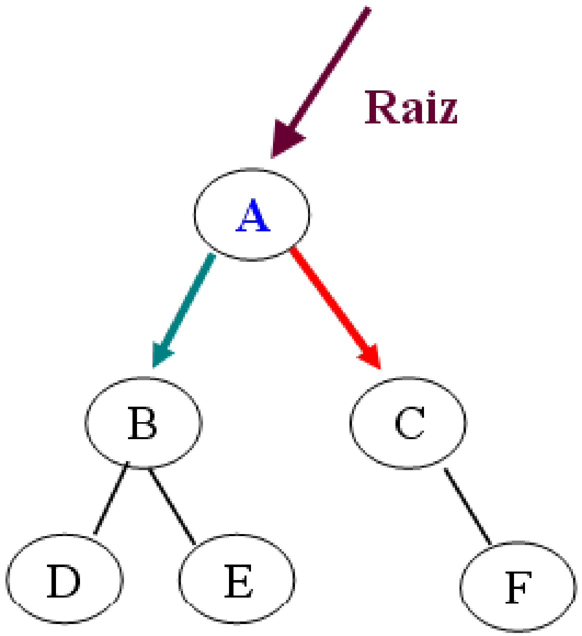
Árvore Binária

Referência:

TENENBAUM, A.; LANGSAM, Y.; AUGENSTEIN, M.- Estruturas de Dados Usando C. São Paulo: Makron Books, 1995. Capítulos 5 e 7.2.

Definição: Árvores Binárias

- Árvores Binárias são aquelas árvores com grau 2. Ou seja, nas árvores binárias, cada nó da árvore tem no máximo 2 filhos.



Terminologia:

- Filho esquerdo
- Filho direito
- Informação

Árvore Binária

- Em cada nó de uma árvore binária temos a **Informação**, um **apontador para o Filho Direito**, e um **apontador para o Filho Esquerdo**.

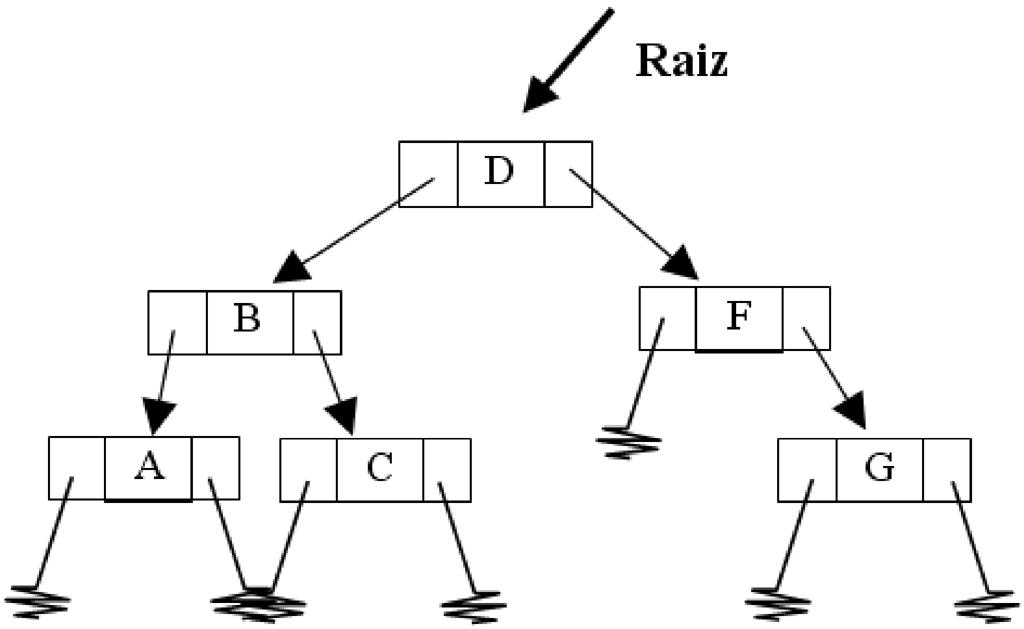


ABB vazia

Declaração em uma Linguagem de Programação

Pascal	C

type ABB = ^node;
node = record
info : char;
dir : ABB;
esq : ABB;
end;
Var Raiz : ABB;
	struct node {
	char info;
	struct node *dir;
	struct node * esq;
	};
	typedef struct node *ABB;
	ABB Raiz;

Árvore Binária de Busca

ABB

Árvores Binárias de Busca - ABB

- Árvores Binárias de Busca também são chamadas de Árvores Binárias de Pesquisa, e Árvores Binárias Ordenadas. Mas o nome mais comum é: Árvores Binárias de Busca ou, abreviadamente, ABBs.

Árvore Binária de Busca

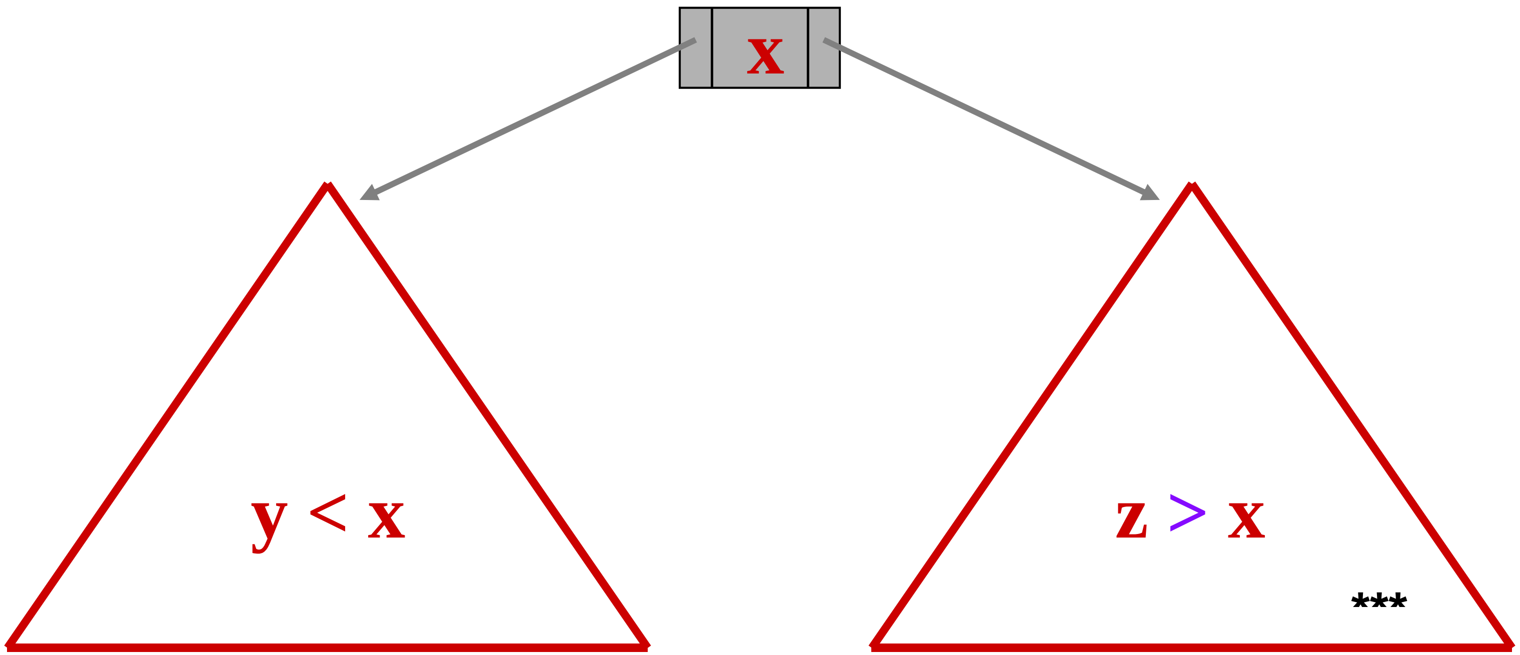
- Foi proposta para ser implementada em memória primária.
- Estrutura de dados dinâmica, ou seja, permite inclusão e remoção de valores.
- São árvores onde os elementos são organizados de forma que:
  - Todos os elementos na sub-árvore esquerda de cada nó *k* têm valor menor ao valor do nó *k*.
  - Todos os elementos na sub-árvore direita de cada nó *k* têm valor maior do que o valor do nó *k*. ***

Definição: Árvore Binária de Busca - ABB

- Uma árvore binária com raiz R é uma ABB se:
  - (1) A chave (informação) de cada nó da sub-árvore esquerda de R é menor (*) do que a chave (informação) do nó R;
  - (2) A chave de cada nó da sub-árvore direita de R é maior (*) do que a chave do nó R; ***
  - (3) As sub-árvores esquerda e direita também são ABBs.

