Estructura de datos (2017-2018)
Grado en Ingeniería Informática
Universidad de Granada

Práctica 1: eficiencia

Manuel Jiménez Bernal

12 de octubre de 2017

1. Ejercicio 1: Ordenación de la burbuja

La eficiencia teórica de este algoritmo es de clasificación cuadrática:

$$\frac{n^2 - n}{2}$$

En notación O (peor caso): $O(n^2)$
Para automatizar la ejecución del algoritmo se preparó un script que invoca el programa compilado con parámetros incrementales.

```
#!/bin/csh
@ inicio = 100
@ fin = 30000
@ incremento = 500
set ejecutable = burbuja
set salida = tiempos_burbuja.dat

@ i = $inicio
echo > $salida
while ( $i <= $fin )
echo Ejecucion tam = $i
echo `./{$ejecutable} $i 10000` >> $salida
@ i += $incremento
end
```

Esta ejecución genera un fichero con los tiempos obtenidos a partir de los distintos valores de tamaño a la hora de crear el vector. Con la herramienta gnuplot se generó una gráfica de puntos a partir de estos datos. Con el siguiente comando:

```
>plot 'tiempos_burbuja.dat'
```

Se genera el siguiente gráfico:

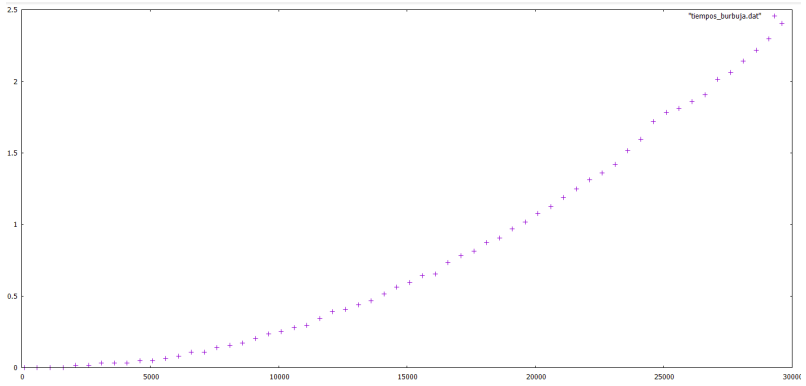

Figura 1: Gráfica de puntos a partir de los datos del fichero tiempos_burbuja.dat

Con la siguiente secuencia de gnuplot puede visualizarse la comparativa gráfica entre los datos de la eficiencia teórica con respecto a la eficiencia empírica:

```
set xrange [0 : 30000]
set x2range [0 : 3000]
set yrange [0 : 3]
set y2range [0 : 5000000]
plot 'tiempos\_burbuja.dat' axes x1y1, (x*x-x)/2 axes x2y2
```

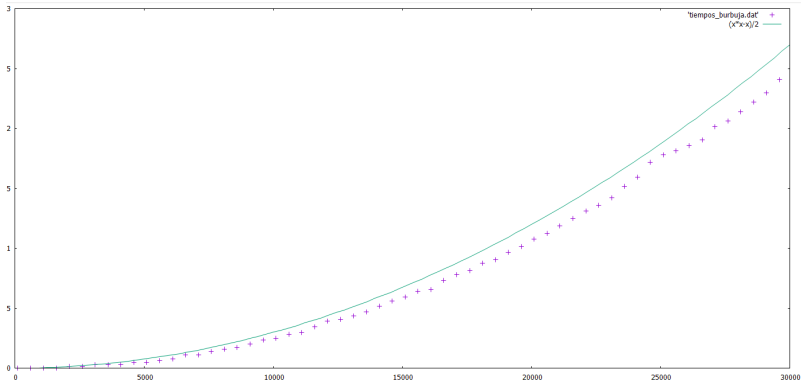

Figura 2: Puede decirse que la eficiencia empírica se ajusta a la eficiencia teórica.

