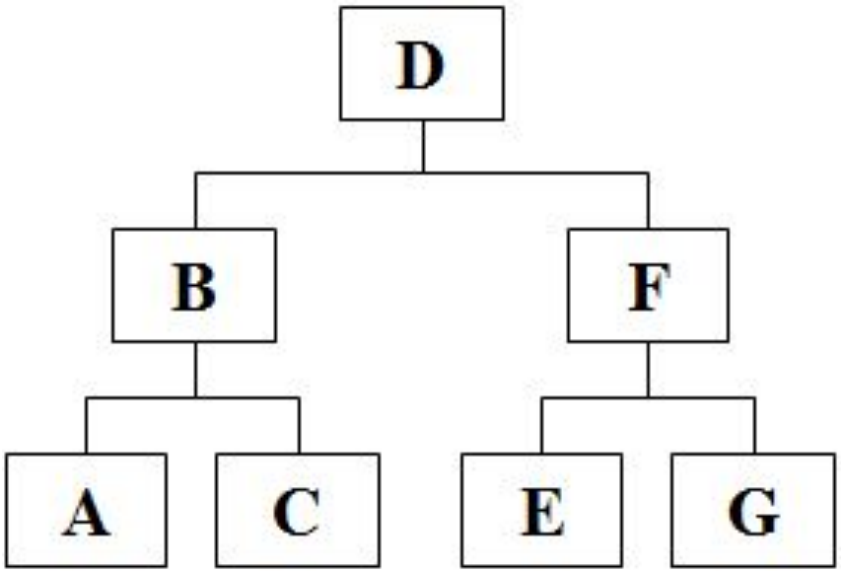
# Árvores Binárias de Busca

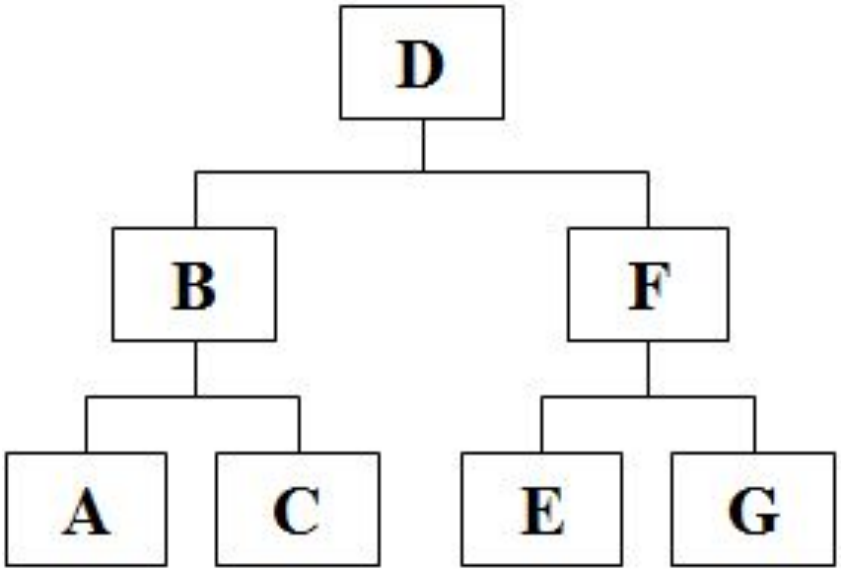
**Prof. Alberto Costa Neto**

# Árvores Binárias de Busca

**Definição:** É uma árvore binária em que **todos os valores na subárvore esquerda são menores que o da raiz** e **todos os valores da subárvore direita são maiores (ou iguais) que o da raiz**.



