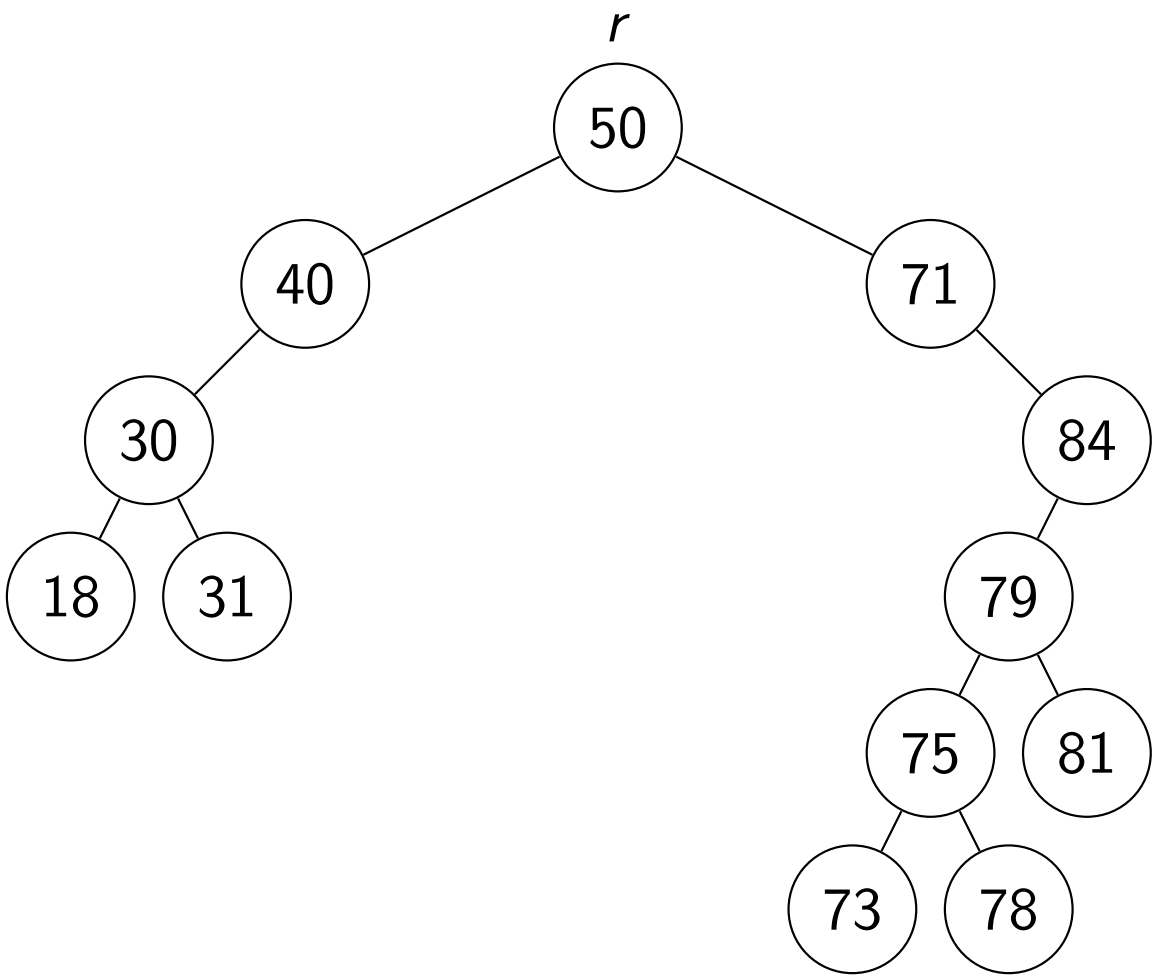
Estruturas de Dados - Árvore Binária de Busca 02

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)

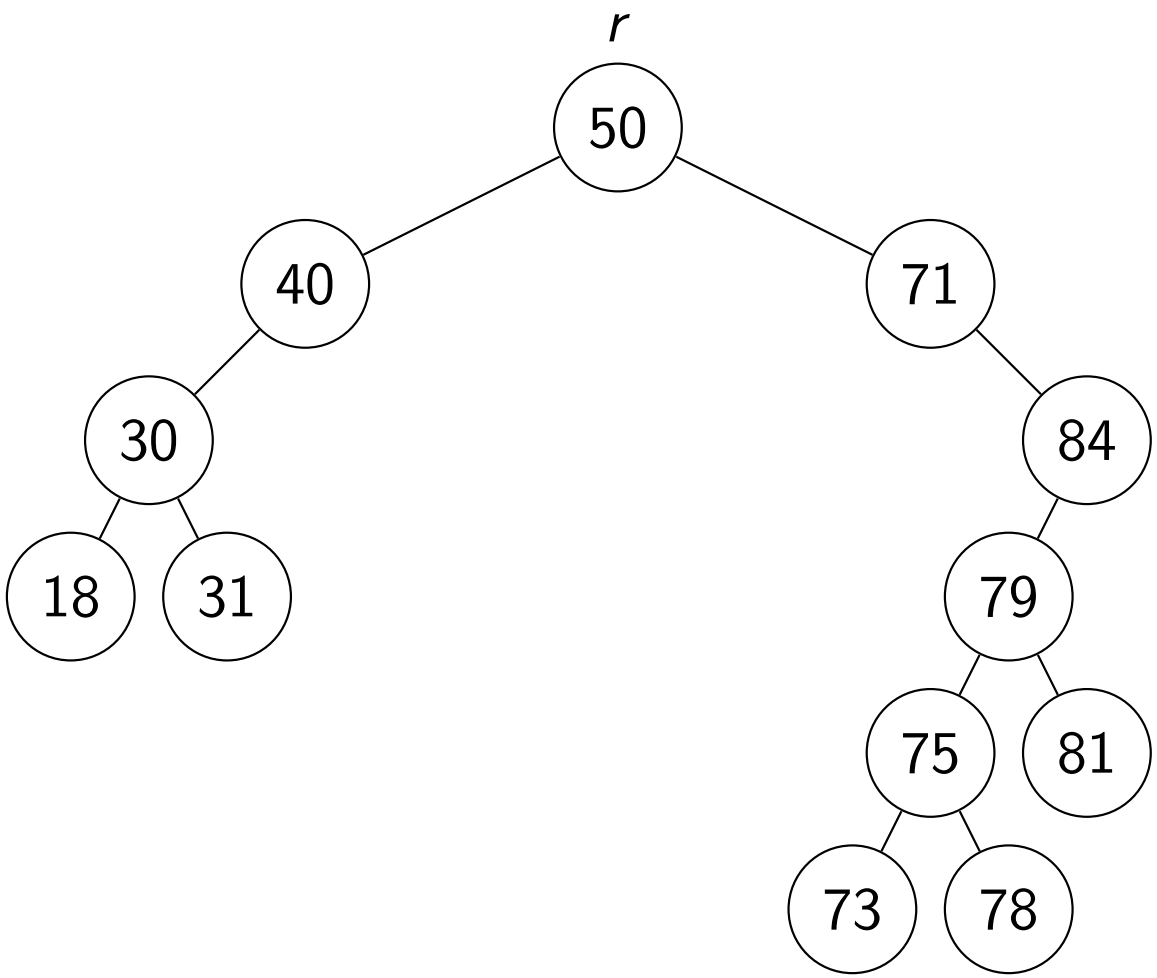
Entrada: nó raiz r da BST

```
