Algoritmos 2
Trabalho Prático 1

Lucas Perez Santos
Matrícula: 2019041566

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

perezlucas2903@gmail.com

1. Introdução

O trabalho consistiu na implementação do método de compressão LZ78, utilizando uma árvore de prefixos (trie) como dicionário. Foi utilizada a linguagem Python para a realização desse trabalho.

2. Implementação

A entrega consiste em dois arquivos, main.py e LZ78.py, e em 10 casos de teste. Além disso, foi utilizada a biblioteca sys para realizar a leitura do arquivo de texto pelo terminal.

2.1. LZ78.py

O arquivo possui três classes, Node, LZ78-compression e LZ78-decompression.
A classe Node cria o nó da árvore de prefixos. Cada nó possui um identificador único, um valor data que equivale ao padrão até aquele nó e um dicionário python que contém seus filhos, onde a chave é um caracter e o valor associado é um Node.

Já a classe LZ78-compression recebe um arquivo .txt com seu respectivo nome e comprime o texto utilizando o encoded-text. Nesse método, ocorre a iteração caracter por caracter no texto e, durante a execução, é montada a trie. Começamos por um nó vazio na raiz. Caso o caracter lido não seja filho do nó atual, escrevemos a tupla (identificador do pai, caracter atual) em binário em um arquivo de mesmo nome .z78. Caso ele seja filho, então tornamos o nó atual como o filho que possuir esse catacter como chave no dicionário do nó e continuamos a execução. Desse modo, caso o caracter atual esteja presente como chave do dicionário do nó atual, sabemos que essa string formada no momento já ocorreu antes e estamos lendo um padrão. Caso o nó não seja filho, chegamos no final de um padrão novo, que consiste de um padrão já obtido, acrescido do caracter atual.

Por fim, a classe LZ78-decompression recebe um arquivo .z78 e o nome desse arquivo e, a partir disso, realiza a descompressão do texto por meio do método decode-text. Nesse método, iteramos até que o arquivo termine e, durante a iteração, ocorre a descompressão do texto. Foram alocados 2 bytes para o identificador e 1 byte para o caracter ao fazer a escrita do .z78 em binário. Desse modo, para recuperar a informação, basta realizar a leitura dos pares de 2 bytes e 1 bytes e converter de volta do binário, obtendo a tupla (identificador do pai, caracter atual). A trie da descompressão é feita

por meio de uma lista, denominada decoder, inicializada com uma string vazia. Ao ler uma tupla, reobtemos o padrão ao concatenar decoder[identificador do pai] e o caracter lido. Como a ordem de leitura mantém a ordem dos identificadores, podemos apenas dar append no padrão e prosseguir com a leitura. Caso seja um padrão novo, ocorre a concatenação de um prefixo vazio com o caracter.

2.2. main.py

No arquivo main.py é feita a leitura do arquivo pelo terminal, utilizando a biblioteca sys. Além disso, ele verifica a extensão do arquivo e decide quais métodos chamar. Caso seja dado um arquivo .txt, ele chama os métodos de compressão e caso seja dado um .z78, os de descompressão.

3. Execução do programa

Para executar o programa, basta utilizar o comando no terminal:

python main.py -c <arquivo>

onde o arquivo deve ter extensão .txt ou .z78 para a compressão e descompressão, respectivamente.

4. Análise da taxa de compressão

Convencionamos a taxa de compressão como a razão entre o tamanho do arquivo .txt e o arquivo .z78. Assim, dentre os 10 casos de teste criados, temos os seguintes dados:

Caso de teste	Tamanho .txt (em kb)	Tamanho .z78 (em kb)	Taxa de compressão
test01	2.11	2.13	0.99
test02	4.43	4.06	1.09
test03	5.43	4.97	1.09
test04	6.79	4.5	1.51
test05	10.6	7.9	1.34
test06	14.3	10	1.43
test07	21.3	13.5	1.58
test08	71.2	35	2.03
test09	365	144	2.53
test10	466	184	2.53

Como é possível evidenciar, a taxa de compressão aumenta à medida em que o tamanho do .txt aumenta. Esse resultado já era esperado, uma vez que, quanto maior o texto, maior a probabilidade de padrões ocorrerem.

5. Análise de Complexidade

Seja n a quantidade de caracteres no texto e m a quantidade de tuplas no texto codificado em binário. Como o método de codificação itera sobre os caracteres, ele possui complexidade de tempo $O(n)$. De forma análoga, o método de decodificação itera sobre as tuplas codificadas em binário, possuindo assim complexidade $O(m)$. Assim, a complexidade do código será dominada pelo maior entre m e n.

6. Conclusão

Nesse trabalho, foi feita com sucesso a compressão dos arquivos por meio do método LZ78 utilizando uma trie. Através desse processo, acredita-se que os objetivos do trabalho de promover a exploração desse método pelo aluno e promover a fixação do conteúdo estudado foi alcançado com sucesso.

7. Bibliografia

Página da wikipedia sobre LZ78: https://pt.wikipedia.org/wiki/LZ78
Slides da aula