Árvore de Pesquisa Balanceada

Árvore 2-3

Introdução

Árvore binária de pesquisa

- Cada nodo guarda
  - uma chave (elemento), e
  - duas ligações, uma para o seu filho esquerdo e a outra para o seu filho direito

Árvore binária de pesquisa balanceada

- Para cada nodo, as alturas das suas subárvores (esquerda e direita) têm valores muito próximos
- As operações de pesquisa e de inserção têm ordens de complexidade logarítmica
- Quanto mais balanceada estiver a árvore, mais rápido é a pesquisa e a inserção de um elemento nessa árvore

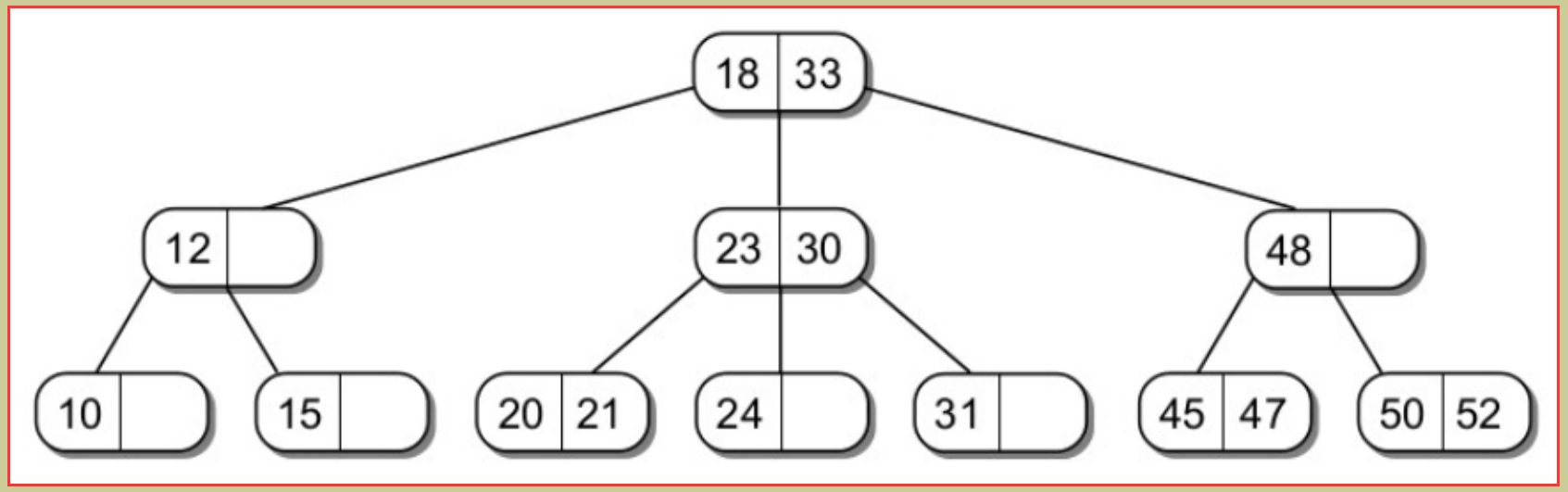
Uma árvore 2-3

- É uma estrutura de dados alternativa que permite a pesquisa e a inserção de um elemento de modo eficiente
- É uma árvore em que cada nodo pode ter até três filhos (árvore ternária)
- É uma árvore de pesquisa que está sempre balanceada

Definição

É uma árvore de pesquisa em que

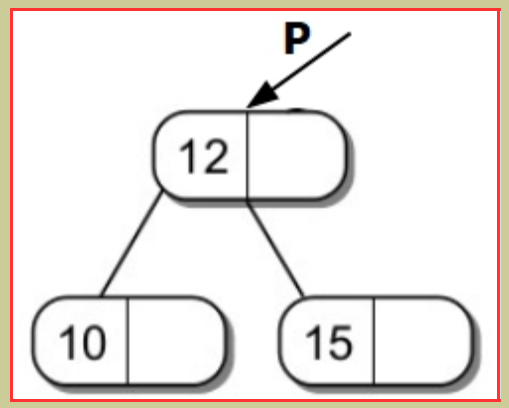
- Cada nodo pode ter uma ou duas chaves
- Todos os nodos folhas (sem filhos) estão no mesmo nível
- Cada nodo que não seja folha (**nodo interno**) deve ter dois ou três filhos
  - se o nodo tem **1** chave, então deve ter **2** filhos
  - se o nodo tem **2** chaves, então deve ter **3** filhos
- Não há nodos com apenas 1 filho



Propriedades

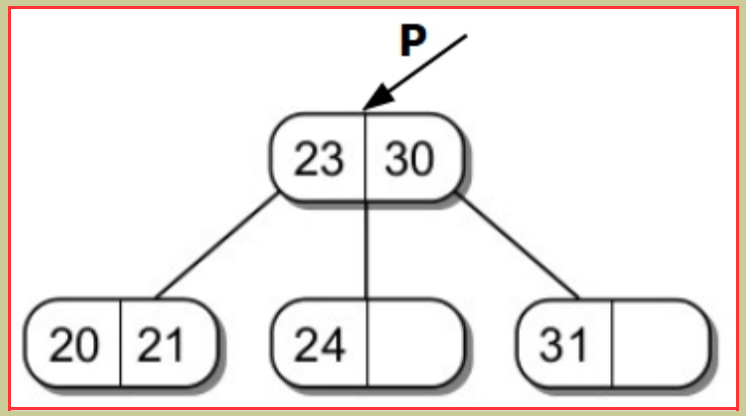
Mantêm as propriedades das árvores binárias de pesquisa

- Para cada nodo P com **1 chave** e, portanto, **2** filhos
  - todos os nodos com chaves menores do que a **chave** de P são guardados na subárvore esquerda de P
  - todos os nodos com chaves maiores do que a **chave** de P são guardadas na subárvore direita de P



Mantêm as propriedades das árvores binárias de pesquisa

- Para cada nodo P com **2 chaves** e, portanto, **3** filhos
  - a primeira chave é menor do que a segunda
  - todos os nodos com chave menor que a primeira chave são guardados na subárvore esquerda de P
  - todos os nodos com chave maior que a primeira chave e menor que a segunda chave são guardadas na subárvore do meio de P
  - todos os nodos com chave maior que a segunda chave são guardados na subárvore direira de P



Operações sobre uma Árvore 2-3

Operações básicas sobre um nodo

- Criar um nodo de uma Árvore 2-3
- Libertar um nodo de uma Árvore 2-3

Operações básicas sobre a árvore

- Criar uma Árvore 2-3 (vazia)
- Libertar/destruir uma Árvore 2-3
- Verificar se uma Árvore 2-3 está vazia
- Determinar o número de elementos de uma Árvore 2-3
- Determinar a altura de uma Árvore 2-3
- Mostrar os elementos de uma Árvore 2-3
- Pesquisar um elemento numa Árvore 2-3
- Inserir um elemento numa Árvore 2-3
- Remover um elemento de uma Árvore 2-3

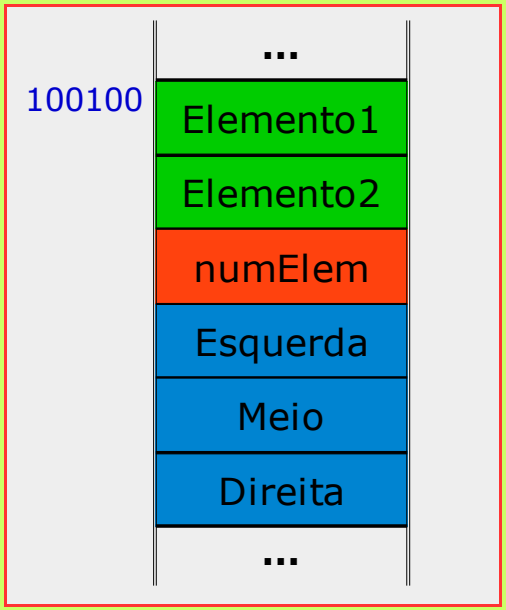
Nodos e ligações

Estrutura

- Um nodo pode ser representado por uma estrutura com os seguintes dados
  - a **informação** que se pretende guardar (pode ser de um ou de dois **elementos**)
  - a **quantidade** de elementos guardados (pode ser um ou dois)
  - as **ligações** a outros dois ou três nodos, correspondentes à
    - raiz da **subárvore esquerda**
    - raiz da **subárvore do meio** (que corresponde à subárvore direita do 1º elemento)
    - raiz da **subárvore direita**

Uma definição possível

```
struct  NodoAP23{
    INFOAP23  Elemento1;  // chave 1
    INFOAP23  Elemento2;  // chave 2
    int    numElem;  // quantidade de elementos
    struct NodoAP23  *Esquerda;
    struct NodoAP23  *Meio;
    struct NodoAP23  *Direita;
};
typedef  struct NodoAP23  *PNodoAP23;
```



Pesquisar um elemento (através da sua chave)

Método

- A pesquisa de um elemento em árvores 2-3 é semelhante à pesquisa em ABP
  - inicia-se na raiz, e
  - continua pelas subárvores devidas, em função dos valores das chaves dos elementos dos nodos e do elemento a procurar
- A única diferença é quando um nodo tem dois elementos
  - é necessário comparar o elemento a pesquisar com os dois elementos daquele nodo, para escolher entre as três possíveis subárvores para qual é que continua a pesquisa
- Tal como numa ABP,
  - uma pesquisa **com sucesso** conduz o processo até um nodo com elemento igual ao procurado
  - uma pesquisa **sem sucesso** conduz a pesquisa a uma árvore vazia (NULL), que corresponde a um filho de uma folha

Algoritmo

```
algoritmo pesquisar (k, T)  { considerando um nodo com dois elementos }
  se (k = T(chave1) ou k = T(chave2)) então
     PARAR (encontrado um nodo com chave igual a k)
  fim_se
  se (k < chave1) então
     pesquisar (k, filho esquerdo de T)
  senão
     se (k > T(chave1) e k < T(chave2)) então
        pesquisar (k, filho do meio de T)
     senão  { k > T(chave2) }
        pesquisar (k, filho direito de T)
     fim_se
  fim_se
fim_algoritmo
```

Inserir um elemento

Método

- O método de inserção é parecido com o método de inserção das árvores binárias, mas com algumas adaptações

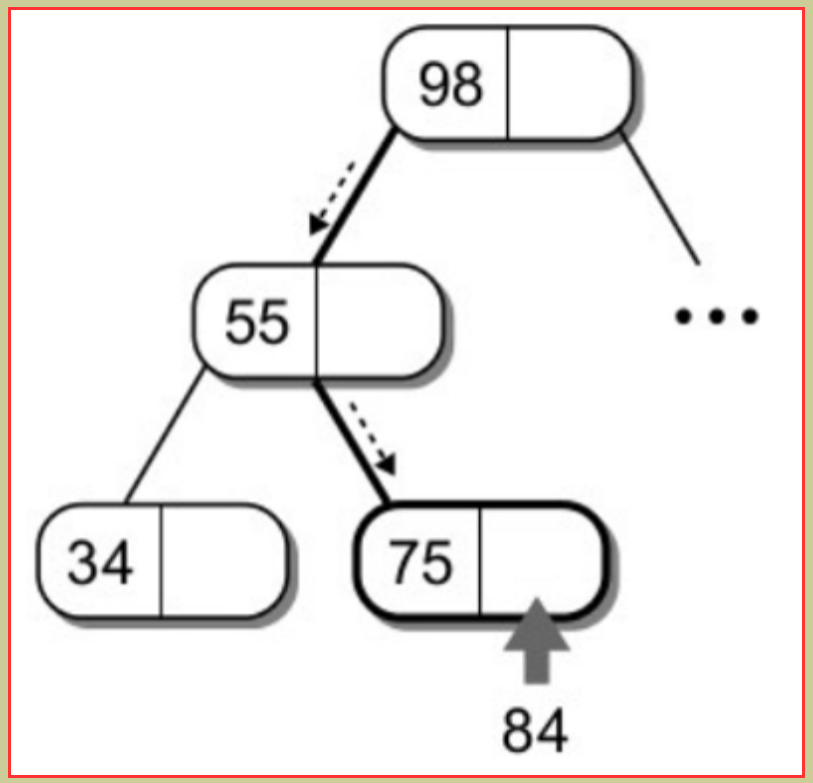
Passo 1: pesquisar o elemento (chave) na árvore

  - a pesquisa de um elemento não existente na árvore (elemento com uma nova chave) leva a uma subárvore vazia (NULL), que corresponde ao filho de um nodo folha

Método

Passo 2: verificar se existe espaço no nodo folha encontrada antes
se o nodo folha contém apenas um elemento, então inserir o novo elemento nesse nodo

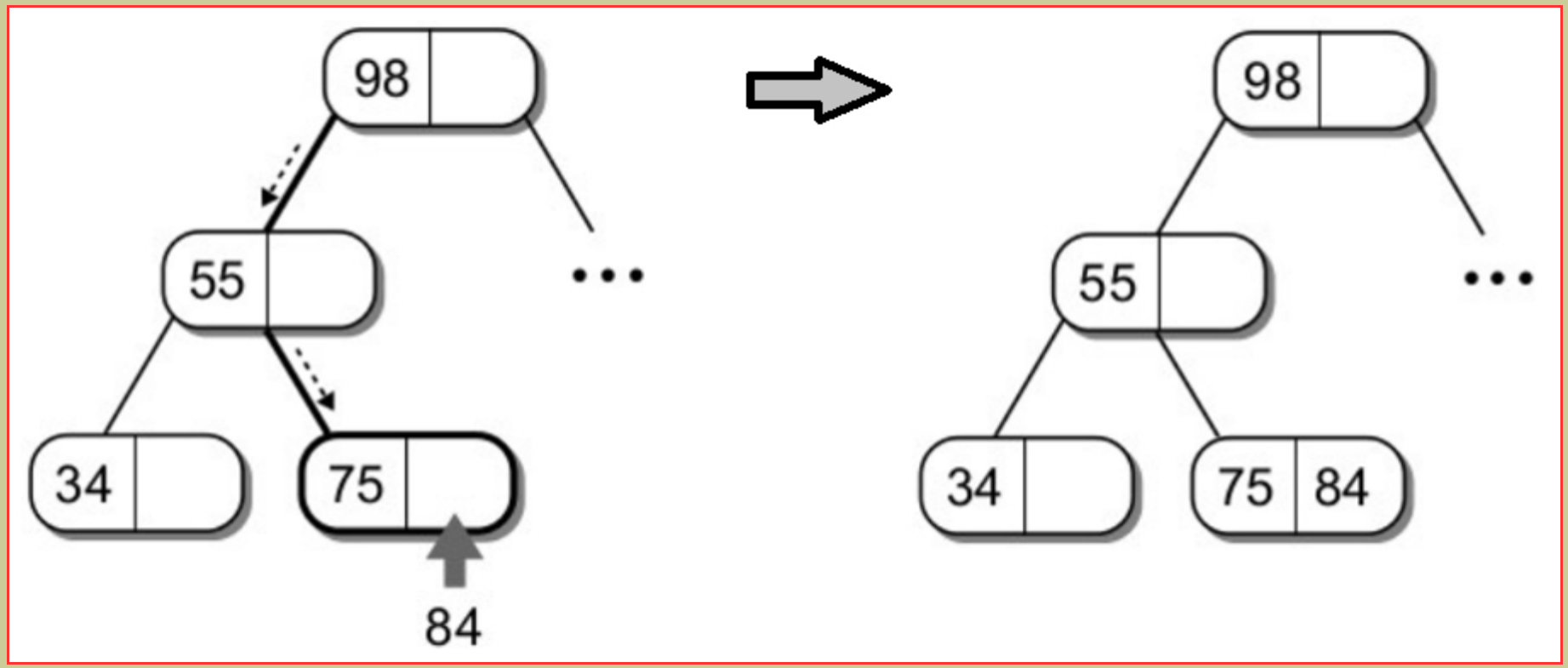
- caso 1: o elemento do nodo folha é menor do que o elemento a inserir
- exemplo: inserir o elemento com chave **84** na árvore em baixo



