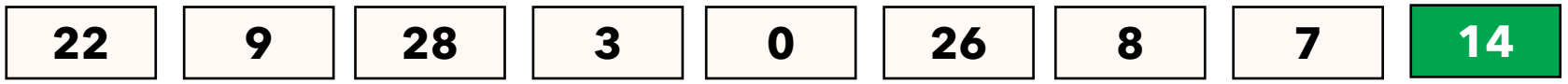
Quicksort

(Tony Hoare, 1959, https://en.wikipedia.org/wiki/Tony_Hoare)

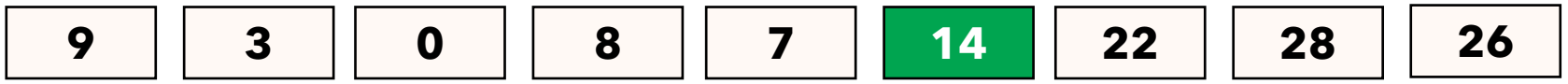
Liceo G.B. Brocchi
Classi seconde Scientifico – opzione scienze applicate
Bassano del Grappa, Maggio 2023
Prof. Giovanni Mazzocchin

Il partizionamento di un array

- *Partizionare* un array secondo un elemento *pivot* («fulcro») significa porre tutti gli elementi <= del *pivot* alla sua sinistra, e gli elementi > del *pivot* alla sua destra
- Per implementare *Quicksort*, sceglieremo come *pivot* l'ultimo elemento dell'array (quello in verde)



- Il partizionamento dovrà modificare l'array in modo da trasformarlo nel seguente:



Il partizionamento di un array

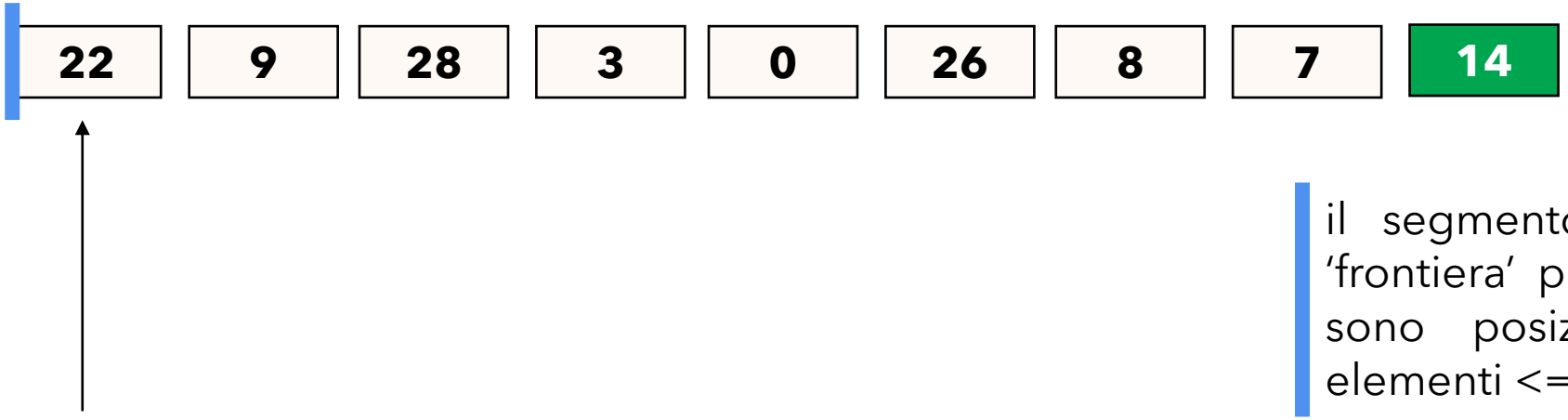
- Perché partizionare una volta sola? ***Divide and conquer!***

- Se partizioniamo ricorsivamente le partizioni, allora sicuramente ordineremo tutto l'array

- Questo è un fatto piuttosto intuitivo: infatti, il partizionamento di una fetta di array sistema il *pivot* al posto giusto, e da lì non si sposterà mai

- *Quicksort,* a differenza di *Mergesort,* è un algoritmo *in place*: non necessita di memoria aggiuntiva (quali sono gli altri algoritmi di ordinamento *in place* che conosci?)

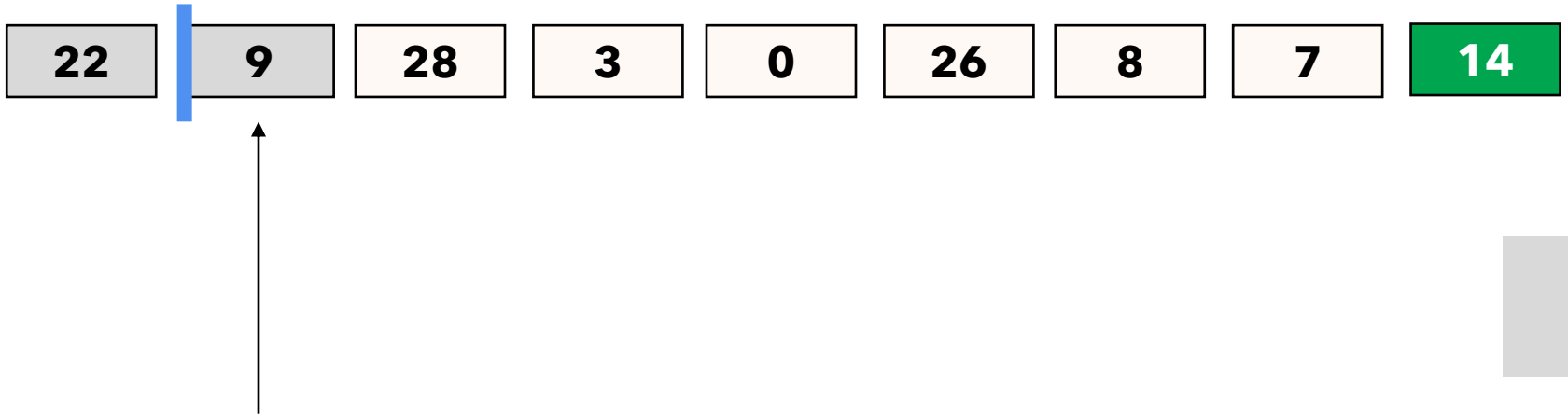
La procedura *partition*



il segmento blu indica la 'frontiera' prima della quale sono posizionati tutti gli elementi <= del *pivot*

