

ALBERI SPLAY

---

# ALGORITMI E STRUTTURE DATI

# ALBERI SPLAY

- ▶ Gli alberi binari di ricerca possono avere il problema che, a seconda dell'ordine di inserimento, possono avere profondità lineare
- ▶ Come primo esempio di albero che si modifica vediamo gli alberi splay (splay tree)
- ▶ Ideati nel 1985 da Sleator e Tarjan
- ▶ Ci serviranno per introdurre il concetto di rotazione per modificare la forma degli alberi

# ALBERI SPLAY

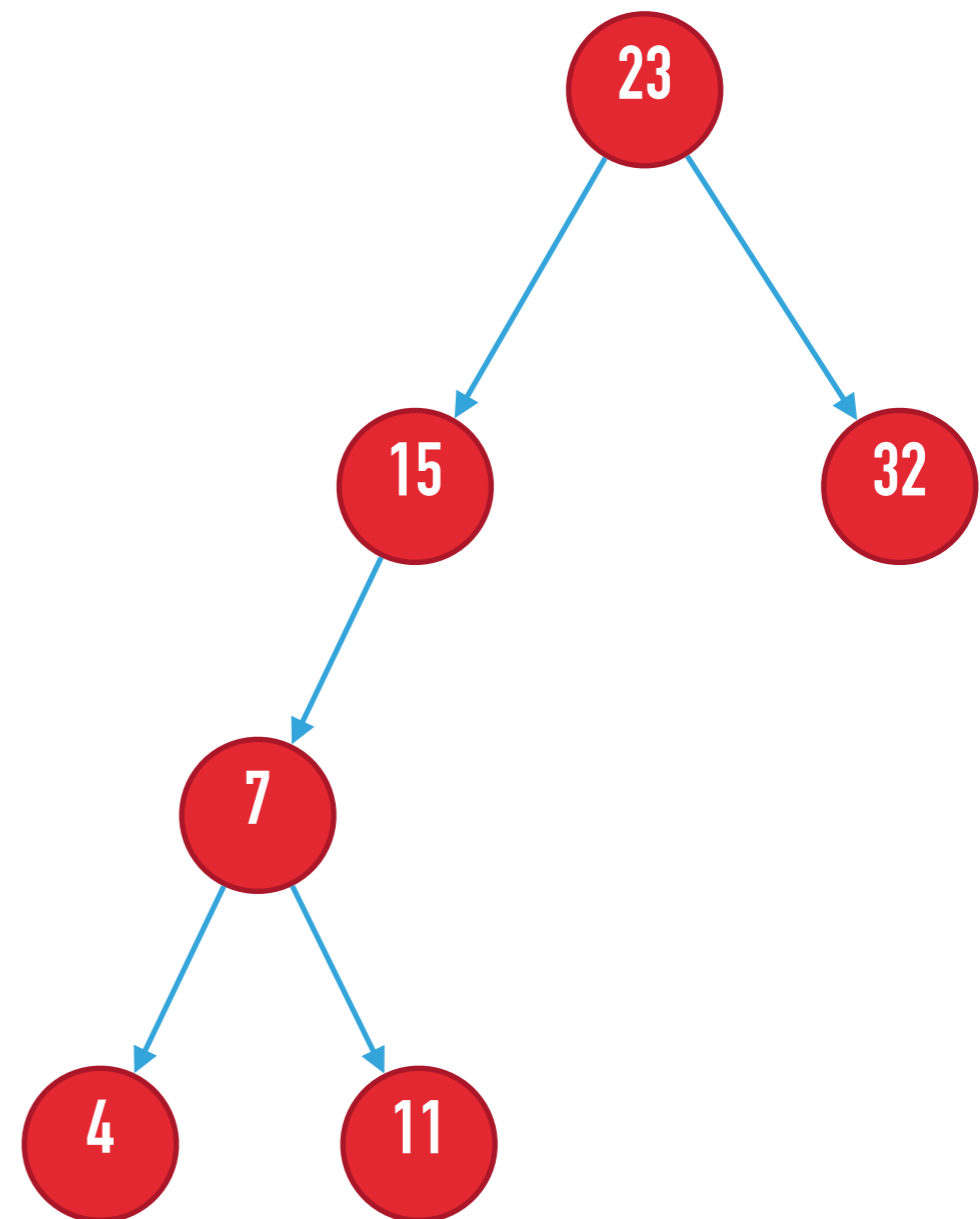
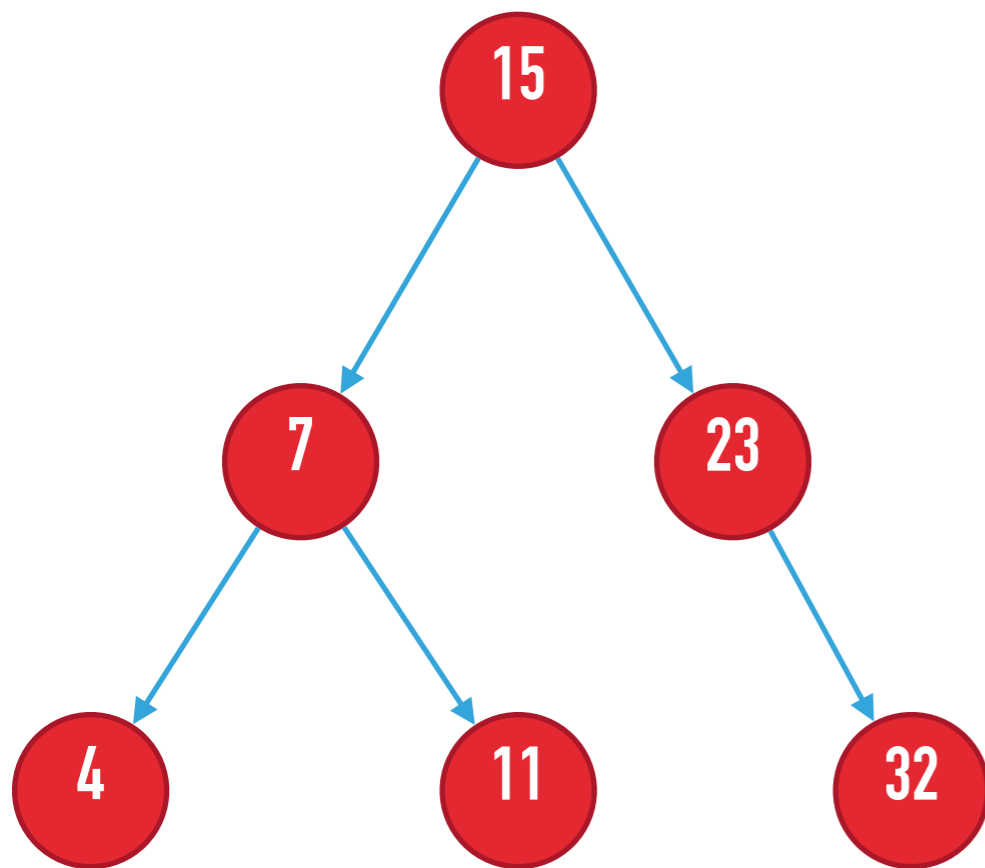
- ▶ L'idea di base è di spostare gli elementi che cerchiamo alla radice dell'albero
- ▶ L'effetto è che tutti gli elementi cercati di recente vengono a trovarsi vicino alla radice
- ▶ Questo non mantiene necessariamente l'albero bilanciato: il caso peggiore è ancora  $O(n)$
- ▶ Ci sono però una serie di altri vantaggi (e.g., costo ammortizzato di una sequenza di operazioni)

