Exclusão de dados em árvores binárias de busca

FURB
UNIVERSIDADE DE BLUMENAU

Operação de exclusão

- Consiste em duas etapas:

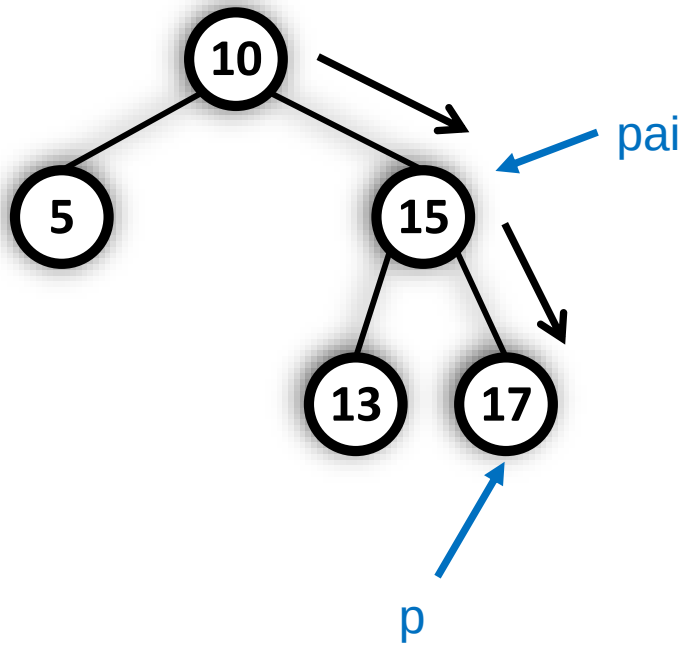
  1) Utilizar um algoritmo para localizar o nó que contém o dado a ser removido

  2) Ao encontra o nó, haverá três casos para considerar:

    - Caso 1 - O nó a ser removido é uma folha
    - Caso 2 - O nó a ser removido tem apenas um filho
    - Caso 3 - O nó a ser removido tem dois filhos

Algoritmo para localizar o nó a ser removido

Exemplo: localizar o nó que contém 17 para remover



filhoEsquerda = falso

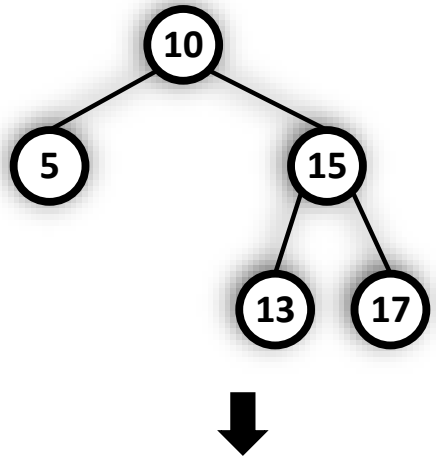
Algoritmo: **retirar(int valor)**

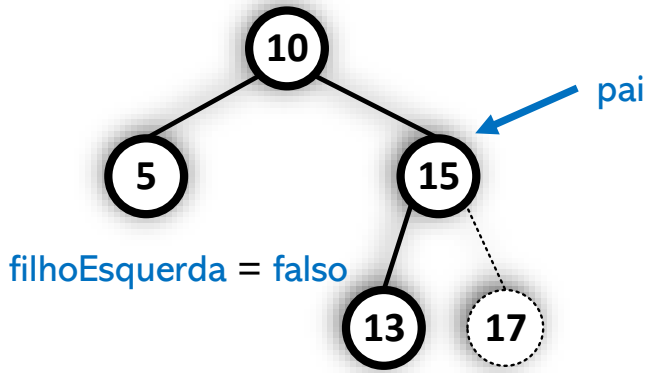
```
NoArvoreBinaria p ← raiz;
NoArvoreBinaria pai ← null;

// localizar o nó a ser removido
enquanto (p ≠null) e (p.info ≠ valor) faça
    pai ← p;
    se (valor < p.info) então
        filhoEsquerda ← verdadeiro;
        p ← p.esquerda;
    senão
        filhoEsquerda ←  falso;
        p ←  p.direita;
    fim-se;
fim-enquanto;
...
```

Caso 1 – Remover uma folha

- Ao identificar que o nó a ser removido é uma folha, basta remover a sua ligação com o nó pai

