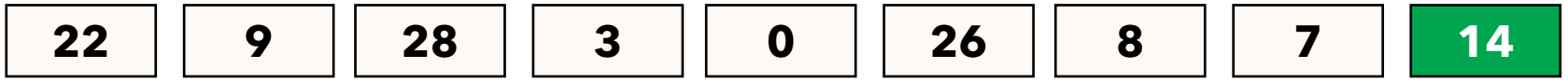
Quicksort

(Tony Hoare, 1959)

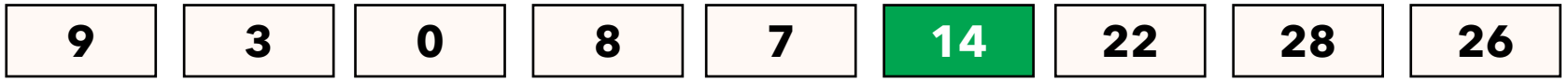
Liceo G.B. Brocchi – Bassano del Grappa (VI)
Liceo Scientifico – opzione scienze applicate
Giovanni Mazzocchin

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)



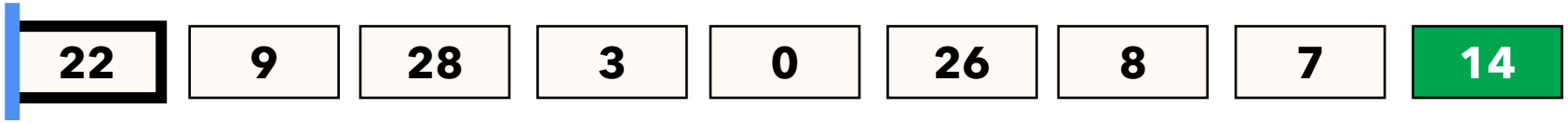
- Il partizionamento dovrà modificare l'array in modo da trasformarlo nel seguente:



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 fatto è piuttosto intuitivo: infatti, il partizionamento di una fetta di array sistema il pivot al posto giusto, e da lì non si sposterà mai. Vedremo inoltre che Quicksort è un algoritmo *in place*

