ALBERI AVL
ALBERI ROSSO-NERI

INFORMATICA

COSA SONO GLI ALBERI AVL

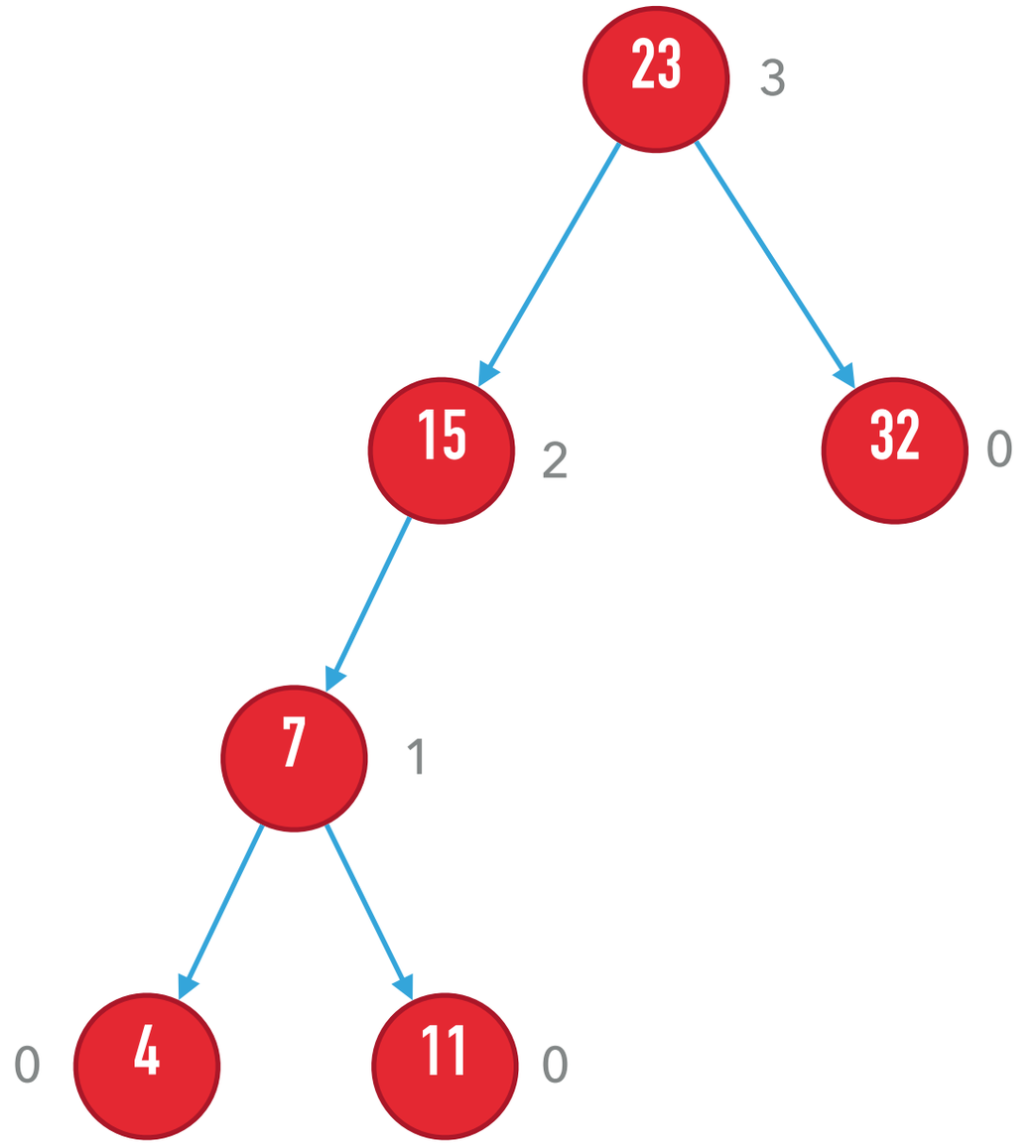
- AVL da Adelson-Velsky e Landis, cognomi dei due ideatori (anno 1962)
- Sono alberi binari di ricerca che sono sempre bilanciati
- Ove bilanciati non significa esattamente profondità $\log n$, ma $O(\log n)$
- Nel caso degli alberi AVL la costante del $\log n$ è bassa

DIFFERENZA DAGLI ALBERI SPLAY

- A differenza degli alberi splay, gli alberi AVL assicurano il bilanciamento...
- ...quindi le operazioni di ricerca richiedono sempre tempo $O(\log n)$
- Le operazioni di inserimento e rimozione modificano l'albero per mantenerlo bilanciato
- Le operazioni di ricerca non modificano l'albero (al contrario degli alberi splay)

ALTEZZA DI UN NODO

Definiamo come altezza di un nodo il percorso più lungo da quel nodo a una foglia

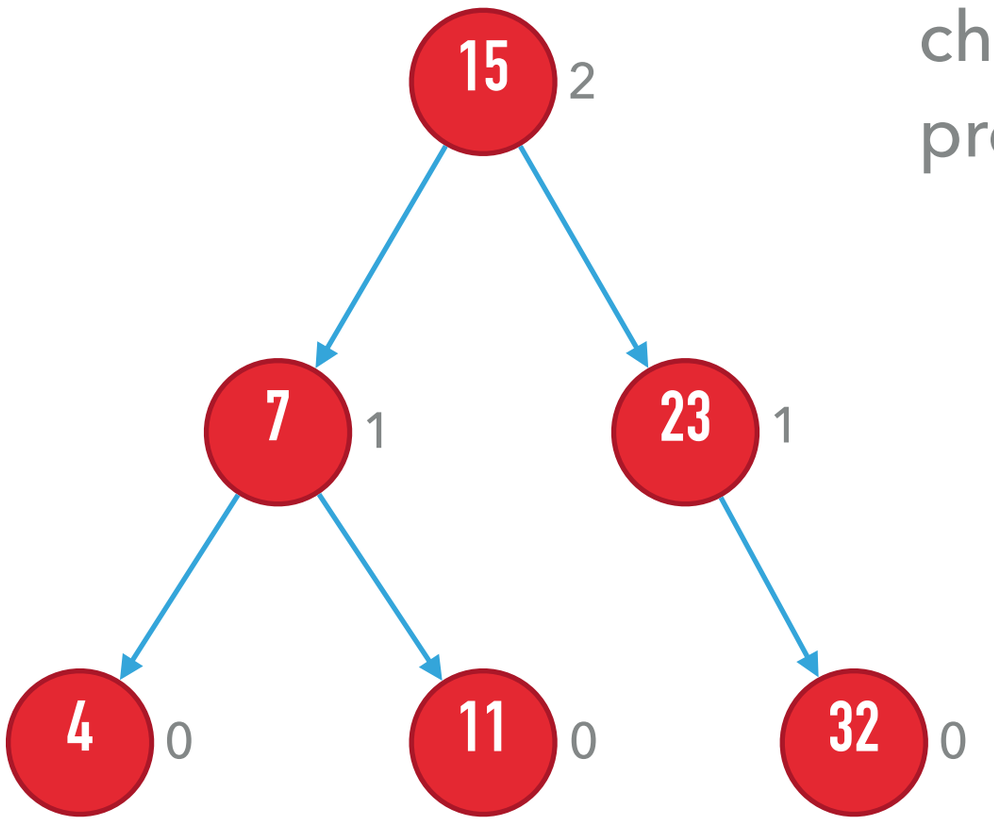


Possiamo facilmente calcolarla per ogni nodo come
1 + max{altezza del figlio sinistro, altezza del figlio destro}

PROPRIETÀ DEGLI ALBERI AVL

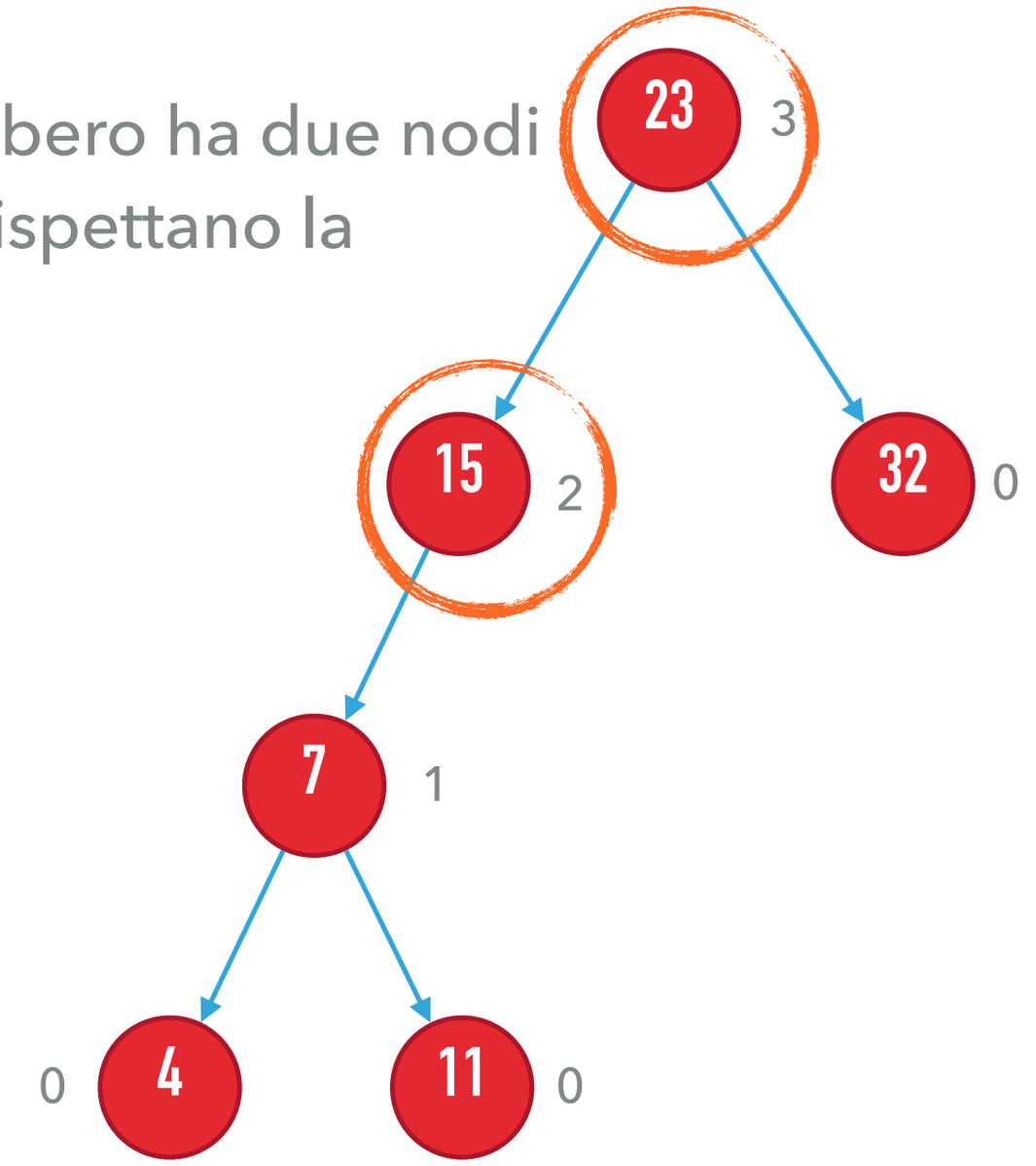
- Gli alberi AVL hanno la proprietà che la differenza in altezza del figlio destro e del figlio sinistro di ogni nodo è al più 1 (in valore assoluto)
- In caso di figlio assente si assume altezza di -1

DIFFERENZA IN ALTEZZA

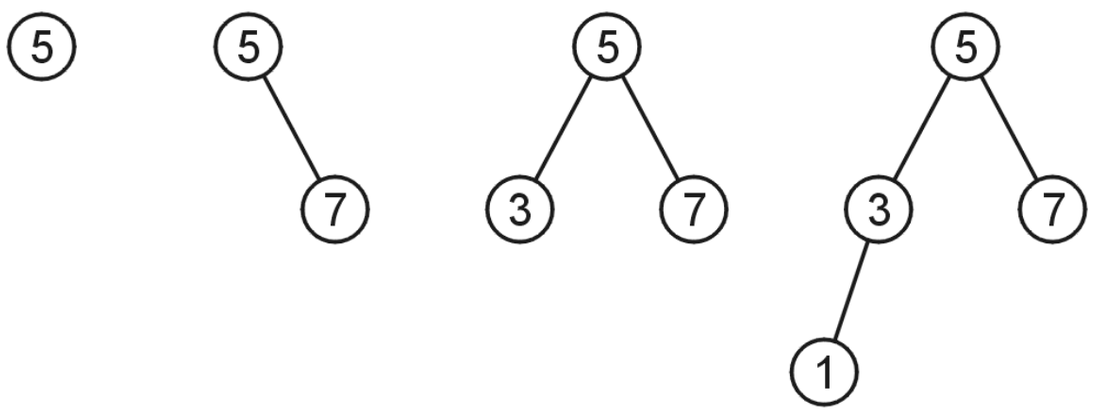


Questo albero rispetta la proprietà
(tutte le differenze sono zero o uno)

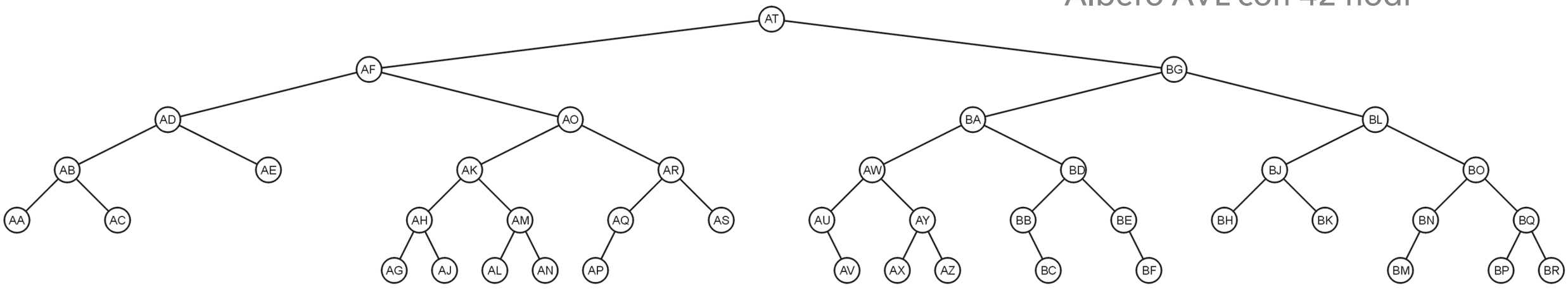
Questo albero ha due nodi che non rispettano la proprietà



ESEMPI DI ALBERI AVL



Alberi AVL con 1, 2, 3 e 4 nodi



Albero AVL con 42 nodi

Sembra abbastanza “piatto”, con un’altezza limitata.

Possiamo dimostrare che un albero AVL ha sempre un’altezza $O(\log n)$ quando contiene n nodi?

