

# Algoritmo Mergesort

El algoritmo de ordenamiento por mezcla (merge sort en inglés) es un algoritmo de ordenamiento externo estable basado en la técnica divide y vencerás.

## Pasos a seguir para medir la eficiencia:

1. Average / Best Case Scenario: Generar vector de enteros de tamaño  $n$  y se llena aleatoriamente
2. Worst Case Scenario: Generar Vector de enteros ordenados
3. Guardamos tiempos antes de ejecutar y después de ejecutar el algoritmo
4. Calcular tiempo.

## Información

## Hardware usado (CPU, velocidad de reloj, memoria RAM, ...)

```
~/Dropbox/UGR/ALG/Practica-1/Algoritmica/Permutacion on - David @ 19:39:40
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    2
Core(s) per socket:    2
Socket(s):             1
NUMA node(s):         1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                142
Model name:            Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
Stepping:              9
CPU MHz:              699.996
CPU max MHz:          3100.0000
CPU min MHz:          400.0000
BogoMIPS:              5423.85
Virtualization:        VT-x
L1d cache:            32K
L1i cache:            32K
L2 cache:             256K
L3 cache:             3072K
NUMA node CPU(s):     0-3
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc perfmon_rplstar eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch eptb invpcid_single intel_pt kaiser tpr_shadow vmmi ftoprriority ept vpid fsgsbase tsc_adjust bti avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt xsaveopt xsavec xgetbv1 dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp
```

~/Dropbox/UGR/ALG/Practica-1/Algoritmica/Permutacion on - David @ 19:39:48

Mergesort DavidCardona@elucidator ~/Dropbox/UGR/ALG/Practica-1/Algoritmica/Mergesort - zsh

Mergesort DavidCardona@elucidator ~/Dropbox/UGR/ALG/Practica-1/Algoritmica/Mergesort - zsh - Atomic Terminal

## Compilador utilizado y opciones de compilación

```
gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/5/lto-wrapper
Target: x86_64-linux-gnu
gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.9)
```

# Compilación

```
g++ -std=c++11 ./src/$PROGRAMA.cpp -o ./bin/$PROGRAMA
```

Usamos el siguiente script:

# Variables:

PROGRAMA=\$1

SALIDA=./data/tiempo\_worst\_case\_\$1.dat

MENSAJE\_INICIO="Se inicia la ejecución del algoritmo \$1:"

MENSAJE\_FINAL="Fin de la ejecución. Se ha creado un fichero con los resultados."

*# Se genera el ejecutable con el algoritmo de ordenación:*

```
g++ -std=c++11 ./src/$PROGRAMA.cpp -o ./bin/$PROGRAMA
```

```
echo "$MENSAJE_INICIO"
```

*# Variables:*

```
INICIO=1000
```

```
FIN=50000
```

```
INCREMENTO=1000
```

```
i=$INICIO
```

```
echo > $SALIDA
```

```
while [ $i -le $FIN ]
```

```
do
```

```
    echo Vector size = $i
```

```
    echo "`./bin/$PROGRAMA $i `" >> $SALIDA
```

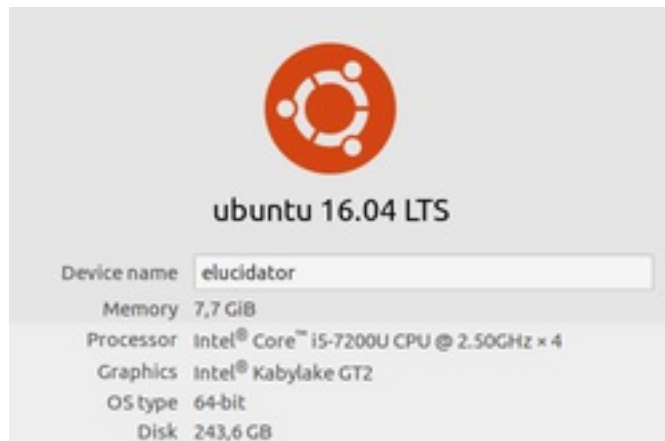
```
    i=$((i+$INCREMENTO))
```

```
done
```

```
rm -fr ./bin/$PROGRAMA
```

```
echo "$MENSAJE_FINAL"
```

## Sistema operativo



## Desarrollo completo del cálculo de la eficiencia teórica y gráfica.

### Calculo de eficiencia $O(n \log n)$

El  $T(n)$  es igual  $2T(n/2) + n$  (No "" + 1") Errata.