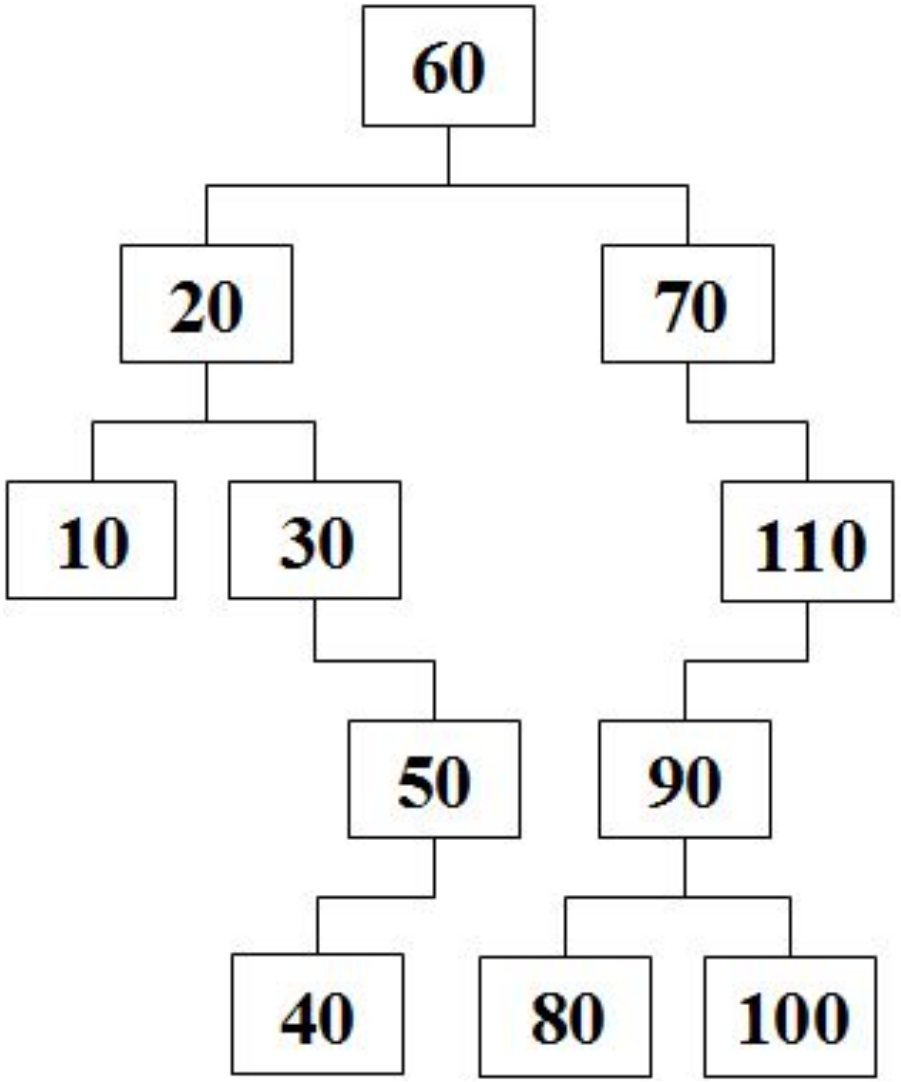
# Exemplo de Árvores Binária de Busca




Usando o **caminhamento central**, teríamos:

A, B, C, D, E, F e G.

