- 1.273e-11	(0.9192%)
a1	= -8.31914e-07	+/- 3.41e-07	(41%)
a2	= 0.00759749	+/- 0.001924	(25.33%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:



4.2. Algoritmos O(nlog(n)).

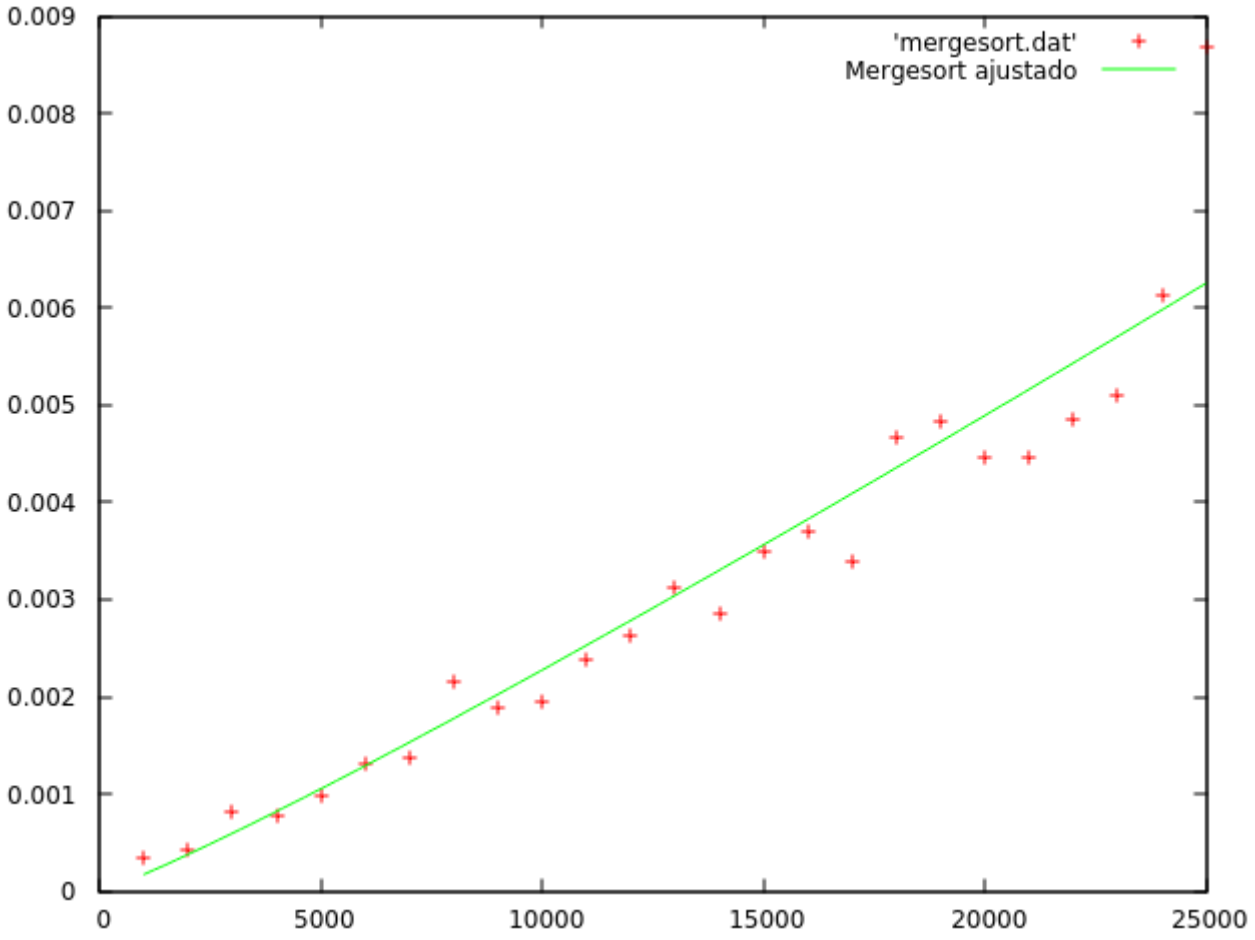
Para los algoritmos O(nlog(n) hemos definido una función como sigue:
f(x) = a0*x*log(a1*x)
Una vez hemos definido esa función, hemos hecho un ajuste como sigue:
fit f(x) 'xxxxxx.dat' via a0,a1
donde xxxxxx indica el nombre del fichero que contiene los datos generados por el algoritmo (tamaño y su correspondiente tiempo de ejecución).
Una vez se ha hecho el ajuste, hemos pintado la función f(x) junto con los datos obtenidos de forma empírica para poder ver las diferencias entre el cálculo teórico y el cálculo empírico.

