Agora imagine que nós temos duas listas

U	5	3	9	7	8	10	13	1	2	14
V	4	8	2	11	6	5	15	3	19	1

A tarefa consiste em

→ *Imprimir todos os números que aparecem nas duas listas*

— *(*i.e., a operação de interseção*)*

Bom, as duas listas estão desordenadas.

Então, não há outra coisa a fazer senão

Verificar, para cada elemento de U, se ele aparece em V

Se aparecer imprime

Se não aparecer não imprime

A coisa fica assim

```
O(1)  i = 1                                  // põe o dedo na lista U
      Enquanto ( i ≤ N )                     // e vai até o fim
O(1)    j = 1;   achei = 0                   // põe o dedo na lista V
        Enquanto ( j ≤ N )                   // e vai até o fim
O(1)      Se ( V[j] == U[i] )
              achei = 1
              Quebra
          j = j + 1
O(1)    Se ( achei == 1 )
            Imprime ( U[i] )
        j = j + 1
```

O(n) O(n)

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

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

Legal.

Agora imagine que as listas estão ordenadas

U	1	2	3	5	7	8	9	10	13	14
V	1	2	3	4	5	6	8	11	15	19

Bom, a ideia aqui é colocar um dedo em cada lista, e verificar se os elementos são iguais.

Fazendo isso, a gente imprime 1, 2, 3 e chega na seguinte situação

U	1	2	3	5	7	8	9	10	13	14
				↑i						
V	1	2	3	4	5	6	8	11	15	19
				↑j						

Aqui a gente vê que o 4 é menor que o 5.

E daí, como as listas estão ordenadas, a gente já sabe que o 4 não está na lista U.

Então, a gente avança o dedo j.

A coisa continua assim até o fim.

E abaixo nós temos o algoritmo (já anotado)

```
O(1)  i = 1;   j = 1                         // um dedo em cada lista
      Enquanto ( i ≤ N  E  j ≤ N )
          Se ( U[i] == V[j] )
              imprime ( U[i] )
              i = i + 1;   j = j + 1         // avançam os dois dedos
O(1)      Sse ( U[i] < V[j] )
              j = j + 1                      // avança apenas o j
          Senão
              i = i + 1                      // avança apenas o i
```

O(n)

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

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

E aqui, mais uma vez, a gente vê que o algoritmo da versão ordenada é muito mais rápido.

◇