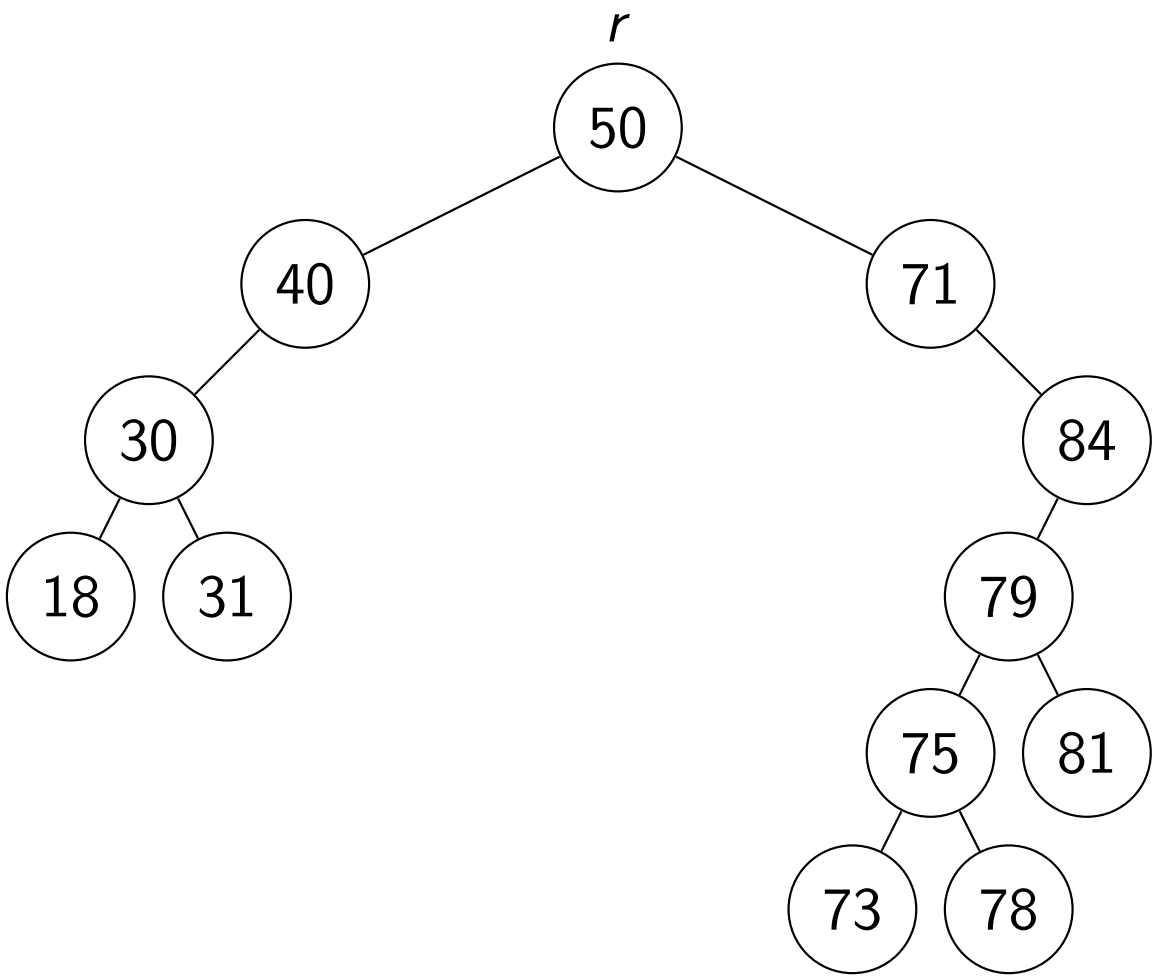
Estruturas de Dados - Árvore Binária de Busca 02

O que já vimos

- Aula passada:
  - Árvore binária de busca
  - Estrutura do nó
  - Métodos:
    - criar BST (retorna árvore vazia)
    - buscar por um valor (retorna o nó)
    - incluir na BST (retorna a raiz)
    - mínimo e máximo de uma BST (retornam o nó)

