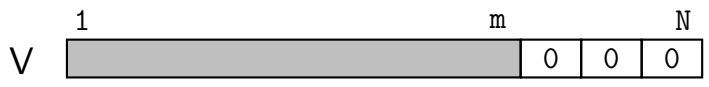
Imagine outra vez que nós temos uma lista com posições vazias no fim



A tarefa é

→ *Inserir o número* `k` *na lista*

Bom, se a lista não está ordenada, isso é a coisa mais fácil do mundo

- basta colocar o `k` na 1a posição vazia, e depois aumentar o valor de m

A coisa fica assim

```
V[m+1] = k
m++
```

E o tempo de execução é $O(1)$.

Legal.

Agora imagine que a lista está ordenada

									m			N
V	1	3	7	8	10	16	19	21	28	0	0	0

E imagine que `k` = 17.

Então, o `k` precisa ser colocado entre o 16 e o 19.

Só que não tem lugar ali.

Quer dizer, as posições vazias estão todas no final da lista.

E agora?

Bom, agora a gente desloca uma posição vazia até ali.

Ou melhor, a gente move todos os elementos a partir do 19 uma posição para a direita

									m			N
V	1	3	7	8	10	16	19	21	28	0	0	0

E daí, a gente pode colocar o `k` no lugar em que o 19 estava.

Certo.

Mas antes de tudo, é preciso descobrir o lugar onde o `k` vai entrar.

Abaixo nós temos o código completo (já anotado)

```
                // 1a Etapa:  encontrar o lugar do k

O(1)  i = 1                                   // coloca o dedo no inicio
      Enquanto ( i ≤ N )                      // e vai adiante ...
O(n)      Se ( V[i] > k  OU  V[i] = 0 )
O(1)          Quebra
          i++

                // 2a Etapa:  deslocar os elementos

O(1)  j = m                                   // coloca o dedo no fim
      Enquanto ( j ≥ N )                      // e vai em direção ao i
O(n)      V[j+1] = V[j]
O(1)      j--

                // 3a Etapa:  inserir o k na lista
O(1)  V[i] = k
      m++
```

Agora basta examinar as anotações para obter o tempo de execução do algoritmo

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

E aqui nós temos o primeiro exemplo onde a versão desordenada do problema é mais rápida.

◇