

Árvore Binária de Pesquisa ou Árvore Binária de Busca - ABB

Prof. D.Sc. Saulo Ribeiro

Sumário

- Objetivo de Aprendizagem
- Revisão de Árvore Binária
- Revisão de recursão
- Árvore Binária de Busca – ABB
- Conceitos Básicos de ABB
- Busca na ABB
- Inserção na ABB
- Remoção na ABB
- Por que uma Árvore Binária de Busca é boa?
- Próximos Passos
- Exercícios de fixação
- Bibliografia

Objetivo de Aprendizagem

- Identificar uma árvore binária de busca/pesquisa
- Aprender os algoritmos de busca, inserção e remoção
- Identificar aplicações da ABB
- Saber porque uma ABB é eficiente na busca

Aplicações de ABB

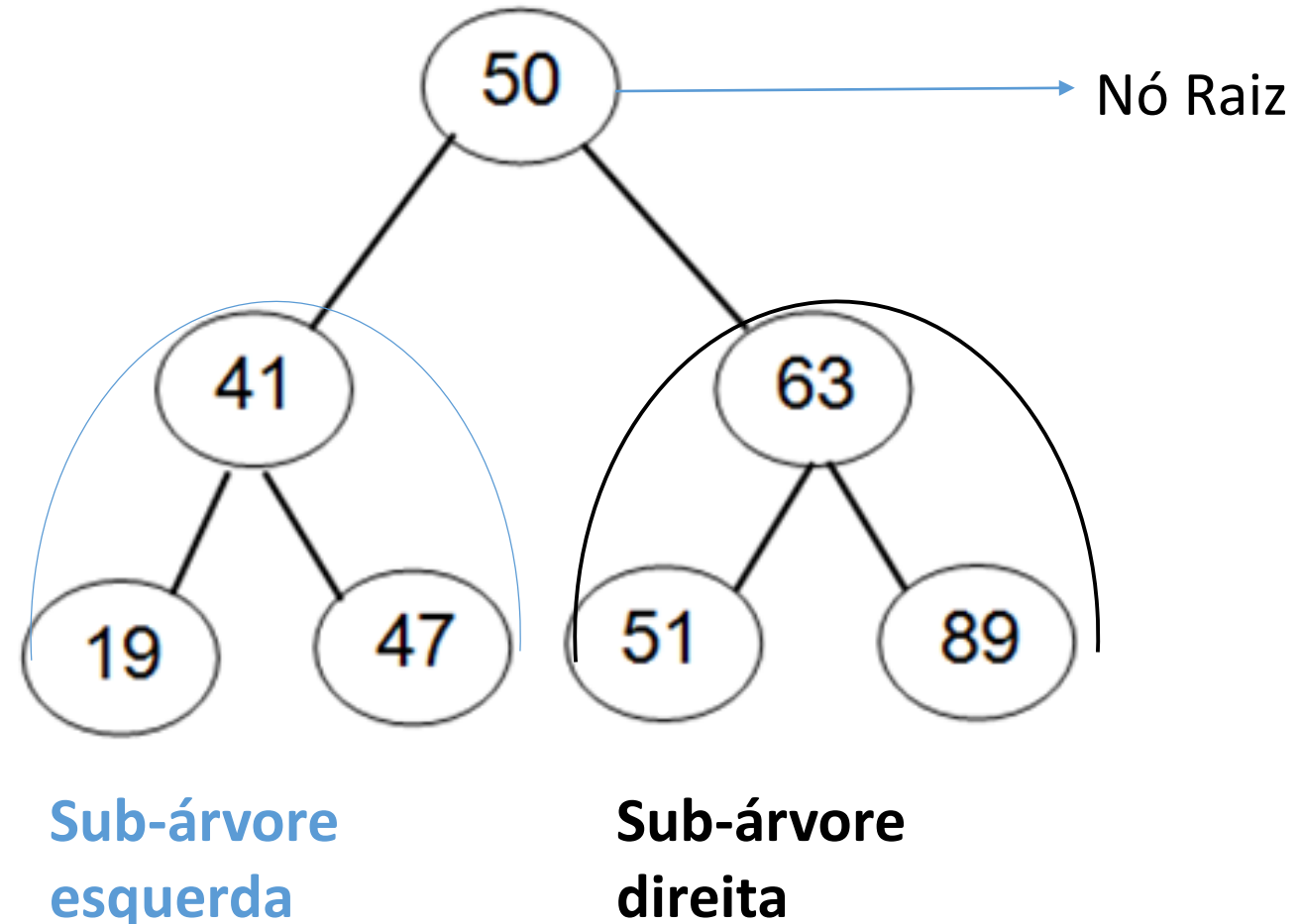
- Uma ABB permite consultas rápidas, mesmo quando a quantidade de elementos é grande.
 - Dicionário
- Provê uma estrutura hierárquica para um sistema de arquivo
- etc

Revisando Árvore Binária

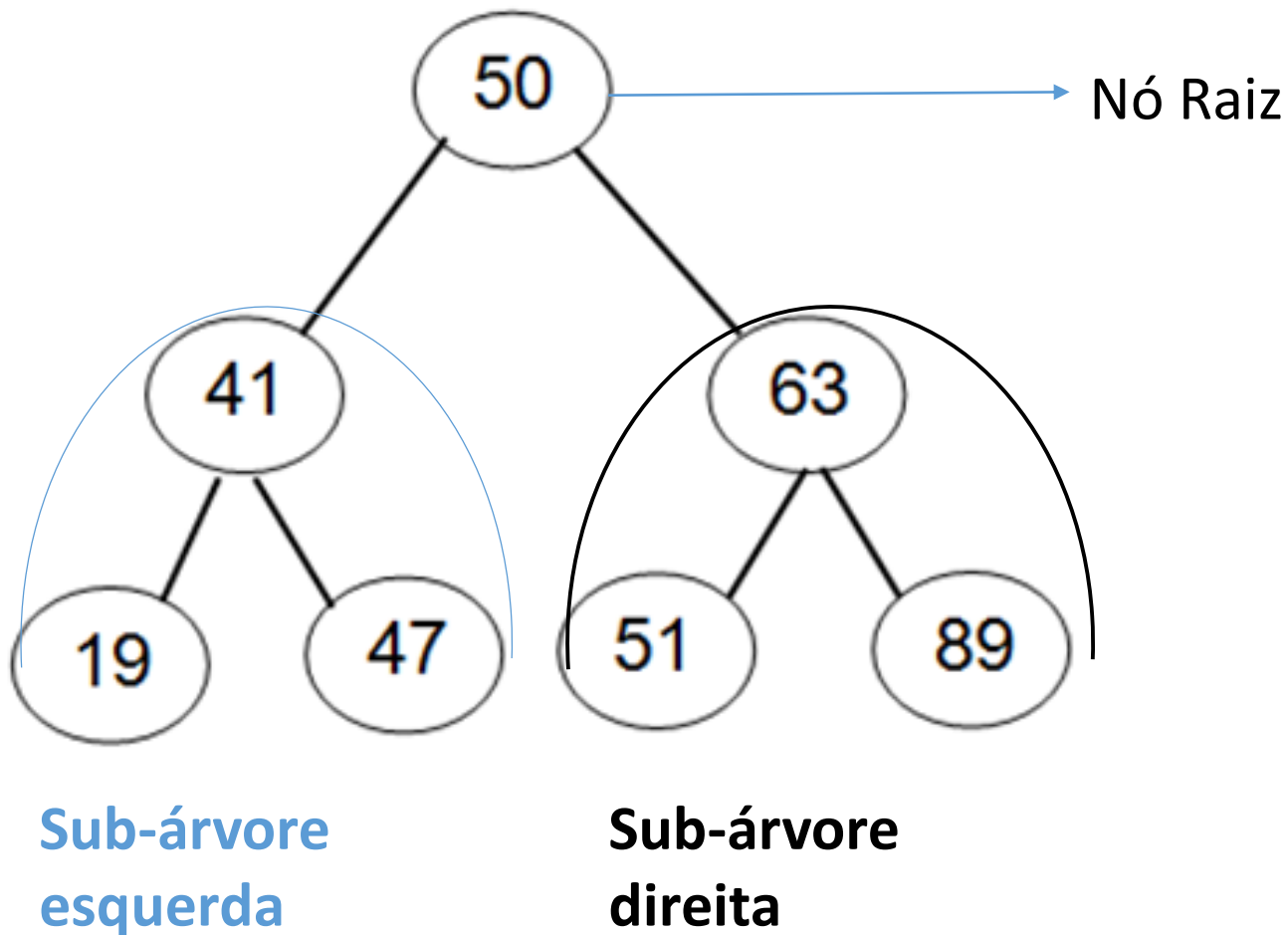
- Conceito de Árvore Binária
- Raiz
- Nós (raiz, interno, folha)
- Sub-árvores esquerda e direita
- Altura da árvore binária
- Níveis

Revisando Árvore Binária

- **Conceito básico:**
- Uma **árvore binária** é uma estrutura **recursiva**, composta por um elemento, denominado de **raiz**, e por duas árvores binárias associadas, denominadas **sub-árvore esquerda** e **sub-árvore direita**.



Revisando Árvore Binária

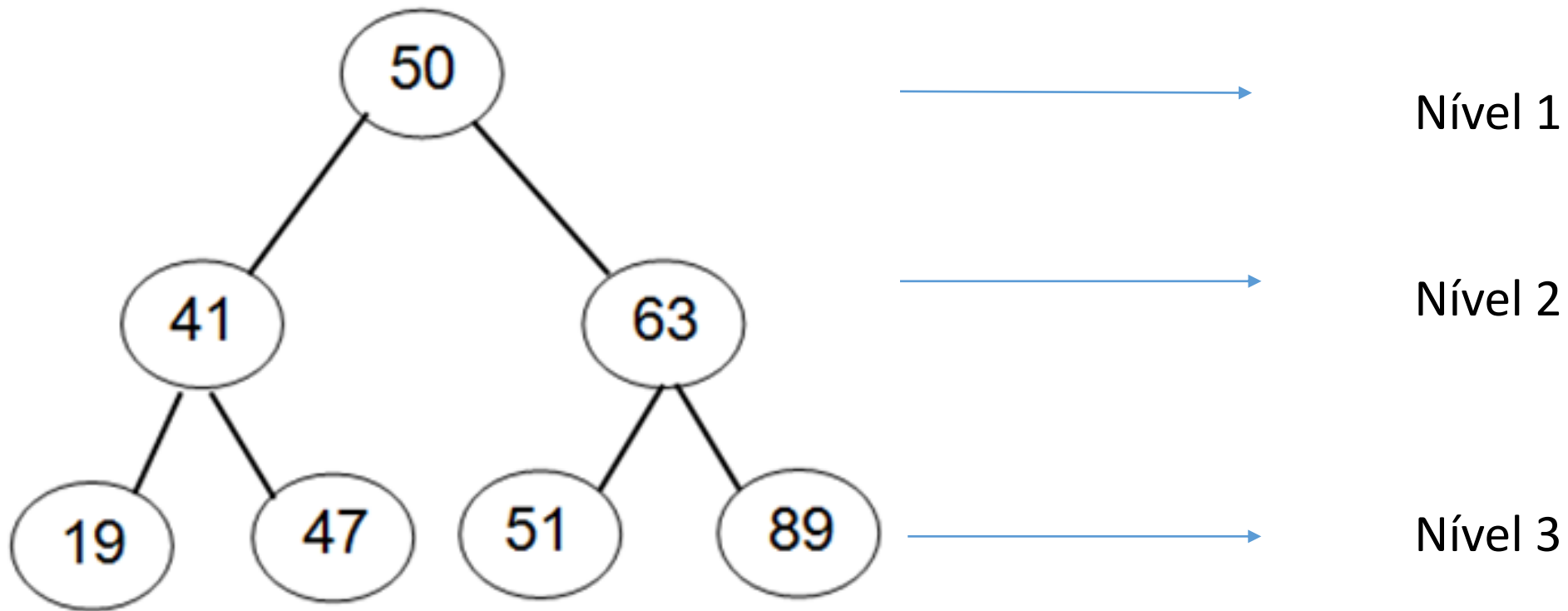


Árvore vazia



**Nó = info,
ref. subArvEsq ,
ref. subArvDir**

Revisando Árvore Binária



Ao Elaborar Algoritmos recursivos...

