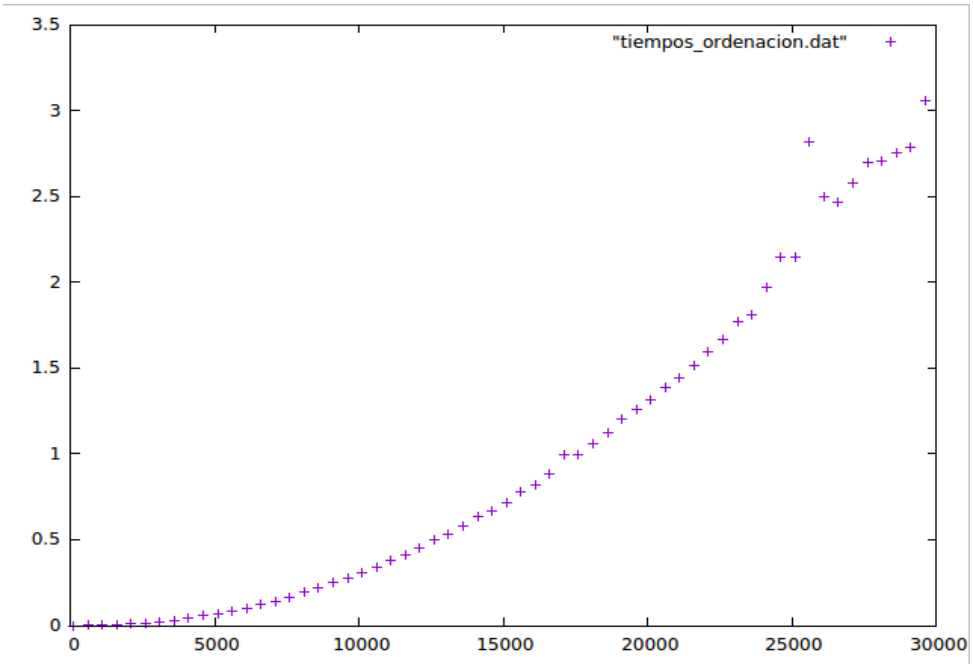
PRACTICA 1 ESTRUCTURA DE DATOS

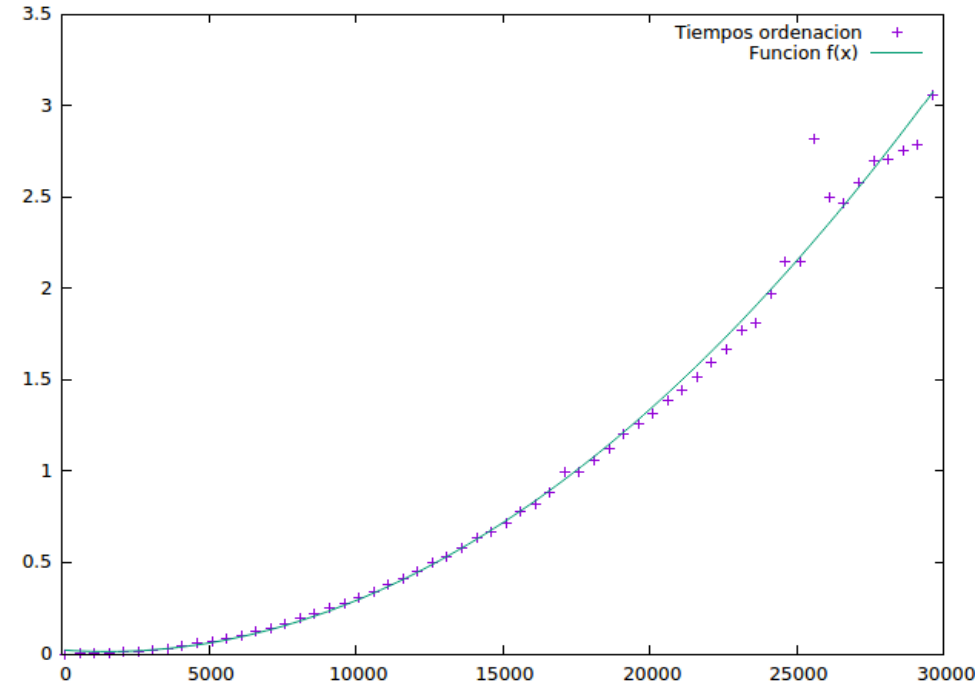
Ejercicio 1 y 2

El algoritmo de la burbuja tiene una eficiencia teórica de O(n^2).

