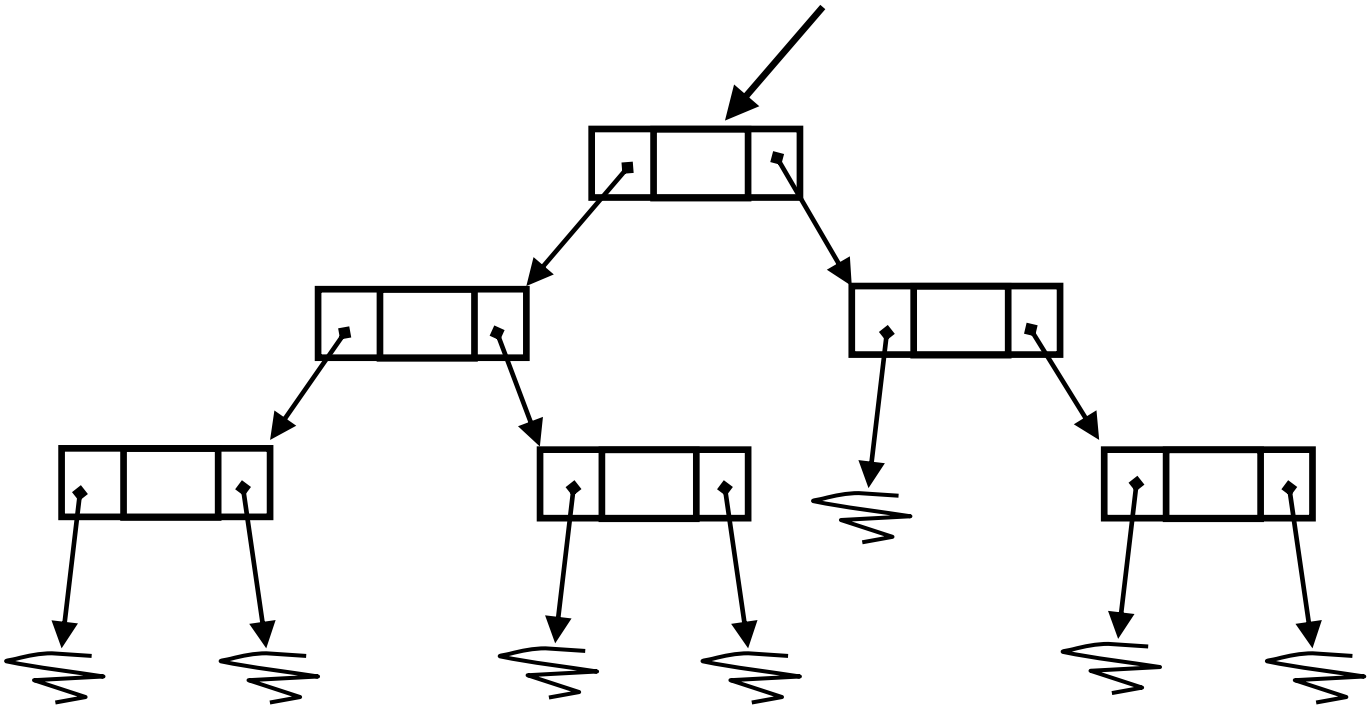
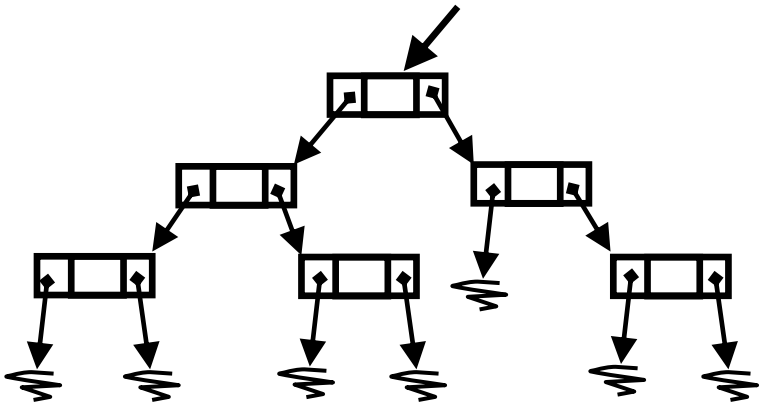
Estruturas de Dados com Jogos



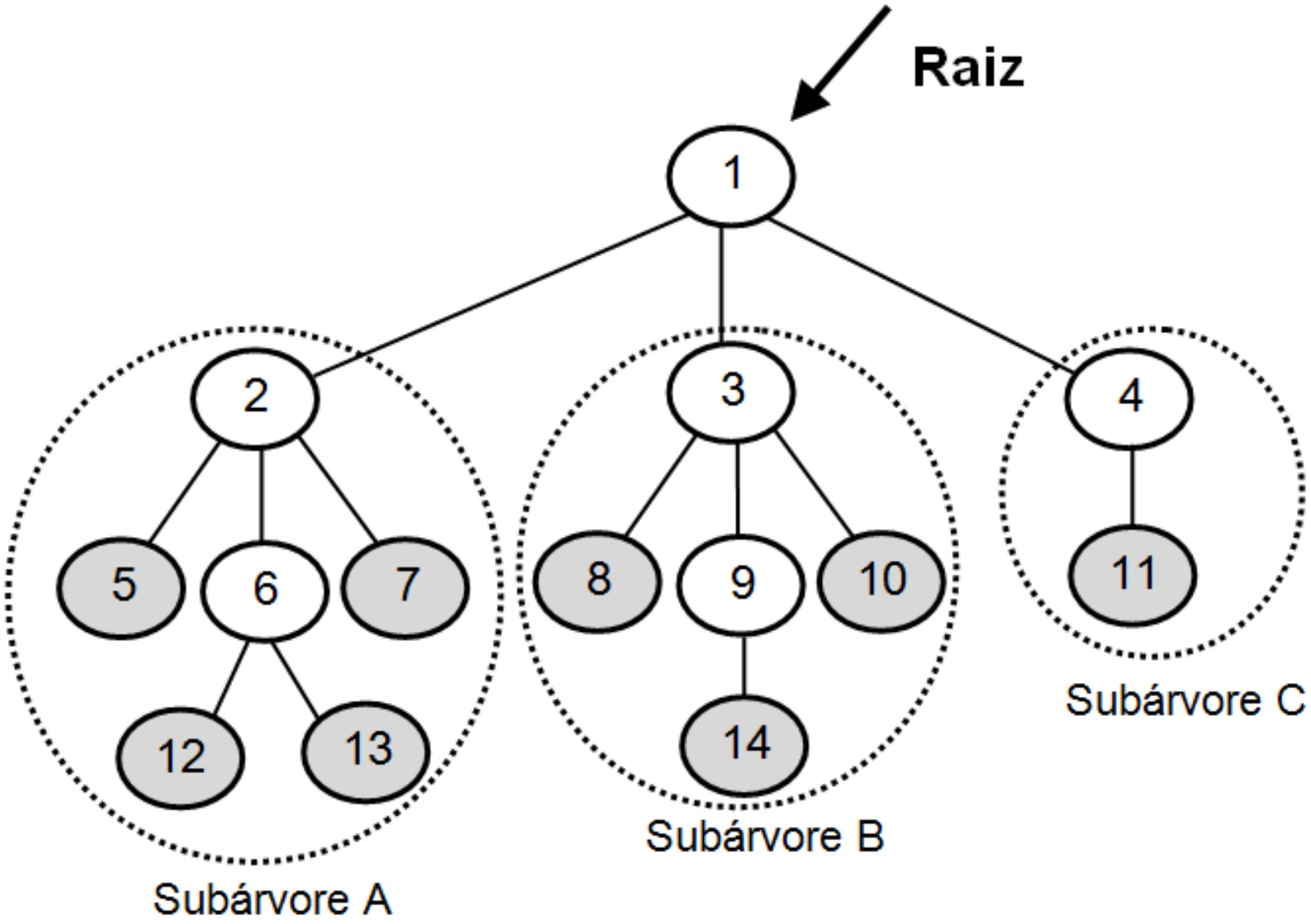
Capítulo 8
Árvores



Seus Objetivos neste Capítulo

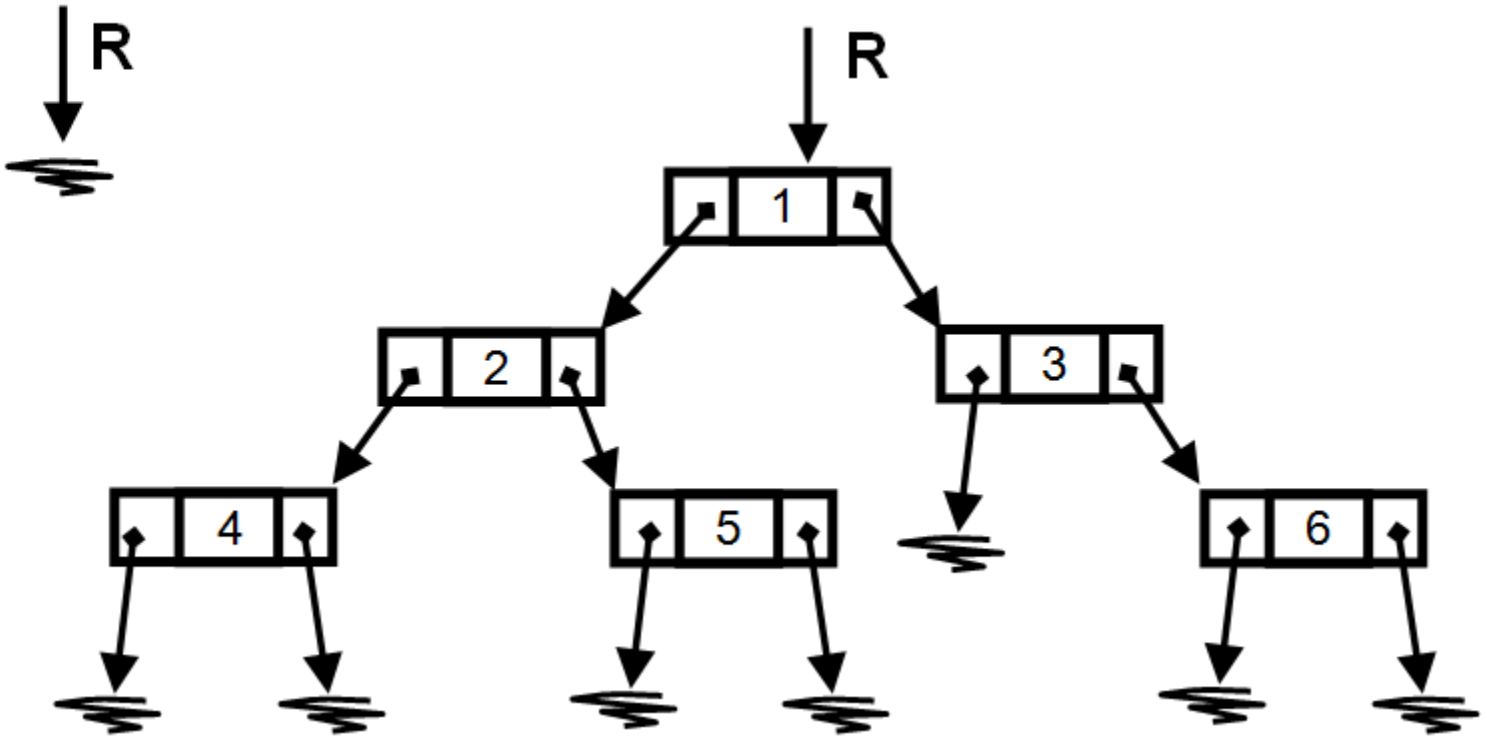
- Entender o conceito, a nomenclatura e a representação usual da estrutura de armazenamento denominada **Árvore**, e de um tipo especial de Árvore denominada **Árvore Binária de Busca - ABB**;
- Desenvolver habilidade para elaborar algoritmos sobre Árvores Binárias de Busca, e sobre Árvores em geral;
- Conhecer aplicações e a motivação para o uso de Árvores; entender as situações em que seu uso é pertinente;
- Iniciar o desenvolvimento do seu jogo referente ao Desafio 4.

