Documentação AEDSII

Igor Silva Batata

July 3, 2018

1 Introdução

Árvore AVL é uma árvore binária de busca balanceada, ou seja, uma árvore AVL são as árvores que minimizam o número de comparações efetuadas no pior caso para uma busca com chaves de probabilidades de ocorrências idênticas. Contudo, para garantir essa propriedade em aplicações dinâmicas, é preciso reconstruir a árvore para seu estado ideal a cada operação sobre seus nós (inclusão ou exclusão), para ser alcançado um custo de algoritmo com o tempo de pesquisa tendendo a O(log N). As operações de busca, inserção e remoção de elementos possuem complexidade O(log n), que são aplicados a árvore de busca binária. O nome AVL vem de seus criadores soviéticos Adelson Velsky e Landis, e sua primeira referência encontra-se no documento "Algoritmos para organização da informação" de 1962. Nessa estrutura de dados cada elemento é chamado de nó. Cada nó armazena uma chave e dois ponteiros, uma para a subárvore esquerda e outro para a subárvore direita.

2 Implementação

2.1 void cria(int *lista_criada, AVL** a)

Essa função cria a primeira célula da arvora AVL. Ela tambem adiciona 1 ao ponteiro para variavel "lista_criada", que é a variável responsavel por certificar que o uruário criou a AVL

2.2 void inserir(Matricula *x, AVL** a)

Função que faz a inserção de valores na arvore AVL. É uma função recursiva que se chama para a esquerda, caso seja menor que o valor, ou a direita, caso seja maior. Também faz a alterção do fator de balancemento. Adicionando um, caso a função vá a direita, ou subtraindo um se for a esquerda. Ao final da função ele chama a função "balancear(AVL** p)".

2.3 void balancear(AVL** a)

Responsavel por analisar o fator de balanceamento de toda a árvore. A partir dos valores, ela reconhece qualquer tipo de desbalanceio na árvore e chama as funções simple_esquerda(AVL**P), simples_direita(AVL**P), dupla_esquerda(AVL**P), dupla_direita(AVL**P). Que são funções que fazem as rotações AVL para balancear a árvore.

2.4 void retirar(Matricula x, AVL* *p)

Funcionamento muito parecido com o a inserção, mas faz a remoção da matricula digitada. Após a remoção, realoca as structs e chama a função "balancear" para reorganizar os conteudos de acordo com seu fator de balanceamento.

2.5 int buscar(Matricula *x, AVL* p)

Uma função que retorna um ou zero caso haja a matricula desejada. É uma função recusrsiva que se chama (esq, dir) diacordo com a comparação dos valores da matricula (maior, menor).

2.6 void central(AVL* p)

Responsavel por fazer o caminhamento central na AVL. Printa os valores em ordem de posição na arvore.

3 Conclusões

Um trabalho muito interessante e prazeroso de ser feito. Vários bons desafios na implementação desse código. Tive algumas dificuldades com coisas basicas de C, como a função fgets e o uso de ponteiros. A parte mais desafiadora do código foram os fatores de balanceamento e as rotações.

4 Referências

https://pt.wikipedia.org/wiki/Árvore_AVL