2. Ejercicio 2: Ajuste en la ordenación de la burbuja

Se realizaron los siguientes comandos de gnuplot:

```
f(x) = (a*x)**2 + b*x + c
fit f(x) "tiempos_burbuja.dat" via a, b, c
```

Tras lo cual se generó un fichero 'fit.log' con el siguiente contenido

```
Thu Sep 28 20:31:15 2017
FIT:    data read from 'tiempos_burbuja.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 9.4779953826e+18 0.00e+00 2.29e+08   1.000000e+00  1.000000e+00  1.000000e+00
12 9.4804551914e-04 -4.34e-05 2.29e-04  2.841925e-09 -3.242199e-06 9.435810e-04

After 12 iterations the fit converged.
final sum of squares of residuals : 0.000948046
rel. change during last iteration : -4.33886e-10

degrees of freedom  (FIT_NDF)                        : 57
rms of residuals    (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.00407828
variance of residuals (reduced chisquare) = WSSR/ndf : 1.66324e-05

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 2.84192e-09      +/- 7.854e-12   (0.2764%)
b               = -3.2422e-06      +/- 2.411e-07   (7.435%)
c               = 0.000943581      +/- 0.001549    (164.2%)

correlation matrix of the fit parameters:
a      b      c
a               1.000
b              -0.968  1.000
c               0.738 -0.861  1.000
```

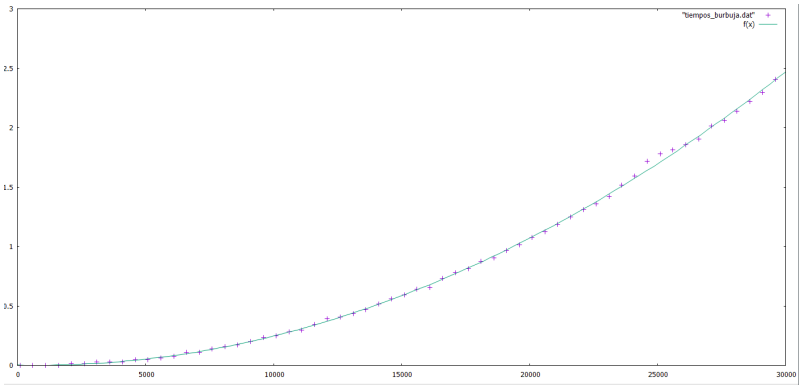

Figura 3: Ajuste por regresión de los datos obtenidos

3. Ejercicio 3: Problemas de precisión

Se trata de un algoritmo de búsqueda binaria para un vector de números ya ordenados. El algoritmo acota la búsqueda en función del valor del número en casa posición, y divide a la mitad el tamaño del problema en las sucesivas iteraciones. En el programa se fuerza el peor caso pasando como argumento un número que no existe en el array.

3.1. Ejercicio 3: Eficiencia teórica

En el peor caso el vector puede dividirse tantas veces como subconjuntos de mitades pueda hacerse de cada uno de esos subconjuntos iterativamente. Este comportamiento implica una eficiencia de orden logarítmico $O(log_2(n))$

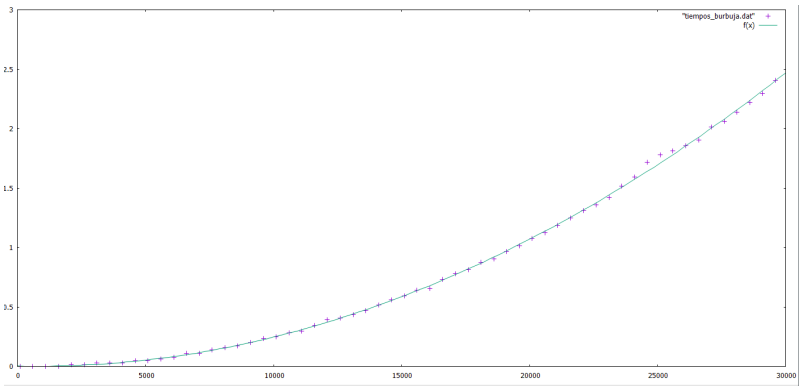

Figura 4: Eficiencia teórica

3.2. Ejercicio 3: Eficiencia empírica

Para medir los tiempos empíricos de este algoritmo se ha realizado un script con el que generar de forma automática vectores de tamaños distintos en cada iteración, tras extraer estos datos se dibujaron gráficas en gnuplot:

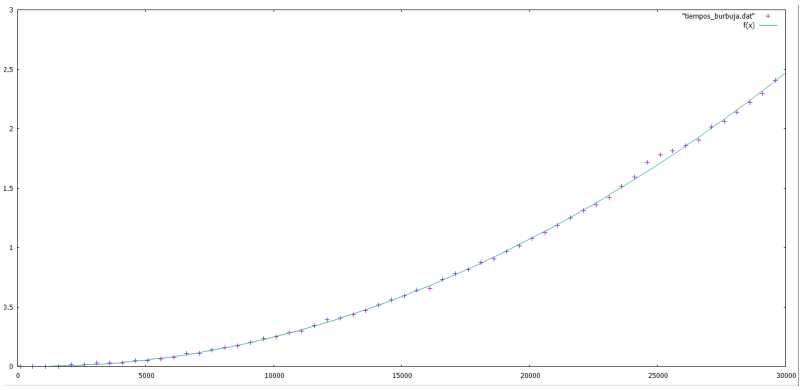

Figura 5: Eficiencia empírica

Como se aprecia en la gráfica hay una anomalía en la representación, ya que eventualmente los tiempos se disparan de forma muy irregular. Esto es debido a que en la implementación no se tiene en cuenta que ,por tanto, una vez corregido el problema se volvió a realizar el estudio empírico de los tiempos obteniendo el siguiente resultado:

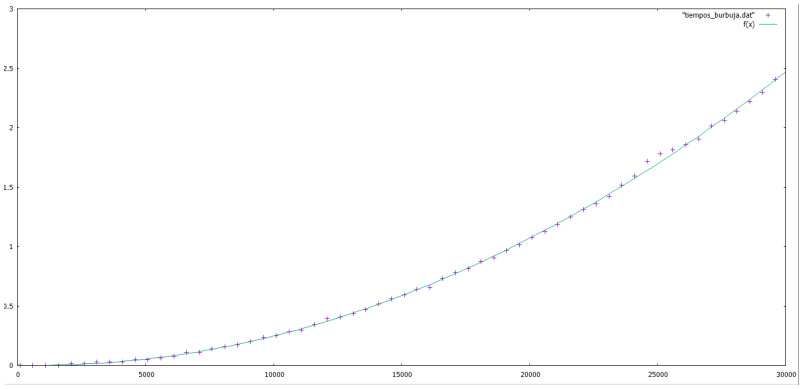

Figura 6: Eficiencia empírica tras corregir condición

El ajuste por regresión resultó de la siguiente manera:

4. Ejercicio 4: Mejor y peor caso

Antes de realizar modificaciones la ejecución por lotes mostraba los siguientes resultados de tiempo:

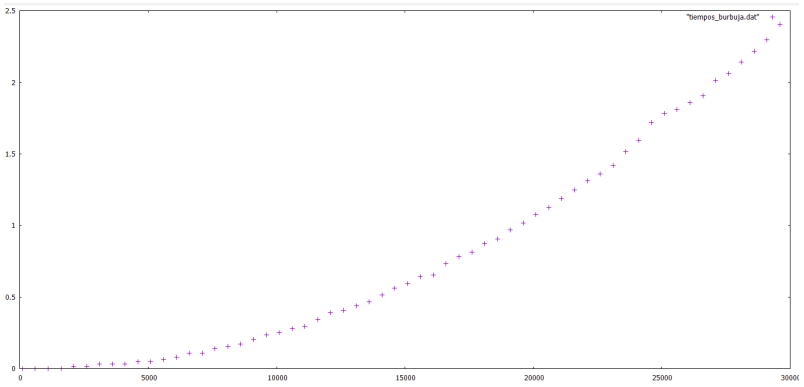

Figura 7: Gráfica de puntos a partir de los datos del fichero tiempos_burbuja.dat sin modificaciones

En el peor caso:

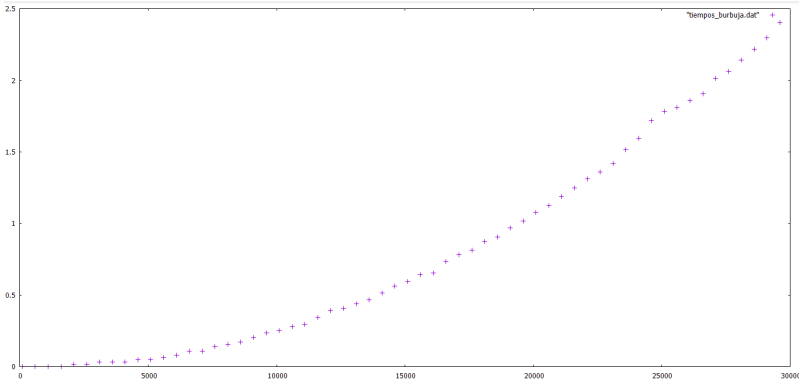

Figura 8: Gráfica de puntos a partir de los datos del fichero tiempos_burbuja.dat sin modificaciones

