Trabalho de Algoritmos e Estruturas de Dados 3

Professor: Marcos Didonet
Aluno: Vinícius Teixeira V. dos Santos
Usuário: vtvs18

Arquivos presentes no trabalho:

main.c: possui somente a função main do programa, fazendo diretamente a leitura da entrada e execução básica do programa.
funcoes_abb.c: possui funções que podem ser utilizadas em qualquer tipo de árvore binária (binária de busca, AVL, rubro-negra, etc).
funcoes_avl.c: possui funções feitas especificamente para a implementação de uma árvore AVL.
arvore.h: possui a estrutura do nó da AVL utilizada no trabalho.

Funções presentes nos arquivos:

funcoes_abb:

visita:
Faz a impressão dos dados como especificado no trabalho(valor_do_nó, altura_do_nó).
pre_ordem, em_ordem e pos_ordem:
Percorrem a árvore para que seja realizada a impressão de dados da mesma. Implementados de forma vista em sala de aula.
altura_arvore:
Calcula qual a altura máxima de uma arvore, percorrendo de sua raiz até suas folhas.

funcoes_avl:

cria_no:
Cria um nó alocando o devido espaço de memória, deixando como sua chave o dado passado.
corrige_altura:
Faz a correção da altura presente no próprio nó passado por parâmetro.
corrige_arv:
Faz a correção de forma recursiva da altura de todos os nós presentes na árvore.
rot_esq e rot_dir:
Fazem as rotações com os nós da árvore como vistos em sala de aula.
insere_arvore:
Insere o nó desejado na árvore de forma recursiva, imprimindo”Nó x já existente” na tentativa de inserir um nó que já foi inserido.
remove_no:
Remove o nó desejado na árvore de forma recursiva, imprimindo”Nó x inexistente para remoção” na tentativa de remover um nó que não exite.
altura_no:
Calcula a altura de um nó até sua raiz.

main.c:

Possui somente a função main, fazendo a leitura de entradas enquanto possível, sempre inserindo ou removendo um nó quando necessário. Caso leia uma letra que não seja i ou r, imprime “Operação não encontrada, saindo...”, o que faz com que também encerre a leitura de entradas.

Observações:

No trabalho estão presentes alguns testes `if (no -> pai != NULL)`. Estes testes servem para evitar *segmentation faults* ao tentar usar o ponteiro do pai do nó (caso seja NULL [nulo], não é possível utilizar seus ponteiros, criando *segmentation faults*).

Para que o programa seja compilado, basta executar `make` no terminal, gerando o executável myavl.

No Makefile também está presente a opção clean. Esta opção pode ser usada executando o comando `make clean` no terminal. Tudo o que ela faz é excluir o executável myavl da pasta atual.

Os demais aquivos .h servem como “biblioteca” das funções criadas em para seus respectivos arquivos .c.