Comparação de implementação de dicionários com árvores vermelho-preto e avl com std::map da linguagem C/C++ *

Ítalo L. F. Portinho[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Av. Gal. Milton Tavares de Souza, s/nº – 24.210-346 – Niterói – RJ – Brazil

italoleite@id.uff.br

Abstract. *This paper presents implementations for an abstract data type dictionary based on binary search trees, more specifically the self balancing red-black tree and the AVL tree. We describe insertion, deletion and search operations in both trees and the actions taken to keep them balanced, and in sequence two tests are presented: the first one is insertion/exclusion intensive, and the second is lookup intensive. The conclusion is that the AVL tree has a lower height so is more recommended to lookup intensive applications, and the red-black tree is more efficient for insertions and deletions.*

Resumo. *Este artigo apresenta implementações para um tipo de dados abstrato dicionário baseado em árvores binárias de busca, mais especificamente as árvores auto-balanceáveis vermelho-preto e AVL. São descritas as operações de inserção, exclusão e busca em ambas as árvores e as ações realizadas para mantê-las balanceadas. São apresentados dois tipos de testes, o primeiro intensivo em inserções e exclusões e o segundo intensivo em buscas e concluímos que as árvores AVL possuem altura menor e são mais eficientes para buscas e as árvores vermellho-preto são mais eficientes para inserções/exclusões.*

*

1. Árvores Binárias de Busca

Dicionários são estruturas de dados que suportam operações do tipo chave-valor, como inserção, exclusão e busca. Existem diversas formas de implementá-los e nesse artigo vamos apresentar uma forma baseada em árvores binárias de busca. Esse tipo de árvore possuí a propriedade de que dado um nó r, todos elementos da sua subárvore direita x são menores do que ele e, todos os elementos da sua subárvore esquerda y são maiores do que ele (Figura 1). Neste artigo e na implementação vamos nos concentrar em árvores que armazenam números inteiros.

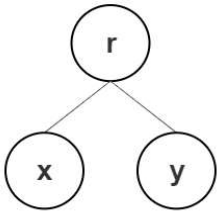

Figura 1. Árvore binária de busca: subárvore x possuí elementos menores do que r, subárvore y possuí elementos maiores do que r.

A operação de inserção de um elemento em uma árvore binária de busca começa comparando ele com a raíz da árvore e seguindo o caminho descendente repetindo as comparações até encontrar um ponteiro nulo, que será a posição do novo elemento (Figura 2). Esse também é o procedimento adotado numa operação de busca de um elemento, sendo que se o elemento não for nem menor nem maior que o valor do nó, significa que encontramos o elemento procurado, e se encontrarmos um ponteiro nulo, o elemento não está presente na árvore.

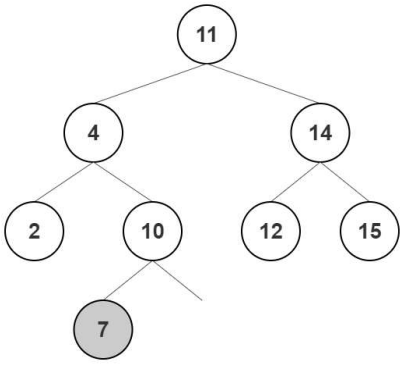

Figura 2. Inserção do elemento 7 em uma árvore binária de busca com raíz 11.

A operação de exclusão começa como uma busca pelo elemento. Se o nó for uma folha basta removê-lo. Se o nó possuir apenas um filho, o nó do elemento é removido e atualizamos os ponteiros para o filho assumir seu lugar. Se o nó possuir dois filhos, ele é substituído pelo seu sucessor na árvore ou seja o menor elemento da sua subárvore direita (Figura 3). As operações de inserção, exclusão e busca possuem complexidade O(h), sendo h a altura da árvore, pois requerem no máximo um percurso descendente na árvore.

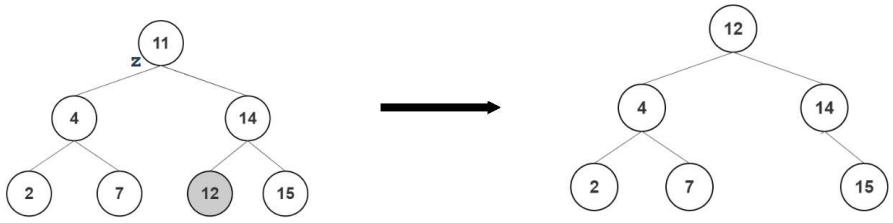

Figura 3. Exclusão de um elemento com dois filhos em uma ABB.

Podemos observar que dependendo da ordem que as operações de inserção e exclusão forem realizadas, elas podem levar a um desbalanceamento da árvore. Por exemplo a inserção de uma sequência ordenada de números tornaria a árvore uma lista, e sua altura seria o número de elementos (n) degradando suas operações de inserção, exclusão e busca para a complexidade O(n) enquanto que se a árvore se mantiver balanceada as operações serão realizadas com complexidade O(lg_n). Nas seções a seguir serão apresentadas duas árvores binárias de busca auto-balanceáveis, árvore vermelho-preto e árvore AVL, que garantem manter a altura da árvore proporcional à lg_n.

2. Árvores vermelho-preto

The first page must display the paper title, the name and address of the authors, the abstract in English and "resumo" in Portuguese ("resumos" are required only for papers written in Portuguese). The title must be centered over the whole page, in 16 point boldface font and with 12 points of space before itself. Author names must be centered in 12 point font, bold, all of them disposed in the same line, separated by commas and with 12 points of space after the title. Addresses must be centered in 12 point font, also with 12 points of space after the authors' names. E-mail addresses should be written using font Courier New, 10 point nominal size, with 6 points of space before and 6 points of space after.

The abstract and "resumo" (if is the case) must be in 12 point Times font, indented 0.8cm on both sides. The word **Abstract** and **Resumo**, should be written in boldface and must precede the text.

3. Árvores AVL

In some conferences, the papers are published on CD-ROM while only the abstract is published in the printed Proceedings. In this case, authors are invited to prepare two final versions of the paper. One, complete, to be published on the CD and the other, containing only the first page, with abstract and "resumo" (for papers in Portuguese).

4. O Teste

Section titles must be in boldface, 13pt, flush left. There should be an extra 12 pt of space before each title. Section numbering is optional. The first paragraph of each section should not be indented, while the first lines of subsequent paragraphs should be indented by 1.27cm.