O PROFESSOR ADVERTE: SLIDES NÃO DEVEM SER UTILIZADOS COMO ÚNICA FONTE PARA ESTUDAR OS CONTEÚDOS MINISTRADOS. PROCURE CONSULTAR, PELO MENOS, A BIBLIOGRAFIA INDICADA NO ÚLTIMO SLIDE.

- Ordenação interna, métodos simples:
  - Ordenação por Seleção;
  - Ordenação por Inserção;
  - Ordenação por Troca (*Bubblesort*);

Prof. Yandre Maldonado e Gomes da Costa
yandre@din.uem.br

Ordenação Interna

- Memória interna (primária) suficiente para armazenar os registros de dados durante o processo de ordenação;
- Medidas de complexidade relevantes em ordenação interna:
  - *C(n)*, número de comparações entre chaves;
  - *M(n)*, número de movimentações de itens no arquivo;

Ordenação Interna

- Considere os seguintes tipos para os algoritmos descritos a seguir:

```
Const
   n=10;

Type
   vetor=array [1..n] of integer;

Var
   Vetor_real: vetor;
   i: integer;
```

Ordenação por Seleção

- Um dos mais simples;
- Princípio: selecione o menor elemento do vetor e troque-o com o item que está na primeira posição do vetor. Repita estas duas operações com os n-1 itens restantes, depois com os n-2 itens, até que reste apenas um elemento.

Ordenação por Seleção

- Exemplo:

	1	**2**	**3**	**4**	**5**	**6**
Chaves Inicias:	O	R	D	E	N	A
I=2	**A**	R	D	E	N	**O**
I=3	A	**D**	**R**	E	N	O
I=4	A	D	**E**	**R**	N	O
I=5	A	D	E	**N**	**R**	O
I=6	A	D	E	N	**O**	**R**

- *Algoritmo:*

```
Procedure Selecao (var A: vetor);
Var
   i, j, x, Min: integer;
Begin
   for i := 1 to n-1 do
   Begin
         min := i;
         for j := i+1 to n do
           if A[j] < A[Min]   then Min := j;
           x := A[Min];
           A[Min] := A[i];
           A[i] := x;
   End;
End;
```

Seleciona o menor

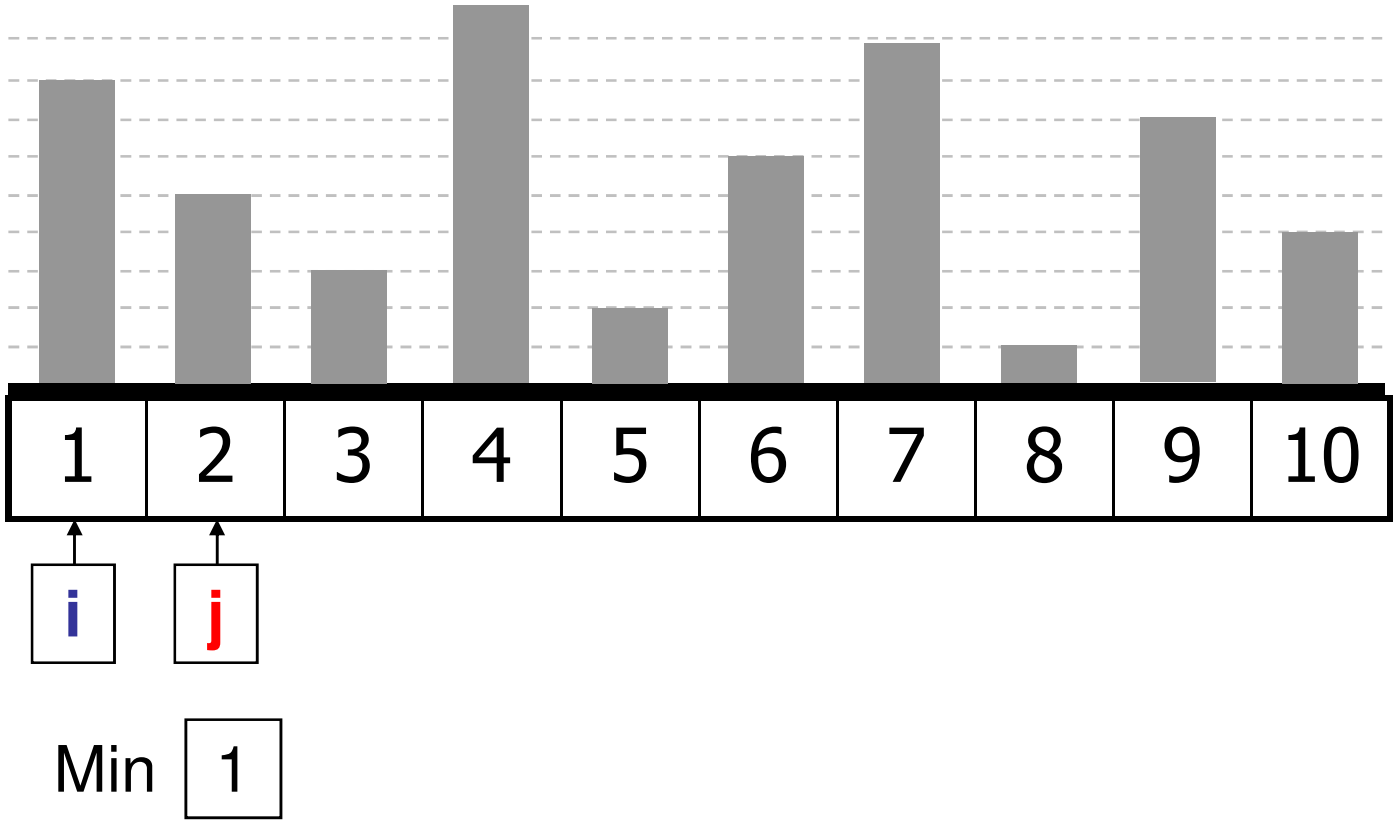
Faz a troca

Ordenação por Seleção

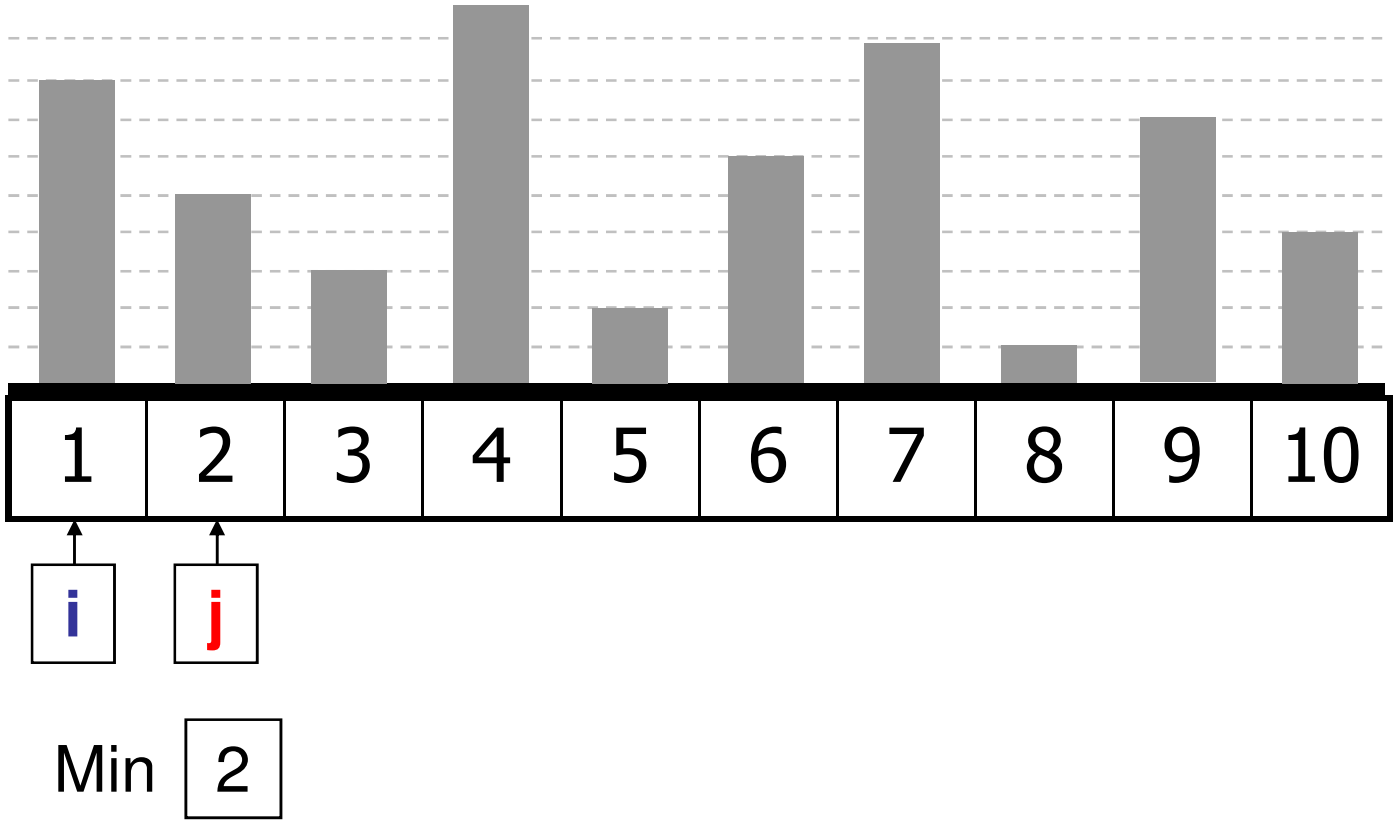
- Análise de complexidade:
  - $C(n) = (n^2/2) – (n/2)$
  - $M(n) = 3\ (n-1)$
- Em condições normais, com chaves do tamanho de uma palavra, este método é bastante interessante para arquivos com até 1000 registros;
- Aspecto negativo: o fato de o arquivo já estar parcialmente ordenado não reduz em nada o custo da ordenação.

```
for i := 1 to n-1 do ←
  Begin
     min := i; ←
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
         x := A[Min];
         A[Min] := A[i];
         A[i] := x;
   End;
```

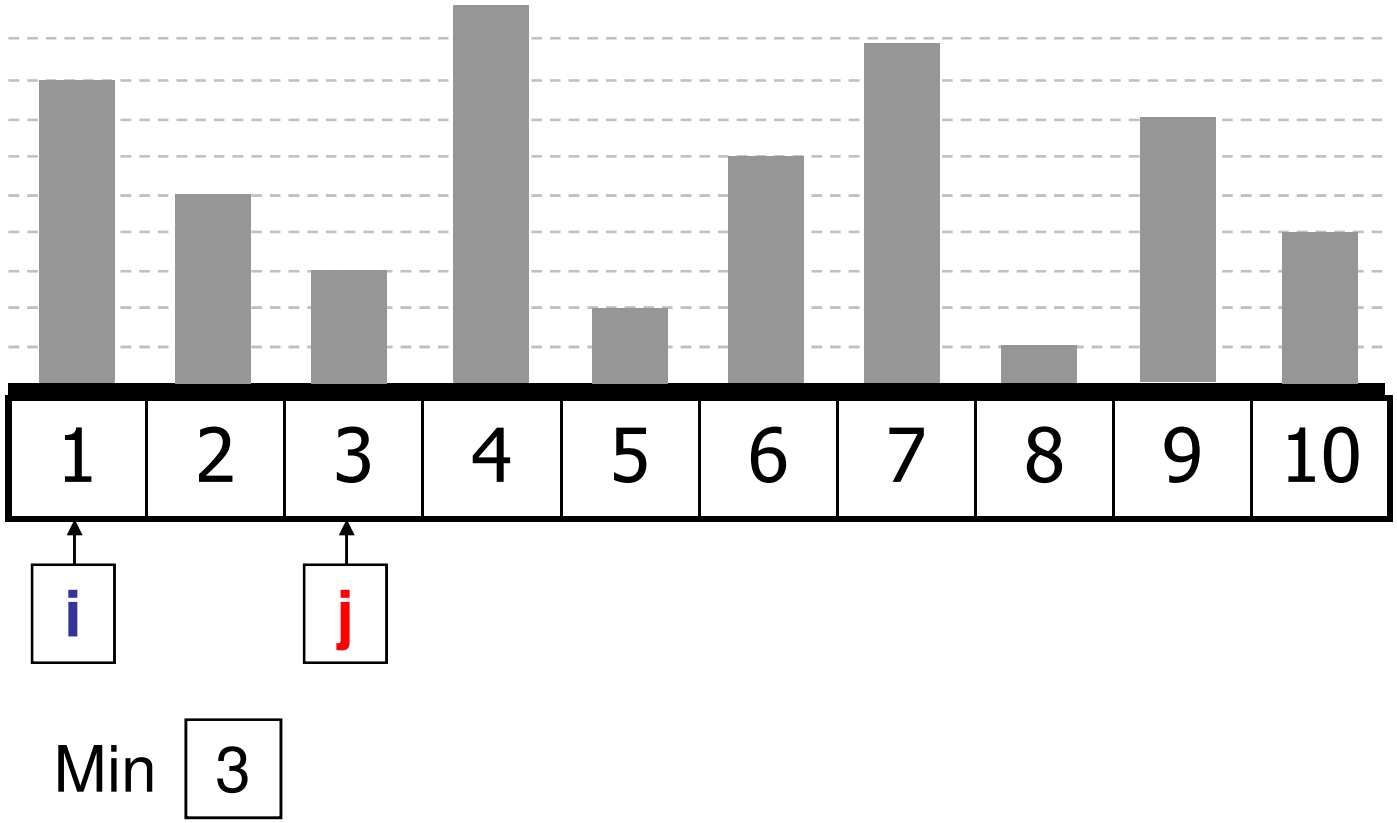
As setas vermelhas indicam operações já realizadas!



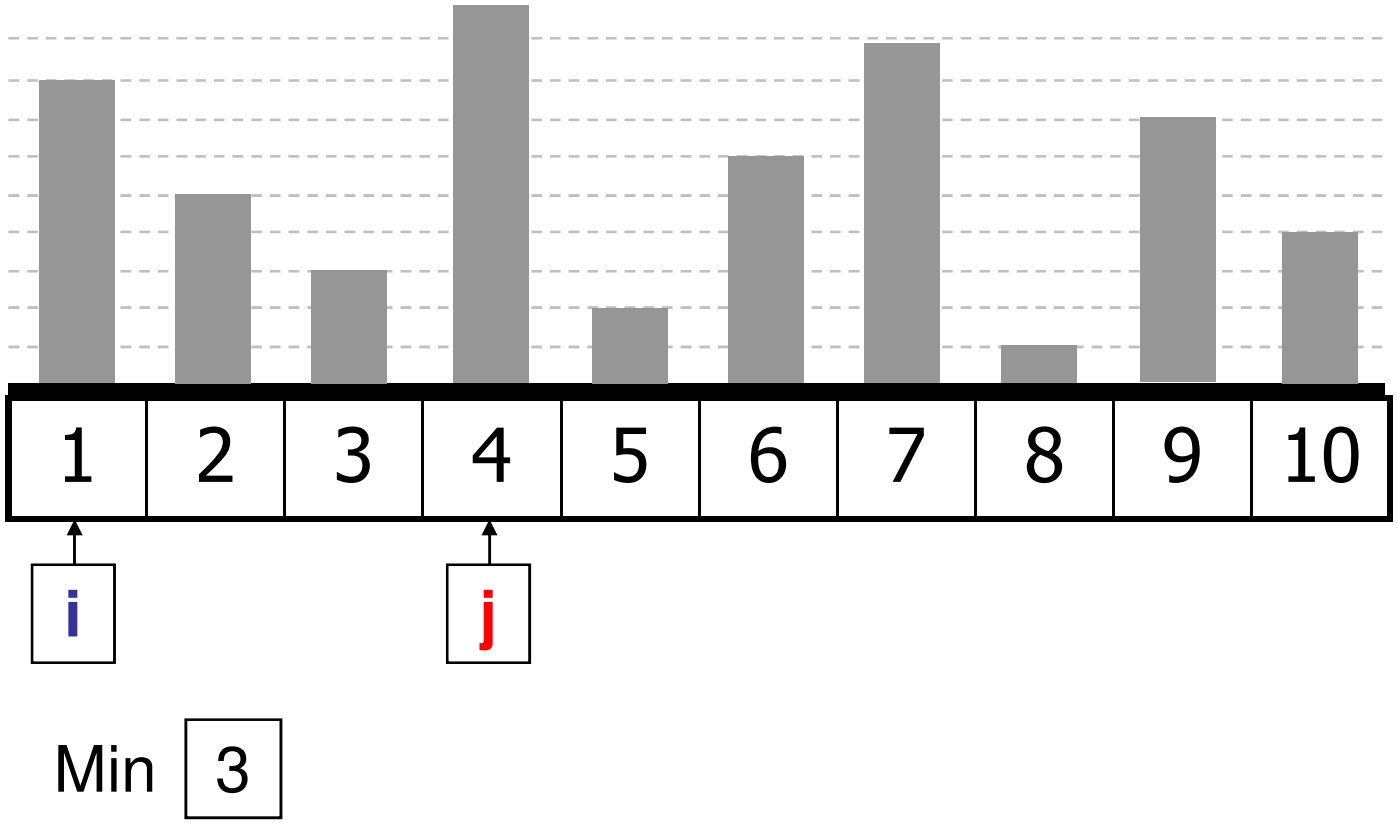
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



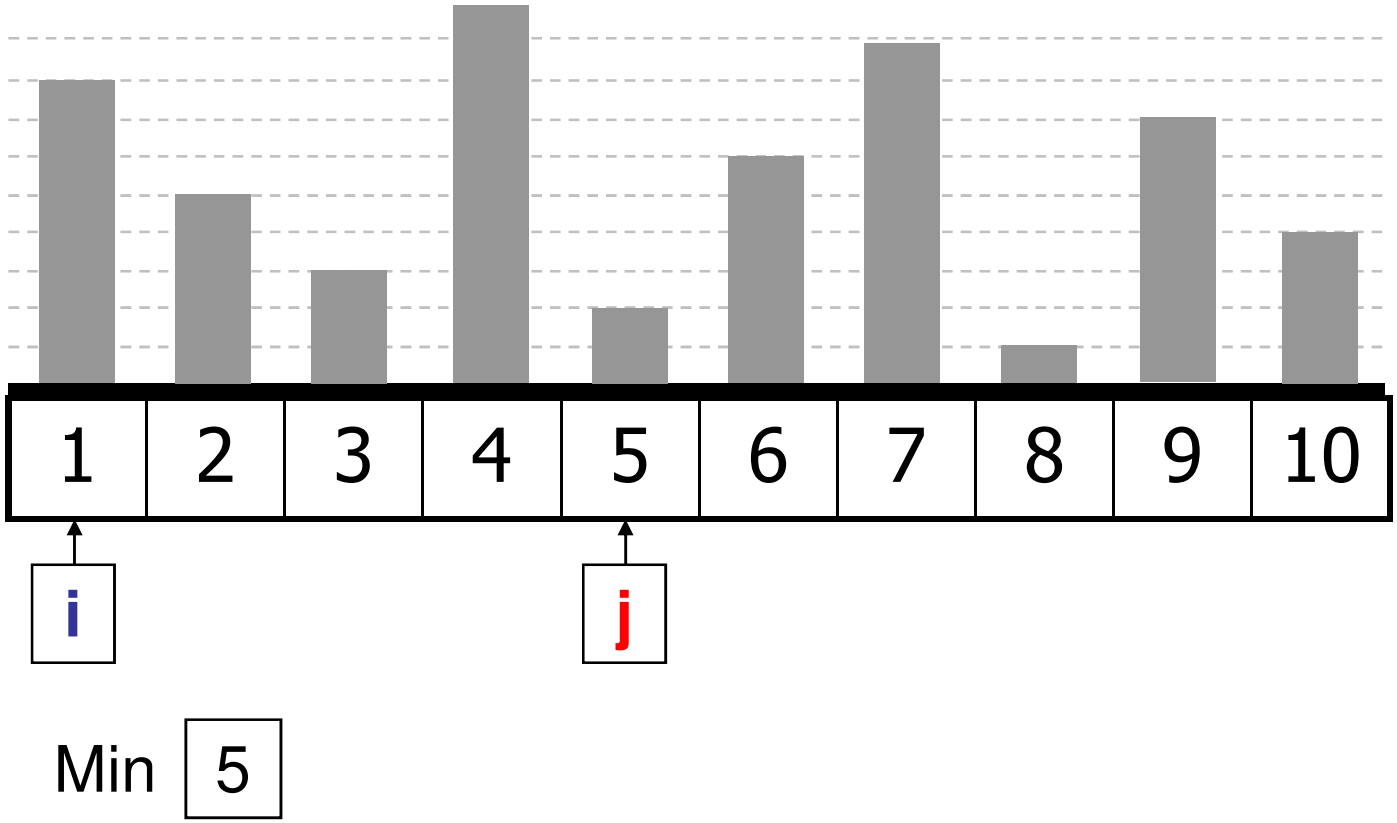
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



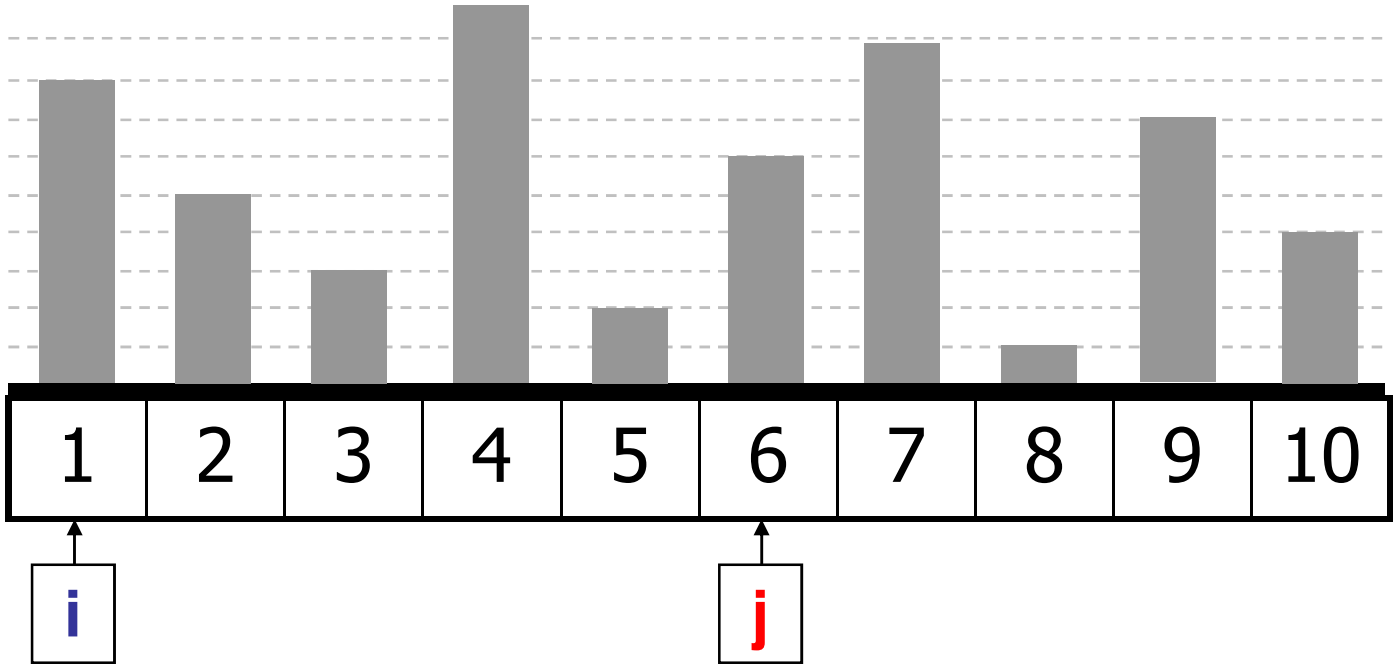
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```

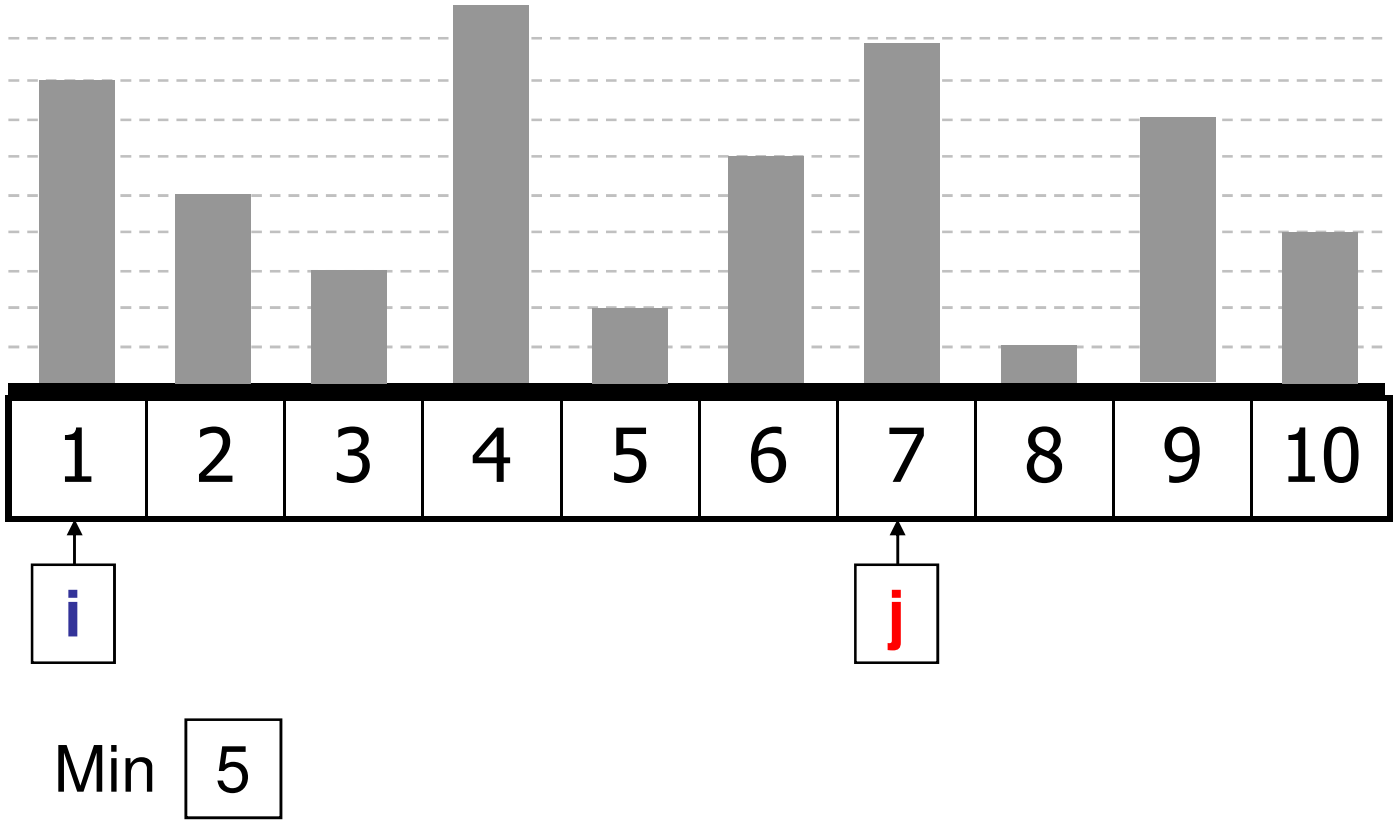


```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```

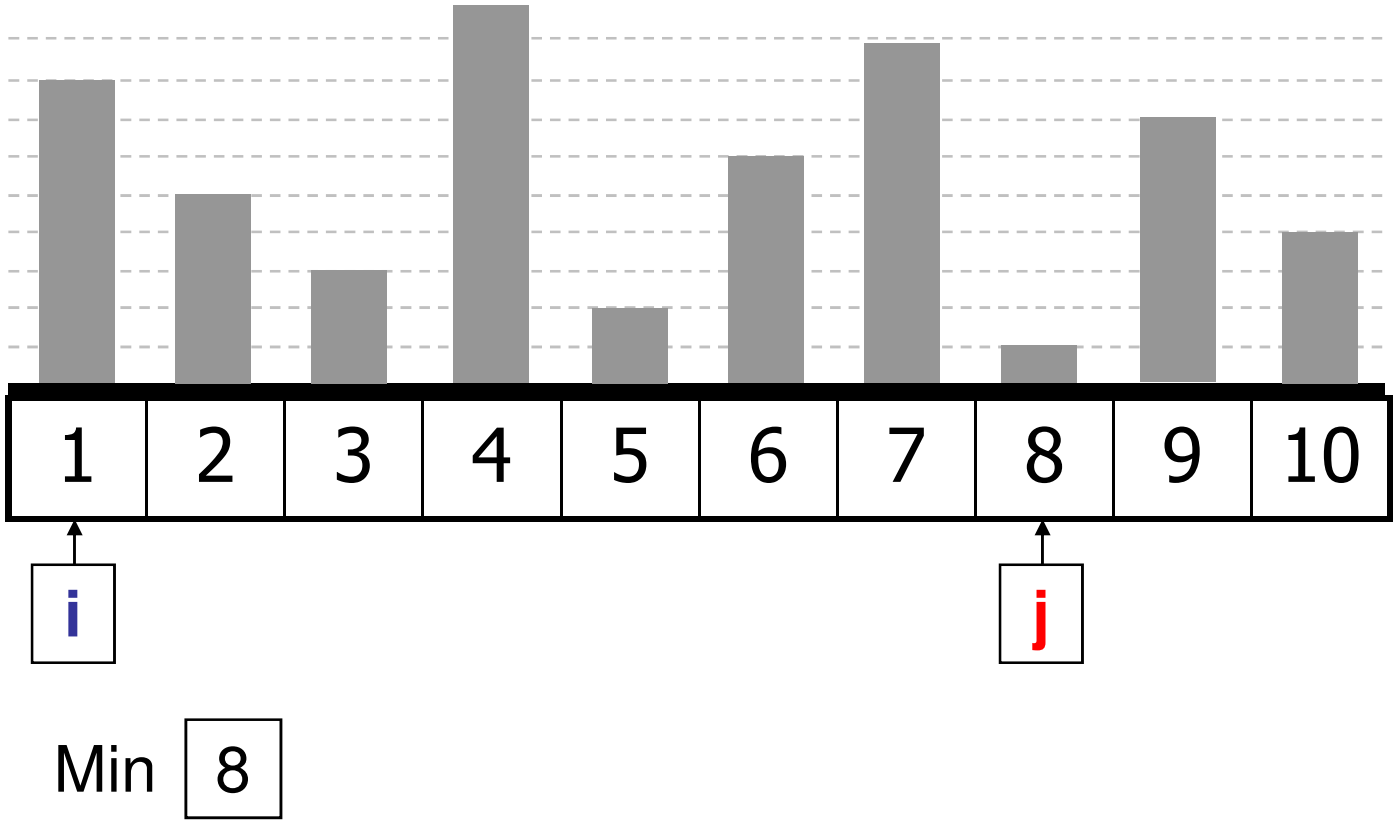


Min 5

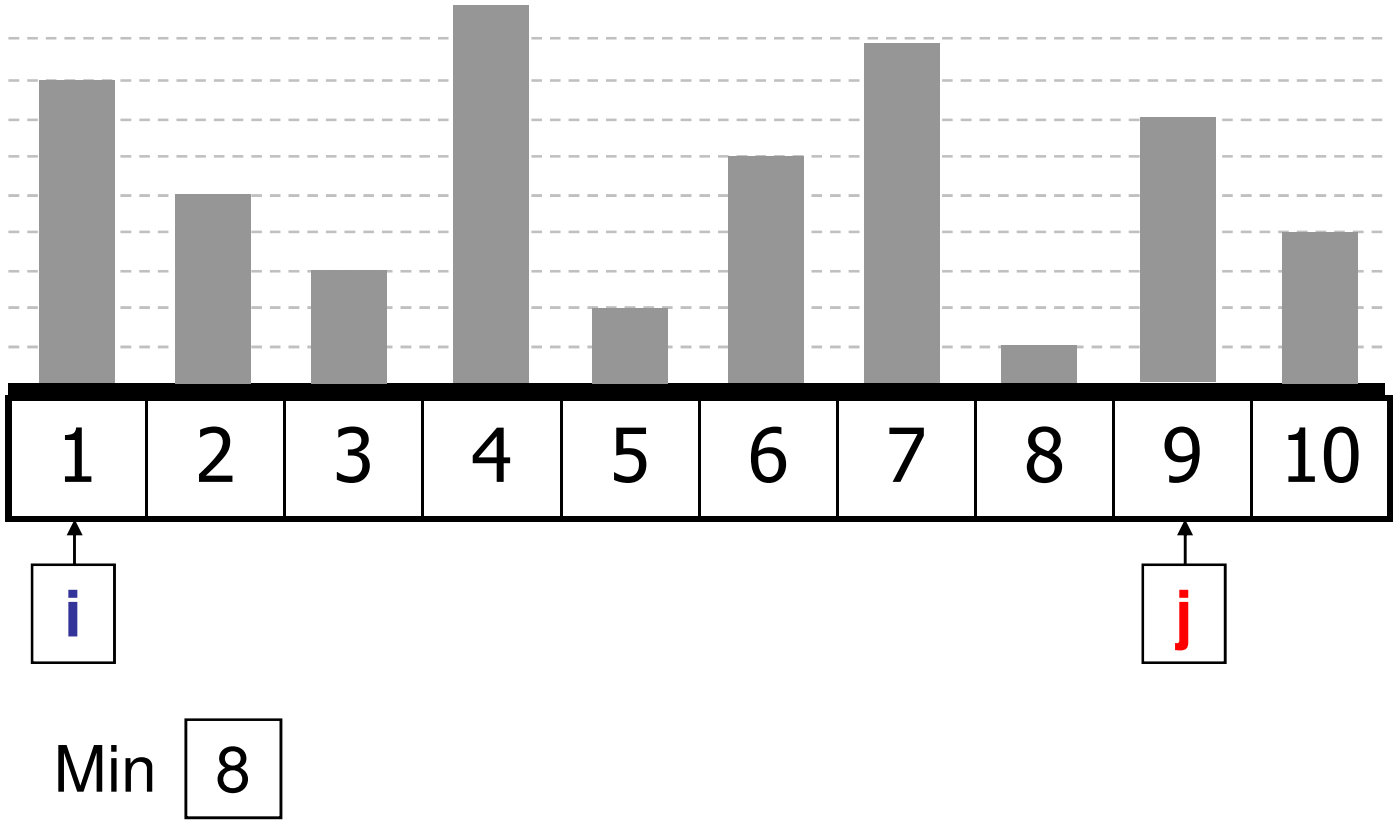
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



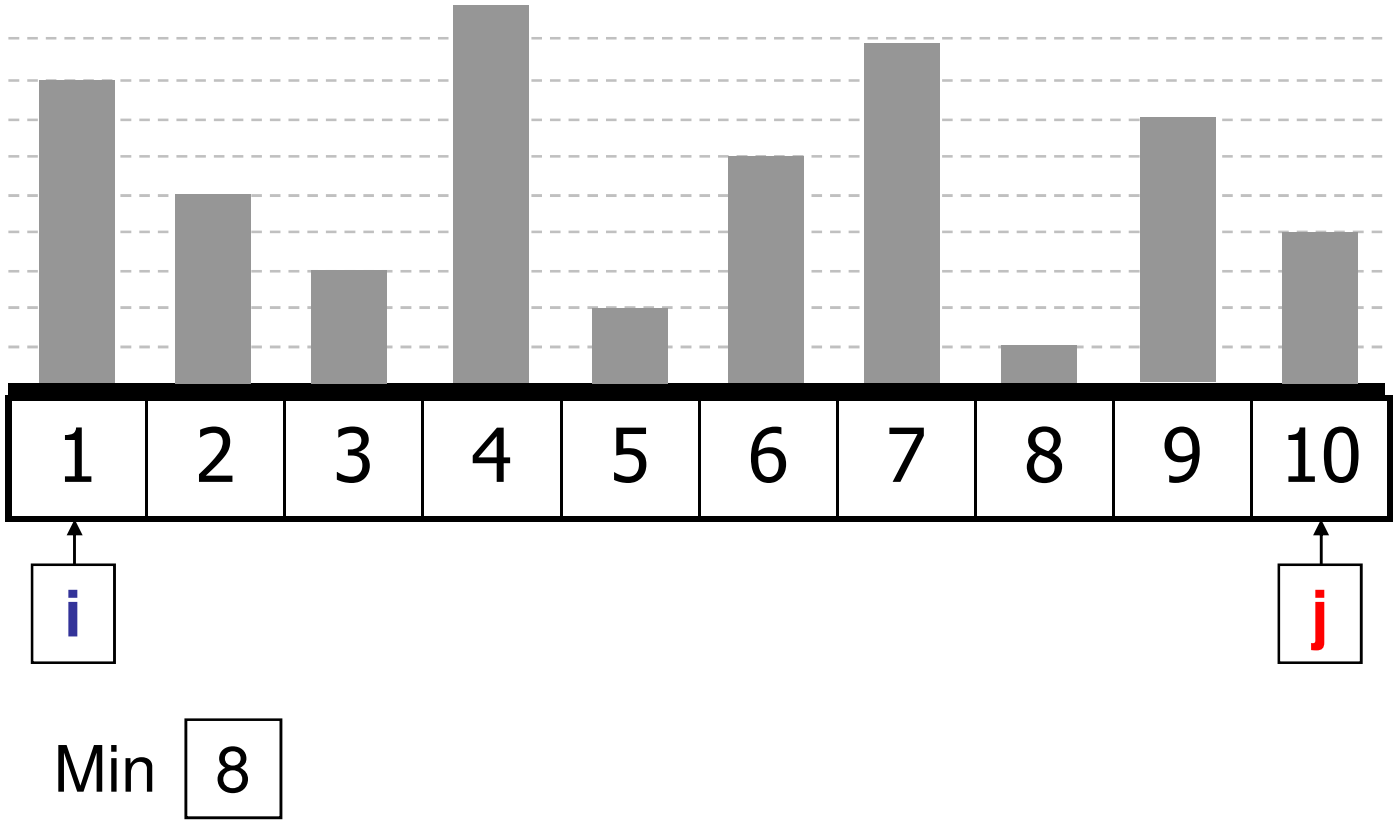
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



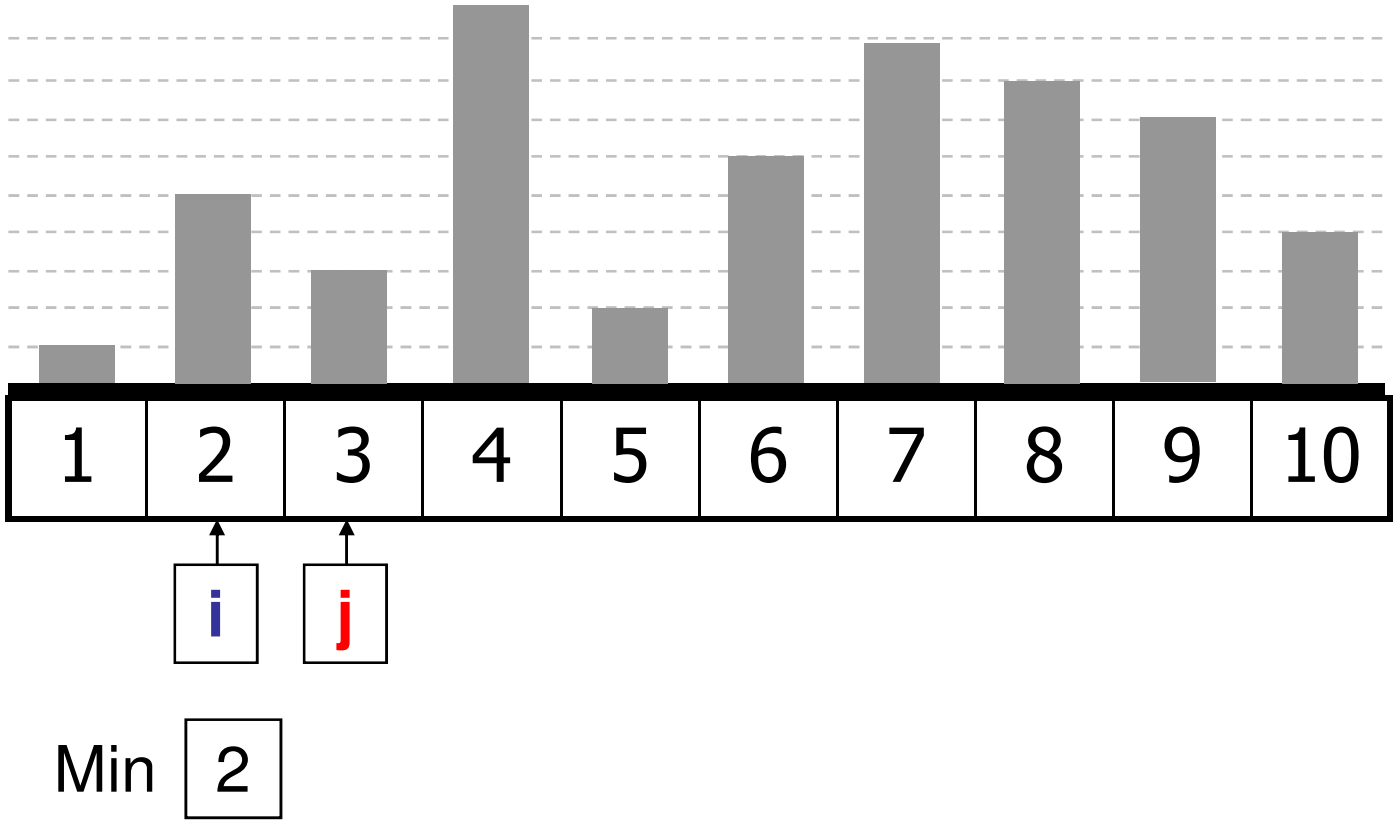
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



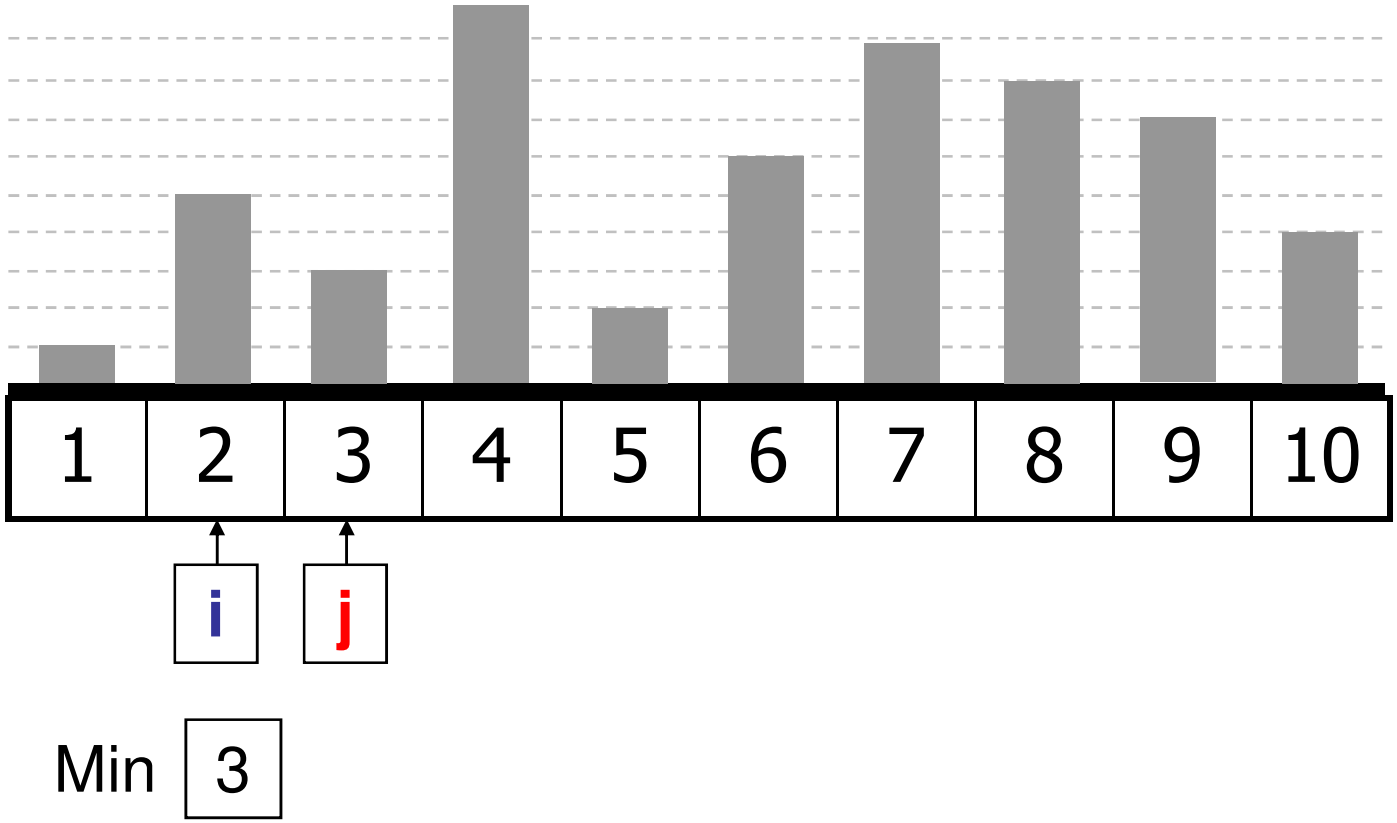
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
         x := A[Min];
         A[Min] := A[i];
         A[i] := x;
   End;
```



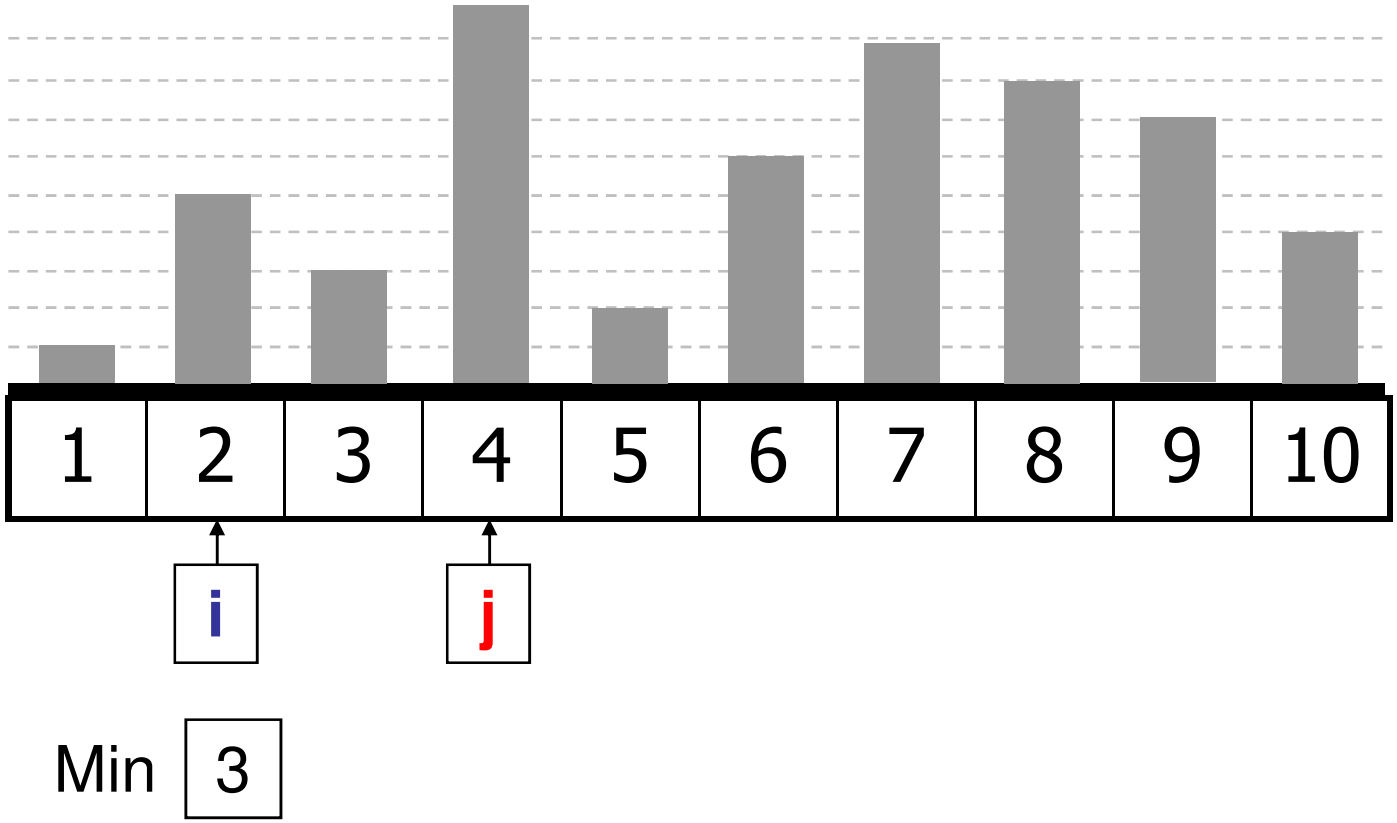
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



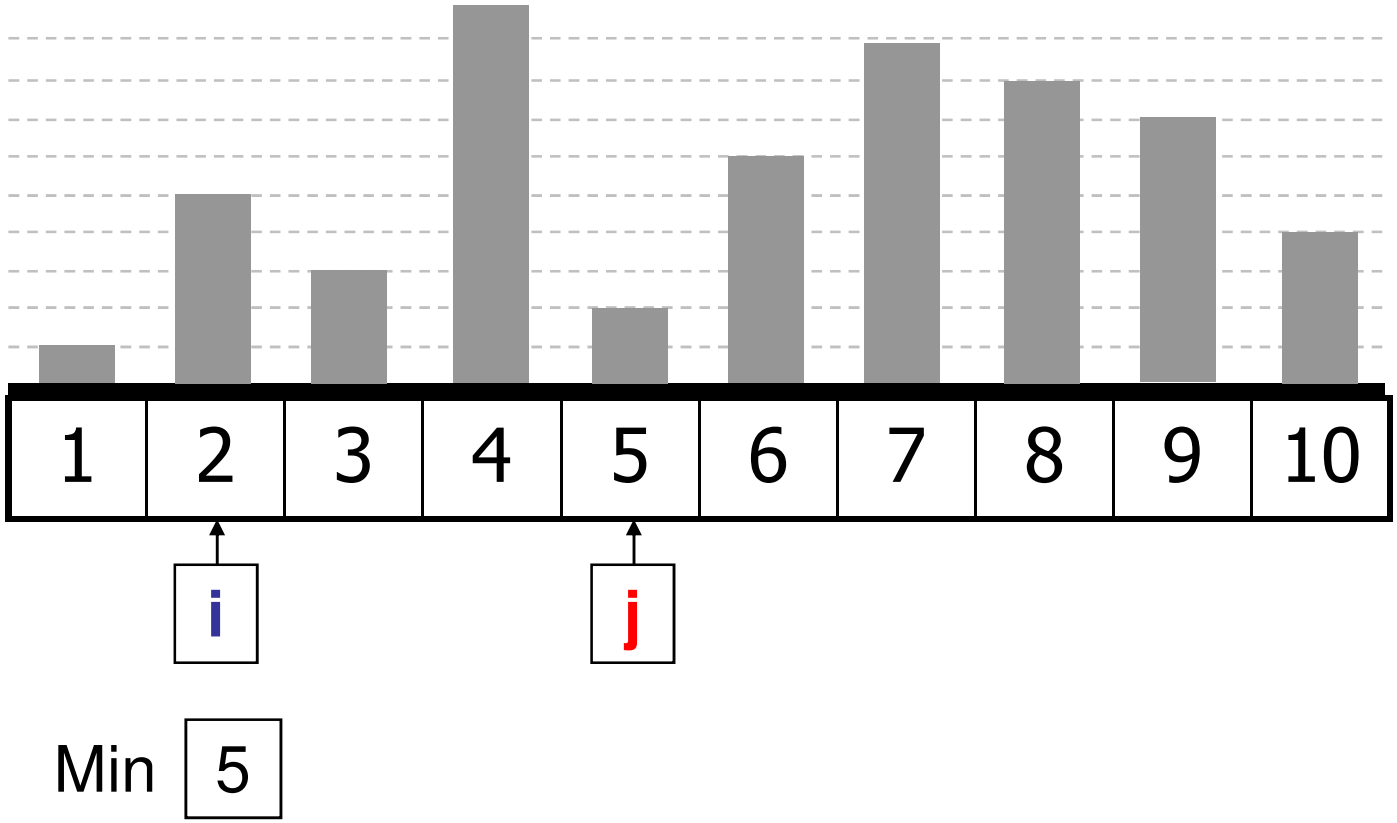
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



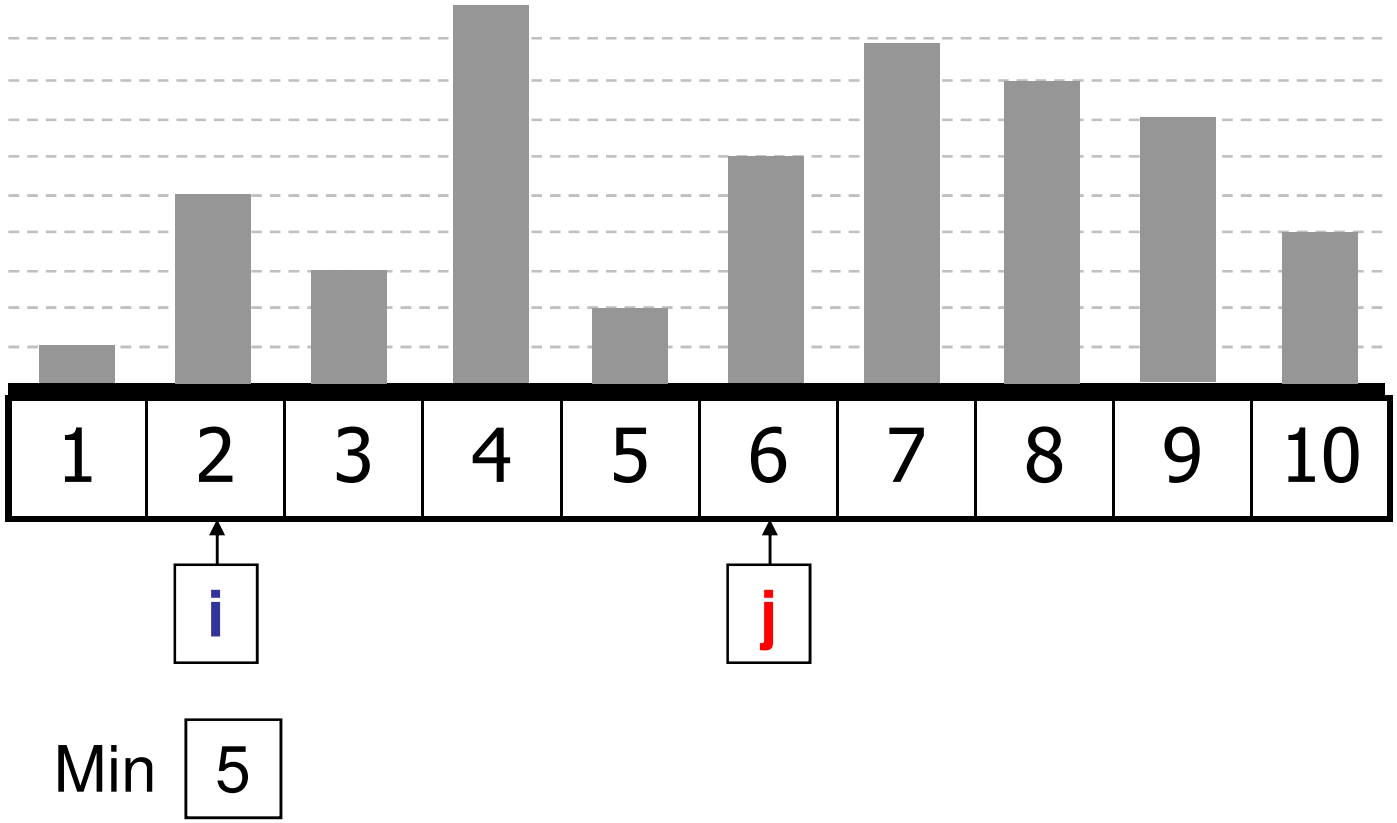
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