Calculando los distintos tiempos de ejecución del ejecutable y representando gráficamente dichos datos mediante gnuplot y el archivo con los tiempos guardados obtenemos la siguiente tabla.



Observamos claramente que la curva se asemeja a una gráfica de la forma ax^2+bx+c. Para replicar el experimento declaramos nuestra función en gnuplot y mediante la opción fit ajustamos la función a nuestros tiempos y representamos ambas como muestra la siguiente tabla.

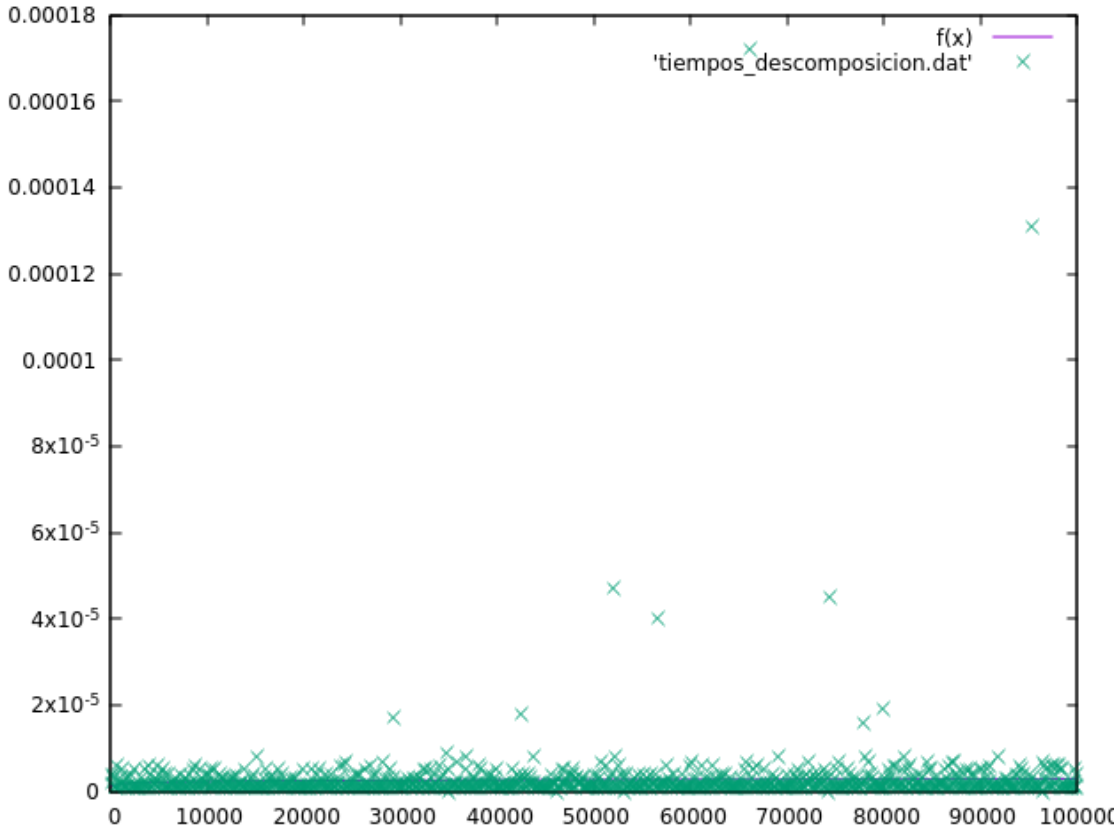


Observamos que se acoplan perfectamente.

