EJERCICIO 6 : Influencia del proceso de compilación

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 6 : Influencia del proceso de compilación

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
jue 13 oct 2016 10:22:05 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++ -O3 -o <nombre_ejecutable> <ejecutable.cpp>
g++ -O3 -o ordenacion_optimizado ordenacion.cpp

EJERCICIO 6 : Influencia del proceso de compilación

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

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

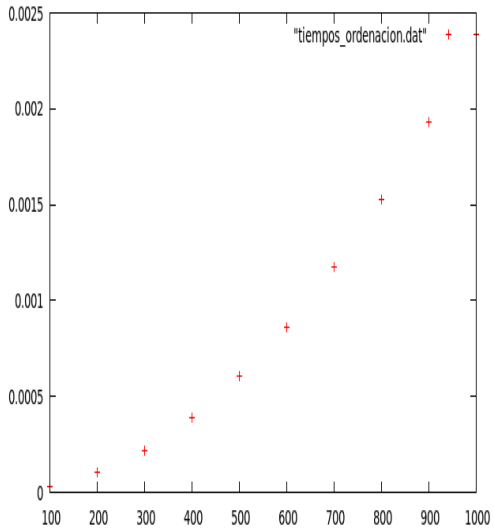
```
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic
io1$ ./ordenacion 100 100
100     2.9505e-05
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic
io1$ ./ordenacion 600 100
600     0.000903693
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic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

