Árvores 2-3

Prof. Kennedy Lopes

22 de março de 2023

Introdução

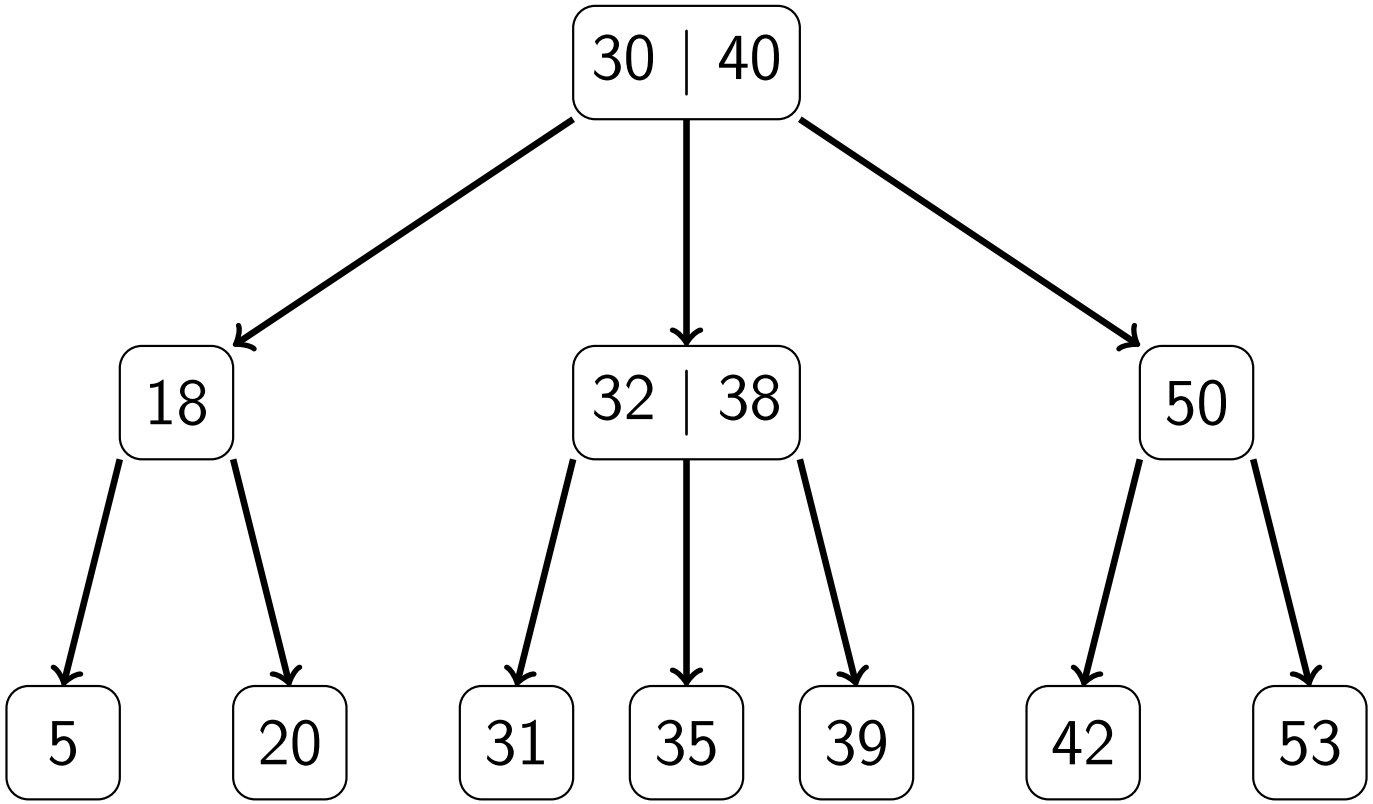
Perguntas:

- O que define a complexidade de uma árvore?
- Como a AVL corrige isso?
- A AVL sempre será cheia/completa?

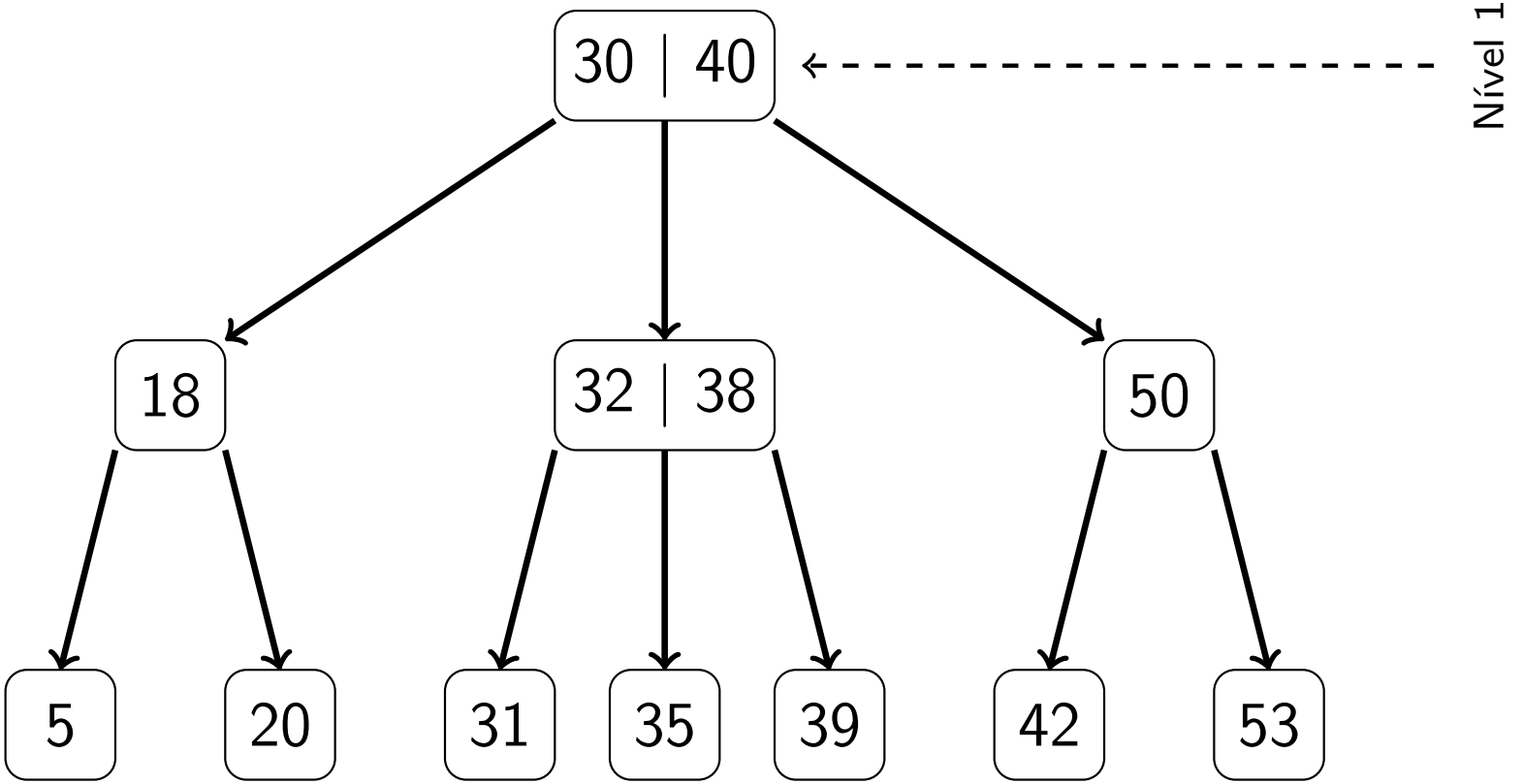
Introdução

Resumidamente, árvore 2-3 é uma árvore na qual os nós podem ter até 3 filhos.