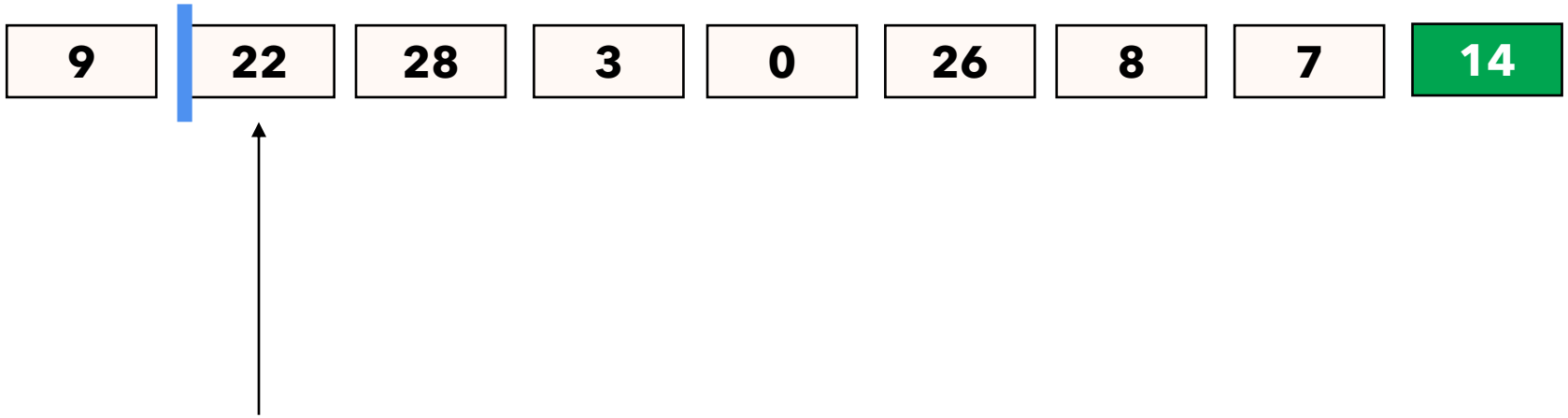
- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del *pivot*, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del *pivot* che avevamo incontrato, che è anche il primo tra gli elementi > del *pivot*, da sinistra, ossia quello appena prima della nuova frontiera

La procedura *partition*



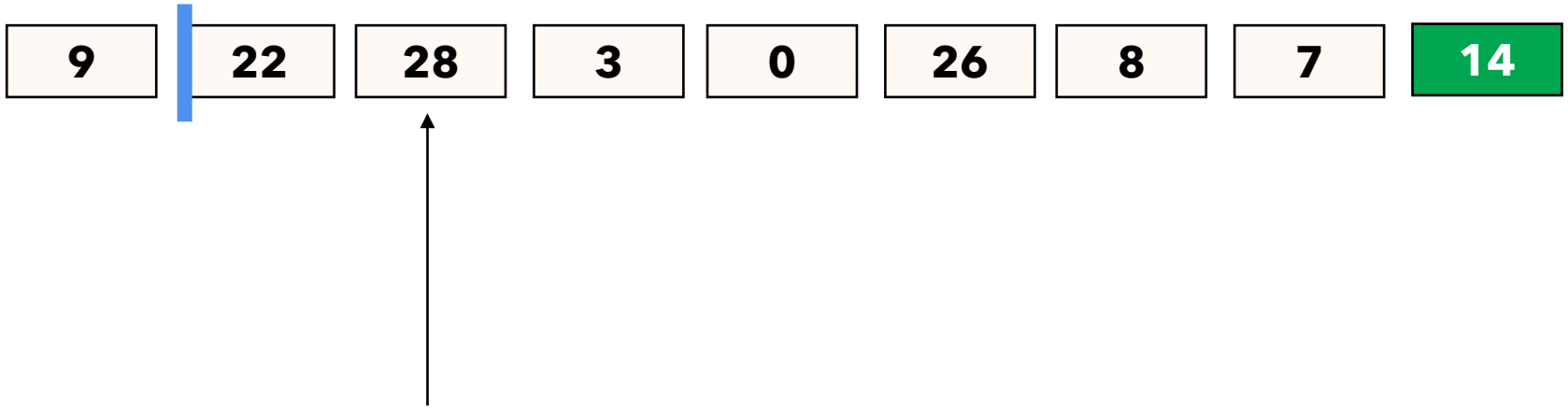
- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del *pivot*, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del *pivot* che avevamo incontrato, che è anche il primo tra gli elementi > del *pivot*, da sinistra, ossia quello appena prima della nuova frontiera

La procedura *partition*



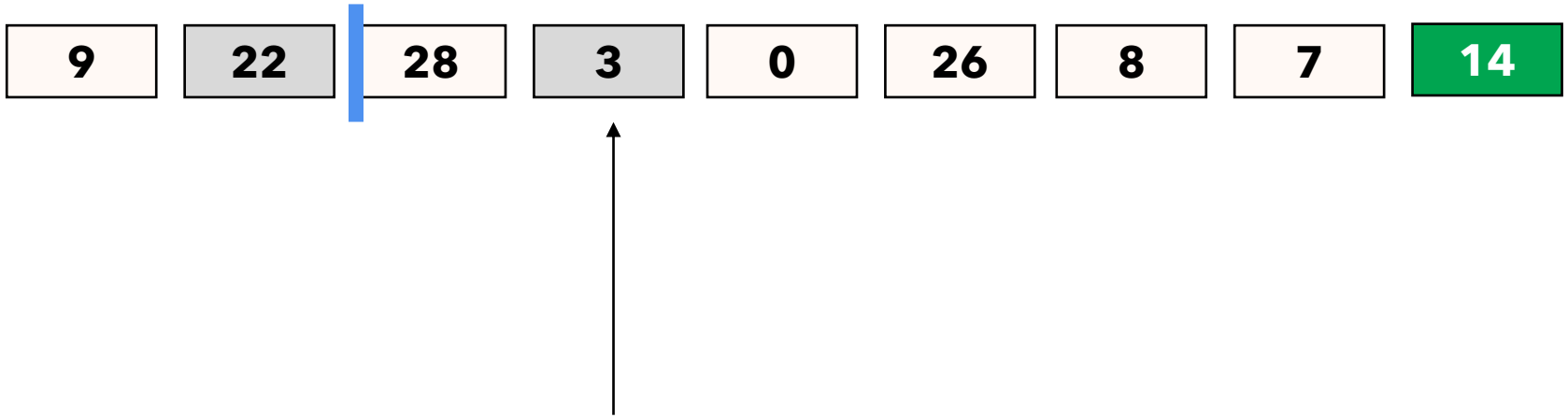
- lo scambio è avvenuto
- procediamo con la lettura degli elementi successivi

La procedura *partition*



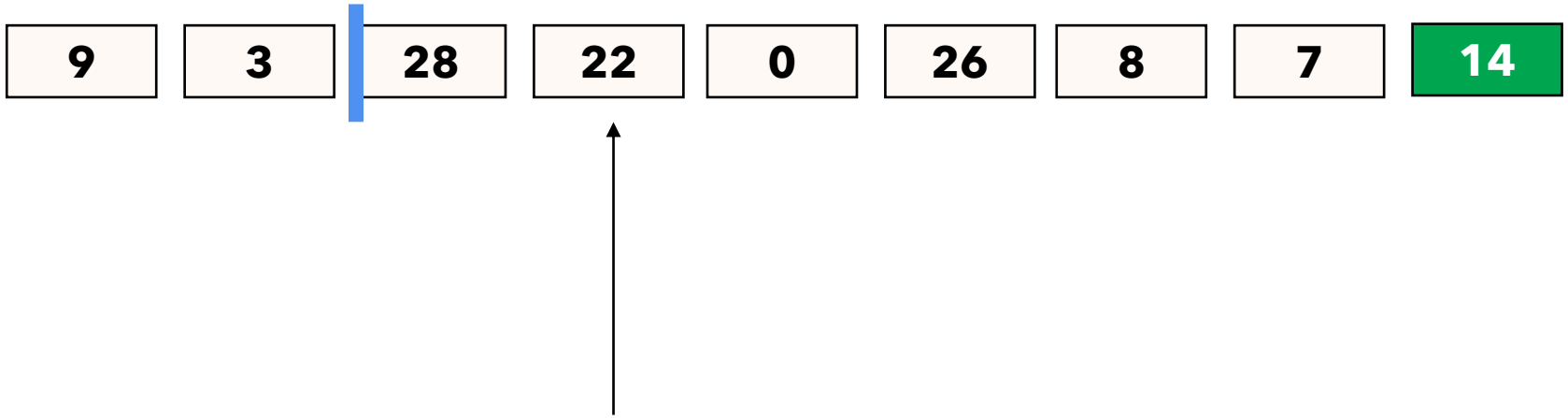
- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del *pivot*, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del *pivot* che avevamo incontrato, che è anche il primo tra gli elementi > del *pivot*, da sinistra, ossia quello appena prima della nuova frontiera