Método

Passo 2: verificar se existe espaço no nodo folha encontrada antes
se o nodo folha contém apenas um elemento, então inserir o novo elemento nesse nodo

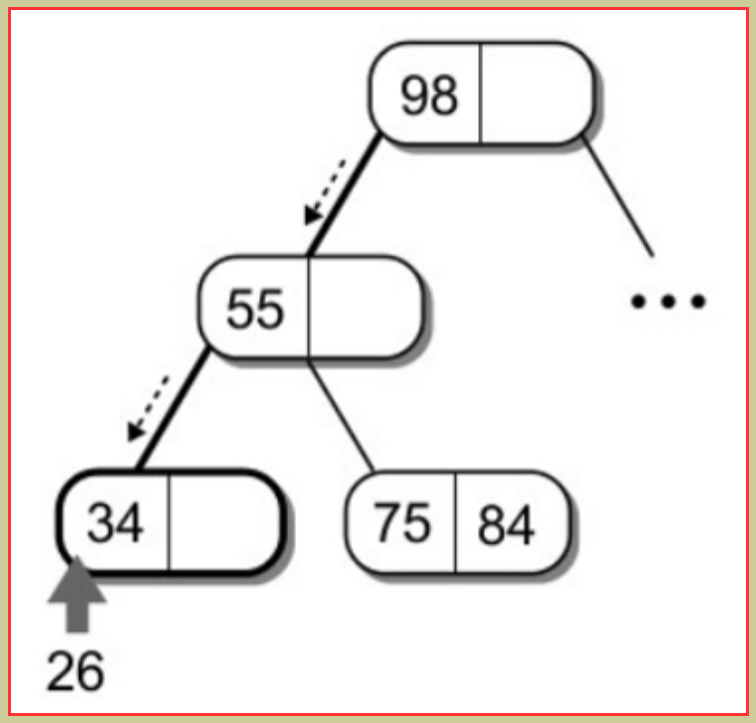
- caso 1: o elemento do nodo folha é menor do que o elemento a inserir
- exemplo: inserir o elemento com chave **84** na árvore em baixo



Método

Passo 2: verificar se existe espaço no nodo folha encontrada antes
se o nodo folha contém apenas um elemento, então inserir o novo elemento nesse nodo

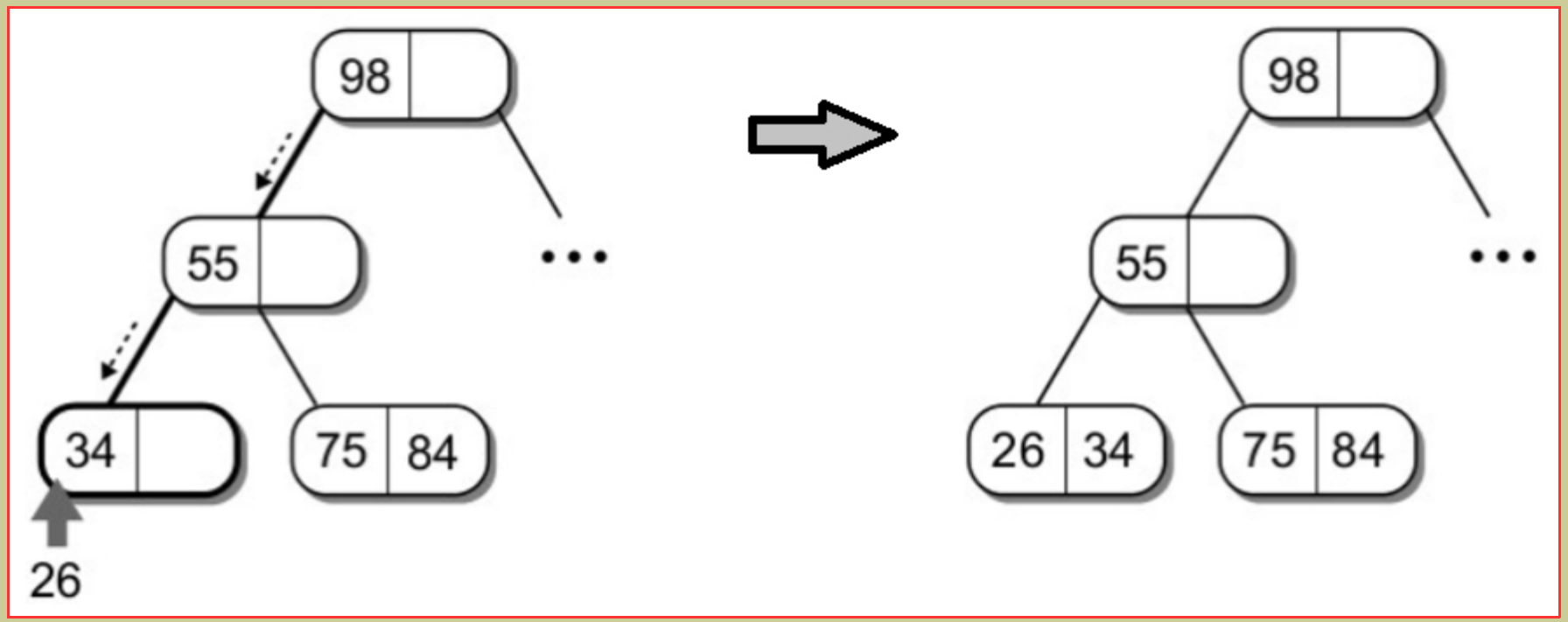
- caso 2: o elemento do nodo folha é maior do que o elemento a inserir
- exemplo: inserir o elemento com chave **26** na árvore em baixo



Método

Passo 2: verificar se existe espaço no nodo folha encontrada antes
se o nodo folha contém apenas um elemento, então inserir o novo elemento nesse nodo

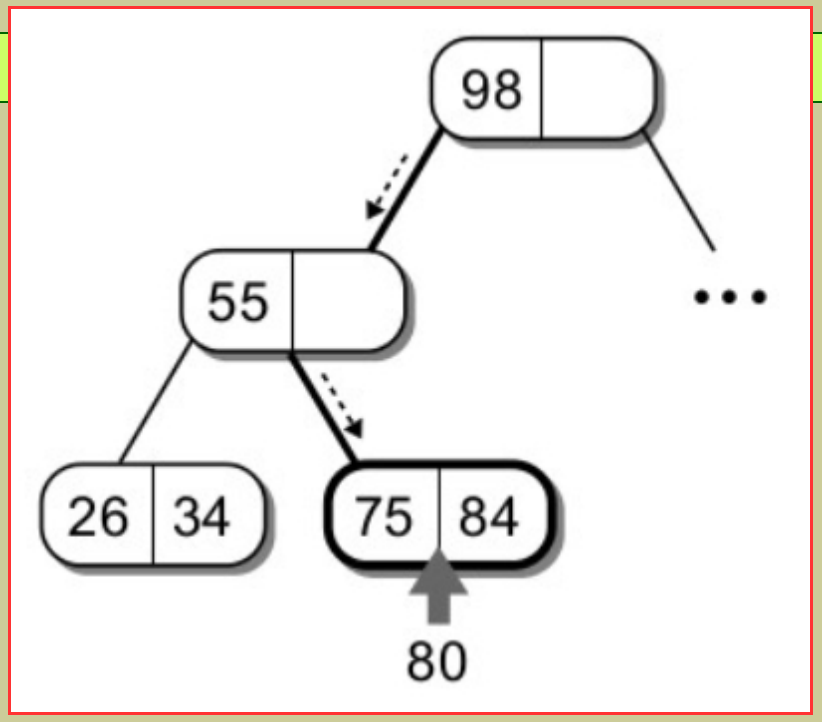
- caso 2: o elemento do nodo folha é maior do que o elemento a inserir
- exemplo: inserir o elemento com chave **26** na árvore em baixo



Método

Passo 2: verificar se existe espaço no nodo folha encontrada antes
se o nodo folha contém dois elementos, então ...

- exemplo: inserir o elemento com chave **80** na árvore em baixo

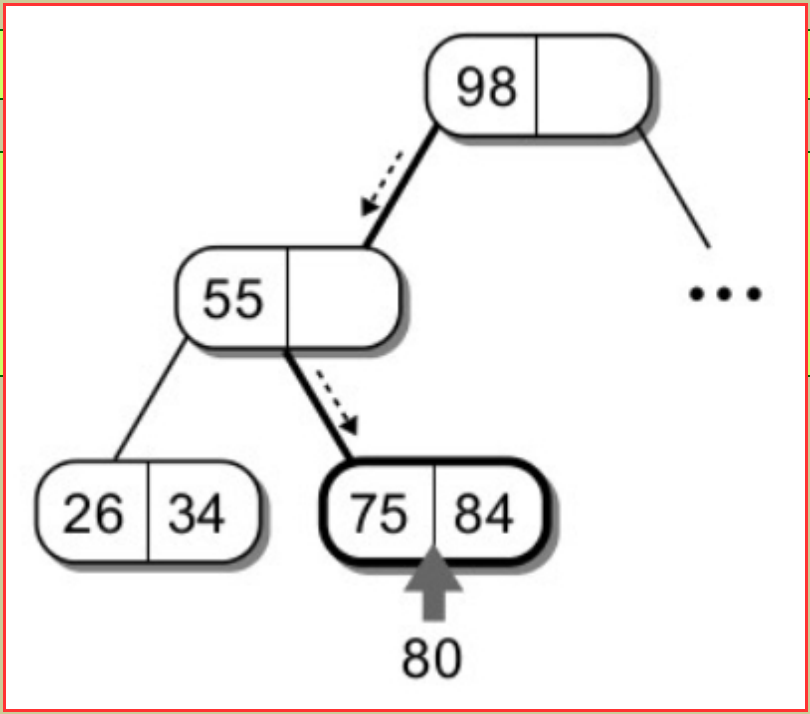


- O que fazer nesta situação?

Método

Passo 2: verificar se existe espaço no nodo folha encontrada antes
se o nodo folha contém dois elementos, então ...

- exemplo: inserir o elemento com chave **80** na árvore em baixo



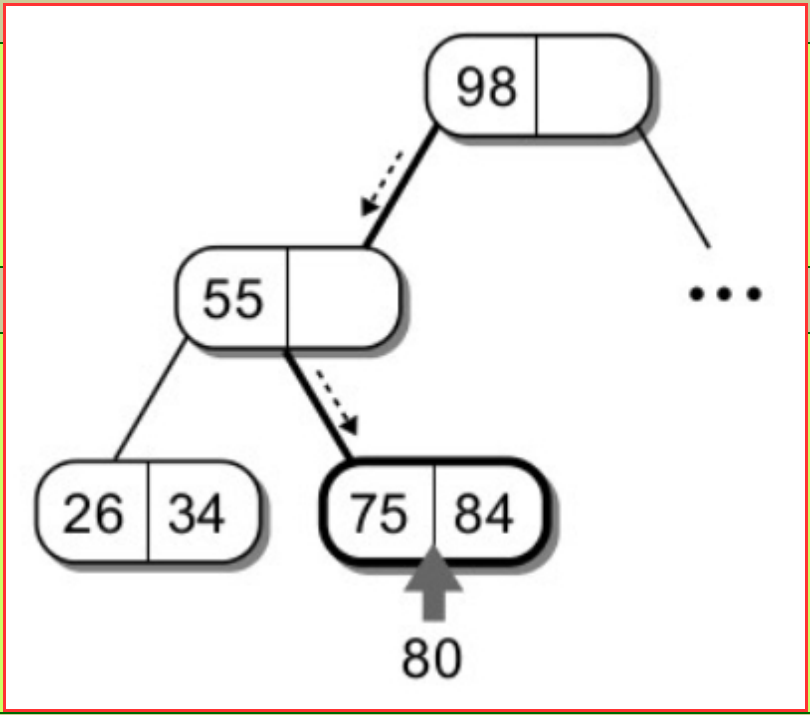
- O que fazer nesta situação?

- Criar um novo nodo com um elemento com chave 80, que será filho do meio do nodo com elementos com as chaves 75 e 84. É correto?

Método

Passo 2: verificar se existe espaço no nodo folha encontrada antes
se o nodo folha contém dois elementos, então ...

- exemplo: inserir o elemento com chave **80** na árvore em baixo
  - o que fazer nesta situação?



- Criar um novo nodo com um elemento com a chave 80, que será filho do meio do nodo com os elementos com 75 e com 84. É correto?

Esta solução viola as propriedades de árvore 2-3:

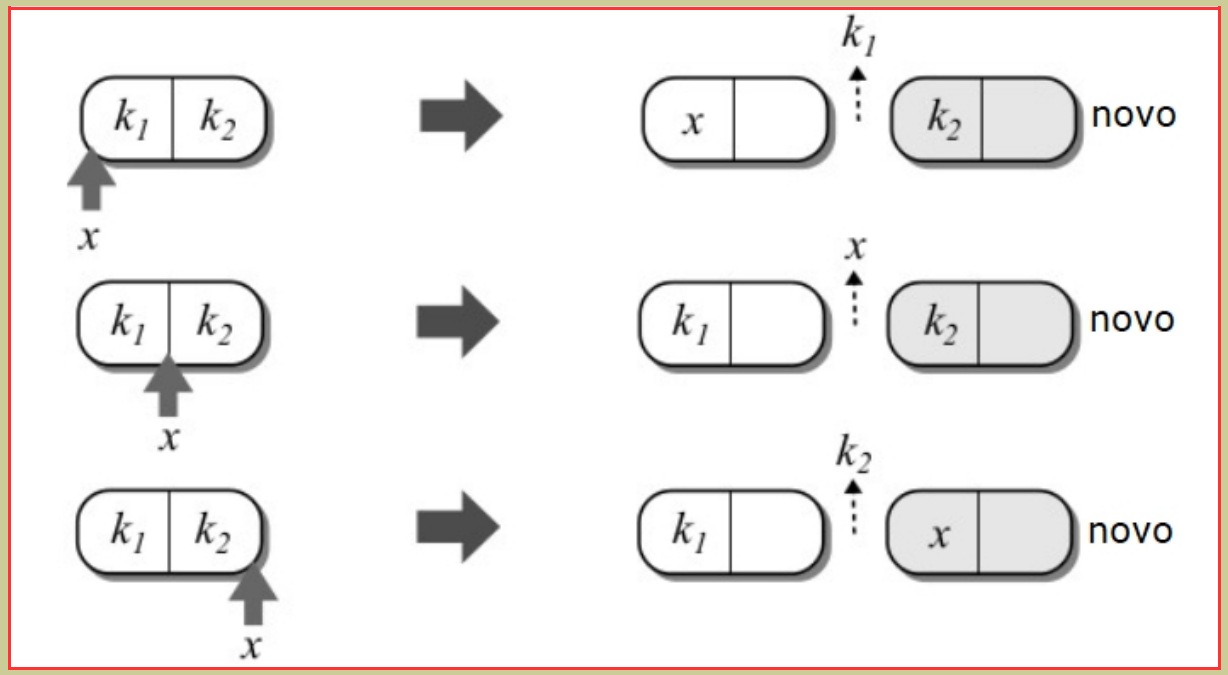
- um nodo com 2 elementos deve ter 3 filhos (neste caso, aquele nodo ficaria com 1 filho)
- o nodo folha com o novo elemento não ficava no mesmo nível das restantes folhas

