Aula 5.1: Árvore de Busca Binária – BST - Remoção

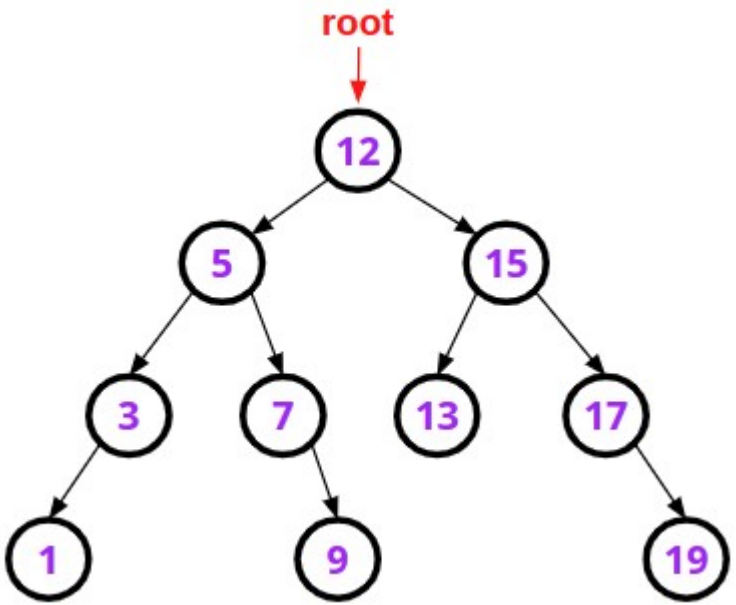
DCC405-Estrutura de Dados II

Prof. Me. Acauan C. Ribeiro

BST – Remoção de um Nó

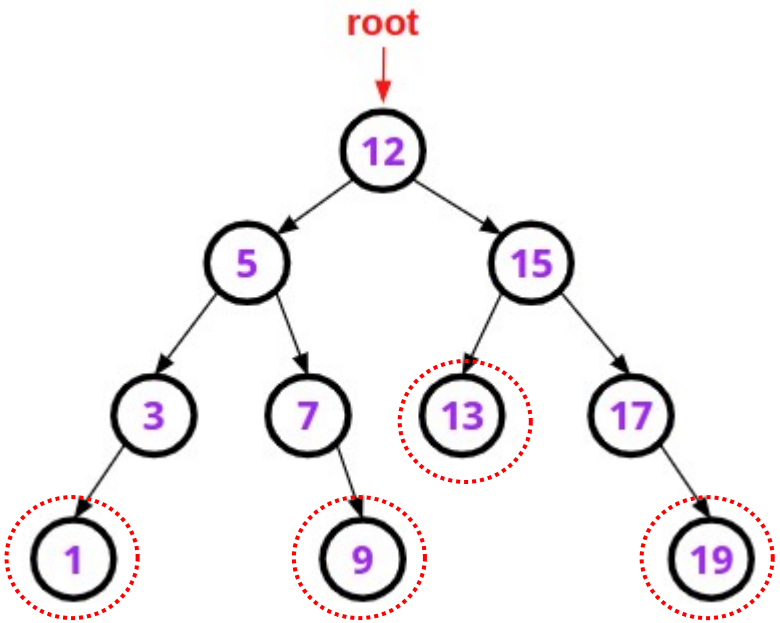
- A **remoção (delete)** de algum elemento na maioria das estruturas de dados é uma das ações que demandam um pouco mais de atenção e muitas vezes mais trabalho para ser feita. Em uma Árvore de Busca Binária não é diferente, pois temos que garantir a propriedade de Busca Binária após a remoção.

BST – Remoção de um Nó



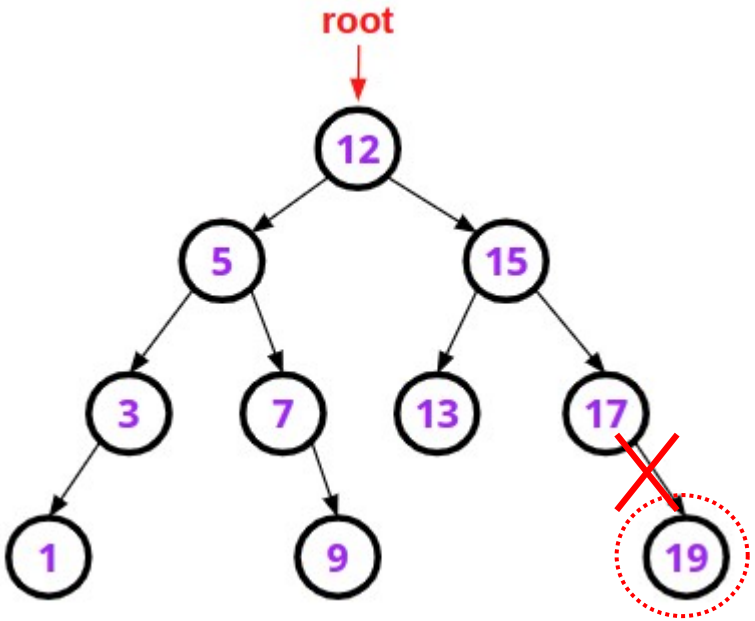
- 1° Caso: Remoção de um nó folha (Node sem filhos)