# Exemplo de Árvores Binária de Busca

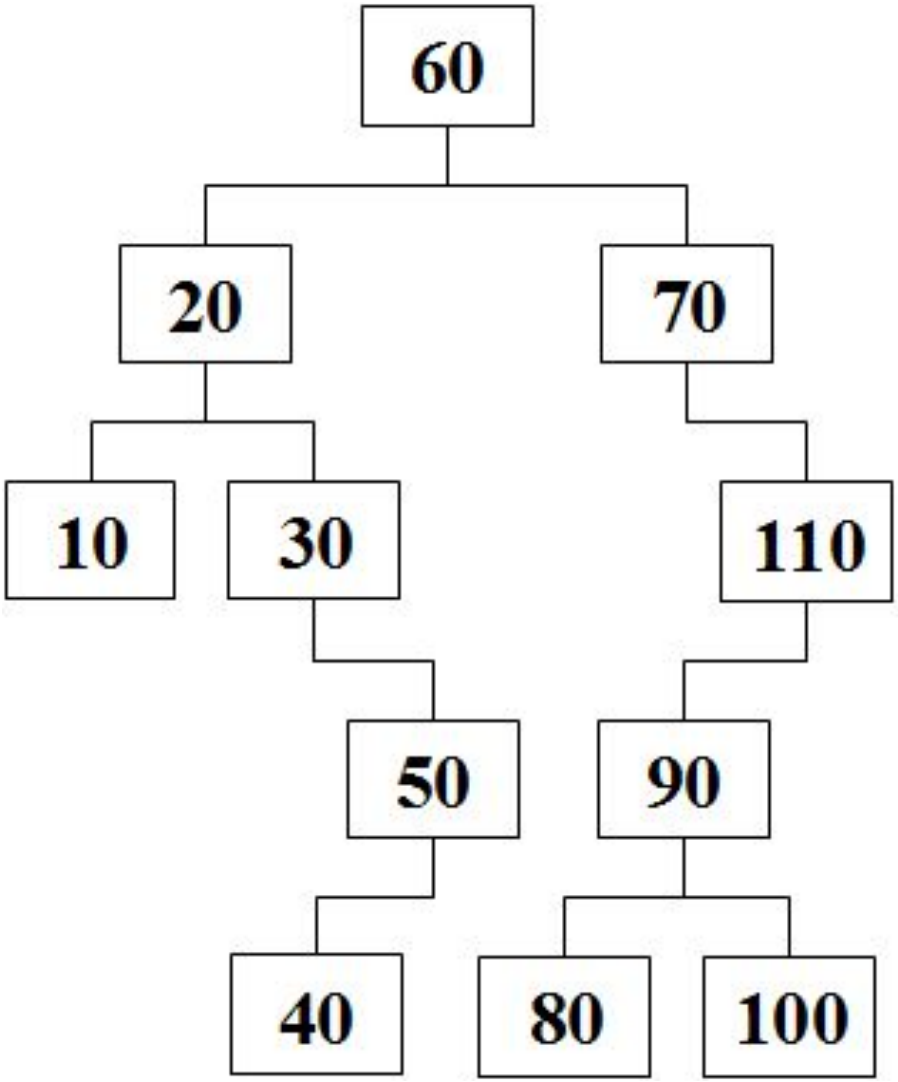



Usando o **caminhamento central**, teríamos:

10, 20, 30, 40, 50, 60, 70, 80, 90, 100 e 110.

# Implementação

# Inserção em Árvore Binária de Busca




Em qual local seriam inseridas as chaves abaixo?