-Mergesort.

Las constantes ocultas obtenidas mediante el ajuste son:

a0	= 2.49647e-08	+/- 2.465e-08	(98.72%)
a1	= 0.901234	+/- 8.648	(959.5%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:

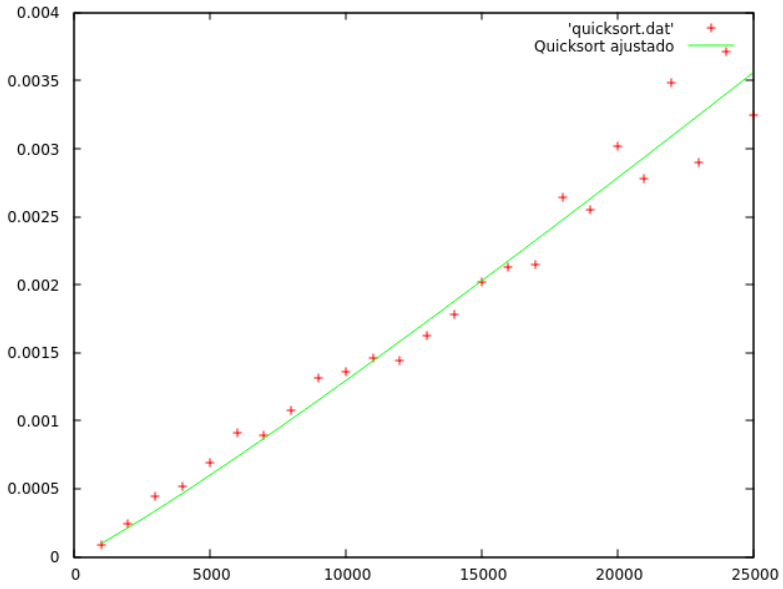


-Quicksort.

Las constantes ocultas obtenidas mediante el ajuste son:

```
a0              = 1.42028e-08      +/- 7.313e-09    (51.49%)
a1              = 0.901234         +/- 4.51         (500.4%)
```

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:

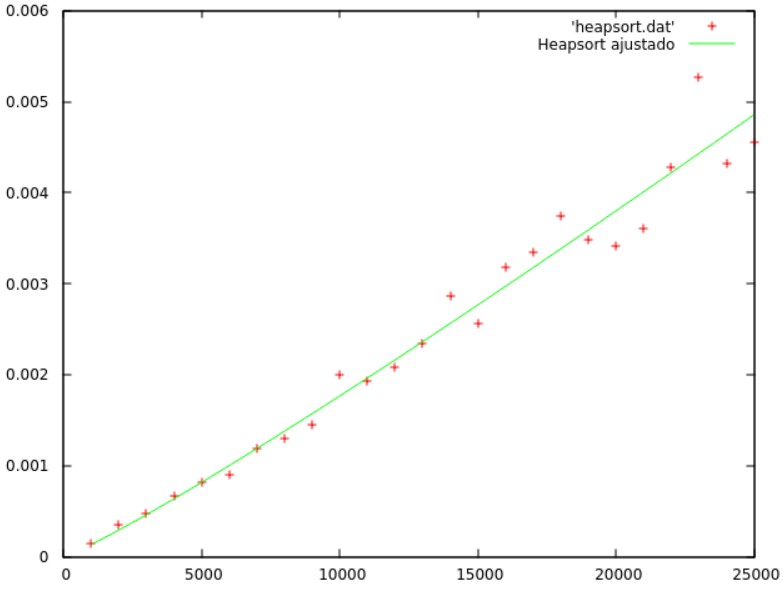


-Heapsort.

