Quick Sort

2012

Algoritmul

QuickSort ca Divide and Conquer

```
procedure QS(Left, Right)
  if (Right - Left + 1) = 1 then
   // (Right - Left + 1) reprezintă dimensiunea vectorului
     return;                                          // nu facem nimic
  else
      // Următorul apel corespunde pasului Divide.
     Partition(Left,Right,q);                  // Returnează pivotul în q.
      // Pivotul este pus la locul său final.
     QS(Left, q-1);                                   // 1 apel recursiv
     QS(q+1, Right);                                  // 2 apel recursiv
      // Pentru pasul Combine nu trebuie să facem nimic.
  endif
endproc
```

Procedura Partition

Pentru a sorta vectorul $A[1..n]$ apelul principal va fi `QS(1,n);`

Scurtă animație a algoritmului

quicksort

Despre partiții la Quick Sort

Partiția Hoare

Procedura Partition

Procedura de partiționare Hoare pe subvectorul $A[Left..Right]$ este următoarea:

Partiția Hoare

Procedura Partition

Procedura de partiționare Hoare pe subvectorul $A[Left..Right]$ este următoarea:

```
procedure Partition(Left, Right)
    // iLeft=indicele curent pentru parcurgerea (2a)
    // iRight=indicele curent pentru parcurgerea (2b)
   iLeft:=Left; iRight=Right;          // inițializarea indicilor curenți pentru parcurgeri
   x:=A[(Left+Right)div 2];                    // algerea pivotului în poziție mediană
   repeat                                                                  // partiția
      while A[iLeft] < x do                                                  // (2a)
         iLeft:= iLeft+1;
      endwhile
      while A[iRight] > x do                                                 // (2b)
         iRight:= iRight - 1;
      endwhile
      if iLeft ≤ iRight then                                                 // (3)
         Interschimbă(A[iLeft], A[iRight]);
         iLeft:= iLeft+1;
         iRight:= iRight-1;
      endif
   until (iLeft > iRight)
endproc
```

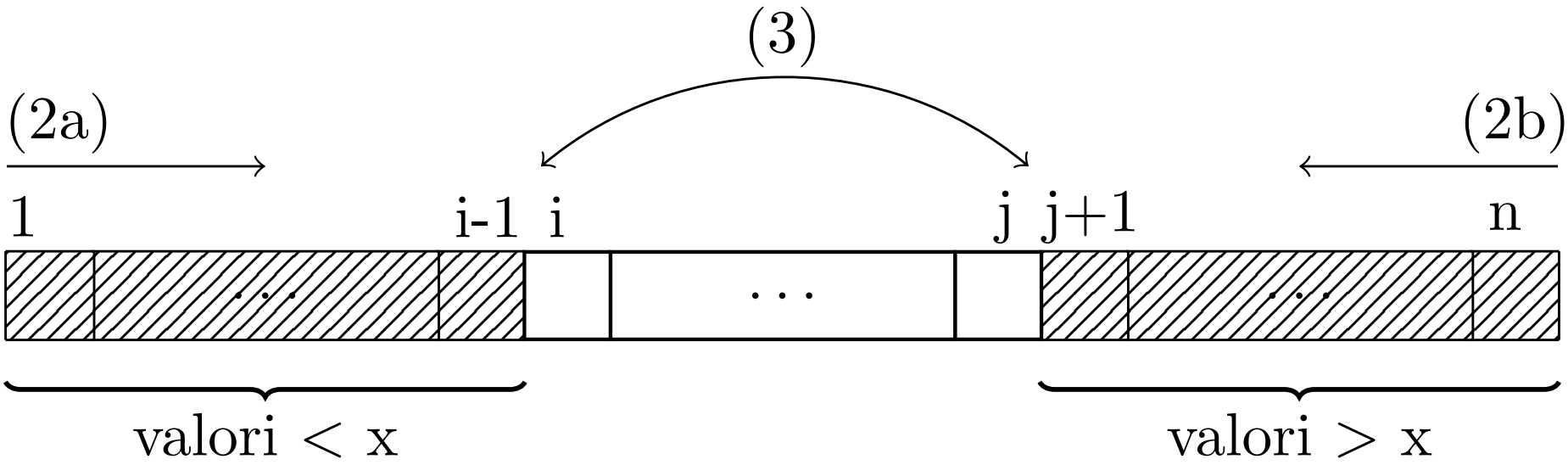


Figure : Procedura de partiționare.

Exemplu:

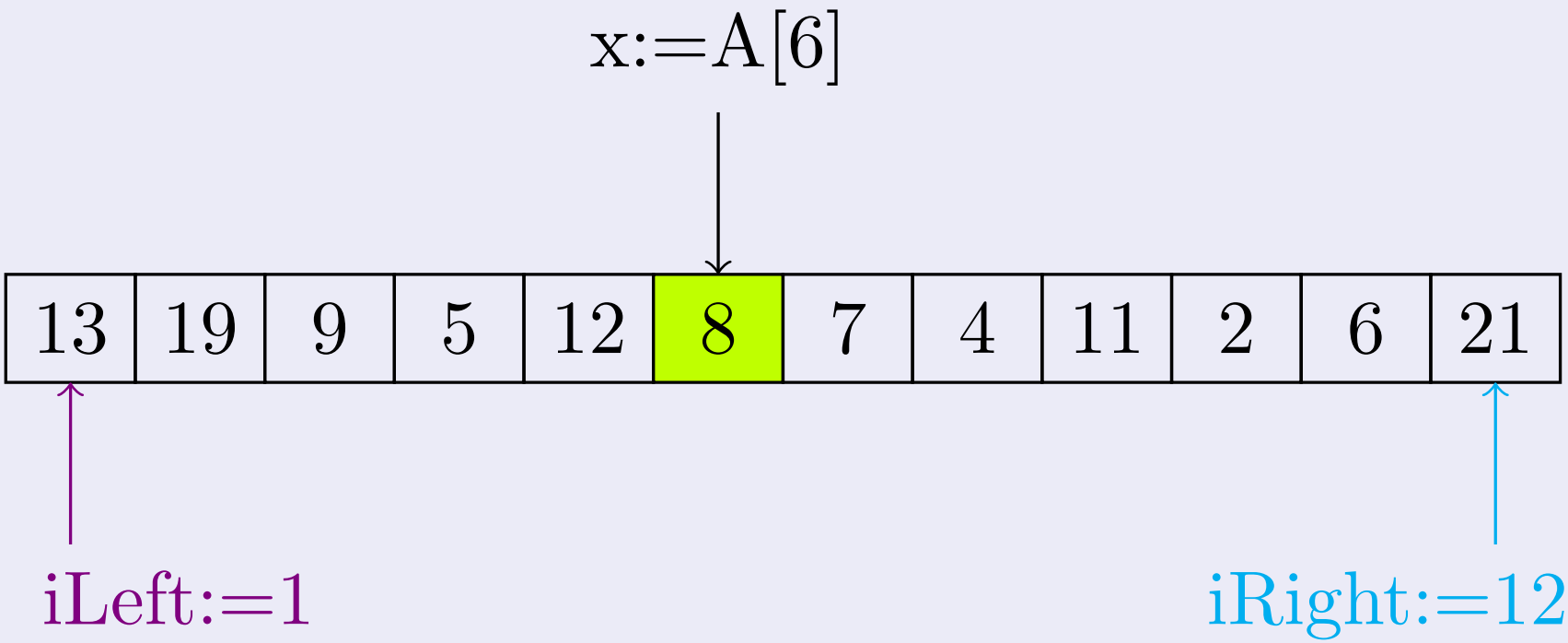


Figure : vectorul A inițial

```
x:=A[(Left+Right)div 2];
  iLeft:=1; iRight=12;
```

Exemplu:

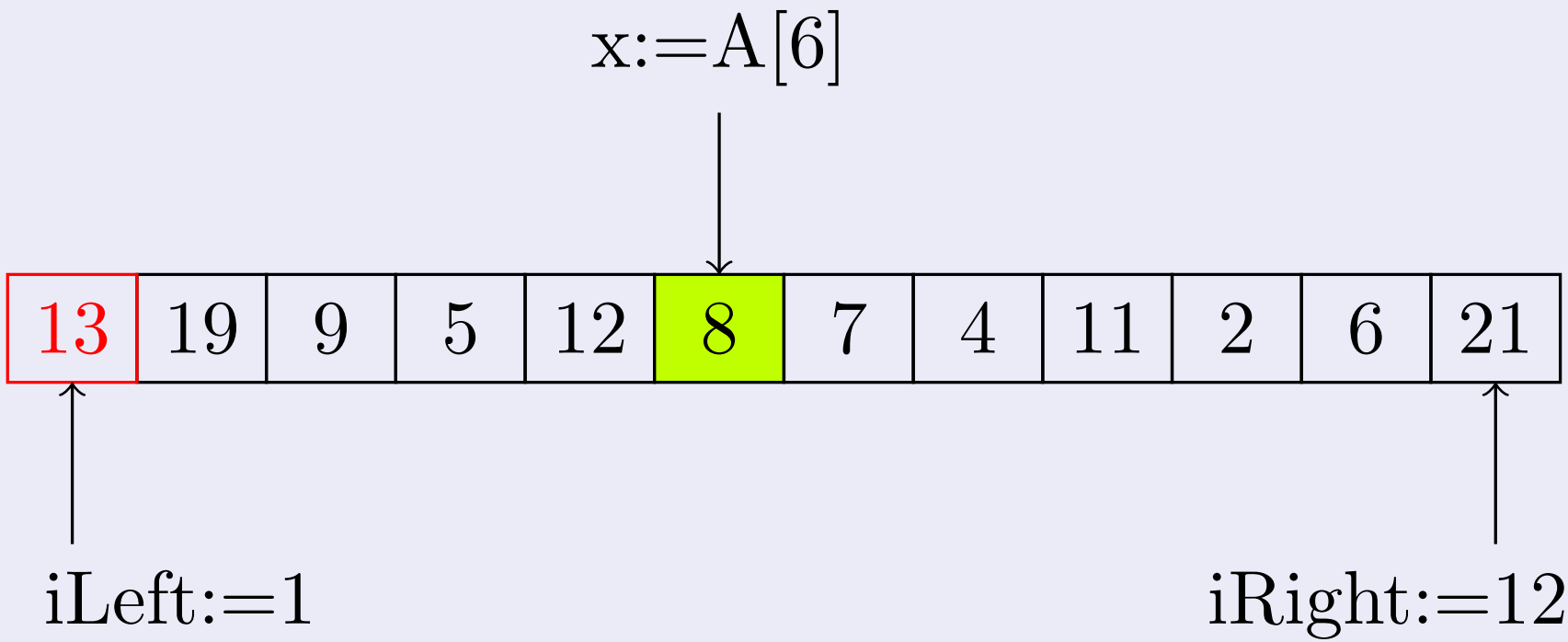


Figure : (2a)

`A[1]` $\geq$ `x`, deci nu putem avansa cu `iLeft`.

Exemplu:

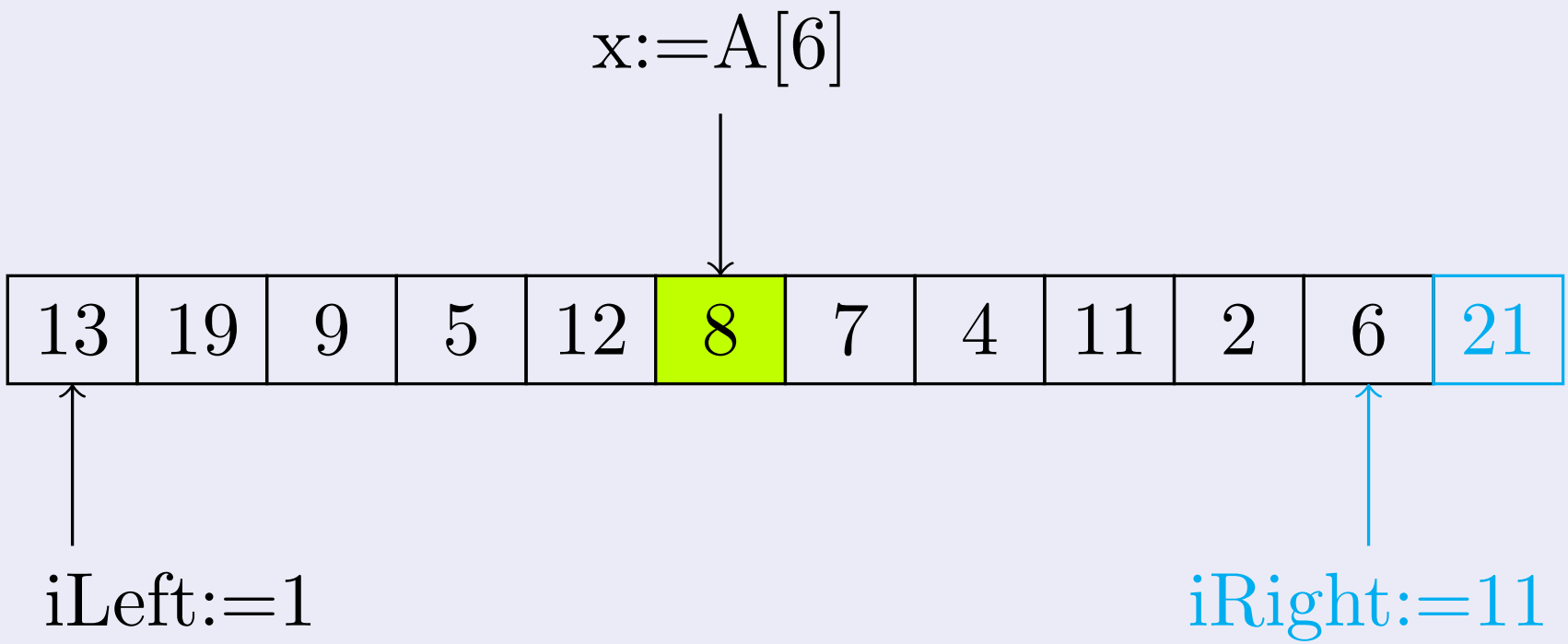


Figure : (2b)

`A[12] > x`, deci `iRight := iRight-1`.

Exemplu:

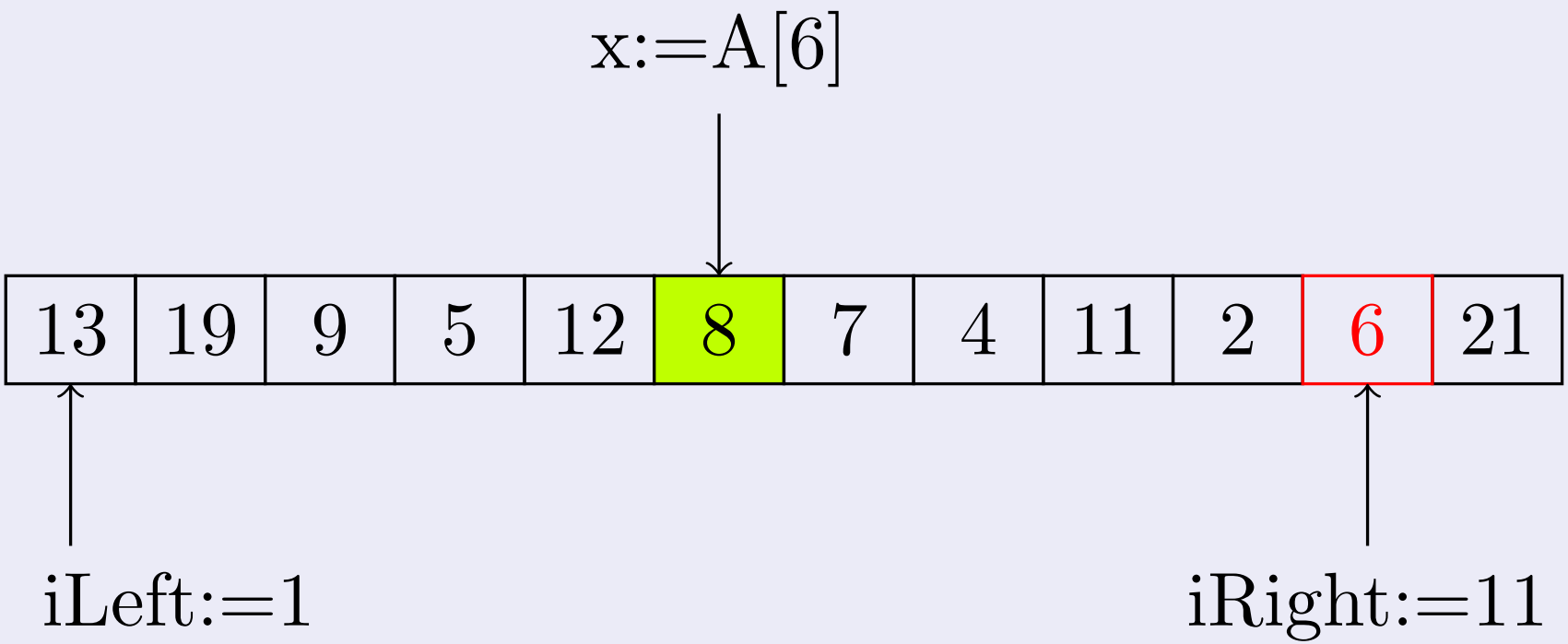


Figure : (2b)

`A[11]` $\leq$ `x`, deci `iRight` nu mai avansează.

Exemplu:

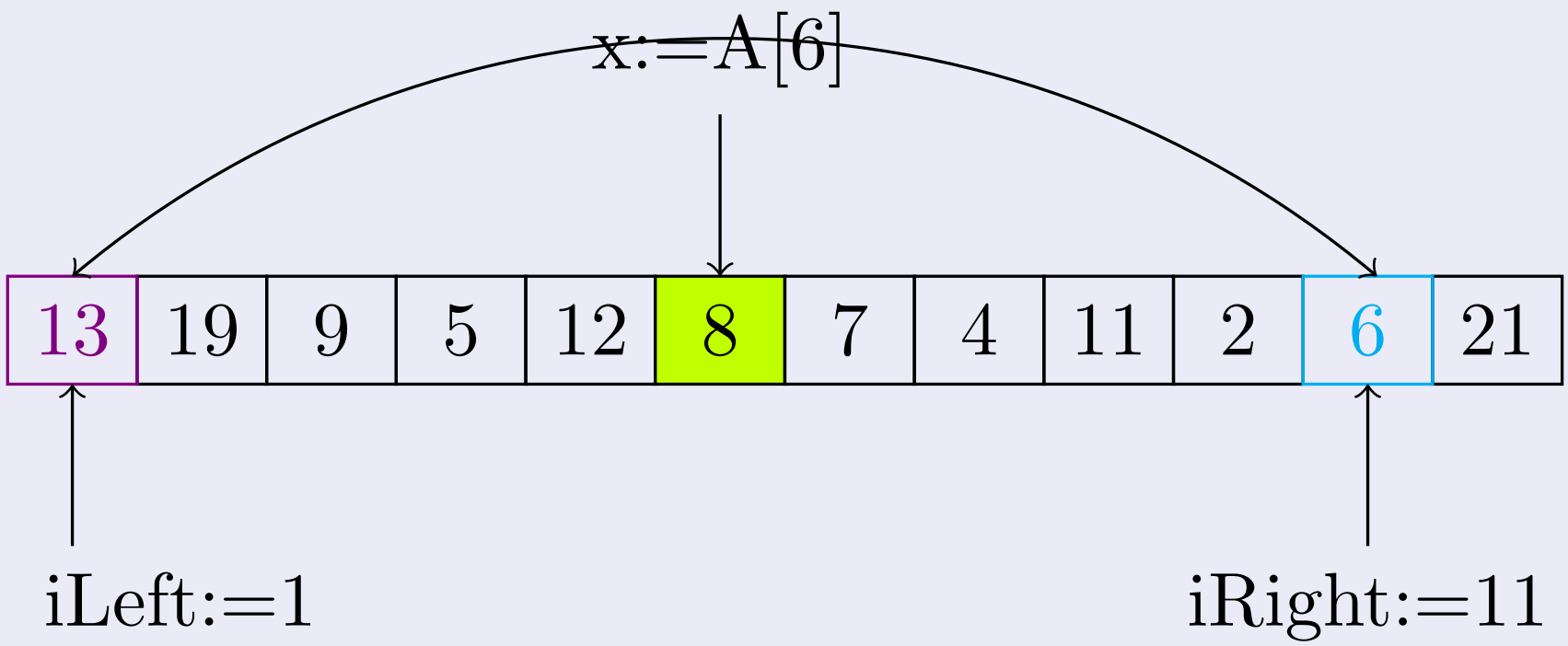


Figure : (3)

Trebuie să interschimbăm `A[1]` cu `A[11]`.

Exemplu:

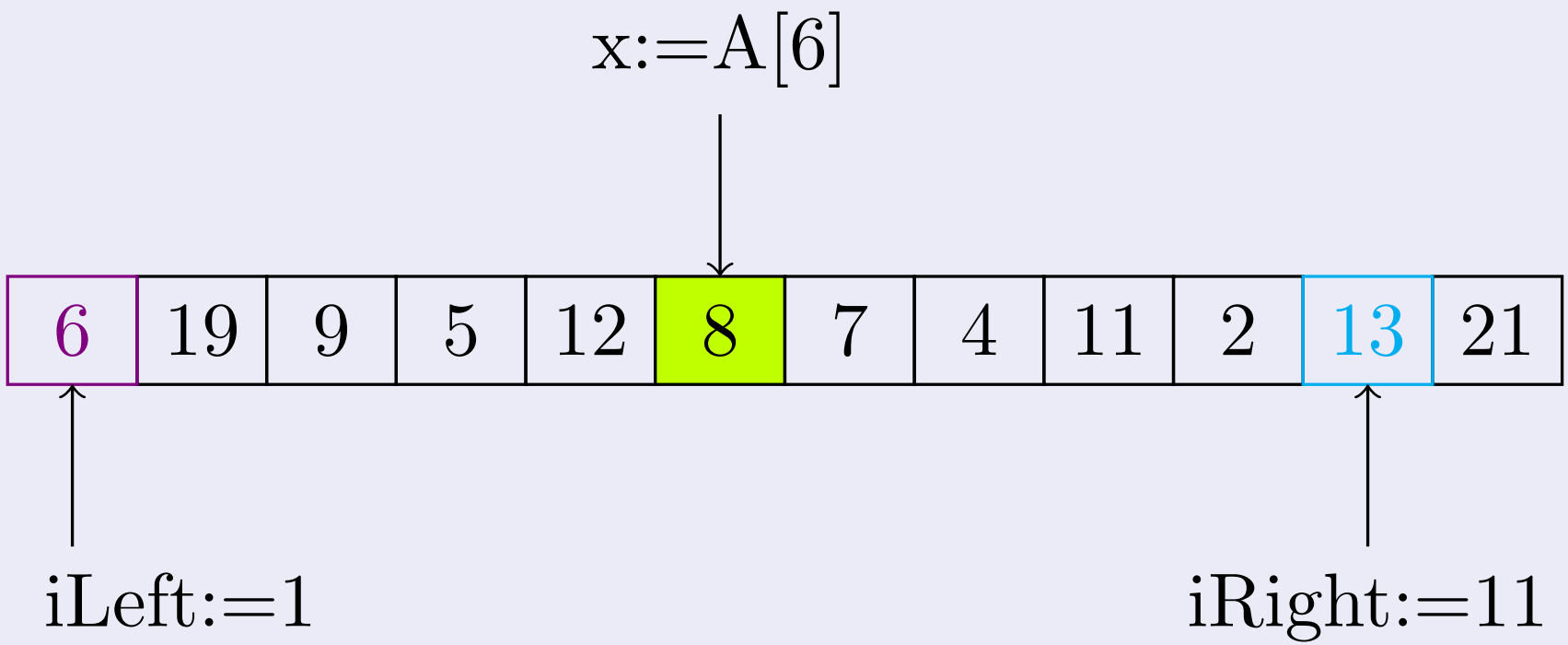


Figure : (3)

Vectorul obținut după interschimbare.

Exemplu:

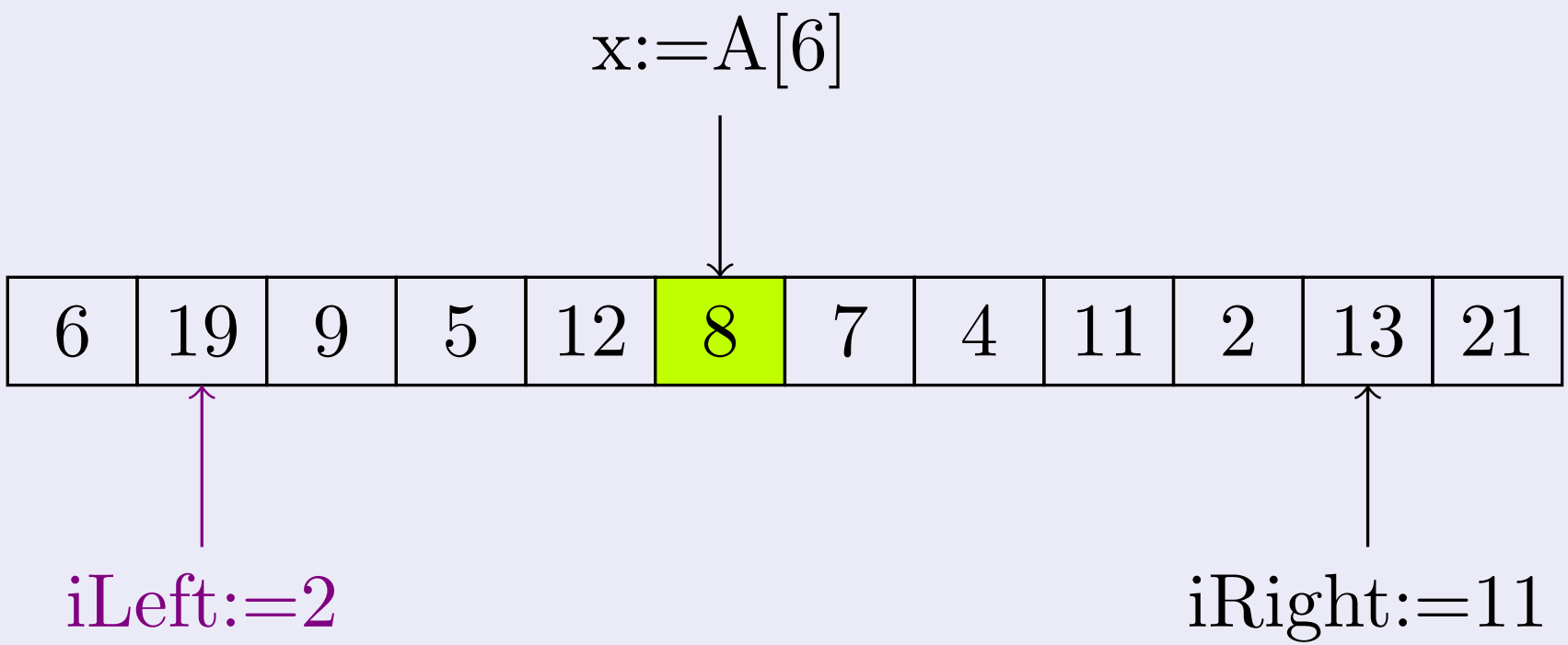


Figure : (3)

```
iLeft:= iLeft+1;
```

Exemplu:

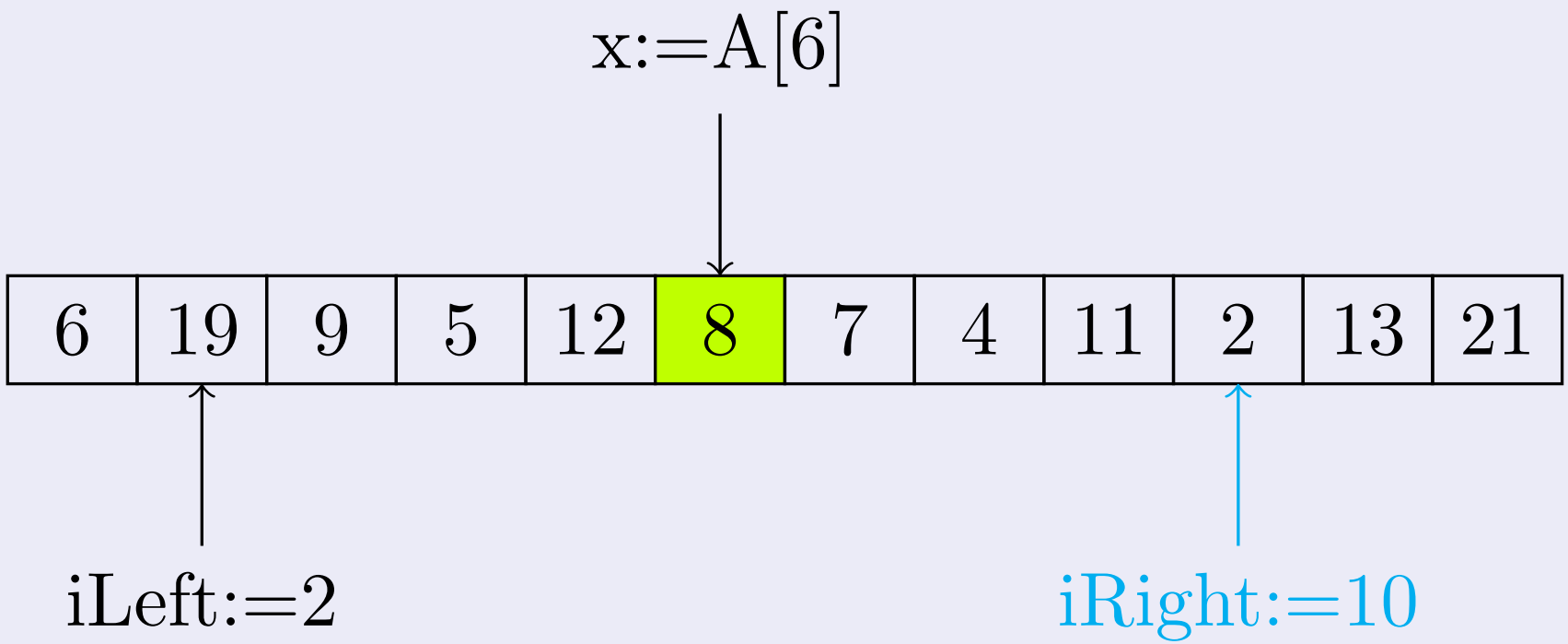


Figure : (3)

```
iRight:= iRight-1;
```

Exemplu:

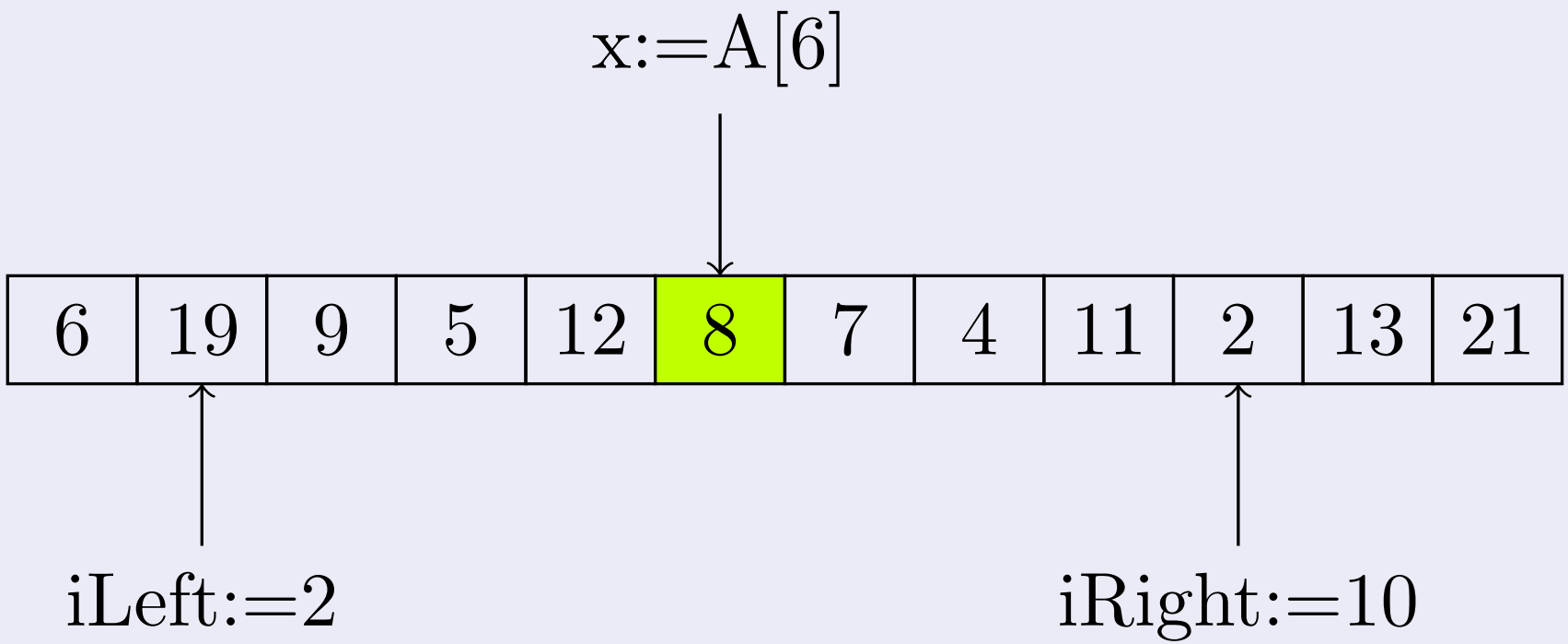


Figure : După prima iterație a ciclului `repeat`

Avem `2 = iLeft` $\leq$ `iRight = 10`, deci continuăm procedura de partiție.

Exemplu:

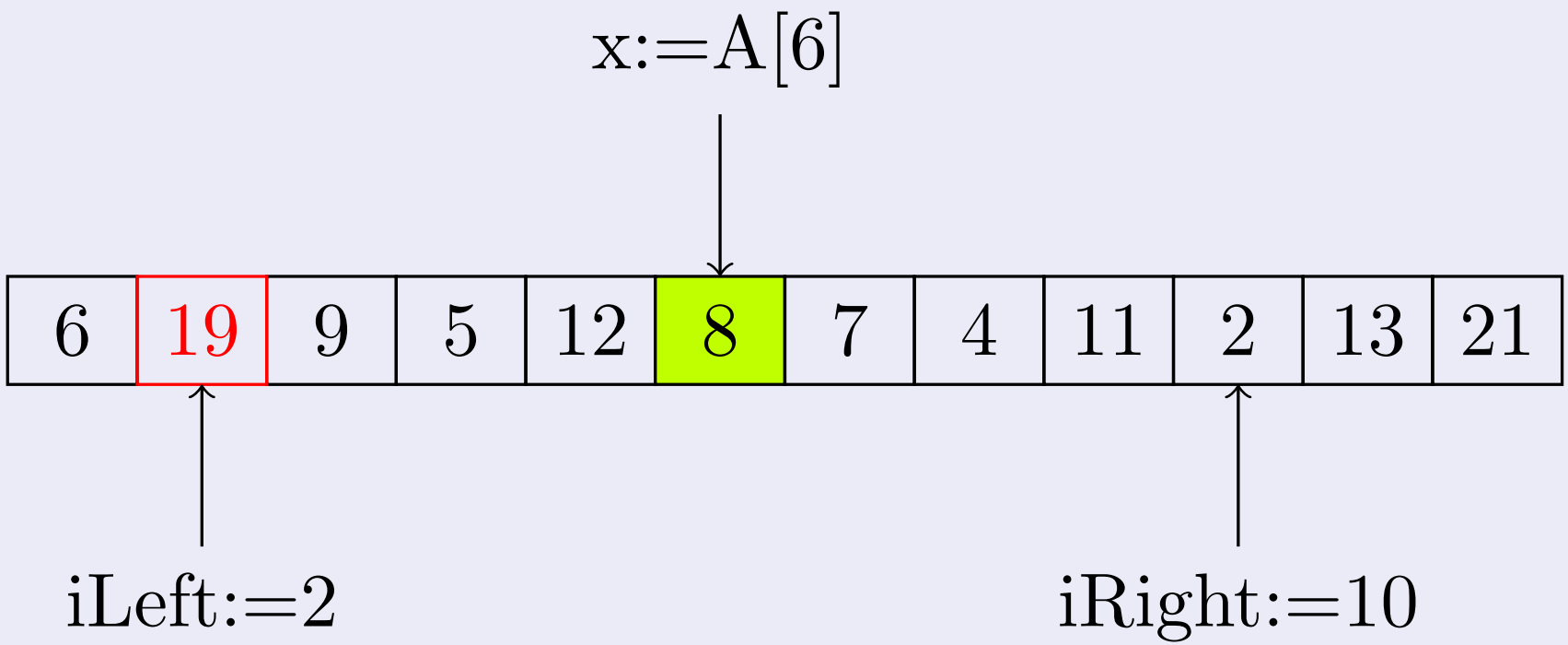


Figure : (2a)

`A[2]` $\geq$ `x`, deci nu putem avansa cu `iLeft`.