QUANTI NODI CONTIENE UN ALBERO AVL

- Invertiamo il problema di chiedere l'altezza dell'albero chiedendoci invece quale sia il numero minimo di nodi che un albero AVL di altezza h può contenere

- Indichiamo con N_h questo numero di nodi

- Dalla slide precedente otteniamo $N_0 = 1, N_1 = 2, N_2 = 4$

QUANTI NODI CONTIENE UN ALBERO AVL

- Quale è il caso peggiore per un albero AVL?
- Quando c'è uno sbilanciamento di 1 tra i due figli, quindi possiamo definire N_h come una ricorrenza
- $N_h = 1 + N_{h-1} + N_{h-2}$
- Come possiamo ottenere un bound per il valore di N_h? Vediamo due modi

QUANTI NODI CONTIENE UN ALBERO AVL

- Dato che $N_{h-1} \geq N_{h-2}$ possiamo riscrivere:
  $N_h = 1 + N_{h-1} + N_{h-2} \geq 1 + 2N_{h-2} \geq 2N_{h-2}$
- Espandendo la ricorrenza otteniamo che per arrivare al caso base (valore di h costante) dobbiamo espandere almeno $h/2$ volte:
- $N_h \geq 2N_{n-2} \geq 4N_{h-4} \geq 8N_{h-6} \geq \ldots \geq 2^i N_{h-2i}$
- Otteniamo quindi $N_h \geq 2^{h/2}$

QUANTI NODI CONTIENE UN ALBERO AVL

- Possiamo ora prendere il logaritmo in base 2 da entrambi i lati:
  $\log_2 N_h \geq \log_2 2^{h/2} = h/2$

- Otteniamo quindi un bound sull'altezza: $h \leq 2 \log_2 N_h$

- Questo significa che per ogni numero n di nodi, anche se disposti nel caso peggiore saranno in un albero di altezza non più di $2 \log_2 n = O(\log n)$

QUANTI NODI CONTIENE UN ALBERO AVL

- Vediamo ora un metodo un poco più raffinato

$N_h = 1 + N_{h-1} + N_{h-2}$ assomiglia molto alla definizione dei numeri di Fibonacci:

$F_0 = 0, F_1 = 1, F_2 = 1\ F_3 = 2, \ldots, F_k = F_{k-1} + F_{k-2}$

Mostriamo per induzione che $N_h = F_{h+3} - 1$

Casi base: $N_0 = 1 = F_3 - 1$ e $N_1 = 2 = F_4 - 1$

QUANTI NODI CONTIENE UN ALBERO AVL

Passo induttivo:

$$
\begin{aligned}
N_h &= N_{h-1} + N_{h-2} + 1 \\
&= F_{h+2} - 1 + F_{h+1} - 1 + 1 \\
&= F_{h+2} + F_{h+1} - 1 \\
&= F_{h+3} - 1
\end{aligned}
$$

Quindi possiamo usare i numeri di Fibonacci per ottenere un bound sull'altezza dell'albero.

QUANTI NODI CONTIENE UN ALBERO AVL

Sappiamo che

$$F_k = \frac{\phi^k - \psi^k}{\sqrt{5}} \text{ con } \phi = \frac{1+\sqrt{5}}{2} \text{ e } \psi = \frac{1-\sqrt{5}}{2}$$

In generale $F_k \geq \frac{\phi^k - 1}{\sqrt{5}}$ e quindi usando $N_h = F_{h+3} - 1$:

$$N_h \geq \frac{\phi^{h+3} - 1}{\sqrt{5}} - 1, \text{ ovvero } N_h \geq \frac{\phi^{h+3}}{\sqrt{5}} - \frac{1+\sqrt{5}}{\sqrt{5}} \geq \frac{\phi^{h+3}}{\sqrt{5}} - 1.45$$

QUANTI NODI CONTIENE UN ALBERO AVL

Riordinando: $\sqrt{5}(N_h + 1.45) \geq \phi^{h+3}$

Prendiamo il logaritmo in base ϕ: $\log_\phi(\sqrt{5}N_h + 1.45) \geq h + 3$

Ovvero: $h \leq \log_\phi(\sqrt{5}(N_h + 1.45)) - 3$

Cambio di base del logaritmo: $h \leq \dfrac{\log_2(\sqrt{5}(N_h + 1.45))}{\log_2 \phi} - 3$

Riscriviamo come $h \leq \dfrac{1}{\log_2 \phi} \log(N_h + 2) + c$

QUANTI NODI CONTIENE UN ALBERO AVL

Sapendo che $\frac{1}{\log_2 \phi} \leq 1.441$ si ha $h \leq 1.441 \log(N_h + 2) + c$

Questo significa che un albero con n nodi ha altezza $O(\log n)$

Possiamo anche dire di più: dato che libero binario più piccolo che contiene n nodi ha profondità $\lceil \log_2 n \rceil$, non siamo molto distanti da quel valore dato che il fattore moltiplicativo è circa 1.44.

COME GARANTIRE IL BILANCIAMENTO

- Per garantire il bilanciamento dobbiamo memorizzare dell'informazione all'interno dei nodi.
- Due possibilità:
  - Altezza del nodo
  - Quale è lo sbilanciamento: $\{-1,0,1\}$
    - Il valore rappresenta la differenza tra altezza del figlio destro e altezza del figlio sinistro

COME GARANTIRE IL BILANCIAMENTO

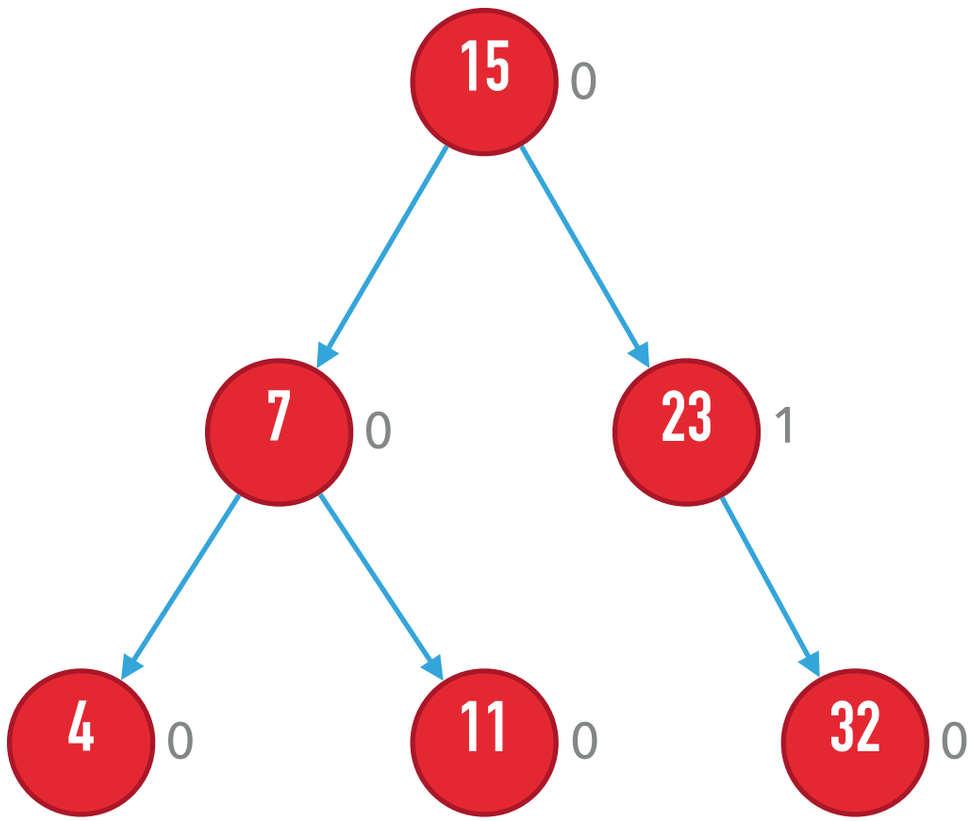
- Entrambe le possibilità sono valide
- Cambiano lievemente le operazioni di aggiornamento
- Noi scegliamo di memorizzare il bilanciamento perché può essere memorizzato in un numero costante di bit (2) indipendentemente dal numero di nodi nell'albero
- Anche se il numero di bit che si usano per memorizzare l'altezza cresce lentamente (l'altezza cresce logaritmicamente rispetto al numero di nodi)

INSERIMENTO

- 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
- Vediamo come ogni sbilanciamento può essere risolto tramite una sequenza di rotazioni

INSERIMENTO

Inseriamo il nodo “19” 19

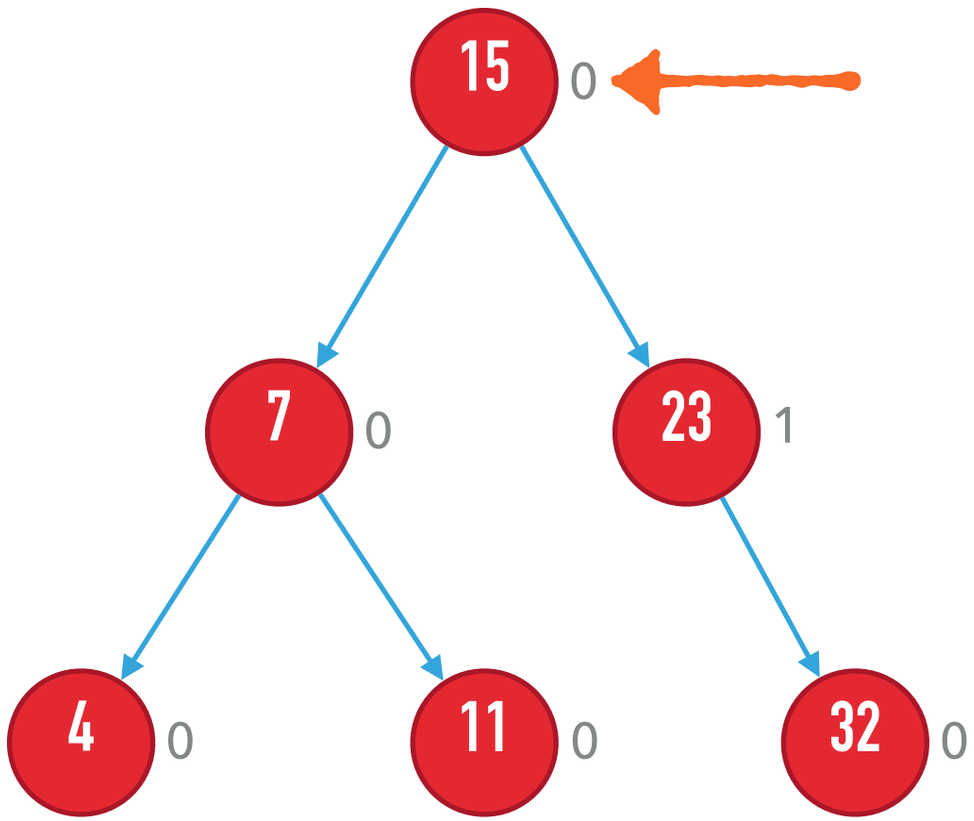


INSERIMENTO

Inseriamo il nodo “19” 19

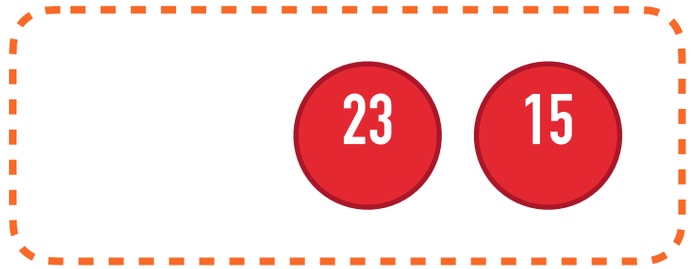
15

Stack dei nodi visitati

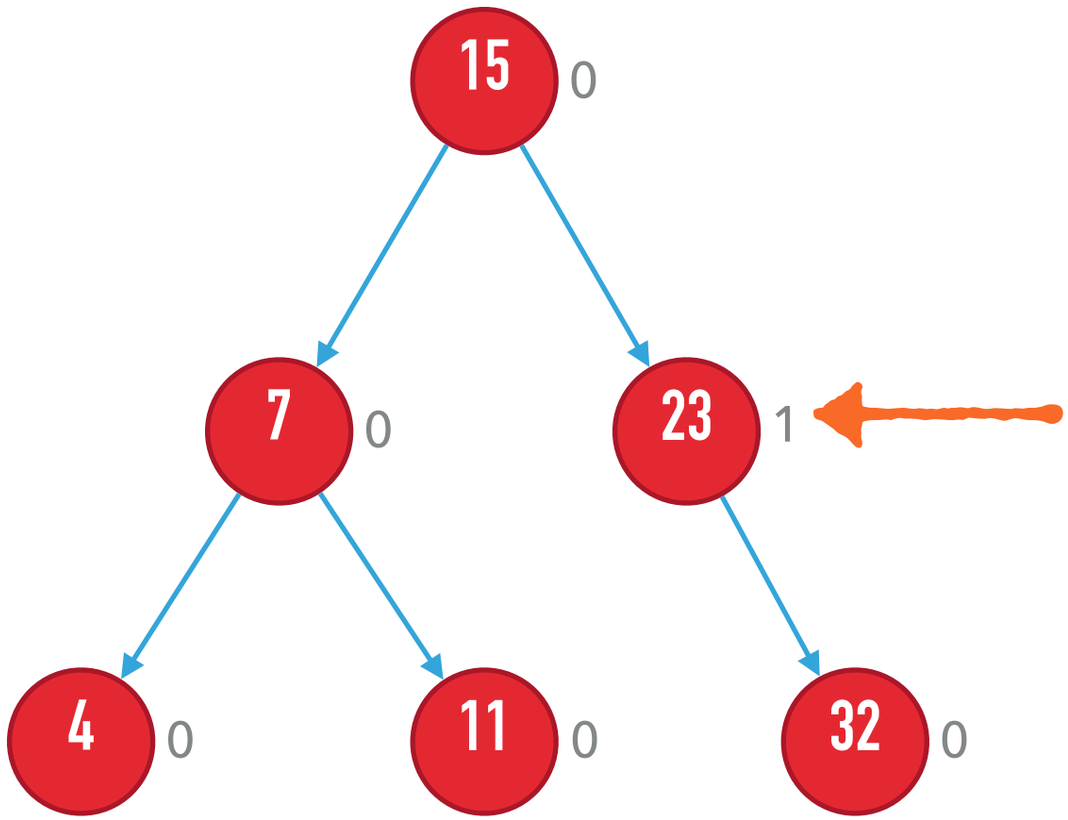


INSERIMENTO

Inseriamo il nodo “19”

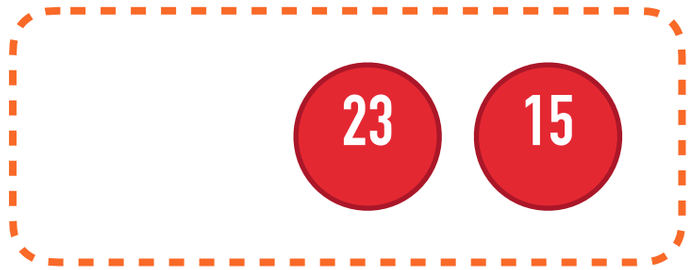


Stack dei nodi visitati

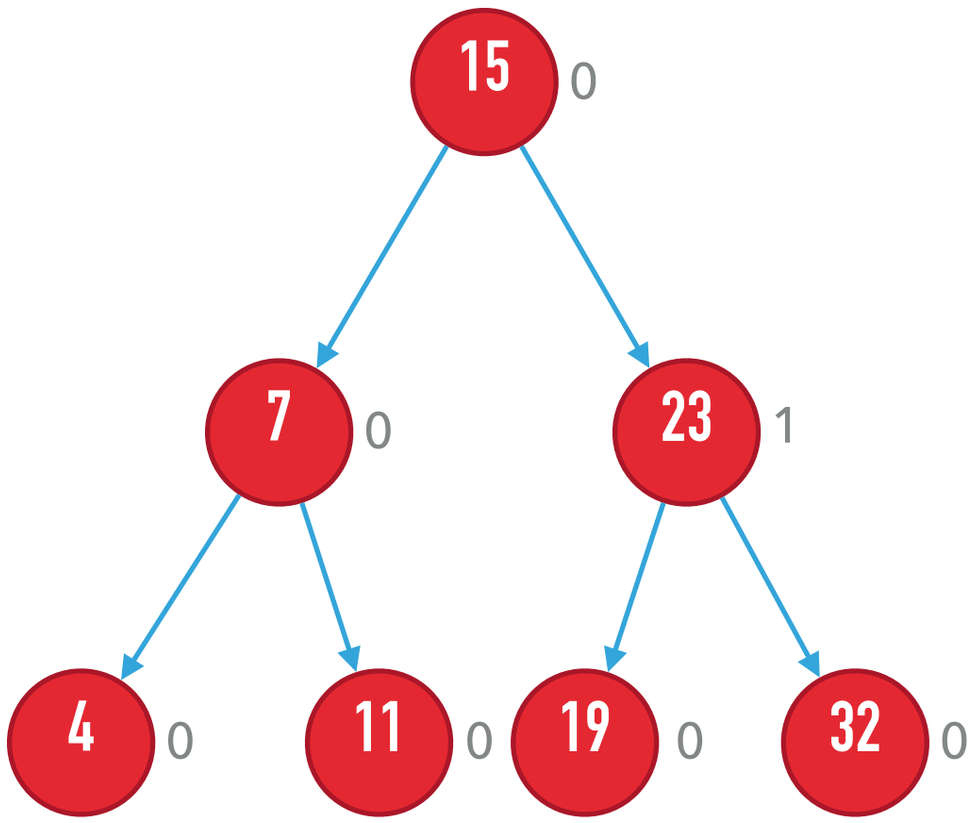


INSERIMENTO

Inseriamo il nodo “19”



