ALBERI AVL
ALBERI ROSSO NERI
DYNAMIC SELECT

ALGORITMI E STRUTTURE DATI

ALBERI AVL

COSA SONO GLI ALBERI AVL

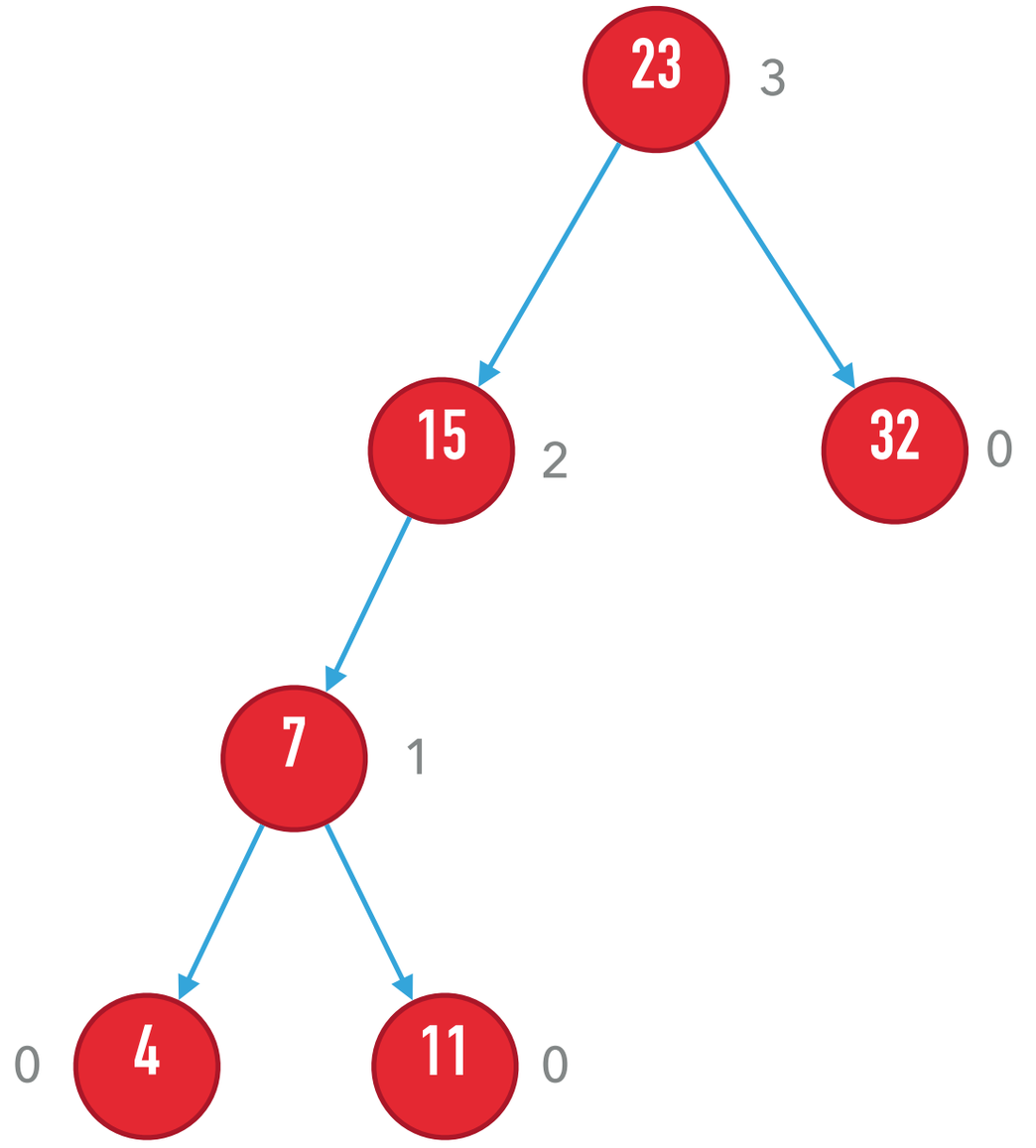
- AVL da Adelson-Velsky e Landis, cognomi dei due ideatori (anno 1962)
- Sono alberi binari di ricerca che sono sempre bilanciati
- Ove bilanciati non significa esattamente profondità $\log n$, ma $O(\log n)$
- Nel caso degli alberi AVL la costante del $\log n$ è bassa

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

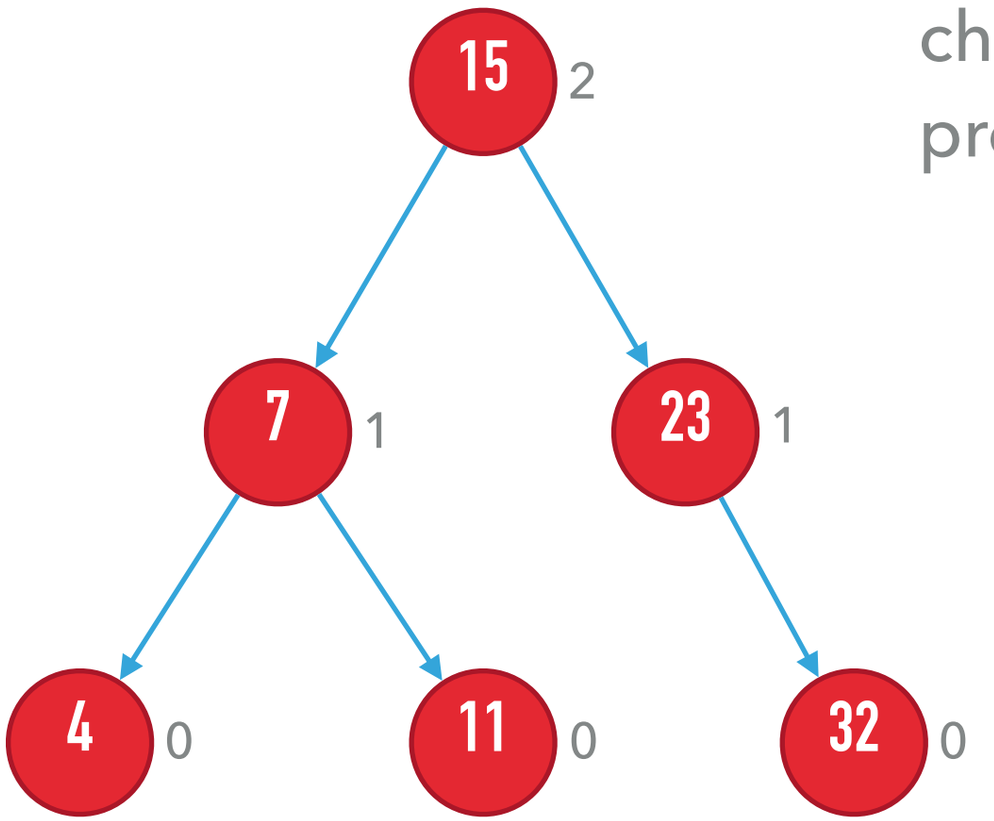


Possiamo facilmente calcolarla per ogni nodo come
1 + max{altezza del figlio sinistro, altezza del figlio destro}

PROPRIETÀ DEGLI ALBERI AVL

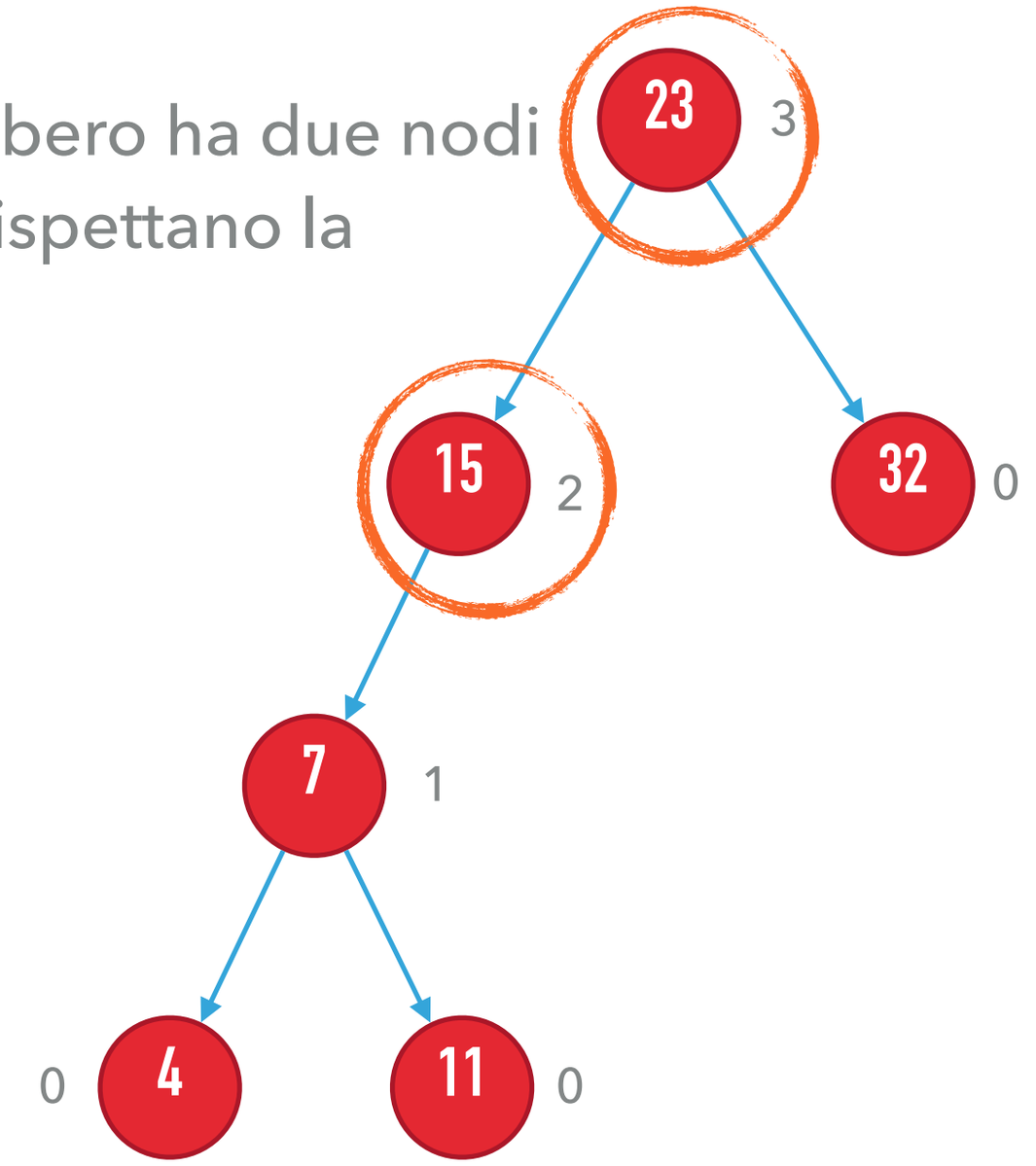
- Gli alberi AVL hanno la proprietà che la differenza in altezza del figlio destro e del figlio sinistro di ogni nodo è al più 1 (in valore assoluto)
- In caso di figlio assente si assume altezza di -1

DIFFERENZA IN ALTEZZA

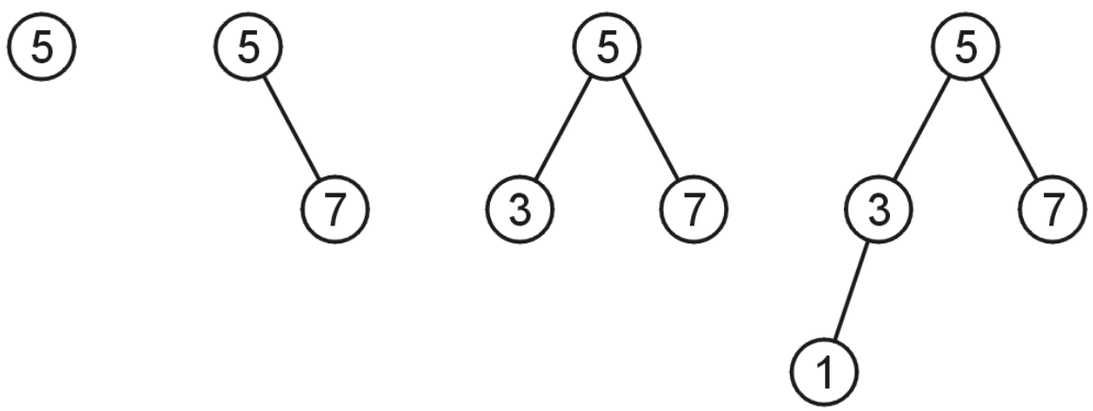


Questo albero rispetta la proprietà
(tutte le differenze sono zero o uno)

