

ALGORITMOS Y ESTRUCTURAS DE DATOS

Trabajo Práctico no. 8

Fecha de entrega: 19/05/2022

Tema: Tipo de datos ARBOL BINARIO - Codificación de Huffman

La compresión de datos consiste en la transformación de una cadena de caracteres de cierto alfabeto en una nueva cadena que contiene la misma información, pero cuya longitud es menor que la de la cadena original. Esto requiere el diseño de un **código** que pueda ser utilizado para representar de manera única todo carácter de la cadena de entrada. El proceso de transformación del mensaje fuente en código se conoce como **codificación**. El proceso inverso se denomina **decodificación**.

El diseño de esquemas de codificación eficientes para la compresión de datos es muy importante porque pueden reducir significativamente la cantidad de memoria y la cantidad de tiempo requeridos para almacenar un archivo de datos.

Si se utiliza un **código de longitud fija**, entonces todos los códigos de los distintos caracteres tendrán la misma longitud. Hoy en día los dos esquemas más usados son el Código Extendido de Intercambio Decimal Codificado en Binario (EBCDIC) y el Código Estándar Americano para Intercambio de Información (ASCII). Ambos códigos tienen en común que la longitud del código es la misma (fija) para todos los caracteres.

Téngase en cuenta que si existen n posibles caracteres en una cadena de entrada, entonces un código de longitud fija requerirá un número fijo de al menos $\lceil \log n \rceil$ bits para codificar cada carácter.

Es posible reducir significativamente el número de bits requeridos para representar los mensajes fuente si se emplea un código de longitud variable. En este caso, el número de bits requeridos puede variar de carácter en carácter. El objetivo es codificar los caracteres que aparecen más frecuentemente utilizando cadenas de bits más cortas, y para los caracteres que aparecen menos frecuentemente cadenas más largas.

Si se dispone de estos esquemas de codificación donde los códigos de los caracteres se representan por un **número variable de bits**, esta técnica de compresión permite ganar cantidad considerable de espacio en archivos de texto (y en muchos otros tipos de archivos). La idea es abandonar la forma como se almacenan habitualmente los archivos de texto: en lugar de emplear siete u ocho bits por carácter, se utilizarán unos pocos bits para los caracteres más frecuentes y algunos más para los que aparecen raramente.

Sin embargo, cuando los caracteres se codifican con un número variable de bits, debe emplearse algún método para determinar bits de inicio y de fin de cada código, de modo que la codificación sea unívocamente decodificable. Una forma de garantizar que una cadena de bits codificada sólo corresponde a una única secuencia de caracteres es asegurar que ningún código aparece como parte inicial de cualquier otro código. Entonces la codificación es unívocamente decodificable. Un código que tenga esta propiedad se llama **código de prefijos (o código libre de prefijos)**.

Como ejemplo considere un mensaje que contenga sólo 6 letras: **a b c d e f**, y tres posibles códigos:

	Código 1	Código 2	Código 3
Carácter	Código de Longitud fija	Código de Longitud variable	Código de Prefijo
a	000	00	11
b	001	111	1000
c	010	0011	1001
d	011	01	1010
e	100	0	0
f	110	1	1011

Usando el código 2, la decodificación no es única, por ejemplo, la cadena 010001 podría ser interpretada como: **deed**, **deaf**, **daef**, **deef**.

Esto no ocurre con el código 3 donde la decodificación es siempre única.

ALGORITMOS Y ESTRUCTURAS DE DATOS

Trabajo Práctico no. 8

Fecha de entrega: 19/05/2022

CODIGOS DE HUFFMAN

El método general para encontrar un código de longitud variable con propiedad de prefijo, fue descubierto por David Albert Huffman en 1952 y se denomina "código de Huffman". Se basa en la idea de construir **árboles binarios** para obtener una cadena de bits de longitud mínima para cualquier mensaje.

El problema planteado es un problema de optimización: **codificar una tira de caracteres con un número mínimo de bits**. Dado un conjunto de caracteres y sus probabilidades, encontrar una codificación con la propiedad del prefijo tal que la longitud promedio del código sea mínima. Se quiere minimizar el promedio de longitud de los códigos, para comprimir así la longitud de los mensajes en promedio.

La técnica para encontrar **códigos óptimos con la propiedad de prefijos**, se llama **Algoritmo de Huffman**. Es una aplicación de la **técnica GREEDY**. Los candidatos son n símbolos del alfabeto original, con la frecuencia de cada símbolo. Con cada uno de los candidatos se crea un árbol hoja, cuyo "peso" es la frecuencia de cada letra. En cada paso se selecciona los dos elementos (árboles) que tengan menor peso y se construye un árbol binario con ambos como hijos izquierdo y derecho y con peso igual a la suma de los valores de sus hijos. Al finalizar la iteración queda armado un solo árbol binario. En ese árbol los caracteres están en las hojas y sus códigos se obtienen de los caminos entre la raíz y esa hoja.

