Práctica 1: Análisis de Eficiencia de Algoritmos

Curso 2016-2017

Jesús Miguel Jaldo Ruiz

Alberto Ortega Ruiz

Patricia Retamero

Pablo Rey Pedrosa

Juan Manuel Salcedo Serrado

1.INTRODUCCIÓN

De cara al cumplimiento de los objetivos marcados para esta práctica se desarrollará el cálculo de la eficiencia empírica e híbrida para el análisis de los siguientes algoritmos:

- Burbuja.
- Inserción.
- Selección.
- Mergesort.
- Quicksort.
- Heapsort.
- Floyd.
- Fibonacci.

Se ejecutarán versiones de dichos algoritmos en los lenguajes C++ y Python.

2.CARACTERÍSTICAS DEL ORDENADOR USADO

Procesador: Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz x4

Memoria: 8GB

Sistema Operativo: Fedora 24

3.ALGORITMOS DE ORDENACIÓN

3.1.EFICIENCIA *O*(n²)

Los algoritmos a estudiar en este apartado son: burbuja, inserción y selección.

Eficiencia empírica

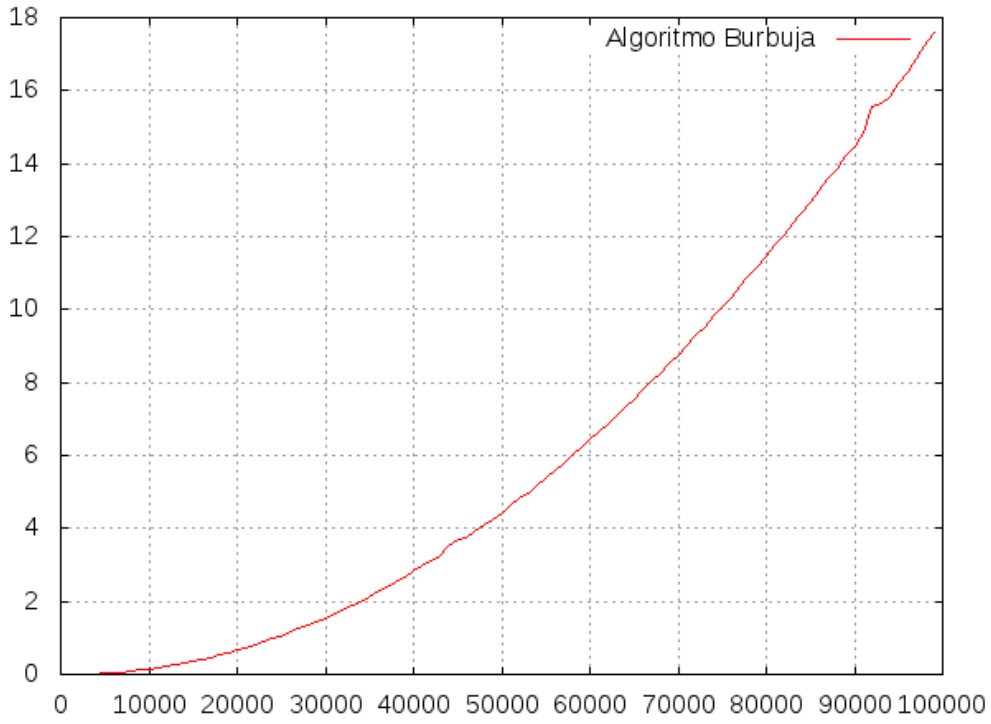
Tablas de resultados

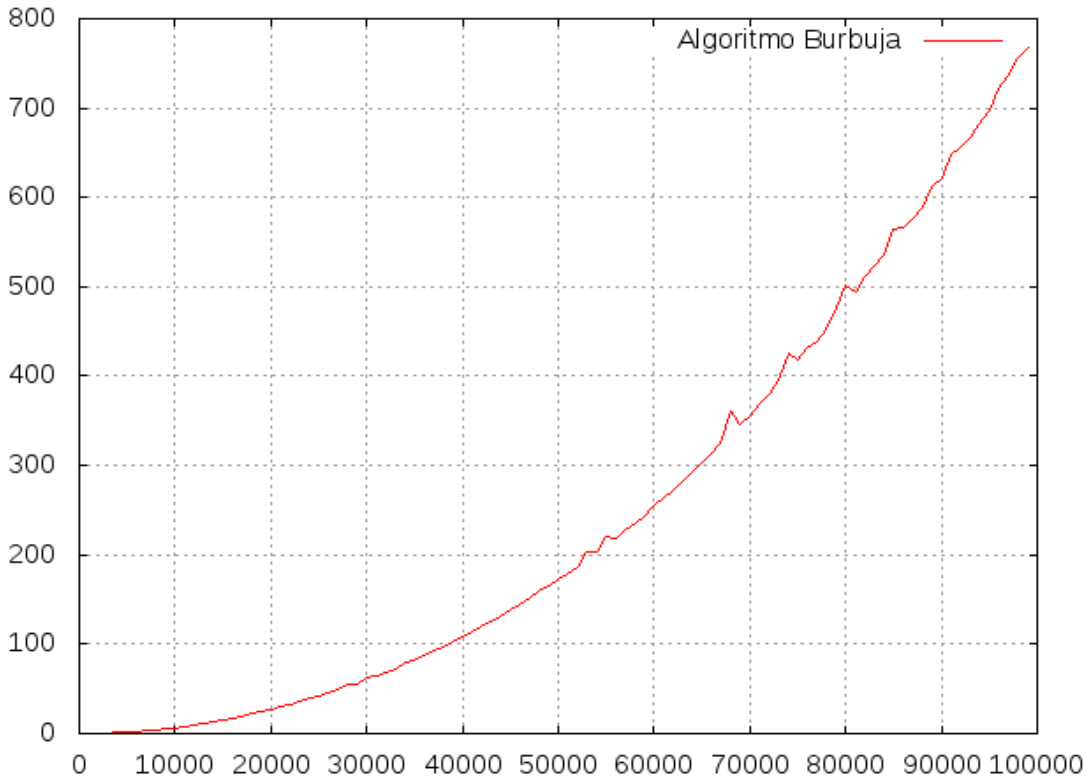
Tamaño	Burbuja (C++)	Inserción (C++)	Selección (C++)
1000	0.000879541	0.000540269	0.000553594
10000	0.143727	0.0568834	0.0385964
20000	0.655584	0.226457	0.151688
35000	2.13808	0.689446	0.460826
50000	4.43309	1.39916	0.93838
60000	6.44763	2.0176	1.34969
70000	8.76553	2.77507	1.83878
80000	11.4739	3.62353	2.39648
90000	14.49	4.58403	3.03055
99000	17.5868	5.52892	3.70871

Tamaño	Burbuja (Python)	Inserción (Python)	Selección (Python)
1000	0.0649600028992	0.0473279953003	0.0228400230408
10000	6.5684299469	5.05886292458	2.46224498749
20000	26.516505003	20.1673858166	10.3445651531
35000	82.3514251709	62.5369391441	33.9319210052
50000	172.346644163	126.830260992	71.7569289207
60000	254.888651133	184.649797916	114.514564991
70000	354.585552931	251.304101944	162.101373911
80000	500.905437231	329.282088995	235.244632959
90000	620.622777939	419.442105055	298.314745903
99000	767.95497489	511.34738493	398.60297513

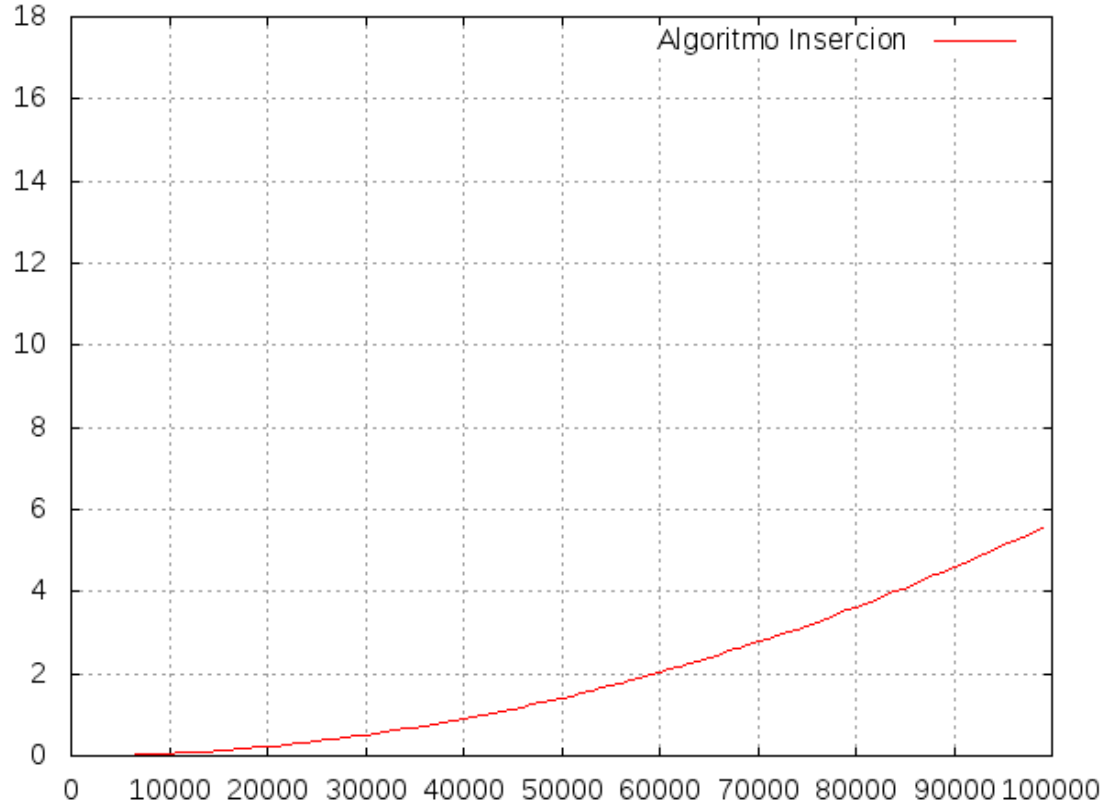
Gráficas de eficiencia empírica(primero C++ y después Python)