BST – Remoção de um Nó



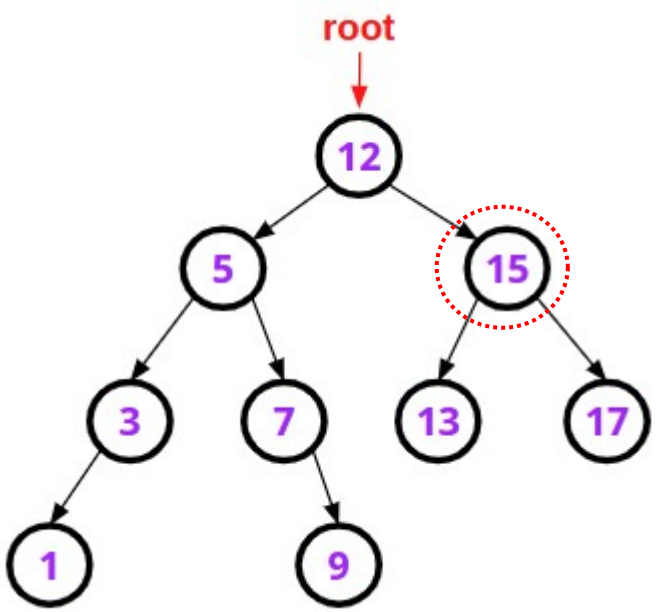
- **1° Caso: Remoção de um nó folha (Node sem filhos)**
  - Precisamos somente cortar a aresta (ligação) que referencia este nó vindo de seu pai.

BST – Remoção de um Nó



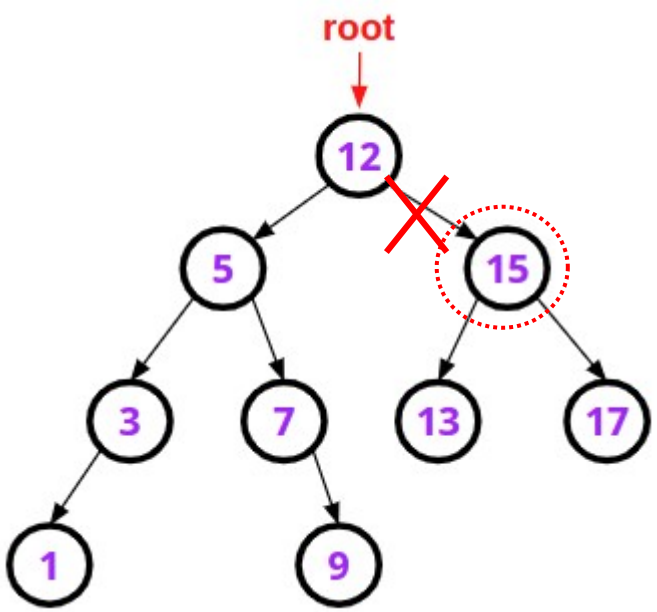
- **1° Caso: Remoção de um nó folha (Node sem filhos)**
  - Precisamos somente cortar a aresta (ligação) que referencia este nó vindo de seu pai.
  - Neste exemplo o **filho direito do 17** vai receber NULL.
  - Em seguida, devemos limpar o objeto deletado da memória, para que ela passa ser novamente utilizada para outra ação.

BST – Remoção de um Nó



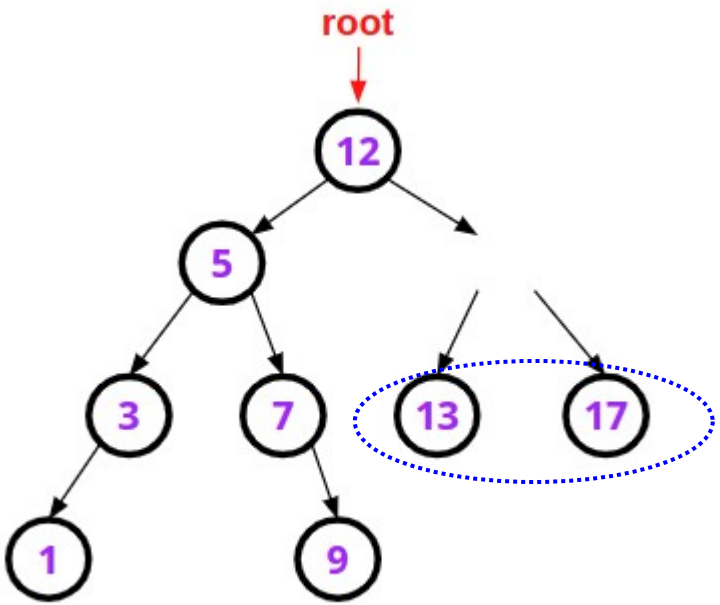
→ Suponha que queremos remover o **nó 15**..

BST – Remoção de um Nó



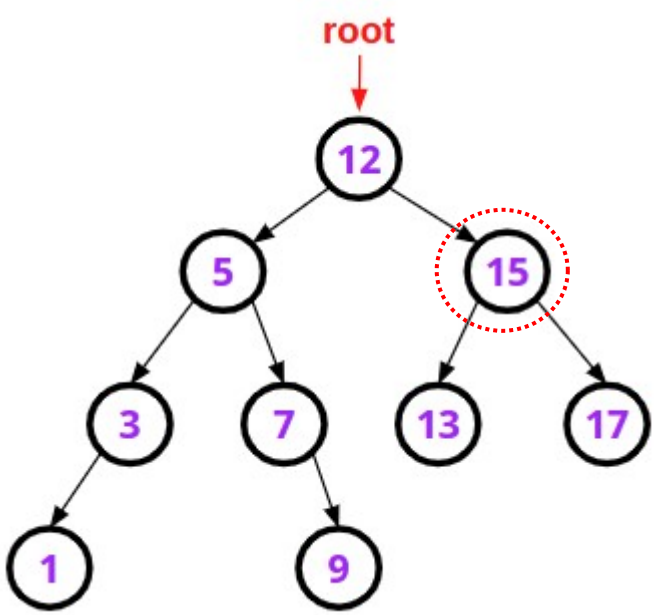
→ Suponha que queremos remover o **nó 15**, será que podemos somente cortar a ligação dele com seu pai?

BST – Remoção de um Nó



→ Suponha que queremos remover o **nó 15**, será que podemos somente cortar a ligação dele com seu pai?

