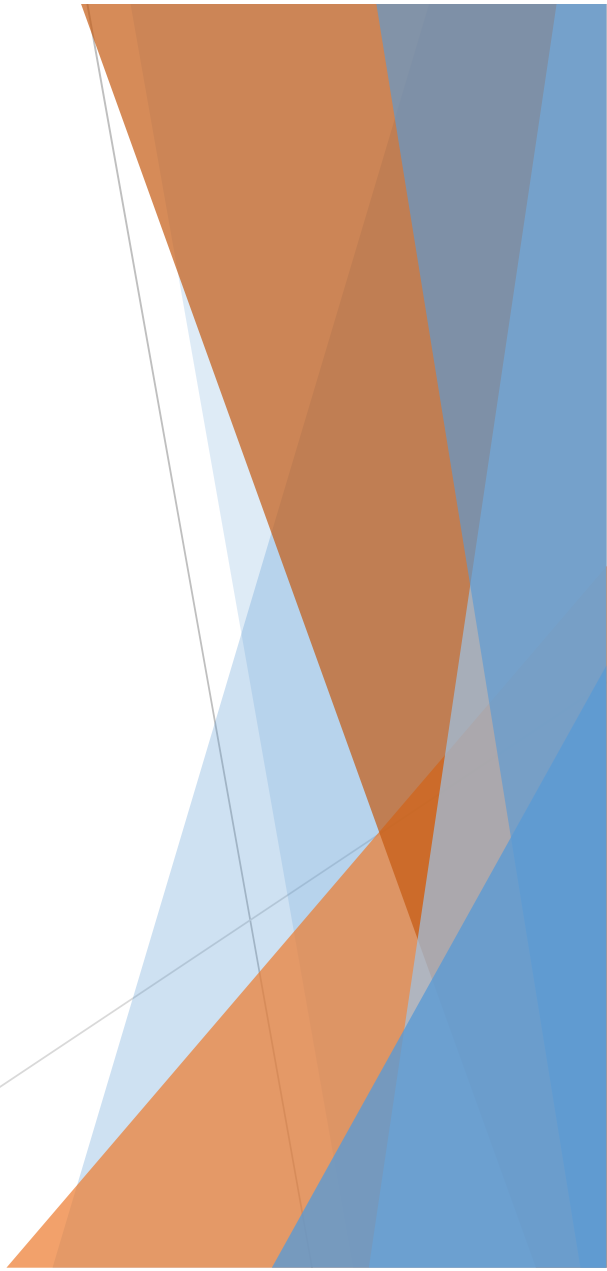
PRÁCTICA 1: EFICIENCIA versión individual

- Ahmed El Moukhtari Koubaa
- Damián Marín Fernández
- Eduardo Segura Richart
- Jesús Martín Zorrilla

Índice



1. Descripción del problema

- Realizar el “análisis” de varios algoritmos.
- Eficiencia empírica: llevar a cabo varias ejecuciones de los algoritmos con diferentes tamaños de entrada.
- Sobre las eficiencias empíricas realizar gráficas comparativas.
- Calcular la eficiencia híbrida, la cual se nos indica realizar con alguna de las herramientas las que disponemos.
- Tener en cuenta los efectos externos.

- **O(n^2)**
  - Burbuja
  - Selección
  - Inserción
- **O(n*log(n))**
  - Heapsort
  - Mergesort
  - Quicksort
- **Los más lentos**
  - Floyd O(n^3)
  - Hanói O(2^n)

2.Implementación del código.

Todo el código usado viene adjuntado en el guion de la práctica.

Se han realizado muy pocos cambios, por comodidad e igualdad.

- Lectura de datos como argumentos por línea de comandos
- Uso de high_resolution_clock en todos.

Estructura de los algoritmos

```
Int main(int argc, char ** argv){
    // Comprobación parámetros, creación de vectores.
    high_resolution_clock::time_point tantes, tdespues;
    duration<double> transcurrido;
    tantes = high_resolution_clock::now();
    // Ejecución del algoritmo
    tdespues = high_resolution_clock::now();
    transcurrido = duration_cast<duration<double>>(tdespues - tantes);
    // mostrar resultados (tamaño, duración)
}
```

Estructura del script

```
INICIO; FINAL; INCRE;
rm algoritmo.dat
for ((tam=$INICIO; tam <= $FINAL; tam+=$INCRE))
do
    ./algortimo $tam >> algoritmo.dat
done
```

3. Mediciones empíricas.

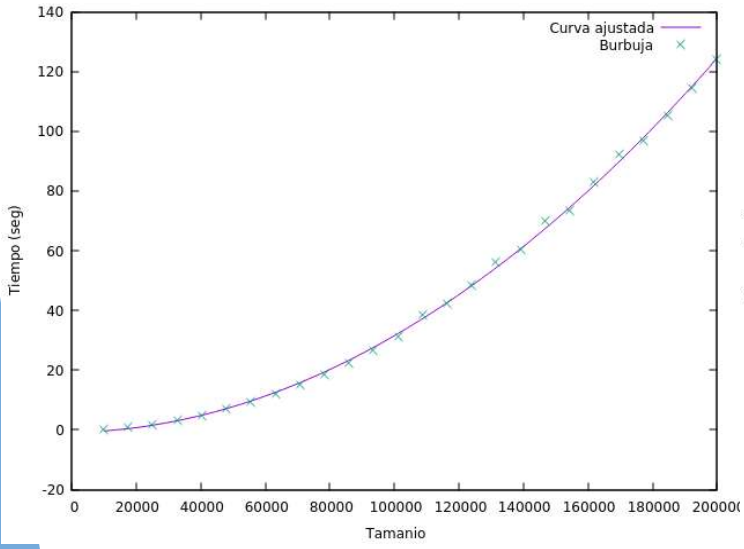
Tamaños	Burbuja	Selección	Inserción	Mergesort	Heapsort	Quicksort
10000	0.255053	0.117256	0.0938404	0.00155668	0.00151286	0.0010819
17600	0.864844	0.359074	0.290766	0.002891	0.00273124	0.00209875
25200	1.8235	0.737175	0.599571	0.004963	0.00400816	0.00305498
32800	3.15053	1.24206	1.01231	0.005487	0.00539753	0.0042423
40400	4.83774	1.88588	1.5128	0.007352	0.00670754	0.00497331
48000	7.03347	2.66334	2.14481	0.009391	0.00810732	0.00609304
55600	9.56826	3.57109	2.88994	0.009507	0.00955758	0.00710712
63200	11.9939	4.61608	3.70183	0.011131	0.0110666	0.00814578
70800	15.1629	5.78523	4.6777	0.013099	0.0123721	0.00908153
78400	18.6327	7.09463	5.73385	0.014929	0.0137789	0.0102033
86000	22.5184	8.53547	6.94295	0.017229	0.0155898	0.011261
93600	26.7134	10.1179	8.2323	0.019244	0.017011	0.0127163
101200	31.315	11.8065	9.62902	0.021434	0.0184207	0.0136462
108800	38.3838	13.6546	11.16	0.019412	0.0200903	0.014416
116400	42.2404	15.6192	13.4192	0.021278	0.0217059	0.0155649
124000	48.3924	17.7472	15.8072	0.023359	0.0231133	0.0167578
131600	56.3024	19.9885	17.5316	0.024953	0.0245736	0.0178021
139200	60.4578	22.3559	19.7099	0.02679	0.0277268	0.0187858
146800	69.8402	24.8439	21.6786	0.028996	0.0281662	0.0203
154400	73.3324	27.5282	24.6134	0.030815	0.0294643	0.0211196
162000	83.1672	30.2532	26.6298	0.033063	0.0309397	0.0220494
169600	92.124	35.2392	27.8852	0.035161	0.0329503	0.0232497
177200	96.8594	36.2317	31.4581	0.037214	0.0344862	0.0248
184800	105.273	39.4081	35.0845	0.039439	0.0359864	0.0257361
192400	114.52	42.7032	34.6138	0.042133	0.0381247	0.0267367
200000	123.941	46.1832	37.6923	0.044077	0.0392406	0.0279626

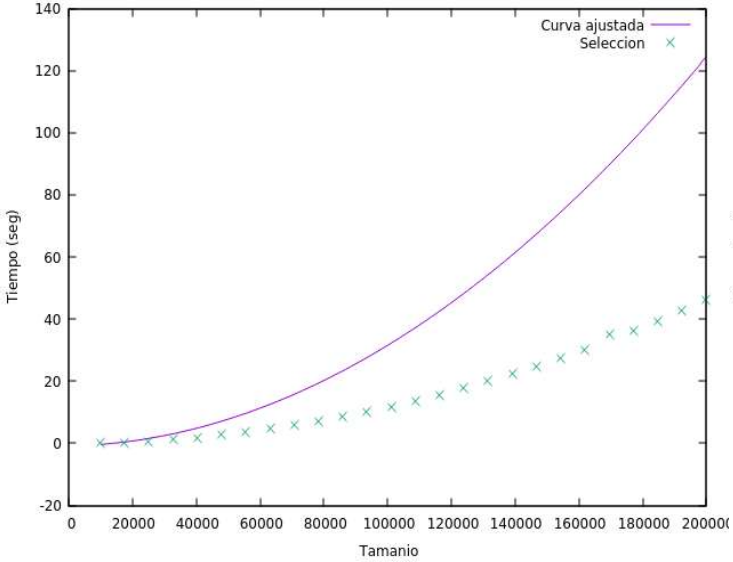
4. Gráficas y órdenes de eficiencia.

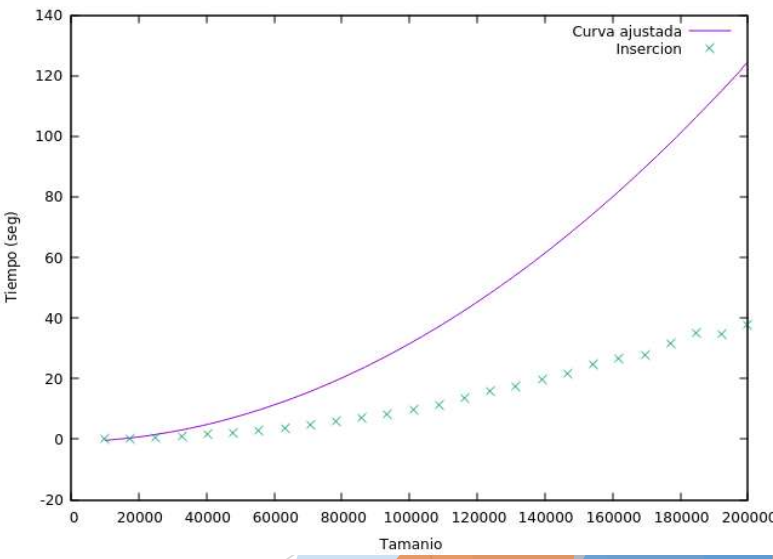
Ver una tabla con muchos números por todas partes no tiene ningún sentido y es difícil de interpretar, por eso procesamos estos datos y los representamos de manera gráfica.

