Aula 06 – Exercícios

Parte 1 – Escrita (individual – entregue via Formulário no Google Classroom)

Exercício 1: em seu caderno, faça, **passo a passo** (como feito em aula), a inserção dos seguintes elementos em uma Árvore AVL, nesta ordem: 30, 51, 15, 44, 48, 22, 18, 5, 7, 41, 37.

Exercício 2: a partir da Árvore AVL obtida no Exercício 1, remova os elementos 30, 5, 51, nesta ordem. Mostre, **passo a passo**, as etapas de rebalanceamento da árvore.

Parte 2 – Codificação (individual OU em dupla OU em trio – entregue via Run Codes)

Exercício 3: escreva, em linguagem C pura (sem nenhum comando próprio de C++), um programa que leia um número inteiro representando uma opção do usuário. As ações a serem executadas dependerão da opção lida, sendo:

- Se a opção lida for 1, seu programa deve ler um novo valor inteiro e inserir este valor em uma Árvore AVL.
- Se a opção lida for 2, seu programa deve ler um novo valor inteiro e pesquisá-lo na Árvore AVL. Se o valor pesquisado estiver presente na Árvore, imprima-o. Se o valor pesquisado não estiver presente na Árvore, imprima um "x" minúsculo como resposta. Pule uma linha após imprimir o número ou o "x".
- Se a opção lida for 3, seu programa deve ler um novo valor inteiro e removê-lo da Árvore AVL.
- Se a opção lida for 4, seu programa deve executar o percurso em pré-ordem na Árvore AVL. Imprima os valores dos elementos em uma única linha, separados por um único espaço. Pule uma linha ao finalizar o percurso.
- Se a opção lida for 5, seu programa deve executar o percurso em-ordem (in-order) na Árvore AVL. Imprima os valores dos elementos em uma única linha, separados por um único espaço. Pule uma linha ao finalizar o percurso.
- Se a opção lida for 6, seu programa deve executar o percurso em pós-ordem na Árvore AVL. Imprima os valores dos elementos em uma única linha, separados por um único espaço. Pule uma linha ao finalizar o percurso.
- Se a opção lida for 7, seu programa deve ler um novo valor, pesquisar pelo nó que contém o valor lido e imprimir o Fator de Balanceamento deste nó (os casos de testes não irão trapacear seu programa pesquisando por um nó inexistente, mas testará apenas o FB para os nós presentes na árvore!). Pule uma linha após imprimir o FB.
- Se a opção lida for 8, seu programa deve finalizar. Note que seu programa encerra apenas nesta opção. Ao terminar de processar qualquer outra opção, o programa deve retornar ao loop inicial para ler a próxima ação que o usuário deseja executar.

LEIA AS OBSERVAÇÕES DA PRÓXIMA PÁGINA

Observação 1: este exercício está dividido em uma parte escrita e uma parte prática de codificação. A entrega só será válida se as duas partes forem entregues.

Observação 2: todos os alunos devem entregar a parte escrita e a parte prática individualmente. Não há problema em entregar o código 100% igual ao de sua dupla (ou trio), porém, cada aluno deve entregar a sua própria cópia.

Observação 3: caso você faça a parte de codificação em dupla ou em trio, insira no início do código um comentário com o nome de todos os autores. Os nomes devem estar nas primeiras linhas, antes mesmo dos #includes. Caso você não faça isso, o Run Codes acusará que você plagiou de sua dupla (ou trio).

Observação 4: o arquivo `avl.c` disponibilizado contém a implementação da fase de construção da Árvore AVL: operações de inserção, rotações e balanceamento. Use-o como base para o Exercício 3. Esta implementação em C é baseada no código de Sedgewick, escrito em Java, e disponibilizado no arquivo `AVLTreeST.java`.

Observação 5 (muito importante!): o código de Sedgewick comete um pequeno deslize, que se opõe a teoria clássica de Árvore AVL. Sedgewick considera o cálculo de FB como *FB = Altura esquerda – Altura direita*, enquanto que a teoria clássica subtrai a altura direita da altura esquerda. Você deve ajustar a função que calcula o FB e a função que realiza o balanceamento para se adequarem à teoria clássica.

Observação 6: as operações de pesquisa e de percursos são idênticas às das árvores binárias de busca, como apresentado em Algoritmos e Estruturas de Dados I. Para a operação de remoção, converta para C o código de Sedgewick. Cuidado com os ponteiros! Estude as funções de construção da árvore fornecidas antes de fazer as suas! Cuidado também com as palavras reservadas do C que são nomes de funções no código Java (`delete`, por exemplo).

Observação 7: simulador de Árvore AVL: `https://www.cs.usfca.edu/~galles/visualization/AVLtree.html` (também disponível em: `https://people.ok.ubc.ca/ylucet/DS/AVLtree.html`)