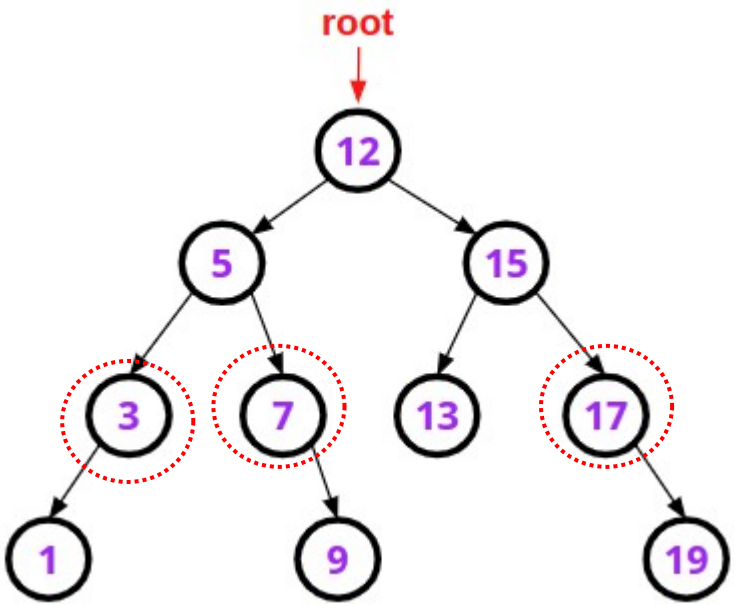
BST – Remoção de um Nó



→ Suponha que queremos remover o **nó 15**, será que podemos somente cortar a ligação dele com seu pai?

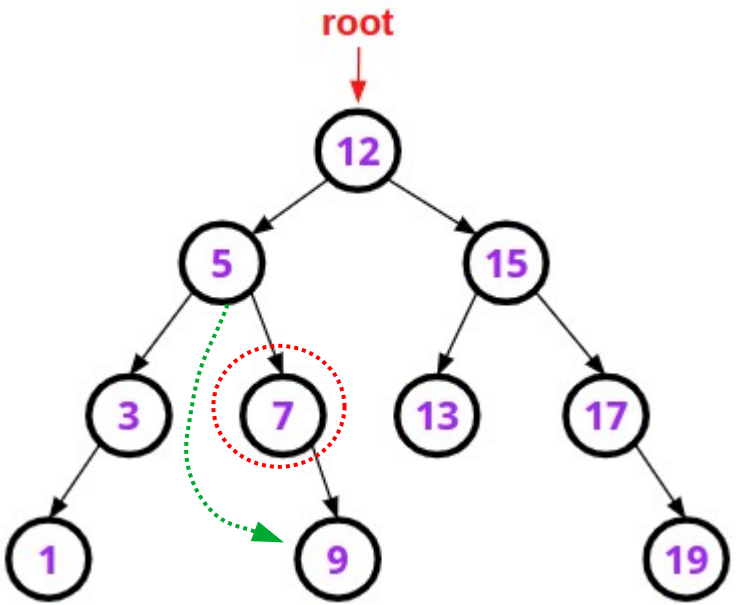
→ O caso do nó 15 e outros que tem mais de um filho é o caso mais complicado, já retornaremos para ele. Primeiro vamois verificar a remoção de um nó com somente um filho.

BST – Remoção de um Nó



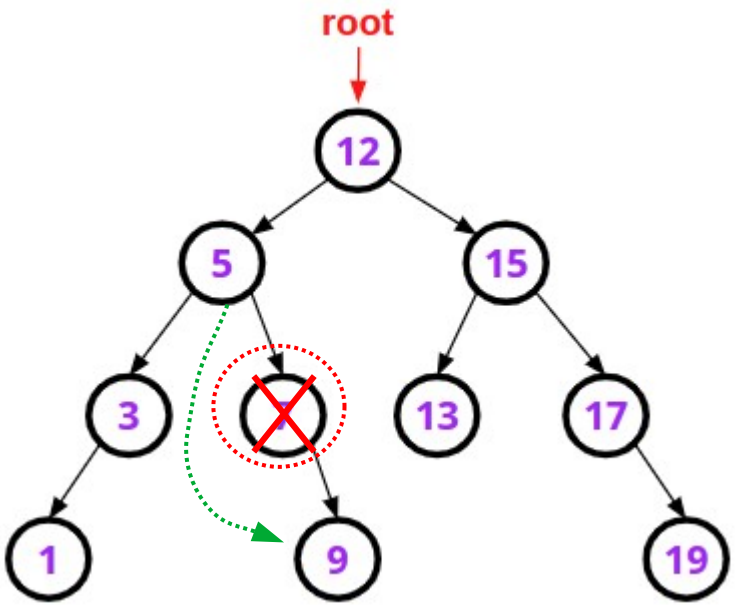
- 2° Caso: Remoção de um nó com apenas 1 filho

BST – Remoção de um Nó



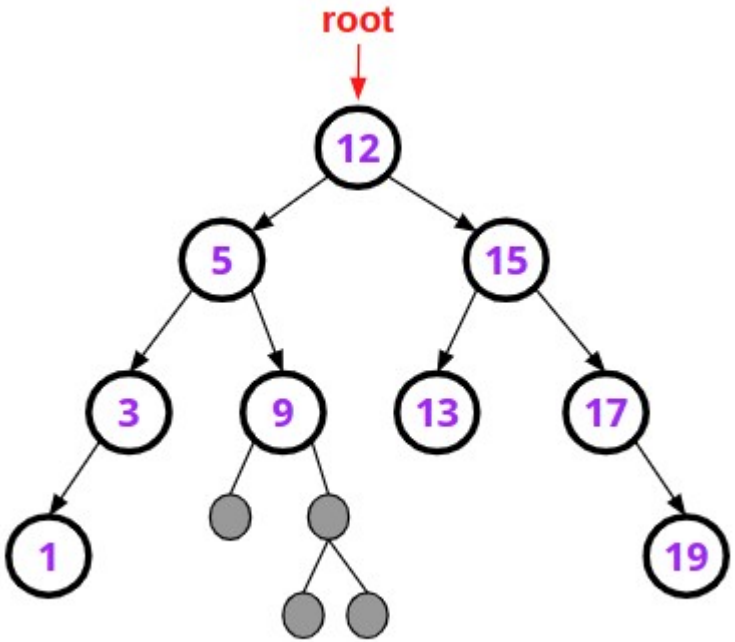
- **2° Caso: Remoção de um nó com apenas 1 filho**
  - Neste caso precisamos linkar o pai deste nó que está sendo removida ao seu único filho. Seria atribuir a guarda do neto ao avô.

