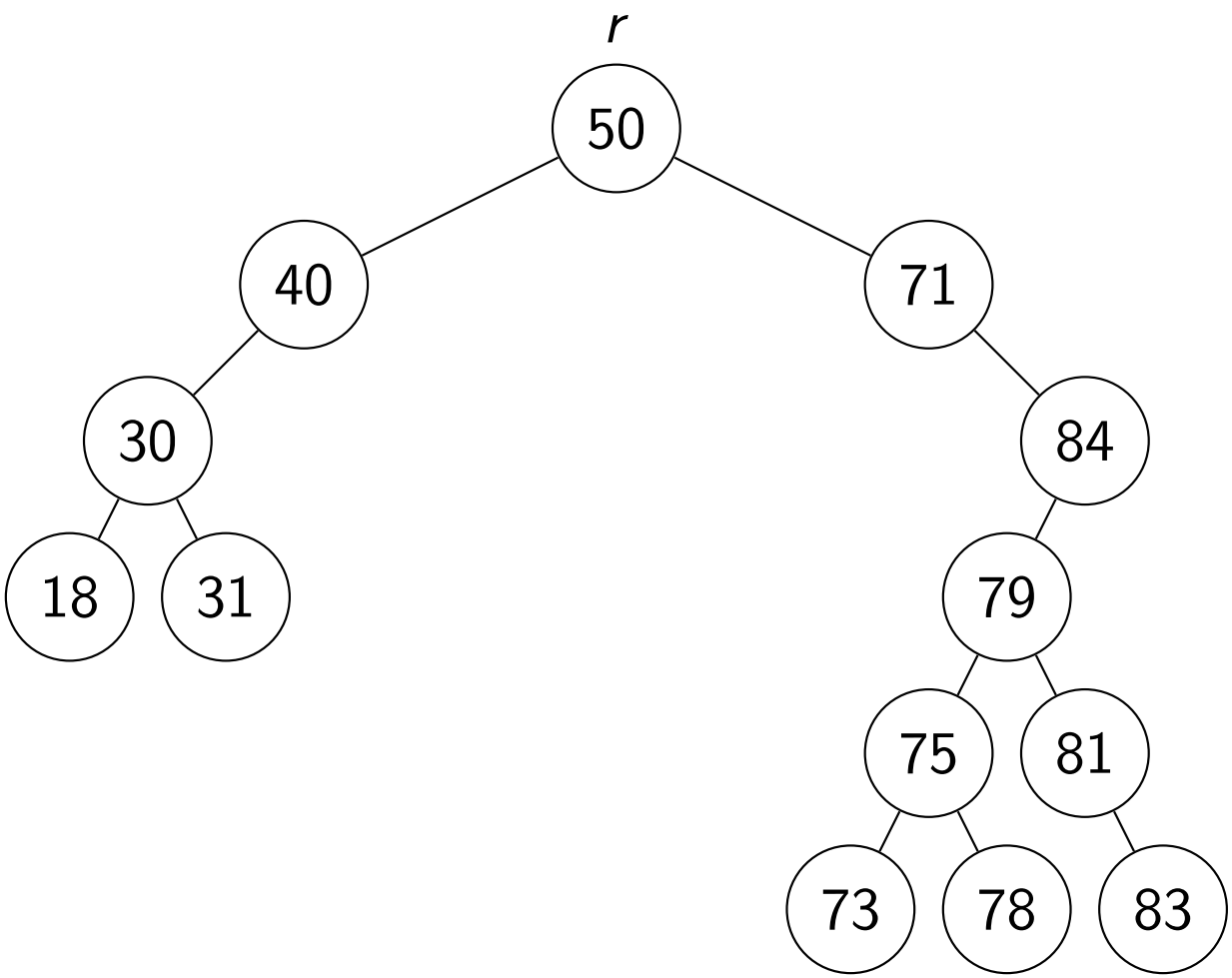
Estruturas de Dados - Árvore Binária de Busca 03

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)

```
1 se v→p == λ então
2 |   r = u
3 senão se v == v→p→esq então
4 |   v→p→esq = u
5 senão
6 |   v→p→dir = u
7 se u ≠ λ então
8 |   u→p = v→p
9 retorne r
```

Complexidade: $O(1)$

Árvore binária de busca: remover

Algoritmo: RemoverBST(r, v)

