Trabalho Prático 3

Pedro Bernardo Goulart Parreira

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

pedrobgoulart@ufmg.br

1 Introdução

Neste trabalho, implementamos o Algoritmo de Huffman para compressão de arquivos caractere a caractere, e avaliamos sua performance.

2 Método

Em vários momentos utilizamos *arrays* indexados pelo código ASCII dos caracteres de forma a facilitar seu manuseio, nomeadamente como dicionário na obtenção das frequências de cada caractere. Além disto, ao construir a árvore a partir do arquivo de entrada, devemos saber a todo momento o menor dos nós em frequência. Inicialmente implementamos isto com diversas ordenações do *array* de nós, porém, dada a alta complexidade assintótica desta solução, ficou evidente que a melhor estrutura para isto seria uma *Heap* implementada com um vetor imitando aquele da STL, a qual podemos usar para manter os itens ordenados na inserção e facilmente extrair o menor. Por fim, o projeto é encapsulado no TAD Huffman, servindo de interface para o usuário.

3 Análise de Complexidade

Seja n o número de caracteres no arquivo. Para construir a árvore de Huffman, precisamos primeiramente descobrir a frequência de cada caractere o que constitui uma operação $O(n)$. Em seguida criamos nós com estes caracteres, mais uma vez, $O(n)$. A construção da árvore em si na compressão é $O(n \log n)$ pela complexidade assintótica das operações de *Heap*. Já na descompressão, assim como a escrita dos caracteres comprimidos, esta operação é linear. Por fim, descobrir o qual caractere corresponde a um dado código é $O(\log n)$ n vezes. Desta forma, temos que a complexidade assinótica da compressão é:

$$O(n) + O(n) + O(2n \log n) = O(n \log n)$$

e na descompressão:

$$O(n) + O(n \log n) = O(n \log n)$$

.

4 Análise Experimental

Após realizar testes com entradas de tamanho variado, podemos perceber como o *overhead* causado por escverver as informações da árvore dentro do arquivo é detrimental em arquivos pequenos; A taxa de compressão para melhorar conforme aumentamos o tamanho do arquivo. Em outro caso, podemos verificar que quanto menor a variadade de caracteres, melhor e mais rápida a compressão pois desta forma conseguimos percorrer a árvore de huffman mais rápido e conseguimos escrever o mesmo texto com códigos menores.

5 Conclusões

Em conclusão, observamos que o algoritmo de Huffman é um método eficiente para compactar e codificar dados com base na frequência de ocorrência de símbolos. Ele cria códigos

de tamanho variável, onde símbolos mais frequentes são representados por códigos mais curtos, enquanto símbolos menos frequentes são representados por códigos mais longos.

6 Bibliografia

1. **CORMEN**. *Algoritmos: Teoria e Prática*. 3ª edição.

7 Instruções de Execução

Compile com make e execute com $./main - c$ entrada saída