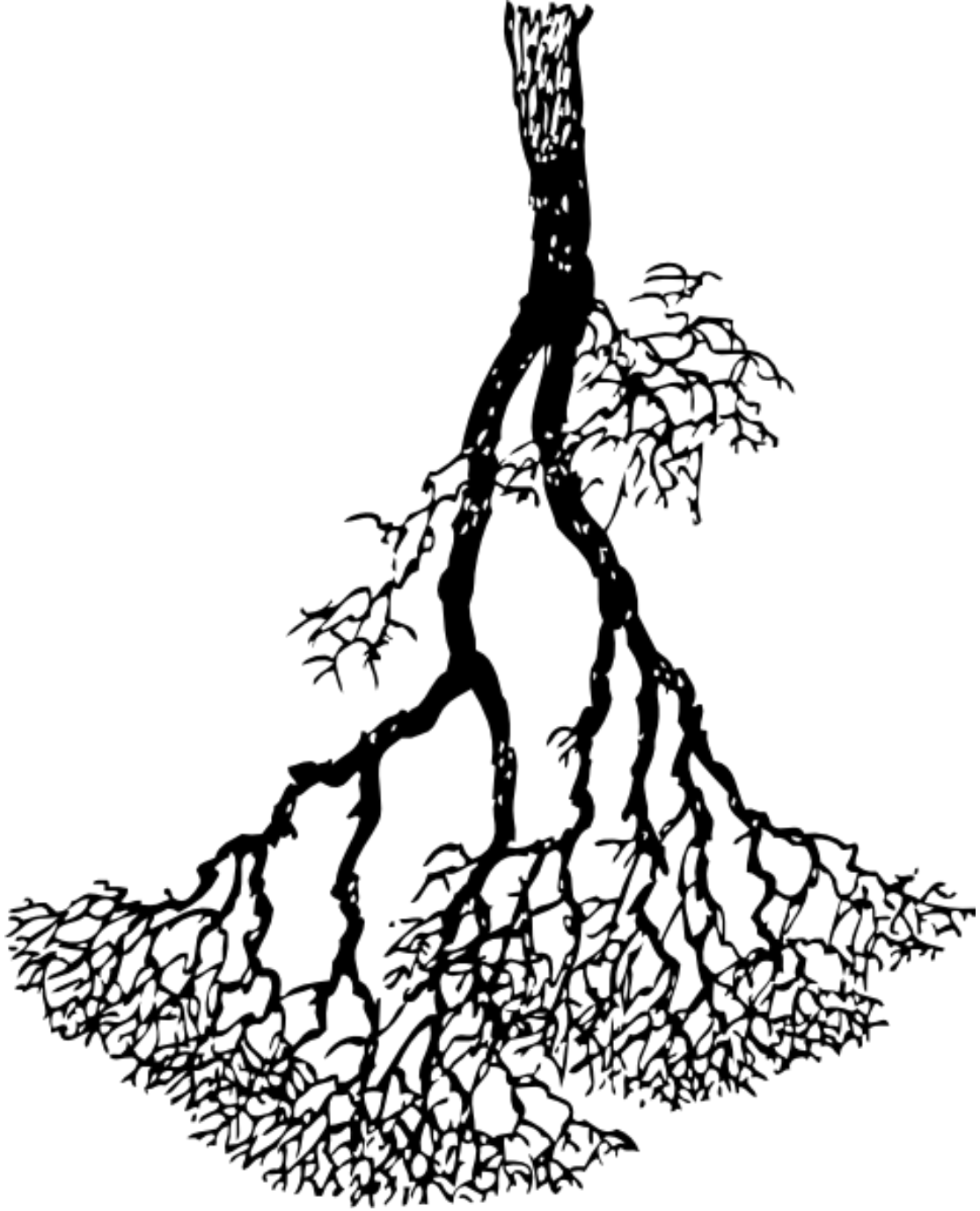
ALGORITMOS E ESTRUTURA DE DADOS

Curso de Engenharia de Software
Lucas Sampaio Leite

Árvores



Fonte imagem: https://www.ime.usp.br/~pf/algoritmos/aulas/bint.html

INSTITUTO FEDERAL
Pernambuco

Árvores

INSTITUTO FEDERAL
Pernambuco

- Árvores são estruturas de dados hierárquicas compostas por nós conectados por arestas.
- São amplamente utilizadas em diversas áreas da Computação, como:
  - Implementação de algoritmos de busca e ordenação;
  - Sistemas de gerenciamento de bancos de dados;
  - Organização de sistemas de arquivos em sistemas operacionais;
  - Representação de interfaces gráficas e documentos estruturados;
  - Compiladores e interpretadores de linguagens de programação.

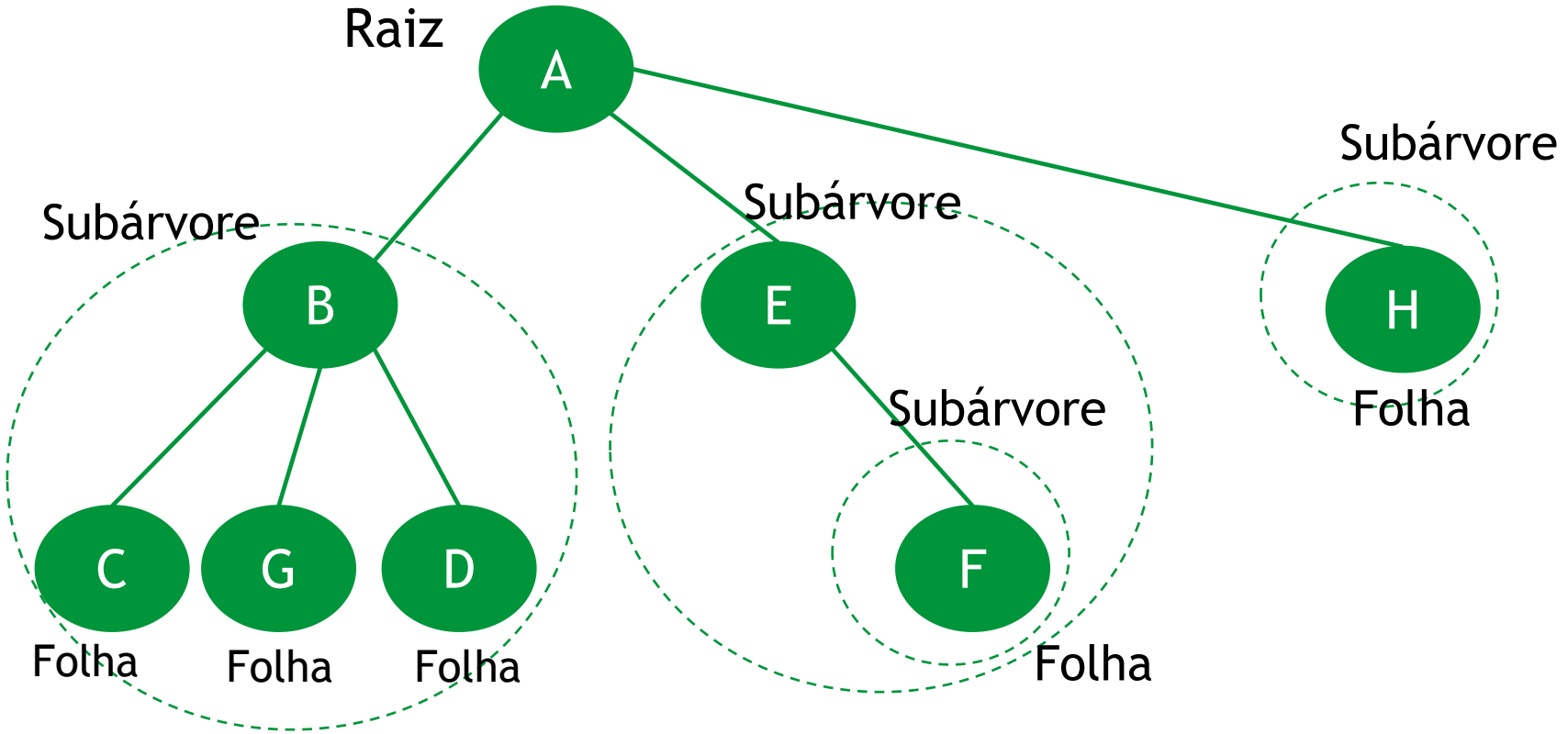
Árvores

INSTITUTO FEDERAL
Pernambuco

- Definição: uma árvore T é um conjunto finito de zero ou mais nós, tal que:
  - se o número de nós = 0, temos uma árvore vazia, ou
  - se o número de nós > 0
    - existe um nó especialmente denominado raiz de T
    - os nós restantes formam $m \geq 0$ conjuntos disjuntos $p_1, p_2, \ldots, p_m$, cada um desses conjuntos é uma árvore em si, chamada subárvore da raiz de T, ou simplesmente subárvore.

Árvores

- Uma árvore é um conjunto de nós formado por uma raiz (root) e por um conjunto de subárvores conectadas a ela.



Árvores

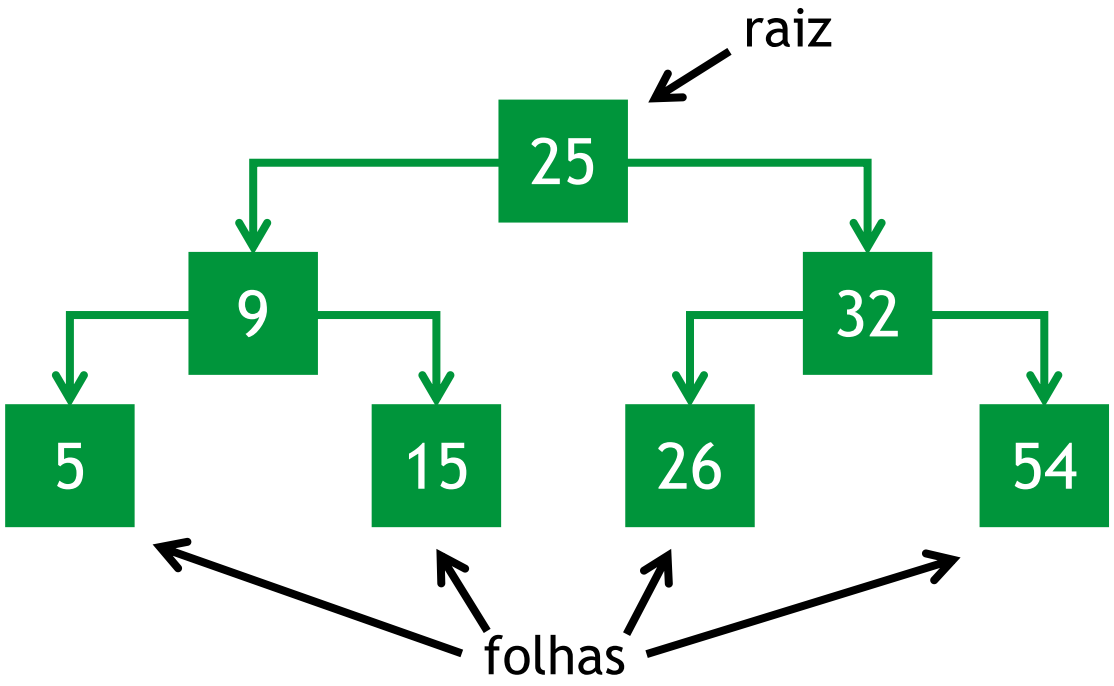
INSTITUTO FEDERAL
Pernambuco

- Cada nó pode possuir zero ou mais filhos.
- Uma árvore pode ser definida recursivamente como:
  - Uma árvore vazia; ou
  - Um nó raiz conectado a zero ou mais subárvores.
  - Como a definição é recursiva, cada subárvore também é uma árvore e pode, inclusive, ser uma árvore vazia.

Árvores

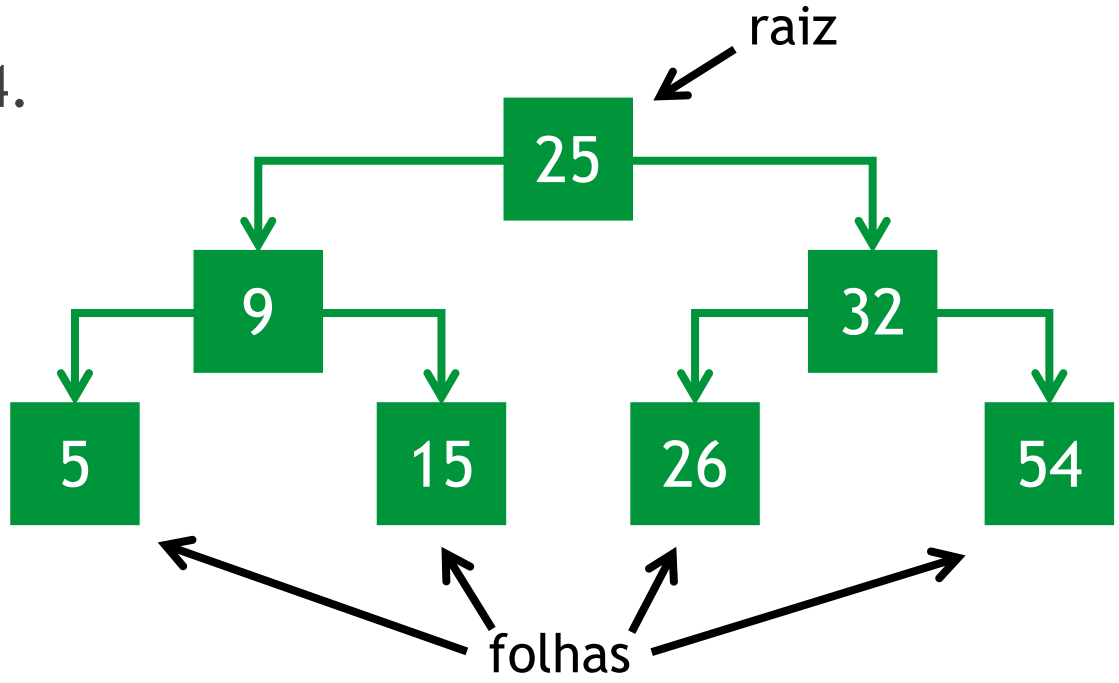
INSTITUTO FEDERAL
Pernambuco

- Nós que não possuem filhos (grau 0) são chamados de folhas ou nós externos.
- Nós que possuem pelo menos um filho são chamados de nós internos.



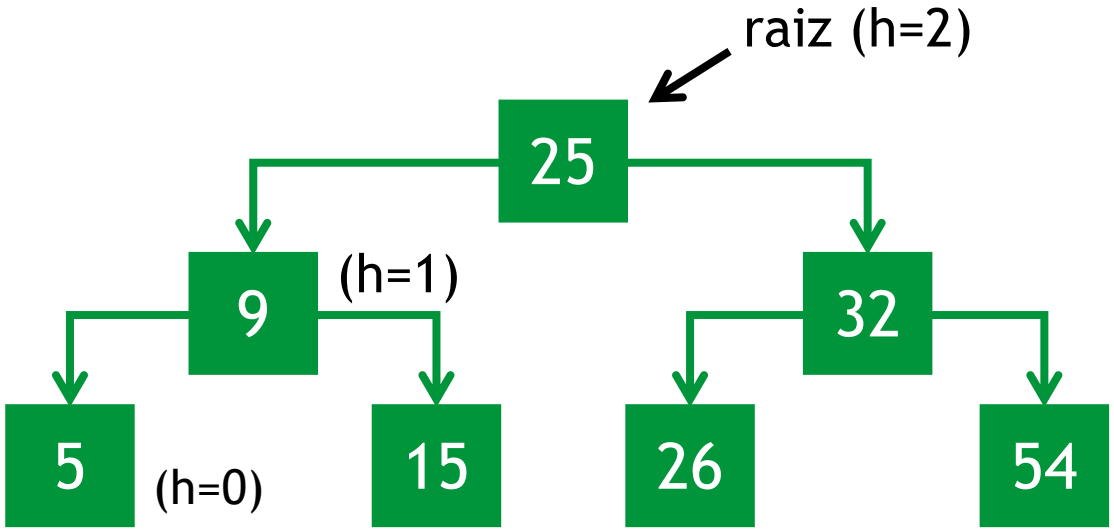
Árvores

- Os nós localizados abaixo de um determinado nó são chamados de descendentes desse nó.
- Exemplos:
  - Descendentes de 9: 5 e 15.
  - Descendentes de 32: 26 e 54.



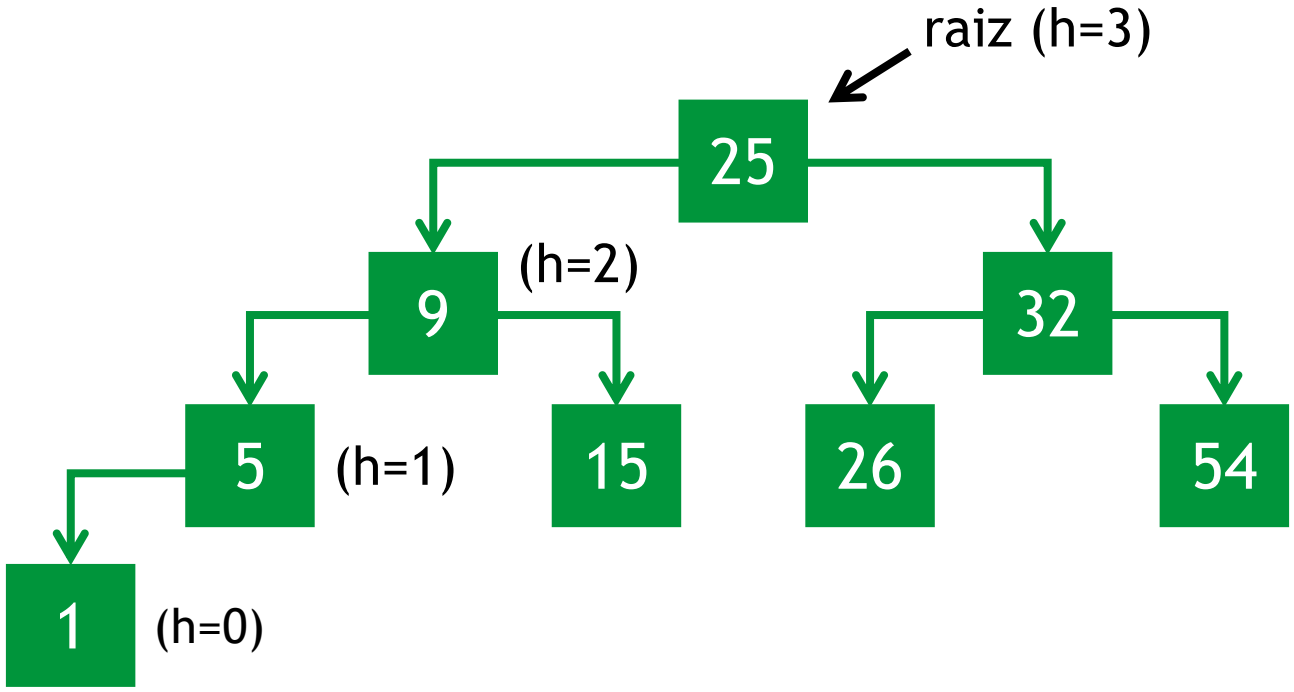
Árvores

- A altura de uma árvore (h) é definida pelo número de arestas do caminho mais longo entre a raiz e uma folha.



Árvores

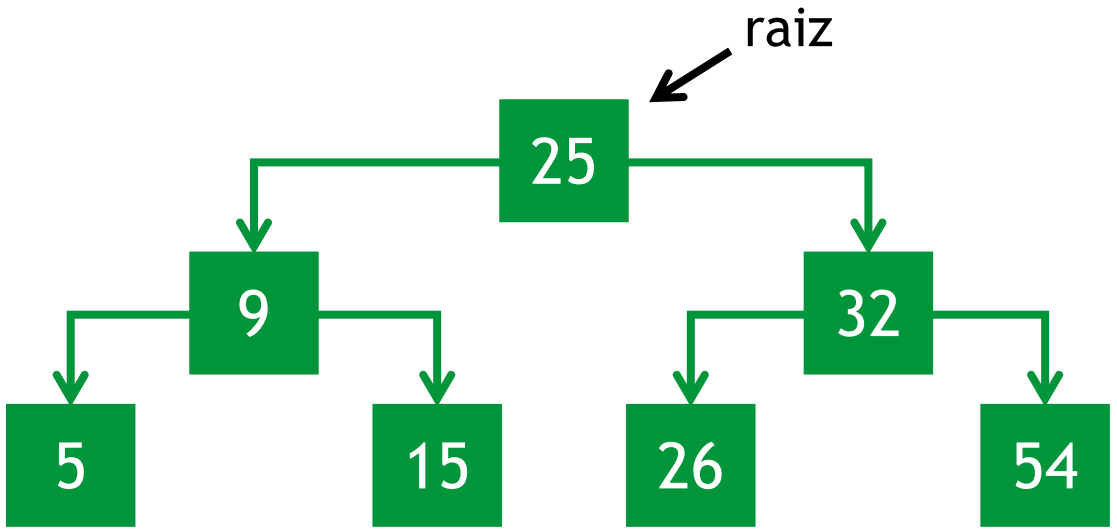
- A altura de uma árvore (h) é definida pelo número de arestas do caminho mais longo entre a raiz e uma folha.



Árvores

INSTITUTO FEDERAL
Pernambuco

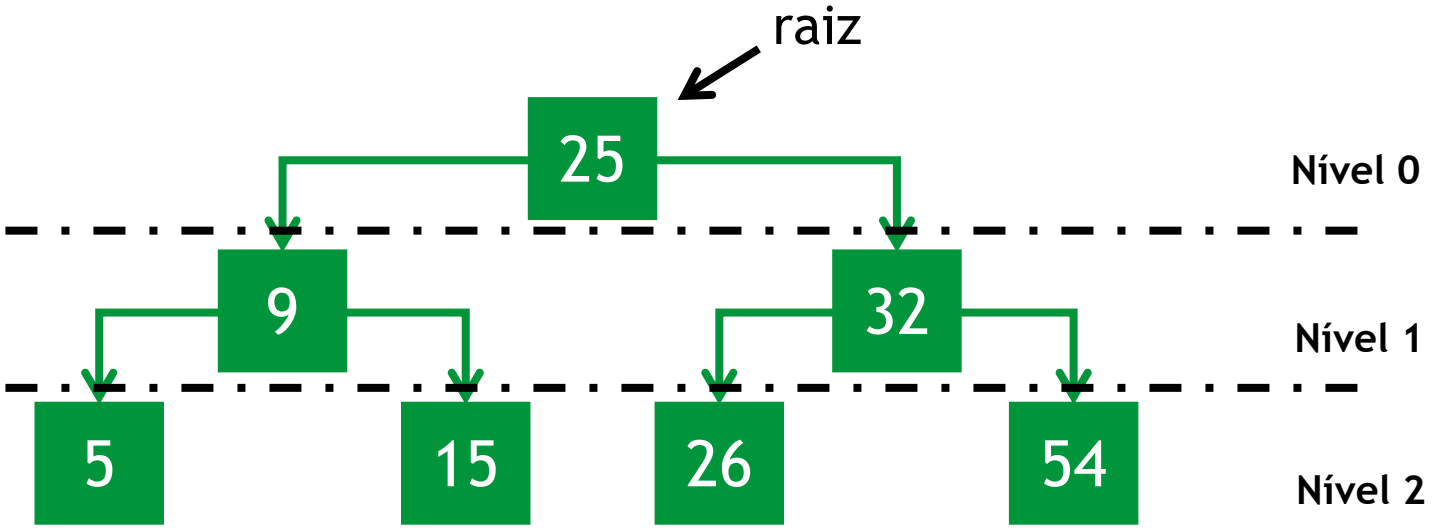
- A profundidade de um nó corresponde ao número de arestas no caminho da raiz até esse nó.
- Exemplos:
  - Profundidade de 25 = 0
  - Profundidade de 9 = 1
  - Profundidade de 5 = 2



Árvores

INSTITUTO FEDERAL
Pernambuco

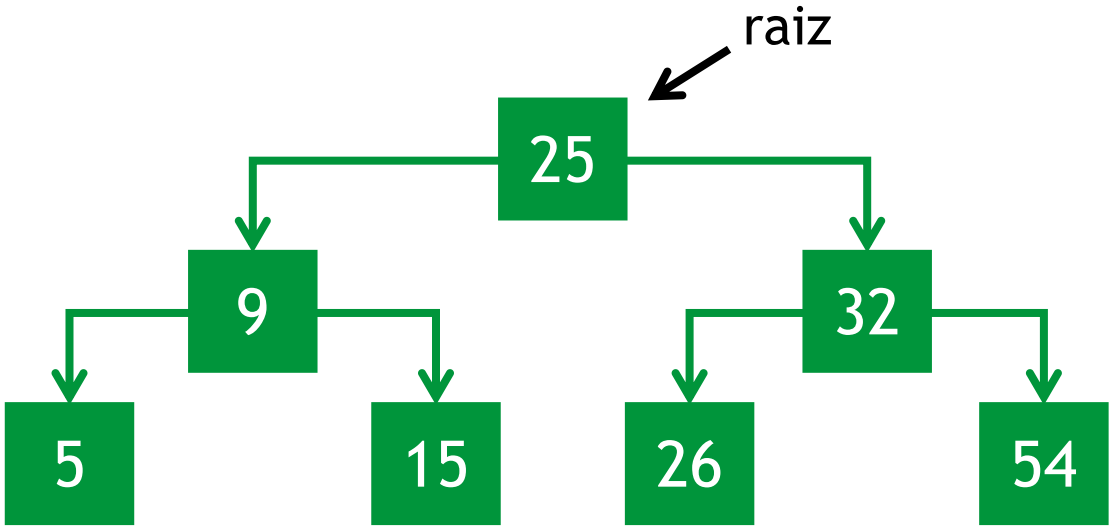
- O nível de um nó em uma árvore é uma medida de sua profundidade em relação à raiz da árvore.
  - O nível da raiz é 0.
  - Os demais níveis são a distância até a raiz.



Árvores binárias

INSTITUTO FEDERAL
Pernambuco

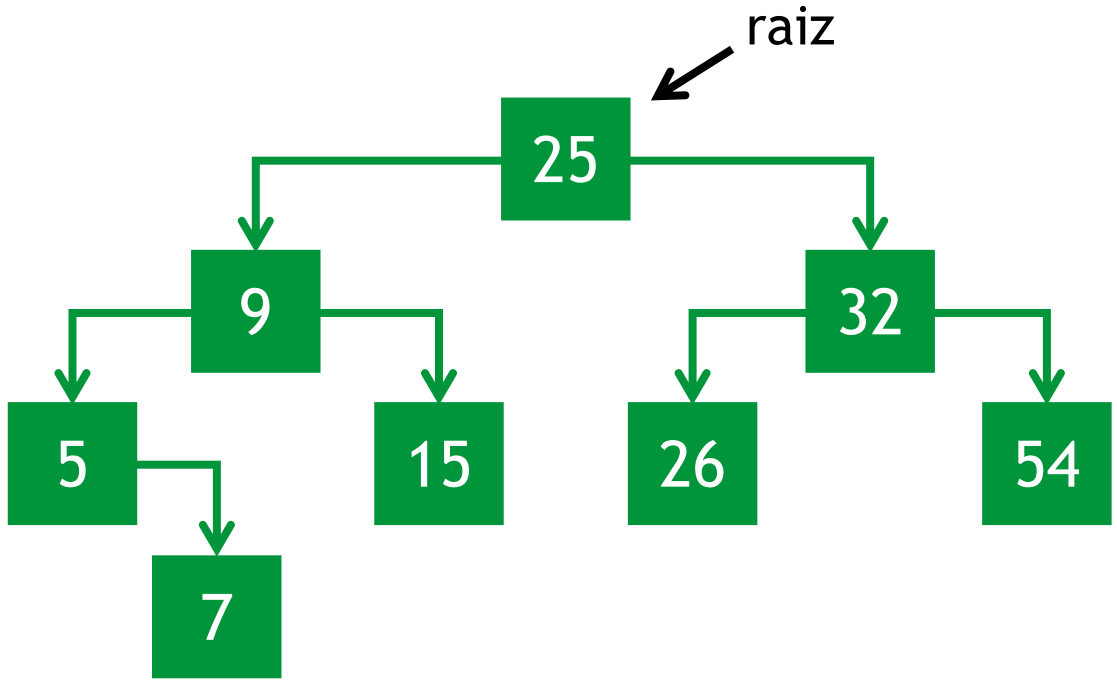
- Uma árvore é binária quando cada nó possui, no máximo, duas subárvores.



Árvores binárias

INSTITUTO FEDERAL
Pernambuco

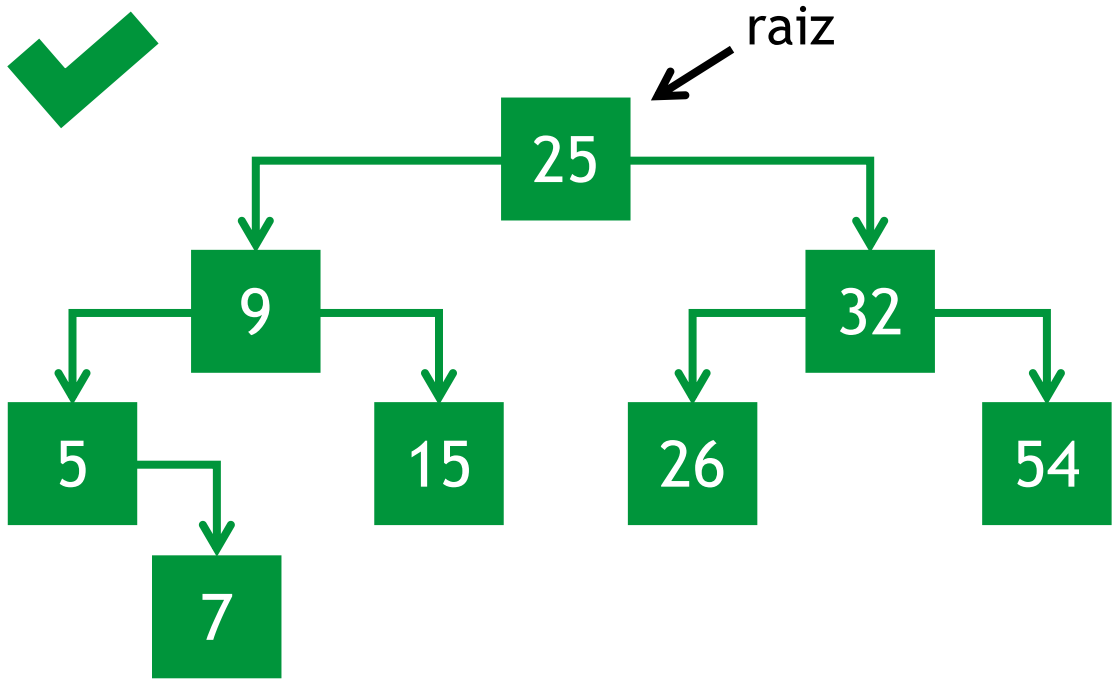
- Uma árvore é binária quando cada nó possui, no máximo, duas subárvores.



Árvores binárias

INSTITUTO FEDERAL
Pernambuco

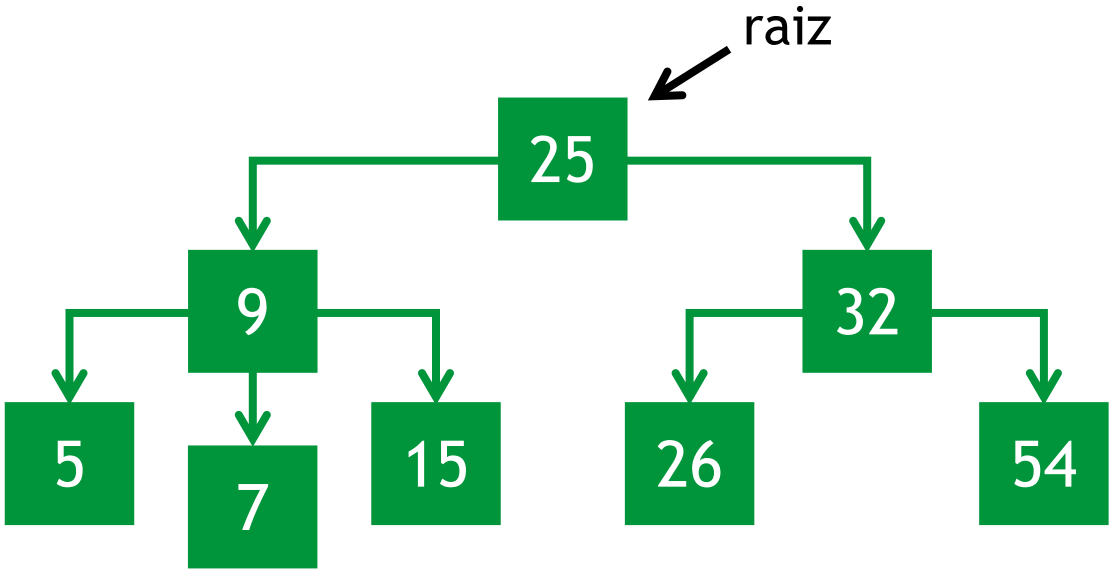
- Uma árvore é binária quando cada nó possui, no máximo, duas subárvores.