- Exemplo:
  - Remover nó que tem chave “17”





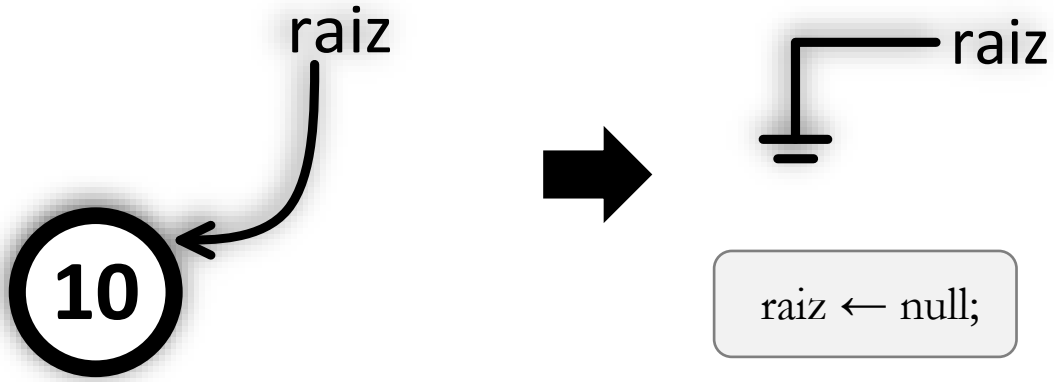
pai.direita ← null;

Quando o nó removido está à direita:
pai.direita ← null;

Quando o nó removido está à esquerda:
pai.esquerda ← null;

