


Merge Sort

Algoritmo di ordinamento

- ▶ *Problema*: partendo da un array di numeri, ordinare l'array.
- ▶ *Array ordinato*: array nel quale ogni posizione contiene un numero \leq (o $=$) del numero nella posizione successiva:
 - `array[i] <= array[i+1]`
(con `i` che va da 0 a `array.size() - 1`)

Merge Sort

- ▶ Il **merge sort** è un algoritmo di ordinamento
 - ▶ È utilizzato per ordinare gli elementi di un array in maniera (de)crescente
 - ▶ Più rapido di altri algoritmi (es: selection sort)
 - ▶ Segue la metodologia ricorsiva del divide-et-impera
- 

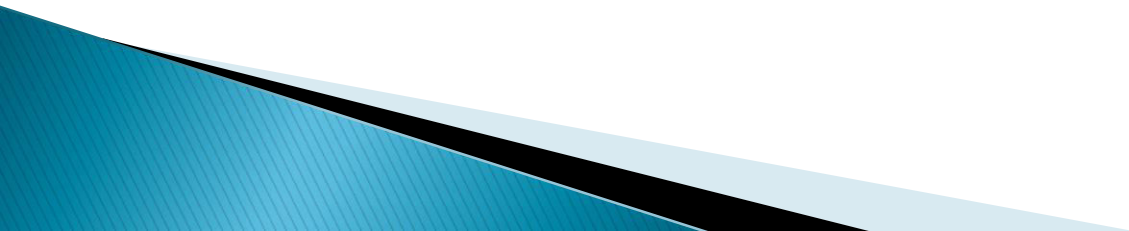
Divide et impera

- ▶ Gli algoritmi Divide et impera (Divide and conquer in inglese) suddividono il problema principale in problemi più piccoli, facilmente risolvibili, per risalire poi alla soluzione generale.
 - *Divide*: divide il problema in sotto-problemi dello stesso tipo, fino ad arrivare a un sotto-problema di base, la cui soluzione è nota.
 - *Impera*: si combinano le soluzioni dei problemi base, fino a risalire al problema principale e alla relativa soluzione


Merge Sort

- ▶ **Divide:** si divide (split) l'array in due parti, in maniera ricorsiva, fino a che non si ottengono array di una elemento
 - A questo punto, la soluzione del problema “ordina un array di un elemento” è facile: l'array è già ordinato
- ▶ **Impera:** partendo da 2 array ordinati, si effettua il “merge” ordinato degli array
 - Si inseriscono gli elementi dei due array in un nuovo array , uno alla volta, scegliendo ogni volta l'elemento più piccolo. Il risultato è un array ordinato

Esempio: ordinamento di un array
tramite Merge sort



Merge Sort

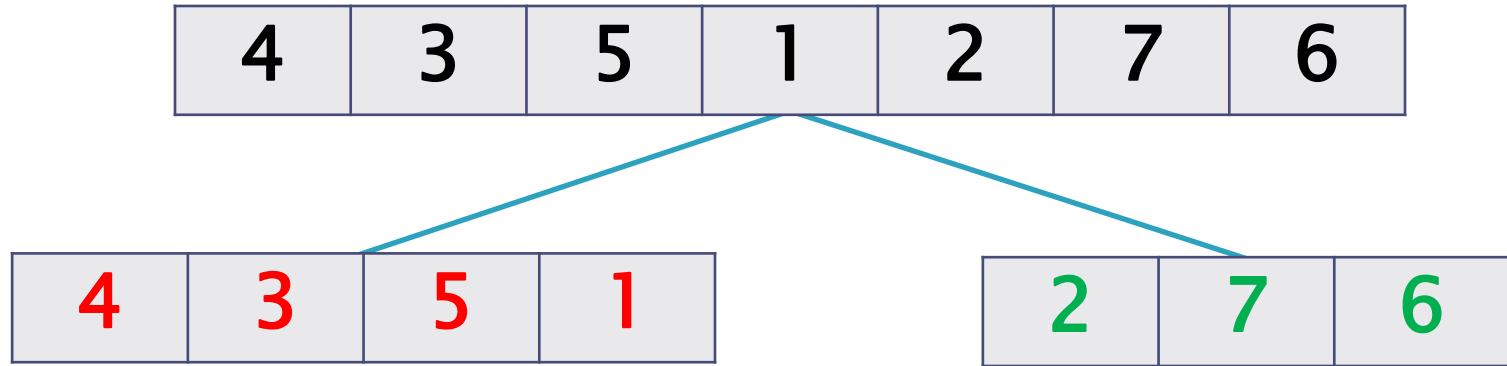
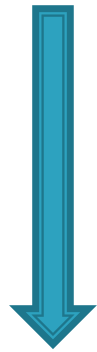
- ▶ Vediamo un esempio di applicazione algoritmo Merge sort.
 - ▶ Verranno usati i seguenti colori per facilitare visivamente il processo:
 - Rosso: array a sinistra
 - Verde: array a destra
 - Freccia rossa verticale: indice dell'array a sinistra durante il merge
 - Freccia verde verticale: indice dell'array a destra durante il merge
- 

Array iniziale

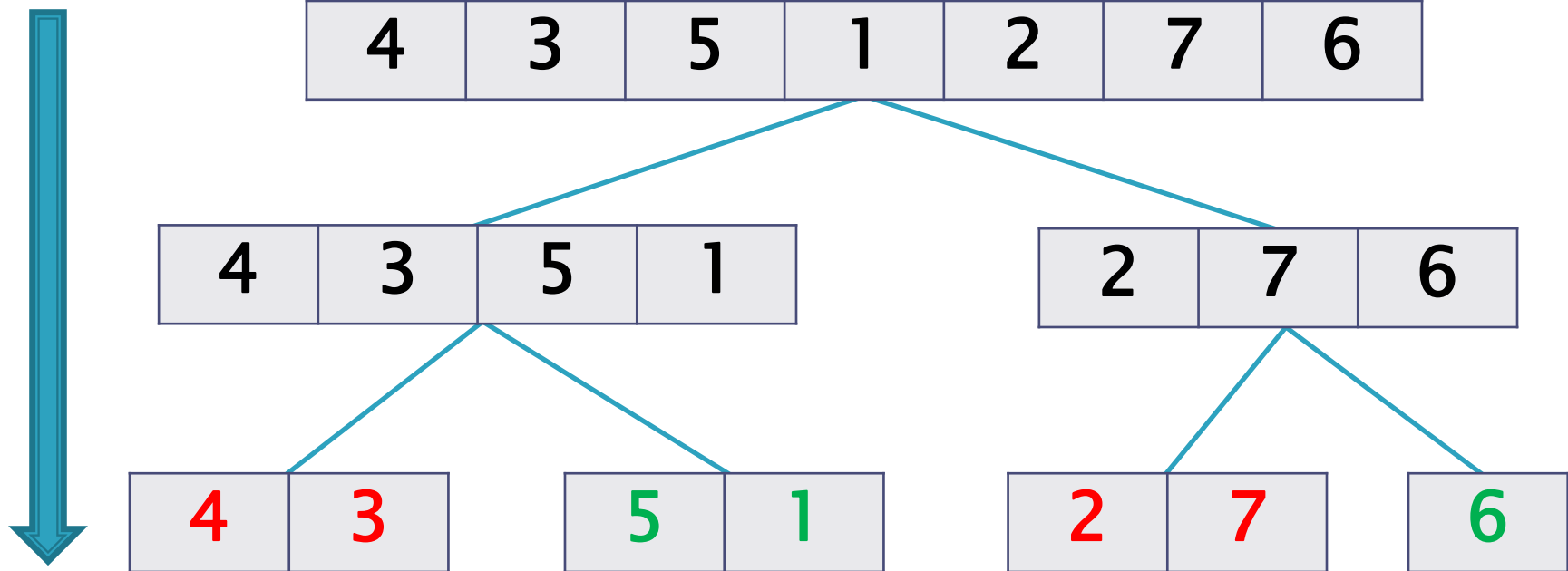
4	3	5	1	2	7	6
---	---	---	---	---	---	---

