UNIVERSIDADE DO ESTADO DO PARÁ - CAMPUS XXII
BACHARELADO EM ENGENHARIA DE SOFTWARE

ATHUS GABRIEL GUARANY SOARES
DIEGO GABRIEL PESSOA AMORIM
MARCELLO COSTA DOS SANTOS
RYAN RICARDO DE SOUZA

ÁRVORES: GERENCIAMENTO DE LIVROS PARA BIBLIOTECA DA UEPA

Árvores: Gerenciamento de livros para biblioteca da UEPA.

ANANINDEUA
2024

SUMÁRIO

1. Introdução

A gestão de acervos em bibliotecas tradicionais enfrenta desafios como busca demorada, organização complexa e atualização manual de dados. Para superar este desafio, o código desenvolvido implementa a estrutura de dados de árvores binárias de busca para armazenar informações bibliográficas, como id, título e autor. Cada livro é representado por um nó na árvore, com ponteiros para seus filhos (esquerdo e direito). A ordenação dos livros é realizada com base em um critério escolhido, como título ou autor, permitindo buscas eficientes.

2. Estrutura do Livro

A **estrutura do Livro** funciona como um armário organizado para cada exemplar em nossa biblioteca particular. Cada livro possui seu próprio compartimento com informações essenciais para identificá-lo e conhecê-lo melhor.

2.1 Identificador Único (ID)

- Um código exclusivo que serve como **carteira de identidade** do livro.
- Garante que cada livro tenha sua própria "casa" na árvore binária, sem confusões.

2.2 Título

- A **capa do livro**, que nos dá a primeira pista sobre a história que ele guarda.
- Armazenado como uma string, permitindo títulos longos e descritivos de até 100 caracteres.

2.3 Autor

- O **criador da obra**, responsável por dar vida à história.
- Também é armazenado como string, capturando nomes completos e pseudônimos de até 100 caracteres.

Exemplo:

Imagine um livro com ID 123, título "O Senhor dos Anéis" e autor J.R.R. Tolkien. Na estrutura do Livro, cada informação teria seu próprio espaço:

- **ID:** 123
- **Título:** "O Senhor dos Anéis"
- **Autor:** "J.R.R. Tolkien"

2.4 Importância da Estrutura do Livro

- **Organização:** Permite armazenar informações de forma estruturada e eficiente.
- **Acessibilidade:** Facilita a busca e recuperação de livros específicos na árvore binária.
- **Flexibilidade:** Suporta diferentes tipos de dados (strings, números), adaptando-se a diversas informações sobre os livros.

Ou seja, a estrutura do Livro é a base para organizar e gerenciar nossa coleção de livros na árvore binária, garantindo que cada exemplar tenha seu lugar e seja facilmente encontrado.

3. Estrutura da Árvore Binária

A estrutura "NoArvore" é como **a espinha dorsal da árvore binária de busca**, dando forma e organização à coleção de livros. Cada nó dessa estrutura representa um ponto crucial na árvore, onde um livro reside e se conecta aos seus "descendentes".

3.1 Estrutura “NoArvore”

- **Livro:** No coração de cada nó reside um **livro**, com suas informações como ID, título e autor.
- **Ponteiros Esquerda e Direita:** Como mãos que apontam para o futuro, os ponteiros esquerda e direita guiam-nos para os **filhos** do nó.

- **Organização Hierárquica:** Através desses ponteiros, os nós se conectam em uma **estrutura hierárquica**, formando uma árvore onde cada livro se encontra em seu lugar correto, ordenado por ID.

3.2 Importância da Estrutura "NoArvore"

- **Eficiência na Busca:** Permite encontrar livros rapidamente, navegando pela árvore através dos ponteiros.
- **Organização:** Mantém os livros ordenados por ID, facilitando a consulta e manipulação da coleção.
- **Flexibilidade:** Adapta-se a qualquer número de livros, expandindo ou diminuindo a árvore conforme necessário.

