INTERFACE TECNOLÓGICA

CONCEITOS IMPORTANTES DE ESTRUTURA DE ÁRVORES: binárias e binária balanceada (AVL)

IMPORTANT TREE STRUCTURE CONCEPTS: binary and balanced binary (AVL)

Diogo Augusto Colombo – diogo.colombo@fatec.sp.gov.br
Faculdade de Tecnologia de Taquaritinga (Fatec) – Taquaritinga – SP – Brasil

Santo Oliani Junior – juninhooliani@gmail.com
Faculdade de Tecnologia de Taquaritinga (Fatec) – Taquaritinga – SP – Brasil

DOI: 10.31510/infa.v20i1.1579
Data de submissão: 20/03/2023
Data do aceite: 29/05/2023
Data da publicação: 30/06/2023

RESUMO

Atualmente o mercado de trabalho na área do desenvolvimento demanda mão de obra qualificada e especializada. Os futuros desenvolvedores devem ter como destaque os métodos de estrutura de dados, e, entre eles, um dos mais importantes e que mais faz o quem sabe utilizá-lo se destacar é a estrutura de árvore, que engloba a estrutura de árvores binárias e binárias balanceadas (AVL). Assim, este artigo tem como objetivo evidenciar, analisar, explicar e descrever como são os processos de utilização dessas estruturas, baseando-se na metodologia de pesquisa bibliográfica por meio de livros revistas e artigos, a fim de evidenciar as diferenças entre árvores binárias e AVL e demonstrar seus fatores semelhantes. Por fim, se conclui que é de suma importância para um programador conhecer e saber implementar.

Palavras-chaves: Árvores. Árvores Binárias.Árvores AVL. Desenvolvedores. Estrutura de Dados.

ABSTRACT

Currently, the labor market in the area of development demands qualified and specialized labor. Future developers should highlight the data structure methods, and, among them, one of the most important and that most makes those who know how to use it stand out is the tree structure, which includes the structure of binary and binary trees balanced (AVL). Thus, this article aims to highlight, analyze, explain and describe how the processes of using these structures are, based on the methodology of bibliographical research through books, magazines and articles, in order to highlight the differences between binary trees and AVL and demonstrate their similar factors. Finally, it is concluded that it is of paramount importance for a programmer to know and know how to implement.

Keywords: Trees. Binary Trees. AVL Trees. Developers. Data Dtructure.

1 INTRODUÇÃO

É impossível pensar na sociedade atual sem associar sistemas de automação, de integração, de jogos e muitos outros, assim, fica claro como a sociedade está dependente dos softwares existentes. A verdade é que, com a vinda desses sistemas a humanidade teve a oportunidade de evoluir com muita mais facilidade do que séculos anteriores.

Os primeiros assim são denominados computadores tinham como única função decodificar as mensagens que os alemães enviavam a suas tropas. Em 1945, Alan Turing começou a desenvolver uma máquina chamada Christopher para um processamento geral, essa máquina utilizava códigos binários para seu processamento. Com esse surgimento, várias técnicas e modelos de programação foram sendo desenvolvidos, mas uma técnica que se destacaria mais tarde é a denominada estrutura de dados, que tem como principal foco organizar os dados de uma forma que seja de fácil utilização durante o uso do programa.

Um programador qualificado deve ter em sua bagagem teórica um conhecimento amplo de estrutura de dados, para que consiga fazer um código que organize e facilite a utilização das informações. Nessa direção, temos dois modelos muito importantes que serão abrangidos por esse artigo: as árvores binárias e árvores de AVL, duas técnicas de suma importância para que um programador tenha relevância e destaque no mundo computacional.

Portanto, o objetivo deste artigo é explicar os modelos e técnicas de utilização de estrutura de dados, com o âmbito de abordagem do método de árvore. Para que os leitores deste artigo tenham uma introdução a um tema tão requerido no mercado de trabalho da atualidade.

2 FUNDAMENTO TEÓRICO

Com o objetivo de conseguir melhor entendimento sobre o assunto de estrutura de dados com foco em árvores binárias e de AVL, alguns conceitos importantes são explicados nas seções a seguir.

