APLICAÇÃO DO ALGORÍTIMO DE HUFFMAN NA COMPRESSÃO DE ARQUIVOS DE TEXTO

Leonardo Nunes de Oliveira

[1]Ciência da Computação – Centro Universitário IESB
Asa Sul – Brasília/DF – Brasil

ti.userleo@hotmail.com

Abstract. *The project consists of programming an application using the Huffman Algorithm data structure for data compression and decompression, seeking less processing and time usage in compressing and decompressing text files compared to the software most used today for this purpose.*

Resumo. *O projeto consiste na programação de uma aplicação utilizando da estrutura Algorítimo de Huffman para compressão e descompressão de dados, buscando um menor uso de processamento e tempo na compressão e descompressão de arquivos de texto em relação a softwares mais usados hoje para essa finalidade.*

1. Introdução

Em 1950, David A. Huffman como um dos grandes pioneiros da Ciência da Computação, enquanto estudante de doutorado desenvolveu um método de compressão/expansão de dados.

Deste método, foram originados várias aplicações e softwares que prestassem serviços de compressão/descompressão de dados, como sendo os mais usados hoje no mundo o WinRAR e o Winzip.

Alta capacidade de processamento e espaço de armazenamento não é um problema para computadores de alta performance hoje, contudo como nem todas as pessoas possuem condições para poder adquirir esses produtos, quando muda-se o cenário para máquinas limitadas com processadores de baixo desempenho e pouco espaço de armazenamento, para realizarem serviços de compressão em uma grande quantidade de dados, como por exemplo em um grande arquivo de texto contendo muitos dados de informações, surgiu a necessidade de uma aplicação de compressão de dados que utilize menos processamento em comparação aos demais softwares mais usados hoje para esta finalidade, para não ocorrer travamentos ou até problemas durante sua execução devido a pouca capacidade do processador.

2. Objetivo geral

Uma das aplicações que podem ser utilizadas como solução e alternativa para estas máquinas de baixo desempenho neste contexto é a aplicação proposta neste projeto.

Compressões de arquivos que antes geravam muita lentidão por conta do baixo desempenho do computador utilizando aplicações e serviços pesados devido a ausência de alternativas, poderão ser feitas utilizando menos processamento, espaço de armazenamento e tempo.

A finalidade deste projeto é programar uma aplicação utilizando o Algorítimo de Huffman, que realize a compressão/descompressão de dados utilizando baixo processamento, tempo e espaço de armazenamento em relação aos demais softwares mais baixados e conhecidos para compressão de dados em computadores.

Pesquisas feitas em sites e fóruns popularmente conhecidos sobre melhores alternativas de softwares utilizados para compressão de arquivos sugerem os seguintes aplicativos para este serviço: WinZip, WinRAR, 7-Zip e IZArc. Diante disso, a aplicação deste projeto terá como objetivo melhores resultados de processamento, espaço de armazenamento e tempo na compressão de arquivos em testes comparando-a com os 4 softwares citados anteriormente.

3. Referencial Teórico

Compressão de dados consiste em reduzir o tamanho de um arquivo em um espaço menor de armazenamento, ocupando uma menor quantidade de bytes. Existem dois tipos de compressão: compressão com perda e compressão sem perda.

A compressão com perda caracteriza-se por possuir informações resultantes do processo de compressão diferentes da informação original, é muito utilizado para compactar arquivos de áudio e vídeo, uma vez que as alterações e perdas sofridas no processo não torne o arquivo inutilizável.

Já na compressão sem perda de dados o resultado do processo de compactação não sofre alterações ou modificações se comparado ao arquivo original, mantendo sua integridade. O método de compressão de Huffman é um tipo de compressão sem perda de dados, para que as informações do resultado da compressão não sofra alterações em relação ao arquivo original.