*Formula Maestra*

Tenemos "a" llamadas recursivas de " $n/b$ " tamaño cuya mezcla requiere  $f(n) \in O(n^p)$

con  $T(n) \in \begin{cases} O(n^{\log_b a}) & \text{si } a > b^p \\ O(n^p \cdot \log n) & \text{si } a = b^p \\ O(n^p) & \text{si } a < b^p \end{cases}$

$a=2$  ;  
 $n/2 \rightarrow b=2$  ;  
 $p=1$

*Aplicamos a mergesort*  $a=2 \rightarrow n/2 \rightarrow b=2$   $p=1$   $O(n)$  mezcla

Entonces Mergesort  $O(n \cdot \log n)$

↓

Teniendo en cuenta que

$$T(n) = 2T(n/2) + n$$

# Eficiencia Empirica

Hemos tomado multiples medidas y sobre esto hemos realizado los ajustes y graficas.

## Ejemplo de medidas Worst Case

N Elementos	Tiempo
1000	0.00012729
2000	0.000245837
3000	0.000381419
4000	0.00051691
5000	0.000681268
6000	0.000793639
7000	0.000932129
8000	0.00107566
9000	0.00124717
10000	0.00137646
11000	0.001584
12000	0.001945
13000	0.001859
14000	0.002107
15000	0.002258
16000	0.00231
17000	0.00332
18000	0.003403
19000	0.00378

20000	0.004421
21000	0.004415
.....	.....

Merge ocupa mucho tiempo extra, por lo tanto suele ser mas lenta que otros algoritmos se ha notado en las graficas realizada, respecto a los otros.