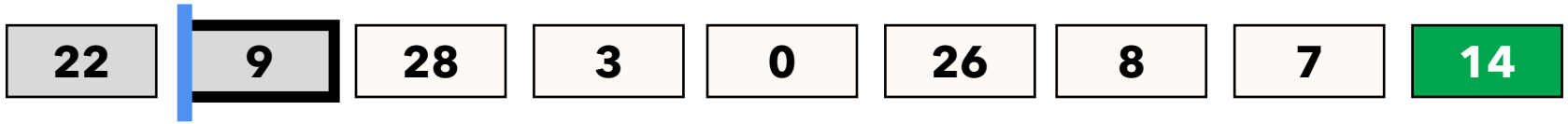
La procedura *partition*



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

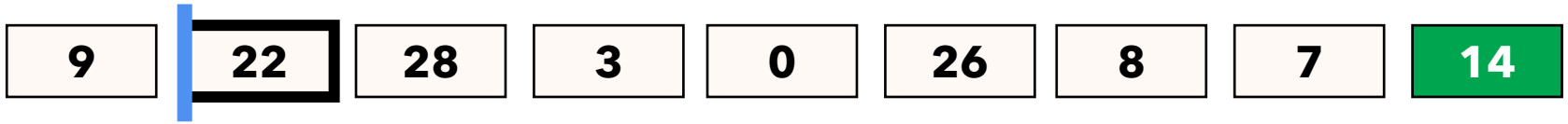
La procedura *partition*



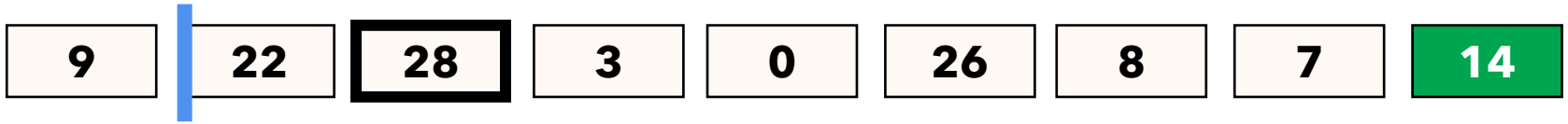
elementi da scambiare

- 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*

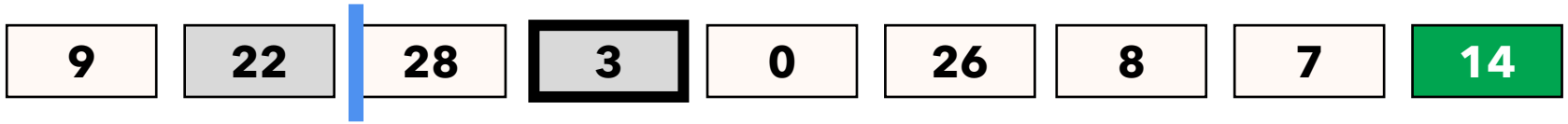


La procedura *partition*



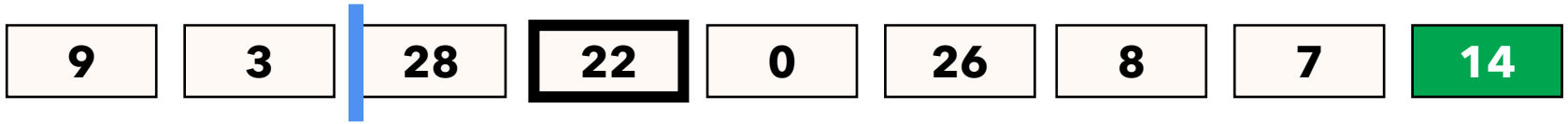
- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del pivot, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del pivot che avevamo incontrato, che è anche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

La procedura *partition*

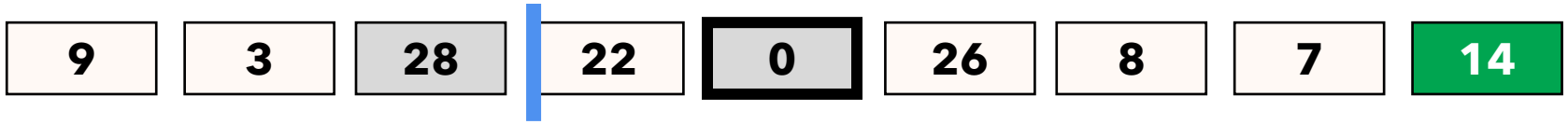


- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del pivot, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del pivot che avevamo incontrato, che è anche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

La procedura *partition*



La procedura *partition*

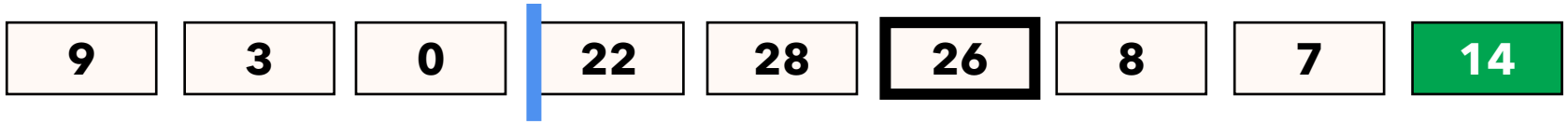


- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del pivot, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del pivot che avevamo incontrato, che è anche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

La procedura *partition*

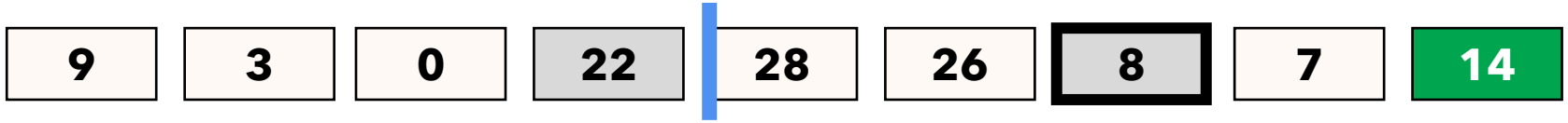
- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del pivot, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del pivot che avevamo incontrato, che è anche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

La procedura *partition*



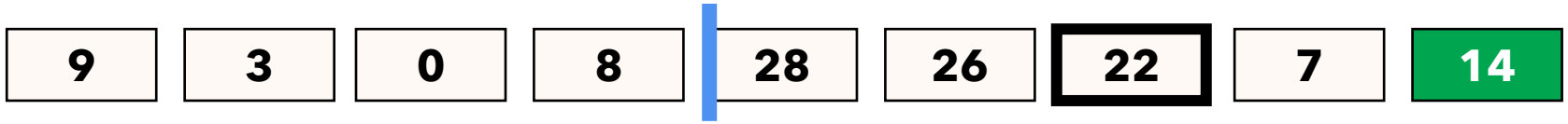
- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del pivot, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del pivot che avevamo incontrato, che è anche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