Árvores binárias

INSTITUTO FEDERAL
Pernambuco

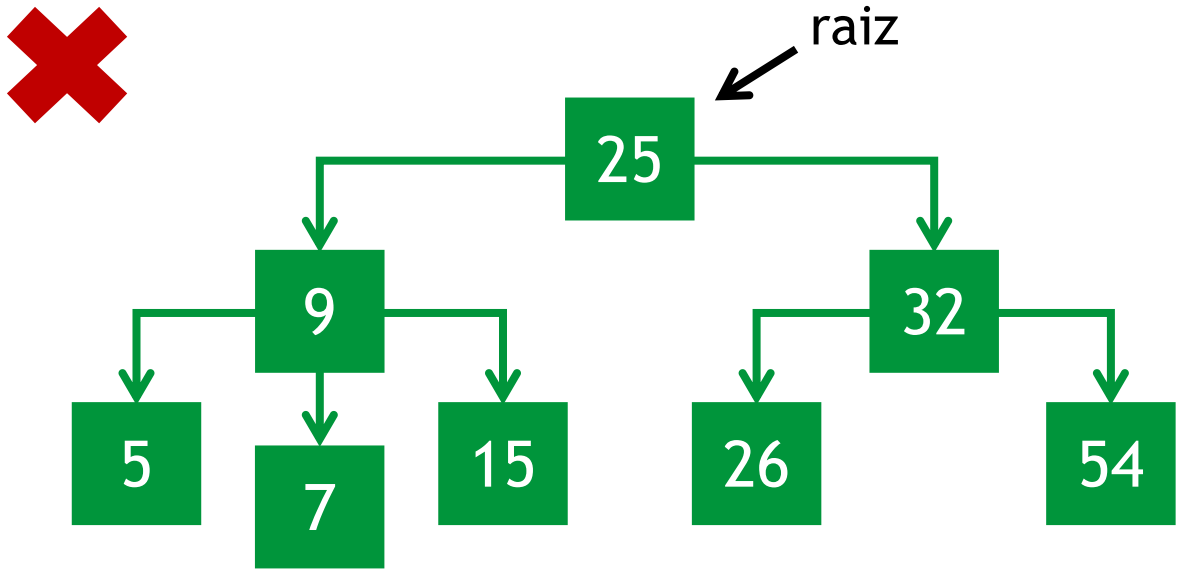
- Uma árvore é binária quando cada nó possui, no máximo, duas subárvores.



Árvores binárias

INSTITUTO FEDERAL
Pernambuco

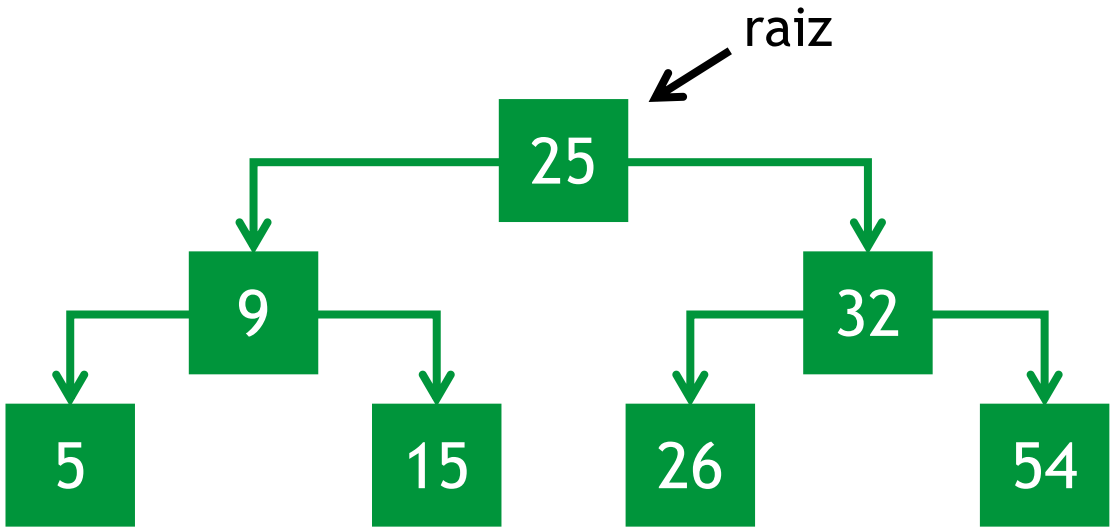
- Uma árvore é binária quando cada nó possui, no máximo, duas subárvores.

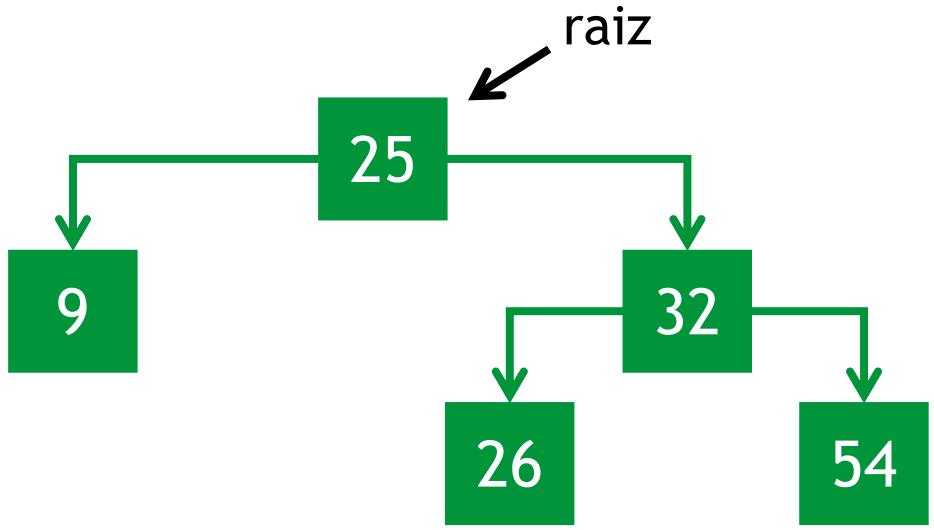


Árvores binárias

INSTITUTO FEDERAL
Pernambuco

- Uma árvore binária é cheia quando todo nó possui 0 ou 2 filhos.

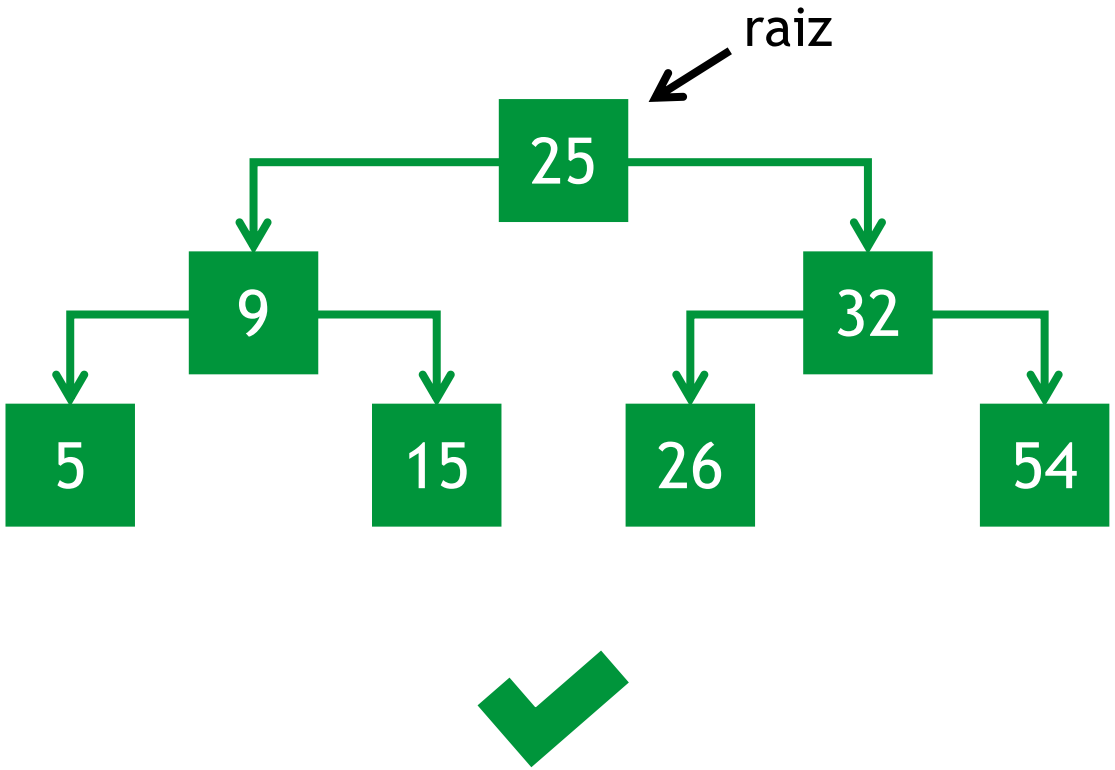


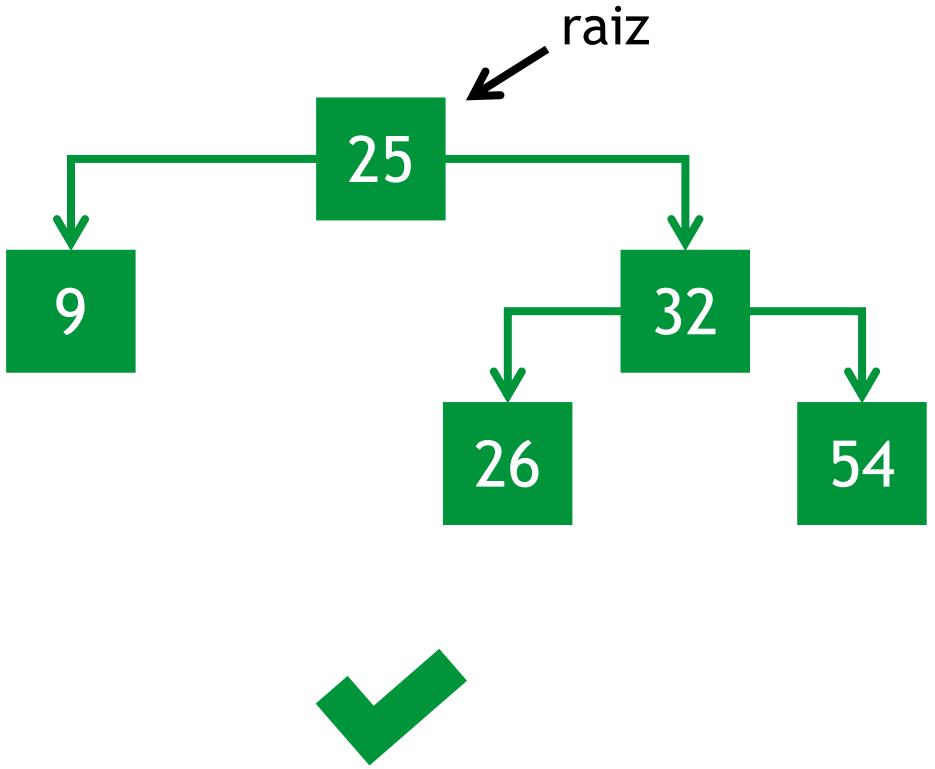


Árvores binárias

INSTITUTO FEDERAL
Pernambuco

- Uma árvore binária é cheia quando todo nó possui 0 ou 2 filhos.

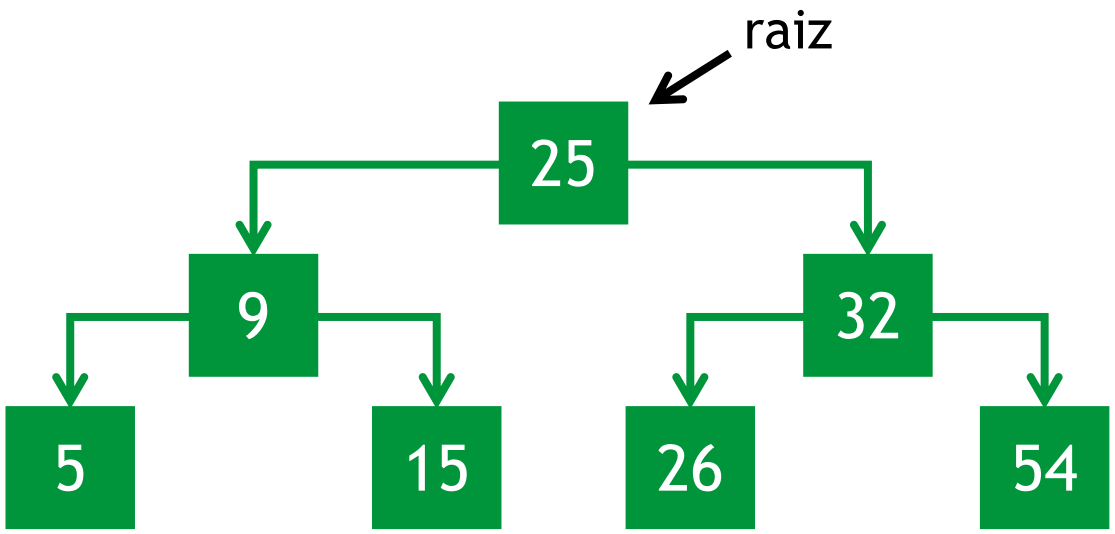


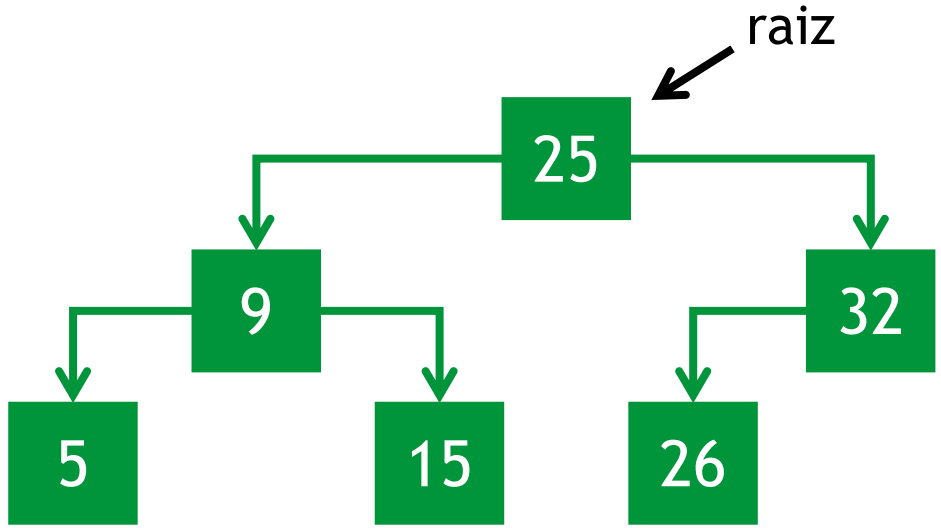


Árvores binárias

INSTITUTO FEDERAL
Pernambuco

- Uma árvore binária é completa quando:
  - Todos os níveis, exceto possivelmente o último, estão totalmente preenchidos.
  - O último nível é preenchido da esquerda para a direita.

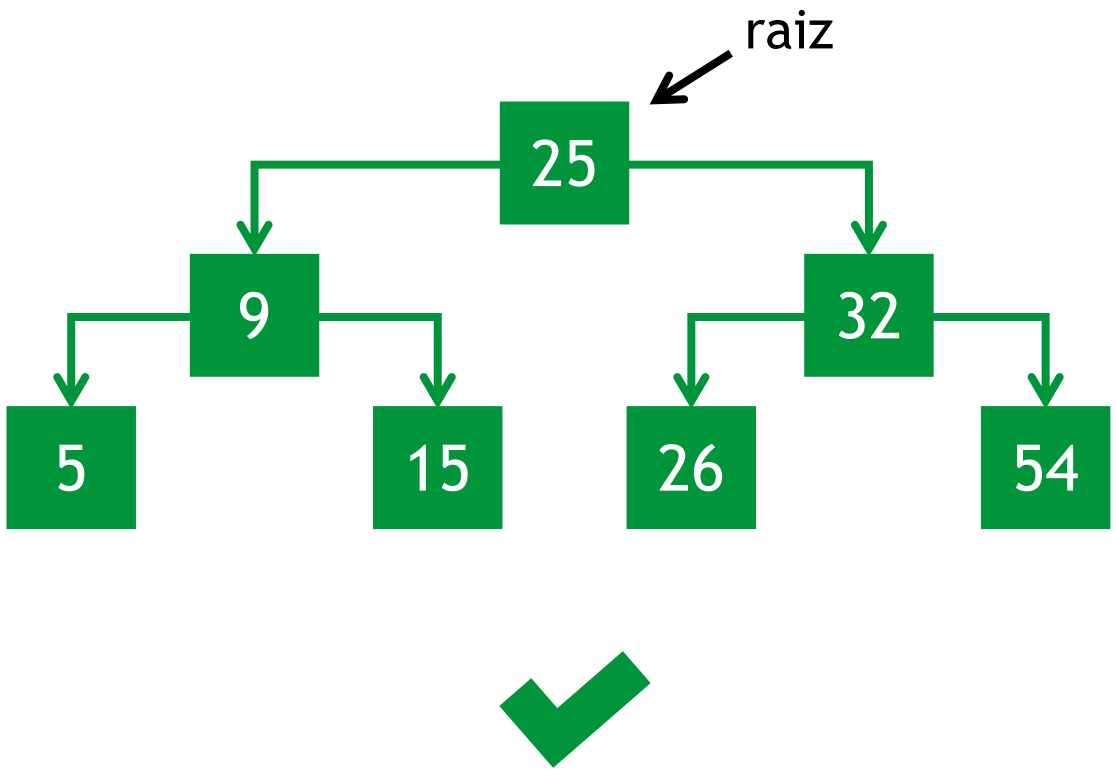


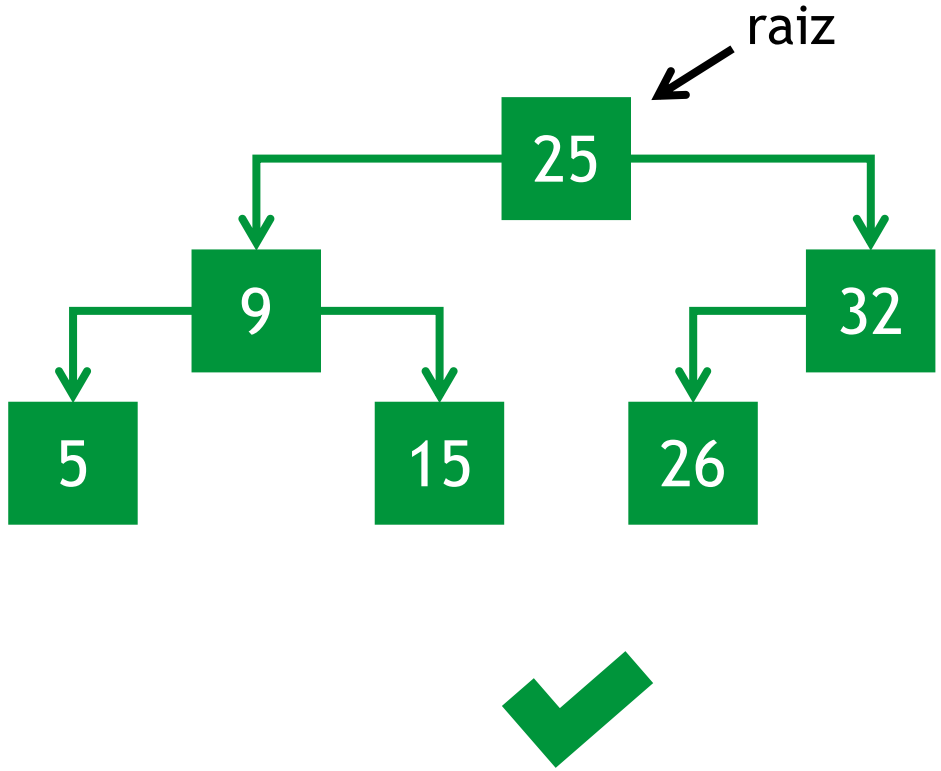


Árvores binárias

INSTITUTO FEDERAL
Pernambuco

- Uma árvore binária é completa quando:
  - Todos os níveis, exceto possivelmente o último, estão totalmente preenchidos.
  - O último nível é preenchido da esquerda para a direita.

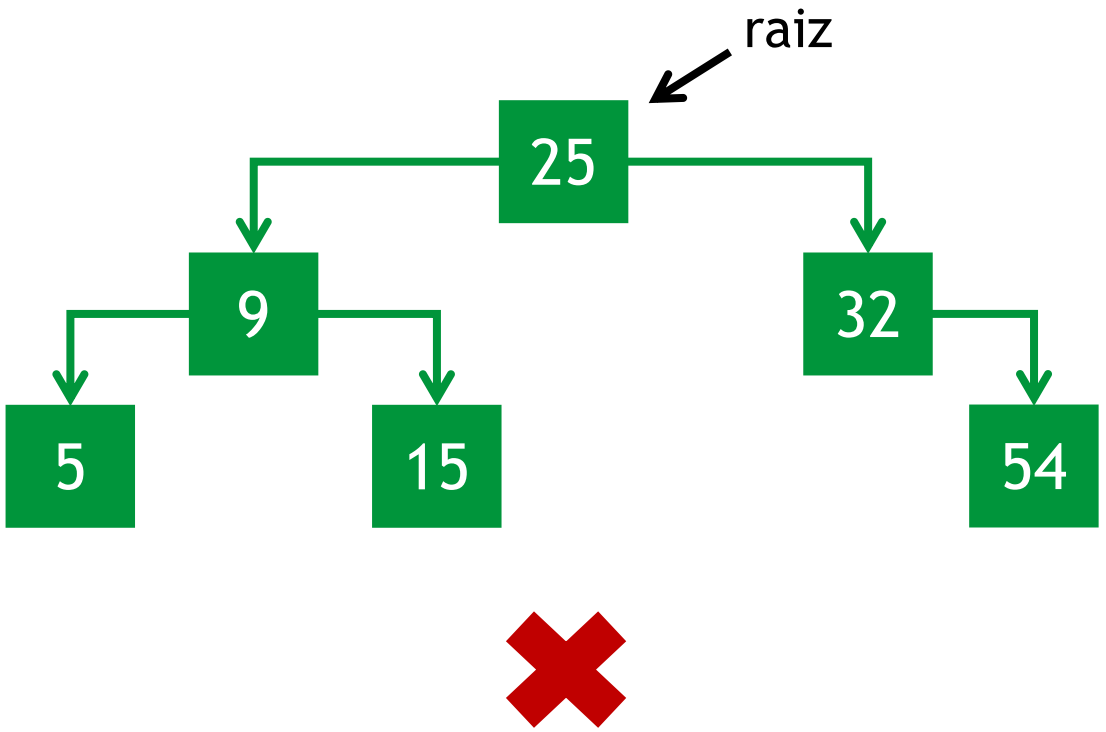




Árvores binárias

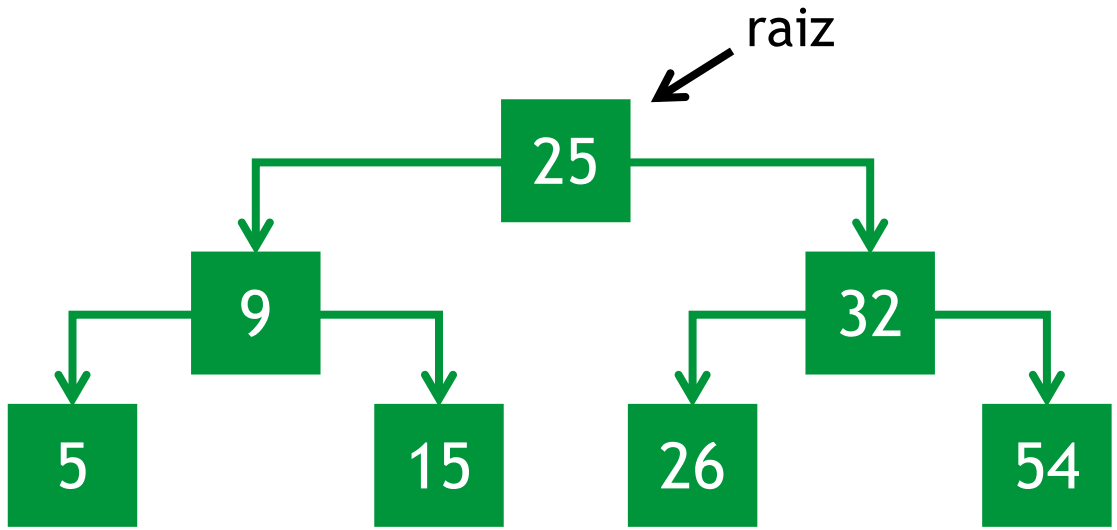
INSTITUTO FEDERAL
Pernambuco

- Uma árvore binária é completa quando:
  - Todos os níveis, exceto possivelmente o último, estão totalmente preenchidos.
  - O último nível é preenchido da esquerda para a direita.



Árvores binárias

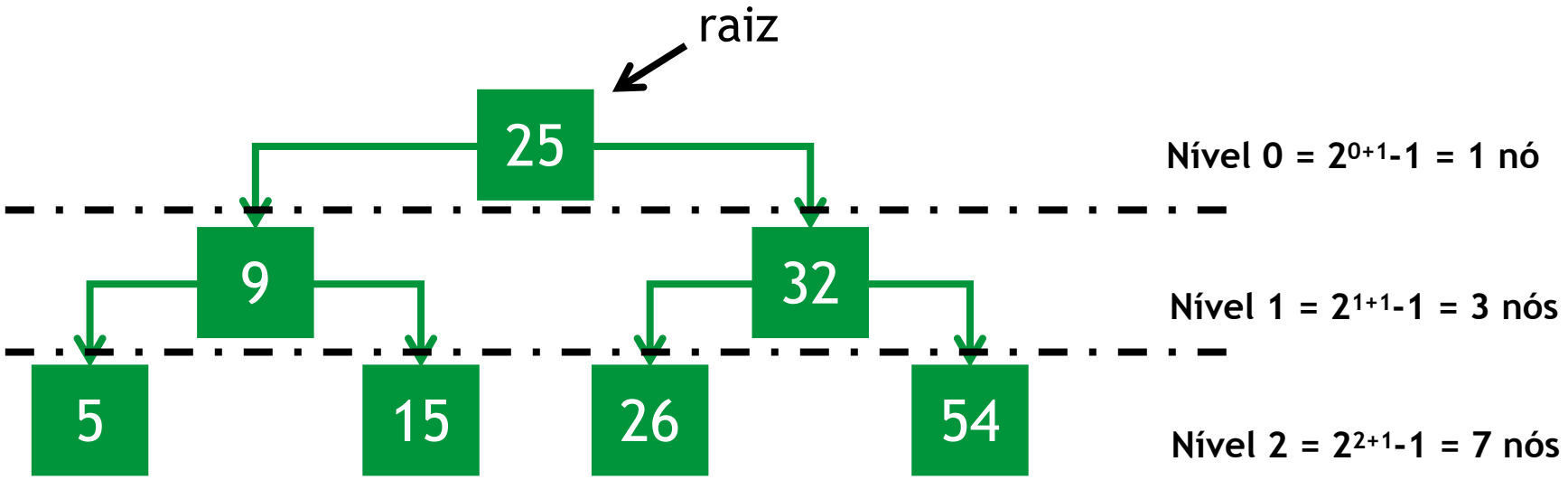
- Uma árvore binária é perfeita quando:
  - Todo nó interno possui exatamente 2 filhos.
  - Todas as folhas estão no mesmo nível.



Árvores binárias

INSTITUTO FEDERAL
Pernambuco

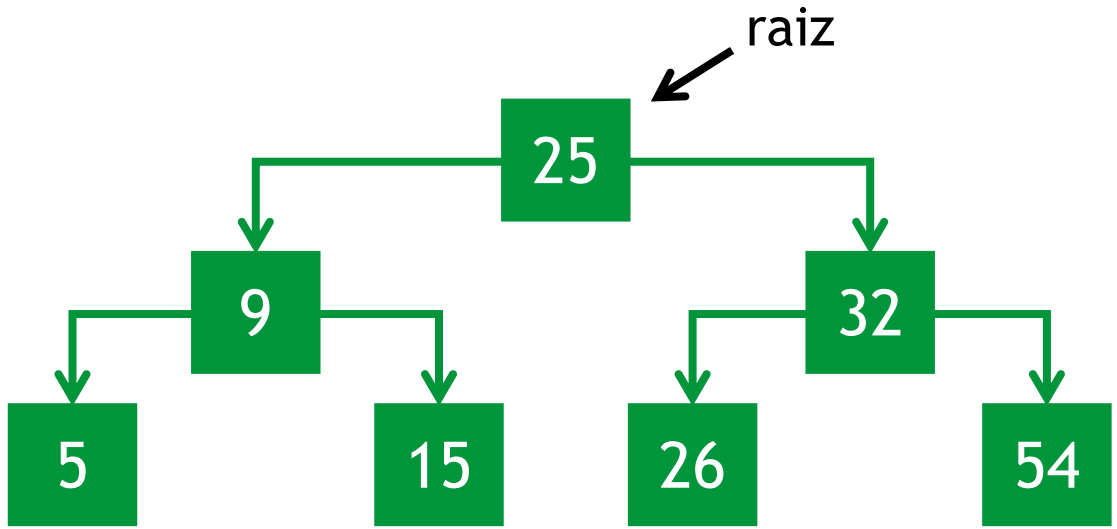
- Uma árvore binária perfeita de altura h possui $2^{h+1} - 1$ nós.
- Uma árvore binária completa com n nós possui altura proporcional a $log_2 n$.



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

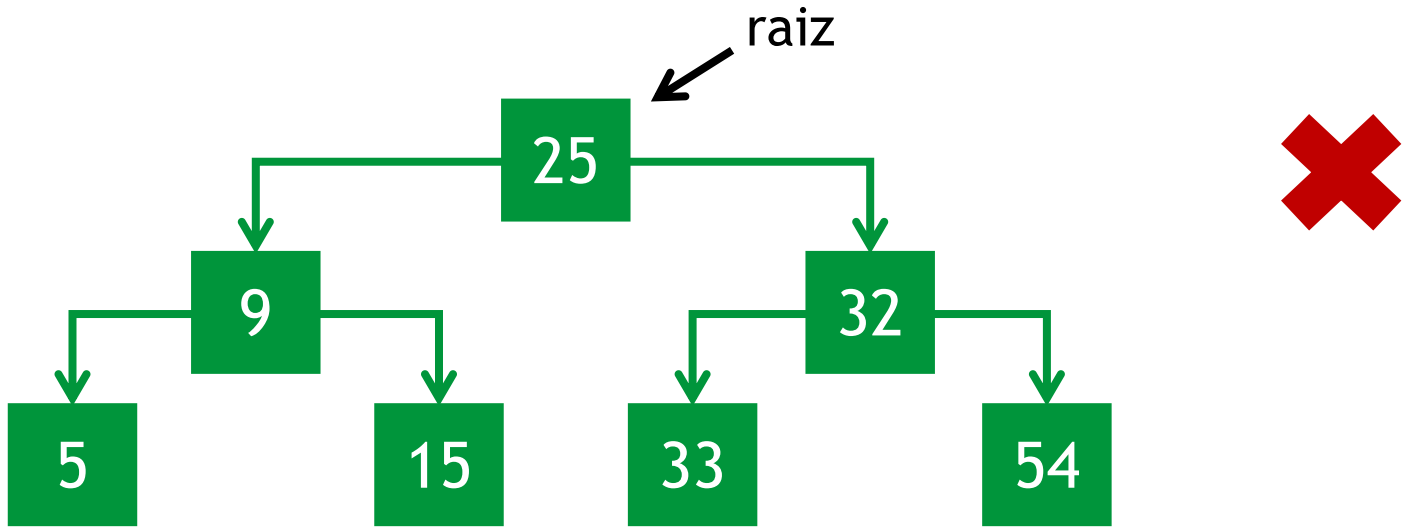
- Uma Árvore Binária de Busca (BST) organiza os dados de forma hierárquica, permitindo operações eficientes de busca, inserção e remoção.
  - Valores menores que o nó são armazenados à esquerda.
  - Valores maiores que o nó são armazenados à direita.



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Uma Árvore Binária de Busca (BST) organiza os dados de forma hierárquica, permitindo operações eficientes de busca, inserção e remoção.
  - Valores menores que o nó são armazenados à esquerda.
  - Valores maiores que o nó são armazenados à direita.