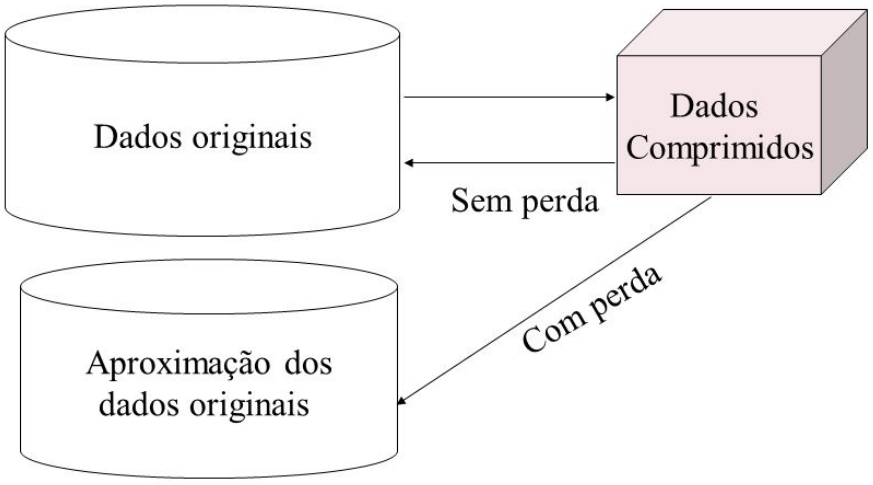

Figure 1. Tipos de compressão de dados

O algoritmo de Huffman caracteriza-se pelo uso de uma quantidade de bits para identificação de um caractere. Na Tabela ASCII(American Standard Code for Information Interchange), que é usada como base para identificação de todos os caracteres que existem atualmente, cada caractere possui o tamanho de 4 bytes, que corresponde a 32 bits. Quando utiliza-se de Huffman para realizar a compressão de um arquivo de texto, a

quantidade de bits usada para identificação de cada caractere é reduzida praticamente em 60 por cento, ocupando menor espaço de armazenamento.

A codificação dos caracteres de um texto feita pelo algorítimo de Huffman baseia-se no caminho percorrido em uma Árvore binária, onde esta armazenará todos os caracteres do texto.

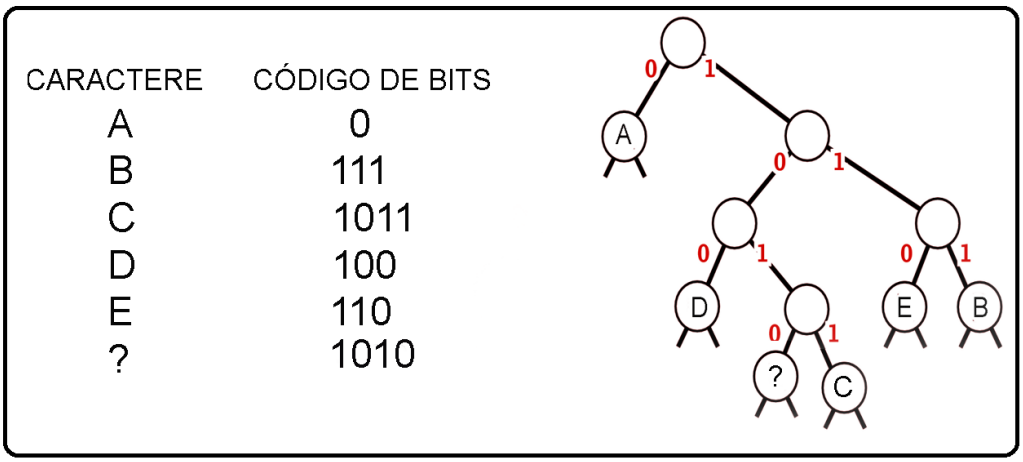

Figure 2. Algorítimo de Huffman utilizando Árvore binária

Segundo Aaron M Tenenbaum, em seu livro ”Estruturas de Dados Usando C” denomina-se Árvore binária como um conjunto finito de elementos que está vazio ou é particionado em três subconjuntos disjuntos. Uma Árvore binária é um armazenamento de um tipo de informações, possuem nós e folhas em sua composição onde as informações são armazenadas.[Tenenbaum 2004]

