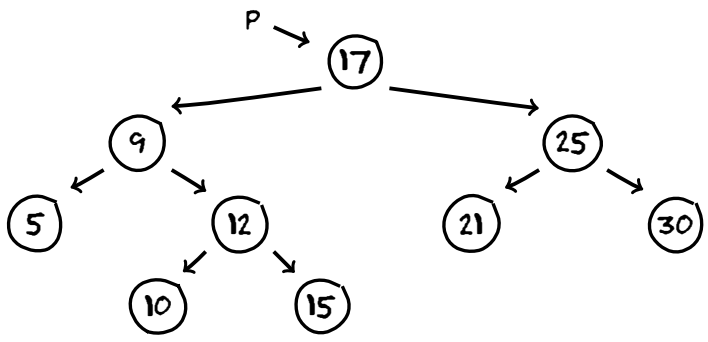
O menor maior do que k

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



A tarefa é

→ *encontrar o menor elemento maior do que k*

No exemplo acima, para `k = 10`, isso nos daria

=¿ 12

Certo.

Para encontrar o menor elemento que é maior do que `k`, nós vamos fazer a descida na árvore

- o ponteiro `mmk` aponta para o candidato que nós temos no momento — (*que pode ser* `Nulo`)
- e o ponteiro `q` vai em busca de um candidato melhor

No exemplo acima, a raiz da árvore (17) já é um candidato a ser o menor elemento maior do que `k` (10) — (*e o ponteiro* `mmk` *aponta para ela*)

Daí, a gente nota que não vai encontrar nenhum candidato melhor no lado direito — (*porque?*)

Então, o ponteiro `q` vai para o lado esquerdo.

A seguir, a gente vê que o elemento apontado por `q` (9) não é maior do que `k` (10).

Logo, ele não é um candidato, e os elementos à sua esquerda também não são.

Então, o ponteiro `q` vai para a direita.

A seguir, a gente vê que o elemento apontado por `q` (12) é maior do que o `k`.

Isso nos permite atualizar o ponteiro `mmk`.

E a gente nota que não vai encontrar nenhum candidato melhor no lado direito — (*porque?*)

Então, o ponteiro `q` vai para o lado esquerdo.

A seguir, a gente vê que o elemento apontado por `q` (10) não é maior do que o `k` (10).

Logo, ele não é um candidato, e os elementos à sua esquerda também não são.

Então, o ponteiro `q` vai para a direita.

Só que, não tem ninguém do lado direito.

Logo, a coisa acaba aqui.

E o código da função fica assim

```
func MmK-Arv (p,k)                                    // Maior menor que k

      mmk = Nulo;   q = p

      Enquanto (q != Nulo)
            Se (q->num ≥ k)   q = q->dir
            Senão
                  mmk = q
                  q = q->esq

      Retorna (mmk)
```

◇