Sesión 6 de Estructura de Datos: **Curso 2011/2012**

El objetivo de esta sesión es evaluar la eficiencia de trabajar con heaps y estructuras de hash, comparándolas con los árboles de búsqueda binaria. Para este objetivo haremos una comparativa respecto al tiempo de ejecución y analizaremos sus ventajas y desventajas. La fecha límite de entrega de esta sesión es el día 20-05-2012. Documentar el código.

En esta sesión vamos a continuar con la aplicación del videoclub. En este caso queremos representar las películas mediante heaps y estructuras hash. La idea de la práctica es comparar un heap y una estructura de hash con un árbol de búsqueda binaria.

Las tareas que deberéis efectuar en esta práctica es el análisis del tiempo necesario para la creación y las búsquedas en las diversas estructuras de datos:

- Árbol de búsqueda binaria "original" (ABB Original)
- Árbol de heap
- Hash

Todas estas funciones deben ser implementadas en la interficie tkInter. En la siguiente figura podemos ver un ejemplo de la aplicación que debéis desarrollar.

En esta interfície tenemos el botón ***ADD MOVIE*** para añadir las películas de nuestro videoclub. Además, tenemos un campo para introducir el rango del ***rating*** de las películas que queremos buscar. Una vez introducidos, se pide buscar la película en las tres estructuras de datos que queremos comparar (ABBOriginal, Heap y Hash). El resultado de la búsqueda serán las películas, el coste temporal para buscar en cada estructura de datos y la profundidad del ABBoriginal, y del heap.

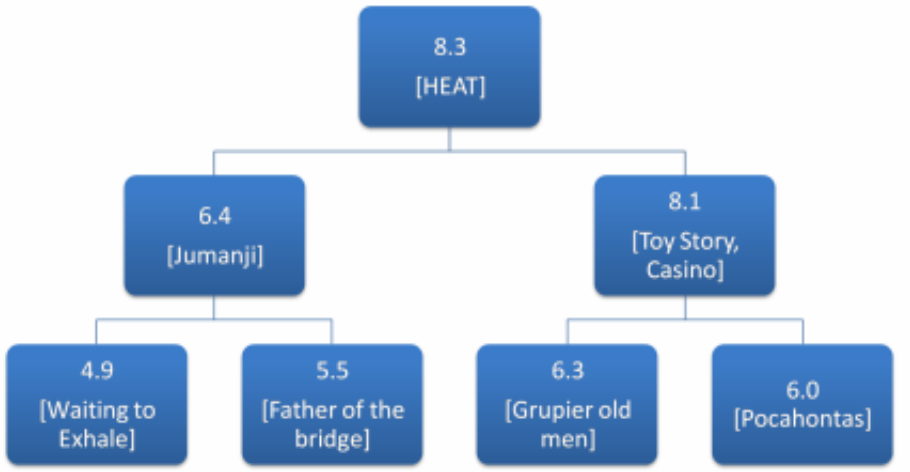
Instrucciones:

Paso 1. Crear el heap teniendo en cuenta que el valor de rating del nodo padre siempre será mayor que el valor de rating de sus hijos. Además, utilizando el código de la practica 5, se ha de crear también el árbol de búsqueda binaria original (ABBoriginal). Todas las estructuras de datos deben ser creadas al apretar el botón ADD MOVIE.

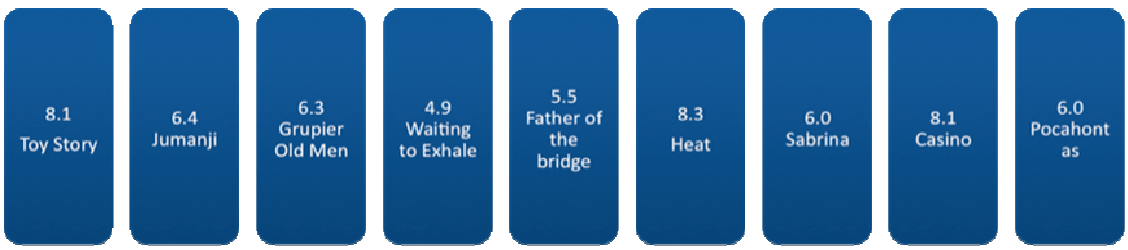
Como hicimos en la práctica anterior cuando usabamos el ABB Original el heap debe ser creado a partir de la lista de películas en el mismo orden que aparece en el archivo *.dat*. Tened en cuenta que cada nodo del árbol de búsqueda binaria incluye una lista de películas con el mismo valor de rating. De este modo, al añadir una nueva película debemos mirar si el nodo con el rating correspondiente existe.

El funcionamiento debera ser el siguiente: en el caso de existir debemos crear el nodo y anadir la pelicula correspondiente en la lista de este nodo, y en caso de que el nodo ya exista unicamente debemos anadir la pelicula en la lista del nodo del heap o ABB que le corresponda segun su rating.

En la siguiente figura tenéis un ejemplo de cómo quedaría el heap una vez insertados los primeros 9 datos del archivo *.dat*.



Notad que para la creación de éste árbol los datos han sido introducidos con el siguiente orden:



Paso 2. Analizar y comparar el tiempo de inserción y de búsqueda de un conjunto de películas utilizando las distintas estructuras de datos implementadas: Heap,

ABBoriginal y Hash. El funcionamiento es el siguiente: el usuario debe introducir el rango del *rating*; al apretar el botón *Search* el programa deberá calcular y enseñar en la interficie el tiempo de ejecución para la búsqueda.

Paso 3. Implementar una tabla hash usando 200 celdas de memorias. Esta estructura de datos también debe ser creada al apretar el botón ADD MOVIE. Para calcular la función de hash utilizaremos el **rating**. Aplicar diferentes funciones hash diseñada por vosotros. Calcular el tiempo de búsqueda de los casos en la tabla.

Paso 4. Estructurar bien la información, documentarla y argumentarla.

NOTA: La entrega del codigo python (un fichero) de la sesión ha de ser únicamente por el Campus Virtual dentro del límite de tiempo predefinido.