

PIBIC – UFAPE

Análise da corretude de algoritmos gerados por Inteligência Artificial

Comparação deles com algoritmos de repositórios existentes

Índice.

3. OBJETIVOS

4. ALGORITMO DE DIJKSTRA

5. ANÁLISE DA CORRETEDE

5.1. CRIAÇÃO DO
REPOSITÓRIO

5.2. GRUPO DE CONTROLE

5.3 TECNOLOGIAS USADAS

5.4 TESTE

6. RESULTADO

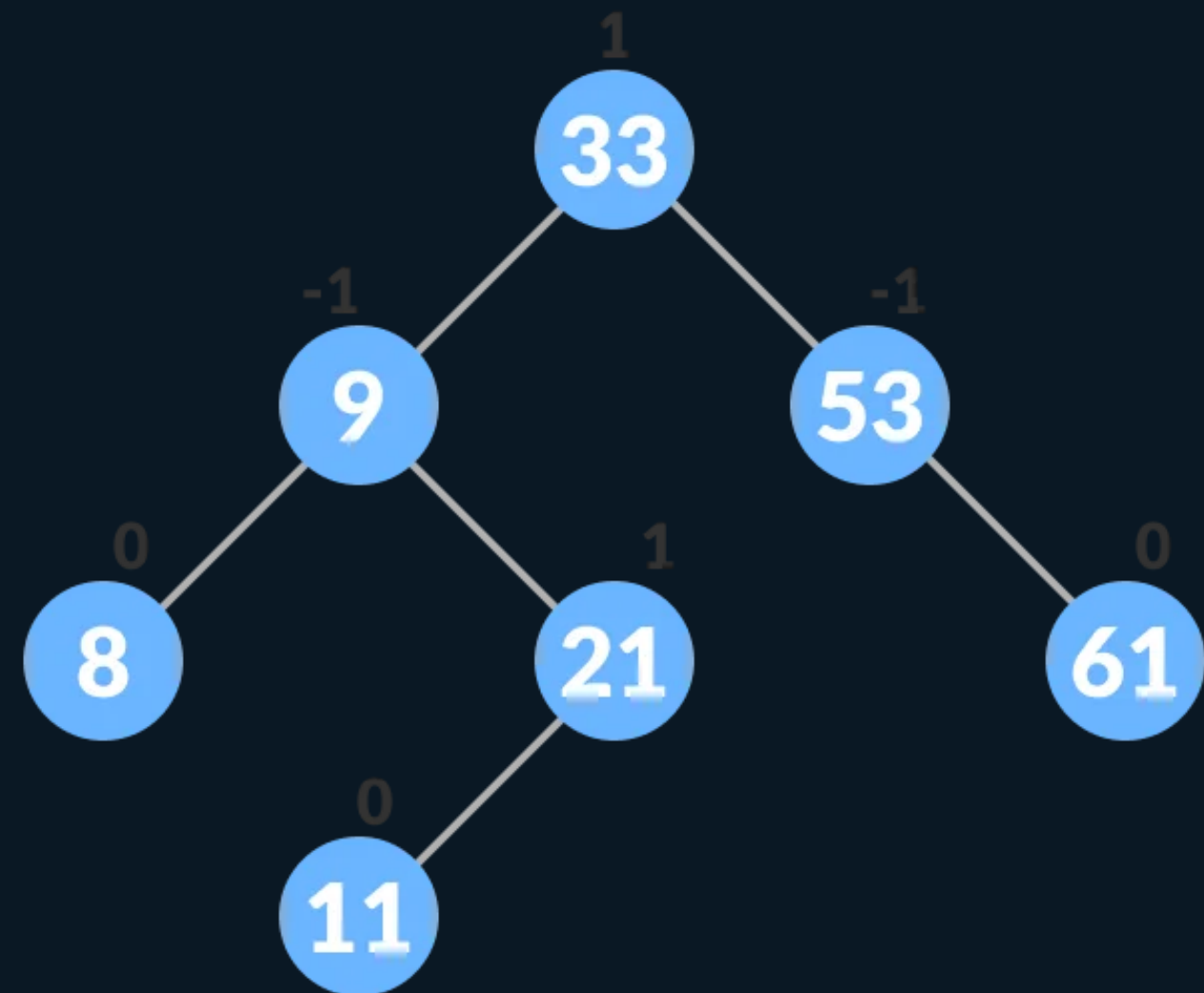
Objetivos.

