

Algoritmi e Strutture Dati

Capitolo 6 Rotazioni AVL (*)

Riepilogo: alberi AVL

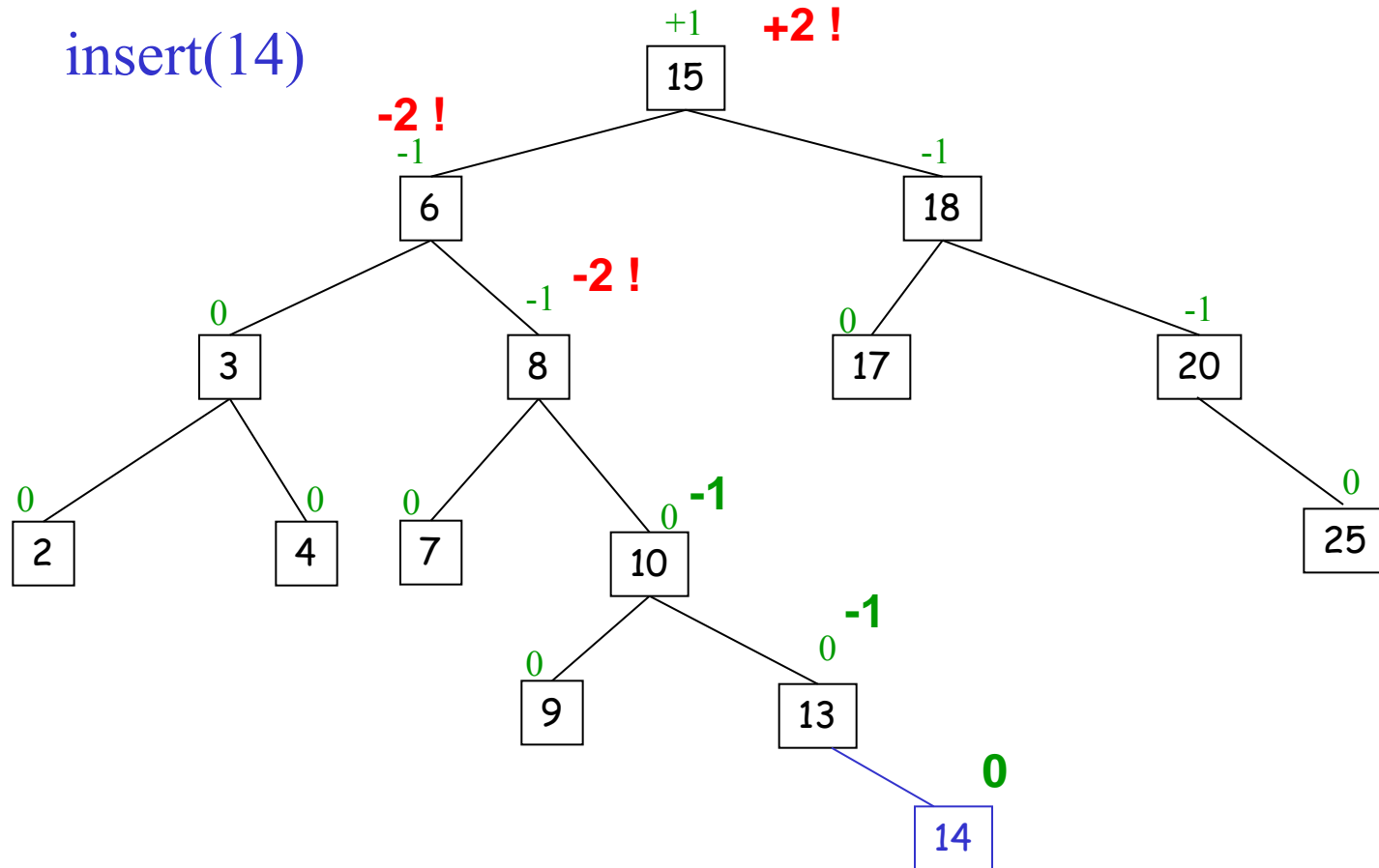
- Un albero **AVL** è un **ABR** in cui ad ogni nodo la differenza tra l'altezza del sottoalbero sinistro e l'altezza del sottoalbero destro (detta **fattore di bilanciamento**) **deve essere** al più pari ad **1 in valore assoluto** (si osservi che la definizione richiede che tale proprietà venga mantenuta nel caso in cui l'ABR sia soggetto ad inserimenti e cancellazioni)
- Abbiamo dimostrato che, grazie a tale proprietà, un AVL con **n** nodi ha altezza **$h = \Theta(\log n)$**

Implementazione delle operazioni

- L'operazione **search** procede come in un ABR, e quindi costerà $O(\log n)$, ma **inserimenti** e **cancellazioni** potrebbero sbilanciare l'albero; faremo quindi vedere che la **insert** e la **delete** potranno essere eseguite in $O(\log n)$, mantenendo **invariante** la proprietà sui fattori di bilanciamento
- ⇒ Dobbiamo far vedere come mantenere **invariante** la proprietà sui fattori di bilanciamento tramite opportune **rotazioni** all'interno dell'AVL

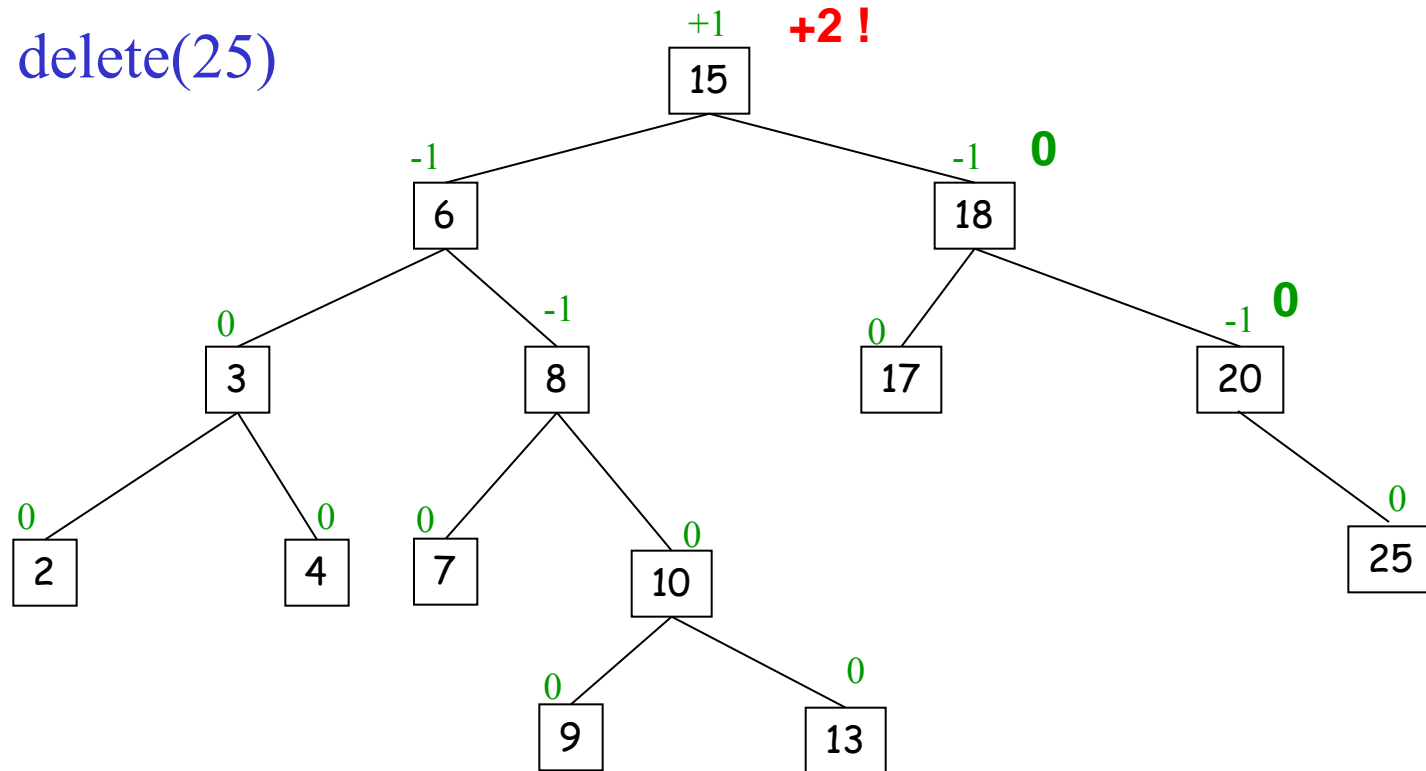
Gli inserimenti possono sbilanciare l'AVL

