Titolo unità didattica: Strutture dati: array [07]

Titolo modulo : Algoritmo di fusione di array ordinati [04-T]

Fusione di due array ordinati in un nuovo array ordinato

Argomenti trattati:

- ✓ array ordinato
- ✓ algoritmo incrementale per la fusione di array ordinati
- ✓ costo dell'algoritmo di fusione

Prerequisiti richiesti: AP-07-01-T

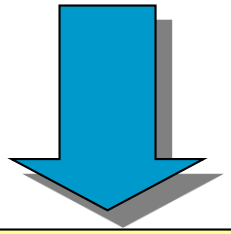
problema:
fusione (**merge**) di due array 1D **ordinati**.
I due array hanno intersezione vuota.

una variabile **a** di tipo array (1D) è **ordinata** (in senso crescente) se

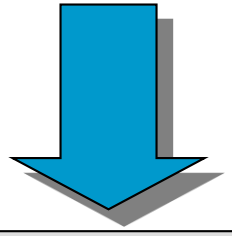
$a[i] \leq a[j]$ per $i < j$

due variabili **a**, **b** di tipo array hanno **intersezione vuota** se non hanno elementi comuni

problema:
fusione (**merge**) di due array 1D **ordinati**.
I due array hanno intersezione vuota.



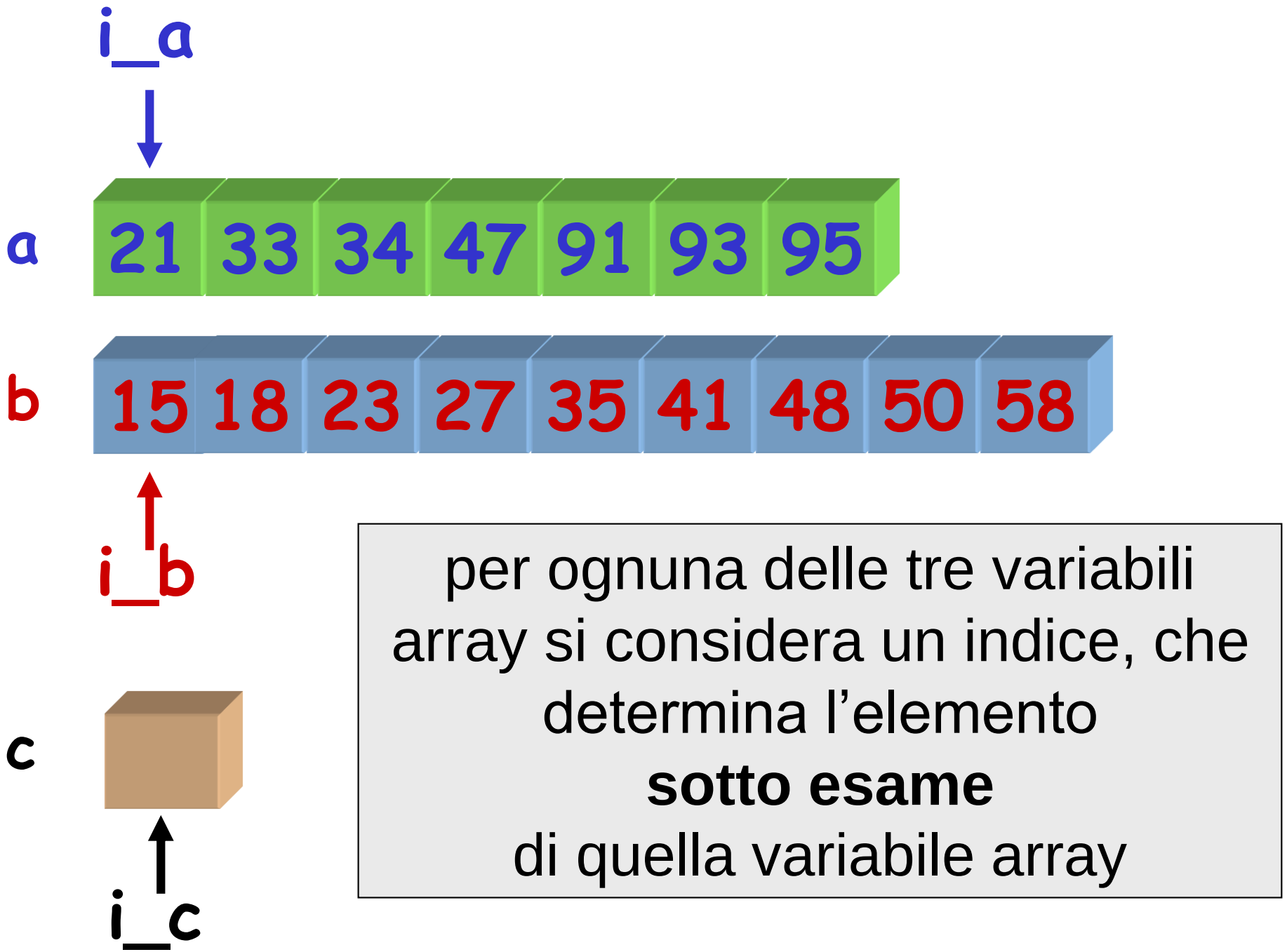
generazione di un array ordinato di size **n** + **m**
a partire da due array ordinati di size **n** e **m**