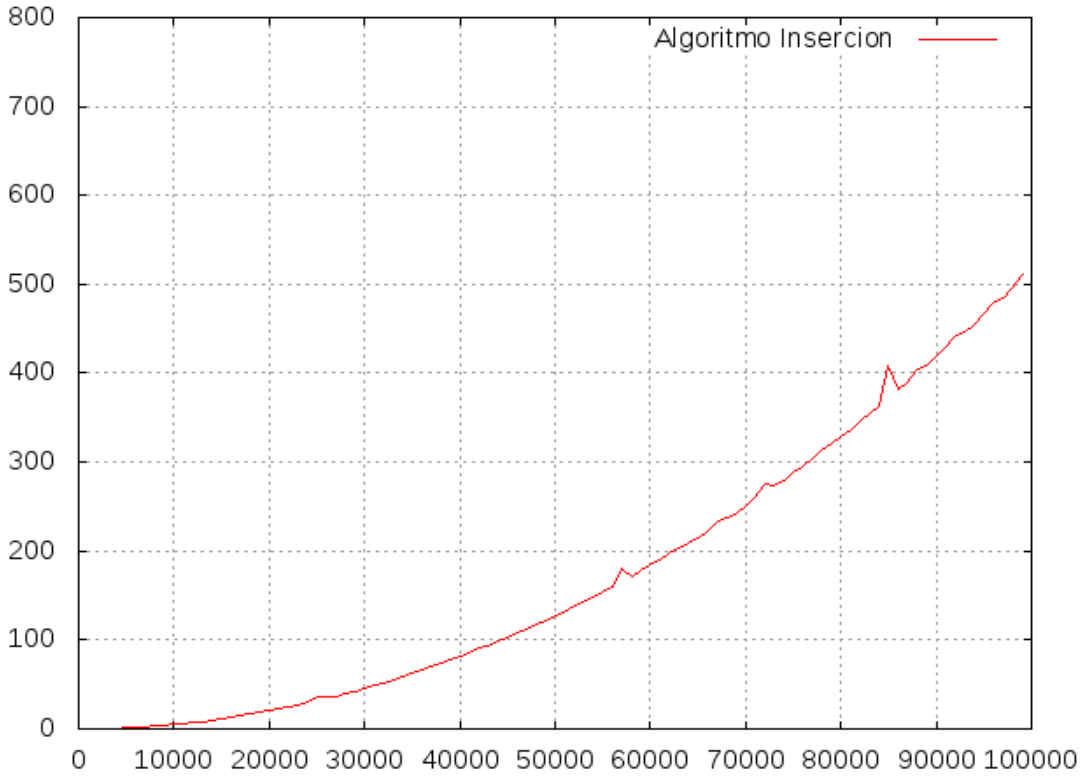
- **Burbuja**



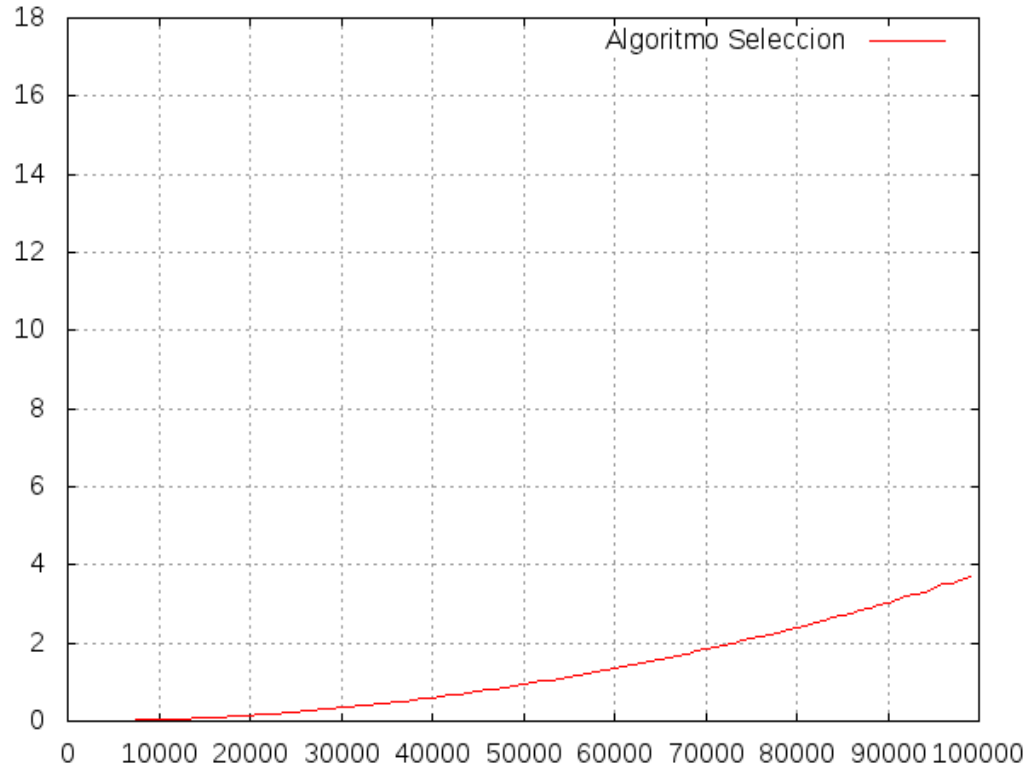


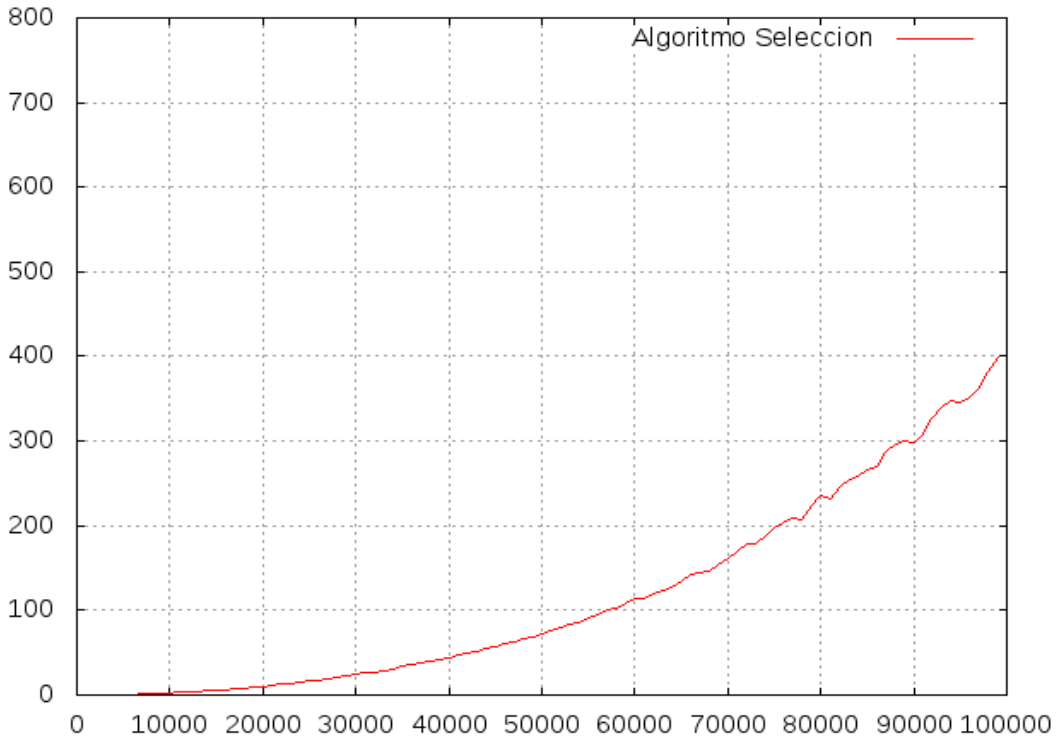
- **Inserción**


Algoritmo Insercion
18
16
14
12
10
8
6
4
2
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000

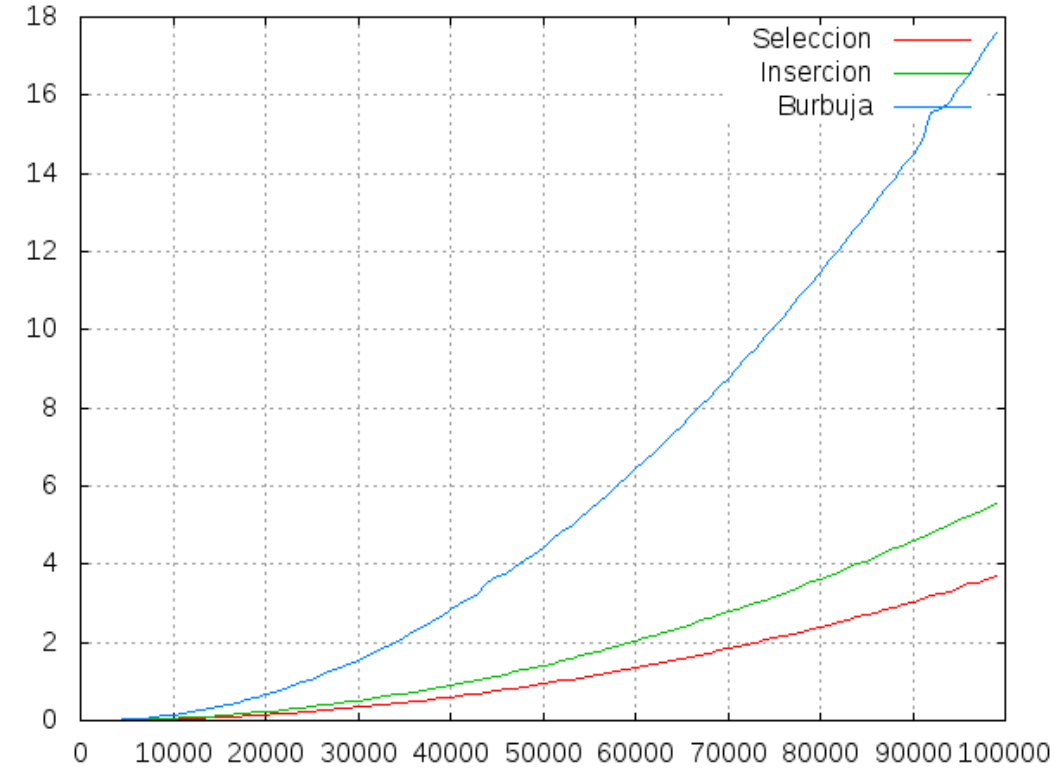

Algoritmo Insercion
800
700
600
500
400
300
200
100
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000

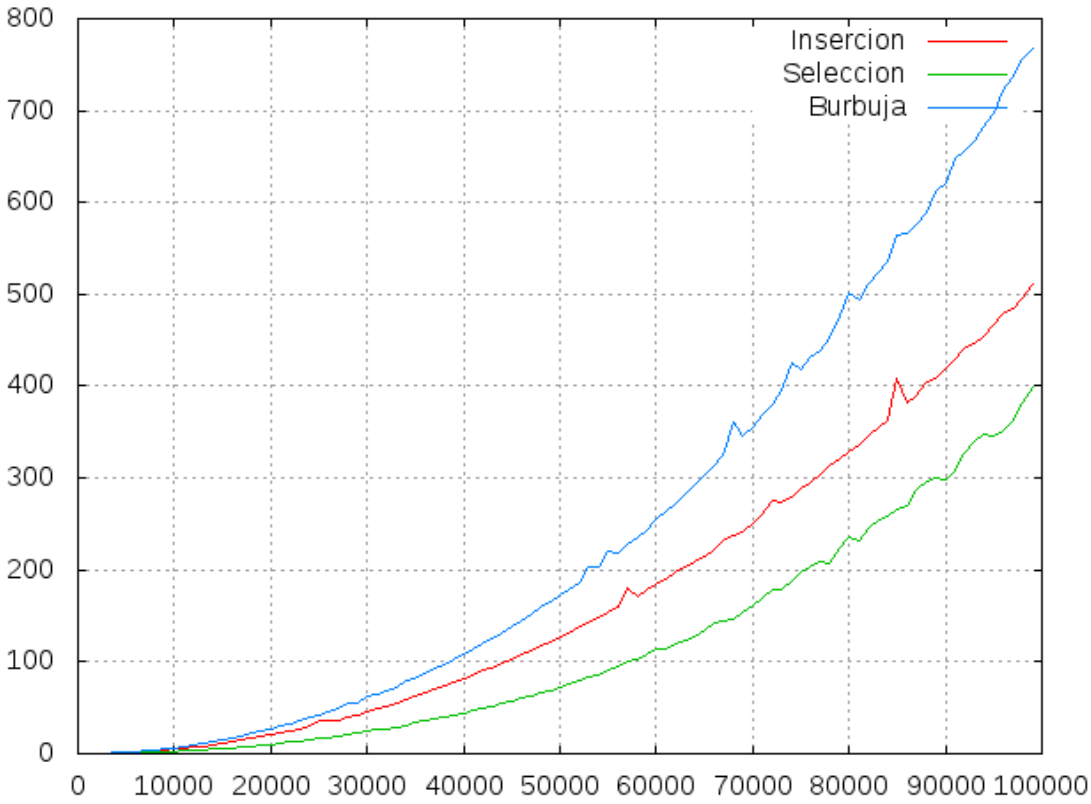
- **Selección**


Algoritmo Seleccion
18
16
14
12
10
8
6
4
2
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000