insert(14)

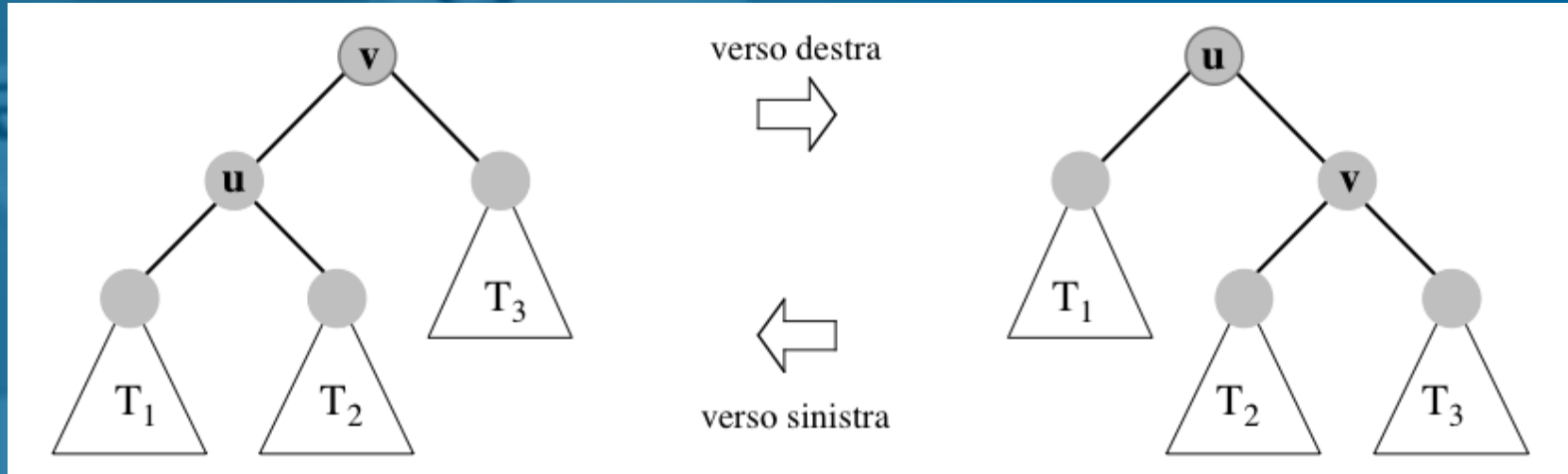


Le cancellazioni possono sbilanciare l'AVL

delete(25)



Rotazione di base verso destra/sinistra sul nodo v/u



- Mantiene la proprietà di **ordinamento totale**
- Richiede tempo $O(1)$ (vanno aggiornati i puntatori dei nodi coinvolti e i fattori di bilanciamento di u e v , e questo può essere fatto in tempo costante poiché manteniamo in ogni nodo le altezze dei sottoalberi radicati)
- Ovviamente cambiano i fattori di bilanciamento dei nodi coinvolti (ovvero u e v), ma sfrutteremo esattamente questa cosa per ristabilire il bilanciamento dell'AVL!

Ribilanciamento tramite rotazioni

- Le rotazioni sono effettuate su nodi sbilanciati
- Sia v un nodo con fattore di bilanciamento $\beta(v) = \pm 2$ (si noti che questo è il massimo sbilanciamento possibile a valle di un inserimento o di una cancellazione in un AVL); allora, il sottoalbero sinistro o destro di v **sbilancia** v (vale a dire, ha un'altezza eccessiva); ai fini dell'applicazione del corretto ribilanciamento, dobbiamo ulteriormente distinguere **il sottoalbero del sottoalbero** che sbilancia v . Sia T tale sottoalbero; allora, a seconda della posizione di T si hanno 4 casi:

$\beta(v)=+2$

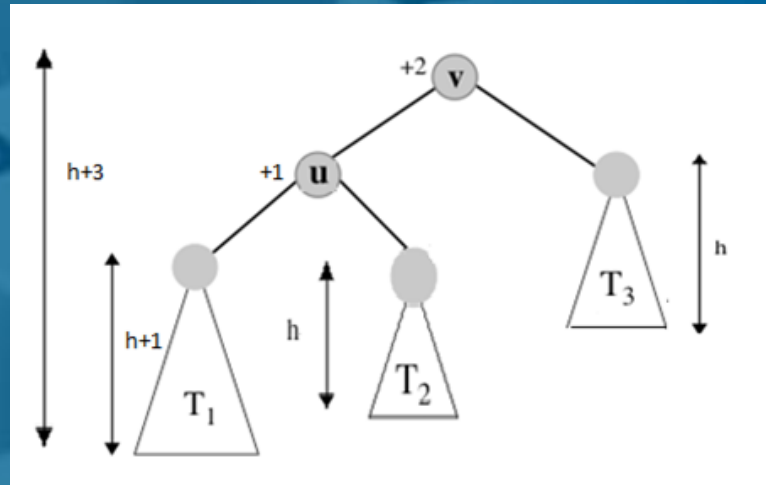
$\beta(v)=-2$

Sinistra - sinistra	(SS)	T è il sottoalbero sinistro del figlio sinistro di v
Destra - destra	(DD)	T è il sottoalbero destro del figlio destro di v
Sinistra - destra	(SD)	T è il sottoalbero destro del figlio sinistro di v
Destra - sinistra	(DS)	T è il sottoalbero sinistro del figlio destro di v

- I quattro casi sono simmetrici a coppie (SS/DD e SD/DS), e quindi noi analizzeremo solo **SS** e **SD**