- Liste todos os casos, identificando-os como Caso 1, Caso 2, e assim por diante;
- Identifique os casos em que é possível dar uma resposta de imediato, e proponha a resposta (Pontos de Parada);
- Identifique os casos em que não é possível resolver de imediato, e procure resolver com uma ou mais chamadas recursivas.

Revisando Algoritmos Recursivos: Fatorial

Fatorial - Definição:

- (i) Fatorial de 0 é 1
-
- (ii) Fatorial de N é $N * \text{Fatorial}(N - 1)$

Dica:

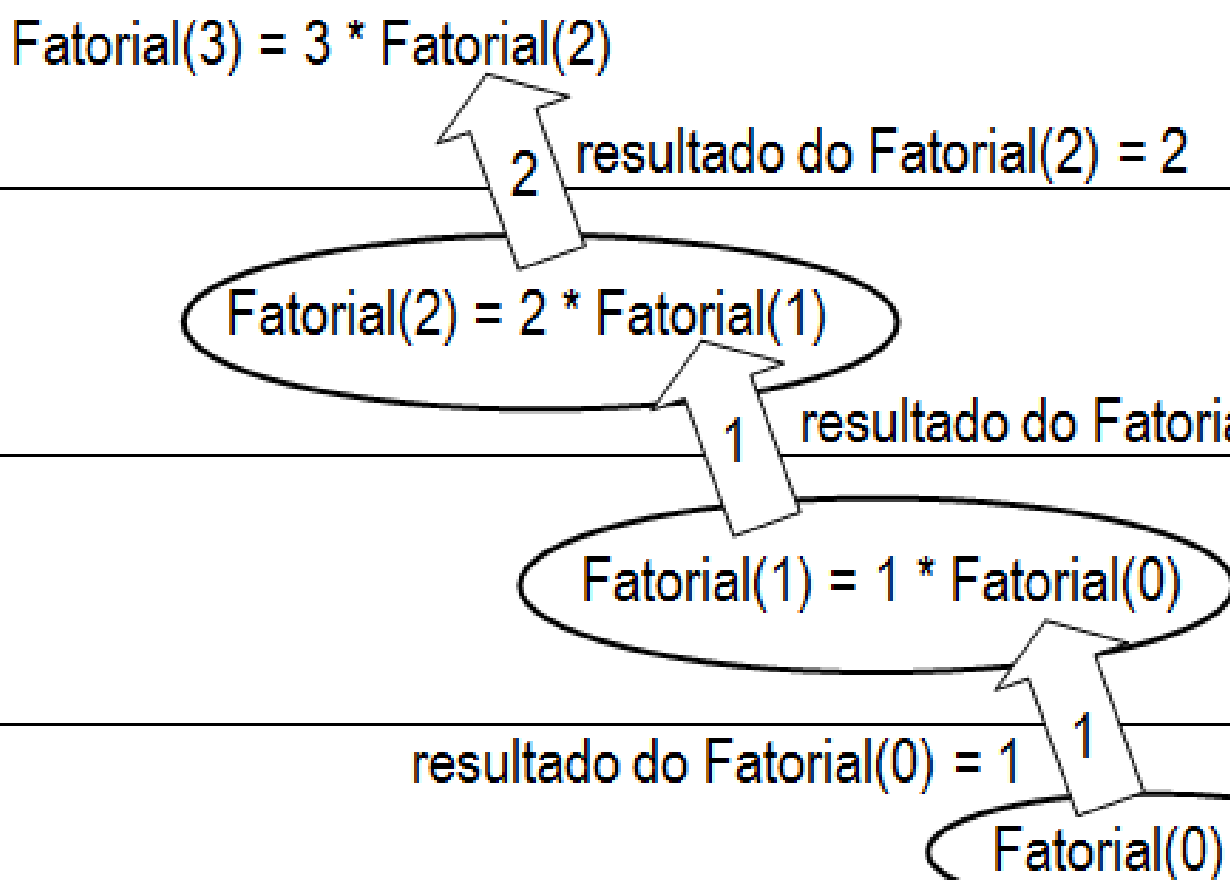
É possível resolver de imediato ↑

Deixamos para resolver depois ↓

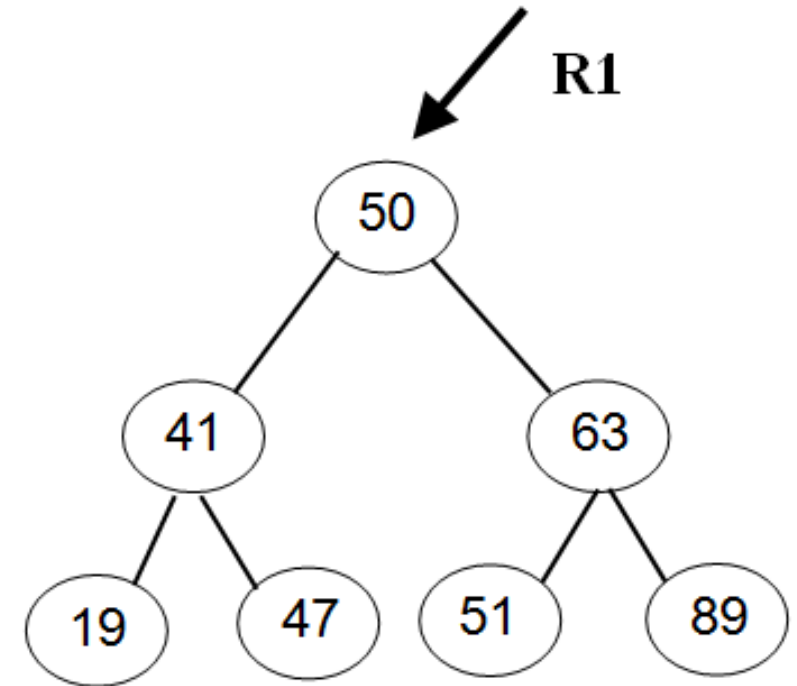
Fatorial - Implementação Recursiva:

```
Inteiro Fatorial (parâmetro N do tipo Inteiro) {  
  Se (N == 0)  
    Então Retorne 1;  
  Senão Retorne (N * Fatorial( N-1 ));  
} // fim Fatorial
```

Cálculo do Fatorial de 3

Chamada	N	Resultado
Primeira	3	$\text{Fatorial}(3) = 3 * \text{Fatorial}(2)$  2 resultado do $\text{Fatorial}(2) = 2$
Segunda	2	$\text{Fatorial}(2) = 2 * \text{Fatorial}(1)$ 1 resultado do $\text{Fatorial}(1) = 1$
Terceira	1	$\text{Fatorial}(1) = 1 * \text{Fatorial}(0)$ 1 resultado do $\text{Fatorial}(0) = 1$
Quarta	0	$\text{Fatorial}(0) = 1$

Árvore Binária de Busca - ABB



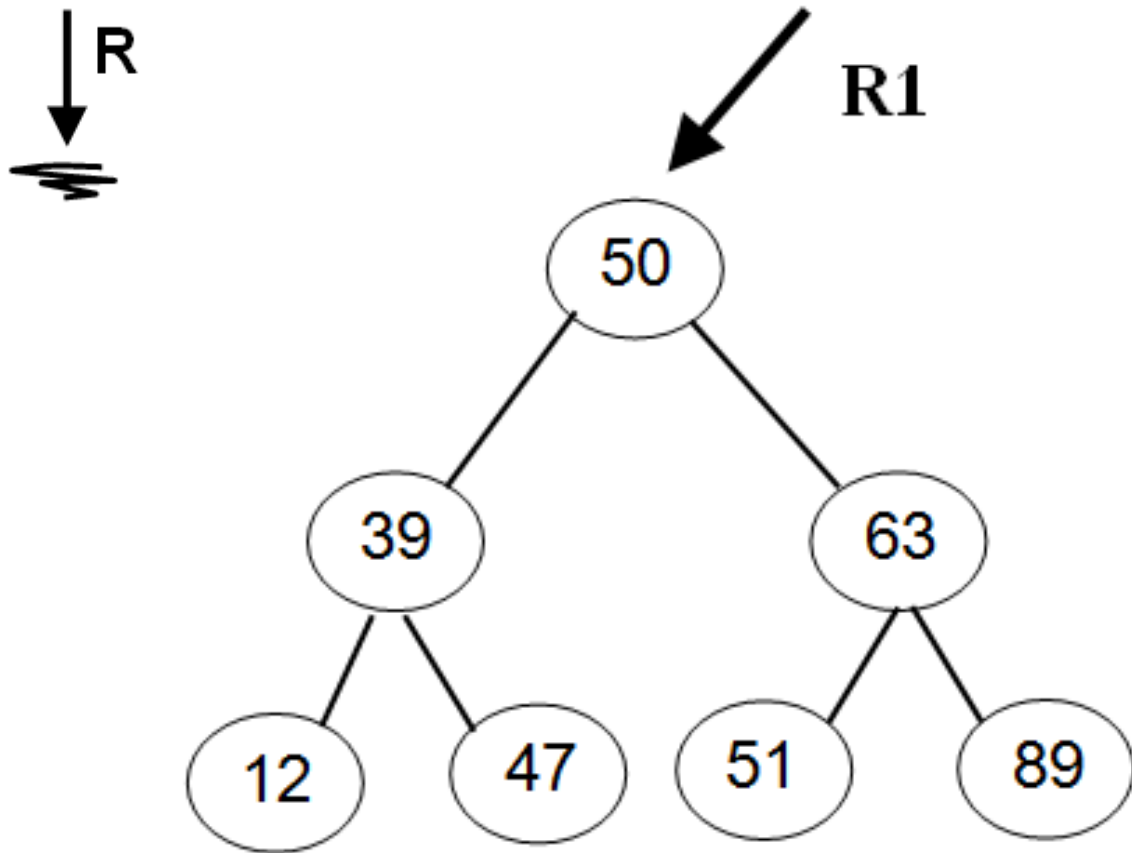
Três Critérios:

1. A Informação de cada Nó da Subárvore Esquerda de R é menor do que a Informação armazenada no Nó apontado por R;
2. A Informação de cada Nó da Subárvore Direita de R é maior do que a Informação armazenada no Nó apontado por R;
3. As Subárvores Esquerda e Direita do Nó apontado por R também são ABBs.
4. Não existem elementos repetidos.

