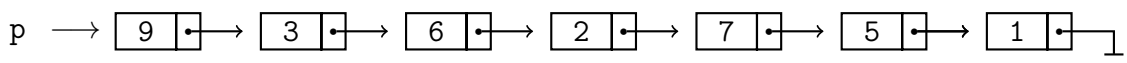
Imagine que o ponteiro p aponta para uma lista encadeada



A tarefa é

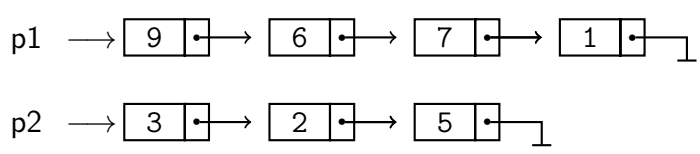
→ *separar os elementos da lista* p *em duas listas* p1 *e* p2
com a mesma quantidade de elementos (aproximadamente)

Bom, a tarefa não diz que nós precisamos quebrar a lista no meio.

Então, nós vamos percorrer a lista p e

- colocar os elementos das posições ímpares em p1
- colocar os elementos das posições pares em p2

No exemplo acima, isso nos daria

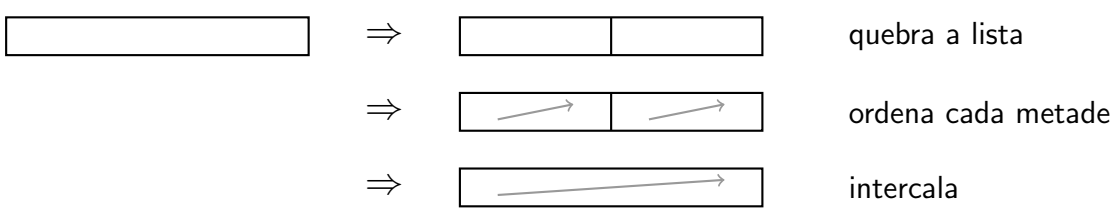


A coisa fica assim

```
func Quebra2 ( p )

      Se ( p == Nulo  OU  p->prox == Nulo )
      Retorna ( p, Nulo )

      pos = 1
      p1 = Nulo;    up1 = Nulo
      p2 = Nulo;    up2 = Nulo
      Enquanto ( p != Nulo )
            Se ( pos é impar )
                  Se ( p1 == Nulo )
                        aux = p;    ult = p )
                  Senão
                        up1->prox = p;    up1 = p )
                  p = p->prox;    up1->prox = Nulo
            Senão
                  Se ( p2 == Nulo )
                        p2 = p;    up2 = p2 )
                  Senão
                        up2->prox = p;    up2 = p )
                  p = p->prox;    up2->prox = Nulo
            pos++

      Retorna (p1,p2)
```

O algoritmo Mergesort

O algoritmo Mergesort ordena um vetor recursivamente, da seguinte maneira



Mas agora, nós já podemos fazer a mesma coisa com a lista encadeada.

A coisa fica assim

```
func Mergesort ( p )

      Se ( p == Nulo )   Retorna (p)

      p1,p2 = Quebra2 (p)

      p1 = Mergesort (p1)

      p2 = Mergesort (p2)

      p = Intercala (p1,p2)

      Retorna (p)
```

◇