

# EJERCICIO 1 : Ordenación de la burbuja

Autor : Jesús Ruiz Castellano,

76439001-L

## 1.- Código fuente : ordenacion.cpp

```
1 #include <iostream>
2 #include <ctime>      // Recursos para medir tiempos
3 #include <cstdlib>    // Para generación de números pseudoaleatorios
4
5 using namespace std;
6
7 void ordenar(int *v, int n) {          // Algoritmo ordenacion por burbuja
8     for (int i=0; i < n-1; i++)
9         for (int j=0; j < n-i-1; j++)
10             if (v[j] > v[j+1]) {
11                 int aux = v[j];
12                 v[j] = v[j+1];
13                 v[j+1] = aux;
14             }
15 }
16
17
18 void sintaxis()
19 {
20     cerr << "Sintaxis:" << endl;
21     cerr << " TAM: Tamaño del vector (>0)" << endl;
22     cerr << " VMAX: Valor máximo (>0)" << endl;
23     cerr << "Se genera un vector de tamaño TAM con elementos aleatorios en [0,VMAX]" << endl;
24     exit(EXIT_FAILURE);
25 }
26
27 int main(int argc, char * argv[])
28 {
29     // Lectura de parámetros
30     if (argc!=3)
31         sintaxis();
32     int tam=atoi(argv[1]);    // Tamaño del vector
33     int vmax=atoi(argv[2]);  // Valor máximo
34     if (tam<=0 || vmax<=0)
35         sintaxis();
36
37     // Generación del vector aleatorio
38     int *v=new int[tam];       // Reserva de memoria
39     srand(time(0));            // Inicialización del generador de números pseudoaleatorios
40     for (int i=0; i<tam; i++)  // Recorrer vector
41         v[i] = rand() % vmax;  // Generar aleatorio [0,vmax[
42
43
44     for (int i=0; i < tam-1; i++)
45         for (int j=0; j < tam-i-1; j++)
46             if (v[j] < v[j+1]) { // ordenamos el vector de manera contraria
47                 int aux = v[j];
48                 v[j] = v[j+1];
49                 v[j+1] = aux;
50             }
51
52     clock_t tini;    // Anotamos el tiempo de inicio
53     tini=clock();
54
55     for (int i = 0 ; i < 1000 ; i++)
56         ordenar(v,tam);    // v esta ordenado de mayor a menor, que es el peor caso
57
58     clock_t tfin;    // Anotamos el tiempo de finalización
59     tfin=clock();
60
61     // Mostramos resultados reduciendo el error en 1000
62     cout << tam << "\t" << ((tfin-tini)/(double)CLOCKS_PER_SEC)/1000.0 << endl;
63
64     delete [] v;    // Liberamos memoria dinámica
65 }
```

# EJERCICIO 1 : Ordenación de la burbuja

## 2.- Hardware usado:

### 2.1- CPU

**vendor\_id** : GenuineIntel  
**model name** : Intel(R) Core(TM) i3 CPU M 330 @ 2.13GHz  
**cpu MHz** : 933.000

### 2.2- Velocidad de Reloj

**Versión** : hwclock de util-linux 2.20.1  
mar 11 oct 2016 23:00:11 CEST -0.563198 segundos

### 2.3- Memoria RAM

**MemTotal** : 3907668 kB  
**SwapTotal** : 4049916 kB

## 3.- Sistema Operativo

Ubuntu 14.04.3 LTS  
**Arquitectura** : x86\_64 (64 bits)

## 4.- Compilador usado y opciones de compilación

gcc - GNU project C and C++ compiler  
**Opción de compilación** : g++ -o <nombre\_ejecutable> <ejecutable.cpp>  
g++ -o ordenacion ordenacion.cpp

# EJERCICIO 1 : Ordenación de la burbuja

## 5.- Desarrollo completo del cálculo de la Eficiencia teórica

### EFICIENCIA TEÓRICA

CÁLCULO DE EFICIENCIA TEÓRICA

DEL ALGORITMO:

```
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]) {  $\in O(1)$   
                int aux = v[j];  $\in O(1)$   
                v[j] = v[j+1];  $\in O(1)$   
                v[j+1] = aux;  $\in O(1)$   
            }
```

$\sum_{j=0}^{n-i-2} 1$   
Es el orden de eficiencia del bloque if

Desarrollo bucle for más interno:

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

Desarrollo bucle for externo:

$$\sum_{i=0}^{n-2} n-i-1 = \underbrace{\sum_{i=0}^{n-2} n}_{(1)} - \underbrace{\sum_{i=0}^{n-2} i}_{(2)} - \underbrace{\sum_{i=0}^{n-2} 1}_{(3)} =$$

$$(1) \quad (n-1)n = n^2 - n$$

$$(2) \quad n \left( \frac{n-2}{2} \right) = \frac{n^2}{2} - \frac{2n}{2} = \frac{n^2}{2} - n$$

$$(3) \quad (n-1)$$

$$\begin{aligned} &= (n^2 - n) - \left( \frac{n^2}{2} - n \right) - (n-1) = n^2 - n - \frac{n^2}{2} + n - n + 1 = \\ &= \frac{n^2}{2} - \frac{2n}{2} + 1 \in \underline{\underline{O(n^2)}} \end{aligned}$$