# ROTAZIONI

- ▶ Ruotare un nodo è l'operazione che ci permette di scambiare la posizione di un nodo e del suo figlio sinistro o destro
- ▶ Rotazione a sinistra di  $x$ : il nodo  $x$  viene sostituito dal suo figlio destro e ne diventa il figlio sinistro
- ▶ Rotazione a destra di  $x$ : il nodo  $x$  viene sostituito dal suo figlio sinistro e ne diventa il figlio destro

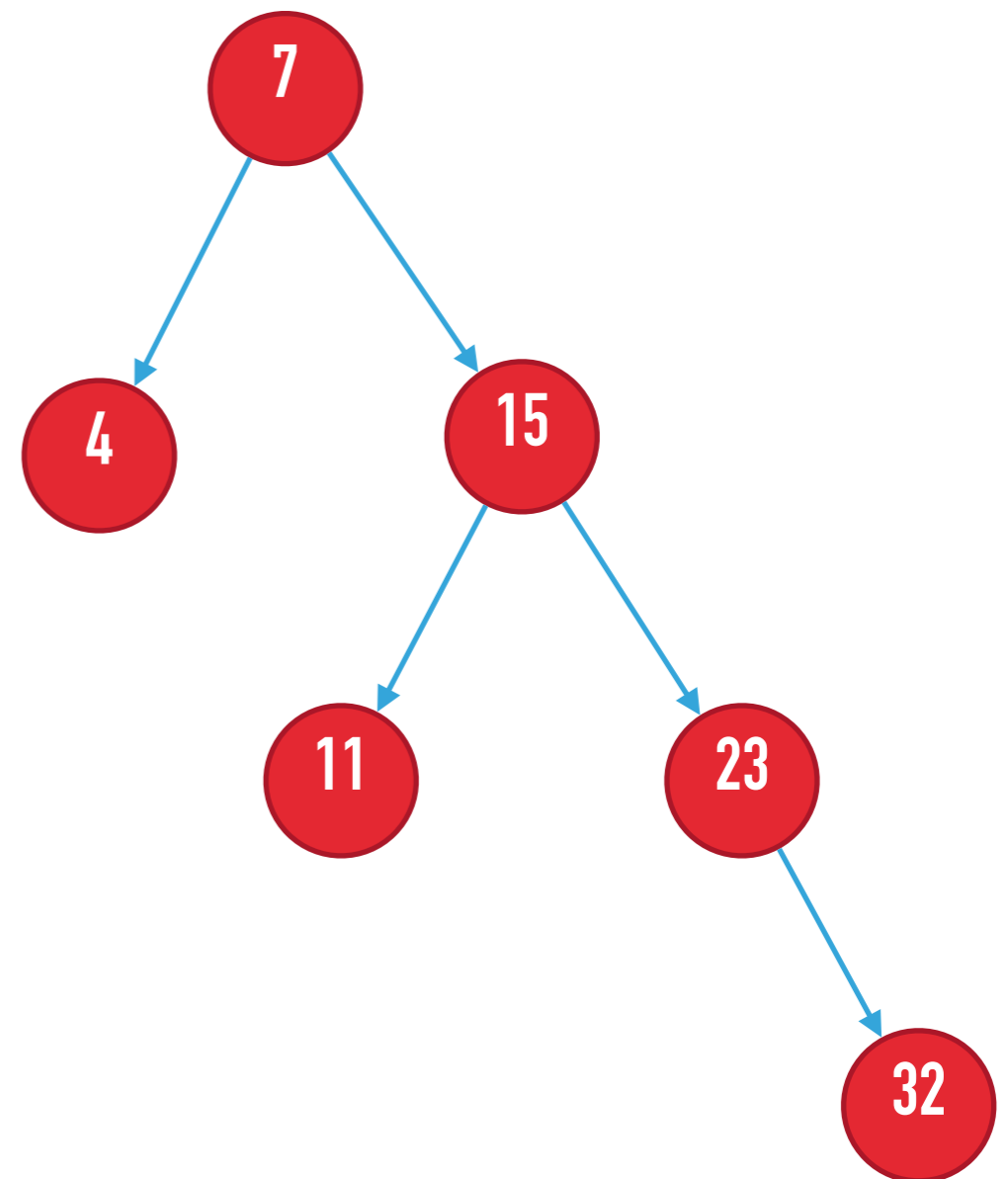
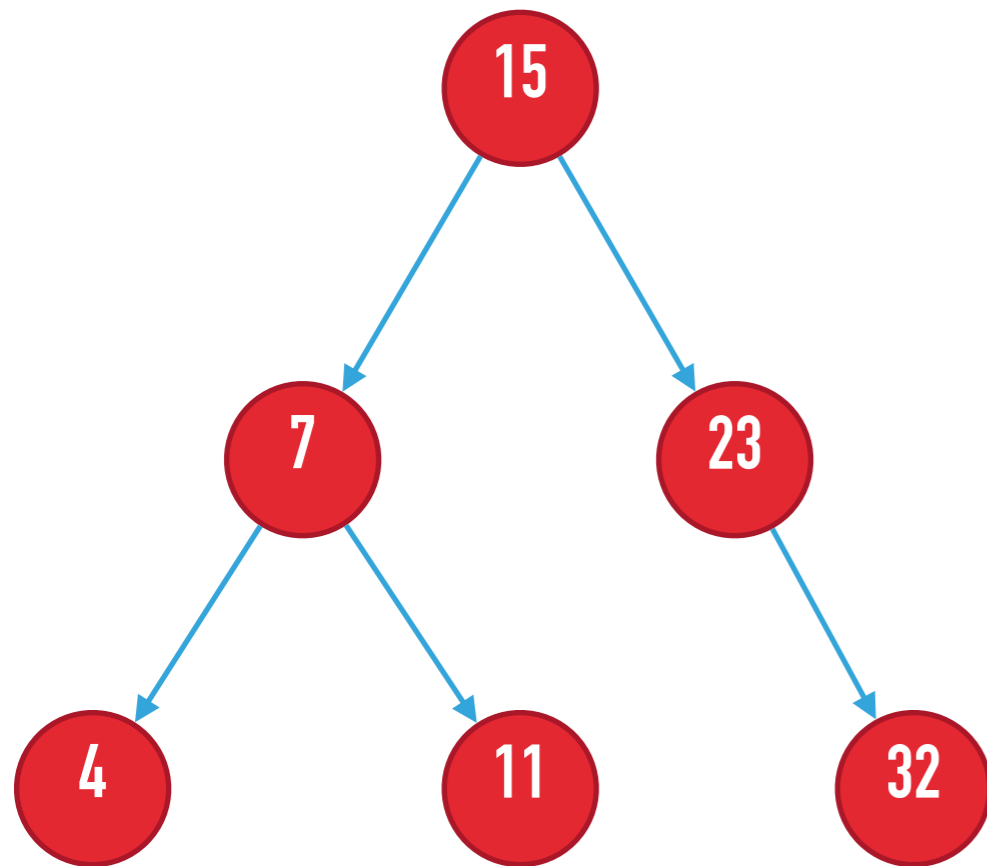
## ROTAZIONE A SINISTRA