Exemplu:

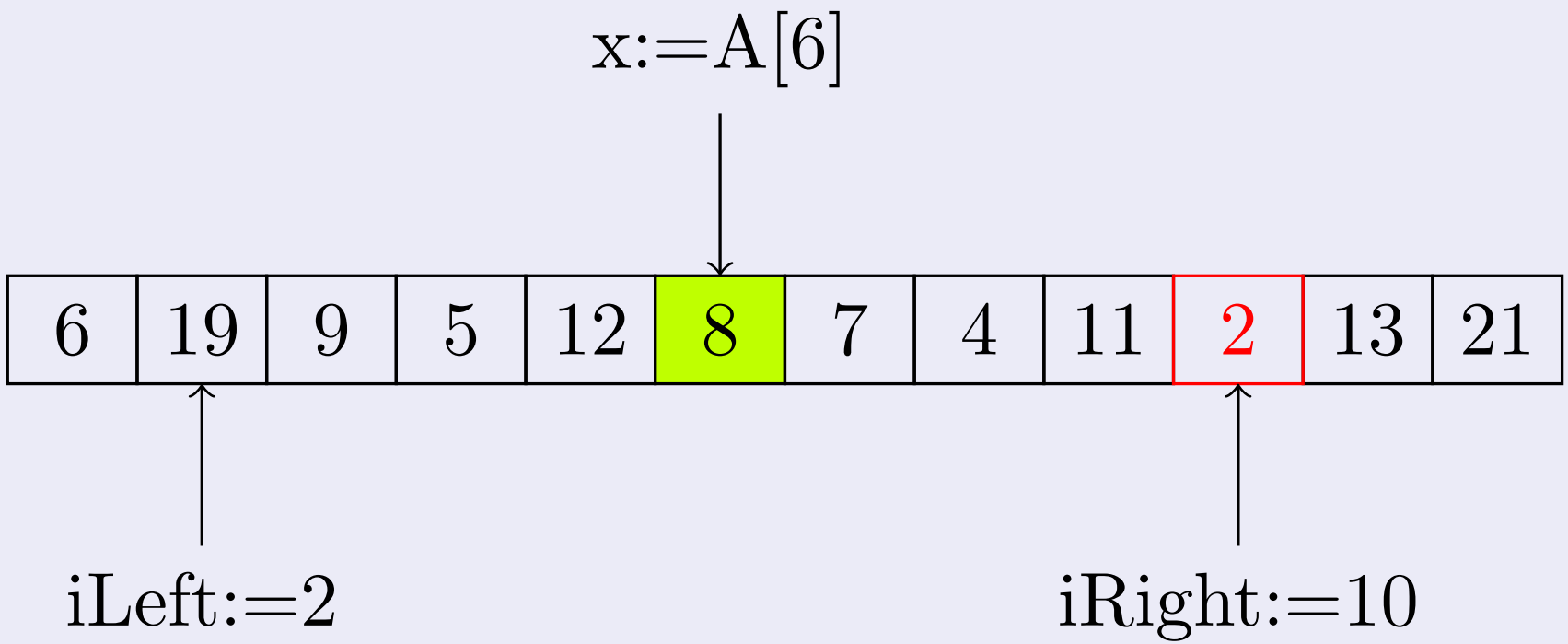


Figure : (2b)

`A[10]` $\leq$ `x`, deci nu putem avansa cu `iRight`.

Exemplu:

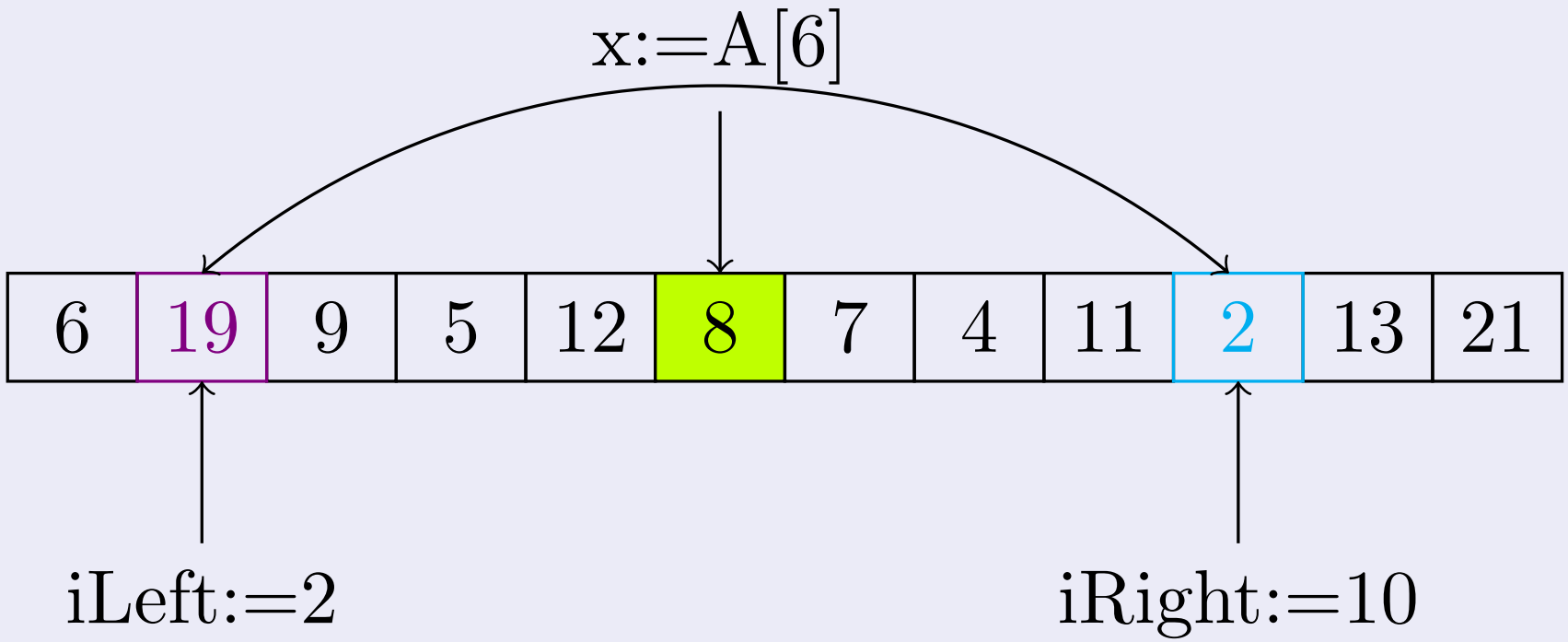


Figure : (3)

Trebuie să interschimbăm `A[2]` cu `A[10]`.

Exemplu:

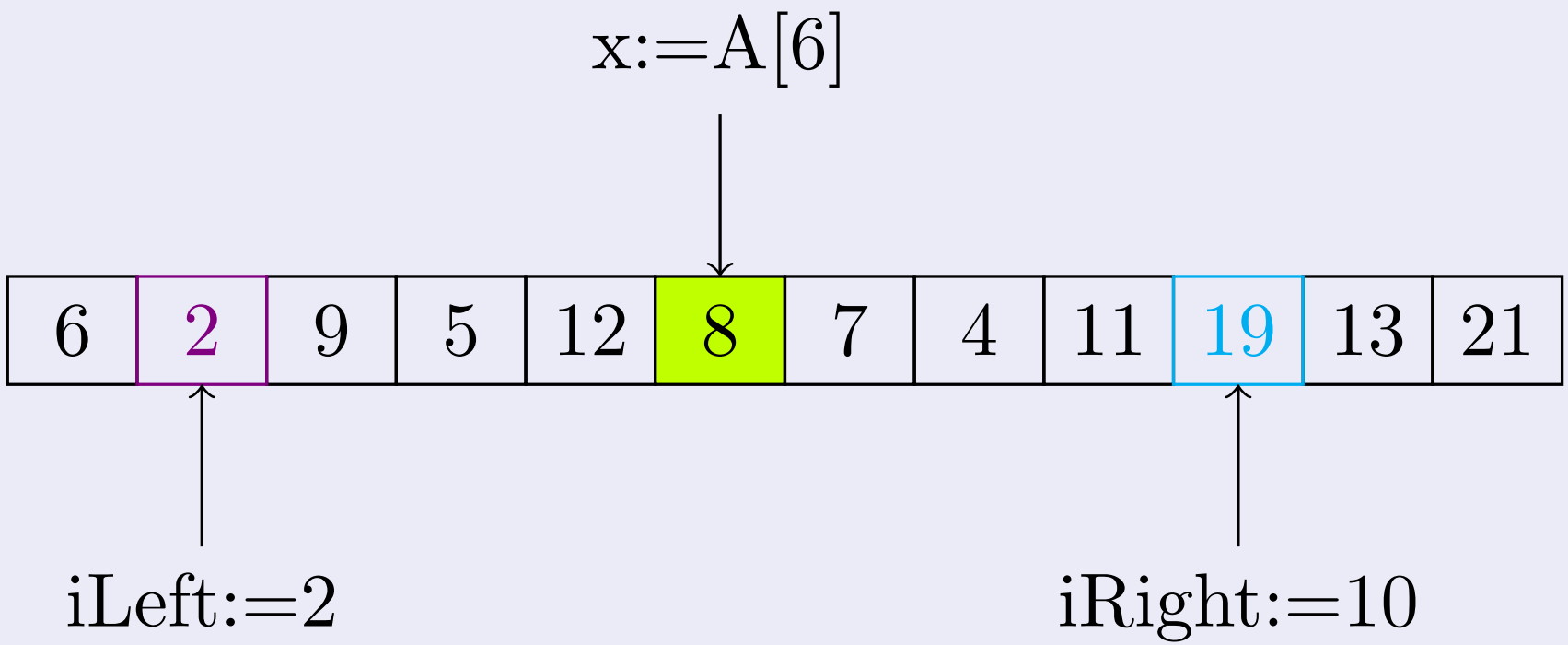


Figure : (3)

Vectorul obținut după interschimbare.

Exemplu:

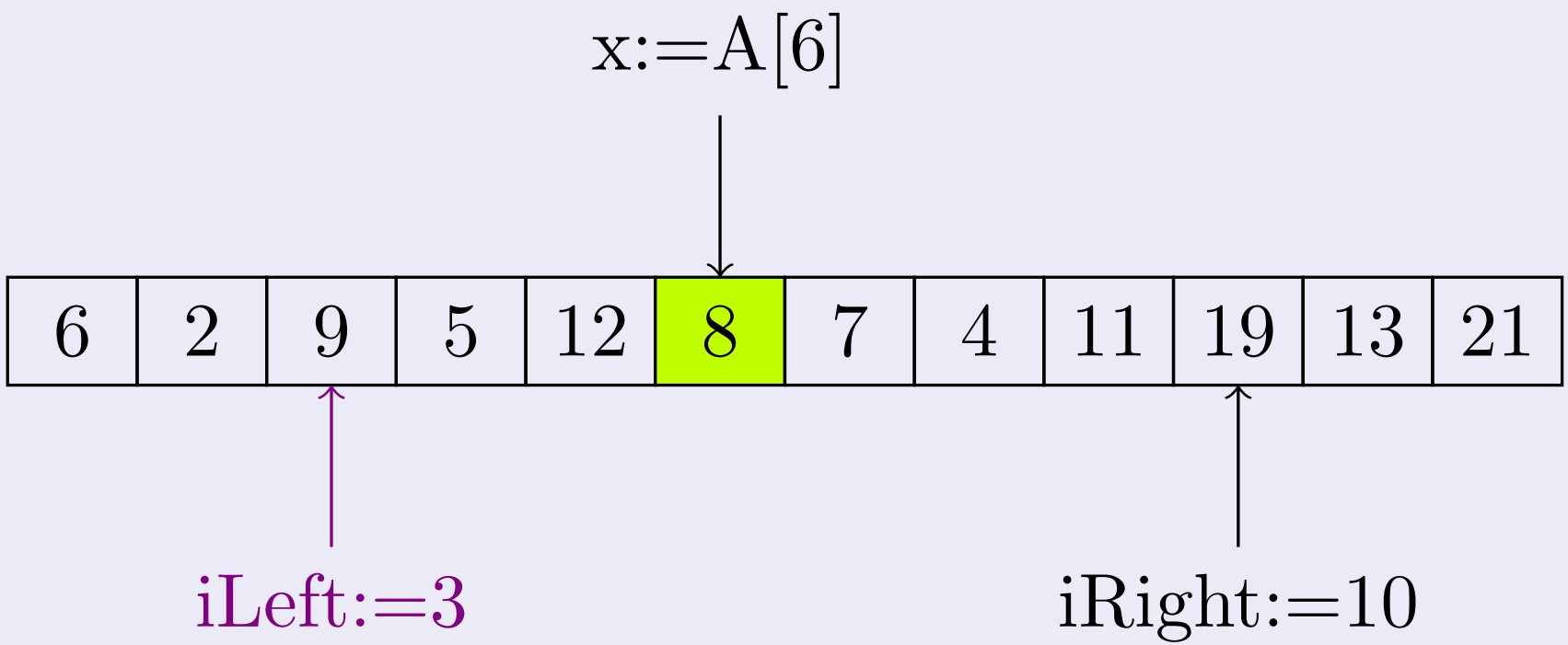


Figure : (3)

```
iLeft:= iLeft+1;
```

Exemplu:

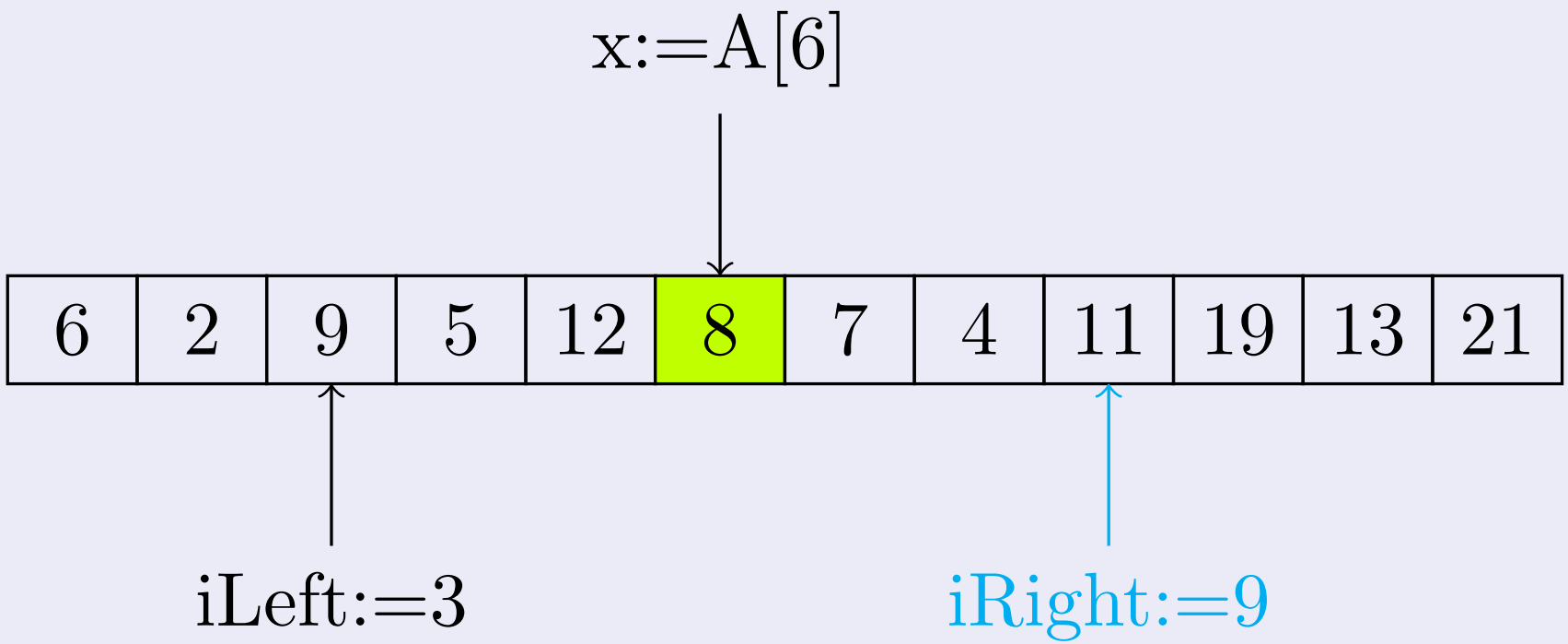


Figure : (3)

```
iRight:= iRight-1;
```

Exemplu:

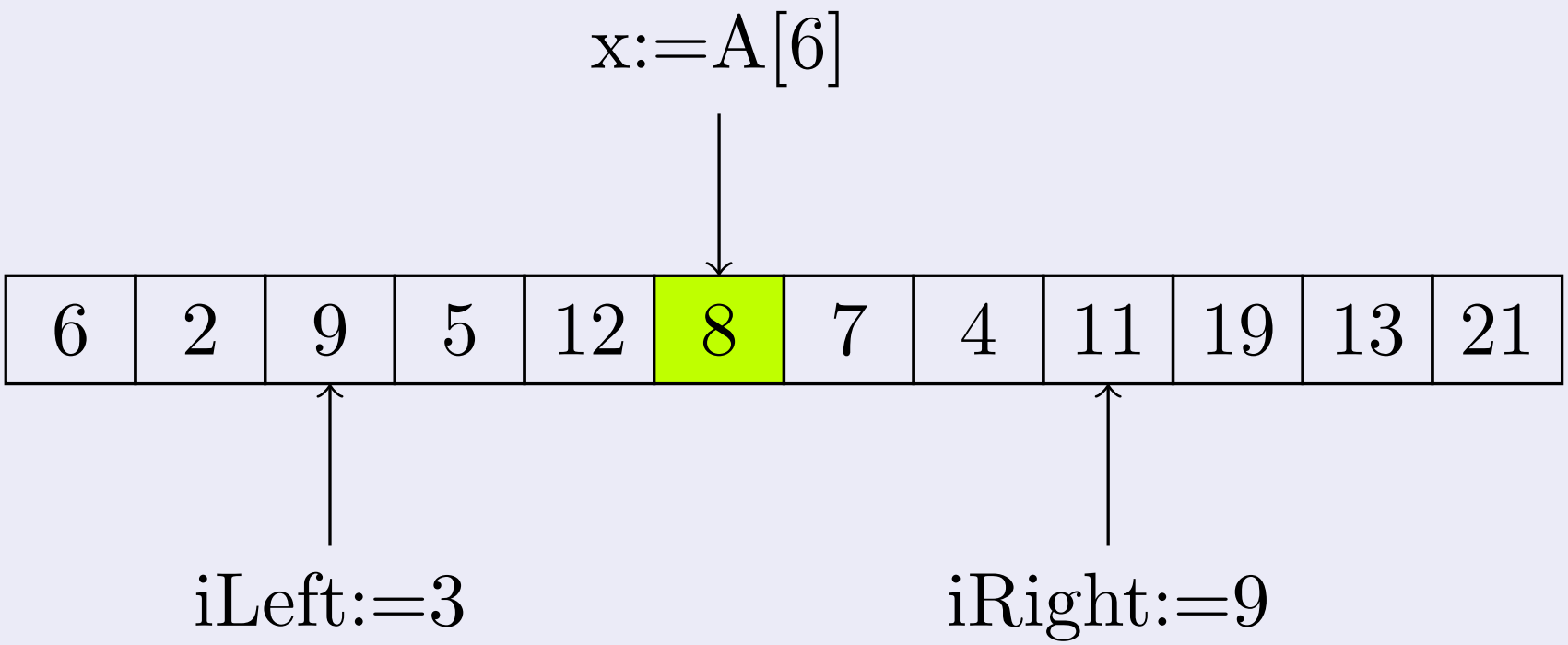


Figure : (Vectorul după a doua iterație a ciclului `repeat`)

Avem `3 = iLeft` $\leq$ `iRight = 9`, deci continuăm procedura de partiție.

Exemplu:

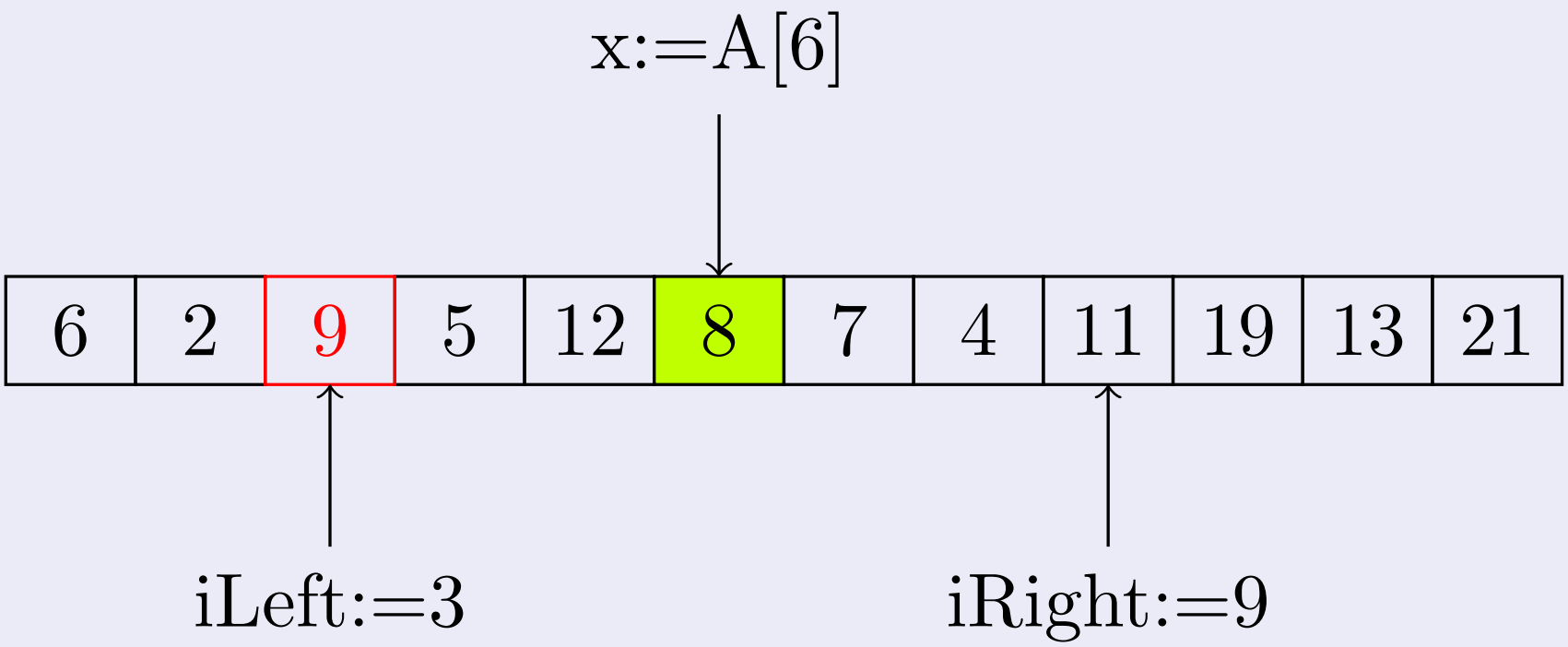


Figure : (2a)

`A[3]` $\geq$ `x`, deci nu putem avansa cu `iLeft`.

Exemplu:

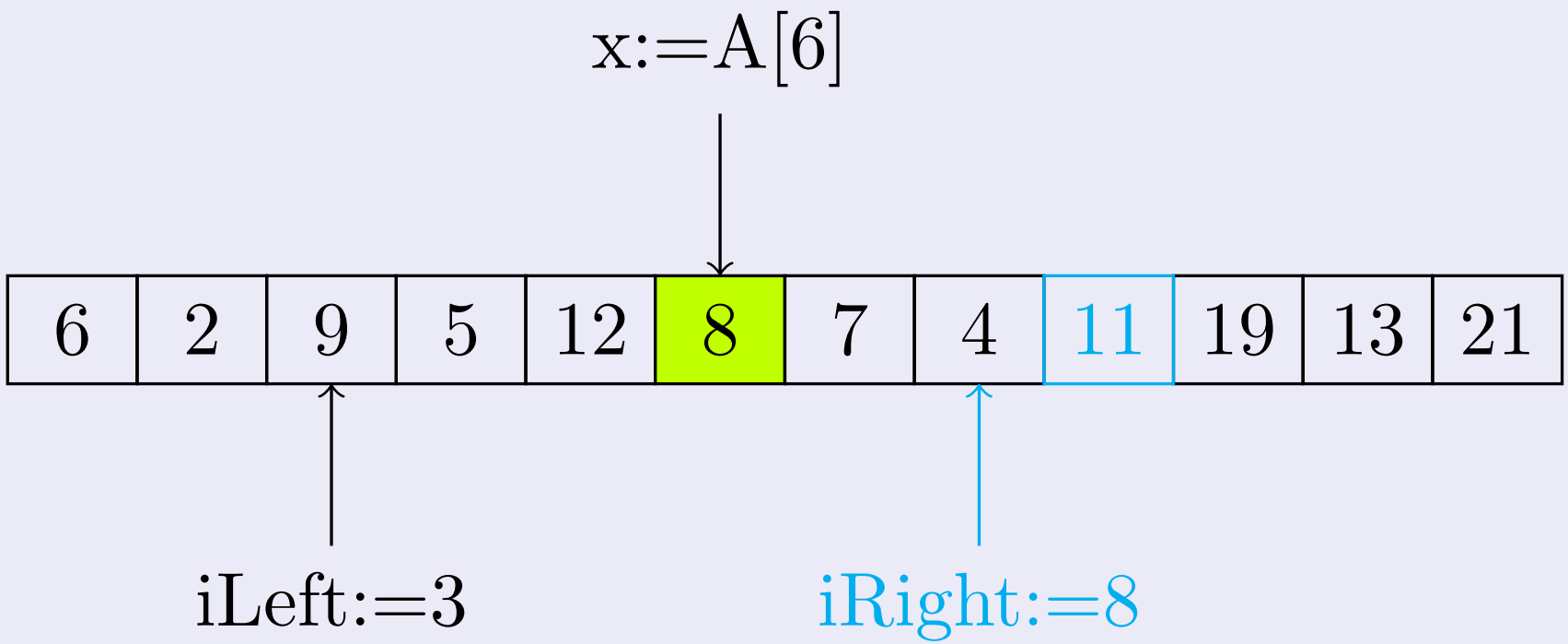


Figure : (2b)

`A[9] > x`, deci `iRight := iRight - 1`.

Exemplu:

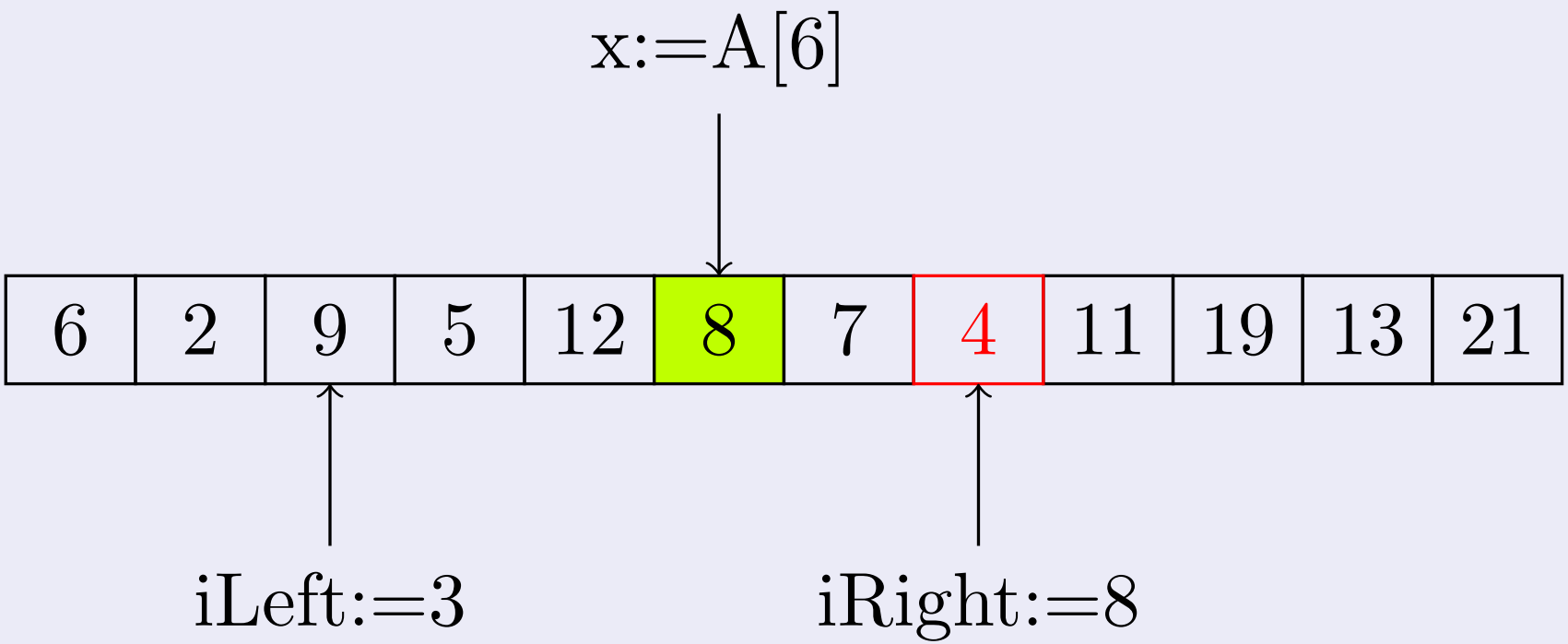


Figure : (2b)

`A[8]` $\leq$ `x`, deci nu putem avansa cu `iRight`.

Exemplu:

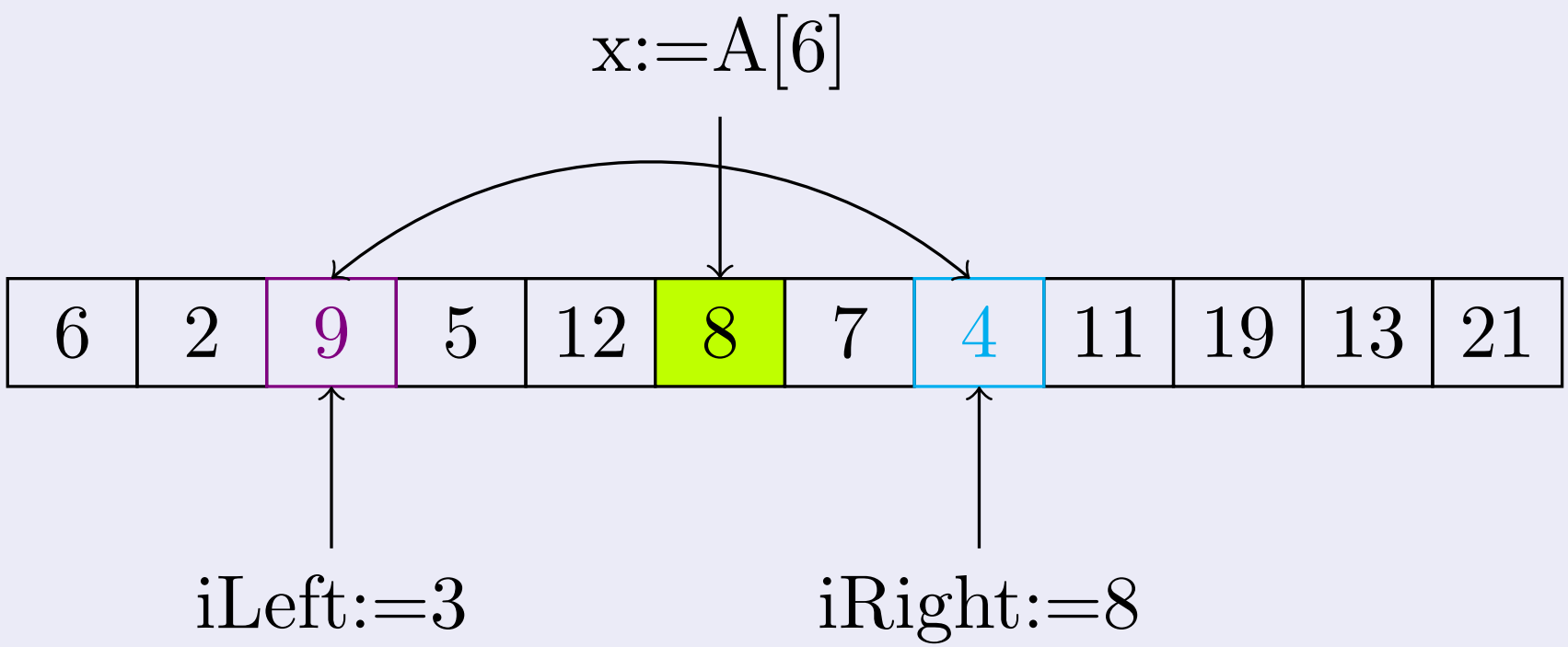


Figure : (2b)

Trebuie să interschimbăm `A[3]` cu `A[8]`.

Exemplu:

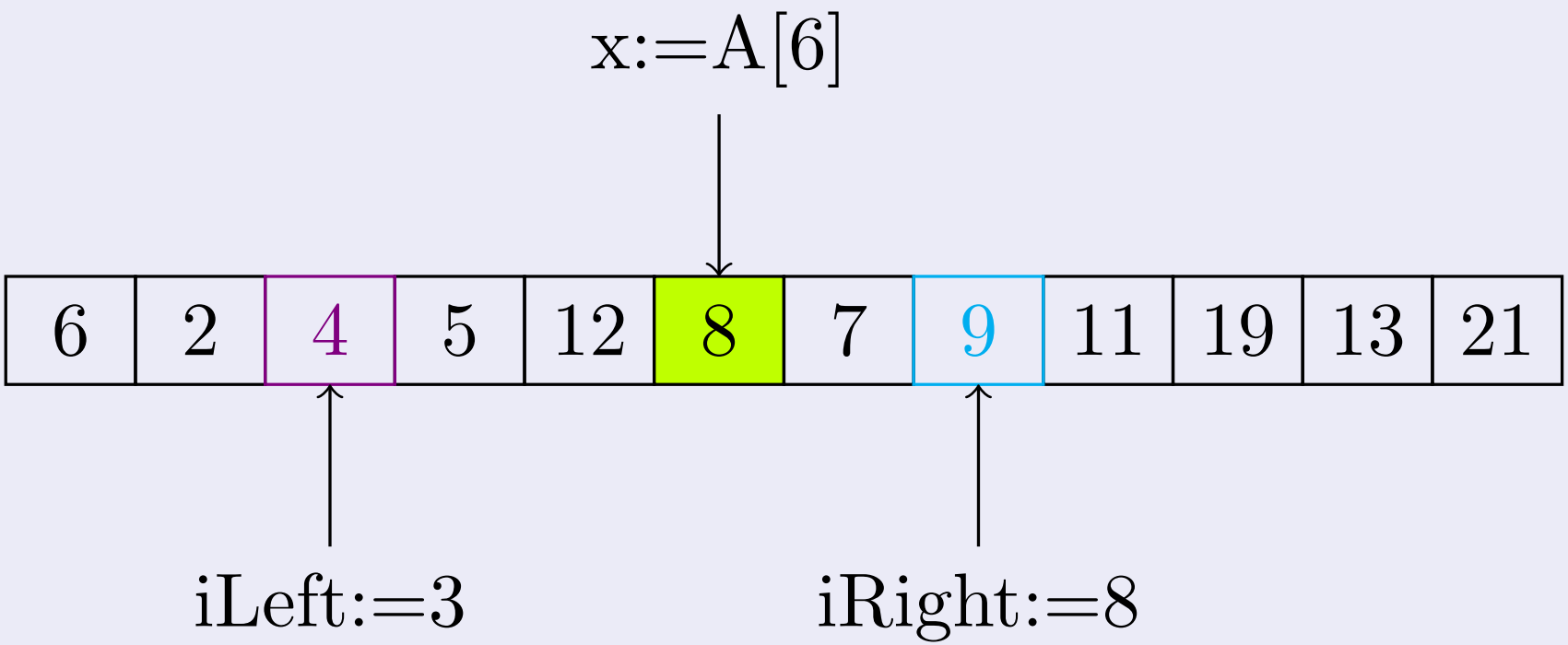


Figure : (3)

Vectorul obținut după interschimbare.