Las constantes ocultas obtenidas mediante el ajuste son:

```
a0              = 1.9381e-08       +/- 1.08e-08     (55.75%)
a1              = 0.901234         +/- 4.883        (541.8%)
```

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:



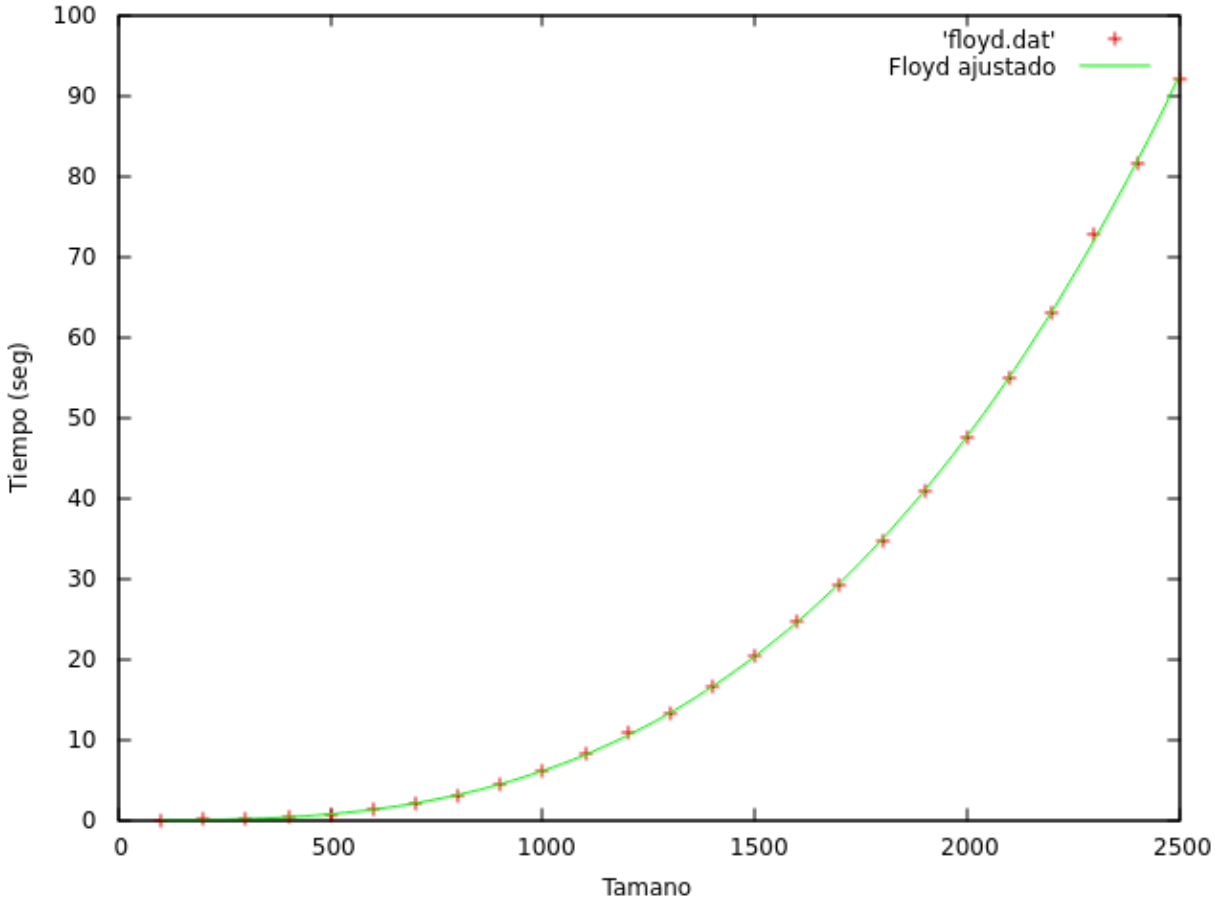
4.3. Floyd O(n^3).

