INSTITUTO FEDERAL
Paraná
Campus Pinhais

Bacharelado em Ciência da Computação
Estruturas de Dados Avançadas — 6 de maio de 2026
Trabalho 2 — v1.0

1 Introdução

O objetivo deste trabalho é a solução de um problema do mundo real que utilize uma estrutura de árvore avançada e algoritmos de compressão.

1.1 O problema

Sistemas operacionais utilizam árvores B (ou similares) para encontrar rapidamente as informações referentes a um arquivo que precisa ser acessado. De maneira similar, sistemas gerenciadores de bancos de dados utilizam árvores desse tipo para encontrar informações sobre uma entrada (registro) no banco de dados. Por exemplo, um índice de uma tabela pode ser construído com uma árvore B, usada para encontrar rapidamente onde estão as informações de cada linha (registro) indexado.

Em ambos os casos, essas estruturas devem ser salvas em um dispositivo de armazenamento não-volátil (como HDs ou SSDs) para que sejam usadas ao longo de várias inicializações do sistema. Grandes bancos de dados ou grandes sistemas de arquivos podem ter árvores enormes, que servem apenas para localizar as informações que são de fato relevantes. Por isso, comprimir essas estruturas ao guardá-las pode ser uma boa ideia.

2 O seu trabalho

Você deve implementar uma árvore B, encontrar uma forma de representar essa árvore em um arquivo e implementar a compressão dessas informações. Não serão aceitos trabalhos que utilizem implementações prontas ou bibliotecas.

Seu trabalho está dividido em dois programas: um para a árvore e outro para a parte de compressão.

2.1 A árvore B

A árvore B deve aceitar chaves do tipo *string*. Você pode escolher como implementar as chaves: usando *strings* mesmo, usando um método que converta em números (parecido com *hash*), ou outras ideias mirabolantes.

Você deve escolher duas ordens (tamanho dos nós) muito diferentes para a árvore B, para observar o comportamento do seu programa em ambos os casos.

Seu programa deve aceitar os seguintes comandos, iterativamente:

- `insert <chave>`: insere nova chave na árvore;
- `read <chave>`: encontra a chave na árvore e imprime as informações;

- `delete <chave>`: remove uma chave da árvore, se ela existir;
- `save <arquivo>`: salva a árvore no arquivo;
- `load <arquivo>`: carrega uma árvore a partir do arquivo.

Você pode escolher a forma de salvar a árvore em um arquivo, justificando sua escolha.

2.2 A compressão

Você deve implementar **outro** programa, responsável por compactar e descompactar a árvore salva.

Você deve implementar dois algoritmos de compressão: LZW e Huffman. Isso permitirá comparar o desempenho de ambos os algoritmos.

2.3 Testes e desempenho

Crie scripts de teste que exercitem sua árvore em diversos tamanhos: desde árvores pequenas, com dezenas de chaves, a árvores grandes, com bilhões de chaves. Crie testes que envolvam várias inserções, remoções e buscas na árvore.

Sua árvore deve medir, para cada operação, quantos nós foram visitados, o tempo gasto na operação e a quantidade de memória usada pela árvore no final da operação.

Você também deve testar cada árvore salva com ambos os algoritmos de compressão, medindo o tempo, memória gasta e a taxa de compressão obtida.

3 Entrega

Você deverá entregar pelo SUAP, até 10/junho, um arquivo *.zip* contendo uma pasta, que por sua vez contém:

- Os arquivos de código e de testes do trabalho;
- Um arquivo de texto chamado `README.md` contendo o nome completo dos integrantes do grupo, uma breve descrição de cada arquivo no trabalho, instruções para execução dos testes e dos programas, e quaisquer outras informações sobre o progresso do trabalho que julgarem relevantes;
- Um arquivo chamado `relatorio.pdf` que contém o relatório da equipe em formato `pdf`;
- Absolutamente mais nada! Remova todos os executáveis e arquivos intermediários que estejam na pasta do trabalho antes de enviá-lo.

O arquivo compactado descrito acima deve ser nomeado com as iniciais do nome de cada integrante separados por um hífen. Por exemplo: se o Fulano da Silva Sousa fez o trabalho com o João de Souza, o arquivo deve ser nomeado `fss-js.zip`. A pasta, dentro do arquivo compactado, deve ter o mesmo nome, a não ser pela extensão.

O trabalho pode ser feito individualmente ou em duplas.

Histórico das Revisões:

- 06/mai/2026 - v1.0: primeira versão.