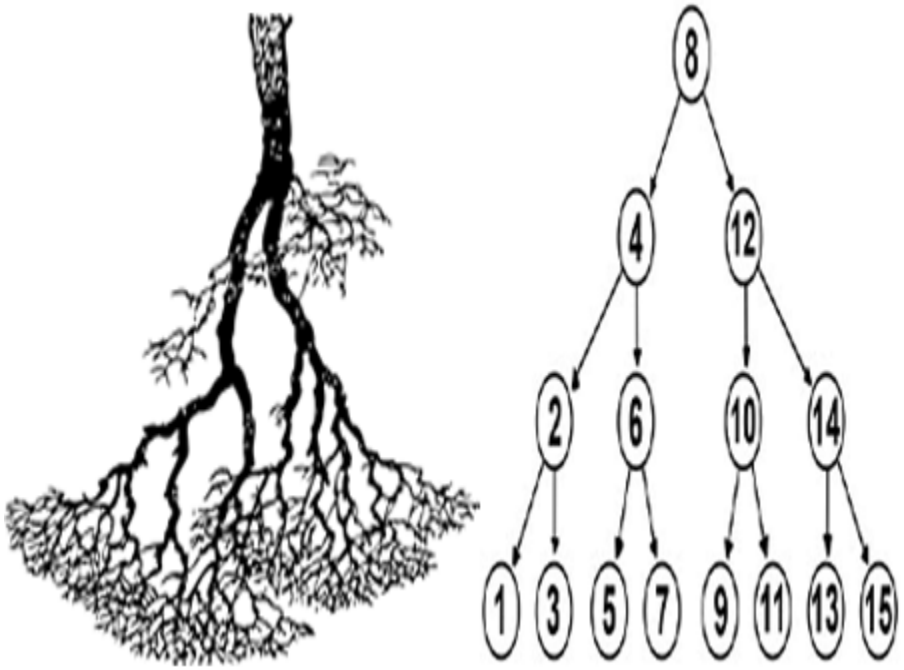

Figure 3. Estrutura de uma Árvore binária

Árvore binária é utilizada em Huffman devido a sua alta capacidade de armazenar

dados de um determinado tipo. Conforme a figura 3, se 8 é a raiz da árvore binária e 4 é a raiz de sua subárvore direita ou esquerda, então 8 é o pai de 4 e 4 é a folha direita ou esquerda de A, também pode denominar-se nó de 2 novas folhas.

É uma estrutura que funciona de forma mais abrangente em relação a uma Lista, por possuir de nós e folhas, nos nós possuem endereços apontando para as folhas no qual se tem os dados armazenados da direita e da esquerda.

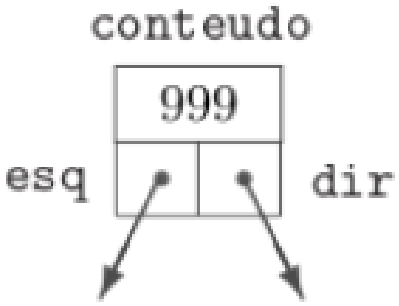

Figure 4. Conteúdo e endereços do nó de uma Árvore binária

O conteúdo de cada folha da árvore será obtido a partir de uma Lista contendo as frequências de cada caractere do texto, em ordem crescente.

Listas funcionam armazenando ordenadamente valores como inteiros, decimais, caracteres, frases, bytes, etc... com o mesmo valor podendo estar uma ou mais vezes na Lista. Uma grande vantagem de se usar Listas em programação é poder armazenar de forma simples e ordenada valores para compor uma coleção de dados.