Algoritmos de ordenación básicos

a0*x*x+a1*x+a2







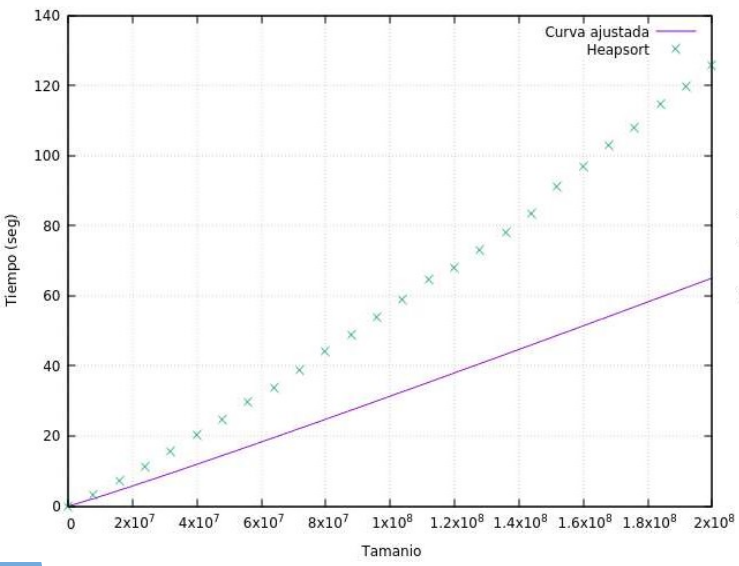
Burbuja

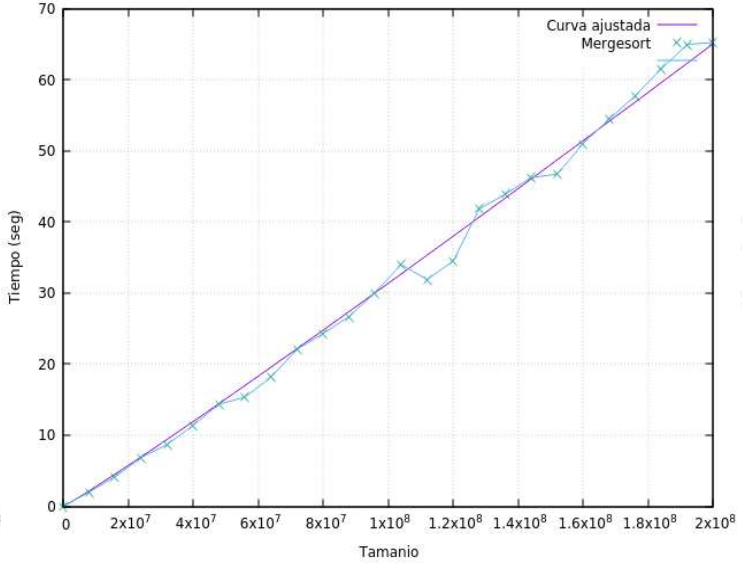
Selección

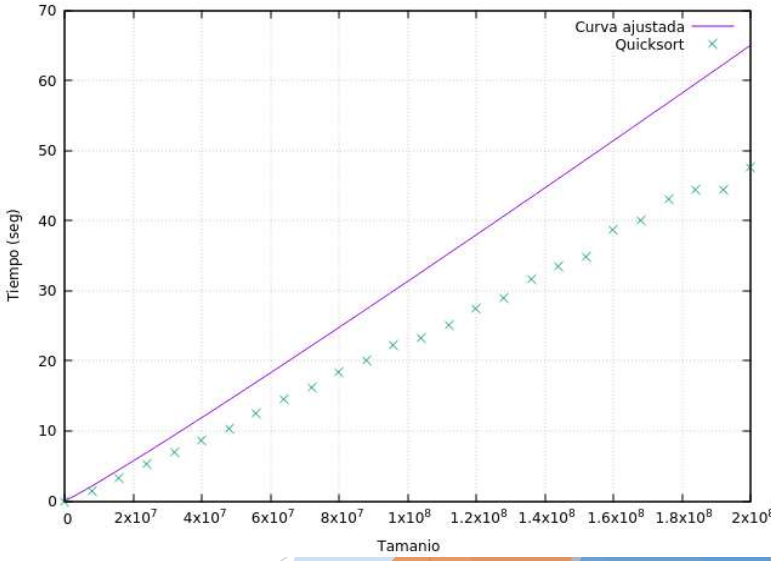
Inserción

Algoritmos de ordenación rápidos

a0*n*log(n) + a1





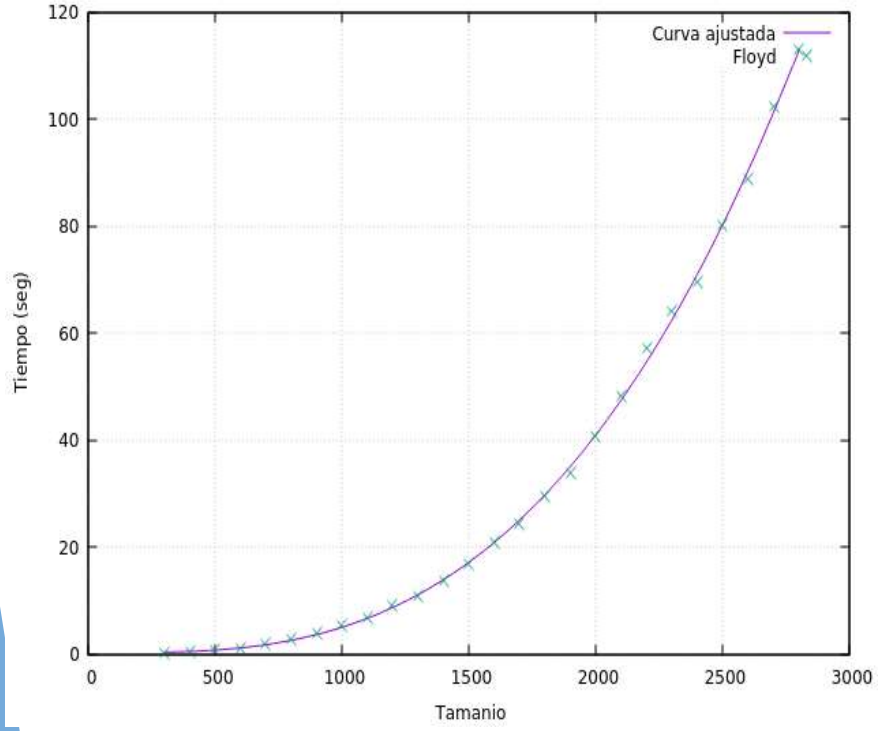


Heapsort

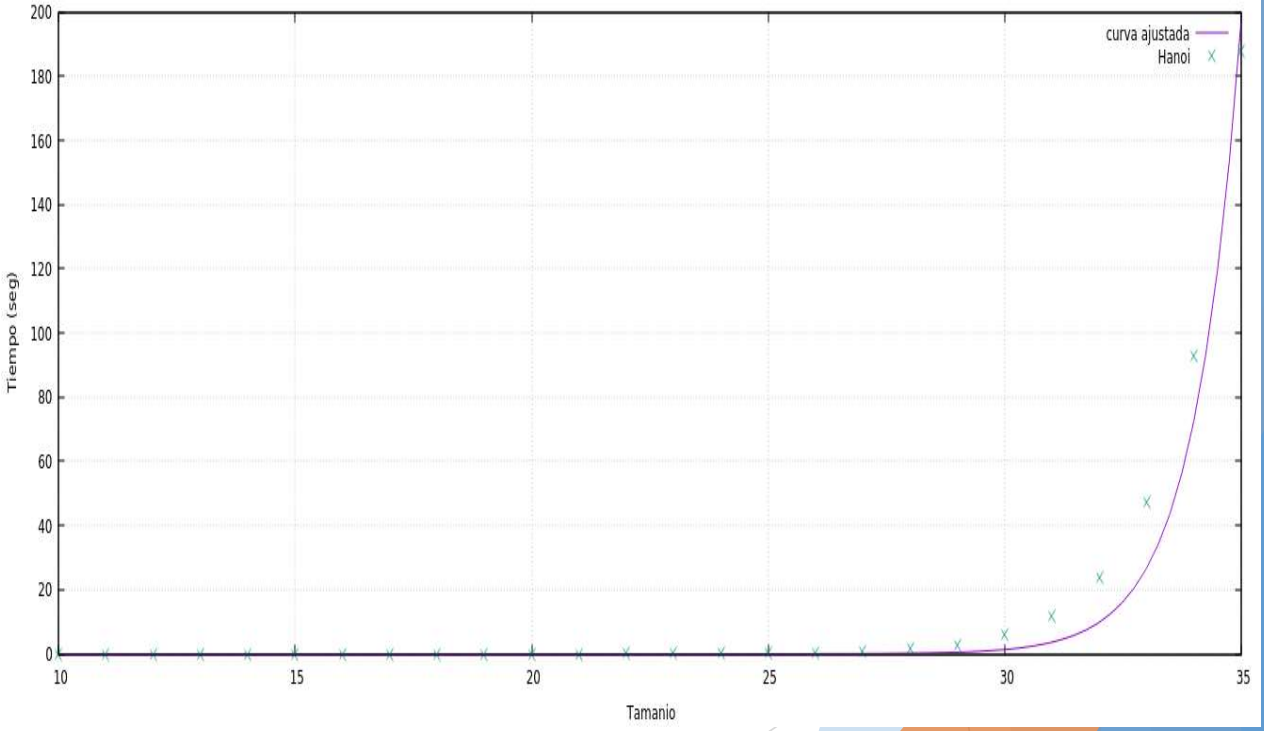
Mergesort

Quicksort

Algoritmos más lentos

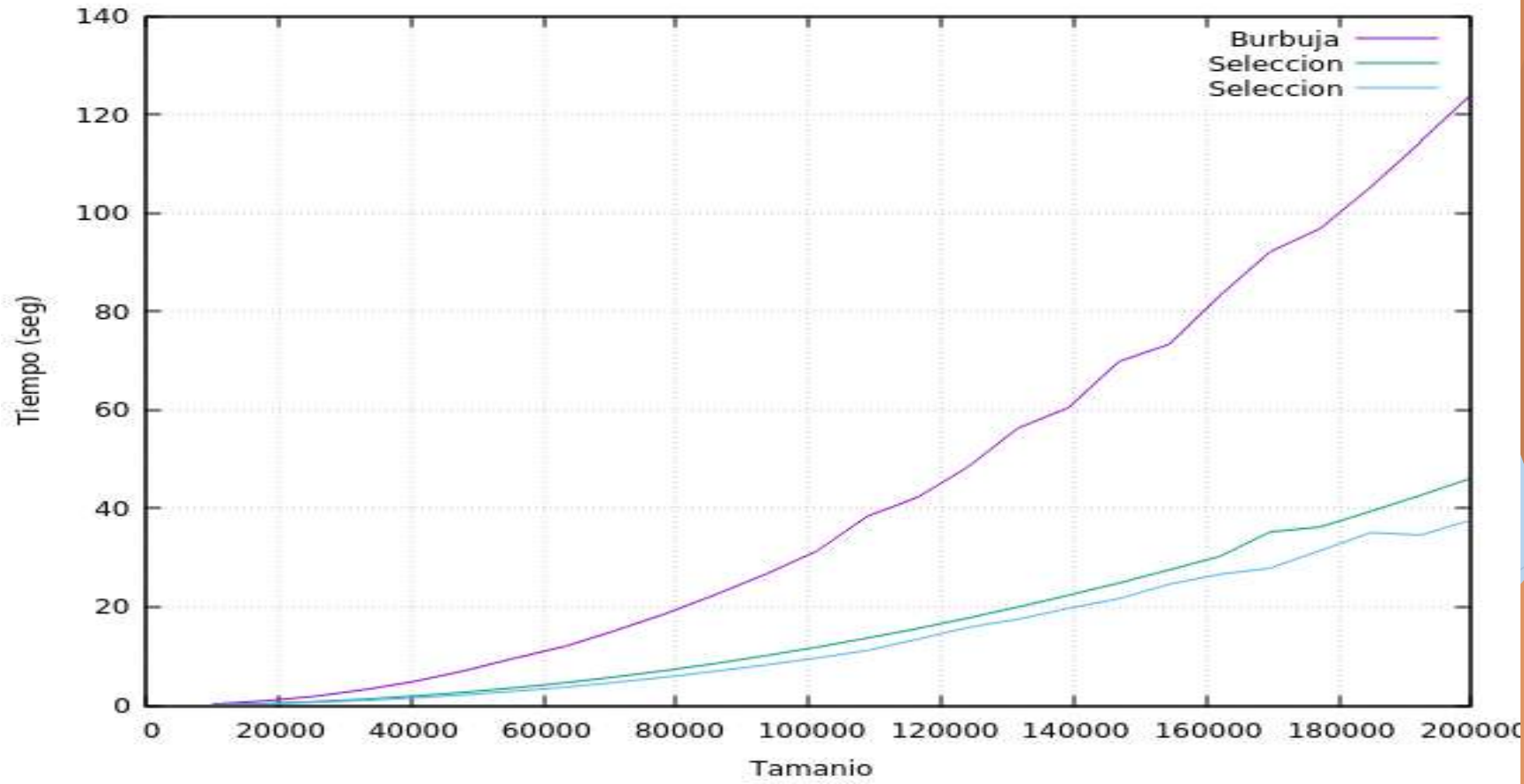


Floyd
a0*x*x*x+a1*x*x+a2*x+a3

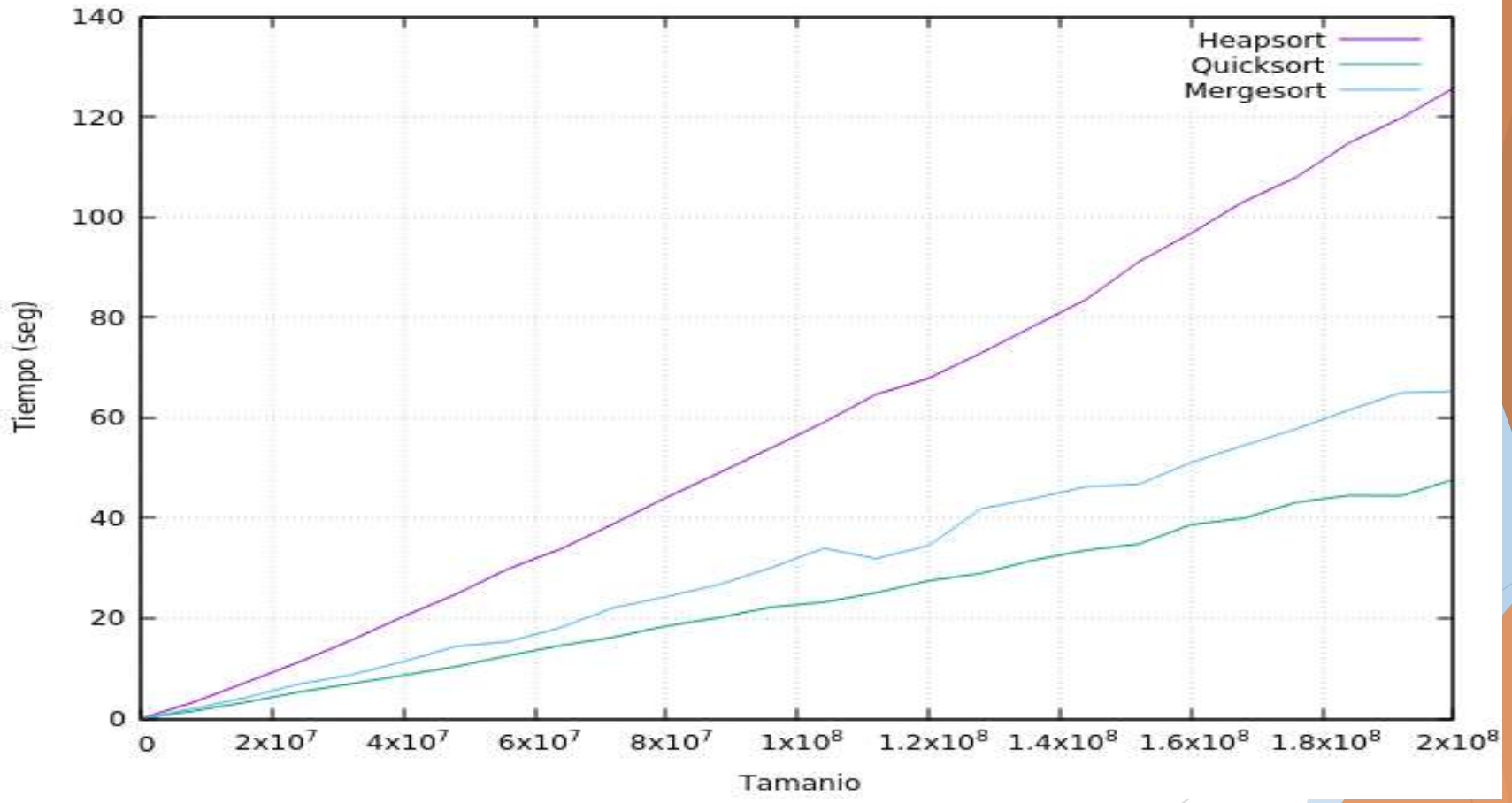


Hanói
a0*2exp(x)