Caso 1 – Remover uma folha

- Caso especial
  - A folha a ser removida é a raiz da árvore



Algoritmo – Remover uma folha

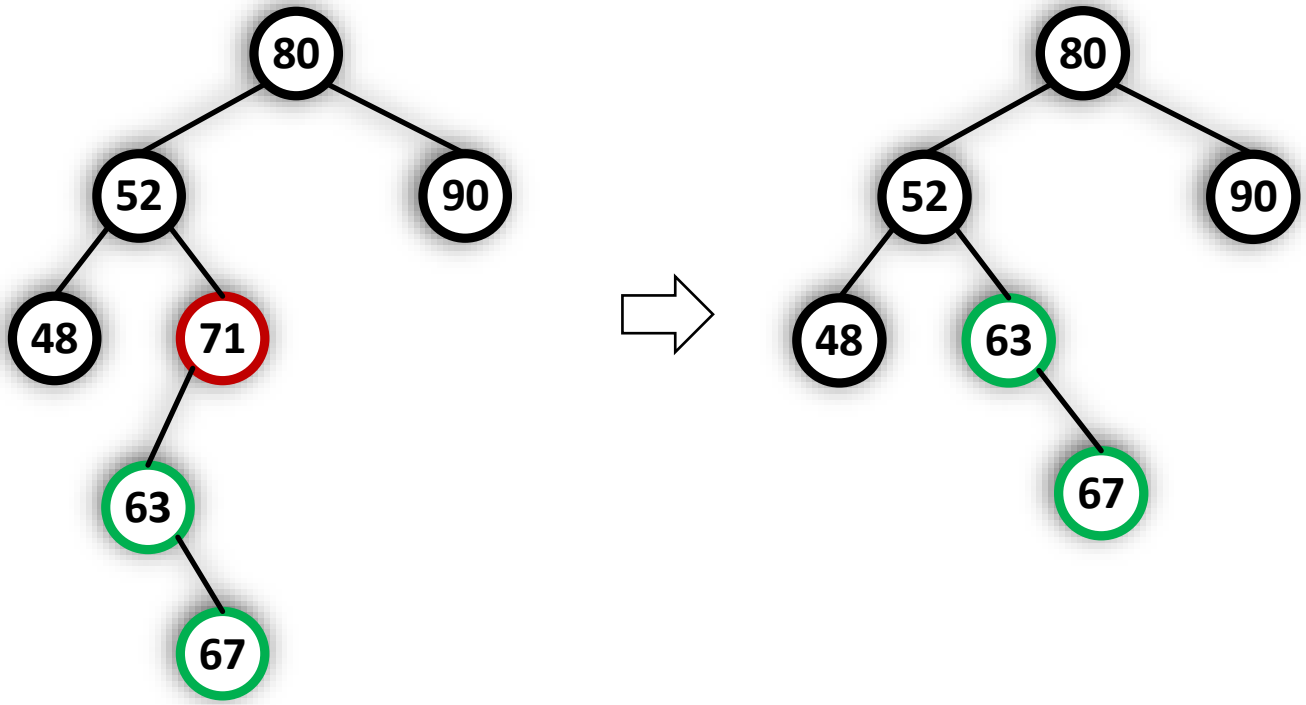
Identifica que está removendo uma folha

Continuação algoritmo**: retirar(int valor)**

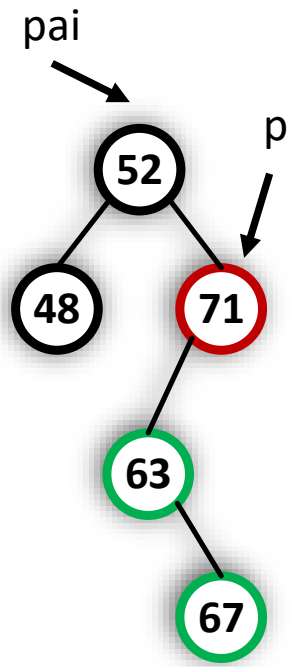
```
se (p ≠ null) então
   se (p.esquerda = null) e (p.direita = null) então
      se (p = raiz) então
         raiz ← null;
      senão
         se (filhoEsquerda = verdadeiro) então
            pai.esquerda ← null;
         senão
            pai.direita ← null;
         fim-se;
      fim-se;
   ... continua
fim-se;
```

Caso 2 – Remover um nó com um filho

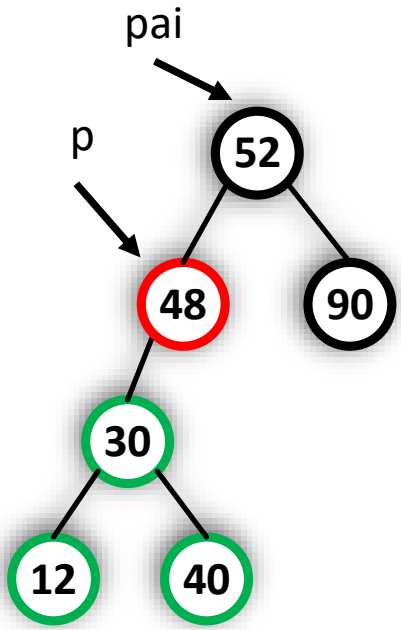
- Exemplo: remover chave “71”