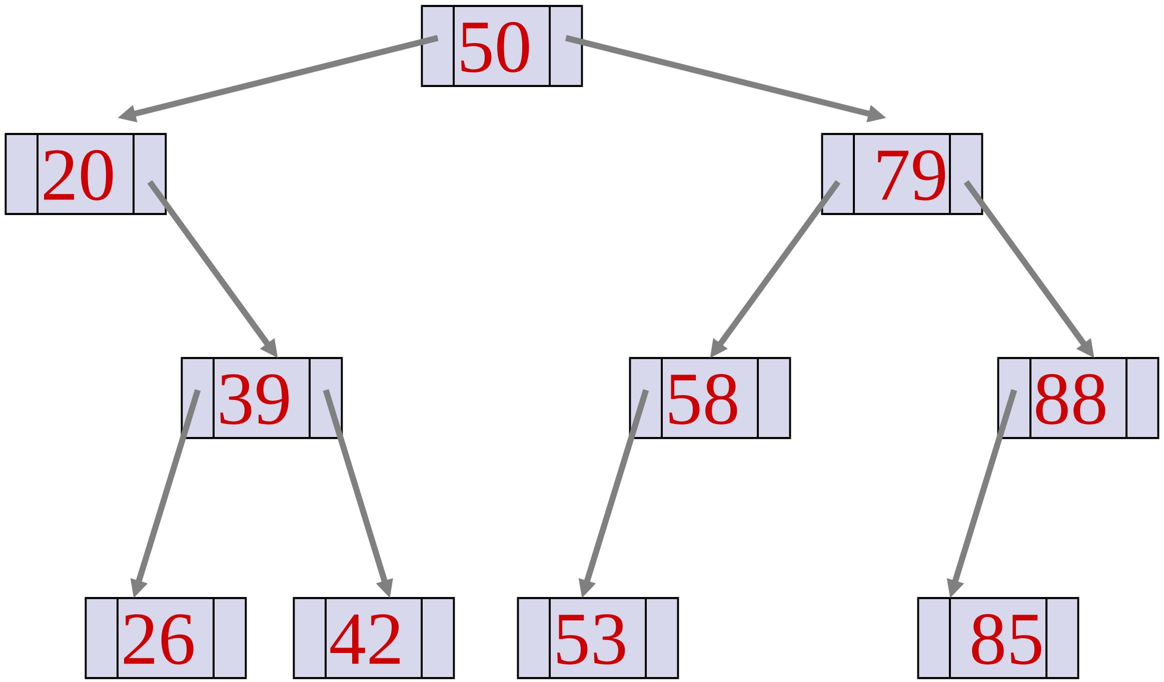
ABB

- Árvores binárias onde os elementos são organizados de forma que:

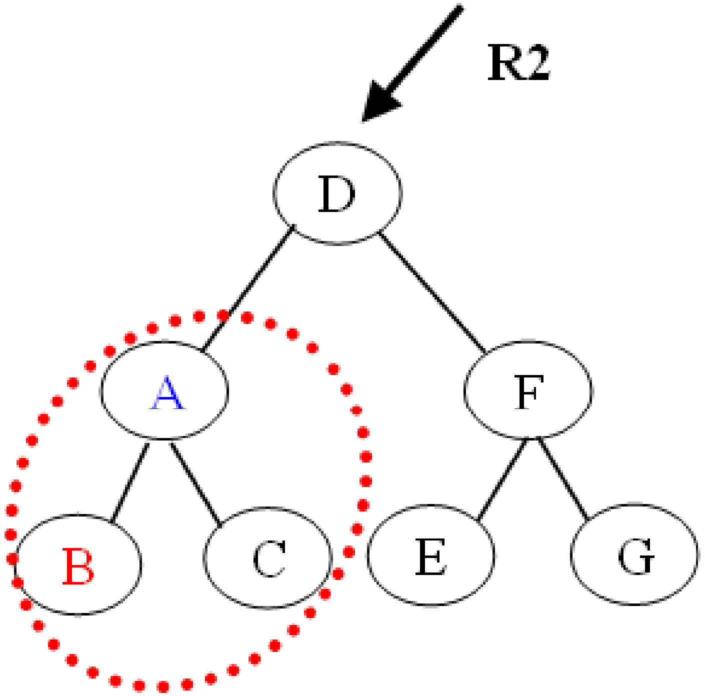


ABB

- Exemplo: 50, 20, 39, 79, 26, 58, 88, 85, 42, 53.



É ABB?



Não é uma ABB

Problemas recursivos

- ***Dica para a Solução de Problemas Recursivos***
  - *Para resolver um problema recursivamente, precisamos identificar os casos em que conseguimos dar a resposta ao problema de imediato, resolvendo de vez o problema, e os casos em que não conseguimos resolver de imediato.*
  - *Nos casos que não conseguimos resolver de imediato, precisamos decompor o problema em problemas menores, nos aproximando da solução.*
  - *Em outras palavras, quando não conseguimos resolver o problema de imediato, "empurramos com a barriga", e deixamos para resolver o problema em um outro momento, fazendo uma chamada recursiva.*

Exercício

- Implemente uma função recursiva que calcule o fatorial de um número.

Definição Recursiva de Fatorial
Fatorial de 0 é 1
Fatorial de N é N * fatorial (N - 1)

<table>
<tr><th>Pascal</th><th>C</th></tr>
<tr><td><pre>function fat(n : integer) : integer;
        begin
        if n = 0
        then fat := 1
        else fat := n * fat (n - 1);
        end;</pre></td><td><pre>int fat(int n) {
   if (n==0)
        return (1);
   else return (n * fat (n-1));
}</pre></td></tr>
</table>

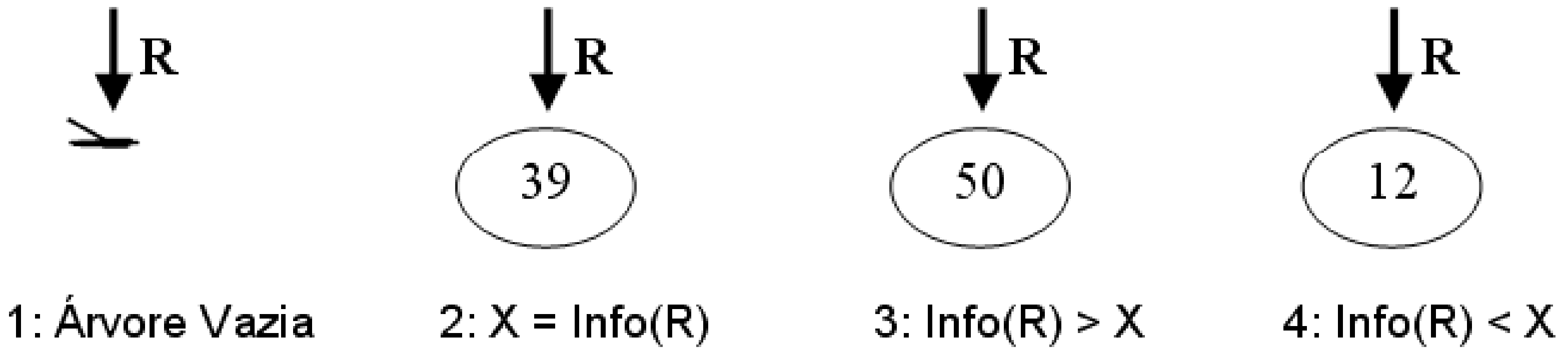
Simular fatorial

Inserção

- Ocorre sempre em uma folha;
  - Top-Down (cima p/ baixo)
- Princípio do algoritmo:
  - Localizar um nó pai que não tem sub-ramo para a chave inserida.

- Algoritmo não garante que os nós estejam sempre completos;

Algoritmo de Busca em ABB



Resolver o problema recursivamente

Busca em ABB
Resumo dos caso

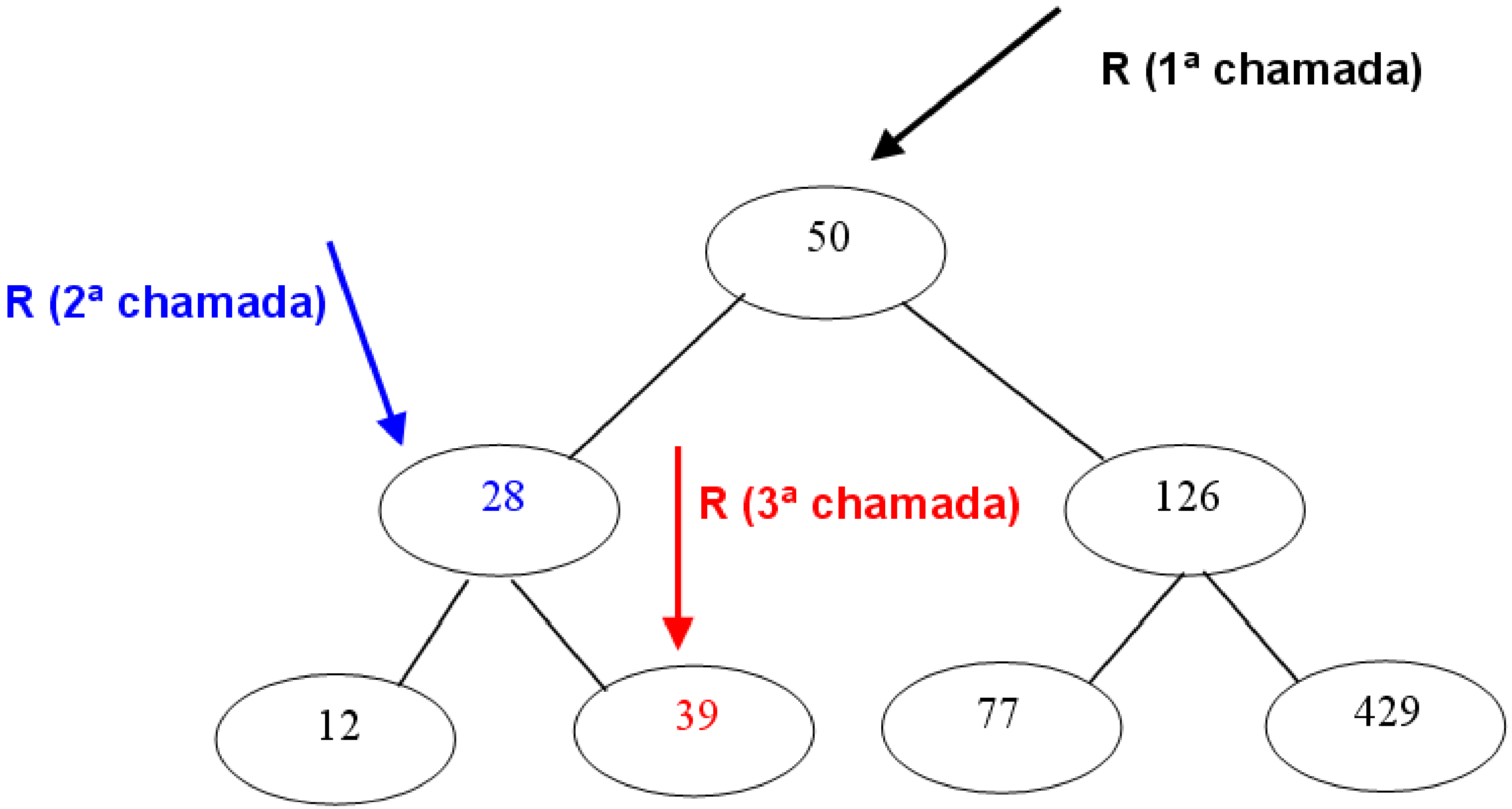
1	Árvore Vazia	X não está na árvore, e acaba o algoritmo.
2	X = Info(R)	X está na árvore, e acaba o algoritmo.
3	X < Info(R)	X PODE ESTAR na sub-árvore esquerda de R. O algoritmo não acaba ainda. Fazemos uma chamada recursiva
4	X> Info(R)	X PODE ESTAR na sub-árvore direita de R. O algoritmo não acaba ainda. Fazemos uma chamada recursiva

