Legal.

Mas, quando a gente entende uma coisa, a gente sempre pode pensar em outro jeito de fazer a mesma coisa.

Quer dizer, a gente acabou de entender que

→ `Para ordenar uma lista, basta`

- `ordenar a primeira metade`
- `ordenar a segunda metade`
- `e depois fazer a intercalação`

Mas, porque é preciso fazer a intercalação?

Bom, isso é necessário porque

- os elementos grandes da 1a metade precisam ir para a 2a metade
- e os elementos pequenos da 2a metade precisam ir para a 1a metade

Mas,

- se a 1a metade só tiver elementos pequenos
- e a 2a metade só tiver elementos grandes

então, não é preciso fazer a intercalação ...

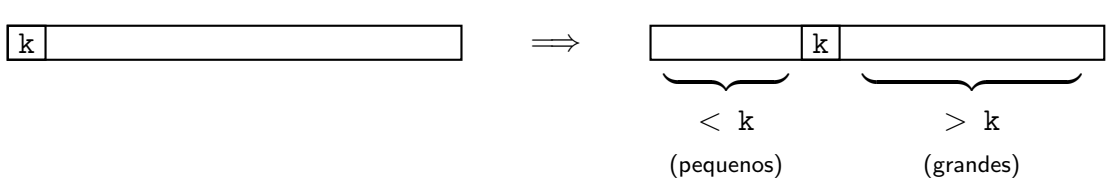
Hmm!

Isso significa que a gente também pode ordenar uma lista assim

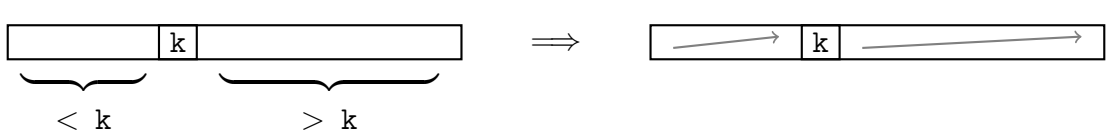
1. `primeiro, a gente separa os elementos pequenos e grandes`
2. `depois, a gente ordena o grupo dos pequenos`
3. `e depois, a gente ordena o grupo dos grandes`

Certo.

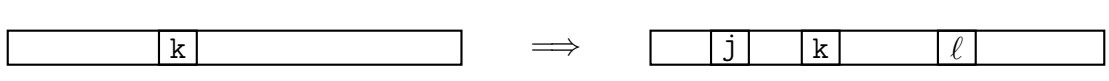
A primeira etapa é realizada pelo procedimento de *partição*



E as etapas 2 e 3 são feitas recursivamente.



Quer dizer, como a ordenação é recursiva, ela é feita por meio de novas chamadas ao procedimento de partição — (*até chegar aos blocos de tamanho 1*)



Daí que, no final das contas, a lista é ordenada apenas por meio de partições.

Quer dizer, agora nós temos um algortitmo de *ordenação por partição*.

A implementação recursiva dessa ideia é conhecida como o *algoritmo quicksort*.

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

    meio  <--  Partição ( L, inicio, fim )     // separa pequenos e grandes

    Quicksort ( L, inicio, meio-1] )           // ordena os pequenos

    Quicksort ( L, meio+1, fim] )              // ordena os grandes
 }
```

Análise

Mas ainda falta dizer uma coisa.

Quer dizer, em geral o procedimento `Partição()`não divide a lista exatamente na metade

`< figura >`

Daí que, em geral, o Quicksort não executa exatamente como o Mergesort.

Daí que, nós não temos a garantia de tempo $O(n \log n)$.

E agora vem uma má notícia ...

Quer dizer, no pior caso as partições acontecem sempre da pior maneira possível.

Por exemplo,

`< figura >`

— (*i.e., todos os elementos (exceto o pivô) ficam no mesmo lado da partição*)

Nesse caso, vão ocorrer n partições em sequência.

E como a partição leva tempo proporcional ao tamanho da lista, isso vai nos dar um total de

$$n \;+\; (n-1) \;+\; (n-2) \;+\; \ldots \;+\; 2 \;+\; 1 \qquad = \qquad O(n^2)$$

Arrrgh!!

Mas isso é apenas uma má notícia teórica.

Quer dizer, na prática isso nunca acontece.

E na prática, o Quicksort é muito, muito rápido — (*é por isso que ele tem esse nome ...*)

Veja só,

```
quicksort:  .....
mergesort:  .....
```

Legal, não é?