Ou seja, a estrutura "NoArvore" é a base fundamental da nossa árvore binária de busca, organizando os livros em uma hierarquia eficiente e permitindo que os encontremos e gerenciemos com facilidade.

4. Função para criar um nó

A função "criarNo" atua como o broto da árvore binária de busca, concedendo vida a novos nódulos que abrigarão novos livros. O broto recebe um livro e retorna um ponteiro para o novo nó criado, pronto para se integrar à árvore.

4.1 Estrutura “criarNo”

1. **Aloca Memória:** O broto **aloca memória** suficiente para o novo nó.
2. **Inicialização:** O broto **inicializa os valores** do novo nó:
    - **Livro:** O livro recebido é armazenado no nó, pronto para ser encontrado e consultado.
    - **Filhos:** Os ponteiros esquerda e direita são inicializados como NULL, pois o novo nó ainda não possui filhos.

Retorno: O broto **retorna um ponteiro** para o novo nó criado, permitindo que ele seja inserido na árvore binária de busca e faça parte da biblioteca.

Exemplo:

Caso o broto receba o livro "O Senhor dos Anéis". Ele aloca memória, inicializa o livro no novo nó e os ponteiros esquerda e direita como NULL. Por fim, retorna o ponteiro para o novo nó, pronto para ser adicionado à árvore.

4.2 Importância da Função "criarNo"

- **Criação de Nódulos:** É essencial para **expandir a árvore binária de busca**, adicionando novos livros à nossa coleção.
- **Gerenciamento de Memória:** Aloca memória de forma eficiente, **evitando desperdícios e garantindo a saúde da árvore**.
- **Inicialização Correta:** Assegura que os novos nódulos sejam inicializados com os valores corretos, **mantendo a organização da árvore**.

Ou seja, a função "criarNo" é a chave para o **crescimento da árvore binária de busca**, permitindo que novos livros sejam acolhidos e organizados de forma eficaz e eficiente.

5. Função para inserir um livro

A função "inserirLivro" atua como **corretor imobiliário da árvore binária de busca**, buscando o **local ideal** para cada novo livro que chega à biblioteca. Ela recebe um livro como cliente e o guia pela árvore, comparando seu "ID" com os dos nós existentes até encontrar o lar ideal, sempre **mantendo a ordem da árvore**.

5.1 Estrutura “inserirLivro”

- **Comparação de IDs:** O corretor inicia a jornada comparando o "ID" do livro com o "ID" do nó raiz da árvore.
    - Se o "ID" do livro for **menor** que o do nó raiz, a busca segue pelo **filho esquerdo**.

    - Se o "ID" do livro for **maior** que o do nó raiz, a busca segue pelo **filho direito**.
- **Descida Recursiva:** O corretor repete o processo de comparação de IDs com os nós **filhos** até encontrar o local ideal.
- **Inserção do Livro:** Ao encontrar o nó correto (onde o "ID" do livro se encaixa na ordenação), o corretor **insere o livro** nesse nó.
    - O livro é armazenado no nó, pronto para ser encontrado e consultado.
    - Os ponteiros esquerda e direita do nó são atualizados para manter a ordenação da árvore.
- **Retorno:** O corretor **retorna a raiz da árvore atualizada**, agora com o novo livro em seu devido lugar.
-

Exemplo:

Caso o corretor receba o livro "O Hobbit" como cliente. Ele inicia a busca no nó raiz, comparando o "ID" do livro com o do nó. Se o "ID" do livro for menor, ela segue pelo filho esquerdo, repetindo o processo até encontrar o local ideal. Ao encontrar, ele insere o livro no nó e atualiza os ponteiros esquerda e direita. Por fim, retorna a raiz da árvore atualizada com o novo livro em seu devido lugar.

5.2 Importância da Função "inserirLivro"

- **Organização Eficiente:** Mantém a **propriedade de ordenação** da árvore binária de busca, facilitando a busca por livros.
- **Gerenciamento da Coleção:** Permite **adicionar novos livros** à nossa biblioteca, expandindo a coleção.
- **Inserção Segura:** Garante que cada livro seja inserido no **local correto**, evitando duplicatas e desordens.

