Trabalho Teórico 4

Unidade I:
Introdução - Algoritmo de Ordenação por Seleção

Slide C.

```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;

    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap(menor, i);
}
```

1)Faça com que nosso código conte o número de movimentações?

```
New code:
int mov = 0;
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap(menor, i);
    mov += 3;
}
System.out.println(mov);
```

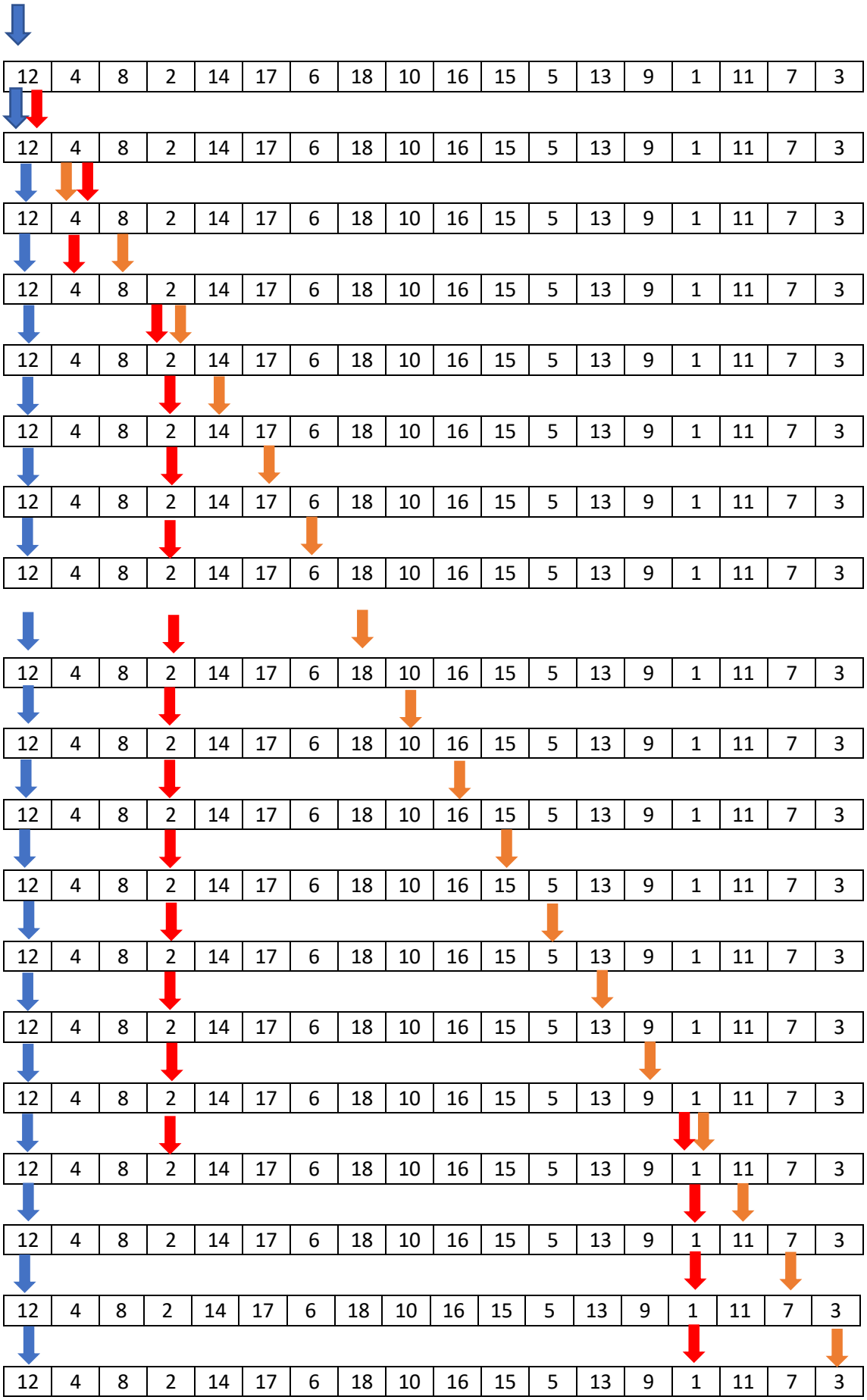
1- Mostre todas as comparações e movimentações do algoritmo anterior para o array abaixo:

```
for (int i = 0; i < (n - 1); i++)
{
        int menor = i;
        for (int j = (i + 1); j < n; j++)
        {
                if (array[menor] > array[j])
                {
                        menor = j;
                }
        }
        swap(menor, i); }
```

