Vamos analisar como os métodos **inOrder** e **inOrderRec** funcionam para percorrer os nós de uma árvore de busca binária e imprimir os valores em ordem crescente.

**Estrutura dos Métodos**

1. **inOrder:**

public String inOrder(){

        if(estaVazia()) return "arvore vazia";

            return inOrderRec(raiz);

    }

1. **inOrderRec:**

public String inOrderRec (No atual) {

  if (atual == null) return "";

return

inOrderRec(atual.getEsquerda()) +

          " " + atual.getInfo() +

          inOrderRec(atual.getDireita());

}

**Funcionamento**

**Método inOrder**

Este é o método inicial chamado para começar a travessia da árvore. Ele faz o seguinte:

* Verifica se a árvore está vazia chamando o método **estaVazia()**. Se estiver vazia, retorna a string **"arvore vazia"**.
* Se a árvore não estiver vazia, chama o método recursivo **inOrderRec** passando a raiz da árvore como primeiro argumento.

**Método inOrderRec**

Este é o método recursivo que faz a travessia in-order. A travessia in-order de uma árvore binária de busca visita os nós na seguinte ordem: subárvore esquerda, nó atual, subárvore direita.

* **Condição de Parada:** Se o nó atual for **null**, retorna uma string vazia **""**. Esta é a condição de parada para a recursão.
* **Chamada Recursiva:**
  1. **Visita a subárvore esquerda:** Chama **inOrderRec(atual.getEsquerda())** para obter os valores da subárvore esquerda em ordem crescente.
  2. **Visita o nó atual:** Adiciona o valor do nó atual **atual.getInfo()** seguido de um espaço **" "**.
  3. **Visita a subárvore direita:** Chama **inOrderRec(atual.getDireita())** para obter os valores da subárvore direita em ordem crescente.
* **Concatenação:** Junta as três partes (subárvore esquerda, nó atual, subárvore direita) em uma única string e a retorna.

**Exemplo Prático**

Vamos considerar uma árvore de busca binária com a seguinte estrutura:

4

/ \

2 6

/ \ / \

1 3 5 7

**Passos da Travessia**

1. **Chamada Inicial:**
   * **inOrder()** é chamado.
   * Verifica **estaVazia()**, que retorna **false**.
   * Chama **inOrderRec(raiz)**, onde **raiz** é o nó com valor **4**.

**4**

/ \

2 6

/ \ / \

1 3 5 7

1. **Primeira Chamada Recursiva:**
   * **inOrderRec(4)**:
     + Chama **inOrderRec(2)** para a subárvore esquerda de **4**.

**4**

**/** \

**2** 6

/ \ / \

1 3 5 7

1. **Segunda Chamada Recursiva:**
   * **inOrderRec(2)**:
     + Chama **inOrderRec(1)** para a subárvore esquerda de **2**.

**4**

**/** \

**2** 6

**/** \ / \

1 3 5 7

1. **Terceira Chamada Recursiva:**
   * **inOrderRec(1)**:
     + Chama **inOrderRec(null)** para a subárvore esquerda de **1**, retorna **""**.
     + Adiciona o valor do nó atual, **"1 "**, na string de retorno.
     + Chama **inOrderRec(null)** para a subárvore direita de **1**, retorna **""**.
   * Retorna **"1 "** para a Segunda Chamada Recursiva:

**4**

**/** \

**2** 6

**/** \ / \

1 3 5 7

1. **Voltando à Segunda Chamada Recursiva:**
   * **inOrderRec(2)**:
     + Recebe **"1 "**, retornado pela subárvore esquerda.
     + Adiciona o valor do nó atual **"2 "**, obtendo **"1 2 "**.
     + Chama **inOrderRec(3)** para a subárvore direita de **2**.

**4**

**/** \

**2** 6

**/** **\** / \

**1** **3** 5 7

1. **Quarta Chamada Recursiva:**
   * **inOrderRec(3)**:
     + Chama **inOrderRec(null)** para a subárvore esquerda de **3**, retorna **""**.
     + Adiciona o valor do nó atual, **"3 "**, na string de retorno.
     + Chama **inOrderRec(null)** para a subárvore direita de **3**, retorna **""**.
     + Retorna **"3 "** para a Segunda Chamada Recursiva.

**4**

**/** \

**2** 6

**/** **\** / \

**1** **3** 5 7

1. **Voltando à Segunda Chamada Recursiva:**
   * **inOrderRec(2)**:
     + Concatena **"1 2 " + "3 "**, obtendo **"1 2 3 "**.
     + Retorna **"1 2 3 "** para a Primeira Chamada Recursiva.

**4**

**/** \

**2** 6

**/** **\** / \

**1** **3** 5 7

1. **Voltando à Primeira Chamada Recursiva:**
   * **inOrderRec(4)**:
     + Concatena **"1 2 3 " + "4 "**, obtendo **"1 2 3 4 "**.
     + Chama **inOrderRec(6)** para a subárvore direita de **4**.

**4**

**/** **\**

**2** **6**

**/** **\** / \

**1** **3** 5 7

1. **Quinta Chamada Recursiva:**
   * **inOrderRec(6)**:
     + Chama **inOrderRec(5)** para a subárvore esquerda de **6**.

**4**

**/** **\**

**2** **6**

**/** **\** **/** \

**1** **3** **5** 7

1. **Sexta Chamada Recursiva:**
   * **inOrderRec(5)**:
     + Chama **inOrderRec(null)** para a subárvore esquerda de **5**, retorna **""**.
     + Adiciona o valor do nó atual, **"5 "**, na string de retorno.
     + Chama **inOrderRec(null)** para a subárvore direita de **5**, retorna **""**.
     + Retorna **"5 "** para a Quinta Chamada Recursiva.

**4**

**/** **\**

**2** **6**

**/** **\** **/** \

**1** **3** **5** 7

1. **Voltando à Quinta Chamada Recursiva:**
   * **inOrderRec(6)**:
     + Adiciona o valor do nó atual, **"6 "**, à string de retorno da subárvore esquerda, obtendo **"5 6 "**.
     + Chama **inOrderRec(7)** para a subárvore direita de **6**.

**4**

**/** **\**

**2** **6**

**/** **\** **/** **\**

**1** **3** **5** **7**

1. **Sétima Chamada Recursiva:**
   * **inOrderRec(7)**:
     + Chama **inOrderRec(null)** para a subárvore esquerda de **7**, retorna **""**.
     + Adiciona **"7 "**.
     + Chama **inOrderRec(null)** para a subárvore direita de **7**, retorna **""**.
     + Retorna **"7 "** para a Quinta Chamada Recursiva.

**4**

**/** **\**

**2** **6**

**/** **\** **/** **\**

**1** **3** **5** **7**

1. **Voltando à Quinta Chamada Recursiva:**
   * **inOrderRec(6)**:
     + Concatena **"5 6 " + "7 "**, obtendo **"5 6 7 "**.
     + Retorna **"5 6 7 "** para a Primeira Chamada Recursiva.
2. **Voltando à Primeira Chamada Recursiva:**
   * **inOrderRec(4)**:
     + Concatena **"1 2 3 4 " + "5 6 7 "**, obtendo **"1 2 3 4 5 6 7 "**.

**Resultado Final**

A impressão dos valores em ordem será: **1 2 3 4 5 6 7**, o que confirma que os métodos funcionam corretamente para percorrer a árvore de busca binária em ordem crescente.