Método

Passo 2: verificar se existe espaço no nodo folha encontrada antes

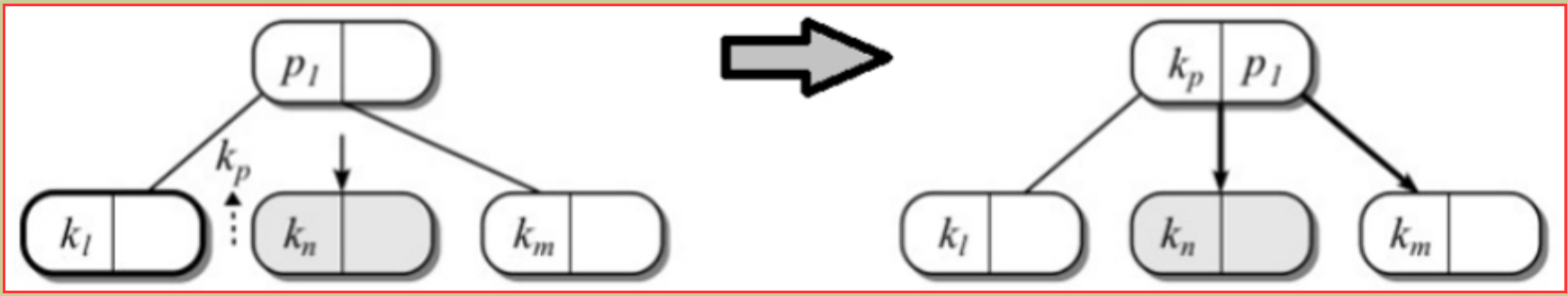
se o nodo folha contém dois elementos, então realizar as seguintes etapas:

1. criar um nodo e comparar o novo elemento com os dois elementos do nodo folha
2. o menor elemento (dos 3) é inserido no nodo folha original
3. o maior elemento é inserido no novo nodo
4. o nodo do meio passa a ser o nodo pai



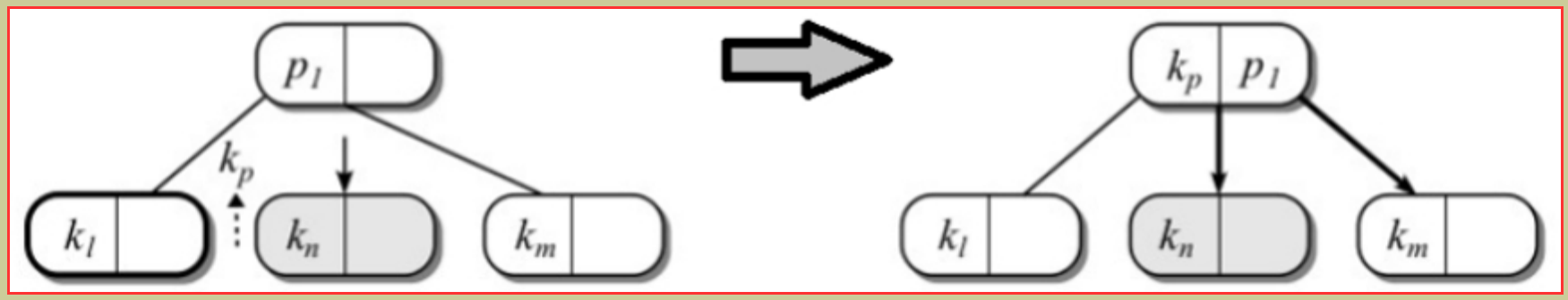
Método

- Quando um nodo é promovido para o nível de pai, ele pode ser inserido de modo similar à inserção de nodo folha
- O procedimento é simples se houver espaço no nodo pai (só com um elemento)
  - caso 1: dividir o filho esquerdo

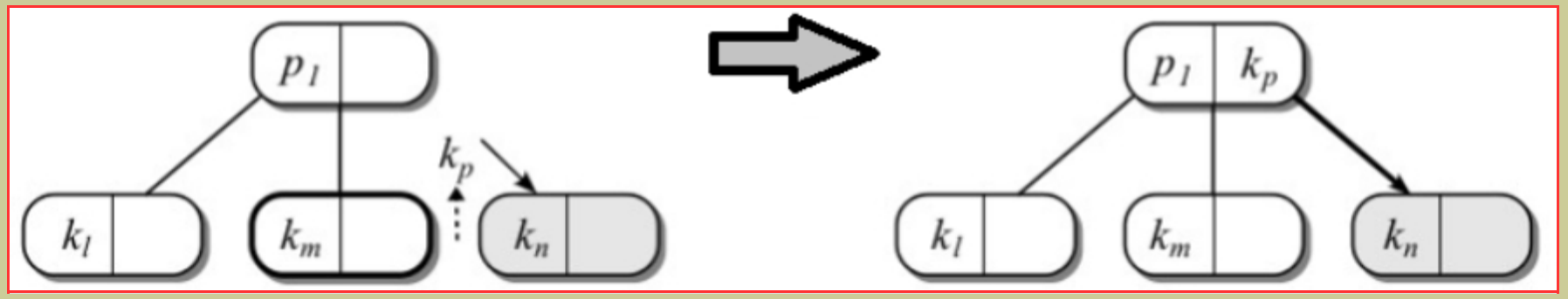


Método

- Quando um nodo é promovido para o nível de pai, ele pode ser inserido de modo similar à inserção de nodo folha
- O procedimento é simples se houver espaço no nodo pai (só com 1 chave)
  - caso 1: dividir o filho esquerdo

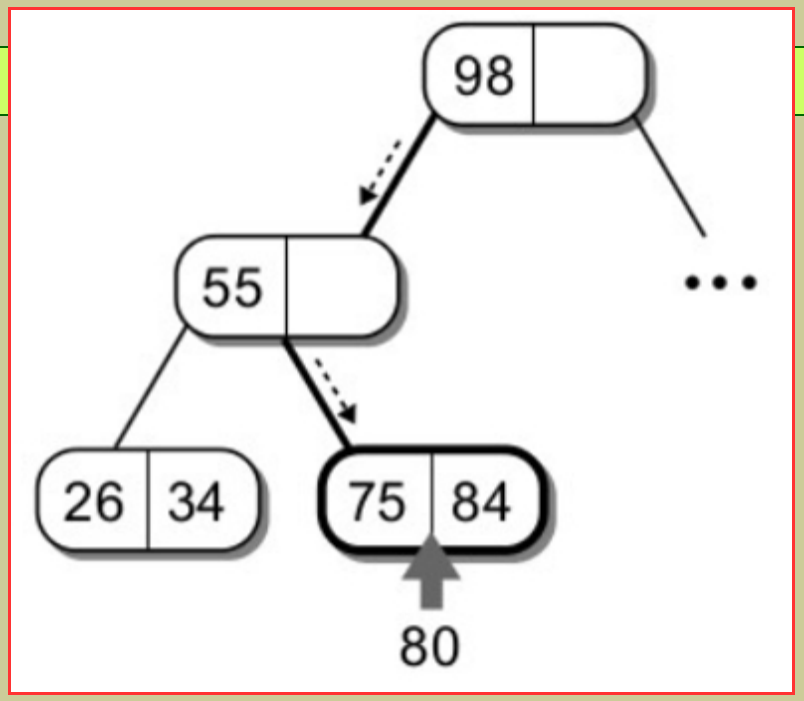


  - caso 2: dividir o filho do meio



Método

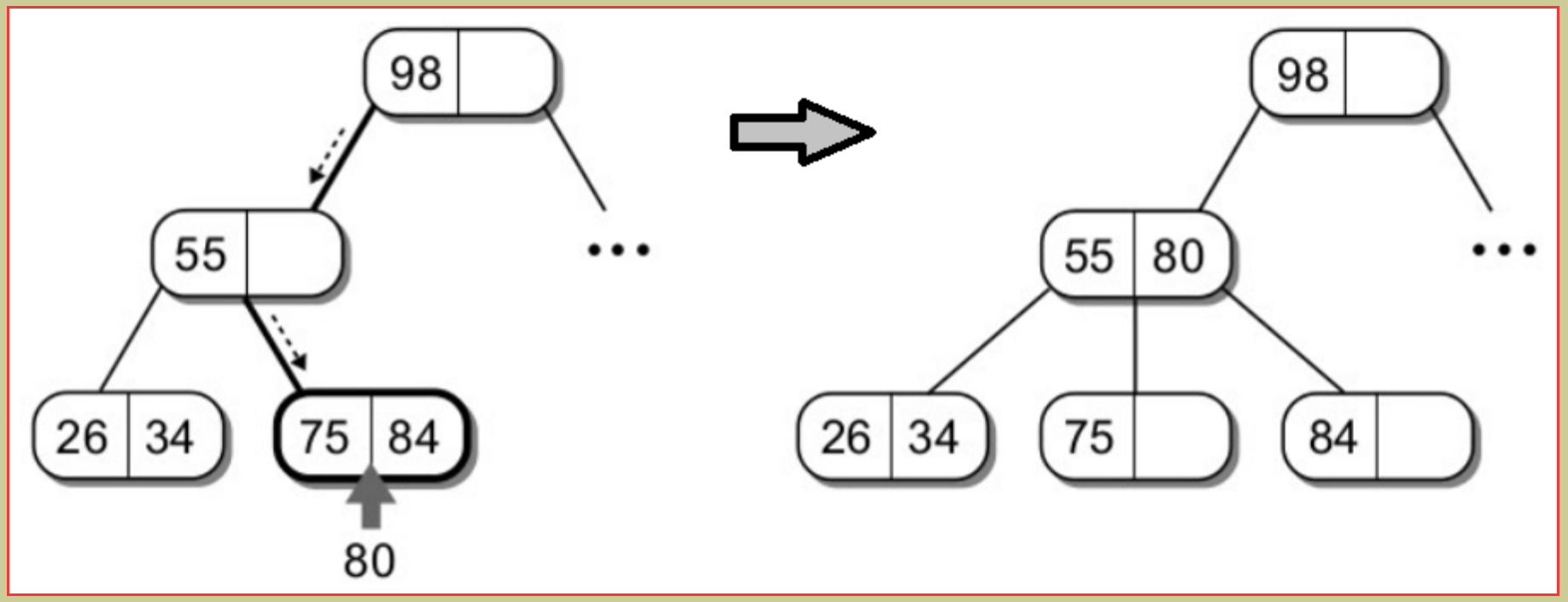
- Quando um nodo é promovido para o nível de pai, ele pode ser inserido de modo similar à inserção de nodo folha
- O procedimento é simples se houver espaço no nodo pai (só com um elemento)
  - exemplo: inserir o elemento com chave **80** na árvore em baixo



- O nodo pai só tem um elemento (com a chave 55)

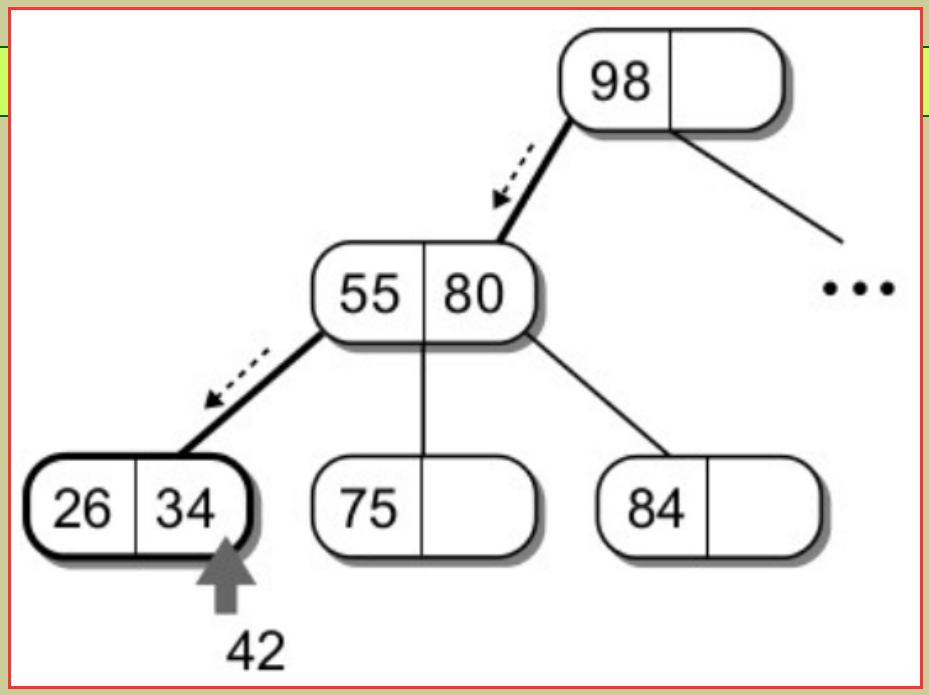
Método

- Quando um nodo é promovido para o nível de pai, ele pode ser inserido de modo similar à inserção de nodo folha
- O procedimento é simples se houver espaço no nodo pai (só com um elemento)
  - exemplo: inserir o elemento com chave **80** na árvore em baixo



Método

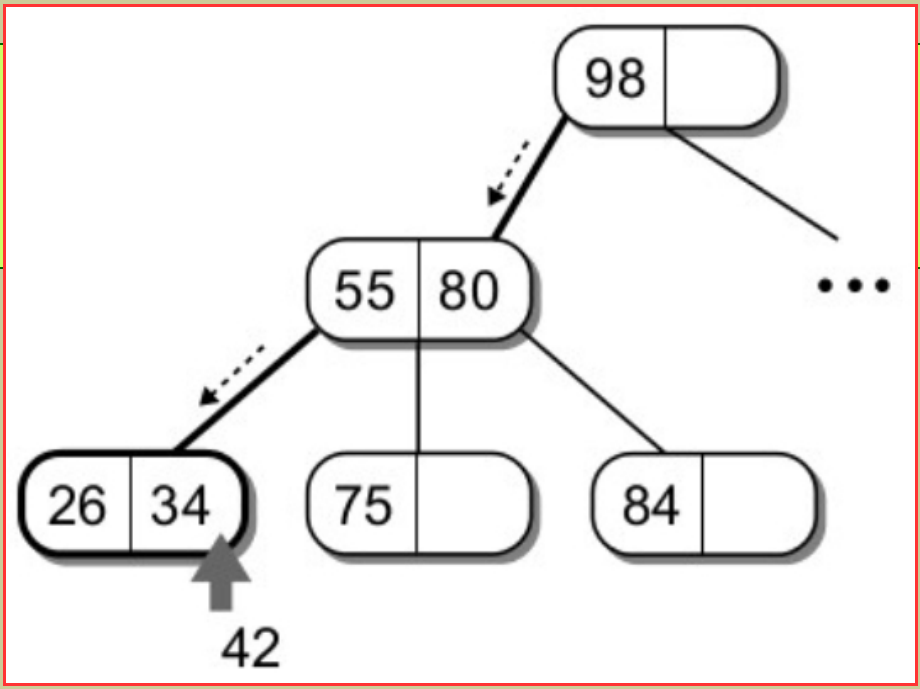
- O que aconteceria se o nodo pai já estivesse cheio (com 2 elementos)?
  - exemplo: inserir o elemento com chave **42** na árvore em baixo