- Obiettivo: ordinare l'array

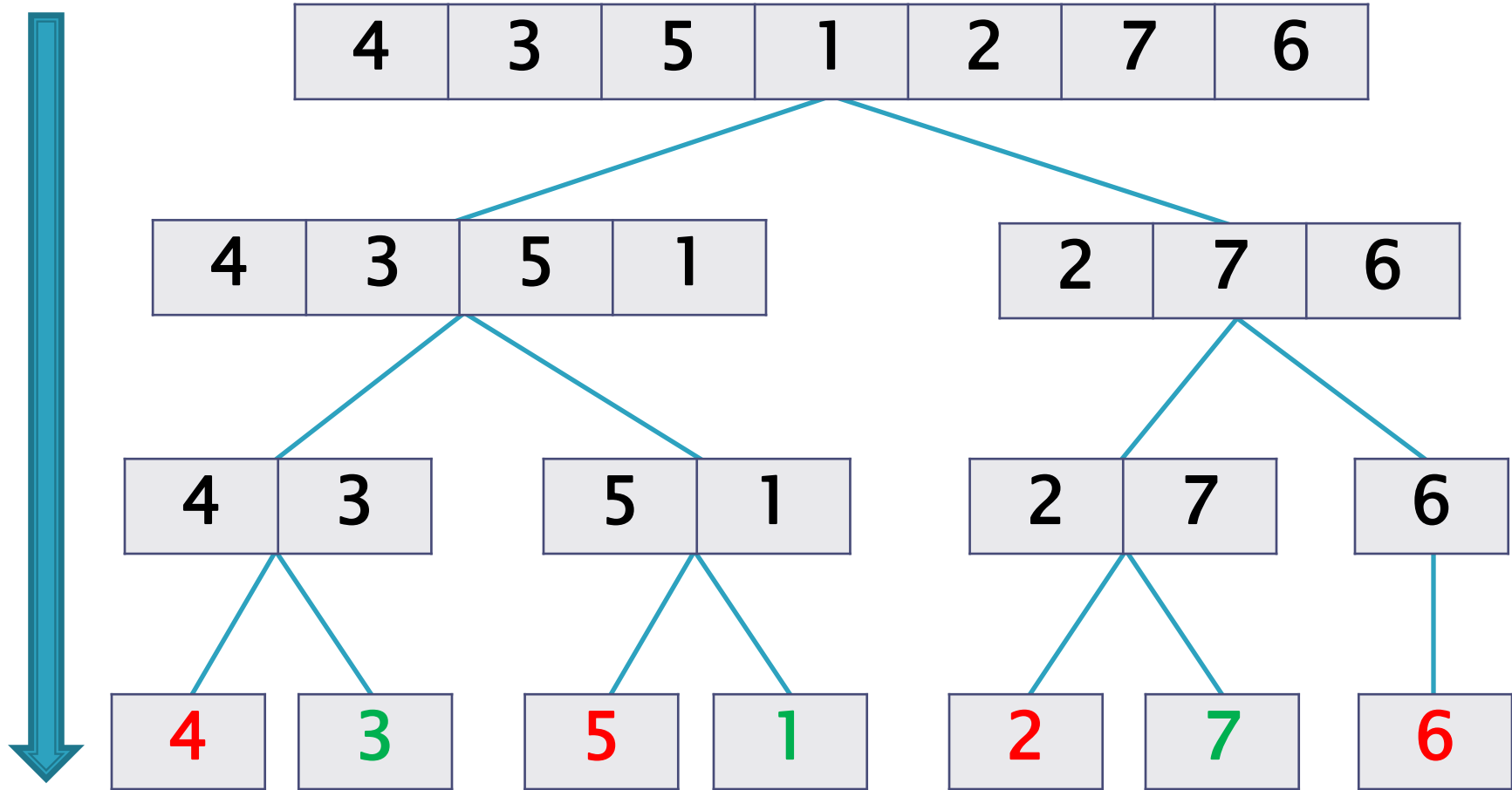
Divide (1)



Divide (2)

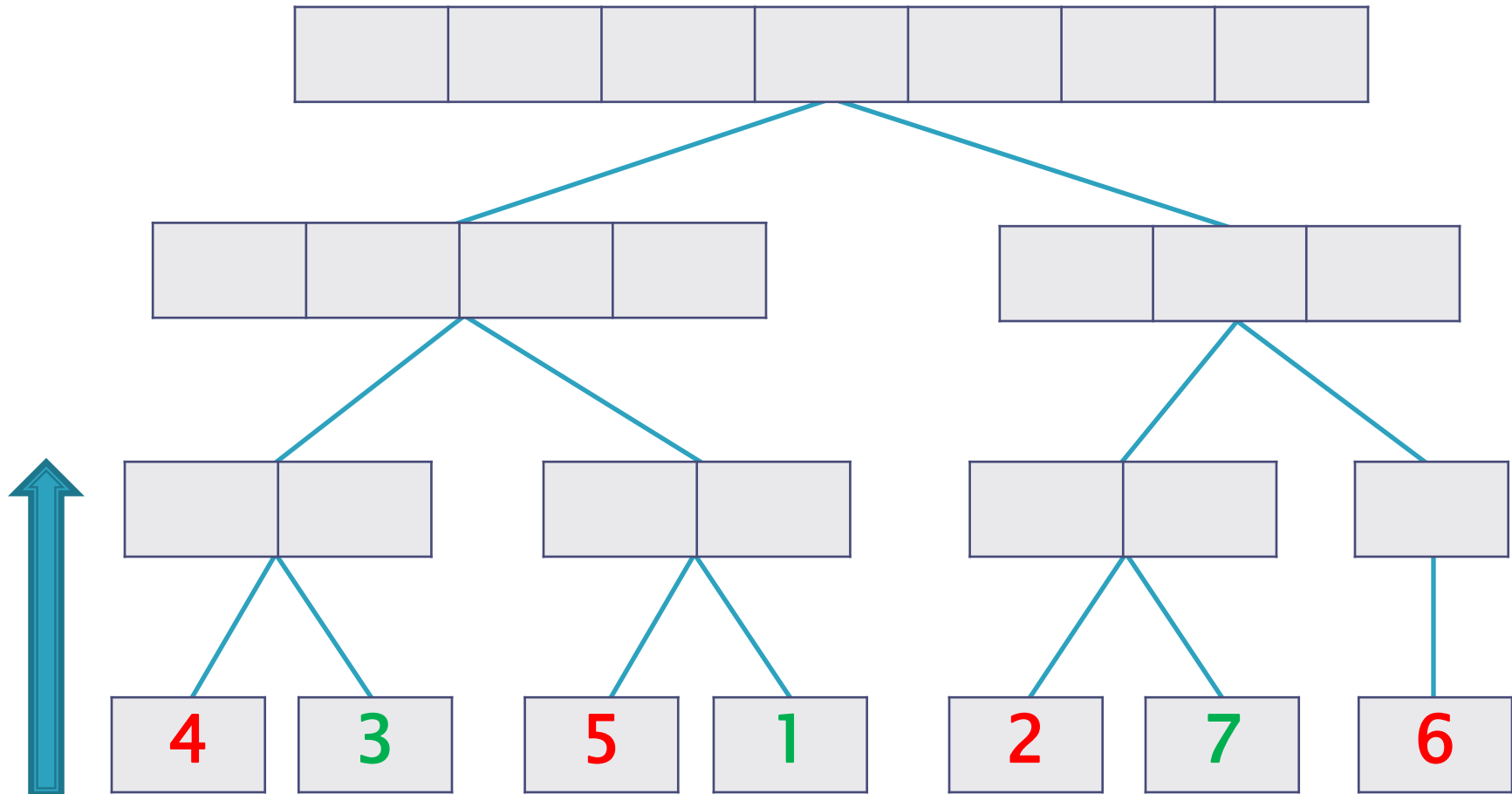


Divide (3)



