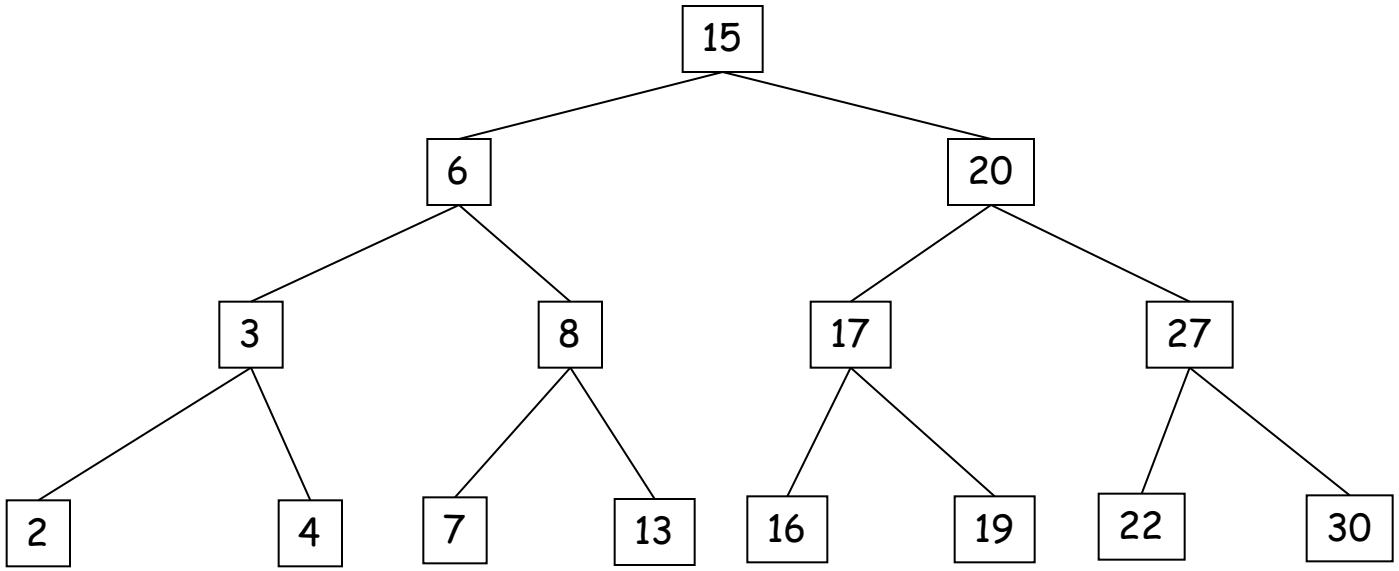
Algoritmi e Strutture Dati

Capitolo 6
Il problema del dizionario: gli alberi AVL

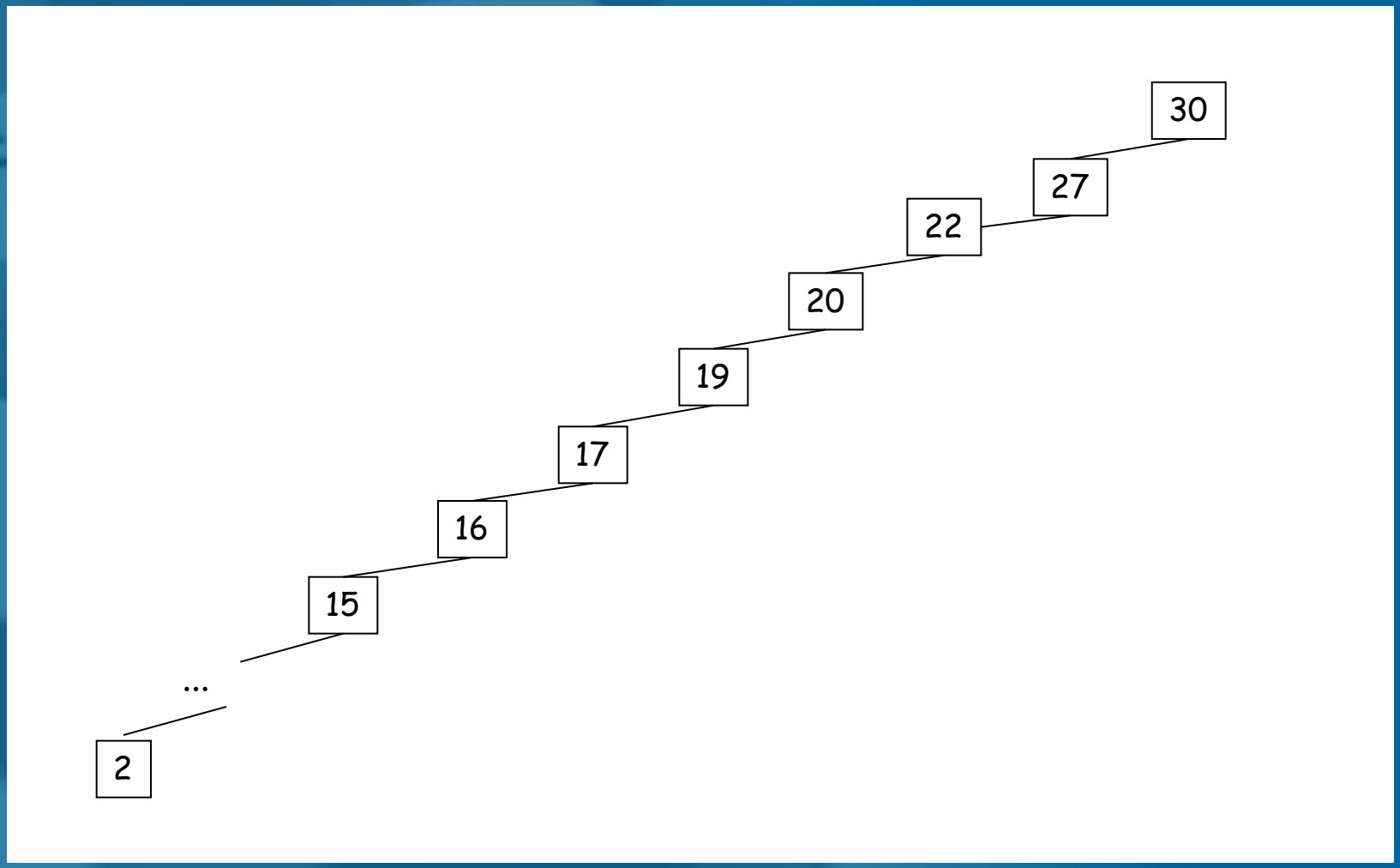
Riepilogo

- **Tipo dato Dizionario:** Insieme di coppie (elemento, chiave), in cui la chiave (univoca) appartiene ad un dominio totalmente ordinato, sul quale eseguire operazioni di search, insert e delete.
- **Implementazioni elementari (array, liste)**: inefficienti (costo $O(n)$)
- **Obiettivo**: implementazione in $O(\log n)$
- **Albero binario di ricerca**: implementazione del dizionario mediante un albero in cui ogni nodo v contiene una coppia (elemento, chiave) del dizionario, con la proprietà (che induce un ordinamento totale) che:
  - le chiavi nel sottoalbero sinistro di v sono < chiave(v)
  - le chiavi nel sottoalbero destro di v sono > chiave(v)
- Operazioni di **search**, **insert** e **delete** sull'ABR costano $O(h)$, ove h è l'altezza dell'albero, ma l'altezza dell'ABR può variare da $\Theta(\log n)$ a $\Theta(n)$

Un albero binario di ricerca di altezza $\Theta(\log n)$



Un albero binario di ricerca di altezza $\Theta(n)$



Analisi critica degli ABR