La procedura *partition*



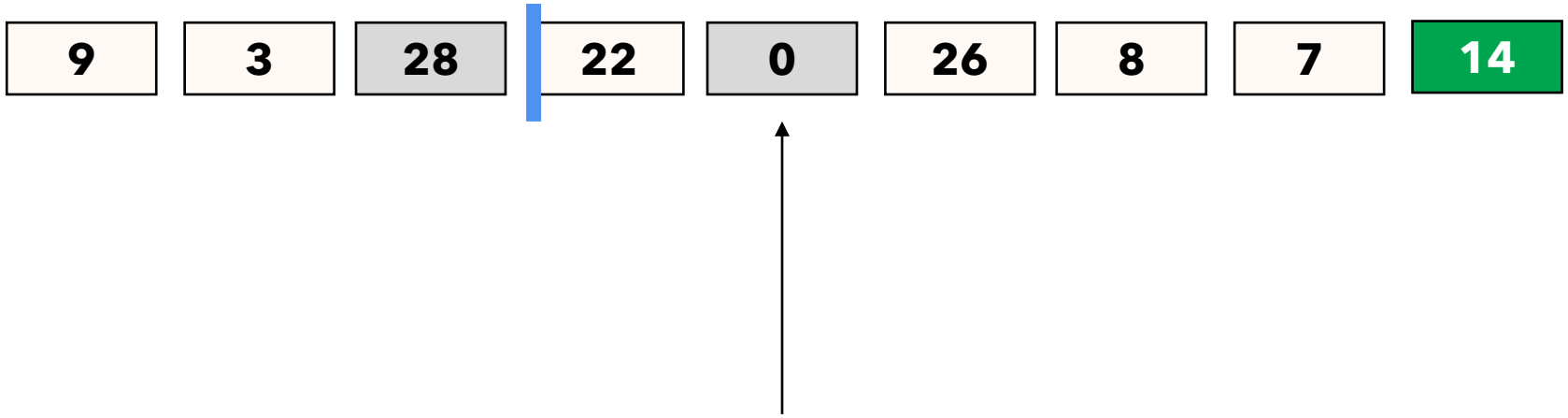
- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del *pivot*, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del *pivot* che avevamo incontrato, che è anche il primo tra gli elementi > del *pivot*, da sinistra, ossia quello appena prima della nuova frontiera

La procedura *partition*



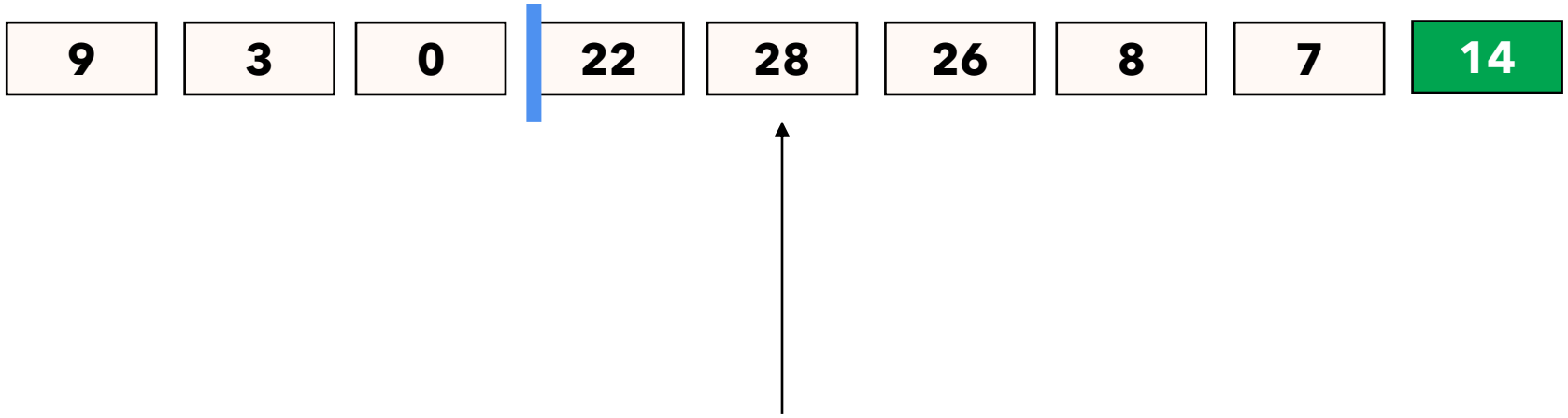
- lo scambio è avvenuto
- procediamo con la lettura degli elementi successivi

La procedura *partition*



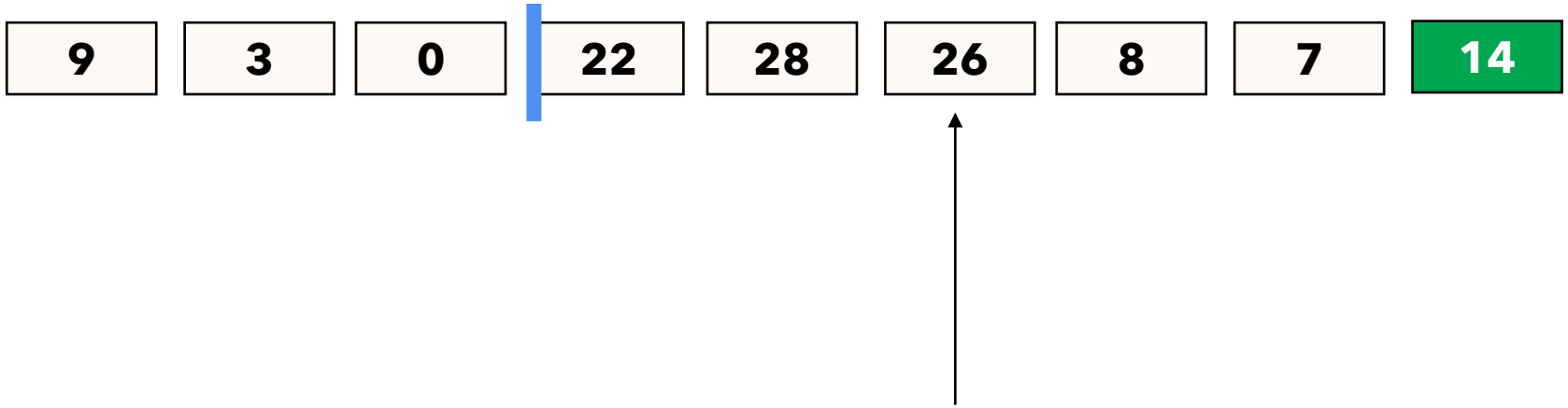
- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del *pivot*, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del *pivot* che avevamo incontrato, che è anche il primo tra gli elementi > del *pivot*, da sinistra, ossia quello appena prima della nuova frontiera

