




	<p>Universidad de Costa Rica</p> <p>Escuela de Ciencias de la Computación e Informática</p> <p>Semestre II - 2021</p> <p>Curso CI-0112 - Programación I</p> <p>Profesor: Edgar Casasola Murillo, Francisco Arroyo, Valeria Zamora</p>
---	--

	<p>Universidad de Costa Rica</p> <p>Escuela de Ciencias de la Computación e Informática</p> <p>Semestre II - 2021</p> <p>Curso CI-0112 - Programación I</p> <p>Profesor: Edgar Casasola Murillo, Francisco Arroyo, Valeria Zamora</p>
---	--

	<p>Universidad de Costa Rica Escuela de Ciencias de la Computación e Informática Semestre II - 2021 Curso CI-0112 - Programación I Profesor: Edgar Casasola Murillo, Francisco Arroyo, Valeria Zamora</p>
---	--

	<p>Universidad de Costa Rica Escuela de Ciencias de la Computación e Informática Semestre II - 2021 Curso CI-0112 - Programación I Profesor: Edgar Casasola Murillo, Francisco Arroyo, Valeria Zamora</p>
---	--

Tarea programada III

Forma de entrega: Grupos 3 personas	Fecha de entrega: 5 de diciembre, 11:55 p.m.
--	---

Forma de entrega: Grupos 3 personas	Fecha de entrega: 5 de diciembre, 11:55 p.m.
--	---

Usted debe programar un compresor/decompresor para archivos de texto utilizando el algoritmo de Huffman.

Debido a que en la actualidad hay muchas implementaciones del algoritmo disponibles en Internet, ustedes tendrán que programar su compresor y descompresor cumpliendo **“en forma estricta y obligatoria”** con las especificaciones siguientes, si no cumple con estas especificaciones su solución no concederá puntos:

1. Para leer el archivo deberán usar la clase Scanner con el método nextByte() para leerlo byte por byte.
2. Para escribir los archivos deberán usar FileOutputStream y guardar usando el método write(byte v[]) que guarda los bytes almacenados en un vector de bytes a disco. La idea es no guardar byte por byte sino varios a la vez (512 bytes).
3. El archivo comprimido será una secuencia de bits con el siguiente formato:

Encabezado	Arbol	Datos
------------	-------	-------

El **Encabezado**: corresponde a los primeros cuatro bytes del archivo. Estos cuatro bytes almacenan en el primer byte un “0x7f” para indicar que se trata de un archivo especial, en el segundo un “H” o “0x48” , en el tercero un “U” o “0x55” y en el cuarto “0xF0bbb” (para formar las iniciales “HUF” casi) donde “bbb” representa un valor binario entre “000” y “111” que indica la cantidad de bits que fue necesario agregar al final del archivo para completar **el último byte de todo el archivo**. Si estos valores son 000 eso quiere decir que todos los valores del último byte quedaron en uso. La idea es usar solo esos bits del último byte que se lee desde el archivo para ignorar los bits restantes al final. Por ejemplo: si del último byte solo se usan los 5 primeros bits y ahí finaliza el archivo, y los otros 3 bits quedaron sin uso, entonces se almacena un valor 3 en binario en el campo destinado al Desplazamiento, este caso se muestra en la Figura 1.

[illegible]

Figura 1. Ejemplo de Desplazamiento con valor 101 ($v="11110011"$, donde los primeros cuatro unos representan el número "0xF" en hexadecimal y "0011" el número 3 de los bits que quedan sin uso). Note que del último byte solo se usan los cinco primeros bits y los últimos tres bits quedan sin uso convirtiéndose en el único espacio desperdiciado en el archivo comprimido. Se marcaron con X los bits que quedan sin uso.