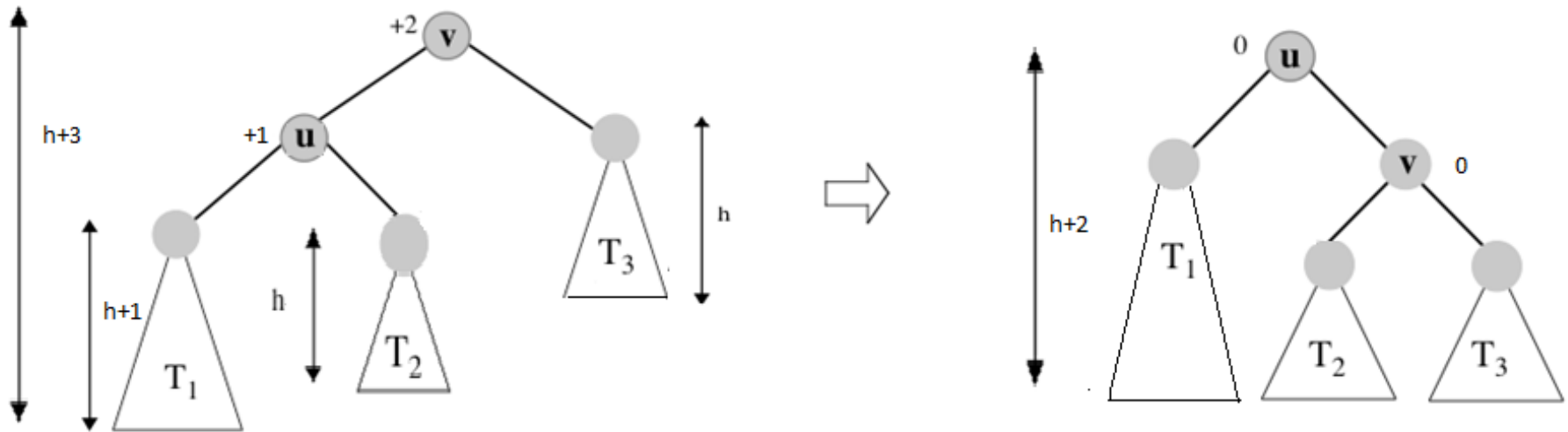
Caso SS

- L'altezza di $T(v)$ (**sottoalbero critico**) è $h+3$, l'altezza di $T(u)$ è $h+2$, l'altezza di T_3 è h , l'altezza di T_1 è $h+1$, e infine l'altezza di T_2 è $h \Rightarrow \beta(v)=+2$ e lo sbilanciamento è provocato da T_1 :



- In questo caso si applica una rotazione semplice verso destra su v

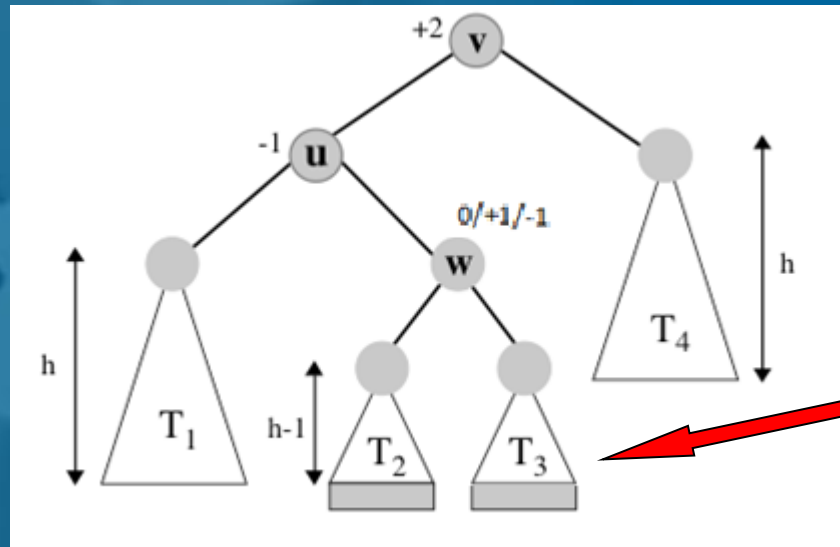
Rotazione nel caso SS



- L'altezza dell'albero coinvolto nella rotazione passa da $h+3$ a $h+2$, e il fattore di bilanciamento di u e v diventa pari a 0
- Il caso SS può essere provocato:
 1. da **inserimenti** in T_1 , che innalzano la sua altezza da h ad $h+1$; si noti che in tal caso l'altezza del sottoalbero critico prima dell'inserimento era $h+2$, diventa $h+3$ dopo l'inserimento, e torna ad $h+2$ dopo il ribilanciamento;
 2. da **cancellazioni** in T_3 che abbassano la sua altezza da $h+1$ ad h ; si noti che in tal caso l'altezza del sottoalbero critico prima della cancellazione era $h+3$, rimane $h+3$ dopo la cancellazione, e diventa $h+2$ dopo il ribilanciamento.

Caso SD

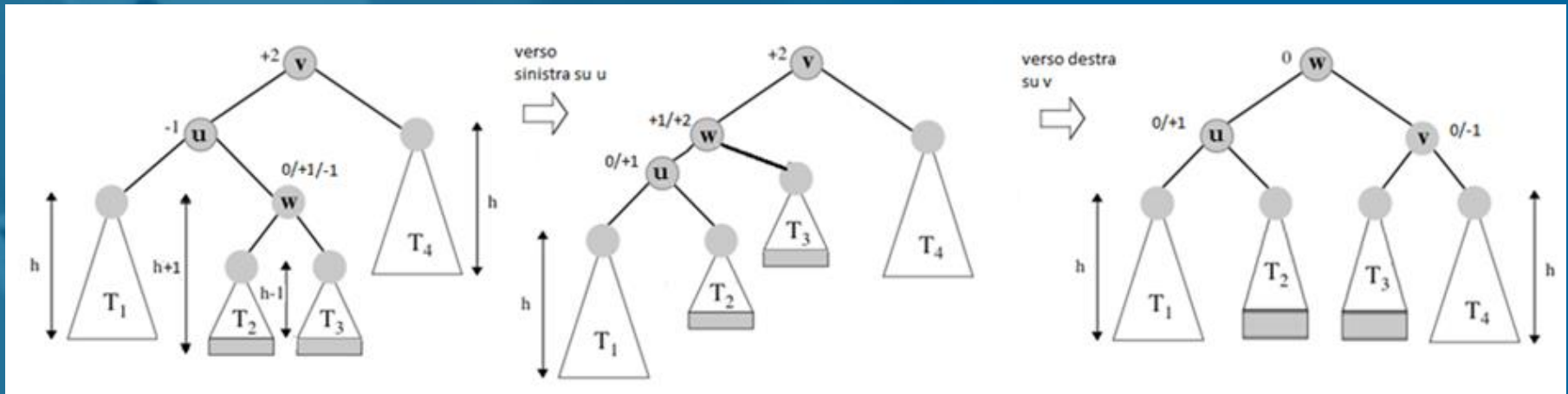
- L'altezza di $T(v)$ è $h+3$, l'altezza di $T(u)$ è $h+2$, l'altezza di T_1 è h , l'altezza di T_4 è h , e l'altezza di $T(w)$ è $h+1 \Rightarrow \beta(v)=+2$, e $\beta(u)=-1$ cioè lo sbilanciamento è provocato dal sottoalbero destro di u