Vogliamo ruotare a sinistra "15"



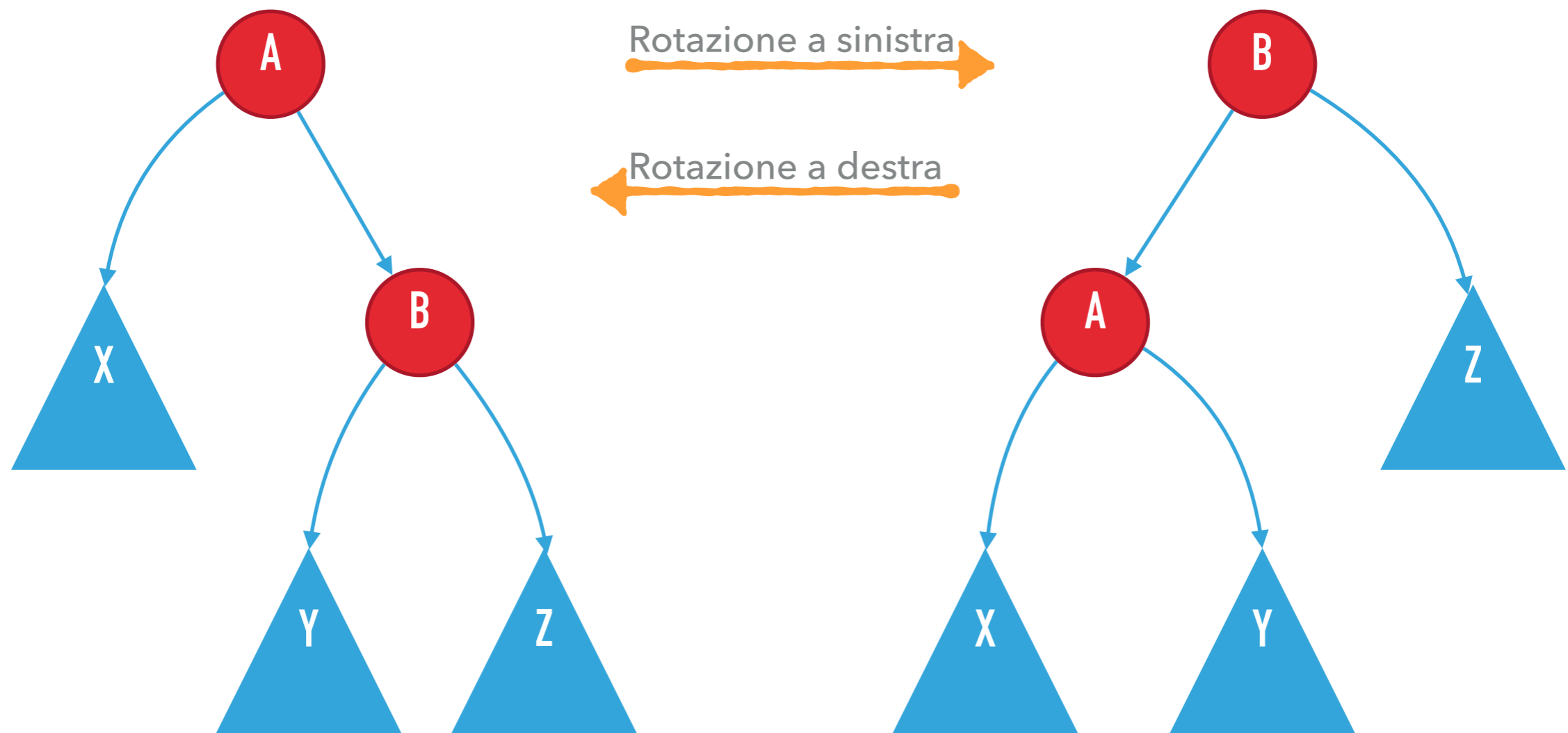
## ROTAZIONE A DESTRA

Vogliamo ruotare a destra "15"