Gerar o algoritmo da Árvore AVL usando Inteligências Artificiais

Comparar a execução deles com dois algoritmos já existentes em repositórios

Árvores AVL.

As árvores AVL foram inventadas por Adelson-Velsky e Landis em 1962. São árvores de busca binária que mantêm o balanceamento automático, garantindo que a diferença entre as alturas das subárvores esquerda e direita de cada nó seja no máximo 1. Isso é alcançado através de rotações que preservam a propriedade de árvore de busca binária. O nome AVL é uma homenagem aos seus inventores.



Árvores AVL.

O **fator de balanço** em árvores AVL é uma medida que indica o equilíbrio de um nó em relação aos seus descendentes e é calculado por.

$$FB(N) = Altura(N_d) - Altura(N_e)$$

Se $-1 \leq fb(n) \leq +1$, o nó está balanceado

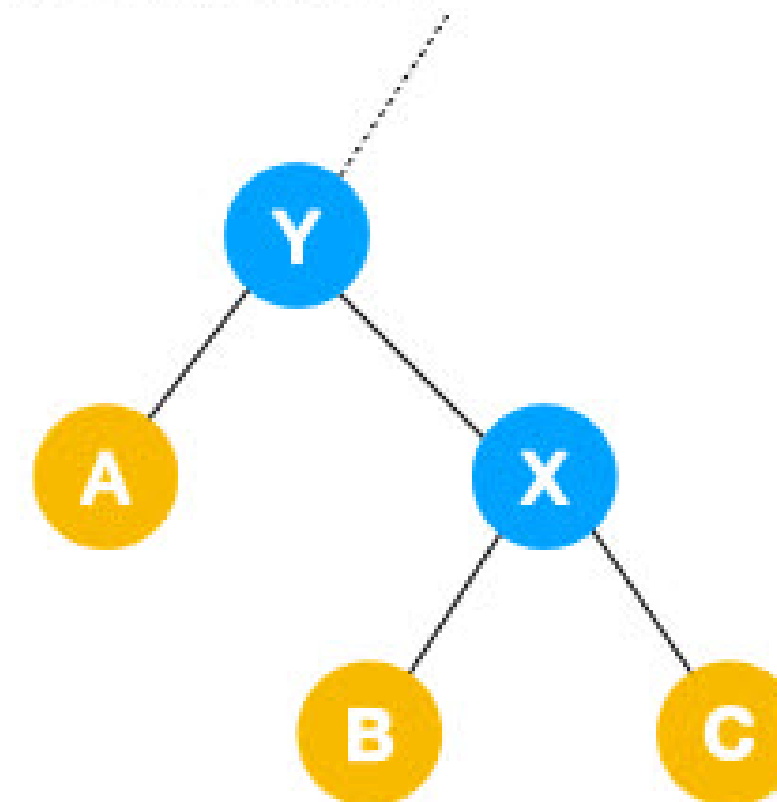
Se $fb(n) = -2$ ou $fb(n) = +2$, o nó está desbalanceado e é necessário rotacionar o nó para equilibrar a árvore

Árvores AVL.

