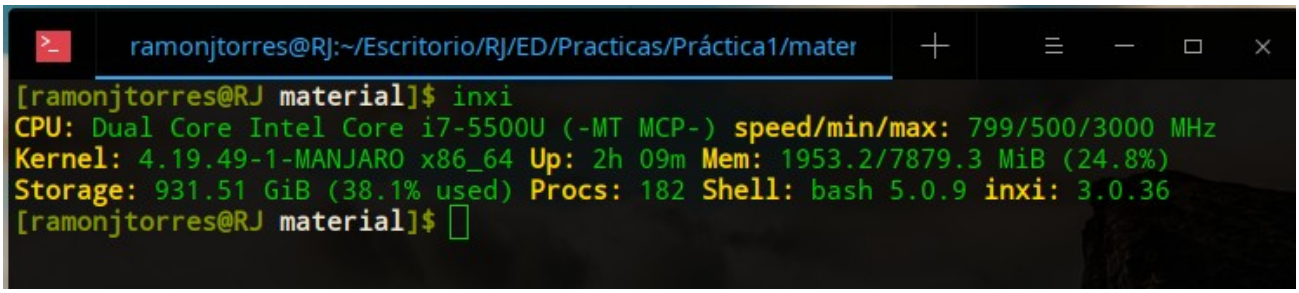
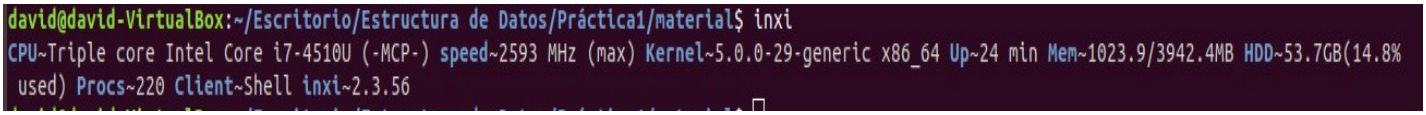
Práctica 1

Ramón Jesús Torres Madrid
David López Maldonado

Características del PC.





Ejercicio 1.

Eficiencia teórica:

Nos encontramos ante el siguiente código:

```
void ordenar(int *v, int n) {

  for (int i=0; i<n-1; i++)
    for (int j=0; j<n-i-1; j++)
      if (v[j]>v[j+1]) {   3OE← O(1)
           int aux = v[j];  2OE← O(1)
      v[j] = v[j+1]; 3OE← O(1)
      v[j+1] = aux; 2OE← O(1)
      }
}
```

Ahora nos centraremos en el **for** más interno:

$$\sum_{j=0}^{n-i-1} O(1) = \sum_{j=0}^{n-i-1} 1$$

Si tenemos en cuenta lo visto en clase, tendremos lo siguiente:

n-i-1-0+1 = n-i

Es decir:

$$\sum_{j=0}^{n-i-1} 1 = n-i$$

Para terminar, nos centramos en el **for** más externo:

$$\sum_{i=0}^{n-1} n-i = \sum_{i=0}^{n-1} n + \sum_{i=0}^{n-1} i = n \cdot \sum_{i=0}^{n-1} 1 - \sum_{i=0}^{n-1} i$$

Por un lado tenemos:

n-1-0+1 = n

$$n \cdot \sum_{i=0}^{n-1} 1 = n \cdot n = n^2$$

Y por el otro una progresión aritmética:

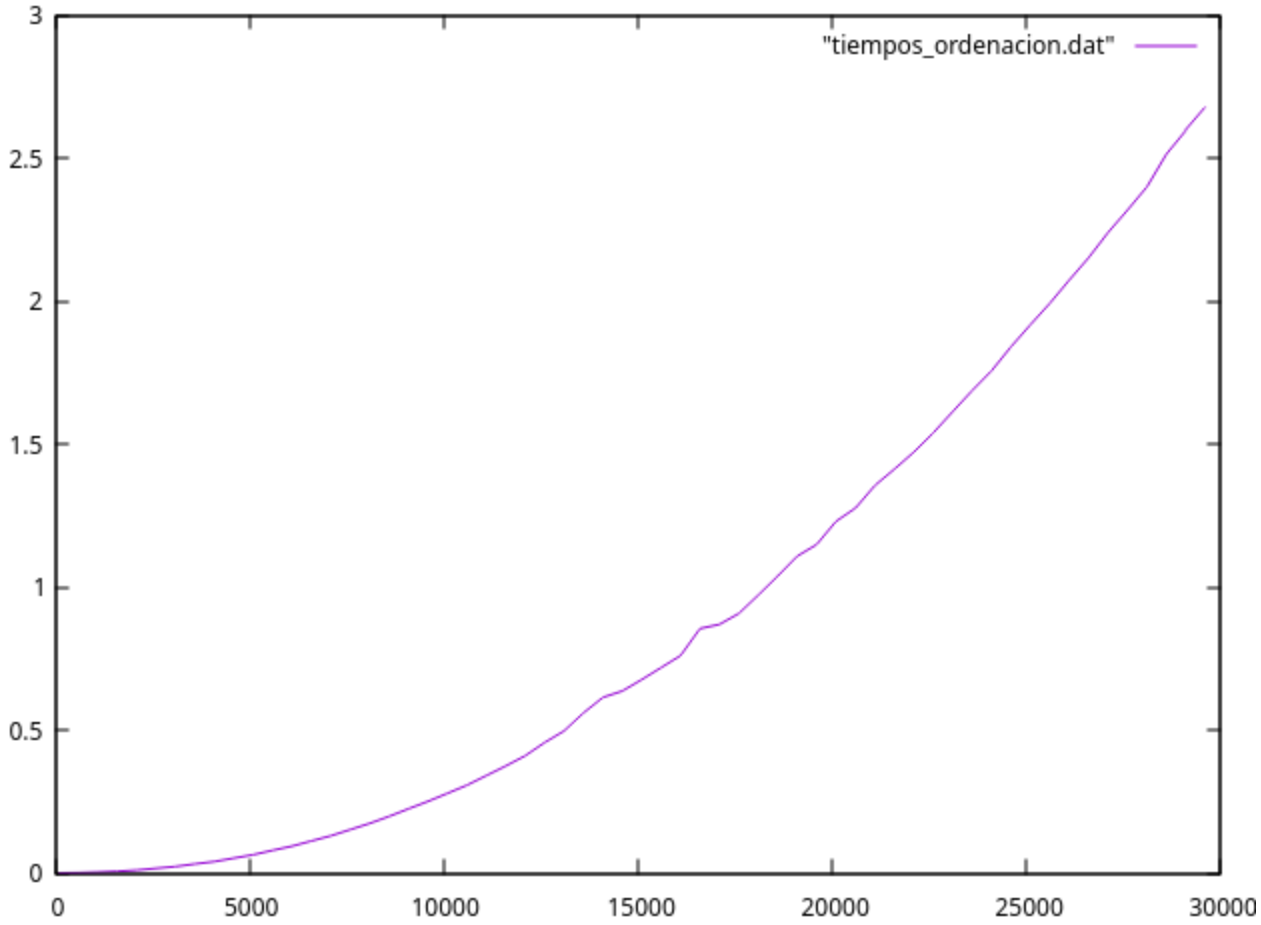
Como vimos en clase, se iban haciendo parejas desde el principio y final, asique:

$$\sum_{i=0}^{n-1} i = \frac{n}{2} \cdot (n-1) = \frac{n \cdot (n-1)}{2}$$

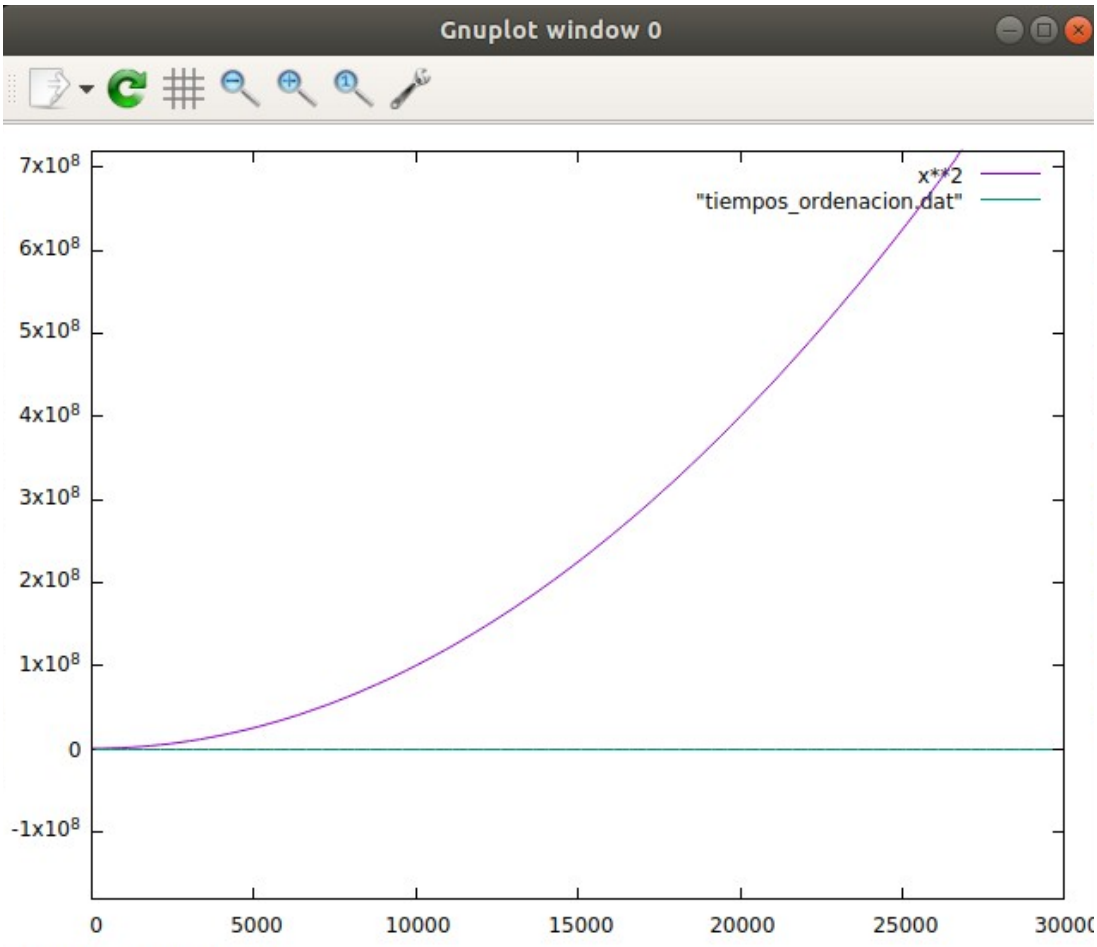
Asique finalmente obtenemos:

$$n \cdot \sum_{i=0}^{n-1} 1 + \sum_{i=0}^{n-1} i = n^2 - \frac{n \cdot (n-1)}{2} = n^2 - \frac{n^2}{2} + \frac{n}{2} = \frac{n^2}{2} + \frac{n}{2} \in O(n^2)$$

Eficiencia empírica:

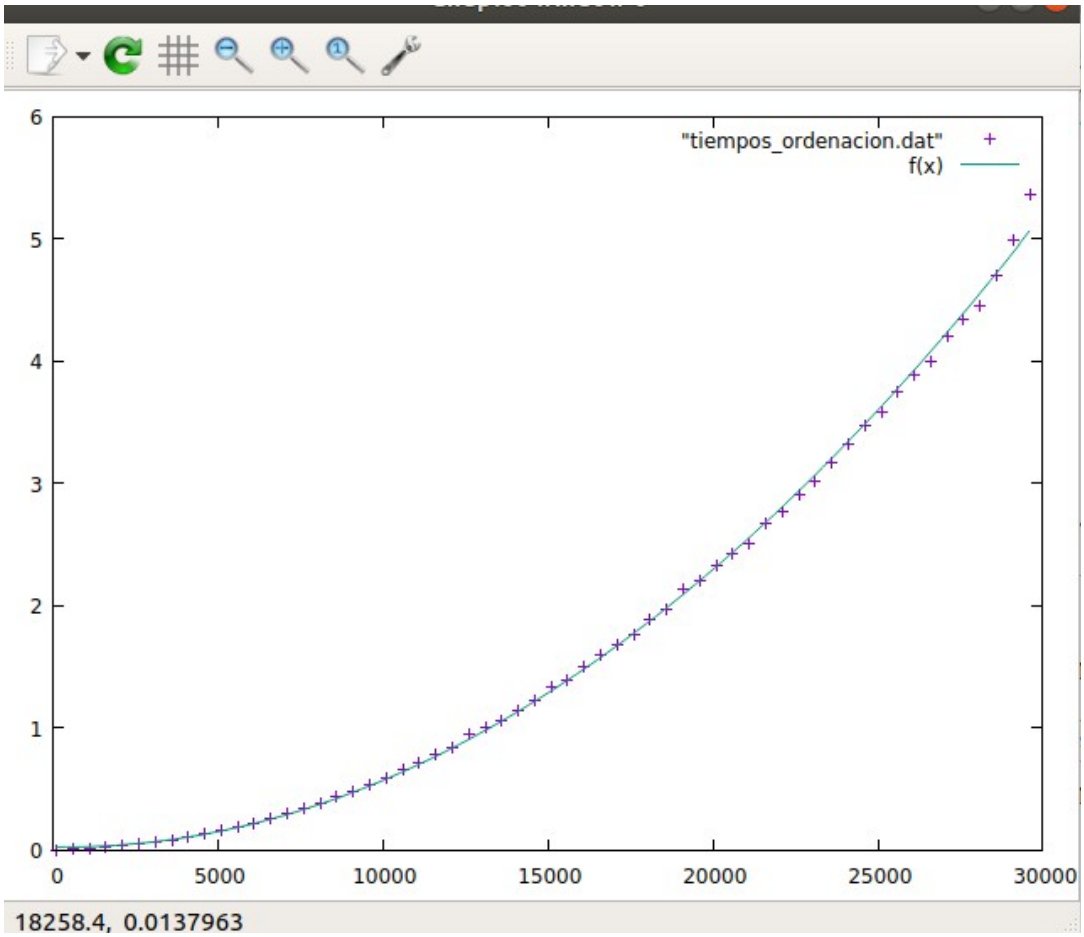


Concatenación de la eficiencia empírica y la eficiencia teórica sin ajuste:



Ejercicio 2.

```
FIT:    data read from "tiempos_ordenacion.dat"
        format = z
        #datapoints = 60
        residuals are weighted equally (unit weight)

function used for fitting: f(x)
      f(x) = a**x+b*x+c
fitted parameters initialized with current variable values

iter      chisq       delta/lim  lambda   a             b             c
   0 1.7731264234e+10   0.00e+00  6.62e+14    1.000000e+00   1.000000e+00   1.000000e+00
   1 1.7731264234e+10  -1.31e-06  6.62e+13    1.000000e+00   1.000000e+00   1.000000e+00

After 1 iterations the fit converged.
final sum of squares of residuals : 1.77313e+10
rel. change during last iteration : -1.30517e-11

degrees of freedom    (FIT_NDF)                       : 57
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 17637.3
variance of residuals (reduced chisquare) = WSSR/ndf   : 3.11075e+08

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 1                +/- 2.287e-12    (2.287e-10%)
b               = 1                +/- 0.2922       (29.22%)
c               = 1                +/- 4679         (4.679e+05%)

correlation matrix of the fit parameters:
                a      b      c
a               1.000
b              -0.436  1.000
c               0.259 -0.864  1.000
```



f(x) = ax+b*x+c**

Ejercicio 3.

Eficiencia teórica:

El algoritmo planteado se trata de la búsqueda binaria, plantea la idea siguiente:

Se busca un elemento en un vector ordenado, al principio se calcula la mitad del vector, en el caso de que éste sea el elemento, se acaba la búsqueda. En caso contrario, se plantean 2 opciones, por un lado, que el elemento buscado sea menor que el elemento de la mitad, por lo que el elemento se encontrará entre inf y med-1, la otra opción es su viceversa, es decir, que el elemento sea mayor que el de la mitad, por lo que se encontraría entre med+1 y sup, de esta manera, no se realizará una búsqueda continua en todo el vector, sino que se irá fraccionando hasta encontrar o no este elemento.

El código planteado es el siguiente:

```
int operacion(int *v, int n, int x, int inf, int sup) {
  int med; 1OE← O(1)
  bool enc=false; 2OE← O(1)
```

```
  while ((inf<sup) && (!enc)) { 3OE← O(1)
    med = (inf+sup)/2;  3OE← O(1)
    if (v[med]==x)  2OE← O(1)
      enc = true; 1OE← O(1)
    else if (v[med] < x) 2OE← O(1)
      inf = med+1; 2OE← O(1)
    else
      sup = med-1; 2OE← O(1)
  }
  if (enc) 1OE← O(1)
    return med;
  else 1OE← O(1)
    return -1;
}
```

Ahora nos centraremos en el **while**:

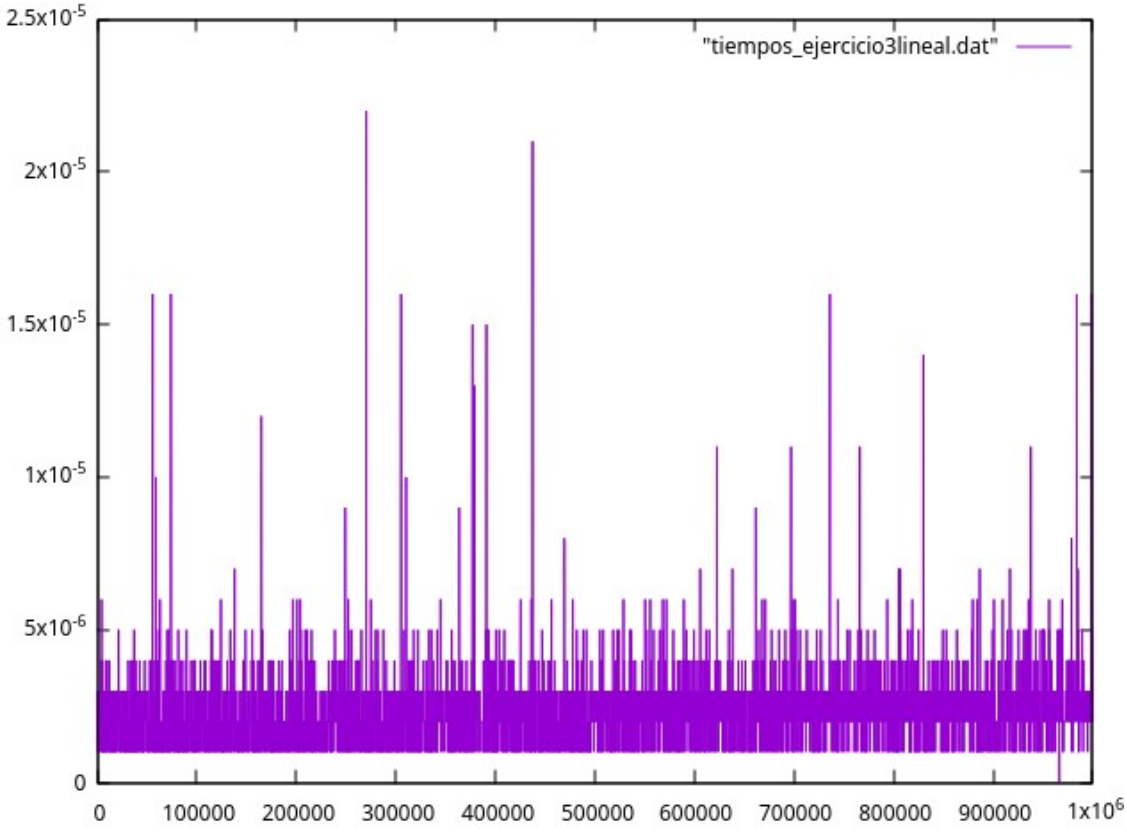
Como vimos en clase, en este caso el while utilizará un caso u otro, es decir, no realizará todas las operaciones, o else if, o else, ya que el primer if lo que haria sería terminar, no se tiene en cuenta.