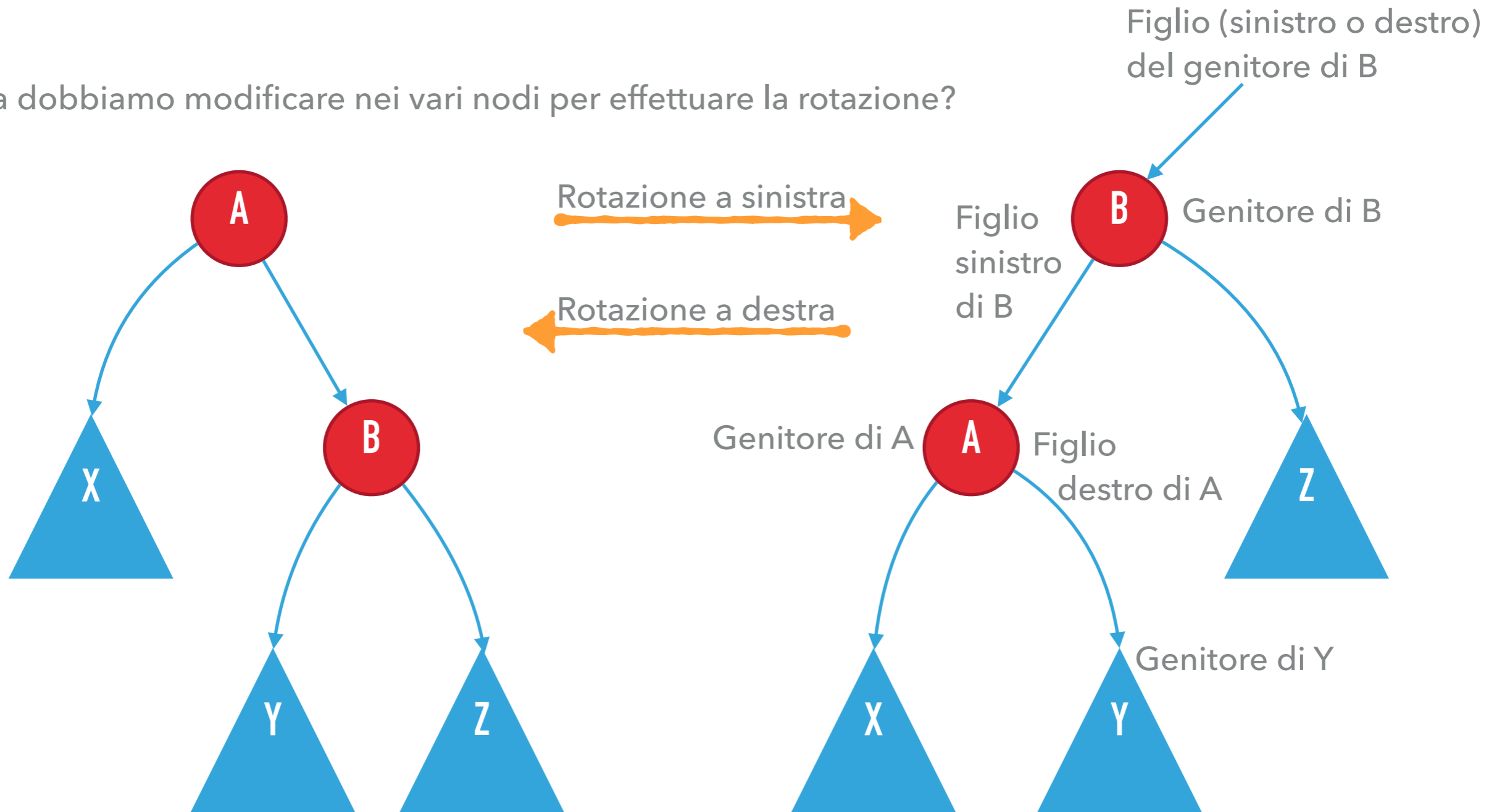
## ROTAZIONE

Astraiamo e vediamo come la rotazione agisce su due nodi generici



# ROTAZIONE: COSA BISOGNA MODIFICARE

Cosa dobbiamo modificare nei vari nodi per effettuare la rotazione?





# ROTAZIONI

- ▶ Parametri: nodo a
- ▶ `b = a.right # nodo che prenderà il posto di a`
- ▶ `a.right = b.left`
- ▶ `if a.right is not None:`
  - ▶ `a.right.parent = a`
- ▶ `b.left = a`
- ▶ `# con queste operazioni abbiamo messo il figlio sinistro di b come figlio destro di a, dobbiamo ancora sistemare i genitori di a e b`
- ▶ `if a.parent is not None: # sistemiamo il genitore di a`
  - ▶ `if a.parent.left is a: # nel caso a fosse figlio sinistro`
    - ▶ `a.parent.left = b`
  - ▶ `else # nel caso a fosse figlio destro`
    - ▶ `a.parent.right = b`
- ▶ `b.parent = a.parent`
- ▶ `a.parent = b`