Hemos definido una función como sigue:
f(x) = a0*x*x*x+a1*x*x+a2*x+a3
Una vez hemos definido esa función, hemos hecho un ajuste como sigue:
fit f(x) 'xxxxxx.dat' via a0,a1,a3,a3
donde xxxxxx indica el nombre del fichero que contiene los datos generados por el algoritmo (tamaño y su correspondiente tiempo de ejecución).
Una vez se ha hecho el ajuste, hemos pintado la función f(x) junto con los datos obtenidos de forma empírica para poder ver las diferencias entre el cálculo teórico y el cálculo empírico.

Las constantes ocultas obtenidas mediante el ajuste son:

a0	= 5.62321e-09	+/- 1.433e-10	(2.549%)
a1	= 8.68854e-07	+/- 5.662e-07	(65.16%)
a2	= -0.000424177	+/- 0.0006401	(150.9%)
a3	= 0.0518761	+/- 0.196	(377.8%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:



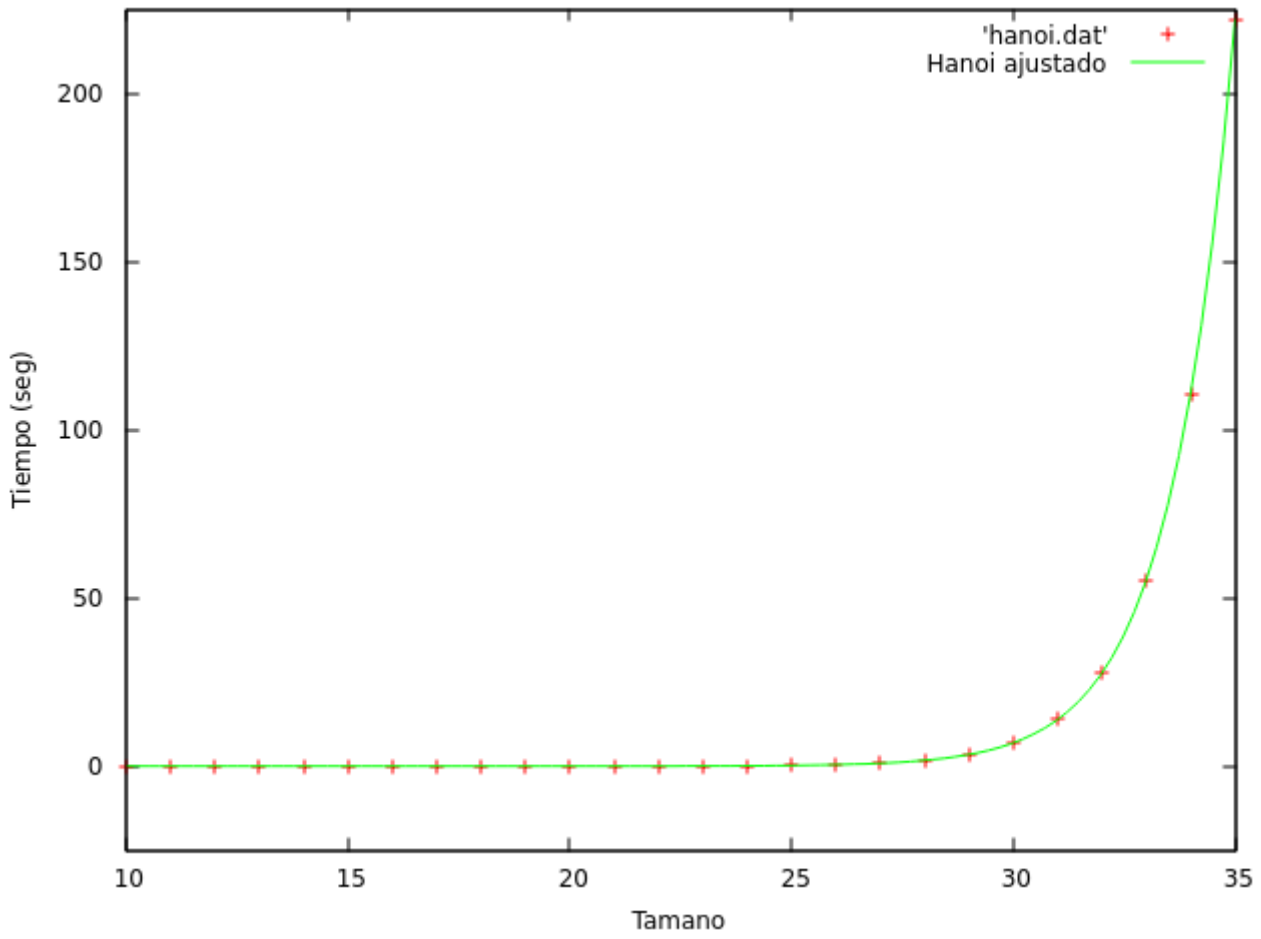
4.4. Hanoi O(2ⁿ).