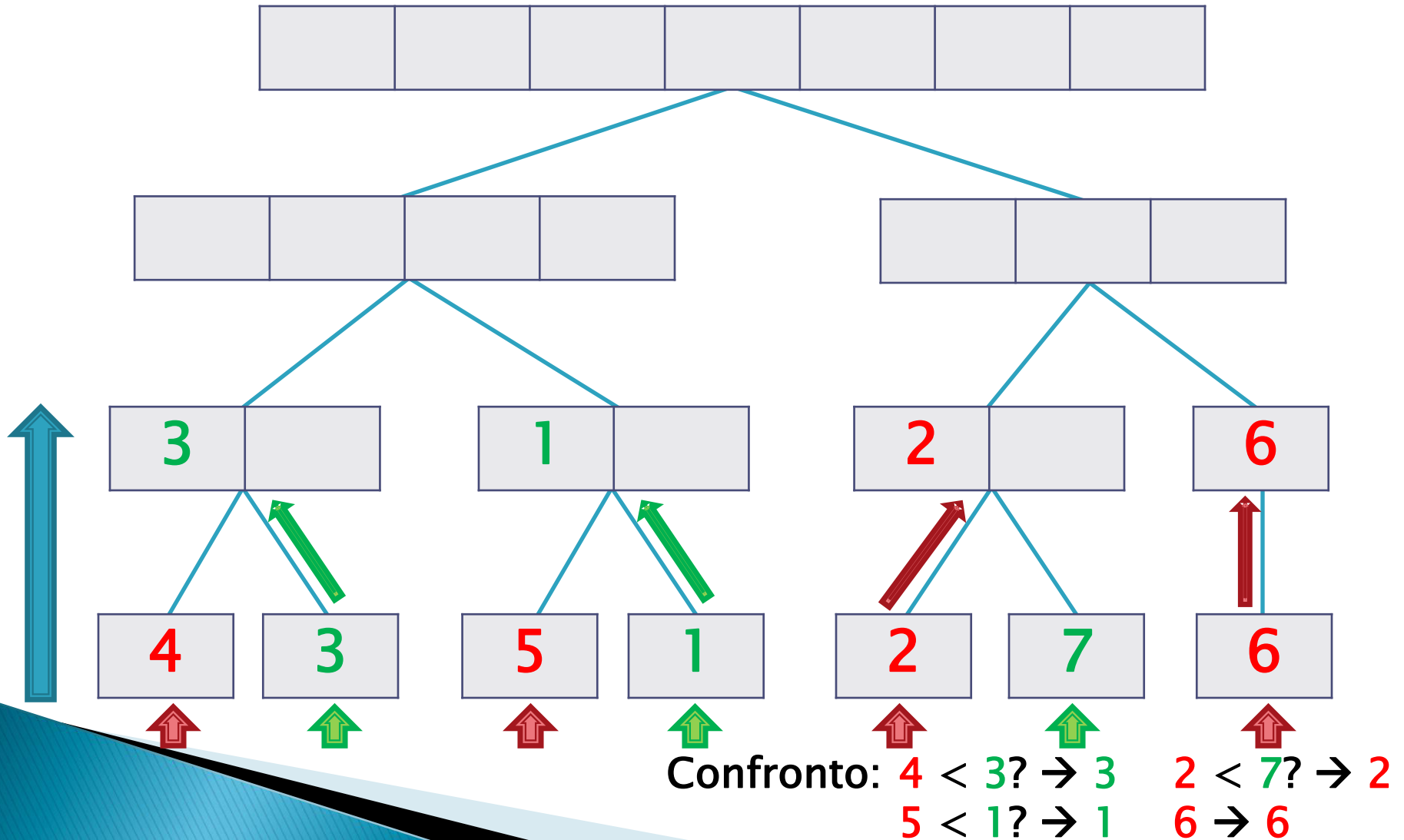
STOP: siamo arrivati alla condizione base: array di un singolo elemento

Impera (1)

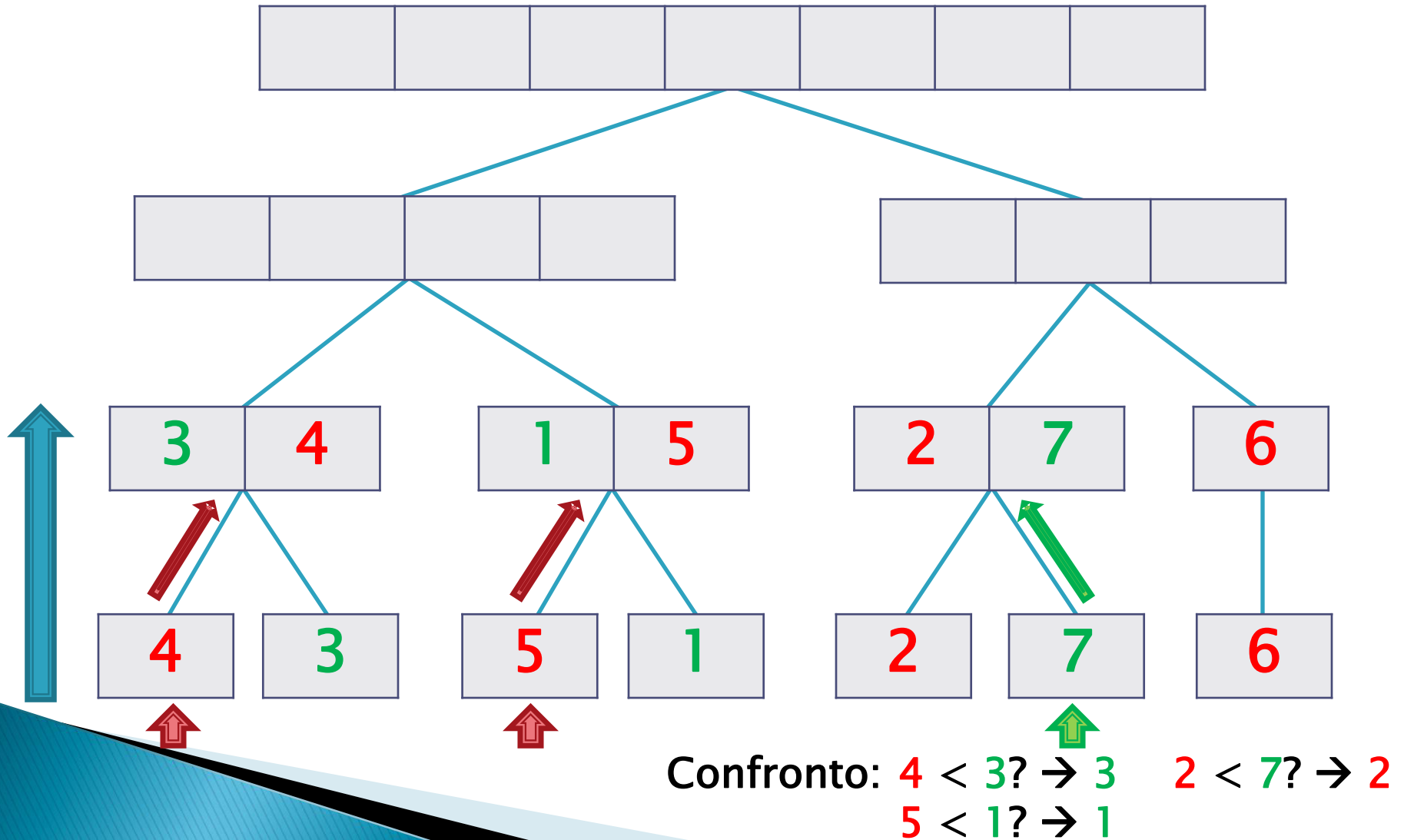


Impera: si risale facendo il merge ordinato degli array, ottenendo un array ordinato

