# Relatório TP1 – Algoritmos 2

Henrique Daniel de Sousa

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais

UFMG

# 1. Introdução

Esse relatório tem o intuito de explicar essa implementação do algoritmo de compressão LZ78 em Python, permite comprimir um arquivo de texto em um arquivo menor com base em padrões que aparecem no texto. O algoritmo é baseado em um dicionário que armazena as substrings já encontradas e seus respectivos índices.

Nessa implementação, usa-se uma trie no algoritmo. Desse modo, como a inserção nos dicionários é uma operação realizada frequentemente em tal aplicação, a trie é uma estrutura que aumenta a eficácia do algoritmo.

# 2. Implementação

O código em questão foi desenvolvido em Pyhton 3.10.6, com o WSL2 do Windows 10. Foram utilizadas as bibliotecas sys e numpy como suporte para o desenvolvimento.

# 3. Modelagem

A implementação usa uma classe TrieNode para representar a árvore de prefixos (trie) usada no algoritmo. Essa classe tem um atributo children, que é um dicionário que mapeia cada caractere para um objeto TrieNode que representa o nó filho correspondente, e um atributo index que é o índice do prefixo representado por esse nó.

A função tuple_to_bin converte uma tupla (índice, caractere) em uma string binária. A string binária é composta pelo índice e pelo caractere codificados em binário, cada um ocupando um número fixo de bits especificado pelos parâmetros index_width e char_size.

A função encoded_file grava uma string de bits em um arquivo binário, onde cada byte é representado por 8 bits.

A função lz78_encode recebe uma string de entrada e retorna uma lista de tuplas (índice, caractere), representando a sequência comprimida. Ele também retorna o número mínimo de bits necessários para representar cada índice e cada caractere.

A função lz78_decode recebe uma lista de tuplas (índice, caractere) e retorna a string original.

Finalmente, o código principal verifica os argumentos da linha de comando para determinar se o programa deve ser usado para codificar ou decodificar um arquivo de texto. Se a opção de codificação (-c) for escolhida, o programa lê o arquivo de entrada, comprime-o usando lz78_encode e grava o resultado em um arquivo de saída binário. Se a opção de decodificação (-x) for escolhida, o programa lê o arquivo binário de entrada, decodifica-o usando lz78_decode e grava o resultado em um arquivo de texto.

Em resumo, o código implementa o algoritmo de compressão LZ78 e permite a compressão e descompressão de arquivos de texto usando esse algoritmo.

# 4. Resultados

Os seguintes resultados foram obtidos através do teste de 10 arquivos de texto, obtidos pelo Projeto Gutenberg.

| Arquivo de entrada | Tamanho original | Tamanho do arquivo de saída | Taxa de compressão |
|---|---|---|---|
| **karamazov.txt** | 2014789 bytes | 1412597 bytes | 1.42 |
| **leviathan.txt** | 413366 bytes | 348302 bytes | 1.42 |
| **metamorphosis.txt** | 139182 bytes | 105486 bytes | 1.31 |
| **my_life.txt** | 1277141 bytes | 957857 bytes | 1.33 |
| **notes_from_ underground.txt** | 262705 bytes | 191090 bytes | 1.37 |
| **quincas_borba.txt** | 485688 bytes | 414092 bytes | 1.17 |
| **the_great_gatsby.txt** | 299891 bytes | 213082 bytes | 1.40 |
| **the_odyssey.txt** | 708411 bytes | 562442 bytes | 1.25 |
| **the_republic.txt** | 1228326 bytes | 893057 bytes | 1.37 |
| **zarathustra.txt** | 476786 bytes | 404402 bytes | 1.17 |

A taxa de compressão é a razão entre a quantidade de bytes do arquivo original pelo arquivo de saída. Essa razão demonstra quantos bytes do arquivo original são codificados em 1 byte do arquivo comprimido. Em média, o algoritmo apresentou uma taxa de compressão de 1.3.