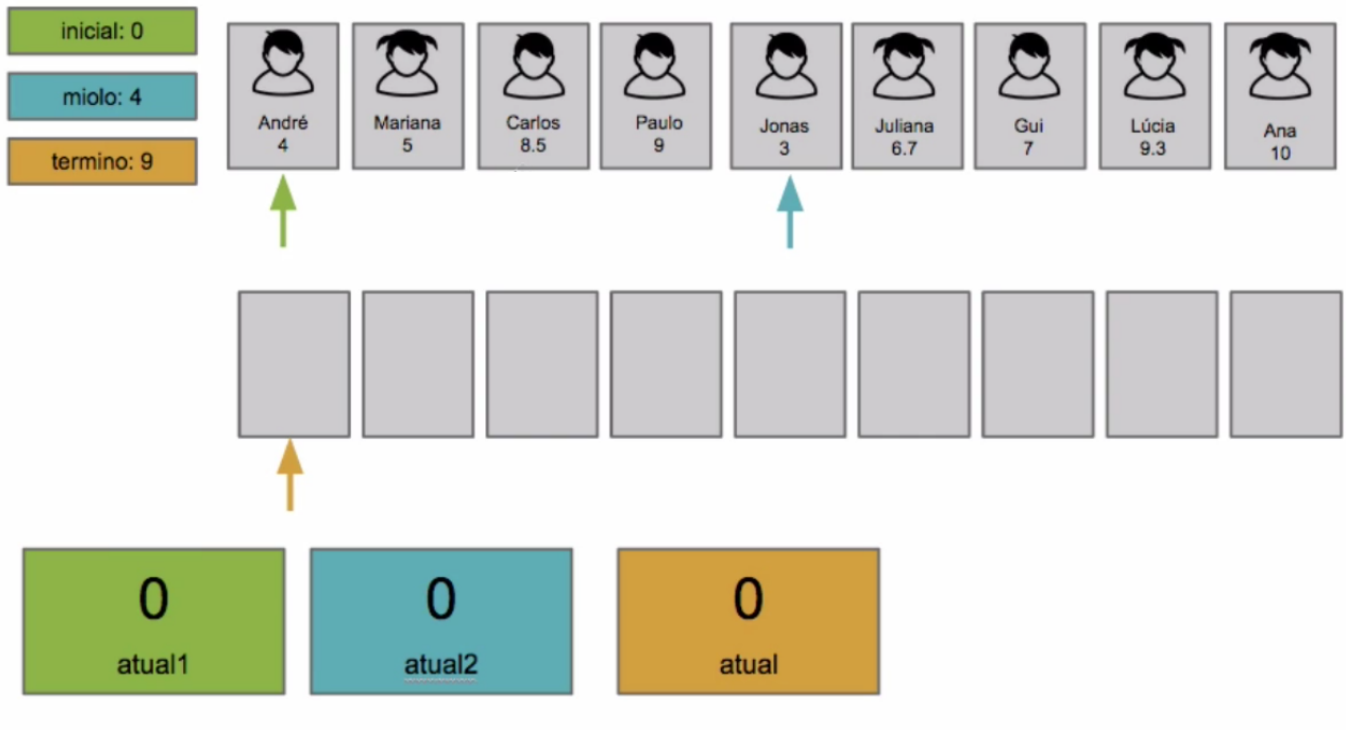
01

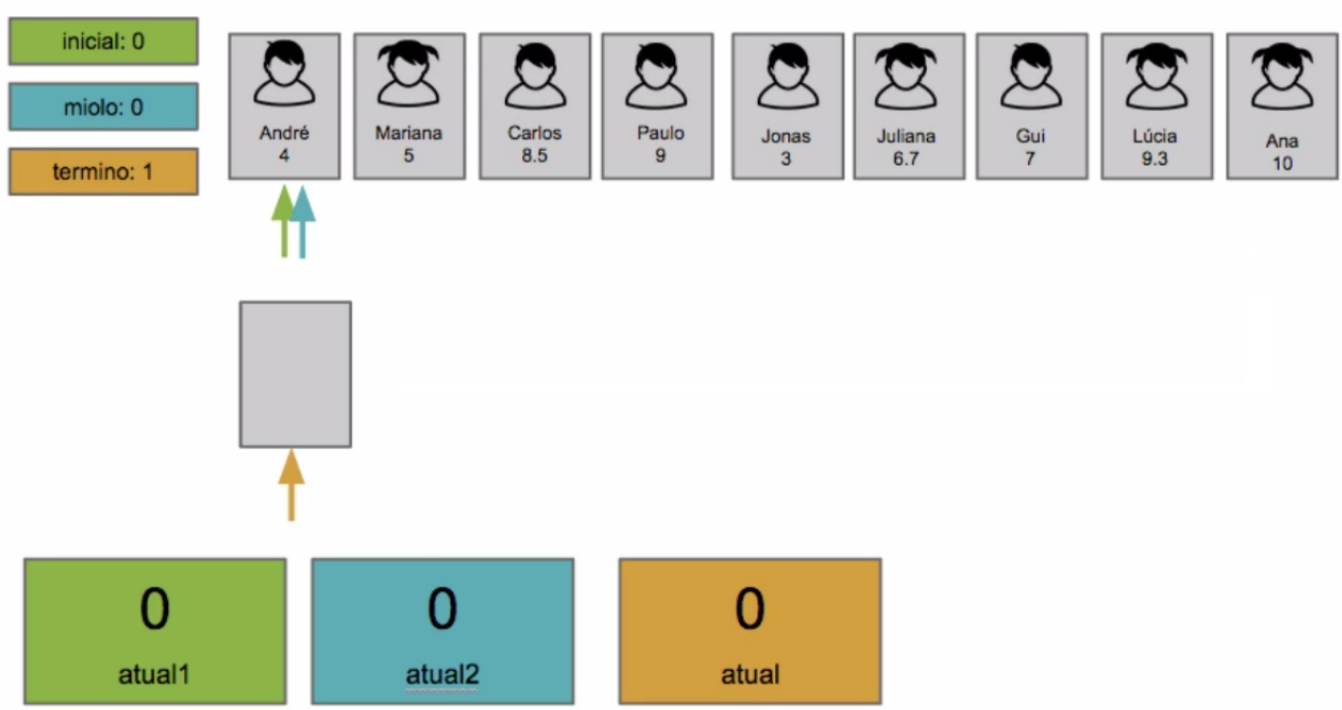
Testando diversas variações do intercala

