Práctica 1: Eficiencia

Antonio Manuel Fernández Cantos

12 de octubre de 2015

Índice

1. Introducción

La práctica consiste en calcular la eficiencia **teórica y empírica** de un código en c++ y **realizar un ajuste de la curva de eficiencia teórica a la empírica.** Se utilizará la biblioteca **ctime** para poder obtener los resultados empíricos. Dentro de la biblioteca ctime tenemos la función **clock()** que devuelve el número de ticks que han transcurrido desde un momento determinado, es esta función la que usaremos para medir la diferencia de tiempo entre el inicio del algoritmo y su finalización.

2. Componentes utilizados

En el cálculo empírico, el algoritmo tardará más o menos en función de:

- **Hardware usado**:
  - CPU
  - RAM
  - HDD
- **Sistema Operativo**
- **Compilador (y sus opciones de compilación)**

- **Bibliotecas**

Todos estos componentes se tienen en cuenta cuando se obtiene el tiempo que tarda nuestro algoritmo en ejecutar todas las sentencias. Dependiendo de la potencia de nuestro ordenador y de las librerias usadas, el algoritmo tardará más o menos. Para la realización del cálculo empírico de los ejercicios, he usado los siguientes componentes:

- **Hardware usado**:
  - Procesador: 8x Intel(R) Core(TM) i7-3630QM CPU@2.40MHz
  - RAM: 6GB
  - CPU clock: 1200 MHz
  - HDD: 750GB
- **Sistema Operativo**: Ubuntu 14.04.3 LTS
- **Compilador**: GCC sin opciones de compilación
- **Bibliotecas**:
  - iostream (E/S)
  - ctime (Para medir el tiempo de ejecución de un algoritmo)
  - cstdlib (Para generar números pseudoaleatorios)