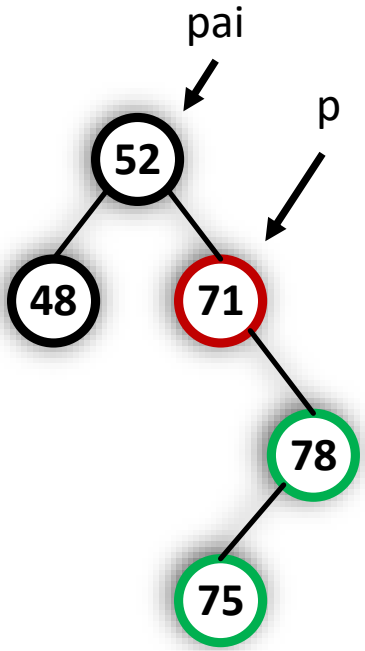
Caso 2 – Remover um nó com um filho



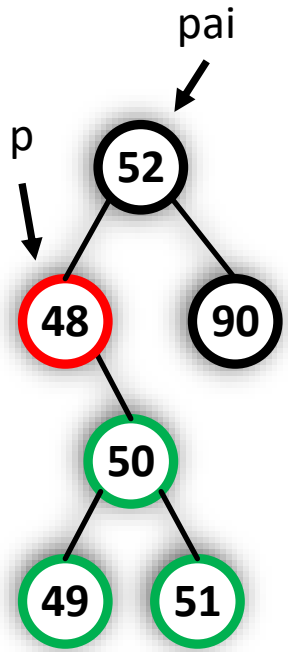
pai.direita ← p.esquerda



pai.esquerda ← p.esquerda



pai.direita ← p.direita



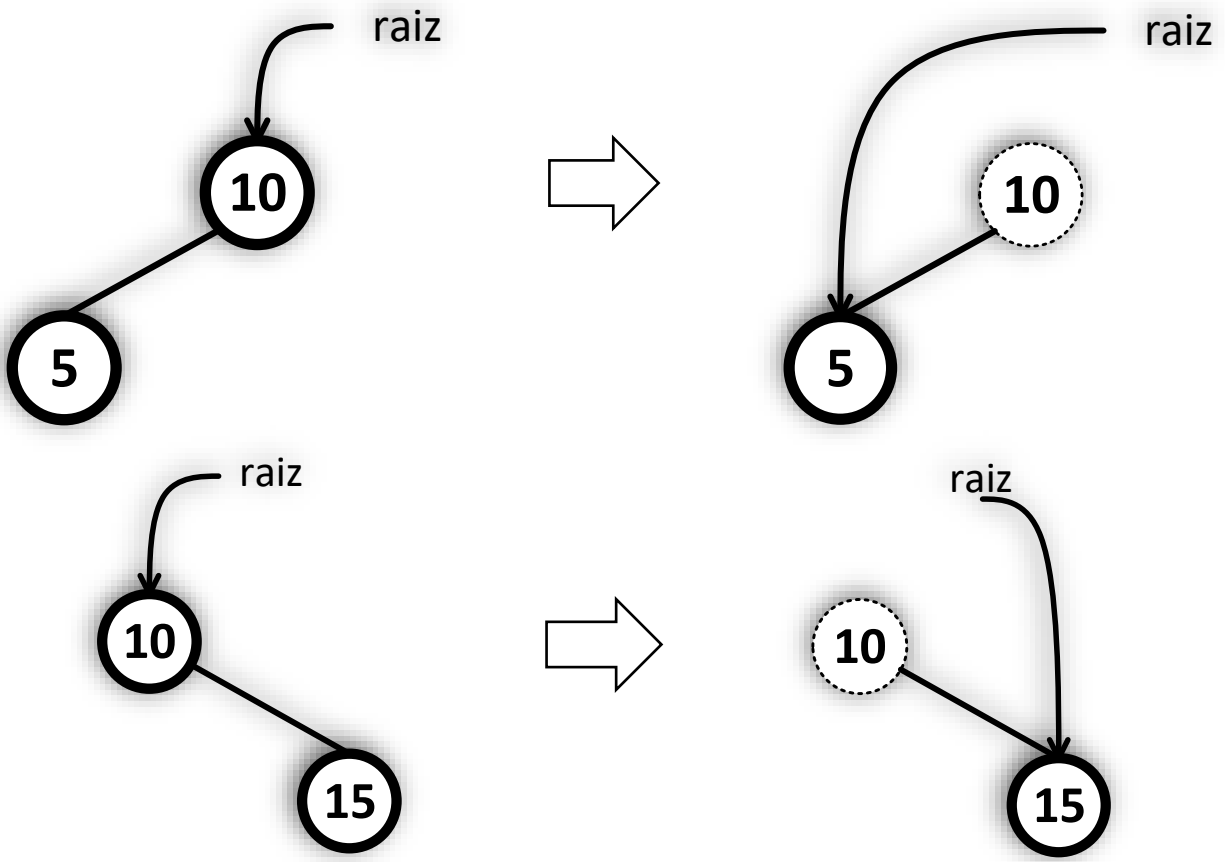
pai.esquerda ← p.direita

Nó a ser removido tem filho à esquerda (`p.direita = null`)

Nó a ser removido tem filho à direita (`p.esquerda = null`)