- Le operazioni di inserimento e cancellazione descritte possono "linearizzare" un ABR.

- Esempi: Supponiamo di inserire un elemento con chiave minore della chiave minima dell'ABR, poi un altro elemento con chiave ancora minore, e cosi via, oppure supponiamo di cancellare da un ABR binario completo tutti gli elementi uno dopo l'altro, lasciando solo una costola dell'albero

$\Rightarrow$ Dobbiamo definire un modo per **mantenere** l'albero "bilanciato" (vogliamo cioè che per ogni nodo interno, le "dimensioni" dei sottoalberi sinistro e destro associati rimangano approssimativamente uguali)

$\Rightarrow$ Innanzitutto dobbiamo formalizzare il concetto di **bilanciamento**

Alberi AVL (*)
(Adel'son-Vel'skii e Landis, 1962)

Formalizzazione del bilanciamento

Fattore di bilanciamento $\beta(v)$ di un nodo v =

altezza del sottoalbero sinistro di v –

altezza del sottoalbero destro di v

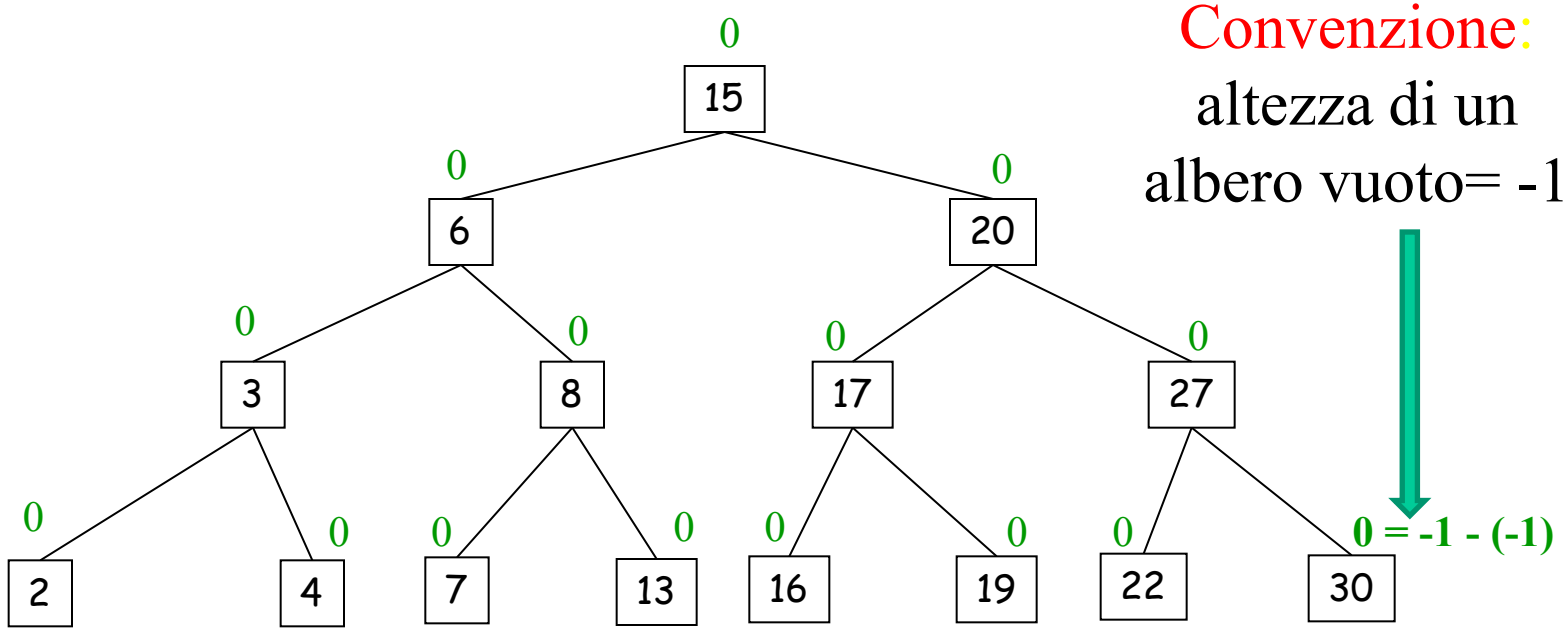
($\beta(v)$ viene mantenuto come informazione addizionale nel record associato a v, assieme alle altezze del sottoalbero sinistro e destro di v)

Def.: Un albero si dice bilanciato in altezza se ogni nodo v ha fattore di bilanciamento in valore assoluto ≤ 1