Exemplu:

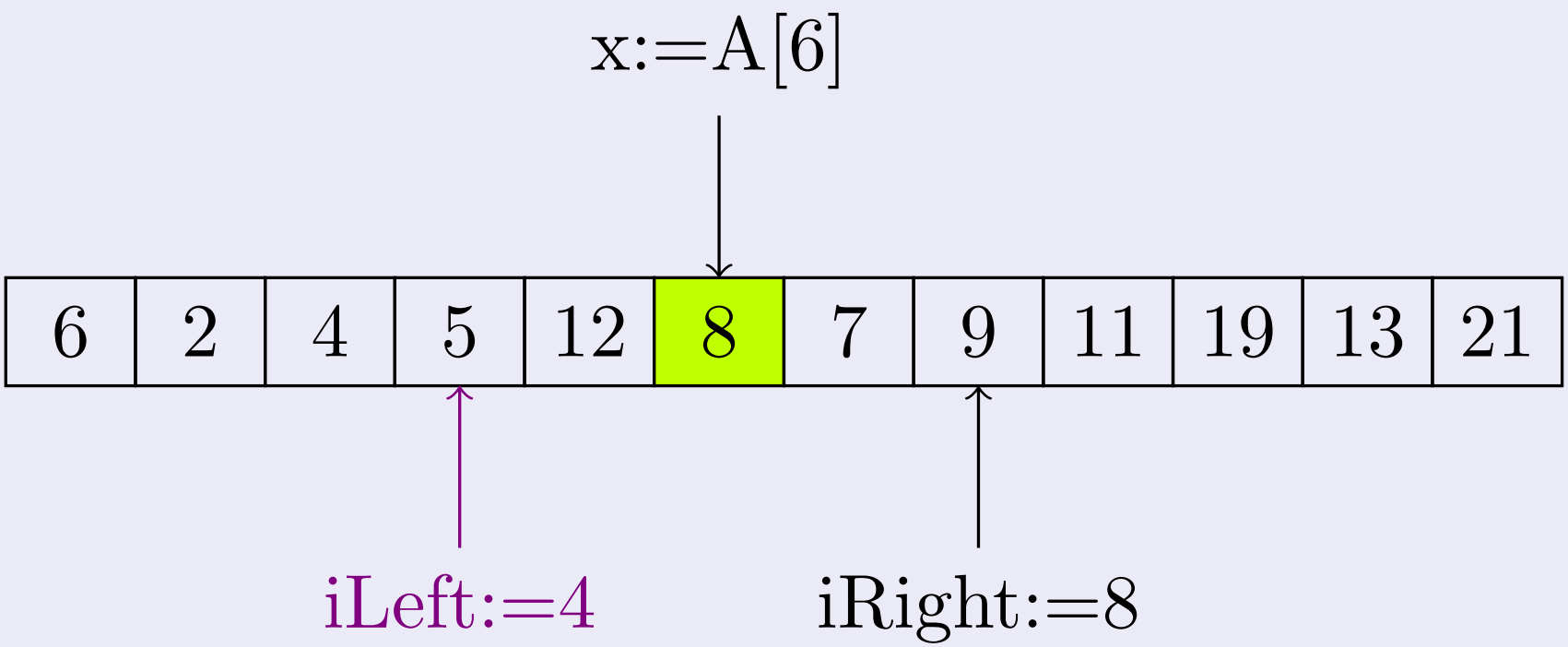


Figure : (3)

```
iLeft:= iLeft+1;
```

Exemplu:

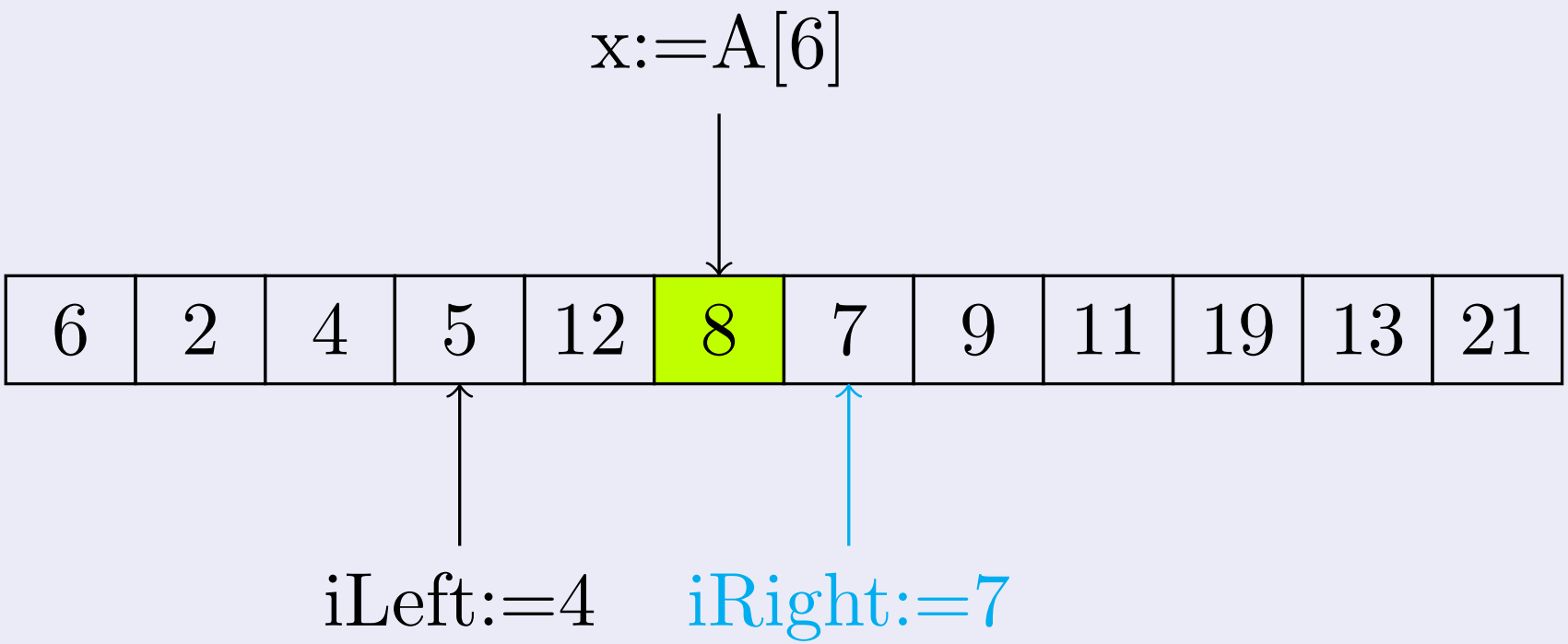


Figure : (3)

```
iRight:= iRight-1;
```

Exemplu:

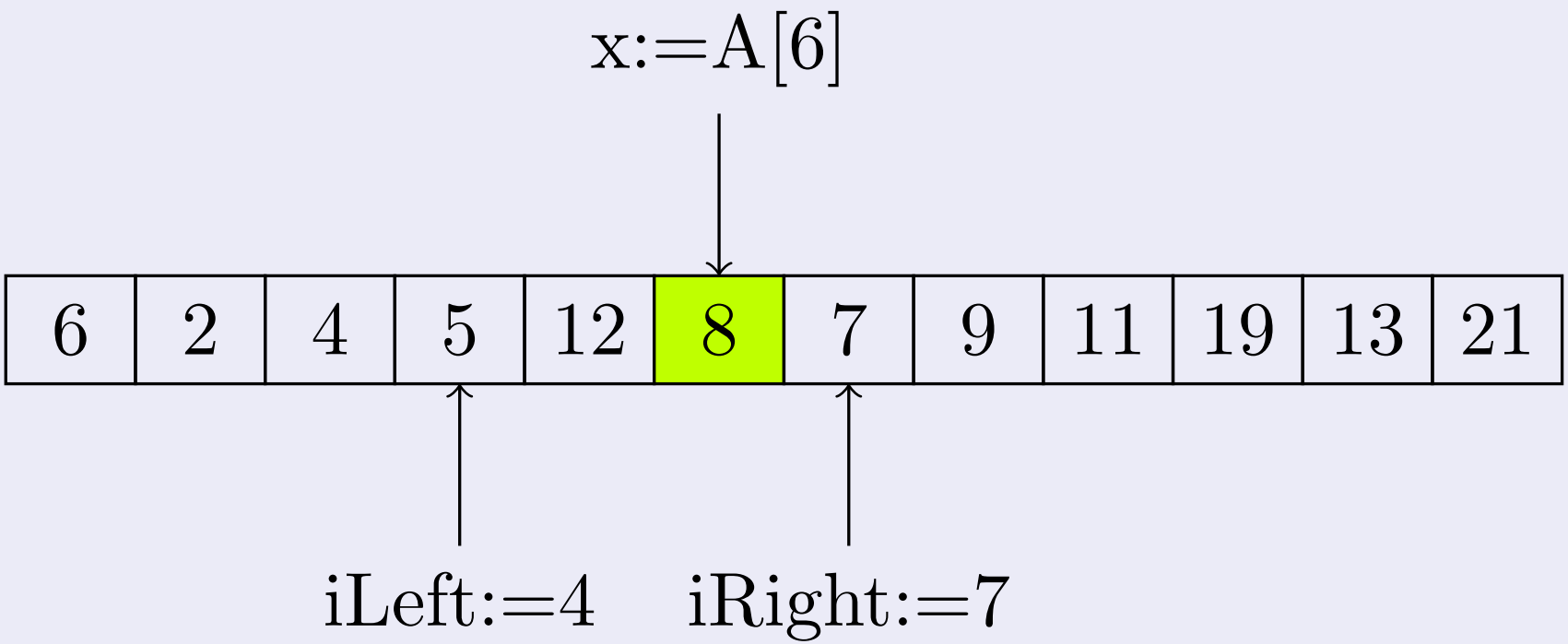


Figure : (Vectorul după a treia iterație a ciclului `repeat`)

Avem `4 = iLeft` $\leq$ `iRight = 7`, deci continuăm procedura de partiție.

Exemplu:

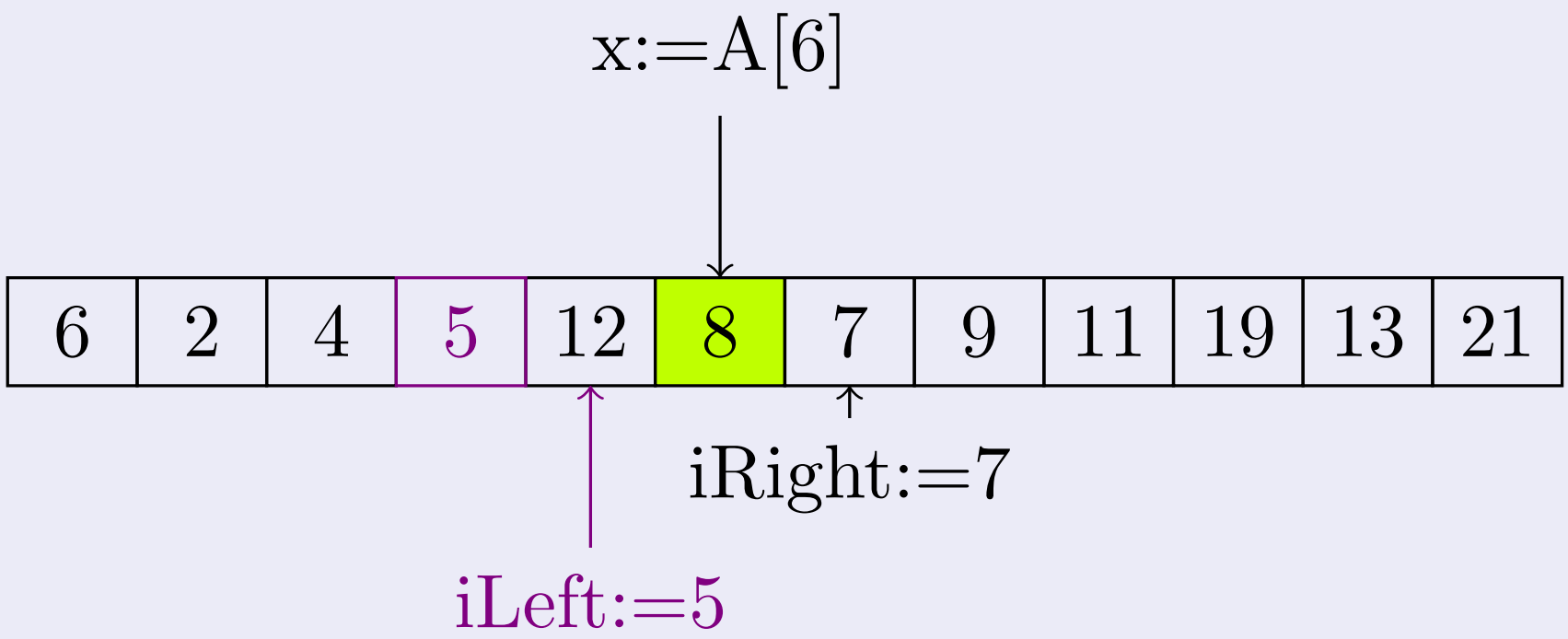


Figure : (2a)

`A[4] < x`, deci `iLeft := iLeft + 1`.

Exemplu:

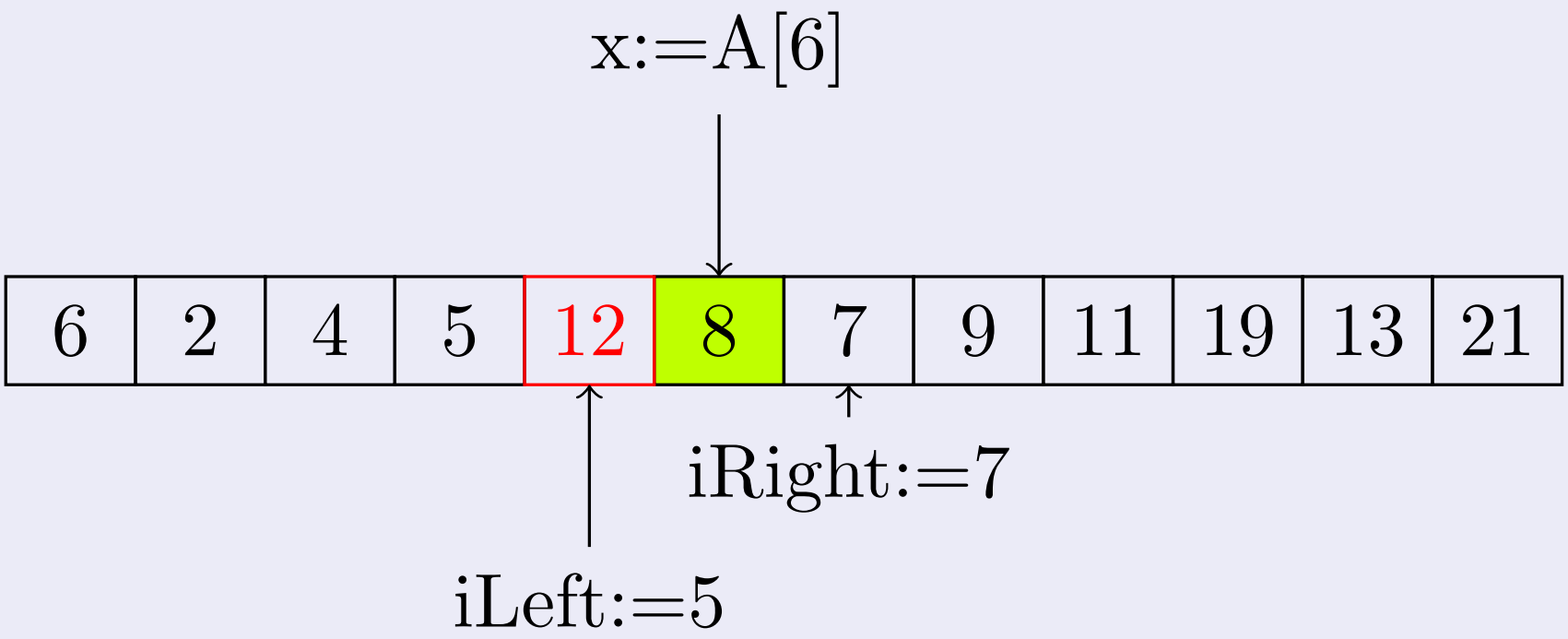


Figure : (2a)

`A[5]` $\geq$ `x`, deci nu putem avansa cu `iLeft`.

Exemplu:

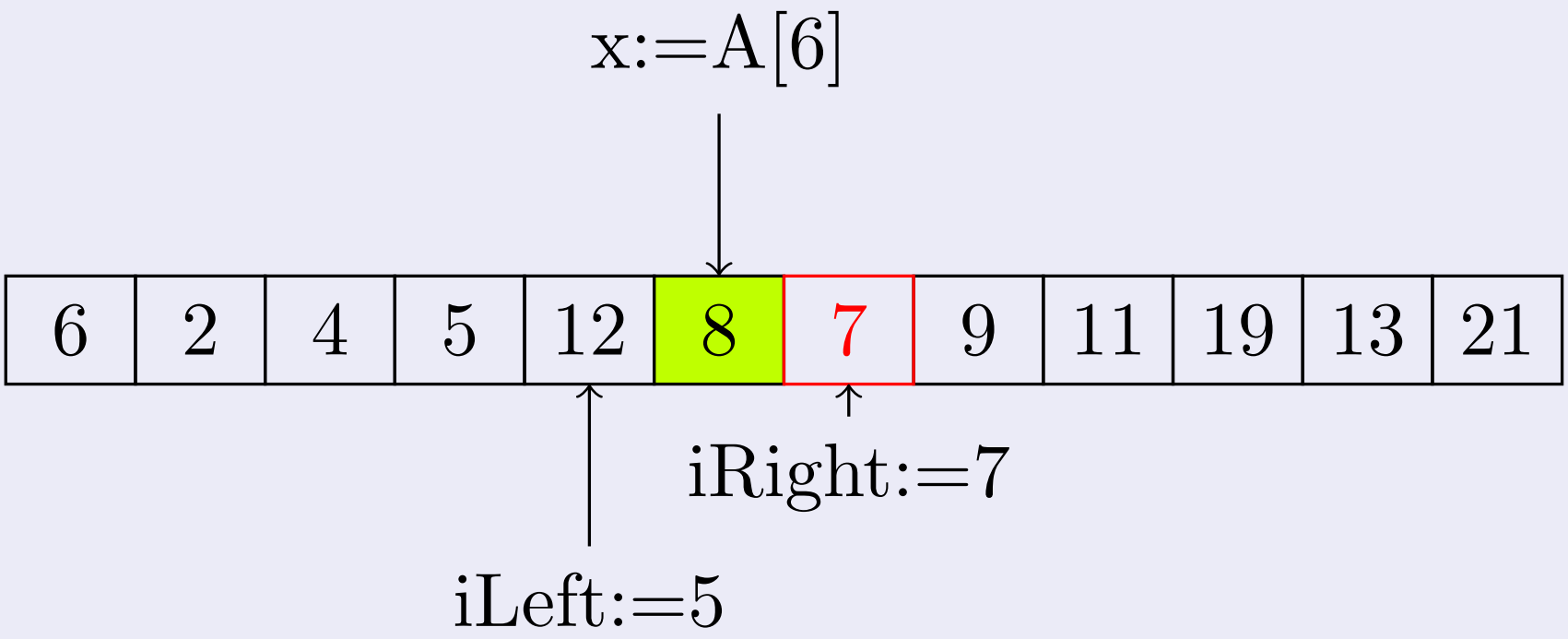


Figure : (2b)

`A[7]` $\leq$ `x`, deci nu putem avansa cu `iRight`.

Exemplu:

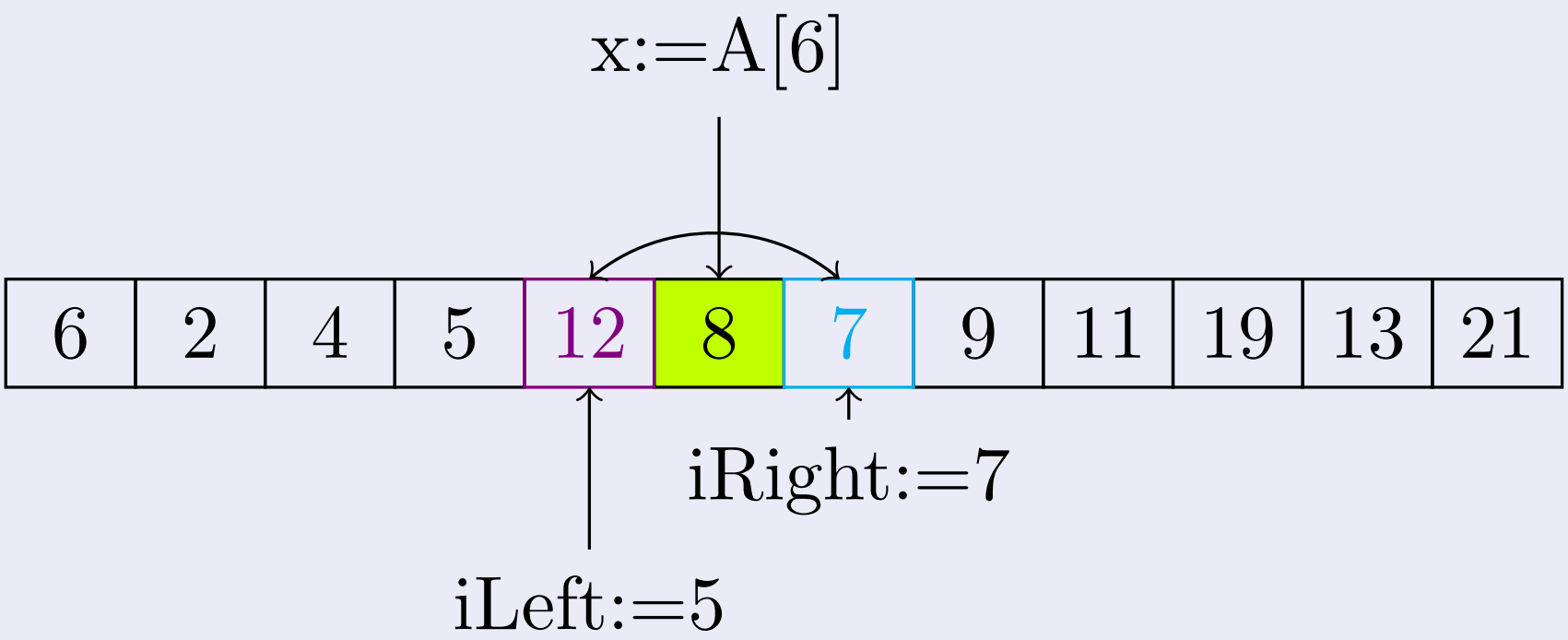


Figure : (3)

Trebuie să interschimbăm `A[5]` cu `A[7]`.

Exemplu:

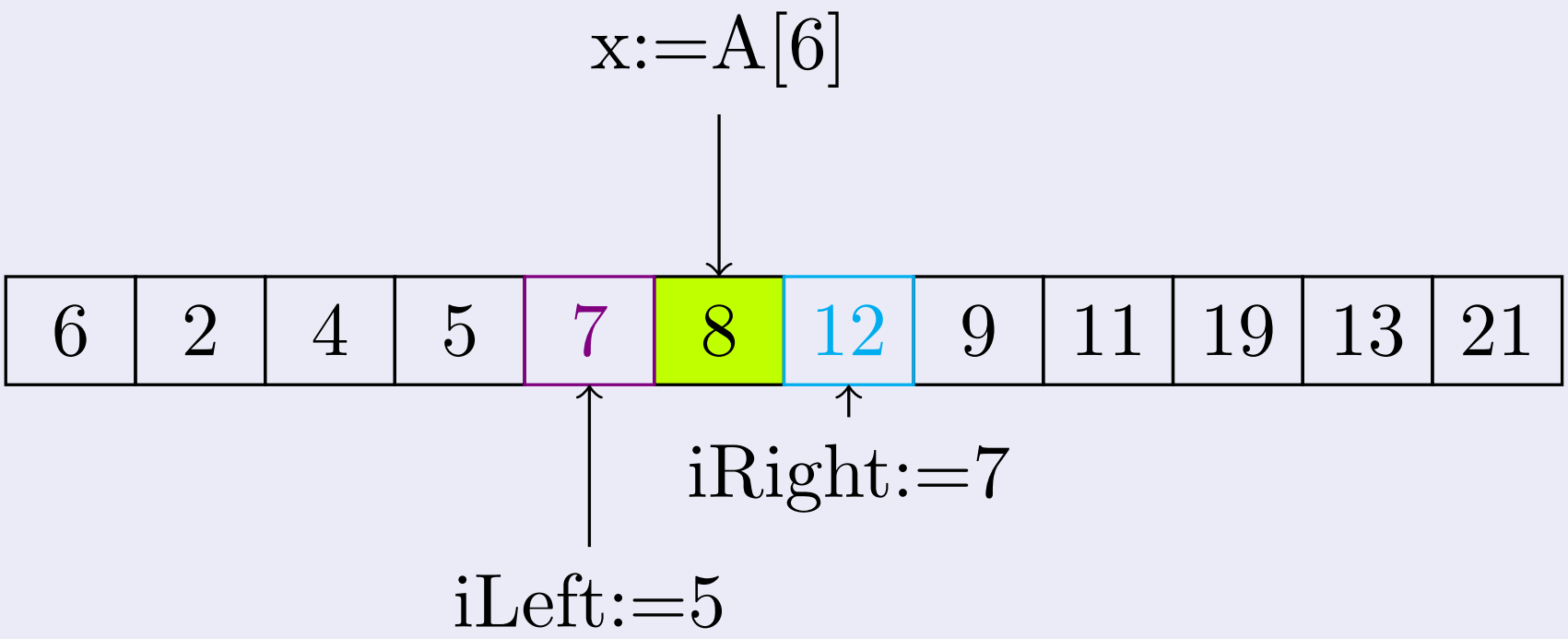


Figure : (3)

Vectorul obținut după interschimbare.

Exemplu:

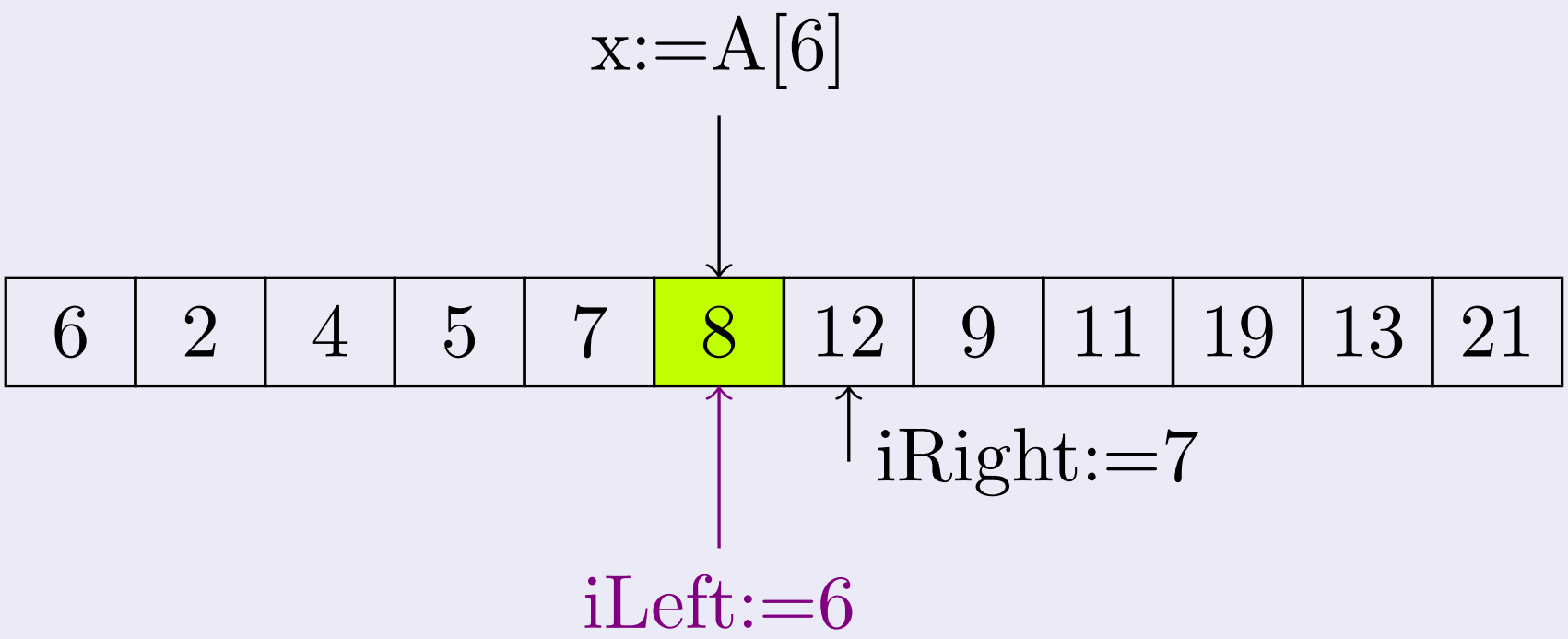


Figure : (3)

```
iLeft:= iLeft+1;
```

Exemplu:

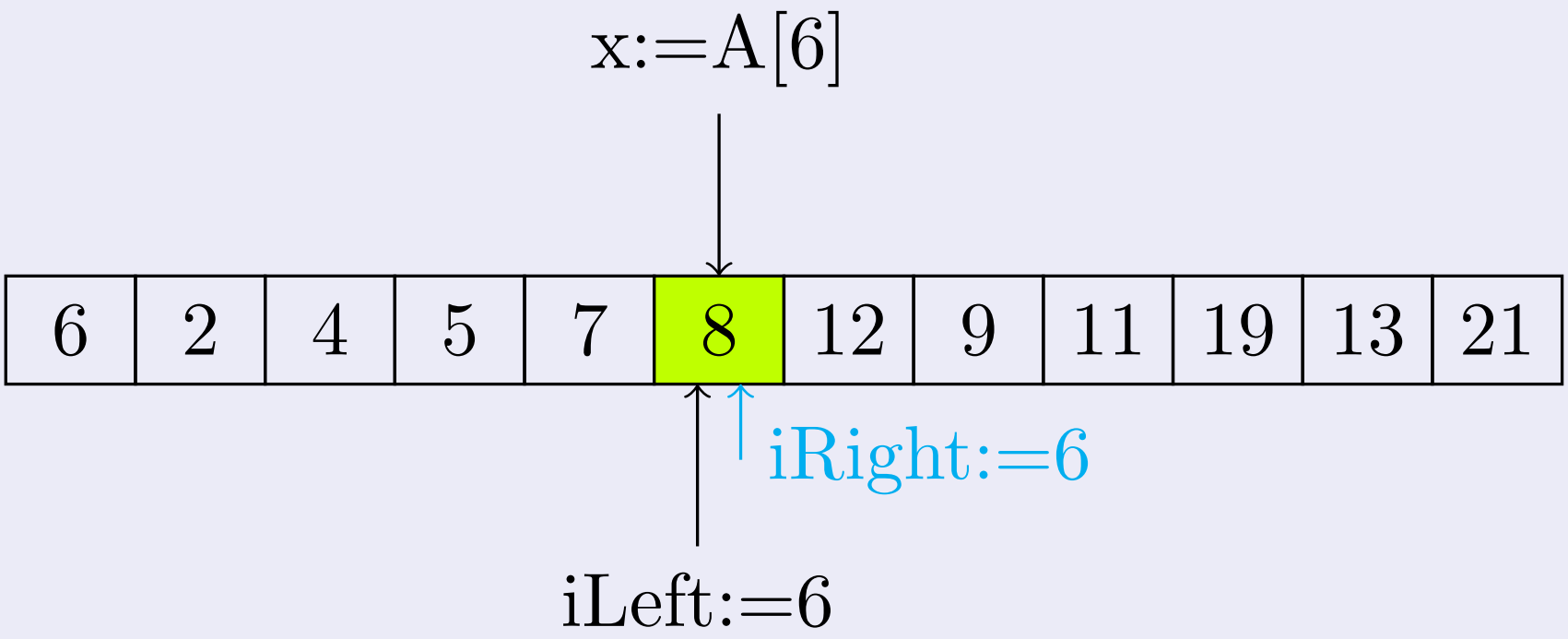


Figure : (3)

```
iRight:= iRight-1;
```

Exemplu:

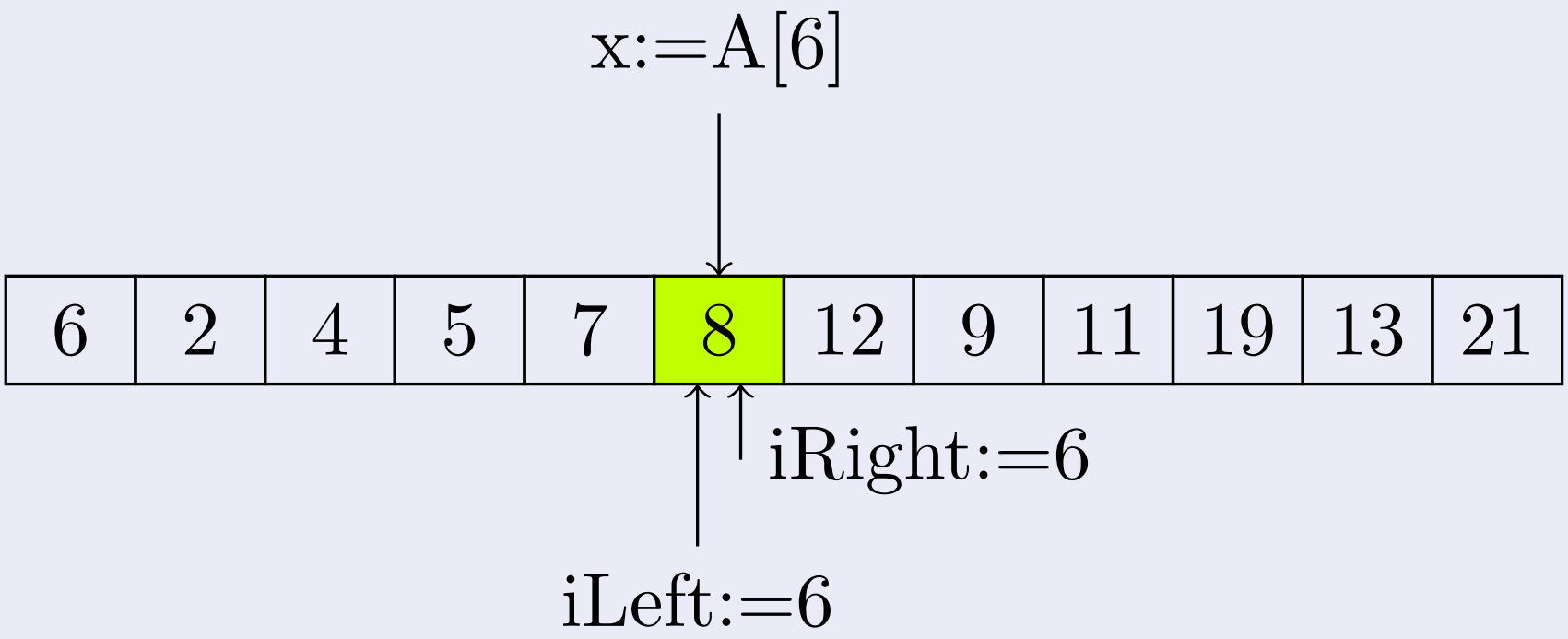


Figure : Vectorul după a patra iterație a ciclului `repeat`

Avem `6 = iLeft` $\leq$ `iRight = 6`, deci continuăm procedura de partiție.

Exemplu:

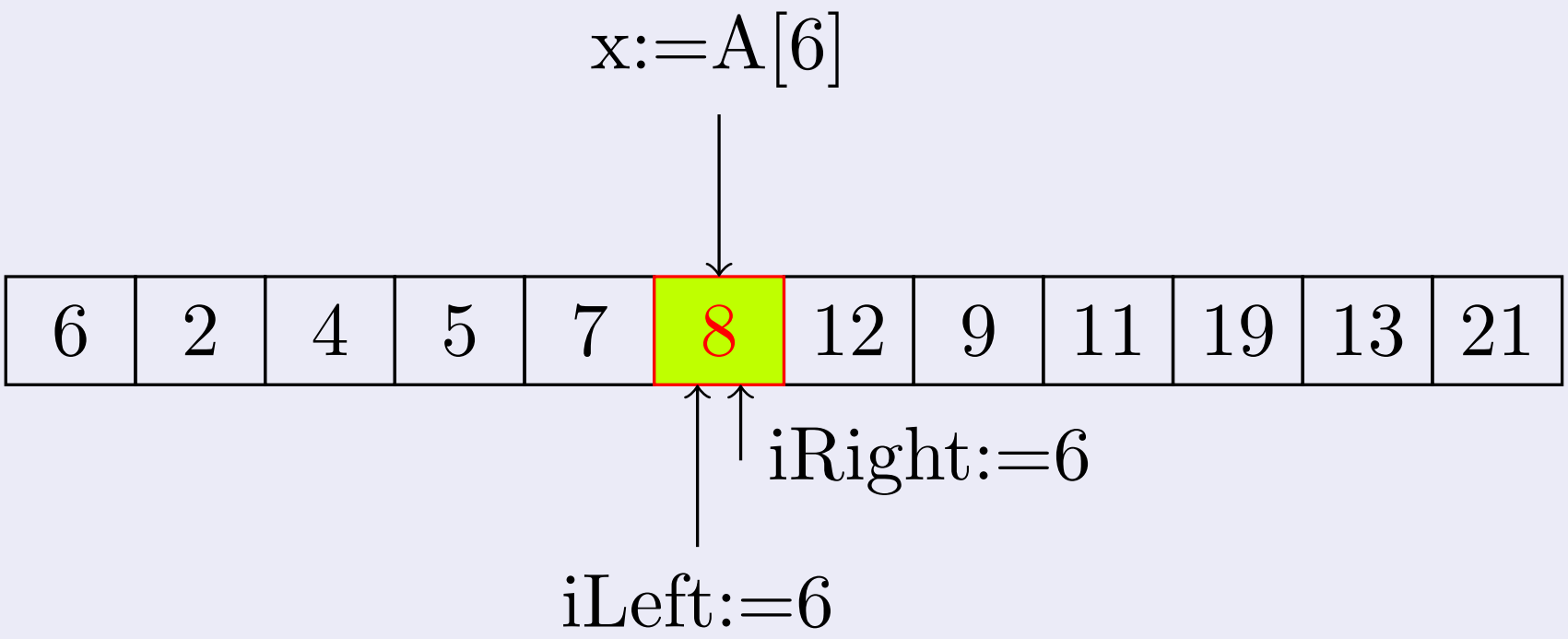


Figure : Vectorul după a patra iterație a ciclului `repeat`

`A[6]` $\geq$ `x`, deci nu putem avansa cu `iLeft`.

Exemplu:

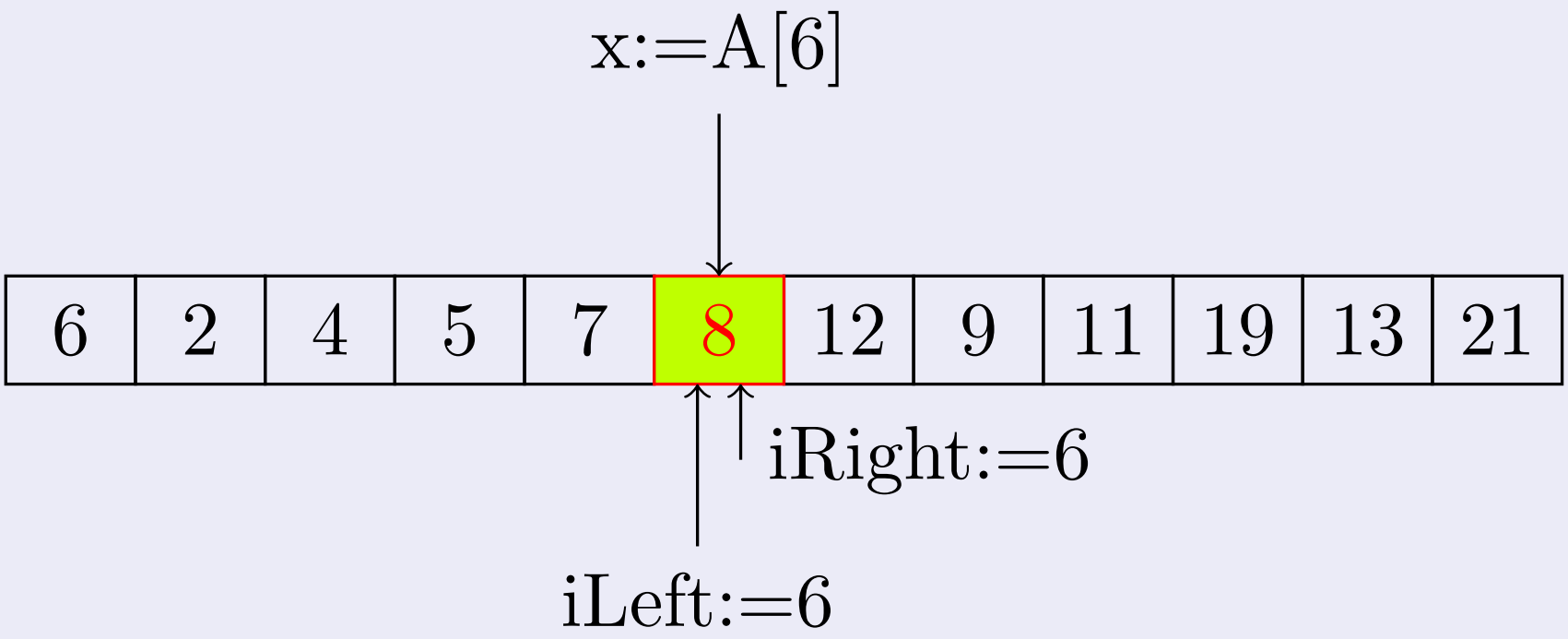


Figure : Vectorul după a patra iterație a ciclului `repeat`

`A[6]` $\leq$ `x`, deci nu putem avansa cu `iRight`.