BST – Remoção de um Nó



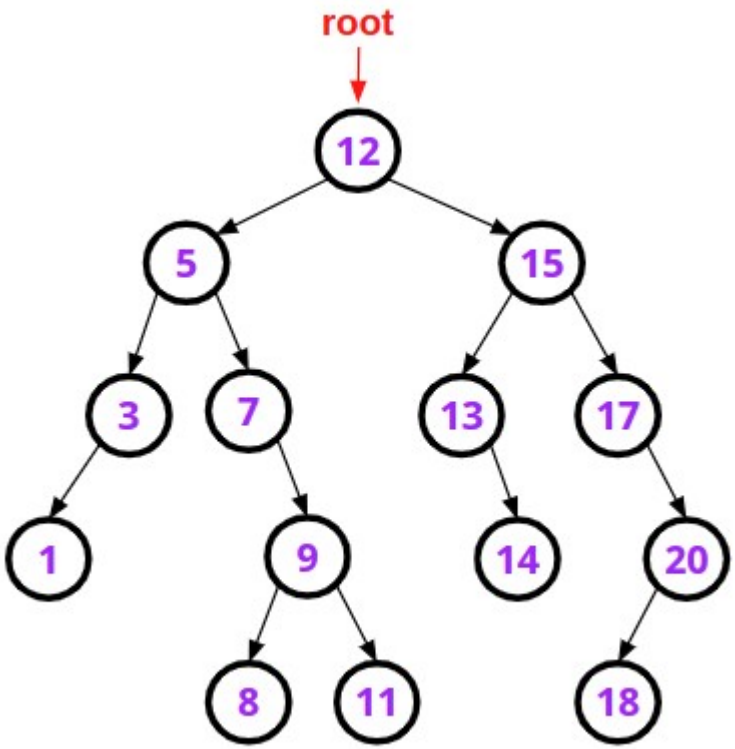
- **2° Caso: Remoção de um nó com apenas 1 filho**
  - Neste caso precisamos linkar o pai deste nó que está sendo removida ao seu único filho. Seria atribuir a guarda do neto ao avô.
  - Em seguida você pode remover o nó que só tinha um filho. No exemplo seria o nó 7.

BST – Remoção de um Nó



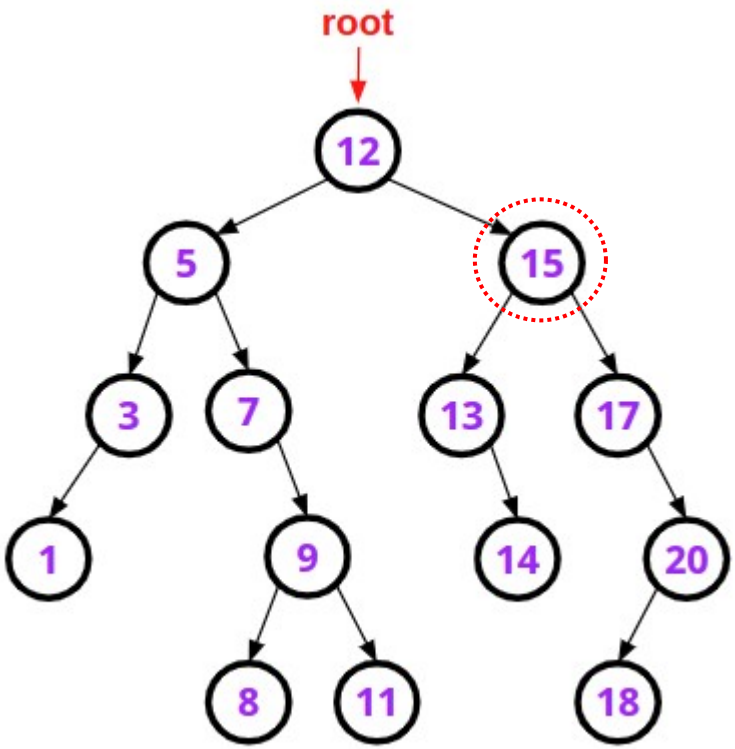
- **2° Caso: Remoção de um nó com apenas 1 filho**
  - Neste caso precisamos linkar o pai deste nó que está sendo removida ao seu único filho. Seria atribuir a guarda do neto ao avô.
  - Em seguida você pode remover o nó que só tinha um filho. No exemplo seria o nó 7.
  - O resultado é que mesmo se o nó filho tivesse outros filhos, a **propriedade de BST** seria preservada.