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Representando um Nó em C:

```c
typedef struct No {
    int valor;
    struct No *esquerda;
    struct No *direita;
} No;
```

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Criando um Nó em C:

```c
No *criarNo(int valor) {

    No *novo = (No *) malloc(sizeof(No));

    novo->valor = valor;
    novo->esquerda = NULL;
    novo->direita = NULL;

    return novo;
}
```

Árvore Binária de Busca (BST)

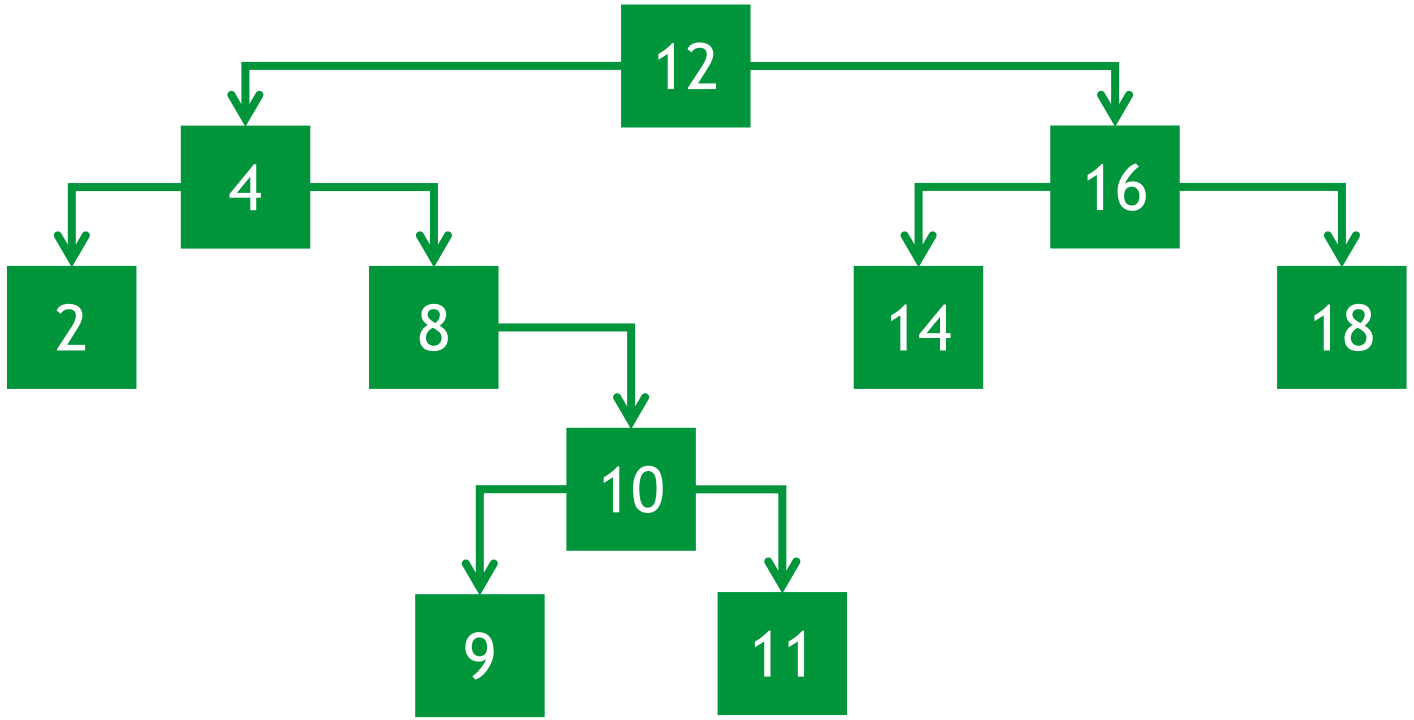
INSTITUTO FEDERAL
Pernambuco

- Busca em uma Árvore Binária de Busca (BST):
    1. Compare o valor armazenado no nó atual com o valor procurado.
    2. Se os valores forem iguais, o elemento foi encontrado e o nó é retornado.
    3. Se o valor procurado for menor que o valor do nó atual, continue a busca na subárvore esquerda.
    4. Se o valor procurado for maior que o valor do nó atual, continue a busca na subárvore direita.
    5. Repita os passos 1 a 4 até encontrar o elemento ou alcançar uma referência nula (NULL).
    6. Se a busca chegar a um nó nulo (NULL), significa que o elemento não está presente na árvore; nesse caso, retorne NULL.

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

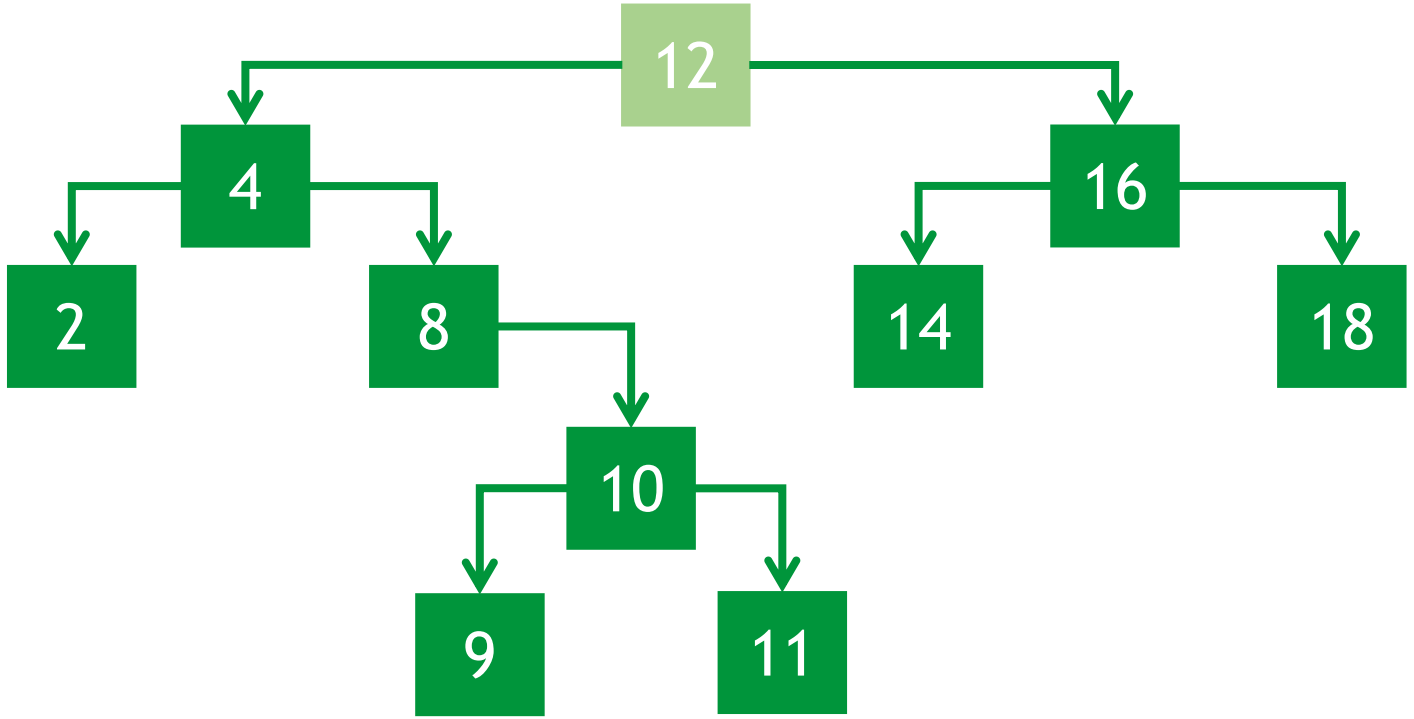
- Buscando o elemento de valor 10:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

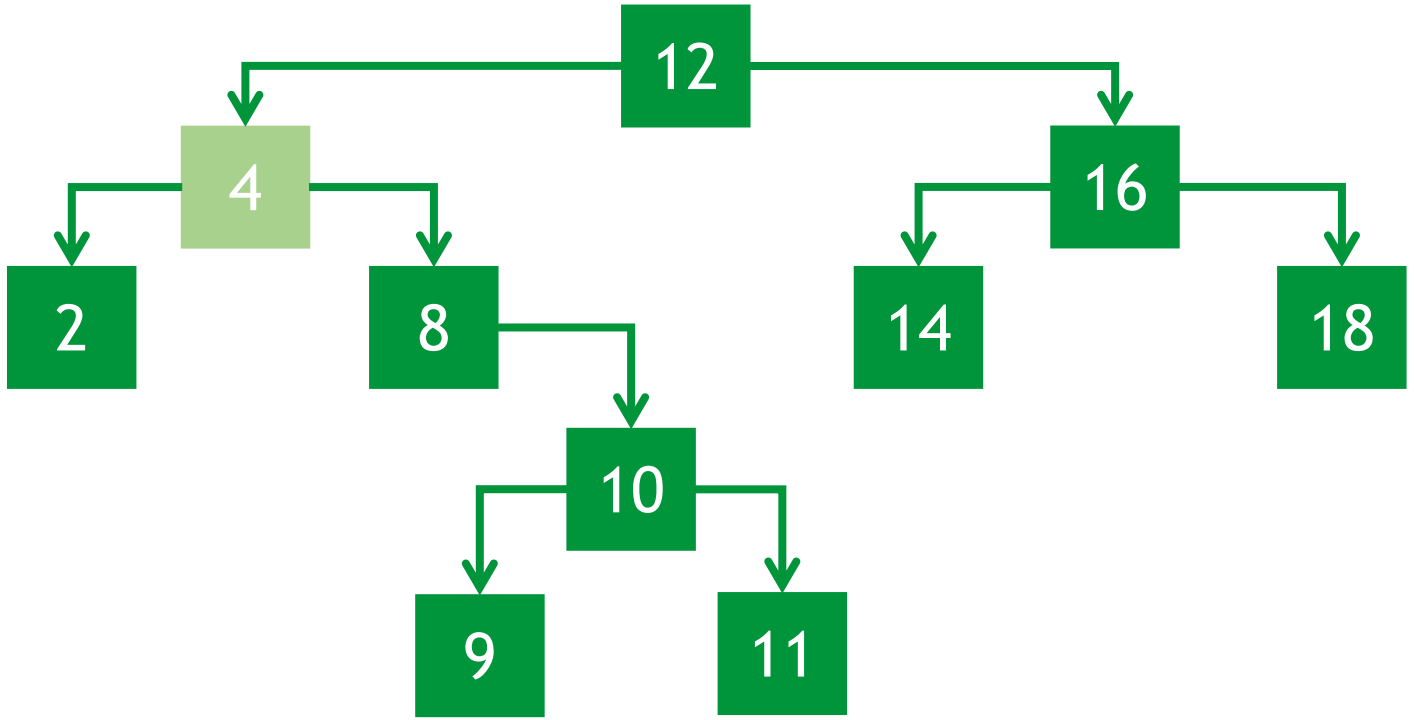
- Buscando o elemento de valor 10:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

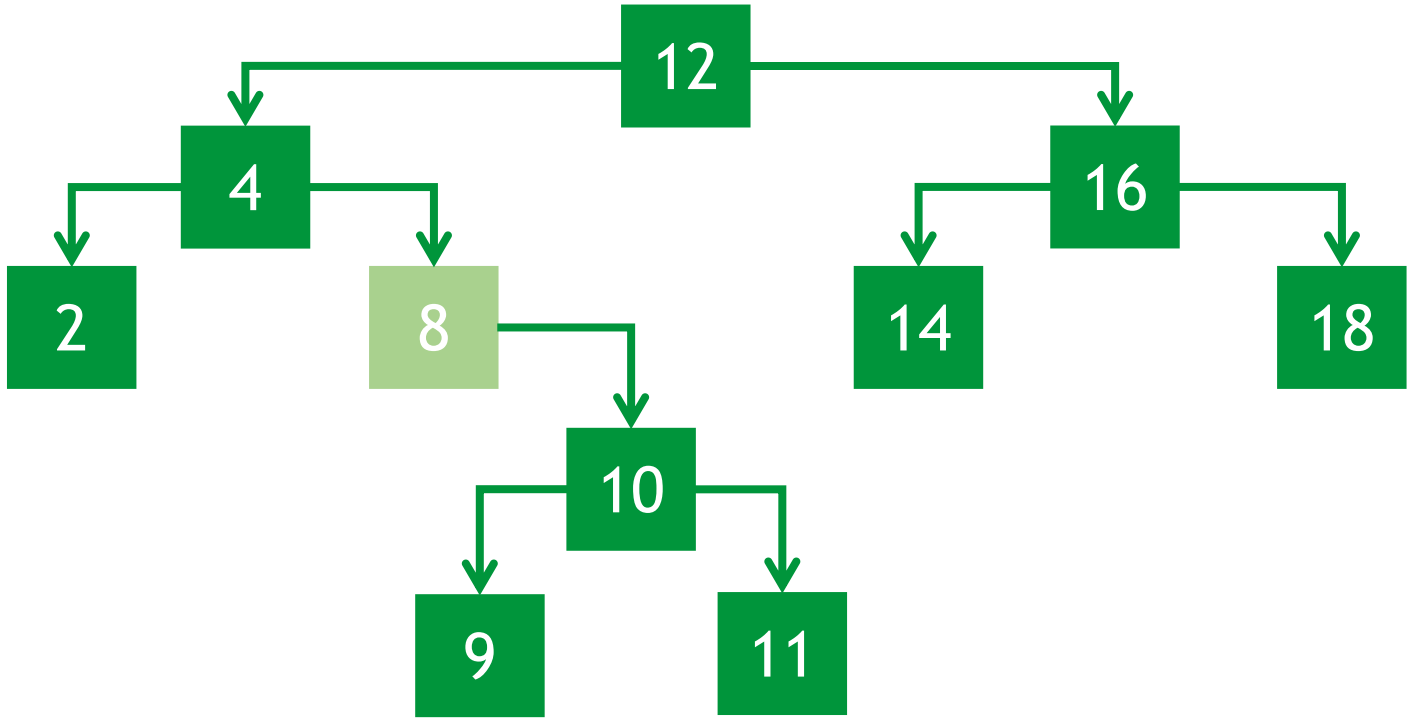
- Buscando o elemento de valor 10:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

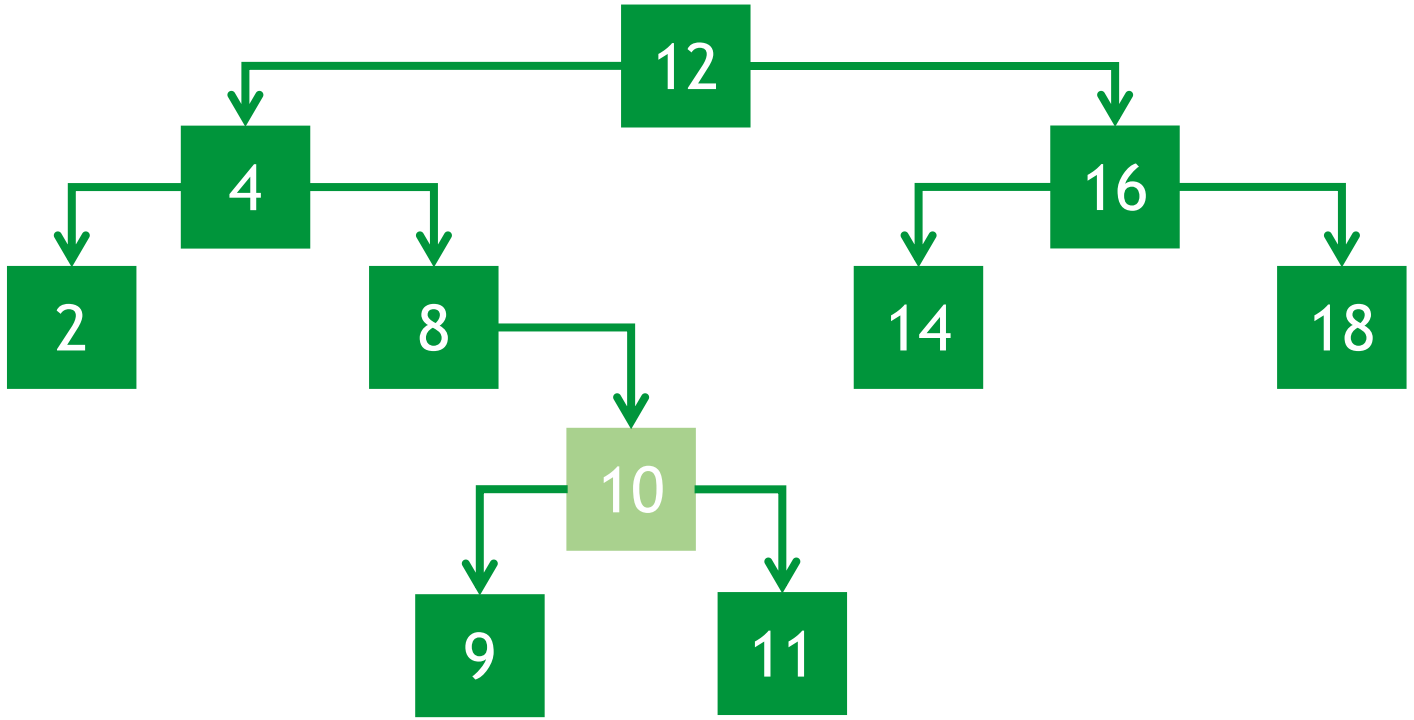
- Buscando o elemento de valor 10:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

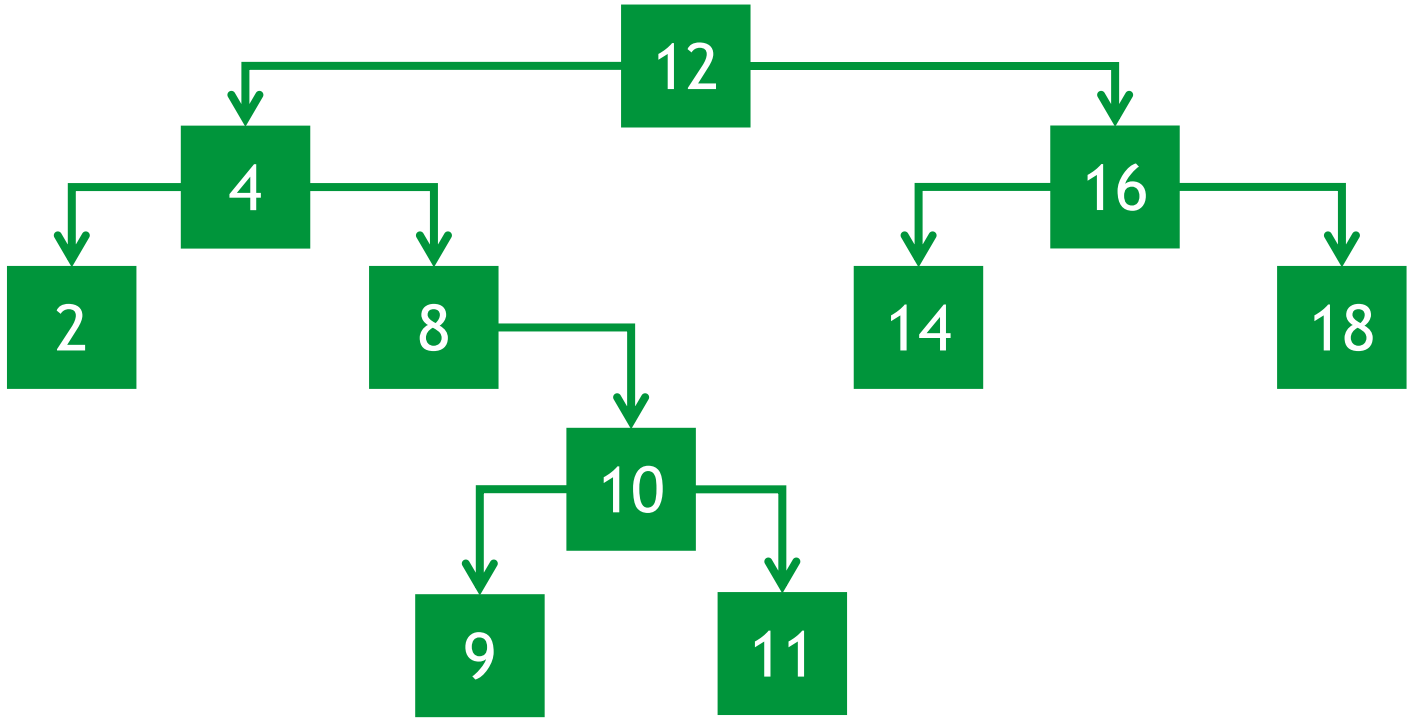
- Buscando o elemento de valor 10:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

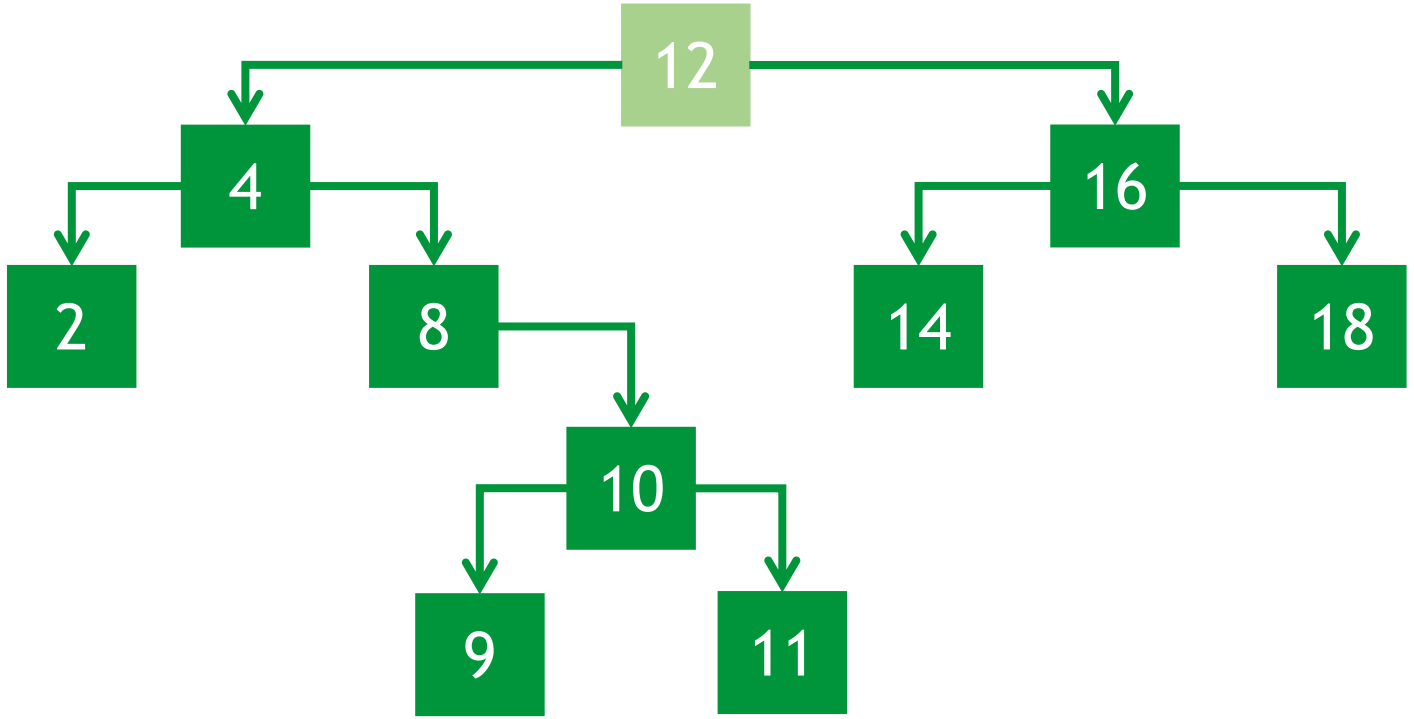
- Buscando o elemento de valor 7:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

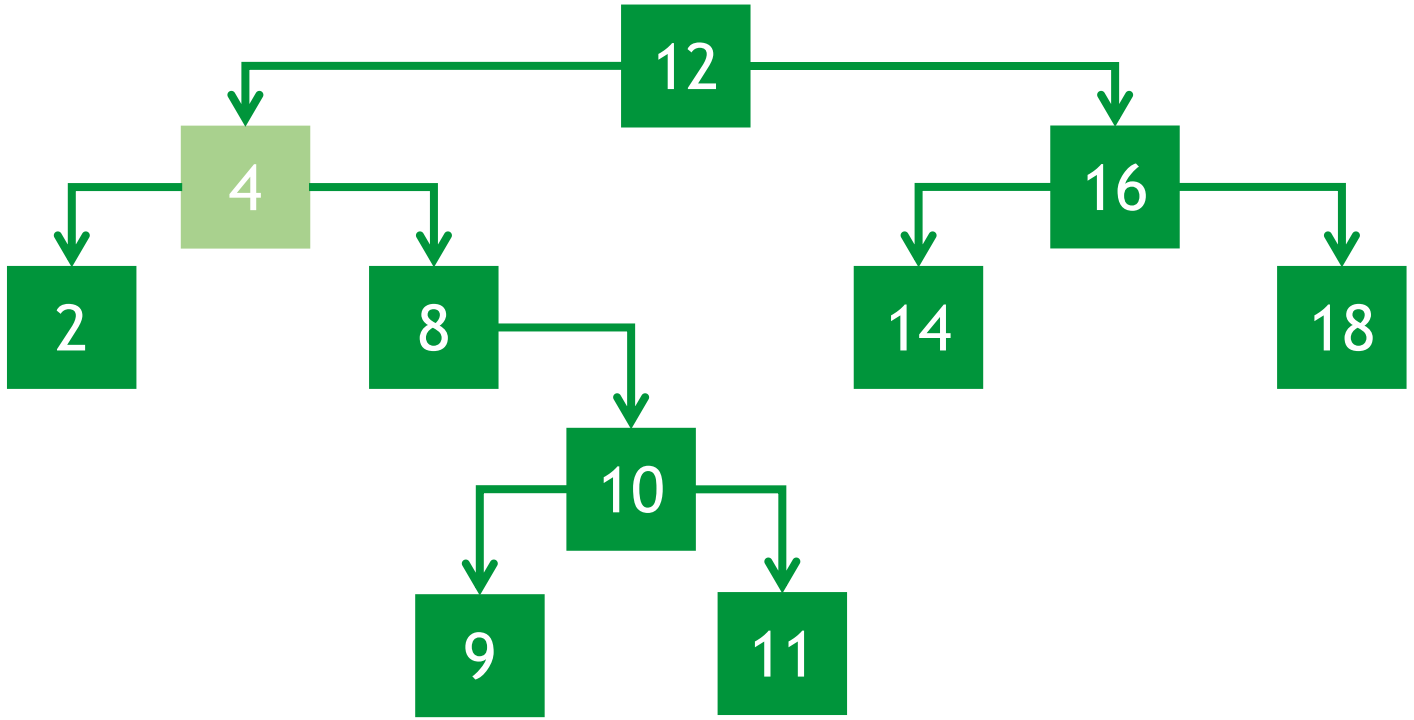
- Buscando o elemento de valor 7:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

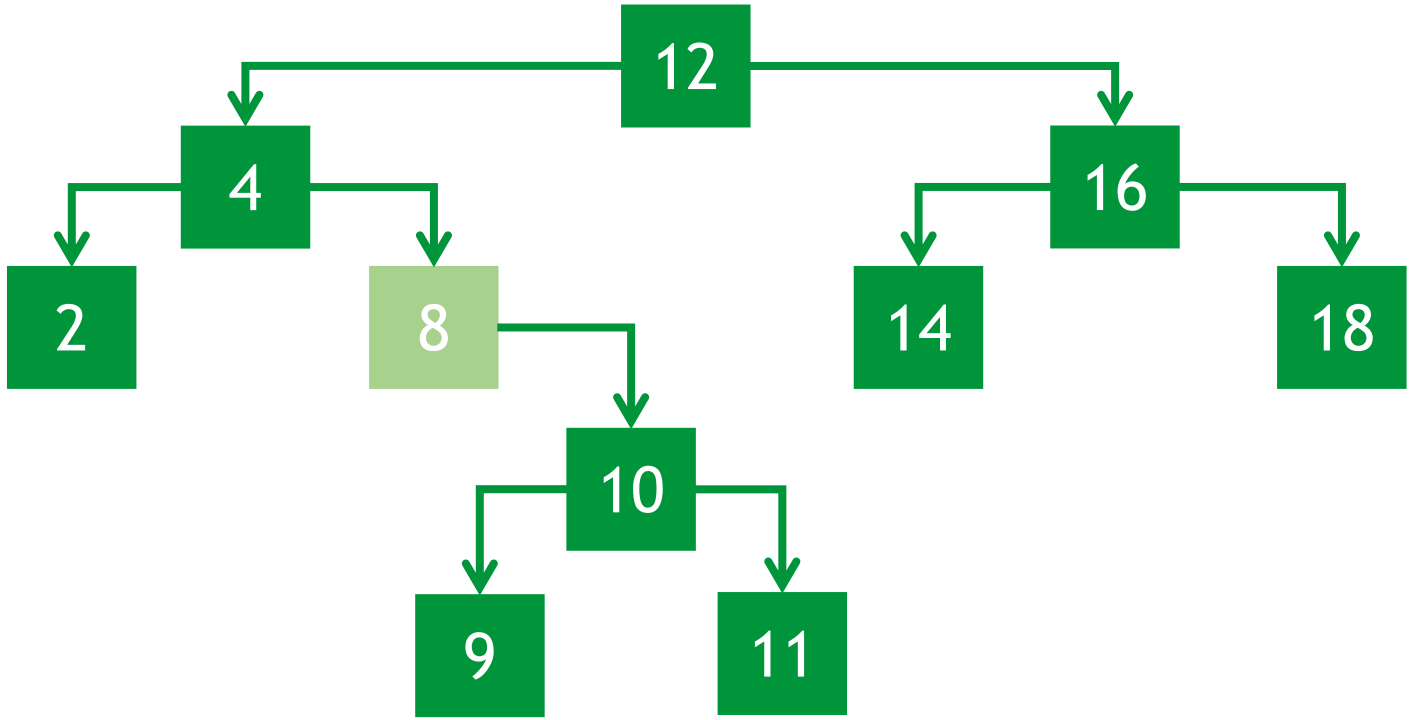
- Buscando o elemento de valor 7:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

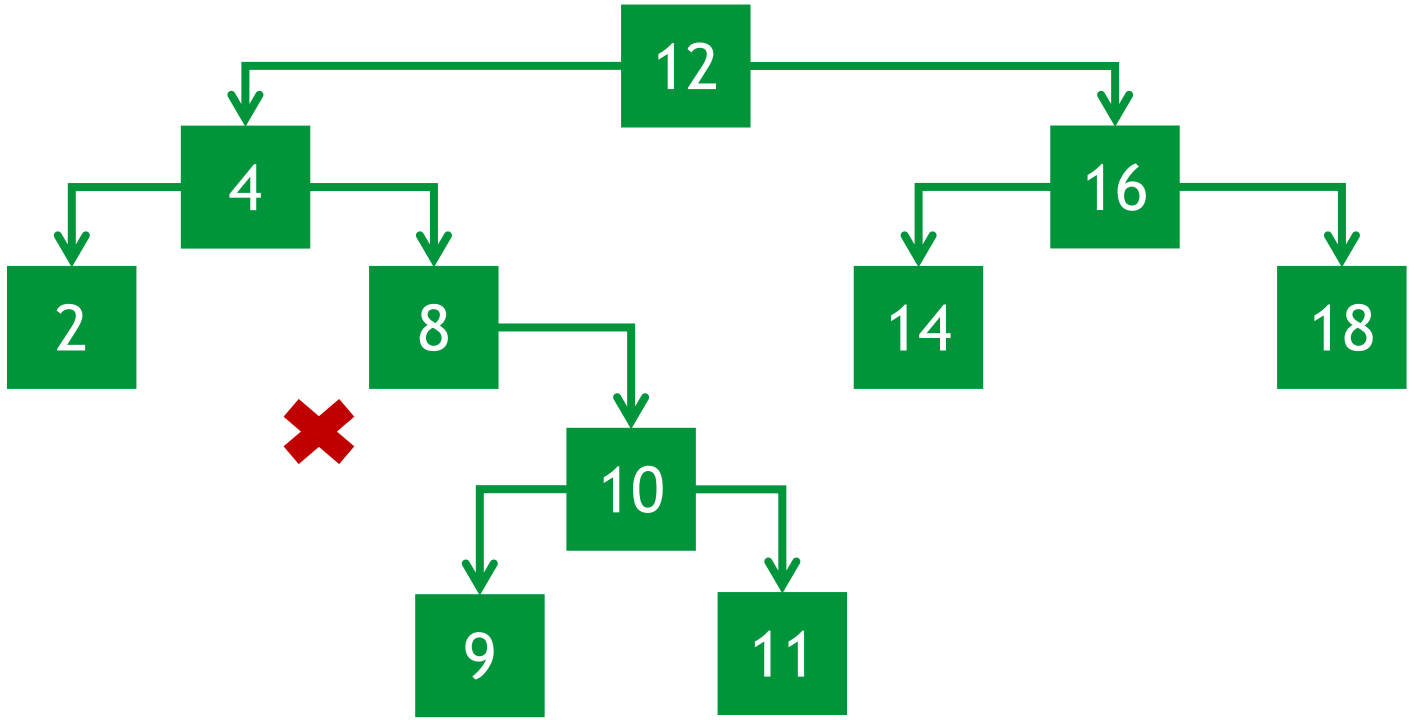
- Buscando o elemento de valor 7:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Buscando o elemento de valor 7:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Implementação da busca em uma BST em C:

```c
No *buscar(No *raiz, int valor) {

    if (raiz == NULL || raiz->valor == valor)
        return raiz;

    if (valor < raiz->valor)
        return buscar(raiz->esquerda, valor);

    return buscar(raiz->direita, valor);
}
```

Árvore Binária de Busca (BST)

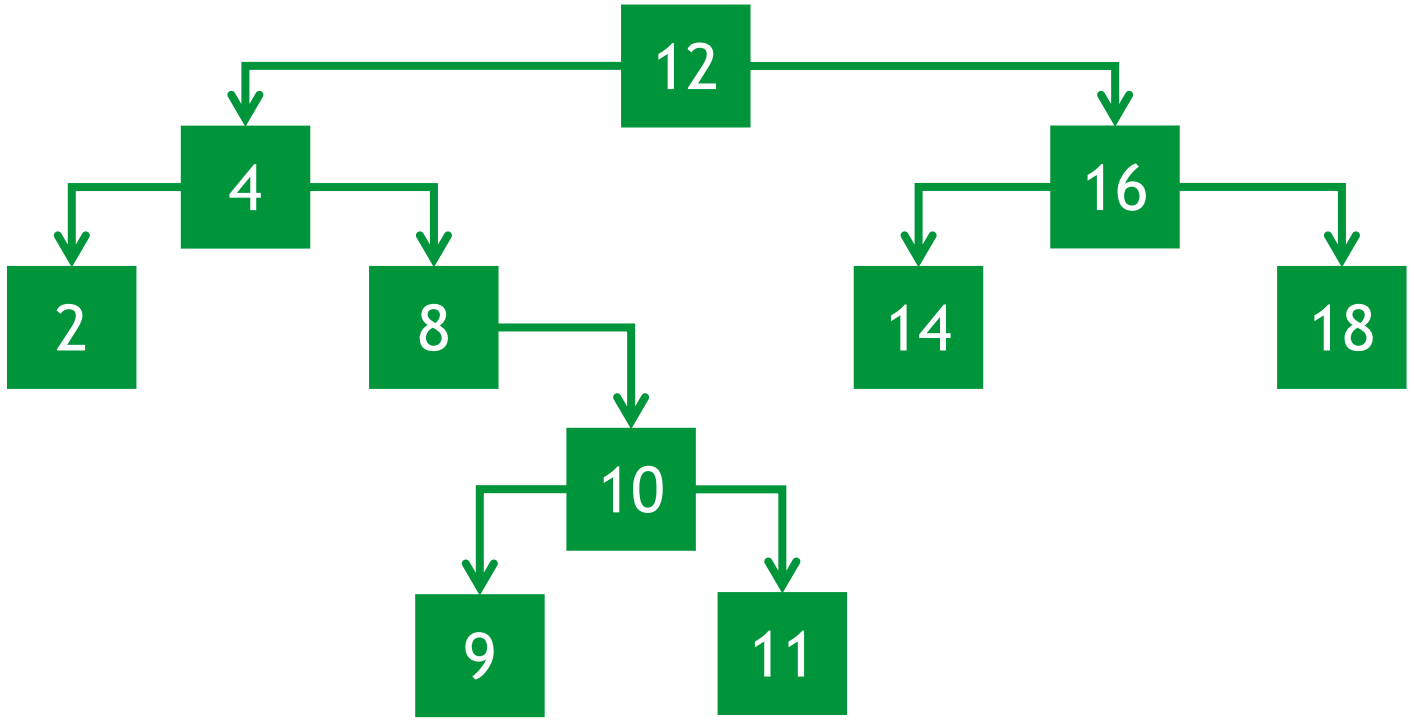
INSTITUTO FEDERAL
Pernambuco

- Inserção em uma Árvore Binária de Busca (BST):
  - A inserção segue a mesma lógica da busca.
  - Compara-se o valor a ser inserido com o valor do nó atual:
    - Se for menor, segue-se para a subárvore esquerda;
    - Se for maior, segue-se para a subárvore direita.
  - O processo continua até que seja encontrada uma referência nula (NULL).
  - A posição correspondente a essa referência nula torna-se o ponto de inserção do novo nó.