Transcrição

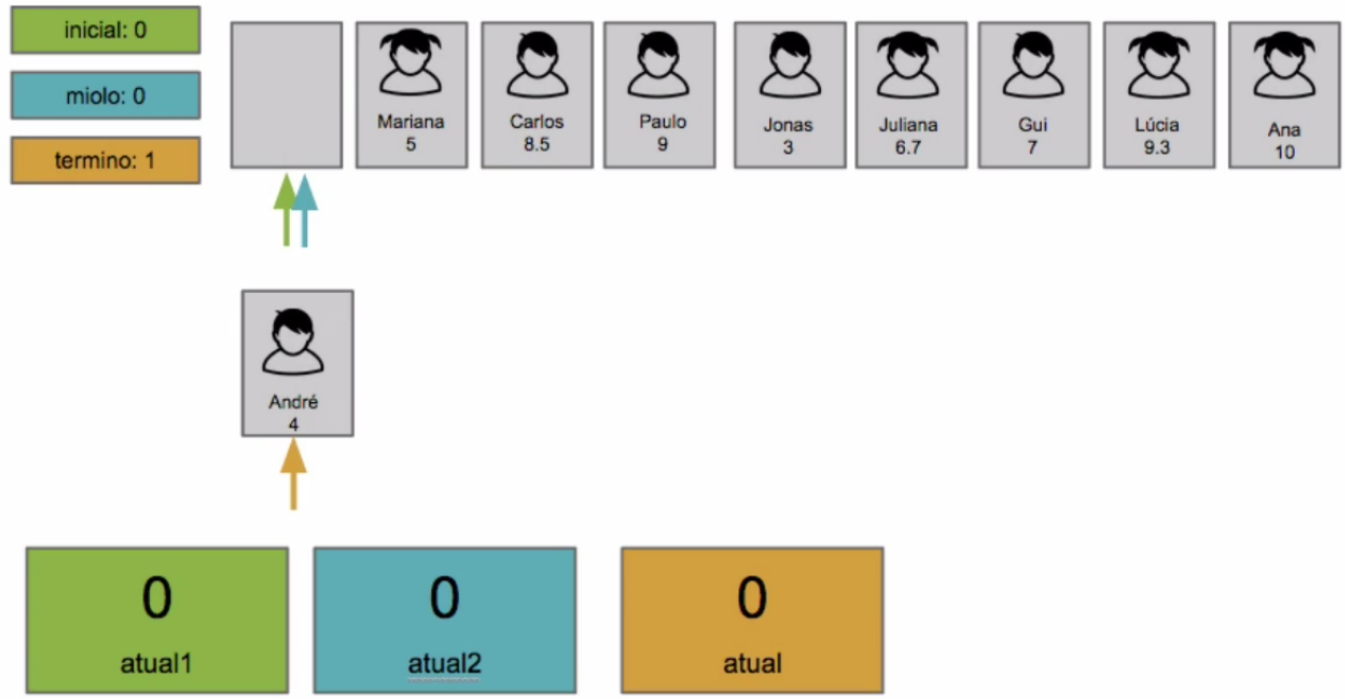
Um outro teste interessante que podemos fazer é: em vez de brincarmos apenas com o `inicial`, vamos alterar também o valor da variáveis `miolo` e `termino`. Não iremos apenas ordenar dois trechos de um *array*, que já estão ordenados. Vamos complicar mais. Nosso `inicial` será