Ejemplo: Para entender mejor el algoritmo supongamos que los únicos caracteres que aparecen en la cadena son: **a,b,c,d,e,f**.

Además supongamos que el tamaño de este mensaje de entrada es de 10000 caracteres y que la frecuencia de cada letra es la siguiente:

a aparece 4500 veces

c aparece 1200 veces

e aparece 900 veces

b aparece 1300 veces

d aparece 1600 veces

f aparece 500 veces






Veamos ahora cuantos bits se necesitan con la codificación de Huffman.

Los candidatos son los caracteres con sus correspondientes frecuencias:

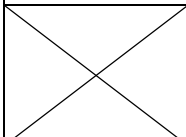

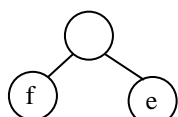



a	b	c	d	e	f
0.45	0.13	0.12	0.16	0.09	0.05

Los pasos del algoritmo son:

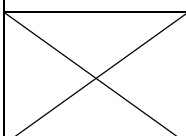
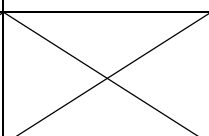
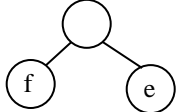


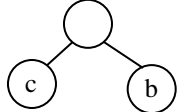
Estado Inicial:

0.05	0.12	0.09	0.45	0.16	0.13
					

Después del paso 1:

	0.12	0.14	0.45	0.16	0.13
					

Después del paso 2:

		0.14	0.45	0.16	0.25
					

ALGORITMOS Y ESTRUCTURAS DE DATOS

Trabajo Práctico no. 8

Fecha de entrega: 19/05/2022

Después del paso 3:

			0.45	0.30	0.25

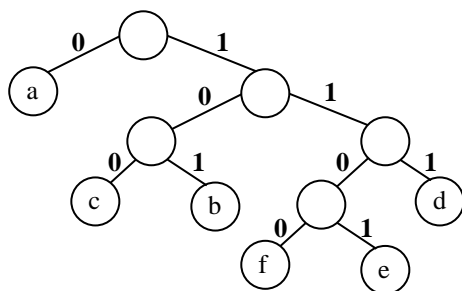
Después del paso 4:

			0.45	0.55	

Después del paso 5:

				1	

Del árbol generado por el algoritmo se obtiene la codificación para cada letra:



a → 0

b → 101

c → 100

d → 111

e → 1101

f → 1100

IMPORTANTE

De la construcción de este árbol, observe que la codificación de Huffman **es óptima pero no es única** para una dada entrada. Además, al construirse sobre un **ab**, existe un solo camino de la raíz a cada hoja y sólo las hojas pueden almacenar caracteres, entonces el código tiene **propiedad de prefijo**. Observe también que el algoritmo siempre produce árboles estrictamente binarios.

Si codificamos el mensaje de 10000 caracteres con el Código 1 se necesitan:
 $3 * 10000 = 30000 \text{ bits}$.

Si codificamos el mismo mensaje con el Código 3 se necesitan:
 $4500*2 + 1300*4 + 1200*4 + 1600*4 + 900*1 + 500*4 = 28300 \text{ bits}$.

Si se codifica con el Código de Huffman se necesitan:
 $4500*1 + 1300*3 + 1200*3 + 1600*3 + 900*4 + 500*4 = 22400 \text{ bits}$.

ALGORITMOS Y ESTRUCTURAS DE DATOS

Trabajo Práctico no. 8

Fecha de entrega: 19/05/2022

Su tarea en este práctico:

1. Escriba el algoritmo de Huffman para armar un árbol binario para las letras A..Z. usando un arreglo cuyos elementos sean de tipo ARBOL BINARIO.
2. Escriba otro algoritmo que dado el ARBOL BINARIO de Huffman devuelva un listado con el código obtenido para cada letra.
3. Escriba una implementación del adt ARBOL BINARIO.
4. Escriba un Programa que dada una tira de caracteres devuelva su codificación según el algoritmo de Huffman.

FRECUENCIA APROXIMADA DE LAS LETRAS ESPAÑOLAS:

A: 0.110845	J: 0.002889	S: 0.079605
B: 0.010895	K: 0.000083	T: 0.051378
C: 0.048778	L: 0.053524	U: 0.041887
D: 0.049769	M: 0.026494	V: 0.009698
E: 0.133336	N: 0.073580	W: 0.000041
F: 0.007965	O: 0.093925	X: 0.001940
G: 0.011638	P: 0.026700	Y: 0.008336
H: 0.006108	Q: 0.008625	Z: 0.002600
I: 0.077790	R: 0.061571	