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

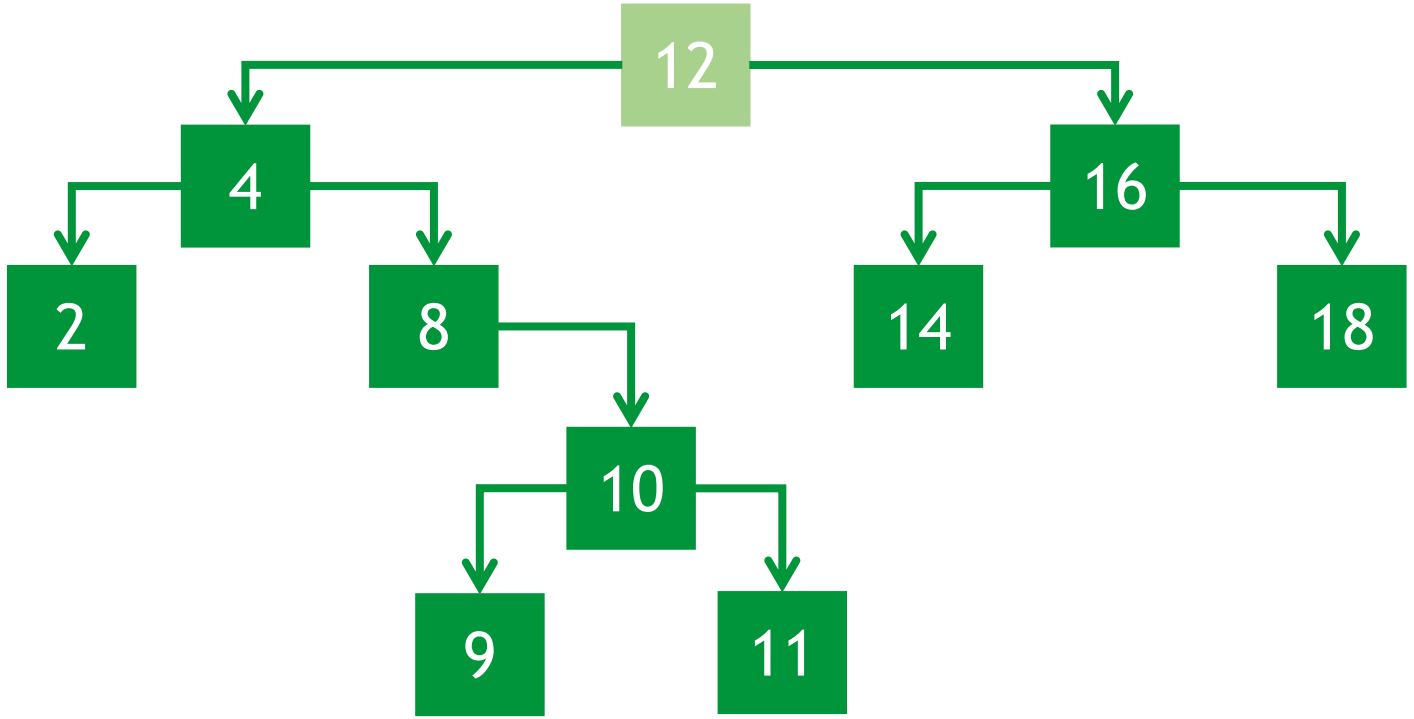
- Inserindo o valor 19:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

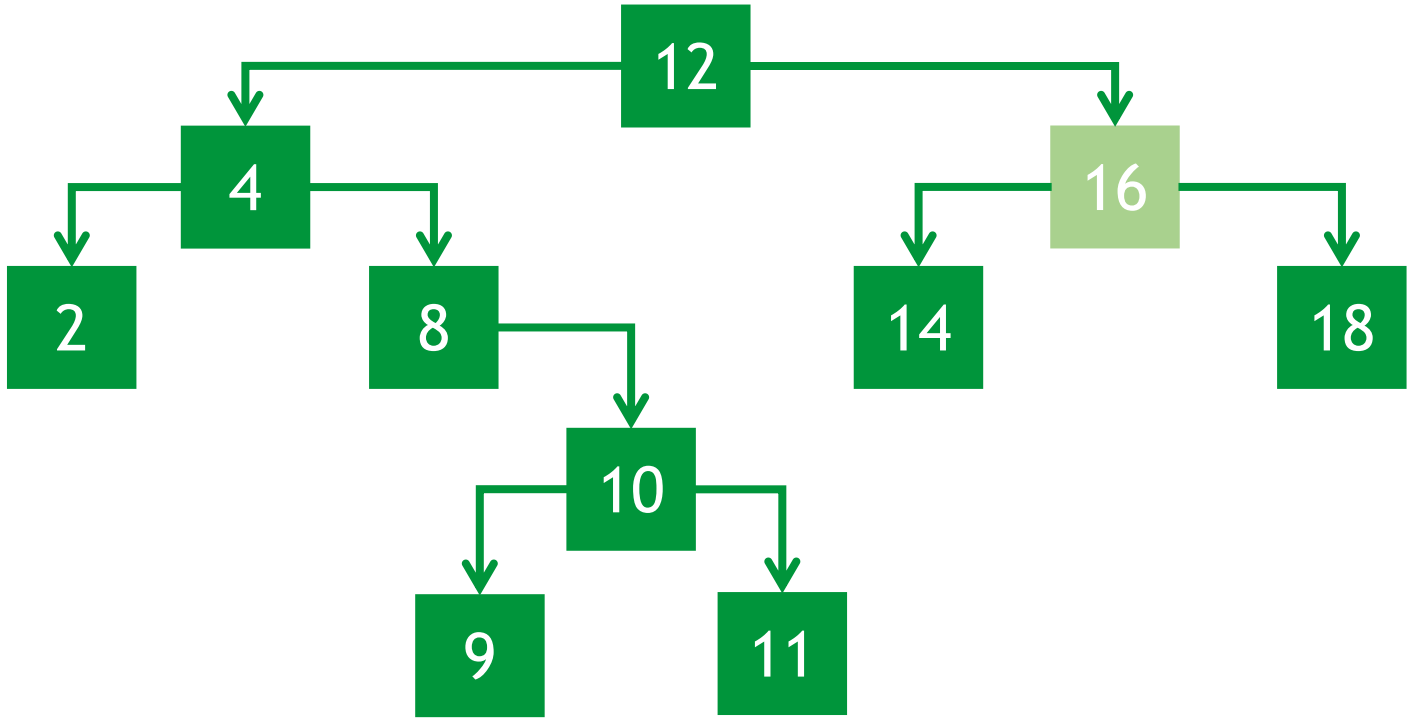
- Inserindo o valor 19:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

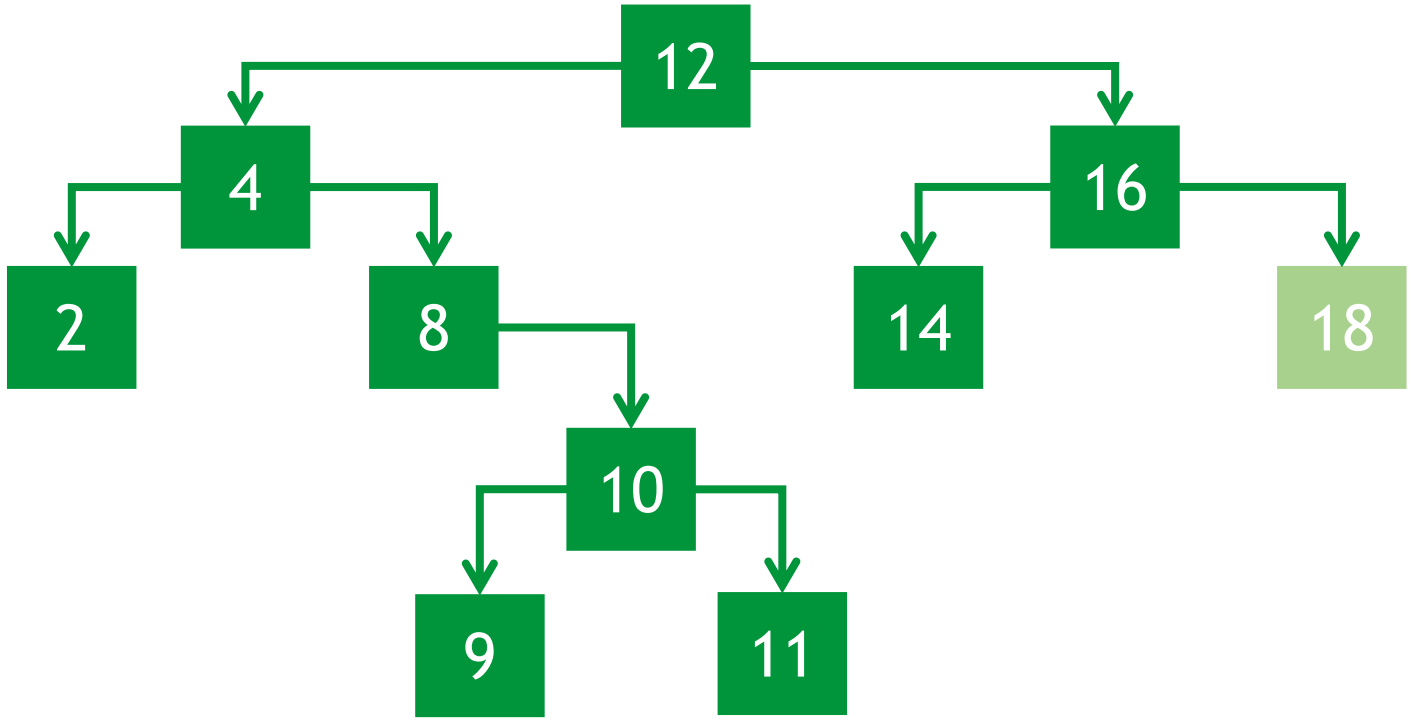
- Inserindo o valor 19:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

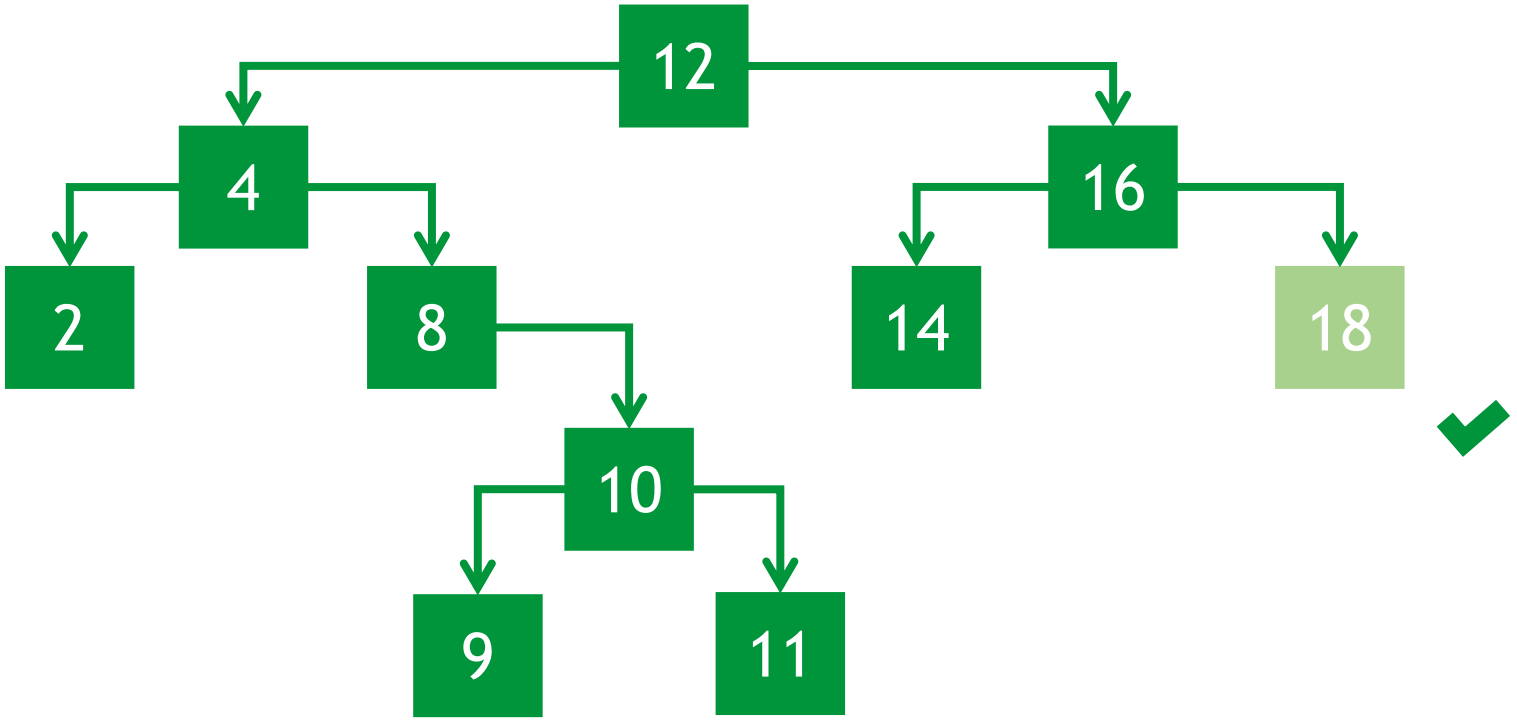
- Inserindo o valor 19:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

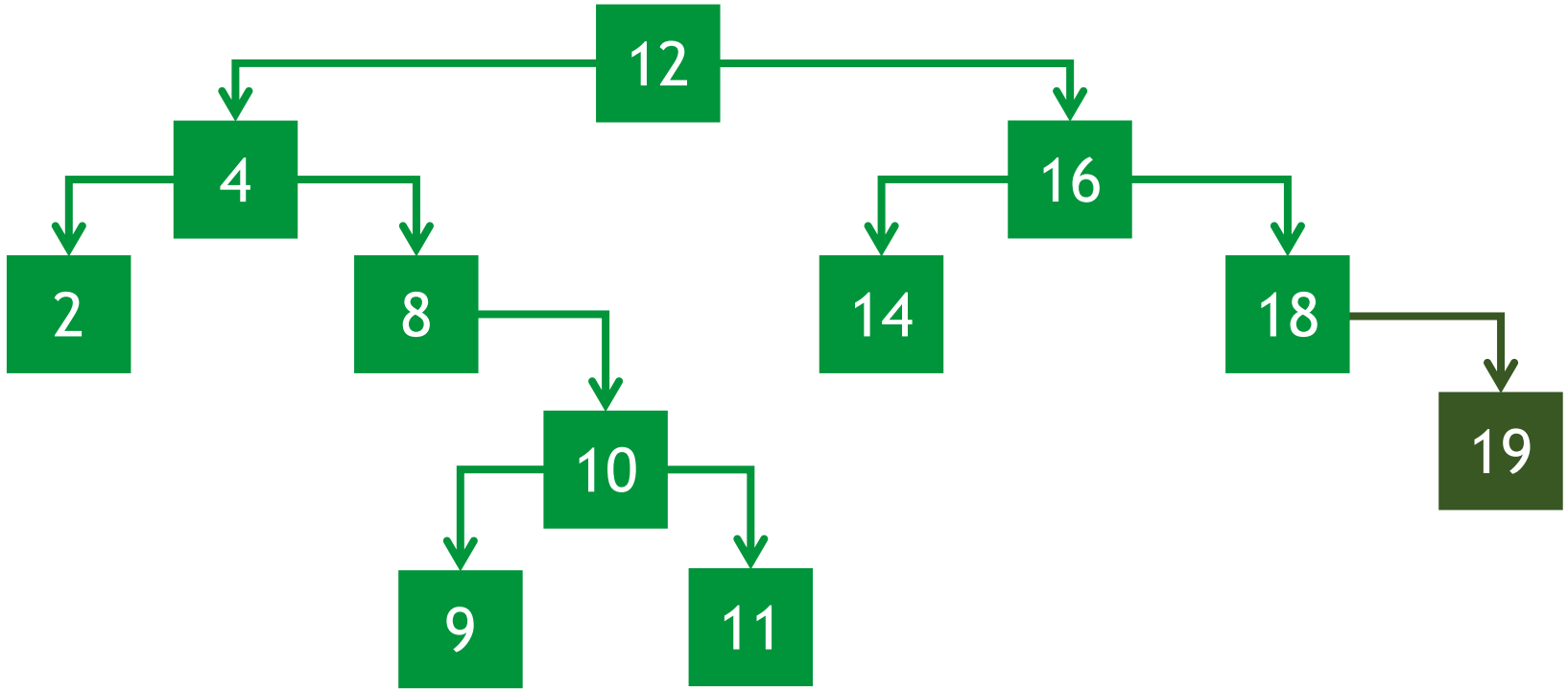
- Inserindo o valor 19:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Inserindo o valor 19:



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Implementação da inserção em uma BST em C:

```c
No *inserir(No *raiz, int valor) {

    if (raiz == NULL)
        return criarNo(valor);

    if (valor < raiz->valor)
        raiz->esquerda = inserir(raiz->esquerda, valor);
    else if (valor > raiz->valor)
        raiz->direita = inserir(raiz->direita, valor);

    return raiz;
}
```

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exercício rápido: Os seguintes valores são inseridos, nesta ordem, em uma Árvore Binária de Busca inicialmente vazia:

100, 200, 140, 50, 105, 70, 80, 30, 400, 350, 117, 42, 65

a) Desenhe a árvore resultante.

b) Qual é a altura da árvore?

c) Qual é a profundidade dos nós 117 e 350?

d) Quais são os nós folha da árvore?

Árvore Binária de Busca (BST)

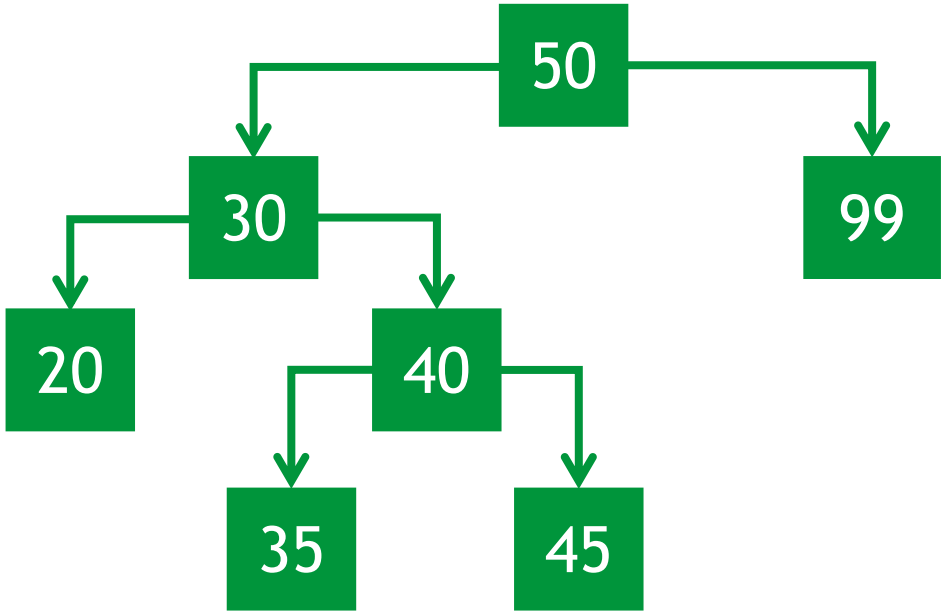
INSTITUTO FEDERAL
Pernambuco

- Remoção em uma Árvore Binária de Busca (BST):
  - Localize o nó a ser removido.
  - Após encontrá-lo, existem três casos possíveis:
    - Caso 1: o nó é uma folha → O nó é simplesmente removido da árvore.
    - Caso 2: o nó possui apenas um filho → O filho assume a posição do nó removido, mantendo a estrutura da árvore.
    - Caso 3: o nó possui dois filhos → O nó removido deve ser substituído por um elemento que preserve a propriedade da BST. Existem duas alternativas:
      - Substituí-lo pelo maior elemento da subárvore esquerda (predecessor em ordem) ou;
      - Substituí-lo pelo menor elemento da subárvore direita (sucessor em ordem).
      - Após a substituição, o nó utilizado deve ser removido de sua posição original.

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

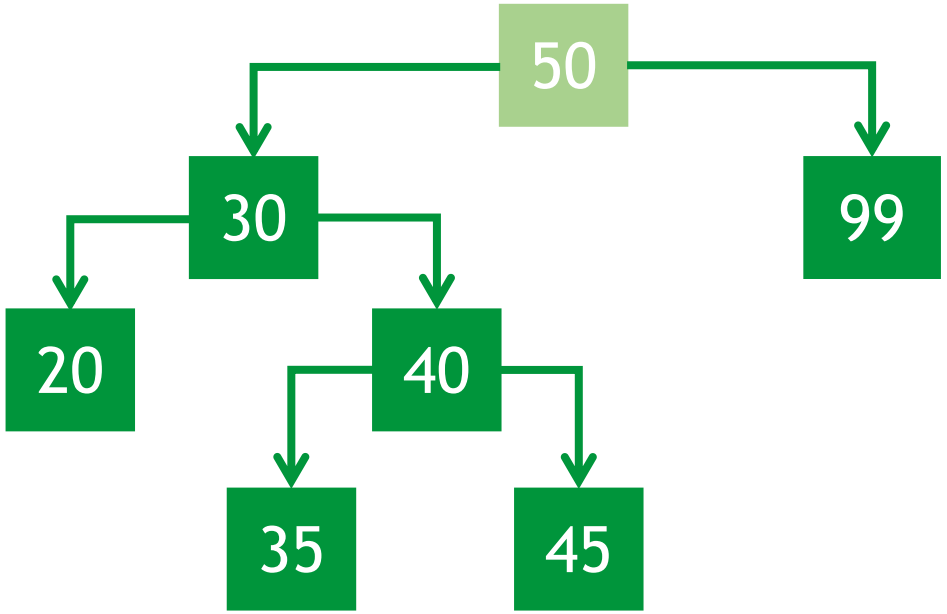
- Removendo o nó 20 (caso 1):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

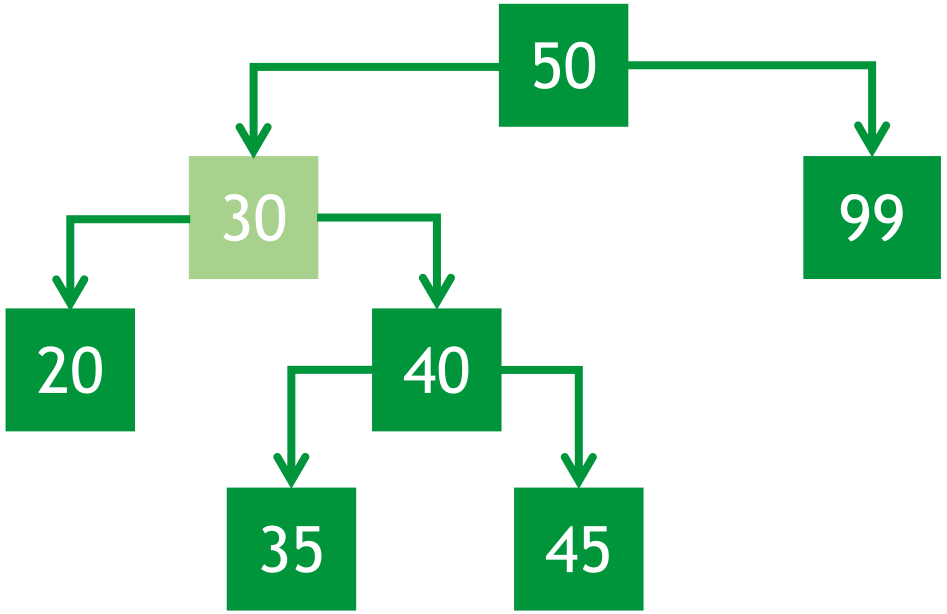
- Removendo o nó 20 (caso 1):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

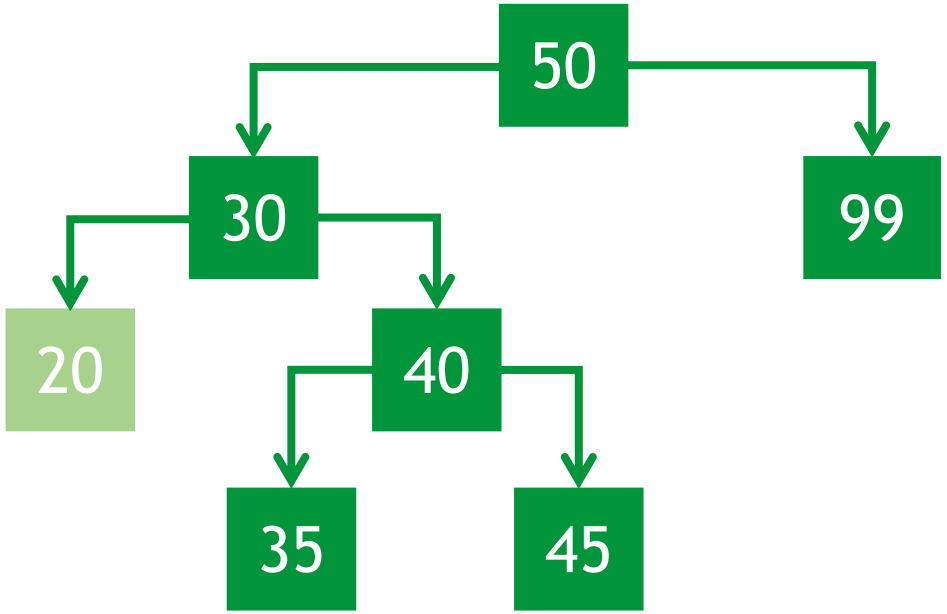
- Removendo o nó 20 (caso 1):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

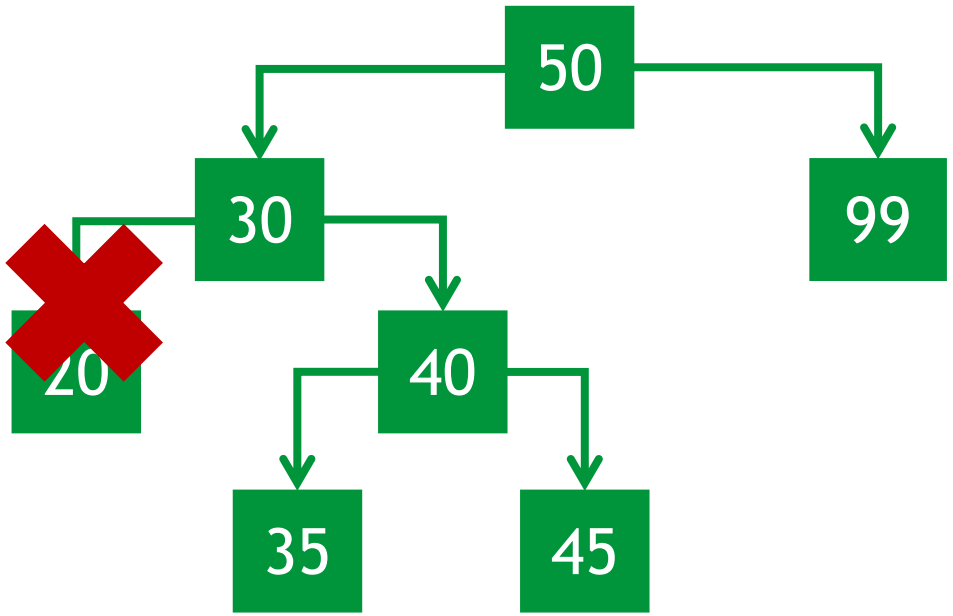
- Removendo o nó 20 (caso 1):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

