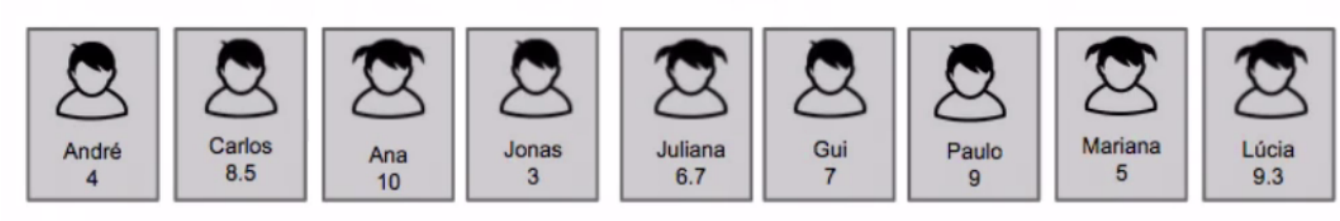
01

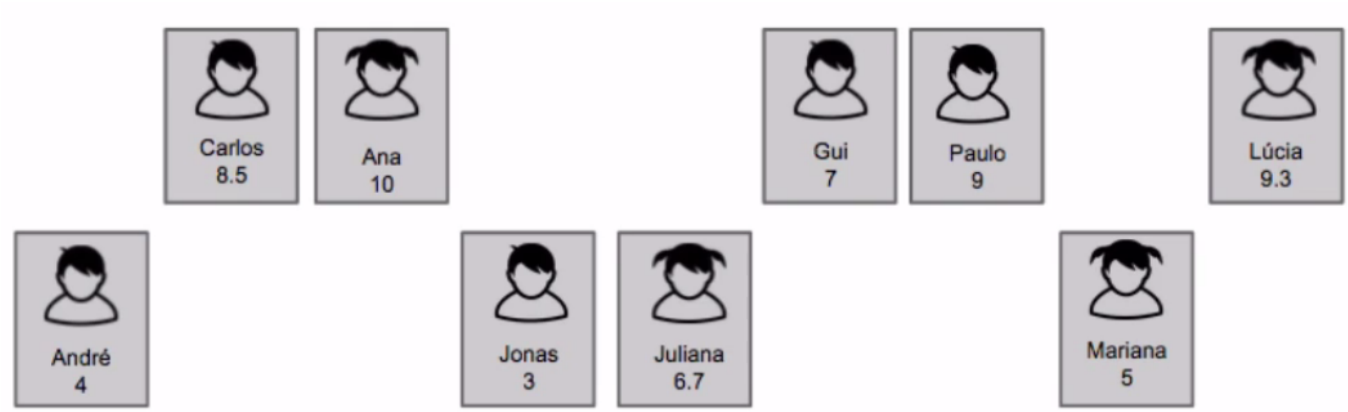
Pensando em ordem e posições absolutas

Transcrição

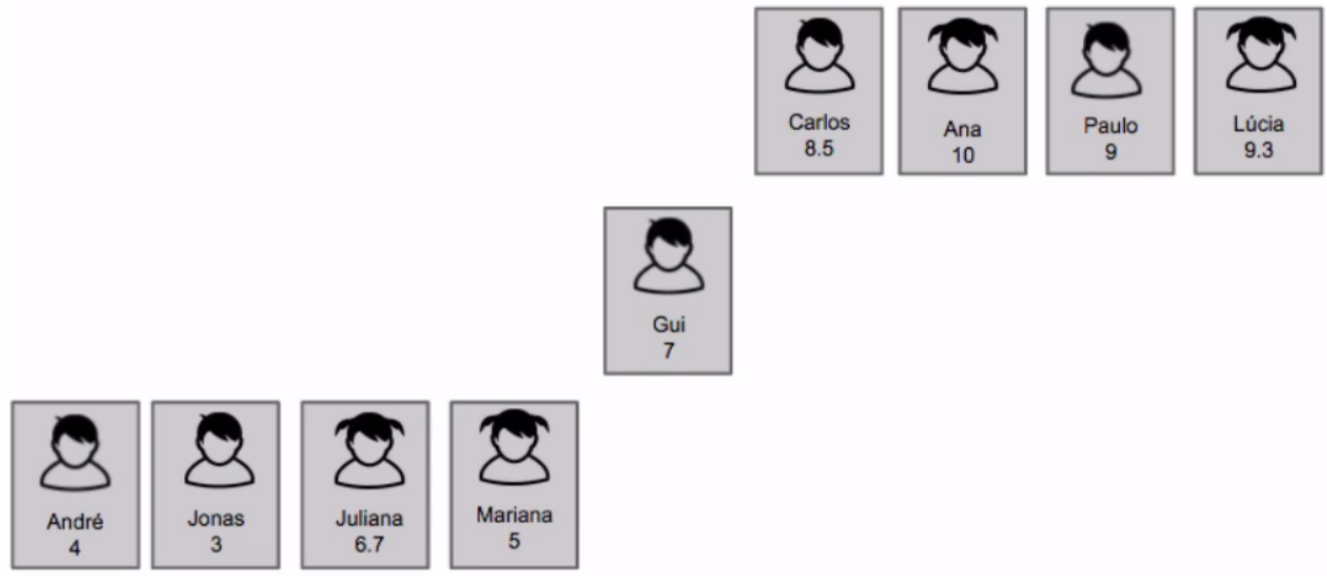
Nós já temos uma maneira de descobrir quantas pessoas se saíram piores do que eu em uma prova. Quantos tiraram uma nota pior. Se eu sei quantas pessoas foram melhores ou piores, é natural imaginar com quem irei estudar na próxima prova...



