Isidora Reina Molina
Pedro Luis Hurtado González

PRÁCTICA DE ABSTRACCIÓN

Estudio de Eficiencia

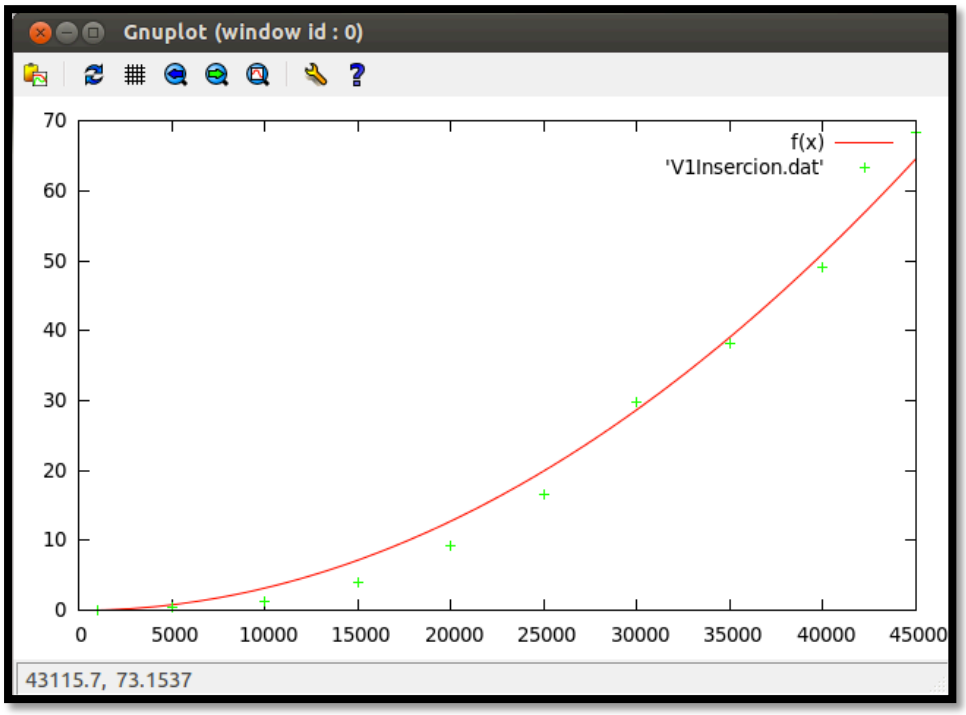
❖ Inserción de Meteoritos:

Versión 1:

La primera versión del diccionario inserta meteoritos de forma desordenada *(push_back).* Para poder representar la gráfica hemos hecho una tabla con la inserción de los meteoritos:

1000 meteoritos 0.03 segundos
5000 meteoritos 0.379 segundos
10000 meteoritos 1.294 segundos
15000 meteoritos 4.021 segundos
20000 meteoritos 9.32 segundos
25000 meteoritos 16.477 segundos
30000 meteoritos 29.875 segundos
35000 meteoritos 38.197 segundos
40000 meteoritos 49.01 segundos
45000 meteoritos 68.335 segundos

Tras el estudio de la gráfica hemos concluido que es de orden cuadrático O(n^2):

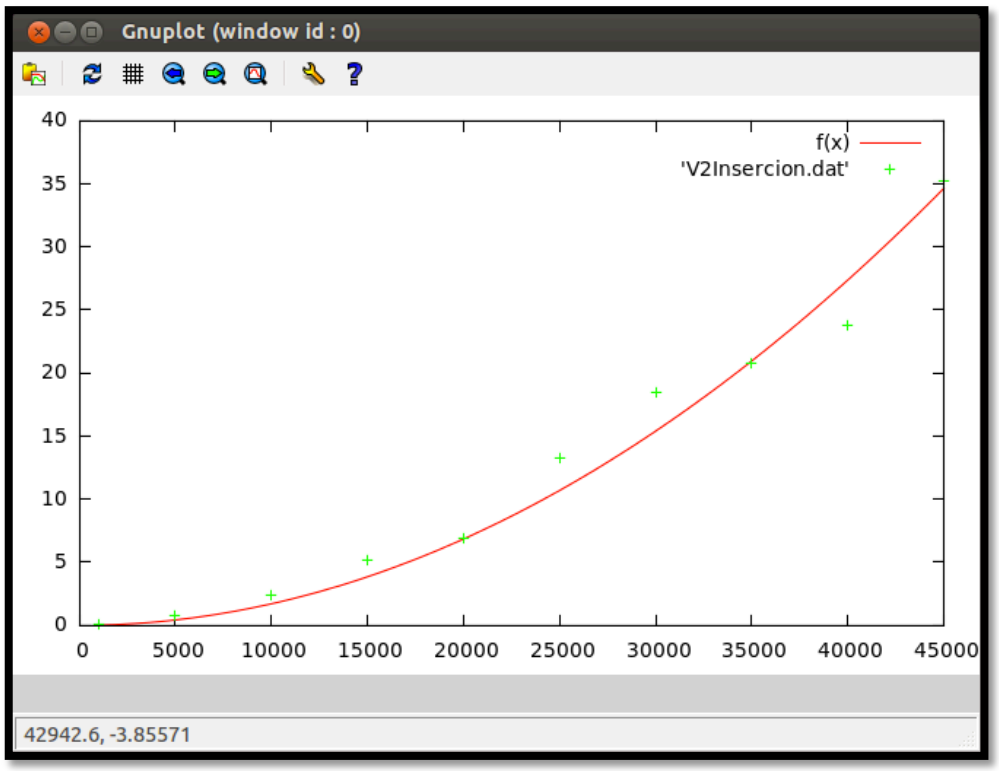


Versión 2:

La segunda versión del diccionario inserta meteoritos de forma ordenada *(stl::insert).* Para poder representar la gráfica hemos hecho una tabla con la inserción de los meteoritos:

1000 meteoritos 0.055 segundos
5000 meteoritos 0.814 segundos
10000 meteoritos 2.361 segundos
15000 meteoritos 5.19 segundos
20000 meteoritos 6.916 segundos
25000 meteoritos 13.219 segundos
30000 meteoritos 18.429 segundos
35000 meteoritos 20.805 segundos
40000 meteoritos 23.764 segundos
45000 meteoritos 35.240 segundos

Tras el estudio de la gráfica hemos concluido que es de orden cuadrático $O(n^2)$:



❖ Búsqueda de Meteoritos:

Para la búsqueda de los meteoritos hemos creado un bucle en el método load() del archivo principal.cpp que recorre todos los identificadores de los meteoritos buscándolos en el diccionario.
Para el cálculo del tiempo de ejecución de éste bucle hemos usado las funciones de la plantilla *<time.h> (start() y end()).*
Hemos obtenido los siguientes resultados:

-Versión desordenada (V1): 65,74 segundos.
-Versión ordenada (V2): 0,05 segundos.

❖ Conclusión:

En la inserción de meteoritos concluimos que en la versión ordenada, las primeras inserciones son más lentas con respecto a la versión desordenada.
Sin embargo, aproximadamente a partir de la mitad de los datos, es más rápida la inserción en la versión ordenada.

En la búsqueda es evidente que la implementación de la versión ordenada es más eficiente que la desordenada.