TP1 - Relatório

Nome: Rodrigo Ferreira Araújo — Matrícula: 2020006990

Algoritmos 2 - Junho 2022

1 Introdução

Esse trabalho prático, implementado em Python, visa aplicar de forma simples e direta o algoritmo de compressão de arquivos de dados (arquivos de texto nesse contexto) chamado LZ78. Idealizado por Abraham Lempel e Jacob Ziv em 1978, o algoritmo visa criar um dicionário que mapeia códigos a caracteres do texto original, de modo que cada código indica a ocorrência de uma nova substring do texto lido. Dessa maneira, por meio desse dicionário, conseguimos comprimir o arquivo aproveitando substrings que se repetem no texto e armazenando-as em códigos sucessivos no dicionário que as representa.

A leitura é feita caracter a caracter no arquivo original e, dinamicamente, verificamos se a substring de leitura acumulada até o momento encontra-se representada no dicionário: se sim, continuamos lendo, senão, registramos o último caracter lido no dicionário e associamos um código a ele que representa a nova substring. Nesse sentido, esse trabalho prático implementa a lógica explicada acima para comprimir arquivos de texto para arquivos ".z78". Somado a isso, também é implementado a descompressão do arquivo .z78 de volta para o .txt original.

2 Execução

Comprimir um arquivo .txt para .z78: python3 main.py -c arquivo.txt

Descomprimir um arquivo .z78 para o .txt: python3 main.py -x arquivo.z78

Nos arquivos de entrega no Teams, será fornecido um link de google drive (apenas quem possui o link tem acesso) com os arquivos .txt de teste. Ao executar a compressão, um arquivo .z78 deve ser gerado no mesmo diretório, e estará pronto para a descompressão.

3 Compressão

3.1 Lógica

O arquivo .txt original é lido caracter a caracter e, simultâneo a leitura, o caracter atual é procurado na árvore Trie. A busca em questão é simples: a partir de um nó pai (raiz da árvore inicialmente), vá para o nó filho cujo caracter armazenado corresponde ao procurado. Nesse sentido, enquanto essa busca for bem sucedida, continuamos a ler novos caracteres do texto e buscando-os na Trie. Quando a busca falhar, significa que encontramos uma substring não registrada do texto original, portanto criamos um novo nó na Trie, que será filho do nó onde a busca falhou e conterá o último caracter lido e um código que representa essa nova substring. Por último, escrevemos a dupla ("código,caracter") no arquivo comprimido e retornamos o nó atual de busca para o nó raiz, para possibilitar a busca adequada de uma nova substring.

O processo descrito segue até que todos os caracteres do texto original forem lidos. Ao final do processo de compressão, o programa reporta (imprimindo na saída padrão) informações relevantes sobre a compressão. Em suma, são reportados os tamanhos dos arquivos original e comprimido em bits e a taxa de compressão. Durante o processamento descrito, o tamanho do arquivo original em bits é calculado contando os caracteres enquanto são lidos multiplicando o resultado por 8, uma vez que cada caracter ocupa 1 byte (8 bits) em um arquivo de texto padrão.

O tamanho do arquivo comprimido é calculado de forma diferente. Desconsiderando a vírgula "," que separa as duplas ("código,caracter") escritas no .z78, cada caracter das duplas consome 8 bits de espaço e assumimos uma quantidade fixa de bits para representar os códigos inteiros das duplas. Essa quantidade fixa é determinada pelo menor número de bits necessário para representar o maior código do dicionário produzido, com o intuito de padronizar

e minimizar os bits que representam os códigos. Esse maior código também é encontrado durante o processamento principal sem adicionar complexidade ao algoritmo.

Digamos que o maior código encontrado no dicionário produzido seja 127 = 1111111, que necessita de 7 bits, e o dicionário possua 150 duplas registradas no .z78. Portanto, o tamanho do arquivo comprimido será 150 * (8 bits do caracter + 7 bits do código) = 2250 bits. Por fim a taxa de compressão é calculada como 1 subtraído da razão dos tamanhos dos arquivos (comprimido / original), ou seja, caso comprimido = 30 bits e original 100 bits, o arquivo comprimido representa 30% do tamanho do original, logo, temos uma compressão de 1 - 30% = 70%.

Note, portanto, que para fins simplificar a implementação, não estamos comprimindo de fato o arquivo, pois estamos escrevendo uma representação do dicionário produzido no arquivo .z78 com caracteres inteiros de 1 byte, sem escrita de binários. Contudo, o cálculo fornecido acima produz resultados consistentes ao comportamento esperado do algoritmo.

3.2 Testes

3.2.1 teste0.txt

Constitui o teste simples fornecido de exemplo na especificação do TP: "A-ASA-DA-CASA".

Tamanho do arquivo original em bits: 104
Tamanho do arquivo comprimido em bits: 77
Taxa de compressão: 0.2596153846153846

3.2.2 teste1.txt

Constitui o próprio código fonte em formato .txt.