Si noti che almeno uno tra T_2 e T_3 è alto h

- Applicare **due rotazioni semplici**: una verso **sinistra** sul figlio sinistro del nodo critico (nodo u), l'altra verso **destra** sul nodo critico (nodo v)

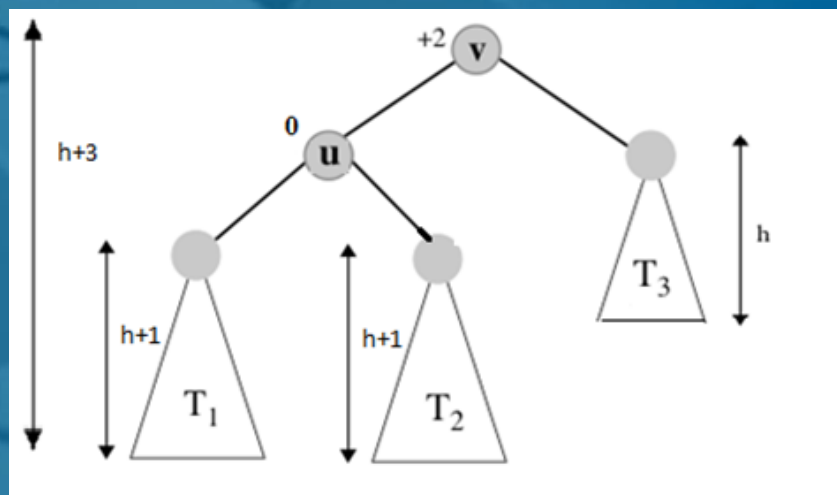
Rotazione nel caso SD



- L'altezza dell'albero dopo la rotazione passa da $h+3$ a $h+2$, poiché T_2 e T_3 sono alti al più h , e il fattore di bilanciamento di w diventa 0 , mentre i fattori di bilanciamento di u e v diventano 0 oppure $+1$ e -1 , rispettivamente.
- Il caso SD può essere provocato:
 1. da un **inserimento** in T_2 o in T_3 , che ne innalza l'altezza da $h-1$ ad h (si noti che in questo caso l'altezza dell'altro sottoalbero fratello deve essere pari ad $h-1$, altrimenti l'AVL sarebbe già stato sbilanciato!); in tal caso l'altezza del sottoalbero critico prima dell'inserimento era $h+2$, diventa $h+3$ dopo l'inserimento, e torna ad essere $h+2$ dopo il ribilanciamento;
 2. da **cancellazioni** in T_4 che abbassano la sua altezza da $h+1$ ad h (si noti che in questo caso almeno uno tra T_2 e T_3 deve essere alto h , ma potrebbero esserlo entrambi); in tal caso l'altezza del sottoalbero critico prima della cancellazione era $h+3$, rimane $h+3$ dopo la cancellazione, e diventa $h+2$ dopo il ribilanciamento.

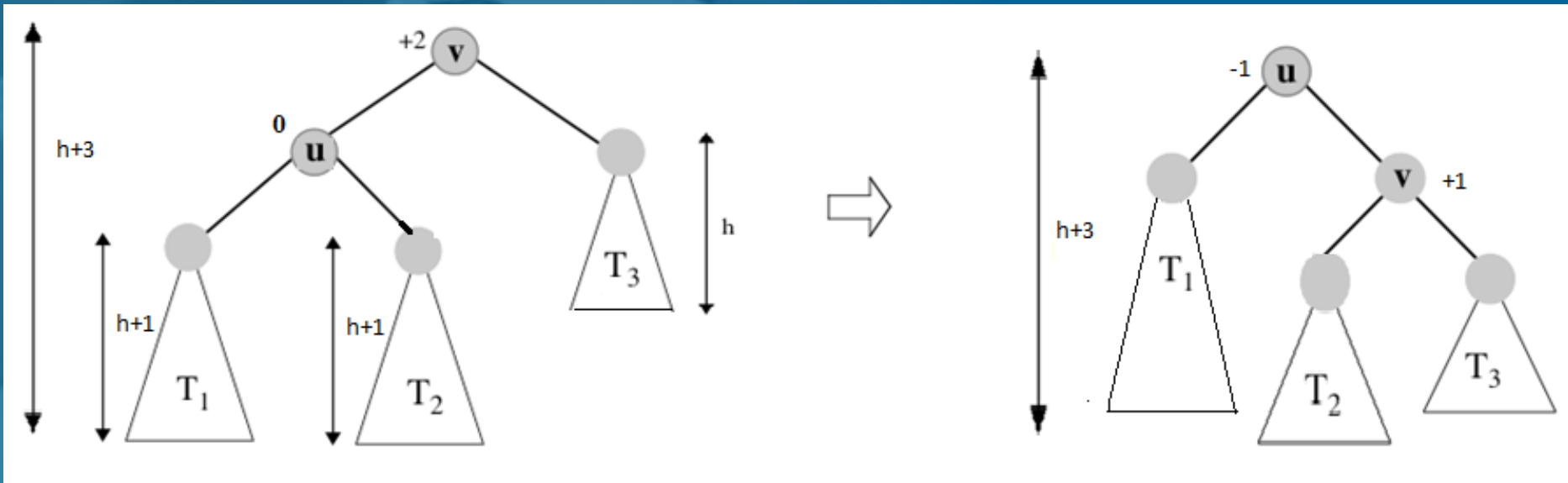
Casi SS e SD contemporanei

- L'altezza di $T(v)$ è $h+3$, l'altezza di $T(u)$ è $h+2$, l'altezza di T_3 è h , e l'altezza di T_1 e T_2 è $h+1 \Rightarrow \beta(v)=+2$ e lo sbilanciamento è provocato **contemporaneamente** da T_1 e T_2 ; si noti che questa situazione si verifica solo nel caso in cui sia stato **cancellato** un elemento da T_3 , poiché un inserimento in T_1 o in T_2 non può cambiare l'altezza di $T(u)$, e quindi l'AVL sarebbe già stato sbilanciato!



- Questo caso viene trattato come il caso SS facendo una rotazione semplice verso destra su v

Rotazione nel caso (SS \wedge SD)



l'altezza dell'albero coinvolto nella rotazione rimane pari a $h+3$, e il fattore di bilanciamento di u diventa pari a -1 , mentre quello di v diventa pari a $+1$; quindi, l'altezza del sottoalbero critico prima della cancellazione in T_3 era $h+3$, e rimane tale dopo la cancellazione e il successivo ribilanciamento

Riepilogo ribilanciamenti

Tipo	Causa	Altezza sottoalbero critico prima dell'operazione	Altezza sottoalbero critico dopo l'operazione	Altezza sottoalbero critico dopo ribilanciamento
SS	Inserimento	x	x+1	x
SS	Cancellazione	x	x	x-1
SD	Inserimento	x	x+1	x
SD	Cancellazione	x	x	x-1
SS \wedge SD	Cancellazione	x	x	x

Ovviamente i casi **DD**, **DS** e **DD \wedge DS** sono simmetrici

Nota bene che la **cancellazione** potrebbe **abbassare di 1** l'altezza del sottoalbero critico, e vedremo che questo può comportare la necessità di altri ribilanciamenti