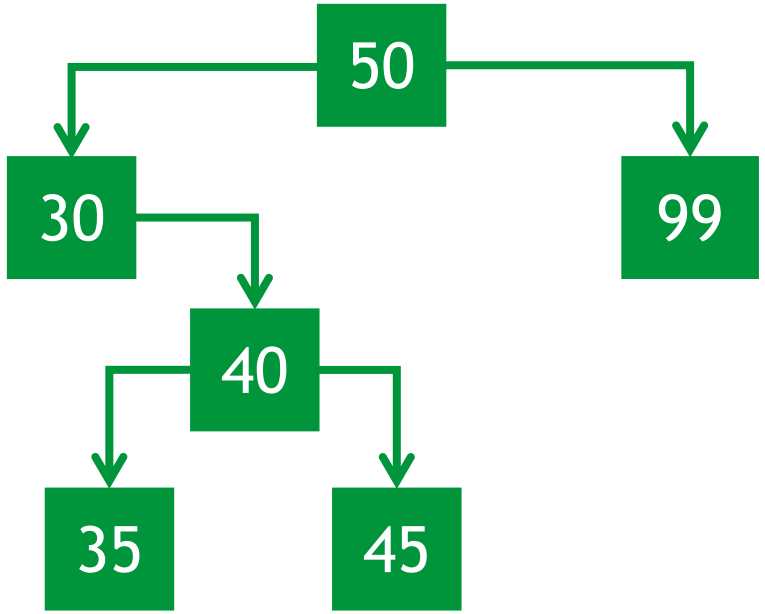
- Removendo o nó 20 (caso 1):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

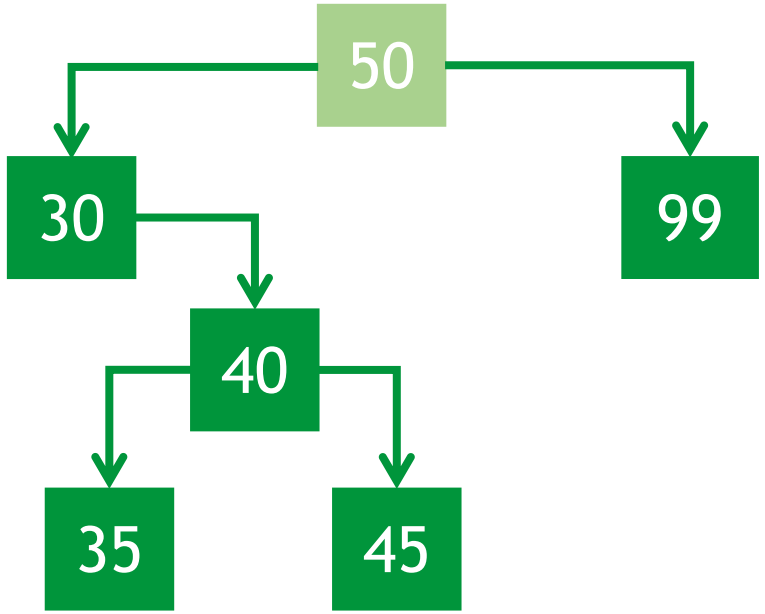
- Removendo o nó 30 (caso 2):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

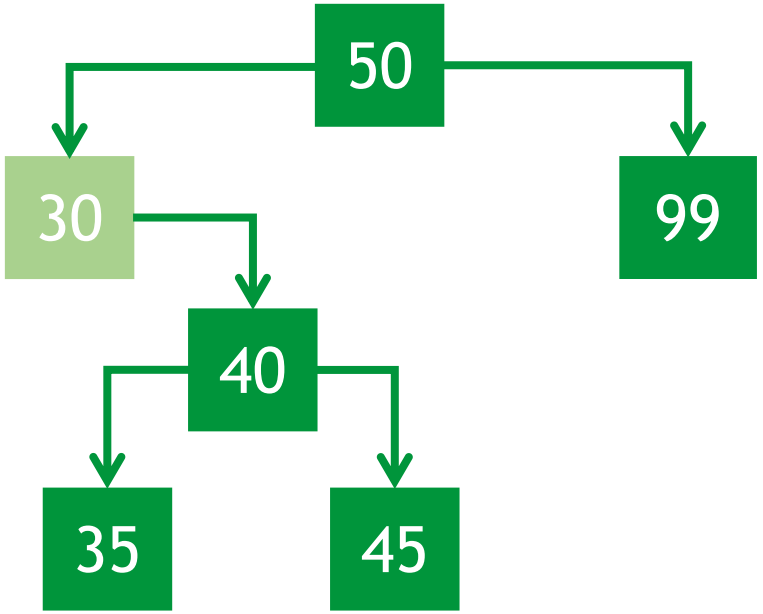
- Removendo o nó 30 (caso 2):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

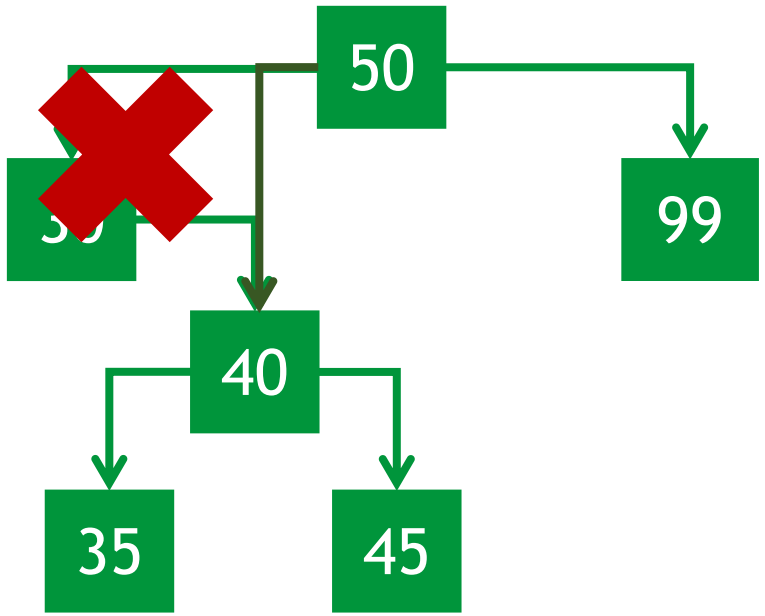
- Removendo o nó 30 (caso 2):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

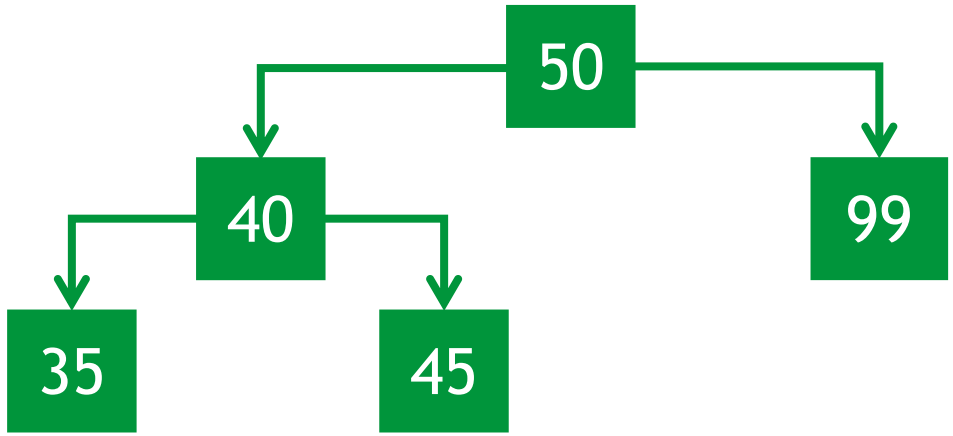
- Removendo o nó 30 (caso 2):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

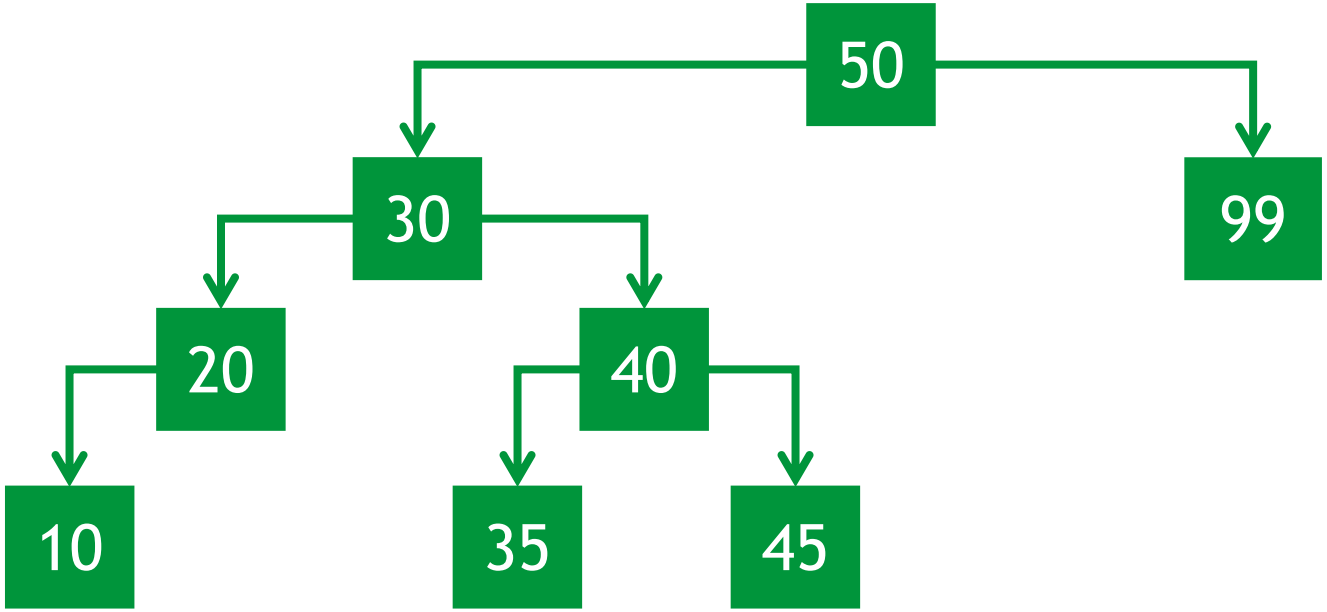
- Removendo o nó 30 (caso 2):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

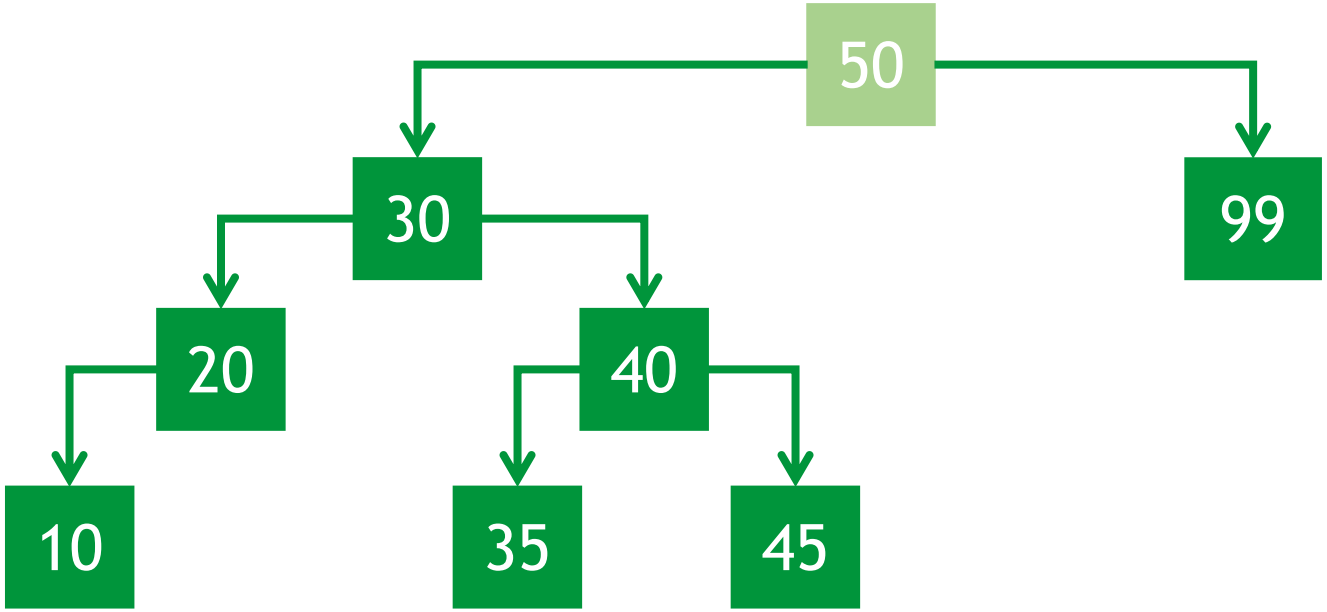
- Removendo o nó 30 (caso 3), substituindo pelo maior elemento da subárvore esquerda (predecessor em ordem) ou :



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

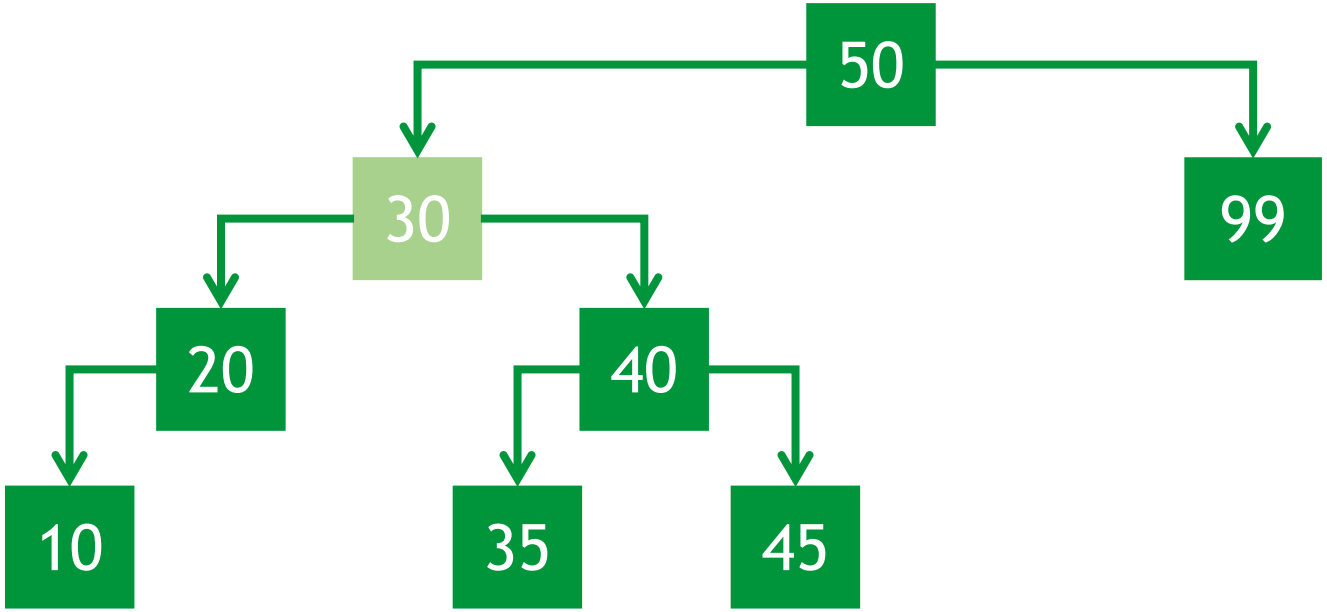
- Removendo o nó 30 (caso 3), substituindo pelo maior elemento da subárvore esquerda (predecessor em ordem) ou :



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

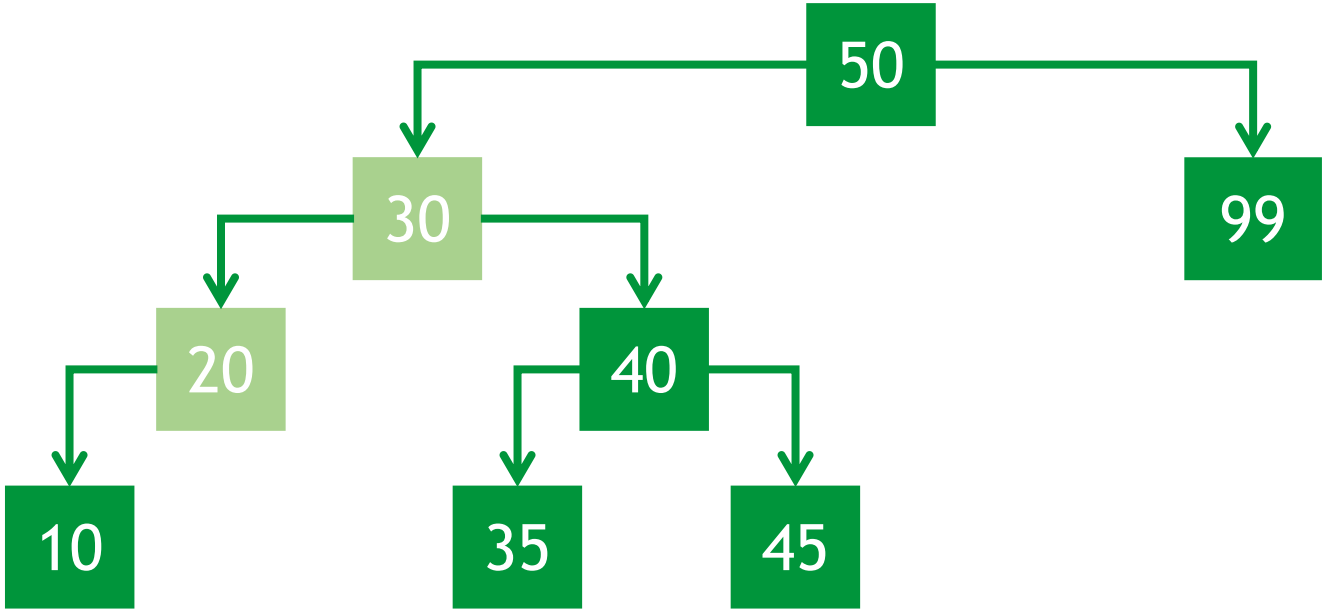
- Removendo o nó 30 (caso 3), substituindo pelo maior elemento da subárvore esquerda (predecessor em ordem) ou :



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

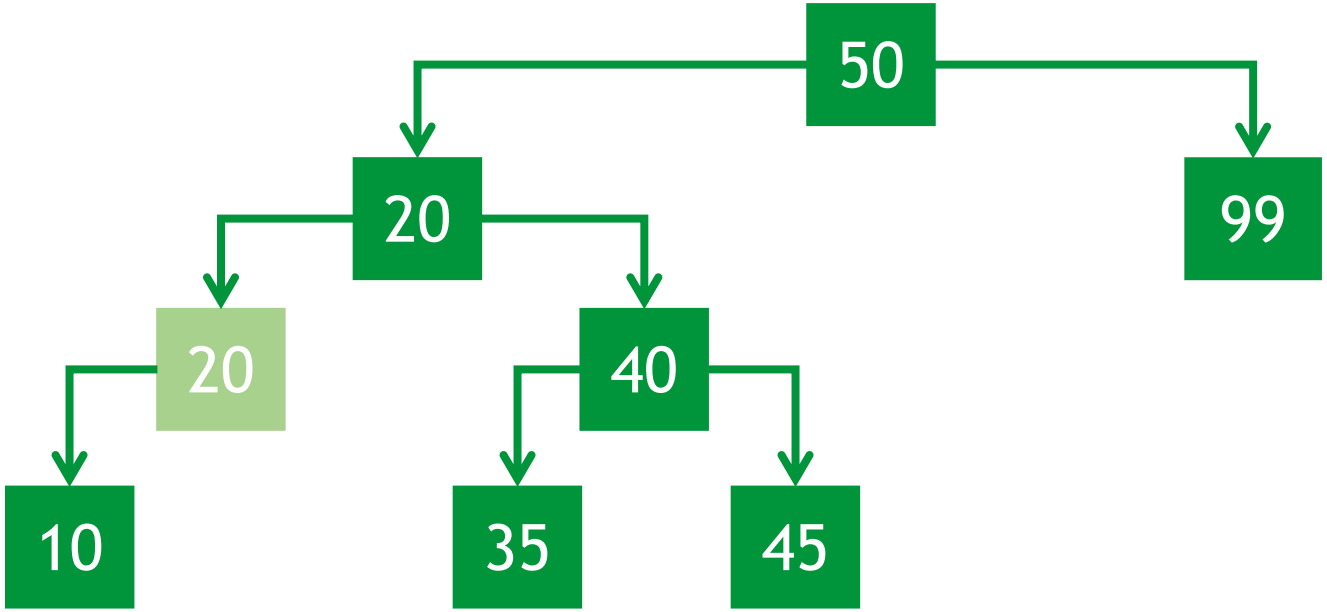
- Removendo o nó 30 (caso 3), substituindo pelo maior elemento da subárvore esquerda (predecessor em ordem) ou :



Árvore Binária de Busca (BST)

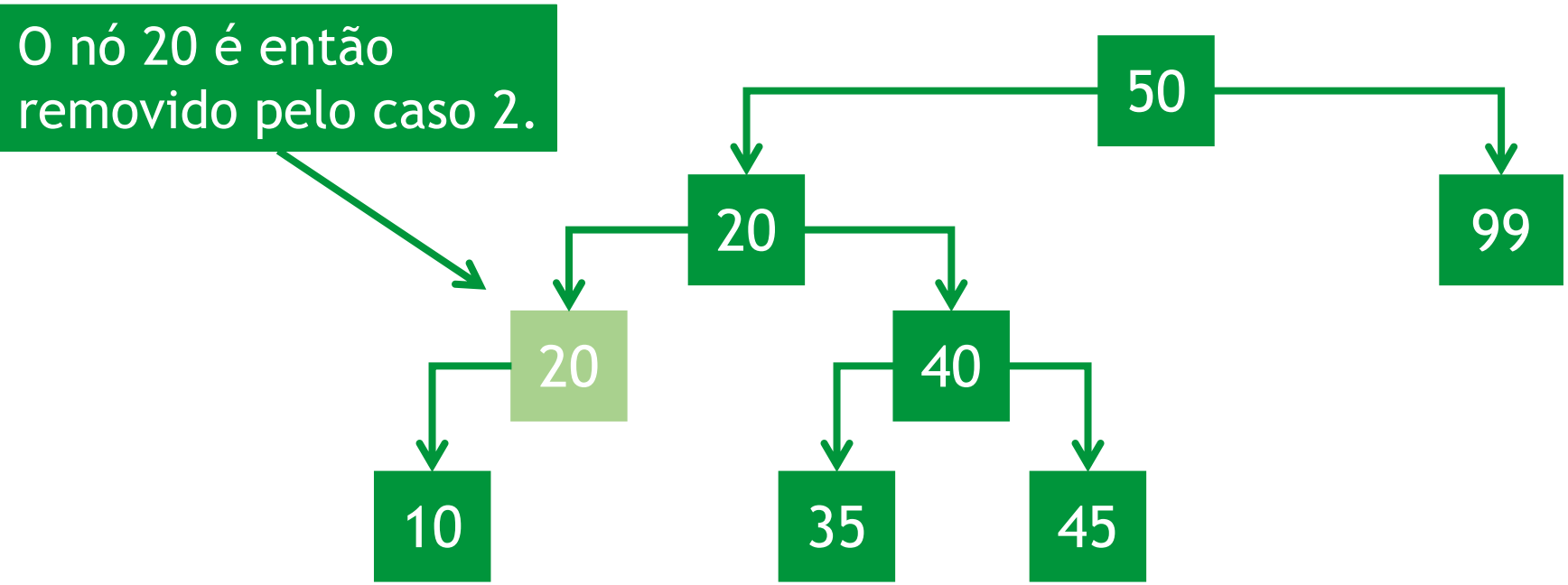
INSTITUTO FEDERAL
Pernambuco

- Removendo o nó 30 (caso 3), substituindo pelo maior elemento da subárvore esquerda (predecessor em ordem) ou :



Árvore Binária de Busca (BST)

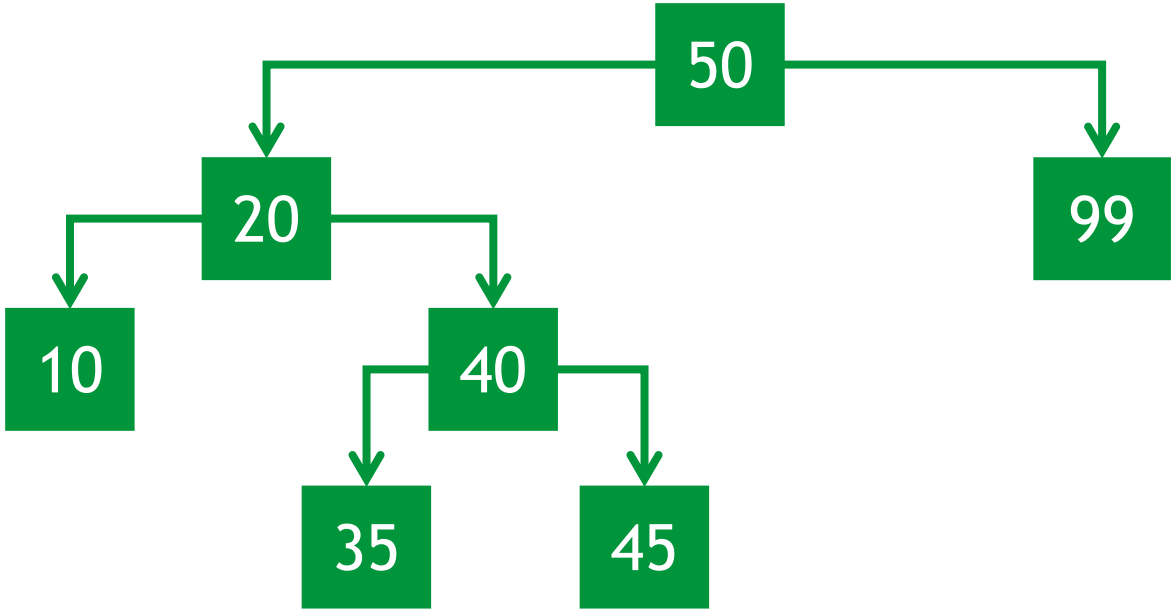
- Removendo o nó 30 (caso 3), substituindo pelo maior elemento da subárvore esquerda (predecessor em ordem) ou :



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

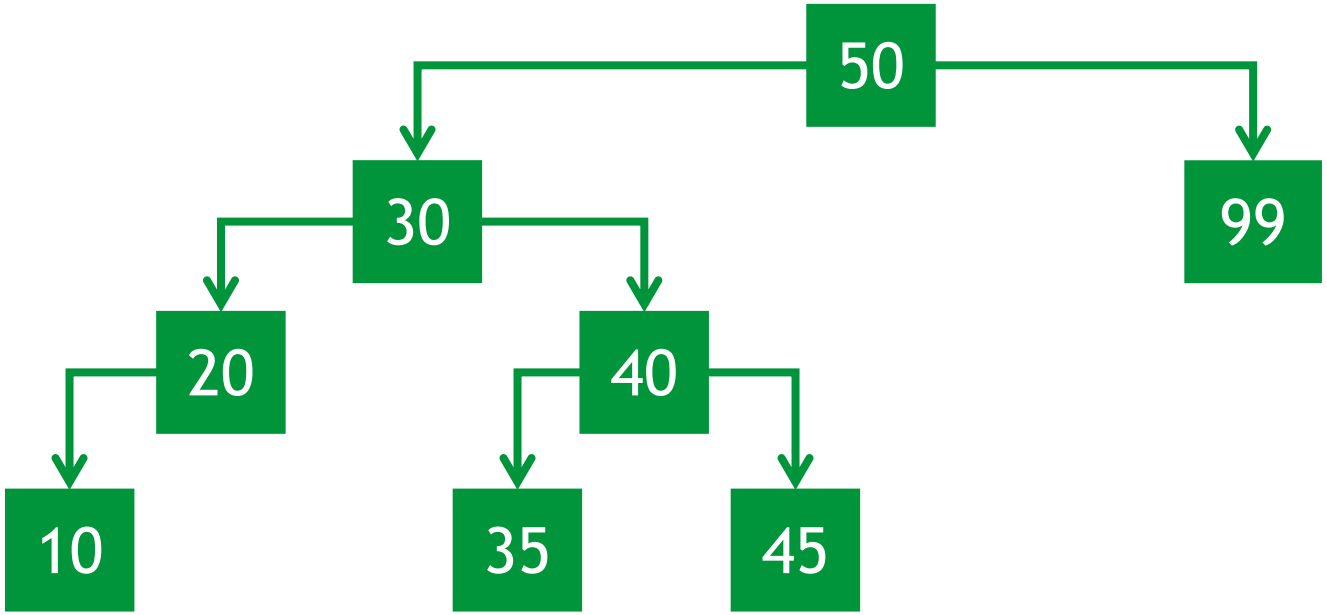
- Removendo o nó 30 (caso 3), substituindo pelo maior elemento da subárvore esquerda (predecessor em ordem) ou :



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

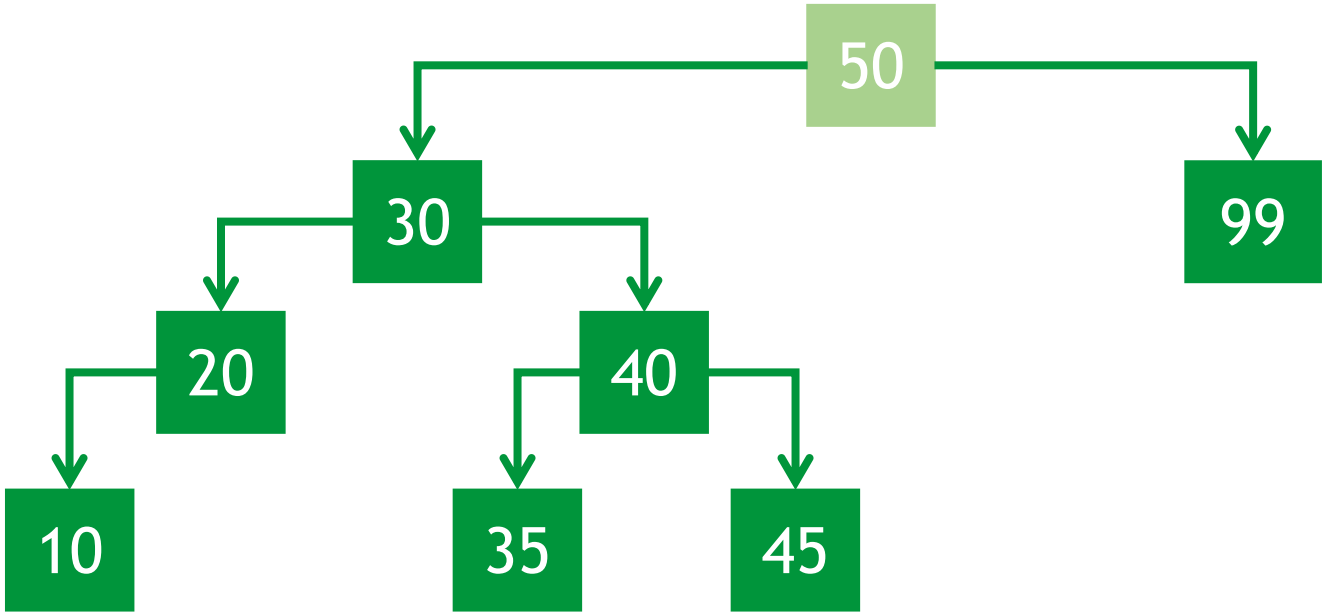
- Removendo o nó 30 (caso 3), substituindo pelo menor elemento da subárvore direita (sucessor em ordem):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

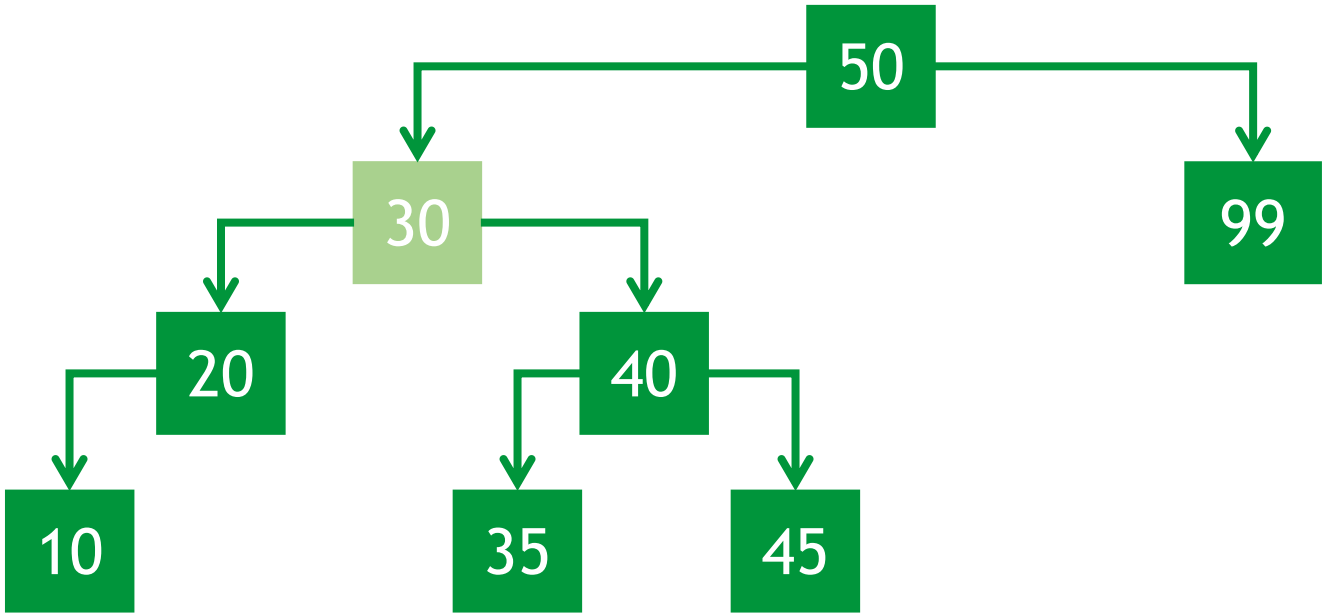
- Removendo o nó 30 (caso 3), substituindo pelo menor elemento da subárvore direita (sucessor em ordem):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