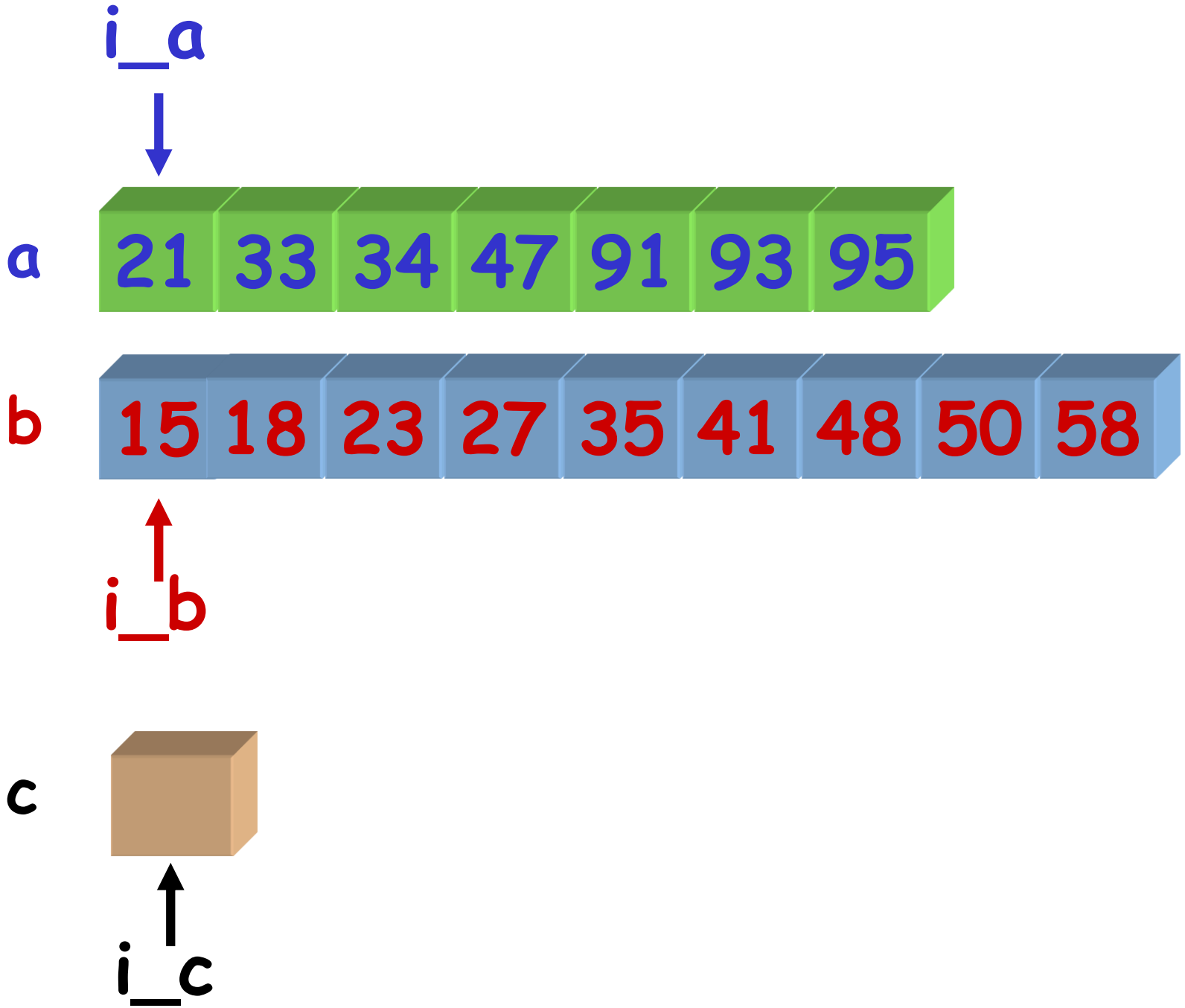
caso particolare del problema di determinazione dell'**unione** di due insiemi