Operações Comuns na ABB

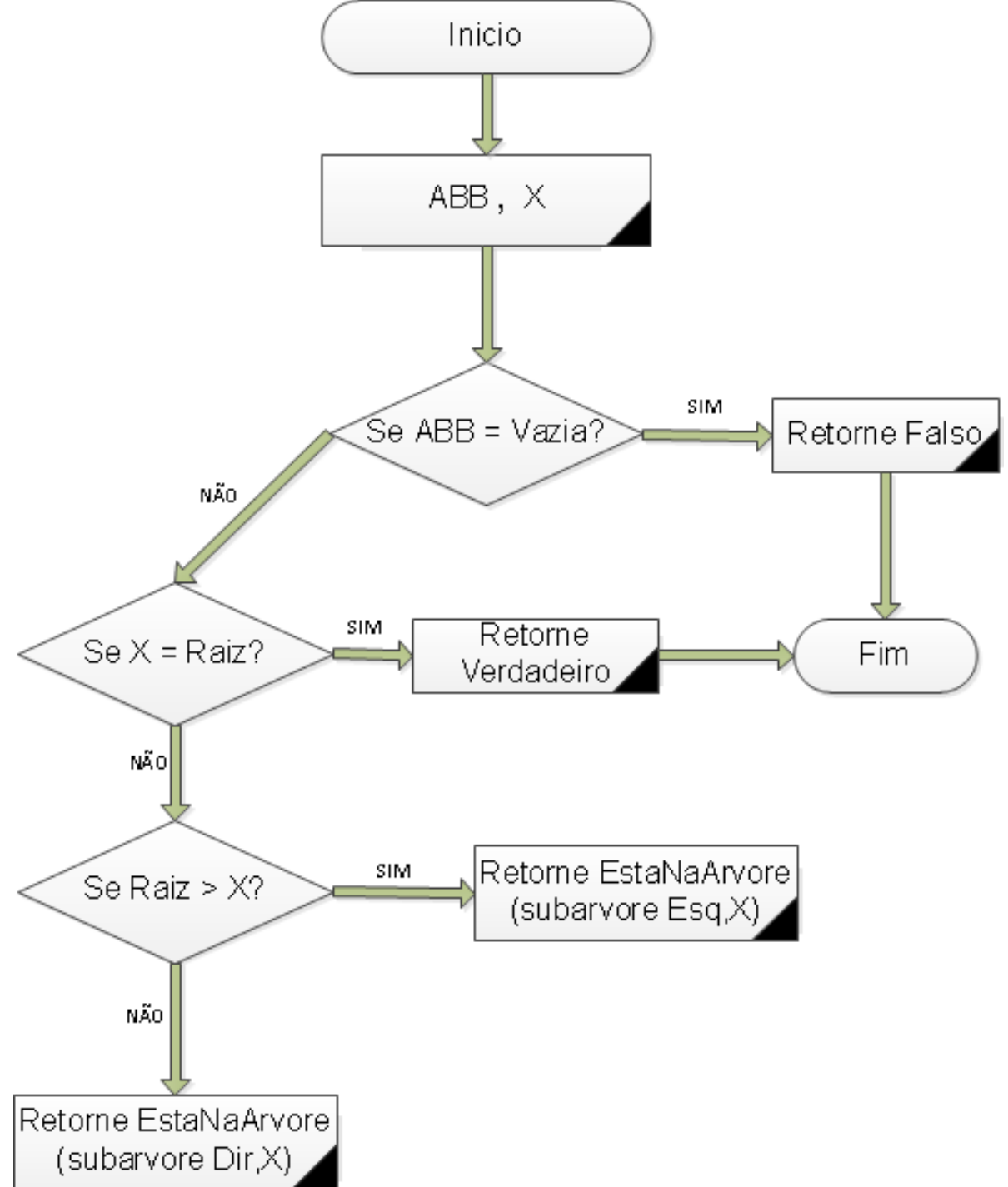
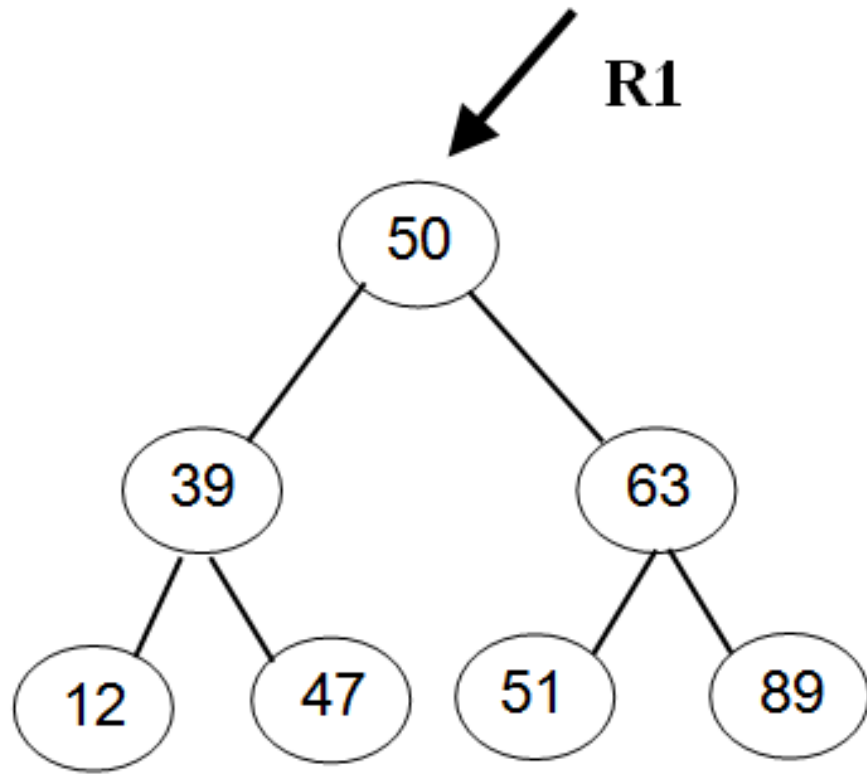
- Busca
- Inserção
- Remoção
- Obter o maior elemento
- Obter o menor elemento
- Percorrer a árvore e exibir seus elementos

Busca na ABB: O valor X (39) está na árvore?
4 Casos.



CASOS
Caso 1: ABB Vazia
Caso 2: Raiz = elemento procurado
Caso 3: Raiz > elemento procurado
Caso 4: Raiz < elemento procurado

EstaNaArvore ?



Função: EstáNaÁrvore?

Boolean **EstáNaÁrvore** (parâmetro por referência **R** do tipo ABB, parâmetro **X** do tipo Inteiro) {

Se (R == Null)

Então Retorne Falso; // **Caso 1: Árvore vazia; X não está na Árvore; acabou**

Senão Se (X == R→Info)

Então Retorne Verdadeiro; // **Caso 2: X está na árvore; acabou**

Senão Se (R→Info > X)

Então Retorne (Está_Na_Árvore (R→Esq, X));

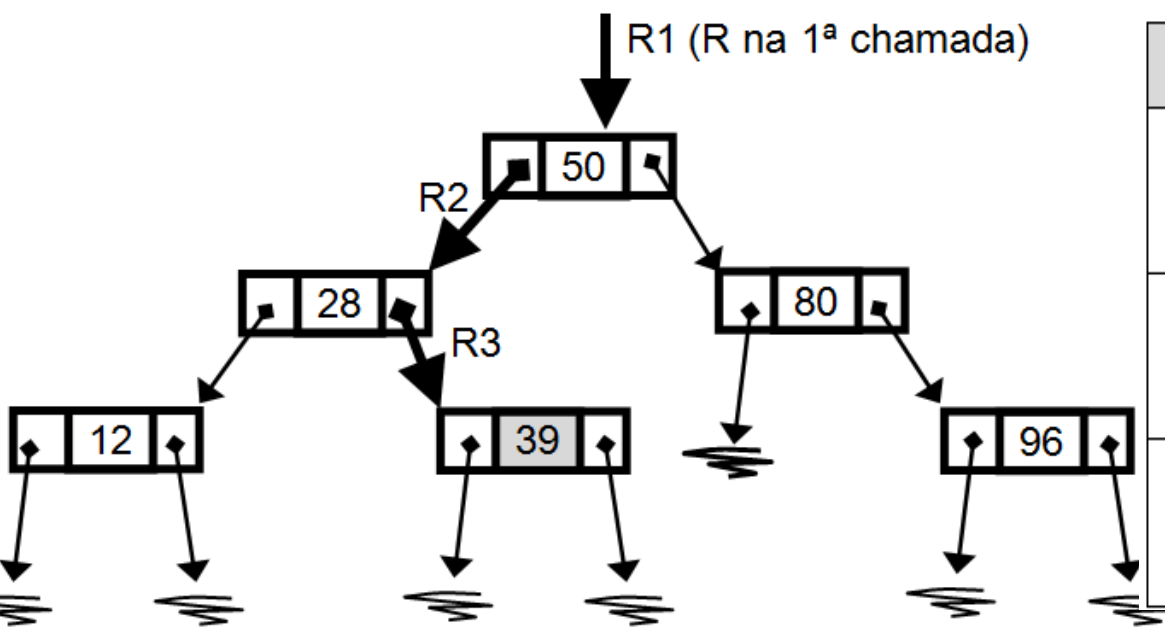
// **Caso 3: se estiver na Árvore, estará na Sub Esquerda**

Senão Retorne (Está_Na_Árvore (R→Dir, X));

