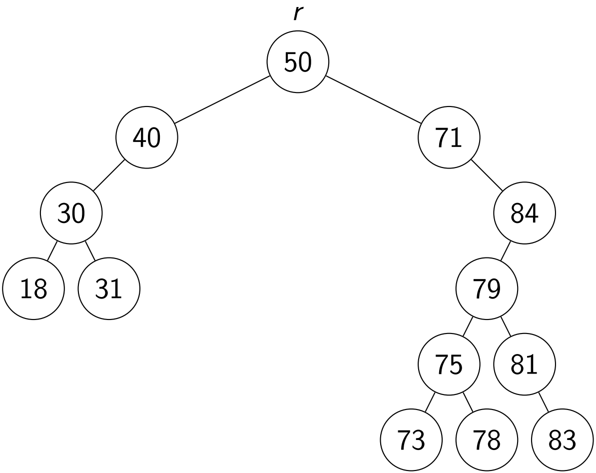
Estruturas de Dados - Árvore Binária de Busca 03

Árvore binária de busca: remoção

- Exemplo: remover um nó desta BST



Árvore binária de busca: remoção

- Remoção de um nó v de uma BST:
  - se v for uma folha (duas sub-árvores vazias), basta removermos v
    - caso trivial de se resolver!
  - se v tiver apenas uma sub-árvore não vazia, sua sub-árvore deve se tornar sub-árvore do pai de v após a remoção
    - caso simples de se resolver!
  - se v tiver duas sub-árvores não vazias, devemos selecionar o sucessor de v e torná-lo a nova raiz, tomando a posição de v
    - caso mais complexo de se resolver!

Árvore binária de busca: remoção

- Remoção de um nó v de uma BST:
  - se v for uma folha (duas sub-árvores vazias), basta removermos v
    - caso trivial de se resolver!
  - se v tiver apenas uma sub-árvore não vazia, sua sub-árvore deve se tornar sub-árvore do pai de v após a remoção
    - caso simples de se resolver!
  - se v tiver duas sub-árvores não vazias, devemos selecionar o sucessor de v e torná-lo a nova raiz, tomando a posição de v
    - caso mais complexo de se resolver!
- Note que a ideia de substituir uma sub-árvore pela outra se repete
- Ideia inicial: criar uma função que substitui, na BST, uma sub-árvore de raiz v por uma sub-árvore de raiz u

Árvore binária de busca: substituir

Algoritmo: Substitui(r, v, u)

Entrada: nós r (raiz), v e u da BST

Saída: raiz da BST completa (com a sub-árvore de raiz v substituída pela sub-árvore de raiz u)

se $v \rightarrow pai == \lambda$ **então**
    $r = u$
senão se $v == v \rightarrow pai \rightarrow esq$ **então**
    $v \rightarrow pai \rightarrow esq = u$
senão
    $v \rightarrow pai \rightarrow dir = u$
se $u \neq \lambda$ **então**
    $u \rightarrow pai = v \rightarrow pai$
retorne r

Complexidade: $O(1)$

Árvore binária de busca: remover

Algoritmo: RemoverBST(r, x)

Entrada: nó raiz r da BST, elemento x a ser removido

Saída: raiz da BST completa

```
v = BuscaBST(r, x)
se v ≠ λ então
    se v→esq == λ então
        r = Substitui(r, v, v→dir)
    senão se v→dir == λ então
        r = Substitui(r, v, v→esq)
    senão
        s = MinimoBST(v→dir) //ou s = SucessorBST(v)
        se s→pai ≠ v então
            r = Substitui(r, s, s→dir)
            s→dir = v→dir
            s→dir→pai = s
        r = Substitui(r, v, s)
        s→esq = v→esq
        s→esq→pai = s
    Desaloca(v)
senão
    print("Elemento não encontrado!")
retorne r
```

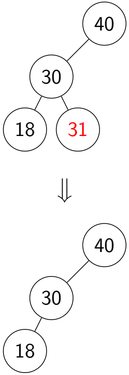
Complexidade: proporcional à altura da BST - $O(h)$