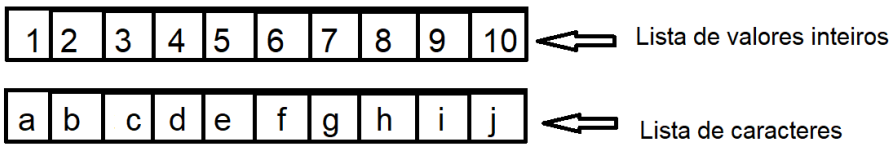

Figure 5. Estrutura de dados Lista

As operações realizadas com Listas são: criação de uma lista, remoção de uma lista, inserção de um elemento na lista, remoção de um elemento da lista e acesso de um elemento na lista.

Apesar de serem muito utilizadas pelo fácil e prático manuseio de seus elementos, uma grande desvantagem das Listas é que necessitam de uma quantidade fixa de

armazenamento antes de ser alocadas, mesmo quando a estrutura estiver usando uma quantidade menor ou possivelmente nenhum armazenamento.

Em sua estrutura contém o valor do elemento armazenado e um ponteiro apontando para o próximo elemento, até que o ponteiro do próximo elemento esteja com o valor nulo, não apontando para nenhum elemento.

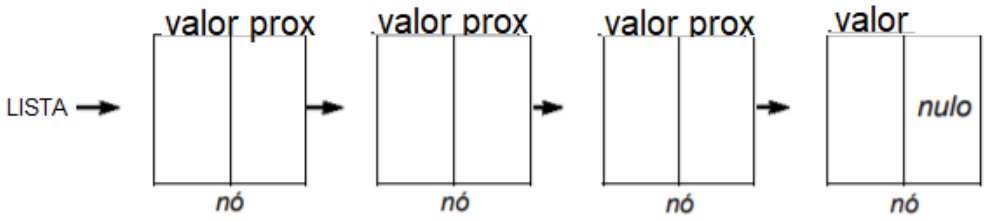

Figure 6. Conteúdo de uma estrutura de uma Lista

4. Desenvolvimento do Algorítimo de Huffman em Java

Figure 7. Linguagem de programação e IDE que serão utilizados no projeto

Utilizando a IDE Eclipse versão 4.15, a aplicação será feita em ambiente de programação Java, utilizando das estruturas de dados Árvore binária e Lista para a implantação do Algorítimo de Huffman.

Por ser uma linguagem de programação com uma arquitetura neutra e portável, podendo ser utilizada em diversos sistemas operacionais, ter alta performance, e apresentar segurança e solidez, Java foi escolhida para implantação da aplicação.

4.1. Classes e Métodos

Serão programadas 4 classes para comporem o algorítimo de Huffman, denominadas: HuffmanThree, HuffmanNode, HuffmanLeaf e HuffmanCode.

4.1.1. HuffmanThree

A classe HuffmanThree, ou Árvore de Huffman, será uma classe abstrata, responsável pela estrutura e armazenamento das frequências contidas nos nós da Árvore binária. Seus métodos serão o método construtor, que é responsável pela inicialização da classe em Java, recebendo como parâmetro a frequência do nó, e o método compareTo que fará a implementação da interface Comparable para ordenação da Lista.