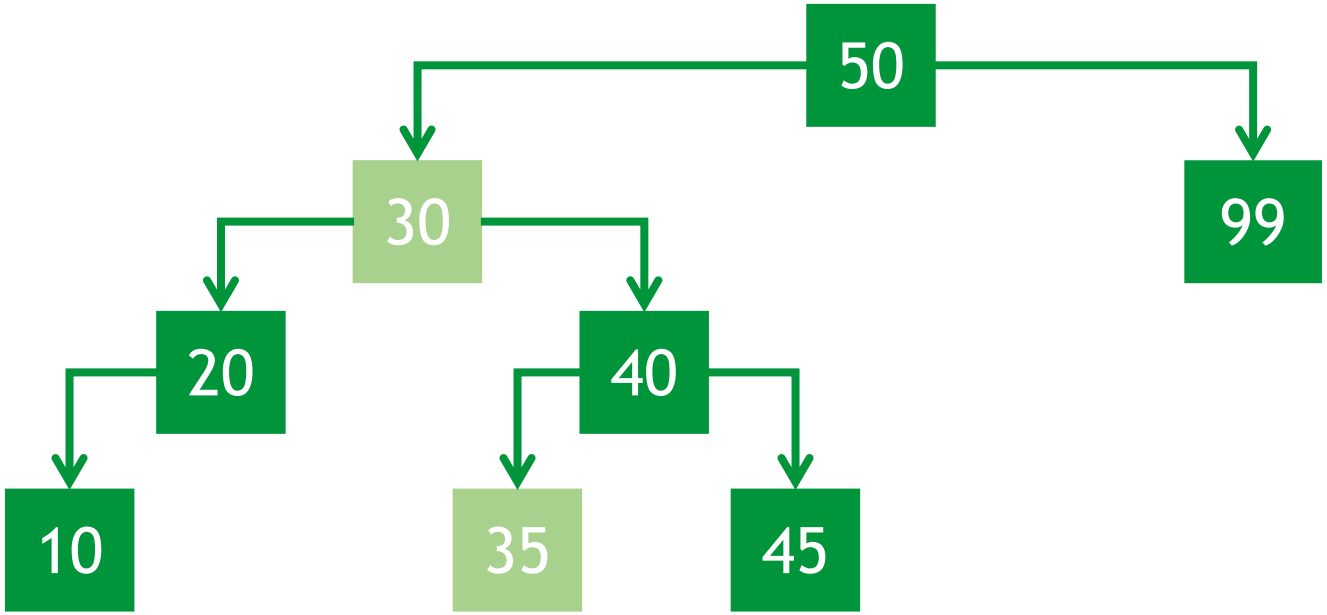
- Removendo o nó 30, substituindo pelo menor elemento da subárvore direita (sucessor em ordem):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

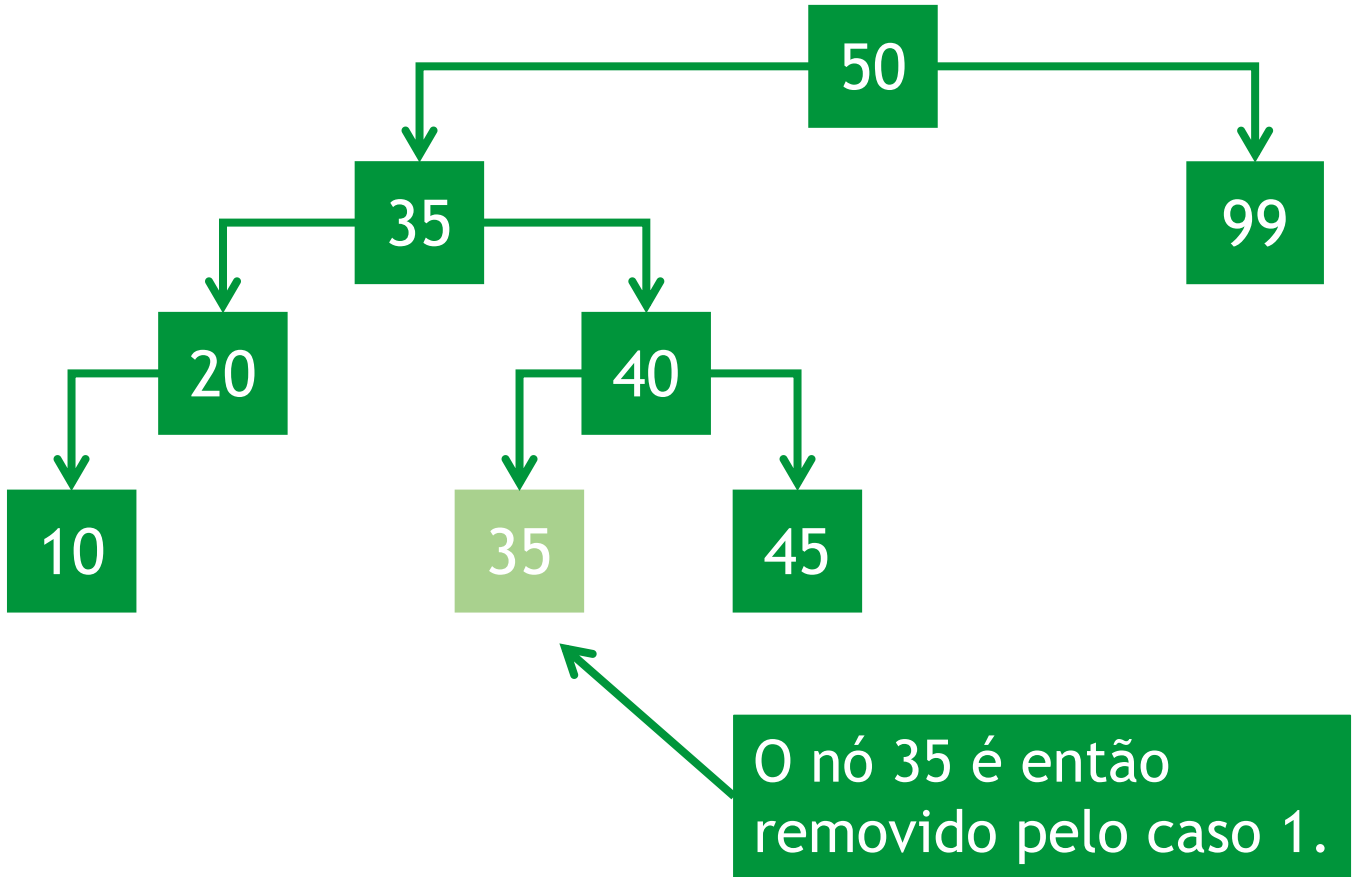
- Removendo o nó 30, substituindo pelo menor elemento da subárvore direita (sucessor em ordem):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

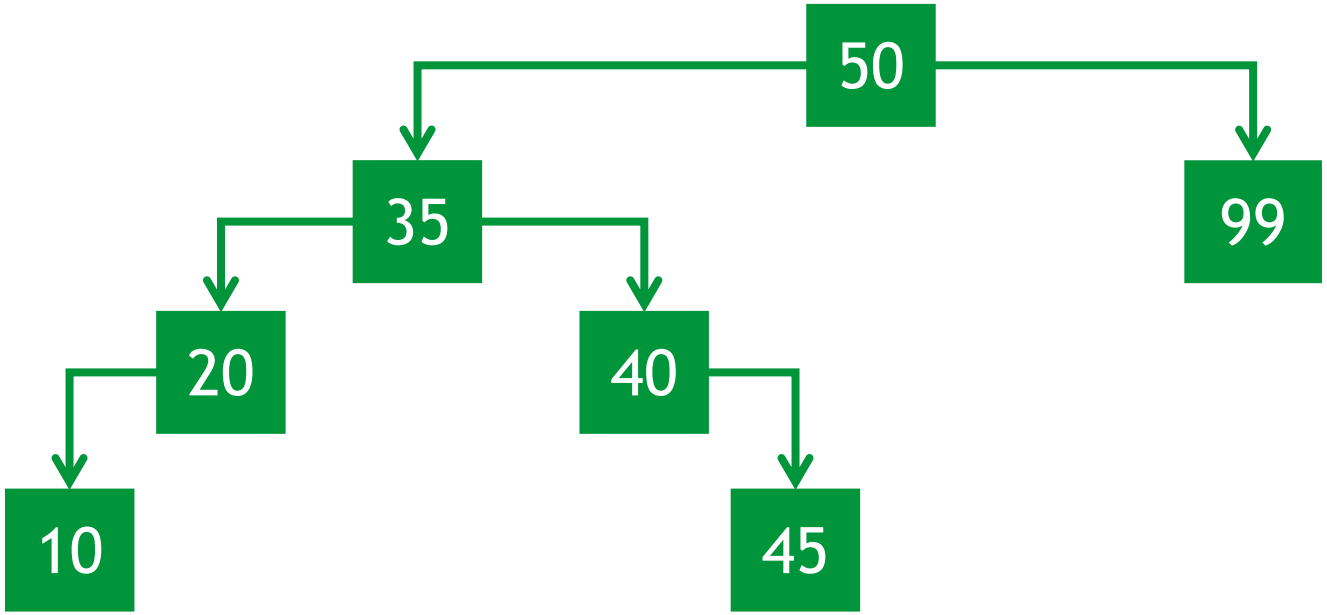
- Removendo o nó 30, substituindo pelo menor elemento da subárvore direita (sucessor em ordem):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Removendo o nó 30, substituindo pelo menor elemento da subárvore direita (sucessor em ordem):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exercício rápido: Os seguintes valores são inseridos, nesta ordem, em uma Árvore Binária de Busca inicialmente vazia:

100, 200, 140, 50, 105, 70, 80, 30, 400, 350, 117, 42, 65

Remova os nós na seguinte ordem: 200, 50, 30, 117 e 65.

Represente a árvore após cada remoção e a árvore final obtida.

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Implementação da remoção em uma BST em C:

```c
No *remover(No *raiz, int valor) {
    if (raiz == NULL)
        return NULL;
    if (valor < raiz->valor)
        raiz->esquerda = remover(raiz->esquerda, valor);
    else if (valor > raiz->valor)
        raiz->direita = remover(raiz->direita, valor);
    else {
        // Caso 1: folha
        if (raiz->esquerda == NULL && raiz->direita == NULL) {
            free(raiz);
            return NULL;
        }
        // Caso 2: apenas filho à direita
        if (raiz->esquerda == NULL) {
            No *temp = raiz->direita;
            free(raiz);
            return temp;
        }
        // Caso 2: apenas filho à esquerda
        if (raiz->direita == NULL) {
            No *temp = raiz->esquerda;
            free(raiz);
            return temp;
        }
        // Caso 3: dois filhos
        No *sucessor = menorValor(raiz->direita);
        raiz->valor = sucessor->valor;
        raiz->direita = remover(raiz->direita, sucessor->valor);
    }
    return raiz;
}
```

Árvore Binária de Busca (BST)


```c
No *remover(No *raiz, int valor) {
    if (raiz == NULL)
        return NULL;
    if (valor < raiz->valor)
        raiz->esquerda = remover(raiz->esquerda, valor);
    else if (valor > raiz->valor)
        raiz->direita = remover(raiz->direita, valor);
    else {
        // Caso 1: folha
        if (raiz->esquerda == NULL && raiz->direita == NULL) {
            free(raiz);
            return NULL;
        }
```

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

```c
// Caso 2: apenas filho à direita
if (raiz->esquerda == NULL) {
    No *temp = raiz->direita;
    free(raiz);
    return temp;
}
// Caso 2: apenas filho à esquerda
if (raiz->direita == NULL) {
    No *temp = raiz->esquerda;
    free(raiz);
    return temp;
}
```

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

```
        // Caso 3: dois filhos
        No *sucessor = menorValor(raiz->direita);
        raiz->valor = sucessor->valor;
        raiz->direita = remover(raiz->direita, sucessor->valor);
    }
    return raiz;
}
```

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

```c
No *menorValor(No *raiz) {

    while (raiz->esquerda != NULL)
        raiz = raiz->esquerda;

    return raiz;
}
```

Árvore Binária de Busca (BST)

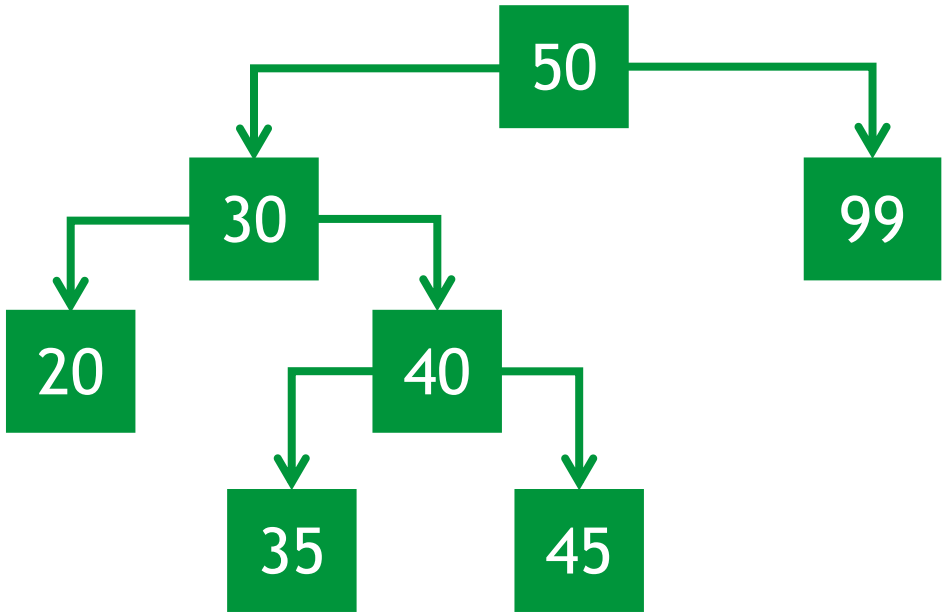
INSTITUTO FEDERAL
Pernambuco

- Travessia (ou percurso) é o processo de visitar todos os nós de uma árvore seguindo uma ordem específica.
- Existem três formas clássicas de percorrer uma árvore binária:
  - Pré-ordem (R E D): visita primeiro a raiz (R), depois a subárvore esquerda (E) e, por fim, a subárvore direita (D).
  - Em ordem ou simétrica (E R D): visita primeiro a subárvore esquerda (E), depois a raiz (R) e, por fim, a subárvore direita (D).
  - Pós-ordem (E D R): visita primeiro a subárvore esquerda (E), depois a subárvore direita (D) e, por fim, a raiz (R).

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

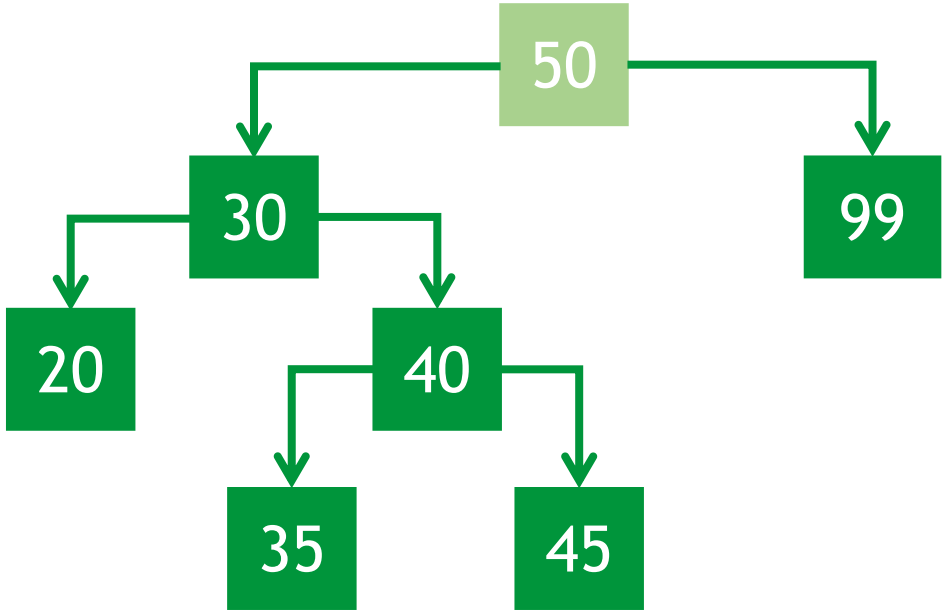
- Exemplo de travessia em Pré-ordem (R E D):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

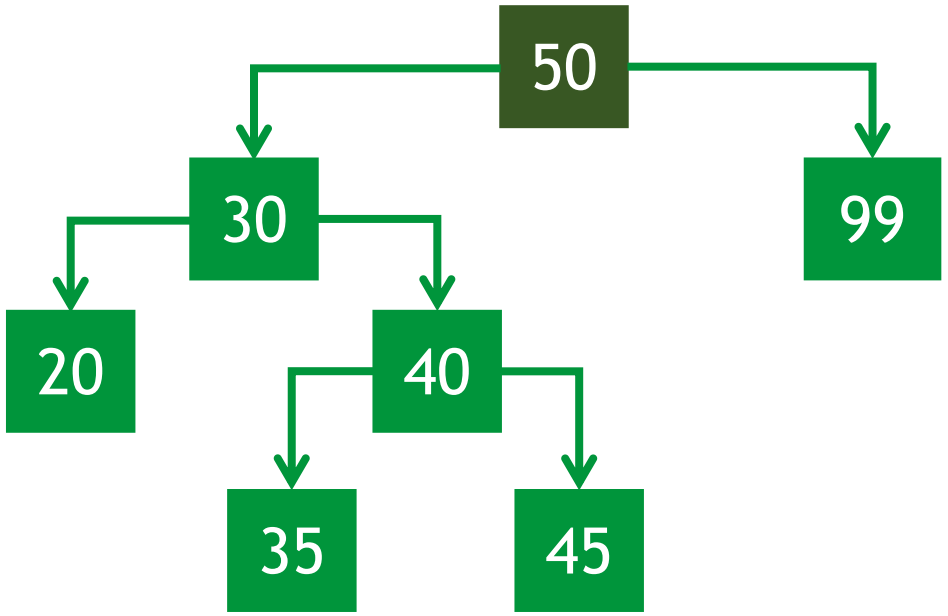


- Pré-ordem (RED):