Para la parte compilada con -O3 si me ha aceptado el valor 10000, pero no 30000.

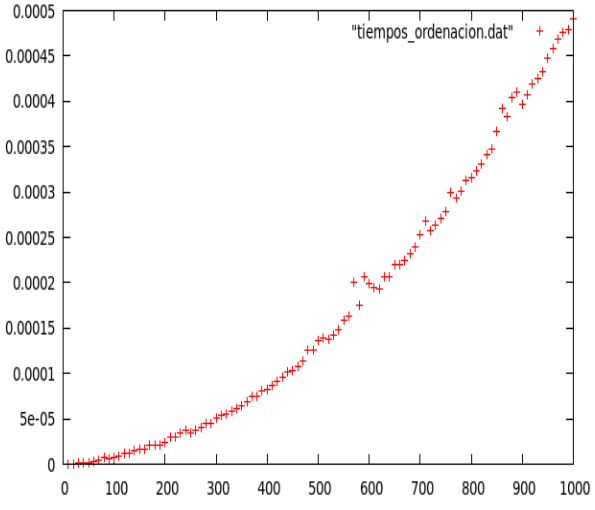
```
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic
io6$ ./ordenacion_optimizado 100 100
100     1.0581e-05
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic
io6$ ./ordenacion_optimizado 600 100
600     0.000182883
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic
io6$ ./ordenacion_optimizado 1100 100
1100    0.000589921
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic
io6$ ./ordenacion_optimizado 6000 100
6000    0.0171533
jesus@jesus-Aspire-xxxx:~/Escritorio/Jesus/2º/ED/Practicas/P1-Eficiencia/Ejercic
io6$ ./ordenacion_optimizado 10000 100
10000   0.0478895
```

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

Compilación normal



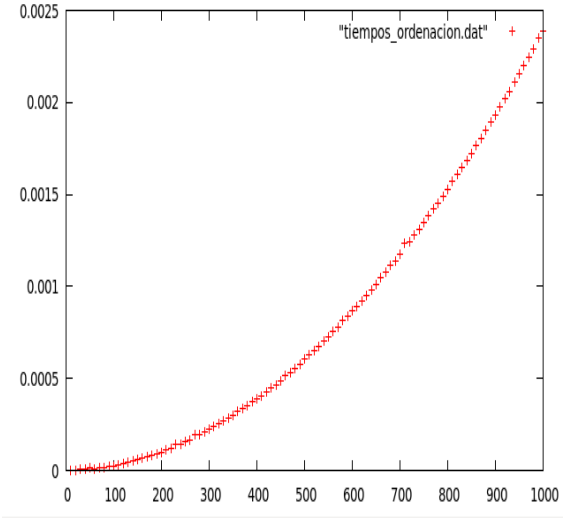
Compilación -O3



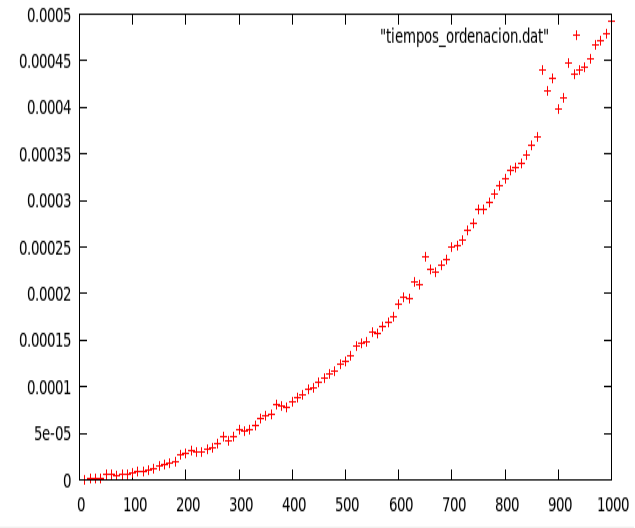
EJERCICIO 6 : Influencia del proceso de compilación

