Sesión 5 de Estructura de Datos: **Curso 2012/2013**

El objetivo de esta sesión es conocer los árboles de búsqueda binaria, y sus implementaciones en Python. La fecha límite de entrega de esta sesión es el día 12-05-2013. Documentar el código.

Ejercicio

En esta sesión vamos a continuar con la aplicación de LastFM. En este caso queremos representar la información de todos los usuarios mediante árboles de búsqueda binaria (ABB). La idea de la práctica es comparar los árboles de búsqueda binaria con las listas enlazadas.

Recordad que tenemos toda la información de los usuarios en un fichero de datos llamado ***LastFM_small.dat***. De este fichero vamos a extraer información de los usuarios.

El objetivo principal es gestionar la colección de usuarios y para ello utilizaremos un árbol de búsqueda binaria. Esta gestión consiste en hacer operaciones de inserción, búsqueda y borrado. La eficiencia de estas operaciones se estimará calculando el tiempo de ejecución de estas operaciones.

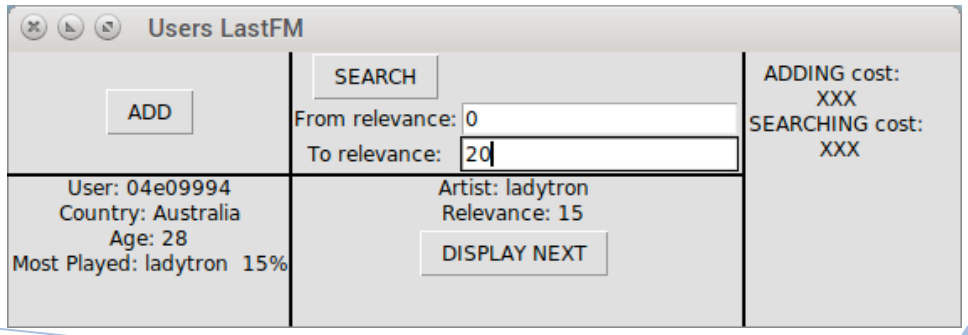
Para implementar la aplicación seguir las instrucciones:

Paso 1. Crear un fichero **ABB_nombre_apellidos.py** donde se definirán las operaciones del árbol de búsqueda binaria. Las operaciones básicas son: crear un ABB, insertar un nuevo elemento, imprimir el ABB, borrar elementos de un ABB.

Como clave se usará el valor de relevancia. Tened en cuenta que cada nodo del árbol puede incluir una lista de usuarios con el mismo valor de relevancia; de este modo, al añadir un nuevo usuario debemos mirar si el nodo con la relevancia correspondiente existe. El funcionamiento deberá ser el siguiente: en el caso de no existir debemos crear el nodo y añadir el usuario correspondiente en la lista de este nodo, y en caso de que el nodo ya exista únicamente debemos añadir el usuario a la lista del nodo del ABB que le corresponda según su relevancia.

Paso 2. Crear un fichero **nombre_apellido_S5.py** y leer el fichero LastFM_small.dat y guardarlo en el árbol de búsqueda binaria.

Paso 3. Crear un fichero **Nombre_Apellidos_ABB_Interface.py** para implementar la interfície y funcionalidades solicitadas. Partiremos de la interfície de la práctica anterior, donde una posibilidad sería:



La interfície nos muestra “From relevance” y “To relevance” que sirven para definir el criterio de búsqueda de los usuarios. Cada vez que se aprieta el botón de “’NEXT”, se nos muestra el siguiente en la lista con el criterio que se ha definido de “From”’ y “To”’. Si se alcanza el final del “To”, el “’NEXT” vuelve a la posición inicial del “From”. En caso que no se haya especificado un criterio, el “From” es la raíz del árbol y el “To” es el final del árbol de búsqueda binaria. Además, la pantalla nos muestra el tiempo computacional consumido para inserir los usuarios (ADDING cost) y buscar (SEARCHING cost) usando la lista enlazada.

En esta ocasión, se añade también un botón de “REMOVE”. Se borraran todos los nodos que estén entre el valor de “From relevance” y “To relevance”. Y en la parte de estadísticas, además del “adding cost” i el “searching cost”, se pondrá también el “removing cost”.

Paso 4. Implementar la interfície de la aplicación en la clase **Nombre_Apellidos_ABB_Interface.py** con las siguientes funcionalidades:

a. ADD: añade 5000 usuarios del fichero en el árbol de búsqueda. Cada vez que se apriete el ADD se añaden los 5000 siguientes usuarios hasta que se acabe el fichero
b. SEARCH: dados dos valores “From relevance” y “To relevance”, recupere todos los usuarios que tienen una relevancia dentro de este intervalo
c. DISPLAY NEXT: muestra secuencialmente el id del usuario, su país, edad, genero, artista y relevancia
d. REMOVE: dados dos valores “From relevance” y “To relevance”, elimine del árbol todos los usuarios que tienen una relevancia dentro de este intervalo

Paso 5. Visualizar el tiempo de ejecución de la inserción (ADD), la búsqueda (SEARCH) en la interfície y el borrado (REMOVE)

Paso 6. Modificar la interfície de la práctica anterior con la cola de prioridad para que inserte el botón ADD 5000 usuarios.

Paso 7. Comparar el tiempo de ejecución promedio de insertar (ADD) y buscar (SEARCH) en el árbol de búsqueda binaria y usando la cola de prioridad de la sesión anterior (como mínimo 10 veces). Para esto podéis utilizar el fichero ***LastFM_big.dat***

NOTAS:

- ***La entrega de la sesión ha de ser únicamente por el Campus Virtual dentro del límite de tiempo predefinido.***
- ***Entregad un fichero comprimido (“NombreApellidos_S5.zip”) que contenga todo el código fuente (las dos aplicaciones).***
- ***Recordad que es obligatorio e*****structurar bien la información, documentarla y argumentarla*****.***