// **Caso 4: se estiver na Árvore, estará na Sub Direita**

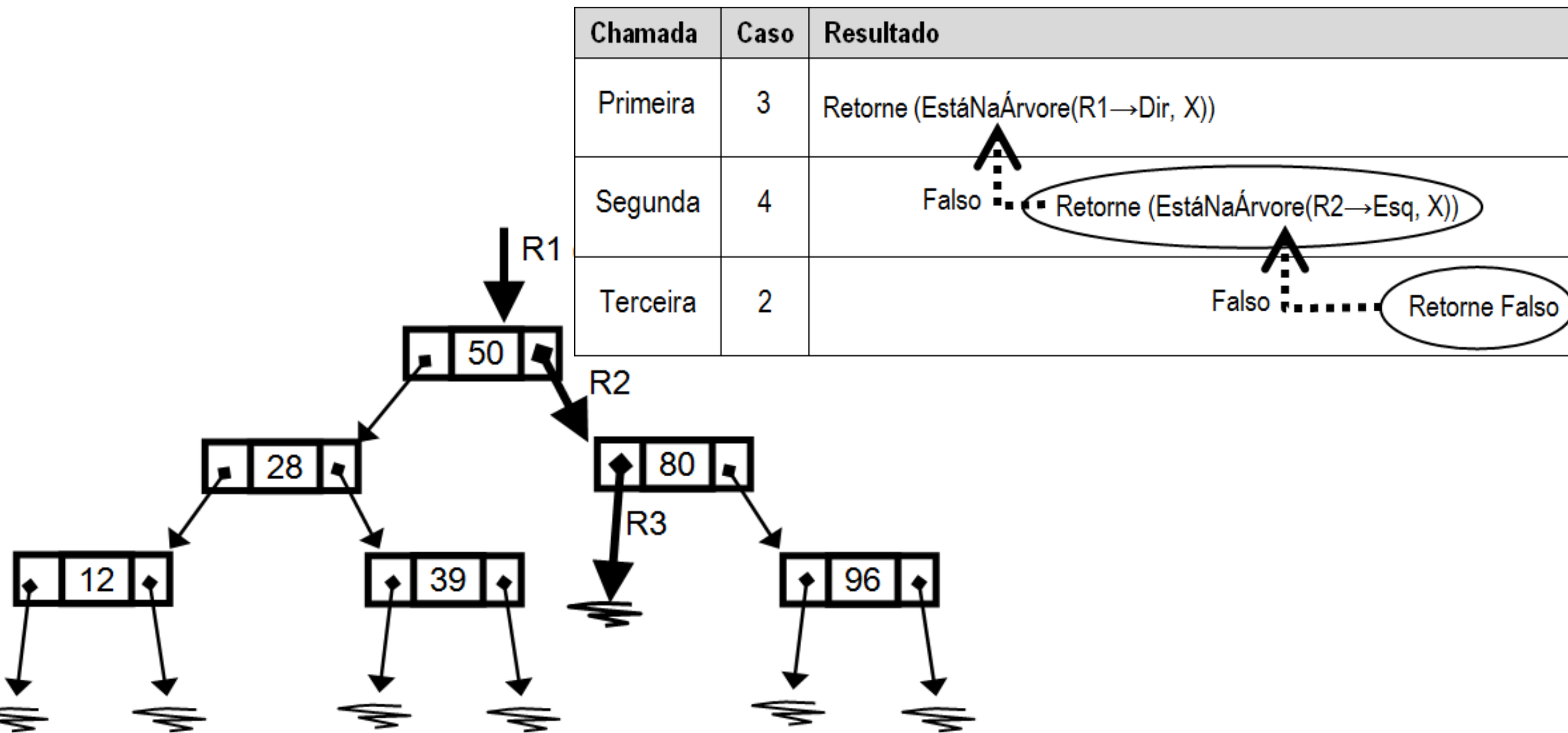
} // fim EstáNaÁrvore

Execução de EstáNaArvore para X=39

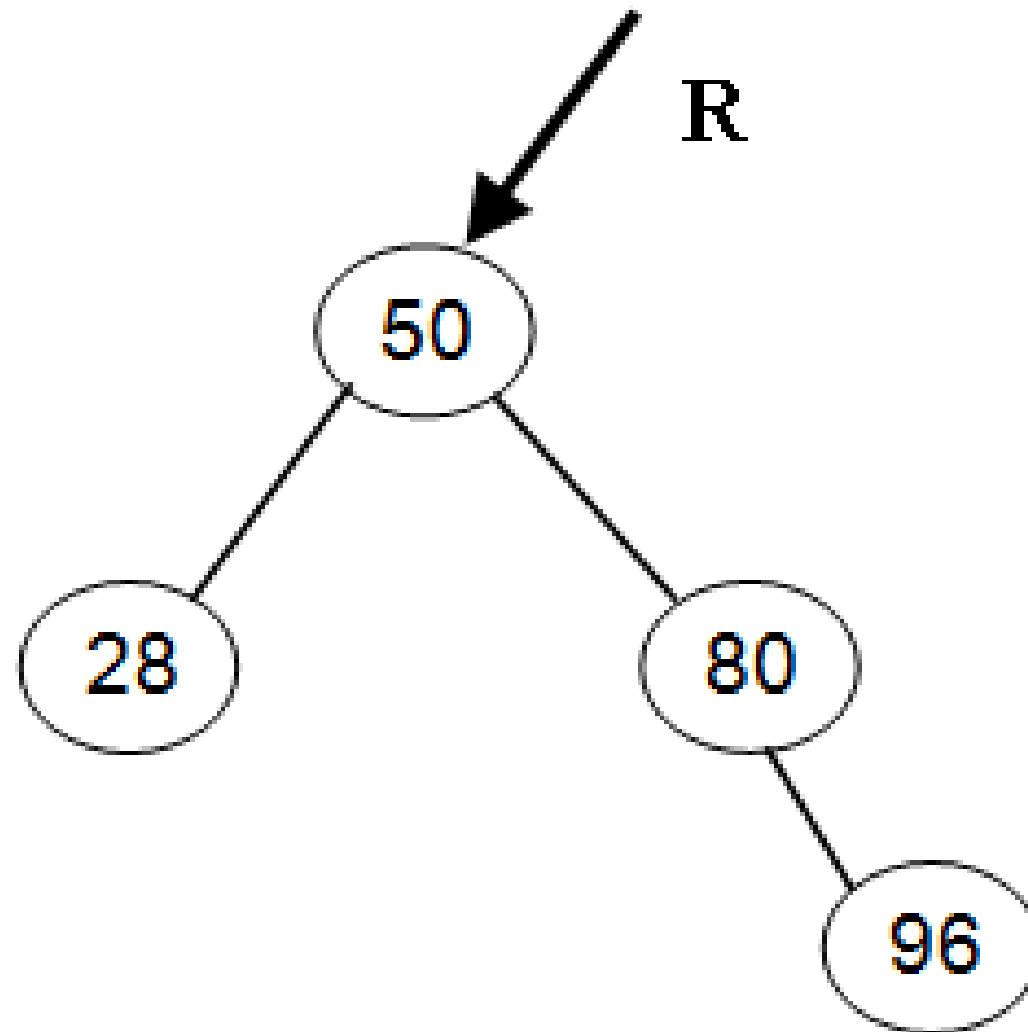


Chamada	Caso	Resultado
Primeira	3	Retorne (EstáNaÁrvore(R1→Esq, X))
Segunda	4	Verdadeiro Retorne (EstáNaÁrvore(R2→Dir, X))
Terceira	2	Verdadeiro Retorne Verdadeiro

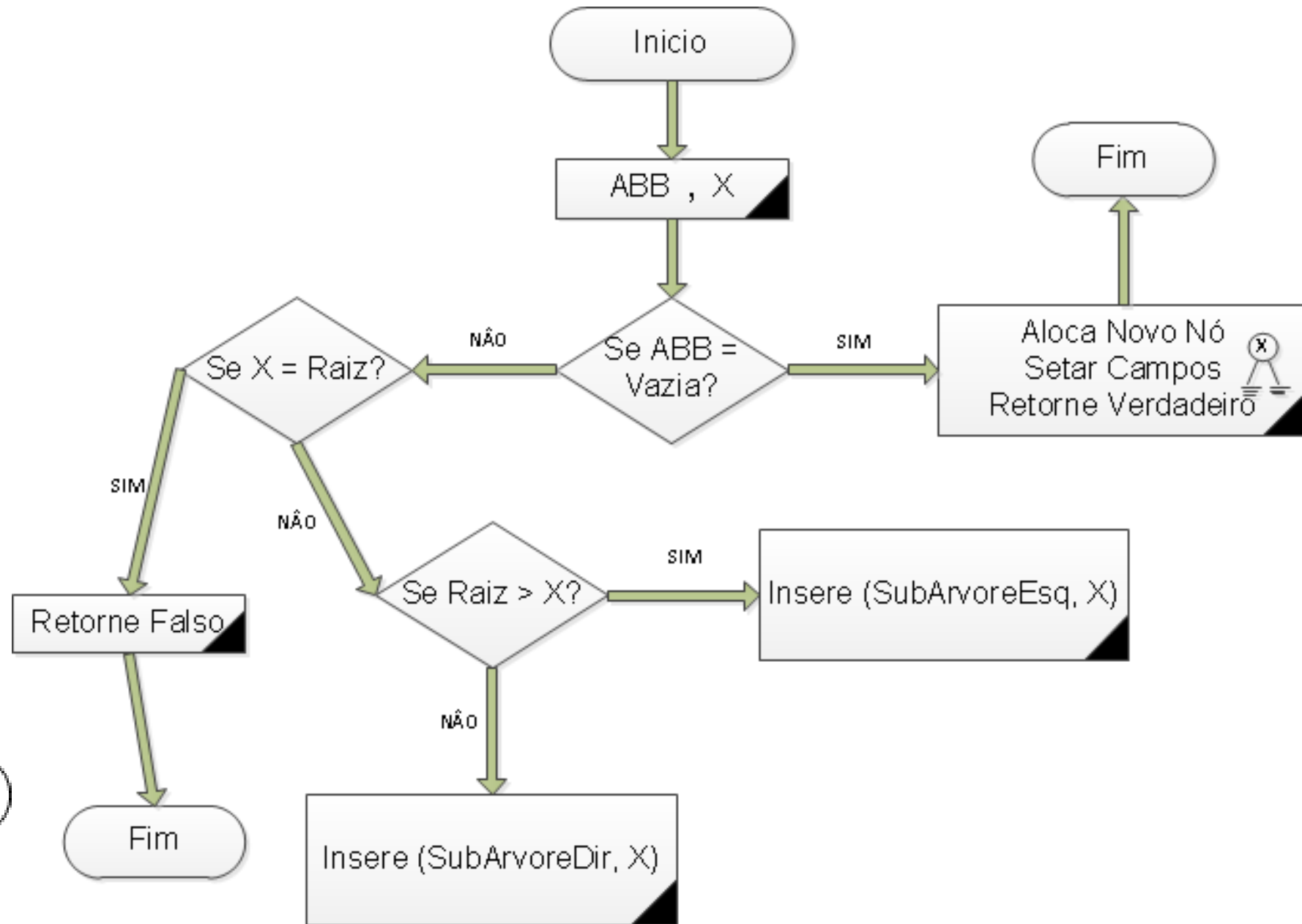
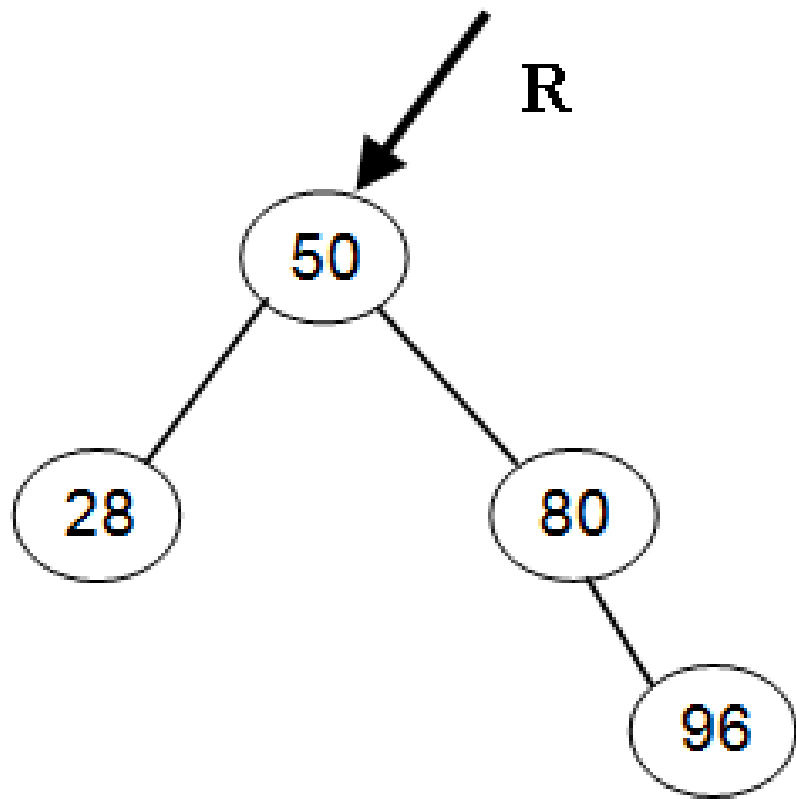
Execução de EstáNaÁrvore para X=70



Inserção na ABB: Onde inserir o 37 ?



