Sistema de Cadastro e Manipulação de Cursos e Disciplinas utilizando Árvores Binárias e Árvores AVL em C

Bruna Heloisa Feitosa Veloso

Resumo do projeto: O projeto visa proporcionar uma aplicação eficiente e funcional para o cadastro e manipulação de informações sobre cursos e disciplinas. As estruturas de árvores binárias e árvores AVL são utilizadas para organizar os dados e garantir uma operação ágil. O experimento consiste em avaliar o desempenho das diferentes estruturas e espera-se aprender sobre os benefícios do uso de estruturas balanceadas e otimizar o programa para melhor atender às necessidades dos usuários.

1. Introdução

Este projeto consiste na implementação de um programa em linguagem C que visa o cadastro e manipulação de informações sobre cursos e disciplinas, utilizando estruturas de árvores binárias e árvores AVL. Através desse programa, é possível realizar operações como impressão dos dados dos cursos, busca de cursos e disciplinas, exclusão de disciplinas e cursos, entre outras funcionalidades.

O paradigma utilizado nesse projeto é o da programação estruturada, onde o programa é organizado em funções e estruturas de dados para garantir uma melhor organização e modularização do código. A linguagem de programação C foi escolhida por sua eficiência e baixo nível de abstração, possibilitando um controle preciso sobre a alocação e manipulação de memória.

O projeto está organizado de forma a separar as informações dos cursos e das disciplinas em estruturas de árvores binárias, onde cada nó da árvore representa um curso ou uma disciplina. Além disso, é utilizado o conceito de árvores AVL, que são árvores binárias balanceadas que garantem uma altura máxima logarítmica e, consequentemente, uma melhor eficiência nas operações de busca, inserção e remoção.

O programa permite ao usuário interagir através de um menu, onde é possível escolher as operações desejadas, como imprimir os dados dos cursos, buscar cursos e disciplinas, excluir disciplinas e cursos, entre outras. O programa também realiza a medição de tempos de execução para inserção de elementos na árvore de cursos e busca de cursos na árvore, utilizando funções da biblioteca time do C.

No decorrer deste trabalho, serão apresentados os detalhes do programa, desde a implementação das estruturas de dados até as funcionalidades disponíveis para o usuário. Além disso, serão abordadas as análises de desempenho realizadas, comparando os tempos de execução das operações em diferentes estruturas de árvore. Por fim, será apresentada a adaptação do projeto para a utilização de árvores AVL, destacando as melhorias obtidas em relação à eficiência das operações.

Com isso, será possível introduzir os objetivos do projeto, observar os paradigmas e tecnologias utilizados, bem como a organização e estrutura do programa. As próximas seções fornecerão detalhes mais aprofundados sobre as implementações e análises realizadas, permitindo uma compreensão completa do projeto e de seus resultados.

2. Seções Específicas

2.1. Estruturas de Dados e Algoritmos Utilizados

Nesta seção, serão apresentadas as estruturas de dados utilizadas no programa, como a árvore binária para o cadastro de cursos e disciplinas, e a árvore AVL para garantir o balanceamento das árvores. Serão descritos os algoritmos de inserção, busca e remoção implementados para cada uma dessas estruturas, destacando suas características e complexidades.

O programa desenvolvido utiliza duas principais estruturas de dados: a árvore binária e a árvore AVL. Essas estruturas são responsáveis pelo armazenamento e organização dos cursos e disciplinas cadastrados.

2.1.1 Árvore Binária

A árvore binária é utilizada para cadastrar os cursos e disciplinas. Cada nó da árvore representa um curso ou uma disciplina, contendo informações como código, nome, quantidade de blocos e número de semanas. A árvore é organizada pelo código do curso ou disciplina, garantindo que não haja repetições.

Os algoritmos implementados para a árvore binária incluem:

- Inserção: O algoritmo de inserção verifica se o código do curso ou disciplina já existe na árvore, caso contrário, cria um novo nó com as informações fornecidas e o insere na posição correta com base no código.
- Busca: O algoritmo de busca percorre a árvore binária, comparando o código do curso ou disciplina com o código do nó atual. Caso o código seja encontrado, as informações são retornadas. Caso contrário, o algoritmo continua a busca nos nós filhos até encontrar o código desejado ou percorrer toda a árvore sem sucesso.
- Remoção: O algoritmo de remoção busca o nó com o código do curso ou disciplina a ser removido. Se o nó for encontrado, são realizadas as operações necessárias para manter a estrutura da árvore, como reorganização dos ponteiros. Caso o nó possua filhos, são aplicadas regras específicas para manter a integridade da árvore.

