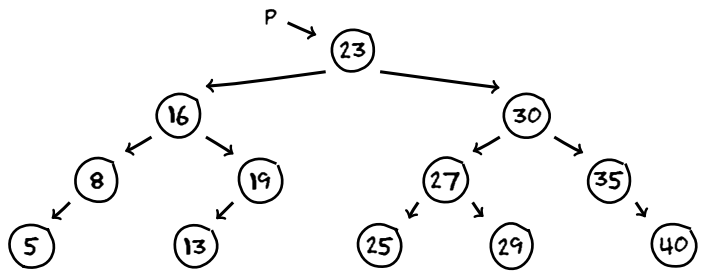
## Partição da árvore

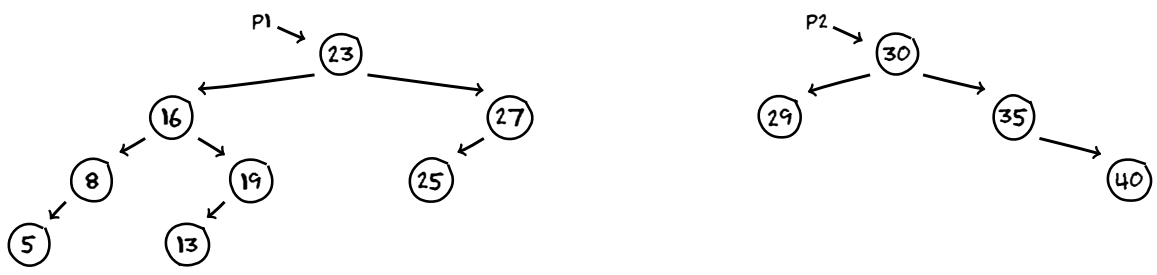
Imagine que o ponteiro `p` aponta para uma árvore binária de busca




A tarefa é

→ *particionar a árvore em duas, onde a primeira contém os elementos* $\leq$ `k`*,*
*e a segundo conté os elementos* $>$ `k`

No exemplo acima, para `k` = 28, isso nos daria




Legal.

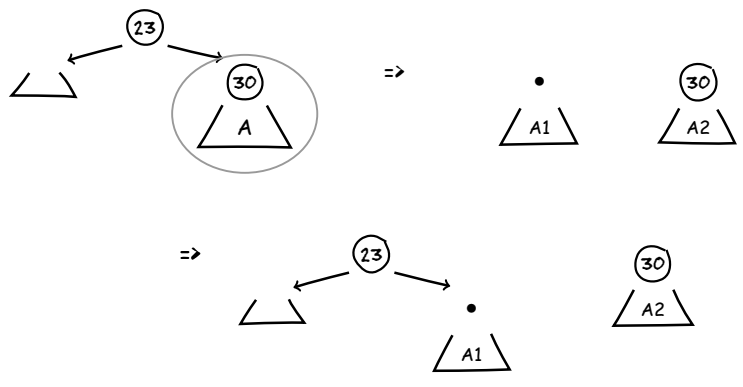
*Mas, como é que a gente faz isso?*

Bom, vamos começar olhando para a raiz.

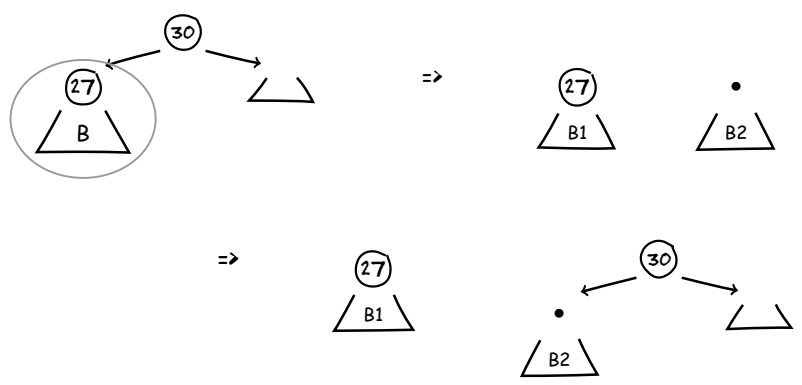
Nesse caso, como a raiz é igual a 23, ela vai ficar na primeira parte da árvore — (*juntamente com a sua subárvore esquerda*)

Daí, a gente particiona a subárvore direita.

A primeira subárvore dessa partição vai ser o filho direito do 23 — (*e a recursão vai acontecer na segunda parte*)




E daí, a gente nota que a partição da subárvore do 30 é análoga




Agora, já não é difícil escrever a função

```
func partição-Rec (q,k)
      Se (q == Nulo)   Retorna ( Nulo, Nulo )
      Sse (q->num ≤ k)
            p1 = q
            a1,a2 = partição-Rec (q->dir,k)
            p1->dir = a1;   p2 = a2
      Senão
            p2 = q
            b1,b2 = partição-Rec (q->esq,k)
            p2->esq = b2;   p1 = b1
      Retorna (p1,p2)
```

◇