Ejercicio 3

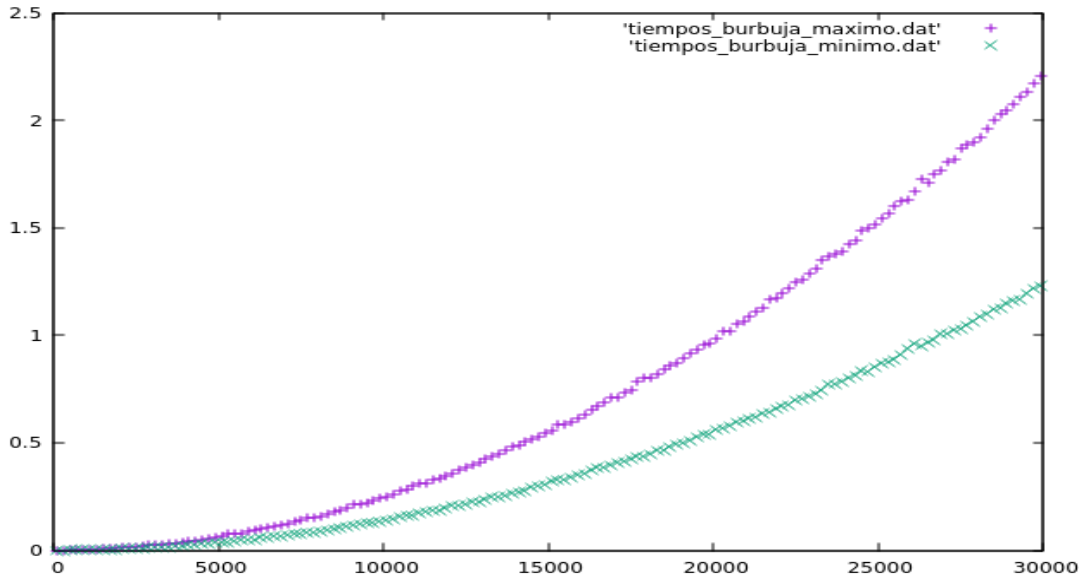
El algoritmo implementado en ejercicio_desc.cpp es un algoritmo de búsqueda realmente eficiente si le pasamos un vector ordenado pues no depende directamente del tamaño del vector. El algoritmo se implementa dividiendo el vector en dos y preguntando si la mitad del vector es nuestro elemento buscado, en caso de serlo se acaba la búsqueda, en caso de no serlo se orienta hacia la parte donde nuestro elemento estará, pues tenemos previamente el vector ordenado.

La eficiencia teórica del logaritmo es O(log(n)). Los problemas que plantea a la hora de calcular la eficiencia empírica es que anota tiempos muy cortos y cercanos al cero.

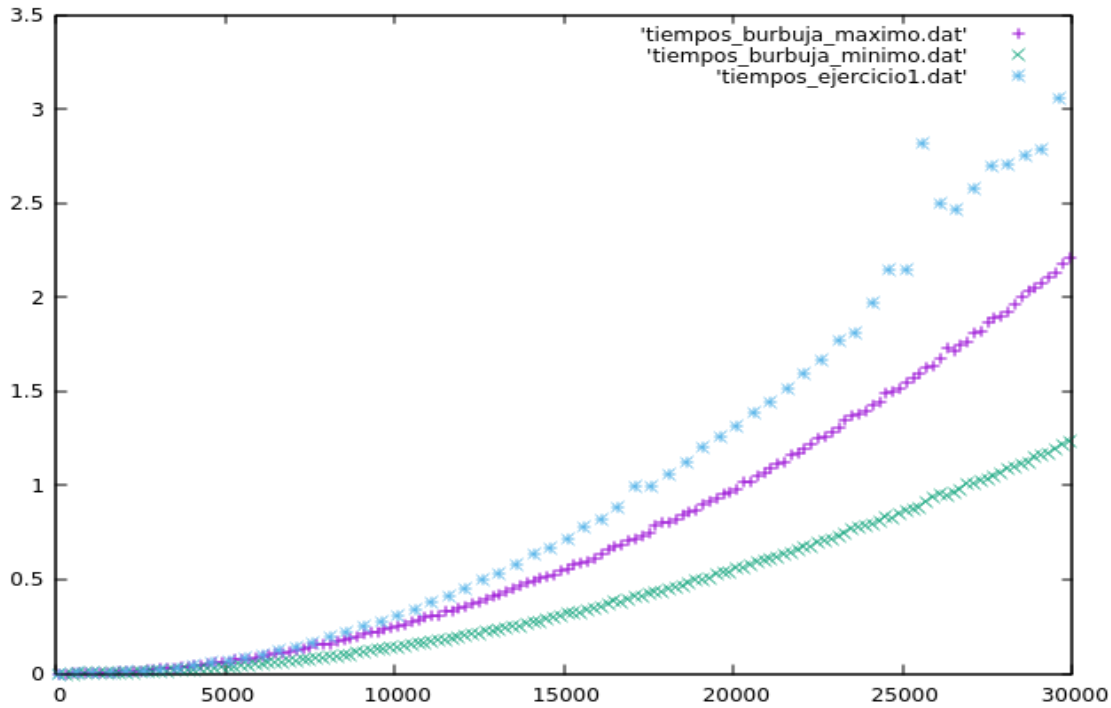


Ejercicio 4

Para llevar a cabo la eficiencia empírica en el mejor y en el peor de los casos llenamos un vector con los números totalmente ordenados para que no tenga que hacer ningún cambio y en el otro lo hacemos al contrario, introducimos los números de manera descendente para que tenga que cambiarlos todos. Los resultados de su eficiencia empírica y su representación gráfica es la siguiente:



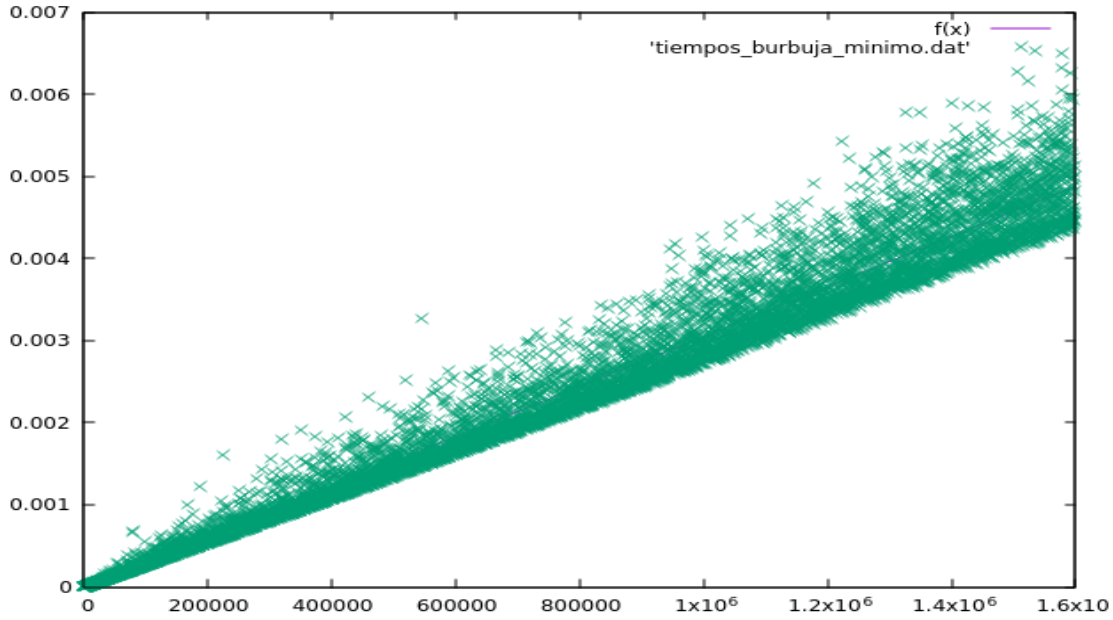
Si representamos gráficamente los datos obtenidos en este experimento y el realizado en el ejercicio 1 obtenemos la siguiente gráfica.



Observamos que el peor caso es cuando los números se introducen aleatoriamente.

