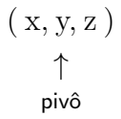
Mas pelo menos nós aprendemos alguma coisa.

Quer dizer, não é uma boa ideia contar as triplas engraçadas uma por uma.

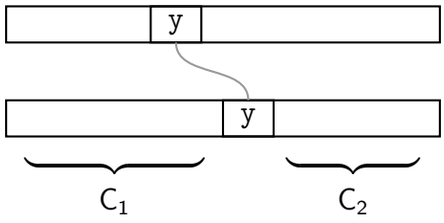
Então agora, nós vamos contar várias triplas engraçadas de uma só vez.

A ideia é considerar o elemento central da tripla como um pivô



Daí, a gente localiza o pivô nas duas listas.

E calcula o número de elementos comuns em U e W, à esquerda e à direita do pivô



Logo, o número de triplas engraçadas que tem y como pivô é igual a $\mathsf{C}_1 \times \mathsf{C}_2$.

Para implementar essa ideia, nós vamos percorrer a lista U tomando cada um dos seus elementos como pivô.

Daí, a gente localiza o pivô em W e faz a contagem dos elementos comuns.

A coisa fica assim

```
func contaTE-2.0 (U,W)

      cont ← 0

      Para i ← 2 Até N-1
            pivo ← U[i]
            j ← Localiza (pivo,W[1..N])
            C1 ← contaComuns (U[1..i-1],W[1..j-1])
            C2 ← contaComuns (U[i+1..N],W[j+1..N])
            cont ← C1*C2

      Retorna (cont)
```

O(n)

Abaixo nós temos as funções utilizadas no código

```
func Localiza (x,W[a..b])

      Para i ← a Até b
            Se ( W[i] = x )   Retorna(i)
      Retorna(-1)


func contaComuns (U[a..b],W[c..d])

      cont ← 0
      Para i ← a Até b
O(n) ←  Se ( Localiza (U[i],W[c..d]) != -1 )
                  cont++
      Retorna(cont)
```

O(n)

O(n)

Agora já não é difícil ver que a complexidade do nosso segundo algoritmo é

$$O(n) \quad \times \quad \Big[O(n) \quad + \quad O(n^2) \Big] \qquad = \qquad O(n^3)$$

Mas isso ainda é muito porcaria.