2.1.2 Árvore AVL

A árvore AVL é uma árvore binária balanceada, onde a diferença de altura entre as subárvores esquerda e direita de cada nó é no máximo 1 e no mínimo -1. Essa estrutura garante uma altura máxima logarítmica e otimiza o tempo de execução das operações de busca, inserção e remoção.

Os algoritmos implementados para a árvore AVL são semelhantes aos da árvore binária, com a diferença de que, após cada operação de inserção ou

remoção, é realizado um balanceamento para manter a propriedade de altura balanceada. Esses algoritmos incluem:

- Inserção AVL: Após a inserção de um novo nó na árvore AVL, são realizadas rotações e ajustes nos nós para garantir o balanceamento. As rotações podem ser simples (Esquerda e Direita) ou duplas (EsquerdaDireita e DireitaEsquerda) e são aplicadas de acordo com as características da árvore.
- Remoção AVL: Após a remoção de um nó na árvore AVL, também são realizadas rotações e ajustes para manter o balanceamento. Assim como na inserção, as rotações simples e duplas são aplicadas com base nas propriedades da árvore.

Esses algoritmos garantem que as operações de inserção, busca e remoção sejam eficientes e que a árvore AVL se mantenha balanceada, proporcionando um desempenho superior em relação à árvore binária tradicional.

2.2. Menu Interativo e Funcionalidades

Esta seção detalha a implementação do menu interativo, que permite ao usuário selecionar as operações desejadas. Serão descritas as opções disponíveis, como a impressão dos dados dos cursos, busca de cursos e disciplinas, exclusão de disciplinas e cursos, entre outras funcionalidades. Para cada funcionalidade, será fornecida uma descrição do algoritmo utilizado, incluindo chamadas de sistemas e tecnologias envolvidas.

O programa implementado conta com um menu interativo que oferece diversas funcionalidades para o usuário. A seguir, serão detalhadas as principais opções disponíveis e as operações realizadas em cada uma delas:

- Impressão dos dados dos cursos
  Essa funcionalidade permite imprimir os dados de todos os cursos cadastrados. O algoritmo percorre a árvore binária de cursos e realiza a impressão das informações de cada nó, garantindo que os dados sejam exibidos em ordem crescente pelo código do curso.
- Busca de cursos e disciplinas
  Nessa opção, o usuário pode buscar informações sobre um curso ou uma disciplina específica. O algoritmo percorre a árvore binária de cursos ou disciplinas, comparando o código fornecido pelo usuário com os códigos dos nós da árvore. Caso seja encontrado um nó correspondente, as informações são exibidas.
- Exclusão de disciplinas e cursos:
  Essa funcionalidade permite excluir disciplinas ou cursos do sistema. O algoritmo busca o nó correspondente na árvore binária e realiza a exclusão, mantendo a estrutura da árvore. No caso da exclusão de cursos, é necessário verificar se existem disciplinas cadastradas para aquele curso antes de realizar a exclusão.

- Outras funcionalidades:
  O programa também oferece opções adicionais, como imprimir os cursos com a mesma quantidade de blocos informada pelo usuário, imprimir as disciplinas de um determinado bloco de um curso, imprimir todas as disciplinas de um curso com a mesma carga horária, entre outras.

O menu interativo e as funcionalidades implementadas permitem uma interação fácil e intuitiva do usuário com o programa, proporcionando acesso rápido às informações dos cursos e disciplinas. As tecnologias utilizadas garantem a eficiência e o correto funcionamento das operações, proporcionando uma experiência satisfatória ao usuário.

2.3. Medição de Tempos de Execução

Será abordada nesta seção a implementação da medição de tempos de execução para as operações de inserção na árvore de cursos e busca de cursos na árvore. Serão detalhados os comandos utilizados para obter o tempo inicial e final de cada operação, bem como a obtenção da média dos tempos para resultados mais precisos. Será discutida a importância dessa medição para avaliar a eficiência do programa e comparar os desempenhos das diferentes estruturas de árvores utilizadas.

