Nós acabamos de ver algoritmos de ordenação baseados em 3 ideias diferentes.

E nós também vimos que os 3 algoritmos executam em tempo $O(n^2)$.

Mas isso não significa que os 3 algoritmos executam exatamente no mesmo tempo.

Quer dizer, lembre que a nossa análise é aproximada.

Daí, faz sentido perguntar

- *Quem é o mais rápido ?*

Bom, uma maneira de decidir a questão consiste em escrever programas de computador que implementam os 3 algoritmos, e fazer uma bateria de testes.

Fazendo isso, a gente obtém o seguinte resultado

```
ordenação por seleção:  ????
   ordenação da bolha:  ????
ordenação por inserção:  ????
```

Quer dizer, a ordenação por inserção é a mais rápida de todas.

E a ordenação da bolha é a mais lenta de todas.

Mas daí alguém pode perguntar: *porque?*

Faz sentido começar comparando a ordenação por seleção e a ordenação da bolha, porque esses algoritmos são baseados nas funções

- `Maior()`
- `Varredura()`

que percorrem a lista inteira da esquerda para a direita.

Daí a gente observa que

- enquanto a função `Maior()` faz comparações com todos os elementos,
  em busca do maior deles
- a função `Varredura()` faz comparações com todos os elementos,
  e também troca um monte deles de lugar

Isso já explica porque a ordenação da bolha é mais lenta do que a ordenação por seleção.

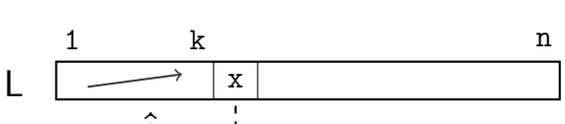
Certo.

Mas porque a ordenação por inserção é a mais rápida de todas?

A resposta é simples

→ *Porque a função* `Inserção()` *nem sempre percorre toda a porção ordenada da lista*

Quer dizer, se a posição correta do elemento da posição `k + 1` na porção ordenada `L[1..k]` é algum ponto no meio do caminho



então a função `Inserção()` só percorre essa parte do caminho.

E isso faz com que a ordenação por inserção seja a mais rápida de todas.

Legal, não é?