Informe Trabajo Práctico Especial

Programación 3 - TUDAI 2017



Alumnos: Pedro di Fonzo.

Lucas Pagadizabal.

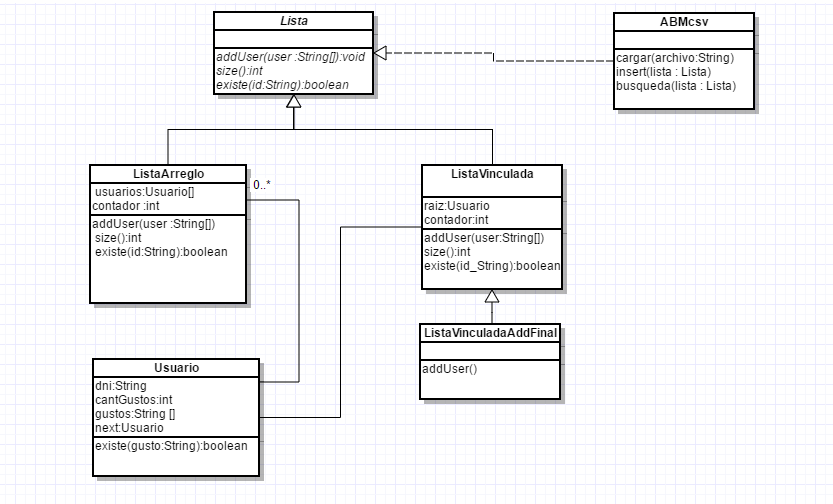
**Decisiones de diseño**

Vimos que en los 3 ejercicios que pedían, se llevaban a cabo 3 funcionalidades iguales (cargar, insertar, buscar), entonces creamos una clase “ABMcsv” donde en esta desarrollamos los 3 métodos necesarios.

Cada uno de estos métodos podría llegar a recibir listas diferentes (ListaArreglo, ListaVinculada), para poder recibir cualquiera de estas dos, pero llevar a cabo el mismo comportamiento, decidimos crear un padre “Lista”. Esta clase es abstracta, donde solo especifica métodos abstractos también, para que sus hijos, “ListaArreglo” y “ListaViculada”, tengan que implementarlos.

En el ejercicio c, decidimos crear una nueva clase “ListaVinculadaAddFinal” donde hereda de “ListaVinculada”.En esta nueva clase redefinimos el método addUser, ya que cambia el comportamiento, que sería ir agregando cada usuario al final. Llevamos a cabo esta decisión para que el “ABMcsv” no se tenga que tocar, y pueda recibir cualquier tipo de “Lista”.

En la clase usuario implementamos el método “existe para verificar que no ingresen gustos iguales en un mismo usuario”



(breve descripción del diseño en diagrama)

Tiempo total y promedio

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ListaArreglo | inserción | búsqueda | Promedio inserción | Promedio Búsqueda |
| 500.000 | 32855771 | 198506063613 | 3285,5771 | 19850606,3613 |
| 1000000 | 31658765 | 397563873053 | 3165,8765 | 39756387,3053 |
| 3000000 | 40049676 | 1173630617382 | 4004,9676 | 11736306,17382 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ListaVinculada | inserción | búsqueda | Promedio inserción | Promedio Búsqueda |
| 500.000 | 33476405 | 206540750817 | 3347,6405 | 20654075,0817 |
| 1000000 | 34329898 | 430318632693 | 3432,9898 | 43031863,2693 |
| 3000000 | 51119740 | 1262784038184 | 5111,9740 | 12627840,38184 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ListaVinculada  (AddFinal) | inserción | búsqueda | Promedio inserción | Promedio Búsqueda |
| 500.000 | 30400176 | 226084829324 | 3040,0176 | 22608482.9324 |
| 1000000 | 36605557 | 437519667532 | 3660,5557 | 43751966.7532 |
| 3000000 | 55816354 | 1278972117643 | 5581,6354 | 12789721.17643 |