Exemplu:

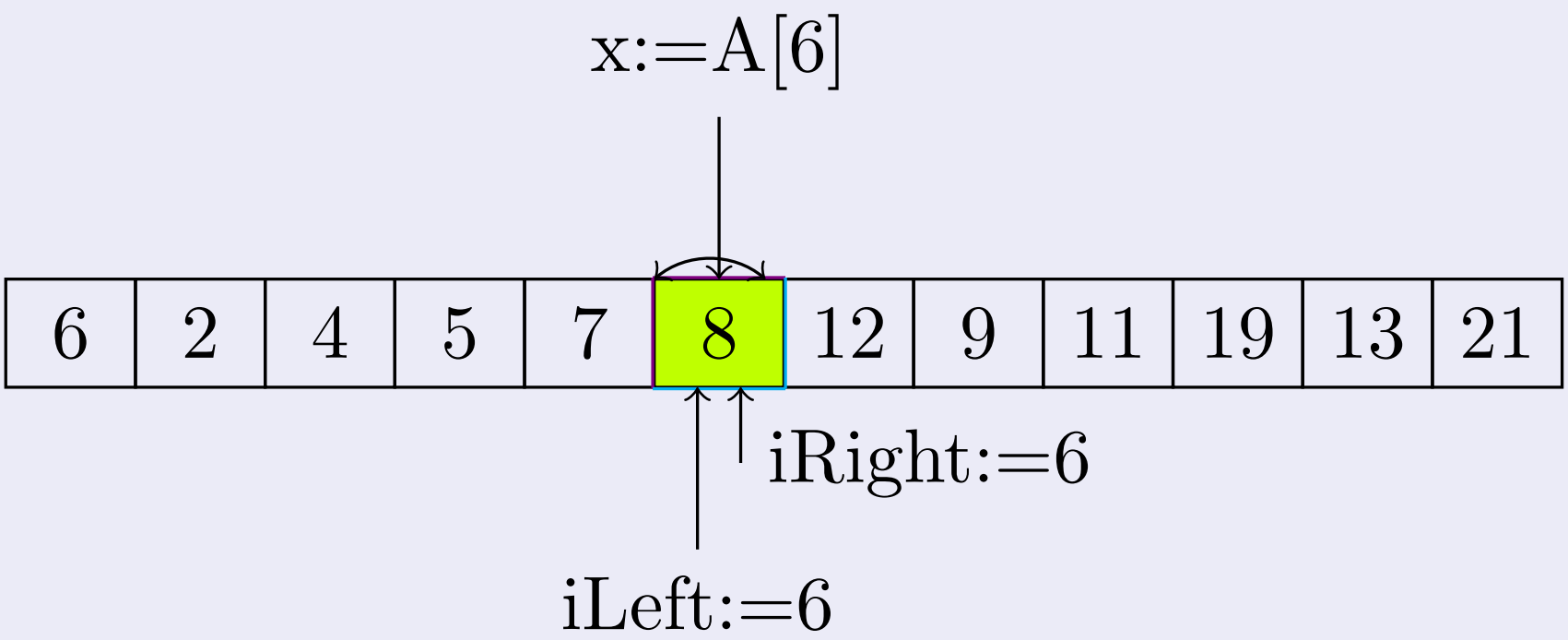


Figure : Vectorul după a patra iterație a ciclului `repeat`

Trebuie să interschimbăm `A[6]` cu `A[6]`.

Exemplu:

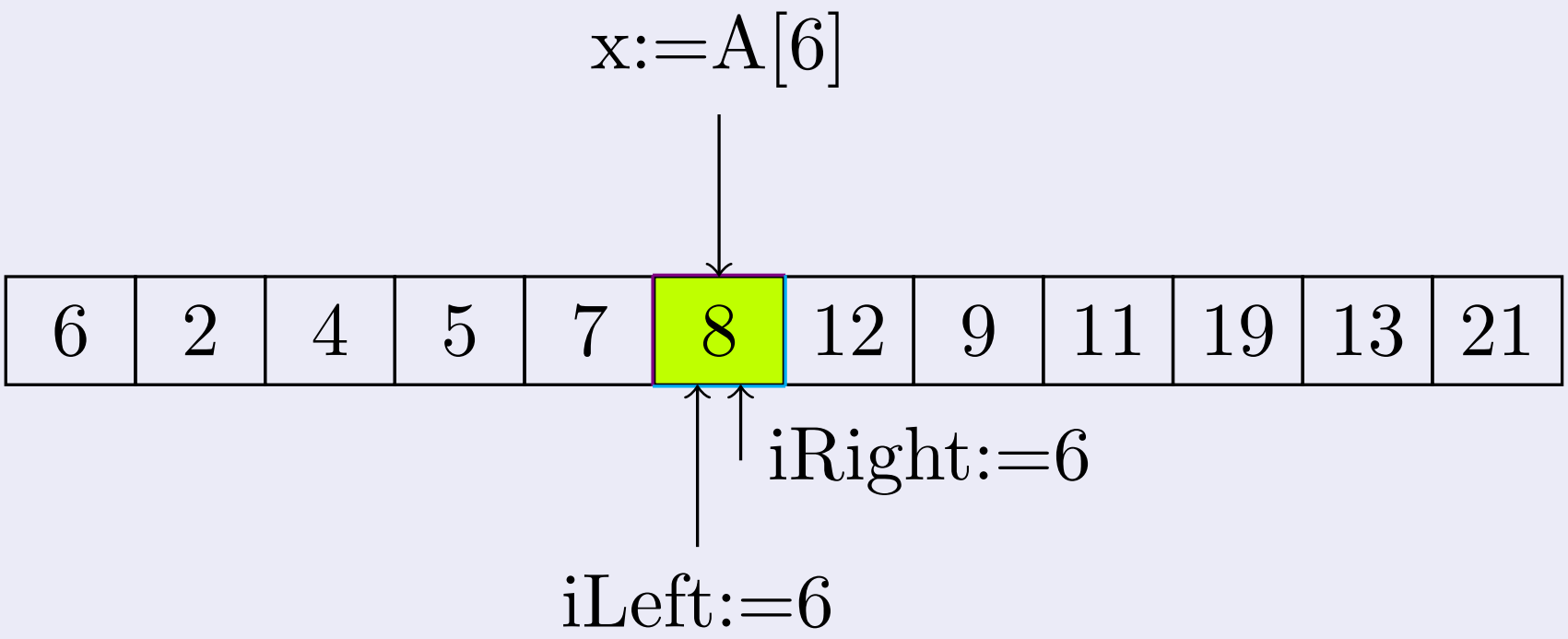


Figure : (3)

Vectorul obținut după interschimbare.

Exemplu:

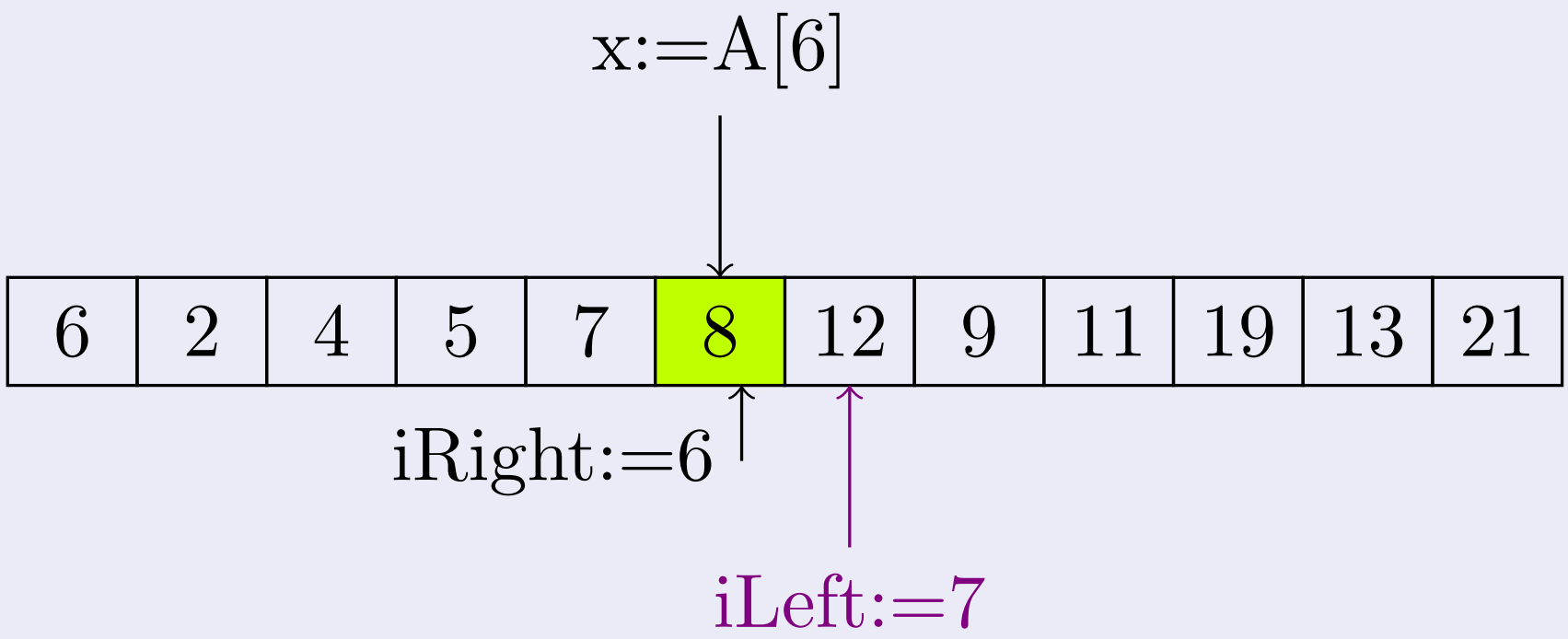


Figure : (3)

```
iLeft:= iLeft+1;
```

Exemplu:

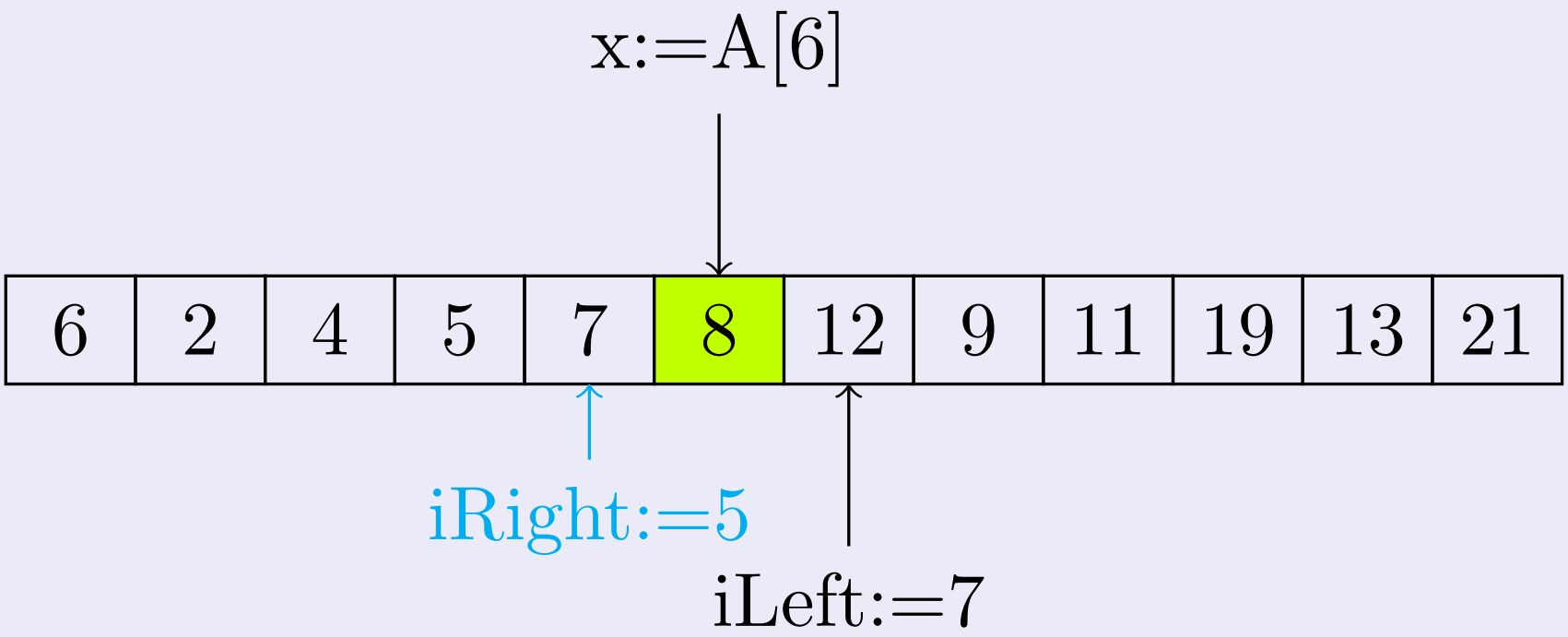


Figure : (3)

```
iRight:= iRight-1;
```

Exemplu:

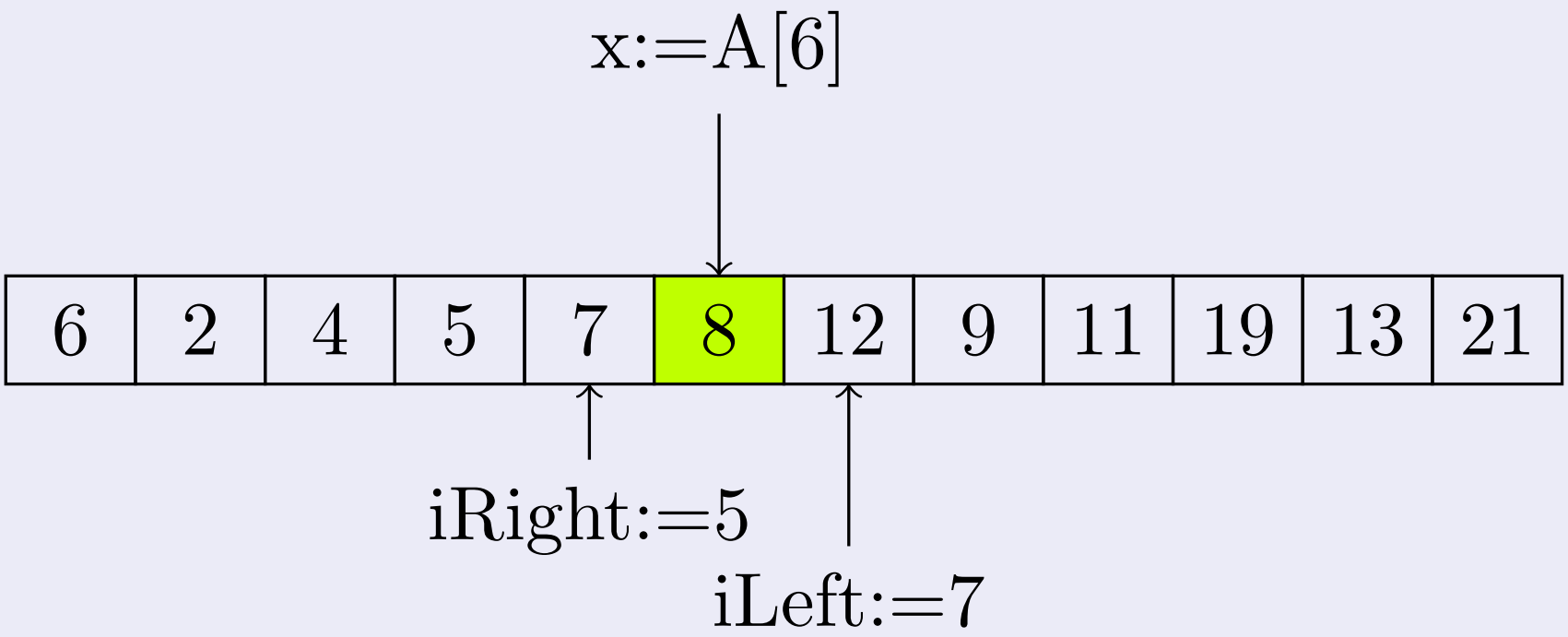


Figure : Vectorul după a patra iterație a ciclului `repeat`

Avem `7 = iLeft > iRight = 5`, deci încheiem procedura de partiție.

Exemplu:

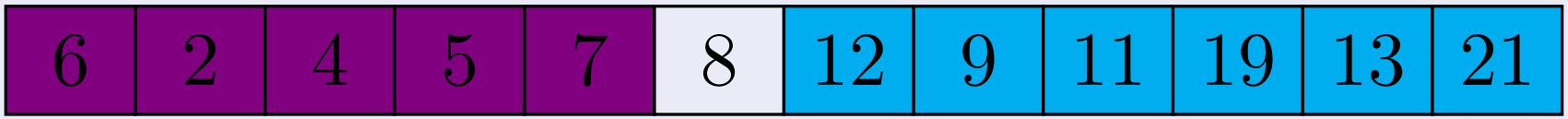


Figure : Final

Am obținut partiția: `A[1..5]`, `A[7..12]`.

Ce se întâmplă dacă vectorul conține chei multiple?

Un alt exemplu..

Exemplu pentru chei multiple:

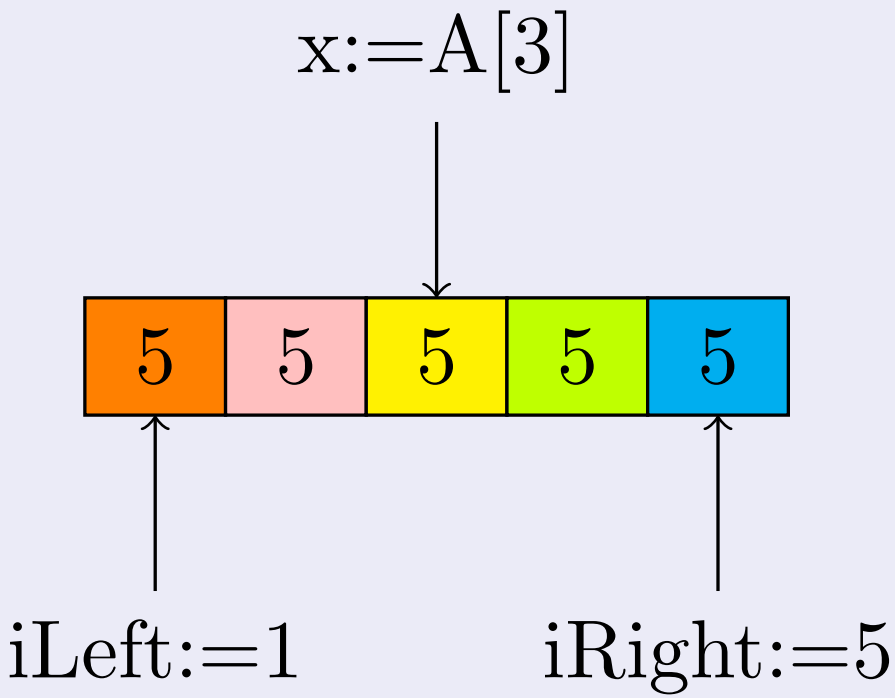


Figure : vectorul *A* inițial

```
x:=A[(Left+Right)div 2];
   iLeft:=1; iRight=5;
```

Exemplu pentru chei multiple:

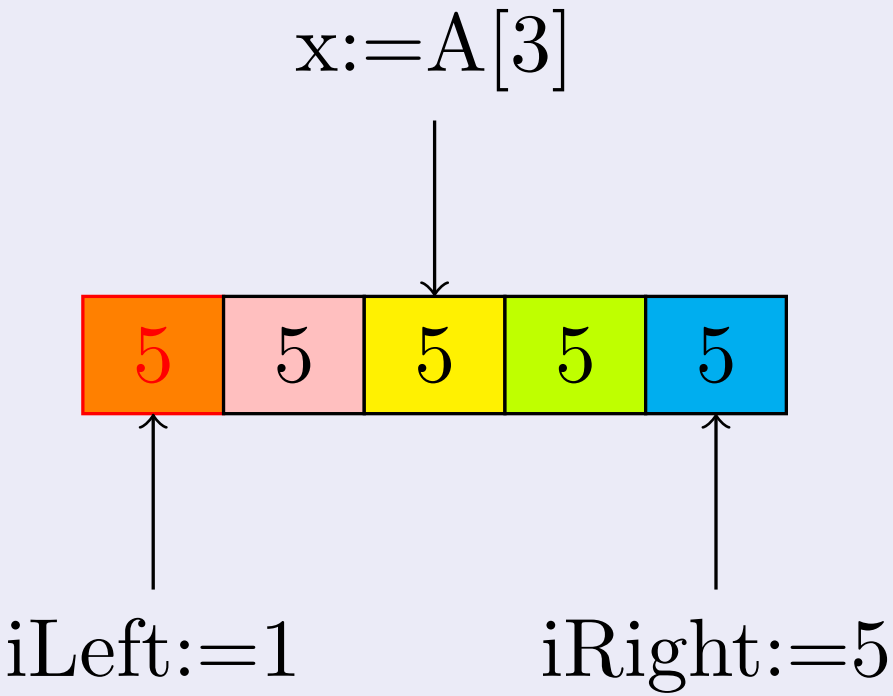


Figure : (2a)

`A[1]` $\geq$ `x`, deci nu putem avansa cu `iLeft`.

Exemplu pentru chei multiple:

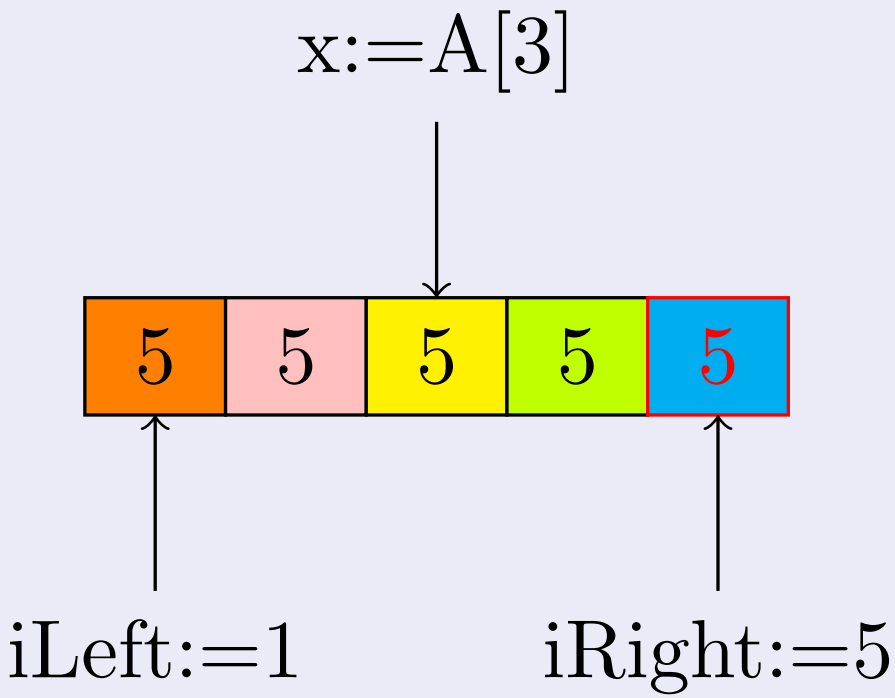


Figure : (2b)

`A[5]` $\leq$ `x`, deci nu putem avansa cu `iRight`.

Exemplu pentru chei multiple:

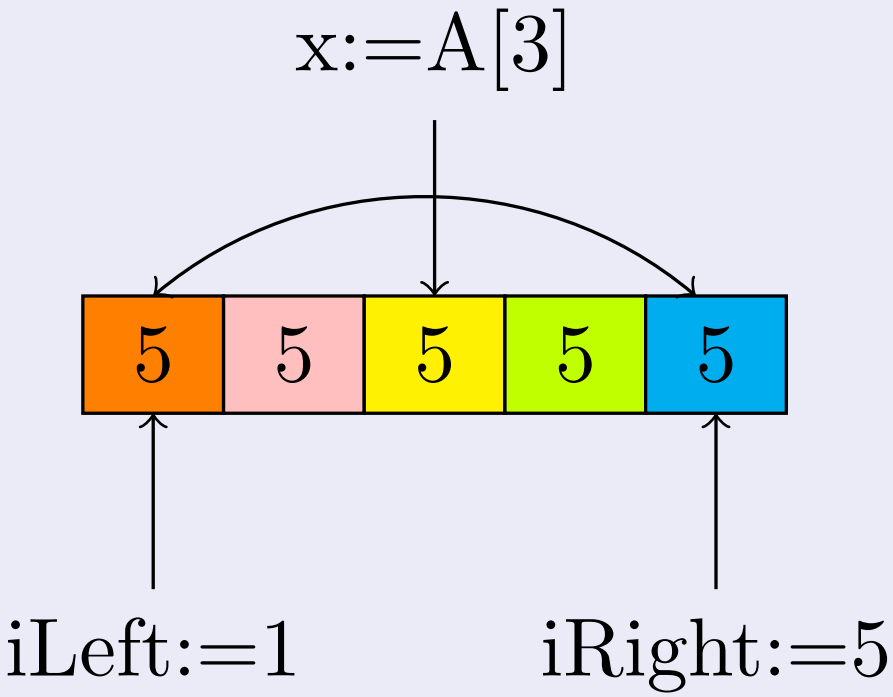


Figure : (2b)

Trebuie să interschimbăm `A[1]` și `A[5]`.

Exemplu pentru chei multiple:

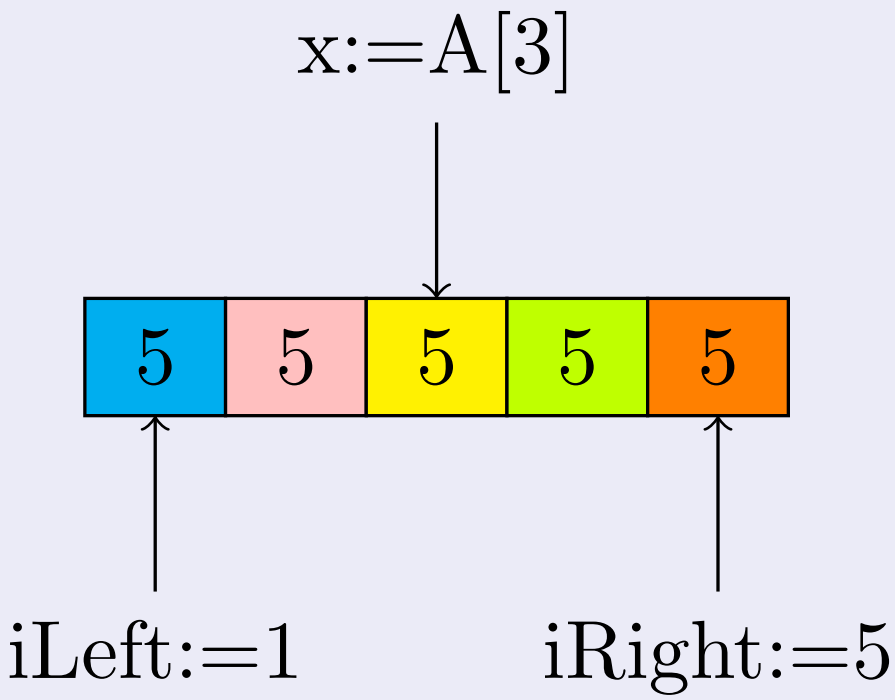


Figure : (3)

Vectorul obținut după interschimbare.

Exemplu pentru chei multiple:

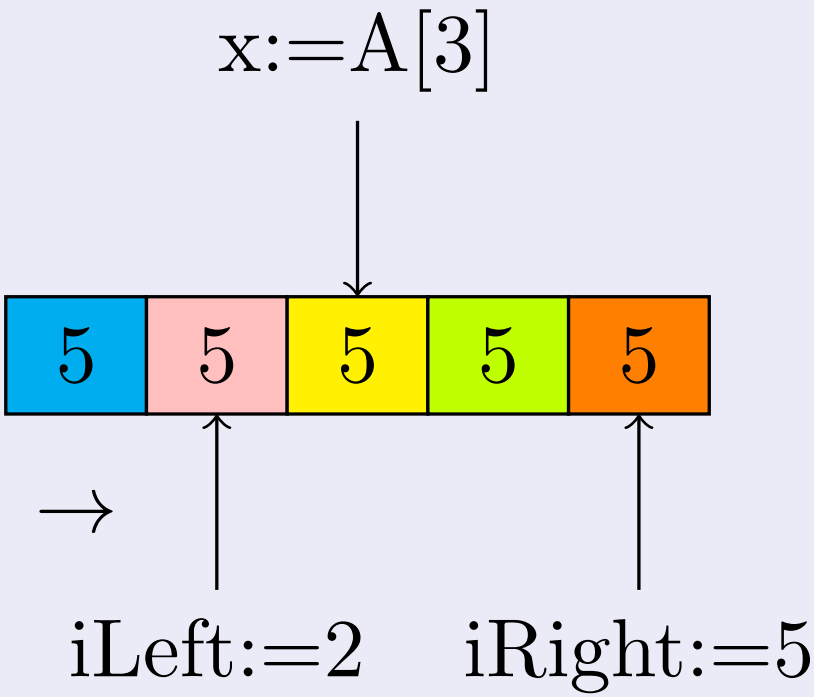


Figure : (3)

```
iLeft:= iLeft+1;
```

Exemplu pentru chei multiple:

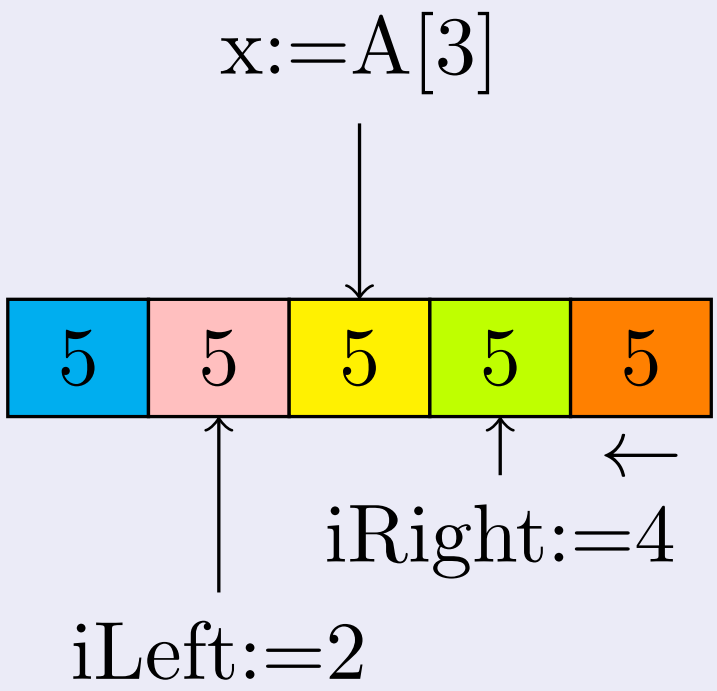


Figure : (3)

```
iRight:= iRight-1;
```

Exemplu pentru chei multiple:

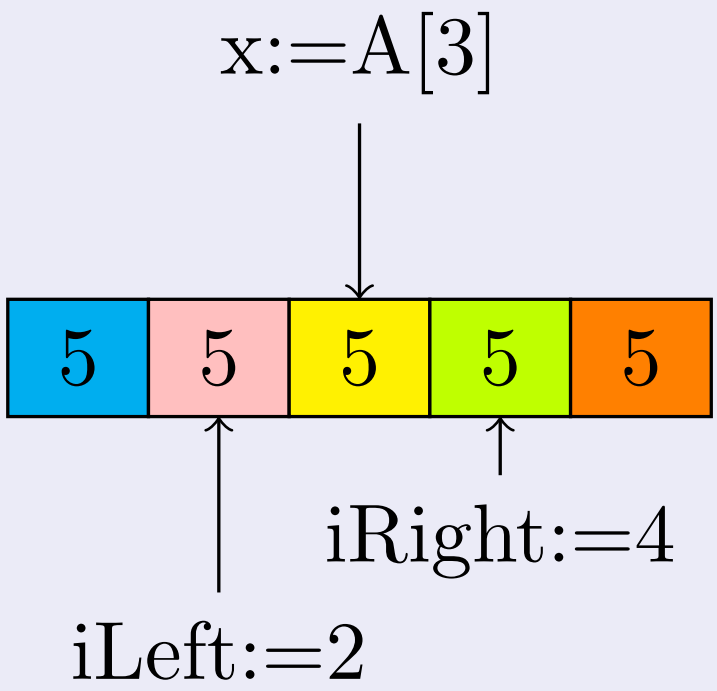


Figure : (3)

Avem 2 = iLeft $\leq$ iRight = 4, deci continuăm procedura de partiție.

Exemplu pentru chei multiple:

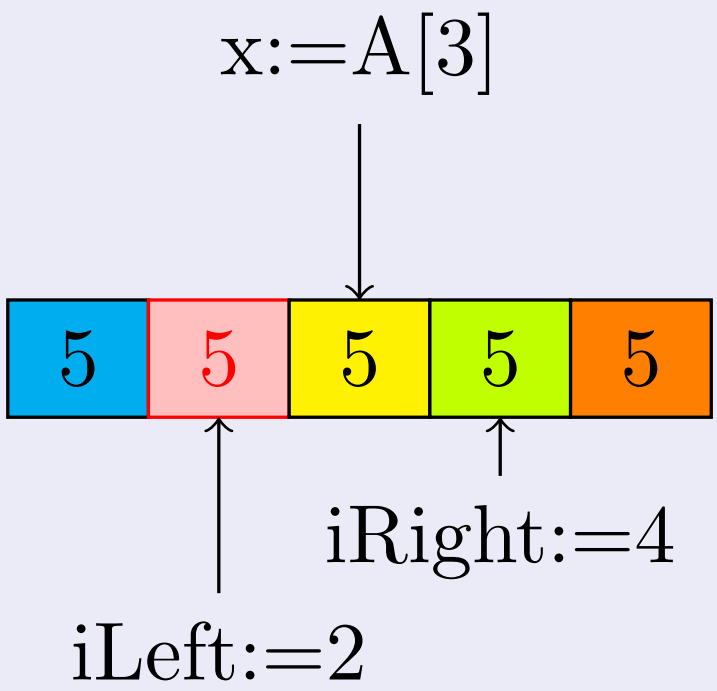


Figure : (3)

`A[2]` $\geq$ `x`, deci nu putem avansa cu `iLeft`.

Exemplu pentru chei multiple:

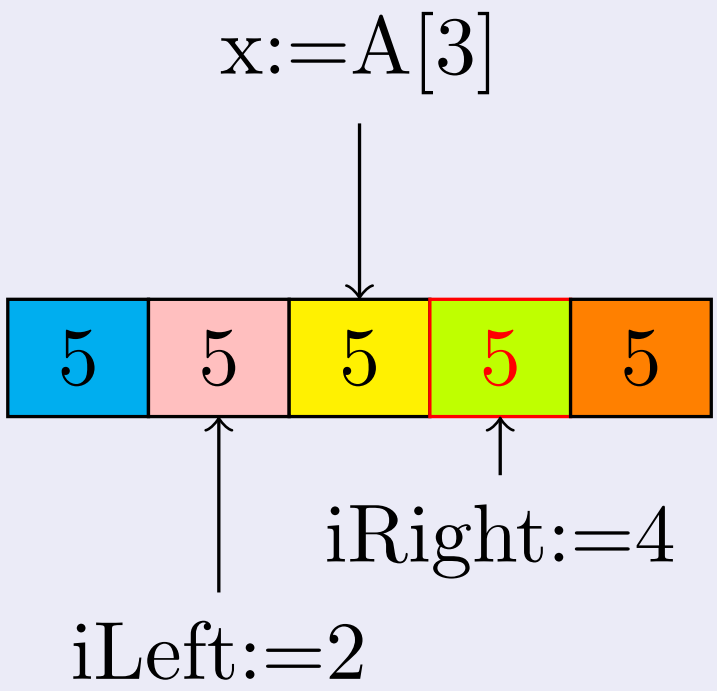


Figure : După prima iterație a ciclului `repeat`

`A[4]` $\leq$ `x`, deci nu putem avansa cu `iRight`.

Exemplu pentru chei multiple:

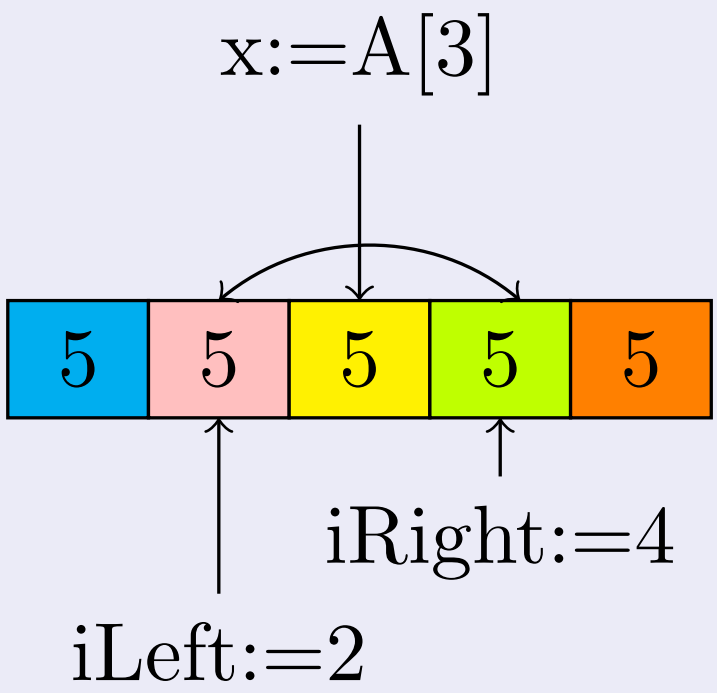


Figure : (2a)

Trebuie să interschimbăm `A[2]` și `A[4]`.