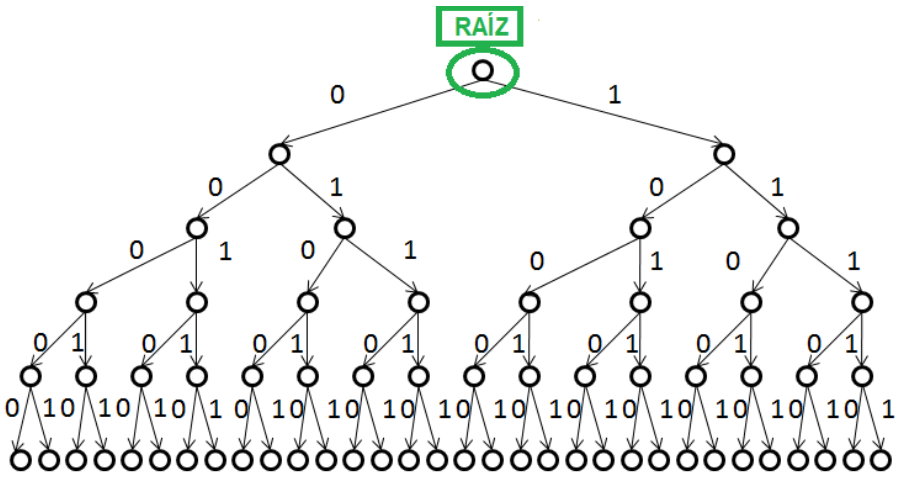

Figure 8. Estrutura da Árvore binária, com início em sua raíz

Pelo fato de ser uma classe abstrata, servirá como modelo para as outras classes HuffmanNode e HuffmanLeaf, já que ambos serão parte da estrutura da árvore e terão que ter características semelhantes.

4.1.2. HuffmanNode

Responsável por fazer o apontamento para os nós da esquerda e da direita da Árvore binária, a classe HuffmanNode ou Nó de Huffman servirá como geradora de sub-árvores da classe HuffmanThree, por meio das frequências recebidas como parâmetro da sua função construtora.

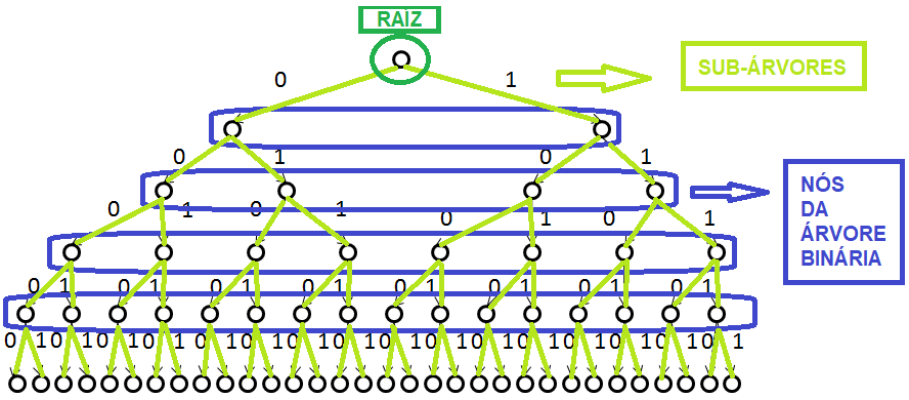

Figure 9. Sub-árvores e nós de uma Árvore binária

4.1.3. HuffmanLeaf

Na última classe da Árvore binária, denominada HuffmanLeaf ou Folha de Huffman, será armazenado o valor do caractere codificado nas folhas.

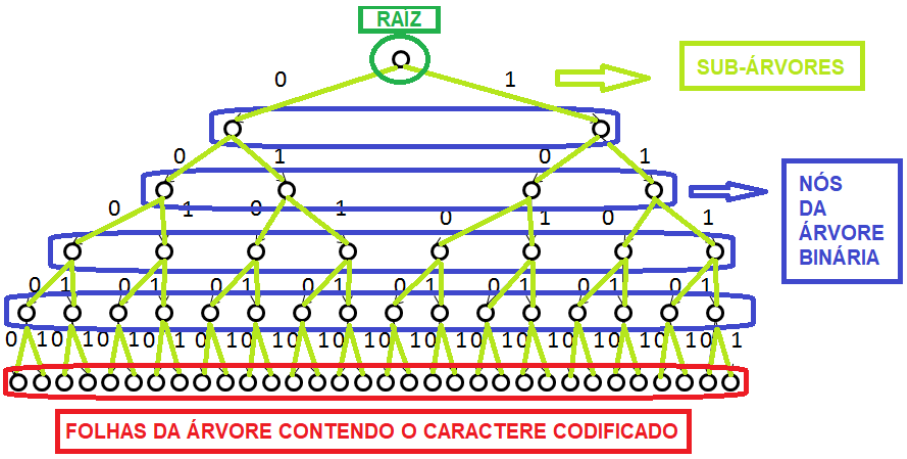

Figure 10. Folhas ou filhos resultantes das sub-árvores e nós de uma Árvore binária

4.1.4. HuffmanCode

Será a classe principal ou main da aplicação, contendo a entrada do arquivo de texto, metódos e funções invocando a Árvore binária para a compressão de seus dados.