## Parámetros usados para el cálculo de la eficiencia empírica y gráfica.

Para el calculo de las gráficas hemos usado el script:

```
#!/bin/bash

#Variables:
OUTPUT=./data/grafica_tiempo_worst_case_mergesort.png
TITULO="Algoritmo Mergesort Worst Case No Insertion"
XLABEL="Longitud del Vector"
YLABEL="Tiempo (segundos)"
LEYENDA="Algoritmo Mergesort  $O(n \cdot \log(n))$ "
FICHERO_DATOS="./data/tiempo_worst_case_mergesort.dat"
COLOR=blue

gnuplot<<FIN
# Terminal para png:
set terminal pngcairo enhanced font 'Verdana,10'
set border linewidth 1.5

# Estilo de línea y color:
set style line 1 lc rgb '$COLOR' lt 1 lw 2 pt 7 pi 0 ps 0.5
set pointintervalbox 0

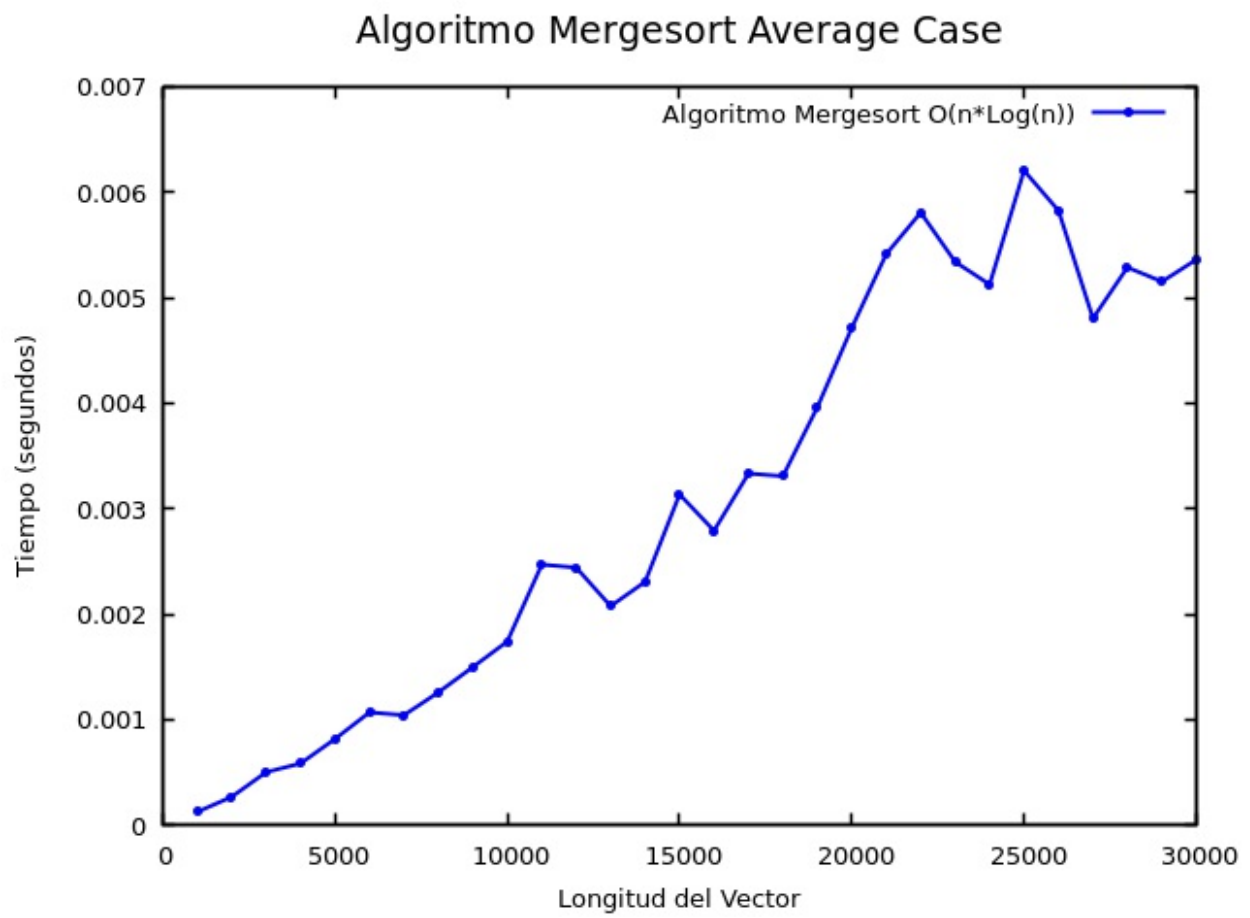
# Nombre de la imagen resultante:
set output '$OUTPUT'

# Título y ejes:
set title "$TITULO" enhanced font 'Verdana,14'
set xlabel "$XLABEL"
set ylabel "$YLABEL"

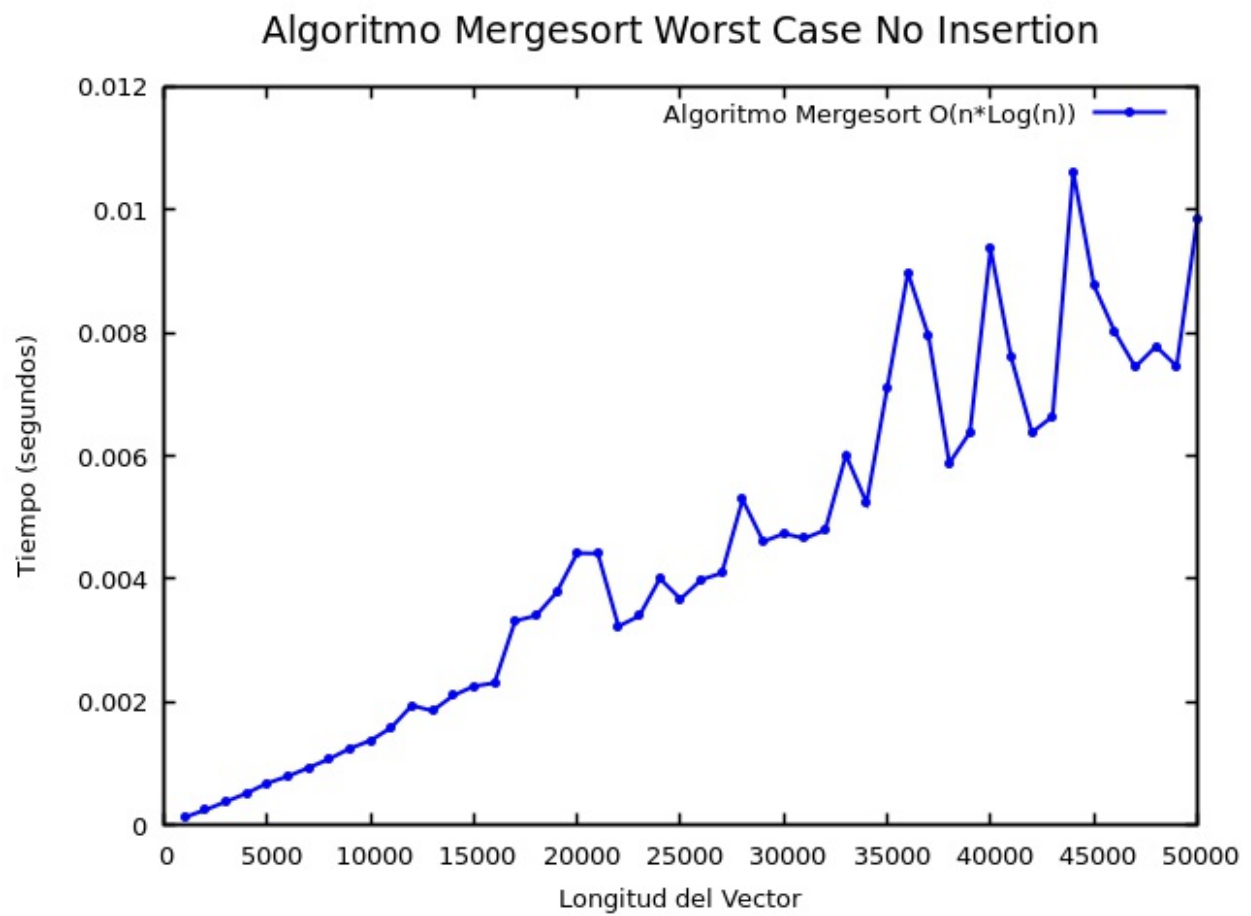
set autoscale

plot "$FICHERO_DATOS" title '$LEYENDA' with linespoints ls 1
FIN
```

