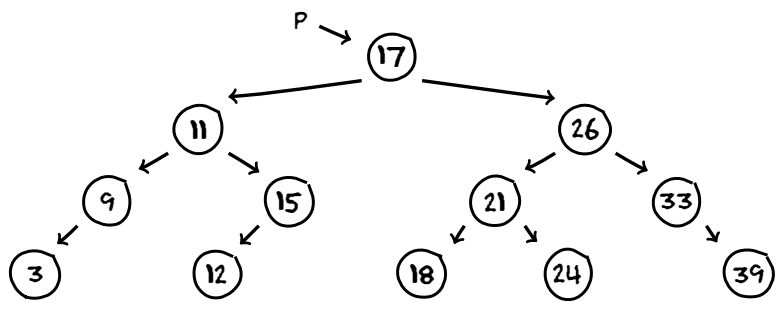
Outro algoritmo de remoção

Imagine que o ponteiro p aponta para uma árvore binária de busca

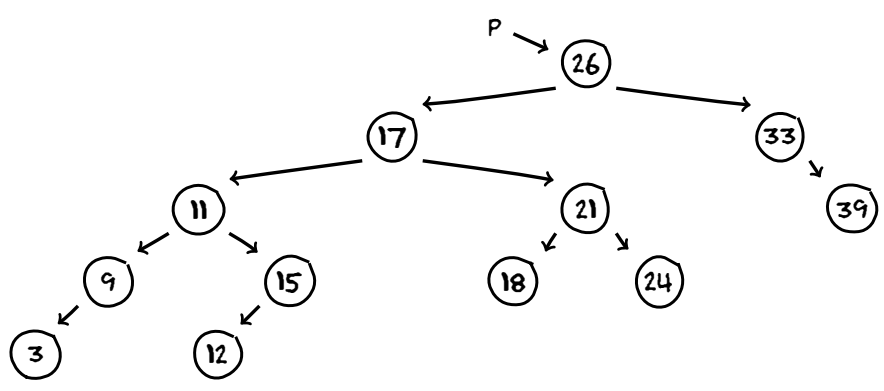


E imagine que nós queremos remover o 17 da árvore.

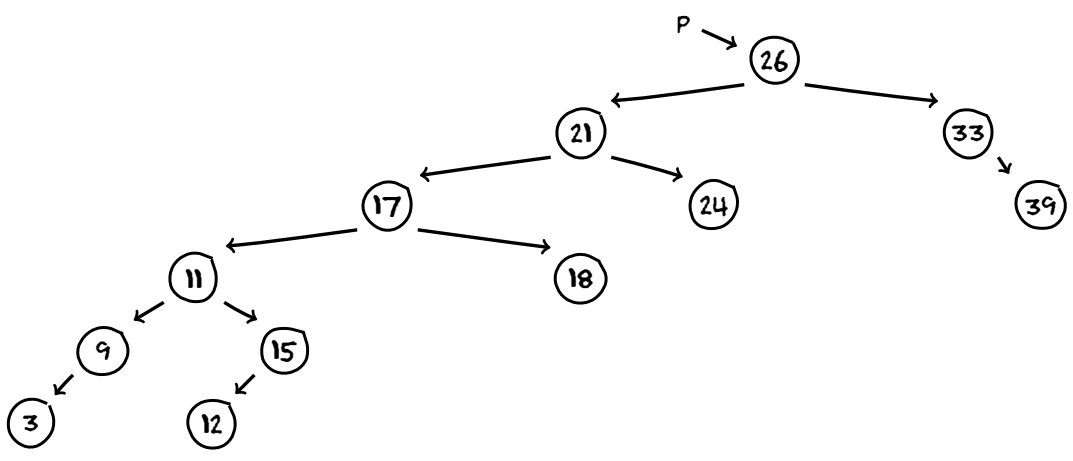
Mas, nós não queremos colocar o menor elemento da subárvore direita no lugar dele.

Como é que a gente faz isso?

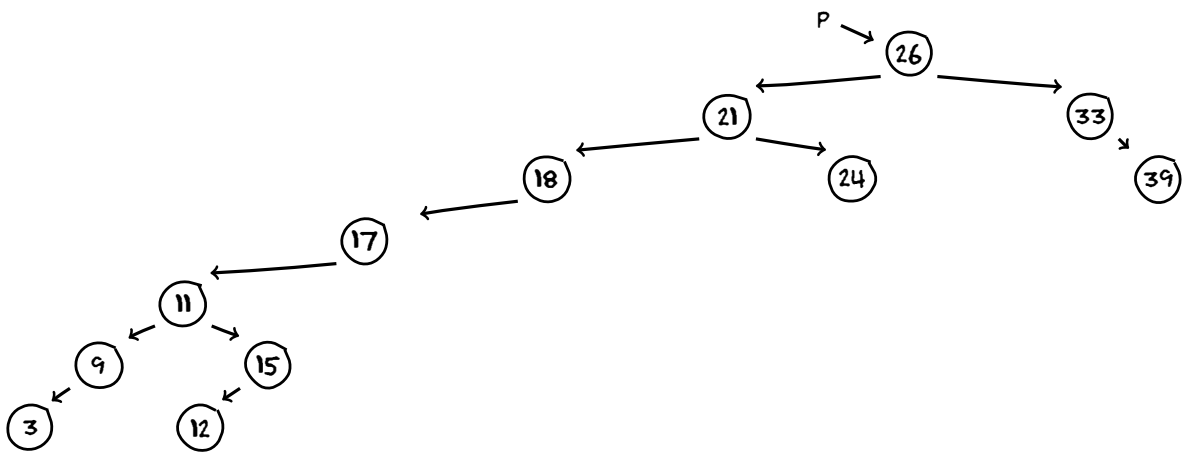
Bom, uma ideia consiste em fazer o 26 tomar o lugar do 17 na raiz da árvore



