INSTITUTO TECNOLÓGICO DE COSTA RICA

SEGUNDO PROYECTO PROGRAMADO

Ariel Herrera
Saúl Zamora

profesor
M. Sc. Saúl Calderón Ramírez

September 29, 2016

I. Introducción

En la actualidad, los sistemas de almacenamiento y comunicación digitales requieren de métodos optimizados para el uso de recursos (energéticos, temporales, de estacio, etc). Para satisfacer tal necesidad de manera efectiva, muchas disciplinas han formulado mútiples algoritmis para comprimir y descomprimir la información. La compresión de datos consiste en aplicar algún método que permita reducir el tamaño original de la información. Los algoritmos de compresión sin pérdida son capaces de aplicar una serie de pasos para construir la información comprimida, para luego, cuando la información original necesite ser accesada, se descomprime y recupera la información original complemtamente idéntica. Los algoritmos de compresión con pérdida en cambio, al implementar la descompresión de la información, no lo gran recuperar el 100% de los datos originales. El algoritmo de Huffman implementado en este proyecto fue propuesto por David A. Huffman en 1952 enfocado en la compresión sin pérdida de datos.

II. Análisis del problema

La técnica de Huffman trabaja al crear un árbol binario de nodos, los cuales pueden ser hojas o nodos internos. Al principio, todos empiezan como hojas, las cuales contienen un símbolo, el peso (*frecuencia*) es opcional, y un enlace al nodo padre, lo cual facilita leer el código comenzando de las hojas. Los nodos internos contienen el peso del símbolo, dos enlaces a nodos hijos y un enlace opcional a un nodo padre.

Como una convención, el bit *0* representa el siguiente hijo izquierdo y el bit *1* el siguiente hijo derecho. Un árbol terminado puede crecer hasta tener (n) hojas y $(n-1)$ nodos internos. Un árbol de Huffman que omite los símbolos que no se usan, produce el código con el largo óptimo.

El proceso inicia con las hojas conteniendolos símbolos a representar, luego un nuevo nodo es creado con los nodos con menor probalilidad como hijos, tal que la probabilidad de dicho nodo es igual a la suma de las probabilidades de sus hijos. Con los nodos anteriores mezclados en uno (ya no son considerados), y considerando al nuevo nodo, el proceso se repite hasta que solo quede un nodo: el árbol de Huffman.

III. Diseño de la solución

A. El algoritmo de Huffman

El algoritmo de Huffman fue diseñado para comprimir señales digitales. Dichas señales están compuestas por un conjunto finito de símbolos, donde cada uno está definido por una cadena de bits.

El algoritmo de Huffman busca crear para cada símbolo, una cadena de bits o código de Huffman, que al reemplazarse por el *token* correspondiente, reduzca el tamaño total del texto. Para construir el diccionario, el algoritmo analiza todo el texto, para construir un diccionario de frecuencias de aparición, el cual defina una entrada por *token*, donde dicho *token* es la llave y el valor de entrada es definido por la cantidad de veces que el *token* aparece en el texto.

El algoritmo busca generar el código más corto para el símbolo más recurrente. Para ello es necesario asegurarse que la codificación de todos los tokens no sea ambigua, es decir, que sea posible reconstruir el texto original a partir de la señal codificada. Para ello, se utiliza el árbol binario como estructura de datos.

B. El árbol binario

El árbol binario es una estructura de datos que está compuesta por un conjunto de nodos. Un nodo se entiende como una estructura que puede contener cualquier tipo de dato en su interior. Los datos dentro del nodo forman la *etiqueta* del mismo. Un nodo puede estar enlazado como máximo, a dos nodos (de ahí el nombre, *árbol binario*), los cuales están en un nivel jerárquico inferior.

Un nodo tiene como propiedades su etiqueta, un hijo izquierdo y un hijo derecho. Tales propiedades definen sus operaciones básicas: *crearNodo(nodo)*, que recibe como mínimo la etiqueta como argumento y las inserciones de los nodos izquierdos y derechos *insertarHijoIzquierdo(nodo)* e *insertarHijoDerecho(nodo)* respectivamente.

1) Estructura básica de un árbol binario:

- Raíz: es el único nodo que no desciende de ningún otro. Es el nodo jerárquicamente superior.
- Nodos hoja: se definen como aquellos nodos que no tienen descendientes.

C. Huffman y el árbol binario

El algoritmo consiste en construir el árbol binario *de abajo hacia arriba* para definir el código de cada token. Al terminar de construir el árbol, existirá un nodo por cada token, y que también contendrá la cantidad total de repeticiones de todos los nodos hijos, en cada nodo.