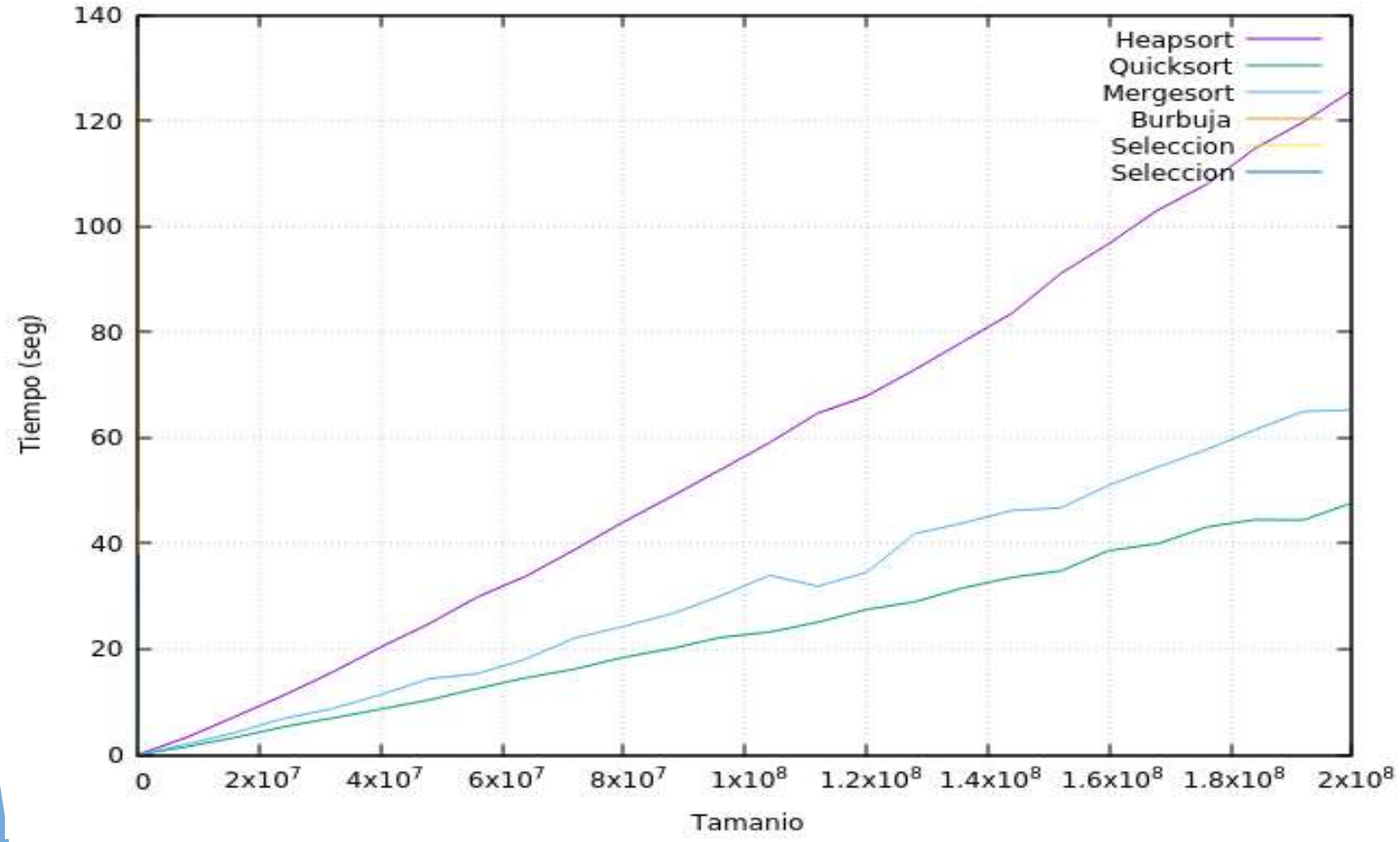
Ordenación lentos



Ordenación rápidos

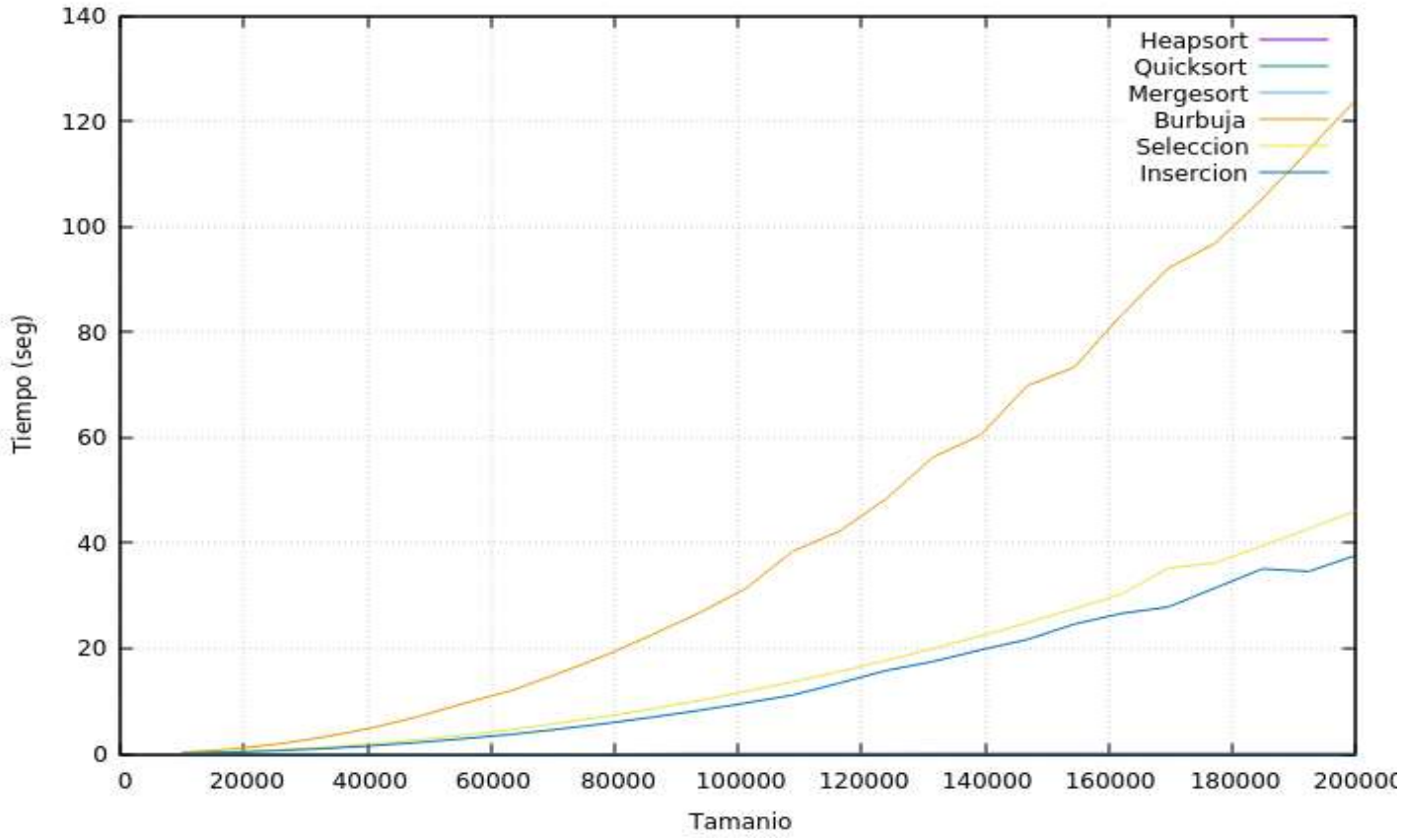


Comparación rápidos con lentos



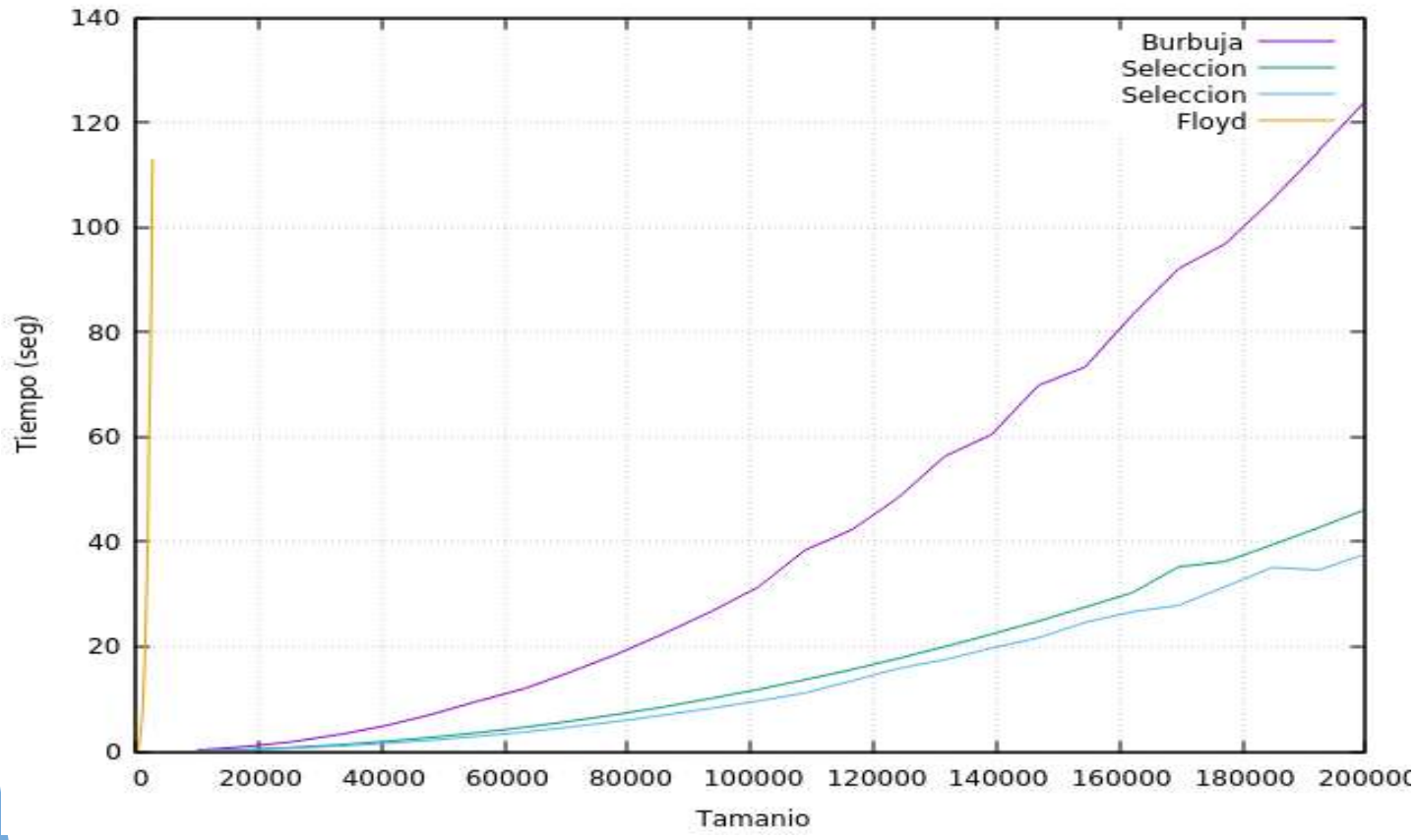
En esta podemos ver que en lo que los básicos ordenan 200 mil elementos los rápidos ordenan 200 millones.

Comparación lentos con rápidos



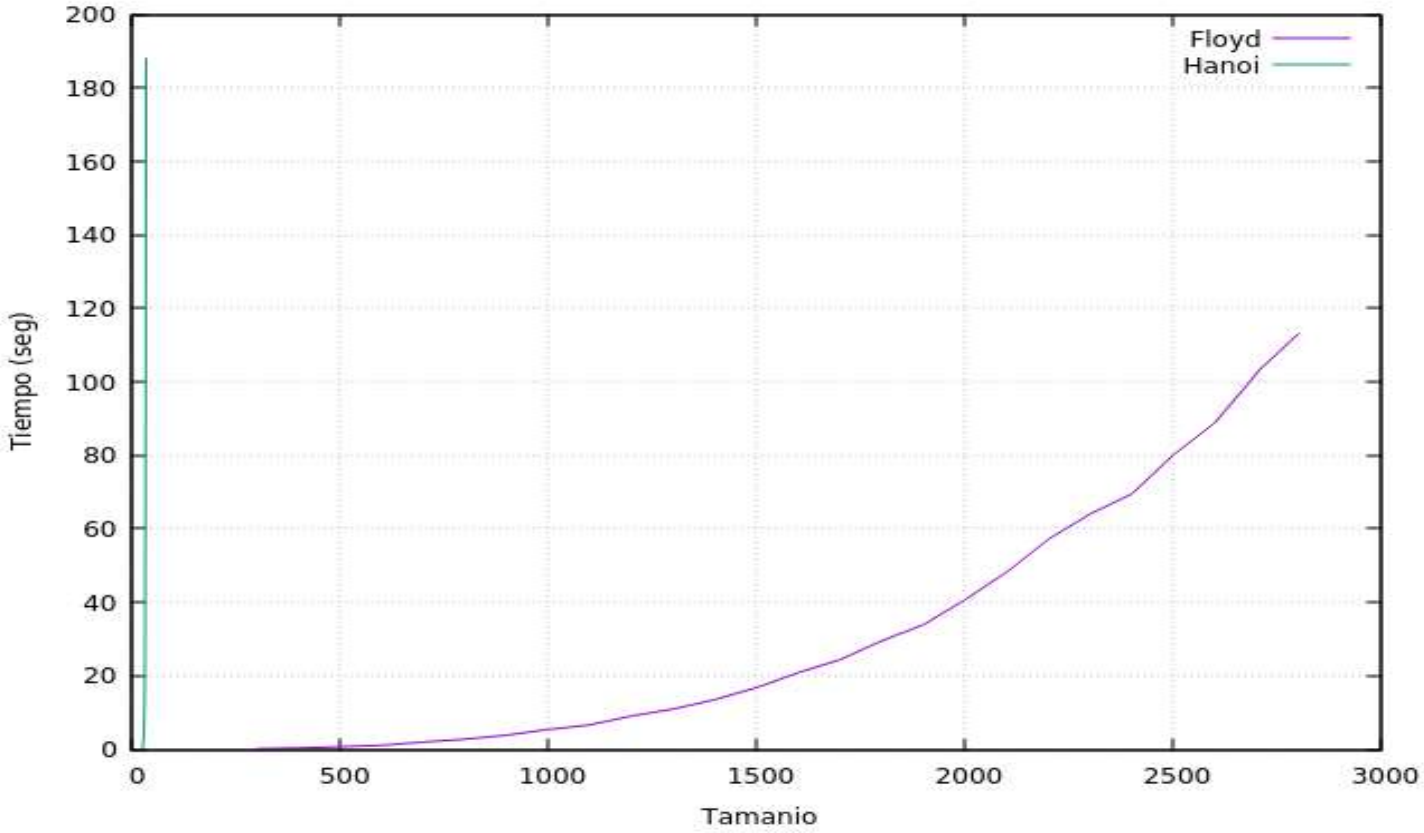