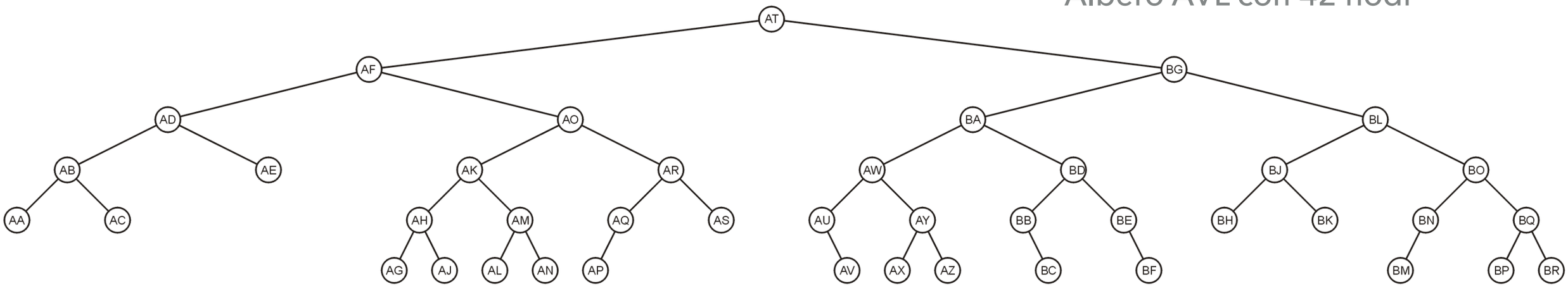
Questo albero ha due nodi che non rispettano la proprietà



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: $h \leq \frac{\log_2(\sqrt{5}(N_h + 1.45))}{\log_2 \phi} - 3$

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

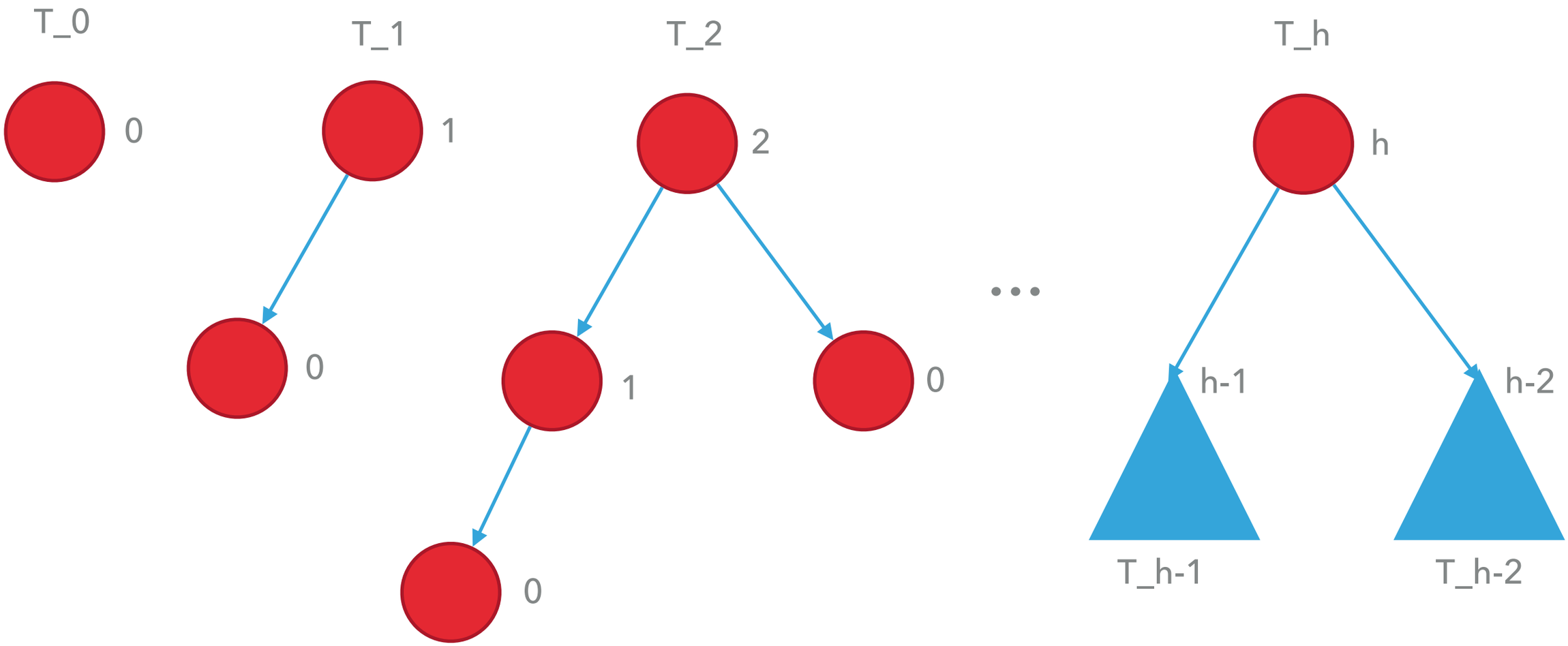
QUANTI NODI CONTIENE UN ALBERO AVL

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.

ALBERI DI FIBONACCI



Sono gli alberi AVL con meno nodi.

COME GARANTIRE IL BILANCIAMENTO

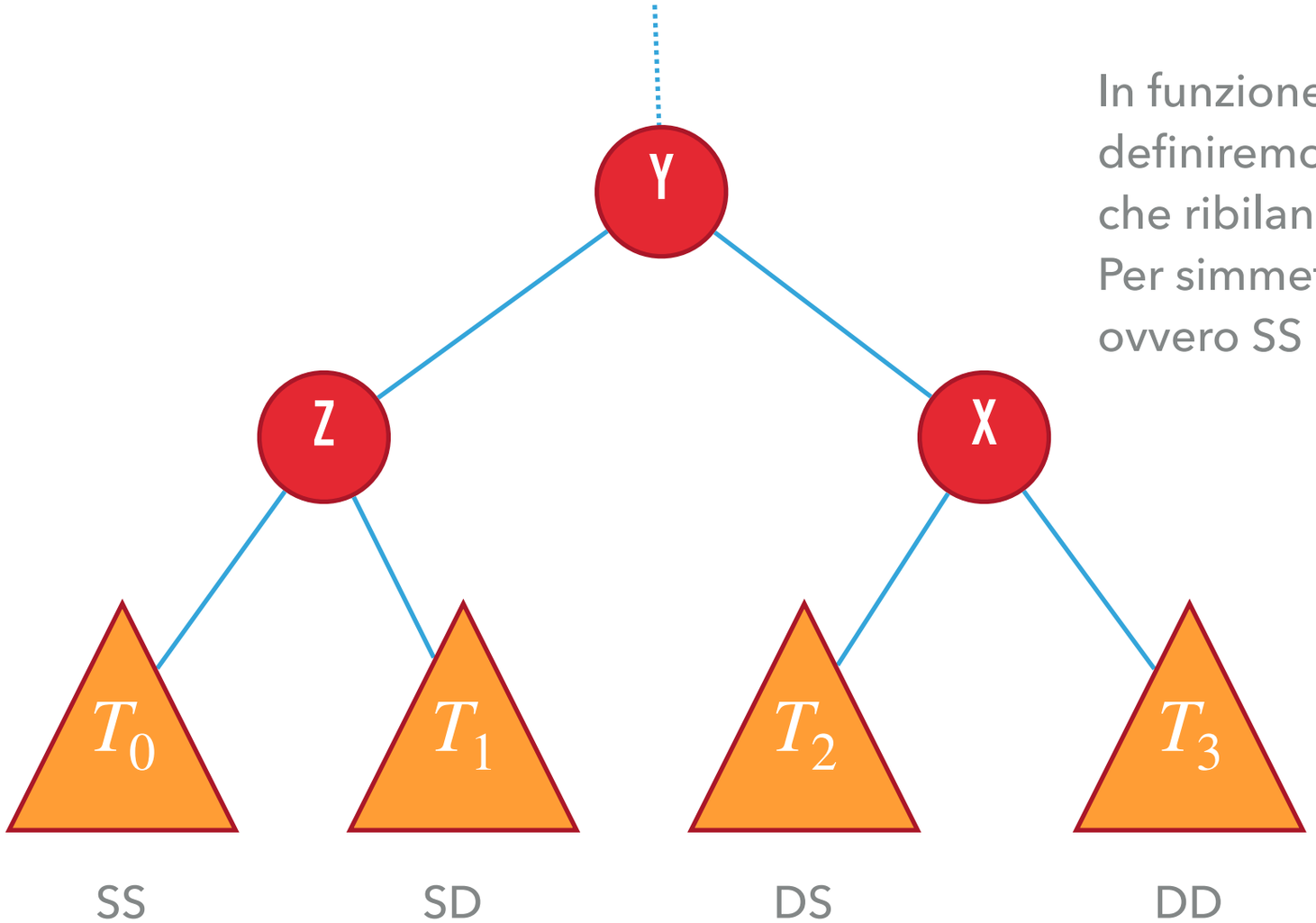
- Per garantire il bilanciamento dobbiamo memorizzare dell'informazione all'interno dei nodi.
- Due possibilità:
  - Altezza del nodo
  - Quale è lo sbilanciamento: $\{-1,0,1\}$
    - Il valore rappresenta la differenza tra altezza del figlio destro e altezza del figlio sinistro

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

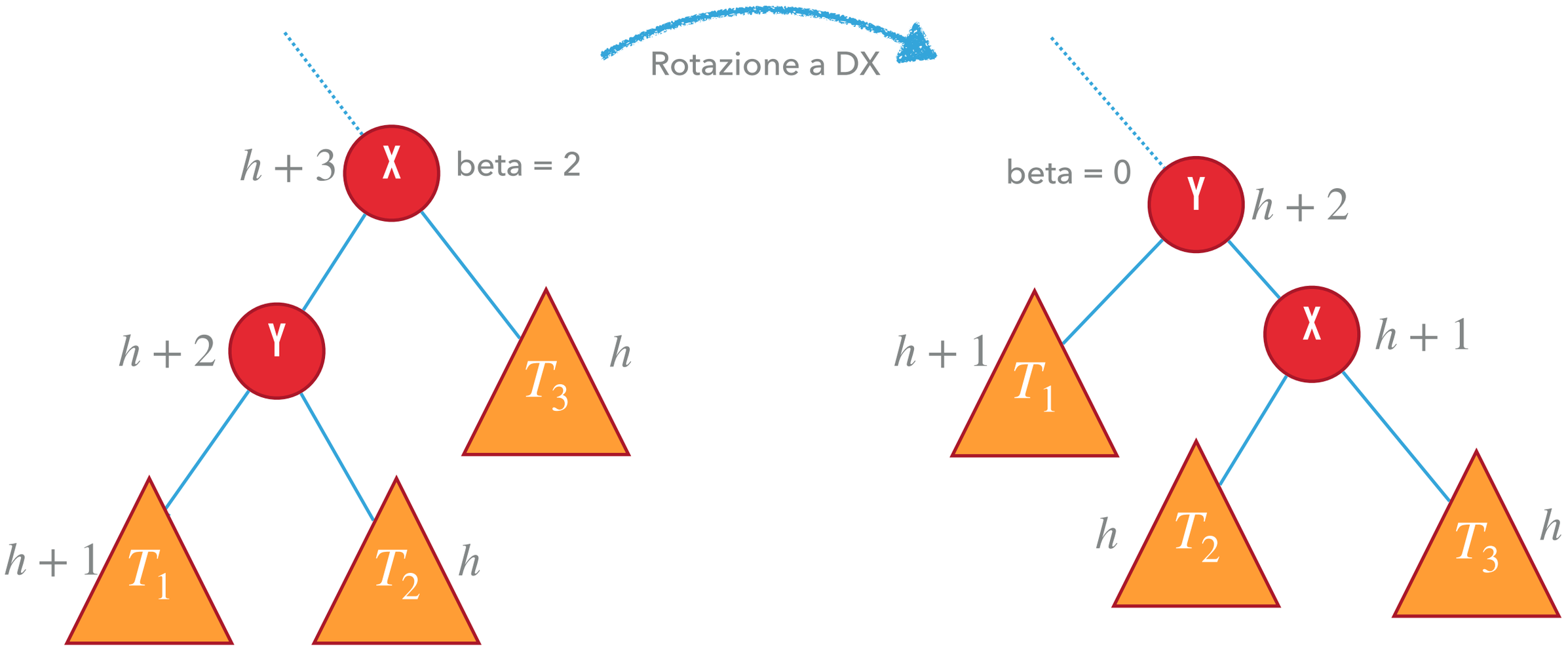
Se emerge uno sbilanciamento dopo un inserimento o una cancellazione di un nodo, allora il fattore di sbilanciamento sarà 2 o -2, e causato da un nodo in più (o in meno) in uno dei 4 sottoalberi T0 T1 T2 T3.



