Atividade 2 de Árvore AVL
(mesmo grupo da atividade anterior)

Observação: As instruções sobre entrega, prazo e avaliação estão nas páginas 2-3 do documento.

Objetivo: Implementar operações de inserção e remoção de nós em uma árvore AVL em C++.

Agora que você implementou o cálculo do fator de balanceamento para cada nó de uma AVL e as operações de rotação, a ideia dessa segunda atividade de árvore AVL é implementar as operações de inserção e remoção de nós em uma árvore AVL.

TEORIA

Lembre-se que podemos verificar se uma árvore está desbalanceada consultando o fator de balanceamento do nó raiz de uma subárvore e dos seus filhos.

Adaptado do enunciado da atividade anterior, temos:

1. **Rotação para esquerda:**
   - Desbalanceamento na subárvore direita do filho direito.
   - Árvore direita-direita: fb(raiz_subárvore) = +2 e fb(filho_direito) = +1

2. **Rotação direita-esquerda ou dupla a esquerda:**
   - Desbalanceamento na subárvore esquerda do filho direito.
   - Árvore direita-esquerda: fb(raiz_subárvore) = +2, fb(filho_direito) = -1

3. **Rotação para direita:**
   - Desbalanceamento na subárvore esquerda do filho esquerdo.
   - Árvore esquerda-esquerda: fb(raiz_subárvore) = -2, fb(filho_esquerdo) = -1

4. **Rotação esquerda-direita ou dupla a direita:**
   - Desbalanceamento na subárvore direita do filho esquerdo.
   - Árvore esquerda-direita: fb(raiz_subárvore) = -2, fb(filho_esquerdo) = +1

Lembre-se também que, após uma alteração na árvore, seja por meio de uma inserção, remoção ou rotação, o fator de balanceamento dos nós devem ser atualizados.

EXERCÍCIOS

(1) Implemente a operação de inserção de nós em uma árvore AVL.

(2) Implemente a operação de remoção de nós em uma árvore AVL.

(3) Para testar o seu código e mostrar que a implementação está funcionando, na main, construa as seguintes árvores (uma árvore por item):
a) Inserir, nessa ordem, os nós 1, 2 e 3.
b) Inserir, nessa ordem, os nós 3, 2 e 1.

c) Inserir, nessa ordem, os nós 3, 1 e 2.
d) Inserir, nessa ordem, os nós 1, 3 e 2.
e) Inserir, nessa ordem, os nós 5, 4, 3, 1, 2, 6, 7, 9 e 8.
f) Remover o nó 4 da árvore do item (e).
g) Remover o nó 5 da árvore do item (f).
h) Remover o nó 1 da árvore do item (g).

Observações:

- Como a AVL é uma BST balanceada, você pode aproveitar a sua implementação da árvore BST e modificar o código para que, após a inserção/remoção de um nó, seu código verifique se há a necessidade de balancear a árvore (em caso positivo, o balanceamento é feito com uma das rotações apresentadas)
- Considere que os nós da AVL usam um número inteiro como chave (valor do nó).
- Para cada item do exercício 3, após a construção da árvore, seu código deve exibir os dados atualizados de todos os nós (pelo menos quem é o pai, o filho esquerdo, o filho direito e o fator de balanceamento). A não exibição dessas informações reduzirá a nota final da atividade.
- Nessa atividade, o seu código já deve fazer a rotação correta, balanceando a árvore.

Desenvolvimento (10,0 pontos)

- Sua solução deve ser escrita apenas com a linguagem C++, sem uso da STL.
- Declare as suas classes em headers (arquivos .h) e implemente-as nos arquivos .cpp.
- Lembre-se do include guard do header.
- Para alocação dinâmica, use new e delete (lembre-se que, para arrays, o delete inclui o [] antes do nome do ponteiro a ser liberado).
- A correta execução do código da main também entra como critério de avaliação (haverá desconto na nota caso falte exibir as informações pedidas no item 4 do enunciado).

Identificação e referências

- Coloque sua identificação - nome e TIA - no início de cada arquivo de código, como comentário (use // no começo de cada linha que queira comentar).
- Inclua como comentário quaisquer referências (livros, artigos, sites, entre outros) usadas para solucionar o problema.

Entrega

- **Código:** Compacte todos os arquivos .cpp/.h ou o projeto completo criado na IDE que você está usando (mas sem os intermediários como bin e obj) no formato zip OU comite todos os arquivos .cpp/.h ou o projeto completo criado na IDE que você está usando (mas sem os intermediários como bin e obj) em um repositório git.
- **Arquivo texto (.txt):**
  - Se o código está em um repositório git, envie um arquivo txt no Moodle contendo sua identificação e o link do repositório.
- **Prazo de entrega:** via link do Moodle até 19/11/2021 23:59.

Informações importantes sobre critérios de avaliação

Embora essa atividade seja uma avaliação da disciplina, sempre considero que as atividades também podem ser usadas para nos acostumarmos com o mercado de trabalho. Portanto, leve em consideração os seguintes critérios que vou aplicar na avaliação:

- Será descontado 1,0 (um) ponto caso a entrega não respeite o enunciado.
  Exemplos:
  - O enunciado pede para enviar um arquivo compactado no formato zip, mas o arquivo enviado está no formato rar.
  - O enunciado pede um arquivo texto no formato txt, mas foi enviado um documento do Word.
  - Não há identificação nem referências (caso aplicável) nos arquivos de código.
- Será descontado 1,0 (um) ponto caso o arquivo zip OU o repositório git contenha pastas e arquivos desnecessários.
  Exemplo:
  - Pastas intermediárias criadas no processo de compilação (Debug, obj, bin, ...).
- O projeto deve ser desenvolvido em linguagem C++ e não em linguagem C. Caso a solução apresentada use funcionalidades da linguagem C e que tenham equivalentes em C++, será descontado 2,0 (dois) pontos.
  Atente-se a esse detalhe quando estiver pesquisando e verificando exemplos na internet e outros materiais, principalmente de assuntos que não vimos até o momento (essa atividade pode ser resolvida só com o que foi visto em aula, com suas devidas adaptações).
  Exemplo:
  - Declarar arrays de tamanho variável (padronizado no C99, mas erro em C++ pois não há suporte para VLA), ex. int n = 10; char arr[n];.
- Projeto que possui erros de compilação ou que trava durante a execução automaticamente perde 50% da nota máxima.
  Sobre erros de compilação: considero apenas erros, não há problema se o projeto tiver warnings (apesar que warnings podem avisar possíveis travamentos em tempo de execução, como loop infinito, divisão por zero etc.).
  Quando há necessidade de entrada de dados por parte do usuário, assumo que o usuário vai inserir as informações corretas (ex. tipos de dados corretos), a menos que o enunciado explicite que você deve garantir que os dados de entrada estejam corretos.

Em uma situação profissional, os itens indicados acima atrapalham (e muito) o trabalho da equipe. E o último item é gravíssimo (o ideal também é remover todos os warnings e sempre validar os dados).