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.

PASOS SEGUIDOS:

Ejercicio1:

1) Hemos instalado el intérprete csh ya que nos mostraba el siguiente error:

   /bin/csh:bad interpreter: No such file or directory

   Solución: >sudo apt-get install

2) Pasamos el código del algoritmo burbuja programa.
3) Ejecutamos el siguiente comando:

   g++ ordenacion.cpp –o ordenación

4) Creamos el archivo “ejecuciones_ordenacion.csh” con los datos:

   inicio=100; fin=30000; incremento=500.

   Set ejecutable=ordenación

   Set salida=ordenación.dat

5) Ejecutamos el archivo ./ejecuciones_ordenacion.csh

   Gracias a esto, el resultado de todas las ejecuciones se guardan en el archivo “ordenacion.dat”

6) Ejecutamos el siguiente comando para instalar gnuplot:

   sudo apt-get install gnuplot

7) Iniciamos gnuplot con el comando >gnuplot

   El cálculo de la eficiencia teórica lo calculamos así:

   /*

   Código del algoritmo

   */

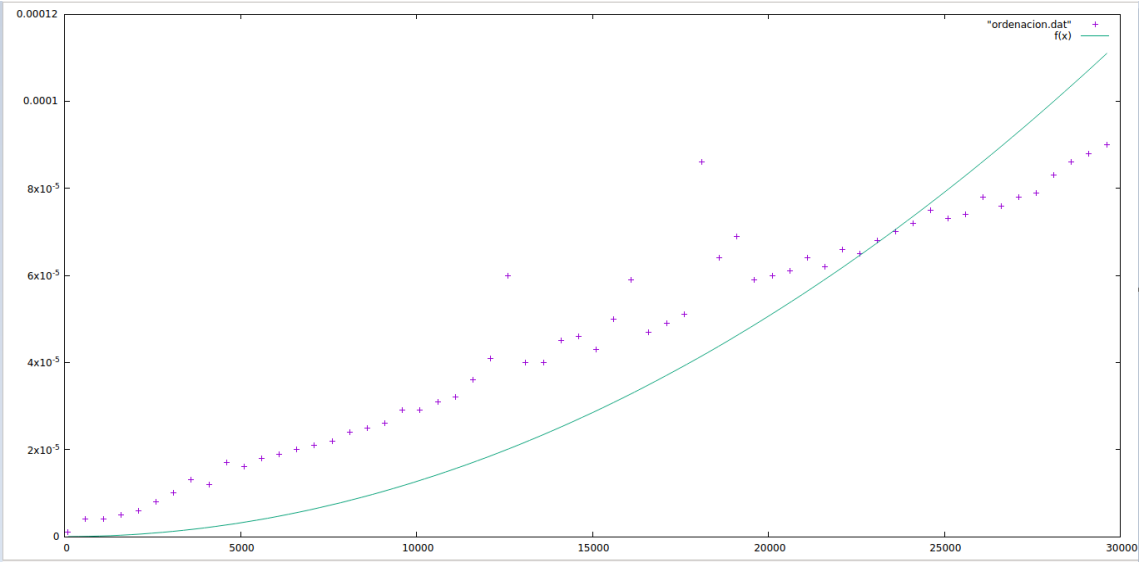
Como nos sale una eficiencia cuadrática, definimos f(x) como O(n^2)

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

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

   gnuplot> plot “ordenacion.dat”, f(x)

La gráfica resultante es:



En la gráfica anterior se representan la eficiencia teórica O(n^2) con una línea continua y la eficiencia empírica con puntitos.