Eu separarei na lista os alunos que se saíram bem ou mal na prova. Quem foi pior, eu deixarei de lado. Para a próxima prova, eu vou tentar estudar com quem se saiu melhor. Seria interessante se pudéssemos separar aqueles que tiraram notas maiores, daqueles que tiraram notas menores.



E colocar cada grupo de um lado...

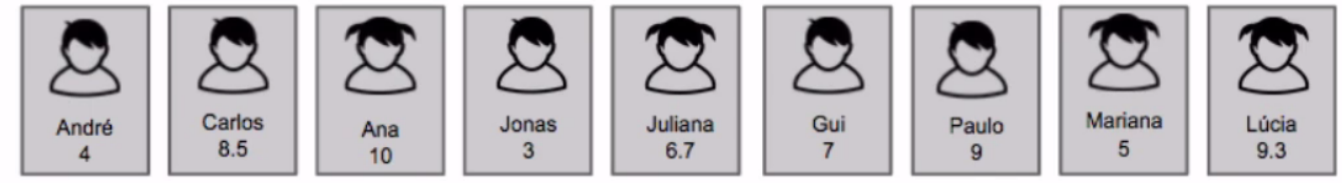


Porque se eu ficar exatamente no meio, saberei que as pessoas na minha esquerda serão pessoas que tiveram resultados piores. Mesmo que elas não estejam ordenadas, isto é irrelevante.