2.4. Adaptação para Árvores AVL

Esta seção descreve a adaptação do projeto para a utilização de árvores AVL. Será explicado como a estrutura de árvore AVL foi implementada, destacando as alterações necessárias em relação à árvore binária e as melhorias obtidas em termos de eficiência. Serão apresentados os algoritmos específicos para as operações de balanceamento da árvore AVL.

Para implementar a árvore AVL, foram realizadas algumas alterações em relação à estrutura da árvore binária. Cada nó da árvore AVL possui, além dos campos padrão, um campo adicional que armazena a altura do nó. Essa informação é utilizada para verificar e corrigir o balanceamento da árvore.

Os algoritmos específicos para as operações de balanceamento da árvore AVL são responsáveis por manter a altura máxima da árvore logarítmica, garantindo que as operações sejam executadas em tempo eficiente. Os principais algoritmos utilizados são:

- Fator Balanceamento: O fator de balanceamento é uma informação associada a cada nó da árvore AVL e indica a diferença entre as alturas do filho esquerdo e do filho direito desse nó. O valor do fator de balanceamento pode ser -1, 0 ou 1, representando um estado balanceado. Caso o fator de balanceamento seja diferente desses valores, indica um desequilíbrio na árvore. Ele pode ser calculado a partir da altura dos filhos de um nó. Após a inserção ou remoção na árvore AVL, é necessário verificar se o fator de balanceamento dos nós foi alterado e, caso tenha ocorrido um desequilíbrio, realizar as rotações apropriadas para reequilibrar a árvore.

- Rotações Simples: As rotações simples são realizadas para corrigir desequilíbrios em um nó e são classificadas em rotação à esquerda e rotação à direita. Essas rotações são utilizadas quando ocorre um desequilíbrio em um dos filhos do nó, causado por uma inserção ou remoção.
- Rotações Duplas: As rotações duplas são combinações de rotações simples e são utilizadas quando ocorre um desequilíbrio em um dos filhos e em seu neto. Elas são realizadas em duas etapas para corrigir o desequilíbrio e manter a estrutura balanceada.
- Alterar Altura: Além do fator de balanceamento e das rotações simples e duplas, outro algoritmo importante utilizado na árvore AVL é o algoritmo de alteração de altura. Esse algoritmo é responsável por atualizar a altura dos nós da árvore após a inserção ou remoção de um elemento.Após a realização de uma operação de inserção ou remoção, é necessário percorrer o caminho percorrido pela operação, a partir do nó afetado até a raiz da árvore, atualizando a altura de cada nó ao longo do caminho. Esse processo é realizado verificando a altura dos filhos de cada nó e atualizando a altura do nó pai de acordo com a altura máxima entre seus filhos.

Ao realizar uma inserção ou remoção na árvore AVL, os algoritmos de balanceamento são acionados para verificar e corrigir possíveis desequilíbrios. Essa correção ocorre percorrendo o caminho percorrido pela inserção ou remoção, atualizando as alturas dos nós e realizando as rotações necessárias.

A utilização da árvore AVL no projeto traz melhorias significativas em relação à eficiência das operações. O balanceamento automático da árvore evita a ocorrência de casos extremos, como uma árvore desbalanceada que levaria a piora no desempenho das operações.

Resultados da Execução do Programa

Nesta seção, serão apresentadas as análises de desempenho realizadas, comparando os tempos de execução das operações nas diferentes estruturas de árvores. Serão discutidas as implicações dos resultados obtidos e as vantagens de se utilizar árvores AVL em relação à eficiência das operações de busca e inserção.

É realizado um teste de desempenho para medir o tempo de inserção de elementos nas árvores de cursos e disciplinas. O teste é importante para avaliar a eficiência das implementações das funções de inserção (inserirCurso, inserirCursoAVL, inserirDisciplina e inserirDisciplinaAVL).

A abordagem do teste consiste em criar um conjunto de números representando os cursos e disciplinas, embaralhar esses números e em seguida realizar a inserção dos elementos nas árvores. A ideia é simular uma situação em que um grande número de cursos e disciplinas precisa ser inserido nas árvores, permitindo avaliar o desempenho das operações de inserção.