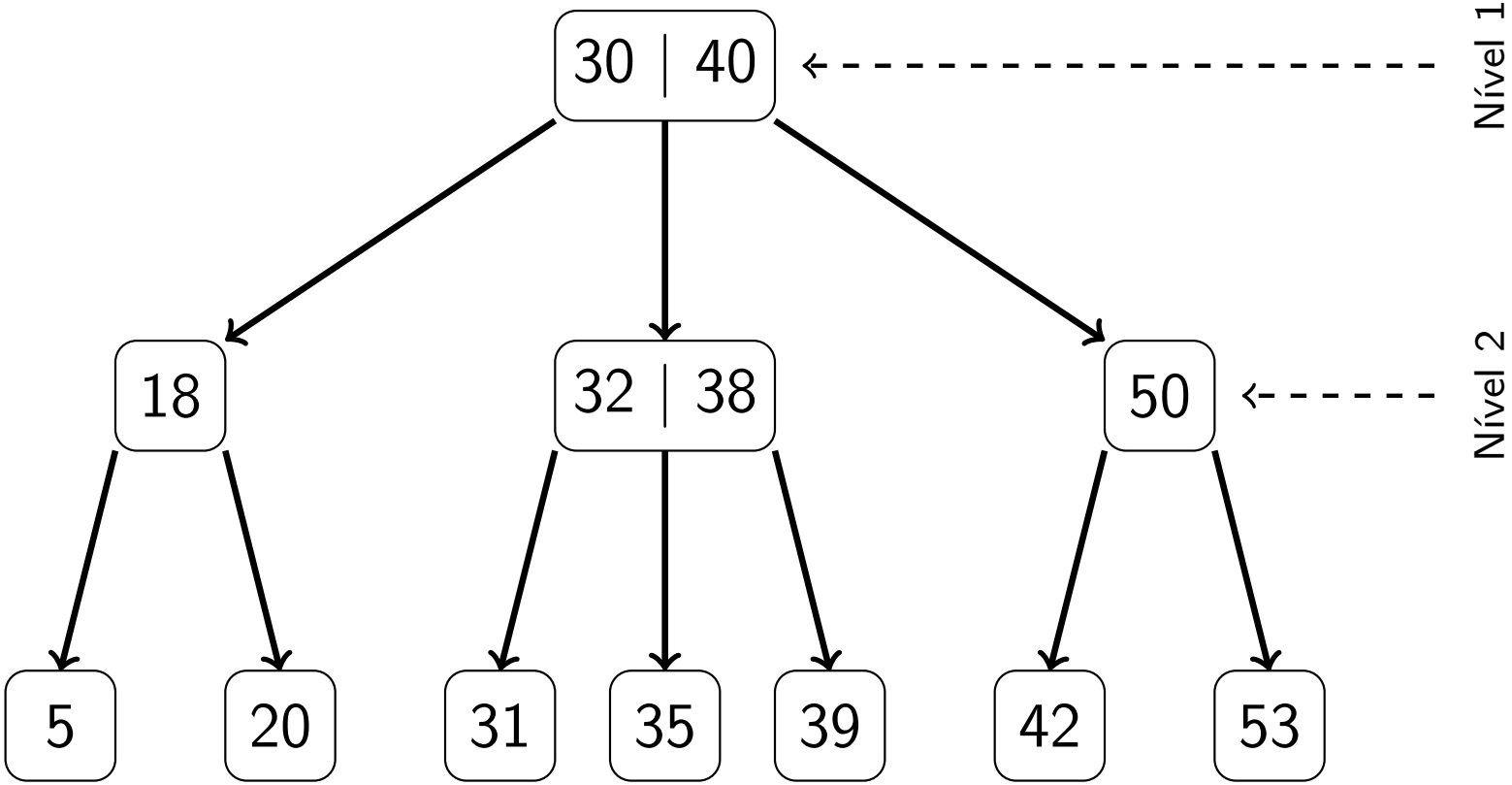
Árvore 2-3



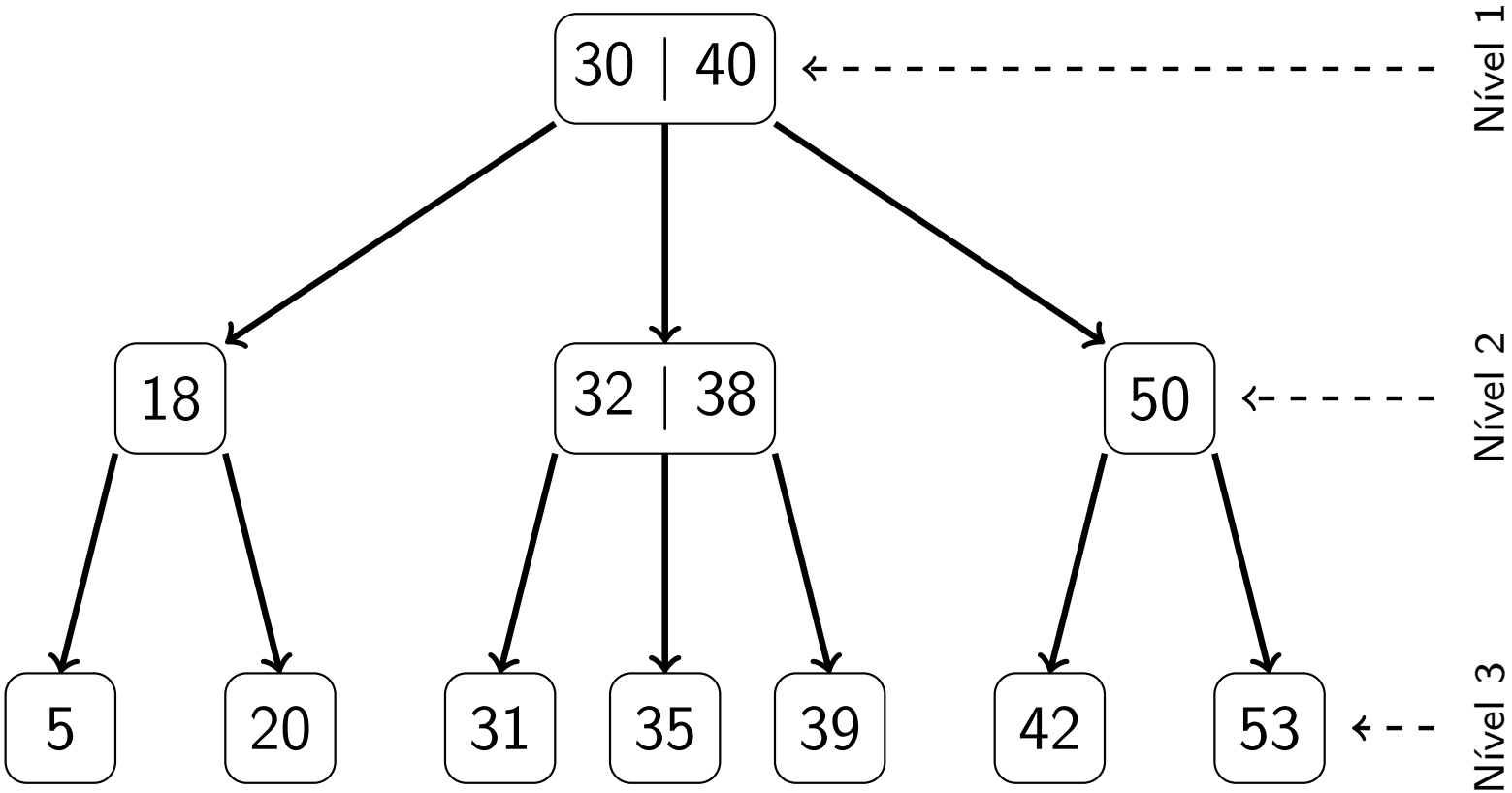
Árvore 2-3



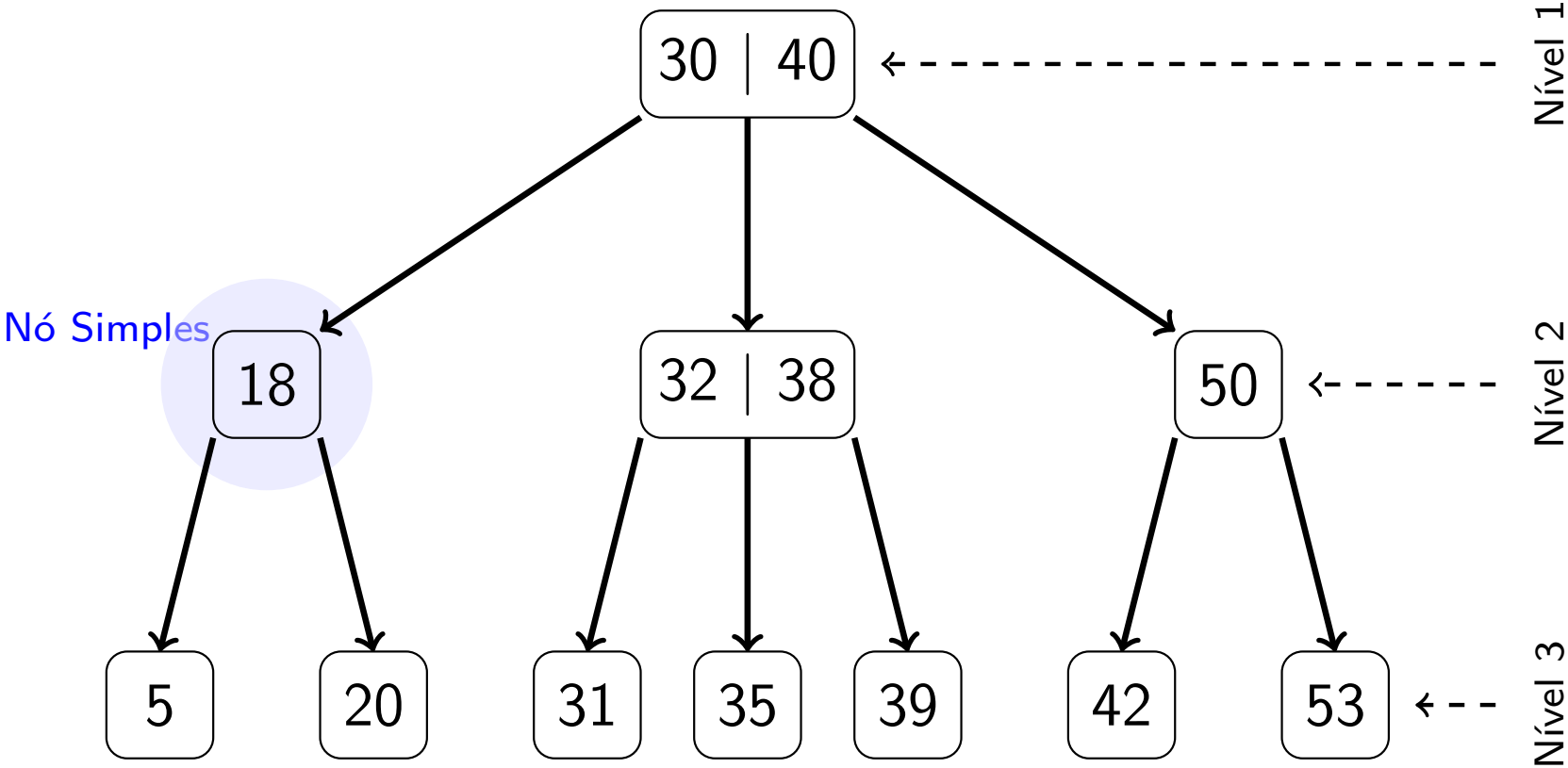
Árvore 2-3



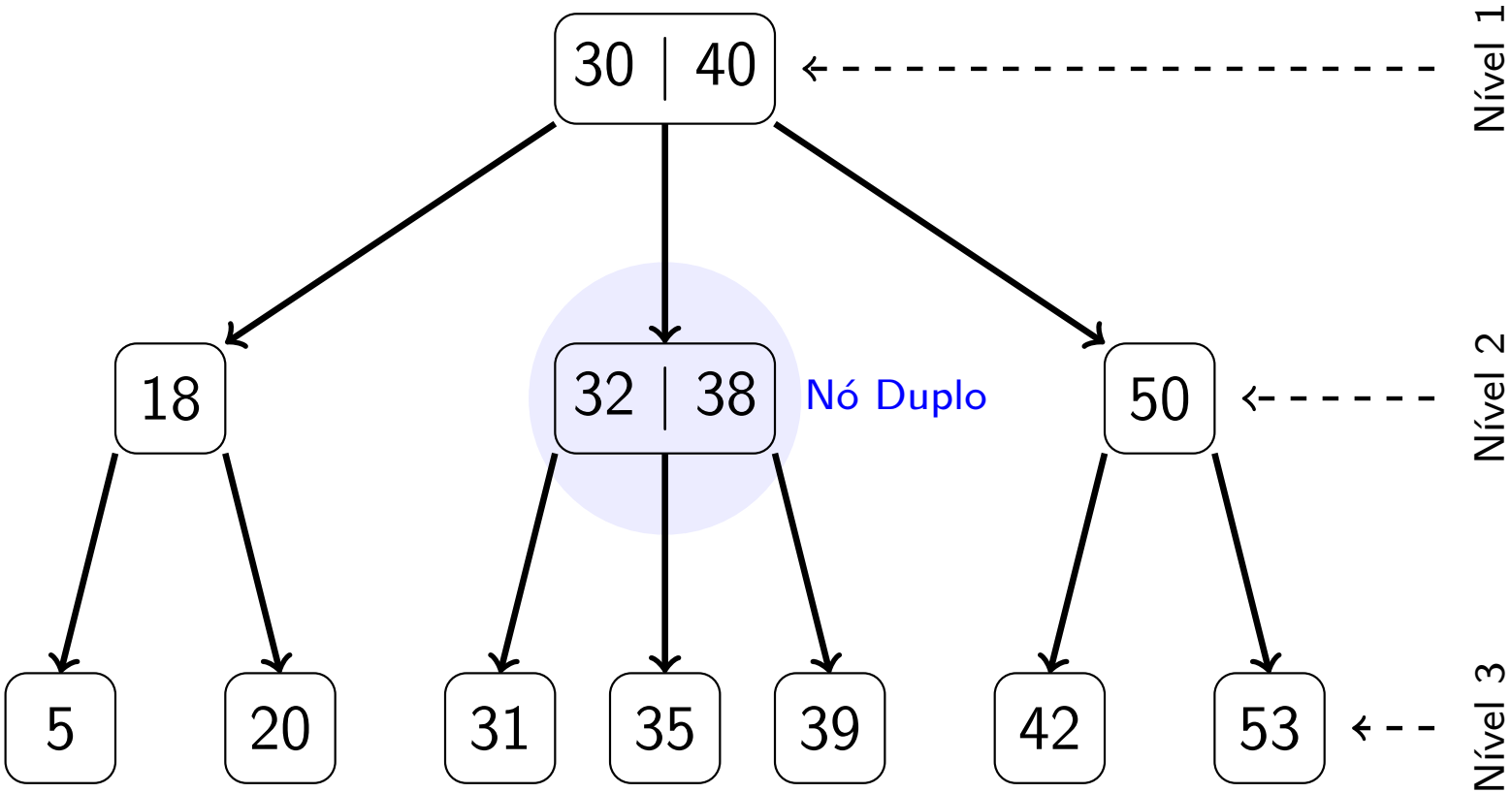
Árvore 2-3



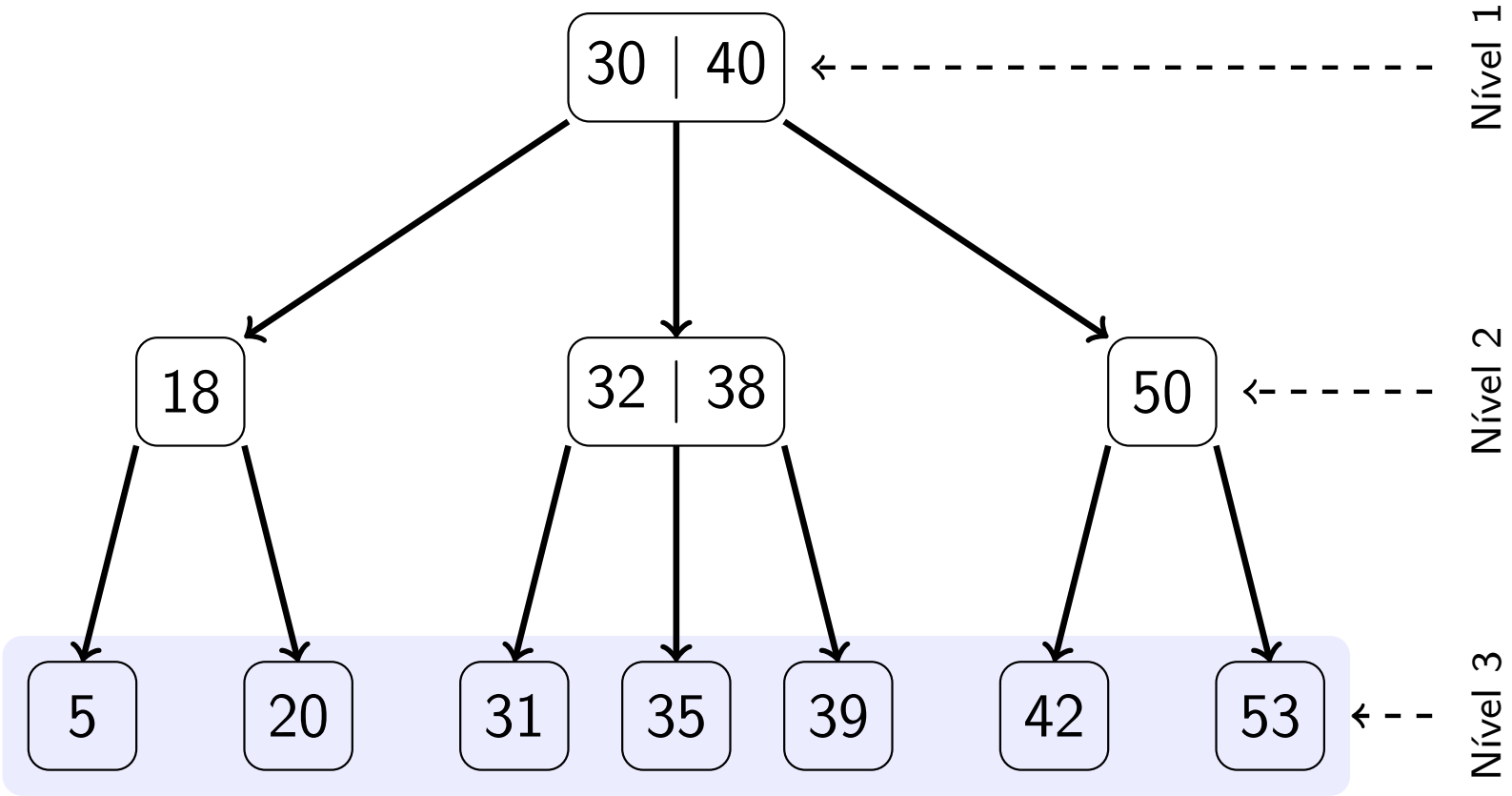
Árvore 2-3


30 | 40
18
32 | 38
50
5
20
31
35
39
42
53
Nó Simples
Nível 1
Nível 2
Nível 3

Árvore 2-3


30 | 40
18
32 | 38
Nó Duplo
50
5
20
31
35
39
42
53
Nível 1
Nível 2
Nível 3

Árvore 2-3



Todos os nós Folhas no último nível

Proriedades

A árvore 2-3 **não binária** obedece as seguintes propriedades:

Proriedades

A árvore 2-3 **não binária** obedece as seguintes propriedades:

- Cada nó contem uma ou duas chaves;

Proriedades

A árvore 2-3 **não binária** obedece as seguintes propriedades:

- Cada nó contem uma ou duas chaves;
- Cada nó interno tem dois (se tem uma chave), ou três (se tem duas chaves) filhos;

Proriedades

A árvore 2-3 **não binária** obedece as seguintes propriedades:

- Cada nó contem uma ou duas chaves;
- Cada nó interno tem dois (se tem uma chave), ou três (se tem duas chaves) filhos;
- Todas as folhas estão no mesmo nível.

Proriedades

A árvore 2-3 **não binária** obedece as seguintes propriedades:

- Cada nó contem uma ou duas chaves;
- Cada nó interno tem dois (se tem uma chave), ou três (se tem duas chaves) filhos;
- Todas as folhas estão no mesmo nível.