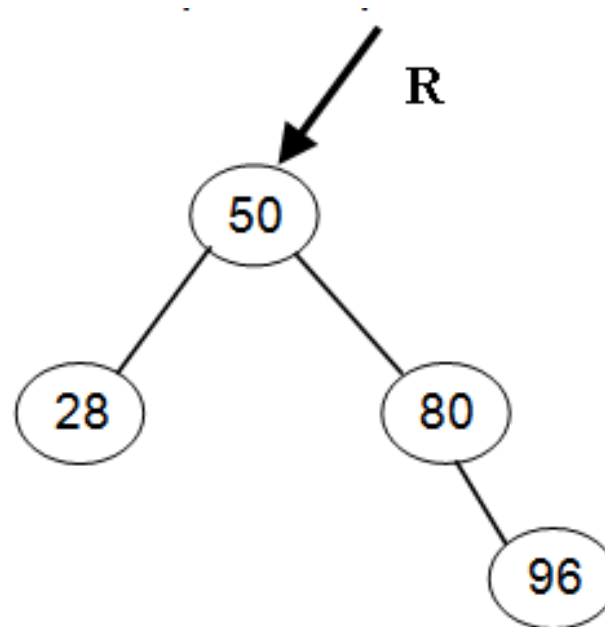
Inserção na ABB:
Onde inserir
o 37 ?



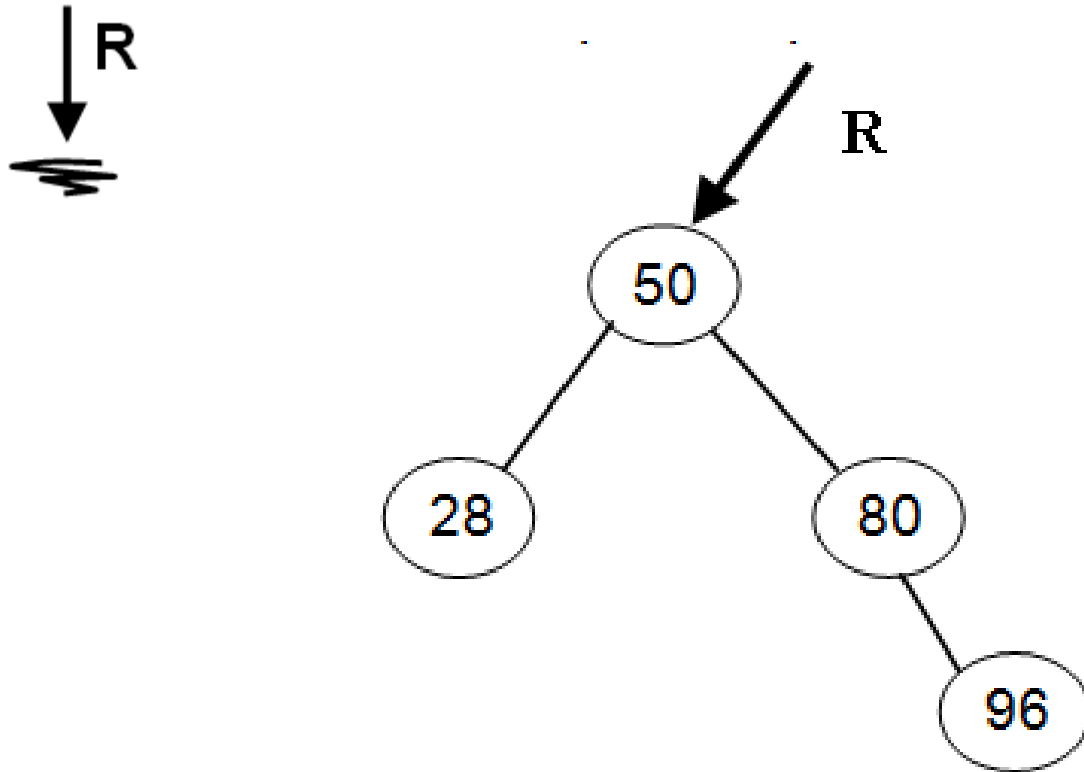
Inserere

Boolean Inserere (parâmetro por referência **R** do tipo ABB, parâmetro **X** do tipo Inteiro);

/ Inserere o valor X na ABB de Raiz R, como um Nó terminal, sem Filhos. Retornar Verdadeiro para o caso de X ter sido inserido, e Falso caso contrário. */*



Inserção na ABB: 4 Casos.



CASOS
Caso 1: ABB Vazia
Caso 2: Raiz = elemento procurado
Caso 3: Raiz > elemento procurado
Caso 4: Raiz < elemento procurado

Boolean **Inserere** (parâmetro **por referência R** do tipo ABB, parâmetro **X** do tipo Inteiro){

Se (R == Null) // **Árvore está vazia**

Então { P = NewNode; // **Caso 1: Achou o lugar; insere e acaba**

 P→Info = X;

 P→Dir = Null;

 P→Esq = Null;

 R = P; P = Null;

 Retorne Verdadeiro;

}

Senão { Se (X == R→Info) // **X é a raiz**

 Então Retorne Falso; // **Caso 2: X já está na árvore; não insere;**

 Senão { Se (R→Info > X)

 Então **Inserere** (R→Esq, X , Ok) // **Caso 3: tenta na Esq**

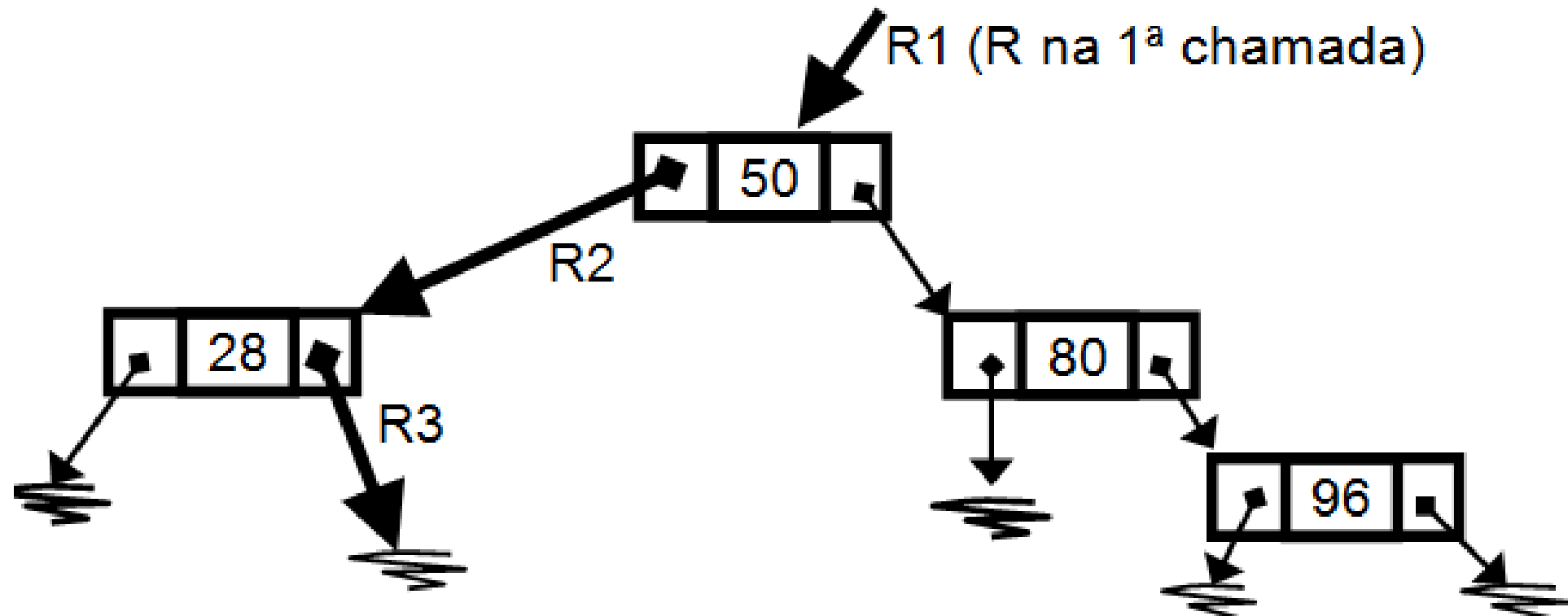
 Senão **Inserere**(R→Dir, X, Ok); // **Caso 4: tenta na Dir**