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

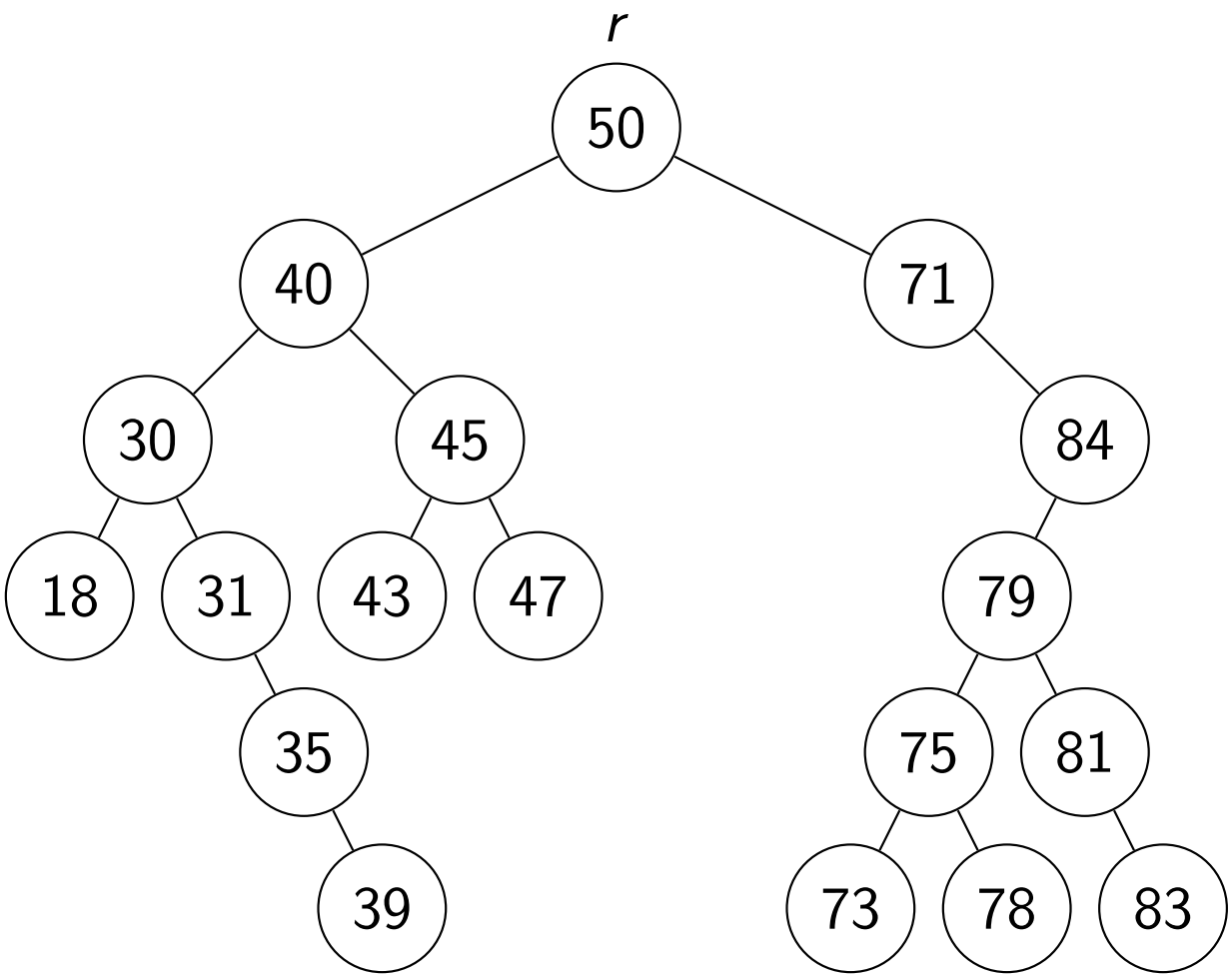
Saída: altura da BST

```
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: 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 f = v→dir
4 enquanto v ≠ λ e f == v→dir faça