igual a 0, o `miolo` também será igual a 0, e o `termino` será igual a 1. Pedimos para o algoritmo analisar apenas um elemento. Será que funciona?



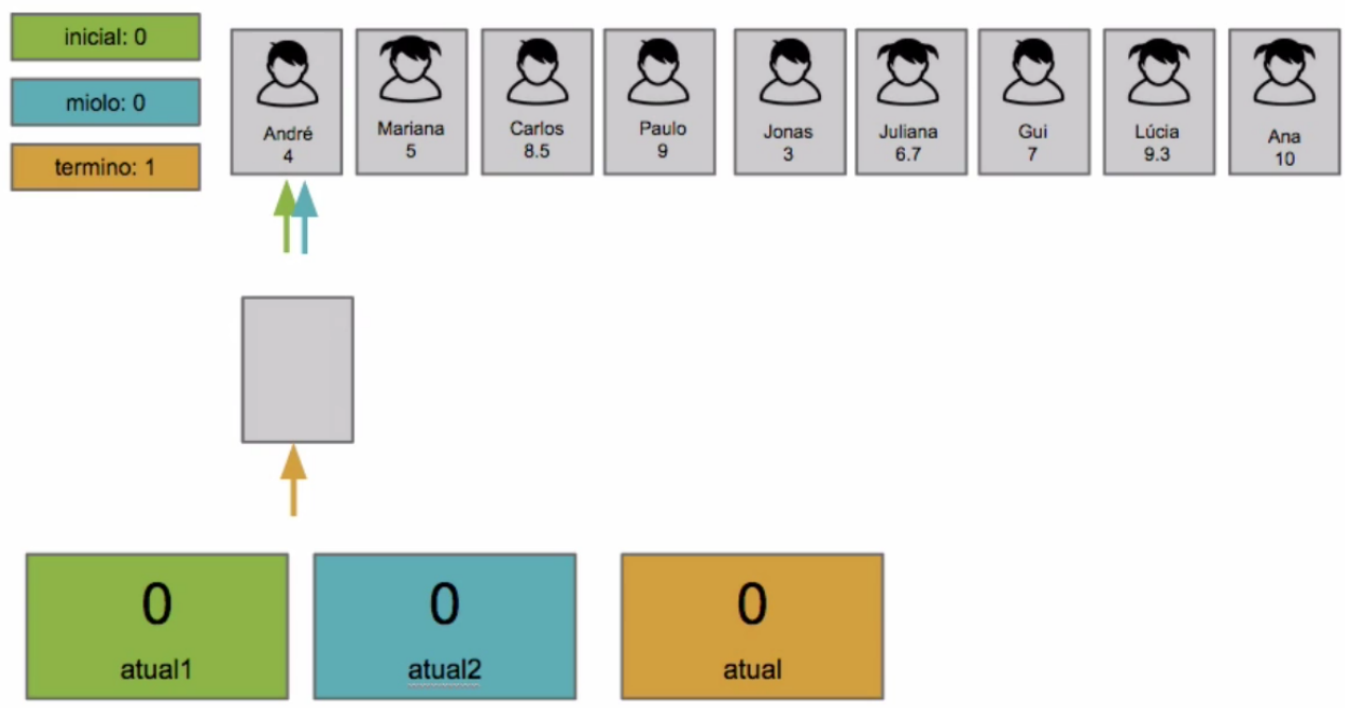
Primeiramente, quantas casinhas serão criadas? Como `termino` menos `inicial` é igual 1, nosso *array* novo terá apenas **uma** casinha.