Árvore binária de busca: percurso em ordem

- Exemplo: visitar os elementos da BST em ordem



Árvore binária de busca: percurso em ordem (ou em ordem simétrica)

Algoritmo: EmOrdemBST(r)

Entrada: nó raiz r da BST

```
1 se r ≠ λ então
2 |   EmOrdemBST(r→esq)
3 |   visitar r→chave
4 |   EmOrdemBST(r→dir)
```

Complexidade: proporcional à quantidade de nós na BST - $O(n)$

Árvore binária de busca: percurso pré-ordem

Algoritmo: PreOrdemBST(r)

Entrada: nó raiz r da BST

```
1 se r ≠ λ então
2 |   visitar r→chave
3 |   PreOrdemBST(r→esq)
4 |   PreOrdemBST(r→dir)
```

Complexidade: proporcional à quantidade de nós na BST - $O(n)$

Árvore binária de busca: percurso pós-ordem

Algoritmo: PosOrdemBST(r)

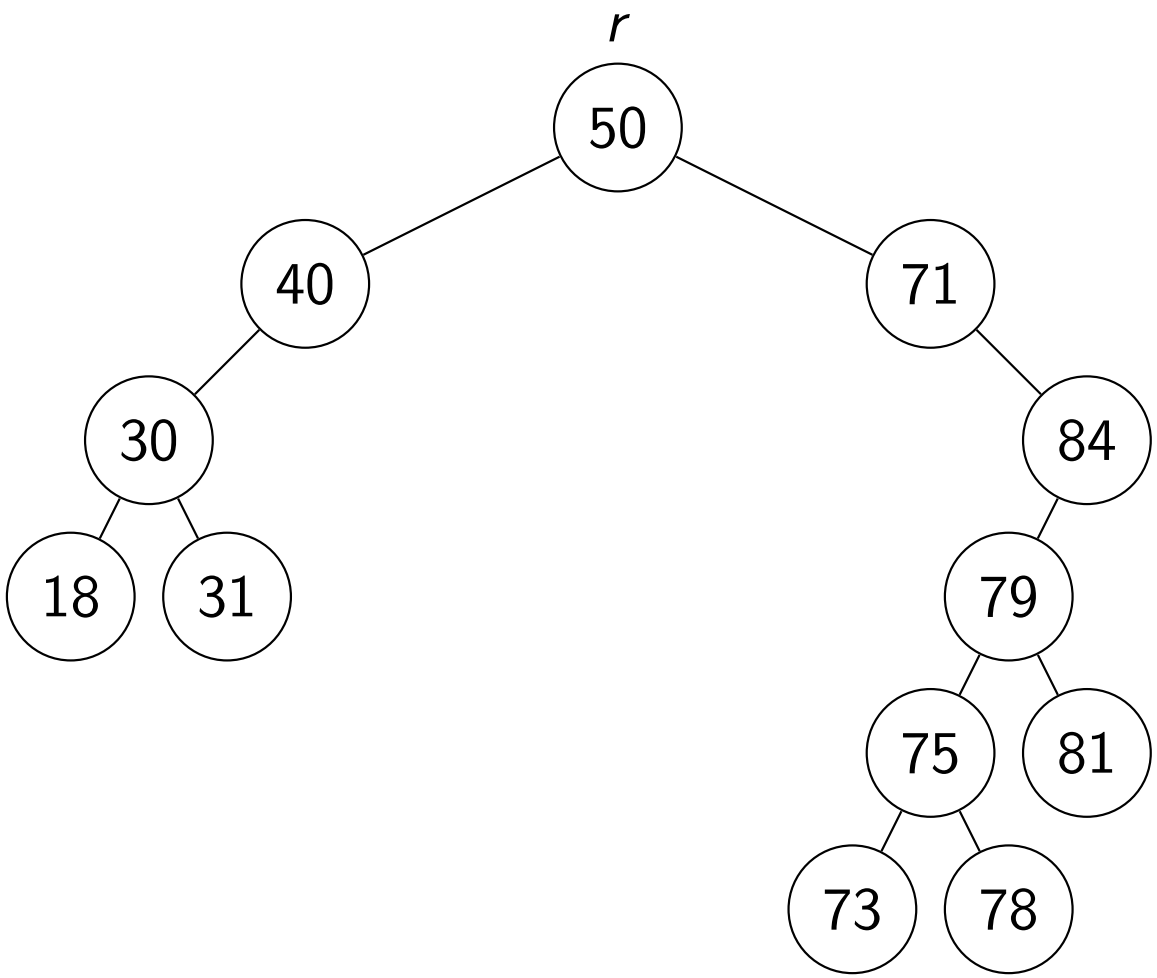
Entrada: nó raiz r da BST

```
1 se r ≠ λ então
2 |   PosOrdemBST(r→esq)
3 |   PosOrdemBST(r→dir)
4 |   visitar r→chave
```