Caso 2 – Remover um nó com um filho

- **Caso especial**: O nó a ser removido é a raiz da árvore



Continuação Algoritmo: **retirar(int valor)**

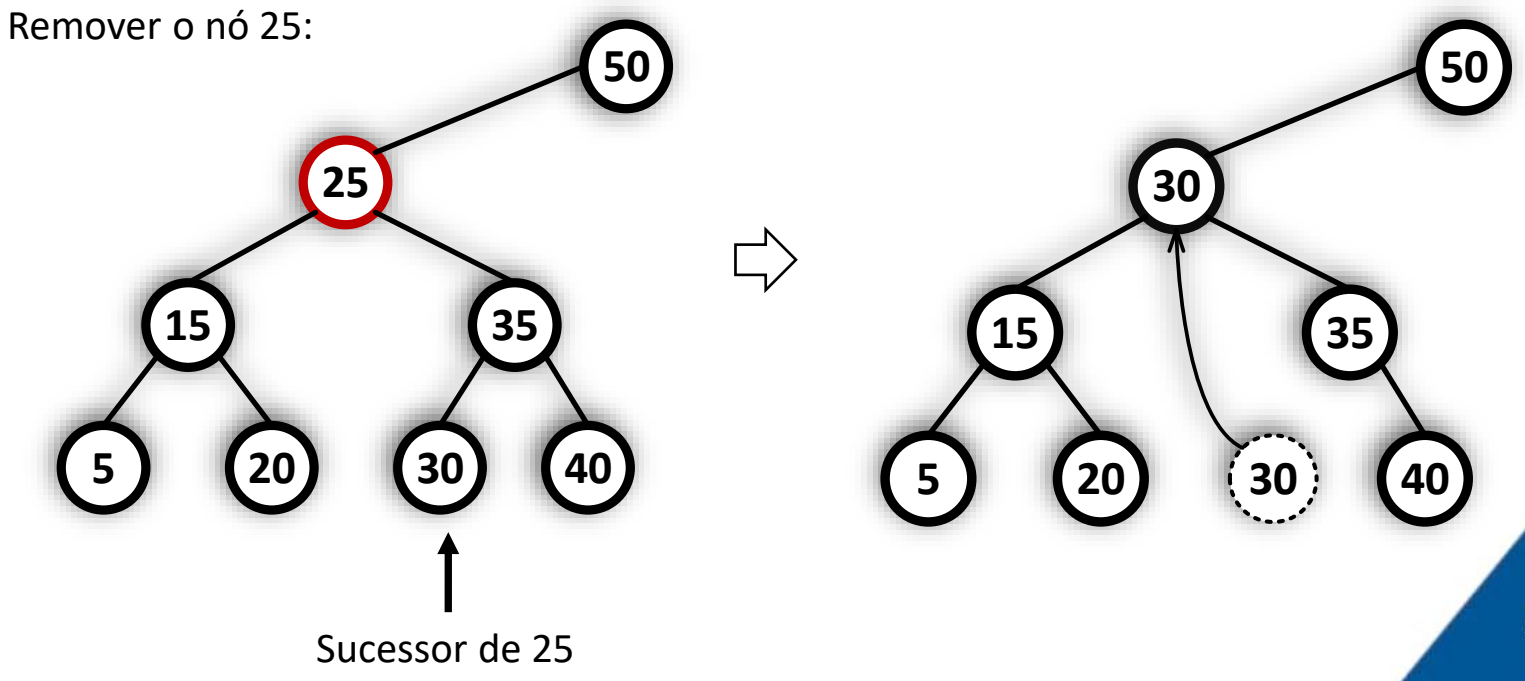
```
...
Senão
  se (p.direita = null) então
    se (p = raiz) então
      raiz ← p.esquerda;
    senão
      se (filhoEsquerda = verdadeiro) então
        pai.esquerda ← p.esquerda;
      senão
        pai.direita ← p.esquerda;
      fim-se
    fim-se;
  senão
    se (p.esquerda = null) então
      se (p = raiz) então
        raiz ← p.direita;
      senão
        se (filhoEsquerda = verdadeiro) então
          pai.esquerda ← p.direita;
        senão
          pai.direita ← p.direita;
        fim-se
      fim-se
    senão
      ... continua
    fim-se
  fim-se
fim-se
```

