Análise de complexidade de um algoritmo para visualização da árvore de frequência da codificação de Huffman

Camila Moser[1], **Lucas de Freitas**[2]

[1]Faculdade de Informática
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Av. Ipiranga, 6681 – Partenon – Porto Alegre – RS – Brasil

Resumo. *A codificação de Huffman é um dos primeiros algoritmos para compressão de dados. A codificação utiliza códigos de diferentes comprimentos, atribuindo códigos mais curtos para caracteres de maior frequência. Para calcular a frequência de caracteres é utilizada uma árvore binária. O objetivo deste trabalho é determinar a complexidade de um algoritmo que permite a visualização da árvore utilizada na compressão. A conclusão é de que a complexidade depende do valor de (indicar valor de entrada) e pode ser descrita como (classe de complexidade).*

1. Introdução

O objetivo deste trabalho é determinar a complexidade de um algoritmo que permite a visualização da árvore utilizada na compressão. A tarefa proposta apresenta três etapas: (a) desenvolver uma implementação do algoritmo de Huffman, (b) exportar a árvore utilizada pela implementação em um arquivo texto no formato utilizado pelo sistema graphviz e (c) determinar a complexidade do algoritmo utilizado na exportação.

2. Metodologia

Foi realizada uma consulta à Wikipédia no verbete Huffman Tree, no qual se pode encontrar uma representação gráfica da árvore utilizada no algoritmo.

Foi implementado este algoritmo que se encontra no endereço https://github.com/CamilaMoser/Huffman. Como referência utilizamos o enunciado proposto na disciplina de Algoritmo e Programação III, onde é demonstrado um exemplo de um valor de entrada e seu resultado de forma gráfica.

3. Implementação da codificação de Huffman

A implementação foi desenvolvida na linguagem Java, para isso foi utilizada duas estruturas de armazenamento auxiliares para gerar a árvore, a primeira é um hashMap para armazenar como chave o carácter, e como valor sua frequência, adotamos esta escolha pois a estrutura não permite chaves (caracteres) repetidos, e também permite relacionar o caractere à sua frequência, também foi utilizado um objeto do tipo Nodo criado no projeto para armazenar informações como: o caractere, a sua frequência, uma flag para indicar se o nodo da árvore é folha, sua sequencia binária e outras três informações para informar com que nodos ele está interligado, estes três atributos irão servir como auxilio para percorrer a árvore. Os passos para a realização da construção desta estrutura está descrito a seguir:

1. Testamos se a palavra está vazia.
2. Criamos a estrutura $mapWord$ para armazenar o caractere como chave e a sua frequência dentro da frase como valor.
3. Criamos a estrutura $list$ e adicionamos nela o objeto $Nodo$ com a chave e o valor do $mapWord$.
4. Após cria a lista é chamado o método recursivo $buildTree()$.

O método $buildTree()$ irá conectar os nodos criados dentro da estrutura $list$, para isso é verificado quais nodos são folhas, neste caso, estes nodos ficarão na ordem inversa do resto da árvore, ou seja, nodos com maior frequência ficarão à esquerda de seu nodo pai, já os de menor frequência ficarão à direita do seu respectivo nodo pai.

4. Exportação no formato graphviz

Após ter sido criada a estrutura da árvore de Huffman, foi implementado o método $writeInGraphviz()$, esta função permite exportar a árvore para um arquivo chamado graph.gv em formato do graphviz, este arquivo está localizado no GitHub no endereço https://github.com/CamilaMoser/Huffman/blob/master/huffman/graph.gv. Para escrever este arquivo, utilizamos a recursivade com forma de percorrer a árvore a partir o nodo raiz até os nodos folhas, cada linha deste arquivo representa um nodo apontando para o seu nodo filho. Neste método o algoritmo percorre primeiramente todos os nodos da esquerda da árvore após percorridos todos, ele começa a percorrer os nodos a direita da arvore, para isso, nos baseamos no algoritmo de percurso pré-ordem, como foi estudado na disciplina de Algoritmos e Programação III.

Na apresentação da estrutura gráfica da árvore pelo graphviz, os nodos folhas apresentados com a informação seu caractere, sua frequência e sua sequência de bits, no seguinte formato: $caractere + frequência + sequênciaDeBits$. Nos demais nodos da árvore, apenas será visualizado a soma das frequências dos nodos filhos deste nodo. Para isso, foi necessário ao inserir os nodos da árvore, saber quais eram folhas.

4.1. Algoritmo

Neste tópico é apresentado o algoritmo implementado para fazer a escrita da árvore no arquivo graph.gv como mostra o exemplo abaixo:

Algorithm 1: método principal que chama o recursivo

Result: Arquivo no formato para leitura do graphviz

```
1 writeGraphviz(raiz);
```

Algorithm 2: Algoritmo para exportar árvore para o graphviz recursivamente

Result: Arquivo no formato para leitura do graphviz

```
if nodo == NULL then
    Retorna;
end
if nodo.left != NULL then
    if nodo.left == leaf then
        Escreve no arquivo nodo.frequency-> nodo.left.character +
          nodo.left.frequency + nodo.left.sequencyBinary; e pula linha;
    else
        Escreve no arquivo nodo.frequency-> nodo.left.frequency e
          pula linha;
    end
    writeGraphviz(nodo.left);
end
if nodo.right != NULL then
    if nodo.right == leaf then
        Escreve no arquivo nodo.frequency-> nodo.right.character +
          nodo.right.frequency + nodo.right.sequencyBinary; e pela
          linha;
    else
        Escreve no arquivo nodo.frequency-> nodo.right.frequency e
          pula linha;
    end
    writeGraphviz(nodo.right);
end
```

