ALBERI AVL: INSERIMENTO E CANCELLAZIONE
ALBERI ROSSO NERI
DYNAMIC SELECT

ALGORITMI E STRUTTURE DATI

INSERIMENTO E CANCELLAZIONE

INSERIMENTO E CANCELLAZIONE

- L’inserimento può violare la proprietà AVL su più nodi nel percorso che va dalla radice al punto dove è stato inserito il nuovo nodo

INSERIMENTO E CANCELLAZIONE

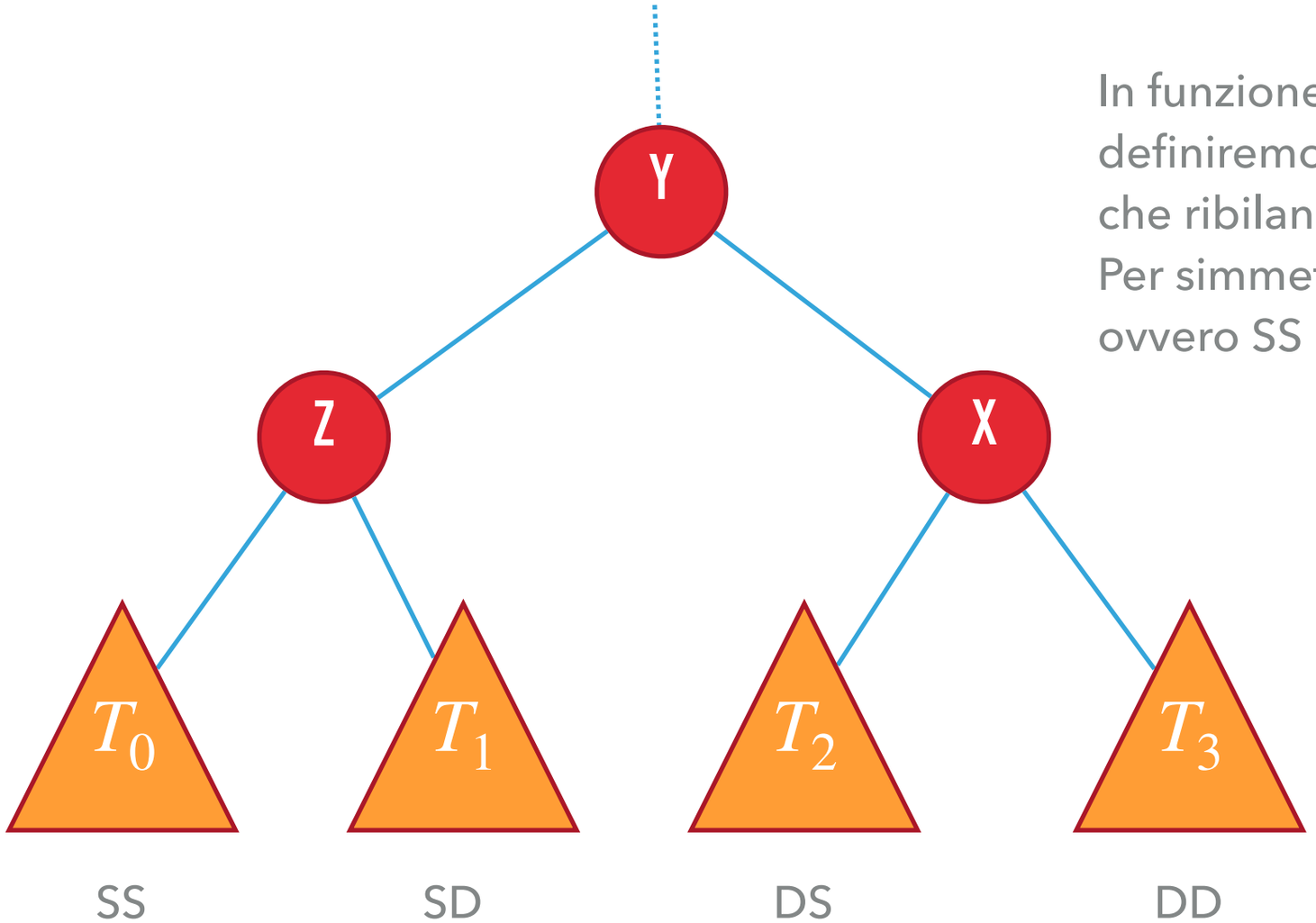
- L'inserimento può violare la proprietà AVL su più nodi nel percorso che va dalla radice al punto dove è stato inserito il nuovo nodo
- La cancellazione analogamente può violare la proprietà AVL nei nodi antenati del nodo cancellato

INSERIMENTO E CANCELLAZIONE

- L'inserimento può violare la proprietà AVL su più nodi nel percorso che va dalla radice al punto dove è stato inserito il nuovo nodo
- La cancellazione analogamente può violare la proprietà AVL nei nodi antenati del nodo cancellato
- Vediamo come ogni sbilanciamento può essere risolto tramite una sequenza di rotazioni

RIBILANCIAMENTO

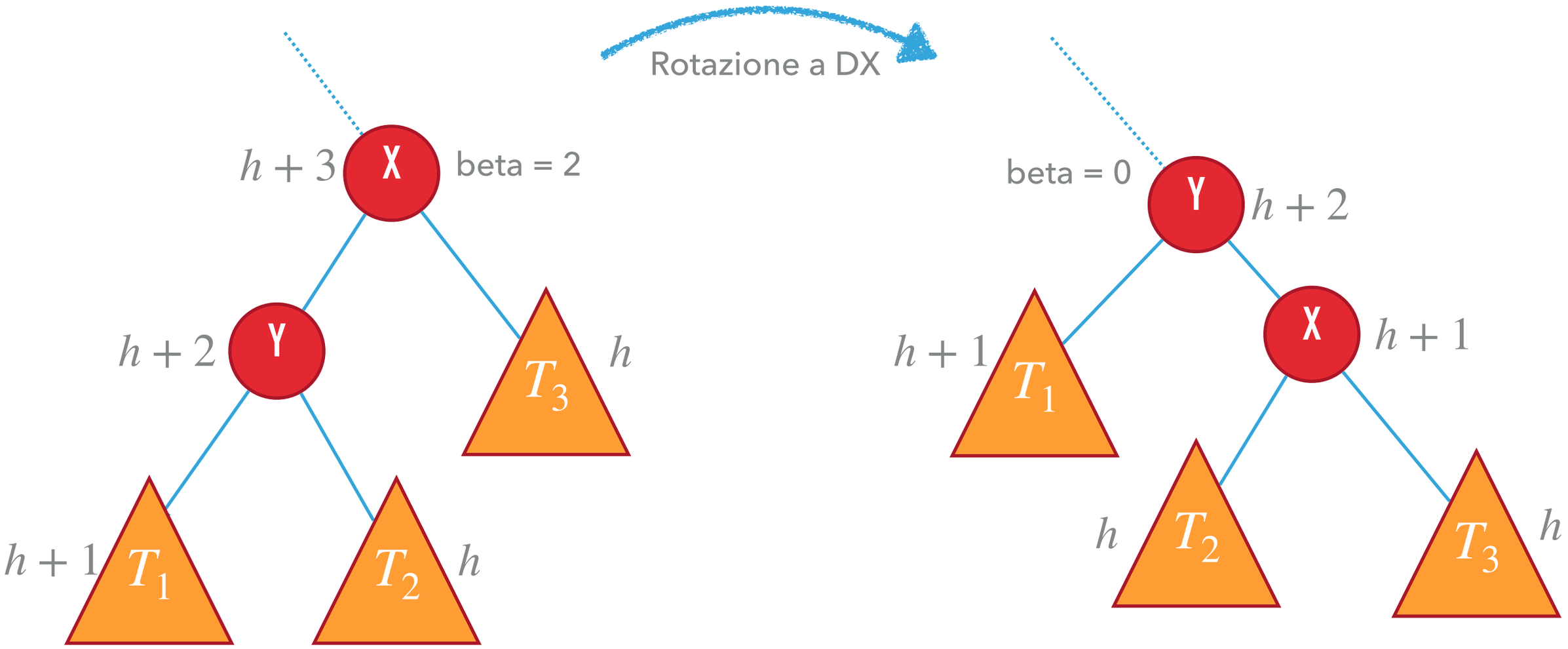
Se emerge uno sbilanciamento dopo un inserimento o una cancellazione di un nodo, allora il fattore di sbilanciamento sarà 2 o -2, e causato da un nodo in più (o in meno) in uno dei 4 sottoalberi T0 T1 T2 T3.



In funzione di dove emerge lo sbilanciamento, definiremo una sequenza di rotazioni opportuna che ribilanci l'albero.
Per simmetria, tratteremo solo due dei 4 casi, ovvero SS e SD.