4.2. Execução da aplicação

Vamos usar a seguinte sequência de caracteres de um arquivo de texto para realizar a compressão:

A inteligência artificial está sempre evoluindo.

Figure 11. Texto usado para compressão da aplicação

4.2.1. Frequência

Inicialmente, o algorítimo vai percorrer a sequência de caracteres do texto, contando as ocorrências de cada caractere e armazenando-as em uma Lista, de forma que estejam ordenadas da menor para a maior.

A classe HuffmanThree é responsável não só por todas as etapas da execução do algorítimo, mas também em especial essa. Nela que serão armazenadas as frequências de cada caractere, por ser uma classe abstrata seus atributos e métodos serão usados também em outras etapas.

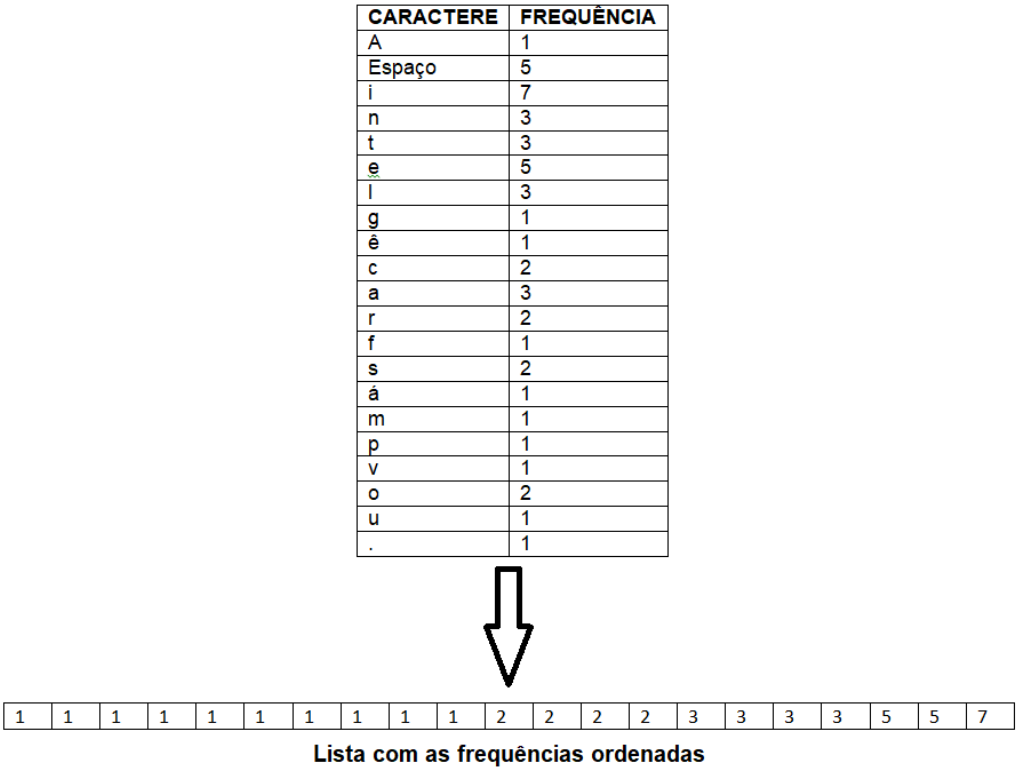

Figure 12. Frequências ordenadas dos caracteres do texto contidas em uma Lista

References

Tenenbaum, A. M. (2004). *Estruturas de Dados Usando C*. São Paulo: Makron Books.