TP1 – Manipulação de sequências

Roger Dornas Oliveira – 2021032196

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

1. Introdução

O objetivo desse trabalho é implementar um programa que comprime e descomprime arquivos de texto. O algoritmo utilizado foi o método LZ78, que basicamente substitui strings que se repetem no texto por códigos. O arquivo gerado pela compressão é um arquivo binário que contém os códigos gerados e algumas outras informações, que serão discutidas a seguir, para que seja possível a descompressão e obter uma cópia do arquivo original.

2. Implementação

O programa foi desenvolvido em Python, utilizando as bibliotecas e módulos *numpy*, *math* e sys. O método LZ78 funciona da seguinte forma: o texto a ser comprimido vai sendo lido caractere a caractere; sempre que um caractere novo é lido, ele é adicionado a uma árvore trie, juntamente com seu índice, que é a posição em que ele foi adicionado; se um caractere lido já pertence à árvore, esse caractere é concatenado com o próximo, e testa novamente se essa nova string está presente na árvore; se não está na árvore, ela é adicionada, juntamente com o índice; se já está na árvore, repete o processo de concatenar com o próximo caractere e continua testando. Simultaneamente com a construção da árvore, são gerados os códigos que vão para o arquivo compactado; esse código segue o seguinte formato: primeiro temos o indice da palavra na árvore que é prefixo da palavra que ele representa, seguido por um caractere que completa a string representada pelo código. Depois de ler todo o arquivo de entrada, teremos todos os códigos para gerar o arquivo compactado; esses códigos serão gravados em um arquivo binário, e para isso, é preciso definir quantos bytes serão necessários para armazenar os códigos e os caracteres. Para isso, pegamos o maior valor do código, tira seu log na base 2 e divide esse valor por 8; em seguida arredondamos esse resultado sempre para cima, e esse será o número de bytes usado para gravar cada código. O mesmo é feito para os caracteres. O número de bytes necessários para os códigos e para os caracteres é gravado no início do arquivo comprimido. Em seguida basta gravar todos os códigos. Essa foi a compressão! Na descompressão, lemos o inicio do arquivo para saber quantos bytes são usados em cada código. Criamos um dicionário vazio, e lemos cada par de código e caractere; com esse par, através do código pegamos o prefixo presente no dicionário, concatenamos com o caractere, atualizamos o dicionário com o par lido, e gravamos na saída a string resultante. Isso é repetido até o arquivo de entrada terminar. Ao fim, teremos uma cópia do arquivo original, obtido através dos códigos que representam o arquivo.

Para rodar o programa para comprimir um arquivo, basta digitar o seguinte comando:
python3 <programa> -c <arquivo_entrada> -o <arquivo_saida>
Para rodar o programa para descomprimir um arquivo, basta digitar o seguinte comando:
python3 <programa> -x <arquivo_entrada> -o <arquivo_saida>
Os parâmetros *-o* e *<arquivo_saida>* são opcionais, e caso não sejam dados, o nome do arquivo de saída será o mesmo do de entrada, mas com a devida substituição na extensão.

3. Exemplos

A seguir temos 10 exemplos de arquivos de texto que serão comprimidos pelo programa. A partir de cada exemplo, vamos analisar o tamanho original do arquivo, o tamanho do arquivo comprimido gerado, e a taxa de compressão. A taxa de compressão é igual a 1 menos a razão entre o tamanho final do arquivo comprimida e o tamanho do arquivo sem compressão; assim, quanto maior é essa taxa, mais comprimido foi o arquivo. Com esses 10 exemplos, poderemos estimar uma taxa de compressão média e alguma relação entre a entrada e a saída. Todos os 10 arquivos de entrada usados podem ser acessados nesse repositório.

O primeiro exemplo é o arquivo *constituicao1988.txt*. Seu tamanho é de 636 kB, e seu arquivo comprimido possui tamanho de 426 kB, e a taxa de compressão foi de $1 - \frac{426}{636} = 0{,}33$, ou seja, o tamanho foi reduzido em 33%.

O segundo exemplo é o arquivo *dom_casmurro.txt*. Seu tamanho é de 400 kB, e seu arquivo comprimido possui tamanho de 285 kB, e a taxa de compressão foi de $1 - \frac{285}{400} = 0{,}29$, ou seja, o tamanho foi reduzido em 29%.

O terceiro exemplo é o arquivo *os_lusiadas.txt*. Seu tamanho é de 336 kB, e seu arquivo comprimido possui tamanho de 188 kB, e a taxa de compressão foi de $1 - \frac{188}{336} = 0{,}44$, ou seja, o tamanho foi reduzido em 44%.

Nos próximos exemplos foram repetidos os mesmos passos, gerando a seguinte tabela:

Nome do arquivo original	Tamanho do arquivo original (kB)	Tamanho do arquivo comprimido (kB)	Taxa de conversão
constituicao1988.txt	636	426	$1 - \frac{426}{636} = 0{,}33$
dom_casmurro.txt	400	285	$1 - \frac{285}{400} = 0{,}29$
os_lusiadas.txt	336	188	$1 - \frac{188}{336} = 0{,}44$
o_pequeno_principe.txt	79,4	63,7	$1 - \frac{63{,}7}{79{,}4} = 0{,}20$
tp1.txt	6,39	5,71	$1 - \frac{5{,}71}{6{,}39} = 0{,}11$
lista_palavras.txt	434	173	$1 - \frac{173}{434} = 0{,}60$
a_divina_comedia.txt	753	607	$1 - \frac{607}{753} = 0{,}19$
richard_iii.txt	214	157	$1 - \frac{157}{214} = 0{,}27$
the_merchant_of_venice.txt	117	68,4	$1 - \frac{68{,}4}{117} = 0{,}42$
hamlet.txt	216	163	$1 - \frac{163}{216} = 0{,}25$

Pela tabela, podemos perceber que em arquivos pequenos, a taxa de conversão é menor, o que faz sentido, já que no arquivo possuem menos strings compartilhando o mesmo prefixo. Em um exemplo em particular, o arquivo *lista_palavras.txt*, que é uma lista de palavras em ordem alfabética todas começando com a letra *a*, possui uma taxa de conversão alta, o que também faz sentido, já que muitos prefixos são compartilhados. A taxa de conversão média desses 10 exemplos é de 0,31, mas como já dito, o tamanho e tipo do arquivo influencia bastante na taxa de compressão.