Complexidade: proporcional à quantidade de nós na BST - $O(n)$

Árvore binária de busca: altura da BST

- Exemplo: calcular a altura desta BST



Árvore binária de busca: altura da BST

Algoritmo: AlturaBST(r)

Entrada: nó raiz r da BST

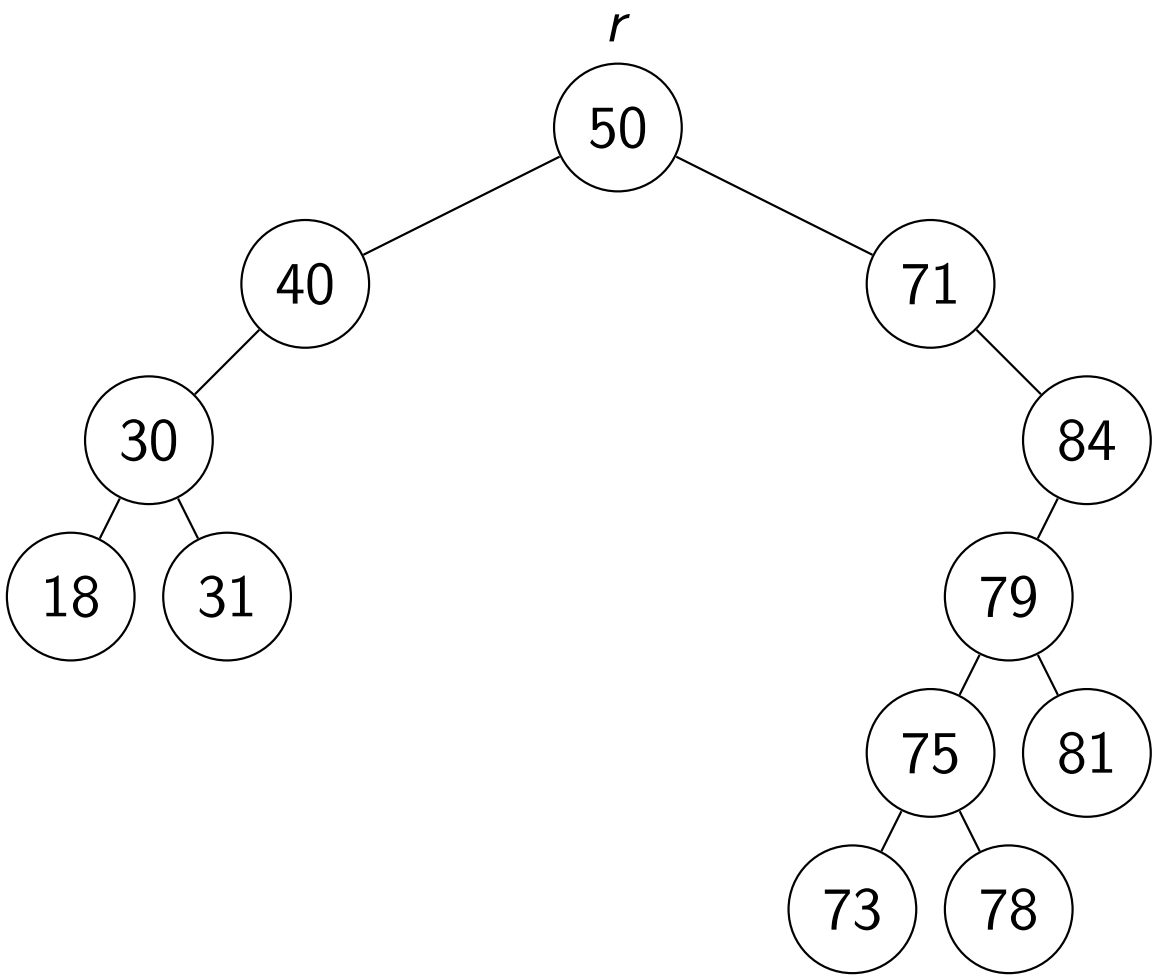
Saída: altura da BST

```
1 se r == λ então
2 |   retorne 0
3 alt_e = AlturaBST(r→esq)
4 alt_d = AlturaBST(r→dir)
5 retorne 1 + max{alt_d, alt_e}
```