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

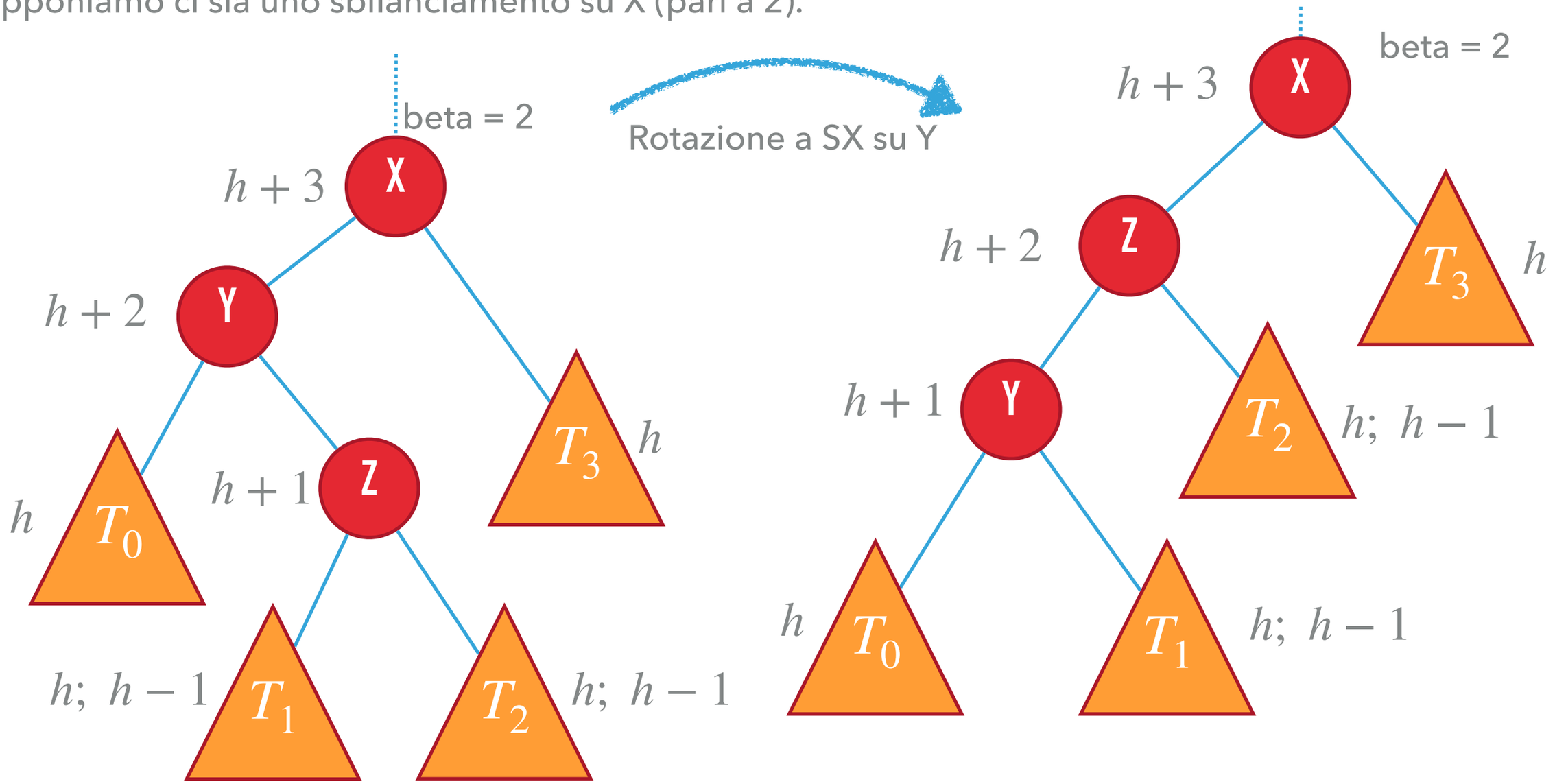
Supponiamo ci sia uno sbilanciamento su X (pari a 2) per la differenza tra le altezze di T1 e T3.



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

RIBILANCIAMENTO – CASO SD

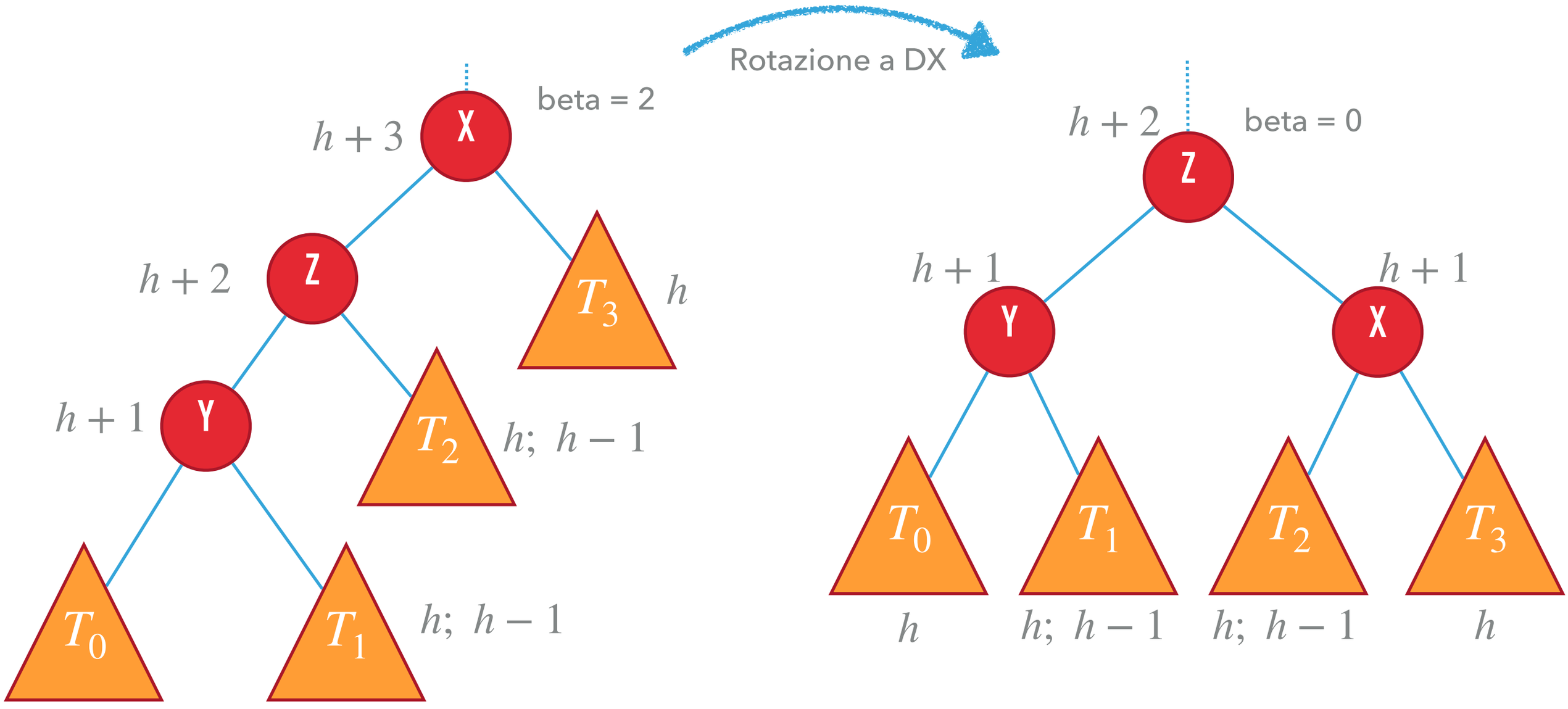
Supponiamo ci sia uno sbilanciamento su X (pari a 2).



Solo uno tra T1 e T2 ha altezza h (causa inserimento)

