

Unidade IV:

Ordenação Interna - Algoritmo de Seleção

Prof. Max do Val Machado



PUC Minas

Instituto de Ciências Exatas e Informática
Curso de Ciência da Computação

Funcionamento Básico

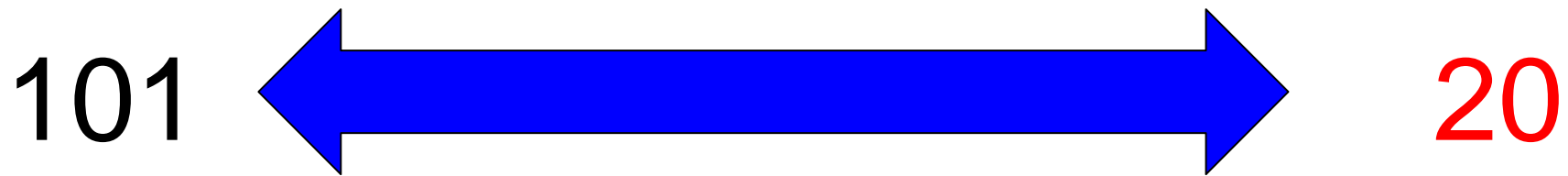
- Procure o menor elemento do *array*
- Troque a posição do menor elemento com o primeiro
- Volte ao primeiro passo e considere o array a partir da próxima posição

Legenda: - menor elemento em vermelho
- parte ordenada está de azul

101 115 30 63 47 20

101 115 30 63 47 20

Menor
elemento



Trocando a posição do menor
elemento com o primeiro

Parte
ordenada

20

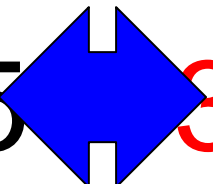
115	30	63	47	101
-----	----	----	----	-----

Parte a ser ordenada

20 115 30 63 47 101

20 115 30 63 47 101

Menor
elemento

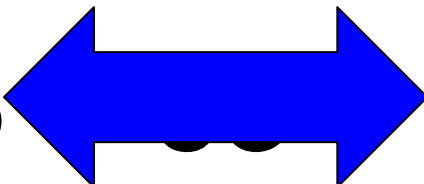
20 115  30 63 47 101

20 30 115 63 47 101

Trocando a posição do menor
elemento com o primeiro

20 30 115 63 47 101

Menor
elemento

20 30 115  47 101

20 30 47 63 115 101

Trocando a posição do menor
elemento com o primeiro

20 30 47 63 115 101

Menor
elemento

20 30 47 63 115 101

Trocando a posição do menor
elemento com o primeiro

20 30 47 63 115 101

Menor
elemento

20 30 47 63 115  101

20 30 47 63 101

115

Trocando a posição do menor
elemento com o primeiro

20 30 47 63 101 115

O algoritmo terminou? Por que?

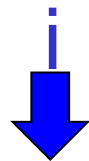
Algoritmo em C *like*

```
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);  
}
```

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

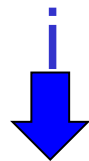


101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

true: $0 < 5$



101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

↓ menor
↓ i

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

↓ menor

↓ i ↓ j

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

true: $1 < 6$

↓ menor
↓ i ↓ j

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

false: 101 > 115

↓ menor

↓ i

↓ j

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

↓ menor
↓ i

↓ j

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

true: $2 < 6$

↓ menor
↓ i

↓ j

101	115	30	63	47	20
0	1	2	3	4	5

Algoritmo em C *like*

```

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);
}
    
```

true: 101 > 30

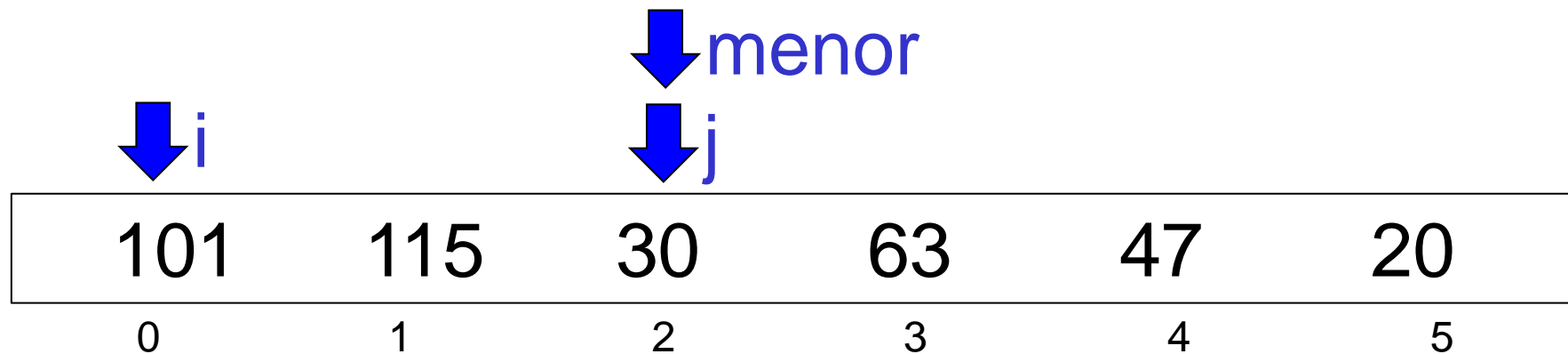
↓ menor
↓ i

↓ j

101	115	30	63	47	20
0	1	2	3	4	5

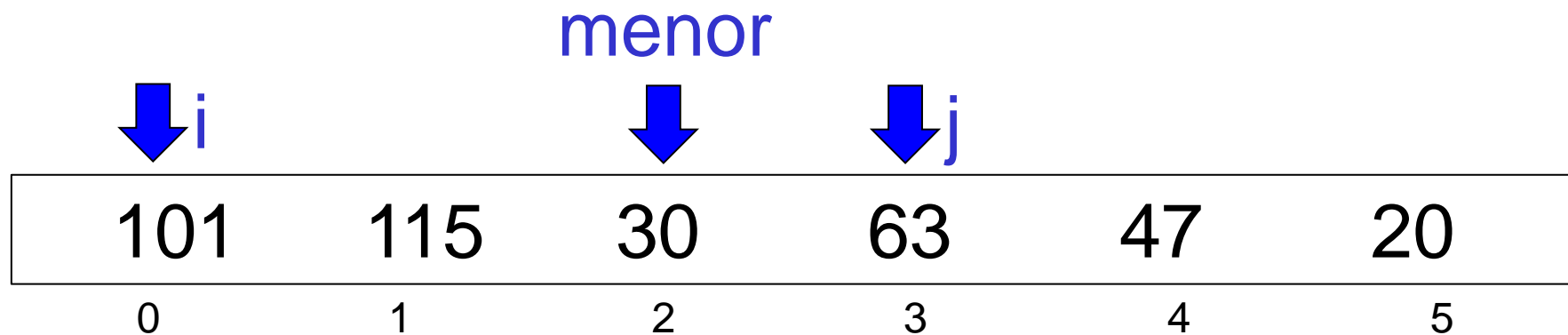
Algoritmo em C *like*

```
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);  
}
```



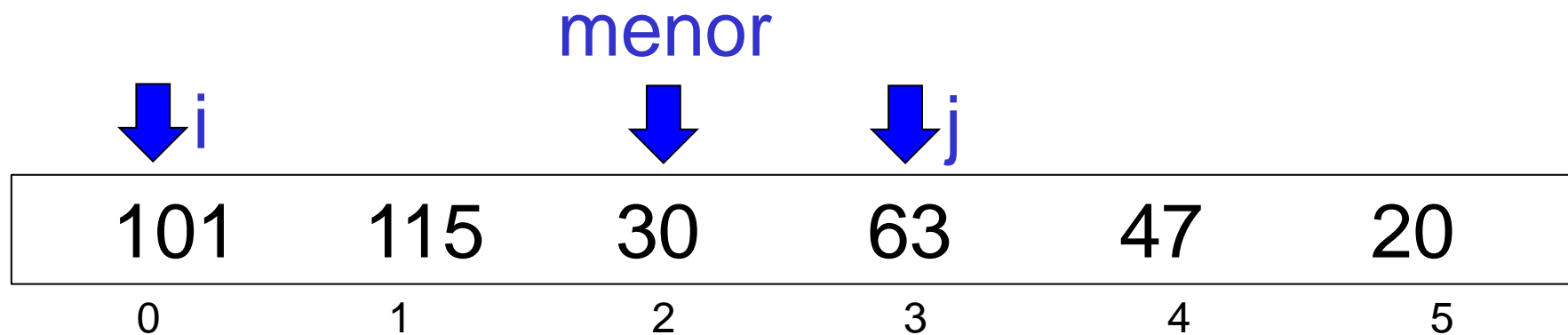
Algoritmo em C *like*

```
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);
}
```



Algoritmo em C *like*

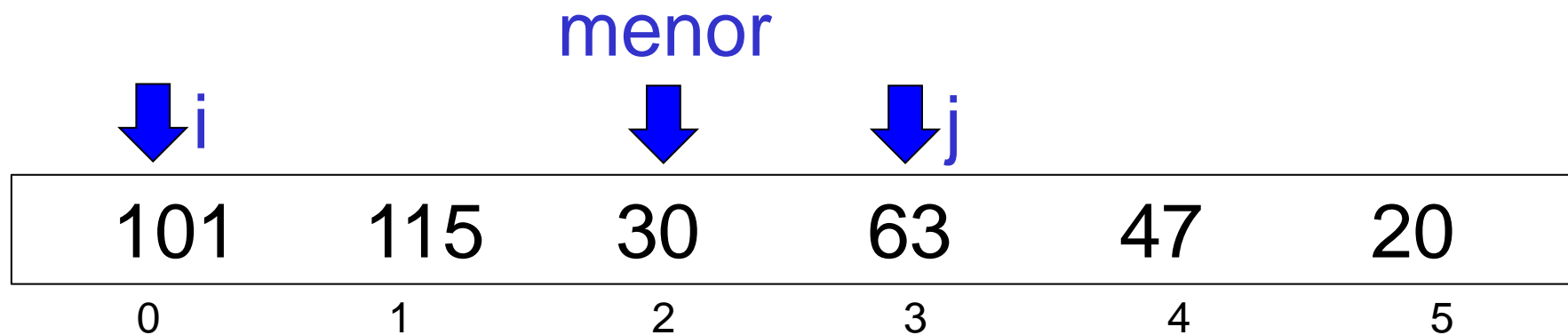
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

true: $3 < 6$ 

Algoritmo em C *like*

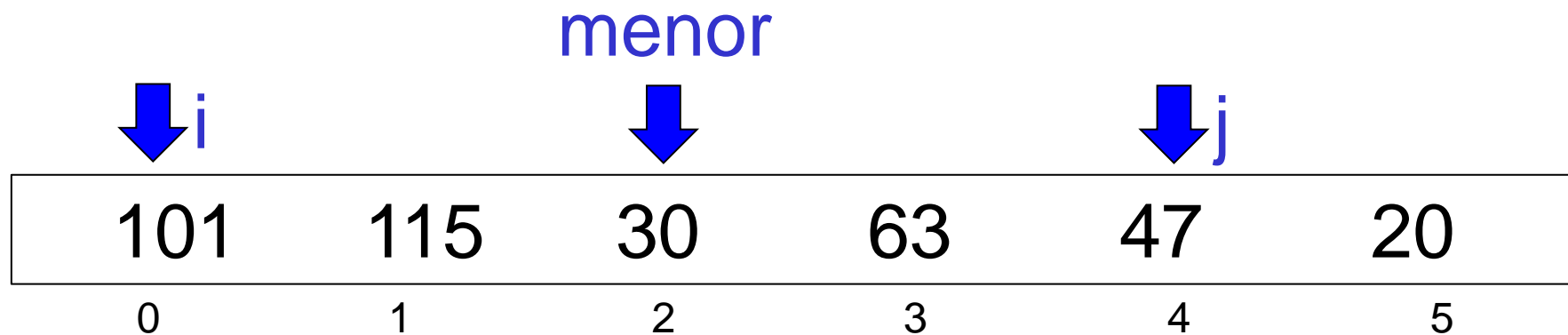
```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap(menor, i);
}
```

false: $30 > 63$



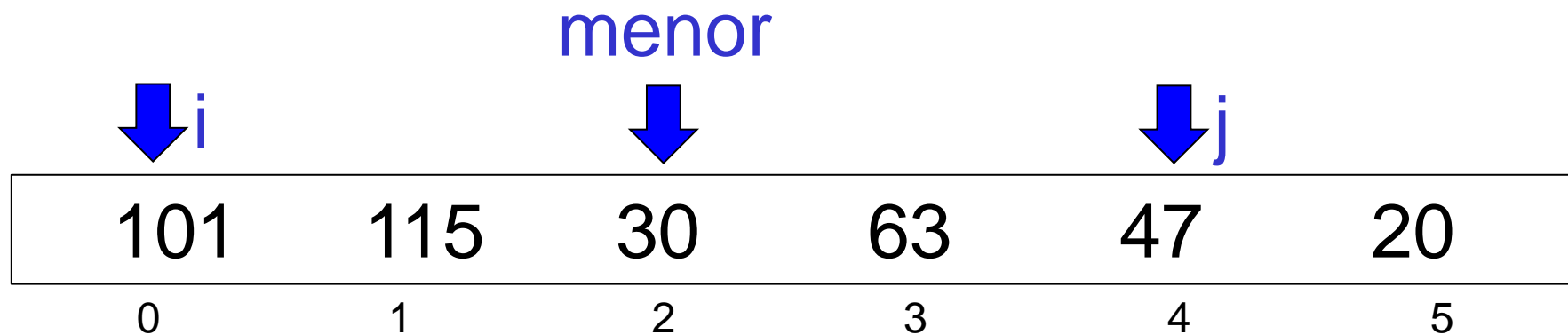
Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