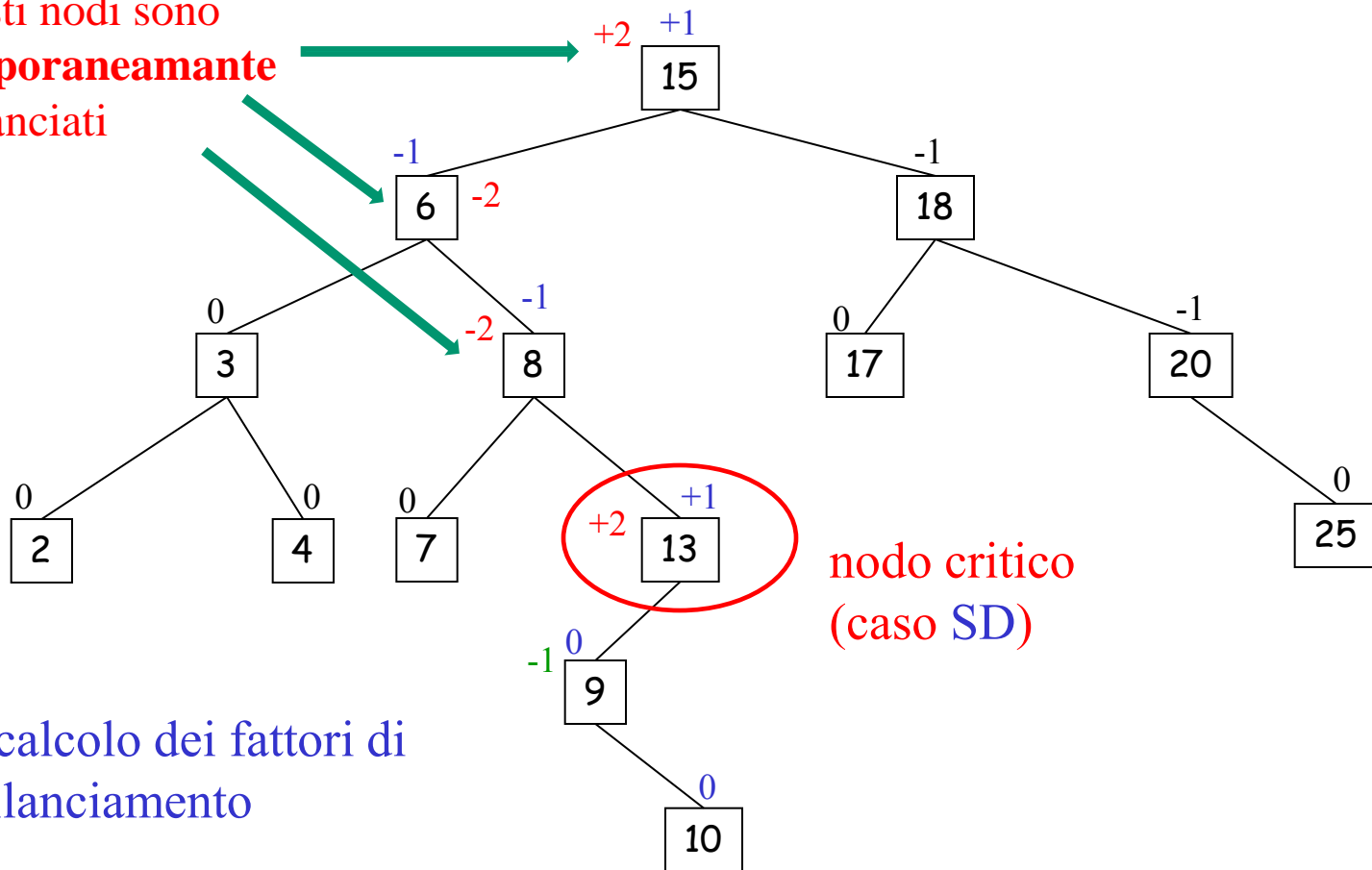
insert(elem e, chiave k)

1. Crea un nuovo nodo z con $elem=e$ e $chiave=k$
2. Inserisci z come in un ABR
3. Ricalcola i fattori di bilanciamento dei nodi nel cammino da z verso la radice, e sia v il primo nodo lungo tale cammino che si è sbilanciato, ovvero il cui fattore di bilanciamento è pari a ± 2 (**nodo critico**)
4. Esegui l'opportuna rotazione semplice (casi SS e DD) o doppia (casi SD e DS) su v , e ricalcola il fattore di bilanciamento dei nodi coinvolti nella rotazione

Osservazione: un solo ribilanciamento è sufficiente, poiché l'altezza del sottoalbero critico coinvolto nella/e rotazione/i torna ad essere **uguale** all'altezza che aveva **prima dell'inserimento** (vedi tabella precedente), ribilanciando così automaticamente tutti i nodi eventualmente sbilanciati nel cammino verso la radice

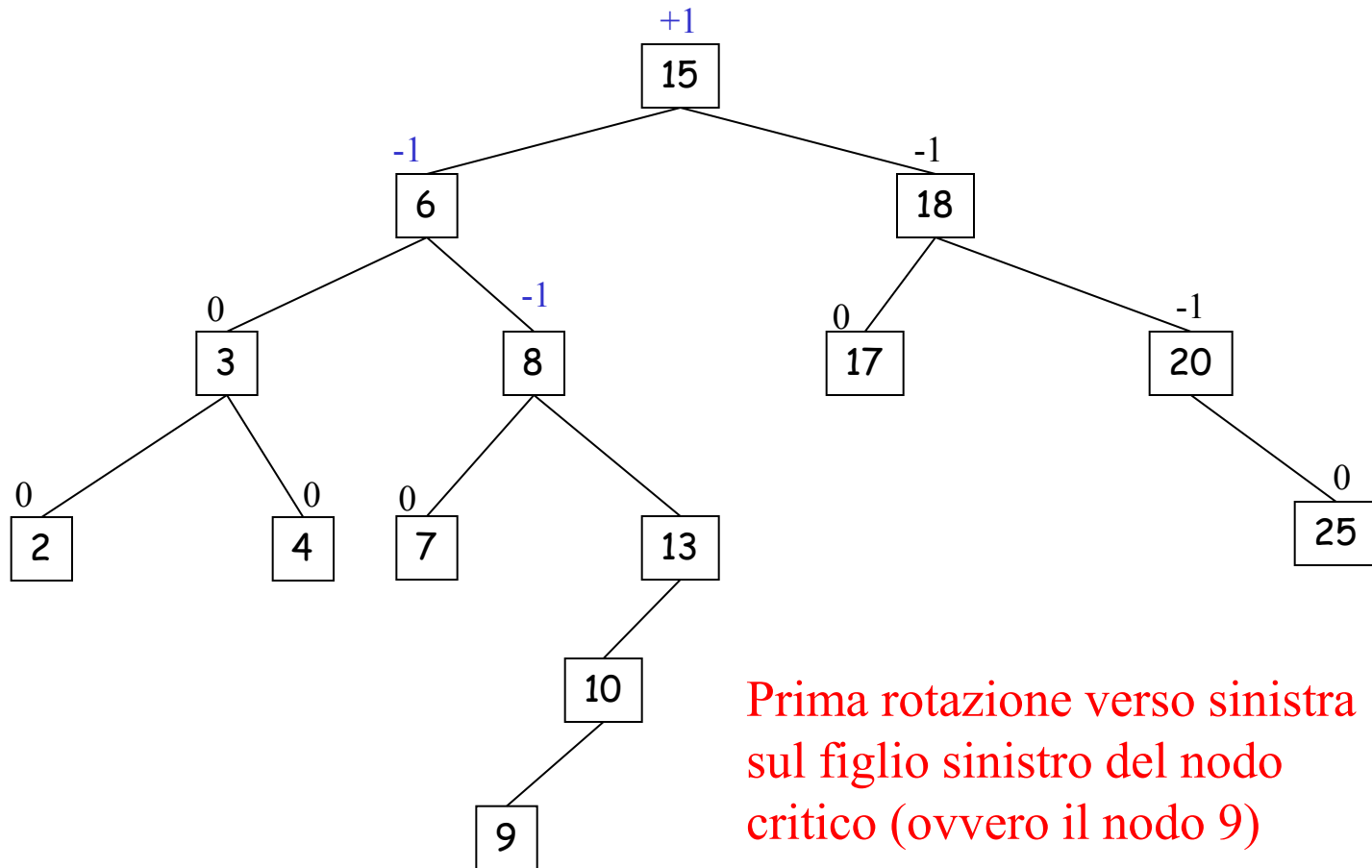
Esempio: insert (10,e)