Exemplu pentru chei multiple:

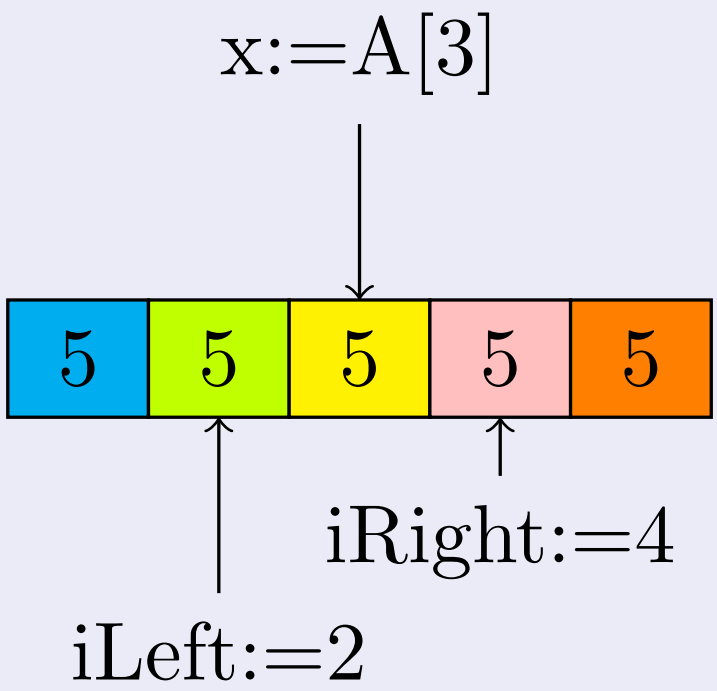


Figure : (2b)

Vectorul obținut după interschimbare.

Exemplu pentru chei multiple:

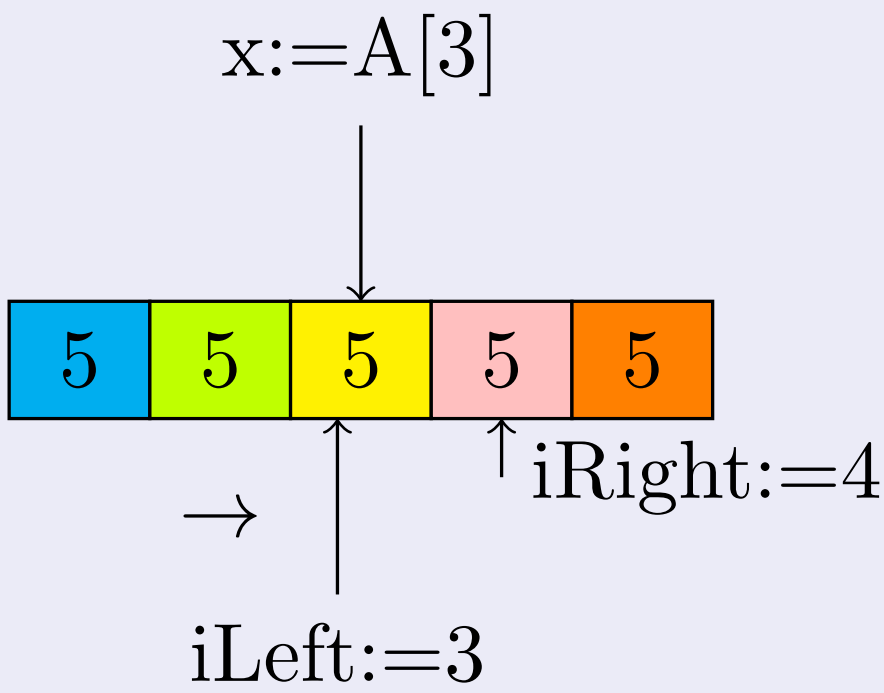


Figure : (3)

```
iLeft:= iLeft+1;
```

Exemplu pentru chei multiple:

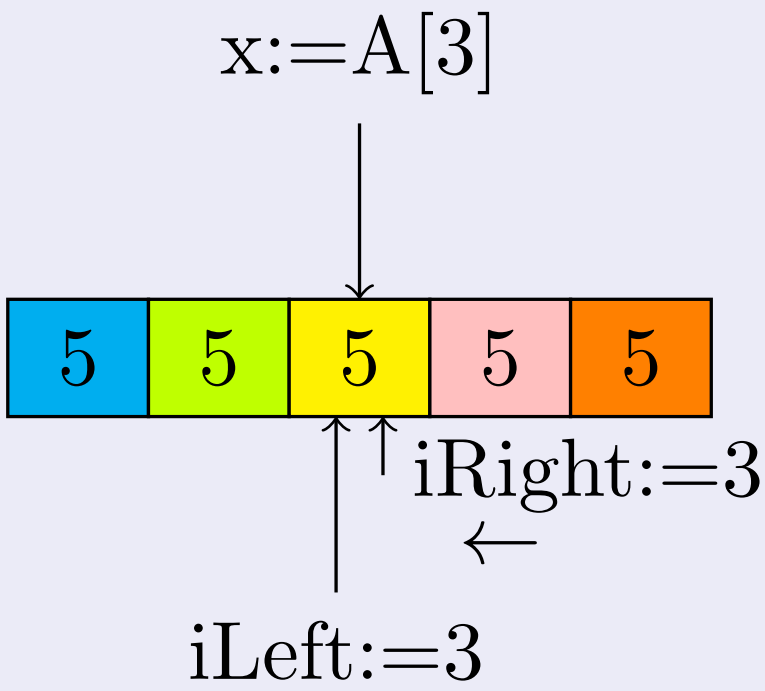


Figure : (3)

```
iRight := iRight-1;
```

Exemplu pentru chei multiple:

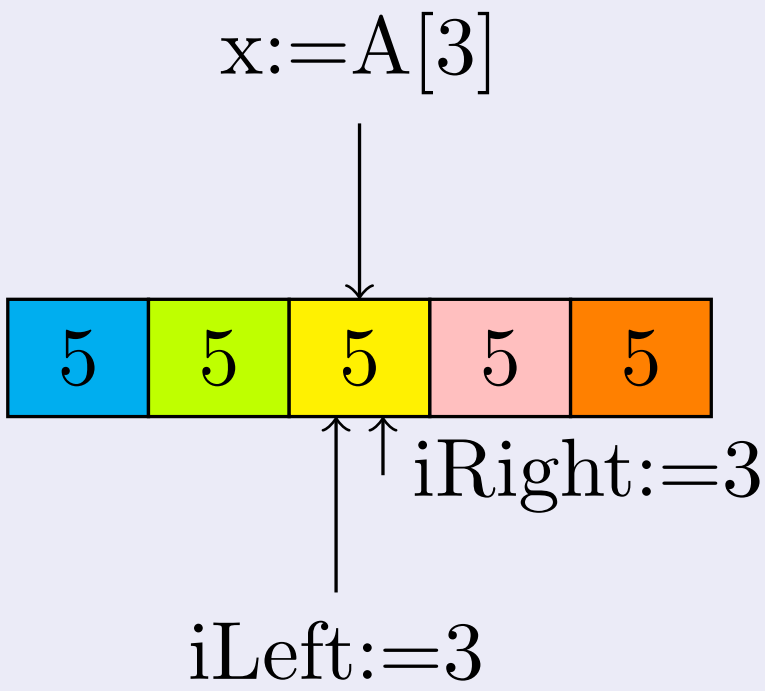


Figure : (3)

Avem 3 = iLeft $\leq$ iRight = 3, deci continuăm procedura de partiție.

Exemplu pentru chei multiple:

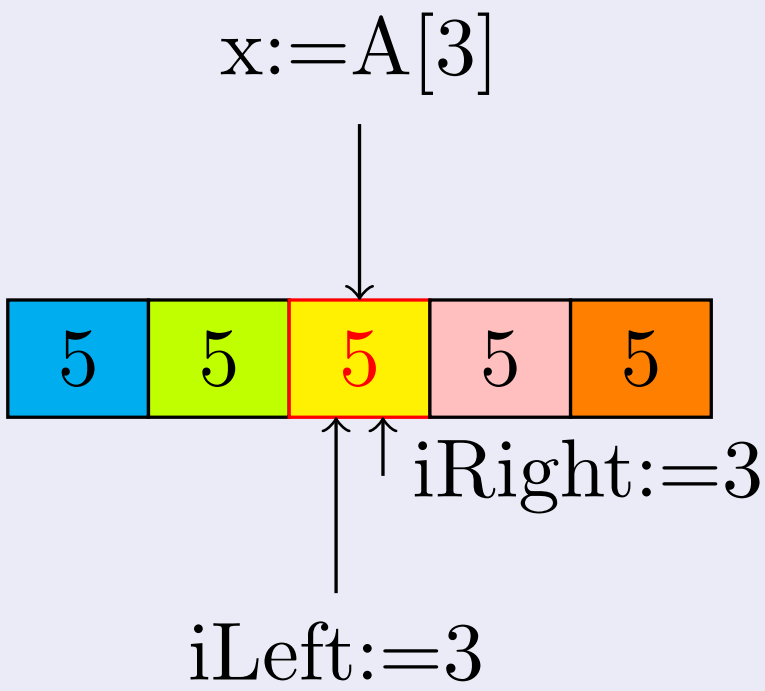


Figure : (3)

A[3 $\geq$ x, deci nu putem avansa cu iLeft.

Exemplu pentru chei multiple:

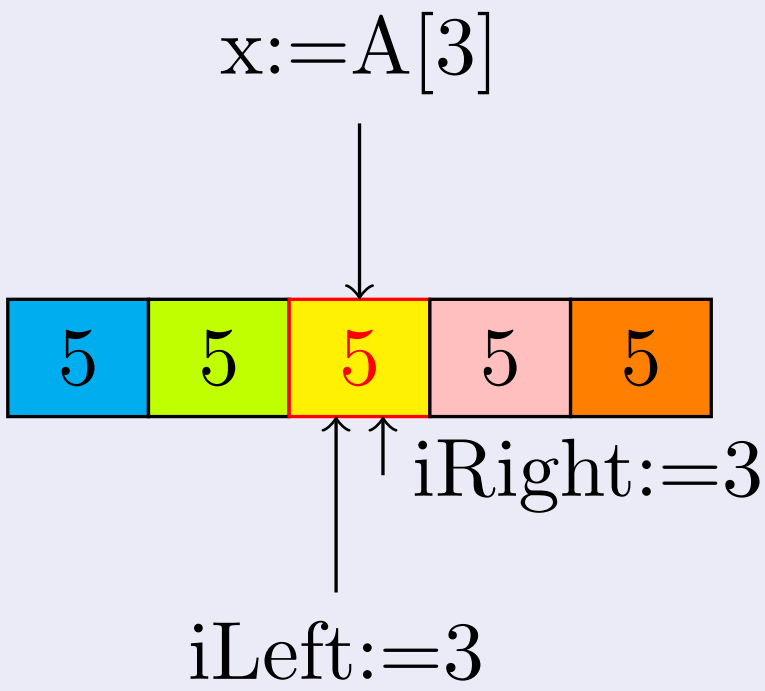


Figure : (Vectorul după a doua iterație a ciclului `repeat`)

`A[3]` $\leq$ `x`, deci nu putem avansa cu `iRight`.

Exemplu pentru chei multiple:

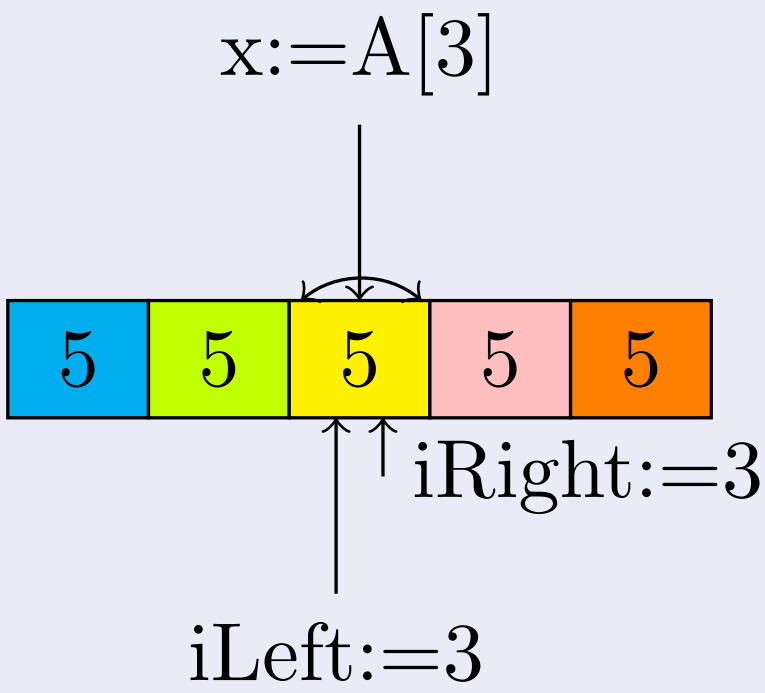


Figure : (2a)

Trebuie să interschimbăm `A[3]` și `A[3]`.

Exemplu pentru chei multiple:

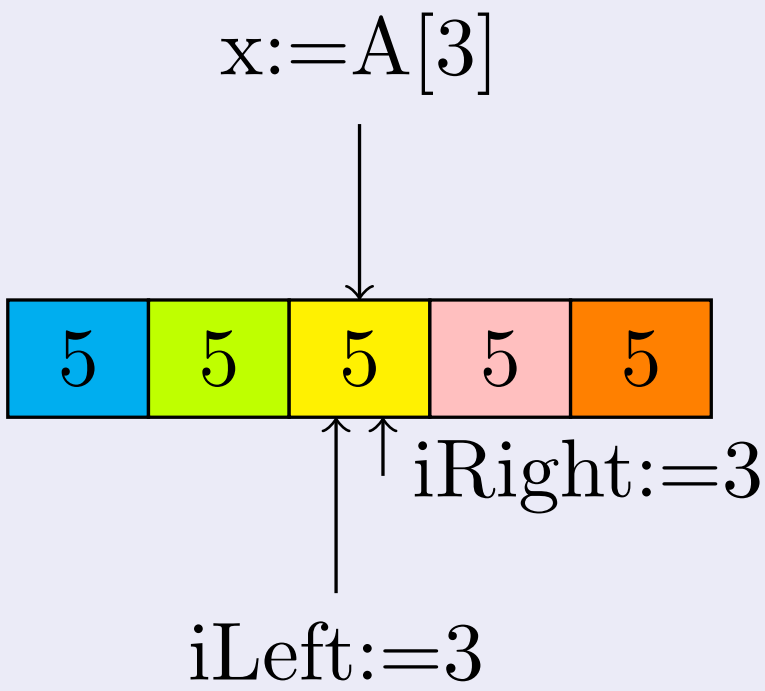


Figure : (2b)

Vectorul obținut după interschimbare.

Exemplu pentru chei multiple:

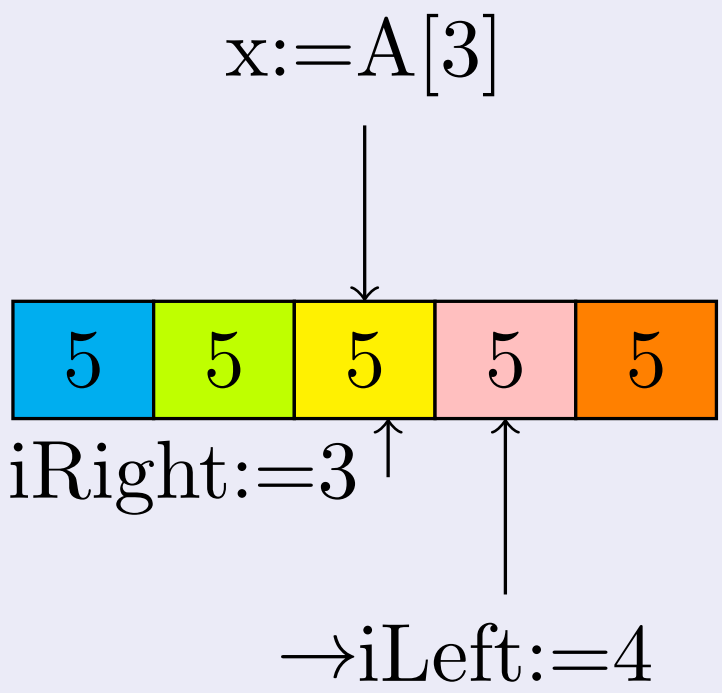


Figure : (2b)

```
iLeft := iLeft+1;
```

Exemplu pentru chei multiple:

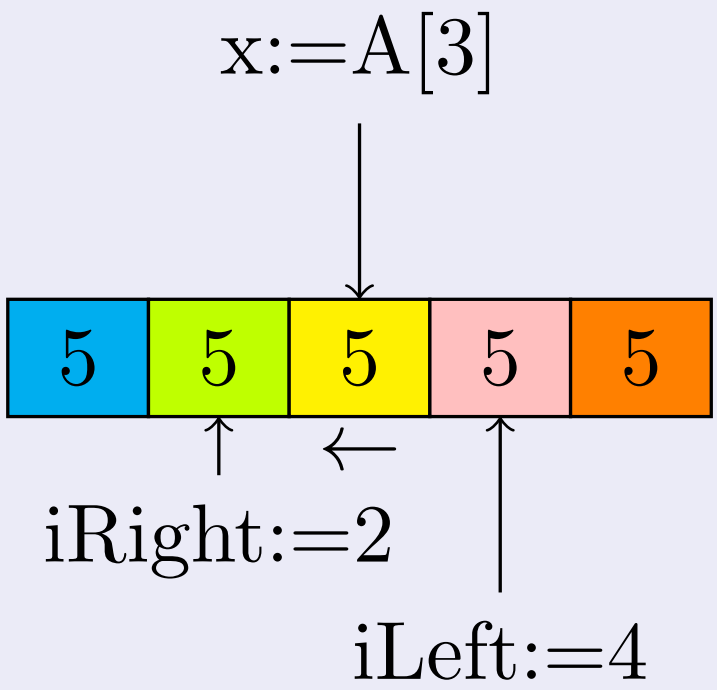


Figure : (2b)

```
iRight := iRight-1;
```

Exemplu pentru chei multiple:

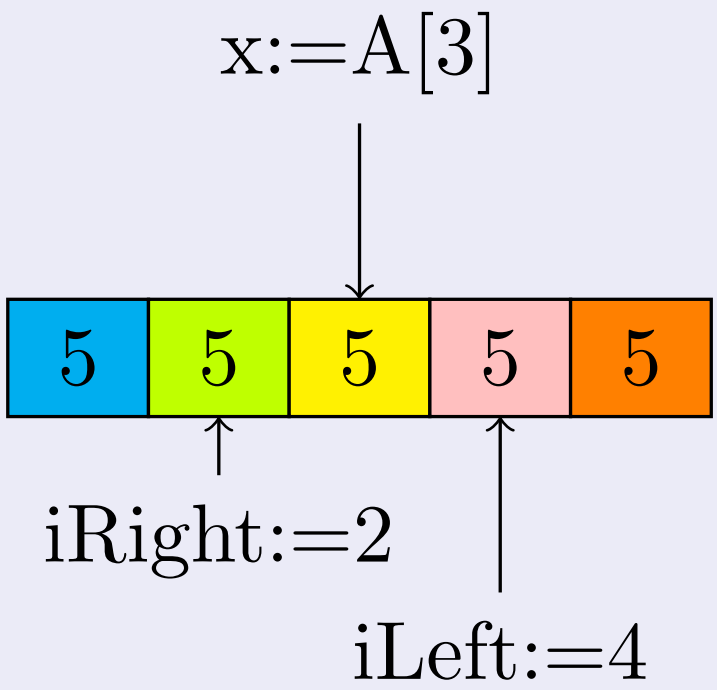


Figure : (3)

Avem `4 = iLeft > iRight = 2`, deci încheiem procedura de partiție.

Exemplu pentru chei multiple:

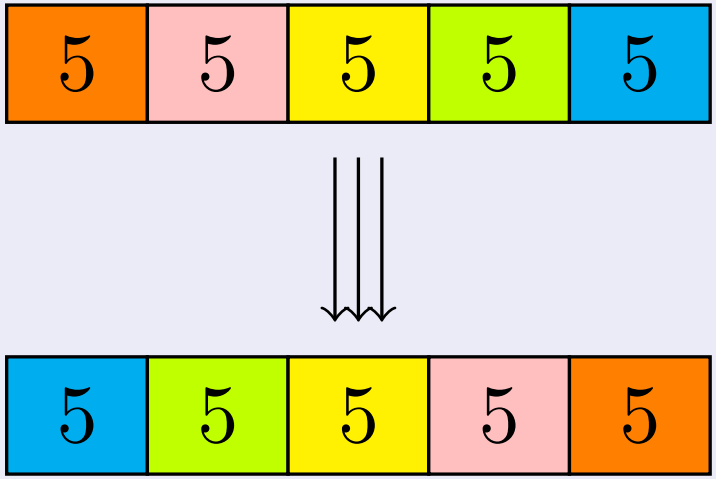


Figure : Final

Observăm că deși valorile erau egale, acestea și-au schimbat poziția.

Exemplu pentru chei multiple:

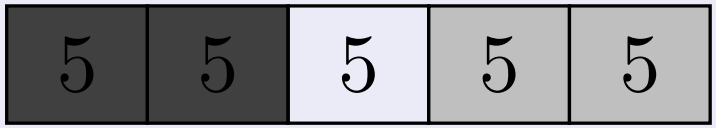


Figure : Final

Am obținut partiția: `A[1..2]`, `A[4..5]`.
Observăm că avem `iLeft < iRight` și partiția dată:
`A[Left..iRight]` și `A[iLeft..Right]`.

Partiția cu pivot într-o extremitate

Până acum am ales ca pivot valoarea mediană din vectorul pe care facem partiția. Ce se întâmplă dacă alegem o **valoare situată la una dintre extremități**?

Partiția cu pivot într-o extremitate

Până acum am ales ca pivot valoarea mediană din vectorul pe care facem partiția. Ce se întâmplă dacă alegem o **valoare situată la una dintre extremități**?

- Să presupunem că, pentru partiționarea vectorului $A[Left..Right]$ alegem $x = A[Left]$.

Partiția cu pivot într-o extremitate

Până acum am ales ca pivot valoarea mediană din vectorul pe care facem partiția. Ce se întâmplă dacă alegem o **valoare situată la una dintre extremități**?

- Să presupunem că, pentru partiționarea vectorului $A[Left..Right]$ alegem $x = A[Left]$.
- Atunci, pasul (2a) din algoritmul de partiționare (parcurgerea de la stânga la dreapta a vectorului până la primul indice i pentru care $A[i] \geq x$) nu mai are sens, căci acest indice este chiar $Left$.

Partiția cu pivot într-o extremitate

Până acum am ales ca pivot valoarea mediană din vectorul pe care facem partiția. Ce se întâmplă dacă alegem o **valoare situată la una dintre extremități**?

- Să presupunem că, pentru partiționarea vectorului $A[Left..Right]$ alegem $x = A[Left]$.
- Atunci, pasul (2a) din algoritmul de partiționare (parcurgerea de la stânga la dreapta a vectorului până la primul indice i pentru care $A[i] \geq x$) nu mai are sens, căci acest indice este chiar $Left$.
- Executăm (2b), adică parcurgem de la dreapta la stânga vectorul până la primul indice j pentru care $A[j] \leq x$.

Partiția cu pivot într-o extremitate

Până acum am ales ca pivot valoarea mediană din vectorul pe care facem partiția. Ce se întâmplă dacă alegem o **valoare situată la una dintre extremități**?

- Să presupunem că, pentru partiționarea vectorului $A[Left..Right]$ alegem $x = A[Left]$.
- Atunci, pasul (2a) din algoritmul de partiționare (parcurgerea de la stânga la dreapta a vectorului până la primul indice i pentru care $A[i] \geq x$) nu mai are sens, căci acest indice este chiar $Left$.
- Executăm (2b), adică parcurgem de la dreapta la stânga vectorul până la primul indice j pentru care $A[j] \leq x$.
- Pasul (3) devine:

```
if Left < j then
  interschimbă (A[Left], A[j])
```

- Observăm că valoarea pivot $x = A[Left]$ a participat la interschimbare, și se află acum plasată în extremitatea dreaptă a subvectorului $A[Left..j]$.

- Observăm că valoarea pivot $x = A[Left]$ a participat la interschimbare, și se află acum plasată în extremitatea dreaptă a subvectorului $A[Left..j]$.
- Reluăm acum parcurgerea de tip (2a) de la stânga la dreapta. Parcurgerea de tip (2b) nu mai are sens.

- Observăm că valoarea pivot $x = A[Left]$ a participat la interschimbare, și se află acum plasată în extremitatea dreaptă a subvectorului $A[Left..j]$.
- Reluăm acum parcurgerea de tip (2a) de la stânga la dreapta. Parcurgerea de tip (2b) nu mai are sens.
- La sfârșit interschimbăm. Observăm că valoarea pivot participă din nou la interschimbare.

- Observăm că valoarea pivot $x = A[Left]$ a participat la interschimbare, și se află acum plasată în extremitatea dreaptă a subvectorului $A[Left..j]$.
- Reluăm acum parcurgerea de tip (2a) de la stânga la dreapta. Parcurgerea de tip (2b) nu mai are sens.
- La sfârșit interschimbăm. Observăm că valoarea pivot participă din nou la interschimbare.
- Procedeul continuă până când cele două parcurgeri se întâlnesc, adică atâta timp cât mai există componente în vector care nu au fost comparate cu pivotul.

- Observăm că valoarea pivot $x = A[Left]$ a participat la interschimbare, și se află acum plasată în extremitatea dreaptă a subvectorului $A[Left..j]$.
- Reluăm acum parcurgerea de tip (2a) de la stânga la dreapta. Parcurgerea de tip (2b) nu mai are sens.
- La sfârșit interschimbăm. Observăm că valoarea pivot participă din nou la interschimbare.
- Procedeul continuă până când cele două parcurgeri se întâlnesc, adică atâta timp cât mai există componente în vector care nu au fost comparate cu pivotul.
- La sfârșit obținem valoarea pivot x plasată la locul ei final în vector.

- Observăm că valoarea pivot $x = A[Left]$ a participat la interschimbare, și se află acum plasată în extremitatea dreaptă a subvectorului $A[Left..j]$.
- Reluăm acum parcurgerea de tip (2a) de la stânga la dreapta. Parcurgerea de tip (2b) nu mai are sens.
- La sfârșit interschimbăm. Observăm că valoarea pivot participă din nou la interschimbare.
- Procedeul continuă până când cele două parcurgeri se întâlnesc, adică atâta timp cât mai există componente în vector care nu au fost comparate cu pivotul.
- La sfârșit obținem valoarea pivot x plasată la locul ei final în vector.
- Fie loc indicele lui A ce va conține pe x. Avem atunci:
  $A[k] \leq x \; \forall k \in [1..loc-1]$
  $A[k] \geq x \; \forall k \in [loc+1..n]$

- Observăm că valoarea pivot $x = A[Left]$ a participat la interschimbare, și se află acum plasată în extremitatea dreaptă a subvectorului $A[Left..j]$.
- Reluăm acum parcurgerea de tip (2a) de la stânga la dreapta. Parcurgerea de tip (2b) nu mai are sens.
- La sfârșit interschimbăm. Observăm că valoarea pivot participă din nou la interschimbare.
- Procedeul continuă până când cele două parcurgeri se întâlnesc, adică atâta timp cât mai există componente în vector care nu au fost comparate cu pivotul.
- La sfârșit obținem valoarea pivot x plasată la locul ei final în vector.
- Fie loc indicele lui A ce va conține pe x. Avem atunci:
  $A[k] \leq x \; \forall k \in [1..loc-1]$
  $A[k] \geq x \; \forall k \in [loc+1..n]$
- Deci subintervalele ce trebuie procesate în continuare sunt $A[1..loc-1]$ și $A[loc+1..n]$.

Procedura `Partition2`

Dăm mai jos noua procedură de partiționare.

- La parcurgeri ea va lăsa pe loc valorile egale cu pivotul.
- Indicele *loc* ține minte tot timpul componenta pe care se află valoarea pivot.
- La sfârșit o transmite procedurii apelante pentru a putea fi calculate noile capete ale subvectorilor rezultați.

```
procedure Partition2 (Left, Right, loc)
    // loc este indicele pe care se va plasa în final valoarea x = A[Left]
    // inițializarea indicilor i și j pentru parcurgerile de la stânga la dreapta, respectiv de la
dreapta la stânga
   i:= Left; j:= Right;
   loc:= Left                                                    // inițializarea indicelui loc
   while i < j do
       // parcurgerea de la dreapta la stânga, urmată de interschimbare
      while (A[loc] ≤ A[j]) and (j ≠ loc) do
         j:= j-1
      endwhile
      if A[loc] > A[j] then
         Interschimbă(A[loc], A[j])
         loc:= j
      endif
       // parcurgerea de la stânga la dreapta, urmată de interschimbare
      while (A[i] ≤ A[loc]) and (i ≠ loc) do
         i:= i +1
      endwhile
      if A[i] > A[loc] then
         Interschimbă (A[loc], A[i])
         loc:= i
      endif
   endwhile
endproc
```

Partiția Lomuto

Avantaje față de `Partition2`

- Pivot într-o extremitate, dar parcurge restul, făcând **separarea valorilor, într-un singur sens**.

Partiția Lomuto

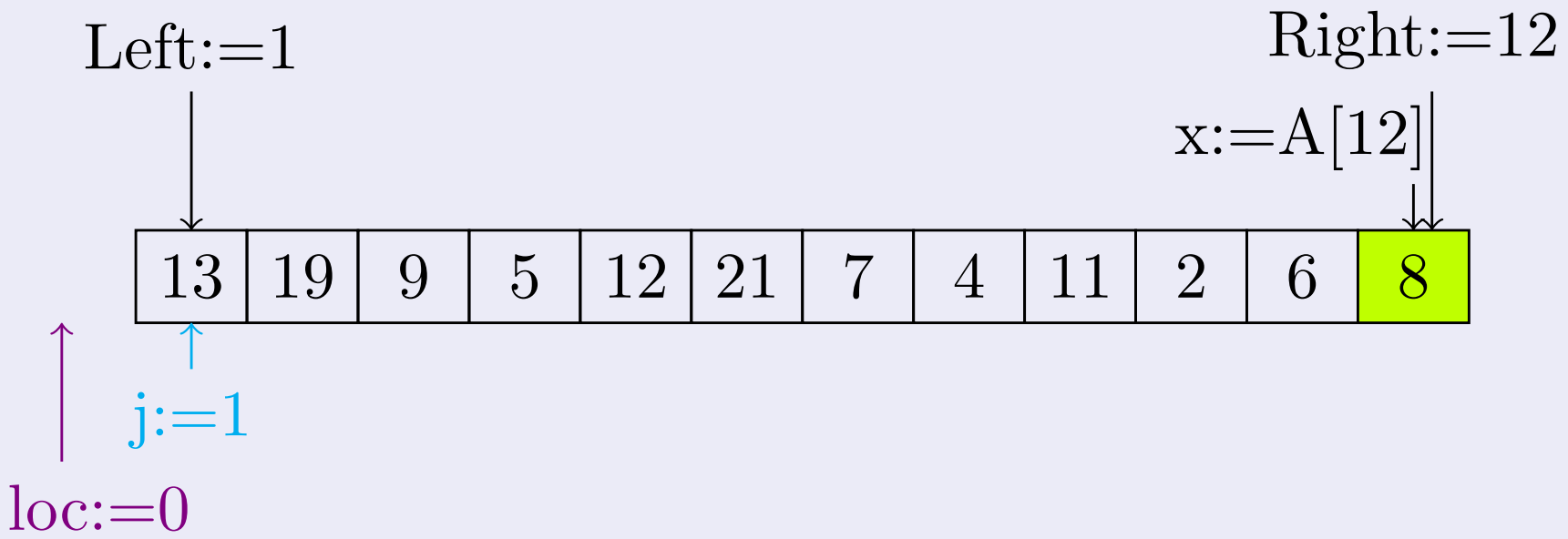
Avantaje față de `Partition2`

- Pivot într-o extremitate, dar parcurge restul, făcând **separarea valorilor, într-un singur sens**.
- Reducerea numărului de interschimbări: Lomuto face $m + 1$, față de $2m$ făcute de `Partition2`.