Para el mismo tamaño vemos que no hay punto de comparación entre los rápidos y los lentos, pues los rápidos ni se aprecian.

Comparación cuadrático con cúbico



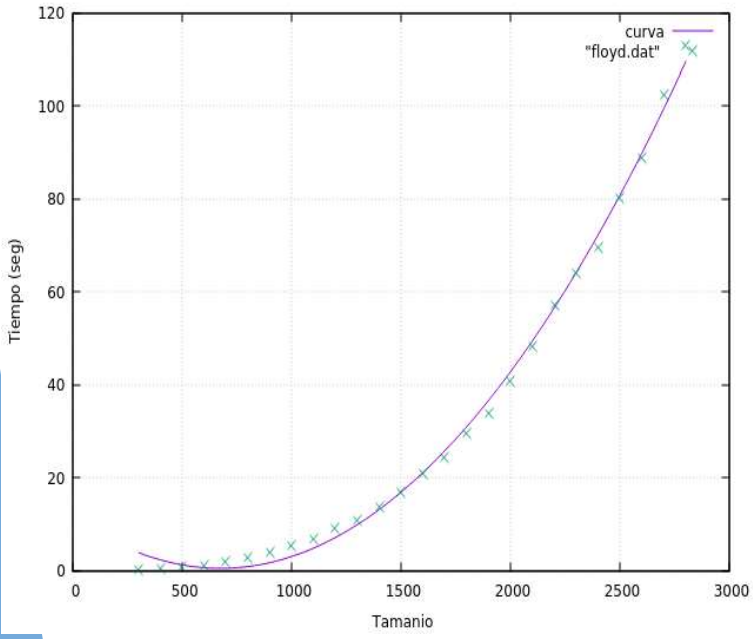
Los algoritmos cuadrados no son tan malos como parecen. Los cúbicos son horribles.

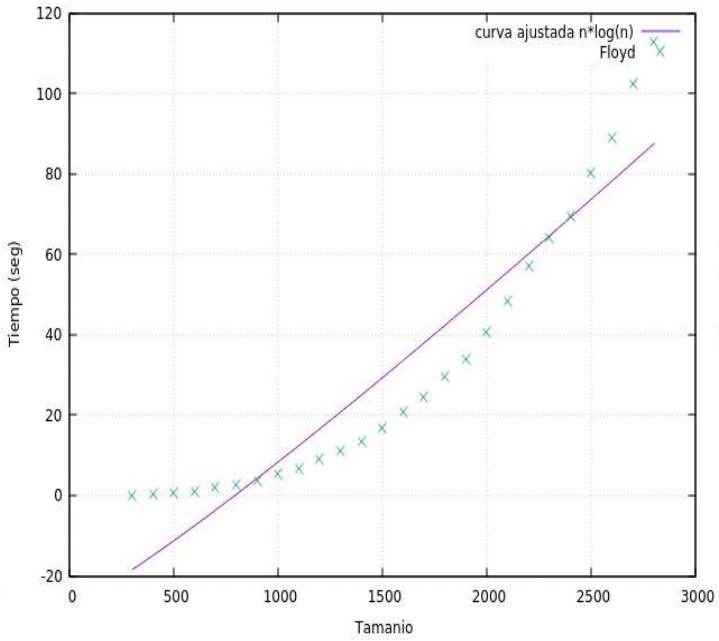
Comparación Cúbico con exponencial

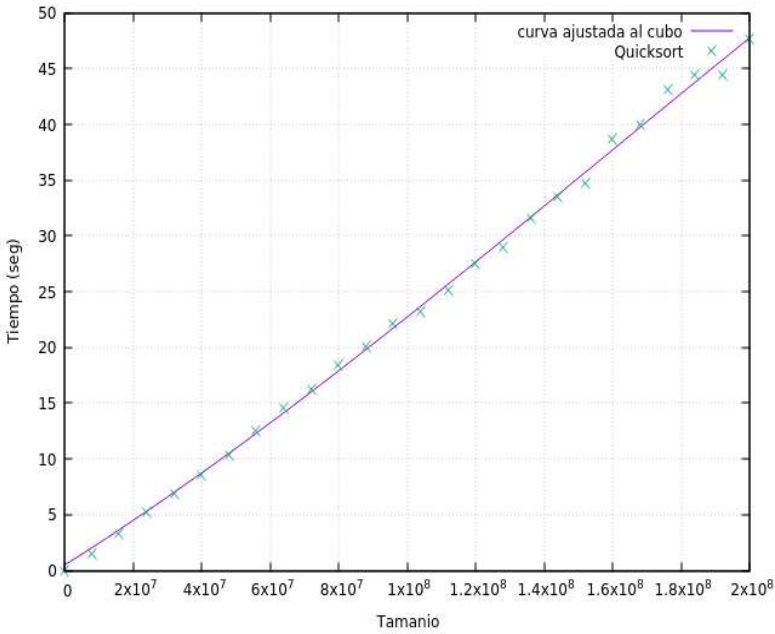


Los cúbicos no son tan malos, son polinomiales. Los exponenciales son horribles son NP ?