Entrada: nó raiz r da BST, nó v a ser removido

Saída: raiz da BST completa

```
1  se v→esq == λ então
2  |   r = Substitui(r, v, v→dir)
3  senão se v→dir == λ então
4  |   r = Substitui(r, v, v→esq)
5  senão
6  |   s = MinimoBST(v→dir) //ou s = SucessorBST(v)
7  |   se s→p ≠ v (ou v→dir ≠ s) então
8  |   |   r = Substitui(r, s, s→dir)
9  |   |   s→dir = v→dir
10 |   |   s→dir→p = s
11 |   r = Substitui(r, v, s)
12 |   s→esq = v→esq
13 |   s→esq→p = s
14 retorne r
```

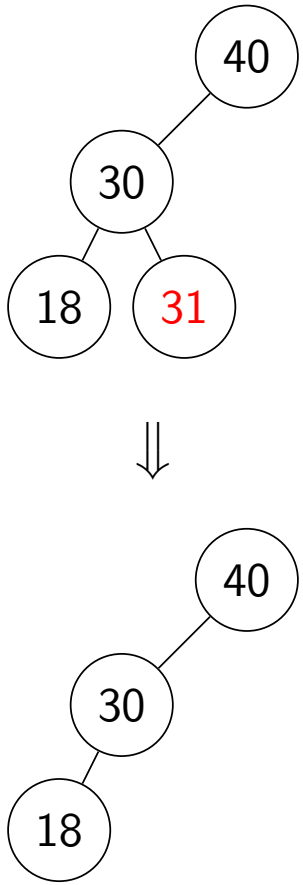
Complexidade: proporcional à altura da BST - $O(h)$

Árvore binária de busca: remover

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

```
1  se v→esq == λ então
2  |   r = Substitui(r, v, v→dir)
3  senão se v→dir == λ então
4  |   r = Substitui(r, v, v→esq)
5  senão
6  |   s = MinimoBST(v→dir) //ou s = SucessorBST(v)
7  |   se s→p ≠ v (ou v→dir ≠ s) então
8  |   |   r = Substitui(r, s, s→dir)
9  |   |   s→dir = v→dir
10 |   |   s→dir→p = s
11 |   r = Substitui(r, v, s)
12 |   s→esq = v→esq
13 |   s→esq→p = s
14 retorne r
```

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



Árvore binária de busca: remover

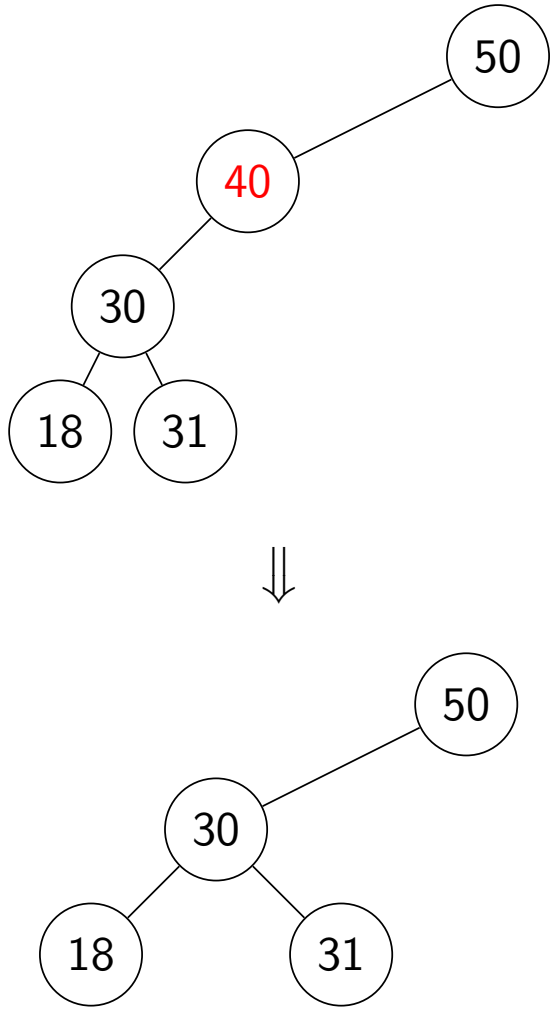
Algoritmo: RemoverBST(r, v)