La prima rotazione non risolve lo sbilanciamento

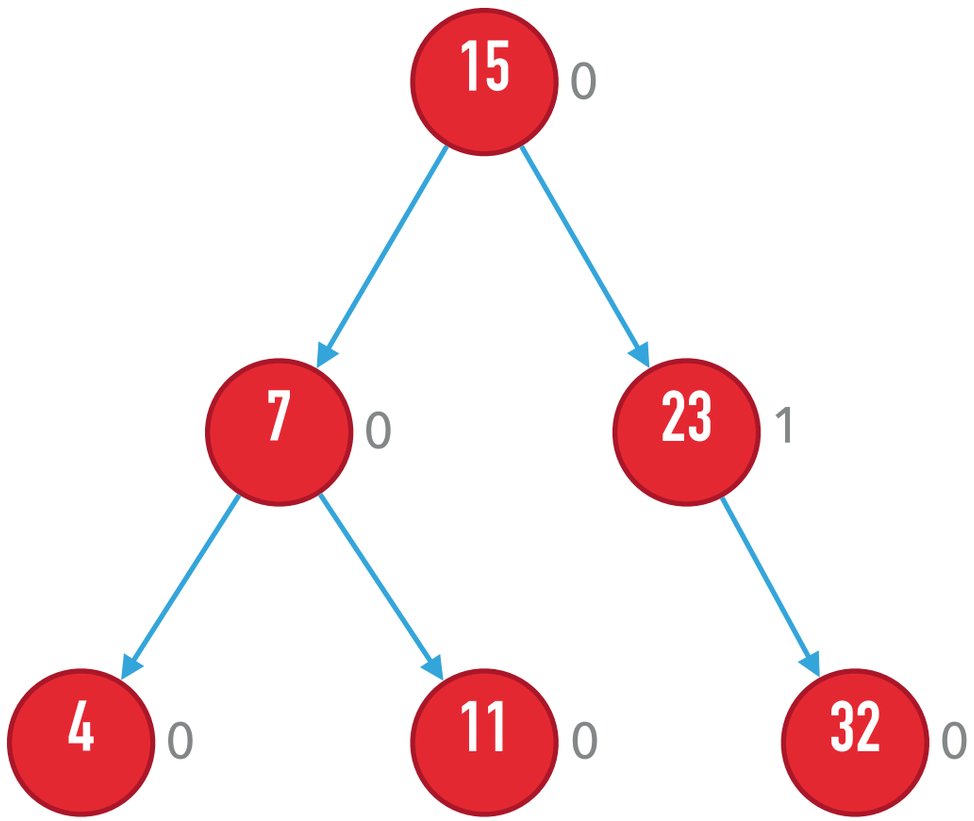
RIBILANCIAMENTO – CASO SD



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

INSERIMENTO

Inseriamo il nodo “19” 19

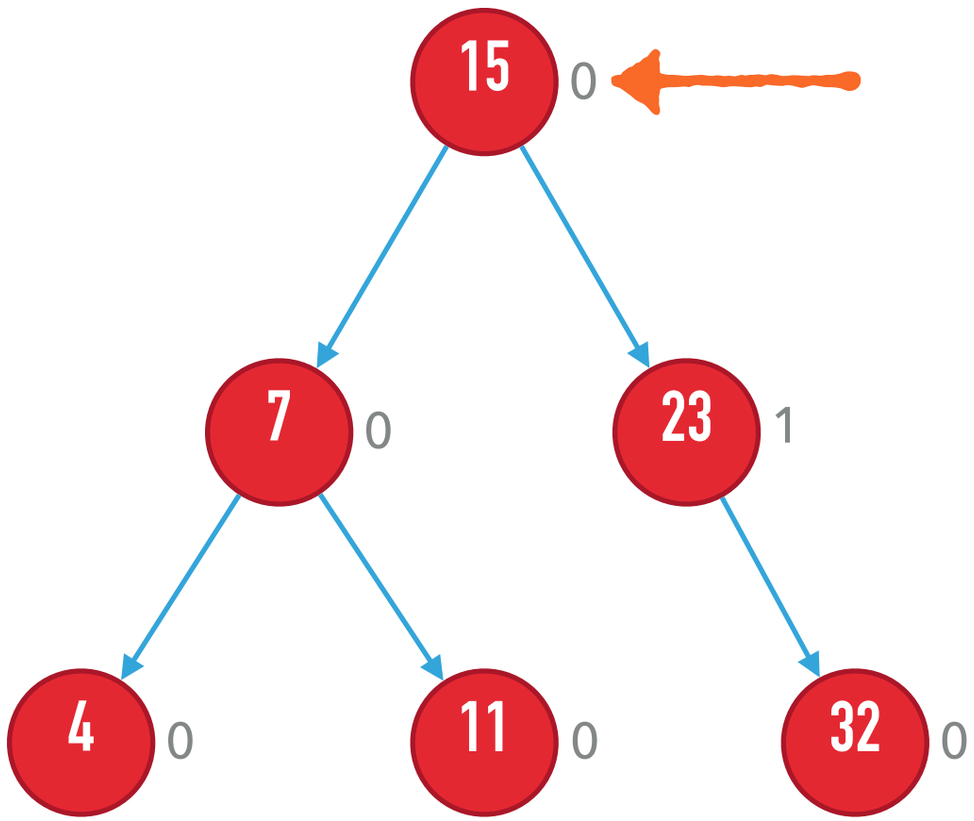


INSERIMENTO

Inseriamo il nodo “19” 19

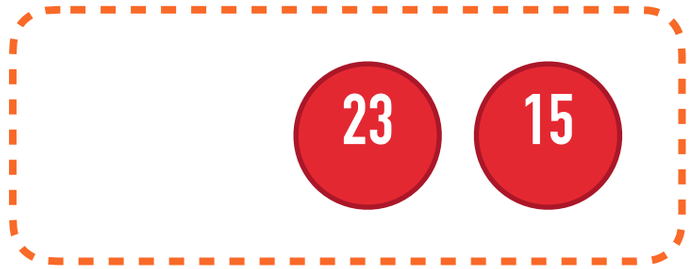
15

Stack dei nodi visitati

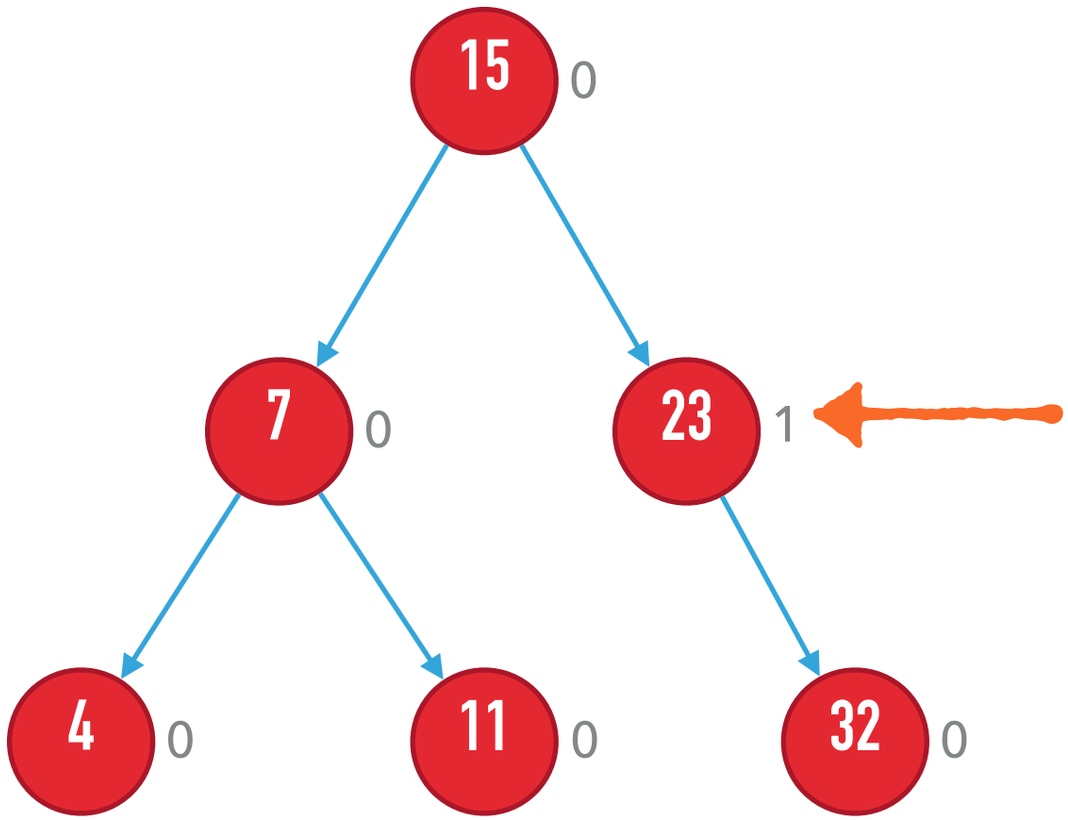


INSERIMENTO

Inseriamo il nodo “19”

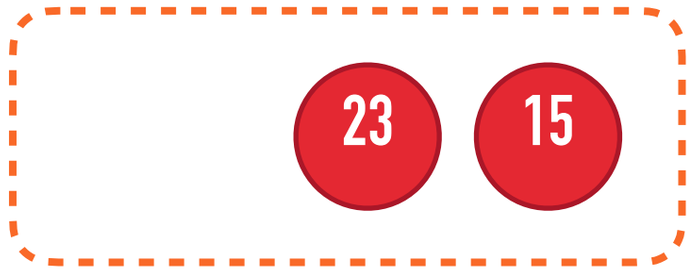


Stack dei nodi visitati

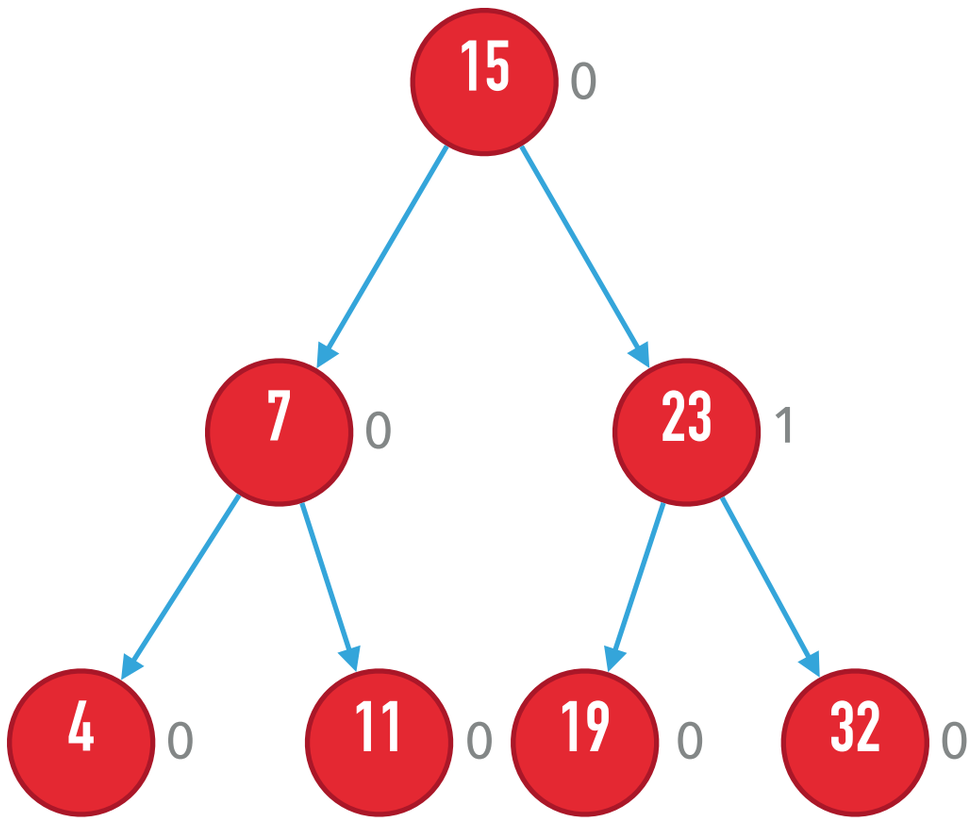


INSERIMENTO

Inseriamo il nodo “19”



Stack dei nodi visitati

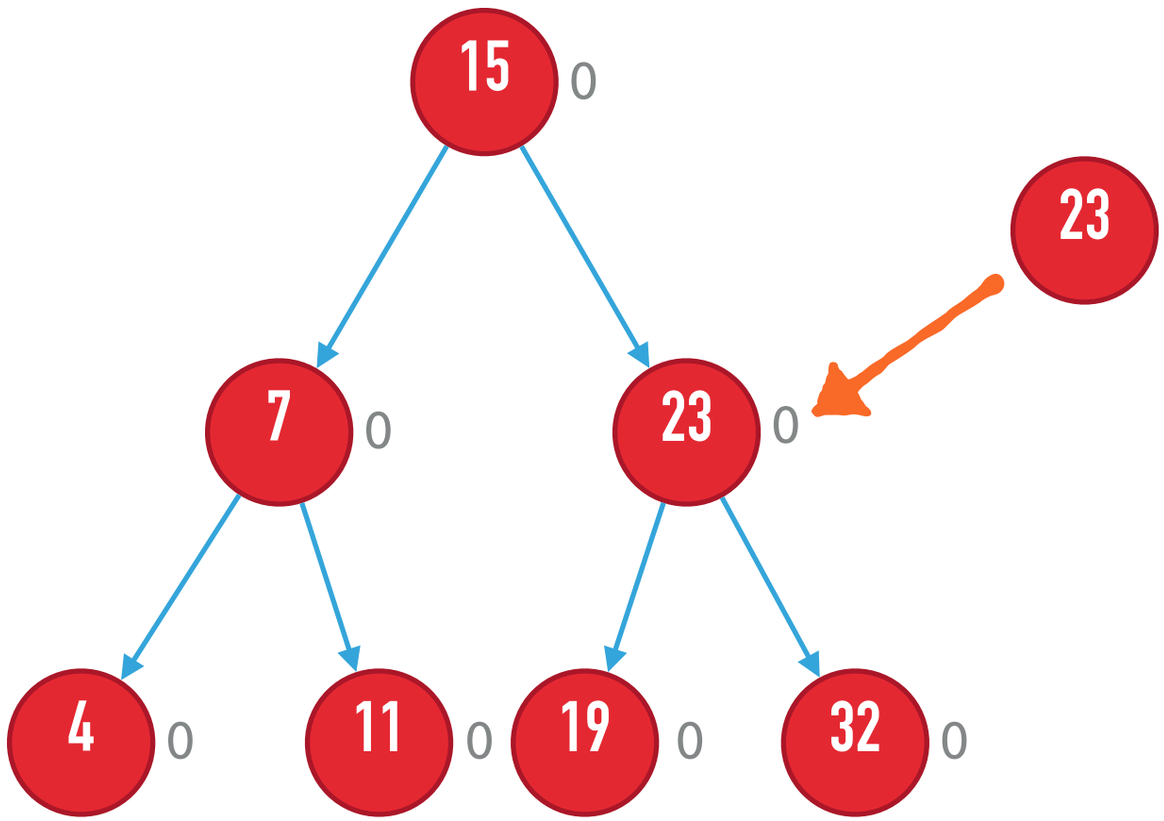


Abbiamo inserito il nodo, ora svuotiamo lo stack e aggiorniamo il bilanciamento

INSERIMENTO

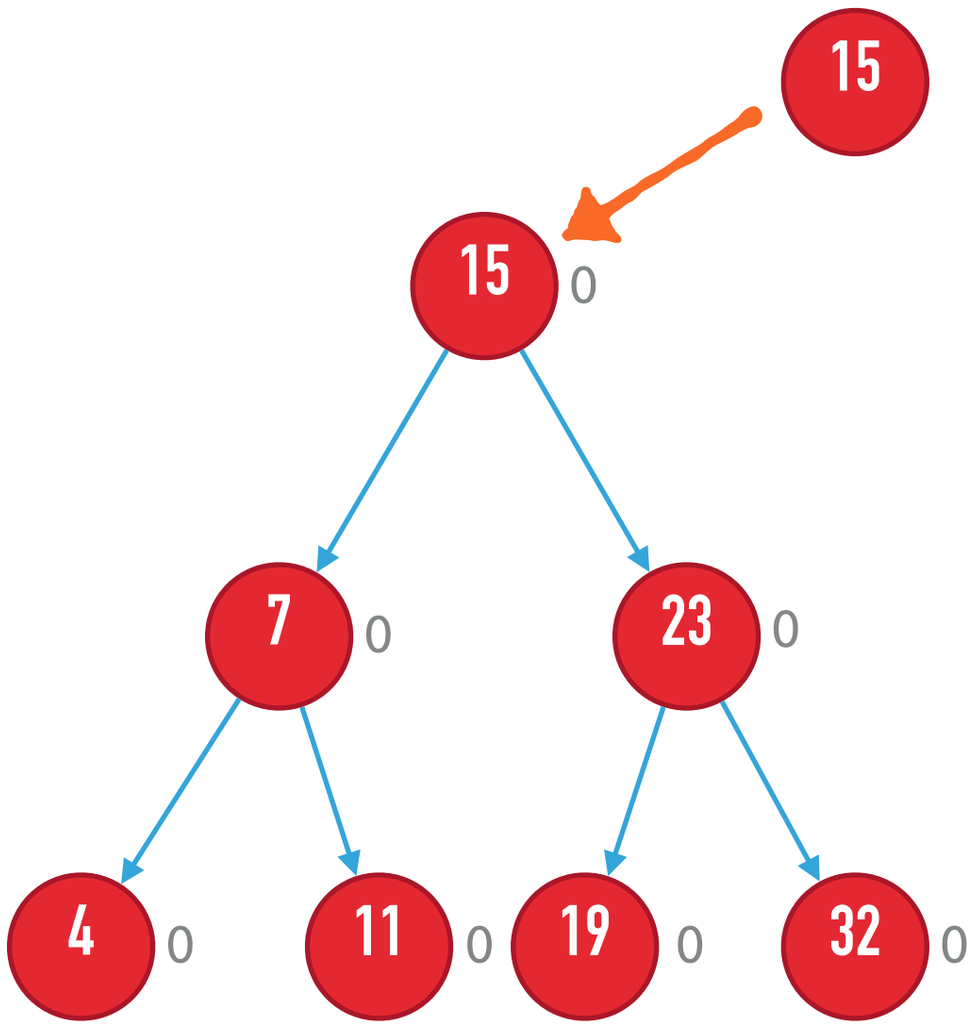
Inseriamo il nodo “19”

Stack dei nodi visitati



INSERIMENTO

Inseriamo il nodo “19”



