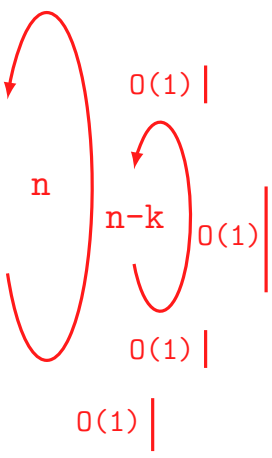
Considere a tarefa

→ *Verificar se a lista contém algum elemento repetido*

Na primeira aula nós apresentamos um algoritmo para esse problema.

E abaixo nós já temos esse código anotado




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

Aqui pela primeira vez nós temos um algoritmo com um laço dentro de um laço.

E para complicar as coisas, o número de voltas do laço mais interno depende da volta em que a gente está no laço mais externo.

Mas a gente pode simplificar as coisas assumindo que o laço mais interno sempre dá n voltas.

Nesse caso, a gente obtém a seguinte estimativa para o tempo de execução do algoritmo

$$
\begin{aligned}
& n \times \Big(O(1) \;+\; n \times O(1) \;+\; O(1) \Big) \;+\; O(1) \\
= \;& n \times O(1) \;+\; n^2 \times O(1) \;+\; n \times O(1) \;+\; O(1) \\
= \;& O(n) \;+\; O(n^2) \;+\; O(n) \;+\; O(1) \\
= \;& O(n^2)
\end{aligned}
$$

Note que dessa vez os termos $O(n)$ também foram descartados, porque na presença de $O(n^2)$ eles são termos menores.

Portanto a análise nos diz que o algoritmo executa em tempo $O(n^2)$.

Mas nesse ponto alguém pode perguntar

→ *Sera que ao reescrever $n-k$ como n a gente não acabou obtendo um resultado maior do que devia ?*

Bom, vamos raciocinar com mais precisão então.

Quer dizer, nós podemos observar que

- na primeira vez, o laço mais interno dá $n-1$ voltas
- na segunda vez, o laço mais interno dá $n-2$ voltas
- na terceira vez, o laço mais interno dá $n-3$ voltas
- e assim por diante ...

Daí que o número total de voltas realizadas pelo laço mais interno pode ser calculado assim

$$(n-1) \;+\; (n-2) \;+\; (n-3) \;+\; \ldots \;+\; 3 \;+\; 2 \;+\; 1$$

O truque mais conhecido para calcular essa soma é o seguinte

$$
\begin{array}{rcccccccccc}
 & (n-1) & + & (n-2) & + & \ldots & + & 2 & + & 1 & \\
+ & 1 & + & 2 & + & \ldots & + & (n-2) & + & (n-1) & \\
\hline
 & n & + & n & + & \ldots & + & n & + & n & = \quad n^2
\end{array}
$$

Quer dizer, duas vezes a soma é igual a n^2.

Logo a soma é igual a $n^2/2$.

Isso significa que a contribuição do laço mais interno para o tempo de execução do algoritmo é $O(n^2/2)$.

E isso já nos permite concluir que o algoritmo executa em tempo $O(n^2)$

— (*porque $O(n^2/2)$ é a mesma coisa que $O(n^2)$*)

Legal, não é?

◇