Complexidade: proporcional à quantidade de nós na BST - $O(n)$

Árvore binária de busca: altura de cada nó (campo *alt*)

- Exemplo: setar o campo *alt* de cada um dos nós desta BST



Árvore binária de busca: altura de cada nó (campo *alt*)

Algoritmo: AlturaBST(r)

Entrada: nó raiz r da BST

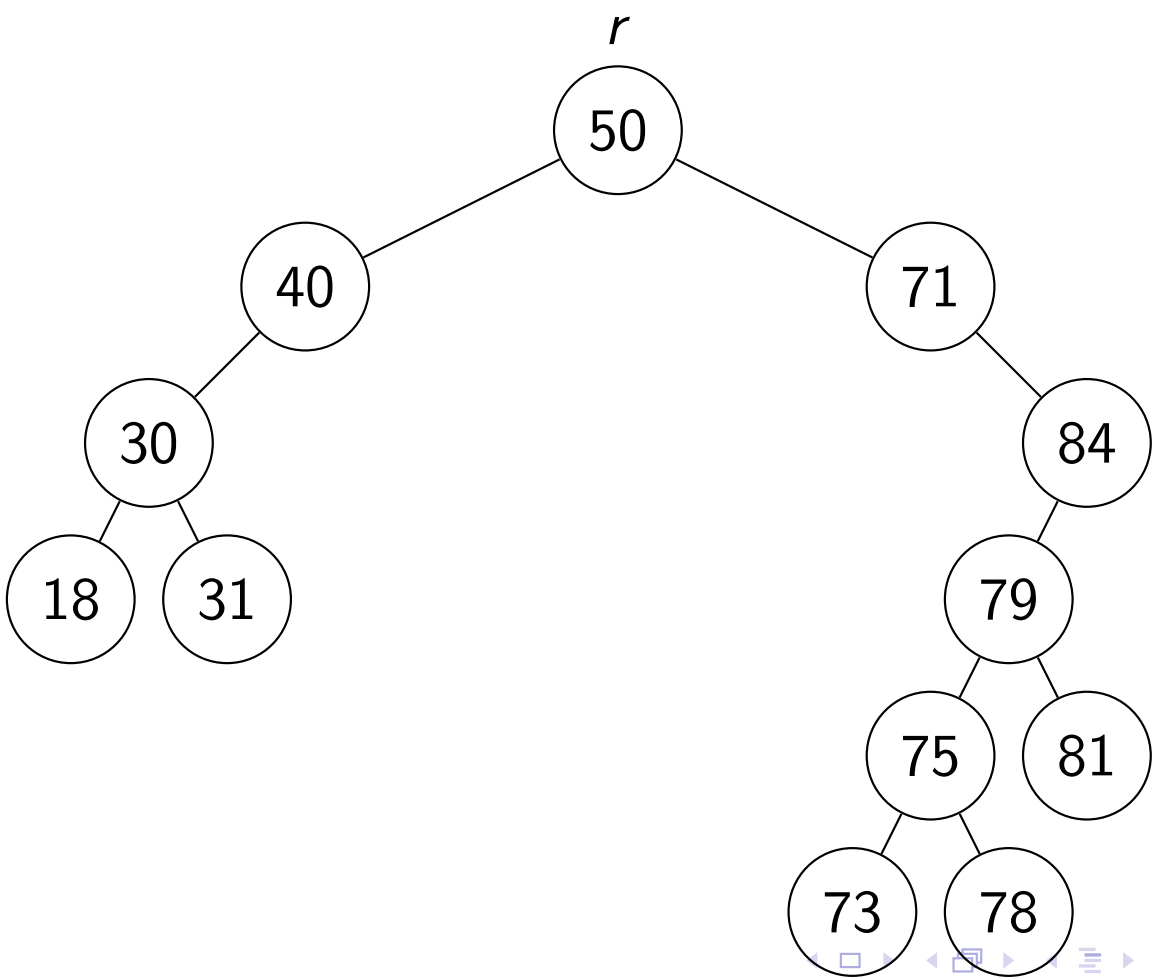
Saída: altura da BST, atualizando os campos *alt*