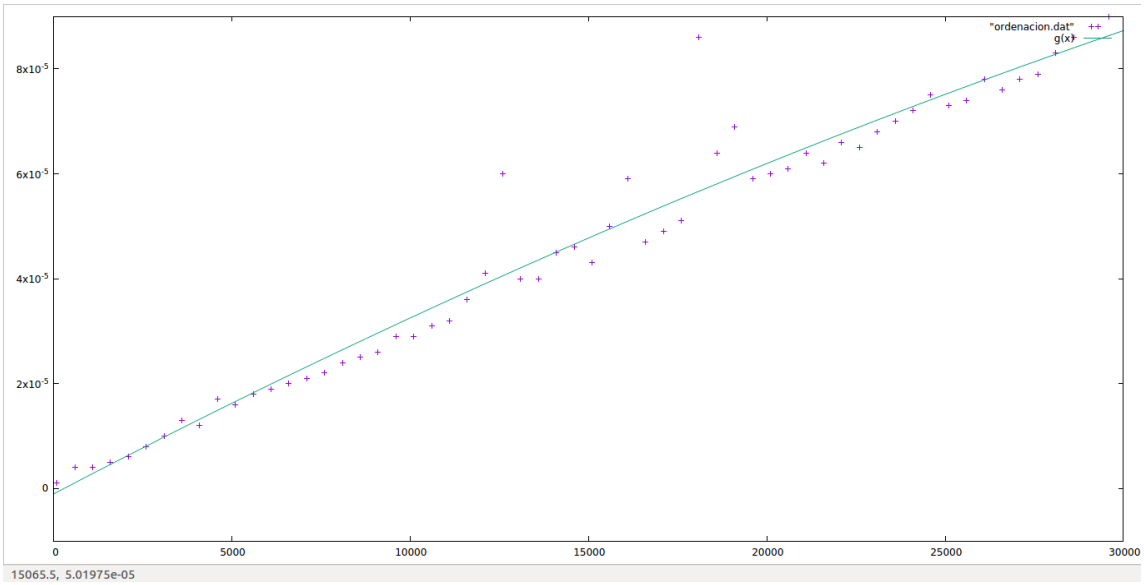
Ejercicio2: Ajuste en la ordenación de la burbuja

Consideramos la función f(x)=a x^2 +bx+c , para ello en gnuplot introducimos:

gnuplot> g(x)=a*x*x +b*x+c

gnuplot> fit g(x) “ordenacion.dat” via a, b ,c

gnuplot> plot “ordenacion.dat”, g(x)



En la gráfica anterior se representan la eficiencia teórica f(x)=ax^2+bx+c con una línea continua y la eficiencia empírica con puntitos.

Observe tambien que los tiempos de ejecucion obtenidos dependen del sistema que este
usando por lo que debera acompañar sus informes siempre de datos que sean relevantes a
este efecto tales como tipo de ordenador, compilador usado, opciones que se han usado para
compilar, etc.

Ejercicio3:

1) Creamos un archivo llamado ejecuciones_desc.csh
2) Mantenemos los mismos valores de inicio, final e incremento del ejercicio anterior.
3) Cambiamos el resto de campos para que la salida se redireccione al archivo desc.dat
4) Ejecutamos el código mediante el comando:

./ejecuciones_desc.csh

5) El resto de pasos los realizaremos haciendo uso de gnuplot ya instalado anteriormente:

/*

Si el número de argumentos es distinto a 2 ó el tamaño del vector es negativo, llama a la función sintaxis() que muestra un mensaje de error. En otro caso, se hace un bucle for desde 0 hasta el tamaño del vector en el que se introducen números aleatorios en él (Los número son menores que el tamaño máximo ya que se usa %). Después, se llama a la función operación() que busca el valor en el vector. Finalmente se muestra en pantalla el tiempo de ejecución.

Ejercicio 4

Mejor caso posible: Relleno el vector con valores ordenados.

