Agenda di oggi

- *Bubblesort*
- *Quicksort*
- Counting sort

ALGORITMO

- IPOTESI: Ordinare un array di **numeri interi non negativi**, entro un intervallo limitato (es. da 0 a k).
- Non è un algoritmo di confronto.

Passi:

1. Conta le occorrenze (**conteggio**)
2. Per ogni valore nell'array di input, aumenta il contatore corrispondente in un array di conteggio B.
3. Calcola le posizioni cumulative
4. Trasforma l'array B nell'array B' in modo che ogni posizione i dica quanti elementi sono ≤ i. Ogni cella deve contenere il valore attuale + il valore della cella precedente (in B').
5. Costruisce l'array ordinato finale (**ordinamento**)
6. Scorre l'array A da destra a sinistra (per **stabilità**).
7. Usa i valori di B per mettere ogni elemento al posto giusto nel nuovo array ordinato.
8. Dopo aver posizionato un elemento, decrementa il suo contatore in B.

ESEMPIO

Array da ordinare

CONTEGGIO DEI VALORI

array A

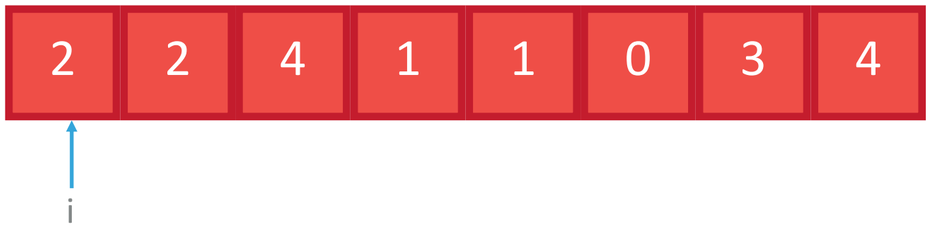
Supponiamo tutti i valori siano in [0,4]

array B

Allochiamo un array di k+1 elementi

CONTEGGIO DEI VALORI

array A

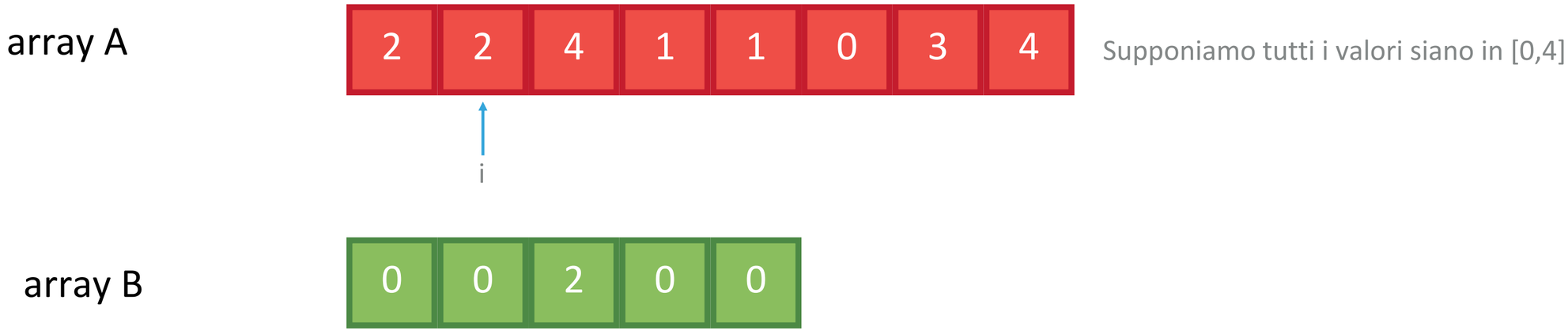


Supponiamo tutti i valori siano in [0,4]

array B

Ogni posizione qui corrisponde ad un valore presente in A da 0 a 4 (ovvero 5 possibili valori)

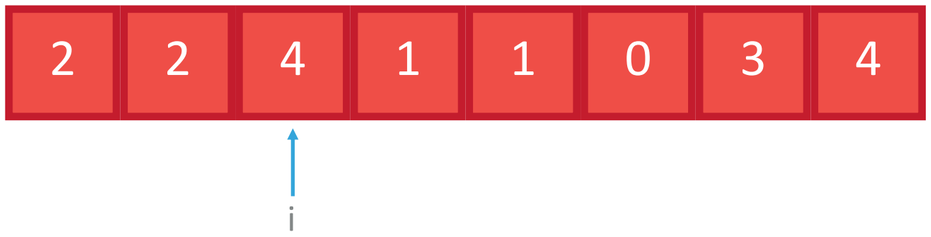
CONTEGGIO DEI VALORI



Supponiamo tutti i valori siano in [0,4]

CONTEGGIO DEI VALORI

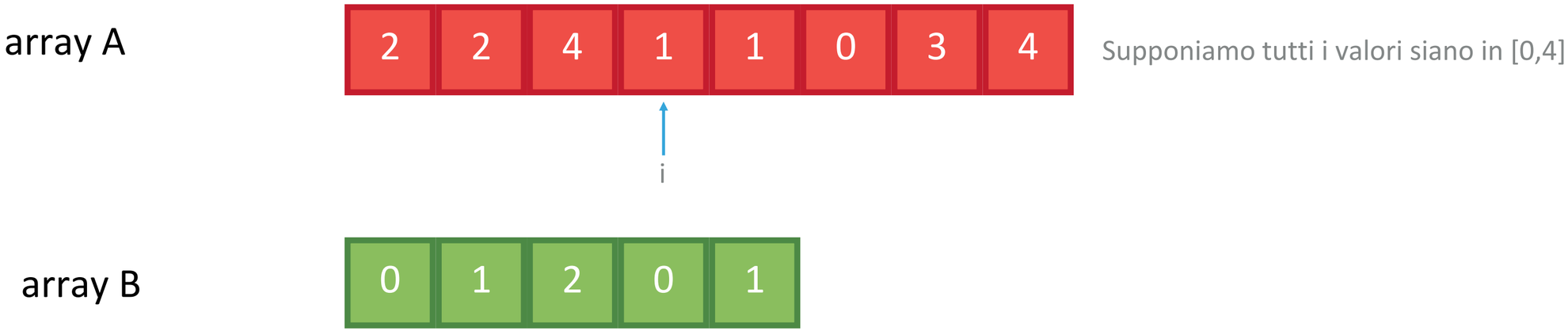
array A



Supponiamo tutti i valori siano in [0,4]