Está Na Árvore?

Boolean **Está_na_Árvore**(variável por referência R tipo ABB, variável X tipo Inteiro)

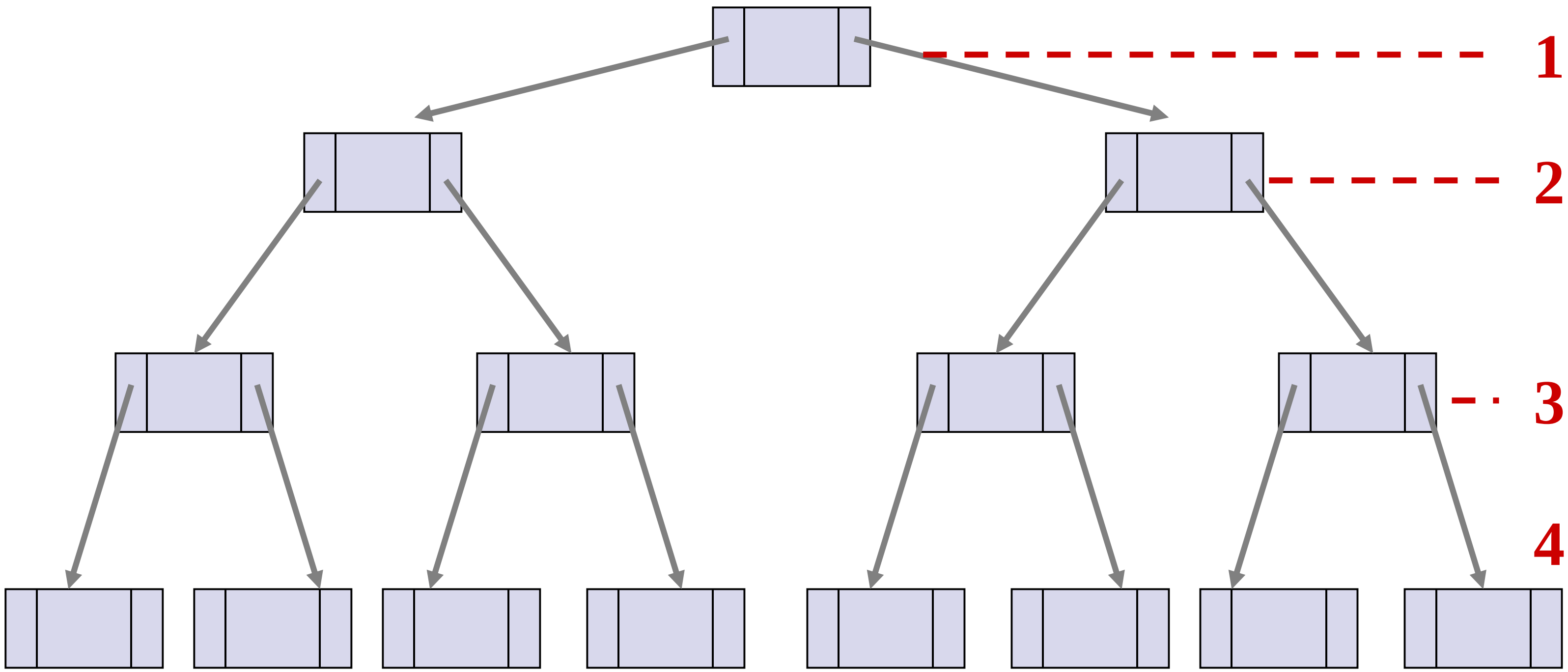
Se R = Null
Então Retorne Falso // caso 1 a árvore é vazia; X não está na ABB
Senão Se X = Info(R)
    Então Retorne Verdadeiro // caso 2: X está na árvore, acabou o algoritmo
    Senão Se Info(R)> X
        Então Retorne (**Está_Na_Árvore**(**Esq**(R), X))
// retorna o resultado da chamada recursiva, para a busca de X na
// sub-árvore esquerda de R (caso 3)
Senão Retorne (**Está_Na_Árvore**(**Dir**(R), X))
// retorna o resultado da chamada recursiva, para a busca de X na
// sub-árvore direita de R (caso 4)

Árvore para Busca X=39



Complexidade

- O algoritmo de busca tem tempo logarítmico.



Exercícios

1. **Imprimir**

   Imprimir(por referência R do tipo ABB)

   // operação para imprimir a informação de cada nó da árvore R

2. **Imprimir em Ordem Crescente**

   Imprimir(por referência R do tipo ABB)

   // operação para imprimir a informação de cada nó da árvore R, em ordem crescente

3. **Imprimir em Ordem Decrescente**

   Imprimir(por referência R do tipo ABB)

   // operação para imprimir a informação de cada nó da árvore R, em ordem decrescente

Imprimir todas as chaves da árvore R (Pré-Ordem)

Imprime(por referência R tipo ABB)

```
Se R <> Null  então
      Escreve( Info( R ) ) // imprime a chave da raiz com printf, ou outro comando
      Imprime( Esq( r ) ) // imprime todas as chaves da sub-árvore esquerda
      Imprime( Dir( r ) ) // imprime todas as chaves da sub-árvore direita
```

imprimir em ordem crescente. (Em-Ordem)

```
Imprime( por referência R tipo  ABB  )
Se R <> Null então
      Imprime( Esq( r ) ) // imprime todas as chaves da sub-árvore esquerda
      Escreve( Info( R ) ) // printf, ou outro comando
      Imprime( Dir( r ) ) // imprime todas as chaves da sub-árvore direita
```

imprimir em ordem decrescente.

Imprime(por referência R tipo ABB)

```
Se R <> Null então
      Imprime( Dir( r ) ) // imprime todas as chaves da sub-árvore esquerda
      Escreve( Info( R ) ) // printf, ou outro comando
      Imprime( Esq( r ) ) // imprime todas as chaves da sub-árvore direita
```

Percurso

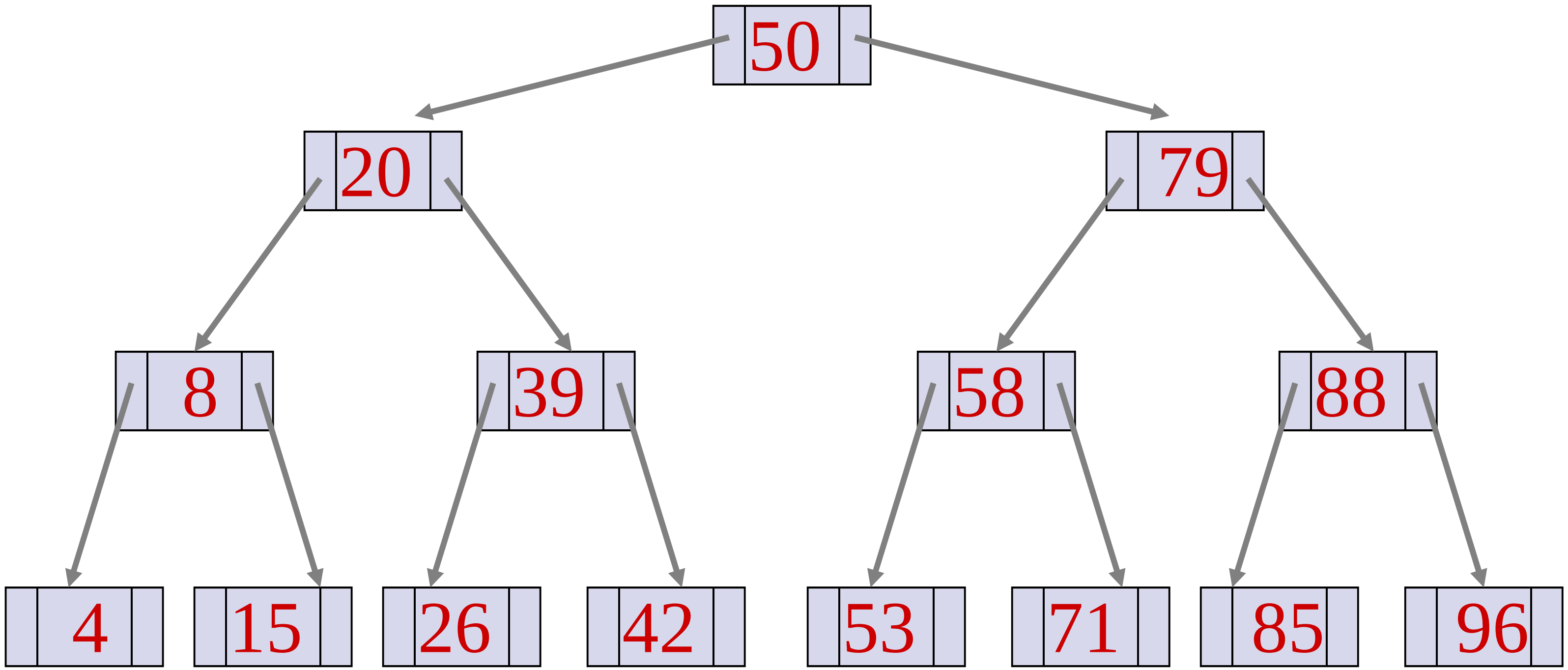
- Percorrer uma árvore visitando cada **nó** uma única vez gera uma seqüência linear
  - então passa a ter sentido falar em sucessor e predecessor de um nó segundo um determinado percurso.
- Há três maneiras recursivas de se percorrer árvores binárias:
  - Travessia Pré-ordem;
  - Travessia Em-ordem;
  - Travessia Pós-ordem;