RIBILANCIAMENTO – CASO SS

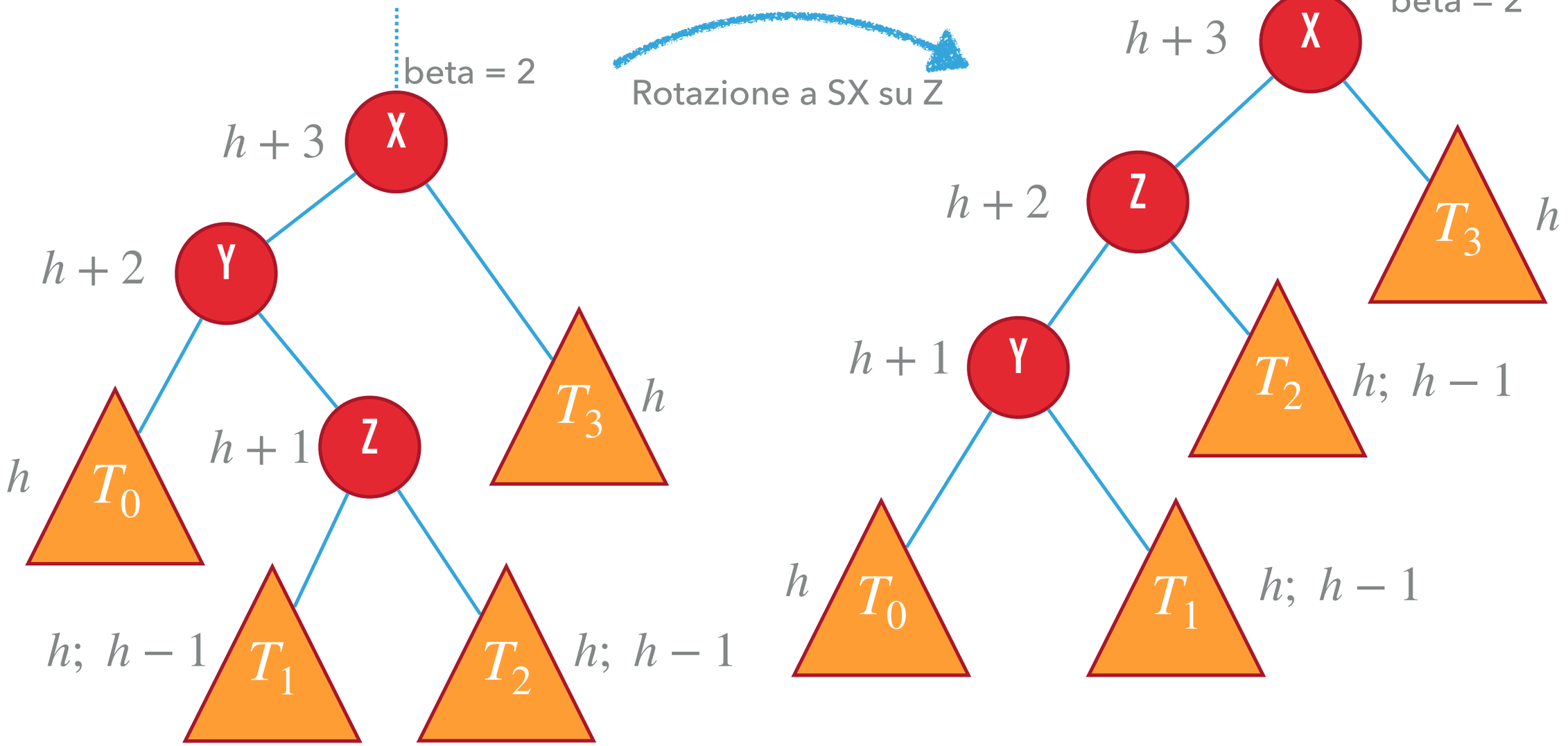
Supponiamo ci sia uno sbilanciamento su X (pari a 2) per la differenza tra le altezze di T1 e T3.



L'altezza del sottoalbero precedentemente radicato in X decresce di uno

RIBILANCIAMENTO – CASO SD

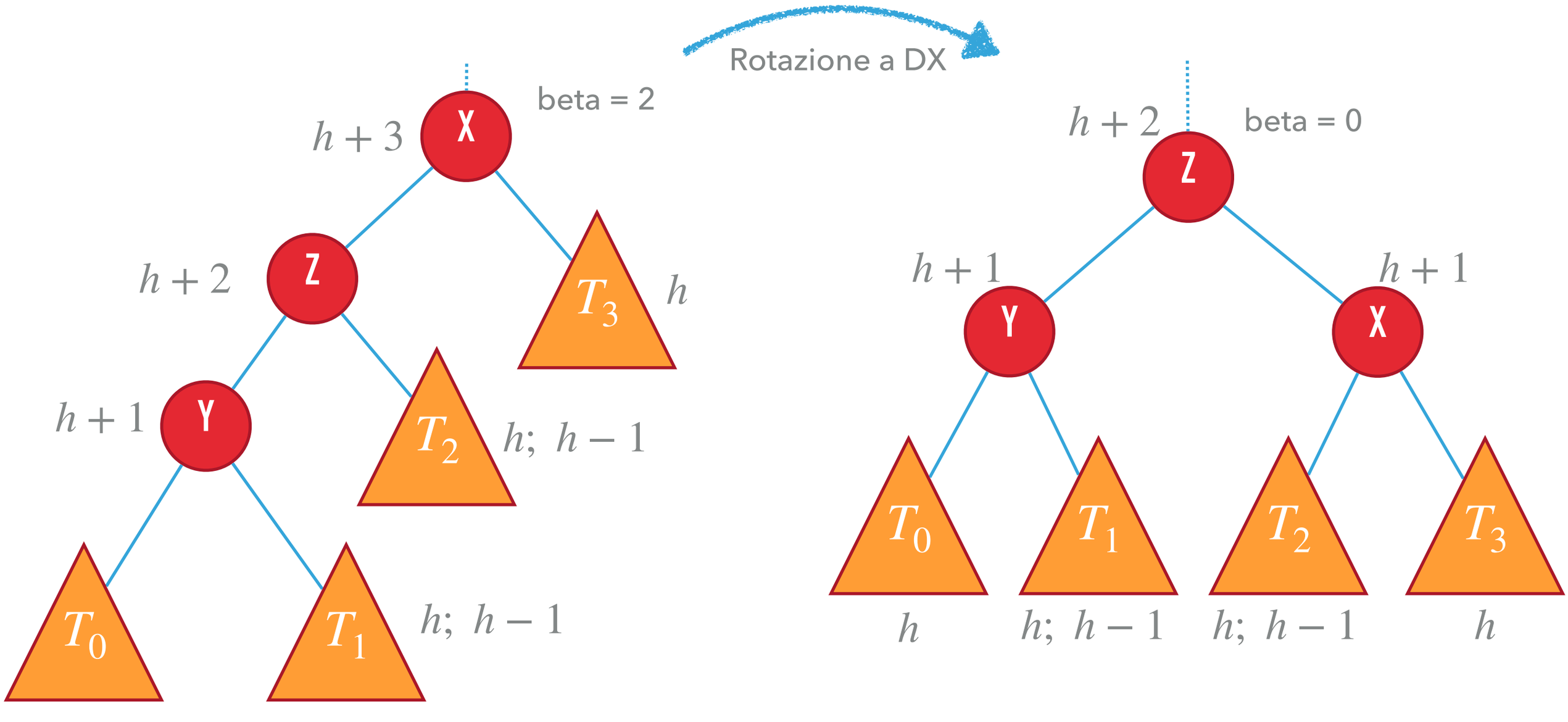
Supponiamo ci sia uno sbilanciamento su X (pari a 2).