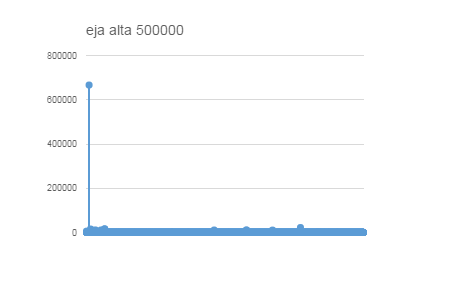
El peor tiempo

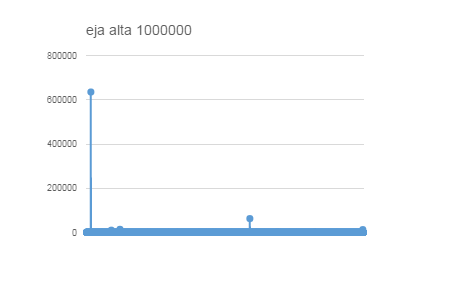
Los peores tiempos se dieron en la estructura de lista vinculada con inserciones al final.

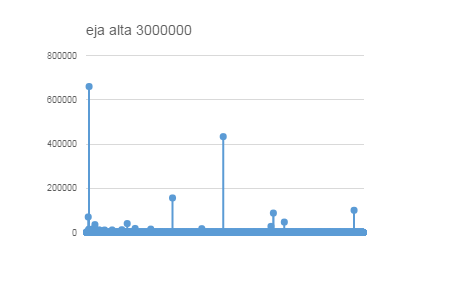
Con un tiempo de 55816354 nanosegundos para la inserción y de 1278972117643 nanosegundos para la búsqueda.

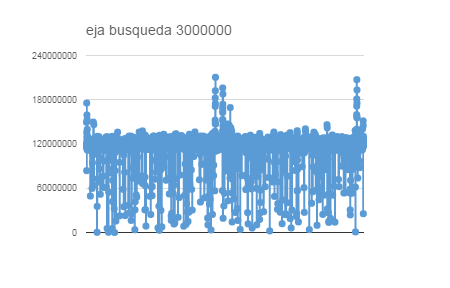
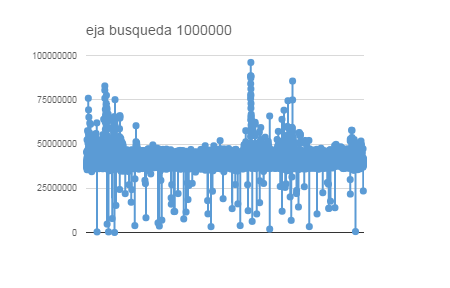
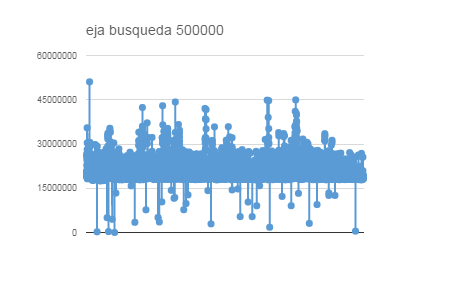
Diagramas

Lista con Arreglo

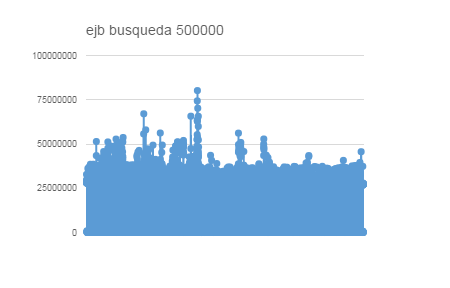
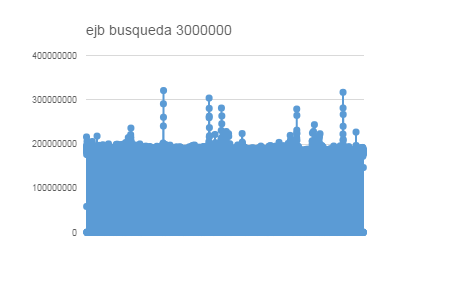
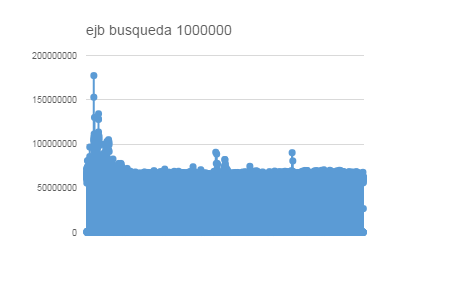
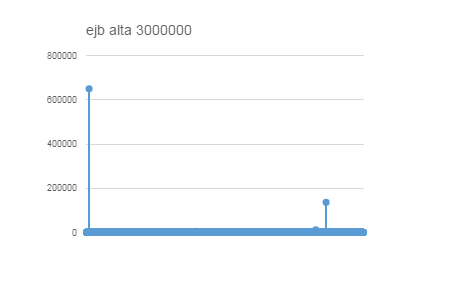
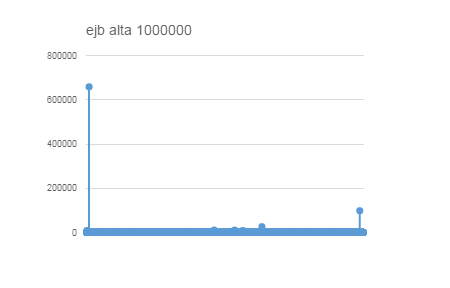
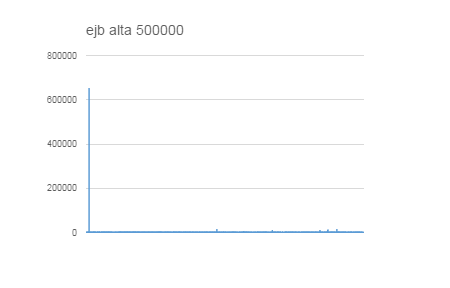




