

Algoritmos e Estruturas de Dados I

# Árvores 2-3

Prof. Tiago Eugenio de Melo

[tmelo@uea.edu.br](mailto:tmelo@uea.edu.br)

[www.tiagodemelo.info](http://www.tiagodemelo.info)

# Observações

- O conteúdo dessa aula é parcialmente proveniente do Capítulo 14 do livro “*Data Structure and Algorithms Using Python*”.
- As palavras com a fonte `Courier` indicam uma palavra-reservada da linguagem de programação.

# Árvores 2-3

# Introdução

# Introdução

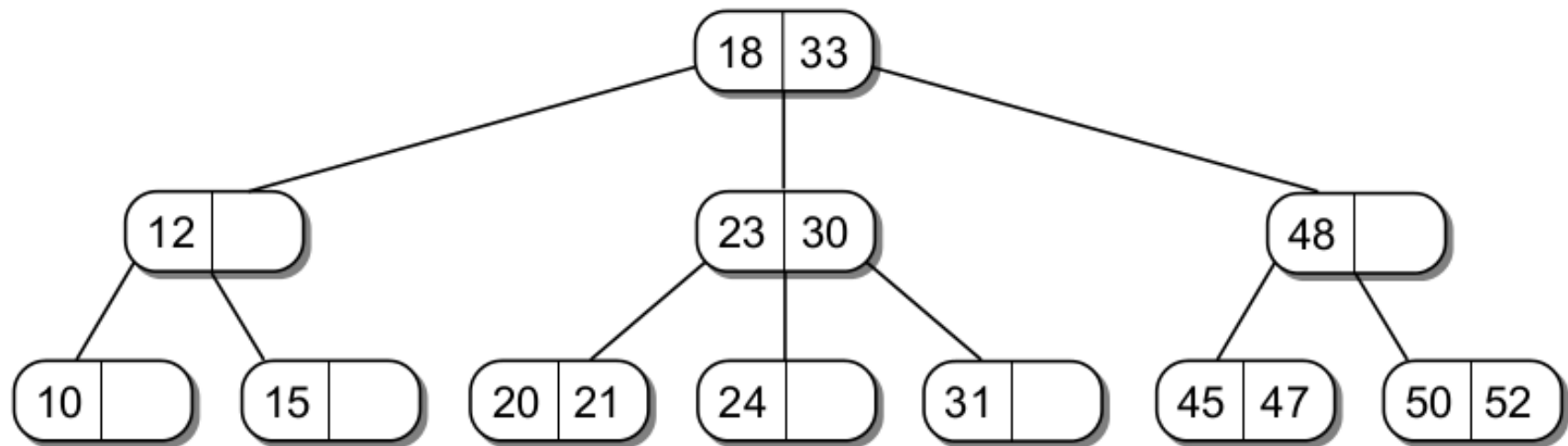
- É uma estrutura alternativa para realização de busca de modo eficiente.

# Introdução

- É uma estrutura alternativa para realização de busca de modo eficiente.
- É uma árvore que pode ter até 3 (três) filhos.

# Introdução

- É uma estrutura alternativa para realização de busca de modo eficiente.
- É uma árvore que pode ter até 3 (três) filhos.



# Propriedades



# Propriedades

- É uma árvore de busca que está **sempre balanceada** e que tem a seguinte definição:

# Propriedades

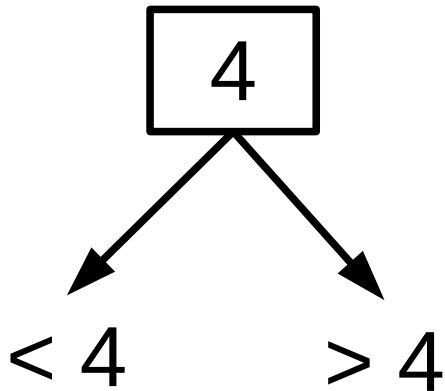
- É uma árvore de busca que está **sempre balanceada** e que tem a seguinte definição:
  - Cada nó pode ter uma ou duas chaves.

# Propriedades

- É uma árvore de busca que está **sempre balanceada** e que tem a seguinte definição:
  - Cada nó pode ter uma ou duas chaves.
  - Cada nó não folha pode ter 2 ou 3 filhos.

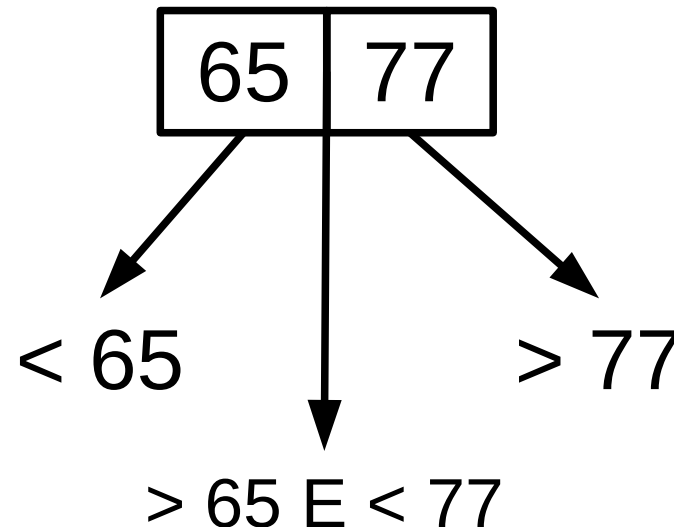
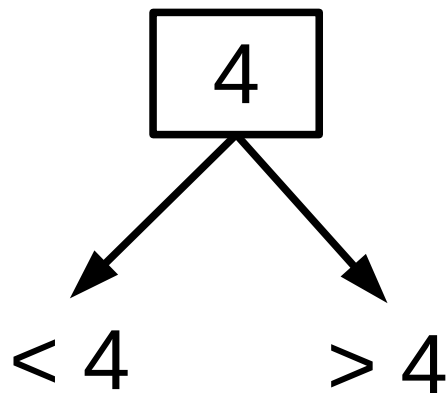
# Propriedades

- É uma árvore de busca que está **sempre balanceada** e que tem a seguinte definição:
  - Cada nó pode ter uma ou duas chaves.
  - Cada nó não folha pode ter 2 ou 3 filhos.



# Propriedades

- É uma árvore de busca que está **sempre balanceada** e que tem a seguinte definição:
  - Cada nó pode ter uma ou duas chaves.
  - Cada nó não folha pode ter 2 ou 3 filhos.



# Propriedades

# Propriedades

- Todos os nós folhas estão no mesmo nível.

# Propriedades

- Todos os nós folhas estão no mesmo nível.
- Portanto, toda árvore 2-3 é perfeitamente balanceada.



# Propriedades

- Todos os nós folhas estão no mesmo nível.
- Portanto, toda árvore 2-3 é perfeitamente balanceada.
- Todo nó interno deve conter dois ou três filhos.

# Propriedades

- Todos os nós folhas estão no mesmo nível.
- Portanto, toda árvore 2-3 é perfeitamente balanceada.
- Todo nó interno deve conter dois ou três filhos.
  - Se o nó tem 1 chave, então ele possui 2 filhos.

# Propriedades

- Todos os nós folhas estão no mesmo nível.
- Portanto, toda árvore 2-3 é perfeitamente balanceada.
- Todo nó interno deve conter dois ou três filhos.
  - Se o nó tem 1 chave, então ele possui 2 filhos.
  - Se o nó tem 2 chaves, então ele possui 3 filhos.

# Propriedades (cont.)

# Propriedades (cont.)

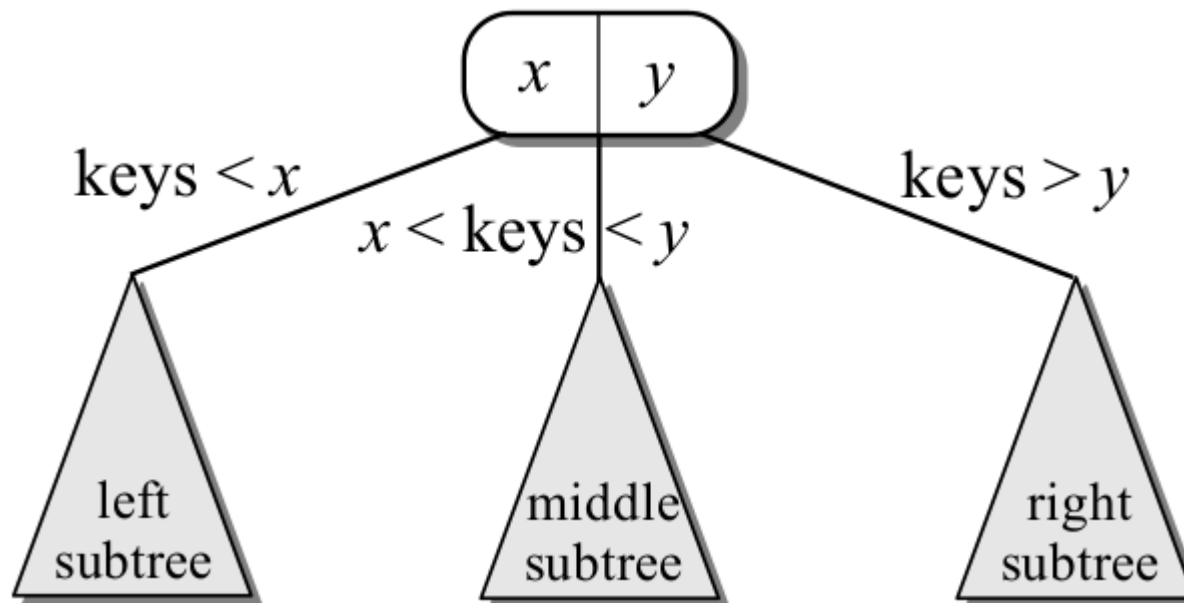
- As árvores 2-3 mantêm as propriedades das árvores binárias de busca.

# Propriedades (cont.)

- As árvores 2-3 mantêm as propriedades das árvores binárias de busca.
  - Todas as chaves que sejam menores do que a primeira chave de  $V$  são armazenadas na subárvore à esquerda de  $V$ .

# Propriedades (cont.)

- As árvores 2-3 mantêm as propriedades das árvores binárias de busca.
  - Todas as chaves que sejam menores do que a primeira chave de  $V$  são armazenadas na subárvore à esquerda de  $V$ .



# Propriedades (cont.)



# Propriedades (cont.)

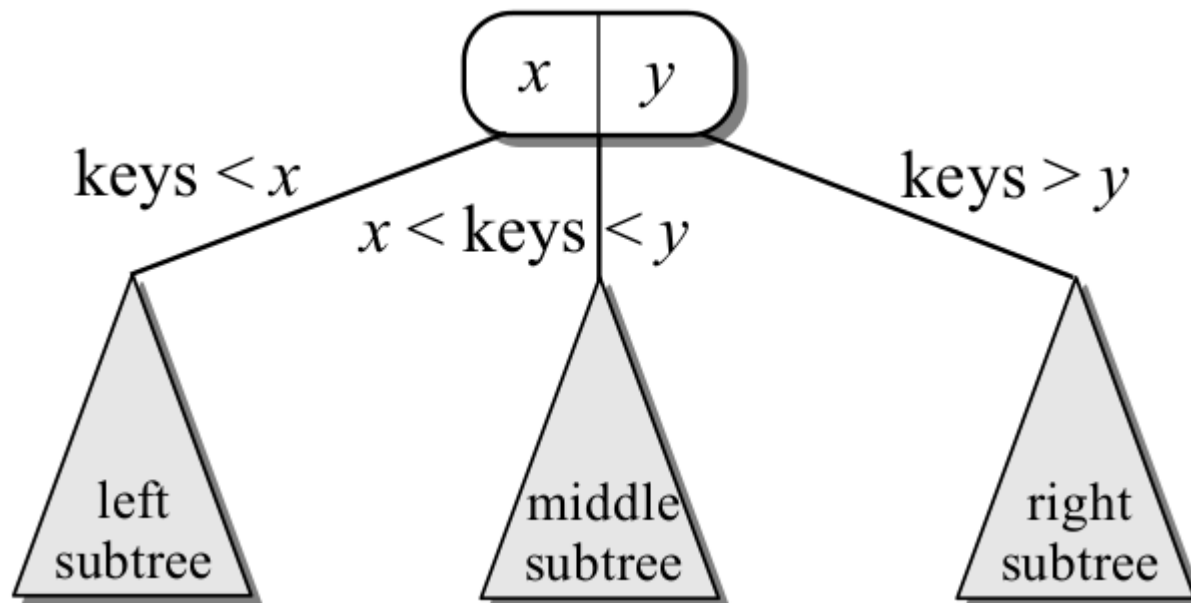
- As árvore 2-3 mantêm as propriedades das árvores binárias de busca.

# Propriedades (cont.)

- As árvore 2-3 mantêm as propriedades das árvores binárias de busca.
  - Se o nó tem dois filhos, então todas as chaves maiores do que a primeira chave do nó  $V$  são armazenadas no meio da subárvore de  $V$ .

# Propriedades (cont.)

- As árvore 2-3 mantêm as propriedades das árvores binárias de busca.
  - Se o nó tem dois filhos, então todas as chaves maiores do que a primeira chave do nó  $V$  são armazenadas no meio da subárvore de  $V$ .



# Propriedades (cont.)

# Propriedades (cont.)

- As árvore 2-3 mantêm as propriedades das árvores binárias de busca.

# Propriedades (cont.)

- As árvore 2-3 mantêm as propriedades das árvores binárias de busca.
  - Se o nó tiver três filhos:

# Propriedades (cont.)

- As árvore 2-3 mantêm as propriedades das árvores binárias de busca.
  - Se o nó tiver três filhos:
    - Todas as chaves que sejam maiores que a primeira chave do  $V$ , mas menores do que o segundo nó, são armazenados no meio da subárvore  $V$ .

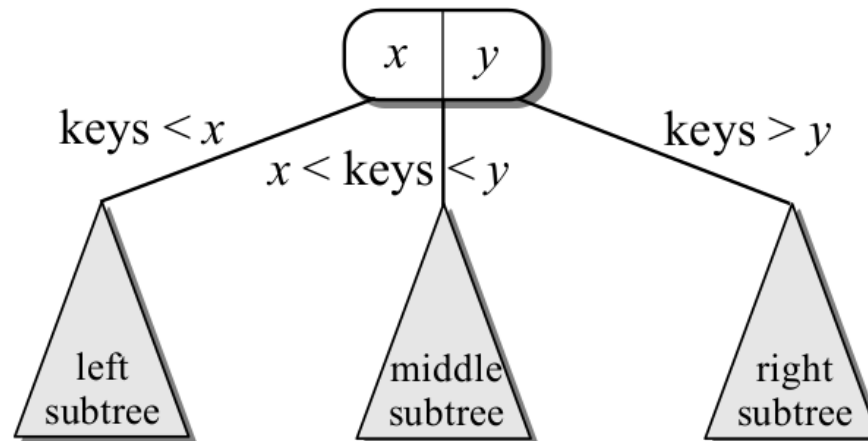
# Propriedades (cont.)

- As árvore 2-3 mantêm as propriedades das árvores binárias de busca.
  - Se o nó tiver três filhos:
    - Todas as chaves que sejam maiores que a primeira chave do V, mas menores do que o segundo nó, são armazenados no meio da subárvore V.
    - Todas as chaves maiores do que a segunda chave são armazenados na subárvore à direita.



# Propriedades (cont.)

- As árvore 2-3 mantêm as propriedades das árvores binárias de busca.
  - Se o nó tiver três filhos:
    - Todas as chaves que sejam maiores que a primeira chave do  $V$ , mas menores do que o segundo nó, são armazenados no meio da subárvore  $V$ .
    - Todas as chaves maiores do que a segunda chave são armazenados na subárvore à direita.



# Busca

# Busca

- A busca nas árvores 2-3 é bastante similar da busca nas ABBs.

# Busca

- A busca nas árvores 2-3 é bastante similar da busca nas ABBs.
- Nós devemos iniciar na raiz e seguir a ramificação apropriada baseada no valor da chave alvo.

# Busca

- A busca nas árvores 2-3 é bastante similar da busca nas ABBs.
- Nós devemos iniciar na raiz e seguir a ramificação apropriada baseada no valor da chave alvo.
- A única diferença é que nós temos que comparar o alvo contra duas chaves, caso o nó possua duas chaves, e então devemos escolher entre as três possíveis subárvores.

# Busca

# Busca

- Assim como numa ABB, uma busca com sucesso levará a chave em um dos nós da árvore, enquanto uma busca sem sucesso levará a um link *null*.

# Busca

- Assim como numa ABB, uma busca com sucesso levará a chave em um dos nós da árvore, enquanto uma busca sem sucesso levará a um link *null*.
- Este link *null* será sempre em um nó folha.



# Busca

- Assim como numa ABB, uma busca com sucesso levará a chave em um dos nós da árvore, enquanto uma busca sem sucesso levará a um link *null*.
- Este link *null* será sempre em um nó folha.
- A razão para isso é que se um nó interior contém uma chave, ele sempre contém dois nós.

# Busca

# Busca

- Algoritmo (busca k):

# Busca

- Algoritmo (busca k):
  - Se  $k$  for igual a alguma chave, então encontramos.

# Busca

- Algoritmo (busca k):
  - Se  $k$  for igual a alguma chave, então encontramos.
  - Se  $k$  é menor do que  $a$ , então realizamos a busca no filho à esquerda.

# Busca

- Algoritmo (busca k):
  - Se  $k$  for igual a alguma chave, então encontramos.
  - Se  $k$  é menor do que  $a$ , então realizamos a busca no filho à esquerda.
  - Se  $k$  está entre  $a$  e  $b$ , então nós realizamos a busca no filho do meio.

# Busca

- Algoritmo (busca k):
  - Se  $k$  for igual a alguma chave, então encontramos.
  - Se  $k$  é menor do que  $a$ , então realizamos a busca no filho à esquerda.
  - Se  $k$  está entre  $a$  e  $b$ , então nós realizamos a busca no filho do meio.
  - Se  $k$  é maior que  $b$ , então realizamos a busca no filho à direita.

# Inserção



# Inserção

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

# Inserção

- O processo de inserção é parecido com o método de inserção das árvores binárias, mas com algumas adaptações.
- O primeiro passo é buscar pelo elemento na árvore.

# Inserção

- O processo de inserção é parecido com o método de inserção das árvores binárias, mas com algumas adaptações.
- O primeiro passo é buscar pelo elemento na árvore.
- A busca por um nó não-existente (novo nó) levará a um nó folha.

# Inserção

- O processo de inserção é parecido com o método de inserção das árvores binárias, mas com algumas adaptações.
- O primeiro passo é buscar pelo elemento na árvore.
- A busca por um nó não-existente (novo nó) levará a um nó folha.
- O próximo passo é verificar se existe espaço nesse nó folha.

# Inserção

# Inserção

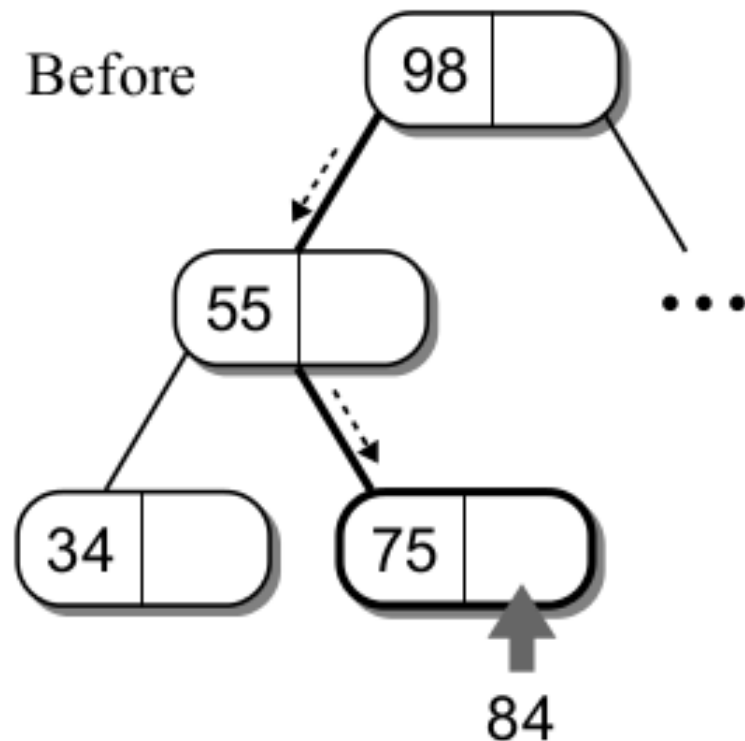
- Se o nó folha contém apenas uma chave, então podemos inserir a chave nesse nó.

# Inserção

- Se o nó folha contém apenas uma chave, então podemos inserir a chave nesse nó.
- Exemplo: inserir o elemento 84

# Inserção

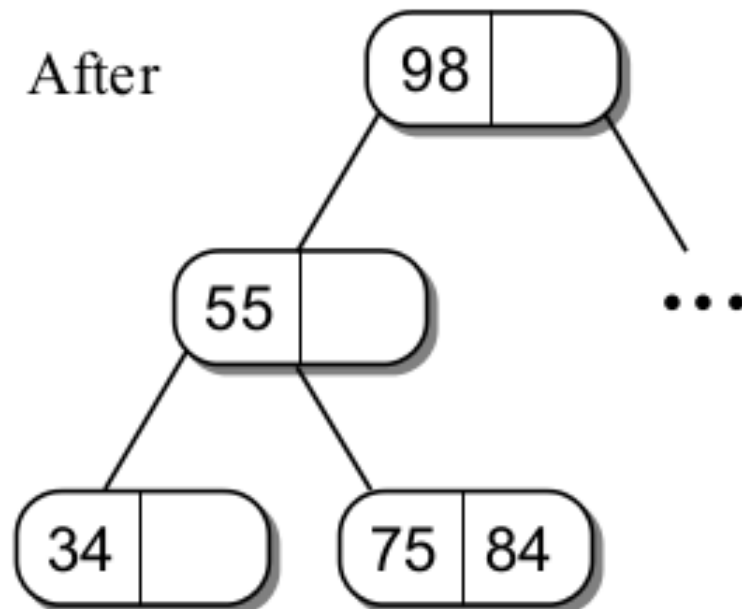
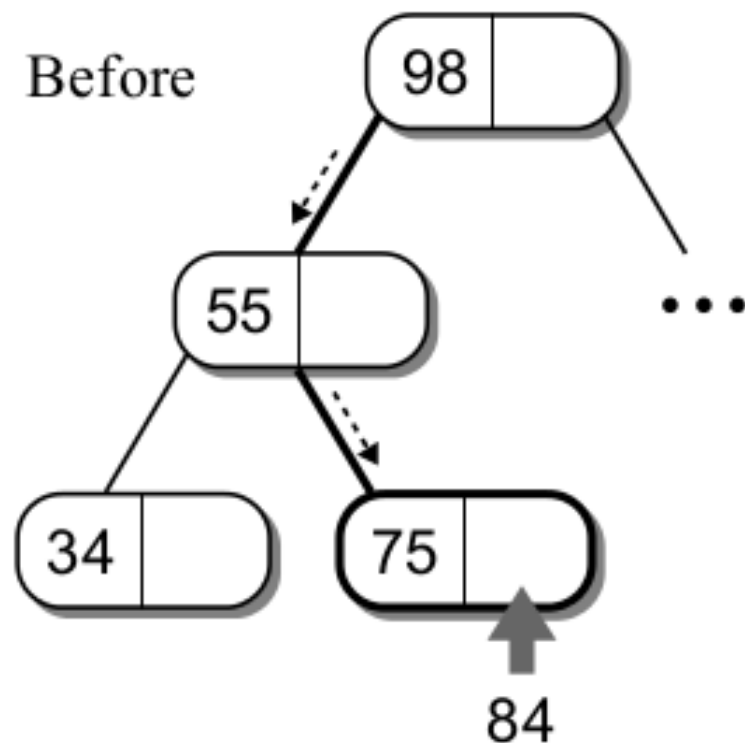
- Se o nó folha contém apenas uma chave, então podemos inserir a chave nesse nó.
- Exemplo: inserir o elemento 84





# Inserção

- Se o nó folha contém apenas uma chave, então podemos inserir a chave nesse nó.
- Exemplo: inserir o elemento 84



# Inserção

# Inserção

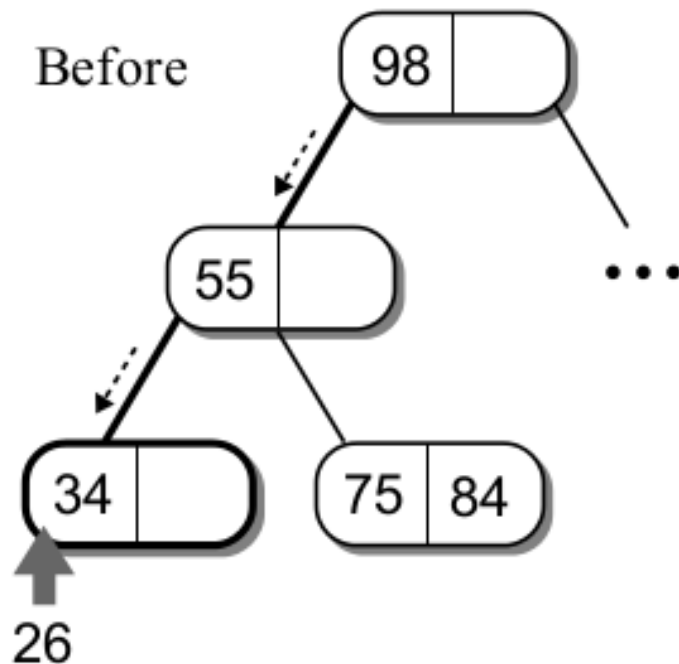
- Mas o que aconteceria se a chave do novo elemento é menor do que a chave armazenada no nó folha?