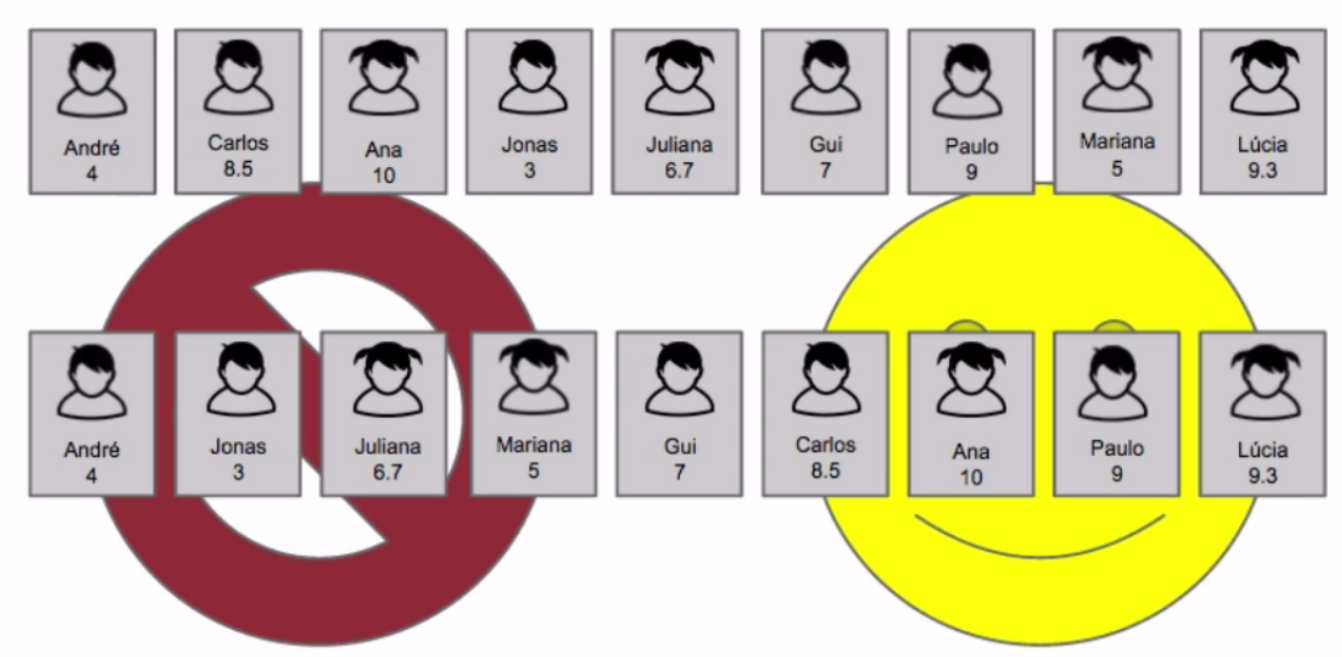
Porém, os alunos que eu querei estudar serão os posicionados à direita, porque foram os que se saíram melhor.

Sabemos contar as que se saíram melhor. O que nos falta aprender é como separar os dois grupos.

Temos a lista original, que está desordenada,:



Queremos colocar quem se saiu melhor para esquerda e quem se saiu melhor para a direita.



Eu permanecerei no meio, porque é o lugar que me corresponde. Se existem quatro pessoas que foram melhores e outras quatro que foram piores, ficarei na posição que mereço. Está tudo bem, fico feliz em permanecer na posição que estou. Se os elementos posicionados na esquerda estão desordenados, não é minha preocupação. Estou mais interessado em saber aonde mereço ficar. Eu ficarei no meio e fico contente com isto.

O objetivo é apenas descobrir quem teve resultados melhores ou piores. Então, seria bom encontrar um algoritmo que, dada a minha posição, eu consiga encontrar quem tirou notas maiores e menores.

Assim saberei com quem eu quero estudar. Observe que os elementos não ficaram ordenados, somente a minha posição está adequada, e quebra a tabela em duas. Eu sou o **pivô* que separa quem foi melhor daqueles que foram pior. E nós só

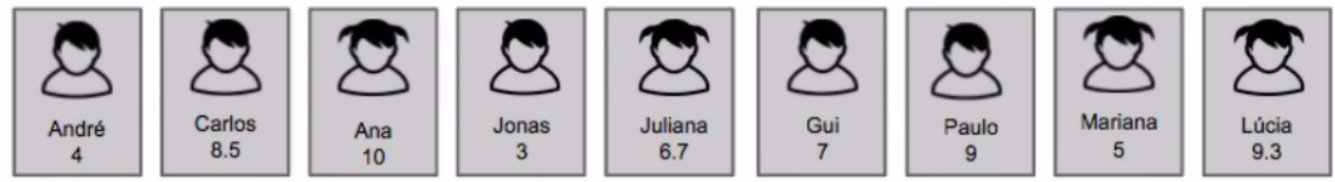
queremos saber quem teve resultados melhores ou piores do que meu na prova. Como podemos fazer isto?

O problema

Agora que temos nosso **array** de elementos, em que estou posicionado no meio, queremos fazer a ordenação. Por isso, quando eu for para a posição adequada, todo os demais que estiverem na minha esquerda terão notas menores. E todos que estiverem à direita terão notas maiores. Mesmo que cada parte não fique ordenada... O importante é que eu particione, que eu divida estes dois pedaços. Os menores irão para esquerda e os maiores, para a direita. Isto é, irei me deslocar para indicar quais são os menores e os maiores. Quem é maior ou menor, é irrelevante para mim. O importante é que eu esteja na posição que eu mereço.