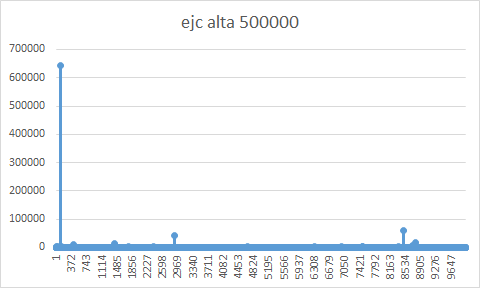


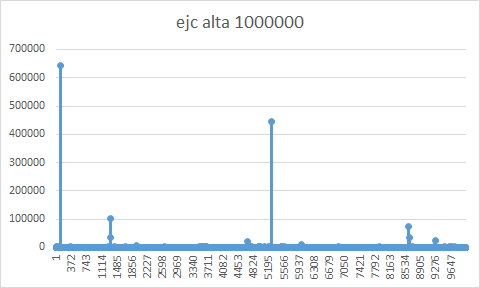


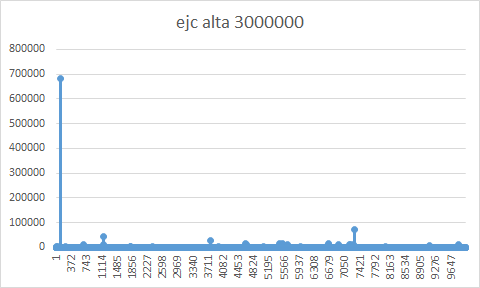
Lista Vinculada

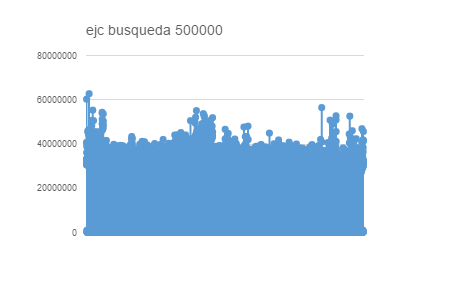


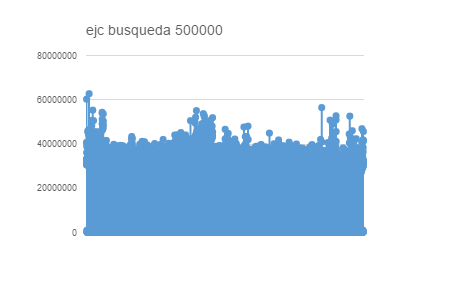
Lista Vinculada (addFinal)