BST – Remoção de um Nó



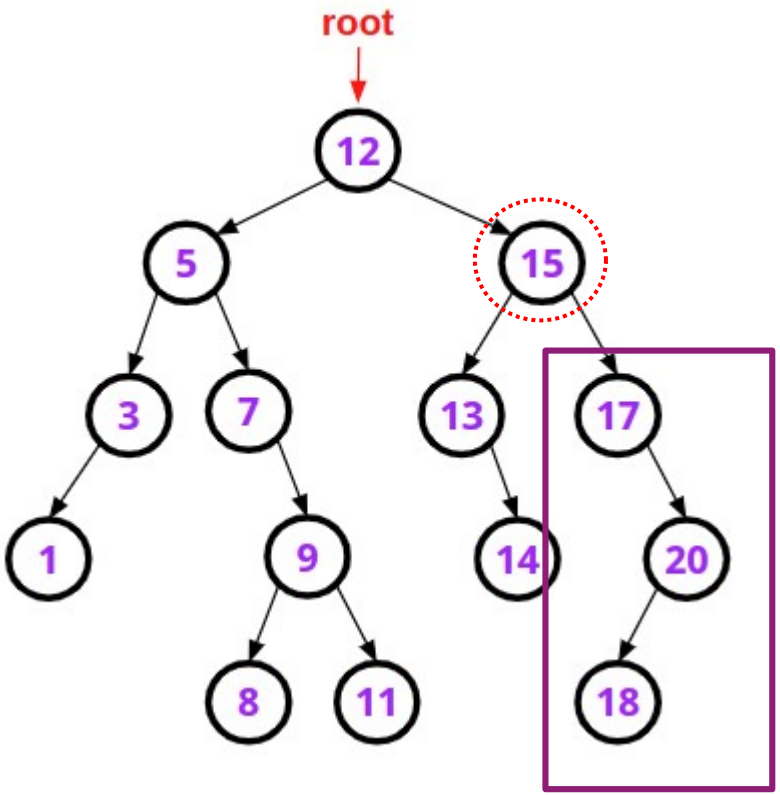
- 3° Caso: Remoção de um nó com 2 filhos

BST – Remoção de um Nó



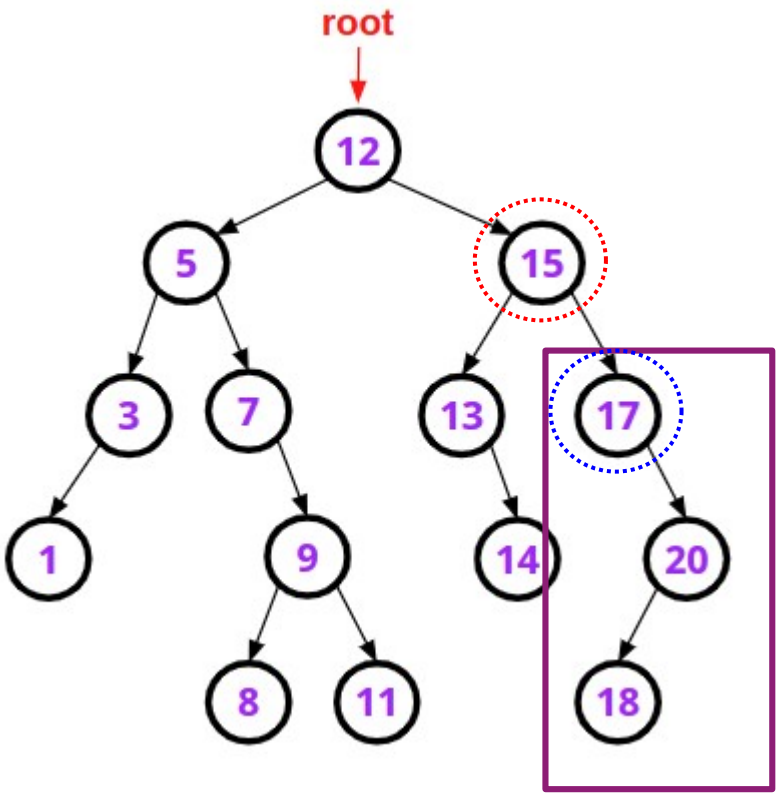
- **3° Caso: Remoção de um nó com 2 filhos**
  - Vamos tentar remover o **nó 15**. Quem deve tomar o lugar dele, que manterá a propriedade BST?

BST – Remoção de um Nó



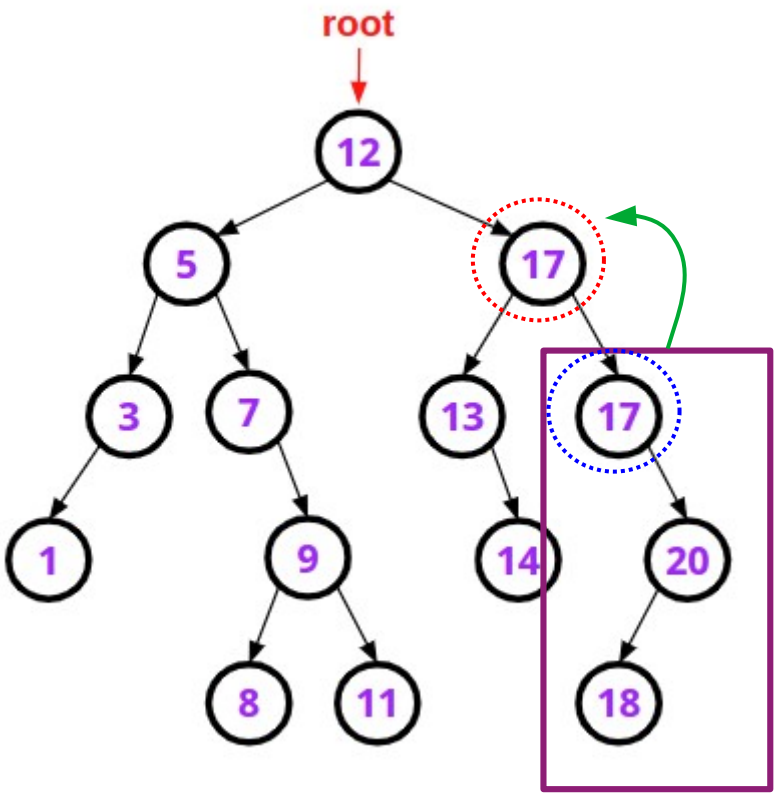
- **3° Caso: Remoção de um nó com 2 filhos**
  - Vamos tentar remover o **nó 15**. Quem deve tomar o lugar dele, que manterá a propriedade BST?
  - Temos duas opções:
    - 3.1 Min(sub-árvore da direita)