Hemos definido una función como sigue:
f(x) = a0*2**x
Una vez hemos definido esa función, hemos hecho un ajuste como sigue:
fit f(x) 'xxxxxx.dat' via a0
donde xxxxxx indica el nombre del fichero que contiene los datos generados por el algoritmo (tamaño y su correspondiente tiempo de ejecución).
Una vez se ha hecho el ajuste, hemos pintado la función f(x) junto con los datos obtenidos de forma empírica para poder ver las diferencias entre el cálculo teórico y el cálculo empírico.

La constante oculta obtenidas mediante el ajuste es:

a0 = 6.45834e-09 +/- 1.281e-12 (0.01983%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:



5. Ajustes erróneos.

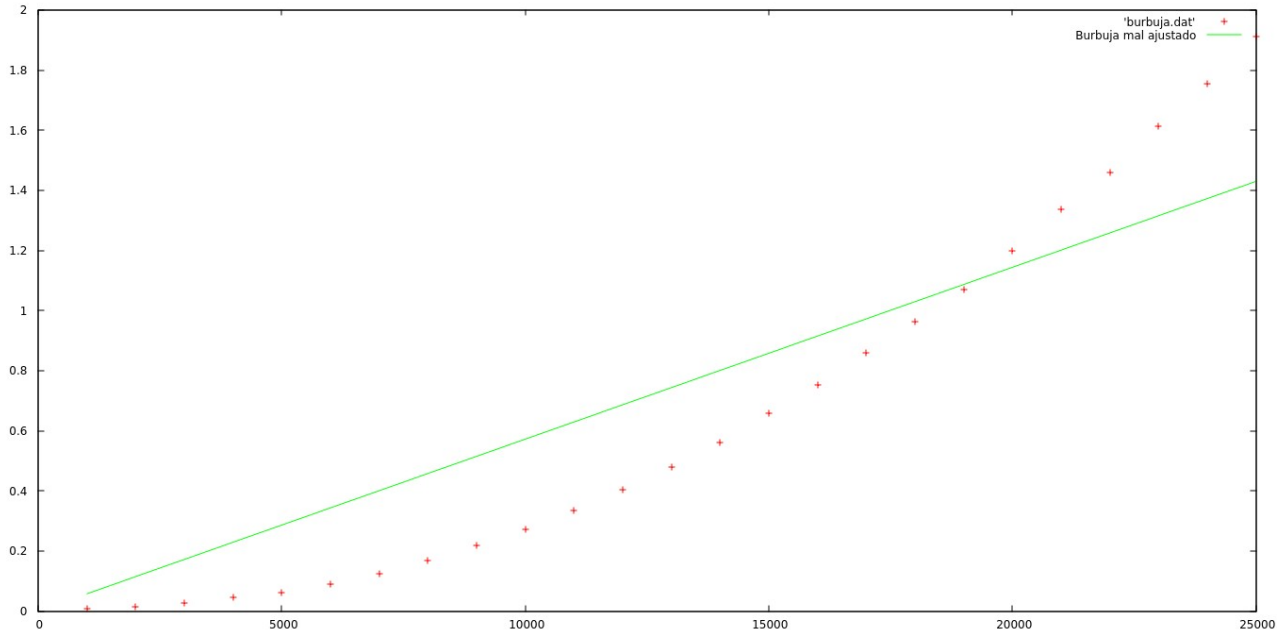
Hemos probado otros ajustes que no se corresponden con la eficiencia teórica. Para los algoritmos de $O(n^2)$, floyd y hanoi los hemos ajustado a función lineal (f(x) = a0*x) y para los algoritmos de O(nlog(n)) los hemos ajustado a una función cuadrática (f(x) = a0*x*x+a1*x+a2).