array B

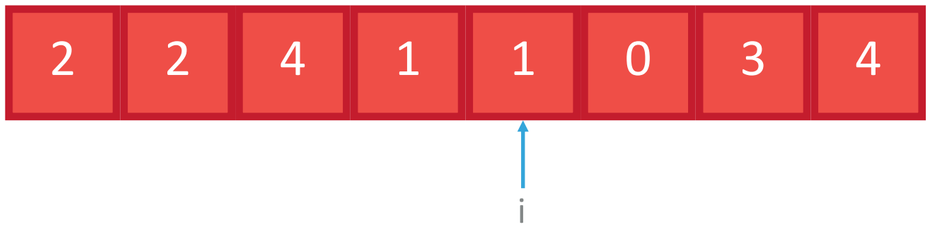
CONTEGGIO DEI VALORI



Supponiamo tutti i valori siano in [0,4]

CONTEGGIO DEI VALORI

array A

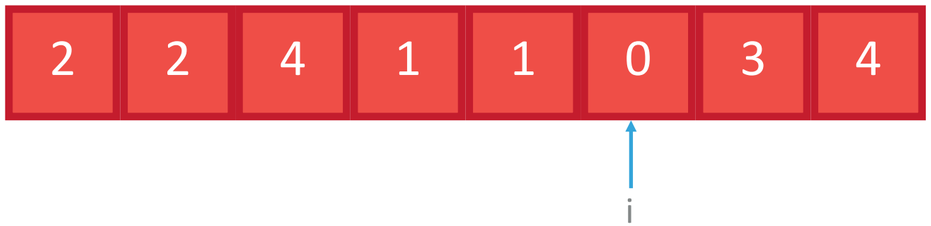


Supponiamo tutti i valori siano in [0,4]

array B

CONTEGGIO DEI VALORI

array A

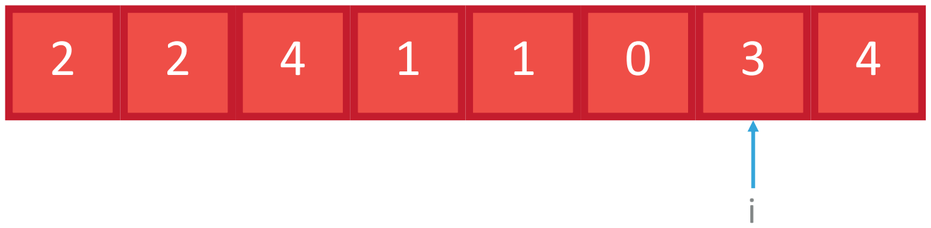


Supponiamo tutti i valori siano in [0,4]

array B

CONTEGGIO DEI VALORI

array A

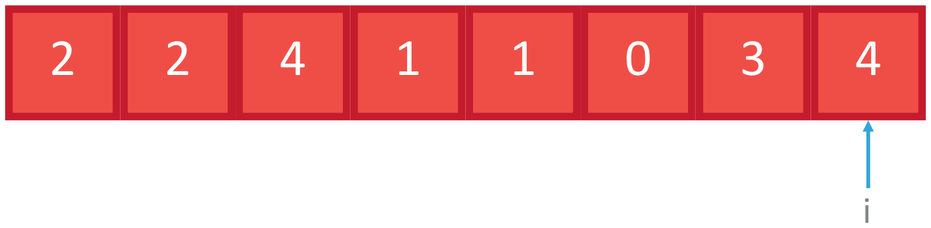


Supponiamo tutti i valori siano in [0,4]

array B

CONTEGGIO DEI VALORI

array A



Supponiamo tutti i valori siano in [0,4]

array B

Ora abbiamo a disposizione per ogni valore in [0,k] il numero di occorrenze

Vogliamo contare per ogni valore il numero di elementi **minori o uguali**
a quel valore che sono presenti nell'array di partenza

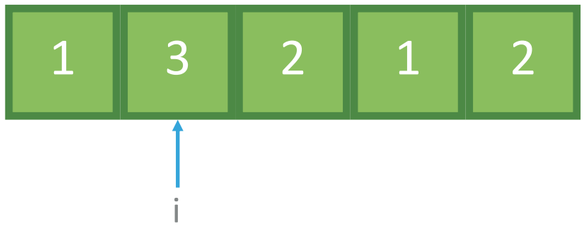
CORRETTA RISPETTO ALLA LEZIONE

CONTEGGIO DEI VALORI

array A

Supponiamo tutti i valori siano in [0,4]

array B’