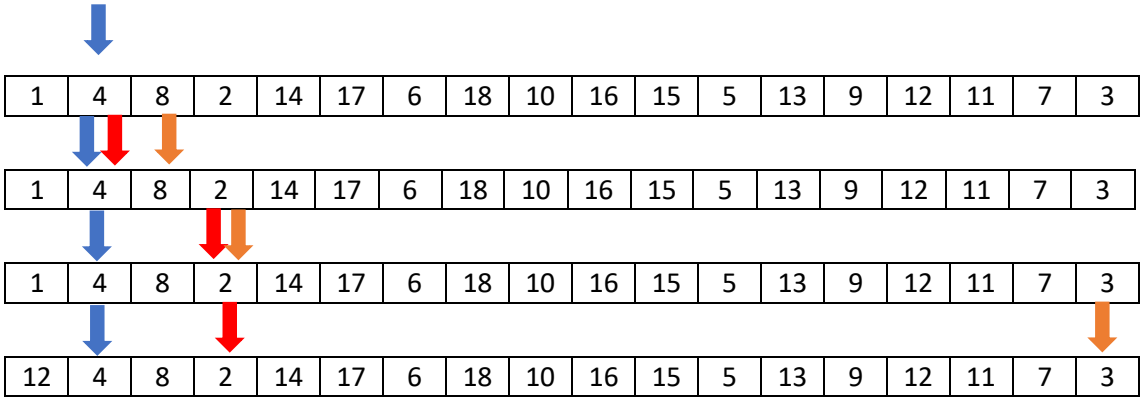
= i

= J

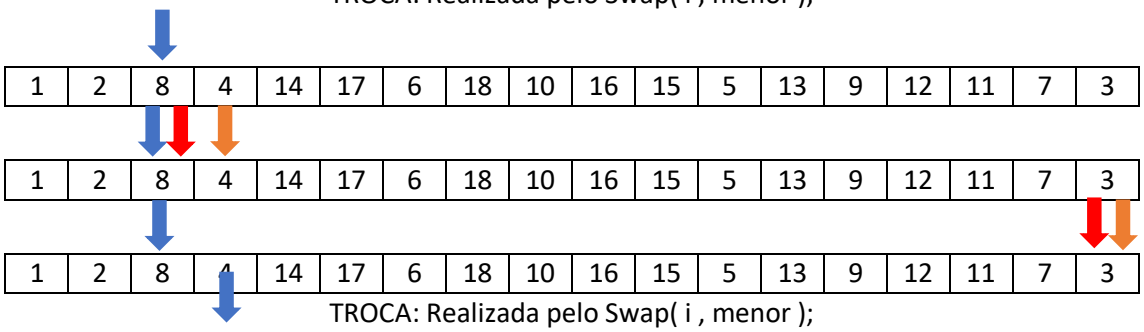
= Menor



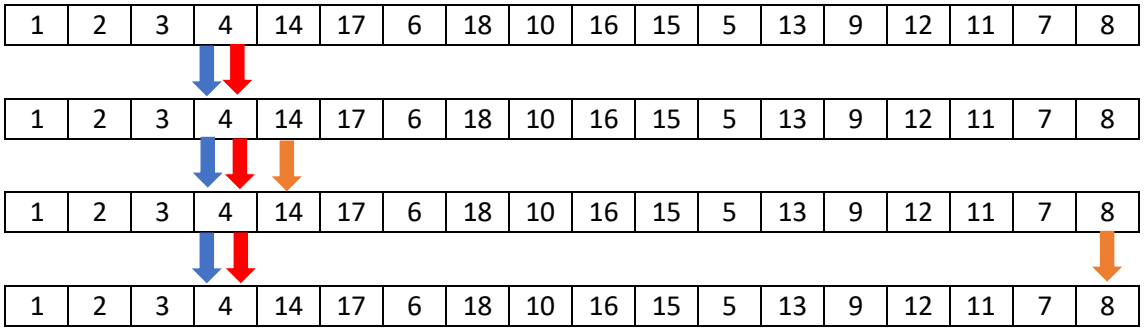
TROCA: Realizada pelo Swap(i , menor);