A variável `atual1` será igual a 0, assim como `atual2` e `atual`. Então, temos que atender a condição de que iremos continuar enquanto `atual1` for menor que o `miolo`. Se não atende a primeira condição, a variável é igual ao `miolo`. Depois seguimos para o segundo laço. Temos mais alguma elemento para copiar na lista? Sim. Temos algo para copiar no `atual2`. Vamos mover o André para a nova lista.



Agora, teremos que copiar o que sobrou de volta para o *array* de origem.

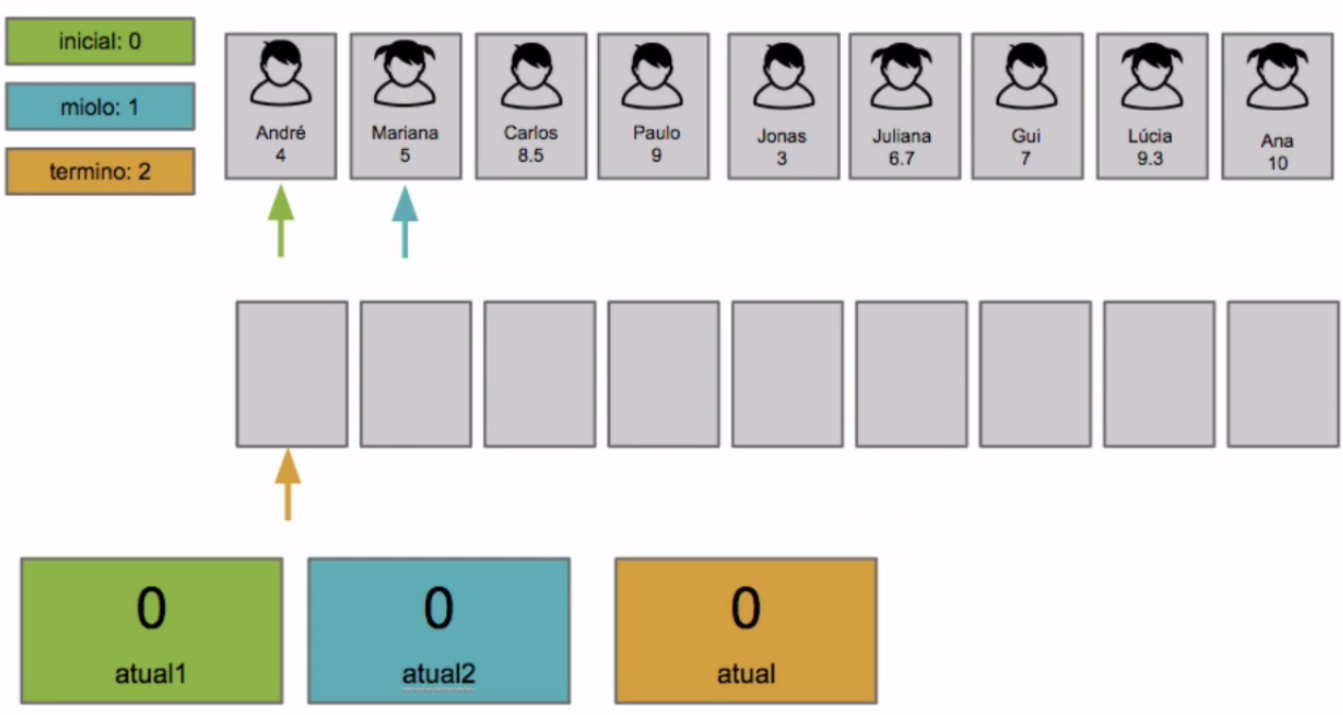


O algoritmo funcionou. Se pedimos para o algoritmo não intercalar nenhum outro elemento, ele funcionará corretamente. Mas é importante saber que eles funcionam bem nos extremos, porque precisa ser assim com todos os valores possíveis. Caso contrário, algo está errado. O algoritmo estará correto, se passarmos os valores válidos: O primeiro trecho do *array* tem tamanho 0, porque o 'miolo' vai até 0. O segundo trecho do *array* tem tamanho de 0 até 1. Com estes dados, nosso algoritmo funciona bem? Sim. Ele envia o elemento para o *array* temporário e o move novamente para o *array* original. Tudo funcionou corretamente.