B'[i] = B[i] + B[i-1]

array B

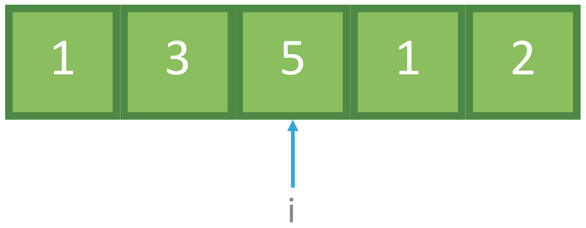
CORRETTA RISPETTO ALLA LEZIONE

CONTEGGIO DEI VALORI

array A

Supponiamo tutti i valori siano in [0,4]

array B’



B'[i] = B[i] + B[i-1]

array B

CORRETTA RISPETTO ALLA LEZIONE

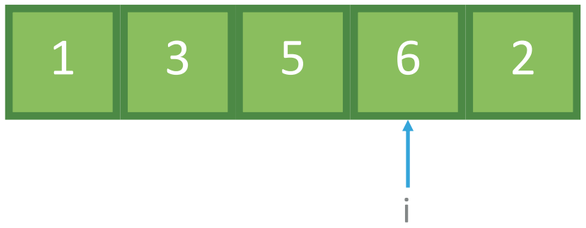
CONTEGGIO DEI VALORI

array A

Supponiamo tutti i valori siano in [0,4]

array B

array B’



B'[i] = B[i] + B[i-1]

CORRETTA RISPETTO ALLA LEZIONE

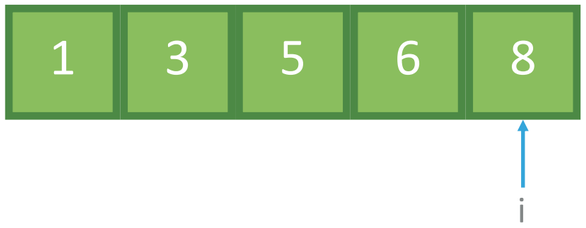
CONTEGGIO DEI VALORI

array A

Supponiamo tutti i valori siano in [0,4]

array B

array B’



B'[i] = B[i] + B[i-1]

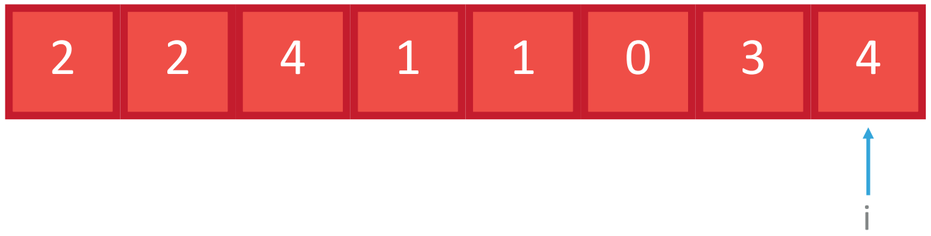
Ora abbiamo in B[i] l’ultima posizione in cui il valore “i” deve apparire

CORRETTA RISPETTO ALLA LEZIONE

ORDINAMENTO

Si parte dal fondo di A

array A



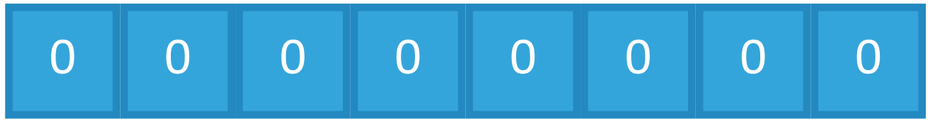
Supponiamo tutti i valori siano in [0,4]

Scorriamo dal fondo

array B’

B'[A[i]] = B'[A[i]] - 1

array C



QUESTA OPERAZIONE RICHIEDE UNA SPIEGAZIONE

C[B'[A[i]]-1] = A[i]