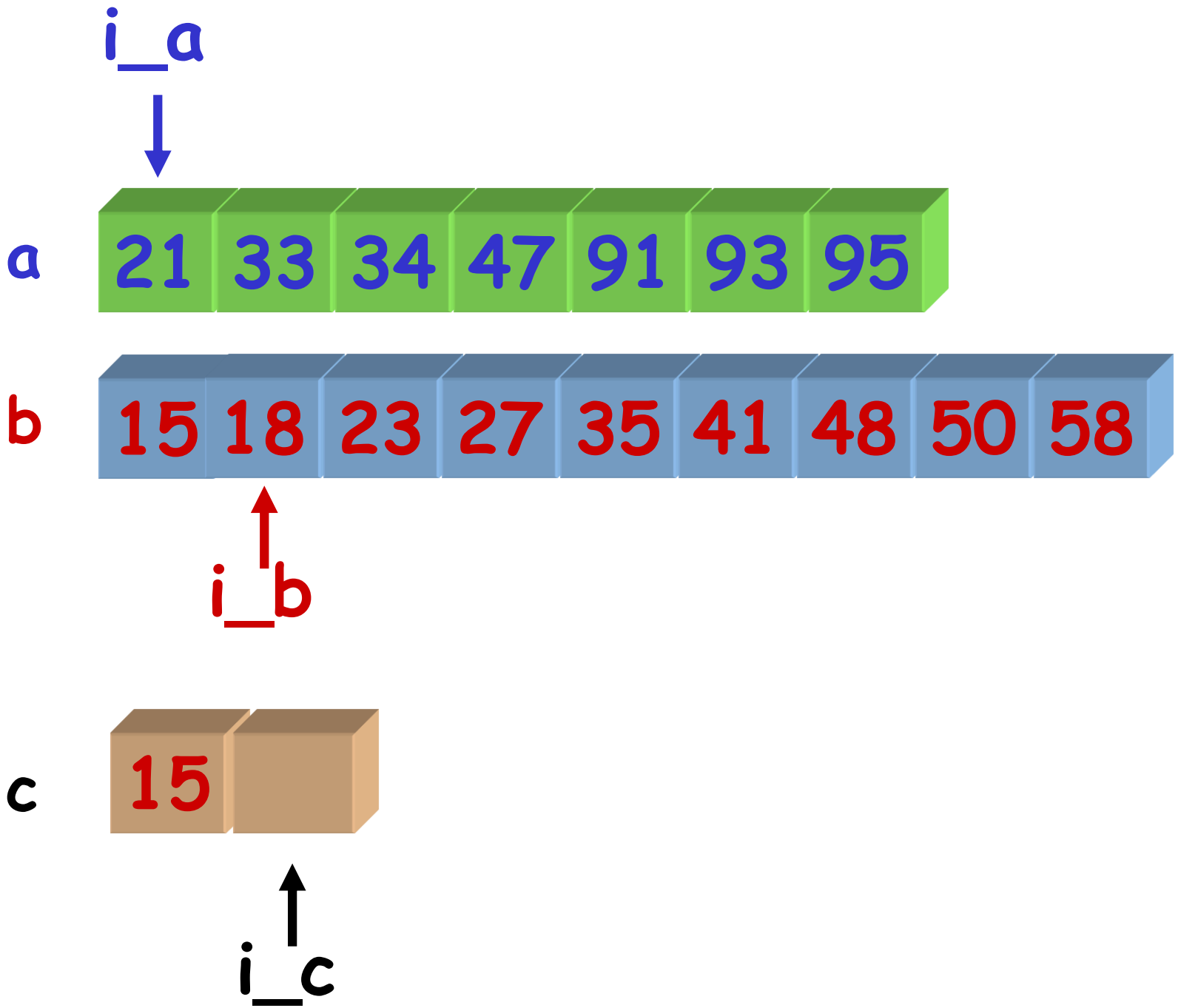
problema:
fusione (**merge**) di due array 1D **ordinati**.
I due array hanno intersezione vuota.

