Algoritmo LZ78 Utilizando Trie

Trabalho Prático 1 - Algoritmos 2 - UFMG

João Luiz Lopes Megale
2020006671

1. Introdução

O trabalho proposto foi a implementação de um programa de compressão e descompressão de arquivos, por meio do algoritmo LZ78, desenvolvido por Abraham Lempel e Jacob Ziv em 1978. Na implementação, foi utilizada uma Trie como estrutura de dados.

2. Estrutura de Dados

Como dito anteriormente, a estrutura de dados utilizada foi uma Trie, pois como o algoritmo é baseado em dicionários e realiza muitas buscas e inserções neste dicionário, esta estrutura facilita bastante nas buscas, deixando o programa mais eficiente. A Trie é composta por duas classes, Node e Trie.

2.1 Classe Node

Representa um nó da árvore; contém o caractere associado ao nó, seu índice, o índice de seu pai e um dicionário, inicialmente vazio, que representa seus filhos. Para o construtor, são passados o índice e o caractere do nó.

2.2 Classe Trie:

Representa a árvore em si; contém um nó raiz, que possui o índice 0, e texto vazio. Além da raiz, possui uma variável referente ao número de nós presente na árvore, que inicialmente será 1, ou seja, apenas a raiz. Esta classe também possui duas funções:

- find: recebe um texto, verifica se ele está totalmente contido na árvore e retorna um valor booleano.
- insert: recebe um texto e, caso ele não esteja contido na árvore, insere o nó necessário na estrutura, retornando este nó que foi adicionado.

Compressão

A compressão segue o algoritmo LZ78, utilizando a trie como um dicionário para armazenar as strings lidas. O programa inicializa uma trie vazia e lê o texto do arquivo a ser comprimido, caractere por caractere, inserindo na trie os caracteres que ainda não estão presentes na árvore. Caso um caractere for encontrado na estrutura, ele é concatenado com o próximo no arquivo, formando uma string, e em seguida essa string resultante é novamente procurada na Trie. Caracteres novos vão sendo adicionados à string, até que forme uma palavra que não se encontra na Trie. Quando isso ocorre, um novo nó contendo o último caractere é adicionado à árvore, e no arquivo de saída é escrito o novo caractere adicionado, juntamente com

o índice do pai desse novo nó. No caso, o índice do pai do nó na árvore é análogo ao índice do dicionário do algoritmo LZ78. A busca por texto na Trie e a inserção de nós são feitas por meio das funções find e insert, introduzidas na seção 2.2 desta documentação. Os dados no arquivo de saída são escritos em bytes, para que haja de fato uma redução no tamanho do arquivo. Um inteiro será representado por 3 bytes, enquanto uma caractere será representado por 2 bytes.

Descompressão

Para realizar a descompressão, o programa primeiramente lê os bytes do arquivo de saída, e os transforma novamente em caracteres e inteiros legíveis. Para ordenar os caracteres lidos de forma a resultar no texto original, é utilizado um vetor para armazenar as palavras lidas do arquivo. Nesse vetor, primeiramente são adicionados os caracteres que possuem índice 0, e à medida em que vão aparecendo caracteres com índices maiores, estes caracteres são concatenados com o texto presente na posição referente ao seu índice no vetor de armazenamento de texto. A cada iteração, o texto adicionado ao vetor de armazenamento é concatenado em uma variável, que será o texto final a ser escrito no arquivo de saída.

Casos de Teste

Para testar a eficácia do algoritmo em comprimir arquivos, foram comprimidos 10 arquivos de texto. Segue o tamanho original, o tamanho comprimido e a taxa de compressão para cada um deles:

nome	**tamanho original**	**tamanho comprimido**	**taxa de compressão**
dom_casmurro.txt	401 KB	358 KB	11%
os_lusiadas.txt	337 KB	314 KB	7%
constituicao1988.txt	637 KB	427 KB	33%
esau_e_jaco.txt	420 KB	362 KB	14%
plano_educacao.txt	83 KB	82 KB	1%
casa_velha.txt	306 KB	279 KB	9%
mao_luva.txt	228 KB	212 KB	7%
moby_dick.txt	1247 KB	992 KB	21%
romeo_juliet.txt	166 KB	161 KB	3%
frankenstein.txt	439 KB	378 KB	14%

Conclusão

Neste trabalho, foi implementado um algoritmo de compressão de arquivos, utilizando uma Trie como estrutura de dados. A atividade foi importante para fixar o conteúdo visto em sala sobre manipulação de sequências, além de servir como oportunidade de conhecer problemas que não são abordados em sala de aula, como a compressão de arquivos. A principal dificuldade do trabalho foi entender como adaptar o funcionamento do algoritmo LZ78 à estrutura de dados Trie. A utilização da estrutura

Quanto à eficiência do algoritmo de compressão, os casos de teste obtiveram uma taxa de compressão em torno de 10%. No pior caso, o arquivo praticamente continuou com o mesmo tamanho, diminuindo em apenas 1%, justamente no teste de menor tamanho. No melhor caso, o arquivo comprimido ficou cerca de um terço menor que o arquivo original.

Referências

- slides da aula
- https://pt.wikipedia.org/wiki/LZ78
- https://en.wikipedia.org/wiki/LZ77_and_LZ78