Rotação Simples Direita

$$fb(x) = -2 \text{ e } fb(y) = -1$$

AVL Tree - Right Rotation

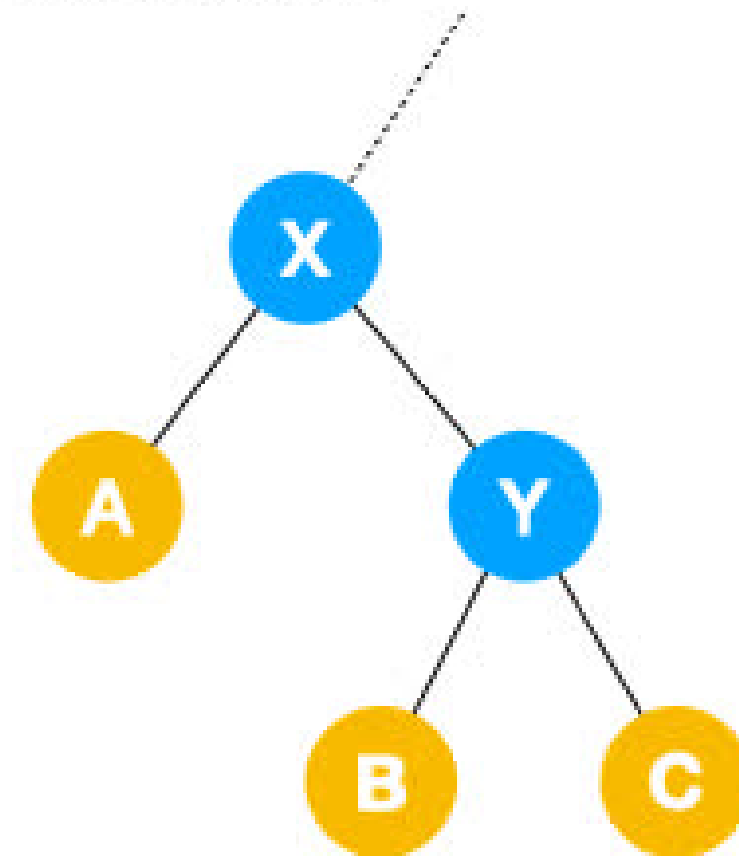


Árvores AVL.

Rotação Simples Esquerda

$$fb(x) = +2 \text{ e } fb(y) = +1$$

AVL Tree - Left Rotation

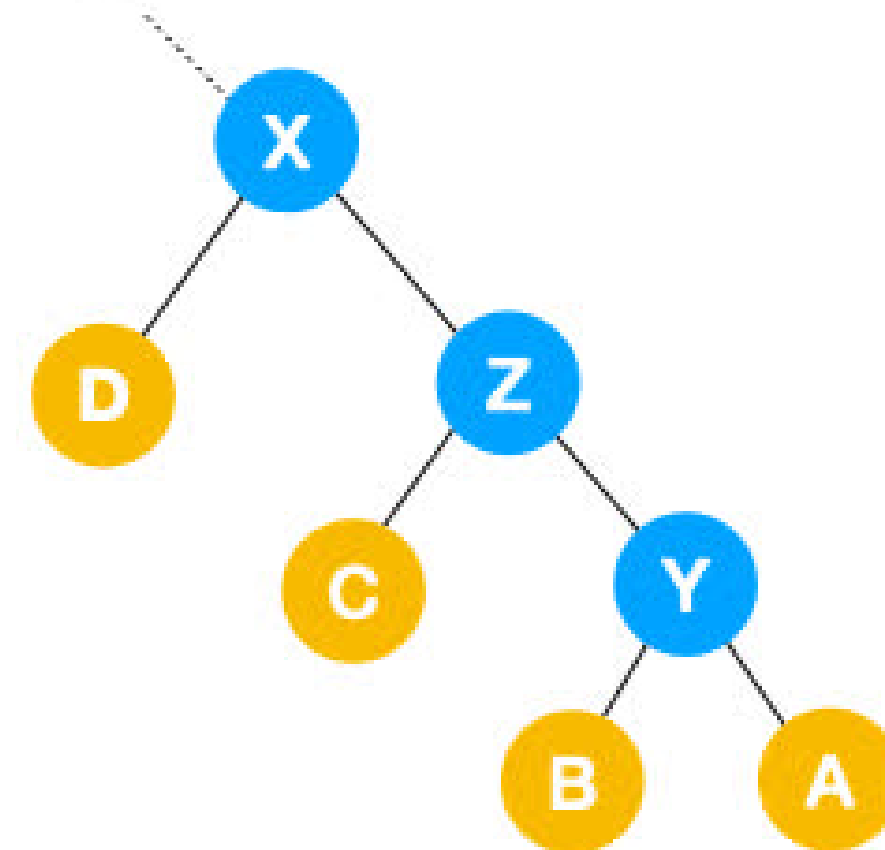


Árvores AVL.