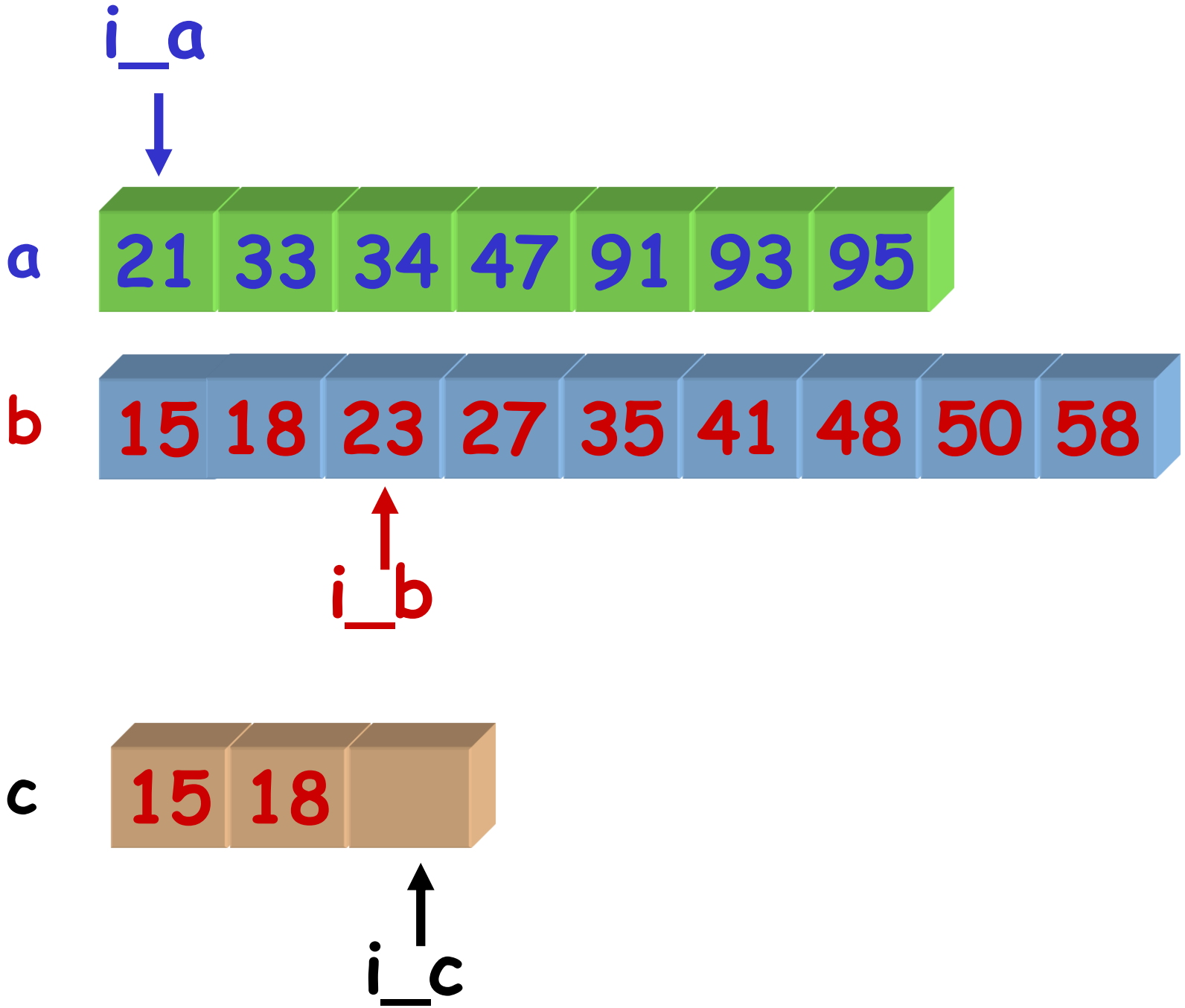
dati di input: il primo array ordinato (variabile **a**), il size del primo array (variabile **n_a**), il secondo array ordinato (variabile **b**), il size del secondo array (variabile **n_b**)
dati di output: l'array "fuso" (variabile **c**)
costrutto ripetitivo: **for**
operazione ripetuta (al generico passo **i**, **i** da **0** a **n_a+n_b-1**):
determinare l'**i**-simo elemento dell'array **c**, considerando il minore tra i due elementi sotto esame di **a** e di **b**