Características:

- Estrutura alternativa para uma busca eficiente.
- Existem um grande número de registros;
- Elementos são procurados em uma faixa de valores.

Características

Características

Possuem organização semelhante a uma árvore binária de busca:

Características

Possuem organização semelhante a uma árvore binária de busca:

1. Os filhos da esquerda são menores que o nó pai;

Características

Possuem organização semelhante a uma árvore binária de busca:

1. Os filhos da esquerda são menores que o nó pai;
2. Os filhos da direita são maiores que o nó pai;

Características

Possuem organização semelhante a uma árvore binária de busca:

1. Os filhos da esquerda são menores que o nó pai;
2. Os filhos da direita são maiores que o nó pai;
3. Os filhos centrais, se existirem, são valores entre as duas chaves do nó pai.

Características

Possuem organização semelhante a uma árvore binária de busca:

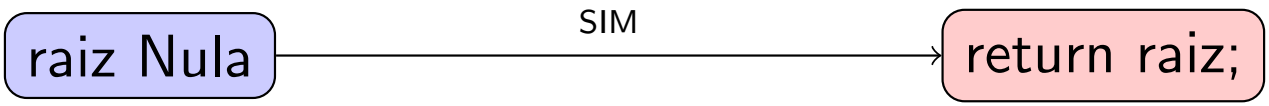
1. Os filhos da esquerda são menores que o nó pai;
2. Os filhos da direita são maiores que o nó pai;
3. Os filhos centrais, se existirem, são valores entre as duas chaves do nó pai.

```
typedef struct No23{
    int chave_esq;
    int chave_dir;
    int num_chaves;
    struct No23* esq;
    struct No23* dir;
    struct No23* central;
} no23;
```

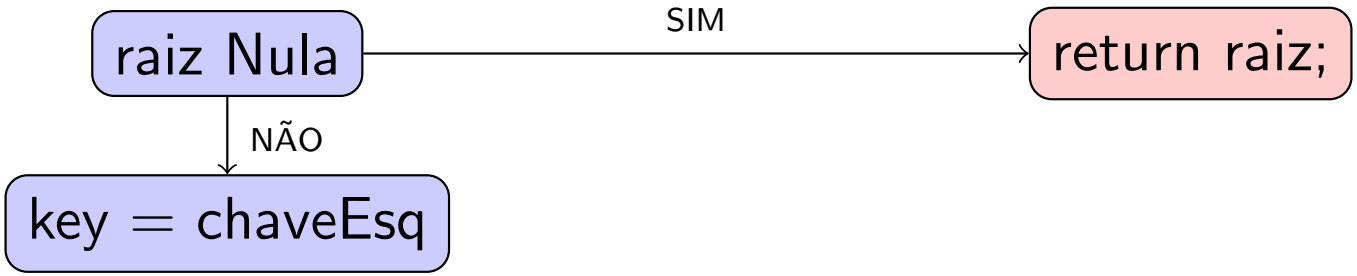
Busca(raiz, key)

raiz Nula

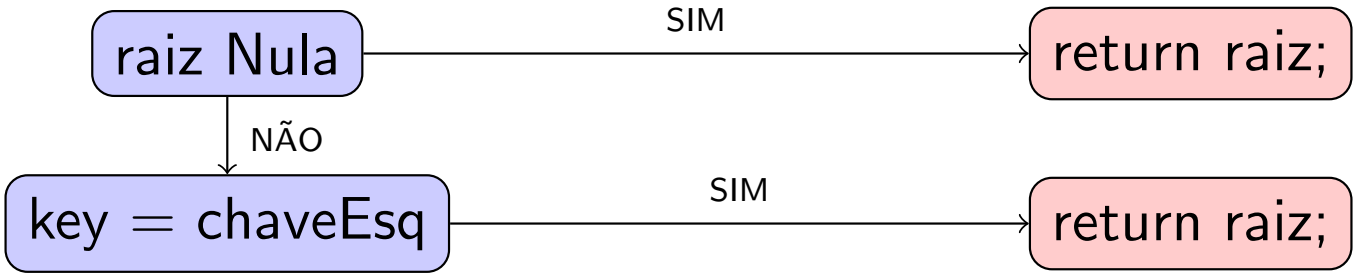
Busca(raiz, key)



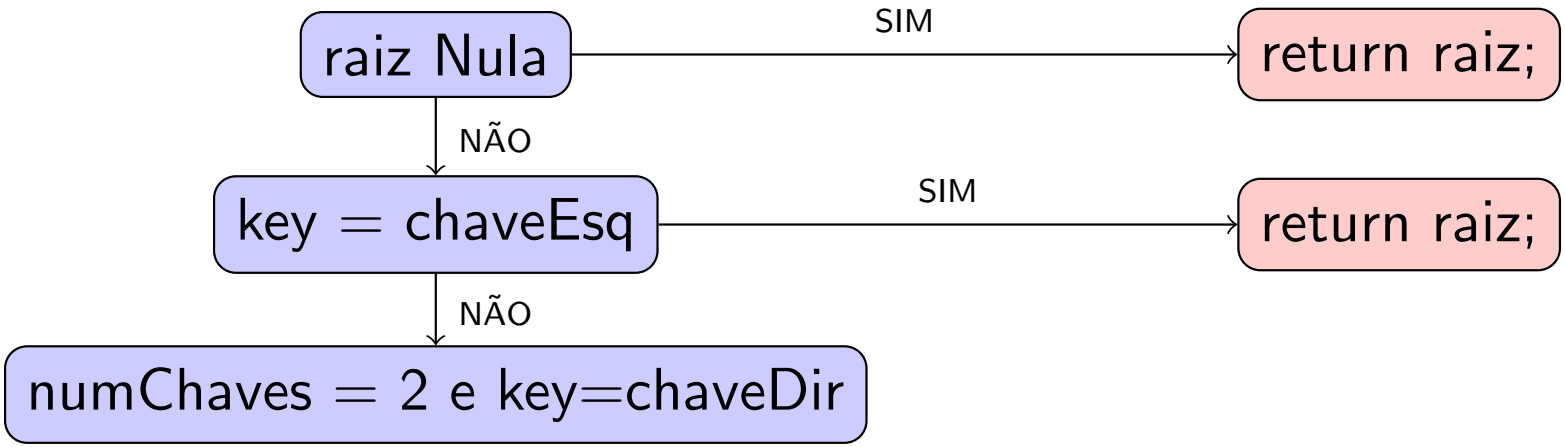
Busca(raiz, key)



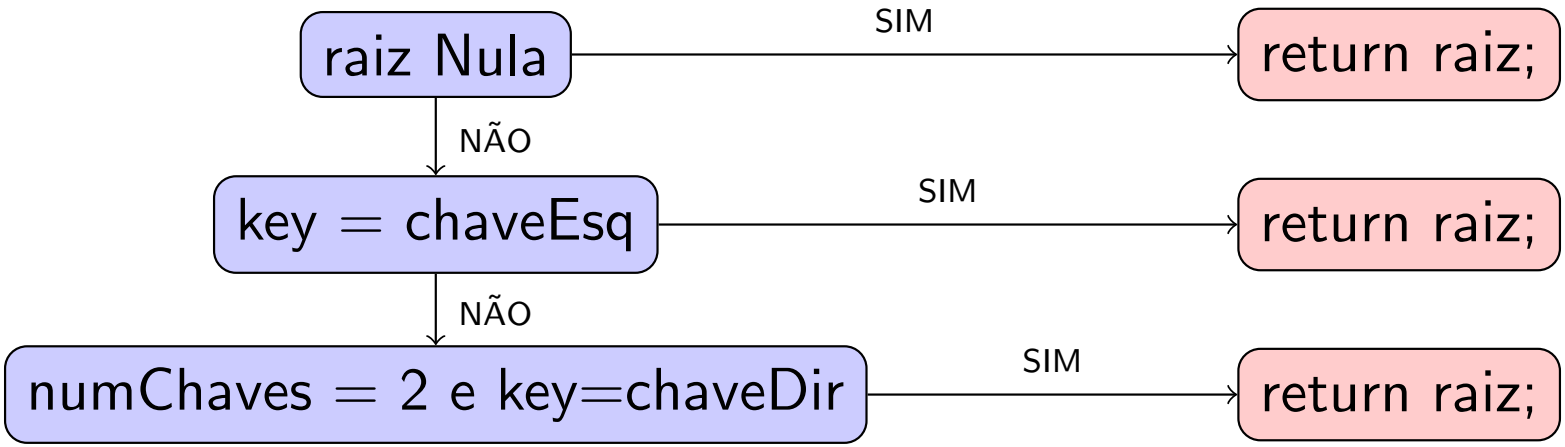
Busca(raiz, key)



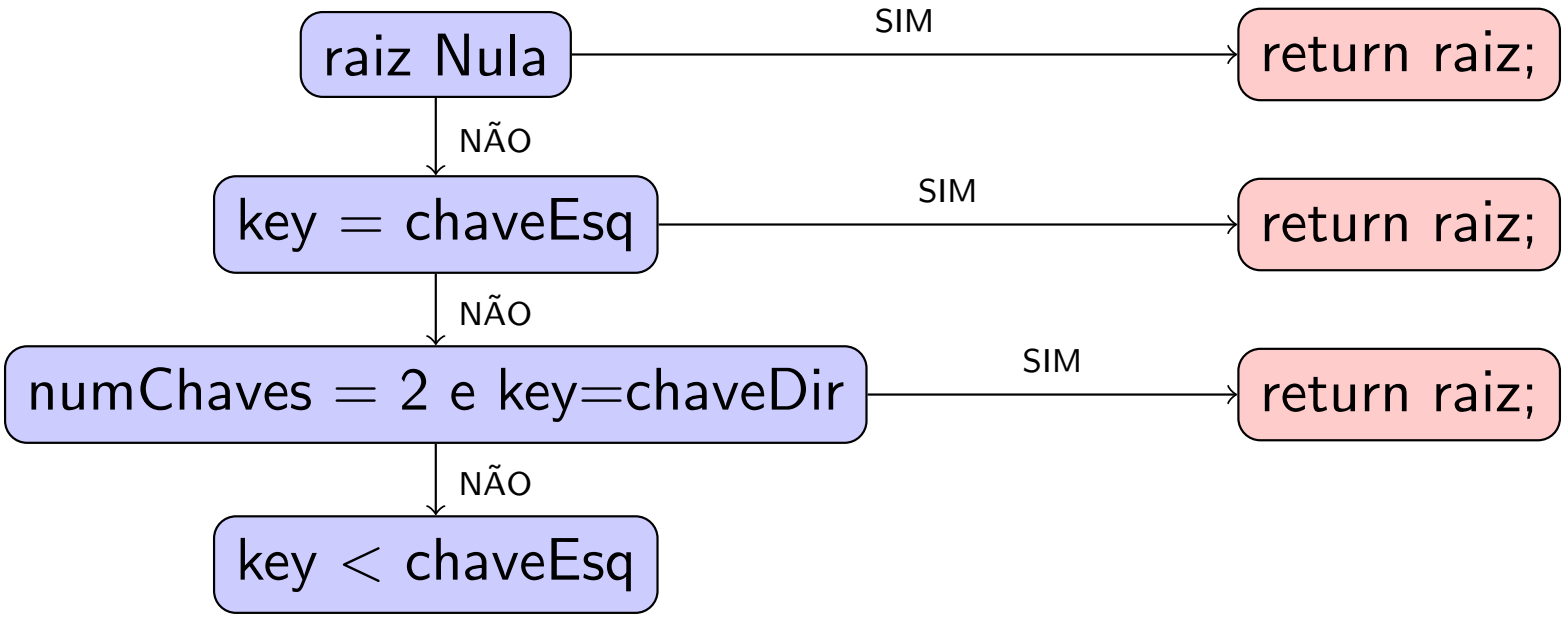
Busca(raiz, key)



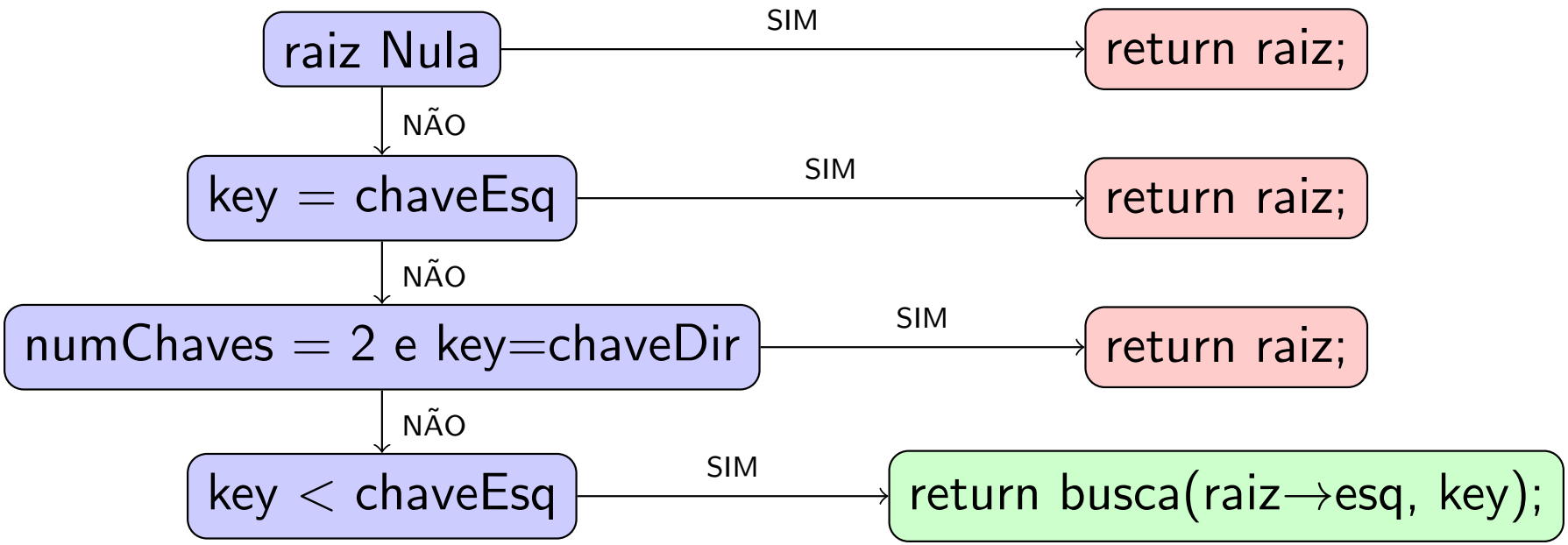
Busca(raiz, key)