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

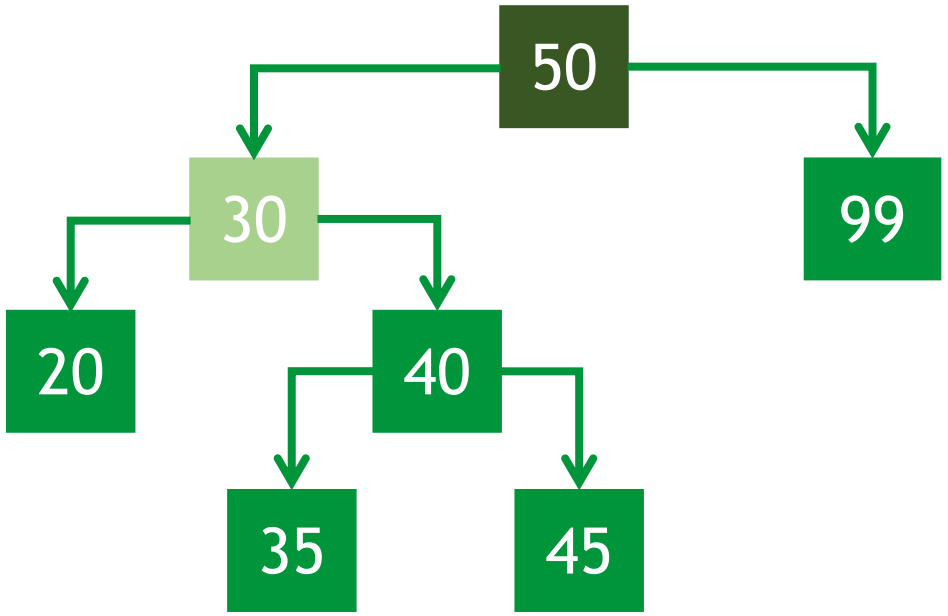


- Pré-ordem (RED): 50

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

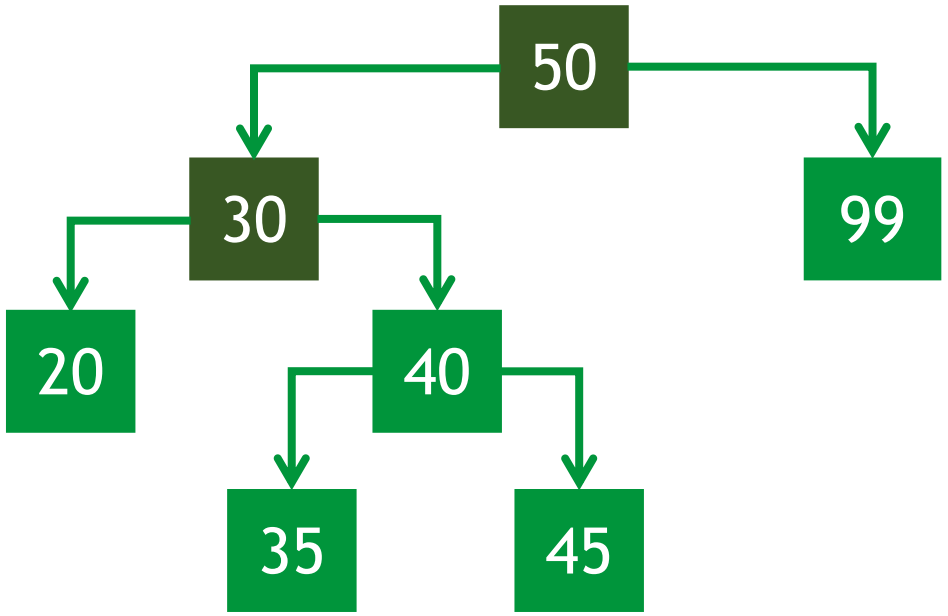


- Pré-ordem (RED): 50

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

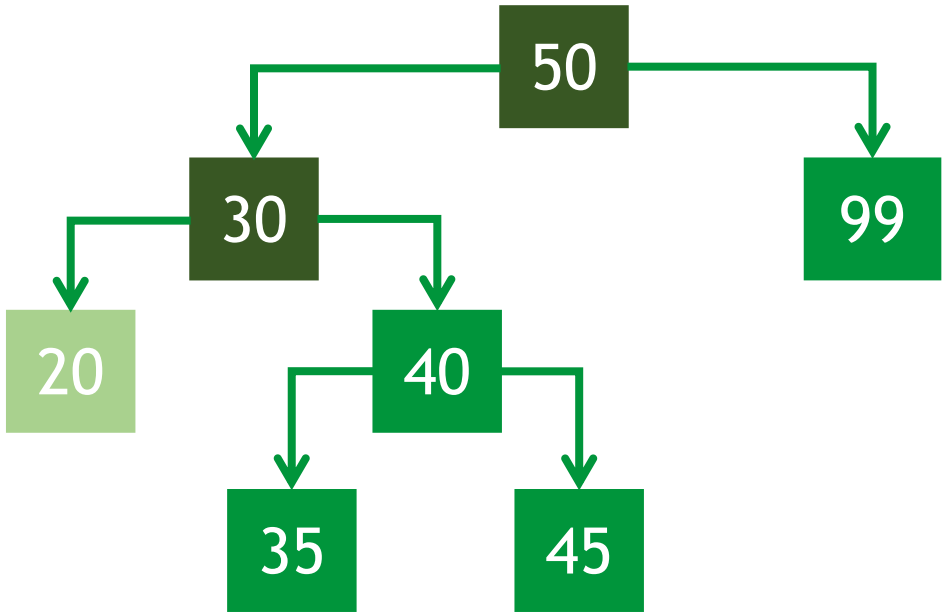


- Pré-ordem (RED): 50 - 30

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

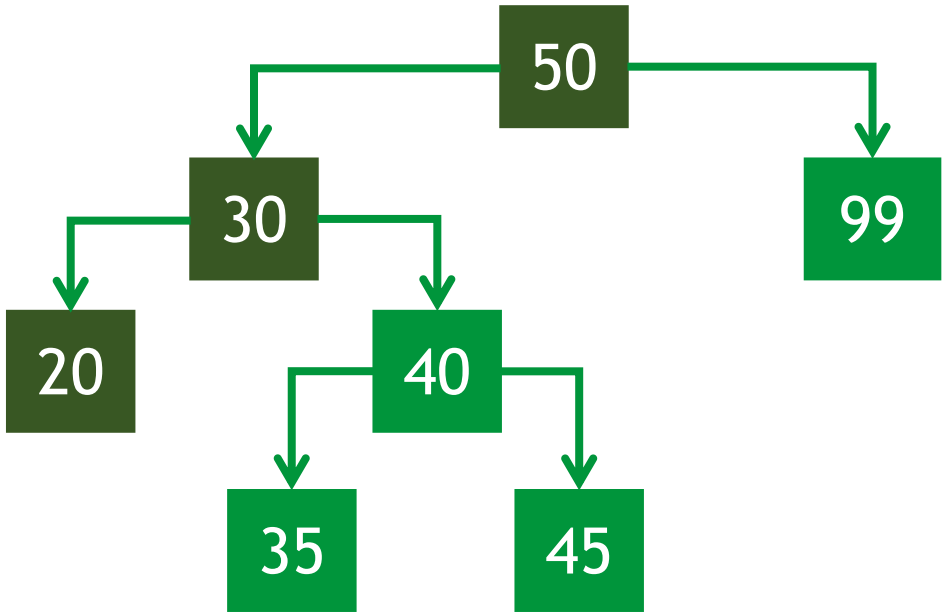


- Pré-ordem (RED): 50 - 30

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

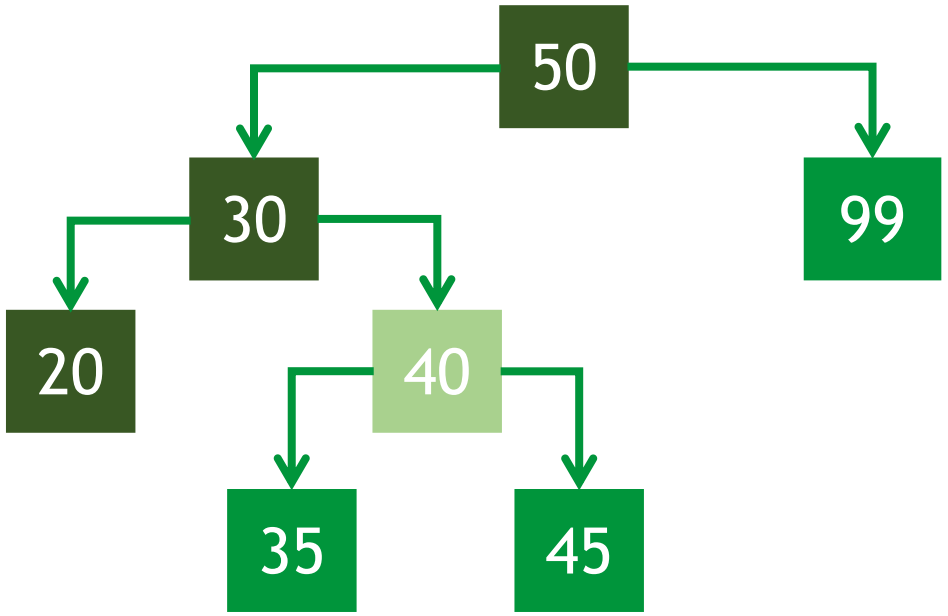


- Pré-ordem (RED): 50 - 30 - 20

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

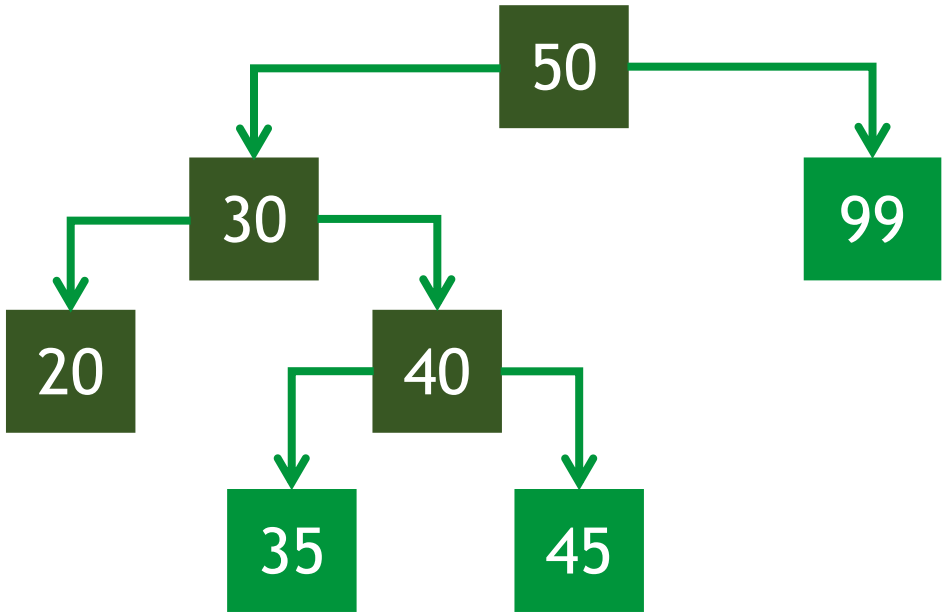


- Pré-ordem (RED): 50 - 30 - 20

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

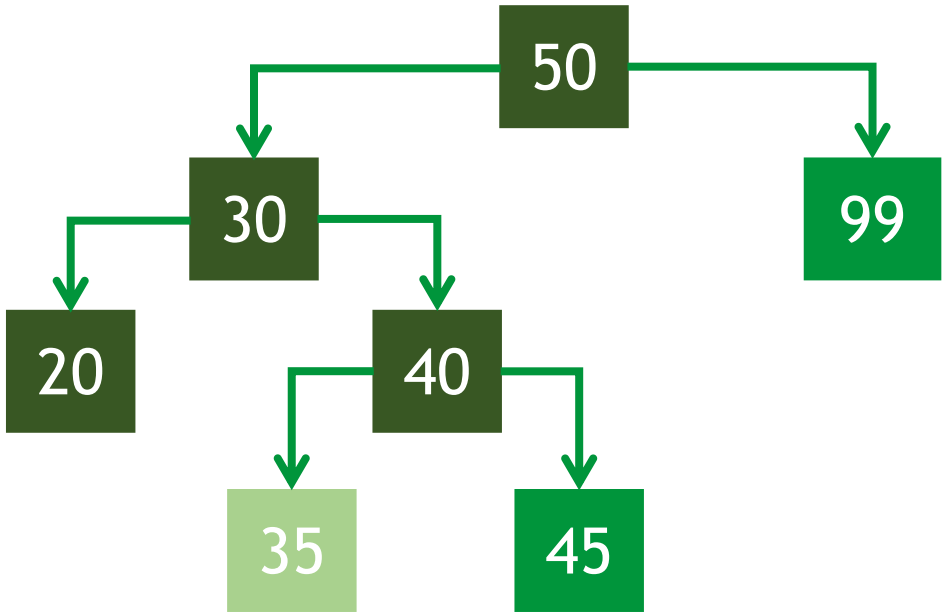


- Pré-ordem (RED): 50 - 30 - 20 - 40

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

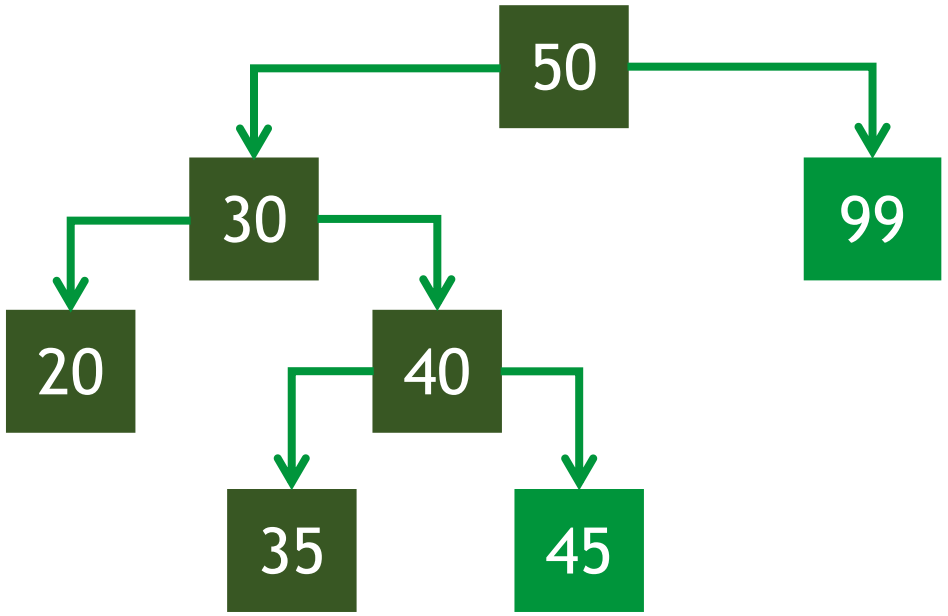


- Pré-ordem (RED): 50 - 30 - 20 - 40

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

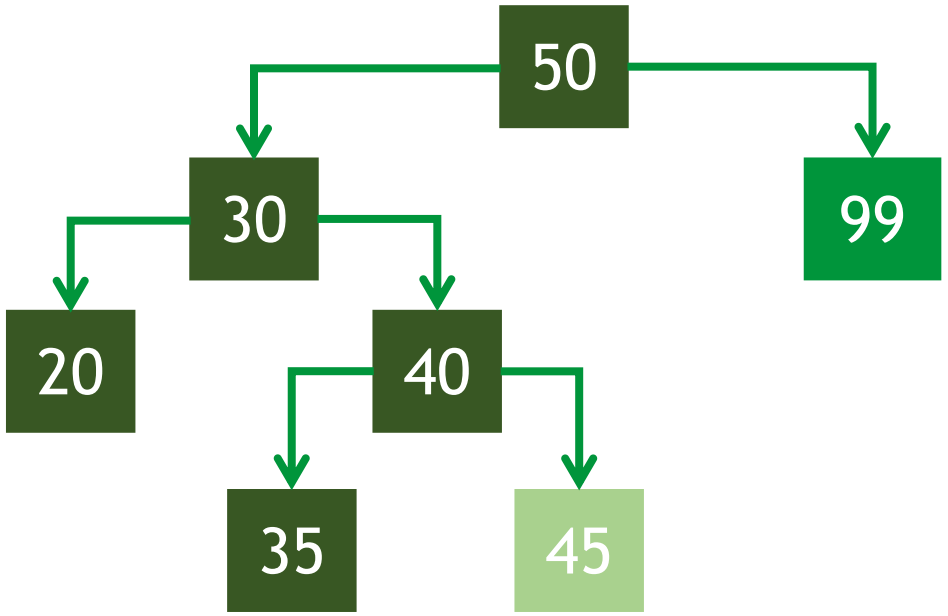


- Pré-ordem (RED): 50 - 30 - 20 - 40 - 35

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

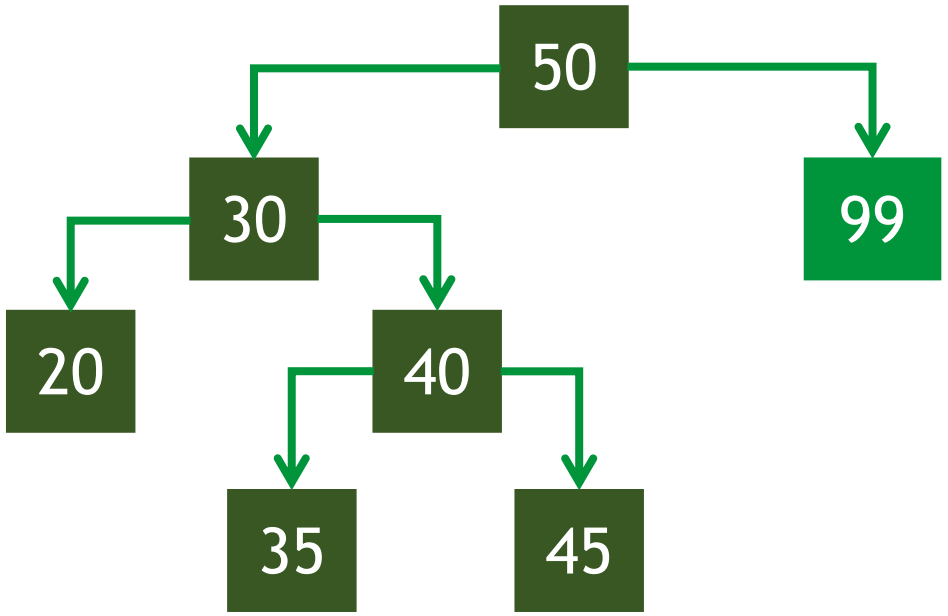


- Pré-ordem (RED): 50 - 30 - 20 - 40 - 35

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

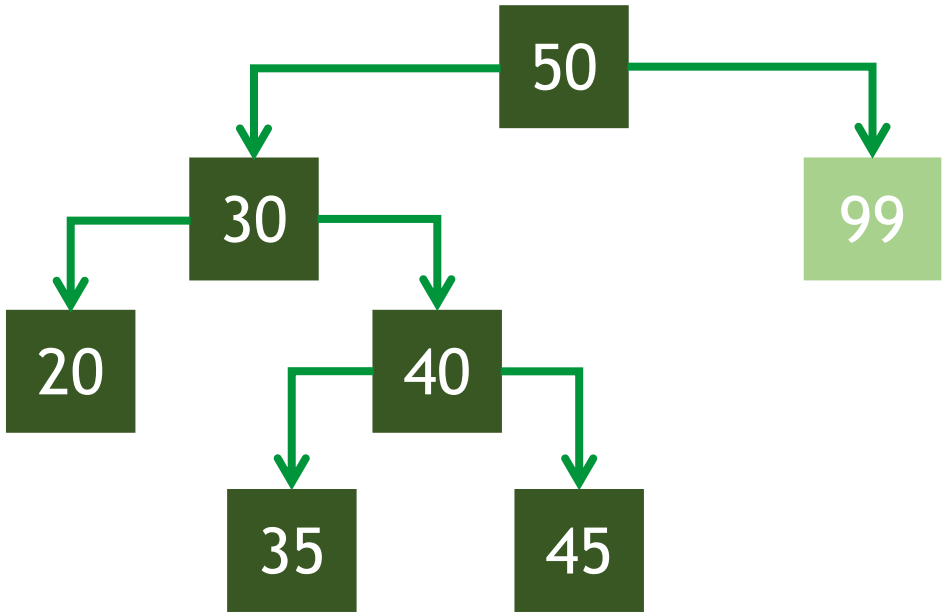


- Pré-ordem (RED): 50 - 30 - 20 - 40 - 35 - 45

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):

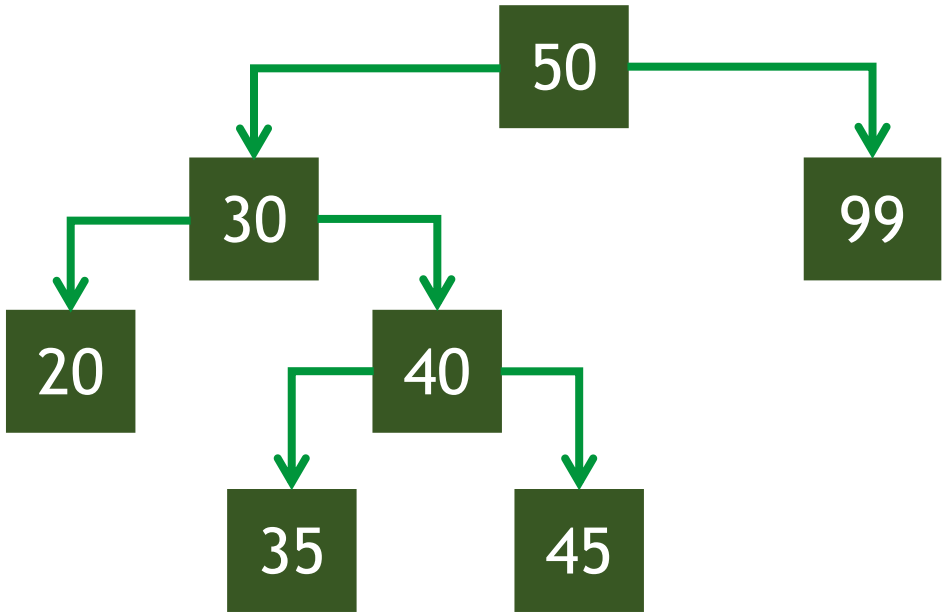


- Pré-ordem (RED): 50 - 30 - 20 - 40 - 35 - 45

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em Pré-ordem (R E D):



- Pré-ordem (RED): 50 - 30 - 20 - 40 - 35 - 45 - 99

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

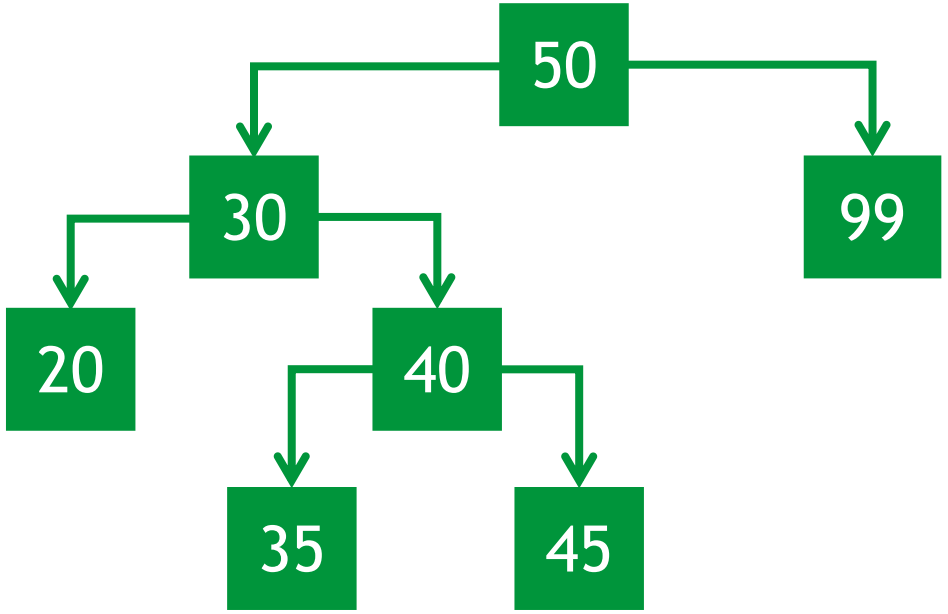
- Implementação da travessia pré-ordem em C:

```c
void preOrdem(No *raiz) {

    if (raiz != NULL) {

        printf("%d ", raiz->valor);

        preOrdem(raiz->esquerda);
        preOrdem(raiz->direita);
    }
}
```

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

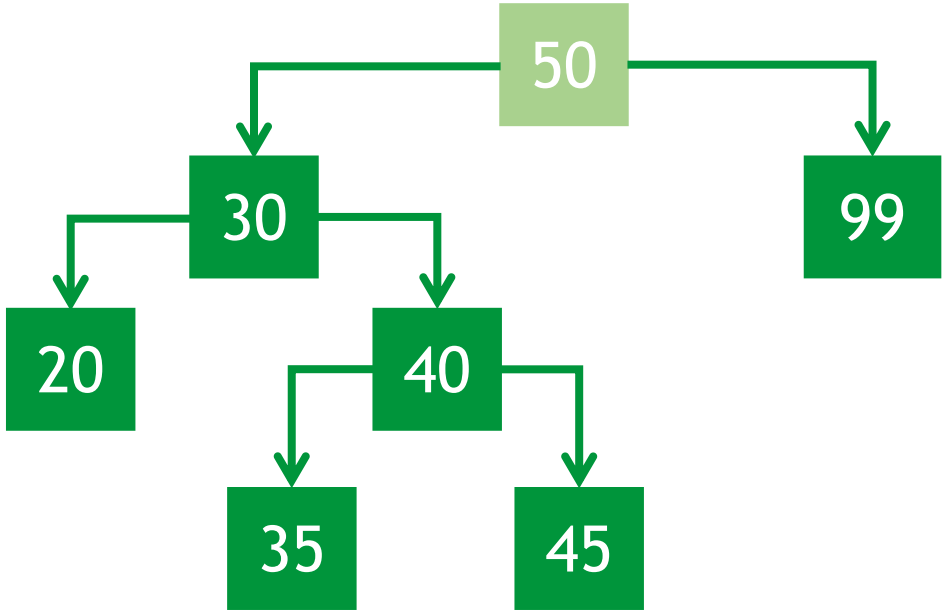
- Exemplo de travessia em ordem ou simétrica (E R D):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

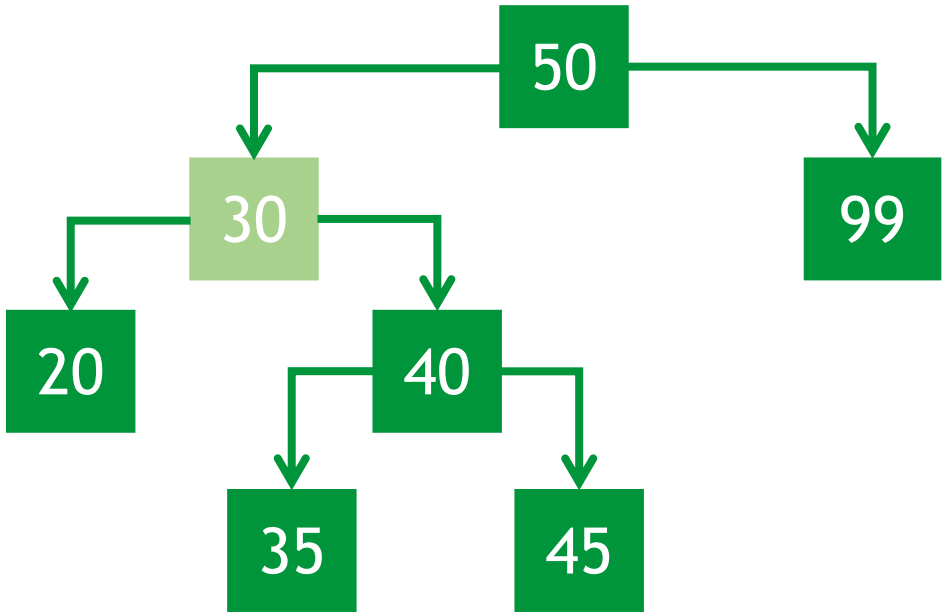


- Ordem ou simétrica (E R D):

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

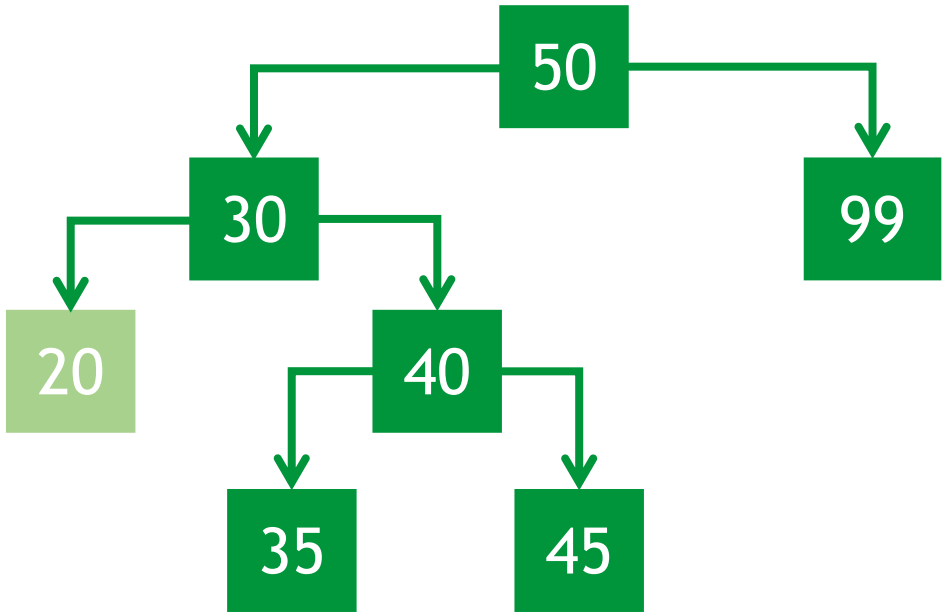


- Ordem ou simétrica (E R D):

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

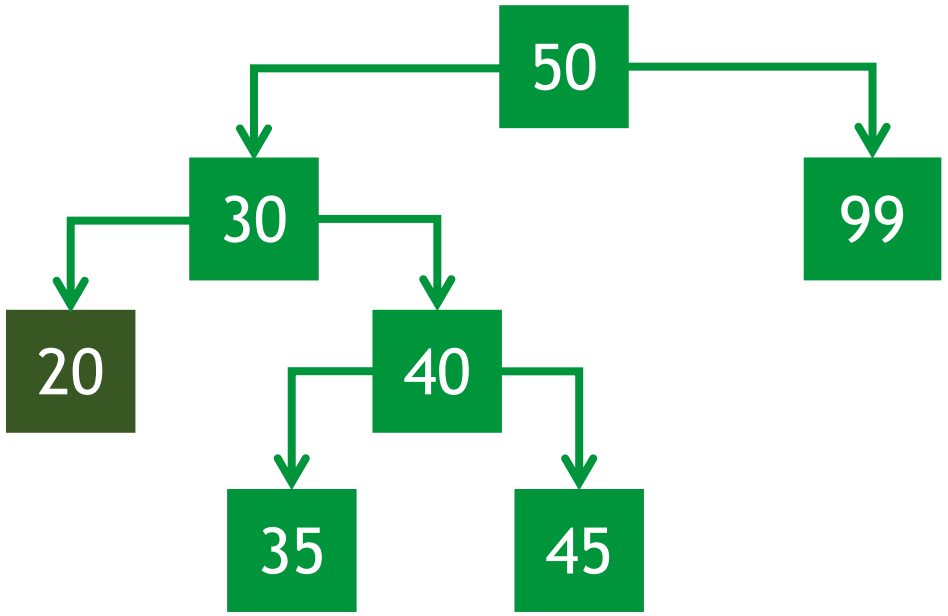


- Ordem ou simétrica (E R D):

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

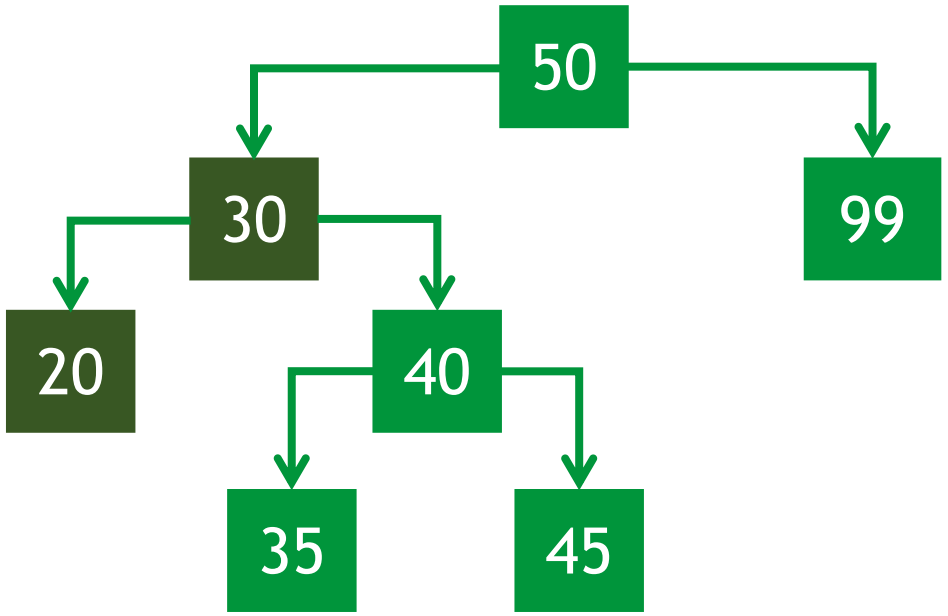


- Ordem ou simétrica (E R D): 20

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

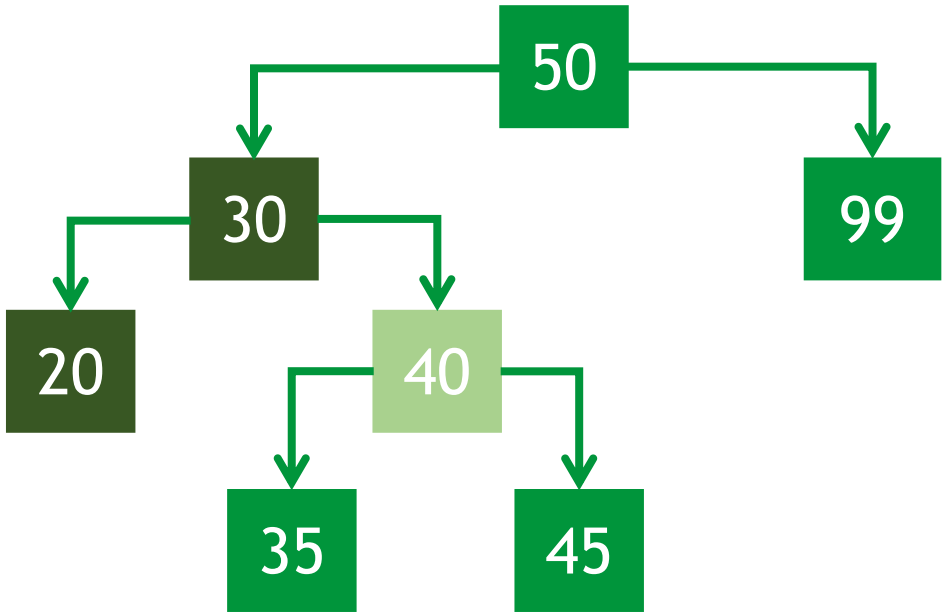


- Ordem ou simétrica (E R D): 20 - 30

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

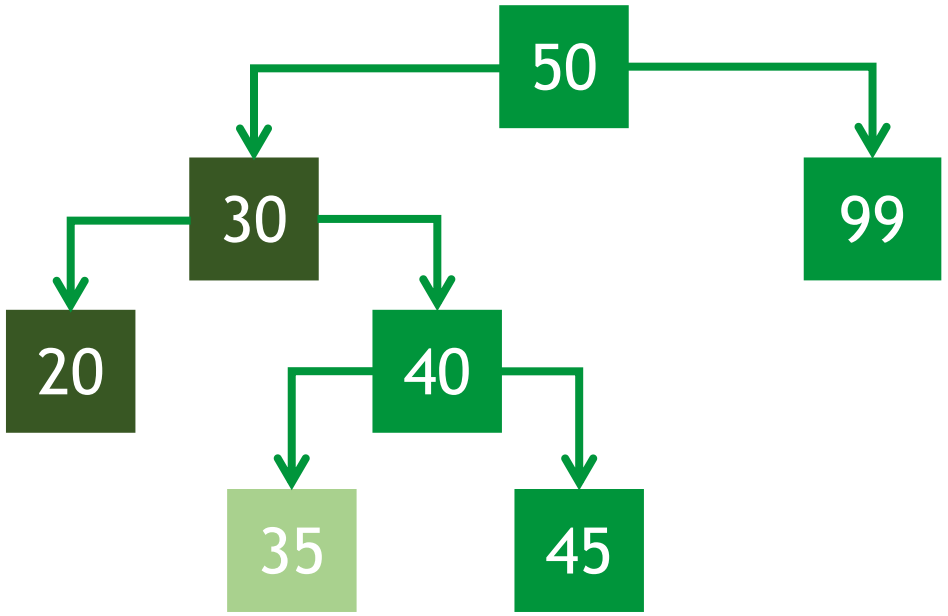


- Ordem ou simétrica (E R D): 20 - 30

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

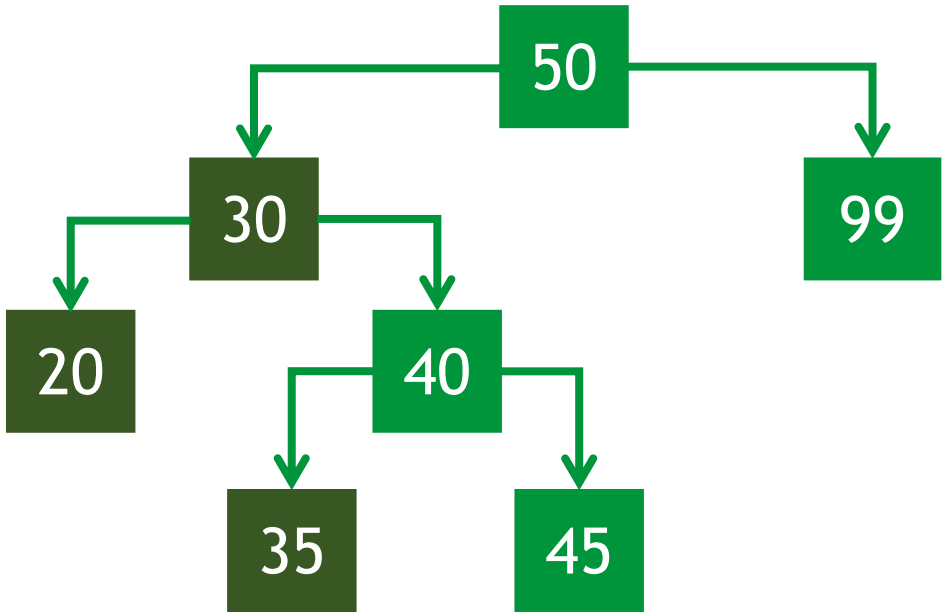


- Ordem ou simétrica (E R D): 20 - 30

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

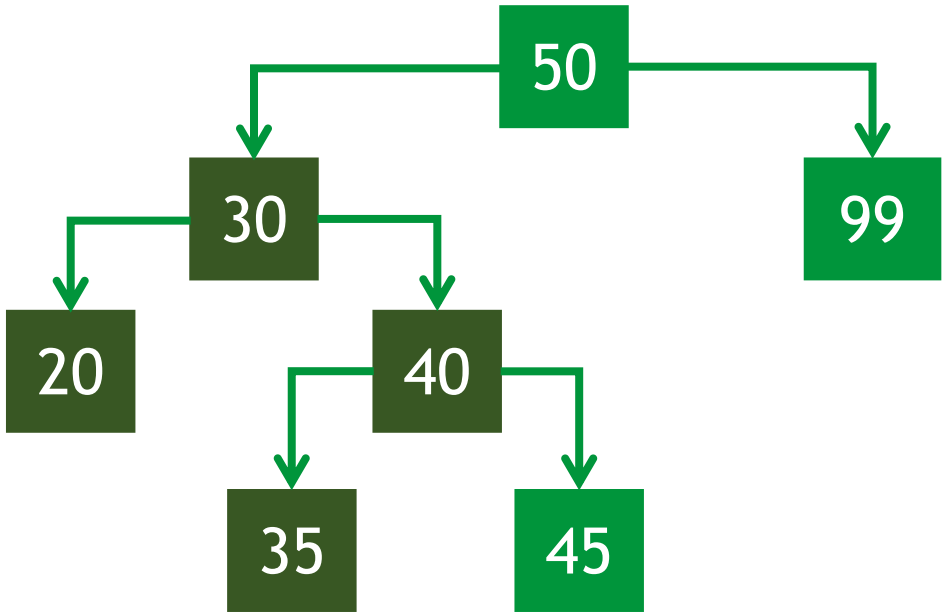


- Ordem ou simétrica (E R D): 20 - 30 - 35

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

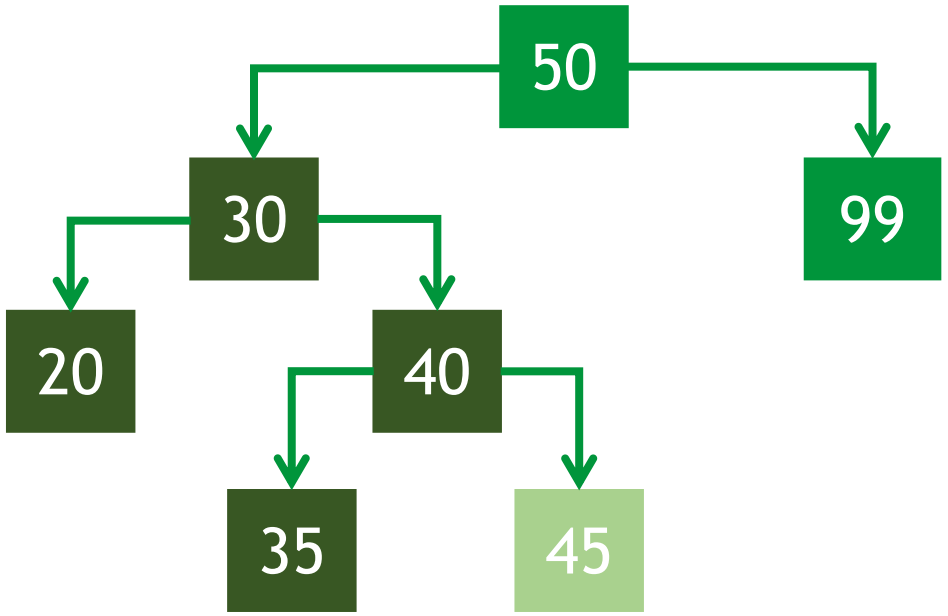


- Ordem ou simétrica (E R D): 20 - 30 - 35 - 40

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

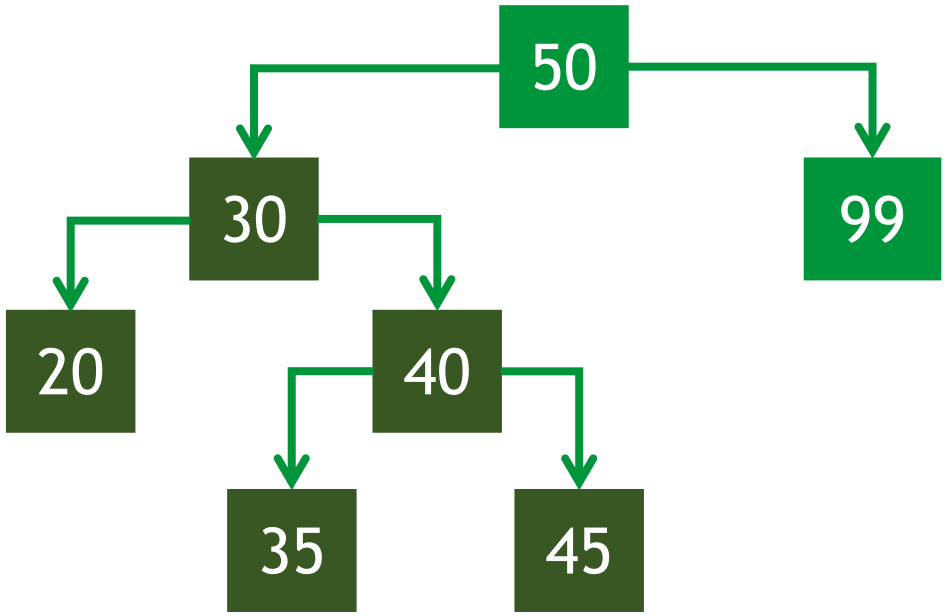


- Ordem ou simétrica (E R D): 20 - 30 - 35 - 40

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

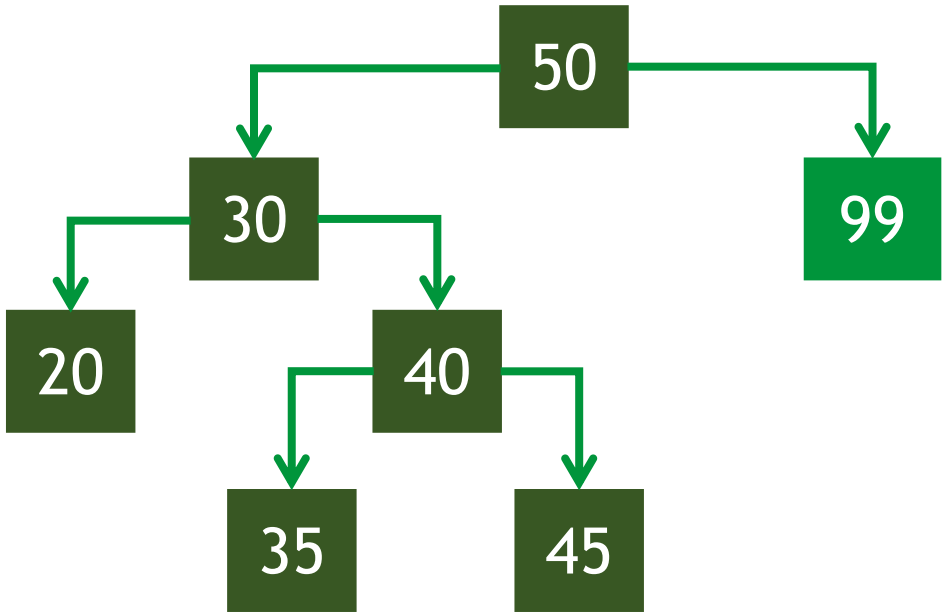


- Ordem ou simétrica (E R D): 20 - 30 - 35 - 40 - 45

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

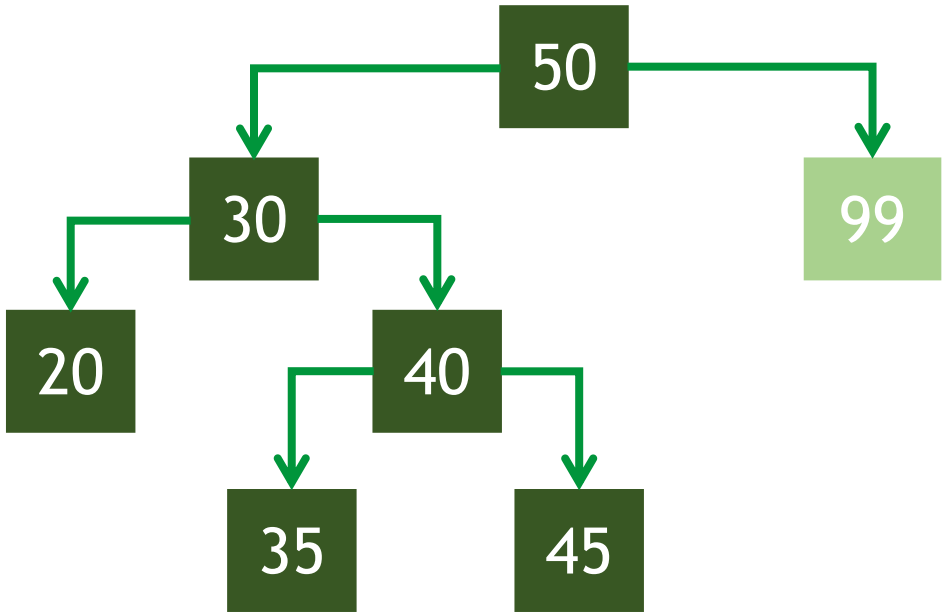


- Ordem ou simétrica (E R D): 20 - 30 - 35 - 40 - 45 - 50

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):