Algoritmo

Algoritmo para remoção, quando for o Caso 2

Caso 3 – Remover nó com dois filhos

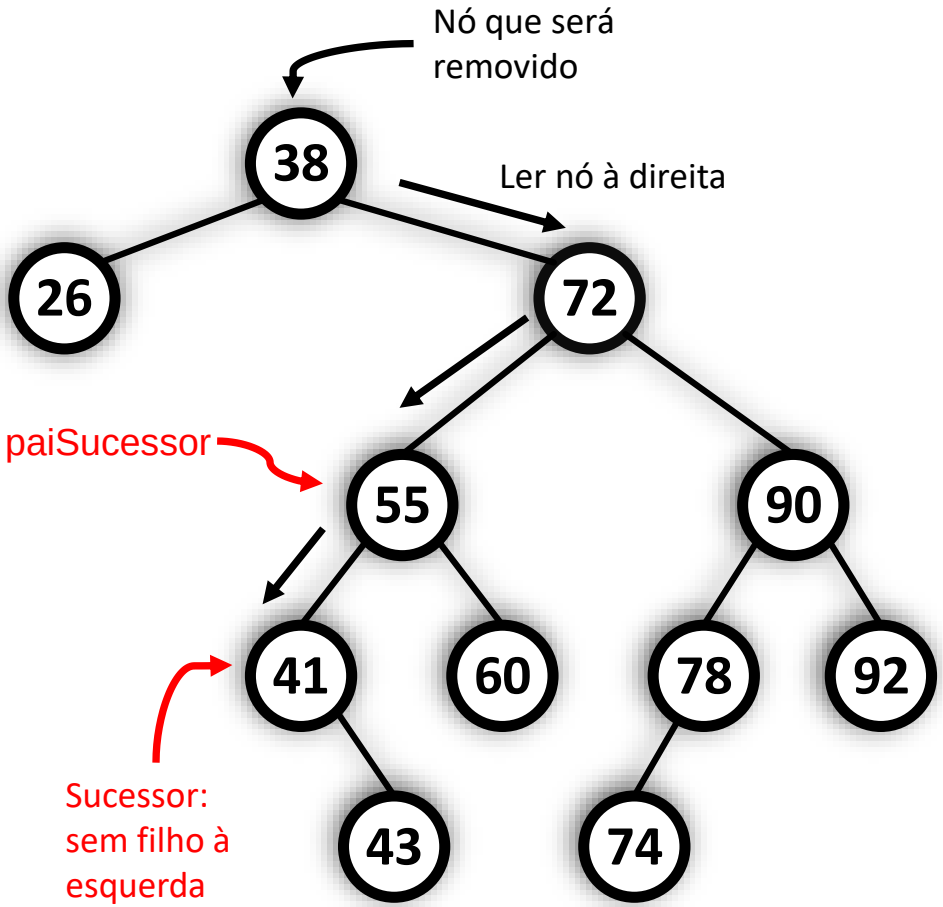
- Deve-se executar as seguintes etapas:
  - Localizar o próximo nó do nó a ser removido (denominado de “nó sucessor”)
  - O nó sucessor deve tomar o lugar do nó a ser removido