Árvores: Conceito e Representação

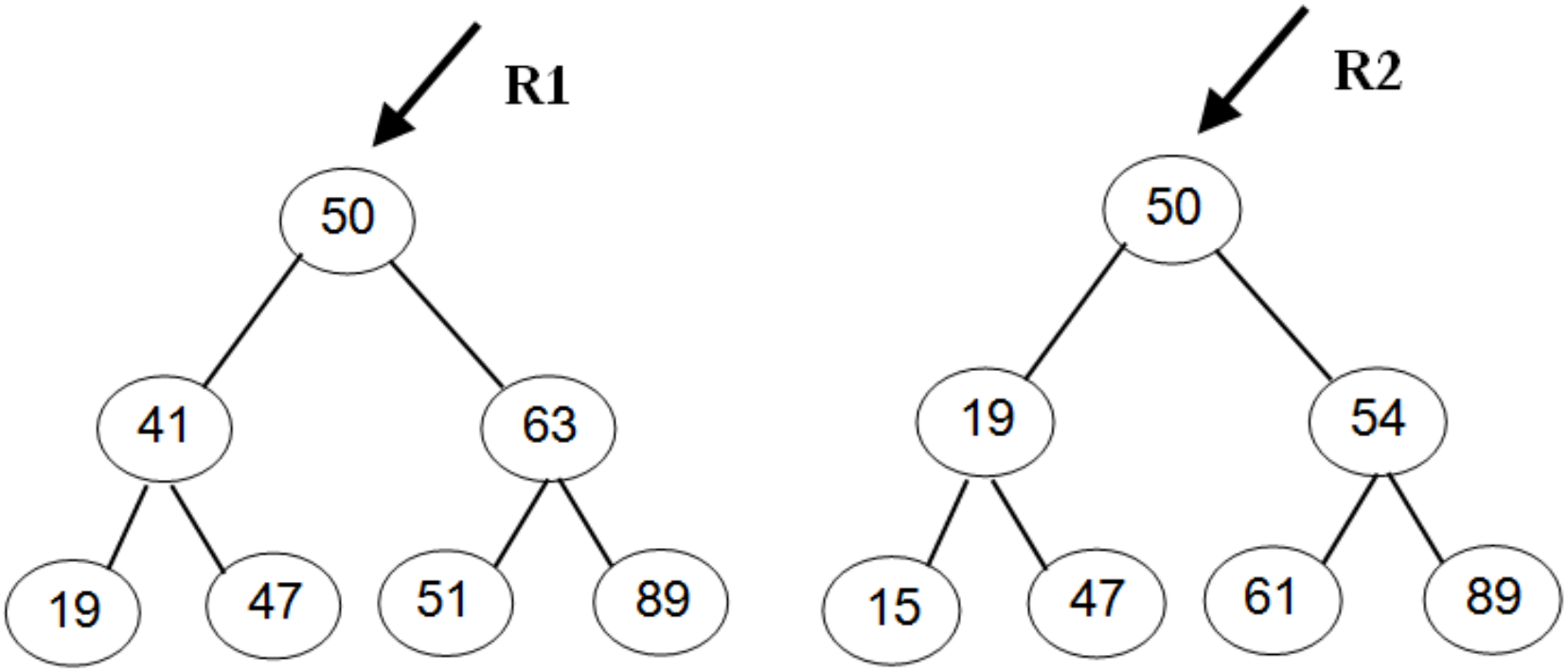


Árvore Binária:

cada Nó possui, no máximo, dois Filhos.



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.

Revisão - Algoritmos Recursivos

Fatorial - Definição:

(i) Fatorial de 0 é 1

(ii) Fatorial de N é N * Fatorial (N - 1)

Dica:

É possível resolver de imediato

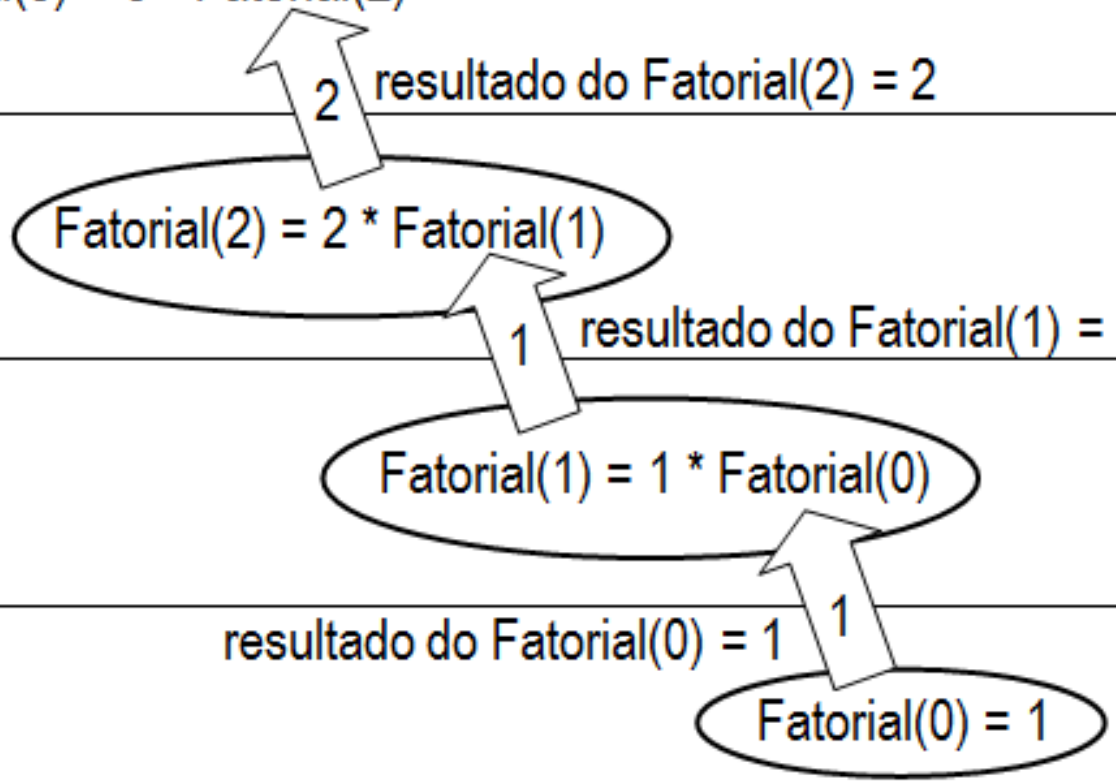
Deixamos para resolver depois

Fatorial - Implementação Recursiva:

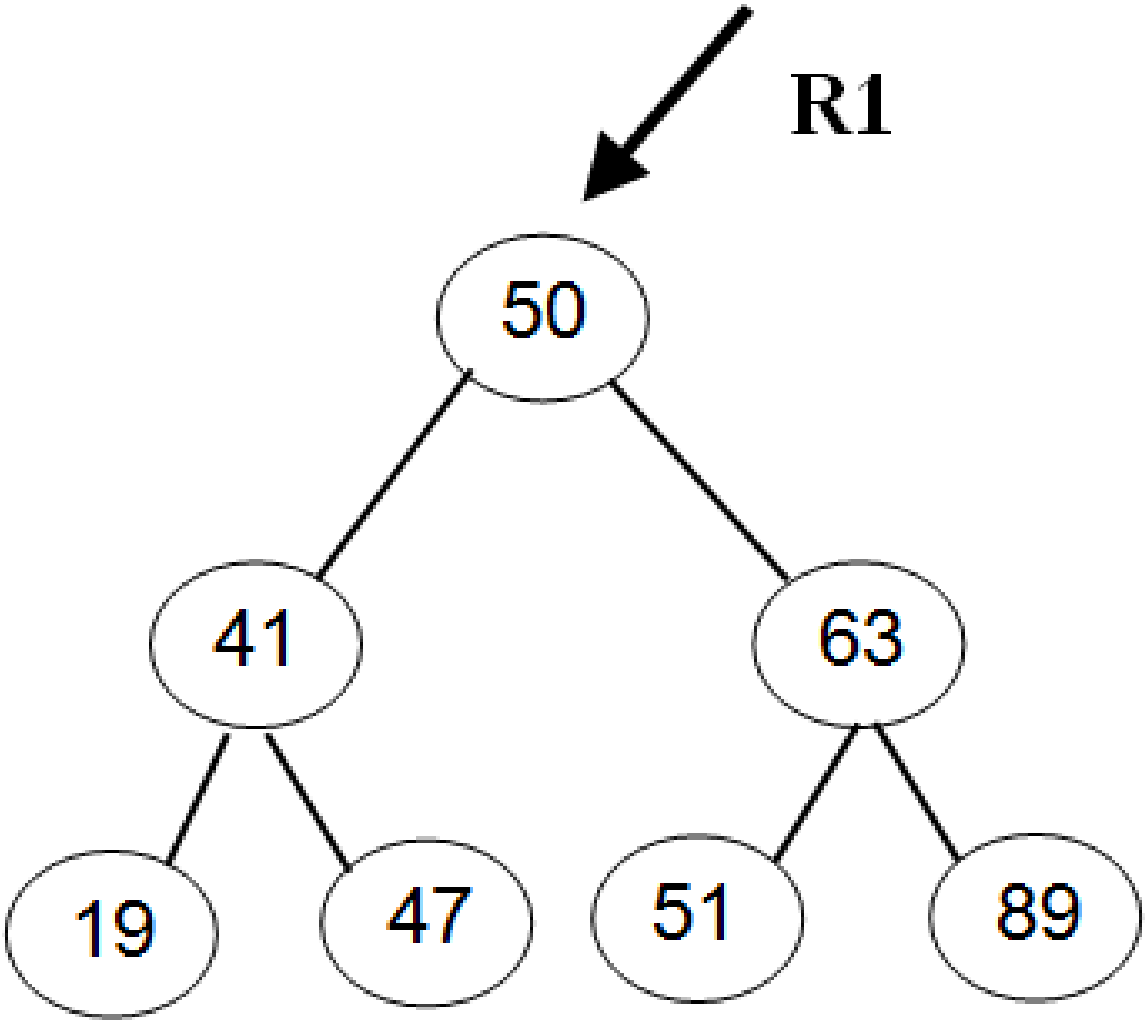
```
Inteiro Fatorial (parâmetro N do tipo Inteiro) {
Se (N == 0)
Então Retorne 1;
Senão Retorne (N * Fatorial( N-1 ));
} // fim Fatorial
```

Cálculo do Fatorial de 3

Chamada	N	Resultado
Primeira	3	Fatorial(3) = 3 * Fatorial(2)
Segunda	2	Fatorial(2) = 2 * Fatorial(1)
Terceira	1	Fatorial(1) = 1 * Fatorial(0)
Quarta	0	Fatorial(0) = 1

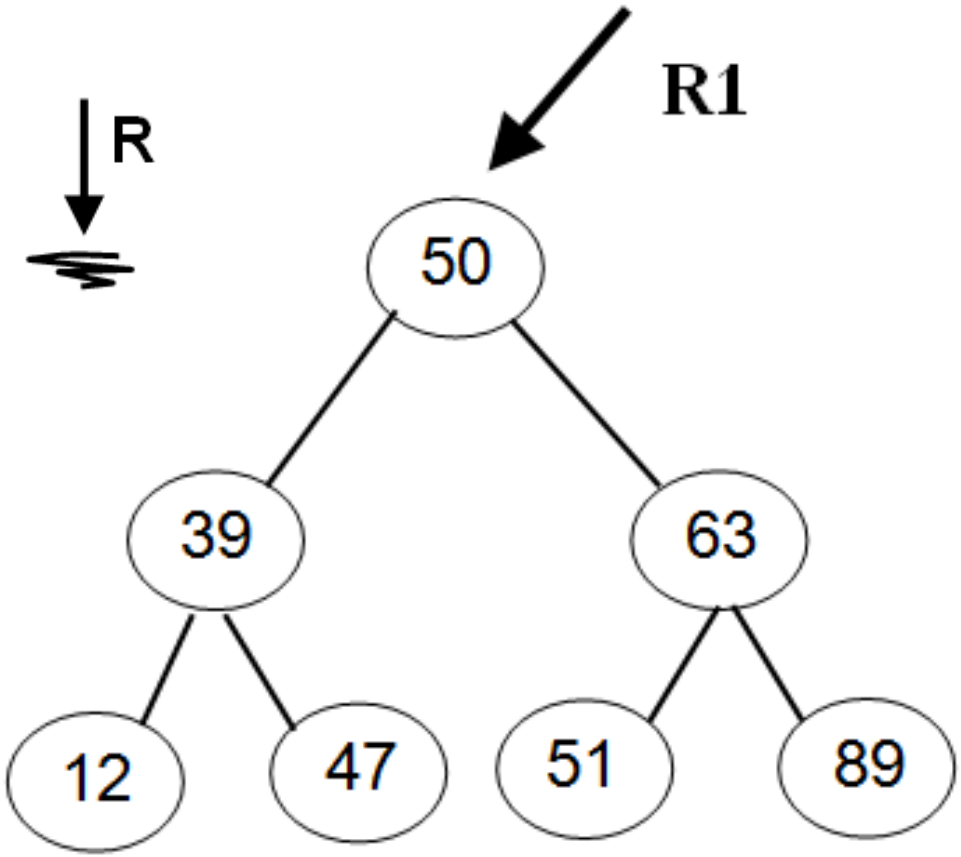


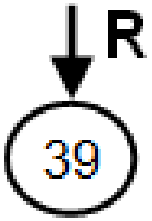
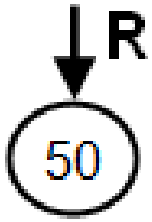
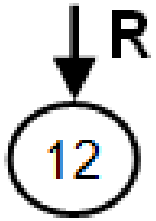
O Valor X Está na Árvore?



O Valor X Está na Árvore?

