1. INTRODUCCIÓN

Los objetivos de este guion de prácticas son los siguientes:

1. Aprender a calcular la eficiencia teórica de un código junto con su orden de eficiencia en el peor de los casos.
2. Aprender a obtener la eficiencia empírica de un código.
3. Aprender a realizar un ajuste de la curva de eficiencia teórica a la empírica.

Para ello, trabajaremos en Ubuntu (Linux) y para representar tanto la eficiencia teórica como empírica usaremos el comando >gnuplot que previamente hemos tenido que instalar mediante 'sudo apt-get install gnuplot-x11'

Ejercicio 1: Ordenación de la burbuja

1. Calcule la eficiencia teórica de este algoritmo. A continuación replique el experimento que se ha hecho antes (búsqueda lineal) con este nuevo código. Debe:

• Crear un fichero ordenacion.cpp con el programa completo para realizar una ejecución del algoritmo.

• Crear un script ejecuciones_ordenacion.csh en C-Shell que permite ejecutar varias veces el programa anterior y generar un fichero con los datos obtenidos.

• Usar gnuplot para dibujar los datos obtenidos en el apartado previo.

Los datos deben contener tiempos de ejecución para tamaños del vector 100, 600, 1100,…, 30000.

Pruebe a dibujar superpuestas la función con la eficiencia teórica y la empírica. .Que sucede?

1) Creamos el fichero 'ordenacion.cpp' en el que incluimos tanto el programa completo como el algoritmo de ordenación de la burbuja.
2) Generamos el ejecutable mediante:

   g++ ordenacion.cpp –o ordenacion

3) Modificamos los campos del archivo “ejecuciones_ordenacion.csh” con los datos que nos han pedido
4) Para que el resultado de todas las ejecuciones se guarden en el archivo “ordenacion.dat”, ejecutamos lo siguiente:

   ./ejecuciones_ordenacion.csh

5) Calculamos ahora la eficiencia teórica del algoritmo:

```
void ordenar(int *v, int n) {
   for (int i=0; i<n-1; i++)        //Se realizan n-1 iteraciones -> O(n)
      for (int j=0; j<n-i-1; j++)   //Se realizan n-i-1 iteracciones -> O(n)
         if (v[j]>v[j+1]) {         //O(1)
            int aux = v[j];         //O(1)
            v[j] = v[j+1];          //O(1)
            v[j+1] = aux;           //O(1)
         }
}
/*
Cabe destacar que las últimas líneas del código son O(1) ya que se realizan
accesos al vector y asignaciones
```

Por la regla de la suma, el cuerpo a partir del if es O(1): O(max(1,1,1))=O(1)

Por la regla del producto, calculamos la eficiencia del código:

O(n*n*1)=O(n^2) → Eficiencia cuadrática

6) Iniciamos gnuplot con el comando >gnuplot y representamos:

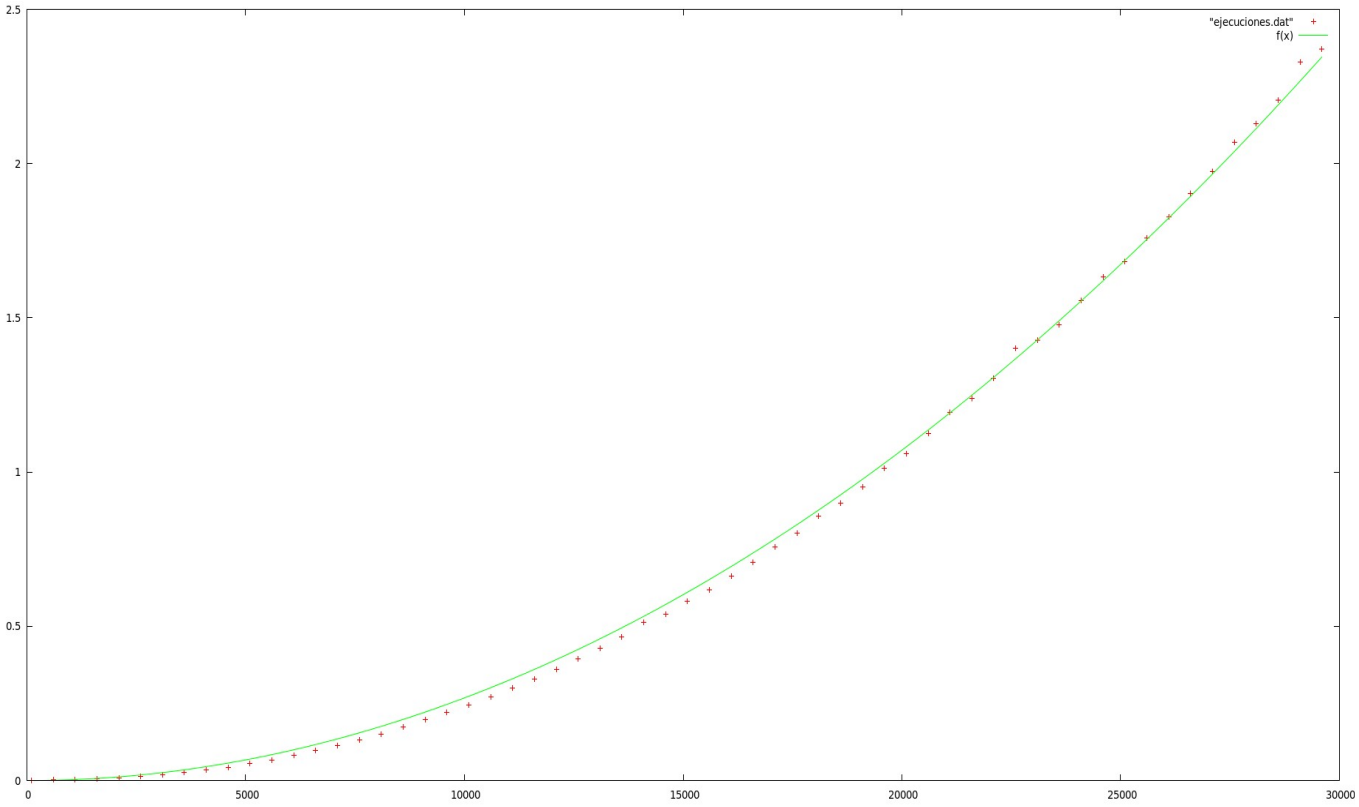
Como nos sale una eficiencia cuadrática, definimos f(x) como O(n^2)

gnuplot> f(x)=a*x*x

gnuplot> fit f(x) “ordenacion.dat” via a

gnuplot> plot “ordenacion.dat”, f(x)

La gráfica resultante es:



Como podemos observar, tanto la eficiencia teórica calculada como la empírica se ajustan

Características del ordenador:

Fabricante	Lenovo
Procesador	Intel Core i7 4710HQ
Sistema Operativo	Ubuntu
Versión SO	14.04
RAM	8192MB
CPU	64 bits