Alberi AVL = alberi binari di ricerca bilanciati in altezza

…qualche esempio…

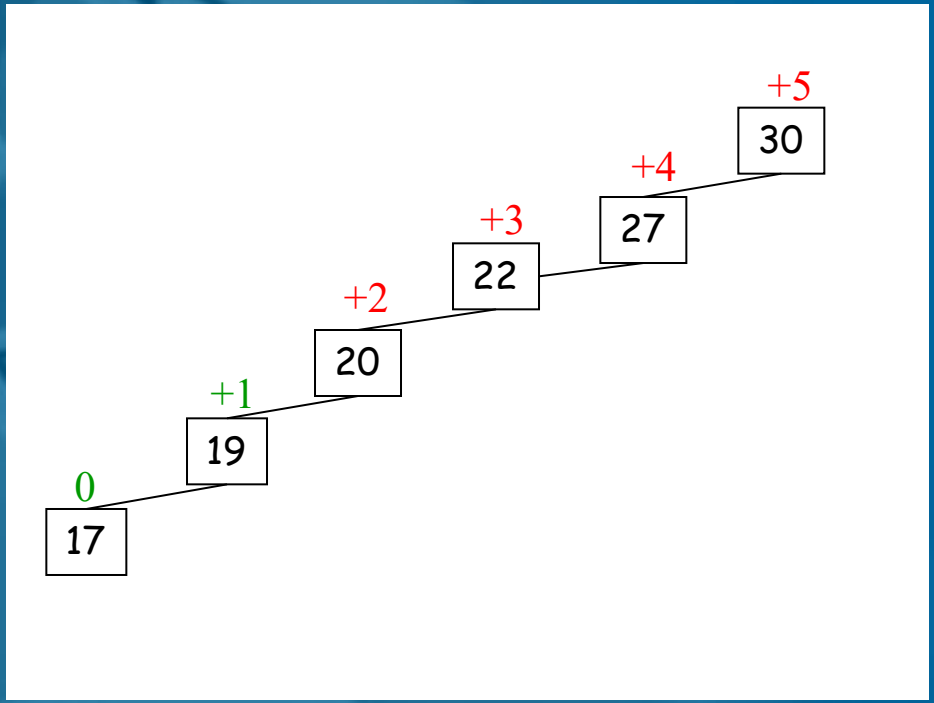
È il seguente un albero AVL?



Sì: è un ABR e tutti i nodi hanno fattore di bilanciamento = 0

…qualche esempio…

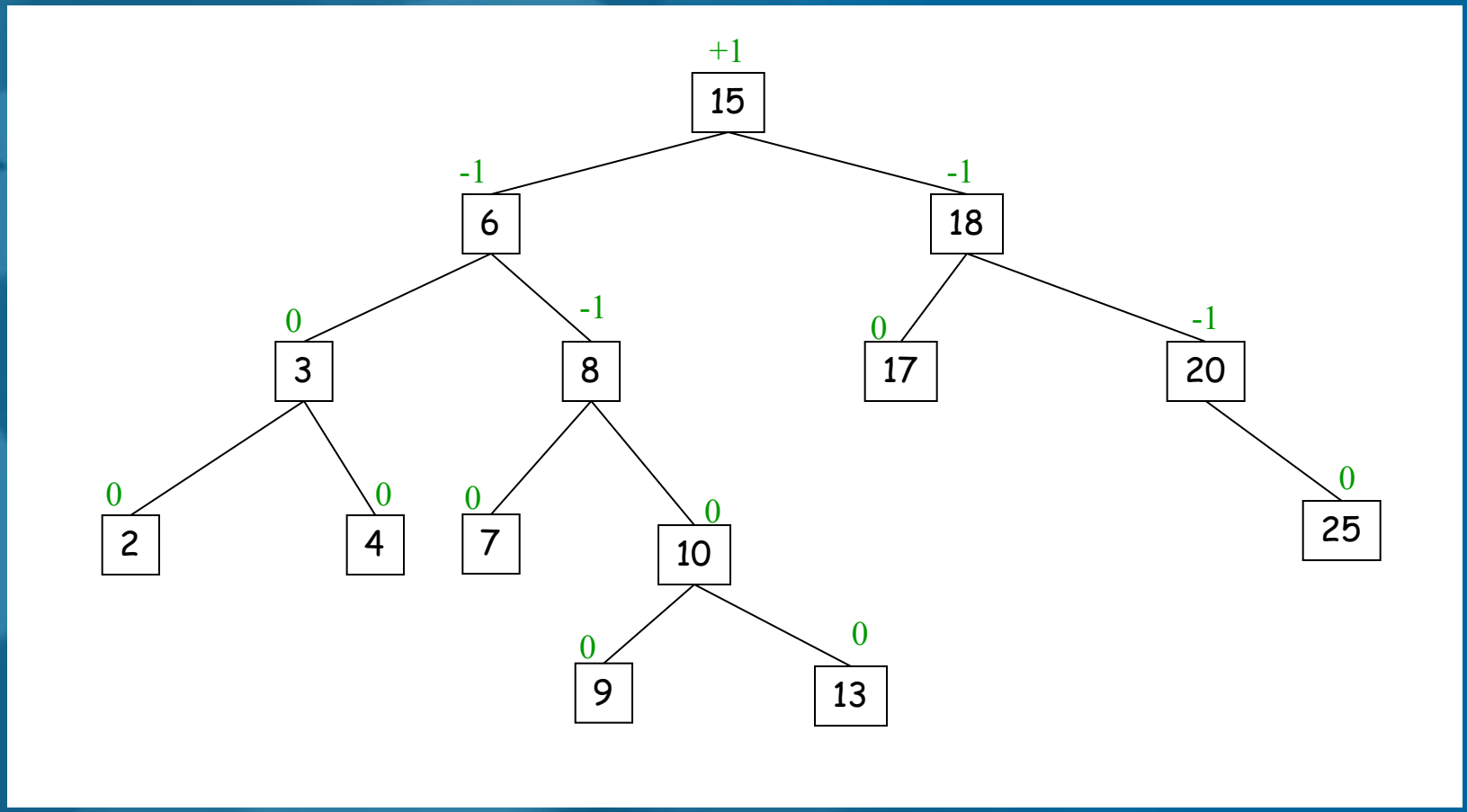
È il seguente un albero AVL?



NO! È un ABR, ma non vale la proprietà sui fattori di bilanciamento!

È il seguente un albero AVL?