- O nodo pai tem 2 elementos (chaves 55 e 80)

Método

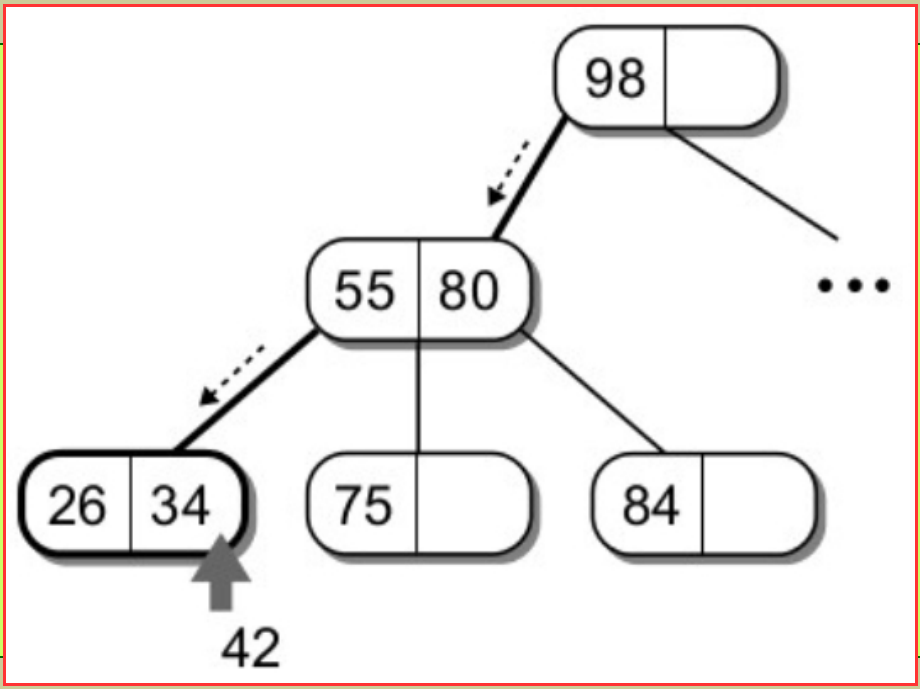
- O que aconteceria se o nodo pai já estivesse cheio (com 2 elementos)?
  - exemplo: inserir o elemento com chave **42** na árvore em baixo



- O nodo pai tem 2 elementos (chaves 55 e 80)
- Os nodos com elementos com chaves 26 e 34 devem ser divididos

Método

- O que aconteceria se o nodo pai já estivesse cheio (com 2 elementos)?
  - exemplo: inserir o elemento com chave **42** na árvore em baixo

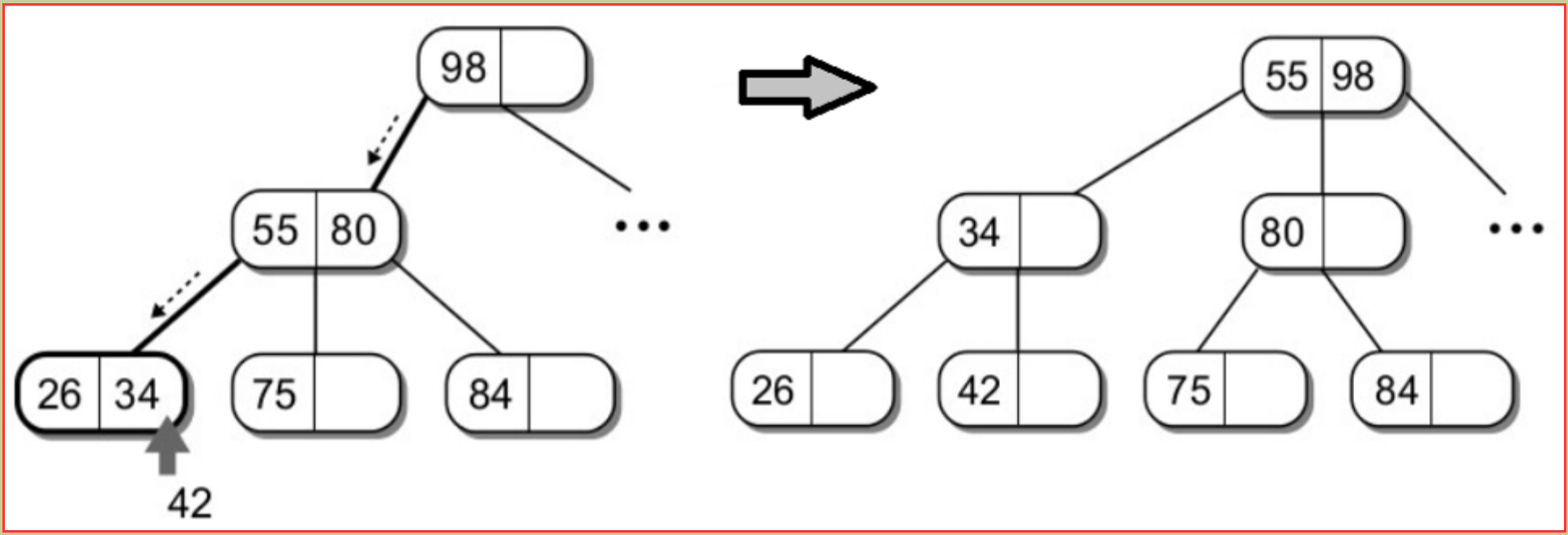


- O nodo pai tem 2 elementos (chaves 55 e 80)
- Os nodos com elementos com chaves 26 e 34 devem ser divididos
- O nodo com o elemento com chave 34 deve ser promovido
- O nodo pai com 2 elementos (chaves 55 e 80) deve ser dividido
- Processo será repetido até à raiz

Método

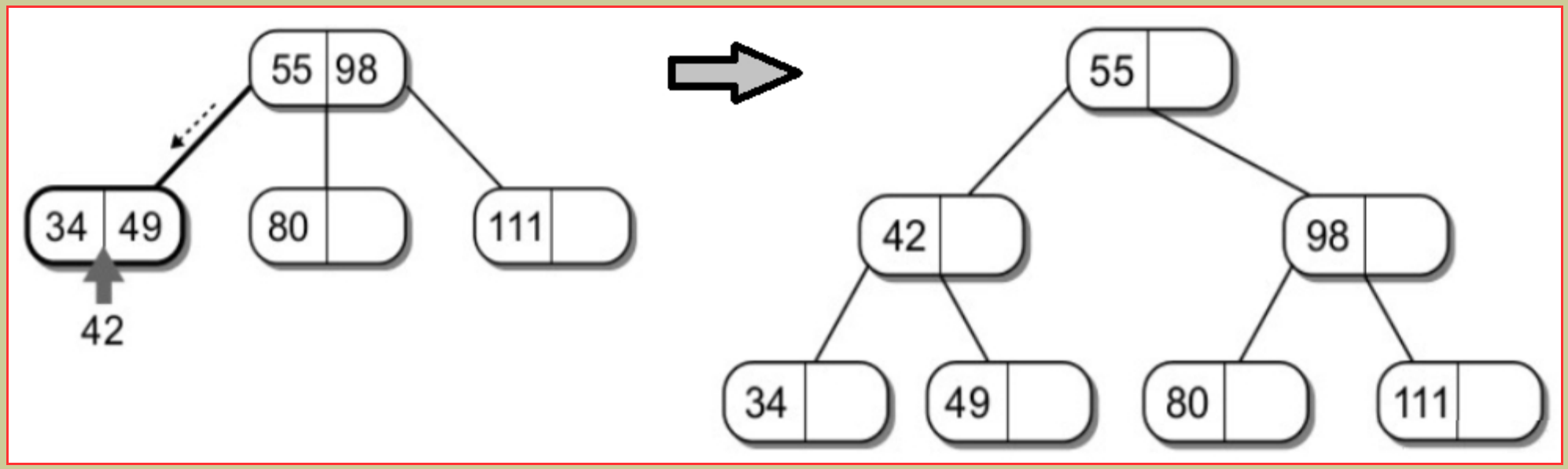
- O que aconteceria se o nodo pai já estivesse cheio (com 2 elementos)?
  - exemplo: inserir o elemento com chave **42** na árvore em baixo

- Os nodos com elementos com chaves 26 e 34 devem ser divididos
- O nodo com o elemento com chave 34 deve ser promovido
- O nodo pai com 2 elementos (chaves 55 e 80) deve ser dividido
- Processo será repetido até à raiz



Método

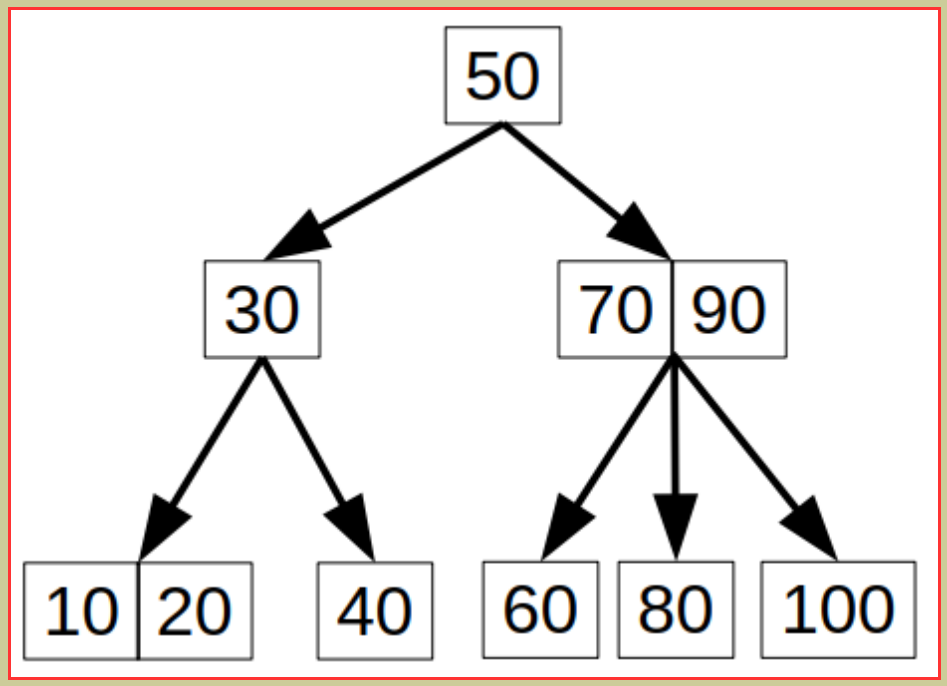
- Quando o novo nodo precisa ser dividido, é criado um novo nodo raiz com o primeiro (ou segundo) elemento do nodo raiz original
  - o nodo raiz original com o segundo (ou primeiro) elemento torna-se o filho direito (ou esquerdo) do novo nodo raiz
  - o novo nodo torna-se o filho esquerdo (ou direito) do novo nodo raiz



Remover um elemento

Método

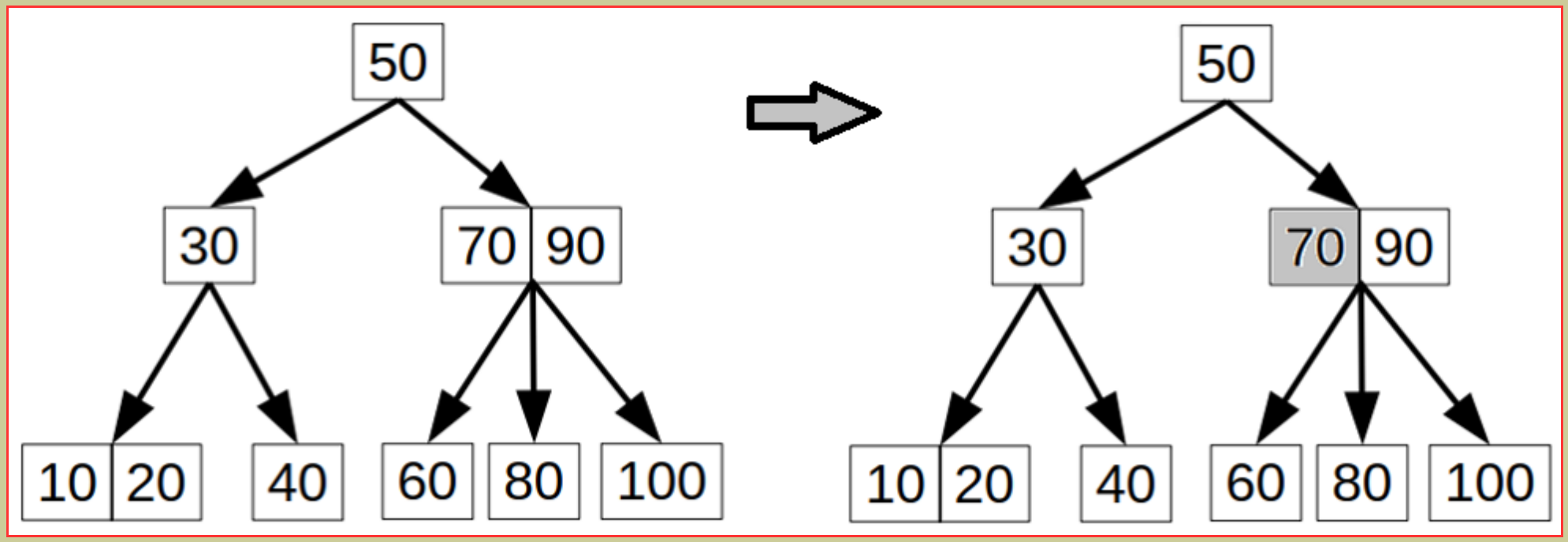
- Os processos de remoção de um elemento são semelhantes aos aplicados às árvores binárias de pesquisa
- Exemplo: remover o elemento com chave **70** da árvore em baixo