Partiția Lomuto

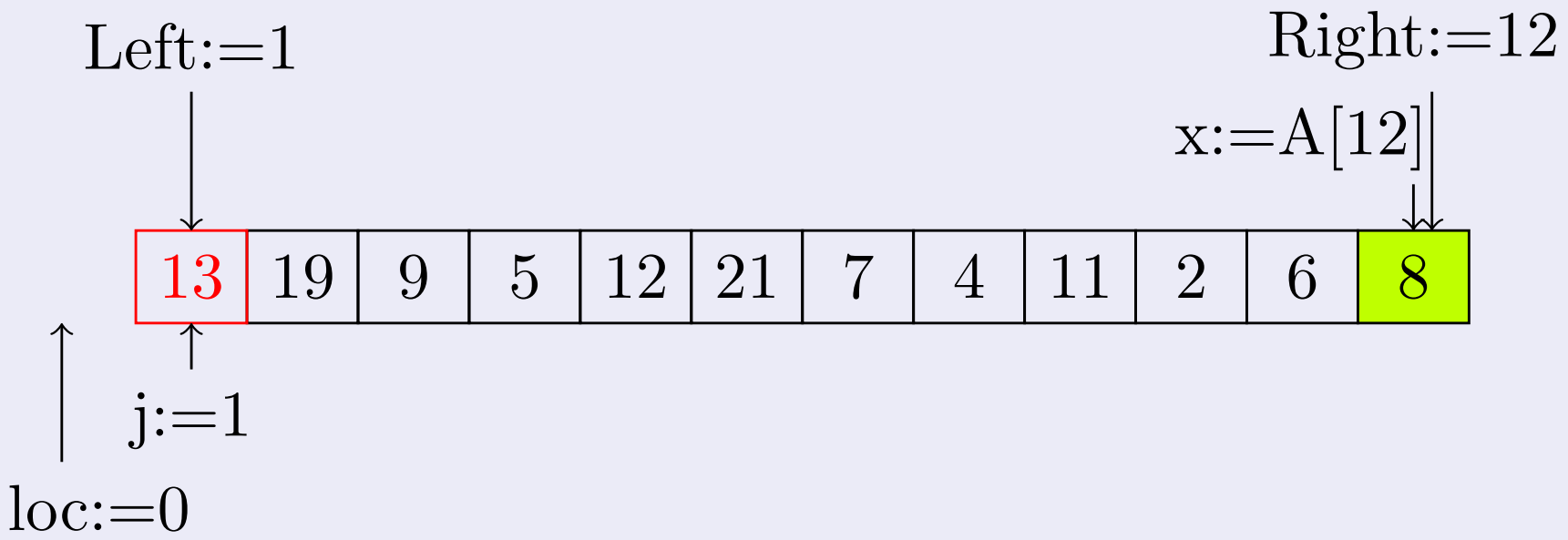
```
procedure PartitionLomuto(Left, Right, loc)
  x:=A[Right];                          // algerea pivotului în într-o extremitate
   // loc este indicele pe care se va plasa în final valoarea x = A[Right]
  loc:=Left-1;
   // inițializarea indicelui j pentru parcurgerea de la stânga la dreapta (un
singur sens)
  j:= Left;
  while j ≤ Right do
    if A[j] ≤ x then
      loc:=loc+1;
      Interschimbă(A[loc],A[j]);
    endif
    j:=j+1;
  endwhile
  if loc ≥ Right then
    loc:=loc-1;
  endif
endproc
```

Exemplu:



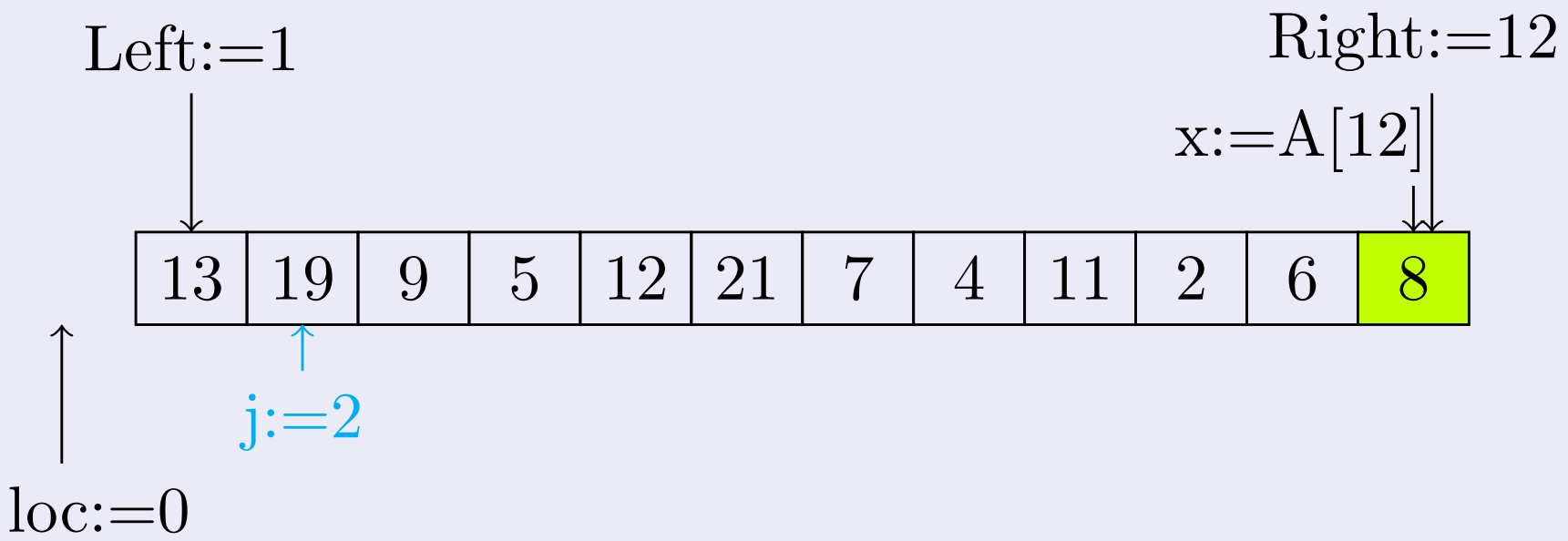

```
x:=A[Right];
loc:=Left-1; j:= Left
```

Exemplu:



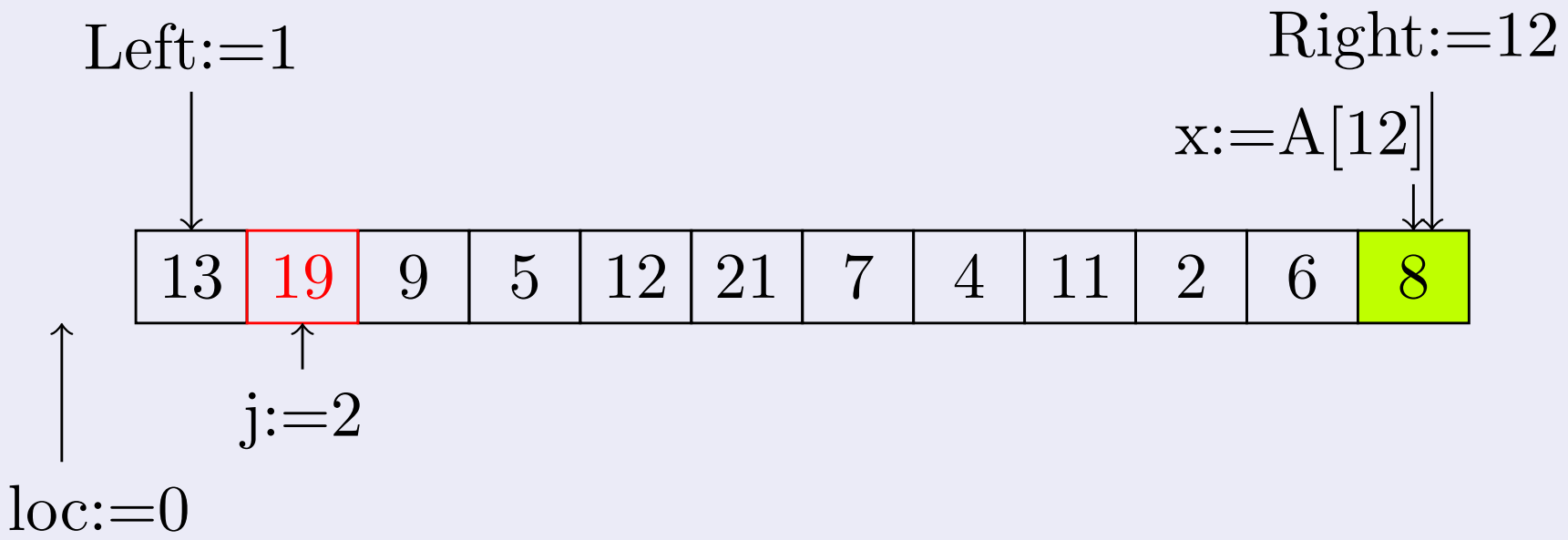
`A[1] > x.`

Exemplu:



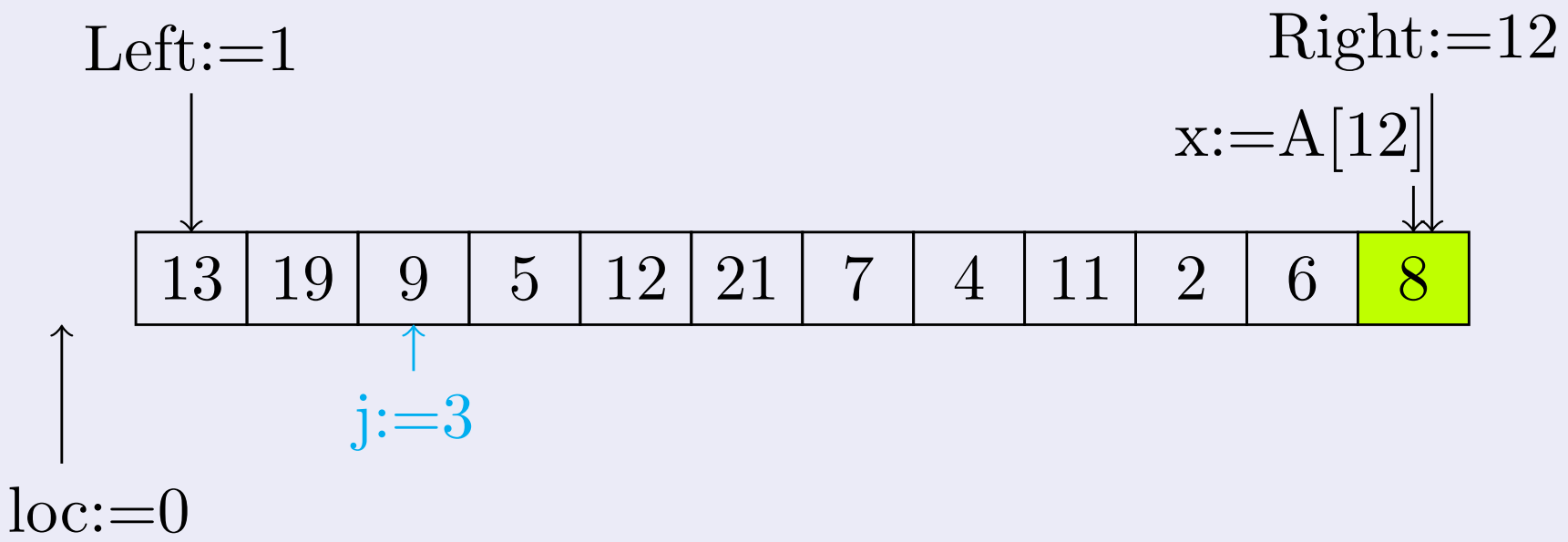
Avansam cu `j`.

Exemplu:



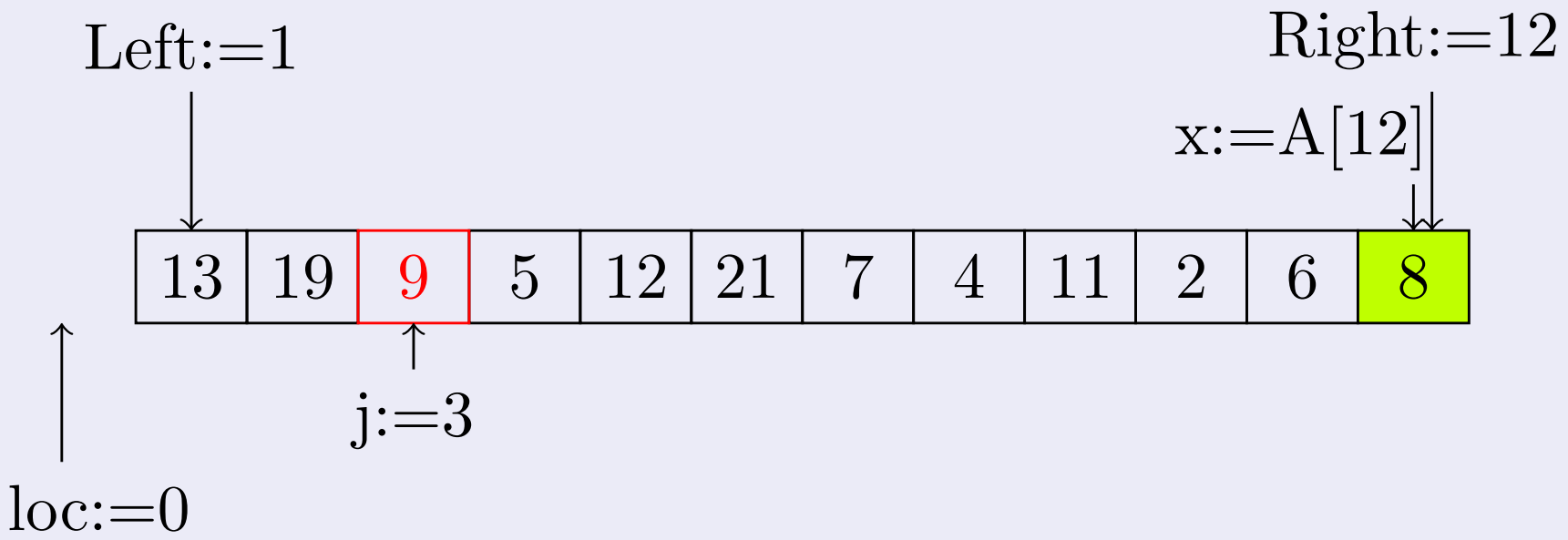
`A[2] > x.`

Exemplu:



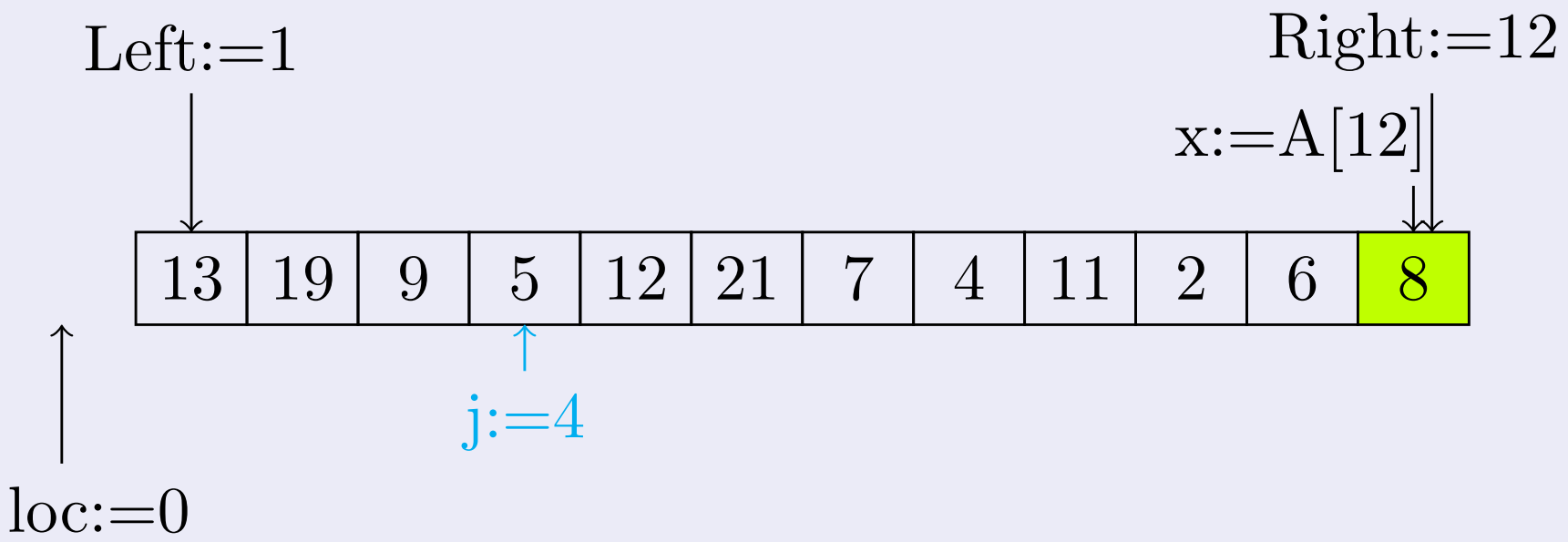
Avansam cu `j`.

Exemplu:



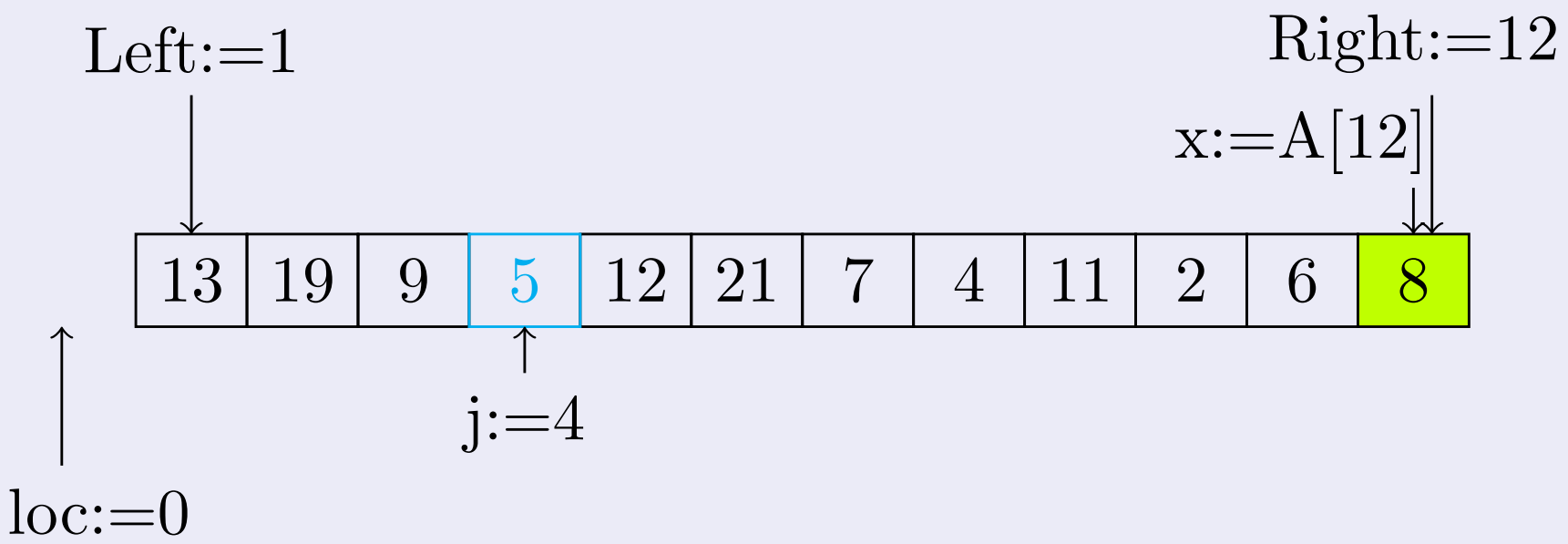
`A[3] > x.`

Exemplu:



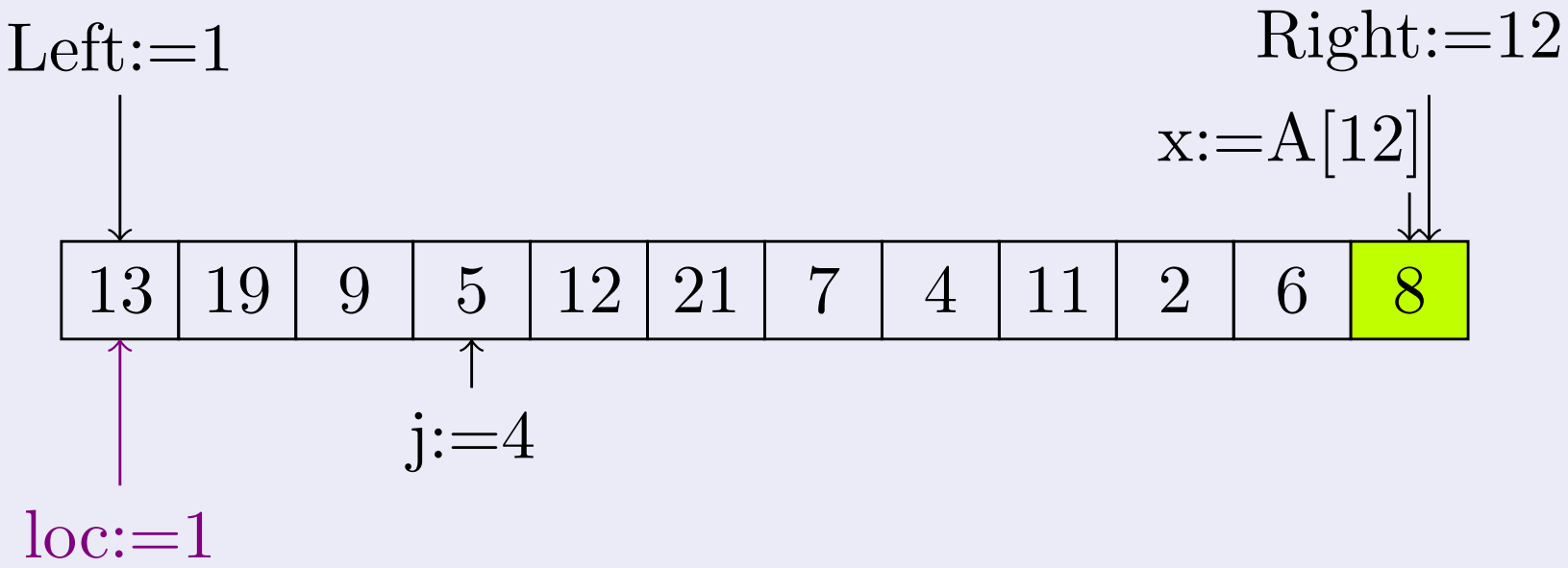
Avansam cu `j`.

Exemplu:



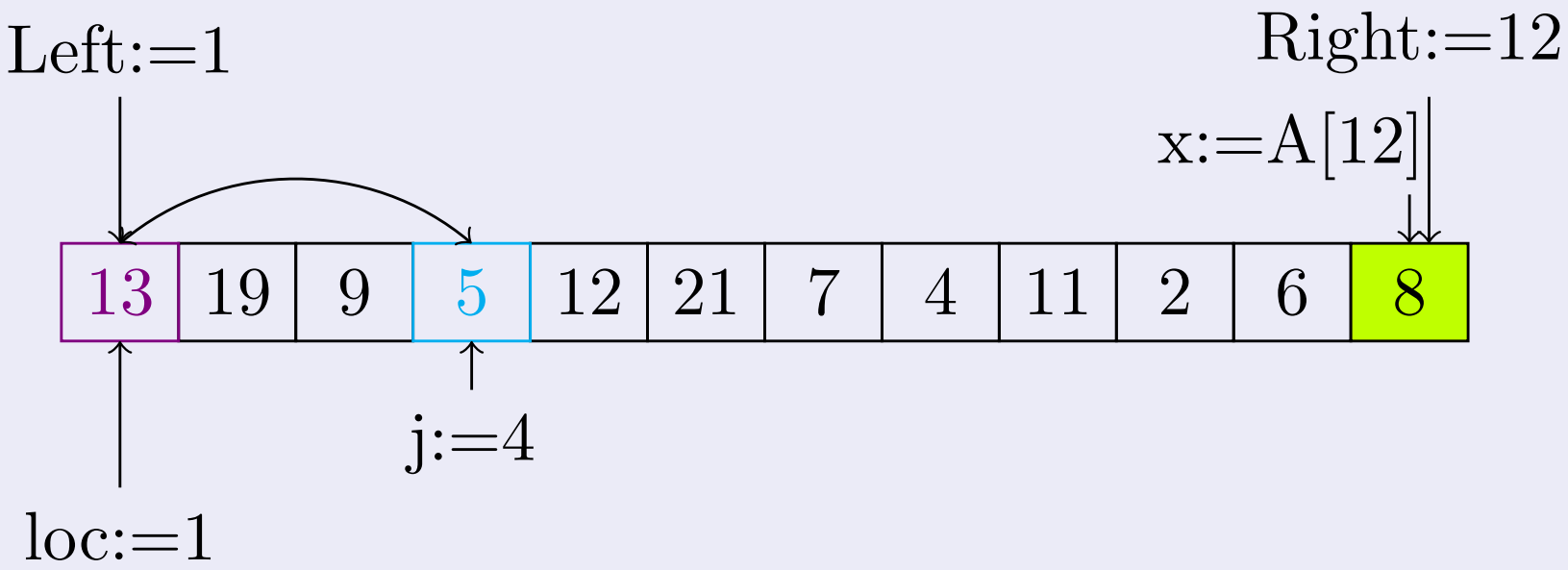
`A[4]` $\leq$ `x`.

Exemplu:



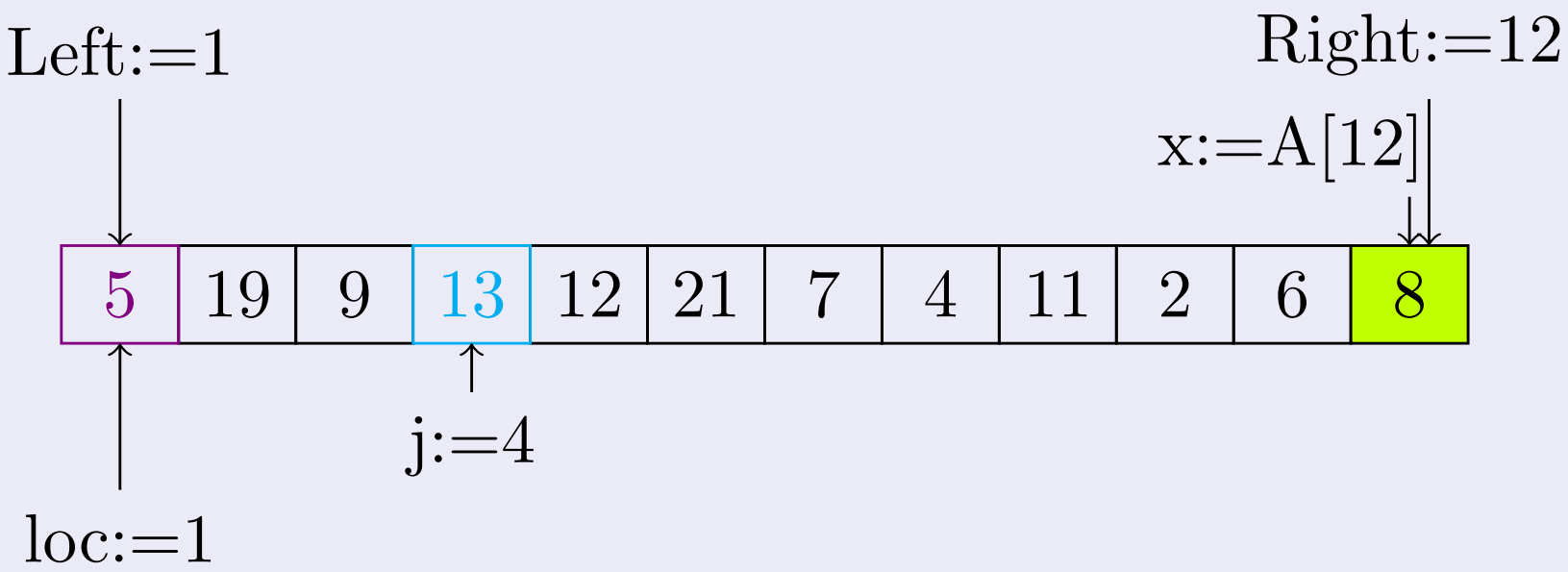
Avansăm cu `loc`.

Exemplu:



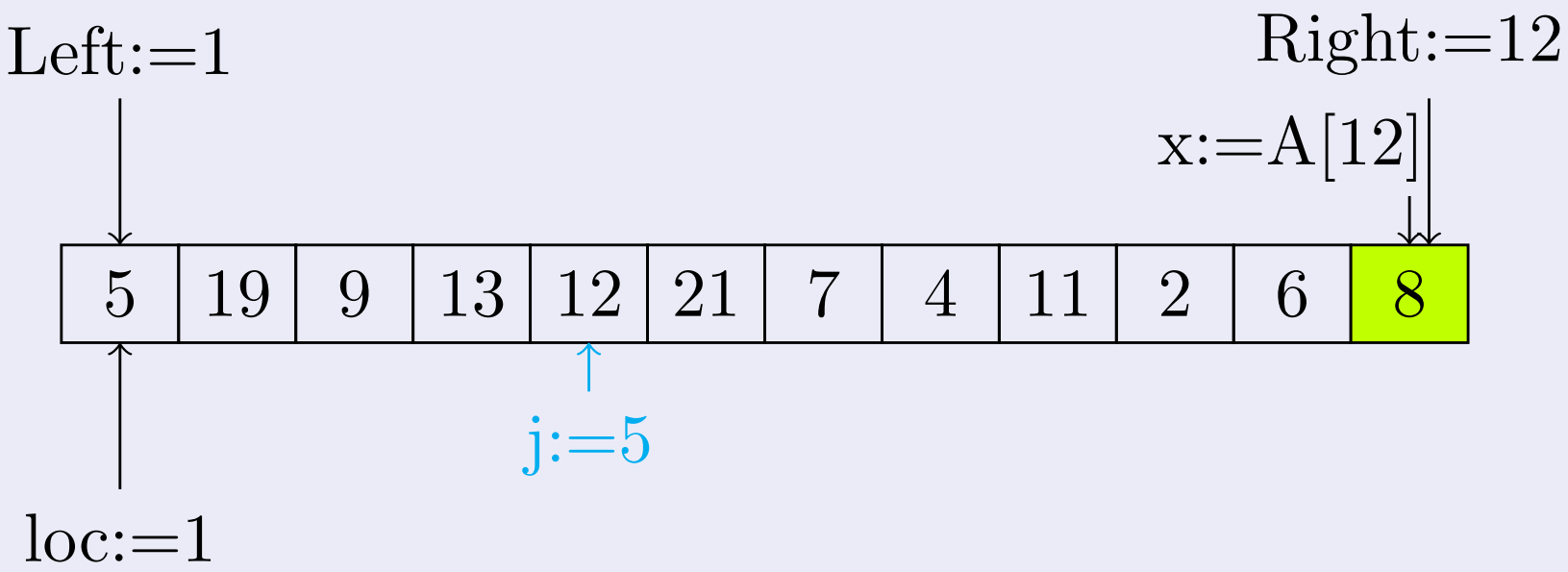
Trebuie să interschimbăm `A[1]` cu `A[4]`.

Exemplu:



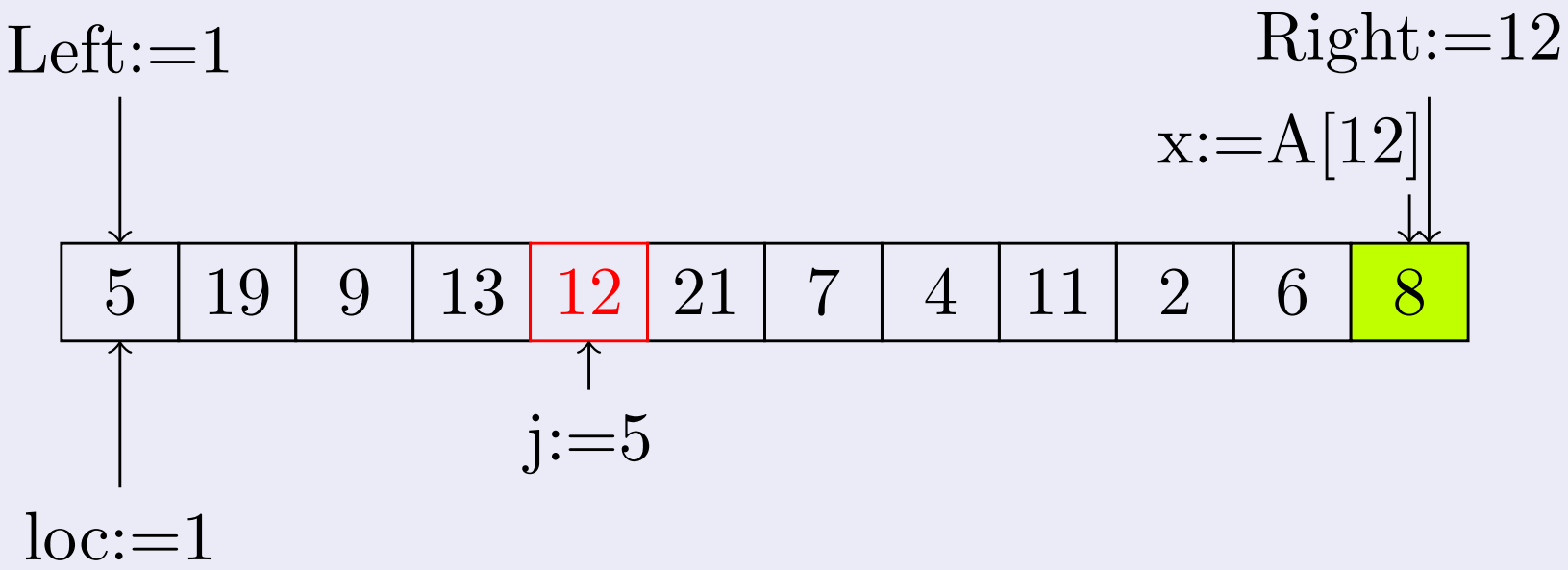
Vectorul după interschimbare.

Exemplu:



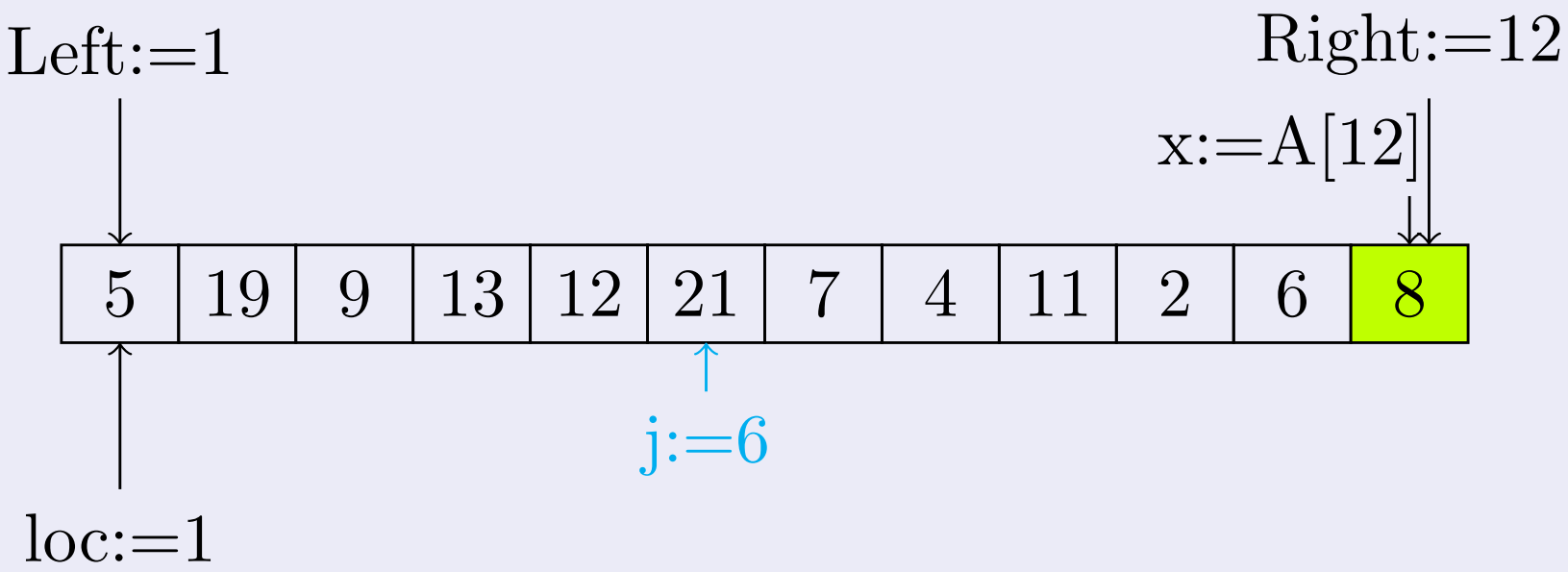
Avansăm cu `j`.

Exemplu:



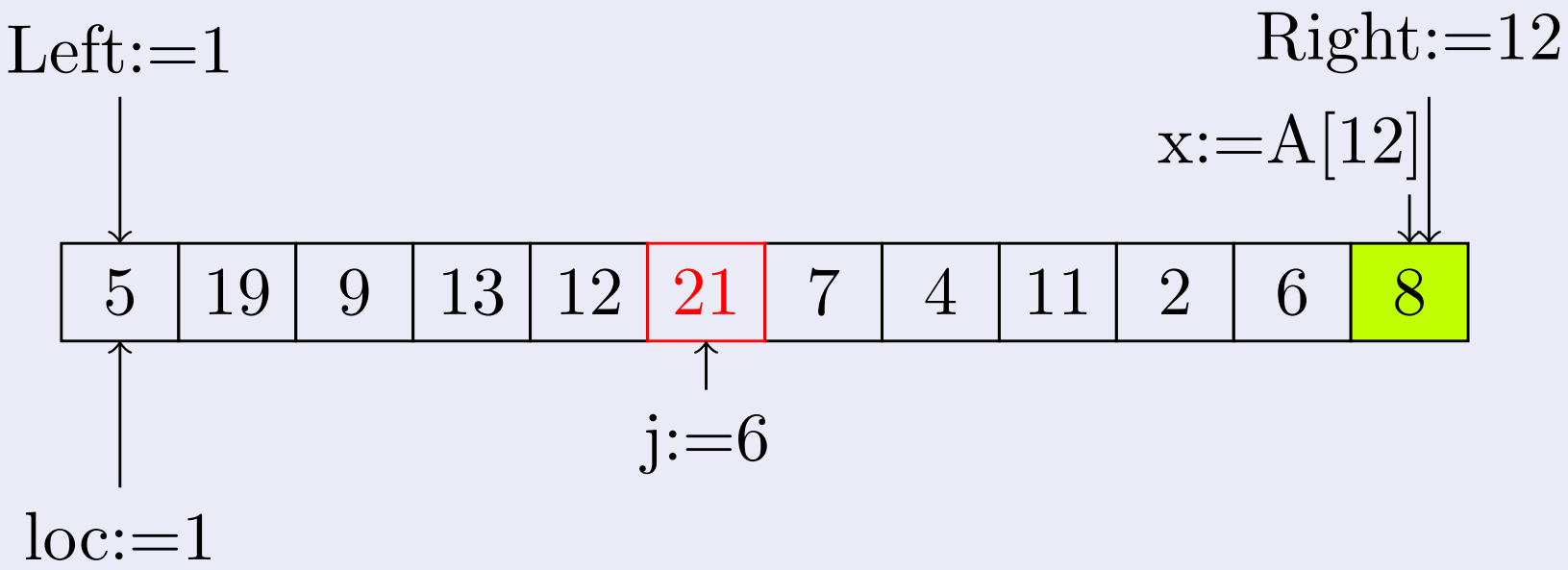
`A[5] > x.`

Exemplu:



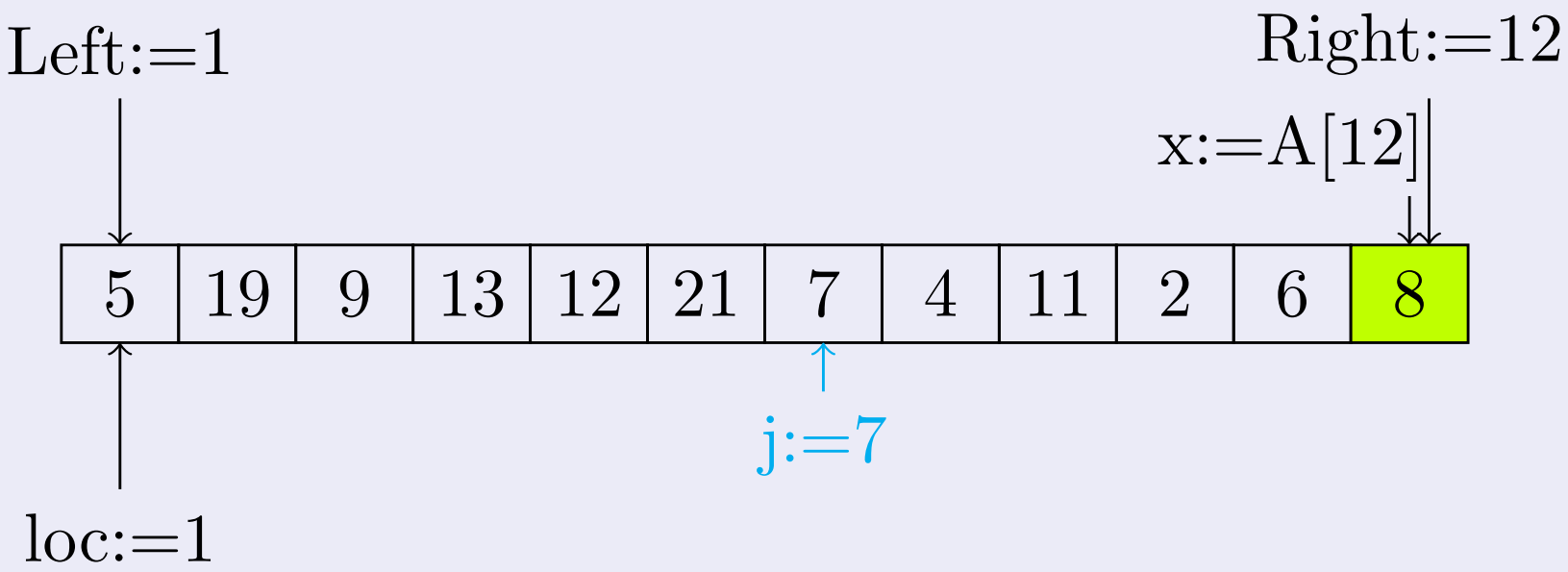
Avansăm cu `j`.