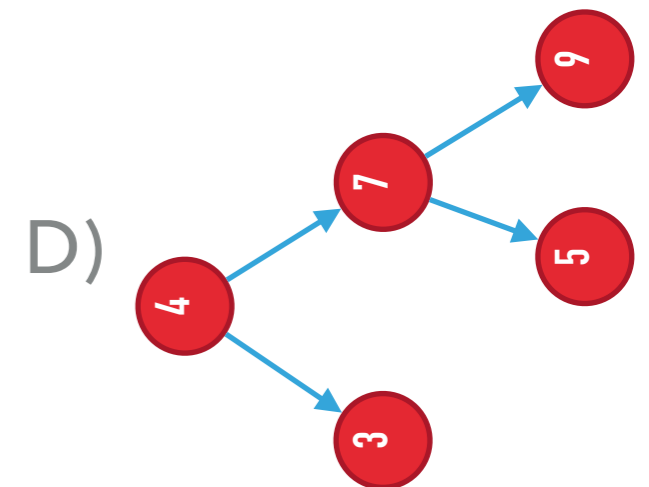
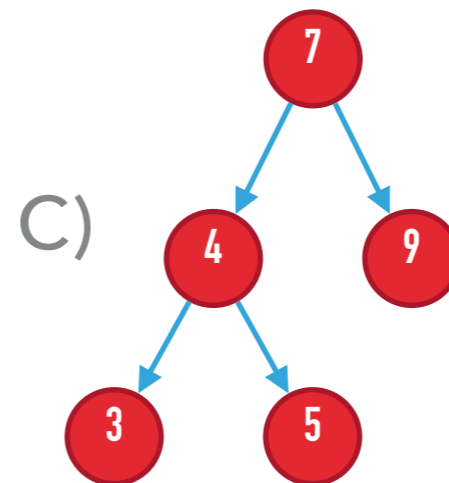
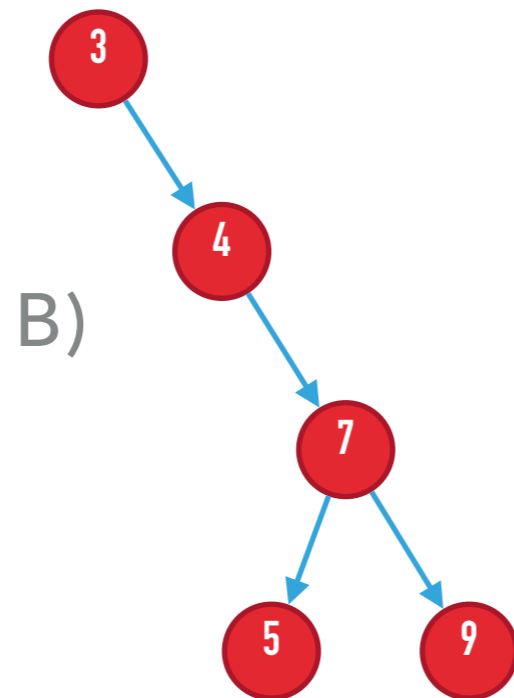
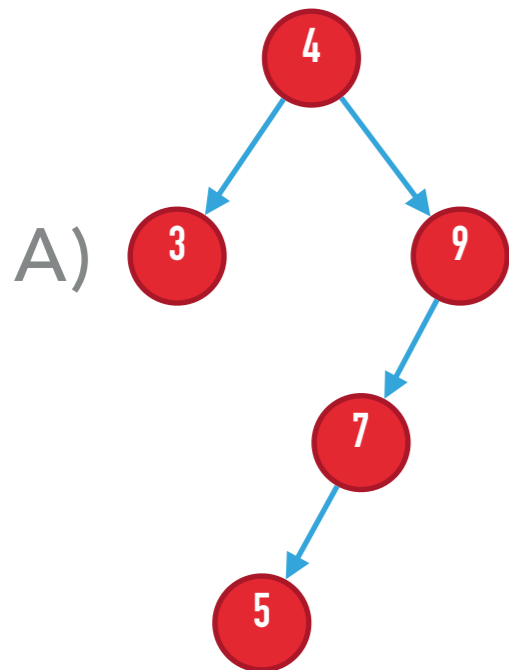
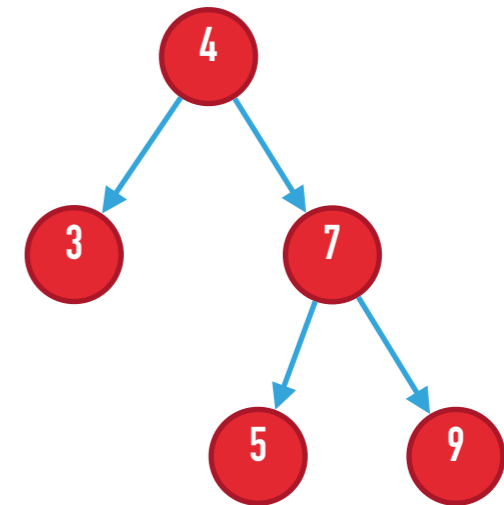
# ROTAZIONI

- ▶ Ogni rotazione richiede tempo costante
- ▶ Ci permette di "muovere" un nodo lungo l'albero facendolo salire o scendere tramite una serie di rotazioni
- ▶ Negli alberi Splay useremo le rotazioni per far risalire un nodo fino a farlo diventare la radice
- ▶ In altri alberi che garantiscono il bilanciamento (come gli alberi AVL) le rotazioni sono usate per bilanciare dopo ogni inserimento o rimozione

## QUIZ: ROTAZIONI

Supponiamo di avere il seguente albero binario:

Quale è il risultato di ruotare a sinistra "4"?

