UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO YUJII SILVA KADOOKA

EXPLORANDO ALGORITMOS DE COMPRESSÃO DE DADOS: TEORIA, IMPLEMENTAÇÃO E DESEMPENHO

BAURU
Novembro/2025

GUSTAVO YUJII SILVA KADOOKA

EXPLORANDO ALGORITMOS DE COMPRESSÃO DE DADOS: TEORIA, IMPLEMENTAÇÃO E DESEMPENHO

Trabalho de Conclusão de Curso em Ciência da Computação apresentado ao Departamento de Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Profa. Dra. Andréa Carla Gonçalves Vianna

BAURU
Novembro/2025

K11e

Kadooka, Gustavo Yujii Silva
Explorando Algoritmos de Compressão de Dados: Teoria, Implementação e Desempenho / Gustavo Yujii Silva Kadooka. -- Bauru, 2025
60 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências, Bauru
Orientadora: Andréa Carla Gonçalves Vianna

1. Compressão de dados. 2. Teoria da informação. 3. Compressão sem perdas – Huffman, LZ77, LZW, GZIP. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

Gustavo Yujii Silva Kadooka

Explorando Algoritmos de Compressão de Dados: Teoria, Implementação e Desempenho

Trabalho de Conclusão de Curso em Ciência da Computação apresentado ao Departamento de Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Profa. Dra. Andréa Carla Gonçalves Vianna
Orientador
Universidade Estadual Paulista “Júlio de Mesquita Filho”
Faculdade de Ciências
Departamento de Ciência da Computação

Professor Convidado 1
Universidade Estadual Paulista “Júlio de Mesquita Filho”
Faculdade de Ciências
Departamento de Ciência da Computação

Professor Convidado 2
Universidade Estadual Paulista “Júlio de Mesquita Filho”
Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, 11 de novembro de 2025.

À minha amada família, fonte inesgotável de inspiração e força.

Agradecimentos

Agradeço à Universidade Estadual Paulista “Júlio de Mesquita Filho” – Faculdade de Ciências, Campus de Bauru –, por ter sido o espaço onde pude me desenvolver como estudante, profissional e indivíduo. Sou grato a todos os professores e funcionários da instituição pelo conhecimento compartilhado e pelo apoio ao longo da graduação.

À minha orientadora, Profa. Dra. Andréa Carla Gonçalves Vianna, pela disponibilidade, paciência diante das dificuldades e por acreditar no meu potencial durante o desenvolvimento deste trabalho.

Aos meus colegas de curso, que estiveram presentes ao longo desta jornada, dividindo aprendizados, desafios e conquistas. O convívio e a troca de experiências tornaram a caminhada mais leve e significativa.

À minha irmã Gabriely, pelas palavras de incentivo e carinho em todos os momentos, mesmo nos mais difíceis.

E, com o meu mais profundo e sincero agradecimento, menciono especialmente meus pais – Flávia e Silvio –, por todo o apoio emocional, pelos ensinamentos de vida, pela confiança depositada em mim e por sempre me encorajarem a seguir em frente. Esta conquista é, acima de tudo, de vocês.

Resumo

O volume crescente de dados digitais gerados diariamente tem intensificado a necessidade de técnicas eficientes de compressão de dados, com o objetivo de otimizar o espaço de armazenamento e a largura de banda para transmissão. Este trabalho apresenta um estudo abrangente, a implementação prática e a análise de desempenho de algoritmos clássicos de compressão de dados, a saber: Huffman, LZ77, LZW e GZIP. São explorados os conceitos teóricos fundamentais de cada algoritmo, seguidos pela implementação prática utilizando a linguagem de programação C++. As implementações suportam múltiplos formatos de arquivo, incluindo texto simples (.txt), imagens bitmap (.bmp) e arquivos de áudio PCM (.wav), demonstrando versatilidade e aplicabilidade em diferentes domínios de dados. Para facilitar a interação do usuário, foi desenvolvida uma interface gráfica utilizando a biblioteca GTK, permitindo a seleção dos algoritmos de compressão e o arquivo de entrada. Ademais, para realizar uma análise comparativa rigorosa, foram utilizados scripts em Python para processar os dados experimentais e gerar representações gráficas dos principais indicadores de desempenho, como a taxa de compressão e o tempo de execução para diferentes tipos de arquivos. Os resultados experimentais indicam que não há um algoritmo que se destaque universalmente, pois cada um apresenta vantagens específicas conforme o tipo de arquivo e as características dos dados. Isso ressalta a importância da escolha do método de compressão adequado às necessidades específicas da aplicação. O projeto contribui tanto com uma ferramenta prática para compressão de dados quanto com um recurso educacional que auxilia na compreensão dos algoritmos clássicos e seu comportamento em cenários reais.