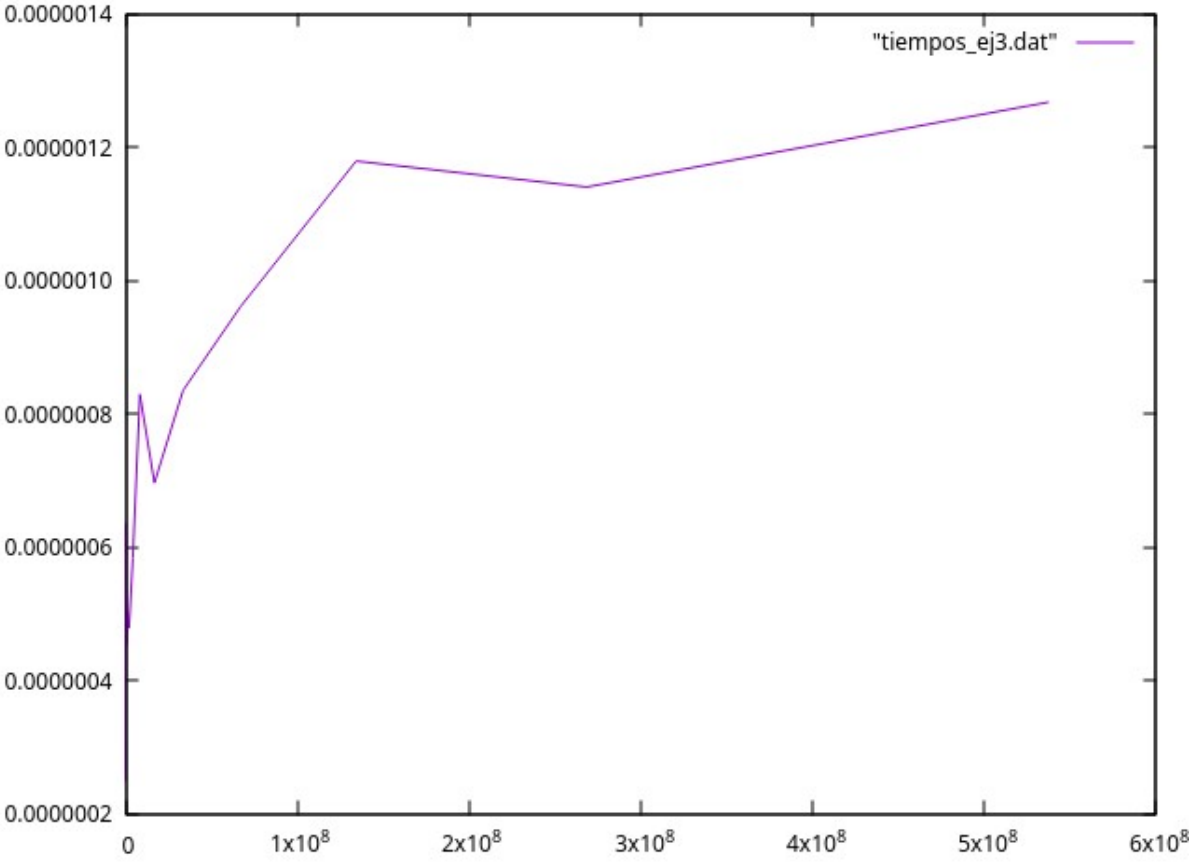
Es por esto que se debe considerar como un $\log_2(n)$:

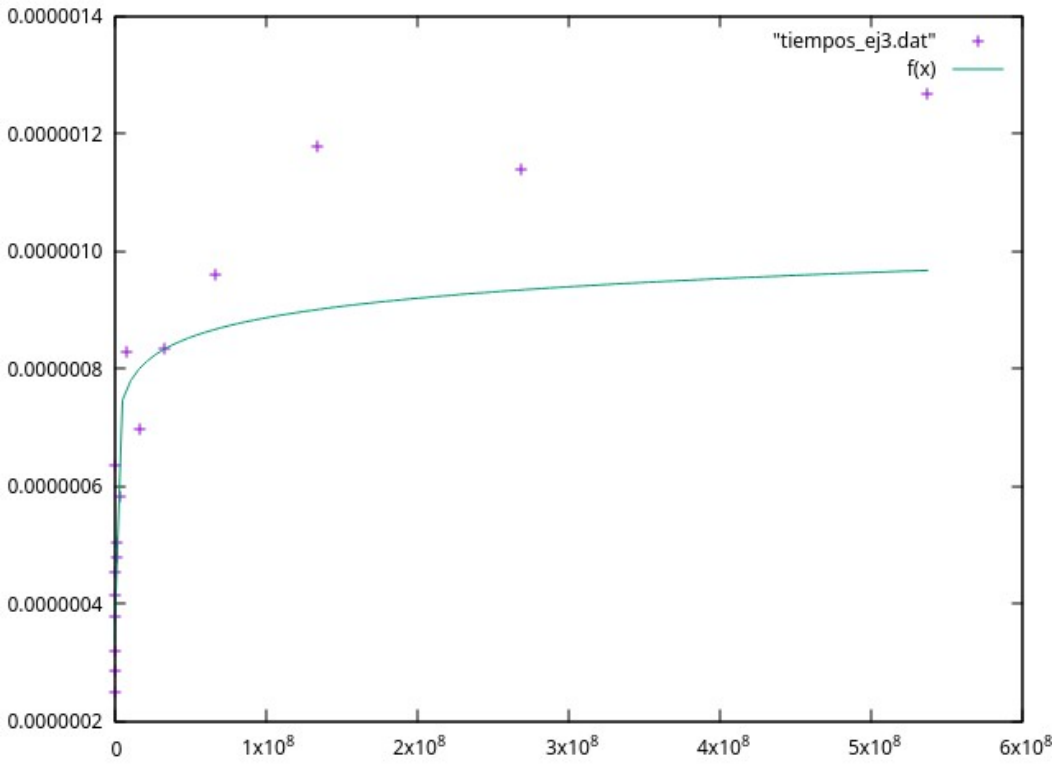
$$\sum_{inf=0}^{\log_2(n)} O(1) = \sum_{inf=0}^{\log_2(n)} 1 = \log_2(n) \in O(\log_2(n))$$

Eficiencia empírica:

La anormalidad es que al tratar enteros que no son potencia de 2, ocurre lo que se puede ver en la gráfica debido a los tiempos de ejecutar $\log_2$ de el conjunto de números dados. Es por esto que al hacero con números potencia de 2, los tiempos del algoritmo se suavizan al no dar tantos números como se obtendría mediante la forma lineal.