Algoritmo Seleccion
800
700
600
500
400
300
200
100
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000

- **Comparación**



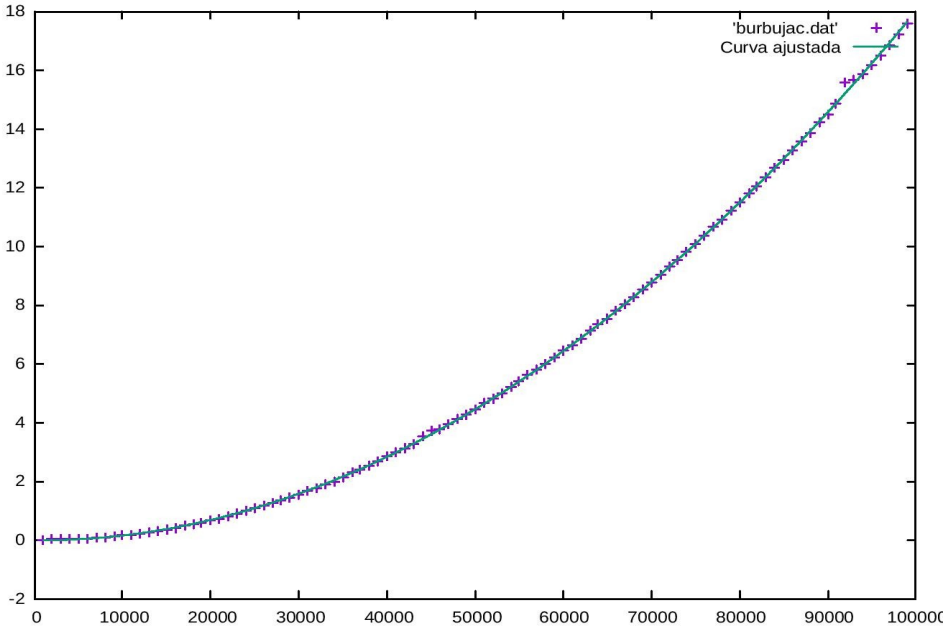


Eficiencia híbrida

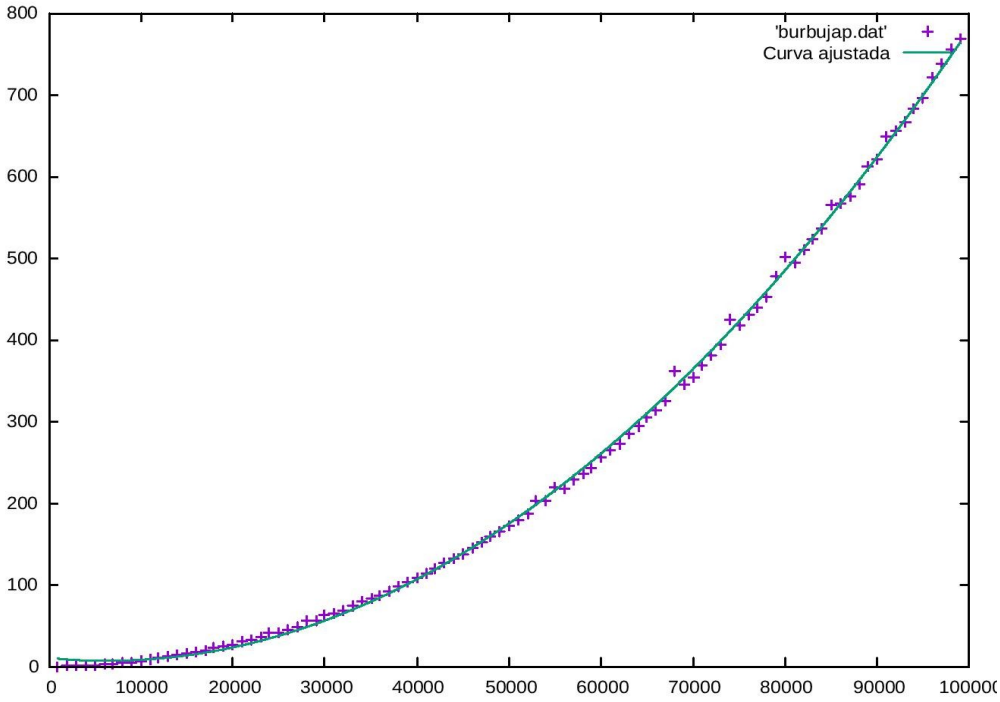
Ajustes de las funciones(primero C++ y después Python):

- **Burbuja,** f(x) = a0*x*x+a1*x+a2

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 1.81597e-09      +/- 6.228e-12    (0.343%)
a1              = -1.52099e-06     +/- 6.428e-07    (42.26%)
a2              = -0.0247509       +/- 0.01393      (56.27%)
```

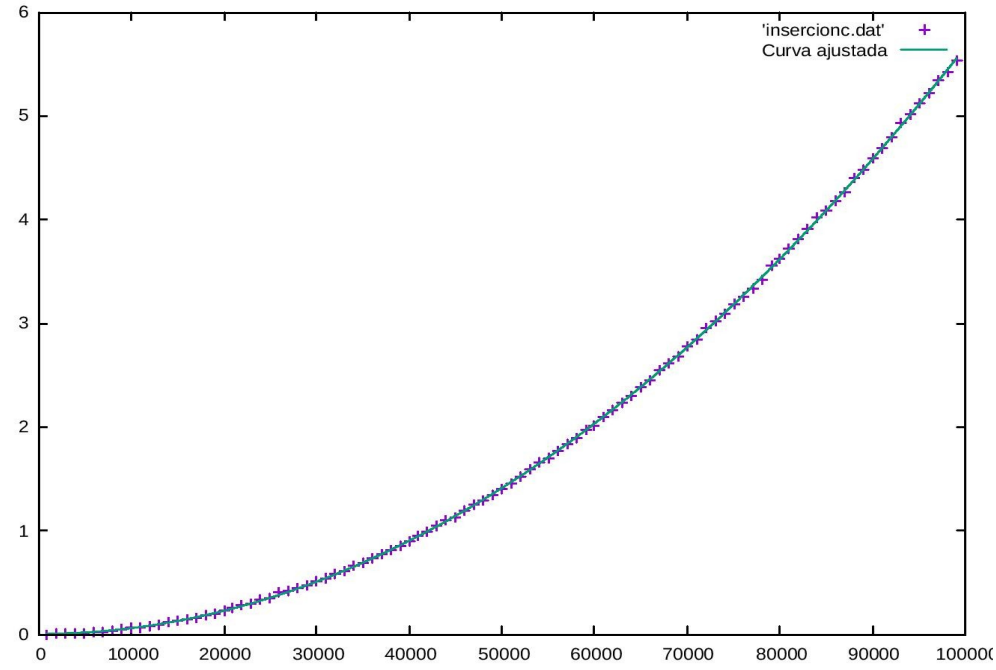



```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 8.8213e-08       +/- 7.789e-10    (0.883%)
a1              = -0.00112098      +/- 8.04e-05     (7.172%)
a2              = 10.0479          +/- 1.742        (17.33%)
```

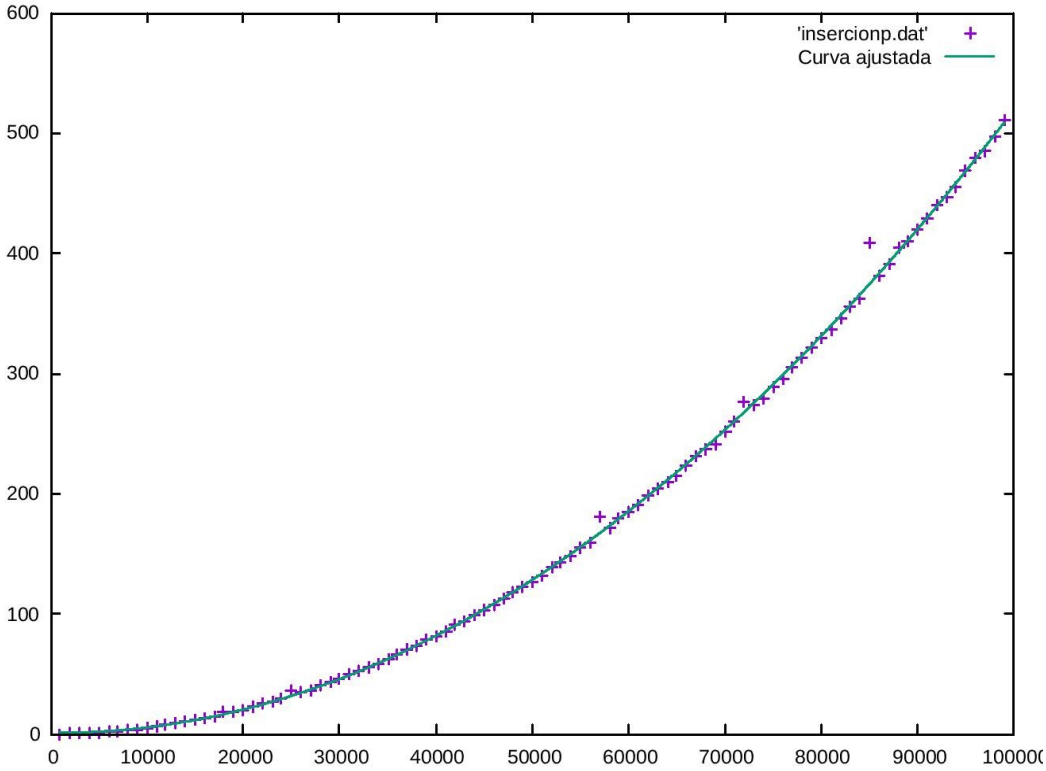


- **Inserción,** f(x) = a0*x*x+a1*x+a2

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 5.70499e-10      +/- 1.285e-12    (0.2253%)
a1              = -4.40932e-07     +/- 1.326e-07    (30.08%)
a2              = 0.00488587       +/- 0.002874     (58.81%)
```

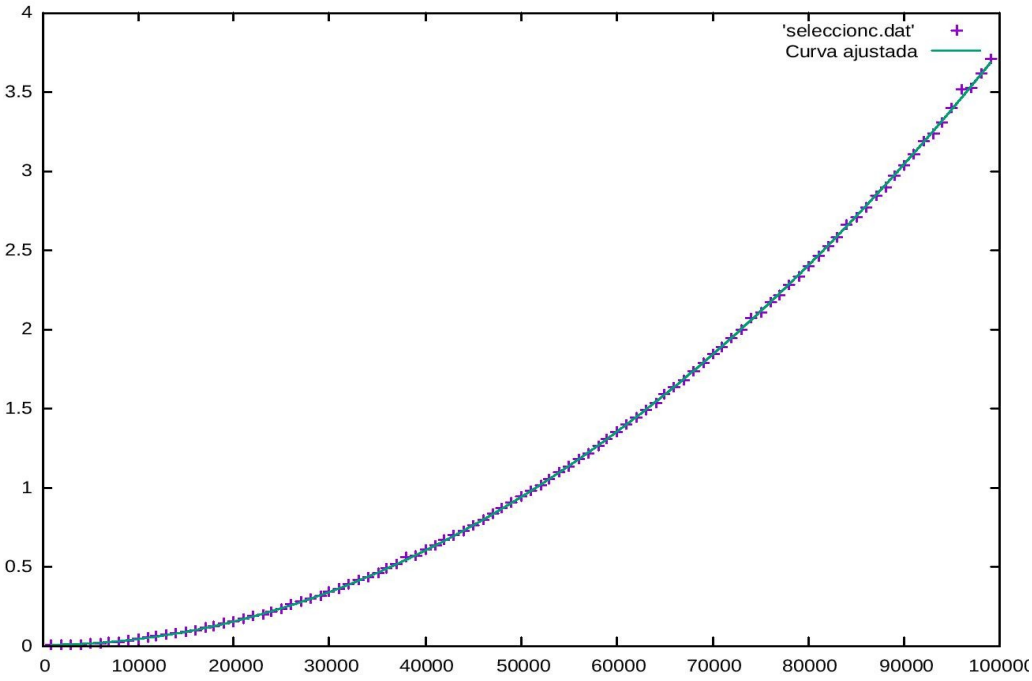



```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 5.28341e-08      +/- 5.785e-10    (1.095%)
a1              = -0.000100087     +/- 5.97e-05     (59.65%)
a2              = 0.990415         +/- 1.293        (130.6%)
```