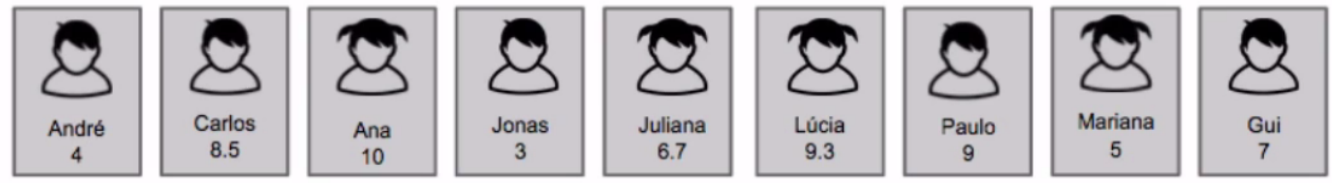
O pivô da separação

Para tentar criar um algoritmo que seja capaz de dividir um *array* em dois, com os elementos maiores e menores separados pelo Guilherme posicionado no meio, receberemos um *array* e o elemento que particiona, que divide.



Quando você tem um grupo e um integrante que divide o grupo no meio, causando uma grande quebra, costumamos chamá-lo de **pivô**. Ele é quem causa a separação. Queremos saber quem será o pivô que irá separar o grupo.

Se escolhermos um pivô do meio da lista, teremos problemas, porque cada vez teremos que escolher um elemento diferente. Será mais complexo do que se usássemos como critério escolher sempre o primeiro ou o último. Vamos facilitar um pouco. Ao invés de escolhermos um pivô aleatório - o que é válido - vamos simplificar. Iremos trocar de lugar o **Guilherme** com a **Lúcia**.



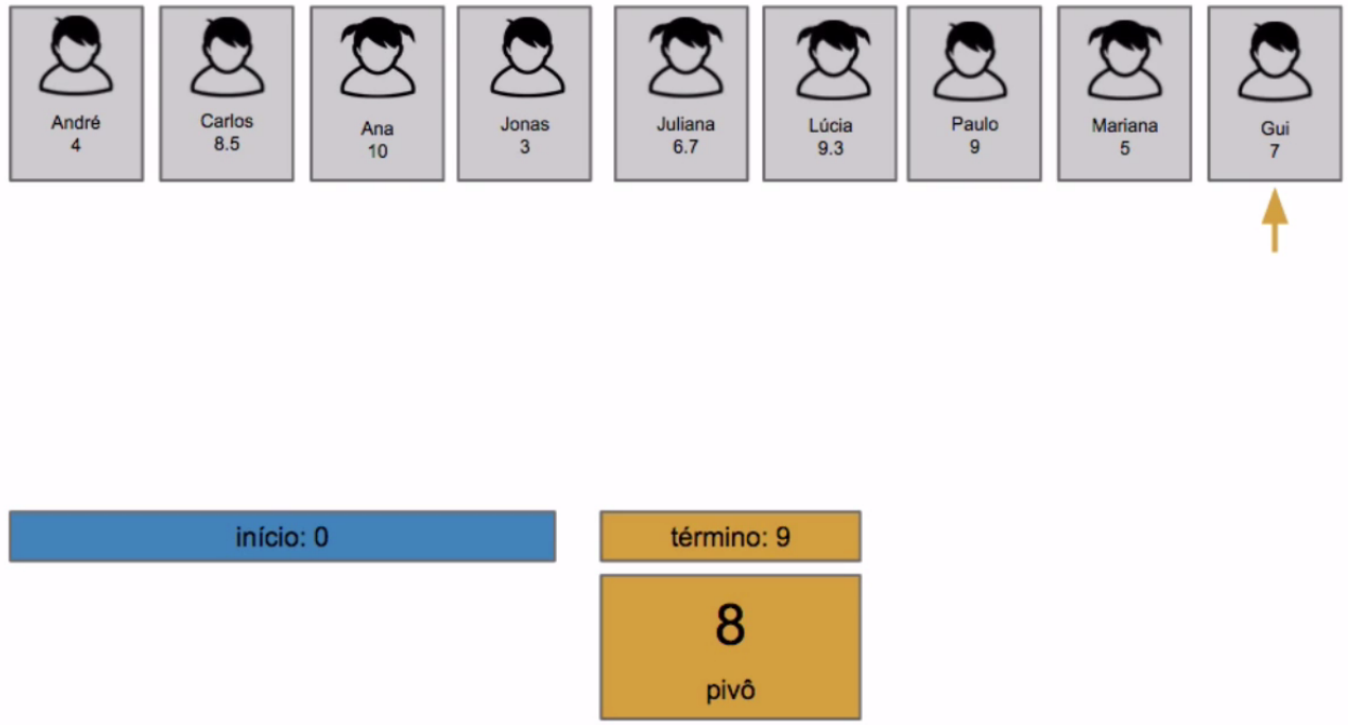
Vamos trabalhar com o pivô sempre ocupando a última posição. Assim será mais fácil para o algoritmo particionar - dividir o *array* em dois pedaços. Sendo que o pivô, no nosso caso o Guilherme, sempre será o último elemento. Dada a nossa lista, como poderemos encontrar a posição adequada do último item? Além de posicionar corretamente o pivô, também queremos colocar na esquerda os menores e na direita, os maiores. O importante é saber que o pivô é o último elemento.

