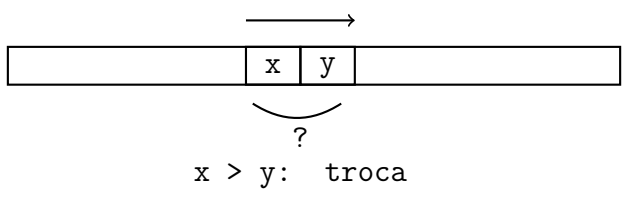
A operação de varredura

A operação de varredura consiste em percorrer uma lista comparando elementos consecutivos, e fazendo a troca sempre que o elemento da esquerda é maior do que o elemento da direita

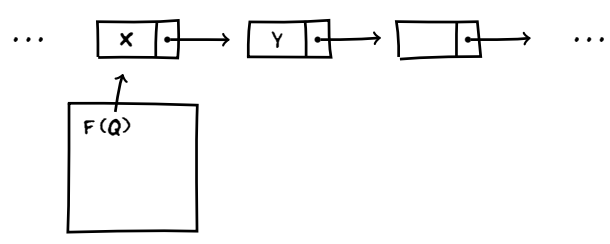


— (*como todo mundo sabe*)

Agora, nós queremos implementar esse procedimento de maneira recursiva.

Bom, a ideia é fazer a coisa na ida.

Quer dizer, cada chamada recursiva vai examinar o seu elemento e o próximo



Daí, se o seu elemento for maior que o próximo, ela faz a troca de posição entre os dois.

E depois, ela faz a chamada recursiva sobre o próximo elemento.

Finalmente, a chamada que vê o último elemento encerra o processo — (*porque ela só tem um elemento para inspecionar*)

A coisa fica assim

```
func varredura-Rec (q)

      Se (q == Nulo  OU  q->prox == Nulo)
            Retorna (q)
      Senão
            Se (q->num  >  (q-prox)->num)
                  m1 = q->prox;    m2 = m1->prox            // segura
                  q->prox = m2                              //
                  m1->prox = q                              // e troca
                  q = m1                                    //
            px = varredura-Rec (q->prox)
            q->prox = px                                    // o truque da re-conexão
            Retorna (q)
```

O algoritmo da bolha

Legal.

Mas agora que nós temos a função de varredura, nós queremos implementar a ordenação da bolha.

Bom, para fazer a ordenação da bolha basta fazer várias chamadas à função varredura, dentro de um laço.

Mas, nós não queremos utilizar um laço.

Nós queremos fazer a coisa de maneira recursiva.

E agora?

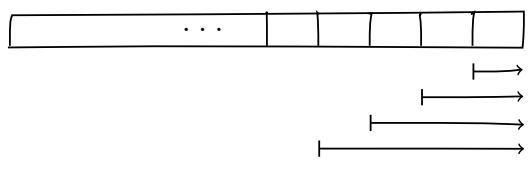
Bom, agora é a hora para um esperteza.

Quer dizer, a ideia é escrever uma função recursiva que vai até o fim da lista.

E daí, ela volta fazendo chamadas à função `varredura-Rec()`.

Mas porque isso funciona?

Bom, vamos ver o que acontece em sucessivas varreduras realizadas dessa maneira



- ia primeira varredura não faz nada — (*porque?*)
- daí, a segunda varredura ordena as duas últimas posições — (*porque?*)
- daí, a terceira varredura ordena as três últimas posições — (*porque?*)
- e por aí vai ...

No final, a lista inteira vai estar ordenada.

Legal, não é?

A coisa fica assim

```
func ordBolha-Rec (q)

      Se (q == Nulo)   Retorna
      Senão
            px = ordBolha-Rec (q->prox)             // vai até o fim
            q->prox = px
            q = varredura-Rec (q->prox)             // e volta fazendo varredura
            Retorna (q)
```

◇