Ajustes de regresión diferentes







Floyd cuadrático

Floyd exponencial

Quicksort cúbico

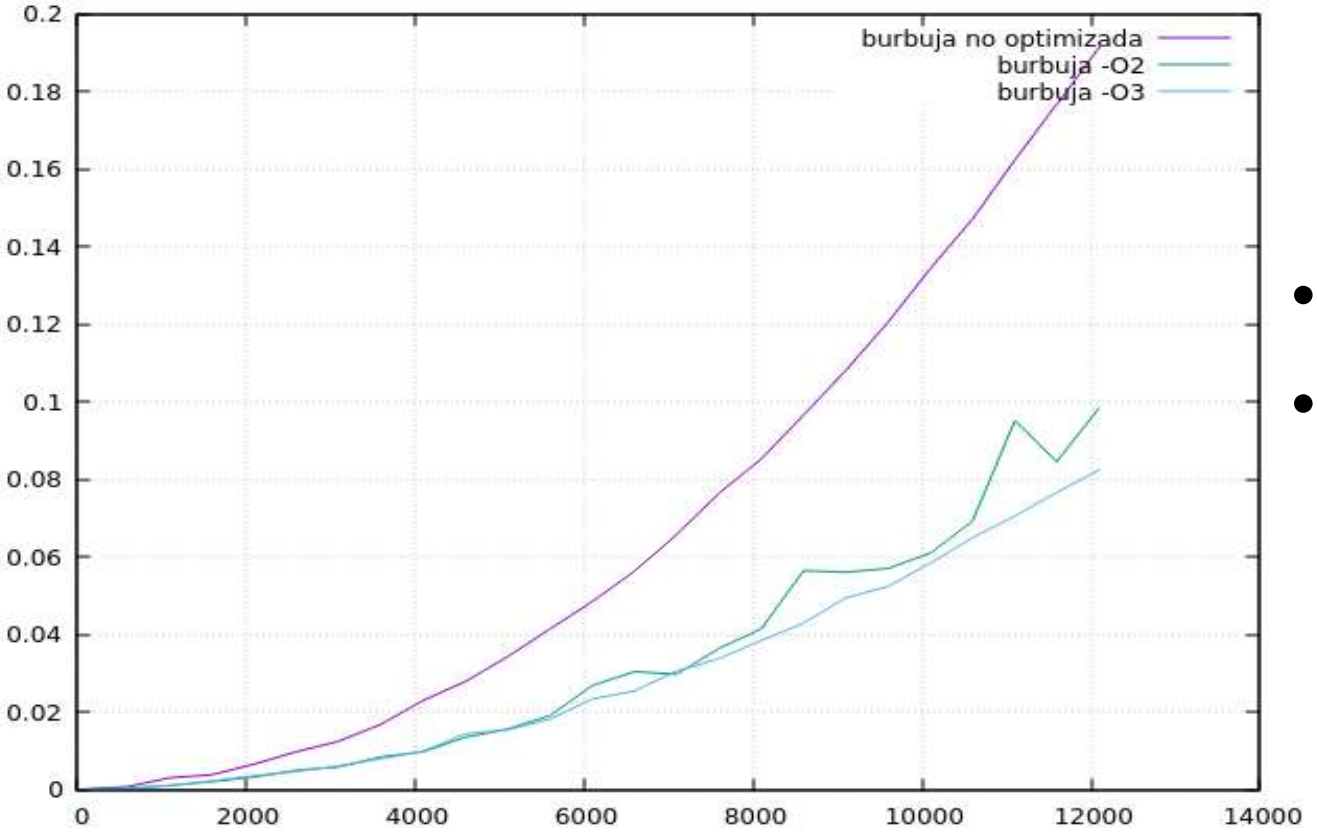
5. Comparaciones entre optimizar (o no).

Hay grandes diferencias entre usar la optimización o no hacerlo en todos los algoritmos aunque se nota más en unos que en otros.

Las opciones usadas al optimizar han sido dos

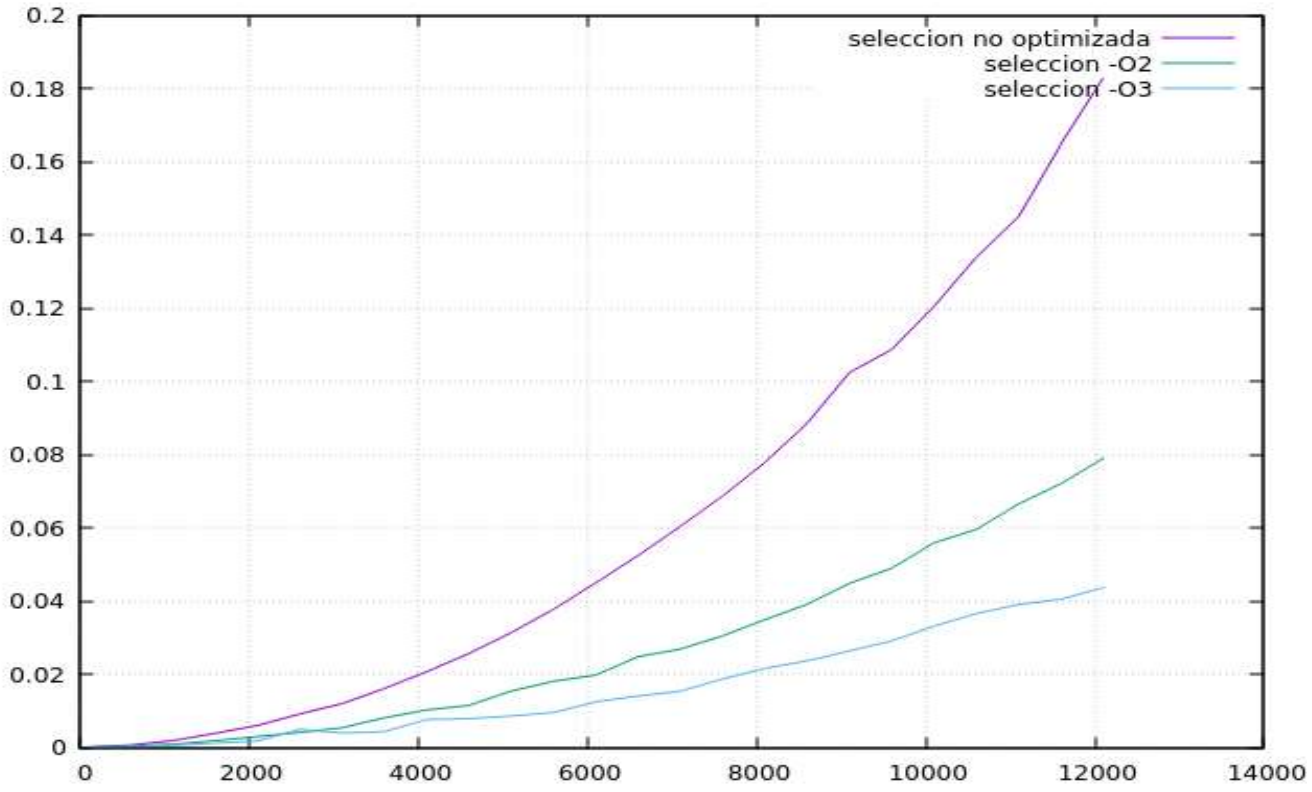
- O2, una optimización notable pero moderada
- O3, la más agresiva de las optimizaciones en g++

BURBUJA



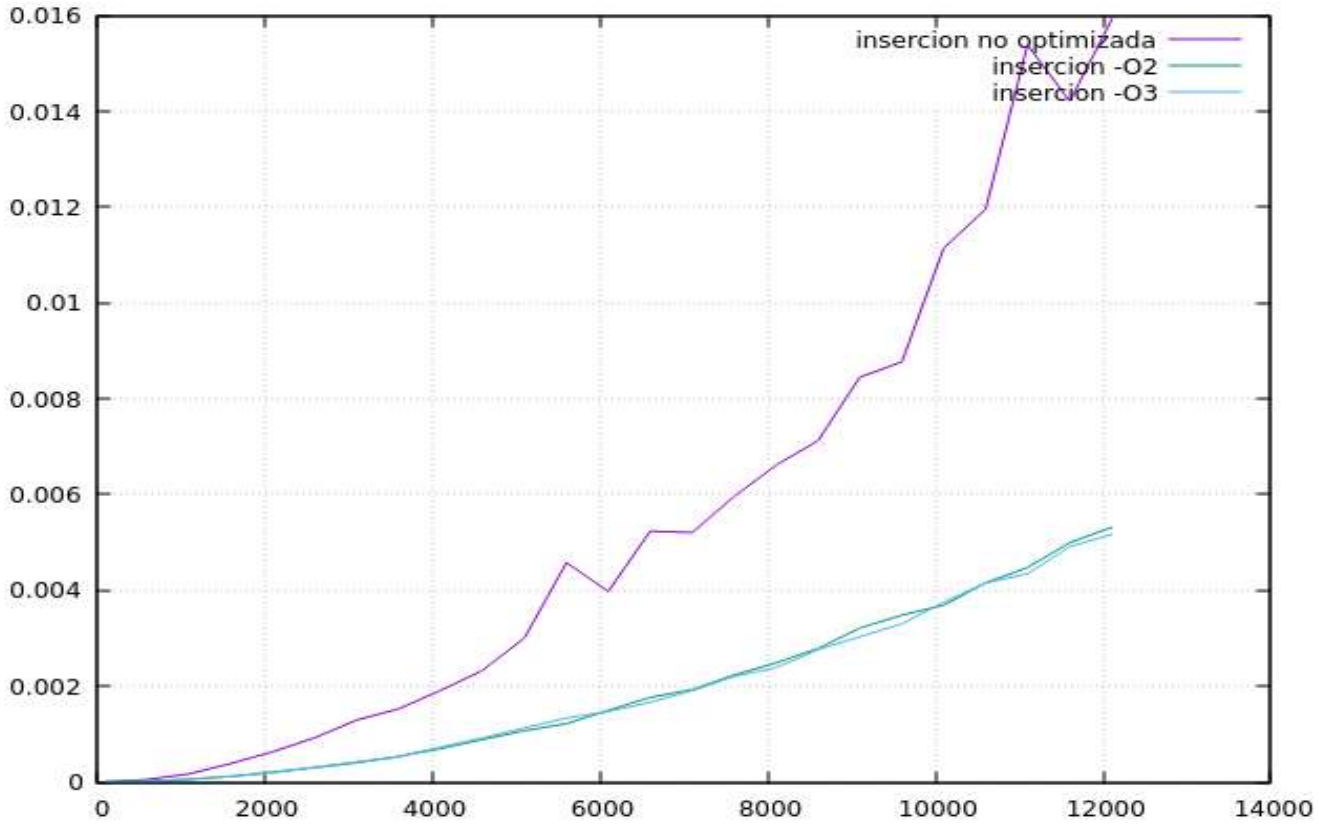
- O2 mejora del 195%
- O3 mejora del 232%

SELECCIÓN



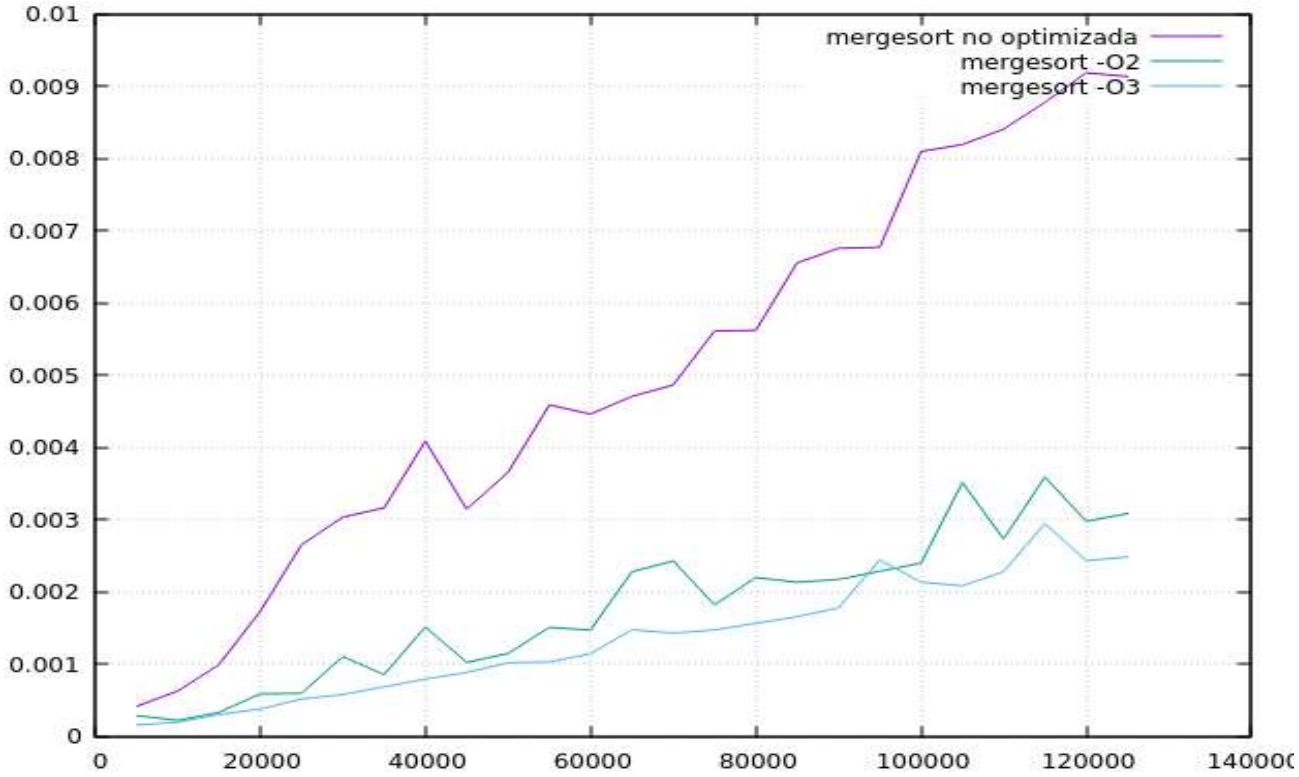
- O2 mejora del 231%
- O3 mejora del 418%

