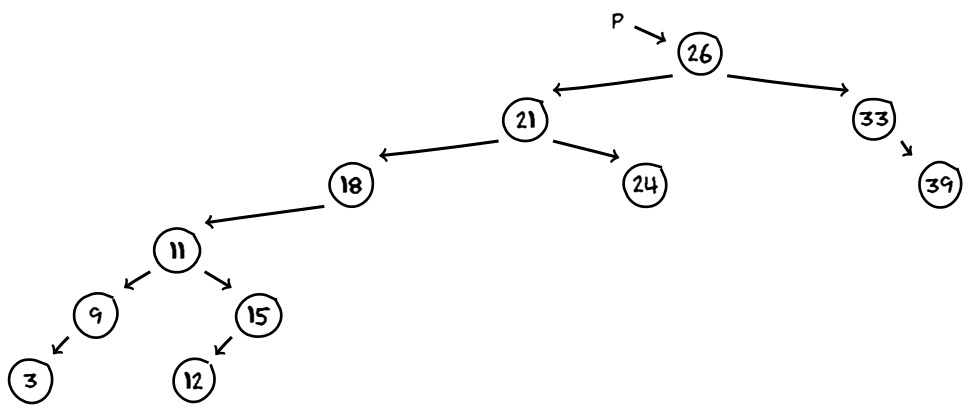
Subindo até a raiz

O truque que nós acabamos de ver é legal.

Mas ele deixa a árvore completamente desbalanceada



Só que isso é uma coisa fácil de resolver.

Quer dizer, olhando para a árvore nós vemos que

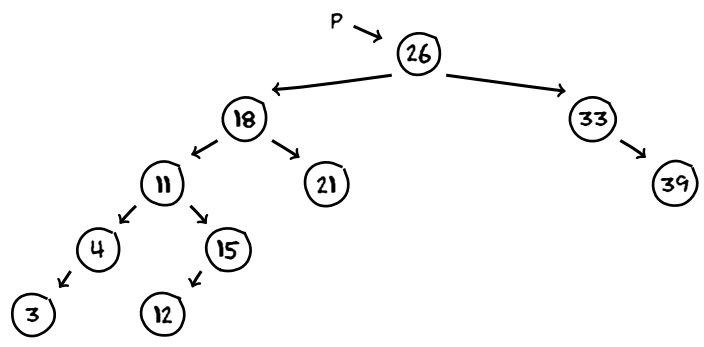
- existem 5 elementos maiores que o 18
- existem 4 elementos menores que o 18

Então, parece uma boa ideia colocar o 18 na raiz da árvore.

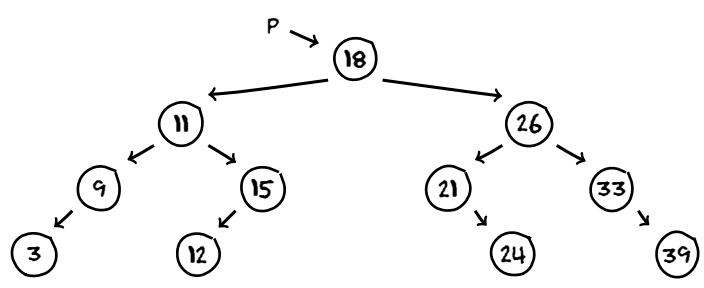
Mas, como é que a gente faz isso?

Bom, basta fazer o 18 tomar o lugar do seu pai sucessivamente, até ele chegar na raiz da árovre.

Primeiro o 18 toma o lugar do 21



E depois o 18 toma o lugar do 26



— (*você viu o que aconteceu, no final das contas?*)

Para implementar essa ideia, basta escrever a função que faz o filho esquerdo tomar o lugar do pai


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

E agora, nós estamos prontos para escrever a função que encontra um elemento e faz ele subir até a raiz

```
func SAR (q,k)                                                  // Sobe Até a Raiz
      Se (q == Nulo)   Retorna ( Nulo )                         // Não está lá ...
      Sse (q->num > k)
            aux = SAR (q->esq,k)
            Se (aux != Nulo)
                  q->esq = aux;   q = SFE (q)
                  Retorna (q)
      Sse (q->num < k)
            aux = SAR (q->dir,k)
            Se (aux != Nulo)
                  q->dir = aux;   q = SFD (q)
                  Retorna (q)
      Senão   Retorna (q)
```