Ou seja, a função "inserirLivro" é a **alma da árvore binária de busca**, responsável por **organizar e gerenciar** a coleção de livros de forma eficiente, sempre buscando o local ideal para cada novo exemplar.

6. Função para buscar um livro por id

A função "buscarLivro" atua como arqueóloga **da árvore binária de busca**, como se fosse uma **caça ao tesouro** para encontrar o livro desejado. Ela recebe o "ID" do livro como mapa e navega pela árvore, comparando o "ID" alvo com os dos nós até encontrar o livro ou chegar a um beco sem saída.

6.1 Estrutura “buscarLivro”

- **Comparação de IDs:** A arqueóloga inicia a jornada comparando o "ID" do livro alvo com o "ID" do nó raiz da árvore.
    - Se o "ID" do livro for **igual** ao do nó raiz, a busca termina e o **livro é encontrado** nesse nó.
    - Se o "ID" do livro for **menor** que o do nó raiz, a busca segue pelo **filho esquerdo**.
    - Se o "ID" do livro for **maior** que o do nó raiz, a busca segue pelo **filho direito**.
- **Descida Recursiva:** A arqueóloga repete o processo de comparação de IDs com os nós **filhos** até encontrar o livro ou chegar a um nó nulo.
- **Livro Encontrado:** Se o livro for encontrado, a arqueóloga **retorna o nó** que o contém, permitindo o acesso às informações do livro.
- **Livro Não Encontrado:** Se o livro não for encontrado em nenhum nó da árvore, a arqueóloga **retorna NULL**, indicando que a caça ao tesouro não teve sucesso.

Exemplo:

Caso arqueóloga recebendo o "ID" 123 como mapa. Ela inicia a busca no nó raiz, comparando o "ID" 123 com o do nó. Se os IDs forem iguais, o livro foi encontrado e a arqueóloga retorna o nó com o livro. Se o "ID" 123 for menor, ela segue pelo filho esquerdo, repetindo o processo até encontrar o livro ou chegar a um nó nulo. Se o livro for encontrado, ela retorna o nó com o livro. Se o livro não for encontrado, ela retorna NULL.

6.2 Importância da Função "buscarLivro"

- **Eficiência na Busca:** Permite encontrar **livros específicos rapidamente**, navegando pela árvore binária de busca.
- **Acesso à Informação:** Fornece o **nó que contém o livro desejado**, permitindo o acesso às suas informações.

- **Verificação de Existência:** Indica se o **livro está presente na árvore**, facilitando o gerenciamento da coleção.

Ou seja, a função "buscarLivro" é a **bússola da árvore binária de busca**, guiando a **busca por livros específicos** de forma eficiente e precisa, permitindo que seja encontrado os tesouros literários almejados.

7. Função para listar os livros em ordem crescente

A função "listarLivros" atua como **narrador da biblioteca**, revelando **a coleção completa de livros** em ordem crescente de "ID". Ele realiza uma **travessia in-order** pela árvore binária de busca, como um contador de histórias experiente que visita cada livro em sua ordem natural.

7.1 Estrutura “listarLivros”

- **Travessia in-order:** O narrador inicia a jornada visitando a **subárvore esquerda** do nó atual.
    - Se a subárvore esquerda não for nula, ele chama a si mesmo recursivamente para listar os livros dessa subárvore.
- **Revelando o Livro Atual:** Após listar a subárvore esquerda, o narrador **imprime as informações do livro** do nó atual:
    - Título: O nome do livro.
    - Autor: A mente de autoria da obra.
    - ID: O código único do livro.
- **Travessia da Subárvore Direita:** Após isso, o narrador visita a **subárvore direita** do nó atual.
    - Se a subárvore direita não for nula, ele chama a si mesmo recursivamente para listar os livros dessa subárvore.