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

```
1  se v→esq == λ então
2  |   r = Substitui(r, v, v→dir)
3  senão se v→dir == λ então
4  |   r = Substitui(r, v, v→esq)
5  senão
6  |   s = MinimoBST(v→dir) //ou s = SucessorBST(v)
7  |   se s→p ≠ v (ou v→dir ≠ s) então
8  |   |   r = Substitui(r, s, s→dir)
9  |   |   s→dir = v→dir
0  |   |   s→dir→p = s
1  |   r = Substitui(r, v, s)
2  |   s→esq = v→esq
3  |   s→esq→p = s
4  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

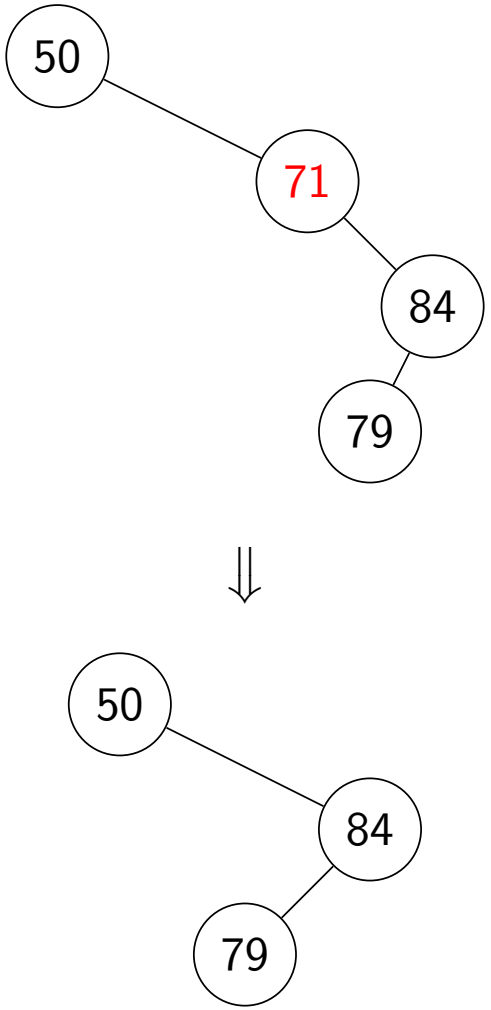
Algoritmo: RemoverBST(r, v)

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

```
1  se v→esq == λ então
2  |   r = Substitui(r, v, v→dir)
3  senão se v→dir == λ então
4  |   r = Substitui(r, v, v→esq)
5  senão
6  |   s = MinimoBST(v→dir) //ou s = SucessorBST(v)
7  |   se s→p ≠ v (ou v→dir ≠ s) então
8  |   |   r = Substitui(r, s, s→dir)
9  |   |   s→dir = v→dir
0  |   |   s→dir→p = s
1  |   r = Substitui(r, v, s)
2  |   s→esq = v→esq
3  |   s→esq→p = s
4  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

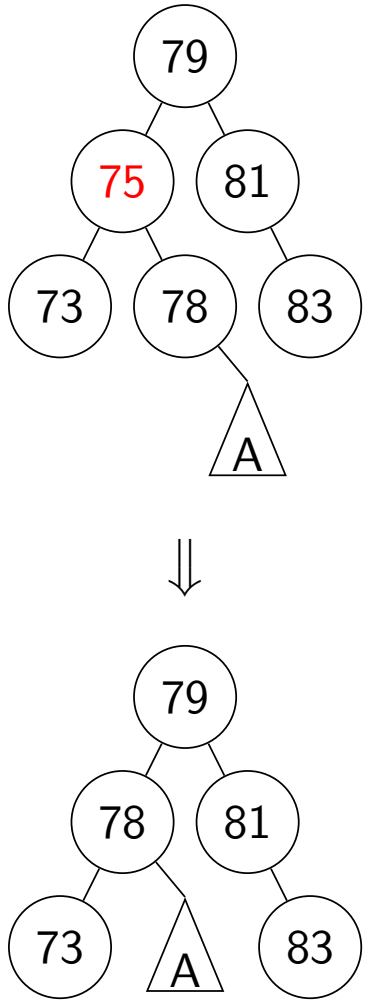
Algoritmo: RemoverBST(r, v)