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

Compilación normal

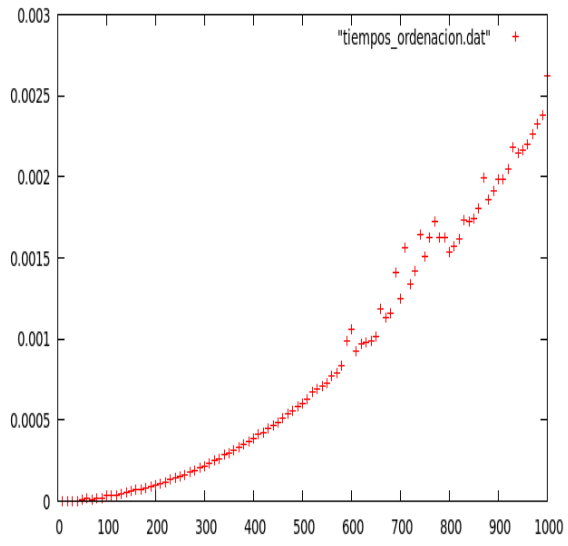


Compilación -O3

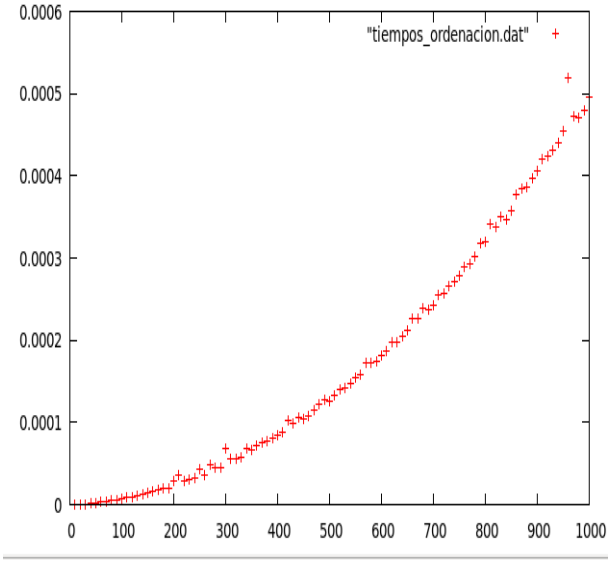


5.3- Gráfica de tiempos para la ejecución del programa con tamaño de vector = 1100

Compilación normal



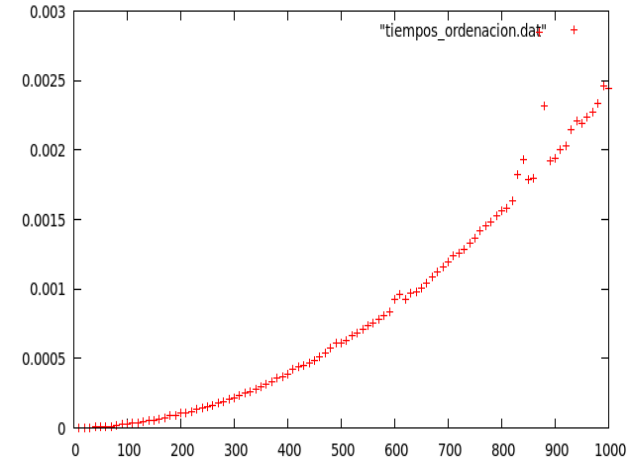
Compilación -O3



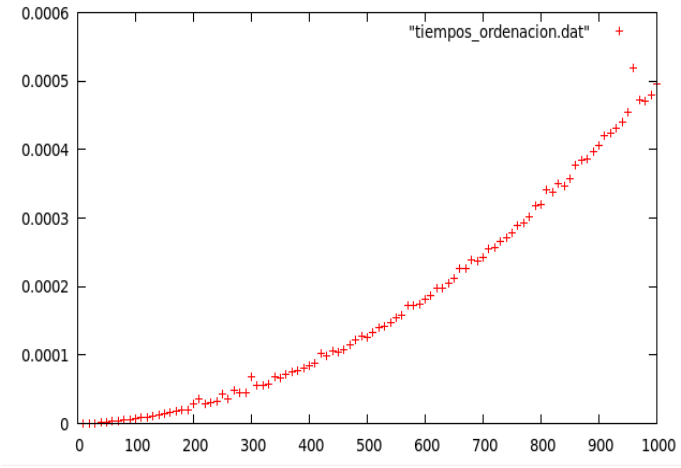
EJERCICIO 6 : Influencia del proceso de compilación

5.4- Gráfica de tiempos para la ejecución del programa con tamaño de vector = 6000

Compilación normal

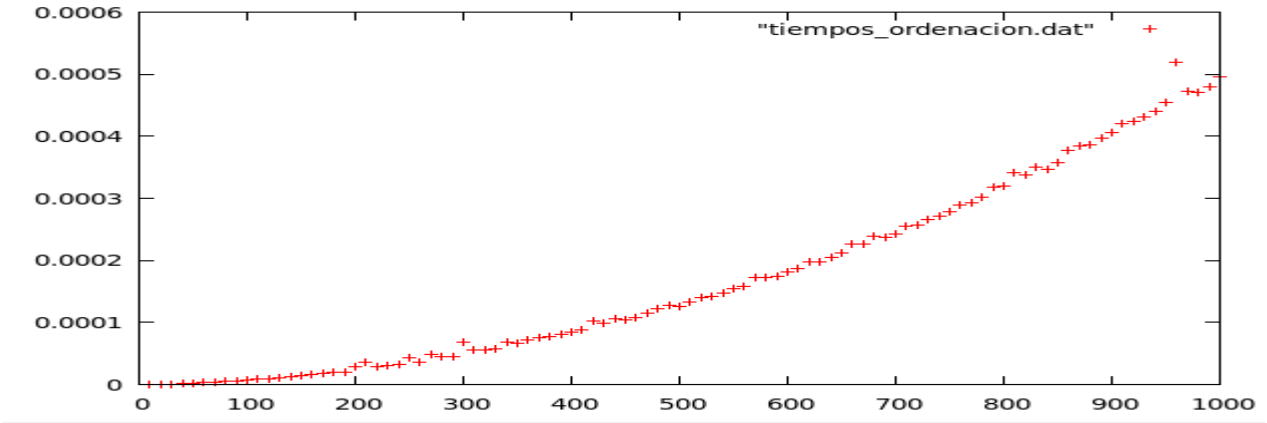


Compilación -O3



5.5- Gráfica de tiempos para la ejecución del programa con tamaño de vector = 10000

Sólo compilación -O3



***Con tamaño del vector > 10000 mi ordenador tarda demasiado en ejecutarlo. Por eso he probado hasta 10000**

***Como vemos, con la optimización en la compilación se consigue una mejora en cuanto al tiempo. Aumenta más rápido con la compilación -O3.**