Variáveis para participar

Como toda função, em todo problema computacional, nós recebemos uma entrada e precisamos devolver uma saída. A entrada é o *array*. Mas quais elementos iremos analisar? Como de costume, precisaremos saber qual é o começo (`início`) e qual é o fim (`término`).

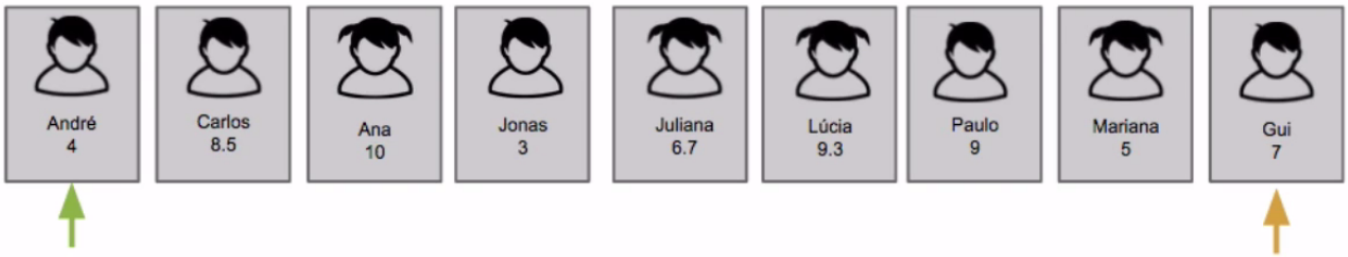
A nossa saída será o *array* já particionado, com o pivô na posição adequada. Todos os elementos da esquerda podem ser menor ou igual, e os demais da direita podem ser maior. Isto é o que importa, independente da posição em que ele caia.

Quem é o nosso pivô? Considerando os dados, ele é o `término` menos 1. Meu pivô é o elemento da posição 8, o Guilherme, que tirou nota 7.

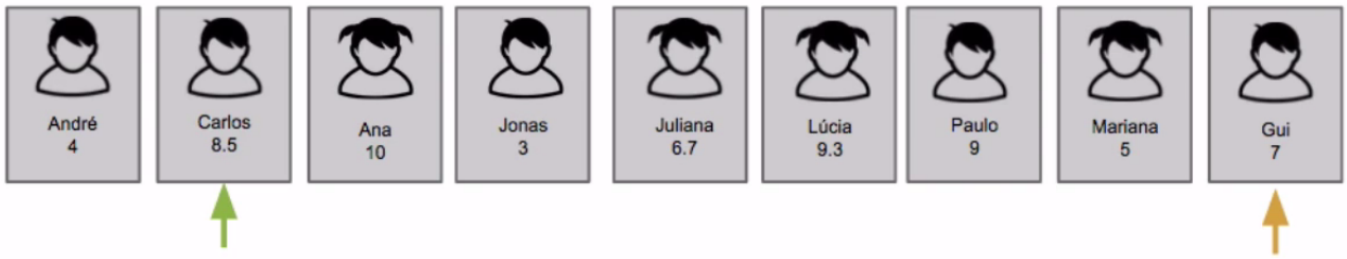


Aonde o Guilherme merece estar? É a grande questão. Vamos colocá-lo no lugar, assim ficaremos satisfeitos se todos que estiverem posicionados a sua esquerda forem menores e se a sua direita, forem maiores. Não importa se ele vai cair na segunda posição ou se a Lúcia deveria ficar na penúltima posição. As posições são irrelevantes. O importante é que dado o pivô, os elementos menores fiquem à esquerda e os menores, à direita. Vamos testar?