La procedura *partition*

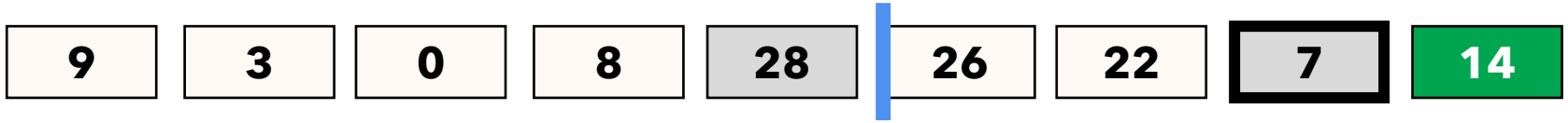


- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del pivot, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del pivot che avevamo incontrato, che è anche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

La procedura *partition*

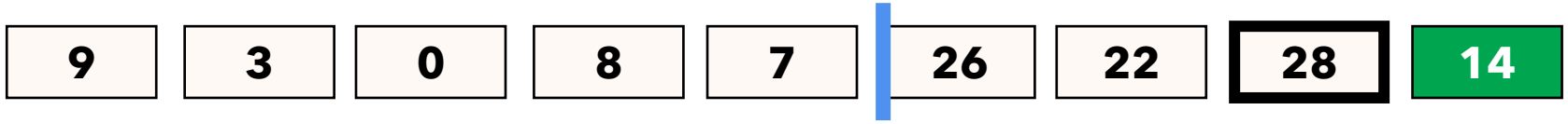


La procedura *partition*

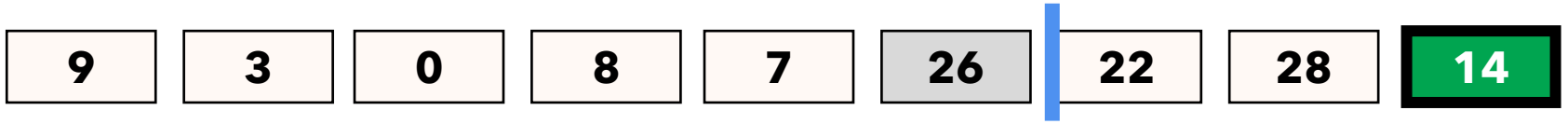


- leggiamo l'array elemento per elemento da sinistra a destra
- se l'elemento corrente è > del pivot, non facciamo niente
- altrimenti, mandiamo avanti la frontiera di una posizione e scambiamo l'elemento corrente con il successore dell'ultimo elemento <= del pivot che avevamo incontrato, che è anche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

La procedura *partition*

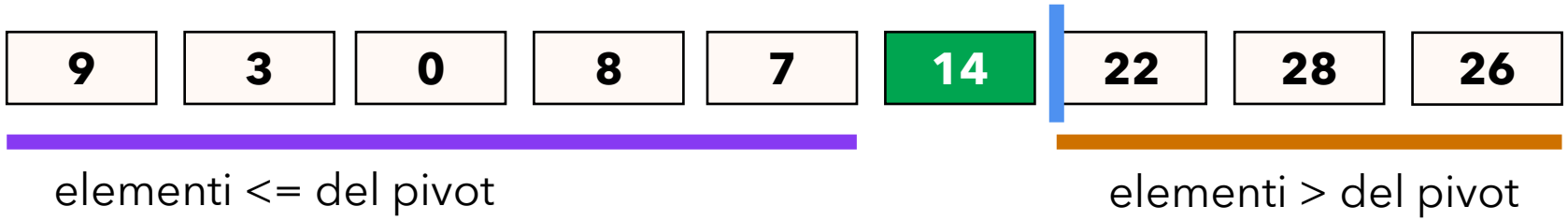


La procedura *partition*



- 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 più da dove l'abbiamo sistemato ora