5

12

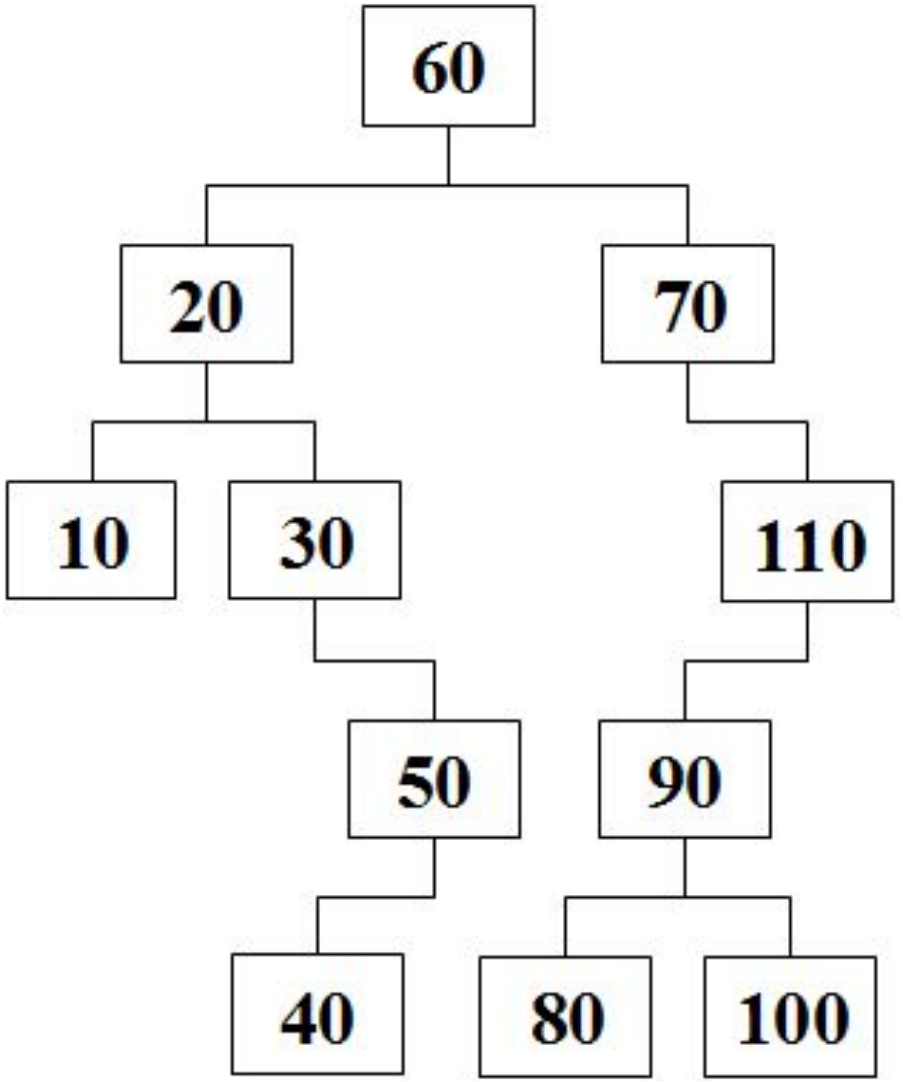
115

98

105

85

# Inserção em Árvore Binária de Busca




Em qual local seriam inseridas as chaves abaixo?

5 (filho esq do 10)

12 (filho dir do 10)

115 (filho dir do 110)

98 (filho esq do 100)

105 (filho dir do 100)

85 (filho dir do 80)

# Inserção (Implementação)

Será usada uma função **encontrarNo** que busca na árvore passada a chave informada.

- Se encontrar, **p** apontará para o nó que a contém e a função retorna true.
- Caso não encontre, a função retorna false e **p** aponta para o nó que seria o seu pai caso existisse.

# Inserção (Implementação)

- A função Inserir usa **encontrarNo** para verificar se a chave já existe. Caso exista, retorna **false**.
- Se não existir, aproveita o apontador **p** (aponta para o pai correto do nó a ser inserido) e:
  - Inclui na **subárvore esquerda** de **p** se a chave sendo inserida **for menor** ou na **direita** se **for maior**, retornando **true**.

