

**RENATO ALEXANDRE DE CARVALHO**

**EDUARDO PIELICH SANCHEZ**

**AVALIAÇÃO BIMESTRAL N3: Estrutura de Dados**

São Caetano do Sul – SP

2024

# **1. RESUMO**

## **1.1. Contexto**

Este documento registra aspectos conceituais e técnicos da atividade realizada em sala de aula na FATEC Antônio Russo, da matéria de Estrutura de Dados, ministrada pelo professor Carlos Veríssimo Pereira. A proposta da atividade consiste na realização de um algoritmo com implementação na linguagem Java, utilizando paradigmas de pesquisa e recursividade.

A solução desenvolvida necessitava implementar uma pesquisa em árvore binária, utilizando um dos três paradigmas de pesquisa: em ordem, pré-ordem e pós-ordem.

## **1.2. Propósito**

O propósito deste documento é reunir as informações relevantes em relação ao programa, tais como: Conceito de árvore binária, conceito de recursividade e modelo de pesquisa. Além de apresentar o próprio código e seus resultados.

## **1.3. Metodologia**

Para cumprir com a proposta da atividade, foi implementado um algoritmo de busca em ordem baseado em técnicas avançadas de estrutura de dados. No início, foi realizada uma análise detalhada das árvores binárias, compreendendo seus conceitos fundamentais e as operações envolvidas. Em seguida, o algoritmo foi desenvolvido e testado em diferentes cenários para verificar sua eficácia e funcionamento.

## **1.4. Resultados**

Os resultados obtidos no estudo englobam tanto a implementação da árvore binária de forma eficiente quanto o modelo de pesquisa escolhido. Durante a execução do código, foi possível perceber que a árvore binária foi construída de maneira correta e com os devidos critérios de organização. Sem contar que os testes realizados com diferentes números provaram que a busca em ordem também funciona com precisão.

Por fim, os resultados também demonstraram a importância da recursividade, que desempenhou um papel fundamental para a implementação da pesquisa em ordem e aperfeiçoou a eficácia e visibilidade do código.

### **1.5. Conclusão**

Em suma, a atividade realizada proporcionou uma compreensão aprofundada dos conceitos de árvore binária, recursividade e modelos de pesquisa. A implementação do algoritmo de busca se mostrou funcional, imprimindo os valores em ordem crescente. Ademais, a recursividade foi aplicada tanto para a busca, tanto para a inserção de valores na árvore binária.

## **2. ARGUMENTAÇÃO TEÓRICA**

### **2.1. Modelo de Pesquisa**

O modelo de pesquisa utilizado na atividade foi em ordem. No código, o método “consulta” ficou responsável por percorrer a árvore binária e imprimir os valores de cada nó em ordem crescente, verificando os nós da esquerda para a direita.

### **2.2. Recursividade**

A recursividade é uma técnica de programação que envolve a chamada de uma função a si mesma, permitindo que um problema complexo seja dividido, tornando-o mais simples. A recursividade é um conceito fundamental na programação e desempenha um papel crucial na implementação eficiente e funcional de algoritmos em estruturas de dados complexas, como por exemplo, árvores binárias.

No caso dessa atividade, a recursividade foi utilizada para percorrer por todos os nós da árvore em ordem crescente. A função vai chamando a si própria várias vezes até que todos os nós tenham sido visitados. Com isso, a pesquisa da árvore binária é feita de maneira mais sistemática e organizada, além de deixar o código mais legível e fácil de entender.

### **2.3. Árvore Binária**

Uma árvore binária é uma estrutura de dados que consiste basicamente em nós conectados por arestas, onde cada nó só pode ter no máximo dois filhos: um nó à esquerda e outro à direita. O primeiro nó, o que está no topo da árvore, é chamado de raiz, e os nós que não possuem nenhum filho são chamados de folhas. A construção da árvore binária é baseada em uma simples regra: valores menores do que o valor do nó atual são colocados à esquerda, enquanto valores maiores são colocados à direita.

A árvore binária serve para organizar dados de maneira hierárquica, permitindo que operações de busca, inserção e exclusão de dados sejam feitas de maneira mais rápida do que estruturas lineares, como listas ou arrays.

No caso dessa atividade, a árvore binária foi implementada de forma que o usuário possa inserir a quantidade de números de sua escolha. A estrutura da árvore é construída automaticamente, de acordo com os valores inseridos pelo usuário. Depois, o usuário pode realizar a busca em ordem, e o sistema retornará os valores dos nós em ordem crescente.