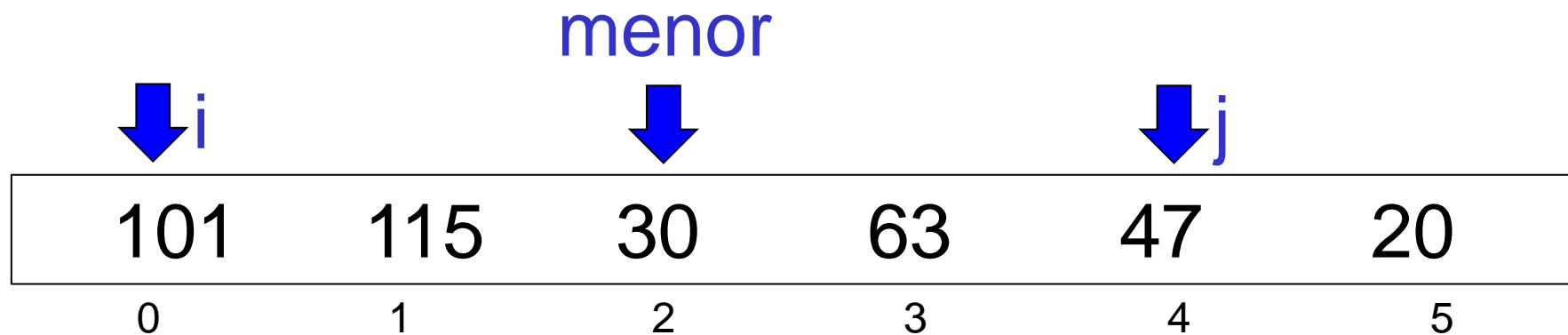
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

true: $4 < 6$ 

Algoritmo em C *like*

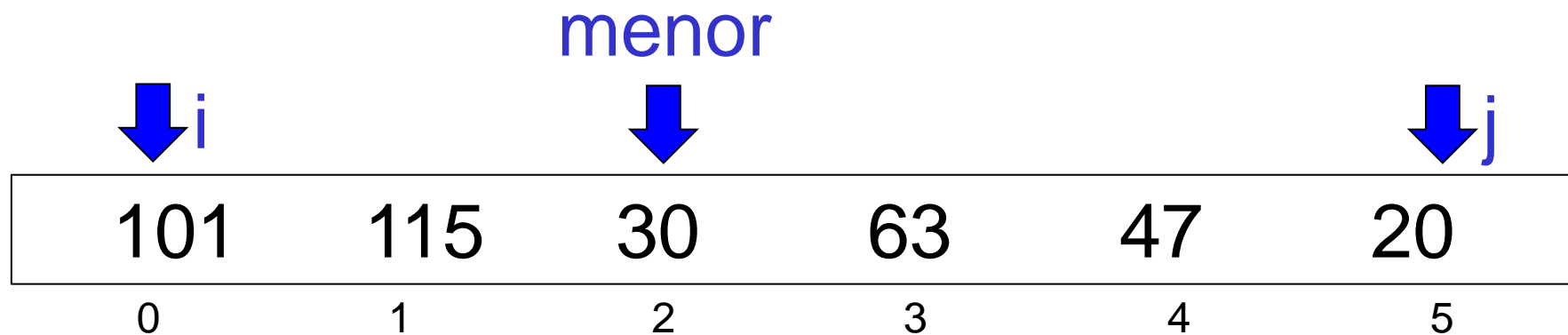
```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap(menor, i);
}
```

false: $30 > 47$



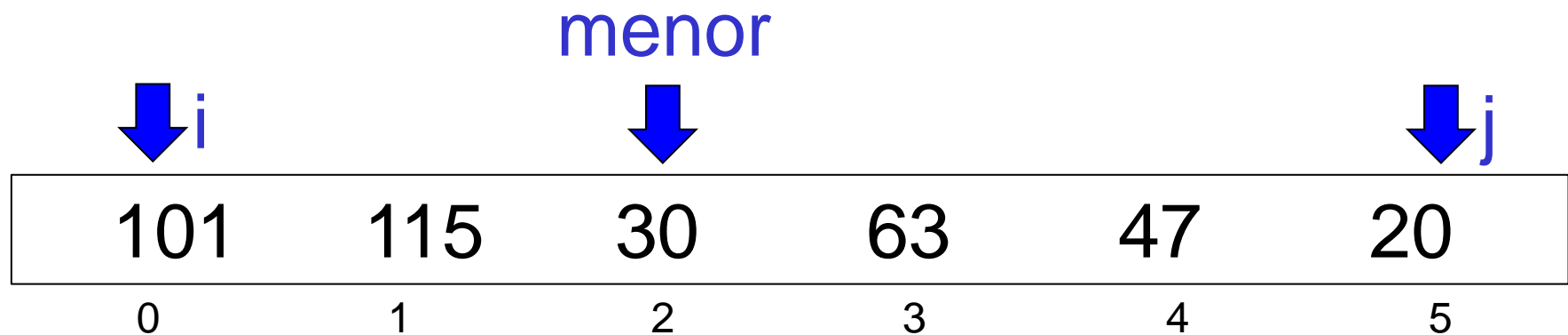
Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

```
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);  
}
```

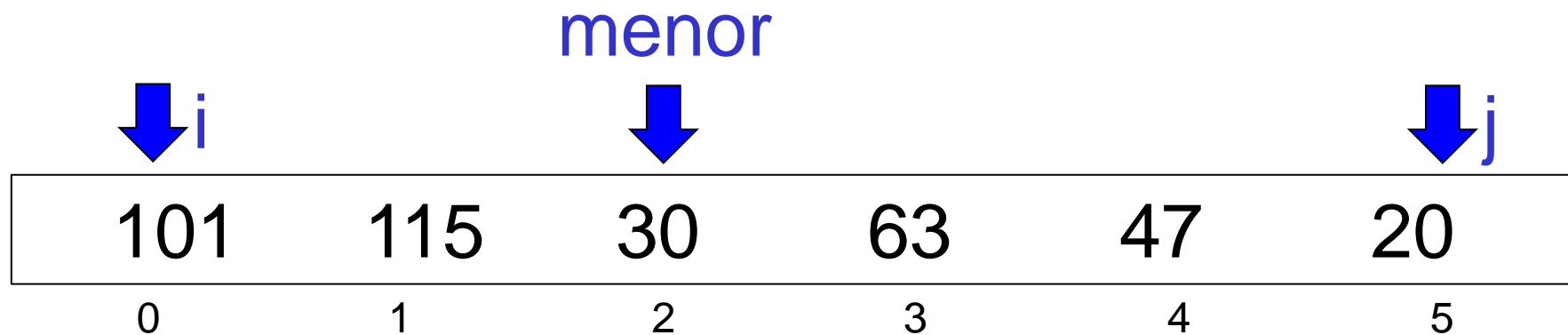
true: $5 < 6$ 

Algoritmo em C *like*

```

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);
}
    
```

true: $30 > 20$

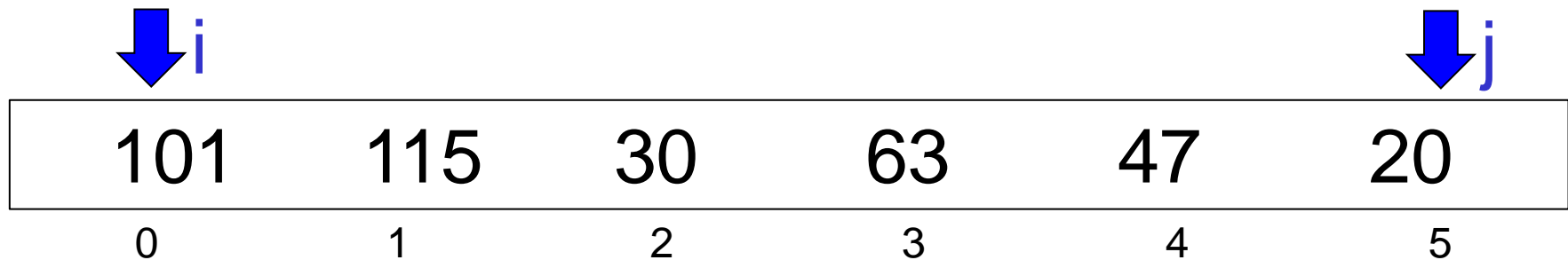


Algoritmo em C *like*

```

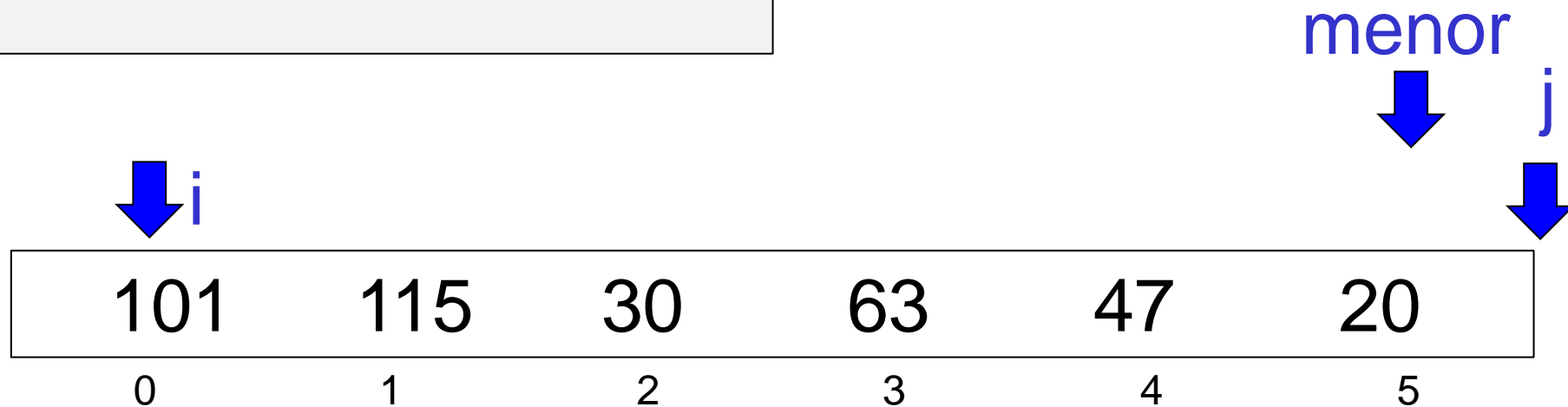
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap(menor, i);
}
    
```

true: $30 > 20$



Algoritmo em C *like*

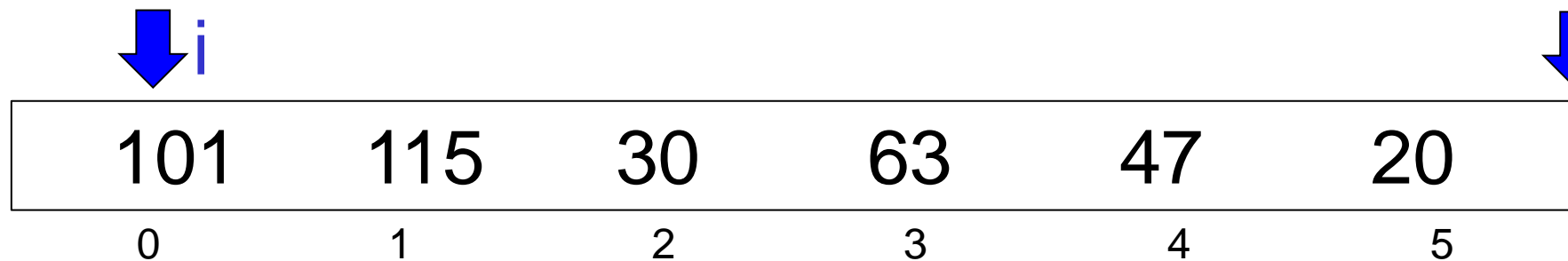
```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap(menor, i);
}
```



