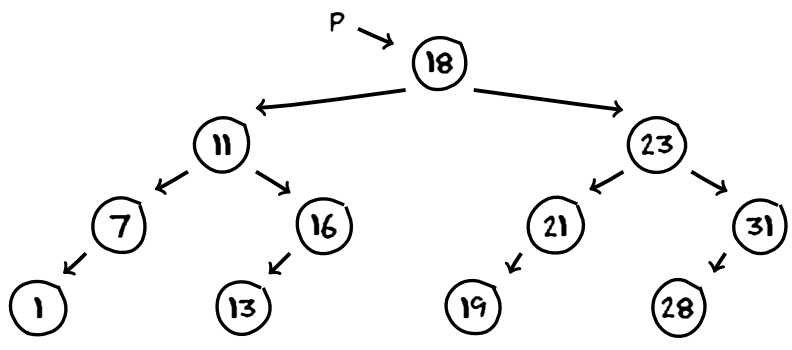
Navegando a árvore com o dedo

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



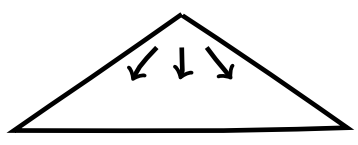
Imagine que a tarefa é

→ *imprimir todos os elementos da árvore*

E imagine que nós queremos realizar a tarefa com um algoritmo iterativo — (*com um dedo que percorre a árvore*)

Como é que a gente faz isso?

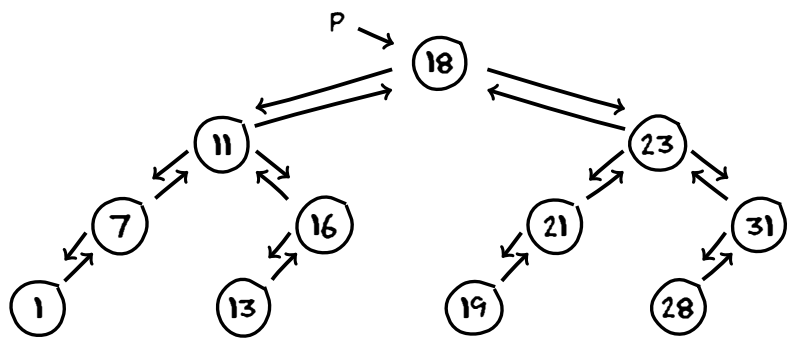
Bom, o primeiro problema é que a árvore só tem ponteiros para baixo



Logo, depois que o dedo desce, ele não tem como subir.

Só que, isso é fácil de resolver.

Quer dizer, basta fazer com que cada elemento aponte para o seu pai — (*e a raiz apoonte para* `Nulo`)



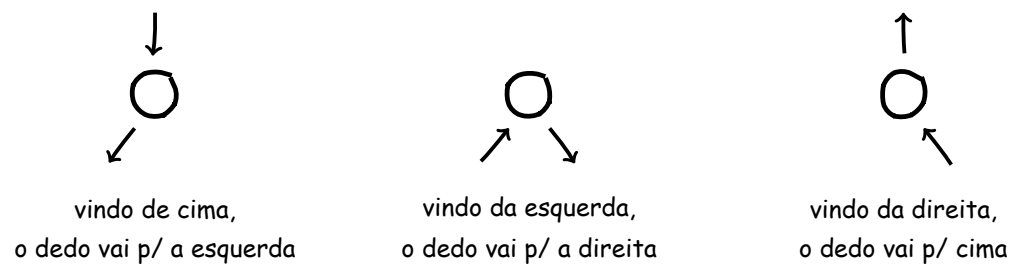
Mas isso ainda não resolve todos os problemas.

Quer dizer, na lista encadeada o dedo só pode ir pra frente e pra trás.

Mas na árvore existem várias maneiras de descer.

E na hora de subir, é preciso saber se a gente vai continuar subindo, ou se a gente vai descer para o outro lado.

Na prática, existem 3 casos



Então, se a gente sabe daonde o dedo está vindo, a gente também sabe pra onde o dedo tem que ir.

E para implementar essa ideia, a gente utiliza dois ponteiros

- q: o dedo que percorre a árvore
- qant: o lugar onde o dedo estava no passo anterior

No início, nós temos

q = p e qant = Nulo

E o processo continua enquanto o q é diferente de `Nulo`.

A coisa fica assim

```
func percorre-Arv (p)

      q = p;   qant = Nulo
      Enquanto ( q != Nulo )
            Se (qant == q->pai)                          // Caso 1:  vindo de cima
                  qant = q
                  Se (q->esq != Nulo)    q = q->esq
                  Sse (q->dir != Nulo)   q = q->dir
                  Senão                     q = q->pai
            Se (qant == q->esq)                          // Caso 2:  vindo da esquerda
                  qant = q
                  Se (q->dir != Nulo)   q = q->dir
                  Senão                     q = q->pai
            Senão                                        // Caso 3:  vindo da direita
                  qant = q;   q  = q->pai
```

Legal.

Mas essa função não faz nada — (*ela só percorre a árvore*)

Só que, é bem fácil modificá-la para que ela faça alguma coisa.

Quer dizer, a gente quer imprimir os elementos da árvore.

E quer fazer isso em ordem crescente.

Então, cada elemento deve ser impresso depois que a sua árvore esquerda já foi percorrida.

Isso significa que nós vamos imprimir o elemento quando nós estivermos subindo pelo filho esquerdo — (*i.e., no caso 2*)

Ou então ao chegar no elemento, quando ele não tiver um filho esquerdo — (*i.e., no caso 1*)

A coisa fica assim

```
func imprime-Arv (p)

      q = p;   qant = Nulo
      Enquanto ( q != Nulo )
            Se (qant == q->pai)                          // Caso 1:  vindo de cima
                  qant = q
                  Se (q->esq != Nulo)    q = q->esq
                  Senão
                        Imprime (q->num)
                        Se (q->dir != Nulo)   q = q->dir
                        Senão                    q = q->pai
            Se (qant == q->esq)                          // Caso 2:  vindo da esquerda
                  Imprime (q->num)
                  qant = q
                  Se (q->dir != Nulo)   q = q->dir
                  Senão                     q = q->pai
            Senão                                        // Caso 3:  vindo da direita
                  qant = q;   q  = q->pai
```