"tiempos_ejercicio3lineal.dat"
2.5x10-5
2x10-5
1.5x10-5
1x10-5
5x10-6
0
0
100000
200000
300000
400000
500000
600000
700000
800000
900000
1x106


"tiempos_ej3.dat"
0.0000014
0.0000012
0.0000010
0.0000008
0.0000006
0.0000004
0.0000002
0
1x108
2x108
3x108
4x108
5x108
6x108



FIT: data read from "tiempos_ej3.dat"
format = z
#datapoints = 17
residuals are weighted equally (unit weight)

function used for fitting: f(x)
f(x) = a*(log(x)/log(2))
fitted parameters initialized with current variable values

iter chisq delta/lim lambda a
0 5.4220138317e-11 0.00e+00 2.49e-06 1.166039e-07
3 4.5669623288e-13 -1.26e-02 2.49e-09 3.334081e-08

After 3 iterations the fit converged.
final sum of squares of residuals : 4.56696e-13
rel. change during last iteration : -1.25576e-07

degrees of freedom (FIT_NDF) : 16
rms of residuals (FIT_STDFIT) = sqrt(WSSR/ndf) : 1.68948e-07
variance of residuals (reduced chisquare) = WSSR/ndf : 2.85435e-14

Final set of parameters Asymptotic Standard Error
========== ==========
a = 3.33408e-08 +/- 1.919e-09 (5.754%)

f(x) = a*(log(x)/log(2))

Ejercicio 4.

```
void ordenar(int *v, int n) {
    bool cambio=true; 2OE← O(1)
    for (int i=0; i<n-1 && cambio; i++) { 6OE← O(1)
        cambio=false; 1OE← O(1)
        for (int j=0; j<n-i-1; j++) 5OE← O(1)
            if (v[j]>v[j+1]) { 3OE← O(1)
                cambio=true; 1OE← O(1)
                swap (v[j],v[j+1]); //incluir algorithm 4OE← O(1)
            }
    }
}
```

En el mejor caso, solo realizaría el bucle interno, puesto que no habría cambio, dejando la variable cambio a **false**, y no entrando de nuevo al primer **for.**

Eficiencia teórica (Mejor Caso):

Si tenemos en cuenta lo visto en el ejercicio 1, tendremos lo siguiente:

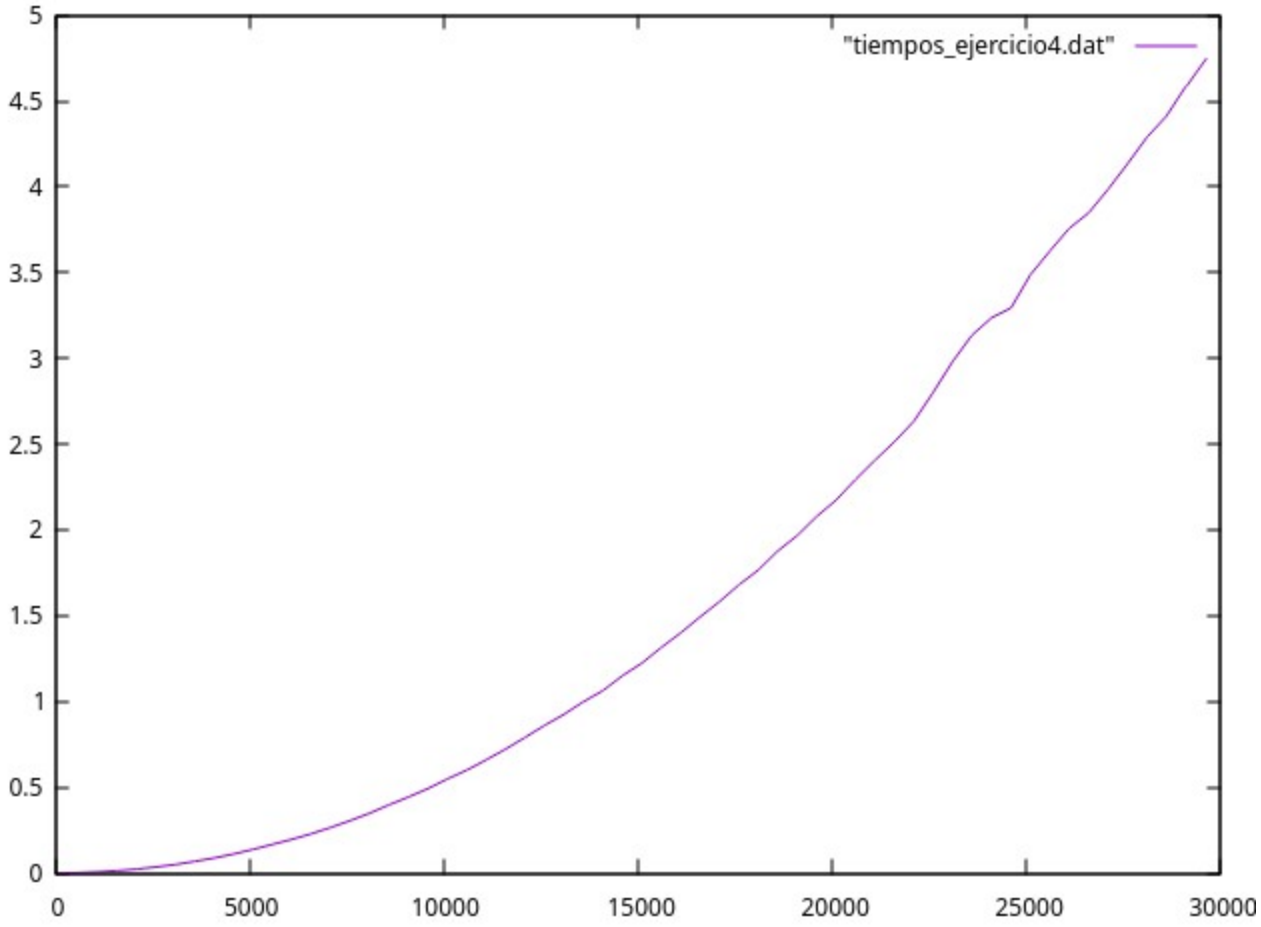
n-i-1-0+1 = n-i

Es decir:

$$\sum_{j=0}^{n-i-1} 1 = n-i$$

Y como se ejecutará una vez, se quedaría en n-i ∈ O(n)