…qualche esempio…



Sì: È un ABR e la proprietà sui fattori di bilanciamento è rispettata

Delimitazione superiore all'altezza di alberi AVL

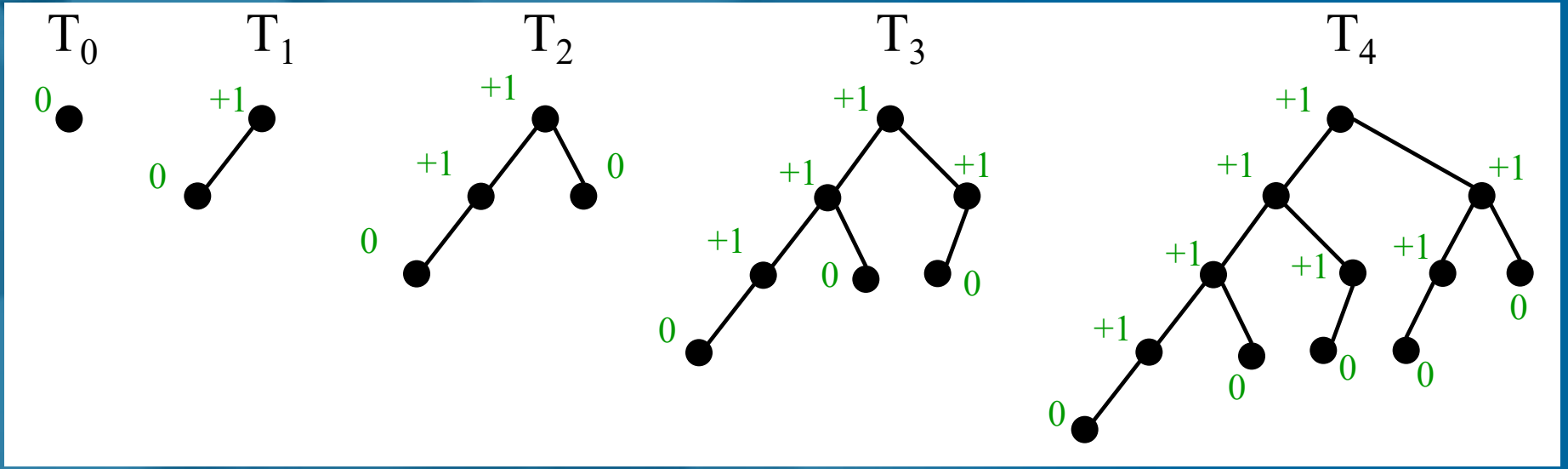
Vogliamo dimostrare che un albero AVL con n nodi ha altezza O(log n)

Idea della dimostrazione: considerare, tra tutti gli AVL di altezza h, quelli con il minimo numero di nodi n_h (che vengono detti alberi di Fibonacci), e dimostrare che $h=O(\log n_h)$

Intuizione: se per gli alberi di Fibonacci di altezza h vale $h=O(\log n_h)$, allora per tutti gli alberi AVL di altezza h con $n \geq n_h$ nodi varrà $h=O(\log n_h)=O(\log n)$

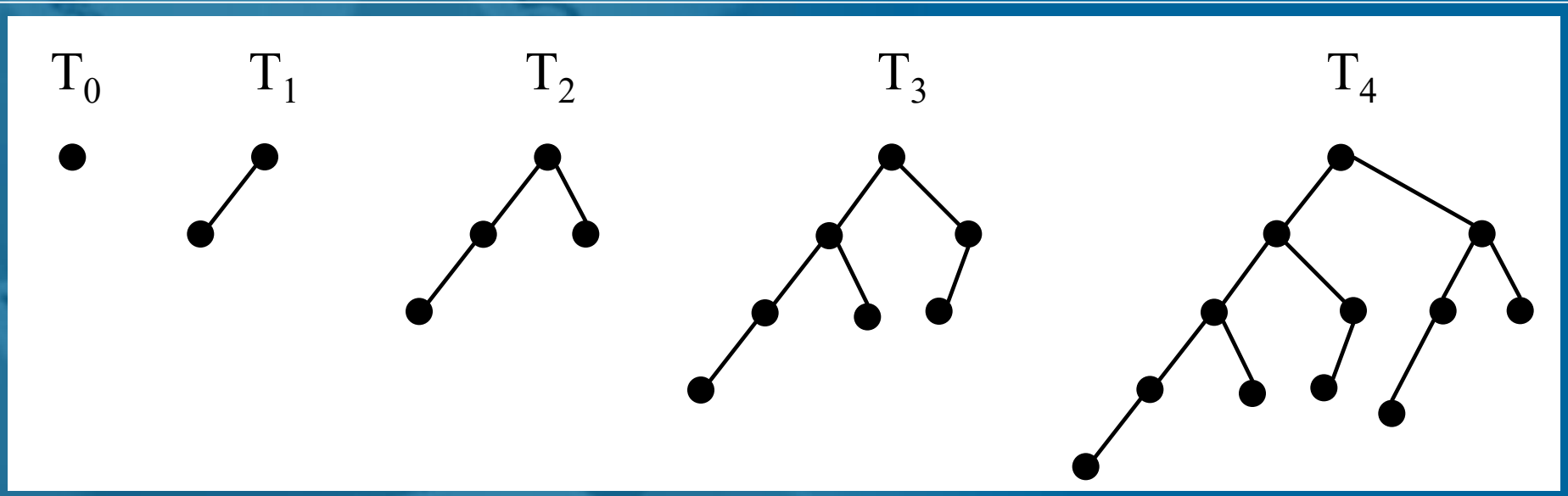
…Alberi di Fibonacci per piccoli valori di altezza…

T_h (albero di Fibonacci di altezza h): albero AVL di altezza h con il minimo numero di nodi $\Rightarrow$ devo massimizzare ad 1 il fattore di bilanciamento di ogni nodo interno

