UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Universidade Federal do Rio de Janeiro
Programa de Engenharia Elétrica
Doutorado em Eng.ª Elétrica

CPE781- COMPRESSÃO DE IMAGENS

JOÃO OLIVEIRA PARRACHO
Professor: Luiz Caloba

Rio de Janeiro, Abril 2022

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

1

CODIGOS DE HUFFMAN

1.1 ALGORITMO

A implementação dos códigos huffman encontra-se dividida em duas grandes fases: codificação, e o descodificação. No processo de codificação, o primeiro passo consiste em ler as informações estatísticas do ficheiro que pretendemos comprimir. Estes dados consistem no numero de símbolos únicos presentes no ficheiro, a frequência e probabilidade dos respectivos símbolos.

Após a leitura dos dados estatísticos, o próximo passo consiste para determinar os códigos de huffman consiste em determinar uma árvore binária composta por n nós folha que representam n os símbolos, e $(n-1)$ nos internos, onde o numero de nós internos é igual. Dado que existe n simbolos, então irão ser necessários $n-1$ nós internos. Os simbolos são ordenados de forma descendente consoante a sua probabilidade de ocorrência e são atribuídos a um nó folha. Durante $n-1$ iterações, os dois nós com menor probabilidade, que ainda não tenham sido utilizados, somam as respectivas probabilidades e atribuem o seu valor a um nó interno que ainda não tenha sido inicializado. Estes nós são associados como nó da esquerda e da direita do nó interno criado, permitindo a navegação na árvore binária . Assim que um novo nó é gerado, existe uma reordenação dos mesmos de acordo com a probabilidade associada a cada. O processo repete-se durante $n-1$ iteração até chegar ao nó root que tem probabilidade igual a 1. A reordenação dos dados, feita de forma diferente pode gerar códigos diferentes. A reordenação no código implementada pode garantir ou não códigos de huffman com a menor variância possível.

Uma vez que a árvore binária esteja construída, o próximo passo consiste em determinar os códigos huffman para cada nó folha que representa um símbolo. Para tal, é feita um varrimento dos nós internos, iniciando no nó root. Para nós a esquerda é atribuído o código 0, e para nos a direita o código 1. Os nós "filhos" tomam o codigo do nó parente mais a adição de um bit consoante se estiverem a esquerda ou direita do respectivo parente. O numero de bits de cada código encontra-se associado com a profundidade na árvore que cada símbolo esta. Símbolos na primeira camada

irão ter tamanho de 1. Percorrendo todos os nós internos e atribuindo os códigos da seguinte forma é possível gerar os códigos de huffman.

Com os códigos de huffman gerados, o próximo passo consiste na codificação dos símbolos utilizando os codigos gerados. Neste processo, o ficheiro de entrada é lido caractere a caractere. Para cada caractere é determinado o código a transmitir. Os códigos são agregados em palavras de 8 bits. É importante realçar, que um código pode ser enviado em duas palavras diferentes, e que o ultimo código lido, pode não encher uma palavra de oito bits na totalidade. O numero de bits que falta para preencher a ultima palavra de 8 bits da stream é então calculado e enviado juntamente no cabeçalho. No codificador, para garantir que a ultima palavra é composta por 8 bits realiza-se um shift de e adicionam-se zeros a direita.

Antes de iniciar o processo de codificação, é necessário escrever um cabeçalho na stream codificada, para que seja possível descodificar a stream sem perdas. Este cabeçalho é composto por 7 bytes , mais 5 bytes por cada símbolo único. No cabeçalho são transmitidas as seguintes informações: Nº de símbolos unicos, nº de bits a descodificar na ultima palavra da stream codificada, nº total de bits comprimidos, se os códigos de huffman utilizam mínima variância e para cada símbolo, o caractere associado e a frequência do caractere no ficheiro de entrada. A Figura 1 mostra o exemplo de um cabeçalho gerado.

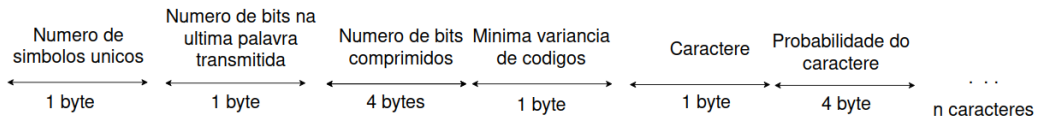

Figure 1: Cabeçalho da stream codificada

No lado do descodificador após a recepção da stream codificada, o primeiro passo para descodificar a stream, consiste na leitura dos dados presentes no cabeçalho. Após a leitura dos mesmo, a árvore binária e os códigos de huffman são reconstruidos, de forma semelhante como foram construídos no codificador. A stream codificada é lida bit a bit. A medida que vamos recebendo bits, estes vão permitir percorrer a árvore binária, começando no nó root, até chegar a um nó que tenha um símbolo associado. Se o bit recebido for um 0 então percorro a árvore pela esquerda, se for um 1 percorro pela direita. Assim que um nó folha é alcançado, o respectivo carácter é escrito para o ficheiro descodificado e a pesquisa reinicia no nó root novamente ate todos os caracteres serem descodificados.