# Inserção

- Mas o que aconteceria se a chave do novo elemento é menor do que a chave armazenada no nó folha?
- Vamos inserir o elemento 26 na árvore abaixo:

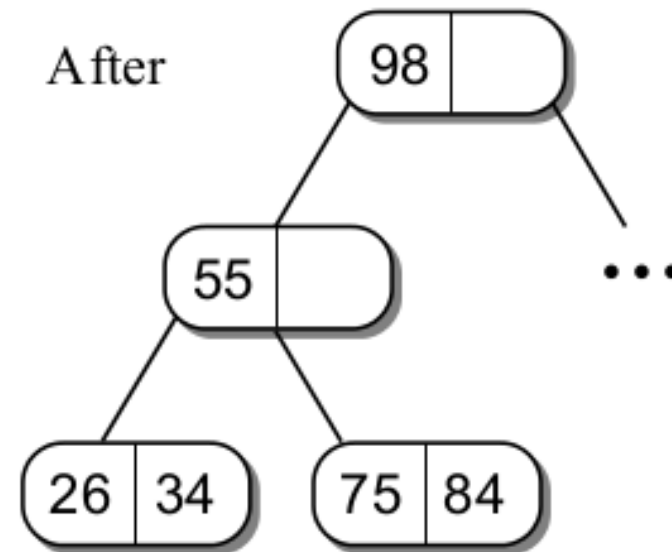
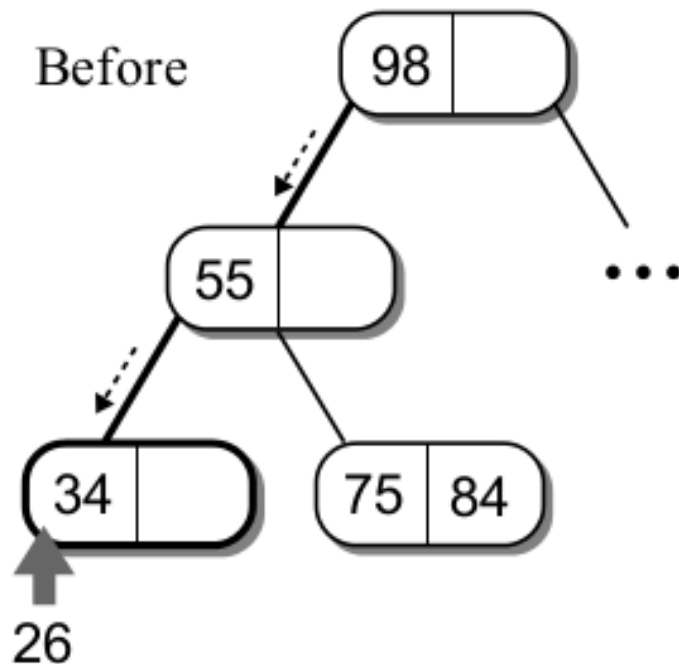
# Inserção

- Mas o que aconteceria se a chave do novo elemento é menor do que a chave armazenada no nó folha?
- Vamos inserir o elemento 26 na árvore abaixo:



# Inserção

- Mas o que aconteceria se a chave do novo elemento é menor do que a chave armazenada no nó folha?
- Vamos inserir o elemento 26 na árvore abaixo:



# Inserção

# Inserção

- As coisas se tornam um pouco mais complicadas (interessantes) quando o nó folha está cheio.

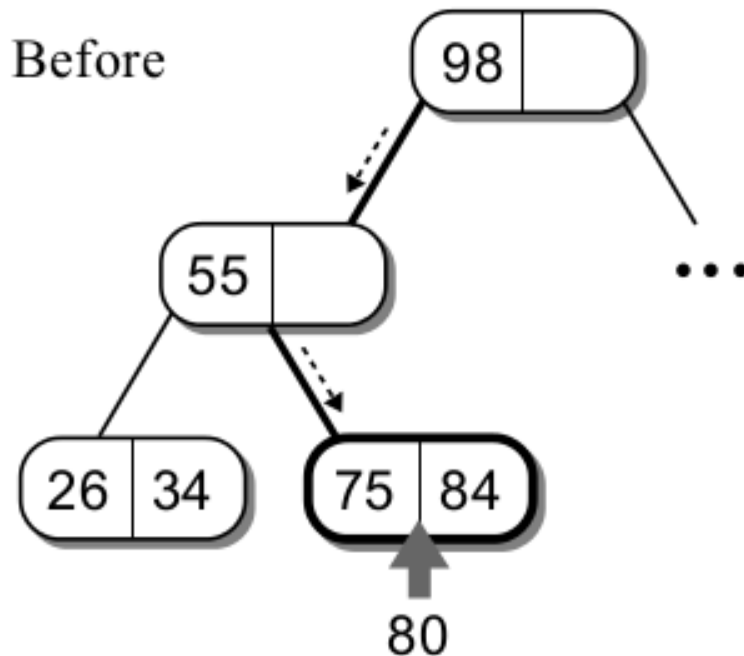


# Inserção

- As coisas se tornam um pouco mais complicadas (interessantes) quando o nó folha está cheio.
- Suponha que vamos inserir o nó 80 na árvore:

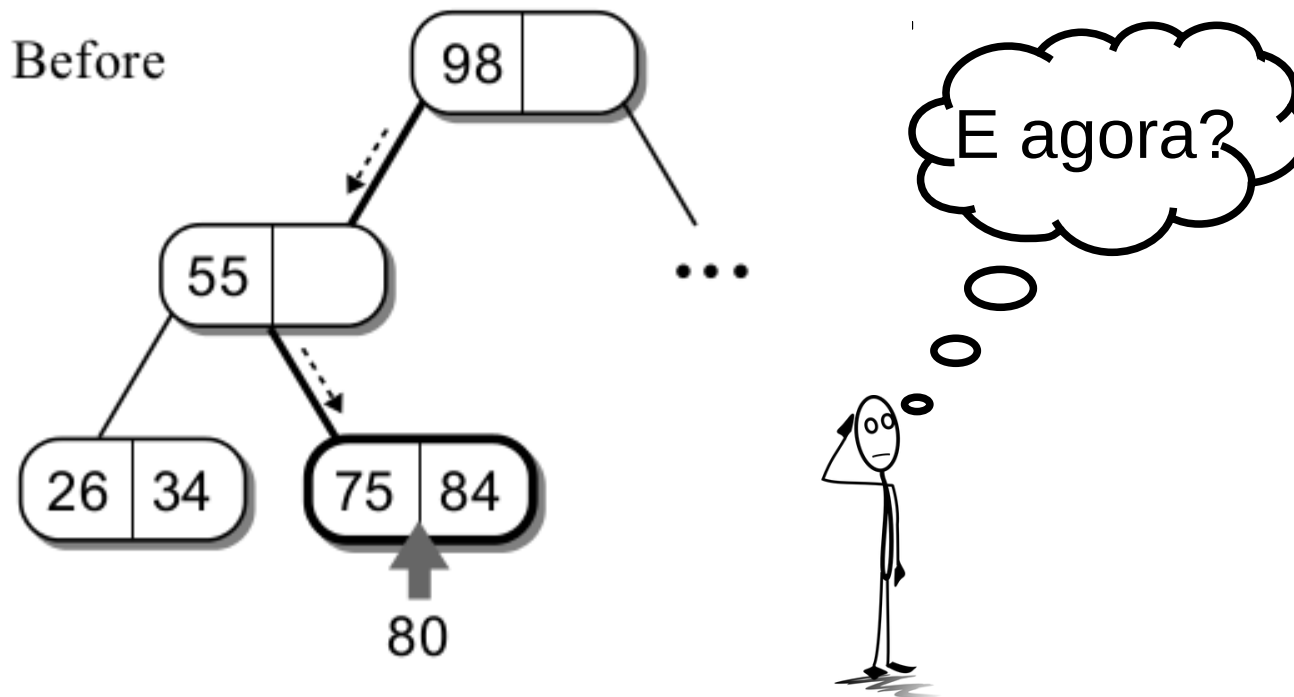
# Inserção

- As coisas se tornam um pouco mais complicadas (interessantes) quando o nó folha está cheio.
- Suponha que vamos inserir o nó 80 na árvore:



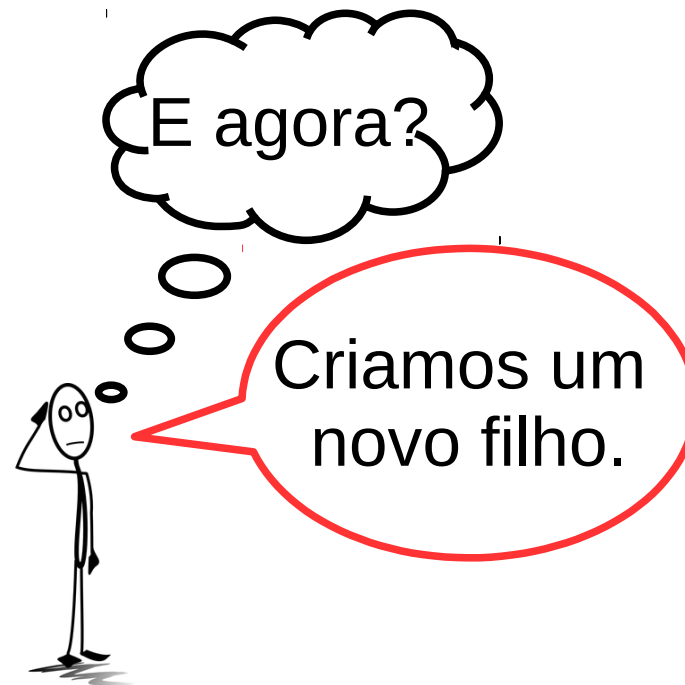
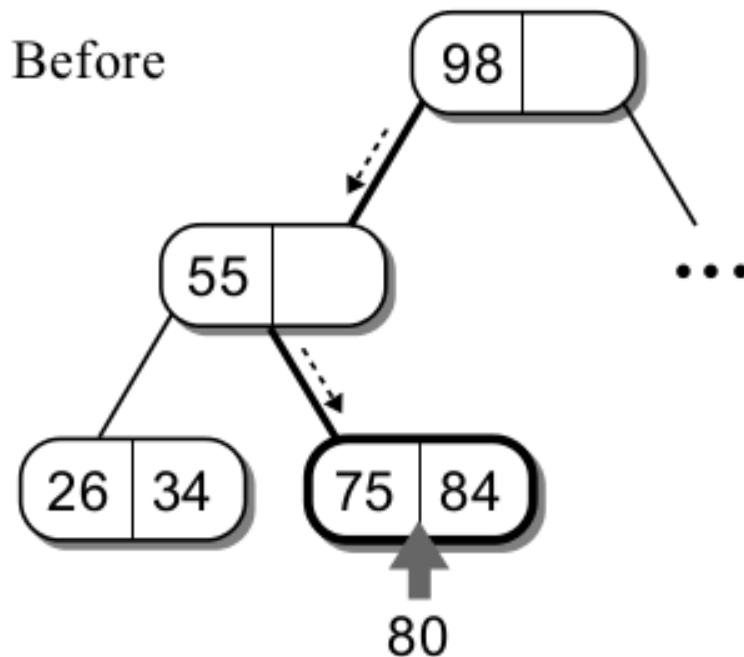
# Inserção

- As coisas se tornam um pouco mais complicadas (interessantes) quando o nó folha está cheio.
- Suponha que vamos inserir o nó 80 na árvore:



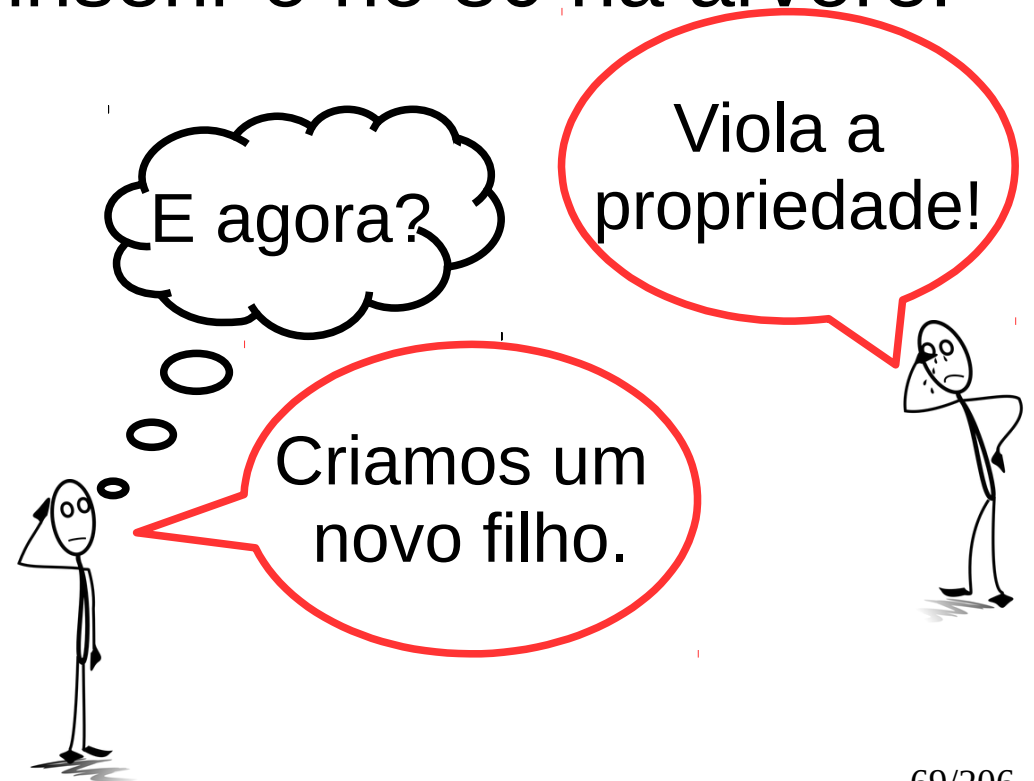
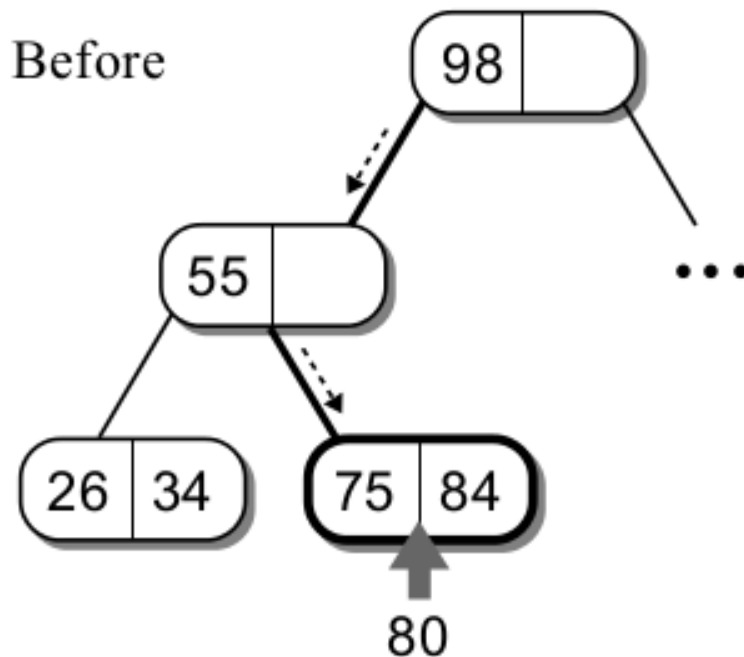
# Inserção

- As coisas se tornam um pouco mais complicadas (interessantes) quando o nó folha está cheio.
- Suponha que vamos inserir o nó 80 na árvore:



# Inserção

- As coisas se tornam um pouco mais complicadas (interessantes) quando o nó folha está cheio.
- Suponha que vamos inserir o nó 80 na árvore:



# Inserção

# Inserção

- Será necessária uma operação de divisão em duas etapas.

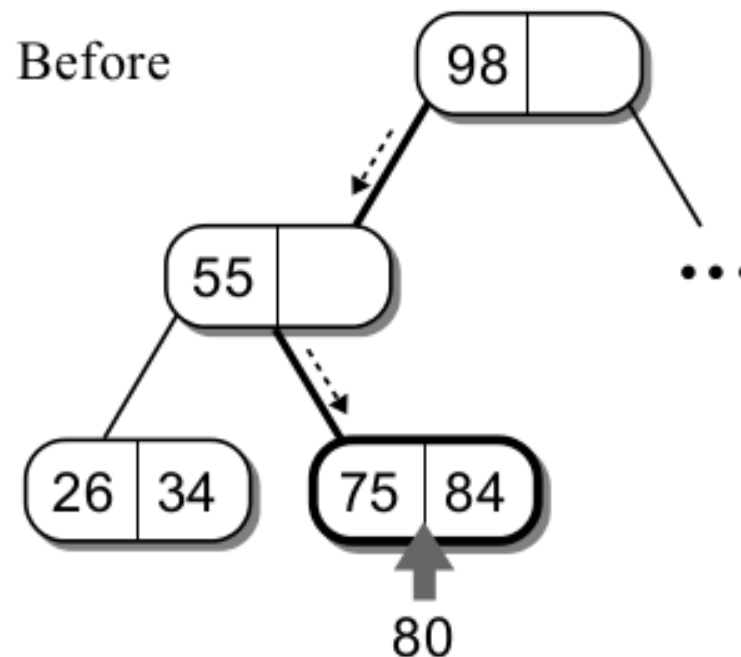
# Inserção

- Será necessária uma operação de divisão em duas etapas.
- Primeiro, criamos um novo nó e então comparamos a nova chave com as duas outras chaves do nó folha (75 e 84).



# Inserção

- Será necessária uma operação de divisão em duas etapas.
- Primeiro, criamos um novo nó e então comparamos a nova chave com as duas outras chaves do nó folha (75 e 84).



# Inserção

# Inserção

- O menor valor é inserido no nó original.

# Inserção

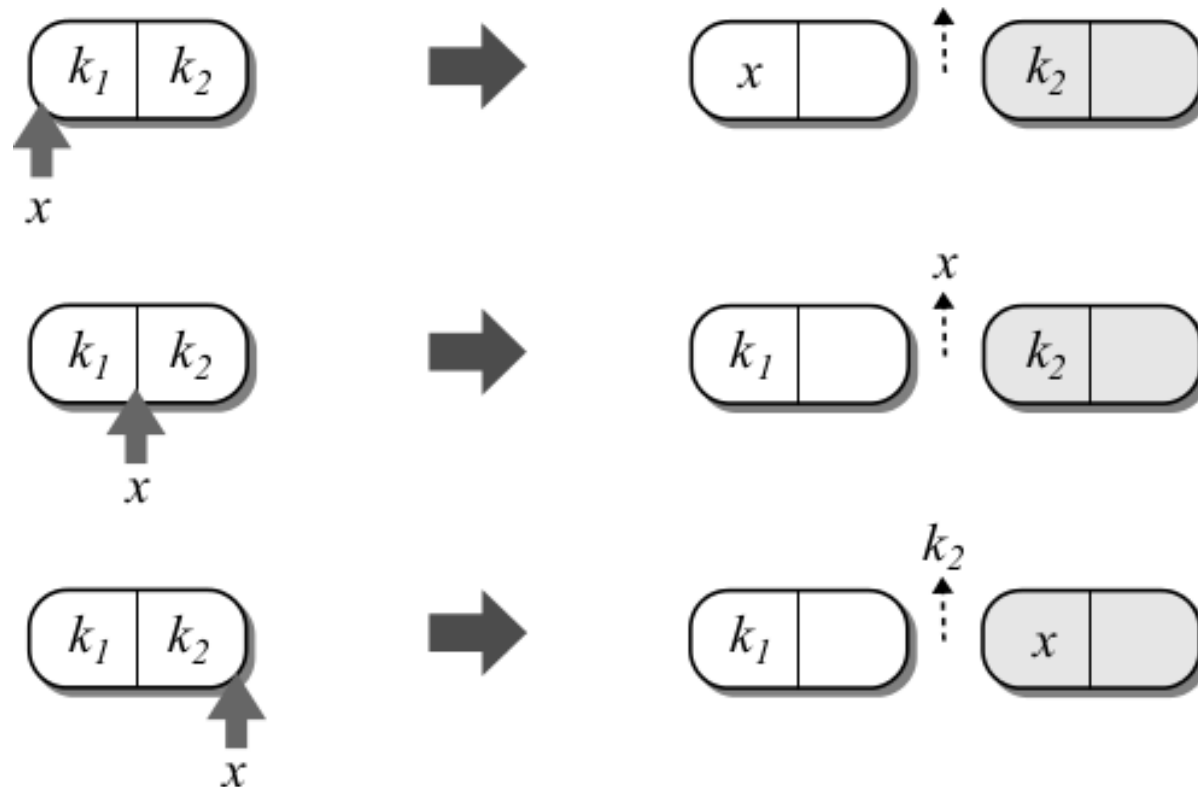
- O menor valor é inserido no nó original.
- O maior valor é inserido no novo nó.

# Inserção

- O menor valor é inserido no nó original.
- O maior valor é inserido no novo nó.
- O nó do meio passará a ser o nó pai.

# Inserção

- O menor valor é inserido no nó original.
- O maior valor é inserido no novo nó.
- O nó do meio passará a ser o nó pai.



# Inserção

# Inserção

- Quando um nó é promovido para o nível de pai, ele pode ser inserido de modo similar a inserção de nó folha.



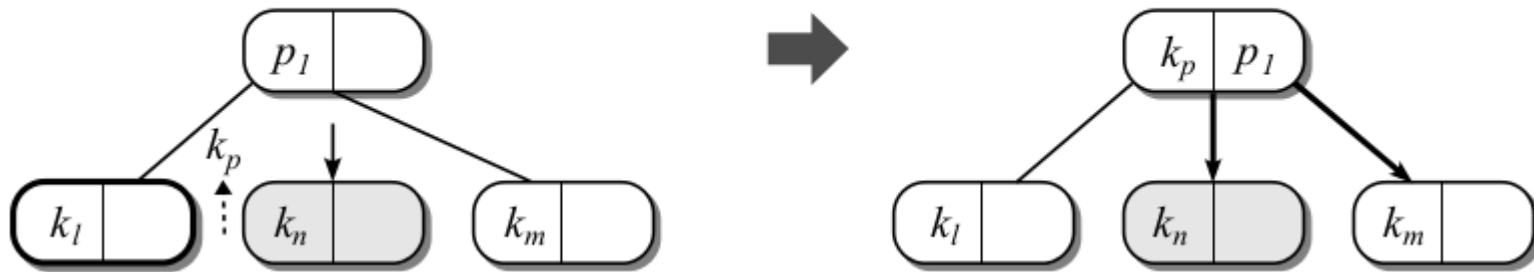
# Inserção

- Quando um nó é promovido para o nível de pai, ele pode ser inserido de modo similar a inserção de nó folha.
- O procedimento é simples se houver espaço.

# Inserção

- Quando um nó é promovido para o nível de pai, ele pode ser inserido de modo similar a inserção de nó folha.
- O procedimento é simples se houver espaço.

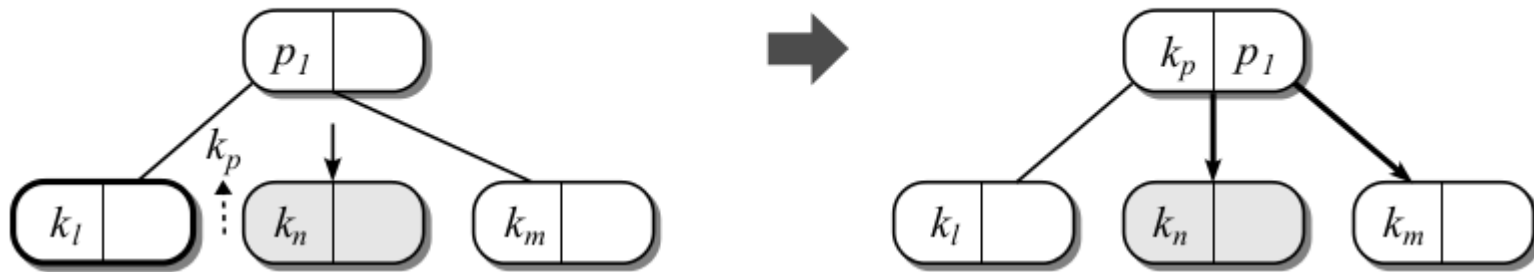
(a) Splitting the left child.



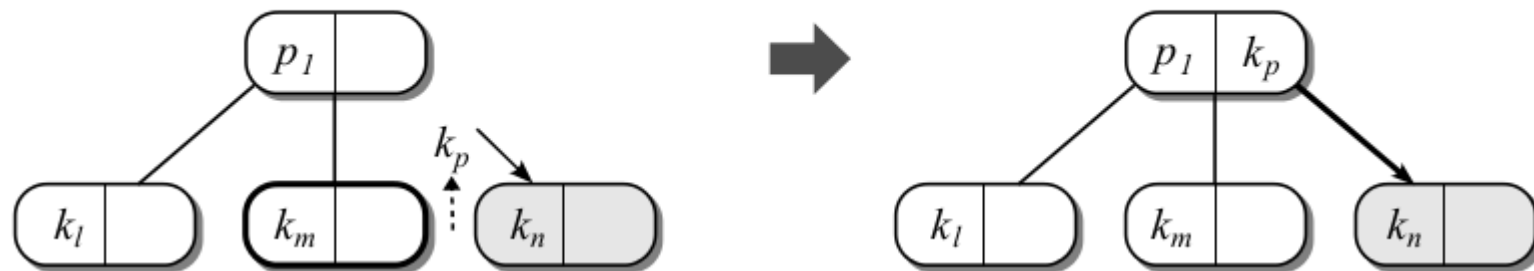
# Inserção

- Quando um nó é promovido para o nível de pai, ele pode ser inserido de modo similar a inserção de nó folha.
- O procedimento é simples se houver espaço.