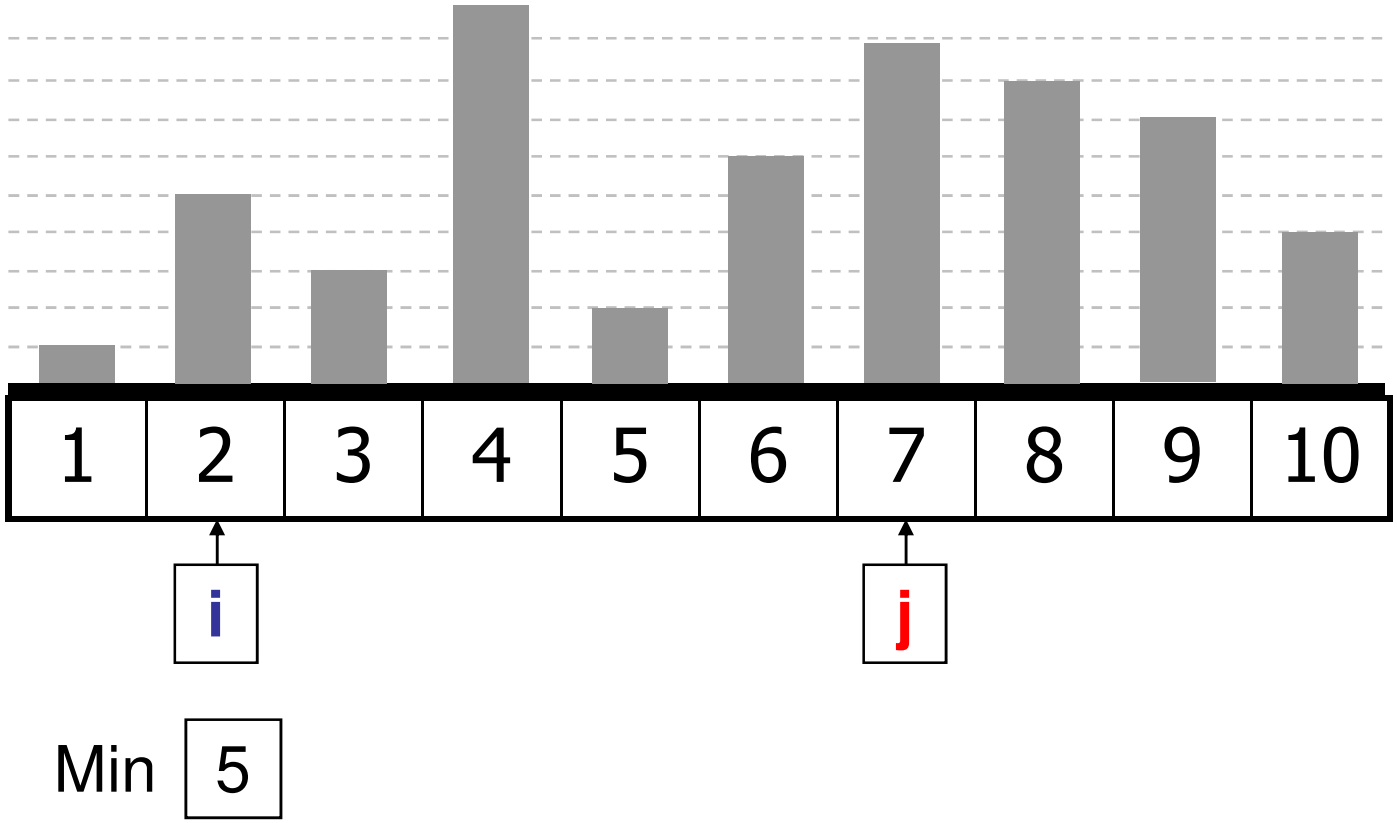
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j; ←
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



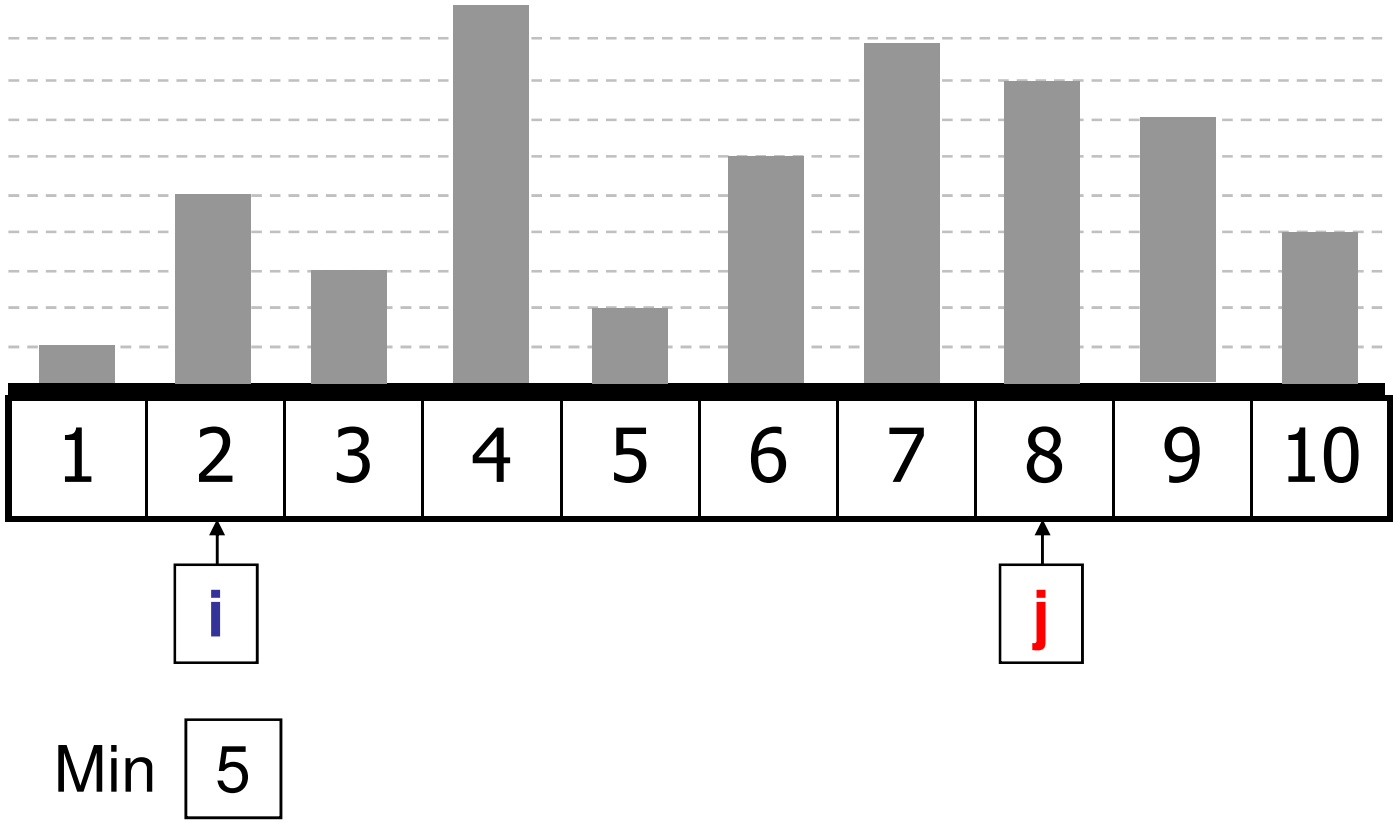
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



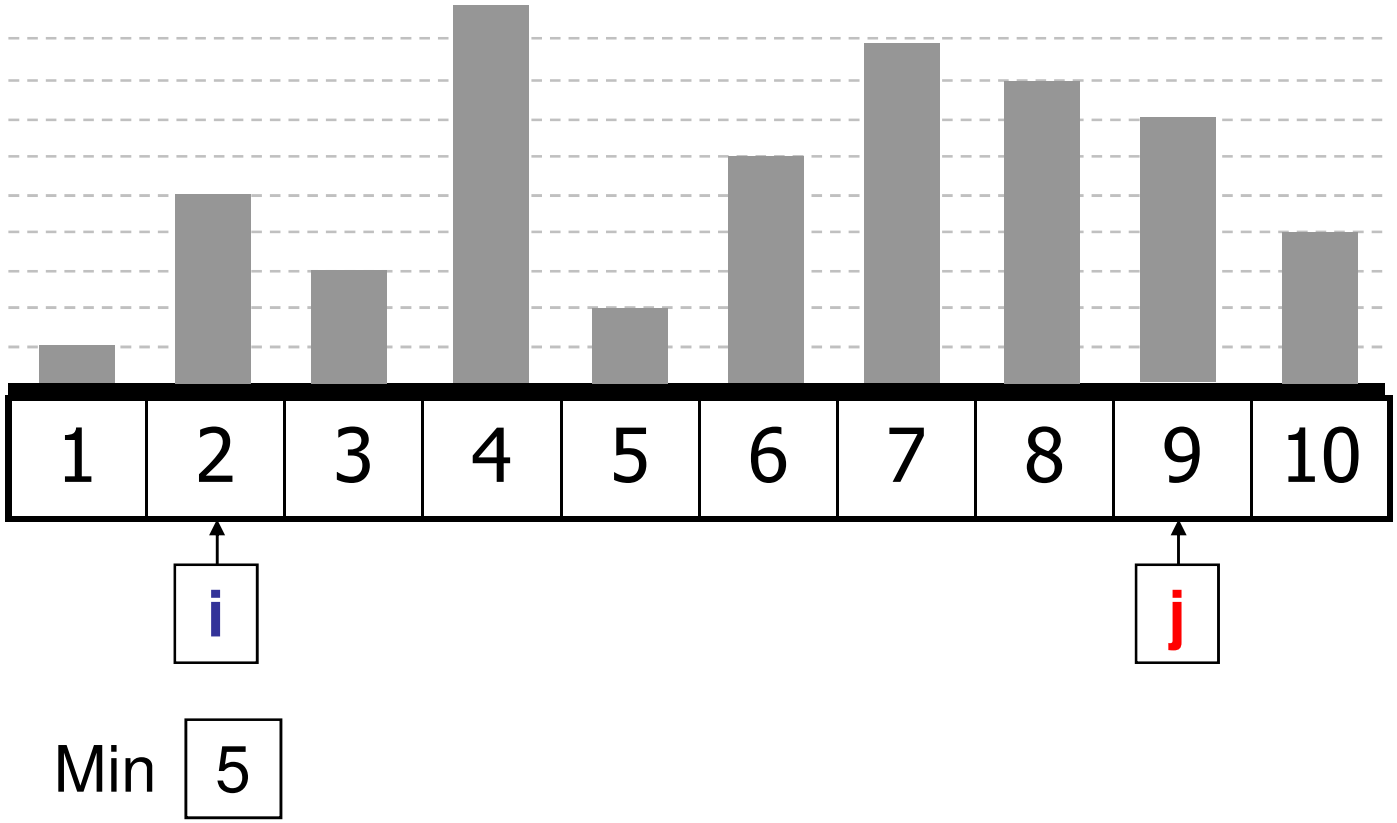
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



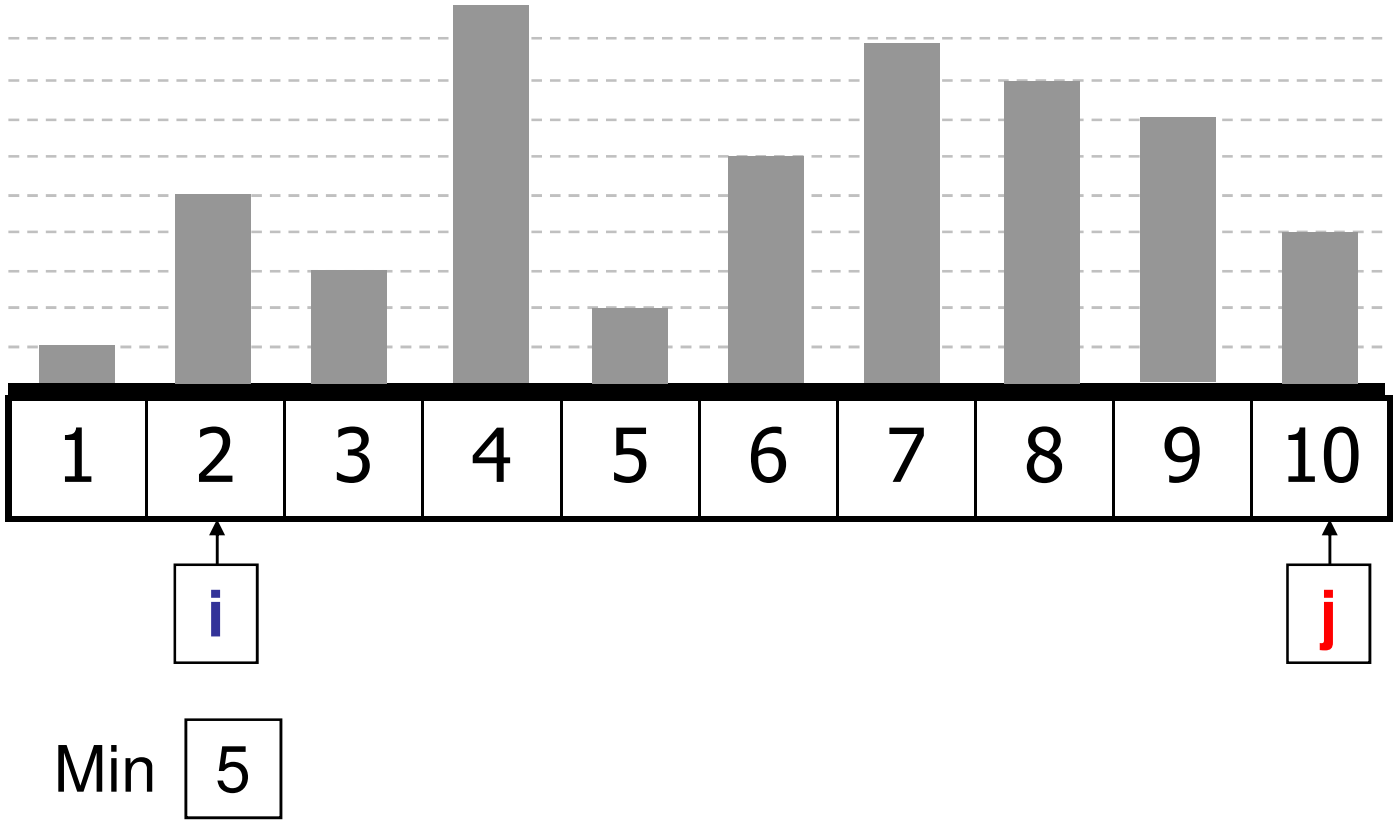
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



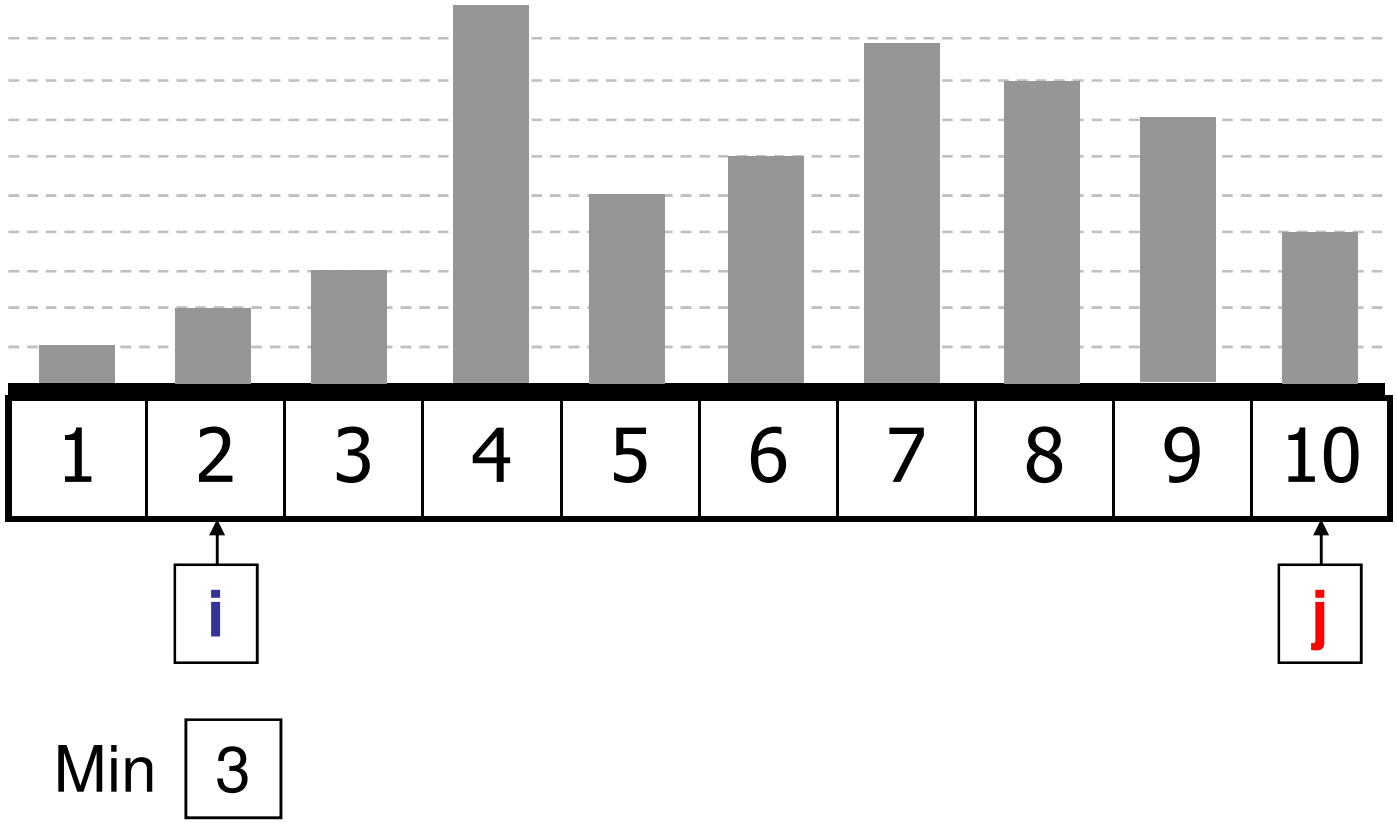
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



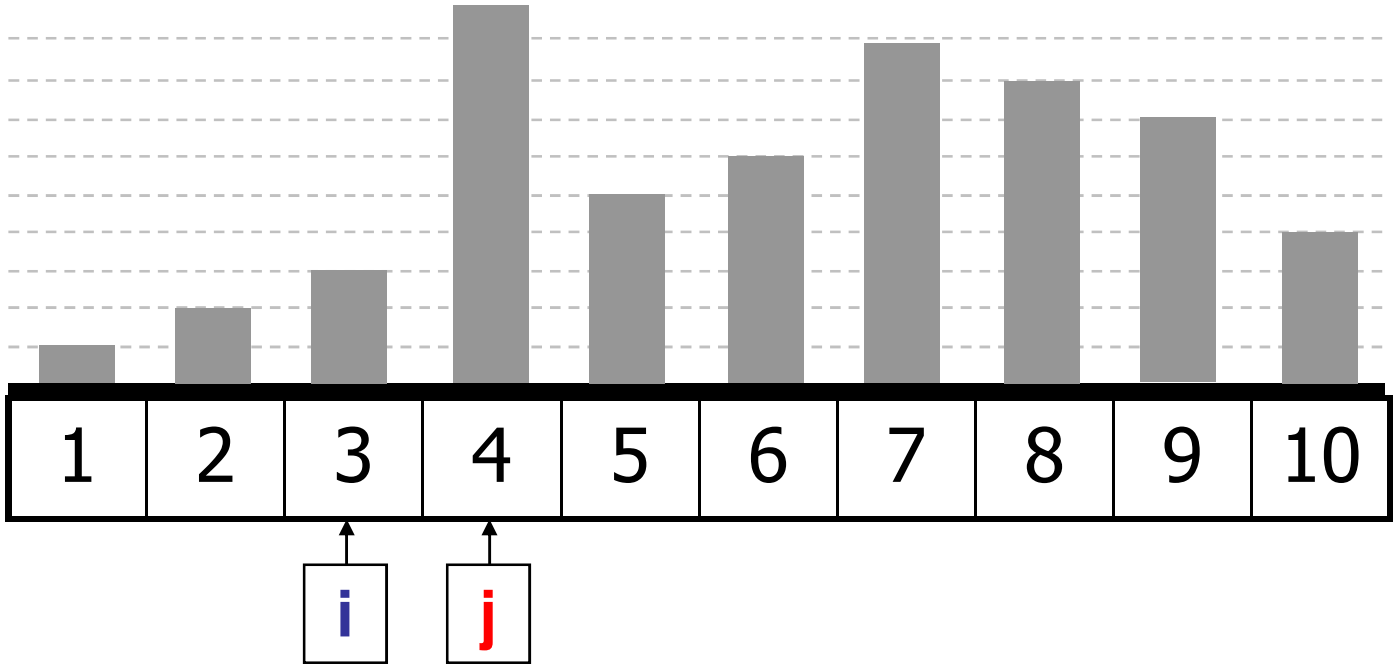
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
         x := A[Min];
         A[Min] := A[i];
         A[i] := x;
   End;
```

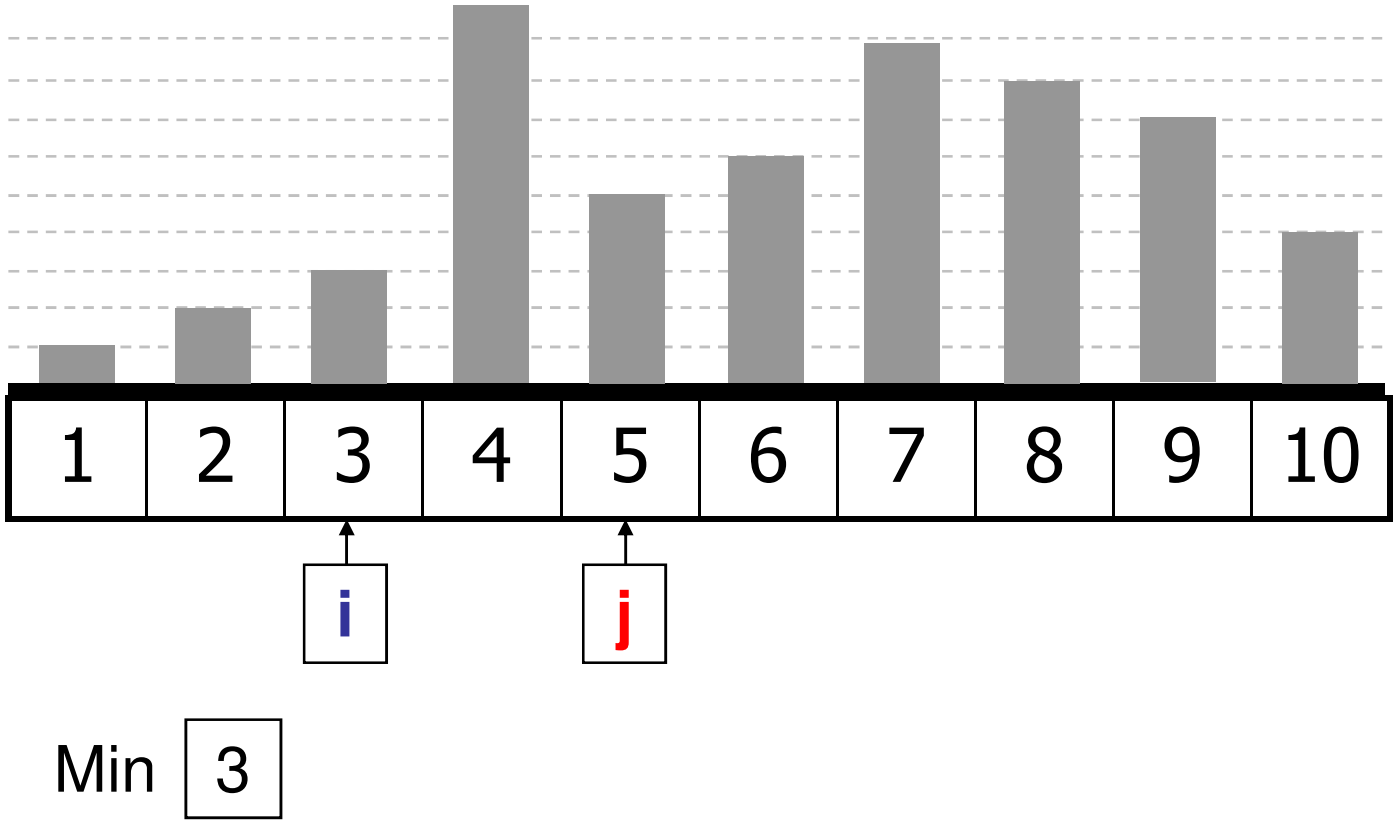


```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```

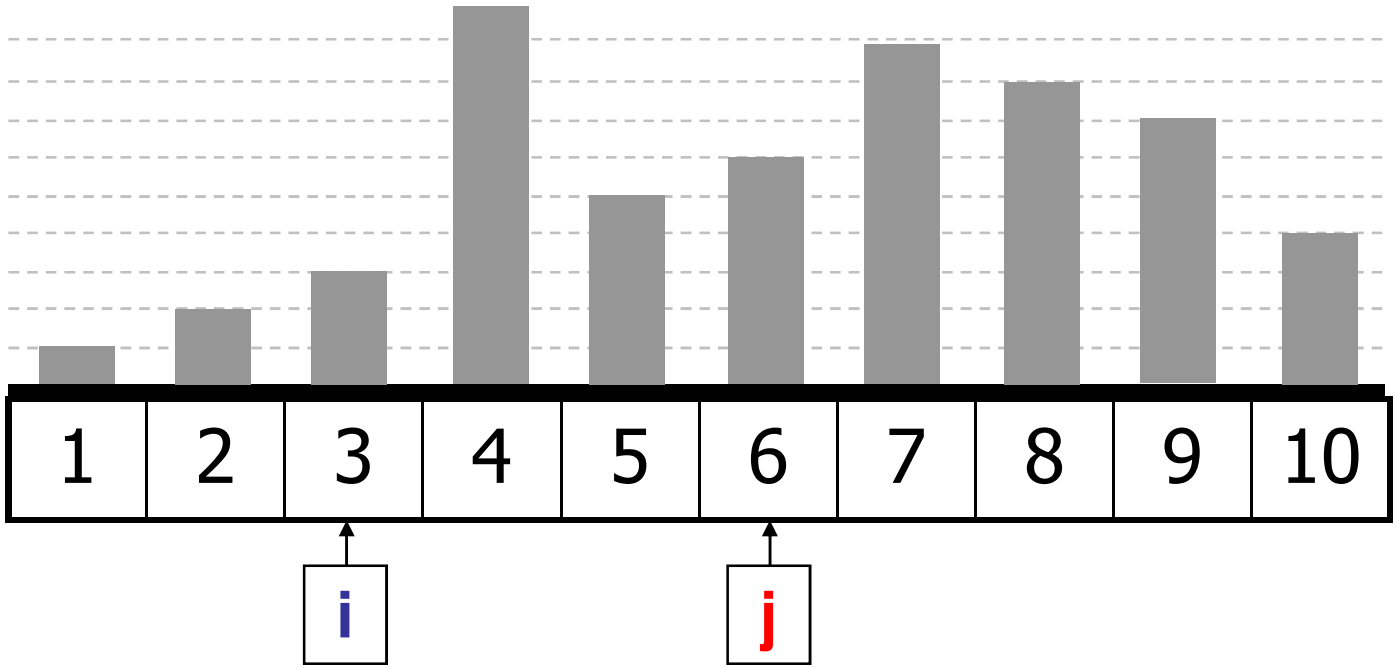


Min 3

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```

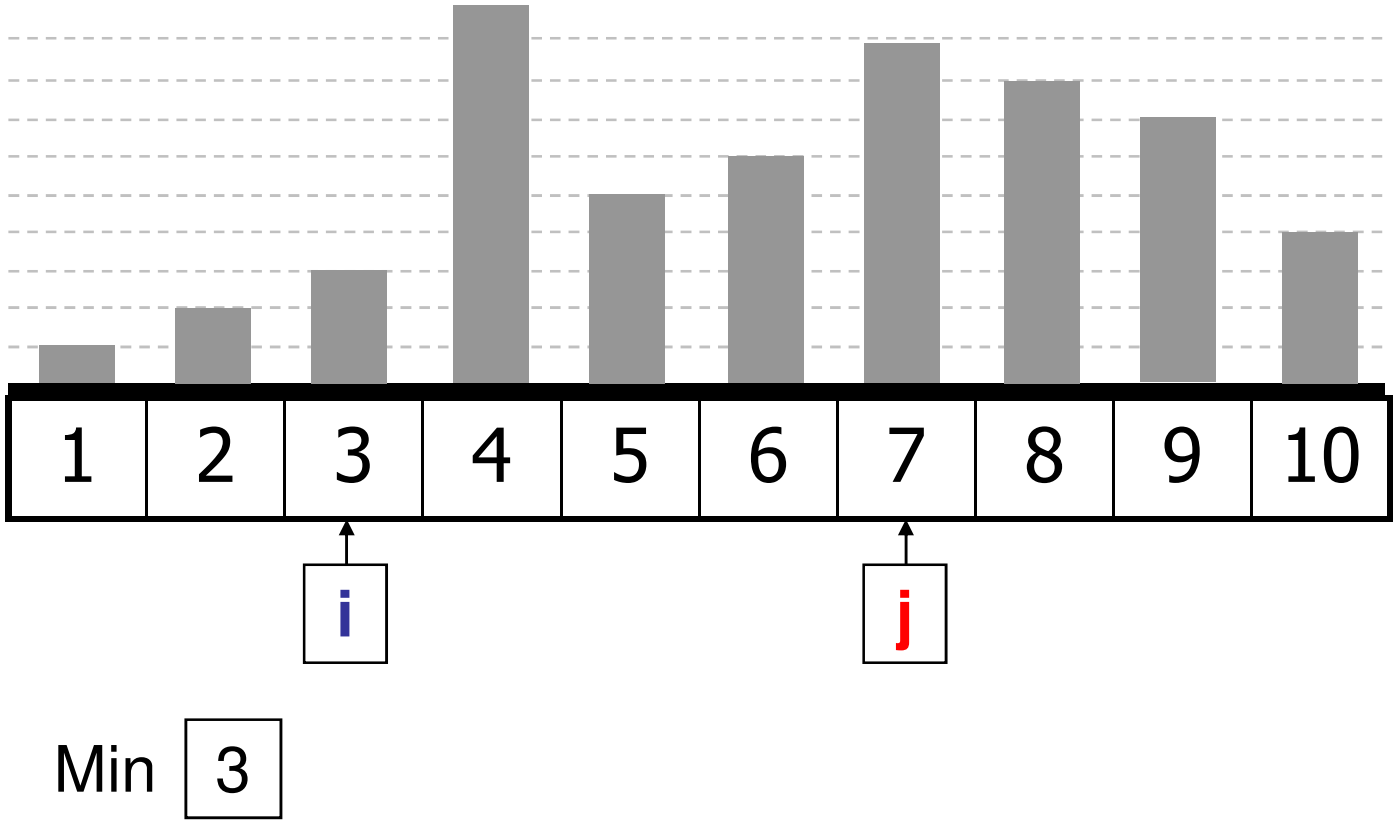


```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```

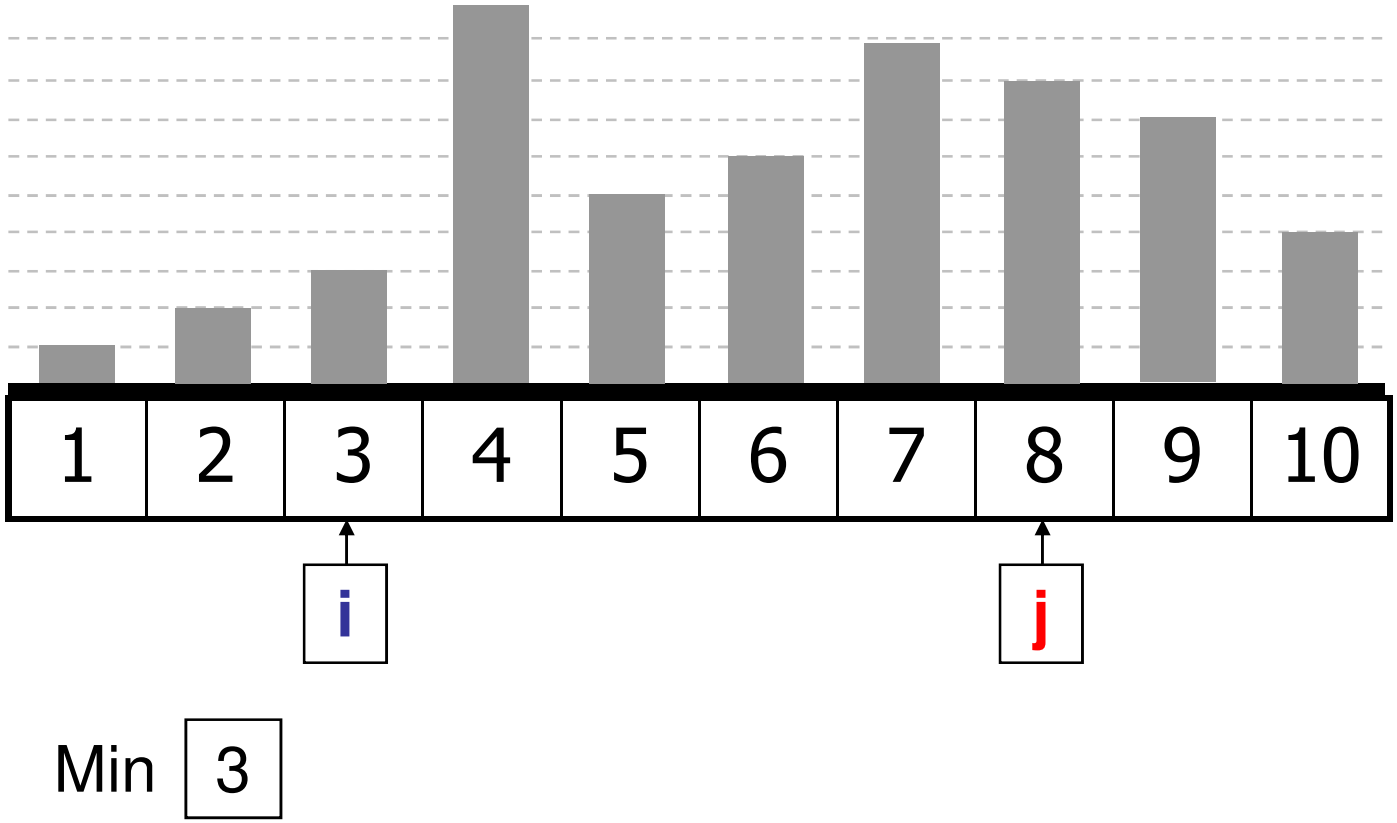


Min 3

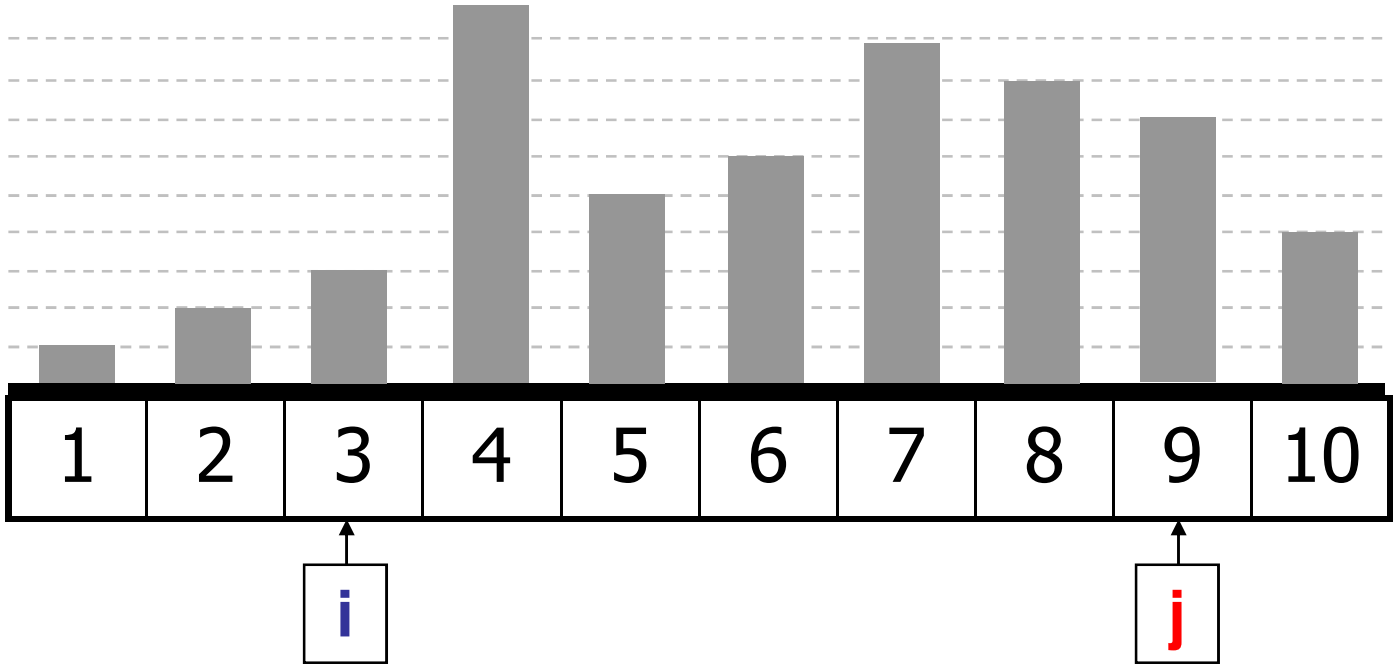
```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```

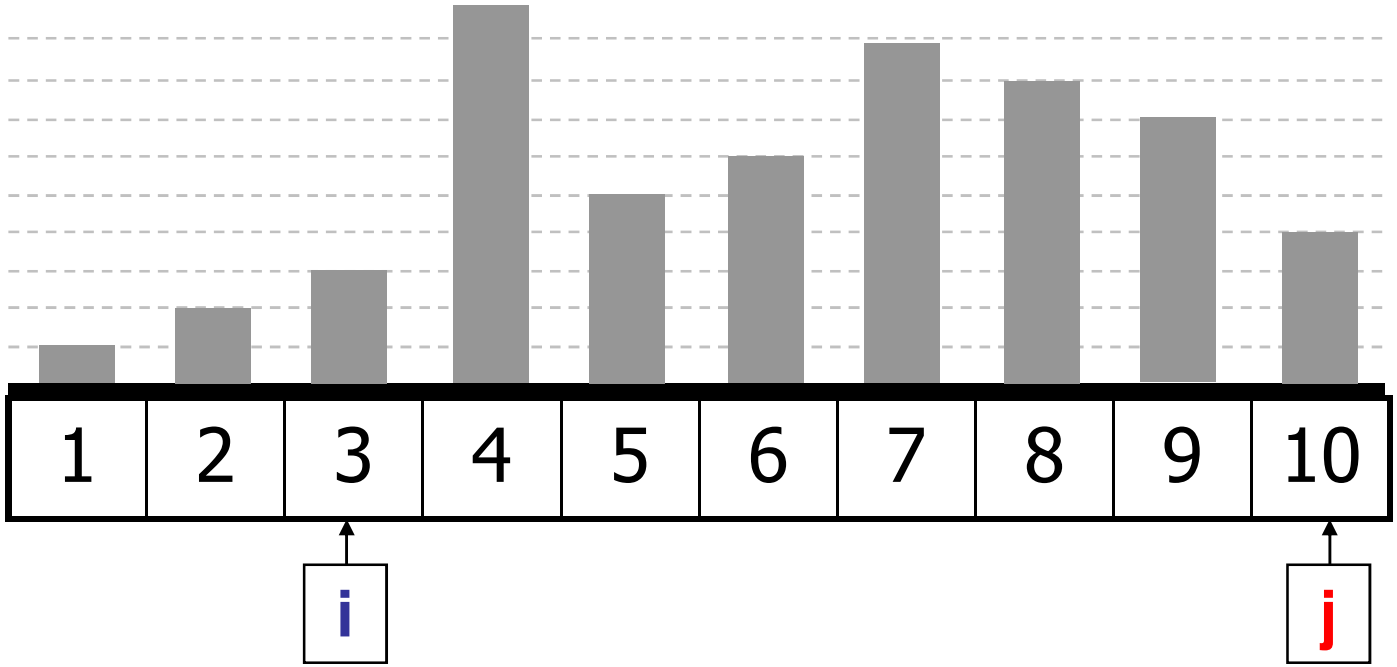


```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



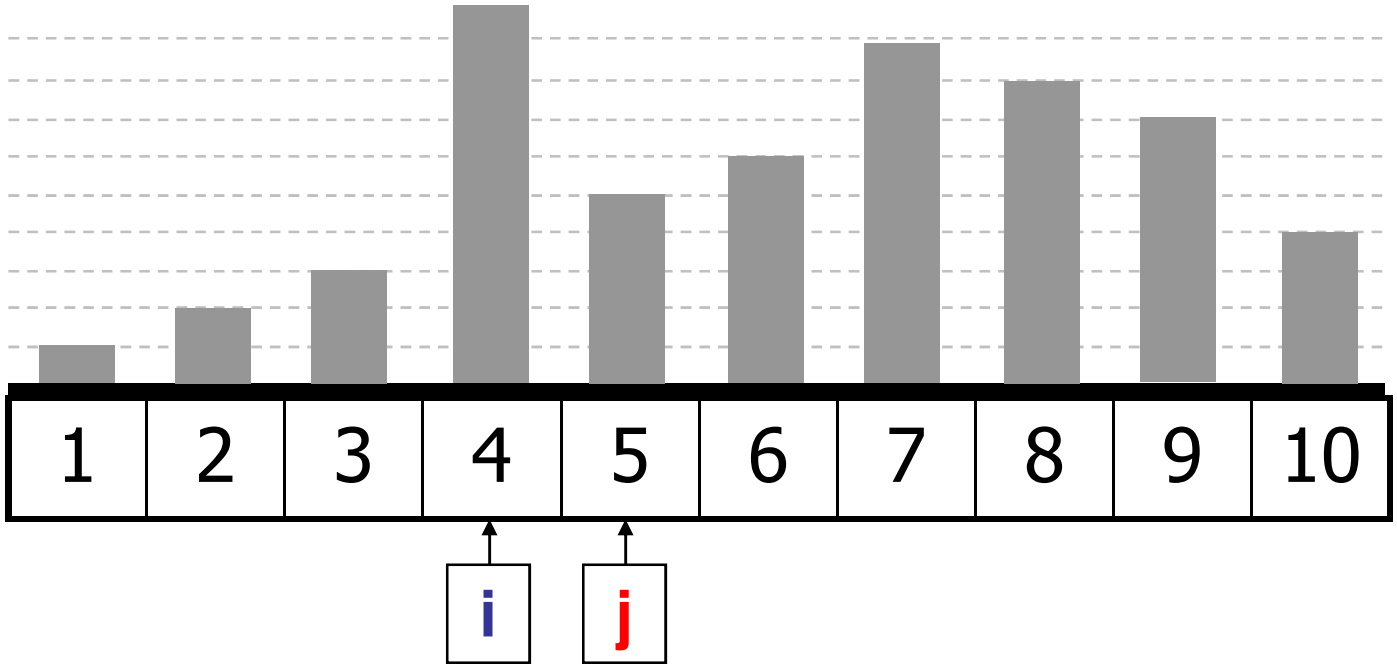
Min 3

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



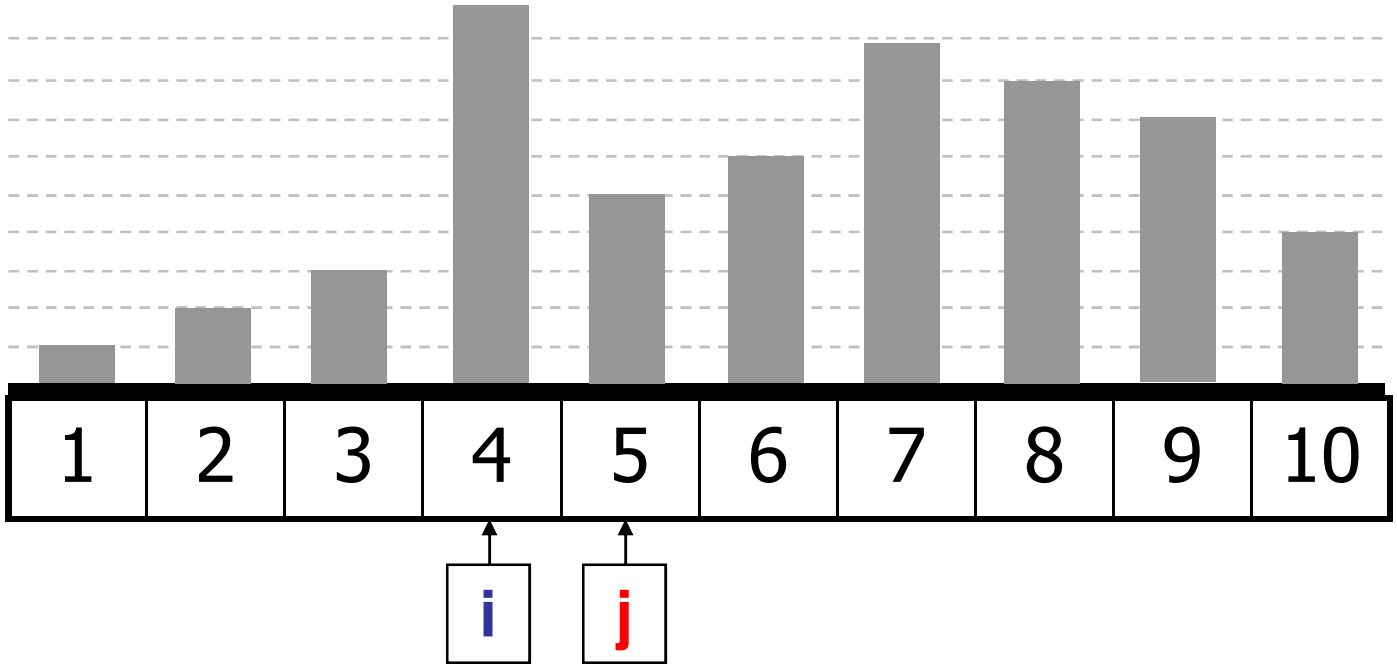
Min 3

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



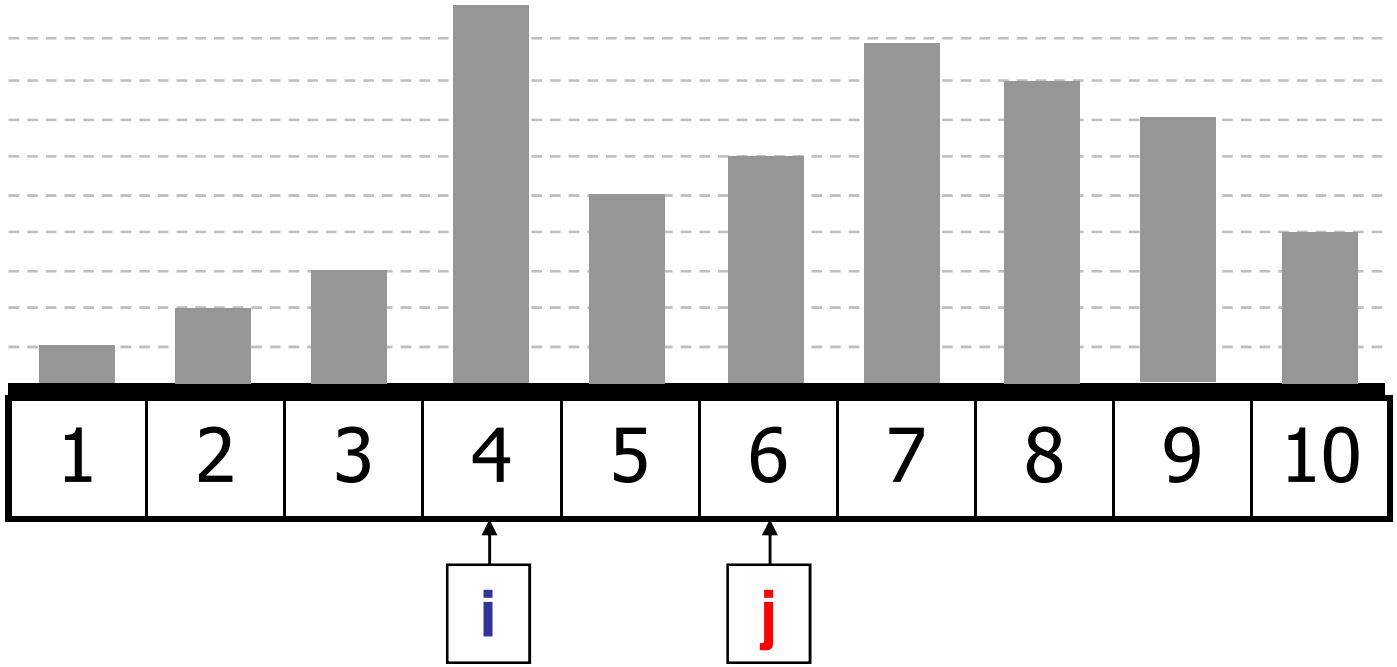
Min 4

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



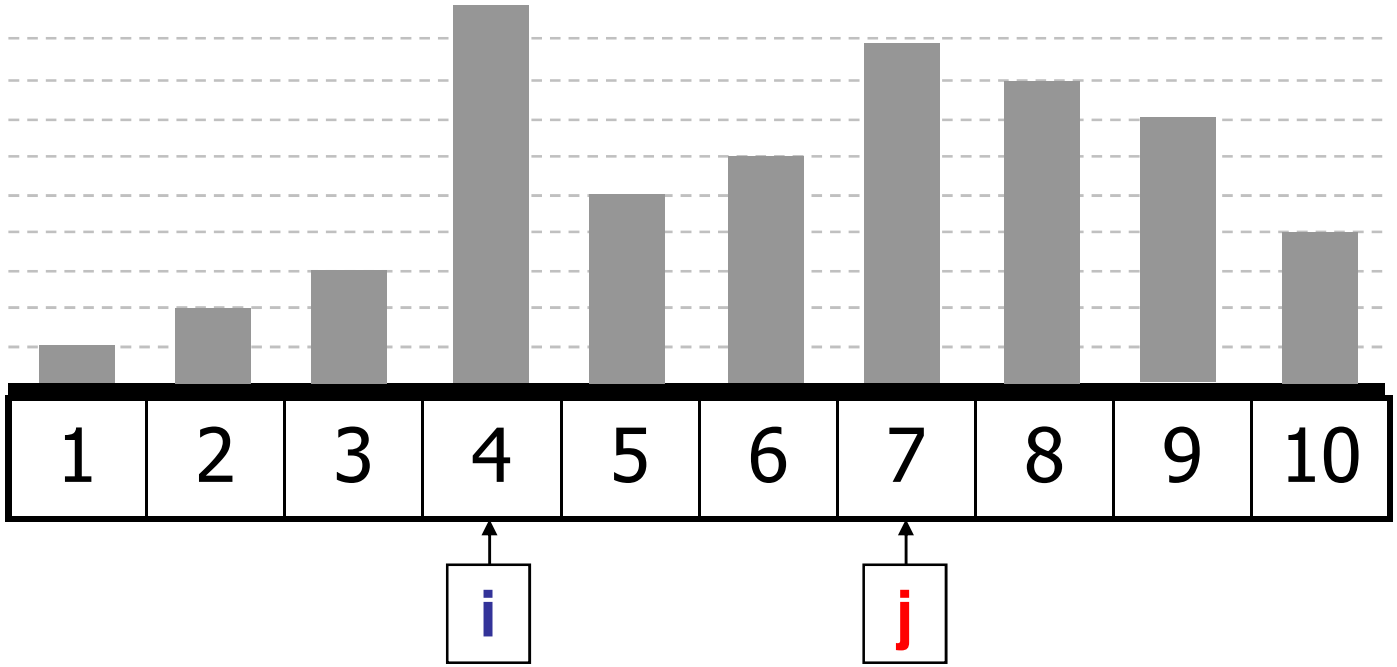
Min 5

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



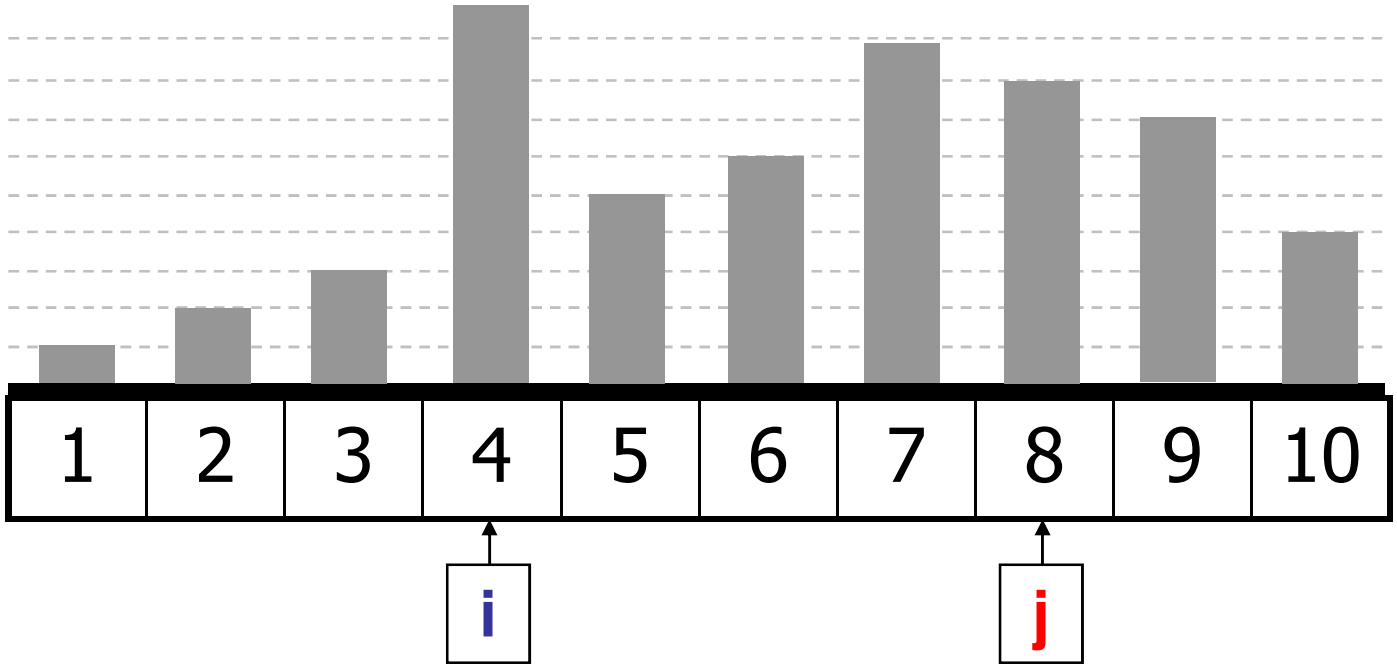
Min 5

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



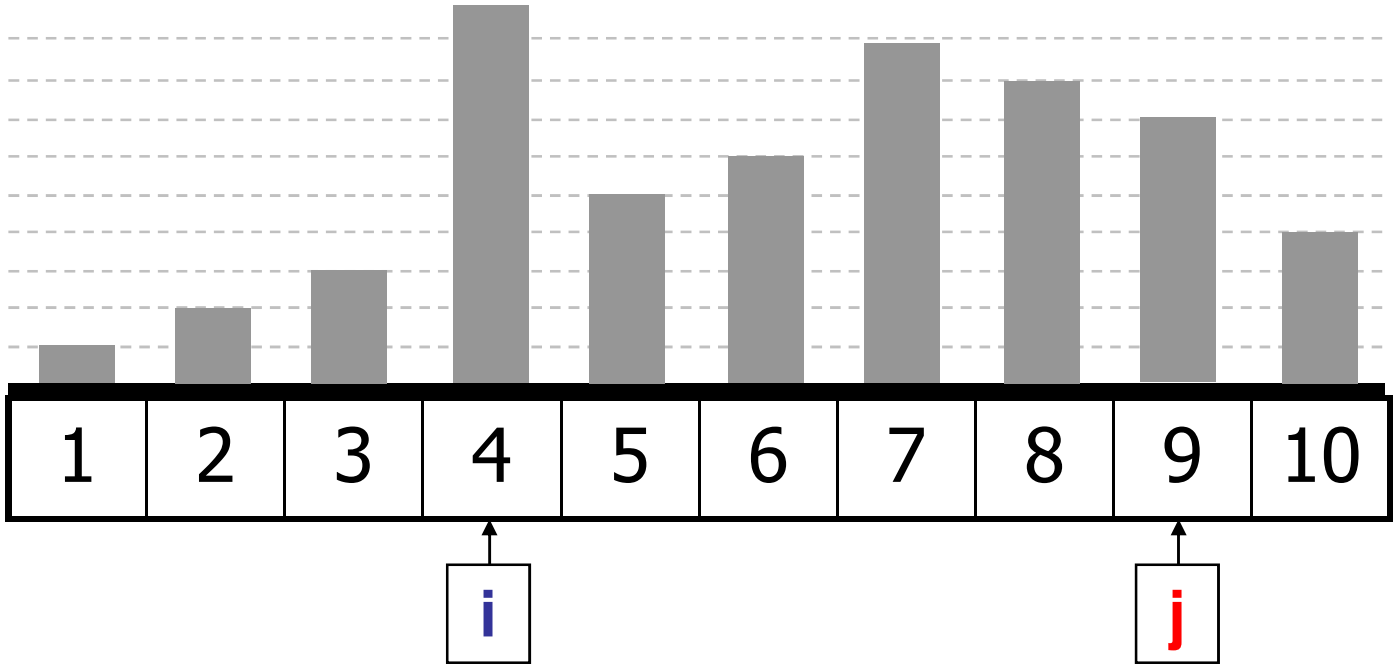
Min 5

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



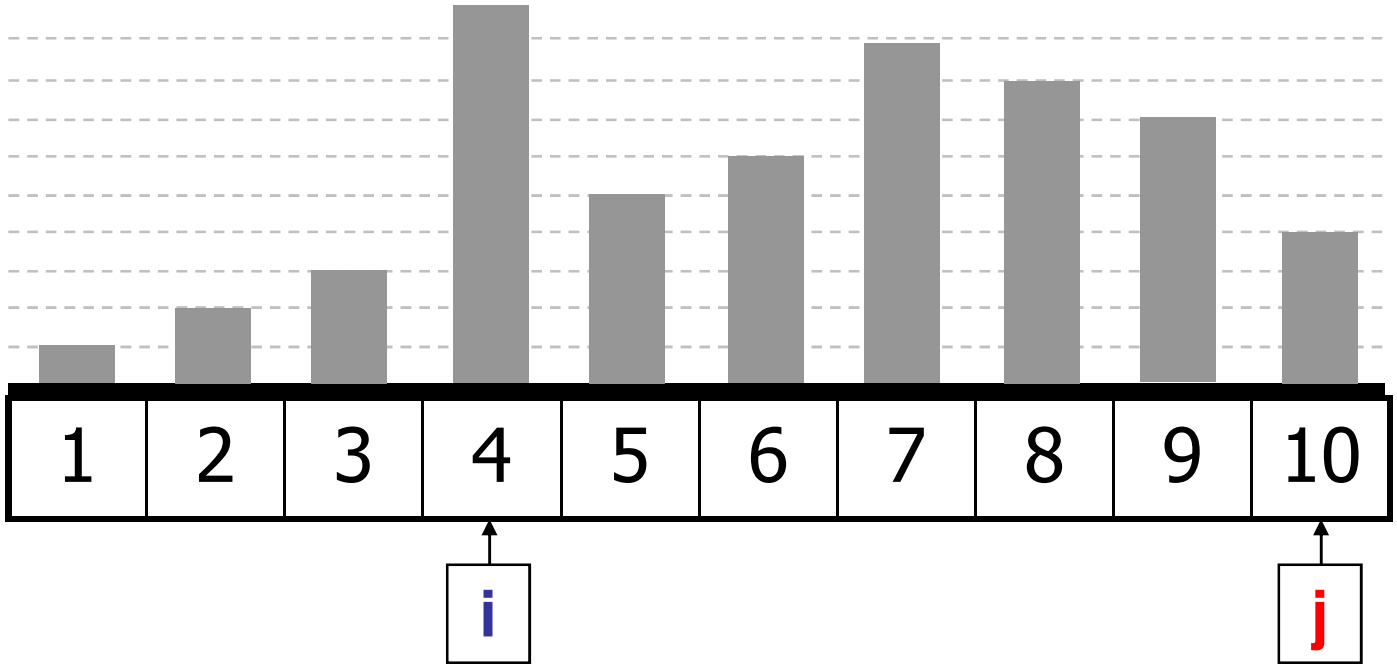
Min 5

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



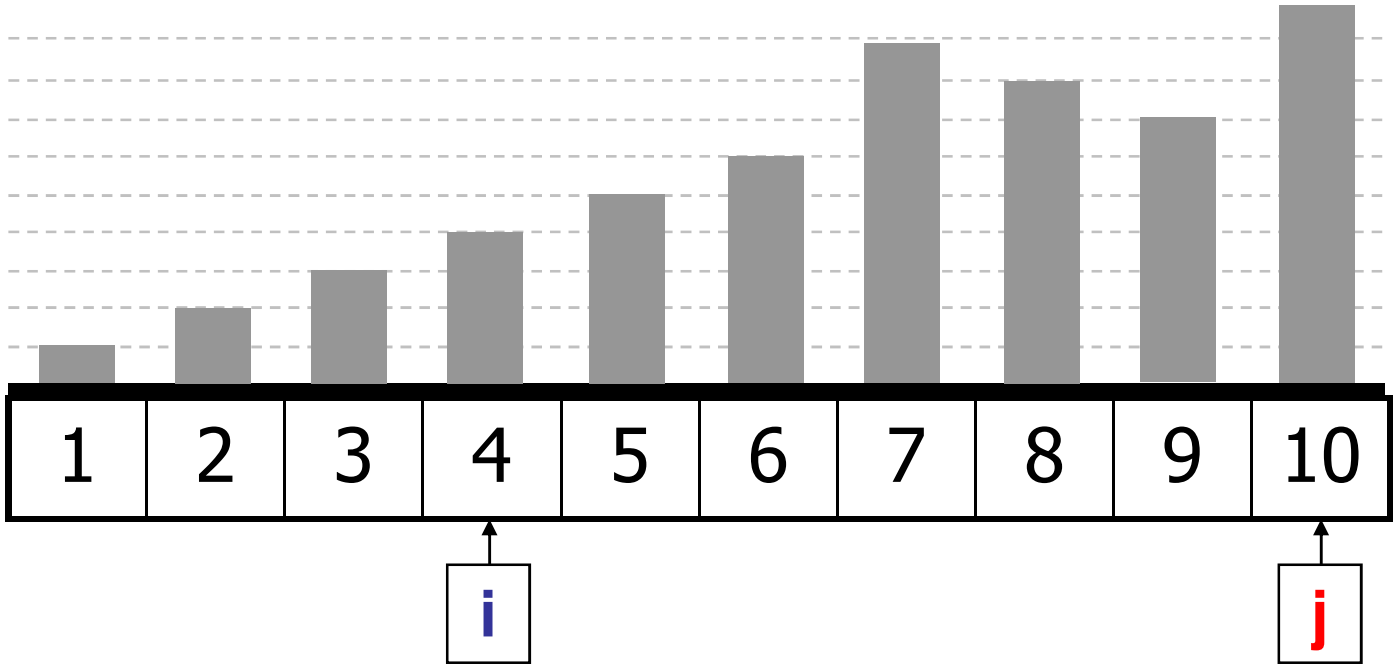
Min 5

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j; ←
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



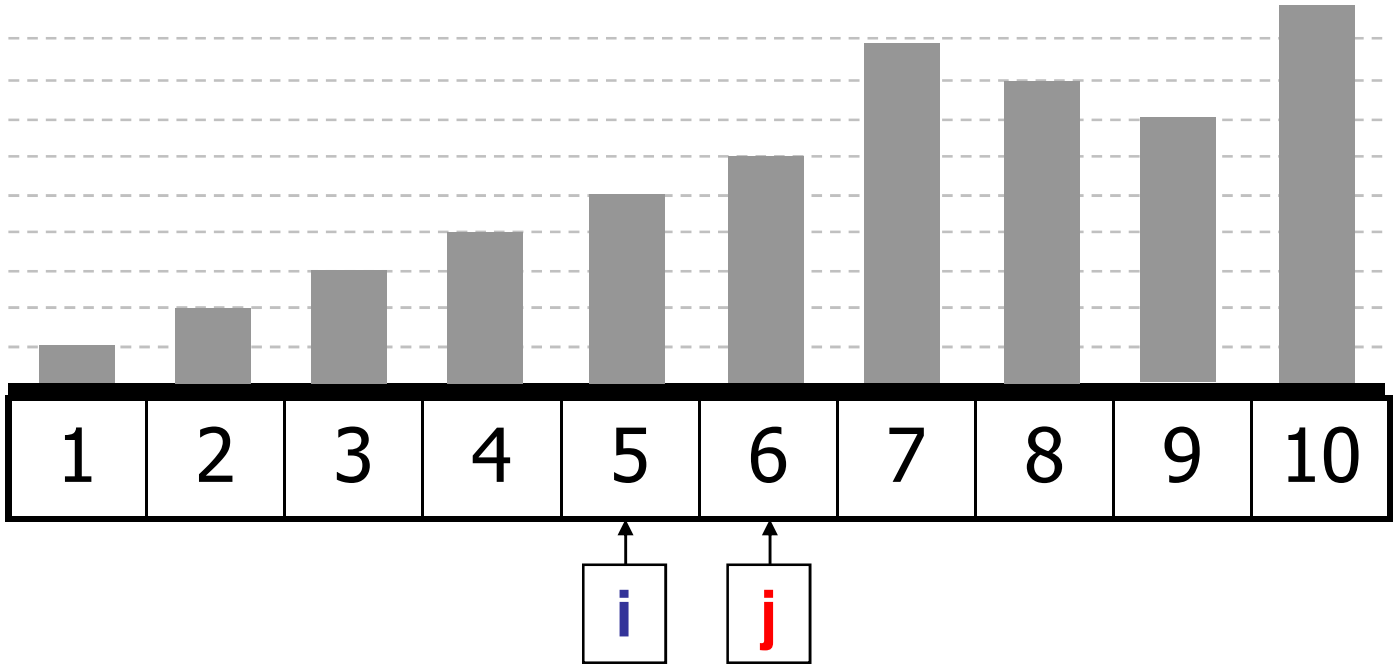
Min 10

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
         x := A[Min];
         A[Min] := A[i];
         A[i] := x;
   End;
```