Algoritmo em C *like*

```
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);
}
```

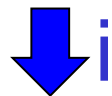
false: $6 < 6$



Algoritmo em C *like*

```
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);  
}
```

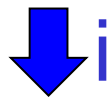
menor



20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

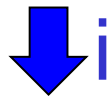


20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

true: $1 < 5$



20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

↓ menor
↓ i

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```



20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

true: $2 < 6$ 

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```

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);
}
    
```

true: $115 > 30$



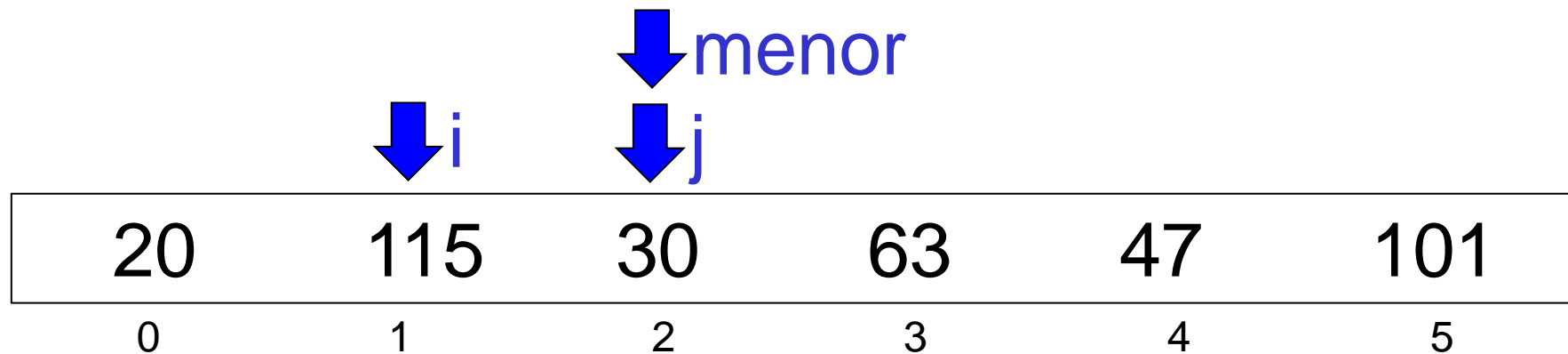
20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```

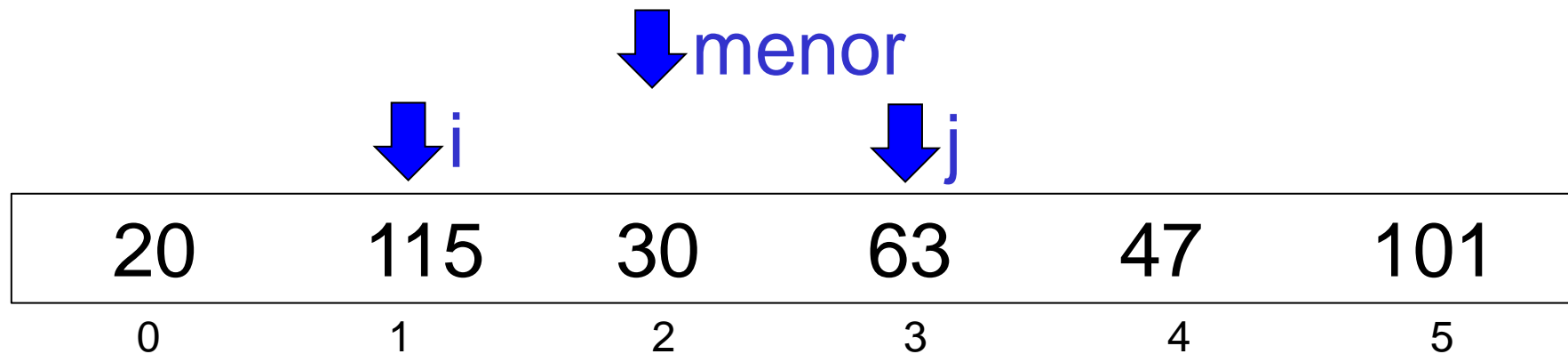
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap(menor, i);
}
    
```

true: 115 > 30



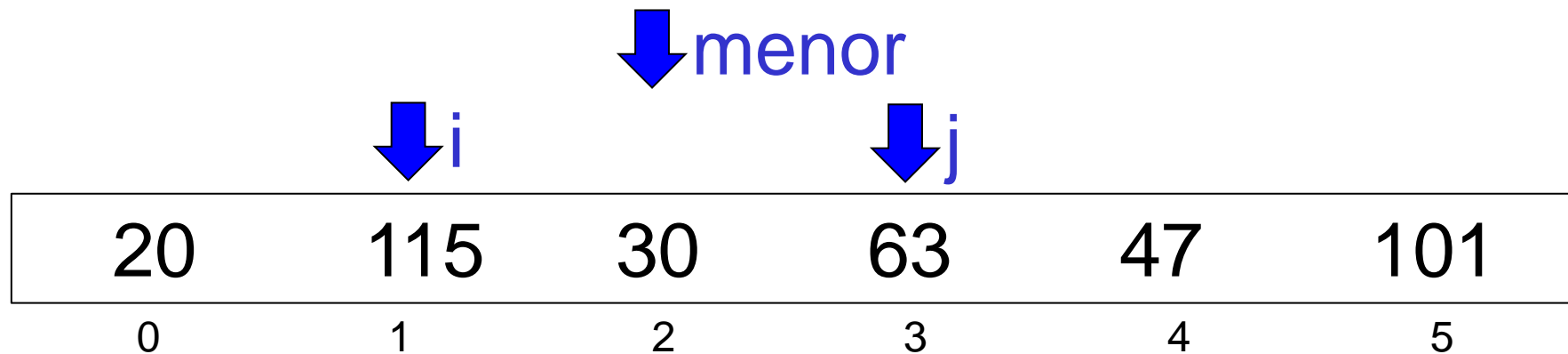
Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

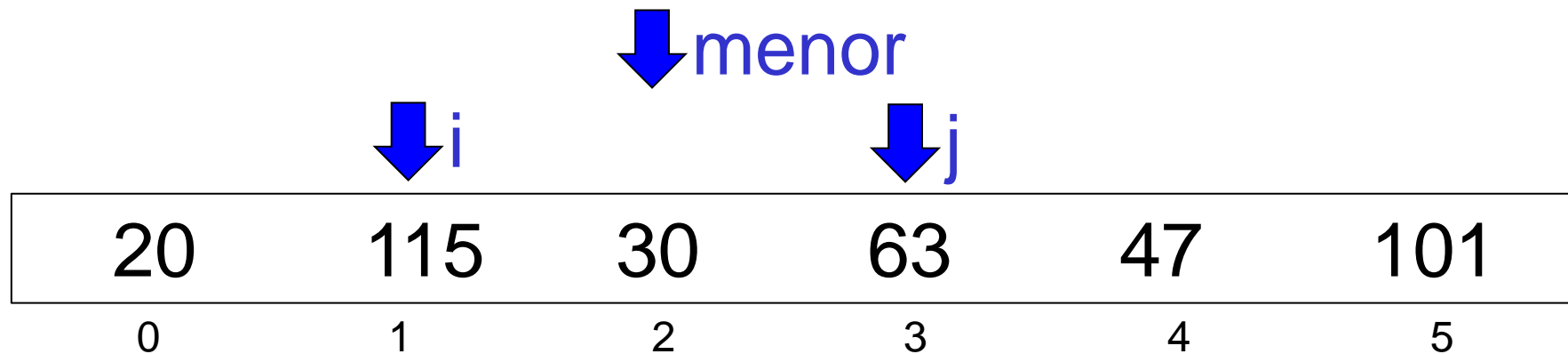
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

true: $3 < 6$ 

Algoritmo em C *like*

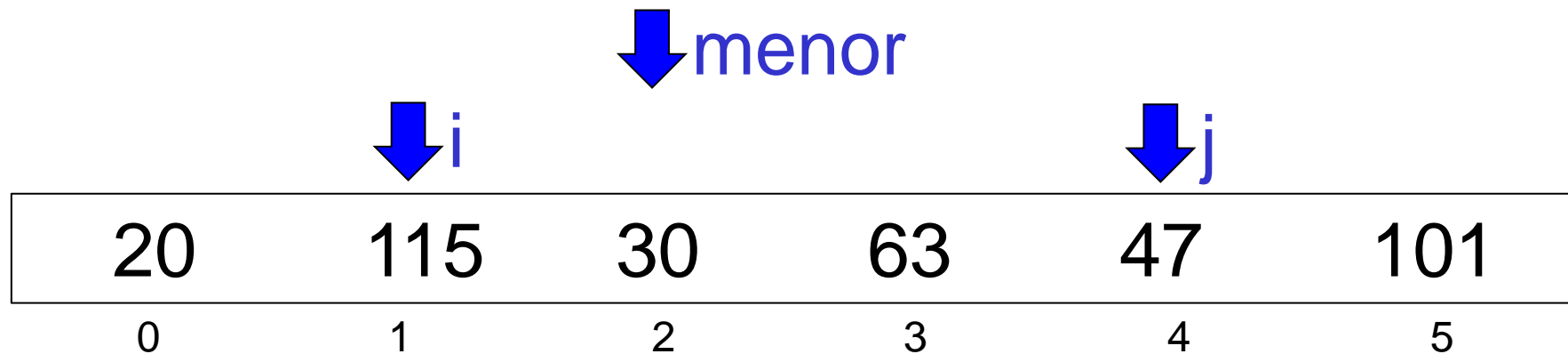
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

false: $30 > 63$



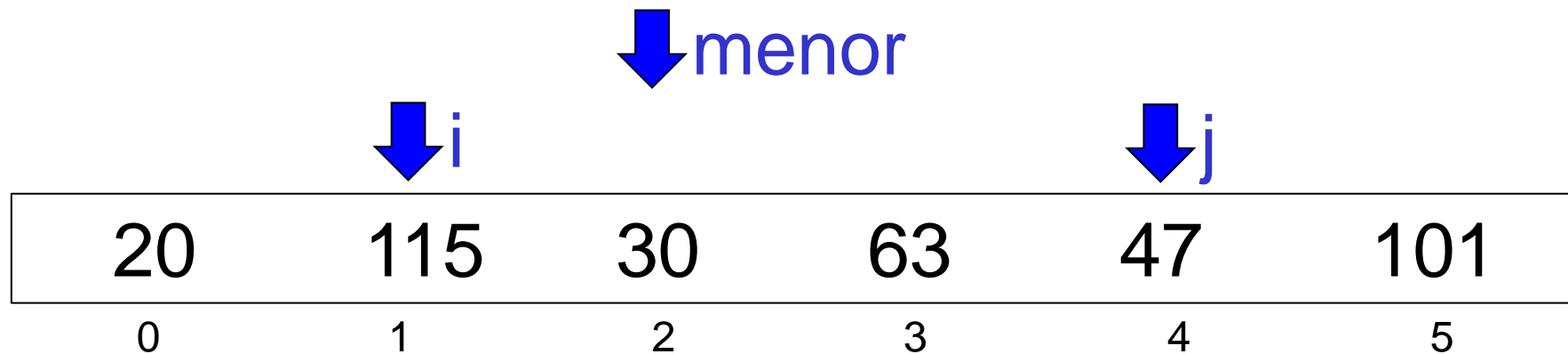
Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