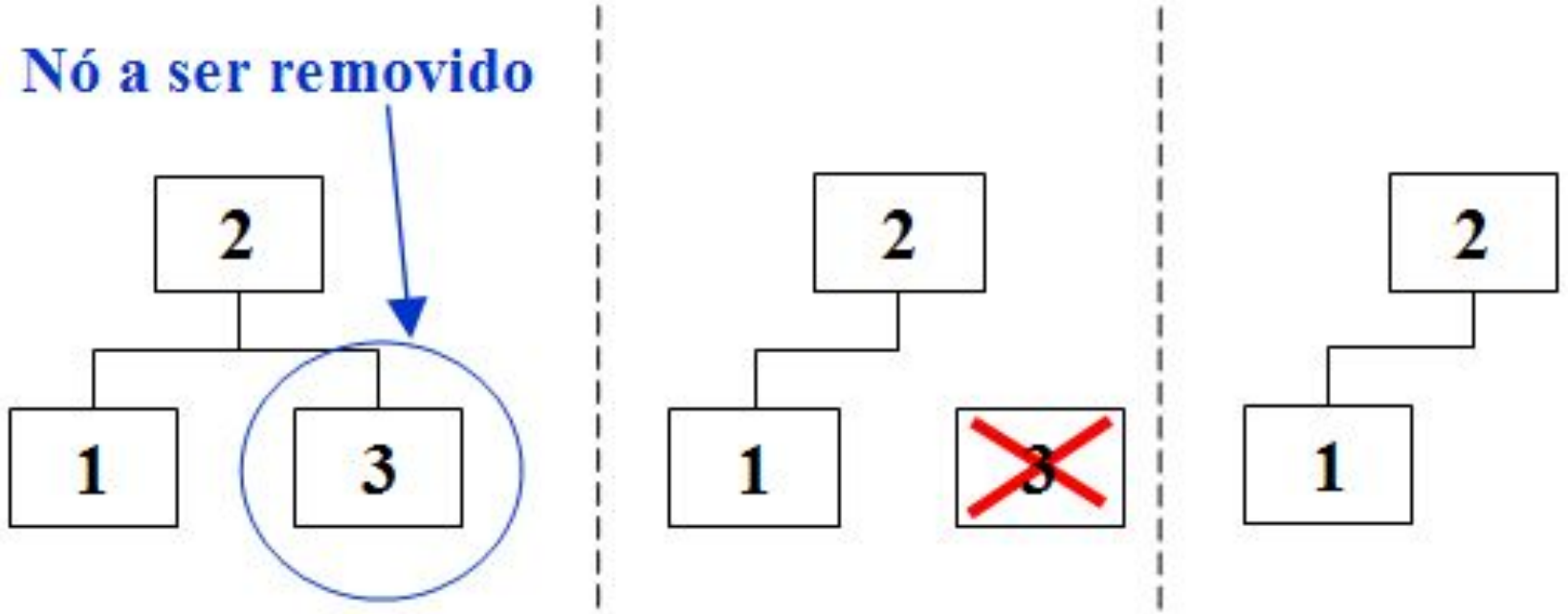
# Remoção

A remoção pode ser dividida em 3 tipos:

- Remoção de um nó folha
- Remoção de um nó com um único filho
- Remoção de um nó com 2 filhos

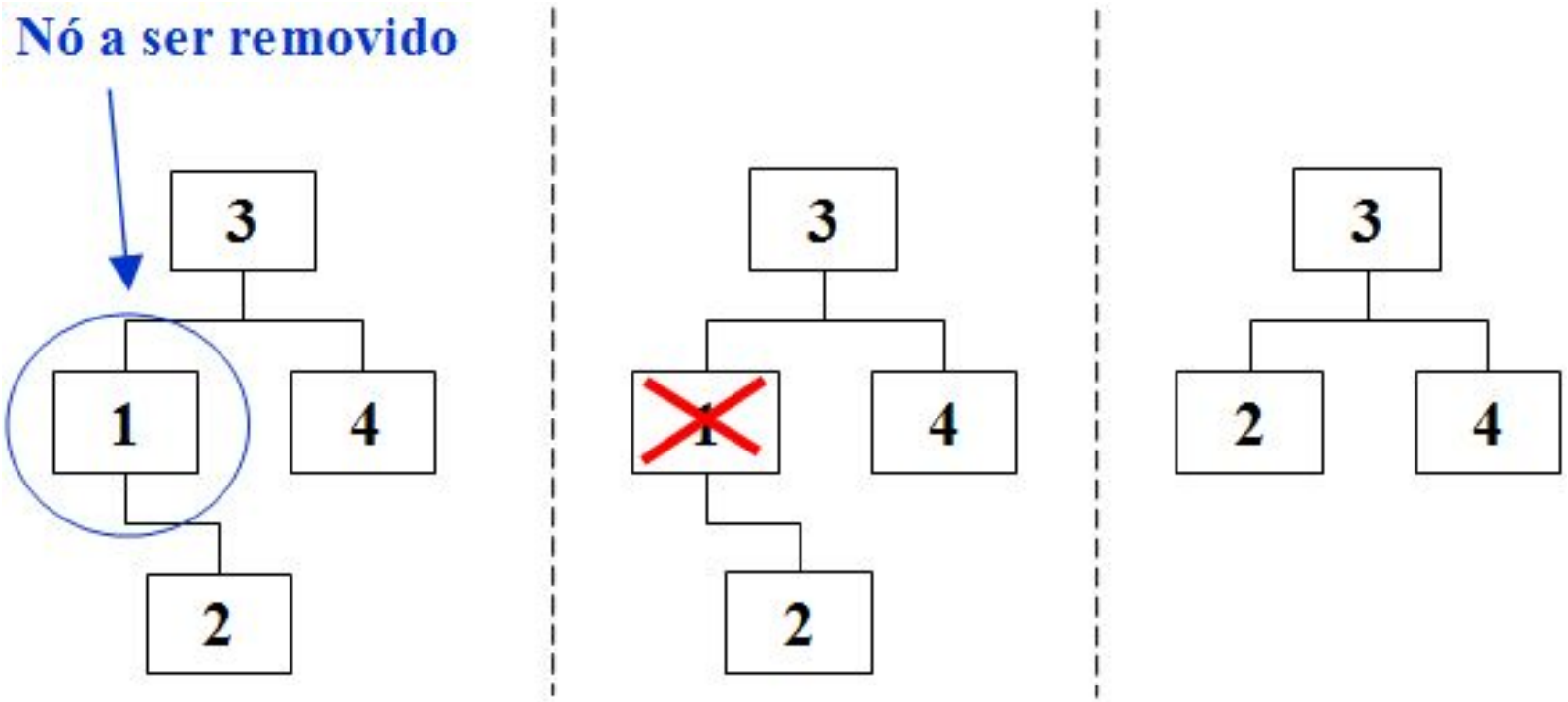
# Removendo nó Folha (sem subárvore)