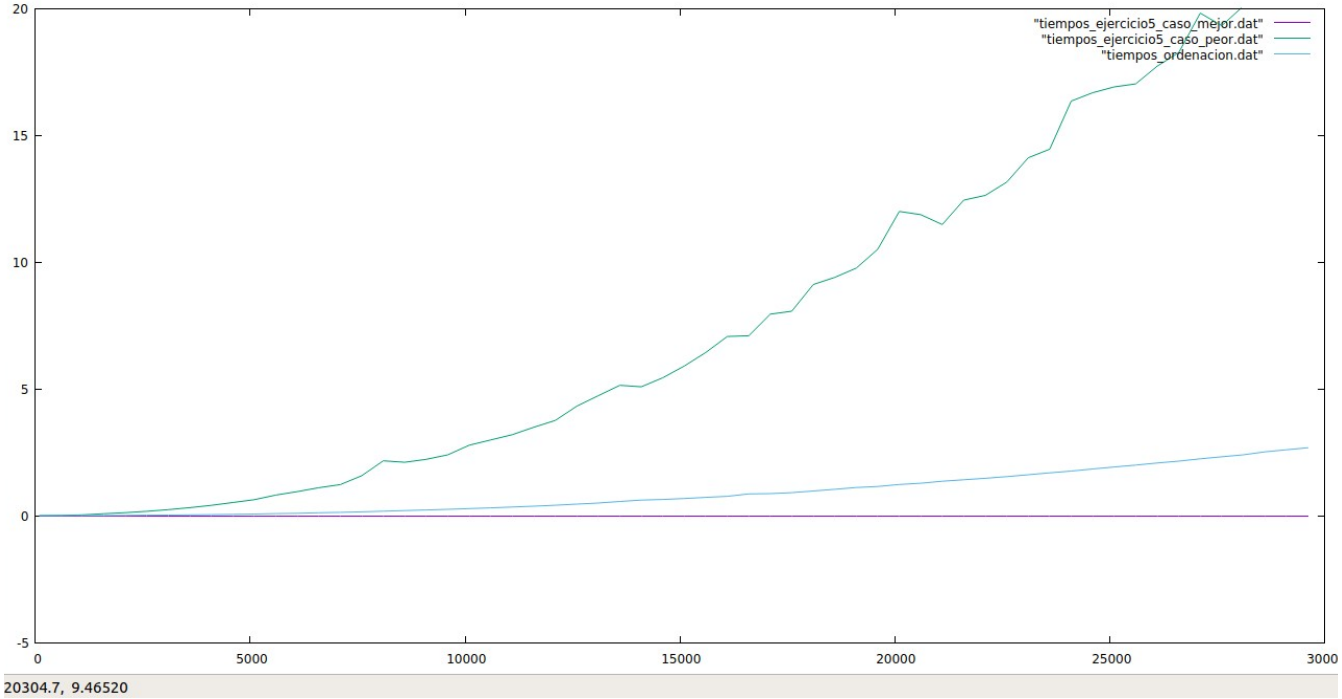
Eficiencia empírica (Mejor Caso):



No contemplo ninguna mejora en comparación con la teórica.

Ejercicio 5.

Hay una gran diferencia entre el mejor de los casos y el peor de los casos. En cuanto al ordenación por burbuja aleatoria observamos que los tiempos son mejores en el caso más ideal. Para generar el mejor de los casos hemos ordenado el vector antes de pasarle el algoritmo. En el caso del peor de los casos hemos ordenado el vector alrevés.



FIT: data read from "tiempos_ordenacion.dat"
format = z
#datapoints = 60
residuals are weighted equally (unit weight)

function used for fitting: f(x)
f(x) = a*x**2 +b*x+c
fitted parameters initialized with current variable values

iter	chisq	delta/lim	lambda	a	b	c
0	9.4779953249e+18	0.00e+00	2.29e+08	1.000000e+00	1.000000e+00	1.000000e+00
12	1.4398209622e-01	-2.74e-07	2.29e-04	5.901385e-09	-4.598669e-06	2.262233e-02

After 12 iterations the fit converged.
final sum of squares of residuals : 0.143982
rel. change during last iteration : -2.73677e-12

degrees of freedom (FIT_NDF) : 57
rms of residuals (FIT_STDFIT) = sqrt(WSSR/ndf) : 0.0502593
variance of residuals (reduced chisquare) = WSSR/ndf : 0.002526

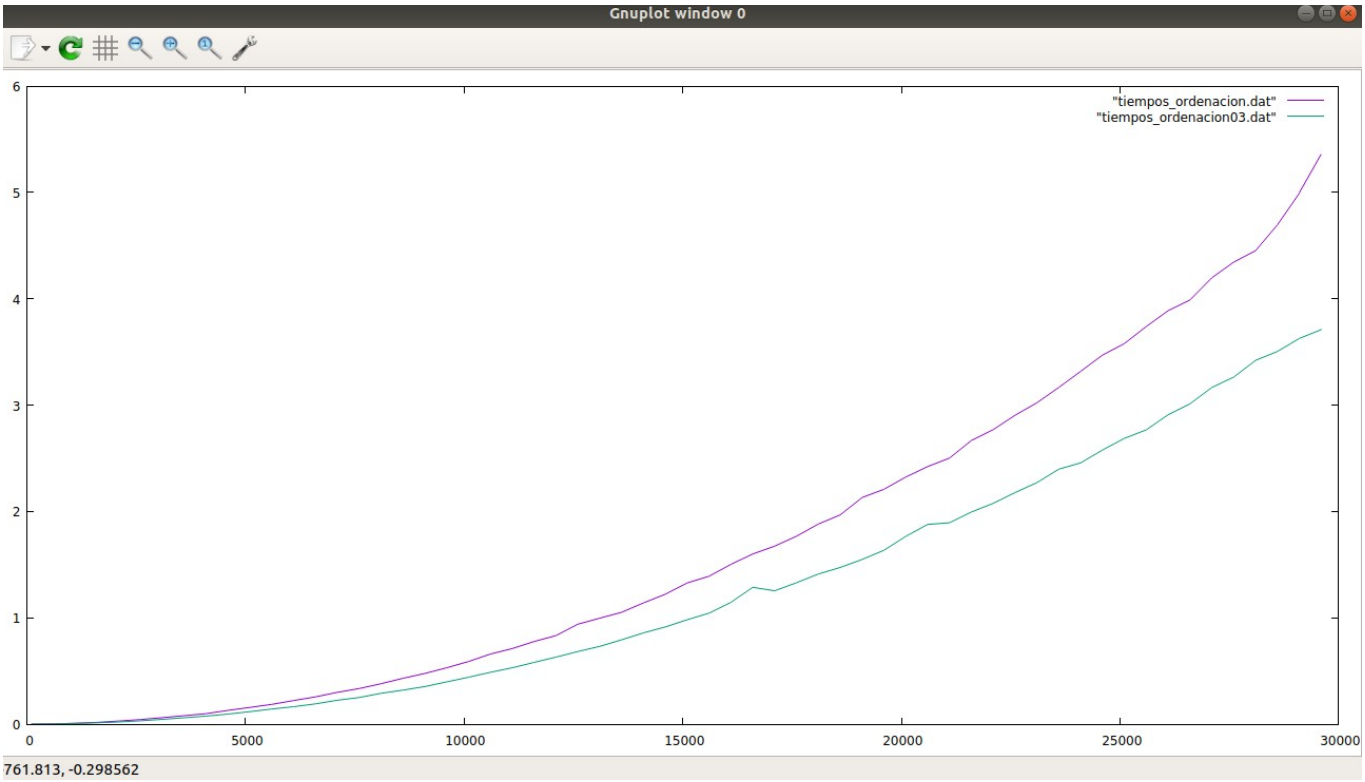
Final set of parameters	Asymptotic Standard Error
==========	==========