Los pasos para construir el árbol es la siguiente:

1) Crear un nodo por cada token, incluyendo en su etiqueta el número de repeticiones en el texto. Agregar todos los nodos a una lista.
2) Tomar y remover los dos nodos de menor frecuencia de la lista y unirlos en un nuevo nodo, el cual tendrá los dos nodos con menor frecuencia como hijos. Este nuevo nodo tendrá como etiqueta un token nulo, y como cantidad de apariciones, la suma de la cantidad de apariciones de los nodos hijos. El nuevo nodo se inserta en la lista de nodos.
3) Repetir el paso 2 hasta que exista un solo nodo en la lista. Cuando exista un único nodo en la lista de nodos, el mismo se toma como raíz del árbol de Huffman.

El árbol de Huffman es utilizado para construir el código de Huffman para cada token. Dada la naturaleza del algoritmo, los tokens con mayor cantidad de apariciones en el texto son incluidos de último en el árbol, por lo que estarán más cerca de la raíz del árbol binario.

D. Pseudocódigo

```
comprimir_huffman(NombreArchivo) <-
        leer_contenido_del_archivo(NombreArchivo, Lista_de_letras),
        convertir_letras_a_palabras(Lista_de_letras, Lista_de_palabras),

        %nodos incluyen cantidad de repeticiones, b sicamente es el histograma
        convertir_palabras_en_nodos(Lista_de_palabras, Lista_de_nodos),
```

```
%Histograma ordenado de la menor cantidad de nodos a la mayor
select_sort(Lista_de_nodos, Histograma),
crear_arbol_huffman(Histograma, Arbol_de_Huffman),

%Diccionario de Huffman
crear_diccionario(Arbol_de_Huffman, Diccionario_de_Huffman),

%Almacena en txt
escribir_diccionario(Diccionario_de_Huffman, NombreArchivo),

%Cambia datos en lista de palabras por su respectivo codigo de compresion
comprimir(Lista_de_palabras, Diccionario_de_Huffman, Lista_de_contenido_compreso),

%Almacena el contenido compreso en el archivo especificado para comprimir
escribir_compresion(CompressedList, NombreArchivo).
```

IV. Pruebas

Fig. 1. Texto de Prueba 1

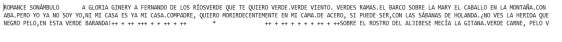


Fig. 2. Diccionario 1

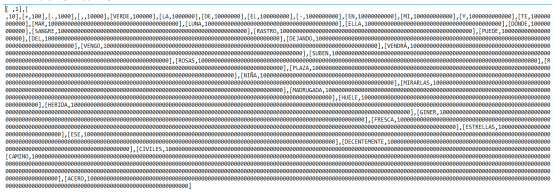


Fig. 3. Texto comprimido 1

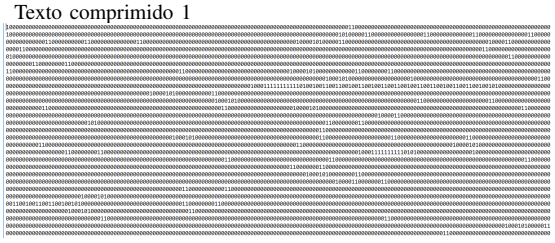


Fig. 4. Diccionario 2

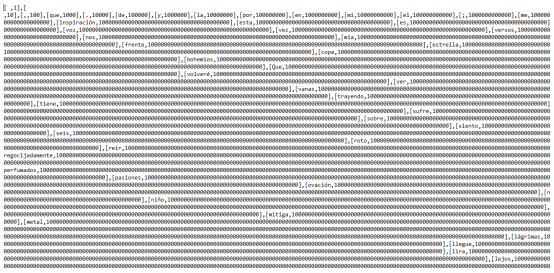


Fig. 5. Texto comprimido 2

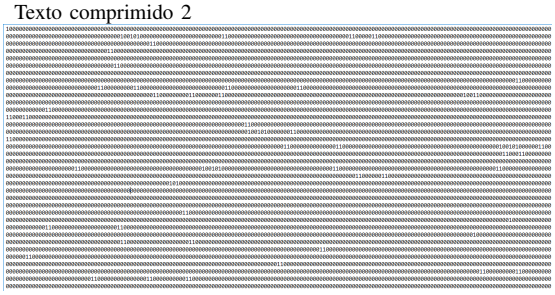
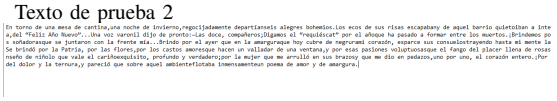


Fig. 6. Texto de prueba 2



V. Referencias

References

[1] Mamta Sharma. Compression using Huffman coding. *IJCSNS International Journal of Computer Science andNetwork Security*, 10(5):133141, 2010.