### 3. Resultados obtidos

#### 3.1. Execução do programa

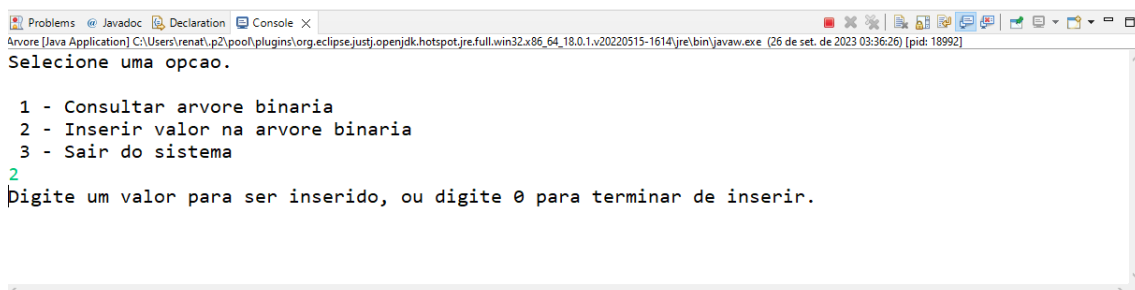
Ao iniciar o programa, é apresentado ao usuário um menu de opções, onde ele pode consultar a árvore binária, inserir valores na árvore binária ou sair do sistema. Caso ele tente consultar a árvore binária enquanto ela estiver vazia, uma mensagem de erro aparecerá.



```
Problems | Javadoc | Declaration | Console X
Arvore [Java Application] C:\Users\renat\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_18.0.1.v20220515-1614\jre\bin\javaw.exe (26 de set. de 2023 03:36:26) [pid: 18992]
Selecione uma opcao.

1 - Consultar arvore binaria
2 - Inserir valor na arvore binaria
3 - Sair do sistema
```

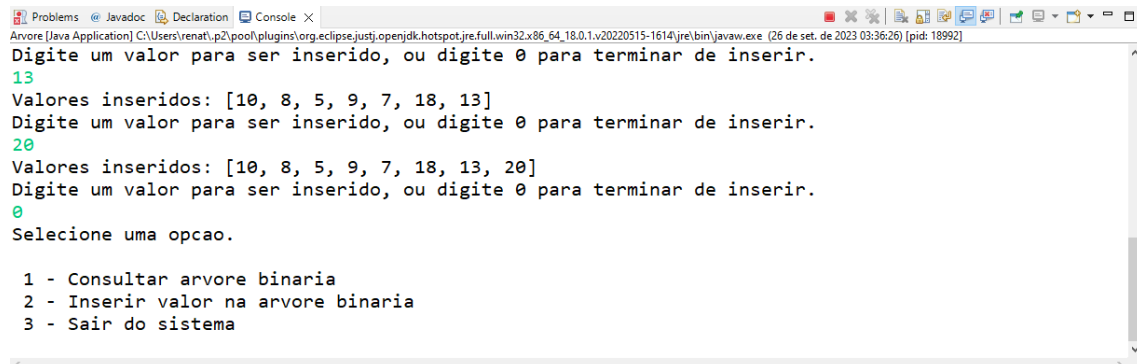
Após selecionar a opção de inserir valores na árvore binária, o usuário poderá adicionar quantos valores ele desejar.



```
Problems | Javadoc | Declaration | Console X
Arvore [Java Application] C:\Users\renat\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_18.0.1.v20220515-1614\jre\bin\javaw.exe (26 de set. de 2023 03:36:26) [pid: 18992]
Selecione uma opcao.

1 - Consultar arvore binaria
2 - Inserir valor na arvore binaria
3 - Sair do sistema
2
Digite um valor para ser inserido, ou digite 0 para terminar de inserir.
```