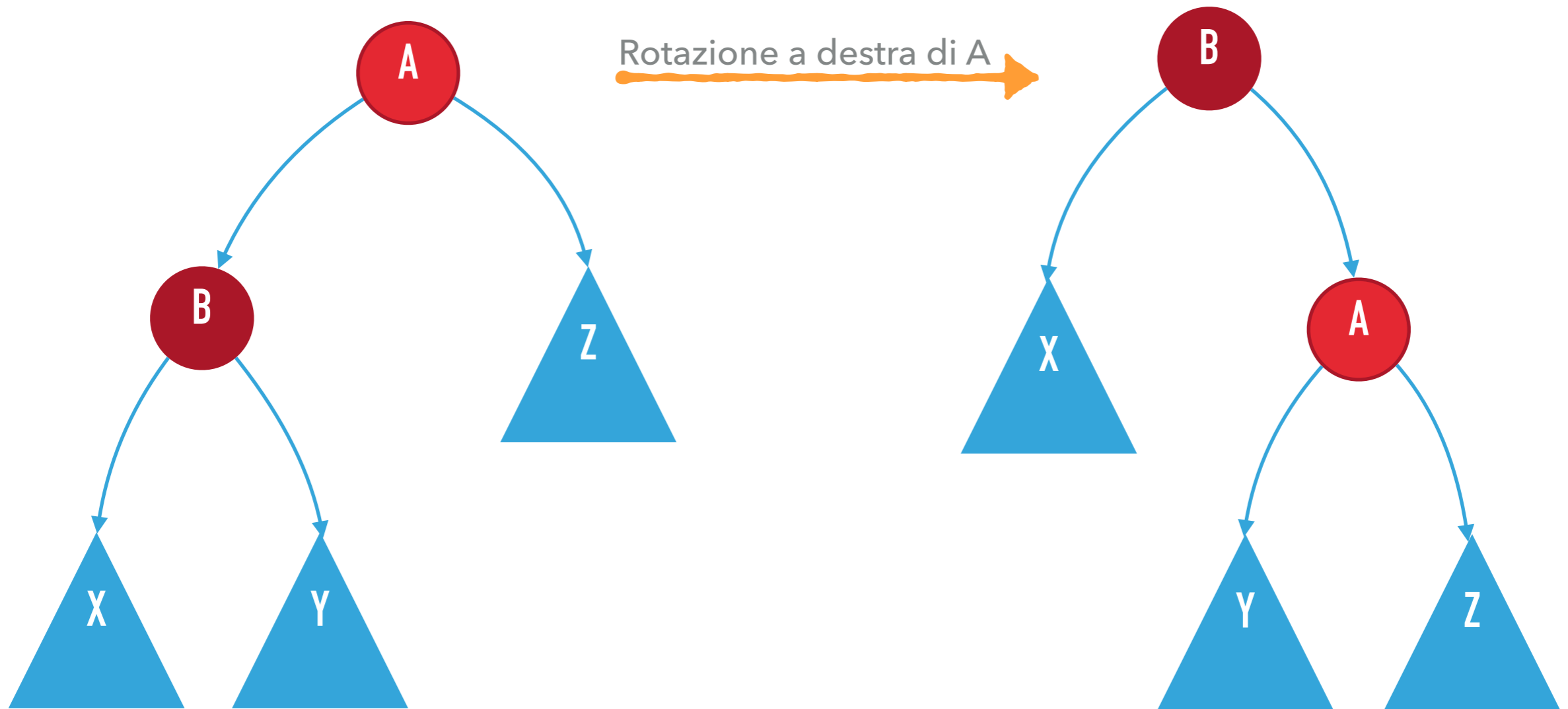


# ALBERI SPLAY

- ▶ Ogni volta che cerchiamo un elemento nell'albero chiamiamo l'operazione di "splay" o "muovi alla radice" che sposta l'elemento cercato alla radice dell'albero
- ▶ Questo viene fatto con rotazioni secondo tre casi:
  - ▶ Zig.
  - ▶ Zig zig.
  - ▶ Zig zag.