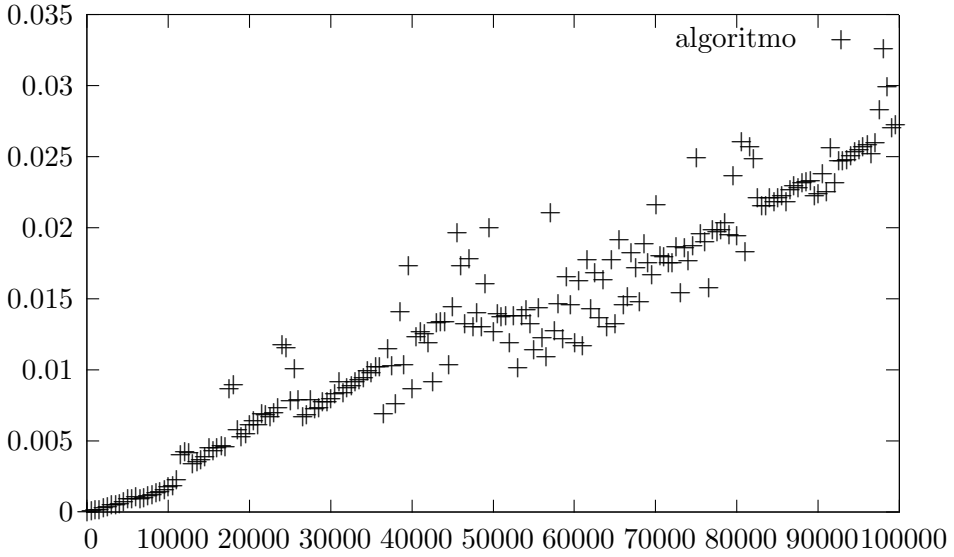
3. Ejercicio 8: Ordenación por Mezcla

3.1. Introducción

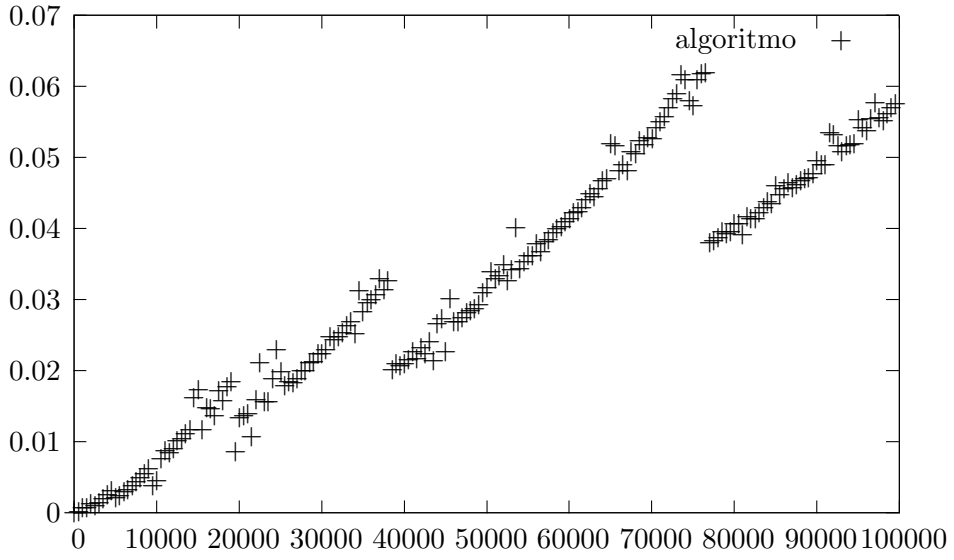
En este ejercicio analizaré la eficiencia del algoritmo mergesort con el inserción. En el algoritmo mergesort se hace una llamada al inserción cuando quedan menos de 100 elementos por ordenar.

3.2. Eficiencia empírica

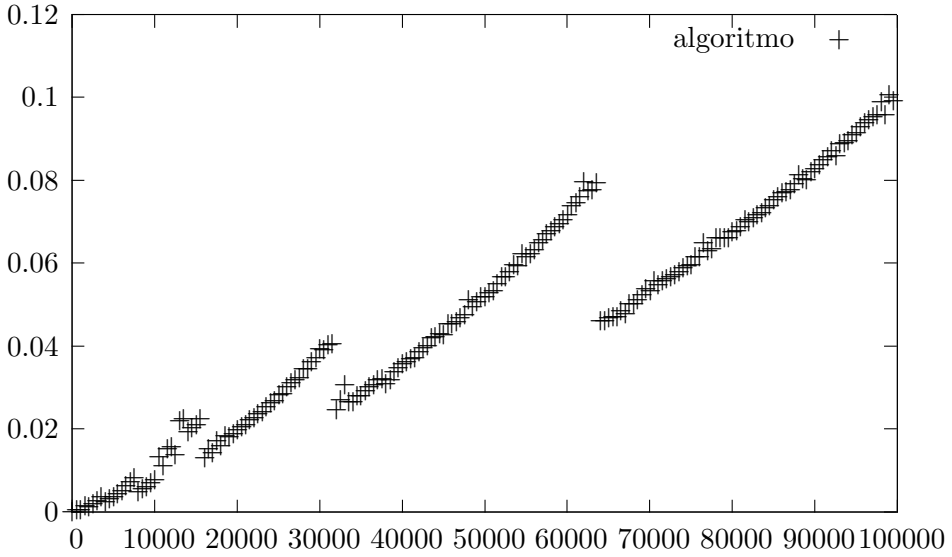
Como ya sabemos la eficiencia teórica del algoritmo mergesort es $O(n * log(n))$ y la eficiencia teórica del algoritmo por inserción es $O(n^2)$. En el algoritmo por inserción apenas gastaremos tiempos ya que trabaja en el código con muy pocos datos. La gráfica de la eficiencia empírica del ejecutable para un $UMBRAL_MS = 100$ es la siguiente:



La eficiencia empírica para um $UMBRAL_MS = 600$ es la siguiente:



Y por útlimo para un $UMBRAL_MS = 1000$



Como podemos observar en las tres gráficas, a medida que aumenta la importancia (el intervalo de datos que maneja) el algoritmo de iserción, aumenta el tiempo de ejecución del algoritmo.