Caso 3 – Remover nó com dois filhos

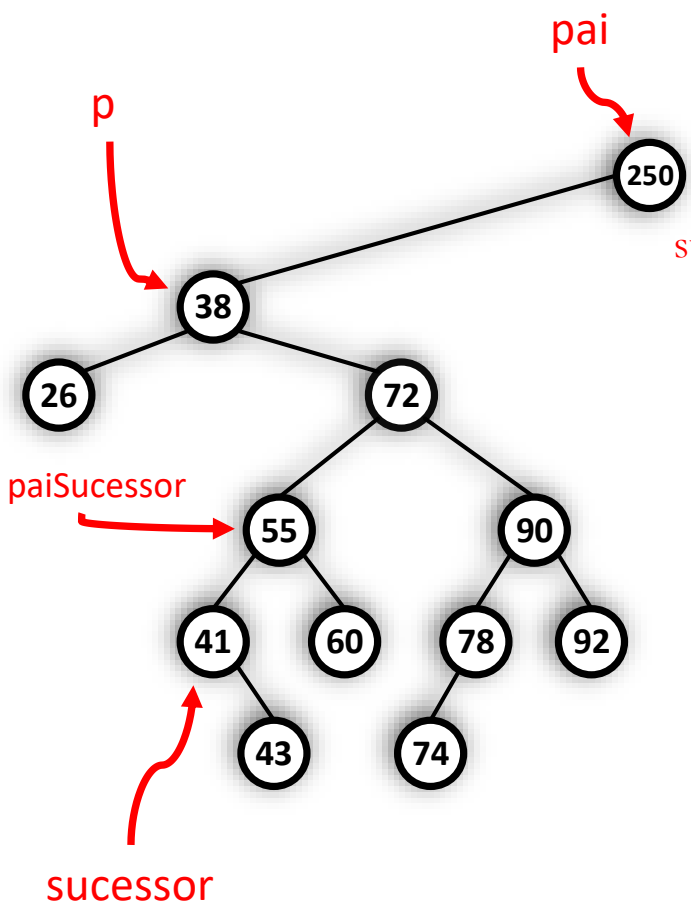
Localizando sucessor

- Para localizar o nó sucessor, deve-se:
  1) Acessar o primeiro nó à direita e
  2) Caminhar até encontrar o último nó a esquerda