En el mejor caso:

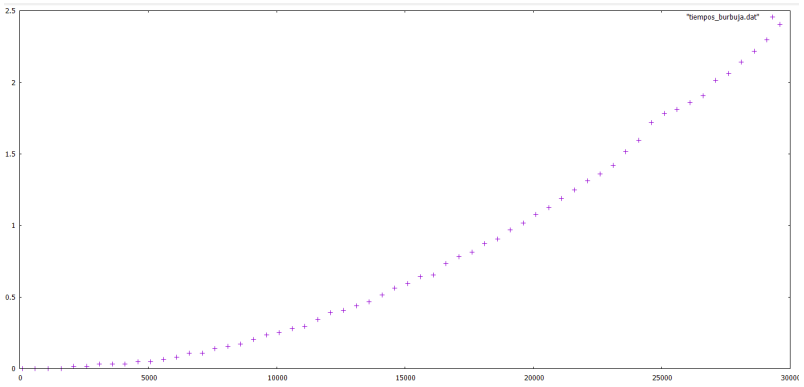

Figura 9: Gráfica de puntos a partir de los datos del fichero tiempos_burbuja.dat sin modificaciones

La representación de los tres escenarios:

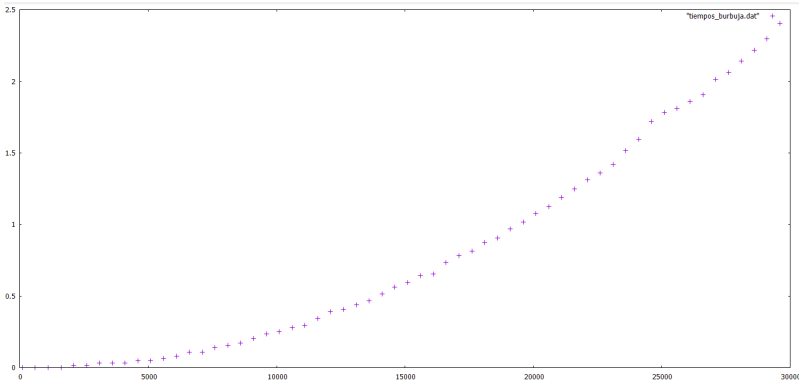

Figura 10: Gráfica de puntos a partir de los datos del fichero tiempos_burbuja.dat sin modificaciones

5. Ejercicio 5

Al introducir el cambio y encontrarse el vector ya ordenado, el algoritmo entra únicamente en la primera iteración al bucle exterior, recorriendo el bucle interior de forma completa una sola vez. Por tanto se trata de una eficiencia lineal $O(n)$ cuando el vector ya está ordenado.

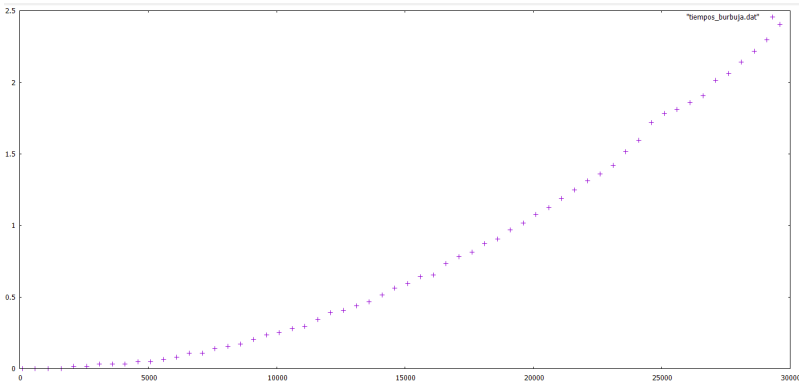

Figura 11: Gráfica de puntos a partir de los datos del fichero tiempos_burbuja.dat sin modificaciones

Referencias