 } // **fim senão**

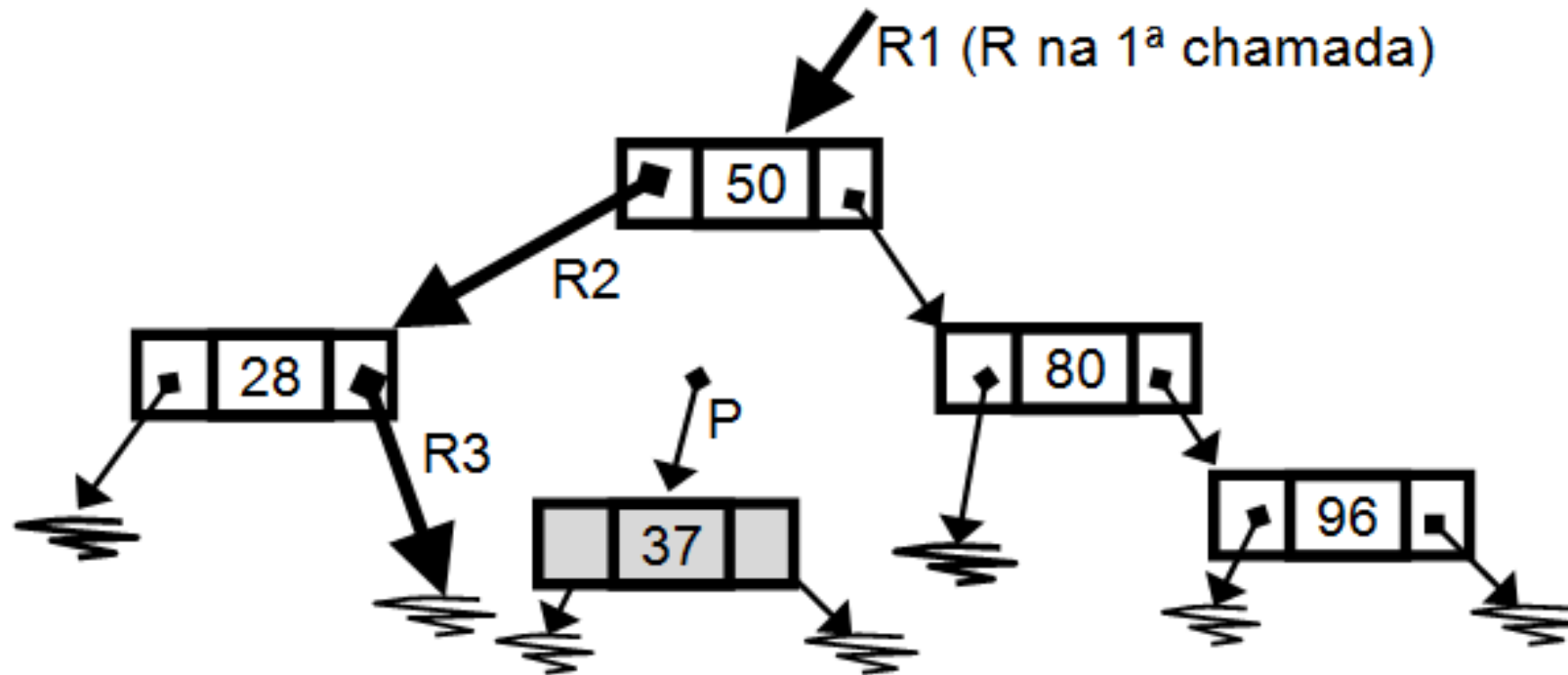
} // **fim senão**

} // **fim Inserere ABB**

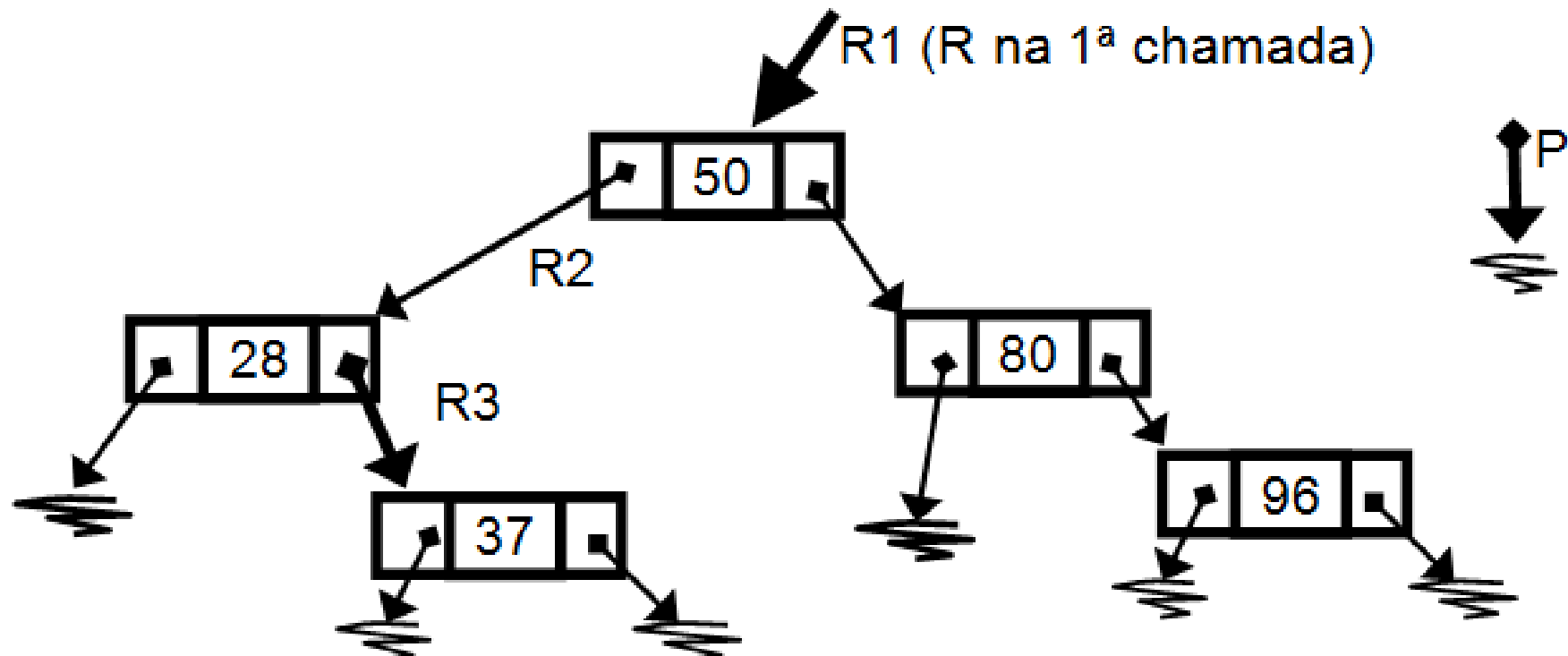
Execução do Insere para X=37



Execução do Insere para X=37



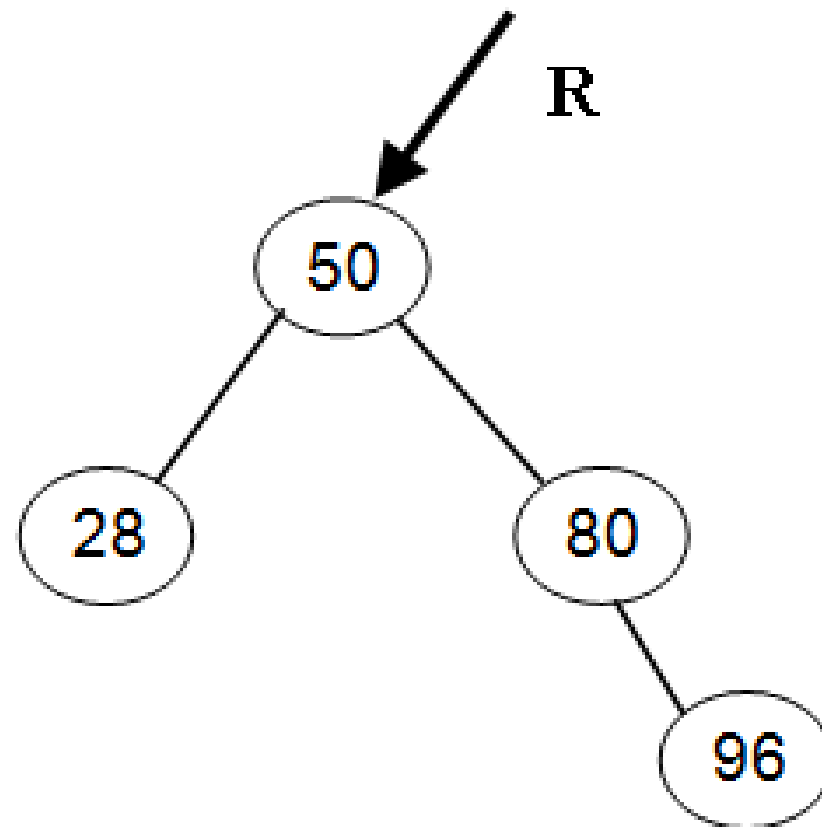
Execução do Insere para X=37



Remoção na ABB: 4 Opções

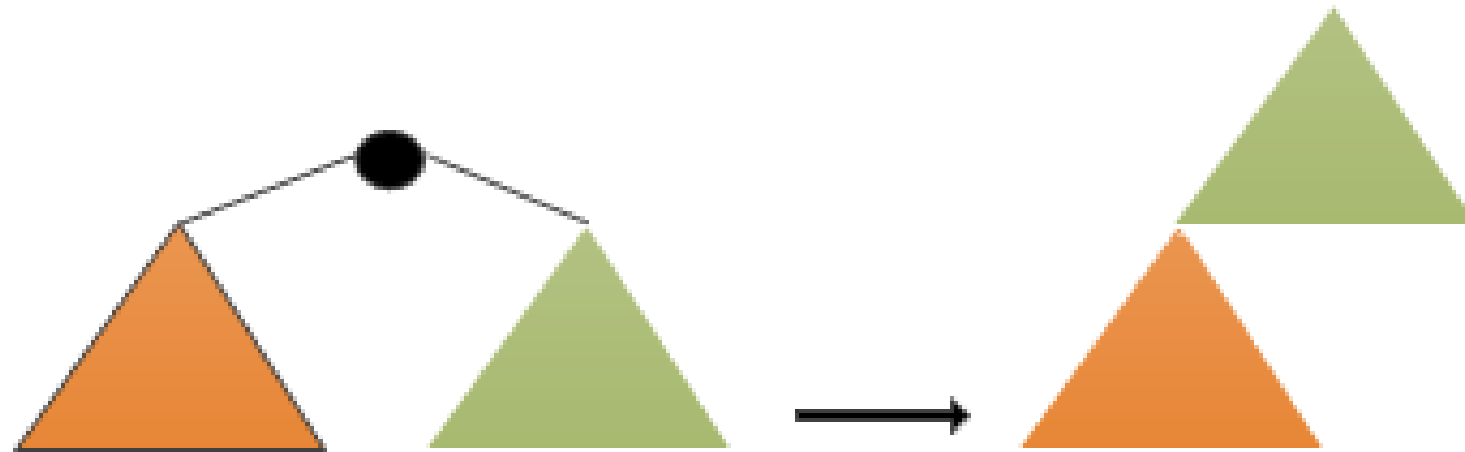
**Como consertar a árvore
ao remover:**

- 28
- 80
- 50



Remoção na ABB: Caso 1

Caso 1: Para eliminar o elemento da raiz, pode-se colocar a SubÁrvoreEsq, a esquerda do menor elemento da SubÁrvoreDir.