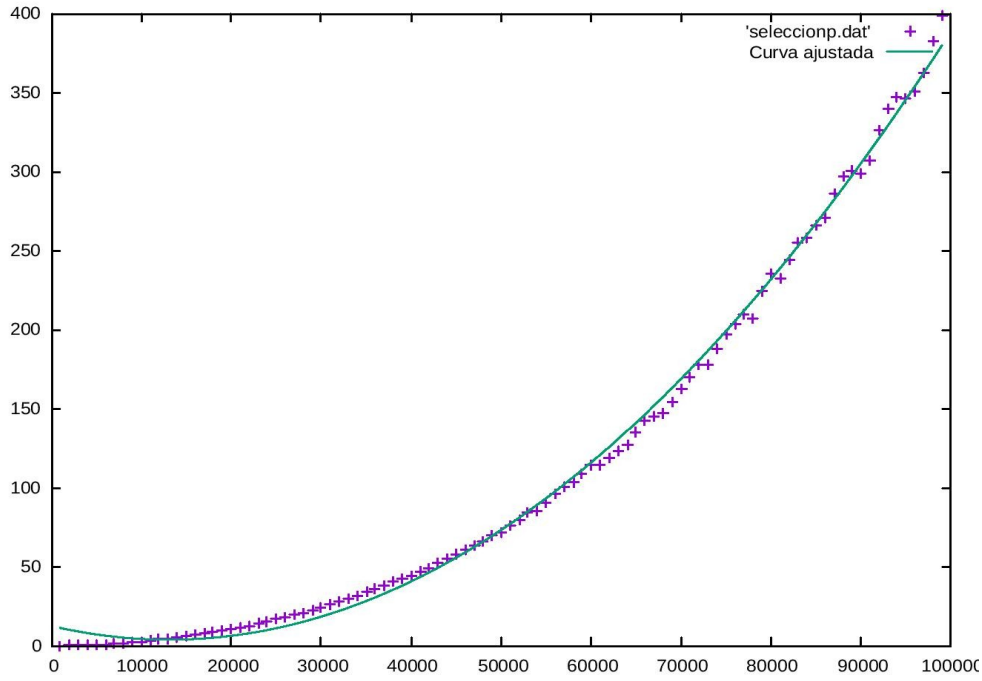


- **Selección,** `f(x) = a0*x*x+a1*x+a2`

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 3.77302e-10      +/- 1.081e-12    (0.2866%)
a1              = -2.00618e-07     +/- 1.116e-07    (55.63%)
a2              = 0.00424519       +/- 0.002418     (56.95%)
```




