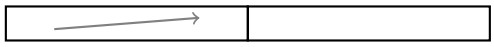
Mas, a gente pode pensar a mesma coisa de um jeito diferente.

Quer dizer, a gente pode organizar a computação de maneira recursiva.

A ideia é que, para ordenar uma lista de tamanho $n$

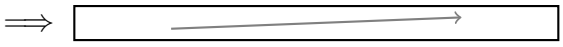
Para ordenar a lista com 4 blocos

1. primeiro a gente ordena a primeira metade



2. depois a gente a gente ordena a segunda metade

3. e daí, a gente faz a intercalação

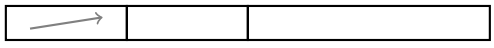


Só que a coisa é recursiva.

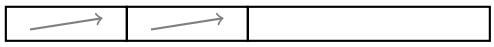
Quer dizer, a ordenação das metades acontece da mesma maneira.

Por exemplo, para ordenar a primeira metade da lista

1. primeiro a gente ordena a primeira metade da metade



2. depois a gente a gente ordena a segunda metade da metade



3. e daí, a gente faz a intercalação

E a recursão continua dividindo a lista, até chegar aos blocos de tamanho 1.

Daí que, no final das contas, só o que existe é intercalação.

A implementação recursiva da ordenação por intercalação é conhecida como o *algoritmo mergesort*

```
void  Mergesort ( L, inicio, fim )
 {
    Se ( inicio = fim )   Retorna            // blocos de tamanho 1

    meio  <--  ( inicio + fim ) / 2          // divide no meio

    Mergesort ( L, incio, meio )             // ordena a primeira metade

    Mergesort ( L, meio+1, fim )             // ordena a segunda metade

    Intercalação ( L, inicio, meio, fim )    // intercala
 }
```

Como esse é essencialmente o algoritmo que nós vimos na seção anterior, o seu tempo de execução é $O(n \log n)$.