Exemplu:



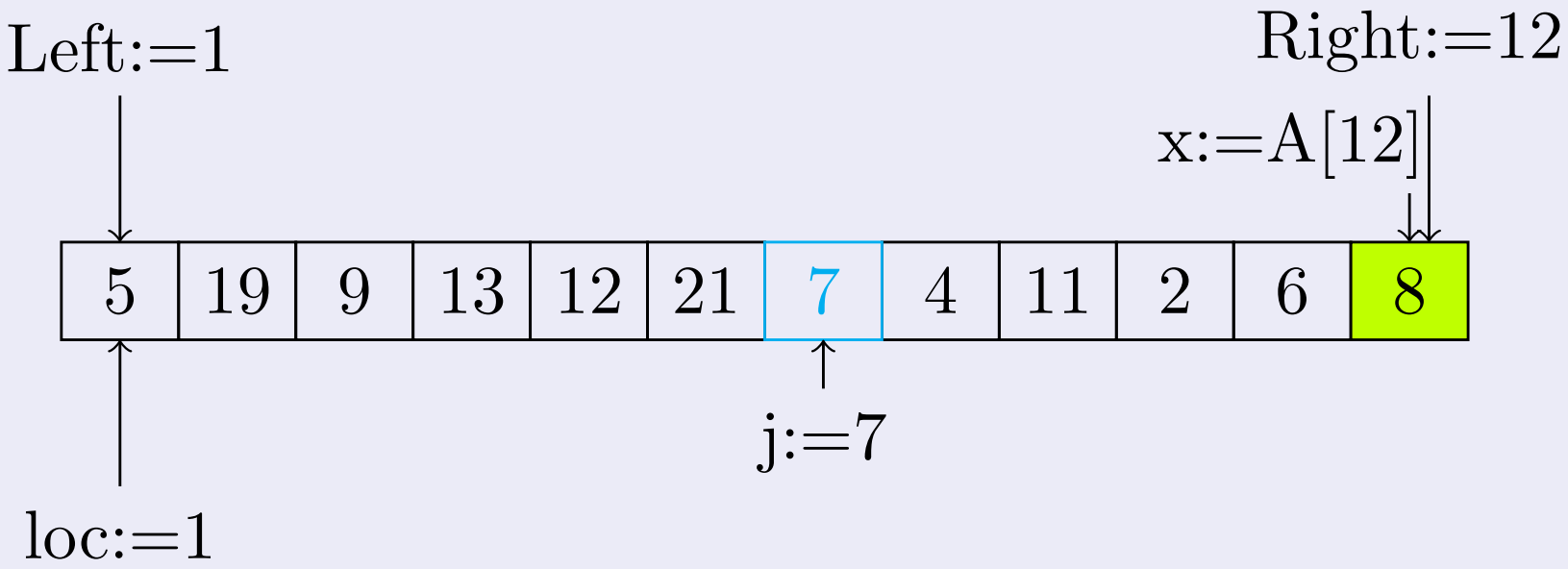
`A[6] > x.`

Exemplu:



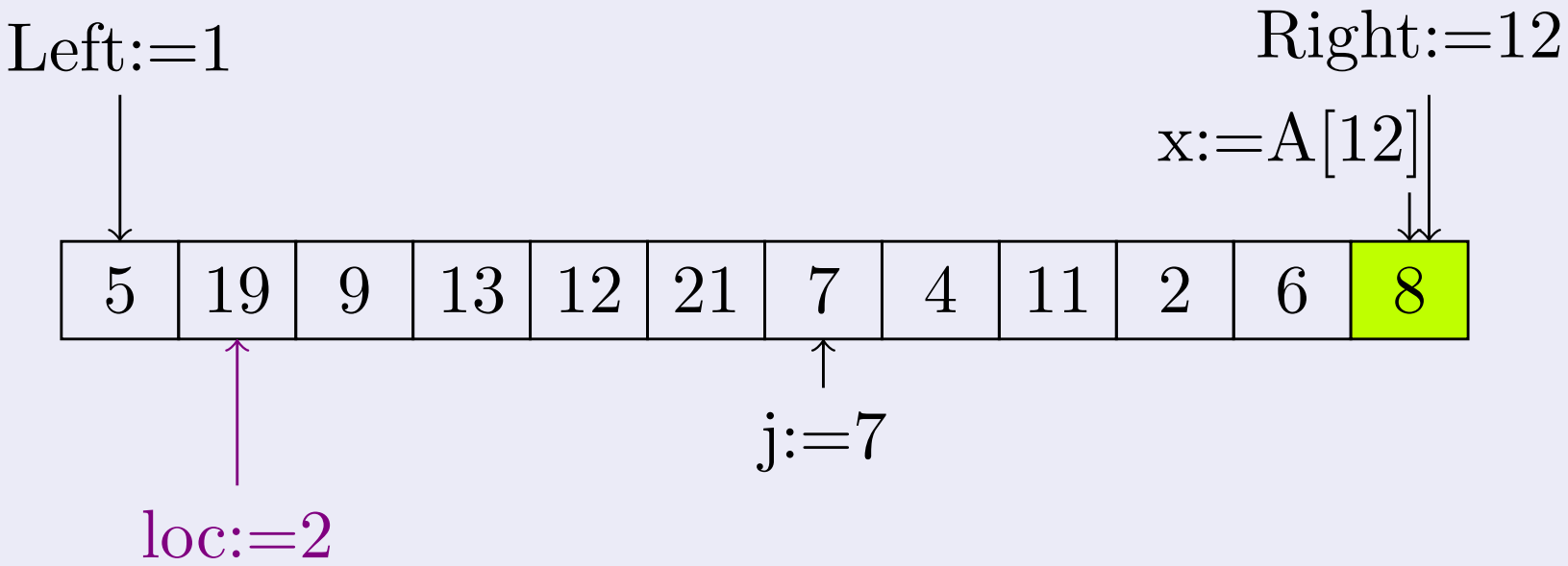
Avansăm cu `j`.

Exemplu:



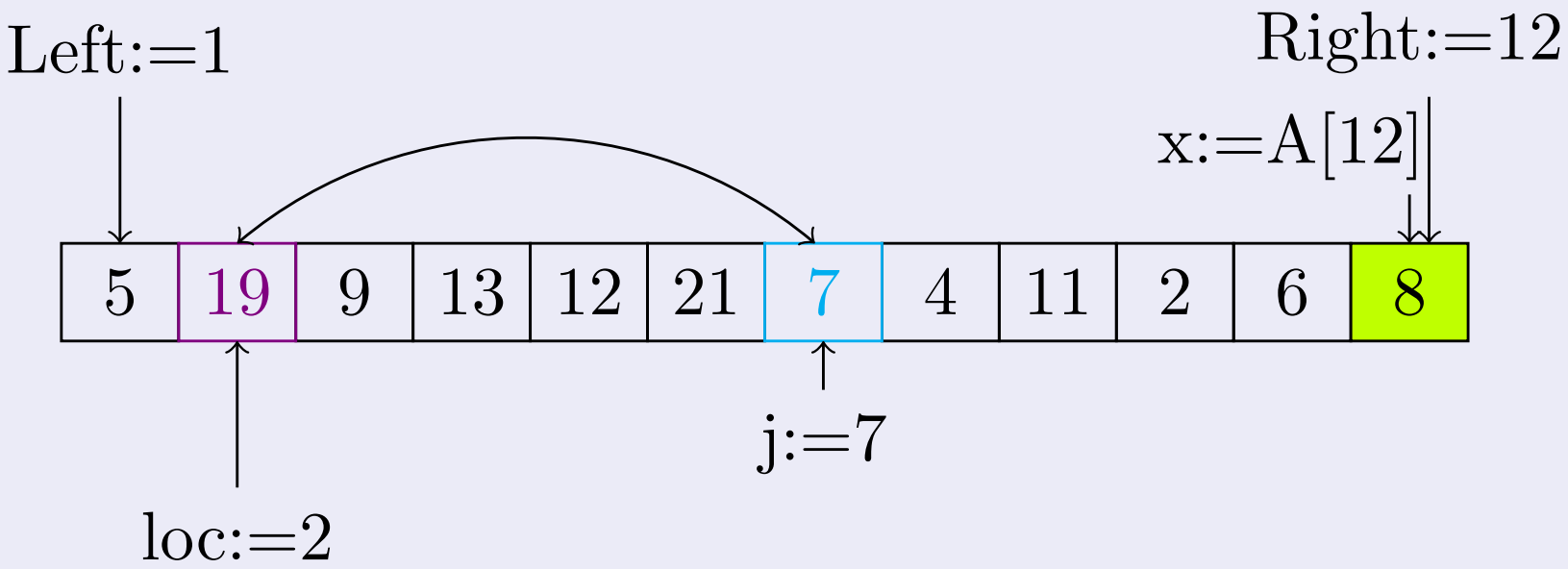
`A[7]` $\leq$ `x`.

Exemplu:



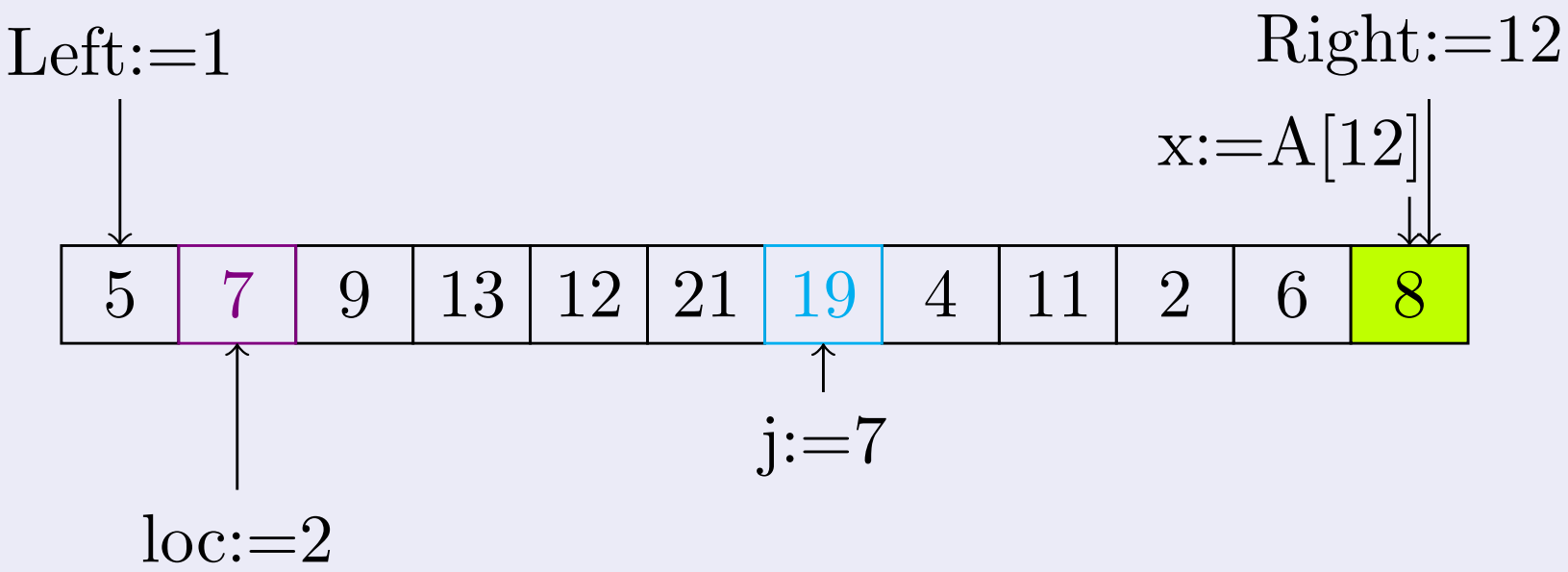
Avansăm cu `loc`.

Exemplu:



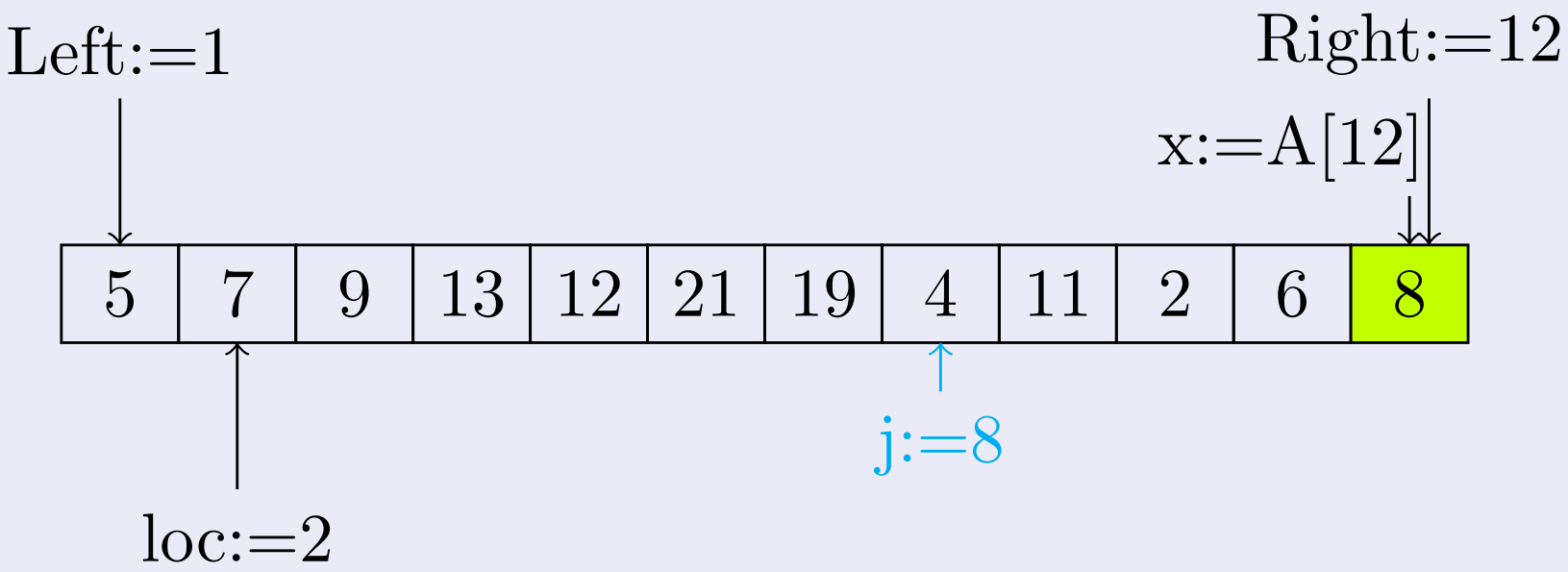
Trebuie să interschimbăm `A[2]` cu `A[7]`.

Exemplu:



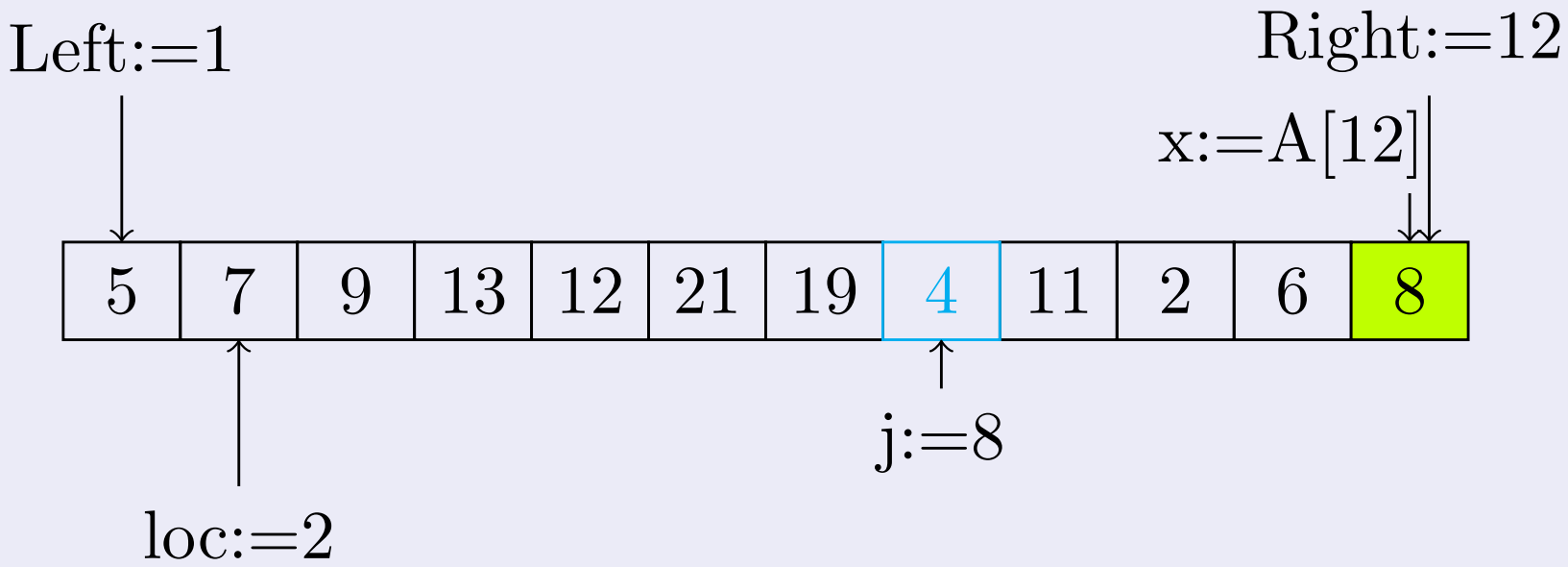
Vectorul după interschimbare.

Exemplu:



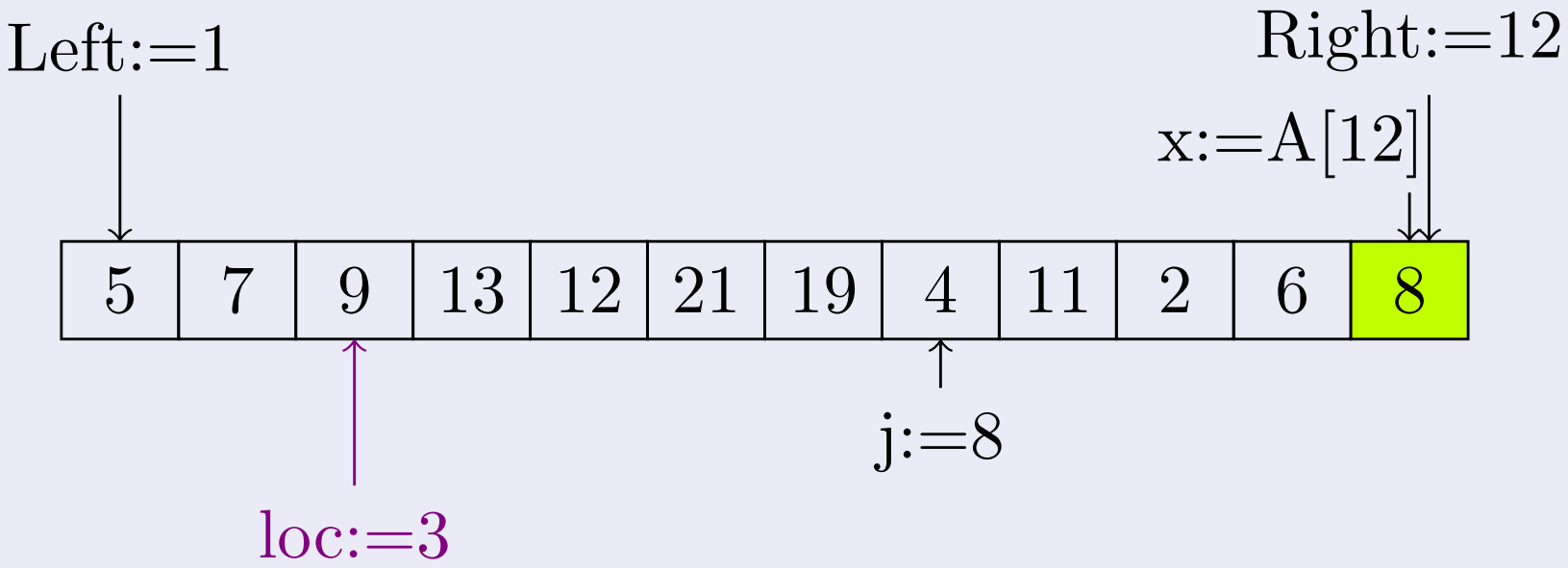
Avansăm cu `j`.

Exemplu:



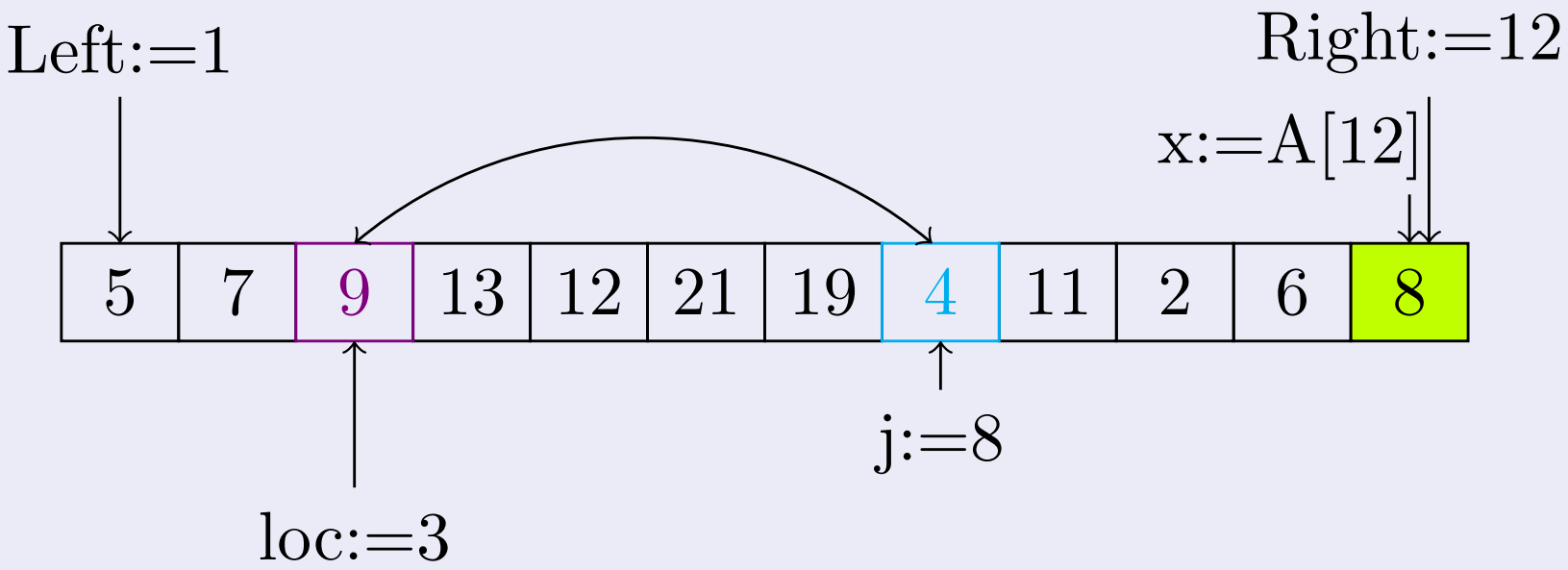
`A[8]` $\leq$ `x`.

Exemplu:



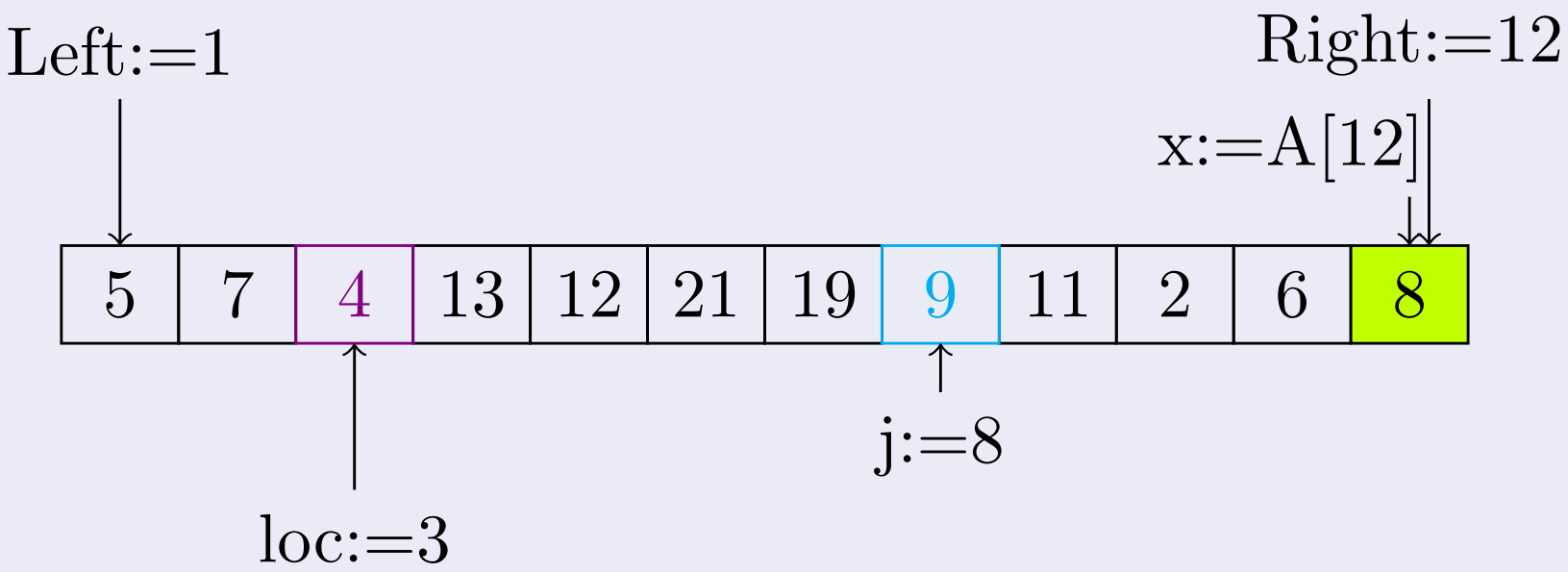
Avansăm cu `loc`.

Exemplu:



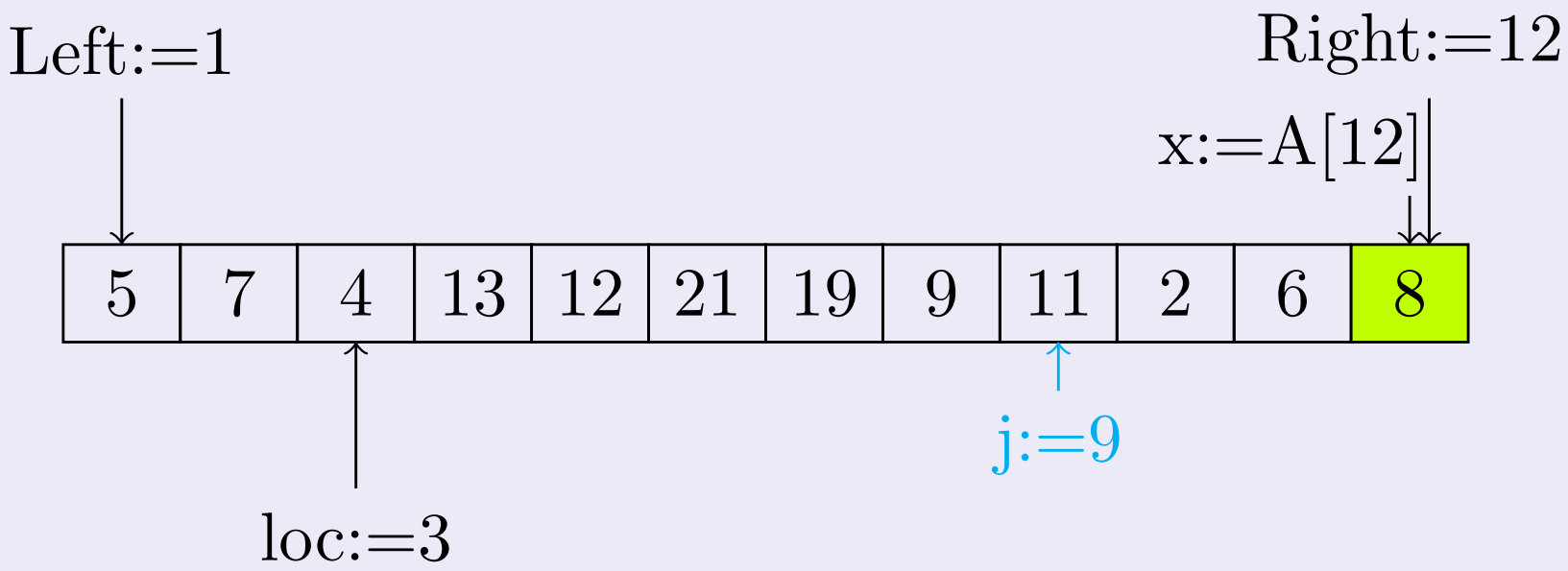
Trebuie să interschimbăm `A[3]` cu `A[8]`.

Exemplu:



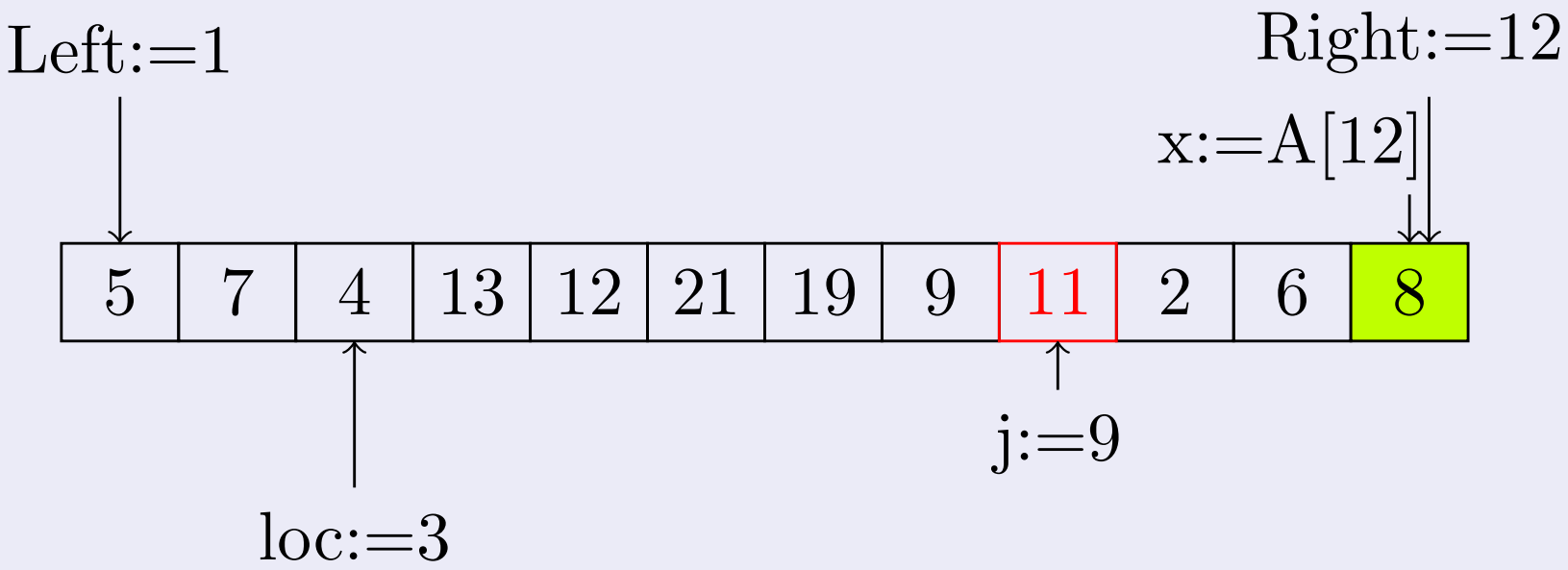
Vectorul după interschimbare.

Exemplu:



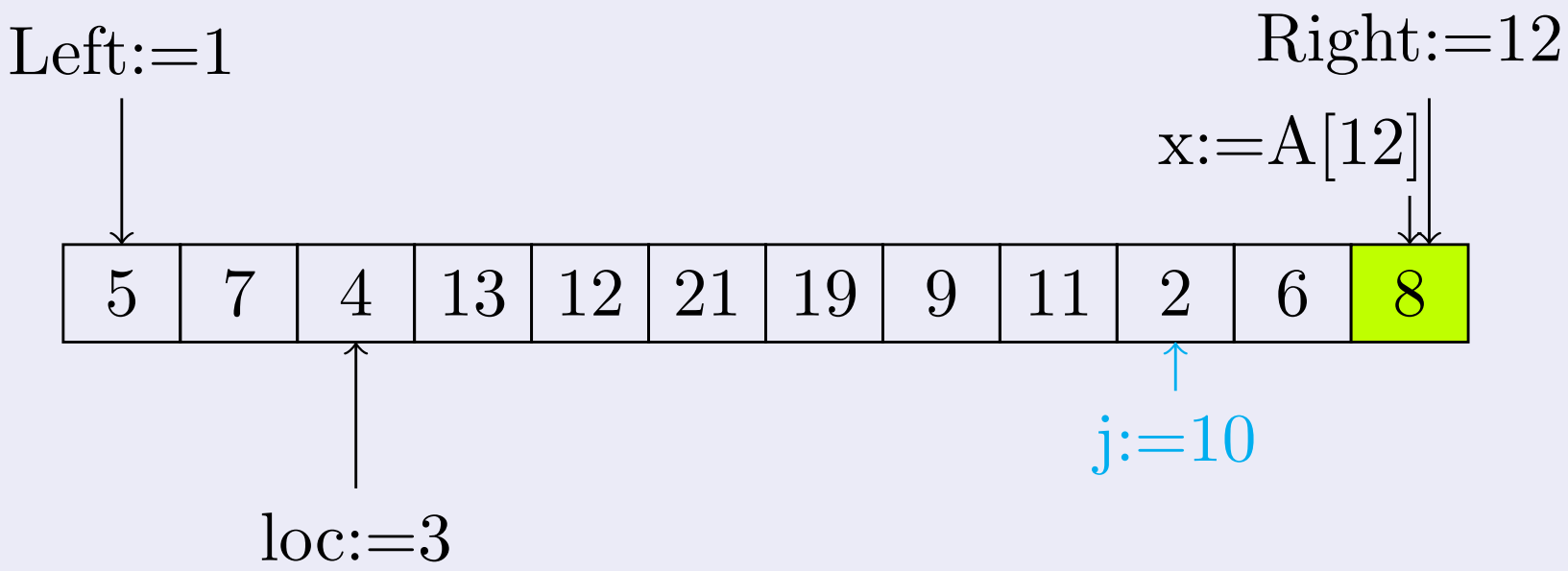
Avansăm cu `j`.

Exemplu:



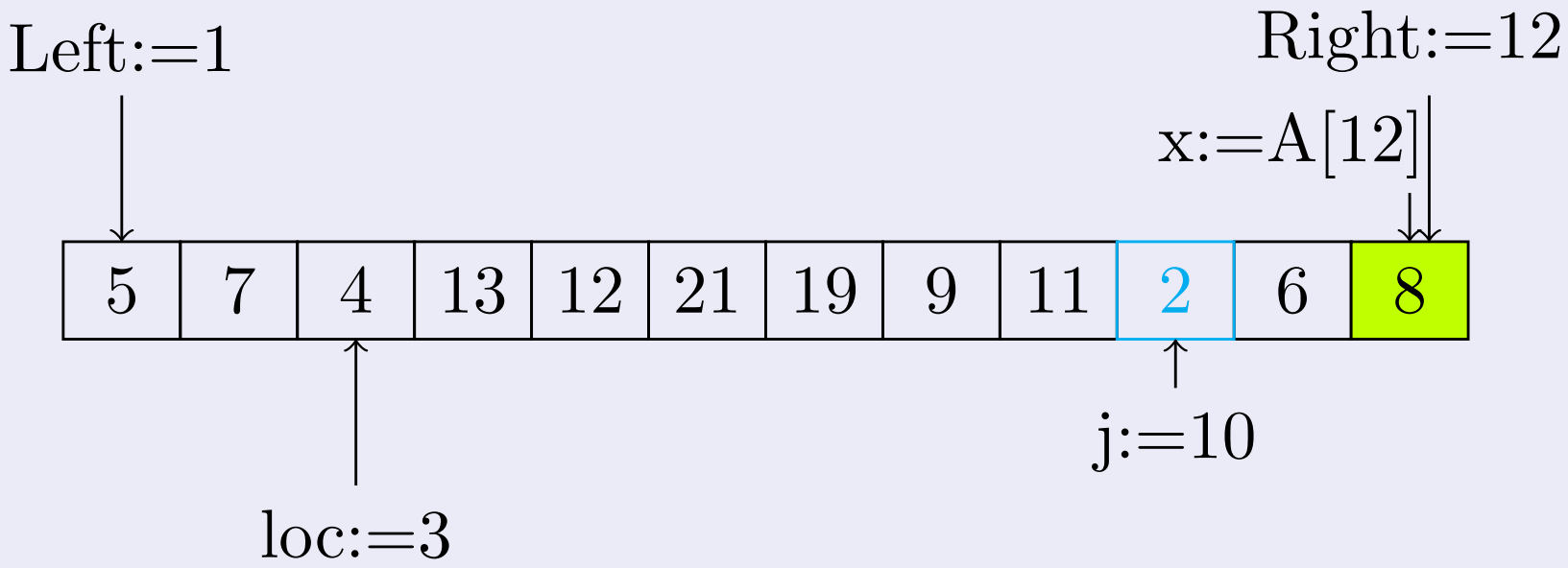
`A[9] > x.`

Exemplu:



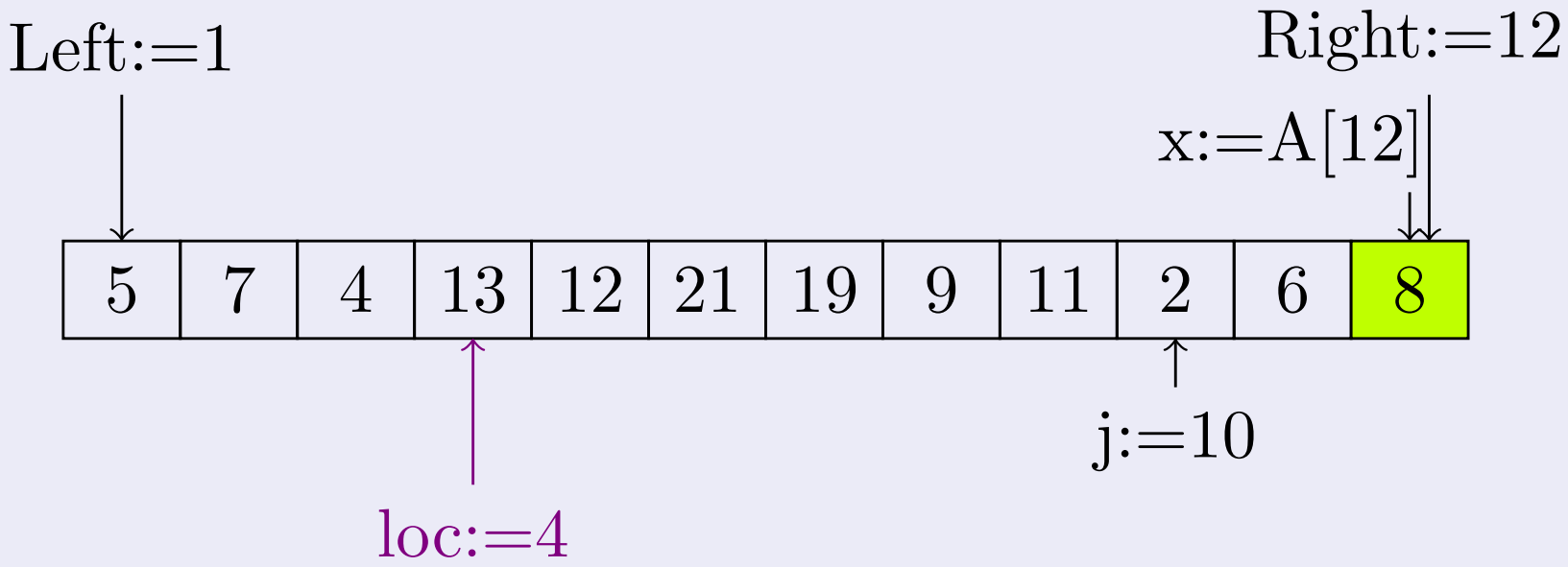
Avansăm cu `j`.