Rotação Dupla à Esquerda

$$fb(x) = +2 \text{ e } fb(y) = -1$$

AVL Tree - Right-Left Rotation

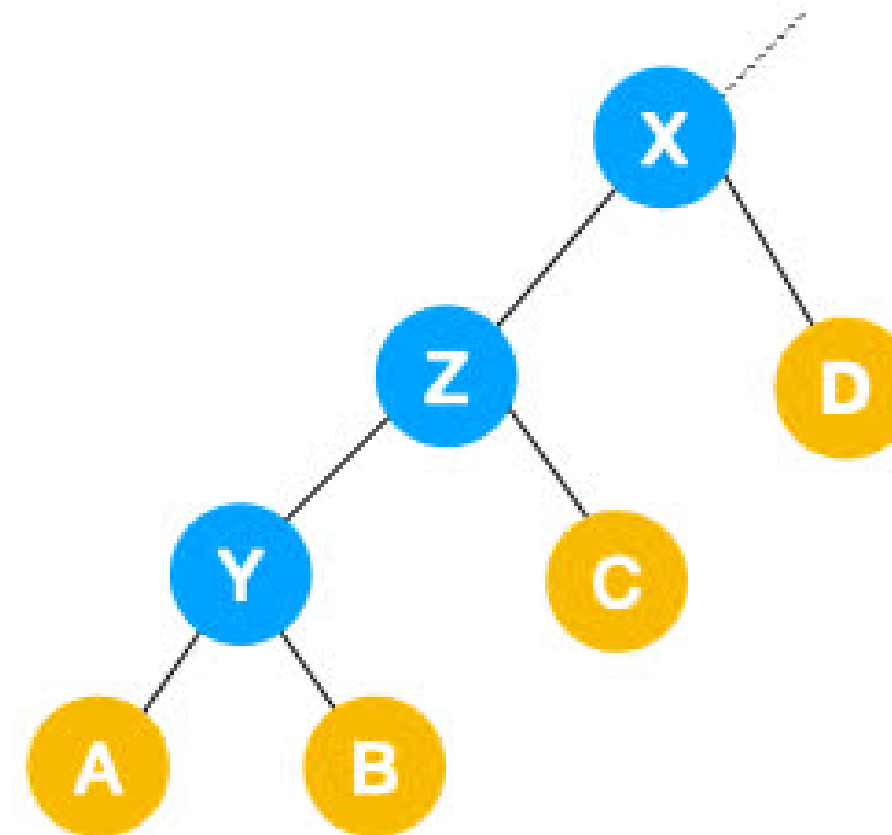


Árvores AVL.

Rotação Dupla Direita

$$fb(x) = -2 \text{ e } fb(y) = +1$$

AVL Tree - Left-Right Rotation



Análise da corretude.

- Criação de um repositório para guardar todos os códigos utilizados;
- Seleção de repositórios para o grupo de controle;
- Escolha das Inteligências Artificiais e das tecnologias necessárias para o desenvolvimento da análise;
- Testar os algoritmos gerados e os do repositório.

Análise da corretude.

- Consideramos a seguinte classificação quanto à corretude:

Correct: códigos sintática e semanticamente equivalentes ao resultado dos repositórios;

Plausible: Códigos sintaticamente corretos, mas com poucas falhas semânticas;

Invalid: Códigos sintaticamente corretos, mas semanticamente incorretos;

Incorrect: Códigos que não compilaram - sintaticamente incorretos.

Criação do repositório.

PIBIC / [com](#) / [avl](#) /

Add file

Jeffersonalanmm updates

0296668 · 6 minutes ago History

Name	Last commit message	Last commit date
..		
Main.java	AVL trees	last week
amazonQ.java	updates	6 minutes ago
blackbox.java	updates	6 minutes ago
chatGPT.java	updates	6 minutes ago
codeium.java	updates	6 minutes ago
copilot.java	AVL trees	last week
gemini.java	AVL trees	last week
rosetta.java	AVL trees	last week
theAlgorithms.java	AVL trees	last week

Grupo de Controle.