INSERCIÓN



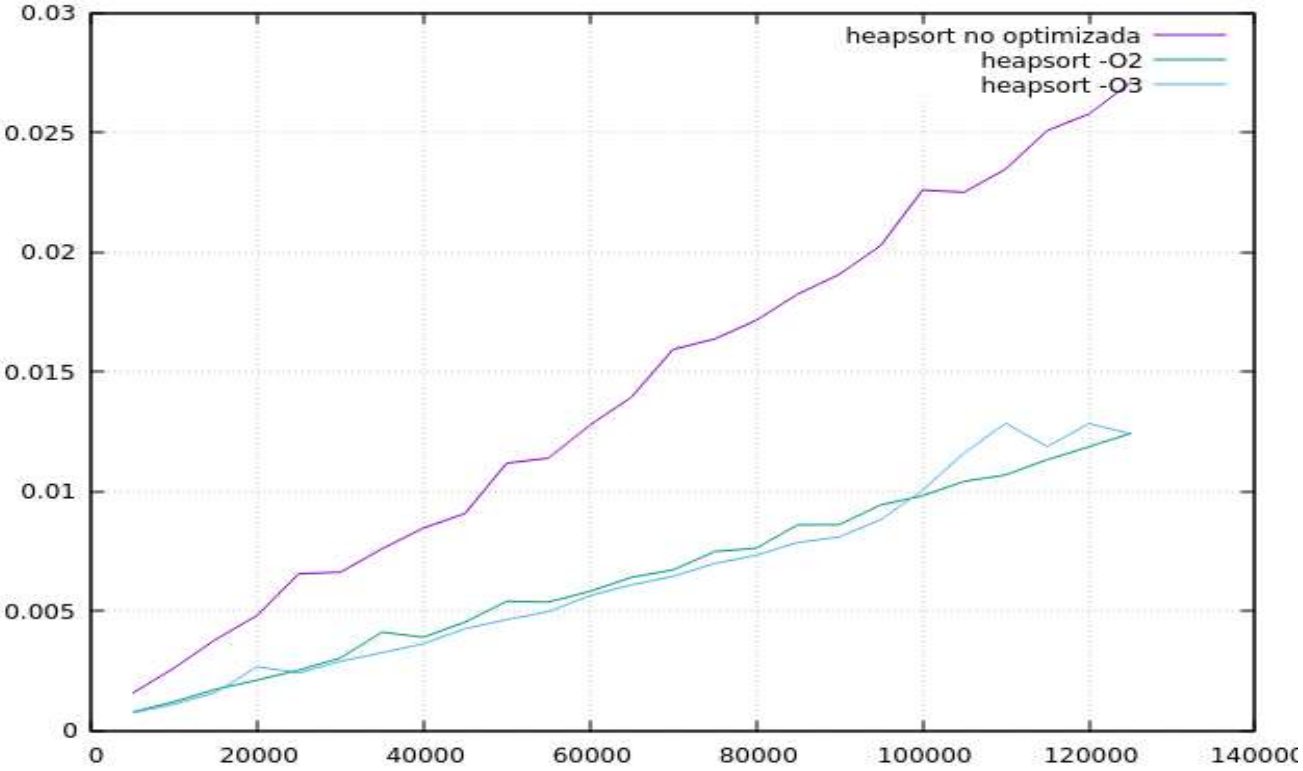
- O2 mejora del 299%
- O3 mejora del 308%

MERGESORT



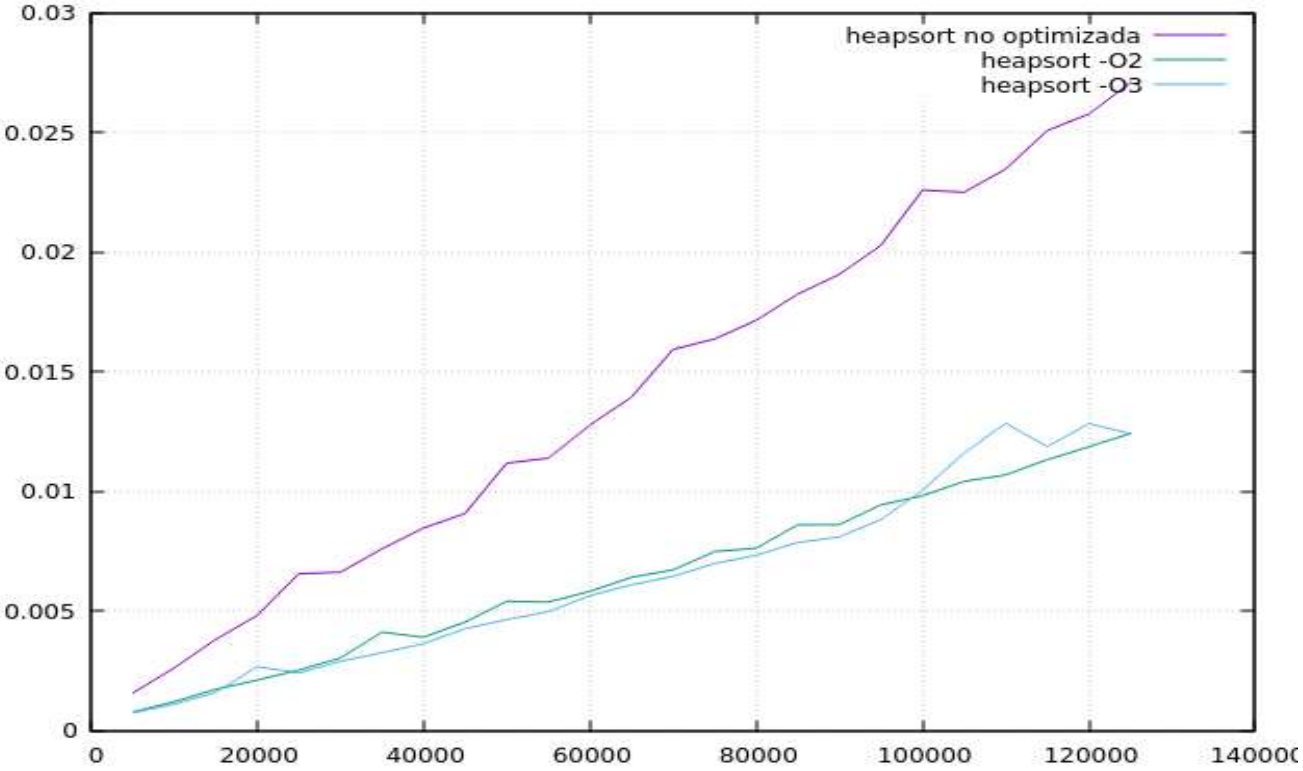
- O2 mejora del 296%
- O3 mejora del 368%

MERGESORT



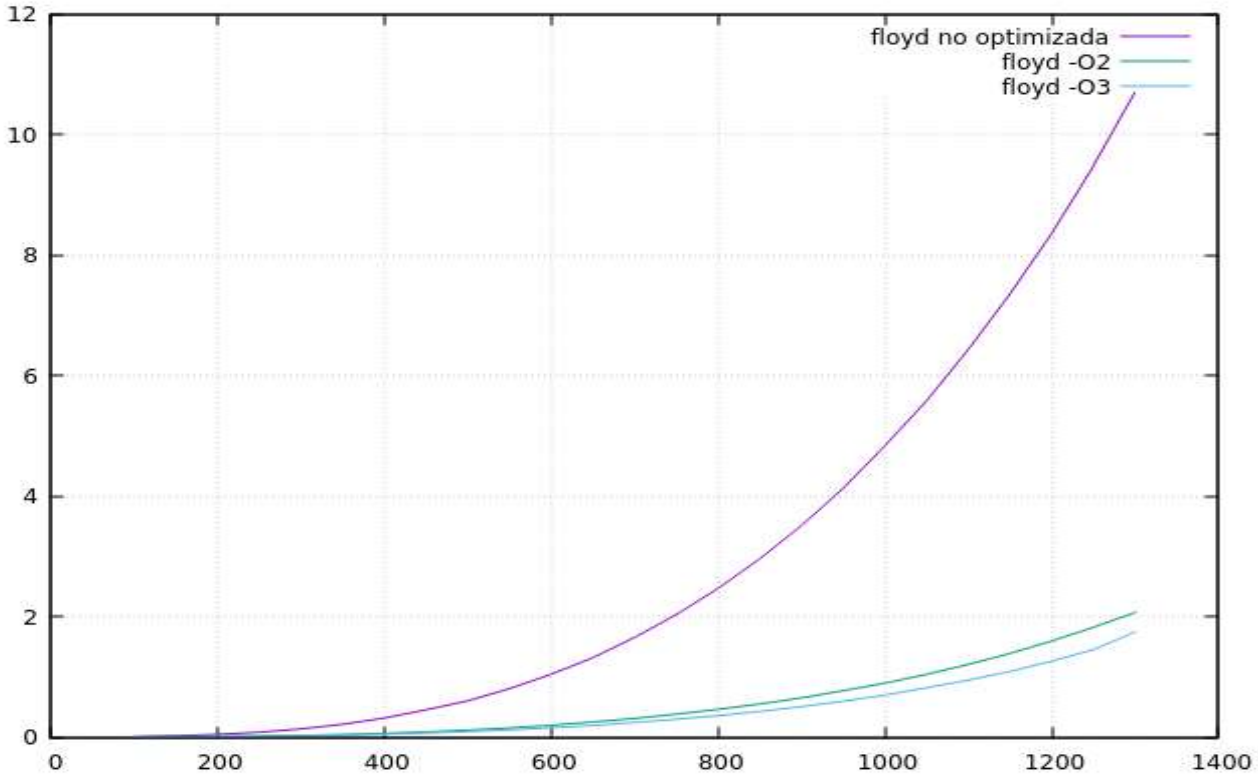
- O2 mejora del 218%
- O3 mejora del 219%

QUICKSORT



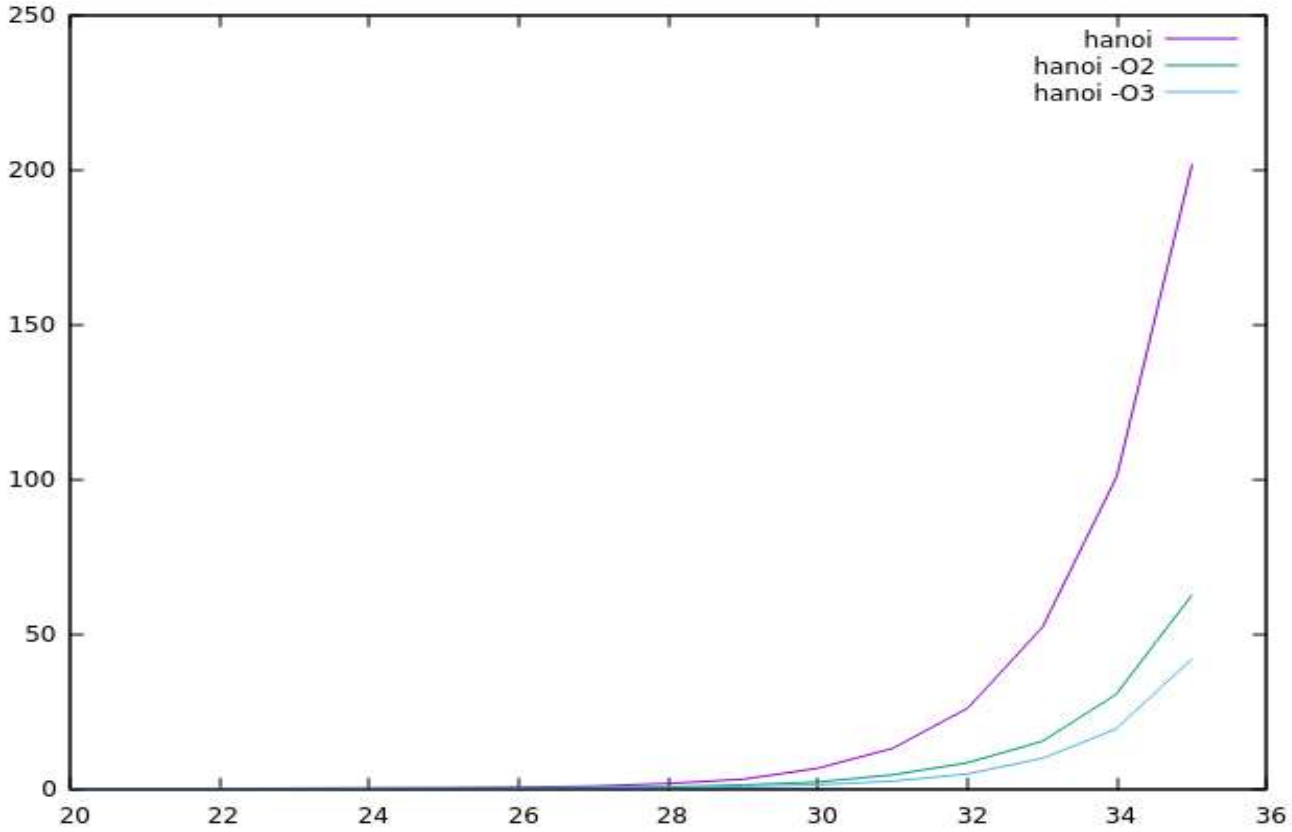
- O2 mejora del 322%
- O3 mejora del 323%