```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 5.09563e-08      +/- 7.634e-10    (1.498%)
a1              = -0.00133427      +/- 7.88e-05     (5.906%)
a2              = 12.3991          +/- 1.707        (13.77%)
```



3.2.EFICIENCIA *O*(n log n)

Los algoritmos a estudiar en este apartado son: mergesort, quicksort, heapsort.

Eficiencia empírica

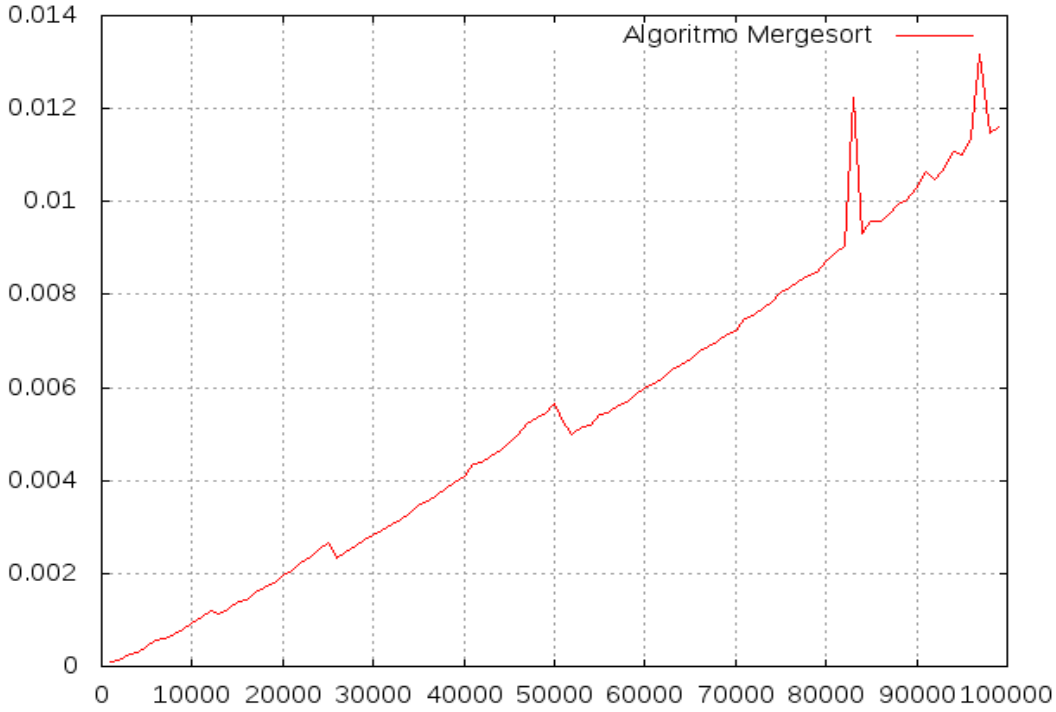
Tablas de resultados

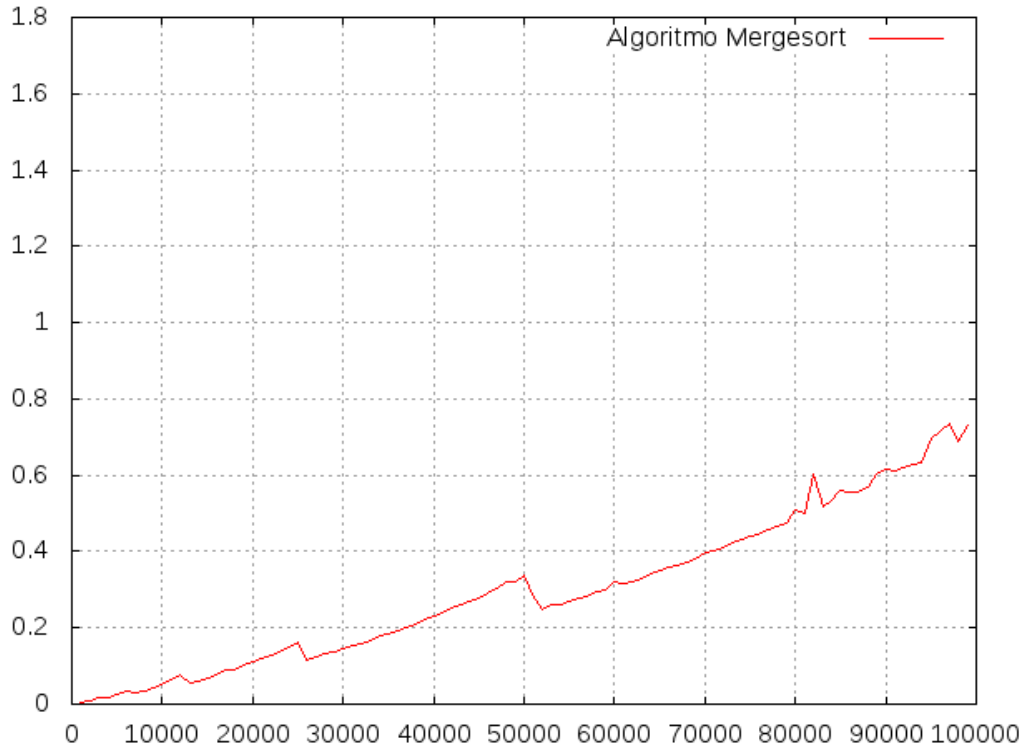
Tamaño	Mergesort(C++)	Quicksort(C++)	Heapsort(C++)
1000	8.6181e-05	6.3762e-05	0.000172567
10000	0.000936577	0.000684618	0.00157897
20000	0.00197736	0.00138921	0.00201405
35000	0.00345	0.00249703	0.00319122
50000	0.00566141	0.00362944	0.00564161
60000	0.00598736	0.00455255	0.00681114
70000	0.00722908	0.00518491	0.00808251
80000	0.00870208	0.00603832	0.00935919
90000	0.0103219	0.00697616	0.0107161
99000	0.0116134	0.00759317	0.0119418

Tamaño	Mergesort(Python)	Quicksort(Python)	Heapsort(Python)
1000	0.00393104553223	0.00788998603821	0.00400590896606
10000	0.0526428222656	0.132952213287	0.0444979667664
20000	0.109836816788	0.391301870346	0.0925769805908
35000	0.185222148895	0.455076932907	0.176299095154
50000	0.336151123047	0.77449798584	0.261458158493
60000	0.320370912552	0.909179925919	0.31378698349
70000	0.39608001709	1.04434084892	0.629143953323
80000	0.508186817169	1.17782497406	0.438916921616
90000	0.614493131638	1.50136518478	0.494750022888
99000	0.731216907501	1.70011091232	0.545054912567

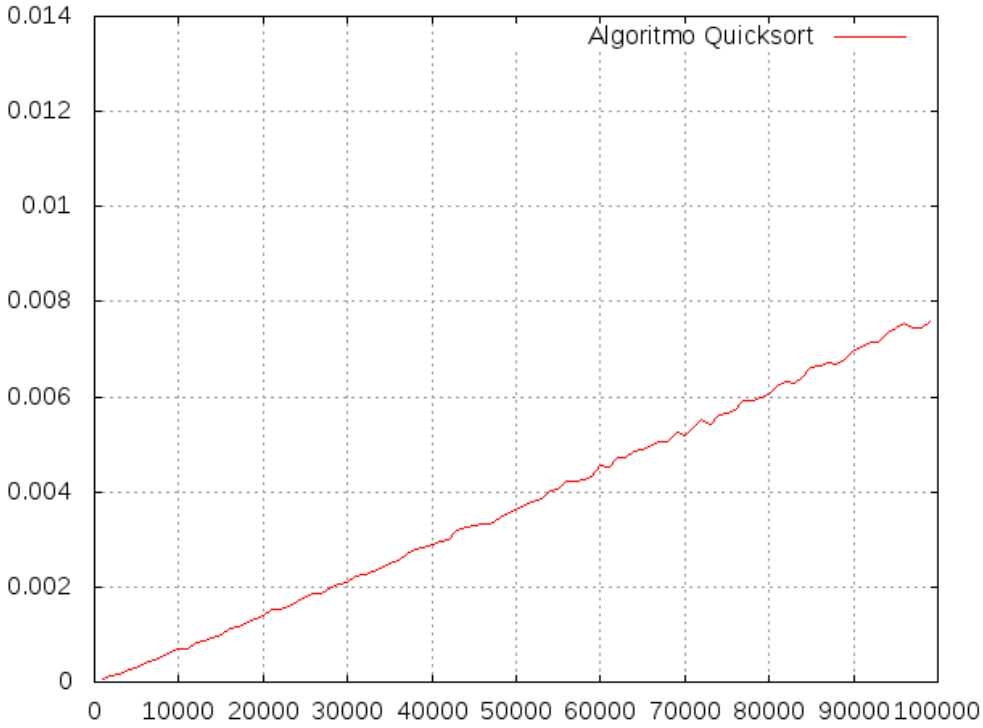
Gráficas de eficiencia empírica(primero C++ y después Python)

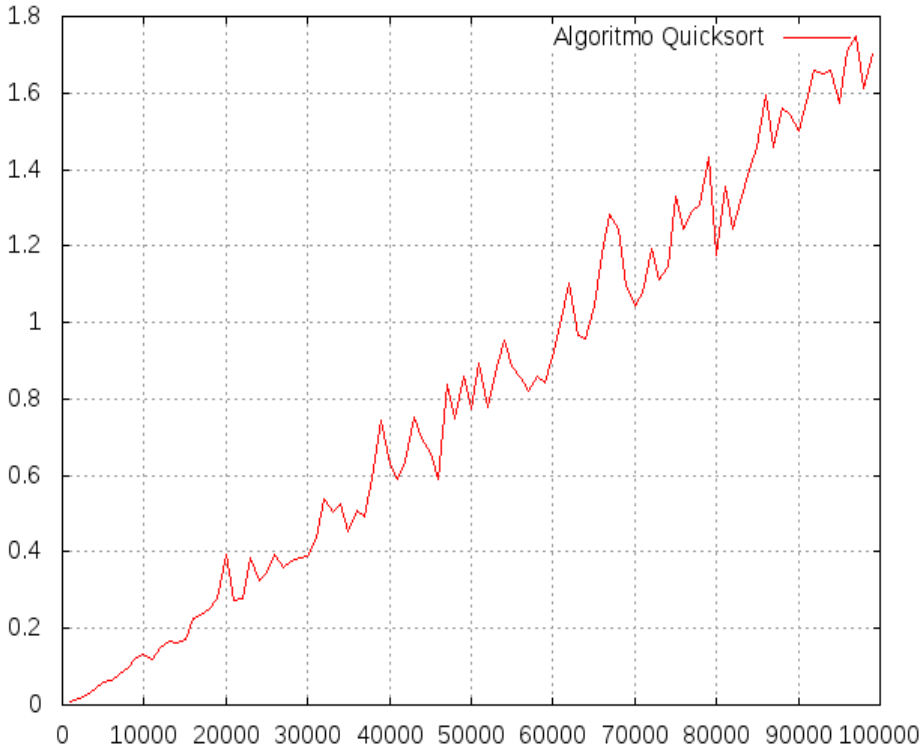
- **Mergesort**



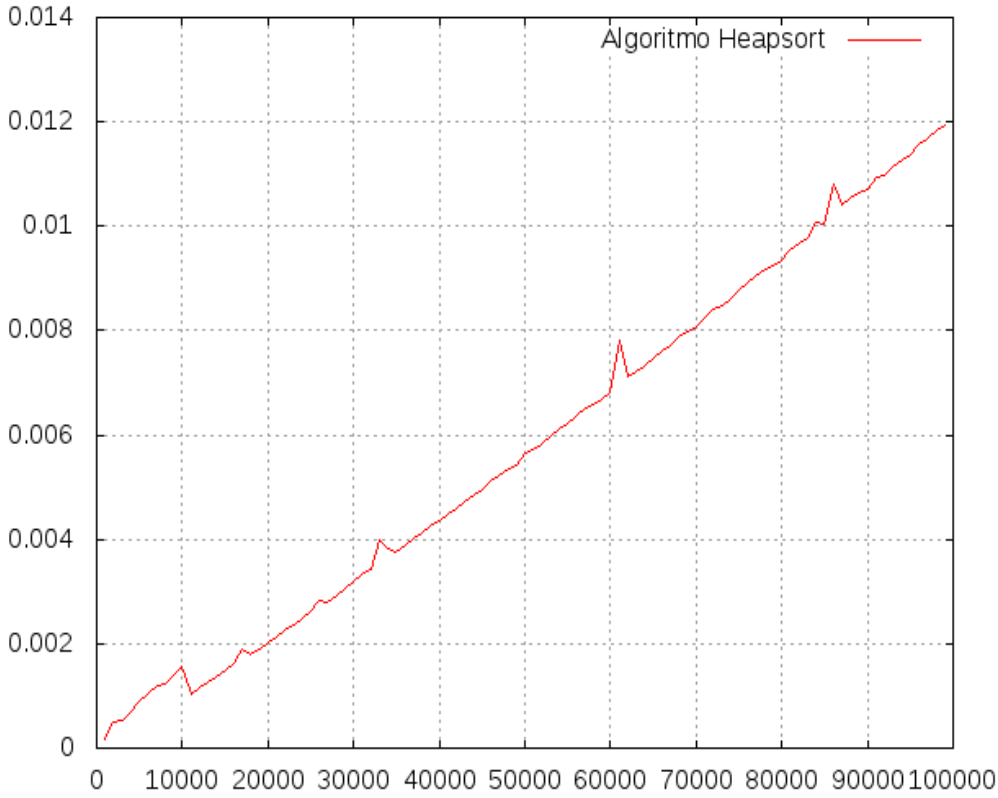


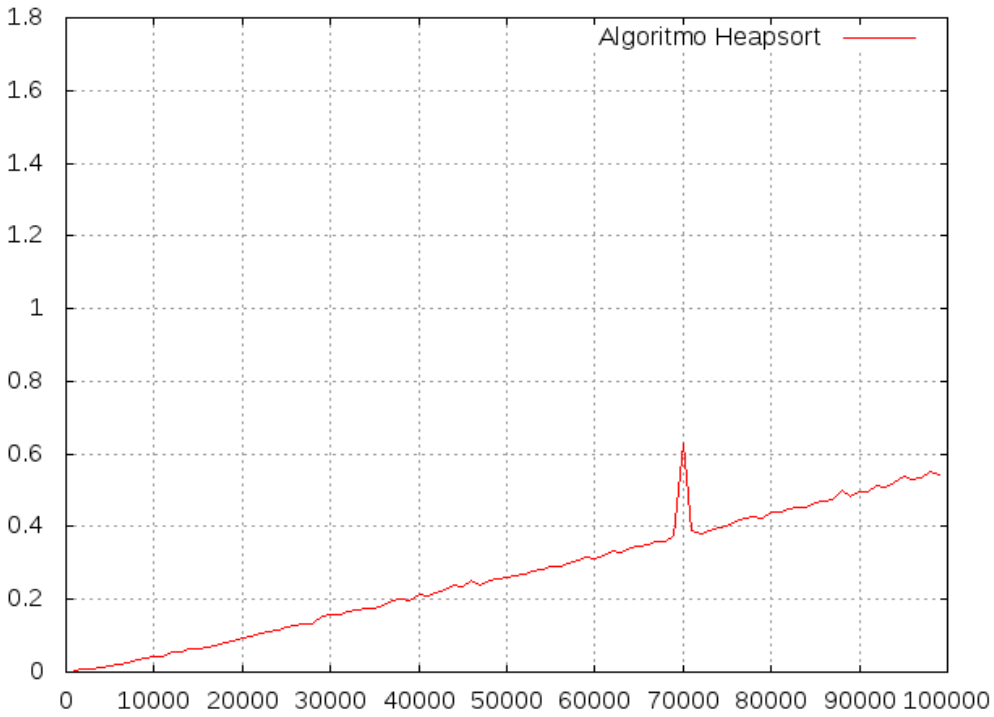
- **Quicksort**


Algoritmo Quicksort
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000

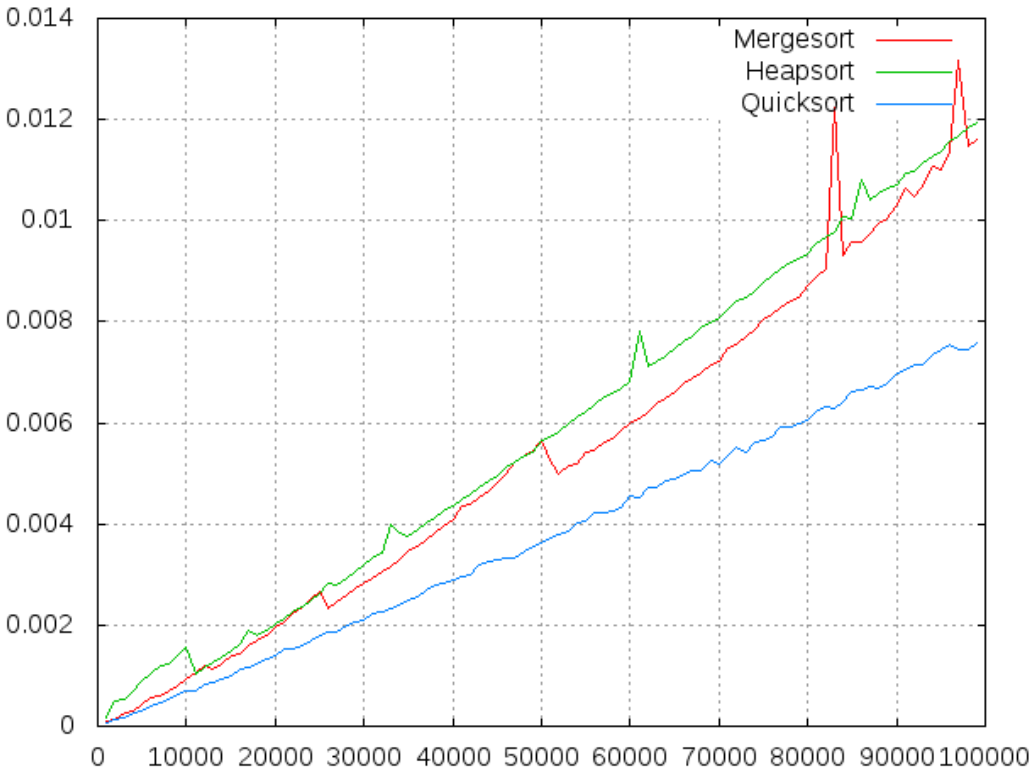

Algoritmo Quicksort
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000

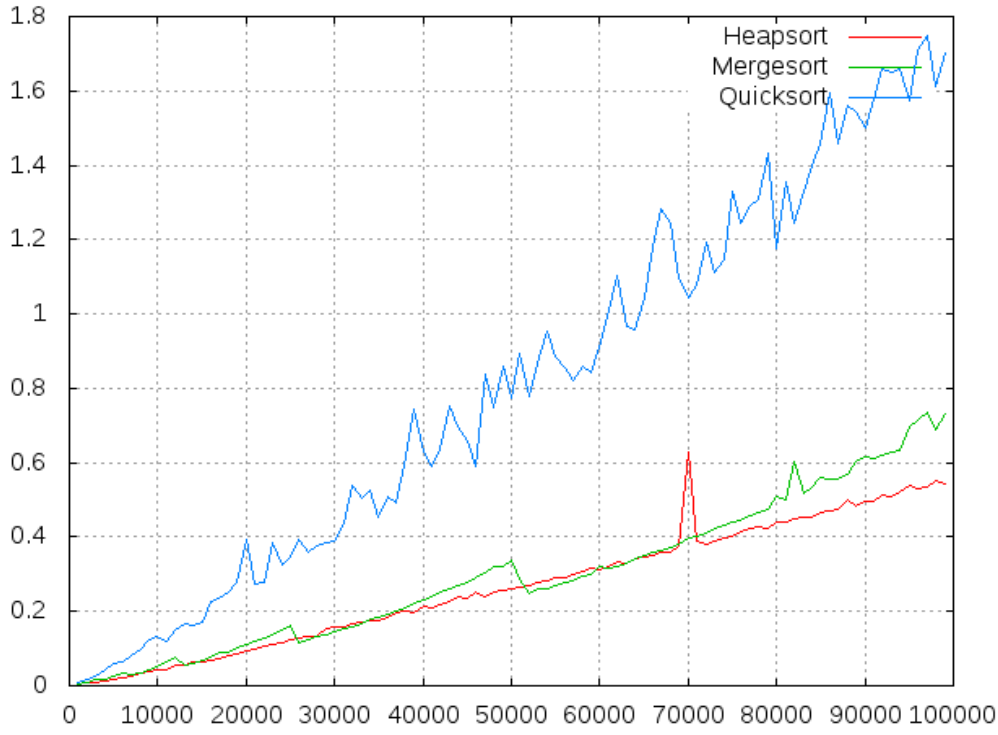
- **Heapsort**


Algoritmo Heapsort
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000


Algoritmo Heapsort
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000

- **Comparaciones**



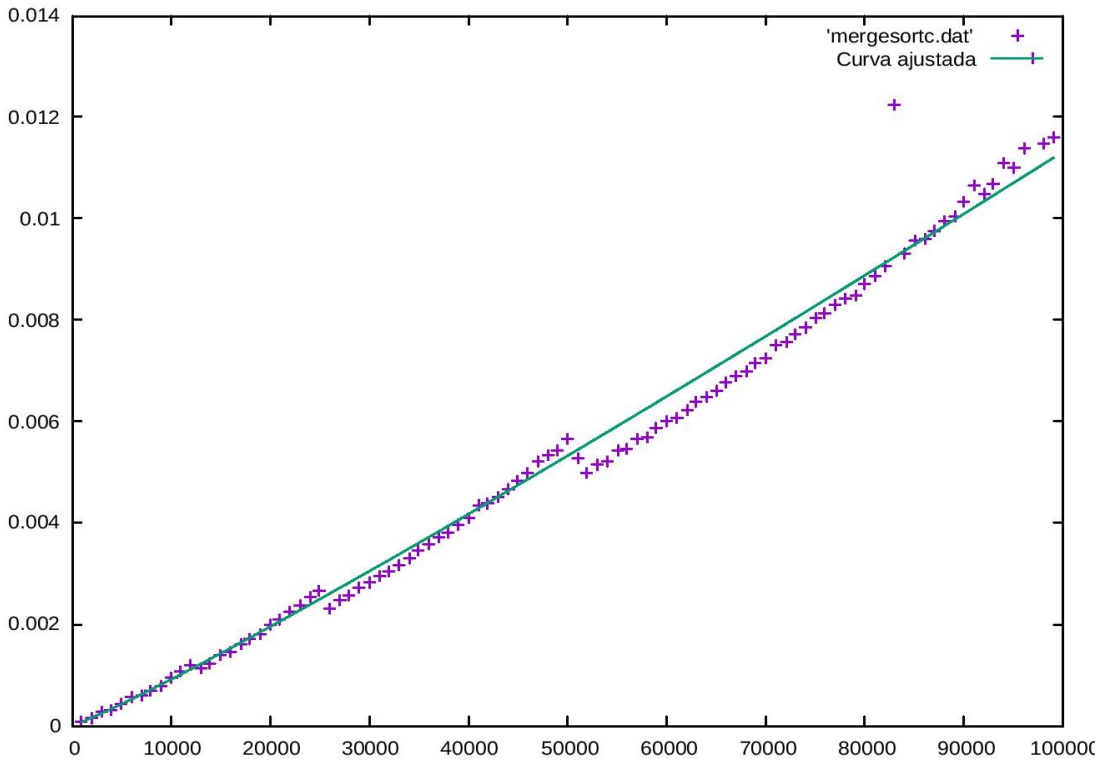


Eficiencia híbrida

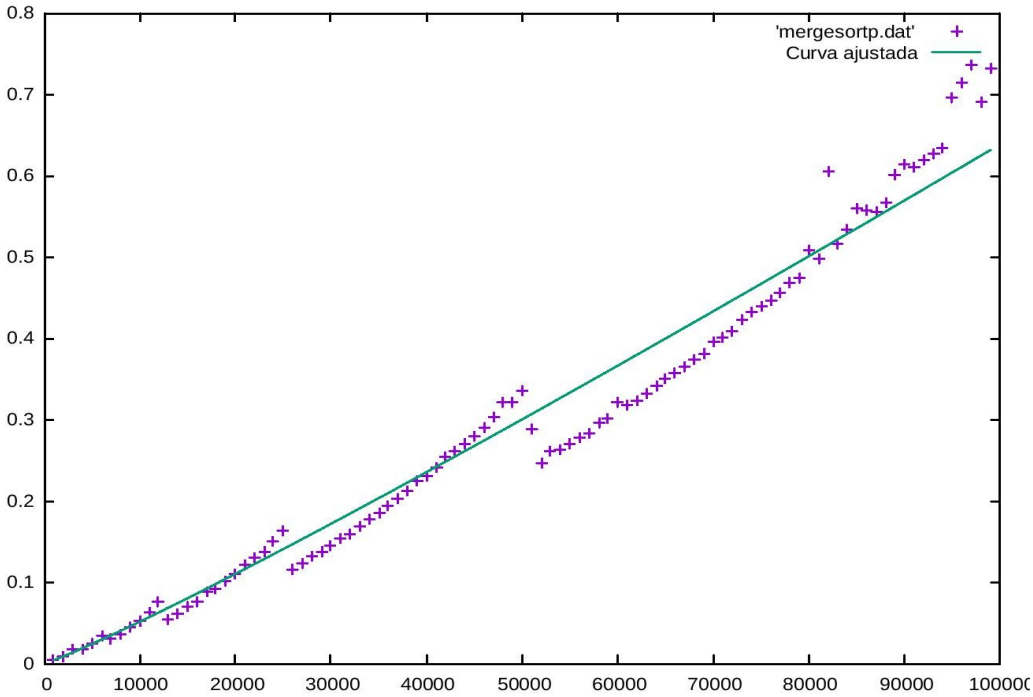
Ajustes de las funciones(primero C++ y después Python):

- **Mergesort,** f(x) = a0*x*(log(x)/log(2))

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 6.81417e-09      +/- 5.09e-11     (0.747%)
```

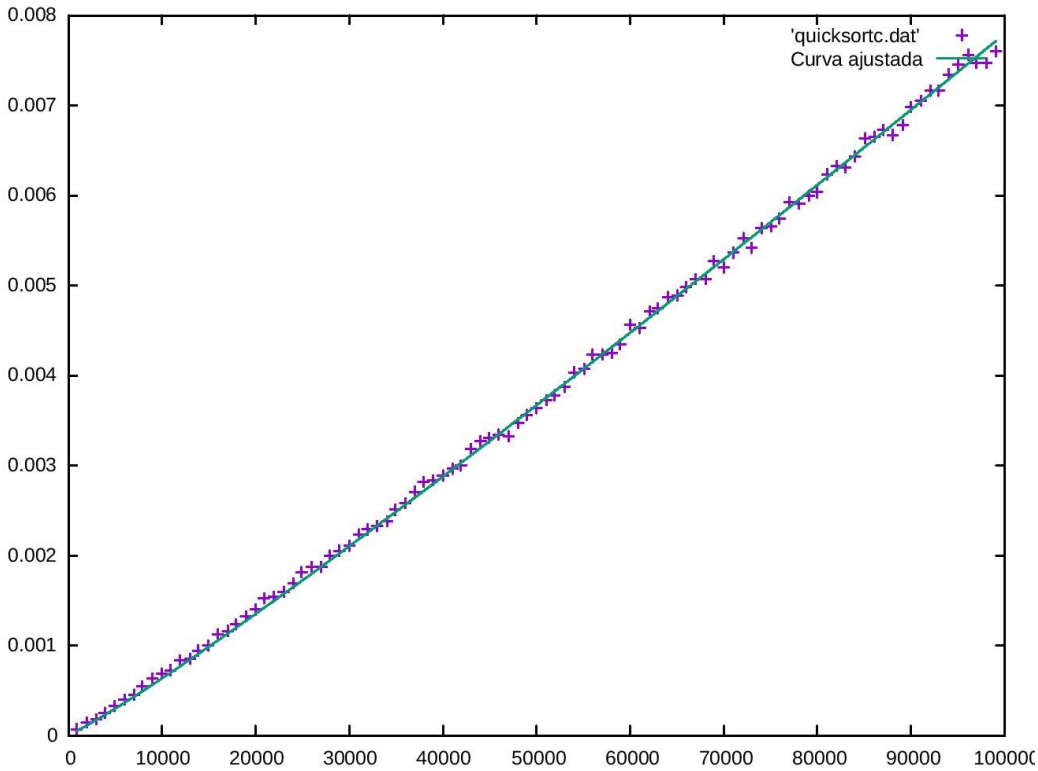



```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 3.84371e-07      +/- 4.137e-09    (1.076%)
```

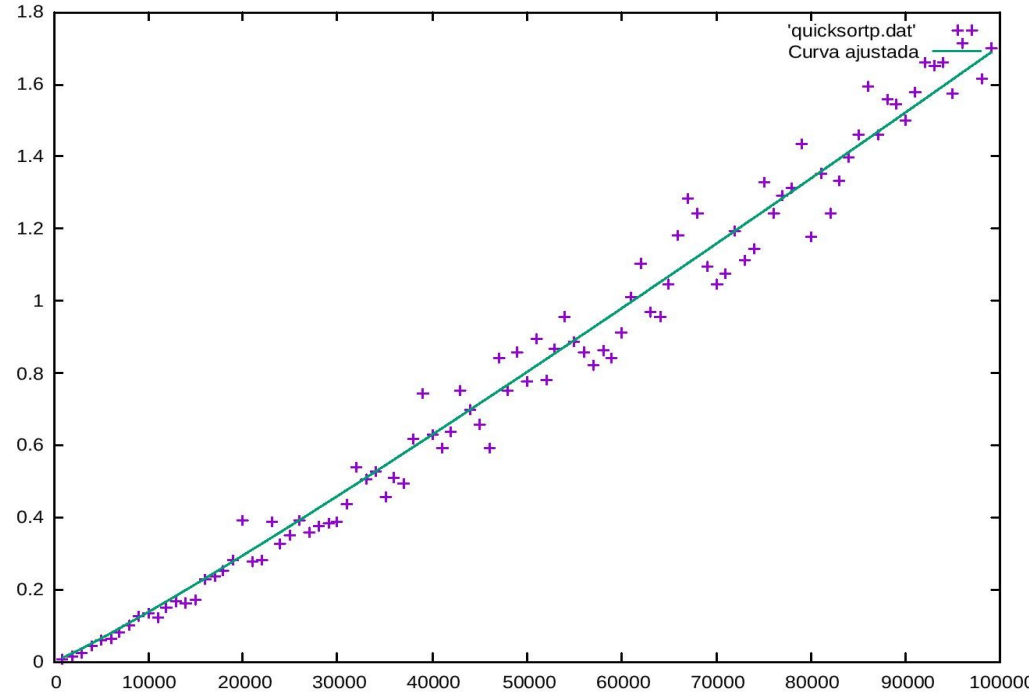


- **Quicksort,** f(x) = a0*x*(log(x)/log(2))

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 4.69096e-09      +/- 6.188e-12    (0.1319%)
```

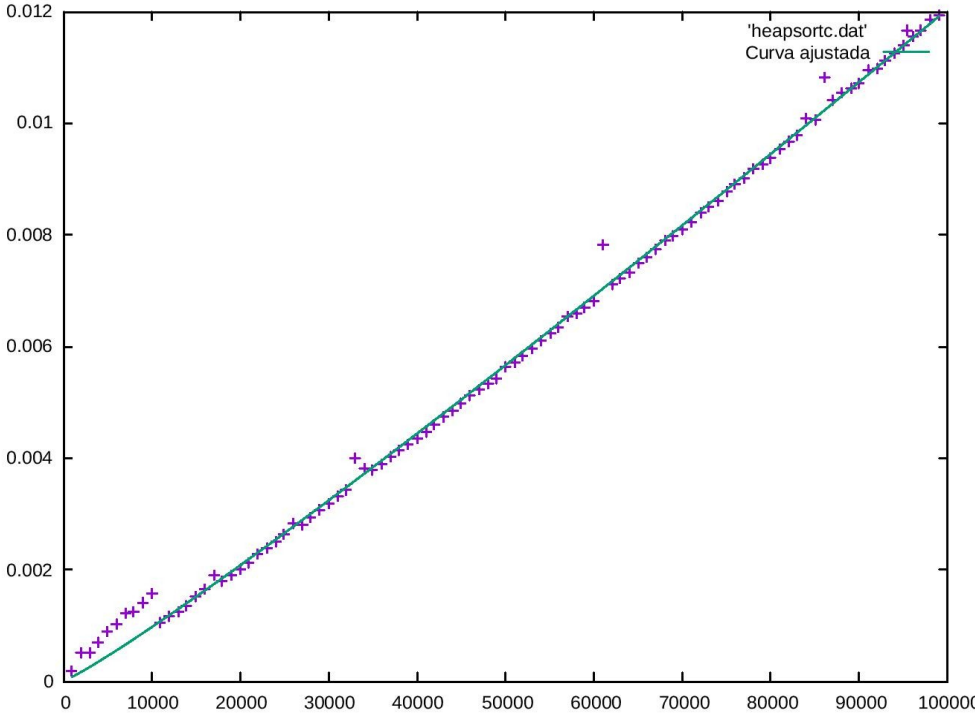



```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 1.02733e-06      +/- 7.217e-09    (0.7025%)
```

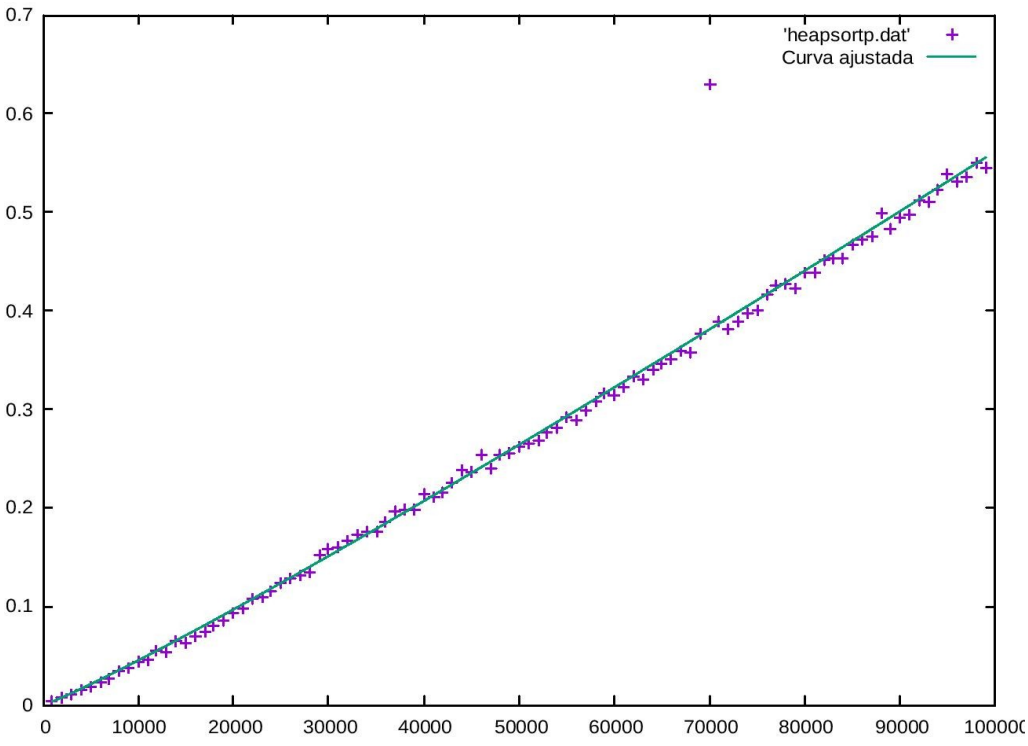


- **Heapsort,** f(x) = a0*x*(log(x)/log(2))

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 7.25048e-09      +/- 2.016e-11    (0.278%)
```




```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 3.38096e-07      +/- 2.785e-09    (0.8238%)
```



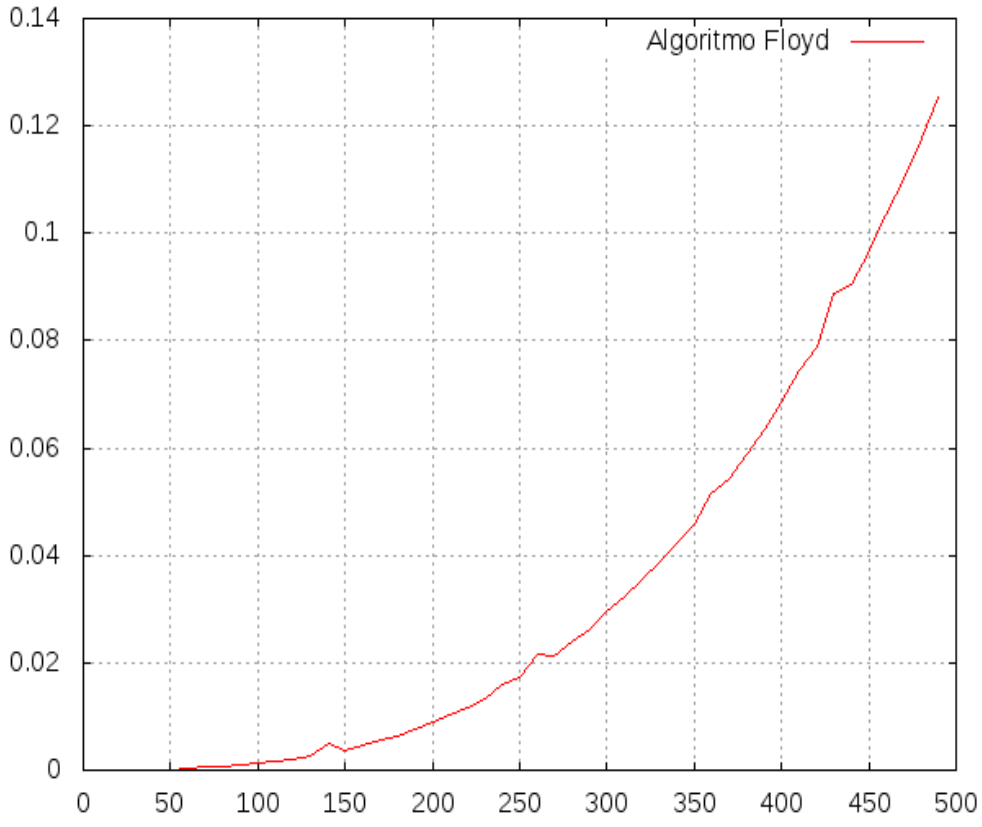
4.ALGORITMO DE FLOYD *O*(n³)

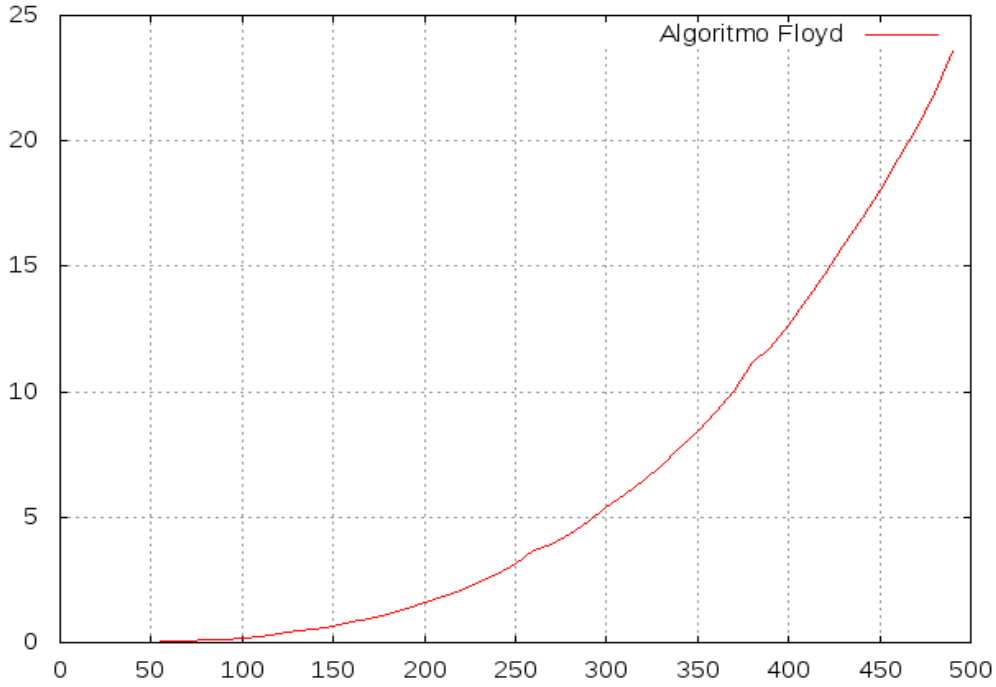
Eficiencia empírica

Tablas de resultados

Tamaño	C++	Python
10	1.482e-06	0.00023889541626
50	0.000155283	0.0250079631805
100	0.00126338	0.201648950577
150	0.0038006	0.668588876724
200	0.00894525	1.57522082329
250	0.0173707	3.17275905609
300	0.029572	5.39323401451
350	0.0459378	8.4226899147
400	0.0684161	12.6614019871
490	0.125531	23.5752220154

Gráficas de eficiencia empírica(primero en C++ y después en Python)



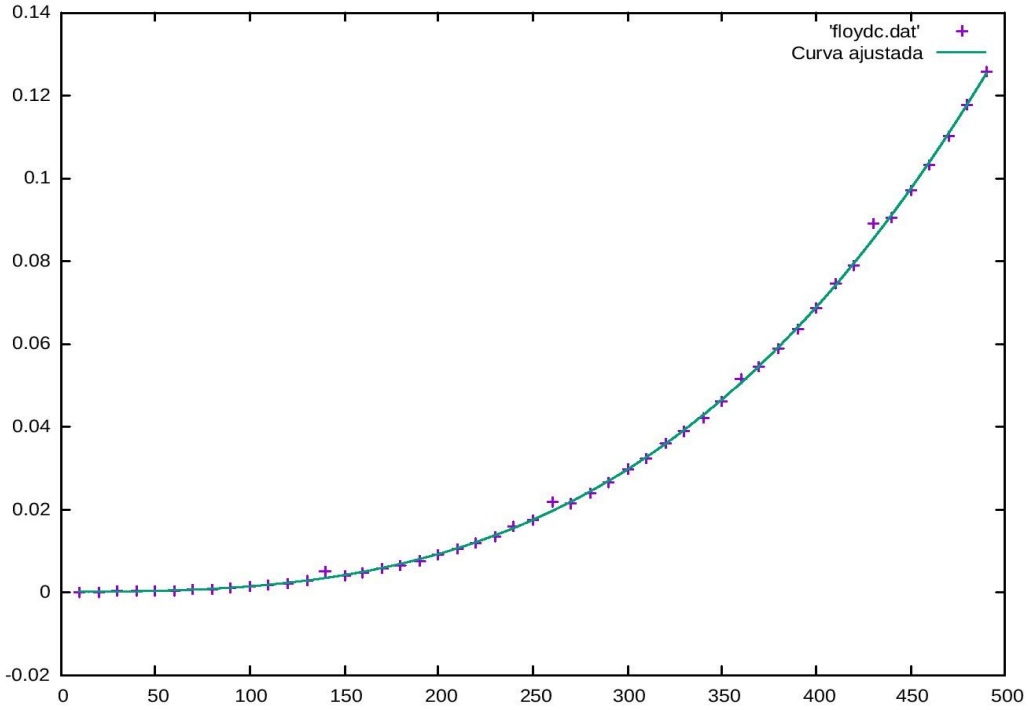


Eficiencia híbrida

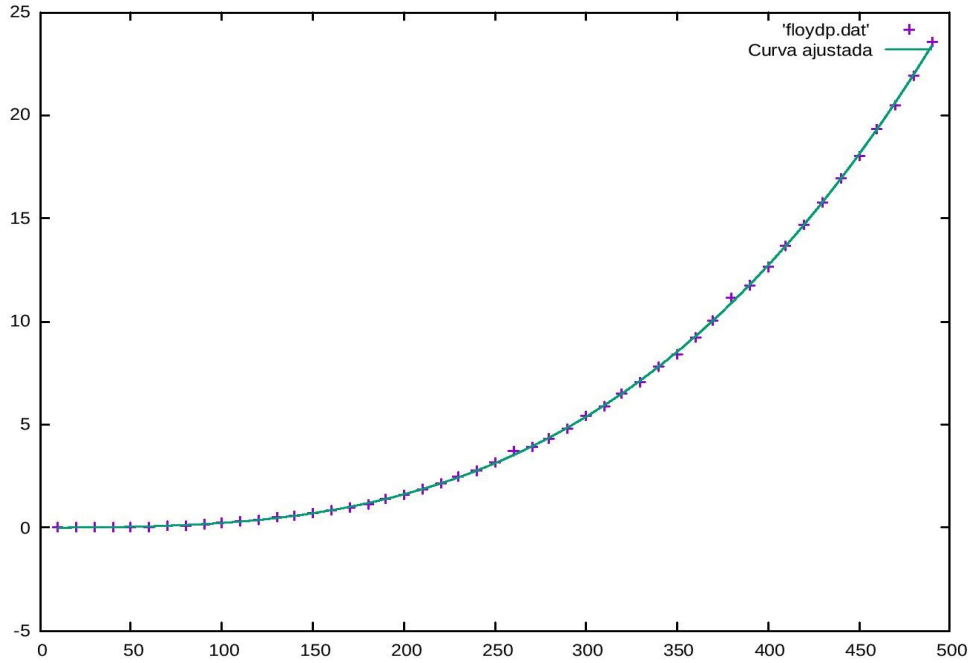
Ajustes de la función (primero C++ y después Python):

- `f(x) = a0*x*x*x+a1*x*x+a2*x+a3`

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 1.03039e-09      +/- 5.024e-11    (4.876%)
a1              = 1.25205e-08      +/- 3.819e-08    (305%)
a2              = 2.20194e-06      +/- 8.263e-06    (375.2%)
a3              = -5.94601e-05     +/- 0.0004819    (810.4%)
```



```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 2.01341e-07      +/- 4.523e-09    (2.246%)
a1              = -2.33625e-06     +/- 3.438e-06    (147.2%)
a2              = 0.000513157      +/- 0.0007438    (145%)
a3              = -0.0193231       +/- 0.04338 (224.5%)
```

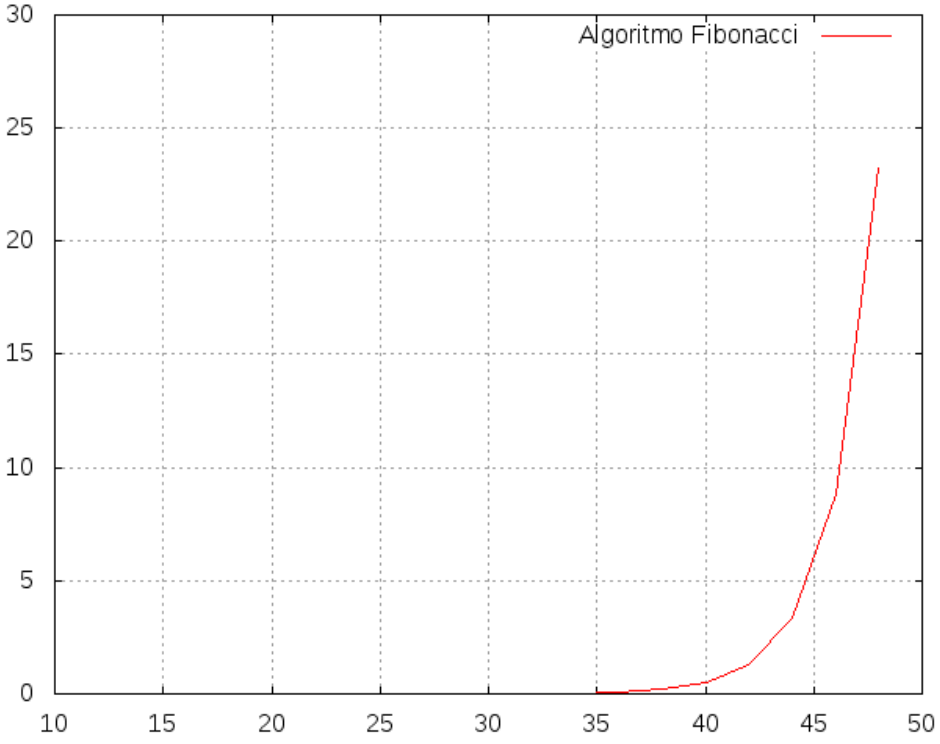


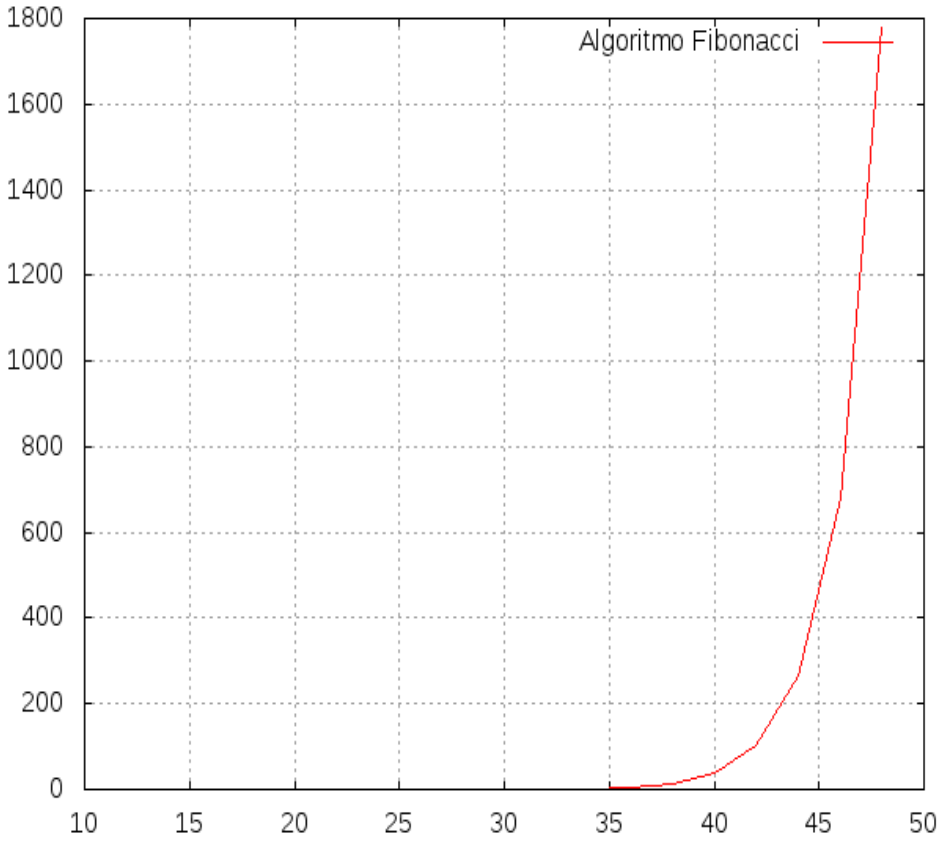
5.ALGORITMO DE FIBONACCI *O*((1+ 2 5) n))

Eficiencia empírica

Tamaño	C++	Python
10	6e-07	4.29153442383e-05
12	1.345e-06	6.103515625e-05
14	2.403e-06	0.000149965286255
18	1.3105e-05	0.000962972640991
22	8.5968e-05	0.00855493545532
26	0.000588946	0.0450170040131
30	0.00421234	0.312088012695
34	0.0275747	2.11239099503
36	0.0723397	5.6723511219
38	0.196365	14.4997828007
40	0.499014	37.7748119831
42	1.29192	101.042599916
44	3.37255	265.218493938
46	8.85076	681.166821957
48	23.2206	1777.12923694

Gráficas de eficiencia empírica (primero en C++ y luego en Python)



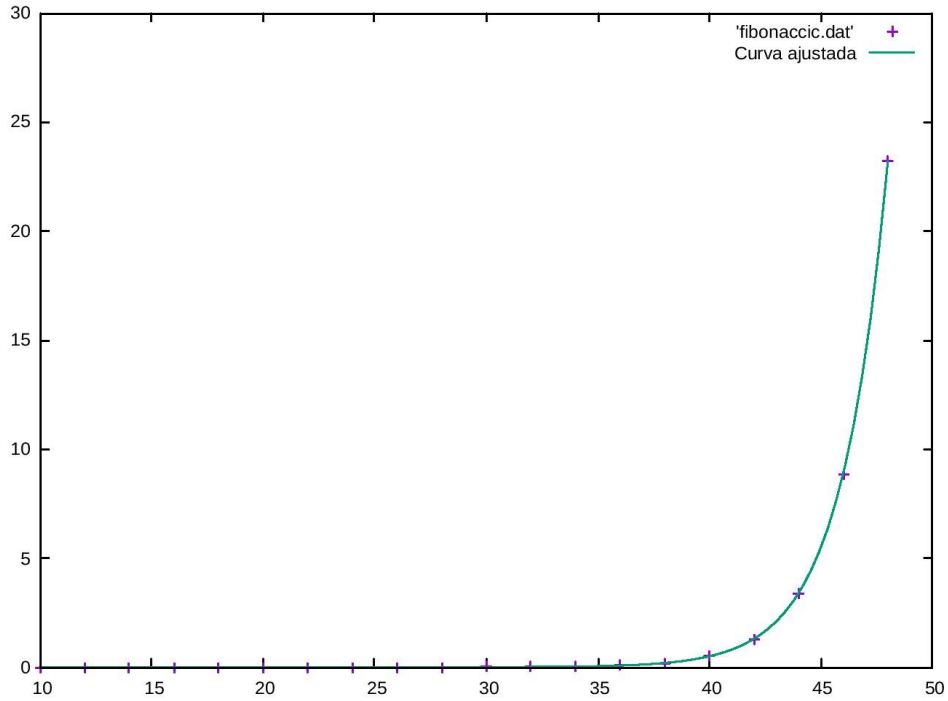


Eficiencia híbrida

Ajustes de la función (primero C++ y después Python):

- `f(x)=a0*((1+sqrt(5))/2)**x`

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 2.15932e-09      +/- 4.813e-13    (0.02229%)
```




```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a0              = 1.65464e-07      +/- 1.277e-10    (0.07715%)
```