Exemplo:

Caso o narrador inicie a jornada no nó raiz. Ele visita a subárvore esquerda, recursando a si mesmo para listar os livros dessa subárvore. Em seguida, revela as informações do livro do nó raiz (título, autor, ID) e por fim, visita a subárvore direita, recursando a si mesmo novamente.

Esse processo se repete até que todos os livros da árvore binária de busca tenham sido listados em ordem crescente de "ID".

7.2 Importância da Função "listarLivros"

- **Organização e Visibilidade:** Permite visualizar a **coleção de livros ordenada por ID**, facilitando a identificação dos livros.
- **Gerenciamento Eficiente:** Auxilia no **gerenciamento da biblioteca**, permitindo verificar a quantidade de livros e identificar duplicatas.
- **Navegação Detalhada:** Permite **explorar a árvore binária de busca** e seus livros de forma estruturada e completa.

Ou seja, a função "listarLivros" é como se fosse a **voz da nossa biblioteca**, guiando pela **coleção de livros** em ordem crescente de "ID" e listando todas as obras literárias.

8. Função para remover livros

A função "removerLivro" atua como uma **faxina da biblioteca**, livrando-a de **livros desatualizados ou indesejados** com base no "ID" fornecido. Através de uma análise precisa e de **três estratégias de remoção**, garante que a biblioteca permaneça organizada e eficiente, mesmo após a despedida de alguns livros.

8.1 Estrutura “removerLivro”

- **Busca pelo Livro Condenado:** A faxina é iniciada **buscando o livro a ser removido** na árvore binária de busca. Ela segue o caminho traçado pelo "ID", comparando-o com os IDs dos nós até encontrar o livro desejado.
- **Analisando o Cenário:** Ao encontrar o livro, a faxina **analisa o cenário** para determinar a melhor estratégia de remoção:
    - **Nó com Zero Filhos:** Se o livro estiver em um nó sem filhos, a faxina simplesmente **remove o nó** da árvore, como se estivesse retirando um livro empoeirado da estante.
    - **Nó com Um Filho:** Se o livro estiver em um nó com apenas um filho, a faxina **promove o filho** para substituir o nó removido, como se estivesse colocando um novo livro no lugar do antigo.

    - **Nó com Dois Filhos:** Se o livro estiver em um nó com dois filhos, a faxina precisa de um **sucessor** para garantir a ordem da árvore. Ela busca o **sucessor mais à esquerda** na subárvore direita do nó a ser apagado, como se estivesse escolhendo o próximo livro da fila para ocupar o lugar vago.
- **Removendo o Livro:** Com a estratégia definida, a faxina **remove o livro desejado** da árvore, atualizando os ponteiros dos nós envolvidos para manter a integridade da estrutura.
- **Retorno:** A faxina conclui sua missão, **retornando o nó raiz da árvore atualizada**, pronta para receber novos livros ou enfrentar novos desafios de remoção.

Exemplo:

Caso a faxina receba o "ID" 123 como missão. Ela busca o livro com esse "ID" na árvore, encontrando-o em um nó com dois filhos. A faxina então identifica o sucessor mais à esquerda na subárvore direita do nó a ser apagado, remove o livro a ser apagado e promove o sucessor para ocupar seu lugar. Por fim, ela atualiza os ponteiros dos nós envolvidos e retorna o nó raiz da árvore atualizada, com a missão cumprida.

8.2 Importância da Função "removerLivro"

- **Organização da Biblioteca:** Permite **remover livros indesejados** da árvore binária de busca, liberando espaço para novos livros e **mantendo a biblioteca organizada**.
- **Eficiência da Estrutura:** Garante que a **remoção de livros não comprometa a estrutura da árvore binária de busca**, preservando sua eficiência para operações como busca e listagem.
- **Gerenciamento Inteligente:** Oferece diferentes estratégias de remoção para **lidar com diversos cenários**, como nós com zero, um ou dois filhos.

