Estructuras de Datos.
Facultad de Ciencias, UNAM 2015-2.
Práctica 08: Comprimiendo un texto mediante códigos de Huffman.

Armando Ballinas Nangüelú.
armballinas@gmail.com

Fecha de entrega jueves 07 de mayo de 2015.

1. Introducción.

Hay ocasiones en las que se comprimir un texto para ahorrar espacio. Una manera de hacerlo es asignarle a los caracteres con mayor frecuencia una representación más corta y a los caracteres con menor frecuencia una representación más larga. Los códigos de Huffman permiten hacer lo anterior y para ello utilizan un árbol binario.

En esta práctica se implementarán los códigos de Huffman implementando a su vez el TDA Árbol Binario. También se utilizarán estructuras de datos auxiliares de los paquetes `Util` y `Collections` de Java.

2. TDA Árbol Binario.

Un árbol binario es un árbol en donde cada nodo puede tener a lo más dos hijos. Los árboles son estructuras de datos no lineales muy utilizadas en la práctica pues permiten realizar operaciones sobre ellos eficientemente y son bastante flexibles.

Las operaciones del árbol binario están descritas en la interfaz `ArbolBinario.java`

3. Códigos de Huffman.

Dado un alfabeto de entrada $\Sigma = \{s_1, \ldots, s_n\}$ construir el código de Huffman para cada caracter significa definir una función f que asocie un caracter de Σ a una cadena binaria. Es decir, $f : \Sigma \to \{0, 1\}^n$. El procedimiento para la construcción de esta función es el siguiente:

1. Se ordenan los símbolos del alfabeto por sus apariciones en el texto a comprimir de manera ascendente. Es decir, primero irán los caracteres menos frecuentes.

2. Se construye un árbol con un solo nodo por cada símbolo, en la información del nodo debe ir el caracter y su frecuencia en el texto y se meten en el orden dado por sus frecuencias en una lista.

3. Mientras la lista tenga más de un árbol:

   a) Se toman y eliminan los primeros dos árboles de la lista y se crea otro árbol cuyos hijos sean estos dos árboles. El caracter asociado a la raíz de este árbol no tiene importancia pero su frecuencia es la suma de las frecuencias de sus hijos.

   b) Este nuevo árbol se introduce a la lista de manera ordenada, es decir, se debe preservar el orden creciente de frecuencias entre los árboles de la lista.

4. Cuando la lista solo tenga un elemento este será la raíz del árbol de Huffman.

En el árbol construido en el proceso anterior se encuentran todos los símbolos del texto y cada nodo que los representa es una hoja del árbol obtenido. Con esto se pueden generar los códigos de Huffman de cada símbolo de la siguiente manera: Se recorre el árbol desde la raíz. Los códigos del hijo izquierdo tienen un cero al inicio y los del hijo derecho un uno. Recursivamente se recorren los hijos izquierdo y derecho. Cuando se llegue a una hoja se le asigna el código resultado. Este será el código de huffman asociado a este símbolo.

4. Estructuras de Datos y clases auxiliares.

Para la obtención de los símbolos y sus frecuencias a partir de un texto se debe usar un diccionario. Un diccionario es una estructura de datos que almacena parejas de datos de la forma: $(llave : valor)$. Es decir, un diccionario tiene una colección de llaves y cada llave tiene asociado un valor. Los valores se acceden por medio de la llave asociada a ellos. Java provee una implementación de diccionarios mediante la clase `HashMap` que utiliza una función *hash* para asociar la llave a un elemento.

En esta práctica se usará esta clase para guardar los símbolos del texto y su frecuencia. La llave será el símbolo y el valor será la frecuencia actual del símbolo. Para obtener la frecuencia total de cada símbolo se debe entonces leer todo el texto de entrada y por cada caracter añadirle uno al valor actual de la llave asociada a ese caracter en el diccionario.

Para crear la lista necesitamos obtener todas las parejas de llaves y valores del diccionario, para esto Java provee el método `HashMap.entrySet()` que devuelve un objeto `Set`. Este objeto provee un iterador para poder obtener los elementos uno a uno. Cada elemento del conjunto es un objeto `Map.Entry<Character,Integer>` y sobre estos objetos se pueden usar los métodos `getKey()` y `getValue()` para obtener la llave y el valor asociado.

Se puede notar que se requieren varias clases auxiliares para obtener los caracteres y su frecuencia a partir del texto. Por lo anterior se debe tener mucho cuidado al hacerlo. En el laboratorio se hará un ejercicio mostrando el funcionamiento de las clases anteriores.

5. Clase de prueba Main.

Se debe crear una clase de prueba en donde se lea un texto por medio de la entrada estándar. No se debe mostrar ningún mensaje al usuario pidiéndole nada, solo se deben leer caracteres hasta encontrar el fin de archivo o el caracter nulo.

Una vez leído el texto se deben calcular las frecuencias de todos los caracteres que aparecen allí. Con la lista de frecuencias por caracter se debe crear el árbol de Huffman para estos caracteres y con él se debe obtener el código de Huffman asociado a cada caracter.

Ya que se tenga el código de Huffman de cada caracter en el texto, se debe imprimir en pantalla una lista. Cada elemento de la lista será un caracter del texto seguido del código de Huffman asociado a él.

Para las pruebas de su programa pueden pasarle un archivo de texto a la clase `Main` mediante el comando de redireccionamiento `<`. En los archivos adicionales se incluyen dos archivos de texto para probar su programa, uno es un texto en español y el otro en inglés, su programa debe funcionar con ambos.

6. Ejercicios.

1. (2 pts.) Implementar el TDA árbol binario implementando la interfaz `ArbolBinario.java`.
2. (2 pts.) Leer los caracteres de un texto de entrada y obtener sus frecuencias usando un objeto de la clase `HashMap`.
3. (5 pts.) Obtener el código de Huffman asociado a cada caracter en el texto de entrada.
4. (1 pt.) Realizar la clase `Main.java`

7. Puntos extra.

Hasta dos puntos extra para los que implementen la insersión del nodo en la lista utilizando búsqueda binaria.