(a) Splitting the left child.



(b) Splitting the middle child.



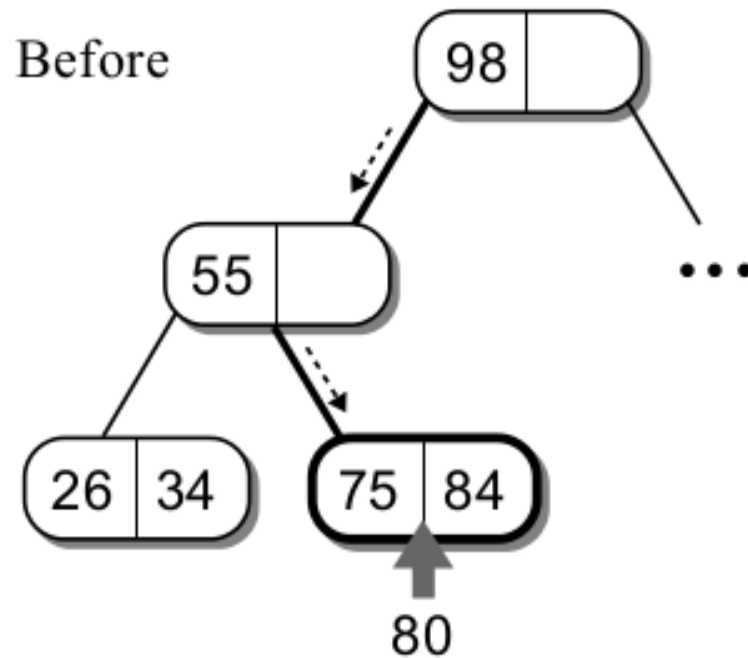
# Inserção

# Inserção

- Inserir o elemento 80 (continuação):

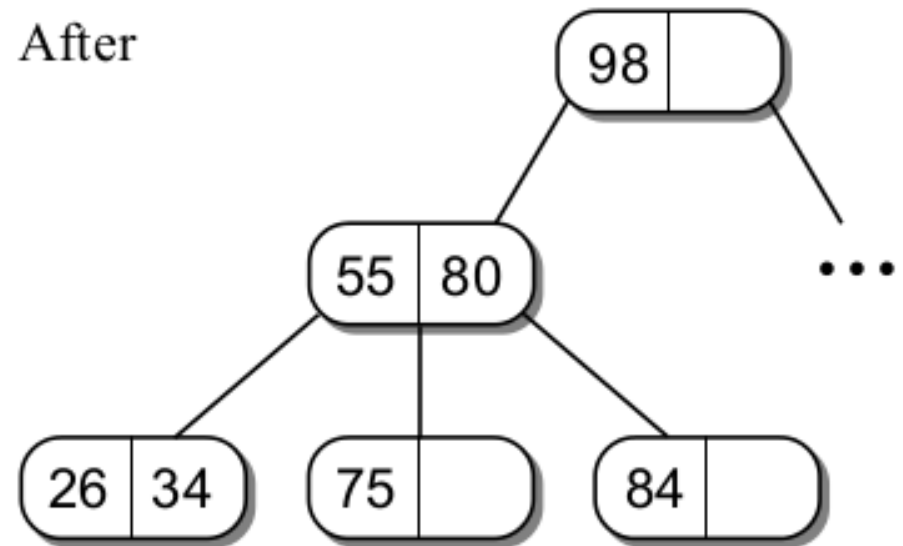
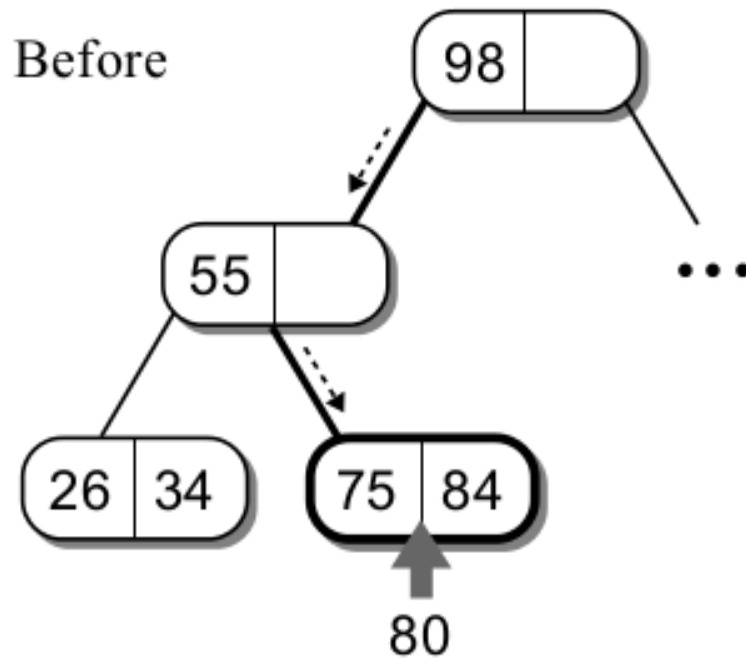
# Inserção

- Inserir o elemento 80 (continuação):



# Inserção

- Inserir o elemento 80 (continuação):



# Inserção



# Inserção

- O que aconteceria se o nó superior (pai) já estivesse cheio?

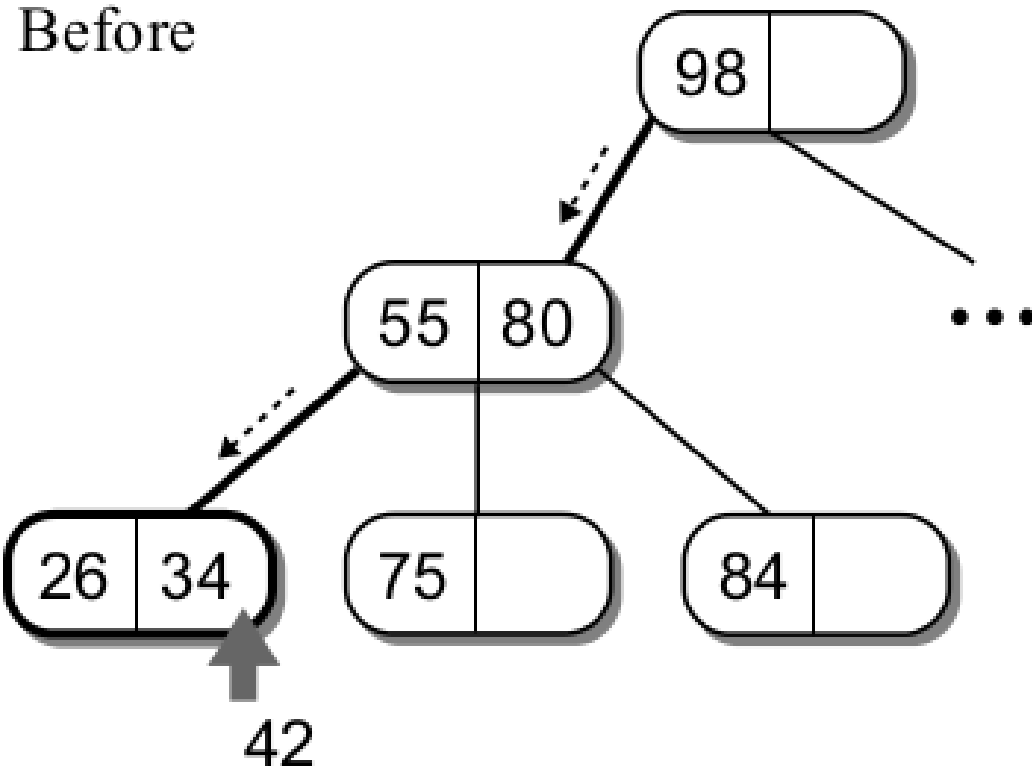
# Inserção

- O que aconteceria se o nó superior (pai) já estivesse cheio?
- Suponha a inserção do nó 42 na árvore abaixo:

# Inserção

- O que aconteceria se o nó superior (pai) já estivesse cheio?
- Suponha a inserção do nó 42 na árvore abaixo:

Before

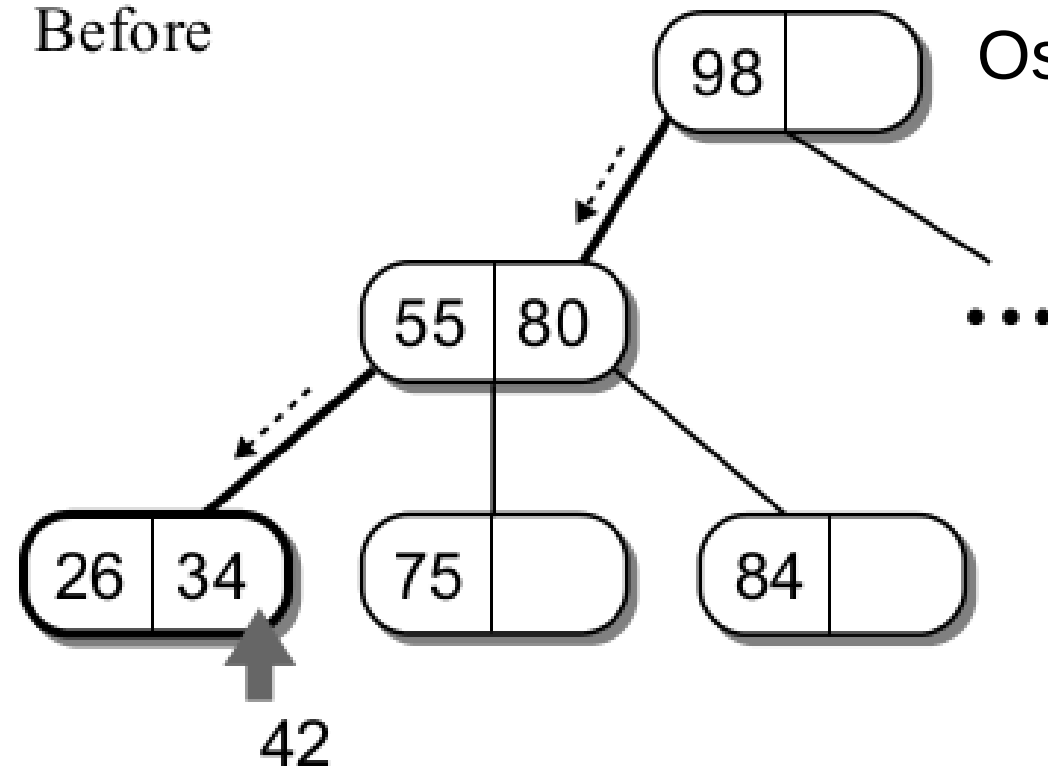


# Inserção

- O que aconteceria se o nó superior (pai) já estivesse cheio?
- Suponha a inserção do nó 42 na árvore abaixo:

Before

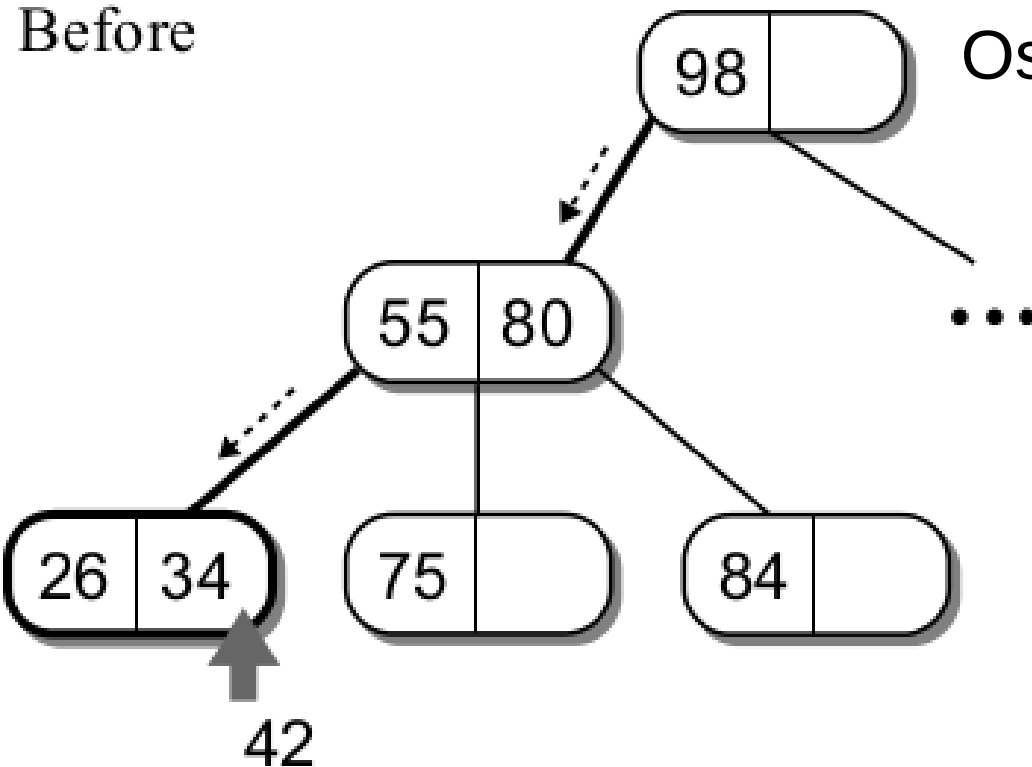
Os nós 26 e 34 devem ser divididos.



# Inserção

- O que aconteceria se o nó superior (pai) já estivesse cheio?
- Suponha a inserção do nó 42 na árvore abaixo:

Before

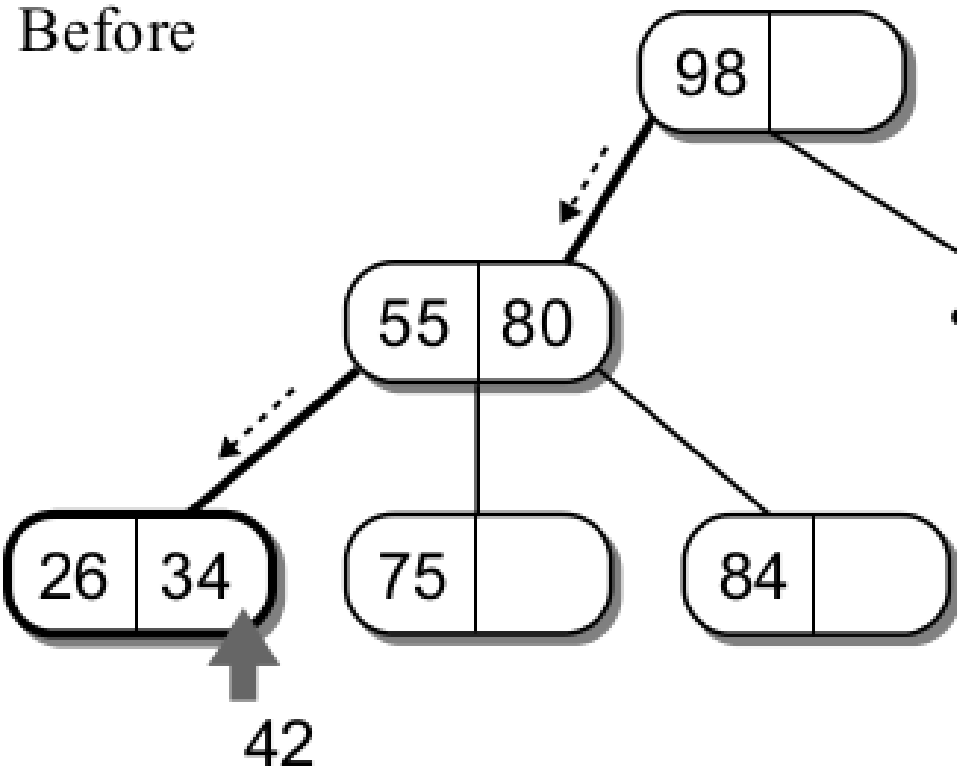


Os nós 26 e 34 devem ser divididos.  
O nó 34 deve ser promovido.

# Inserção

- O que aconteceria se o nó superior (pai) já estivesse cheio?
- Suponha a inserção do nó 42 na árvore abaixo:

Before

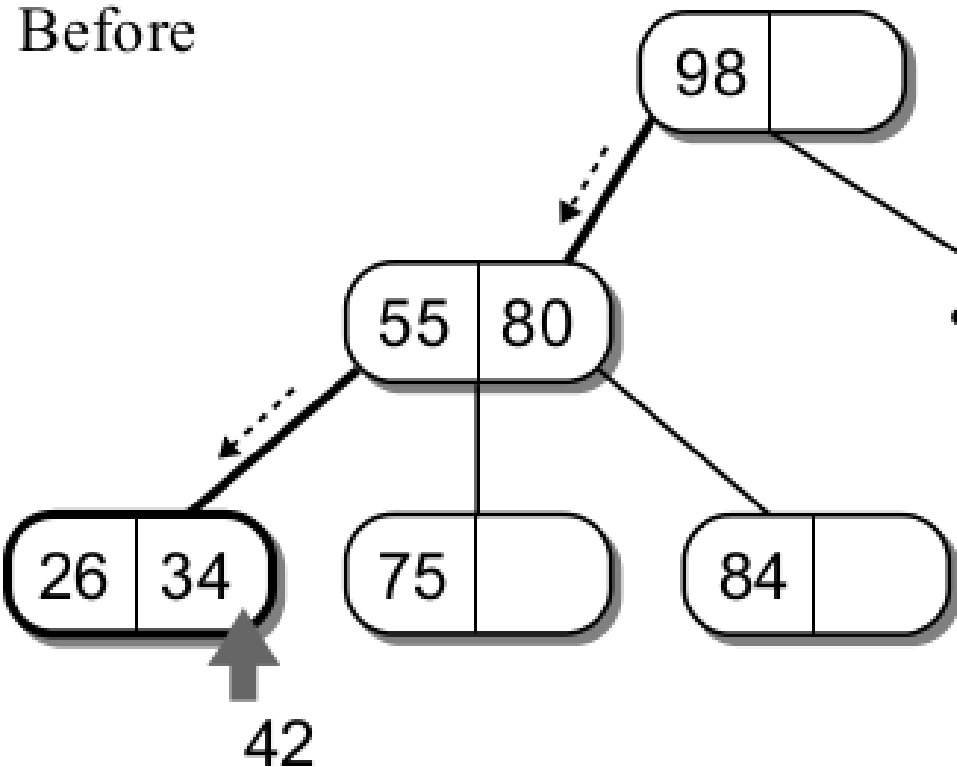


Os nós 26 e 34 devem ser divididos.  
O nó 34 deve ser promovido.  
O nó pai tem duas chaves (55,80).  
...

# Inserção

- O que aconteceria se o nó superior (pai) já estivesse cheio?
- Suponha a inserção do nó 42 na árvore abaixo:

Before

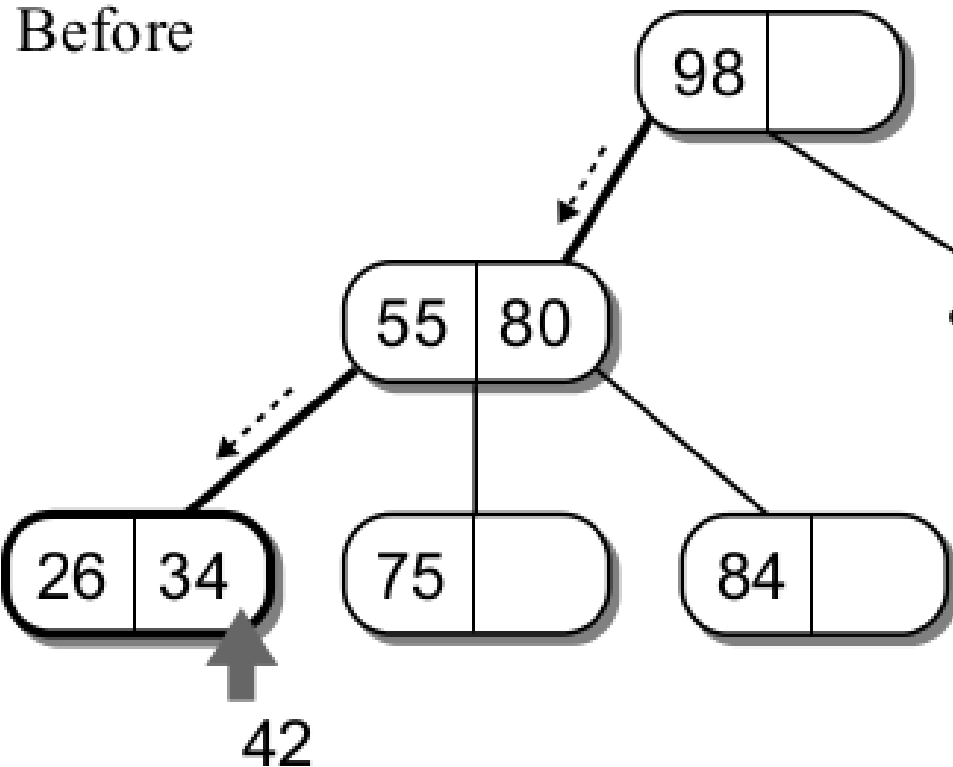


Os nós 26 e 34 devem ser divididos.  
O nó 34 deve ser promovido.  
O nó pai tem duas chaves (55,80).  
...  
Esse nó deverá ser dividido.

# Inserção

- O que aconteceria se o nó superior (pai) já estivesse cheio?
- Suponha a inserção do nó 42 na árvore abaixo:

Before



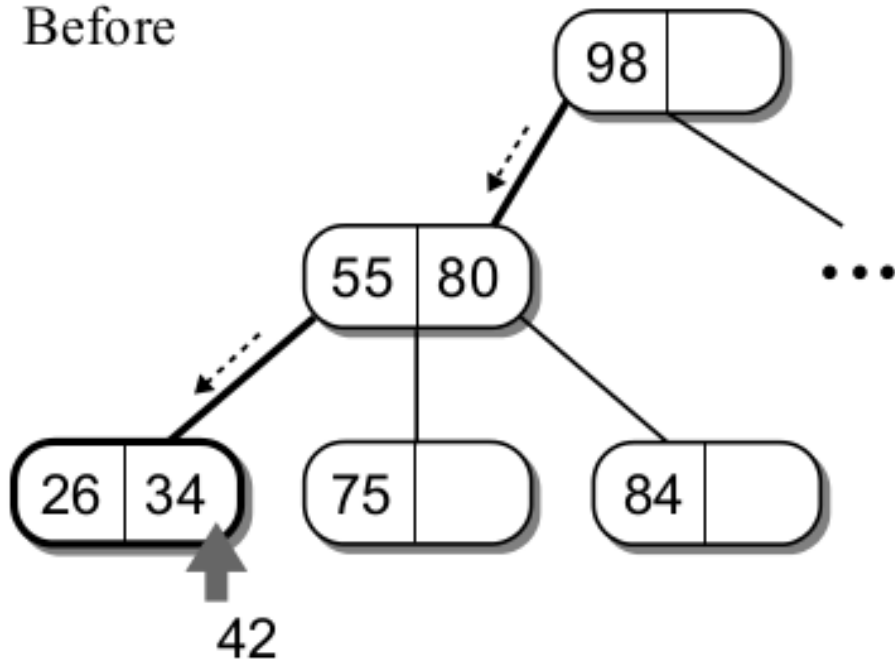
Os nós 26 e 34 devem ser divididos.  
O nó 34 deve ser promovido.  
O nó pai tem duas chaves (55,80).  
...  
Esse nó deverá ser dividido.  
Processo será repetido até a raiz.