Tamanho do arquivo original em bits: 66936
Tamanho do arquivo comprimido em bits: 42400
Taxa de compressão: 0.36655910123102664

3.2.3 teste2.txt

Um trecho de uma página de palavras com caracteres aleatórios. Trecho retirado do site Babel Library.

Tamanho do arquivo original em bits: 25960
Tamanho do arquivo comprimido em bits: 25175
Taxa de compressão: 0.030238828967642517

Note a baixa taxa de compressão, resultado da aleatoriedade das palavras do texto, o que configura pouquíssimo aproveitamento de substrings.

3.2.4 teste3.txt

Arquivo formado com um gerador de palavras em latim com o famoso padrão "lorem Ipsum".

Tamanho do arquivo original em bits: 25688
Tamanho do arquivo comprimido em bits: 18450
Taxa de compressão: 0.28176580504515725

3.2.5 teste4.txt

Primeiros versículos do livro de Gênesis da Bíblia em inglês.

Tamanho do arquivo original em bits: 33600
Tamanho do arquivo comprimido em bits: 21204
Taxa de compressão: 0.3689285714285714

3.2.6 teste5.txt

Arquivo constituído por todo os conteúdos textuais da página da Wikipedia fornecida na especificação deste TP.

Tamanho do arquivo original em bits: 54704
Tamanho do arquivo comprimido em bits: 38532
Taxa de compressão: 0.29562737642585546

3.2.7 teste6.txt

Arquivo constituído pelo código fonte da página da Wikipedia oficial da Segunda Guerra Mundial em inglês traduzido para .txt.

Tamanho do arquivo original em bits: 1973600
Tamanho do arquivo comprimido em bits: 1112448
Taxa de compressão: 0.43633563032022704

Note um crescimento na taxa de compressão, que pode ser explicado pelos seguintes fatores: semelhança entre palavras em inglês e, pela natureza do algoritmo e da estrutura de dados usada, a taxa de compressão cresce a medida que o tamanho do arquivo original cresce.

3.2.8 teste7.txt

Livro de Machado de Assis, Memórias Póstumas de Brás Cubas, convertido para txt.

Tamanho do arquivo original em bits: 4314544
Tamanho do arquivo comprimido em bits: 1922875
Taxa de compressão: 0.5543271780285471

3.2.9 teste8.txt

Arquivo txt com o roteiro transcrito (em inglês) do filme Harry Potter e a Pedra Filosofal.

Tamanho do arquivo original em bits: 958576
Tamanho do arquivo comprimido em bits: 562833
Taxa de compressão: 0.41284467793894275

3.2.10 teste9.txt

Arquivo constituído por todos os conteúdos textuais do PDF de notas de aula da Disciplina de Pesquisa Operacional.

Tamanho do arquivo original em bits: 3024320

Tamanho do arquivo comprimido em bits: 1147632
Taxa de compressão: 0.620532218812824

3.2.11 teste10.txt

Arquivo constituído por todos os conteúdos textuais do PDF de notas de aula da Disciplina de Cálculo.

Tamanho do arquivo original em bits: 15394928
Tamanho do arquivo comprimido em bits: 3419546
Taxa de compressão: 0.7778784025492032

3.2.12 teste11.txt

Arquivo constituído caracteres aleatórios e contíguos.

Tamanho do arquivo original em bits: 8388608
Tamanho do arquivo comprimido em bits: 8281332
Taxa de compressão: 0.01278829574584961

3.2.13 teste12.txt

Arquivo constituído caracteres aleatórios e contíguos com aproximadamente o dobro de tamanho do anterior.

Tamanho do arquivo original em bits: 16777216
Tamanho do arquivo comprimido em bits: 16216620
Taxa de compressão: 0.03341412544250488

Note que, apesar do crescimento da taxa de compressão devido ao aumento do tamanho do arquivo, sua natureza completamente aleatória mantém essa taxa baixíssima.

3.2.14 teste13.txt

Outro arquivo formado com um gerador de palavras em latim com o famoso padrão "lorem Ipsum".

Tamanho do arquivo original em bits: 6051448

Tamanho do arquivo comprimido em bits: 1914575
Taxa de compressão: 0.6836170450444257

4 Descompressão

A lógica de descompressão do arquivo usa apenas o dicionário produzido no arquivo .z78 para escrever as substrings do arquivo original. Os pares (código, caracter) são lidos um a um do arquivo de texto comprimido e armazenados em separado. Desse modo, para um par código e carcater lido, acumulamos à esquerda o caracter atual em uma string "X" e usamos o código para navegar para uma posição anterior (correspondente ao código lido) do dicionário armazenado até então. Repetimos esse processo até atingir o início do dicionário e, por fim, escrevemos a string acumulada "X" no arquivo .txt original e "X" é limpa e pronta para receber uma nova substring. Repetimos toda essa lógica até que as duplas (código, caracter) se esgotem no arquivo comprimido.