Método

- Exemplo: remover o elemento com chave **70** da árvore em baixo

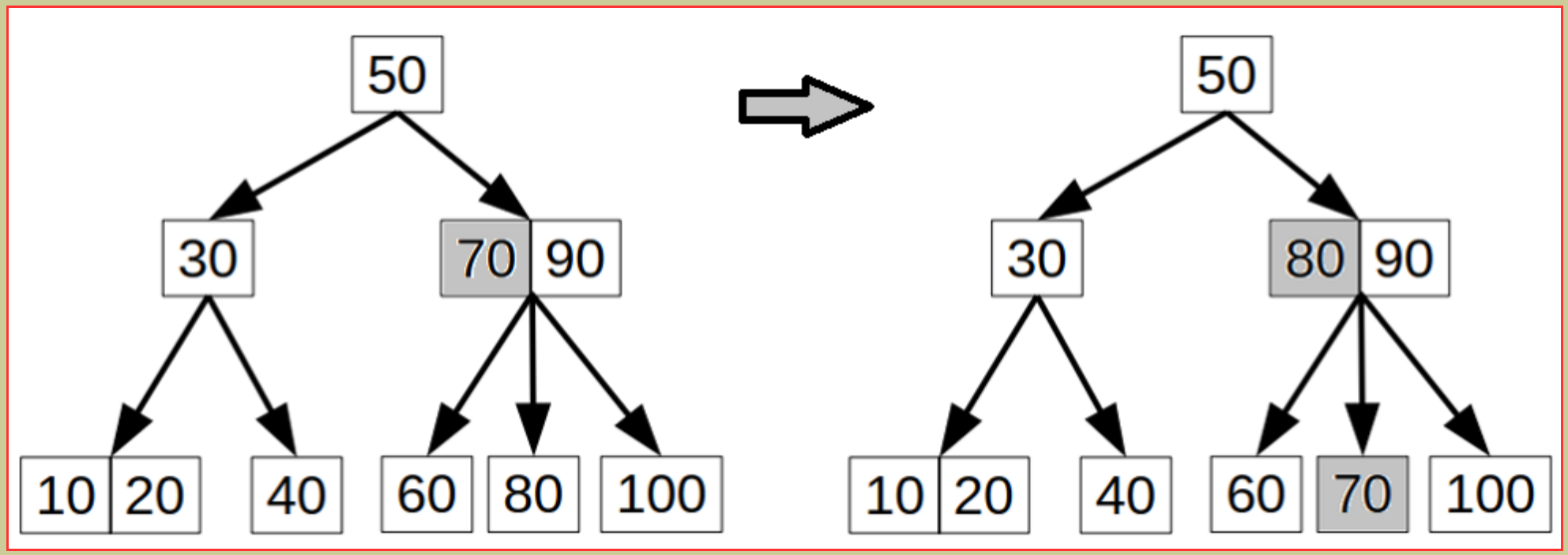
Passo 1: localizar o nodo com o elemento com chave 70 (elemento a remover)



Método

- Exemplo: remover o elemento com chave **70** da árvore em baixo

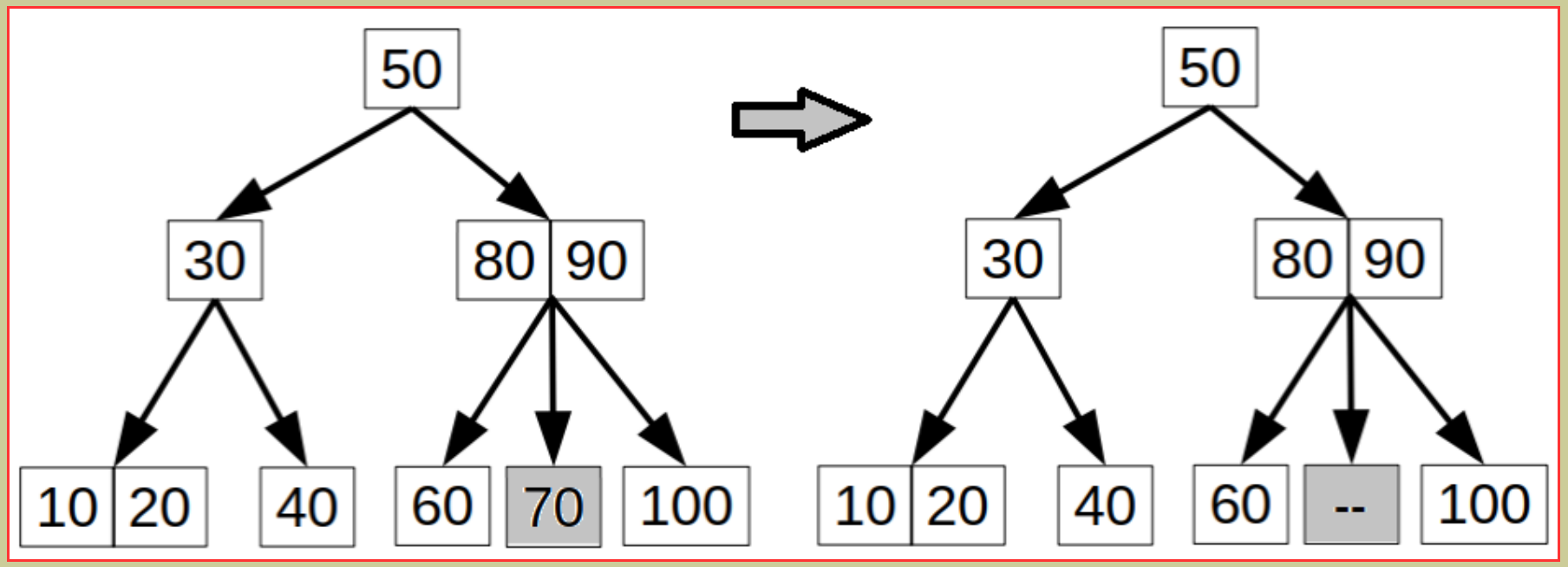
Passo 2: o nodo com o elemento com a chave 70 não é uma folha
- trocar o elemento com chave 70 pelo seu sucessor (elemento com chave 80)



Método

- Exemplo: remover o elemento com chave **70** da árvore em baixo

Passo 3: o nodo com o elemento com chave 70 torna-se uma folha e só tem um elemento
- o elemento com chave 70 deve ser removido, ficando a folha vazia

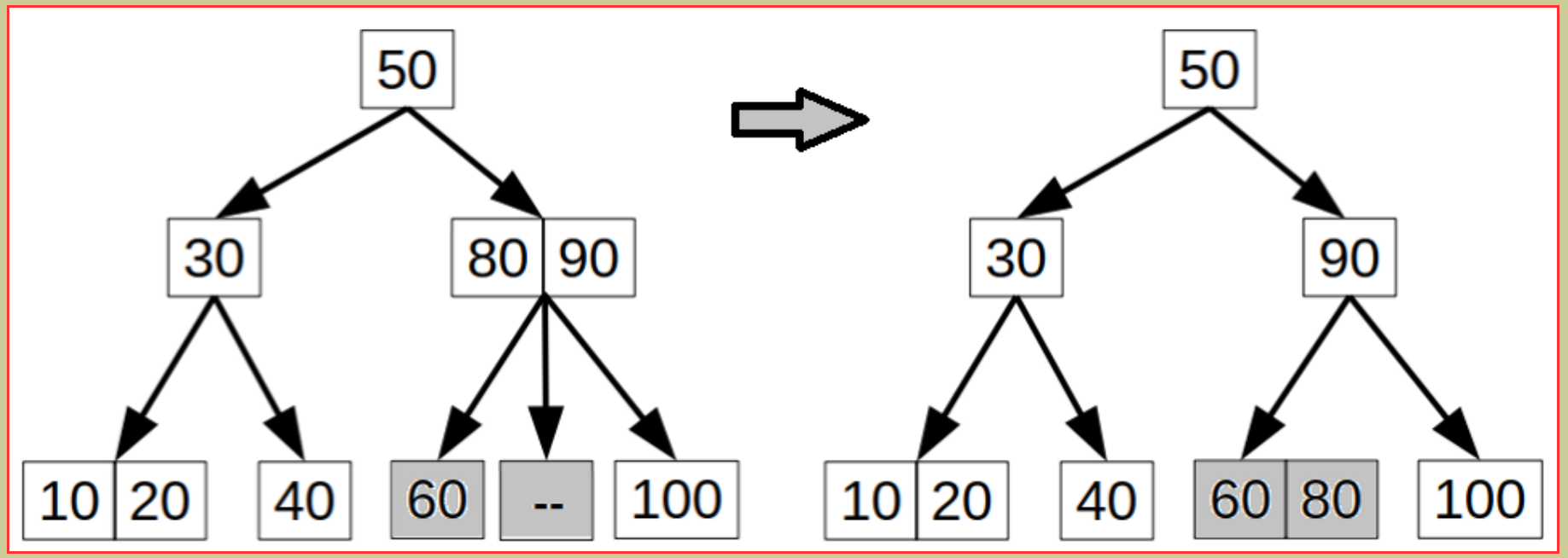


- com a remoção do nodo com chave 70, as propriedades de árvore 2-3 são violadas
  - um nodo com 2 elementos (chaves 80 e 90) tem apenas 2 nodos filhos (com chaves 60 e 100), e deve ter 3

Método

- Exemplo: remover o elemento com chave **70** da árvore em baixo

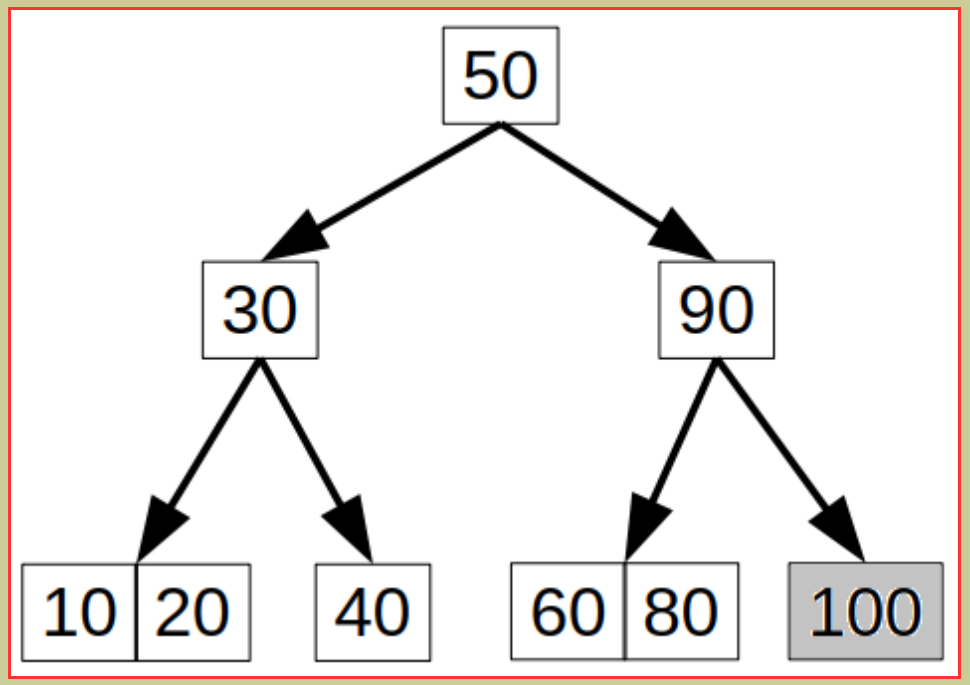
Passo 3: o nodo com o elemento com chave 70 torna-se uma folha e só tem um elemento
- o elemento com chave 70 deve ser removido, ficando a folha vazia
- a redistribuição dos irmãos não é possível (são 2 elementos em 2 nodos)
  - juntar o pai (com chave 80) e os nodos irmãos da folha vazia, movendo o elemento com chave 80 para baixo



Método

- Exemplo: remover o elemento com chave **100** da árvore em baixo

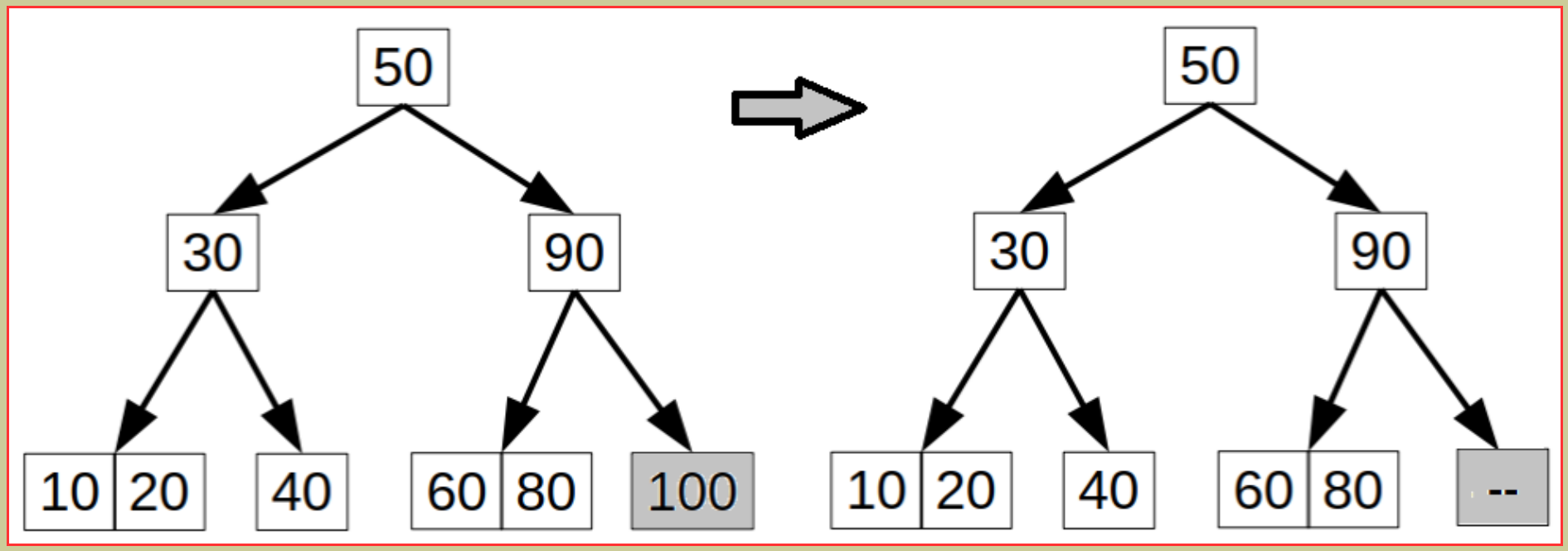
Passo 1: localizar o nodo com o elemento com chave 100



Método

- Exemplo: remover o elemento com chave **100** da árvore em baixo

Passo 3: o nodo com o elemento com chave 100 é uma folha
- o elemento com chave 100 deve ser removido, ficando a folha vazia

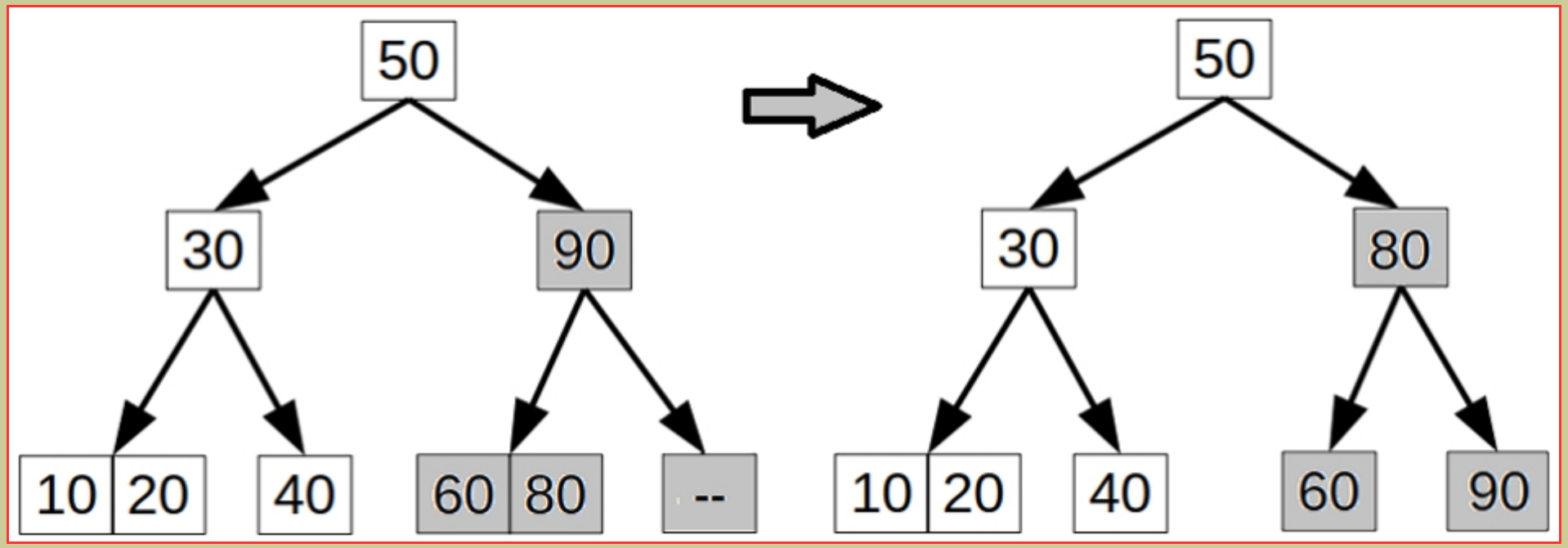


- com a remoção do nodo com a chave 100, as propriedades de árvore 2-3 são violadas
  - um nodo com 1 elemento (chave 90) tem apenas 1 nodo filho (elementos com as chaves 60 e 800), e deve ter 2