Exemplu:



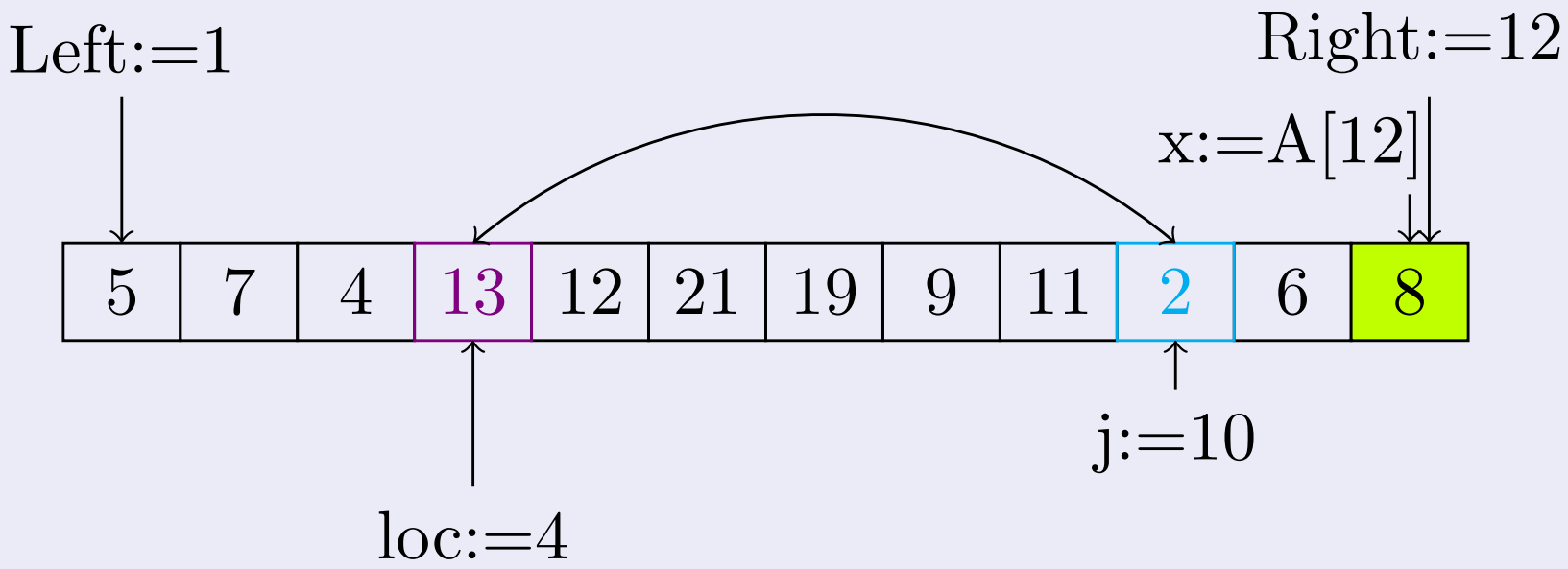
`A[10]` $\leq$ `x`.

Exemplu:



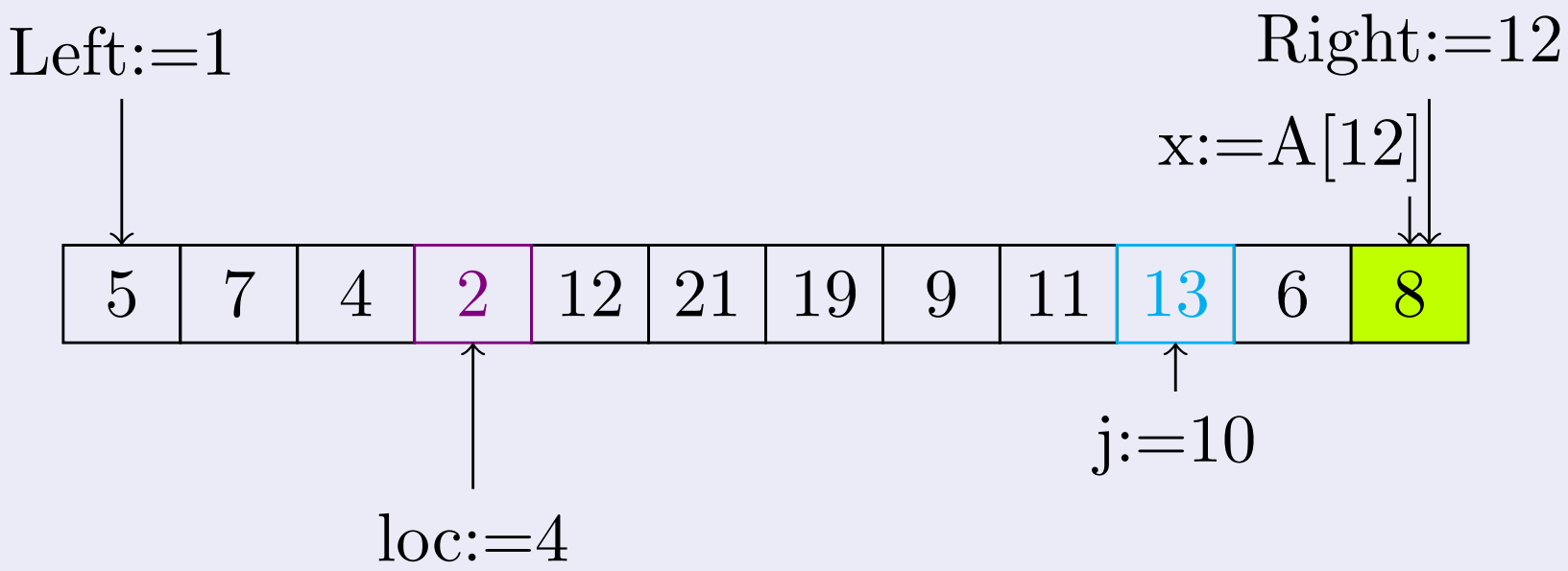
Avansăm cu `loc`.

Exemplu:



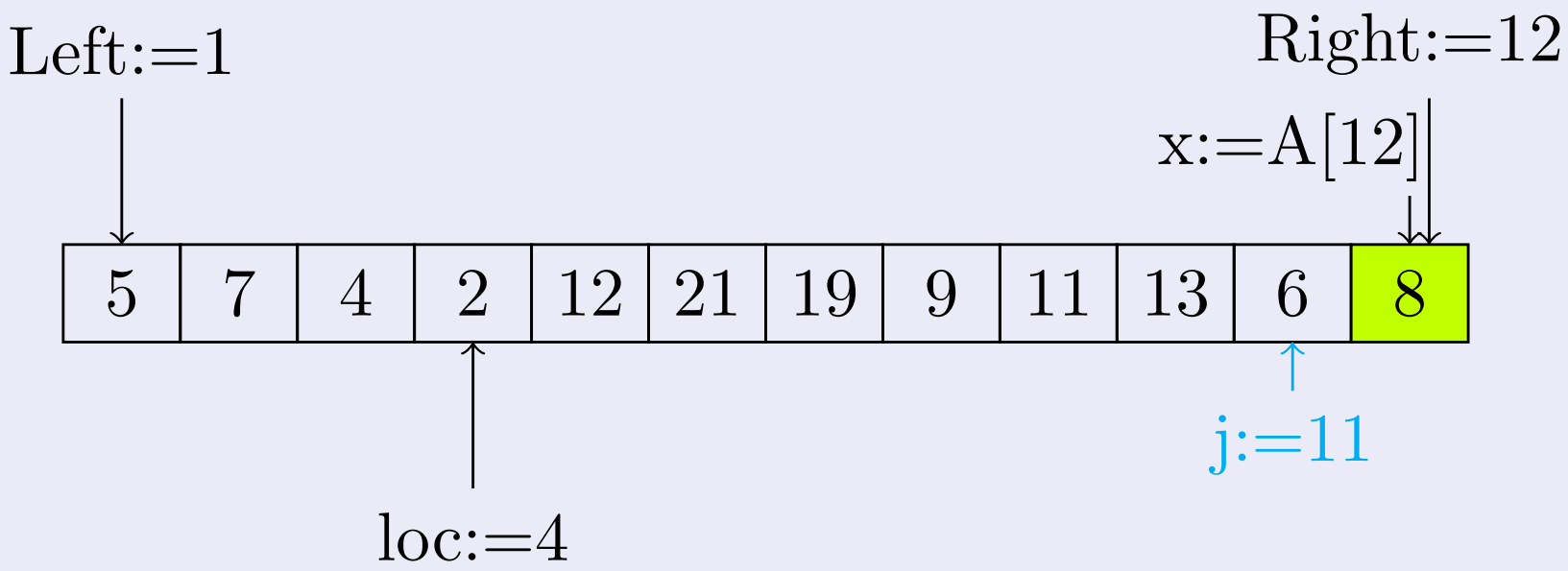
Trebuie să interschimbăm `A[4]` cu `A[10]`.

Exemplu:



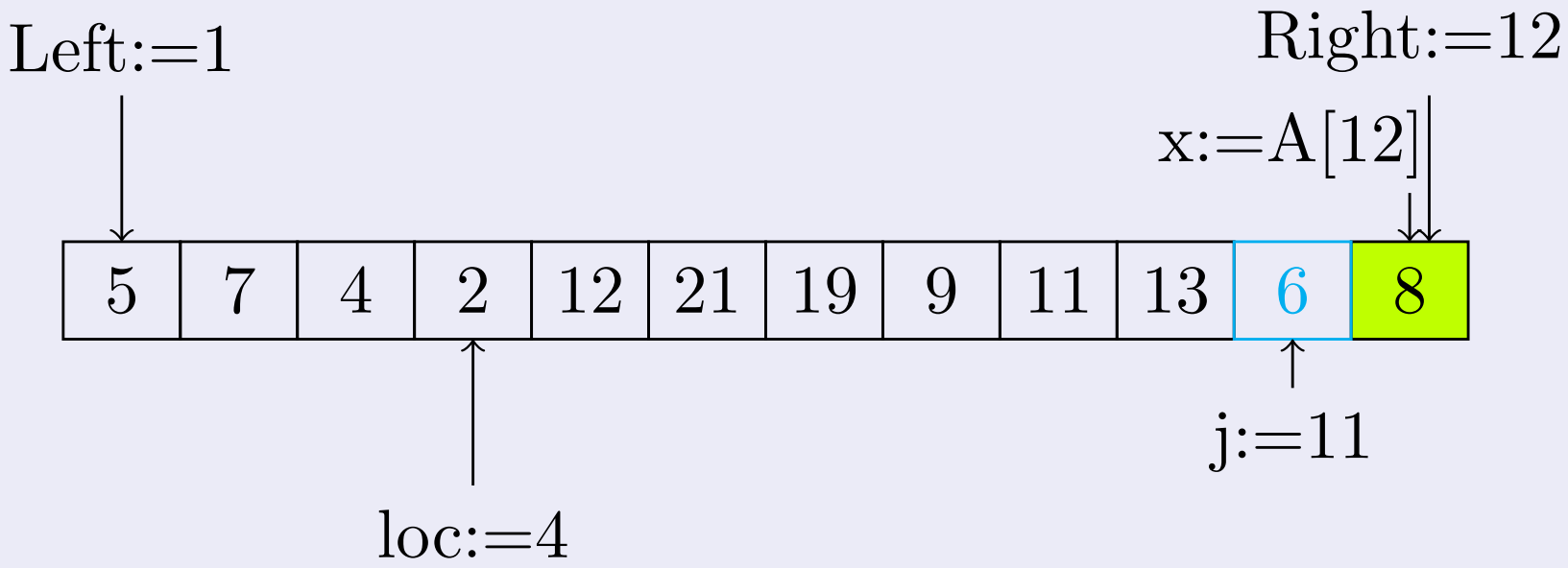
Vectorul după interschimbare.

Exemplu:



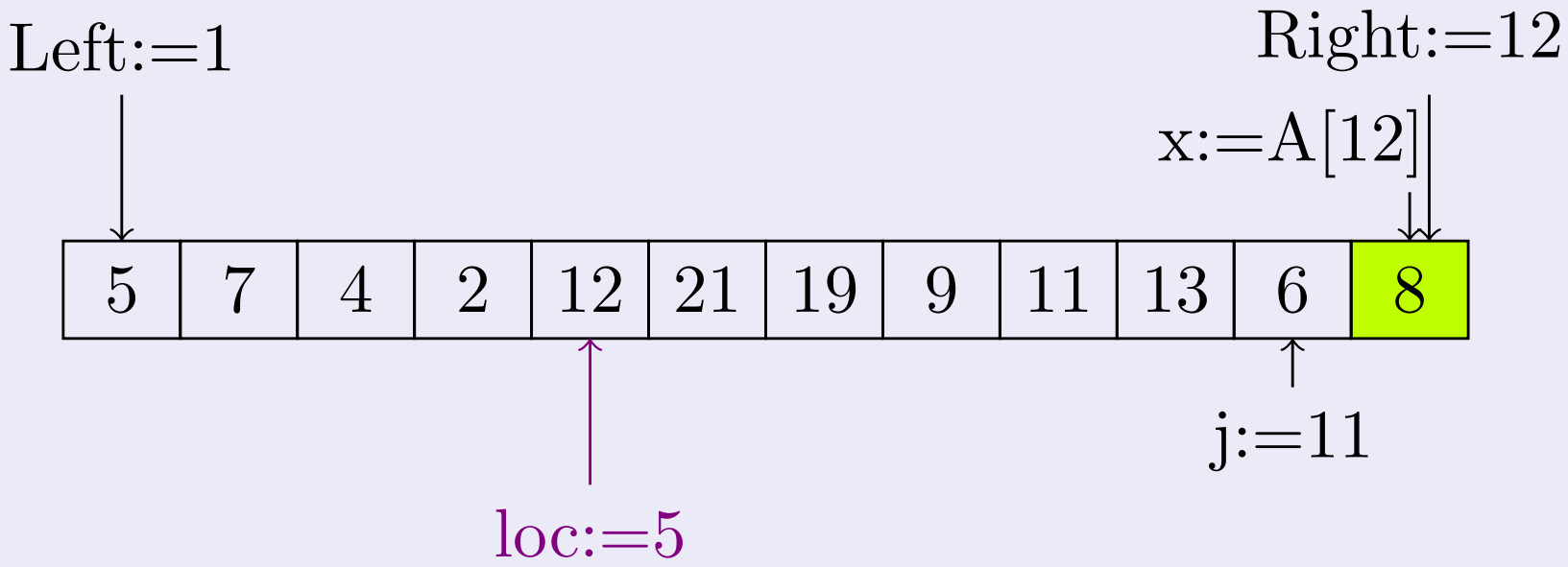
Avansăm cu `j`.

Exemplu:



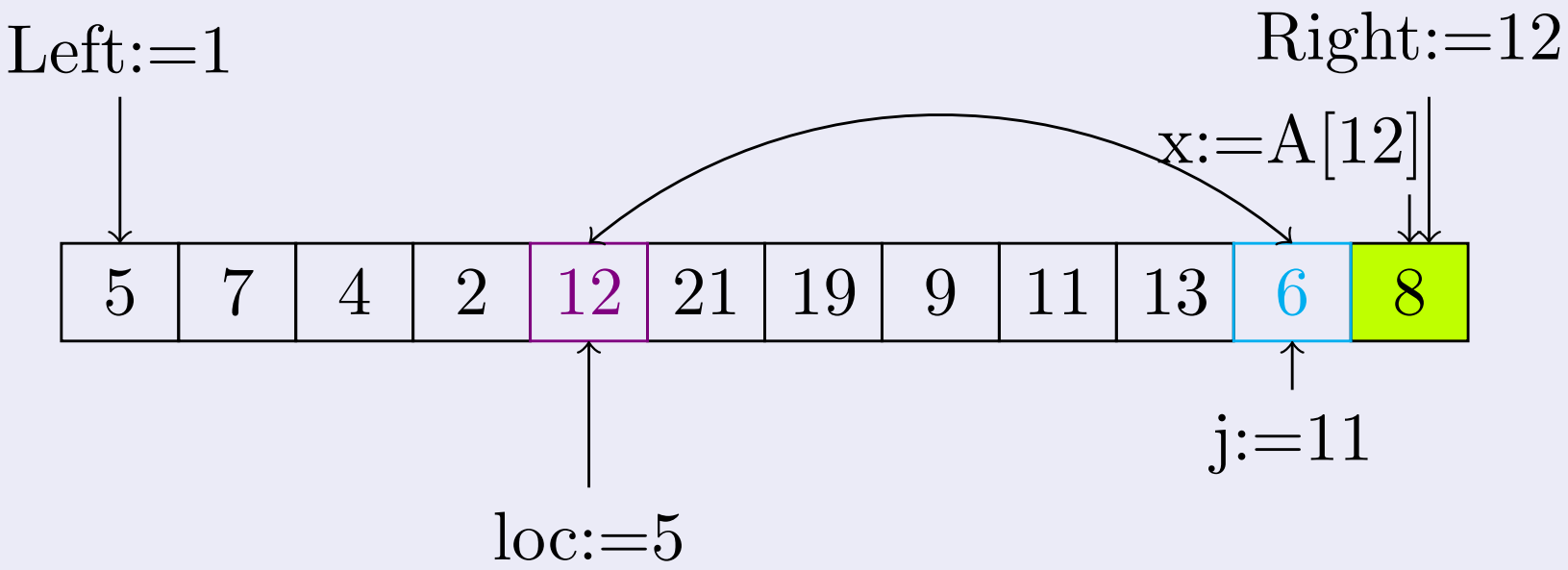
`A[11]` $\leq$ `x`.

Exemplu:



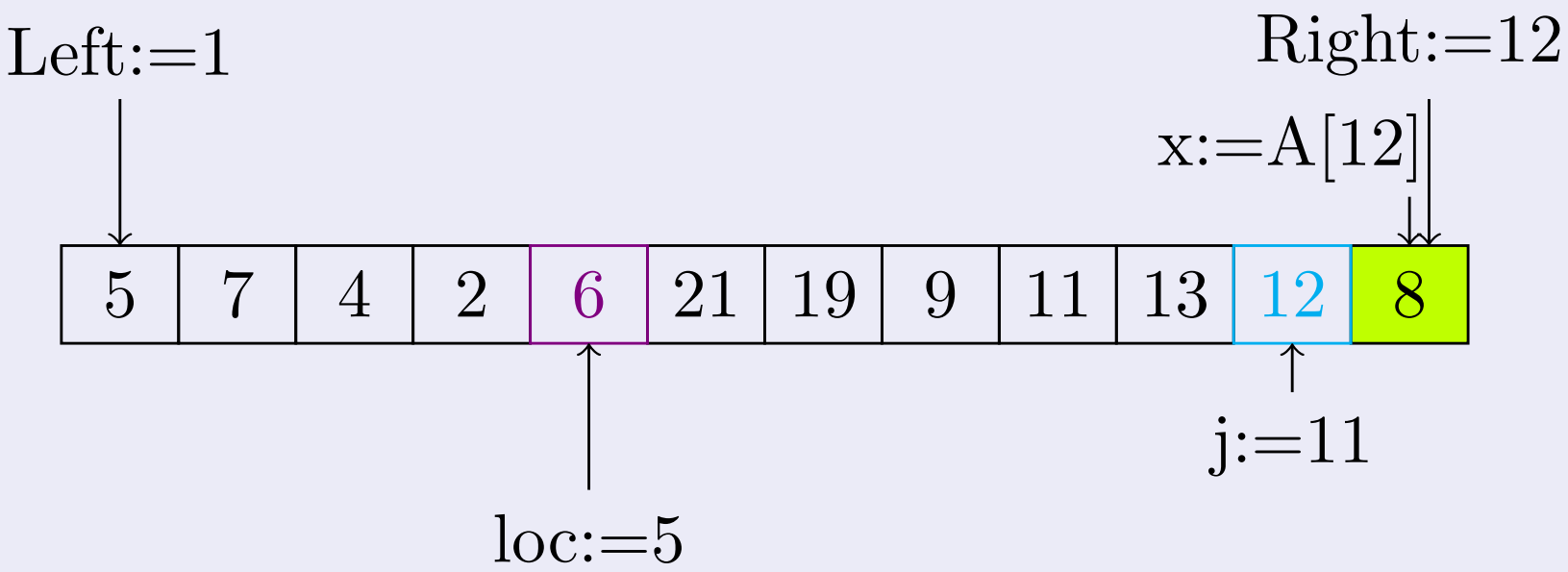
Avansăm cu `loc`.

Exemplu:



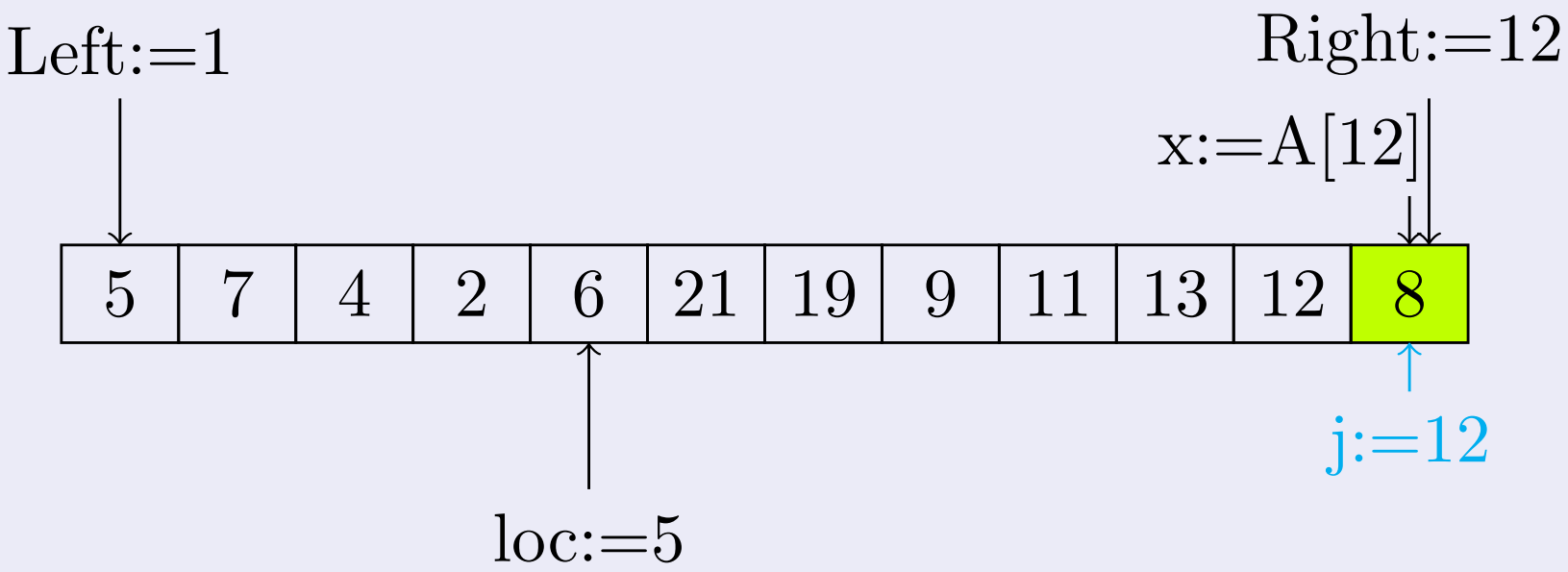
Trebuie să interschimbăm `A[5]` cu `A[11]`.

Exemplu:



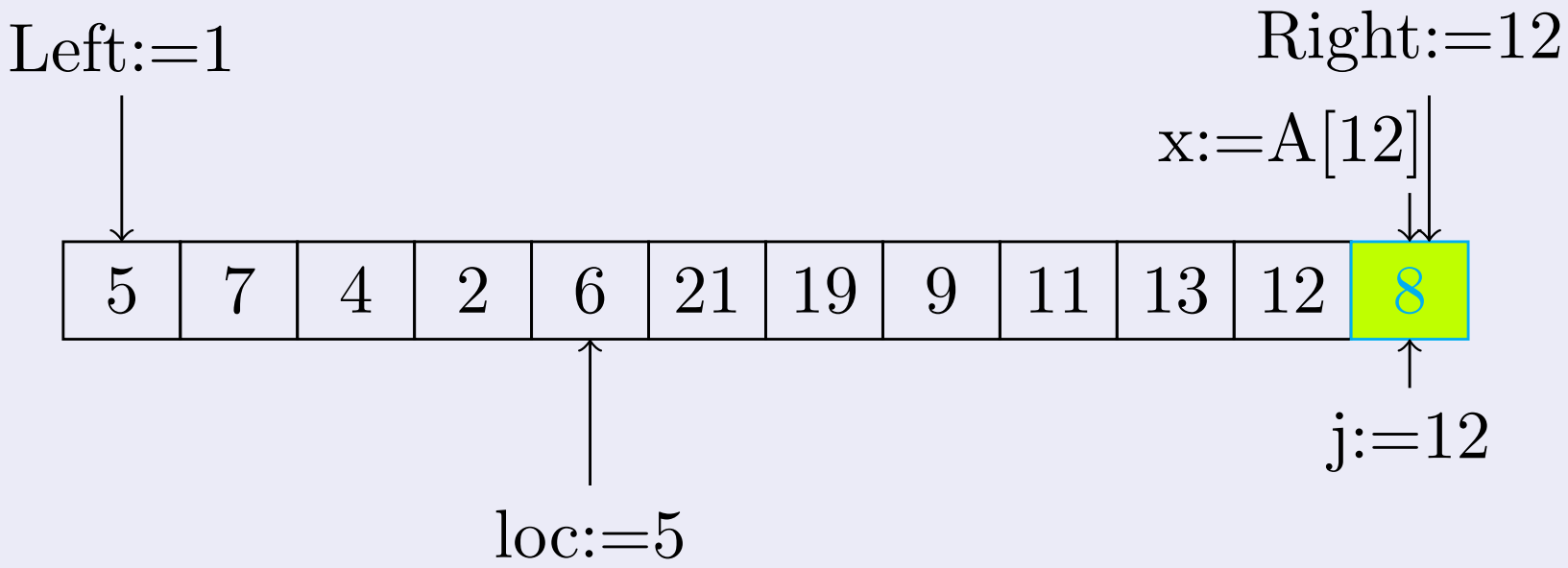
Vectorul după interschimbare.

Exemplu:



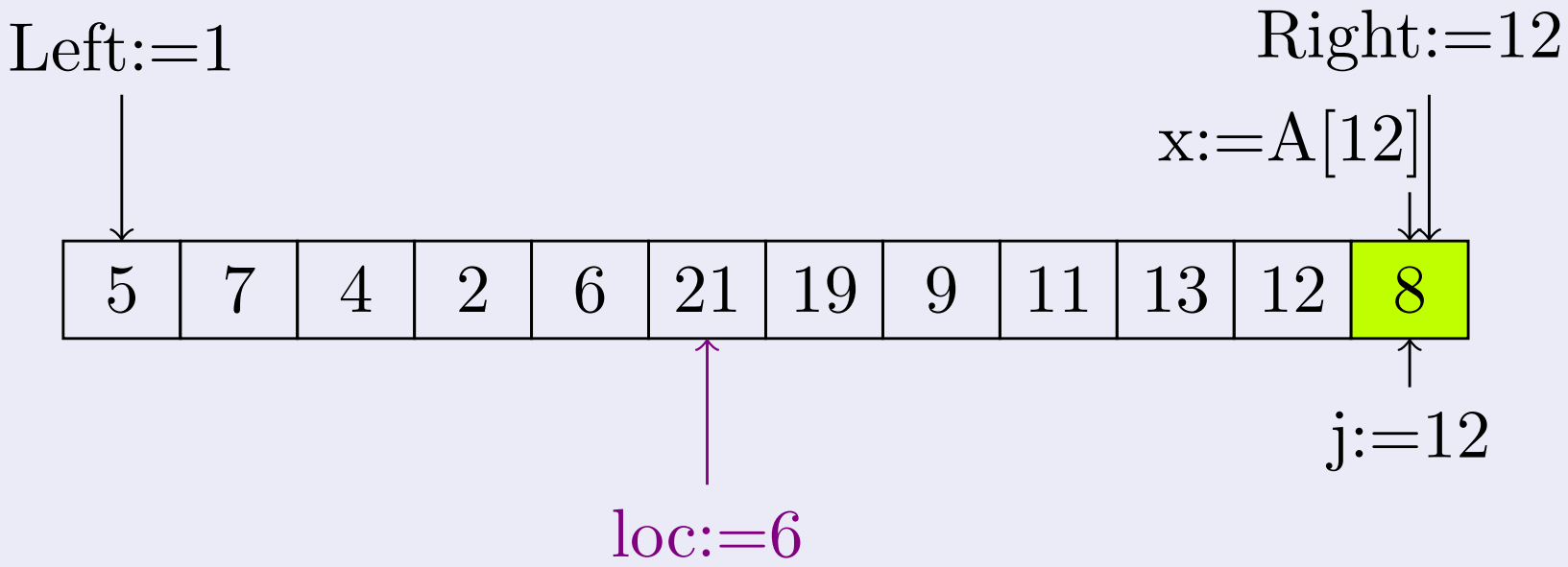
Avansăm cu `j`.

Exemplu:



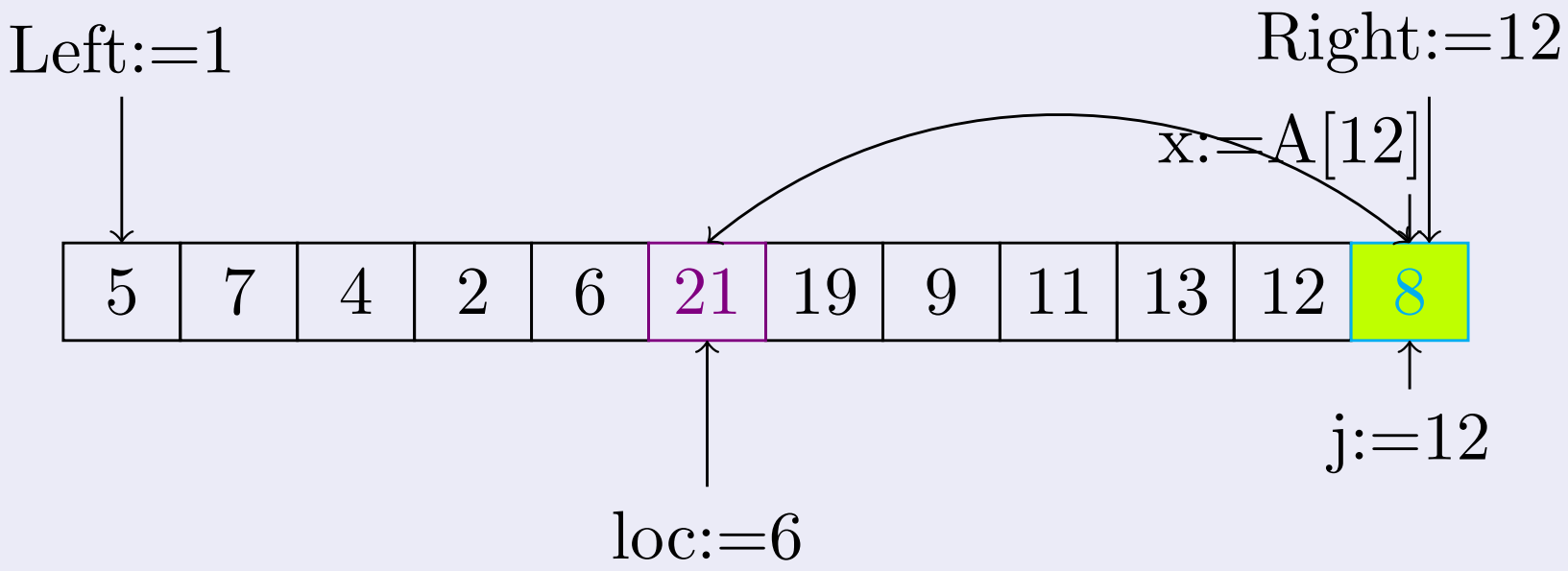
`A[8]` $\leq$ `x`.

Exemplu:



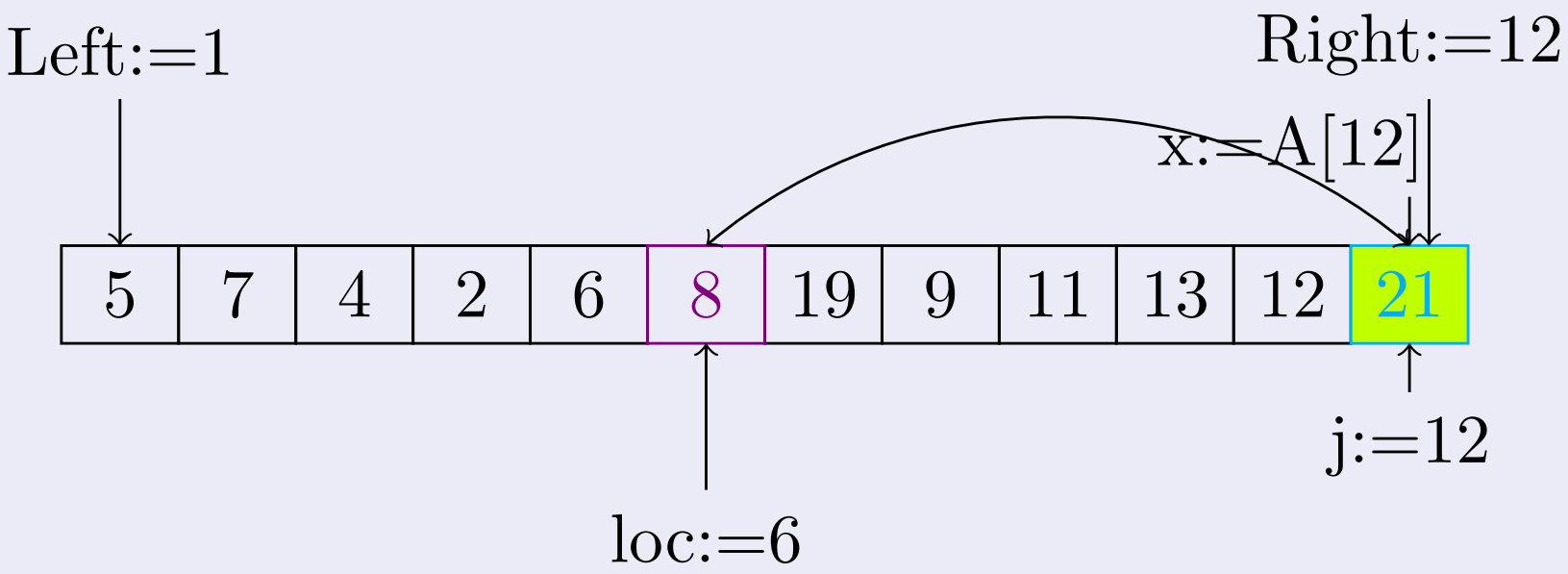
Avansăm cu `loc`.

Exemplu:



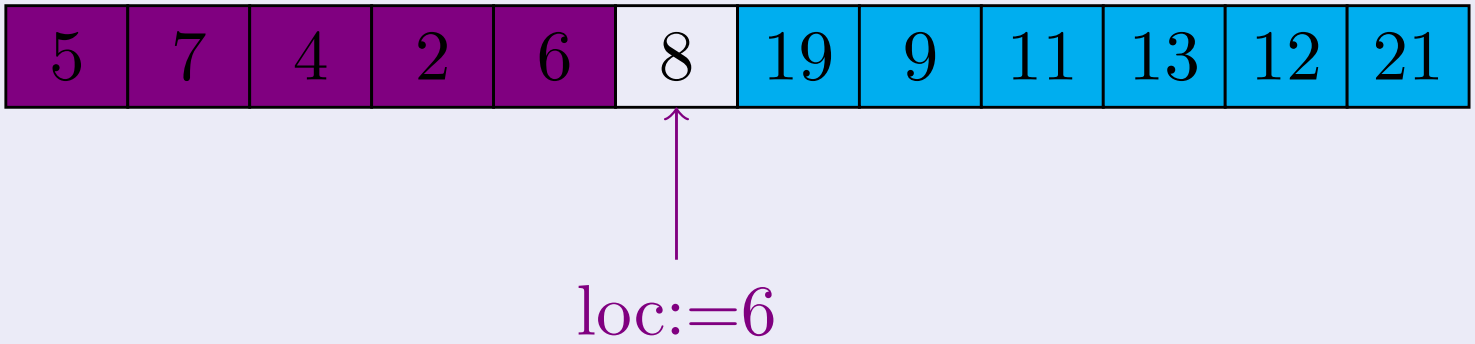
Trebuie să interschimbăm `A[6]` cu `A[12]`.

Exemplu:



Vectorul după interschimbare. Algoritmul se termină.

Exemplu:



A 6-a valoare din `A` este pe poziția `loc = 6`.Am obținut partiția: `A[1..5]`, `A[7..12]`.