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

```
1  se v→esq == λ então
2  |   r = Substitui(r, v, v→dir)
3  senão se v→dir == λ então
4  |   r = Substitui(r, v, v→esq)
5  senão
6  |   s = MinimoBST(v→dir) //ou s = SucessorBST(v)
7  |   se s→p ≠ v (ou v→dir ≠ s) então
8  |   |   r = Substitui(r, s, s→dir)
9  |   |   s→dir = v→dir
0  |   |   s→dir→p = s
1  |   r = Substitui(r, v, s)
2  |   s→esq = v→esq
3  |   s→esq→p = s
4  retorne r
```

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

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



Árvore binária de busca: remover

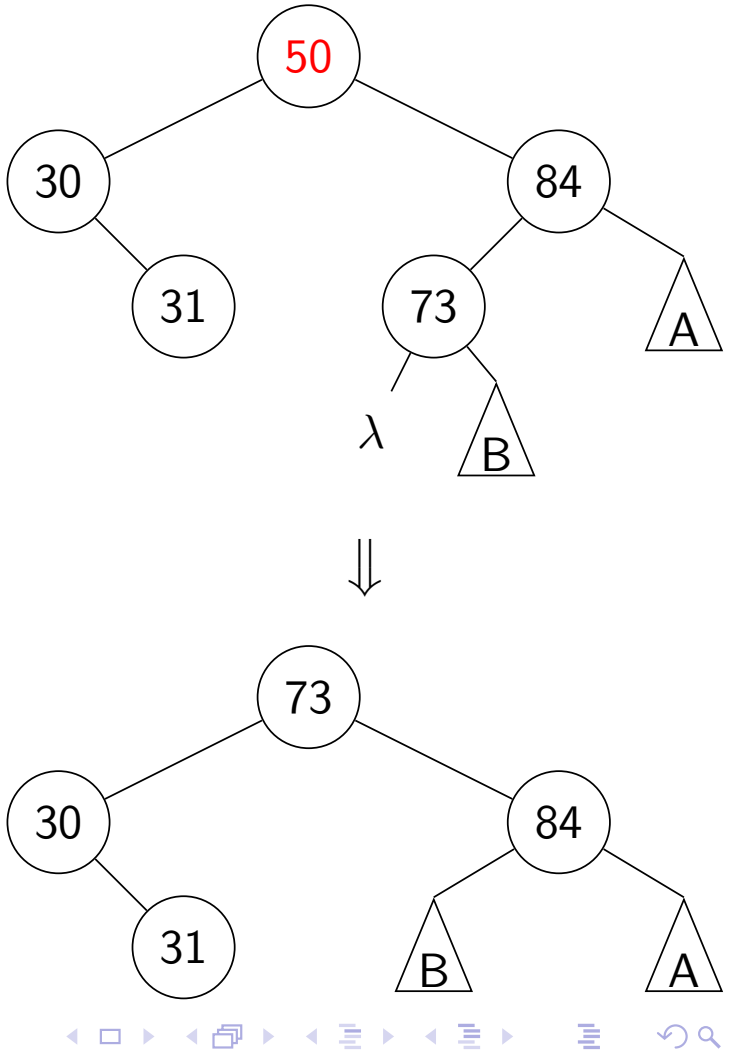
Algoritmo: RemoverBST(r, v)

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

```
1  se v→esq == λ então
2  |   r = Substitui(r, v, v→dir)
3  senão se v→dir == λ então
4  |   r = Substitui(r, v, v→esq)
5  senão
6  |   s = MinimoBST(v→dir) //ou s = SucessorBST(v)
7  |   se s→p ≠ v (ou v→dir ≠ s) então
8  |   |   r = Substitui(r, s, s→dir)
9  |   |   s→dir = v→dir
0  |   |   s→dir→p = s
1  |   r = Substitui(r, v, s)
2  |   s→esq = v→esq
3  |   s→esq→p = s
4  retorne r
```

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

Caso 5: $v{\rightarrow}esq \neq \lambda$ e $v{\rightarrow}dir \neq \lambda$ e $v \neq s{\rightarrow}p$ (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 $v = s \rightarrow p$ (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
- v não tem sucessor se e somente se $v \rightarrow dir = \lambda$
- ou seja, esse não é um caso possível