Exemplo

Remover nó 38

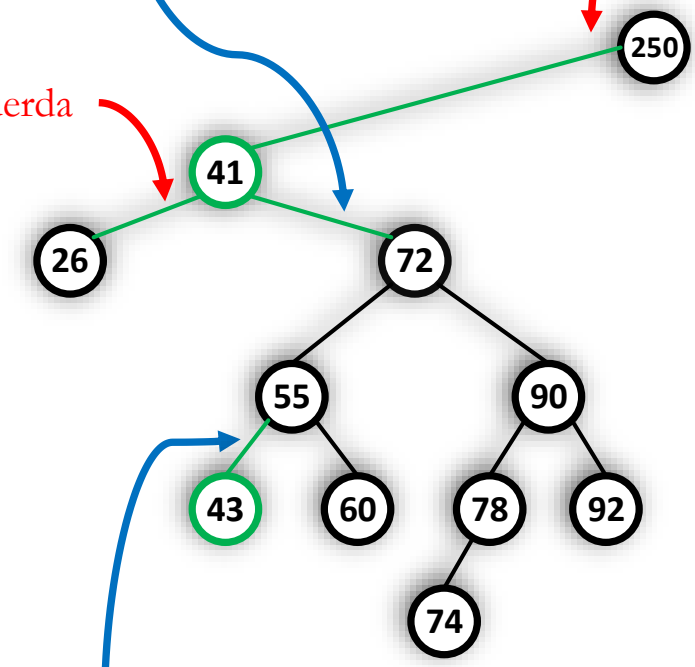


Poderia ser :
pai.direita ← sucessor

pai.esquerda ← sucessor

sucessor.direita ← p.direita

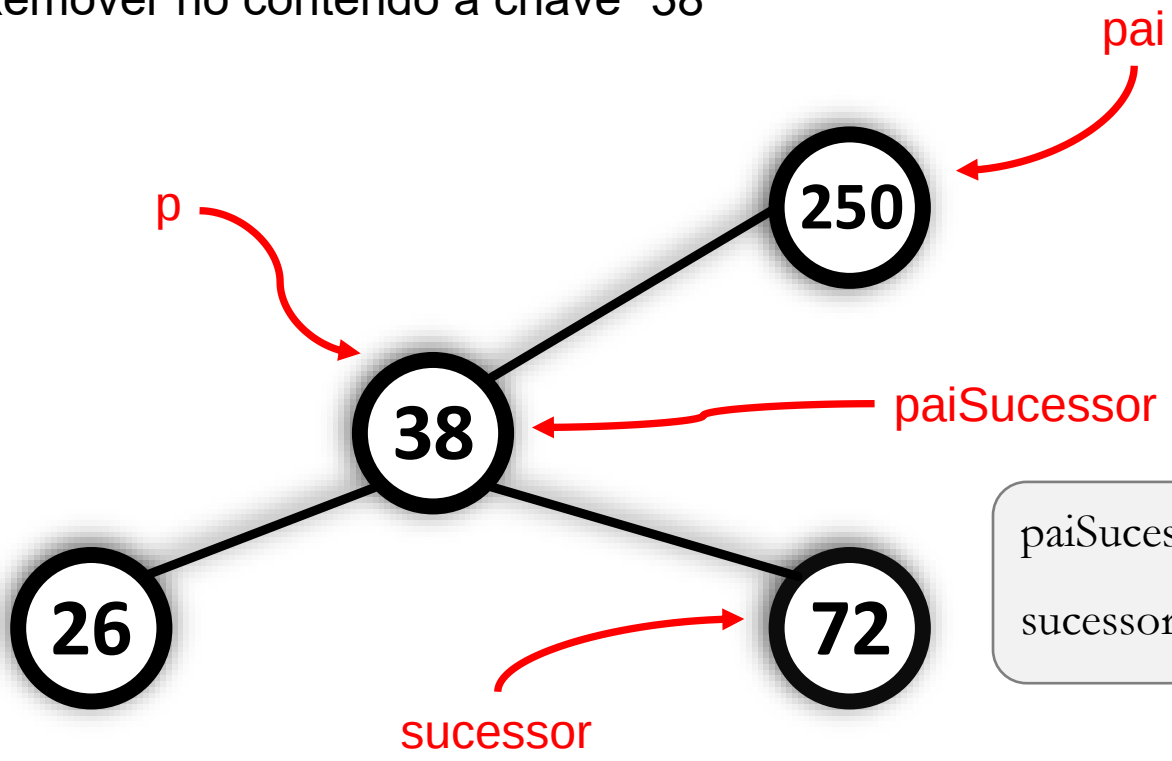
sucessor.esquerda ← p.esquerda



paiSucessor.esquerda ← sucessor.direita;

Caso 3 – Caso especial

Remover nó contendo a chave “38”



paiSucessor.esquerda ← sucessor.direita;

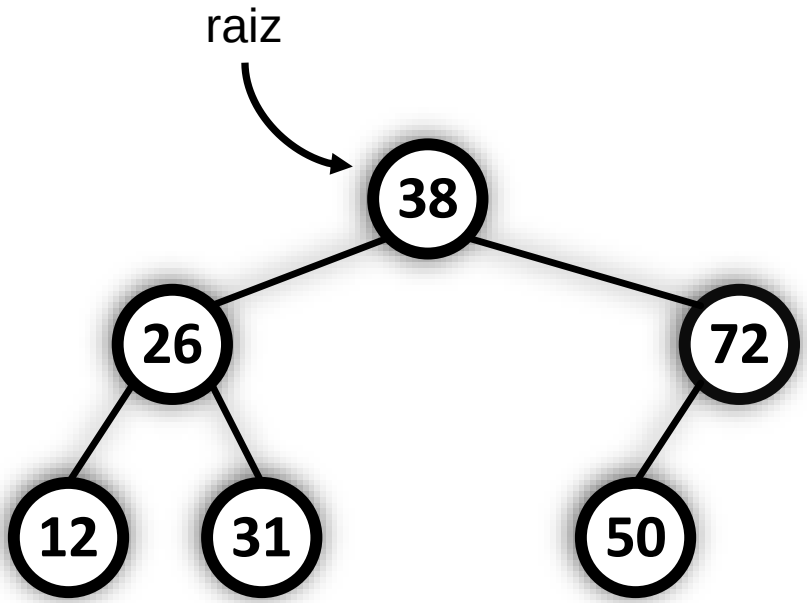
sucessor.direita ← p.direita

Não podemos executar

Caso especial: sucessor é filho à direita do nó a ser excluído

Caso 3 – Caso especial

- O nó a ser removido é a raiz:



Não pode executar:

pai.esquerda ← sucessor

Deve atualizar variável raiz:

raiz ← sucessor

Caso 3 – Algoritmo para remoção

Continuação Algoritmo**: remover(int valor)**

```
...
Senão // nó com dois filhos
   NoArvoreBinaria sucessor ← extrairSucessor(p);

   se (p = raiz) então
      raiz ← sucessor;
   senão
      se (filhoEsquerda = verdadeiro) então
         pai.esquerda = sucessor;
      senão
         pai.direita = sucessor;
      fim-se;
   fim-se;

   sucessor.esquerda ← p.esquerda;
fim-se;
```

Algoritmo**: extrairSucessor(NoArvoreBinaria p)**

```
NoArvoreBinaria atual ← p.direita;
NoArvoreBinaria paiSucessor ← p;
NoArvoreBinaria sucessor ← p;

enquanto (atual ≠ null) faça
   paiSucessor ← sucessor;
   sucessor ← atual;
   atual ← atual.esquerda;
fim-enquanto;

se (sucessor ≠ p.direita) então
   paiSucessor.esquerda ← sucessor.direita;
   sucessor.direita ← p.direita;
fim-se;

retornar sucessor;
```