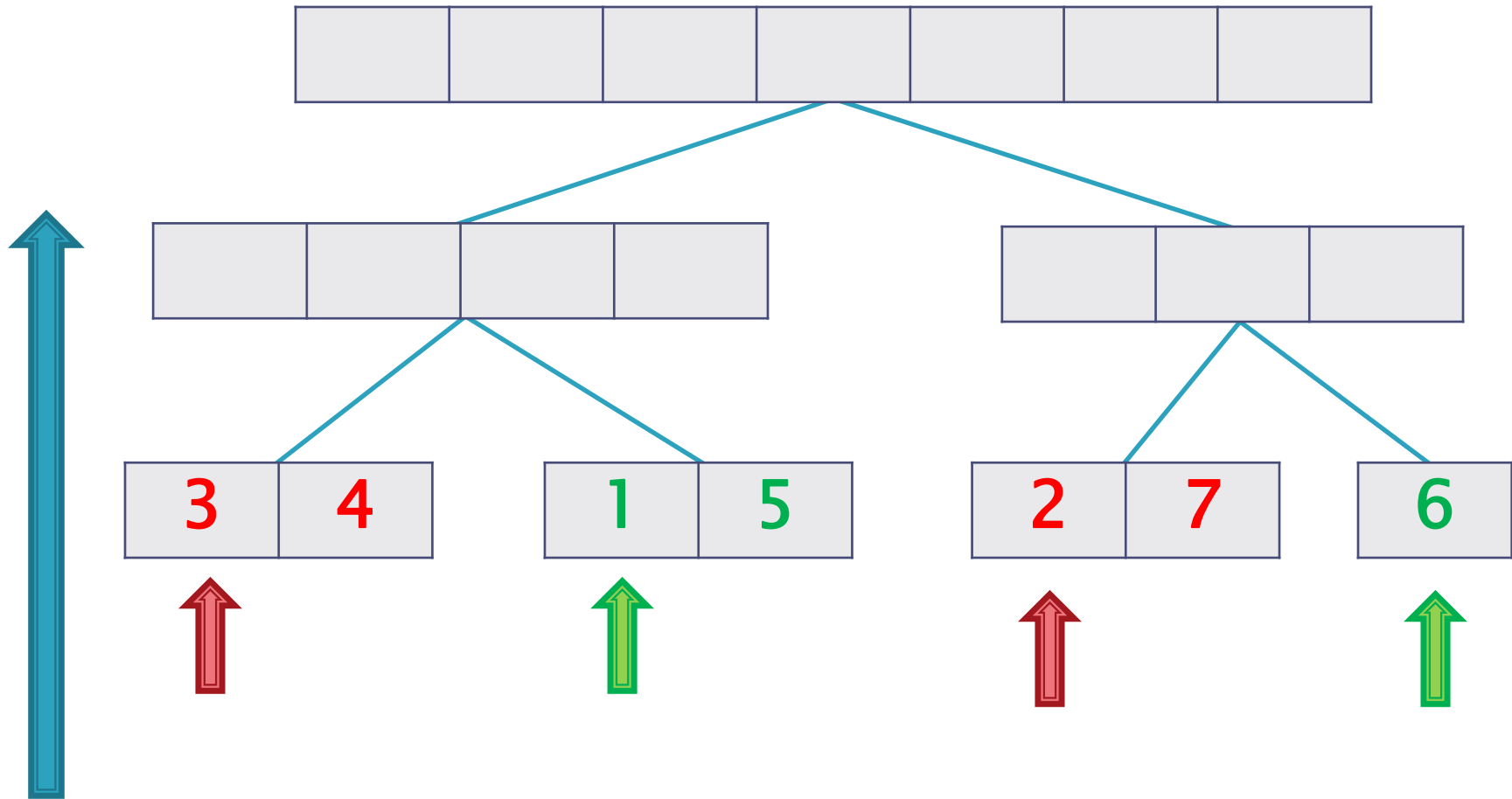
Impera (1.2)



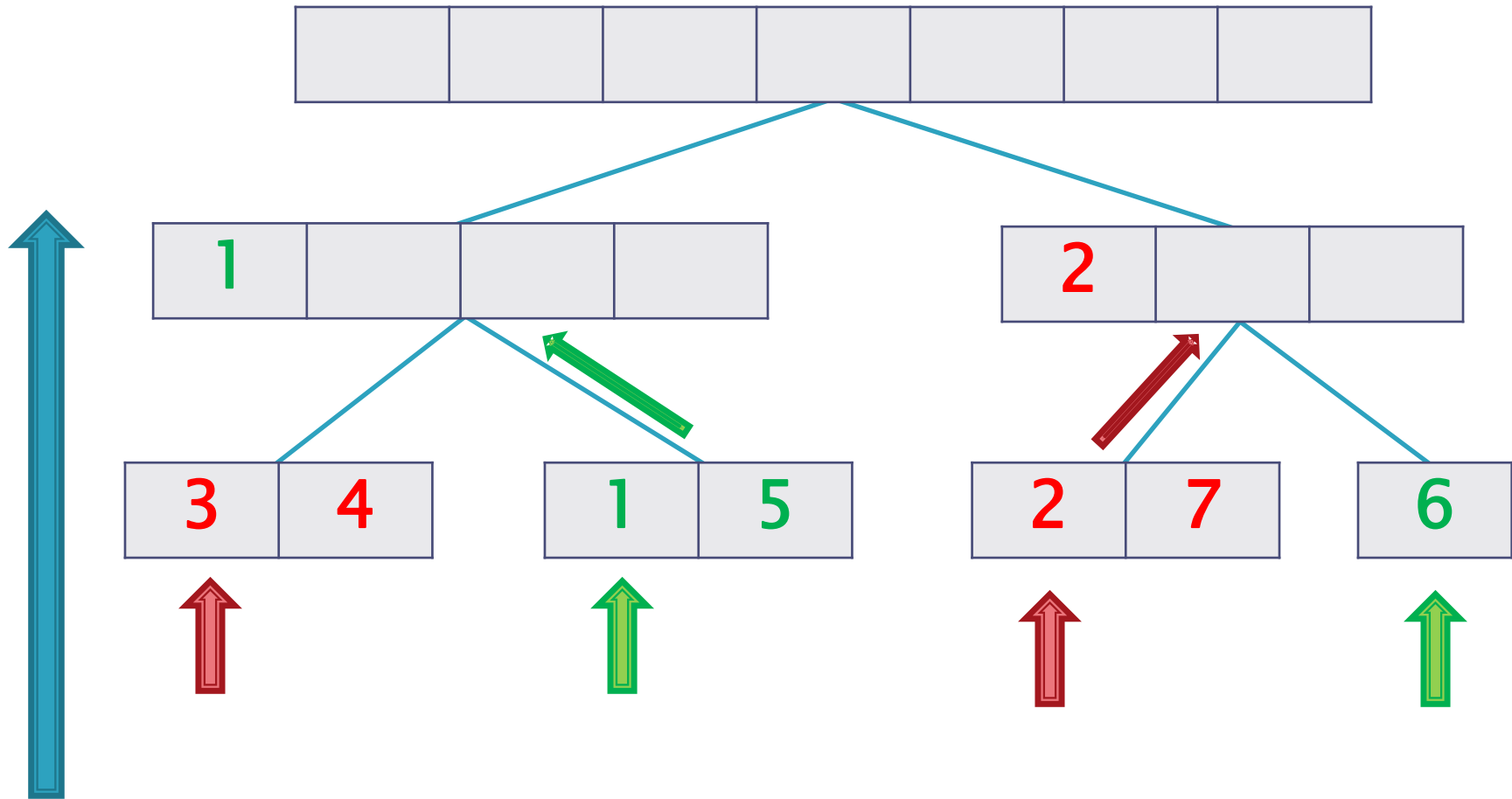
Impera (1.3)



Impera (2)