2.1 Conceitos de árvores binárias

Uma árvore em estrutura de dados tem como função agrupar informação em formato de árvore em um conjunto de elementos não lineares que não possuem uma sequência e não são encadeados.

O conceito de árvore binária pode ser estabelecido como um conjunto de elementos que são chamados nós, em que o primeiro elemento é denominado como raiz. Tem-se 3 subconjuntos denominados nó ou raiz, subárvore esquerda e subárvore direita (ASCENCIO; ARAÚJO, 2010). Um exemplo simples sobre o funcionamento das árvores binárias é a busca de números repetidos em uma lista. Para isso ser feito, o primeiro número da lista é colocado na raiz e o restante do número da lista é comparado como ele, caso for menor, é comparado com a árvore da esquerda, caso for maior, com a da direita, estas subárvores estarão vazias assim é alocado o número não repetido em um nó.

2.1.1 Níveis de árvores binarias

Os nós das árvores binárias possuem níveis, sendo o nível da raiz zero e os demais nós um nível acima de seus nós pai. O nível é dado da distância da raiz até a folha. Se uma árvore tiver 2 nós no nível 2, no nível acima que é o 3, ela poderá chegar no máximo s 4 nós, ou seja, 1 se estiver no nível A e possui C nós o nível acima que é X+1 ele terá 2C

Para identificar se temos uma árvore binária quase completa deve-se identificar alguns pontos, na quais são para Tenenbaum (1995):

1-A árvore precisa estar no nível d ou no nível X-1, para cada folha.

2-Para cada nó Xd da árvore com que possui descendente direto do nível d, todos os descendentes esquerdos que forem para folha estiverem no nível.

3-É denominado árvore binária quase completa quando é atribuído 1 para raiz e as raízes filhas recebem o dobro da dos seus pais.

2.1.2 Árvore estritamente binária

É necessário explicitar o que é uma árvore estritamente binária, sua definição se dá como: quando o nó não folha numa árvore binária tiver subárvores esquerda e direita com algum valor, a árvore é considerada uma árvore estritamente binária, ou seja, basicamente se um nó que não é folha estiver com suas respectivas subárvores com algum “valor” ela será estritamente binária.

2.1.3 Nós internos e externos

Os nós folhas não possuem filhos, assim chega-se à afirmação que os ponteiros esquerdam e direita só são necessários em nós, não em folhas. São usados dois tipos de nós para não folhas e folhas, left, right e info são usados para não folhas, ou seja, os nós-folha são os

elementos finais da árvore, enquanto os nós não-folha são os nós intermediários que conectam e estruturam os nós folha. A distinção entre esses dois tipos de nós é importante ao trabalhar com operações de busca, inserção, remoção ou qualquer outra manipulação de uma árvore binária.

Devidamente feita a distinção entre nós folhas e não folhas deve-se pontuar que os nós folhas são chamados de nós externos e os nós não folha de internos. Há duas maneiras de identificar onde o ponteiro dentro de um nó interno deve apontar (externo ou interno): a primeira maneira é declarar dois tipos diferentes de nós e usar a união de nós internos, e a segunda é usar um tipo só de ponteiro e sua união contém os campos esquerdos e direitos (TENENBAUM, 1995).

A seguir temos um trecho de código de implementação de uma árvore binária.

Figura 1 –Código de árvores binárias.

```
void imprime (link h)
{
if (h == NULL) return; printf("%d\n", h->item);
 imprime(h->l);
 imprime(h->r);
              }
```

Fonte:(FEOFILOFF, 2002)

2.3 Árvores de AVL

A estrutura árvore AVL foi criada em 1962 por Adelson-Velesky, ela se basicamente uma árvore binária balanceada que atende todos os requisitos de uma árvore binárias e em cada árvore apresenta diferença de altura entre as sub árvores esquerda e direita (ASCENCIO; ARAÚJO, 2010).

2.3.1 Definição de balanceamento

O balanceamento acontece quando a altura dos nós da subárvores são diferentes de mais que 1. Ou seja, para uma estrutura de árvore estar balanceada ela deve estar com todos os nós das sub árvores subsequentes no mesmo nível.