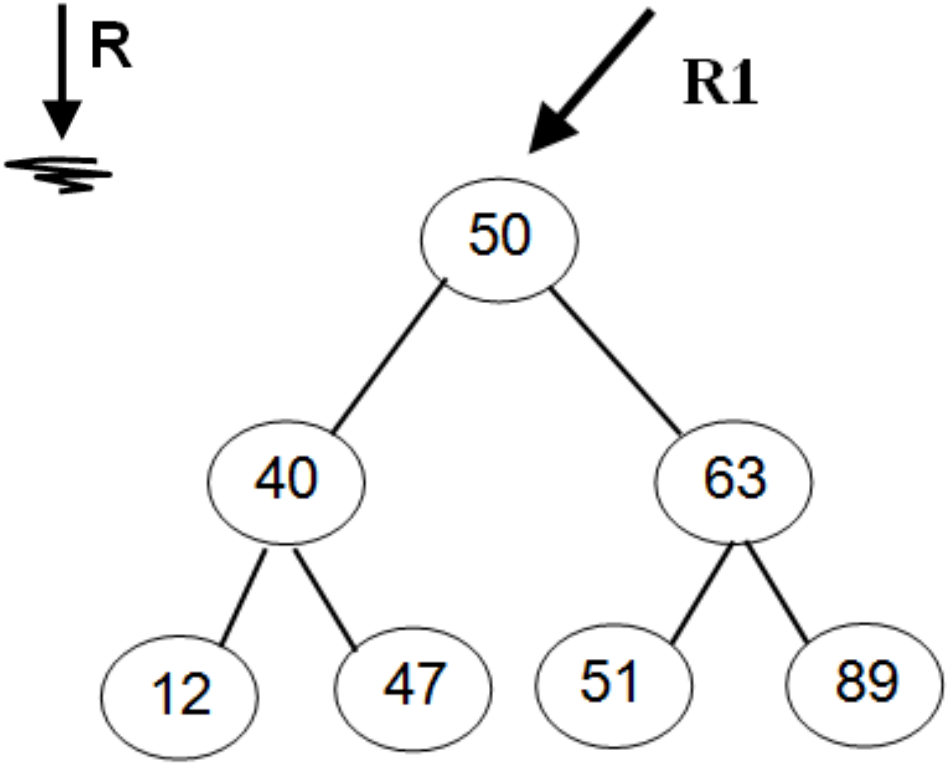
4 casos

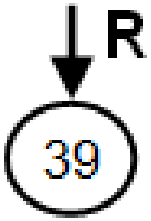
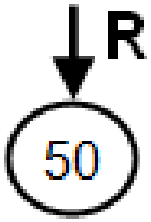
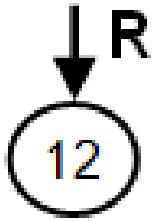


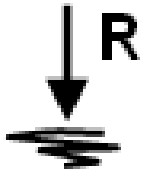
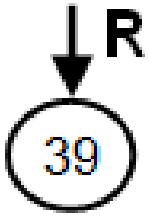
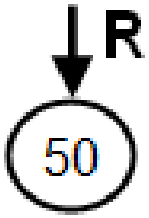
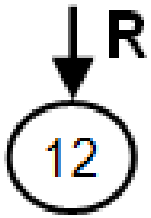
Caso	X	R
Caso 1: Árvore Vazia	39	
Caso 2: R⟶Info = X	39	
Caso 3: X < R⟶Info	39	
Caso 4: X > R⟶Info	39	

O Valor X Está na Árvore?

4 casos



Caso	X	R
Caso 1: Árvore Vazia	39	
Caso 2: R⟶Info = X	39	
Caso 3: X < R⟶Info	39	
Caso 4: X > R⟶Info	39	

Caso	X	R	Conclusão
Caso 1: Árvore Vazia	39		X não está na árvore. Encerramos o algoritmo.
Caso 2: R→Info = X	39		X está na árvore. Encerramos o algoritmo.
			É possível resolver de imediato e encerrar o algoritmo
			Deixamos para resolver depois, com recursividade
Caso 3: X < R→Info	39		Se X estiver na Árvore, estará na Subárvore Esquerda de R. O algoritmo não acaba ainda. Fazemos uma chamada recursiva.
Caso 4: X > R→Info	39		Se X estiver na Árvore, estará na Subárvore Direita de R. O algoritmo não acaba ainda. Fazemos uma chamada recursiva.

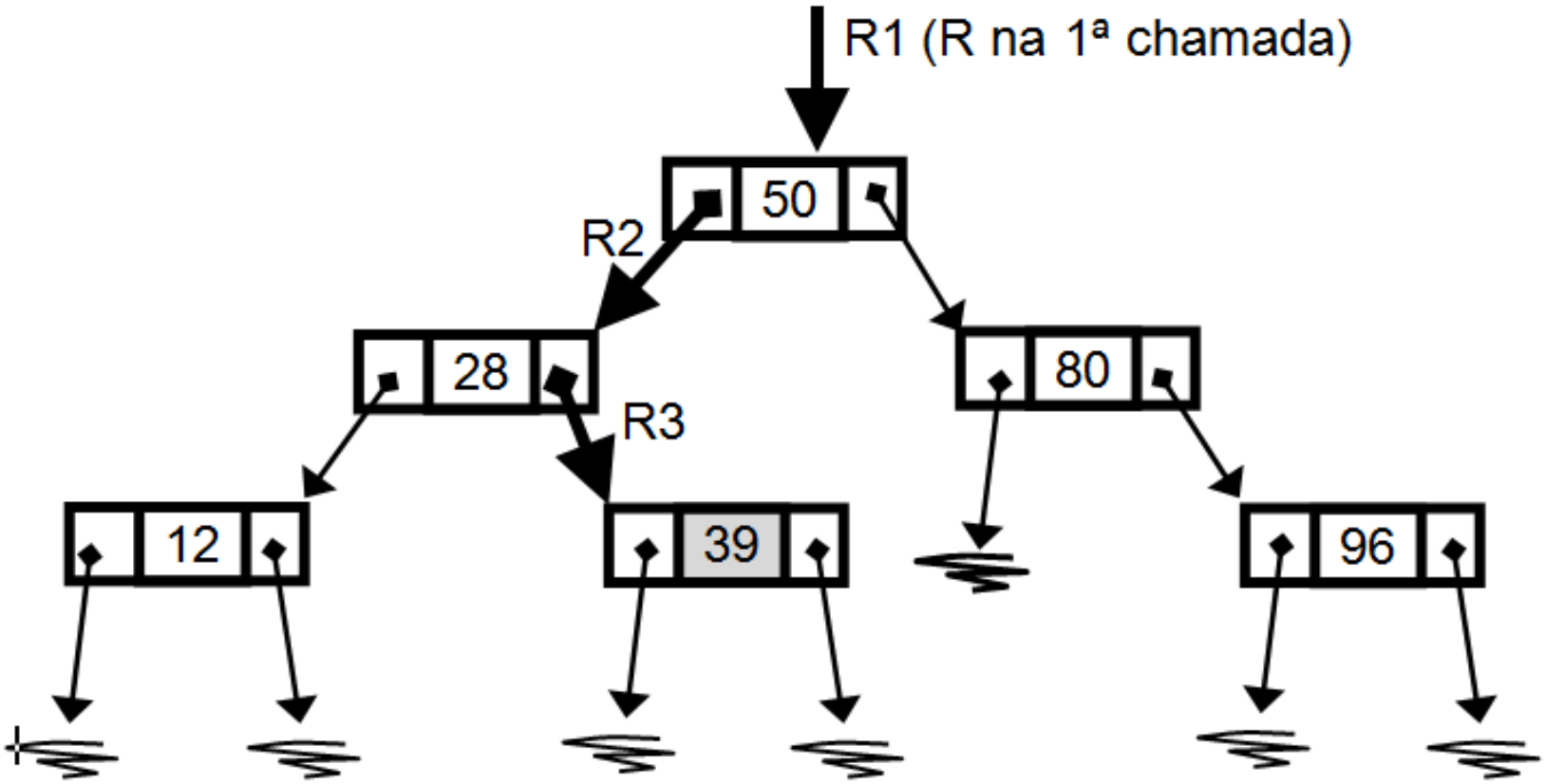
EstáNaÁrvore?

```
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
```

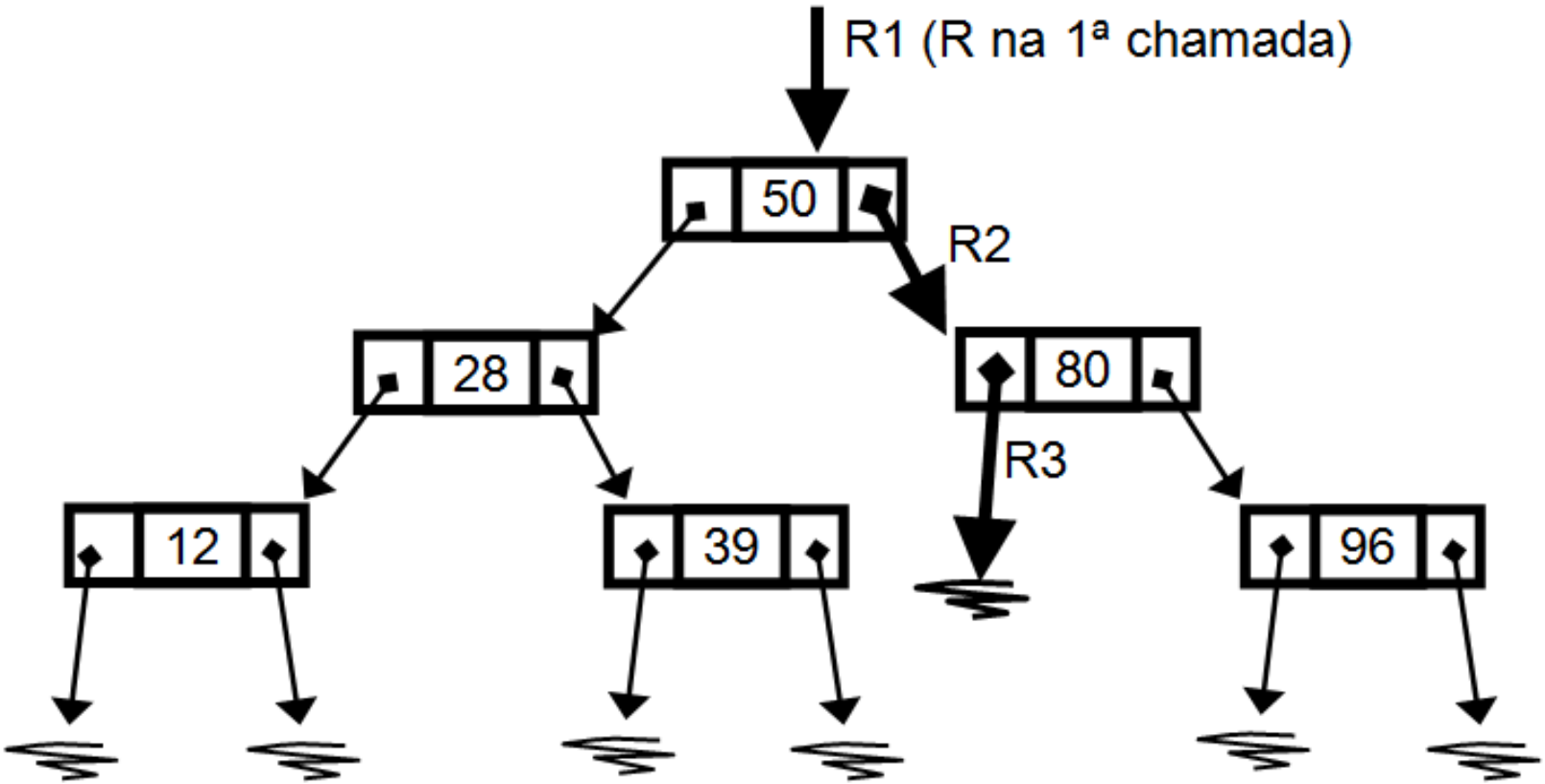
Execução de EstáNaÁrvore para X = 39



Execução de EstáNaÁrvore 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



Execução de EstáNaÁrvore para X = 70

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

Ao Elaborar Algoritmos Recursivos...