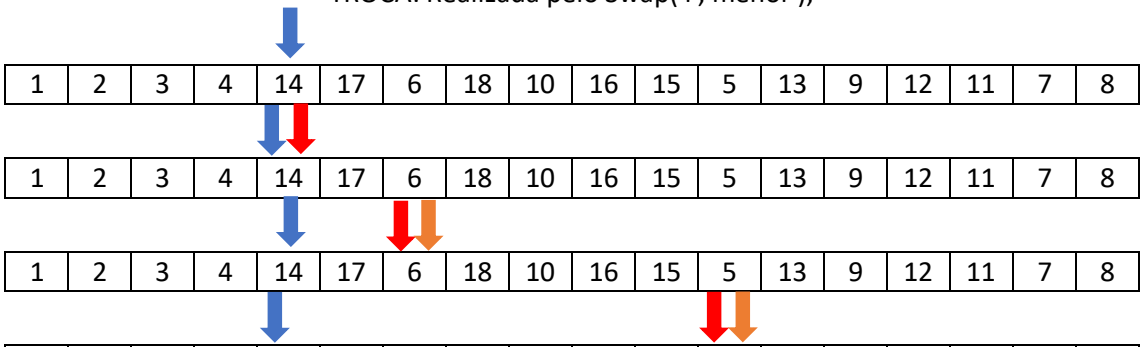
TROCA: Realizada pelo Swap(i , menor);



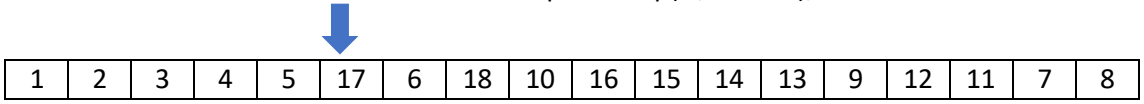
TROCA: Realizada pelo Swap(i , menor);

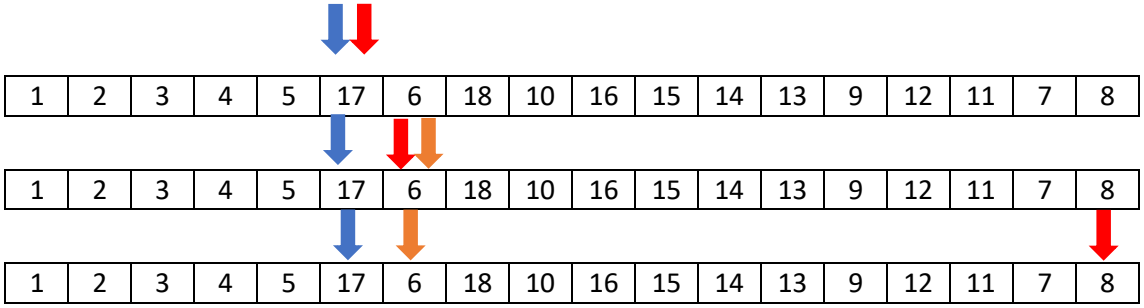


TROCA: Realizada pelo Swap(i , menor);

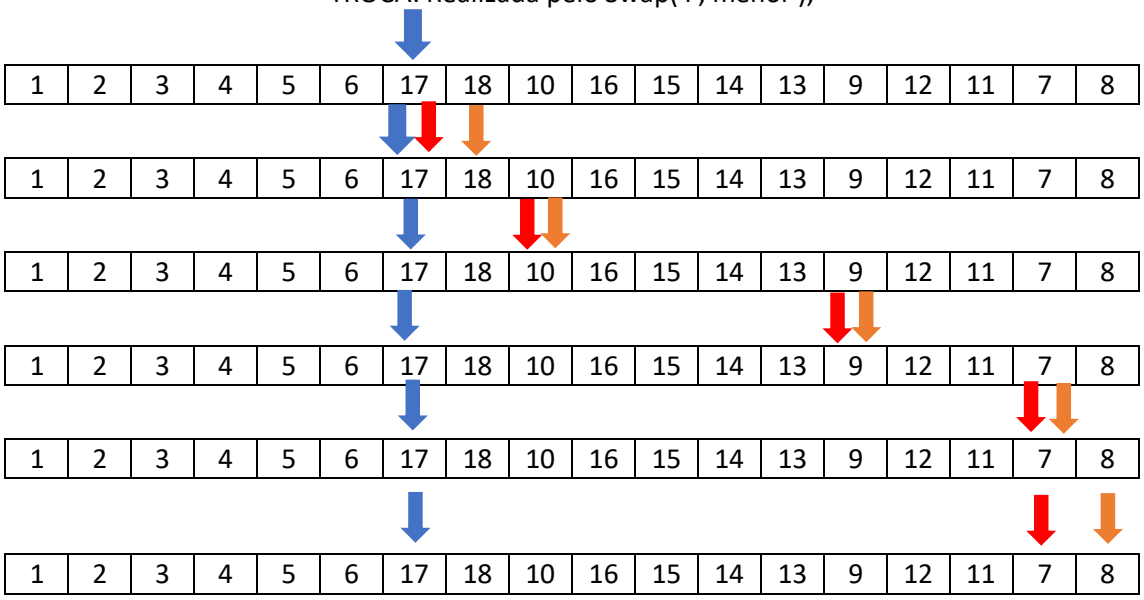


TROCA: Realizada pelo Swap(i , menor);