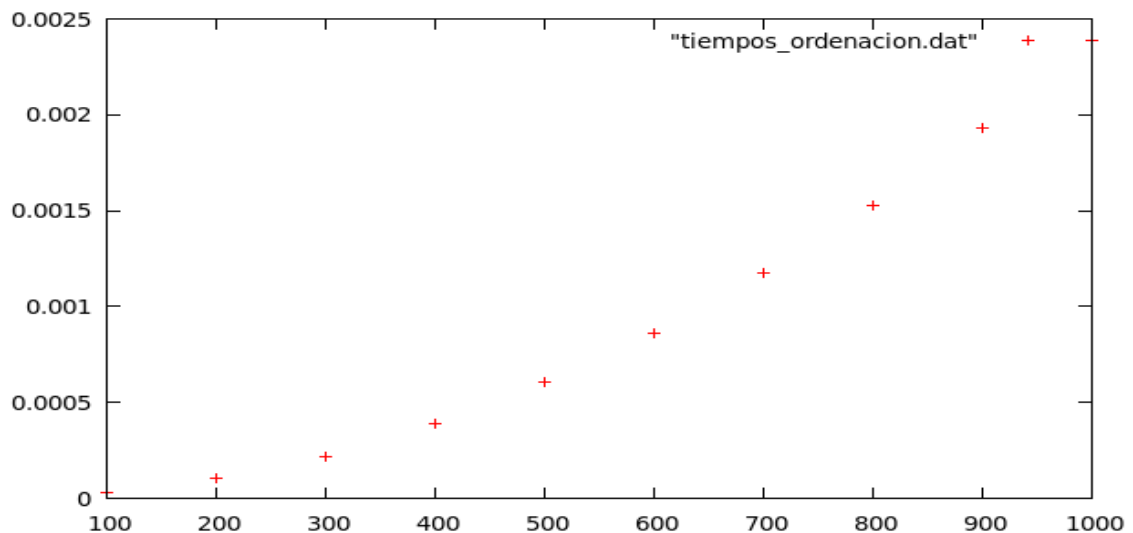
# EJERCICIO 1 : Ordenación de la burbuja

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

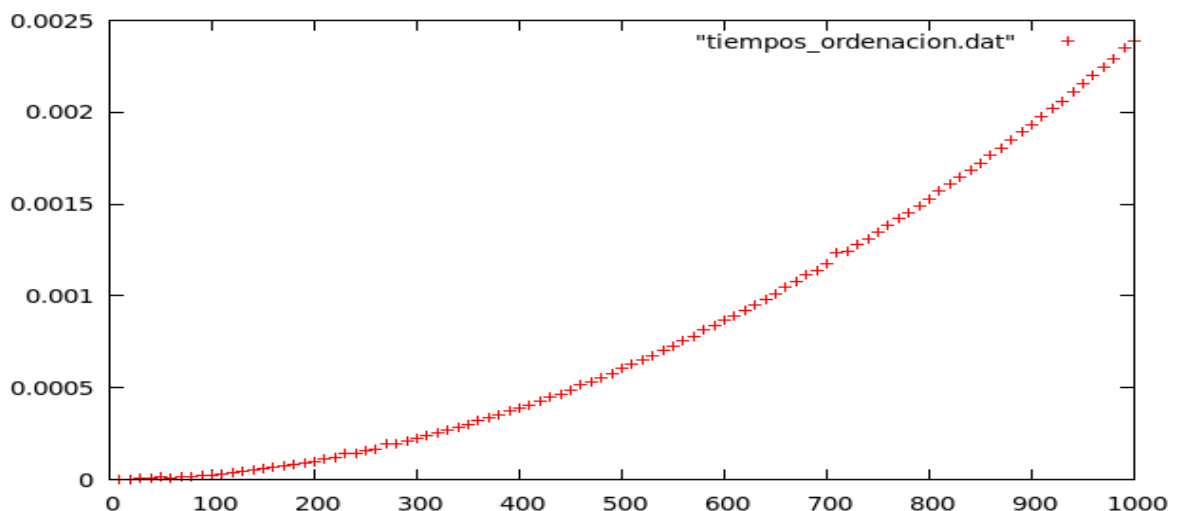
Para ésta parte he ejecutado el programa con los siguientes valores para tamaño máximo del vector : 100, 600, 1100 y 6000.

```
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic
io1$ ./ordenacion 100 100
100      2.9505e-05
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic
io1$ ./ordenacion 600 100
600      0.000903693
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic
io1$ ./ordenacion 1100 100
1100     0.00300963
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic
io1$ ./ordenacion 6000 100
6000     0.0869036
```