Daí, nós podemos fazer o 21 tomar o lugar do 17

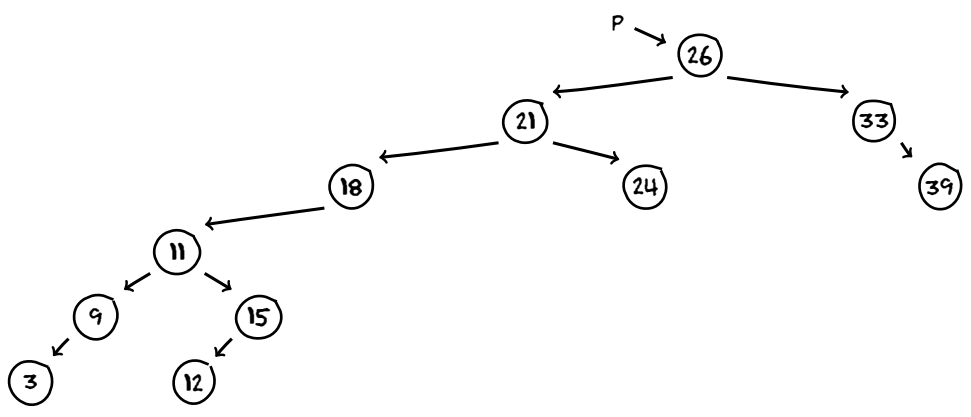


E depois, nós colocamos o 18 no lugar do 17



Agora, o 17 não tem mais um filho direito.

Logo, ele pode ser removido da árvore colocando o filho esquerdo no seu lugar



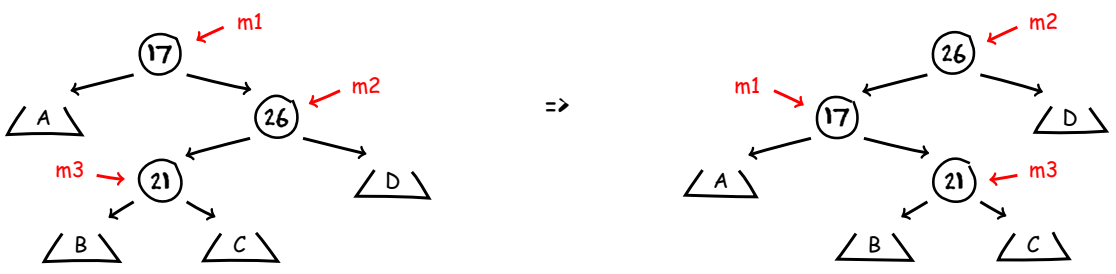
Sim! nós descobrimos outro algoritmo para fazer a remoção na árvore binária de busca.

E a sua ideia é bem simples

1. Fazer o filho direito tomar o lugar do elemento, até que não exista mais um filho direito.
2. E daí, remover o elemento da árvore

A operação importante acontece no passo 1: *o filho toma o lugar do pai.*

E nós observamos que ela pode ser implementada com a ajuda de 3 mãozinhas



— (*e apenas 2 instruções*)

Agora, nós podemos colocar essa operação dentro de uma função auxiliar

```
func SFD (q)                                                    // Sobe Filho Direito
      Se (q == Nulo   OU   q->dir == Nulo)
            Retorna ( q )
      Seão
            m1 = q;   m2 = m1->dir;   m3 = m2->esq            // coloca as mãozinhas
            m2->esq = m1;   m1->dir = m3                      // e rearranja
            Retorna (m2)
```

E a função de remoção fica assim

```
func remoção-Rec (q,k)
      Se (q == Nulo)   Retorna ( q )                          // já removi
      Sse (q->num > k)   q->esq = remoção-Rec ( q->esq,k )    // descendo ...
      Sse (q->num < k)   q->dir = remoção-Rec ( q->dir,k )    // descendo ...
      Sse (q->dir  ==  Nulo)
            fe = q->esq;   free(q)                            // remoção do elem
            Retorna (fe)
      Senão
            q = SFD(q)                                        // sobe o filho
            q->esq = remoção-Rec ( q->esq,k )                 // descendo ...
```