Solo uno tra T1 e T2 ha altezza h

La prima rotazione non risolve lo sbilanciamento

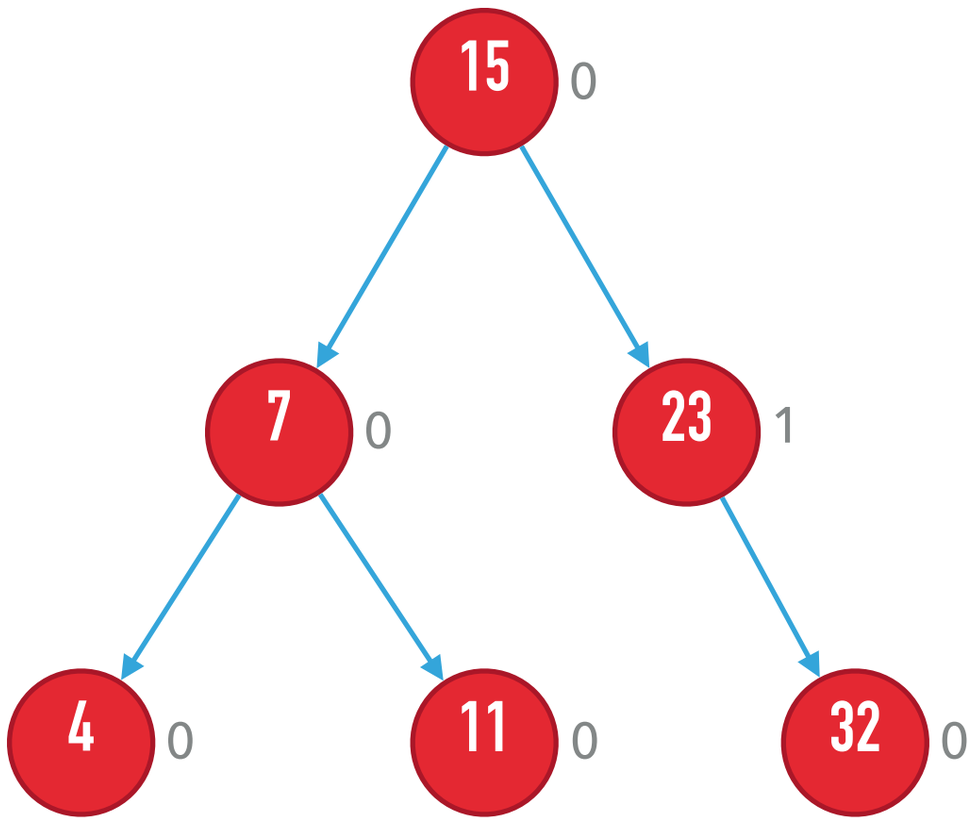
RIBILANCIAMENTO – CASO SD



L'altezza del sottoalbero precedentemente radicato in X decresce di uno

INSERIMENTO

Inseriamo il nodo “19” 19

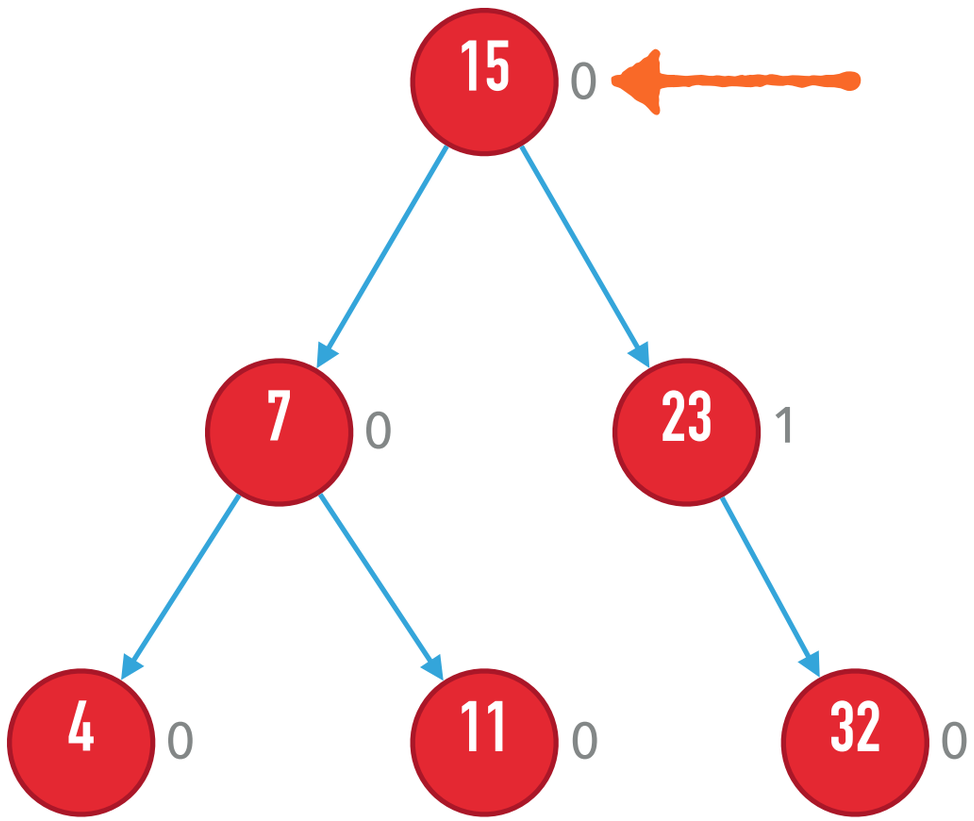


INSERIMENTO

Inseriamo il nodo “19” 19

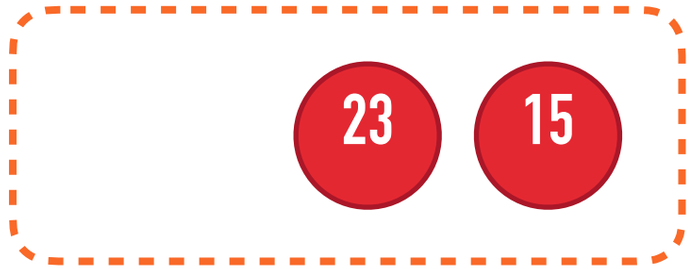
15

Stack dei nodi visitati

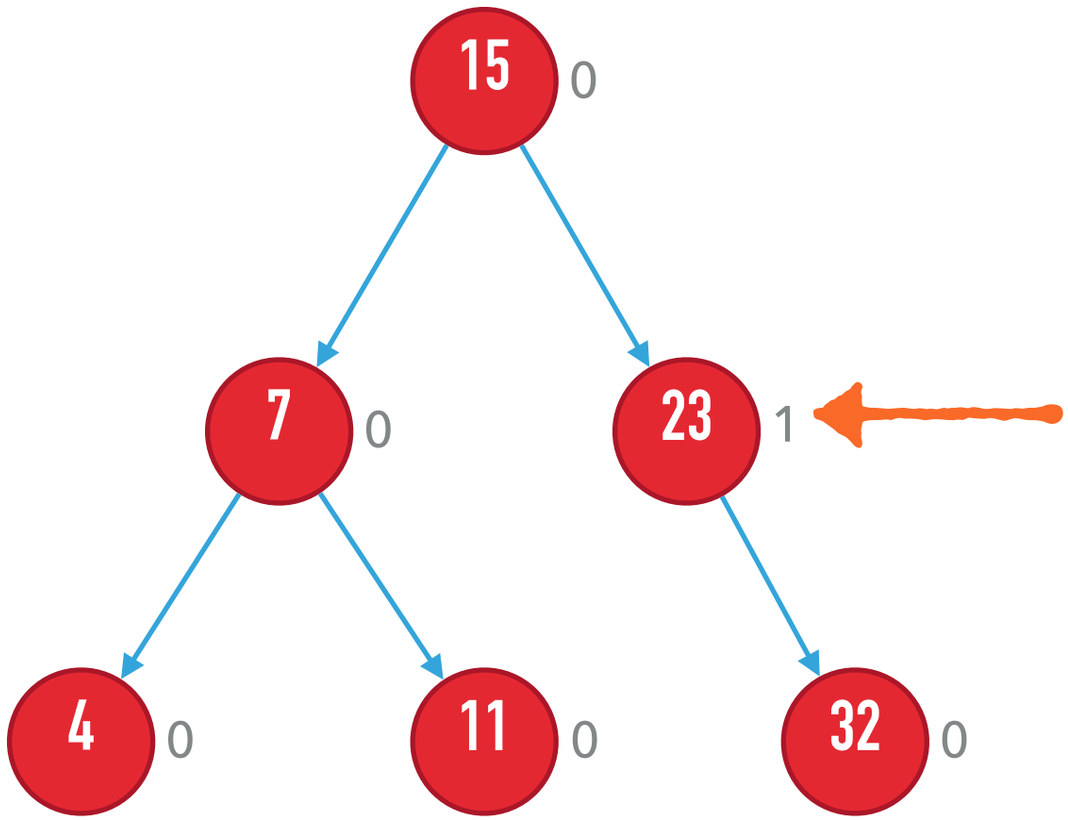


INSERIMENTO

Inseriamo il nodo “19”

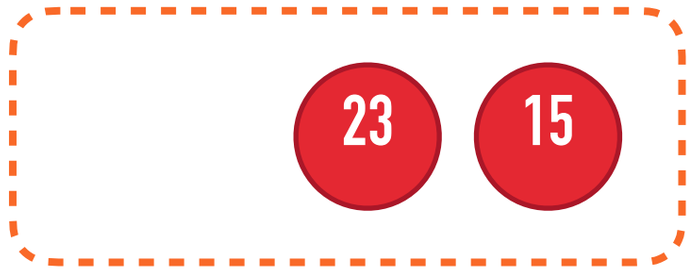


Stack dei nodi visitati

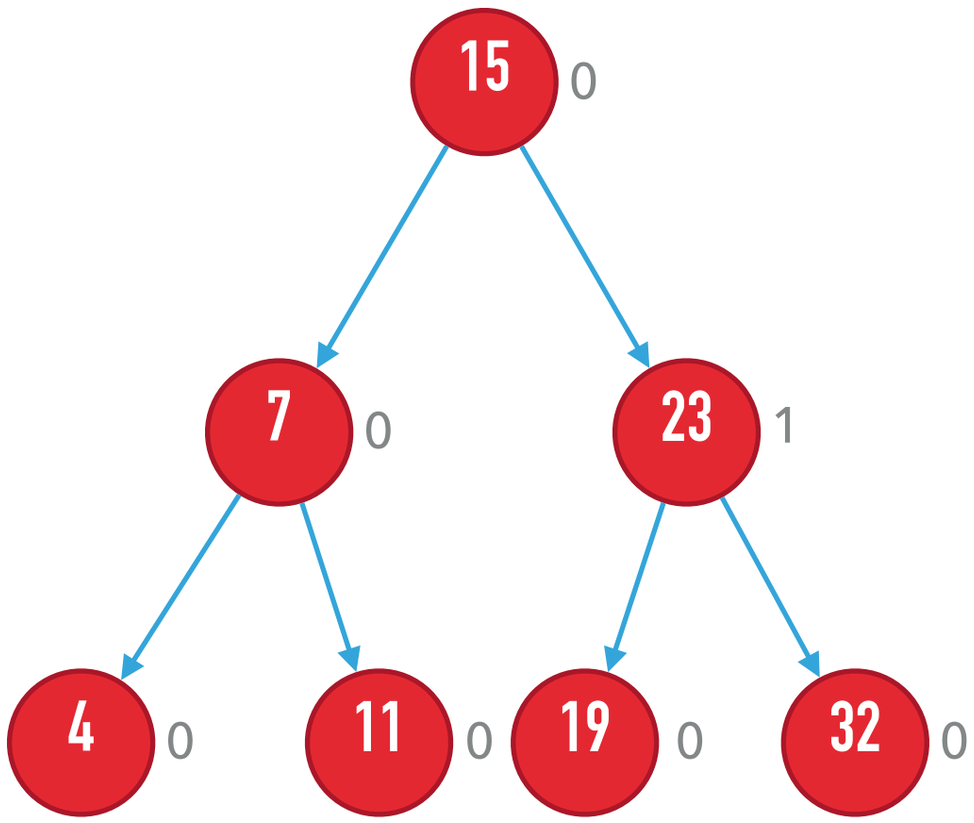


INSERIMENTO

Inseriamo il nodo “19”