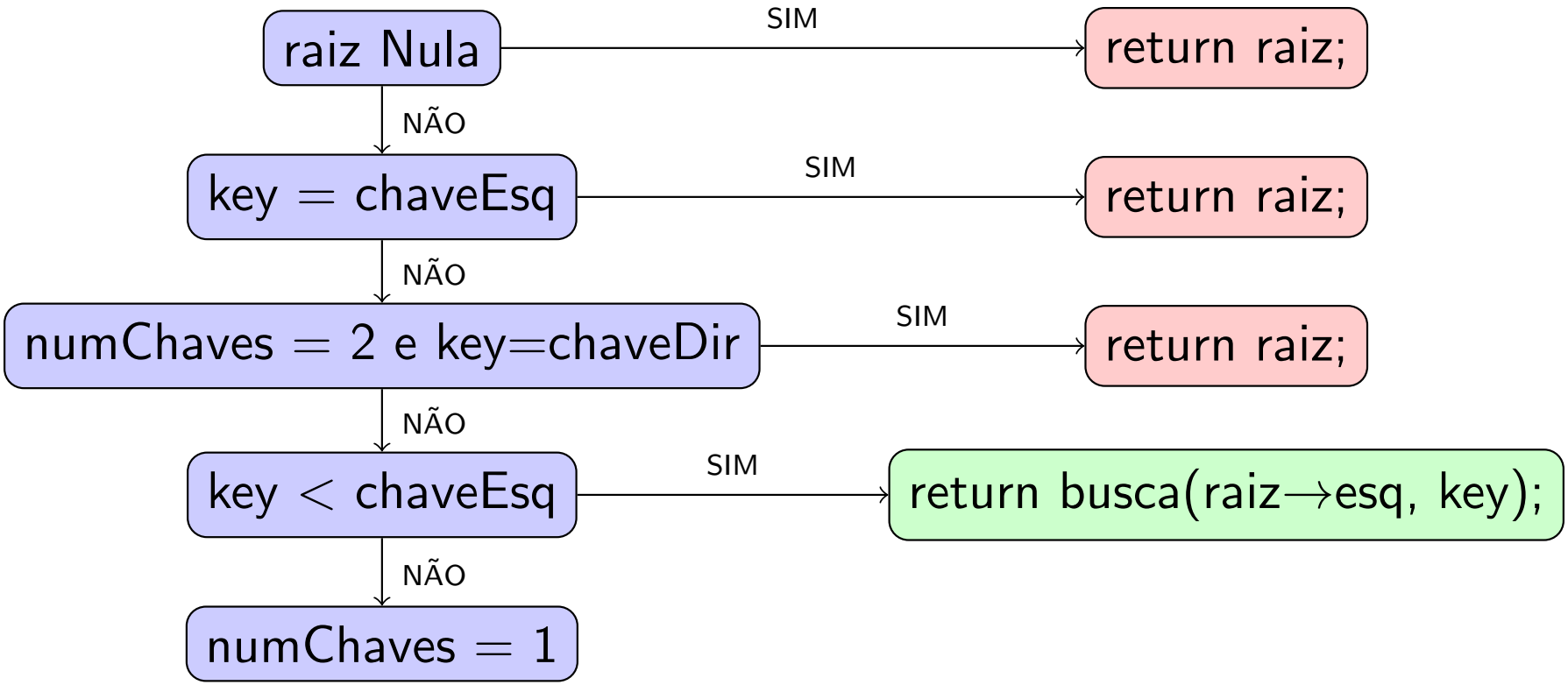
Busca(raiz, key)



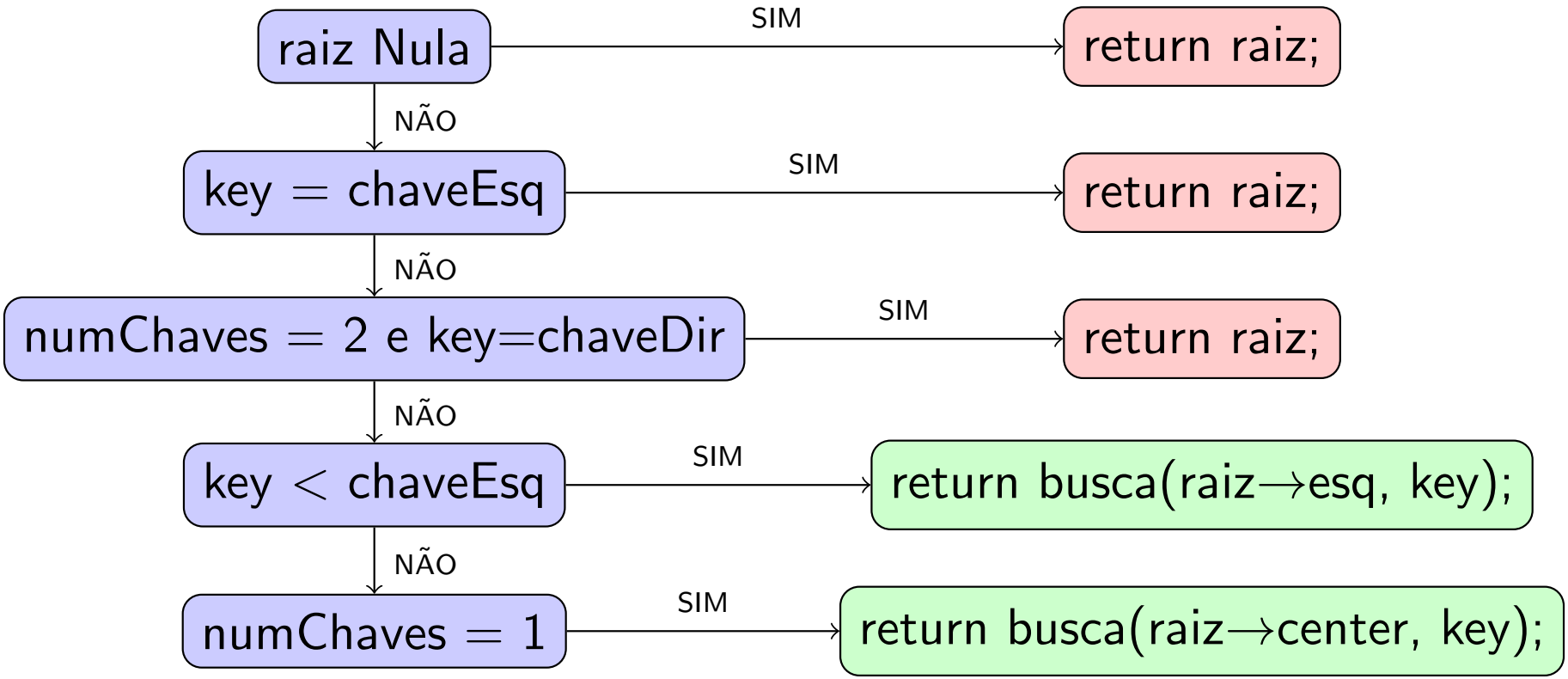
Busca(raiz, key)



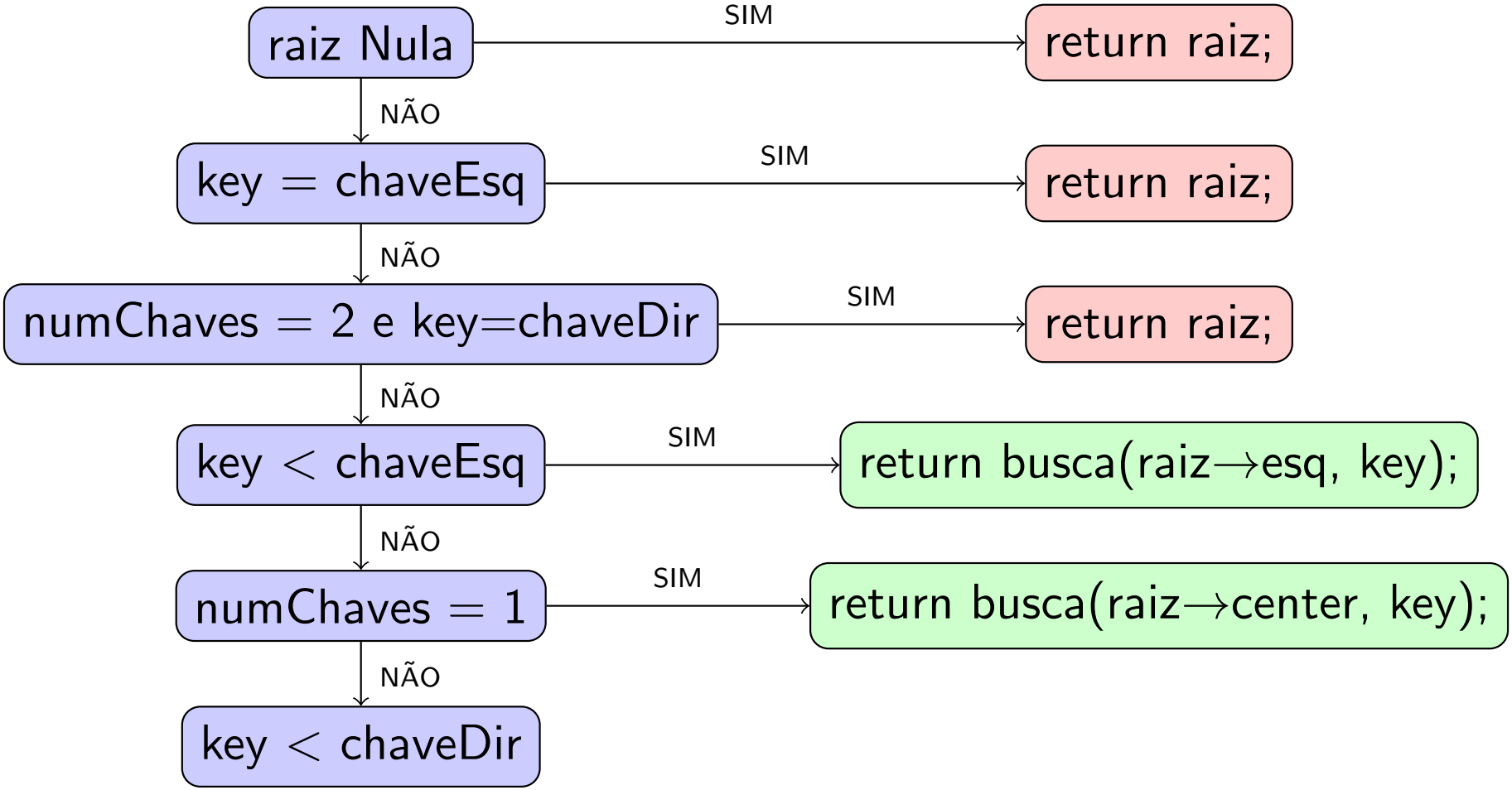
Busca(raiz, key)



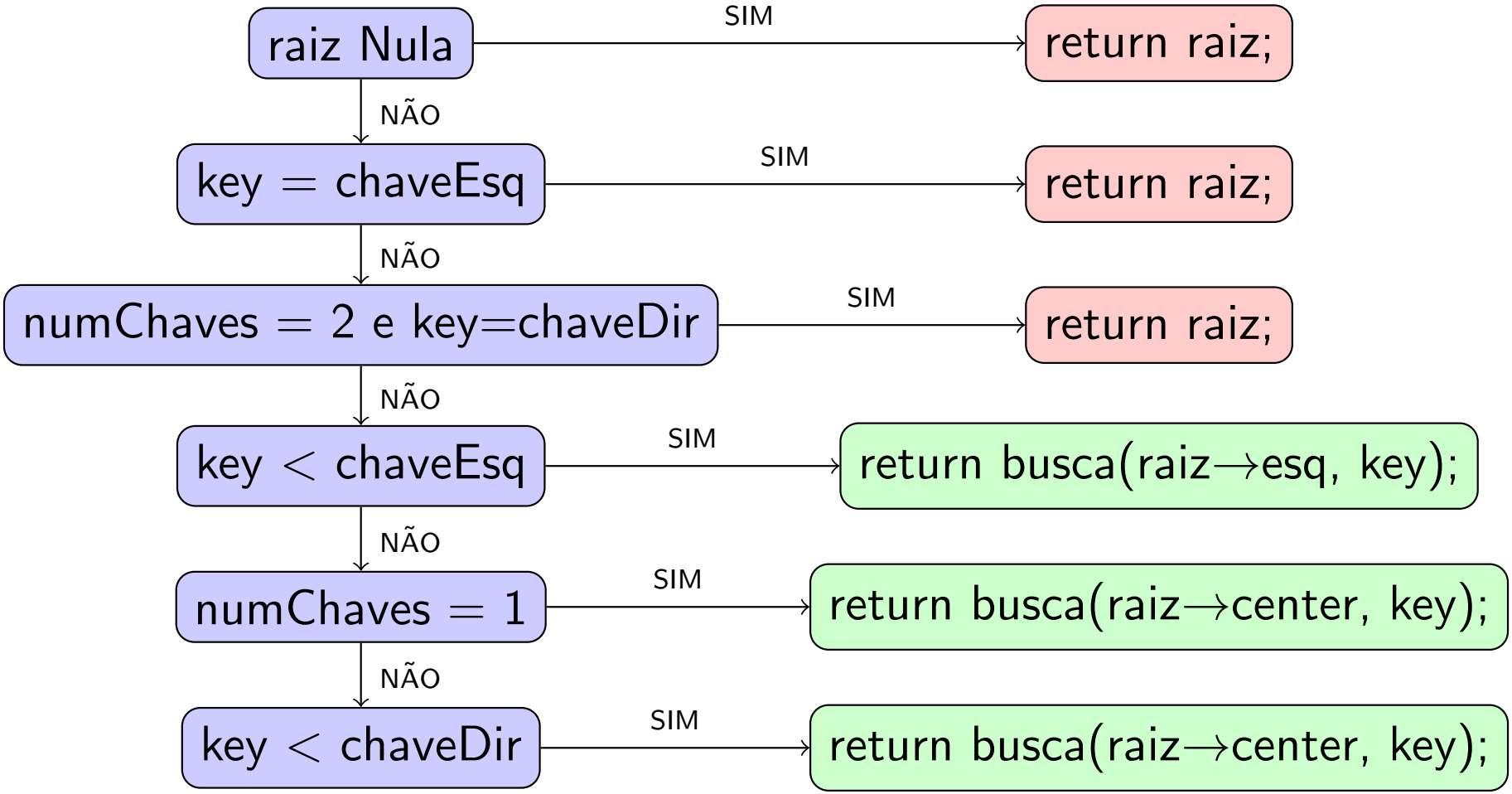
Busca(raiz, key)