Travessia Pré-Ordem

- Notação: pré-fixada
- Visita o nó antes de passar na sua sub-árvore esquerda.
- Algoritmo:
  1. se árvore vazia; fim
  2. **imprime conteúdo do nó raiz**
  3. percorrer em pré-ordem a sub-árvore esquerda
  4. percorrer em pré-ordem a sub-árvore direita

Travessia Pré-Ordem

- Exemplo: 50, 20, 8, 4, 15, 39, 26, 42, 79, 58, 53, 71, 88, 85, 96

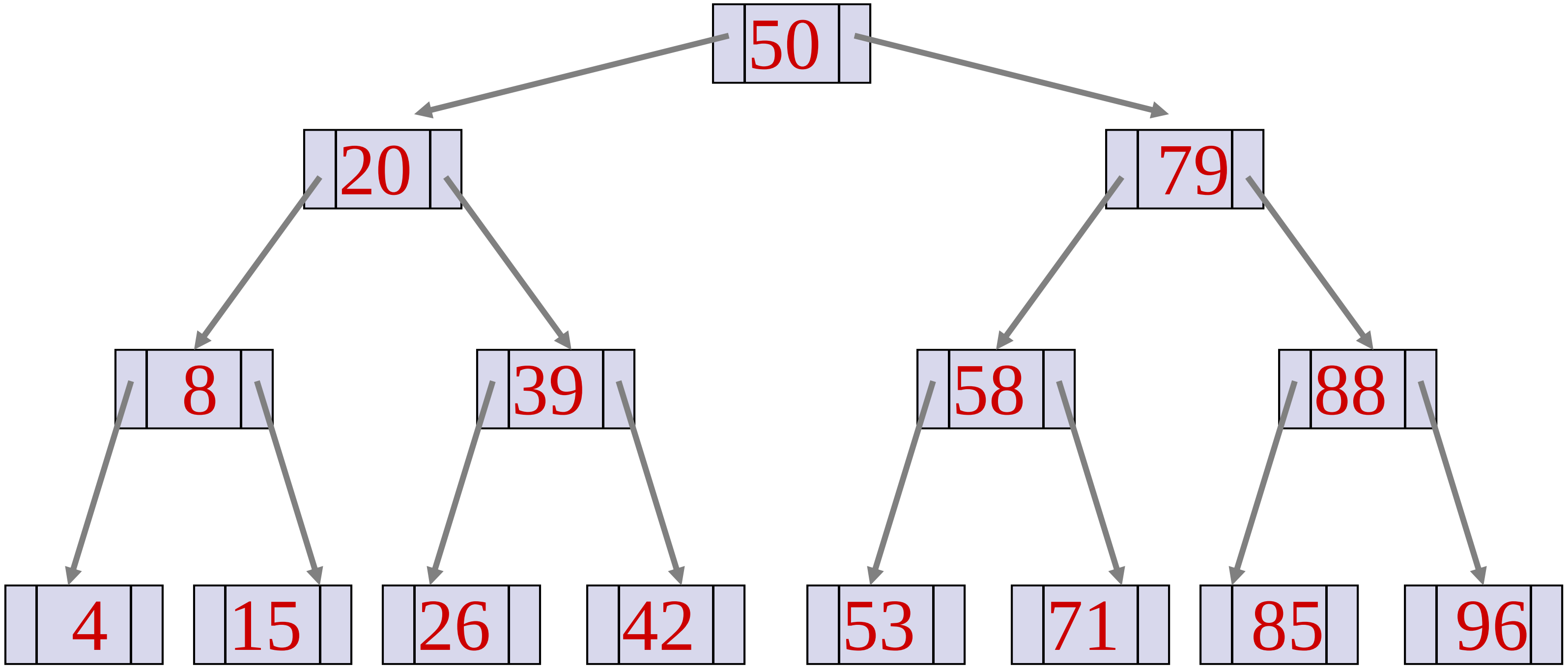


Travessia Em-Ordem

- Notação: In-fixada
- Visita o nó após passar a sua sub-árvore esquerda.
- Algoritmo:
  1. se árvore vazia; fim
  2. percorrer em-ordem a sub-árvore esquerda
  3. **imprime conteúdo do nó raiz**
  4. percorrer em-ordem a sub-árvore direita

Travessia Em-Ordem

- Exemplo: 4, 8, 15, 20, 26, 39, 42, 50, 53, 58, 71, 79, 85, 88, 96

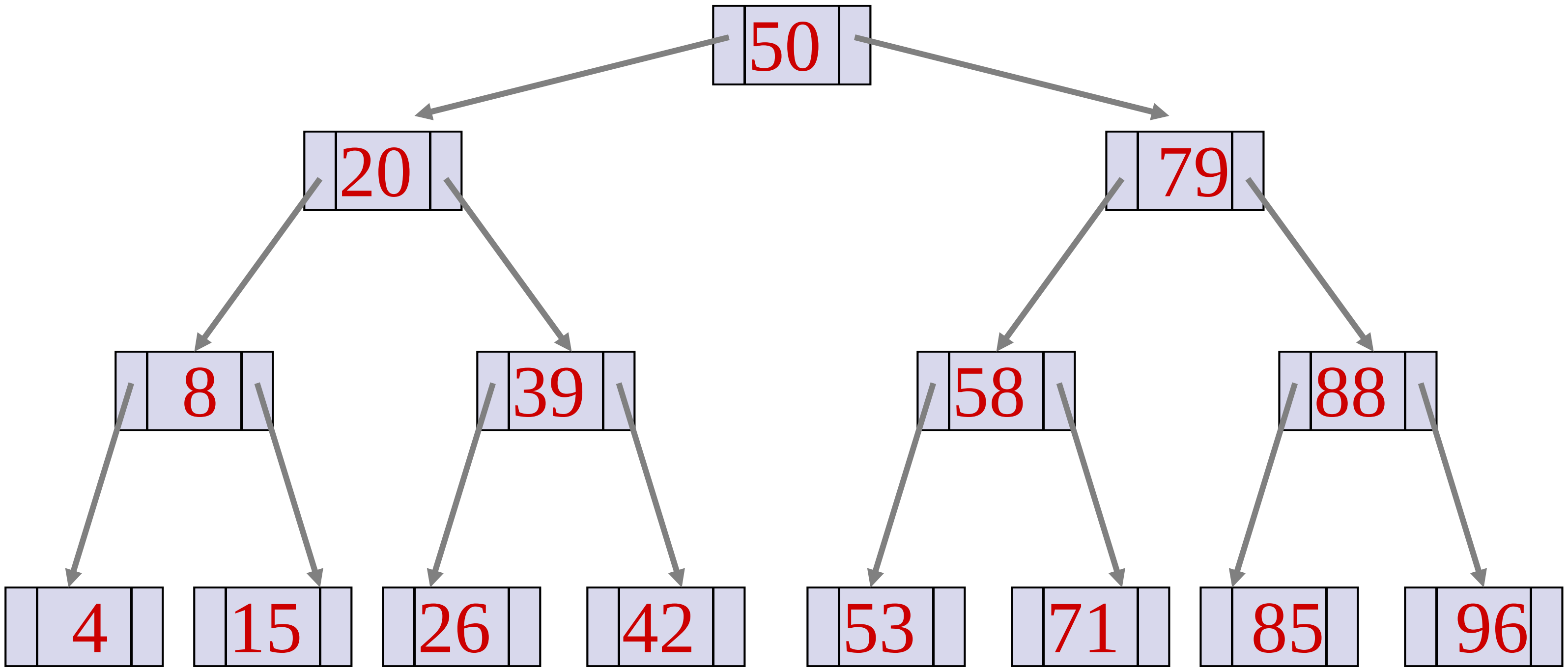


Travessia Pós-Ordem

- Notação: pós-fixada
- Visita o nó após passar a sua sub-árvore direita.
- Algoritmo:
  1. se árvore vazia; fim
  2. percorrer em pós-ordem a sub-árvore esquerda
  3. percorrer em pós-ordem a sub-árvore direita
  4. **imprime conteúdo do nó raiz**

Travessia Pós-Ordem

- Exemplo: 4, 15, 8, 26, 42, 39, 20, 53, 71, 58, 85, 96, 88, 79, 50



Exercícios

- Fazer exercícios do fim da Unidade 16 do livro virtual.