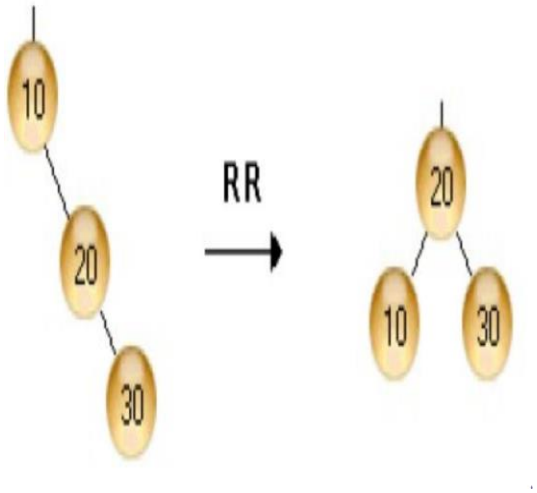
Para fazer a conta da altura para verificar se está balanceada, temos que fazer a altura da subárvore esquerda menos a da direita.

Todos os nós de uma árvore têm como fator de balanceamento 1, -1,0. Para árvores balanceadas, a subárvore estará balanceada quando estiver +1 a esquerda mais alta que a direita, -1 direita mais alta que a esquerda, a esquerda é igual a direita.

2.3.2 Tipos de rotação

Quando é inserido uma árvore AVL há dois métodos generalizados para balancear a árvore, que são rotação simples e rotação dupla. “Se a diferença de altura entre das subárvores está desbalanceada haverá uma rotação” (ASCENCIO; ARAÚJO, 2010, p. 329). Ainda na inserção, são empregados 4 métodos mais comum de balancear que, junto com as rotações simples e duplas, são inseridos, elas são chamadas de Right-Right (rotação a direita), Left-Left (rotação a esquerda), Left-Right (rotação esquerda-direita) e Right-Left (rotação direita-esquerda). A seguir, há imagens que esquematizam essas rotações, elas também se classificam como rotação simples devido que elas rotacionam apenas uma vez para que haja balanceamento.

Figura 2- Rotação a esquerda.

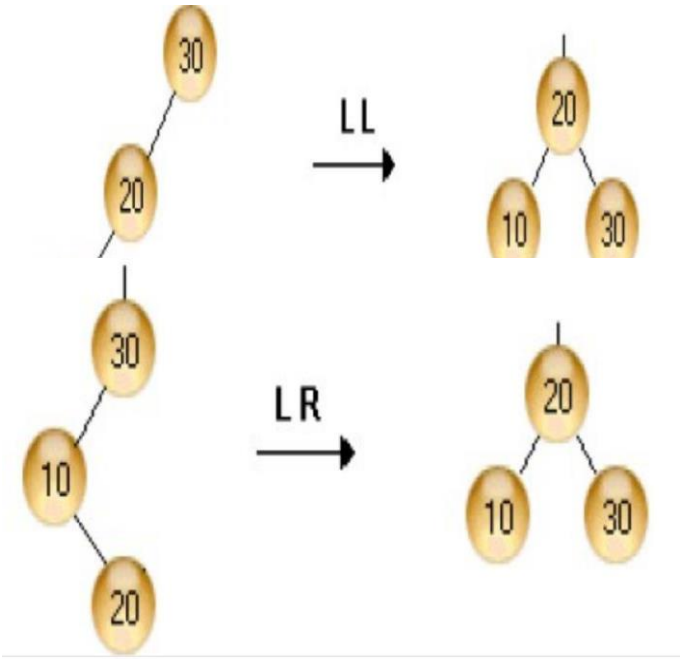


Fonte: (SOUZA, 2009, p. 17)

Figura 3- Rotação a direita.