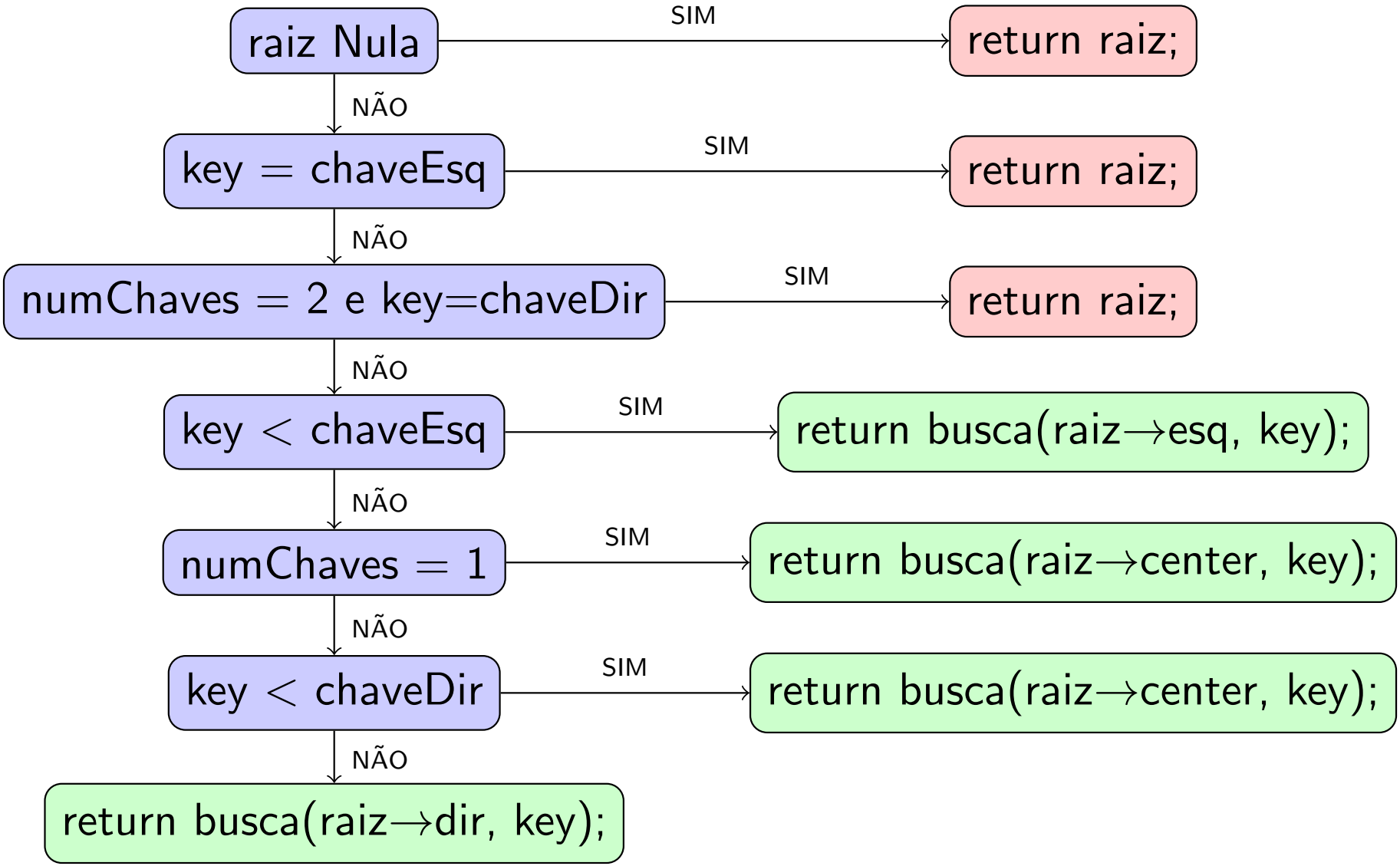
Busca(raiz, key)



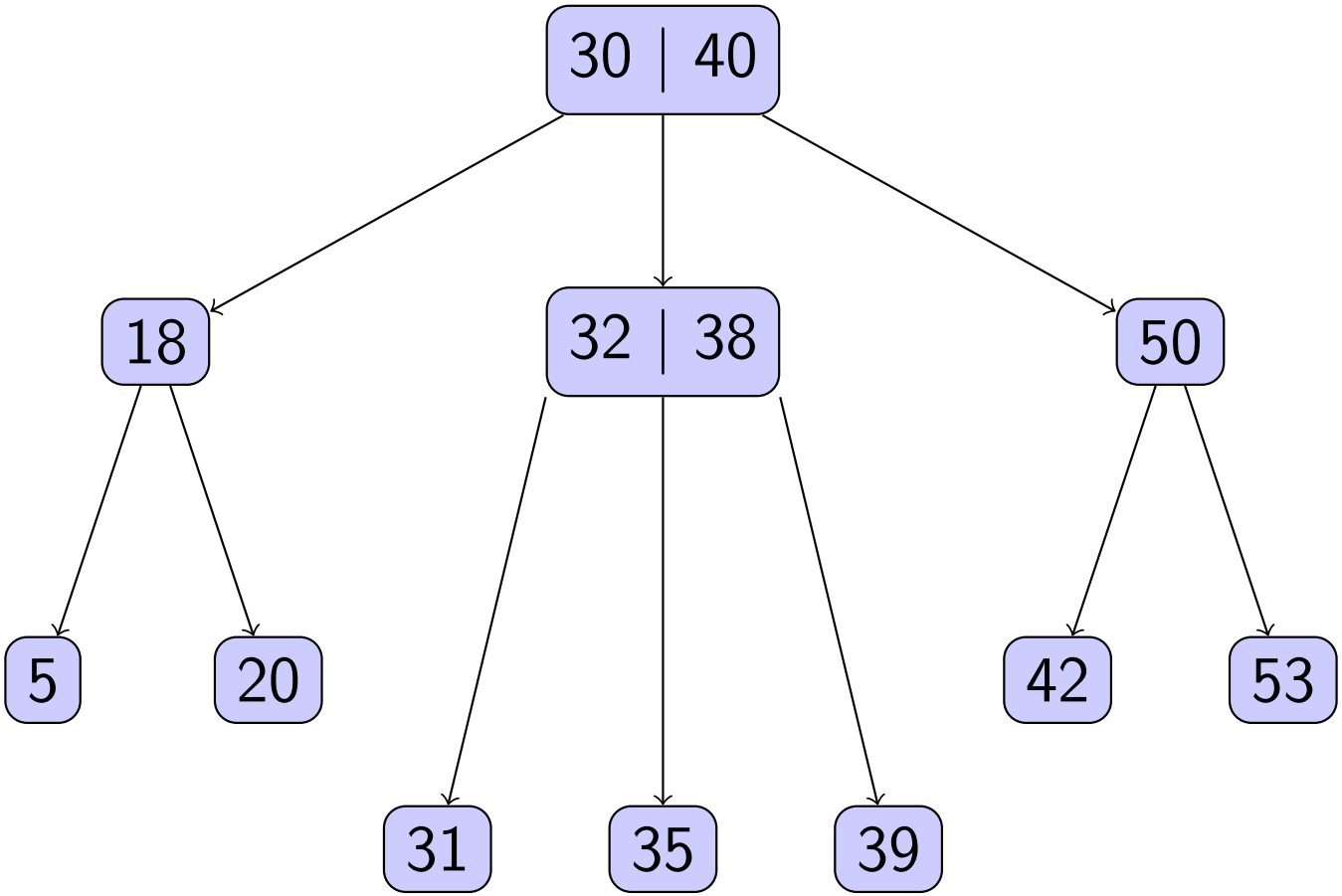
Busca(raiz, key)



Busca(raiz, key)

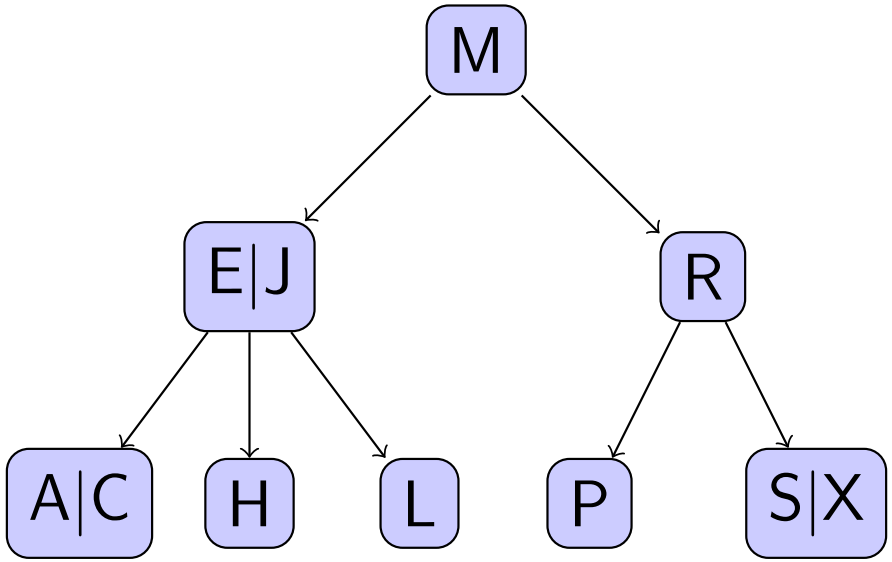


Exemplo



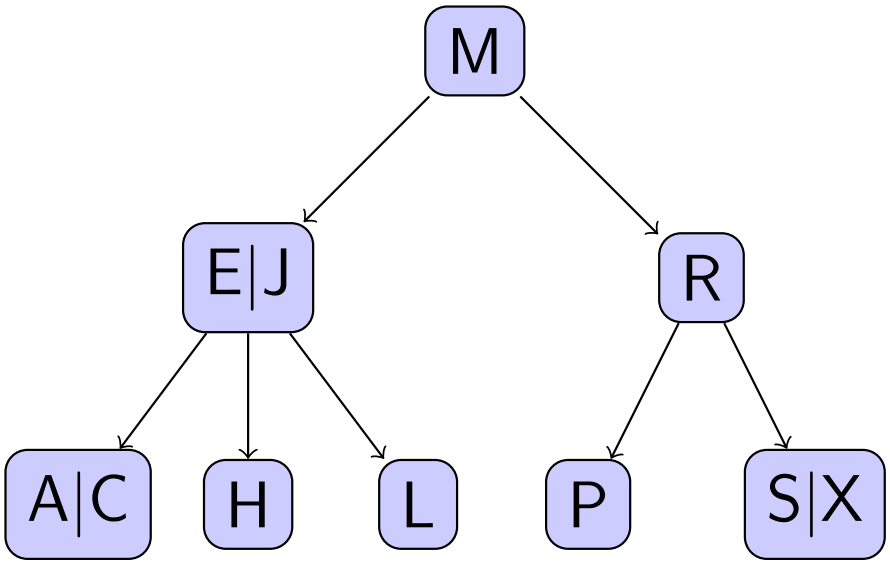
Inserção (a)

Inserção em um nó simples:
(Inserir o elemento K)



Inserção (a)

Inserção em um nó simples:
(Inserir o elemento K)



Inserção (b)

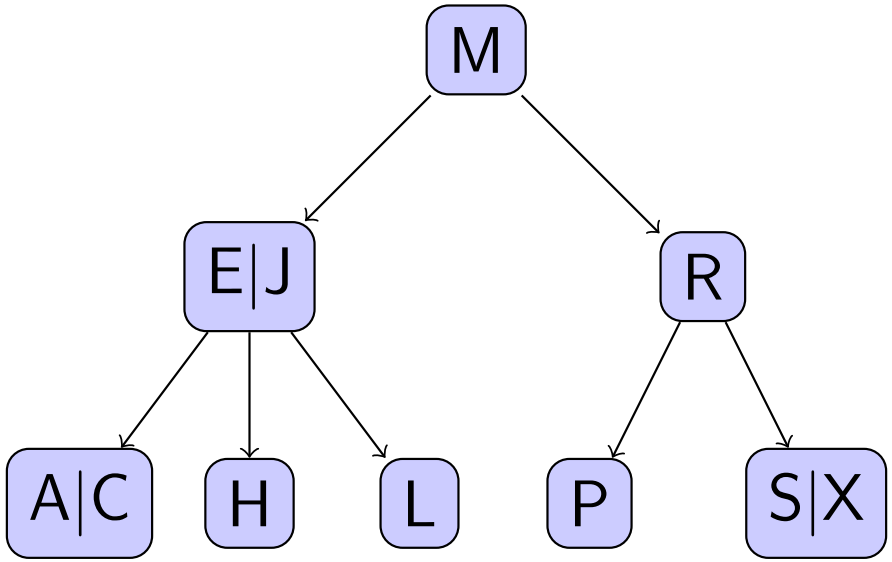
Inserção em um nó duplo isolado:
(Inserir o elemento S)

Inserção (b)

Inserção em um nó duplo isolado:
(Inserir o elemento S)

