Trabalho Prático 2 – Compressão de Arquivos LZ78

Luiz H. da Silva Gonçalves[1]

[1] Universidade Federal de MInas Gerais (UFMG)
Belo Horizonte – MG – Brasil

luiz10hdsg@ufmg.br

1. Trabalho Prático: Compressão de arquivos com o Algoritmo LZ78

Resumo. *Este relatório tem como objetivo descrever o desenvolvimento do Algoritmo LZ78, usado para compressão e descompressão de arquivos.Dessa forma, essa documento tem o intuido de descrever com detalhes cada etapa de implementação do algoritmo e no final apresentar uma análise da execução do programa.*

2. Introdução

O objetico desse trabalho é abordar aspectos práticos de manipulação de sequências, usando para tal uma arvore de prefixo como estrutura de dados auxiliar.Sendo assim, o propósito do trabalho e implementar e analisar a execução do algoritmo LZ78 para comprimir e descomprimir arquivos. A ideia é desenvolver as abilidades técnicas decorrentes da executação desses algoritmos.

3. Algoritmo LZ78

O algoritmo LZ78 foi um dos algoritmos de compressão de dados desenvolvidos por Abraham Lempel e Jacob Ziv em 1978, juntamente com o outro algoritmo de compressão LZ77 publicado em 1977.O algoritmo LZ78 se baseia na construção de um dicionário com os dados encontrados anteriormente no arquivo a ser comprimido. No início o dicionário se encontra vazio. A medida que o arquivo vai sendo lido, caractere por caractere, cada sequência de caracteres não encontrada no dicionário é introduzida no dicionário e ganha um código. As sequências que já se encontram no dicionário são substituídas pelo seu código no dicionário.

4. Metodo

O método implementado utiliza uma arvore de prefixo como dicionario de dados brevemente encontrado que representa uma tabela de códigos usada na compressão e descompressão dos arquivos.O dicionário inicializa com a string nula na posição zero. Enquanto os símbolos são recebidos e codificados , as strings são adicionadas no dicionário nas posições 1,2 e assim sucessivamente.Para cada nó da arvore possui dois campos, um campo para indice que representa a posição na tabela e outro campo que representa uma sequencia de caracteres.Nos proximos topicos iremos apresentar o funcionamento de cada etapa do programa.

4.1. Compressão

O algoritmo LZ78 guarda os prefixos encontrados no texto em um dicionário representado pela arvore de prefixos que mapeia cada prefixo em um código que é chamado de índice.O processo de compressão fuciona da seguinte maneira:Adicionamos um par prefixo: "(vazio)" e índice: 0 no dicionário.Defimos o padrão P lido como "" (vazio) e um índice E como 0 e índice total T como 0. Proceguimos lendo o proximo caracter C1 do texto, caso a comcatenação P+C1 não esteja presente no dicionário, adicione um novo par prefixo:C1 índice: E no dicionário e incremente T em 1.Por outro lado se P+C1 estiver no dicionario, definimos P = P + C1 e guardamos em E o índice I referente ao par aonde foi achado a combinação P+L1, fazemos esse teste de comcatenação P+C1(se estar presente no dicionario)para cada caracter do texto ate o seu fim.

A trie implementada no programa atua como dicionario e seus nós associam o padrão que ele representa com o seu código e índice.Dessa forma, com a Trie é possível procurar por um padrão e adicionar novos a ela.Ao longo da criação da Trie, são guardados os indices que deverão ser lidos para a descompressão do arquivo.

4.2. Descompressão

Tendo o arquivo comprimido,pegamos o texto original com as tuplas gravadas.Após ler a forma binária que foram salvos na compressão e transformar em seus valores associados, vamos adicionando as tuplas, uma a uma de forma a recriar texto comprimido de acordo com as tuplas lidas.

5. Implementação

O programa foi desenvolvido na linguagem C++,compilada pelo compilador G++ da GNU Compilar Colletion e executado no sistema operacional Linux , na distribuição Ubuntu. A entrada dos dados é a entrada padrão do sistema(stdin).A saída também é a padrão do sistema(stdout).

5.1. Estrutura dos Arquivos

Para modularizar a implementação do sistema, foram montados sete arquivos distindos. No arquivo main.cpp contém os algoritmos referentes ao funcionamento do sistema de compactação e descompactação e os codigos que fazem leitura e escrita dos arquivos com as particularidades referente ao nome e extenção dos arquivos de saida e entrada.Nos arquivos TrieNode.h, Trie.h e CompressorLZ78.h foram feitos as declarações das estruturas de dados e variáveis na declaração da classe e os prototicos das funções principais.Nos arquivos TrieNode.h, Trie.h e CompressorLZ78.h contém os algoritmos referente a implementação da arvore de prefixo e tambem os codigos dos metodos de compressão e descompressão segundo o algoritmo LZ78.