Palavras-chave: Compressão de dados. Algoritmos de compressão. Huffman. LZ77. LZW. GZIP. C++. Python. GTK.

Abstract

The growing volume of digital data generated daily has increased the need for efficient data compression techniques to optimize storage space and transmission bandwidth. This work presents a comprehensive study, implementation, and performance analysis of classical data compression algorithms, namely Huffman, LZ77, LZW, and GZIP. The theoretical concepts behind each algorithm are explored in depth, followed by practical implementations using C++ programming language. The implementations support multiple file formats including plain text (.txt), bitmap images (.bmp), and PCM audio files (.wav), demonstrating versatility and applicability in different data domains. To facilitate user interaction, a graphical user interface (GUI) was developed with the GTK library, enabling selection of compression algorithms, and input files. Furthermore, to conduct a rigorous comparative analysis, Python scripts were employed to process experimental data and generate graphical representations of key performance metrics such as compression ratio and execution time for different file types. The experimental results indicate that no single algorithm universally outperforms others; rather, each one exhibits specific advantages depending on the file type and data characteristics. This highlights the importance of choosing the appropriate compression method tailored to the application's needs. The project contributes as a practical tool for data compression and an educational resource that aids in understanding fundamental compression algorithms and their behavior in real-world scenarios.

Keywords: Data compression, compression algorithms, Huffman coding, LZ77, LZW, GZIP, C++, Python, GTK.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

LZ77	Lempel-Ziv 1977 (algoritmo de compressão)
LZW	Lempel-Ziv-Welch (algoritmo de compressão)
GZIP	GNU zip (algoritmo de compressão)
GTK	GIMP Toolkit (biblioteca gráfica para interface)

Sumário

1 Introdução

Para iniciar a produção em .tex é necessário instalar os pacotes básicos da linguagem e seus compiladores. O MiKTeX é um pacote básico para o Windows (miktex.org/download) e o MacTeX um pacote básico para o Mac (tug.org/mactex) que contém o mínimo necessário de TeX/LaTeX para rodar. Ele já vem com os compiladores nativos da linguagem e uma IDE (TeXworks, para edição do texto) que possui o compilador integrado.

Normalmente é utilizado o modo pdfLaTeX + MakeIndex + BibTeX para compilar um arquivo .tex. Existem outros formatos de compiladores, mas essa opção é capaz de gerar um .pdf automático após a compilação e ainda por cima adicionar as funcionalidades do BibTeX (recursos para criação e montagem automática de fontes bibliográficas).

Além disso, também é necessária a instalação do pacote abnTeX2. Esse tutorial <https://github.com/abntex/abntex2/wiki/Instalacao> provém o passo a passo de como instalar cada componente do TeX, em qualquer sistema operacional (Linux, Mac OS e Windows). Caso esteja utilizando o MiKTeX, ele é capaz de efetuar o download do pacote automaticamente, apenas instale-o, abra o projeto.tex e compile-o, ele irá requisitar a autorização para baixar automaticamente os pacotes que faltam para efetuar a compilação.

Com tudo em mãos e o compilador funcionando, é hora de abrir o modelo (projeto.tex) e começar a escrever o texto. É possível perceber no código a estrutura do arquivo e os campos possíveis de edição. Ao escrever o texto, ele é escrito normalmente, sendo que existem diversos comandos para estilizá-lo, criar tabelas, figuras, dentre outros. A seguir abordaremos os principais comandos e funções que podem ser utilizadas em um projeto básico de TCC. Para outras funções e pacotes, procure no Google, a comunidade é ativa e provavelmente já deve ter feito o que é de sua necessidade.

O arquivo projeto.tex contém os pacotes e comandos básicos que definem a estrutura desse texto já no formato requisitado pela ABNT. Dentro dele é possível ver que estamos importando outros dois arquivos .tex (introducao e conclusao), ou seja, esses arquivos estão sendo basicamente concatenados com o comando “input”. A divisão não é necessária, mas pode ser que auxilie na escrita do texto ao deixar as coisas mais separadas e organizadas, não sendo um único arquivo cheio de linhas e linhas de código.