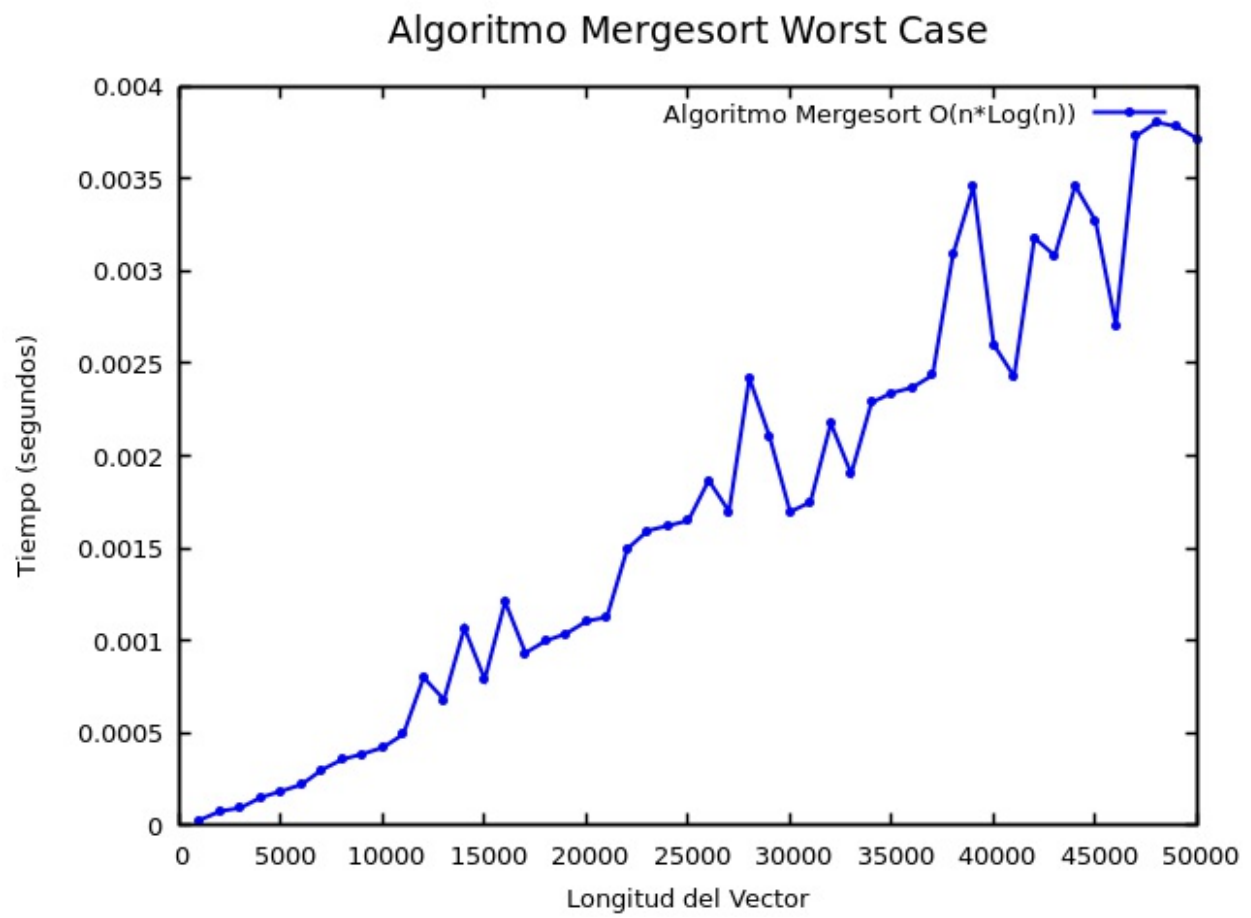
## Average



## No Insertion



## Insertion