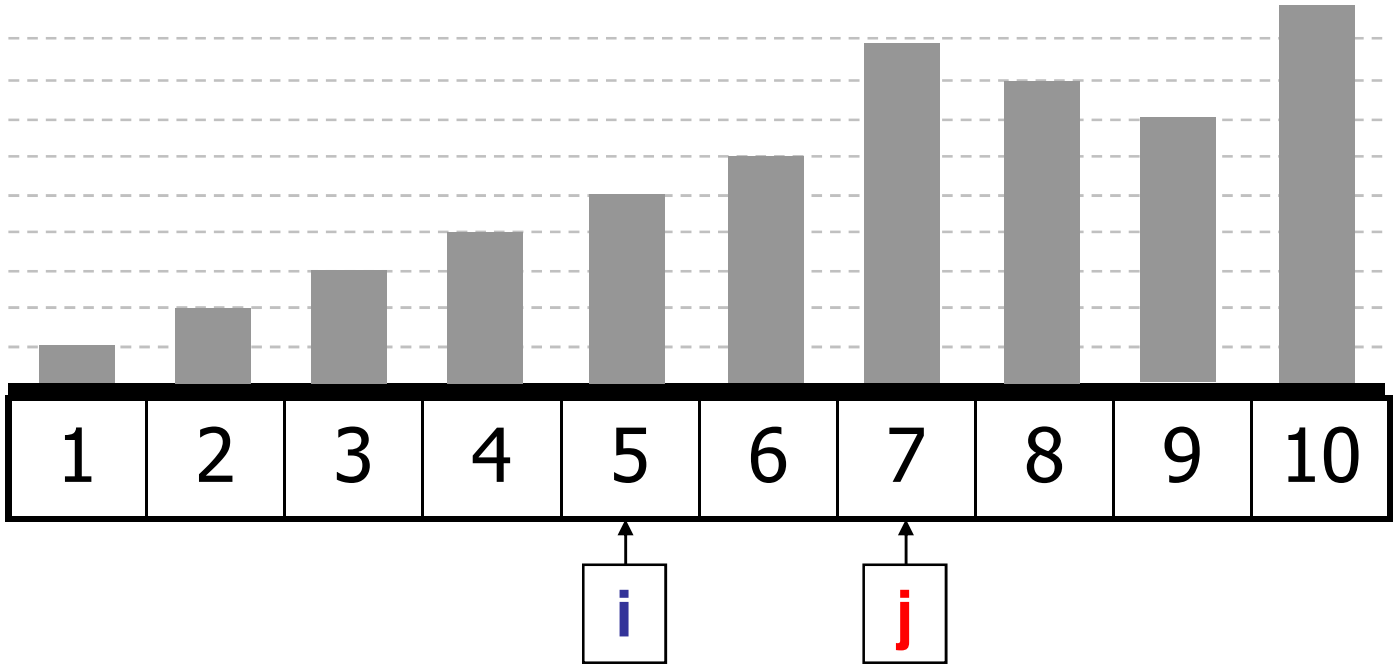
Min 5

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



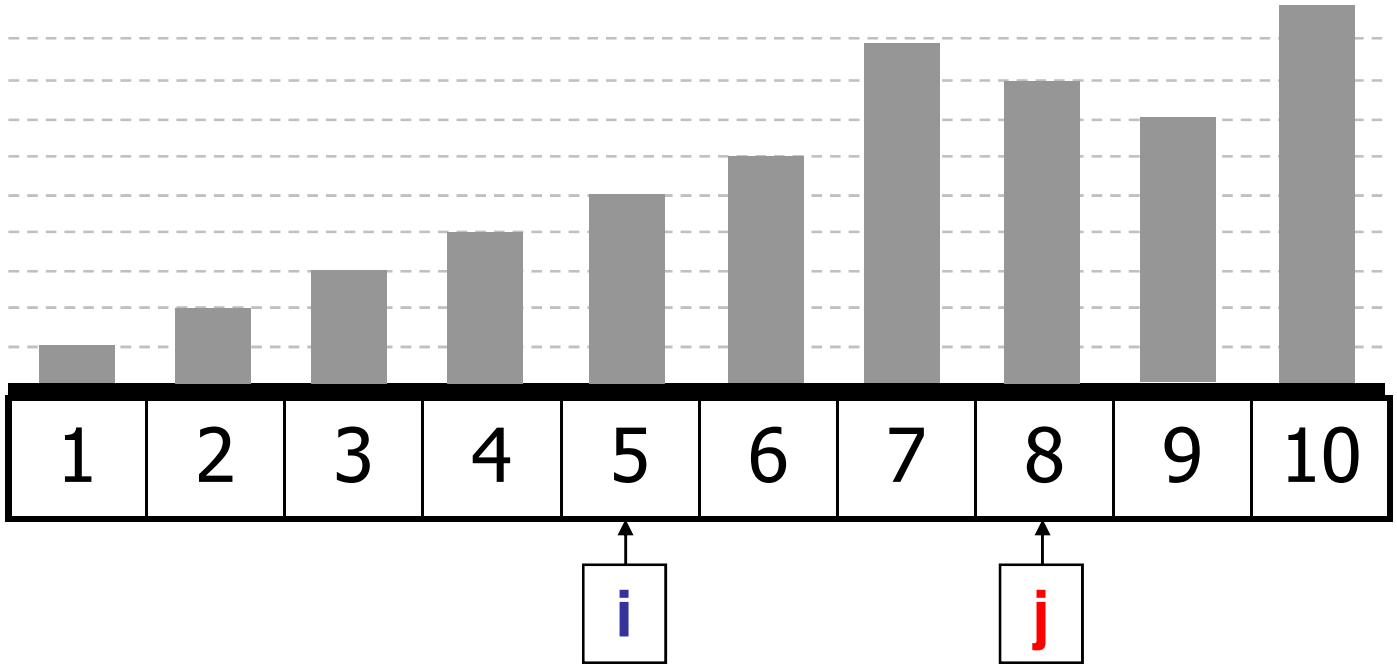
Min 5

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



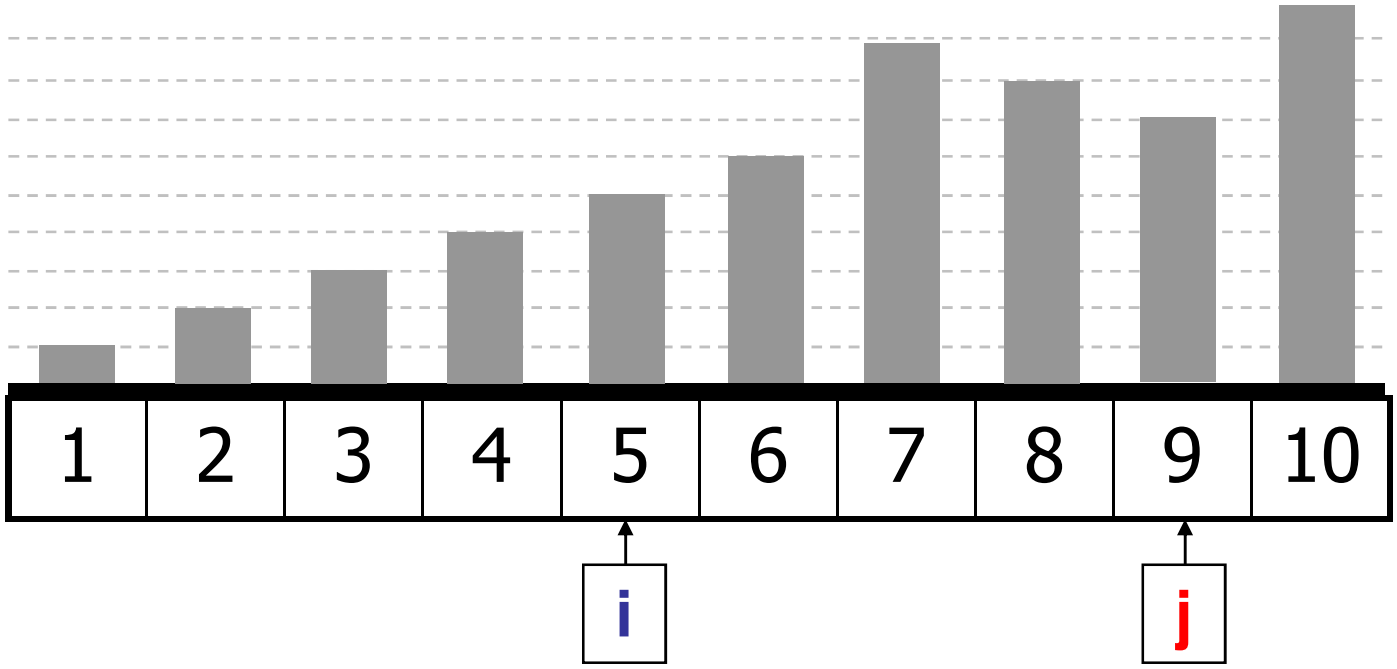
Min 5

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



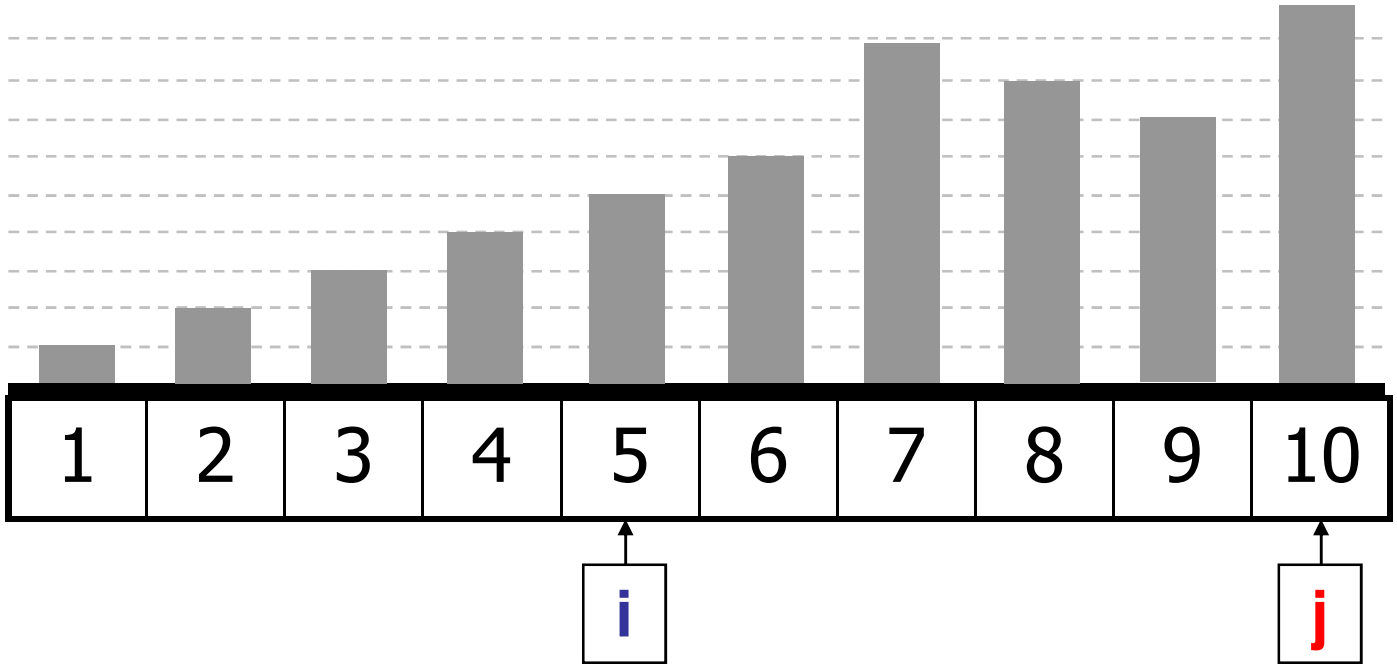
Min 5

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



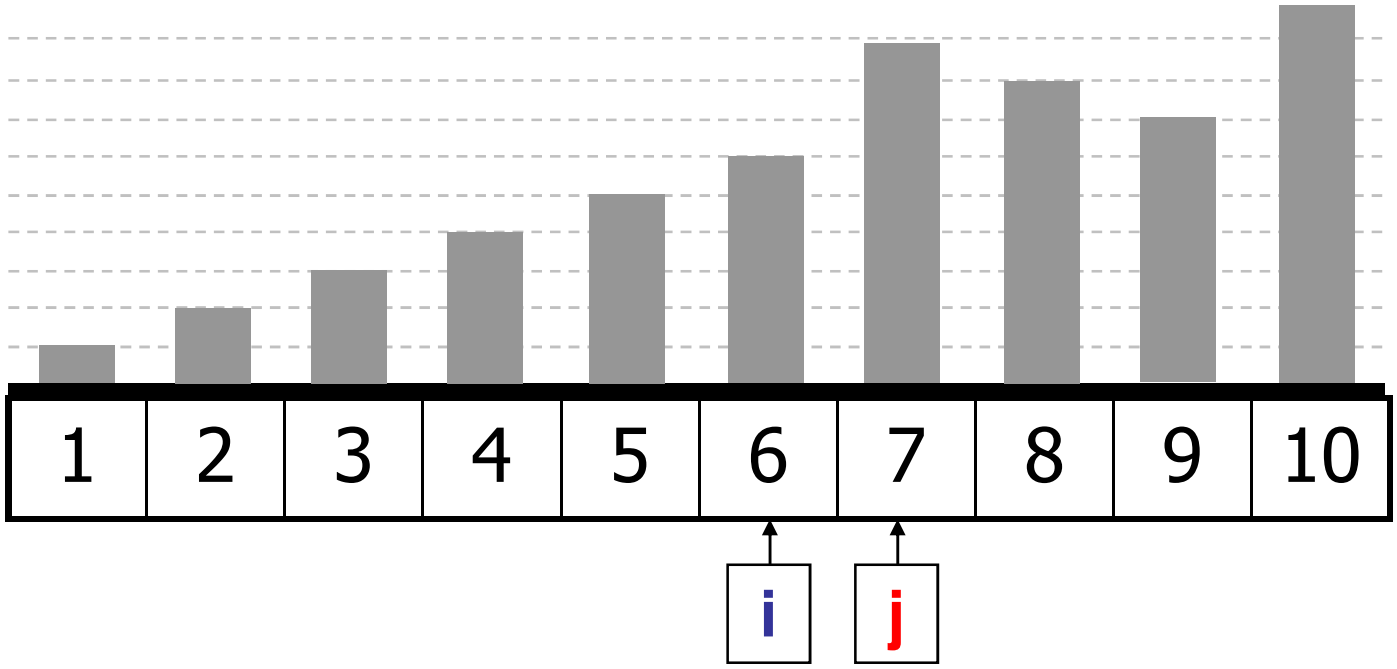
Min 5

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



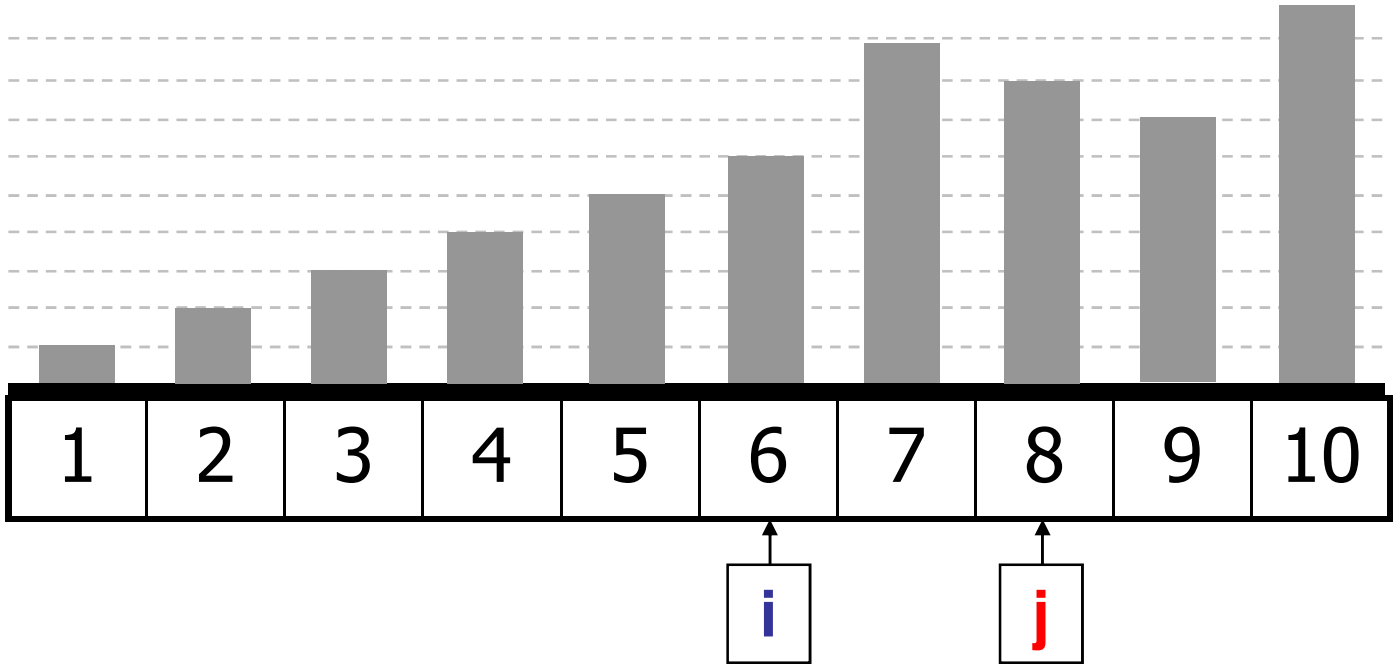
Min 5

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



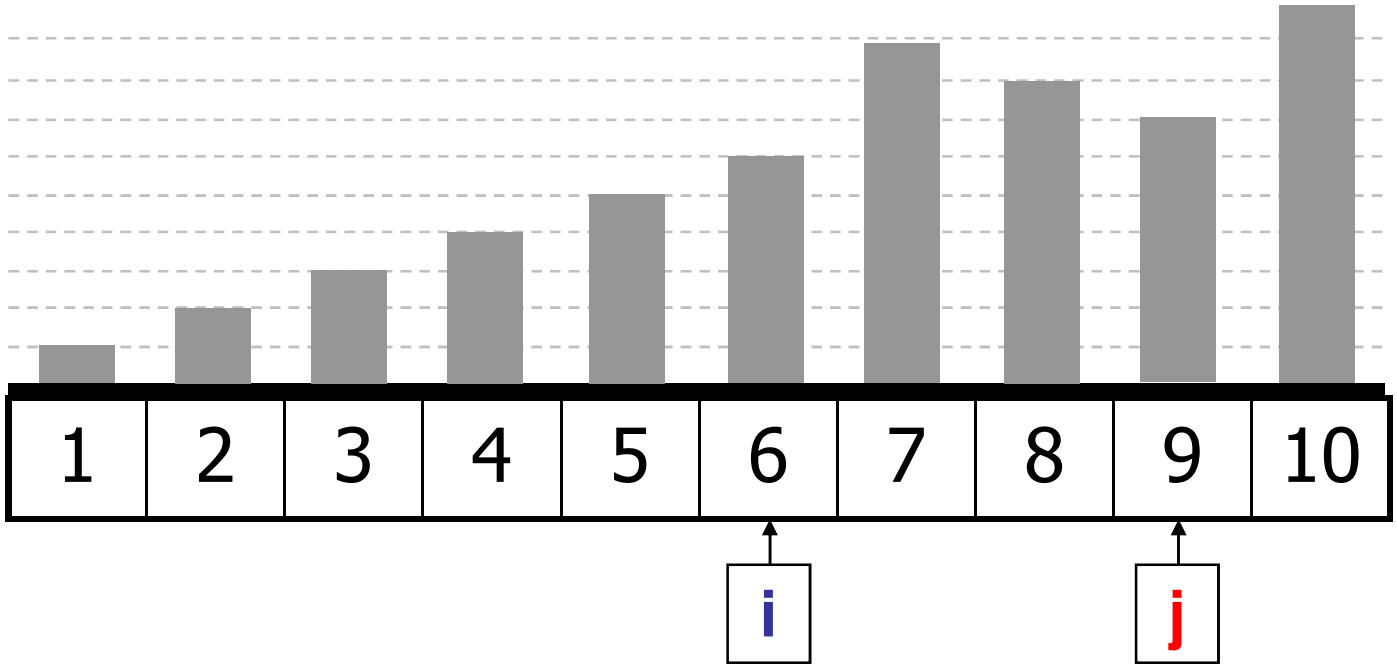
Min 6

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



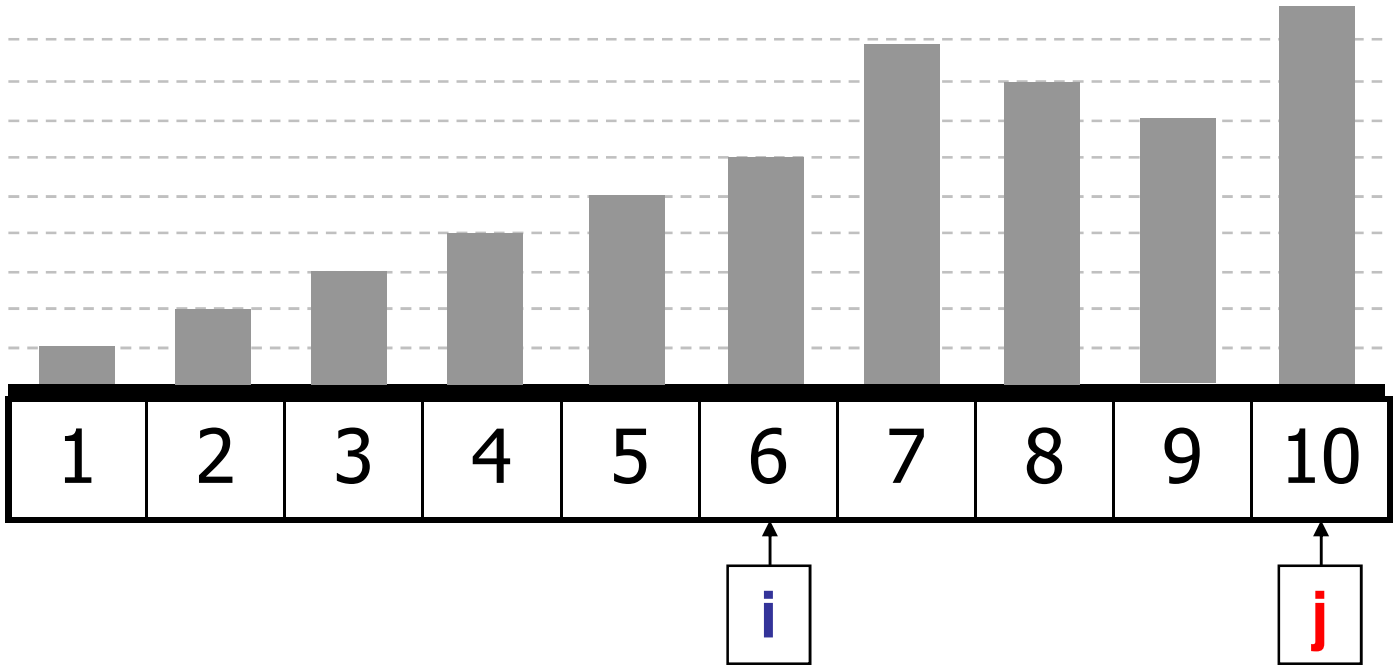
Min 6

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



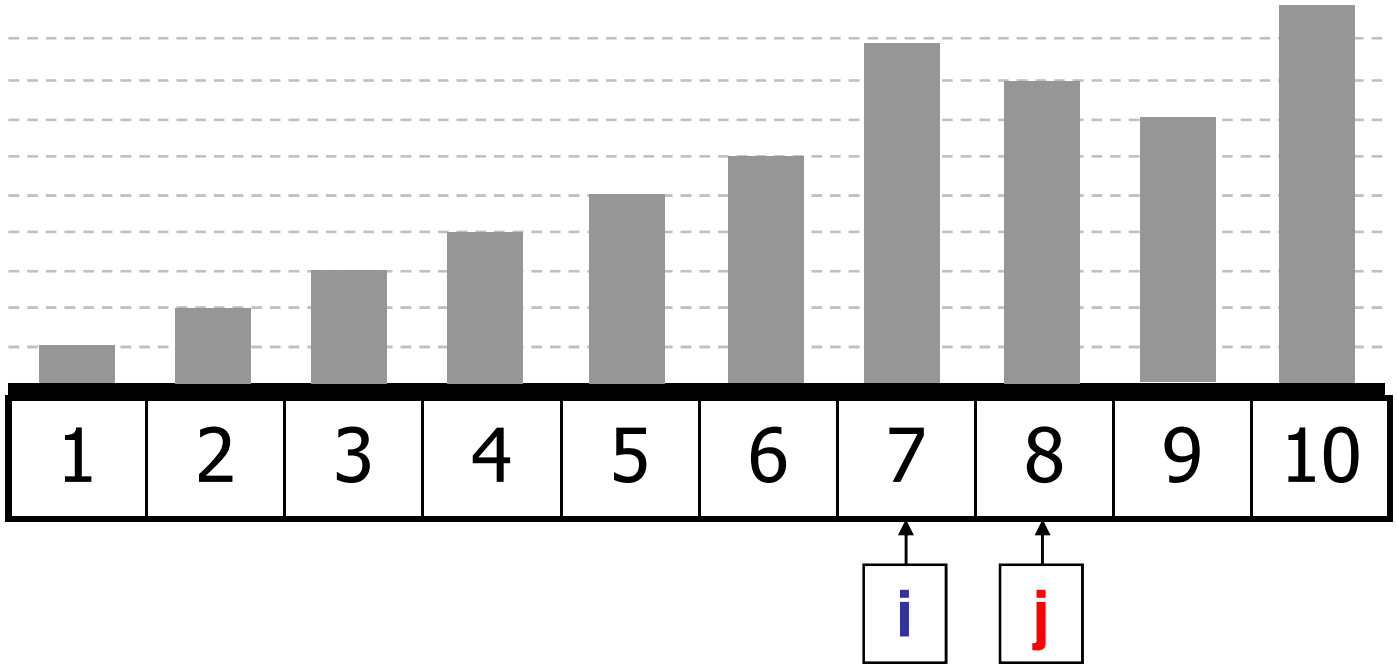
Min 6

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



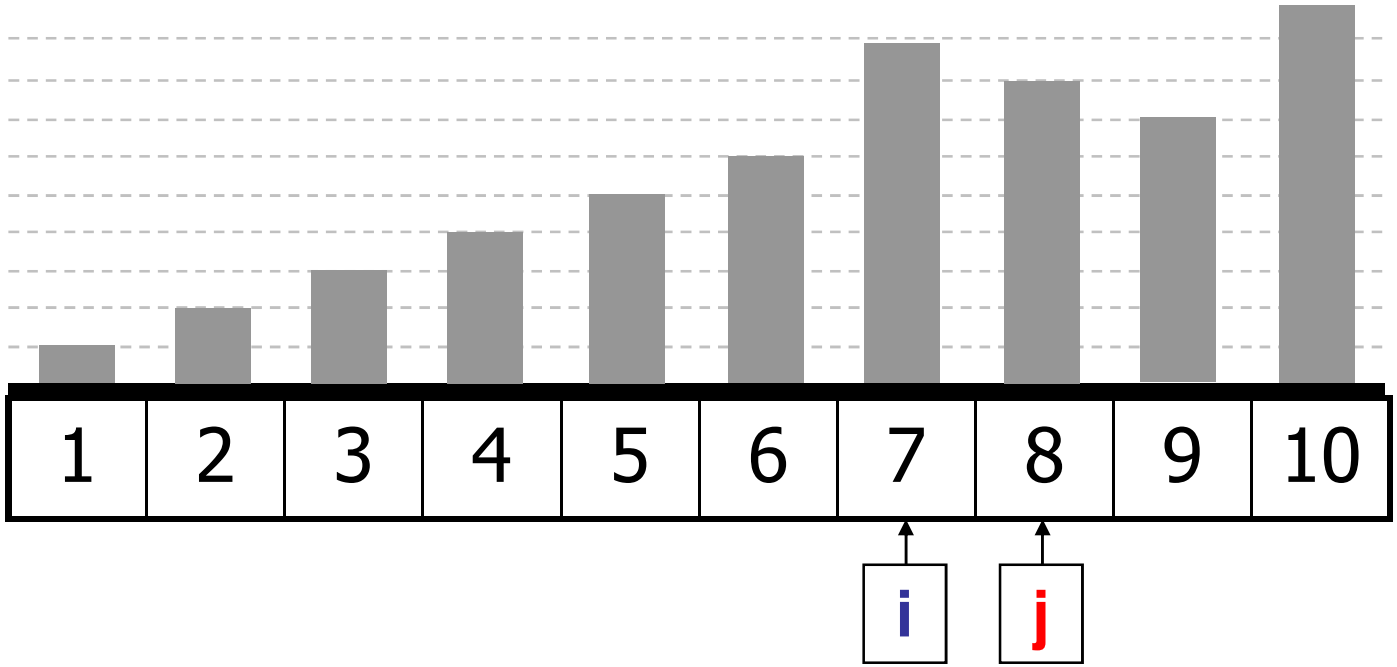
Min 6

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



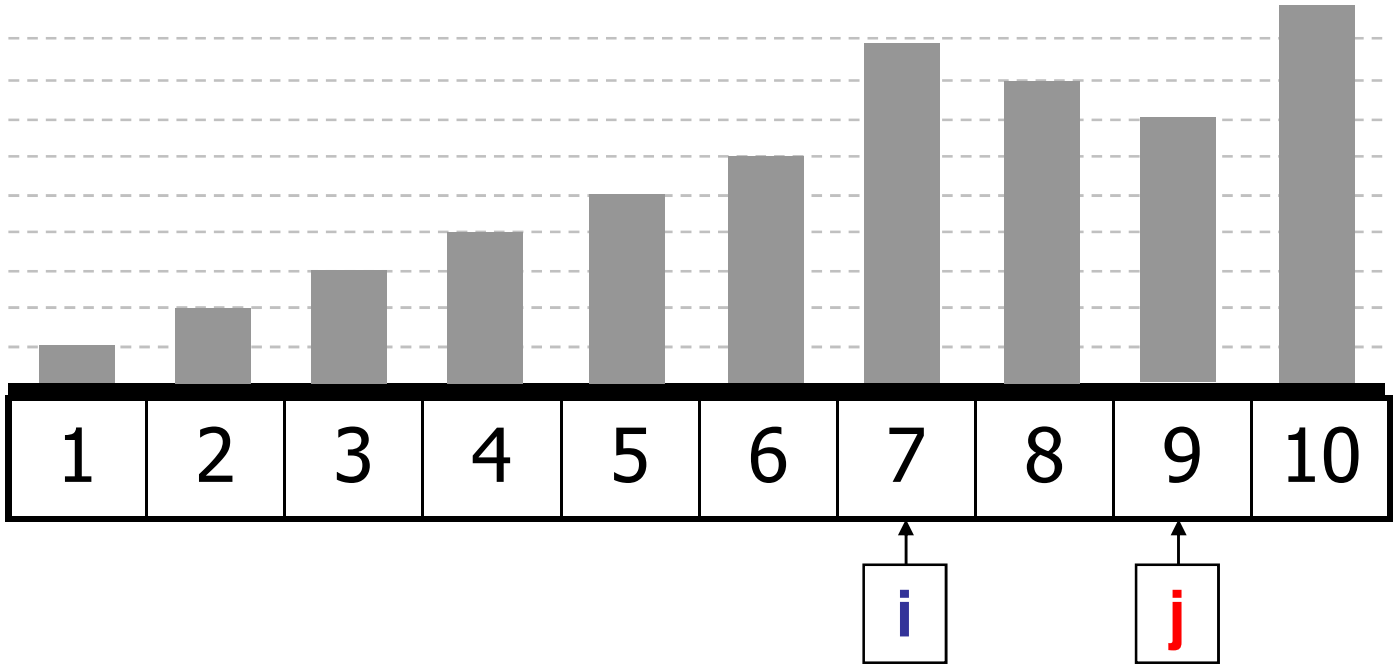
Min 7

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



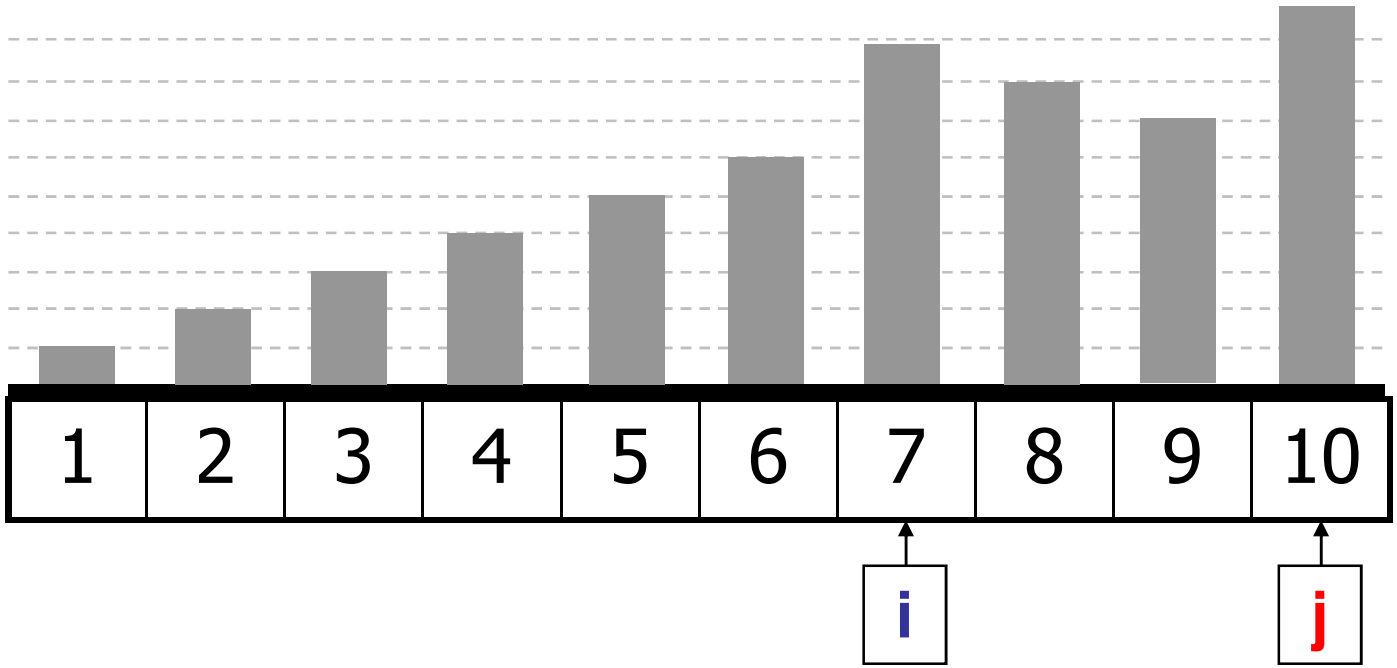
Min 8

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j; ←
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



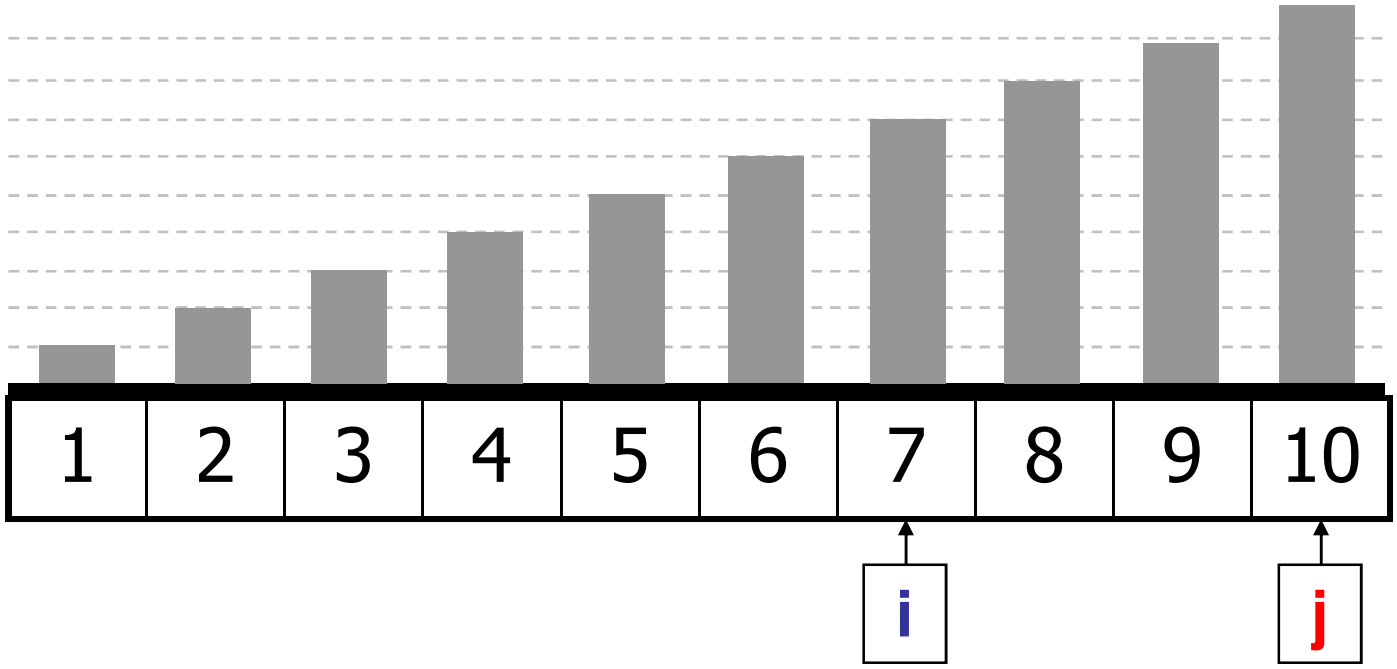
Min 9

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



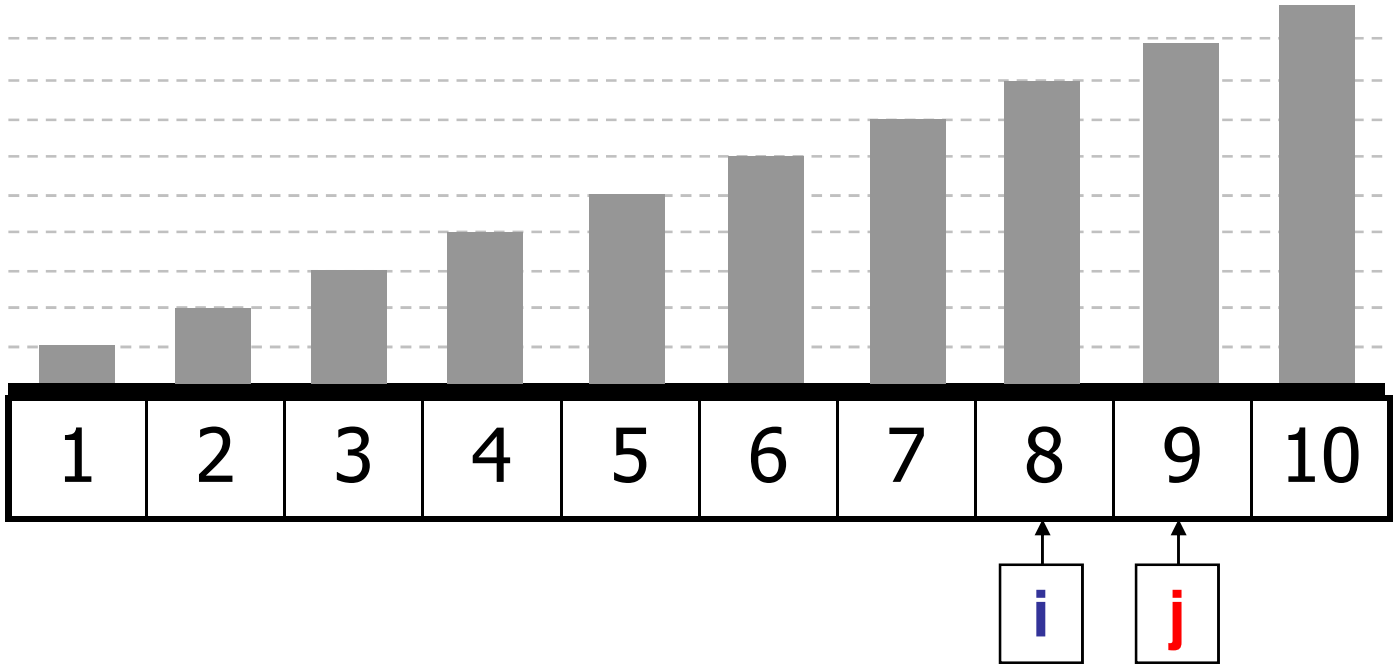
Min 9

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
         x := A[Min];
         A[Min] := A[i];
         A[i] := x;
   End;
```



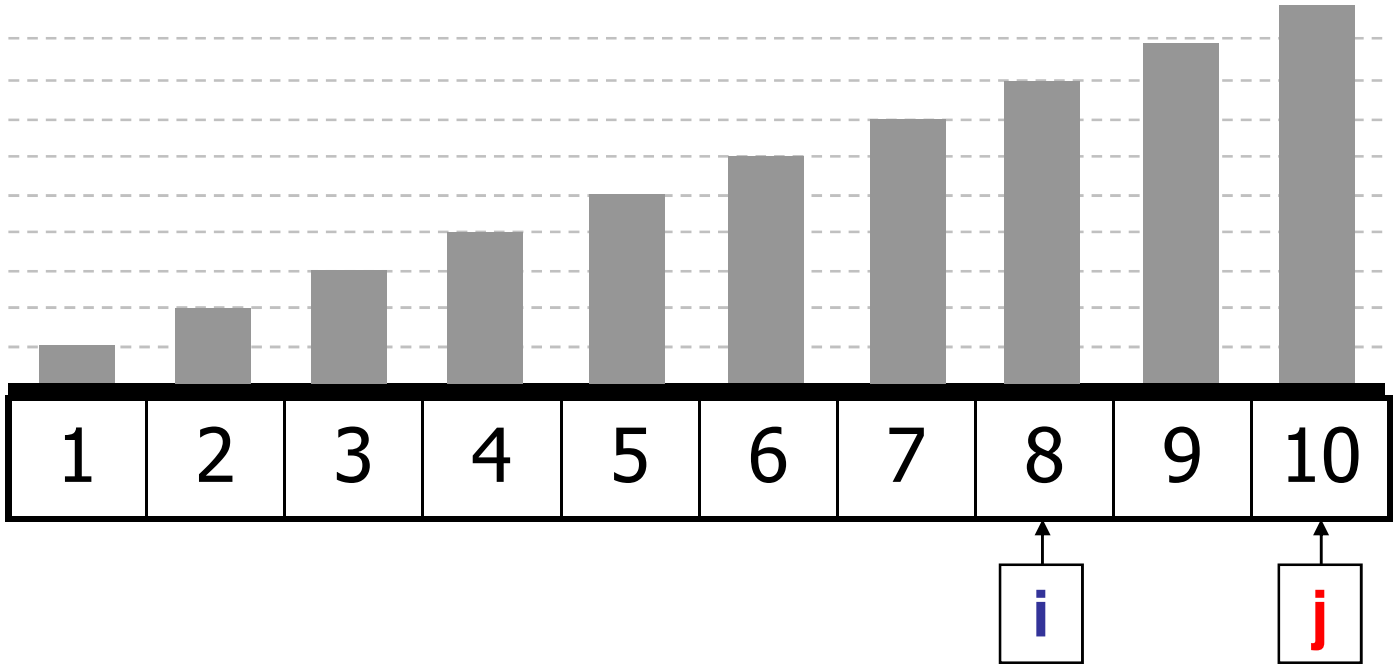
Min 9

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



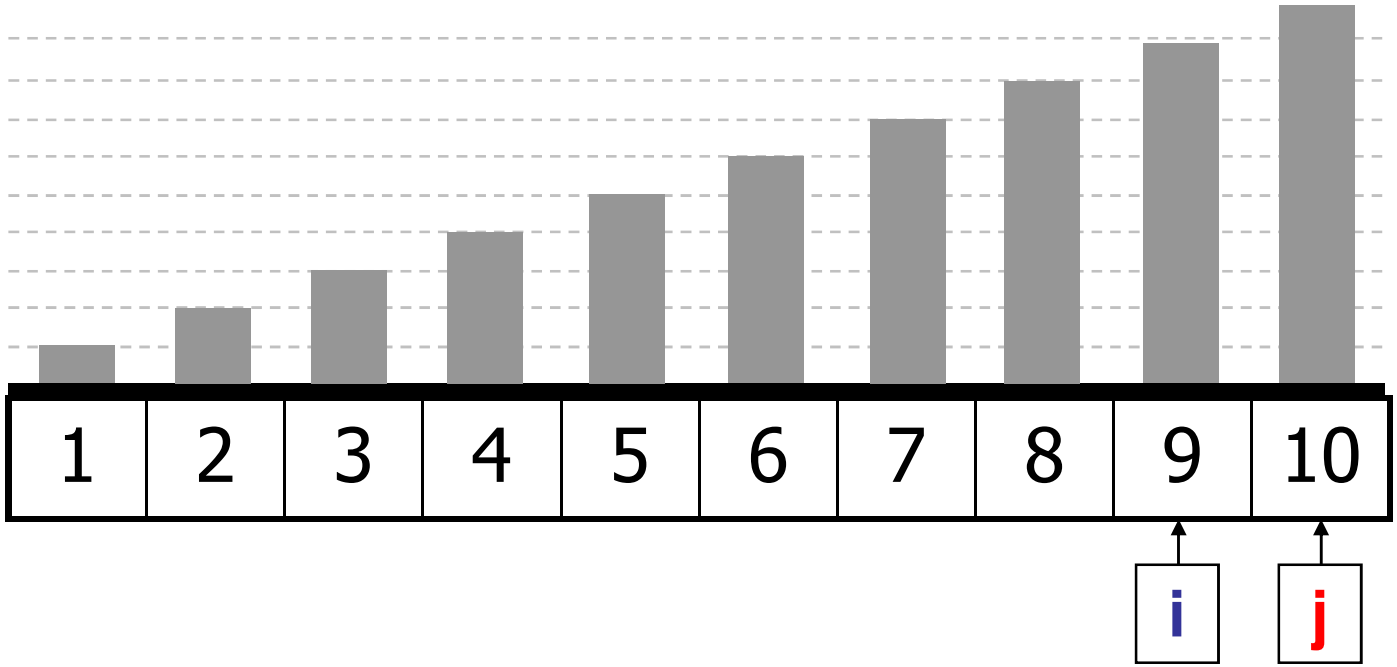
Min 8

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do ←
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



Min | 8 |

```
for i := 1 to n-1 do
  Begin
     min := i;
     for j := i+1 to n do
     if A[j] < A[Min]   then Min := j;
          x := A[Min];
          A[Min] := A[i];
          A[i] := x;
   End;
```



Min 9

FIM!

Ordenação por Inserção

- Neste método, a seqüência a ser ordenada é varrida a partir da posição 2, e cada item apanhado na seqüência é inserido no seu lugar apropriado em relação aos item anteriores da seqüência;
- A colocação do item no seu lugar apropriado na seqüência destino é realizada movendo-se os itens com chaves maiores para a direita e então inserindo o item na posição deixada vazia.

Ordenação por Inserção

- Exemplo:

	1	2	3	4	5	6
Chaves Inicias:	**O**	R	D	E	N	A
I=2	**O**	**R**	D	E	N	A
I=3	**D**	**O**	**R**	E	N	A
I=4	**D**	**E**	**O**	**R**	N	A
I=5	**D**	**E**	**N**	**O**	**R**	A
I=6	**A**	**D**	**E**	**N**	**O**	**R**

- ***Algoritmo:***

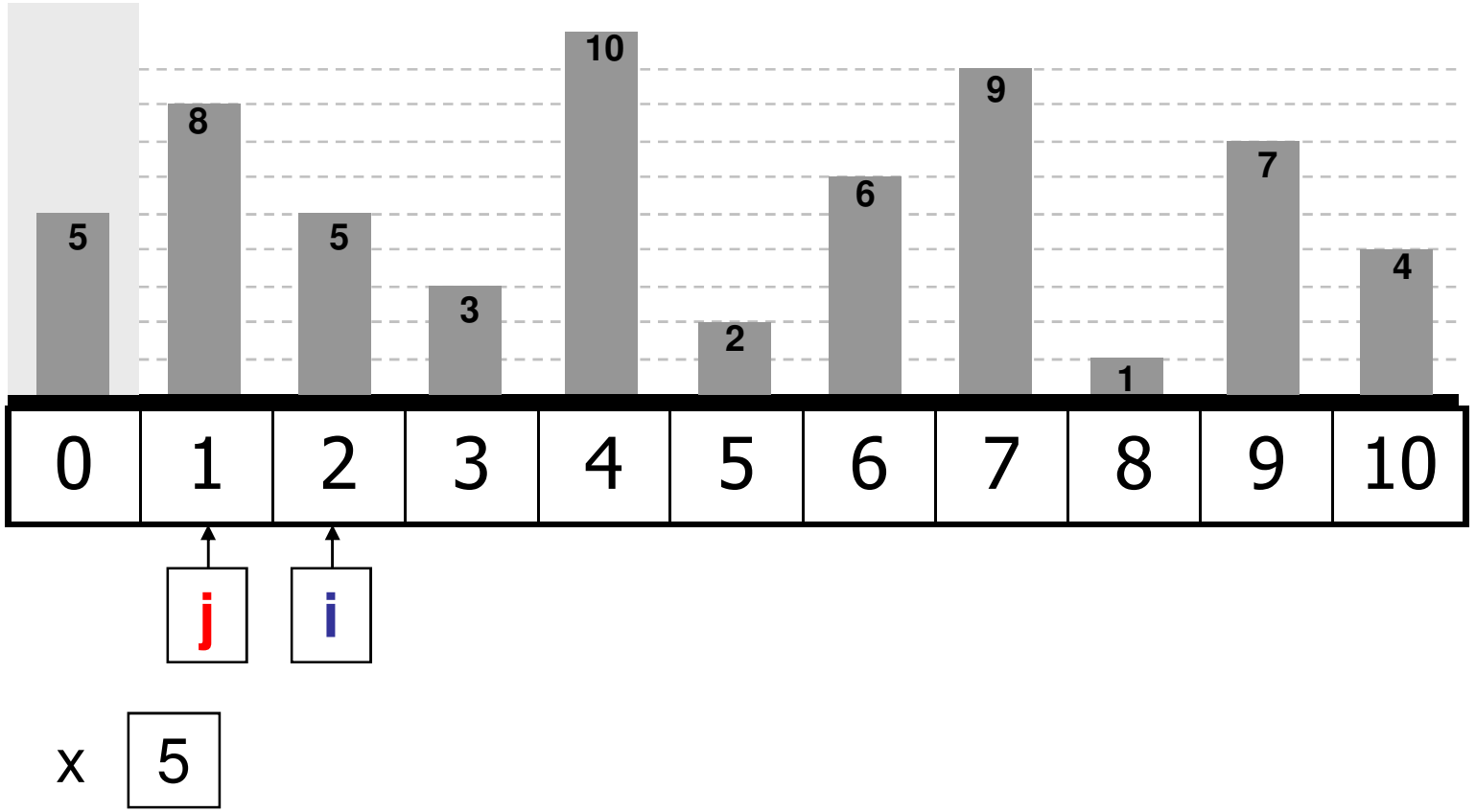
```
Procedure Insercao (var A: vetor);
Var
    i, j, x: integer;
Begin
    for i := 2 to n do
    Begin
          x := A[i];   {guardando cópia do pivô}
          j := i-1;
          A[0] := x;  {sentinela} {neste caso vetor=array [0..n]}
          while x < A[j] do
            Begin
              A[j+1] := A[j];          {Empurra maiores p/ direita}
              j := j-1;
            End;
          A[j+1] := x;      {Insere na posição apropriada}
    End;
End;
```

Ordenação por Inserção

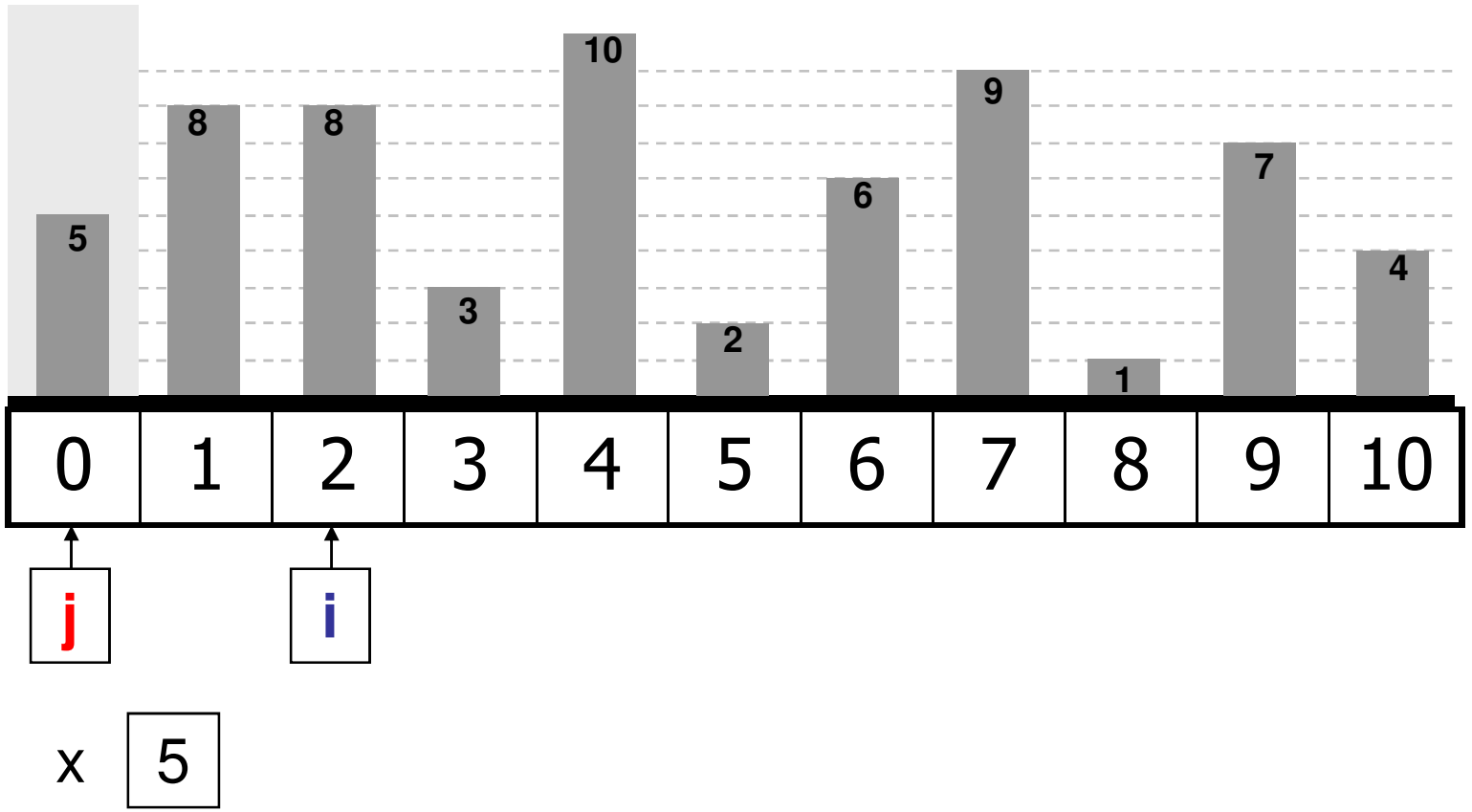
- O método de inserção é adequado para os casos em que as informações já estão “quase” ordenadas, pois neste caso seu custo é muito baixo;
- É também um bom método quando se deseja adicionar uns poucos itens à uma seqüência já ordenada.