FLOYD



- O2 mejora del 517%
- O3 mejora del 613%

HANOI



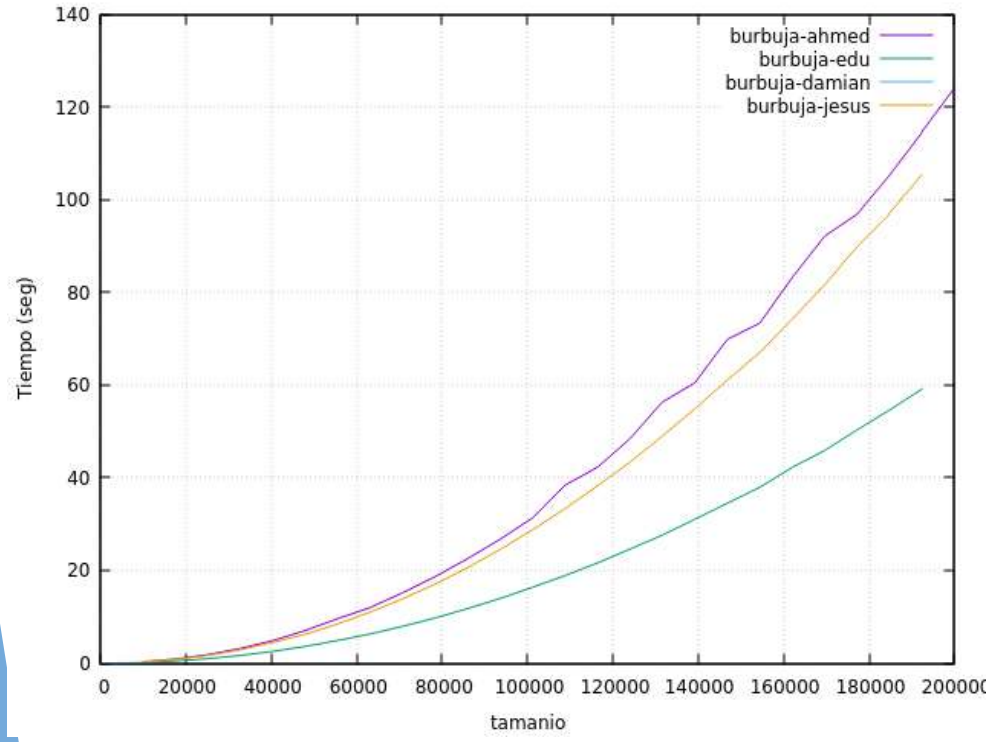
- O2 mejora del 323%
- O3 mejora del 418%

6. Comparación entre hardware

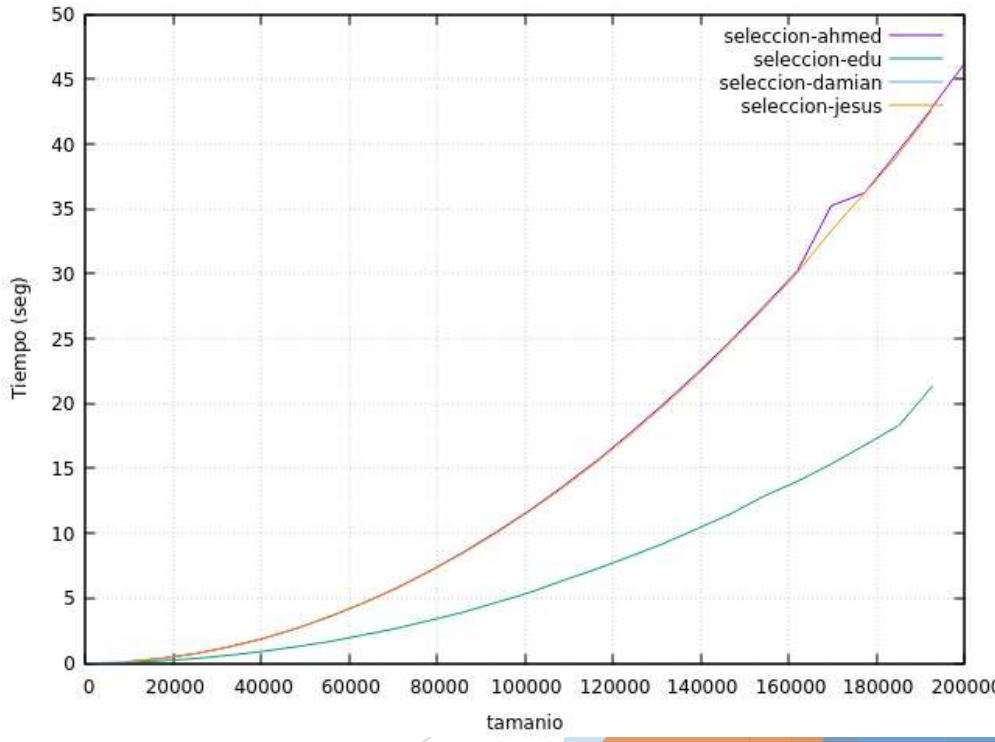
Comparaciones realizadas en 4 equipos diferentes uno de cada integrante del grupo, nos referiremos a cada equipo por el nombre de su dueño.

- Ahmed → I7 6700HQ 3.2 Ghz | 16 GB RAM
- Damián → I7 6700U 3.2 Ghz | 8 GB RAM
- Eduardo → I7 6500U 2.5 Ghz | 12 GB RAM
- Jesús → I7 7500U 2.6 Ghz | 8 GB RAM