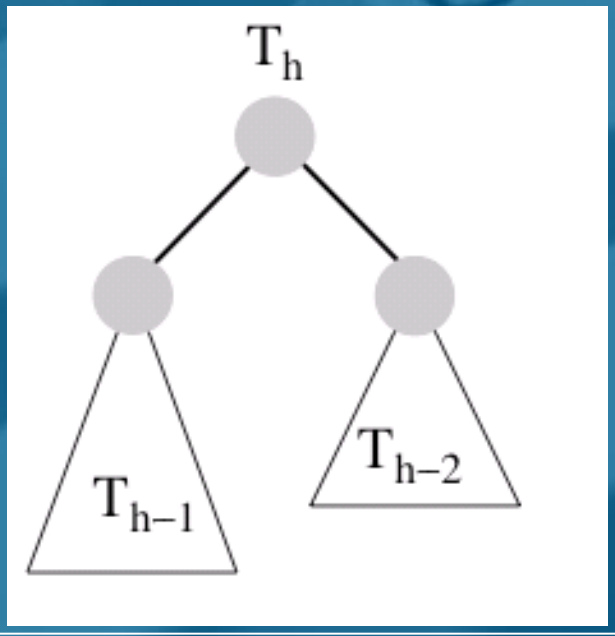


Nota: se a T_h tolgo una qualsiasi foglia (esclusa quella più in basso, che ne caratterizza l'altezza), diventa sbilanciato!

intravedete uno schema per generare l'i-esimo albero di Fibonacci a partire dai precedenti?



Lo schema



Lemma Sia n_h il numero di nodi di T_h. Risulta $n_h = F_{h+3} - 1$.

Dim.: Per induzione su h:

- $h=0$: $n_0 = 1$ $F_{h+3} - 1 = F_3 - 1 = 2 - 1 = 1$
- h generico:

$n_h = 1 + n_{h-1} + n_{h-2} = 1 + (F_{(h-1)+3} - 1) + (F_{(h-2)+3} - 1)$
$= 1 + (F_{h+2} - 1) + (F_{h+1} - 1) = F_{h+3} - 1.$ ■

Teorema:

Sia T un albero AVL con n nodi e di altezza h. Allora, $h=O(\log n)$.

Dim.: Si consideri l'albero di Fibonacci T_h; dal lemma

$n_h = F_{h+3} - 1 = \Theta(\phi^{h+3}) = \Theta(\phi^h)$

Ricorda che vale:

$F_h = \Theta(\phi^h)$

$\phi = 1.618\ldots$ sezione aurea

$h = \Theta(\log_\phi n_h) = \Theta(\log n_h)$,

e quindi, per l'albero T, poiché $n \geq n_h$

$h = O(\log n)$.

Delimitazione inferiore all'altezza di alberi AVL

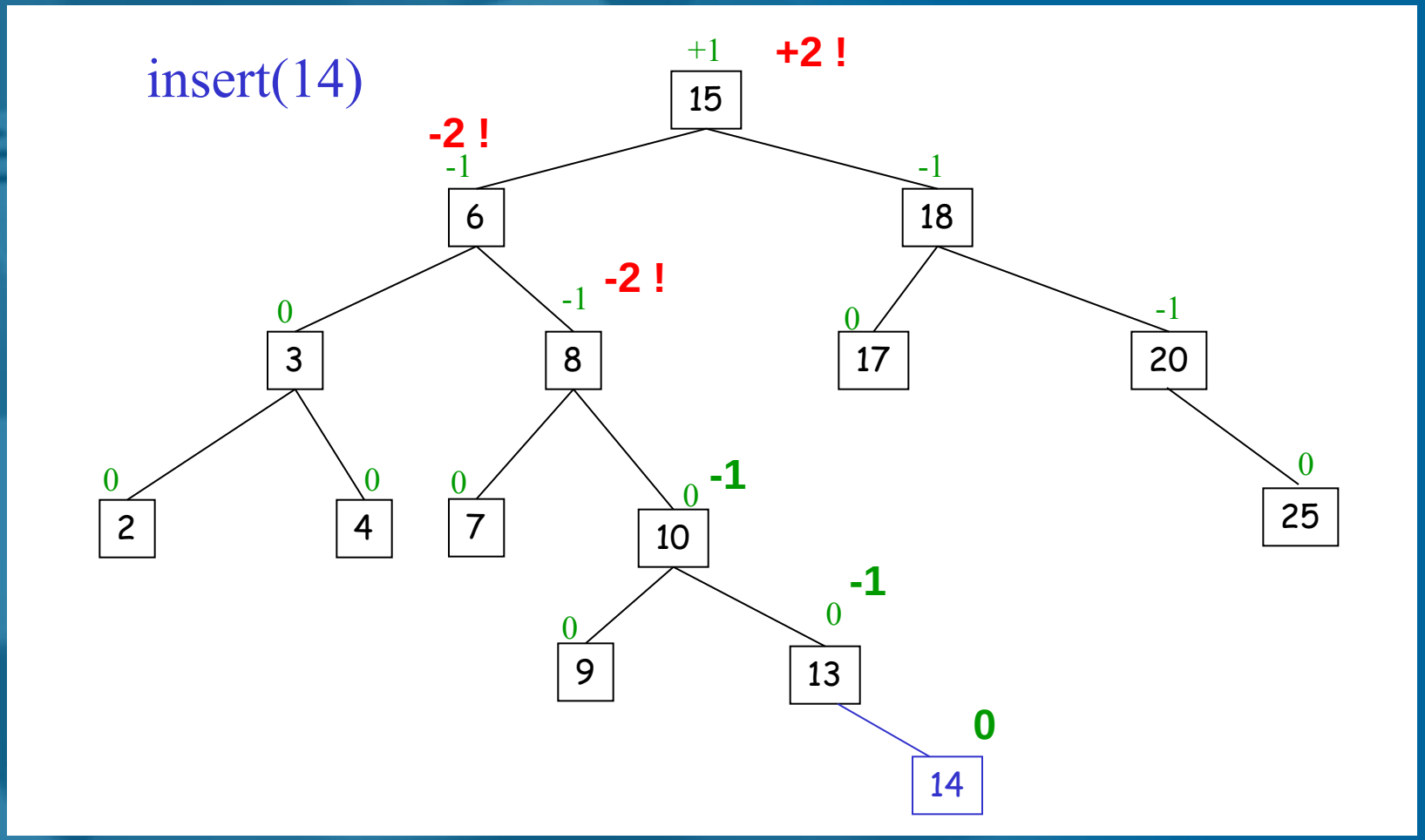
In linea di principio, l'altezza dell'AVL potrebbe essere $o(\log n)$. Tuttavia, ciò è falso, in quanto ogni albero binario di n nodi deve avere altezza $\Omega(\log n)$, poiché la minima altezza si ottiene quando l'albero binario è completo, e in tal caso sappiamo che $n=2^{h+1}-1$, cioè $h=\Theta(\log n)$.