Stack dei nodi visitati

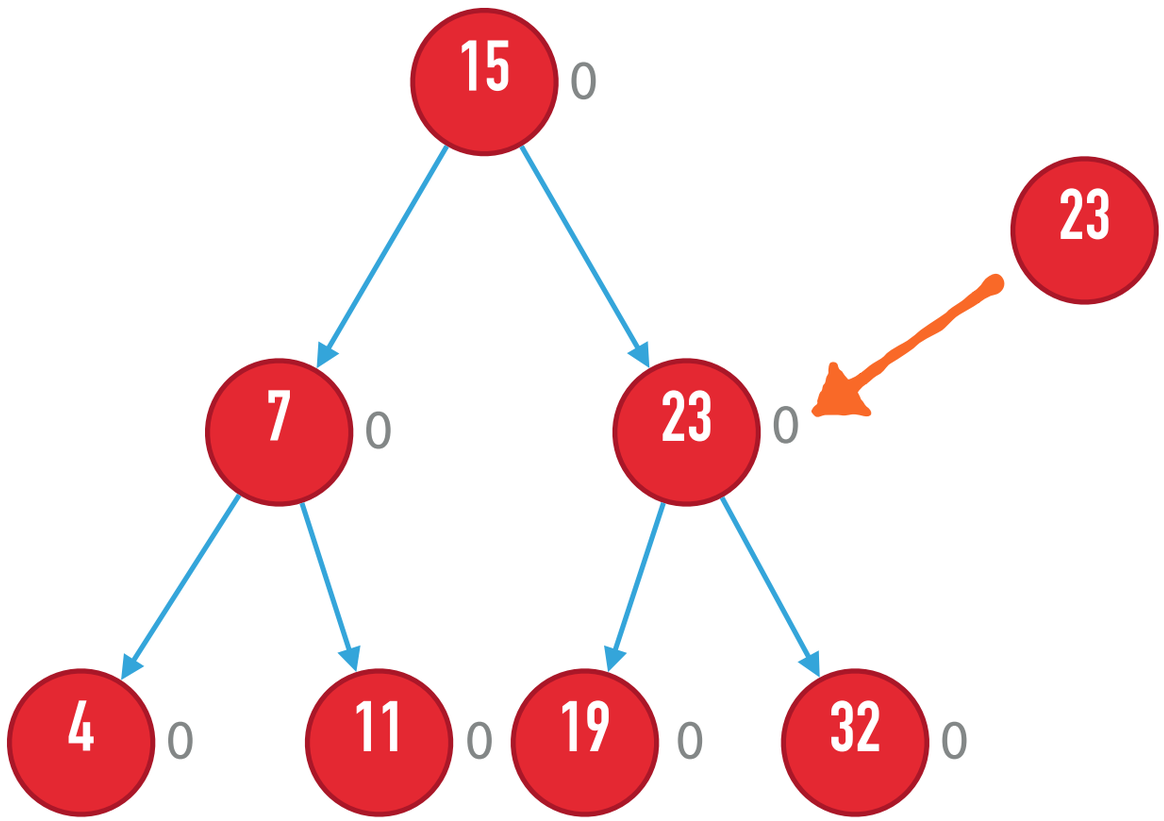


Abbiamo inserito il nodo, ora svuotiamo lo stack e aggiorniamo il bilanciamento

INSERIMENTO

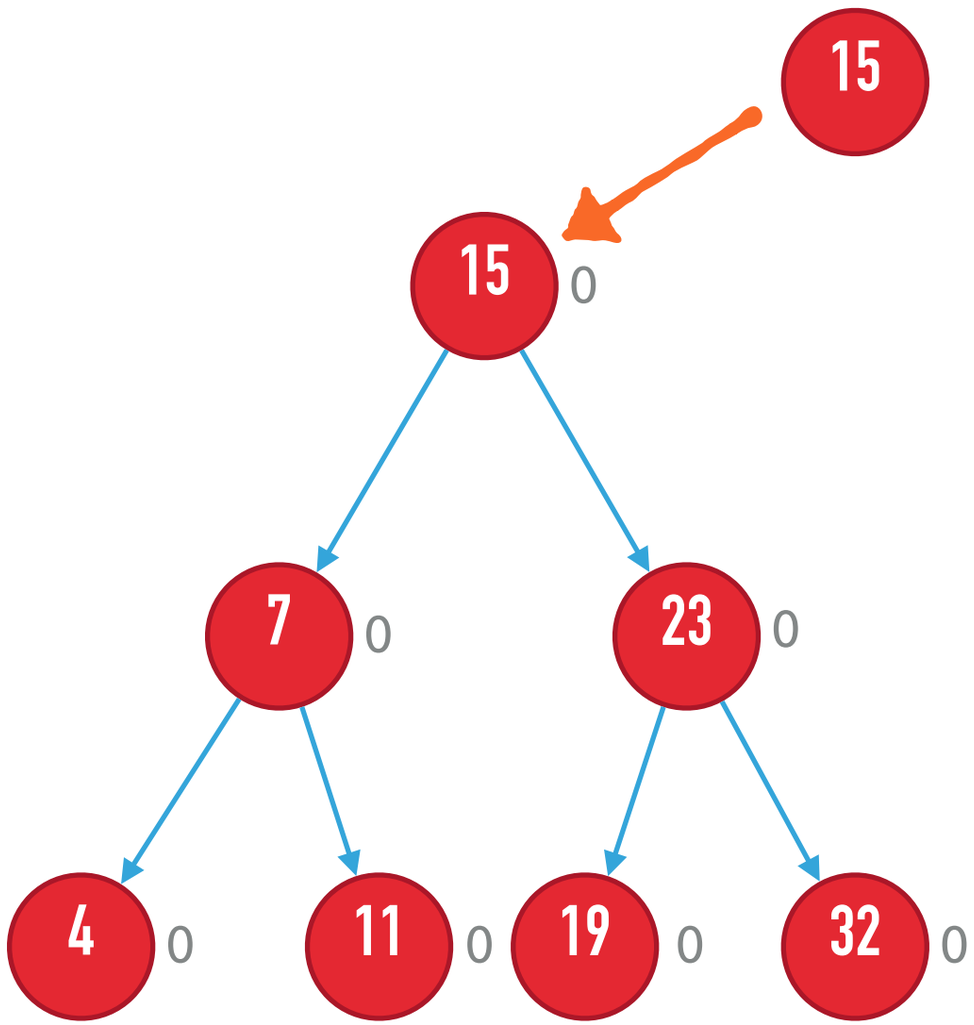
Inseriamo il nodo “19”

Stack dei nodi visitati



INSERIMENTO

Inseriamo il nodo “19”



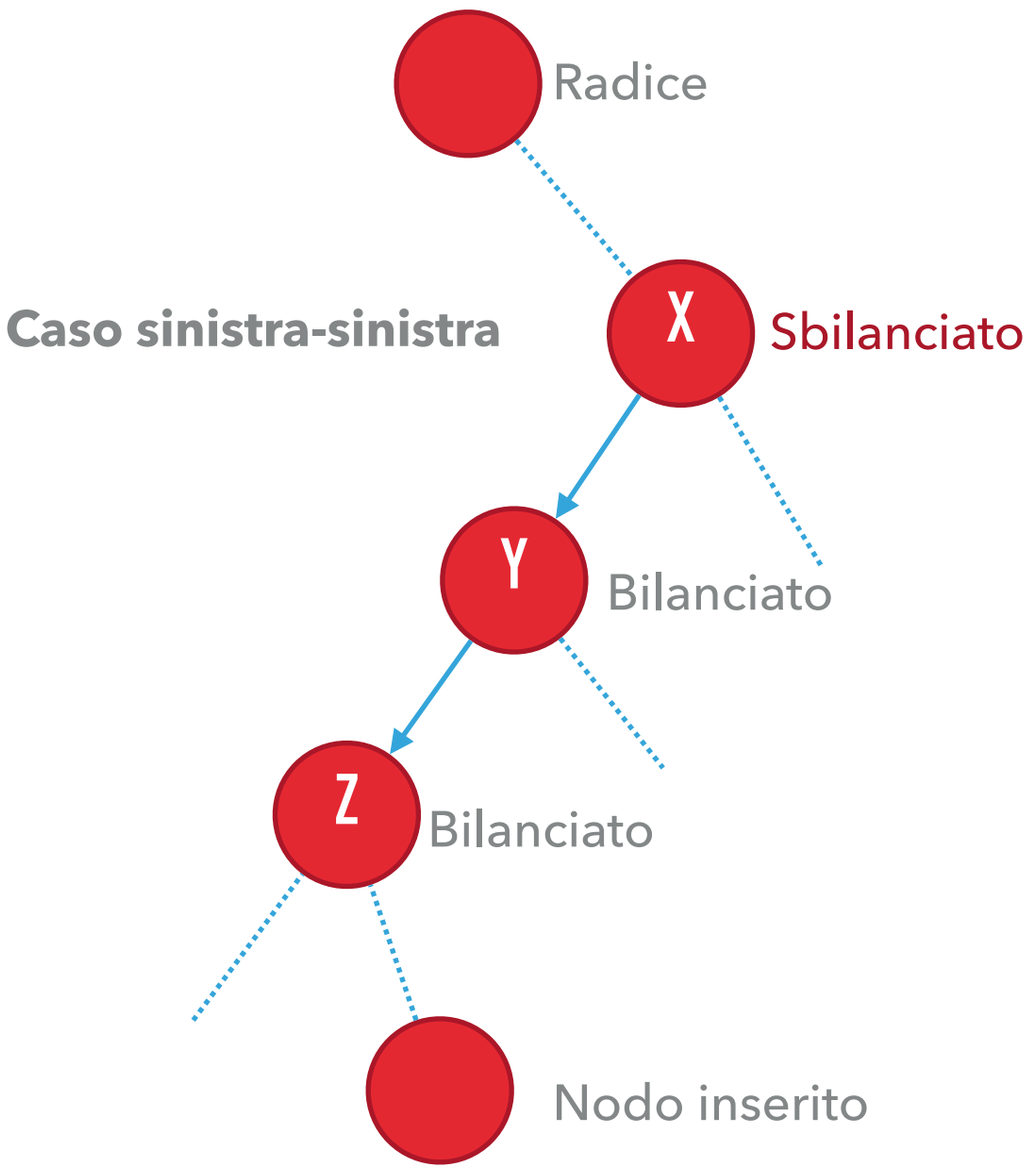
Stack dei nodi visitati

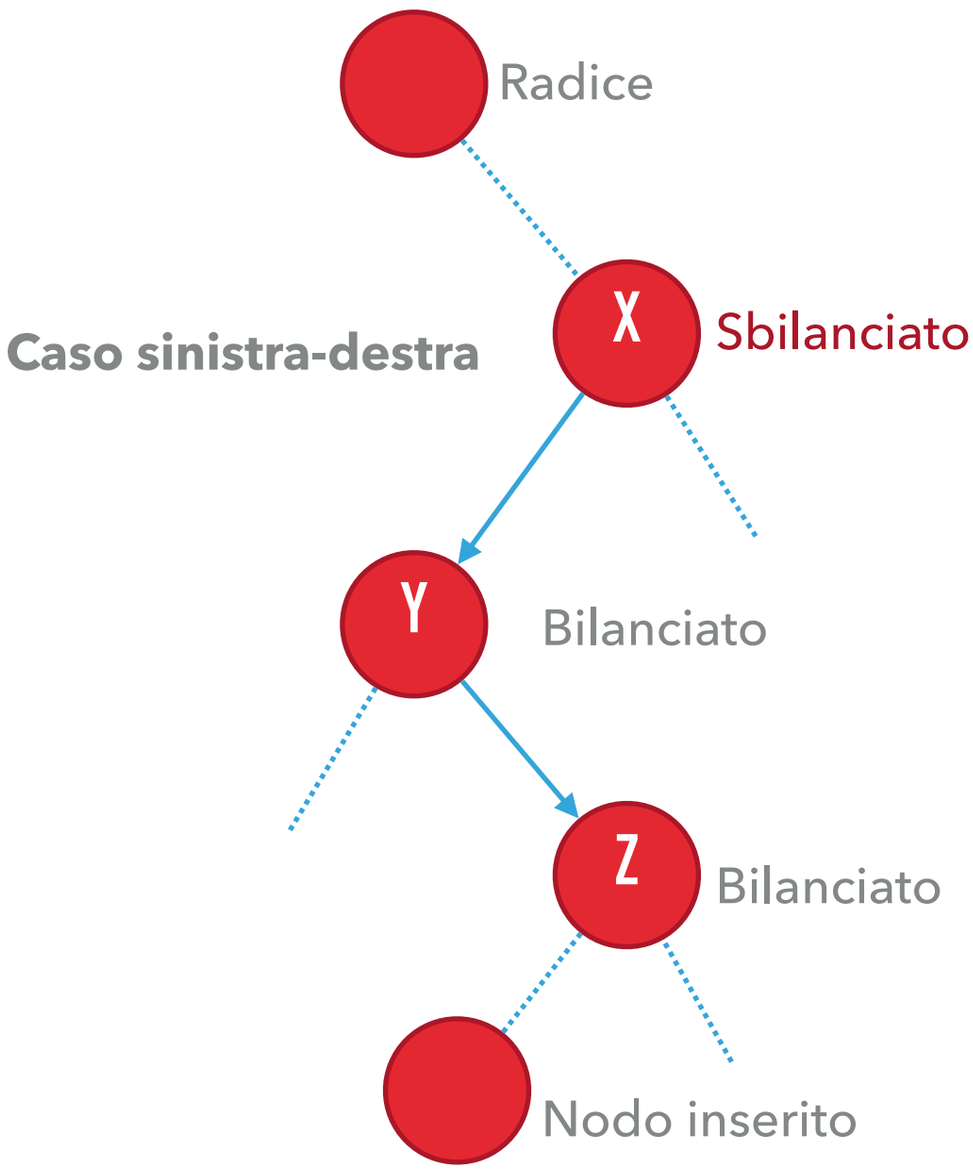
INSERIMENTO

- 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.