Árvore binária de busca: remover

Algoritmo: RemoverBST(r, x)
Entrada: nó raiz r da BST, elemento x a ser removido
Saída: raiz da BST completa

v = BuscaBST(r, x)
se $v \neq \lambda$ **então**
    se $v \rightarrow esq == \lambda$ **então**
        r = Substitui$(r, v, v \rightarrow dir)$
    senão se $v \rightarrow dir == \lambda$ **então**
        r = Substitui$(r, v, v \rightarrow esq)$
    senão
        s = MinimoBST$(v \rightarrow dir)$ //ou s = SucessorBST(v)
        se $s \rightarrow pai \neq v$ **então**
            r = Substitui$(r, s, s \rightarrow dir)$
            $s \rightarrow dir = v \rightarrow dir$
            $s \rightarrow dir \rightarrow pai = s$
        r = Substitui(r, v, s)
        $s \rightarrow esq = v \rightarrow esq$
        $s \rightarrow esq \rightarrow pai = s$
    Desaloca(v)
senão
    print("Elemento não encontrado!")
retorne r

Caso 1: $v \rightarrow esq = \lambda$ e $v \rightarrow dir = \lambda$
Exemplo: remover 31



Árvore binária de busca: remover

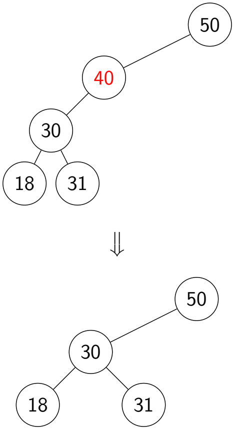
Algoritmo: RemoverBST(r, x)

Entrada: nó raiz r da BST, elemento x a ser removido
Saída: raiz da BST completa

$v = \text{BuscaBST}(r, x)$
se $v \neq \lambda$ **então**
    se $v \rightarrow esq == \lambda$ **então**
        $r = \text{Substitui}(r, v, v \rightarrow dir)$
    senão se $v \rightarrow dir == \lambda$ **então**
        $r = \text{Substitui}(r, v, v \rightarrow esq)$
    senão
        $s = \text{MinimoBST}(v \rightarrow dir)$ //ou $s = \text{SucessorBST}(v)$
        se $s \rightarrow pai \neq v$ **então**
            $r = \text{Substitui}(r, s, s \rightarrow dir)$
            $s \rightarrow dir = v \rightarrow dir$
            $s \rightarrow dir \rightarrow pai = s$
        $r = \text{Substitui}(r, v, s)$
        $s \rightarrow esq = v \rightarrow esq$
        $s \rightarrow esq \rightarrow pai = s$
    Desaloca(v)
senão
    print("Elemento não encontrado!")
retorne r

Complexidade: proporcional à altura da BST - $O(h)$

Caso 2: $v \rightarrow esq \neq \lambda$ e $v \rightarrow dir = \lambda$
Exemplo: remover 40



Árvore binária de busca: remover

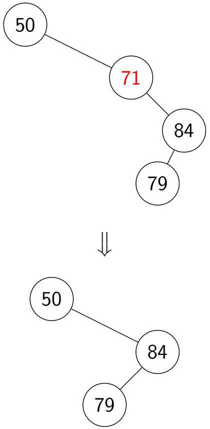
Algoritmo: RemoverBST(r, x)

Entrada: nó raiz r da BST, elemento x a ser removido
Saída: raiz da BST completa

$v =$ BuscaBST(r, x)
se $v \neq \lambda$ **então**
  se $v \rightarrow esq == \lambda$ **então**
    $r =$ Substitui$(r, v, v \rightarrow dir)$
  senão se $v \rightarrow dir == \lambda$ **então**
    $r =$ Substitui$(r, v, v \rightarrow esq)$
  senão
    $s =$ MinimoBST$(v \rightarrow dir)$ //ou $s =$ SucessorBST(v)
    se $s \rightarrow pai \neq v$ **então**
      $r =$ Substitui$(r, s, s \rightarrow dir)$
      $s \rightarrow dir = v \rightarrow dir$
      $s \rightarrow dir \rightarrow pai = s$
    $r =$ Substitui(r, v, s)
    $s \rightarrow esq = v \rightarrow esq$
    $s \rightarrow esq \rightarrow pai = s$
  Desaloca(v)
senão
  print("Elemento não encontrado!")
retorne r

Complexidade: proporcional à altura da BST - $O(h)$

Caso 3: $v \rightarrow esq = \lambda$ e $v \rightarrow dir \neq \lambda$
Exemplo: remover 71



Árvore binária de busca: remover

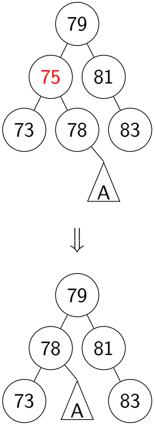
Algoritmo: RemoverBST(r, x)