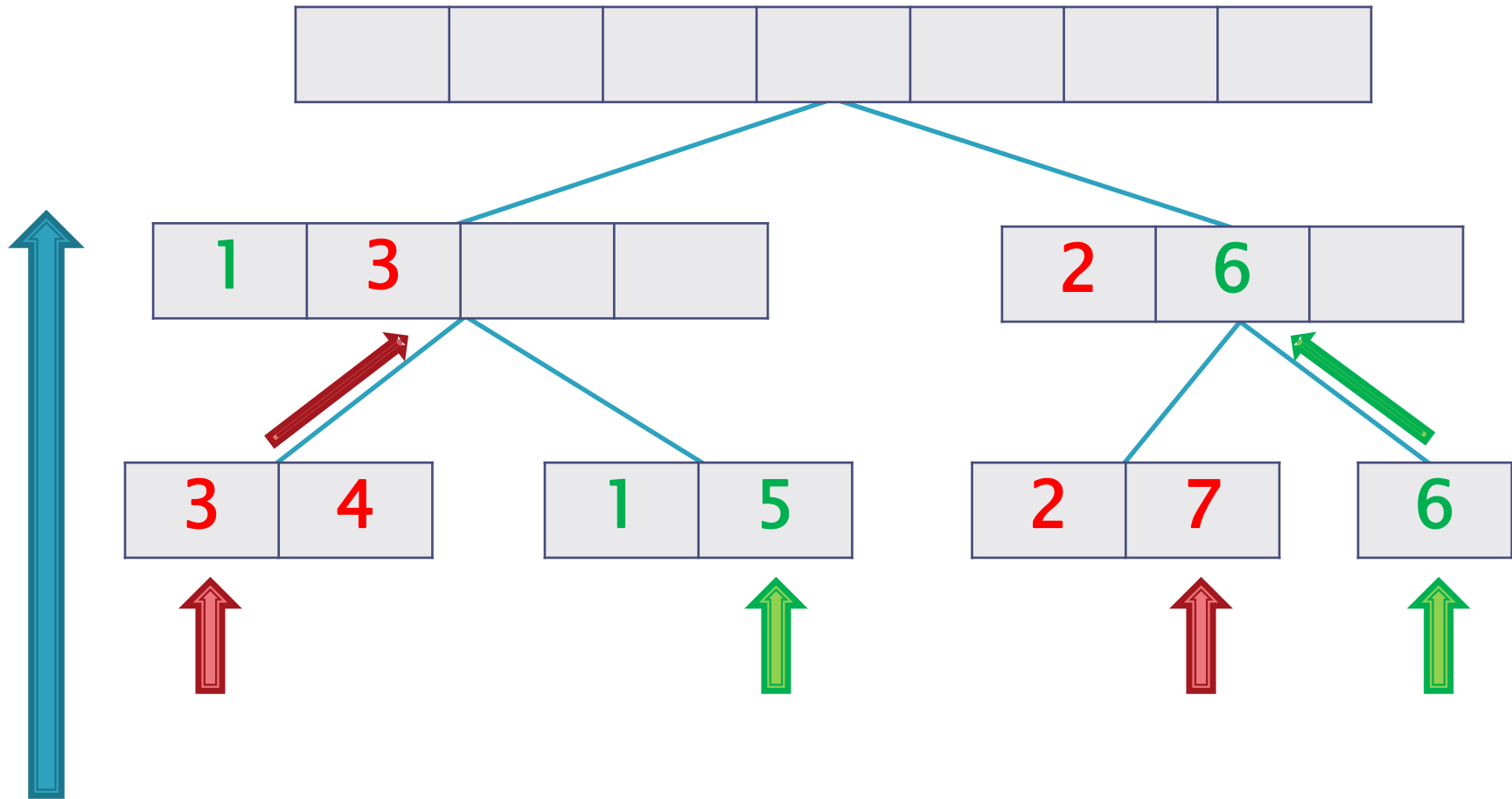


Impera (2.2)



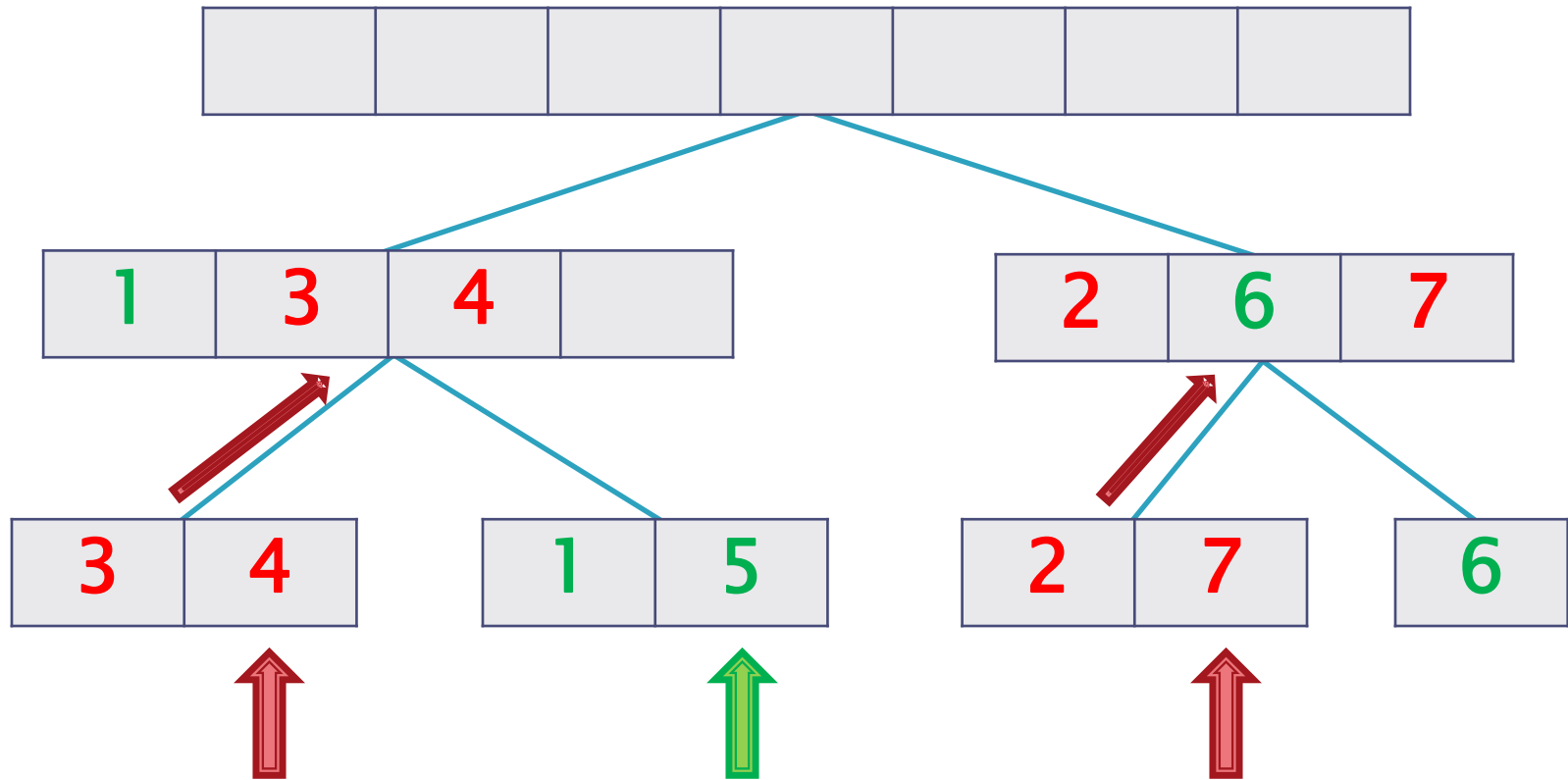
Confronto: $3 < 1? \rightarrow 1$
 $2 < 6? \rightarrow 2$

Impera (2.3)



