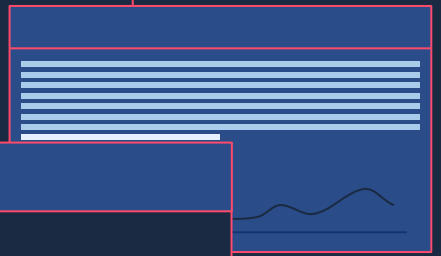
ÁRVORE AVL



Antônio Louigi Souza Bacelar
Gabriel Felipe Assunção de Souza

Índice

01 Problema inicial

02 Funcionamento do algoritmo

03 Desenvolvimento do código

04 Execução do programa

05 Explicação do desempenho

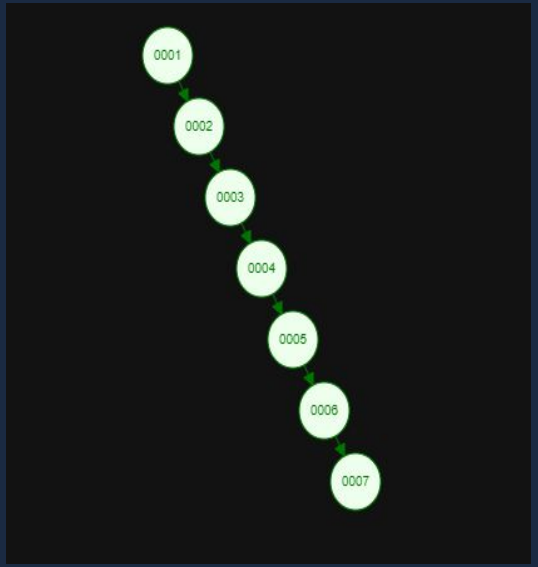
06 Comparação com a árvore binária

O problema das árvores binárias

- Nas árvores, a maioria das estruturas e operações nós temos tempo de execução O(h). Isso acaba sendo um problema quando temos árvores desbalanceadas, conforme a figura ao lado:

- Com isso precisamos definir uma forma de modificar a árvore a fim de obter uma altura logarítmica. Ou seja, O(log n), que é igual a uma árvore balanceada

	Lista encadeada		Árvore Binária	
	Médio caso	Pior caso	Médio caso	Pior caso
Inserção	O(1)	O(n)	O(log n)	O(n)
Deleção	O(1)	O(n)	O(log n)	O(n)
Busca	O(n)	O(n)	O(log n)	O(n)

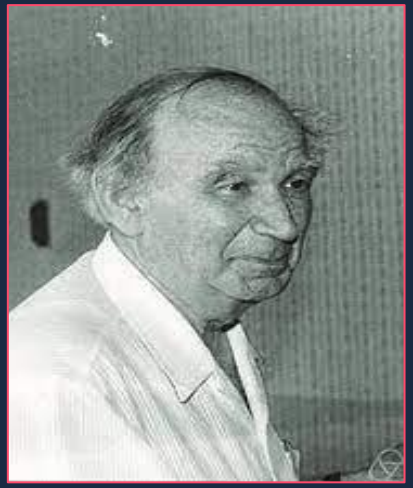


h = linear = O(n)

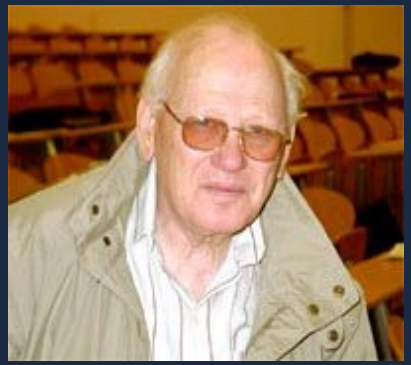
História da Árvore AVL

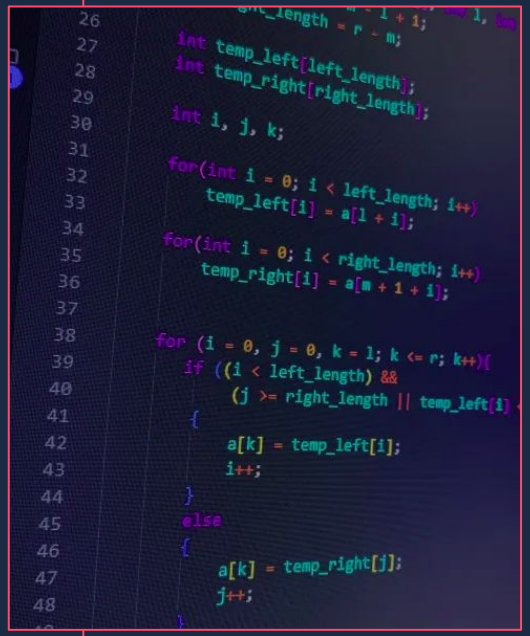
- Criada em 1962 pelos soviéticos Adelson Velsky e Landis
- Surgiu com a necessidade de otimizar as operações de busca em árvores binárias
- Tem como princípio fundamental manter o balanceamento da árvore

