GOVERNO DO ESTADO DO PIAUÍ UNIVERSIDADE ESTADUAL DO PIAUÍ
CAMPUS PROFESSOR ANTÔNIO GIOVANNI ALVES DE SOUSA
CURSO DE BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO
DISCIPLINA: Estrutura de Dados
ALUNO: Eduardo Alves de Sousa

1. Introdução

A estrutura de dados desempenha um papel importante e fundamental no requisito desenvolvimento de software, influenciando diretamente o desempenho e principalmente a eficiência das operações realizadas sobre os dados fornecidos nos algoritmos. Uma escolha cuidadosa da estrutura é essencial para atender aos requisitos específicos de cada aplicação que venha a ser escolhida dependendo de cada caso. Neste contexto, a árvore binária de busca surge como uma opção versátil para operações de inserção, busca e remoção, proporcionando uma ordenação automática dos elementos inseridos.
Este trabalho apresenta uma análise crítica da eficiência da árvore binária de busca, destacando principalmente seus pontos fortes e identificando áreas que podem ser aprimoradas. O código-fonte implementado em linguagem C é explorado em detalhes, considerando sua clareza e fácil entendimento para melhor compreensão do código em si.

2. Operações básicas na árvore binária

- Inserir um elemento na árvore.
- Remover um elemento.
- Procurar por um elemento.
- Excluir um elemento da árvore.

3. Eficiência da Estrutura Escolhida (Árvore Binária de Busca)

- **Pontos Fortes**
  - **Inserção e Busca Eficientes:** A inserção e busca em uma árvore binária de busca têm uma complexidade média de O(log n) em árvores balanceadas, o que é eficiente, onde n é o número de nós da árvore.
  - **Ordenação Automática:** A árvore binária de busca mantém todos os elementos de forma ordenada automaticamente, para que assim tenha uma maior facilidade na busca ordenada de seus valores e uma melhor compreensão pela busca de algum valor específico na árvore.
- **Pontos Fracos**
  - **Desbalanceamento:** Se a árvore não for balanceada, pode até mesmo acarretar em uma lista ligada, ou seja levando a uma pior complexidade de tempo (O(n)) onde n é o número de nós da árvore.
  - **Remoção de Nós com Dois Filhos:** A remoção de um nó com dois filhos envolve encontrar o sucessor in-order, o que pode ser custoso e levar a operações demoradas.

- Seja uma árvore binária de busca de n nós e de altura h. A inserção precisa localizar o local para fazer a inserção.
    - **A complexidade de inserção é O(h).**
    - **No pior caso, a altura de uma árvore binária de busca pode ser O(n).**

4. Código-fonte da aplicação

- **Pontos Positivos**
    - **Clareza:**
        - O código é relativamente claro e fácil de entender, com boas práticas de nomenclatura e estruturação de código.
    - **Modularidade:**
        - A divisão em funções facilita a compreensão e manutenção do código.

4.1. Estrutura do código

- **Estrutura de Nó (struct Node):**
    - Define a estrutura de um nó na árvore binária de busca. Cada nó contém uma chave (key) e ponteiros para os nós à esquerda (left) e à direita (right).
- **Função para Criar um Novo Nó (createNode):**
    - Aloca dinamicamente memória para um novo nó, inicializa a chave e os ponteiros esquerdo e direito como nulos.
- **Função para Inserir um Nó na Árvore (insert):**
    - Insere um novo nó na árvore. Se a árvore estiver vazia, cria um novo nó. Caso contrário, percorre a árvore para encontrar o local apropriado para inserção.
- **Função para Encontrar o Nó com a Chave Mínima (findMin):**
    - Encontra o nó com a chave mínima em uma subárvore, percorrendo continuamente os nós mais à esquerda até encontrar o último nó.
- **Função para Remover um Nó da Árvore (removeNode):**
    - Remove um nó com uma chave específica da árvore. Manipula casos de remoção para garantir que a propriedade da árvore binária de busca seja mantida.
- **Função para Buscar um Nó na Árvore (search):**
    - Realiza uma busca na árvore para encontrar um nó com uma chave específica.
- **Função para Impressão em Ordem (inorderTraversal):**
    - Realiza uma travessia em ordem na árvore, imprimindo os nós em ordem crescente.
- **Função Principal (main):**

    - Implementa um menu interativo que permite ao usuário realizar operações como inserir, remover, buscar e imprimir a árvore em ordem e até mesmo ver qual a altura da árvore de acordo com os valores fornecidos.

5. Resultados Obtidos nas Análises de Desempenho

- **Pontos Positivos**
    - **Inserção e Busca Eficientes em Árvores Balanceadas**
    - Se a árvore for mantida balanceada, as operações de inserção e busca devem se manter de forma eficiente.
- **Pontos a Melhorar**
    - **Testes em Cenários Extremos**
    - É importante realizar testes em cenários extremos para avaliar como a árvore se comporta em situações de extremo limite, especialmente em termos de desbalanceamento onde sua complexidade se torna muito mais complexa de se analisar.

6. Conclusão

A árvore binária de busca é uma estrutura bastante versátil, pois proporciona com bastante eficiência as operações com uma boa gestão do balanceamento da árvore. Vale ressaltar que entender sua complexidade de maneira mais completa devemos utilizar valores e uma estrutura mais complexa para melhor compreensão do poder do algoritmo que é a árvore binária de busca, deve ser compreendido principalmente suas características e limitações é extremamente essencial para uma boa escolha informada em cenários específicos. A implementação da árvore binária de busca em C apresenta uma solução eficiente para operações de inserção, remoção e busca, proporcionando ordenação automática dos elementos inseridos pelo usuário.

Referências

- Introduction to binary tree - data structure and algorithm tutorials. (2013, March 31). GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-binary-tree-data-structure-and-algorithm-tutorials/?ref=lbp
- Properties of binary tree. (2015, May 5). GeeksforGeeks. https://www.geeksforgeeks.org/properties-of-binary-tree/?ref=lbp
- Applications, advantages and disadvantages of binary tree. (2022, May 25). GeeksforGeeks. https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-binary-tree/?ref=lbp
- Insertion in a Binary Tree in level order. (2017, August 29). GeeksforGeeks. https://www.geeksforgeeks.org/insertion-in-a-binary-tree-in-level-order/?ref=lbp
- Deletion in a binary tree. (2017, August 31). GeeksforGeeks. https://www.geeksforgeeks.org/deletion-binary-tree/?ref=lbp
- Feofiloff, P. (n.d.). Como inserir e remover um nó de uma árvore binária de busca. Usp.Br. Retrieved November 23, 2023, from https://www.ime.usp.br/~pf/algoritmos/aulas/binst.html
- Stumm, V., Jr. (2015, January 19). Árvore Binária de Busca em Python. Python Help. https://pythonhelp.wordpress.com/2015/01/19/arvore-binaria-de-busca-em-python/