Sustituímos la función srand que rellena el vector aleatoriamente por un contador que se va incrementando y rellena el vector con números ordenados:

Código:

/*

Int contador=0;

For(int i=0; i<tam;i++){

v[i]=contador;

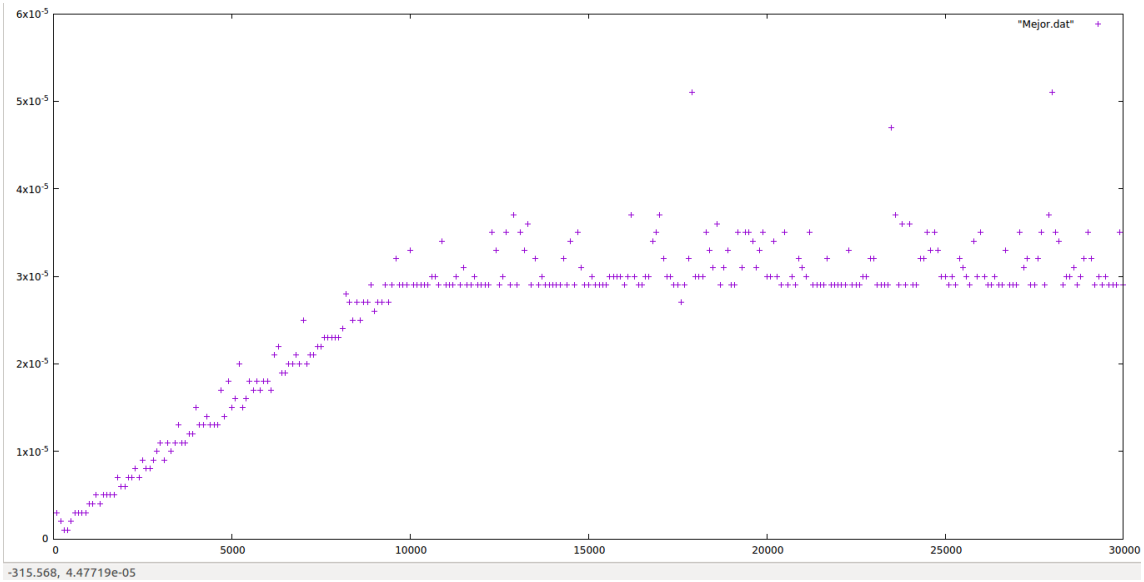
contador++;

}

*/

1) Lo valores de inicio, fin e incremento los mantenemos.
2) Compilamos el archivo mejorcasoposible.cpp
3) Ejecutamos Mejor.csh y obtenemos Mejor.dat con los tiempos

La gráfica resultante sale:

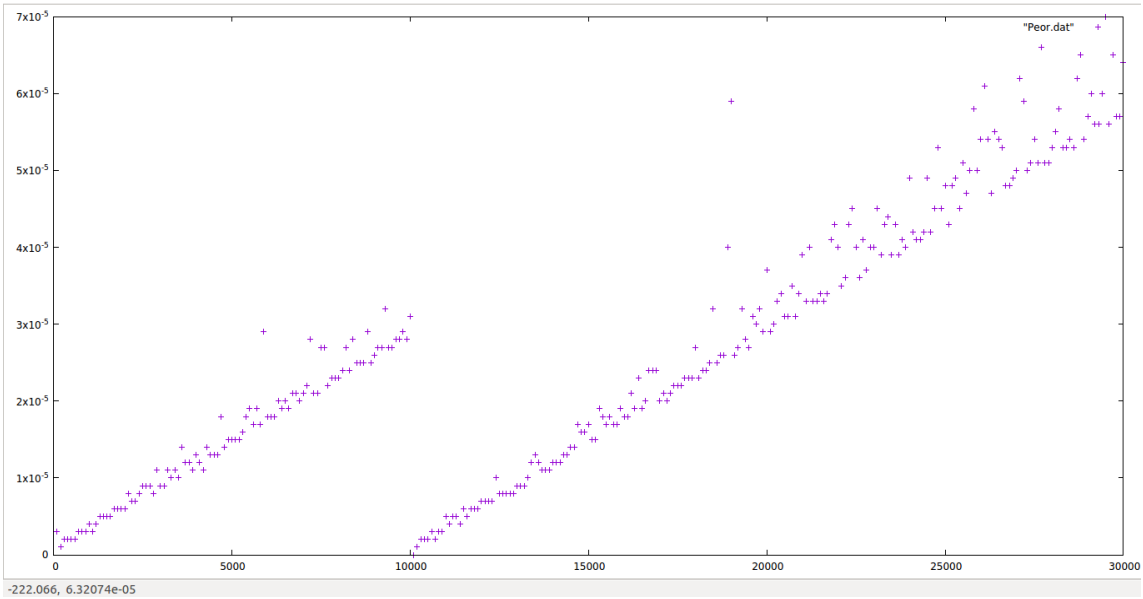


Peor caso posible: Relleno el vector con valores ordenados del final al principio.

Sustituímos la función srand que rellena el vector aleatoriamente por un contador que se va incrementando y rellena el vector con números ordenados pero en orden inverso:

Código:

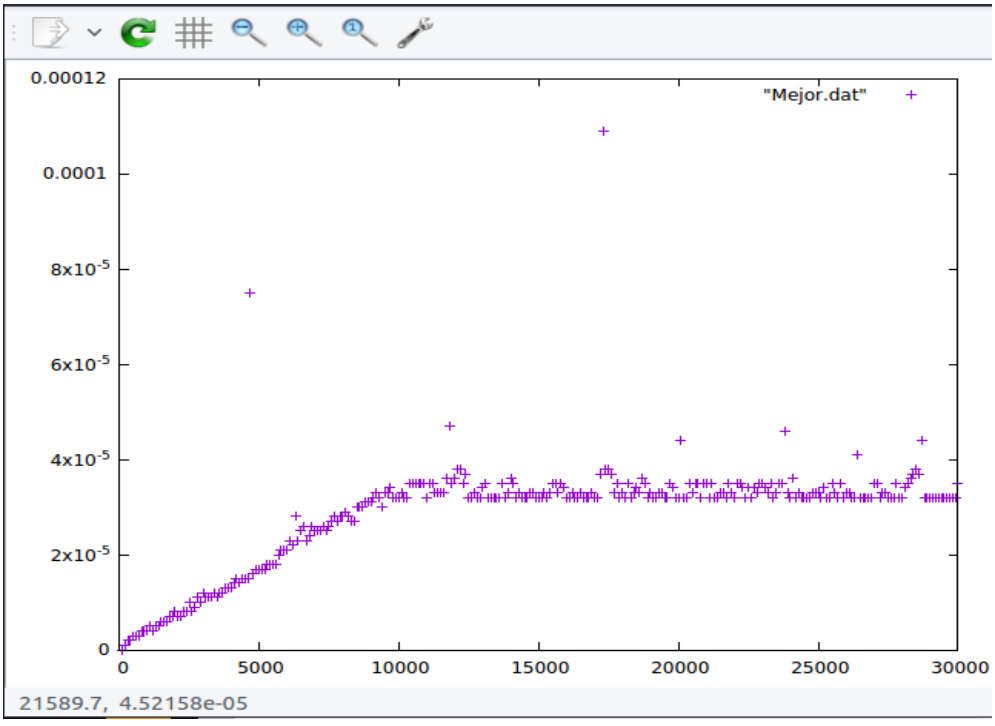
```
/*
Int contador=0;
For(int i=tam-1; i>0;i--){
        v[i]=contador;
        contador++;
}
*/
```



Ejercicio 5

1) Sustituímos el algoritmo de ordenación por el del ejercicio 5
2) Calculamos la eficiencia teórica de dicho algoritmo cuya notación O grande es O(n^2)

Gráfica resultante:



Como podemos observar, este algoritmo es más eficiente ya que los puntos están más juntos y el tiempo de ejecución del programa es de 3*