1 **se** $r == \lambda$ **então**
2 | **retorne** 0
3 alt_e = AlturaBST($r \rightarrow esq$)
4 alt_d = AlturaBST($r \rightarrow dir$)
5 $r \rightarrow alt = 1 + \max\{alt\_d, alt\_e\}$
6 **retorne** $1 + \max\{alt\_d, alt\_e\}$

Complexidade: proporcional à quantidade de nós na BST - $O(n)$

Árvore binária de busca: pai de cada nó (campo p)

- Exemplo: setar o campo p de cada um dos nós desta BST
  Obs: o pai da raiz é λ!



Árvore binária de busca: pai de cada nó (campo p)

Algoritmo: PaisBST(r)

Entrada: nó raiz r da BST

```
1 r→p = λ
2 se r→esq ≠ λ então
3 |   PaisBST(r→esq)
4 |   r→esq→p = r
5 se r→dir ≠ λ então
6 |   PaisBST(r→dir)
7 |   r→dir→p = r
```

Complexidade: proporcional à quantidade de nós na BST - $O(n)$

Árvore binária de busca: incluir (setando o campo p)

Algoritmo: IncluiBST(r, v)

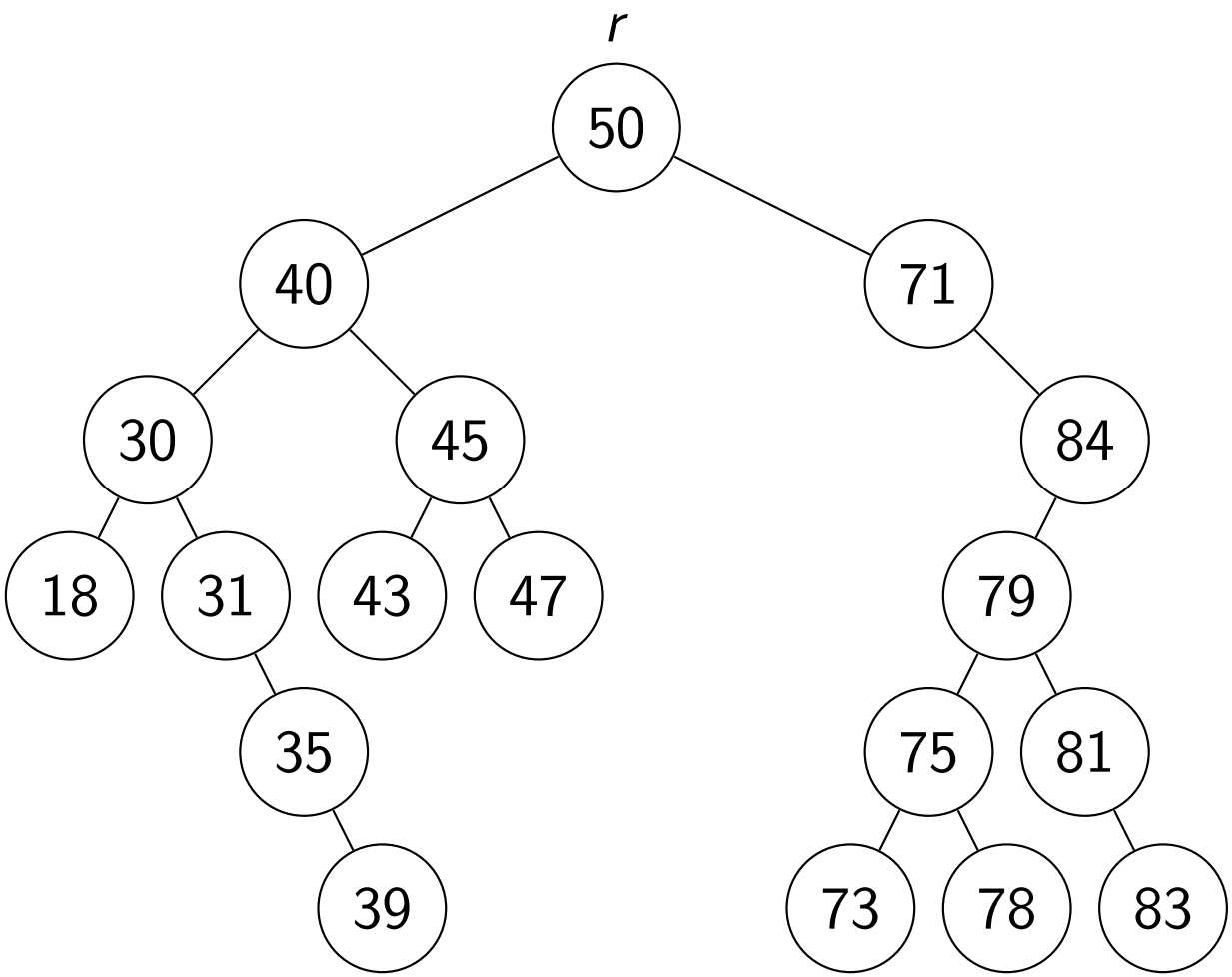
Entrada: nó raiz r da BST, nó v a ser incluído
Saída: raiz da árvore resultante

```
1  se r == λ então
2  |   v→p = λ
3  |   retorne v
4  t = r
5  criar novo nó u
6  enquanto r ≠ λ faça
7  |   u = r
8  |   se v→chave ≤ r→chave então
9  |   |   r = r→esq
10 |   senão
11 |   |   r = r→dir
12 se v→chave ≤ u→chave então
13 |   u→esq = v
14 senão
15 |   u→dir = v
16 v→p = u
17 retorne t
```

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

Árvore binária de busca: sucessor e antecessor de um nó

- Exemplo: encontrar o sucessor e o antecessor de um nó nesta BST



Árvore binária de busca: sucessor

Algoritmo: SucessorBST(v)

Entrada: nó v (não nulo)

Saída: nó sucessor de v na BST (ou λ caso v seja o máximo da BST)

```
1 se v→dir ≠ λ então
2 |   retorne MinimoBST(v→dir)
3 u = v→dir
4 enquanto v ≠ λ e u == v→dir faça
5 |   u = v
6 |   v = v→p
7 retorne v
```

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

Árvore binária de busca: antecessor

Algoritmo: AntecessorBST(v)

Entrada: nó v (não nulo)

