Computación y Algoritmos - Compresor de Archivos de texto

Proyecto 7

Este proyecto consiste en codificar un sencillo compresor y descompresor de archivos de texto. La idea para hacer esto es asignar códigos de tamaño pequeño a los caracteres que mas veces aparecen en el texto (a diferencia de la manera natural de asignar codigos del mismo tamaño a todos los caracteres).

Para el programa compresor se da como entrada un archivo texto grande. La salida será un archivo comprimido, el cual incluye un árbol de compresión.

Para el programa descompresor se da como entrada un archivo comprimido. La salida deberá de ser el archivo original.

1 Descripción de la metodología

El código de Huffman es una codificación óptima de modo prefijo que garantiza una decodificación única de un archivo comprimido. En si, es una técnica para asignar códigos binarios a secuencias de símbolos de un alfabeto. El objetivo es asignar el menor número de bits para cada símbolo (letra).

El codigo ASCII es un ejemplo de un código de tamaño fijo (todos los caracteres están codificados en 8 bits). Por otro lado, en un código de tamaño variable, como lo es el caso de Huffman, la idea es asignar menos bist a los símbolos que aparecen mas frecuentemente. Esta idea se utiliza para hacer compresión sin pérdida (esto significa que cuando descomprimimos el contenido del archivo es idéntico al original).

Supóngase que tenemos un alfabeto con n caracteres y también la frecuencia de aparición de cada uno de ellos. Si utilizamos un código de tamaño fijo que requiere $ceil(\log n)$ bits la tabla en Figura 1 presenta la codificación y el tamaño total de bits utilizados.

Por otro lado, utilizando el algoritmo de Huffman podemos reducir el tamaño de codificación. El algoritmo construye un árbol binario de abajo hacia arriba (bottom-up) de acuerdo a los contadores de frecuencia que le indica como fusionar sub-árboles.

El algoritmo más simple para la contrucción de este árbol utiliza adicionalmente una cola de prioridad (donde un número de frecuencia pequeño significa alta prioridad). Este algoritmo es explicado a continuación.

1. Crear un nodo hoja para cada símbolo y agregarlo a la cola de prioridad.

Character	Code	Freq	Bits
a	000	10	30
e	001	15	45
i	010	12	36
s	011	3	9
t	100	4	12
sp	101	13	39
nl	110	1	3
Total			**174**

Figure 1: Ejemplo de un código de tamaño fijo.

2. Mientras haya más de un nodo en la cola de prioridad
   (a) Desencolar los 2 nodos con la mas alta prioridad (contadores mas pequeños).
   (b) Crear un nuevo nodo interno con estos 2 nodos como hijos y con el contador de frecuencia igual a la suma de los 2 nodos.
   (c) Agregar el nuevo nodo a la cola.
3. El último nodo que queda es la ráiz del arbol que está completo.

La Figura 2 esquematiza el uso del algoritmo anterior para construir el árbol. En este árbol, la convensión utilizada es que en cada nodo del arbol (empezando desde la raiz) un bit "0" indica seguir el hijo izquierdo y un bit "1" indica seguir por el hijo derecho. De tal forma que el recorrido del arbol desde la raiz hasta cada hoja (símbolo) no da como resultado la tabla con códigos de longitud variable mostrada en la Figura 3. Nótese que el tamaño de bits utilizados se ha reducido.

2 Detalles de implementación

Solamente usaremos caracteres dentro del código ASCII del 0 al 127, es decir, no hay ñ's, no hay letras acentuadas, etc.

Una ves contruido el arbol, para comprimir es necesario ubicar el caracter en cuestión y escribir en el archivo los bits que definen el recorrido de arriba hacia abajo desde la raiz hasta la hoja (se puede recorrer de la hoja a la raíz y escribir el código invertido).

Para descomprimir seguimos el código binario desde la raíz hasta llegar a una hoja, y ese es el caracter asociado al código.

El algoritmo de compresión debe de construir el árbol para cada archivo, es decir, debe de hacer el conteo de frecuencias dinámicamente cada ves antes de comprimirlo.

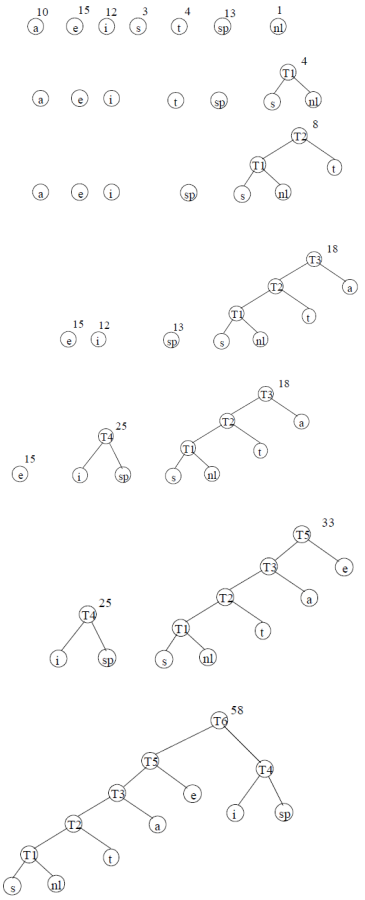

Figure 2: Construccion del árbol binario del algoritmo de Huffman para el codigo y tabla de frecuencias en Figure 1.

Character	Code	Freq	Bits
a	001	10	30
e	01	15	30
i	10	12	24
s	00000	3	15
t	0001	4	16
sp	11	13	26
nl	00001	1	5
Total			**146**

Figure 3: Código de tamaño variable construido con el algoritmo de Huffman.