Método

- Exemplo: remover o elemento com chave **100** da árvore em baixo

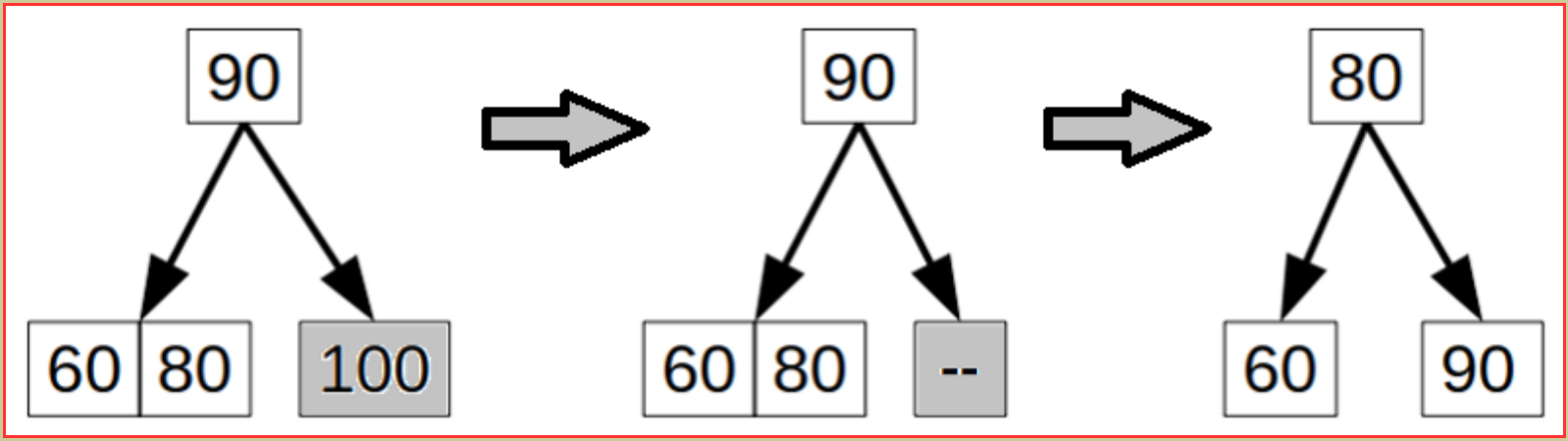
Passo 3: o nodo com o elemento com chave 100 é uma folha
- o elemento com chave 100 deve ser removido, ficando a folha vazia
- redistribuir os elementos dos nodos irmãos e pai do nodo com a chave 100, para que o nodo pai (só com um elemento) tenha 2 filhos



Método

- Exemplo: remover o elemento com chave **100** da árvore em baixo
  - resumo do passo 3 do processo, considerando apenas os nodos envolvidos

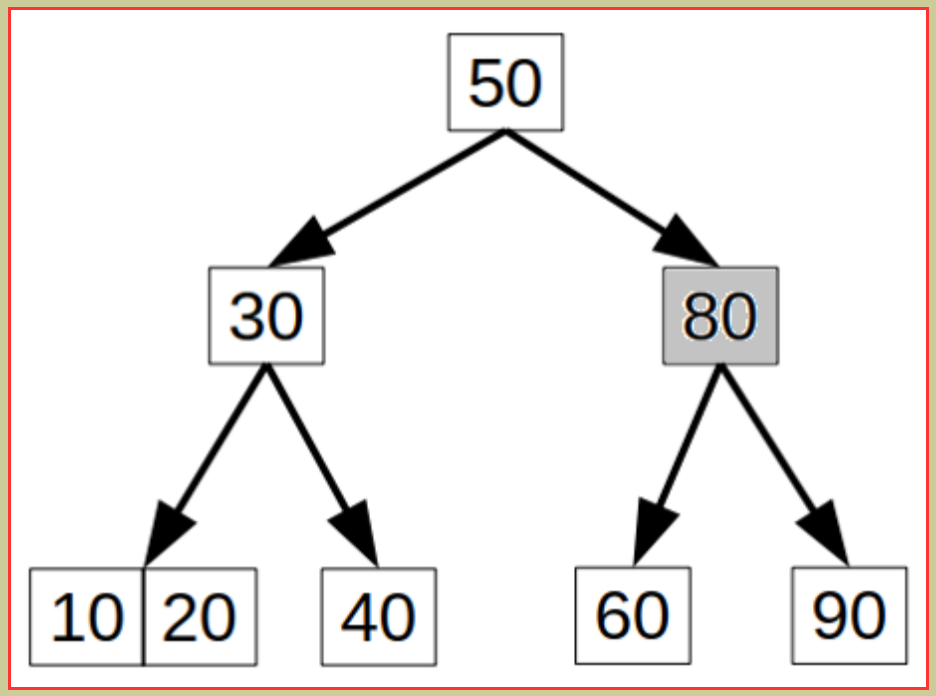
Passo 1: localizar o nodo com o elemento com chave 100
Passo 2: o nodo com a chave 100 é uma folha, então passar ao passo 3
Passo 3: o nodo com a chave 100 é uma folha apenas com um elemento
- remover a folha com a chave 100
- redistribuir os elementos (pai e filhos), pois o nodo com 90 deve ter 2 filhos



Método

- Exemplo: remover a chave 80 da árvore em baixo

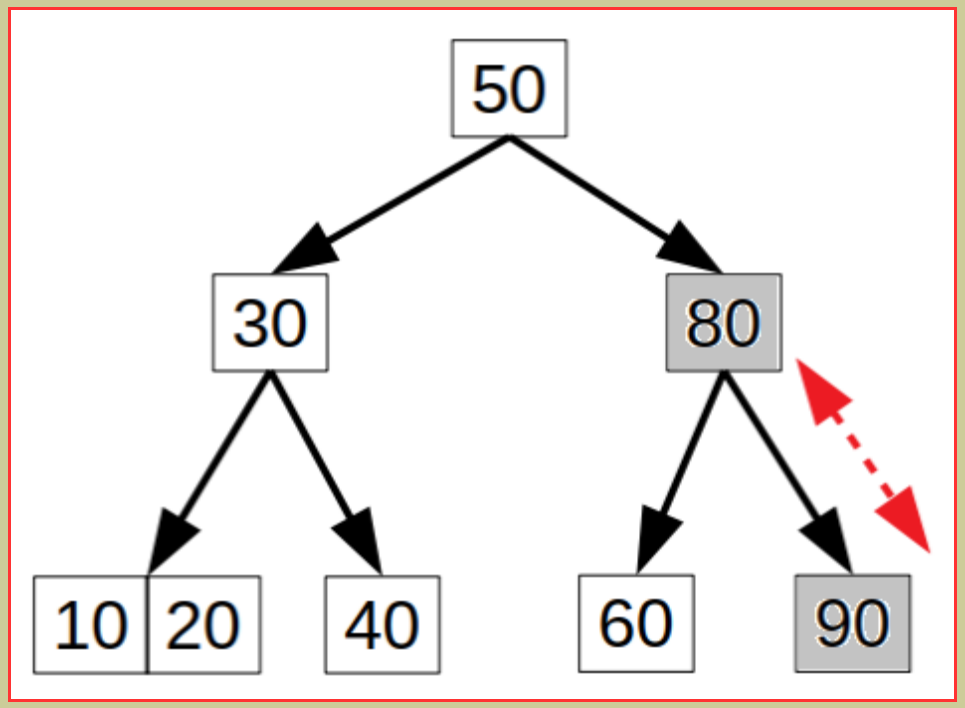
Passo 1: localizar o nodo com **80**



Método

- Exemplo: remover a chave 80 da árvore em baixo

Passo 2: o nodo com 80 não é uma folha
- trocar a chave **80** pela chave do seu sucessor (**90**)

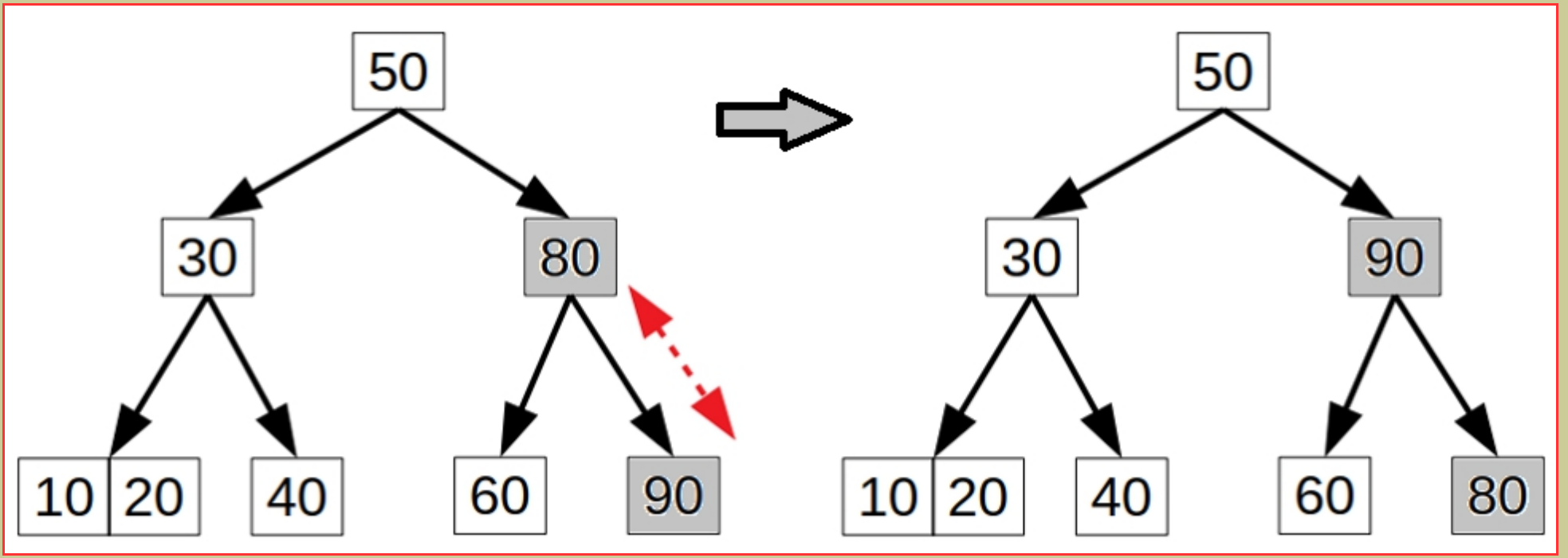


Método

- Exemplo: remover a chave 80 da árvore em baixo

Passo 2: o nodo com 80 não é uma folha
- trocar a chave **80** pela chave do seu sucessor (**90**)

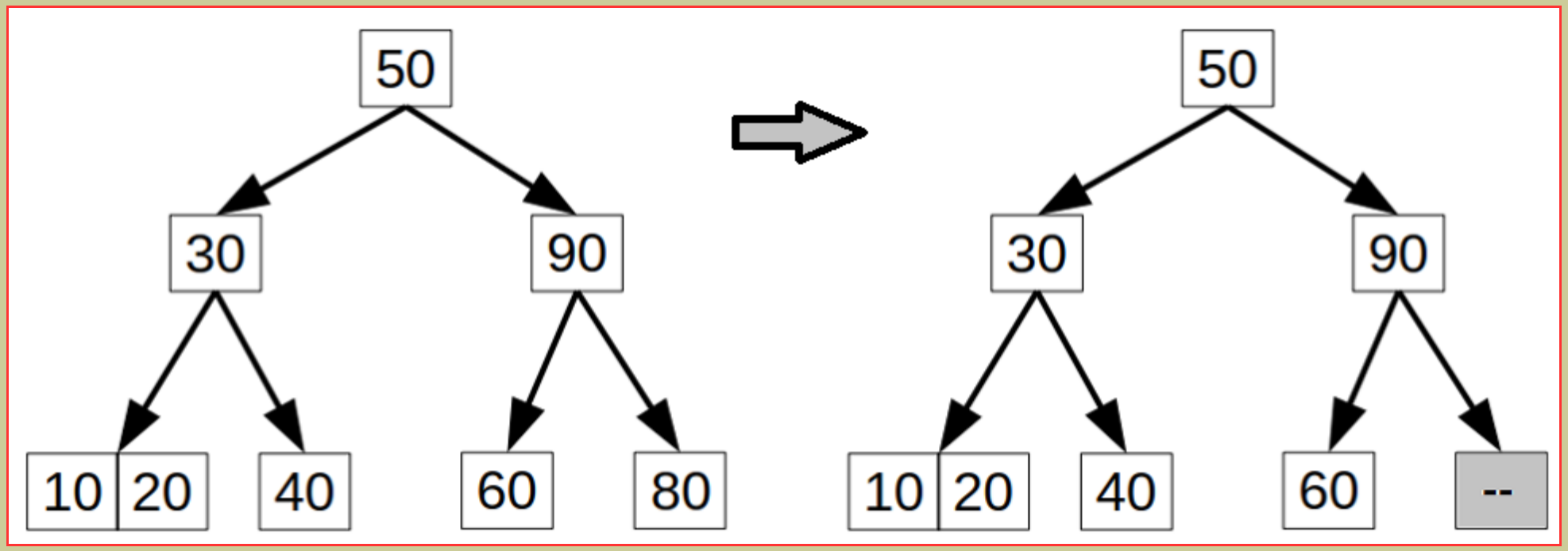
(o nodo com 90 deixa de ser filho e passa a ser pai do nodo com 80)



Método

- Exemplo: remover a chave 80 da árvore em baixo

Passo 3: o nodo com 80 é (agora) uma folha apenas com uma chave
- remover o nodo folha com **80** (fica uma folha vazia)

