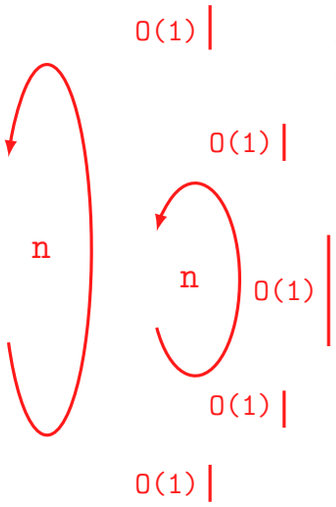
Imagine que $\mathsf{L}_1$ e $\mathsf{L}_2$ são duas listas de tamanho $n$.

E considere a tarefa

→ *Contar quantos elementos de* $\mathsf{L}_1$ *são maiores que todos os elementos de* $\mathsf{L}_2$

Na primeira aula nós vimos o seguinte algoritmo para esse problema

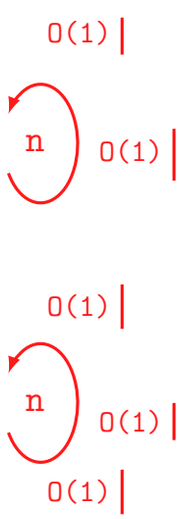



```
cont  <--  0
Para  k <-- 1  Até  n
 {
    x  <--  L1[k];   Aux  <--  0
    Para  i <-- 1  Até  n
     {
        Se ( L2[i] >= x )
         {
            Aux  <--  1;   Break
     }   }
    Se ( Aux = 0 )   cont ++
 }
Imprime ("A resposta é:", cont)
```

E com base nas anotações nós obtemos a seguinte estimativa para o tempo de execução do algoritmo

$$O(1) \quad + \quad n \cdot \Big( O(1) \quad + \quad n \cdot O(1) \quad + \quad O(1) \Big) \quad + \quad O(1) \qquad = \qquad O(n^2)$$

Mas nós também vimos um algoritmo que resolve o problema de maneira mais inteligente




```
// 1a ETAPA: encontrar o maior elemento de L2

M  <--  L2[1]
Para  i <-- 1  Até  n
 {
    Se ( L2[i] > M )    M  <--  L2[i]
 }

// 2a ETAPA: contar elementos de L1 maiores do que M
cont  <--  0
Para  k <-- 1  Até  n
 {
    Se ( L1[k] > M )   cont ++
 }
Imprime ("A resposta é:", cont)
```

Com base nas anotações nós obtemos a seguinte estimativa para o tempo de execução do algoritmo

$$O(1) \quad + \quad n \cdot O(1) \quad + \quad O(1) \quad + \quad n \cdot O(1) \quad + \quad O(1) \qquad = \qquad O(n)$$

Quer dizer, as análises nos deram os seguintes resultados

- `primeira versão:` $O(n^2)$
- `versão inteligente:` $O(n)$

de modo que a versão inteligente é realmente bem mais eficiente.

◇