Stack dei nodi visitati

INSERIMENTO

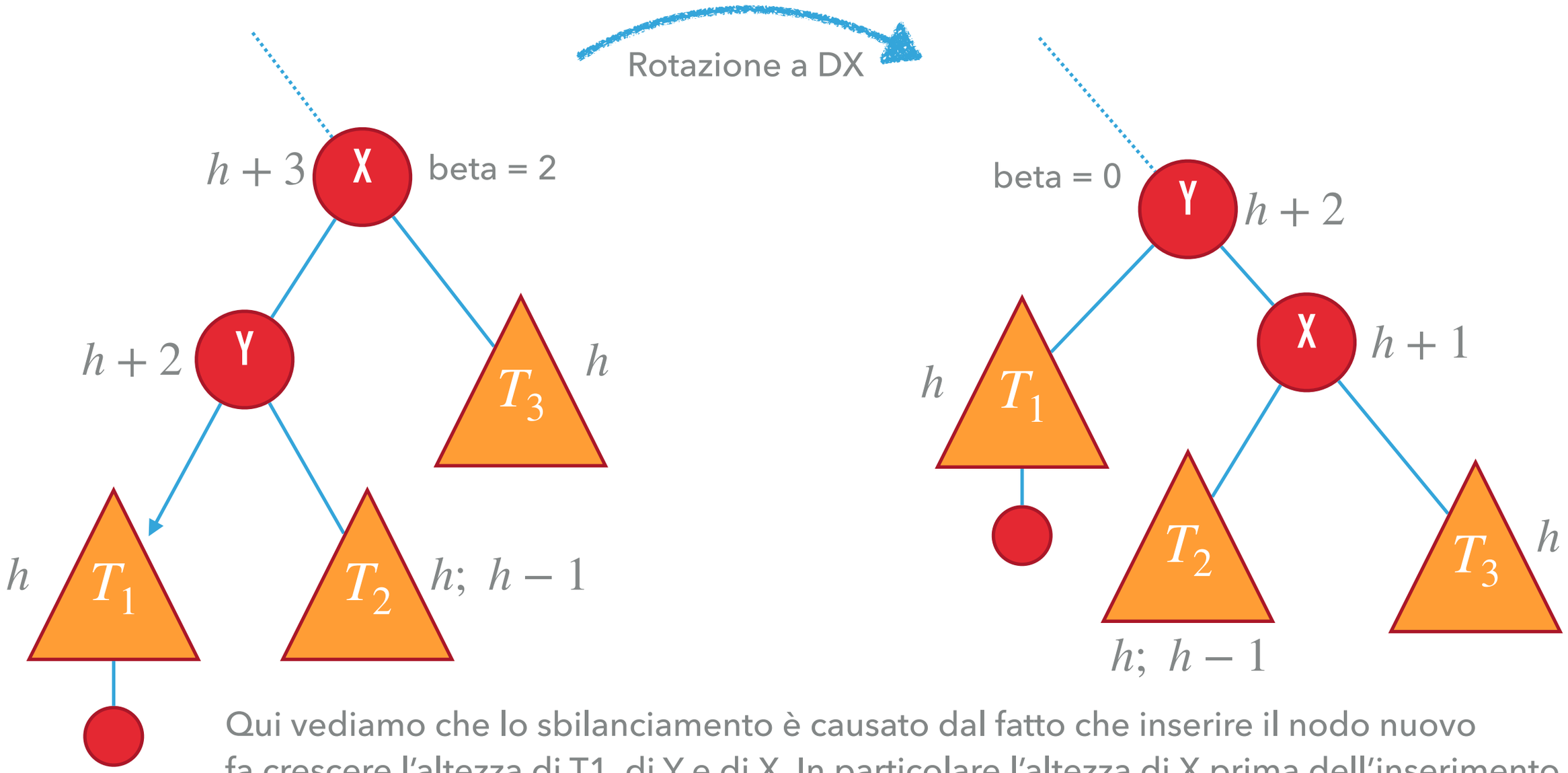
- Il caso migliore è quello in cui non otteniamo nessun valore $+2$ o -2 e aggiorniamo solamente il bilanciamento
- Notate come se c'è uno sbilanciamento, esso deve essere con un valore ± 2, dato che l'aggiunta di un nodo modifica l'altezza di al più 1.
- Nel caso vi sia sbilanciamento cerchiamo il **primo**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

INSERIMENTO

- 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



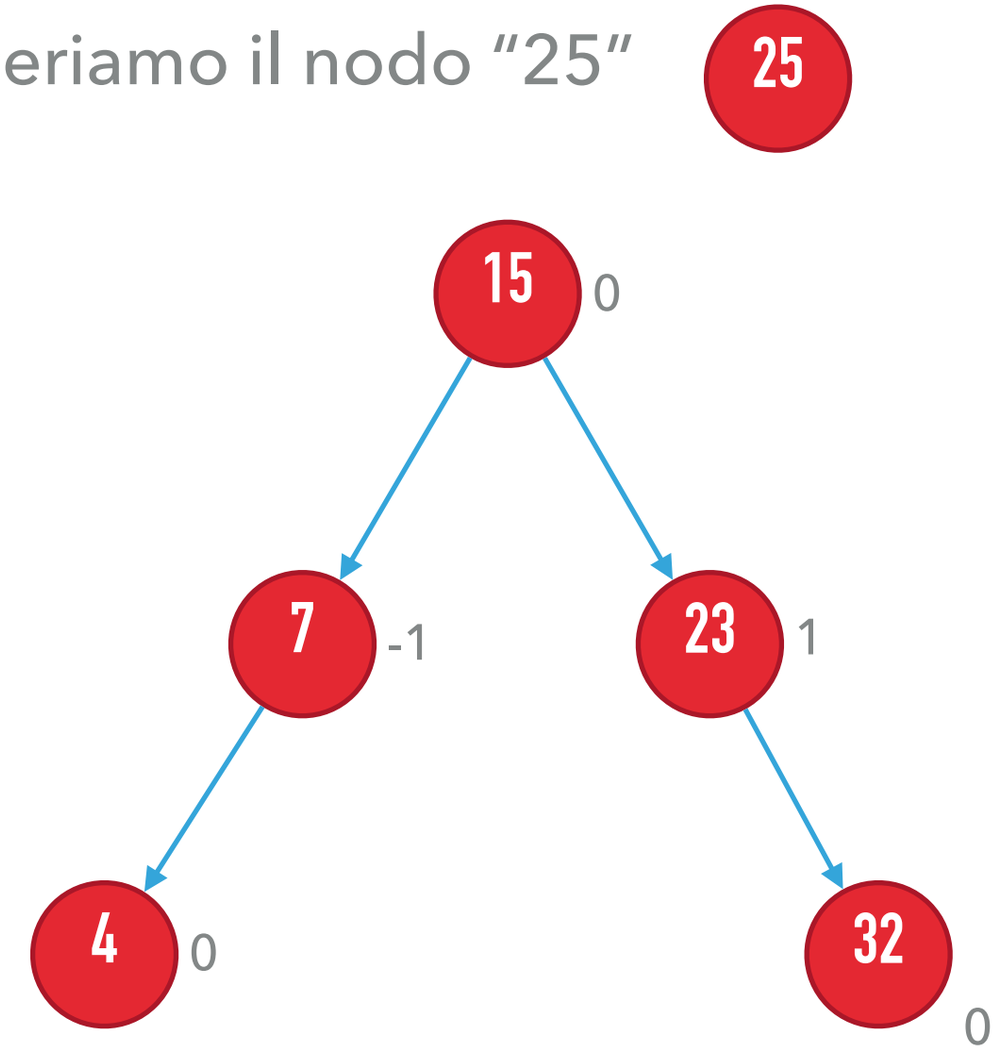
Qui vediamo che lo sbilanciamento è causato dal fatto che inserire il nodo nuovo fa crescere l'altezza di T1, di Y e di X. In particolare l'altezza di X prima dell'inserimento è h+2, e l'altezza del nodo Y, che sostituisce X, dopo la rotazione è di nuovo h+2.
Non solo l'albero si ribilancia, ma anche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

INSERIMENTO

- 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

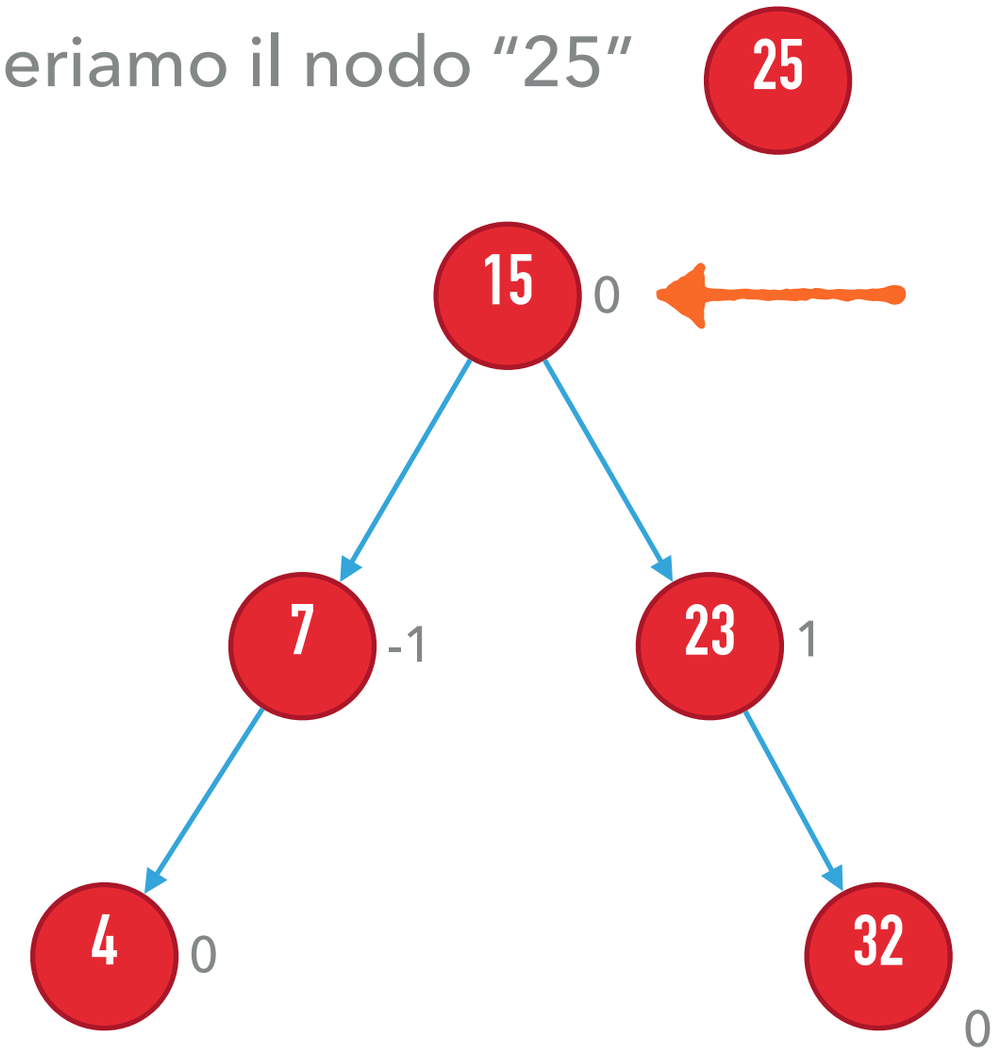
INSERIMENTO

Inseriamo il nodo “25”