Si noti che anche questi nodi sono temporaneamente sbilanciati



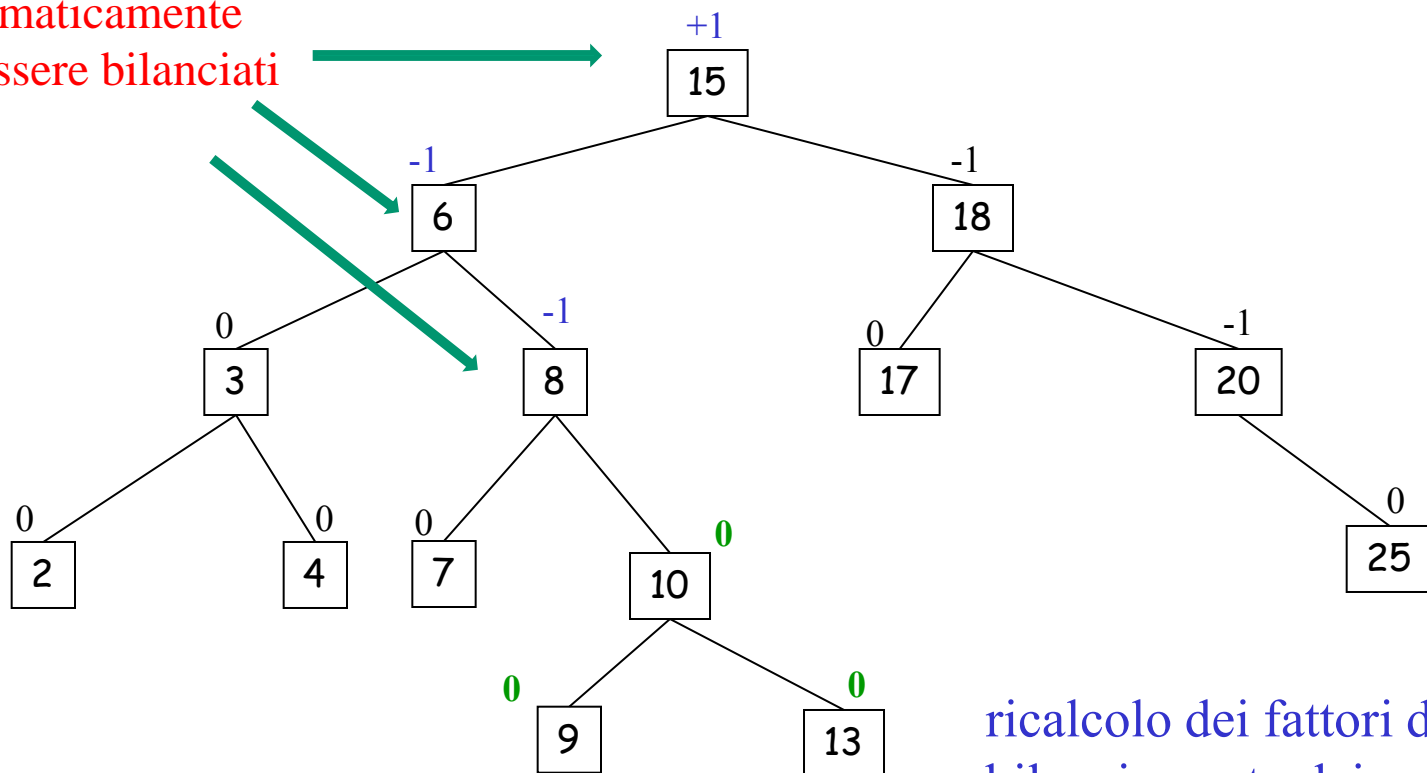
ricalcolo dei fattori di bilanciamento

Esempio: insert (10,e)



Esempio: insert (10,e)

Questi nodi tornano automaticamente ad essere bilanciati



Seconda rotazione verso destra sul nodo critico (ovvero il nodo 13)

