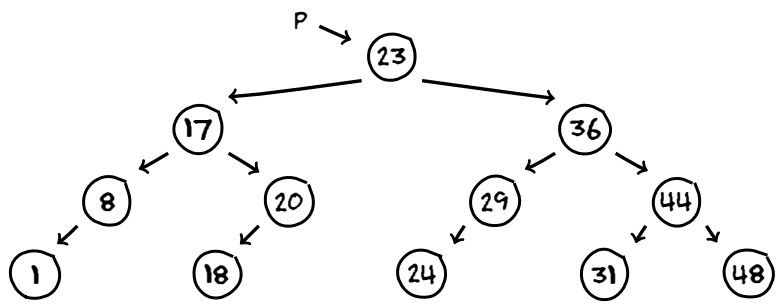
Cortando no nível k

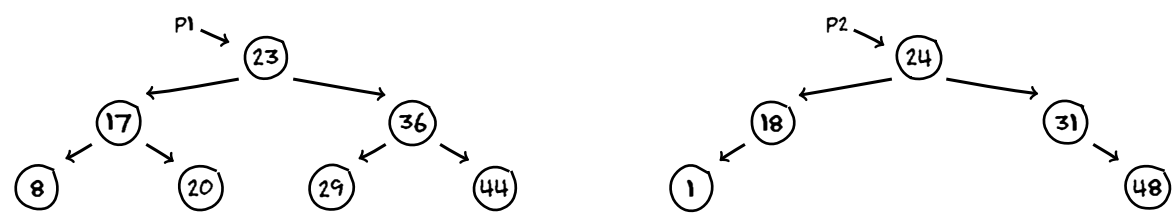
Imagine que o ponteiro `p` aponta para uma árvore binária de busca



A tarefa é

→ *remover todos os elementos que estão abaixo do nível* `k`
colocando os elementos removidos em outra árvore

No exemplo acima, para `k = 3`, isso nos daria

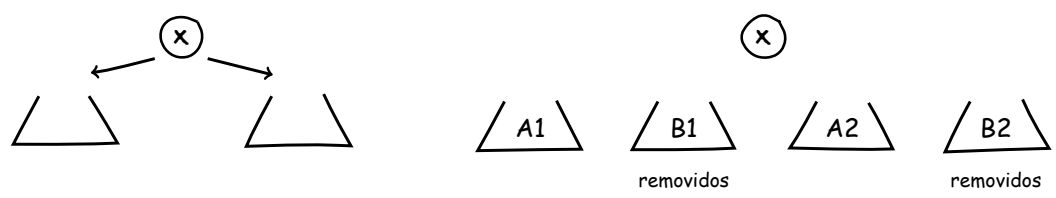


Certo.

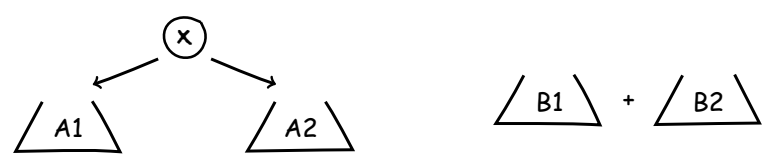
Mas como é que a gente faz isso?

Bom, a gente faz isso raciocinando recursivamente.

Quer dizer, imagine que os elementos abaixo do nível `k` nas subárvores esquerda e direita já foram removidos — (*e já foram colocados em subárvores separadas*)



Então, a solução do problema fica assim



E a gente usa o truque do exemplo anterior para juntar as árvores B_1 e B_2

Além disso, nós também utilizamos um parâmetro para indicar o nível em que a gente está na recursão.

A coisa fica assim

```
func CNNk-Rec (q,k,nivel)                                  // Corte No Nível K
      Se (q == Nulo)   Retorna ( Nulo, Nulo )
      Se (nivel > k)   Retorna ( Nulo, q )
      Senão
            a1,b1 = CNNK-Rec (q->esq,k,nivel+1)
            a2,b3 = CNNK-Rec (q->dir,k,nivel+1)
            q->esq,dir = a1,a2
            r = juntaArv (b1,b2)
            Retorna (q,r)
```

◇