Comparativo de tempos de execução de Cadastro de Curso e Disciplinas entre Árvores Binárias de Busca e AVL

João dos Santos Neto
Universidade Federal do Piauí - Picos

Resumo

Foi requisitado um sistema de cadastro de cursos e disciplinas em duas estruturas de dados, Árvores Binárias de Busca e Árvores AVL, onde Curso e Disciplinas são essas estruturas. Esta implementação permite que o usuário realize funções como a inserção, visualização e remoção dos cursos e disciplinas. A utilização dessas estruturas permite uma maior eficiência na execução dessas funções citadas.

Palavras-chave: inserção, remoção, busca, ABB, AVL.

1. Introdução

O relatório demonstra aspectos de implementação de Árvores Binária de Busca(ABB) e Árvores AVL em um Sistema de Cadastro de Curso e Disciplinas. Tendo como foco permitir que o sistema insira, remova e busque de forma eficiente os cursos e disciplinas.

Existem diversas árvores binárias, mas em questão, foram requisitadas apenas duas ABB e AVL. Essas árvores compartilham o mesmo aspecto de organização, um nó que possui referências para os nós seguintes,onde de forma de organizar a os elementos é feito: os elementos menores sejam ajustado a esquerda, e os maiores a direita,porém a diferença presente entre elas está apenas no balanceamento, essa solução foi precisa pois na ABB os elementos podem deixar a árvore pendentes para um lado dela(esquerda ou direita),a ABB é uma estrutura de dados não balanceada, porém a AVL possui esse aspecto.

2. Metodologia

A metodologia presente para a implementação foi feita da seguinte forma: Definir o escopo de cada função e estrutura requisitada e separação de arquivos.

3. Definição das estruturas de dados

A fim de organização do sistema, foram gerados arquivos .c e .h, tanto para ABB e AVL. Esses arquivos possuem o nome curso.c disciplina .c curso.h disciplina.h. Dos arquivos .c são as implementações das funções e estruturas, já os arquivos .h são as definições dessas funções e estruturas.

Para implementação do código foi utilizada a Linguagem C, onde a definição das estruturas de Curso e Disciplinas:

1. Árvore Binária de Busca
   - Curso: código do curso,nome do curso, quantidade de blocos, número de semanas e suas respectivas disciplinas(esta é uma referência para a árvore de disciplinas).
   - Disciplina: código da disciplina, nome da disciplina, bloco, e carga horária.
2. AVL
   - Curso: código do curso,nome do curso, quantidade de blocos, número de semanas , altura e suas respectivas disciplinas(esta é uma referência para a árvore de disciplinas).
   - Disciplinas: código da disciplina, nome da disciplina, bloco, altura e carga horária.

4. Implementações das funções

Para implementação foi utilizada a linguagem C, diversas funções foram feitas para alcançar o objetivo fornecido.Abaixo estão todas as funções e seus respectivos objetivos,vale ressaltar que estas atendem tanto para ABB e AVL:

1. showBlocks: função que tem a finalidade de mostrar todos os cursos com a mesma quantidade de blocos.
2. inserCourse: essa função de gerar um novo nó(curso) e inseri-lo na árvore.
3. insertTree: essa função procura o lugar ideal para inserir o nó dentro da árvore.
4. searchCourse: essa função tem a finalidade de procurar um nó da árvore e exibir seus dados, caso exista.
5. printInOrdem: essa função irá mostrar todos os elementos da estrutura em ordem crescente pelo ID do curso.
6. showCursoDisciplinaCod: essa função mostra todas as disciplinas de um determinado curso, sabendo o ID do curso.
7. printCursoDisciplina: essa função iŕa mostrar os dados de uma determinada disciplina de um determinado curso, sabendo os respectivos IDs.
8. printAllDiscipline: essa função exibe na tela toda a árvore de disciplinas de um determinado curso em um determinado bloco.
9. printCursoDisciplinaWorkload: exibe na tela todas as disciplinas de um curso que tem a mesma carga horária.
10. removeAvrCursoDisciplina: essa função remove toda a árvore de disciplina de um curso.
11. create: essa inicializa a estrutura.
12. createNode: essa função cria um nó alocando seu espaço.
13. searchNode: essa função procura por um nó na estrutura através do seu ID.
14. ehFolhaCurso e ehFolha: essas funções tem o objetivo de verificar se o nó encontrado não possui nenhuma referência.
15. umFilhoCurso e umFilho: verificam se o nó tem apenas uma referência.
16. doisFilhosCurso e doisFilho: verificam se o nó possui duas referências.
17. buscaMaiorEsqCurso e buscaMaiorEsq: função tem a finalidade de buscar o maior elementos na árvore pela esquerda do nó que irá ser removido.
18. removerCursoDiscipline: remove a disciplina de um curso.
19. removeCurso: remove um curso caso este não possua uma árvore de disciplinas.
20. ehFolhaIntCurso e ehFolhaInt: estas verificam com verdadeiro ou falso se o nó não possui nenhuma referência.
21. umFilhoIntCurso e umFilhoInt: estas verificam com verdadeiro ou falso se o nó possui apenas uma referência.
22. doisFilhoIntCurso e doisFilhoInt: estas verificam com verdadeiro ou falso se o nó possui duas referência.