Stack dei nodi visitati

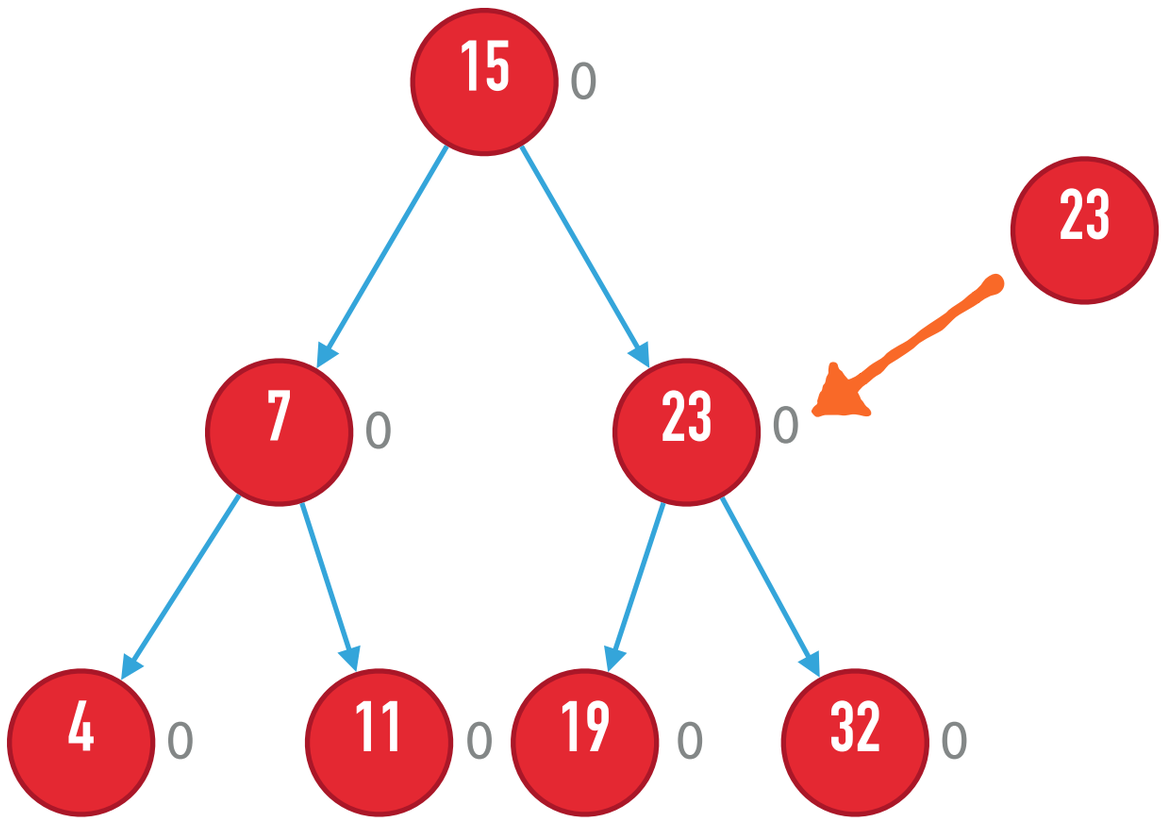


Abbiamo inserito il nodo, ora svuotiamo lo stack e aggiorniamo il bilanciamento

INSERIMENTO

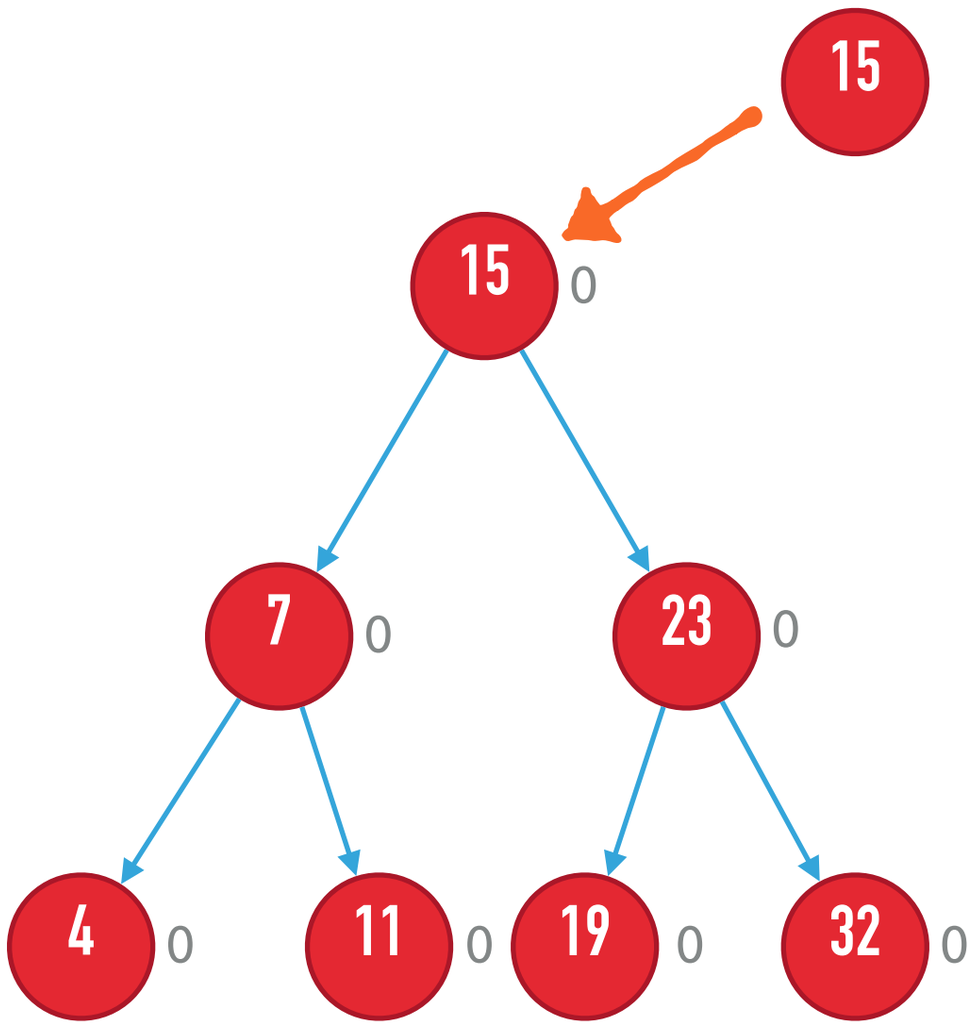
Inseriamo il nodo “19”

Stack dei nodi visitati



INSERIMENTO

Inseriamo il nodo “19”



Stack dei nodi visitati

INSERIMENTO

INSERIMENTO

- Il caso migliore è quello in cui non otteniamo nessun valore $+2$ o -2 e aggiorniamo solamente il bilanciamento

INSERIMENTO

- Il caso migliore è quello in cui non otteniamo nessun valore $+2$ o -2 e aggiorniamo solamente il bilanciamento
- Notate come se c'è uno sbilanciamento, esso deve essere con un valore ± 2, dato che l'aggiunta di un nodo modifica l'altezza di al più 1.

INSERIMENTO

- Il caso migliore è quello in cui non otteniamo nessun valore $+2$ o -2 e aggiorniamo solamente il bilanciamento
- Notate come se c'è uno sbilanciamento, esso deve essere con un valore ± 2, dato che l'aggiunta di un nodo modifica l'altezza di al più 1.
- Nel caso vi sia sbilanciamento cerchiamo il **primo** nodo – risalendo dal nodo appena inserito – che è sbilanciato - e applichiamo uno degli schemi di rotazione visti prima.

INSERIMENTO

INSERIMENTO

- Ma come facciamo a provare che queste operazioni rendono un albero bilanciato?

INSERIMENTO

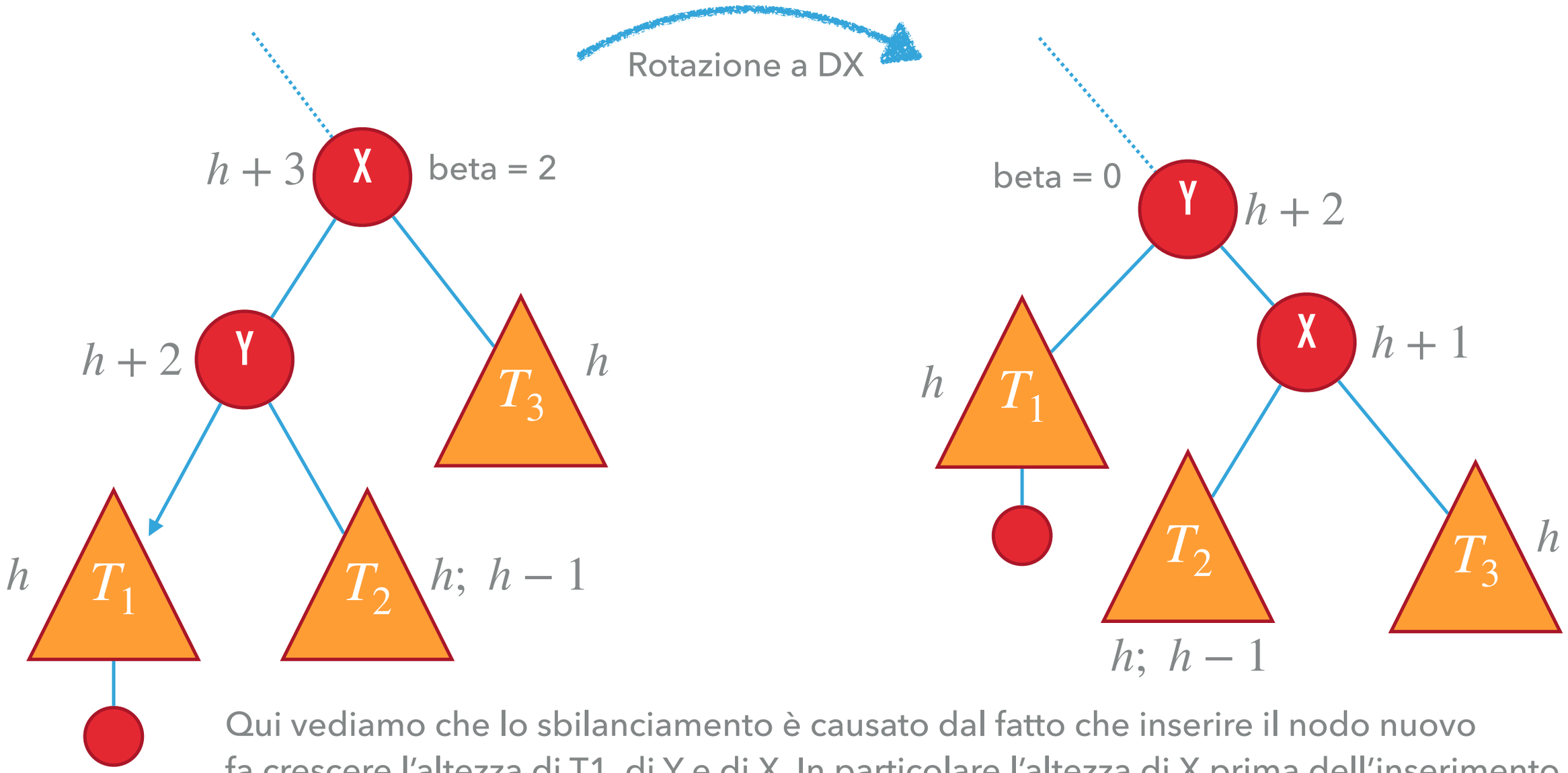
- Ma come facciamo a provare che queste operazioni rendono un albero bilanciato?
- Non è direttamente intuitivo, lo proviamo per un caso (sinistra-sinistra), per gli altri casi è equivalente