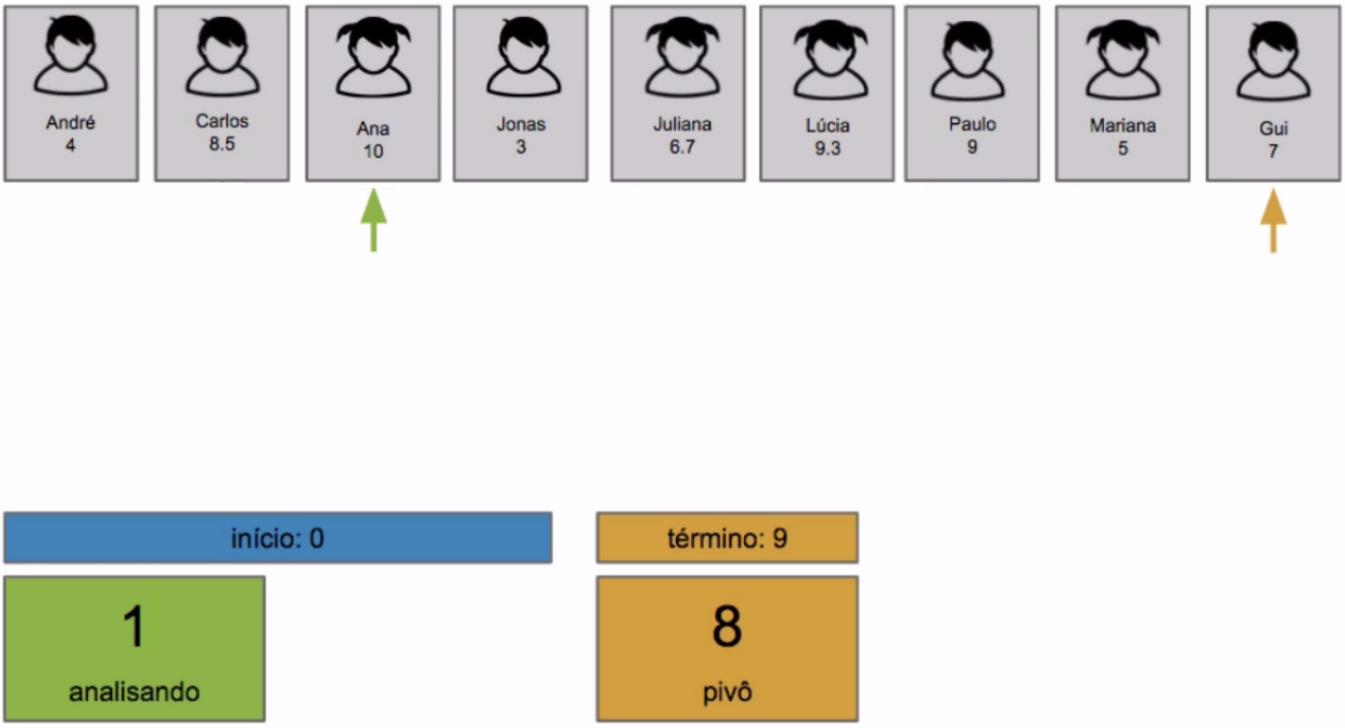
Teremos que varrer o *array*, porque precisamos saber quem é menor que o Guilherme. Se existirem três notas menores, ele precisará ocupar a quarta posição. Se existirem duas notas menores, ele precisará ocupar a terceira posição. Se existirem 17 notas, ele precisará ocupar 18ª posição. Iremos varrer o *array*, contando quantos são os menores. Usaremos indicadores para andar no *array*, indicando quais elementos serão analisados. A variável `analisando` começará do `início` e irá até o `termino`.