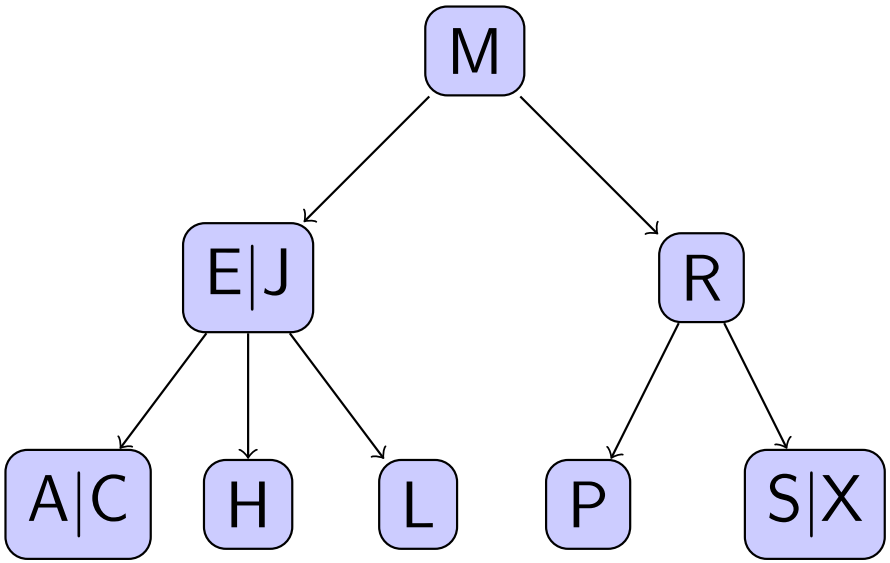
Inserção (c)

Inserção em um nó duplo com pai em um nó simples:
(Inserir o elemento Z)



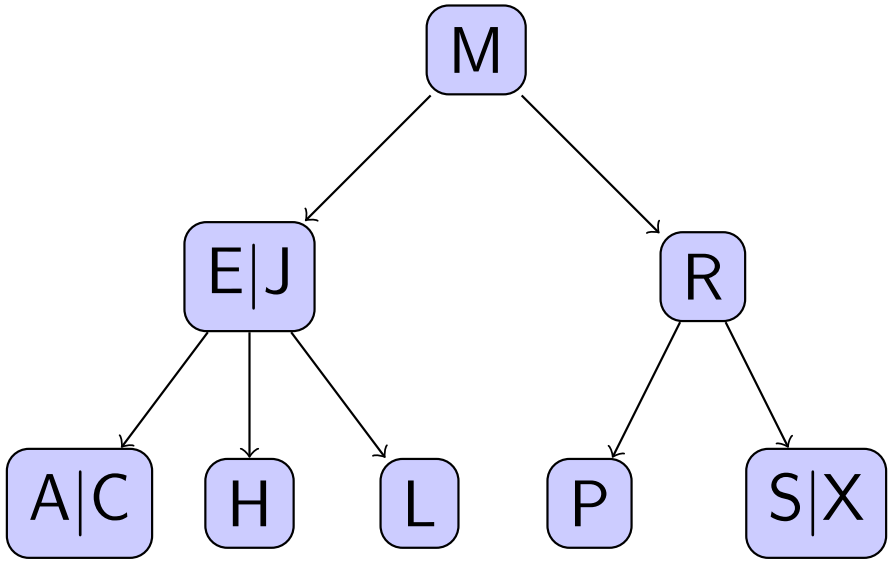
Inserção (c)

Inserção em um nó duplo com pai em um nó simples:
(Inserir o elemento Z)



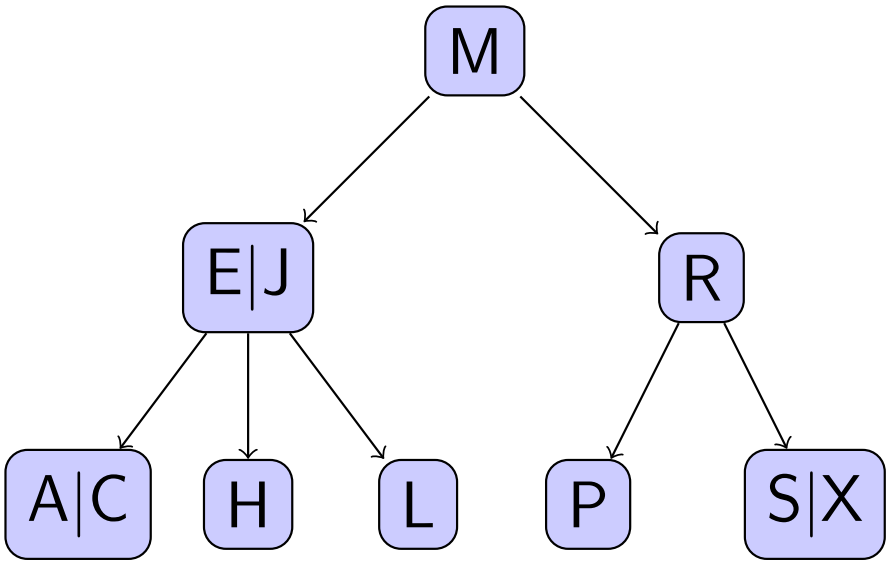
Inserção (d)

Inserção em um nó duplo com pai em um nó duplo:
(Inserir o elemento D)



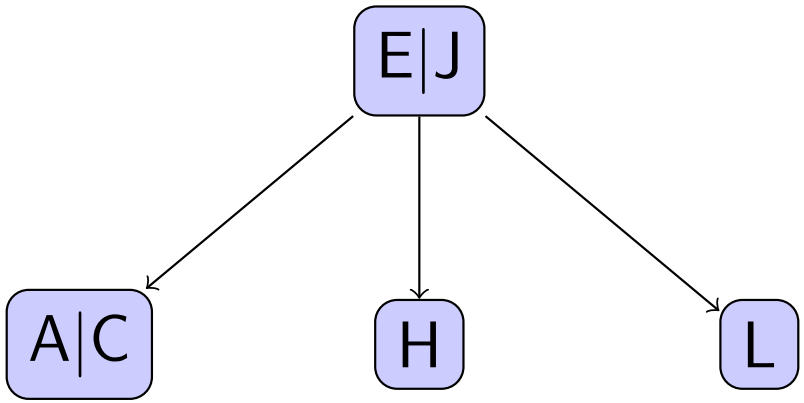
Inserção (d)

Inserção em um nó duplo com pai em um nó duplo:
(Inserir o elemento D)



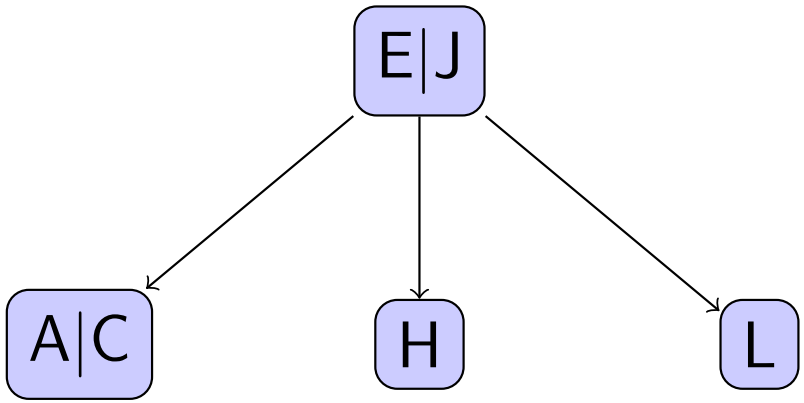
Inserção (e)

Inserção em um nó duplo cujo pai em um nó duplo, sucessivamente até a raiz: (Inserir o elemento D)



Inserção (e)

Inserção em um nó duplo cujo pai em um nó duplo, sucessivamente até a raiz: (Inserir o elemento D)



Observações:

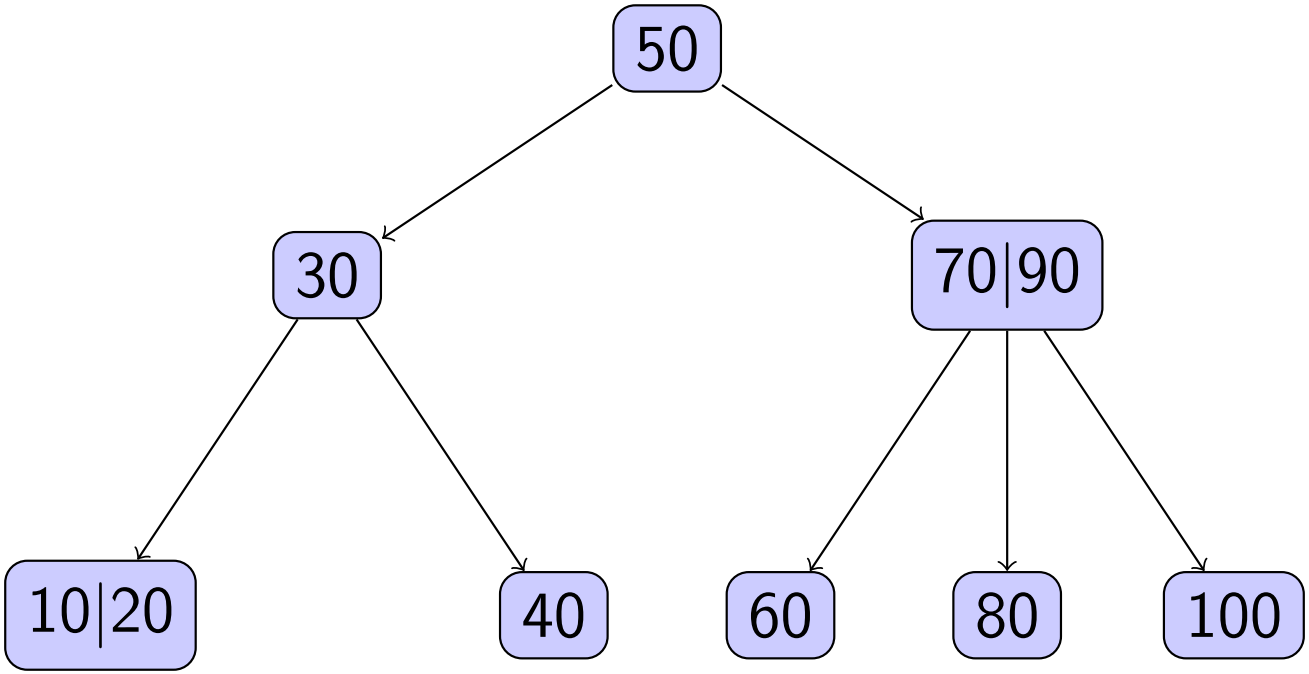
- As transformações envolvem modificações locais, portanto tem complexidade constante.
- Na pior hiótese, temos todos os pais como nós duplos, a complexidade seria portanto $O(h) = O(logn)$.
- As transformações preservam as propriedades globais da árvore. A árvore se mantem em ordem e balanceada.

Exercício: Insira, em ordem crescente, numa árvore 2-3 os elementos $L = \{1, 2, 3 \ldots 9\}$. Compare com a inserção em uma BST e uma AVL.

Exercício

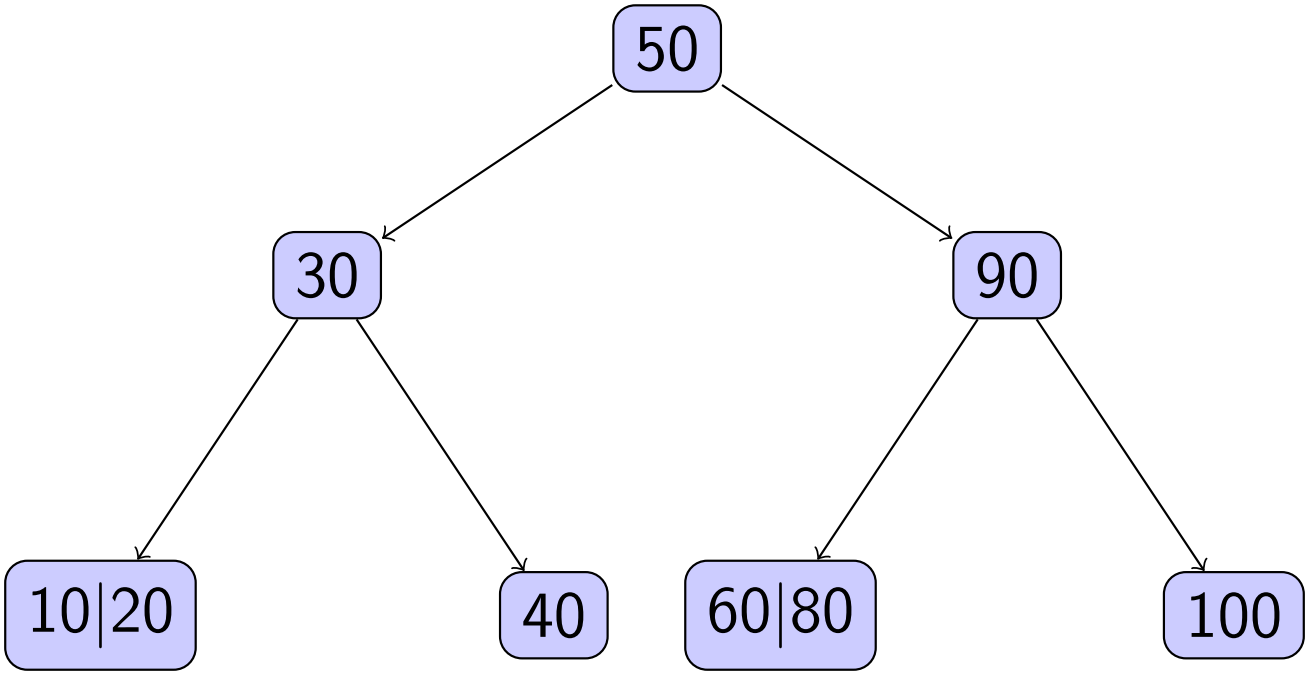
Remoção (a)