La procedura *partition*



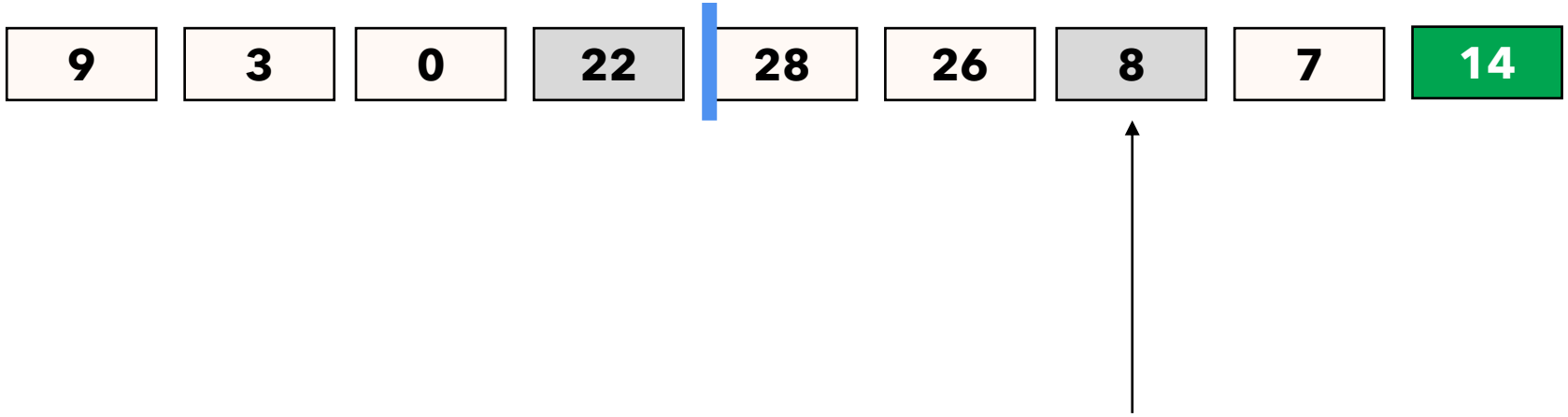
- lo scambio è avvenuto
- procediamo con la lettura degli elementi successivi

La procedura *partition*



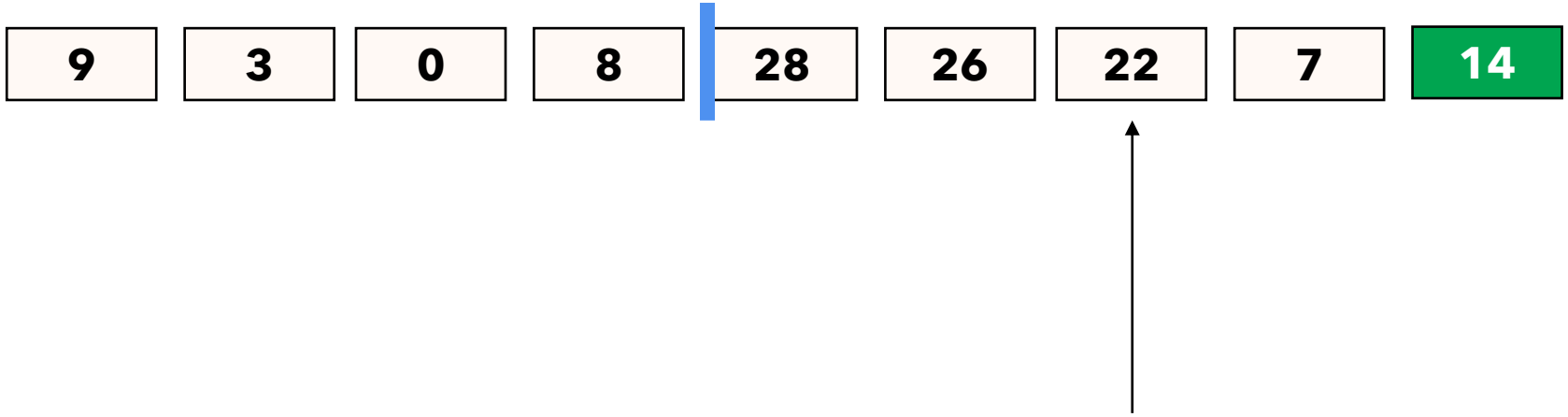
- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del *pivot*, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del *pivot* che avevamo incontrato, che è anche il primo tra gli elementi > del *pivot*, da sinistra, ossia quello appena prima della nuova frontiera

La procedura *partition*



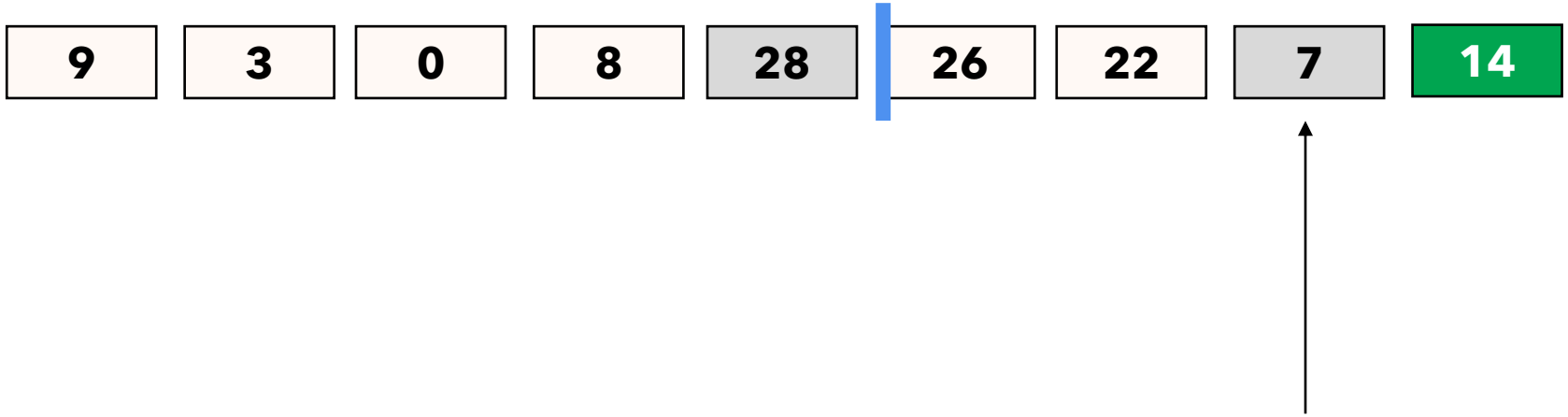
- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del *pivot*, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del *pivot* che avevamo incontrato, che è anche il primo tra gli elementi > del *pivot*, da sinistra, ossia quello appena prima della nuova frontiera