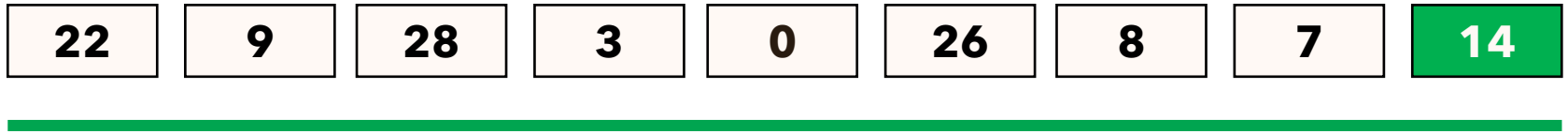
La procedura *partition*

```
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

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

Quicksort



pivot corrente

applichiamo la procedura *partition* sulla parte verde

ex pivot sistemato al posto giusto

Quicksort

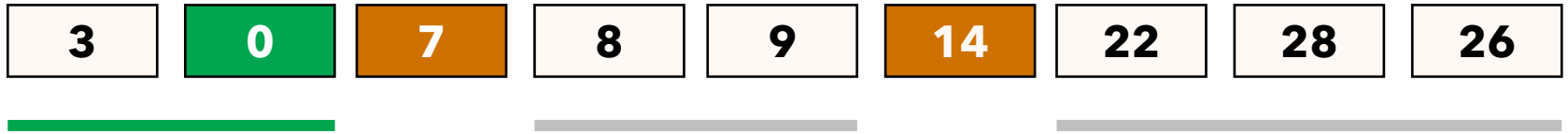
pivot corrente

applichiamo la procedura *partition* sulla parte verde

le parti grigie sono in sospeso

ex pivot sistemato al posto giusto

Quicksort



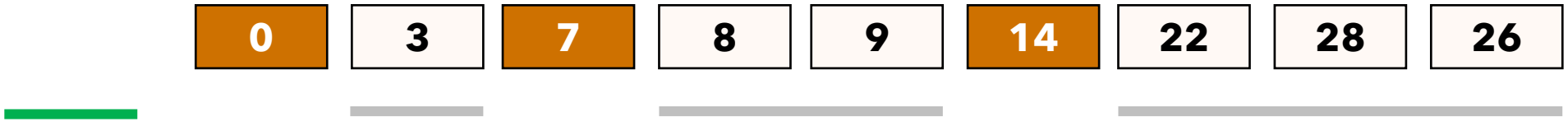
pivot corrente

applichiamo la procedura *partition* sulla parte verde

le parti grigie sono in sospeso

ex pivot sistemato al posto giusto

Quicksort

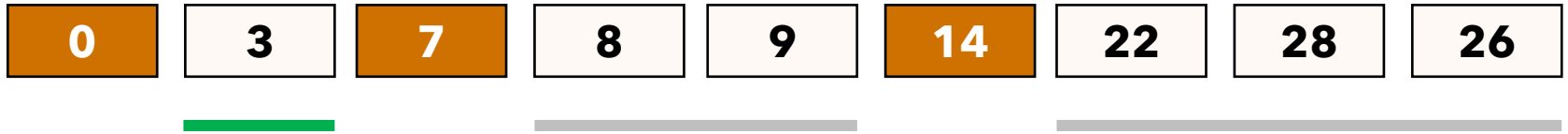


pivot corrente

0 elementi, niente da fare

ex pivot sistemato al posto giusto

Quicksort



pivot corrente

1 elemento, niente da fare

ex pivot sistemato al posto giusto

Quicksort

pivot corrente

applichiamo la procedura *partition* sulla parte verde

le parti grigie sono in sospeso

ex pivot sistemato al posto giusto

Quicksort

pivot corrente

1 elemento, niente da fare

ex pivot sistemato al posto giusto

Quicksort

pivot corrente

applichiamo la procedura *partition* sulla parte verde

ex pivot sistemato al posto giusto

Quicksort

pivot corrente

1 elemento, niente da fare

le parti grigie sono in sospeso

ex pivot sistemato al posto giusto

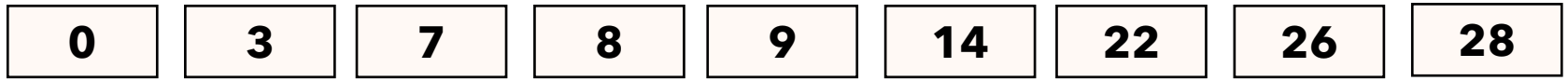
Quicksort

pivot corrente

1 elemento, niente da fare

ex pivot sistemato al posto giusto

Quicksort



pivot corrente

l'array è ordinato

ex pivot sistemato al posto giusto