a = 5.90138e-09 +/- 9.679e-11 (1.64%)
b = -4.59867e-06 +/- 2.971e-06 (64.6%)
c = 0.0226223 +/- 0.01909 (84.38%)

correlation matrix of the fit parameters:

	a	b	c
a	1.000		
b	-0.968	1.000	
c	0.738	-0.861	1.000

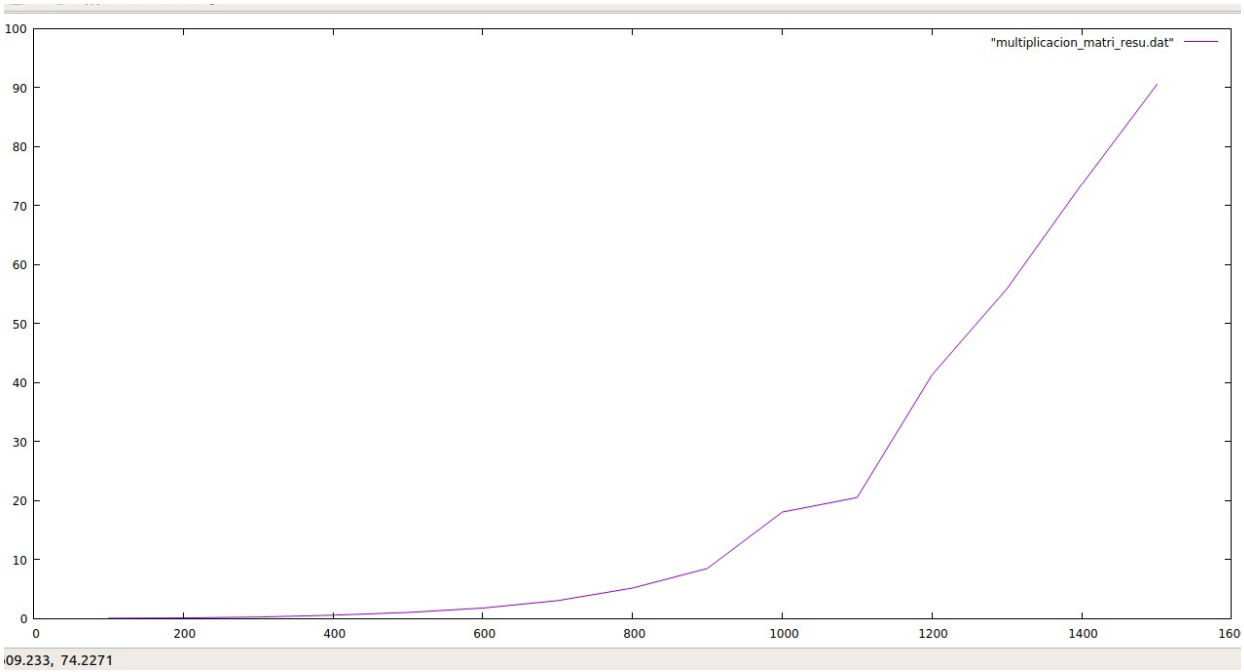
Ejercicio 6.



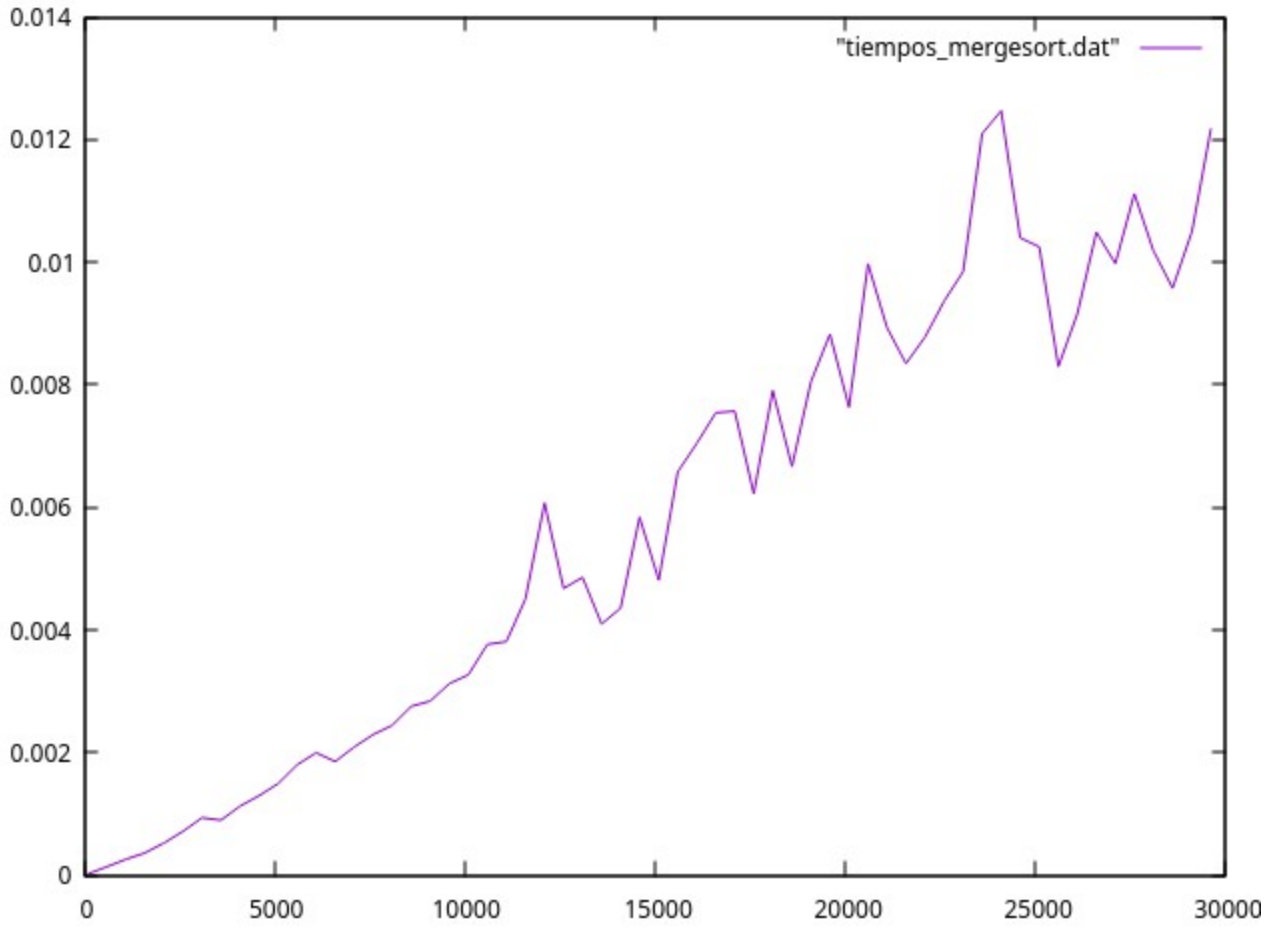
Con optimización 03 hay una mejora en los tiempos esto se debe a que al optimizar el compilador usa un menos número de instrucciones por lo que es más eficiente.