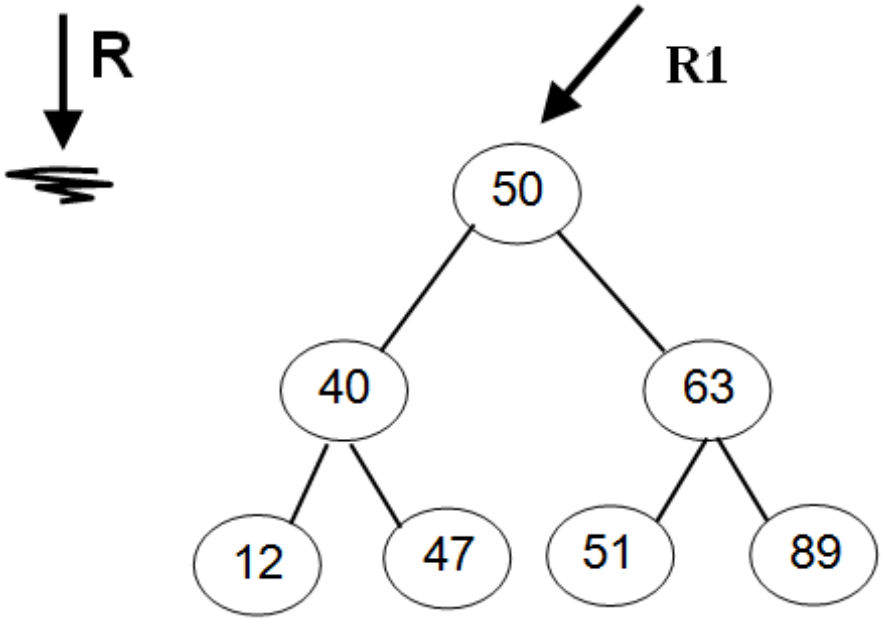
- Liste todos os casos, identificando-os como Caso 1, Caso 2, e assim por diante;
- Identifique os casos em que é possível dar uma resposta de imediato, e proponha a resposta;
- Identifique os casos em que não é possível resolver de imediato, e procure resolver com uma ou mais chamadas recursivas.

Exercícios

Imprimir uma Árvore

ImprimeTodos (parâmetro por referência R do tipo ABB);

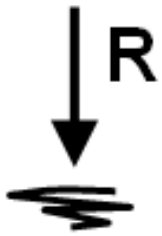
/* Imprime todos os elementos da Árvore de Raiz R */

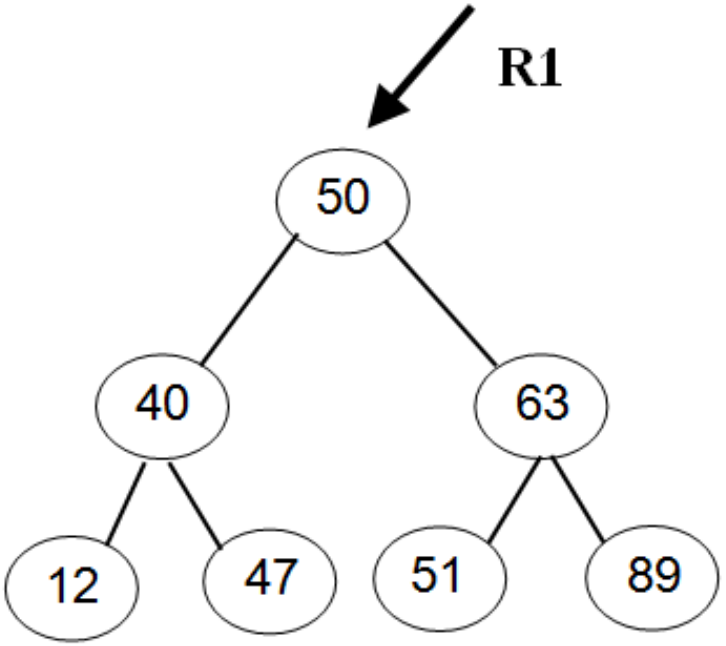


Imprimir uma Árvore

ImprimeTodos (parâmetro por referência R do tipo ABB);

```
Se (R != Null)
Então {   Escreva(R→Info);          // imprime a informação da raiz
          ImprimeTodos(R→Esq);      // imprime todos da Subárvore Esquerda
          ImprimeTodos(R→Dir); }    // imprime todos da Subárvore Direita
} // fim ImprimeTodos - PréOrdem
```

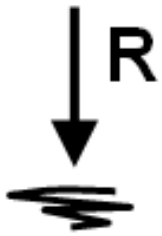


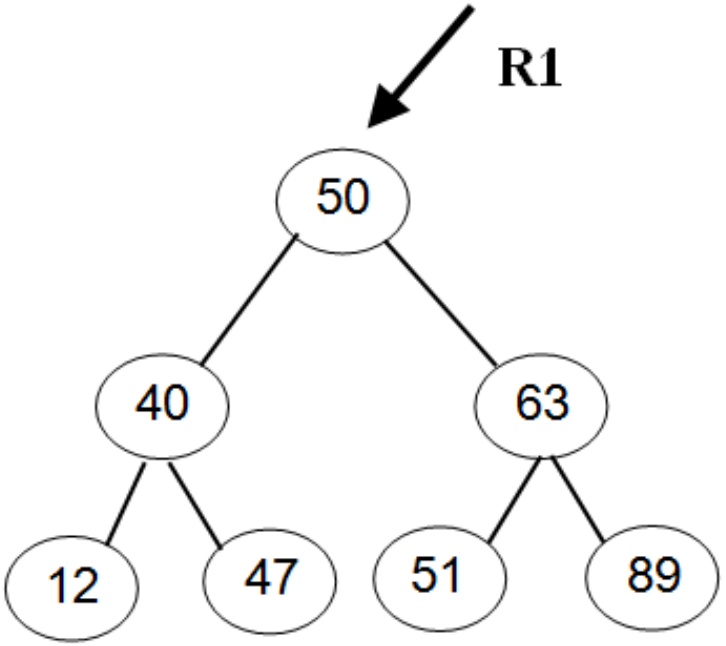


Imprimir uma Árvore

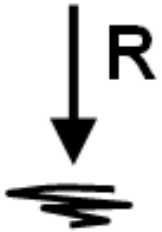
ImprimeTodos (parâmetro por referência R do tipo ABB);

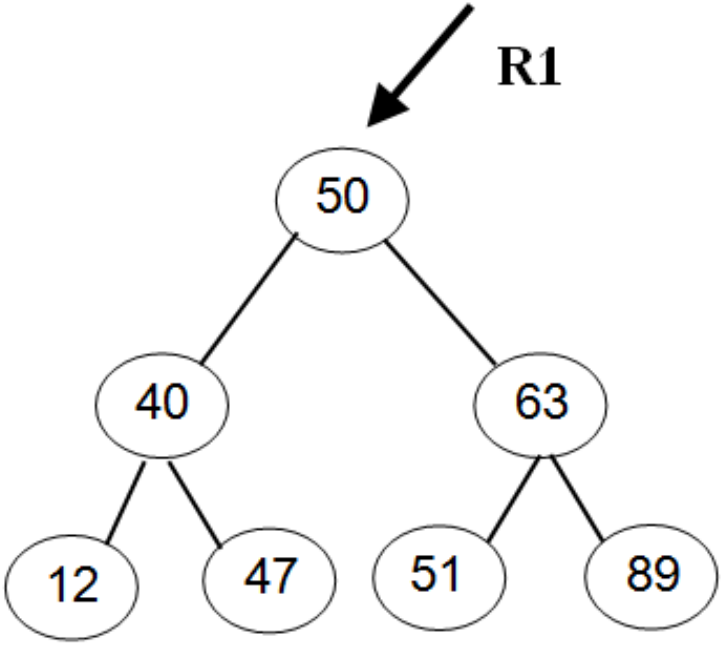
```
Se (R != Null)
Então {   ImprimeTodos(R→Esq);     // imprime todos da Subárvore Esquerda
          Escreva(R→Info);         // imprime a informação da raiz
          ImprimeTodos(R→Dir); }   // imprime todos da Subárvore Direita
} // fim ImprimeTodos - InOrdem
```





Exercícios





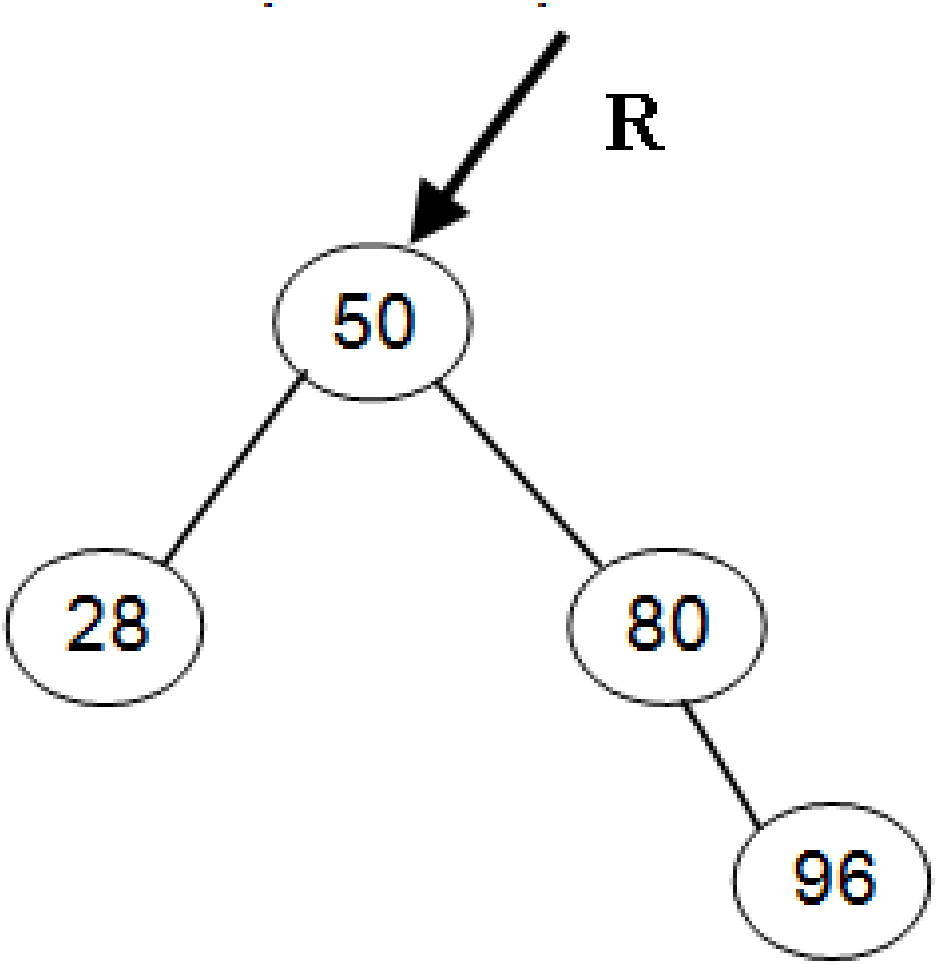
Exercício 8.4 Soma dos Elementos de uma Árvore

Exercício 8.5 Número de Nós com um Único Filho

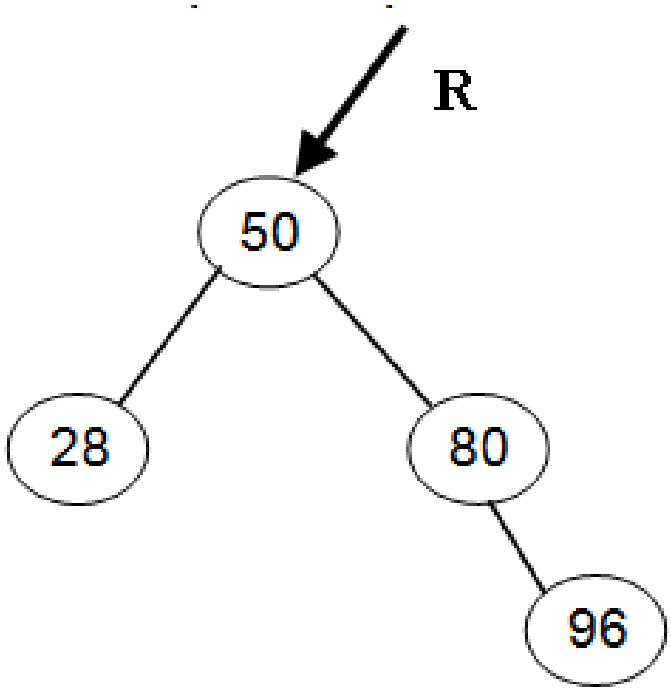
Exercício 8.6 Árvores São Iguais?

Exercício 8.7 É Árvore Binária de Busca?

ABB: Onde Inserir o 37?



Inserindo Novos Valores em uma ABB

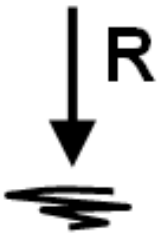


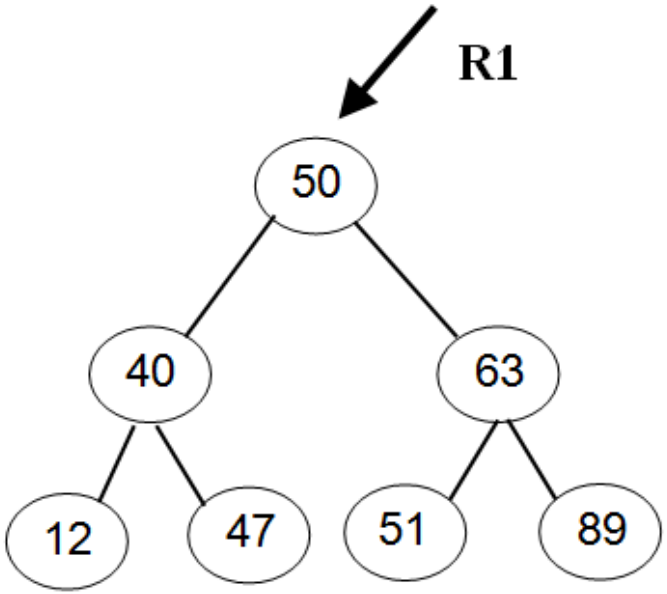
- Inserir novos elementos como Nós Terminais (sem Filhos);
- Procurar o lugar certo, considerado o critério que define uma ABB, e então inserir.