1. Método principal chama o método recursivo passando como parâmetro o nodo raiz.
2. Dentro do método recursivo, é verificado se o nodo é igual a nulo, se for, ele sai do método recursivo pois toda a arvore já foi percorrida.
3. Verifica se o nodo esquerdo é igual a nullo
4. Logo após verifica se o nodo esquerdo é um folha, se for, escreve o nodo com a $frequencia$ apontando para o nodo esquerdo, sendo que este nodo estará no formato $caractere + frequência + sequenciaDeBits$.
5. Se o nodo não for folha, imprime o nodo e o seu nodo esquerdo no mesmo formato $frequência$.
6. Finalizada a condição, é chamado o método novamente passando como parâmetro o nodo esquerdo
7. Após percorrer todos os nodos à esquerda do raiz, o método começa a percorrer todos os nodos à direta

4.2. Exemplos

Abaixo segue o exemplo de uma árvore gerada pelo algoritmo writeInGraphviz() a partir da string ”aaaaabbbbcccdde”que se encontra no enunciado do trabalho final de Algoritmos e Programação III, neste exemplo a árvore foi gerada atráves do site https://webgraphviz.com

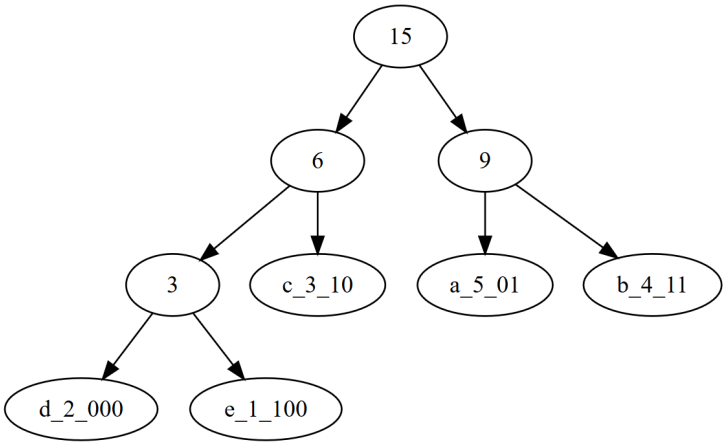

Figura 1. árvore gerada pela string ”aaaaabbbbcccdde”no graphviz

4.3. Análise

Para fins de análise da complexidade do algoritmo writeInGraphviz(), foi elabordo testes com 10 (dez) sequências de caracteres, nelas não há repetições de seus caracteres, os testes foram realizados desta maneira, para que a observação da complexidade do algoritmo seja baseada no número de nodos dentro de uma árvore, e somente desta forma poderia ser realizada. Abaixo segue uma tabela com a coluna esquerda preenchida com a sequencia dos caracteres e em outra, possui a contagem de operadores do algoritmo.

Tebela de teste	
sequência de caracteres	contagem de operadores
AB	40
ABC	68
ABCD	96
ABCDE	124
ABCDEF	152
ABCDEFG	180
ABCDEFGH	208
ABCDEFGHI	236
ABCDEFGHIJ	264
ABCDEFGHIJK	292

Podemos concluir que este algoritmo tem como complexidade linear O(n), chegamos a esta conclusão, a partir do gráfico que foi criado através dos dados gerados pelos testes feitos. Abaixo segue o gráfico onde o eixo linha demonstra o número de nodos gerados na arvore a partir da sequência de caracteres que é passado como parâmetro, e no eixo coluna o numero de operadores que foram executados ao executar o algoritmo $writeInGraphviz$

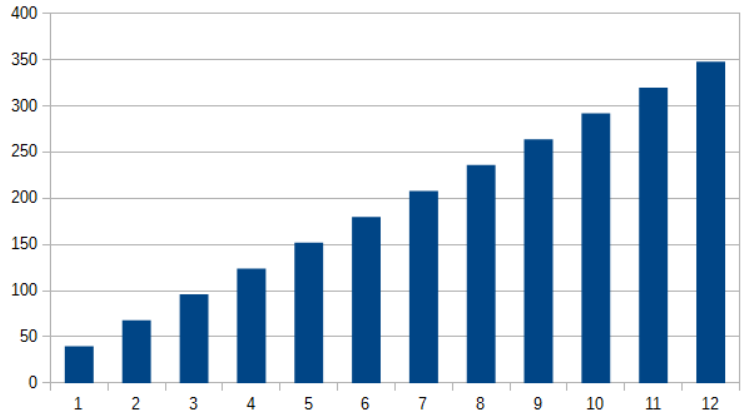

Figura 2. Gráfico gerado

5. Conclusão

No desenvolvimento do trabalho foi encontrado como problema, o modo como iria ser gerado o arquivo, com pesquisas na internet e exemplos mostrados em aula, encontramos um formato simples de escrever a árvore no formato do graphviz e gerar a árvore da maneira correta. Outro problema detectado foi a forma como deveríamos escrever os nodos folhas com as informações necessárias para serem apresentadas na árvore gráfica, para isso, chegamos na solução de armazenar no objeto nodo a informação se ele é um nodo folha ou não, a sequencia de bits que é correspondida para cada nodo que seja folha, seu caractere e sua frequência dentro da palavra ou frase.

Referências

https://en.wikipedia.org/wiki/Huffman$_coding$