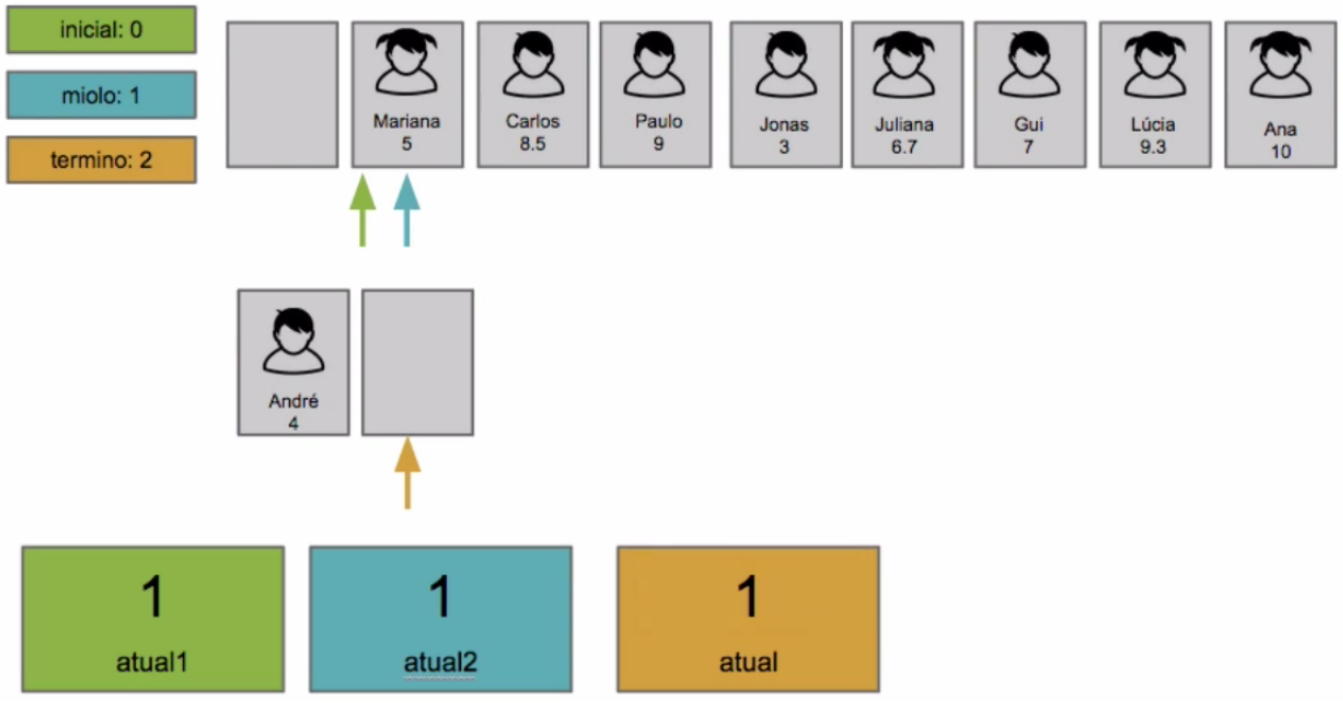
Tamanhos válidos para o intercala

Em vez de pegarmos o *array* inteiro, ou ignorarmos algum item do começo, ou do fim, o que aconteceria se criássemos um grupo pequeno com apenas dois