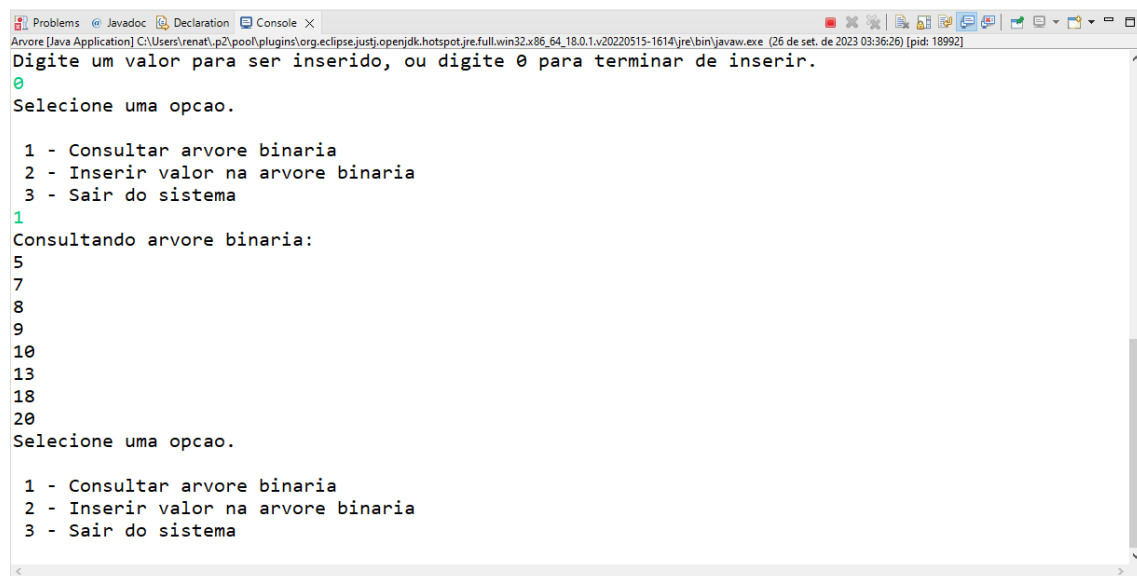
Nesse caso, foram inseridos os valores: 10, 8, 5, 9, 7, 18, 13, e 20. Quando o usuário não quiser mais inserir valores, ele pode digitar 0 para terminar de inserir, retornando ao menu principal.



```
Problems | Javadoc | Declaration | Console X
Arvore [Java Application] C:\Users\renat\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.1.v20220515-1614\jre\bin\javaw.exe (26 de set. de 2023 03:36:26) [pid: 18992]
Digite um valor para ser inserido, ou digite 0 para terminar de inserir.
13
Valores inseridos: [10, 8, 5, 9, 7, 18, 13]
Digite um valor para ser inserido, ou digite 0 para terminar de inserir.
20
Valores inseridos: [10, 8, 5, 9, 7, 18, 13, 20]
Digite um valor para ser inserido, ou digite 0 para terminar de inserir.
0
Selecione uma opcao.

1 - Consultar arvore binaria
2 - Inserir valor na arvore binaria
3 - Sair do sistema
```

Agora, o usuário pode selecionar a opção 1 para consultar a árvore binária, que retornará os valores inseridos em ordem crescente, que foi o modelo de pesquisa escolhido.



```
Problems | Javadoc | Declaration | Console X
Arvore [Java Application] C:\Users\renat\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.18.0.1.v20220515-1614\jre\bin\javaw.exe (26 de set. de 2023 03:36:26) [pid: 18992]
Digite um valor para ser inserido, ou digite 0 para terminar de inserir.
0
Selecione uma opcao.

1 - Consultar arvore binaria
2 - Inserir valor na arvore binaria
3 - Sair do sistema
1
Consultando arvore binaria:
5
7
8
9
10
13
18
20
Selecione uma opcao.

1 - Consultar arvore binaria
2 - Inserir valor na arvore binaria
3 - Sair do sistema
```

### 3.2. Cálculo do tempo

```
public void consulta(No no) {  
    if (no != null) {  
        consulta(no.Esquerdo);  
        System.out.println(no.Conteudo);  
        consulta(no.Direito);  
    }  
}
```

Primeira linha: `if (no != null) {`

Vale t.

Segunda linha: `consulta(no.Esquerdo);`

Vale n.

Terceira linha: `System.out.println(no.Conteudo);`

Vale t.

Quarta linha: `consulta(no.Direito);`

Vale n.

### CÁLCULO

$$t + n(t + n + t + n) + t + n(t + n + t + n) =$$

$$t + n(2t + 2n) \cdot 2 =$$

$$\underline{t + 4tn + 4n^2}$$

**RESULTADO:**  $t + 4tn + 4n^2$ , onde “n” é o número de nós na árvore.