BST – Remoção de um Nó



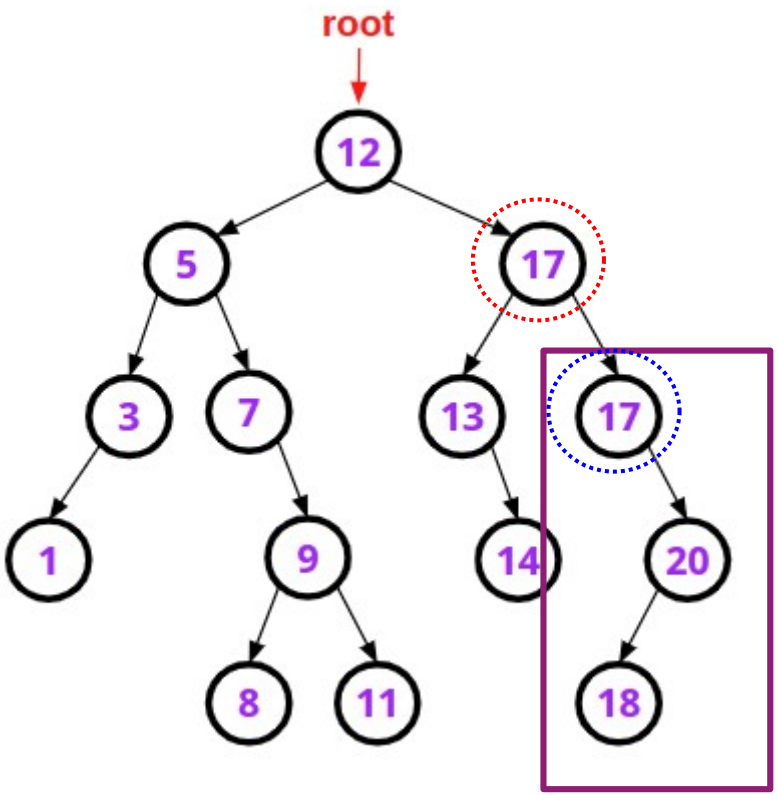
- **3° Caso: Remoção de um nó com 2 filhos**
  - Vamos tentar remover o **nó 15**. Quem deve tomar o lugar dele, que manterá a propriedade BST?
  - **Temos duas opções:**
    - 3.1 Min(sub-árvore da direita) = **17**

BST – Remoção de um Nó



- **3° Caso: Remoção de um nó com 2 filhos**
  - Vamos tentar remover o **nó 15**. Quem deve tomar o lugar dele, que manterá a propriedade BST?
  - **Temos duas opções:**
    - 3.1 Min(sub-árvore da direita) = **17**
      - **Copia-se o minimo para o lugar do nó que será removido**

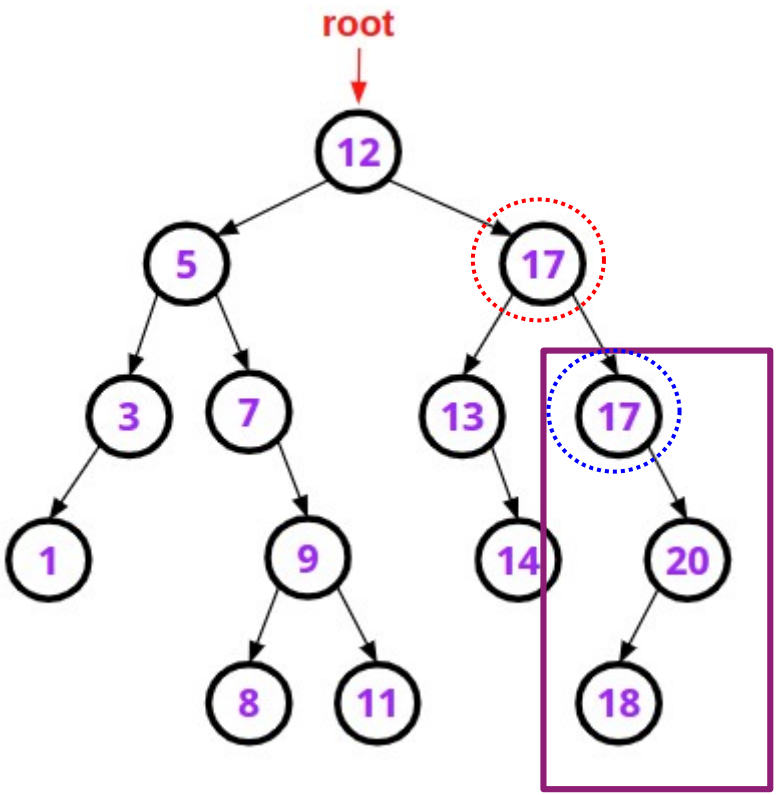
BST – Remoção de um Nó



- **3° Caso: Remoção de um nó com 2 filhos**
  - Vamos tentar remover o **nó 15**. Quem deve tomar o lugar dele, que manterá a propriedade BST?
  - **Temos duas opções:**
    - 3.1 Min(sub-árvore da direita) = **17**

      Observe que em uma árvore ou subárvore, se um nó tiver o valor mínimo, ele não terá um filho esquerdo.