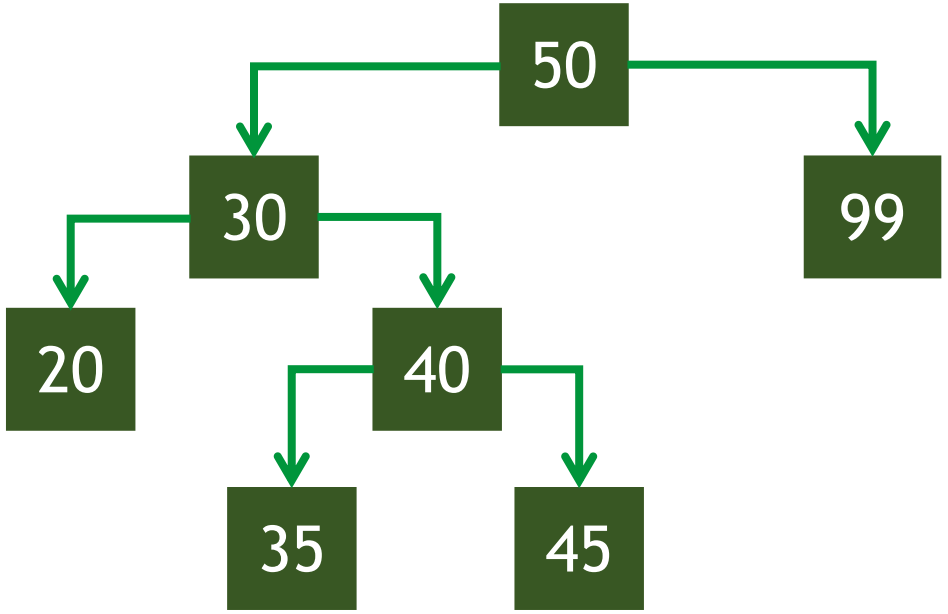


- Ordem ou simétrica (E R D): 20 - 30 - 35 - 40 - 45 - 50

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em ordem ou simétrica (E R D):



- Ordem ou simétrica (E R D): 20 - 30 - 35 - 40 - 45 - 50 - 99

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

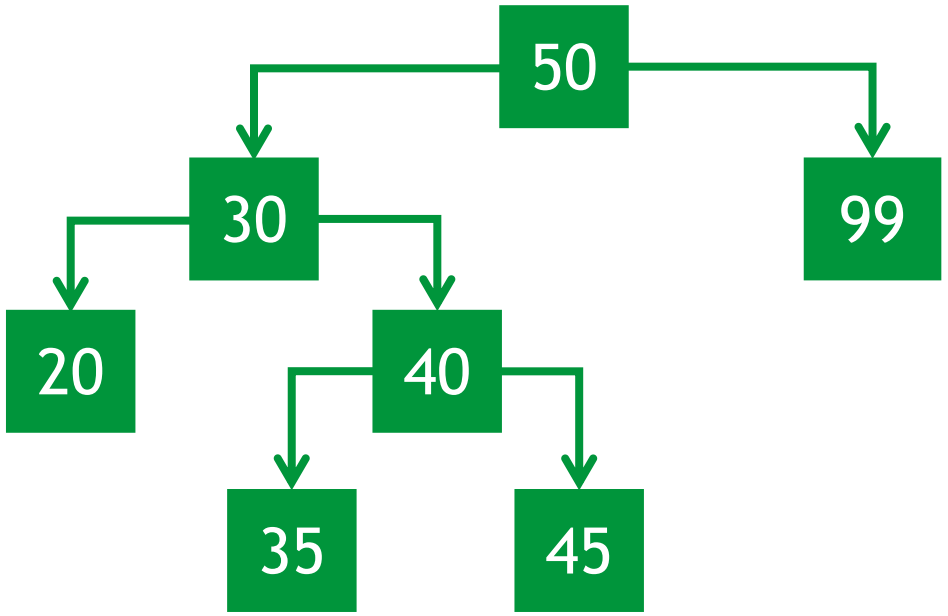
- Implementação da travessia em C:

```c
void emOrdem(No *raiz) {

    if (raiz != NULL) {

        emOrdem(raiz->esquerda);

        printf("%d ", raiz->valor);

        emOrdem(raiz->direita);
    }
}
```

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

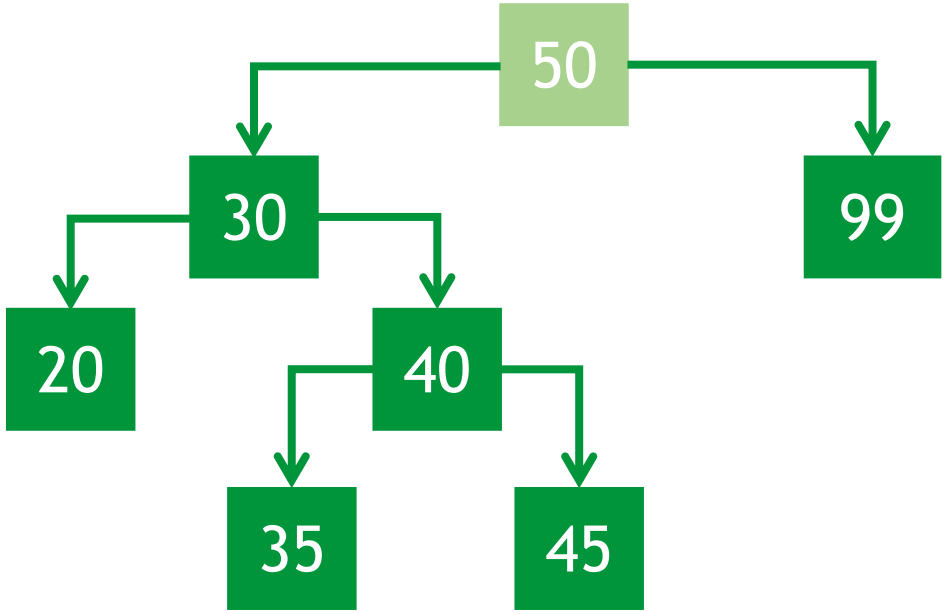
- Exemplo de travessia em pós-ordem (E D R):



Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

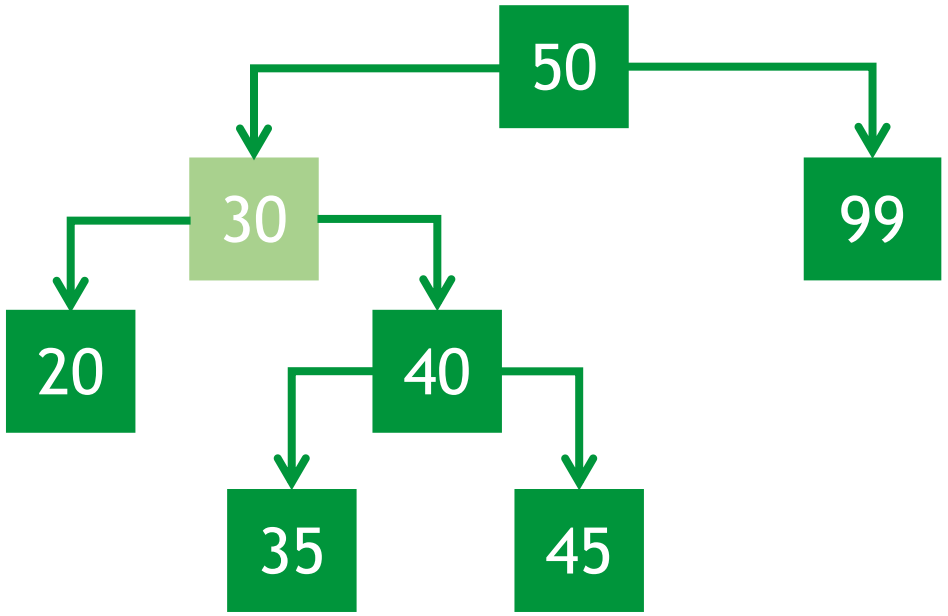


- Pós-ordem (E D R):

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

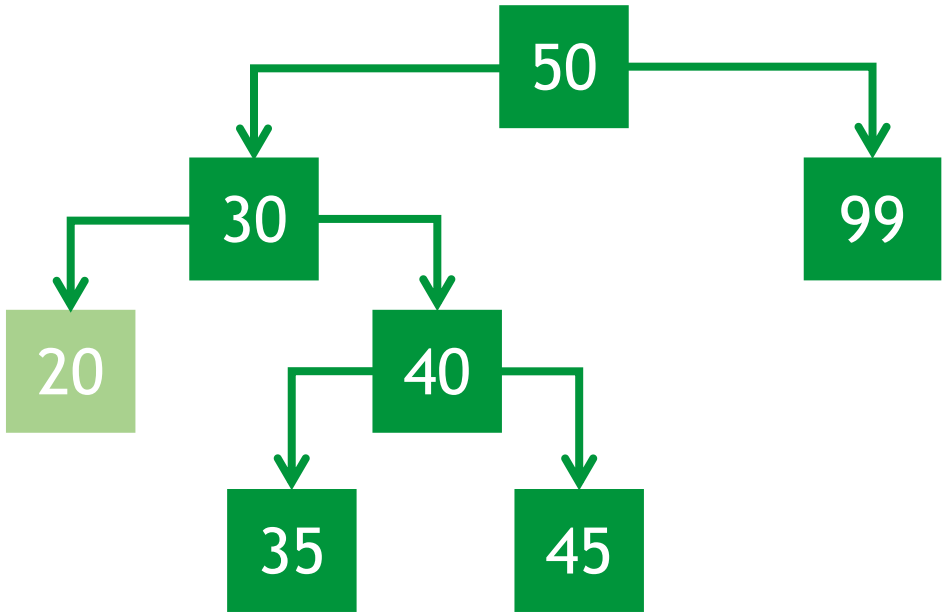


- Pós-ordem (E D R):

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

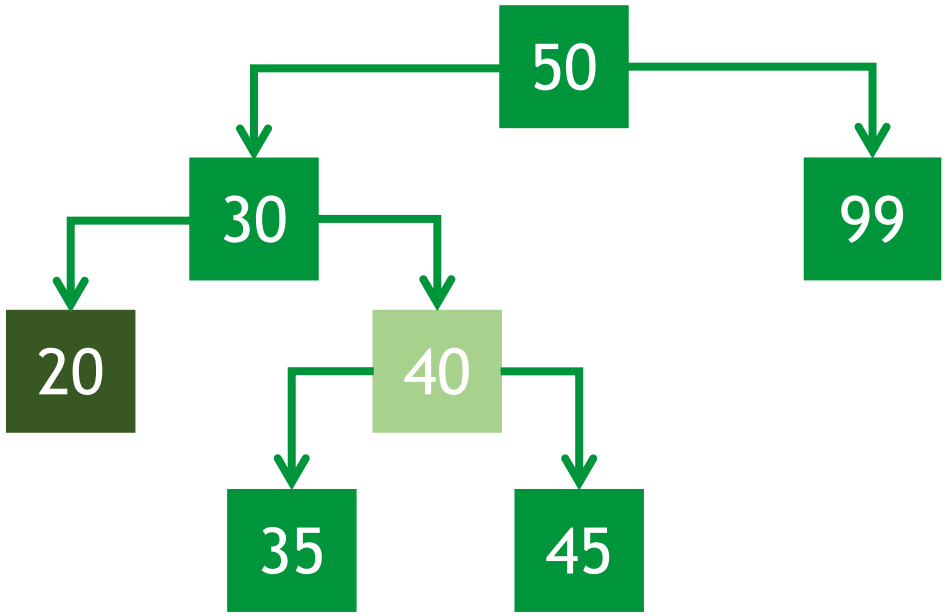


- Pós-ordem (E D R):

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

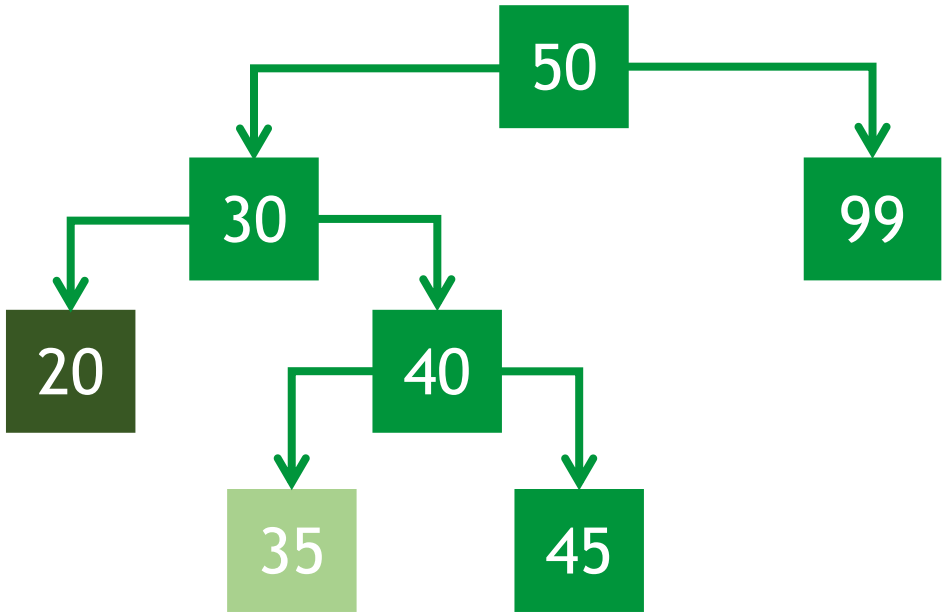


- Pós-ordem (E D R): 20

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

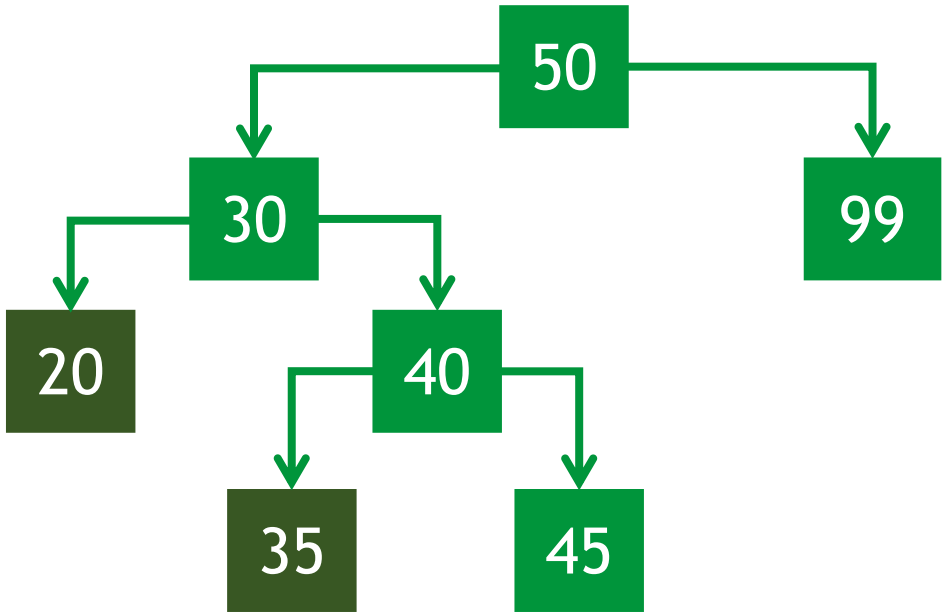


- Pós-ordem (E D R): 20

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

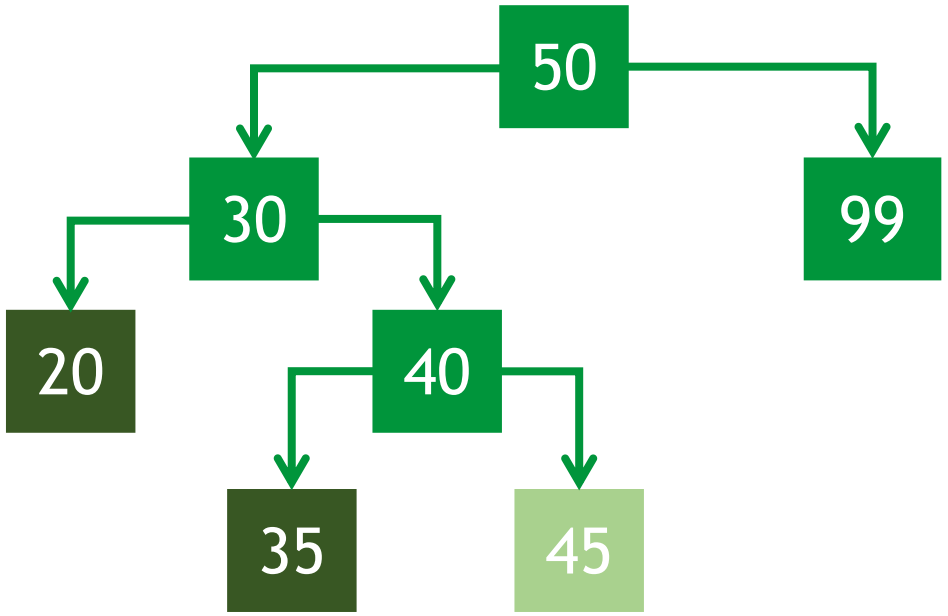


- Pós-ordem (E D R): 20 - 35

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

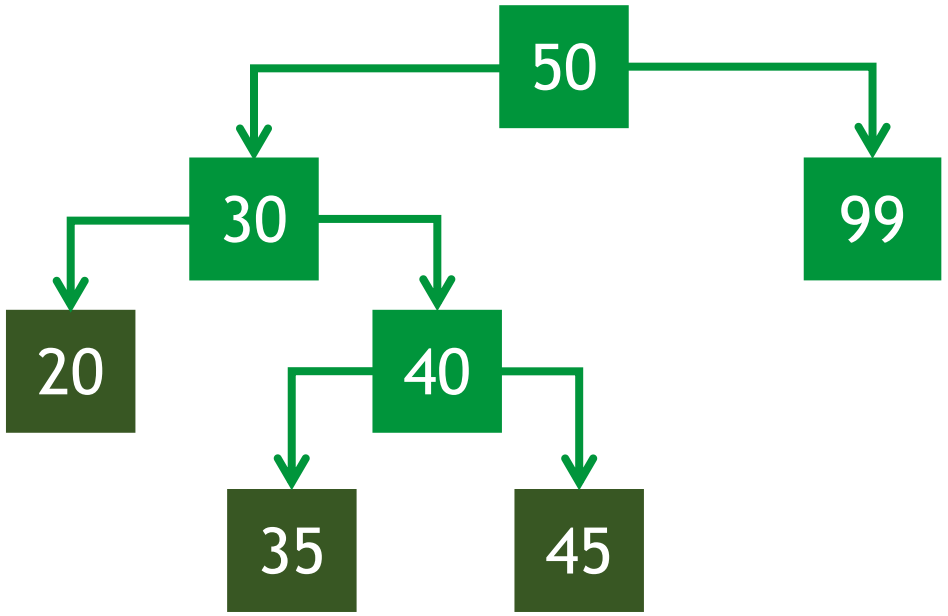


- Pós-ordem (E D R): 20 - 35

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

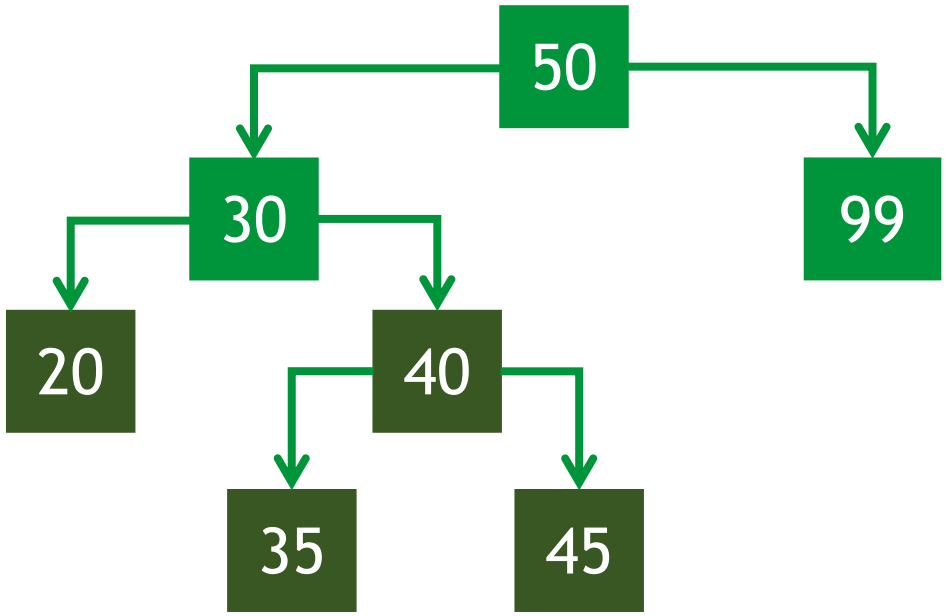


- Pós-ordem (E D R): 20 - 35 - 45

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

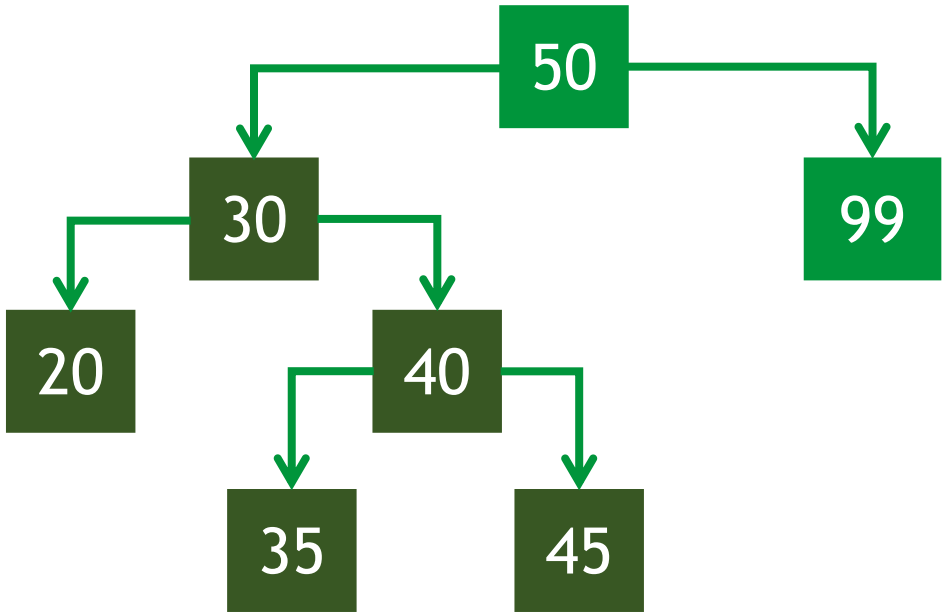


- Pós-ordem (E D R): 20 - 35 - 45 - 40

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

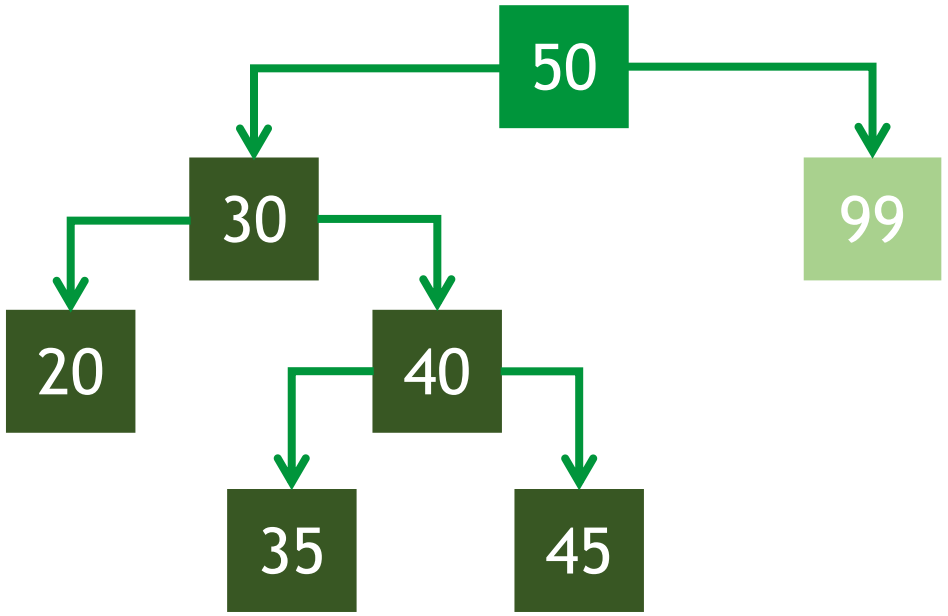


- Pós-ordem (E D R): 20 - 35 - 45 - 40 - 30

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

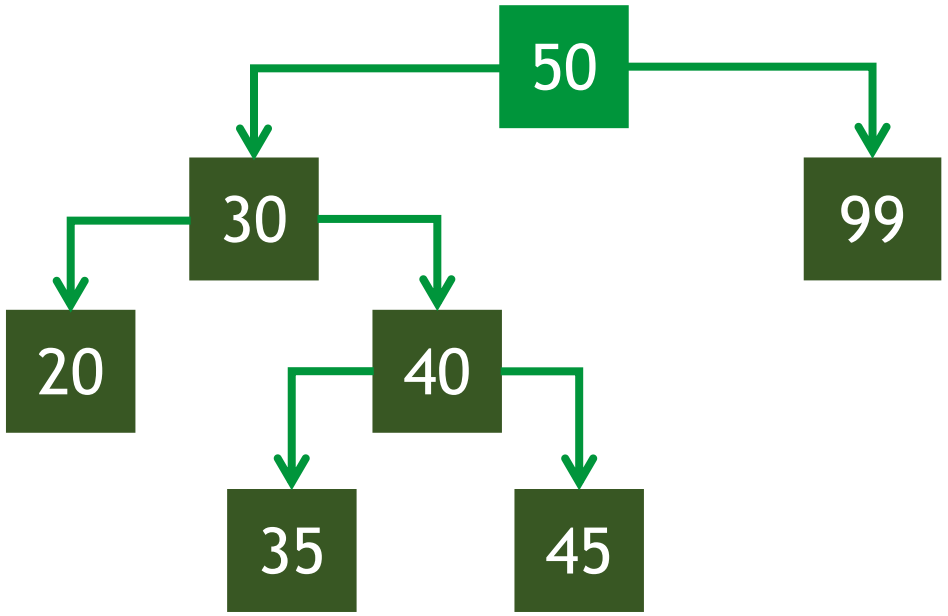


- Pós-ordem (E D R): 20 - 35 - 45 - 40 - 30

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):

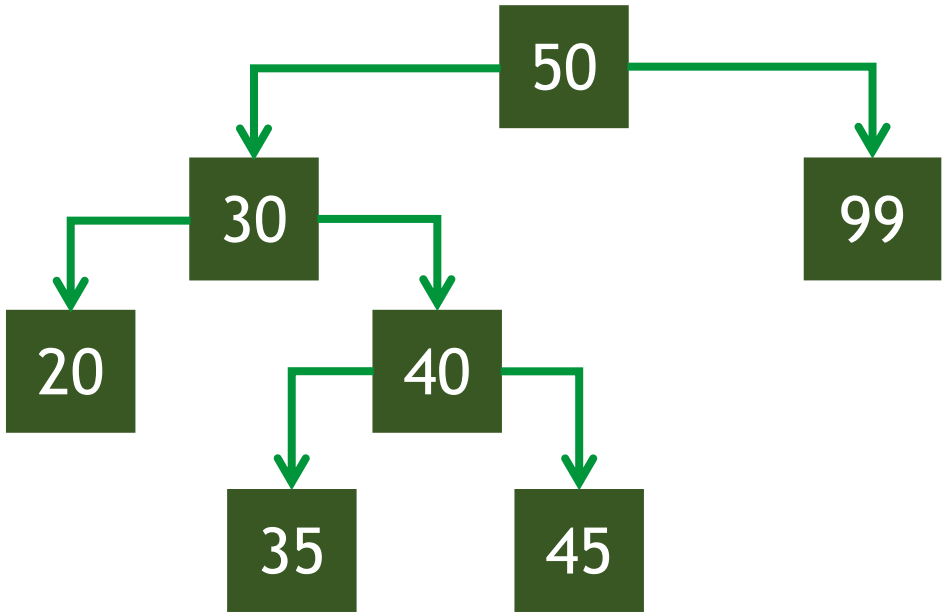


- Pós-ordem (E D R): 20 - 35 - 45 - 40 - 30 - 99

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Exemplo de travessia em pós-ordem (E D R):



- Pós-ordem (E D R): 20 - 35 - 45 - 40 - 30 - 99 - 50

Árvore Binária de Busca (BST)

INSTITUTO FEDERAL
Pernambuco

- Implementação da travessia em C:

```c
void posOrdem(No *raiz) {

    if (raiz != NULL) {

        posOrdem(raiz->esquerda);
        posOrdem(raiz->direita);

        printf("%d ", raiz->valor);
    }
}
```

Exercícios

INSTITUTO FEDERAL
Pernambuco

1. Considere uma BST inicialmente vazia. Insira os seguintes valores, nesta ordem:

50, 30, 70, 20, 40, 60, 80, 35, 45, 65

a) Desenhe a árvore.

b) Qual é a altura da árvore?

c) A árvore é:

( ) Cheia

( ) Completa

( ) Perfeita

Exercícios

INSTITUTO FEDERAL
Pernambuco

2. Escreva o resultado das seguintes travessias para a árvore gerada na questão 1.

a) Pré-ordem
b) Em ordem
c) Pós-ordem

Exercícios

INSTITUTO FEDERAL
Pernambuco

3. Considerando a árvore resultante da questão 1, mostre o caminho percorrido pela busca dos valores:

a) 65
b) 45
c) 90

Exercícios

INSTITUTO FEDERAL
Pernambuco

4. Considerando a árvore resultante da questão 1, realize as remoções na ordem indicada. Após cada remoção, desenhe a árvore:

a) Remover o nó 20
b) Remover o nó 60
c) Remover o nó 30
d) Remover o nó 50

Dúvidas

INSTITUTO FEDERAL
Pernambuco

ALGORITMOS E ESTRUTURA DE DADOS

Curso de Engenharia de Software
Lucas Sampaio Leite