Trabalho Prático I

LZ98

João Victor Taufner Pereira
2017098315

1. Implementação

O trabalho foi desenvolvido utilizando a versão 3.9.6 da linguagem Python.

1.1. Estrutura de Dados

Para a implementação do algoritmo LZ98, foi utilizada como estrutura de dados a árvore de prefixos, implementada na classe Node que representa cada vértice da árvore. A classe Node armazena um valor que representa se o nó é ou não terminal e seus filhos, que são inicialmente não inicializados. A Trie é inicializada com apenas um nó raiz que representa o caractere vazio. Para buscar uma cadeia, são percorridos os nós associados a cada caractere da cadeia até que a cadeia seja totalmente consumida. Para a inserção de uma cadeia, é feita uma busca e o valor do nó encontrado é atualizado apenas se o nó não for terminal de alguma cadeia já inserida.

1.2. Textos escolhidos

Os 10 textos escolhidos para o teste do algoritmo foram retirados do site que se encontra nas referências deste documento. Para a resolução do trabalho, foram considerados apenas caracteres que são representados por apenas um byte. Dessa forma, os textos utilizados de input precisaram de passar por um tratamento em que os caracteres que não respeitassem essa restrição fossem eliminados. Para isso, foi utilizada uma extensão da IDE em que trabalhei, VSCode.

1.3. Compressão

a compressão utiliza a Trie como um dicionário para armazenar o texto lido da entrada. Uma cadeia vazia é considerada inicialmente e, se o dicionário contém a cadeia, ela é concatenada com o último caractere lido. Caso contrário, o código associado ao caractere é o valor associado à cadeia no dicionário, e são escritos na saída o tamanho do código, o código e o caractere, todos em binário. Em seguida, a cadeia

concatenada ao caractere é adicionada ao dicionário e a cadeia é resetada para vazia. Ao consumir todos os caracteres da entrada, são escritos o tamanho do código restante, o código em si e uma cadeia vazia. Dessa forma, cada valor em um nó terminal do dicionário corresponderá ao código do caractere que ele está representando.

1.4. Descompressão

Para executar a descompressão, inicialmente é inicializado um array de cadeias contendo apenas a cadeia vazia. A partir disso, até que toda a entrada seja consumida, executa-se uma simples sequência de passos. Primeiro, são lidos o tamanho do código em bytes, o código e o caractere. Em seguida, é escrita na saída uma cadeia igual à salva no índice do código lido concatenada ao caractere. Por fim, essa cadeia é adicionada ao fim do array. Quando toda a entrada for consumida, o processo de descompressão terá sido terminado.

2. Testes

Os resultados do algoritmo de compressão encontram-se dispostos na tabela abaixo, que contém o tamanho original do arquivo, o tamanho do arquivo comprimido e a taxa de compressão, que nos indica percentualmente o tamanho do arquivo comprimido comparado ao arquivo original.

Entrada	Tamanho Original em KB	Tamanho comprimido em KB	Taxa de compressão
A Tale of Two Cities, by Charles Dickens	768	540	70,31%
Adventures of Huckleberry Finn, by Mark Twain	579	407	70,29%
Crime and Punishment, By Fyodor Dostoevsky	1.138	775	68,10%
Metamorphosis, by Franz Kafka	137	104	75,91%
Moby-Dick; or The Whale, by Herman Melville	1.232	885	71,83%
Persuasion, by Jane Austen	479	334	69,72%
Romeo and Juliet, by William Shakespeare	163	126	77,30%
Sense and Sensibility, by Jane Austen	682	465	68,18%
The Republic, by Plato	214	157	73,36%
Thus Spake Zarathustra, by Friedrich Niezche	653	467	71,51%

3. Referências

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

-Top 100 | Project Gutenberg

- David Salomon. Data Compression: The complete reference. 4th edition. Springer