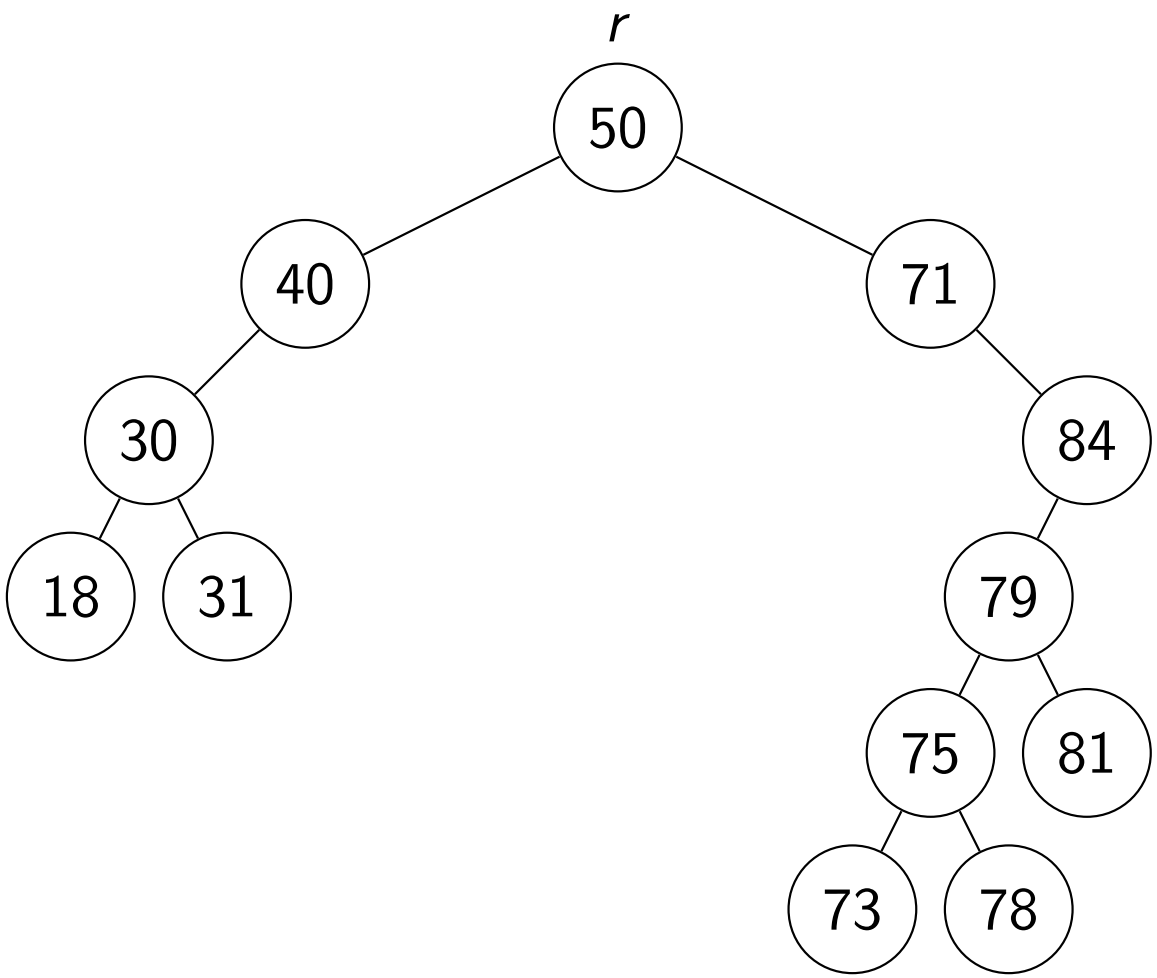
Saída: nó antecessor de v na BST (ou λ caso v seja o mínimo da BST)

```
1 se v→esq ≠ λ então
2 |   retorne MaximoBST(v→esq)
3 u = v→esq
4 enquanto v ≠ λ e u == v→esq faça
5 |   u = v
6 |   v = v→p
7 retorne v
```

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

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!