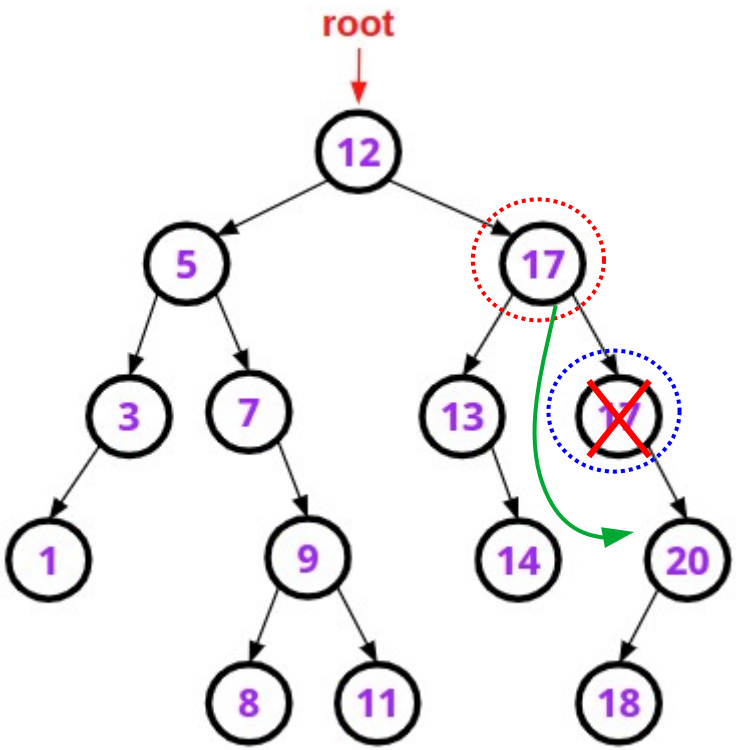
BST – Remoção de um Nó



- **3° Caso: Remoção de um nó com 2 filhos**
  - Vamos tentar remover o **nó 15**. Quem deve tomar o lugar dele, que manterá a propriedade BST?
  - **Temos duas opções:**
    - 3.1 Min(sub-árvore da direita) = **17**

      Caindo em algum dos dois casos vistos anteriormente:
      - **1° Caso: Nó folha**
      - **2° Caso: Nó com um filho**

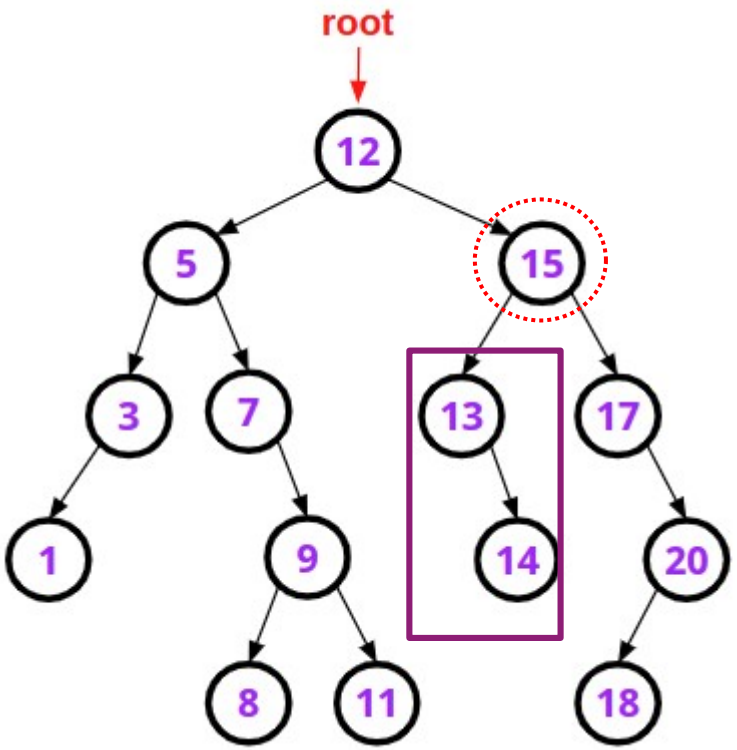
BST – Remoção de um Nó



- **3° Caso: Remoção de um nó com 2 filhos**
  - Vamos tentar remover o **nó 15**. Quem deve tomar o lugar dele, que manterá a propriedade BST?
  - **Temos duas opções:**
    - 3.1 Min(sub-árvore da direita) = **17**

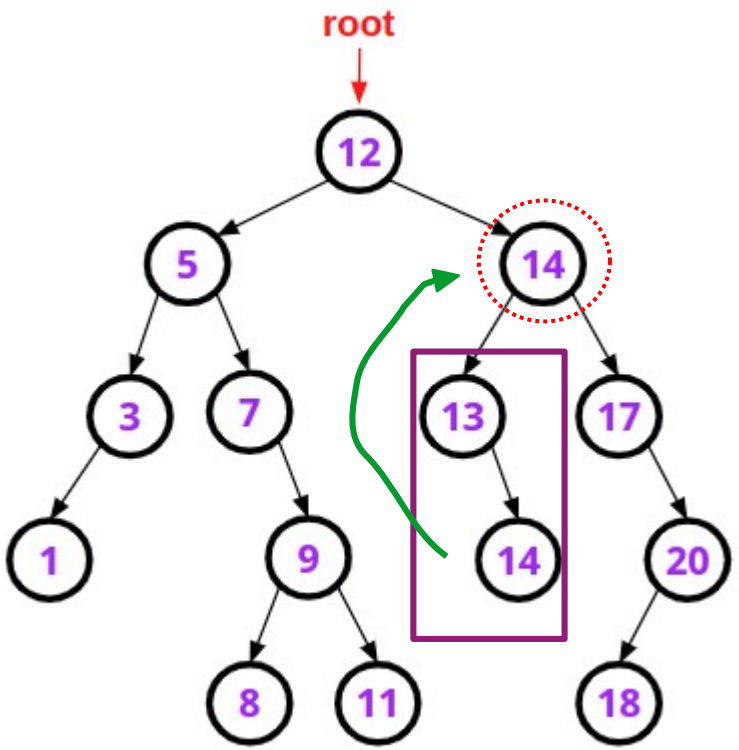
      Caindo em algum dos dois casos vistos anteriormente:
      - **1° Caso: Nó folha**
      - **2° Caso: Nó com um filho**