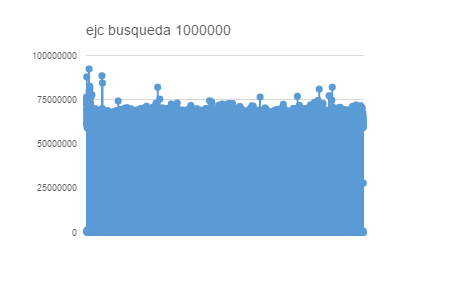












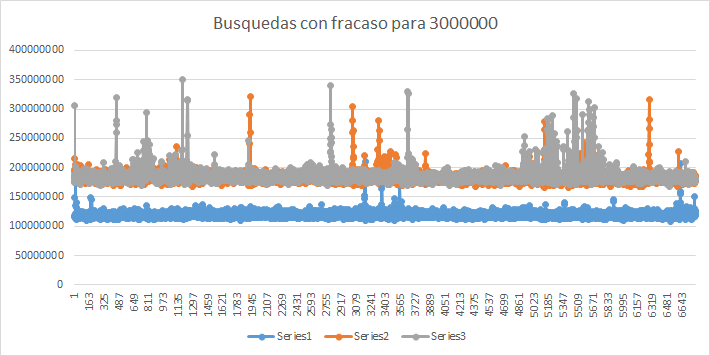
En los gráficos de alta de las 3 estructuras notamos que al principio tarda porque pide la estructura, luego la misma queda guardada en la caché de la computadora, entonces acorta el tiempo de acceso.

En la estructura de arreglo y la estructura de lista vinculada con inserción al final, el tiempo de alta de los 10.000 usuarios es lineal, O(1). En cambio en la lista vinculada con inserción al final el tiempo de inserción va aumentando a medida que aumenta la cantidad de elementos ya que debe recorrerlos todos para realizar la misma. Tiene un comportamiento O(n).

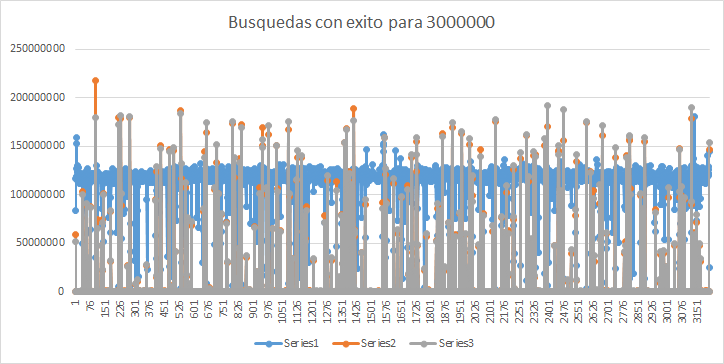
Al obtener gráficos de búsquedas muy parecidos referidos al tiempo, decidimos crear dos gráficos de búsquedas con éxito y otro sin éxito

El siguiente gráfico muestra los tiempos de los fracasos al realizar una búsqueda. Cuando la misma fracasa tiene que recorrer toda la estructura buscando el dato.

Notamos que en el arreglo, representado azul los tiempos son más bajos ya que es menor el costo para ir avanzando posición a posición en comparación a una lista de punteros como en las listas vinculadas representadas en color naranja(inserción al principio) y gris(inserción al final).



En el gráfico de búsquedas con éxito se puede observar que la diferencia de tiempos para las distintas estructuras se basa en el tiempo de acceso a la misma y no tanto en su recorrido. En este caso hay mayoría de tiempos bajos para las estructuras representadas con listas de punteros ya que no es necesario realizar recorridos completos.



Conclusión

Como conclusión podemos observar que antes de desarrollar estructuras para guardar datos, hay que tener en cuenta sobre qué datos vamos a trabajar, qué requerimientos debe llevar a cabo la estructura, del hardware dónde se va a ejecutar y otras cuestiones de diseño ya que dependiendo de lo que se necesite, se podría llegar a optar por las diferentes estructuras implementadas en el trabajo desarrollado anteriormente.

Al trabajar con datos muy grandes (500000, 1000000, 3000000) nos dimos cuentas de lo costosos computacionalmente que son algunos algoritmos. Para llevar a cabo los enunciados del práctico tuvimos que ejecutar los diferentes algoritmos que implementamos en una computadora con un hardware potente, ya que nuestras propias computadoras con datos de 1000000 ya no soportaban la cantidad de datos.

La computadora que utilizamos nos dio gráficos muy parecidos, entonces nos costó desarrollar conclusiones sobre los mismos. A raíz de este problema desarrollamos las ideas con los pocos datos que pudimos obtener de los gráficos y lo que teóricamente debería pasar.