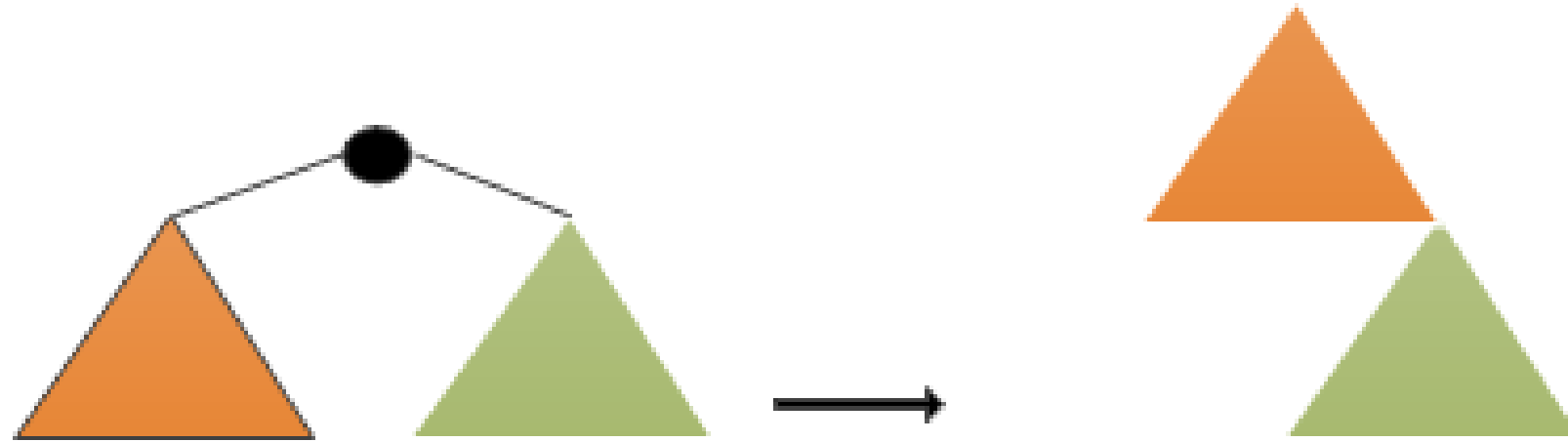


Por exemplo, ao se eliminar o elemento 20 tem-se:

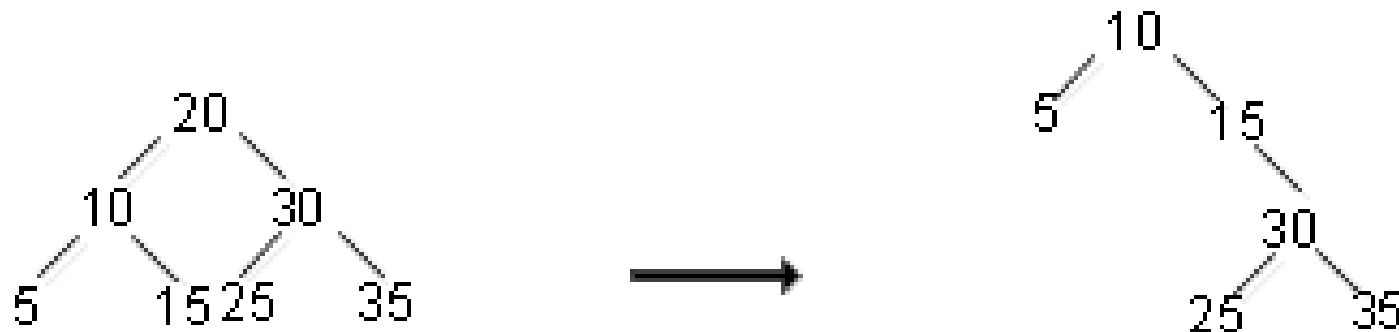


Remoção na ABB: Caso 2

Caso 2: Para eliminar o elemento da raiz, pode-se colocar a SubArvoreDir, a direita do menor elemento da SubArvoreEsq.



Por exemplo, ao se eliminar o elemento 20 tem-se:



Remoção na ABB: Caso 3

Caso 3: Para eliminar o elemento da raiz, pode-se substituir a raiz pelo menor elemento da SubArvoreDir.



Por exemplo, ao se eliminar o elemento 20 tem-se:



Remoção na ABB: Caso 4

Caso 4: Para eliminar o elemento da raiz, pode-se substituir a raiz pelo maior elemento da SubArvoreEsq.



Por exemplo, ao se eliminar o elemento 20 tem-se:

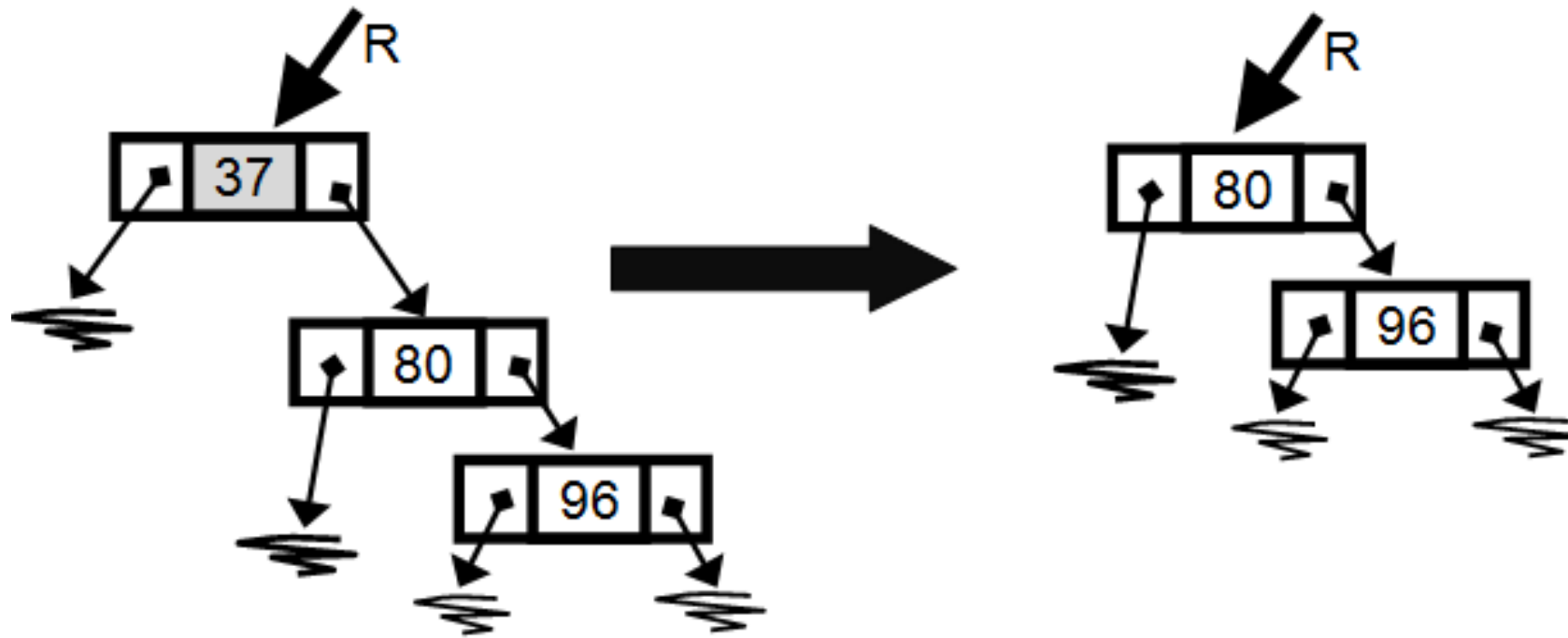


Remoção na ABB: Nó sem Filhos



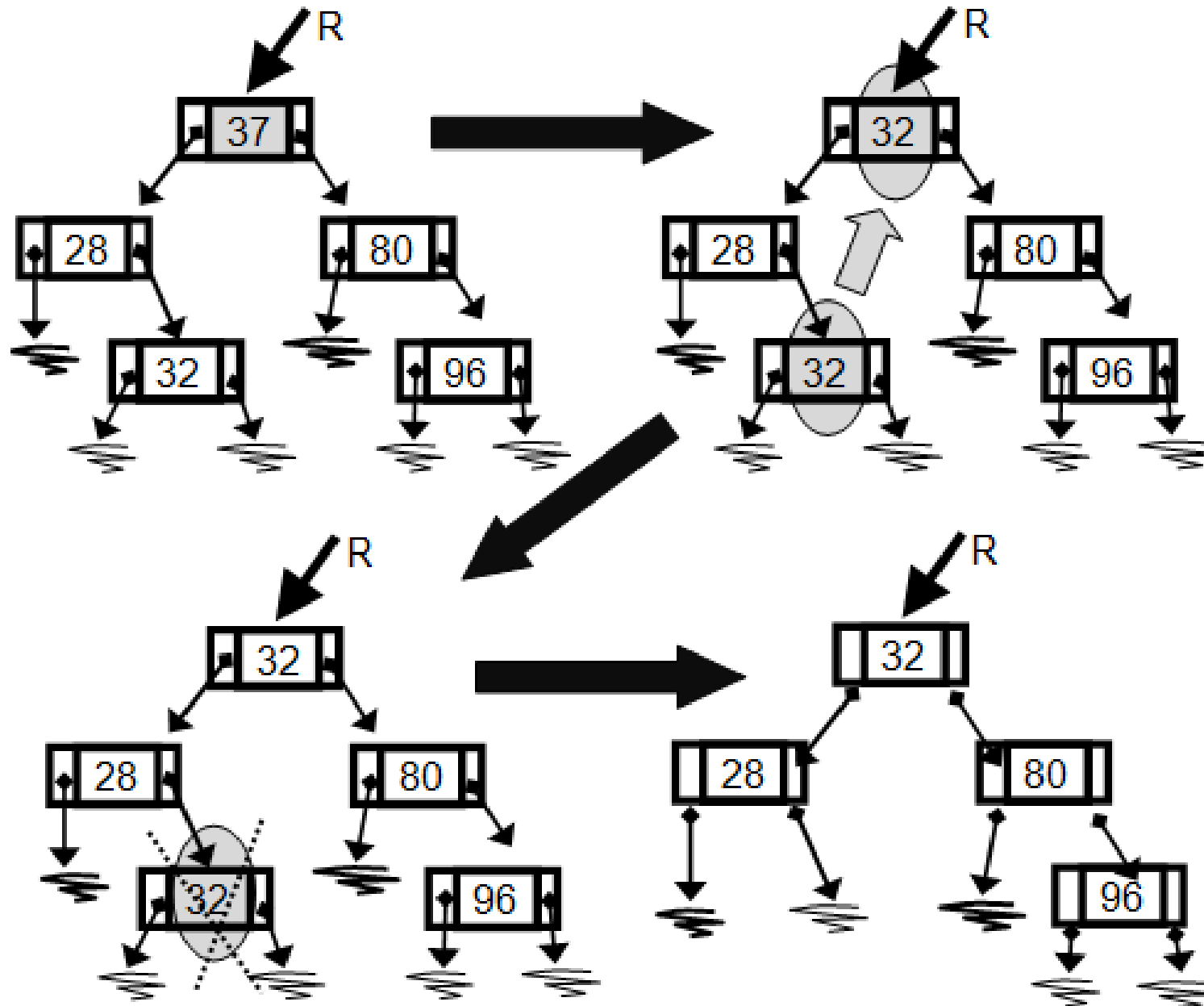
(a) Caso de Remoção 2a: Nó Sem Filhos

Remoção na ABB: Nó com Um Único Filho



(b) Caso de Remoção 2b: Nó com Um Único Filho

Remoção na ABB: Nó com dois Filhos



(c) Caso de Remoção 2a: Nó com Dois Filhos

Por que uma Árvore Binária de Busca é boa?

