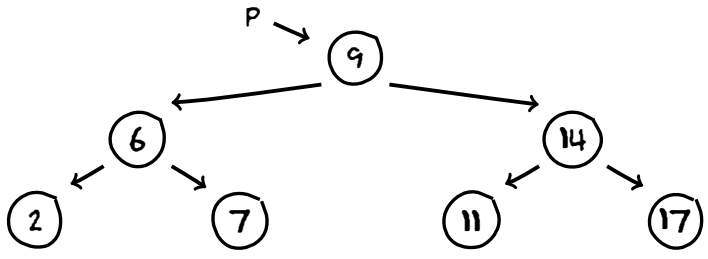
A operação de remoção

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



A tarefa é

→ *remover o elemento* `k` *da árvore*

Bom, nós já sabemos que essa tarefa envolve a remoção do menor elemento de uma subárvore.

Então nós começamos escrevendo a função que faz isso

```
func removeMenor-Arv (q)

      m = q;   mant = Nulo                          // descida pela esquerda
      Enquanto ( m->esq != Nulo )
            mant = m;   m = m->esq

      fd = m->dir;   fd->pai = mant                 // remoção do menor
      m->pai,dir = Nulo,Nulo

      Se (mant == Nulo)       Retorna (fd,m)
      Senão                      mant->esq = fd;   Retorna (q,m)
```

Nota: Dessa vez, nós não estamos fazendo a atualização das alturas — (*para simplificar as coisas*)

Agora, a gente imagina que também já tem a função de busca — (*que retorna um ponteiro para o elemento* `k`*, ou* `Nulo` *se ele não está lá*)

E daí, a coisa fica assim

```
func remoção-Arv (p,k)

      r = busca-Arv (p,k)                           // encontra o elemento
      Se (r == Nulo)    Retorna (p,Nulo)

      Senão
            Se (r->dir == Nulo)                     // filho esq.  no lugar do elemento
                  fe = r->esq;   fe->pai = r->pai
                  r->esq,pai = Nulo,Nulo
                  Se (fe->pai == Nulo)          Retorna (fe,r)
                  Sse ((fe->pai)->num > k)     (fe->pai)->esq = fe
                  Senão                            (fe->pai)->dir = fe



            Senão                                 // menor à dir.  no lugar do elemento
                  fe = r->esq;   pr = r->pai
                  fd,m = removeMenor-Arv (r->dir)
                  m->esq,pai,dir = fe,pr,fd
                  Se (pr == Nulo)                Retorna (m,r)
                  Sse (pr->num > k)             pr->esq = m
                  Senão                            pr->dir = m

      Retorna (p,r)
```

◇