Trabalho Prático 3 - B-Tree | Trie

Algoritmos e Classificação de Dados

Otávio Garcia de Oliveira Farias, Pedro Afonso Barcellos Pires & Thiago V. Pfeifer

Instituto de Ensino Superior – Universidade Federal do Pampa (UNIPAMPA) - Engenharia de Computação

otaviogarcia.aluno@unipampa.edu.br, pedropires.aluno@unipampa.edu.br, thiagopfeifer.aluno@unipampa.edu.br

Resumo: *Este trabalho tem como objetivo implementar uma aplicação que utiliza uma estrutura de Árvore B. Para este propósito, desenvolvemos um programa, em Java, capaz de ler dados de um arquivo "musica.csv" e organizá-los de forma eficiente em uma árvore B.*

Abstract: This work aims to implement an application that uses a B-Tree structure. For this purpose, we developed a program, in Java, capable of reading data from a "musica.csv" file and organizing them efficiently in a B-tree.

Introdução:

Este trabalho propôs a utilização das estruturas de dados Árvore B ou Trie para resolver um problema prático escolhido pelo grupo. Optamos pela Árvore B devido à sua versatilidade na organização de grandes volumes de dados e sua eficiência em operações de busca, inserção e remoção. O domínio do problema definido para este trabalho foi inspirado no cenário de 'Júlio', que estava cansado de buscar manualmente suas músicas preferidas para escutar. Ele resolveu organizar todas as músicas em uma Árvore B para facilitar o acesso e poder curtir a trilha sonora que desejar, com agilidade para buscar, excluir e adicionar novas faixas. A aplicação desenvolvida visa organizar informações sobre músicas, incluindo artista, nome da música, letra e código da música (utilizado como chave), facilitando a consulta e manipulação desses dados.

Desenvolvimento da Solução:

A primeira etapa do desenvolvimento foi a definição dos requisitos. O programa deveria ser capaz de ler um arquivo CSV chamado "musica.csv", que contém registros com artista, nome da música, letra e código da música. Esses dados lidos deveriam ser organizados em uma estrutura de Árvore B, permitindo operações eficientes de busca, inserção e remoção. Esses requisitos orientaram todo o processo de projeto e implementação da aplicação.

A estrutura da Árvore B foi projetada para armazenar como chave o código da música, enquanto os valores associados incluíam informações adicionais como o artista, nome da música e letra. Para possibilitar isso, foram desenvolvidas funções responsáveis pela

leitura e pelo parsing dos dados do arquivo CSV, organizando-os adequadamente para a inserção na estrutura. O programa foi estruturado de forma modular, com três classes principais: **ArvoreB.java,** que implementa a estrutura da árvore B e inclui métodos de busca, inserção, remoção e carregamento de dados de arquivos CSV; **CriaArvoreB.java**, que gerencia a interação com o usuário e oferece um menu interativo para realizar as operações na árvore; e **Musica.java**, que representa os dados de cada música e inclui métodos para manipulação e comparação desses dados.

A implementação foi realizada inteiramente em Java, utilizando conceitos de orientação a objetos para organizar o código de forma eficiente. As operações básicas da Árvore B, como inserção e busca, foram desenvolvidas com base nos algoritmos descritos por Ziviani (2012). A remoção foi implementada com suporte a critérios como chave, nome da música, artista e letra. Além disso, o programa permite carregar músicas de arquivos CSV e exibir os resultados das buscas realizadas. A persistência dos dados foi garantida através do uso de arquivos binários, que armazenam a estrutura completa da árvore e os dados nela contidos.

A validação da solução foi realizada por meio de testes manuais. Testes de funcionalidade foram aplicados a cada operação da árvore, como busca, inserção e remoção, verificando o comportamento correto em cenários simples. Para validar o carregamento de dados, utilizamos um arquivo CSV contendo um pequeno conjunto de músicas. O programa foi testado em diferentes critérios de busca, incluindo por chave, artista, nome da música e letra, confirmando o funcionamento esperado em todos os casos. Apesar de não realizarmos testes de desempenho automatizados ou comparações diretas com outras estruturas, as funcionalidades implementadas atenderam aos requisitos propostos.

Resultados e Discussões:

Os testes realizados demonstraram que o programa desenvolvido atendeu aos objetivos propostos. A Árvore B se mostrou eficiente na organização e recuperação de dados, mantendo a integridade mesmo após inserções e remoções consecutivas. As buscas por critérios específicos, como artista, letra e nome da música, demonstraram ser práticas e eficazes, enquanto a remoção de músicas com base nesses mesmos critérios confirmou a flexibilidade da solução. Apesar disso, foram observadas algumas limitações, como a ausência de testes de desempenho em bases de dados maiores e a falta de comparações diretas com outras estruturas de dados. Esses pontos podem ser abordados em trabalhos futuros.

Conclusão:

A aplicação desenvolvida demonstrou a versatilidade e eficiência da estrutura de Árvore B. O programa alcançou todos os objetivos definidos, validando a escolha dessa estrutura para o problema proposto. Como sugestões de trabalhos futuros, recomenda-se a implementação de uma interface gráfica para facilitar o uso da aplicação e a exploração de outras estruturas, como Tries, para resolver problemas semelhantes. A inclusão de suporte a bases de dados ainda maiores e o aprimoramento do desempenho em operações de remoção também podem ser explorados.

Referências Bibliográficas:

- Ziviani, N. (2012). *Projeto de Algoritmos*.
- Cormen T. H., et al. (2024). *Algoritmos: Teoria e Prática*.
- Sedgewick, R., & Wayne, K. (2011). *Algorithms*.
- Arquivos e slides disponibilizados no Moodle.
- Acervo “musica.csv” pelo professor Julio Saraçol Domingues Junior