Considere, por exemplo, a situação em que os elementos de uma lista de tamanho n são exatamente os números $1, 2, \ldots, n$ (em uma ordem qualquer).

Nesse caso, basta examinar o valor de um elemento para saber qual é a sua posição correta na lista — isto é, não é preciso fazer nenhuma comparação.

Abaixo, nós temos um algoritmo que ordena uma lista desse tipo:

```
Procedimento ord-Permutação ( V[1..n] )

    k ← 1
    Enquanto ( k < n )
        Se ( V[k] ≠ k )       // Você está fora do seu lugar
            p ← V[k];   Troca (k,p)
        Senão
            k++
```

Qual o tempo de execução desse algoritmo?

Bom, a cada iteração do laço pode ocorrer uma das seguintes coisas:

- o valor de `k` é incrementado
- um elemento da lista é trocado de posição

É fácil ver que o valor de `k` só pode ser incrementado n vezes.

Por outro lado, sempre que um elemento é trocado de posição ele vai para a sua posição correta, e nunca mais sai de lá.

Isso significa que podem ocorrer no máximo n trocas durante a execução do algoritmo.

Essas duas observações nos permitem concluir que o laço executa no máximo

$$n \ + \ n \quad = \quad 2n \text{ iterações}$$

Portanto, esse algoritmo de ordenação executa em tempo $O(n)$.