Remoçao de um nó duplo interno: (Remover o 70)



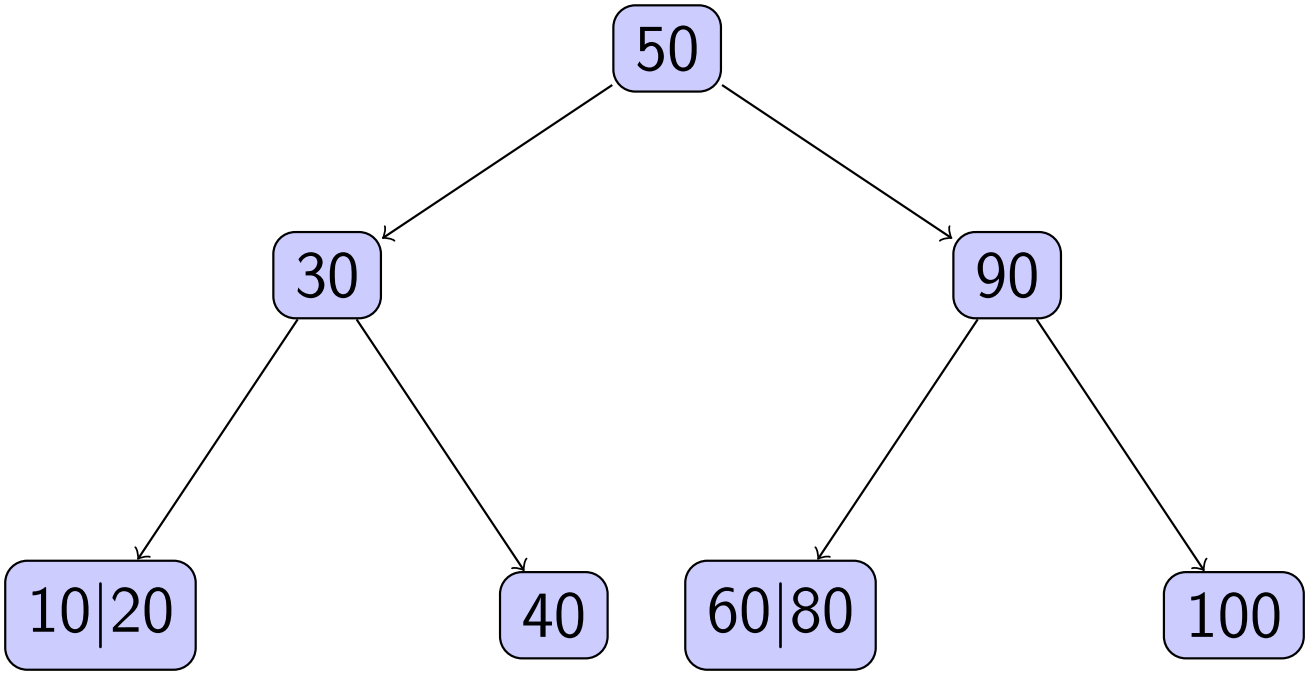
Remoção (a)

Remoçao de um nó duplo interno: (Remover o 70)



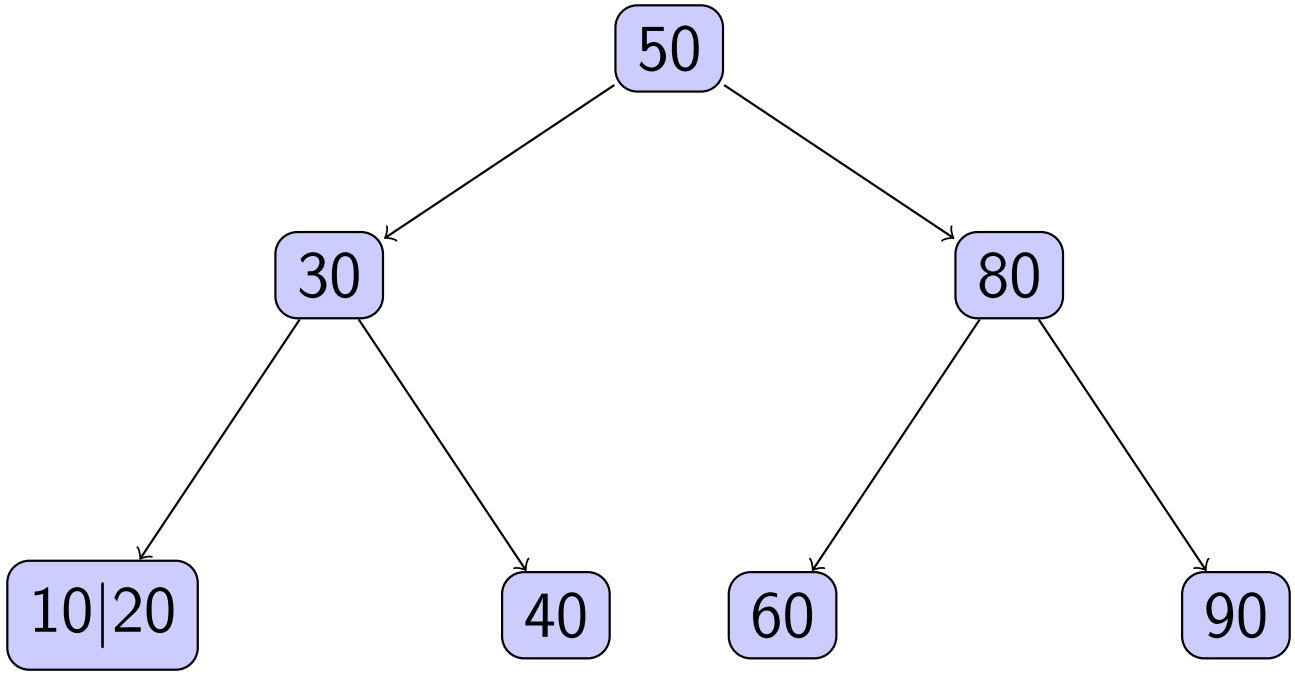
Remoção (b)

Remoçao de um nó folha: (Remover o 100)



Remoção (b)

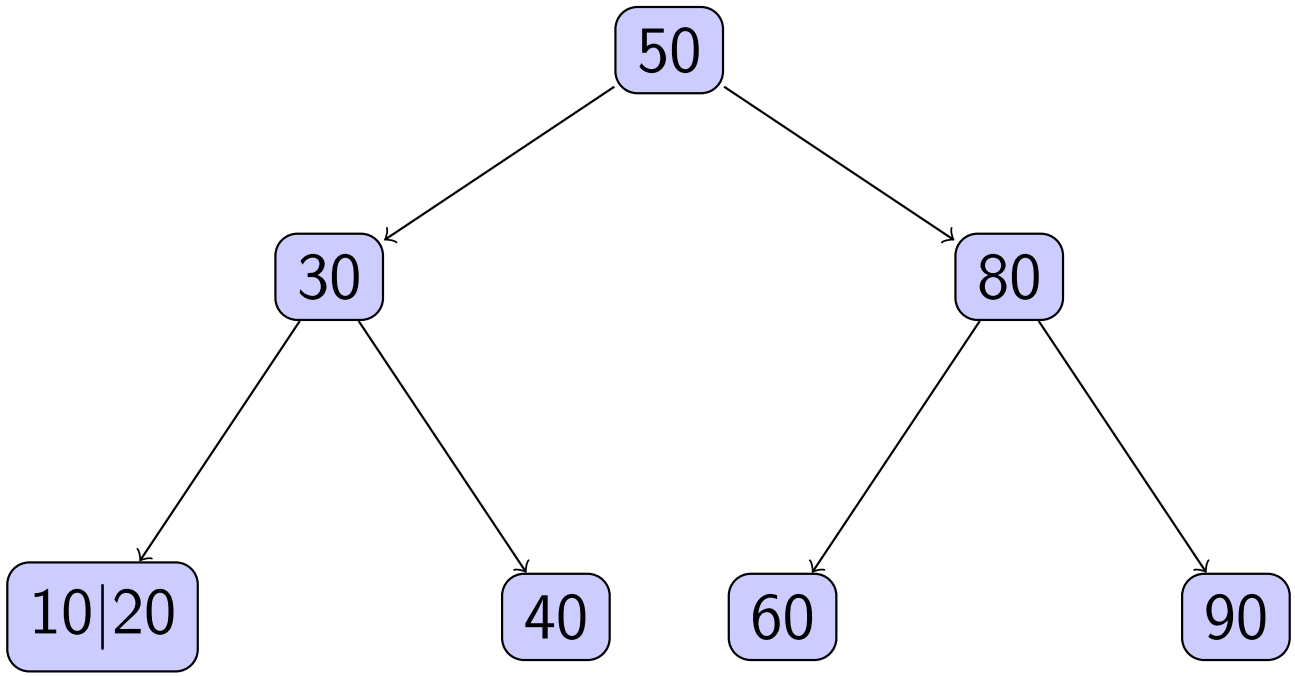
Remoçao de um nó folha: (Remover o 100)



Remoção (c)

Remoçao de um nó simples interno: (Remover o 80)

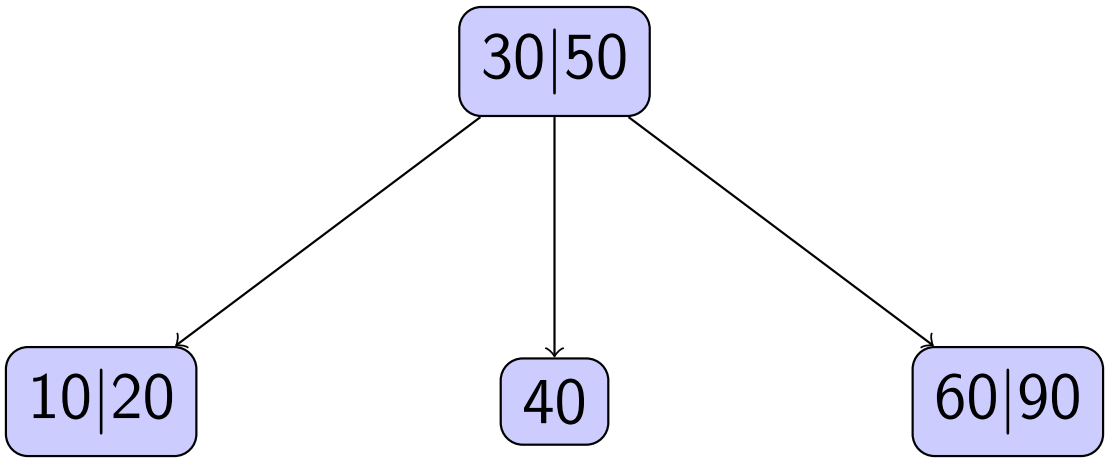
- Rebaixar os nós ancestrais;
- Processo contrário a promoção de um termo central.



Remoção (c)

Remoçao de um nó simples interno: (Remover o 80)

- Rebaixar os nós ancestrais;
- Processo contrário a promoção de um termo central.



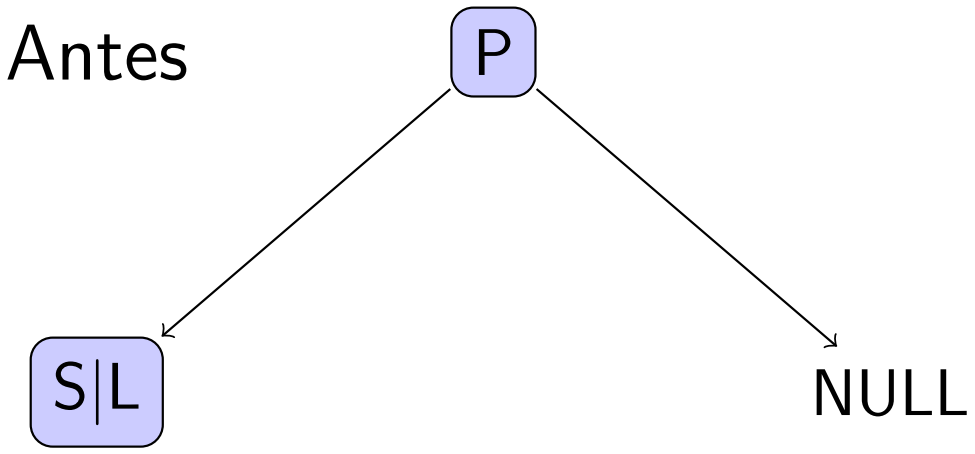
Remoção

Algoritmo: (Remover o elemento X)

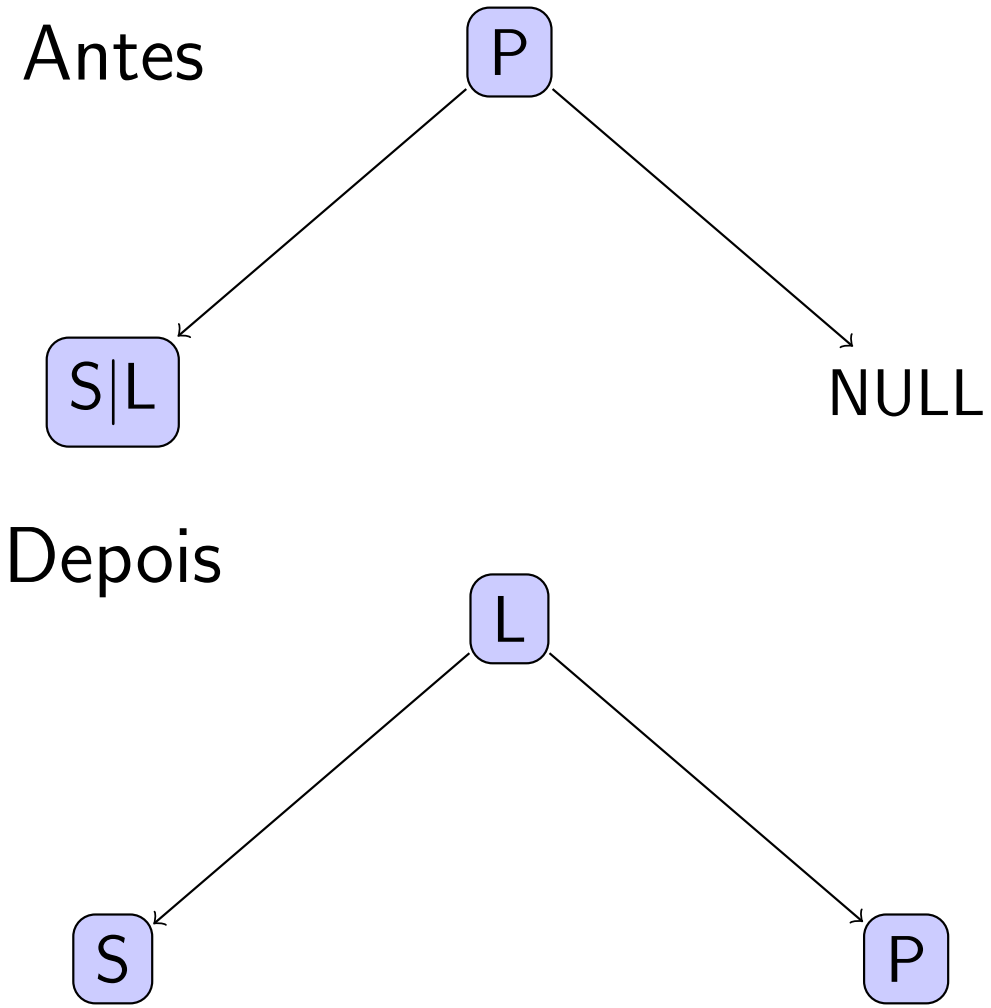
1. Localizar o nó N que contem o X;
2. Se não é folha:
   - Trocar X por seu sucessor;
   - Remoção **sempre** será pelas folhas;
3. Se N contem outro item, apague X, senão:
   - Tente redistribuir os nós irmãos;
   - Caso contrário, proceda com a união dos irmãos.