Árvore de Busca Binária

- Problema:
  - A árvore binária pode gerar para uma estrutura próxima a uma lista ligada, tanto no algoritmo de inserção quanto no algoritmo de remoção, e o tempo de acesso deixa de ser logarítmico;
- Solução:
  - Árvore AVL!!!

Inserção Não Recursiva

```
Inclui (raiz,  ch)
   atual = raiz, pai =null
   Enquanto(atual !=null) {
    pai = atual;
    Se (ch == chave(atual))
        retorne "Chave Já Existente"
     Se( ch< chave(atual))
        atual = esquerda(atual)
     Senão
        atual = direita(atual)
   }
   atual = novo No(ch)
   Se(pai == null)
      raiz = atual
   Senão
      Se(key < chave(pai))
         esquerda(pai) = atual
      Senão
         direita(pai) = atual
```

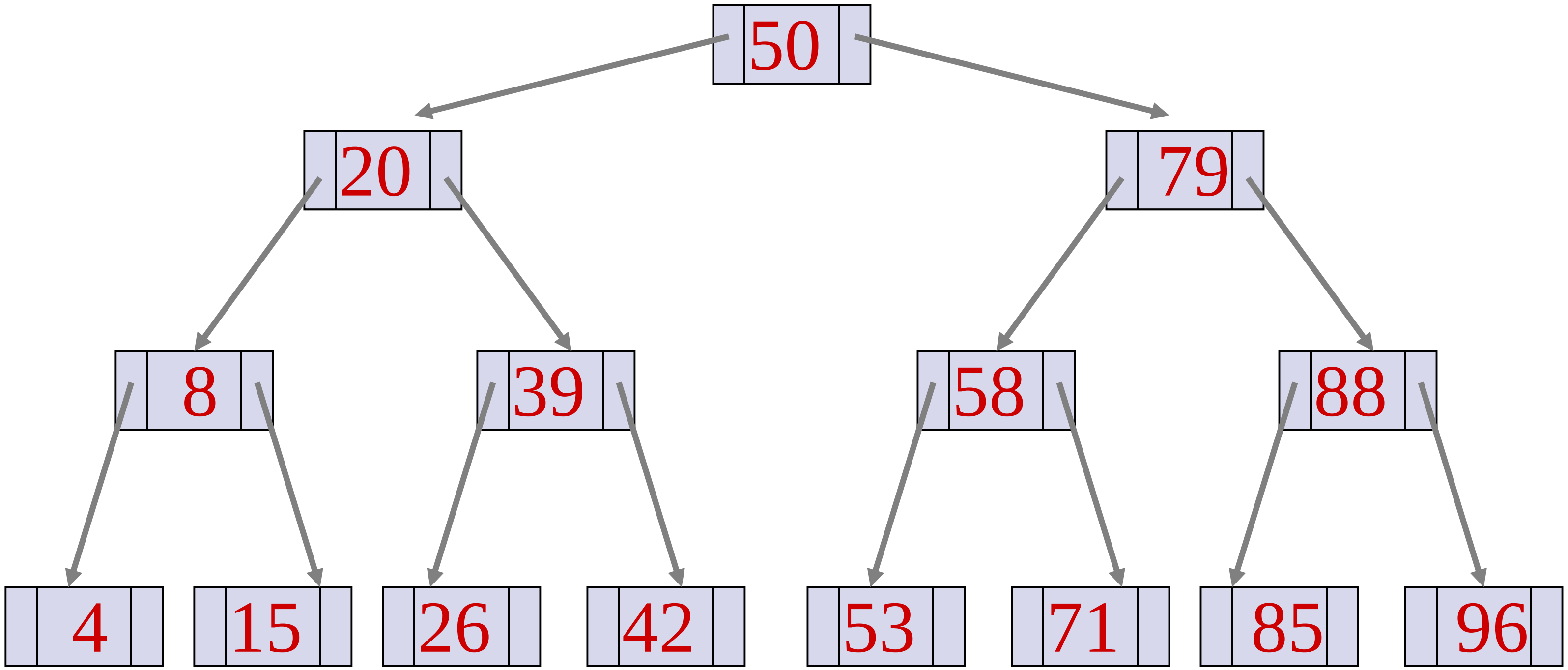
procura pela chave, se não achar insere um novo nó

Inserção Recursiva

```
Inclui (raiz,  ch)
   Se(raiz==NULL)
        raiz = novo no(ch)
   Senão se chave(raiz) == ch
         retorne "Chave Já Existente"
   Senão se chave(raiz) < ch então
         Se esquerda(raiz) != NULL então
             Inclui (esquerda(raiz), ch)
         Senão
             esquerda(raiz) = novo no(ch);
   Senão
         Se  direita(raiz) != NULL então
             Inclui (direita(raiz), valor)
         Senão
             direita(raiz) = novo no(valor);
```

Exemplo de inserção

- Exemplo: 50, 20, 39, 8, 79, 26, 58, 15, 88, 4, 85, 96, 71, 42, 53.



Remoção

- Situações:
  - Chave não existe:
    - Não remove;
  - Nó a ser eliminado não tem filhos:
    - Remoção sem ajustes na árvore
  - Nó tem somente uma sub-árvore esquerda ou direita:
    - Seu único filho é movido para cima para tomar o seu lugar
  - Se o nó a ser removido tem duas sub-árvores:
    - Escolha o menor elemento (o mais a esquerda) de sua sub-árvore direita e coloque-o no lugar.

Remoção

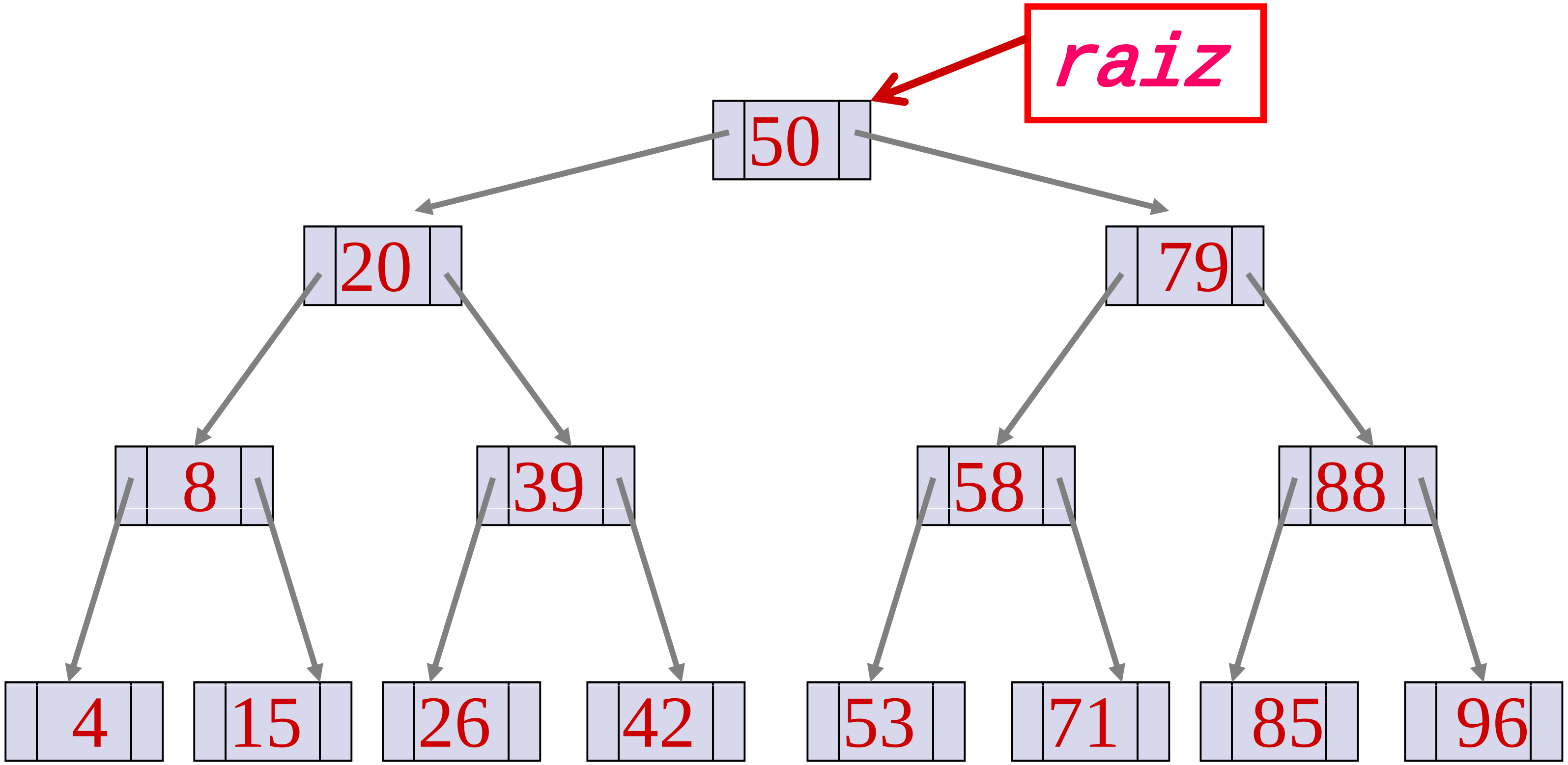
Procedimento Remove (*raiz*, *x*)