Evgenii Landis



Adelson Velsky



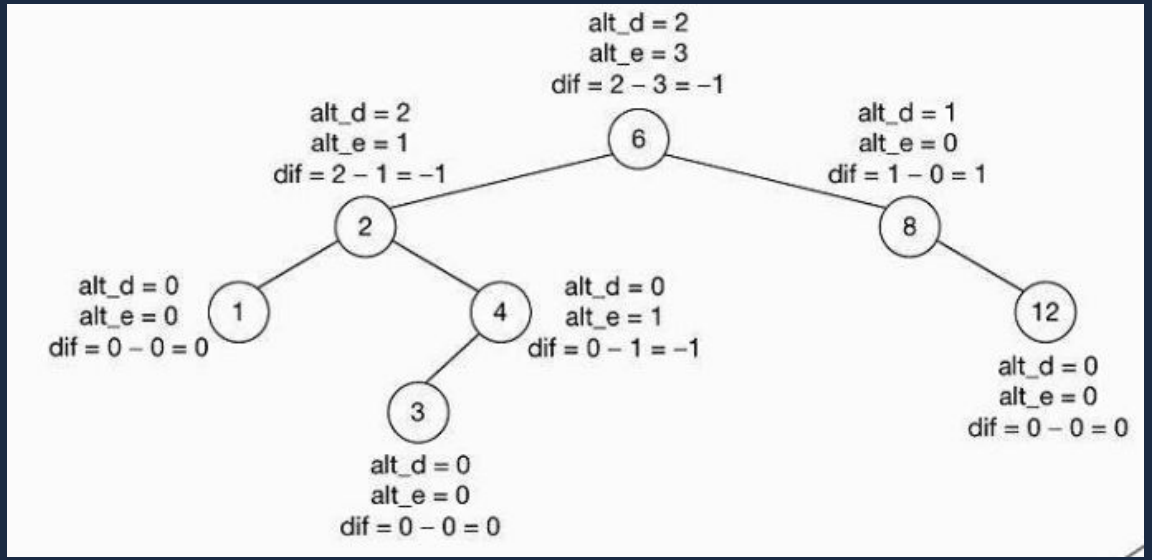


02

Funcionamento do algoritmo

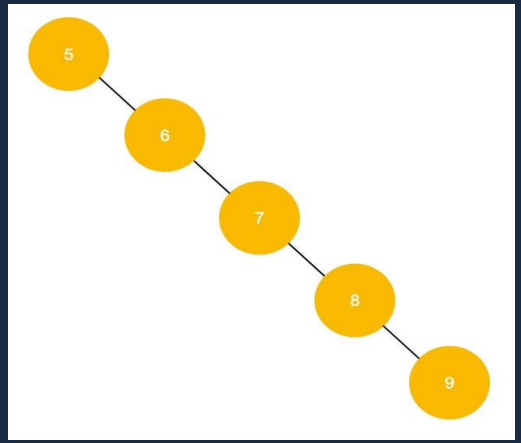
Conceito de balanceamento

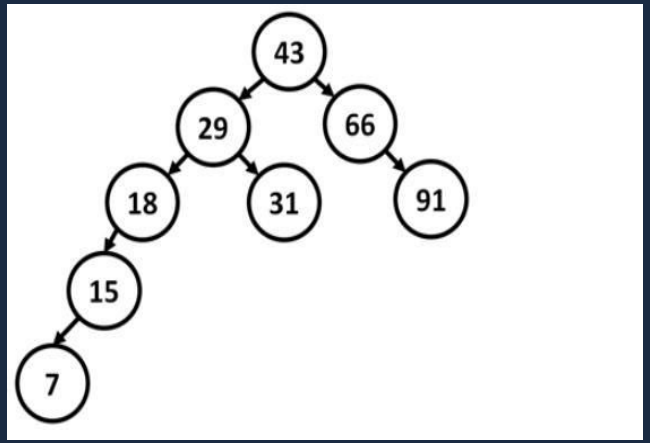
A Árvore AVL obedece a uma propriedade importante: cada nó apresenta diferença de altura entre as sub-árvores direita e esquerda de 1, 0 ou -1. Esse valor é chamado de fator de balanceamento do nó. Conforme a figura abaixo:



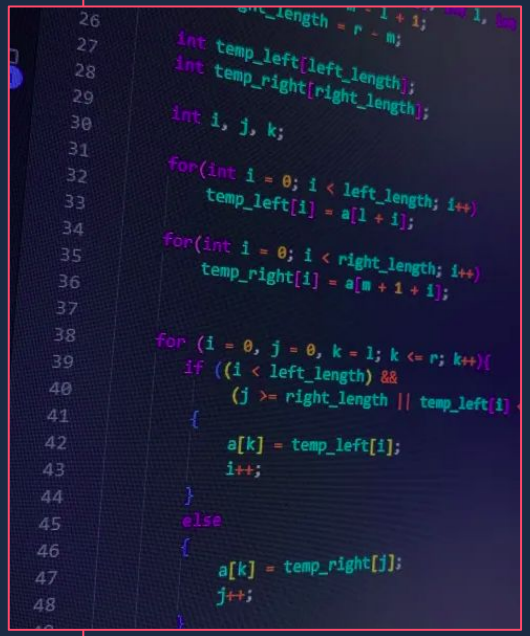
E se houver um desbalanceamento?

Caso haja um desbalanceamento entre quaisquer nós da árvore, precisaremos aplicar rotações para balancear a árvore de forma que a altura h da árvore se mantenha o mais próximo possível de log n.





Exemplos de árvores desbalanceadas

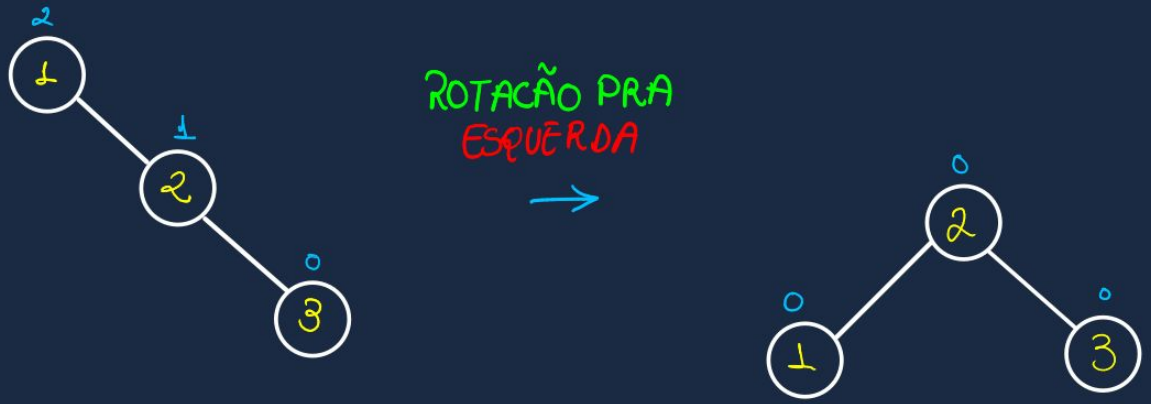

int temp_left[left_length];
int temp_right[right_length];
int i, j, k;
for(int i = 0; i < left_length; i++)
temp_left[i] = a[l + i];
for(int i = 0; i < right_length; i++)
temp_right[i] = a[m + 1 + i];
for (i = 0, j = 0, k = l; k <= r; k++){
if ((i < left_length) &&
(j >= right_length || temp_left[i]
{
a[k] = temp_left[i];
i++;
}
else
{
a[k] = temp_right[j];
j++;