5.1. Algoritmos $O(n^2)$.

Hemos usado el algoritmo burbuja como ejemplo. La constante oculta obtenidas mediante el ajuste es:

a0	= 2.26855e-09	+/- 1.071e-10	(4.722%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:

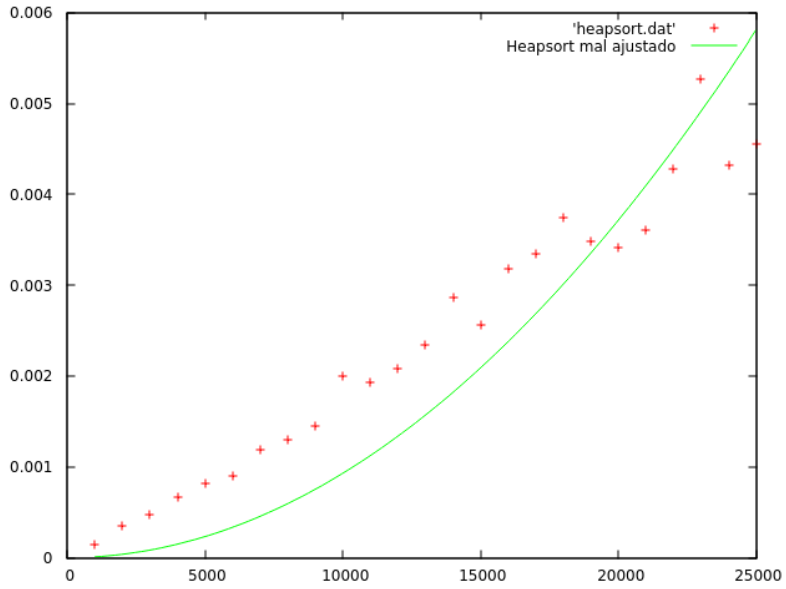


5.2. Algoritmos O(nlog(n)).

Hemos usado el algoritmo heapsort como ejemplo. Las constantes ocultas obtenidas mediante el ajuste son:

```
a0              = 2.45534e-08      +/- 1.557e-08    (63.41%)
a1              = 1                +/- 2.077        (207.7%)
```

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:

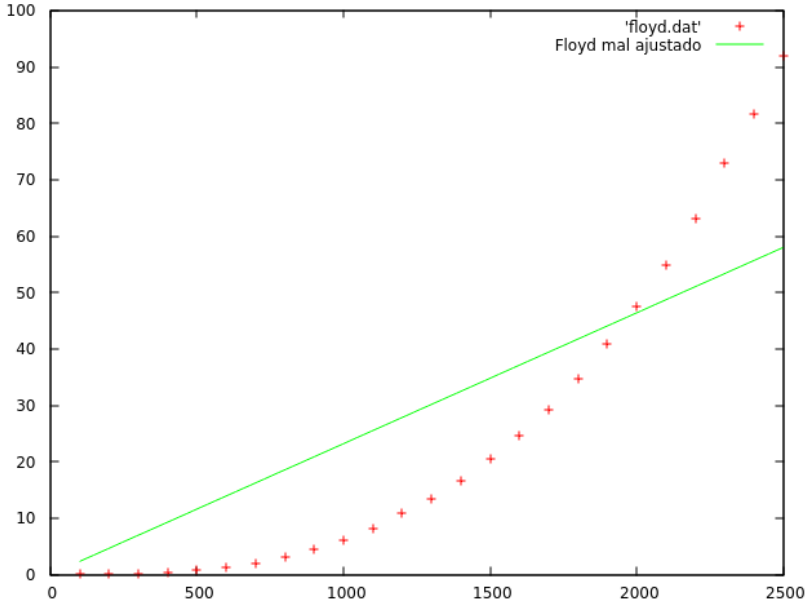


5.3. Floyd O(n^3).

La constante oculta obtenidas mediante el ajuste es:

```
a0              = 2.40491e-09      +/- 1.125e-10    (4.677%)
```

