02

MergeSort: Ordenando através de intercalações

Transcrição

Nós vimos que chamar o `intercala` para ordenar um *array* que não está ordenado, não funciona. Porém, vimos também utras ideias. Em vez de chamar o `intercala` do 0 até o 4...

```
intercala(notas, 0, 4, notas.lenght);
for(Nota nota : notas) {
    System.out.println(nota.getAluno() +" " + nota.getValor());
```

Iremos pedir para intercalar apenas dois elementos. Isto funciona, assim como intercalar apenas um - por se temos uma parte com um item e outra sem nenhum, só teremos o mesmo. Se mandarmos ordenar dois elementos, cada um já estará ordenado. Ou iremos trocá-los de posição, ou eles permaneceram como estão. `intercala` com dois elementos funciona muito bem.

Então, não iremos ordenar todos, como se fosse uma grande ordenação, pediremos para intercalar o primeiro e o segundo elemento.

```
intercala(notas, 0, 1, 2);
```

Isto é, queremos que intercale entre 0 e 2 (excluindo o 2), e o `miolo` será igual a 1. Queremos intercalar só os dois itens. Qual será o resultado?

```
andre 4.0
carlos 8.5
ana 10.0
jonas 3.0
juliana 6.7
guilherme 7.0
paulo 9.0
mariana 5.0
lucia 9.3
```

Ele intercalou os dois primeiros e deixou o restante nas posições em que estavam. Nós já tínhamos testado este código. Ao intercalar dois elementos que já estavam em ordem, ele mantinha os dois como estavam. Vamos tentar intercalar os próximos dois elementos: a Ana e o Jonas. Vou especificar que iremos intercalar a partir da posição 2, o `miolo` é igual a 3, iremos até a posição 4.

```
intercala(notas, 0, 1, 2);
intercala(notas, 2, 3, 4);
```

Ao intercalarmos os elementos, o algoritmo trocou a posição do Jonas com a da Ana.

```
andre 4.0
carlos 8.5
jonas 3.0
ana 10.0
juliana 6.7
guilherme 7.0
paulo 9.0
mariana 5.0
lucia 9.3
```

Antes o Jonas e a Ana estavam fora de ordem. Agora, tanto do 0 até 2, como do 2 até 4, já estão ordenados. Isto significa que já podemos intercalar do 0, usando o 2 como `miolo` , até o 4.

```
intercala(notas, 0, 1, 2);
intercala(notas, 2, 3, 4);
intercala(notas, 0, 2, 4);
```

Se rodarmos novamente, o resultado será:

```
jonas 3.0
andre 4.0
carlos 8.5
ana 10.0
juliana 6.7
guilherme 7.0
paulo 9.0
mariana 5.0
lucia 9.3
```

Os quatros elementos foram ordenados: Jonas, André, Carlos e Ana. Por quê? Porque intercalamos os dois primeiros e os dois seguintes. Como eles já tinham sido ordenados, era possível intercalar os quatro. O resultado devolveu os elementos ordenados. Fazendo primeiro uma intercalação pequena e depois, seguimos para a intercalação maior. E assim, parte do meu *array* foi ordenado.

Se ao intercalarmos uma parte funcionou, vamos continuar com os dois próximos, 5 e 6. Iremos intercalar a partir do 4, porque ele ainda não foi intercalado.

```
intercala(notas, 0, 1, 2);
intercala(notas, 2, 3, 4);
intercala(notas, 0, 2, 4);
intercala(notas, 4, 5, 6);
```

Depois, iremos repetir o processo para 6,7 e 8.

```
intercala(notas, 6, 7, 8);
```

Ao testarmos, o resultado será:

```
jonas 3.0
andre 4.0
carlos 8.5
ana 10.0
juliana 6.7
guilherme 7.0
mariana 5.0
paulo 9.0
lucia 9.3
```

Observe os quatro elementos: Juliana, Guilherme, Mariana e Paulo. O que podemos fazer agora? Intercalar. Então vamos intercalar as notas do 4 parando no 6 e depois seguimos até o 8.

```
intercala(notas, 4, 6, 8);
```

O resultado será:

```
jonas 3.0
andre 4.0
carlos 8.5
ana 10.0
mariana 5.0
juliana 6.7
guilherme 7.0
paulo 9.0
lucia 9.3
```

O quatro elementos foram ordenados corretamente: Mariana, Juliana, Guilherme e Paulo. Nosso *array* está ficando organizado.

Agora os quatro primeiros itens já estão ordenados, assim como os quatro seguintes. Isto significa que podemos intercalar os elementos. É o que faremos.

```
intercala(notas, 0, 1, 2);
intercala(notas, 2, 3, 4);
intercala(notas, 0, 2, 4);
intercala(notas, 4, 5, 6);
intercala(notas, 6, 7, 8);
intercala(notas, 4, 6, 8);
intercala(notas, 0, 4, 8);
```

Iremos intercalar do 0 até 4, que estavam ordenados e seguiremos até 8, uma parte que também já foi ordenada. Estamos intercalando os blocos de elementos que estão organizados. Vamos testar se está funcionando bem?

```
jonas 3.0
andre 4.0
```

```
mariana 5.0
juliana 6.7
guilherme 7.0
carlos 8.5
paulo 9.0
ana 10.0
lucia 9.3
```

Do Jonas até a Ana, os elementos estão ordenados. Faltou apenas o último. Como restou apenas um, basta intercalarmos a Lúcia com os demais, porque ela está também ordenada. Então iremos intercalar do 0 até 8, em que todos estão ordenados, e depois até 9, que igualmente está ordenado.

```
intercala(notas, 0, 8, 9);
```

Vamos testar novamente?

```
jonas 3.0
andre 4.0
mariana 5.0
juliana 6.7
guilherme 7.0
carlos 8.5
paulo 9.0
lucia 9.3
ana 10.0
```

Agora todos estão ordenados! Observe que conseguimos fazer diversas invocações ao `intercala` e com isto ordenar o nosso *array* inteiro. Ele é capaz de tal ação.

Intercalando passo a passo

Nós vimos que se chamamos o `intercala` apenas uma vez, não ordenamos o *array*. Mas se formos chamando de uma maneira inteligente, conseguimos quebrar o *array* grande em pedaços menores e começamos a ordenar por eles. Ao começar assim, o `intercala` seguirá ordenando até todos os elementos estarem organizados.

```
intercala(notas, 0, 1, 2);
intercala(notas, 2, 3, 4);
intercala(notas, 0, 2, 4);
```

O que fizemos aqui? Nós primeiro ordenamos do 0 até 2, apenas 2 elementos. Depois do 2 até 4, mais dois elementos. Então, ordenamos do 0 até o 4, quatro elementos.

```
intercala(notas, 4, 5, 6);
intercala(notas, 6, 7, 8);
intercala(notas, 4, 6, 8);
```

Seguimos ordenando mais dois elementos. E dois seguintes. Depois ordenamos de 4 até 8, os quatro elementos.

```
intercala(notas, 0, 4, 8);
```

Ordenamos os oito elementos, de 0 até 8.

```
intercala(notas, 0, 8, 9);
```

Depois ordenamos o último que sobrou com os restantes.

Nós fomos intercalando cada pedaço. Para intercalarmos um *array* grande, precisamos dividir em dois pedaços menores. Depois, intercalamos as partes ordenadas. Mas para fazer isto, as divisões precisam ser pequenas o suficiente. Enquanto não for assim, preciso ir quebrando o *array* até que o `intercala` funcione.

Em vez de mandarmos intercalar todos elementos de uma vez, precisamos dividi-los em blocos menores e então, poderemos intercalá-los.