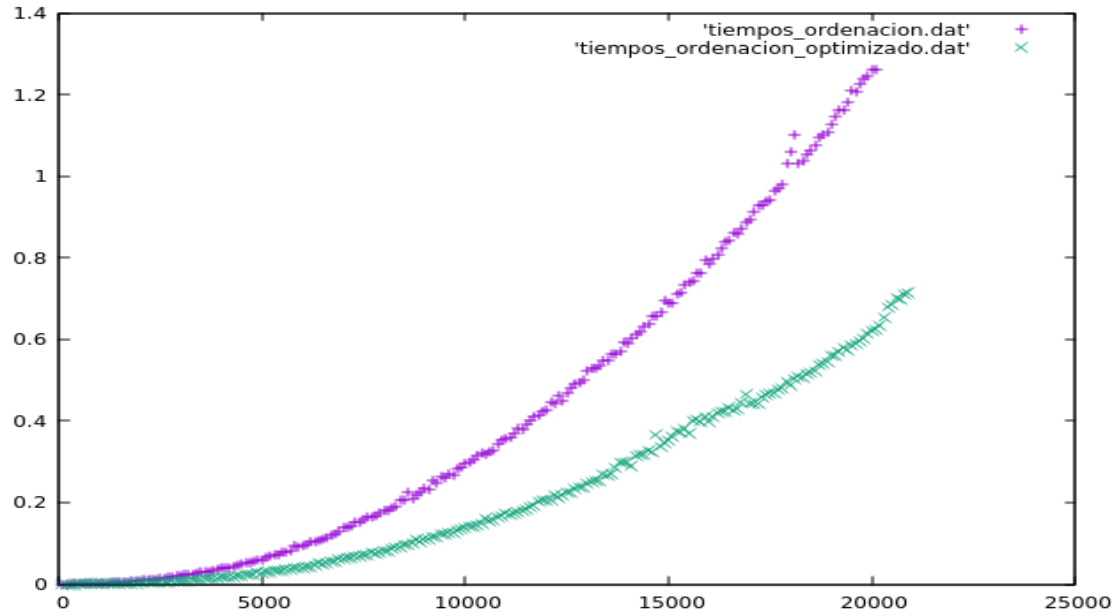
Ejercicio 5

En el algoritmo facilitado la mejora incluida permite que si el vector no esta modificado no se entre en el segundo bucle. Esto permite mejorar la eficiencia bastante, si le pasamos un vector ordenado nunca entraría en el bucle interno, con lo que la eficiencia en este caso seria O(n). Calculando la eficiencia empírica y representándola gráficamente observamos que los tiempos se ajustan una recta, ajustándose a la previsión.



Ejercicio 6

Al optimizar el código mediante la orden g++ -03 ordenacion.cpp ordenación_optimizado observamos una gran mejora en la eficiencia empírica



Ejercicio 7

El algoritmo de multiplicación matricial:

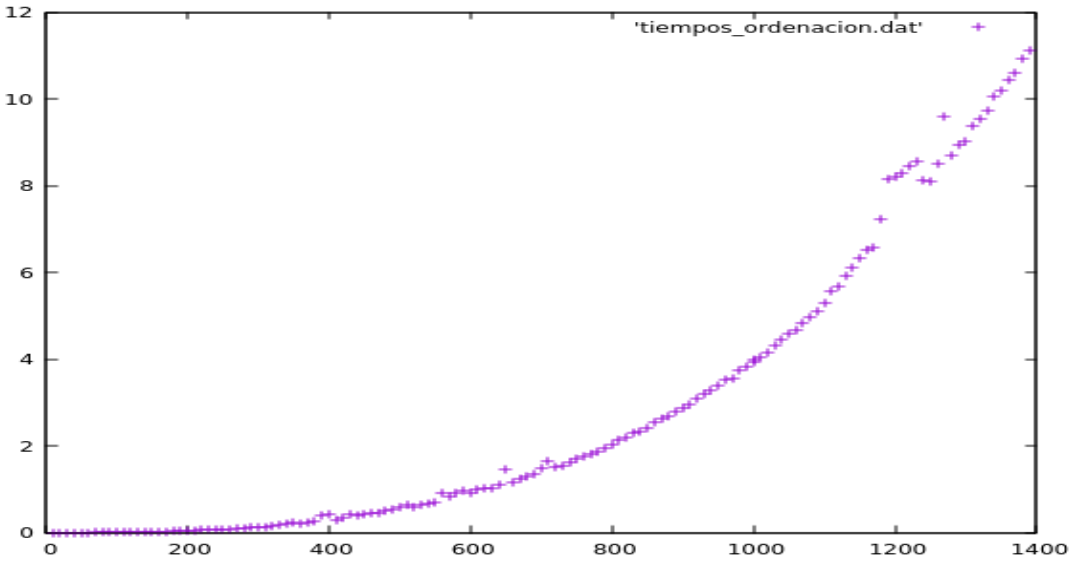
```
for(int i=0; i<tam;)

    for(int j=0; j<tam; ++j)

        for(int z=0; z<tam; ++z)

            m3[i+j] += m1[i+z] * m2[z+j];
```

Podemos observar que la línea que mas veces se ejecuta es la ultima, que esta dentro de tres bucles que itera n veces, por tanto esta orden se ejecutara n^3 veces. Lo que se traduce en una eficiencia teórica de O(n^3). Calculando su eficiencia empírica y representándola gráficamente obtenemos la siguiente tabla.



Si ajustamos nuestra gráfica a una ecuación f(x) del orden cúbica observaremos que se ajusta perfectamente a la previsión.