$\Rightarrow$ Possiamo concludere che poiché un albero AVL con n nodi ha altezza $h=O(\log n)$ e $h=\Omega(\log n)$, ne consegue che $h=\Theta(\log n)$.

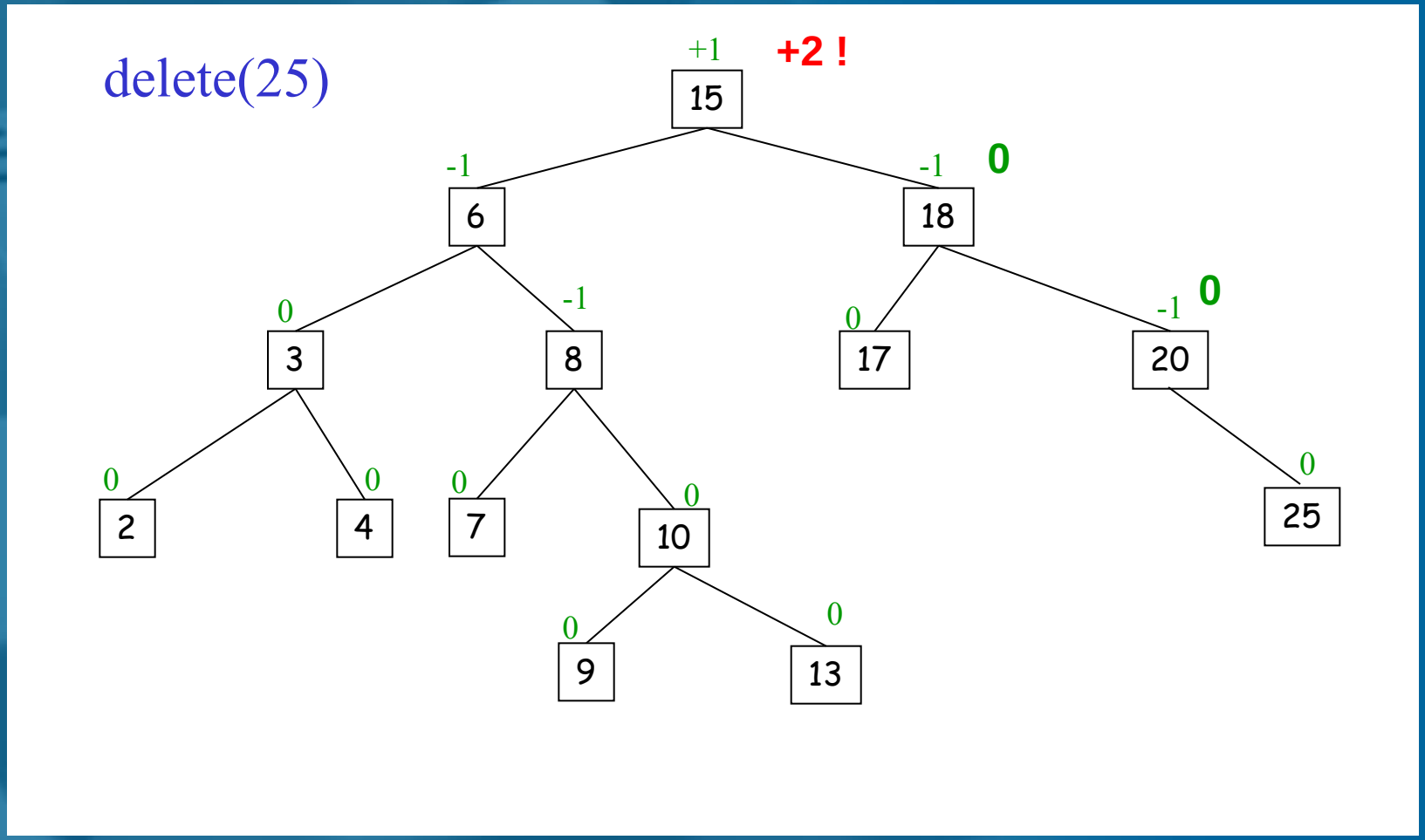
Implementazione delle operazioni negli AVL

- 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 tramite opportune rotazioni all'interno dell'AVL

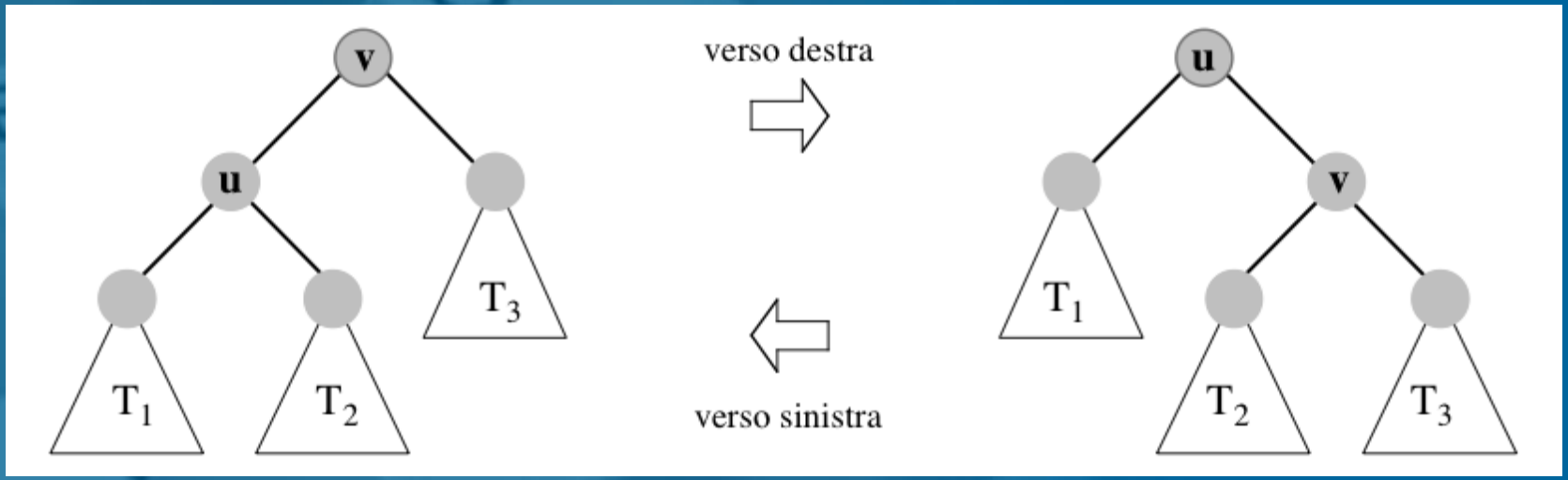
Gli inserimenti possono sbilanciare l'AVL