Ejercicio 7.

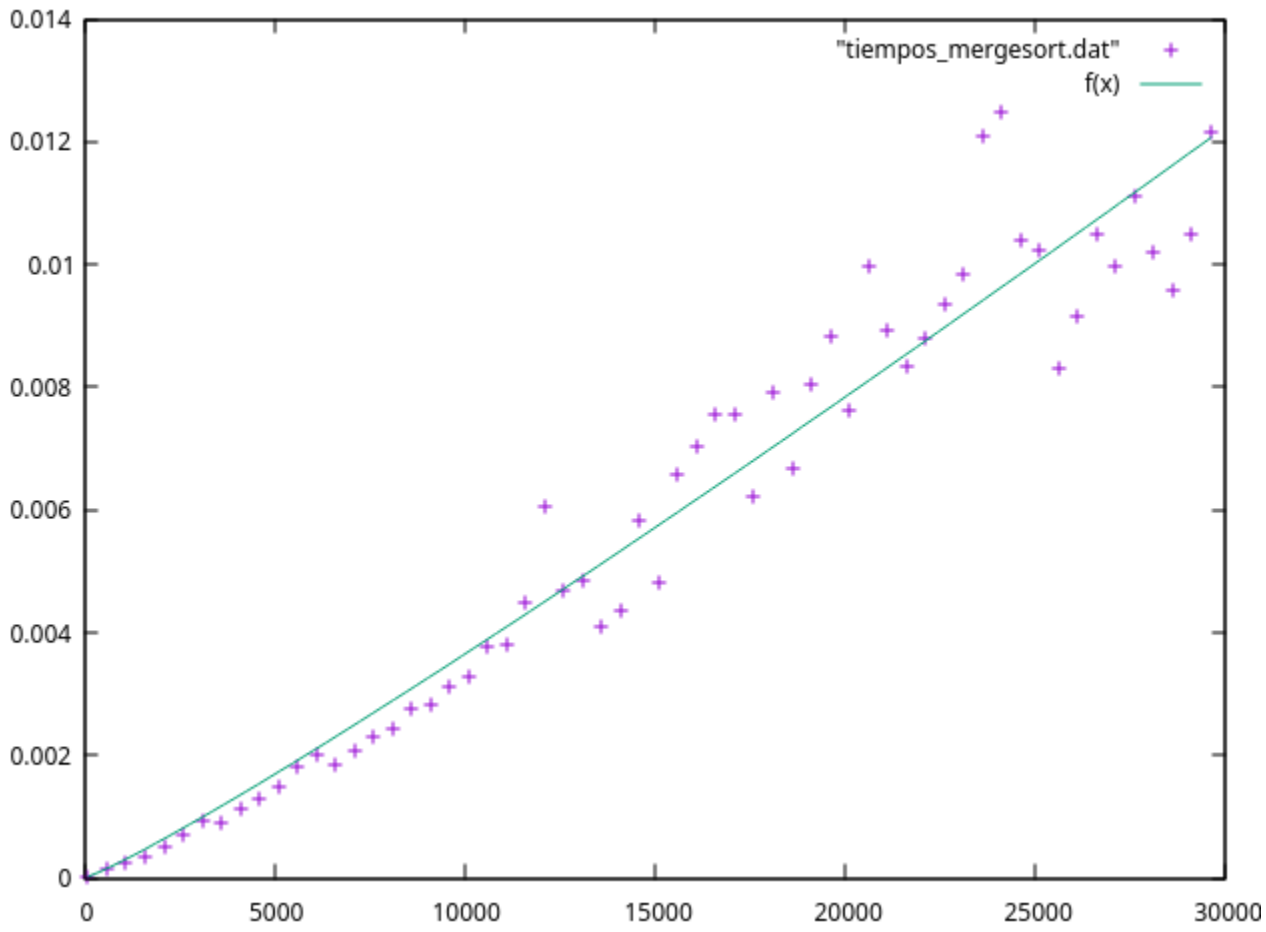
Hemos modificado el fichero .csv ya que con los valores que utilizabamos para las anteriores funciones el odenador no era capaz de operar. El tamaño comienza en 100, tiene un crecimiento de 100 hasta 1500. La matriz es de dimensiones tam x tam. En la gráfica podemos observar como el aumento del tiempo es exponencial a cuanta mayor cantidad de datos:


"multiplicacion_matri_resu.dat"
100
90
80
70
60
50
40
30
20
10
0
0
200
400
600
800
1000
1200
1400
160
09.233, 74.2271

Ejercicio 8.


"tiempos_mergesort.dat"
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
5000
10000
15000
20000
25000
30000

Ajuste con n(log)n:



FIT: data read from "tiempos_mergesort.dat"
format = z
#datapoints = 60
residuals are weighted equally (unit weight)

function used for fitting: f(x)
f(x) = a*x*(log(x)/log(2))
fitted parameters initialized with current variable values

iter chisq delta/lim lambda a
0 3.6726828941e+12 0.00e+00 2.47e+05 1.000000e+00
4 4.8303243741e-05 -1.58e-01 2.47e+01 2.744131e-08

After 4 iterations the fit converged.
final sum of squares of residuals : 4.83032e-05
rel. change during last iteration : -1.57615e-06

degrees of freedom (FIT_NDF) : 59

rms of residuals (FIT_STDFIT) = sqrt(WSSR/ndf) : 0.00090482
variance of residuals (reduced chisquare) = WSSR/ndf : 8.18699e-07

Final set of parameters		Asymptotic Standard Error	
a	= 2.74413e-08	+/- 4.721e-10	(1.721%)