Ou seja, a função "removerLivro" é a **guardiã da organização** da nossa biblioteca binária de busca. Através de sua atuação precisa e estratégica, ela garante que a biblioteca permaneça limpa, organizada e eficiente, mesmo após a remoção de livros, permitindo que novos títulos sejam acolhidos e que a experiência de leitura seja sempre a melhor possível.

9. Função principal

A função main atua como o **portal principal** para a biblioteca, fornecendo ao usuário um **menu de opções** que permite navegar pela árvore binária de busca e realizar diversas operações com os livros. Através de um **laço while infinito**, ela se transforma em um portal sempre aberto, pronto para atender aos comandos do usuário até que este decida partir.

9.1 Função Main

- **Apresentação do Menu:** O portal se abre, exibindo um **menu de opções** convidativo:
    - **Inserir Livro:** Adiciona um novo livro à biblioteca, expandindo a coleção.
    - **Buscar Livro:** Encontra um livro específico na biblioteca.
    - **Remover Livro:** Exclui um livro da biblioteca, liberando espaço para outros novos.
    - **Listar Livros:** Apresenta todos os livros da biblioteca em ordem crescente de ID, como uma estante organizada.
    - **Verificar se a Árvore Está Vazia:** Informa se a biblioteca está sem livros.
    - **Sair:** Fecha o portal e encerra o programa.
- **Aguardando o Comando do Usuário:** O portal se coloca à disposição, **aguardando o comando do usuário**. O usuário digita sua escolha, pronto para interagir com a biblioteca.
- **Analisando a Escolha:** O portal analisa a escolha do usuário, como um decifrador de códigos.
    - **Inserir Livro:** Chama a função inserirLivro para adicionar o novo livro à árvore.
    - **Buscar Livro:** Chama a função buscarLivro para encontrar o livro desejado.
    - **Remover Livro:** Chama a função removerLivro para excluir o livro selecionado.
    - **Listar Livros:** Chama a função listarLivros para apresentar a lista completa de livros.
    - **Verificar se a Árvore Está Vazia:** Verifica se a árvore está sem livros e informa o resultado ao usuário.
    - **Sair:** Fecha o portal e encerra o programa, finalizando a interação.
- **Loop Infinito:** O portal se mantém aberto, pronto para receber novos comandos do usuário, até que a escolha seja "Sair".

- **Fechando o Portal:** Quando o usuário escolhe "Sair", o portal se fecha educadamente, **encerrando o programa**.

Exemplo:

Imagine o usuário digitando "Inserir Livro". O portal analisa a escolha, chama a função inserirLivro e solicita as informações do livro ao usuário. Após receber as informações, a função insere o livro na árvore e o portal retorna ao menu, aguardando o próximo comando.

9.2 Importância da Função main:

- **Interação com o Usuário:** Permite que o usuário **interaja com a árvore binária de busca** de forma amigável e intuitiva.
- **Gerenciamento da Biblioteca:** Oferece funcionalidades para **gerenciar a coleção de livros**, como adicionar, buscar, remover e listar livros.
- **Controle do Programa:** Permite ao usuário **controlar o fluxo do programa**, escolhendo as operações que deseja realizar e encerrando-o quando desejar.

Em resumo a função main é a **alma interativa** da biblioteca binária de busca, conectando o usuário à vasta coleção de livros e permitindo que ele explore, organize e gerencie a biblioteca de forma eficiente e prazerosa. Através do menu de opções e do laço while infinito, ela se torna um portal sempre aberto, pronto para atender às necessidades do usuário e garantir uma experiência de leitura enriquecedora.

10. Conclusão

O sistema **ÁRVORES: Gerenciamento de Livros para Biblioteca** representa um avanço significativo na gestão de acervos, oferecendo uma solução eficiente, organizada e acessível para bibliotecas de todos os portes. Sua implementação contribui para a otimização dos serviços bibliotecários e a promoção do acesso à informação para toda a comunidade acadêmica.