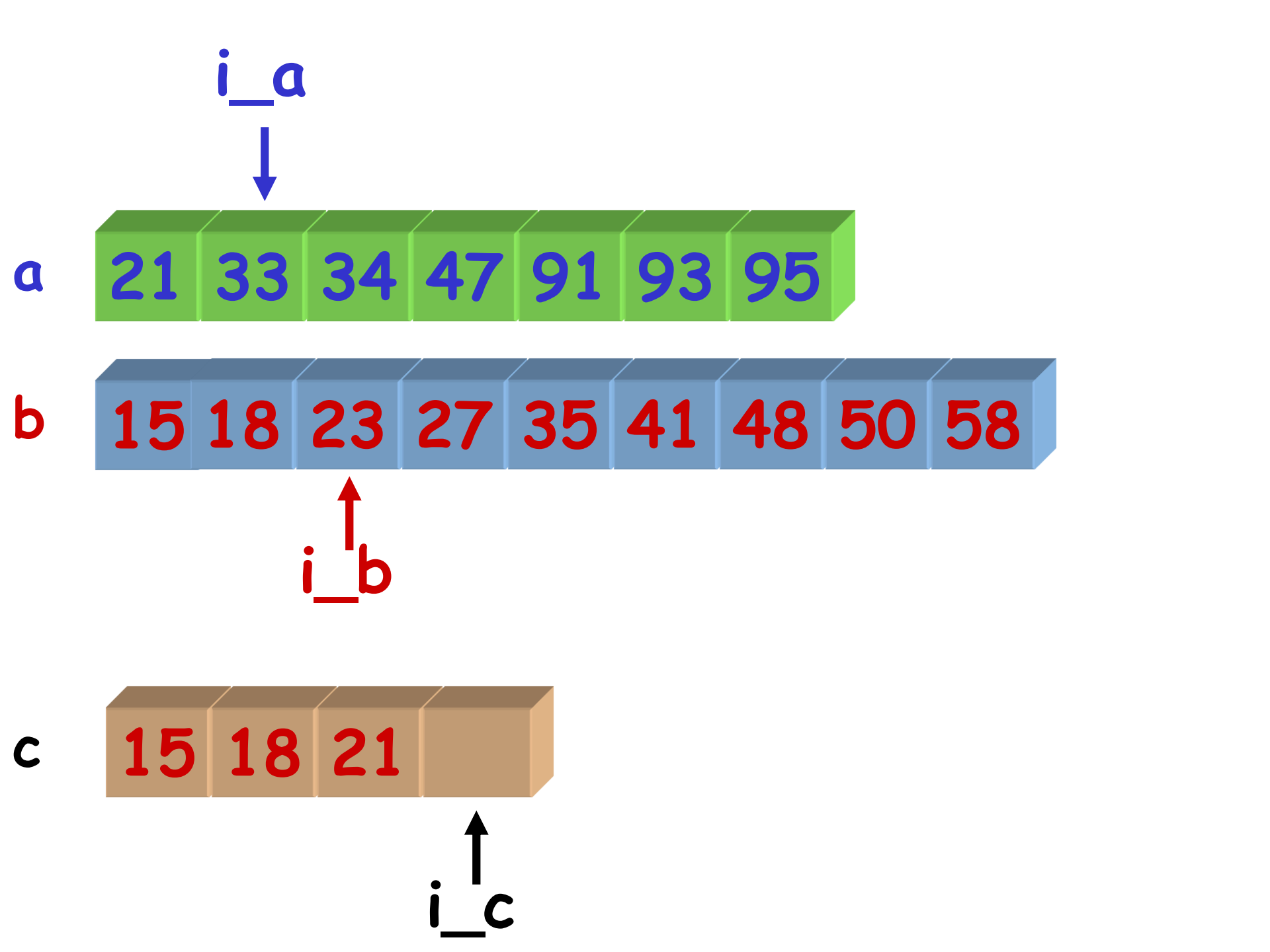


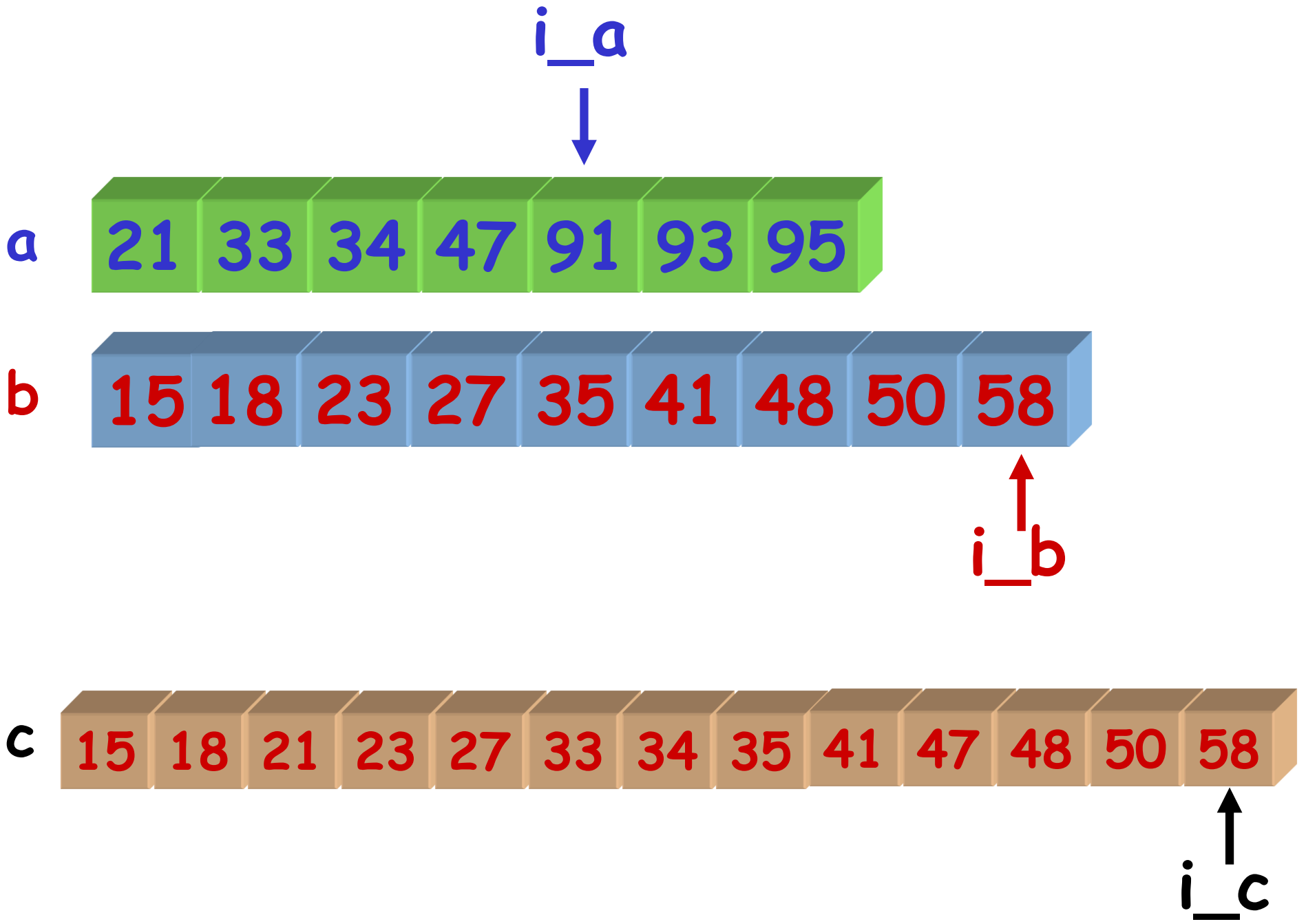
per ognuna delle tre variabili array si considera un indice, che determina l'elemento **sotto esame** di quella variabile array