La procedura *partition*



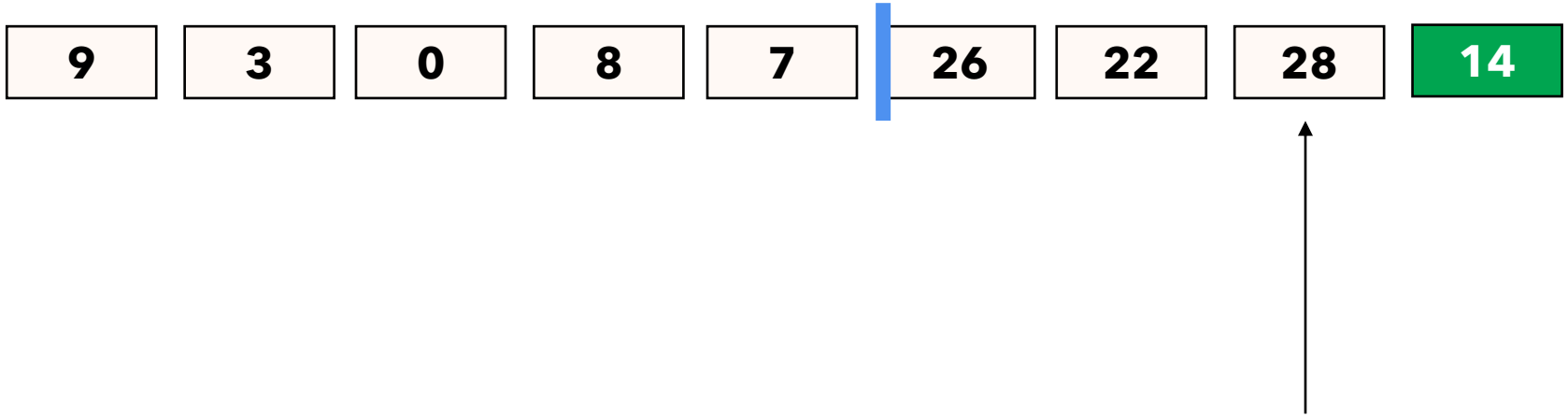
- lo scambio è avvenuto
- procediamo con la lettura degli elementi successivi

La procedura *partition*



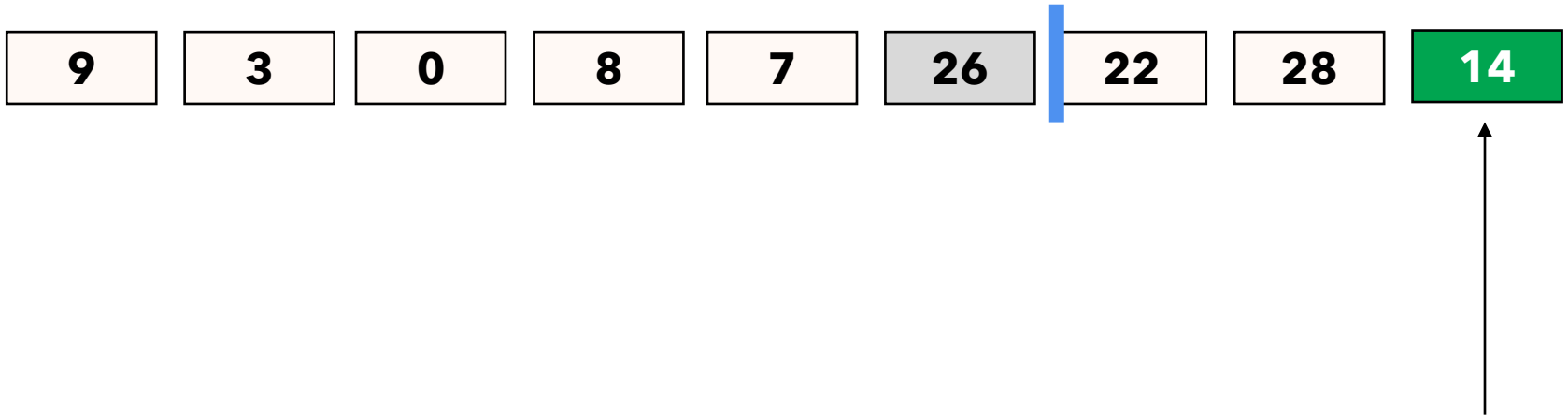
- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del *pivot*, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del *pivot* che avevamo incontrato, che è anche il primo tra gli elementi > del *pivot*, da sinistra, ossia quello appena prima della nuova frontiera

La procedura *partition*



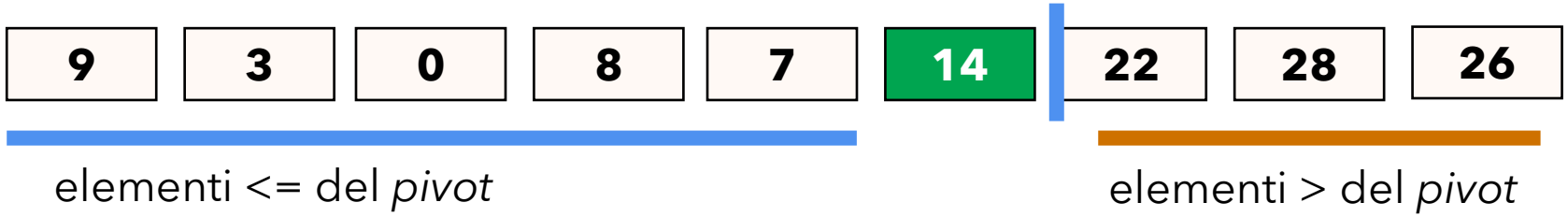
- lo scambio è avvenuto
- procediamo con la lettura degli elementi successivi

La procedura *partition*



- siamo giunti al *pivot*, che ovviamente è <= a sé stesso
- quindi, dopo aver mandato avanti la frontiera, va scambiato con il successore dell'ultimo elemento <= del *pivot* che avevamo incontrato, che è anche il primo tra gli elementi > del *pivot*, da sinistra, ossia quello appena prima della nuova frontiera

La procedura *partition*



- la partizione è completa
- sicuramente, ordinando l'array, il *pivot* 14 non si sposterà mai da dove l'abbiamo sistemato ora