Le cancellazioni possono sbilanciare l'AVL



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:

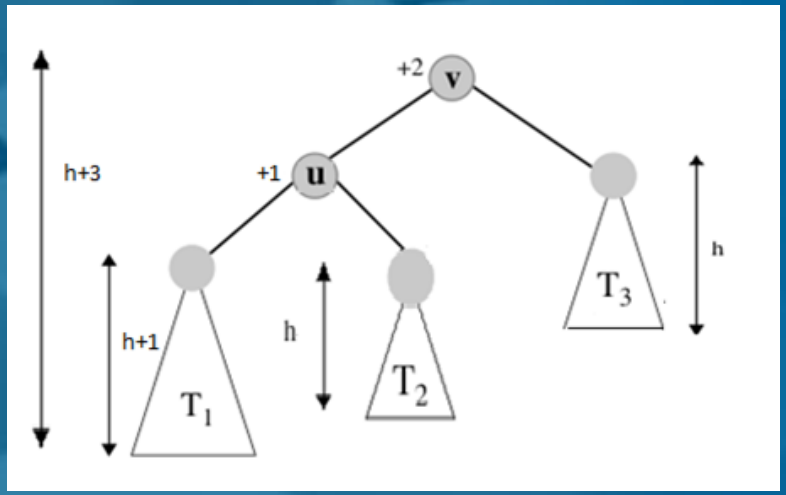
$\beta(v)=+2$ $\beta(v)=-2$

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

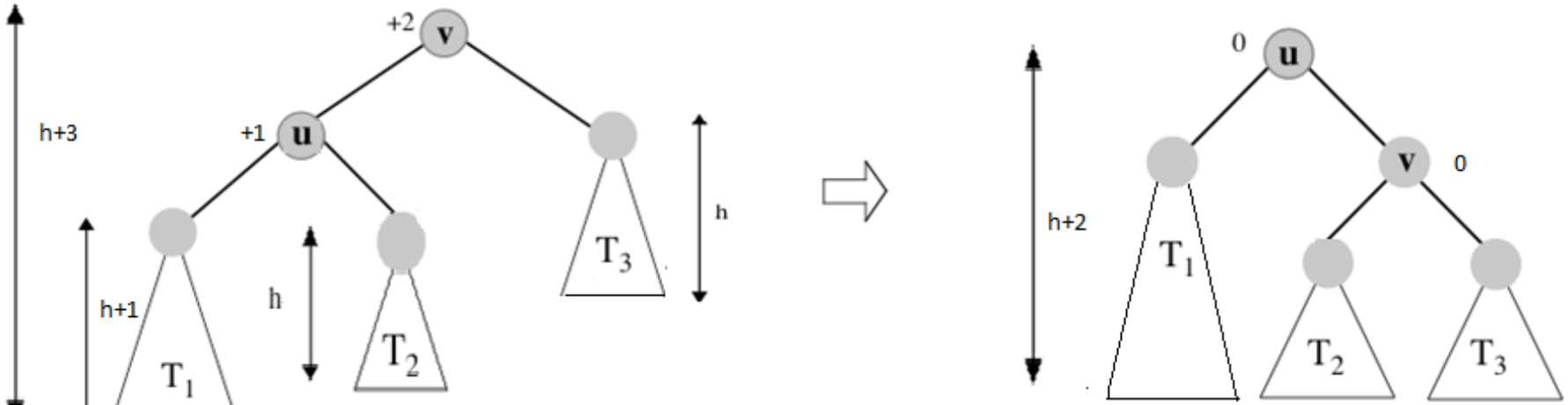
Caso SS

- 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

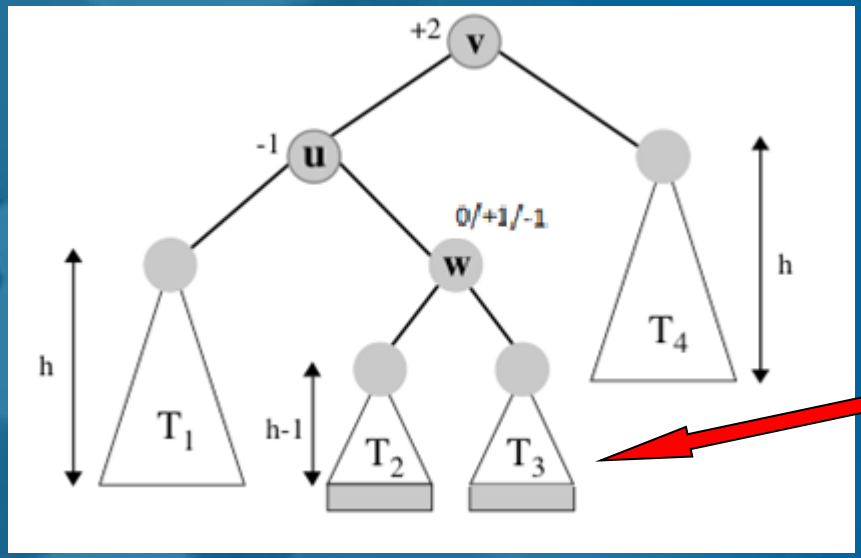
Rotazione nel caso SS



- 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

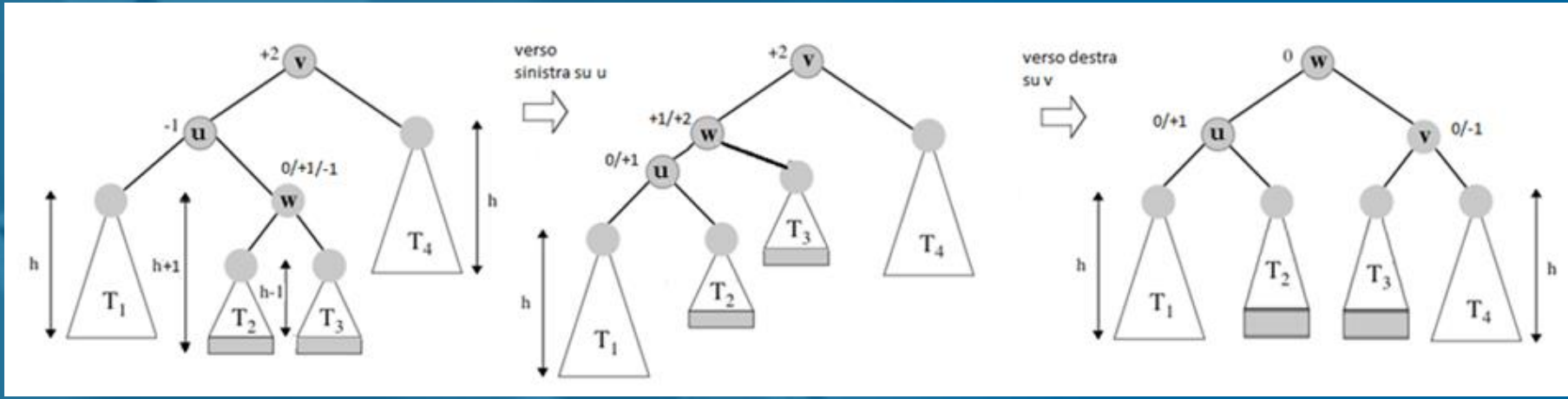
- L'altezza di T(v) è h+3, l'altezza di T(u) è h+2, l'altezza di T_1 è h, l'altezza di T_4 è h, e