Laboratório 2: Balanceamento ABB

Entrega até segunda, 25/3, às 6h

Nesse laboratório, vamos continuar a implementação dos métodos para uma árvore binária de busca, implementados em **arvore.h** e **arvore.c**.

Os testes são sugestões. Você pode pensar em outras situações específicas ou outras sequencias que seriam interessantes de verificar. Use o arquivo **teste.c** como base para seus testes. Compile o programa e execute-o.

O programa agora recebe um argumento definindo o tamanho de duas árvores a serem geradas. Repare que, em função da ordem de inserção de nós usada para cada árvore, a primeira árvore será **desbalanceada** (na verdade, será uma árvore degenerada), e a segunda será **balanceada**.

Caso esteja utilizando o gcc, pode executar o programa passando o tamanho das árvores como um parâmetro via linha de comando:

```
gcc -o tlab teste.c arvore.c
./tlab 10 (criando duas árvores com 10 nós)
```

Você pode adicionar o parâmetro no Visual Studio através das propriedades do projeto, ou então, comentar as linhas 17 a 21 e definir um tamanho diretamente no código (pela variável **tam_arv**).

A partir disso, faça as seguintes tarefas:

1. Definimos como **altura** de uma árvore o maior nível dentre todos os seus nós (o que equivale ao comprimento do caminho mais longo da raiz até uma folha). Por sua vez, definimos que:
    a. a raiz de uma árvore tem nível 0
    b. se um nó tem nível n, seus filhos tem nível n+1

    Com base nessa definição, implemente a função **altura** da interface **arvore.h**. Assuma que uma árvore vazia tem altura **-1**. Use as duas árvores que o programa cria, mas dessa vez use tamanhos pequenos...

    Dica: a **altura** é muito semelhante com o percurso de **pós-ordem**, pois primeiro as subárvores são visitadas antes de avaliar o nó raiz.

2. Numa árvore binária balanceada, a diferença entre as alturas da subárvores esquerda e direita **de qualquer nó** é no máximo 1. Utilizando a função **altura** do exercício anterior, implemente a função **e_balanceada** da interface **arvore.h**. Essa função deverá retornar 0 (falso) caso a árvore não seja balanceada, e um valor

diferente de 0 (verdadeiro) caso a árvore seja balanceada. Não se preocupe em otimizar o cálculo de altura.

Obs: Você pode usar a função **abs** para obter o valor absoluto de uma expressão...

Novamente, você pode usar as duas árvores que o programa cria. Experimente também com uma árvore onde o "desbalancemento" seja encontrado em uma subárvore. Por exemplo, se você inserir chaves na ordem **3,1,2,4,5,6** o desbalanceamento será encontrado na subárvore direita. Crie então uma terceira árvore, inserindo os nós com as chaves nessa ordem, e teste se sua função fornece a resposta correta para ela.

3. Vamos medir o tempo de criação e busca das duas árvores inicialmente geradas. Para isso, modifique o arquivo **teste.c**. Use a função **clock** do módulo **time.h** para medir o tempo de busca de elementos em cada uma das duas árvores, para diferentes valores de tamanho. Sua estrutura de medição deve ficar algo como:

   ```
   // tempo inicial
   tstart = clock();

   // busca/insere elementos na árvore

   // tempo final
   tend = clock();

   // calcula e mostra tempo total
   printf("Tempo: %lf segundos\n", ((double)(tend-tstart)) / CLOCKS_PER_SEC);
   ```

   ATENÇÃO: Não coloque chamadas a printf ou outras funções de entrada e saída "dentro" do trecho cujo tempo de execução estiver medindo.

   Para ver resultados interessantes, use valores bem variados de tamanho da árvore, como 1000, 10000, 20000, ... Relate na área de texto da tarefa os testes executados e os tempos obtidos. Comente os resultados do seu teste.

4. Implemente a função **balancear** da interface **arvore.h**. Essa função deverá realizar as mudanças necessárias na árvore para balanceá-la, quando necessário. Lembre-se que, em uma árvore balanceada, **todas** as subárvores devem estar balanceadas!

   Para implementar sua função, você pode usar quaisquer das outras funções implementadas, por exemplo (e não apenas) a função **num_nos** para calcular a quantidade de nós em cada uma das subárvores.

Se quiser, você também pode utilizar métodos auxiliares que retornam um ponteiro para o menor elemento de uma árvore, e para o maior elemento de uma árvore. Usando esse ponteiro, você pode mover os dados de um nó para outro (por exemplo, para a raiz).

Faça upload dos arquivos **arvore.c** e **teste.c** no EAD até dia 25 de março, segunda, às 6h. Indique na área de texto, que funções você implementou, e se o resultado verificado para cada uma está correto ou não. Lembre-se de fazer a entrega mesmo que não tenha chegado ao final do exercício.