INSERIMENTO

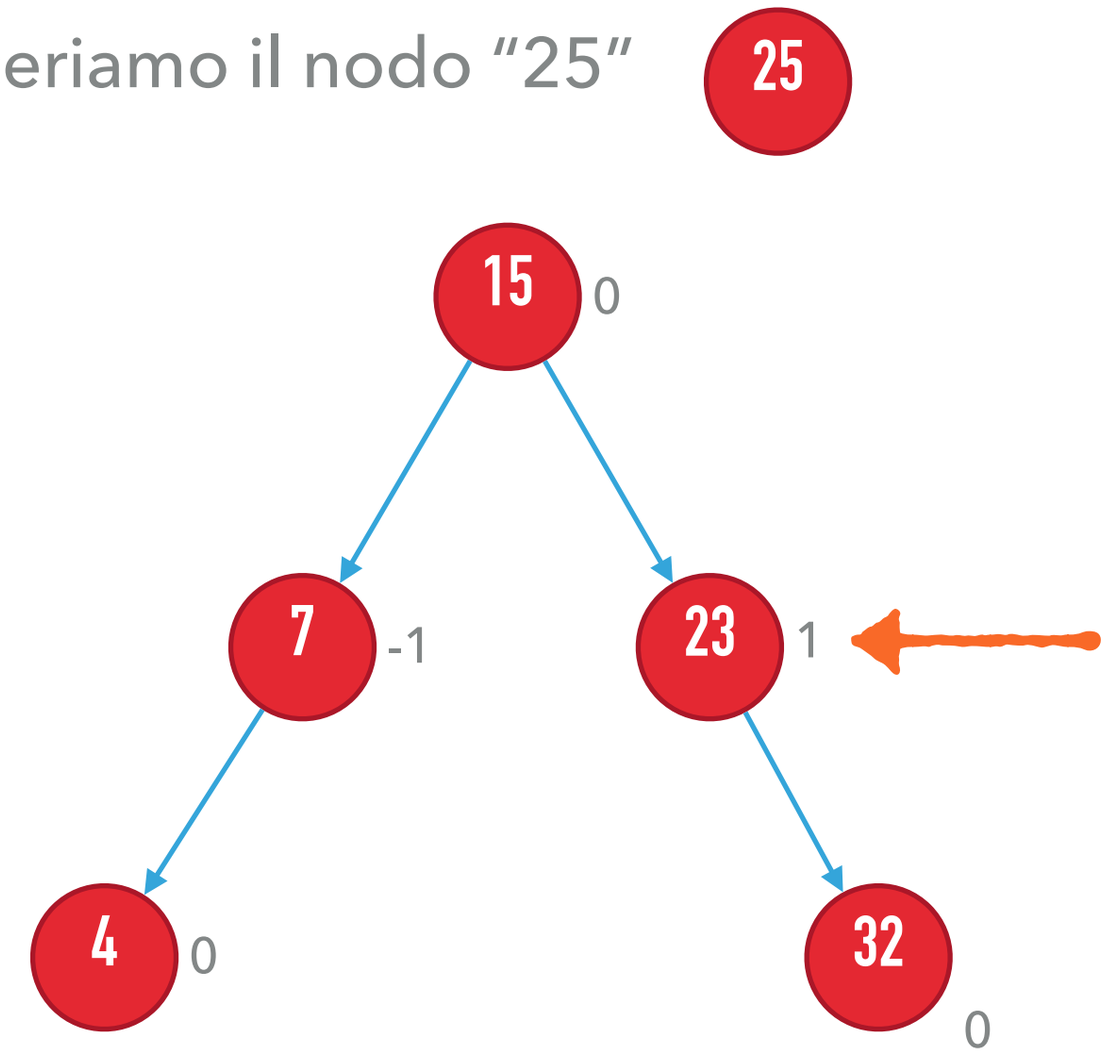
Inseriamo il nodo “25”

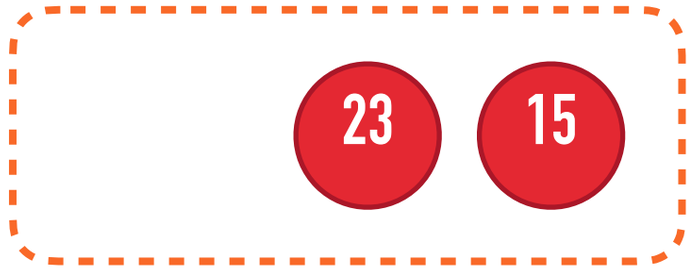


Stack dei nodi visitati

INSERIMENTO

Inseriamo il nodo “25”

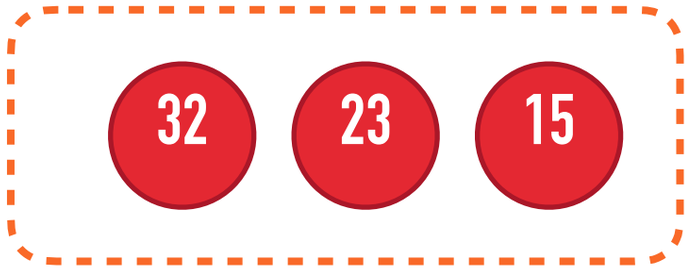




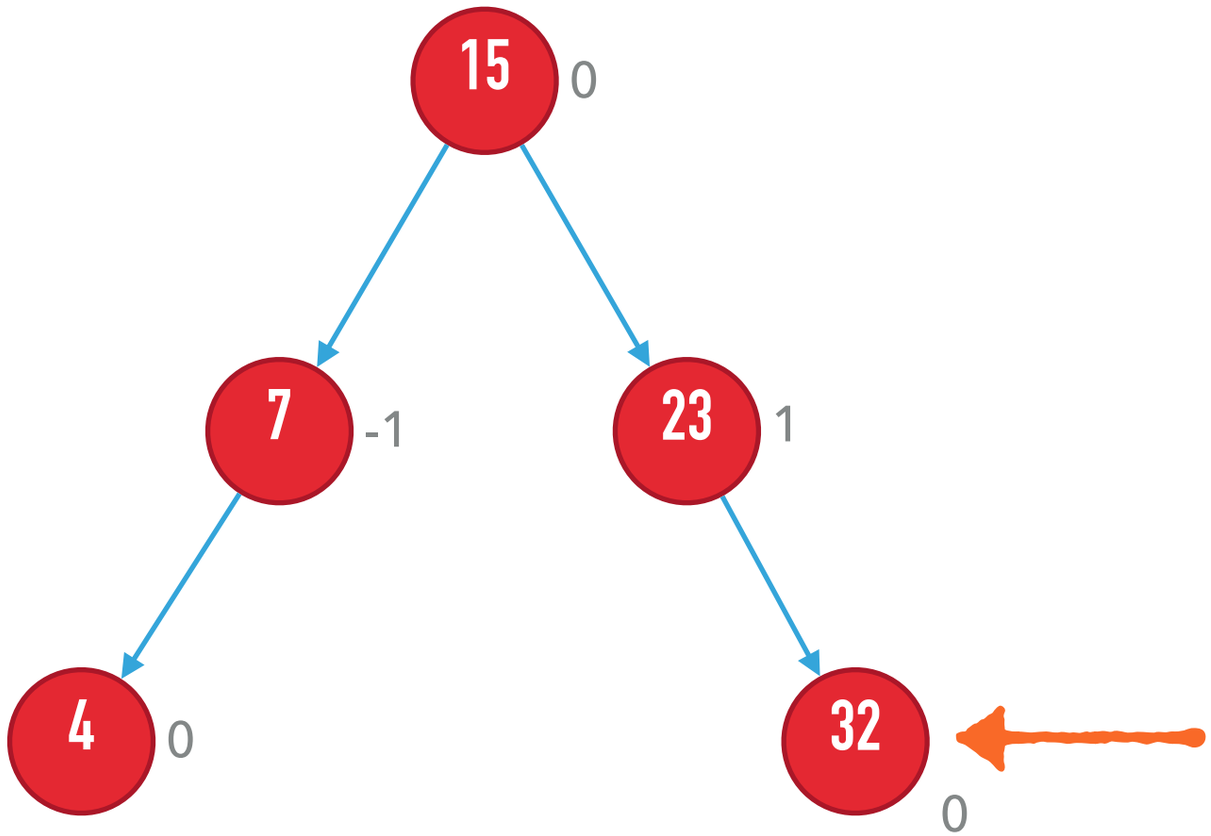
Stack dei nodi visitati

INSERIMENTO

Inseriamo il nodo “25”

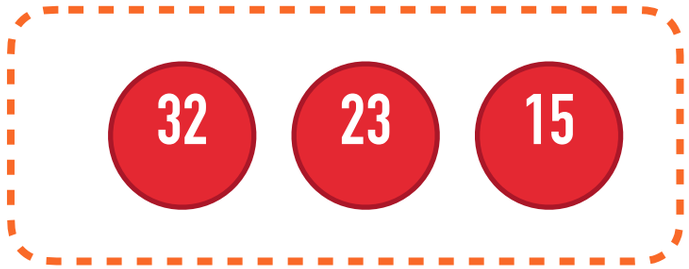


Stack dei nodi visitati

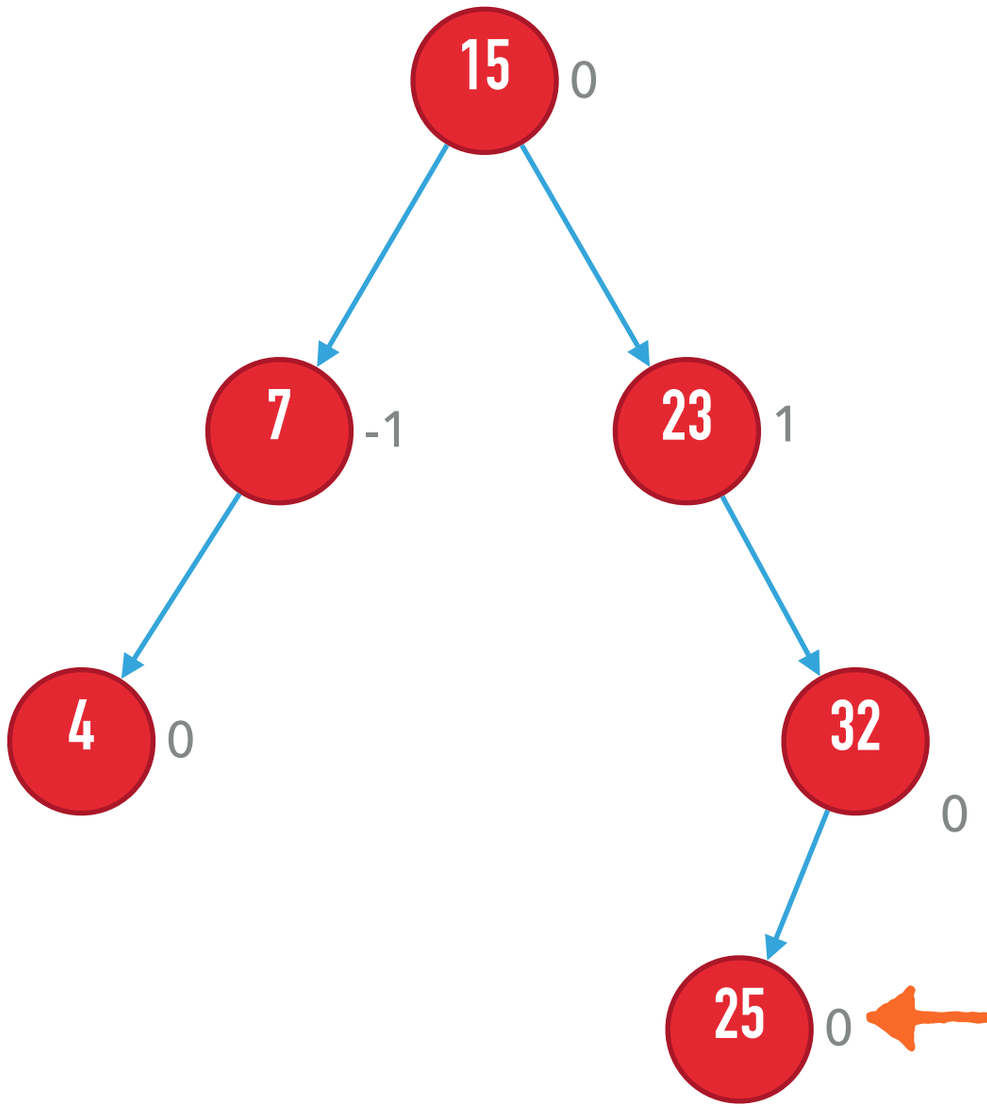


INSERIMENTO

Inseriamo il nodo “25”