1.1 Modificadores de Texto

Os modificadores de texto mais simples utilizados são o negrito (“textbf”) **texto em negrito** e o itálico (“emph”) *texto em itálico*.

1.2 Seções

Seções podem ser criadas a partir do comando “section” e hierarquizadas abaixo do capítulo principal. É possível referenciá-las, por exemplo, Seção 1.2 corresponde a seção atual em que estamos. Já se quisermos referenciar alguma outra coisa, é só utilizarmos o comando “ref” presente no código desse texto, por exemplo, Capítulo 1.

1.2.1 Subseções

Subseções também podem ser criadas com o comando “subsection” e referenciadas 1.2.1.

1.2.1.1 Sub-subseções

Também há mais um nível que pode ser criado com o comando “subsubsection”.

1.3 Alíneas

a) As alineas devem ser criadas desse modo, com o comando begin{alineas}. Isso é necessário para que estejam no formato definido pelo pacote abnTeX2 e, consequentemente, no formato definido pela ABNT.

b) Cada item da alínea pode ser invocado com um comando item.

c) O fim de cada alínea é determinado por end{alineas}.

1.4 Tabelas

As tabelas também podem ser referenciadas como se fossem seções ou figuras, por exemplo, esta é a Tabela 1.

Quando uma tabela é criada com begin{table}, ela é automaticamente adicionada à Lista de Tabelas.

Tabela 1 – Exemplo de transações de mercado.

TID	Conjunto de Itens
1	{Pão, Leite}
2	{Pão, Fralda, Cerveja, Ovos}
3	{Leite, Fralda, Cerveja, Coca-Cola}
4	{Pão, Leite, Fralda, Cerveja}
5	{Pão, Leite, Fralda, Coca-Cola}

1.5 Quadros

Este modelo vem com o ambiente `quadro` e impressão de Lista de quadros configurados por padrão. Verifique um exemplo de utilização:

Quadro 1 – Exemplo de quadro.

Pessoa	Idade	Peso	Altura
Marcos	26	68	178
Ivone	22	57	162
...	...	...	...
Sueli	40	65	153

Fonte: Elaborado pelo autor.

Este parágrafo apresenta como referenciar o quadro no texto, requisito obrigatório da ABNT. Primeira opção, utilizando `autoref`: Ver o seção 1.5. Segunda opção, utilizando `ref`: Ver o Quadro 1.5.

1.6 Algoritmos

O pacote nicealgo incluído nos arquivos desse projeto é responsável por disponibilizar comandos extras, não inerentes ao básico TeX, para a criação de algoritmos. Um exemplo do Algoritmo 1 é escrito a seguir. Eles também pode ser referenciados como se fossem tabelas ou figuras.

Algoritmo 1 – *ALGORITMO AIS*

ENTRADA: Conjunto Frequente L = 0 e Grupo de Fronteira F = 0.

1. ***Enquanto*** $F \neq 0$, ***faça***
2. ***Seja*** *conjunto candidato* $C = 0$;
3. ***Para cada*** *tuplas* t *da base de dados,* ***faça***
4. ***Para cada*** *conjuntos de itens* f *em* F, ***faça***
5. ***Se*** t *contém* f, ***então***
6. ***Seja*** C_f = *conjuntos de itens candidatos extensões de* f *e contidos em* t;
7. ***Para cada*** *conjunto de itens* c_f *em* C_f, ***faça***
8. ***Se*** $c_f \in C$, ***então***
9. c_f.*contagem* = c_f.*contagem*$+1$;
10. ***Se não***
11. c_f.*contagem* $= 0$;
12. $C = C + c_f$;
13.
14.
15.
16. ***Seja*** *F = 0;*
17. ***Para cada*** *conjunto de itens* c *em* C, ***faça***
18. ***Se*** $contagem(c)/tamanho\_db > minsupport$, ***então***
19. $L = L + c$;
20. ***Se*** c *deve ser usado como a próxima fronteira,* ***então***
21. $F = F + c$;
22.
23.

1.7 Códigos

Códigos podem ser criados a partir do comando begin{lstlisting} e end{lstlisting}. É possível passar parâmetros para essa função, como por exemplo, a linguage do código e a legenda dele. Por exemplo: \begin{lstlisting}[language=Python, caption=Exemplo de código em Python]

Código 1 – Exemplo de código em Python.