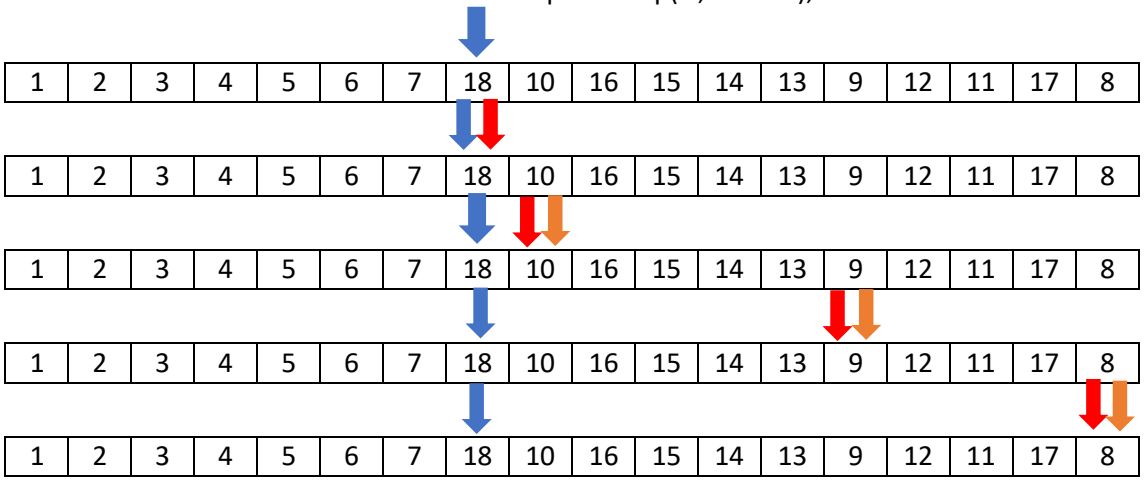




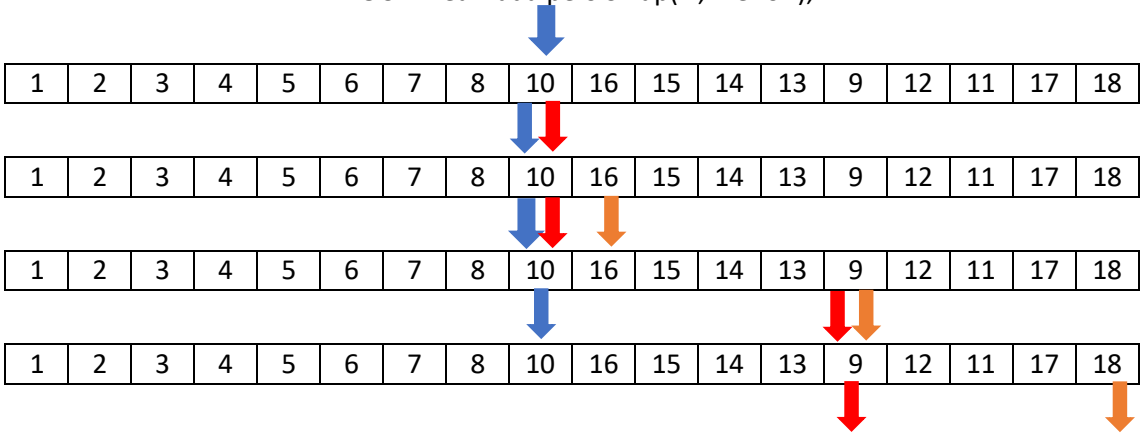
TROCA: Realizada pelo Swap(i , menor);