INSERIMENTO

Nel percorso che va dalla radice al nodo inserito possiamo trovarci in uno di questi quattro casi:

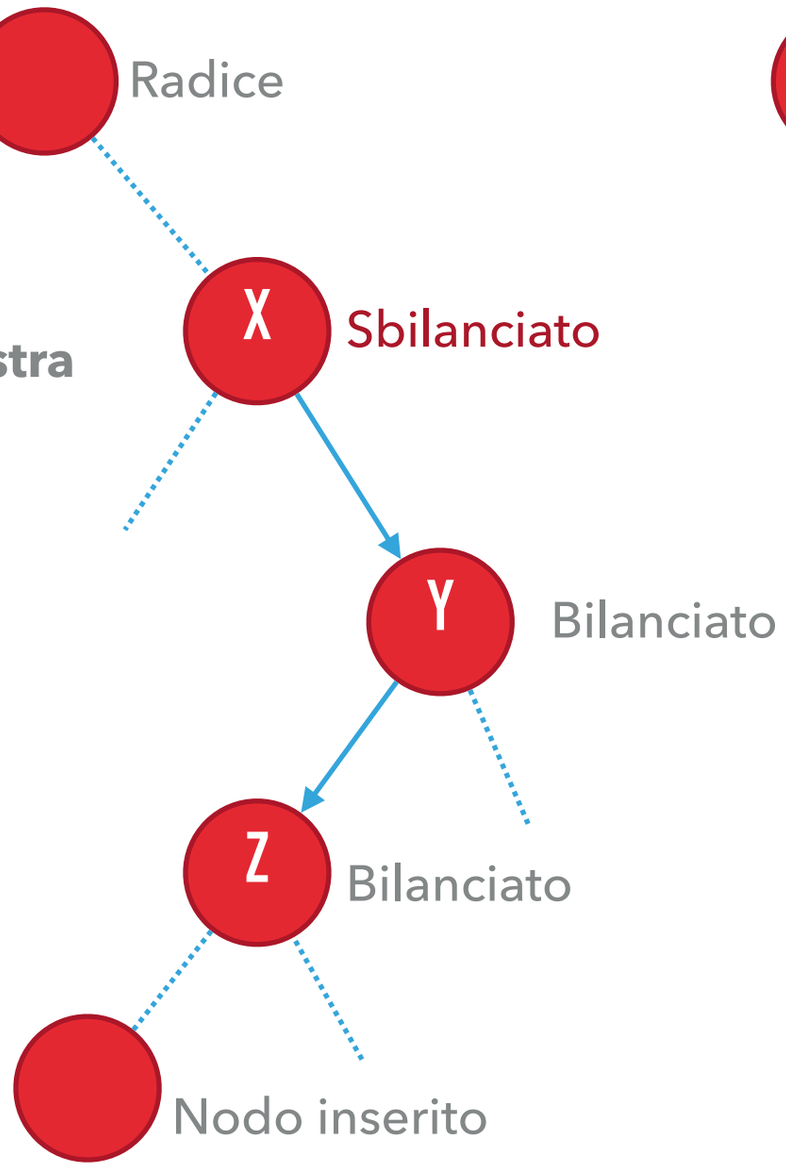




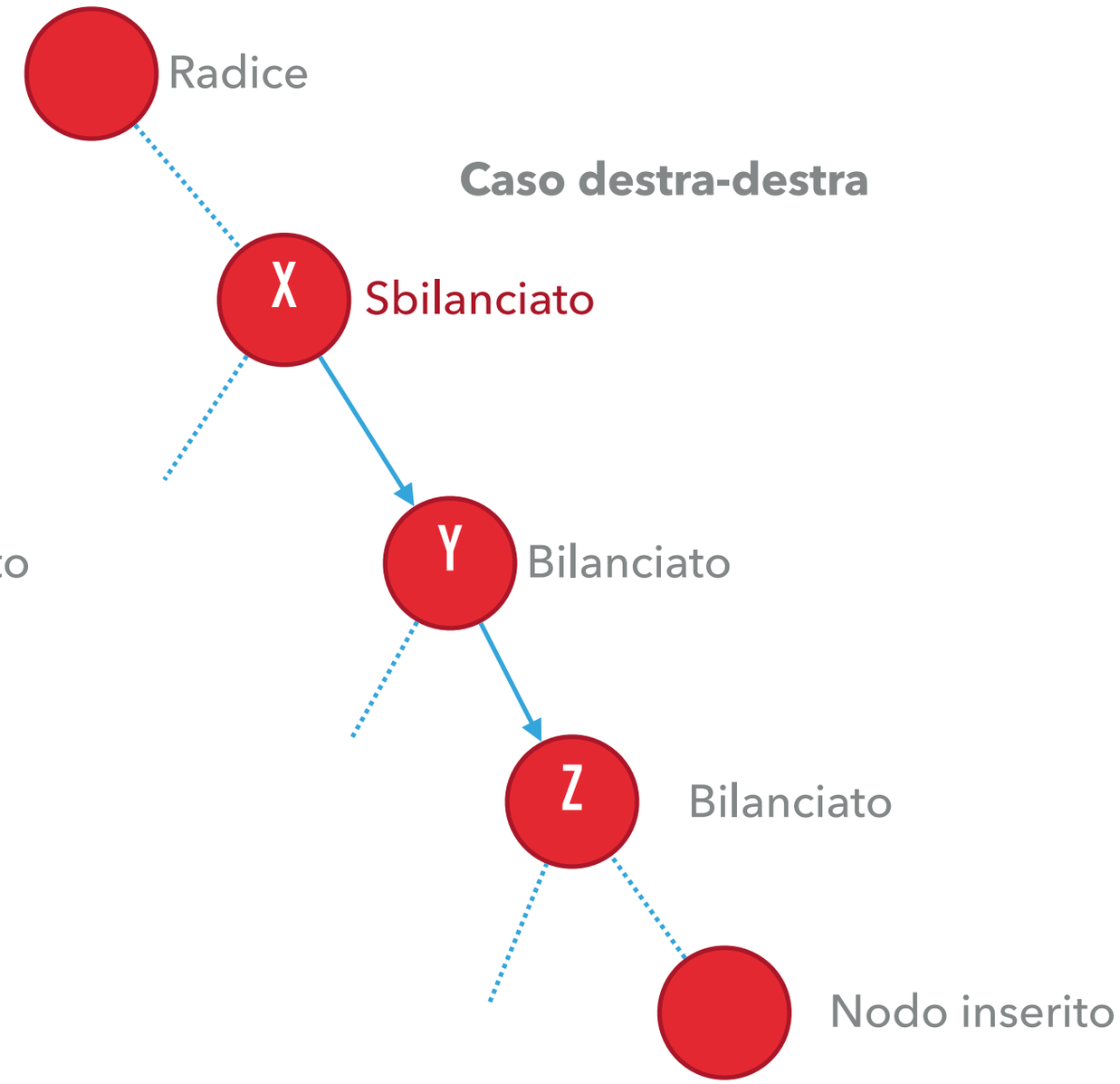
INSERIMENTO

Nel percorso che va dalla radice al nodo inserito possiamo trovarci in uno di questi quattro casi:

Caso destra-sinistra

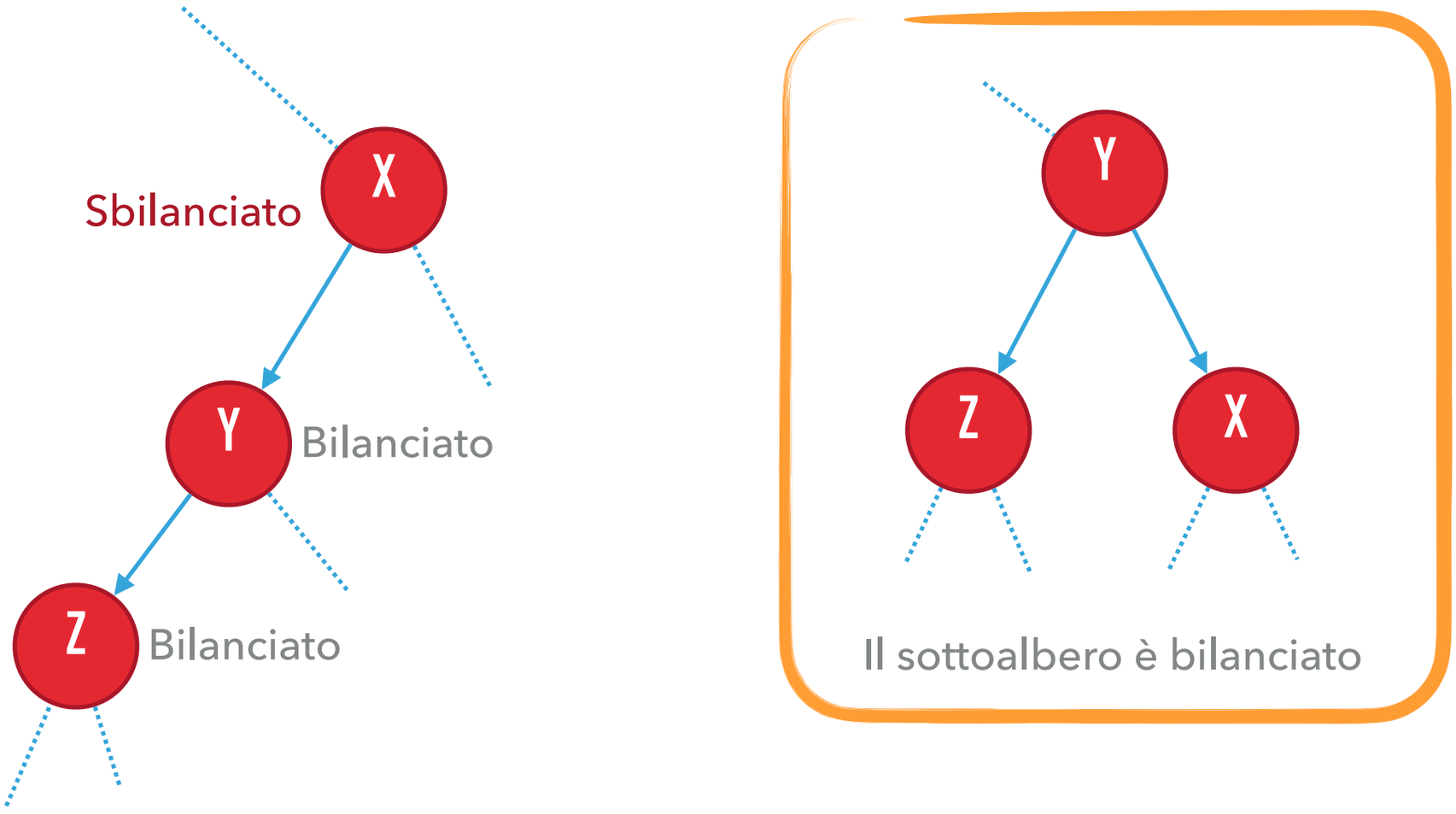


Caso destra-destra



INSERIMENTO

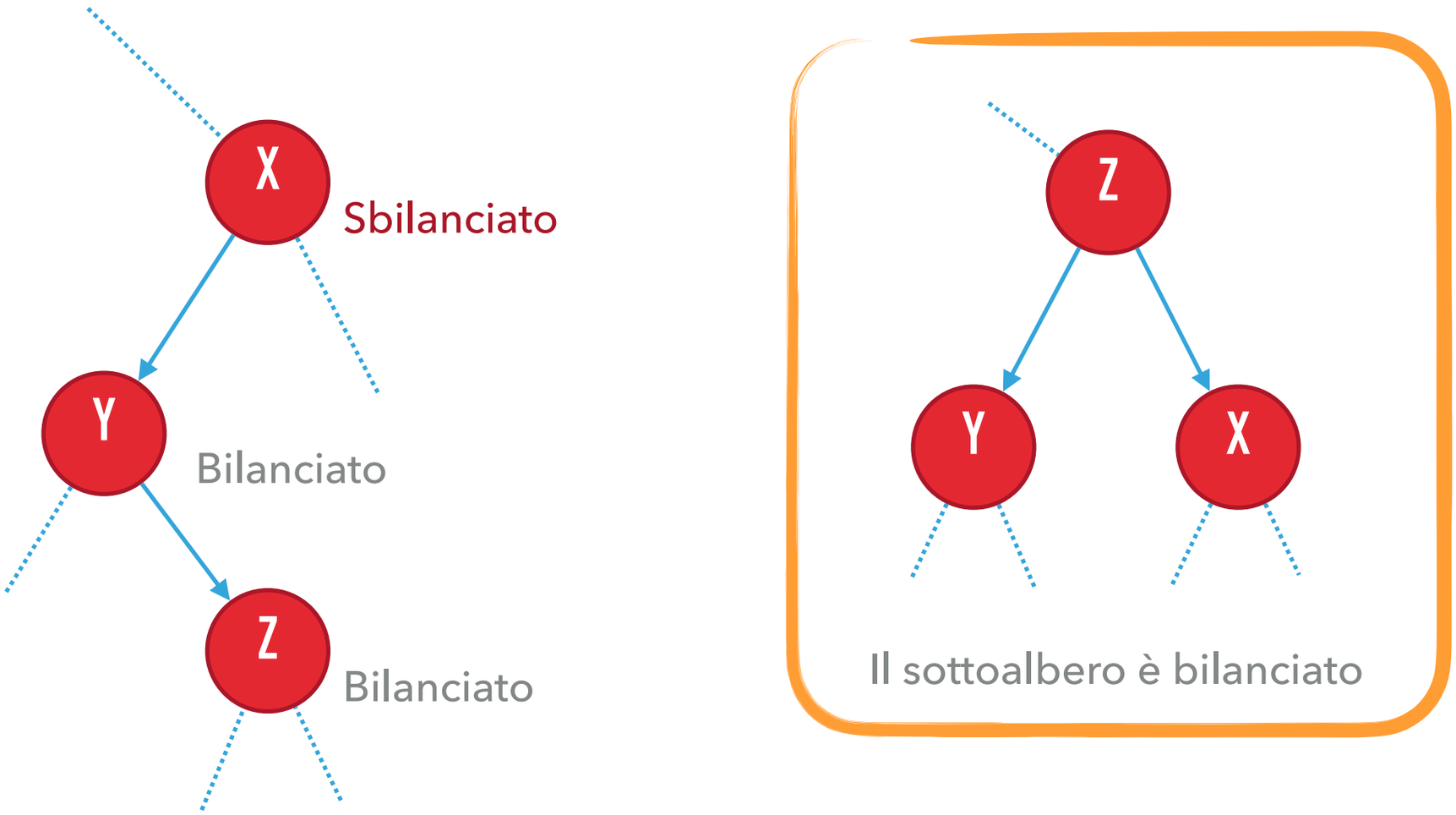
Rotazioni che ri-bilanciano l'albero:



Caso sinistra-sinistra

INSERIMENTO

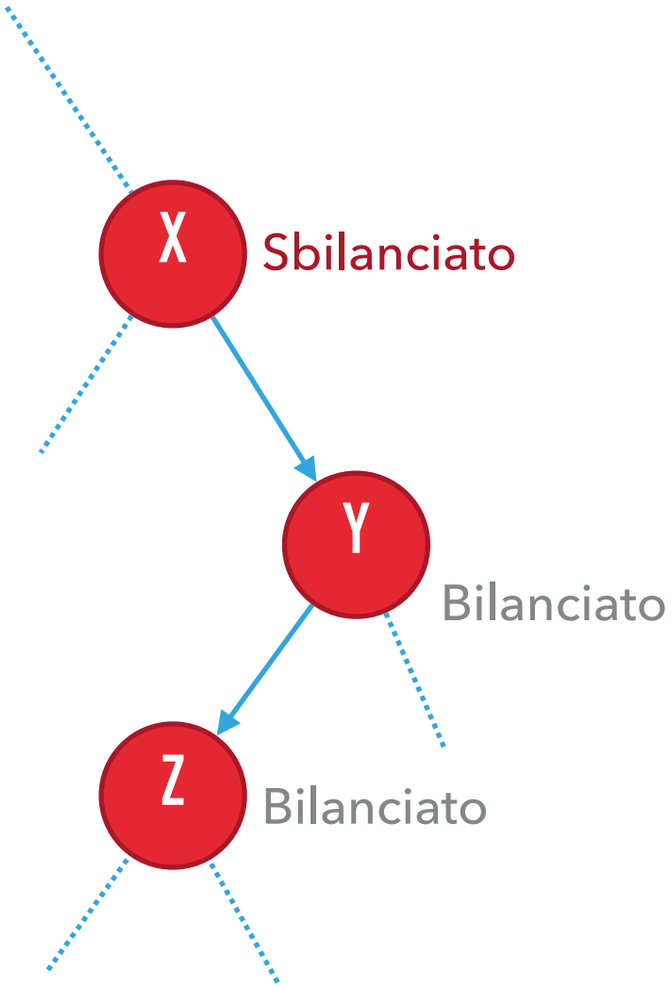
Rotazioni che ri-bilanciano l'albero:

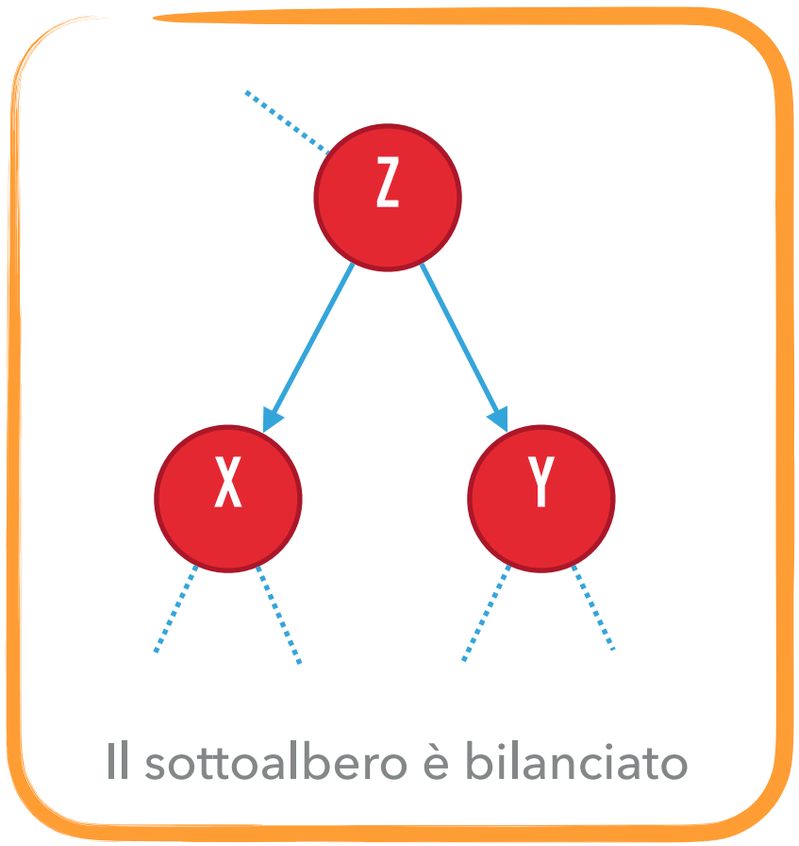


Caso sinistra-destra

INSERIMENTO

Rotazioni che ri-bilanciano l'albero:

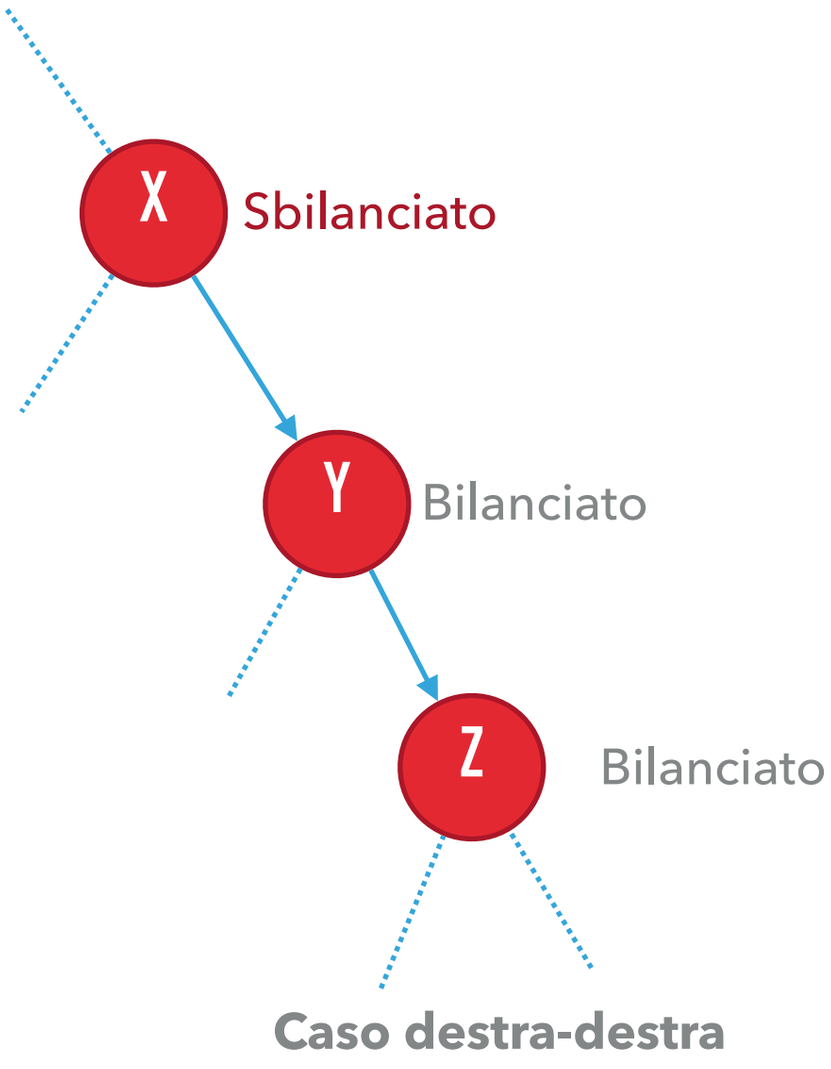




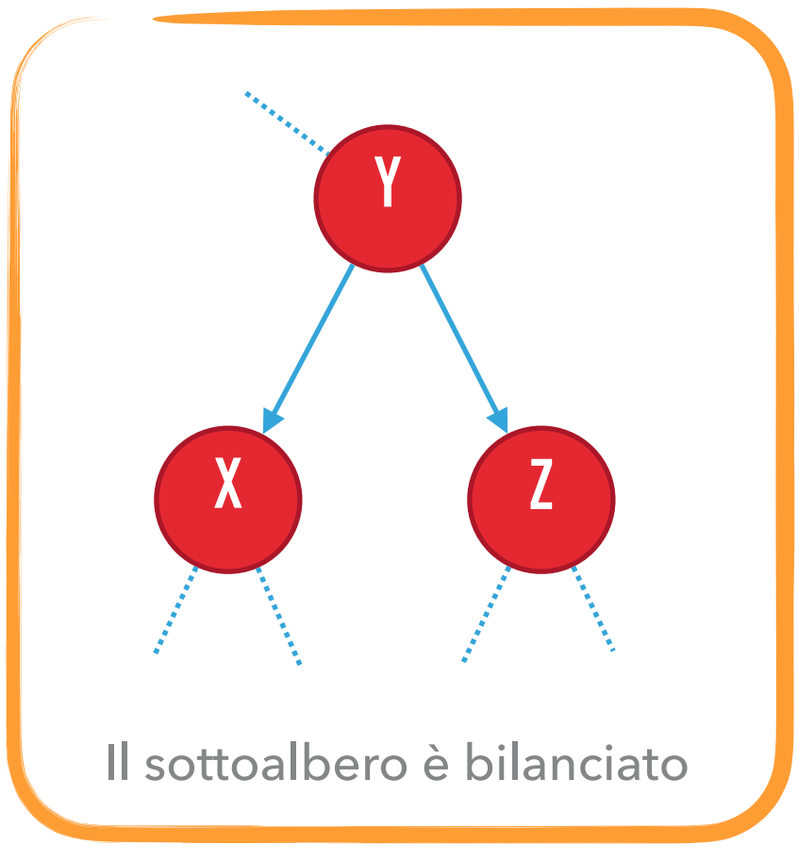
Caso destra-sinistra

INSERIMENTO

Rotazioni che ri-bilanciano l'albero:



Caso destra-destra

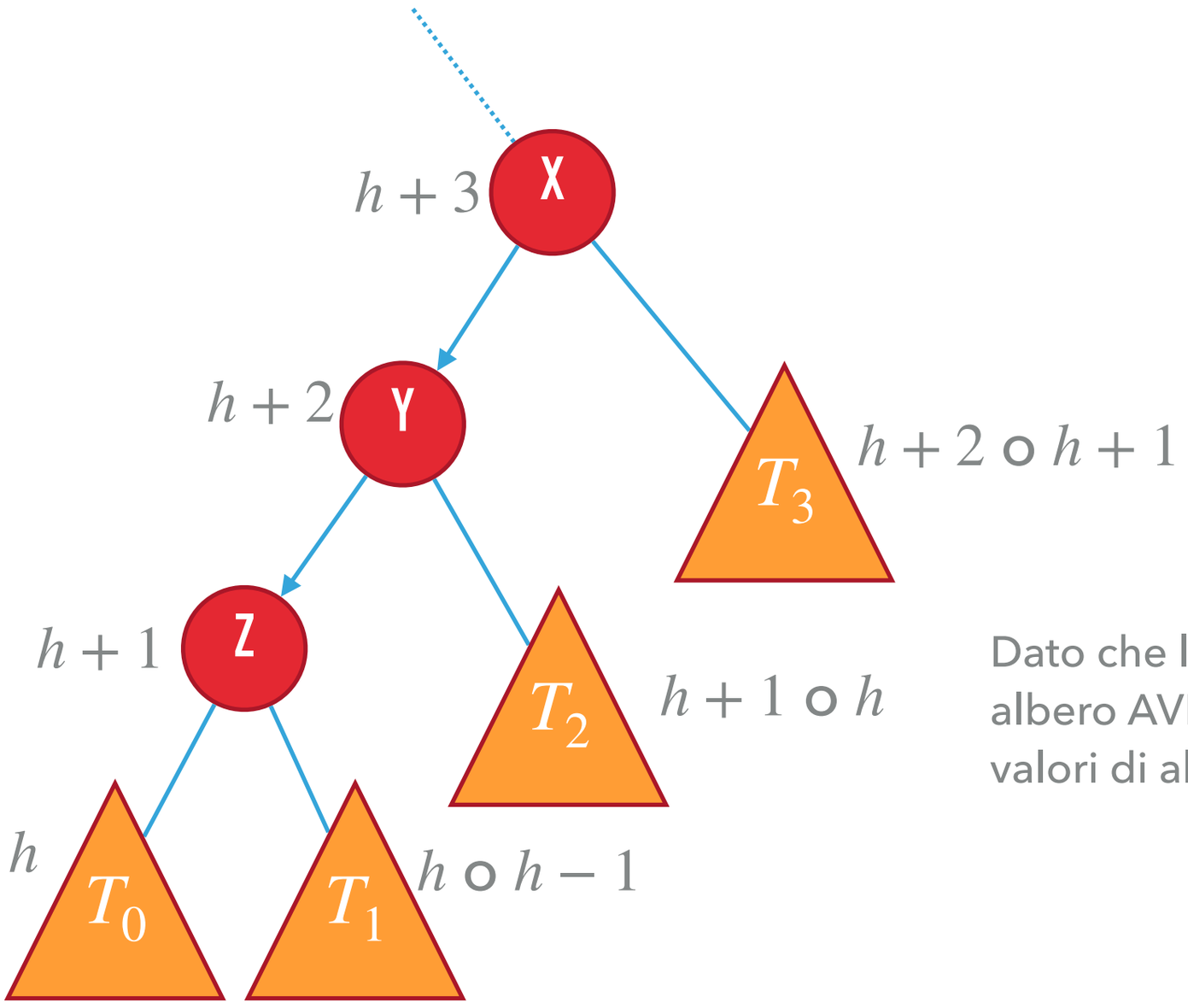


INSERIMENTO

- 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

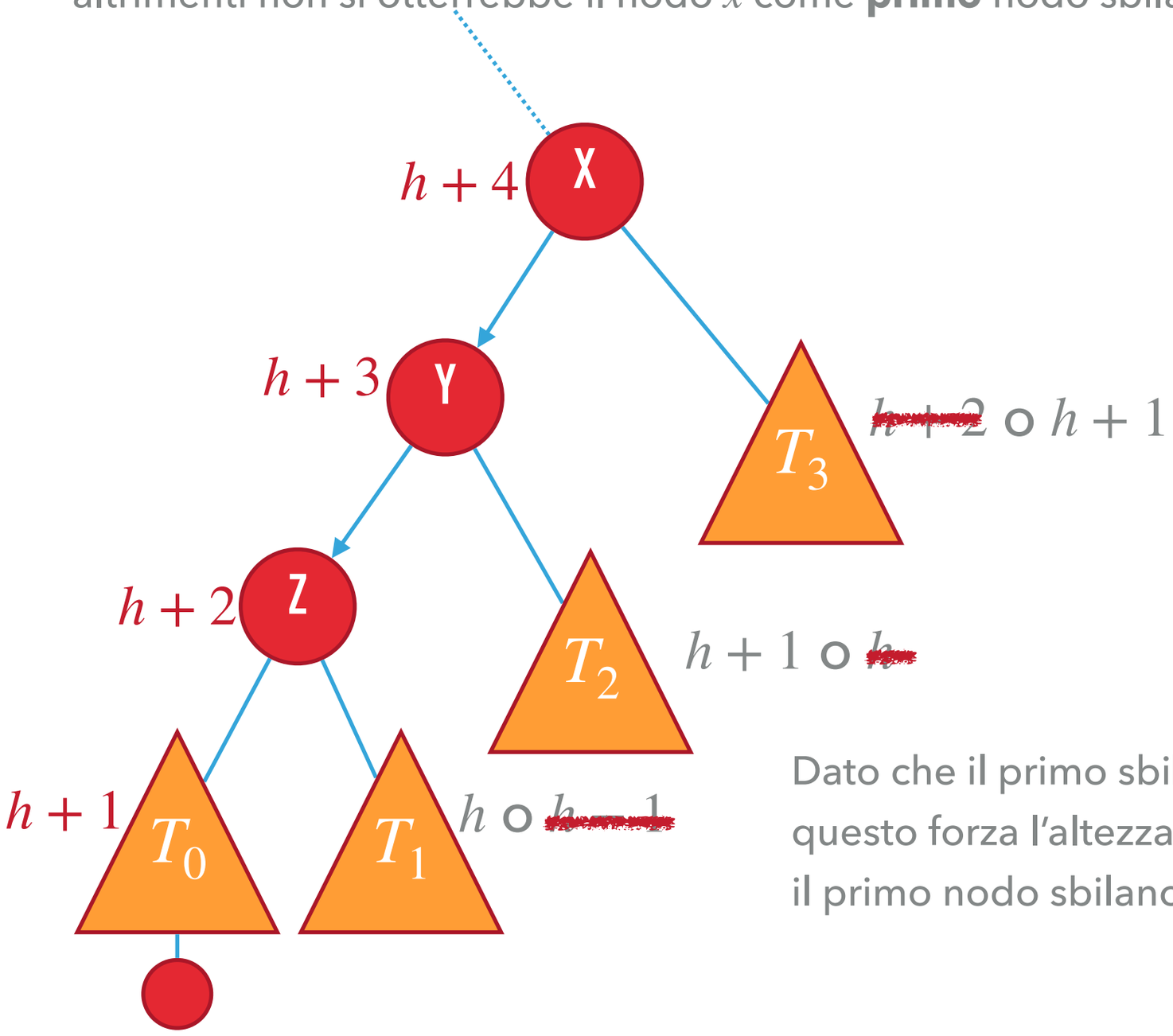
Prima dell'inserimento (supponiamo il nuovo nodo venga inserito in T_0