INSERIMENTO

- Ma come facciamo a provare che queste operazioni rendono un albero bilanciato?
- Non è direttamente intuitivo, lo proviamo per un caso (sinistra-sinistra), per gli altri casi è equivalente
- Associamo ad ogni nodo la sua altezza e vediamo come questa viene modificata dalle operazioni di rotazione

INSERIMENTO

Se effettuiamo la rotazione notiamo come la proprietà degli alberi AVL venga ripristinata



Qui vediamo che lo sbilanciamento è causato dal fatto che inserire il nodo nuovo fa crescere l'altezza di T1, di Y e di X. In particolare l'altezza di X prima dell'inserimento è h+2, e l'altezza del nodo Y, che sostituisce X, dopo la rotazione è di nuovo h+2.
Non solo l'albero si ribilancia, ma anche ogni antenato di X torna ad essere bilanciato.

INSERIMENTO

INSERIMENTO

- Abbiamo visto che per un caso le rotazioni ribilanciano il sottoalbero.

INSERIMENTO

- Abbiamo visto che per un caso le rotazioni ribilanciano il sottoalbero.
- Queste rotazioni possono creare problemi più in altro nell'albero?

INSERIMENTO

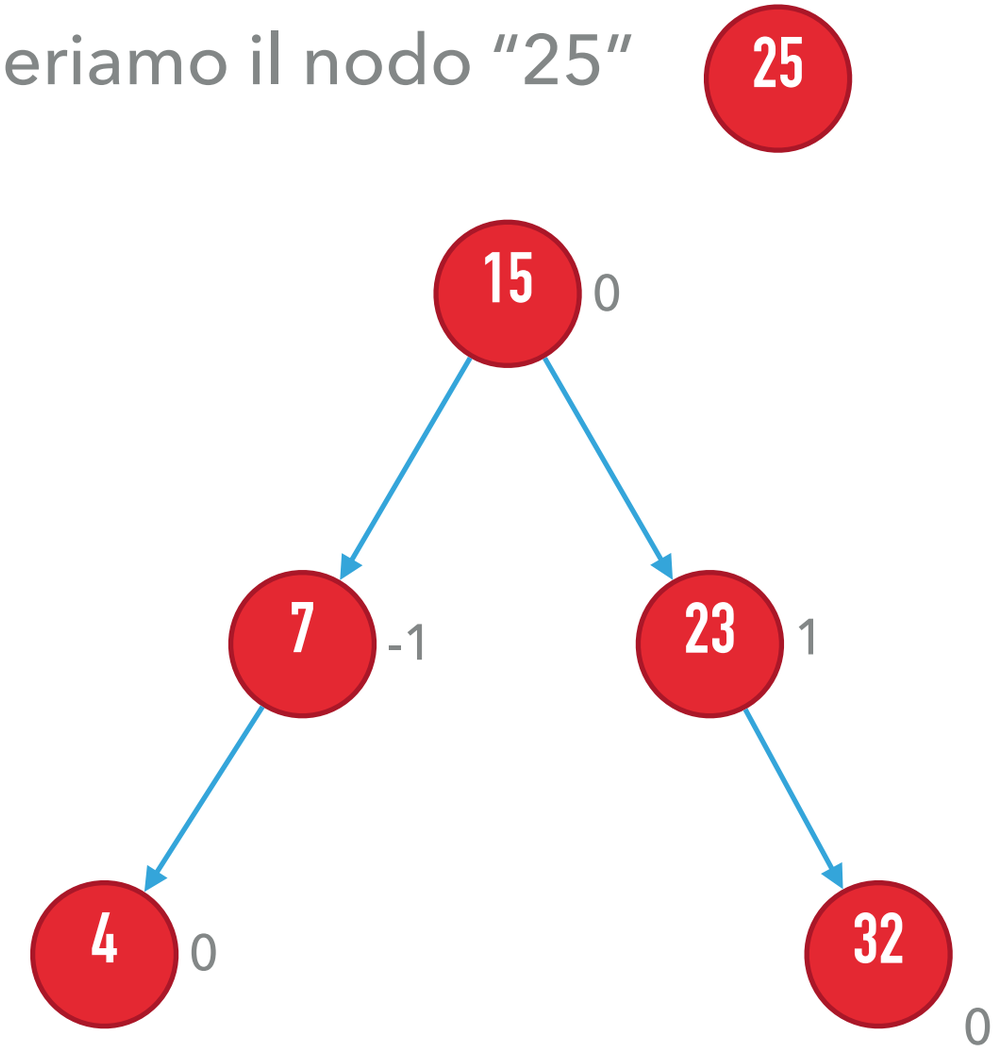
- Abbiamo visto che per un caso le rotazioni ribilanciano il sottoalbero.
- Queste rotazioni possono creare problemi più in altro nell'albero?
- No, perché l'albero ottenuto dopo le rotazioni ha esattamente la stessa altezza dell'albero **prima** dell'inserimento

INSERIMENTO

- Abbiamo visto che per un caso le rotazioni ribilanciano il sottoalbero.
- Queste rotazioni possono creare problemi più in altro nell'albero?
- No, perché l'albero ottenuto dopo le rotazioni ha esattamente la stessa altezza dell'albero **prima** dell'inserimento
- Quindi se i nodi antenati erano bilanciati prima dell'inserimento lo rimangono anche dopo

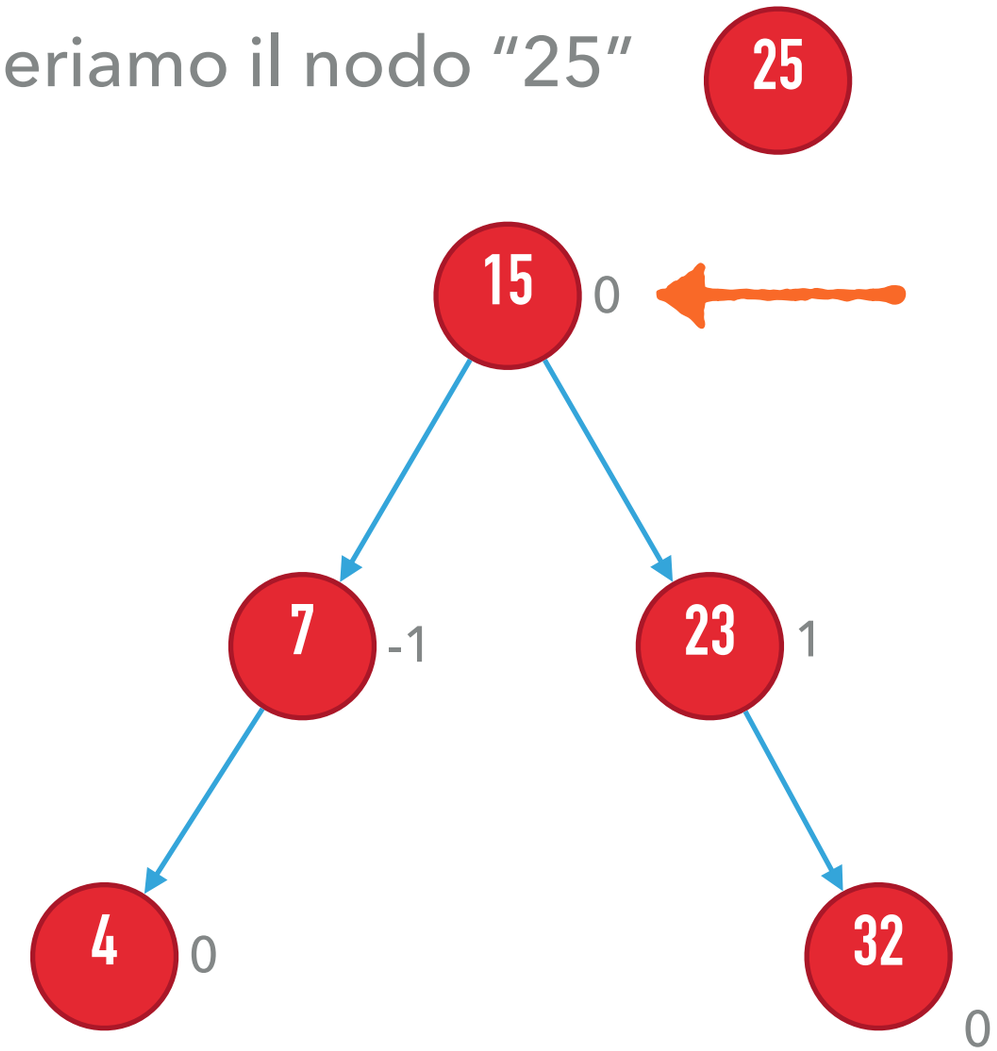
INSERIMENTO

Inseriamo il nodo “25”