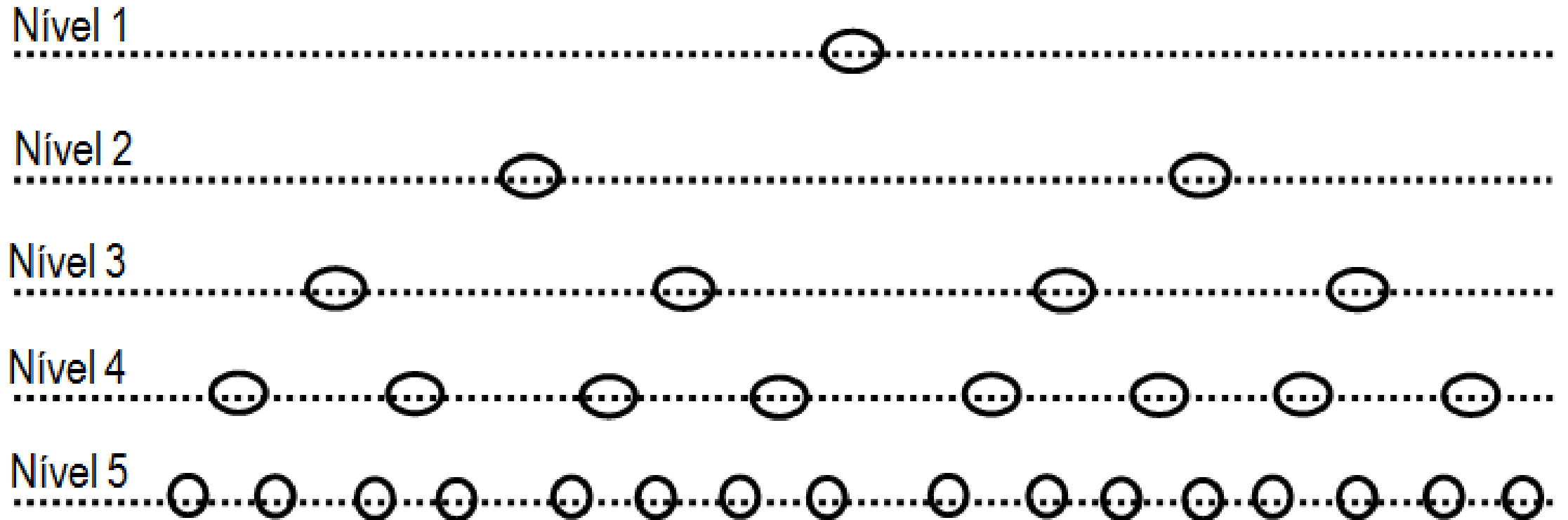


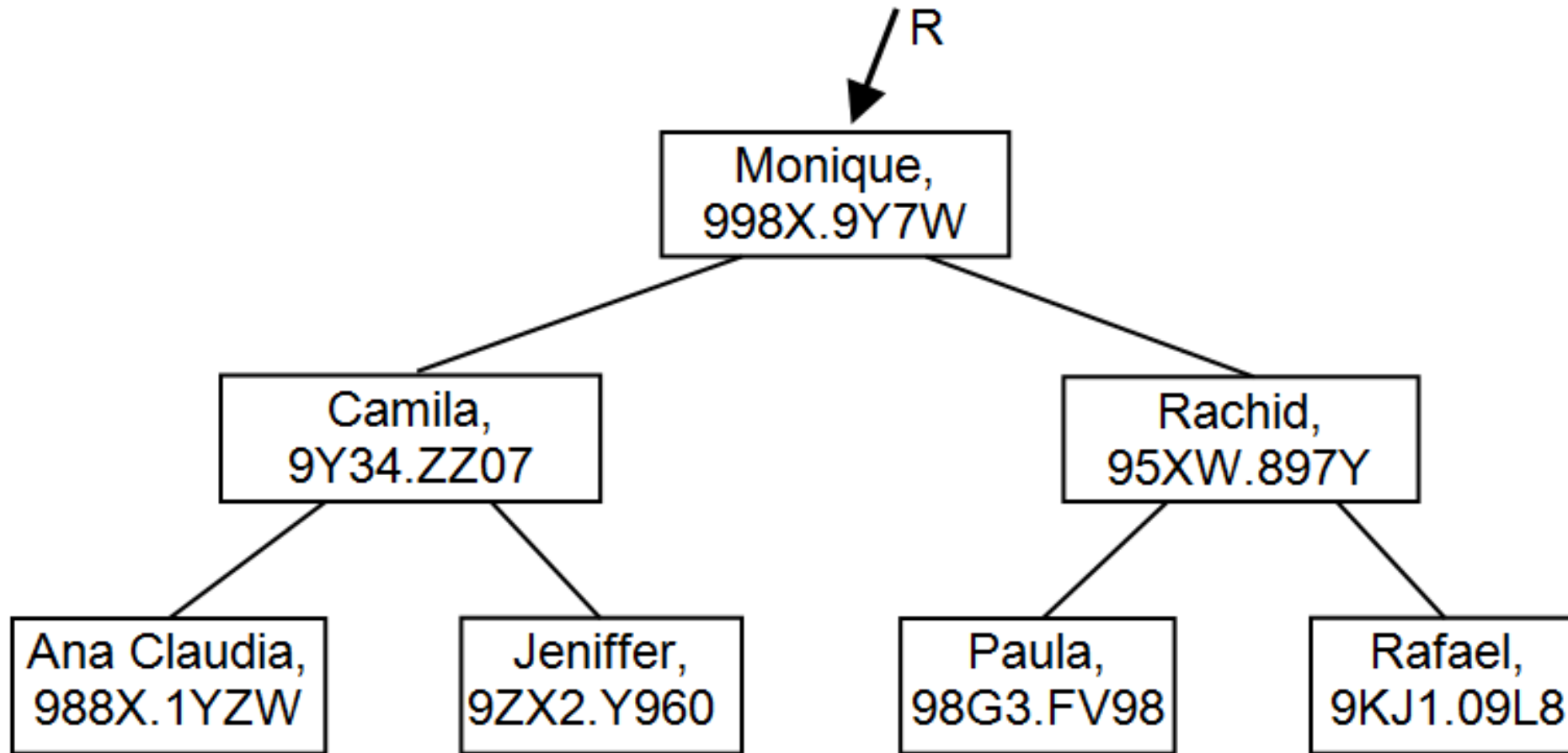
ABB Uniformemente Distribuída

Por que uma Árvore Binária de Busca é boa?

**ABB Uniformemente
Distribuída**

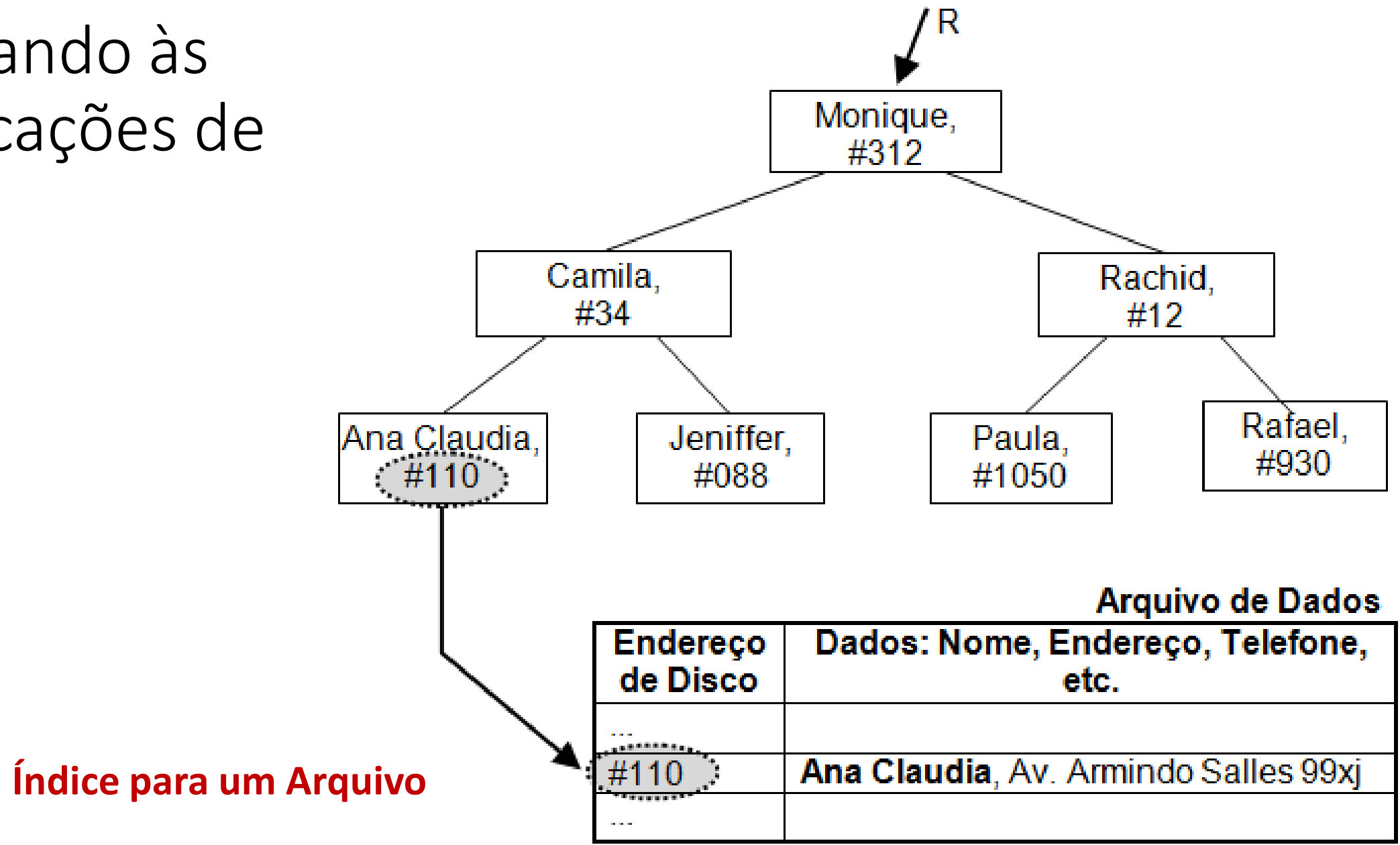
Níveis na Árvore	Quantos Nós Cabem na Árvore
1	1
2	3
3	7
4	15
5	31
N	$2^N - 1$
10	1023
13	8191
16	65535
18	262143
20	1 milhão (aprox)
30	1 bilhão (aprox)
40	1 trilhão (aprox)

Voltando às Aplicações de ABB



Chave de Busca e Outras Informações no Nó

Voltando às
Aplicações de
ABB



Próximos Passos

- Implementação da ABB em Linguagem C
- Árvores Binárias Ordenadas Balanceadas: AVL (Adelson-Velskii e Landis)

Exercícios de Fixação

- Imprimir todos os elementos da ABB
- Computar a soma dos elementos da ABB
- Determinar se uma dada árvore é uma ABB
- Determinar se duas árvores ABB são iguais

Bibliografia

- Projeto de Algoritmos com Implementações em Pascal e C: Nivio Ziviani. 2ª edição, Editora Thomson.
- Introdução a Estrutura de Dados: Waldemar Celes, Renato Fontoura de Gusmão, José Lucas Mourão Rangel Netto. Editora Elsevier.
- Diseño y Manejo de Estructuras de Datos em C: Jorge A. Villalobos. Editora McGraw-Hill.