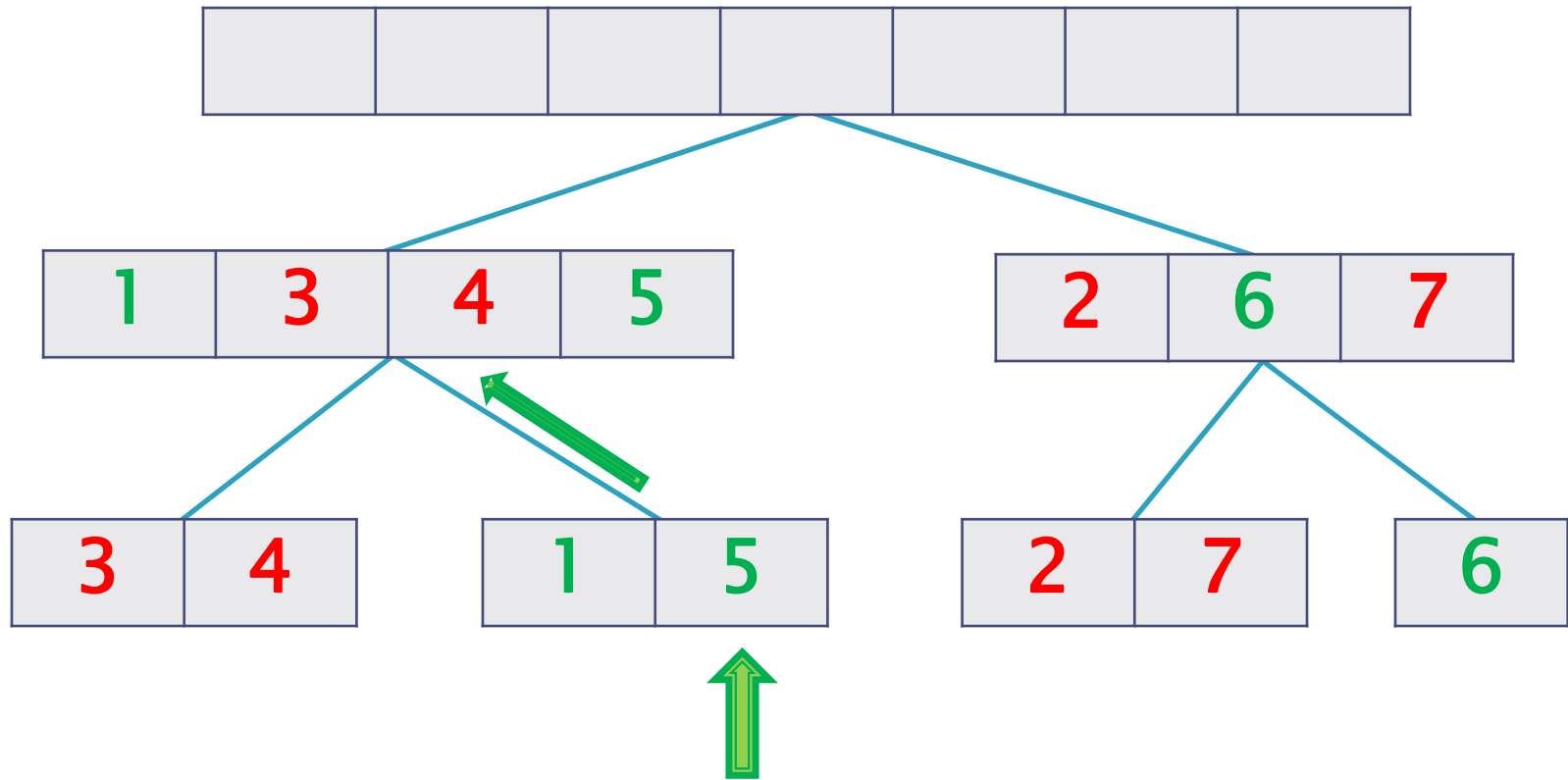
Confronto: $3 < 5? \rightarrow 3$
 $7 < 6? \rightarrow 6$

Impera (2.4)



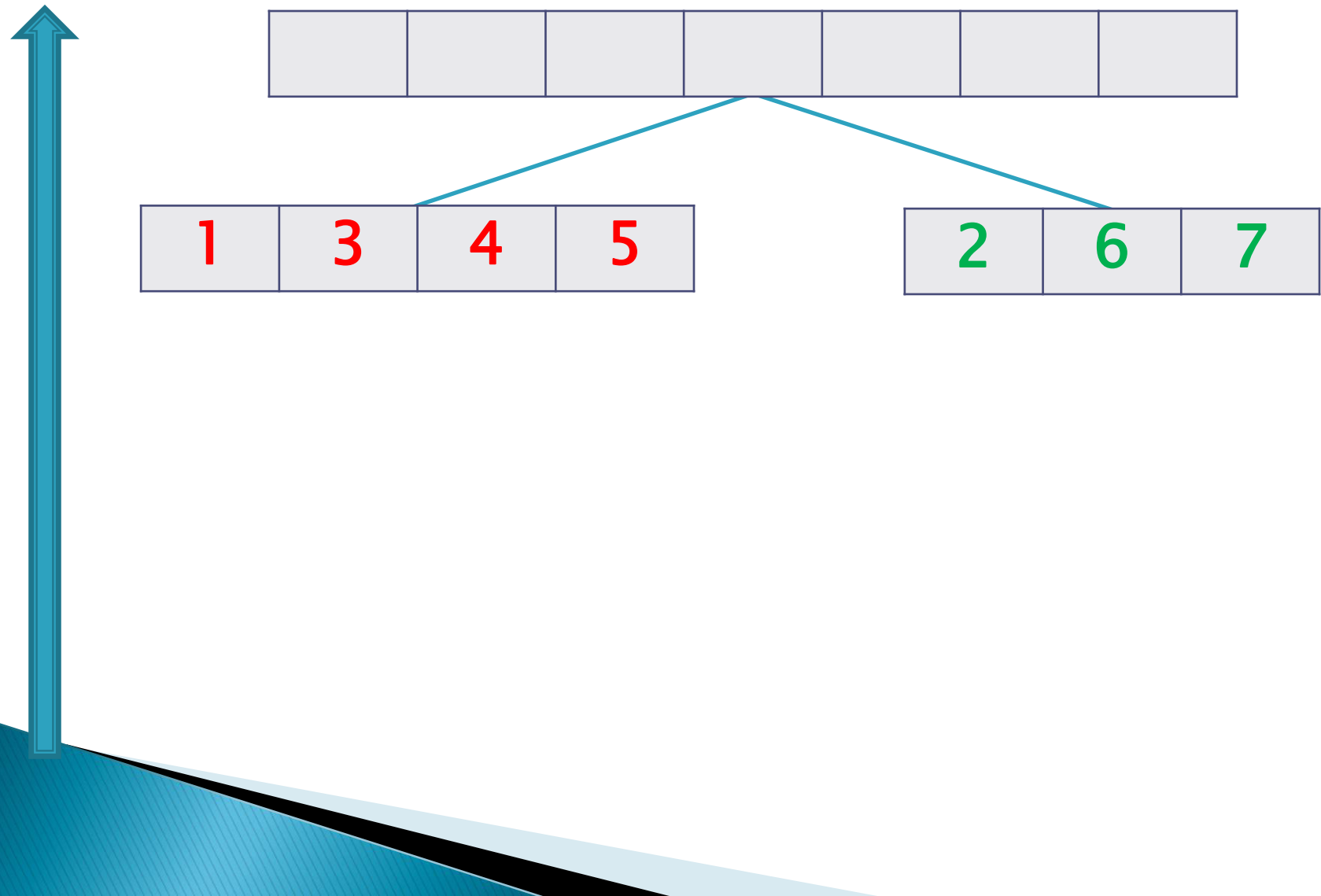
Confronto: $4 < 5? \rightarrow 4$
 $7 \rightarrow 7$

Impera (2.5)

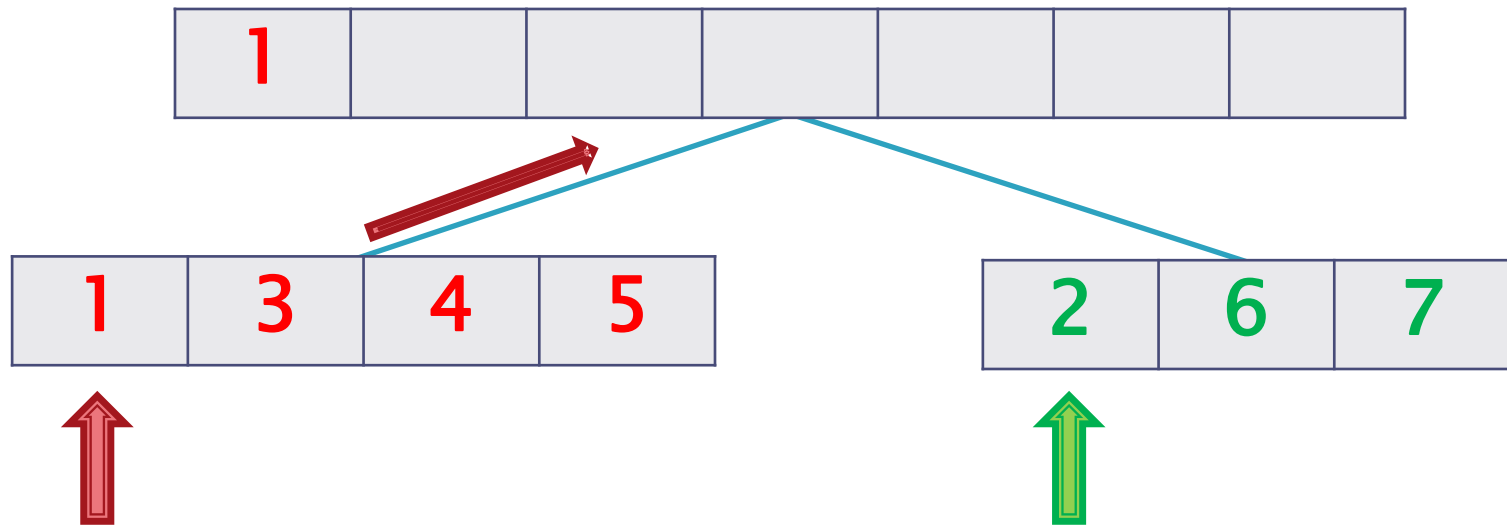


Confronto: 5 → 5

Impera (3)