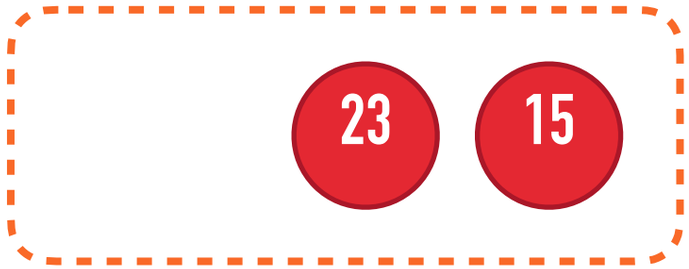
Stack dei nodi visitati



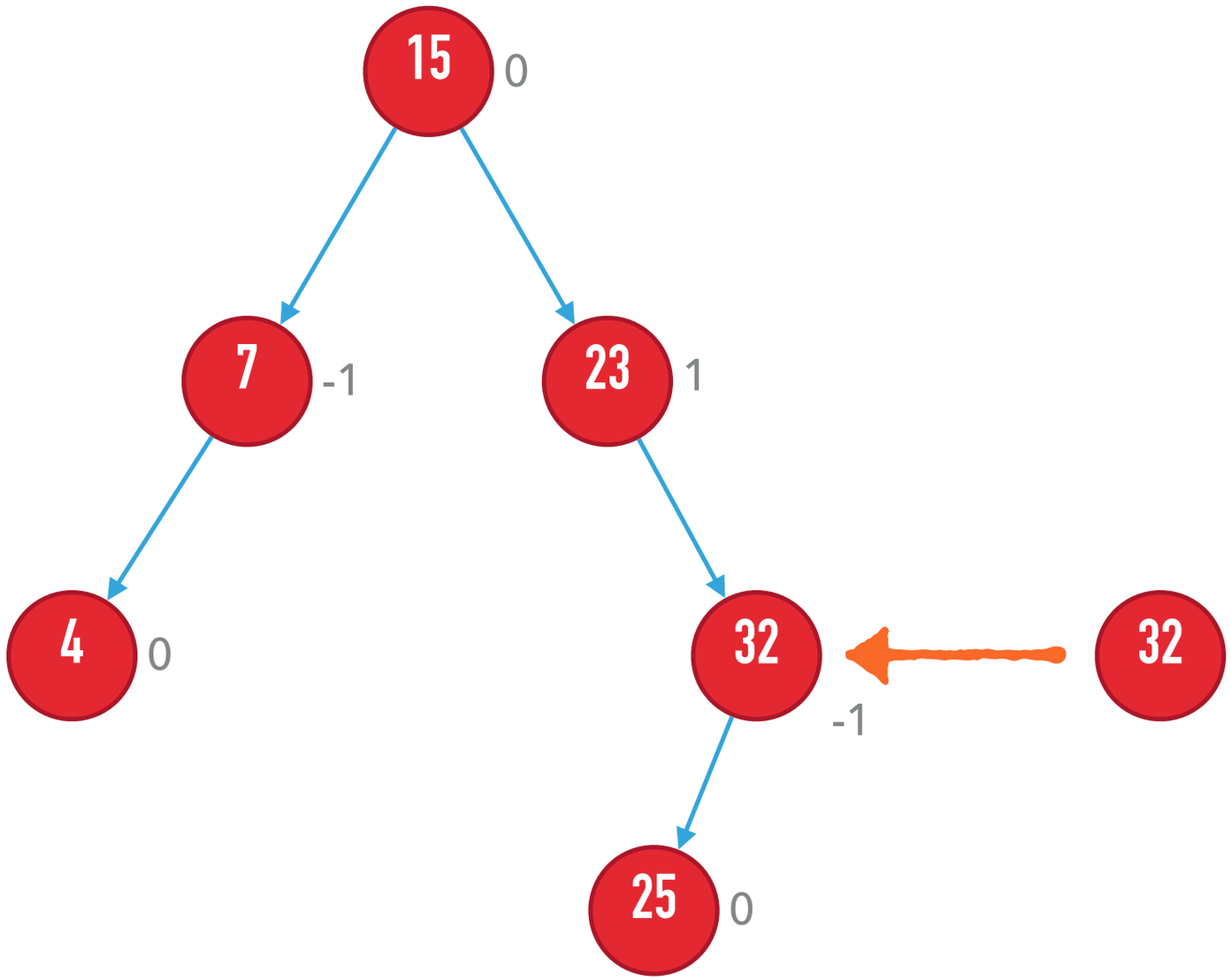
Inseriamo il nodo ed iniziamo ad aggiornare lo sbilanciamento

INSERIMENTO

Inseriamo il nodo “25”



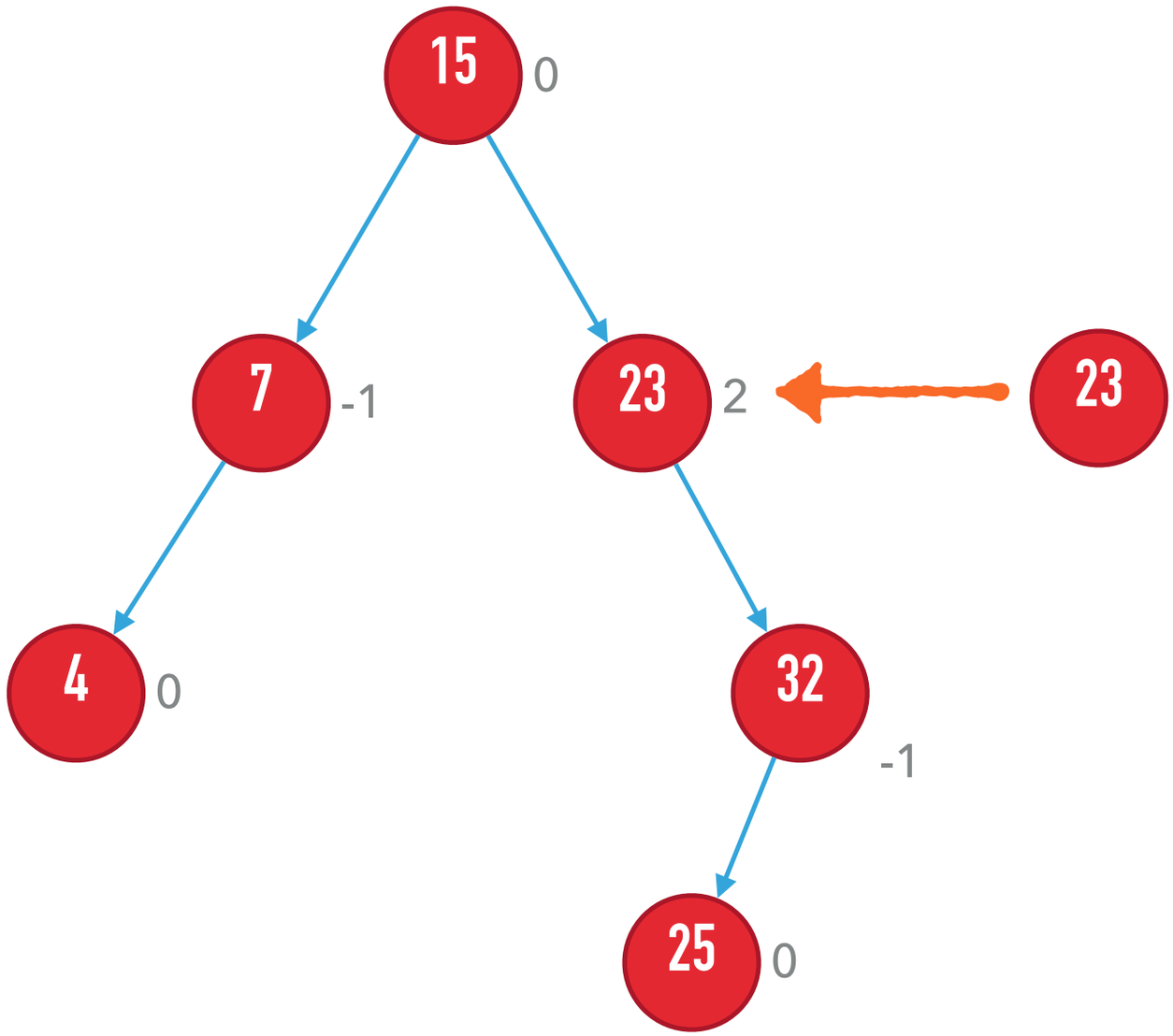
Stack dei nodi visitati



INSERIMENTO

Inseriamo il nodo “25”

Stack dei nodi visitati

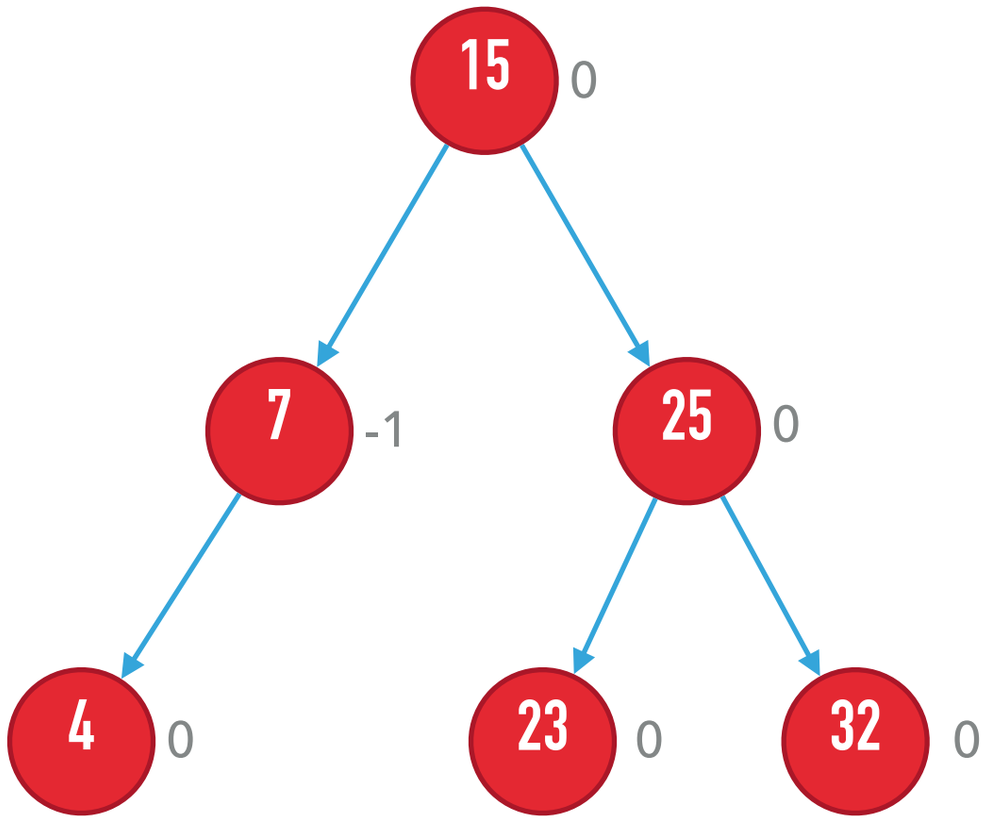


Abbiamo trovato il primo nodo sbilanciato: caso destra-sinistra

INSERIMENTO

Inseriamo il nodo “25”

Stack dei nodi visitati



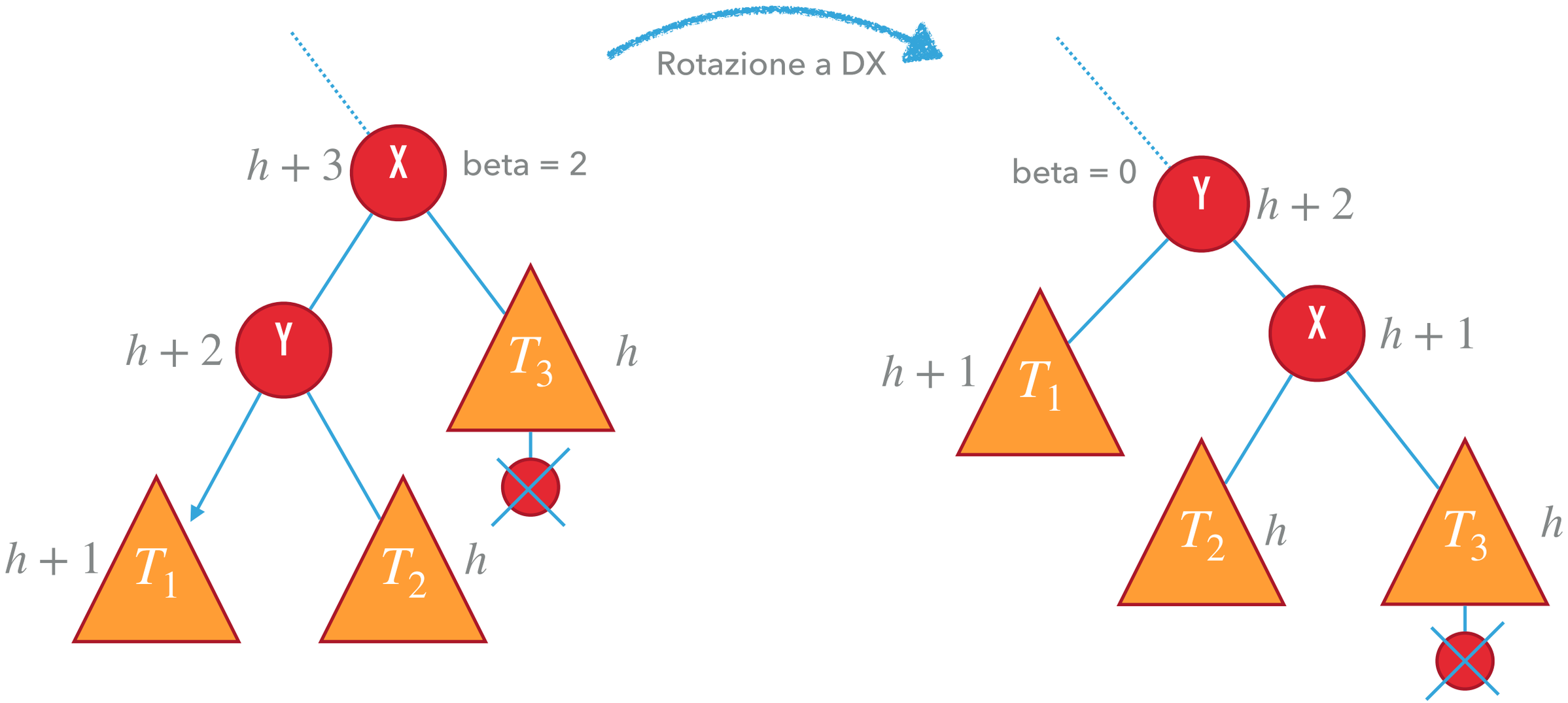
Abbiamo ribilanciato l’albero

CANCELLAZIONE

- 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

- 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)