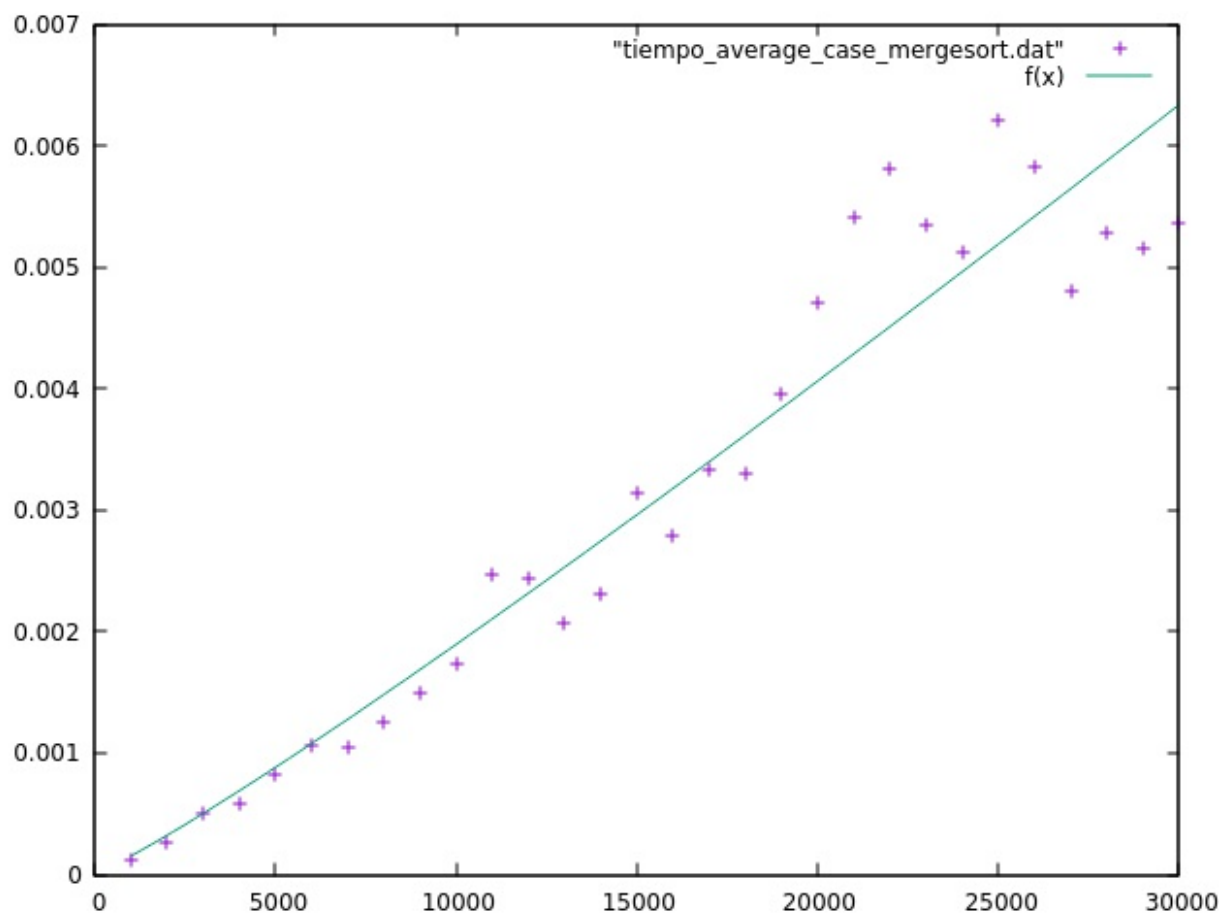




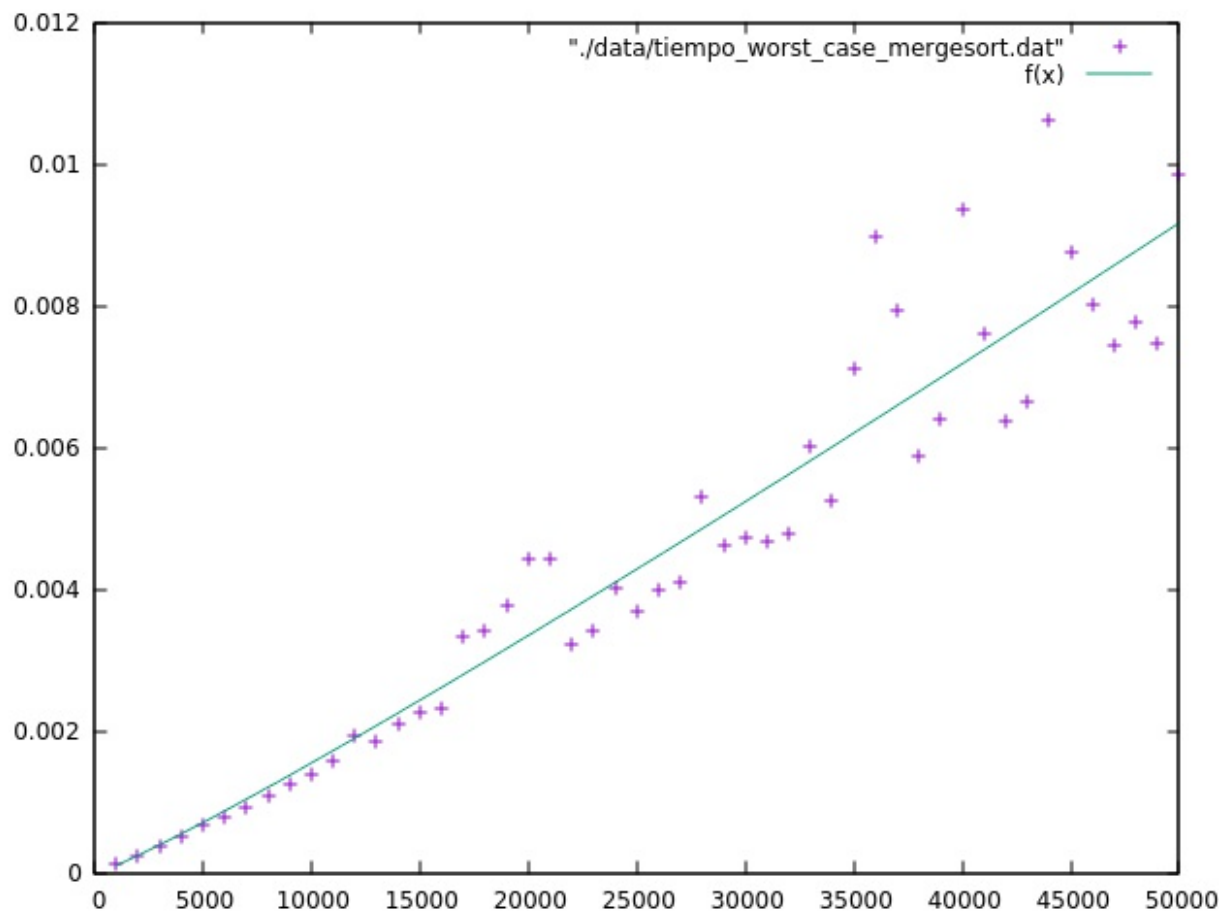
## Ajuste de la curva teórica a la empírica: mostrar resultados del ajuste y gráfica.