# Inserção

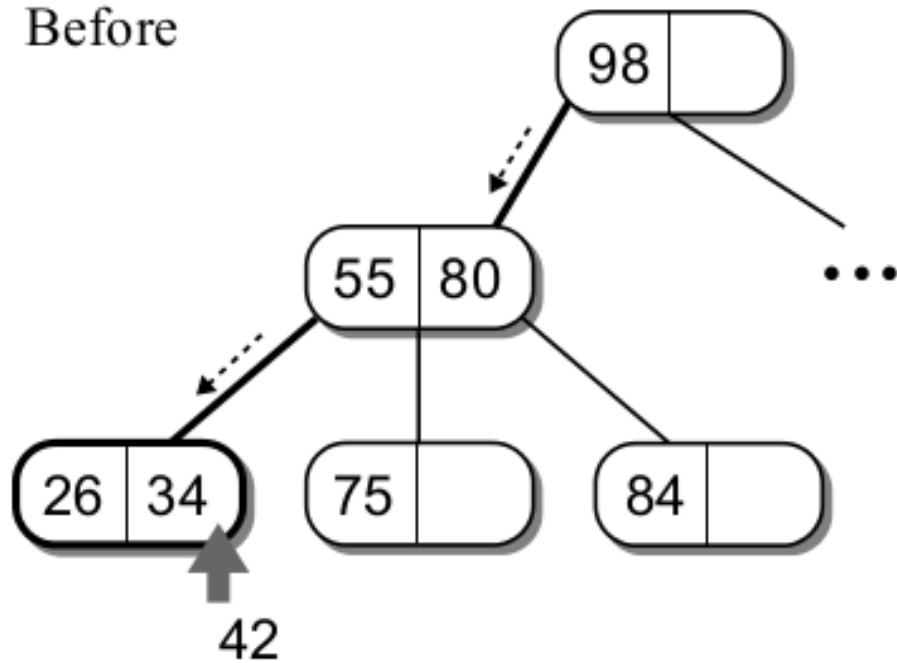
# Inserção

Before

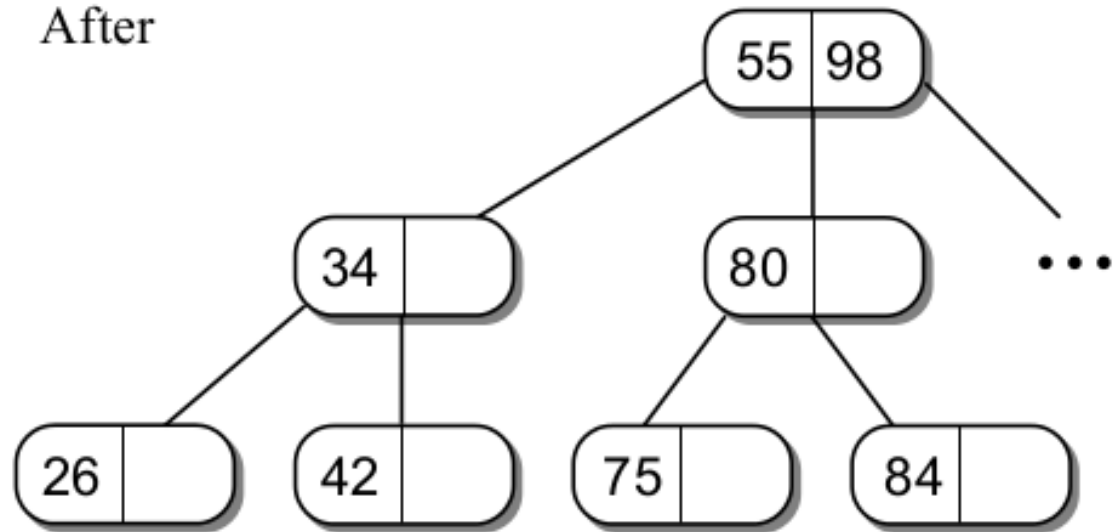


# Inserção

Before



After



# Inserção

# Inserção

- Quando o novo nó precisa ser dividido, um novo nó raiz é criado.

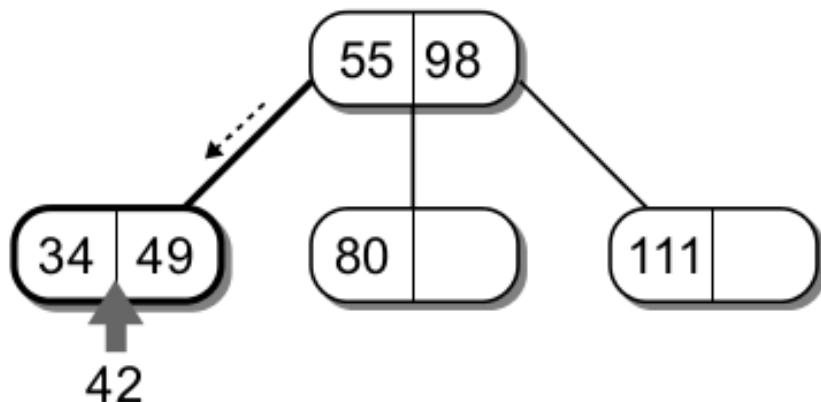
# Inserção

- Quando o novo nó precisa ser dividido, um novo nó raiz é criado.
- O nó raiz original se torna o filho esquerdo e o novo nó se torna o filho do meio.

# Inserção

- Quando o novo nó precisa ser dividido, um novo nó raiz é criado.
- O nó raiz original se torna o filho esquerdo e o novo nó se torna o filho do meio.

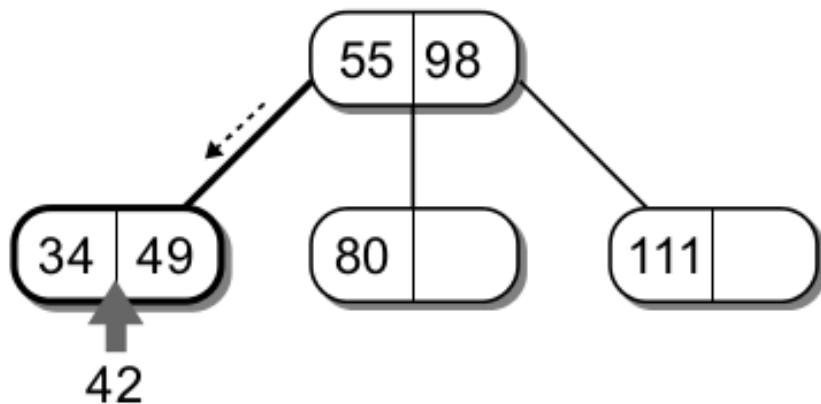
Before



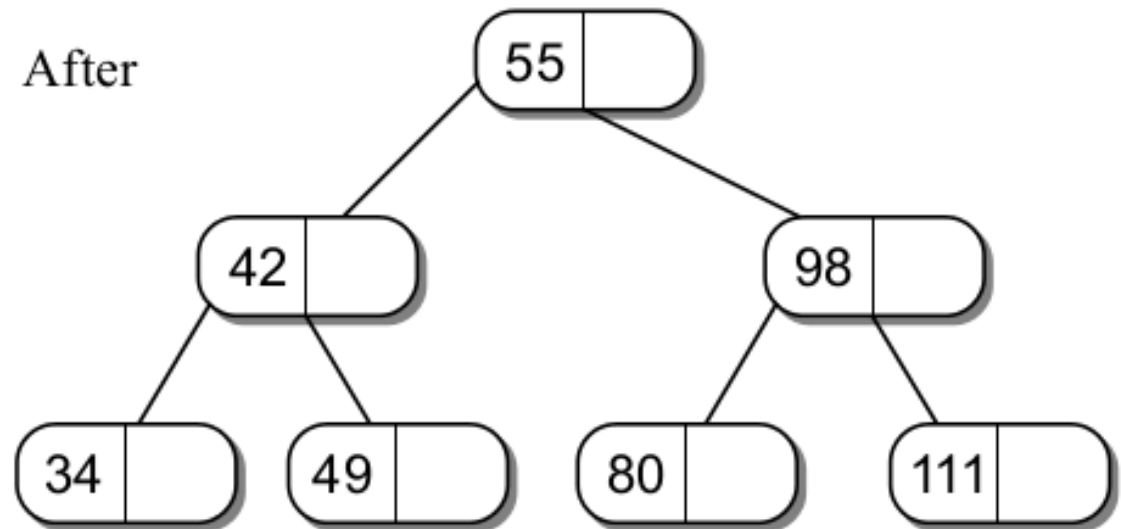
# Inserção

- Quando o novo nó precisa ser dividido, um novo nó raiz é criado.
- O nó raiz original se torna o filho esquerdo e o novo nó se torna o filho do meio.

Before



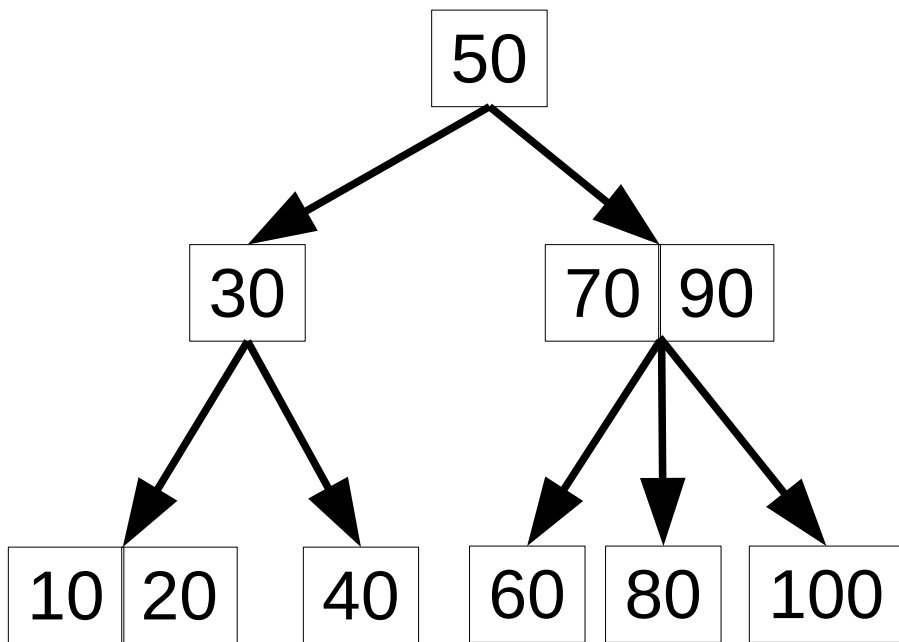
After





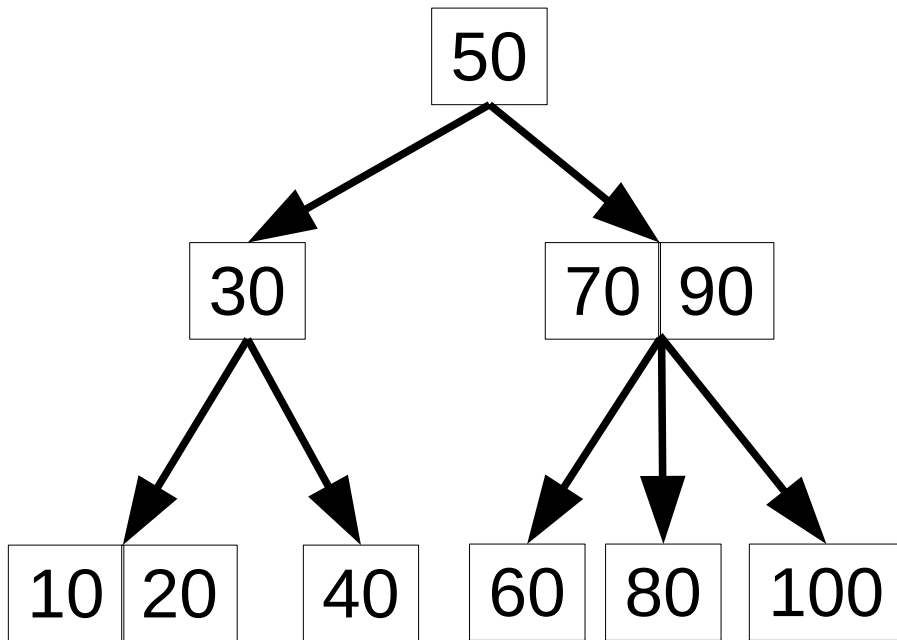
# Remoção

# Remoção



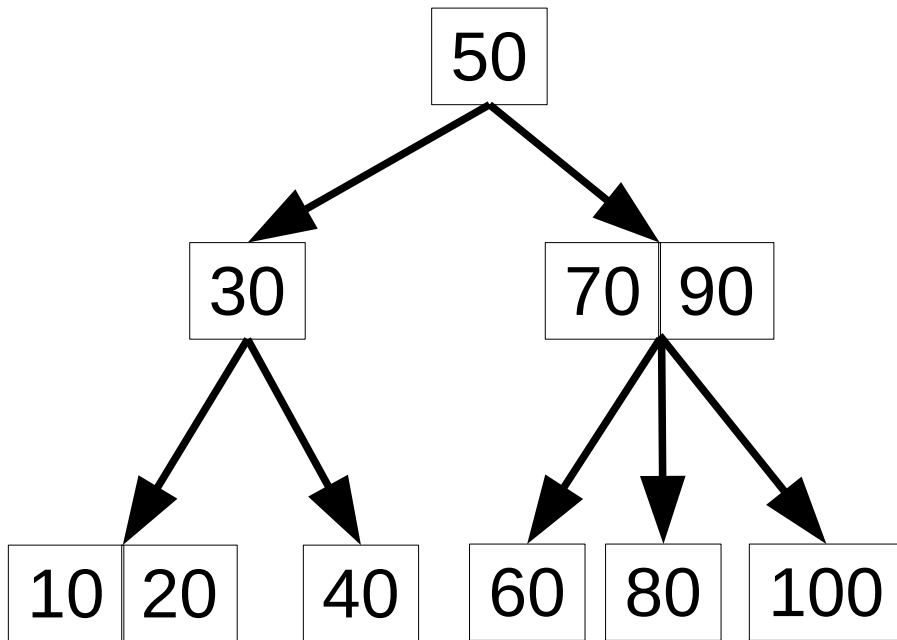
# Remoção

- Remover nó 70.



# Remoção

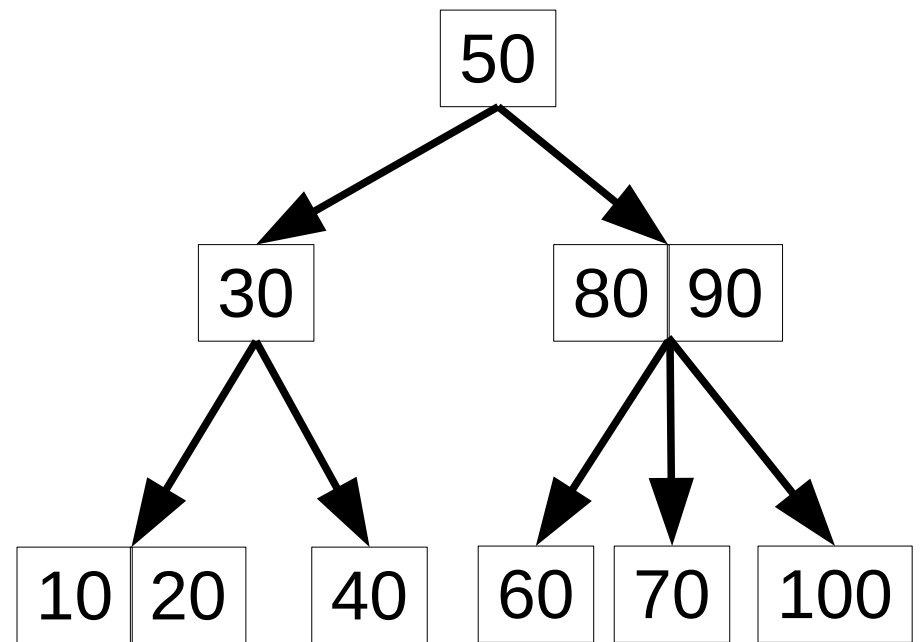
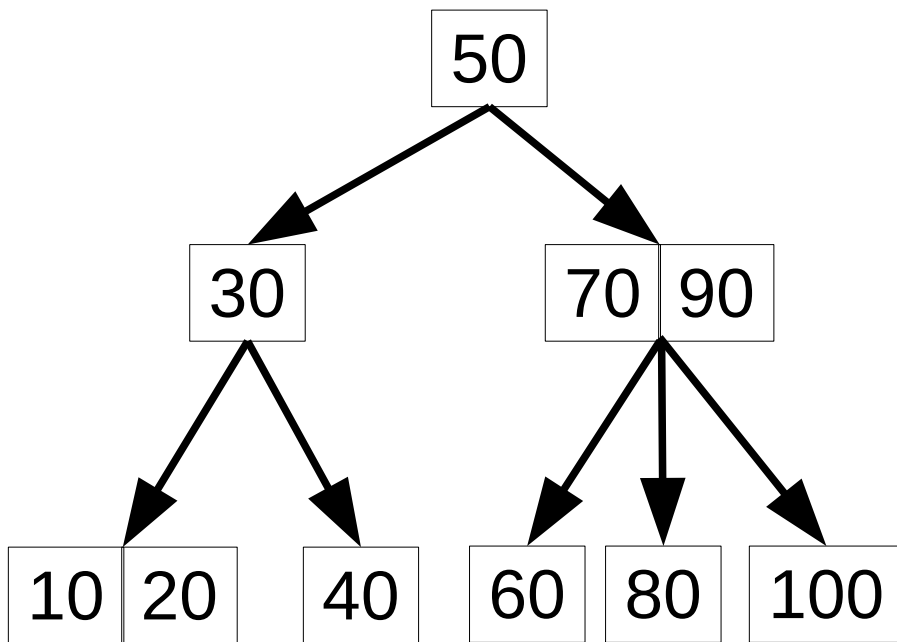
- Remover nó 70.



Trocar a chave (70) pelo seu sucessor (80).

# Remoção

- Remover nó 70.



Trocar a chave (70) pelo seu sucessor (80).

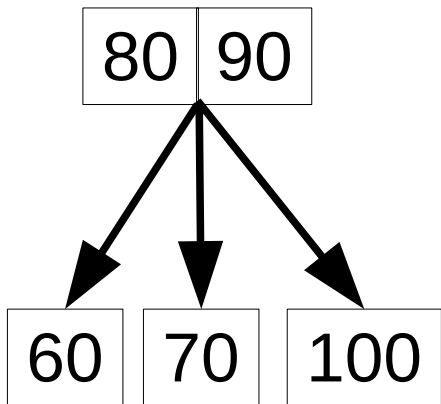
# Remoção

# Remoção

- Remover nó 70 (continuação)...

# Remoção

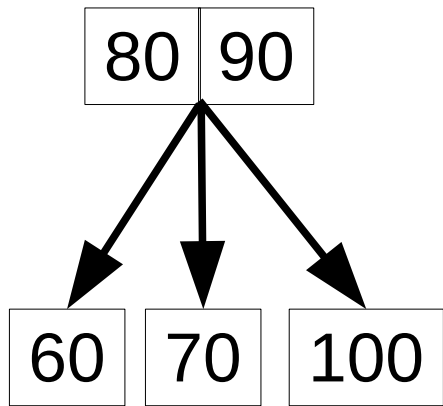
- Remover nó 70 (continuação)...





# Remoção

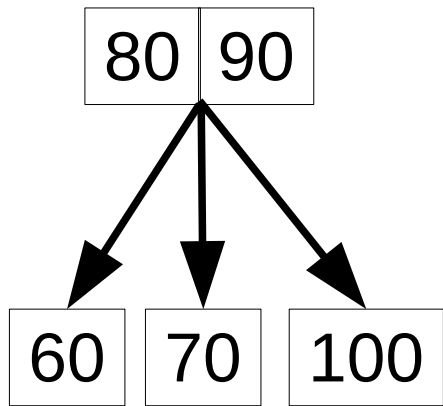
- Remover nó 70 (continuação)...



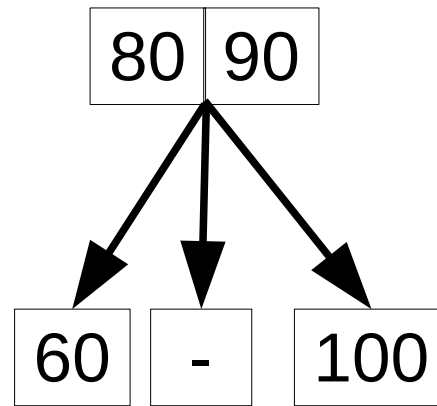
Remover a folha (70).

# Remoção

- Remover nó 70 (continuação)...

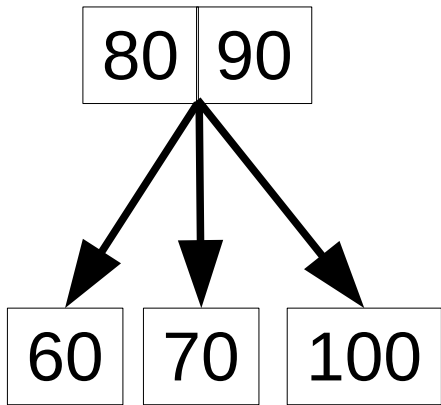


Remover a folha (70).

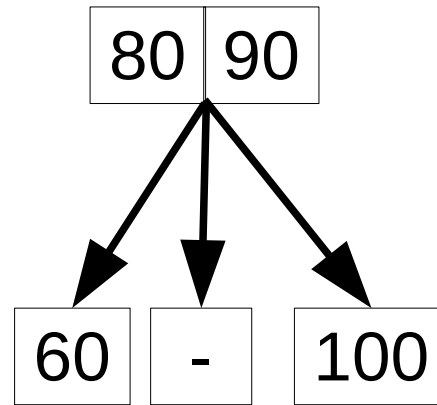


# Remoção

- Remover nó 70 (continuação)...



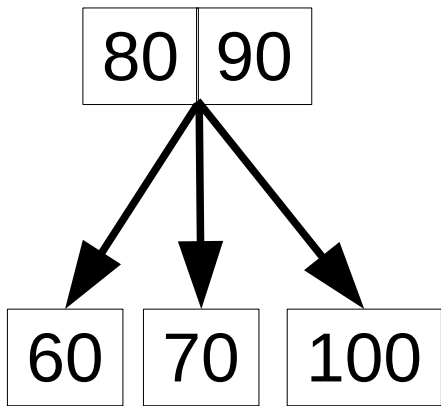
Remover a folha (70).



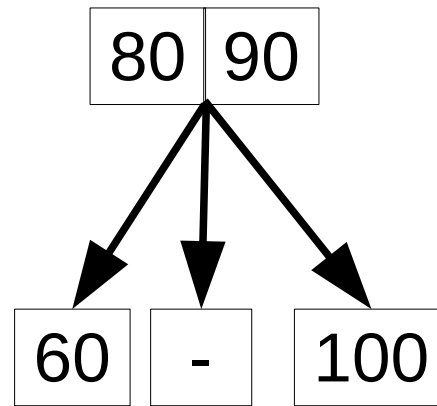
Unir os nós da folha vazia

# Remoção

- Remover nó 70 (continuação)...



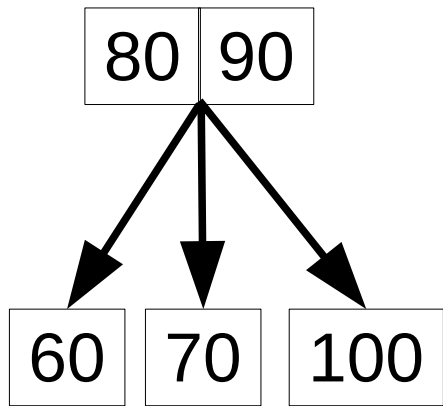
Remover a folha (70).



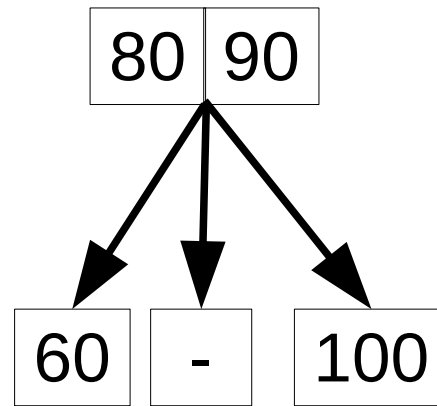
Unir os nós da folha vazia e mover o nó 80 para baixo.

# Remoção

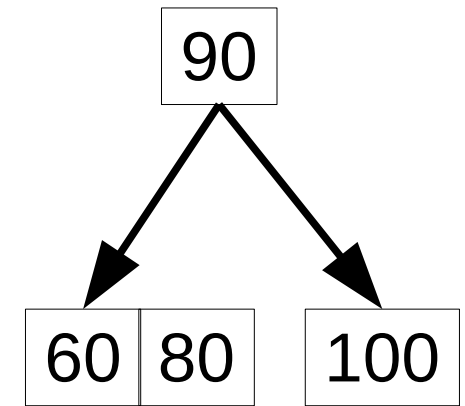
- Remover nó 70 (continuação)...



Remover a folha (70).



Unir os nós da folha vazia e mover o nó 80 para baixo.



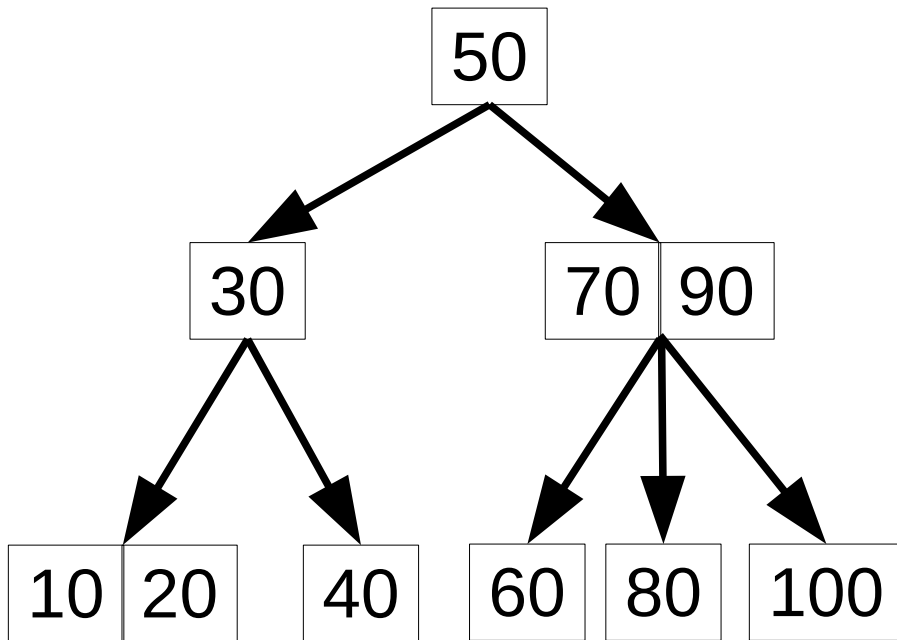
# Remoção

# Remoção

- Remover nó 70 (continuação)...