Insere

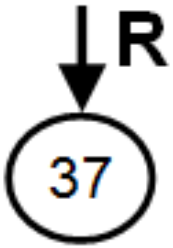
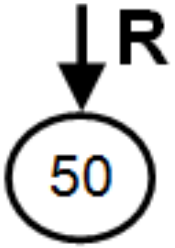
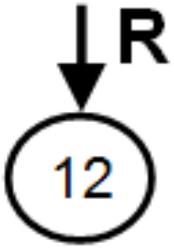
Insere (parâmetro por referência **R** do tipo ABB, parâmetro **X** do tipo Inteiro, parâmetro por referência **Ok** do tipo Boolean);

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





Insere

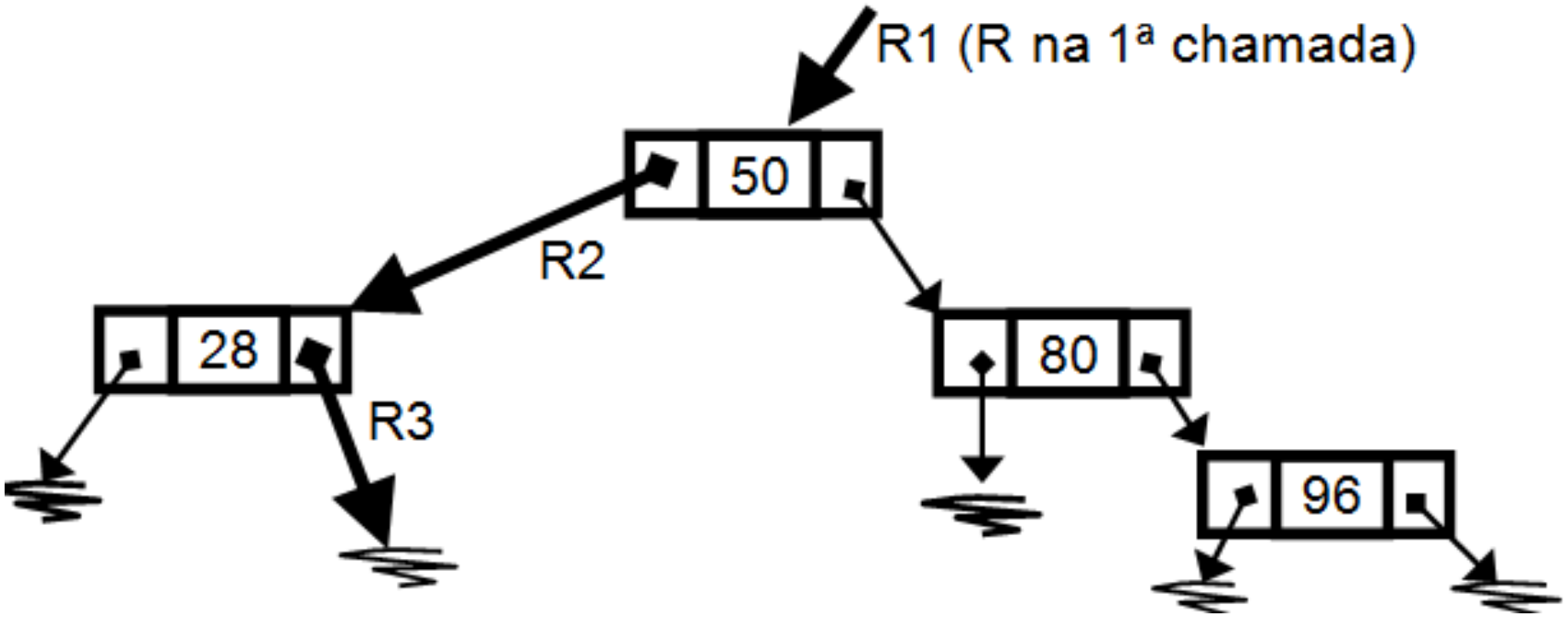
Caso	X	R
Caso 1: Árvore Vazia	37	
Caso 2: R→Info = X	37	
Caso 3: X < R→Info	37	
Caso 4: X > R→Info	37	

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

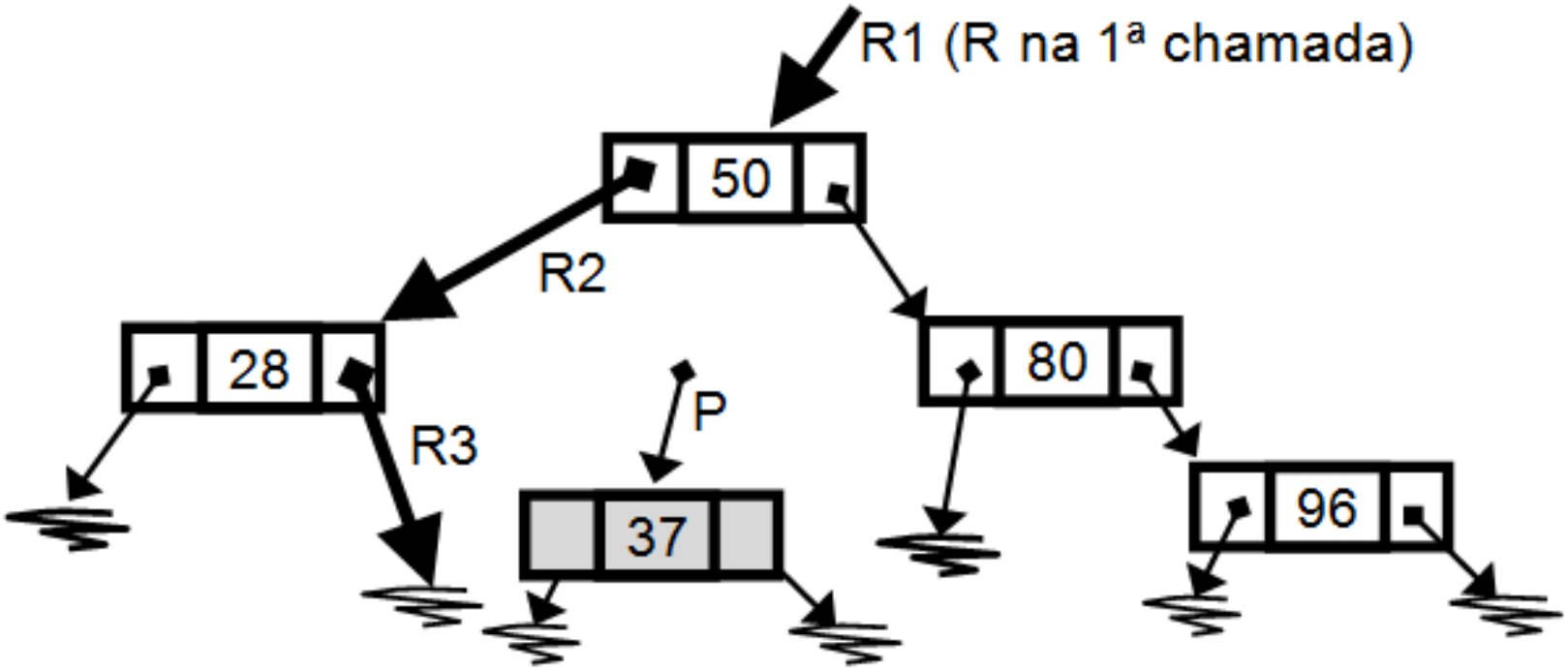
Variável P do tipo NodePtr;

```
Se (R == Null)
Então {   P = NewNode;      // Caso 1: Achou o lugar; insere e acaba
          P→Info = X;
          P→Dir = Null;
          P→Esq = Null;
          R = P;
           P = Null;
          Ok = Verdadeiro; }
Senão {  Se (X == R→Info)
           Então  Ok = Falso; // Caso 2: X já está na árvore; não insere;
           Senão { Se (R→Info> X)
                      Então Insere (R→Esq, X , Ok)  // Caso 3: tenta na Es
                      Senão Insere(R→Dir, X, Ok); // Caso 4: tenta na Dir
                     } // fim senão
         } // fim senão
} // fim Insere ABB
```

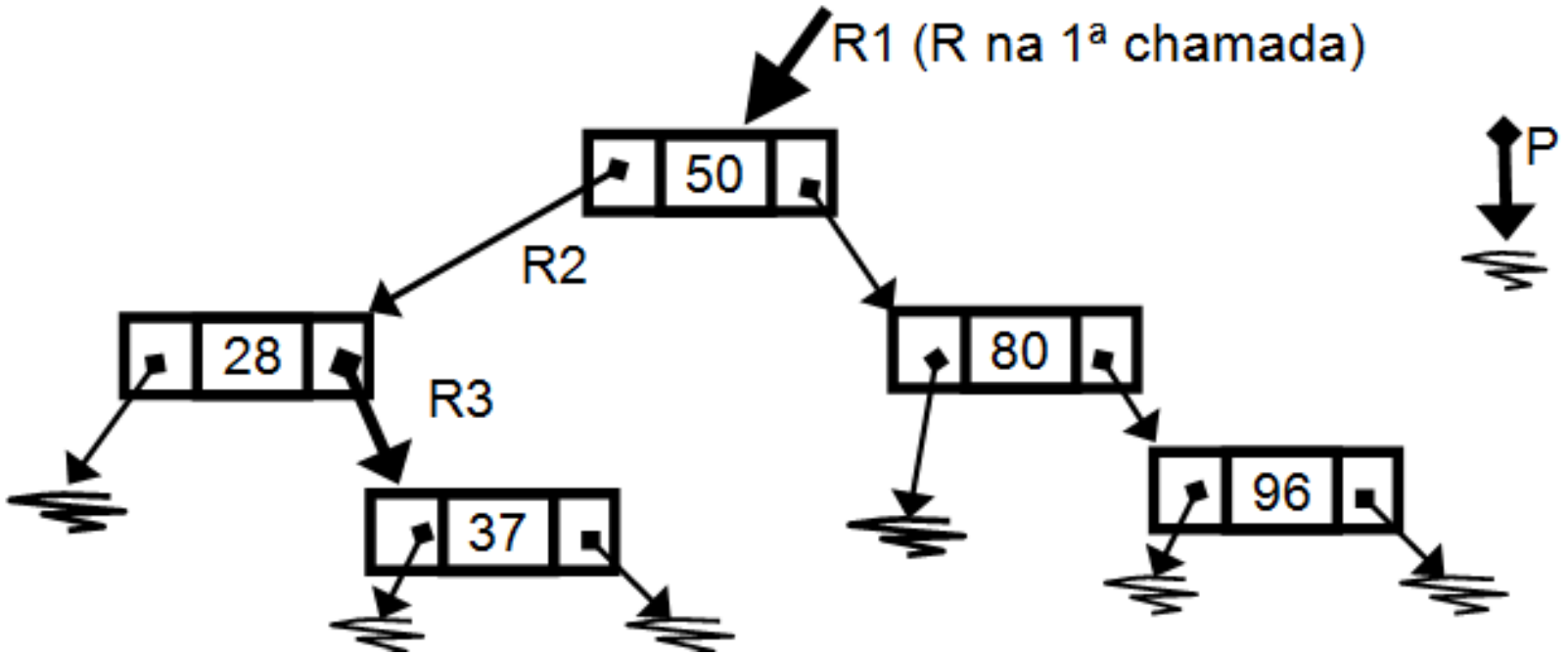
Execução de Insere para X = 37



Execução de Insere para X = 37



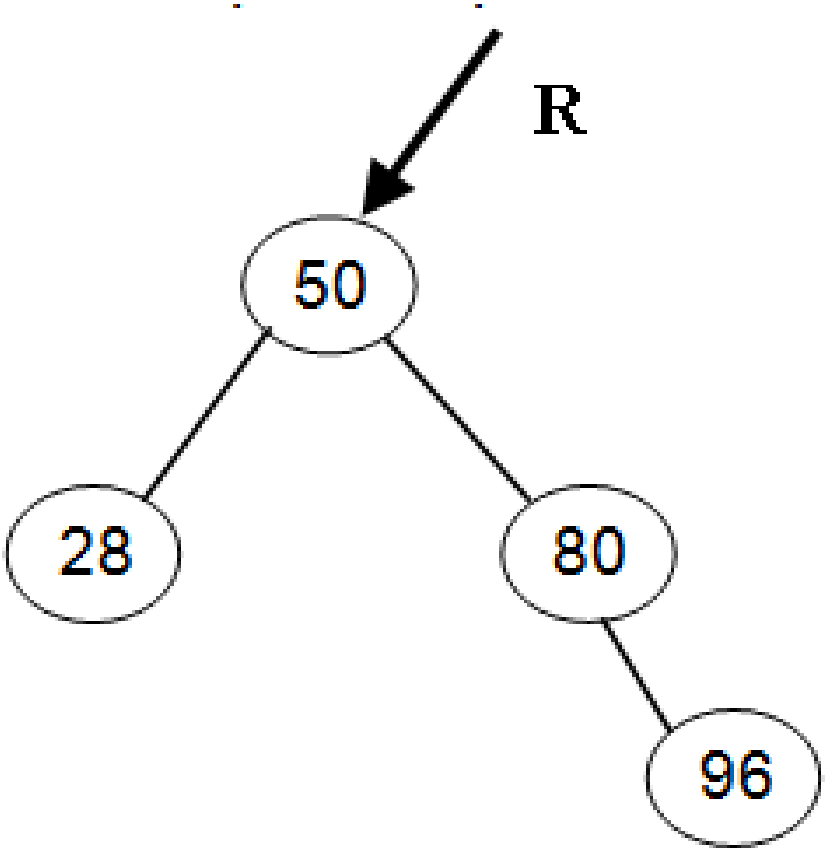
Execução de Insere para X = 37



Remove

Como consertar a árvore ao remover:

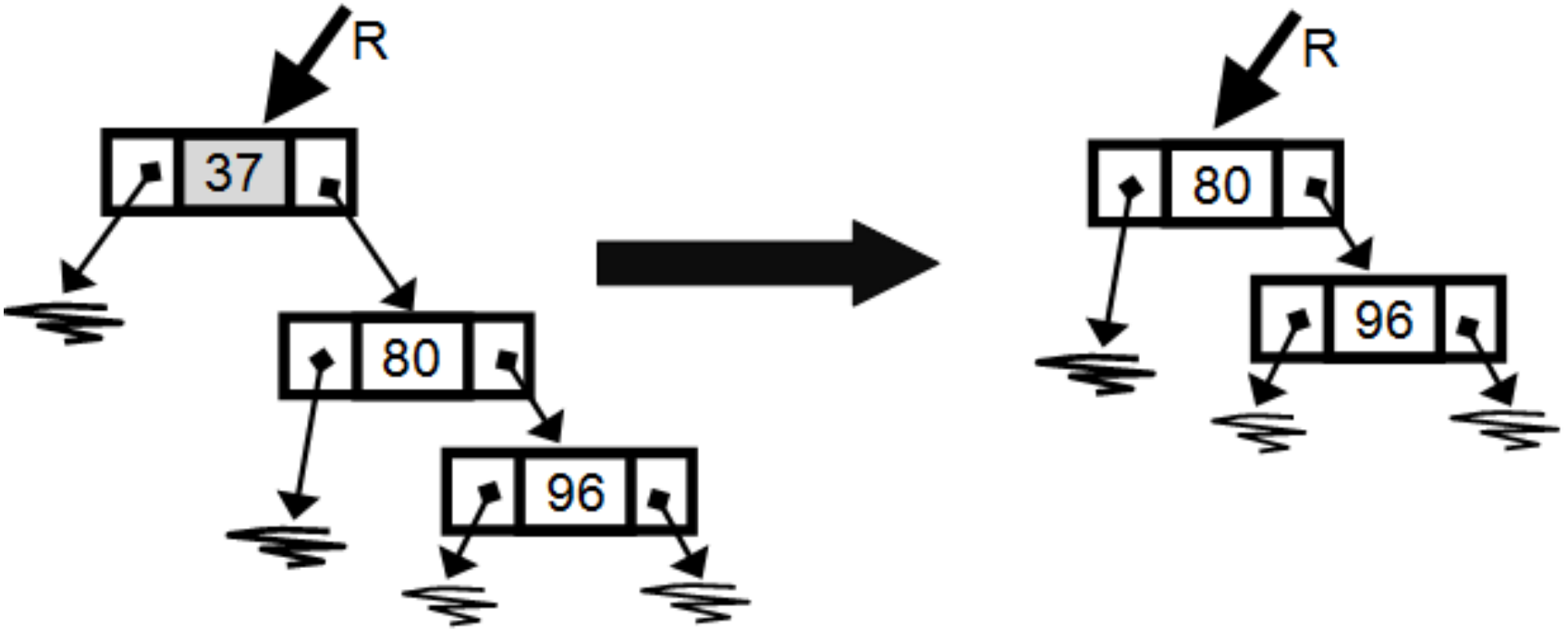
- 28
- 80
- 50



Remove

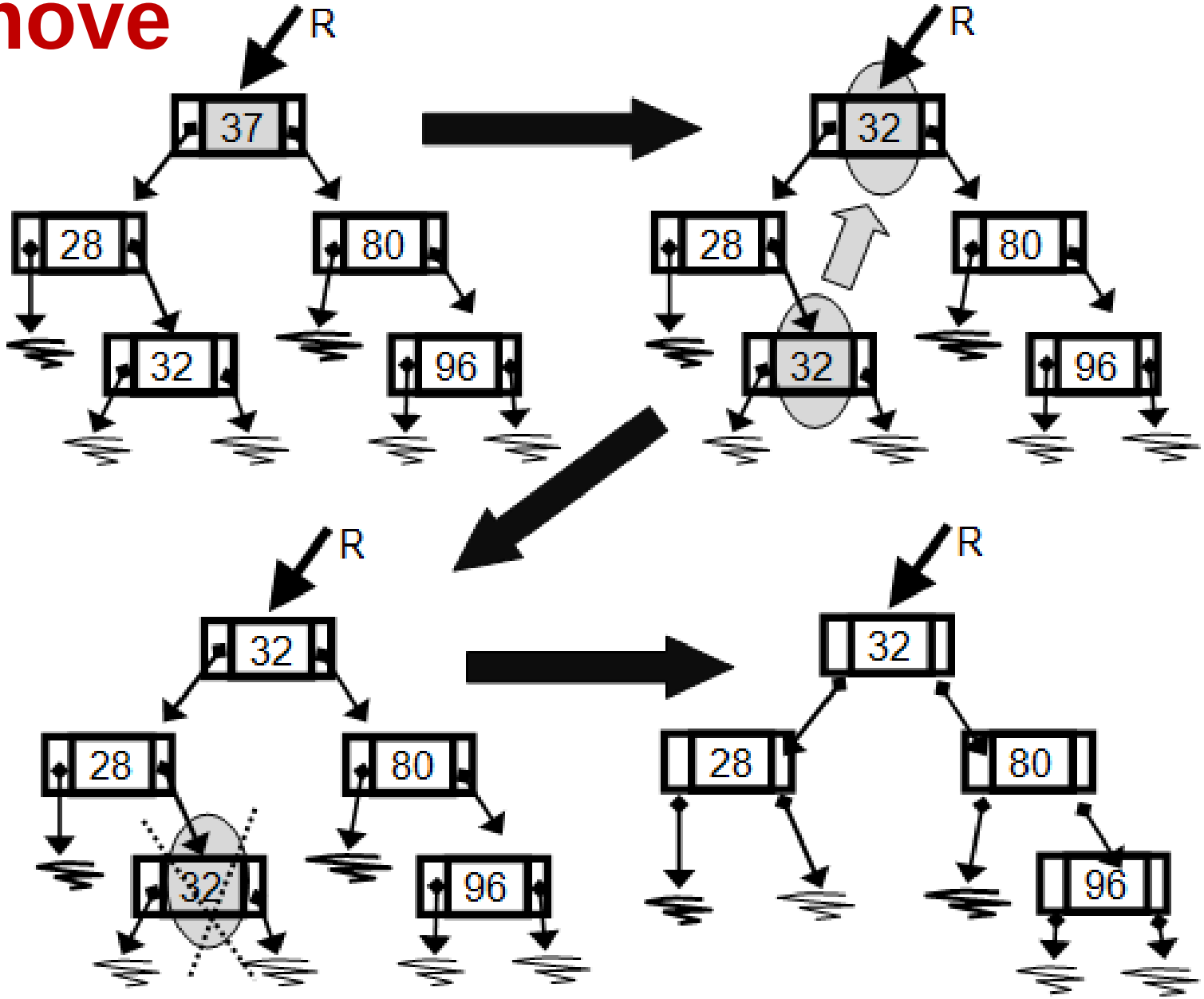
(a) Caso de Remoção 2a: Nó Sem Filhos

Remove



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

Remove



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

Exercício 8.9 – Remove de ABB

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

/* Remove o valor X da ABB de Raiz R. Ok retorna Verdadeiro para o caso de X ter sido encontrado e removido, e Falso caso contrário. */

Exercício 8.10 – Cria e Vazia

Por Que uma Árvore Binária de Busca É Boa?

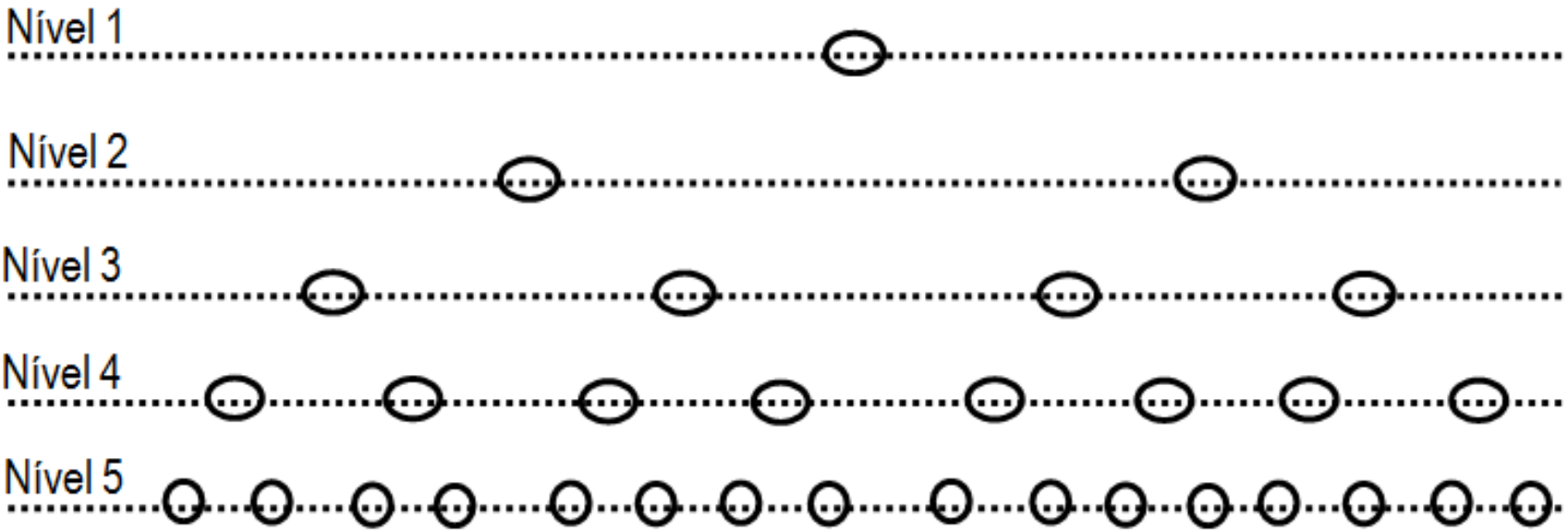


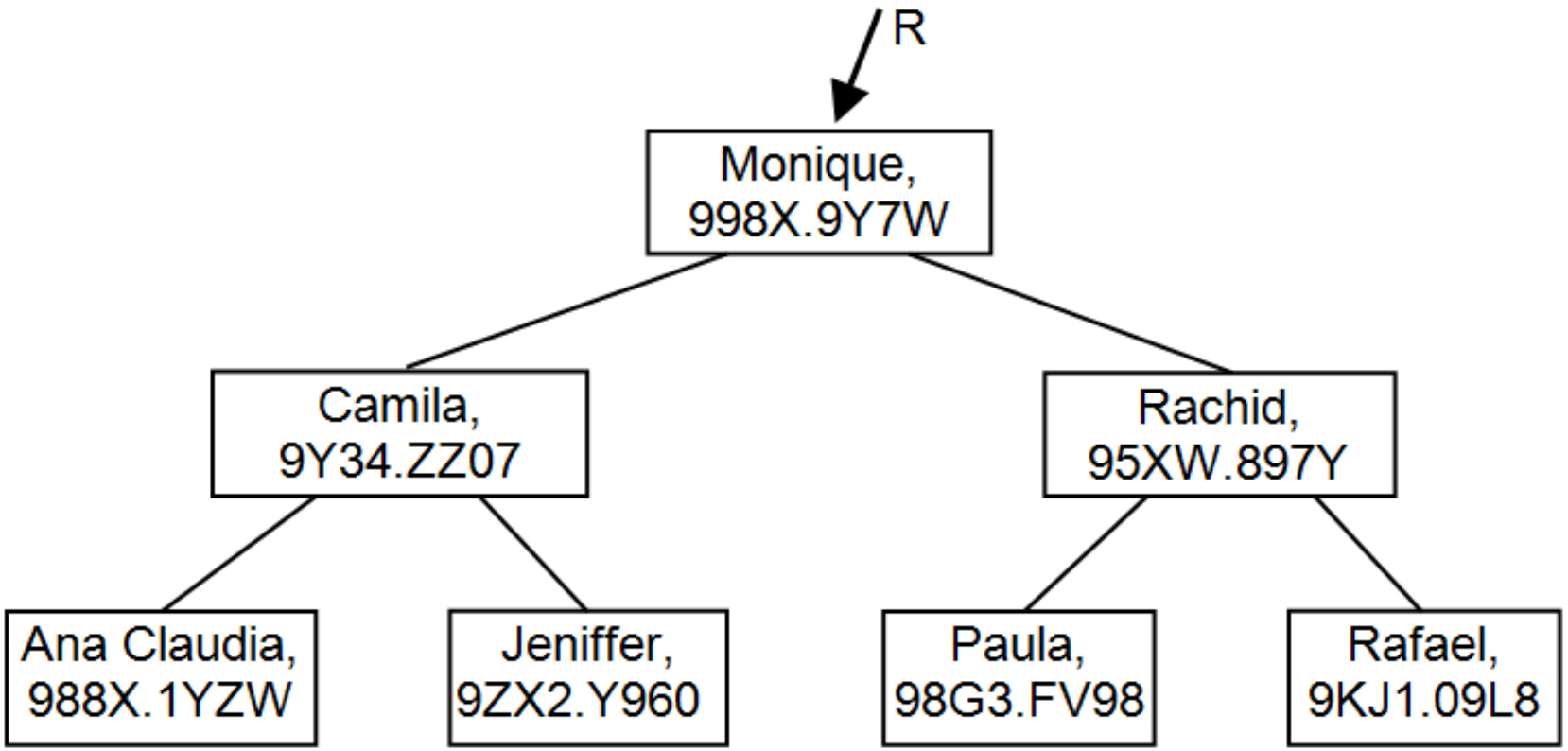
ABB Uniformemente Distribuída

Por Que uma Árvore Binária de Busca É Boa?