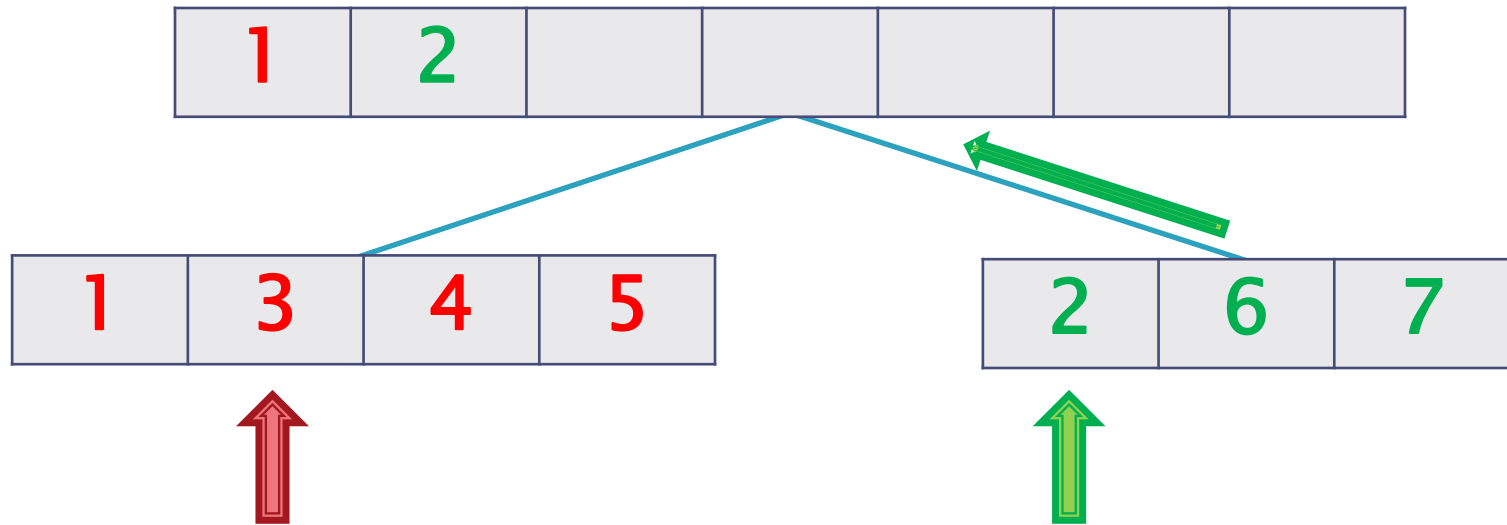


Impera (3.2)



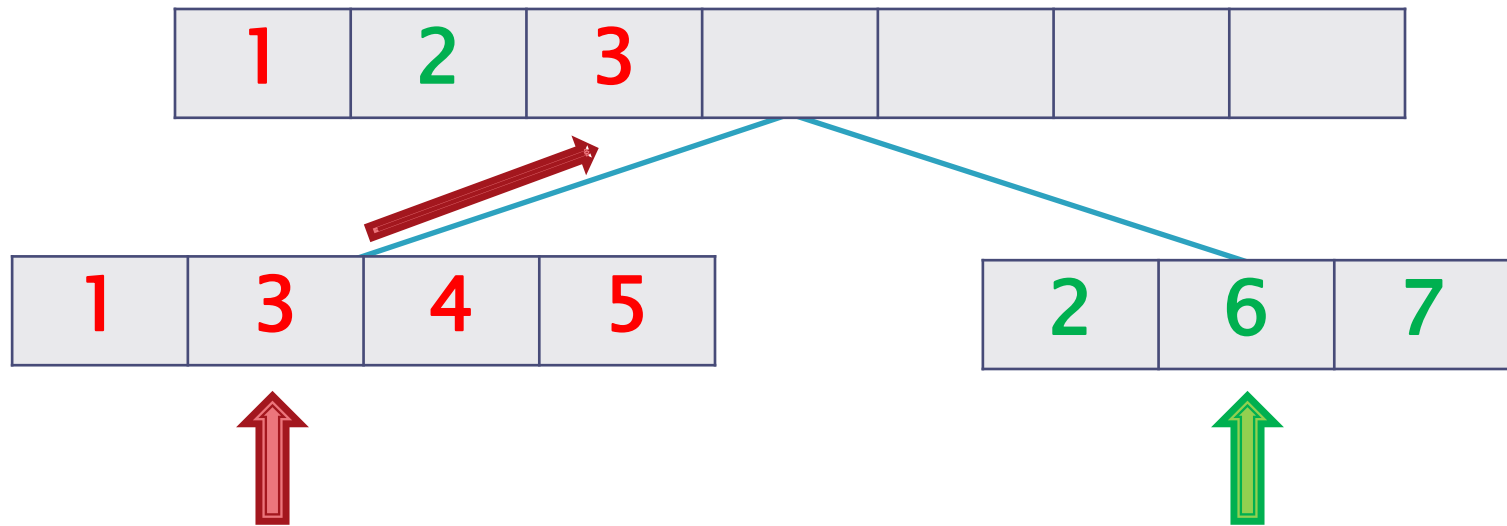
Confronto: $1 < 2? \rightarrow 1$

Impera (3.3)



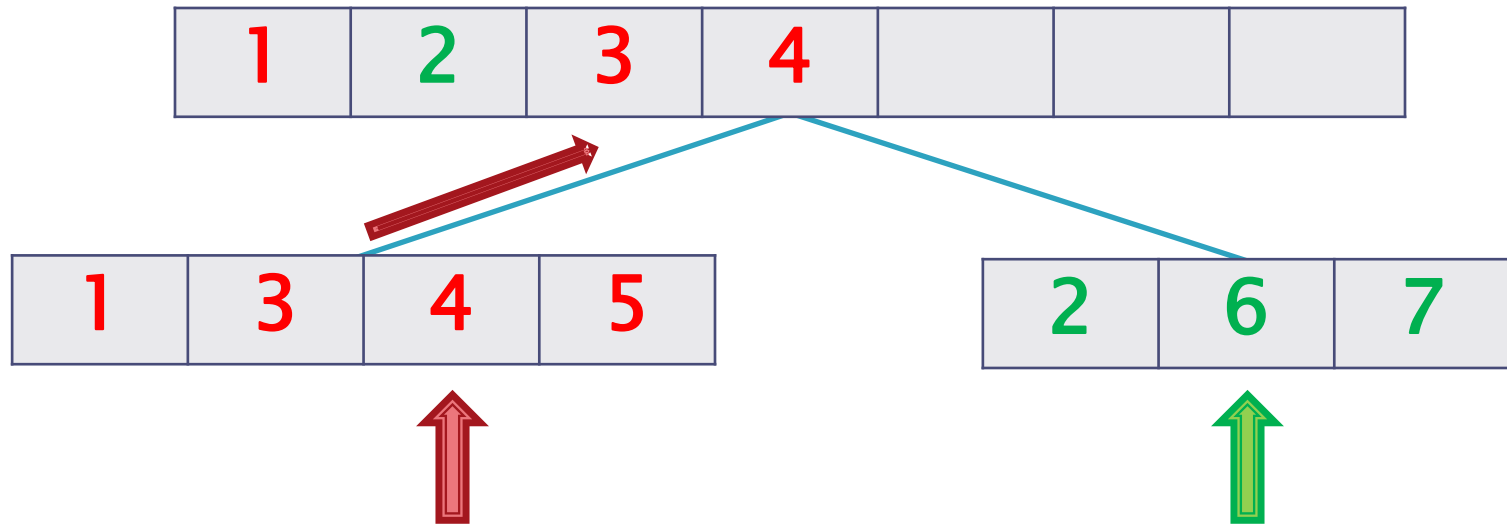
Confronto: $3 < 2? \rightarrow 2$

Impera (3.4)