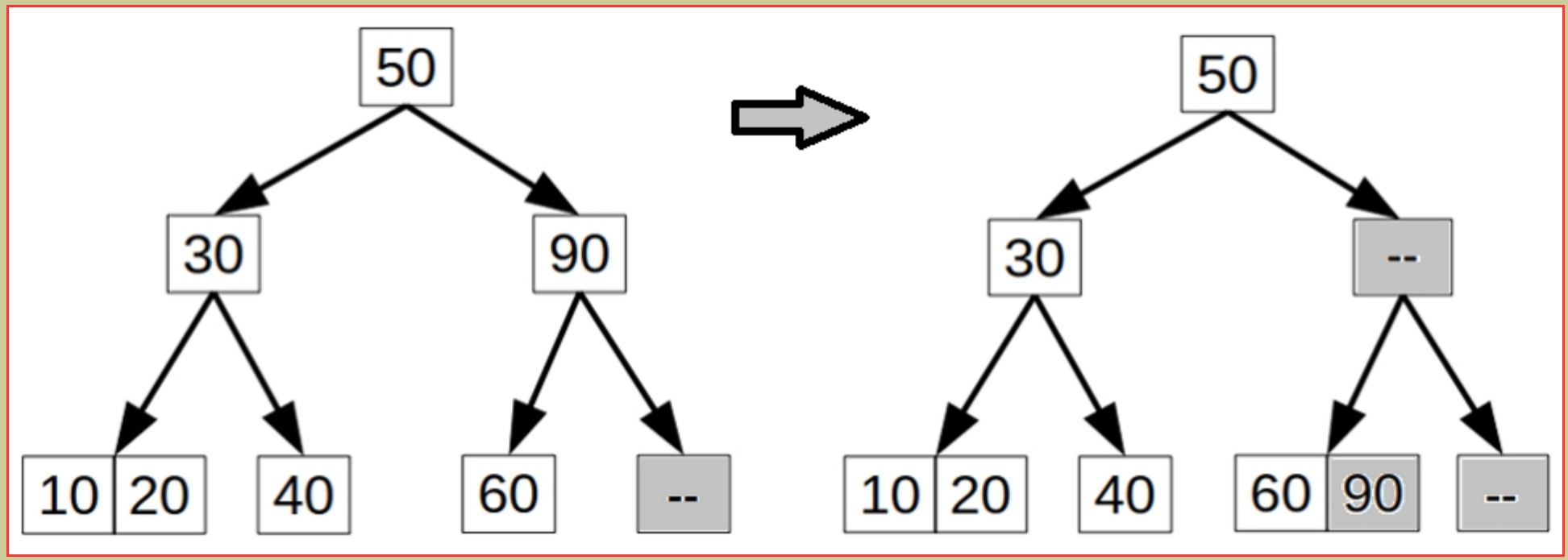


- com a remoção do nodo com 80, as propriedades de árvore 2-3 são violadas
  - um nodo com 1 chave (90) tem apenas 1 nodo filho (60), e deve ter 2

Método

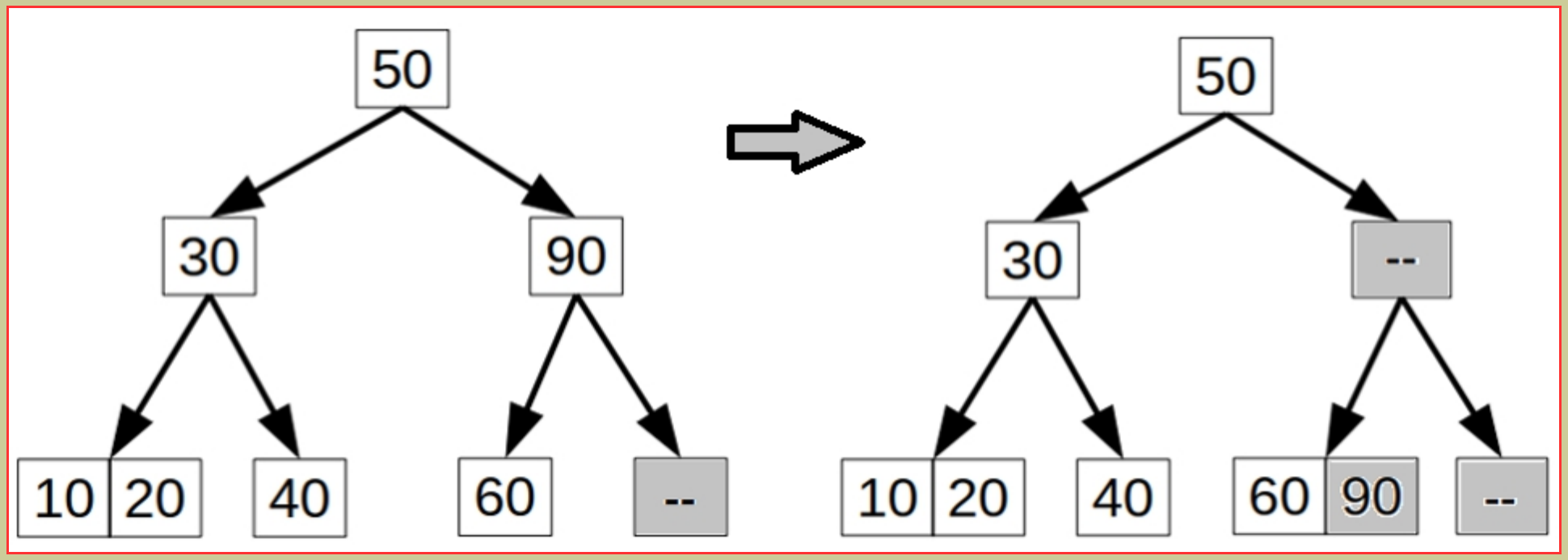
- Exemplo: remover a chave 80 da árvore em baixo

Passo 3: o nodo com 80 é (agora) uma folha apenas com uma chave
- remover o nodo folha com **80** (fica uma folha vazia)
- redistribuir os nodos irmãos da folha vazia não é possível (1 pai para 1 filho)
  juntar os nodos pai e irmão da folha vazia, movendo o nodo com **90** para baixo



Método

- Exemplo: remover a chave 80 da árvore em baixo

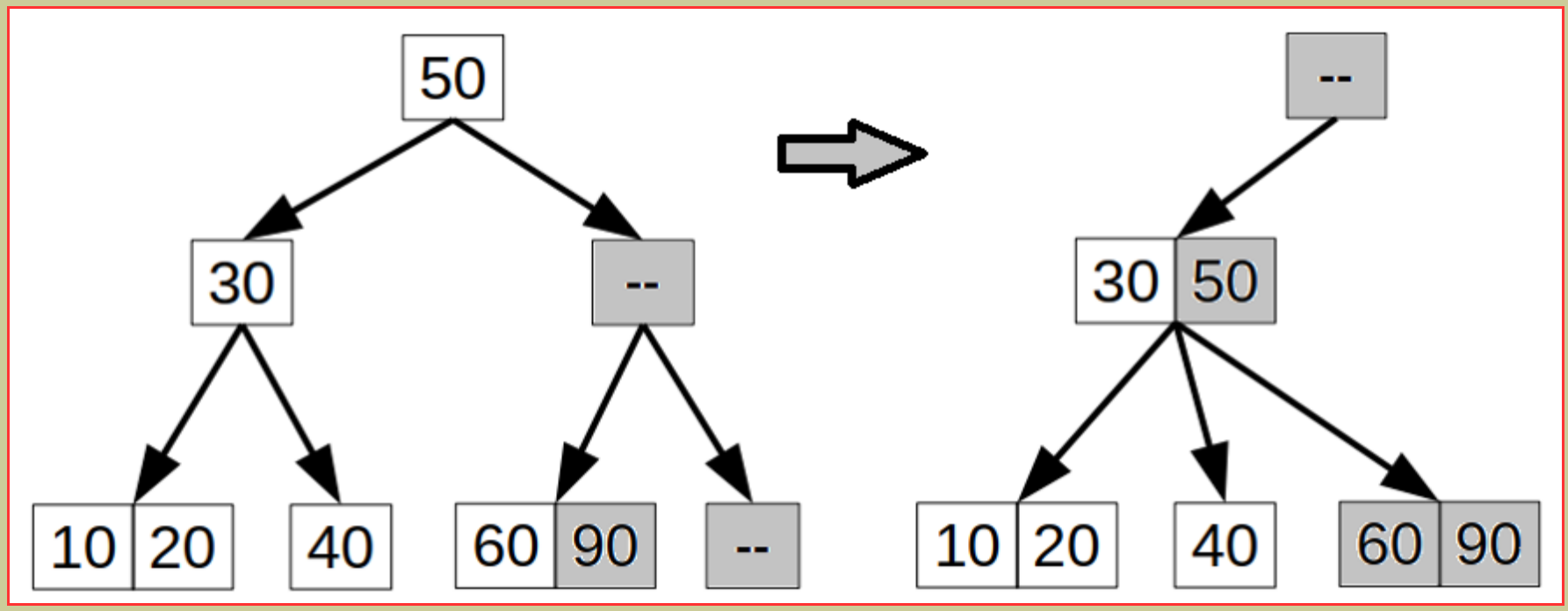


- o pai do nodo com 60 e 90 é um nodo vazio
- ligar o nodo com 50 ao nodo com 60 e 90 não é possível (viola as propriedades de árvore 2-3)
  - as folhas ficam em níveis diferentes

Método

- Exemplo: remover a chave 80 da árvore em baixo

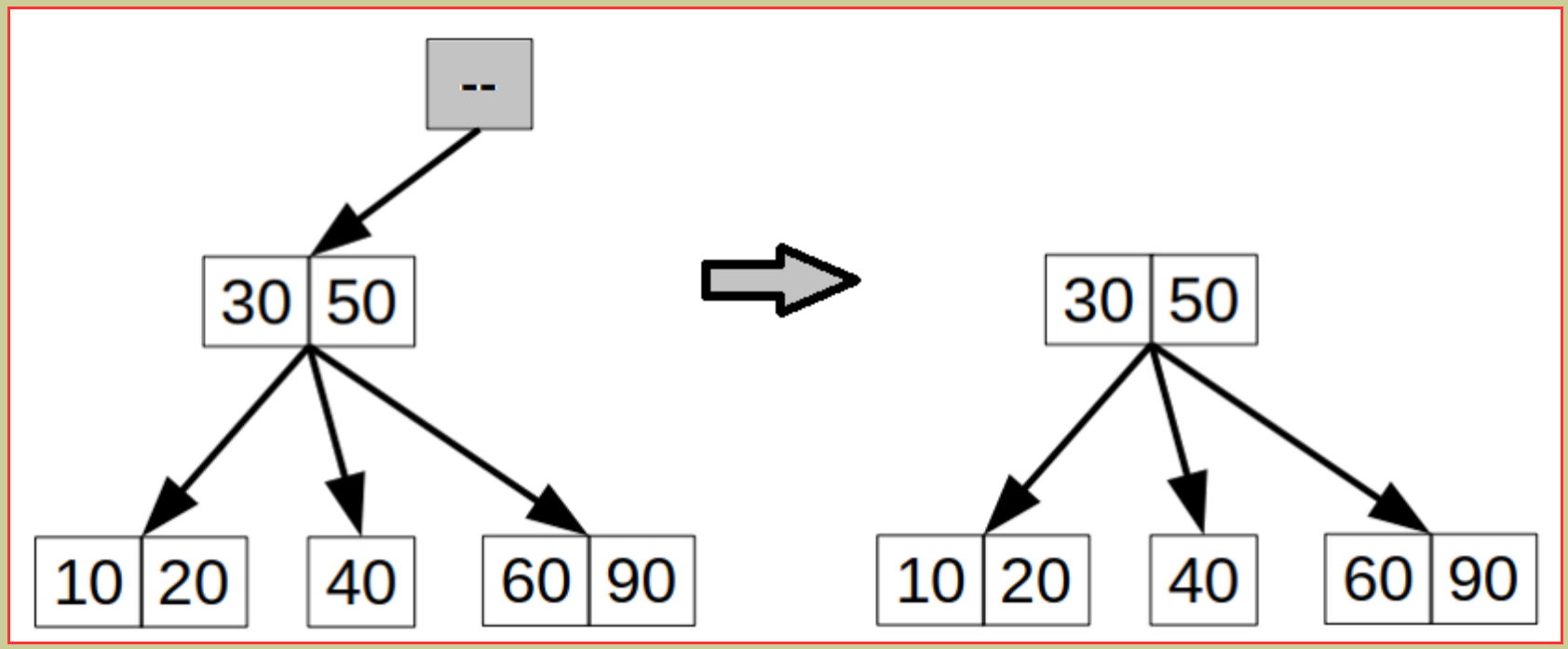
Passo 3: juntar os nodos da folha com vazia e mover o nodo com **50** para baixo



Método

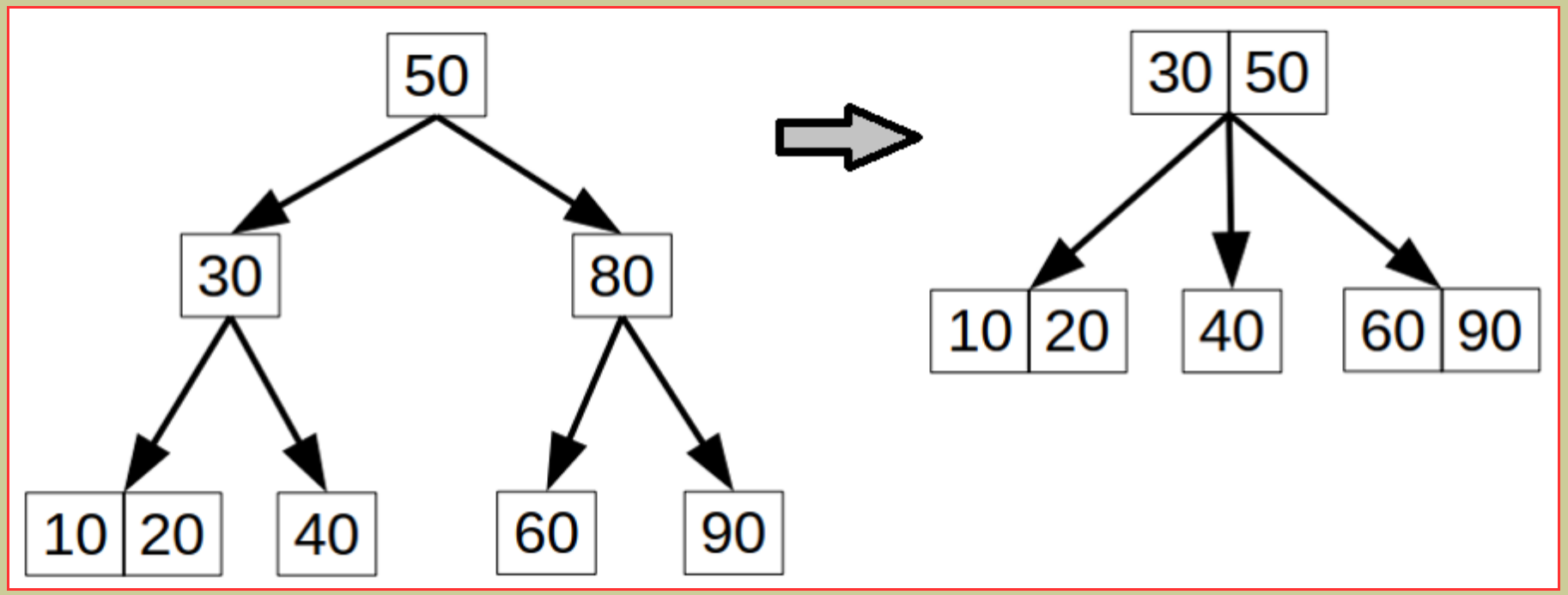
- Exemplo: remover a chave 80 da árvore em baixo

Passo 3: remover a raiz (que está vazia)



Método

- Exemplo: remover a chave 80 da árvore em baixo (resumo)