BST – Remoção de um Nó



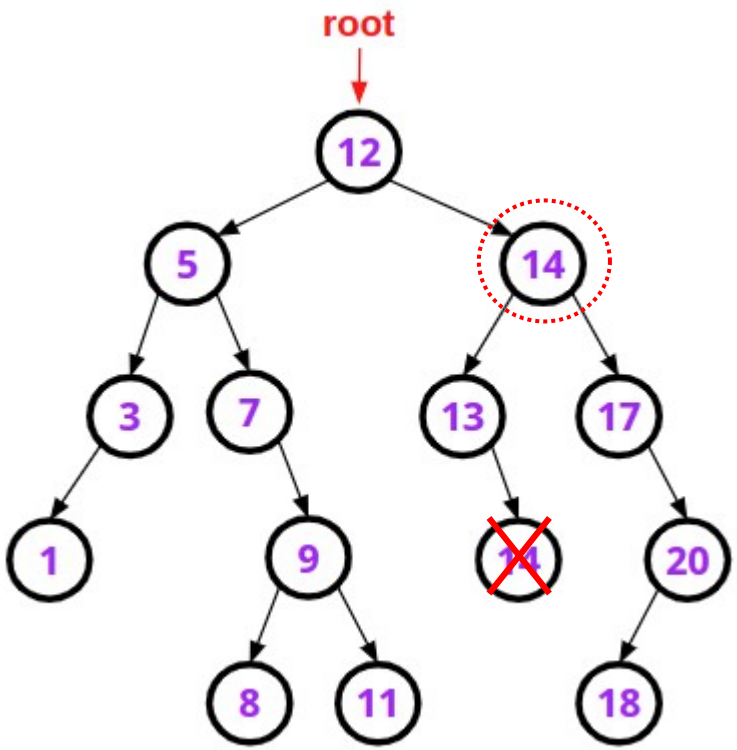
- **3° Caso: Remoção de um nó com 2 filhos**
  - Vamos tentar remover o **nó 15**. Quem deve tomar o lugar dele, que manterá a propriedade BST?
  - **Temos duas opções:**
    - 3.1 Min(sub-árvore da direita)
    - 3.2 Max(sub-árvore esquerda) = **14**

BST – Remoção de um Nó



- **3° Caso: Remoção de um nó com 2 filhos**
  - Vamos tentar remover o **nó 15**. Quem deve tomar o lugar dele, que manterá a propriedade BST?
  - **Temos duas opções:**
    - 3.1 Min(sub-árvore da direita)
    - 3.2 Max(sub-árvore esquerda) = **14**
      - **14 vai para lugar do 15 e cai no Caso 1: exclusão de folha**

BST – Remoção de um Nó



- **3° Caso: Remoção de um nó com 2 filhos**
  - Vamos tentar remover o **nó 15**. Quem deve tomar o lugar dele, que manterá a propriedade BST?
  - **Temos duas opções:**
    - 3.1 Min(sub-árvore da direita)
    - 3.2 Max(sub-árvore esquerda) = **14**
      - **14 vai para lugar do 15 e cai no Caso 1: exclusão de folha**

BST – Remoção de um Nó

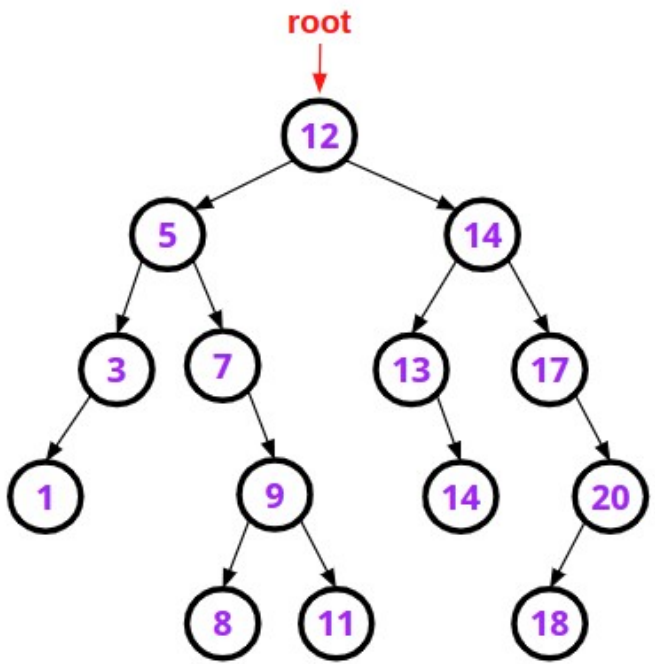
Eliminação

A estratégia global para eliminar um nó z de uma árvore de busca binária T tem três casos básicos, mas, como veremos, um deles é um pouco complicado.

- Se z não tem nenhum filho, então simplesmente o removemos modificando seu pai de modo a substituir z por NIL como seu filho.
- Se o nó tem apenas um filho, então elevamos esse filho para que ocupe a posição de z na árvore modificando o pai de z de modo a substituir z pelo filho de z.
- Se z tiver dois filhos, encontramos o sucessor de z, y, que deve estar na subárvore direita de z, e obrigamos y a tomar a posição de z na árvore. O resto da subárvore direita original de z tornase a nova subárvore direita de y, e a subárvore esquerda de z tornase a nova subárvore esquerda de y. Esse é o caso complicado porque, como veremos, o fato de y ser ou não o filho à direita de z é importante.

Cap 12.3 - Algoritmos - Teoria e Prática – Cormen et. al. – 3 Edição

BST – Remoção de um Nó - Resumo



- **1° Caso: folha → Deleta folha**
- **2° Caso: 1 filho → Liga nó anterior (seu pai) com seu filho**
- **3° Caso: 2 filhos →** encontrar o **min da sub-árvore a direita** ou o **max da sub-árvore a esquerda**, substituir. E por recursão cair em alguns dos outros 2 casos mais triviais.

BST – Remoção de um Nó : Implementação

Let's code </>...