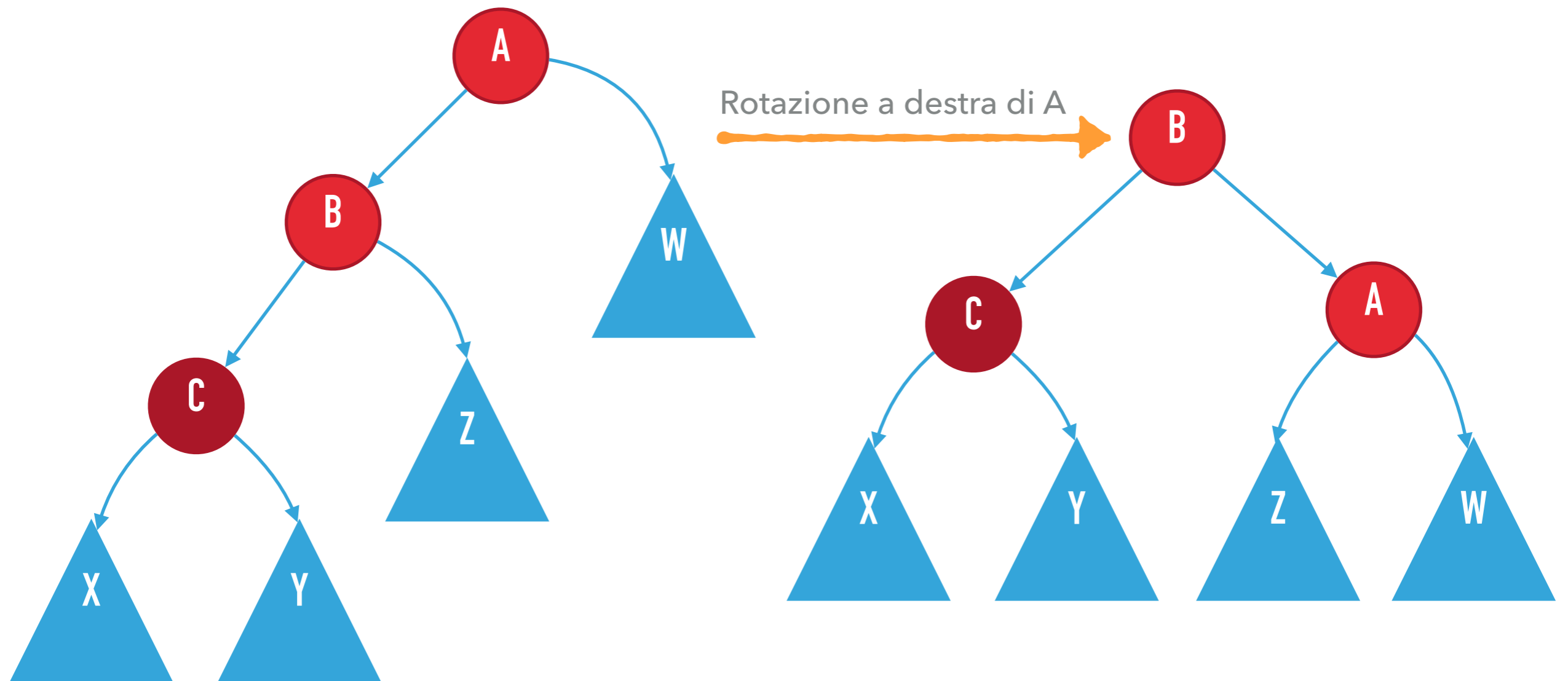
## CASO ZIG

Il nodo da muovere è figlio sinistro della radice



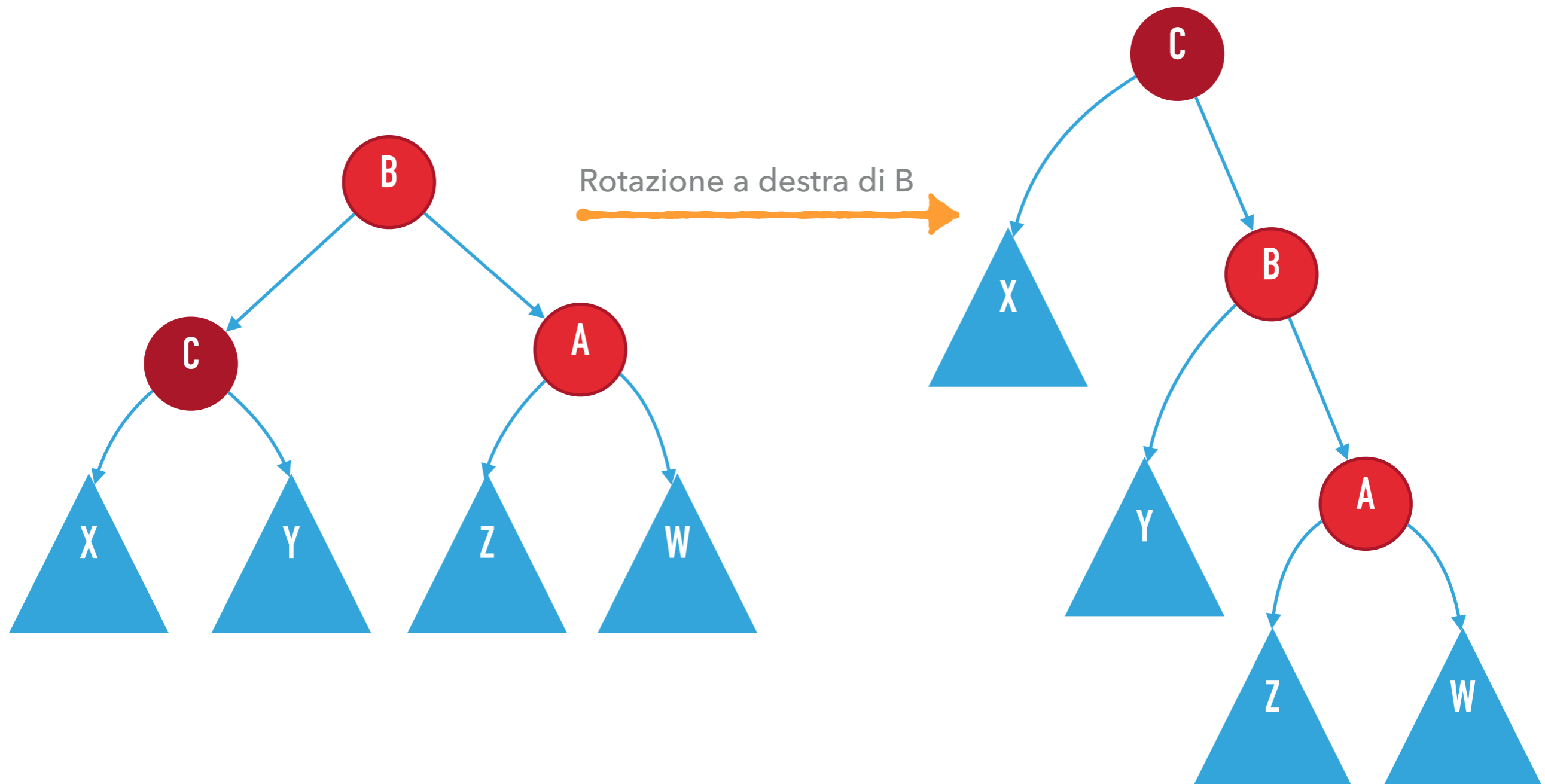
## CASO ZIG ZIG

Il nodo da muovere è figlio sinistro di un nodo che è a sua volta figlio sinistro



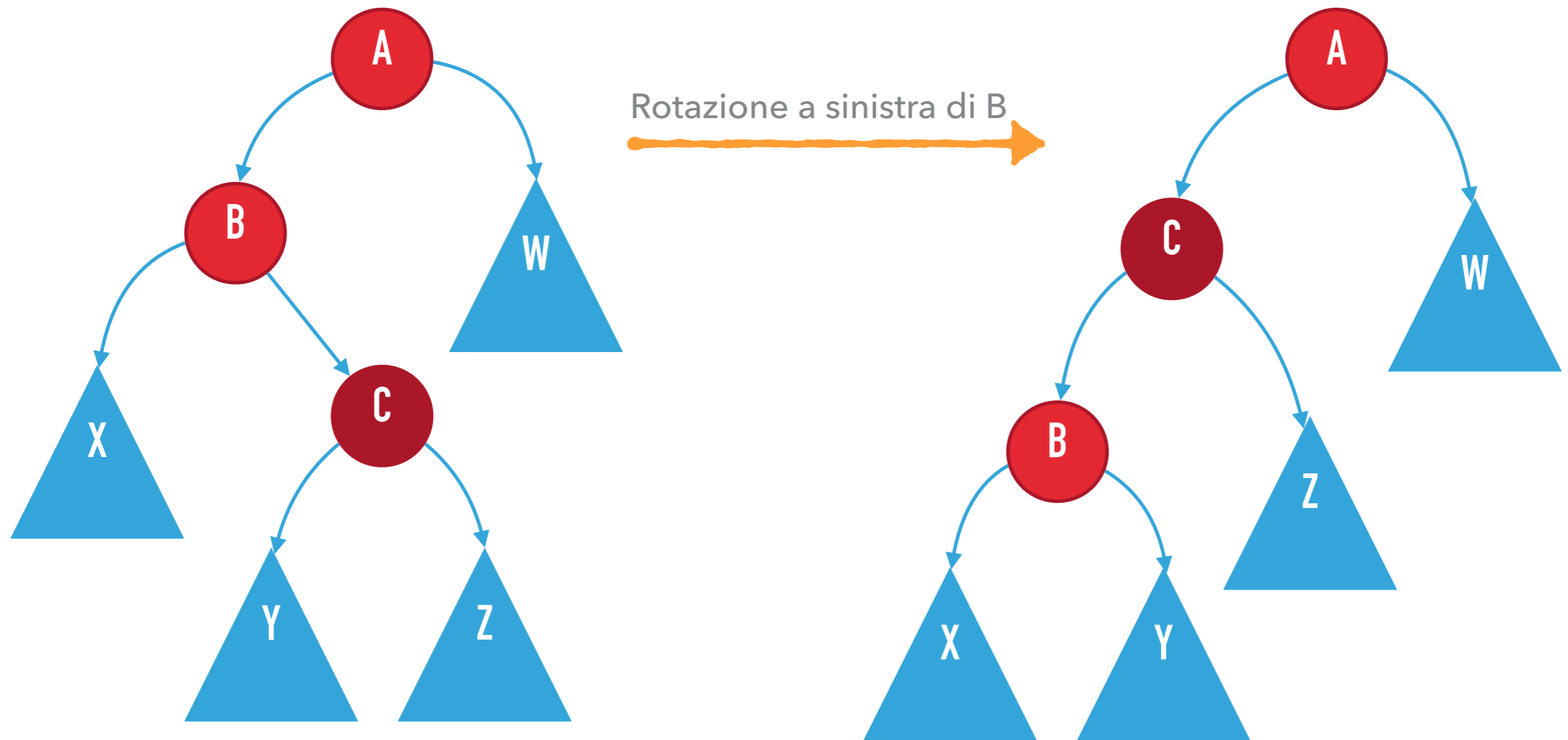
## CASO ZIG ZIG

Il nodo da muovere è figlio sinistro di un nodo che è a sua volta figlio sinistro