ALBERI ROSSO-NERI

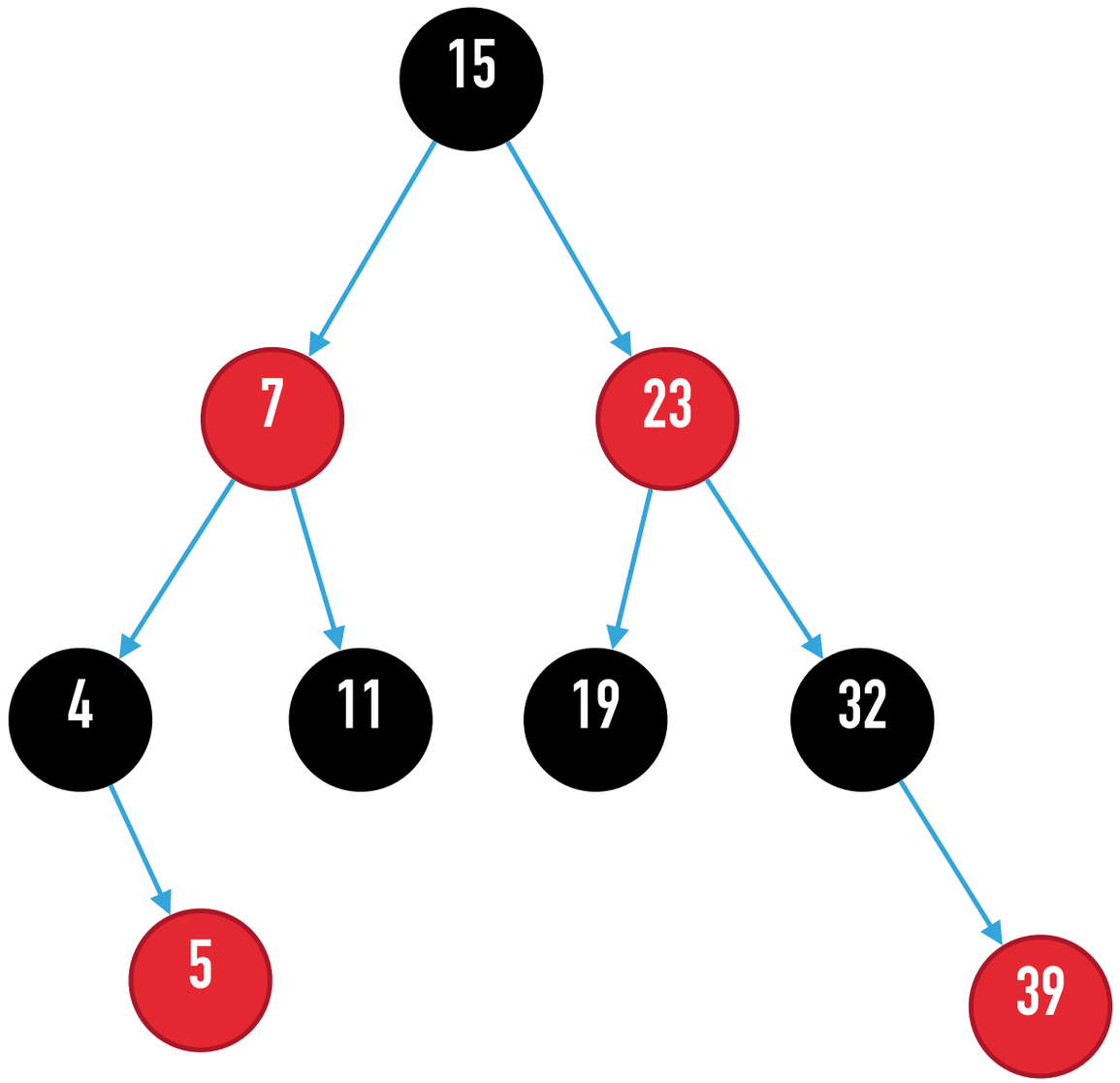
ALTRE TIPOLOGIE DI ALBERI BINARI DI RICERCA BILANCIATI

- 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.
- 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

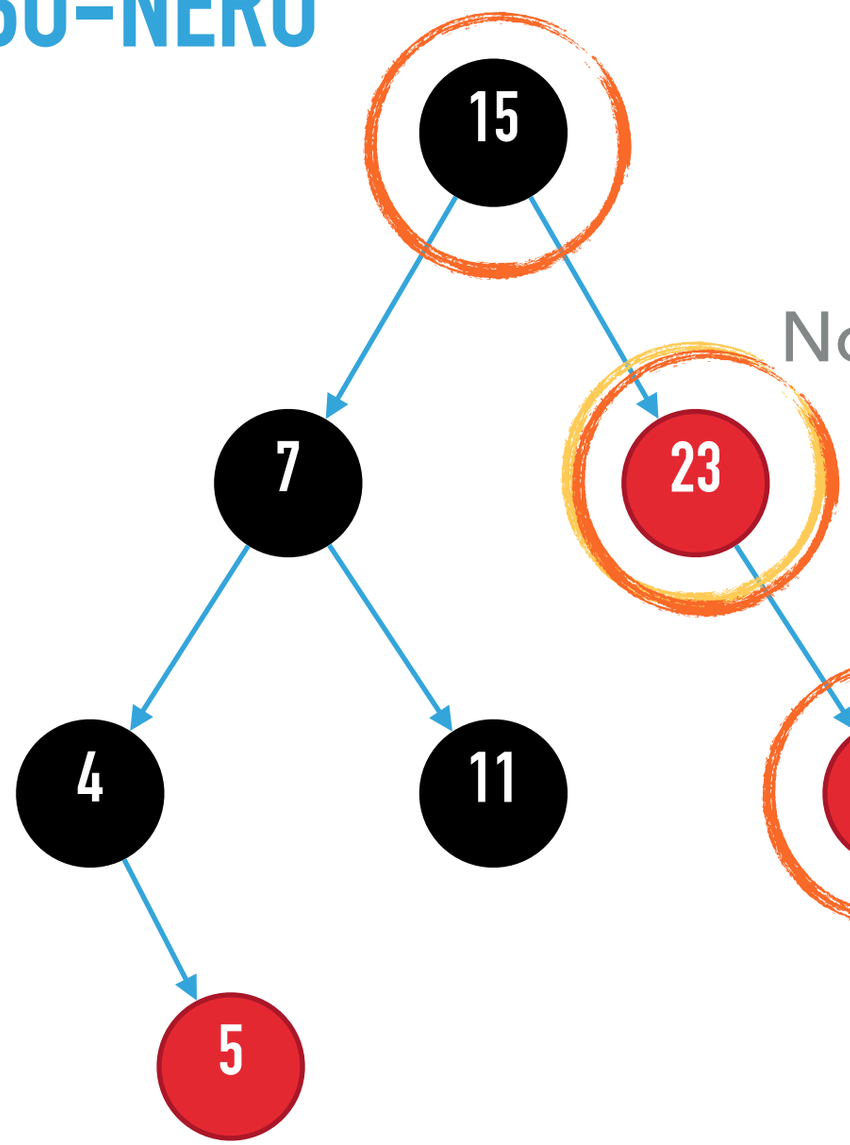
- Ogni nodo è colorato di rosso o di nero
- La radice deve essere colorata di nero
- I valori None si considerano colorati di nero
- I figli di un nodo rosso devono essere nodi neri
- Ogni percorso da un qualsiasi nodo ad un valore None nel sottoalbero del nodo (equiv. un percorso per arrivare alle foglie o ai nodi con 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 $O(\log n)$.

DYNAMIC SELECT

CALCOLARE I PERCENTILI IN UN INSIEME DINAMICO

Supponiamo di avere un albero binario di ricerca. Come facciamo a trovare il p-simo percentile, e.g. la mediana?

Partiamo dalla radice. Ci sono tre casi:

- a sx della radice *r* ci sono $m = \left\lfloor \frac{n}{2} \right\rfloor$ nodi: *r* è la mediana

- a sx della radice *r* ci sono $<m$ nodi: la mediana è a dx

- a sx della radice *r* ci sono $>m$ nodi: la mediana è a sx

AUMENTARE UNA STRUTTURA DATI

Per poter risolvere il problema di select efficientemente, devo quindi sapere quanti nodi contiene un sottoalbero.

Aggiungo ad ogni nodo *x* una variabile *size*, che conta il numero di nodi nel sottoalbero radicato in *x*. Vale
x.size = x.left.size + x.right.size +1

Devo estendere le operazioni su un BST per mantenere aggiornato il campo *size*. In particolare per INSERT e DELETE.

INSERT PER DYNAMIC SELECT

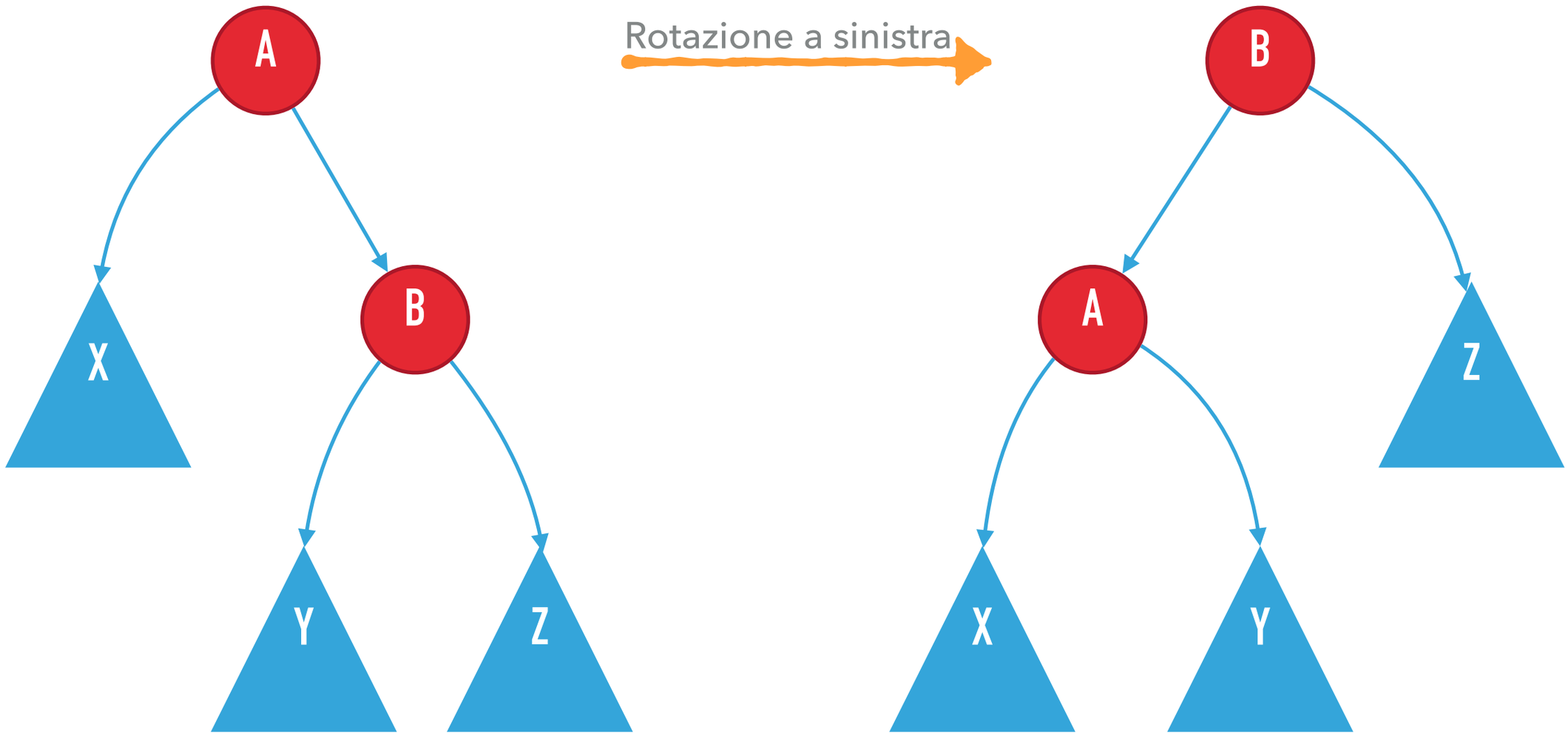
Inserisco il nodo normalmente come foglia. Quali nodi avranno un campo *size* diverso?

Risalgo dal nodo inserito alla radice dell'albero ed incremento *size* di 1 per ogni nodo.

Nel caso di un albero AVL, in cui posso fare delle rotazioni per bilanciare l'albero, devo modificare le rotazioni in modo che preservino il valore corretto di *size*.

ROTAZIONI PER DYNAMIC SELECT

Cambiano solo i valori di size in A e B



size(A) = size(A) - size(B.right) - 1
size(B) = size(B) + size(A.left) + 1

ROTAZIONE SINISTRA PER DYNAMIC SELECT

- Parametri: nodo a
- b = a.right # nodo che prenderà il posto di a
- **b.size = b.size + a.left.size + 1 # aggiorno size b**
- **a.size = a.size - b.right.size - 1 # aggiorno size 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