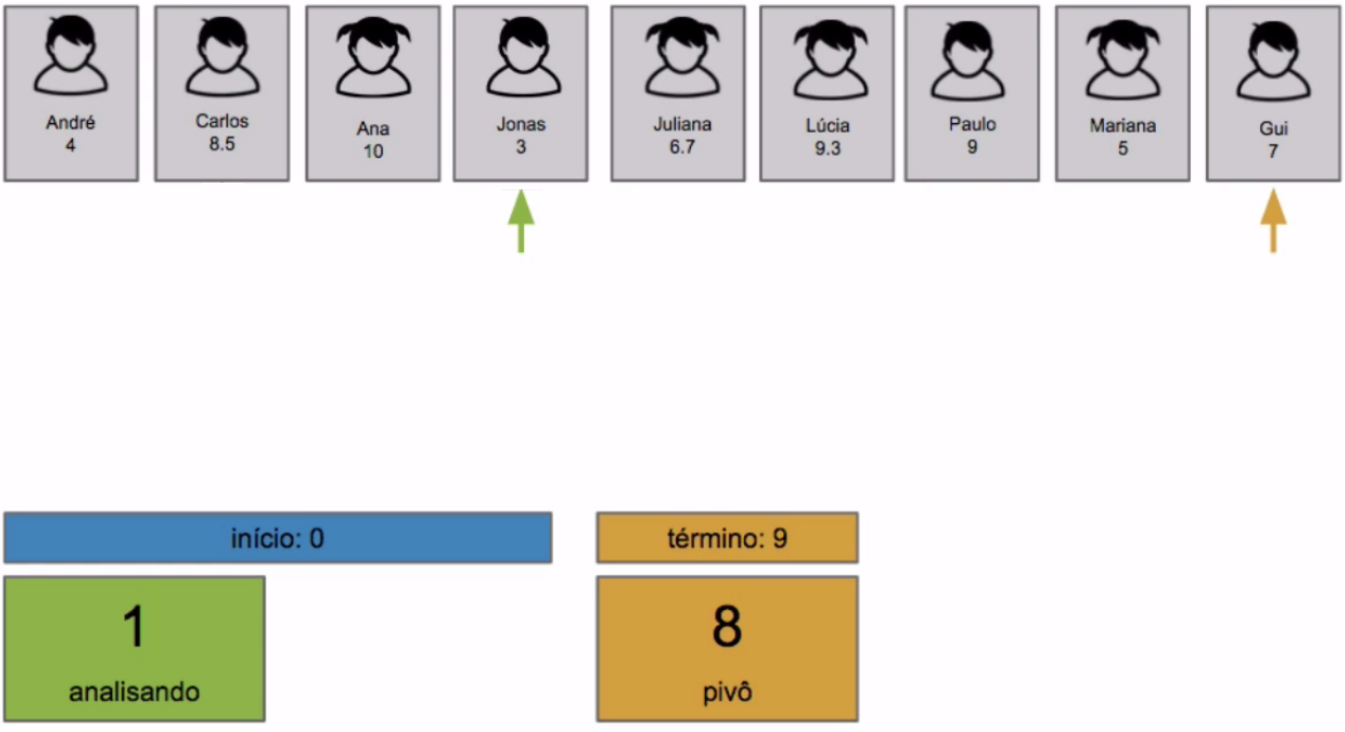
Ao analisarmos o André, ele é menor do que o Gui? Sim. Vamos somar +1 ao `analisando`. Então, quantos já temos menores? **Um** elemento.



Agora, o Carlos. Ele é maior ou menor? Maior. Seguiremos para o próximo.



A Ana é menor ou maior? Maior.



O Jonas é menor ou maior? Menor. Temos **duas** pessoas com notas menores.

Existe um forma de anotarmos no computador as pessoas que tiraram notas menores? Sim. Voltaremos ao início do processo para criar a variável `menoresAtéAgora` que armazenará o número de elementos menores que o pivô até agora. No começo, o valor será igual a 0, porque ninguém será menor que o pivô. Vamos testar novamente o nosso algoritmo. Iremos analisar cada um dos elementos, da esquerda para direita, tentando encontrar quem tirou notas menores que a do Guilherme.