EndNo = Busca (*x*);

se *EndNo==NULL* então ***retorne insucesso***

Exemplo de Remoção

- Exemplo: quando a **chave não existe →** Remove (raiz, 60)



Remoção

Procedimento Remove (*raiz*, *x*)

EndNo = Busca (*x*);

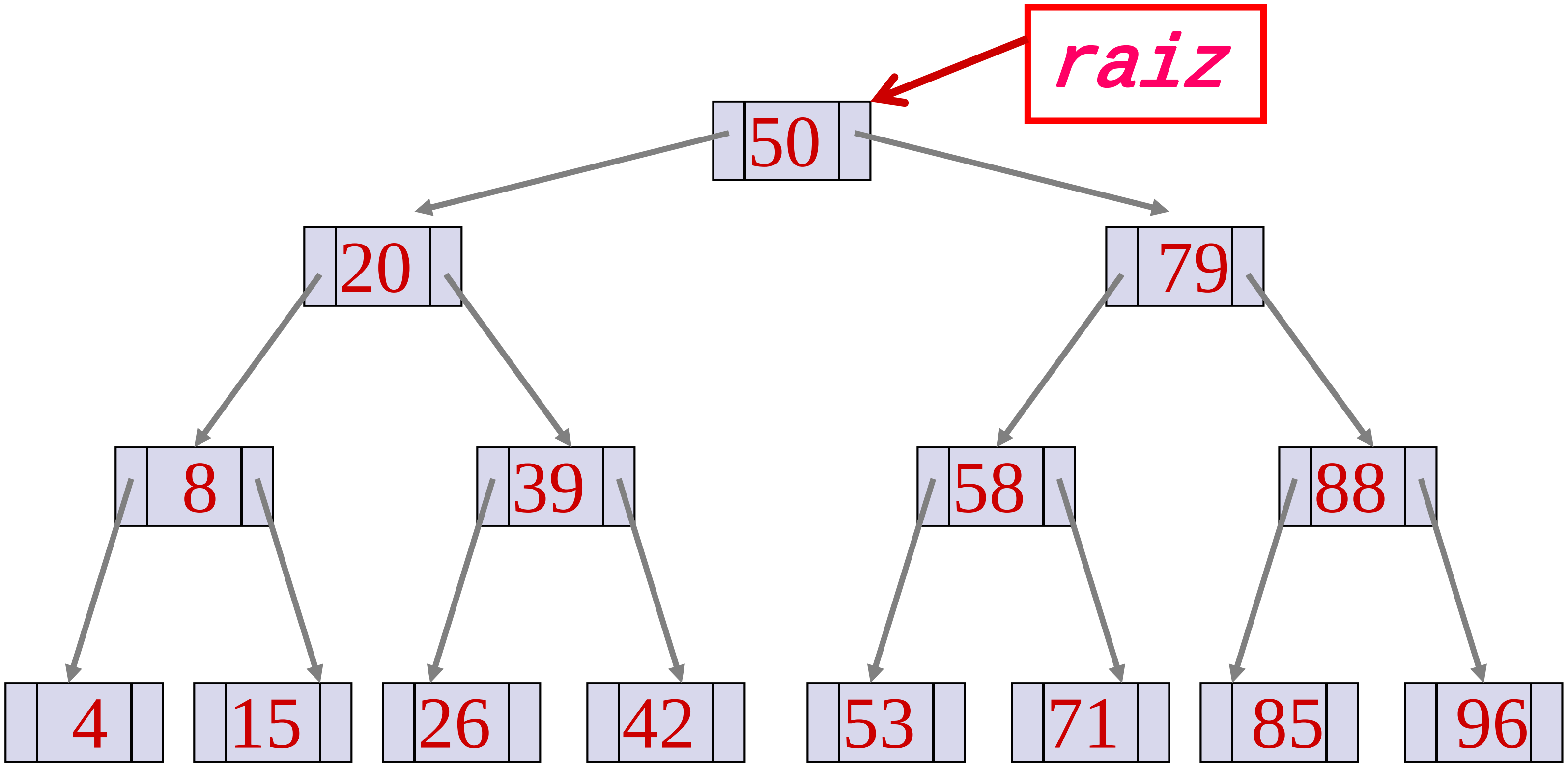
se *EndNo*==*NULL* então ***retorne insucesso***

senão se *EndNo* é folha

então apague apontador pai;

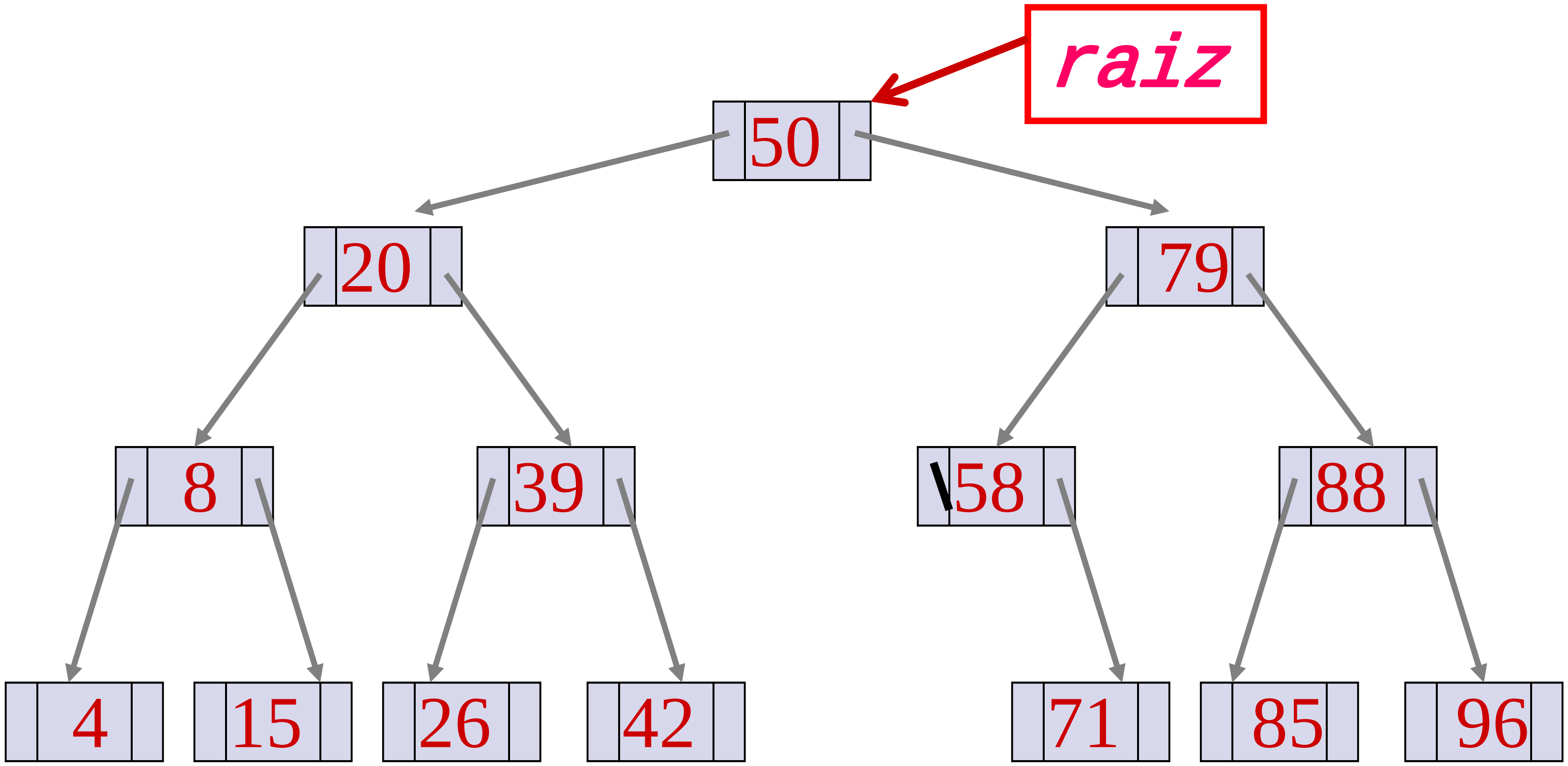
Exemplo de Remoção

- Exemplo: quando o **nó não tem sub-árvores: esquerda e direita** → Remove (raiz, 53)



Exemplo de Remoção

- Exemplo: quando o **nó não tem sub-árvores: esquerda e direita** → Remove (raiz, 53)



Remoção

Procedimento Remove (*raiz*, *x*)

EndNo $\leftarrow$ Busca (*x*);

se *EndNo=NULL* então ***retorne insucesso***

senão se *EndNo* é folha (*esq(EndNo)==NULL* AND *dir(EndNo)==NULL*)

