EJERCICIO 2 : Ajuste en la ordenación de la burbuja

Autor : Jesús Ruiz Castellano, 76439001-L

1.- Código fuente : ordenacion.cpp

```cpp
#include <iostream>
#include <ctime>    // Recursos para medir tiempos
#include <cstdlib>  // Para generación de números pseudoaleatorios

using namespace std;

void ordenar(int *v, int n) {          // Algoritmo ordenacion por burbuja
        for (int i=0; i < n-1; i++)
                for (int j=0; j < n-i-1; j++)
                        if (v[j] > v[j+1]) {
                                int aux = v[j];
                                v[j] = v[j+1];
                                v[j+1] = aux;
                        }
}


void sintaxis()
{
  cerr << "Sintaxis:" << endl;
  cerr << "  TAM: Tamaño del vector (>0)" << endl;
  cerr << "  VMAX: Valor máximo (>0)" << endl;
  cerr << "Se genera un vector de tamaño TAM con elementos aleatorios en [0,VMAX[" << endl;
  exit(EXIT_FAILURE);
}

int main(int argc, char * argv[])
{
  // Lectura de parámetros
  if (argc!=3)
    sintaxis();
  int tam=atoi(argv[1]);     // Tamaño del vector
  int vmax=atoi(argv[2]);    // Valor máximo
  if (tam<=0 || vmax<=0)
    sintaxis();
```

```cpp
  // Generación del vector aleatorio
  int *v=new int[tam];       // Reserva de memoria
  srand(time(0));            // Inicialización del generador de números pseudoaleatorios
  for (int i=0; i<tam; i++)  // Recorrer vector
    v[i] = rand() % vmax;    // Generar aleatorio [0,vmax[


  for (int i=0; i < tam-1; i++)
        for (int j=0; j < tam-i-1; j++)
                if (v[j] < v[j+1]) {    // ordenamos el vector de manera contraria
                        int aux = v[j];
                        v[j] = v[j+1];
                        v[j+1] = aux;
                }

  clock_t tini;    // Anotamos el tiempo de inicio
  tini=clock();

  for (int i = 0 ; i < 1000 ; i++)
      ordenar(v,tam);           // v esta ordenado de mayor a menor, que es el peor caso

  clock_t tfin;    // Anotamos el tiempo de finalización
  tfin=clock();

  // Mostramos resultados reduciendo el error en 1000
  cout << tam << "\t" << ((tfin-tini)/(double)CLOCKS_PER_SEC)/1000.0 << endl;

  delete [] v;     // Liberamos memoria dinámica
}
```

EJERCICIO 2 : Ajuste en la 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 23:20:10 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)

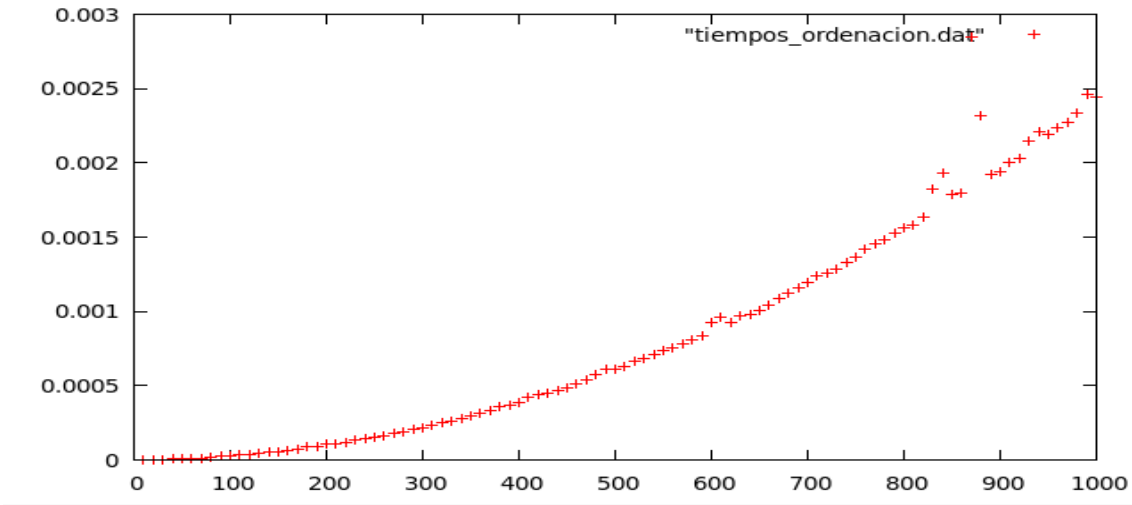
4.- Compilador usado y opciones de compilación

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

EJERCICIO 2 : Ajuste en la ordenación de la burbuja

5.- Parámetros usados para el cálculo

Gráfica inicial para tamaño del vector = 6000



Órdenes utilizadas para hacer el ajuste por regresión, en gnuplot

```
gnuplot> f(x)= a*x*x + b*x + c
gnuplot> fit f(x) "tiempos_ordenacion.dat" via a, b, c
```

Datos obtenidos tras la última orden

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

a               = 2.73472e-09      +/- 6.871e-11    (2.513%)
b               = -2.38034e-07     +/- 7.163e-08    (30.09%)
c               = 3.48922e-05      +/- 1.567e-05    (44.92%)


correlation matrix of the fit parameters:

                a      b      c
a               1.000
b              -0.969  1.000
c               0.753 -0.871  1.000
```

EJERCICIO 2 : Ajuste en la ordenación de la burbuja

Orden para obtener las gráficas de f(x), definida antes, y la obtenida con los valores de " tiempos_ordenacion.dat "

```
gnuplot> plot "tiempos_ordenacion.dat", f(x)
```

Gráfica obtenida