- Caso mais simples e requer apenas que o nó seja apagado e a referência do pai para ele seja ajustada.

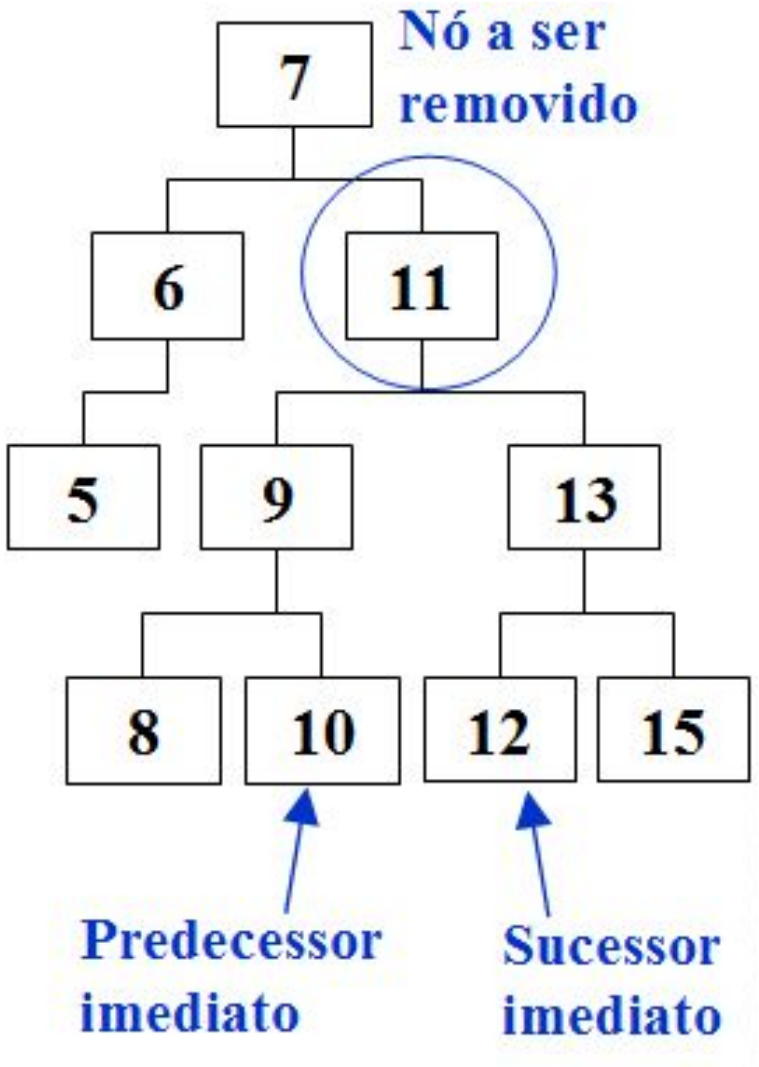


# Removendo nó com uma subárvore

- Neste caso, basta fazer com que o único filho do nó a ser removido assuma seu lugar