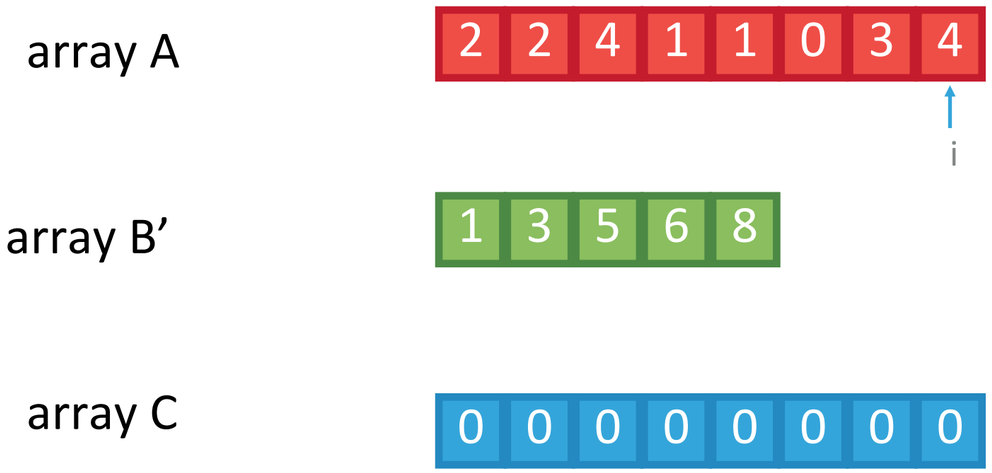
CORRETTA RISPETTO ALLA LEZIONE

ORDINAMENTO

Elemento da ordinare

$$C[B'[A[i]]-1] = A[i]$$

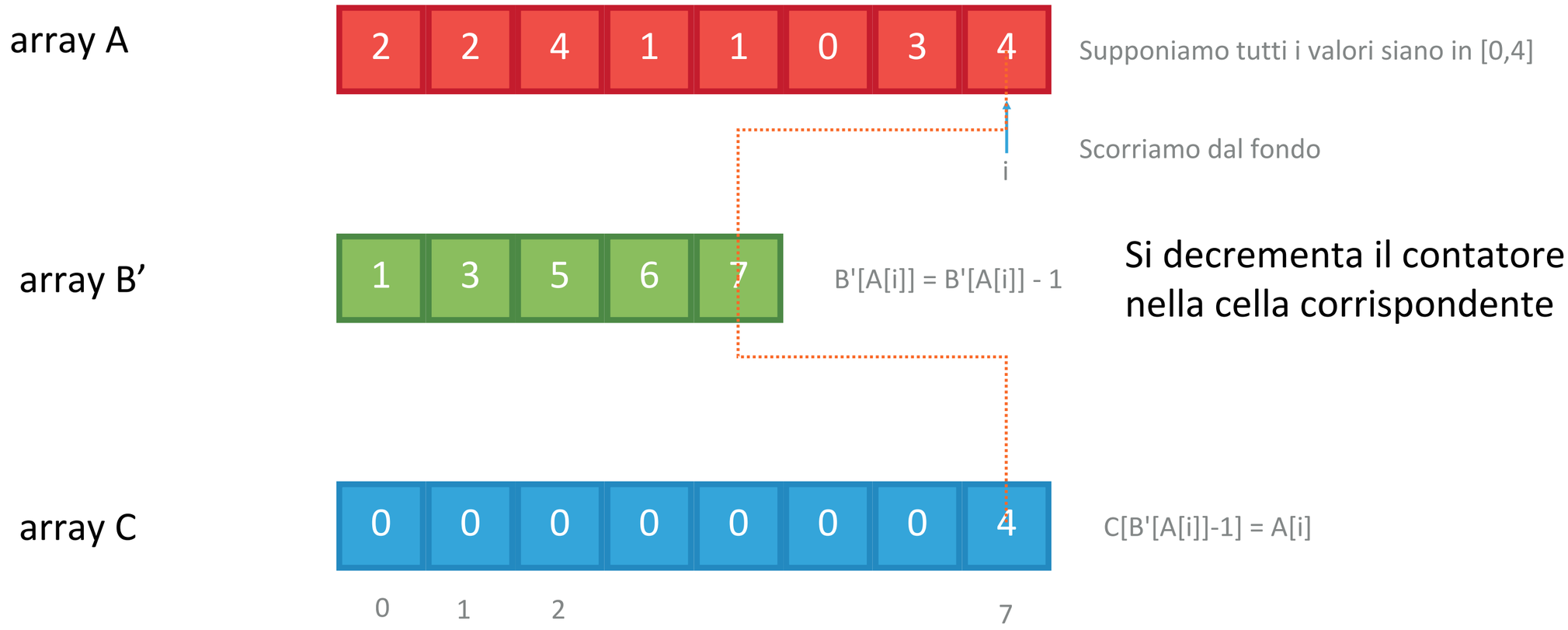
Posizione in cui mettere l'elemento
(indicizzata dall'elemento stesso)