Dato che l'albero di partenza è un albero AVL abbiamo questi possibili valori di altezze

INSERIMENTO

Dopo l'inserimento sappiamo che alcune altezze si modificano, altrimenti non si otterrebbe il nodo x come **primo** nodo sbilanciato

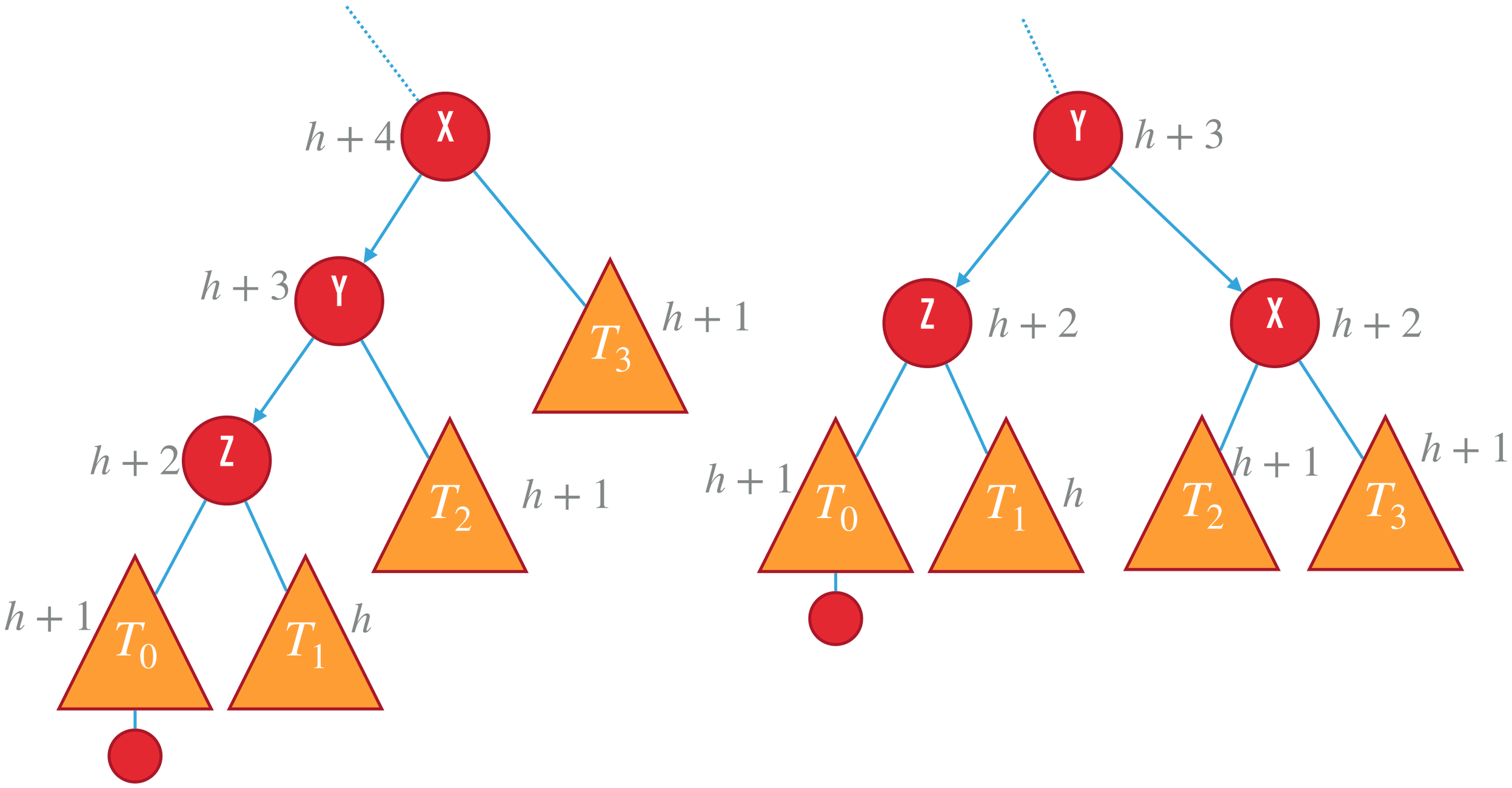


Dato che x è sbilanciato, T_3 doveva avere altezza $h+1$, altrimenti l'inserimento non avrebbe sbilanciato x

Dato che il primo sbilanciamento è in x, questo forza l'altezza di T_1 e T_2. Altrimenti il primo nodo sbilanciato sarebbe z o y

INSERIMENTO

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

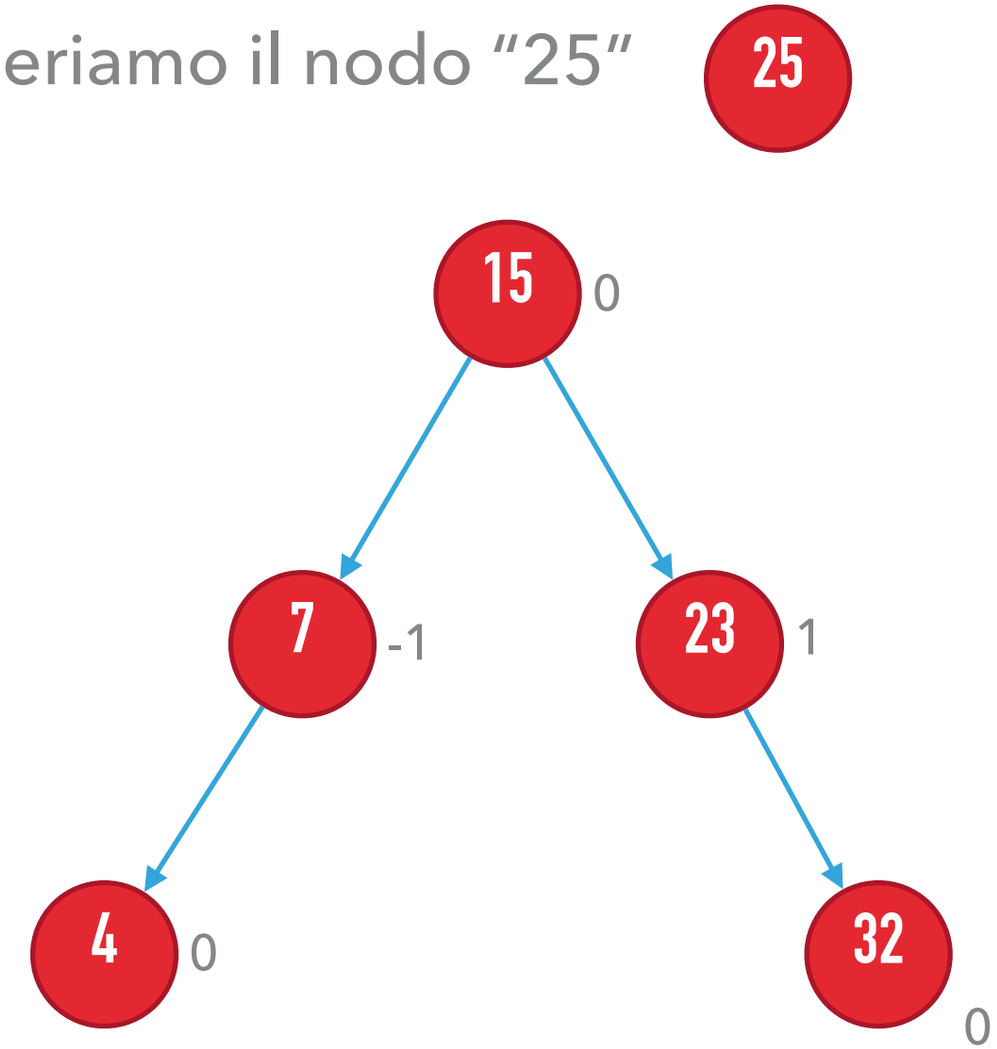


INSERIMENTO

- 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 $h + 3$ dell'albero che **prima** dell'inserimento
- Quindi se i nodi precedenti erano bilanciati prima dell'inserimento lo rimangono anche dopo

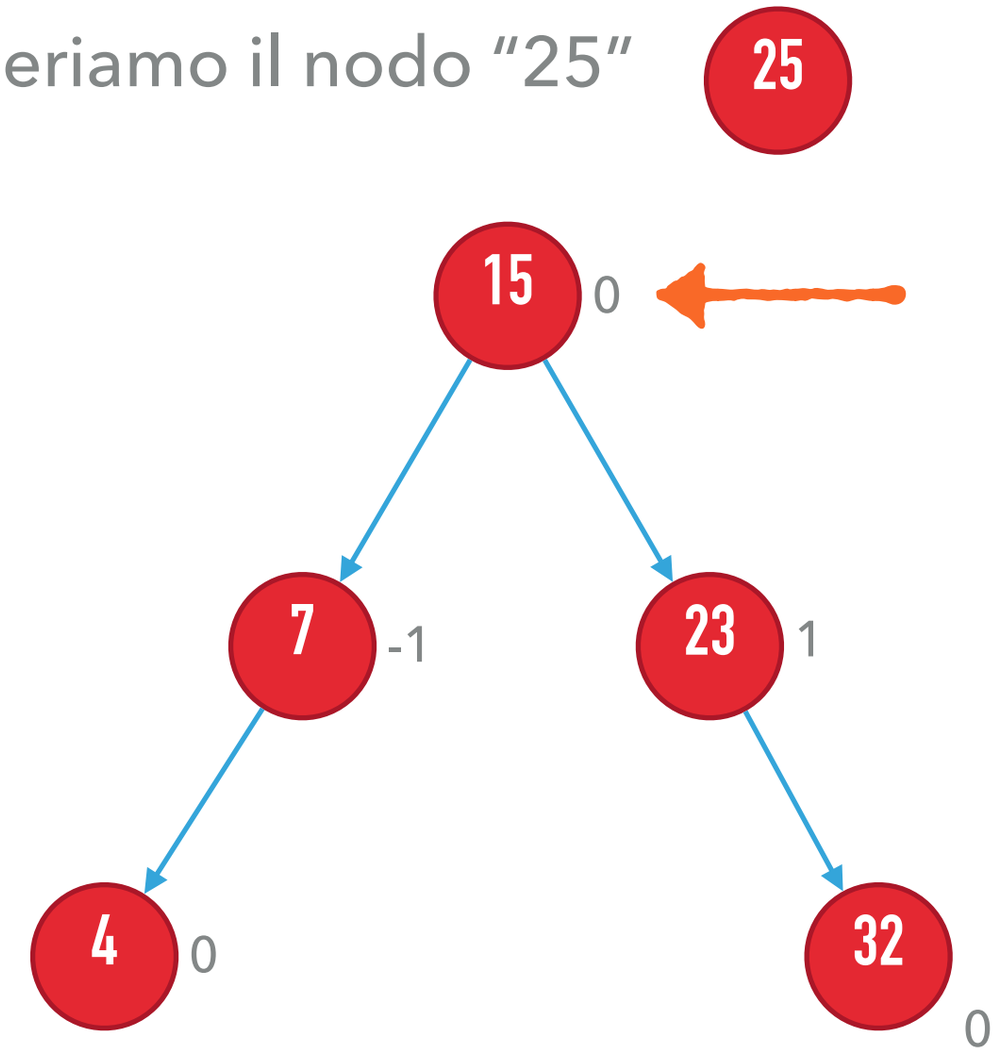
INSERIMENTO

Inseriamo il nodo “25”



INSERIMENTO

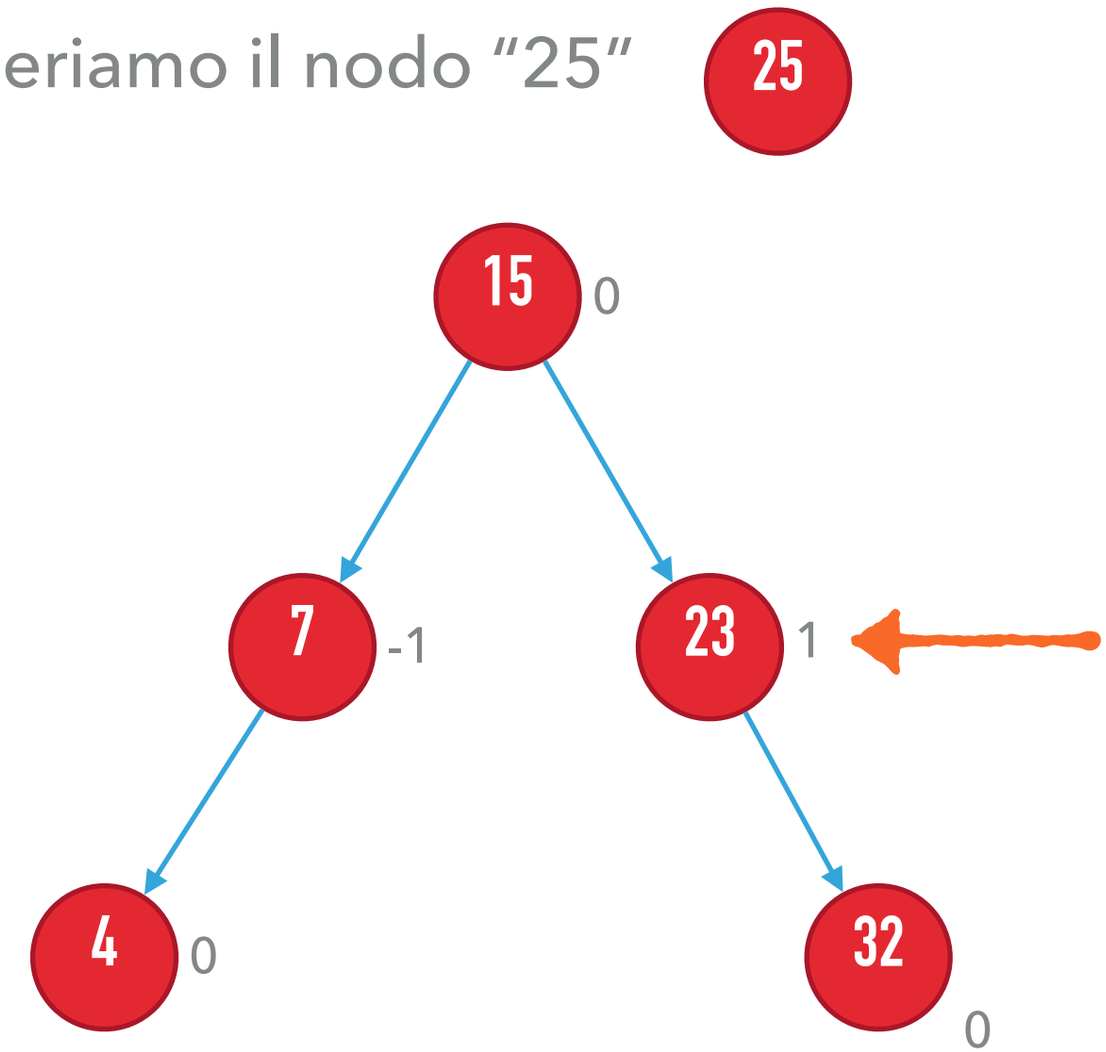
Inseriamo il nodo “25”