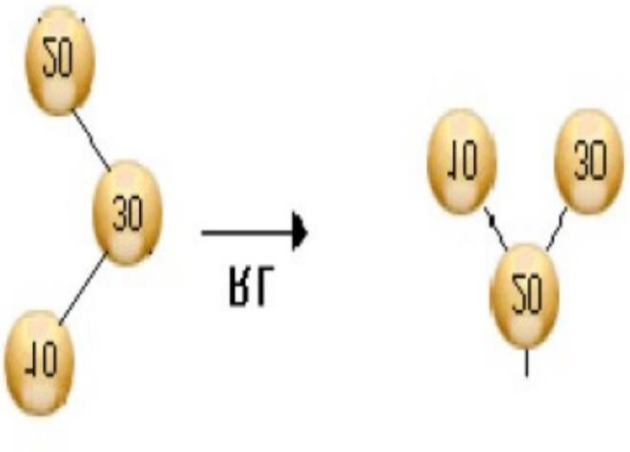
Fonte: (SOUZA, 2009, p. 18)

Figura 4- Rotação esquerda-direita



Fonte: (SOUZA, 2009, p. 19)

Figura 5- Rotação direita-esquerda

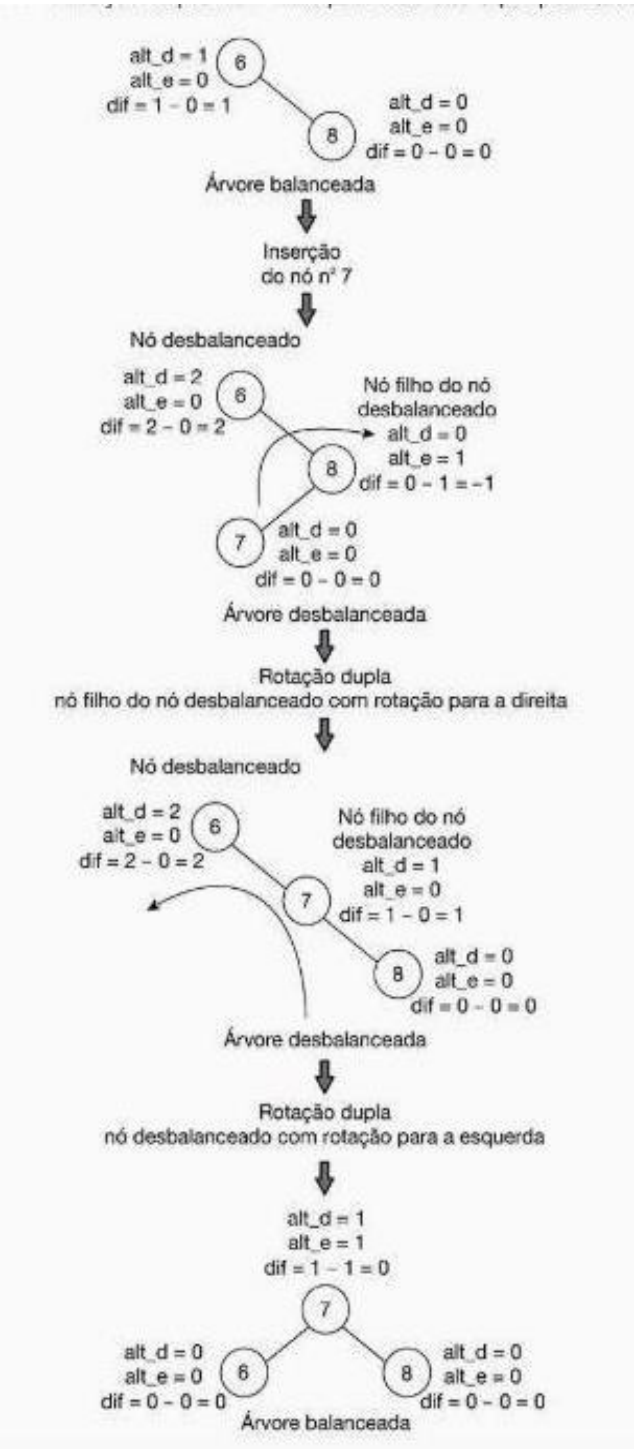


Fonte: (SOUZA, 2009, p. 20)

2.3.2.1 Rotação dupla

Outro método de balanceamento de árvores binárias é a rotação dupla, na imagem abaixo é perceptível que ele modifica uma vez a estrutura e logo em seguida modifica pela segunda vez para balanceá-la.

Figura 7 - Rotação dupla



Fonte: (ASCENCIO; ARAÚJO, 2010, p 332.)