# Remoção

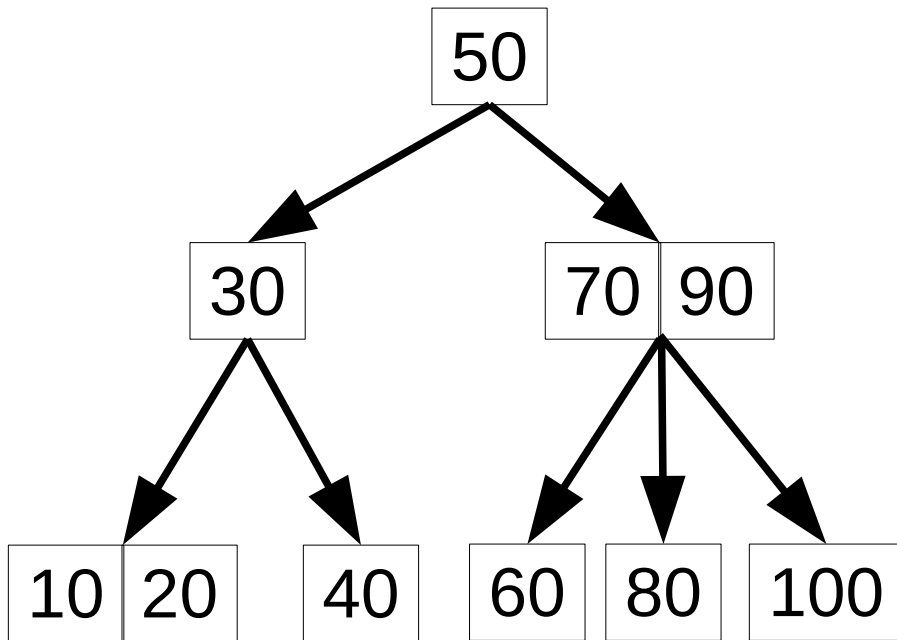
- Remover nó 70 (continuação)...





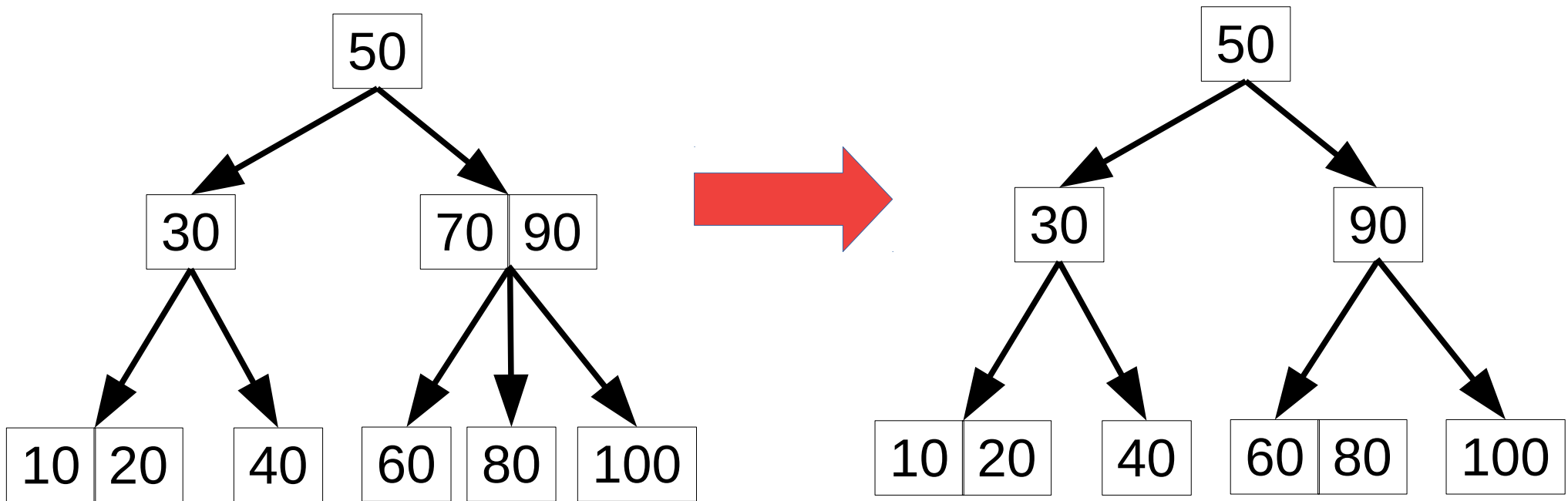
# Remoção

- Remover nó 70 (continuação)...



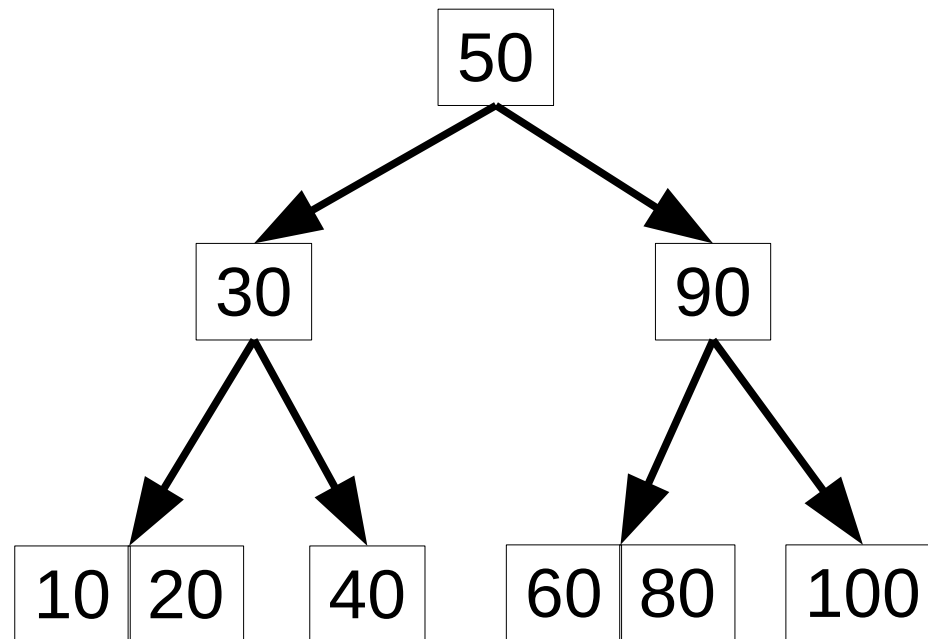
# Remoção

- Remover nó 70 (continuação)...



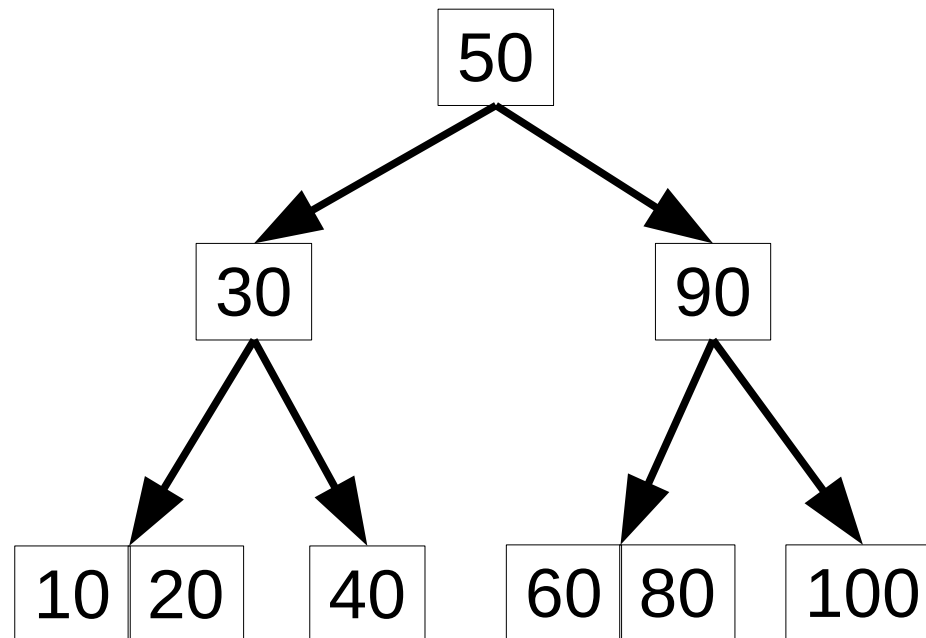
# Remoção

# Remoção



# Remoção

- Remover nó 100



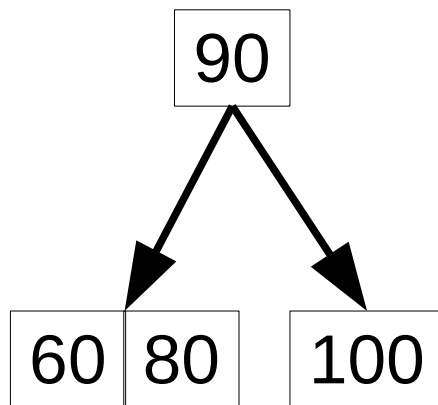
# Remoção

# Remoção

- Remover nó 100 (continuação)...

# Remoção

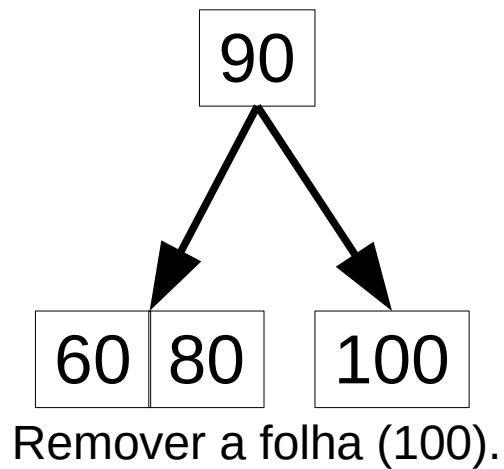
- Remover nó 100 (continuação)...





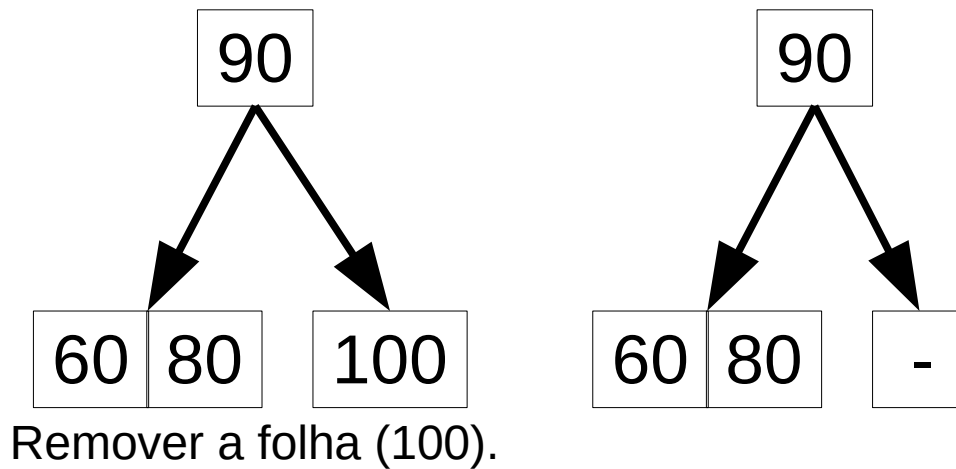
# Remoção

- Remover nó 100 (continuação)...



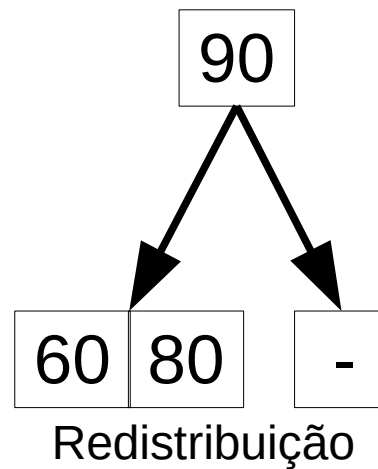
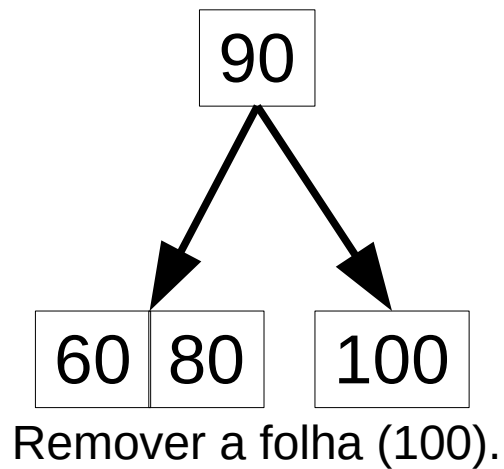
# Remoção

- Remover nó 100 (continuação)...



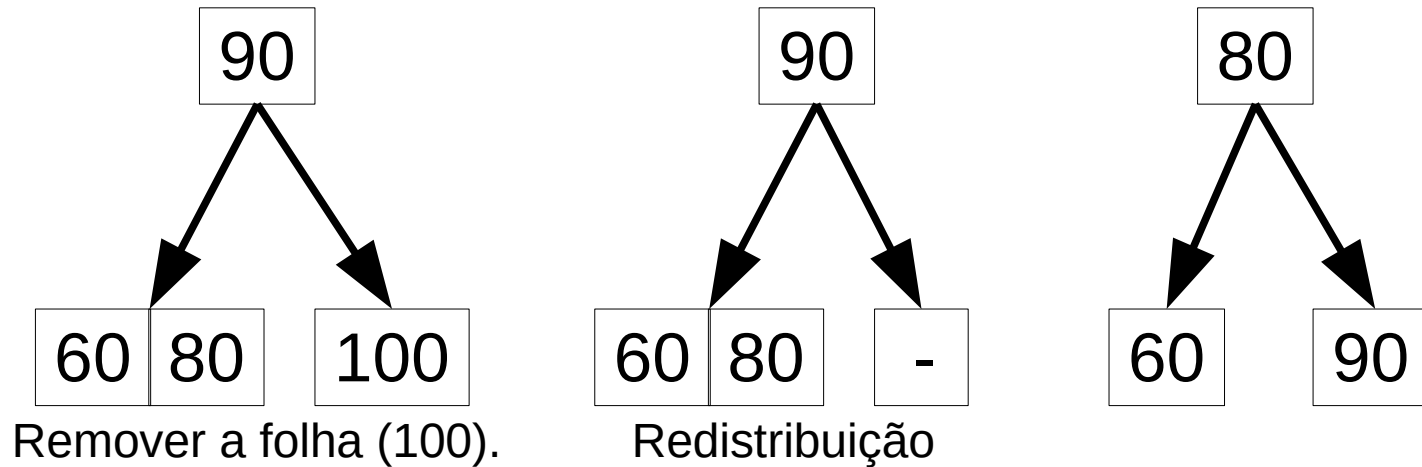
# Remoção

- Remover nó 100 (continuação)...



# Remoção

- Remover nó 100 (continuação)...



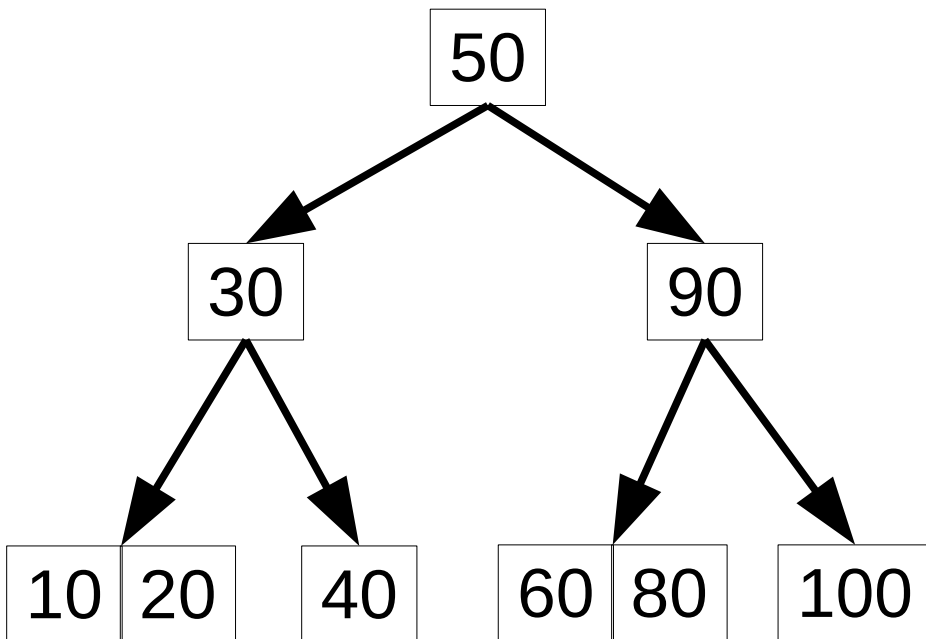
# Remoção

# Remoção

- Remover nó 100 (continuação)...

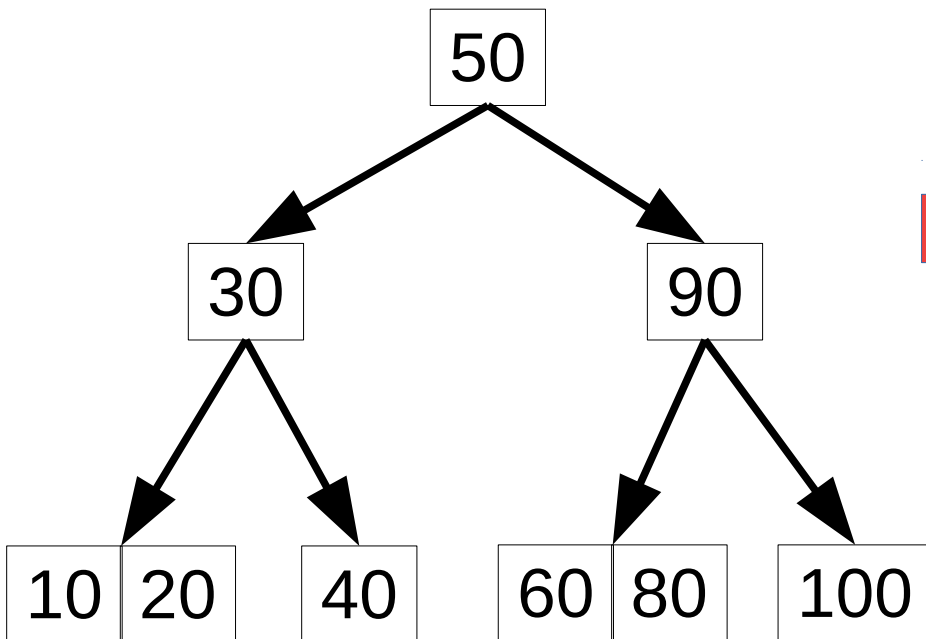
# Remoção

- Remover nó 100 (continuação)...



# Remoção

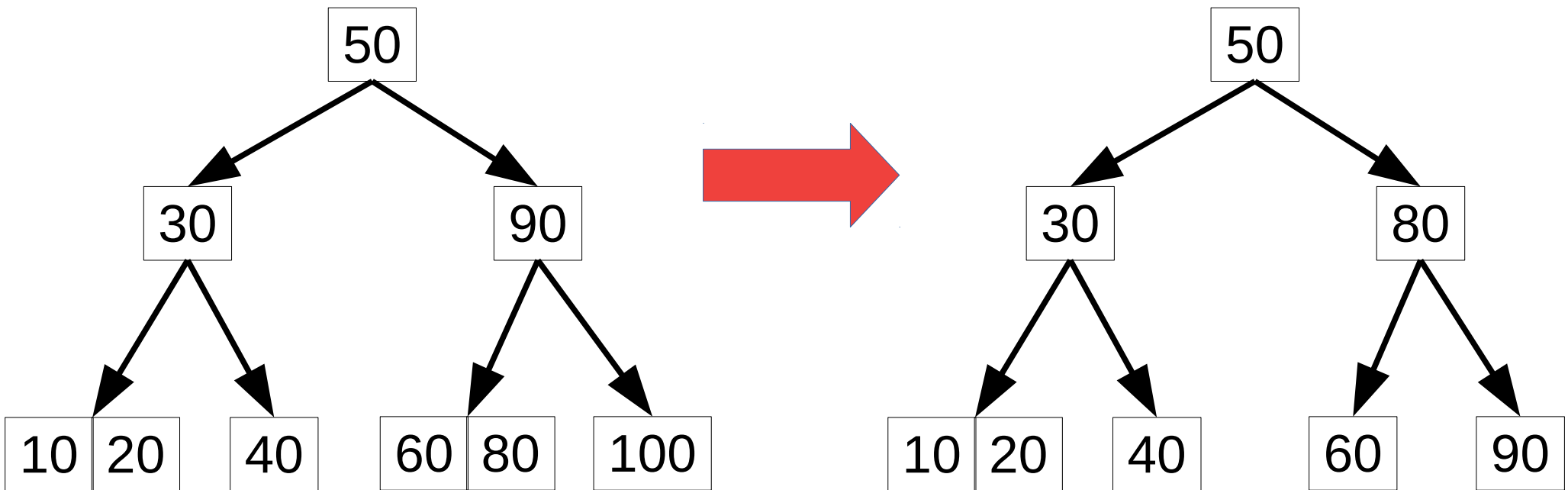
- Remover nó 100 (continuação)...





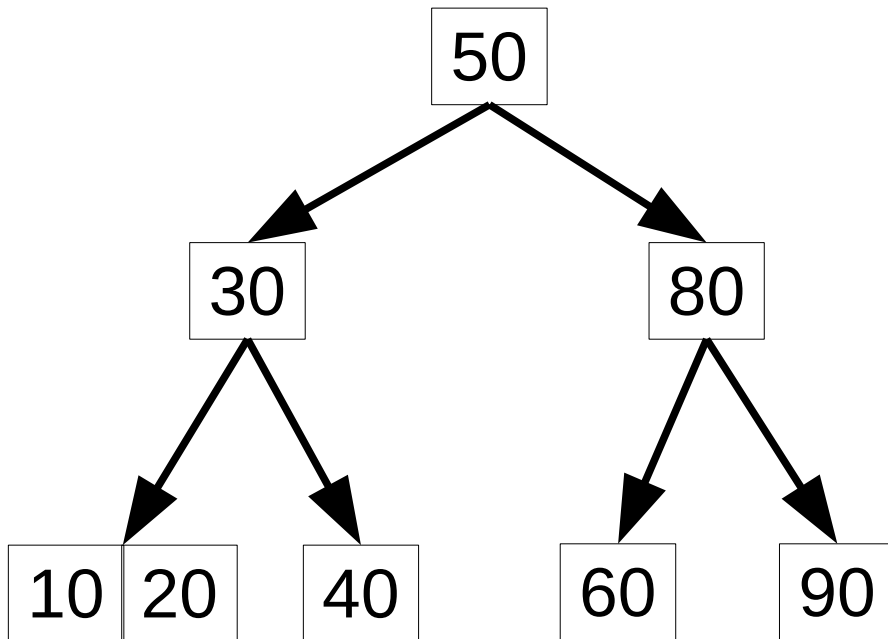
# Remoção

- Remover nó 100 (continuação)...



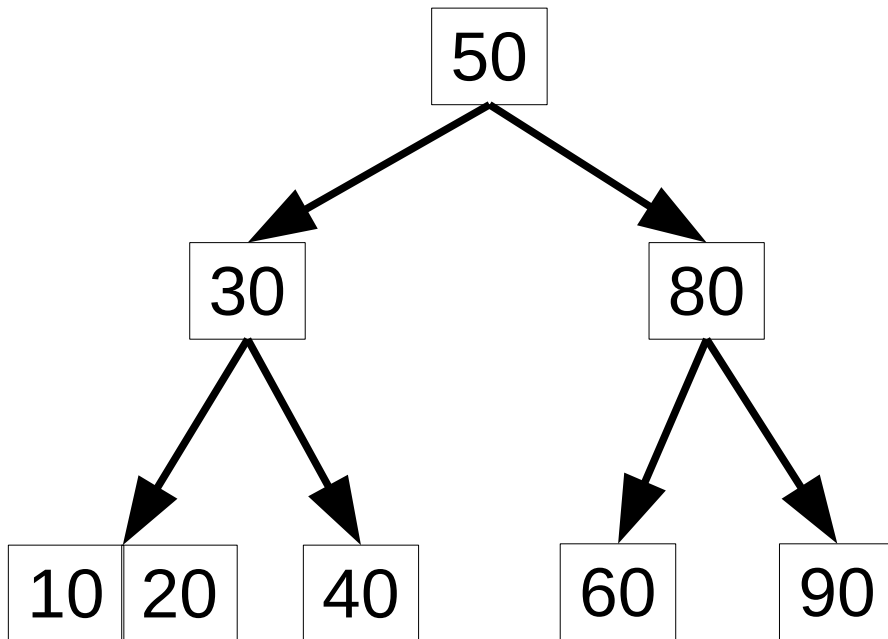
# Remoção

# Remoção



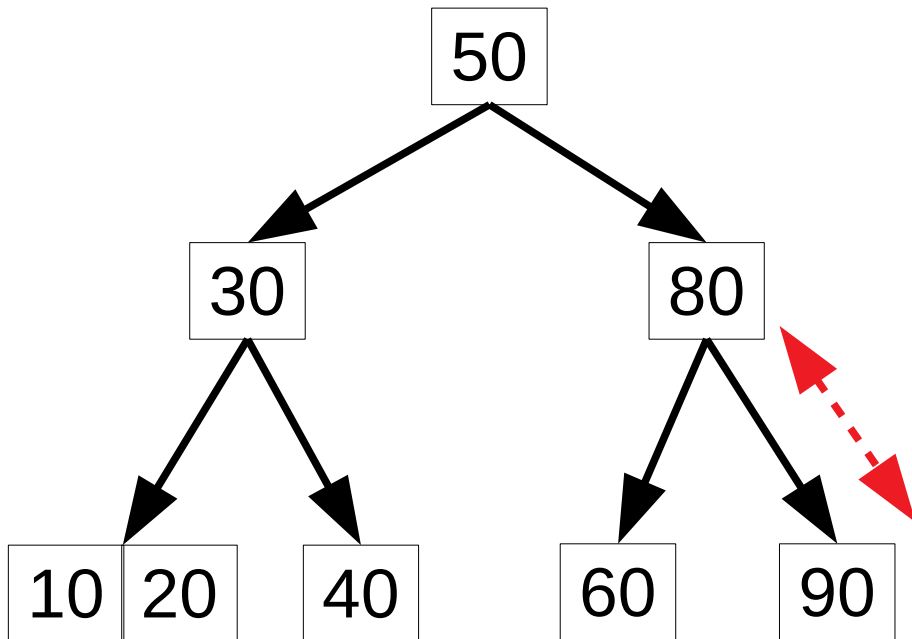
# Remoção

- Remover nó 80.