## CASO ZIG ZAG

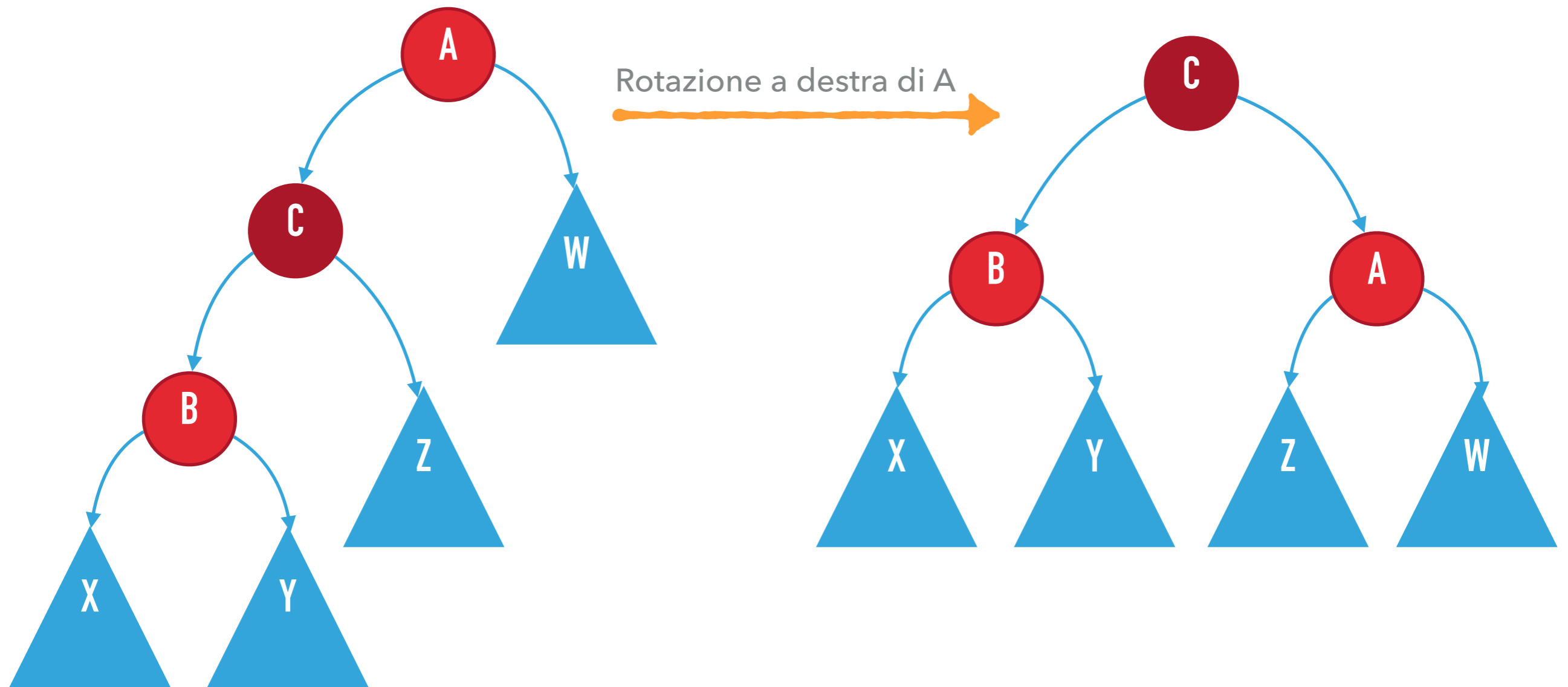
Il nodo da muovere è figlio destro di un nodo che è figlio sinistro



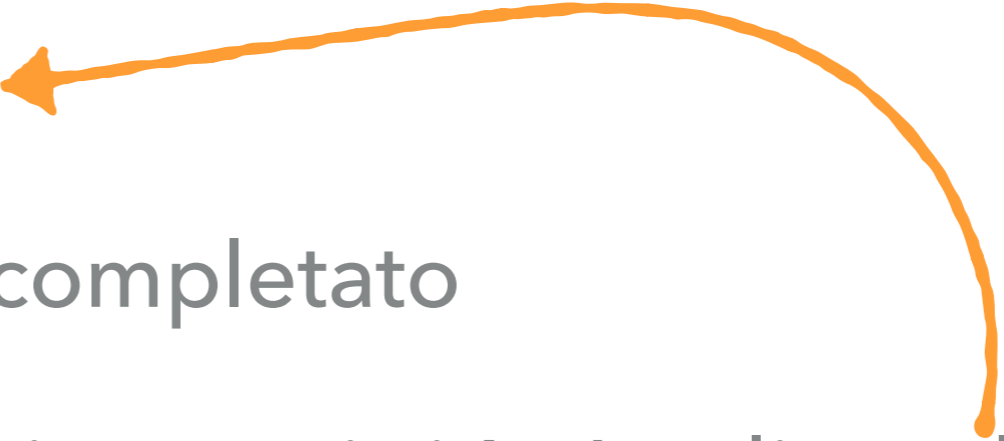


## CASO ZIG ZAG

Il nodo da muovere è figlio destro di un nodo che è figlio sinistro



# ALBERI SPLAY

- ▶ Applicando uno tra questi tre casi (o i loro simmetrici) ad ogni passo possiamo spostare un nodo fino alla radice
- ▶ La struttura di base è:
  - ▶ Il nodo è alla radice? 
  - ▶ Sì: allora abbiamo completato
  - ▶ No: vedi in quali dei tre casi si è. Applicare le rotazioni.

# ALBERI SPLAY

- ▶ Non vedremo una analisi accurata del tempo di calcolo delle operazioni di ricerca
- ▶ Esiste però il seguente teorema (**Balance Theorem**)
- ▶ Data una sequenza di  $m$  operazioni di ricerca in un albero splay di  $n$  elementi, il costo di effettuare la sequenza di operazioni è  $O(m \log n + n \log n)$
- ▶ Questo significa che se facciamo almeno  $n$  operazioni il costo medio di ogni operazione è logaritmico