INSERIMENTO

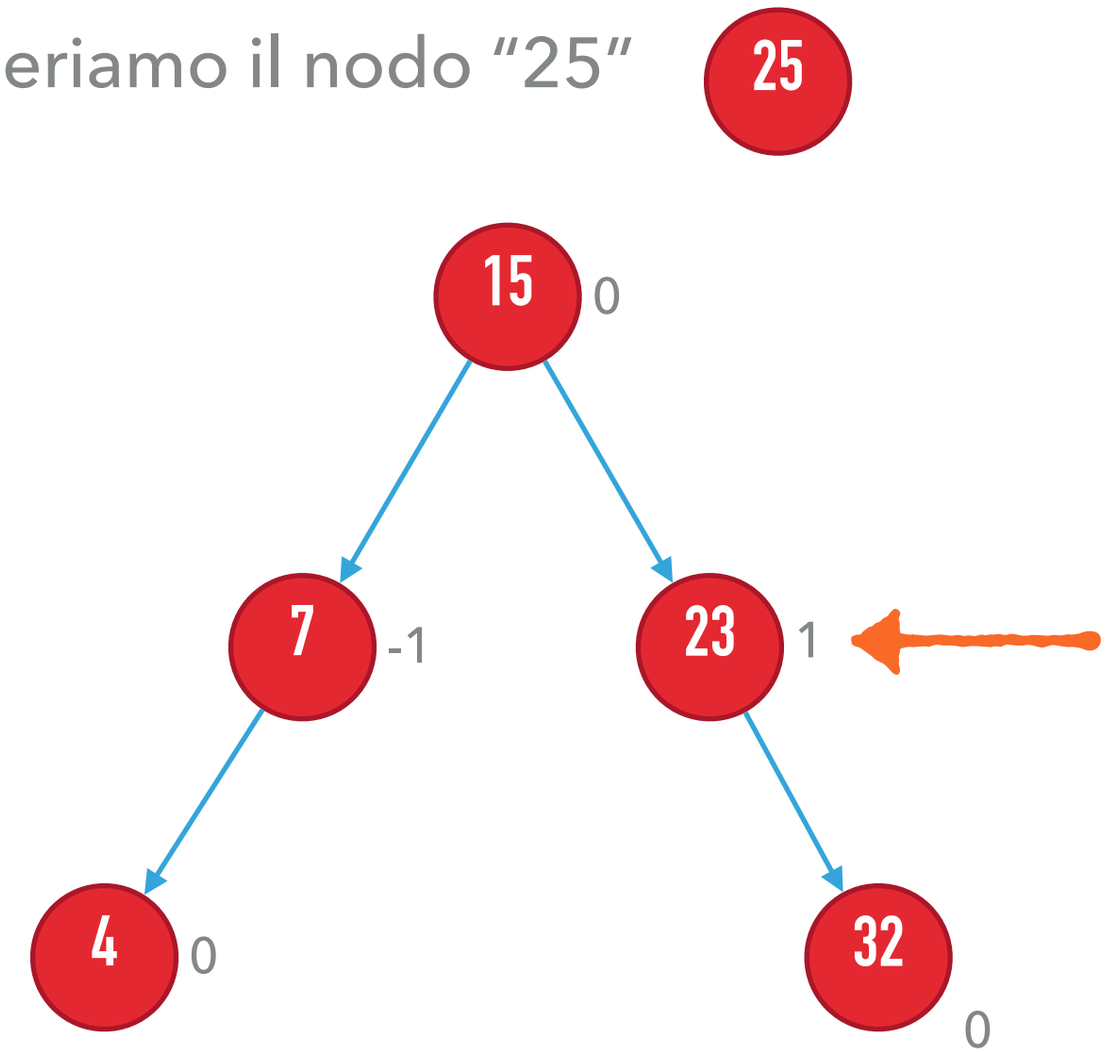
Inseriamo il nodo “25”

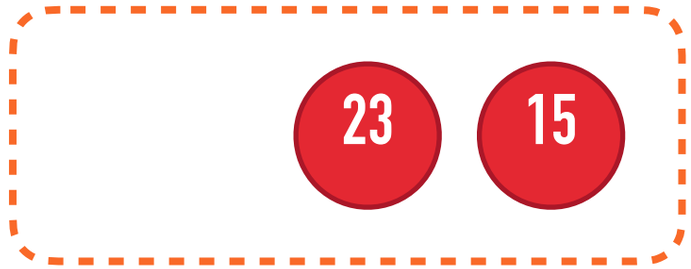


Stack dei nodi visitati

INSERIMENTO

Inseriamo il nodo “25”

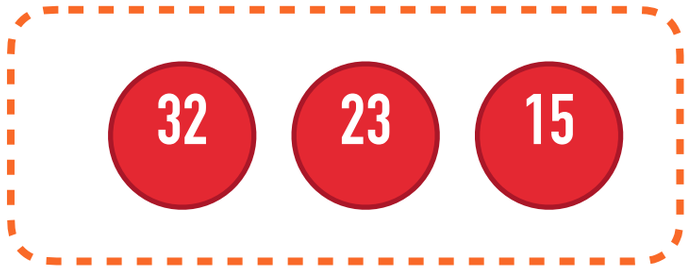




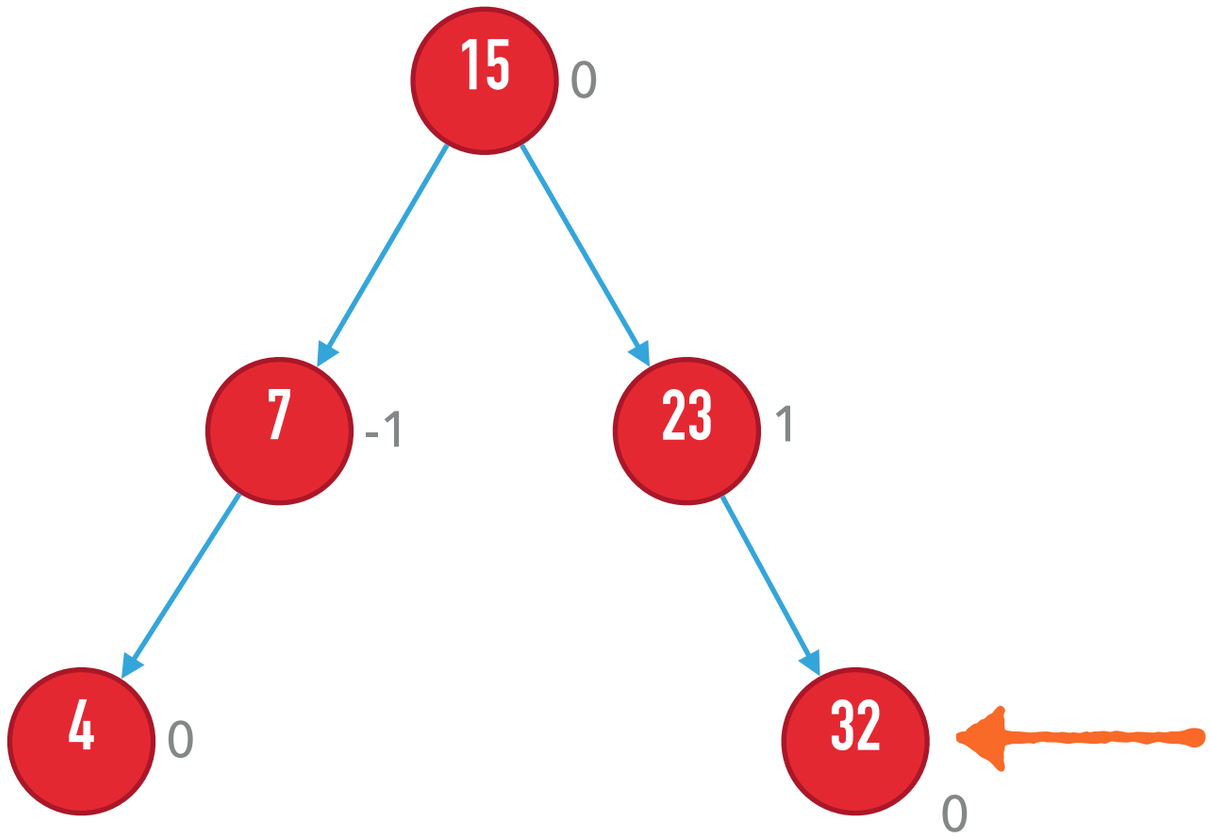
Stack dei nodi visitati

INSERIMENTO

Inseriamo il nodo “25”