### 6.1- Gráfica de tiempos para la ejecución del programa con tamaño de vector = 100

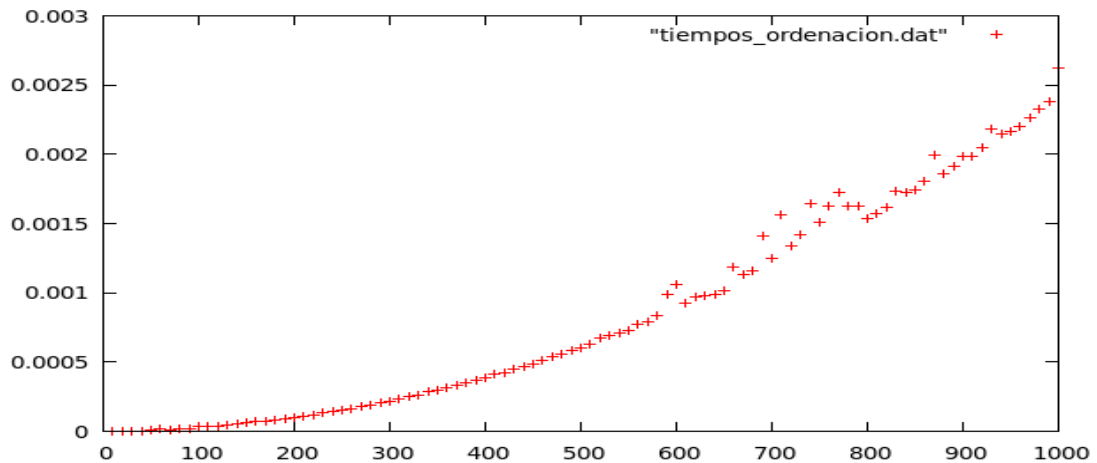


### 6.2- Gráfica de tiempos para la ejecución del programa con tamaño de vector = 600

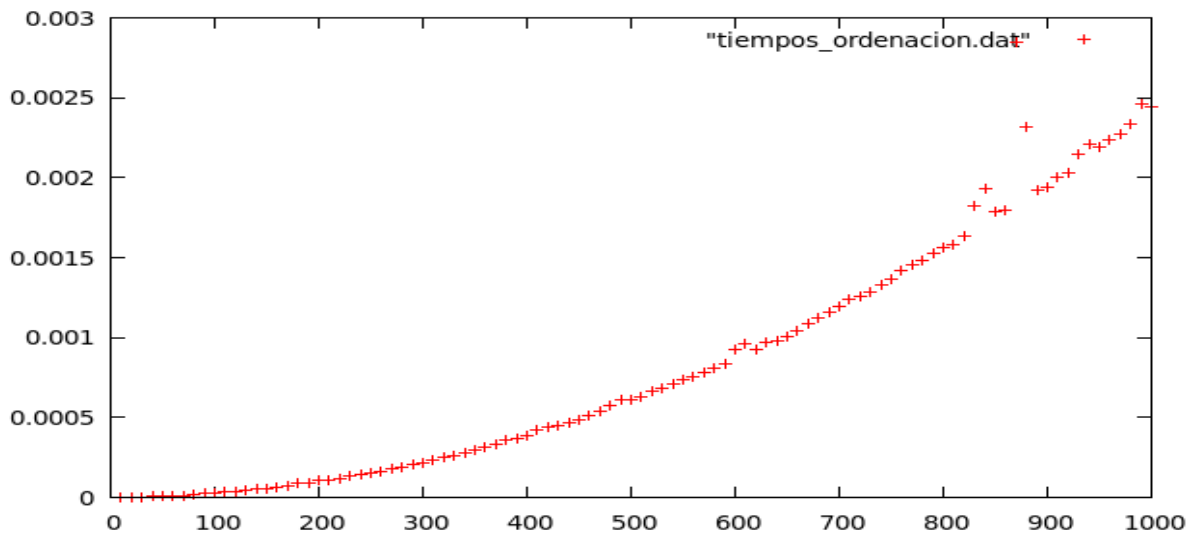


# EJERCICIO 1 : Ordenación de la burbuja

## 6.3- Gráfica de tiempos para la ejecución del programa con **tamaño de vector = 1100**



## 6.4- Gráfica de tiempos para la ejecución del programa con **tamaño de vector = 60000**



\*Con tamaño del vector = 10000 en adelante mi ordenador tarda demasiado en ejecutarlo. Por eso he probado hasta 6000

\*Como se ve, en el caso en el que el vector tiene tamaño = 600, la gráfica se ajusta más a la forma que tiene que tener, dada por su valor de Eficiencia teórica ( $O(n^2)$ ).

Para el Ejercicio 2 utilizaré la ejecución con tamaño del vector = 6000.