6. Testes

Para os testes foram feitas 10 medições de compressões de livros.O intuido dos testes é medir a taxa de ganho em relação ao tamanho do arquivo comprimido.Para cada livro comprimido é feito uma comparação em relação ao tamanho do arquivo original em relação ao arquivo compactado.A taxa de compressão que vamos avaliar é tamanho do arquivo compactado dividido pelo tamanho do arquivo original.

NOME DO ARQUIVO	TAMANHO (bytes)	TAXA DE COMPRESSÃO
ABoysTown	tam original: 439.279 bytes tam compactado: 2.196.395 bytes	5
Baartock	tam original: 146.822 bytes tam compactado: 734.110 bytes	5
Clemence	tam original: 432.565 bytes tam compactado: 2.162.825 bytes	5
JourneyWork	tam original: 48.473 bytes tam compactado: 242.365 bytes	5
JustAGirl	tam original: 851.911 bytes tam compactado: 4.259.555 bytes	5
MenandWomen	tam original: 232.442 bytes tam compactado: 1.162.210 bytes	5
RedRidingHood	tam original: 25.743 bytes tam compactado: 128.715 bytes	5
TheAdventureofWisteriaLodge	tam original: 84.031 bytes tam compactado: 420.155 bytes	5
TheCaptives	tam original: 1.084.580 bytes tam compactado:5.422.900 bytes	5
ThePapersAndWritingsOfAbrahamLincoln	tam original: 229.631 bytes tam compactado: 1.148.155 bytes	5

Figura 1: Tabela de testes

Resultados Obtidos: Podemos observar que a taxa de compressão foi fixa para todos os arquivos teste.Dessa forma, é possivel tecer uma analise do comportamento do algoritmo em relação aos arquivos de entrada.Observamos a maioria dos livros comprimidos tem um tamanho superior a 1MB e sua taxa de compressão foi equivalente aos de tamanho inferior, mesmo sendo mais provavel que haja reocorrencia de sequências em arquivos maiores.Sendo assim, podemos apontar que a razão pelo aumento dos dados comprimidos é que pode acontecer que não há grande repetição de sequências, a medida que o algoritmo processa as sequencias recorrentes, seu processamento codifica a sequencia adicionando mais caracteres nos arquivos de forma codificada.Entretanto para os nossos testes indepedentemente do tamanho dos arquivos as informações inseridas no arquivo comprimido foram maiores do que o original.

7. Instruções de Execução

Para executação do codigo é preciso antes processar o arquivo Mikefire para geração do arquivo executavel presente nos arquivos entregaveis do projeto prático.Ao executar

o comando *make* o processo de compilação irá gerar um arquivo executavel chamado *tp1*, e é esse executavel que séra utilizado pra processar os métodos de descompressão e compressão em arquivos.

Para executar a compressão basta utilizar o comando ./tp1 -c *arquivo-de-entrada arquivo-de-saida* na mesma pasta onde se encontra o executável. Se o executavel não estiver na mesma pasta é preciso substituir no lugar de *arquivo-de-entrada* o caminho do arquivo de entrada que se deseja comprimir, e *arquivo-de-saida* para o caminho desejado do arquivo de saida.Se o nome para o arquivo de saıda não for fornecido, ele será igual ao do arquivo de entrada com a extensão ".z78".

Para executar a descompressão basta utilizar o comando ./tp1 -x *arquivo-de-entrada arquivo-de-saida*, seguindo a mesma regra em relação ao caminho relativo dos arquivos de entrada e saida.Se o nome para o arquivo de saıda não for fornecido, ele será igual ao do arquivo de entrada com a extensão ".txt".

8. Análise de Complexidade

Os dois processos implementados, compressão e descompressão possui complexidade *O(n)* em relação ao tamanho do arquivo de entrada.Na compressão, todo o algoritmo é executado dentro de um loop que itera sobre cada caractere do arquivo de entrada, para cada caracter lido é feito uma consulta na arvore de prefixo e escreve no arquivo de saida,com complexidade constante.De forma similar ocorre com a descompressão, a cada iteração do loop principal é lido a tupla de campos no arquivo de entrada e busca no dicionario, que tem tempo constante.

9. Conclusão

Certo de que o trabalho tinha por objetivo o desenvolvimento das capacidades técnicas em relação a implementação de um algoritmo de compressão de arquivos e aspectos práticos de manipulação de sequências, pode-se certificar que toda abordagem platica e técnica foi aplicada no desenvolvimentos dos algoritmos implementados, concluindo com êxito assim o objetivo do projeto prático.

10. Referências

https://pt.wikipedia.org/wiki/LZ78