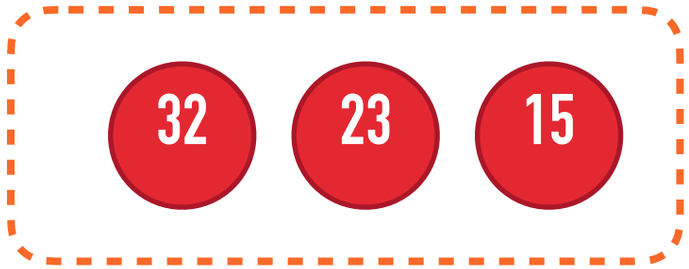


Stack dei nodi visitati

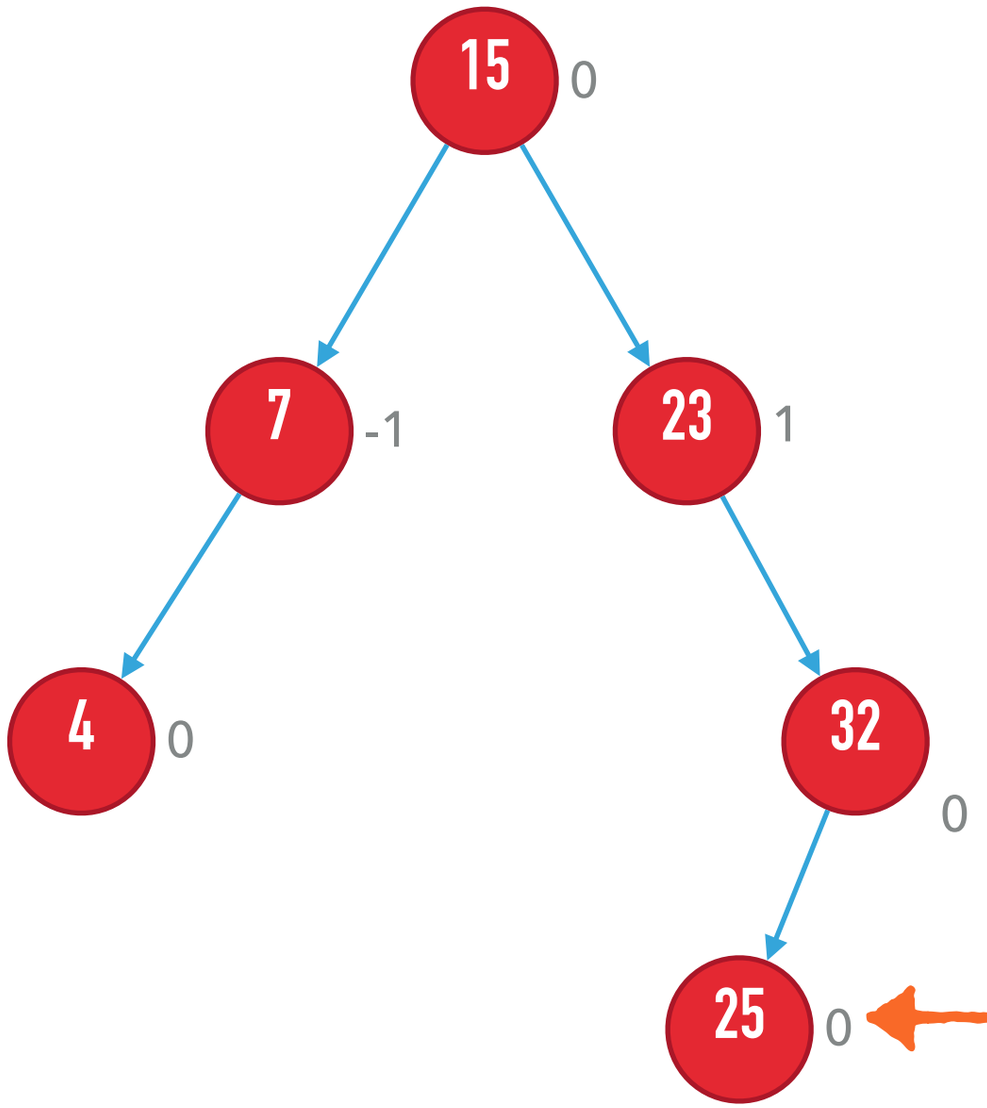


INSERIMENTO

Inseriamo il nodo “25”



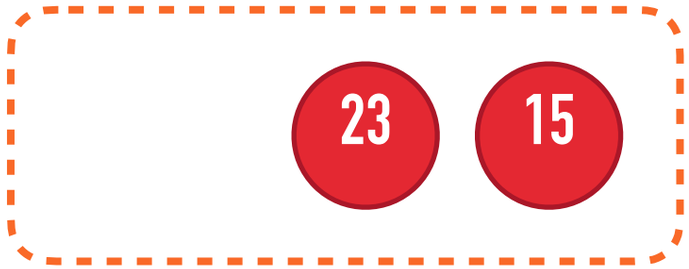
Stack dei nodi visitati



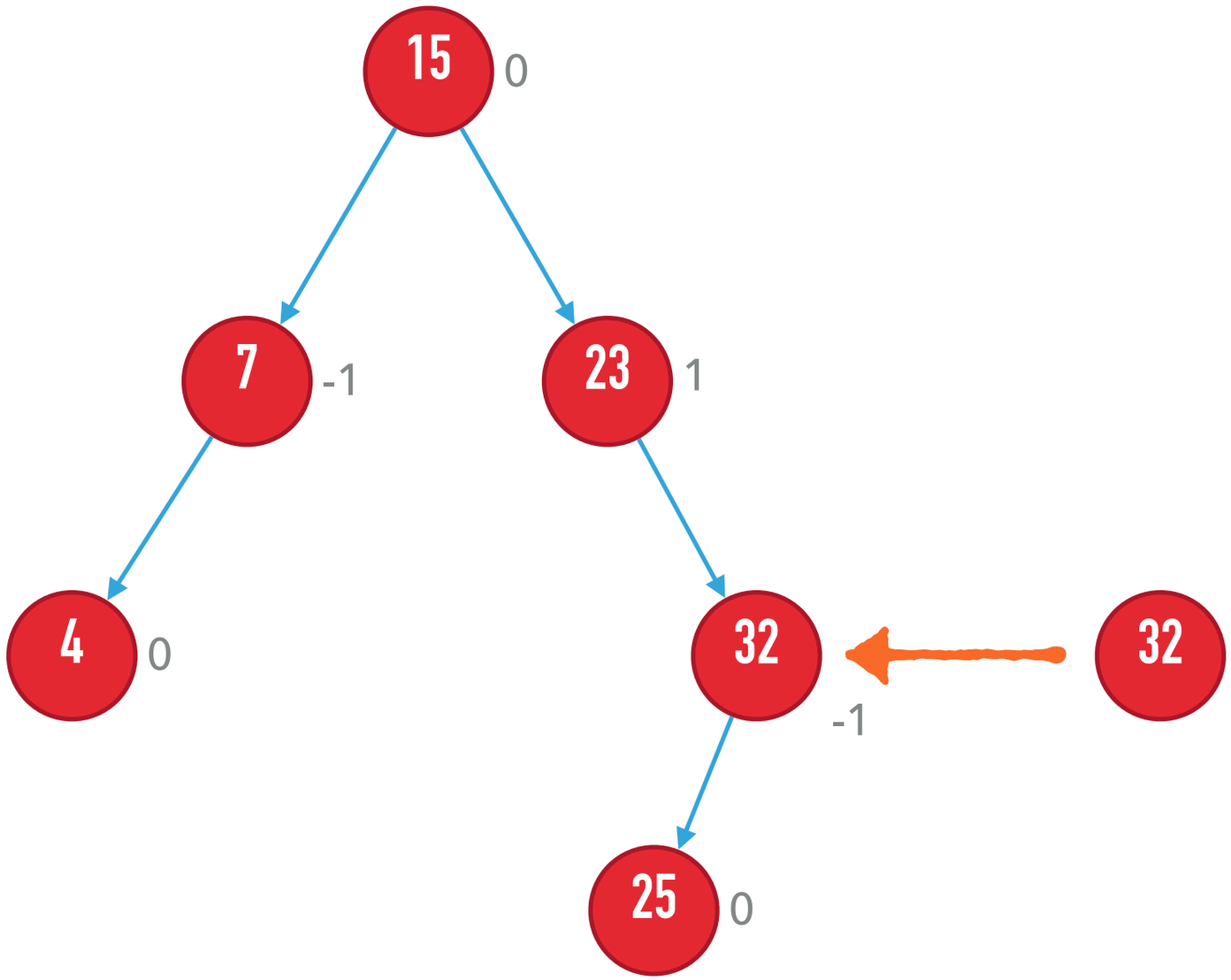
Inseriamo il nodo ed iniziamo ad aggiornare lo sbilanciamento

INSERIMENTO

Inseriamo il nodo “25”



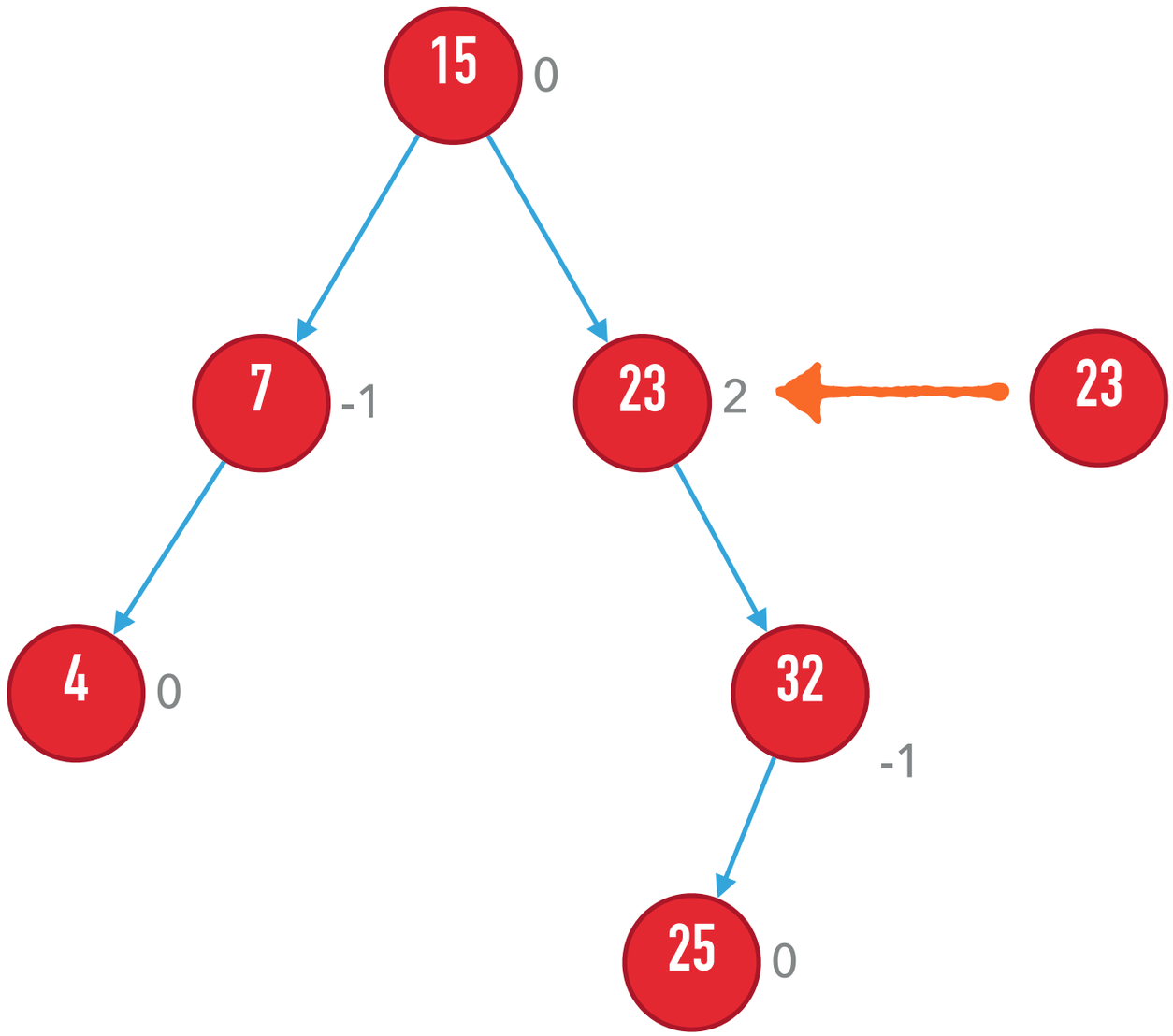
Stack dei nodi visitati



INSERIMENTO

Inseriamo il nodo “25”