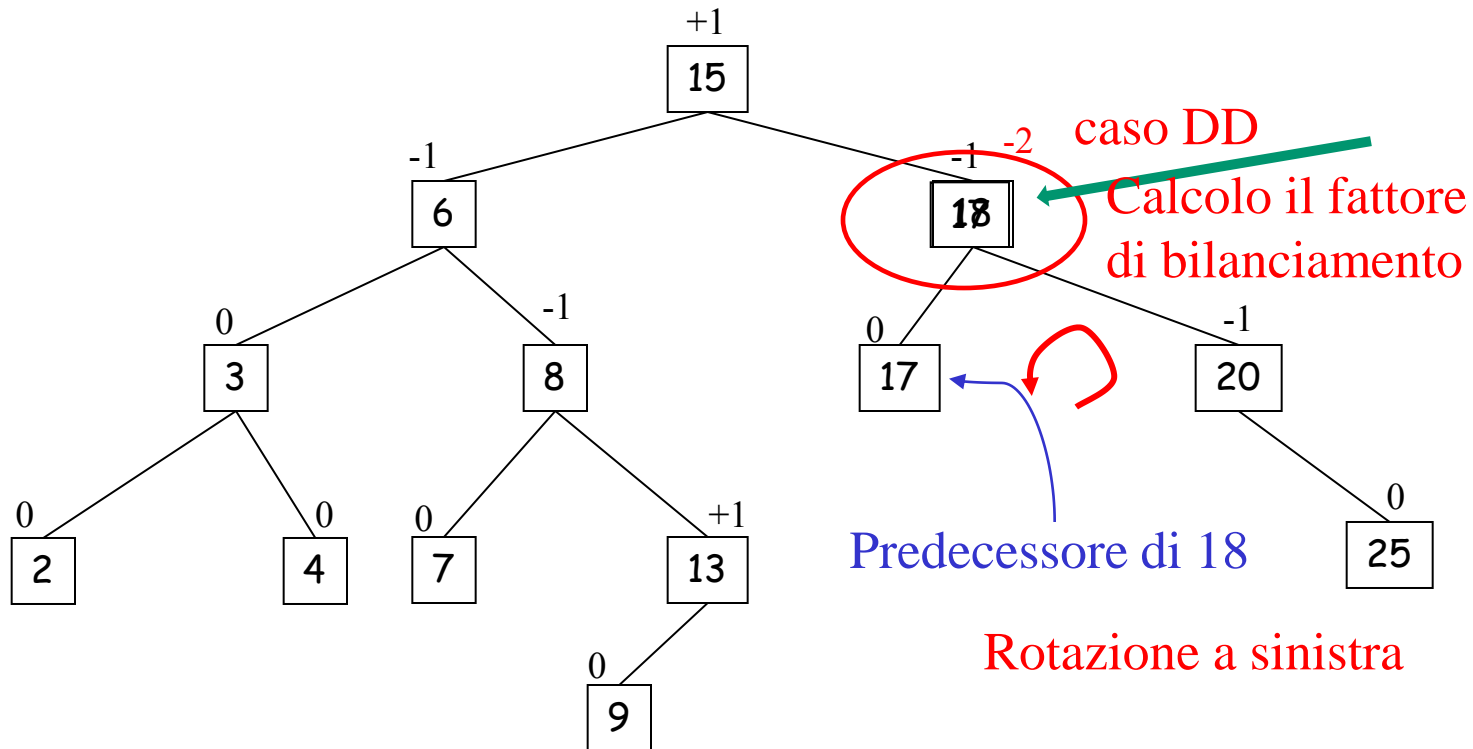
ricalcolo dei fattori di bilanciamento dei nodi coinvolti nella rotazione

delete(elem e)

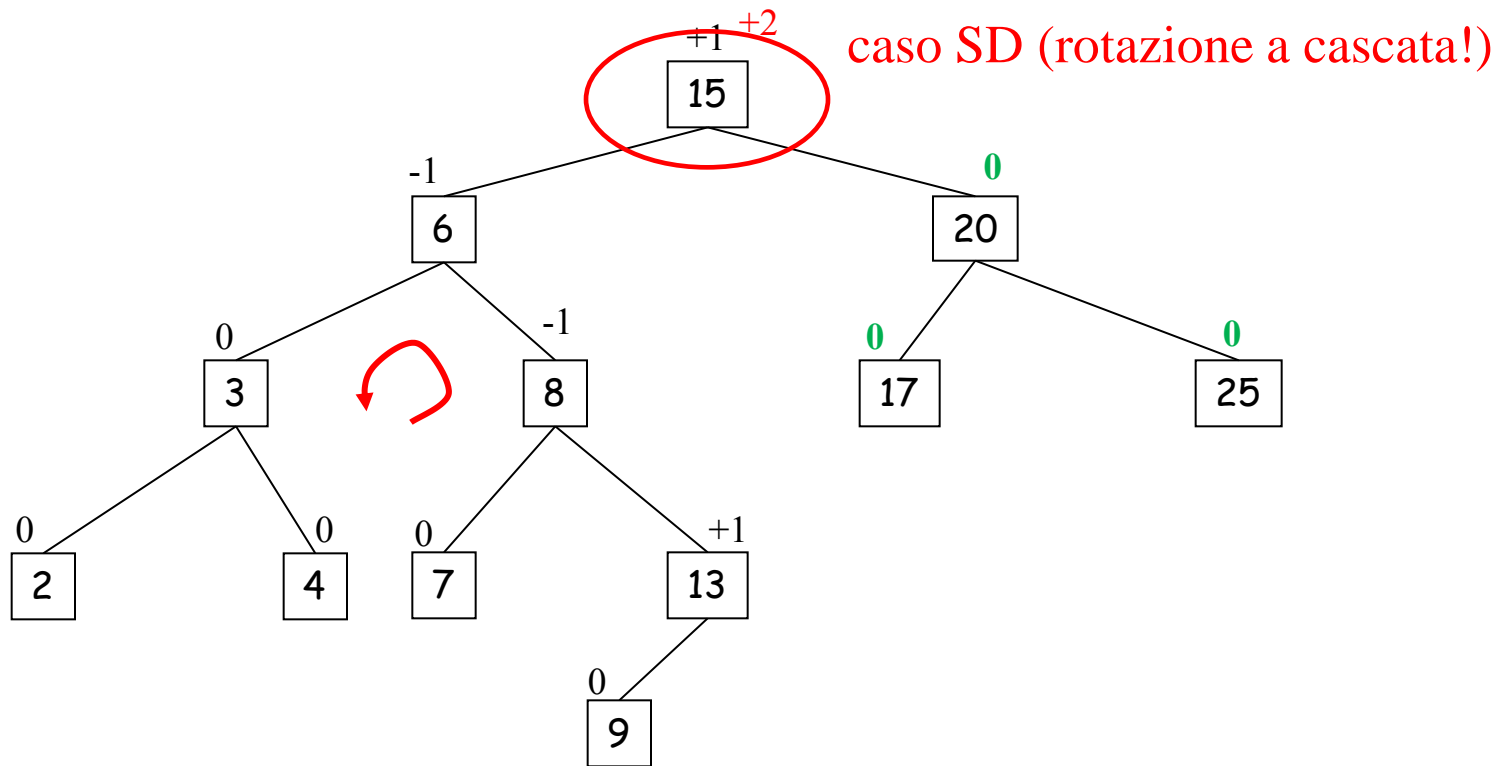
1. Cancella il nodo come in un ABR
2. Ricalcola il fattore di bilanciamento del **padre del nodo eliminato fisicamente** (che potrebbe essere quello contenente il predecessore di e , se e ha due figli), ed esegui l'opportuna rotazione semplice (casi SS e DD) o doppia (casi SD e DS) ove necessario
3. Se l'altezza del sottoalbero appena ribilanciato è uguale a quella che aveva prima della cancellazione (caso **SS \wedge SD** oppure **DD \wedge DS**, vedi tabella), termina. Invece, se tale altezza è diminuita, calcola il fattore di bilanciamento del nodo padre del sottoalbero appena ribilanciato, e se sbilanciato applica l'opportuno ribilanciamento e riesegui il passo 3.

