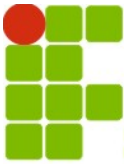

INSTITUTO FED. DE EDUCAÇÃO, CIÊNC. E TEC. DE PERNAMBUCO
CURSO: TEC. EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: ALGORITMOS E ESTRUTURAS DE DADOS
PROFESSOR: RAMIDE DANTAS
ASSUNTO: ÁRVORE BINÁRIA DE BUSCA (BST) E AVL

Prática 08

Parte 0: Preparação

Passo 1: Crie um novo projeto chamado **Pratica8**.

Configure o projeto para utilizar C++11.
(*Project Properties* > *C++ Build* > *Settings*: no caso do Cygwin procure por "*Dialect*" e selecione "*ISO C++11*" em "*Language standard*")

Passo 2: Adicione os arquivos que acompanham a prática 8 ao projeto.

main.cpp: contém a função main. Cria a árvore e realiza inserções, buscas e remoções.

bst.h e **bst.cpp:** declaração e implementação da árvore binária de busca, respectivamente.

avl.h e **avl.cpp:** declaração e implementação da árvore AVL.

Passo 3: Compile e rode o código para verificar se não há erros.

Nesse ponto deve apenas compilar e rodar mas não produzir resultados; algumas funções não estão implementadas.

Passo 4: Estude o código para se familiarizar.

Veja o material de aula se necessário. Faça comentários nos trechos mais complicados. Refatore o código se achar que vai ajudar seu trabalho. Teste novamente antes de fazer modificações mais profundas.

Parte 1: Implementando funções básicas na Árvore Binária de Busca (BST)

Passo 1: Implemente a função de inserção na BST.

A função a ser implementado é a versão privada, cujo corpo está em **bst.cpp**. A implementação mais direta é recursiva. A função é chamada passando um nó da árvore (inicialmente a raiz, depois esquerdo ou direito), e o retorno permite atualizar os ponteiros do nó pai (ou a raiz, quando for a primeira inserção). Depois de inserir o novo elemento, a altura do nó deve ser atualizada (chame `updateH()` ao final da inserção); isso será usado pela árvore AVL.

Importante: caso a chave já exista na árvore, não faça nada. Duplicações na árvore quebram o funcionamento da árvore AVL.

Passo 2: Implemente a função de busca na BST.

Há duas versões da função `search()`: uma pública e outra privada ; você deve trabalhar na versão privada (ela é usada pela pública). Siga o material da aula.

Passo 3: Implemente a função de exibição da BST.

A função `show()` privada deve varrer e exibir os nós da árvore em ordem. A função deve exibir não só o valor da chave e mas também a altura do nó no formato "(chave, altura)".

Passo 4: Compile e teste a aplicação.

Verifique se os elementos inseridos foram encontrados pela busca.

Parte 2: Implementando Sucessor

Passo 1: Implemente o método `successor()` em **bst.cpp**.

Em uma árvore de busca, o predecessor de um valor X presente na árvore é o maior valor Y também presente na árvore que é menor que X. Isto é, se pegássemos os elementos da árvore em ordem, Y seria o valor que viria imediatamente antes de X (se houver). O método `predecessor()` realiza essa busca.

Simetricamente, o sucessor de X é o valor W que viria logo após X na sequência de elementos presentes na árvore. Implemente o método `successor()` se baseando na implementação de `predecessor()`. Estude esse método para fazer os modificações necessárias.

Passo 2: Compile e teste a aplicação.

Rode a aplicação, verificando se o valor retornado por `successor()` é o correto.

Parte 3: Implementando Rotações na Árvore AVL

Passo 1: Implemente as métodos `rotateLeft()` e `rotateRight()` em **avl.cpp**.

Esses métodos realizam as rotações simples à esquerda e à direita respectivamente na árvore AVL. Veja como esses métodos são usados dentro de `rebalance()`. Nesses métodos, após realizar a rotação em si (ajuste dos ponteiros), é preciso atualizar a altura dos nós envolvidos. Atenção que a ordem de atualização é importante.

Passo 2: Implemente a função privada `show()` em avl.cpp.

A função deve exibir não só o valor da chave e a altura do nó, mas também o fator de balanceamento de cada nó no formato "(chave, altura, BF)". Modifique a versão da BST.

Passo 3: Compile e teste a aplicação.

Na função `main()`, certifique-se de mudar o código para usar a árvore AVL. Mude o código também de forma a inserir elementos de forma ordenada na árvore. Verifique na saída se a árvore permanece válida e se os fatores de balanceamento e alturas estão corretos.