2.1

Rotações

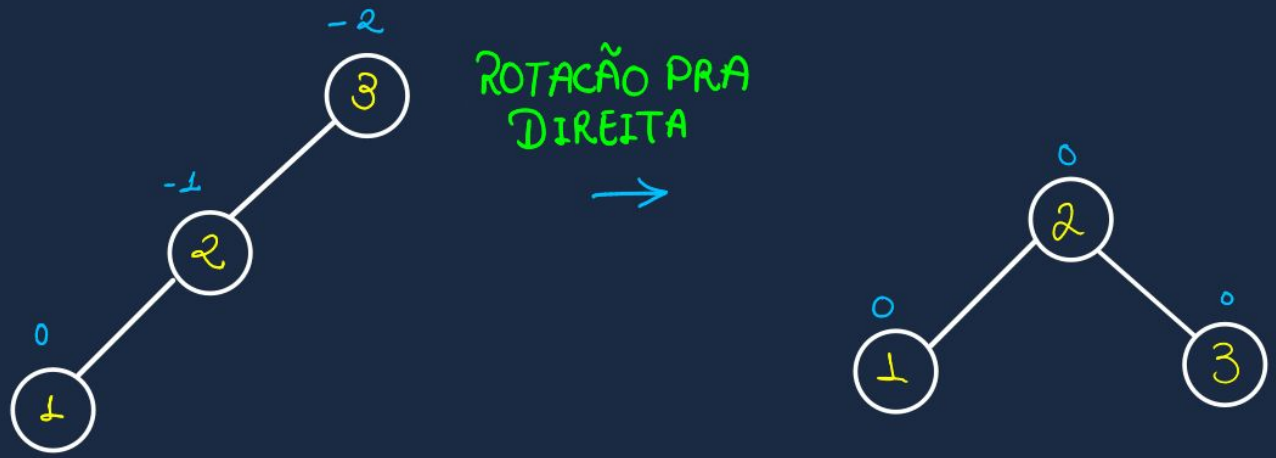
Rotação para a esquerda

A rotação para a esquerda é utilizada quando o fator de balanceamento de um nó é igual a 2. E o fator de balanceamento do filho da direta vale 0 ou 1. Abaixo segue um exemplo de rotação para a esquerda:



Rotação para a direita

A rotação para a direita é utilizada quando o fator de balanceamento de um nó é igual a -2. E o fator de balanceamento do filho da esquerda vale 0 ou -1. Abaixo segue um exemplo de rotação para a direita:



Rotações duplas

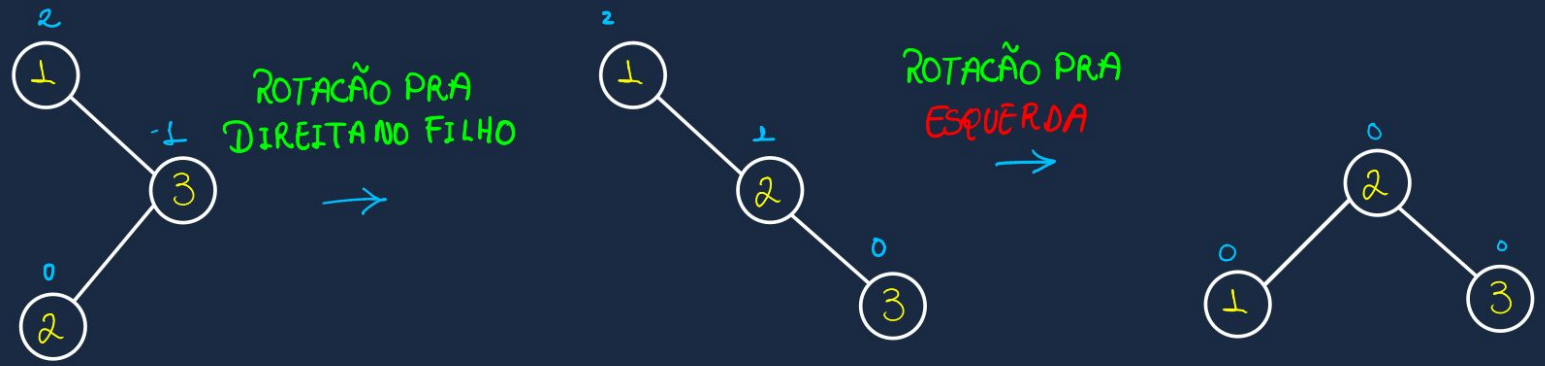
Algumas vezes precisaremos fazer mais de uma rotação para balancear um nó. Essas rotações são chamadas de rotações duplas

Dividimos essas rotações em:

- Rotação Direita Esquerda (filho pra direita e pai pra esquerda)
- Rotação Esquerda Direita (filho pra esquerda e pai pra direita)