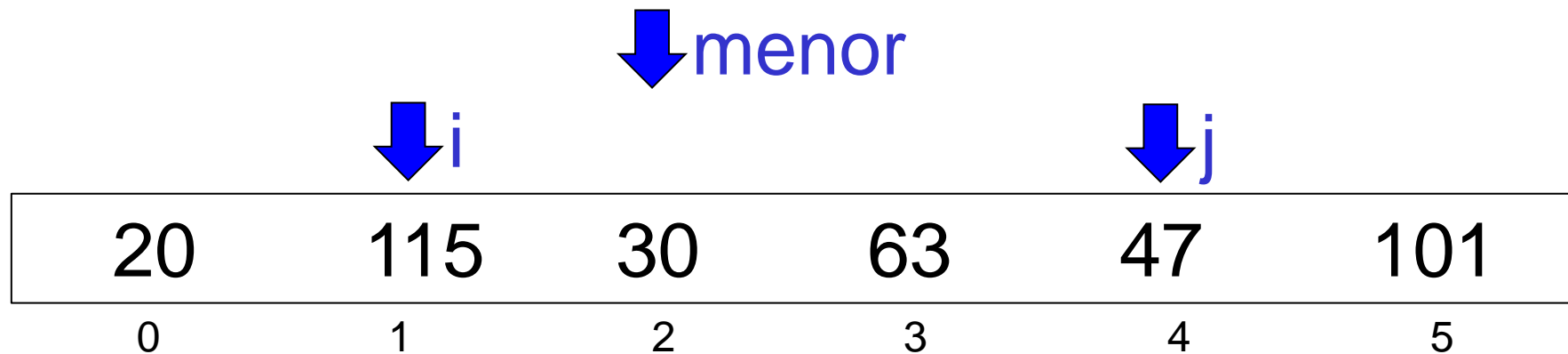
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

true: $4 < 6$ 

Algoritmo em C *like*

```
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);
}
```

false: 30 > 47



Algoritmo em C *like*

```
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);  
}
```

↓ menor

↓ i

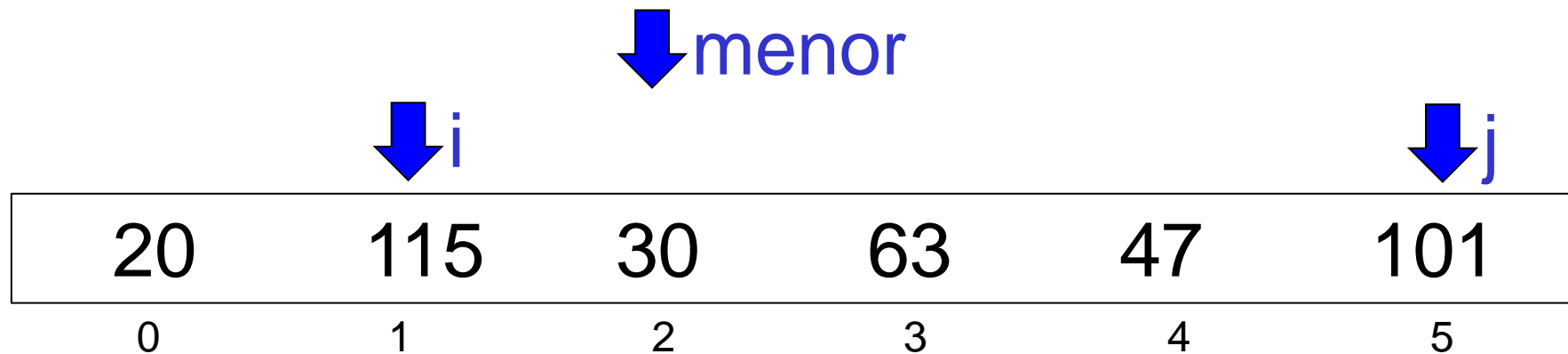
↓ j

20	115	30	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);
}
```

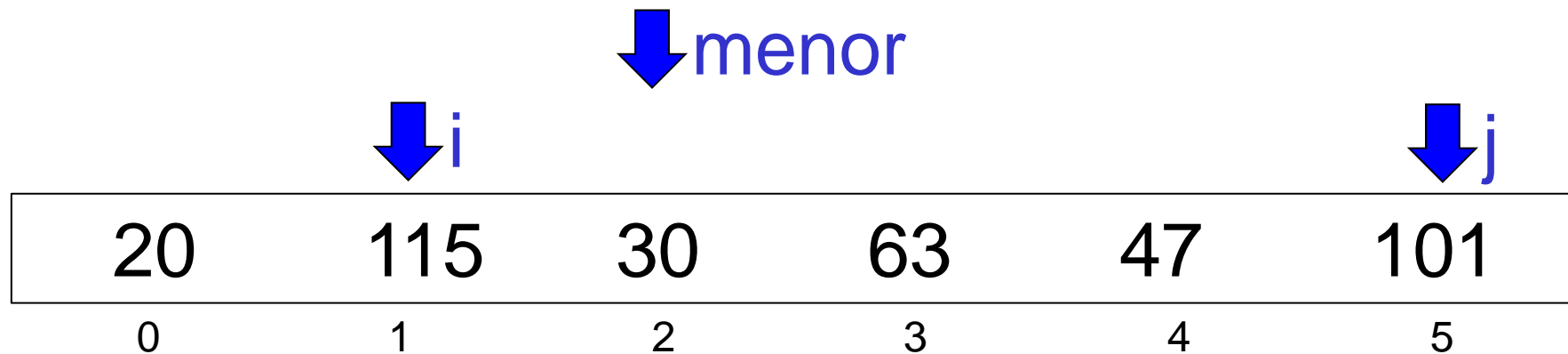
true: $5 < 6$



Algoritmo em C *like*

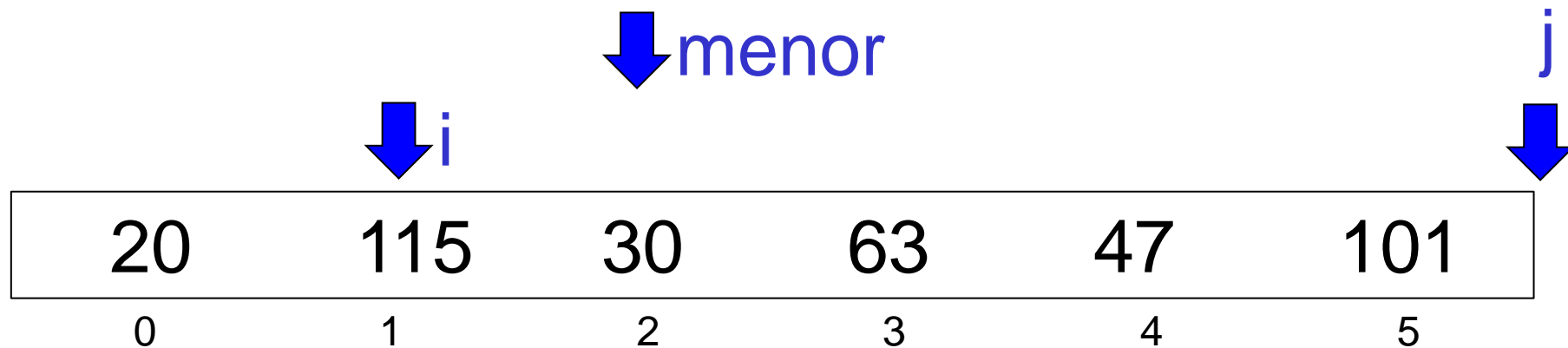
```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap(menor, i);
}
```

false: $30 > 101$



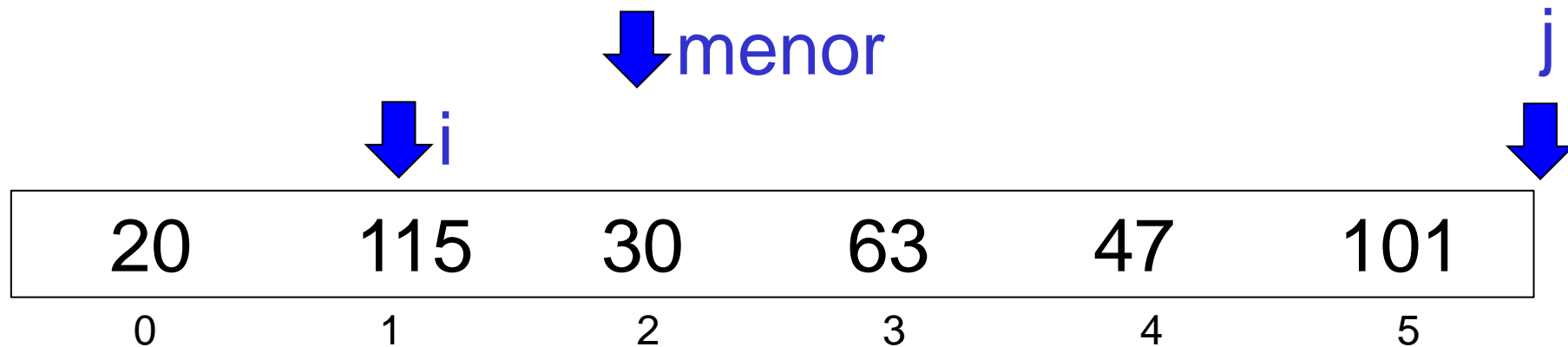
Algoritmo em C *like*

```
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);  
}
```



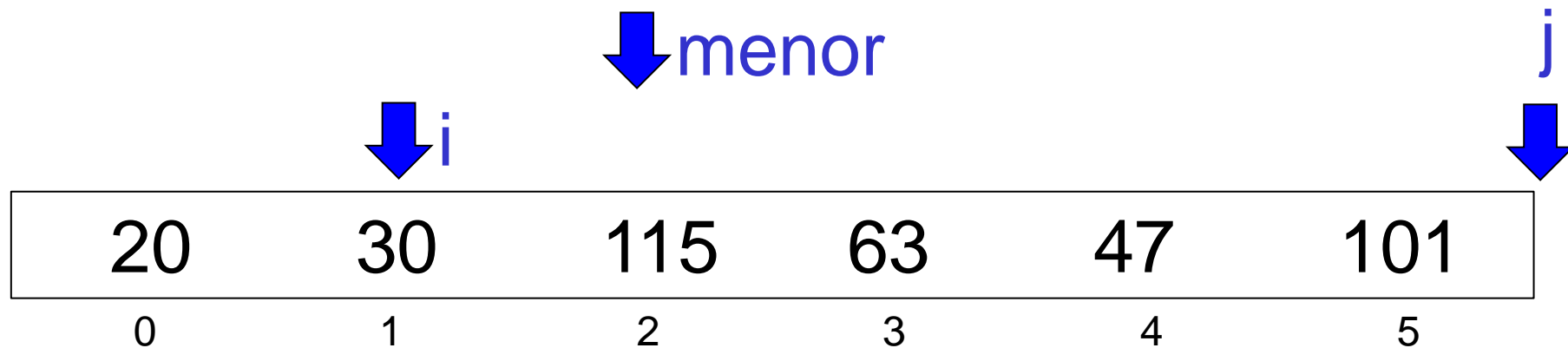
Algoritmo em C *like*

```
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);  
}
```

false: $6 < 6$ 

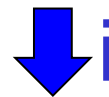
Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

```
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);  
}
```

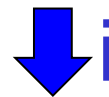


20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

true: $2 < 5$



20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

↓ menor
↓ i

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

↓ menor
↓ i ↓ j

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);
}
```

true: $3 < 6$

↓ menor
↓ i ↓ j

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```

true: $115 > 63$

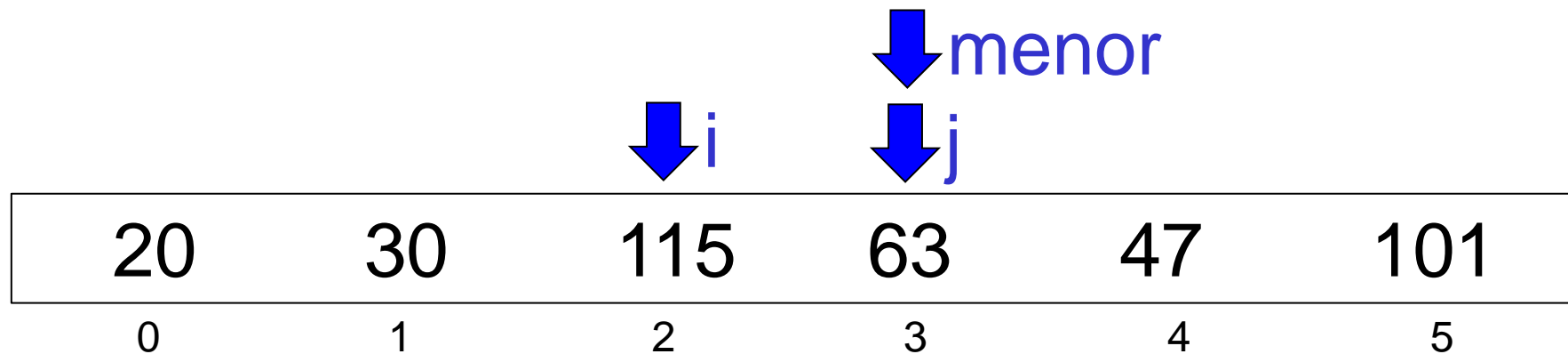
↓ menor
↓ i ↓ j

20	30	115	63	47	101
0	1	2	3	4	5

Algoritmo em C *like*

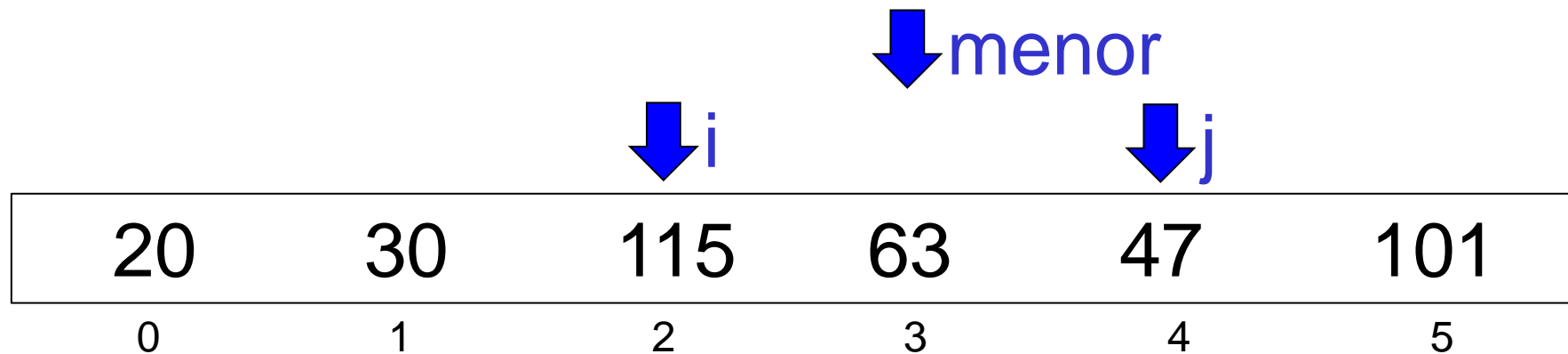
```

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);
}
    
```



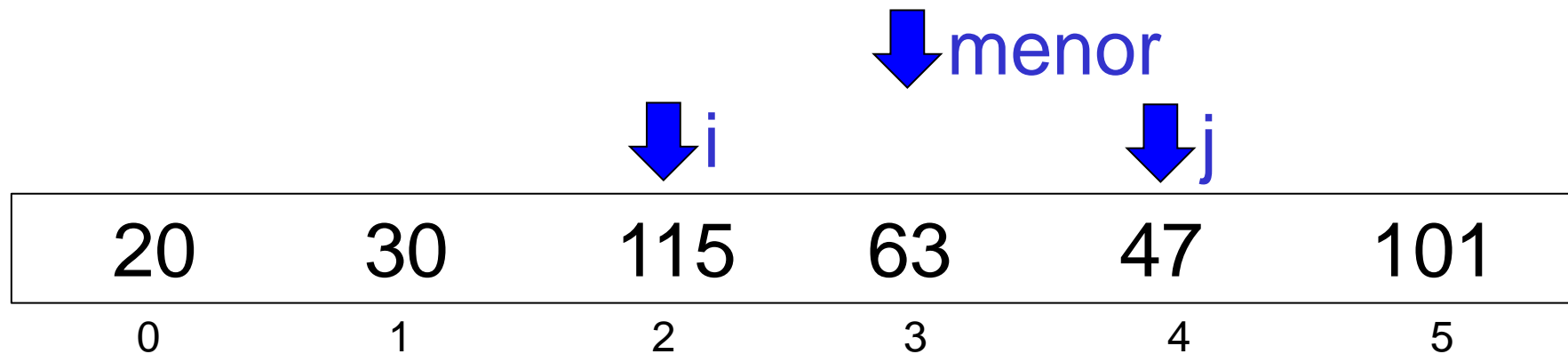
Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

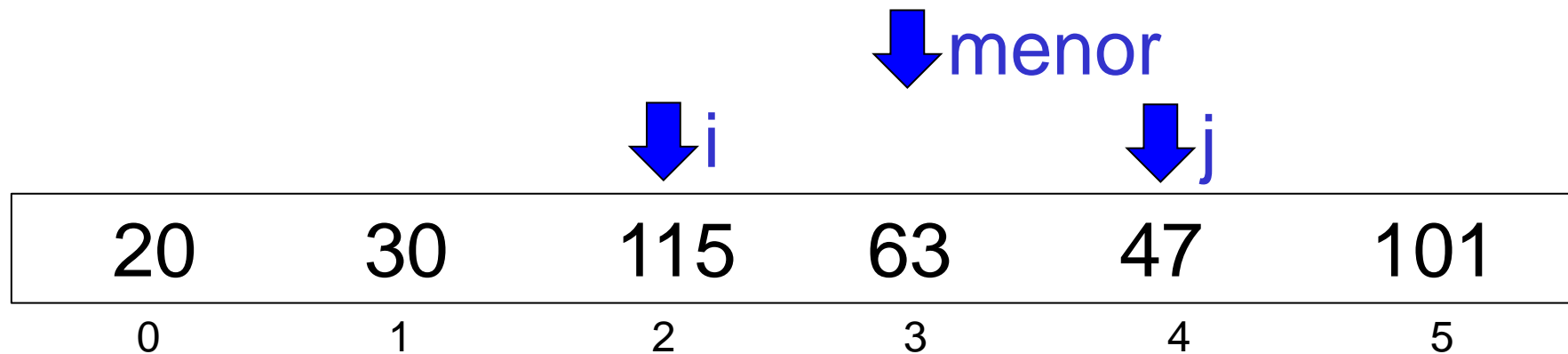
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

true: $4 < 6$ 

Algoritmo em C *like*

```
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);  
}
```

true: $63 > 47$

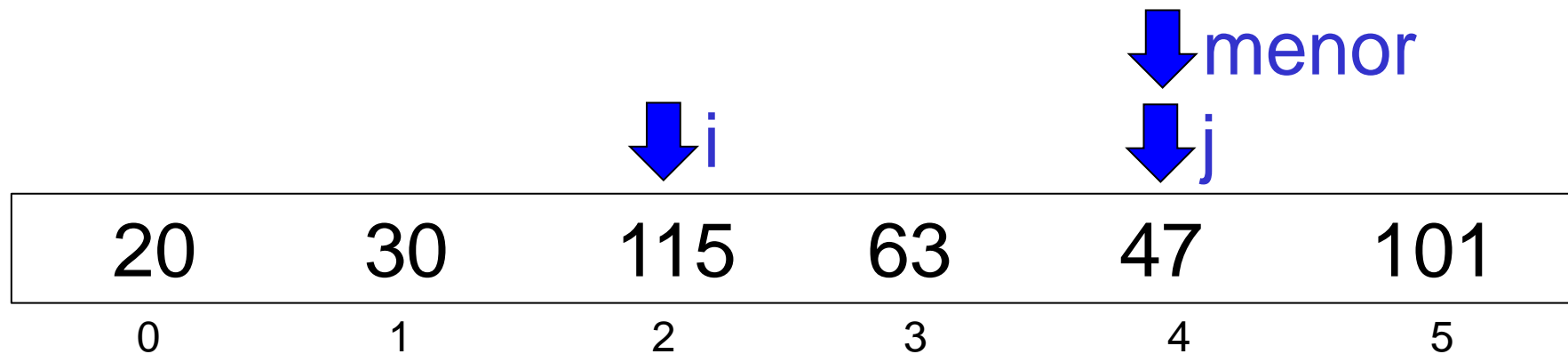


Algoritmo em C *like*

```

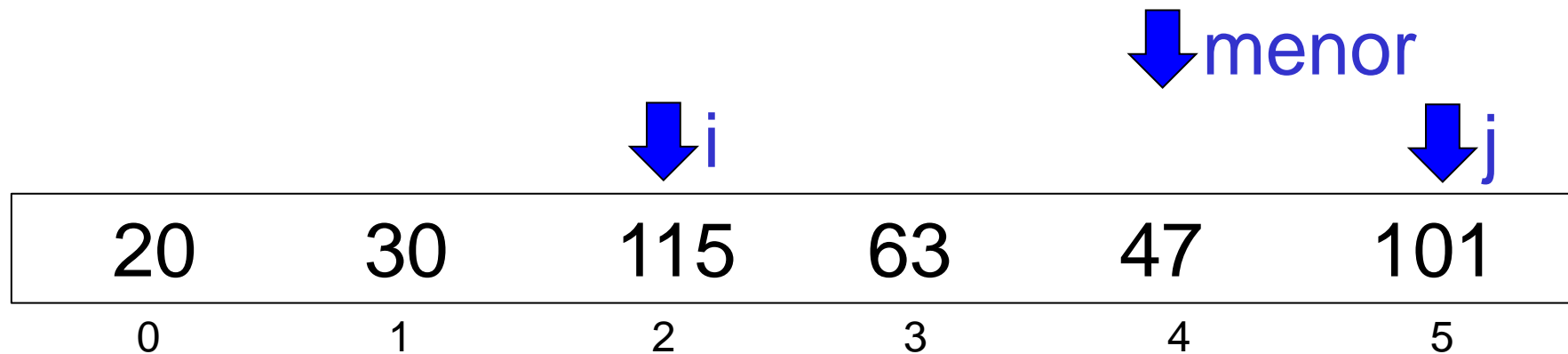
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap(menor, i);
}
    
```

true: 63 > 47



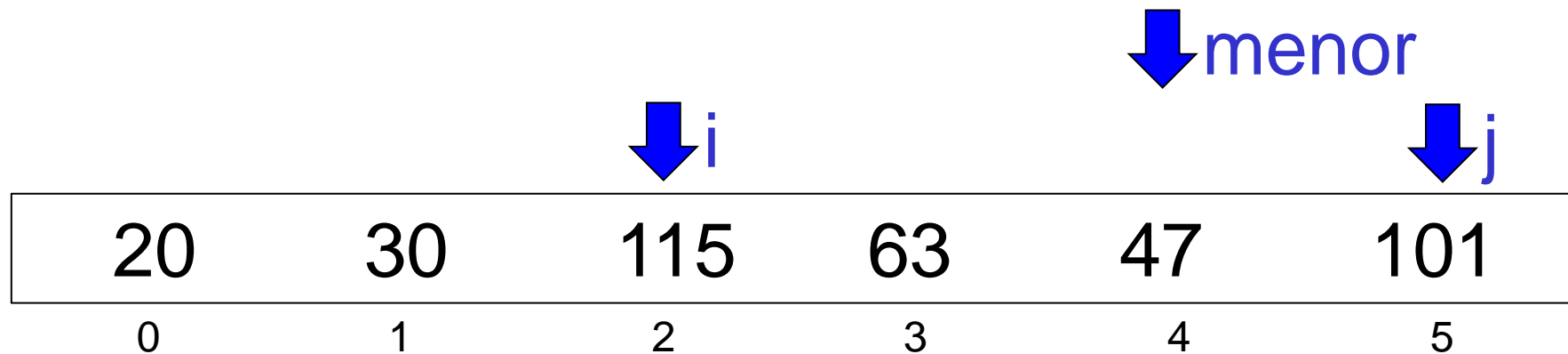
Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

```
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);  
}
```

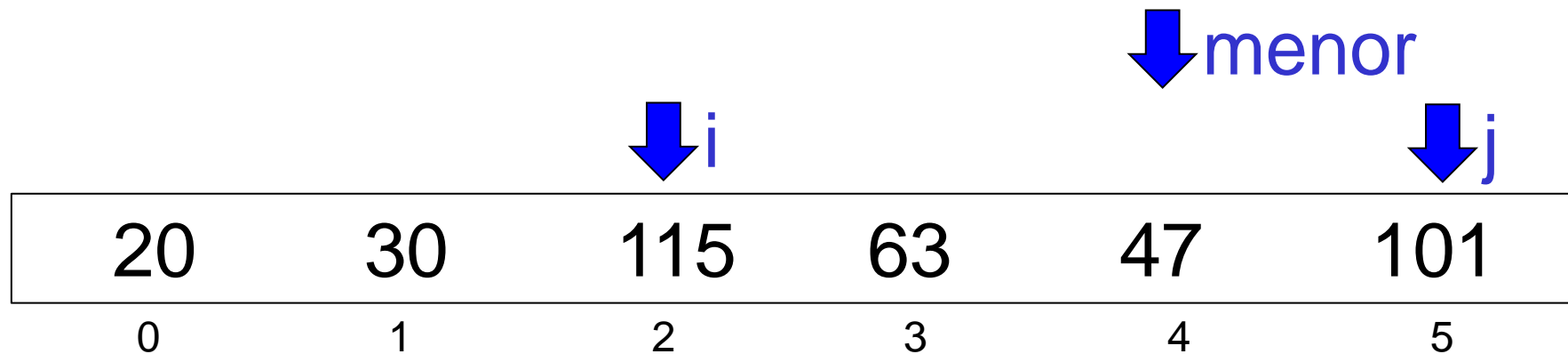
true: $5 < 6$ 

Algoritmo em C *like*

```

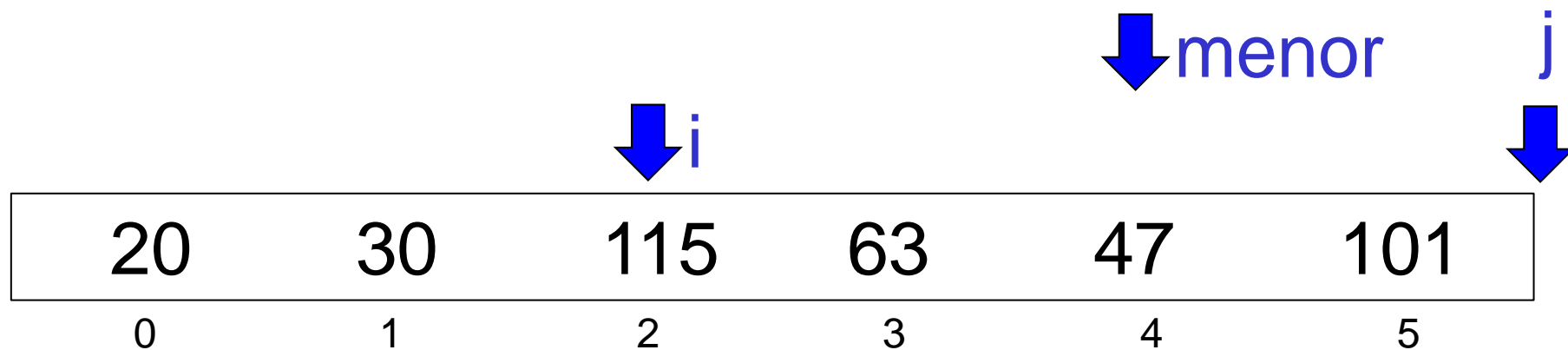
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap(menor, i);
}
    
```

false: $47 > 101$



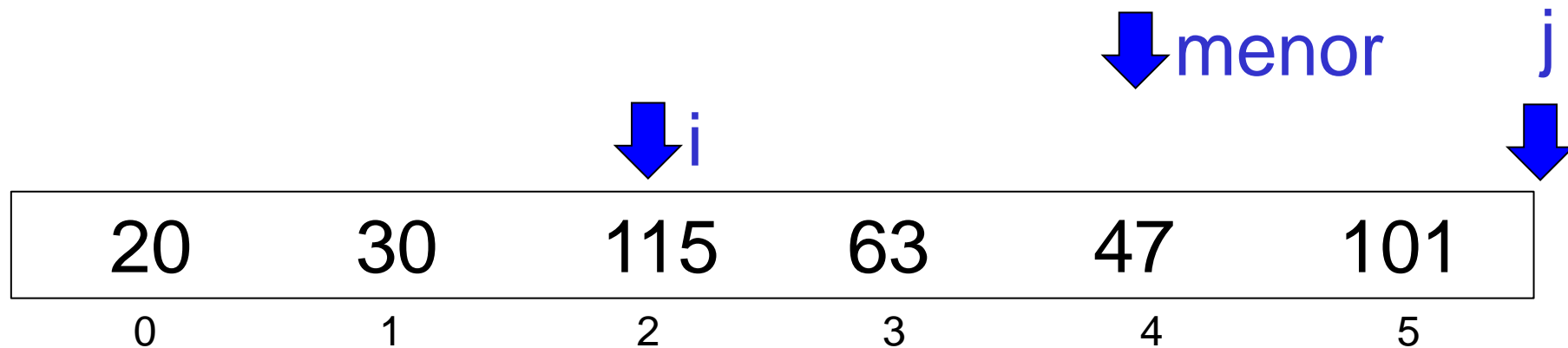
Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

```
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);  
}
```

false: $6 < 6$ 

Algoritmo em C *like*

```
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);  
}
```

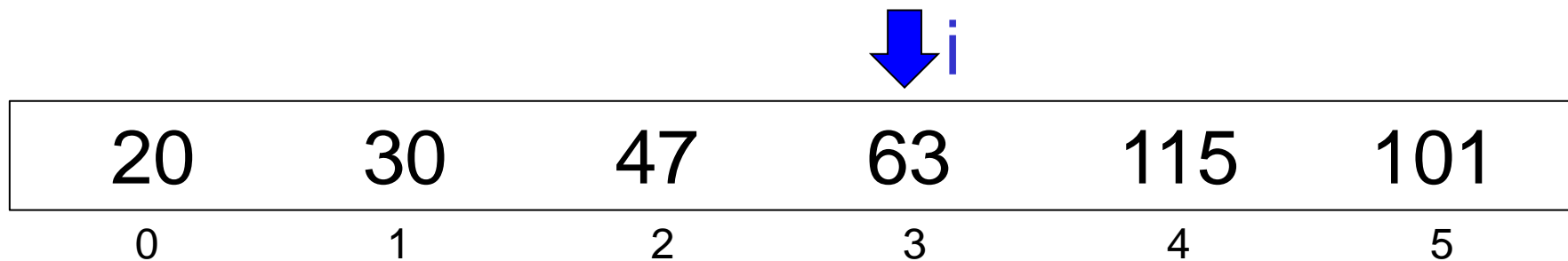
↓ menor

↓ i

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo em C *like*

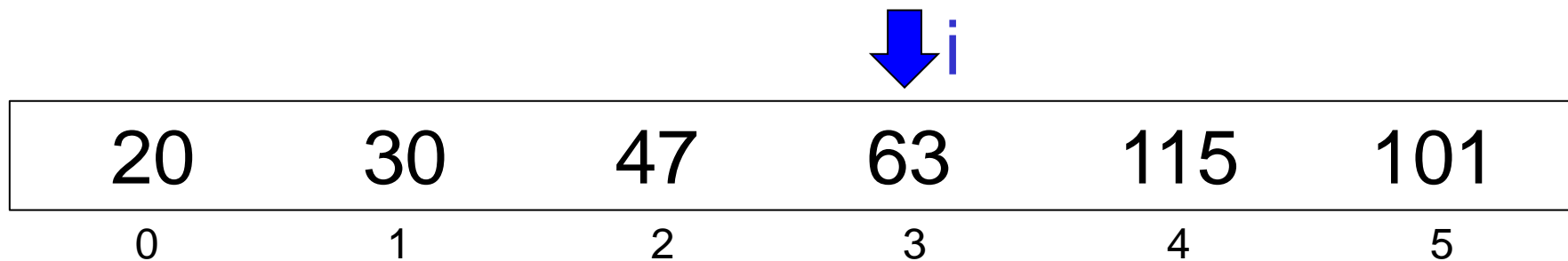
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```



Algoritmo em C *like*

```
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);  
}
```

true: $3 < 5$



Algoritmo em C *like*

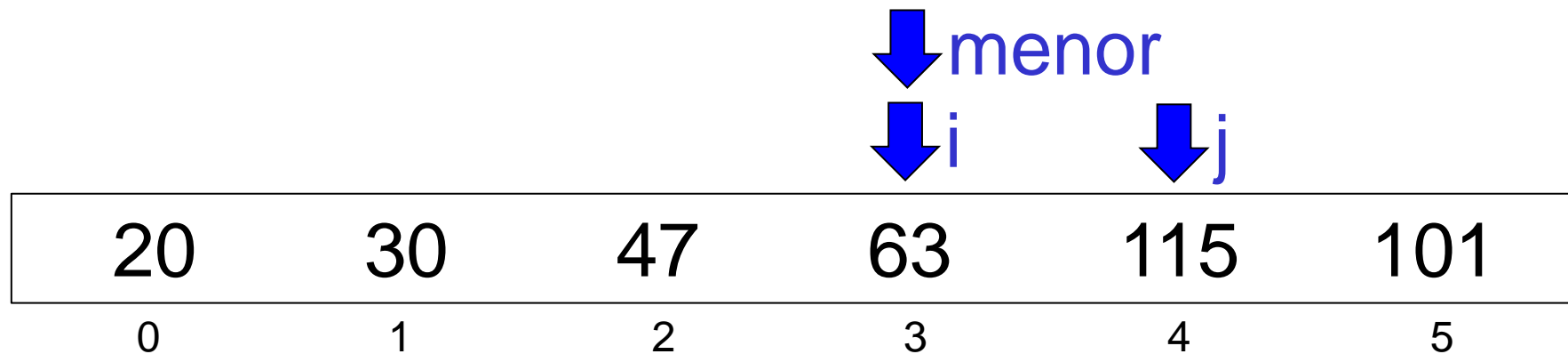
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

↓ menor
↓ i

20	30	47	63	115	101
0	1	2	3	4	5

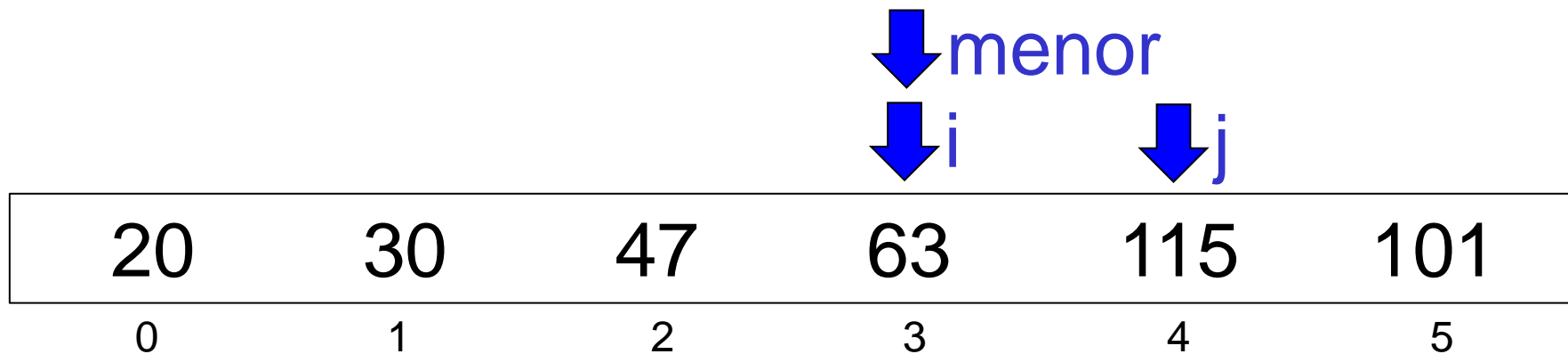
Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

```
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);  
}
```

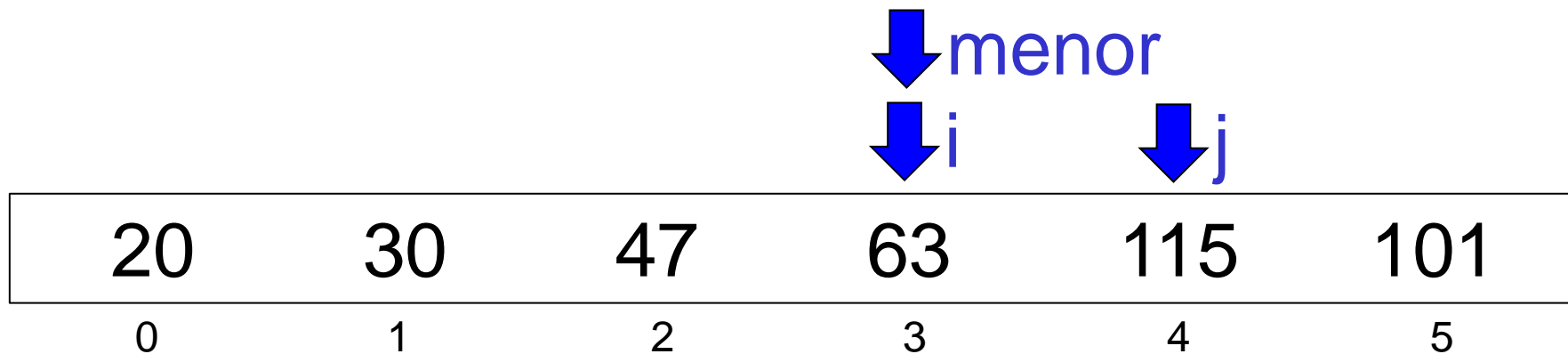
true: $4 < 6$ 

Algoritmo em C *like*

```

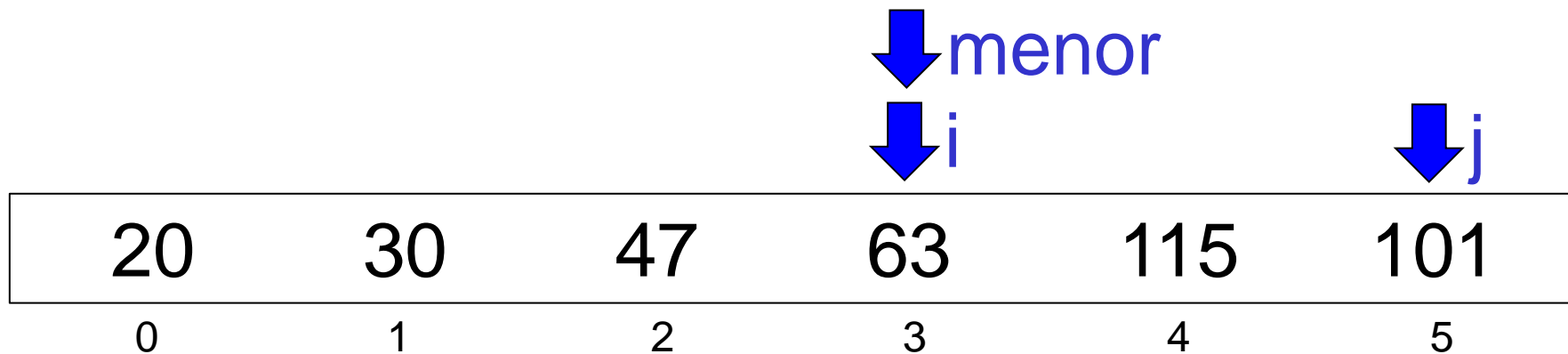
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++){
        if (array[menor] > array[j]){
            menor = j;
        }
    }
    swap(menor, i);
}
    
```

false: $63 > 115$



Algoritmo em C *like*

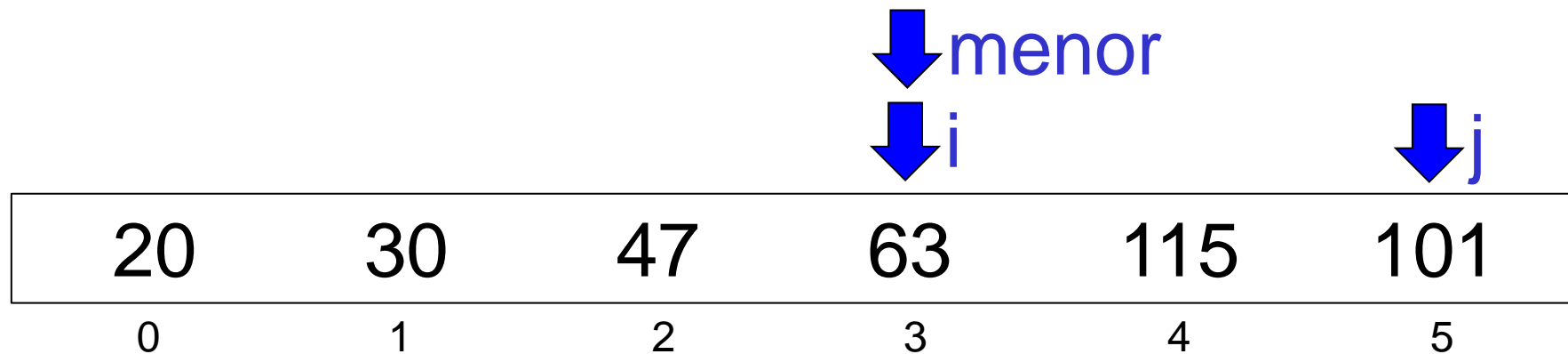
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```



Algoritmo em C *like*

```
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);
}
```

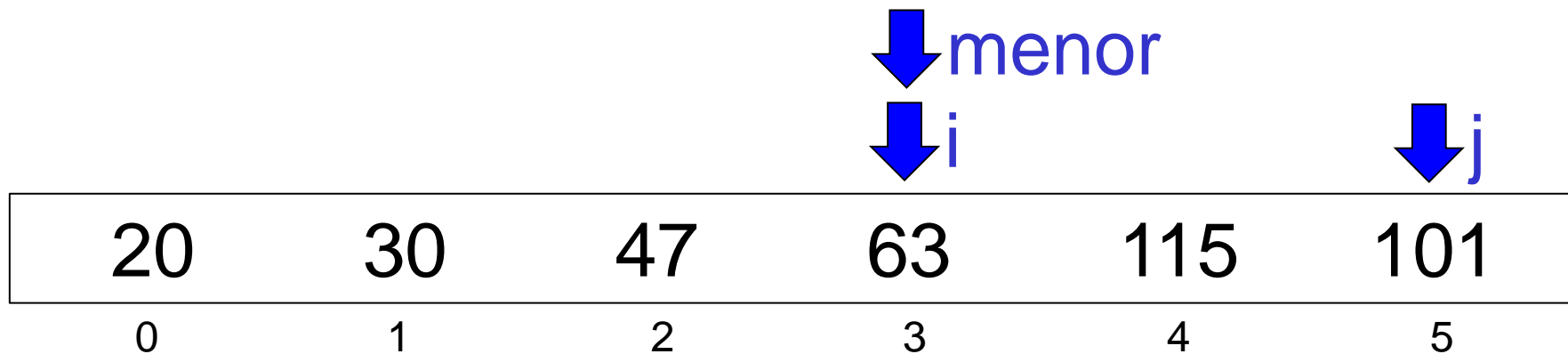
true: $5 < 6$



Algoritmo em C *like*

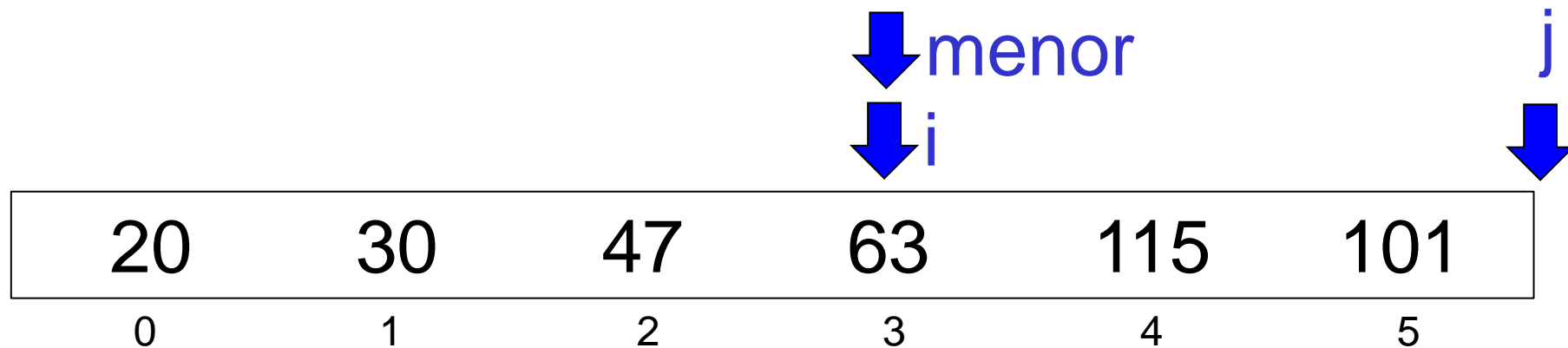
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

false: $63 > 101$



Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

```
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);
}
```

false: $6 < 6$

20	30	47	63	115	101
0	1	2	3	4	5

↓ menor
↓ i

j ↓

Algoritmo em C *like*

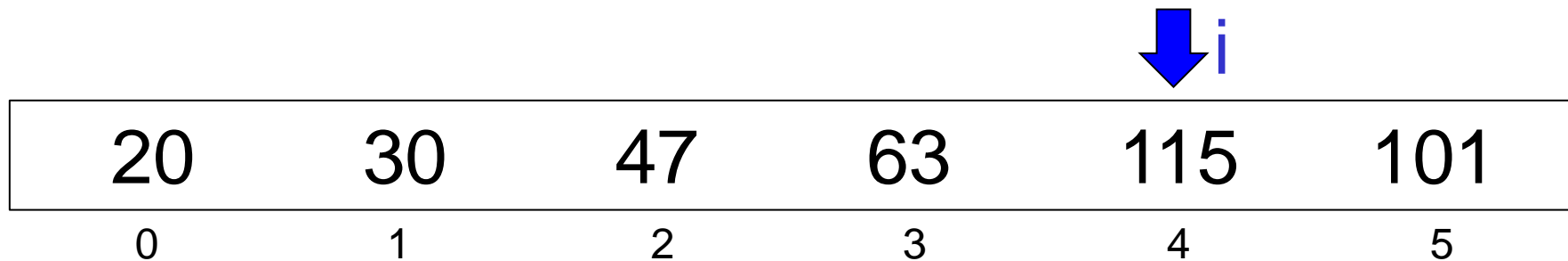
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

↓ menor
↓ i

20	30	47	63	115	101
0	1	2	3	4	5

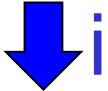
Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

```
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);  
}
```

true: $4 < 5$ 

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo em C *like*

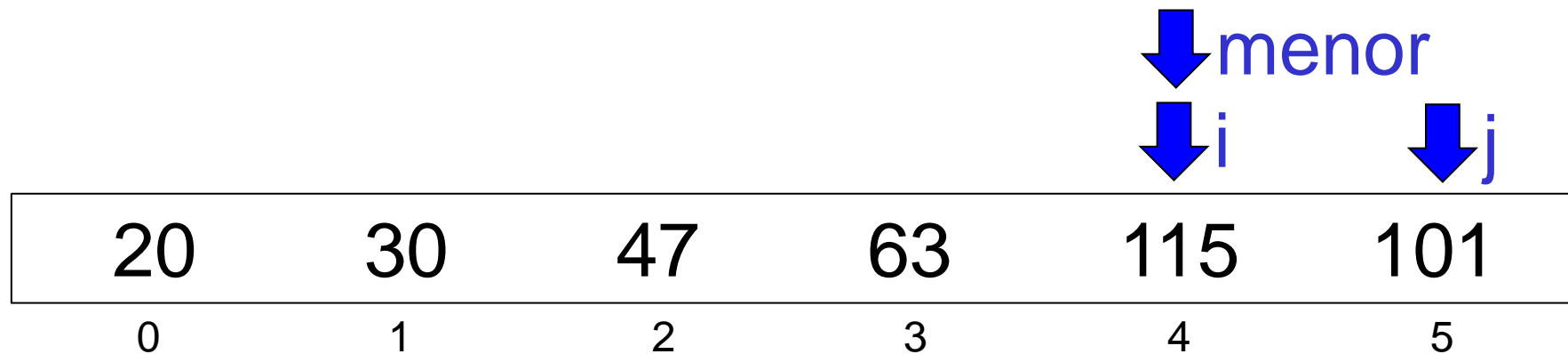
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

↓ menor
↓ i

20	30	47	63	115	101
0	1	2	3	4	5

Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

```
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);  
}
```

true: $5 < 6$

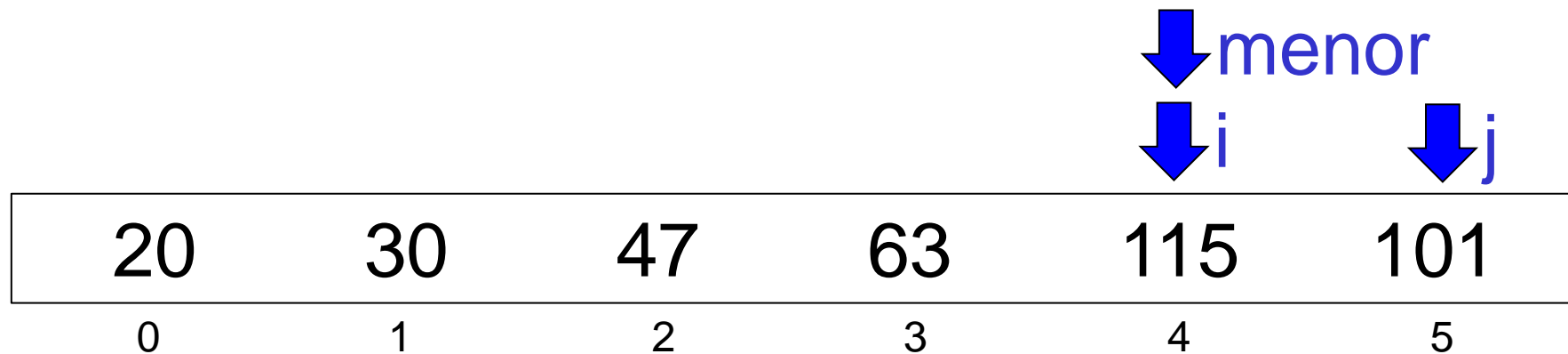
20	30	47	63	115	101
0	1	2	3	4	5

↓ menor
↓ i ↓ j

Algoritmo em C *like*

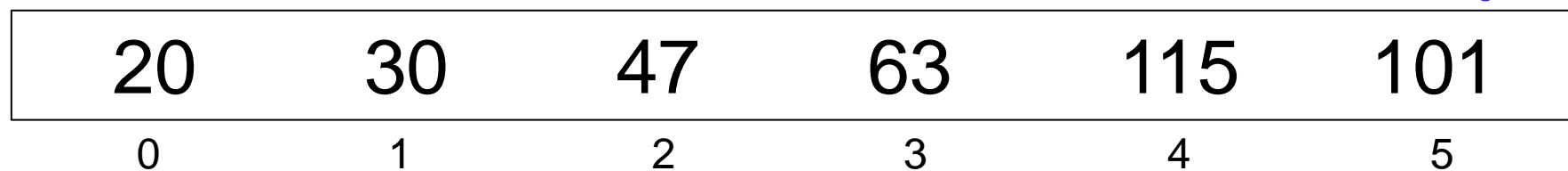
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```

true: $115 > 101$



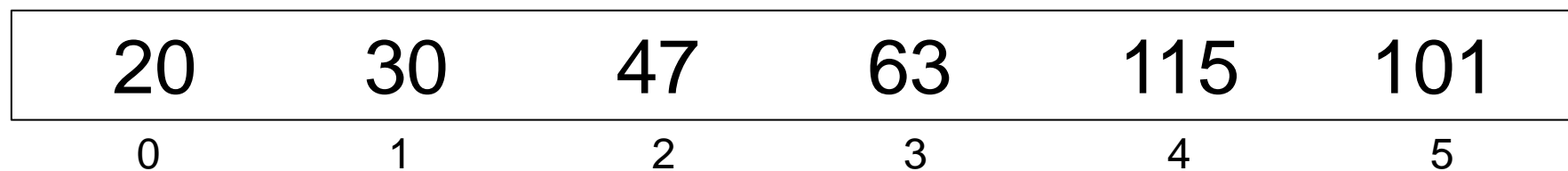
Algoritmo em C *like*

```
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);  
}
```



Algoritmo em C *like*

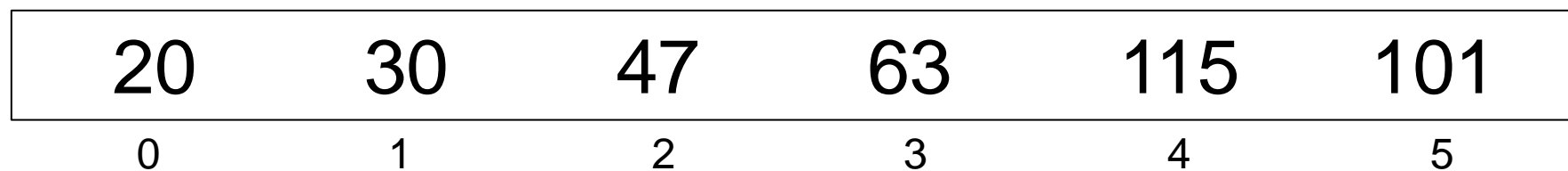
```
for (int i = 0; i < (n - 1); i++) {  
    int menor = i;  
    for (int j = (i + 1); j < n; j++){  
        if (array[menor] > array[j]){  
            menor = j;  
        }  
    }  
    swap(menor, i);  
}
```



Algoritmo em C *like*

```
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);
}
```

false: $6 < 6$



Algoritmo em C *like*

```

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);
}
    
```

20	30	47	63	101	115
0	1	2	3	4	5

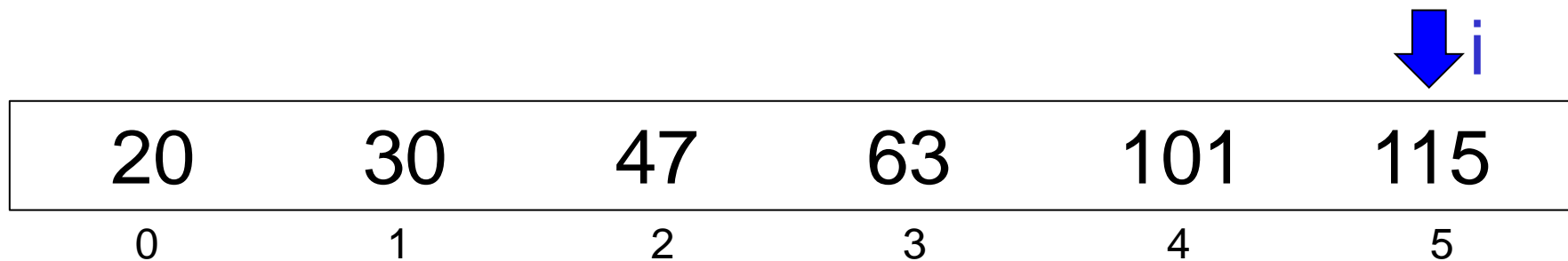


menor



Algoritmo em C *like*


```
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);  
}
```



Algoritmo em C *like*

```
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);  
}
```

false: $5 < 5$



20	30	47	63	101	115
0	1	2	3	4	5

Análise do Número de Comparações

```
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);  
}
```


Análise do Número de Comparações

```
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);  
}
```

Temos somente um comando de comparação entre elementos do array

Análise do Número de Comparações

```
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);  
}
```

Executamos o laço interno
 $(n - (i + 1))$ vezes

Ou seja, $(n - i - 1)$ vezes

Análise do Número de Comparações

```
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);  
}
```

Executamos o laço interno
 $(n - (i + 1))$ vezes

Ou seja, $(n - i - 1)$ vezes

Exemplo: $n = 5$

Para $i = 0$, os valores de j serão 1, 2, 3 e 4 $(5 - 1 - 0) = 4$ vezes

Para $i = 1$, os valores de j serão 2, 3 e 4 $(5 - 1 - 1) = 3$ vezes

Para $i = 2$, os valores de j serão 3 e 4 $(5 - 1 - 2) = 2$ vezes

Para $i = 3$, o valor de j será 4 $(5 - 1 - 3) = 1$ vez

Análise do Número de Comparações

```
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);  
}
```

Executamos o laço externo $(n - 1)$ vezes

Ou seja, os valores de i serão 0, 1, 2 e 3

Análise do Número de Comparações

- Como o laço interno é executado $(n - 1 - i)$ vezes e o externo $(n - 1)$ vezes, logo:

$$C(n) = \frac{n^2}{2} - \frac{n}{2} = \Theta(n^2)$$

Análise do Número de Comparações

- Como o laço interno é executado $(n - 1 - i)$ vezes e o externo $(n - 1)$ vezes, logo:

$$C(n) =$$

Resolução de somatórios

Livro do Knuth

Livro do Cormen

Análise do Número de Comparações

- Como o laço interno é executado $(n - 1 - i)$ vezes e o externo $(n - 1)$ vezes, logo:

$$= \underset{i=0}{(n-1-0)} + \underset{i=1}{(n-1-1)} + \underset{i=2}{(n-1-2)} + \dots + \underset{i=n-2}{1}$$

$$C(n) = (n - 1) + (n - 2) + (n - 3) + \dots + (n - (n - 2) - 1)$$

Análise do Número de Comparações

- Assim, temos:

$$i = 0 \quad i = 1 \quad i = 2 \quad \dots \quad i = n - 2$$

$$C(n) = (n - 1) + (n - 2) + (n - 3) + \dots + (n - (n - 2) - 1)$$

$$C(n) = (n - 1) + (n - 2) + (n - 3) + \dots + 1$$

Análise do Número de Comparações

- Sendo,

$$C(n) = (n - 1) + (n - 2) + (n - 3) + \dots + 1$$

- Podemos colocar da forma abaixo?

$$C(n) = \sum_{i=0}^{i < n-1} (n - i - 1)$$

Análise do Número de Comparações

- Sendo,

$$C(n) = (n - 1) + (n - 2) + (n - 3) + \dots + 1$$

- Podemos colocar da forma abaixo?

$$C(n) = \sum_{i=0}^{n-1} (n - i - 1)$$

- E assim?

$$C(n) = \sum_{i=0}^{n-2} (n - i - 1)$$

Análise do Número de Comparações

- Em Matemática Discreta, vamos aprender que:

$$C(n) = \sum_{i=0}^{n-2} (n - i - 1) = \sum_{i=0}^{n-2} (n) - \sum_{i=0}^{n-2} (i) - \sum_{i=0}^{n-2} (1)$$

Análise do Número de Comparações

- Agora, podemos fazer as duas substituições abaixo, certo?

$$C(n) = \sum_{i=0}^{n-2} (n - i - 1) = \sum_{i=0}^{n-2} (n) - \sum_{i=0}^{n-2} (i) - \sum_{i=0}^{n-2} (1)$$

$n * (n-1)$
 $- (n-1)$

Análise do Número de Comparações

- Agora, podemos fazer as duas substituições abaixo, certo?

$$C(n) = \sum_{i=0}^{n-2} (n - i - 1) = \sum_{i=0}^{n-2} (n) - \sum_{i=0}^{n-2} (i) - \sum_{i=0}^{n-2} (1)$$

$n * (n-1)$
 $- (n-1)$

- Logo:

$$C(n) = (n - 1)(n) - (n - 1) - \sum_{i=0}^{n-2} (i)$$

Análise do Número de Comparações

- Deturpando o somatório, podemos fazer:

$$C(n) = (n-1)(n) - (n-1) - \sum_{i=0}^{n-2} (i)$$

$$C(n) = (n-1)(n) - (n-1) - \sum_{i=1}^{n-1} (i-1)$$

Análise do Número de Comparações

- Separando o “i” e “-1” em dois somatórios, temos:

$$C(n) = (n - 1)(n) - (n - 1) - \sum_{i=1}^{n-1} (i - 1)$$

$$C(n) = (n - 1)(n) - (n - 1) - \sum_{i=1}^{n-1} (i) + \sum_{i=1}^{n-1} (1)$$

Análise do Número de Comparações

- Resolvendo o segundo somatório, temos:

$$C(n) = (n-1)(n) - (n-1) - \sum_{i=1}^{n-1} (i) + \sum_{i=1}^{n-1} (1)$$

$$C(n) = (n-1)(n) - (n-1) - \sum_{i=1}^{n-1} (i) + (n-1)$$

Análise do Número de Comparações

- Simplificando – $(n-1) + (n-1)$, temos:

$$C(n) = (n-1)(n) - (n-1) - \sum_{i=1}^{n-1} (i) + (n-1)$$

$$C(n) = (n-1)(n) - \sum_{i=1}^{n-1} (i)$$

Análise do Número de Comparações

- Em Matemática Discreta, vamos aprender:

$$\sum_{i=1}^{n-1} (i) = 1 + 2 + \dots + (n-1) = \frac{(n-1)(n)}{2}$$

- Assim:

$$C(n) = (n-1)(n) - \sum_{i=1}^{n-1} (i) = (n-1)(n) - \frac{(n-1)(n)}{2}$$

Análise do Número de Comparações

- Simplificando, temos:

$$C(n) = (n-1)(n) - \frac{(n-1)(n)}{2} = \frac{(n-1)(n)}{2} = \frac{n^2}{2} - \frac{n}{2}$$

- Finalmente:

$$C(n) = \frac{n^2}{2} - \frac{n}{2} = \Theta(n^2)$$

Análise do Número de Movimentações

- O laço externo realiza $(n - 1)$ trocas, ou seja, $3(n - 1)$ movimentações

$$M(n) = 3(n - 1)$$

Análise do Número de Movimentações

- O laço externo realiza $(n - 1)$ trocas, ou seja, $3(n - 1)$ movimentações

$$M(n) = 3(n - 1)$$

No algoritmo de Seleção, é mais fácil calcular o número de movimentações!!!

- Vantagem: o número de movimentações é linear e isso é interessante quando os registros são "grandes"
- Desvantagens:
 - $\Theta(n^2)$ comparações
 - Não há melhor caso
 - Algoritmo não Estável

Exercício

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

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