5 |   f = v
6 |   v = v→pai
7 retorne v
```

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

Árvore binária de busca: antecessor

Algoritmo: AntecessorBST(v)

Entrada: nó v (não nulo)

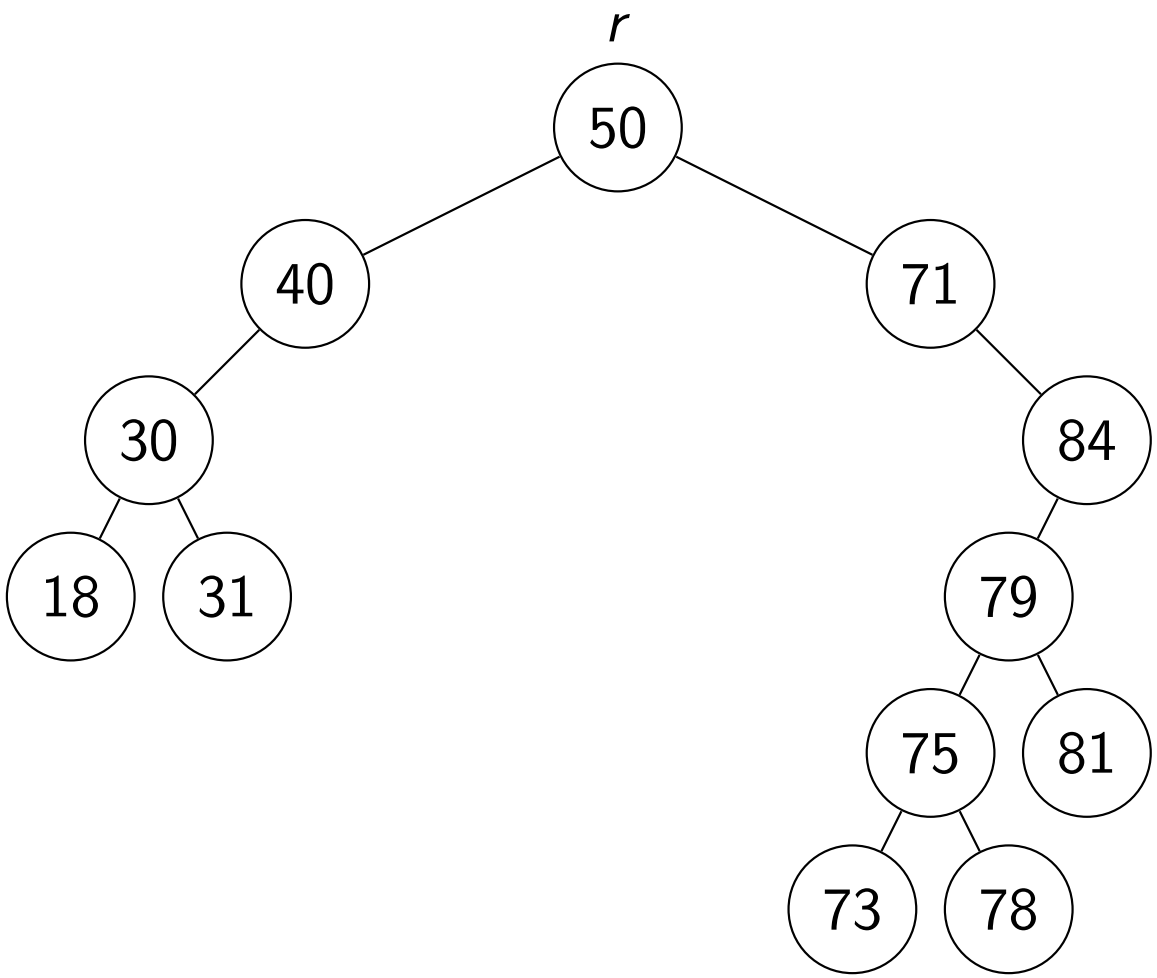
Saída: nó antecessor de v na BST (ou λ caso v seja o mínimo da BST)

```
1 se v→esq ≠ λ então
2 |   retorne MaximoBST(v→esq)
3 f = v→esq
4 enquanto v ≠ λ e f == v→esq faça
5 |   f = v
6 |   v = v→pai
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!