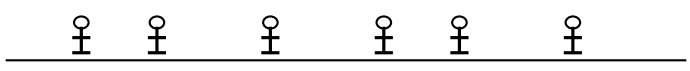
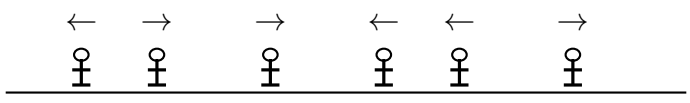
Contando encontros

Imagine que existe n pessoas sobre uma linha



E imagine que as posições das pessoas estão armazenadas no vetor `A[1..n]`.

Imagine que as pessoas vão andar para uma outra posição da linha — (*todas com a mesma velocidade*)



E imagine que as posições destino das pessoas estão armazenadas no vetor `B[1..n]`.

Daí, algumas dessas pessoas vão se encontrar.

E quando elas se encontram, elas se cumprimentam: "*Olá, bom dia!*".

O problema é

→ *Contar a quantidade de encontros que vão acontecer*

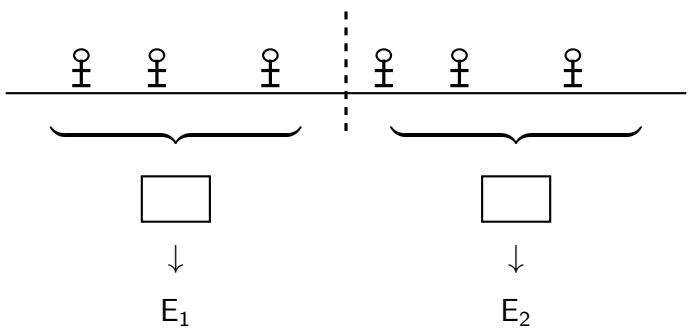
Certo.

É bem fácil resolver esse problema em tempo $O(n^2)$ — (*como?*)

Mas, $O(n^2)$ é uma porcaria ...

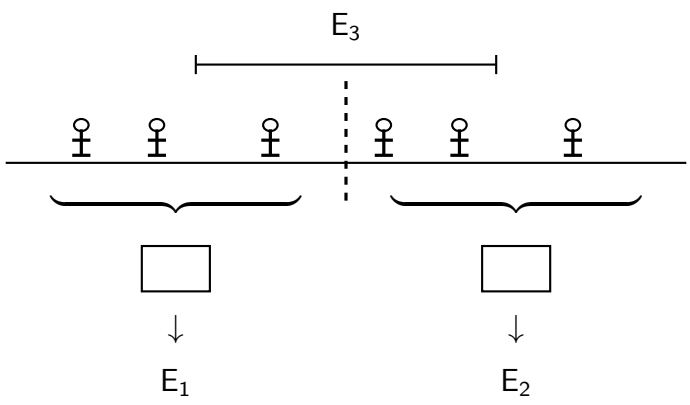
Então a gente vai tentar obter algo melhor com a divisão e conquista.

Quer dizer, imagine que a gente quebra a lista no meio, e resolve os problemas de cada lado — (*recursivamente*)



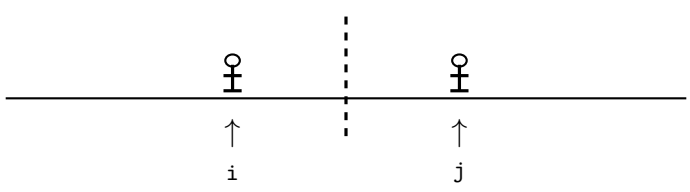
Isso, é claro, não resolve o nosso problema.

Porque ainda é preciso contar os encontros entre pessoas de lados diferentes



Então agora, a gente se concentra nesse problema.

Para isso a gente foca a atenção em uma pessoa de cada lado



E nota que, em qualquer caso, o encontro vai acontecer se

$$\texttt{B[i]} \quad > \quad \texttt{B[j]}$$

Então, para contar quantos encontros a pessoa `i` vai ter, basta contar quantas pessoas `j` do lado direito satisfazem a desigualdade acima.

Para fazer essa contagem de maneira eficiente, a gente ordena o lado direito da lista B.

E faz a contagem usando a busca binária.

A coisa fica assim

```
func encontrosMeio ( A, B, inicio, meio, fim )

    B-aux ← ordena ( B,meio+1,fim )

    C ← 0

    Para i ← meio Até inicio

        j ← buscaBin ( B[i], B-aux )

        C ← C + ( j − meio + 1 )

    Retorna ( C )
```

É bem fácil ver que esse algoritmo executa em tempo $O(n \log n)$.

Agora, nós podemos escrever o nosso algoritmo de divisão e conquista

```
T(n) =  func contaEncontros-DC ( A, B, inicio, fim )

        Se ( inicio = fim )     Retorna ( 0 )
O(1)
        meio ← ( inicio + fim ) / 2

T(n/2) ← E1 ← contaEncontros-DC ( A,B,inicio,meio )

T(n/2) ← E2 ← contaEncontros-DC ( A,B,meio+1,fim )

O(n log n) ← E3 ← encontrosMeio ( A,B,inicio,meio,fim )

        Retorna ( E1 + E2 + E3 )
```

Análise

Seguindo as anotações, nós obtemos a seguinte equação de recorrência para o tempo de execução do algoritmo

$$T(n) \quad = \quad 2 \cdot T\left(\frac{n}{2}\right) \quad + \quad O(n \log n)$$

O que nos dá outra vez

$$T(n) \quad = \quad O(n \log^2 n)$$