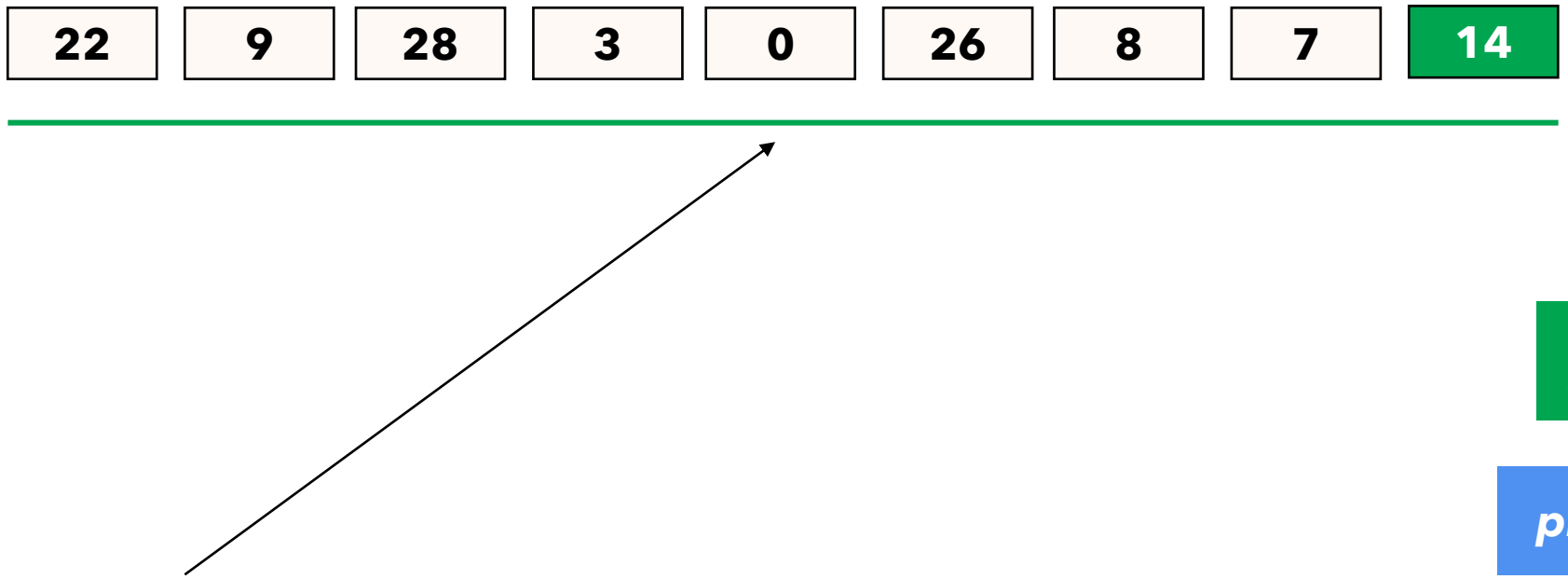
La procedura *partition* - pseudocodice

```
partition(A, low, high):
    i = low - 1  #the border
    pivot = A[high]
    for j from low to high:
        if A[j] <= pivot:
            i = i + 1
            swap(A[i], A[j])
    return i
```

Quicksort, in pseudocodice

```
quick_sort(A, low, high):
    if low >= high:
        return
    pivot_index = partition(A, low, high)
    quick_sort(A, low, pivot_index - 1)
    quick_sort(A, pivot_index + 1, high)
```

Quicksort, informalmente

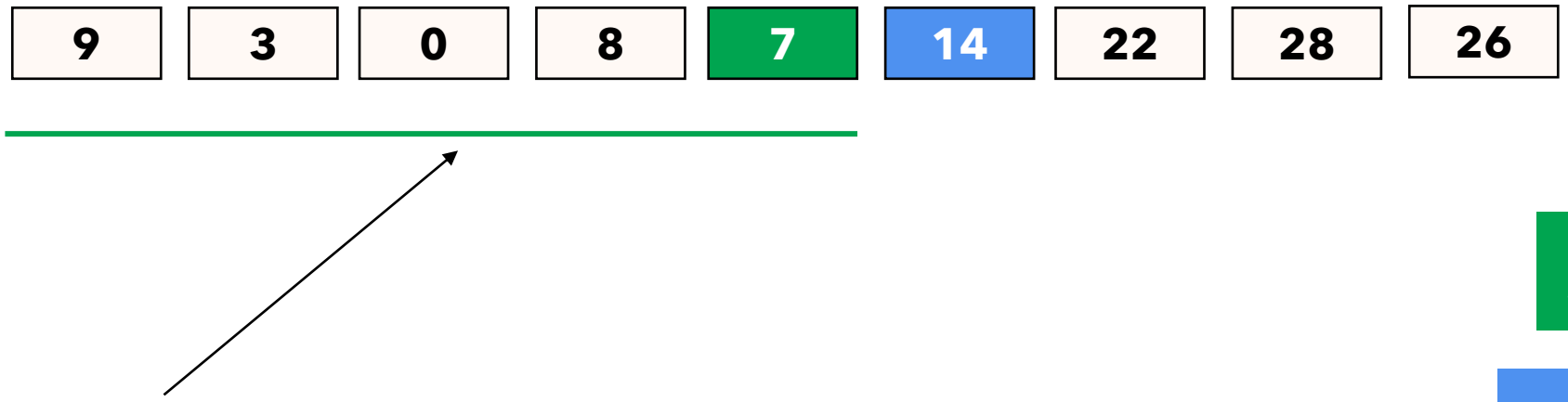


***pivot* corrente**

***pivot* precedente**

applichiamo la procedura *partition* sulla parte verde

Quicksort, informalmente

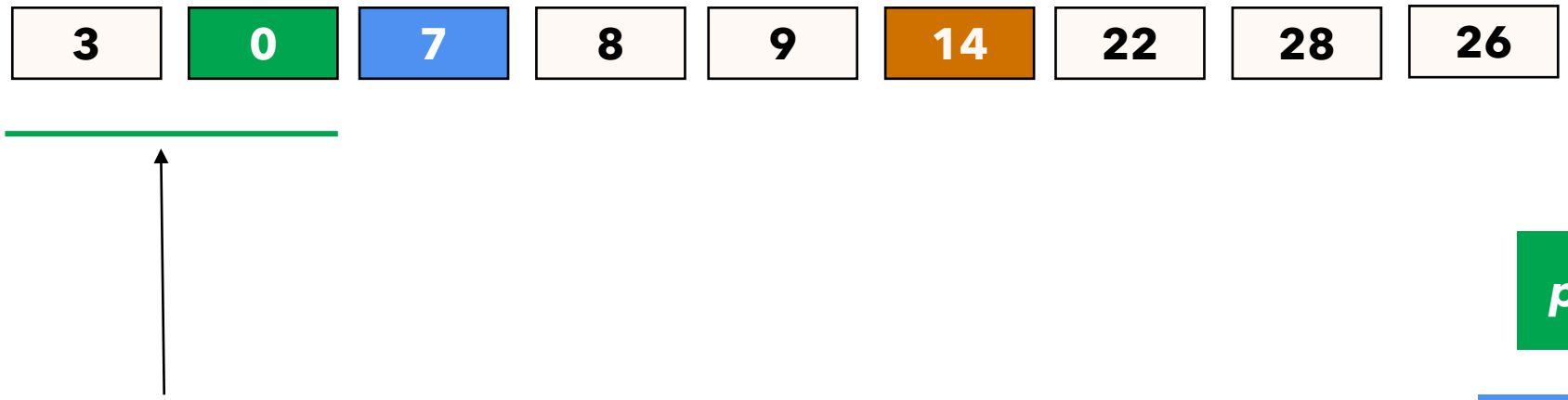


applichiamo la procedura *partition* sulla parte verde

***pivot* corrente**

***pivot* precedente**

Quicksort, informalmente



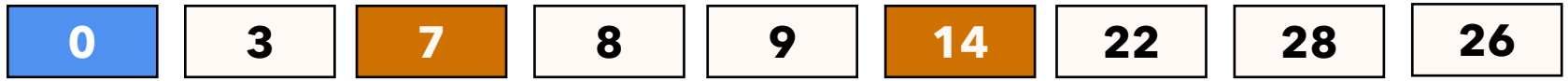
applichiamo la procedura *partition* sulla parte verde