Entrada: nó raiz r da BST, elemento x a ser removido
Saída: raiz da BST completa

$v = $ BuscaBST(r, x)
se $v \neq \lambda$ **então**
  se $v \rightarrow esq == \lambda$ **então**
    $r = $ Substitui$(r, v, v \rightarrow dir)$
  senão se $v \rightarrow dir == \lambda$ **então**
    $r = $ Substitui$(r, v, v \rightarrow esq)$
  senão
    $s = $ MinimoBST$(v \rightarrow dir)$ //ou $s = $ SucessorBST(v)
    se $s \rightarrow pai \neq v$ **então**
      $r = $ Substitui$(r, s, s \rightarrow dir)$
      $s \rightarrow dir = v \rightarrow dir$
      $s \rightarrow dir \rightarrow pai = s$
    $r = $ Substitui(r, v, s)
    $s \rightarrow esq = v \rightarrow esq$
    $s \rightarrow esq \rightarrow pai = s$
  Desaloca(v)
senão
  print("Elemento não encontrado!")
retorne r

Complexidade: proporcional à altura da BST - $O(h)$

Caso 4: $v \rightarrow esq \neq \lambda$ e $v \rightarrow dir \neq \lambda$ e $v = s \rightarrow pai$ (ou $v \rightarrow dir = s$)
Exemplo: remover 75



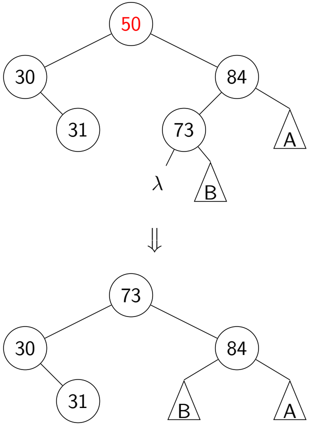
Algoritmo: RemoverBST(r, x)

Entrada: nó raiz r da BST, elemento x a ser removido
Saída: raiz da BST completa

$v =$ BuscaBST(r, x)
se $v \neq \lambda$ **então**
  se $v \rightarrow esq == \lambda$ **então**
    $r =$ Substitui$(r, v, v \rightarrow dir)$
  senão se $v \rightarrow dir == \lambda$ **então**
    $r =$ Substitui$(r, v, v \rightarrow esq)$
  senão
    $s =$ MinimoBST$(v \rightarrow dir)$ //ou $s =$ SucessorBST(v)
    se $s \rightarrow pai \neq v$ **então**
      $r =$ Substitui$(r, s, s \rightarrow dir)$
      $s \rightarrow dir = v \rightarrow dir$
      $s \rightarrow dir \rightarrow pai = s$
    $r =$ Substitui(r, v, s)
    $s \rightarrow esq = v \rightarrow esq$
    $s \rightarrow esq \rightarrow pai = s$
  Desaloca(v)
senão
  print("Elemento não encontrado!")
retorne r

Complexidade: proporcional à altura da BST - $O(h)$

Caso 5: $v \rightarrow esq \neq \lambda$ e $v \rightarrow dir \neq \lambda$ e $v \neq s \rightarrow pai$ (ou $v \rightarrow dir \neq s$)
Exemplo: remover 50



Árvore binária de busca: remoção

Casos possíveis:

1. $v \rightarrow esq = \lambda$ e $v \rightarrow dir = \lambda$
   - basta remover o nó
2. $v \rightarrow esq \neq \lambda$ e $v \rightarrow dir = \lambda$
   - substituímos v por $v \rightarrow esq$
3. $v \rightarrow esq = \lambda$ e $v \rightarrow dir \neq \lambda$
   - substituímos v por $v \rightarrow dir$
4. $v \rightarrow esq \neq \lambda$ e $v \rightarrow dir \neq \lambda$ e $v = s \rightarrow pai$ (ou $v \rightarrow dir = s$)
   - substituímos v por s diretamente, sem necessidade de mexer mais

? E se $v \rightarrow esq \neq \lambda$ e $v \rightarrow dir \neq \lambda$, mas v não tiver sucessor?
   - Se $v \rightarrow dir \neq \lambda$, então existe um sucessor de v.
   - Se v não tem sucessor, então $v \rightarrow dir = \lambda$.
   - Ou seja, esse não é um caso possível.