Para fornecer interatividade com o usuário, foi determinada uma seção no código que tem a respectiva finalidade.

Dessa interatividade encontra-se:

- Exibir a Árvore de Cursos.
- Exibir um curso.
- Exibir todos os cursos com o mesmo número de blocos.
- Exibir a árvore de disciplinas de um curso.
- Exibir disciplina de um curso.
- Exibir todas as disciplinas de um curso de um bloco.
- Exibir todas as disciplinas com a mesma carga horária.
- Excluir uma disciplina de um curso.
- Excluir um curso.
- Excluir toda a árvore de disciplina de um curso.
- Inserir um curso.

5. Resultados da Execução do Programa

Para verificar o desempenho do programa, foram gerados 3 arquivos que contêm 50000 números de 1 a 50000 em ordem crescente, decrescente e desordenado. Estes foram inseridos nas estruturas e resultou nos tempos,em microssegundo, abaixo:

Média de tempo: Entrada de Dados X Estrutura de Dado	Árvore ABB	Árvore AVL
Crescente	Média de tempo: 48280.000000 µs	Média de tempo: 122080.000000 µs
Aleatória	Média de tempo:40.000000 µs	Média de tempo: 202140.000000 µs
Decrescente	Média de tempo: 40820.000000 µs	Média de tempo: 129460.000000 µs

Pode-se perceber que na inserção Aleatória ABB é bastante rápida, pois a mesma não dá prioridade no balanceamento dos elementos. Agora, vejamos como ambas se saem em relação à busca de um mesmo elemento, para verificação o mesmo elementos foi procurado 30 vezes a fim de resultar na média de desempenho:

Tempo de busca de um elemento : Entrada de dados X Estrutura de Dado	Árvore ABB	Árvore AVL
Crescente	Tempo de busca: 2189.566667 µs	Tempo de busca: 0.266667 µs
Aleatória	Tempo de busca: 0.566667 µs	Tempo de busca: 0.366667 µs
Decrescente	Tempo de busca: 2114.733333 µs	Tempo de busca: 0.200000 µs

6. Conclusão

Bem como citado o propósito do sistema fornecido, percebe-se que algumas características de funcionalidade como busca e inserção estão presentes nas estruturas estudadas. Da funcionalidade de inserção é notável que a ABB é uma estrutura de dado ideal para o sistema, porém em funcionalidade de busca essa característica se encaixa perfeitamente na AVL, que possui um tempo de busca constante.

Das estruturas fornecidas, o sistema de acordo com os tempo de execução se encaixam perfeitamente aos requisitos de funcionalidade, é claro que existem outras estruturas de dados mais eficiente e mais complexas que podem substituí-las.

Pode-se concluir que de acordo com a implementação feita, o objetivo de cada estruturas, vemos que a Árvore Binária de Busca possui um tempo de inserção mais rápido, pois está apenas insere, já a Árvore AVL tem a diferença que insere e balanceia todos os nós da árvore, isso torna mais eficiente na busca pois os elementos estão mais centralizados, porém o custo de tempo de inserção é maior.

Ambas apresentam suas vantagens e desvantagens, isso pode depender muito da entrada de dados fornecida. Em entradas ordenadas, a ABB é ineficiente, pois esta retorna a aspectos de uma lista encadeada, porém para entrada aleatória ela é bastante eficiente, por outro lado com sua inserção eficiente a sua busca é demorada, pois os elementos podem estar pendentes para um lado.

Por esse motivo que AVL se torna uma árvore mais complexa, pelo fato de que seus nós estão balanceados, oque possibilita uma busca mais rápida por qualquer elemento, tornando seu tempo quase constante.