El Arbol: Es una representación en bits generado en preorden para reconstruir el árbol en forma recursiva. Si se tiene un bit en 1 eso indica que a continuación viene la lectura de los hijos del árbol, y en caso de que el bit leído sea un 0 eso indica que los siguientes 8 bits corresponden al byte que se encuentra codificado en ese nodo. En otras palabras 1 indica que hay que leer un nodo con dos hijos, y un 0 que hay que leer una hoja. Por ejemplo: En la Figura 2 se muestra el árbol que se reconstruye a partir de la cadena indicada. Para visualizar mejor la composición de la cadena se colocó un espacio entre los bits, de esa forma se tiene algo como sigue: 1 0 00000000 1 0 11111111 0 10101010 0 11001100. Al leer un 1 se sabe que hay que crear un nodo con dos hijos, primero se leerá el hijo izquierdo y luego el derecho. Al leer el hijo izquierdo se lee un 0 que indica que se debe crear un nodo hoja con el código que se construye con los 8 bits siguientes, en este caso 00000000. Eso concluye la lectura del hijo izquierdo así que hay que leer el hijo derecho, un 1 indica que hay que leer un sub-árbol con hijos y se procede a leer su hijo izquierdo de ese sub-árbol que al tener un 0 indica que es un código en este caso 11111111. Al leer la rama derecha de ese sub-árbol se tiene que sigue un 1 lo que indica que hay que leer un nuevo sub-árbol con dos hijos nuevamente. Se lee a continuación un 0 que indica que el hijo izquierdo de ese sub-árbol es un código y tiene el valor 10101010 y luego el 0 indica que a la derecha hay una hoja con el código 11001100. En ese punto ya se leyó el hijo derecho completando el sub-árbol que completa el sub-árbol de la rama derecha del árbol raíz. Eso finaliza la lectura del árbol completo.

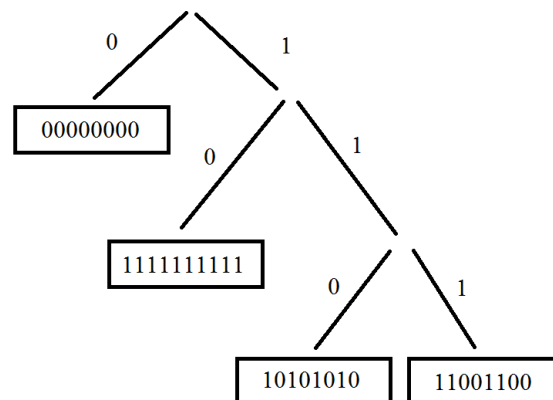


Figura 2. Árbol reconstruido desde disco al leer en preorden la cadena de bits 10000000001011111111010101010011001100.

Datos: Son los bits que se deben identificar desde la raíz del árbol para llegar a los códigos que se producen como producto de la decodificación del archivo comprimido. Cada vez que se llega a una hoja, se escribe en la salida el código y se regresa a la raíz del árbol para bajar por la ruta indicada en los datos hasta llegar a un nuevo código. Ese proceso se lleva a cabo hasta llegar al último byte del que se procesarán solamente los bits indicados en el campo de **desplazamiento**. Esa cantidad de bits deben terminar exactamente en una ruta que lleva

a un código, de no ser así eso indicaría que existió algún error de codificación o decodificación en el archivo.

El grupo debe programar:

a) Una versión que mediante dos comandos en consola permita comprimir y descomprimir archivos.

java Huffman -c nombreArchivo.ext comprime el archivo llamado nombreArchivo.ext y lo deja en un archivo llamado nombreArchivo.ext.huf

java Huffman -d nombreArchivo.ext.huf descomprime el archivo en un archivo llamado nombreArchivo.ext

b) Una versión que mediante una interfaz gráfica permita leer un archivo usando un FileChooser y mediante botones permita comprimir o descomprimir el archivo seleccionado.

Dado que el formato está estrictamente especificado, todo compresor de cualquier grupo de estudiantes deberá poder descomprimir archivos creados con los programas de cualquier otro grupo.

Pueden usar las clases BitFileReader y BitFileWriter proporcionadas con el enunciado.