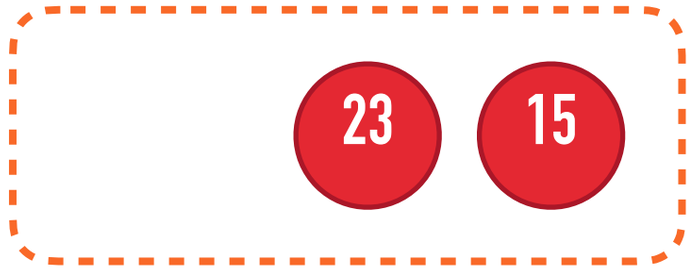


Stack dei nodi visitati

INSERIMENTO

Inseriamo il nodo “25”

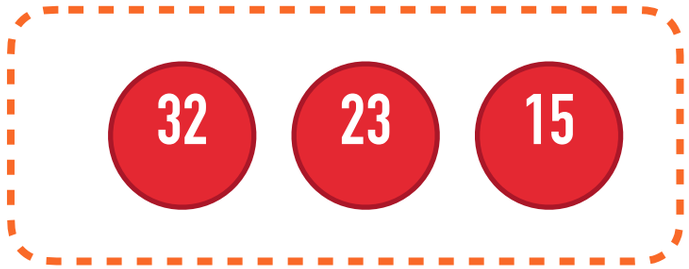




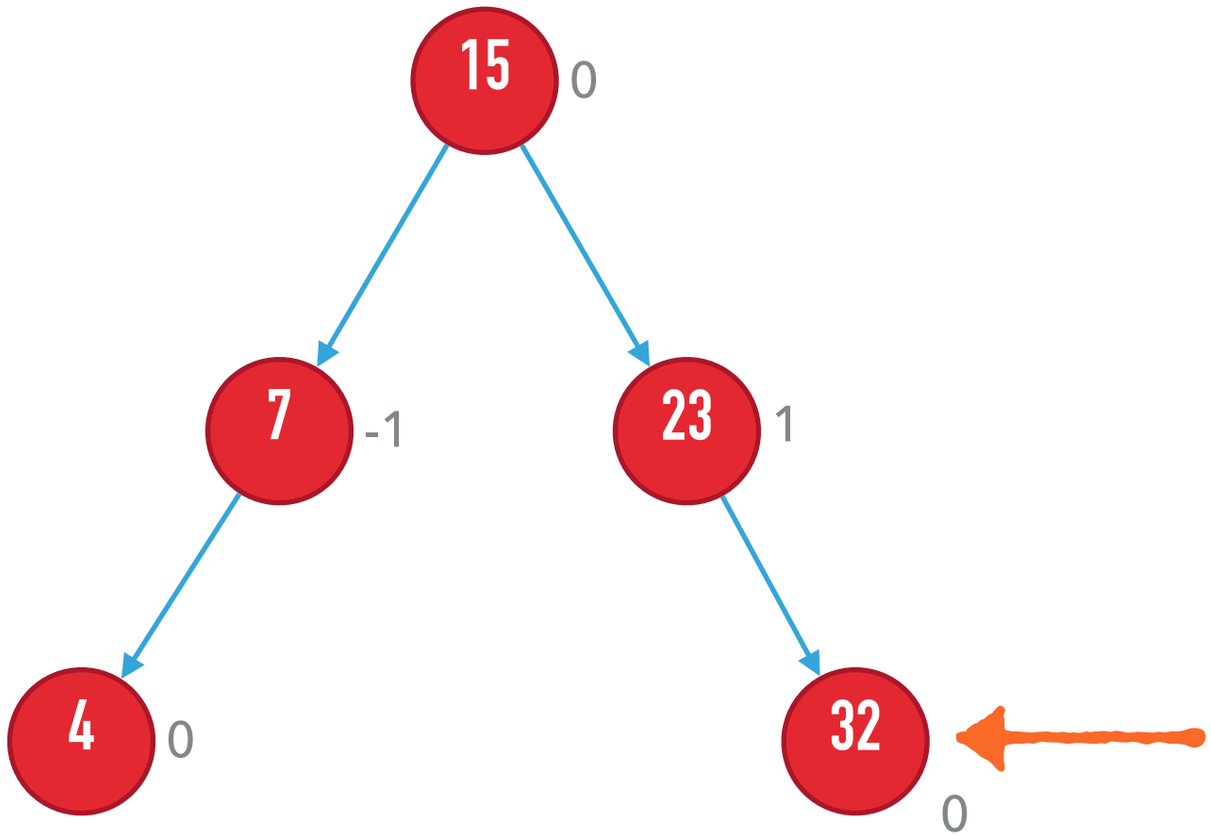
Stack dei nodi visitati

INSERIMENTO

Inseriamo il nodo “25”

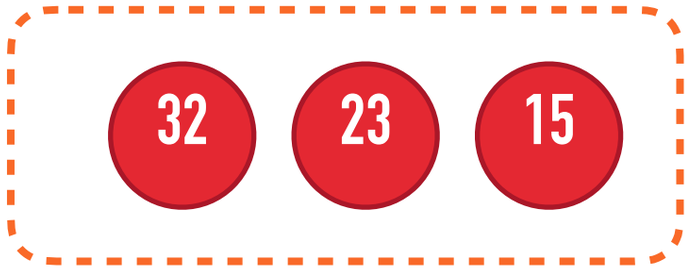


Stack dei nodi visitati

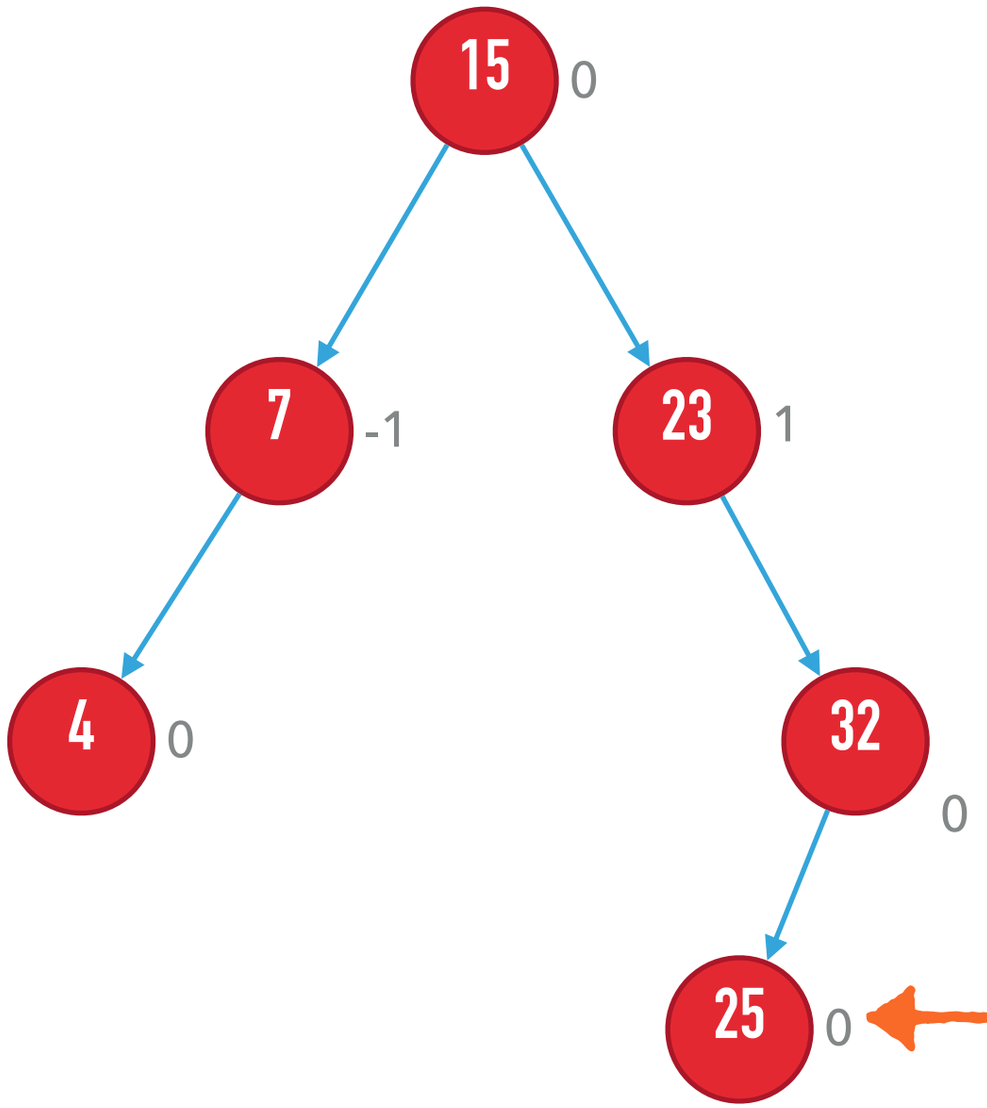


INSERIMENTO

Inseriamo il nodo “25”



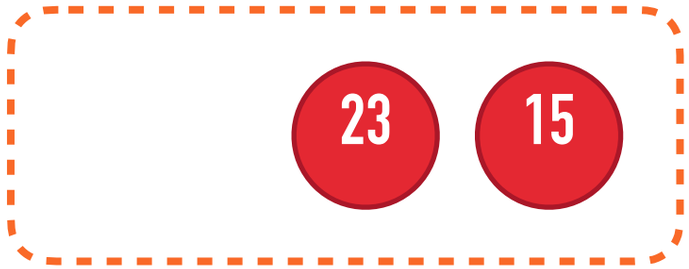
Stack dei nodi visitati



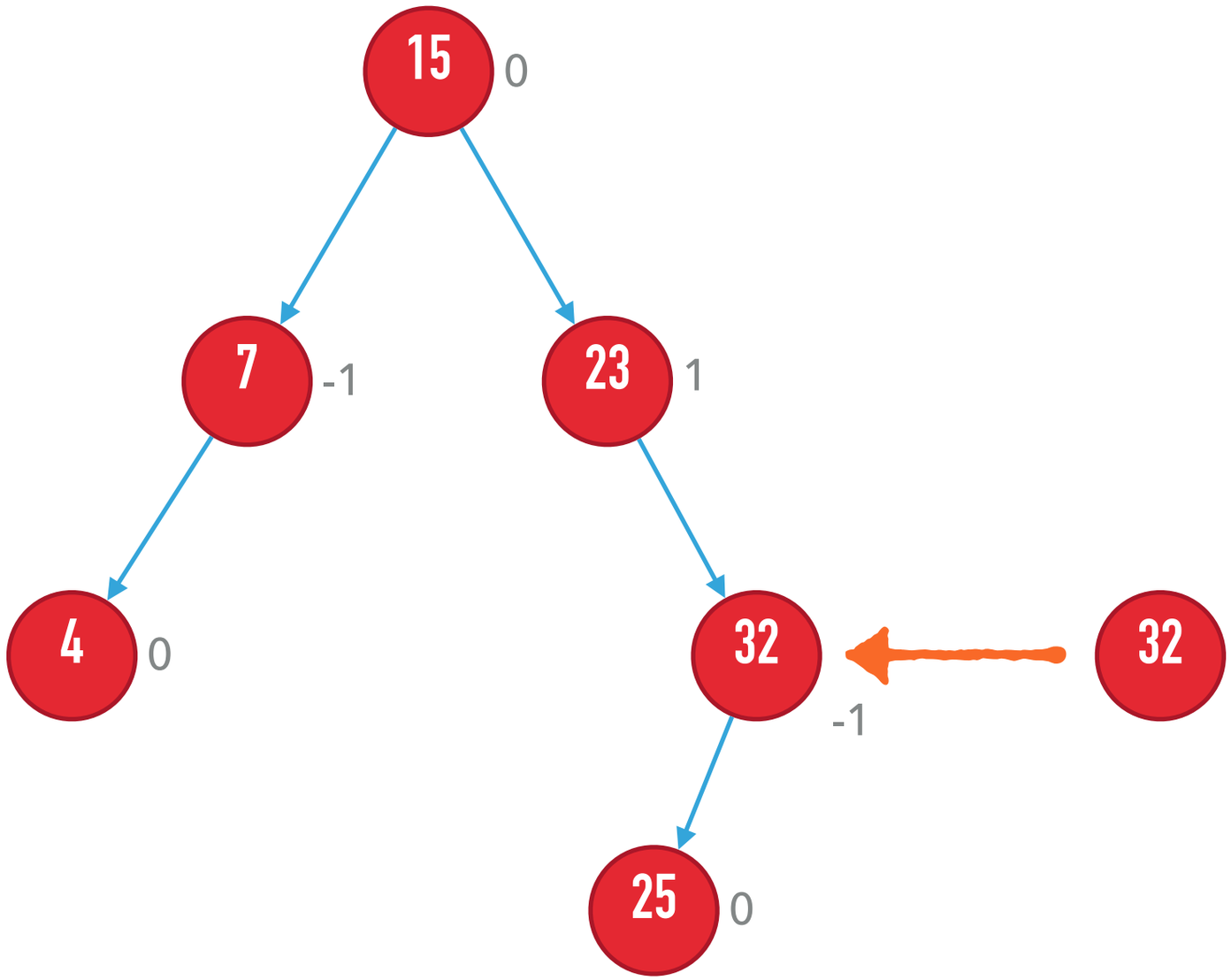
Inseriamo il nodo ed iniziamo ad aggiornare lo sbilanciamento

INSERIMENTO

Inseriamo il nodo “25”