---

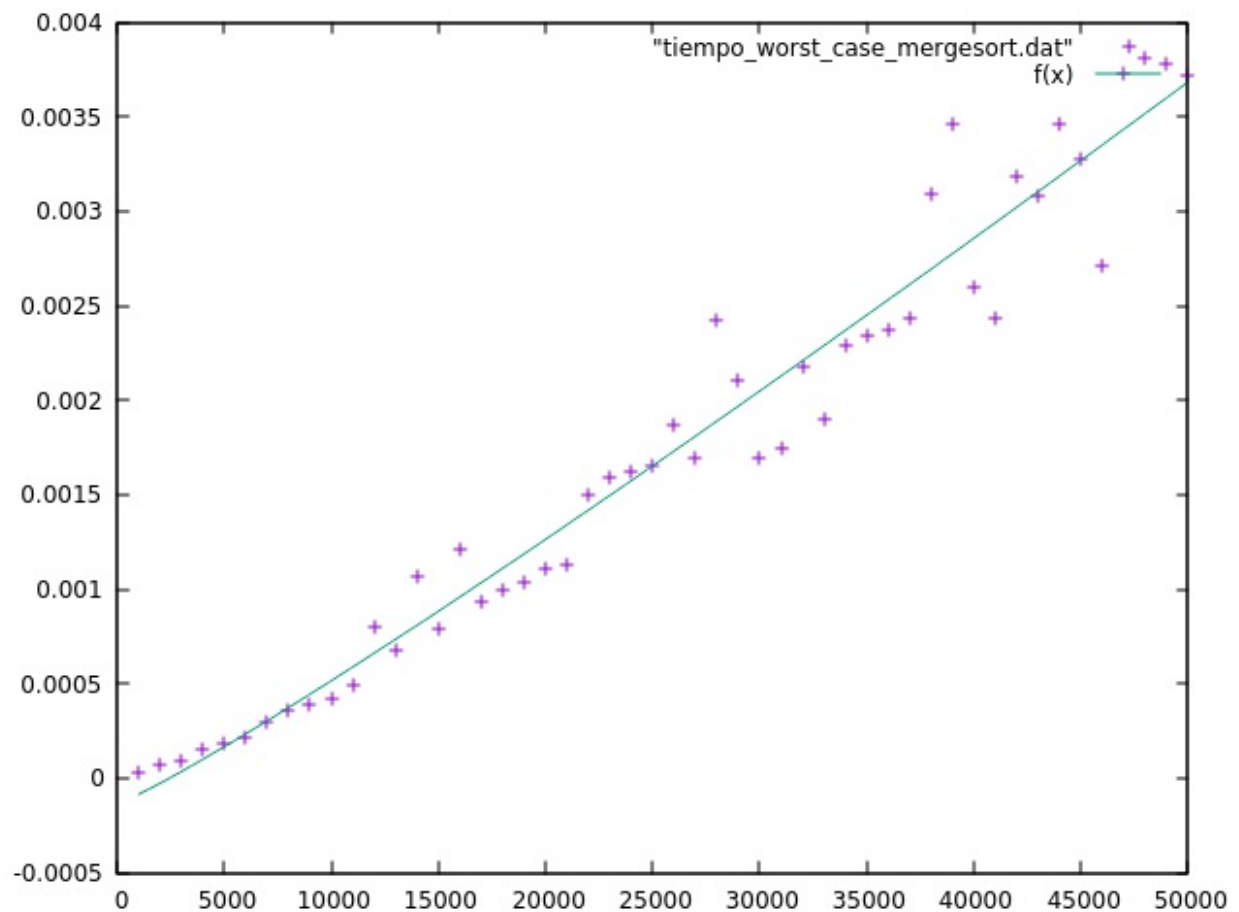
### Average



## Worst No Insertion



## Worst Insertion



## Ajuste

\*\*\*\*\*

Sat Mar 17 03:02:37 2018

FIT: data read from "tiempo\_average\_case\_mergesort.dat"  
format = z  
#datapoints = 30  
residuals are weighted equally (unit weight)

function used for fitting:  $f(x)$   
 $f(x) = (a \cdot x \cdot \log(x) / \log(2)) + b$   
fitted parameters initialized with current variable values

iter	chisq	delta/lim	lambda	a	b
0	1.9669559893e+12	0.00e+00	1.81e+05	1.000000e+00	1.000000e+00
9	8.9112223740e-06	-2.13e-04	1.81e-04	1.418250e-08	9.523383e-06

After 9 iterations the fit converged.

final sum of squares of residuals : 8.91122e-06  
rel. change during last iteration : -2.12717e-09

degrees of freedom (FIT\_NDF) : 28  
rms of residuals (FIT\_STDFIT) = sqrt(WSSR/ndf) : 0.000564144  
variance of residuals (reduced chisquare) = WSSR/ndf : 3.18258e-07

Final set of parameters		Asymptotic Standard Error	
=====		=====	
a	= 1.41825e-08	+/- 7.835e-10	(5.524%)
b	= 9.52338e-06	+/- 0.0002006	(2107%)

correlation matrix of the fit parameters:

	a	b
a	1.000	
b	-0.858	1.000

\*\*\*\*\*  
Sat Mar 17 03:04:05 2018

FIT: data read from "tiempo\_worst\_case\_mergesort.dat"  
format = z  
#datapoints = 50  
residuals are weighted equally (unit weight)

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

iter	chisq	delta/lim	lambda	a	b
0	9.0520846224e-04	0.00e+00	4.45e-03	1.418250e-08	9.523383e-06
5	2.8449514907e-06	-9.73e-01	4.45e-08	4.893749e-09	-1.354714e-04

After 5 iterations the fit converged.  
final sum of squares of residuals : 2.84495e-06  
rel. change during last iteration : -9.7309e-06

degrees of freedom (FIT\_NDF) : 48  
rms of residuals (FIT\_STDFIT) = sqrt(WSSR/ndf) : 0.000243454  
variance of residuals (reduced chisquare) = WSSR/ndf : 5.92698e-08

Final set of parameters		Asymptotic Standard Error	
=====		=====	
a	= 4.89375e-09	+/- 1.501e-10	(3.066%)
b	= -0.000135471	+/- 6.662e-05	(49.17%)

correlation matrix of the fit parameters:

	a	b
a	1.000	
b	-0.856	1.000