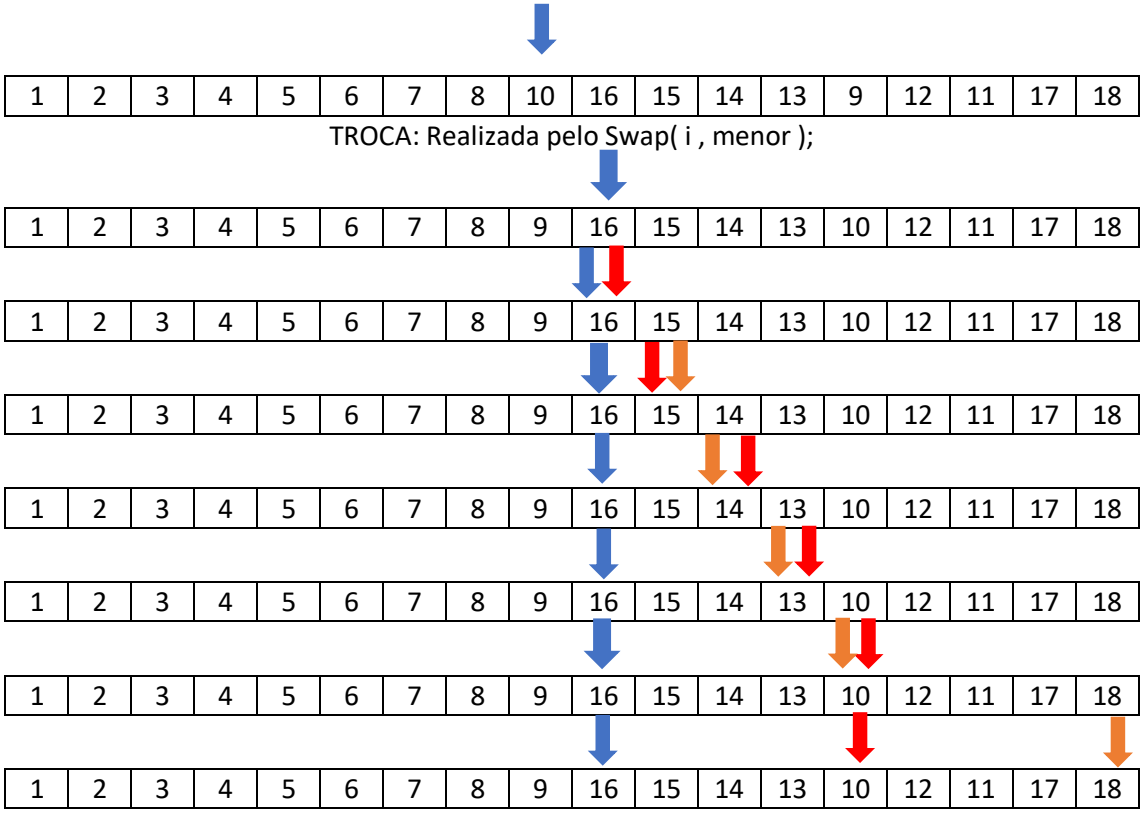


TROCA: Realizada pelo Swap(i , menor);

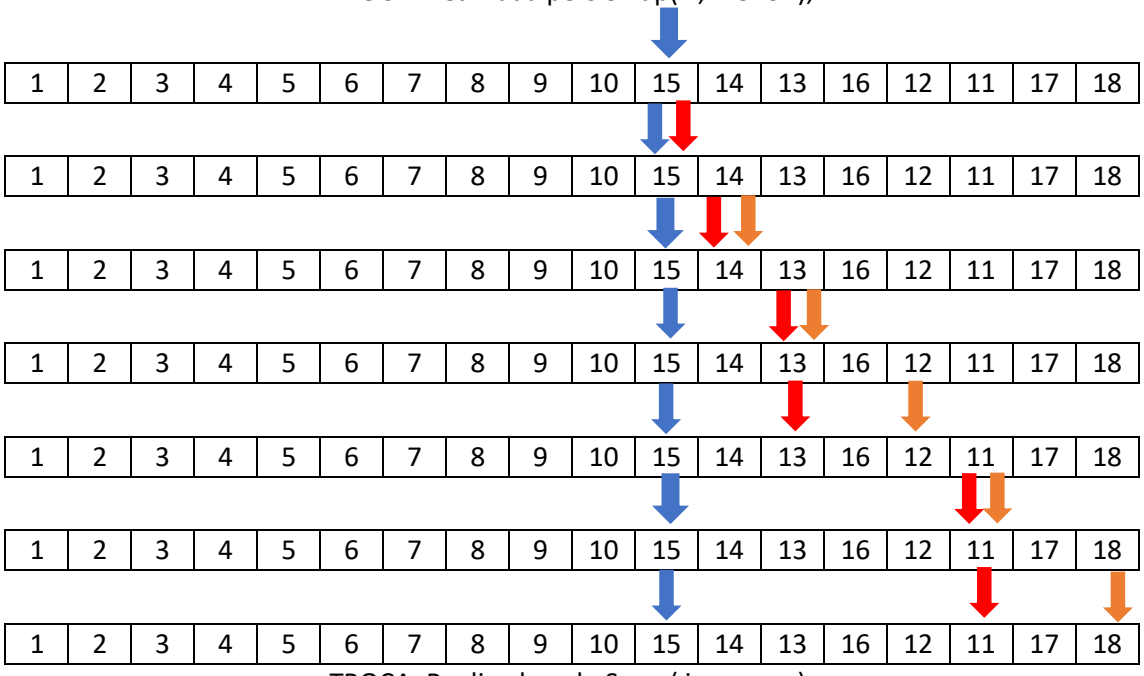


TROCA: Realizada pelo Swap(i , menor);

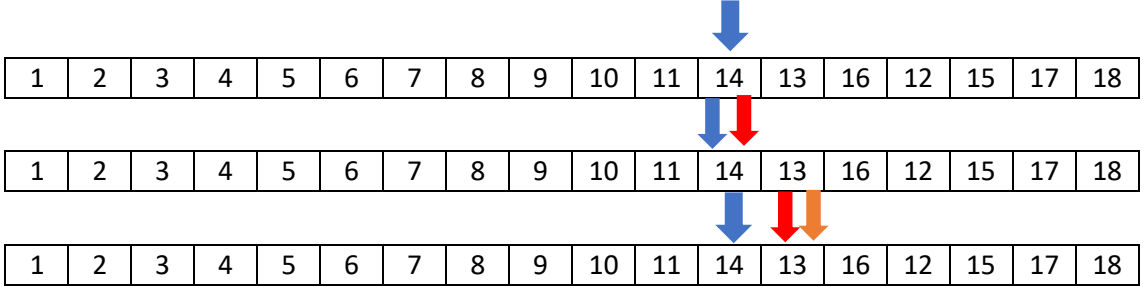


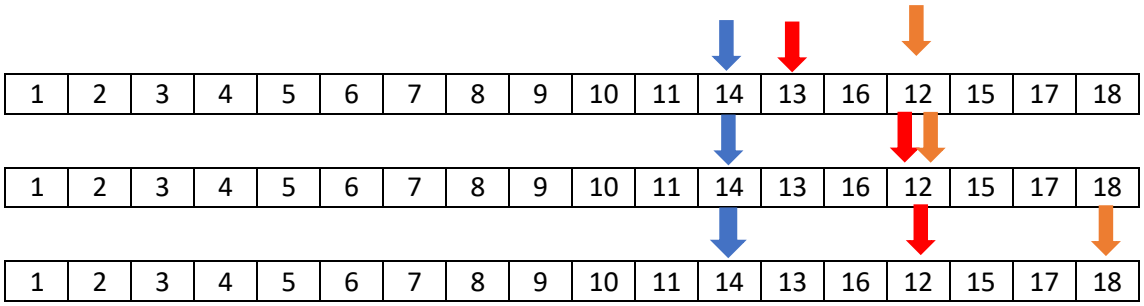


TROCA: Realizada pelo Swap(i , menor);

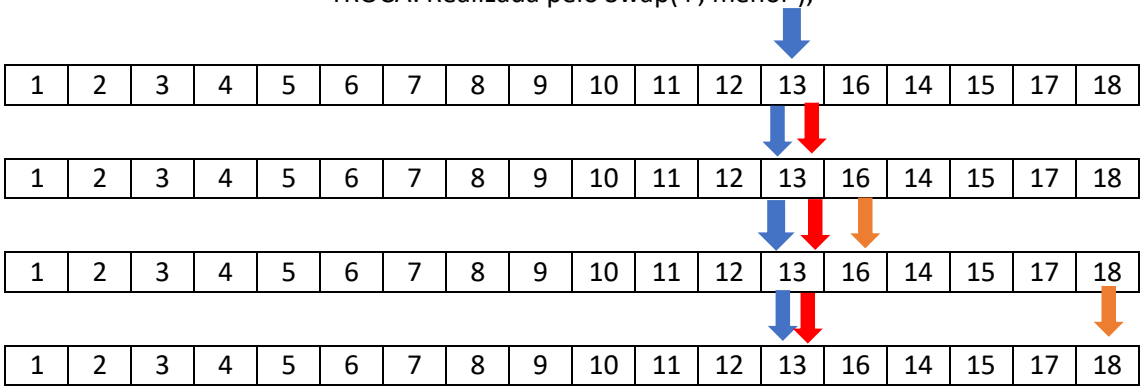


TROCA: Realizada pelo Swap(i , menor);

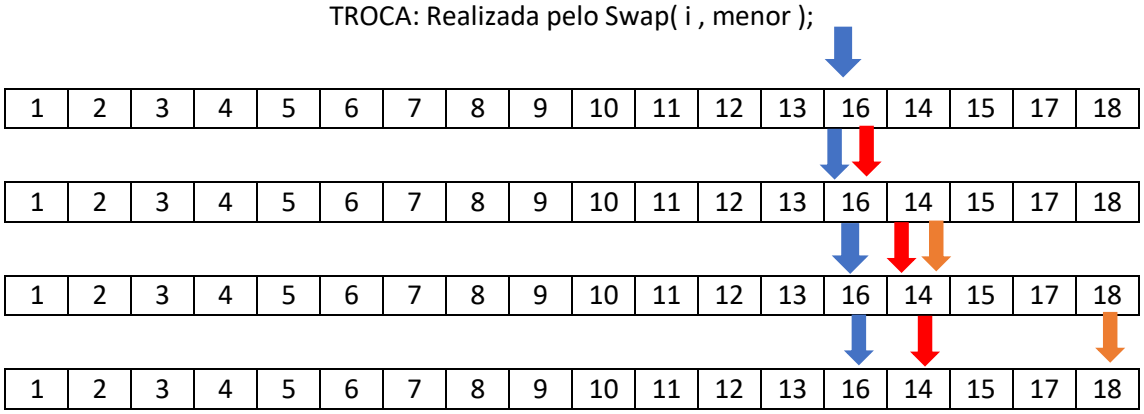




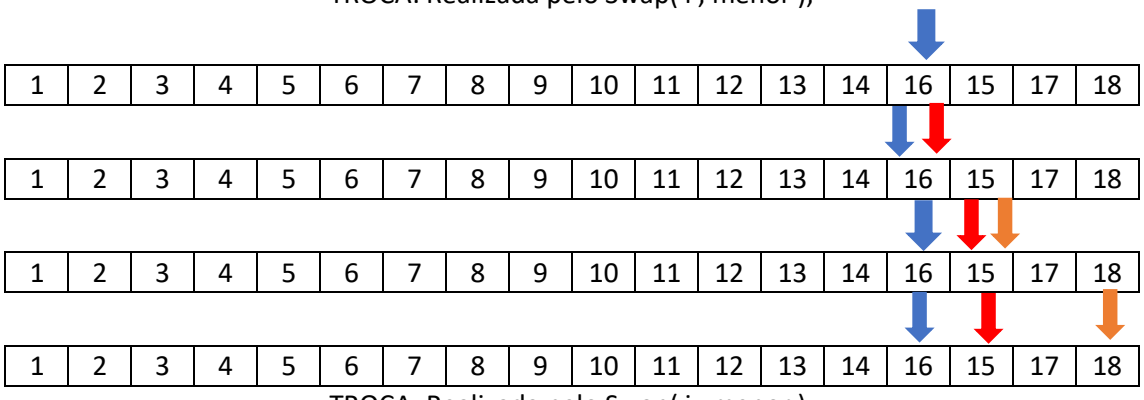
TROCA: Realizada pelo Swap(i , menor);



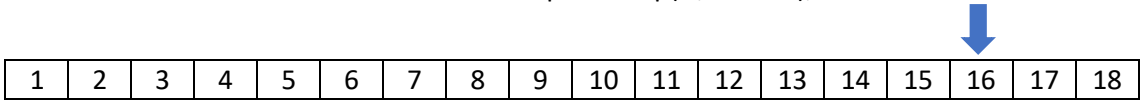
TROCA: Realizada pelo Swap(i , menor);



TROCA: Realizada pelo Swap(i , menor);



TROCA: Realizada pelo Swap(i , menor);



O vetor já está ordenado mais as comparações acontecerão mais 2x ate o fim do programa.

VETOR ORDENADO NO FIM.

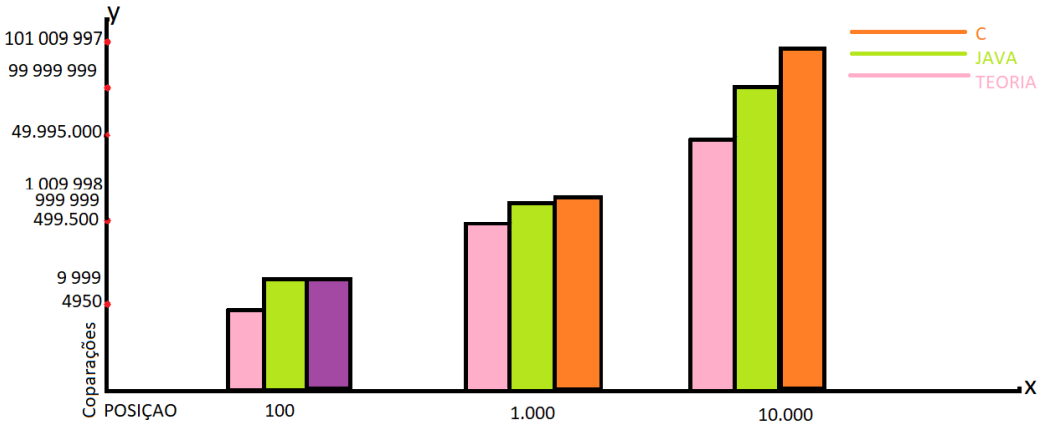
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

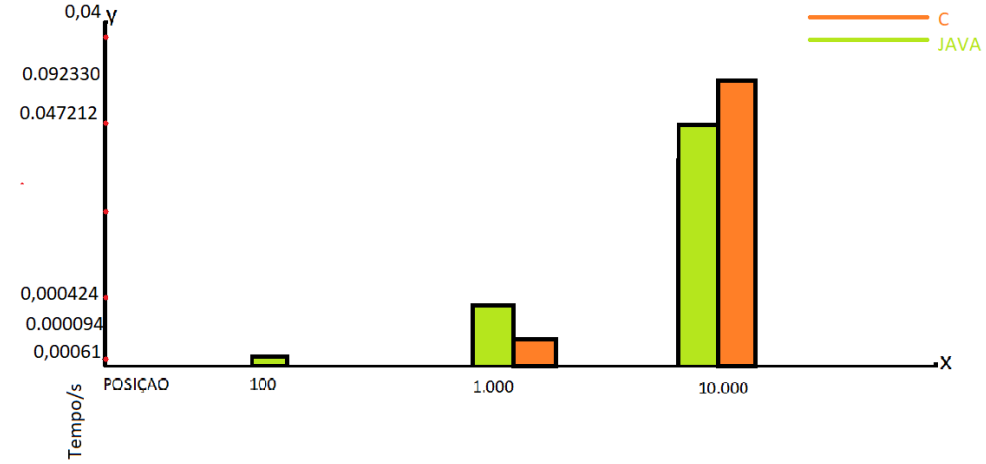
2Execute a versão abaixo do Seleção para arrays gerados aleatoriamente. Em seguida, discuta sobre os números de comparações inseridas e movimentações evitadas pela nova versão do algoritmo

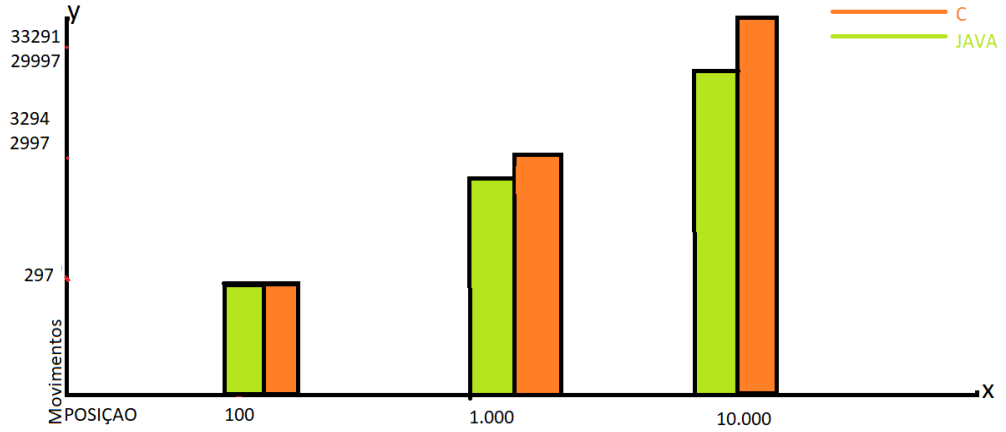
```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;

    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
    menor = j;
    }
}
    if (menor != i){
        swap(menor, i);
    }
}
```

Resp: Serão inseridas N-1 comparações a mais, com o intuito reduzir o número de trocas, mas vale em teoria mais apena pagar a movimentação do que a comparações. Média do tempo de execução dos códigos de Arraes de 100 000 posições.
Teste:
Sem if(10.451)
Com if(8.450)

3) Contabilize os números de comparações e movimentações entre elementos do array; calcule os valores teóricos para as duas métricas; e contabilize o tempo de execução. Em seguida, para os códigos em Java e C, gere arrays aleatórios (seed 0) com tamanhos 100, 1000 e 10000. Para cada instância (variação de linguagem e tamanho de vetor), faça 33 execuções. Faça um gráfico para os valores médios de cada métrica avaliada comparação, movimentações e tempo de execução) variando o tamanho do array. Nos gráficos de comparações e movimentações, mostre também os resultados teóricos. Cada gráfico terá uma curva para cada linguagem. Interprete os resultados obtidos. Repita o processo para arrays gerados de forma crescente e decrescente.


y
101 009 997
99 999 999
49.995.000
1 009 998
999 999
499.500
9 999
4950
Coparações
POSIÇAO
100
1.000
10.000
x
C
JAVA
TEORIA


0,04
y
0.092330
0.047212
0,000424
0.000094
0,00061
Tempo/s
POSIÇAO
100
1.000
10.000
x
C
JAVA


y
33291
29997
3294
2997
297
Movimentos
POSIÇAO
100
1.000
10.000
x
C
JAVA