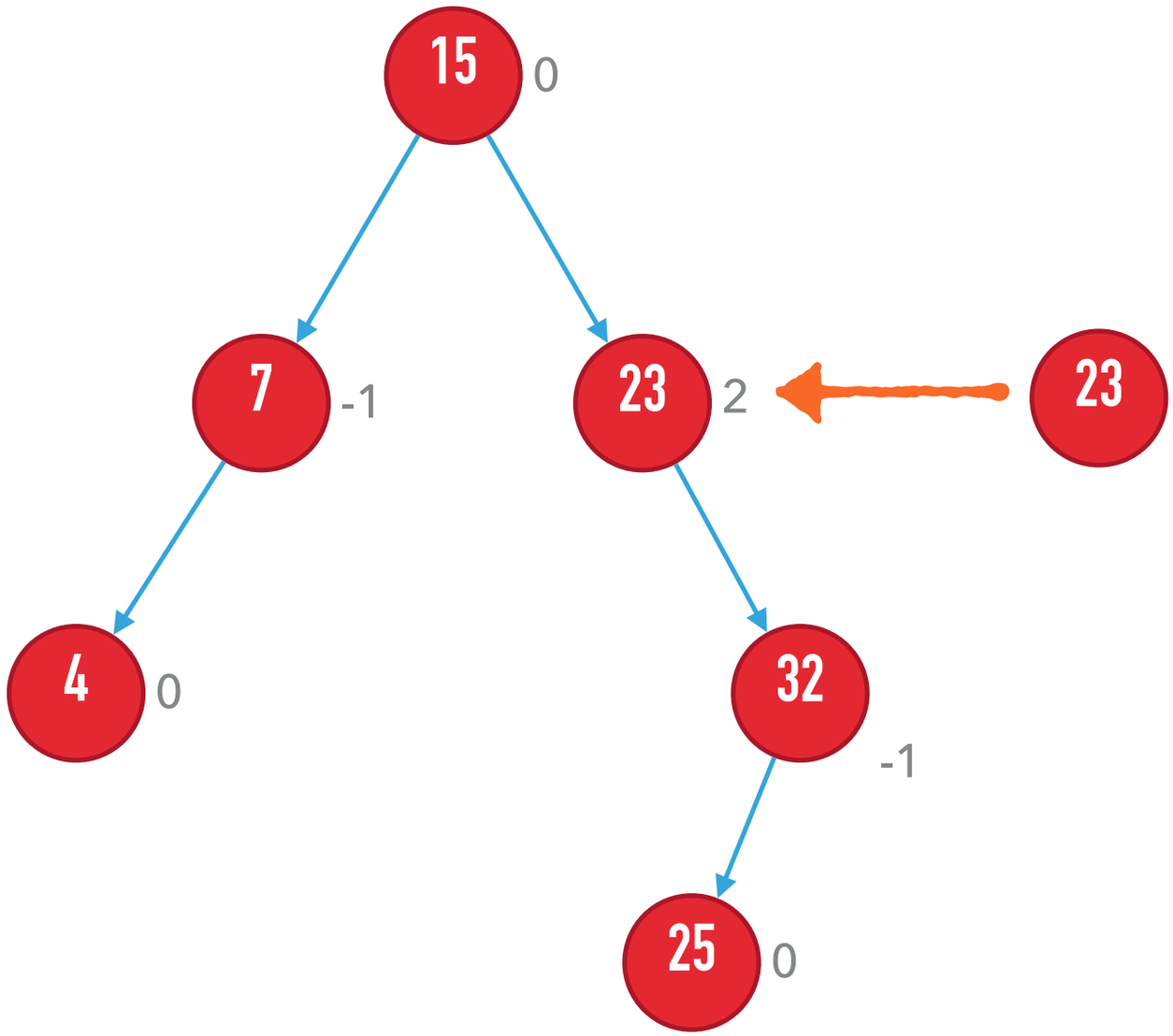
Stack dei nodi visitati



INSERIMENTO

Inseriamo il nodo “25”

Stack dei nodi visitati

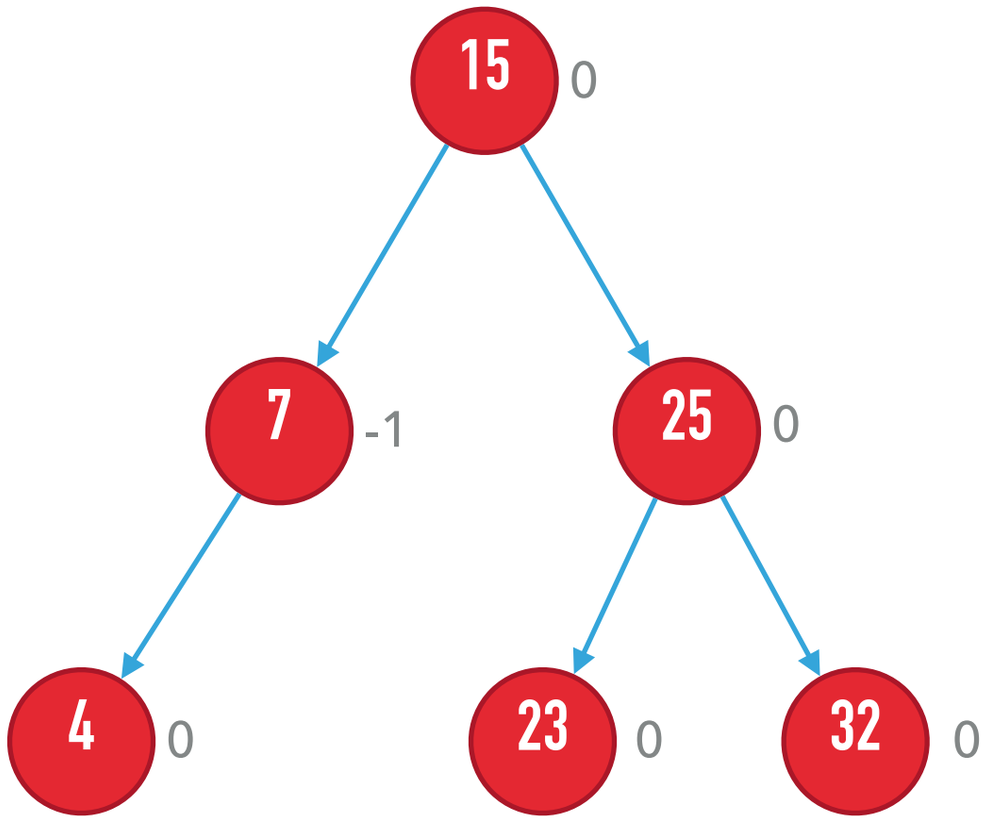


Abbiamo trovato il primo nodo sbilanciato: caso destra-sinistra

INSERIMENTO

Inseriamo il nodo “25”

Stack dei nodi visitati



Abbiamo ribilanciato l’albero

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)

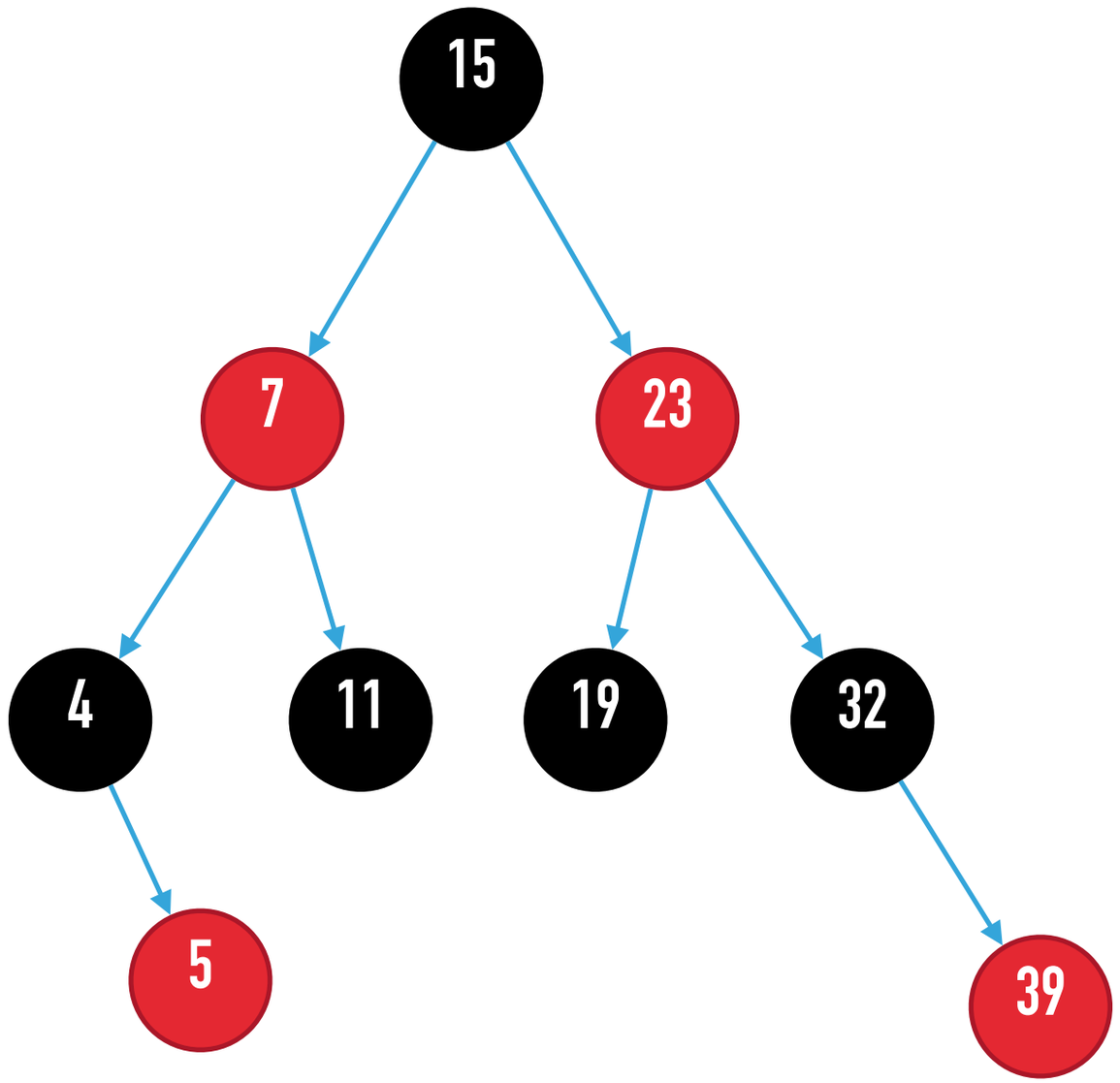
ALTRE TIPOLOGIE DI ALBERI BINARI DI RICERCA BILANCIATI

- Gli alberi AVL non sono l'unico tipo albero binario di ricerca bilanciato
- Un altro tipo di albero che si incontra spesso "in the wild" sono i **red-black tree** o alberi rosso-neri (no, niente a che vedere con squadre di calcio)
- Idea: ogni nodo contiene un bit di informazione aggiuntivo che dice se il nodo è colorato di nero o di rosso

PROPRIETÀ DEGLI ALBERI ROSSO-NERI

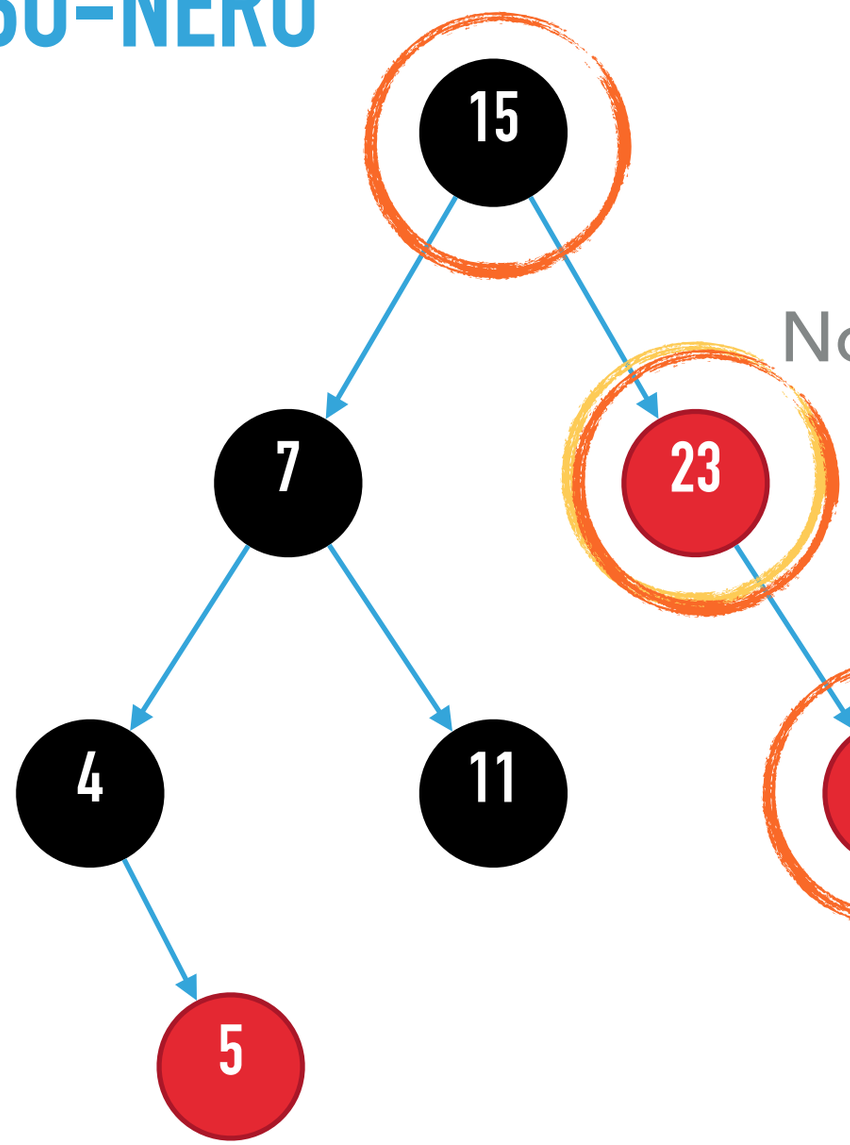
- Ogni nodo è colorato di rosso o di nero
- La radice deve essere colorata di nero
- I valori None si considerano colorati di nero
- I figli di un nodo rosso devono essere nodi neri
- Ogni percorso da un qualsiasi nodo alle foglie (o ai nodi con un reference a None) attraversa lo stesso numero di nodi neri

ALBERO ROSSO-NERO



Questo albero rispetta tutte le proprietà di albero rosso-nero

ALBERO ROSSO-NERO



Nodo rosso con figlio rosso

Numero differente di nodi neri per arrivare alle foglie o ai nodi con reference a None

Questo albero NON rispetta
tutte le proprietà di albero rosso-nero

ALBERI ROSSO-NERI: INSERIMENTO

- L'inserimento negli alberi rosso-neri inizia come per gli alberi binari di ricerca normali
- Il nodo da inserire viene colorato di rosso
- Se una proprietà degli alberi rosso-neri viene violata si eseguono una serie di ricolorazioni e rotazioni per ripristinarla (sempre in tempo $O(\log n)$)

ALBERI ROSSO-NERI: NODI CONTENUTI

- Mostriamo ora che un albero rosso-nero con n nodi ha altezza $O(\log n)$
- Dato che ogni nodo negli alberi rosso-neri ha lo stesso numero di nodi neri da lui stesso alle foglie e ai nodi con reference a None, possiamo definire la black height del nodo
- Dato un nodo x, definiamo $bh(x)$ il numero di nodi neri che si devono attraversare per arrivare da x a una qualsiasi foglia e nodo con reference a None

ALBERI ROSSO-NERI: NODI CONTENUTI

Mostriamo per induzione che ogni sottoalbero avente il nodo x come radice contiene almeno $2^{bh(x)-1}$ nodi interni (i.e., foglie escluse).

La proprietà è vera è vero se x è una foglia: $bh(x) = 0$, e contiene $2^{bh(x)} - 1 = 2^0 - 1 = 0$ nodi interni.

Sia x un nodo con $bh(x) > 0$. Questo significa che ha due figli di altezza $bh(x)$ o $bh(x) - 1$ a seconda del colore.

ALBERI ROSSO-NERI: NODI CONTENUTI

Per ipotesi induttiva ciascuno dei sottoalberi ha almeno $2^{bh(x)-1} - 1$ nodi interni.

Quindi x ha almeno $(2^{bh(x)-1} - 1) + (2^{bh(x)-1} - 1) + 1$, ovvero $2^{bh(x)} - 1$ nodi interni, che è quello che volevamo provare.

Consideriamo ora l'altezza h della radice r. Dato che la non possono esserci due nodi consecutivi rossi e che la radice è nera, essa ha $bh(r) \geq h/2$

ALBERI ROSSO-NERI: NODI CONTENUTI

Ma, per la proprietà precedente, l'intero albero può contenere al più $2^{bh(r)} - 1 = 2^{h/2} - 1$ nodi.

Da questo deriviamo

$$n \geq 2^{h/2} - 1$$
$$\log_2(n-1) \geq \log_2(2^{h/2})$$
$$2\log_2(n-1) \geq h$$

E quindi $h \leq 2\log_2(n-1)$, quindi l'altezza dell'albero è al più due volte $\log_2(n-1)$, quindi $O(\log n)$.