2.4 Código de árvore binaria balanceada(AVL).

A imagem a seguir mostra um código de estruturação de dados na qual utiliza o método de árvores binárias balanceadas, ou mais conhecido como árvores de AVL.

figura 8-Código de AVL

```
#include <stdlib.h>
#include <stdio.h>
typedef struct no {
 int v; int bal; /* hdir - hesq */ struct no *esq, *dir;} No;
int altura(No* t) {
 if (t == NULL)
 return 0; int hesq = altura(t->esq);
 int hdir = altura(t->dir); return hesq > hdir ? hesq + 1 : hdir + 1;}
No* cria(int v, No* esq, No* dir) {
 No* n = (No*) malloc (sizeof(No));
 n->v = v; n->bal = altura(dir) - altura(esq); n->esq = esq;
 n->dir = dir;
 return n;
}
/* Versão ineficiente! Você sabe implementar outra? */ /* Não verifica se é uma árvore
de busca. */
int verifica_AVL(No* t) { if (t == NULL)  return 1;
 return abs(altura(t->dir) - altura(t->esq)) <= 1; }
void escreve(No* r) { if (r != NULL) { escreve(r->esq);
 printf("%d(%d)", r->v, r->bal);  escreve(r->dir);
 } }
void escreve_verifica(No* t) {  if (!verifica_AVL(t))  printf("Não é AVL!\n");
 escreve(t);  printf("\n");}
void LL(No** r) {  No* b = *r;  No* a = b->esq;  b->esq = a->dir;  a->dir = b;  a-
>bal = 0;  b->bal = 0;  *r = a;
} void RR(No** r) {  No* a = *r;  No* b = a->dir; a->dir = b->esq;  b->esq = a;
a->bal = 0;  b->bal = 0; *r = b; }
void LR(No** r) {No *c = *r; No *a = c->esq; No *b = a->dir; c->esq = b->dir;
a->dir = b->esq;  b->esq = a;  b->dir = c;switch(b->bal) {
 case -1: a->bal = 0; c->bal = 1;  break;
 case 0: a->bal = 0; c->bal = 0; break;  case +1:
 a->bal = -1; c->bal = 0; break; }
```

Fonte: modificado de Velskii (1962).

Figura 9-Código de AVL

```
b->bal = 0; *r = b;}

void RL(No** r) { No *a = *r;  No *c = a->dir; No *b = c->esq;

 c->esq = b->dir; a->dir = b->esq;  b->esq = a;  b->dir = c;

 switch(b->bal) { case -1: a->bal = 0;  c->bal = 1;  break;

 case 0:  a->bal = 0; c->bal = 0; break;  case +1: a->bal = -1; c->bal = 0; break; }
b->bal = 0; *r = b; }

/* *cresceu indica se a árvore cresceu

 após a inserção */

int aux_insere(No** t, int v, int *cresceu) { if (*t == NULL) { *t = cria(v, NULL,
NULL); *cresceu = 1;

 return 1; }

 if (v == (*t)->v)  return 0;   if (v < (*t)->v) {  if (aux_insere(&(*t)->esq, v,
cresceu)) {

 if (*cresceu) { switch ((*t)->bal) {case -1: if ((*t)->esq->bal == -1) LL(t); else
LR(t); *cresceu = 0; break;case 0: (*t)->bal = -1; *cresceu = 1; break; case +1:

 (*t)->bal = 0; *cresceu = 0; break;} } return 1; } else  return 0;}


if (aux_insere(&(*t)->dir, v, cresceu)) {if (*cresceu) {switch ((*t)->bal) {case -1:
(*t)->bal = 0; *cresceu = 0;break;  case 0: (*t)->bal = +1;

 *cresceu = 1;  break;case +1: if ((*t)->dir->bal == +1) RR(t); else
RL(t);*cresceu= 0;  break;} }  return 1; } else  return 0;}

/* Retorna 1 se inseriu ou 0 se    o elemento ? repetido. */

int insere(No **t, int v) { int cresceu;return aux_insere(t, v, &cresceu);}

int main() { No* t = cria(15,

cria(10, NULL, cria(12, NULL, NULL)),  cria (20, cria(18, NULL, cria(19, NULL,
NULL)), cria(25, NULL, NULL)));  insere(&t, 11); escreve_verifica(t);

 return 0;}
```

Fonte: modificado de Velskii (1962).