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)

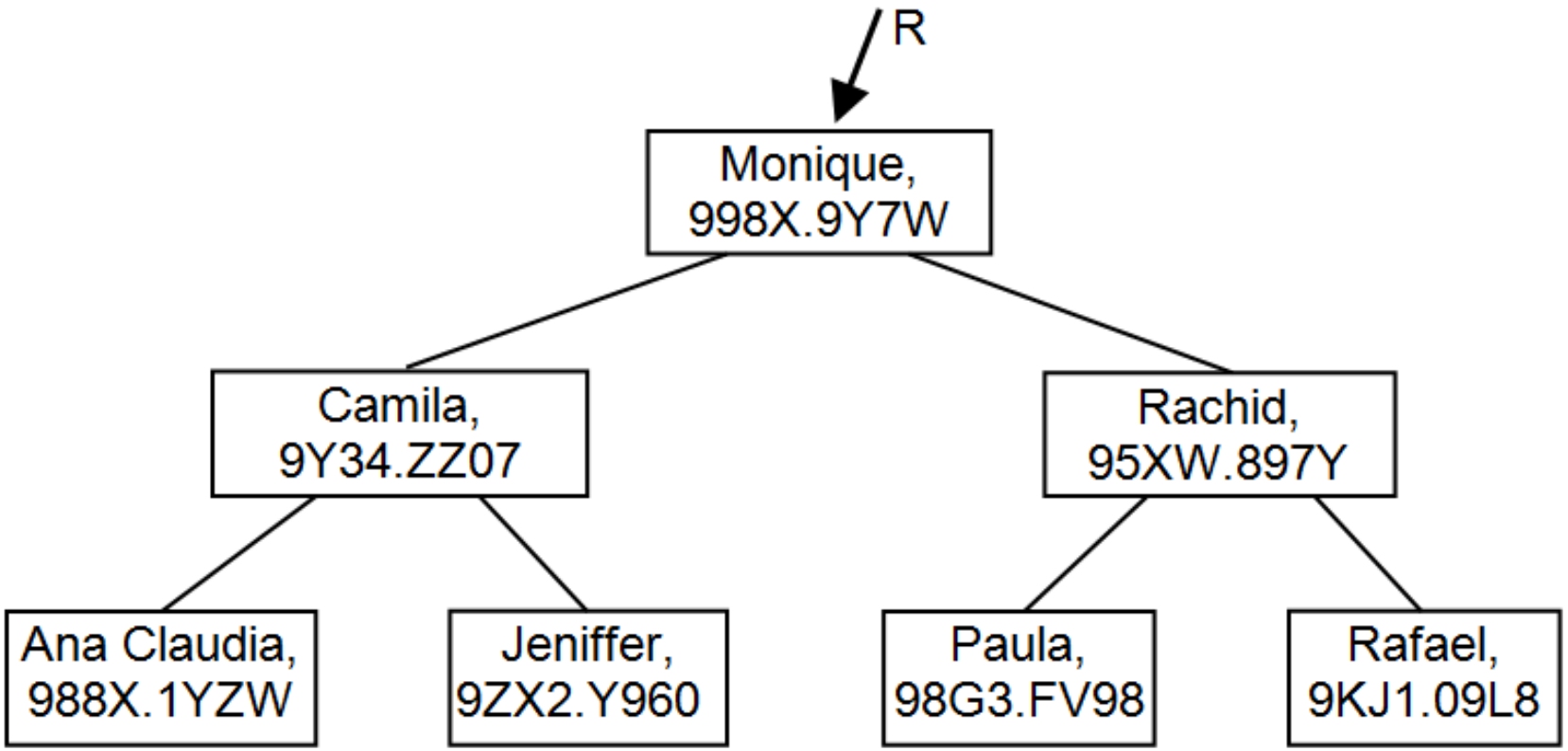
ABB Uniformemente Distribuída

Aplicações de Árvores



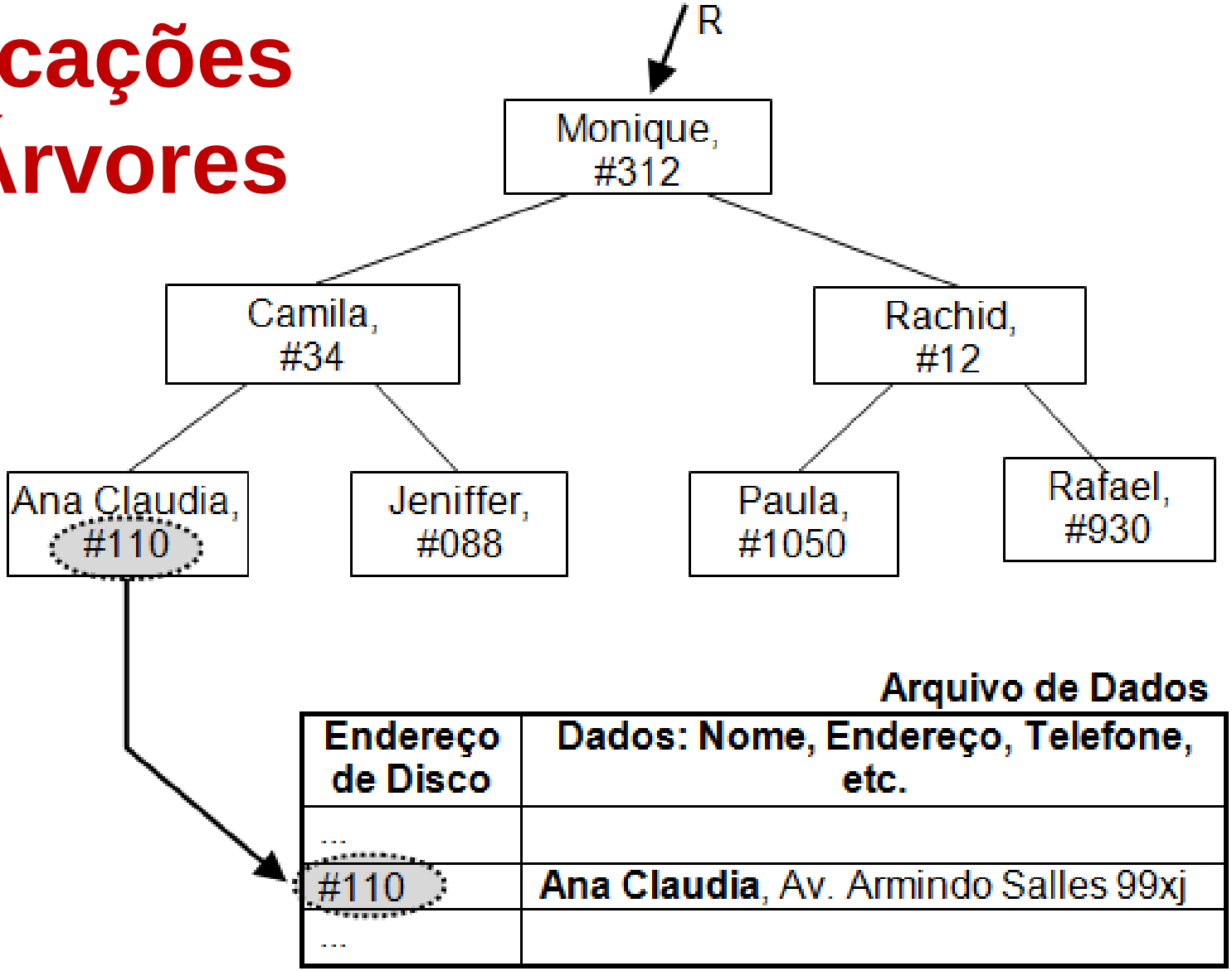
Chave de Busca e Outras Informações no Nó

Aplicações de Árvores



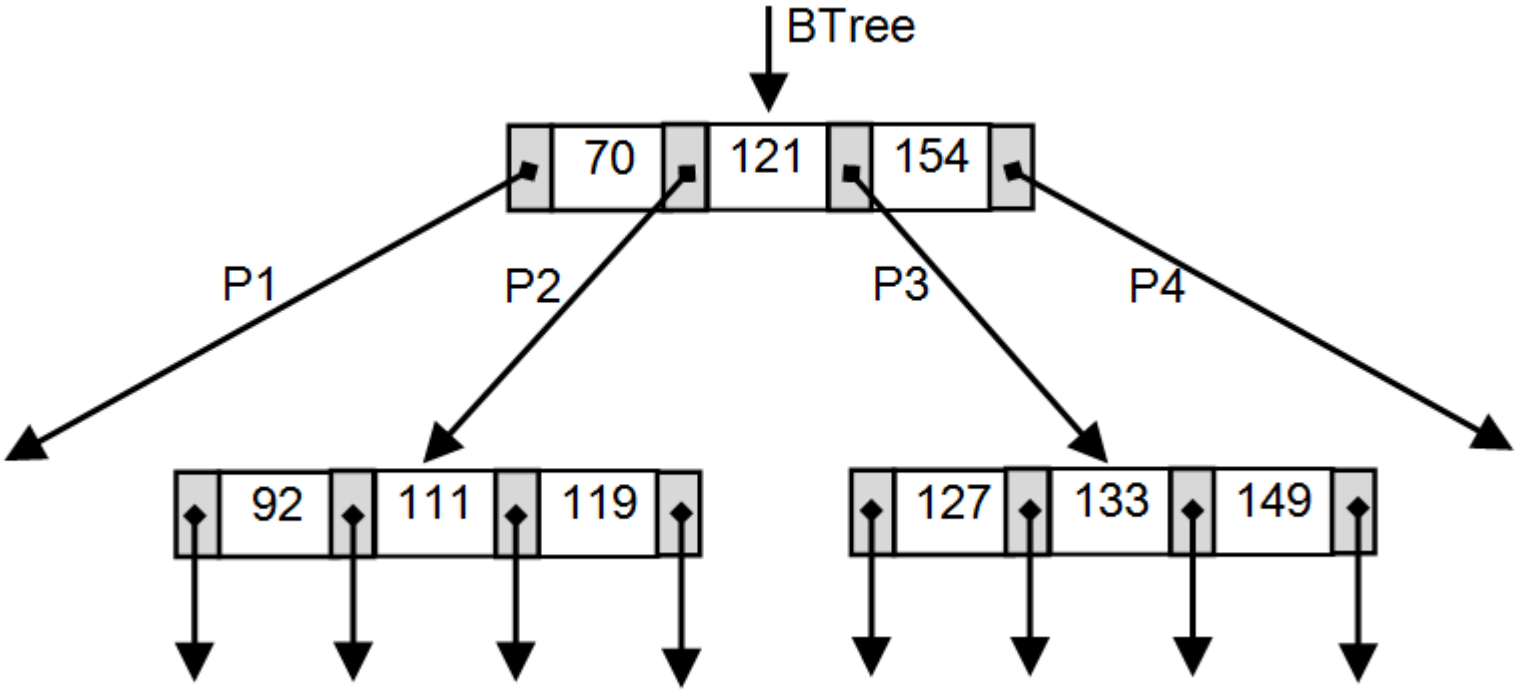
Chave de Busca e Outras Informações no Nó