Para utilizar o codificador implementado, são fornecidos dois executáveis, um para codificar ("./enconde") e outro para descodificar ("./decode"). O ficheiro de

codificação espera receber dois argumentos, o caminho para o ficheiro a ser codificado, e o caminho para o ficheiro onde pretende-mos escrever a stream com o ficheiro codificado. Adicionalmente pode-se adicionar dois argumentos, um para fazer com que os códigos sejam de menor variância possível ("-m"), e outro para escrever no terminal as informações estatísticas do resultado de codificação ("-v"). Um exemplo de como correr o ficheiro de codificação será: "./encode Inputfile OutputFile -v -m". O ficheiro para descodificar espera receber apenas dois argumentos, o caminho para a stream a ser descodificada, e o caminho para o ficheiro descodificado. Neste ficheiro não é necessário indicar se os códigos São de menor variância uma vez que esta informação ja é enviada através do cabeçalho da stream. Um exemplo de como correr este ficheiro sera: "./decode Inputfile OutputFile". O ficheiro de codificação tem um mecanismo implementado que informa ao utilizador caso a compressão com os codigos huffman seja ineficiente, ou seja, se o tamanho do ficheiro de stream é maior que o tamanho do ficheiro a ser comprimido. Isto pode acontecer, em ficheiros de texto pequenos, onde o overhead do cabeçalho é muito grande.

Para verificar o funcionamento dos codigos de Huffman com menor variância possível, a mensagem 'ABCDVV' foi codificada. Os nós internos não atribuído estão representados com uma frequência de 0 e sem caractere, os internos atribuídos estão representados com o caracter 'z'. Os nos foram se organizando da seguinte forma:

- V - 2; D - 1; A - 1; B - 1; C - 1; - 0; - 0; - 0; - 0;
- (1º Iter): z - 2; V - 2; D - 1; A - 1; B - 1; C - 1; - 0; - 0; - 0;
- (2º Iter): z - 2; z - 2; V - 2; D - 1; A - 1; B - 1; C - 1; - 0; - 0;
- (3º Iter): z - 4; z - 2; z - 2; V - 2; D - 1; A - 1; B - 1; C - 1; - 0;
- (4º Iter): z - 6; z - 4; z - 2; z - 2; V - 2; D - 1; A - 1; B - 1; C - 1;

Como se pode verificar, sempre que um no interno obtém a mesma frequência que um símbolo este aparece primeiro após a ordenação dos nós, garantindo a menor variância dos códigos.

1.2 RESULTADOS

Table 1: Precisão dos agrupamentos realizados

Caractere	**Frequência**	**Tamanho código**	**Código**
Espaço	700260	3	111
E	426854	3	110
A	399089	3	100
O	343164	3	000
S	284436	4	1101
R	198881	4	0010
I	169567	5	11011
D	153718	5	10011
U	138492	5	10101
M	133496	5	11001
N	132967	5	01001
T	127353	5	00001
C	87715	6	101011
L	76815	6	100011
,	72864	6	000011
P	70200	6	100101
LF	65072	6	110001
V	46816	6	101010
H	46151	6	001010
.	42512	7	1001011
Q	38334	7	0001011
F	32408	7	1010001
B	30173	7	0010001
G	27124	7	1011010
;	17330	8	01000101
J	14890	8	11111010
Z	14717	8	01111010
1	14153	8	00111010
2	9955	8	00011010
:	8455	9	011000101
3	6359	9	010011010
4	4580	10	0011000101
X	4182	10	0010111010
5	3741	10	0010111010
6	3345	10	0010111010
7	3116	10	0010111010
8	2942	10	0010011010
9	2802	10	0010011010
0	2709	11	00011000101
]	1218	12	000011000101
[	1217	12	000011000101

Para testar a eficiência do algoritmo implementado, foi utilizado um ficheiro de texto com as seguintes características:

- Texto em português
- Numero de símbolos únicos: 42 símbolos
- Tamanho de ficheiro: 3992708 bytes
- Tamanho médio de cada símbolo: 8 bits/símbolo

Os resultados da construção dos códigos huffman utilizando códigos com menor variância encontram se sumarizados na Tabela 1. De acordo com a tabela verifica-se, tal como seria esperado num texto em português, que os dois caracteres mais frequentes são o espaço e a letra A.

Os resultados obtidos após a compressão foram os seguintes:

- Tamanho da stream em bytes sem cabeçalho: 2156070 bytes
- Tamanho do cabeçalho: 217 bytes
- Tamanho da stream: 2156289 bytes
- Tamanho médio de cada símbolo: 4.29 bits/símbolo
- Taxa de compressão: 1.87
- Data rate saving: 48.43 %

Como se pode verificar, a compressão do ficheiro utilizando códigos de huffman foi realizada com sucesso obtendo uma taxa de compressão de 1.86, reduzindo o tamanho médio d cada símbolo de 8 bits/símbolo para 4.29 bits/símbolo. Em termos de compressão não se verificou diferença em utilizar códigos d huffman com mínima variância.