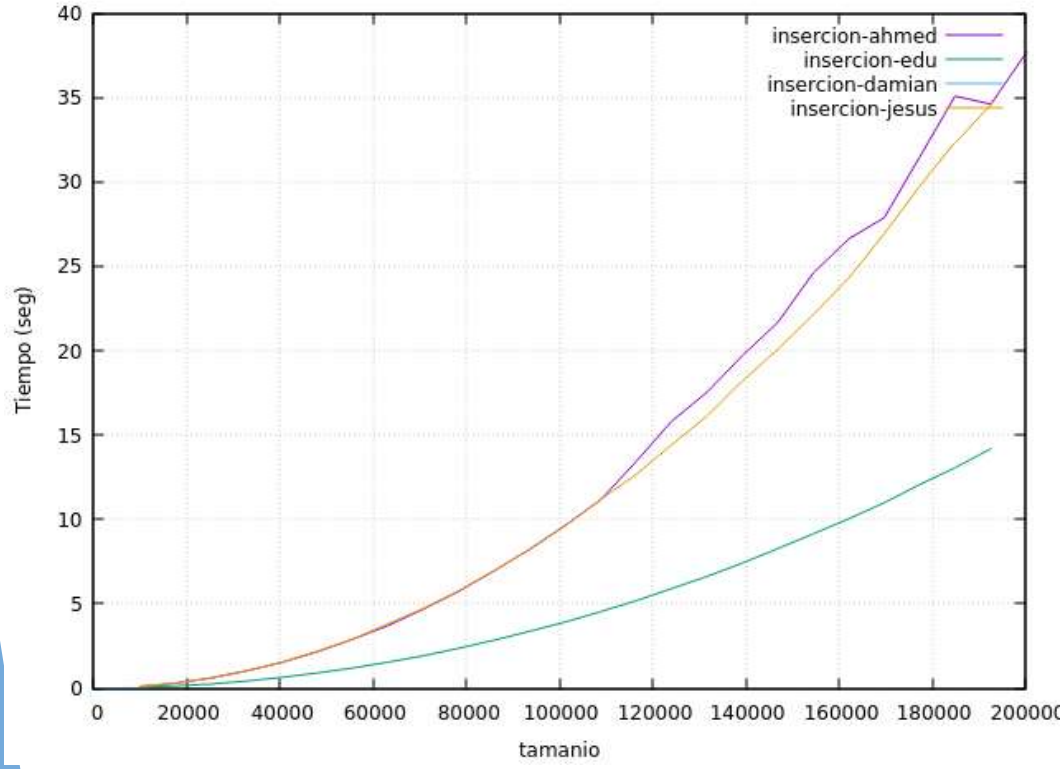
BURBUJA



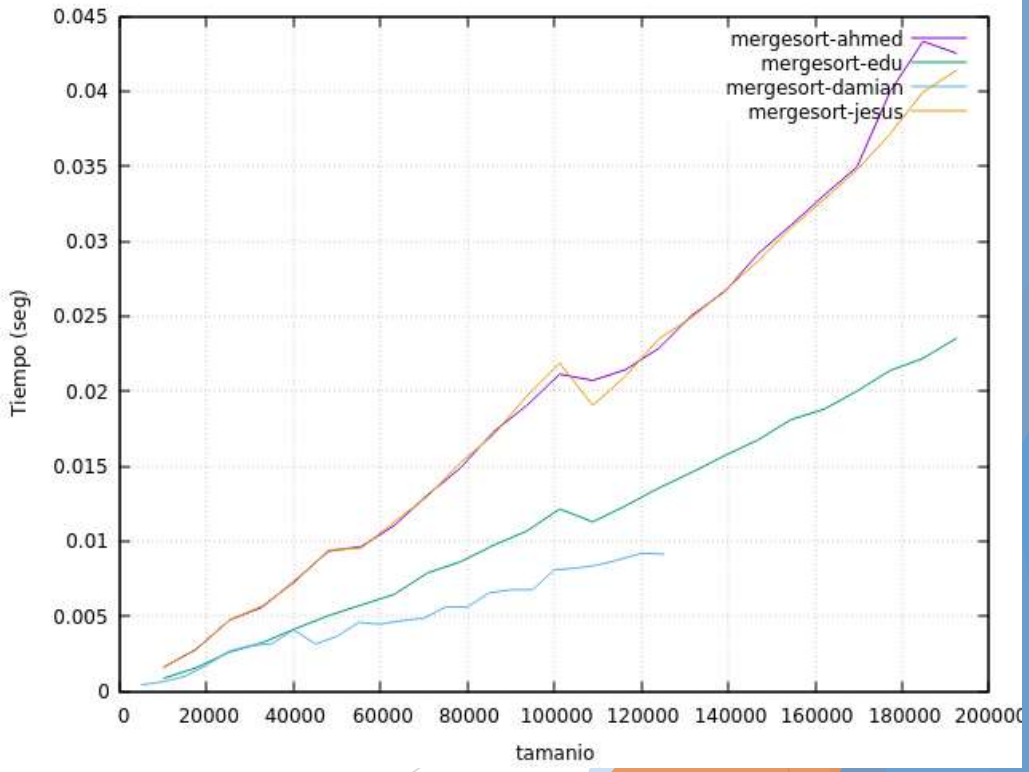
SELECCIÓN



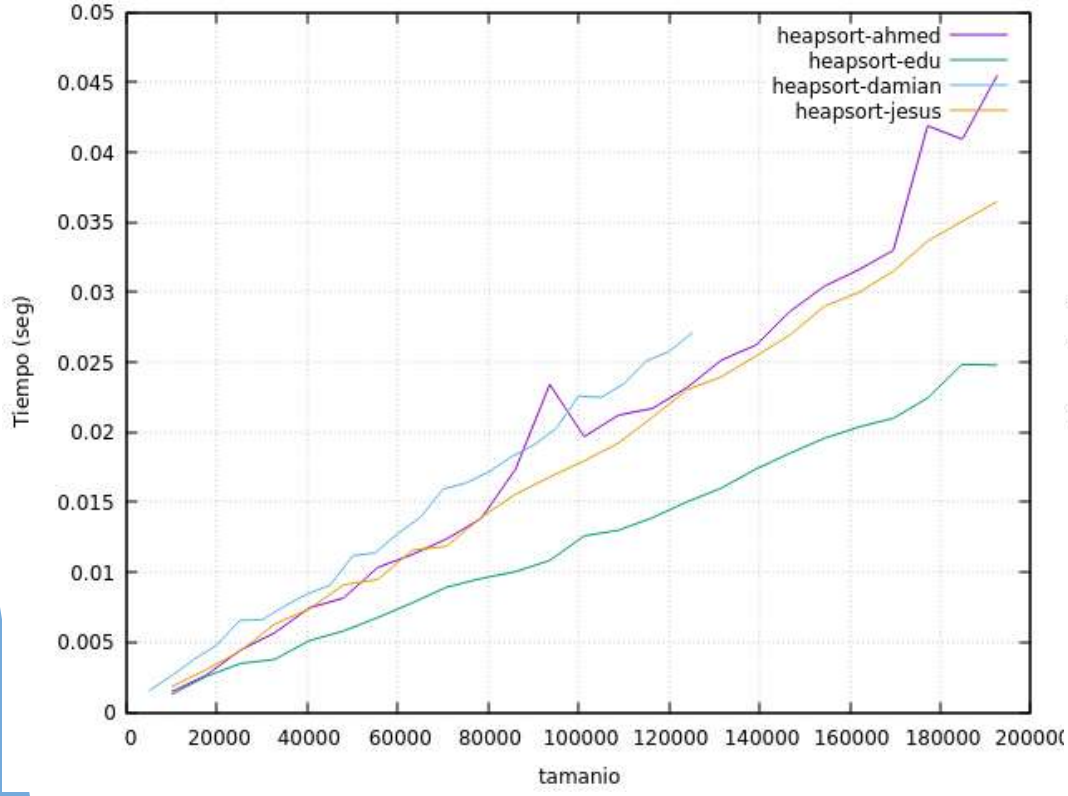
INSERCIÓN



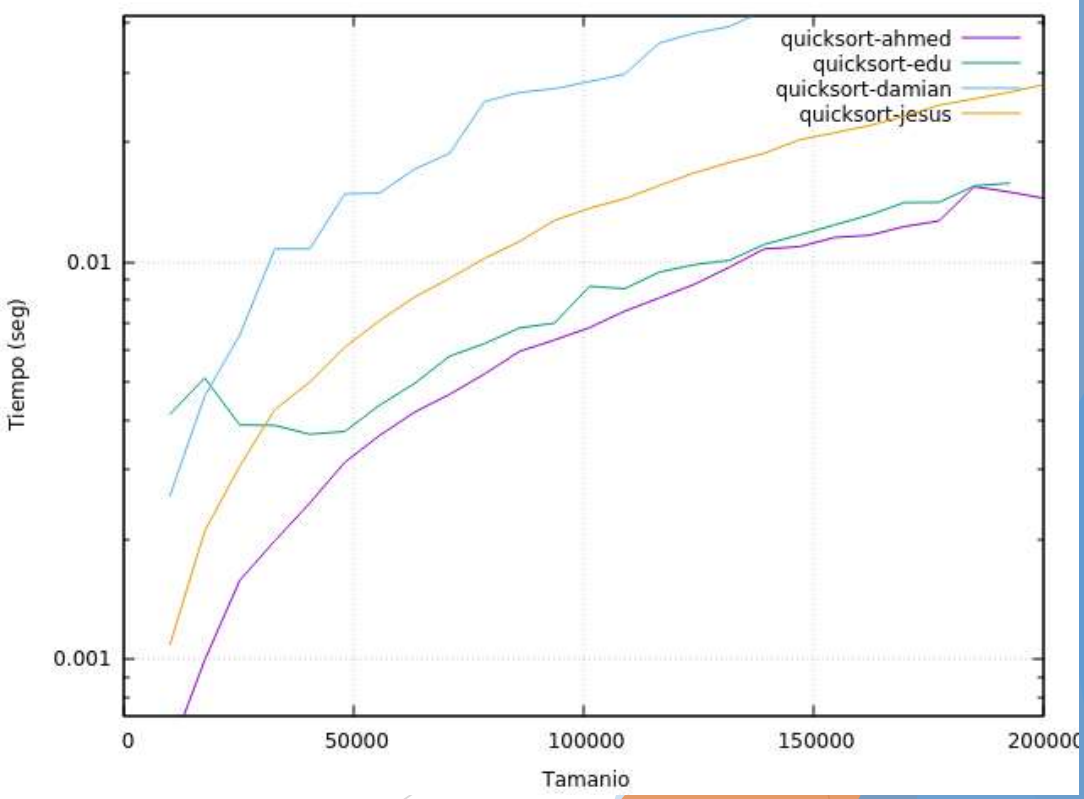
MERGESORT



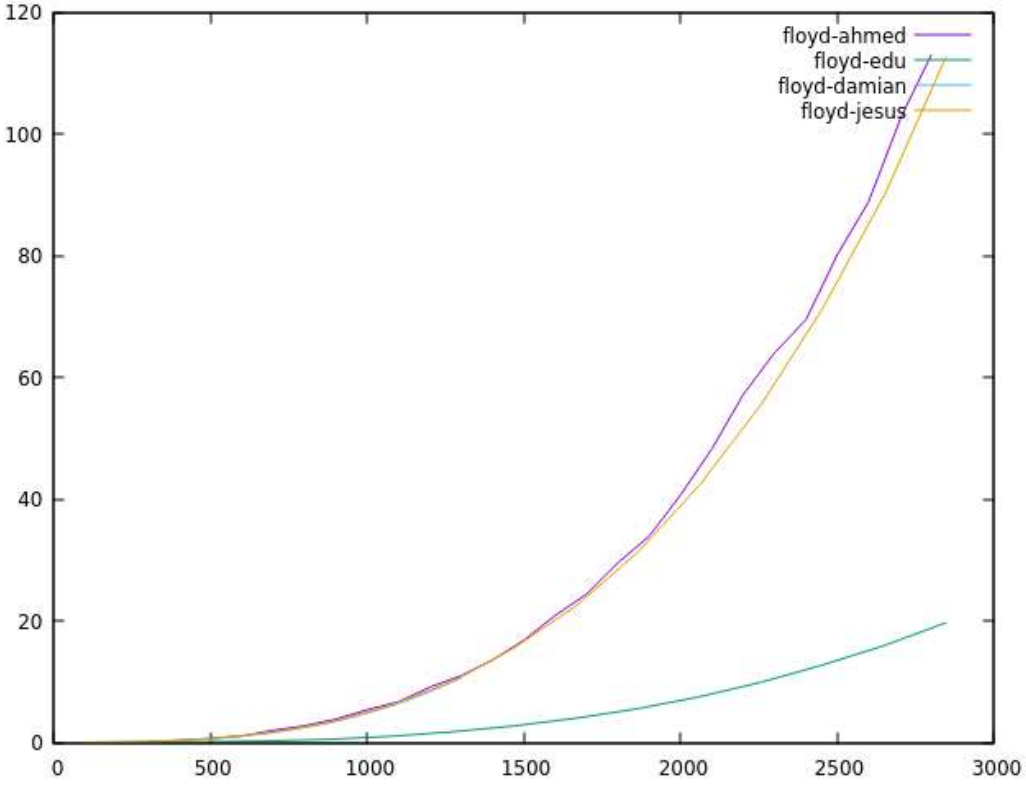
HEAPSORT



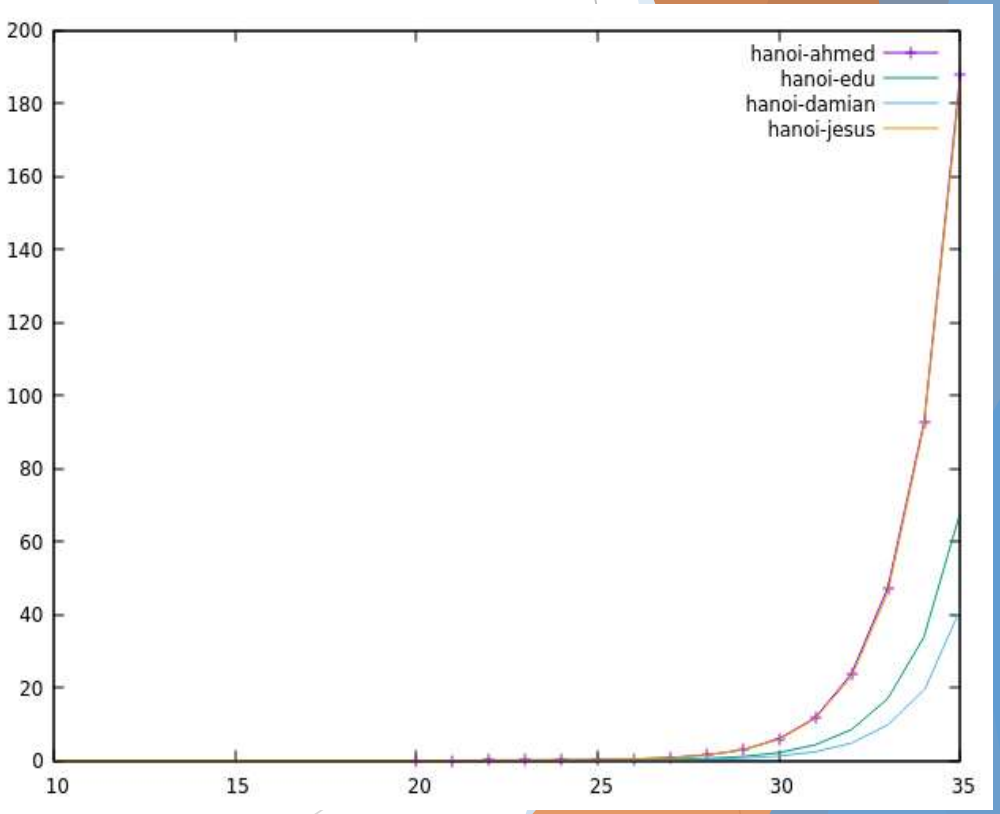
QUICKSORT



FLOYD


floyd-ahmed
floyd-edu
floyd-damian
floyd-jesus
120
100
80
60
40
20
0
0
500
1000
1500
2000
2500
3000

HANOI


hanoi-ahmed
hanoi-edu
hanoi-damian
hanoi-jesus
200
180
160
140
120
100
80
60
40
20
0
10
15
20
25
30
35

7. Comparación entre Sistemas Operativos

Comparación en dos distribuciones Linux

- UBUNTU 18.04 LTS
- LINUX MINT O MINT

En todos los cálculos hechos Mint es entre un 3% y un 8% más lento que Ubuntu y apenas hay diferencias.

Comparando Ubuntu con Windows 10 tampoco hay grandes diferencias, aunque en este caso Windows es ligeramente más veloz

8. Conclusiones

- El algoritmo más eficiente para ordenación es el quicksort
- El algoritmo de ordenación más lento es el algoritmo de burbuja
- Destaca mergesort por un gran consumo de memoria
- Los cúbicos son bastante más lentos que los cuadráticos
- Los cúbicos son bastante mejores que cualquier algoritmo exponencial
- El algoritmo más lento es el de Hanói, es exponencial no polinomial
- La optimización hace una gran diferencia pero no altera la curva del algoritmo
- El algoritmo que mas se beneficia de la optimización es Floyd hasta 6 veces más rápido.
- El algoritmo que menos se beneficia es el burbuja solo el doble de rápido.
- A mayor potencia de cómputo menor tiempo independientemente de la RAM ya que en estos casos no se ha hecho mucho uso de ella.
- Apenas hay diferencias debido al sistema operativo, la diferencia observada no alcanza ni un 10%