Aplicações de Árvores

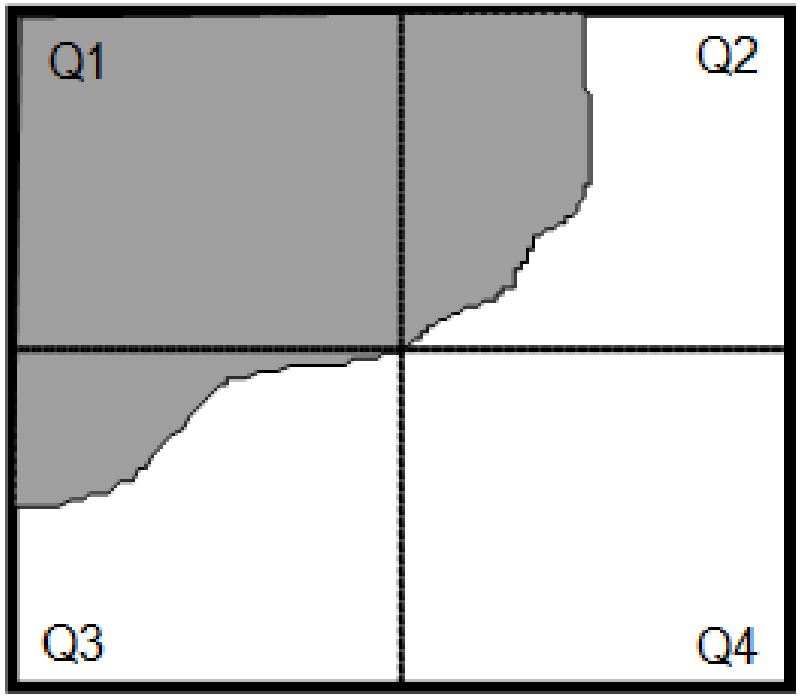


Índice para um Arquivo

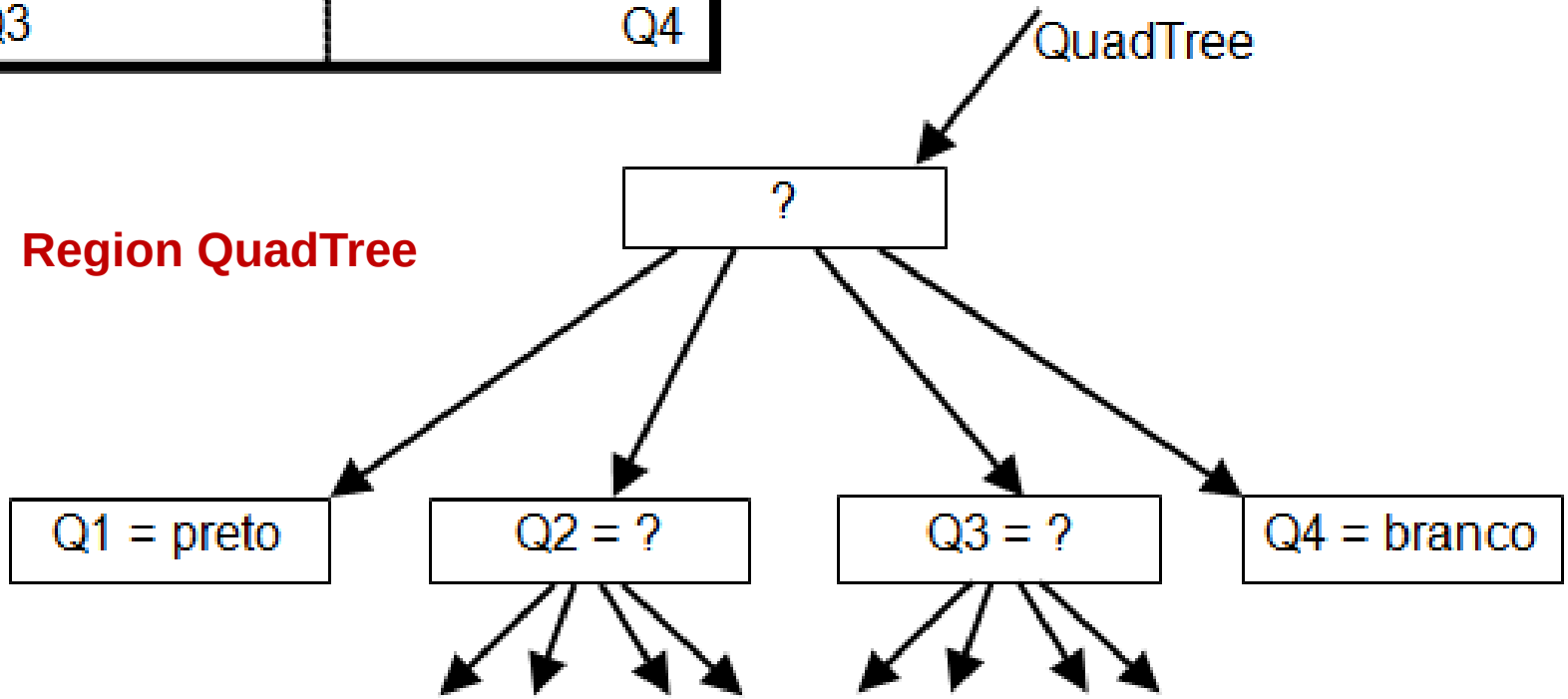
Aplicações de Árvores



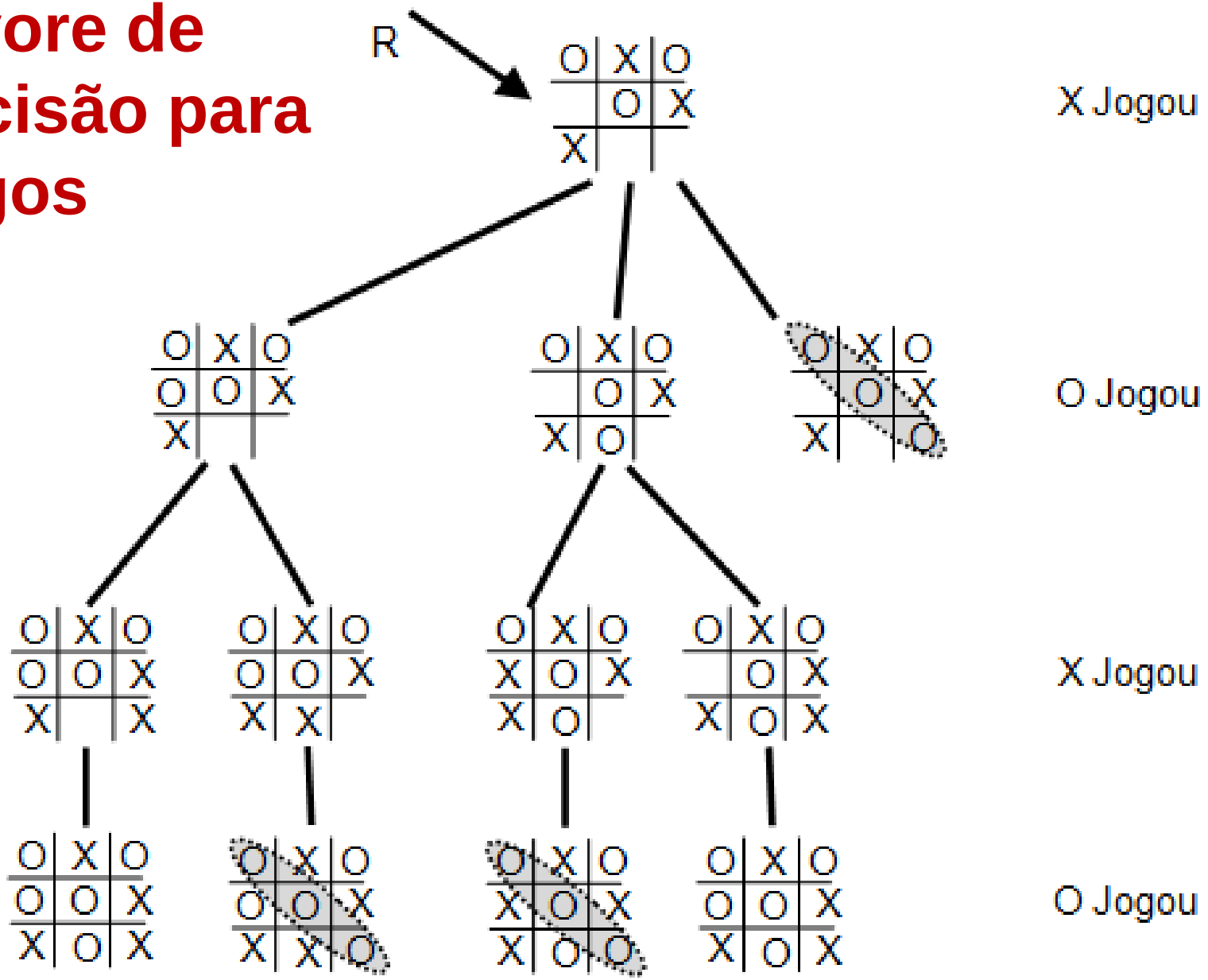
B-Trees



Aplicações de Árvores: QuadTrees



Árvore de Decisão para Jogos



Agilidade e Suporte a Decisões

Uma ABB permite consultas rápidas, mesmo quando a quantidade de elementos é grande.

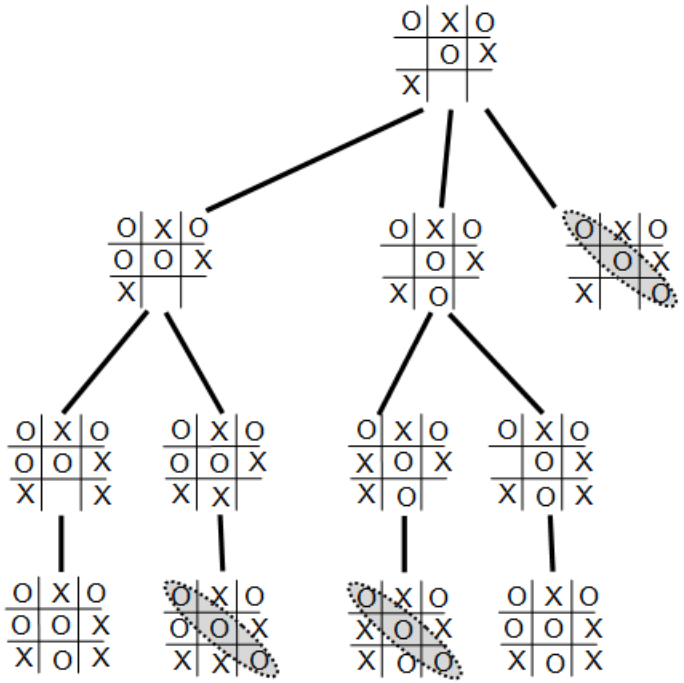
Árvores de Decisão podem ser utilizadas para dar inteligência a um jogo.

É possível propor Árvores diferenciadas que atendam a necessidades de sua aplicação.

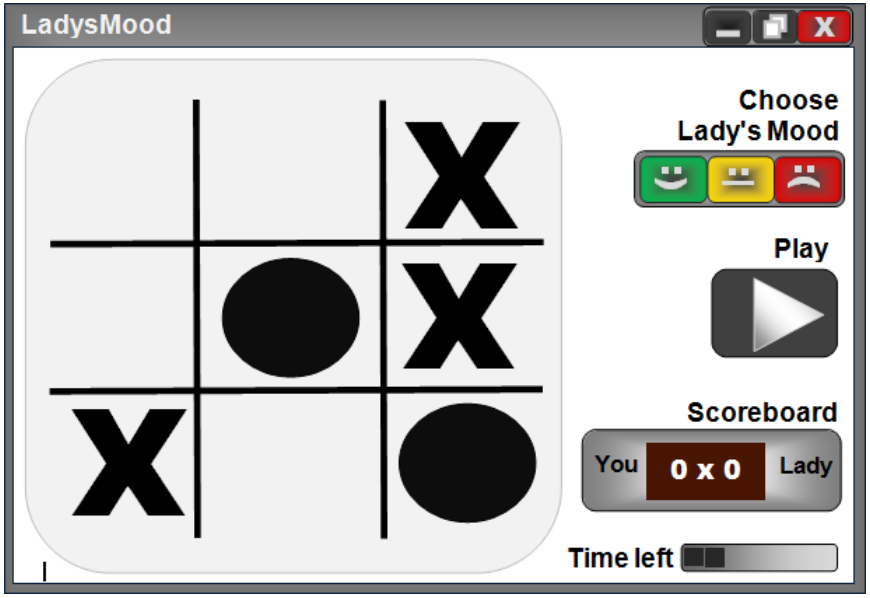
Avanço de Projeto

Exercício 8.12 Discutir Aplicações de Árvores em Jogos

Exercício 8.13 Avançar o Projeto do Desafio 4: Defina Regras, Escolha um Nome e Inicie o Desenvolvimento do Seu *Jogo*



Comece a Desenvolver Seu Jogo Agora!



Dê personalidade, e dê inteligência ao seu jogo!
Mostre isso para os seus amigos!

Estruturas de Dados com Jogos

Aprender a programar pode ser divertido!