Redistribuição

Redistribuição

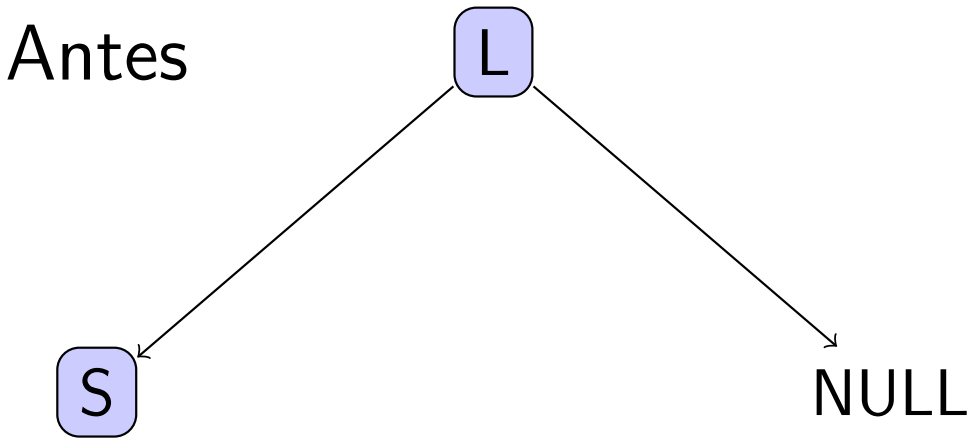


Redistribuição

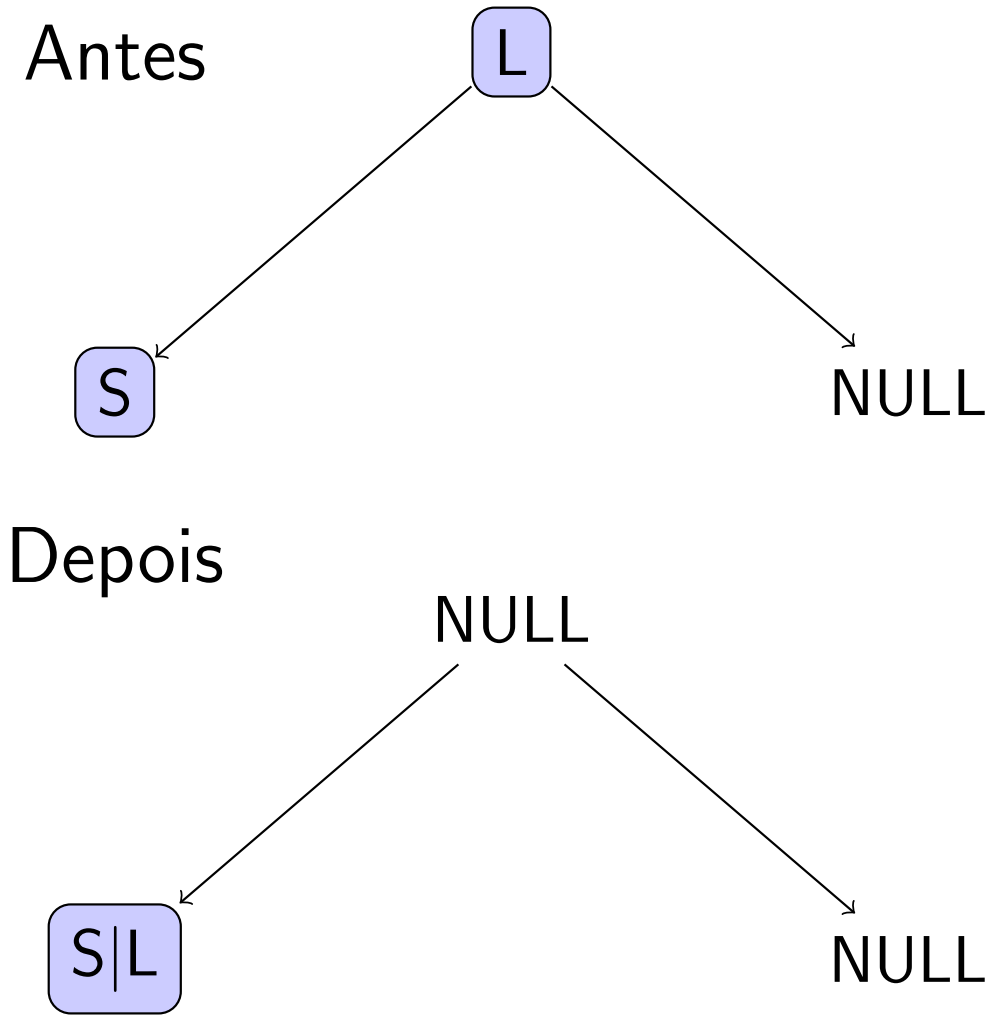


União

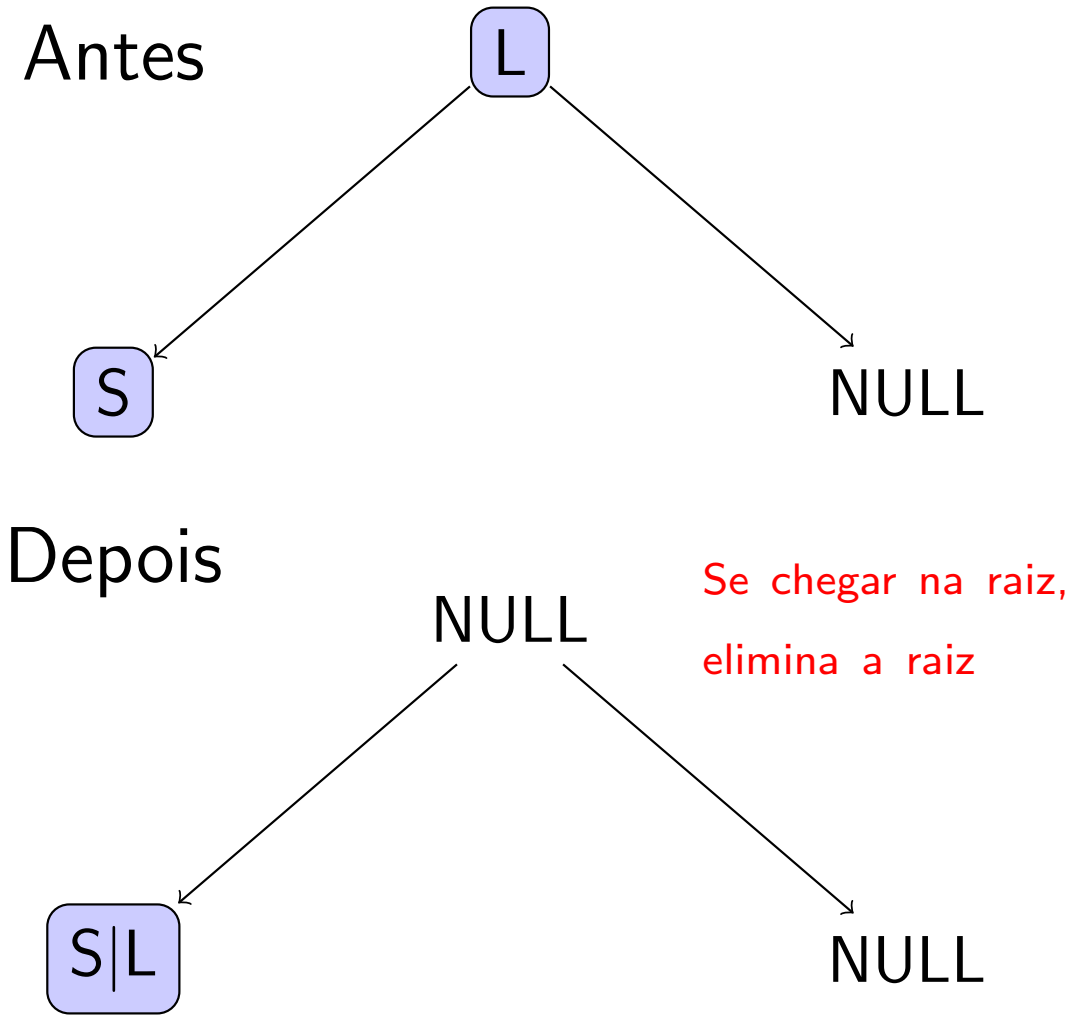
União



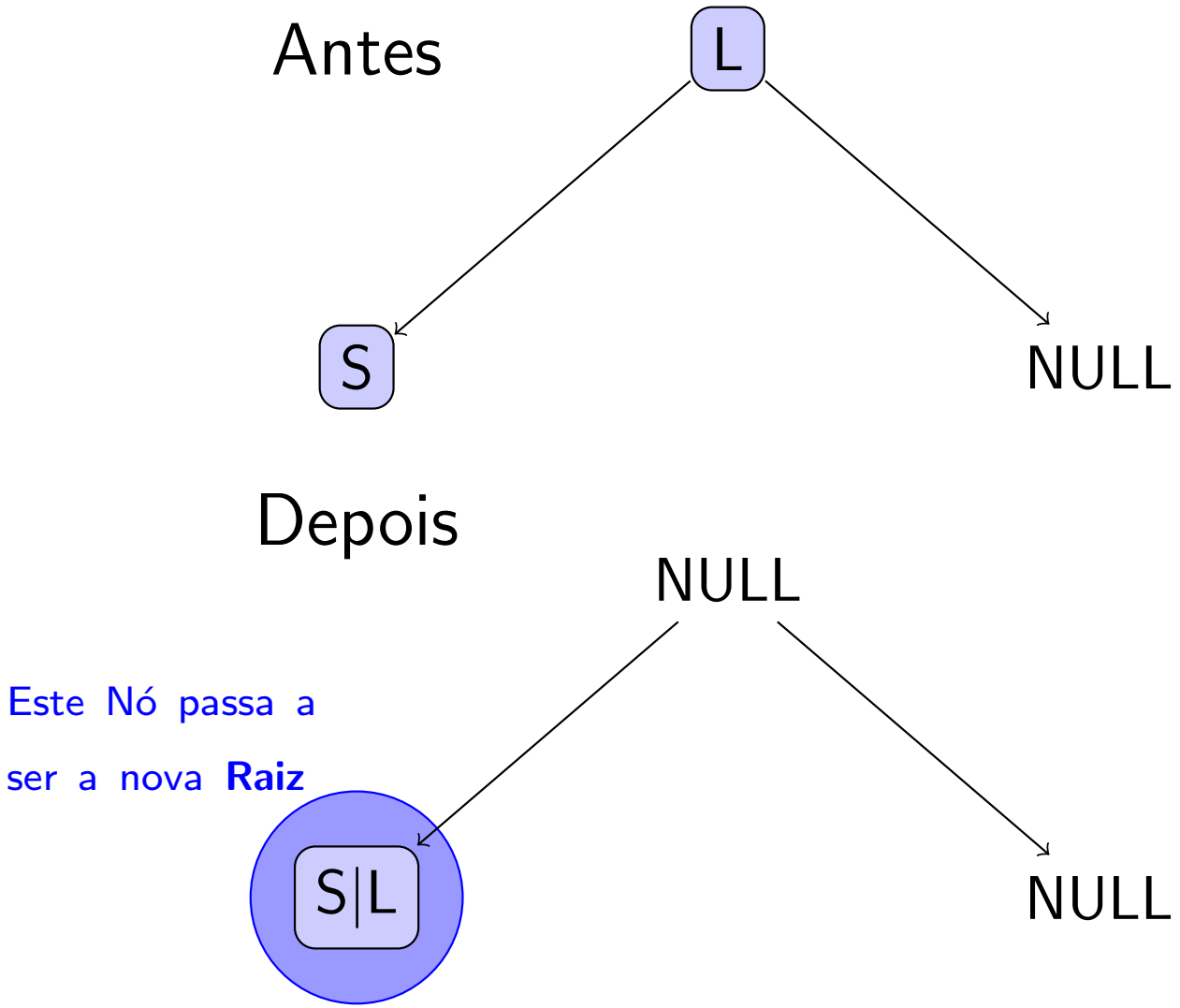
União



União


Antes
L
S
NULL
Depois
NULL
S|L
NULL
Se chegar na raiz,
elimina a raiz

União


Antes
L
S
NULL
Depois
NULL
Este Nó passa a
ser a nova **Raiz**
S|L
NULL