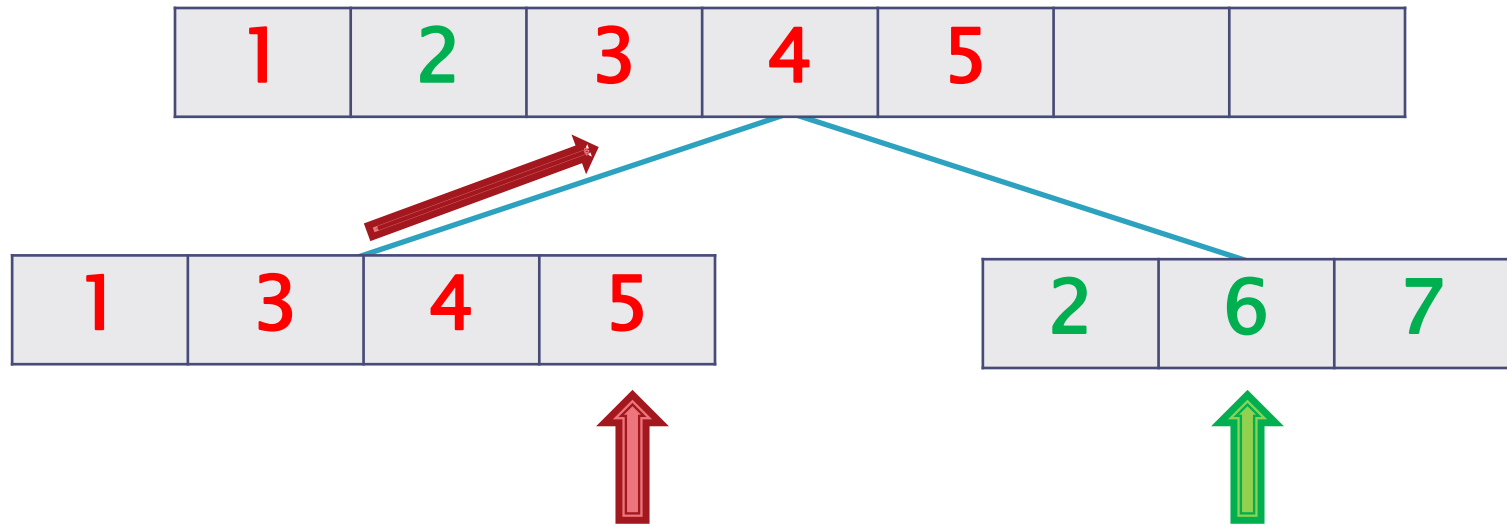
Confronto: $3 < 6? \rightarrow 3$

Impera (3.5)



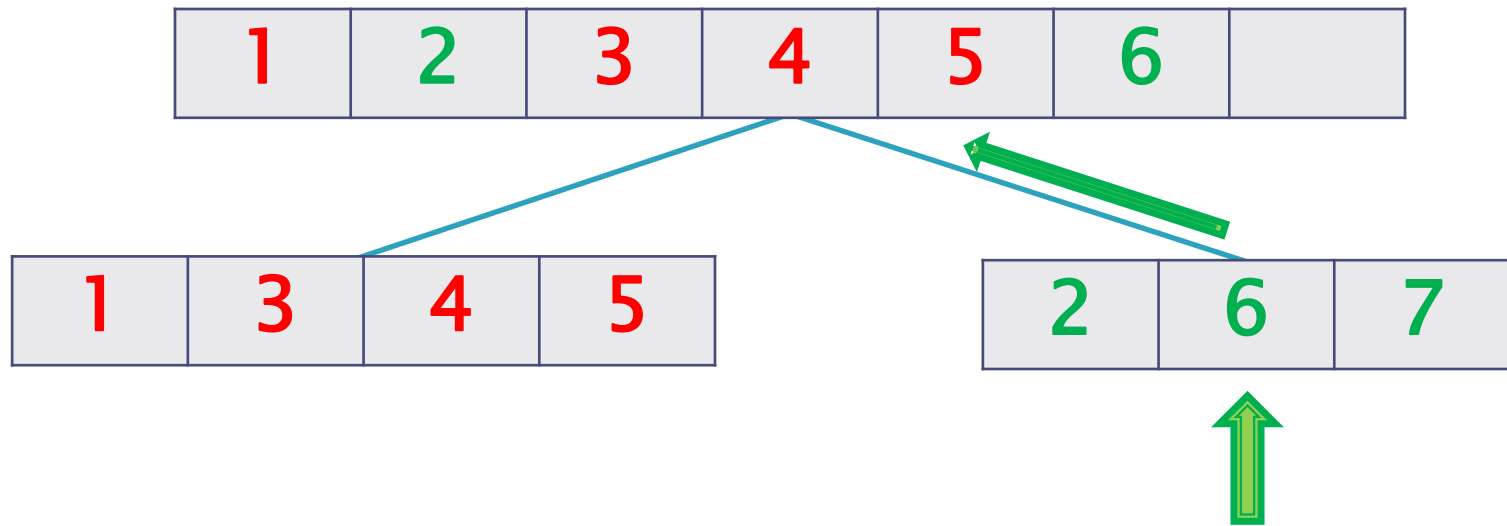
Confronto: $4 < 6? \rightarrow 4$

Impera (3.6)



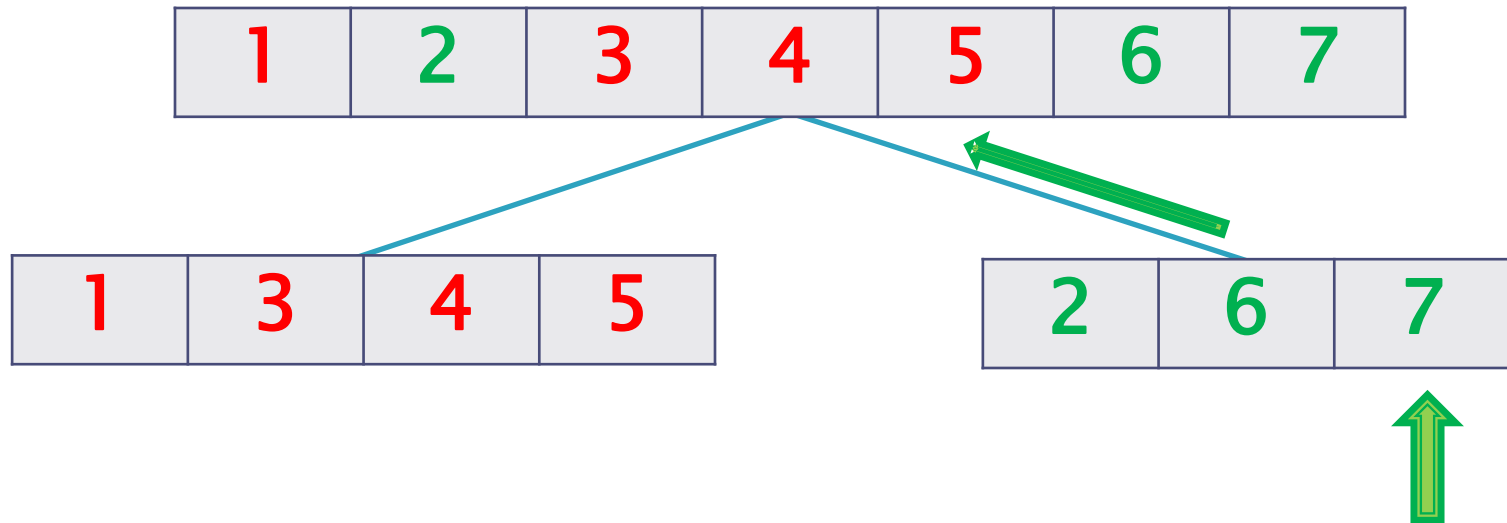
Confronto: $5 < 6? \rightarrow 5$

Impera (3.7)



Confronto: 6 → 6

Impera (3.8)



Confronto: 7 → 7

Impera (3.9)

1	2	3	4	5	6	7
---	---	---	---	---	---	---

- ▶ Le varie parti dell'array sono state unite
- ▶ L'array è ordinato

Altro esempio (in una slide)