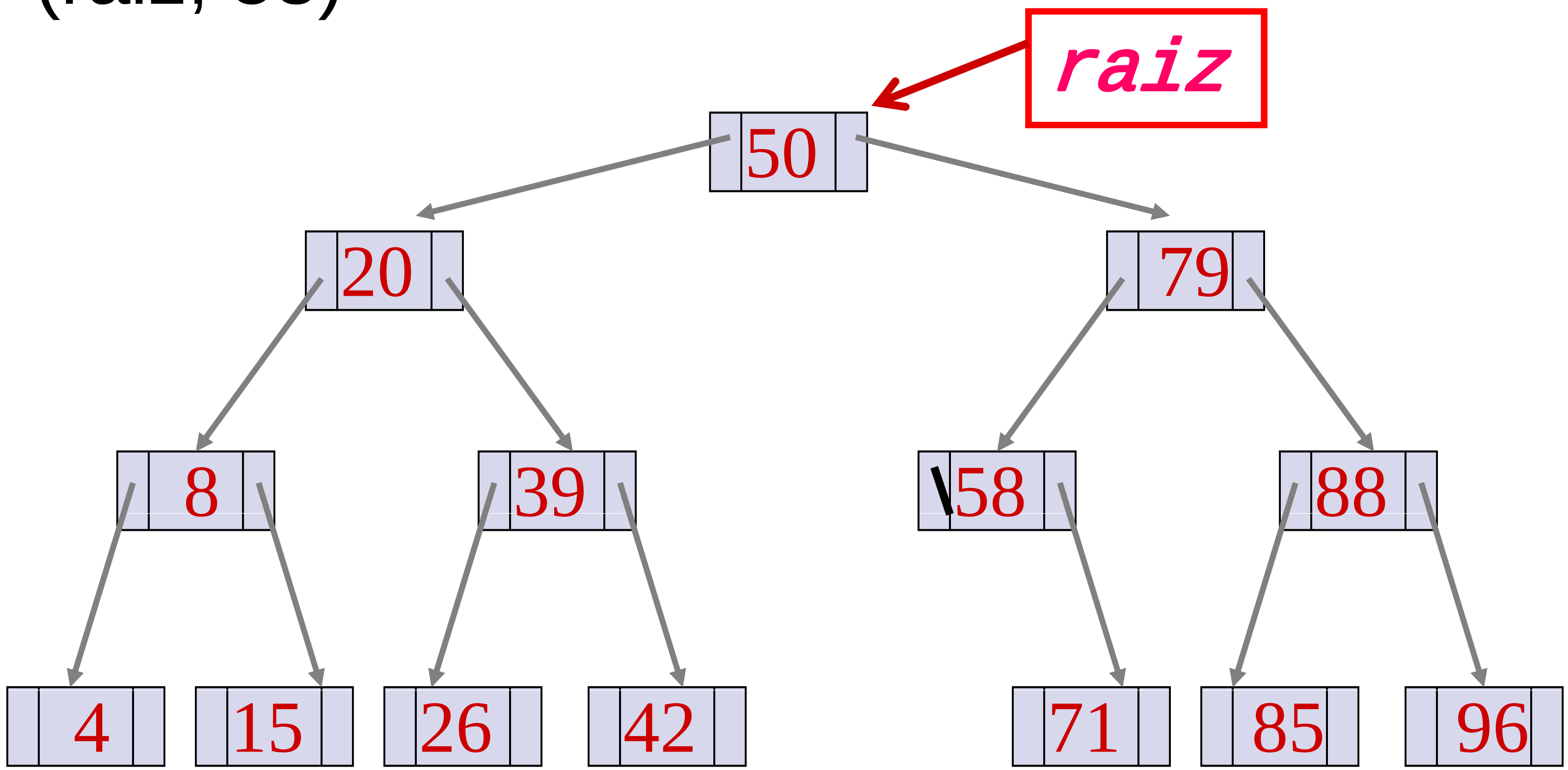
então apague apontador pai;

senão (*esq(EndNo)==NULL* OR *dir(EndNo)==NULL*)

faça pai de EndNo apontar para filho de EndNo

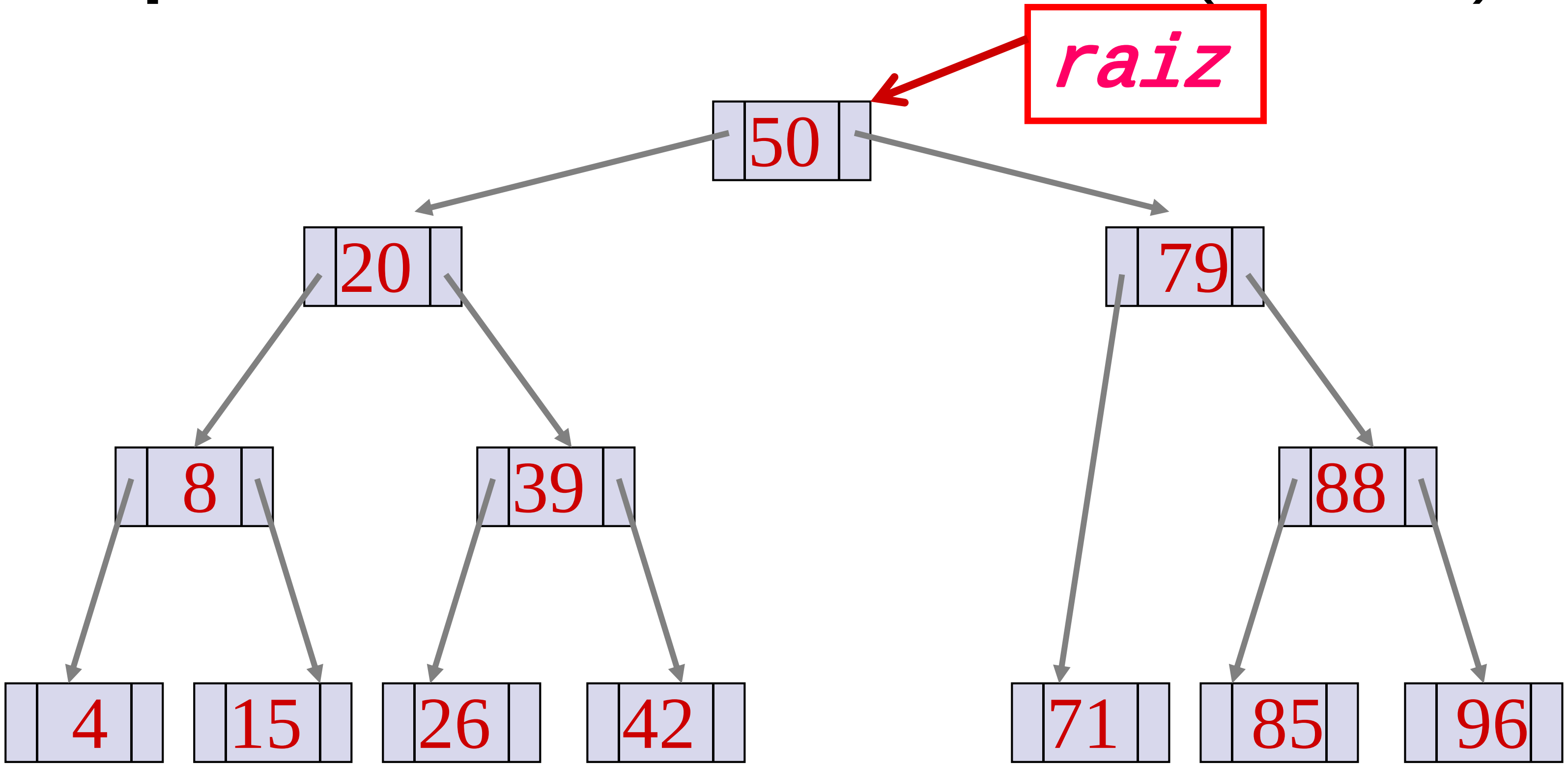
Exemplo de Remoção

- Exemplo: quando o **nó tem uma sub-árvore: esquerda ou direita** → Remove (raiz, 58)



Exemplo de Remoção

- Exemplo: quanto o **nó tem uma sub-árvore esquerda ou direita** → Remove (raiz, 58)



Remoção

Procedimento Remove (*raiz*, *x*)

EndNo ← Busca (*x*);

se *EndNo=NULL* então ***retorne insucesso***

senão se *EndNo* é folha (*esq(EndNo)==NULL* AND *dir(EndNo)==NULL*)

então apague apontador pai;