Stack dei nodi visitati

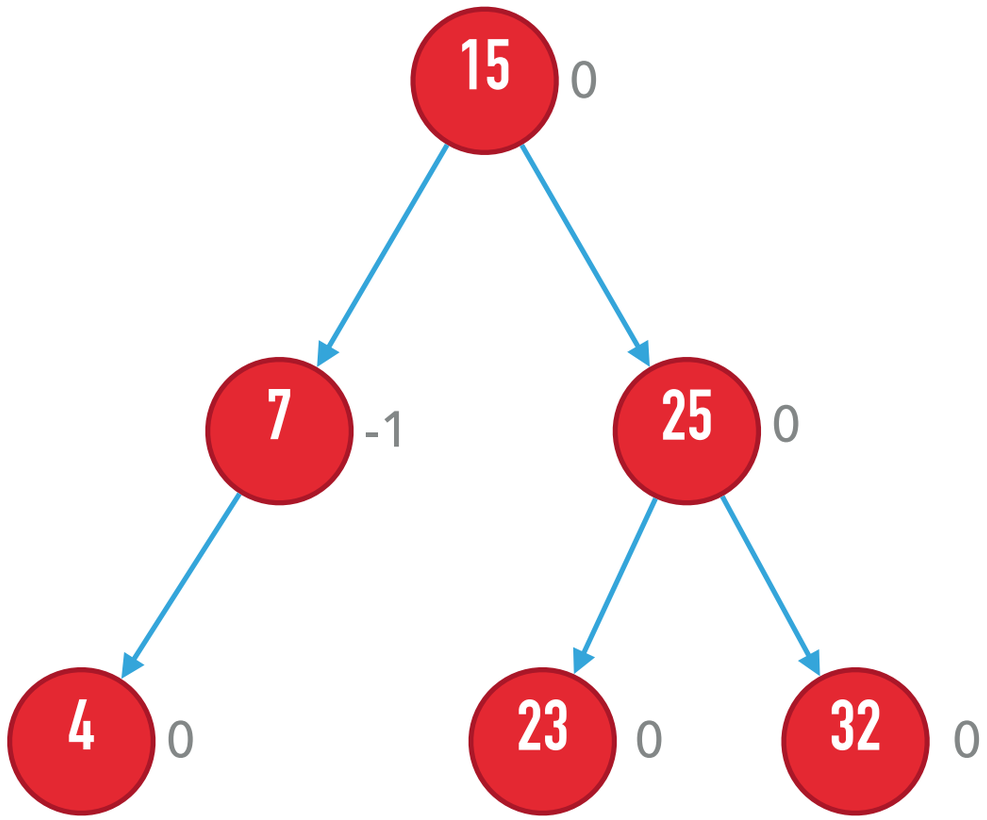


Abbiamo trovato il primo nodo sbilanciato: caso destra-sinistra

INSERIMENTO

Inseriamo il nodo “25”

Stack dei nodi visitati



Abbiamo ribilanciato l’albero

CANCELLAZIONE

CANCELLAZIONE

- La cancellazione avviene come per un Albero Binario di Ricerca, ma rimuovendo un nodo si può creare uno sbilanciamento.

CANCELLAZIONE

- La cancellazione avviene come per un Albero Binario di Ricerca, ma rimuovendo un nodo si può creare uno sbilanciamento.
- Utilizzando gli schemi di rotazioni visti prima possiamo ribilanciare.

CANCELLAZIONE

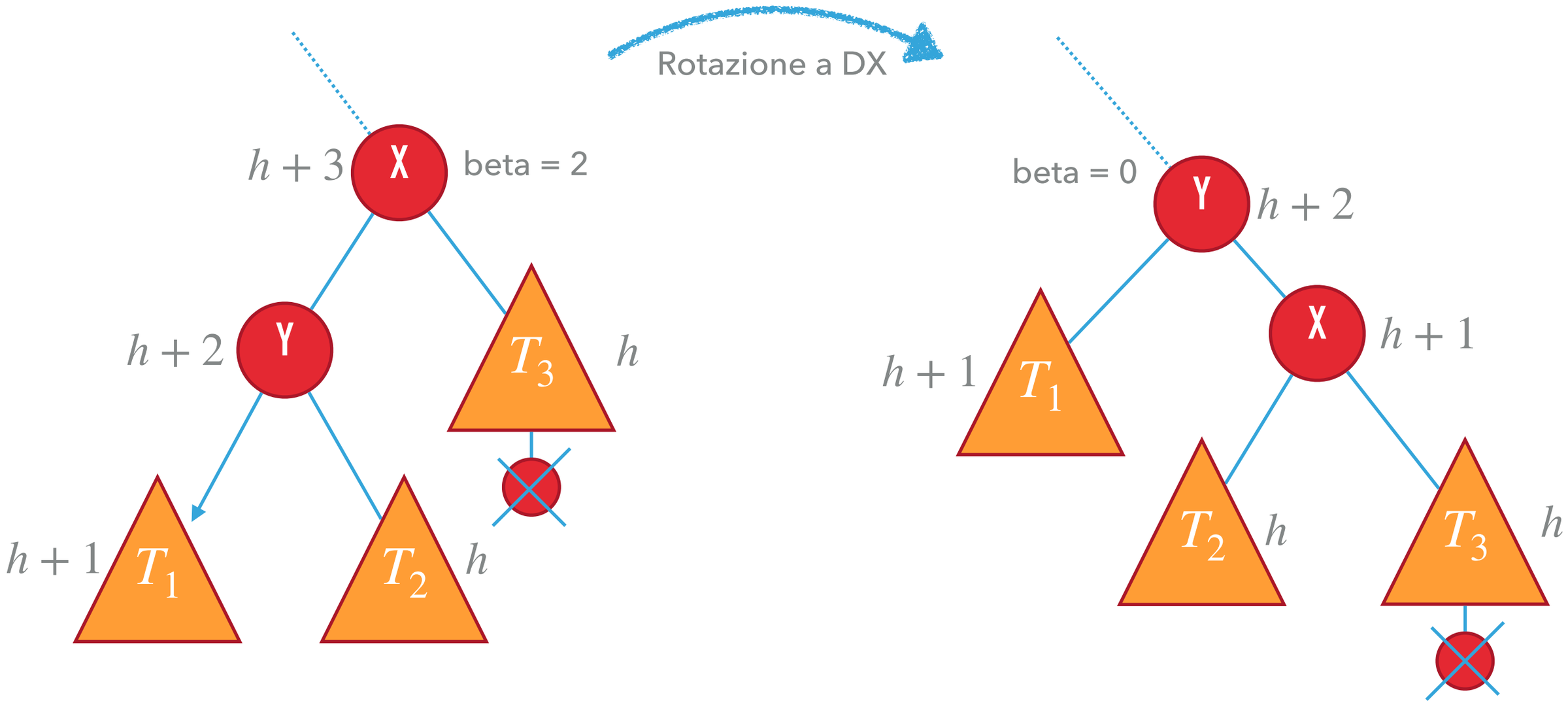
- La cancellazione avviene come per un Albero Binario di Ricerca, ma rimuovendo un nodo si può creare uno sbilanciamento.
- Utilizzando gli schemi di rotazioni visti prima possiamo ribilanciare.
- In questo caso però, l'altezza dopo il ribilanciamento descresce di una unità rispetto a prima della cancellazione.

CANCELLAZIONE

- La cancellazione avviene come per un Albero Binario di Ricerca, ma rimuovendo un nodo si può creare uno sbilanciamento.
- Utilizzando gli schemi di rotazioni visti prima possiamo ribilanciare.
- In questo caso però, l'altezza dopo il ribilanciamento descresce di una unità rispetto a prima della cancellazione.
- Quindi potremmo dover ribilanciare anche in tutti i nodi antenati. Quindi la cancellazione costa $O(h)$ passi.

CANCELLAZIONE

Se effettuiamo la rotazione notiamo come la proprietà degli alberi AVL venga ripristinata



Vediamo che l'albero torna ad essere bilanciato ma l'altezza decresce di uno.

ALBERI AVL

ALBERI AVL

- Gli alberi AVL sono alberi con profondità $O(\log n)$

ALBERI AVL

- Gli alberi AVL sono alberi con profondità $O(\log n)$
- Le operazioni di ricerca non cambiano rispetto agli alberi binari di ricerca normali

ALBERI AVL

- Gli alberi AVL sono alberi con profondità $O(\log n)$
- Le operazioni di ricerca non cambiano rispetto agli alberi binari di ricerca normali
- Le operazioni di inserimento e rimozione rimangono $O(h)$, con h l'altezza dell'albero, quindi $O(\log n)$

ALBERI AVL

- Gli alberi AVL sono alberi con profondità $O(\log n)$
- Le operazioni di ricerca non cambiano rispetto agli alberi binari di ricerca normali
- Le operazioni di inserimento e rimozione rimangono $O(h)$, con h l'altezza dell'albero, quindi $O(\log n)$
- Richiedono però informazione aggiuntiva salvata nei nodi (come minimo 2 bit/nodo)