# Remoção

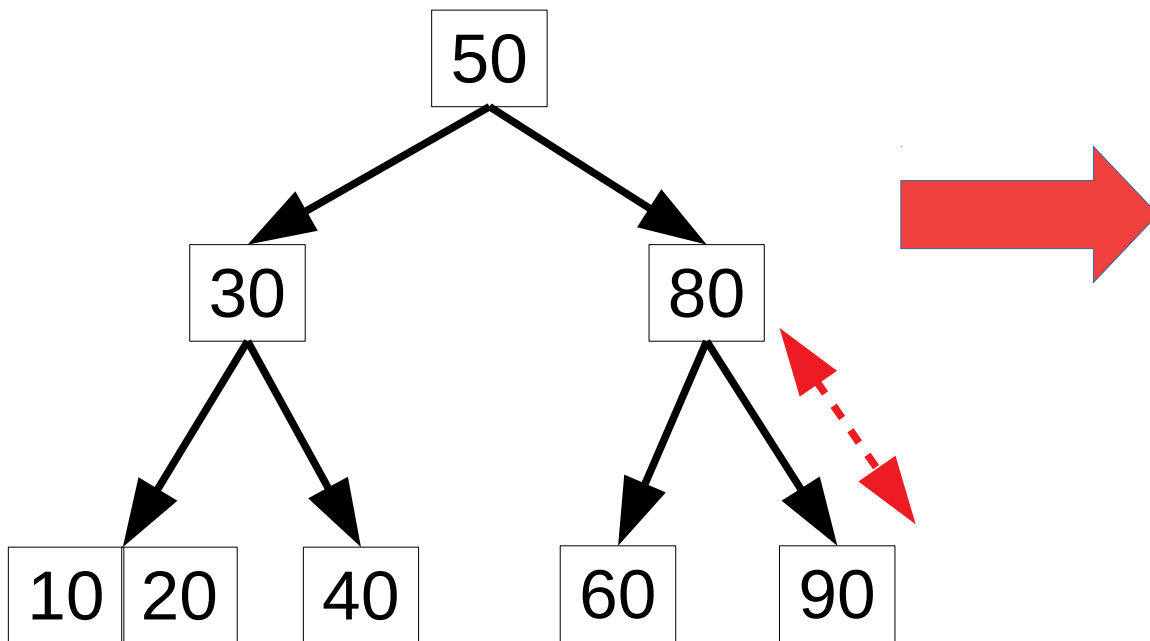
- Remover nó 80.



Trocar a ordem com o seu sucessor!

# Remoção

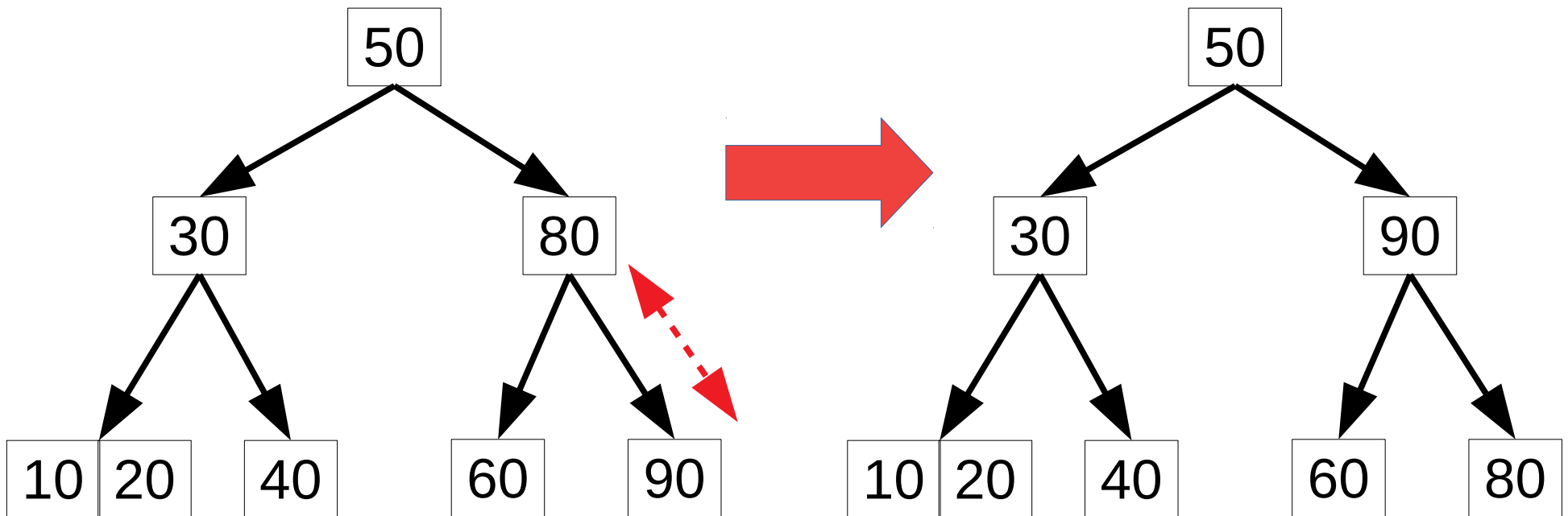
- Remover nó 80.



Trocar a ordem com o seu sucessor!

# Remoção

- Remover nó 80.



Trocar a ordem com o seu sucessor!

# Remoção

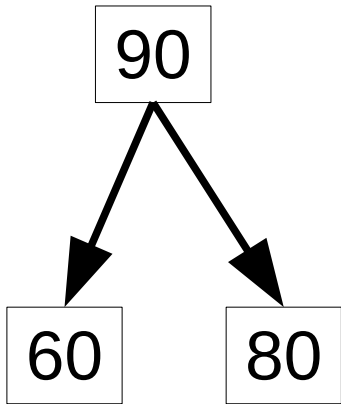


# Remoção

- Remover nó 80 (continuação)...

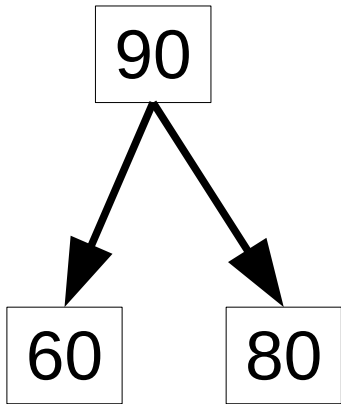
# Remoção

- Remover nó 80 (continuação)...



# Remoção

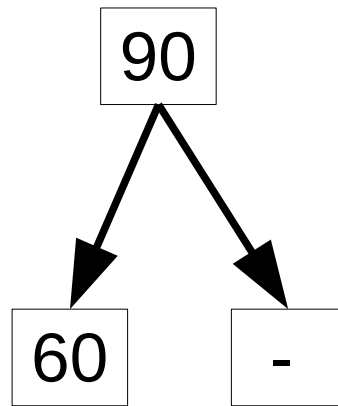
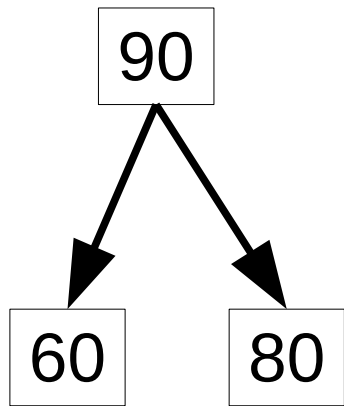
- Remover nó 80 (continuação)...



Remover a folha (80).

# Remoção

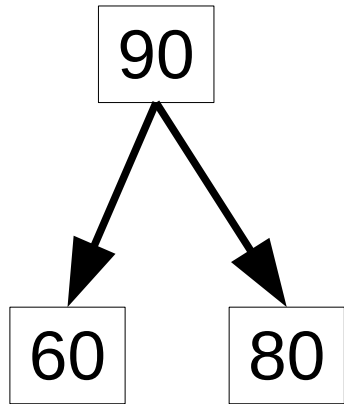
- Remover nó 80 (continuação)...



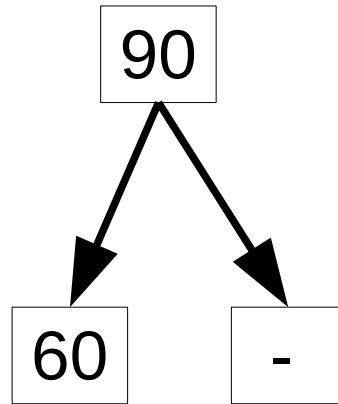
Remover a folha (80).

# Remoção

- Remover nó 80 (continuação)...



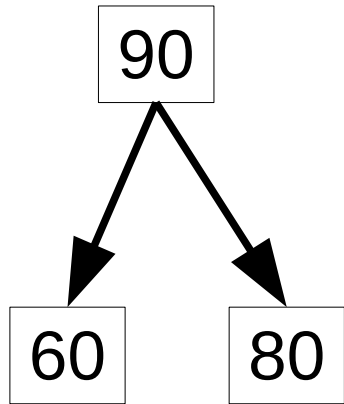
Remover a folha (80).



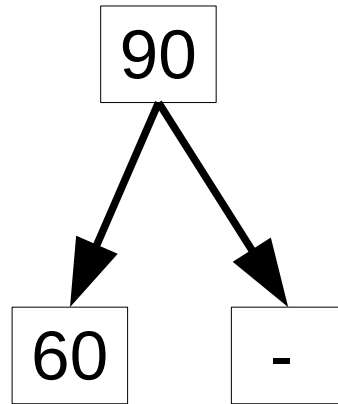
Unir os nós da folha vazia

# Remoção

- Remover nó 80 (continuação)...



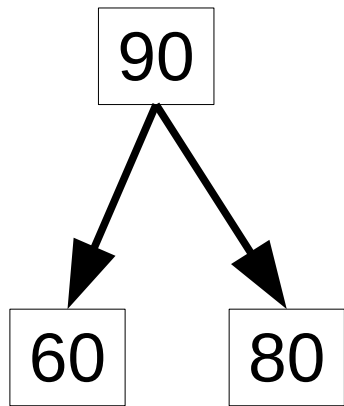
Remover a folha (80).



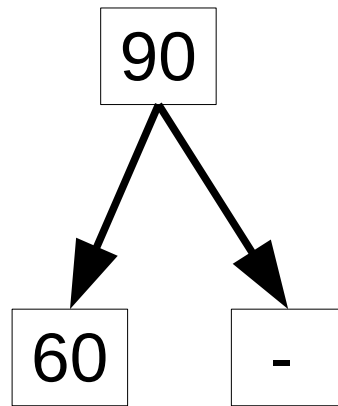
Unir os nós da folha vazia e mover o nó 90 para baixo.

# Remoção

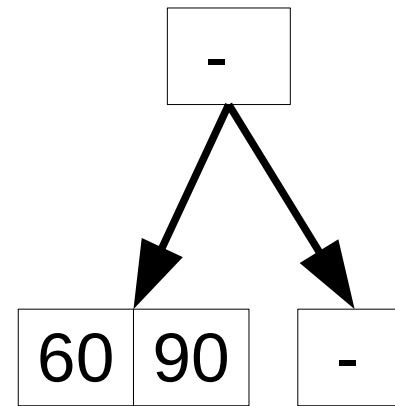
- Remover nó 80 (continuação)...



Remover a folha (80).



Unir os nós da folha vazia e mover o nó 90 para baixo.



# Remoção

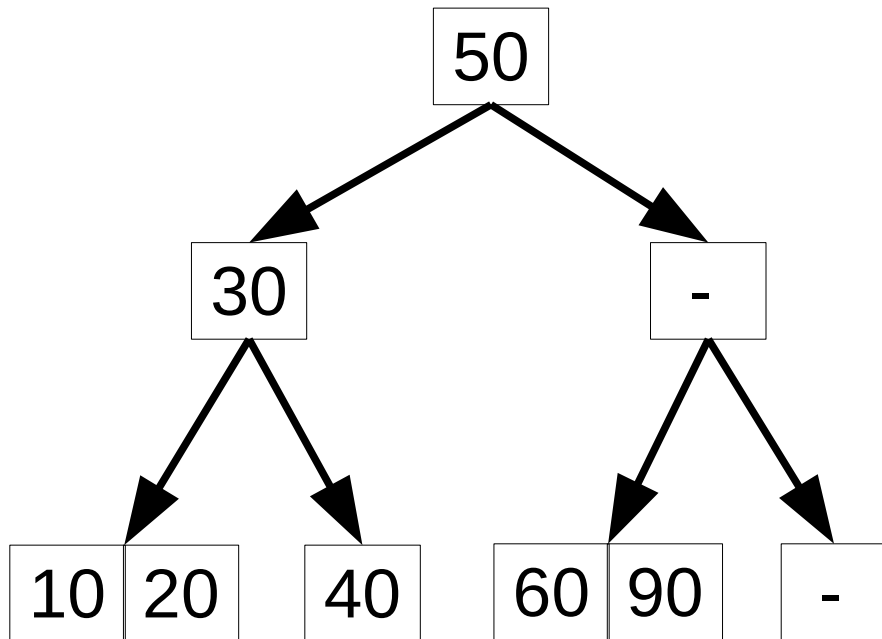


# Remoção

- Remover nó 80 (continuação)...

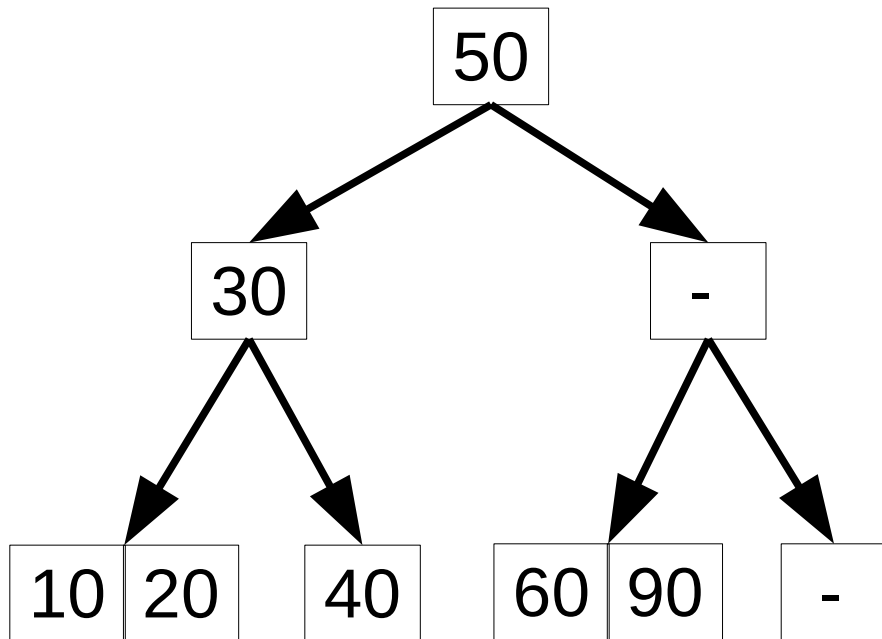
# Remoção

- Remover nó 80 (continuação)...



# Remoção

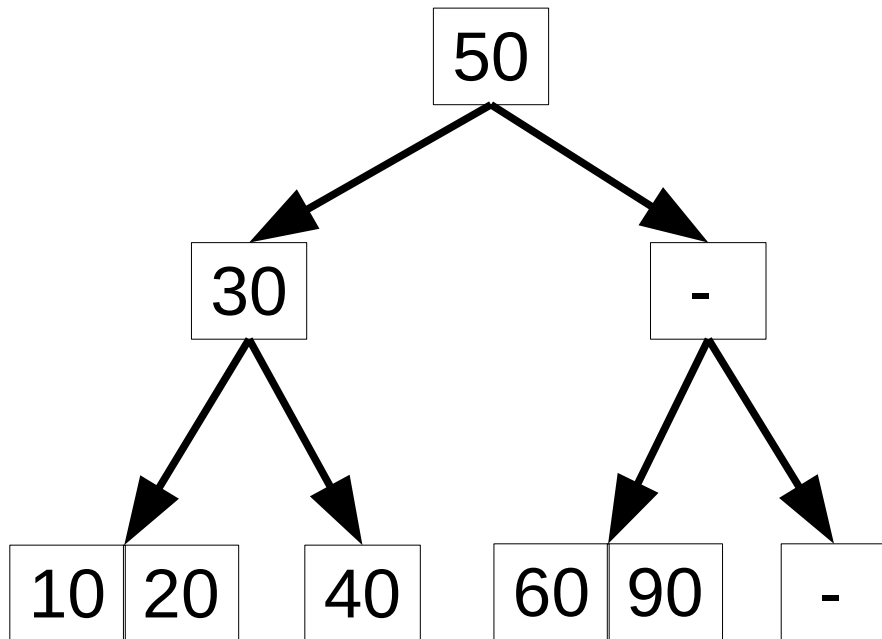
- Remover nó 80 (continuação)...



Unir os nós da folha vazia

# Remoção

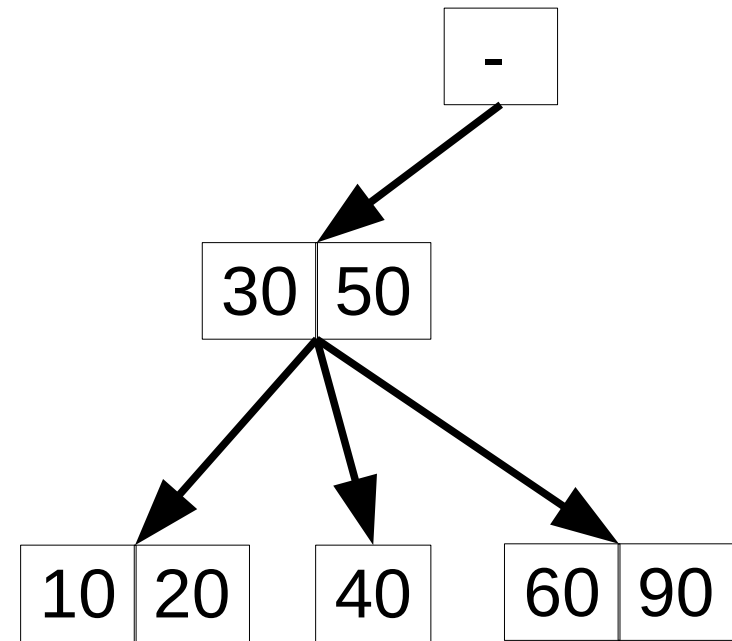
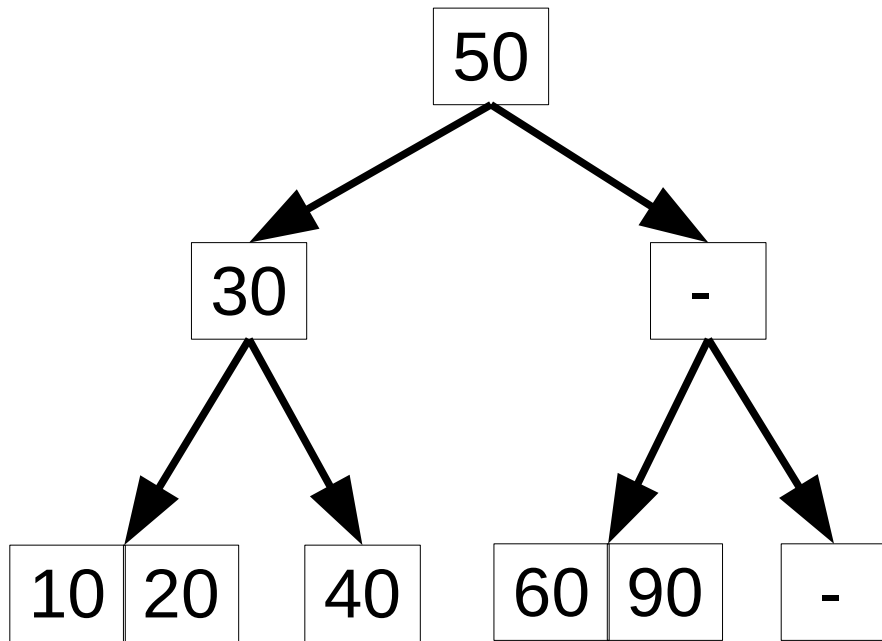
- Remover nó 80 (continuação)...



Unir os nós da folha vazia  
e mover o nó 50 para baixo.

# Remoção

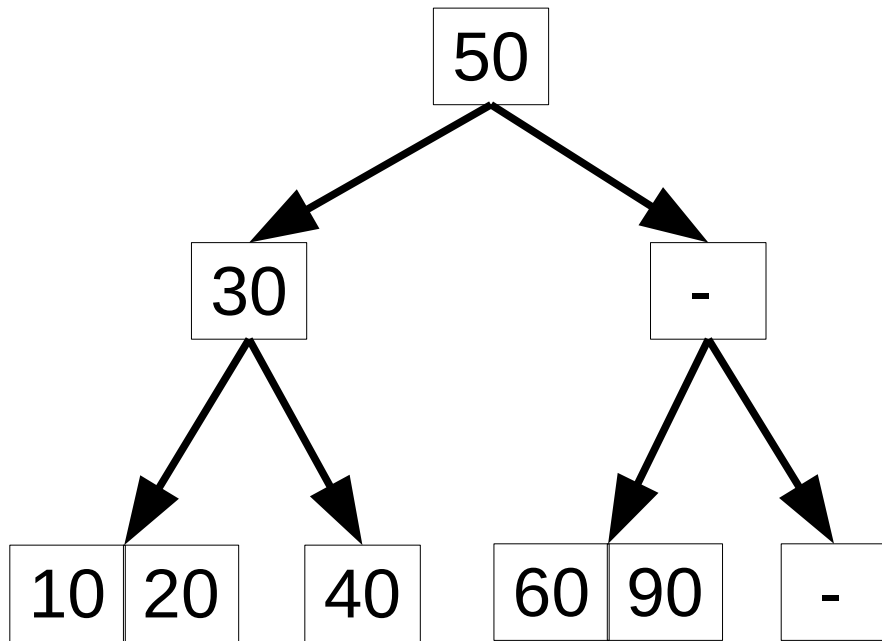
- Remover nó 80 (continuação)...



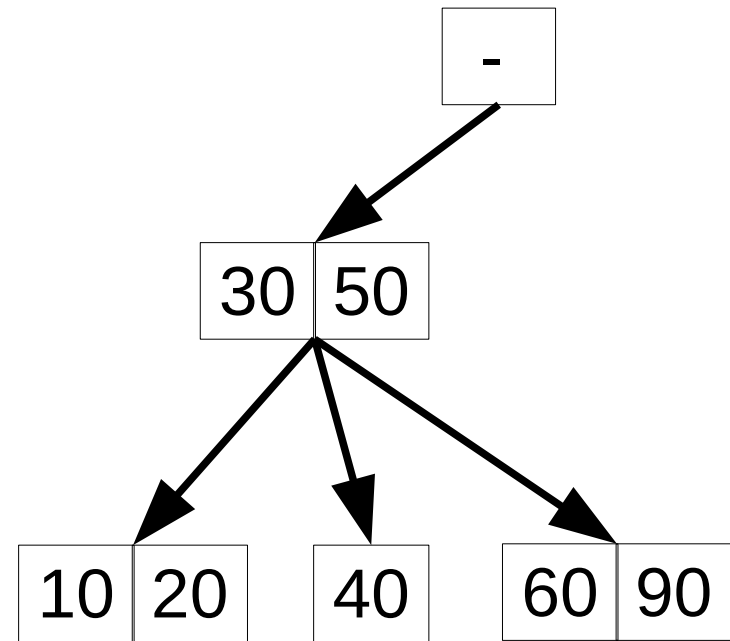
Unir os nós da folha vazia  
e mover o nó 50 para baixo.

# Remoção

- Remover nó 80 (continuação)...



Unir os nós da folha vazia e mover o nó 50 para baixo.



Remove o nó raiz.

# Remoção

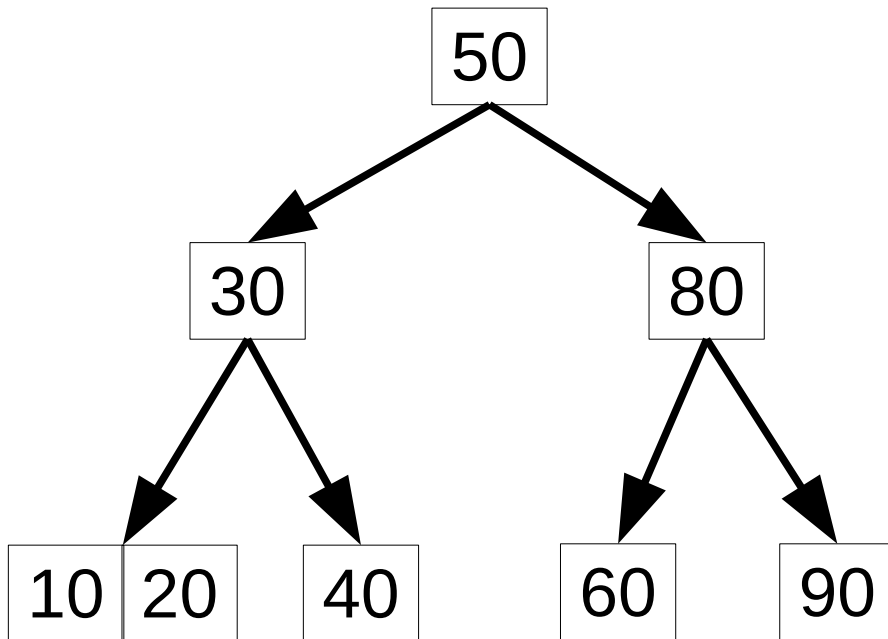
# Remoção

- Remover nó 80 (continuação)...