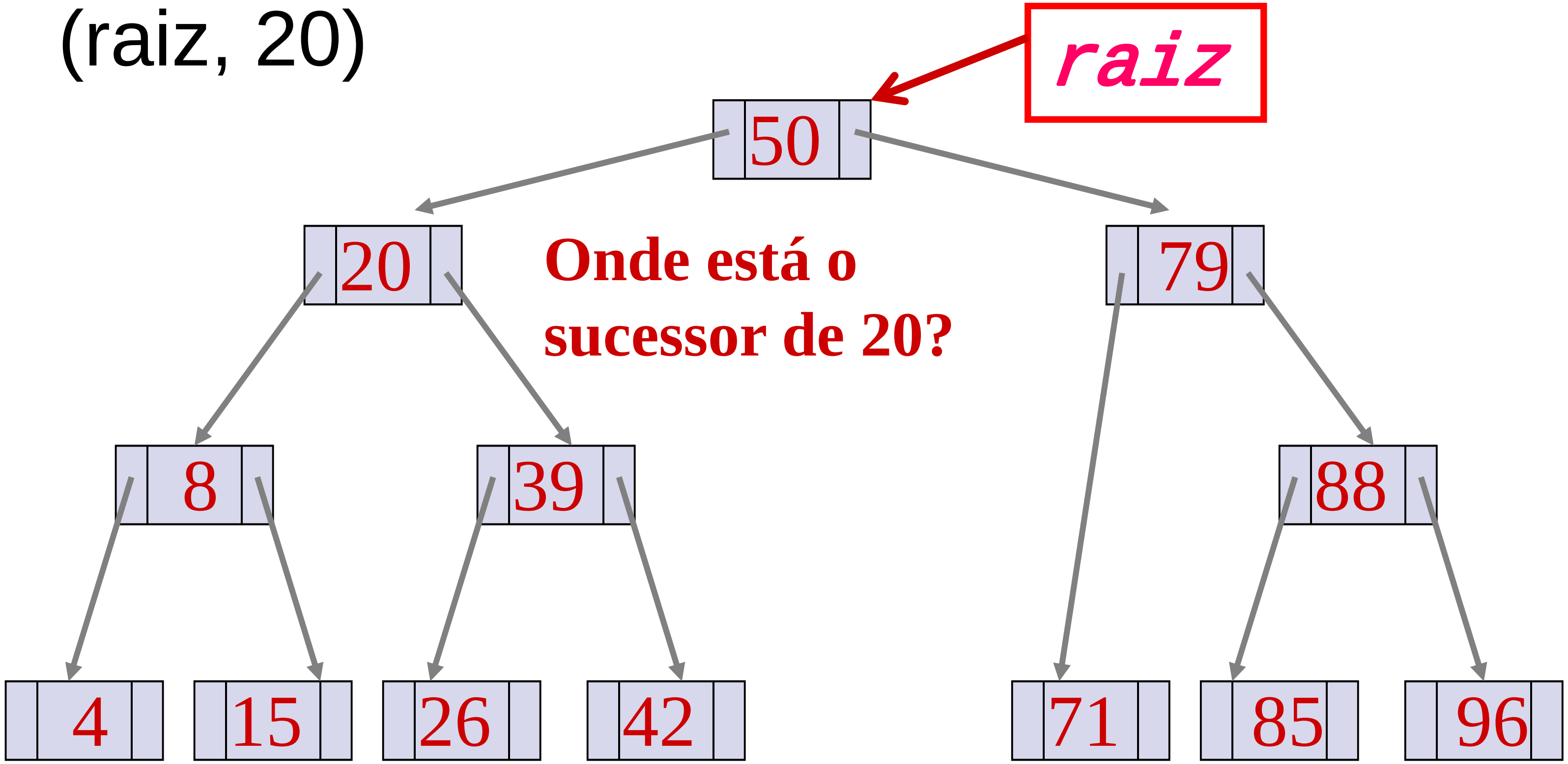
senão (*esq(EndNo)==NULL* OR *dir(EndNo)==NULL*)

faça pai de EndNo apontar para filho de EndNo

senão ***substitua EndNo pelo seu sucessor***

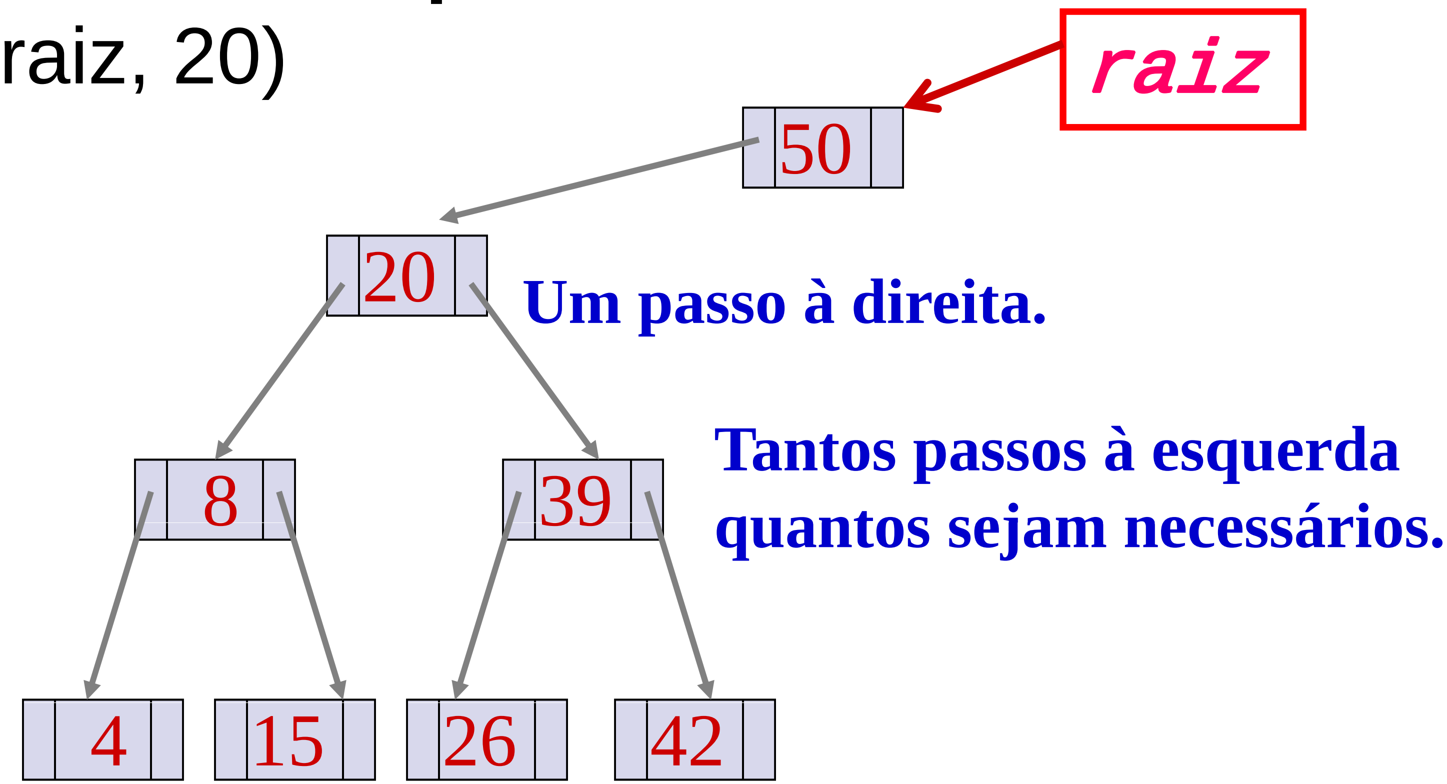
Exemplo de Remoção

- Exemplo: quando o **nó tem duas sub-árvores: esquerda e direita** → Remove (raiz, 20)



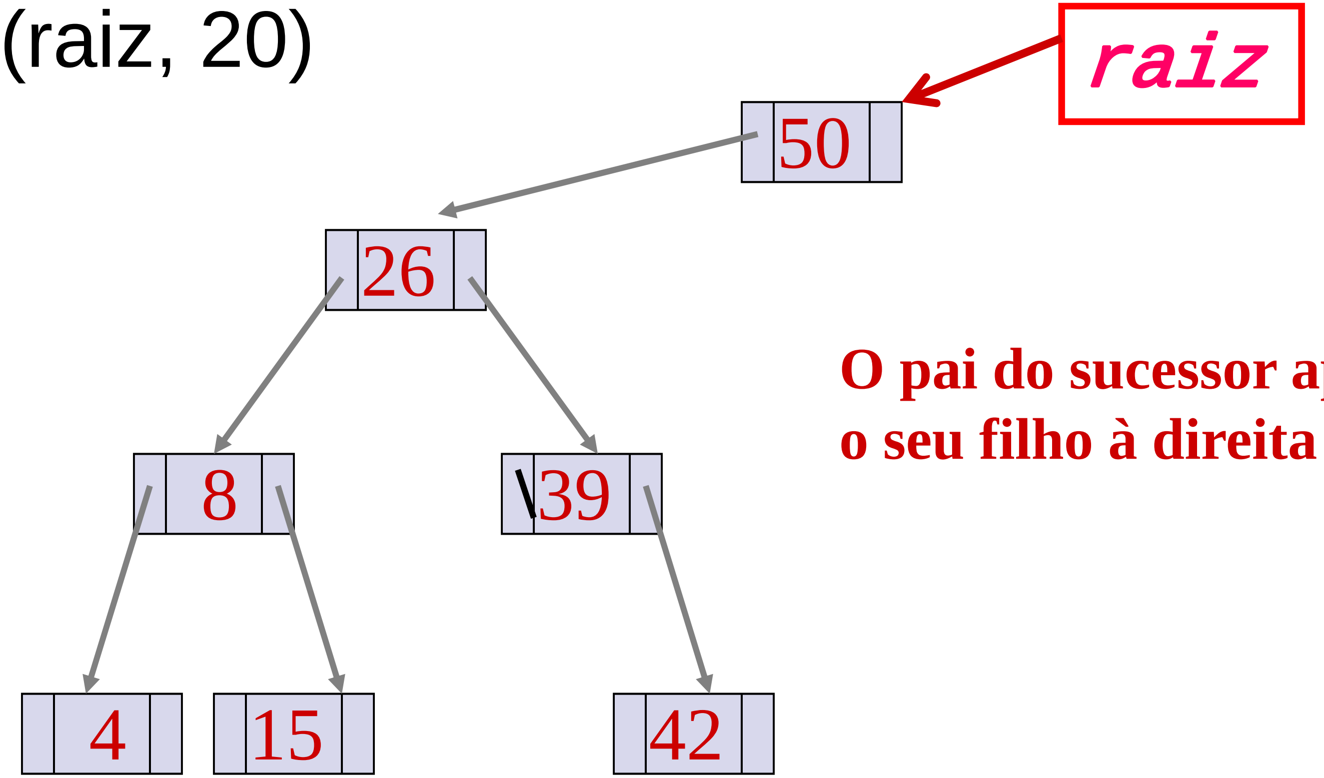
Exemplo de Remoção

- Exemplo: quando o **nó tem duas sub-árvores: esquerda e direita** → Remove (raiz, 20)



Exemplo de Remoção

- Exemplo: quando o **nó tem duas sub-árvores: esquerda e direita** → Remove (raiz, 20)



O pai do sucessor apontará para o seu filho à direita do sucessor.