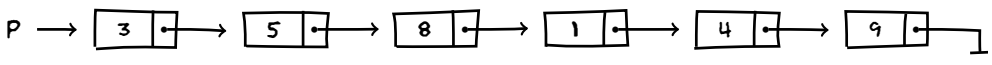
Invertendo a ordem da lista

Imagine que o ponteiro p aponta para uma lista encadeada



A tarefa é

→ *Inverter a ordem dos elementos da lista*

Bom, nós queremos fazer isso de maneira recursiva.

E a maneira mais fácil é fazer isso na ida.

Quer dizer, a lista invertida vai ser construída no parâmetro pi que é passado para a função.

A função `main()` inicializa o parâmetro pi como `Nulo`.

E cada chamada recursiva coloca o seu elemento no início da lista apontada por pi.

Daí, como a lista p é percorrida da esquerda para a direita, os elementos acabam ficando na ondem contrária.

Finalmente, a chamada que vê o `Nulo` retorna o ponteiro pi.

E todas as demais repassam esse ponteiro para a função `main()`.

A coisa fica assim

```
func inverte-Rec (q,pi)

      Se ( q == Nulo )   Retorna (q)
      Senão
            m = q->prox                              // segura o próximo elemento
            q->prox = pi                             // insere q em pi
            pi = q                                   //
            resp = inverte-Rec (m,pi)
            Retorna (resp)
```

Legal.

Mas, essa não é a única maneira de fazer as coisas.

Quer dizer, a gente também pode inverter a lista na volta.

Só que dessa vez, a gente não pode colocar os elementos no início da lista que está sendo construída.

Porque a lista p vai ser percorrida da direita para a esquerda.

E os elementos iam acabar ficando na mesma ordem que na lista p.

Então, os elementos vão ser inseridos no fim.

E para facilitar essa inserção, nós vamos ter um ponteiro pi para o primeiro elemento, e um ponteiro ui para o último elemento.

Esses dois ponteiros são passados como valor de retorno da função.

No início, a chamada que vê o `Nulo` retorna `Nulo,Nulo`.

E cada chamada recursiva adiciona o seu elemento no fim da lista apontada por pi, ui.

A coisa fica assim

```
func inverte2-Rec (q,pi)

      Se ( q == Nulo )   Retorna (Nulo,Nulo)          // começando a voltar
      Senão
            pi,ui = inverte2-Rec (q->prox)
            q->prox = Nulo
            Se (ui == Nulo)                          //
                  ui = q;   pi = q                   // insere no fim
            Senão                                    //
                  ui->prox = q;   ui = q             //

            Retorna (pi,ui)
```

E a função `main()` fica assim

```
func main ()

      pi,ui = inverte2-Rec (p)
      p = pi
```