CORRETTA RISPETTO ALLA LEZIONE

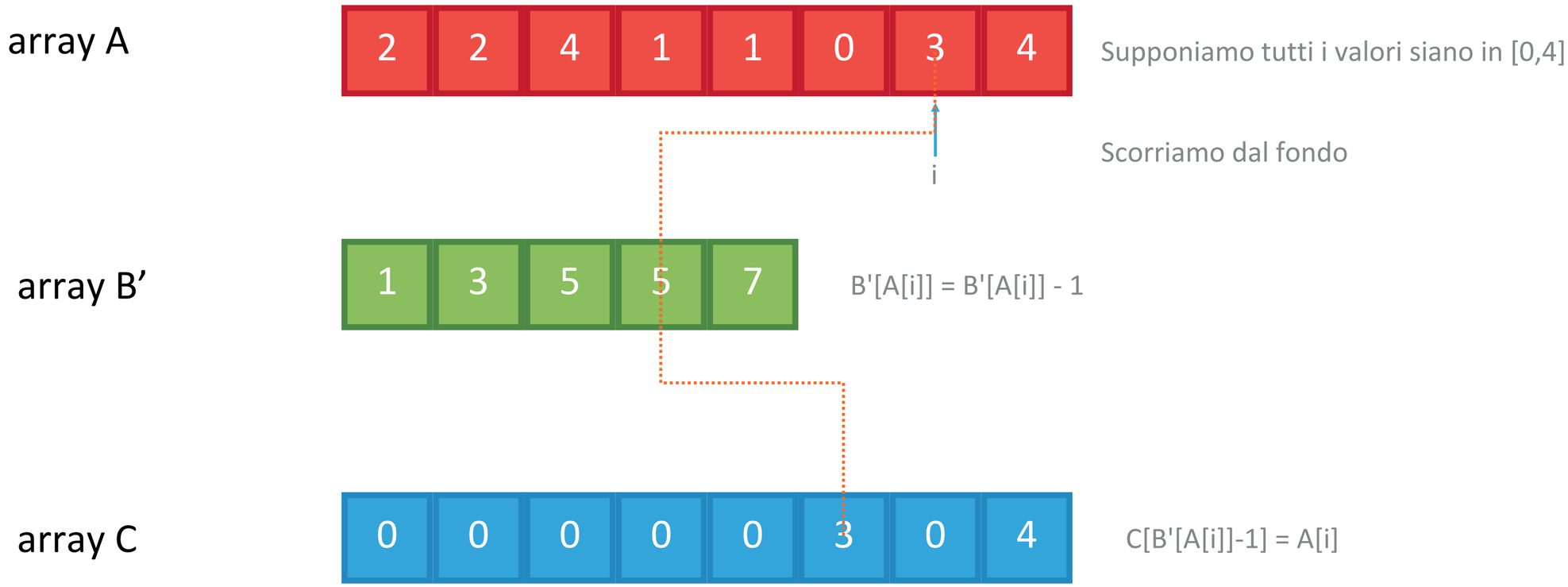
ORDINAMENTO

Si parte dal fondo di A



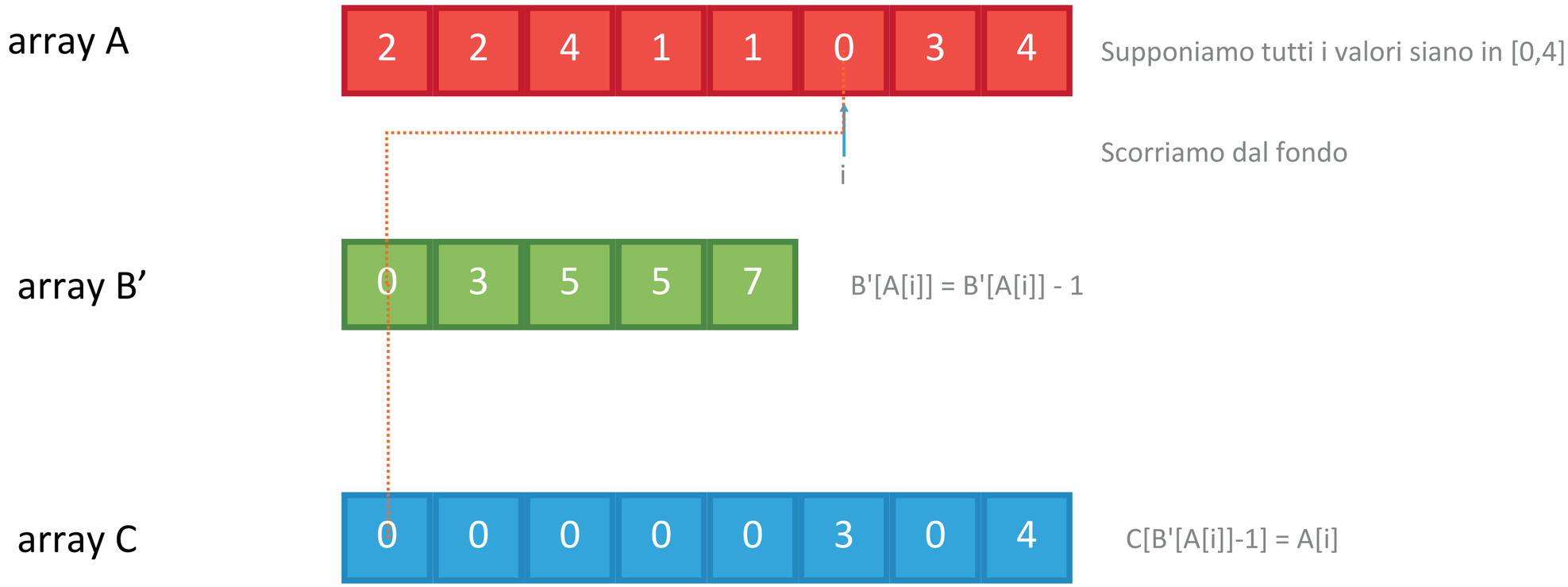
CORRETTA RISPETTO ALLA LEZIONE

ORDINAMENTO



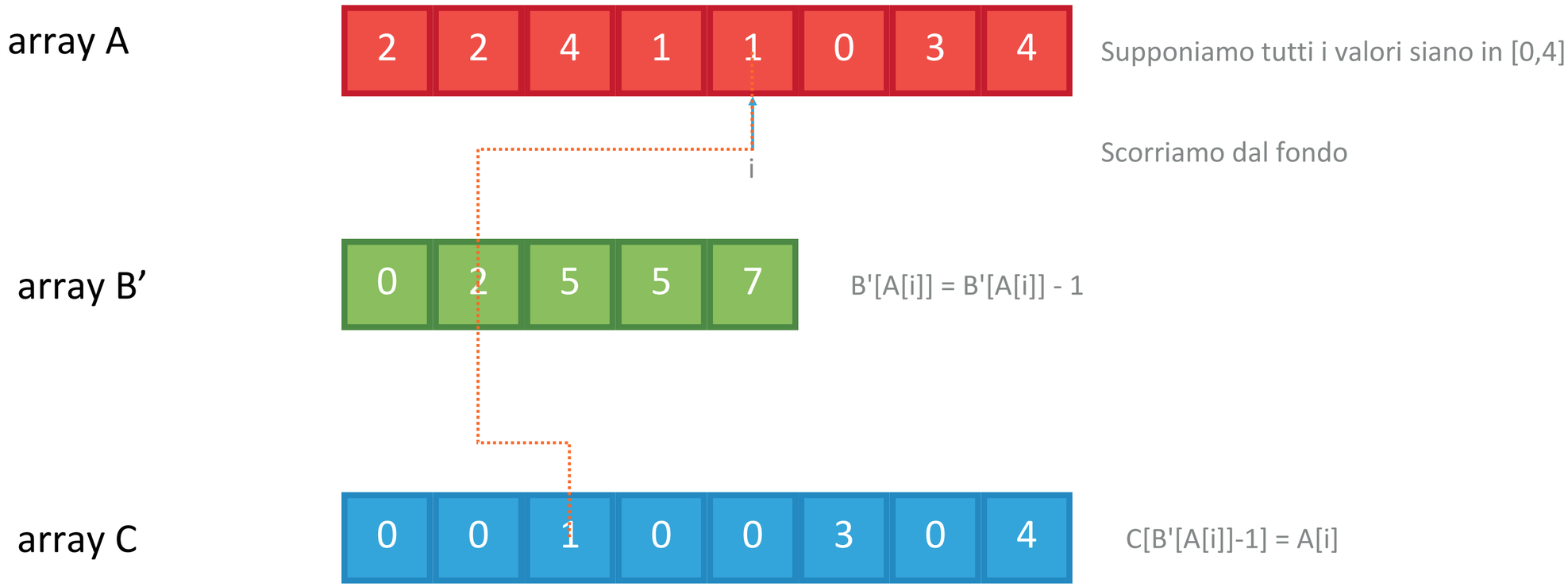
CORRETTA RISPETTO ALLA LEZIONE

ORDINAMENTO



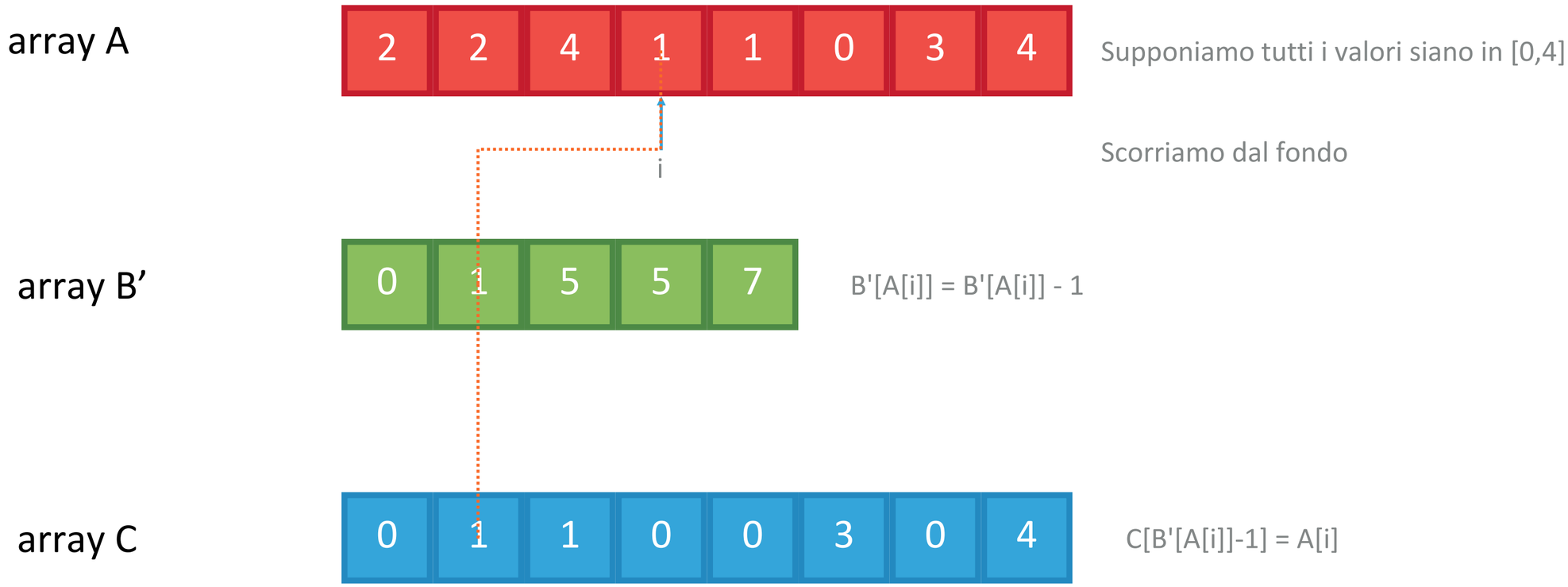
CORRETTA RISPETTO ALLA LEZIONE

ORDINAMENTO



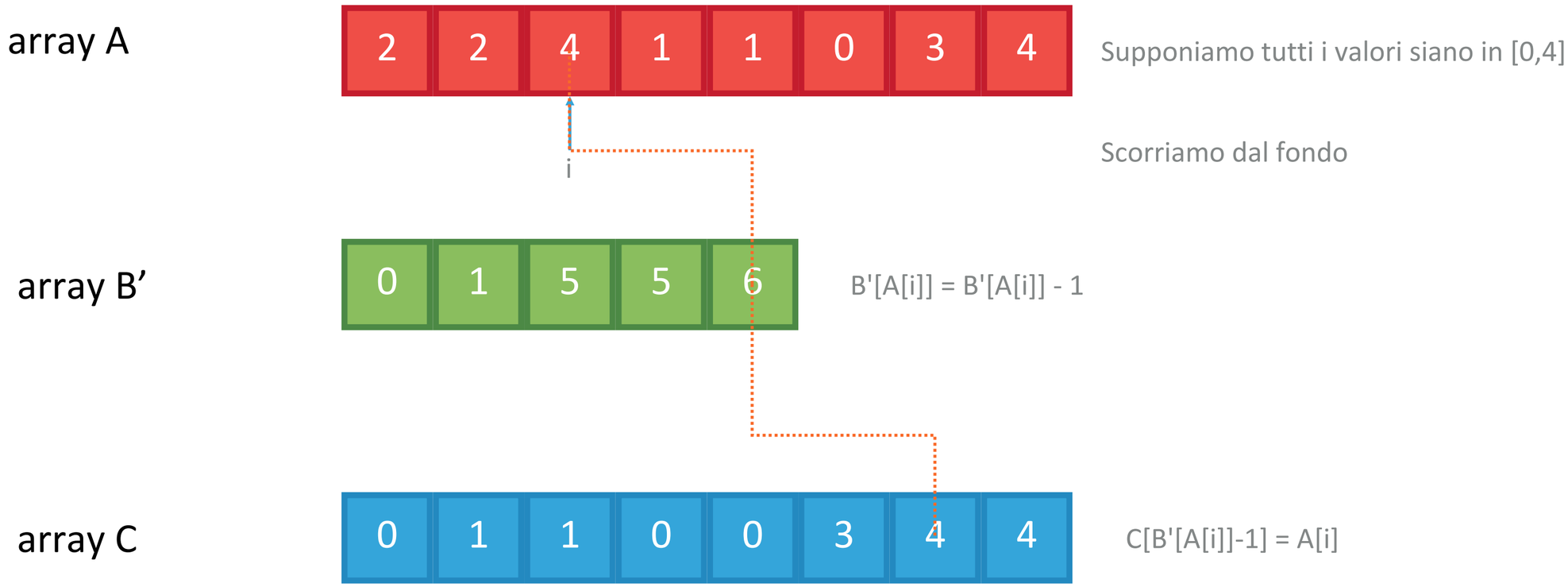
CORRETTA RISPETTO ALLA LEZIONE

ORDINAMENTO



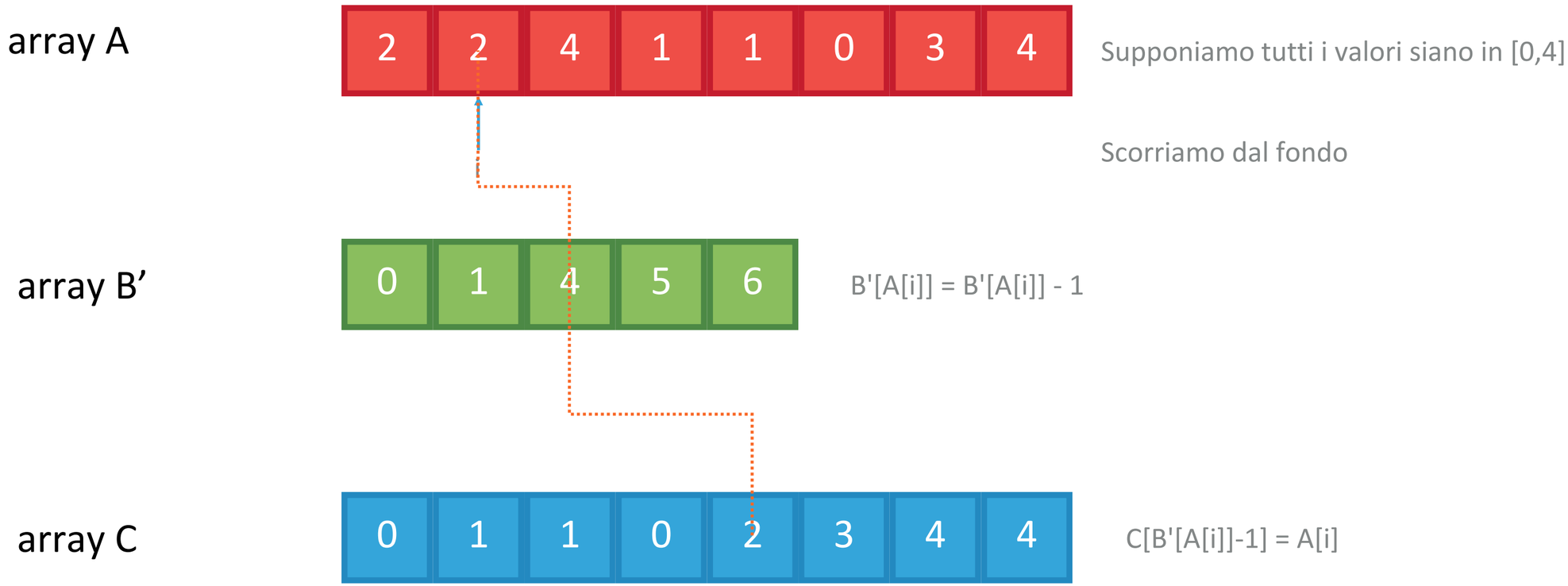
CORRETTA RISPETTO ALLA LEZIONE

ORDINAMENTO



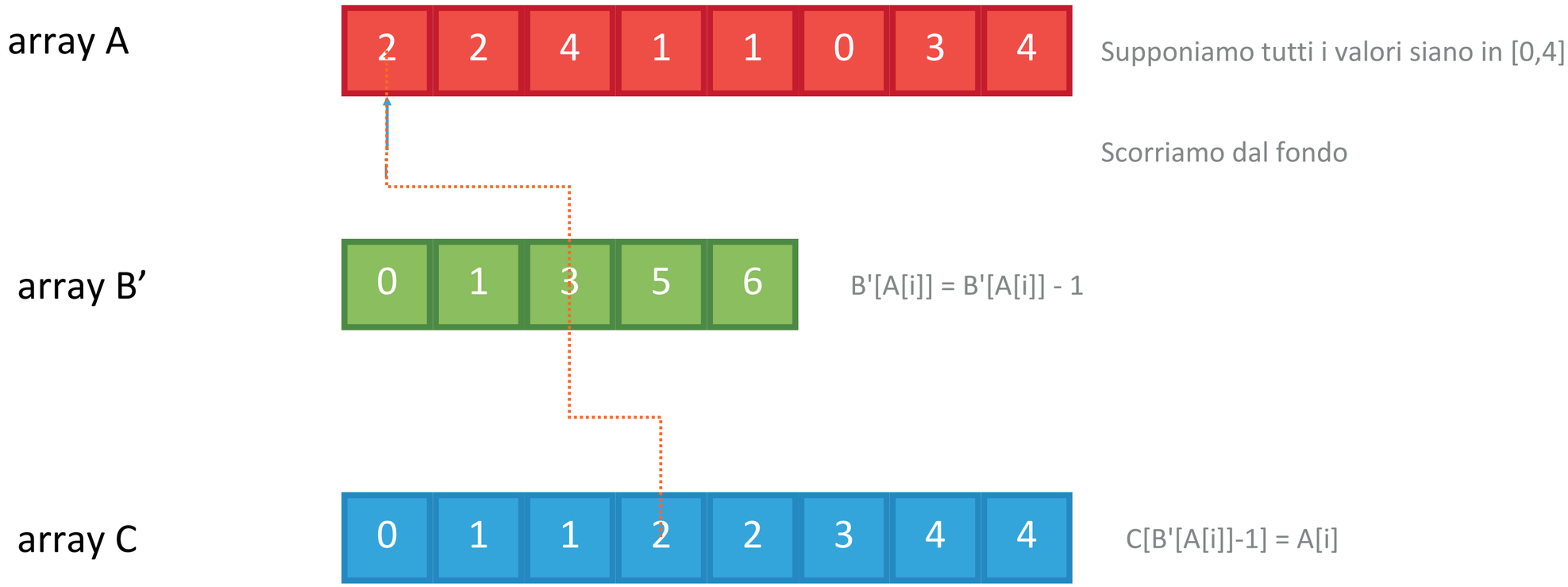
CORRETTA RISPETTO ALLA LEZIONE

ORDINAMENTO



CORRETTA RISPETTO ALLA LEZIONE