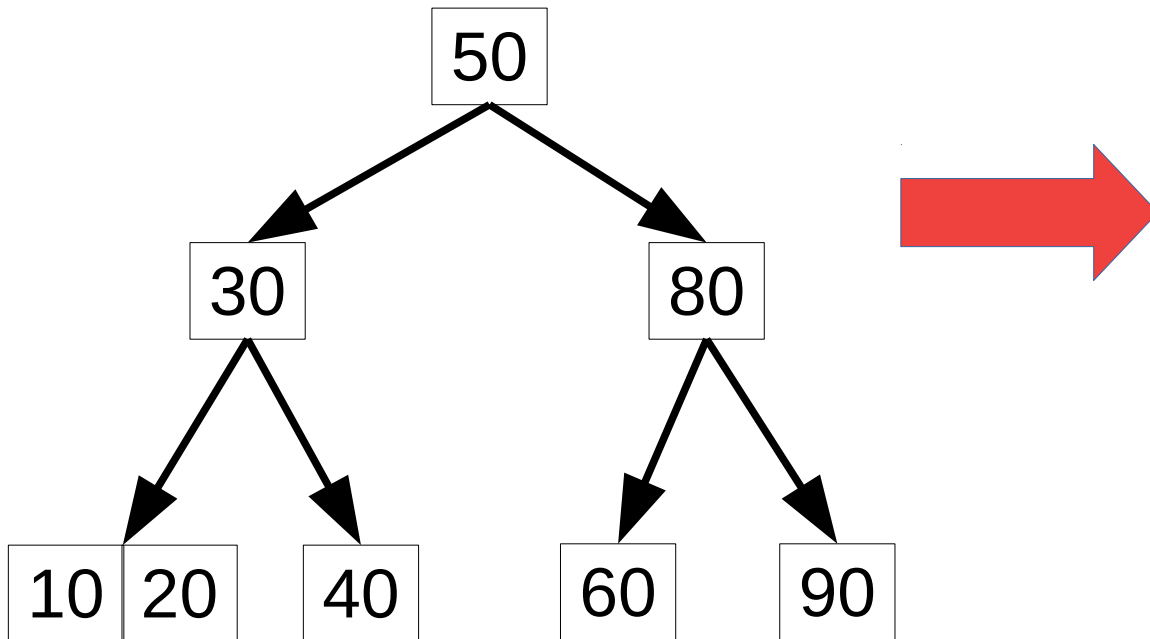
# Remoção

- Remover nó 80 (continuação)...



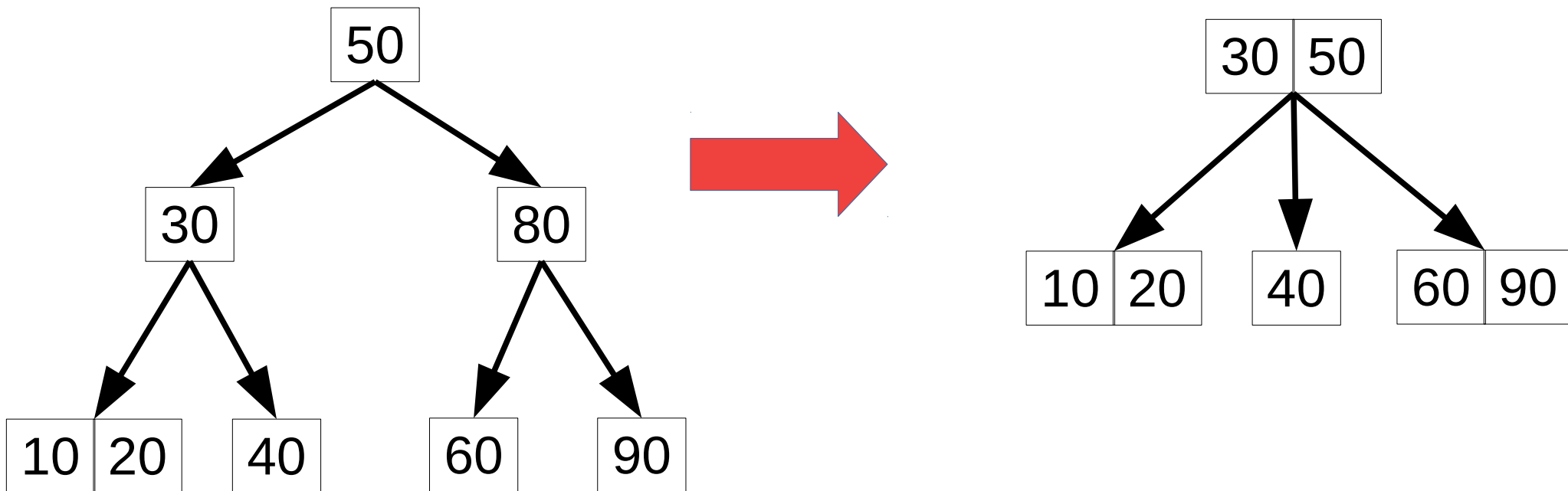
# Remoção

- Remover nó 80 (continuação)...



# Remoção

- Remover nó 80 (continuação)...



# Remoção

# Remoção

- Algoritmo Remover X

# Remoção

- Algoritmo Remover X
  1. Localizar o nó N que contém o nó X.

# Remoção

- Algoritmo Remover X
  1. Localizar o nó N que contém o nó X.
  2. Se N não é uma folha

# Remoção

- Algoritmo Remover X
  1. Localizar o nó N que contém o nó X.
  2. Se N não é uma folha
    1. Trocar X por seu sucessor.



# Remoção

- Algoritmo Remover X
  1. Localizar o nó N que contém o nó X.
  2. Se N não é uma folha
    1. Trocar X por seu sucessor.
    2. Remoção sempre será nas folhas.

# Remoção

- Algoritmo Remover X
  1. Localizar o nó N que contém o nó X.
  2. Se N não é uma folha
    1. Trocar X por seu sucessor.
    2. Remoção sempre será nas folhas.
  3. Se o nó folha N contém outro item, apague X, senão, tente redistribuir os nós irmãos, se não for possível, junte os nós.

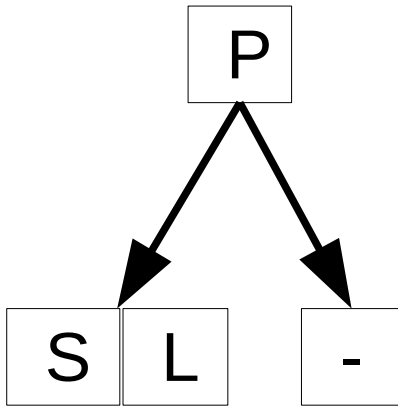
# Remoção

# Remoção

- Redistribuição

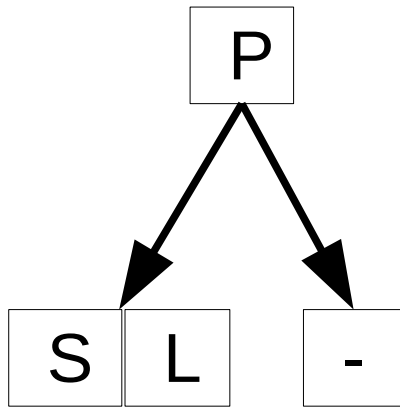
# Remoção

- Redistribuição



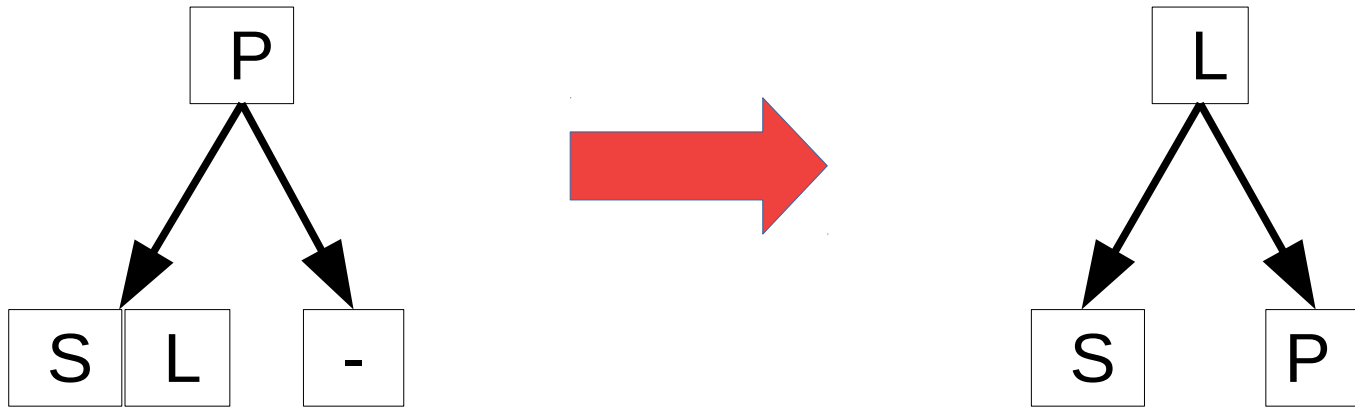
# Remoção

- Redistribuição



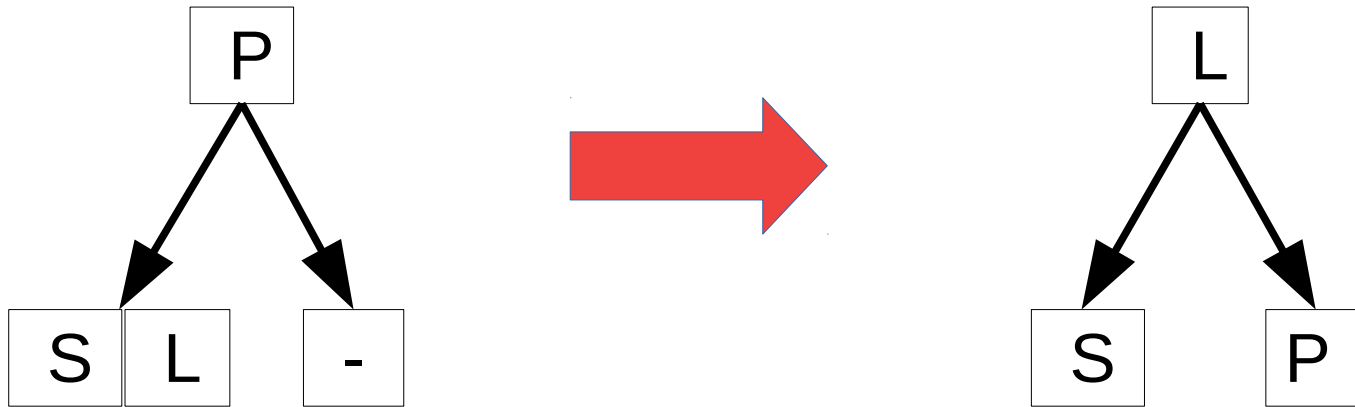
# Remoção

- Redistribuição



# Remoção

- Redistribuição

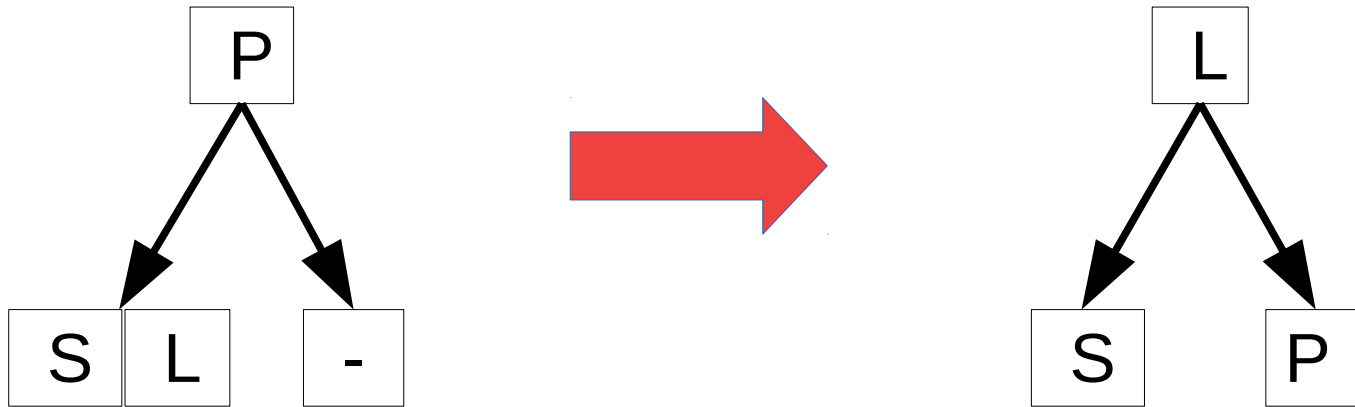


- Juntando (*merge*)

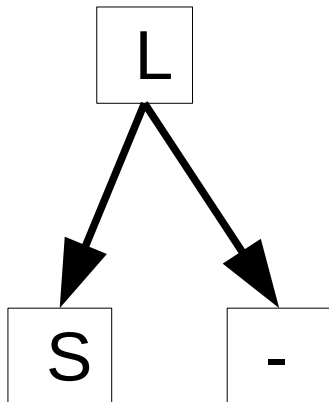


# Remoção

- Redistribuição

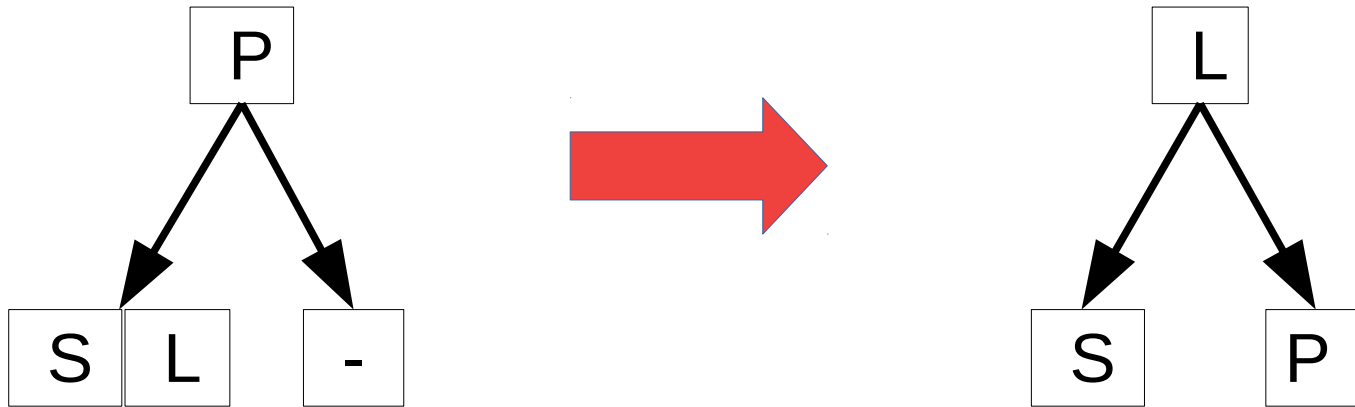


- Juntando (*merge*)

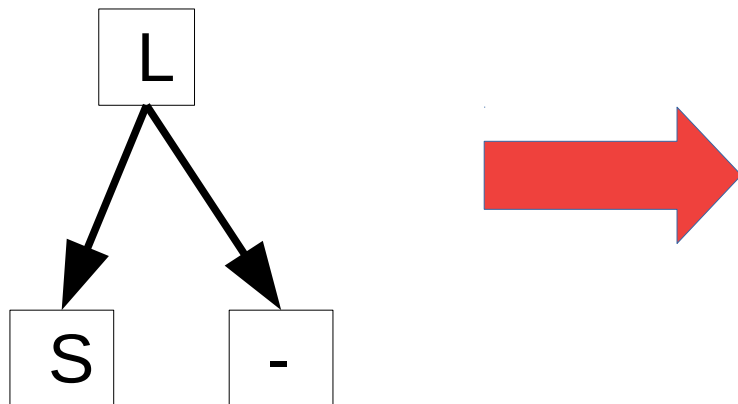


# Remoção

- Redistribuição

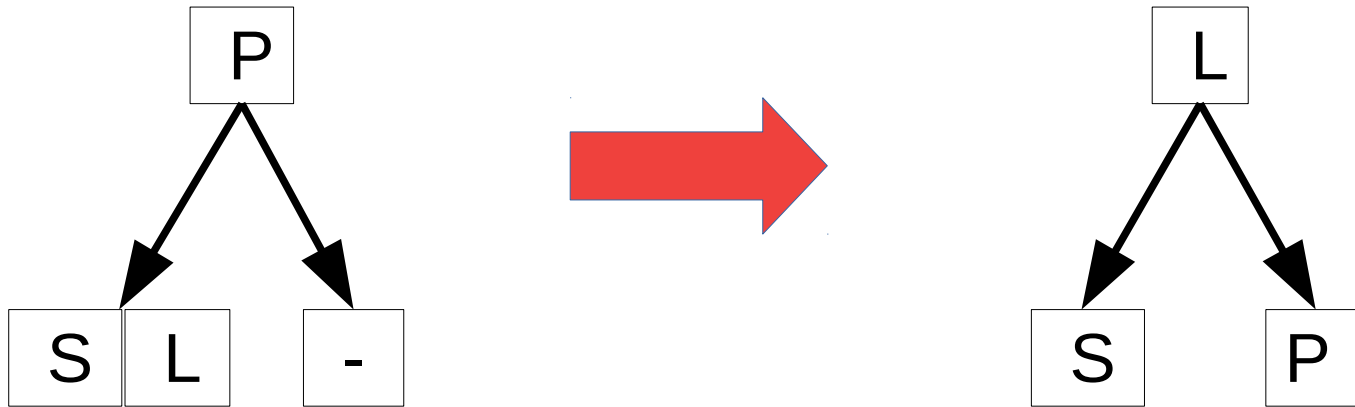


- Juntando (*merge*)

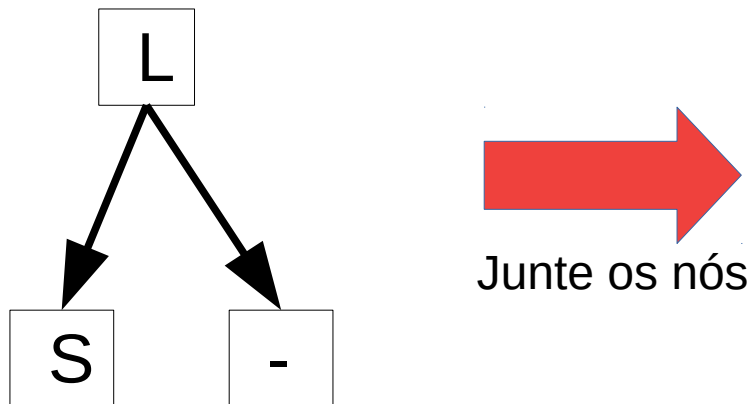


# Remoção

- Redistribuição

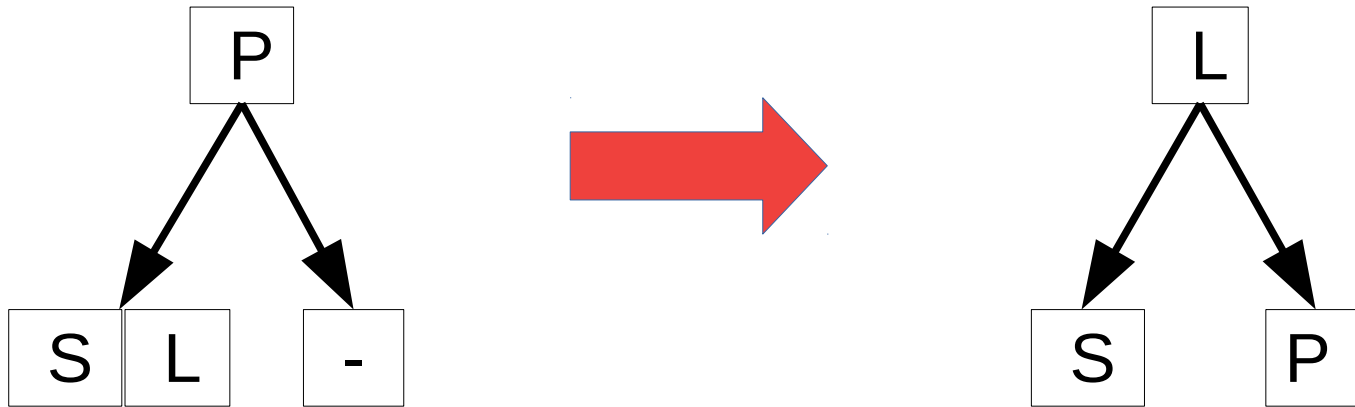


- Juntando (*merge*)

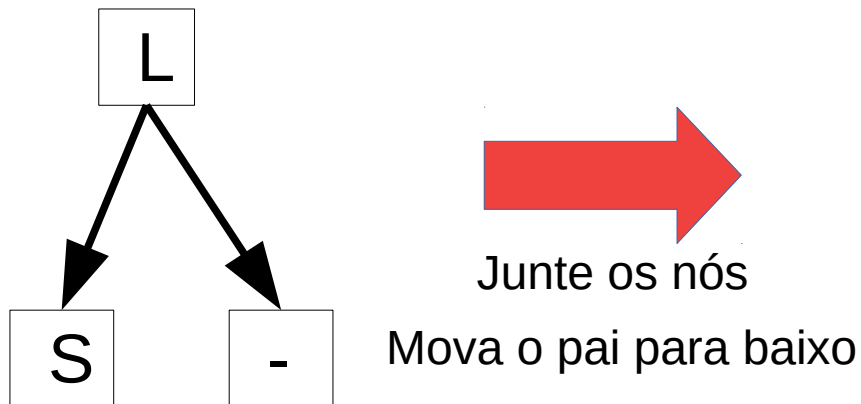


# Remoção

- Redistribuição

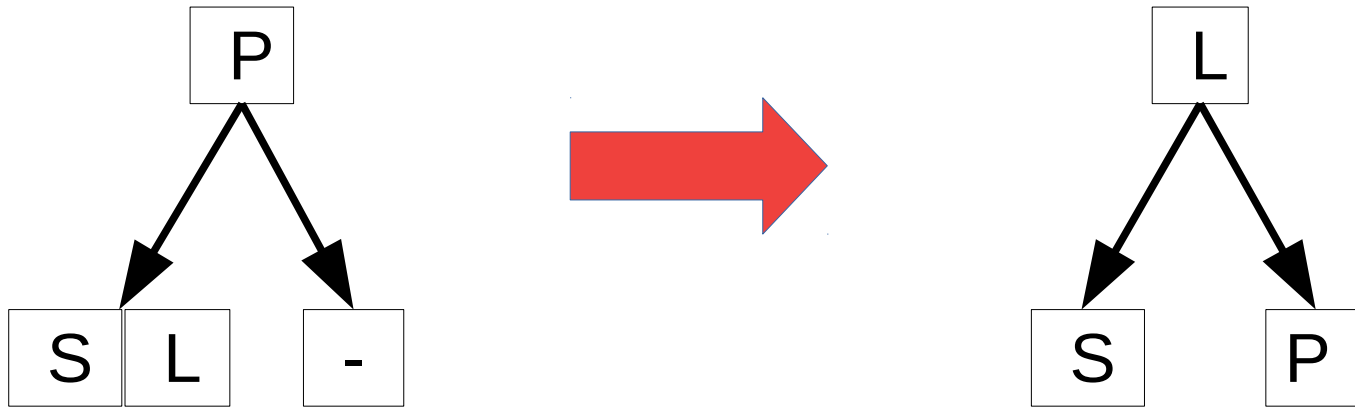


- Juntando (*merge*)

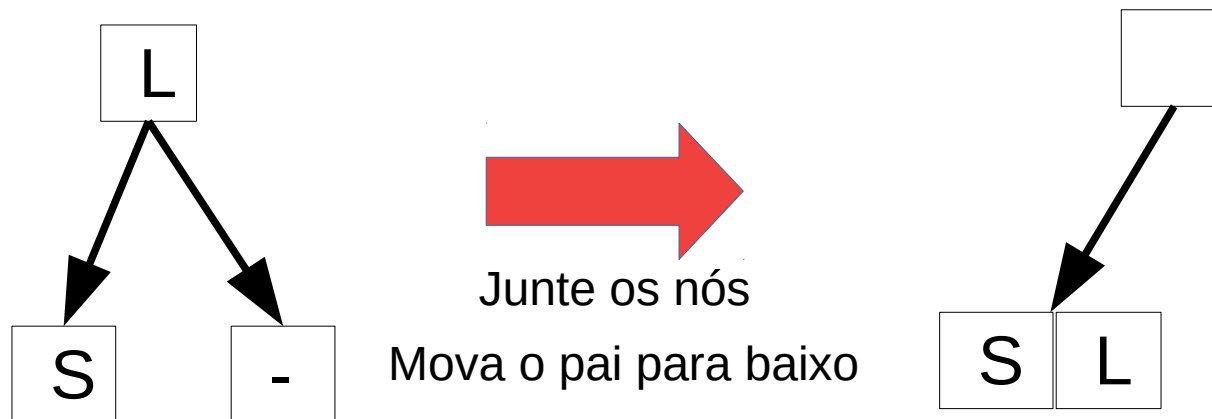


# Remoção

- Redistribuição



- Juntando (*merge*)



# Remoção

# Remoção

- Redistribuição

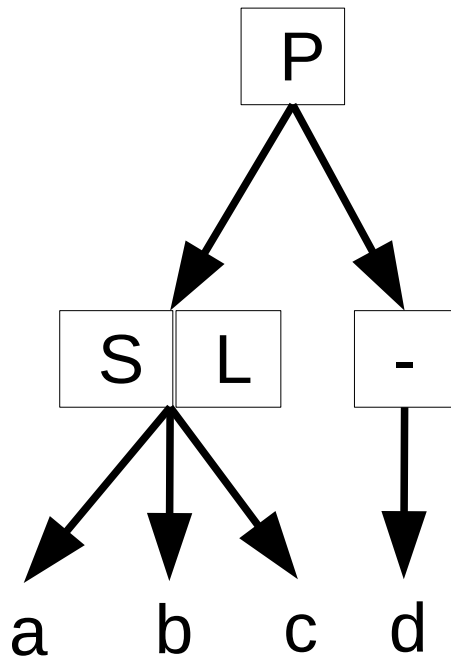
# Remoção

- Redistribuição
  - Nó interno não tem item a esquerda.