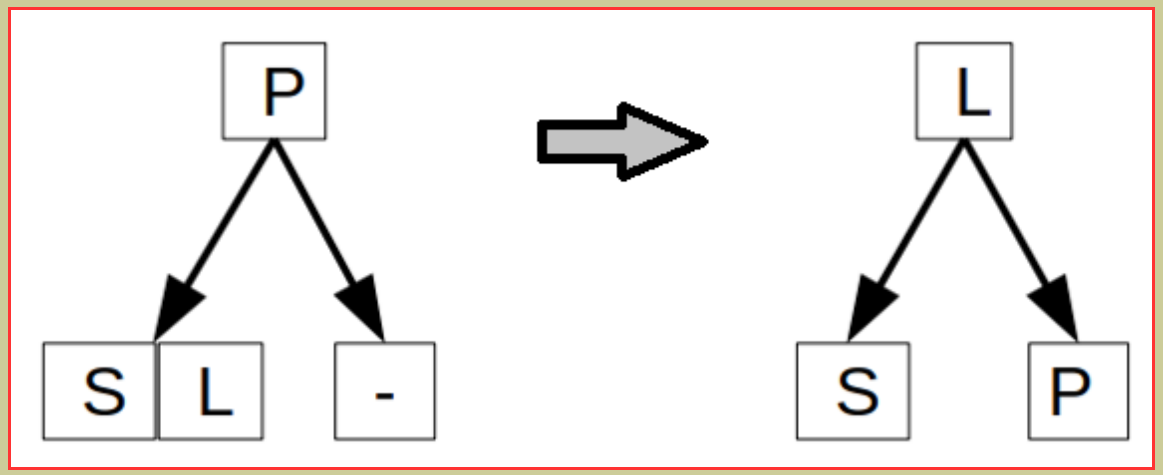


Algoritmo

algoritmo remover(X, T)
  Passo 1:
    localizar o nodo que contém a chave X
  Passo 2:
    se o nodo não é uma folha **então**
      trocar X pela chave do seu sucessor
      (a remoção será sempre nas folhas)
    fim_se
  Passo 3:
    se o nodo folha com X contém outra chave **então**
      remover a chave X
    senão
      remover o nodo com a chave X
      redistribuir os nodos irmãos (se for possível), ou
      juntar os nodos pai aos irmãos da folha removida
    fim_se

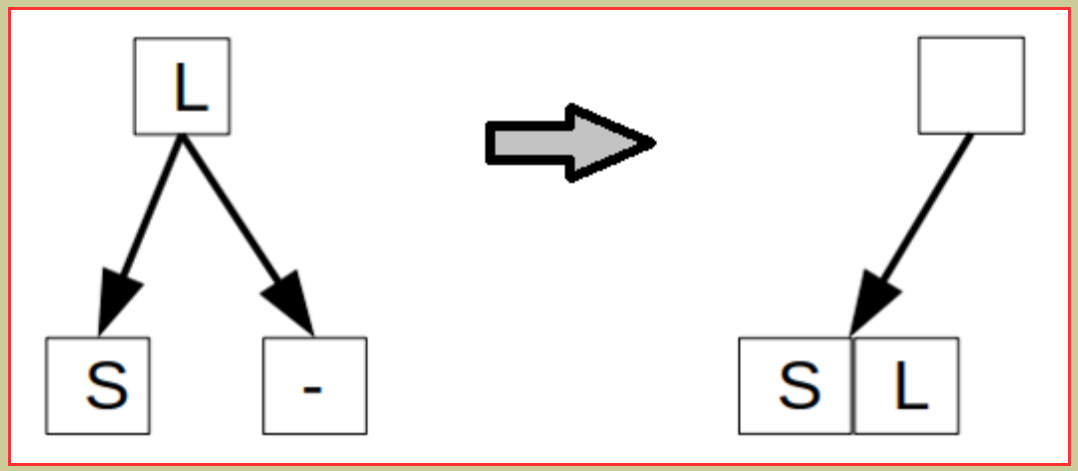
Algoritmo

- Redistribuir nodos (P, S e L)



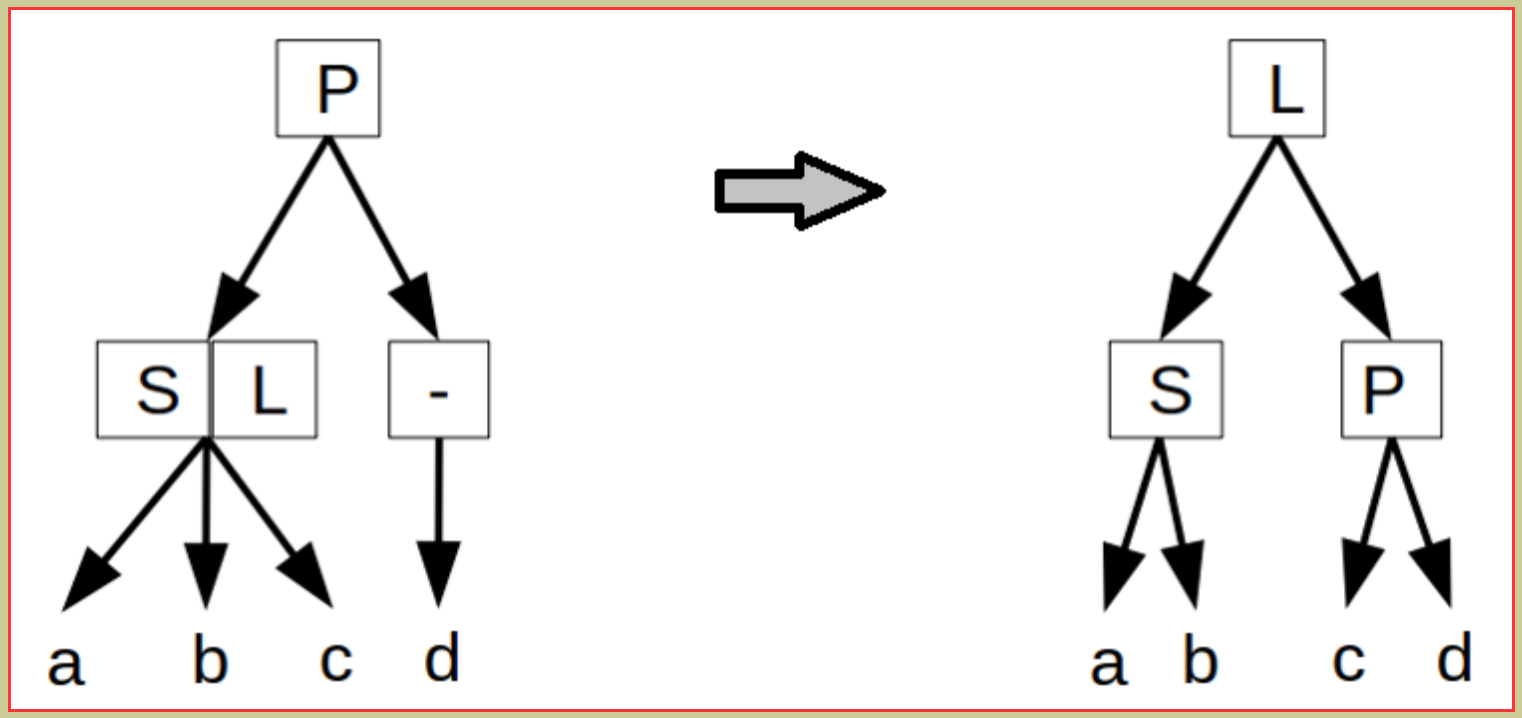
Algoritmo

- Juntar nodos (L e S)
  - redistribuição não é possível
  - mover o nodo pai (L) para baixo



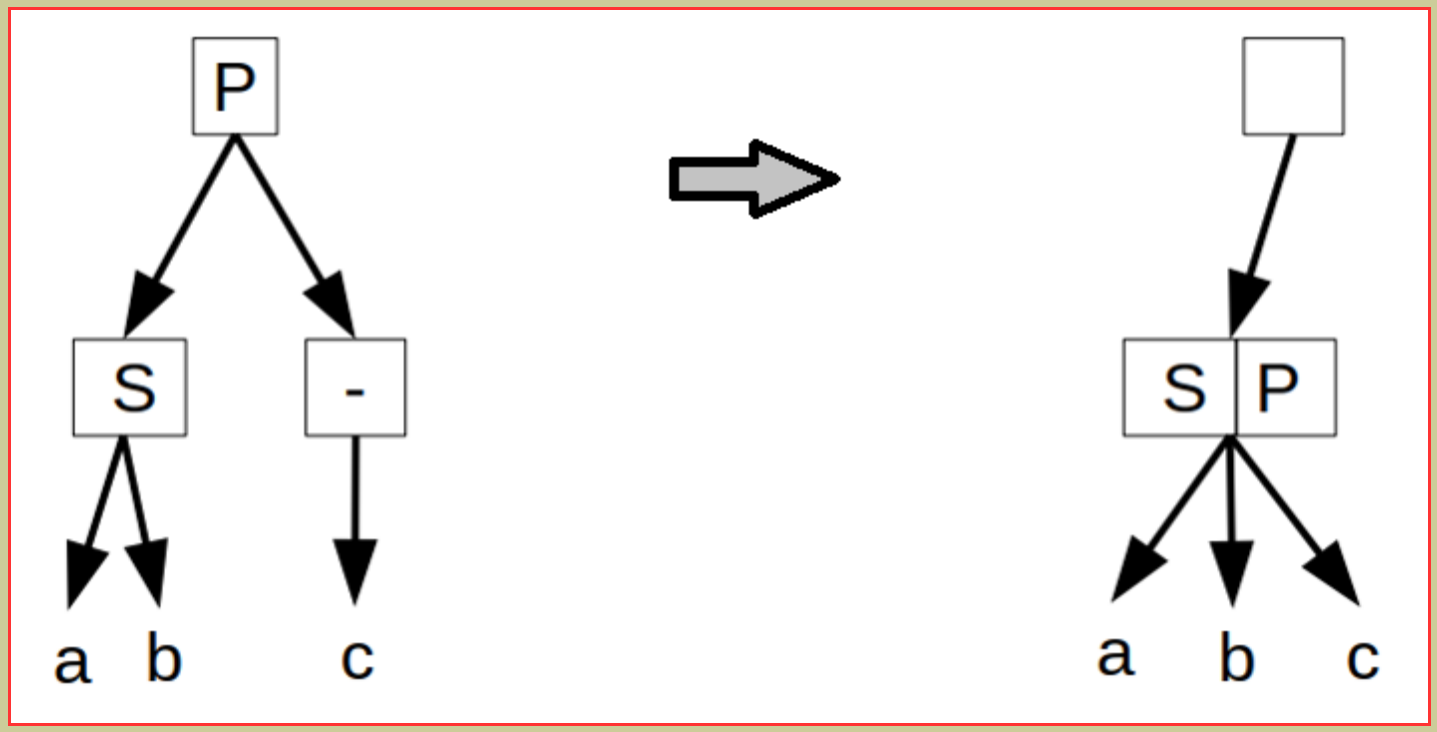
Algoritmo

- Redistribuição de nodos
  - nó interno não tem filho à direita



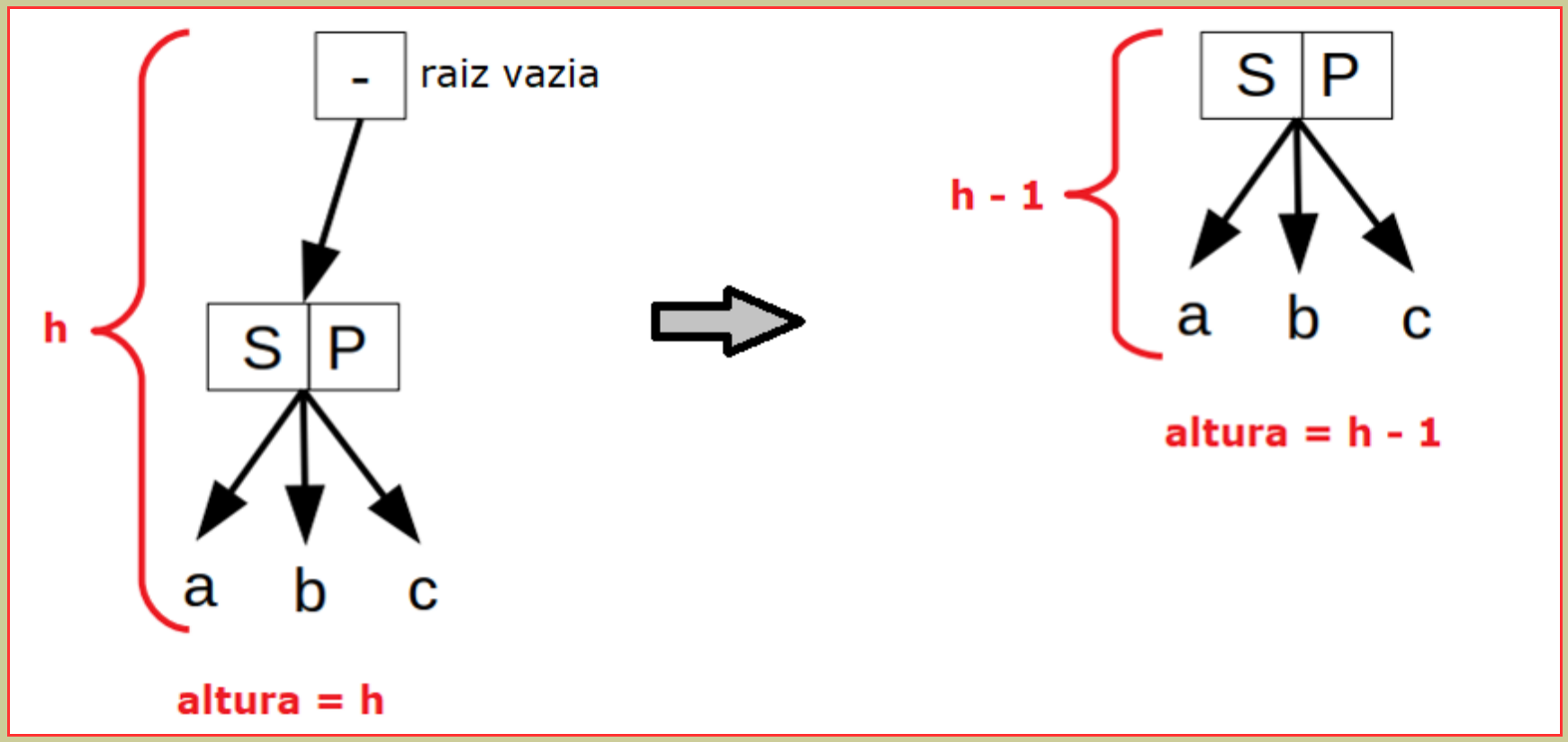
Algoritmo

- Juntar nodos (L e S)
  - redistribuição não é possível
  - mover o nodo pai (L) para baixo



Algoritmo

- Juntar nodos (L e S)
  - se o processo de juntar nodos chegar até à raiz e a raiz estiver vazia então remover a raiz



Eficiência

- Por definição, uma árvore 2-3 tem altura balanceada com todos nodos folhas no mesmo nível
- No pior caso, todos os nodos contêm uma única chave e todos os nodos interior apenas têm dois filhos
- Considerando que a altura da árvore 2-3 é sempre **log n**, então a operação de pesquisa de uma chave não necessita mais do que **log n** comparações
  - resultando O (log n) no pior caso