Trabalho prático - AVL

Thalita Maria do Nascimento

Departamento de Informática
Universidade Federal do Paraná – UFPR
Curitiba, Brasil
thalitanascimento@ufpr.br

18 de julho de 2022

Resumo

O presente relatório visa explicar as especificidades das funções utilizadas para o trabalho prático de programar a Árvore *Adelson-Velsky e Landis*(AVL).

1 Introdução

No trabalho, foram desenvolvidas 14 funções na biblioteca da AVL, sendo elas *criaNo*, *inclui*, *busca*, *exclui*, *emOrdem*, *sucessor*, *minimo*, *transplante*, *rotacaoDir*, *rotacaoEsq*, *ajustaBalanco*, *alturaArvore*, *alturaNo* e *liberaNos*. Para a exclusão, foi escolhido utilizar a função *transplante* no lugar de rotações.
A função *alturaArvore* é a função comumente conhecida como *altura* e calcula a altura de uma subárvore. As funções *exclui*, *ajustaBalanco*, *alturaNo* e *liberaNos* são particulares e merecem atenção em sua descrição.

2 Funções com particularidades

A função *exclui* tem seus parâmetros diferentes dos vistos em aula. Ela recebe um ponteiro para o nó, e não a chave para realizar a busca internamente. A mudança foi feita para facilitar a desalocar a memória dos nós.

A função *ajustaBalanco* é utilizada para manter a propriedade da AVL de ser uma árvore binária balanceada. Ela trata quatro casos que precisam ser balanceados.

A função *alturaNo* calcula a altura do nó passado como parâmetro.

A função *liberaNos* exclui todos os nós presentes na árvore, liberando a memória alocada dinamicamente.

3 Documentação

Retorna o nó criado: *tNo *criaNo(int chave);*

Inclui um nó na árvore, retorna a raiz: *tNo *inclui(tNo *no, int chave);*

Busca pelo nó utilizando a chave, a função é utilizada para "exclui". Retorna a raiz: *tNo *busca(tNo *no, int chave);*

Exclui o nó, caso não seja NULL: *tNo *exclui(tNo *raiz, tNo *no);*

Travessia em ordem alterada para imprimir a altura também: *void emOrdem(tNo *no);*

Retorna o sucessor, que é o nó mais a esquerda da sua subárvore à direita: *tNo *sucessor(tNo *no);*

Procura pela chave mínima à esquerda: *tNo *minimo(tNo *no);*

Corrige os ponteiros do pai em caso de exclusão: *void transplante(tNo *no, tNo *next);*

Rotaciona o nó para a direita, elevando seu filho da esquerda: *tNo *rotacaoDir(tNo *x);*

Rotaciona o nó para a esquerda, elevando seu filho da direita: *tNo *rotacaoEsq(tNo *x);*

Função para ajustar a AVL. Caso a diferença entre as alturas das subárvores direita e esquerda de um nó sejam -2 (desbalanceada à esquerda) ou 2 (desbalanceada à direita), é feito o ajuste. O ajuste trata 4 casos, filhos profundamente balanceados e em zig-zag, cada um com sua variação para esquerda ou direita: *tNo *ajustaBalanco(tNo *no);*

Retona a altura da subárvore: *int alturaArvore(tNo *no);*

Retorna a altura do nó atual: *int alturaNo(tNo *no);*

Remove a raiz até desalocar todos os nós: *void liberaNos(tNo *raiz);*