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
```

```
M = None #to become the incidence matrix
VT = np.zeros((n*m,1), int)  #dummy variable

#compute the bitwise xor matrix
M1 = bitxormatrix(genl1)
M2 = np.triu(bitxormatrix(genl2),1)

for i in range(m-1):
    for j in range(i+1, m):
        [r,c] = np.where(M2 == M1[i,j])
        for k in range(len(r)):
            VT[(i)*n + r[k]] = 1;
            VT[(i)*n + c[k]] = 1;
            VT[(j)*n + r[k]] = 1;
            VT[(j)*n + c[k]] = 1;

            if M is None:
                M = np.copy(VT)
            else:
                M = np.concatenate((M, VT), 1)

            VT = np.zeros((n*m,1), int)

return M
```

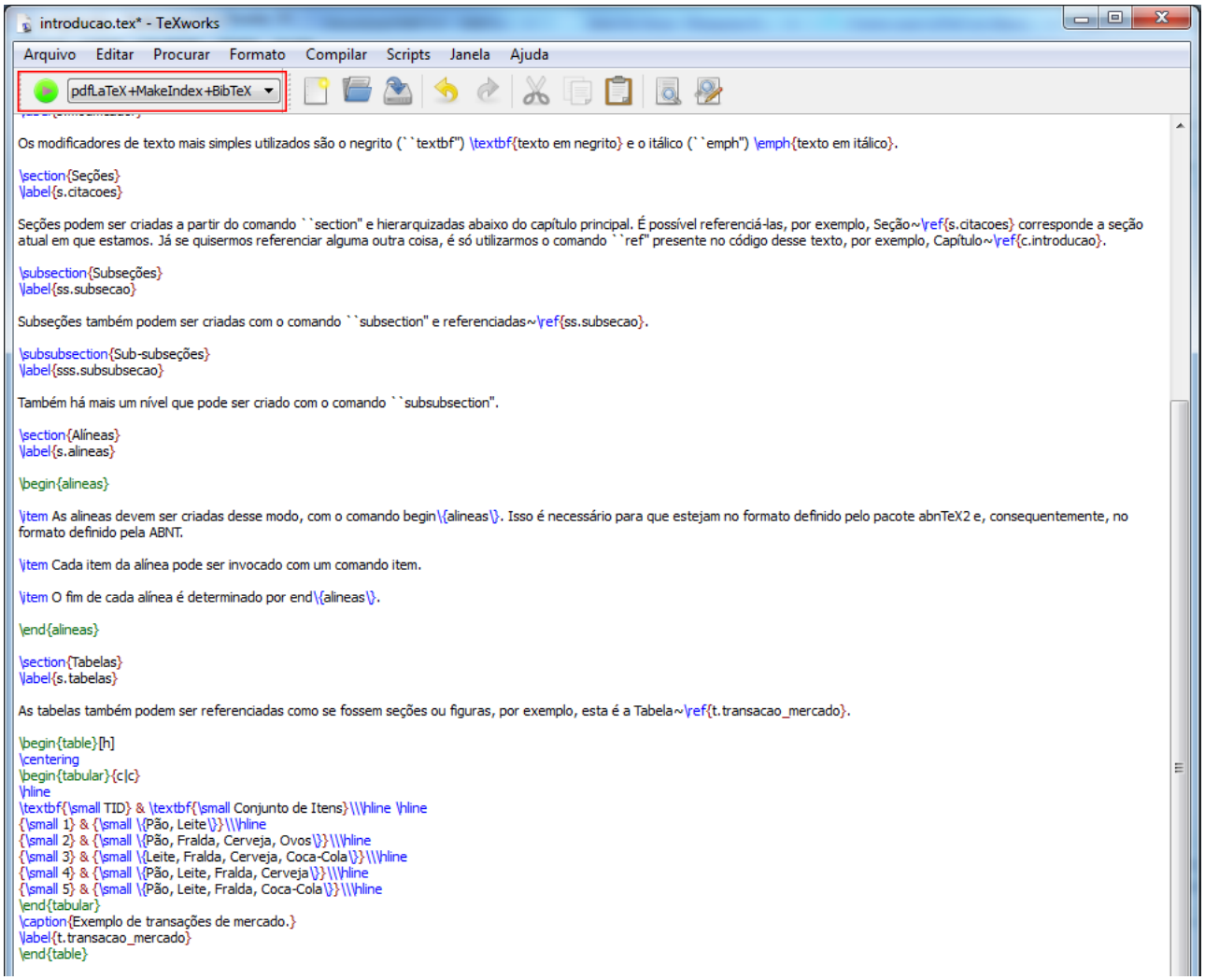
1.8 Figuras

Este parágrafo apresenta como referenciar figura no texto, requisito obrigatório da ABNT, por exemplo a Figura 1. Atente-se ao código para perceber um possível redimensionamento com a função scale e o caminho de onde a figura deve ser retirada. Quando uma figura é criada com begin{figure}, ela é automaticamente adicionada à Lista de Ilustrações.

1.9 Equações

O LaTeX também é muito famoso pela forma em que consegue tratar funções e símbolos matemáticos. A partir da utilização de dois cifrões ($codigo matemático$) é possível identificar ao compilador que a escrita a seguir são símbolos e códigos originários do pacote matemático do LaTeX, por exemplo $\phi = 1 + x$.

Figura 1 – Exemplo do ambiente TeXworks.



Fonte: Elaborada pelo autor.

Também podemos definir equações utilizando os comandos begin{equation} e end{equation}. Por exemplo:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i=1}^{m}\sum_{j=1}^{n} v_i h_j w_{ij}, \tag{1.1}$$

$$P(\mathbf{v},\mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum\limits_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}, \tag{1.2}$$

$$\hat{\phi}^j = \begin{cases} \hat{\phi}^j \pm \varphi_j \varrho & \text{com probabilidade PAR} \\ \hat{\phi}^j & \text{com probabilidade (1-PAR).} \end{cases} \tag{1.3}$$

Existem diversos sites no Google que contém códigos de símbolos e funções matemáticas de todos os tipos. Exemplo:

estudijas.lu.lv/pluginfile.php/14809/mod_page/content/16/instrukcijas/matematika_moodle/LaTeX_Symbols.pdf.

1.10 Como citar as referências

Aqui está um exemplo de como podemos referenciar as bibliografias utilizadas no trabalho. Elas são guardadas na forma de metadados (tags) no arquivo .bib a qual é importada no projeto principal (projeto.tex).

E podemos citá-las de acordo com os identificadores atribuídos para cada referência, por exemplo, (STONEBRAKER et al., 1993), (ROCHA; CAPPABIANCO; FALCÃO, 2009) e (KERAS. . . , 2018).

Após citar um item de referência bibliográfica com o comando “cite”, ela será automaticamente padronizada e incluída na página de Referências de seu arquivo. Atualmente os maiores sites portadores de artigos, periódicos, dentre outros (IEEE, Springer, etc) já conseguem exportar a publicação desejado no formato BibTeX, sendo facilmente adicionado ao arquivo .bib de seu trabalho.

Muitas vezes não é possível exportar publicações diretamente para o formato BibTeX, como, por exemplo, na citação de sites. Para mais informações sobre como criar manualmente arquivos BibTeX: <https://github.com/abntex/limarka/wiki/Adicionando-referÃłncias>

2 Problema

3 Objetivos

4 Fundamentação Teórica

5 Implementação

6 Análise Comparativa

7 Conclusão

Os arquivos estão sendo concatenados. Podemos continuar a nossa escrita em outro arquivo .tex desde que ele seja importado no projeto principal, que é sempre o utilizado para efetuar a compilação.

Referências

KERAS Documentation. 2018. Disponível em: <<https://keras.io/>>. Acesso em 22 de Outubro de 2018.

ROCHA, L. M.; CAPPABIANCO, F. A. M.; FALCÃO, A. X. Data clustering as an optimum-path forest problem with applications in image analysis. *International Journal of Imaging Systems and Technology*, Wiley Periodicals, v. 19, n. 2, p. 50–68, 2009.

STONEBRAKER, M.; AGRAWAL, R.; DAYAL, U.; NEUHOLD, E. J.; REUTER, A. Dbms research at a crossroads: The vienna update. In: *VLDB '93 Proceedings of the 19th International Conference on Very Large Data Bases*. [S.l.]: Morgan Kaufmann Publishers Inc., 1993. p. 688–692.