[LinkedIn](#)
[The Algorithms](#)
[chat](#)
[357 online](#)

[lp](#)
[receives 0.40 USD/week](#)
[Stars](#)
[365k](#)
[Follow us](#)

We are a group of programmers helping each other build new things, whether it be writing complex encryption programs, or simple ciphers. Our goal is to work together to document and model beautiful, helpful and interesting algorithms using code. We are an open-source community - anyone can contribute. We check each other's work, communicate and collaborate to solve problems. We strive to be welcoming, respectful, yet make sure that our code follows the latest programming guidelines.

[View all](#)

● Python ● C++ ● TypeScript ● HTML
● Zig

[algorithms](#)
[hacktoberfest](#)
[data-structures](#)
[algorithm](#)
[search](#)

Developer Program Member

[Report abuse](#)

Rosetta Code

[Main Page](#) [Discussion](#)

Read View source View history Tools ▾

Rosetta Code is a [programming chrestomathy](#) site. The idea is to present solutions to the same task in as many different languages as possible, to demonstrate how languages are similar and different, and to aid a person with a grounding in one approach to a problem in learning another. Rosetta Code currently has 1,264 [tasks](#), 404 [draft tasks](#), and is aware of 931 [languages](#), though we do not (and cannot) have solutions to every task in every language.

Places to start

Recently-Updated Tasks

Base64 decode data
Zsigmondy numbers
Append a record to the end of a text file
Quine
Anadromes
Barnsley fern

Tecnologias usadas.



Tecnologias usadas.

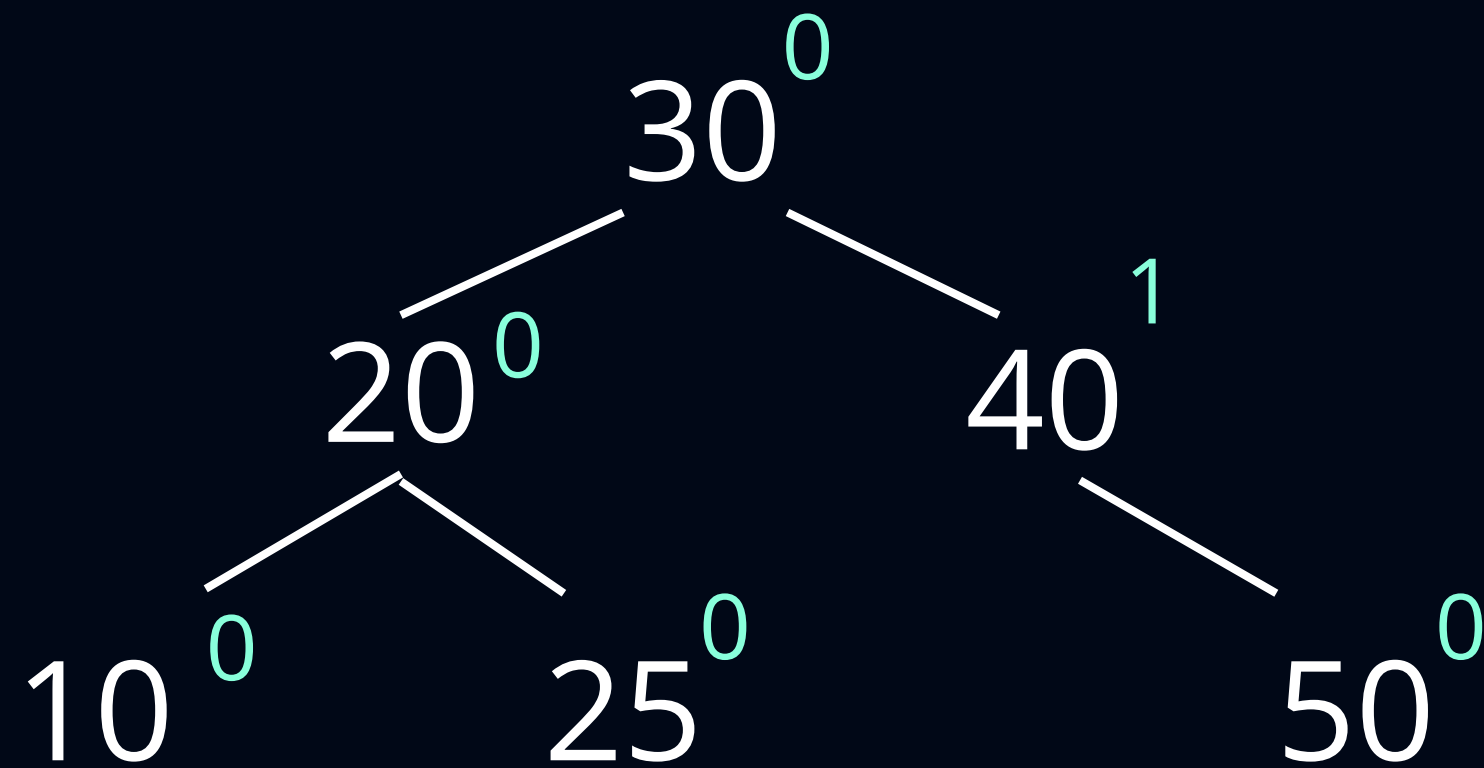
Please write your best implementation of the AVL tree algorithm in the Java programming language.