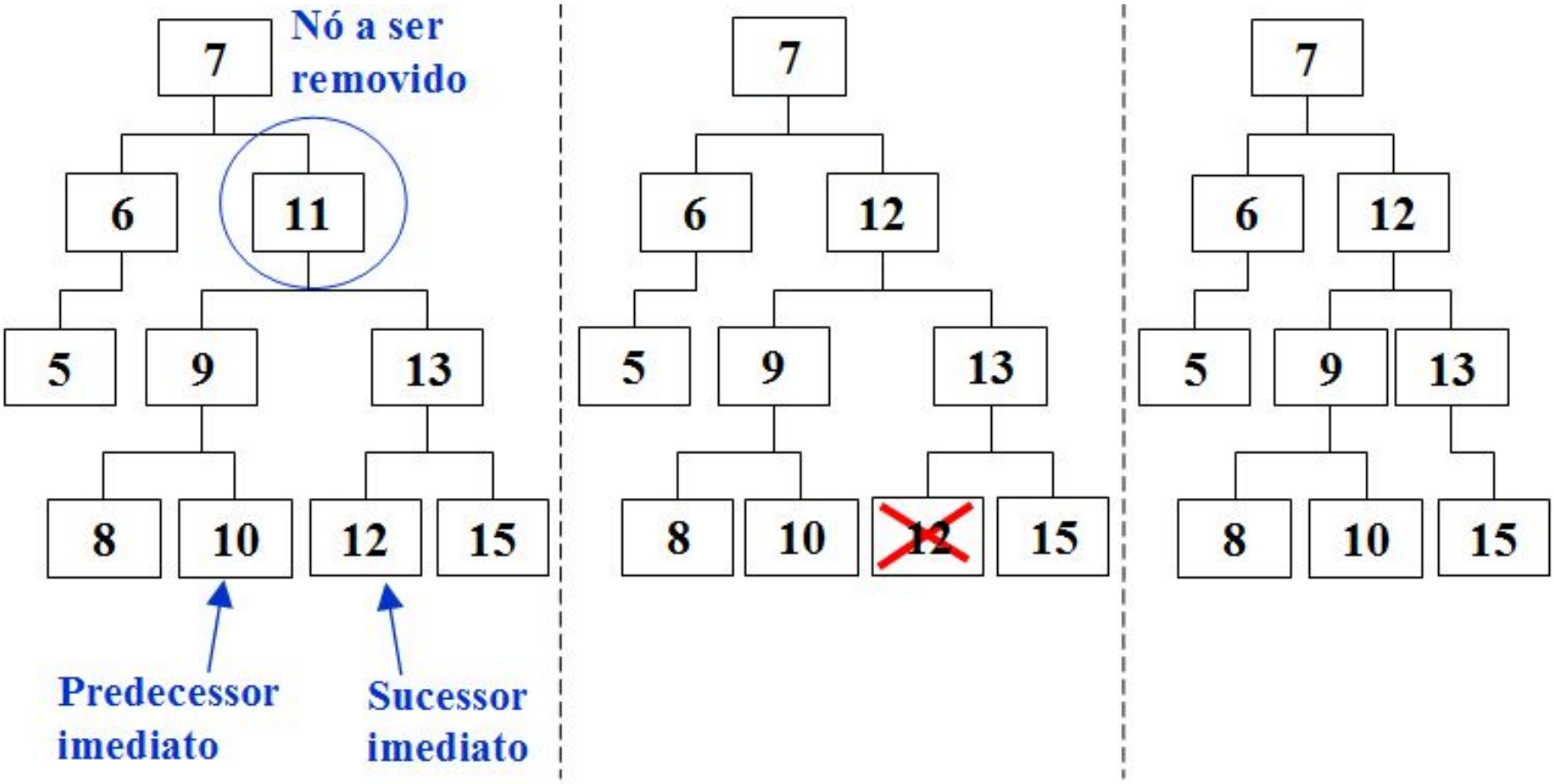
# Removendo nó com 2 subárvores




Nesse caso é necessário encontrar o **sucessor imediato**, **copiar o seu conteúdo para o nó a ser removido** e **remover o nó que continha o sucessor imediato**.

# Removendo nó com 2 subárvores

- O sucesso imediato é colocado no lugar do nó removido.



# Sugestões de Estudo

**Estruturas de Dados (Nina Edelweiss)**

- Seção 6.4

**Projeto de Algoritmos com implementações em Java e C++ (Nivio Ziviani)**

- Seção 5.3.1

**Estruturas de dados (Paulo Veloso)**

- Seção 7.10