Osservazione: potrebbero essere necessarie $O(\log n)$ rotazioni: infatti eventuali diminuzioni di altezza indotte dalle rotazioni (caso SS/DD, o caso SD/DS) possono propagare lo sbilanciamento verso l'alto nell'albero **fino alla radice** (l'altezza del sottoalbero critico in cui è avvenuto il ribilanciamento **diminuisce di 1** rispetto a quella che aveva **prima della cancellazione**)

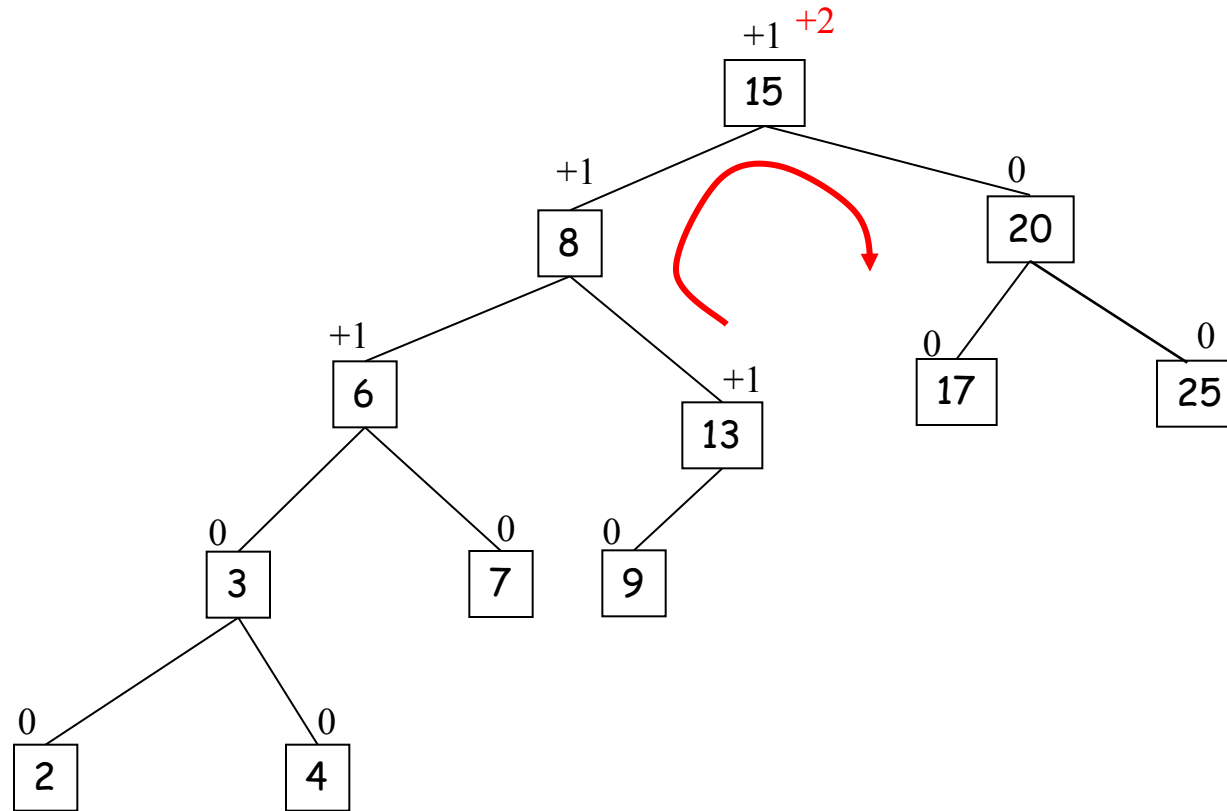
Esempio: delete (elemento con chiave 18)



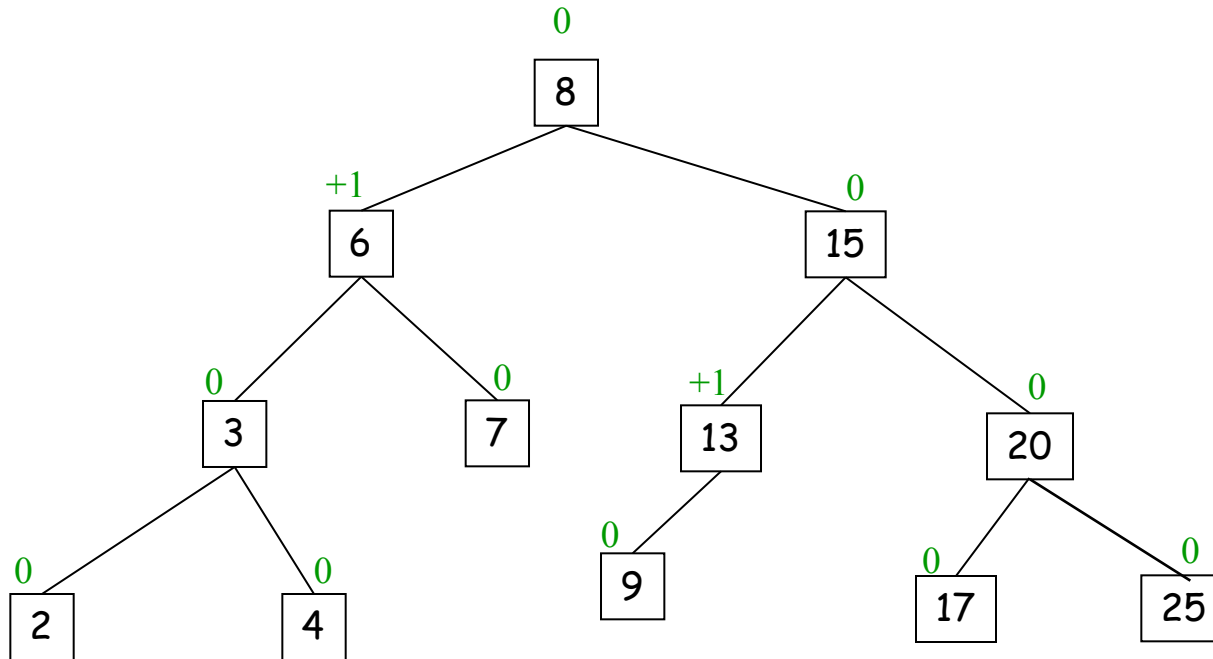
Ribilanciamento DD e aggiornamento del fattore di bilanciamento del padre del sottoalbero ruotato



Ribilanciamento SD



Albero ribilanciato



Classe AlberoAVL

classe AlberoAVL **estende** AlberoBinarioDiRicerca:

dati: $S(n) = O(n)$
 albero binario di ricerca T ereditato, più il fattore di bilanciamento di ogni nodo.

operazioni:

search(*chiave* k) \rightarrow *elem* $T(n) = O(\log n)$
 ereditata.

insert(*elem* e , *chiave* k) $T(n) = O(\log n)$
 chiama *insert*() ereditata, poi ricalcola i fattori di bilanciamento ed eventualmente ribilancia tramite $O(1)$ rotazioni.

delete(*elem* e) $T(n) = O(\log n)$
 chiama *delete*() ereditata, poi ricalcola i fattori di bilanciamento ed eventualmente ribilancia tramite $O(\log n)$ rotazioni.

- Tutte le operazioni hanno costo $O(\log n)$ poiché l'altezza dell'albero è $\Theta(\log n)$ e ciascuna rotazione richiede solo **tempo costante**
- **Approfondimento:** Dimostrare che la *insert* costa $\Theta(\log n)$

Riepilogo

- **Mantenere il bilanciamento** è risultato cruciale per ottenere buone prestazioni
- Esistono vari approcci per mantenere il bilanciamento:
 - Tramite **rotazioni** (alberi AVL)
 - Tramite **fusioni o separazioni** di nodi (alberi 2-3, B-alberi)
- In tutti questi casi si ottengono tempi di esecuzione logaritmici nel caso peggiore