elementos? Será que funcionaria? Vamos testar.

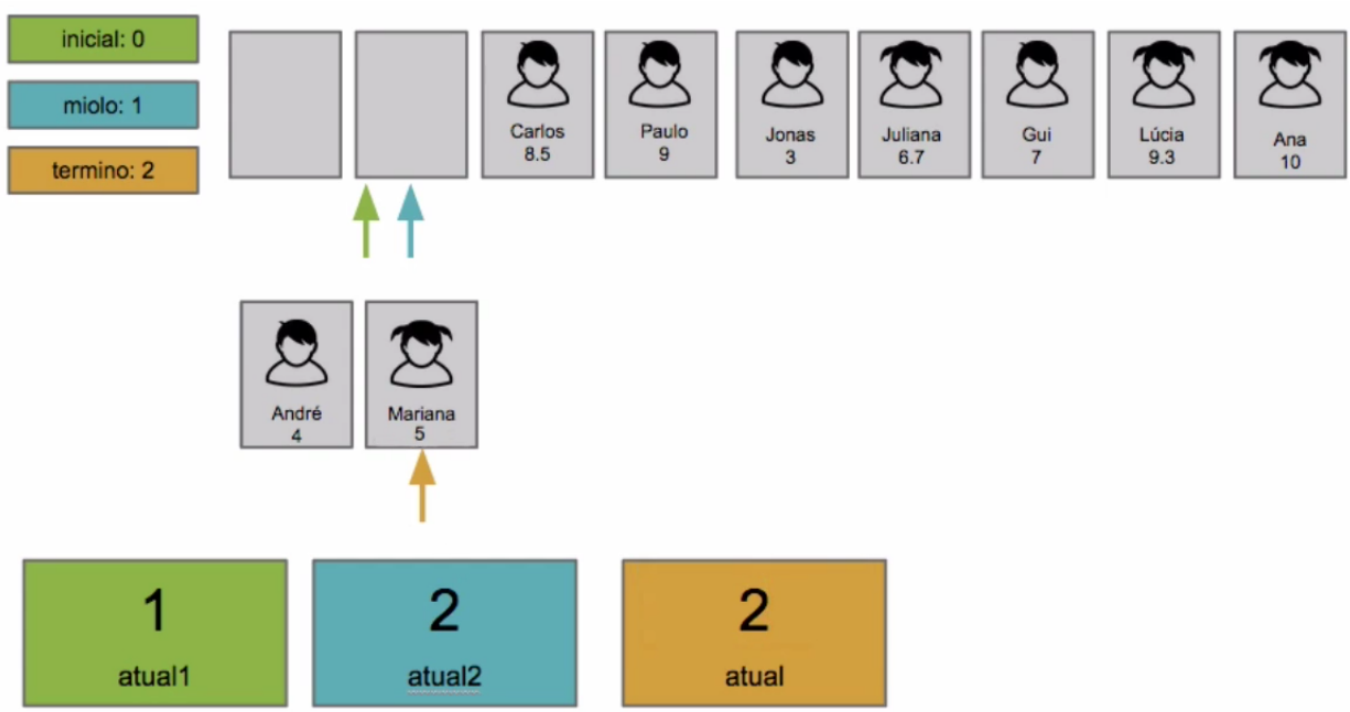
Se selecionarmos dois elementos. O primeiro trecho será da posição 0 e o `miolo` 1, e o segundo trecho será do 1 até o `termino` 2.



