Giovana Assis da Matta Machado
Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil
giovana2ma@gmail.com

1. Introdução

O trabalho explorou aspectos de manipulação de sequência e implementação de árvores de prefixo, por meio do algoritmo de compressão e descompressão LZ78. A execução do trabalho ajudou a fixar o conteúdo dado em sala de uma forma mais prática e a compreender como algoritmos de compressão funcionam.

2. Implementação

O programa foi desenvolvido na linguagem C, compilada pelo compilador G++ da GNU Compiler Collection.

3. Método

Para a implementação do trabalho, foi utilizado uma árvore trie de prefixos, essa foi a implementação escolhida, já que existia um interesse em reduzir o custo de busca e espaço, operações realizadas constantemente nesse algoritmo. Além disso, o código foi dividido em dois arquivos principais, um que armazena os prefixos na árvore, o outro arquivo caminha na árvore formando os prefixos armazenados, usado no momento de descompressão.

3.1 Árvore

3.1.0 Estrutura

Os nó presentes na árvore possuem o campo cadeia que guarda o caractere, código que guarda o código referente a aquele caractere, o número de filhos, uma lista de filhos com tipo nó e um ponteiro para o pai.

3.1.1 Cria Árvore

A função responsável por criar a árvore aloca na memória a raiz e a inicializa com string vazia e código zero.

3.1.2 Insere Árvore

A função recebe um ponteiro para o nó pai, onde será armazenado um novo nó filho, a posição em que esse filho será inserido, equivalente a quantidade de filhos do nó pai, o caractere a ser inserido bem como seu código.

A função então aloca mais uma posição no campo Filhos e guarda as informações do novo nó, por fim atualiza a quantidade de filhos do pai.

3.1.2 Busca Árvore

A função recebe o nó de partida da busca e o caractere a ser encontrado, então caminha sobre os filhos do nó de partida comparando o caractere, caso seja encontrado retorna o nó onde o caractere foi encontrado, caso contrário retorna um ponteiro nulo.

3.2 Palavra

3.2.1 Palavra

Função recebe o nó de partida e enquanto caminha na árvore guarda em um vetor, palavra, os caracteres encontrados, o caminhamento acaba quando o pai do nó atual é nulo, ou seja, chegou na raiz.

Ao final do caminhamento temos um vetor que guarda a palavra formada na ordem invertida, já que o caminhamento é feito de baixo para cima. Então ao final é chamado uma função para inverter a palavra. Por fim, a função retorna o tamanho da palavra formada.

3.2.2 Inverte

A função é responsável por inverter o vetor, ela guarda as variáveis de início e fim (tamanho da palavra menos um) enquanto o início não ultrapassa o fim ela realiza a troca dos caracteres.

3.3 Main

O arquivo principal recebe os parâmetros passados na linha de comando, como o arquivo de entrada, o arquivo de saída e qual função deve ser realizada, compressão ou descompressão.

3.3.1 Compressão

Para a compressão é criado uma árvore e a medida que os caracteres do arquivo são lidos busca o caractere na árvore, iniciando na raiz, caso o caractere seja encontrado atualiza o nó de partida e continua a leitura do arquivo.

Caso o caractere não seja encontrado na árvore é escrito no arquivo de saída o código do padrão formado antes da leitura do último caractere e qual foi o caractere que quebrou o padrão. Insere essa caractere na árvore como filho do último

caractere do padrão e incrementa o código, além de atualizar a busca para partir da raiz.

Por fim, verifica se o padrão formado com o último caractere está presente na árvore, caso não esteja é escrito o código do padrão encontrado e qual foi o caractere que quebrou esse padrão.

3.3.2 Descompressão

Para a descompressão é criada uma árvore, inicializada com a string vazia, para guardar os padrões encontrados e uma lista de nós. O arquivo possui a estrutura um índice e um caractere, o índice é referente ao código criado na compressão e o caractere que quebrou o padrão.

O caractere lido deve ser inserido como filho do nó referente ao índice lido, a lista de nós guarda qual nó foi inserido, assim o pai pode ser encontrado facilmente. Por fim é chamado a função Palavra e escrito no arquivo de saída a palavra formada.

4. Taxa de compressão

A taxa de compressão foi feito como 1 - comprimido/real

4.1 Exemplo 1 (O Guarany: romance brazileiro Vol. 01 (of 2))

Tamanho real: 380K
Tamanho comprimido: 336K
Taxa de compressão: 11,57%

4.2 Exemplo 2 (O Guarany: romance brazileiro Vol. 02 (of 2))

Tamanho real: 352K
Tamanho comprimido: 312K
Taxa de compressão: 11,36%

4.3 Exemplo 3 (The Blue Castle: a novel)

Tamanho real: 420K
Tamanho comprimido: 368K
Taxa de compressão: 12,38%

4.4 Exemplo 4 (The Adventures of Ferdinand Count Fathom)

Tamanho real: 972K
Tamanho comprimido: 764K
Taxa de compressão: 21,39%

4.5 Exemplo 5 (My Life Volume 1)

Tamanho real: 1.3M
Tamanho comprimido: 976K
Taxa de compressão: 24,92%

4.6 Exemplo 6 (Dracula)

Tamanho real: 864K
Tamanho comprimido: 700K
Taxa de compressão: 18,98%

4.7 Exemplo 7 (Four Plays of Gil Vicente)

Tamanho real: 436K
Tamanho comprimido: 412K
Taxa de compressão: 5,50%

4.8 Exemplo 8 (Os Lusíadas)

Tamanho real: 348K
Tamanho comprimido: 324K
Taxa de compressão: 6,89%

4.9 Exemplo 9 (CONSTITUIÇÃO DA REPÚBLICA FEDERATIVA DO BRASIL DE 1988)

Tamanho real: 644K
Tamanho comprimido: 436K
Taxa de compressão: 32,39%

4.10 Exemplo 10 (Dom Casmurro)

Tamanho real: 412K
Tamanho comprimido: 368K
Taxa de compressão: 10,67%

5. Conclusão

A premissa proposta pelo trabalho é de certa forma simples, manipular sequências e implementar estruturas já implementadas antes.

O conceito principal aprendido com o trabalho é a compressão e descompressão de arquivos, o trabalho foi uma boa forma de aprender de forma prática como esses algoritmos funcionam, visto que muitas vezes esses conceitos podem ser abstratos, por mais que utilizemos essas ferramentas diariamente.

6. Bibliografia

Para a realização desse trabalho foram usados os conteúdos disponibilizados sobre manipulação de sequência e árvores trie, além de conhecimentos adquiridos ao longo do curso. Para a construção do algoritmo LZ78 foi utilizado a bibliografia disponibilizada nas instruções do TP.

7. Instruções para compilação e execução

Para a execução do programa é necessário adicionar o arquivo de entrada na raiz do projeto, estar no diretório onde a pasta se encontra e usar o comando make, depois basta colocar "bin/tp1.out" na linha de comando e os parâmetros com o nome do arquivo de entrada e arquivo de saída (deve ser diferente do arquivo de entrada). A saída saíra em um arquivo na raiz do projeto.

8. Instruções para compilação e execução

Para a execução do programa é necessário adicionar o arquivo de entrada na raiz do projeto, estar no diretório onde a pasta se encontra e usar o comando make, depois basta colocar "bin/tp2.exe" na linha de comando e os parâmetros com o nome do arquivo de entrada, arquivo de saída (deve ser diferente do arquivo de entrada), valor da mediana e valor da partição. A saída saíra em um arquivo na raiz do projeto.