```
Arquitectura:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Orden de bytes:        Little Endian
CPU(s):                1
On-line CPU(s) list:   0
Hilo(s) por núcleo:    1
Núcleo(s) por zócalo:1
Socket(s):             1
Nodo(s) NUMA:          1
ID del vendedor:       GenuineIntel
Familia de CPU:        6
Modelo:                22
Stepping:              1
CPU MHz:               1861.849
BogoMIPS:              3723.69
caché L1d:             32K
caché L1i:             32K
caché L2:              1024K
NUMA node0 CPU(s):     0
```

Ejercicio 1: Ordenación de la burbuja .

El siguiente código realiza la ordenación mediante el algoritmo de la burbuja:

```
void ordenar(int *v, int n) {
    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;
            }
}
```

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. ¿Qué sucede?

Eficiencia teorica:

void ordenar(int *v, int n) {

for (int i=0; i<n-1; i++) $\leftarrow T(n)= \sum_{i=0}^{n-2} n-i-1 = \sum_{i=0}^{n-2} n - \sum_{i=0}^{n-2} i - \sum_{i=0}^{n-2} 1 =$

$= n(n-1)- n((n-1)/2)-(n-1) = n^2-n -(n^2/2)+(n/2)-n+1=$

$= (n^2-3n+2)/2 \in$ **$O(n^2)$**

for (int j=0; j<n-i-1; j++) $\leftarrow \sum_{j=0}^{n-i-2} 1= n-i-1$

if (v[j]>v[j+1]) { $\leftarrow$El if y lo que hay dentro vale O(1)

int aux = v[j];
v[j] = v[j+1];
v[j+1] = aux;
}
}

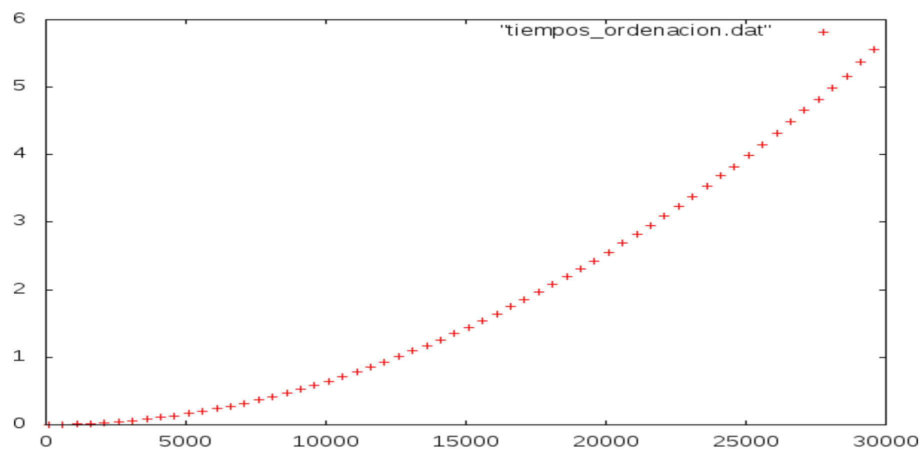
Puesto que $T(n)=(n^2-3n+2)/2 \in O(n^2)$ podemos afirmar que el orden de eficiencia del algoritmo de ordenación por burbuja es $O(n^2)$.

Eficiencia empírica:

Para medir el tiempo de ejecución del algoritmo de ordenación por burbuja, generamos un vector ordenado de mayor a menor , provocando de esta forma que se dé el peor caso posible. El programa tiene un argumentos que se le suministra en la línea de órdenes, el tamaño del vector.

- He creado el fichero **ordenacion.cpp** para realizar la ejecución del algoritmo.
- He creado el script ejecuciones_ordenacion.csh para ejecutar varias veces el programa anterior para tamaños del vector 100, 600, 1100, ..., 30000 y generar un fichero con los datos obtenidos.
- He usado gnuplot para dibujar los datos obtenidos en el apartado previo, resultando:

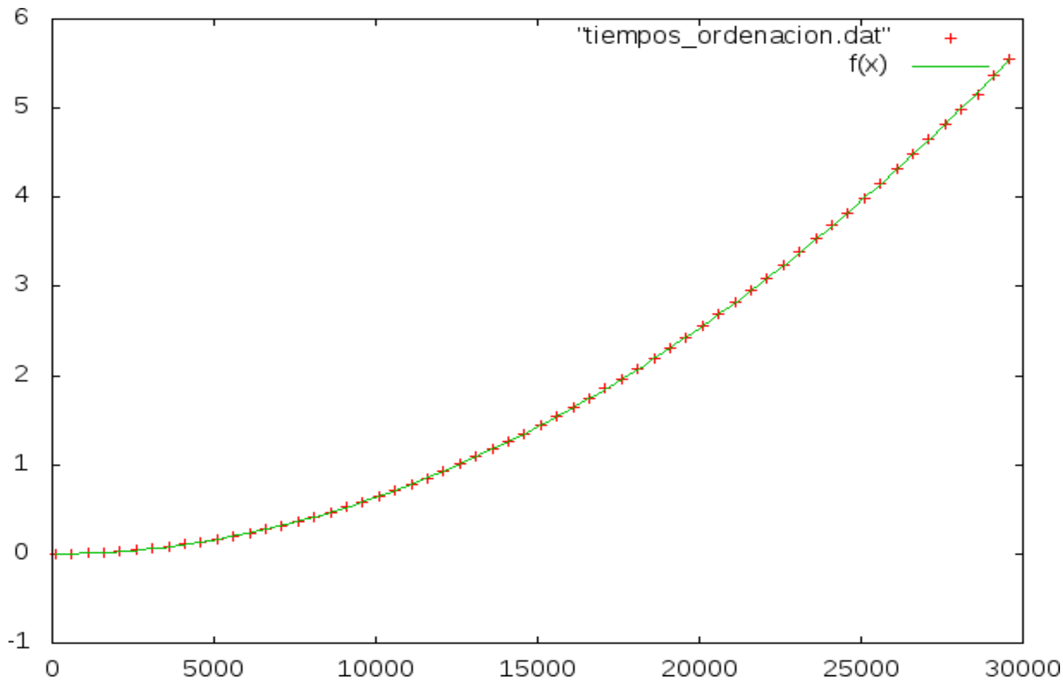
gnuplot> plot "tiempos_ordenacion.dat"



Dibujar superpuestas la función con la eficiencia teórica y la empírica:

Para el calculo del ajuste teorico con el empirico primero he declarado la funcion f(x) como "f(x)=a*x**2+b*x+c" ya que para el caso peor es $0(n^2)$.
Una vez que he declarado la funcion he pasado a ajustar con la orden "fit" para calcula la a, b y c. Despues he calculado el ajuste dibujando las dos graficas.
Pasos:

1) gnuplot> f(x)=a*x2+b*x+c**
2) fit f(x) "tiempos_ordenacion.dat" via a,b,c
3) gnuplot> plot "tiempos_ordenacion.dat" , f(x)



La grafica de la eficiencia teorica se ajusta con la de eficiencia empírica.