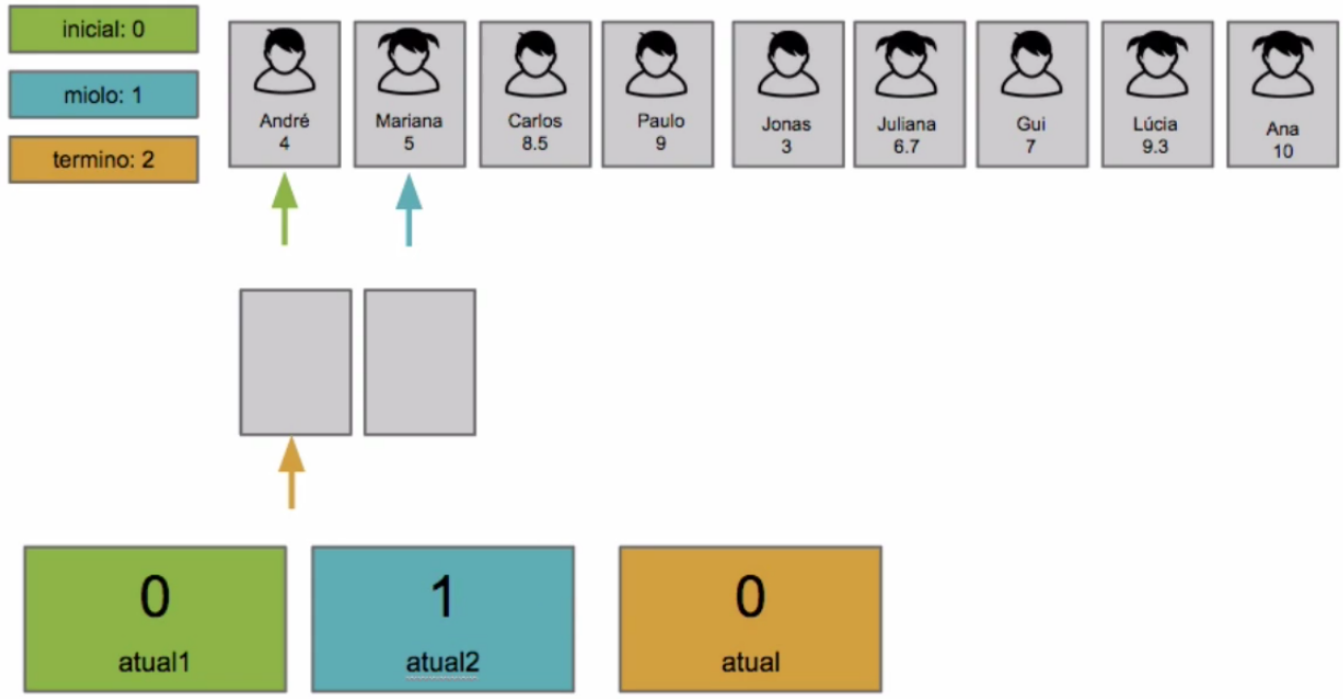
Qual será o tamanho do *array* que iremos criar? Será o tamanho 2. Em seguida rodaremos o algoritmo. `atual2` começará com o valor do `miolo` igual a 1. Como funciona o algoritmo? Iremos comparar o André, com a Mariana. Qual é o menor? O André, então iremos movê-lo para o outro *array*. Depois somaremos +1 no `atual1` e no `atual` .



Sobrou algum elemento no *array* no primeiro trecho? Não, porque já chegamos no `miolo` . Sobrou no segundo? Sim, a Mariana. Vamos movê-la para o novo *array*.



Depois aumentaremos +1 na variável `atual` e acabou o nosso *array*. O que faremos agora? Copiaremos de volta os elementos para o *array* original.



Para dois elementos, nosso algoritmo funciona. Porém, funcionou porque os elementos já estavam ordenados em posições válidas. Mas e se os dois elementos estiverem com as posições trocadas? Teria funcionado? Vamos testar com o algoritmo com dois elementos trocados. Vamos começar com o `inicial` igual a 3 , o `miolo` será igual a 4, e o `termino` igual a 5.