ORDINAMENTO



CORRETTA RISPETTO ALLA LEZIONE

PSEUDOCODICE

```
COUNTING-SORT(A,k)     //input: array A con valori da 0 a k
B = array di k+1 elementi inizialmente zero
C = array di len(A) elementi //array finale ordinato
for i=0 to len(A)-1
  B[A[i]] = B[A[i]] + 1 //incrementa il numero di valori A[i] trovati
for i=1 to k
  B'[i] = B[i] + B[i-1] //B'[i] contiene il numero di elementi ≤ i
for i=len(A)-1 downto 0 //contiamo dalla fine all'inizio
  C[B'[A[i]]-1] = A[i] //trasferiamo A[i] nella sua posizione in C
  B'[A[i]] = B'[A[i]] - 1
return C
```

CORRETTEZZA E PRESTAZIONI

Counting Sort è **corretto** perché conta esattamente quante volte compare ogni valore, poi calcola le posizioni finali cumulative e inserisce ogni elemento nel posto giusto, preservando l'ordine (**stabile**). Questo garantisce che l'output sia ordinato correttamente.

La complessità è funzione di due parametri: n (la dimensione dell'array) e k (il numero di valori distinti)

Due cicli for che sono lunghi n cicli

Un ciclo for che è lungo $k - 1$ cicli

Risultato: $\boldsymbol{\Theta(n + k)}$

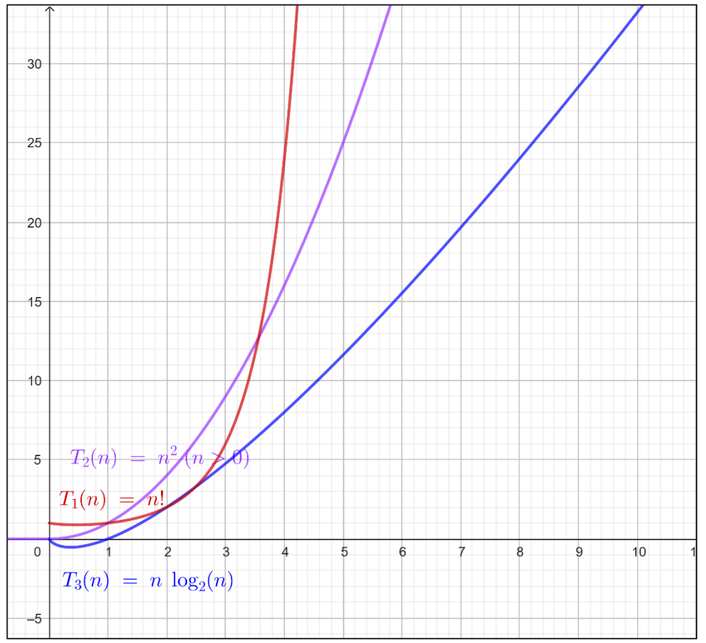
Se abbiamo che $k = O(n)$ otteniamo che l'**algoritmo richiede tempo** $\boldsymbol{\Theta(n)}$

CONFRONTO CON ALTRI ALGORITMI DI ORDINAMENTO

Algoritmo	Caso peggiore	Caso migliore
Bogosort	O(n!)	O(n)
Insertion sort	$O(n^2)$	O(n)
Merge sort	$O(n\log_2 n)$	$O(n\log_2 n)$
Bubble sort	$O(n^2)$	O(n)
Quick sort	$O(n^2)$*	$O(n\log_2 n)$
Counting sort	O(n+k)	O(n+k)

*Caso medio: $O(n\log_2 n)$

Insertionsort Mergesort
Bogosort Bubblesort Quicksort



Note
k piccolo → Counting Sort migliore.
k grande → Counting Sort inefficiente.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

AVVISI

- Seminario Prof. Raganato: annullato → possibile alternativa: vedere in classe il seminario dello scorso anno (per maturazione bonus 0.5/30 ai fini dell'esame)
- Possibile tutorato extra (con Dott. Matteo Gallo) su lab07+lab08 il giorno 20/05 h.14

Prossima lezione (lab): 12 ***maggio, h.14:00, aula 4C***