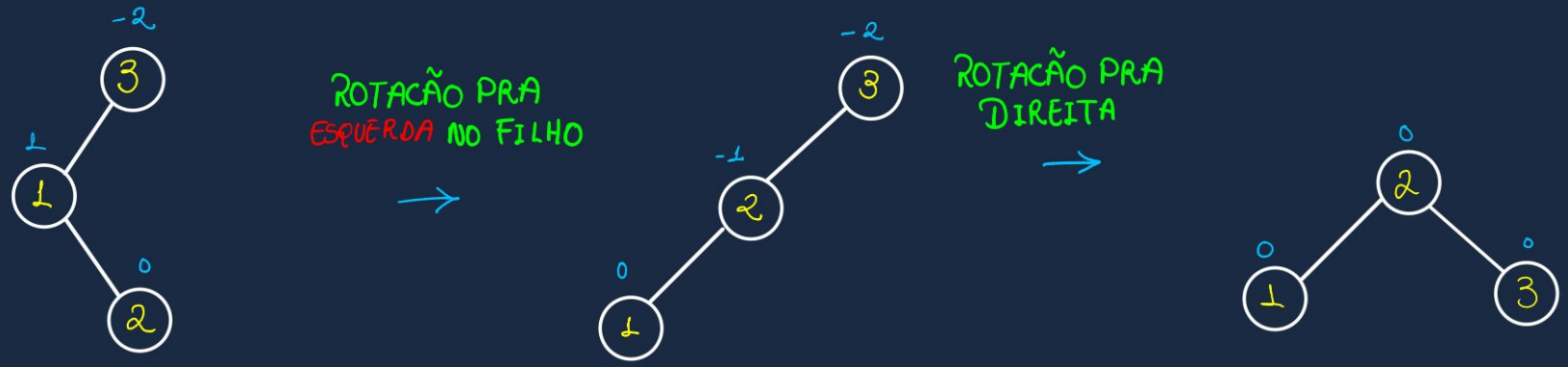
Rotação direita esquerda

A rotação RL é utilizada quando o fator de balanceamento de um nó é igual a 2. E o fator de balanceamento do filho da direta vale -1. Abaixo segue um exemplo de rotação direita esquerda:



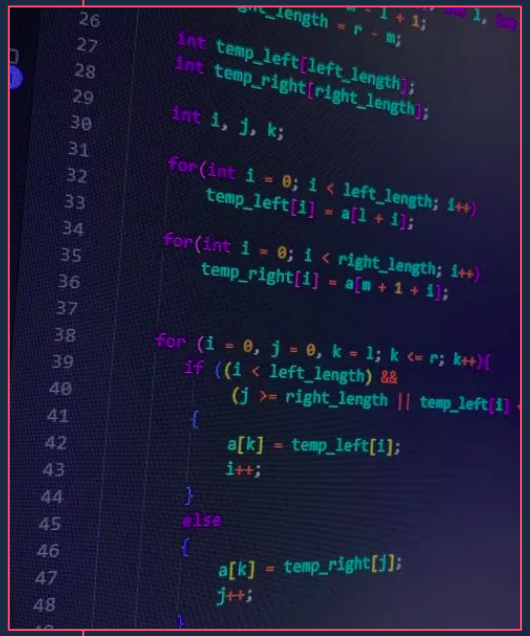
Rotação esquerda direita

A rotação LR é utilizada quando o fator de balanceamento de um nó é igual a -2. E o fator de balanceamento do filho da esquerda vale 1. Abaixo segue um exemplo de rotação esquerda direita:



Em resumo:

Balanceamento de um nó	Balanceamento do nó filho	Rotação
2	1	esquerda
	0	esquerda
	-1	direita esquerda
-2	1	esquerda direita
	0	direita
	-1	direita



3

Desenvolvimento do código

Comparação com a árvore binária

	Árvore binária		Árvore AVL	
	Médio caso	Pior caso	Médio caso	Pior caso
Inserção	O(log n)	O(n)	O(log n)	O(log n)
Deleção	O(log n)	O(n)	O(log n)	O(log n)
Busca	O(log n)	O(n)	O(log n)	O(log n)

Muito obrigado!!

Perguntas??

Referências:

- https://pt.wikipedia.org/wiki/%C3%81rvore_AVL
- https://github.com/free-educa/books/blob/main/books/Estrutura%20de%20Dados%2C%20algoritmos%2C%20an%C3%A1lise%20da%20complexidade%20e%20implementa%C3%A7%C3%B5es%20em%20Java%20e%20C%2C%20C--.pdf