i_a
a
21 33 34 47 91 93 95
b
15 18 23 27 35 41 48 50 58
i_b
c
i_c


i_a
a
21 33 34 47 91 93 95
b
15 18 23 27 35 41 48 50 58
i_b
c
15
i_c


i_a
a
21 33 34 47 91 93 95
b
15 18 23 27 35 41 48 50 58
i_b
c
15 18
i_c


i_a
a
21 33 34 47 91 93 95
b
15 18 23 27 35 41 48 50 58
i_b
c
15 18 21
i_c


i_a
a
21 33 34 47 91 93 95
b
15 18 23 27 35 41 48 50 58
i_b
c
15 18 21 23 27 33 34 35 41 47 48 50 58
i_c

dati di input: il primo array ordinato (variabile **a**), il size del primo array (variabile **n_a**), il secondo array ordinato (variabile **b**), il size del secondo array (variabile **n_b**)

dati di output: l'array "fuso" (variabile **c**)

costrutto ripetitivo: **for**

operazione ripetuta (al passo **i_c**):

generare l'**i_c**-simo elemento dell'array **c**

selezione tra l'elemento corrente di **a** e l'elemento corrente di **b**

i_a indica l'elemento corrente di **a** e **i_b** indica l'elemento corrente di **b**

```
void fusione(in: char a[],int n_a, char b[], int
n_b; out: char c[]) {
 int i_c;
 for (i_c = 0; i_c < n_a+n_b; i_c++) {

    COSTRUIRE c[i_c]

  }
}
```

prima versione (da raffinare) dell'algoritmo di fusione

```
void fusione(in: char a[],int n_a, char b[], int
n_b; out: char c[]) {
 int i_c;
 for (i_c = 0; i_c < n_a+n_b; i_c++) {
    if  (ci sono sia elementi di a  sia di  b  da
      prendere in considerazione )
      {
        c[i_c] è il minimo tra l'elemento corrente
        di  a e l'elemento corrente di  b
      }
....
```

seconda versione (da raffinare)
dell'algoritmo di fusione

```
/* uno dei due array è stato inserito
    completamente in c */
else { if (l'array b non deve essere più considerato)
       { c[i_c] = l'elemento corrente di a }
       else
       { c(i_c] = l'elemento corrente di b }
  }
 }
}
```

commenti

seconda versione (da raffinare)
dell'algoritmo di fusione

```
void fusione(in: char a[],int n_a, char b[], int
n_b; out: char c[]) {
 int i_a, i_b, i_c;
 i_a = 0 ;
 i_b = 0 ;
 for (i_c = 0; i_c < n_a+n_b; i_c++) {
    if (i_a < n_a && i_b < n_b) {
/* ci sono sia elementi di a sia di b da
   prendere in considerazione */
        if (a[i_a] < b[i_b])
          { c[i_c] = a[i_a] ;
            i_a = i_a+1 ; }
      ...
```

commenti

ATTENZIONE: da modificare in C

```
            else
              {c[i_c] = b[i_b] ;
               i_b = i_b+1 ; } }
/* uno dei due array non deve essere più usato */
     else { if (i_b >= n_b)
     /* considerare solo a  */
              { c[i_c] = a[i_a];
                i_a = i_a+1 ; }
            else
     /* considerare solo b  */
              { c[i_c] = b[i_b] ;
                i_b = i_b+1 ; }
       }
    }
}
```

ATTENZIONE: da modificare in C

costo dell'algoritmo di fusione

```
i_a = 0 ; i_b = 0 ;
  for (i_c=0; i_c < n_a+n_b; i_c++) {
    if (i_a < n_a && i_b < n_b) {
        if (a[i_a] < b[i_b])
            c[i_c] = a[i_a], i_a = i_a+1 ;
        else
            c[i_c] = b[i_b], i_b = i_b+1;
     }
     else { if (i_b >= n_b)
            c[i_c] = a[i_a], i_a = i_a+1 ;
          else
            c[i_c] = b[i_b], i_b = i_b+1 ;
    }
  }
```

n_a+n_b

confronti (tra elementi di **a** e di **b**)

al più