Atividade Acadêmica - Estrutura de Dados I

Paulo Francisco da Silva Freitas

[1]Departamento de Tecnologias e Linguagens -
Universidade Federal Rural do Rio de Janeiro (UFRRJ)
R. Governador Roberto Silveira S/N - Nova Iguaçu -
Rio de Janeiro - RJ - Brasil

paulofrsf@gmail.com

Resumo. *Neste relatório será apresentado a proposta de trabalho e informações e testes sobre a Codificação de Hufffman. Com a codificação de Huffman podemos pegar textos e compacata-los atras da Codificação de Huffman, que codifica os caracteres de acordo com a sua frequência. Tal processo é feito através da construção de uma árvore de Huffman que separa os caracteres de acordo com a sua frequência no texto, e quanto maior for ela, mais próximo da raiz ficará o caractere desejado. O algoritmo é bastante efetivo em textos com pouca variedade de caracteres, porém, conforme a sua diversidade aumenta, o taxa de compactação diminui.*

1. Problema

Compactar um arquivo de texto usando a codificação de Huffman, com isso é gerada uma árvore que nos permite codificar os caracteres de acordo com a sua frequência. Quanto maior a frequência do caracter, maior é a prioridade dele.

2. Codificação de Huffman

O Método de Huffman consiste em organizar os caracteres em uma árvore de acordo com a sua frequência, e ao percorrê-la registra 1 se percorrer o nó esquerdo/direito (depende da implementação) e 0 se percorerr o outro caminho até chegar ao nó folha e repetir isso de maneira recursiva para todos os nós.

O resultado será uma sequência binária, que será referente a cada caracter, com seus respectivos tamanhos. Depois, o texto original será traduzido para seu código correspondente e guardado em um array. Esse array então será separado em partes de 8, e esses 8 bytes serão convertidos em 1 byte, então é escrito no arquivo para obter-se a compactação. Também é gerada uma tabela que contém o codigo referente de cada caractere, e o tamanho do array antes de ser compactado, já que se ele não for um múltiplo de 8, o último byte sera preenchido com bits que não fazem parte da codificação e serão ignorados na descompactação.

No processo de descompactação, será feito uma associação do código em binário com o seu caracter correspondente na tabela, e como a codificação de Huffman gera códigos aonde não há codigos prefixos de outros, não há uma sobreposição. Quando o número de bits atinge o mesmo numero do texto original, a descompactação acaba, fazendo os bits adicionais que foram adicionados(se necessário) serem ignorados e salva em um novo arquivo a mensagem descompactada.

Pseudocódigo

while(tamanhodoalfabeto¿1)

s0:pegamenorfreq(alfabeto)

s1:pegamenorfreq(alfabeto)

u=novono

u.filho0 = s0

u.filho1 = s1

u.freq = s0.freq + s1.freq

insere u no alfabeto

fim while

for(cada folha em folhas(u))

codificacao[folha]= percorrecaminhoatefolha(folha)

fim for

3. Complexidade

A complexidade da codificação de Huffman é O(n log n) em um conjunto de n caracteres. Um texto grande necessita de mais comunicações com o buffer ao transcrever os códigos no arquivo compactado, o que implica que mesmo que o arquivo tenha uma baixa quantidade de caracteres diferentes, se ele for longo, o tempo de compactação aumentará.

4. Testes

Foram realizados casos testes com o algoritmo de codificação de Huffman, os casos foram escolhidos por mim, baseando-se no critério de tamanho do texto e diversidade de caractéres. Segue abaixo o conteúdo de cada caso e a tabela com as variáveis observadas.

Caso 1.

Texto ”Hello World”

Caso 2.

Texto ”A programming language is a formal computer language or constructed language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs to control the behavior of a machine or to express algorithms.”

Caso 3. ”The description of a programming language is usually split into the two components of syntax (form) and semantics (meaning). Some languages are defined by a specification document (for example, the C programming language is specified by an ISO Standard), while other languages (such as Perl) have a dominant implementation that is treated as a reference.”

os tempos estão em segundos

Input	Tamanho Antes	Tamanho Depois	Tempo Compactação	Tempo Descompactação	Taxa de Redução
Caso 1	11	4	0.000030	0.000020	63.64%
Caso 2	268	139	0.000073	0.000054	48.13%
Caso 3	394	194	0.000104	0.000056	45.20%

Pode-se observar na tabela que o tempo de compactação varia mais que o tempo de descompactação, e que a taxa de redução cai de acordo com o tamanho e a o número de caracteres diferentes no texto, o que comprova que a eficiência da codificação de Huffman cai de acordo com essas variações, como foi citado a cima.

5. Conclusão

A codificação de Huffman é uma boa opção para a compactação de dados, pois é uma maneira efetiva para comprimi-los. Ele também é relativamente rápido, devido a sua boa complexidade. Podemos observar que quanto maior a redundancia do texto, conseguimos uma compressão maior, já que haverá maior repetição de caracteres.

Porém um ponto negativo é que quando se usa uma árvore de codificação predefinida baseada na probabilidade das frequencias de cada caracter, se um texto não satisfizer as frequências da árvore,ou a escolha das frequências for feita de maneira errônea o algoritmo fica mais lento, mas para os casos em que a aparição dos caracteres é de acordo com a da árvore, a codificação é bastante eficiente.

Conclui-se que a Codificação de Huffman, em geral, é uma técnica efetiva de compressão de dados, porém sua eficiência pode cair conforme a escolha da codificação na tabela for feita.

6. Referências bibliográficas

T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 3rd edition.