Teste.

```
public class Main {  
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | X  
    public static void main(String[] args) {  
        BlackBox blackBoxTree = new BlackBox();  
        int[] values = {10, 20, 30, 40, 50, 25};  
        System.out.println(x:"Árvore AVL resultante (BlackBox):");  
        testTree(blackBoxTree, values);  
  
        chatGPT chatGPTTree = new chatGPT();  
        System.out.println(x:"Árvore AVL resultante (chatGPT):");  
        testTree(chatGPTTree, values);  
  
        codeium codeiumTree = new codeium();  
        System.out.println(x:"Árvore AVL resultante (codeium):");  
        testTree(codeiumTree, values);  
  
        rosetta rosettaTree = new rosetta();  
        System.out.println(x:"Árvore AVL resultante (rosetta):");  
        testTree(rosettaTree, values);  
  
        theAlgorithms theAlgorithmsTree = new theAlgorithms();  
        System.out.println(x:"Árvore AVL resultante (theAlgorithms):");  
        testTree(theAlgorithmsTree, values);  
  
        gemini<Integer> geminiTree = new gemini<>();  
        System.out.println(x:"Árvore AVL resultante (gemini):");  
        testTree(geminiTree, values);  
    }  
}
```

```
Codeium: Refactor | Explain | Generate Javadoc | X  
static void testTree(gemini<Integer> tree, int[] values) {  
    for (int value : values) {  
        tree.root = tree.insert(tree.root, value);  
    }  
    tree.inOrderTraversal(tree.root);  
    System.out.println();  
}  
Codeium: Refactor | Explain | Generate Javadoc | X  
static void inOrder(com.avl.Node node) {  
    if (node != null) {  
        inOrder(node.left);  
        System.out.print(node.key + " ");  
        inOrder(node.right);  
    }  
}
```

Árvore AVL gerada.



Resultado.

	AmazonQ	BlackBox	ChatGPT	Codeium	Copilot	Gemini
<i>Correct</i>		X	X	X	X	X
<i>Plausible</i>						
<i>Incorrect</i>						
<i>Invalid</i>	X					

Referências.

- CORSO, Vincenzo et al. **Generating Java Methods: An Empirical Assessment of Four AI-Based Code Assistants**. arXiv preprint arXiv:2402.08431, 2024. Acesso em fevereiro de 2024.
- LIBÓRIO, Felipe Tenório de Holanda Rocha. **Análise da Ferramenta ChatGPT para a Geração de Código-fonte de Algoritmos de Ordenação em Diferentes Linguagens de Programação**. Brasil. Acesso em fevereiro de 2024.
- GALLES, David. **Visualizing AVL Trees**. Disponível em: <https://www.cs.usfca.edu/~galles/visualization/AVLtree.html>. Acesso março de 2024.
- VANDERLEI, Igor Medeiros. **Árvores Balanceadas**. Apresentação de slides. Universidade Federal Rural de Pernambuco. Acesso em março de 2024.