Os passos do teste são os seguintes:

1. **Criação dos números de cursos e disciplinas:** São criados dois arrays, numerosCursos e numerosDisciplinas, contendo os números de identificação dos cursos e disciplinas, respectivamente. Os números são preenchidos em ordem crescente.
2. **Embaralhamento dos números:** A função embaralhar é utilizada para embaralhar os elementos dos arrays numerosCursos e numerosDisciplinas. Isso garante que os elementos sejam inseridos de forma aleatória nas árvores, simulando uma situação mais realista.
3. **Criação do arquivo de saída:** É criado um arquivo chamado "numeros.txt" para registrar os números dos cursos e disciplinas antes de serem inseridos nas árvores. Os números são escritos em arquivos e separados por espaços.
4. **Inserção dos elementos nas árvores:** São realizadas as operações de inserção dos elementos nas árvores de cursos e disciplinas. Para cada número do array numerosCursos ou numerosDisciplinas, é criado um objeto do tipo Curso ou Disciplina e é realizado a inserção na árvore correspondente.
5. **Medição do tempo de inserção**: São utilizadas as funções clock() para medir o tempo de início e fim da inserção dos elementos nas árvores. Em seguida, é calculado o tempo decorrido em segundos.

Ao final do teste, teremos os tempos de inserção para as árvores de cursos e disciplinas, tanto para as implementações de árvores de busca binária quanto para as implementações de árvores AVL. Esses tempos permitem avaliar o desempenho das operações de inserção nas diferentes estruturas e comparar a eficiência das implementações.

Essa abordagem de teste de desempenho é importante para identificar possíveis ineficiências nas implementações das árvores e buscar maneiras de otimizar o código, caso necessário.

Os resultados dos testes de inserção demonstraram claramente os benefícios da árvore AVL em termos de eficiência. Mesmo com um grande número de elementos a serem inseridos, a árvore AVL manteve um tempo de execução relativamente baixo em comparação com a árvore binária simples. Isso ocorre devido ao balanceamento automático que ocorre após cada operação de inserção na árvore AVL, garantindo que a altura da árvore seja mantida em um nível ótimo.

Além disso, os testes também mostraram que a árvore AVL é especialmente eficiente quando há inserções frequentes em tempo real. Mesmo com um fluxo contínuo de novos elementos sendo inseridos na árvore, o tempo de execução das operações de inserção permaneceu estável e previsível, sem degradação significativa no desempenho.

Esses resultados indicam que a estrutura balanceada da árvore AVL permite uma manipulação mais rápida e eficiente dos dados durante as operações de inserção. Isso não apenas beneficia o desempenho do sistema, mas também proporciona uma experiência mais fluida ao usuário, uma vez que as operações de inserção são concluídas de forma eficiente, sem atrasos perceptíveis.

Conclusão

O projeto em questão apresentou a implementação de um programa em linguagem C para o cadastro e manipulação de cursos e disciplinas, utilizando estruturas de árvores binárias e árvores AVL. O objetivo principal do experimento foi avaliar o desempenho e a eficiência das diferentes estruturas de árvores utilizadas, bem como verificar se o propósito do programa, que é possibilitar o cadastro e manipulação dos dados de forma eficiente, foi alcançado.

A realização dos experimentos também proporcionou aprendizados significativos. Foi possível compreender a importância da medição de tempos de execução para avaliar o desempenho de um programa, demonstrou a importância de ter uma visão quantitativa e ajudou a identificar oportunidades de melhorias. Além disso, foi mostrada a importância de escolher a estrutura de dados adequada para cada situação. A árvore AVL mostrou-se particularmente eficiente para o cenário de cadastro e manipulação de cursos e disciplinas, proporcionando uma experiência mais fluida ao usuário e facilitando a manutenção e organização dos dados.

Dessa forma, o projeto reforçou a importância de buscar constantemente soluções otimizadas e eficientes no desenvolvimento de programas, levando em consideração não apenas a funcionalidade, mas também o desempenho e a eficiência das estruturas de dados utilizadas.

Referências

TENENBAUM, A. M. et al., Estruturas de Dados Usando C. Ed. 1. Editora Pearson. 1995.