UNIVERSIDADE FEDERAL
DE MINAS GERAIS

LZ78 usando Trie Comprimida

Álvaro Cândido de Oliveira Neto
djalv009@ufmg.br

[1]Departamento de Ciência da Computação - DCC - Universidade Federal de Minas Gerais

Resumo. *Este artigo tem como objetivo apresentar os resultados da execução do algoritmo LZ78 para compressão de arquivos .txt*

1. Introdução

O LZ78 foi proposto por Lempel e Ziv em 1978. A ideia do algoritmo é substituir strings que se repetem no texto por um código, diminuindo assim o número de bytes gravados na saída. O dicionário é inicializado com a string vazia associada ao código 0. Então, o texto é processado da esquerda para a direita, caracter a caracter. Uma string 'padrão' é mantida para verificar se essa substring já ocorreu anteriormente no texto. Se o padrão acrescido do próximo caracter lido ocorrer no dicionário, então mais um símbolo é lido e o ciclo se repete. Caso contrário, o par (códigoPadrão, caracter) deve ser emitido na saída, e a sequência padrão + caracter deve ser inserida no dicionário com um novo código. O ciclo se repete até que todo o arquivo tenha sido processado. A descompressão do arquivo segue o mesmo algoritmo. O dicionário é inicializado com a string vazia associada ao código 0. Sempre que um par (códigoPadrão, caracter) for lido, o padrão associado ao código deve ser emitido na saída, e um novo código deve ser associado à sequência padrão + caracter. O ciclo se repete até que todo o arquivo tenha sido processado.

2. Implementação

Para a implementação do algoritmo LZ78 foi utilizado uma Trie Comprimida ou Patricia.

Para buscar um palavra na Trie é percorrido os nós coletando os prefixo de tamanho maior que 0 e armazenando o nó em uma lista, logo é consumido o prefixo comum da palavra buscada e chamado recursivamente o metodo de busca. Ao fim da execução é retornado uma lista que contem o caminho da busca realizada.

Para inserir, é feita uma busca para retornar o caminho a ser percorrido, o ultimo nó do caminho é o nó em que vai receber o novo nó e se o prefixo desse nó for maior que 1 ele será dividido. Percorremos o caminho para descobrir o prefixo que já está sendo representado na Trie e então retiramos esse prefixo da palavra da entrada formando assim a nova palavra a ser inserida e então colocamos esse nó na Trie.

Foi criada tambem uma funçao que corrige os indices dos nós na Trie, basicamente um BFS para percorrer e corrigir os indices.

A função encode presente na Trie tambem faz o uso do BFS para caminhar sobre a arvore e escrever no arquivo de saída a codificação do LZ78

Para decodificar, é armazenado as tuplas em uma lista e assim formando palavras, também é armazendo os indices das palavras que foram codificados. Onde estiver presente um indice significa que uma palavra está completa, logo é feita uma associaçao das lista para descobrir todas as palavras e os indices, logo é escrito no arquivo de saída.

3. Experimentos e resultados

Considerando o sistema ASCII que utiliza 8 bits para representa uma letra e descartando as quebras de linha, virgulas, parenteses e espaços temos a seguinte quantidades de bits para os arquivos de teste:
dom casmurro de 2561352 bits para 1713216 bits, reduçao de 0.67

bootstrap de 32664 bits para 27144 bits, reduçao de 0.83

san andreas de 24384 bits para 21312 bits, reduçao de 0.87

superando desafios de 10016 bits para 9408 bits, reduçao de 0.94

conde monte cristo de 47688 bits para 37160 bits, reduçao de 0.78

opm hack de 141256 bits para 100504 bits, reduçao de 0.71

constituicao1988 de 3874856 para 1990256 bits, reduçao de 0.51

red dead2 de 49584 bits para 40880 bits, reduçao de 0.82

os lusiadas de 2138200 bits para 1443328 bits, reduçao de 0.67

github ddos de 41400 bits para 32264 bits, reduçao de 0.78
Para esses arquivos tivemos uma média de reduçao de 0.758

4. Arquivos de teste

Os arquivos de teste foram retirados dos seguintes conteudos:
O paradoxo de Bootstrap
Grand Theft Auto: San Andreas
A(s) genialidade(s) de Red Dead Redemption 2
GitHub Survived the Biggest DDoS Attack Ever Recorded
Inside the Cyberattack That Shocked the US Government
O Conde de Monte Cristo

5. Conclusão

O LZ78 de fato mostra alguma redução, mas como dito na introdução é um algoritmo de 1978, portanto existem algoritmos de compressão mais eficientes que ele. O grande problema do LZ78 é a espaço que a Trie ocupa, como visto em sala de aula, Trie são muito eficientes no quesito de tempo mas em espaço ela é muito ineficiente

Referências

[1] Renato Vimieiro - Slides de aula
[2] How Data Compression Works: Exploring LZ78
[3] LZ78
[4] 18. LZ78 encoding and decoding with examples.