3 PROCESSOS METODOLÓGICOS

O método usado neste trabalho foi a revisão bibliográfica que tem como base a pesquisa

de artigos, bibliografia, livros com foco no procedimento de estruturação de dados, sempre buscando trazer a interpretação dos dados com clareza e manter sua veracidade.

4 RESULTADOS E DISCUSSÃO

O resultado alcançado por esta pesquisa, é dado quando as diferenças entre árvores binárias e de AVL ficam claras, árvores binárias são conjuntos de nós que possuem um fim, ou seja, são finitas e em geral pode haver sub árvores à esquerda e à direita, ou ser vazio. Já as árvores de AVL são árvores em que os nós são sempre balanceados usando como critério que nunca pode haver uma altura distinta entre eles mais que uma unidade.

Assim é perceptível que árvores binárias serão usadas para estruturas que tem como intuito a tomada de decisão bidirecional. Como por exemplo na escolha de repetição de letras nas quais a primeira letra é colocada como nó raiz e a partir dele é comparado as outras letras que vem em sequência, assim quando a letra coincidir com a que já possui na árvore haverá a repetição.

Por tanto, árvores de AVL são árvores binárias que tem como intuito de manter as sub árvores de menor altura possível e eliminando ao máximo a aparência de um vetor ou de uma lista encadeada. Por fim, sendo usado para qualquer situação que tenha como intuito de implementação com extenso fluxo de dados, tornando a busca muito mais rápida.

É perceptível que uma árvore AVL é um pouco mais extensa e trabalhosa para se codificar, mas é muito mais eficaz em programas que possuem um conjunto de dados volumosos. É necessário identificar de maneira correta o momento de seusar uma ou outra, visto que em programas que tem poucos dados e precisam ser estruturados não é vantajoso a utilização de AVL, então deve ser imprescindível saber quando usá-las de maneira correta.

5 CONCLUSÃO

Para chegar ao nível mais avançado como programador é preciso ter um amplo conhecimento das técnicas e maneiras que devem ser utilizadas para criação de um software, é perceptível que desenvolvedores possuem “obrigação” de saber utilizar as melhores técnicas de estrutura de dados. Uma dessas maneiras é a que foi abordada neste trabalho: a utilização de árvores.

Fica muito nítido quando se observa a diferença da implementação de estruturas lineares como, fila e pilha e não lineares como árvores binárias e AVL, a implementação de filas não

lineares acaba em determinados processos muito mais rápido e organizados para encontrar os dados em questão, já que o dado em uma árvore não é necessário percorrer todos os dados antes.

Por fim, fica evidente é que a utilização de árvores binárias e AVL para criação de um sistema eficaz e o mais rápido possível é de suma importância.

REFERÊNCIAS

ASCENCIO, A. F. G.; ARAÚJO, G. S. DE (2010). **Estruturas de dados algoritmos analise da complexidade e implementações em java e C/C++,** Perarson, 2010.

DEVIREDDY, Pravallika. **TREES- Binary Trees, Binary Search Trees, AVL Trees**. [*S. l.*], 2021. Disponível em: https://medium.com/about-data-structures/trees-binary-trees-binary-search-trees-avl-trees-be0470eb533. Acesso em: 3 mar. 2023.

FEOFILOFF, P. **Árvores binárias**. 2002.Disponível em: <https://www.ime.usp.br/~pf/mac0122-2002/aulas/trees.html> Acesso em: 23 fev. 2023.

SOUZA, J. F.**Árvores AVL: Estrutura de dados 2.** 2009. disponível em: <https://www.ufjf.br/jairo_souza/files/2009/12/5-Indexa%C3%A7%C3%A3o-**Árvore-AVL**.pdf> Acesso em: 20 fev. 2023.

TENENBAUM, Aaron; LANGSAM, Yedidyah; AUGENSTEIN, Moshe. **Estruturas de Dados Usando C**. São Paulo: MAKRON Books do Brasil Editora Ltda., 1995.

VELSKII , Adelson. **Fator de balanceamento = hdir - hesq. Arvores de AVL,** 1962. Disponível em: https://www.ic.unicamp.br/en/~islene/mc202/aula17/avl.c. Acesso em: 09 mar. 2023.