```
for i := 2 to n do ←
 Begin
   x := A[i]; ←
   j := i-1; ←
  A[0] := x; ← {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

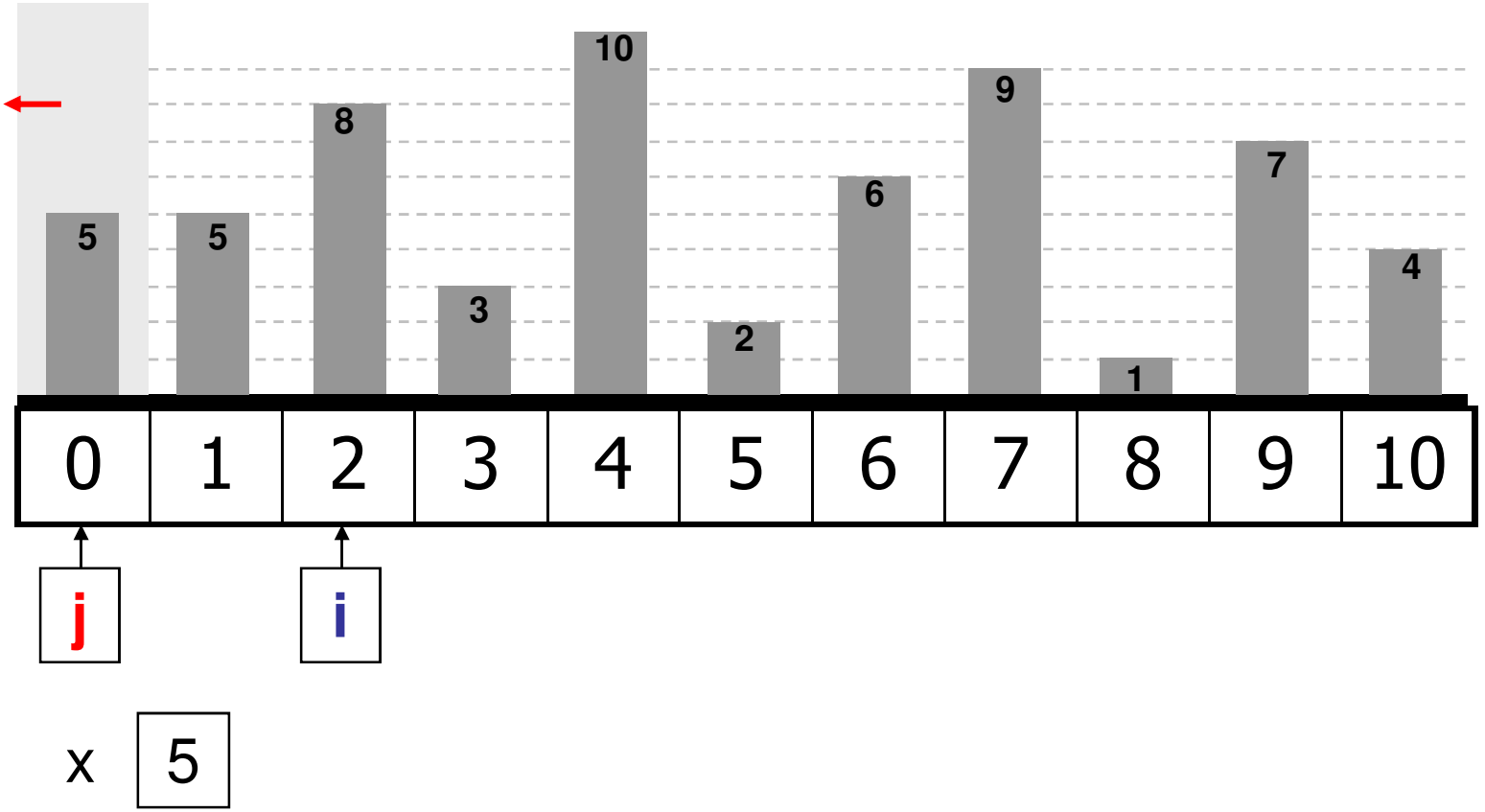
As setas vermelhas indicam operações já realizadas!



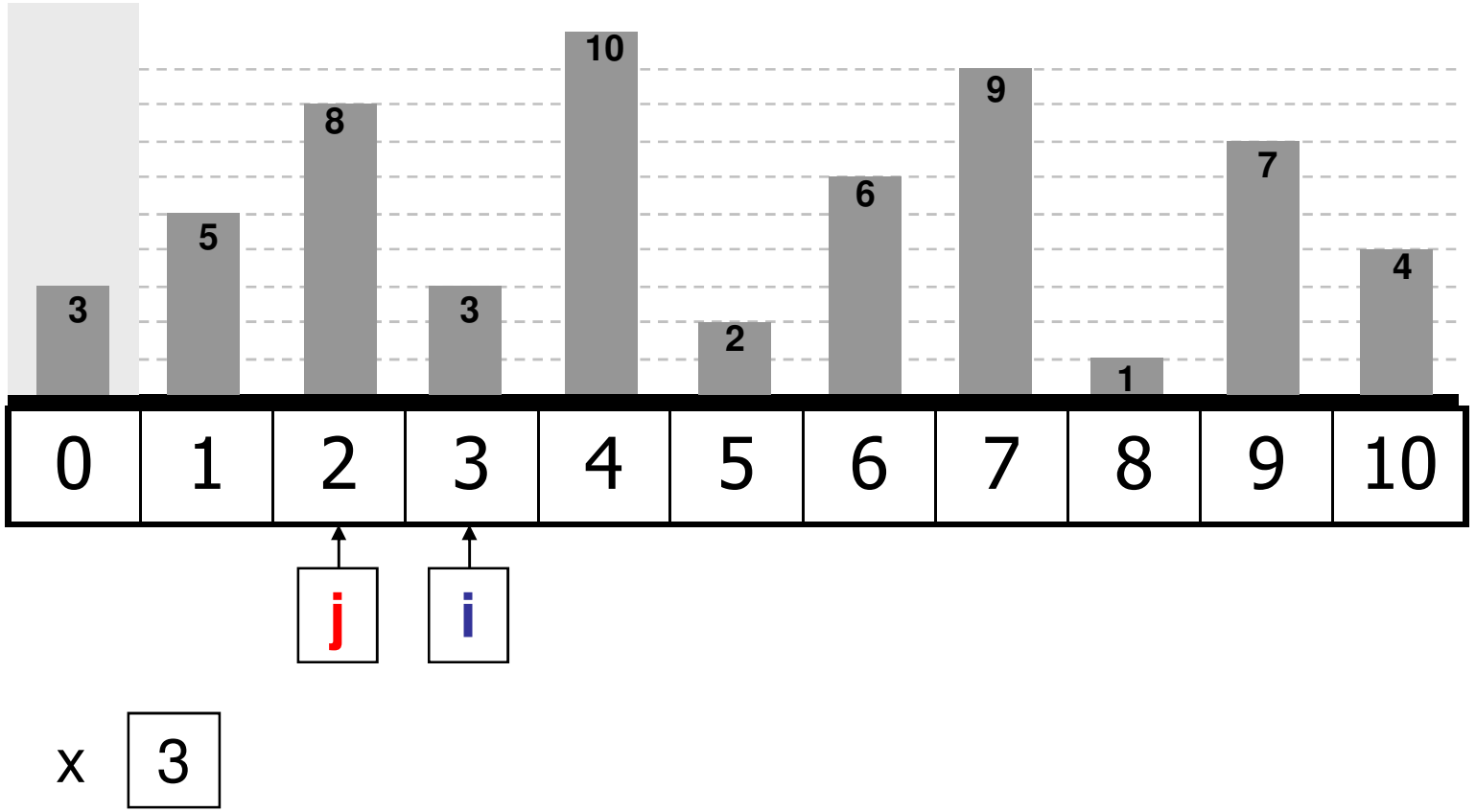
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



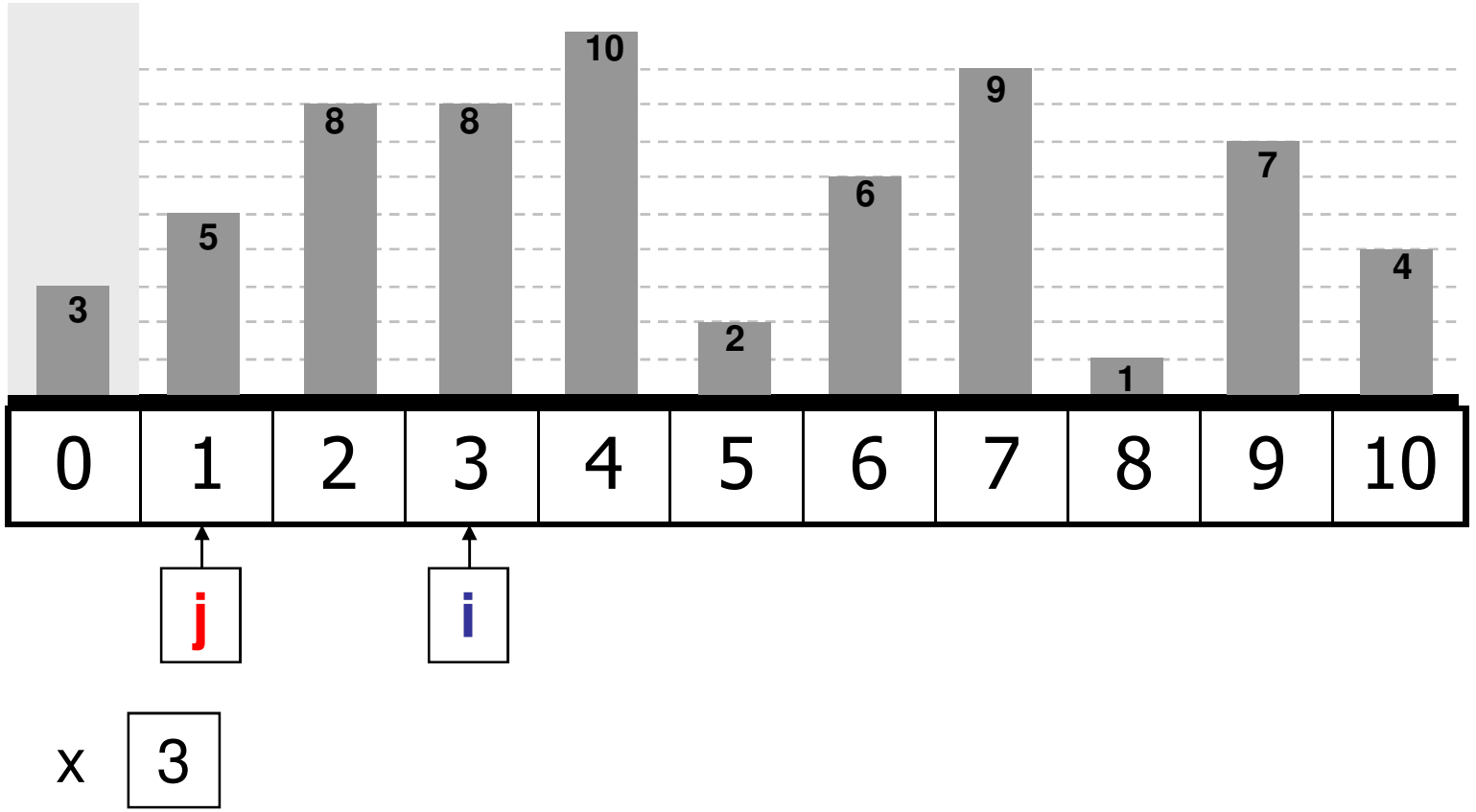
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



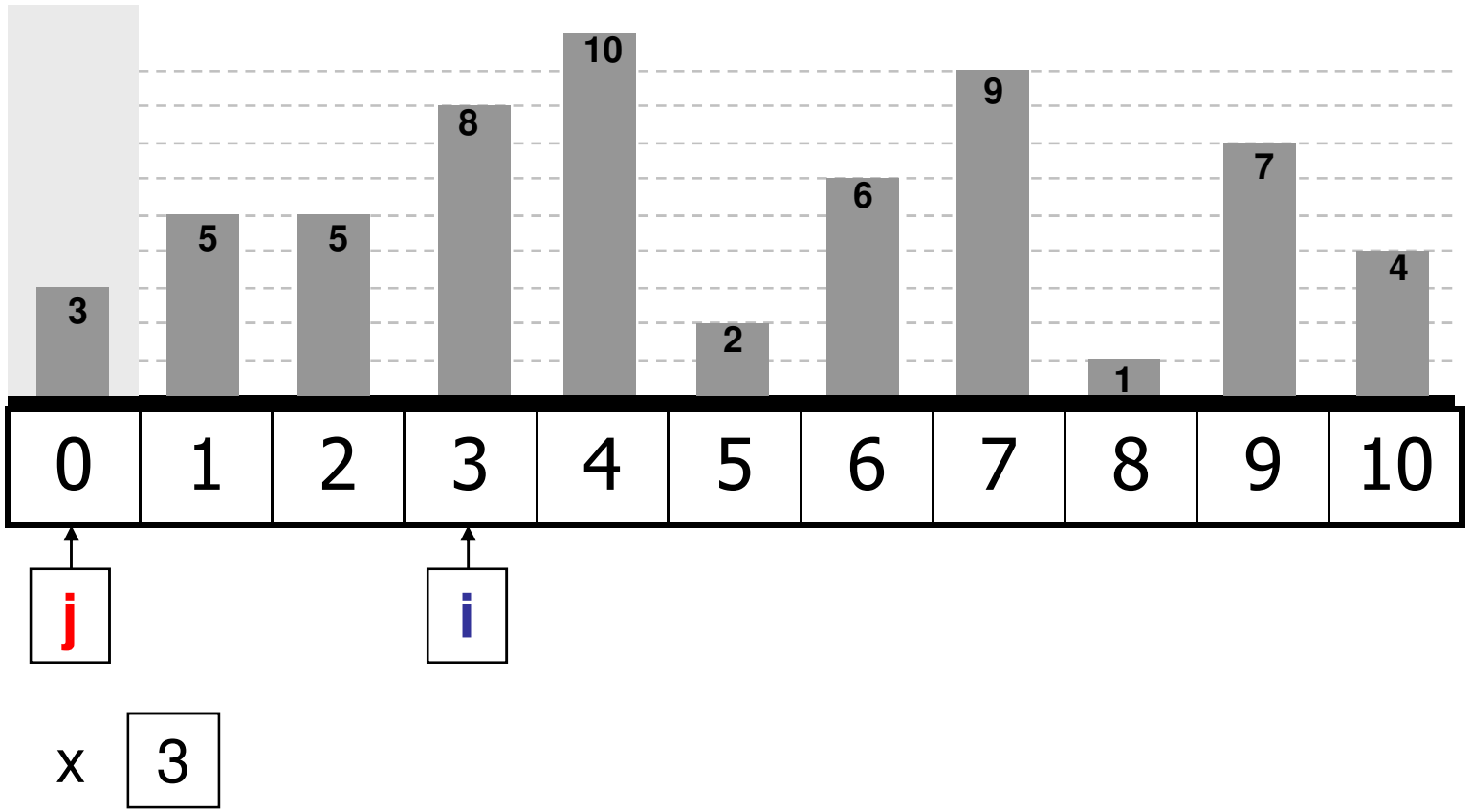
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x; {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



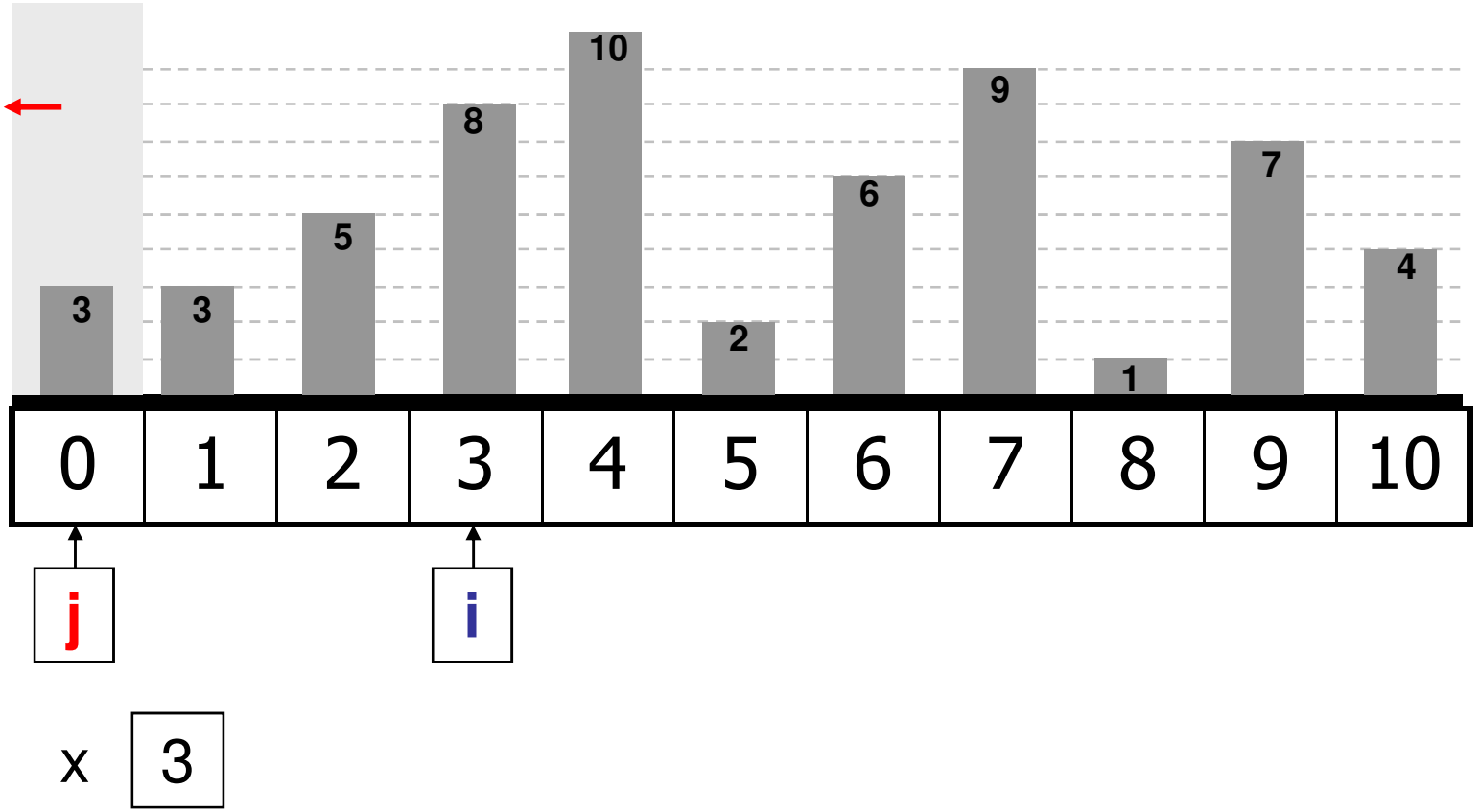
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



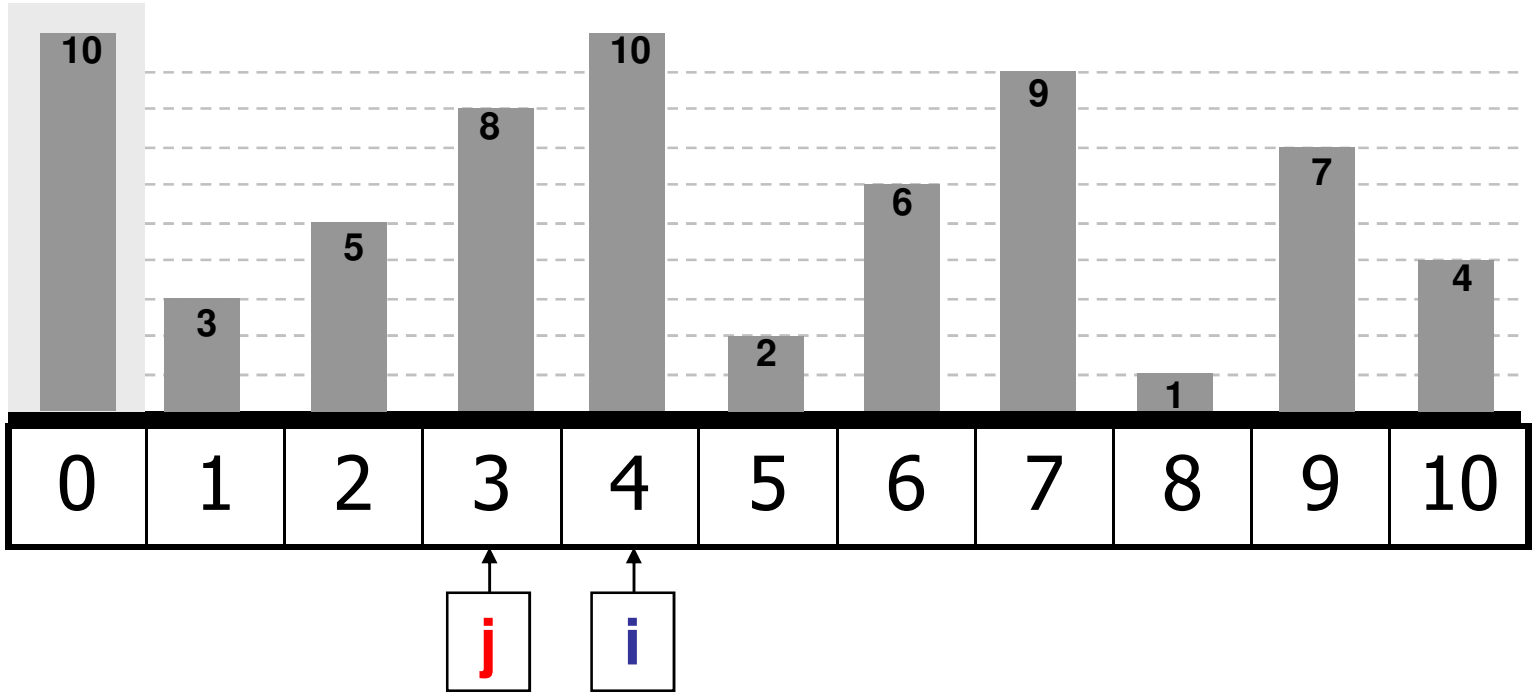
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

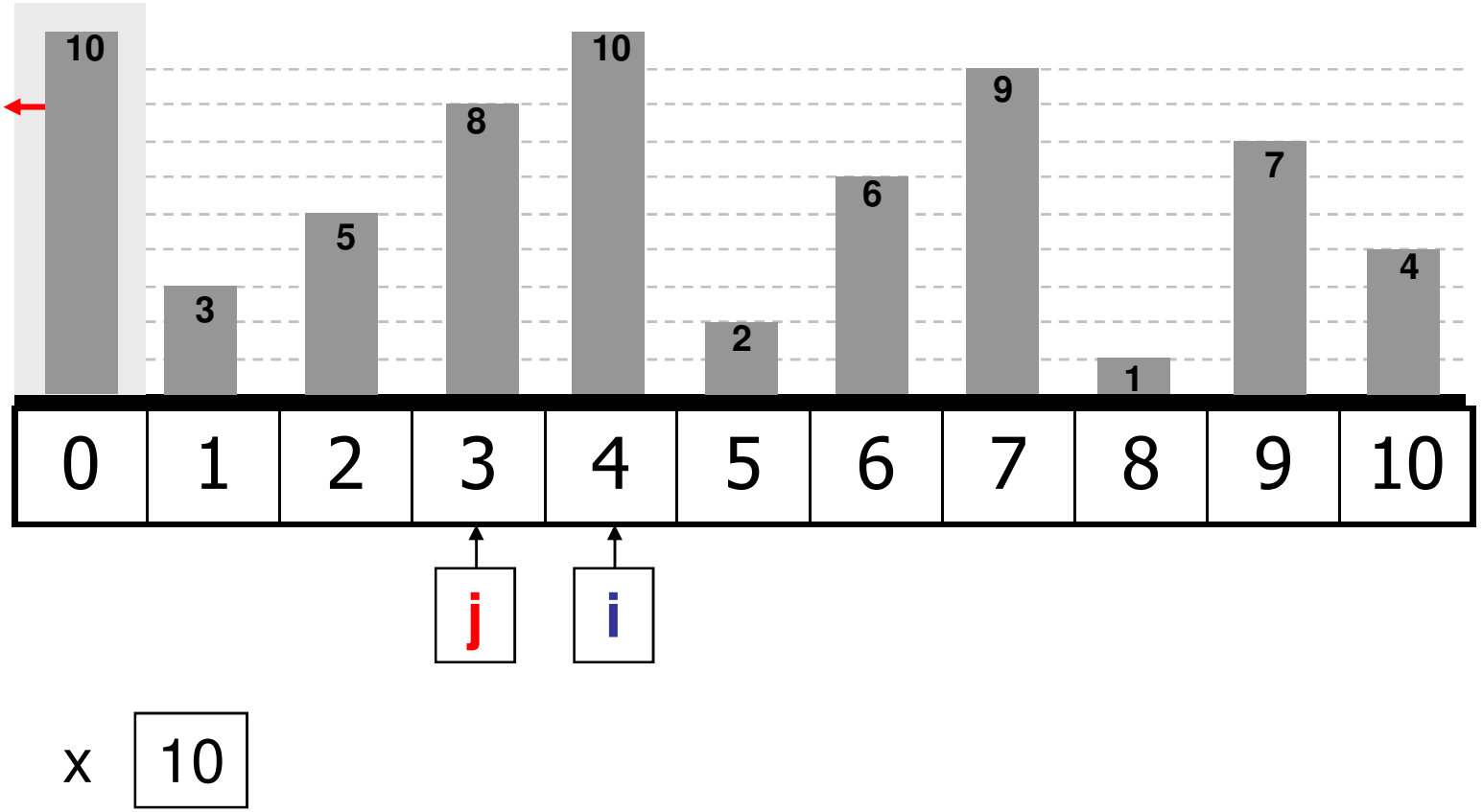


```
for i := 2 to n do ←
 Begin
   x := A[i]; ←
   j := i-1; ←
  A[0] := x; ← {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

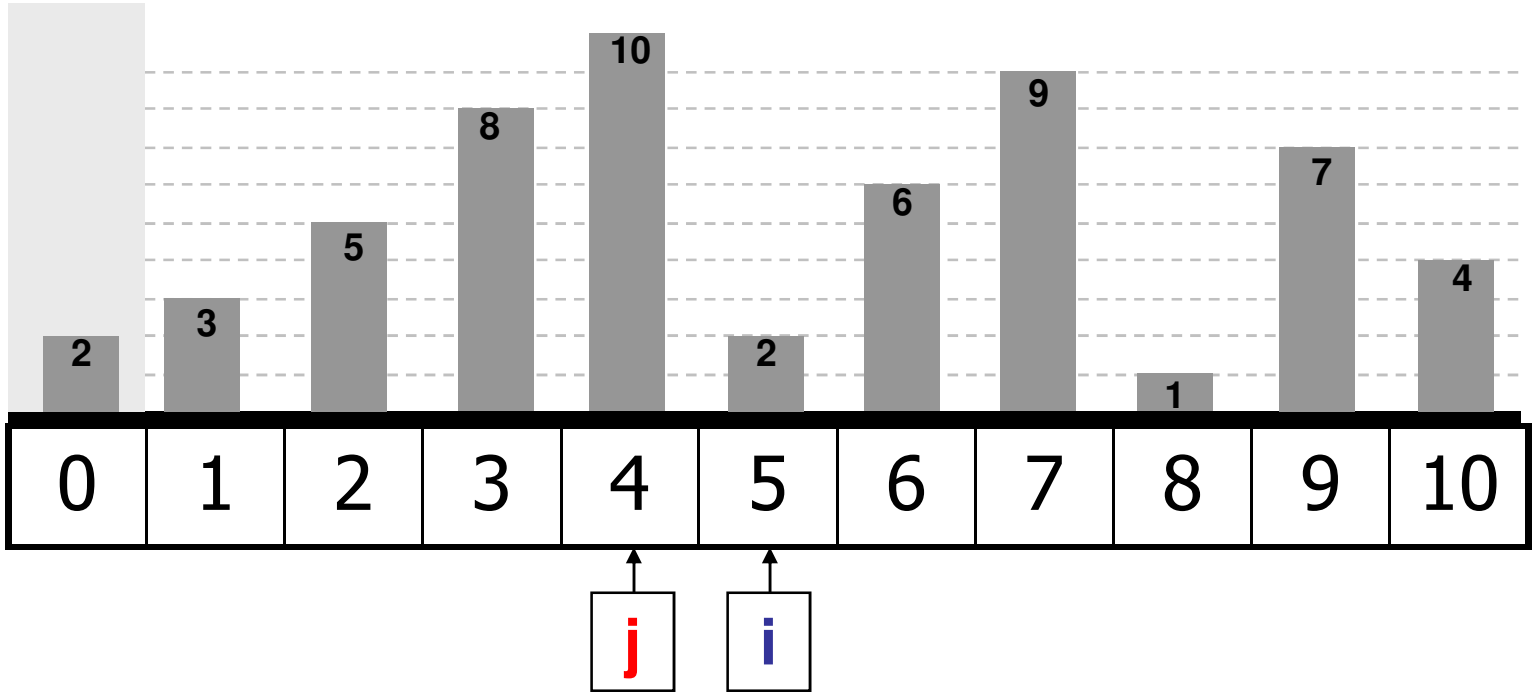


x 10

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

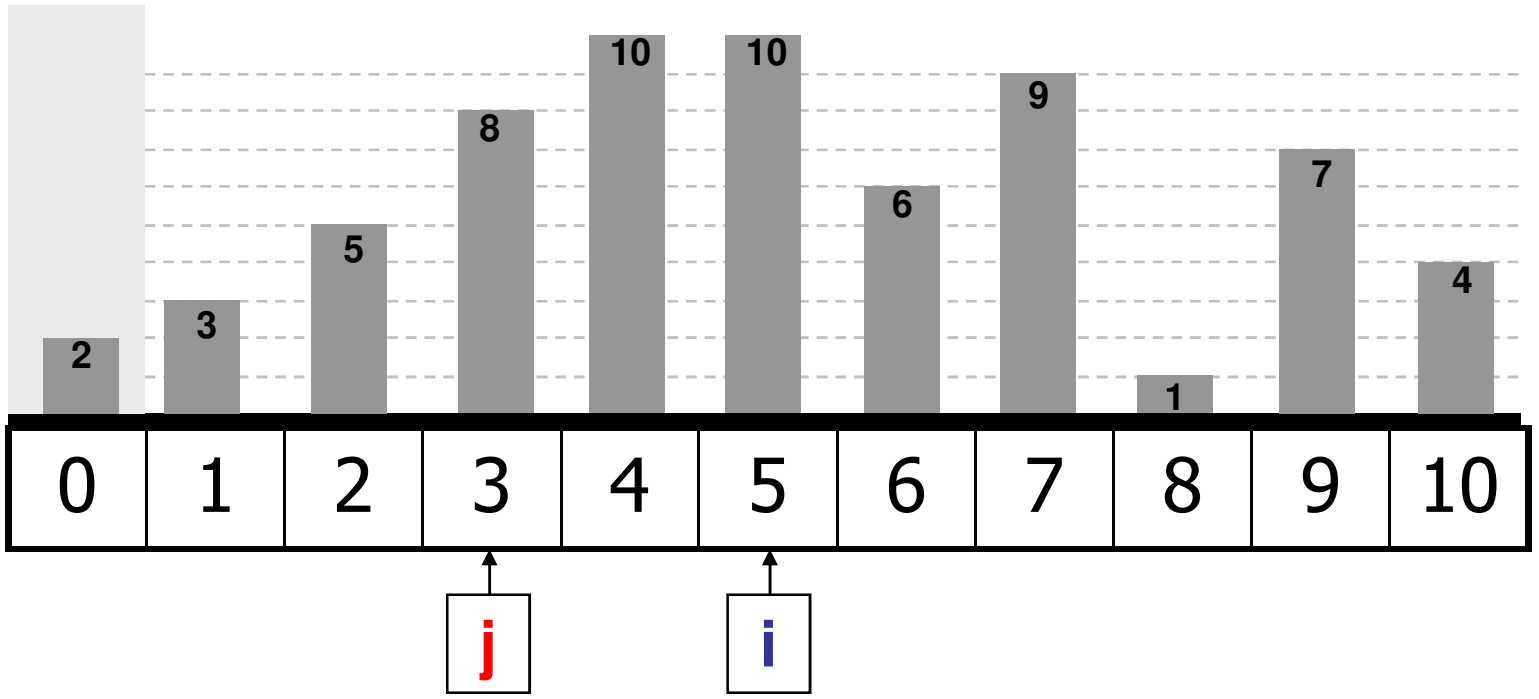


```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x; {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



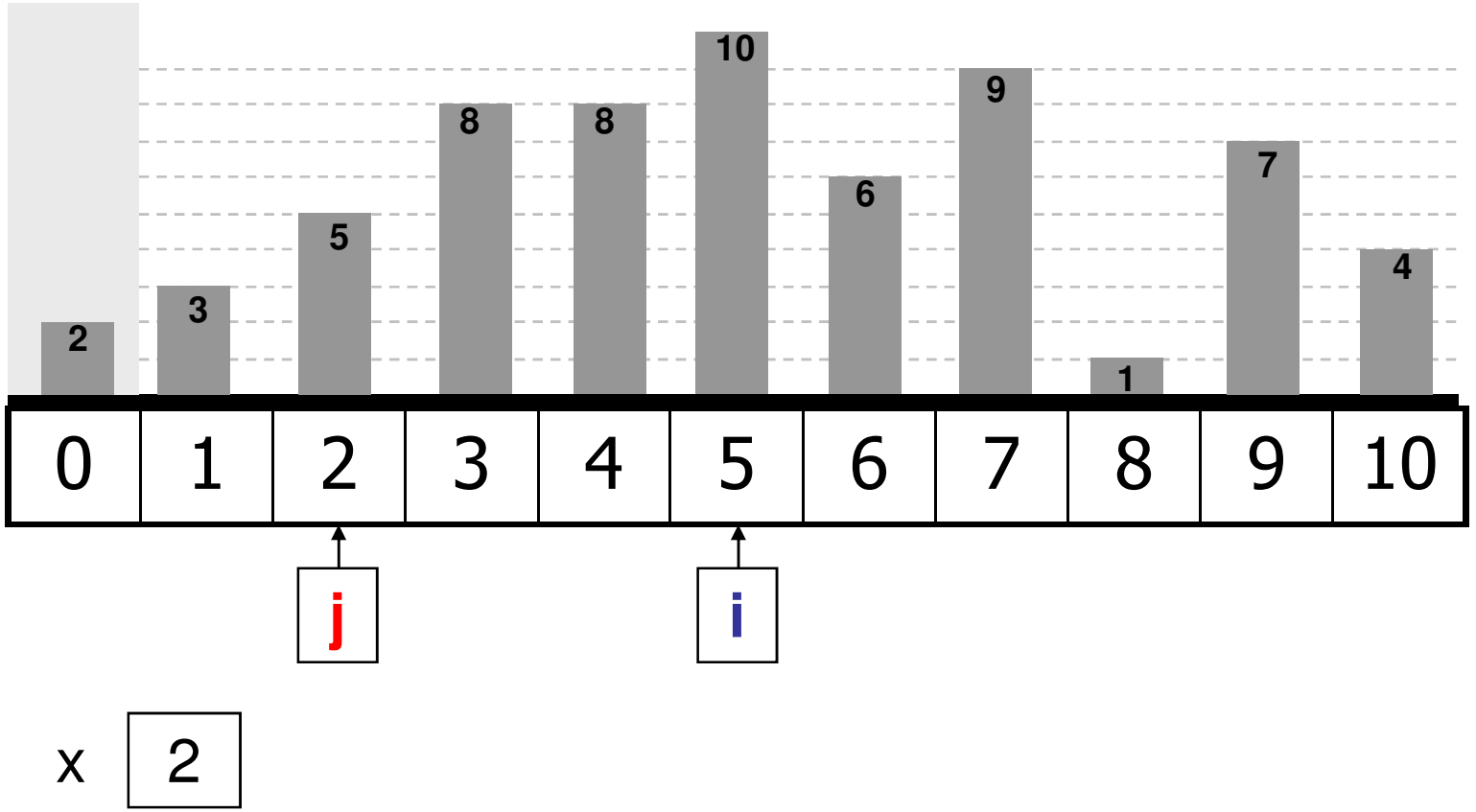
x 2

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];  {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

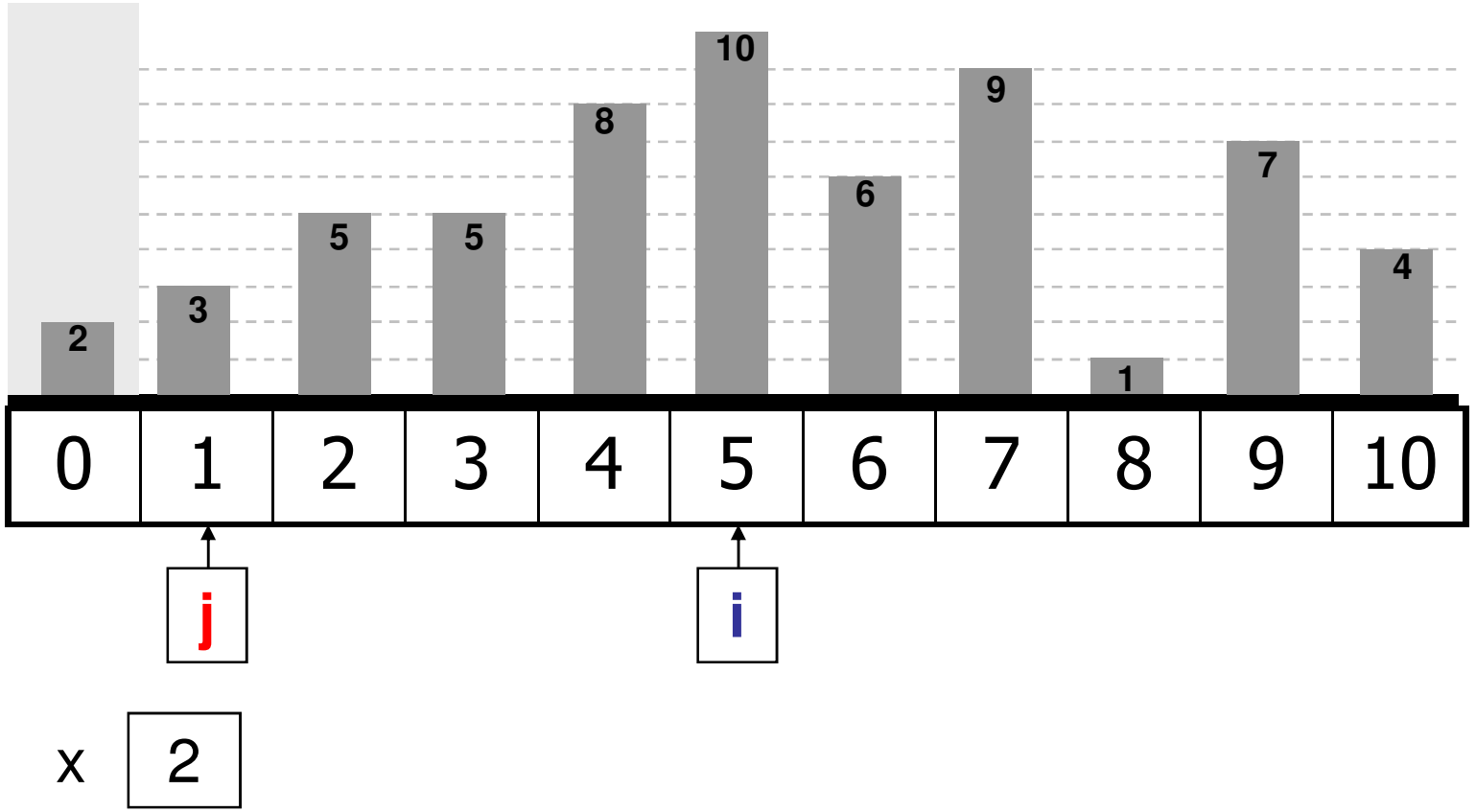


x | 2

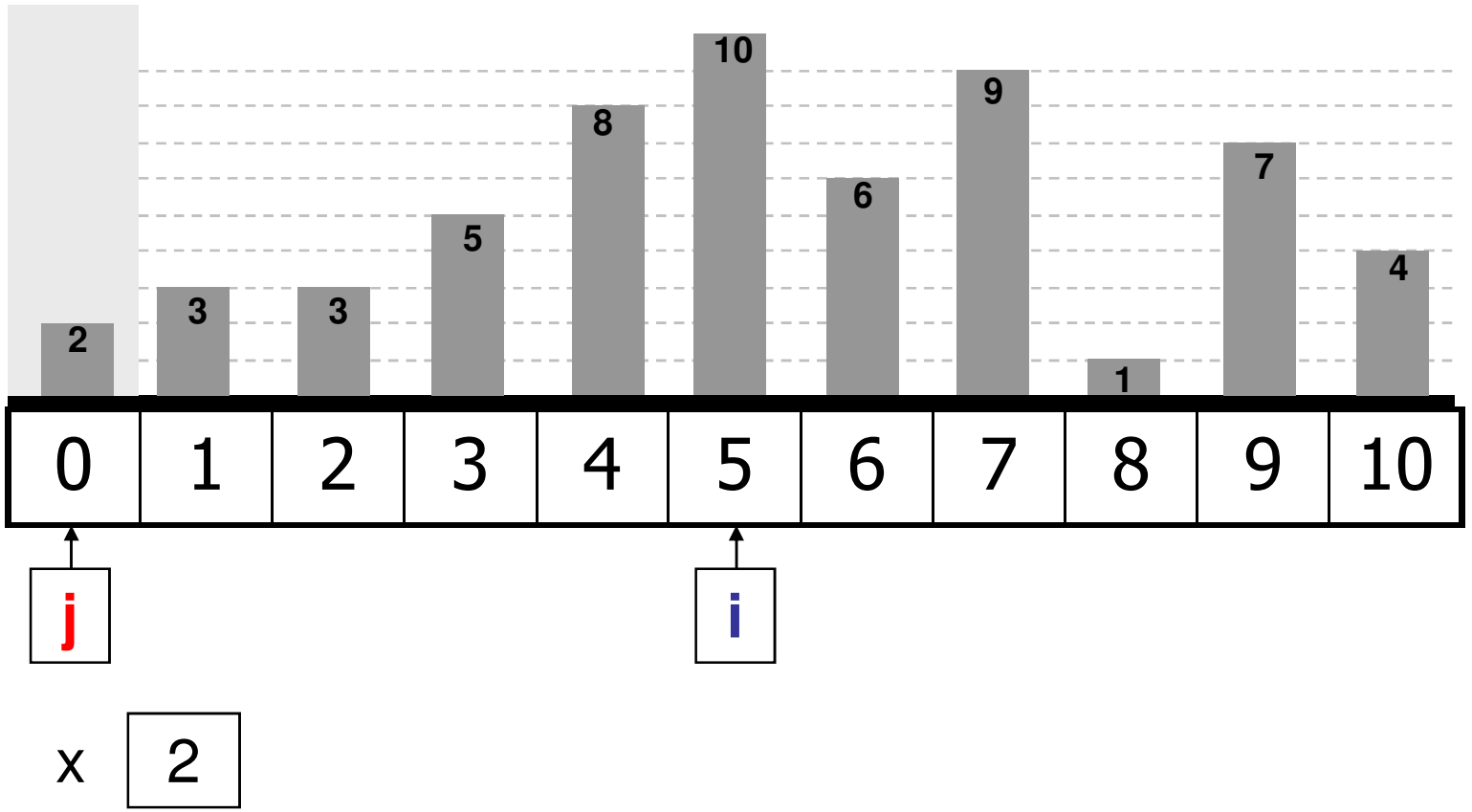
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



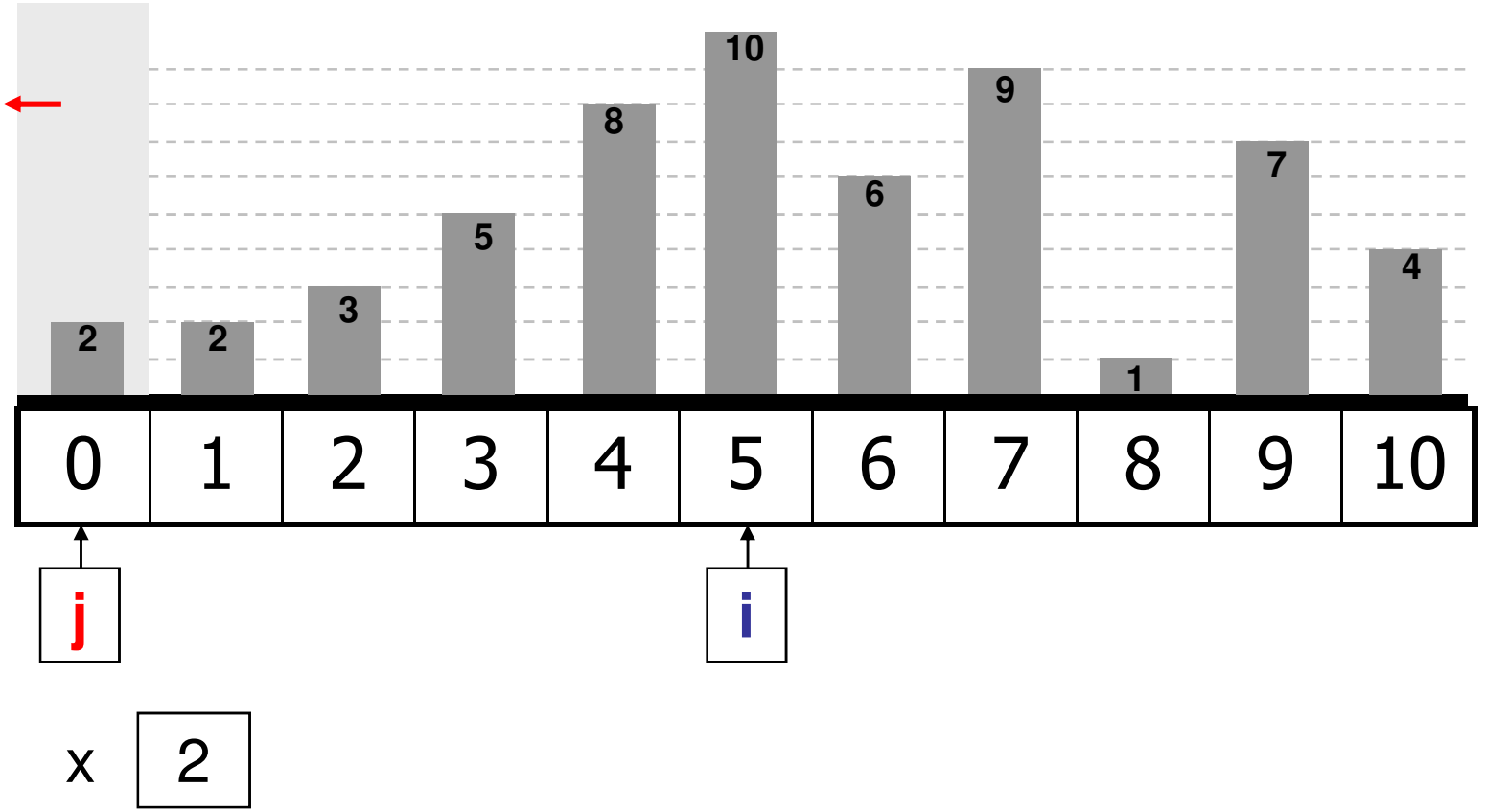
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];  {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



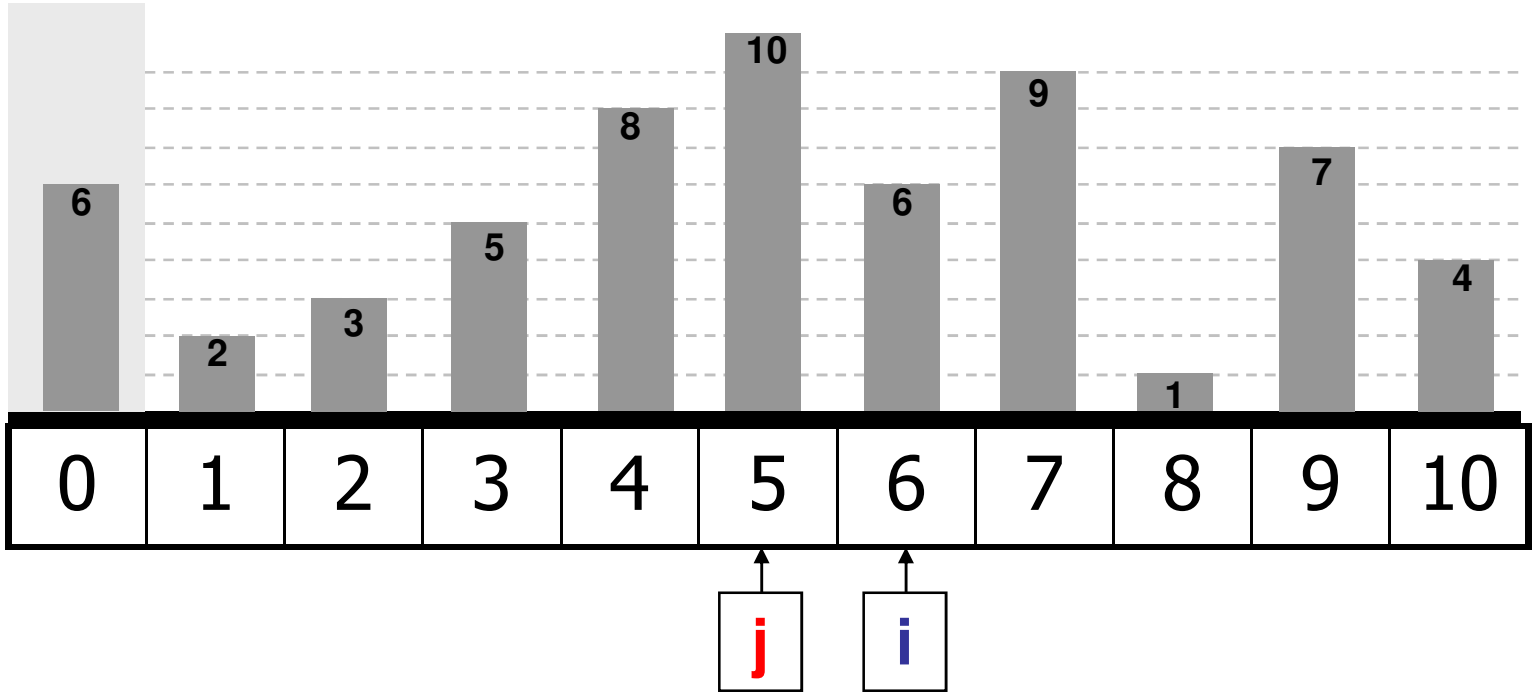
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];  {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

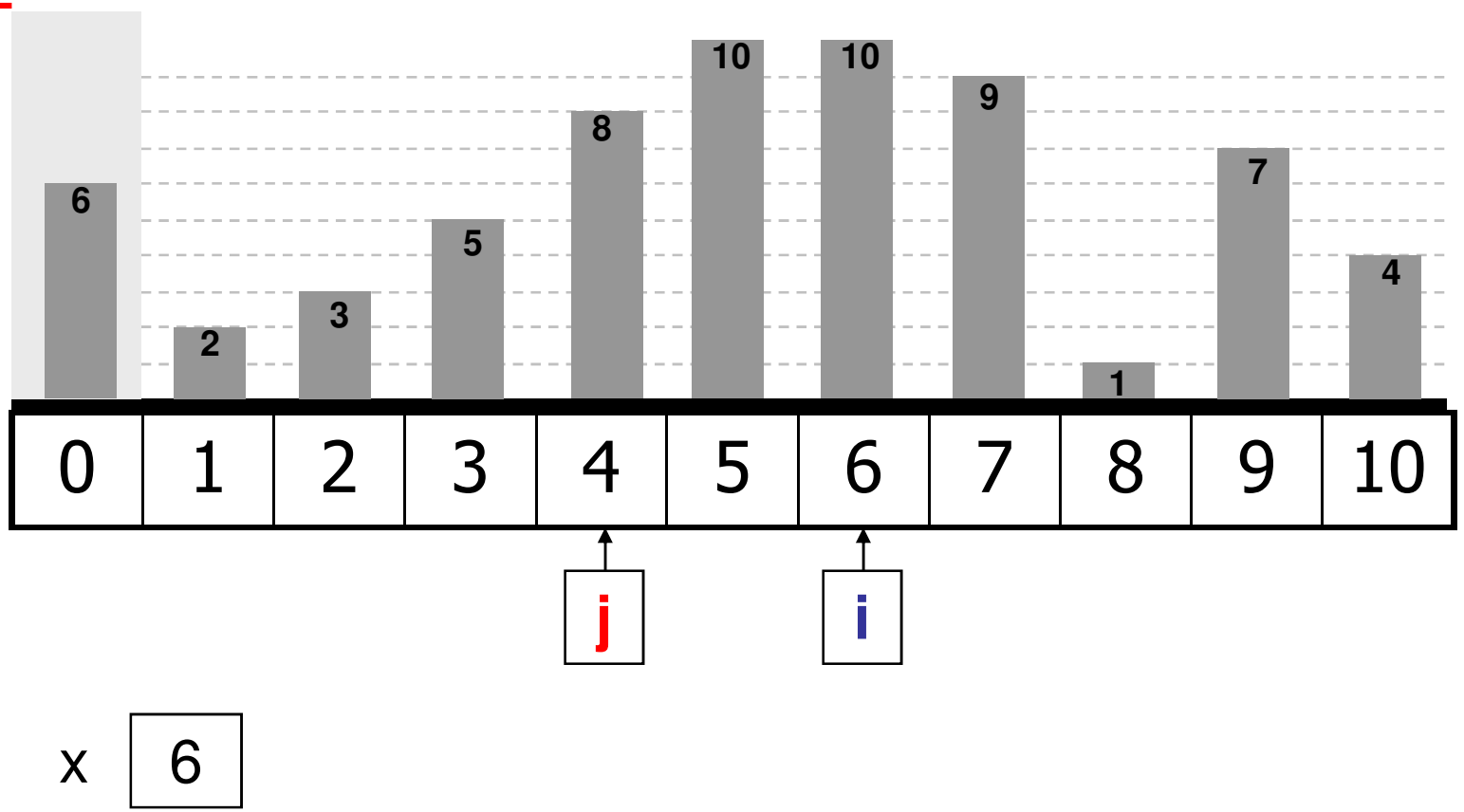


```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x; {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

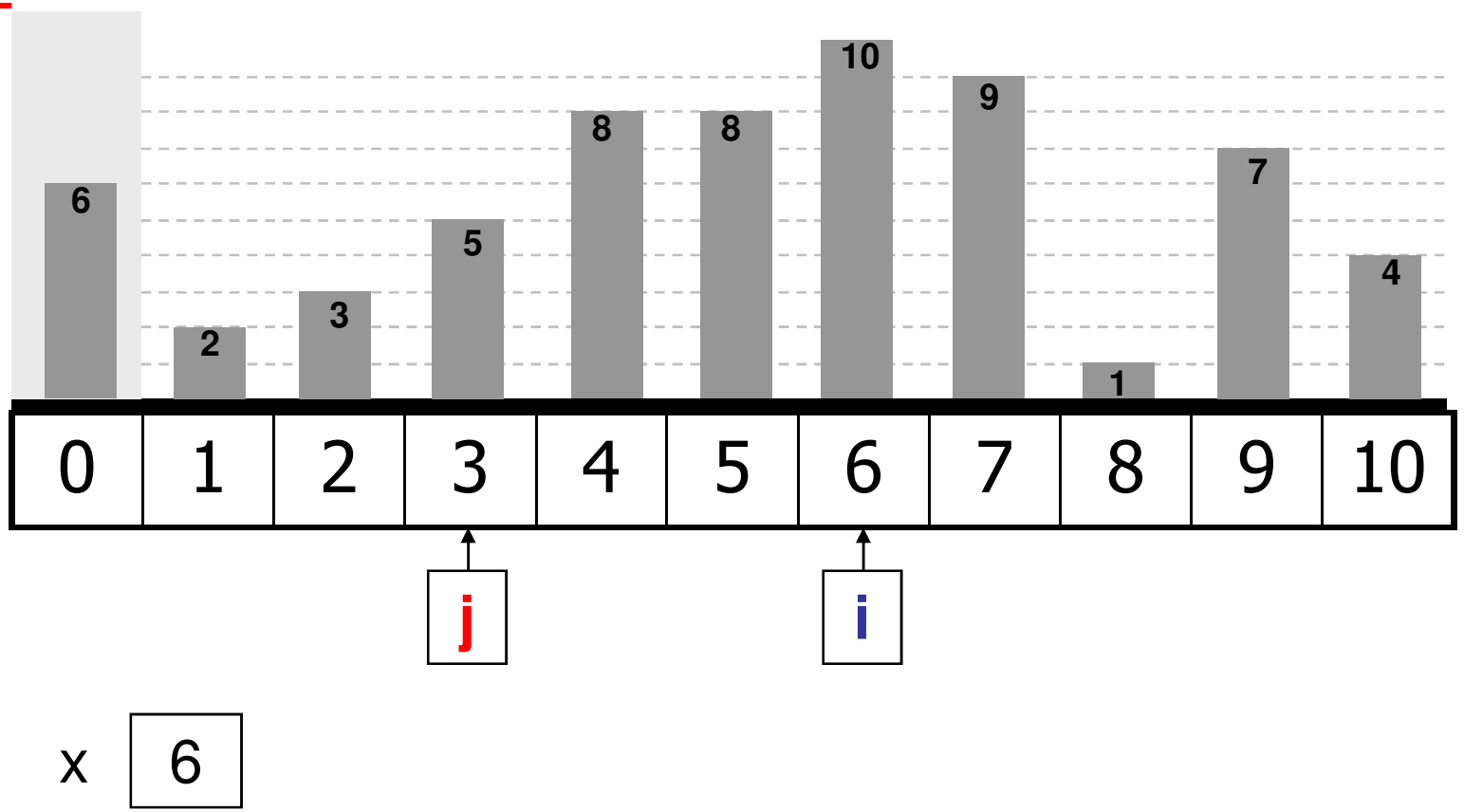


x 6

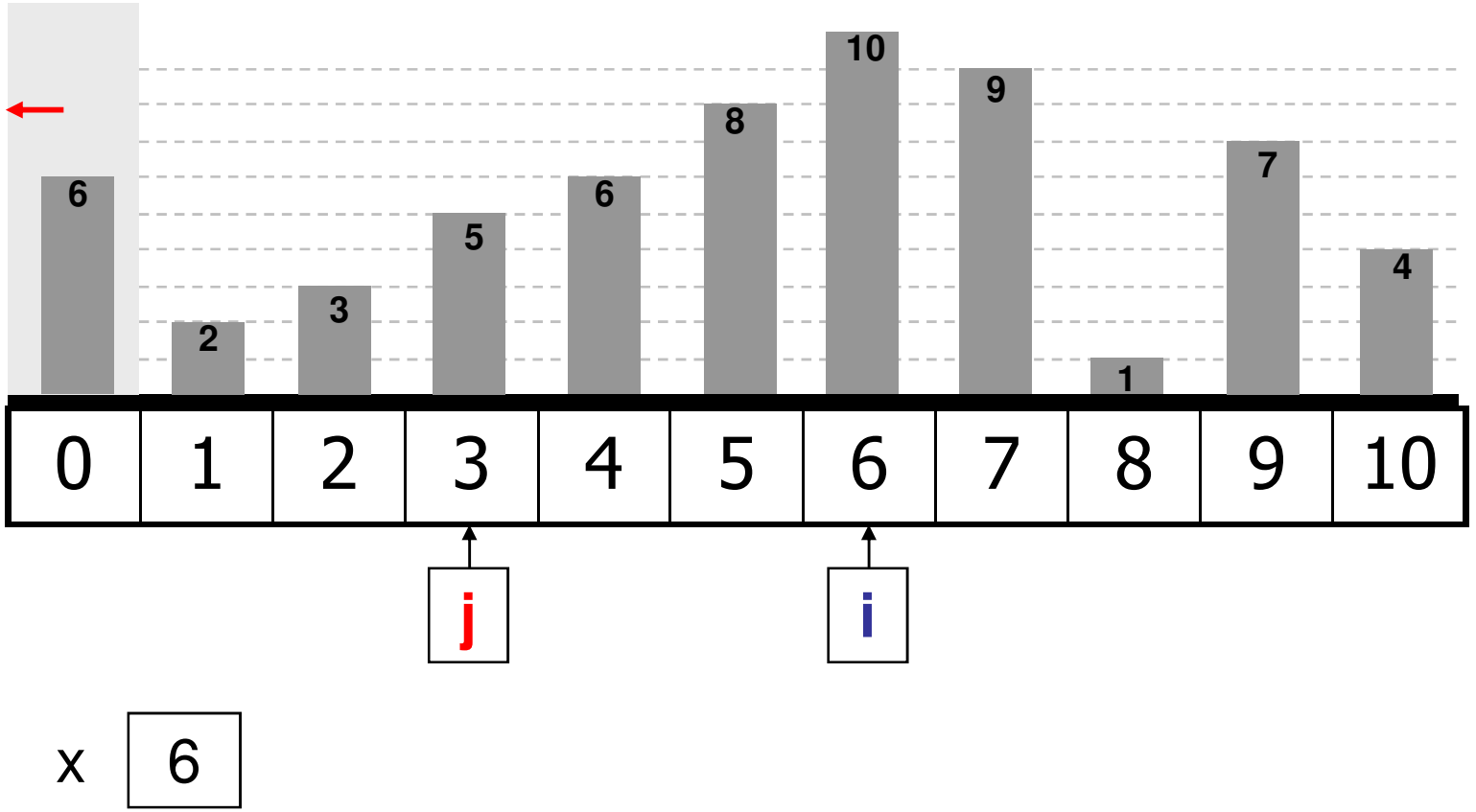
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];  {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



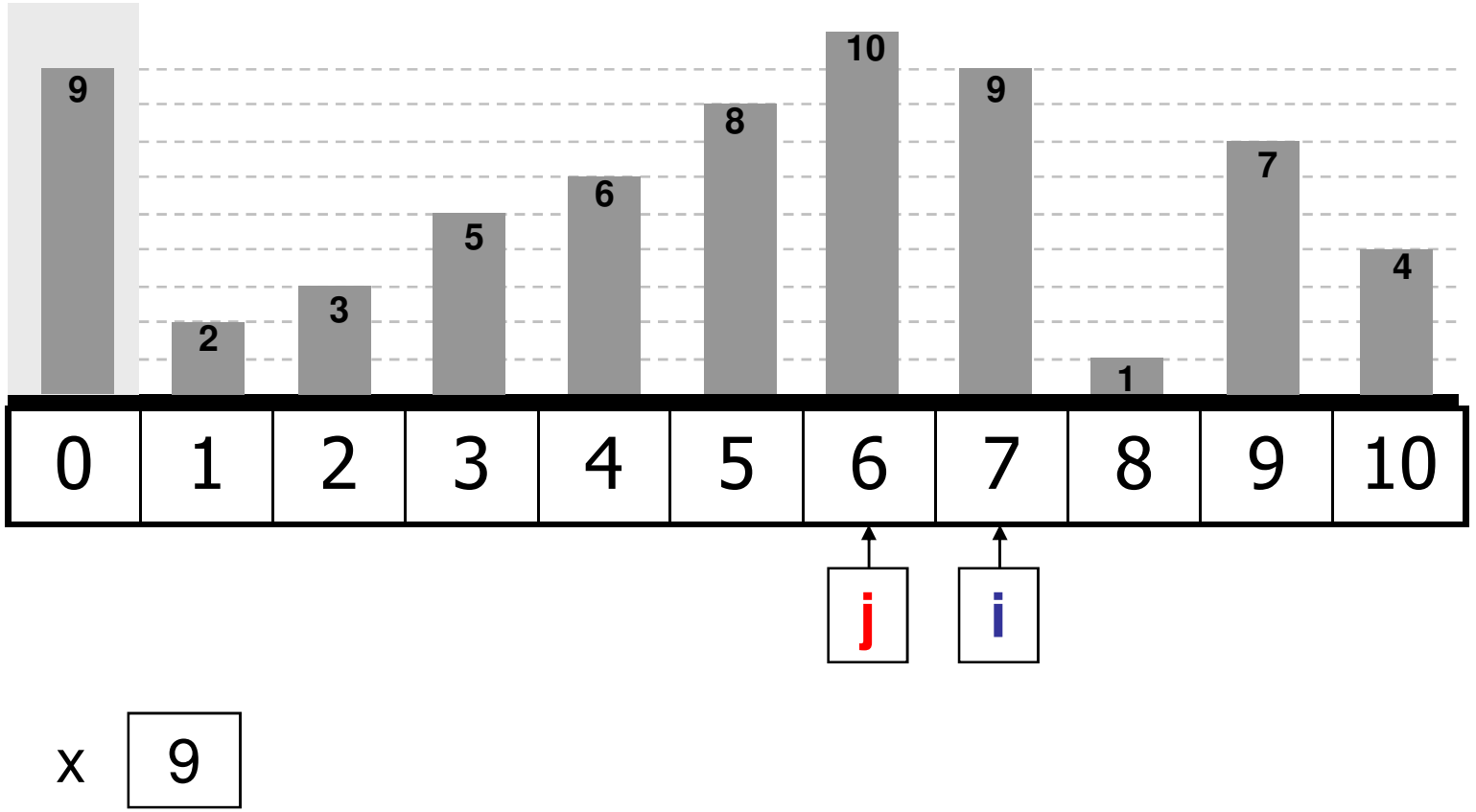
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



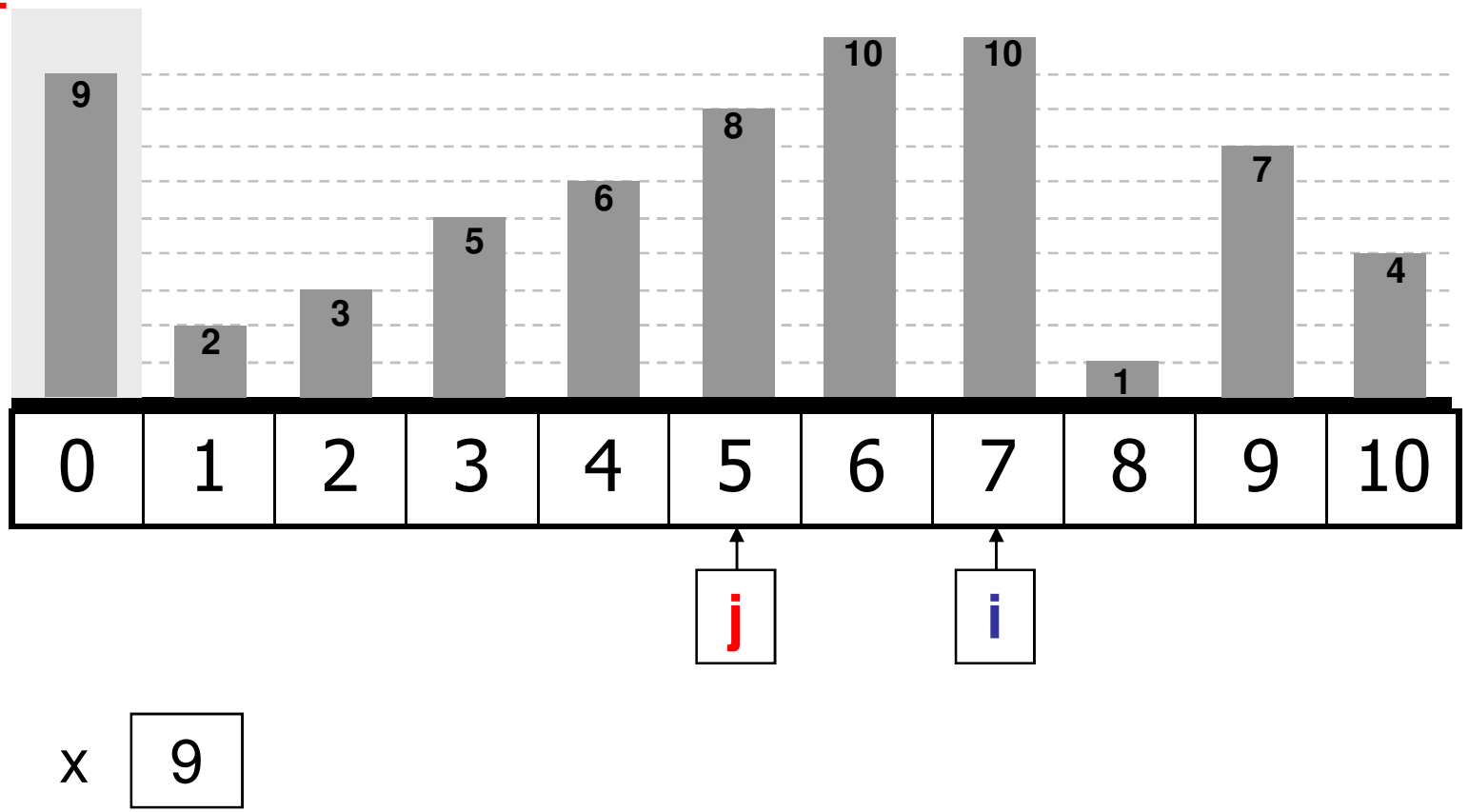
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



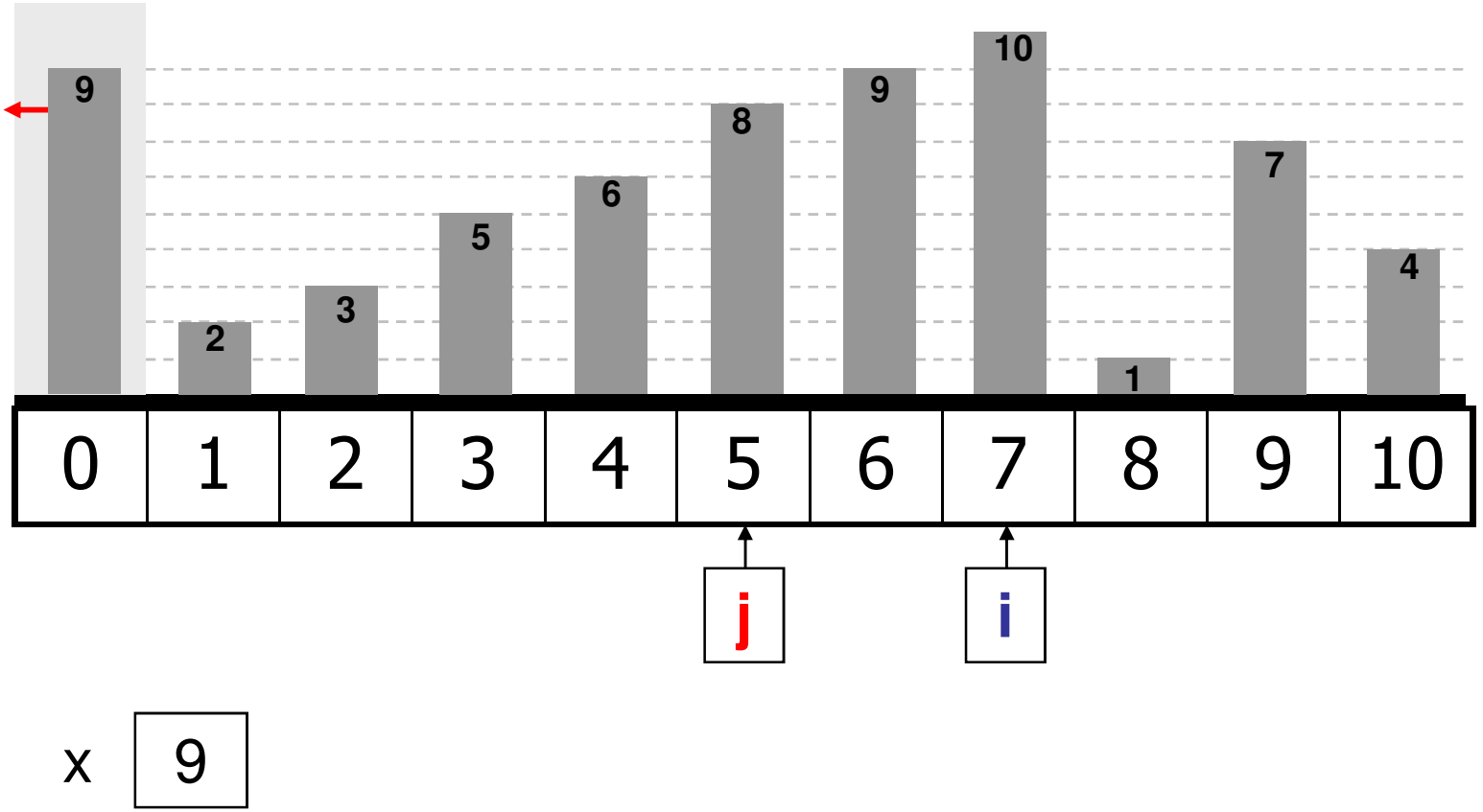
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x; {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



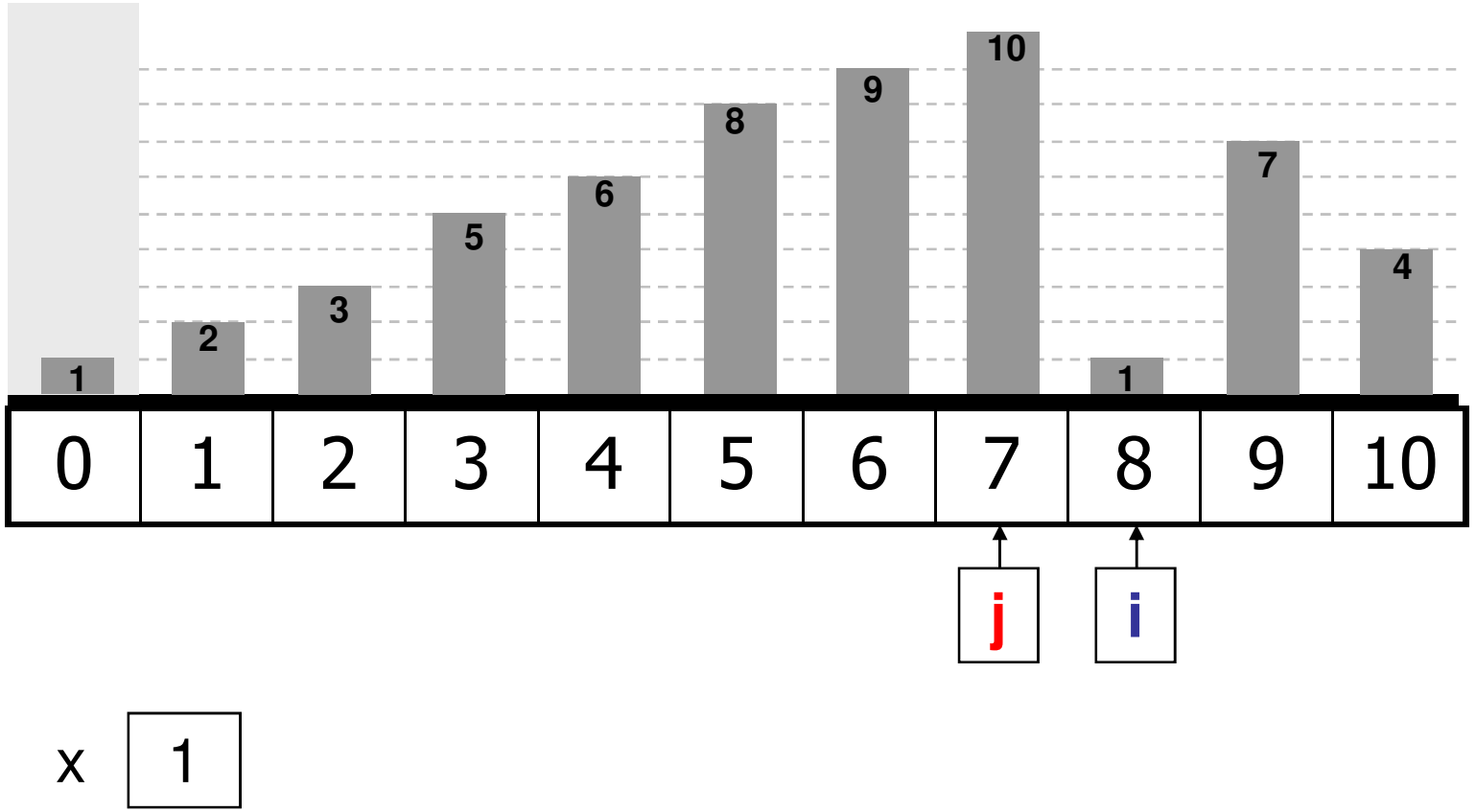
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];  {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

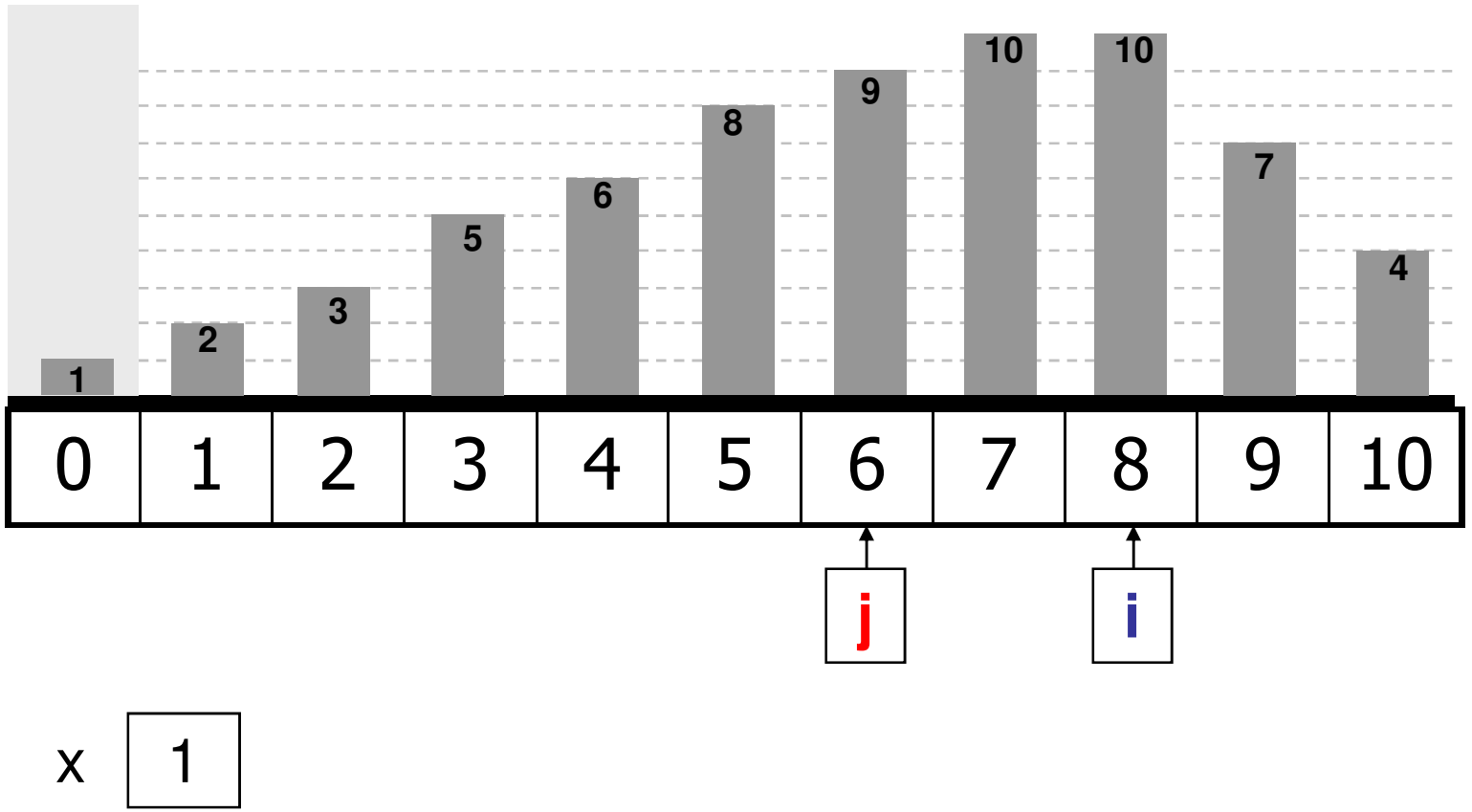


```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x; {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

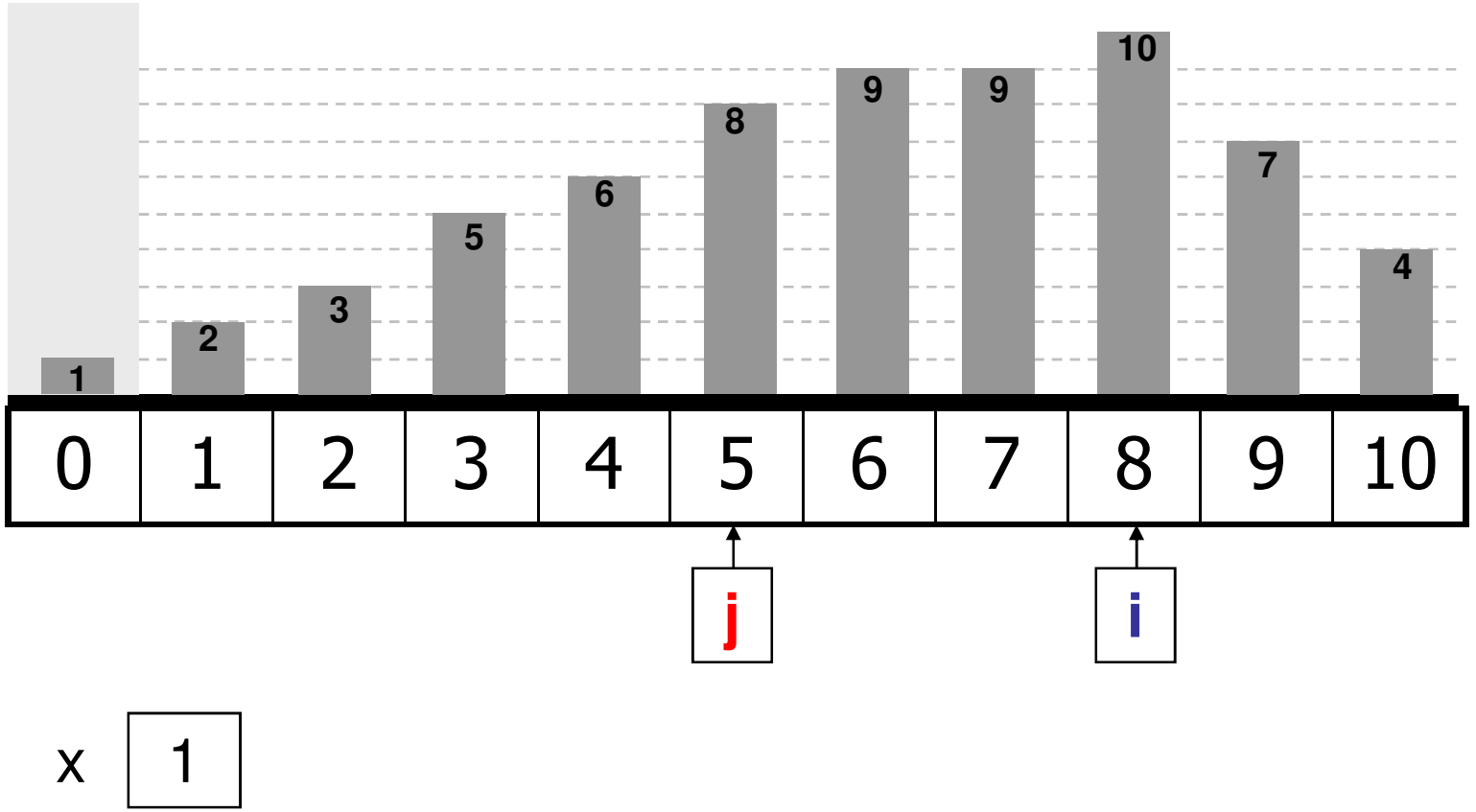


x 1

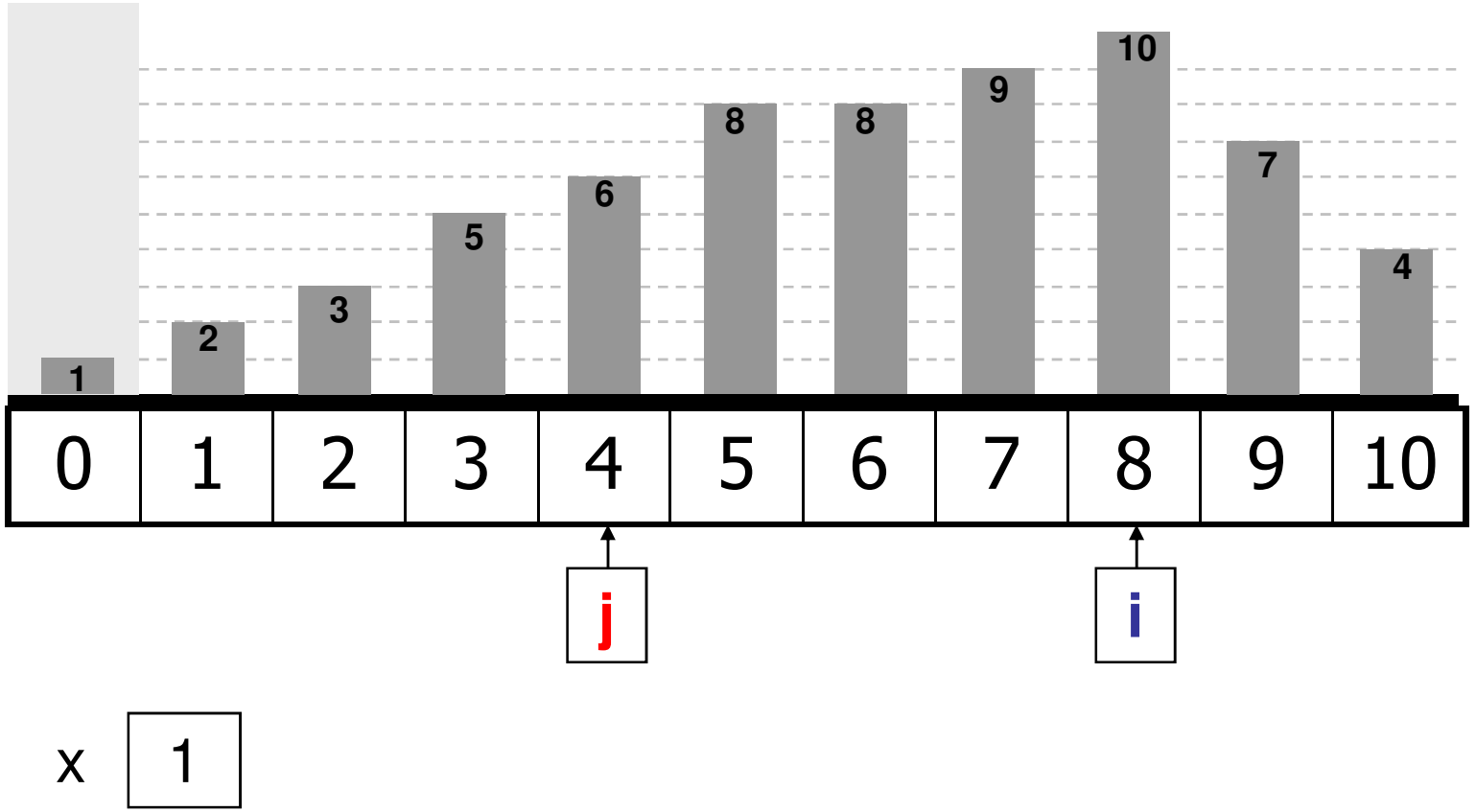
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



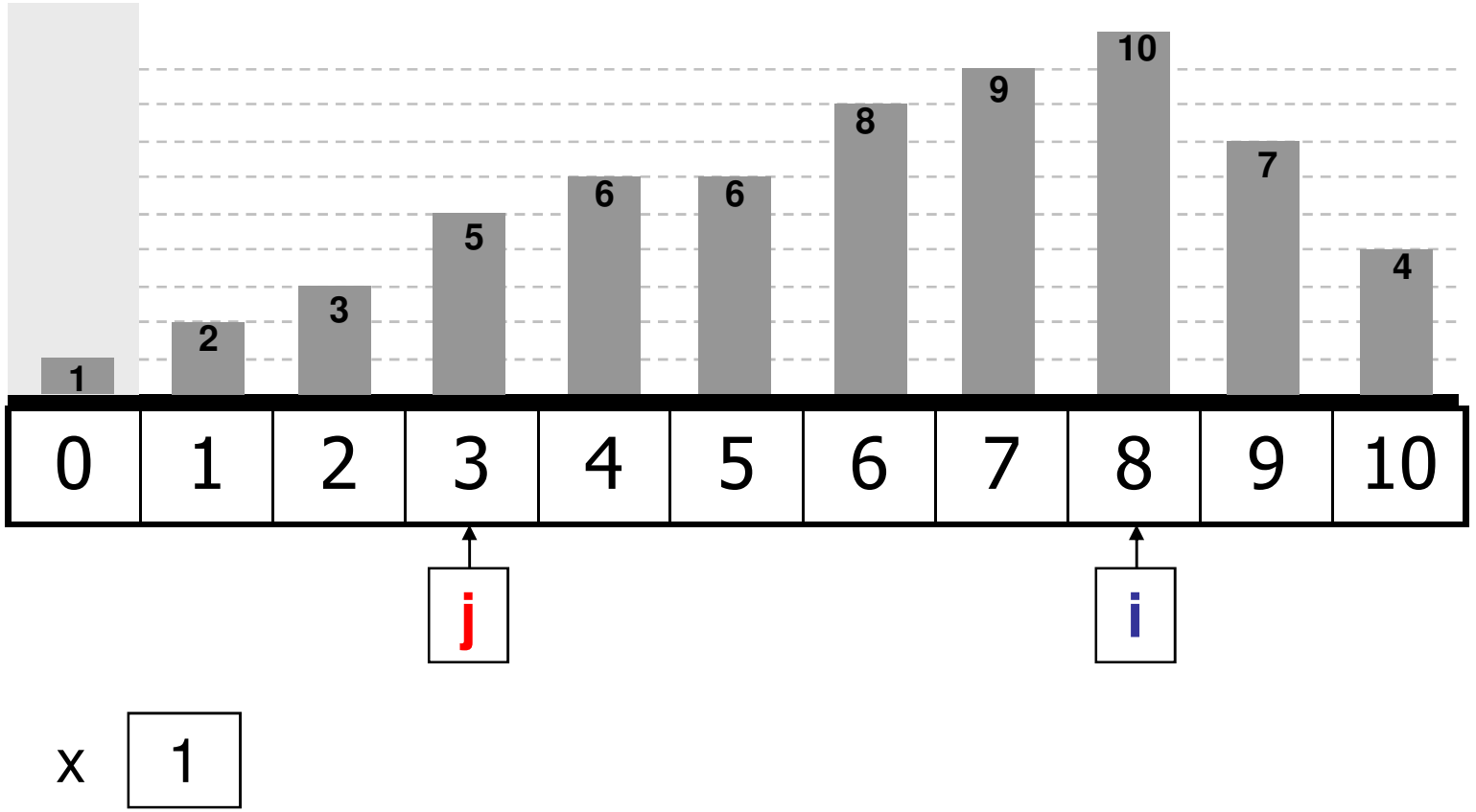
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



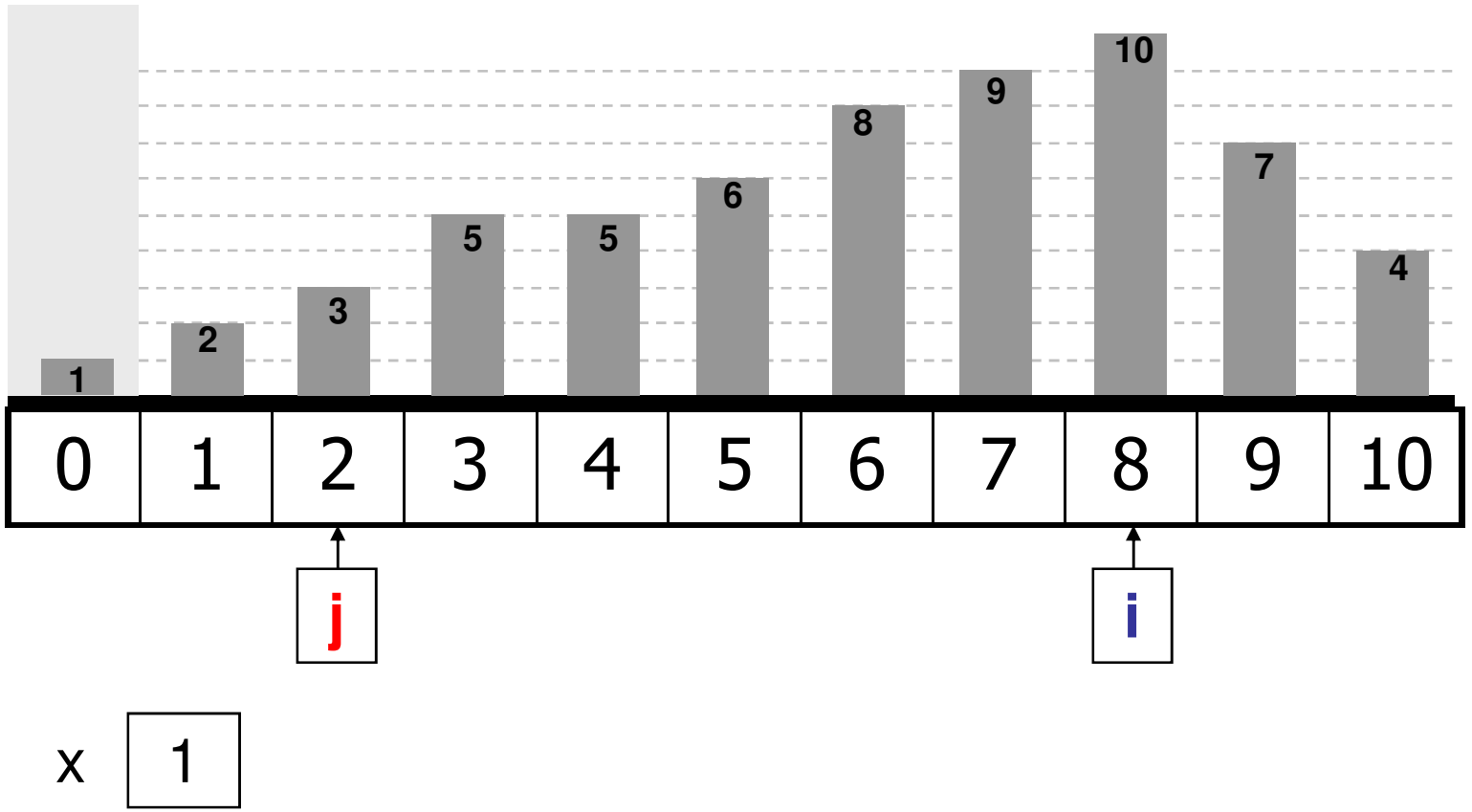
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



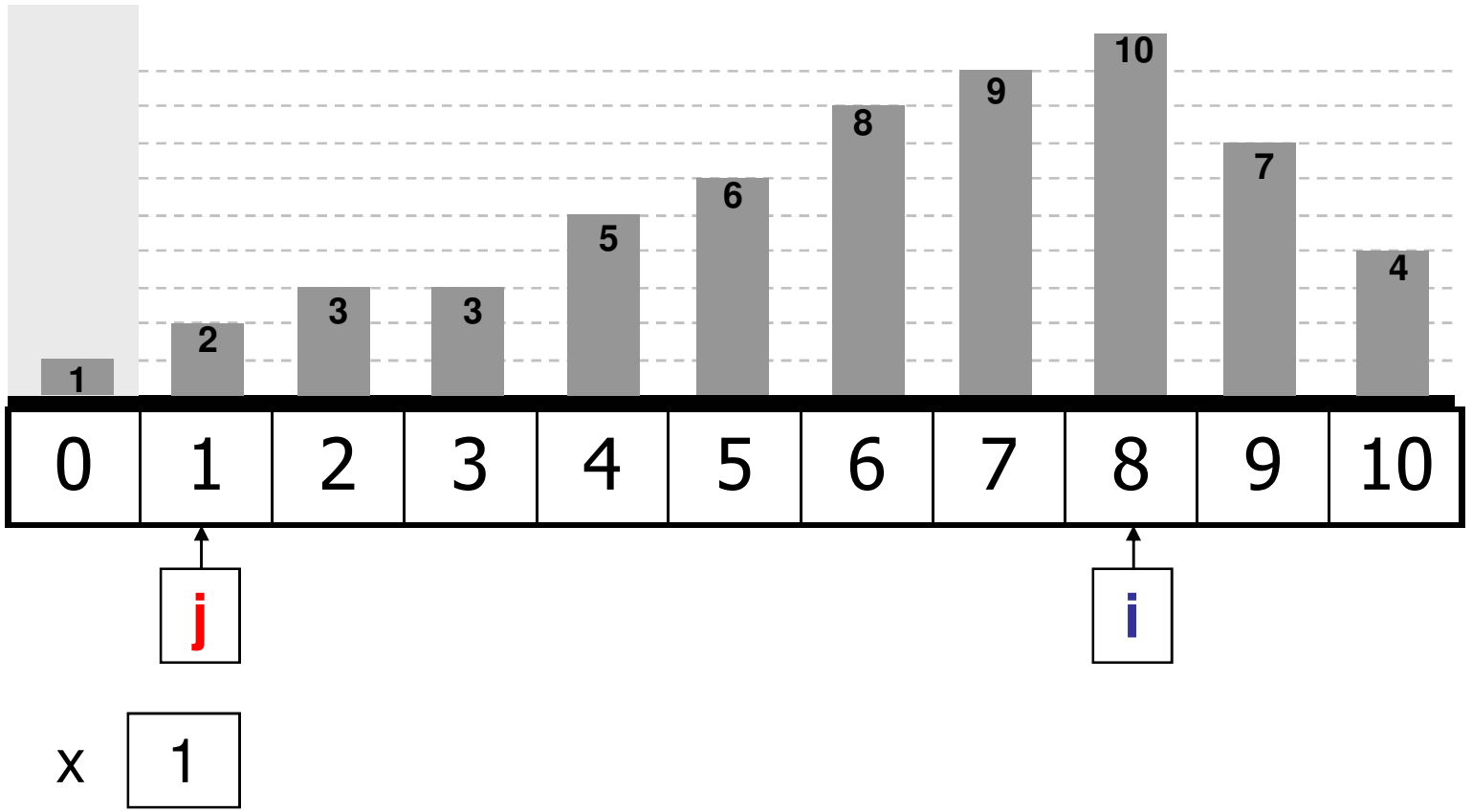
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



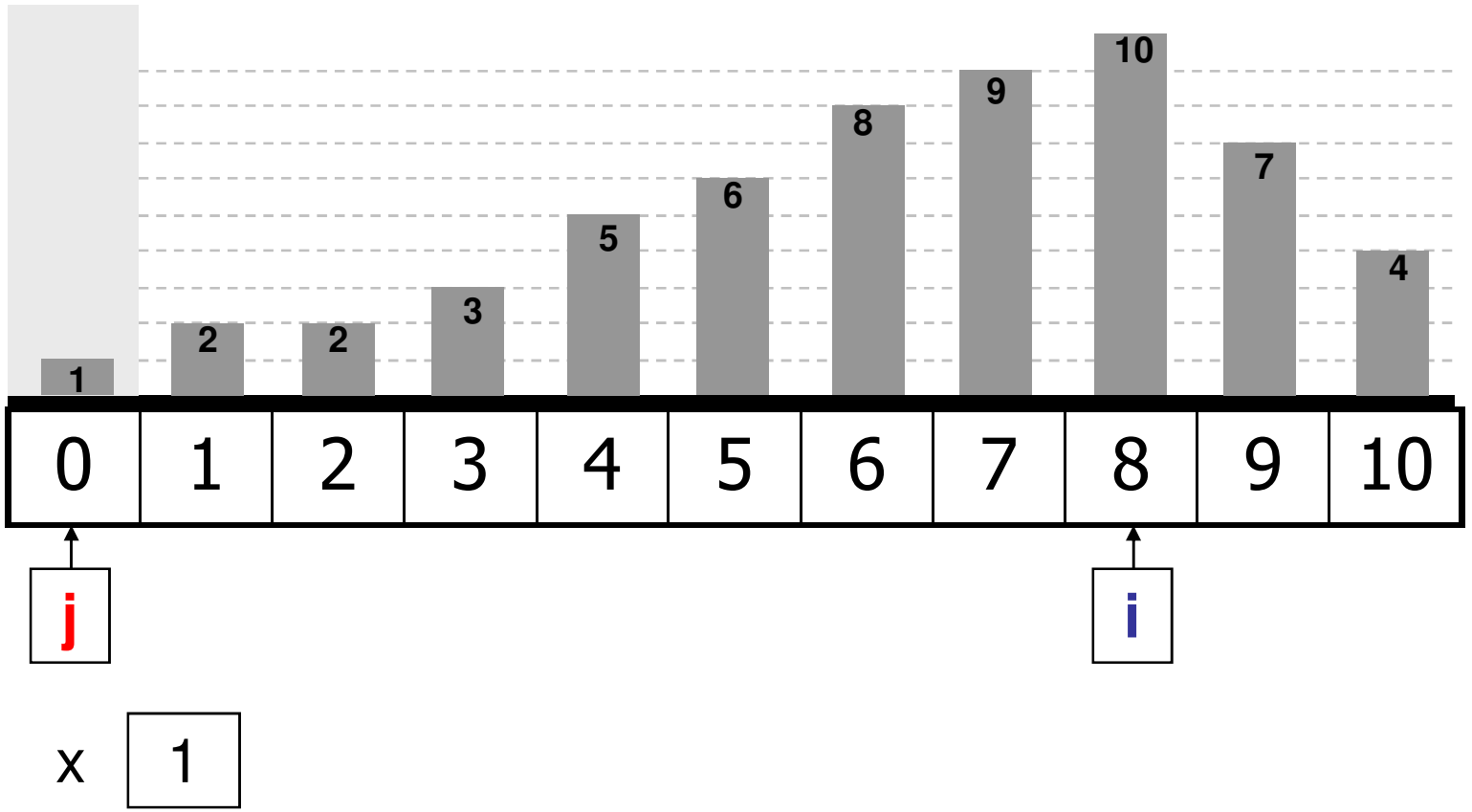
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



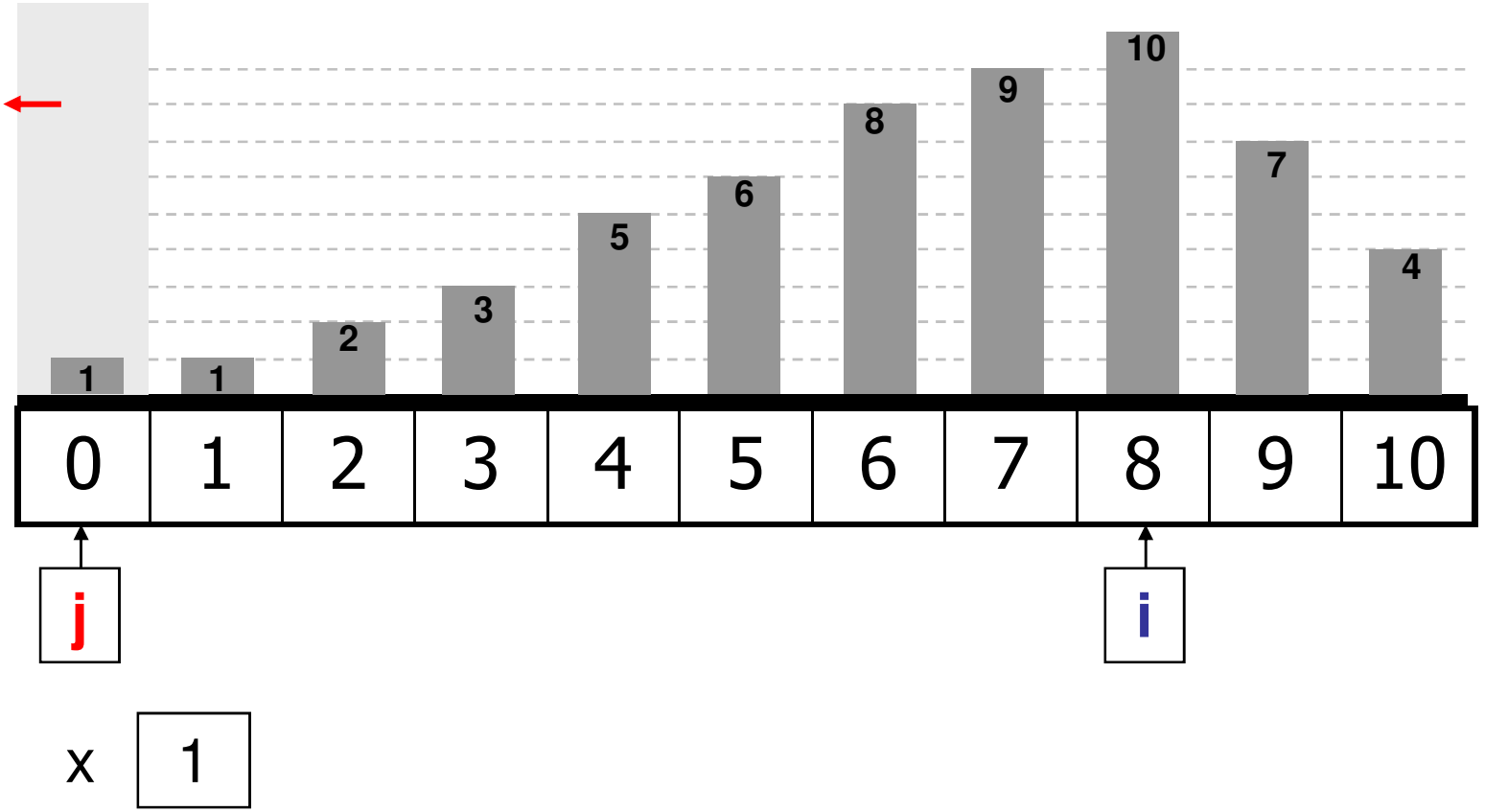
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



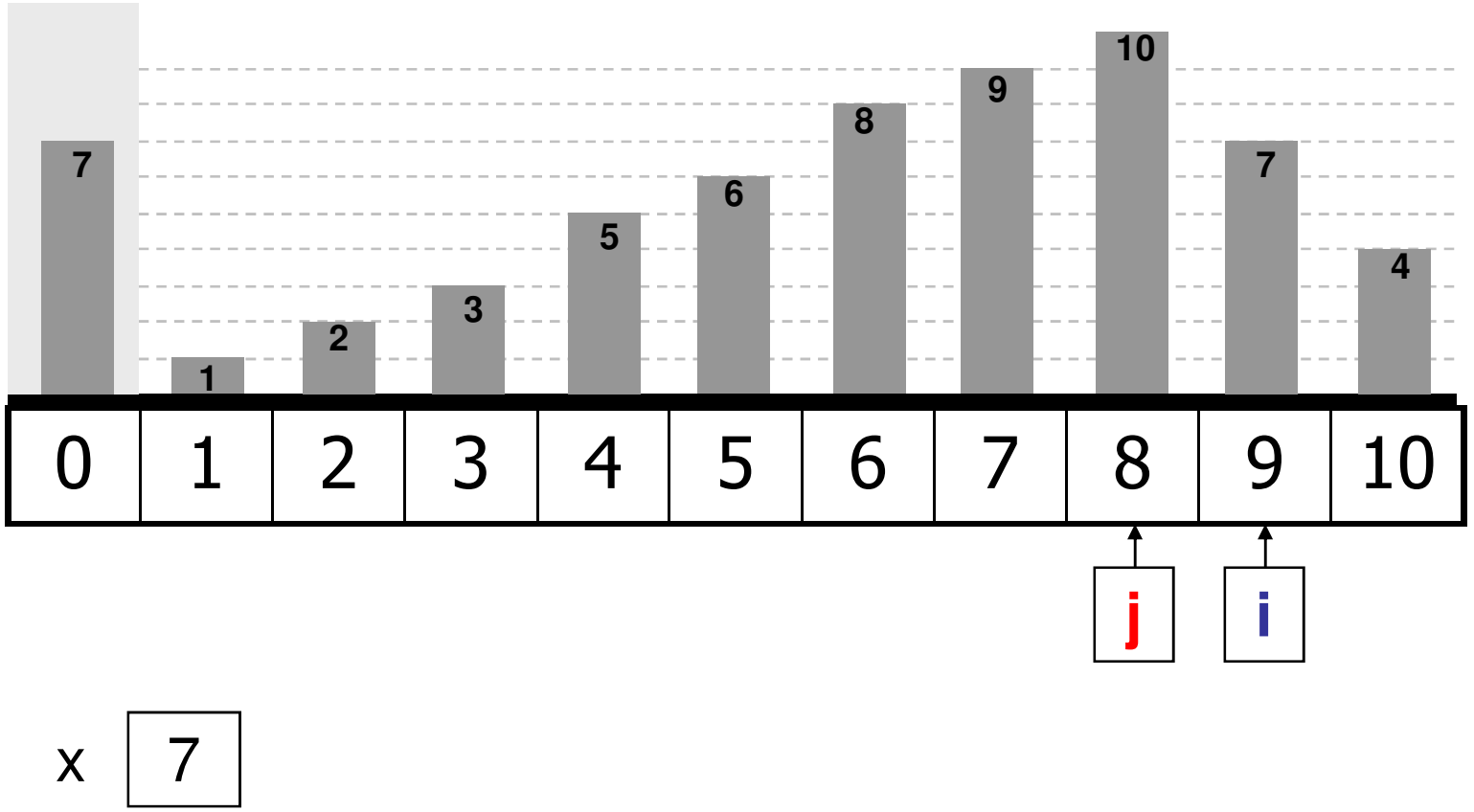
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



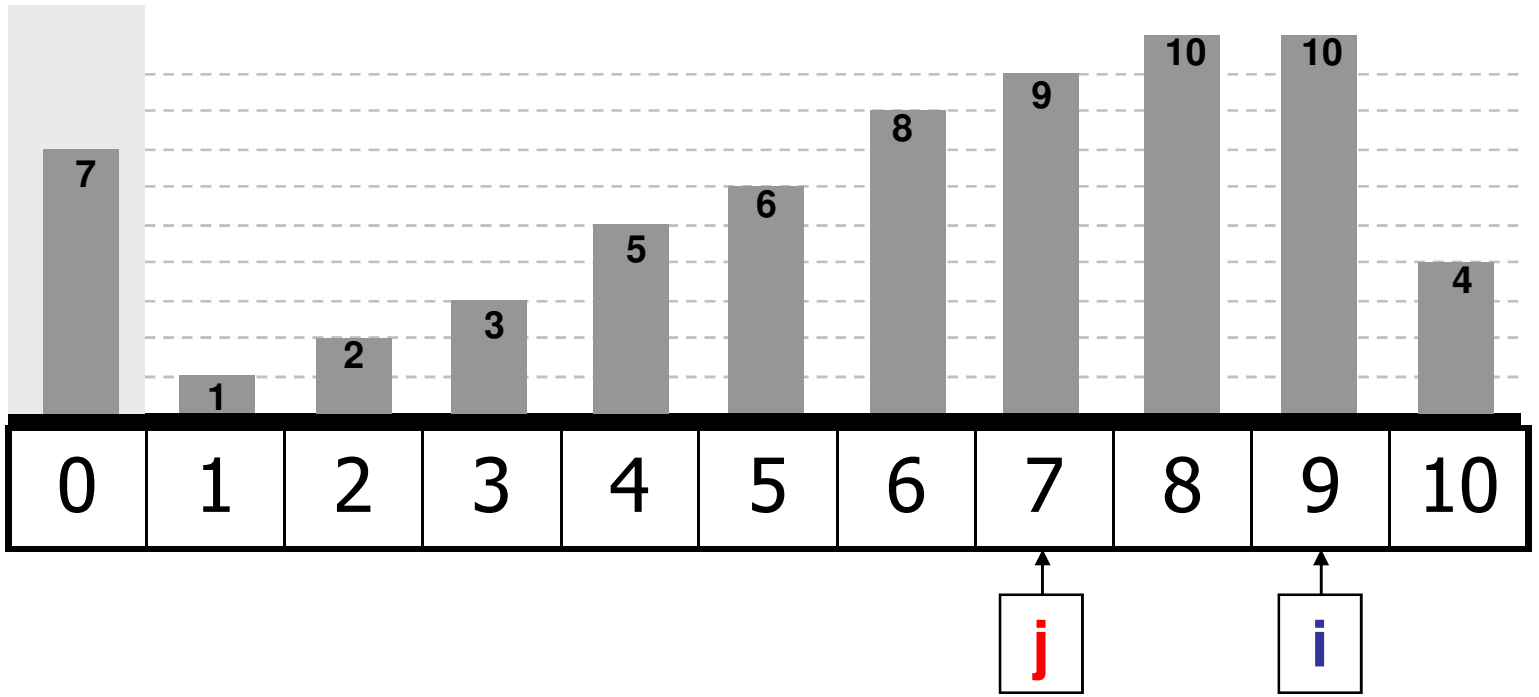
```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x; {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

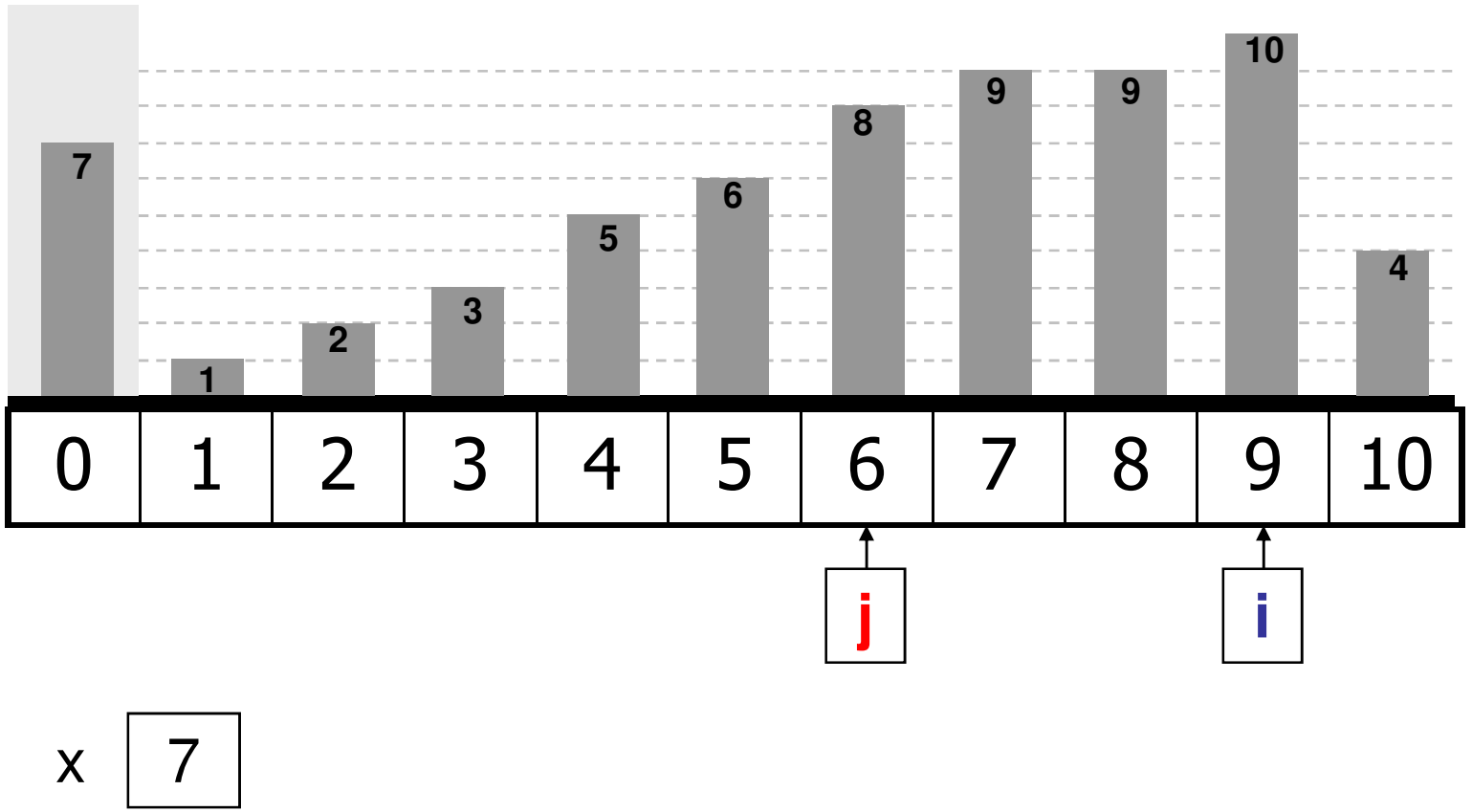


```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

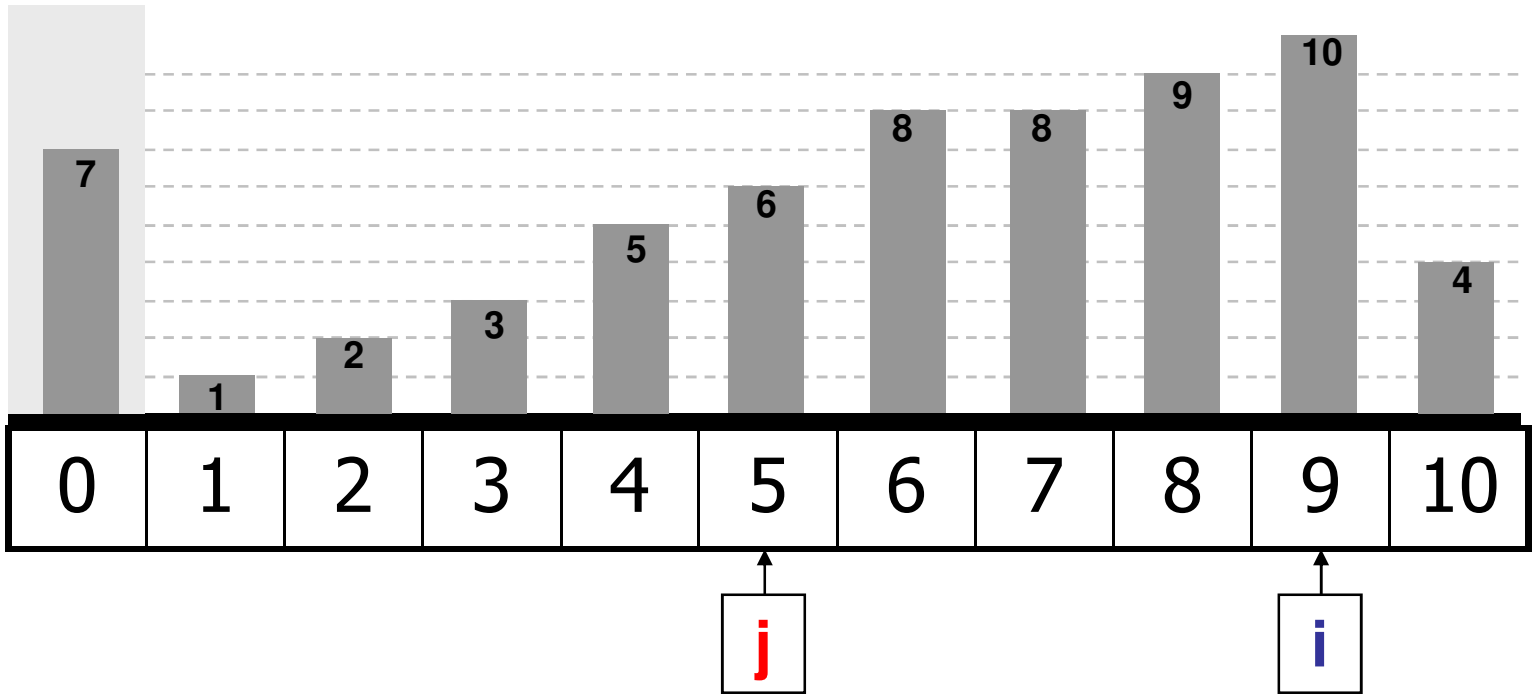


x 7

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```

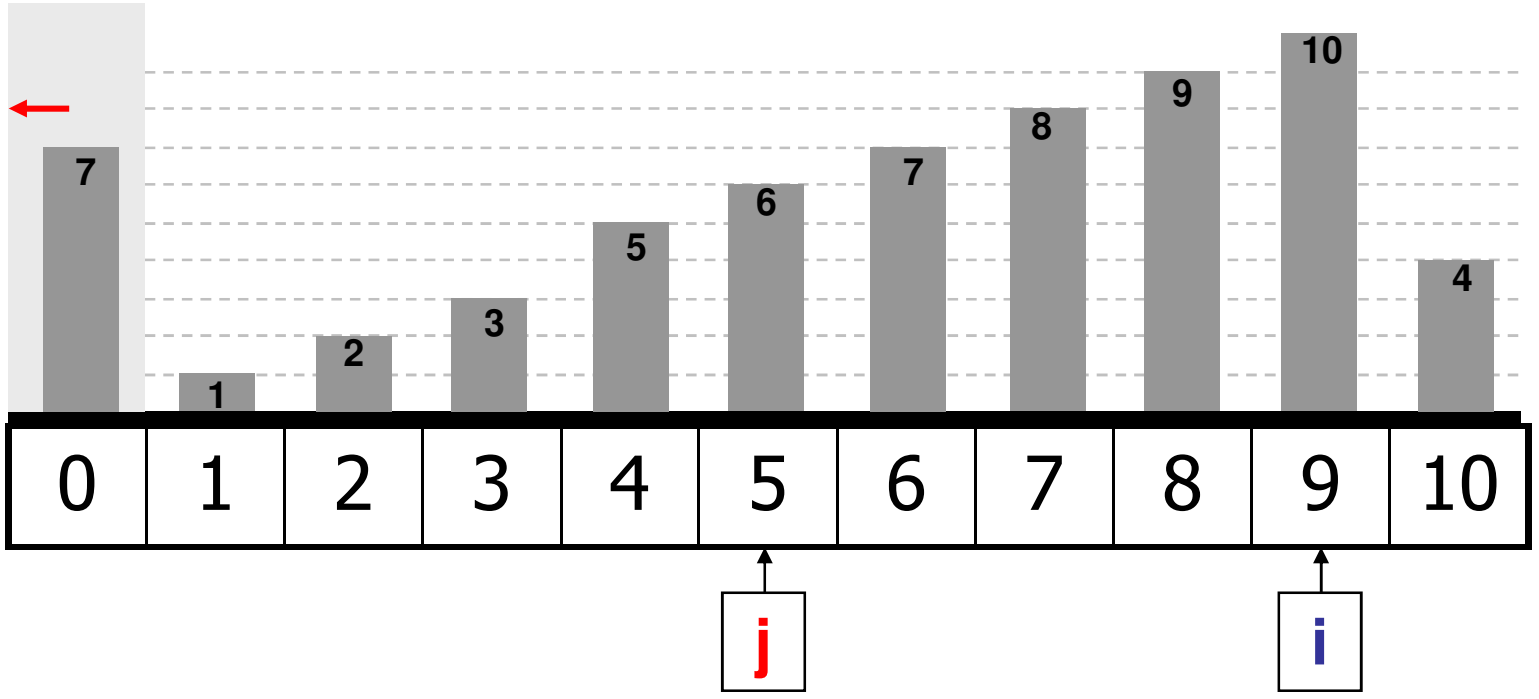


```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



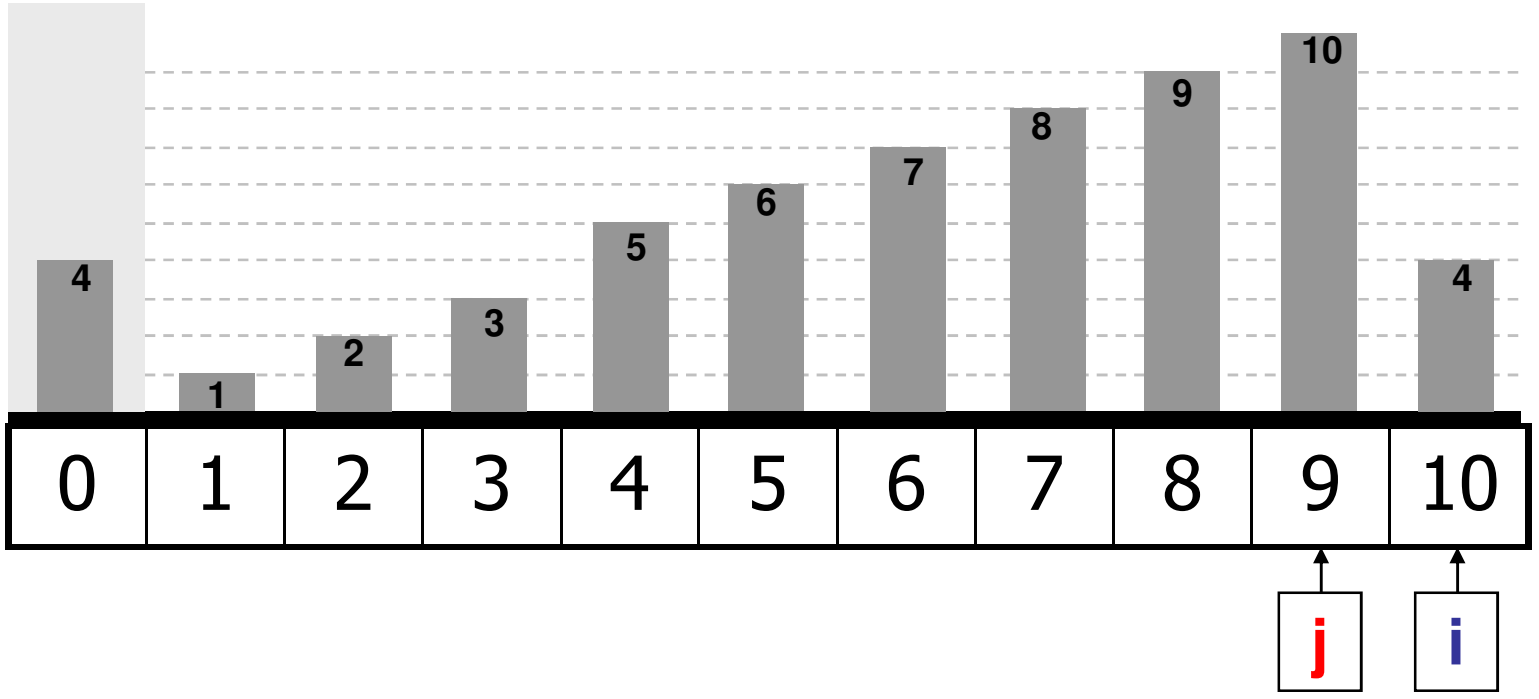
x 7

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



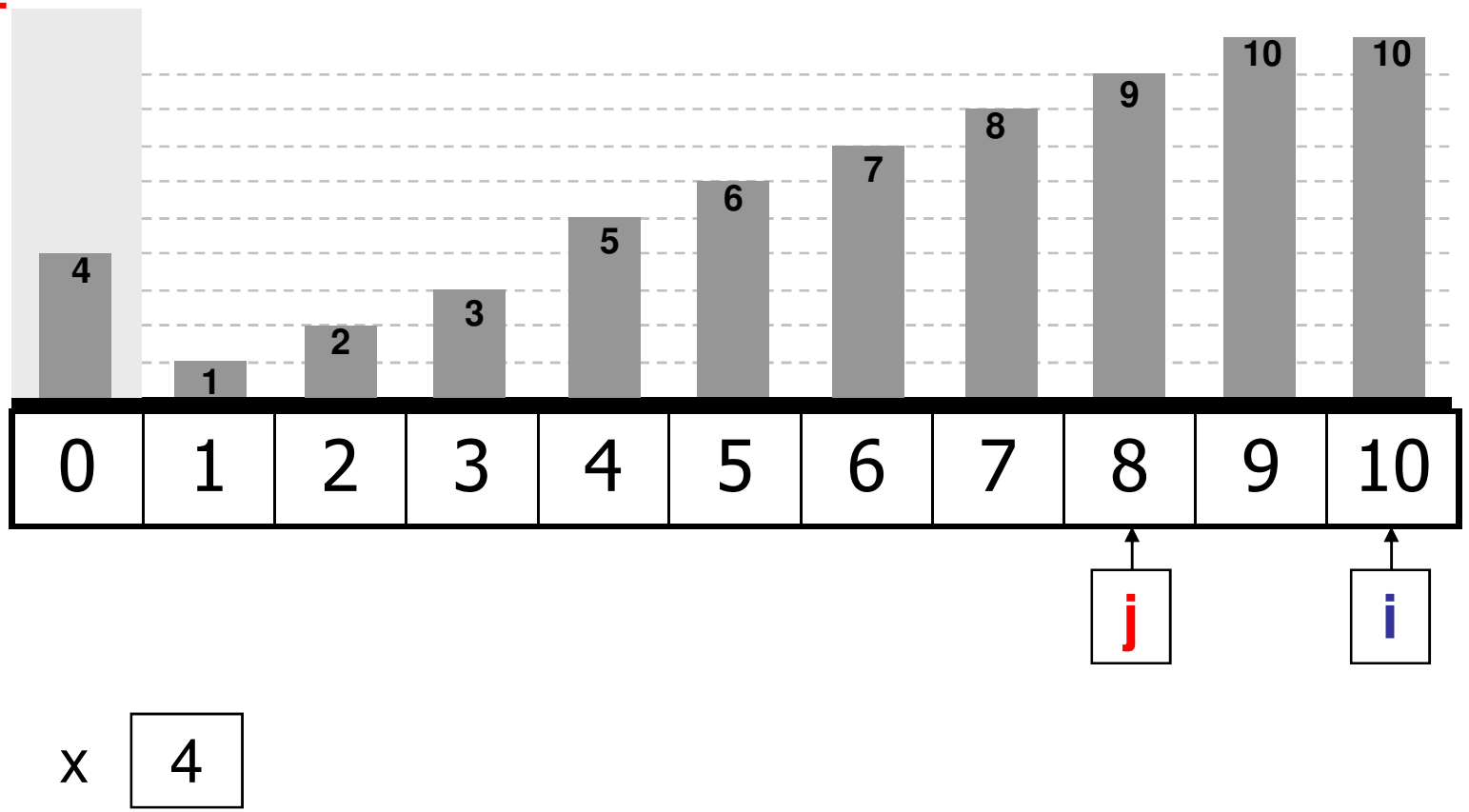
x 7

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x; {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



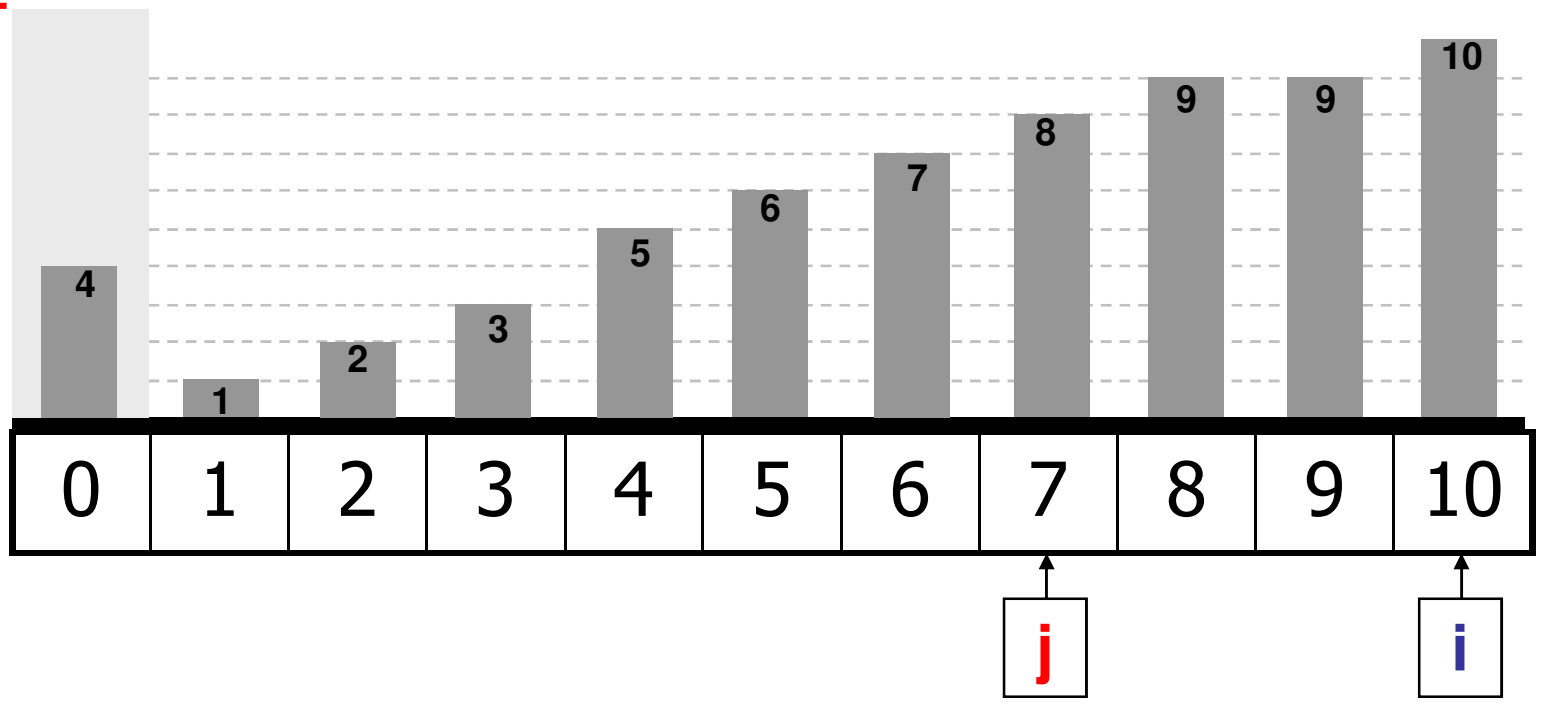
x | 4

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



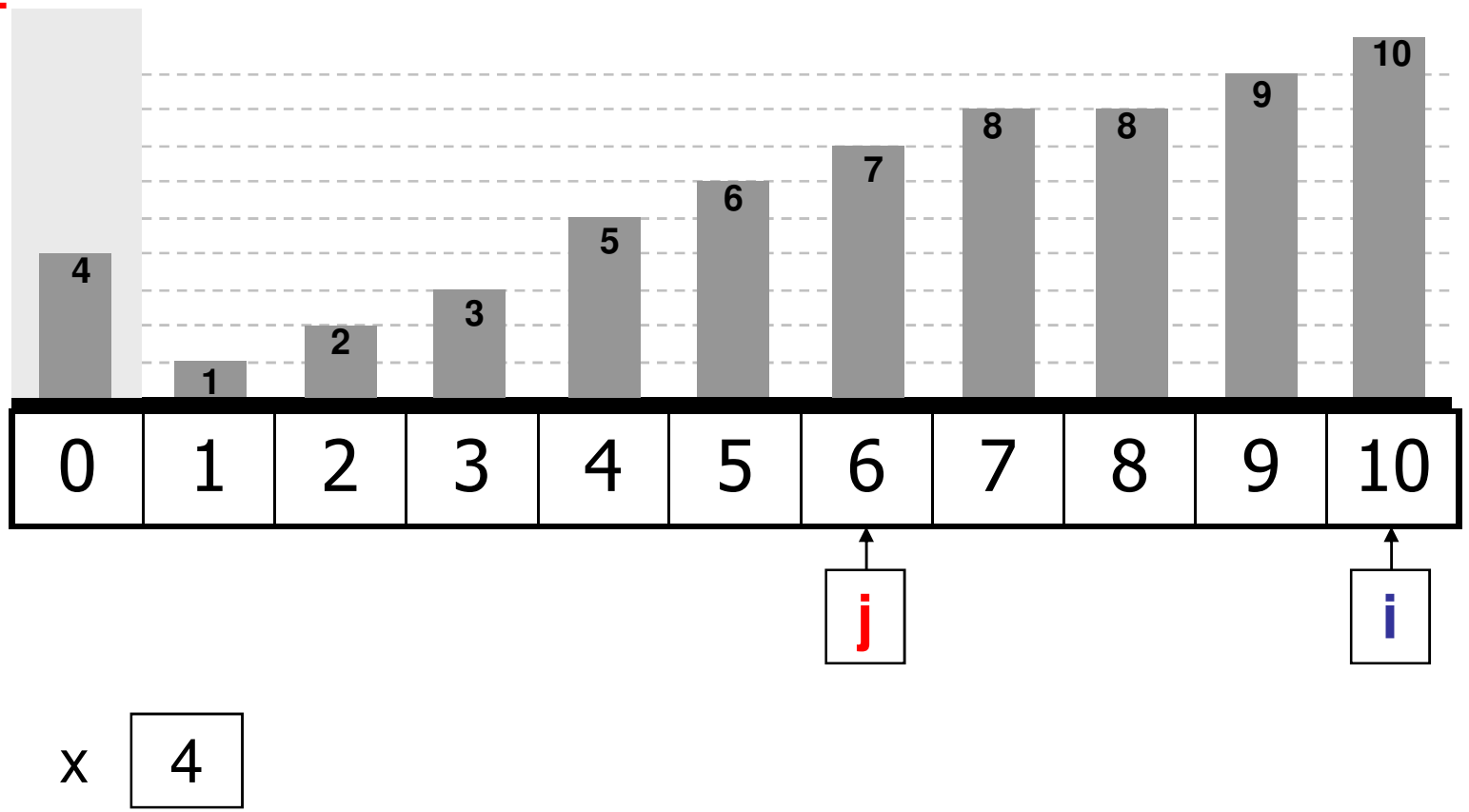
x 4

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



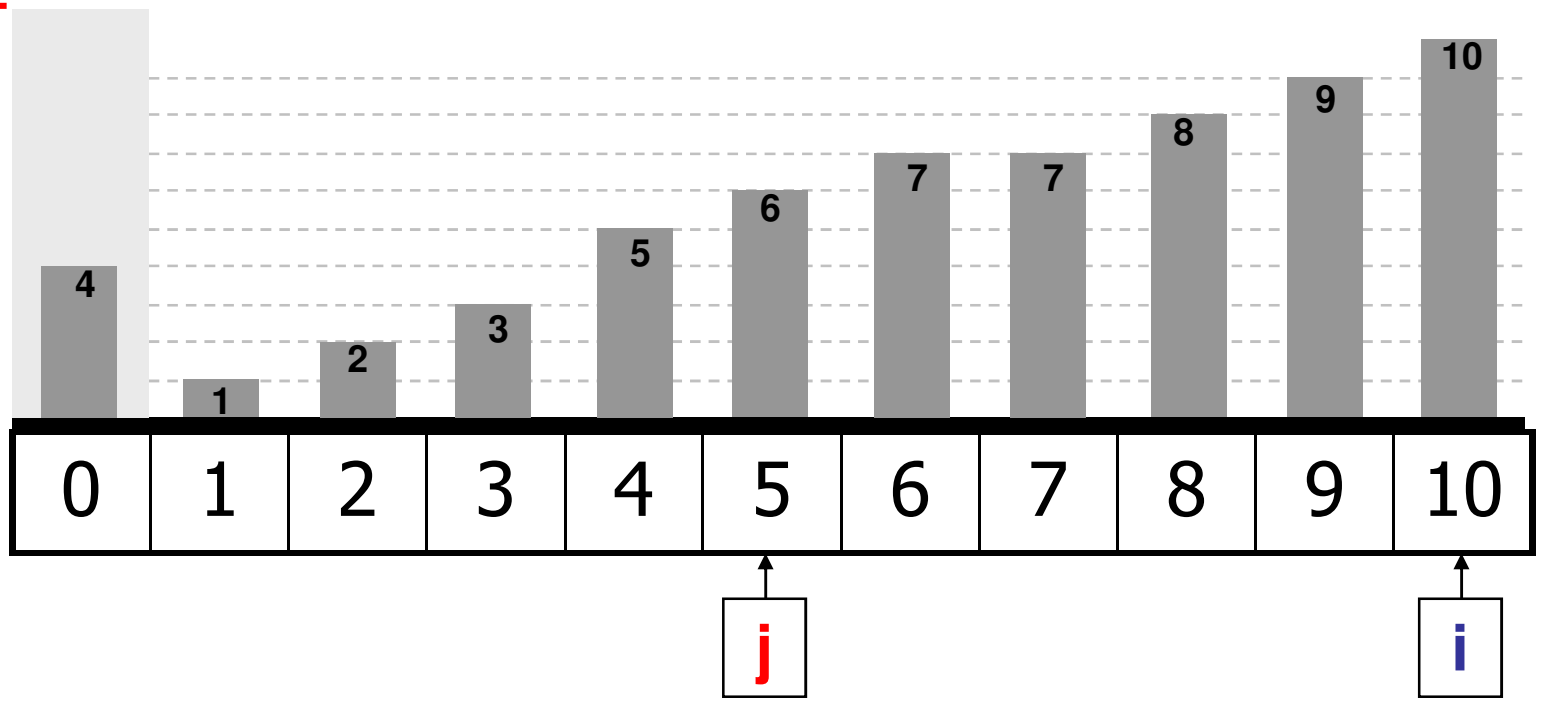
x | 4

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



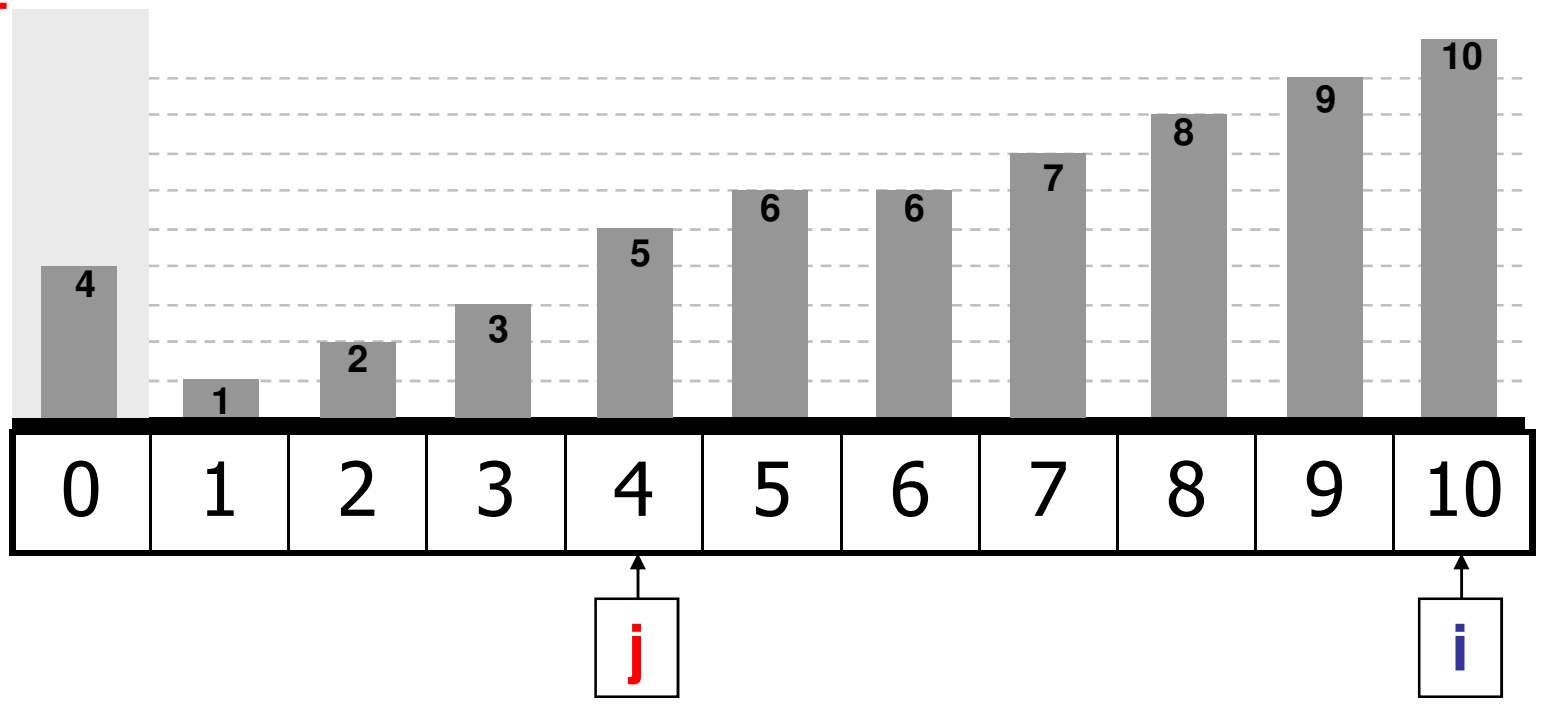
x | 4

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



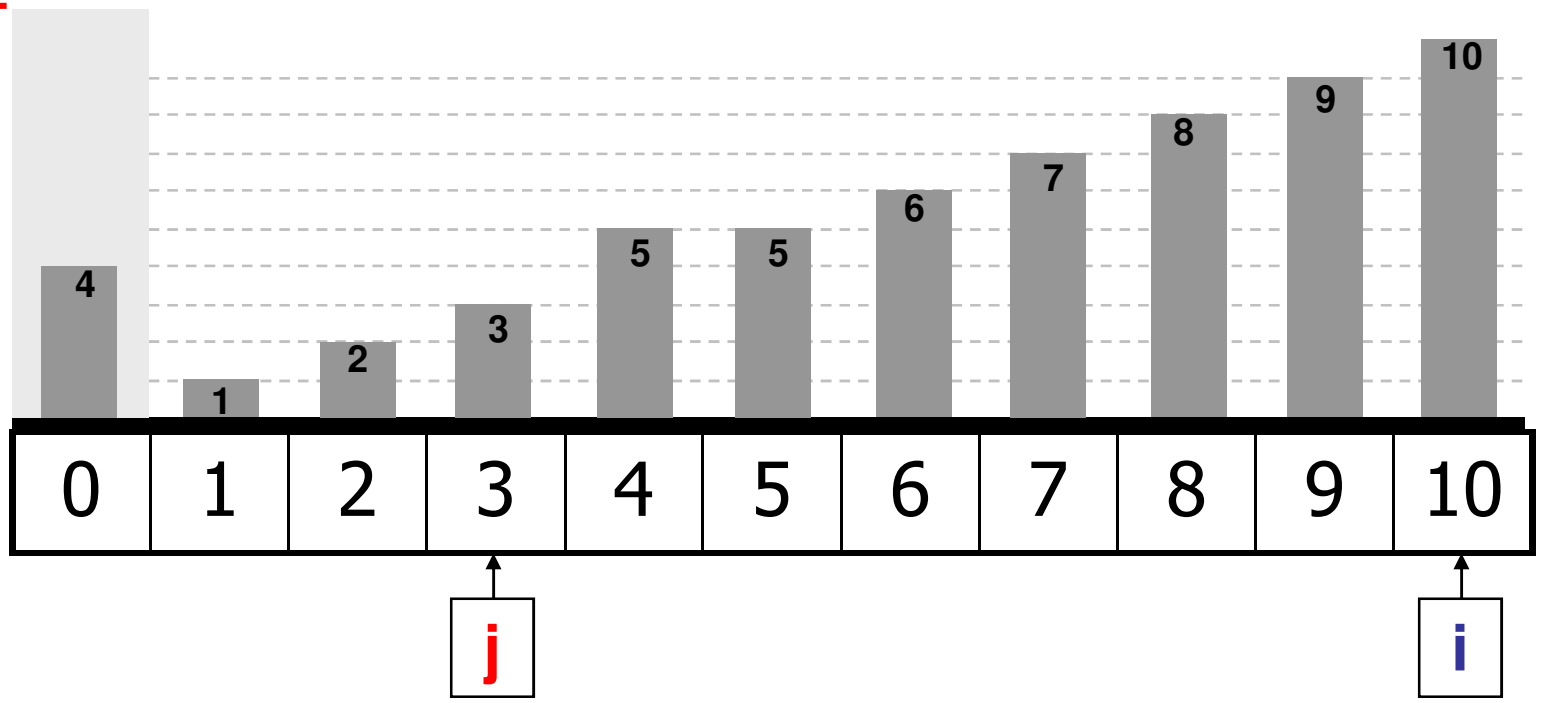
x 4

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



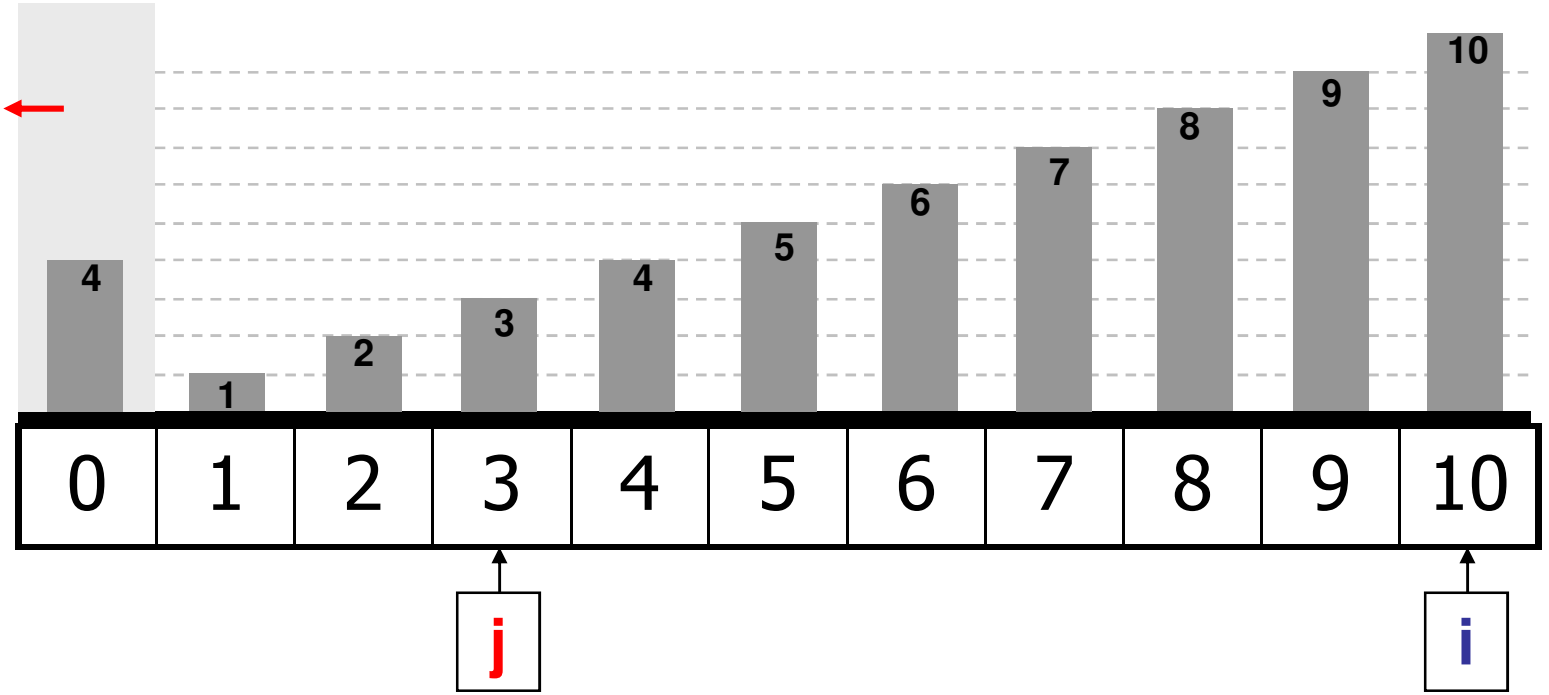
x | 4

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j]; {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



x 4

```
for i := 2 to n do
 Begin
   x := A[i];
   j := i-1;
  A[0] := x;      {sentinela}
  while x < A[j] do
   Begin
    A[j+1] := A[j];      {Empurra maiores p/ direita}
    j := j-1;
  End;
  A[j+1] := x;
 End;
```



x | 4

FIM!

Ordenação por Troca

- Este método permuta sistematicamente pares de elementos que estão fora de ordem até que não haja mais pares não-ordenados;
- Estratégia bolha, ou *bubblesort*: consiste em “borbulhar” o maior elemento para o fim da lista;

Ordenação por Troca

- Inicialmente, percorre-se a lista comparando pares de elementos consecutivos e trocando os que estiverem fora de ordem. Nesta primeira varredura, o maior elemento é “empurrado” para o fim da lista;
- Na segunda varredura, o segundo maior elemento será empurrado para a penúltima posição, e assim por diante;

Ordenação por Troca

- Exemplo:

Varredura 1	**1**	**2**	**3**	**4**	**5**	**6**
Não troca	**O**	R	D	E	N	A
Troca 2 e 3	O	**R**	D	E	N	A
Troca 3 e 4	O	D	**R**	E	N	A
Troca 4 e 5	O	D	E	**R**	N	A
Troca 5 e 6	O	D	E	N	**R**	A
Maior elemento no final	O	D	E	N	A	**R**

Ordenação por Troca

- Exemplo (continuação):

Varredura 2	**1**	**2**	**3**	**4**	**5**	**6**
Troca 1 e 2	**O**	D	E	N	A	**R**
Troca 2 e 3	D	**O**	E	N	A	**R**
Troca 3 e 4	D	E	**O**	N	A	**R**
Troca 4 e 5	D	E	N	**O**	A	**R**
Segundo maior elemento “empurrado” para a penúltima posição	D	E	N	A	**O**	**R**

Ordenação por Troca

- Para uma lista de tamanho n, são necessárias n-1 varreduras, pois cada varredura leva um elemento para a sua posição definitiva no fim da sublista. O menor elemento de todos vai “sobrar” na primeira posição;
- Pior caso: quando a lista está em ordem invertida, neste caso o número de trocas é igual a (n-1)+(n-2)+(n-3)+...+1;

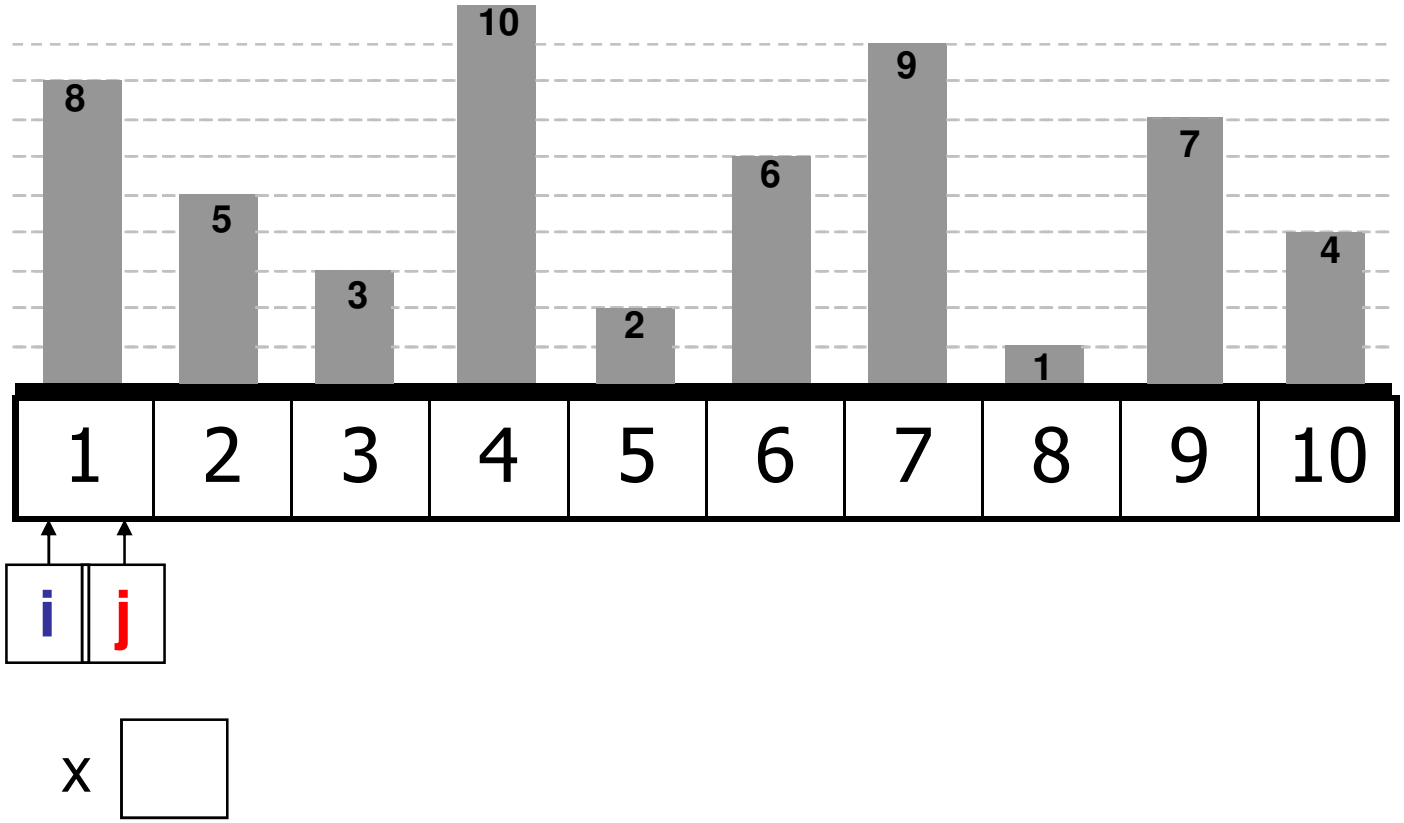
Ordenação por Troca

- ***Algoritmo:***

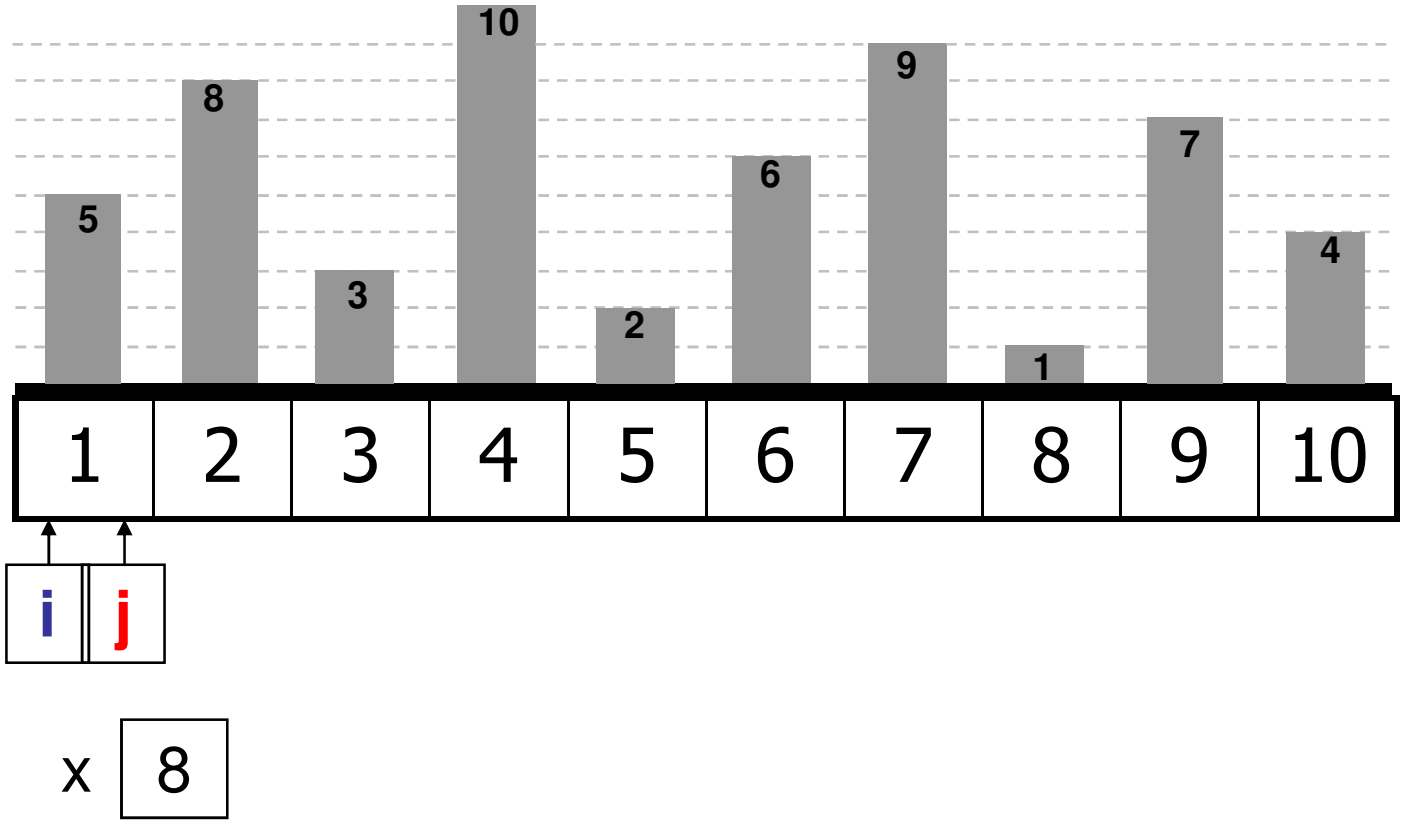
```
Procedure Troca (var A: vetor);
Var
   i, j, x: integer;
Begin
   for i := 1 to n-1 do
     for j := 1 to n-i do          {varredura}
         if A[j] > A[j+1] then
           Begin
             x := A[j];
             A[j] := A[j+1];
             A[j+1] := x;
           End;
End;
```

```
for i := 1 to n-1 do ←
  for j := 1 to n-i do ←      {varredura}
    if A[j] > A[j+1] then
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```

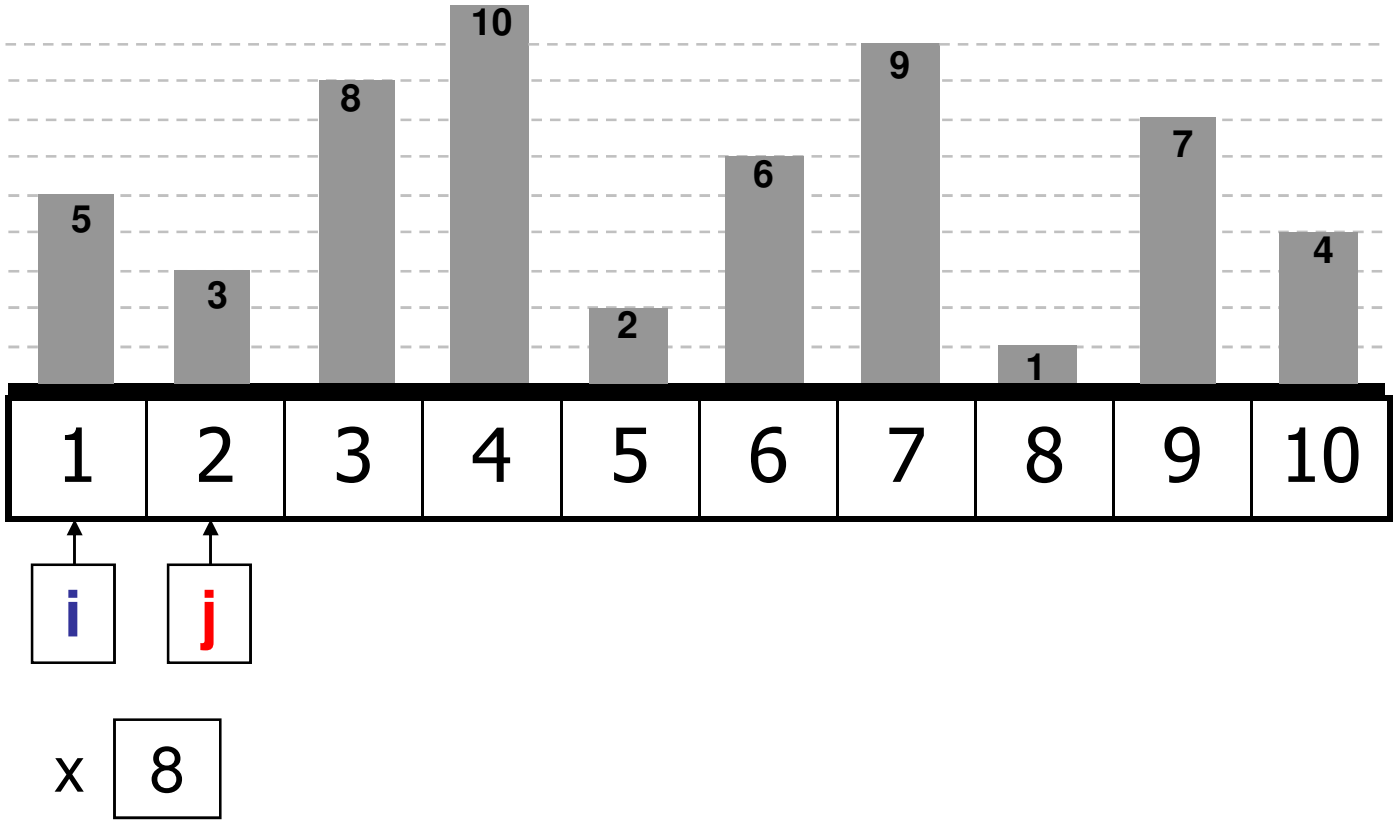
As setas vermelhas indicam operações já realizadas!



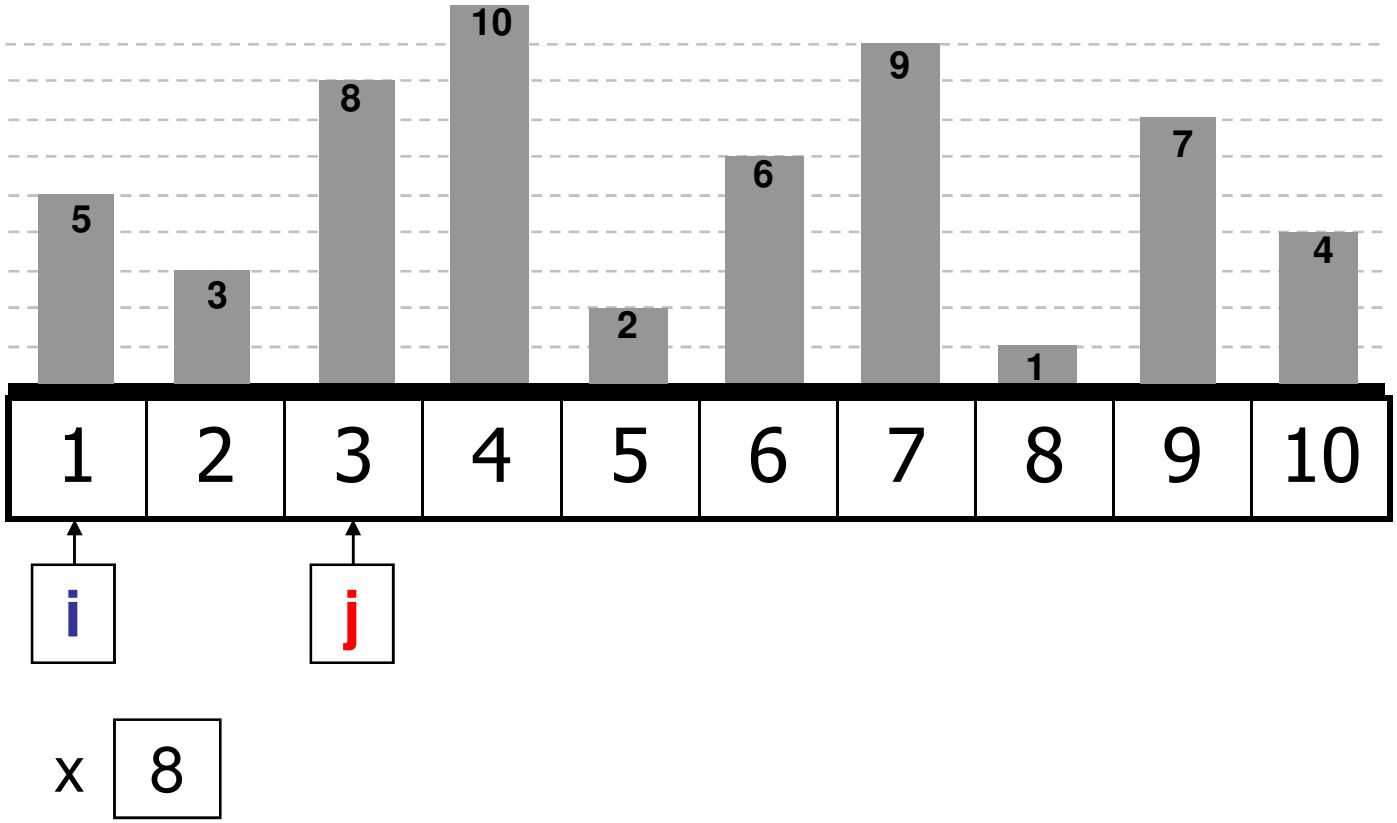
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



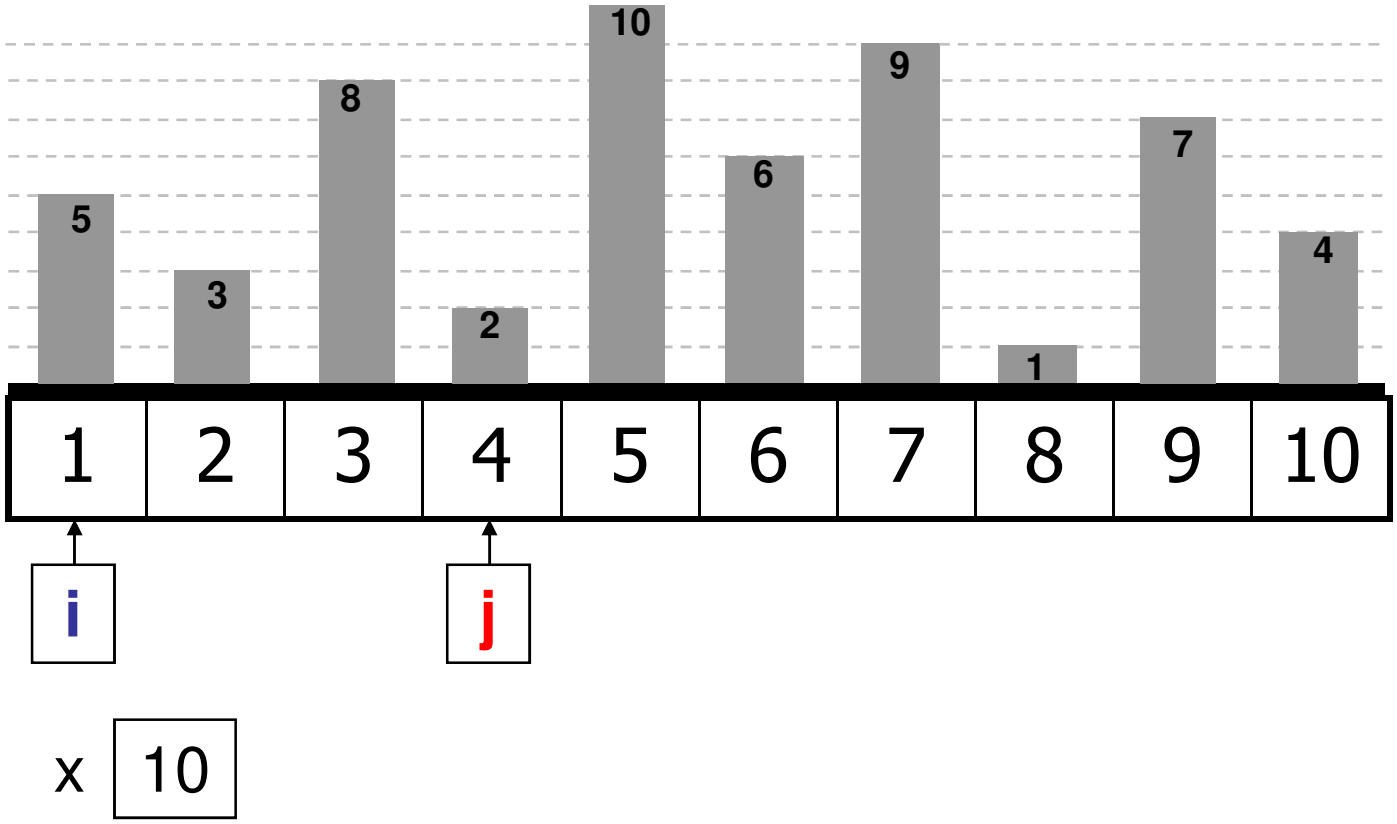
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



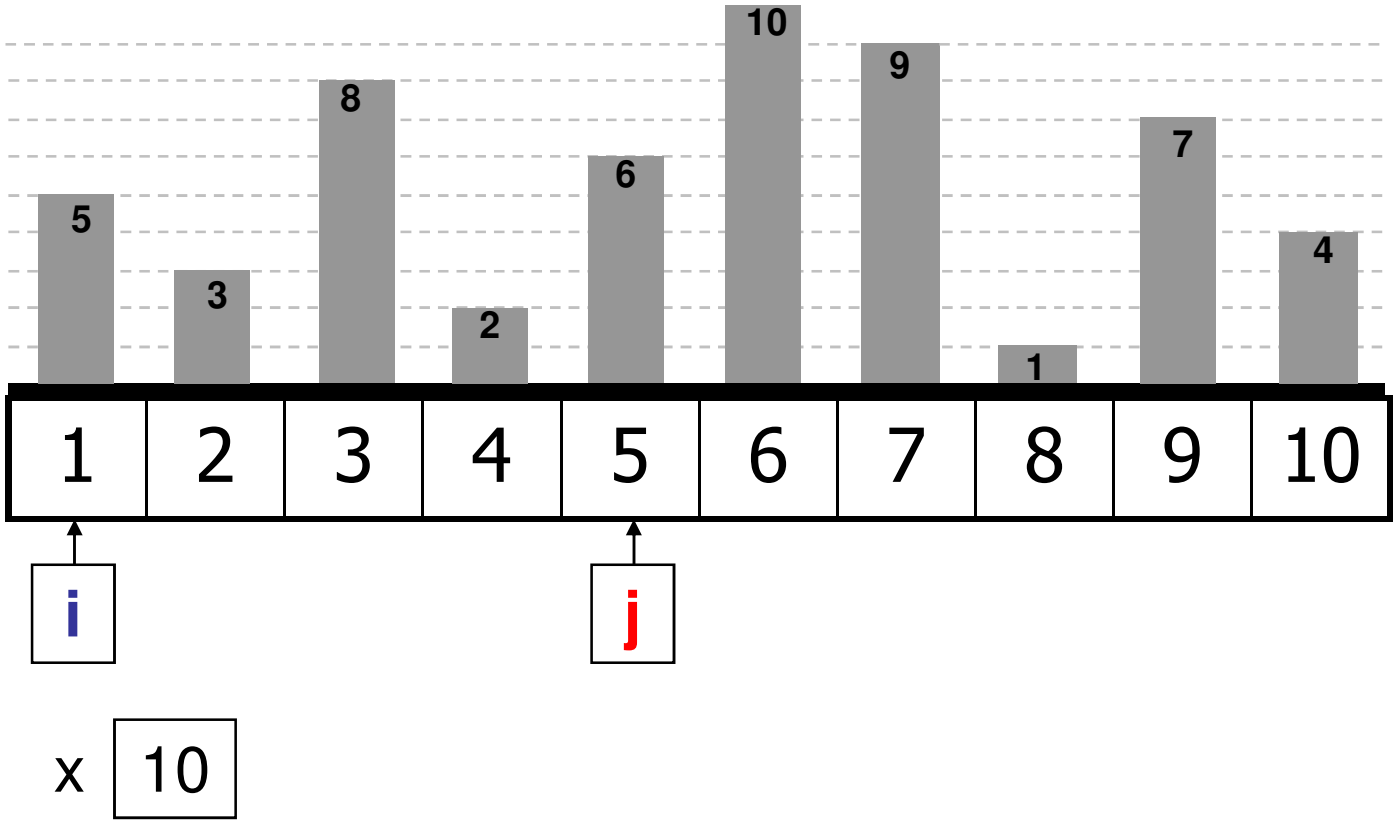
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



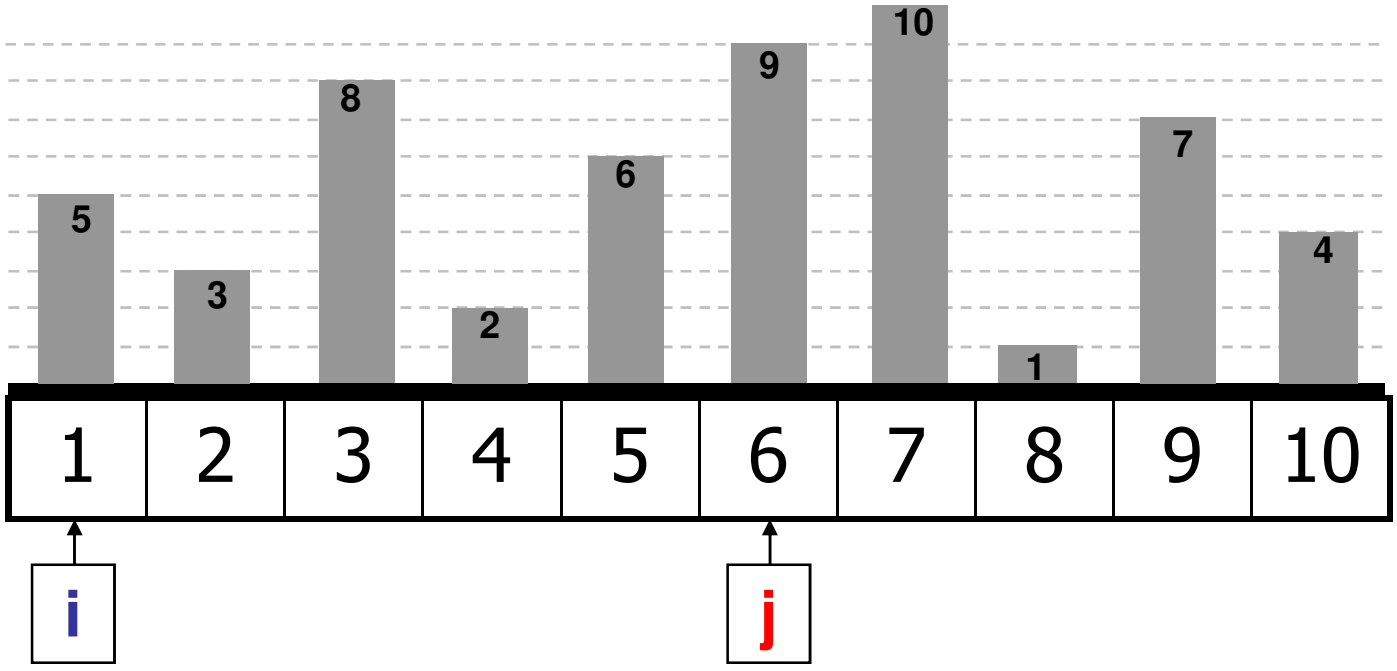
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



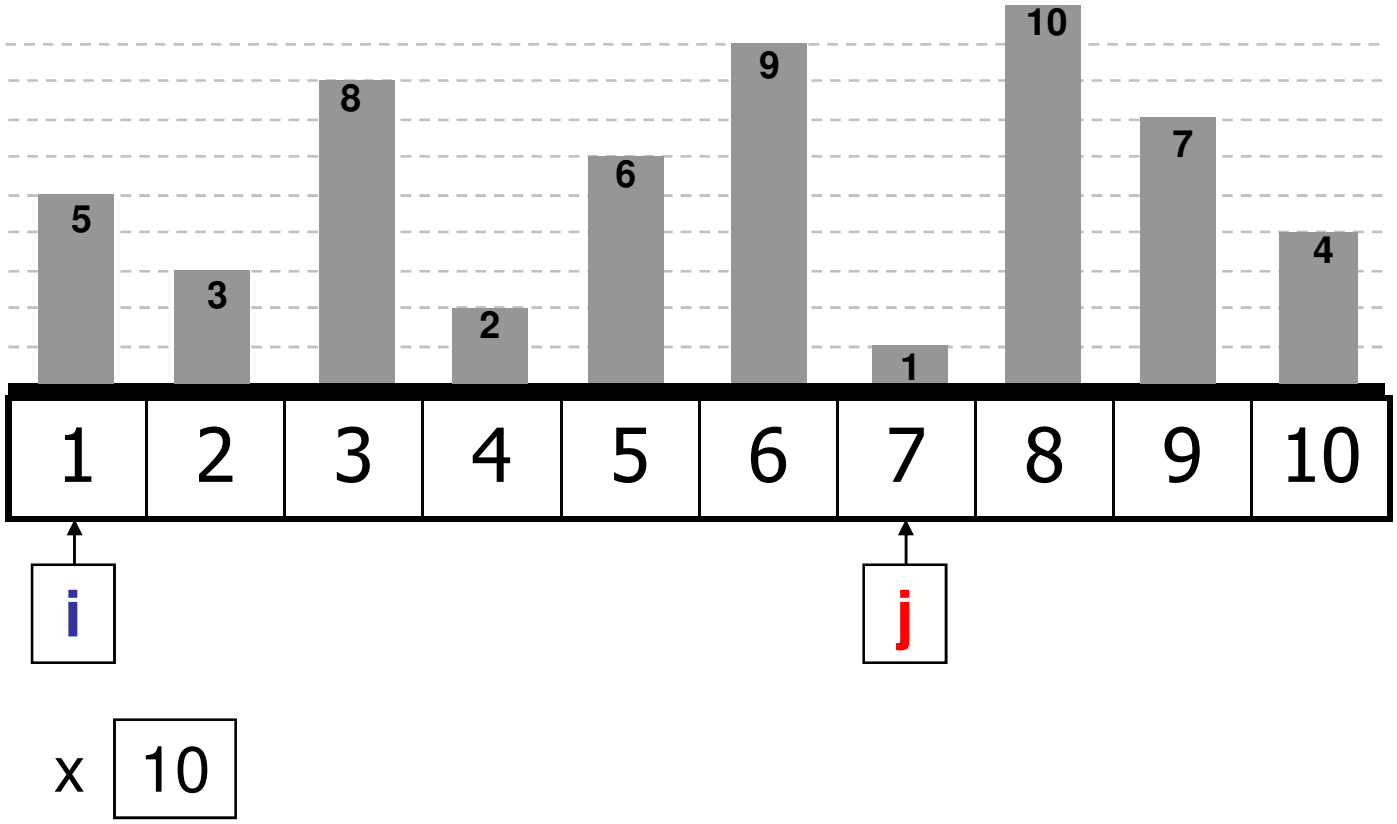
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



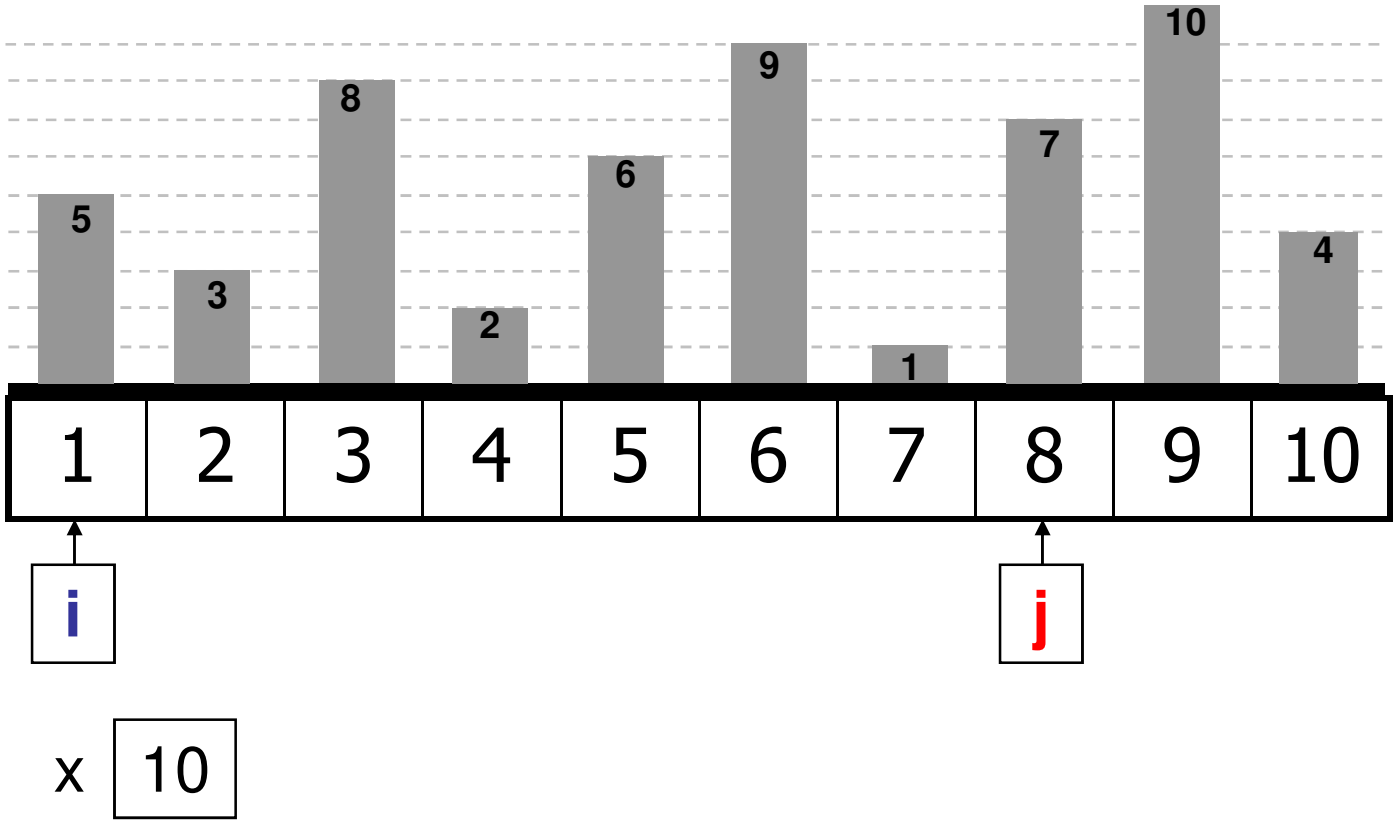
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



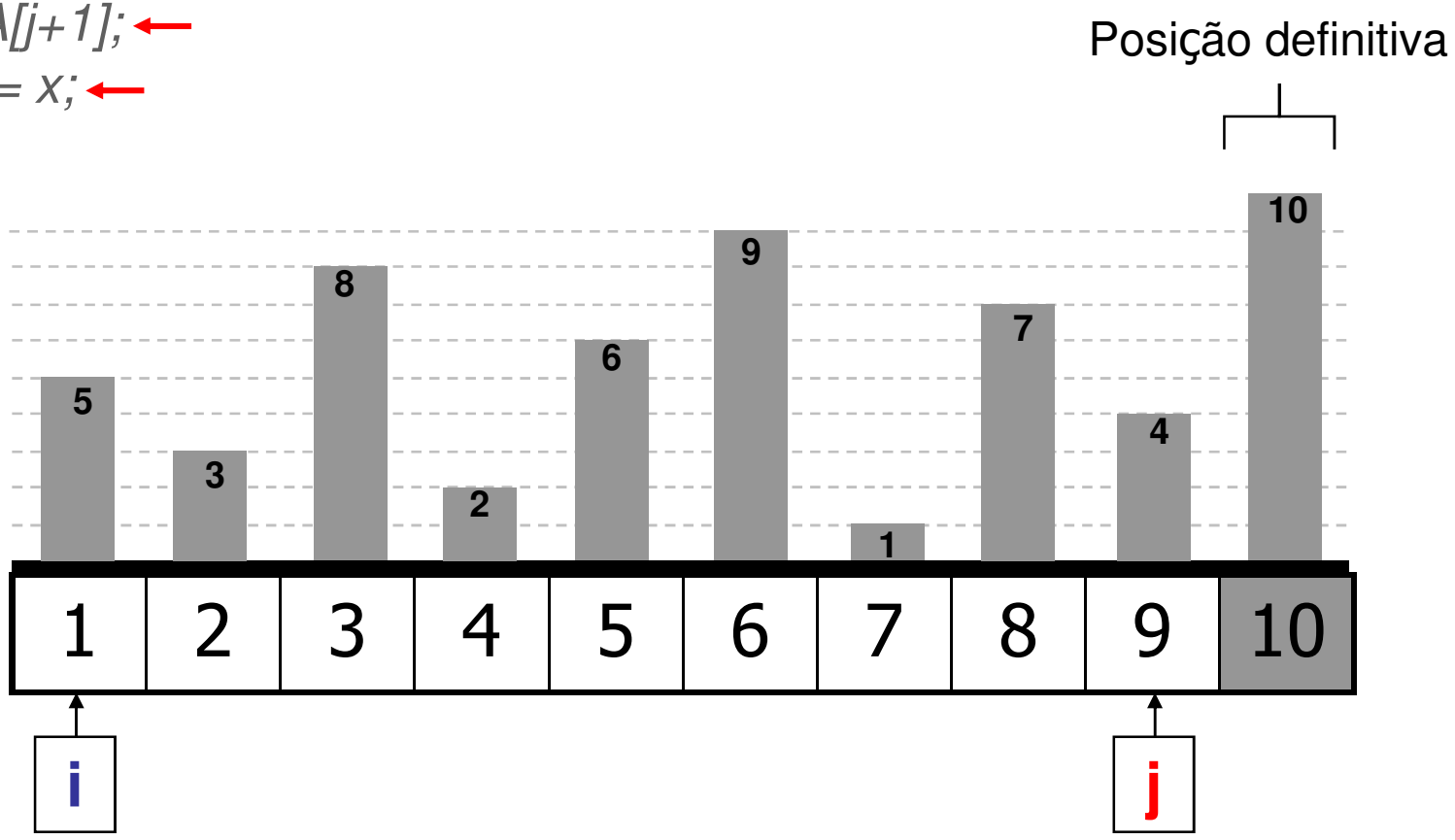
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



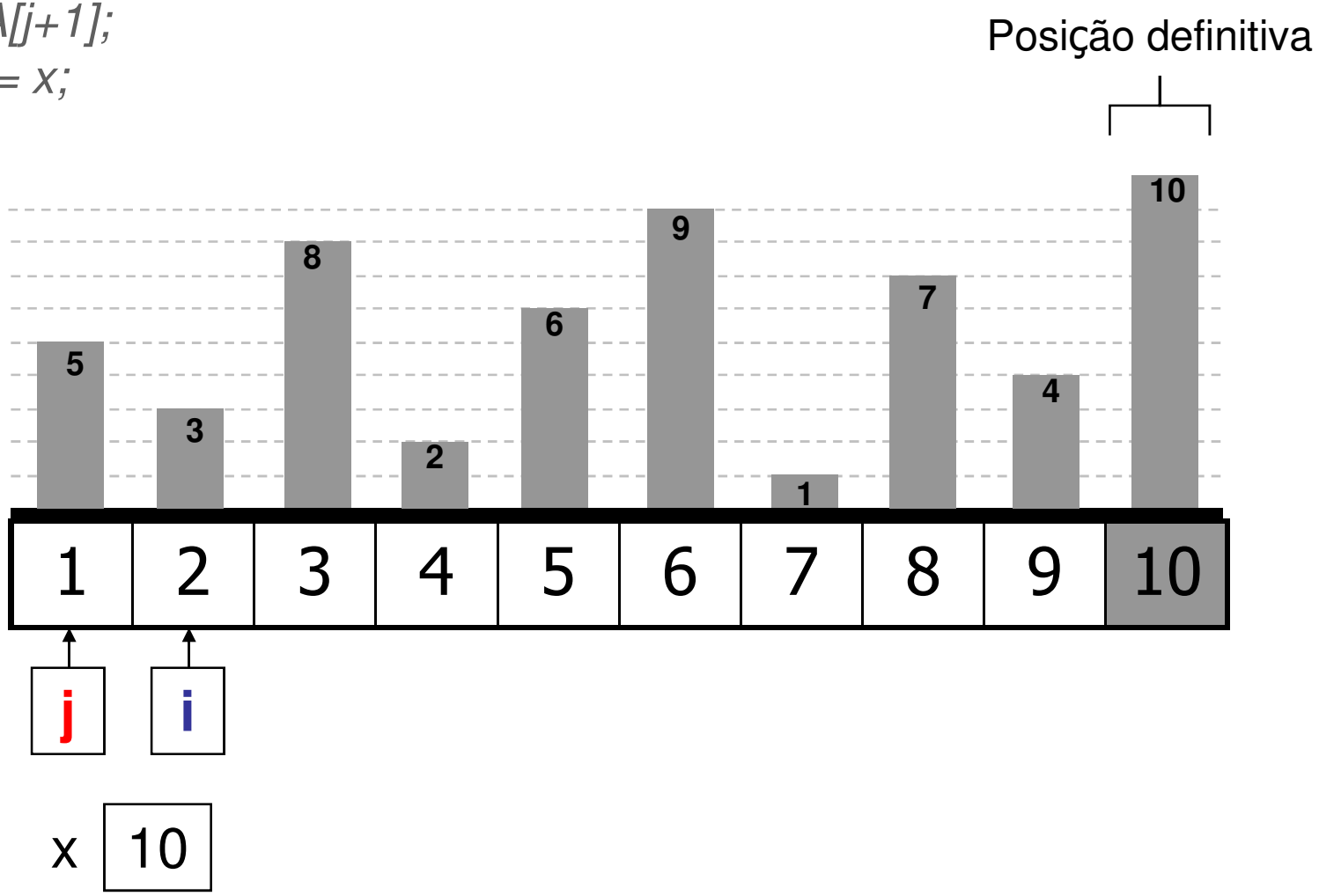
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



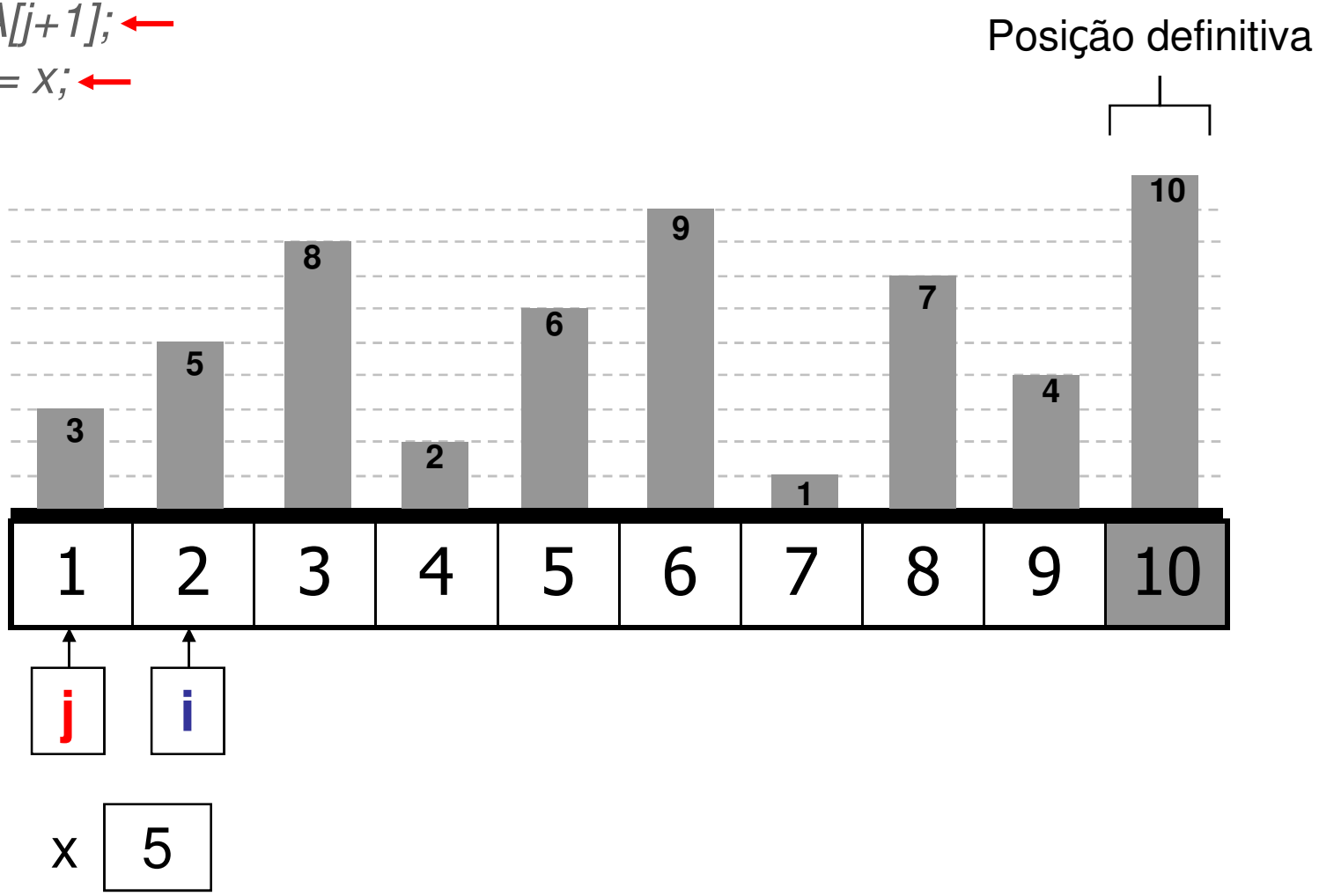
x 10

Final da primeira varredura.

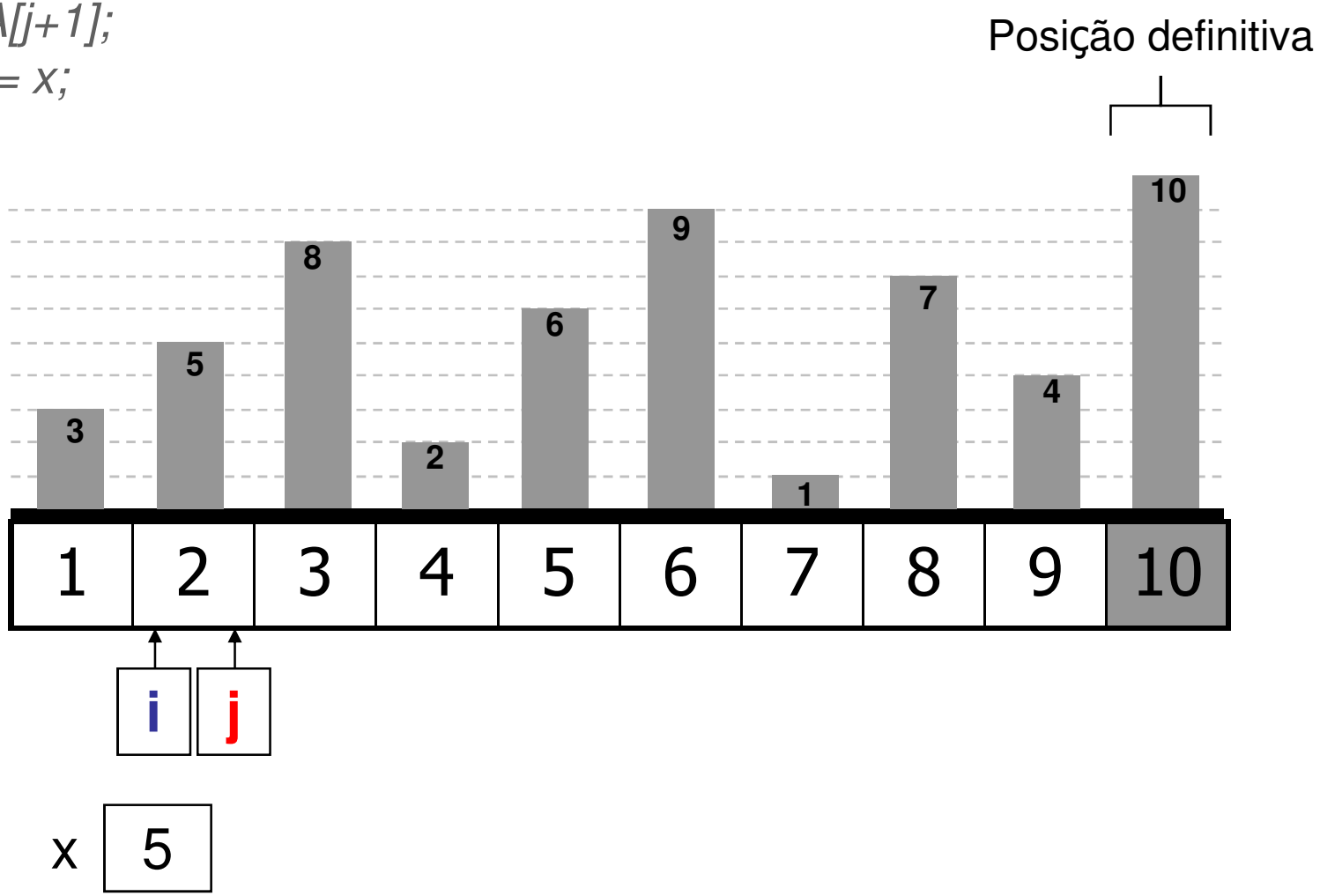
```
for i := 1 to n-1 do ←
  for j := 1 to n-i do ←      {varredura}
    if A[j] > A[j+1] then
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



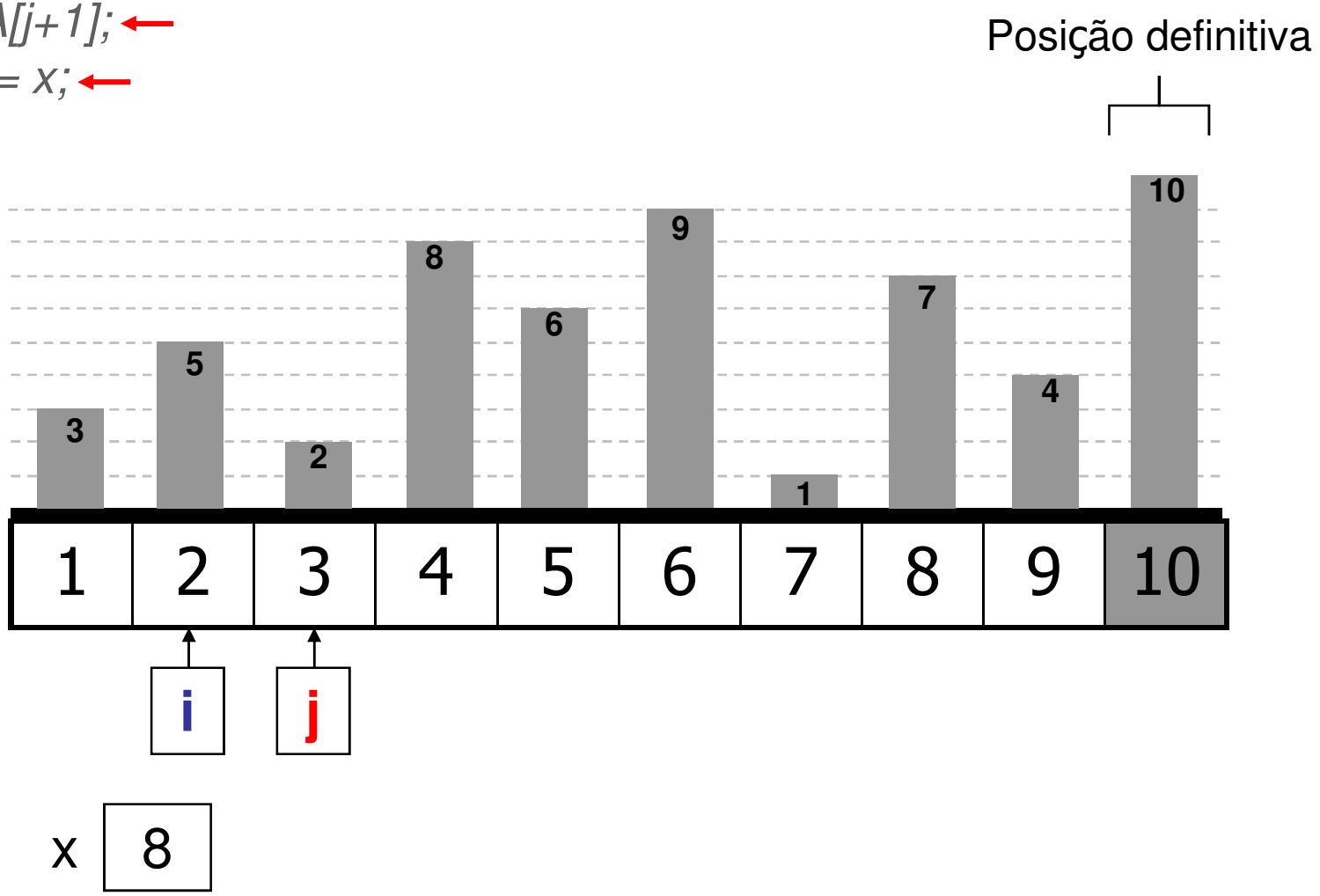
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



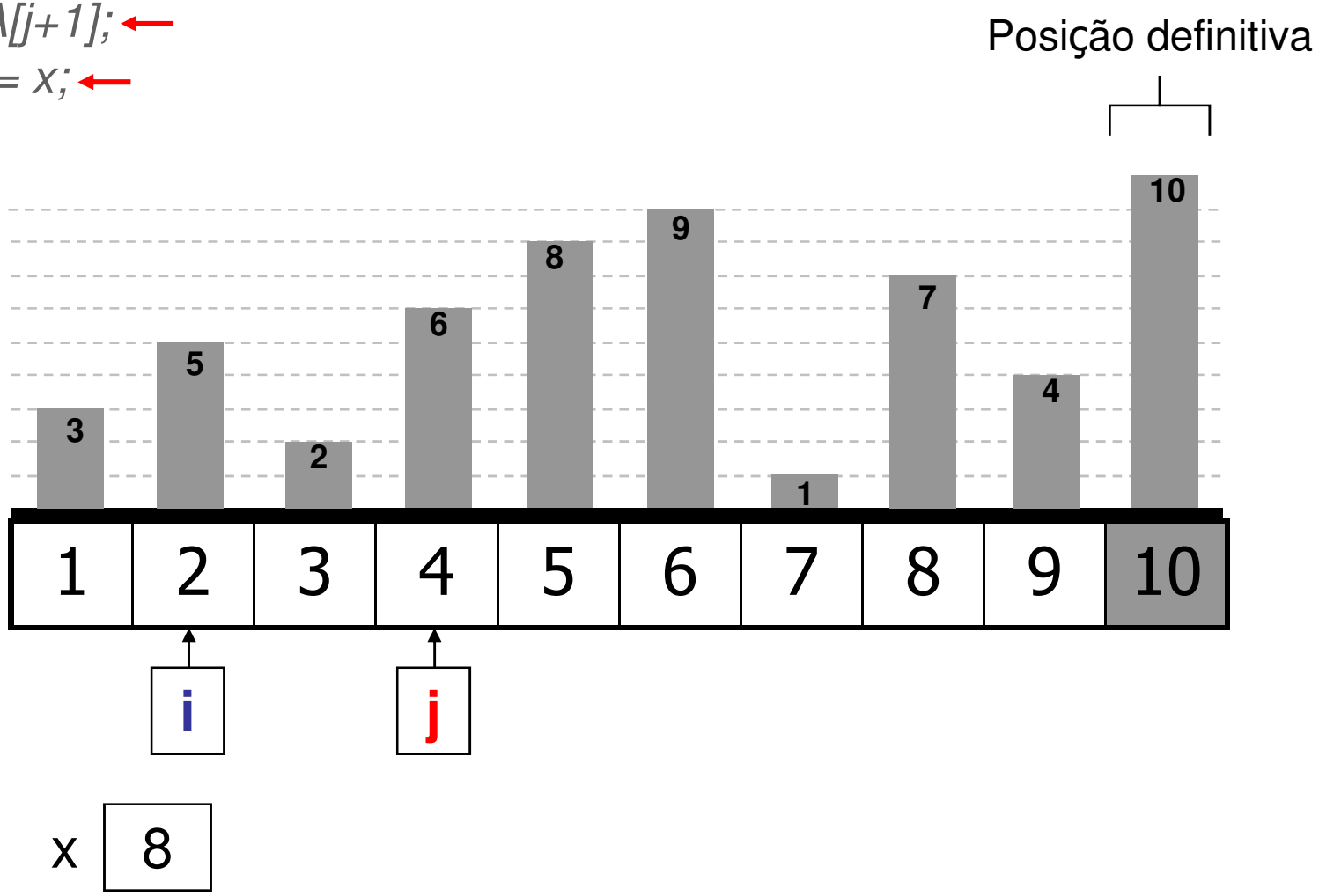
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



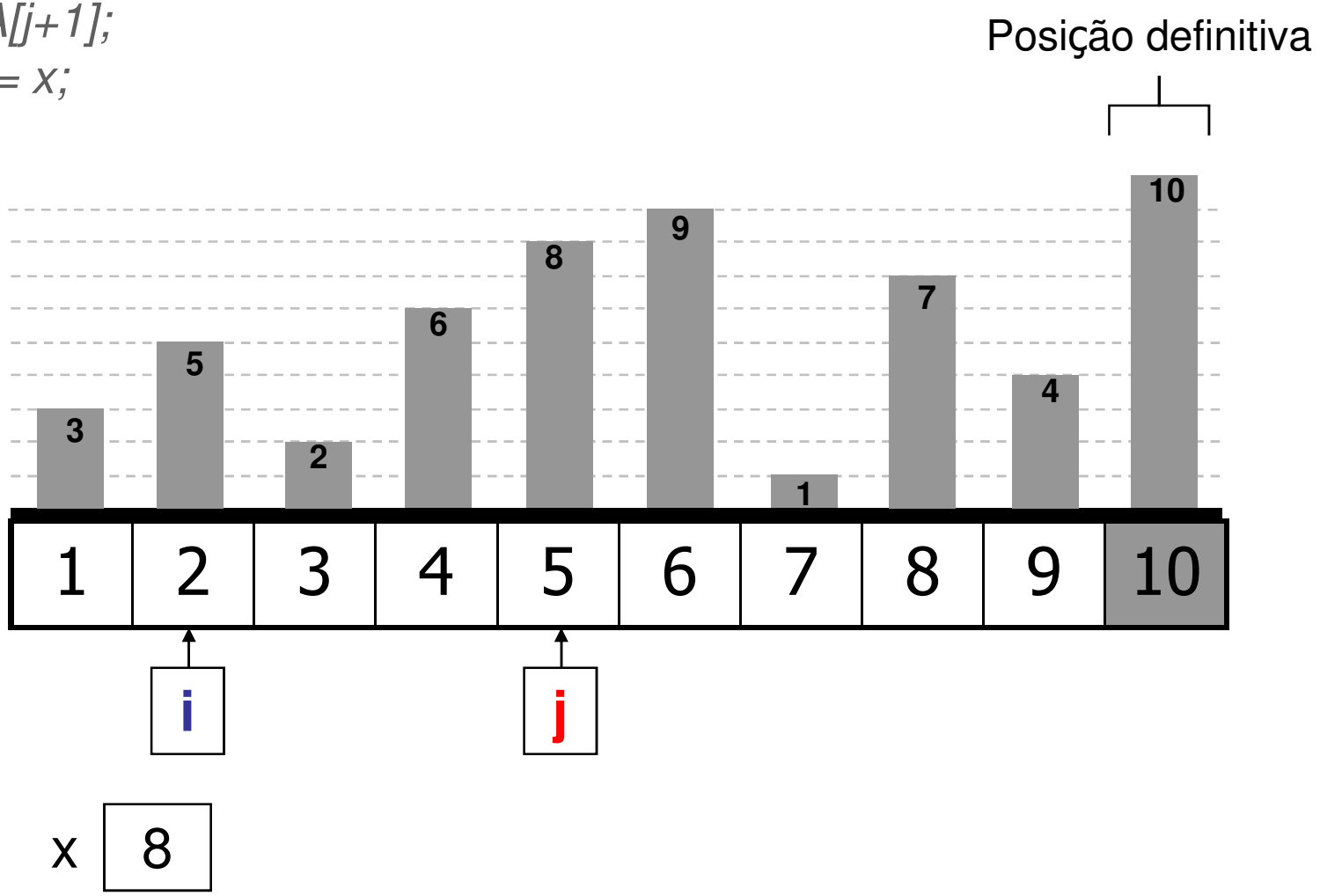
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



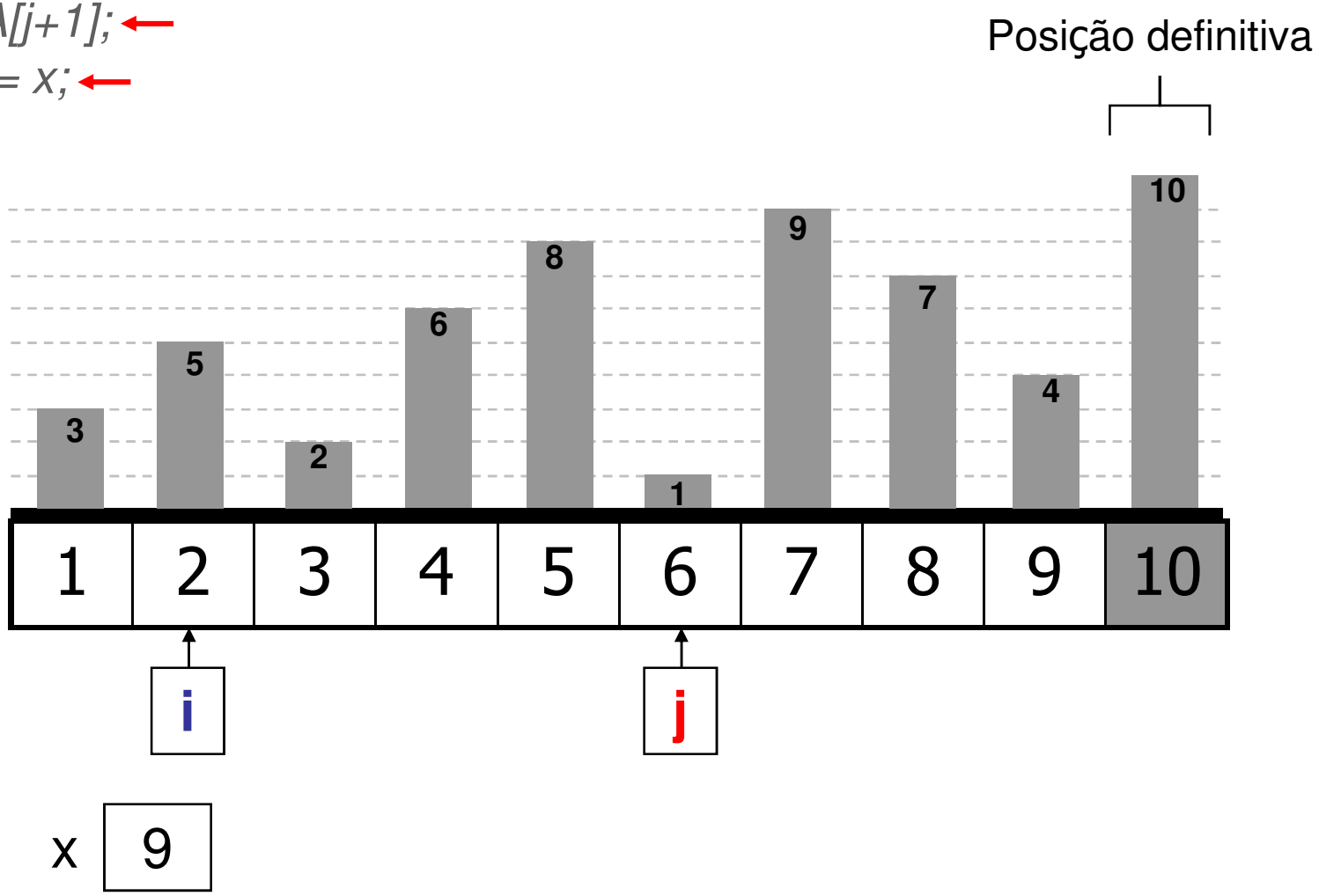
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



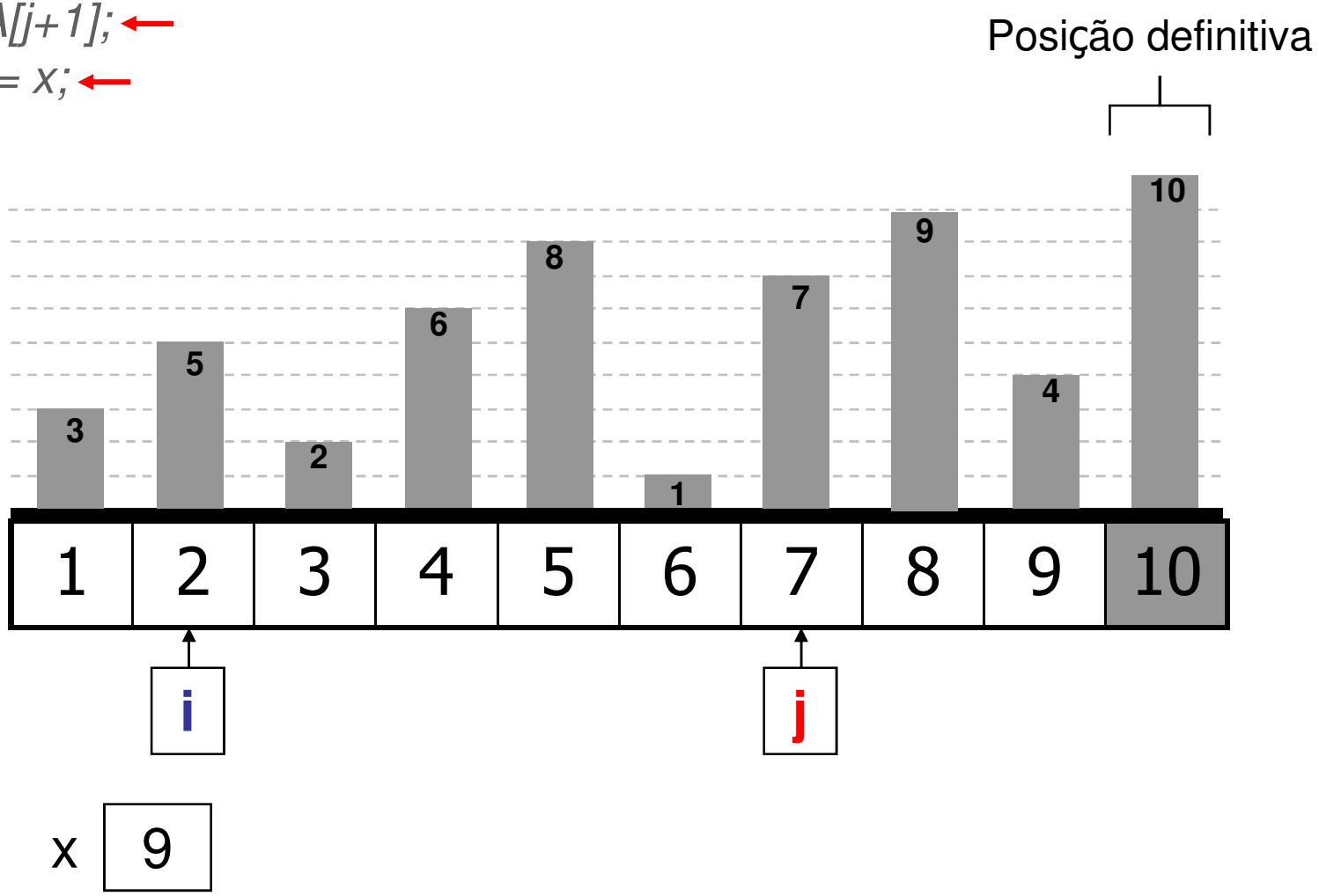
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



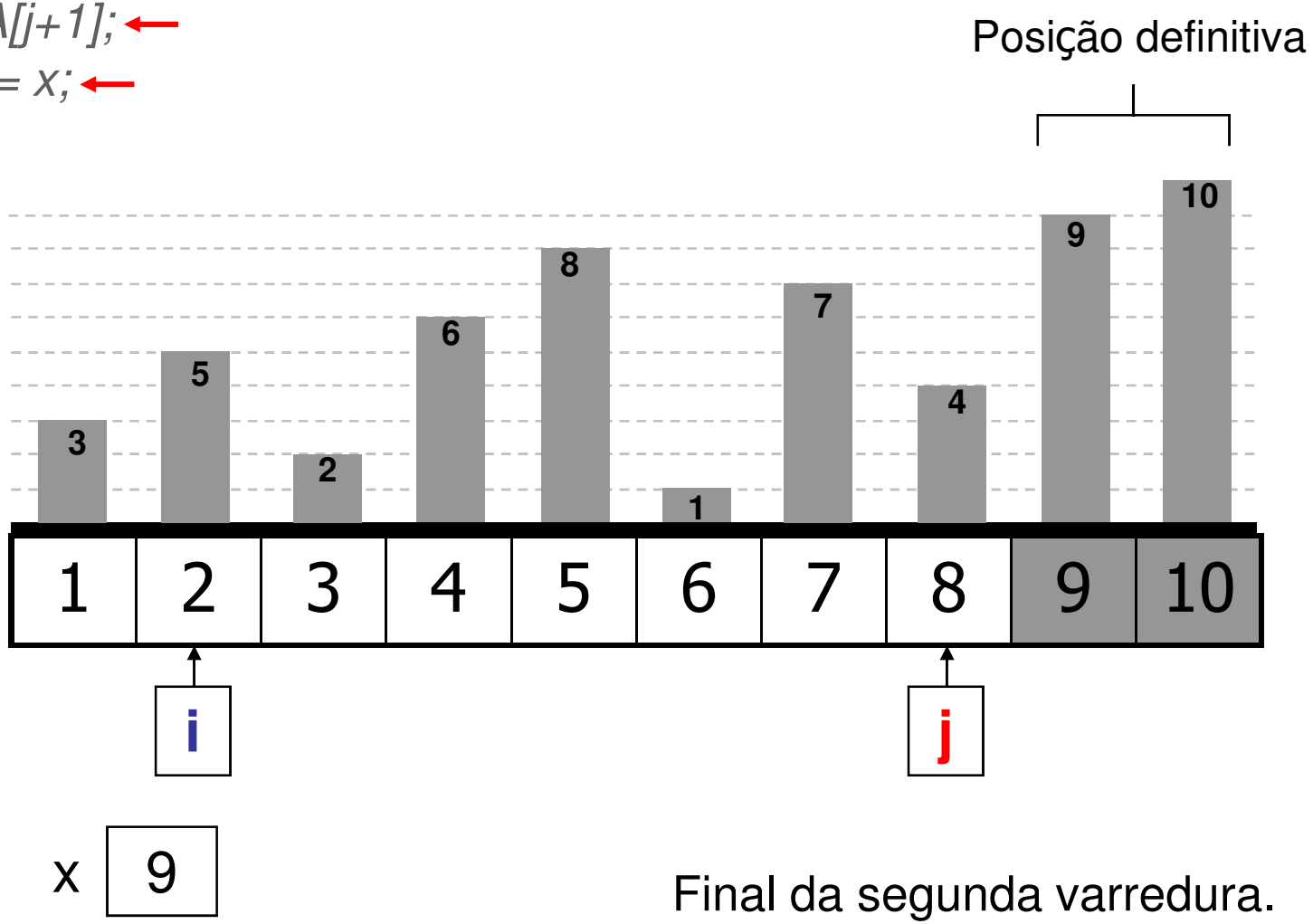
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```

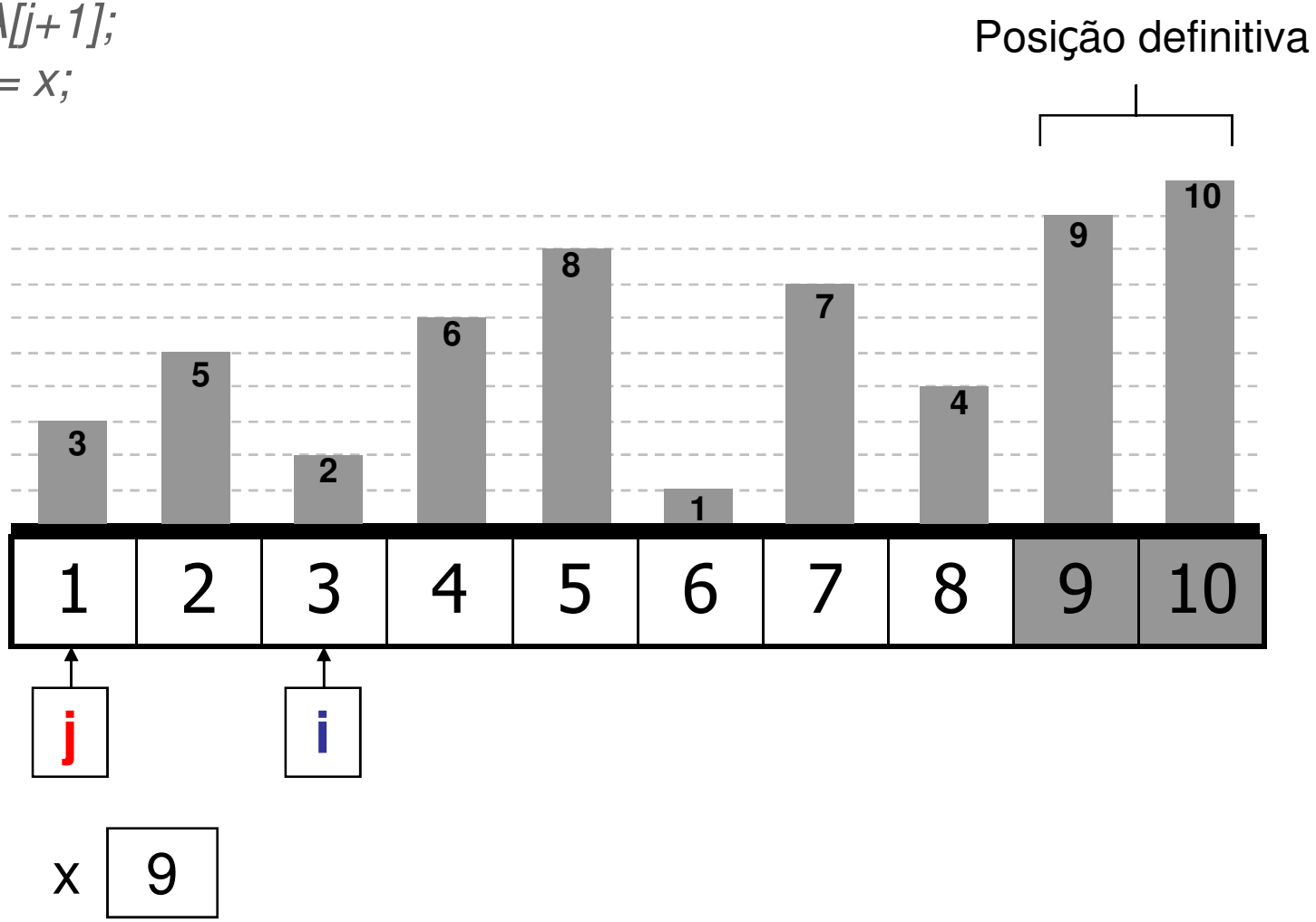


```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```

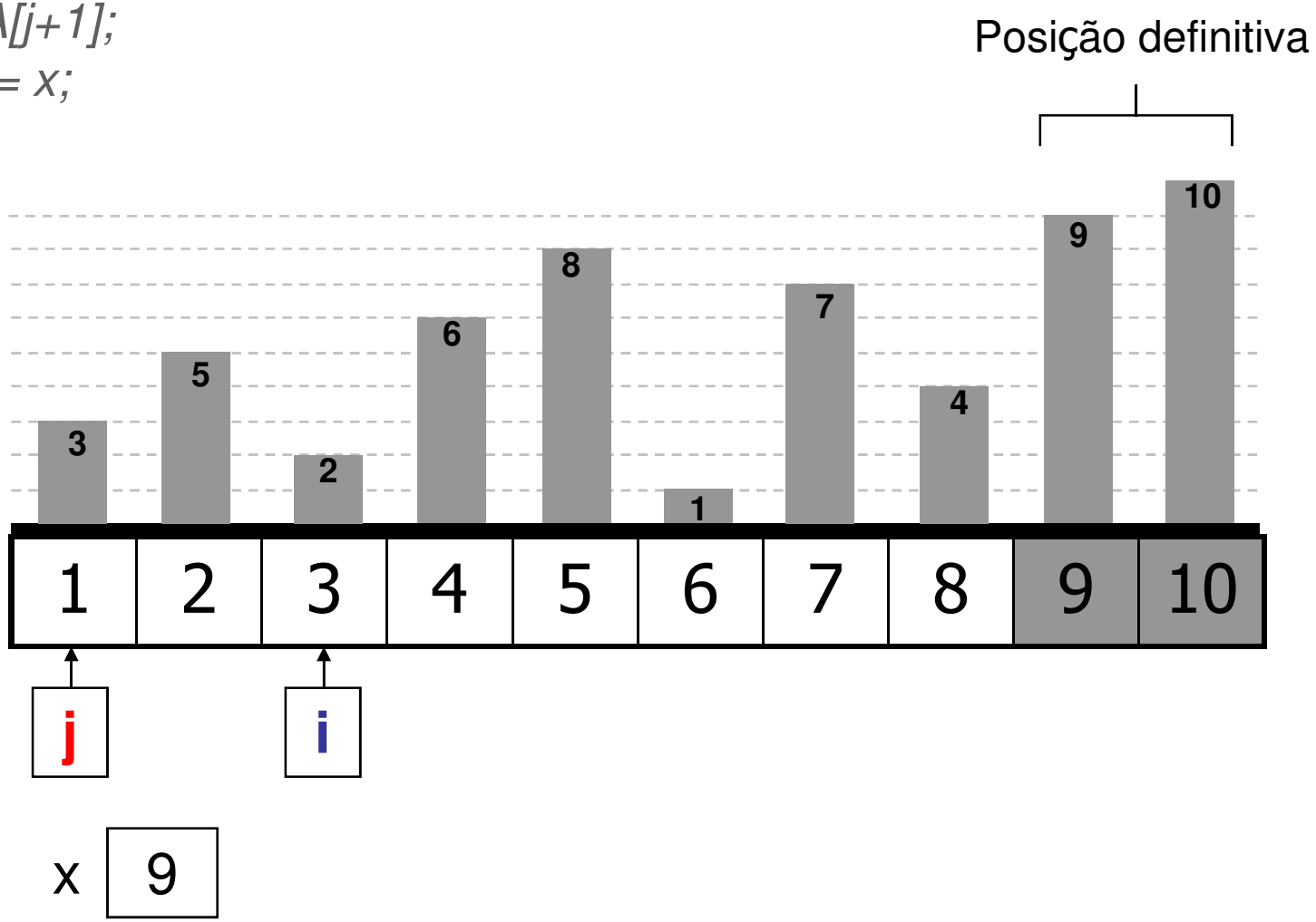


Final da segunda varredura.

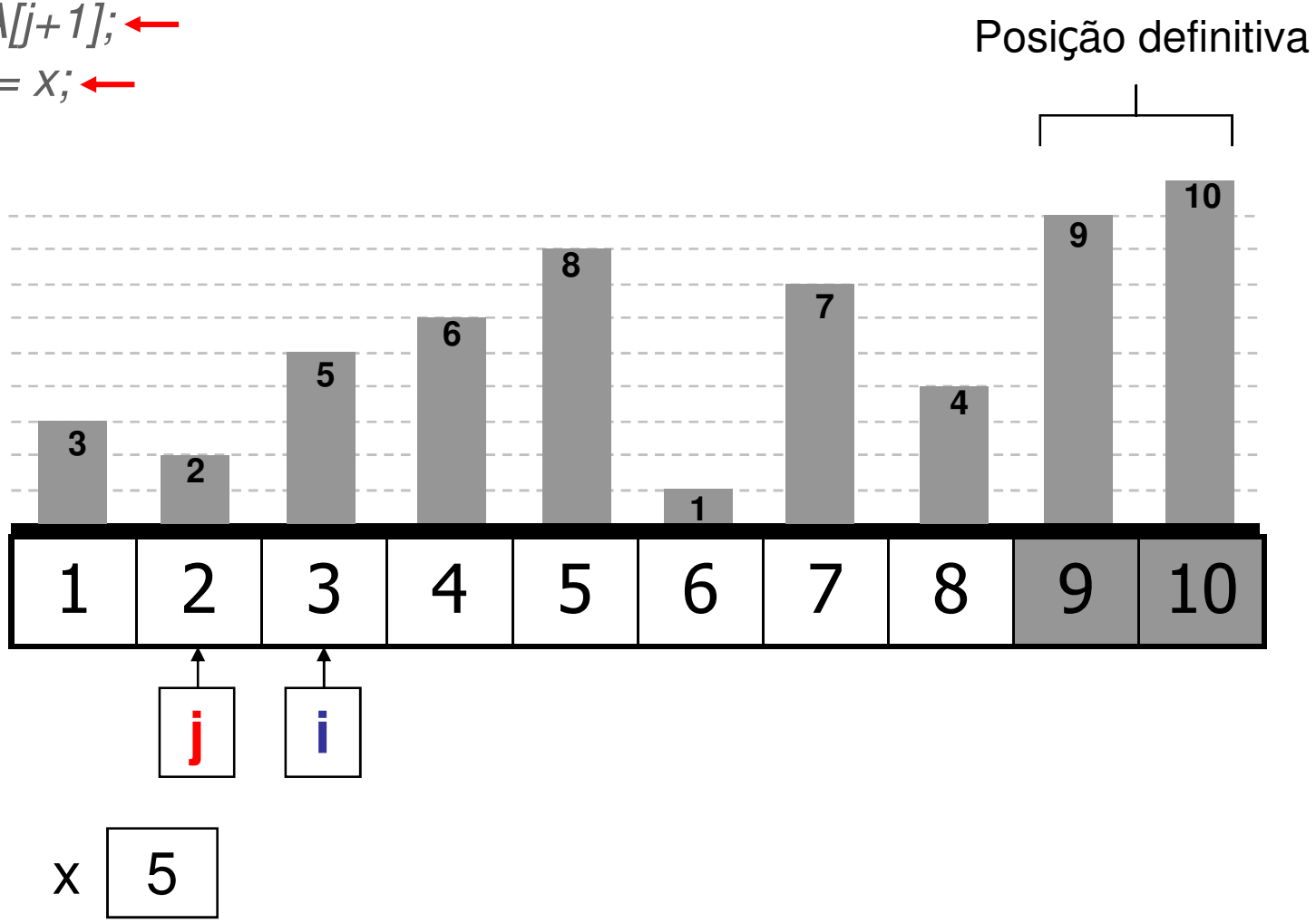
```
for i := 1 to n-1 do ←
  for j := 1 to n-i do ←      {varredura}
    if A[j] > A[j+1] then
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



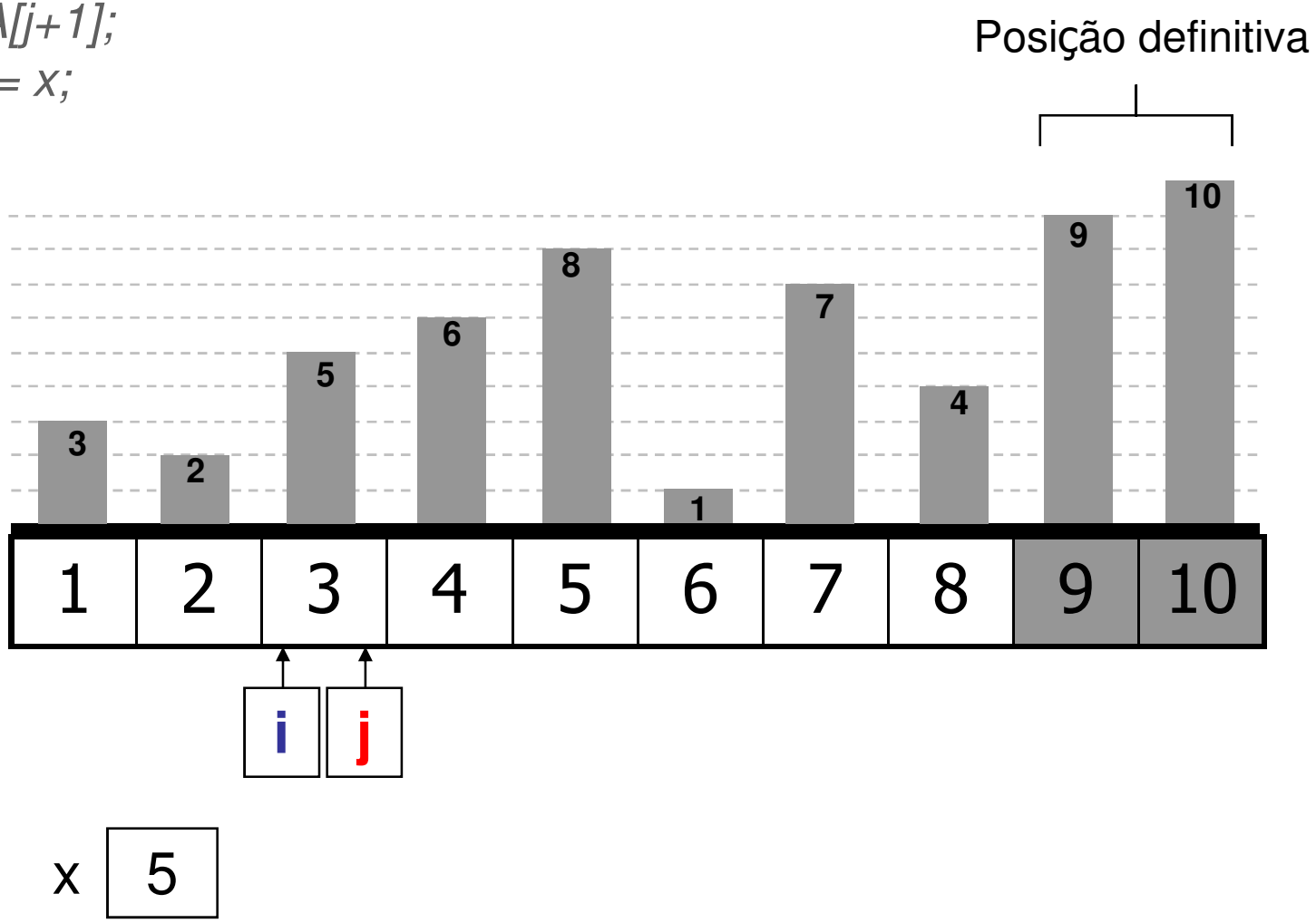
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



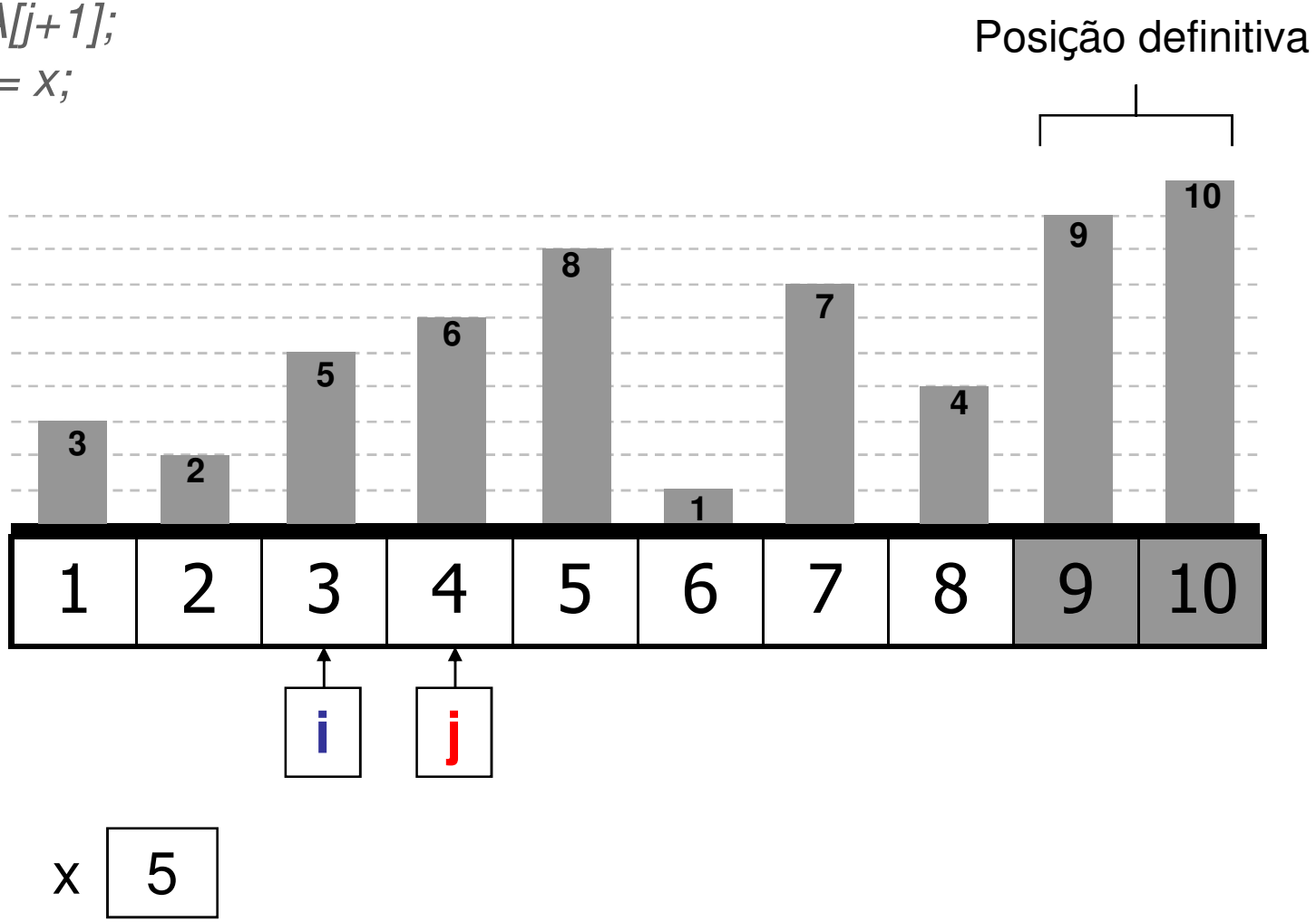
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



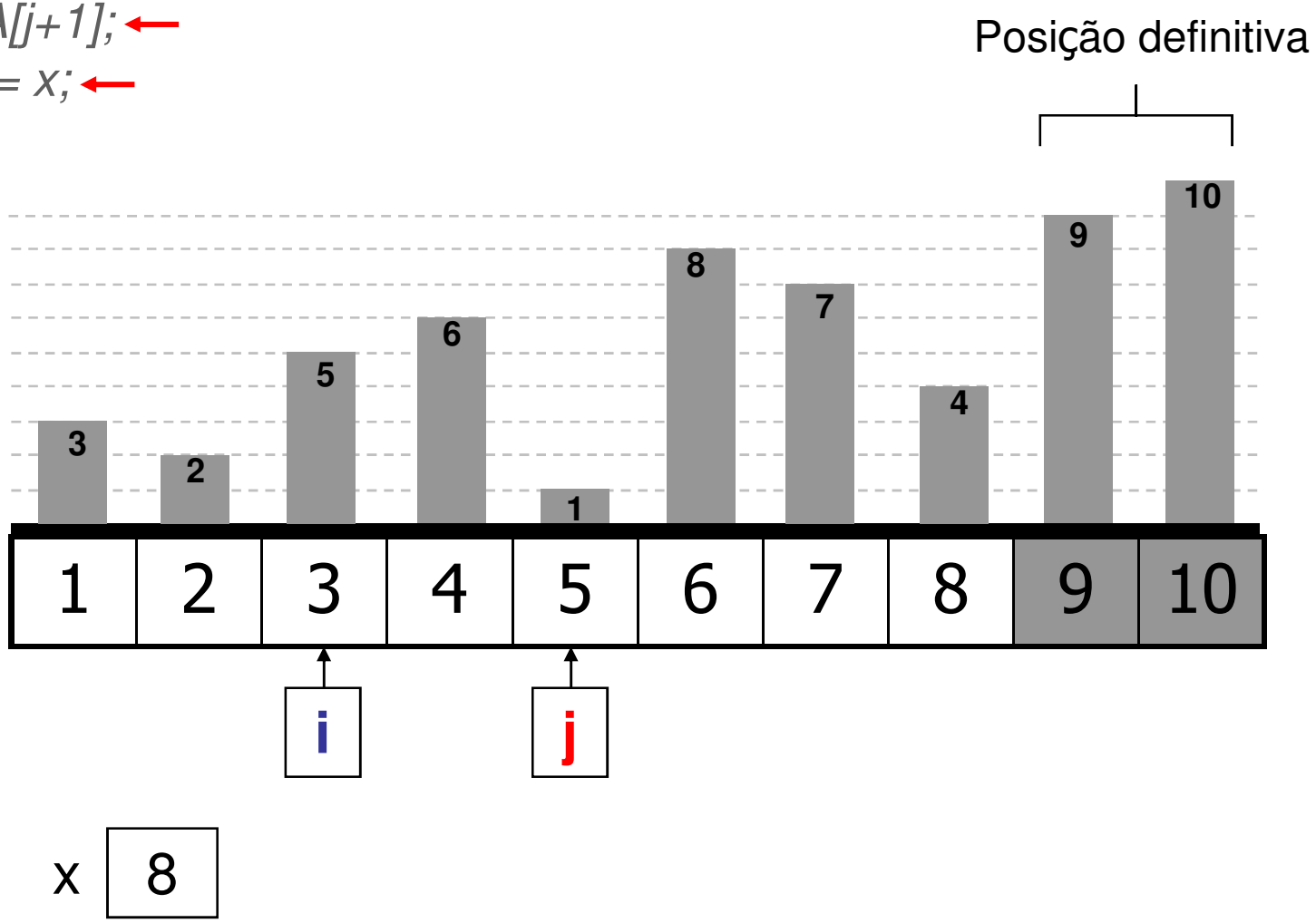
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



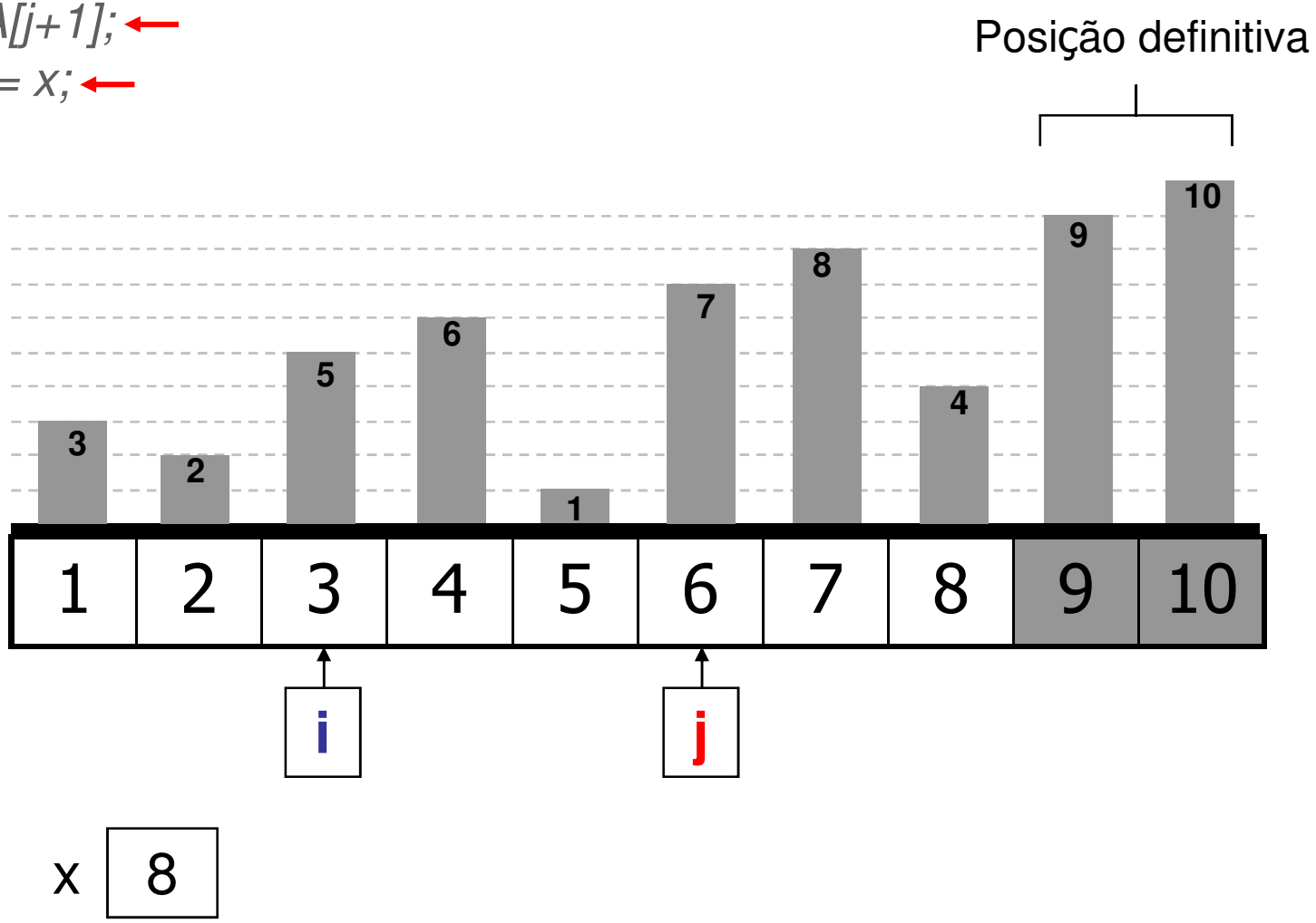
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



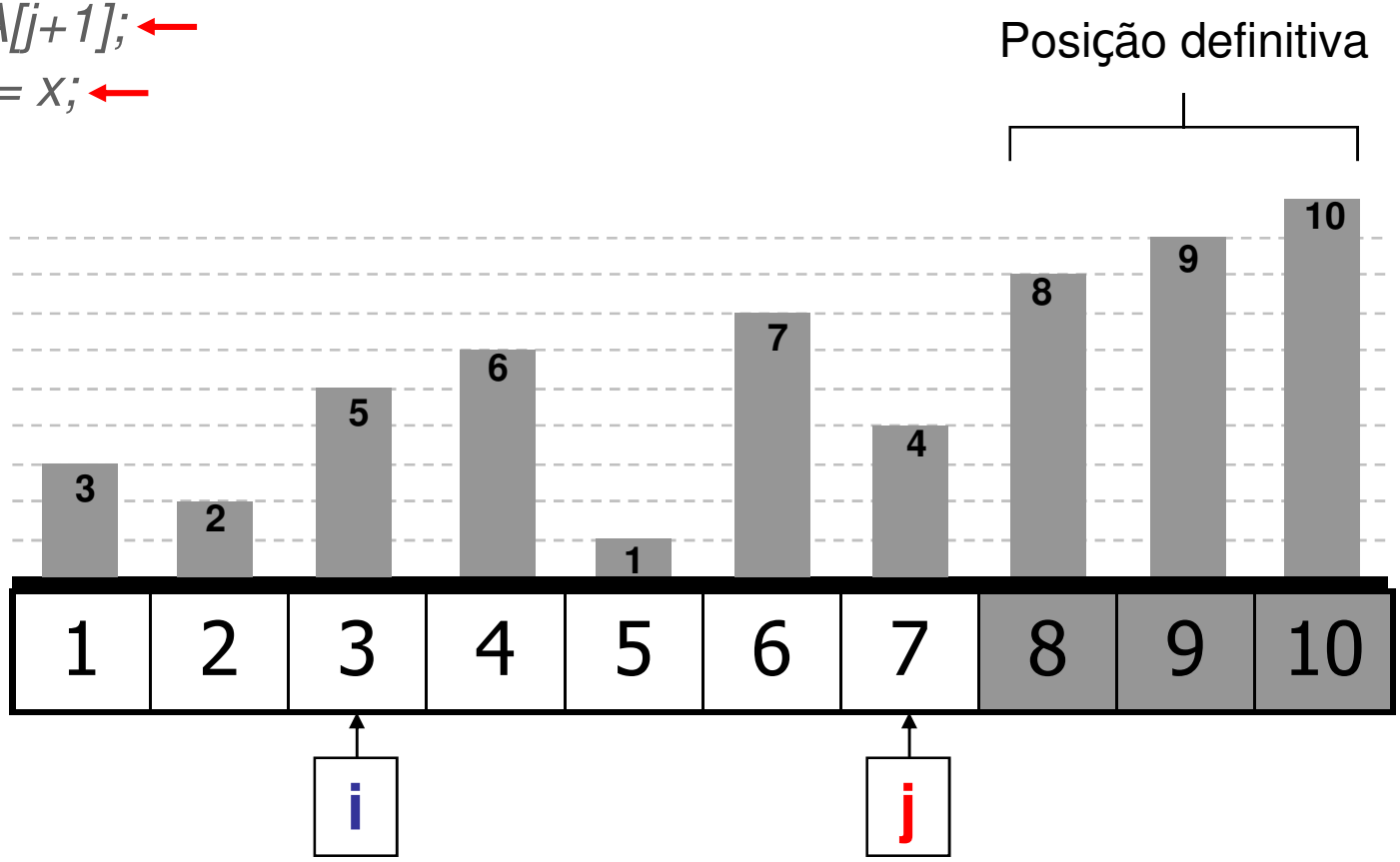
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



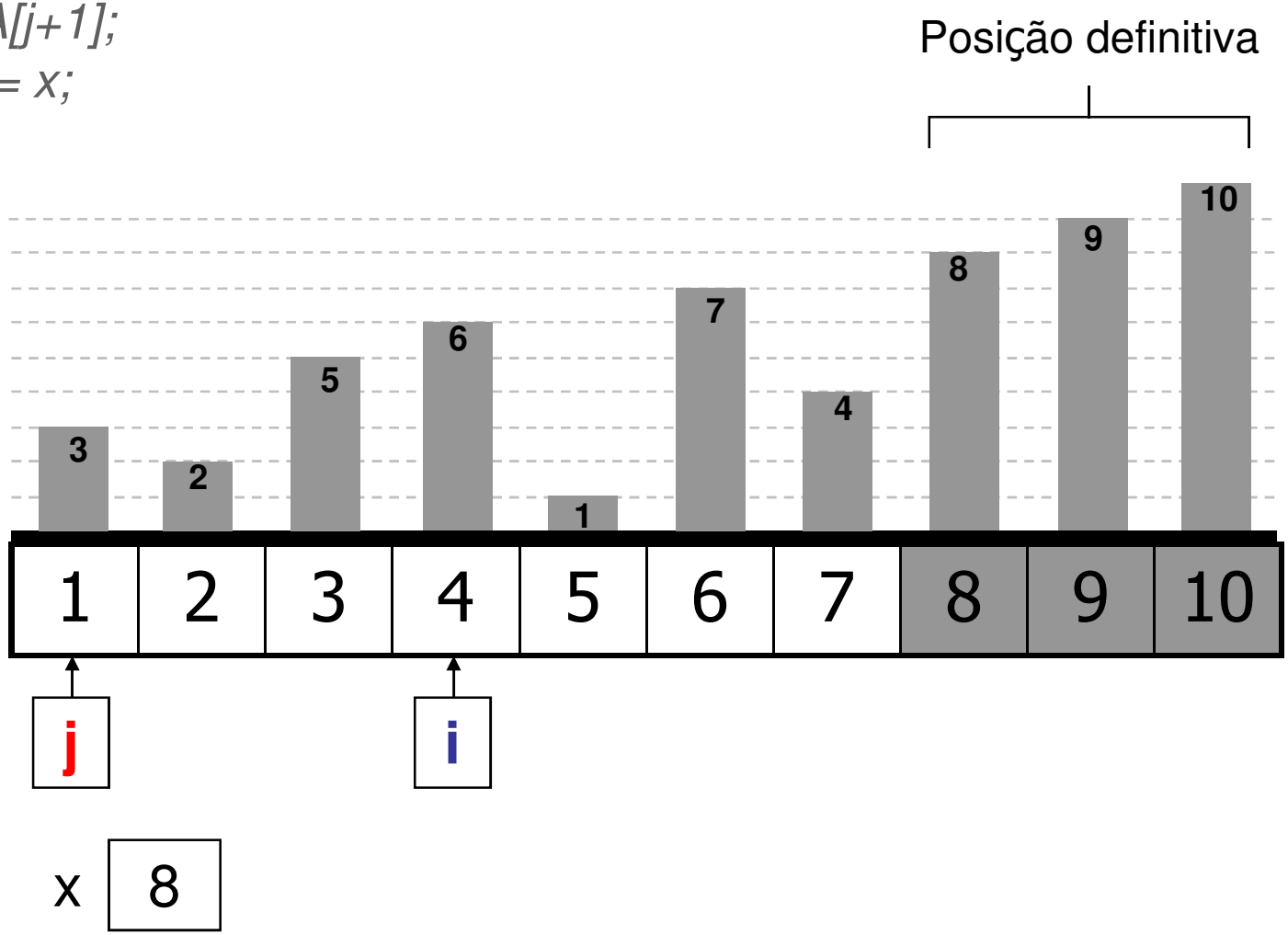
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



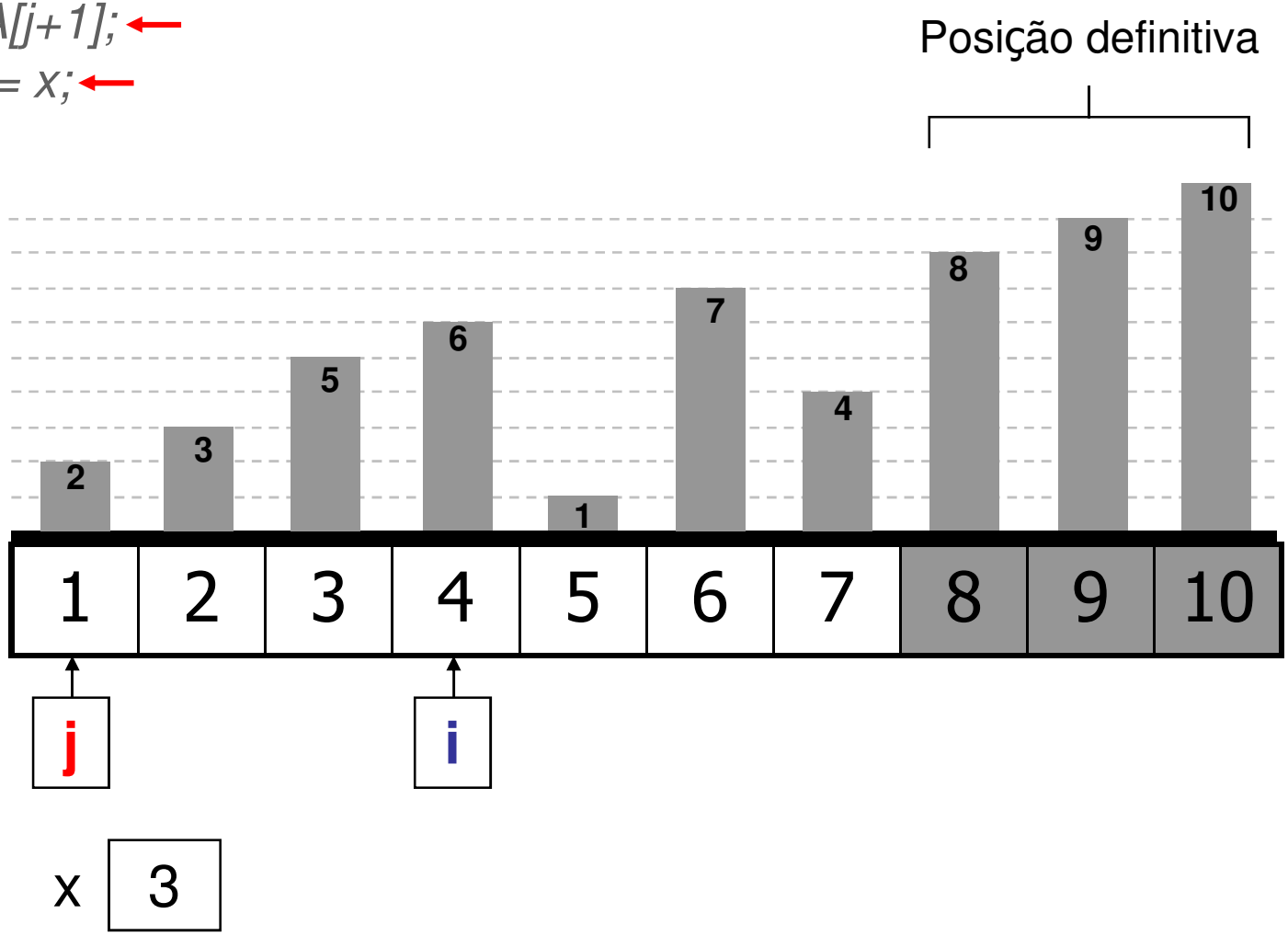
x | 8

Final da terceira varredura.

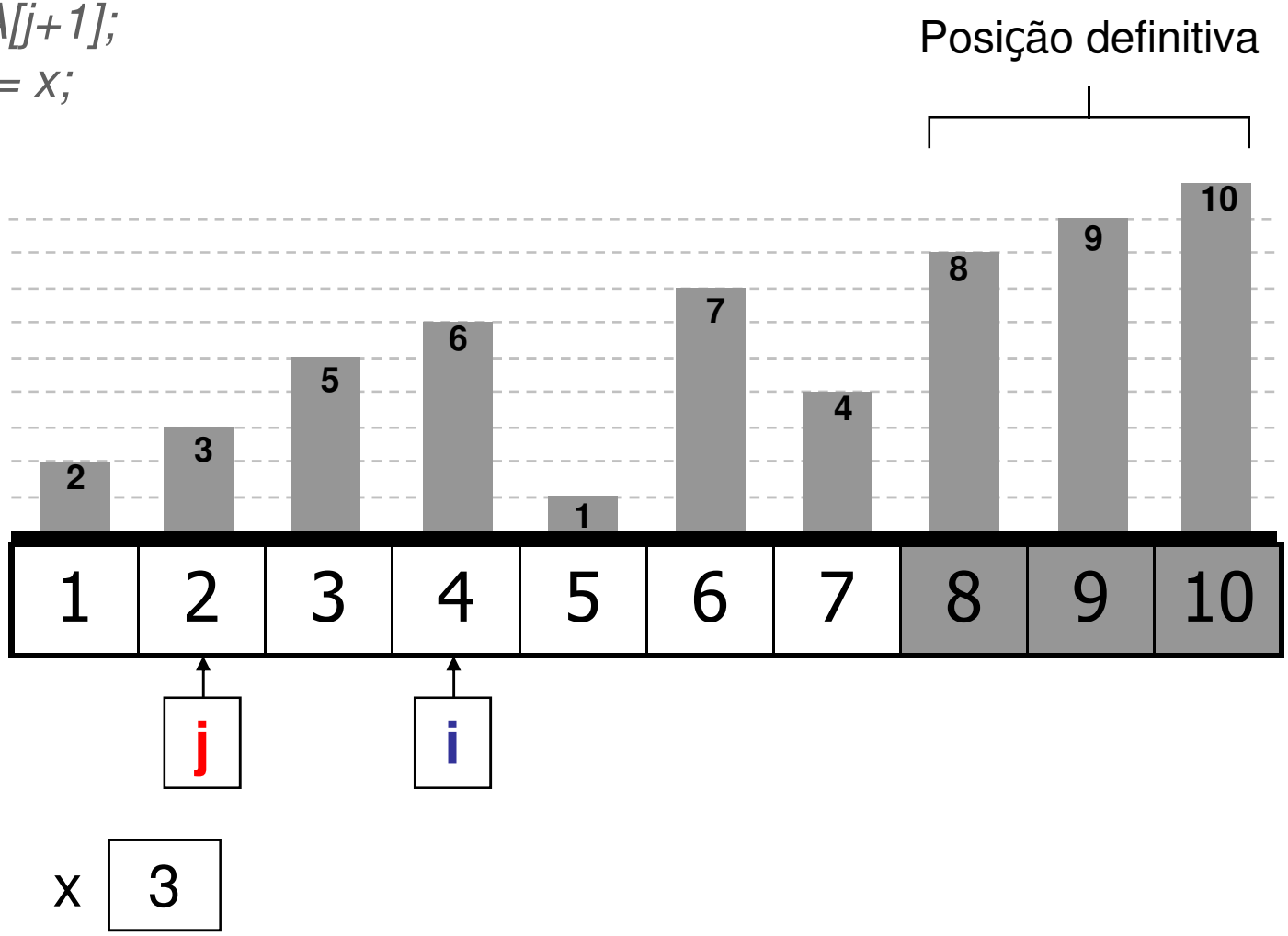
```
for i := 1 to n-1 do ←
  for j := 1 to n-i do ←      {varredura}
    if A[j] > A[j+1] then
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



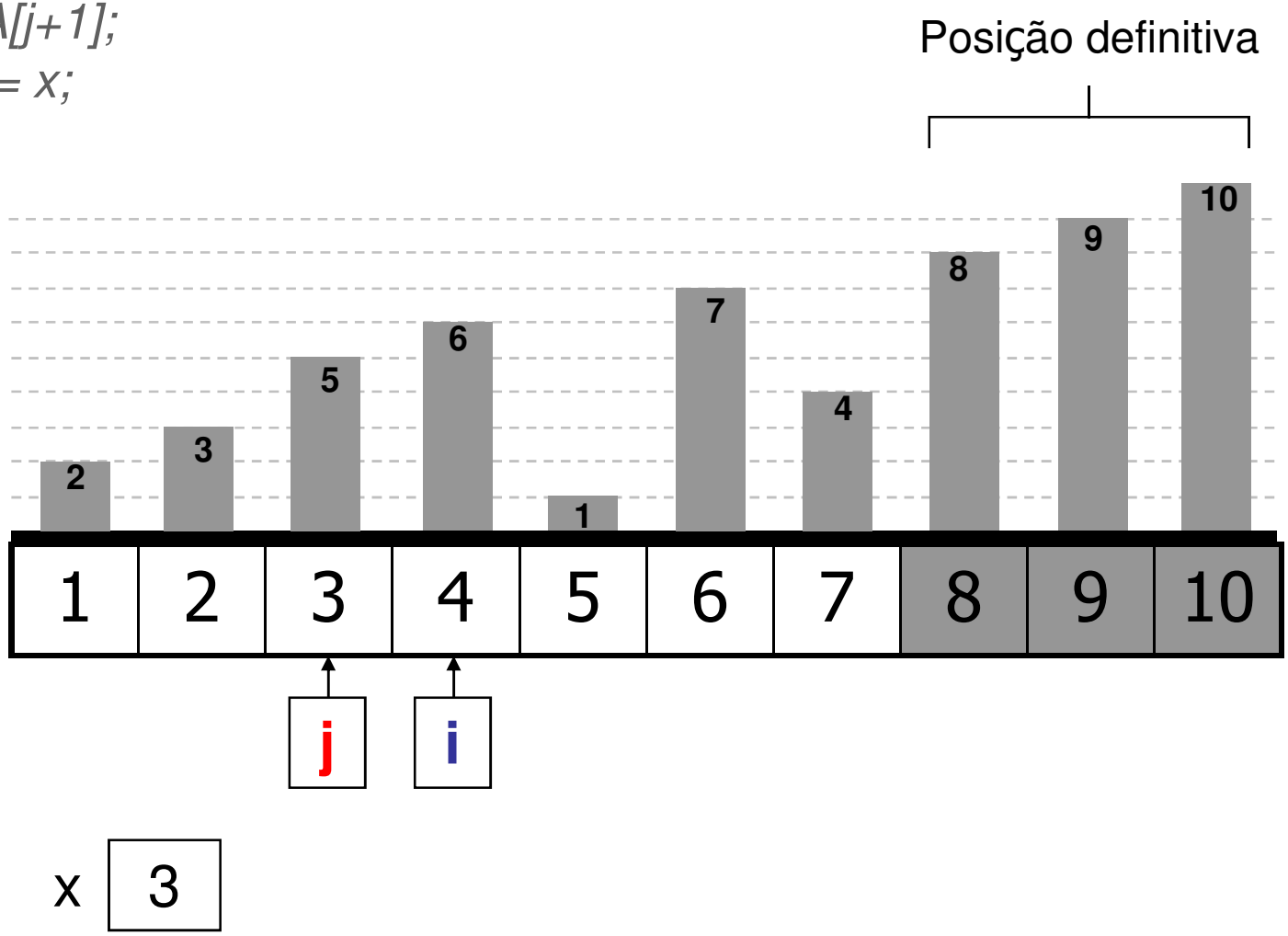
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];←
      A[j] := A[j+1];←
      A[j+1] := x;←
    End;
```



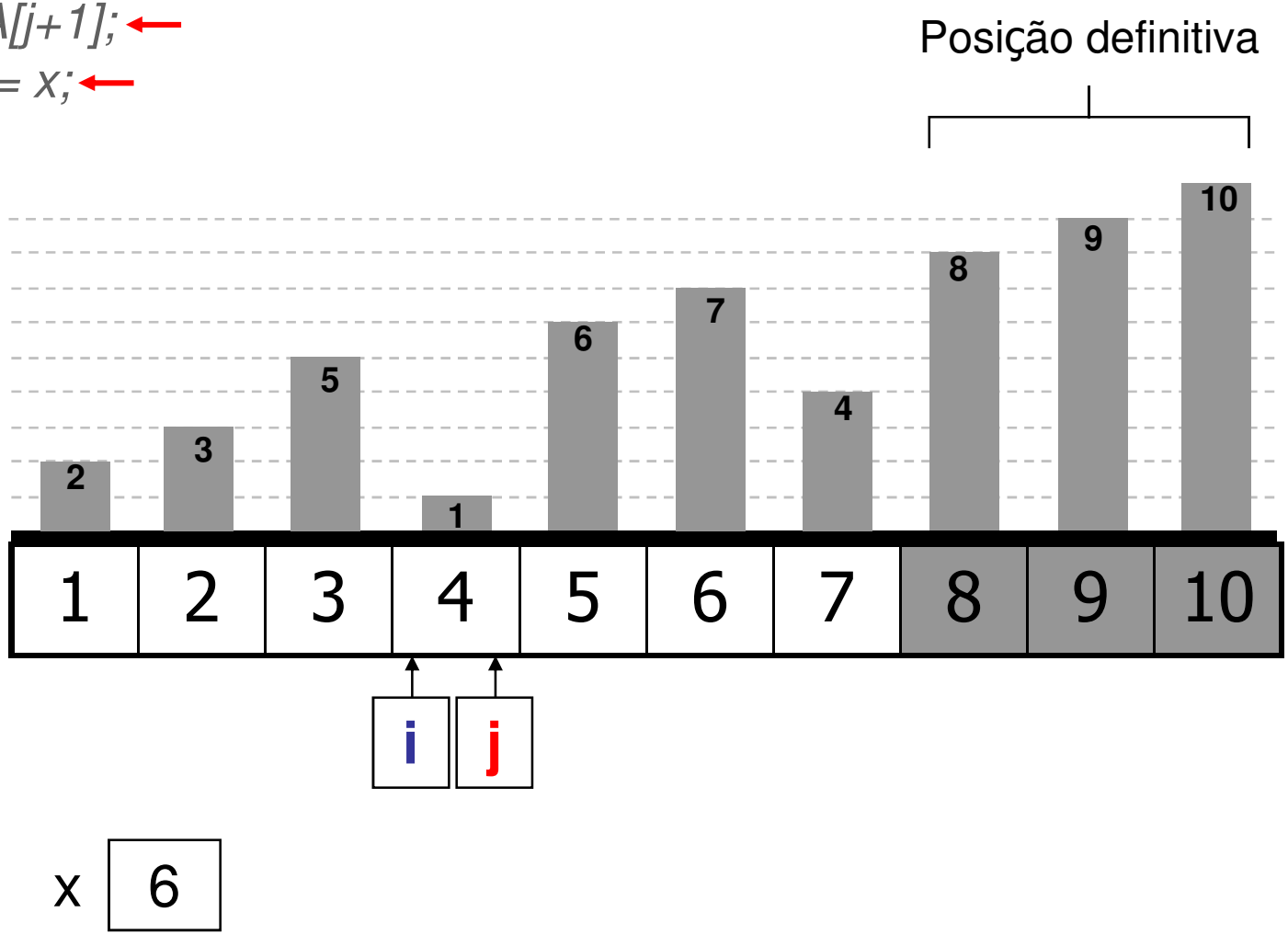
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



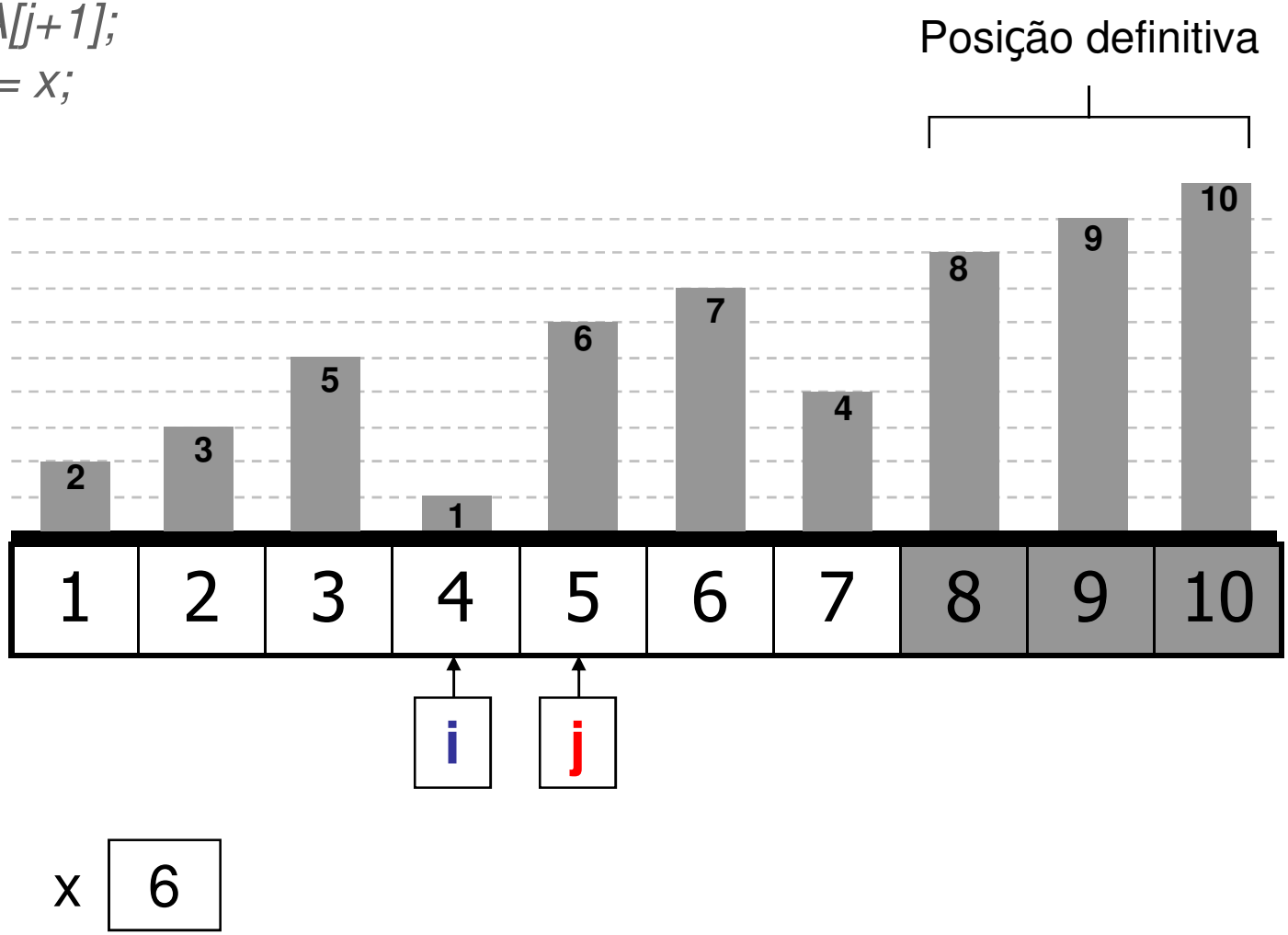
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



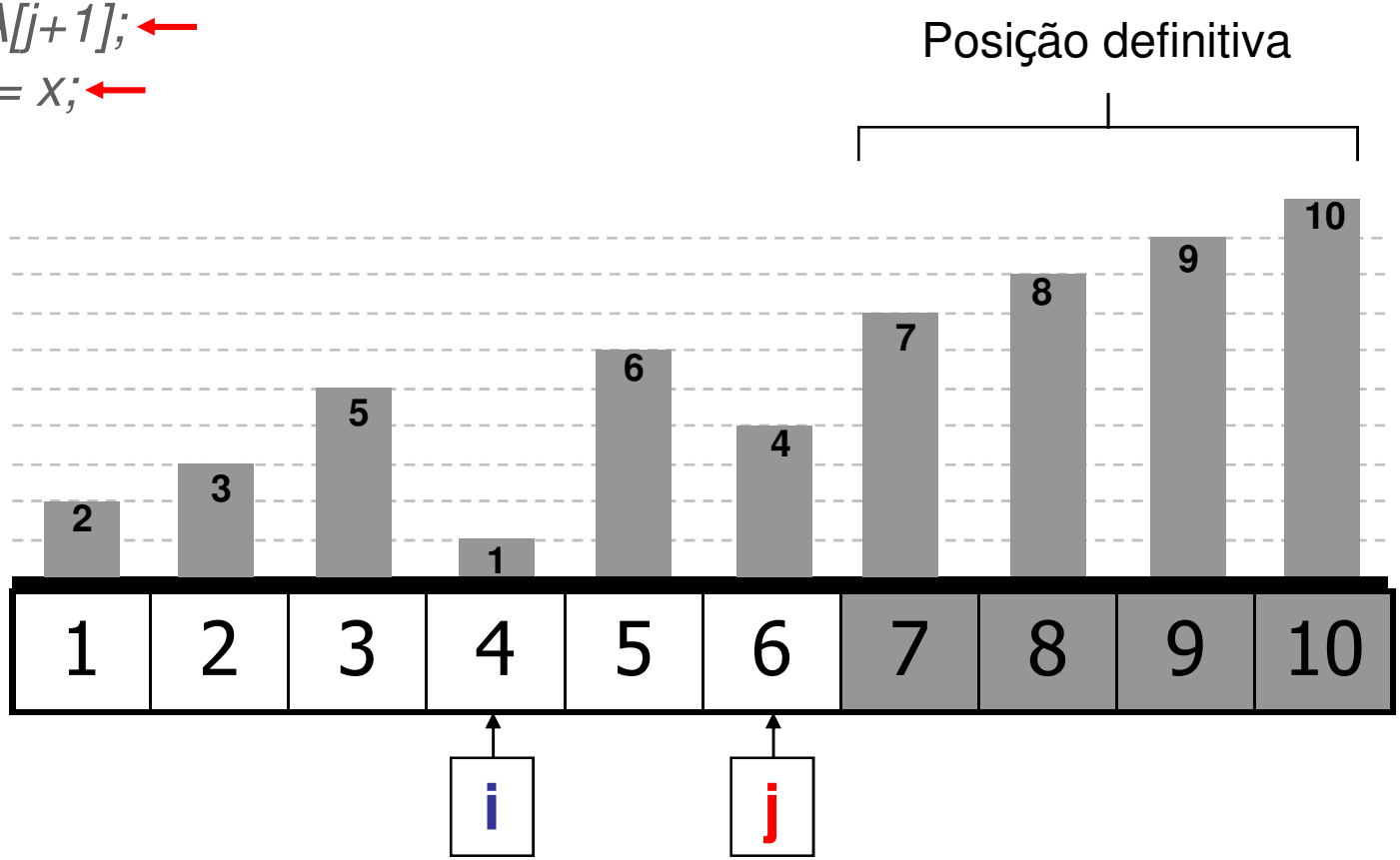
```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then ←
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



```
for i := 1 to n-1 do
  for j := 1 to n-i do          {varredura}
    if A[j] > A[j+1] then
     Begin
      x := A[j];
      A[j] := A[j+1];
      A[j+1] := x;
    End;
```



x | 7

Final da quarta varredura.

Ordenação Interna

- Bibliografia recomendada:
  - Ziviani, Nivio. Projeto de Algoritmos, Editora Pioneira;
  - Salvetti, Dirceu Douglas & Barbosa, Lisbete Madsen. Algoritmos, Makron Books;
  - Wirth, Niklaus. Algortimos e Estruturas de Dados, Editora LTC;