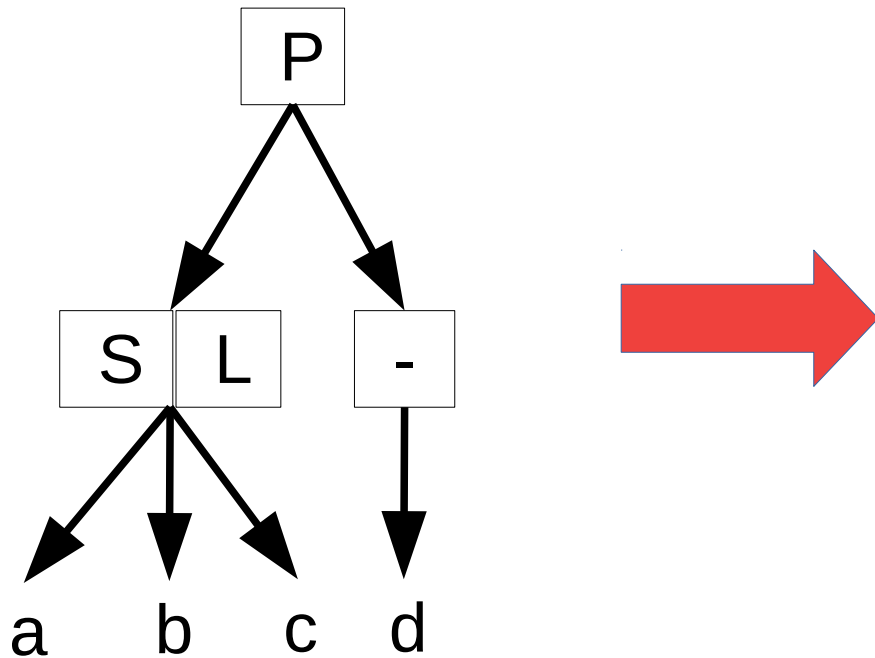
# Remoção

- Redistribuição
  - Nó interno não tem item a esquerda.



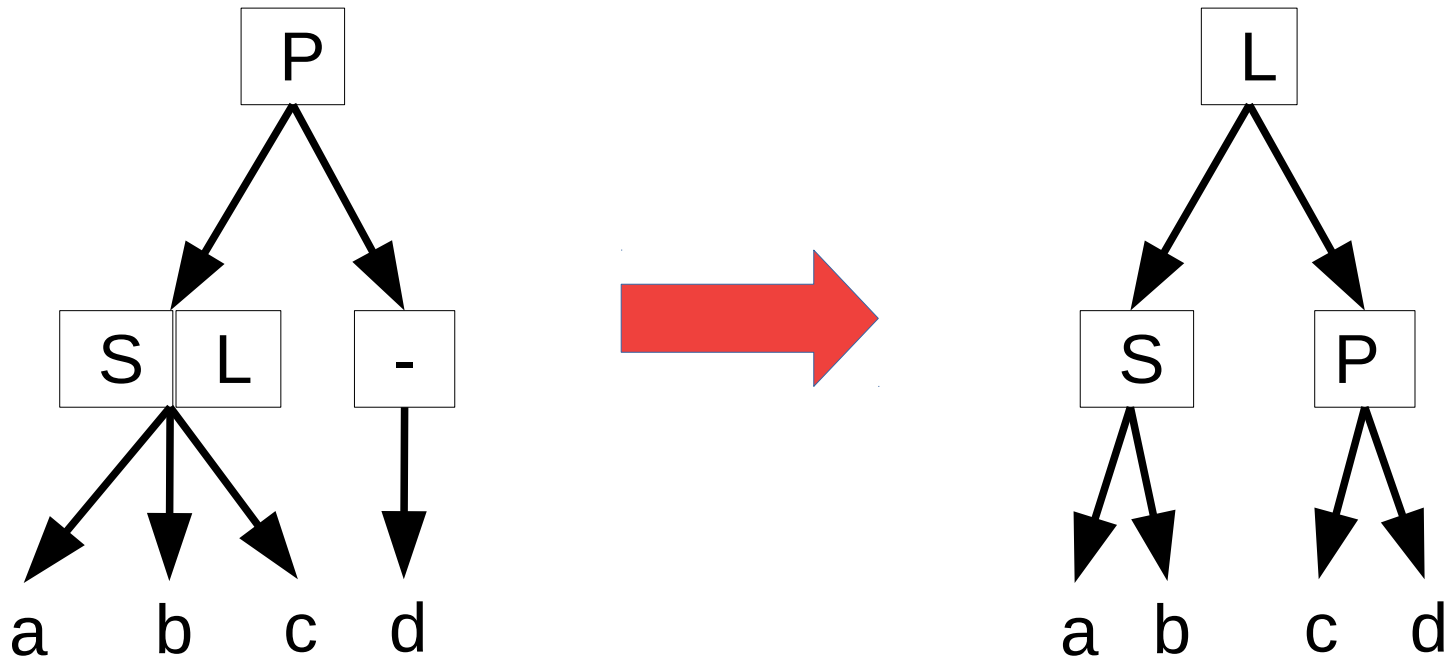
# Remoção

- Redistribuição
  - Nó interno não tem item a esquerda.



# Remoção

- Redistribuição
  - Nó interno não tem item a esquerda.



# Remoção

# Remoção

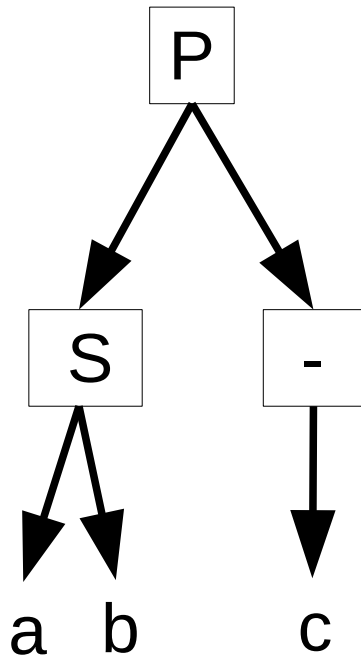
- Juntando (*merge*)

# Remoção

- Juntando (*merge*)
  - Redistribuição não é possível.

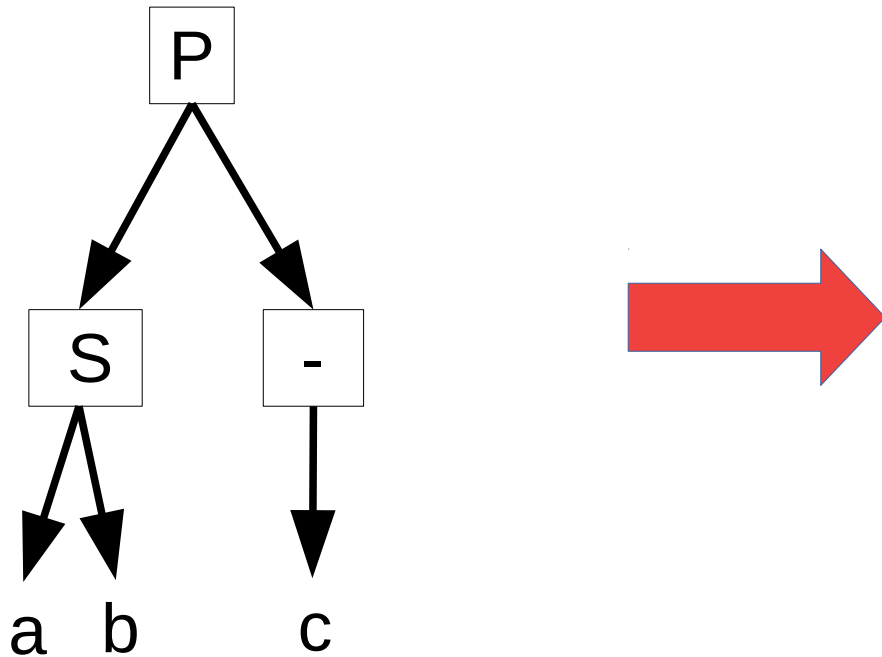
# Remoção

- Juntando (*merge*)
  - Redistribuição não é possível.



# Remoção

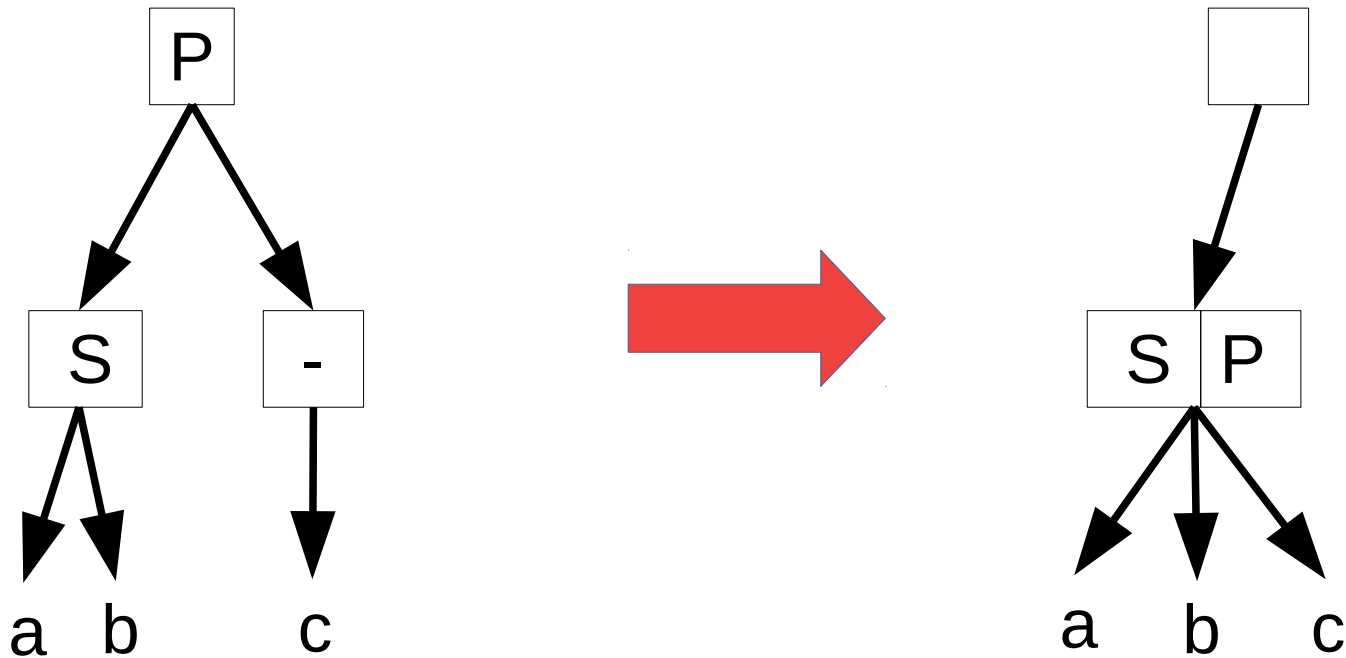
- Juntando (*merge*)
  - Redistribuição não é possível.





# Remoção

- Juntando (*merge*)
  - Redistribuição não é possível.



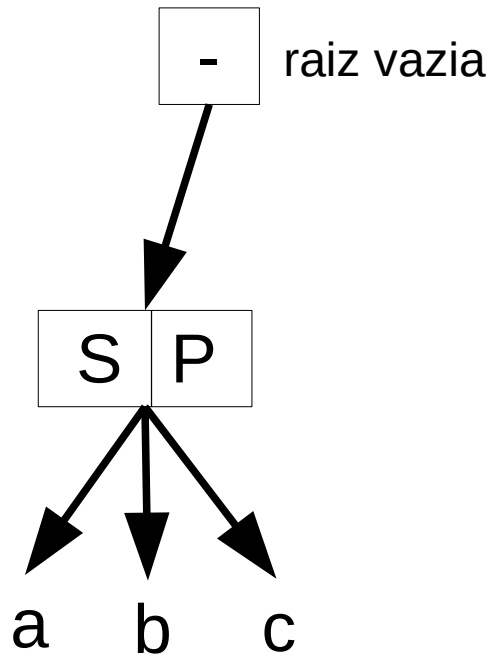
# Remoção

# Remoção

- Se o processo de *merging* chegar até a raiz e a raiz estiver vazia, então remova a raiz.

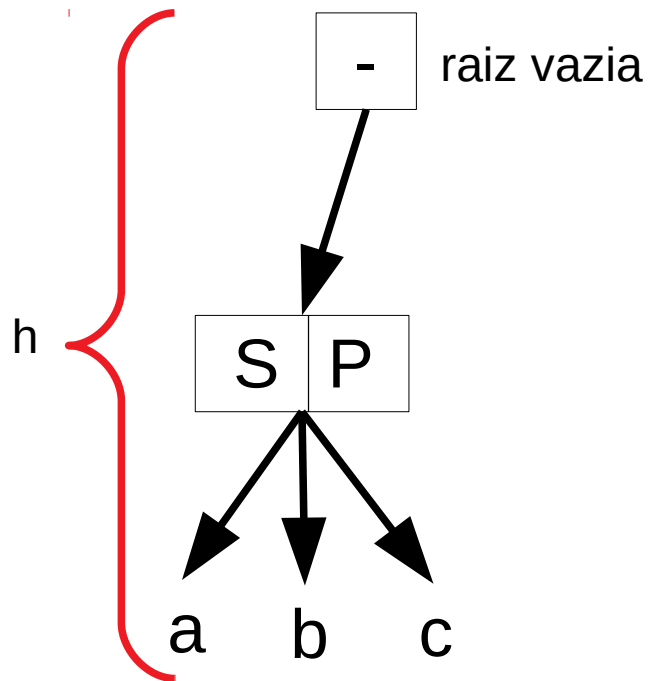
# Remoção

- Se o processo de *merging* chegar até a raiz e a raiz estiver vazia, então remova a raiz.



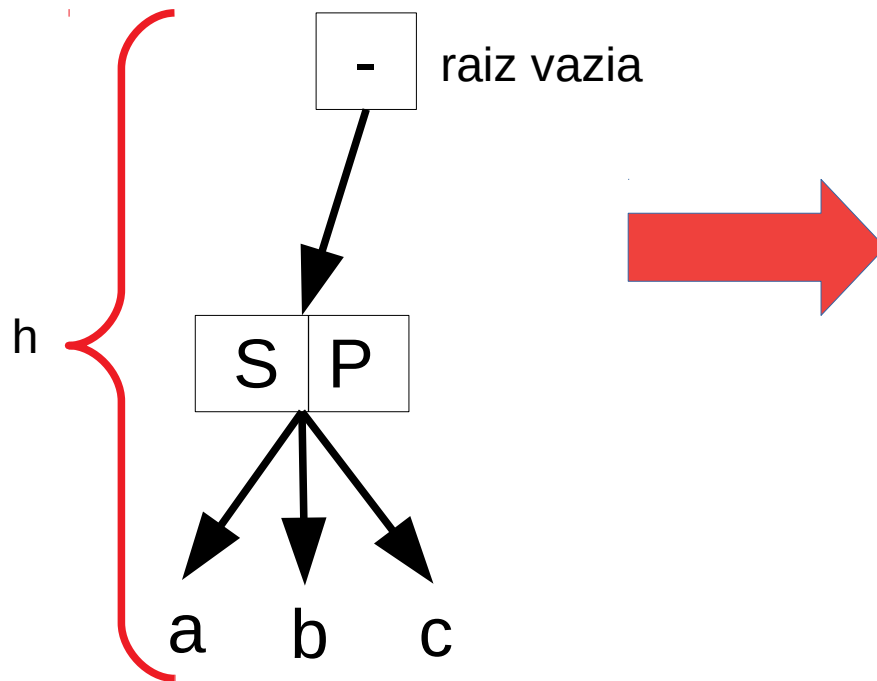
# Remoção

- Se o processo de *merging* chegar até a raiz e a raiz estiver vazia, então remova a raiz.



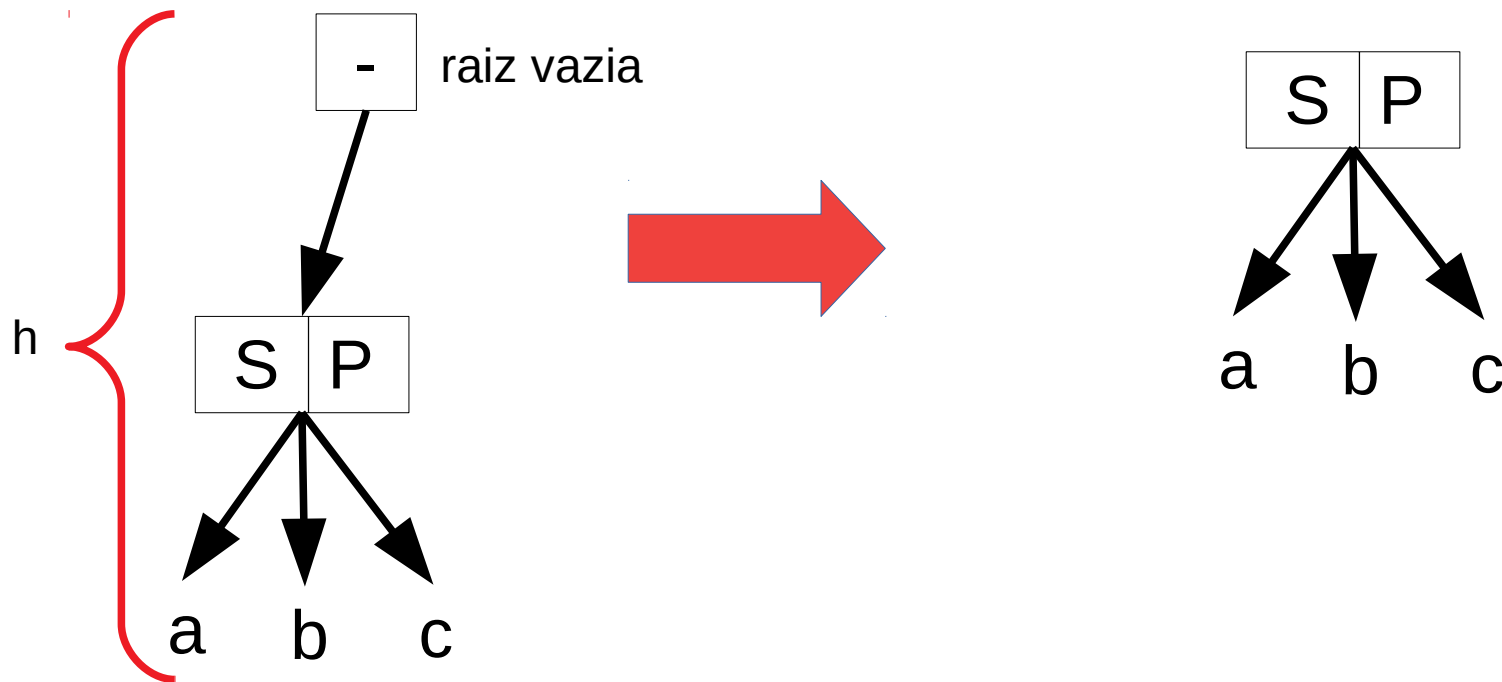
# Remoção

- Se o processo de *merging* chegar até a raiz e a raiz estiver vazia, então remova a raiz.



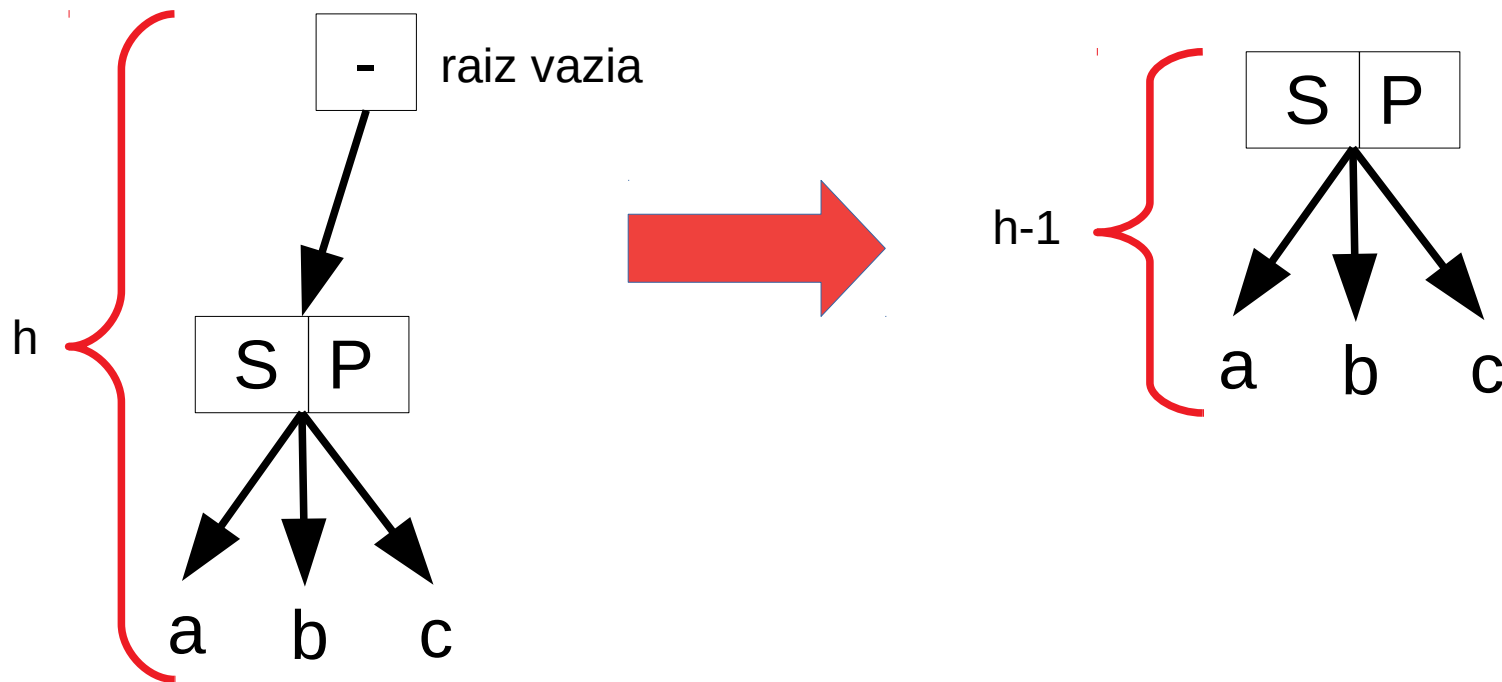
# Remoção

- Se o processo de *merging* chegar até a raiz e a raiz estiver vazia, então remova a raiz.



# Remoção

- Se o processo de *merging* chegar até a raiz e a raiz estiver vazia, então remova a raiz.





# Eficiência

# Eficiência

- Por definição, uma árvore 2-3 tem altura balanceada com todos nós folhas no mesmo nível.

# Eficiência

- Por definição, uma árvore 2-3 tem altura balanceada com todos nós folhas no mesmo nível.
- No pior caso, todos os nós contêm uma única chave e todos os nós interior somente terão dois filhos.

# Eficiência

- Por definição, uma árvore 2-3 tem altura balanceada com todos nós folhas no mesmo nível.
- No pior caso, todos os nós contêm uma única chave e todos os nós interior somente terão dois filhos.
- Considerando que a altura da árvore 2-3 sempre será  $\log n$ , a operação de busca não tomará mais do que  $\log n$  comparações, resultando  $O(\log n)$  no pior caso.

# Exercícios

- Quantas chaves, no máximo, pode conter uma árvore 2-3 de altura 2? Qual o número mínimo de chaves em uma árvore 2-3 de altura 2? Pense e/ou desenhe exemplos dessas árvores.
- Desenhe a árvore 2-3 que resulta da inserção das chaves [10, 1, 20, 30, 18, 25, 24, 11, 3], nesta ordem, em uma árvore inicialmente vazia.
- O que representa o 2-3 no nome da árvore 2-3?
- Os dados numa árvore 2-3 são mantidos ordenados? Justifique a sua resposta.

# Exercícios

- Qual é a vantagem do uso de árvores 2-3 quando comparadas com árvores binárias de busca?