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:

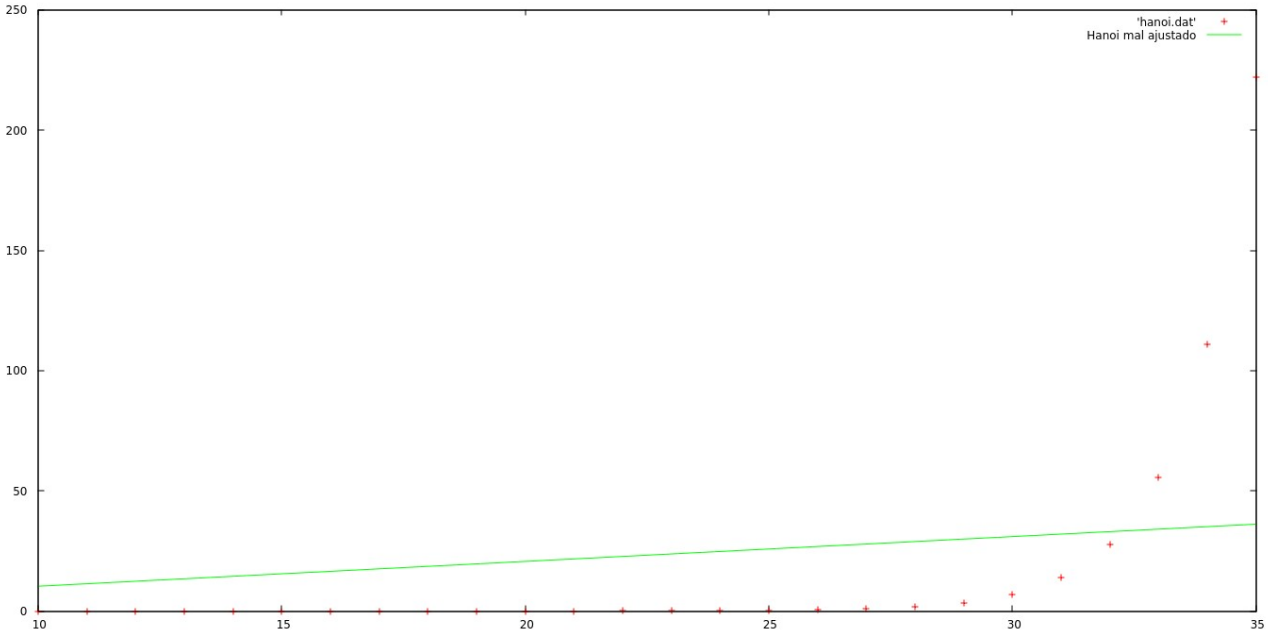


5.4. Hanoi O(2ⁿ).

La constante oculta obtenidas mediante el ajuste es:

a0	= 2.26889e-09	+/- 1.066e-12	(0.04698%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:



6. Algoritmos con otras optimizaciones.

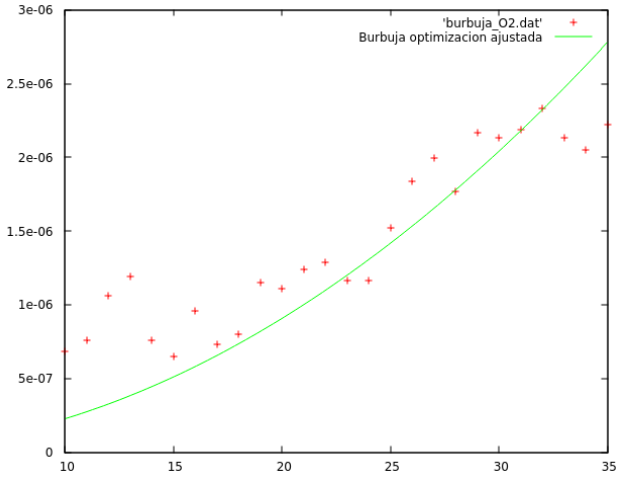
Hemos compilado con optimización -O2, para ver como afecta esto al tiempo de ejecución de los distintos algoritmos.

