Considere a seguinte tarefa

→ *Verificar se o número x está na lista* `L` *ou não*

Na aula 00.1, nós resolvemos esse problema para o caso ordenado e o caso não-ordenado.

E aqui as coisas se complicam um pouquinho outra vez.

Quer dizer, os dois algoritmos utilizam o comando `Break`.

E isso significa que nós não sabemos quantas voltas os laços vão dar


```
// Caso não ordenado

Aux  <--  0
Para  i <-  1  Até  n
 {
    Se ( L[i] = x )
     {
        Aux  <--  i
        Break
 }   }
Se ( Aux != 0 )   Imprime("Sim");
Senão             Imprime("NÃo");
```

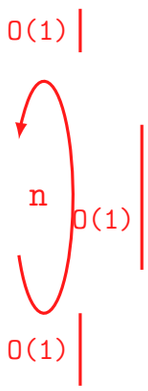

```
// Caso ordenado

Aux  <--  0
Para  i <-- 1  Até  n
 {
    Se ( L[i] = x )
     {
        Aux  <--  i;    Break
     }
    Se ( L[i] > x )   Break
 }
Se ( Aux != 0 )   Imprime("Sim");
Senão             Imprime("NÃo");
```

E agora?

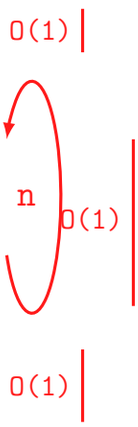
Bom, agora a gente lembra que a nossa análise é apenas mais ou menos.

Daí que a gente pode considerar aquilo que acontece no pior caso.

No pior caso, os laços dois dois algoritmo realizam n voltas — (*porque?*)




```
// Caso não ordenado

Aux  <--  0
Para  i <-  1  Até  n
 {
    Se ( L[i] = x )
     {
        Aux  <--  i
        Break
 }   }
Se ( Aux != 0 )   Imprime("Sim");
Senão             Imprime("NÃo");
```




```
// Caso ordenado

Aux  <--  0
Para  i <-- 1  Até  n
 {
    Se ( L[i] = x )
     {
        Aux  <--  i;    Break
     }
    Se ( L[i] > x )   Break
 }
Se ( Aux != 0 )   Imprime("Sim");
Senão             Imprime("NÃo");
```

Com bases nas anotações, nós chegamos ao seguinte resultado

- `caso não ordenado:` $O(n)$
- `caso ordenado:` $O(n)$

Quer dizer, o ganho de eficiência que a gente obteve no caso ordenado não aparece na análise.

Por um lado, a gente pode interpretar isso como uma limitação do nosso método de análise.

Por outro lado, a gente pode pensar que os ganhos de eficiência que a gente obtém só são significativos se eles aparecem na análise.

E isso também faz sentido.

◇