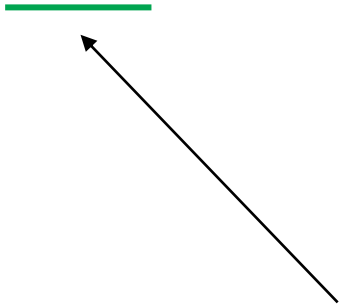
***pivot* corrente**

***pivot* precedente**

ex *pivot* sistemato al posto giusto

Quicksort, informalmente





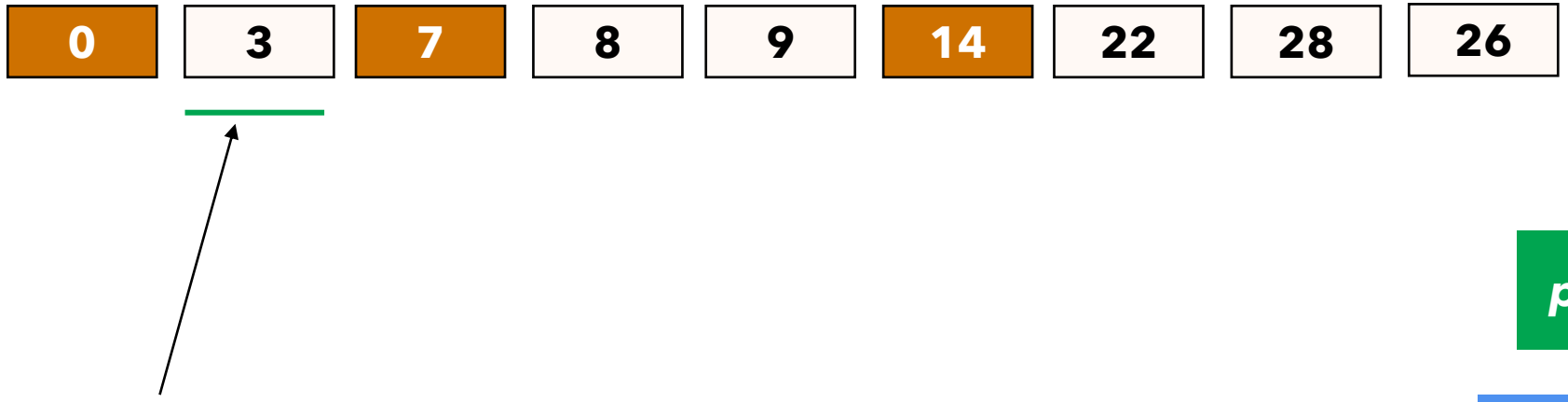
a sinistra del pivot non ci sono elementi, quindi non facciamo niente

***pivot* corrente**

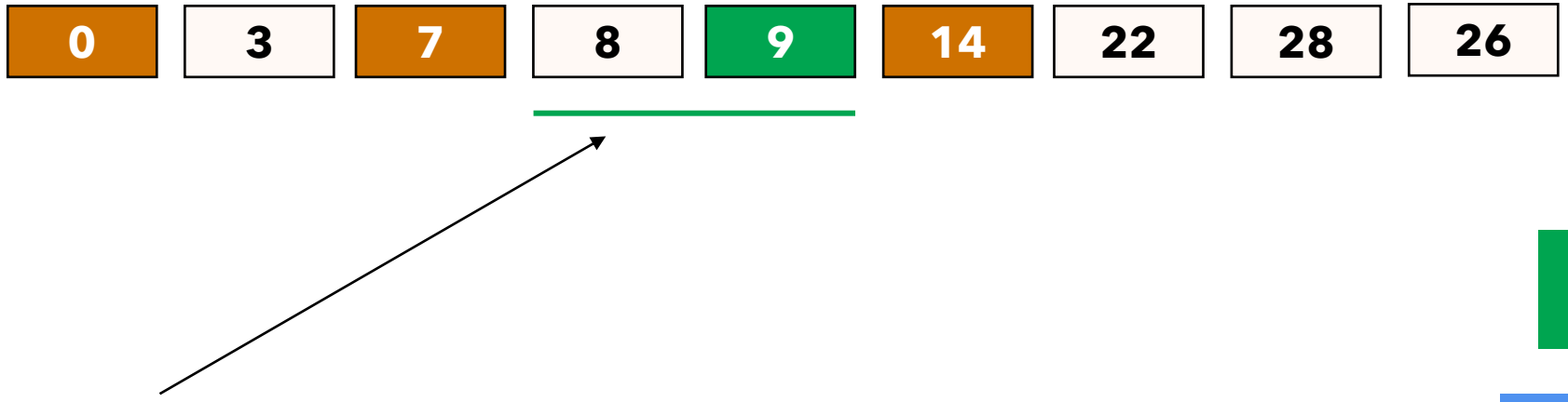
***pivot* precedente**

ex *pivot* sistemato al posto giusto

Quicksort, informalmente



Quicksort, informalmente



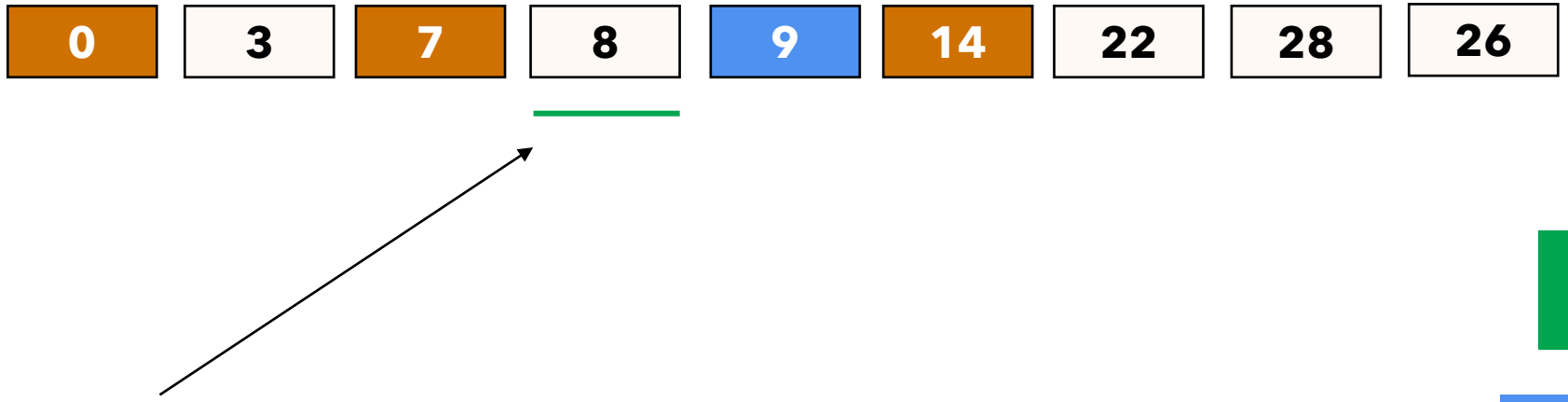
applichiamo la procedura *partition* sulla parte verde