6.1. Algoritmos O(n^2).

Hemos usado el algoritmo burbuja como ejemplo. La constante oculta obtenidas mediante el ajuste es:

a0	= 2.26855e-09	+/- 1.071e-10	(4.722%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:

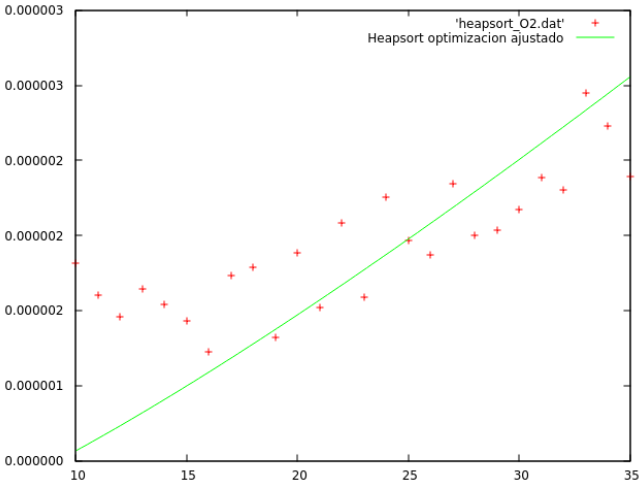


6.2. Algoritmos O(nlog(n)).

Hemos usado el algoritmo heapsort como ejemplo. Las constantes ocultas obtenidas mediante el ajuste son:

a0	= 2.45534e-08	+/- 1.557e-08	(63.41%)
a1	= 1	+/- 2.077	(207.7%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:

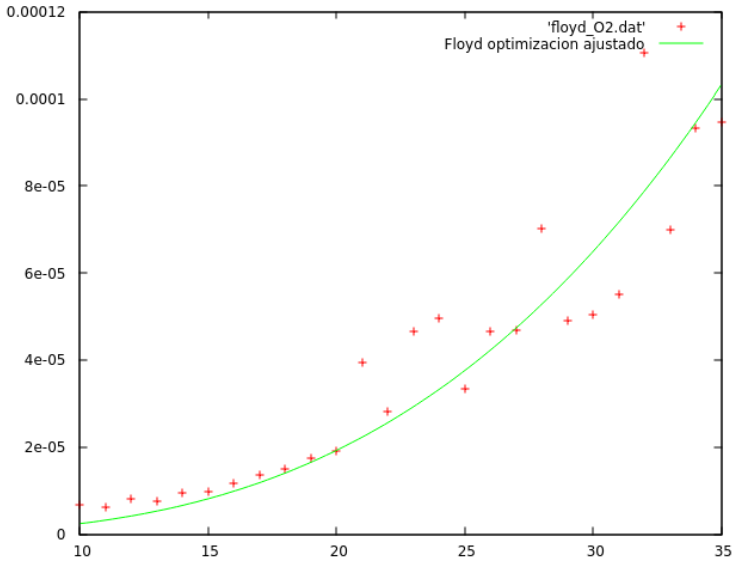


6.3. Floyd O(n^3).

La constante oculta obtenidas mediante el ajuste es:

a0	= 2.40491e-09	+/- 1.125e-10	(4.677%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente:



6.4. Hanoi O(2^n).

La constante oculta obtenidas mediante el ajuste es:

a0	= 2.26889e-09	+/- 1.066e-12	(0.04698%)

El gráfico con la función y los datos obtenidos empíricamente es el siguiente: