



PRÁCTICA 1 ALGORÍTMICA: ANÁLISIS DE EFICIENCIA DE ALGORITMOS



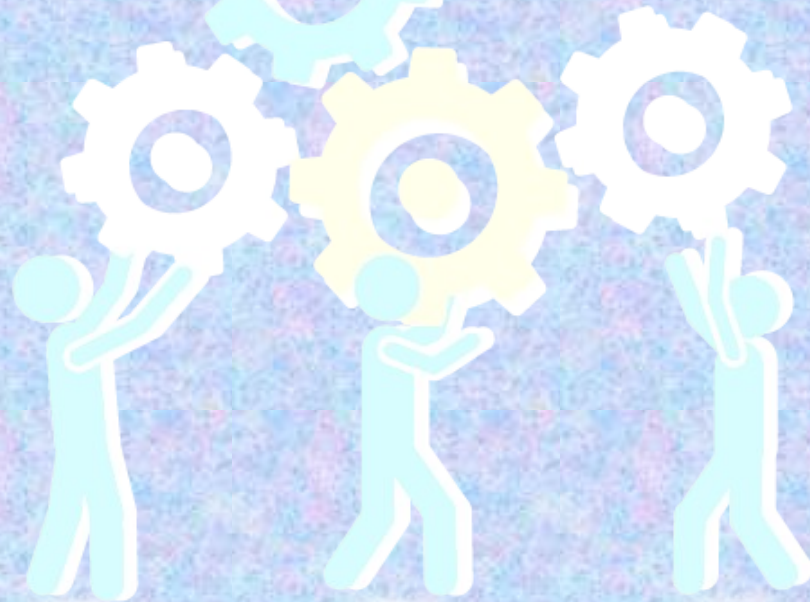
Componentes del Grupo:

Daniel Bolaños Martínez

José María Borrás Serrano

Santiago De Diego De Diego

Fernando De la Hoz Moreno

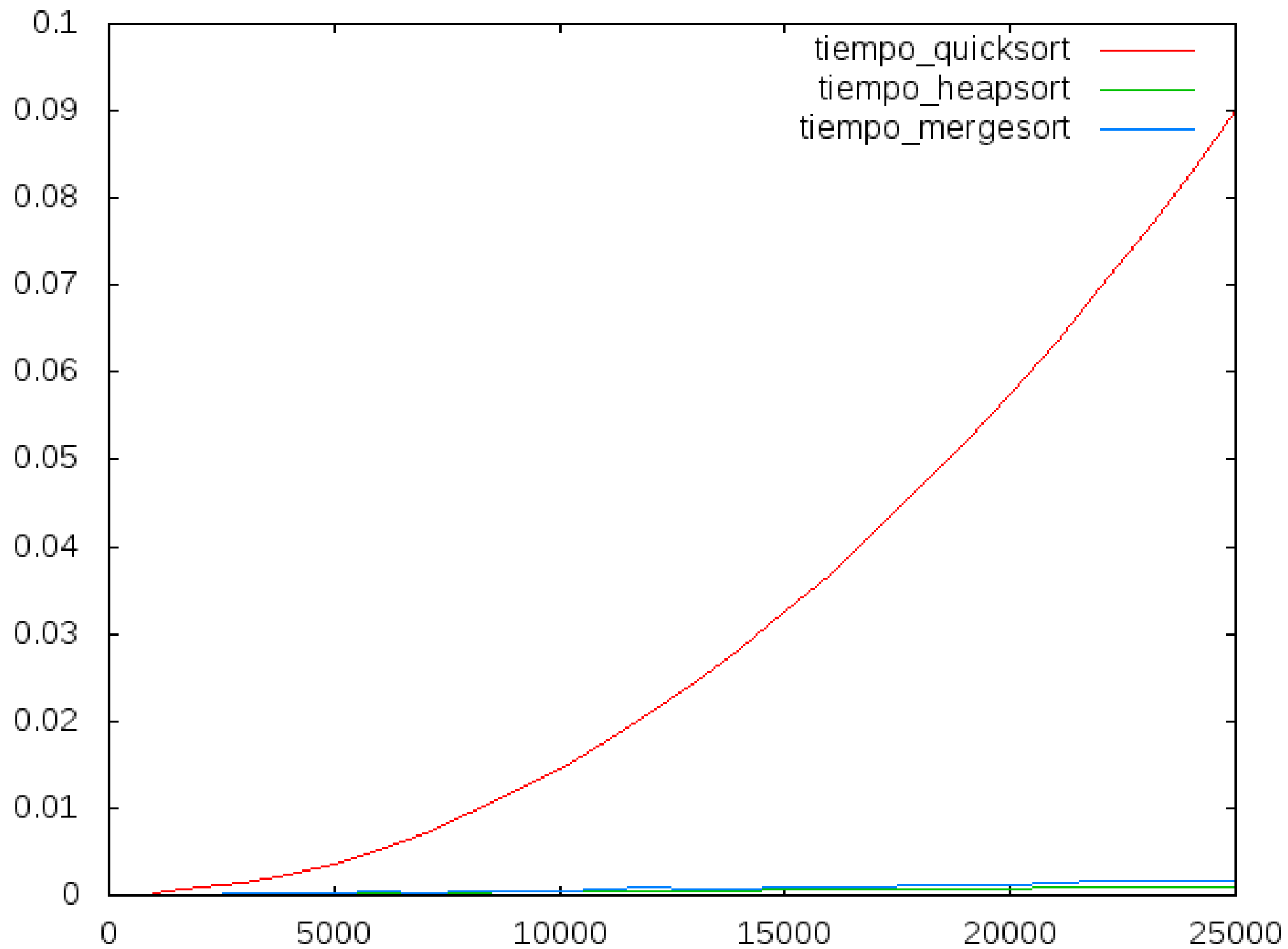


Algoritmos $O(n \log(n))$

- **Heapsort:** almacena los elementos a ordenar en un árbol binario descendente y luego extrae el nodo raíz en sucesivas iteraciones.
- **Mergesort y Quicksort:** ambos se basan en la técnica de divide y vencerás.

Tamaño	Mergesort	Quicksort	Heapsort
1000	0.000128245	9.7e-05	0.000131
2000	0.000274738	0.000205	0.000286
3000	0.000523958	0.000324	0.00044
4000	0.000607748	0.000456	0.000623
5000	0.000854426	0.000585	0.000832
6000	0.00111811	0.000844	0.00103
7000	0.00107502	0.001171	0.001426
8000	0.00129427	0.000891	0.001201
9000	0.00154438	0.00104	0.001362
10000	0.00180574	0.001358	0.001609
11000	0.002111	0.001461	0.001791
12000	0.002448	0.001531	0.002013

Tamaño	Mergesort	Quicksort	Heapsort
13000	0.002119	0.001566	0.002078
14000	0.002362	0.001614	0.002252
15000	0.002603	0.00175	0.002411
16000	0.002832	0.001959	0.002837
17000	0.003074	0.002254	0.00278
18000	0.003349	0.002153	0.002984
19000	0.003615	0.002251	0.003144
20000	0.004063	0.002544	0.003328
21000	0.004241	0.002587	0.003489
22000	0.00448	0.002632	0.003703
23000	0.004788	0.002786	0.003872
24000	0.005128	0.002919	0.004059
25000	0.005399	0.003045	0.004241

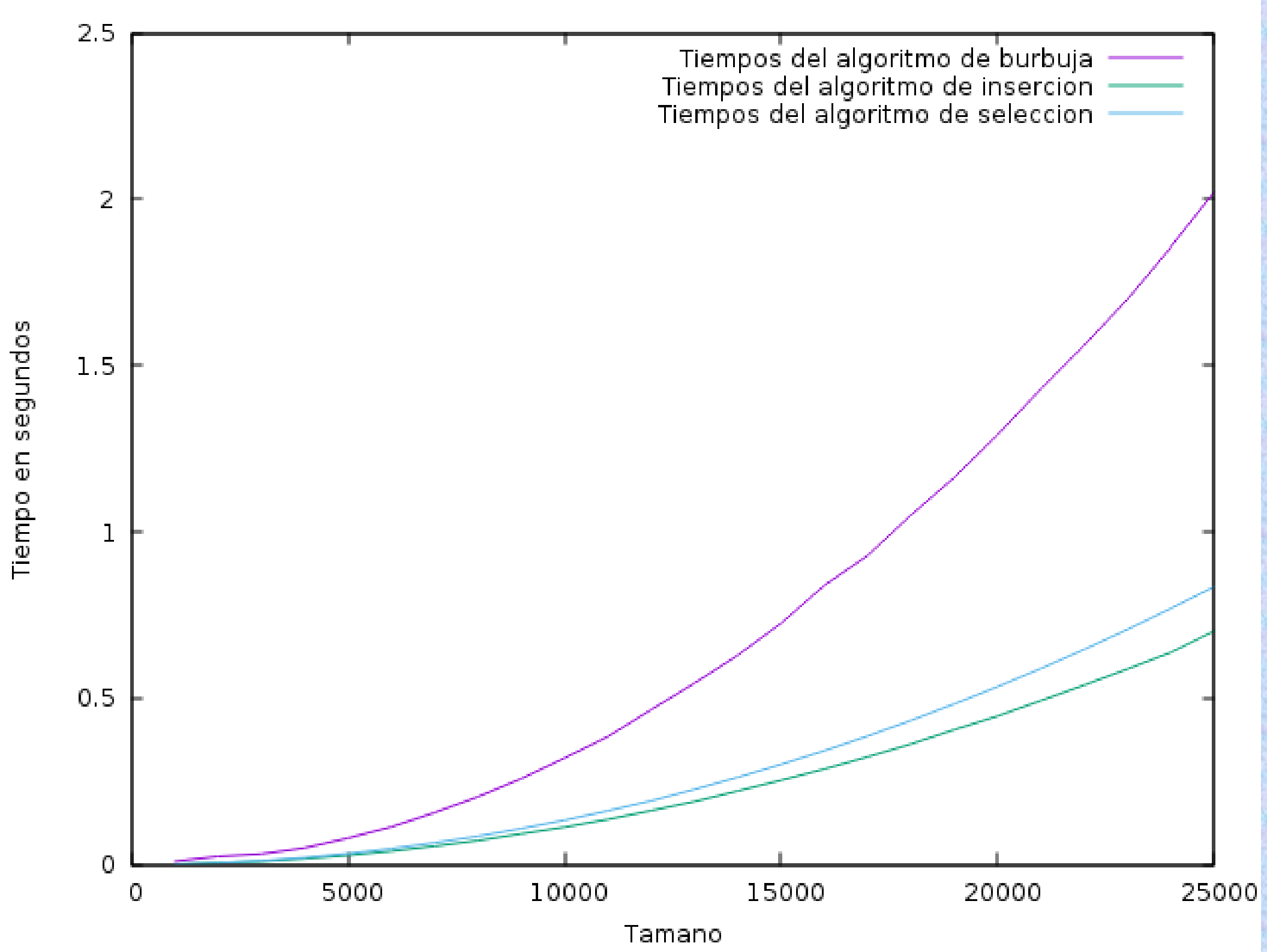


Algoritmos $O(n^2)$

- **Burbuja:** revisa cada elemento de la lista con el siguiente, es necesario revisar varias veces la lista.
- **Inserción:** Se compara cada elemento con todos los anteriores.
- **Selección:** Encuentra el menor de la lista y lo pone el primero, luego el segundo más pequeño y así sucesivamente.

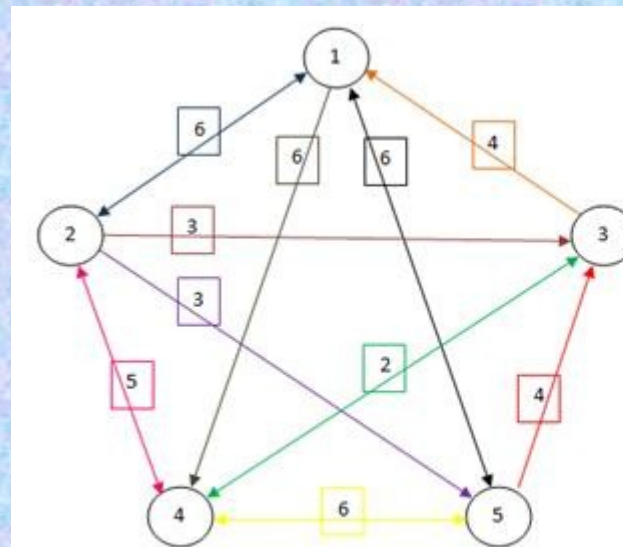
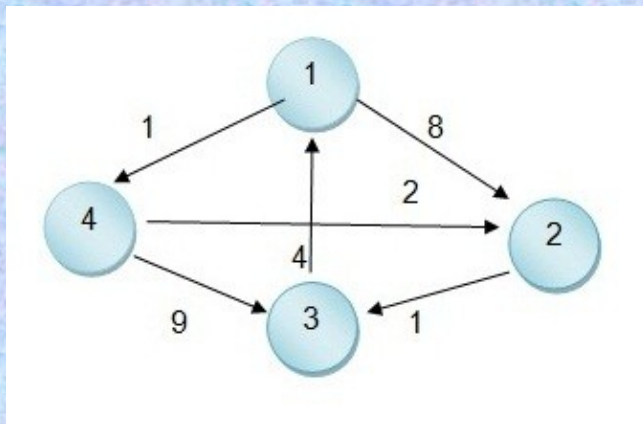
Tamaño	Burbuja	Selección	Inserción
1000	0.009918	0.00141602	0.001259
2000	0.024856	0.00542193	0.006108
3000	0.032287	0.012142	0.010358
4000	0.050257	0.02147	0.018124
5000	0.079186	0.033433	0.027975
6000	0.113281	0.048073	0.040309
7000	0.156463	0.065319	0.055311
8000	0.204092	0.085219	0.071996
9000	0.258562	0.108276	0.092417
10000	0.32005	0.133356	0.112756
11000	0.384988	0.161458	0.136253
12000	0.465959	0.192131	0.162078

Tamaño	Burbuja	Selección	Inserción
13000	0.545303	0.226143	0.189631
14000	0.629492	0.262543	0.22141
15000	0.725537	0.301038	0.253554
16000	0.839554	0.342276	0.287533
17000	0.929566	0.386806	0.324087
18000	1.05121	0.43345	0.362635
19000	1.16358	0.482941	0.405348
20000	1.2925	0.535169	0.446799
21000	1.43041	0.58983	0.493072
22000	1.56164	0.647236	0.539583
23000	1.70039	0.707328	0.588799
24000	1.85597	0.770895	0.638557
25000	2.01983	0.835793	0.702854



$$O(n^3)$$

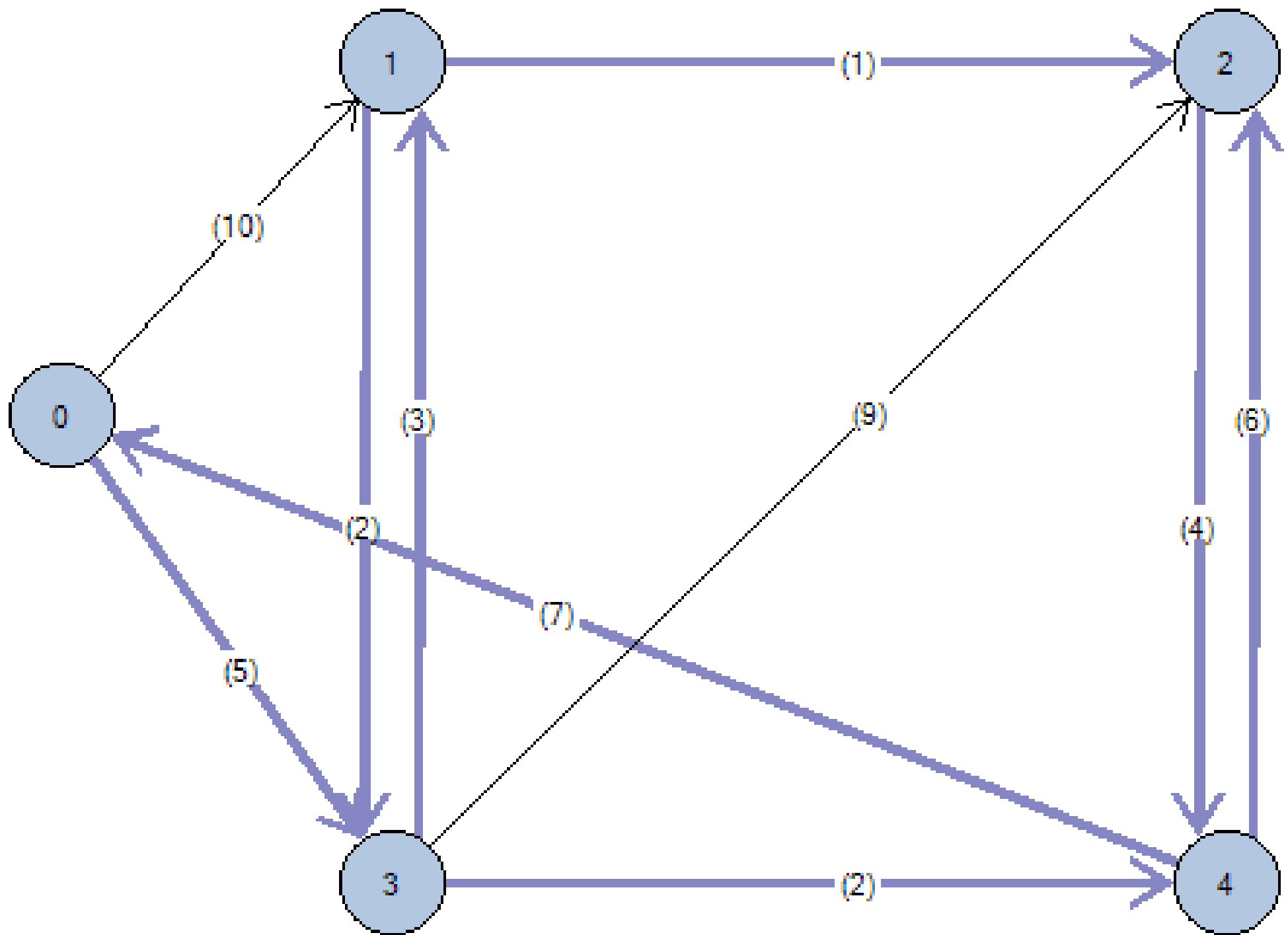
- **Floyd:** Determina el camino mínimo en grafos dirigidos ponderados. Es un ejemplo de programación dinámica.



Mejoras en los recorridos:

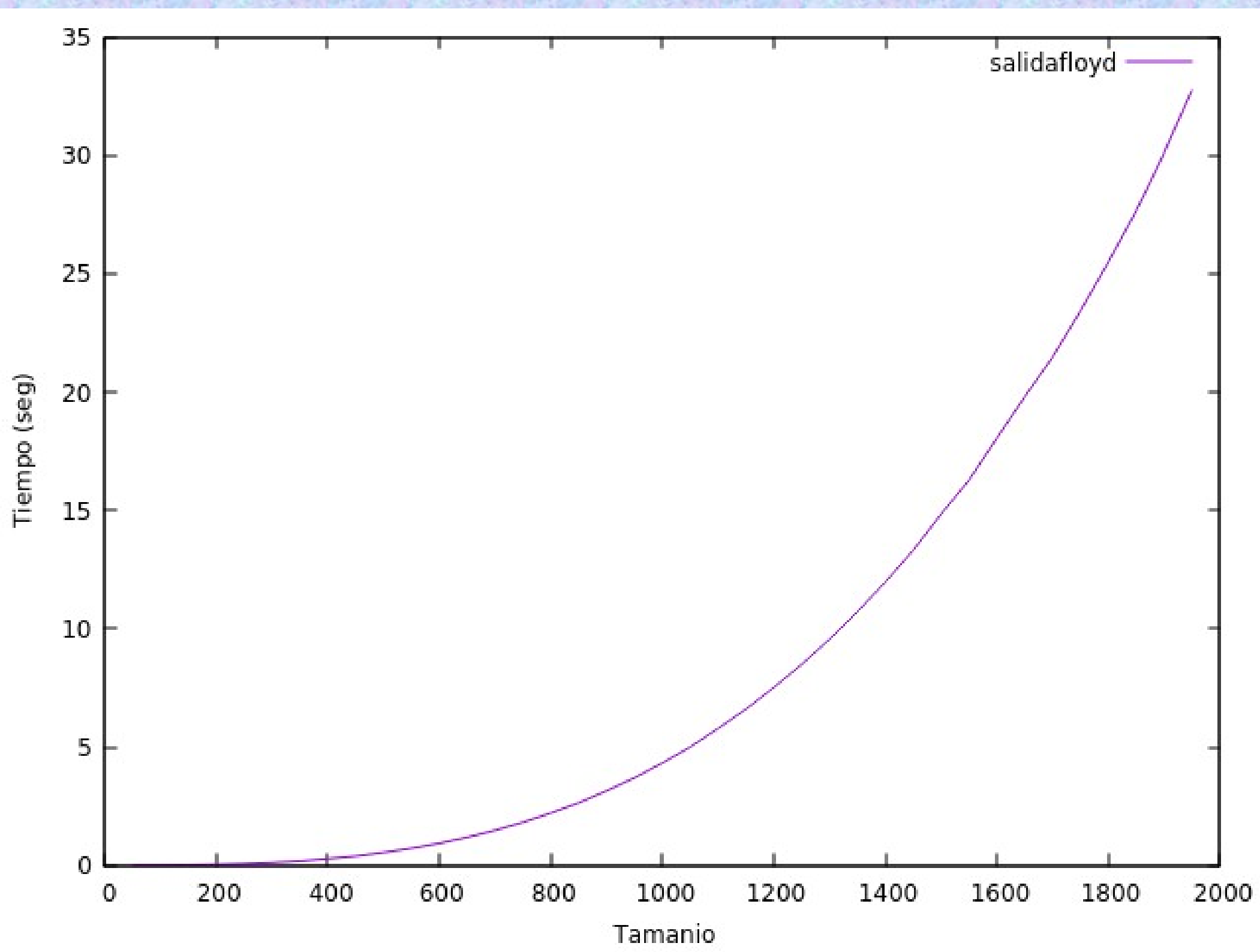
	Antes	Ahora
De 1 a 3	∞	8 por 4
De 3 a 2	∞	7 por 4
De 3 a 5	∞	8 por 4
De 4 a 1	∞	6 por 3
De 5 a 2	∞	11 por 4

De 1 a 3 se reduce a 8 km pasando por el nodo 4
 De 3 a 2 se reduce a 7 km pasando por el nodo 4
 De 3 a 5 se reduce a 8 km pasando por el nodo 4
 De 4 a 1 se reduce a 6 km pasando por el nodo 3
 De 5 a 2 se reduce a 11 km pasando por el nodo 4



Tamaño	Floyd
40	0.001656
80	0.011078
120	0.028261
160	0.034968
200	0.045051
240	0.071313
280	0.113485
320	0.167839
360	0.23765
400	0.324802
440	0.437196
480	0.559942
520	0.716048

Tamaño	Floyd
560	0.887893
600	1.09473
640	1.32449
680	1.5848
720	1.90991
760	2.21144
800	2.57918
840	2.98543
880	3.43734
920	3.92441
960	4.4626
1000	5.03838



Algoritmos $O(2^n)$

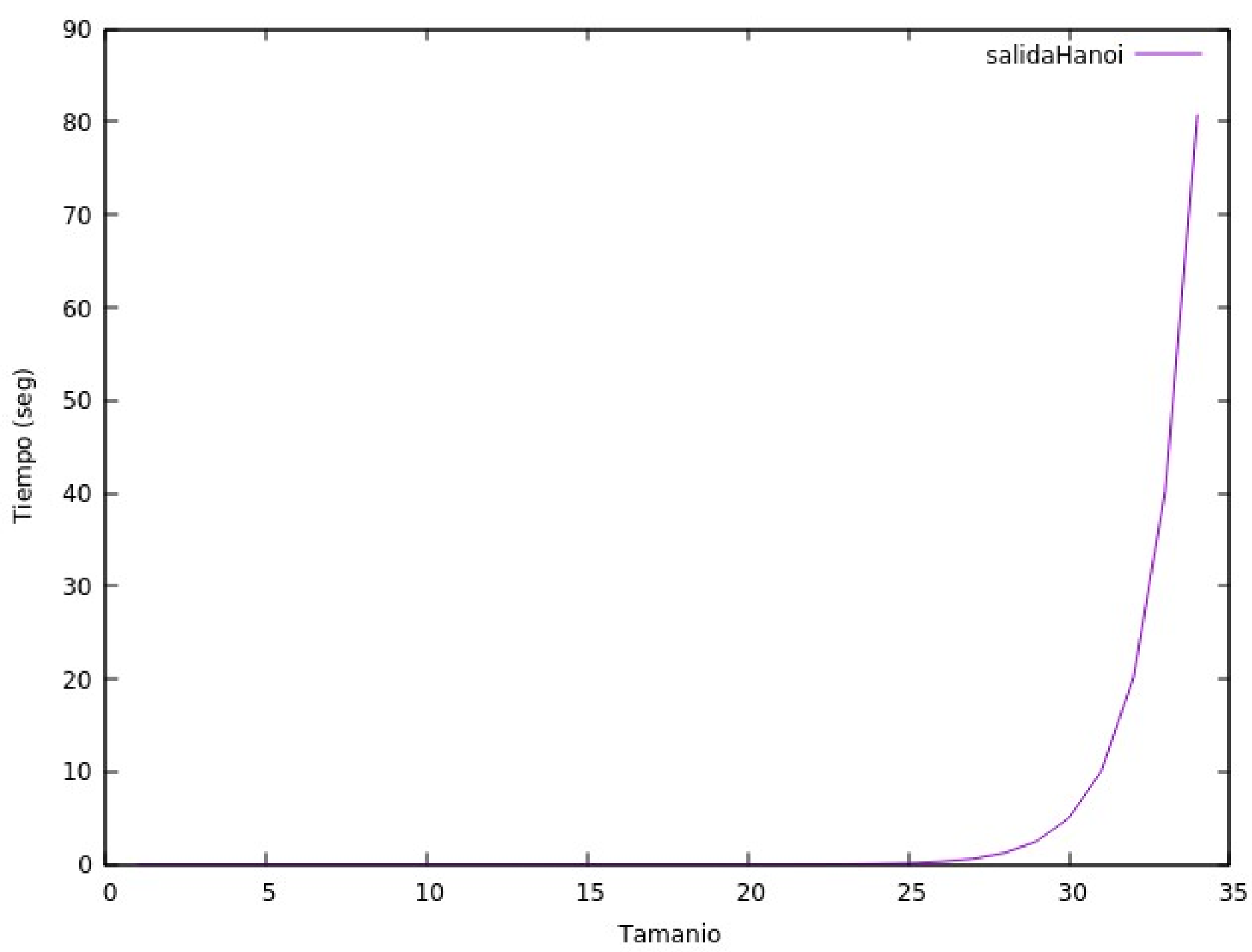
- **Hanoi:** Consiste en trasladar una serie de discos de una varilla a otra. Es un problema muy conocido en la ciencia de la computación y los movimientos mínimos necesarios para n discos es $2^n - 1$

Las Torres De Hanoi

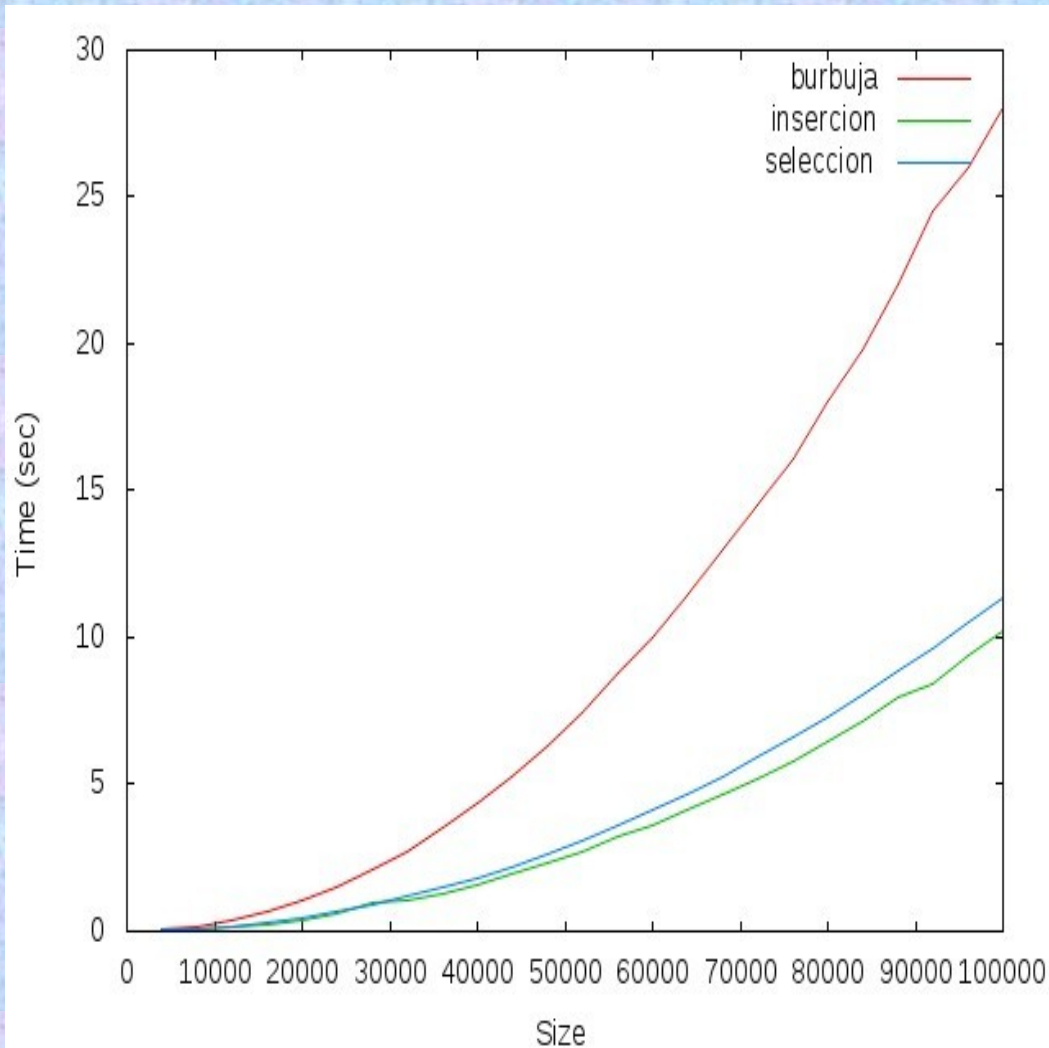


Tamaño	Hanoi
10	3e-06
11	6e-06
12	1.1e-05
13	2.1e-05
14	4.2e-05
15	8e-05
16	0.000158
17	0.000328
18	0.000629
19	0.001297
20	0.002567
21	0.005341
22	0.009641

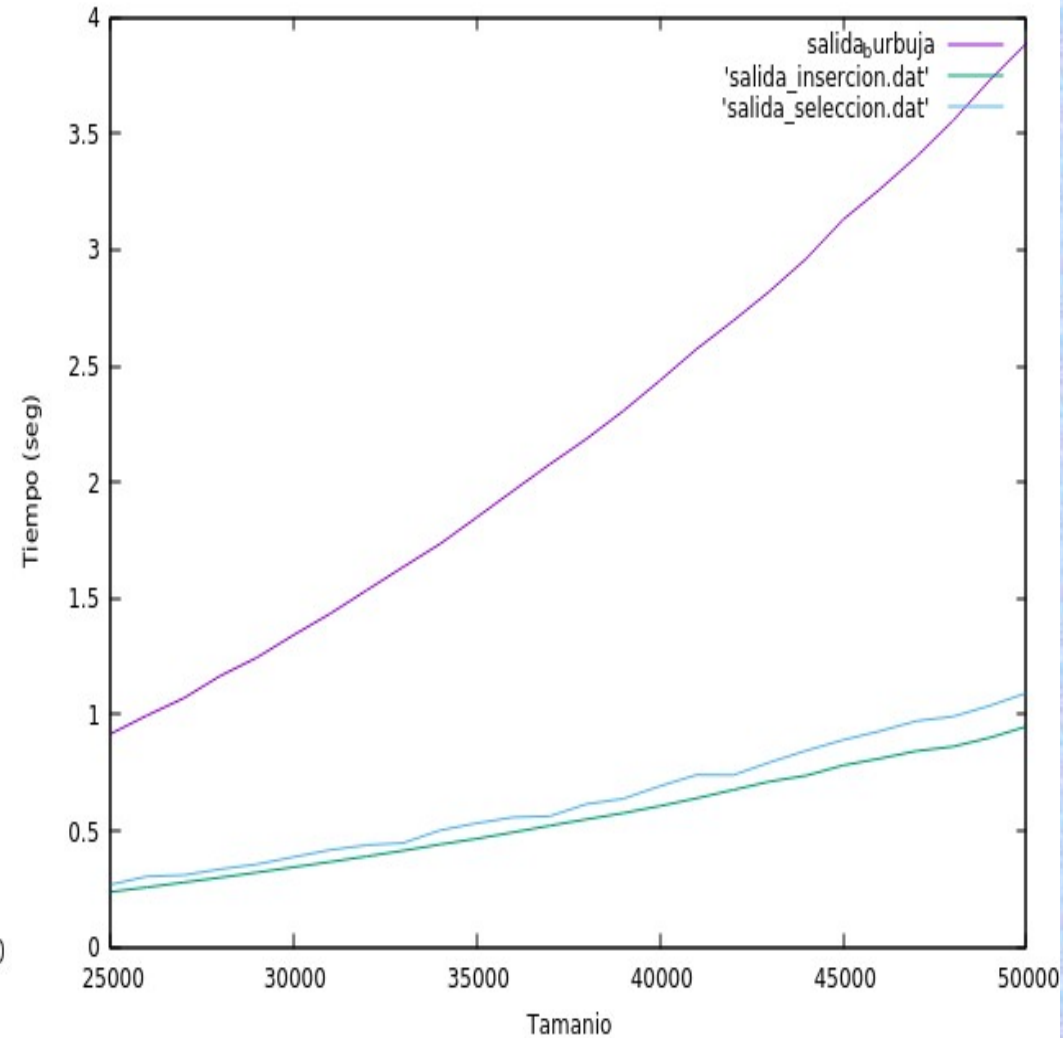
Tamaño	Hanoi
23	0.017469
24	0.034855
25	0.071372
26	0.141154
27	0.281754
28	0.553998
29	1.10018
30	2.20384
31	4.40097
32	8.8083
33	17.6036
34	35.1961
35	70.3265
36	140.648



Comparación de compilación

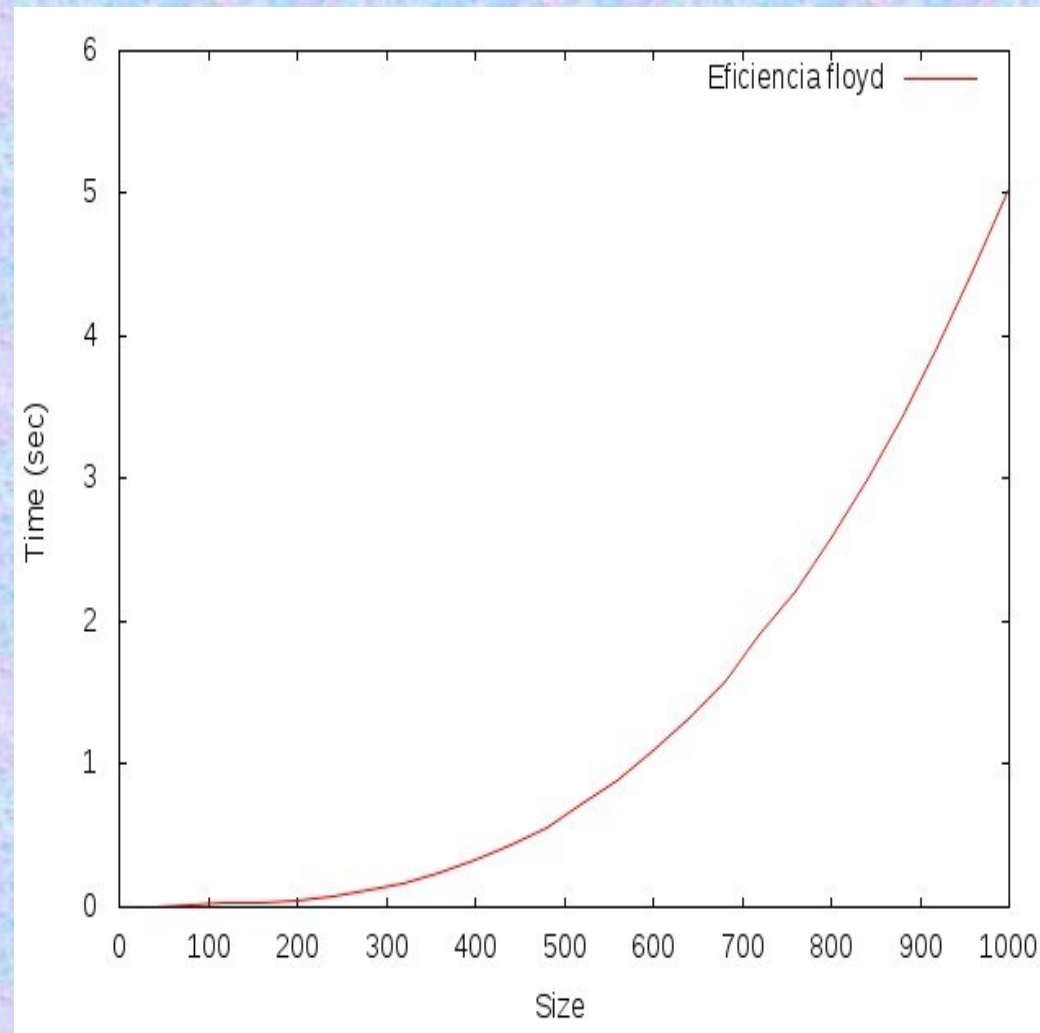


**Procesador: Intel(R) Core(TM) i7-4720HQ
CPU @ 2.60GHz x4(Código sin optimizar)**

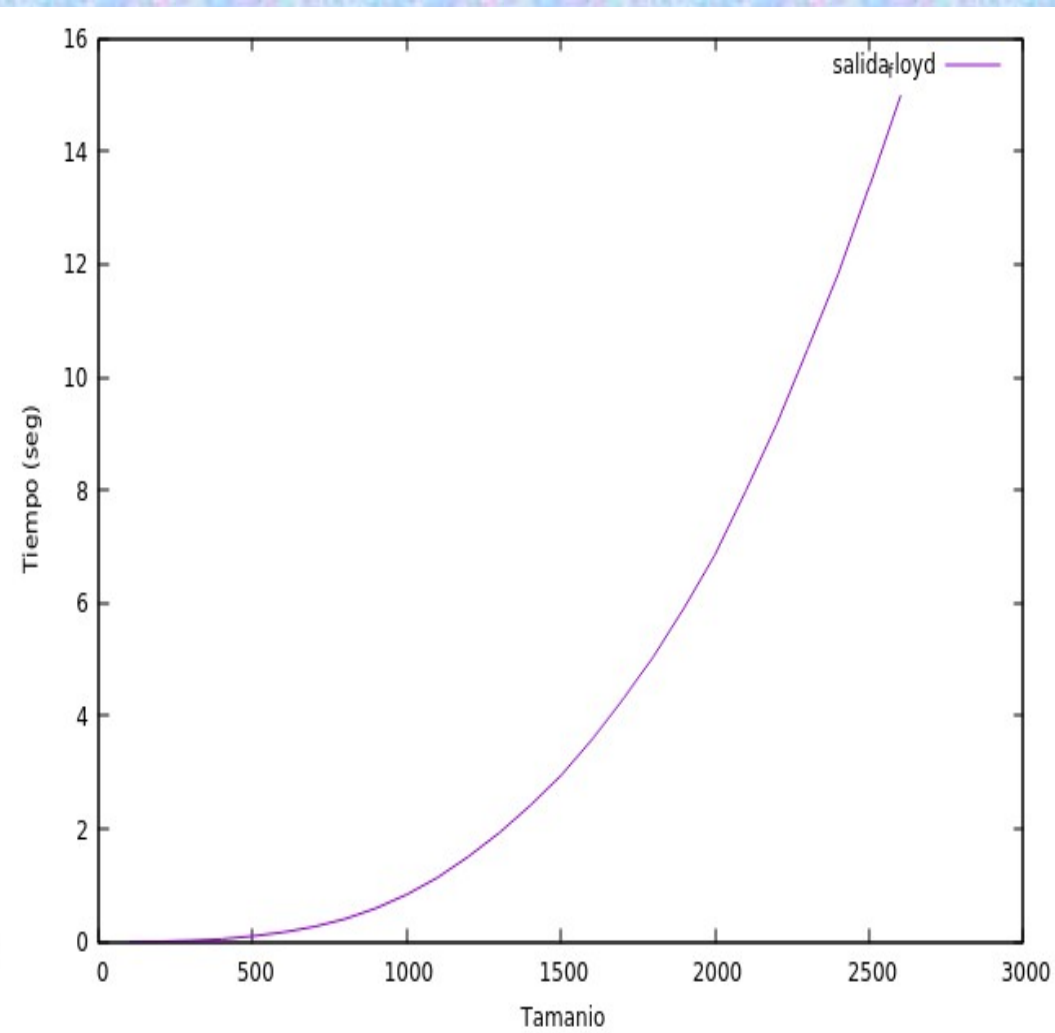


**Procesador: Intel Core i5-7200U
CPU@ 2.50 Ghz x4 (Optimización O2)**

Comparación de compilación



**Procesador: Intel(R) Core(TM) i7-4720HQ
CPU @ 2.60GHz x4(Código sin optimizar)**



**Procesador: Intel Core i5-7200U
CPU@ 2.50 Ghz x4 (Optimización O2)**

Comparación de compilación

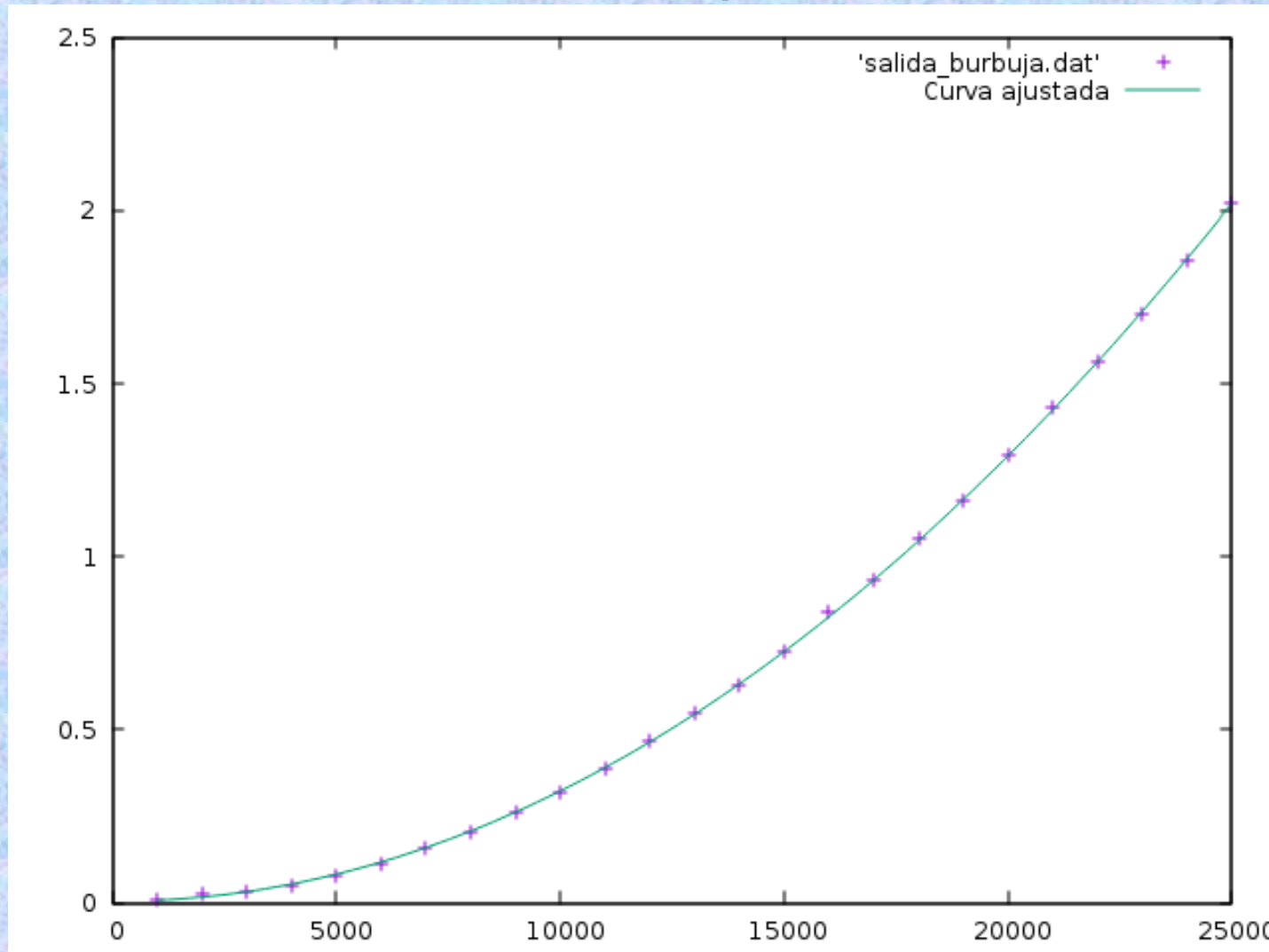
- Como podemos ver la optimización del código influye notablemente en la eficiencia de nuestros programas, así como el procesador de cada máquina, ya que un i5 de 7th generación supera al i7 de 4th.



Eficiencia híbrida

- Para calcular el tiempo exacto que tarda en ejecutarse un programa en un caso particular nos interesa tener una ecuación que nos de esta información. La eficiencia teórica nos da la expresión general, pero asociada a cada término de esta aparece una constante de valor desconocido, la cual se descubre ajustando la función por mínimos cuadrados al conjunto de datos.

Burbuja



Coeficientes de regresión:

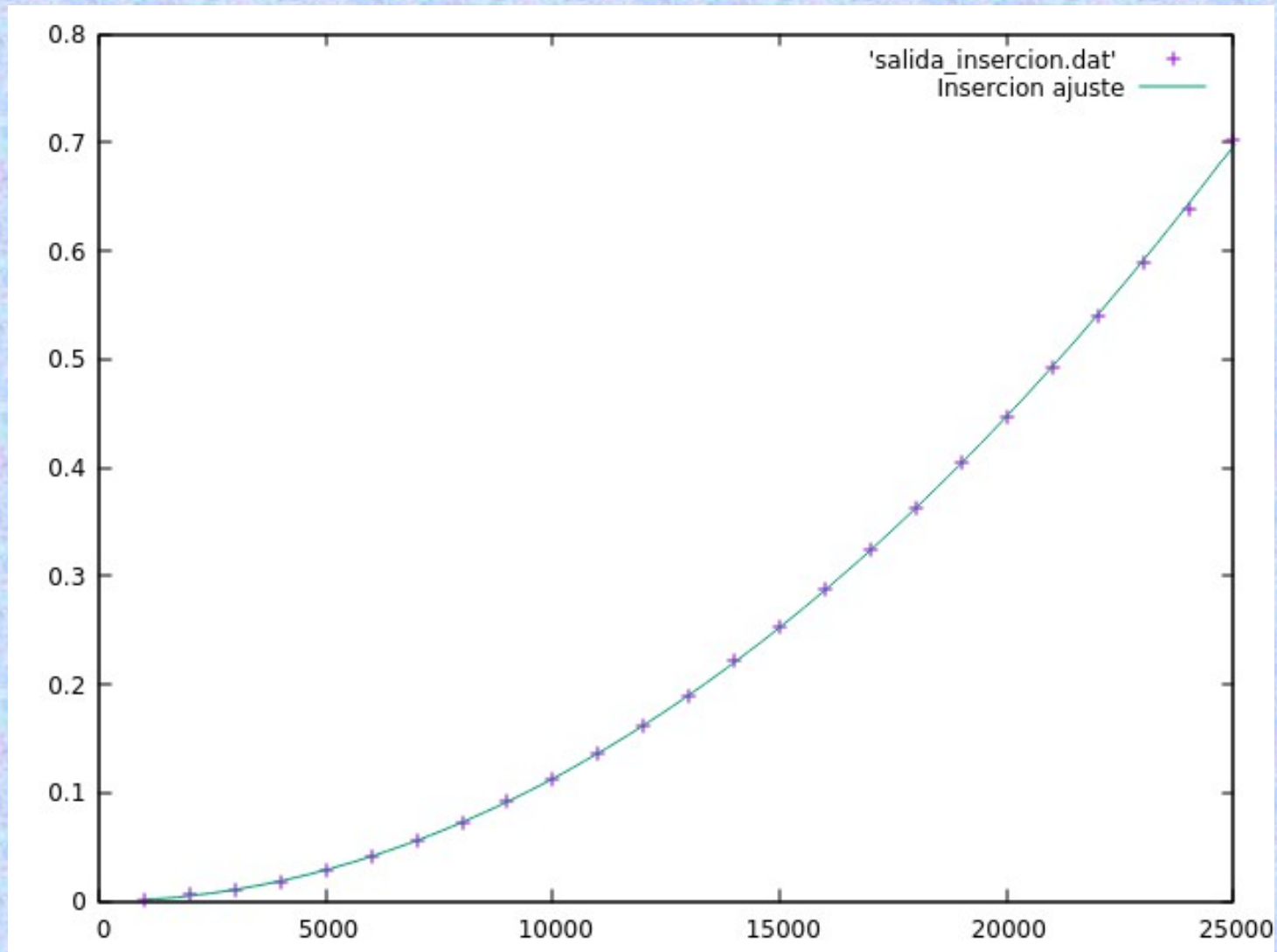
$$T(n)=a0*n^2 +a1*n +a2$$

$$a0 = 3.25431e-09$$

$$a1 = -7.85455e-07$$

$$a2 = 0.00480453$$

Inserción



$$T(n)=a0*n^2 +a1*n +a2$$

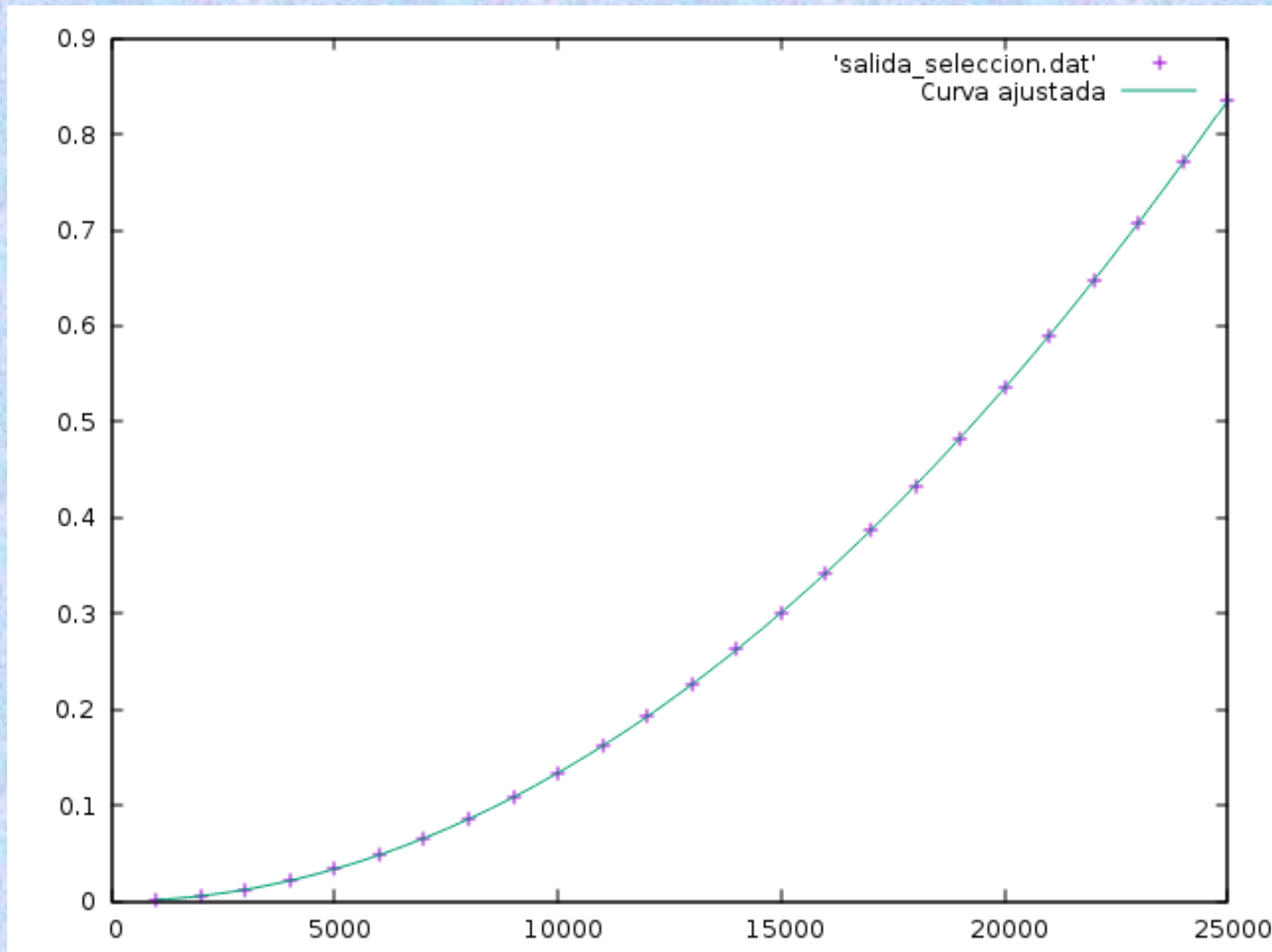
Coeficientes de regresión:

$$a0 = 1.10634e-09$$

$$a1 = 2.4927e-07$$

$$a2 = -0.000188397$$

Selección



$$T(n)=a0*n^2 +a1*n +a2$$

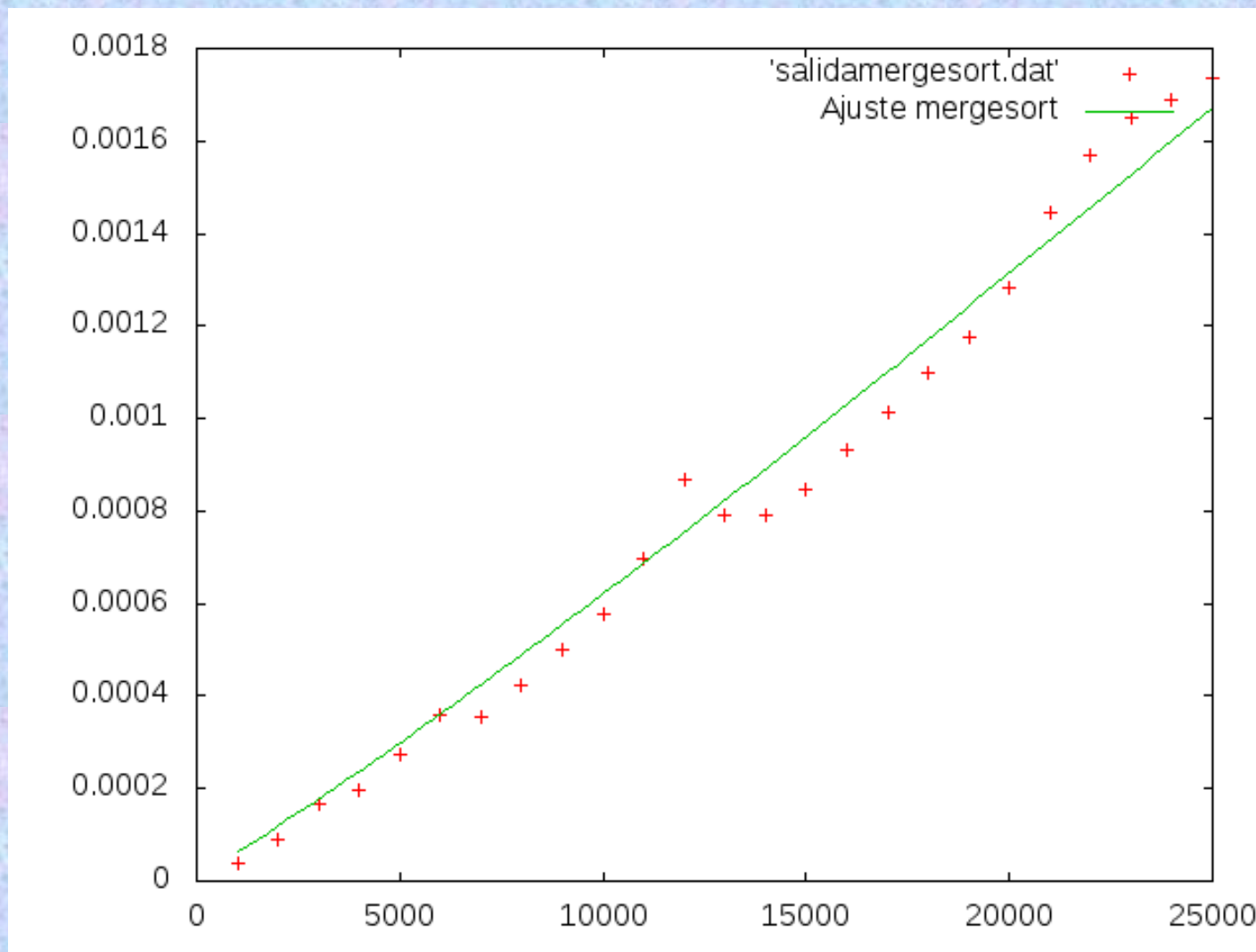
Coeficientes de regresión:

$$a0 = 1.33866e-09$$

$$a1 = -2.46031e-08$$

$$a2 = 4.2861e-05$$

Mergesort



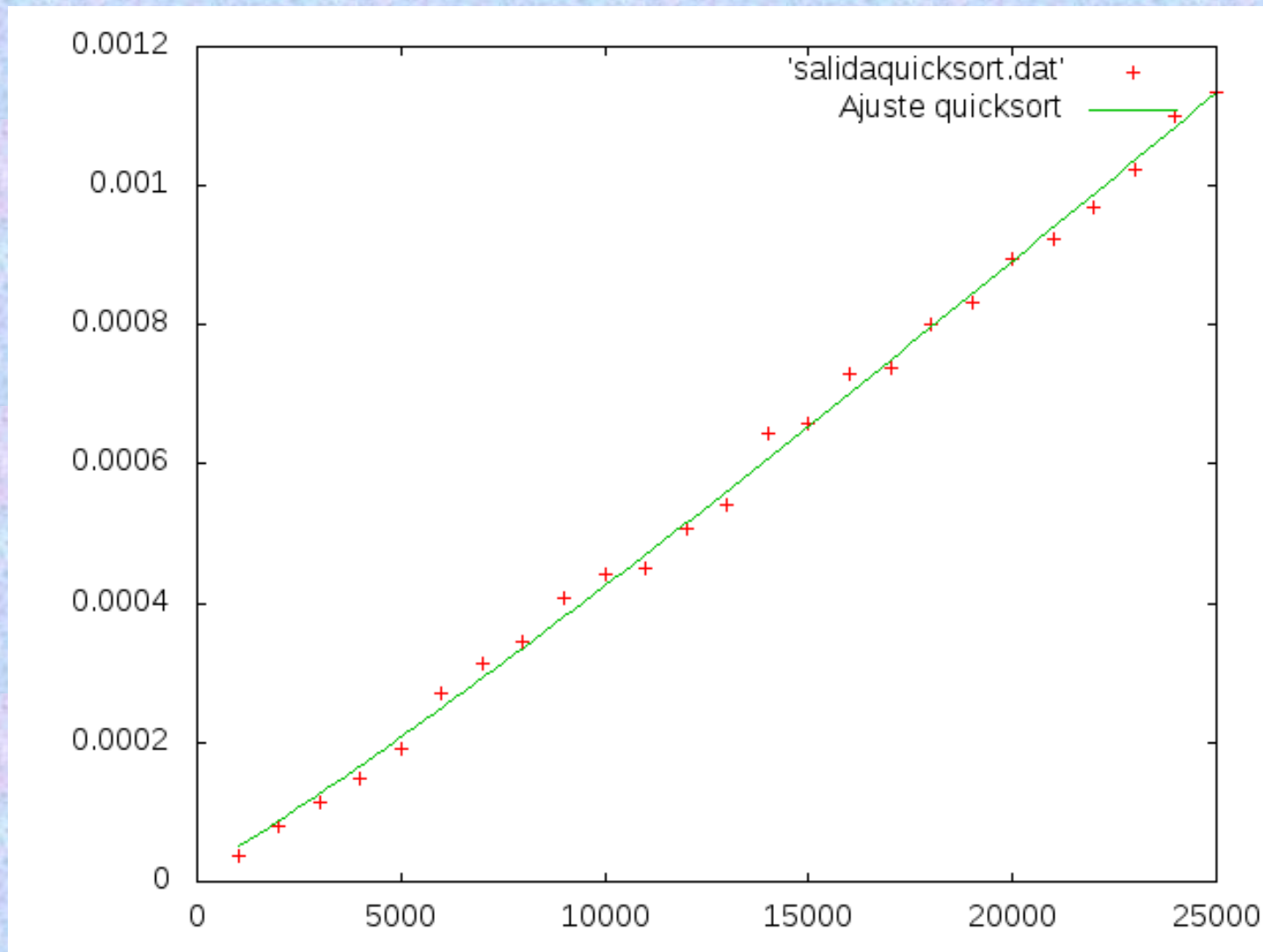
$$T(n) = a_0 * n * \log(n) + a_1$$

Coeficientes de regresión:

$a_0 = 4.40174e-09$

$a_1 = 1.98678e-05$

Quicksort



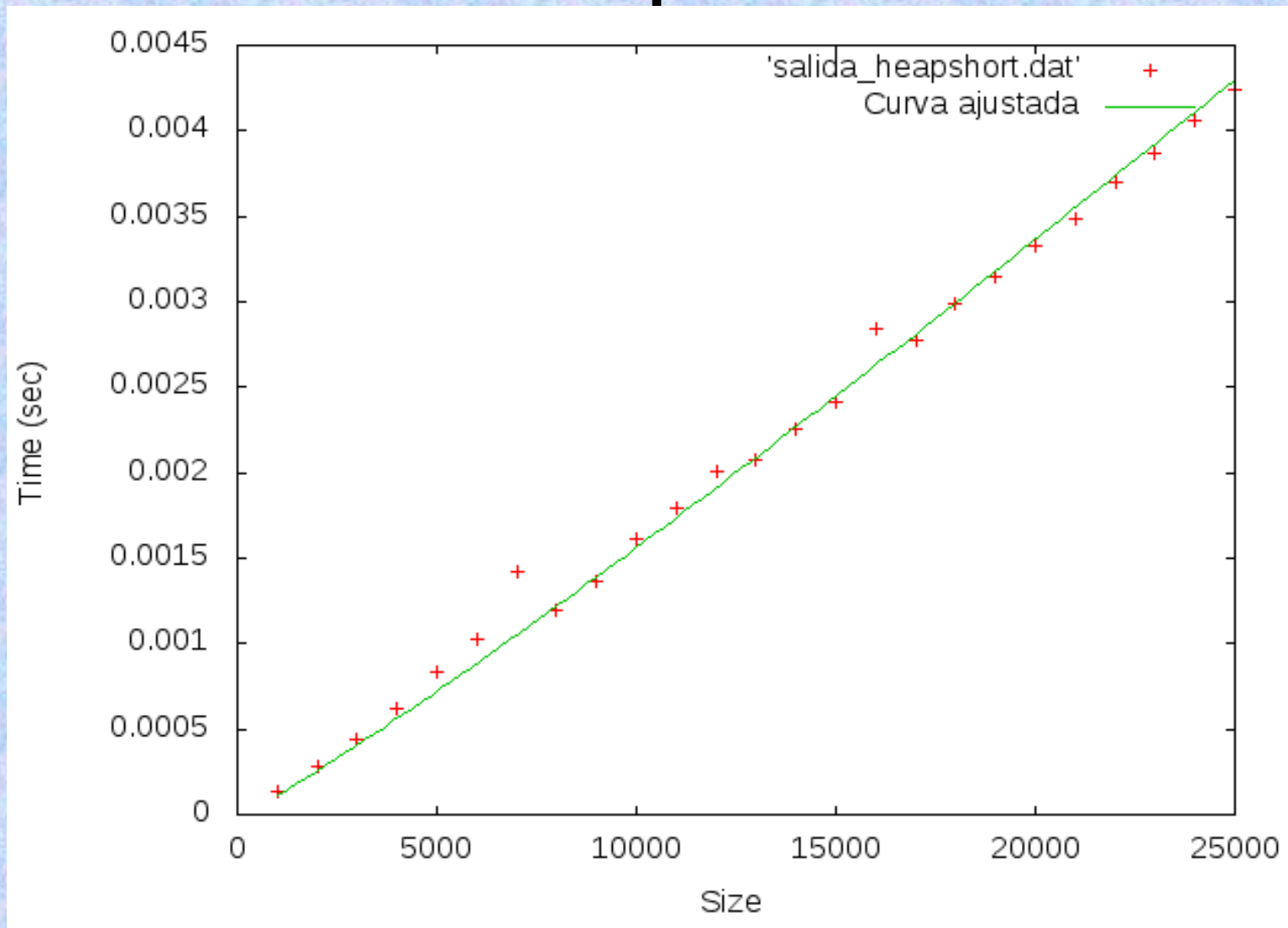
$$T(n) = a_0 * n * \log(n) + a_1$$

Coefficientes de regresión:

$a_0 = 1.03034e-09$

$a_1 = 0.373644$

Heapsort



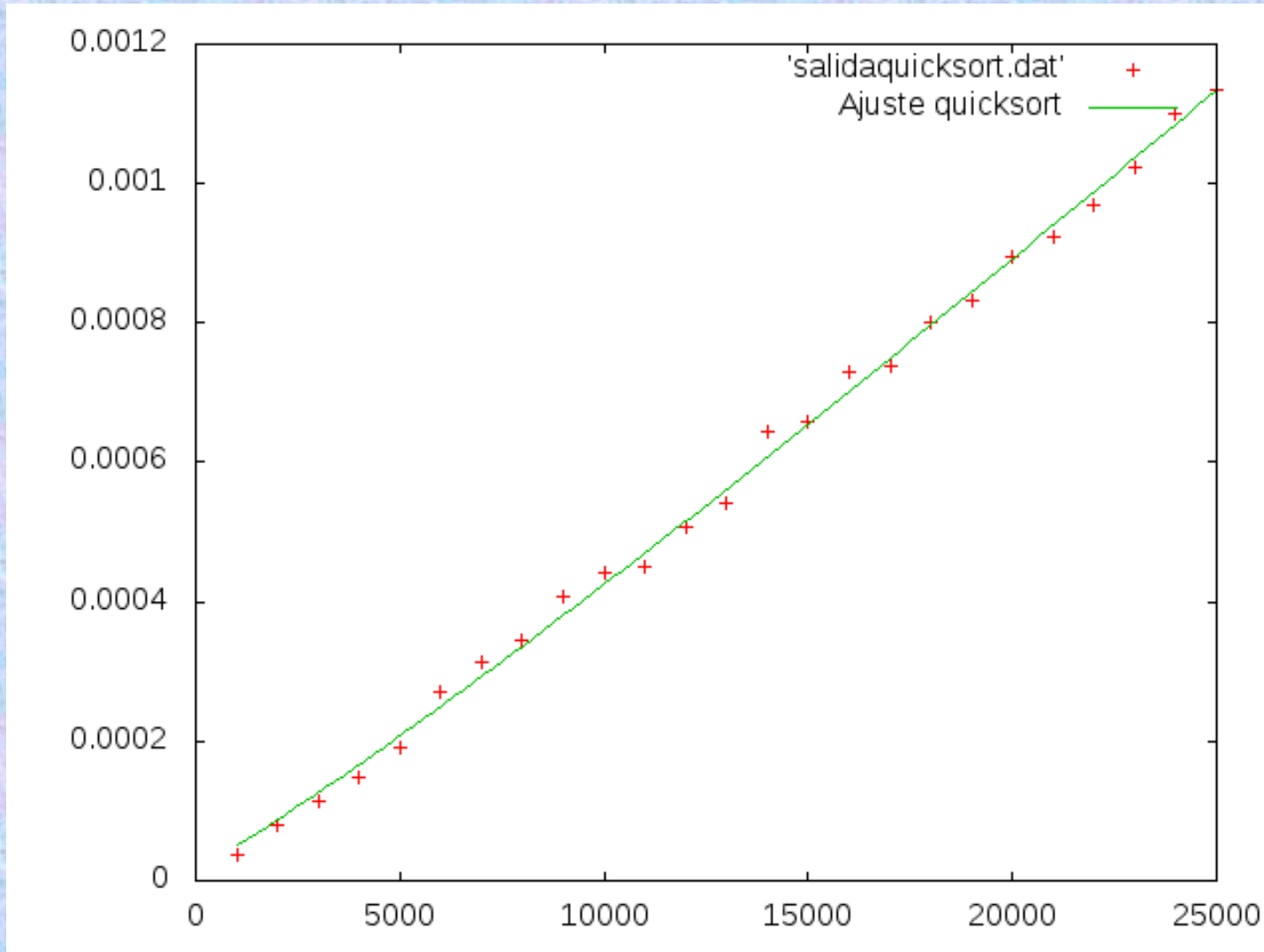
$$T(n)=a_0*n*\log(n) +a_1$$

Coeficientes de regresión:

$$a_0 = 6.5289e-09$$

$$a_1 = 1.98674e-05$$

Quicksort



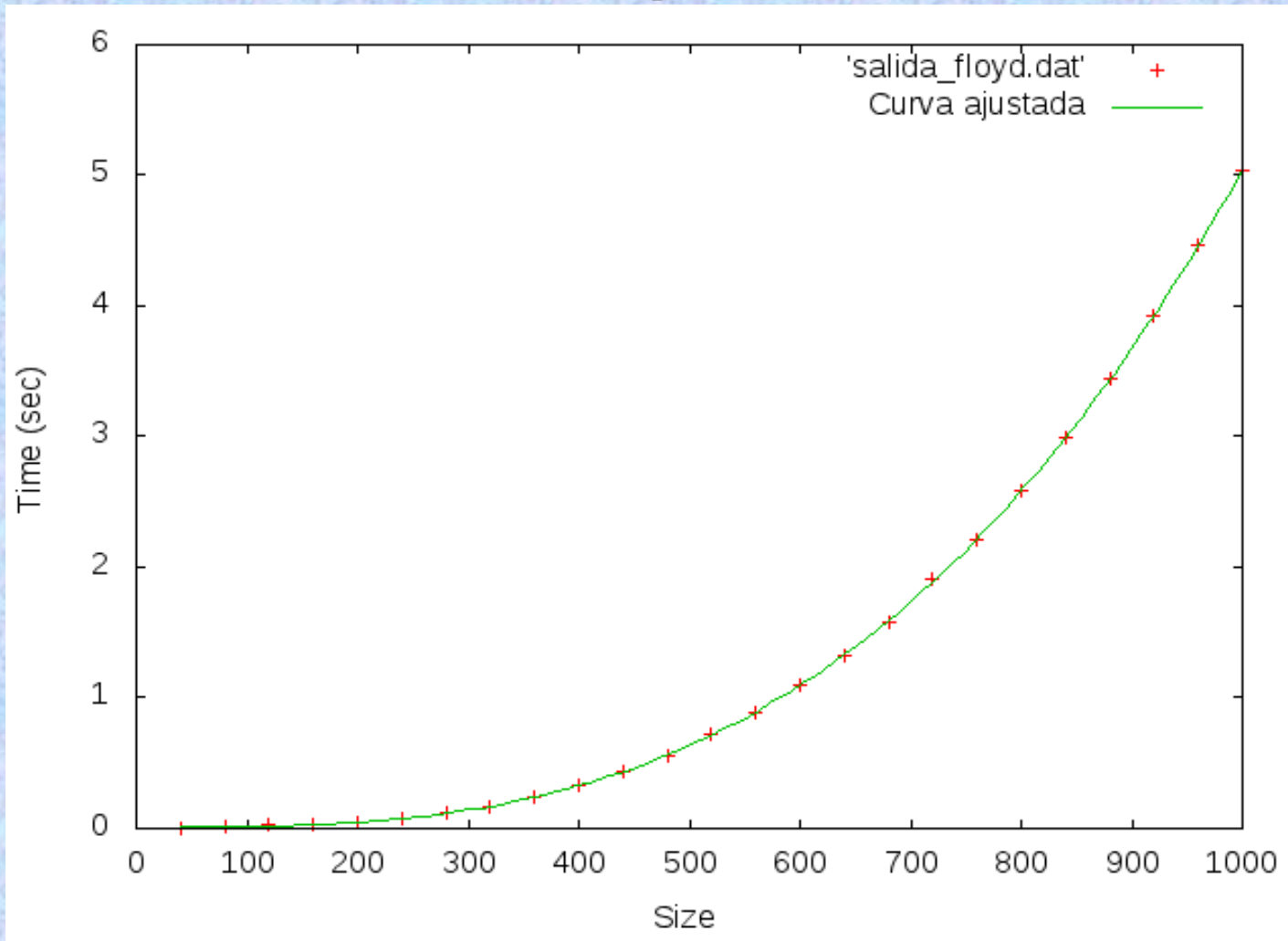
$$T(n) = a_0 * n * \log(n) + a_1$$

Coeficientes de regresión:

$$a_0 = 6.5289e-09$$

$$a_1 = 1.98674e-05$$

Floyd



$$T(n) = a_0 \cdot n^3 + a_1 \cdot n^2 + a_2 \cdot n + a_3$$

Coeficientes de regresión:

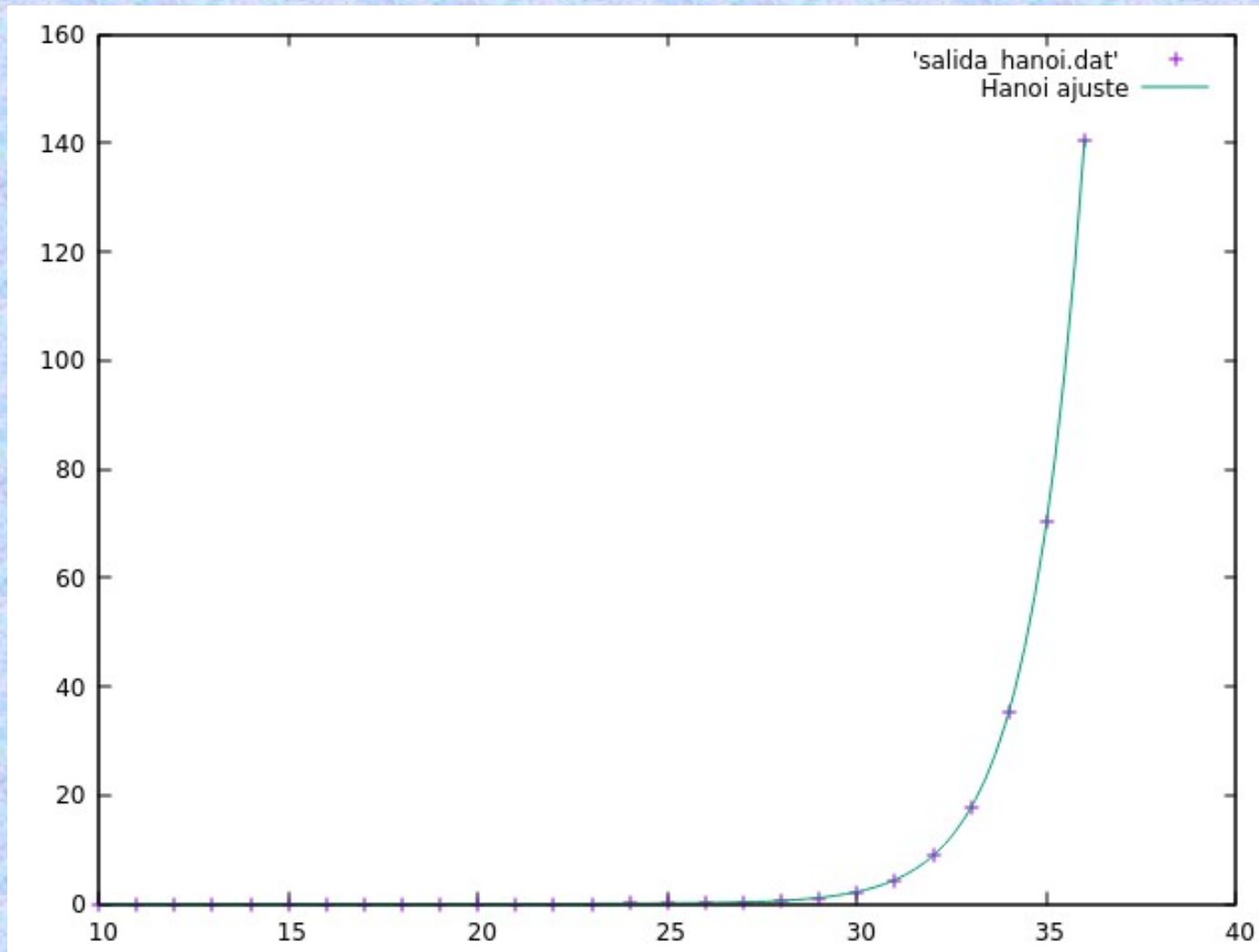
$$a_0 = 4.98387e-09$$

$$a_1 = 8.4389e-08$$

$$a_2 = -4.09129e-05$$

$$a_3 = 0.0113909$$

Hanoi



$$T(n) = a_0 \cdot 2^n + a_1$$

Coeficientes de regresión:

$$a_0 = 0.999508$$

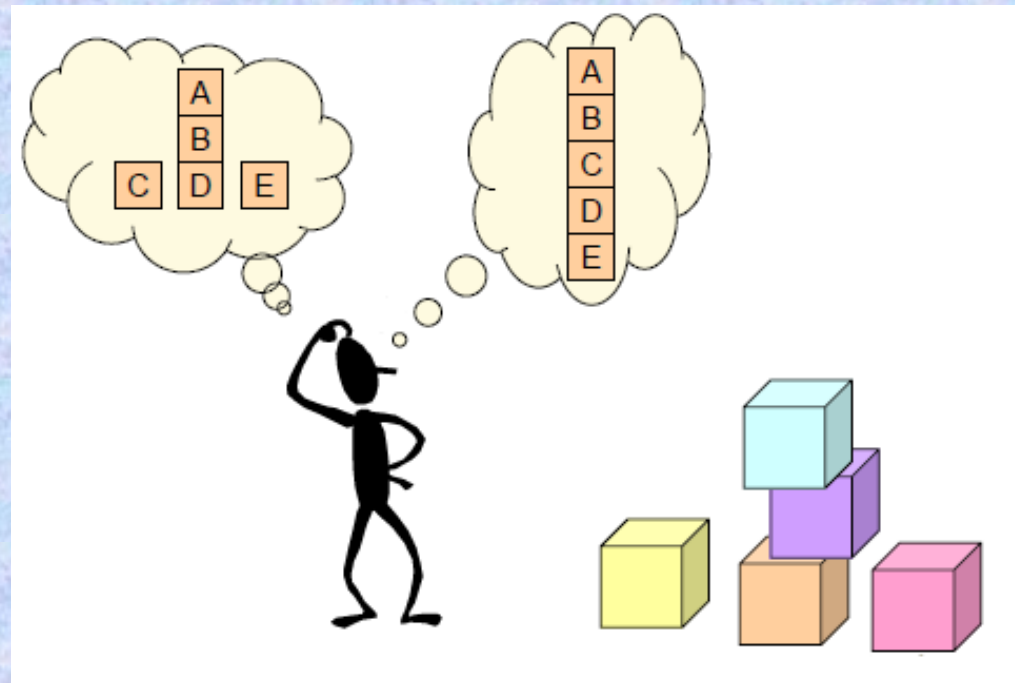
$$a_1 = -28.8464$$

Eficiencia Híbrida

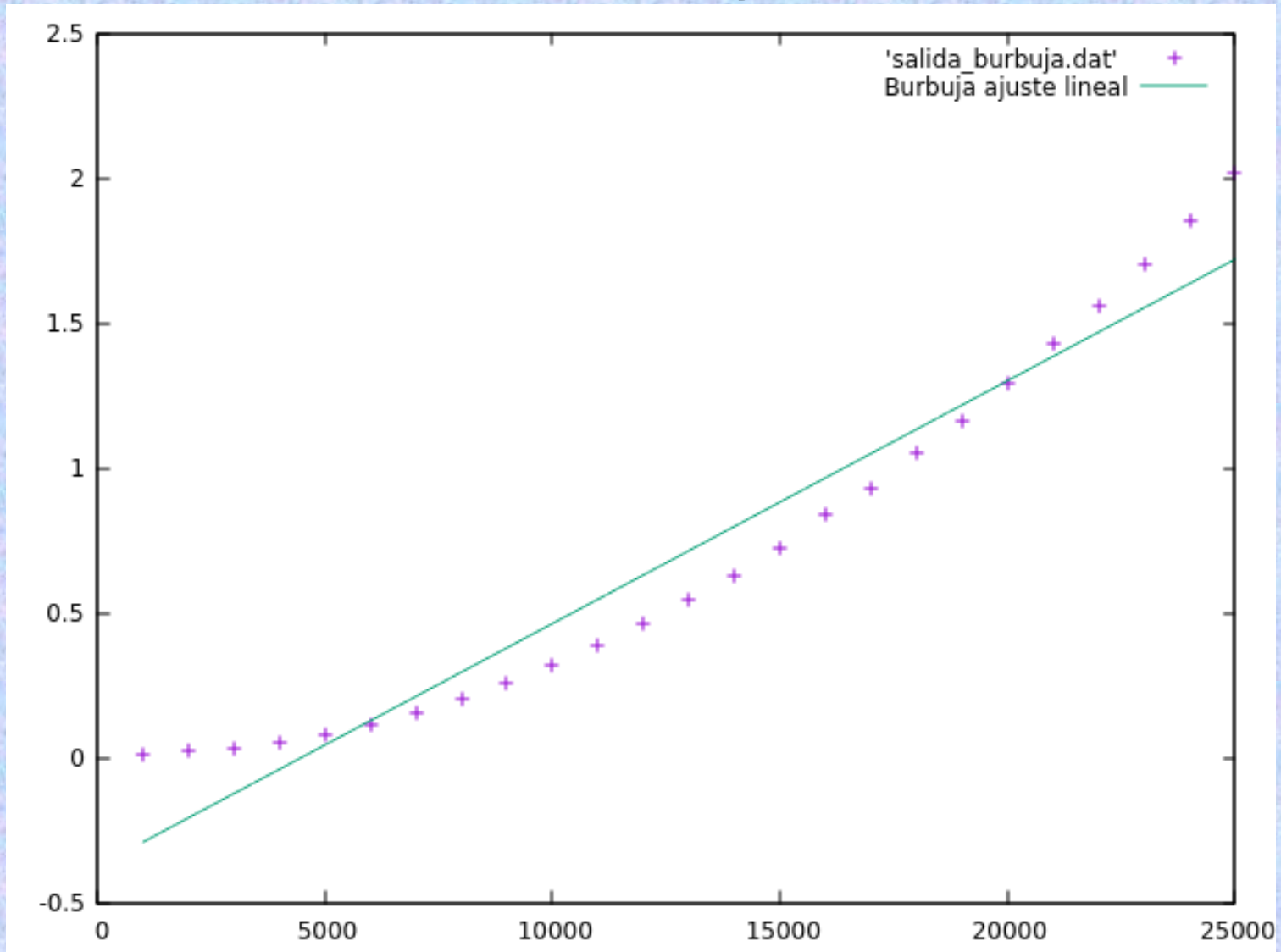
- Como podemos ver las funciones calculadas a partir de la eficiencia teórica se ajustan bastante bien a los datos y con las constantes ocultas ya encontradas se puede hacer una predicción bastante buena del tiempo de ejecución según el tamaño de la entrada de datos.

Eficiencia Híbrida

- Ahora vemos algunos ejemplos de ajustes de funciones que no corresponden con la eficiencia teórica del algoritmo.

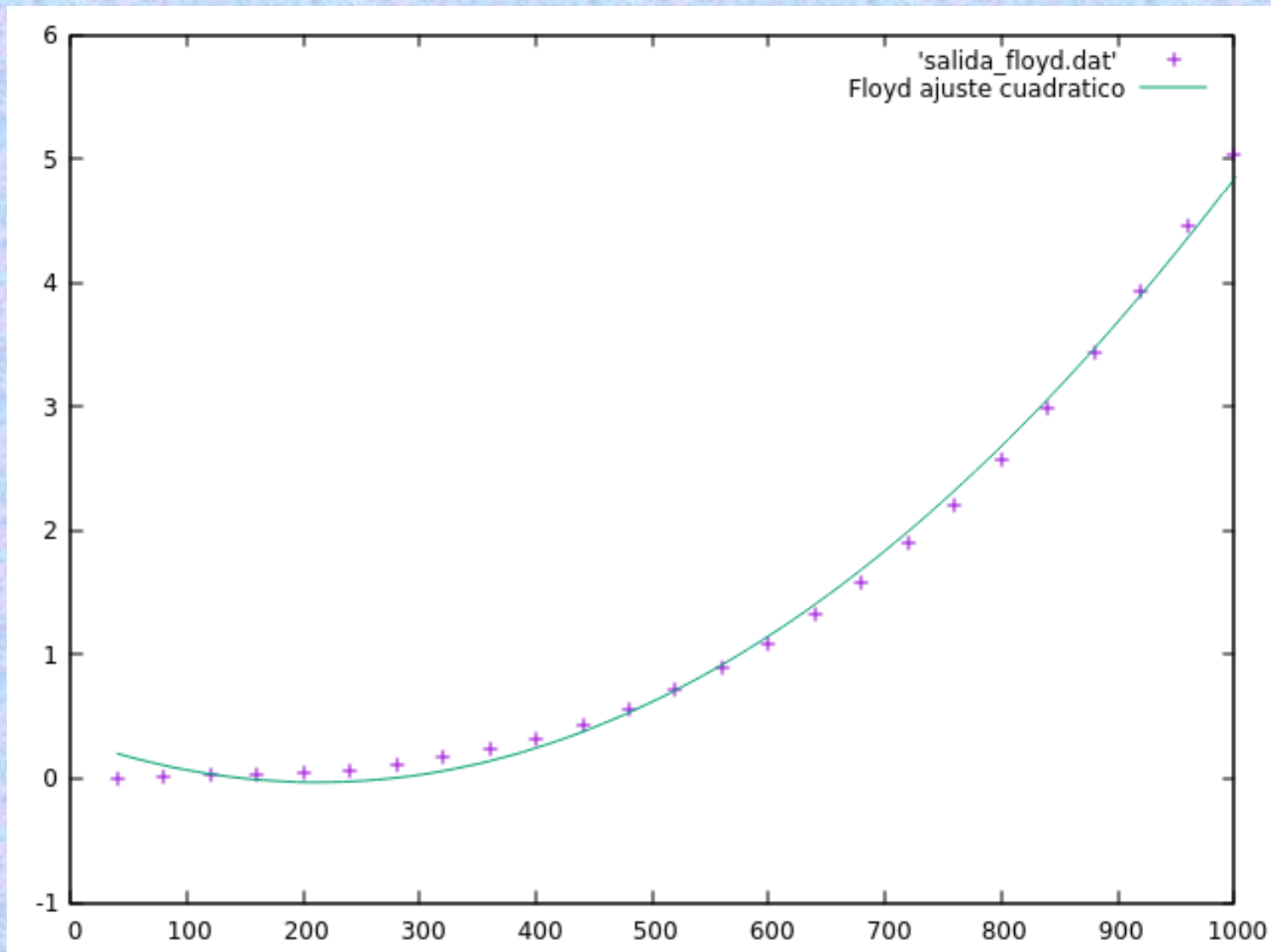


Burbuja



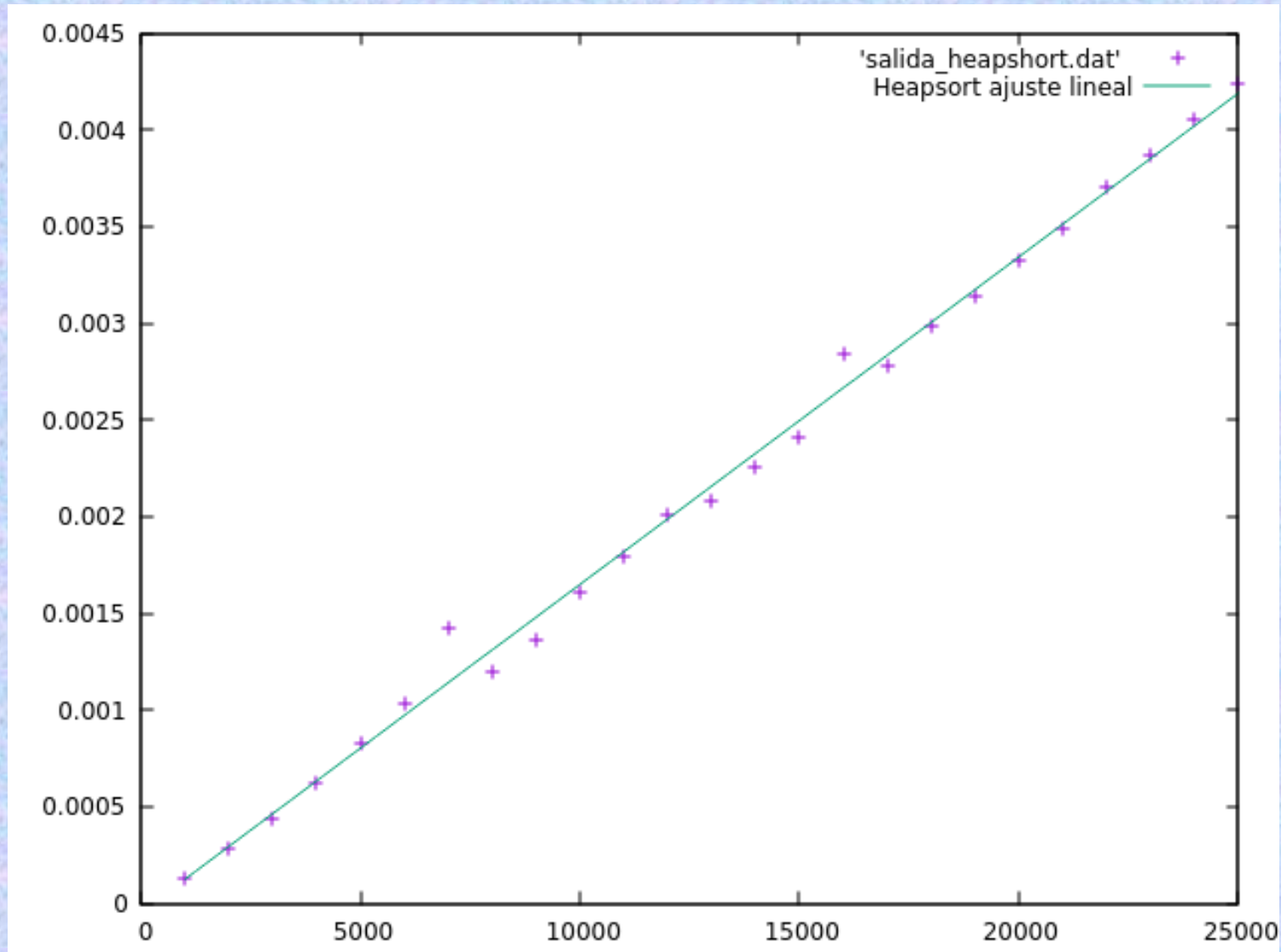
$$T(n) = a_0 * n + a_1$$

Floyd



$$T(n) = a_0 \cdot n^2 + a_1 \cdot n + a_2$$

Heapsort



$$T(n) = a_0 * n + a_1$$

Eficiencia Híbrida

- En general se ve que el ajuste de estas funciones a los datos es peor que el de las funciones que la eficiencia teórica nos da por lo que la aproximación que las funciones ajustadas nos dan del tiempo de ejecución según el tamaño de la entrada de datos será peor.