***pivot* corrente**

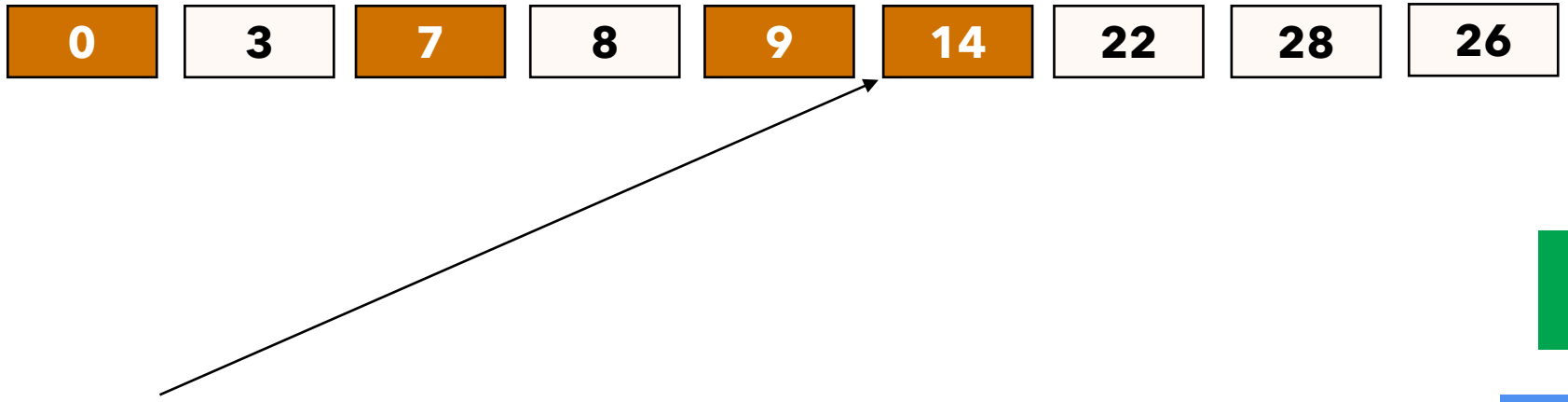
***pivot* precedente**

ex *pivot* sistemato al posto giusto

Quicksort, informalmente



Quicksort, informalmente



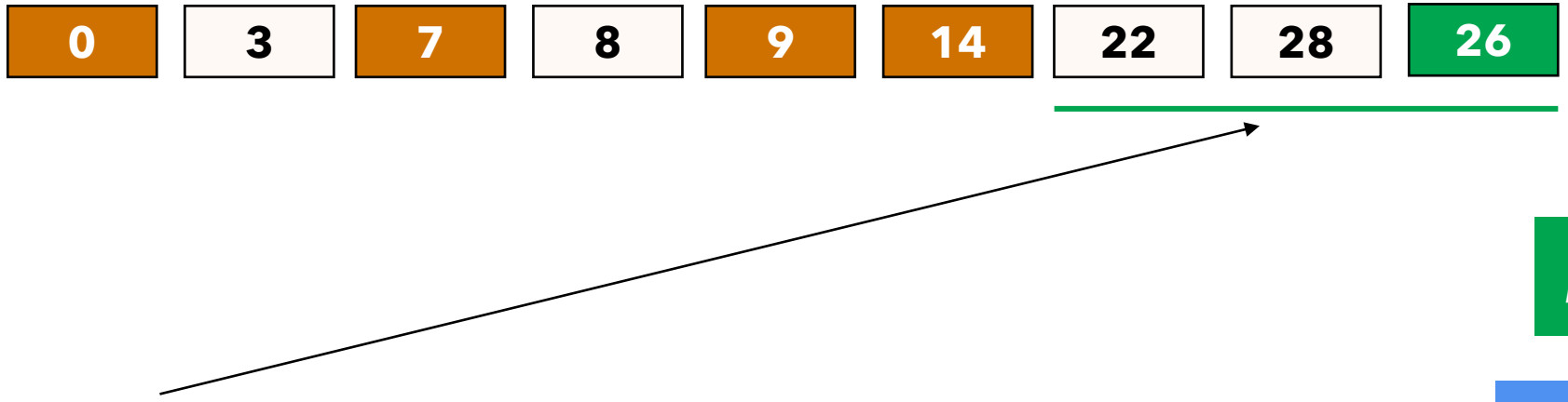
0 elementi a destra della slice corrente, niente da fare

***pivot* corrente**

***pivot* precedente**

ex *pivot* sistemato al posto giusto

Quicksort, informalmente



applichiamo la procedura *partition* sulla parte verde

***pivot* corrente**

***pivot* precedente**

ex *pivot* sistemato al posto giusto

Quicksort, informalmente

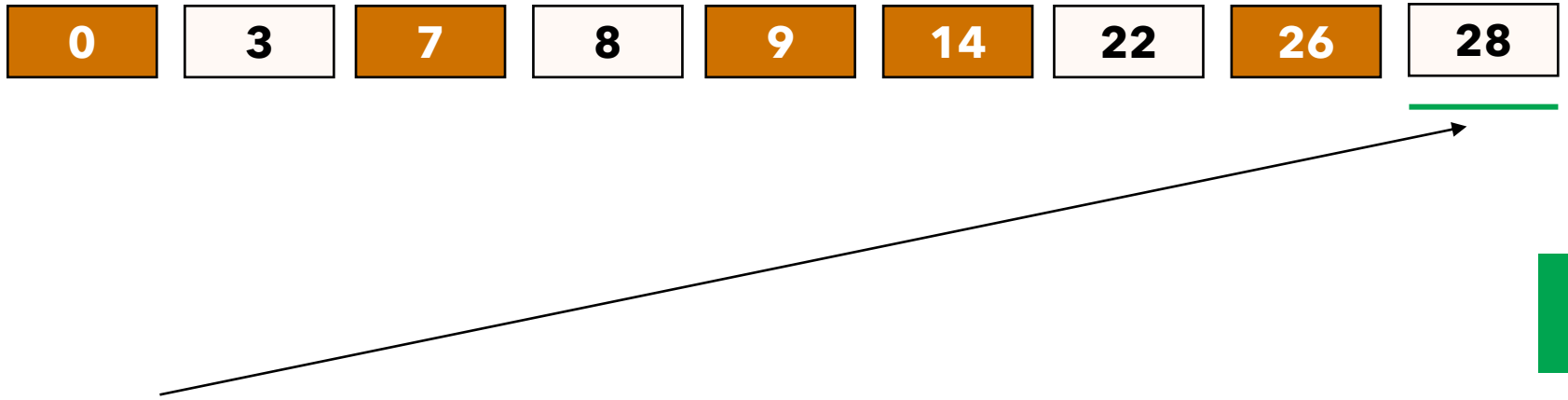
1 solo elemento, niente da fare

***pivot* corrente**

***pivot* precedente**

ex *pivot* sistemato al posto giusto

Quicksort, informalmente



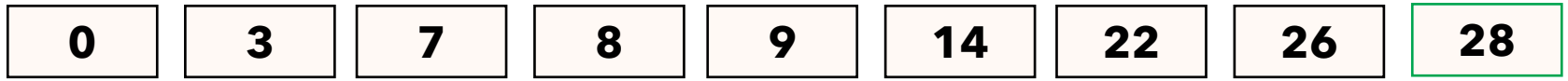
1 solo elemento, niente da fare

***pivot* corrente**

***pivot* precedente**

ex *pivot* sistemato al posto giusto

Quicksort, informalmente



l'array è ordinato

***pivot* corrente**

***pivot* precedente**

ex *pivot* sistemato al posto giusto