l'altezza di T(w) è h+1 $\Rightarrow$ $\beta(v)=+2$, e $\beta(u)=-1$ cioè lo sbilanciamento è provocato dal sottoalbero destro di u



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)

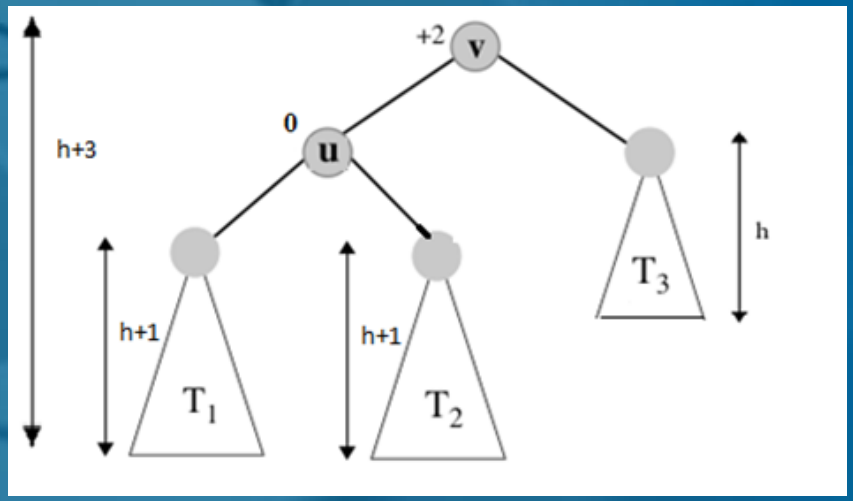
Rotazione nel caso SD



- 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.

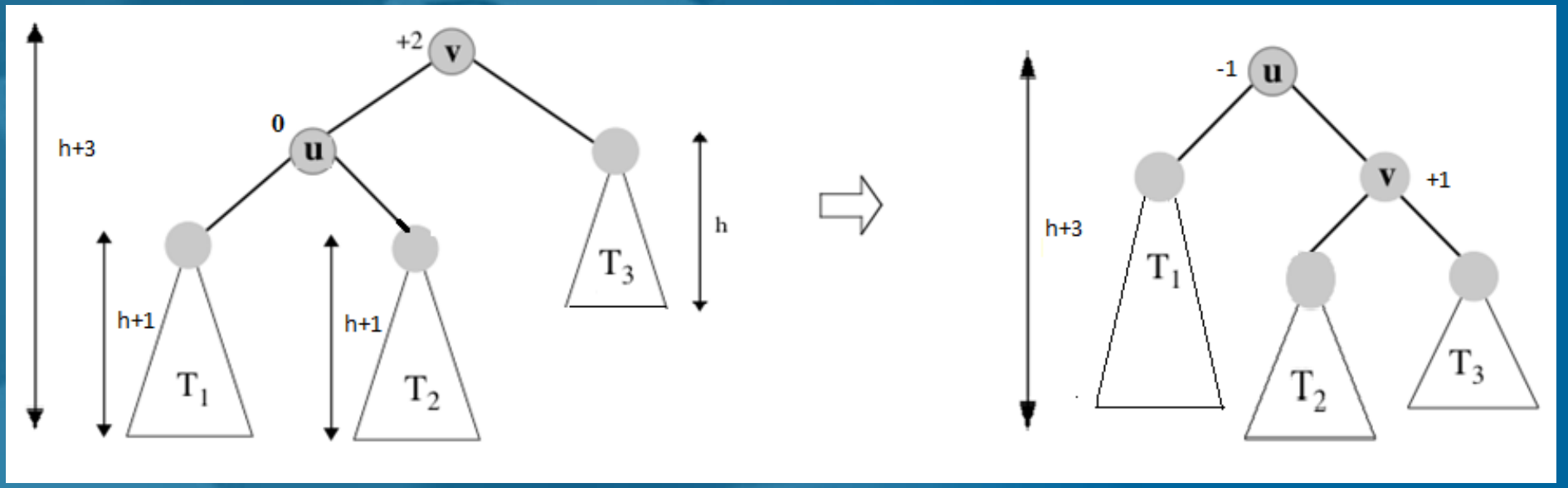
Casi SS e SD contemporanei

- 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 ∧ 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∧SD	Cancellazione	x	x	x

Ovviamente i casi DD, DS e DD∧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

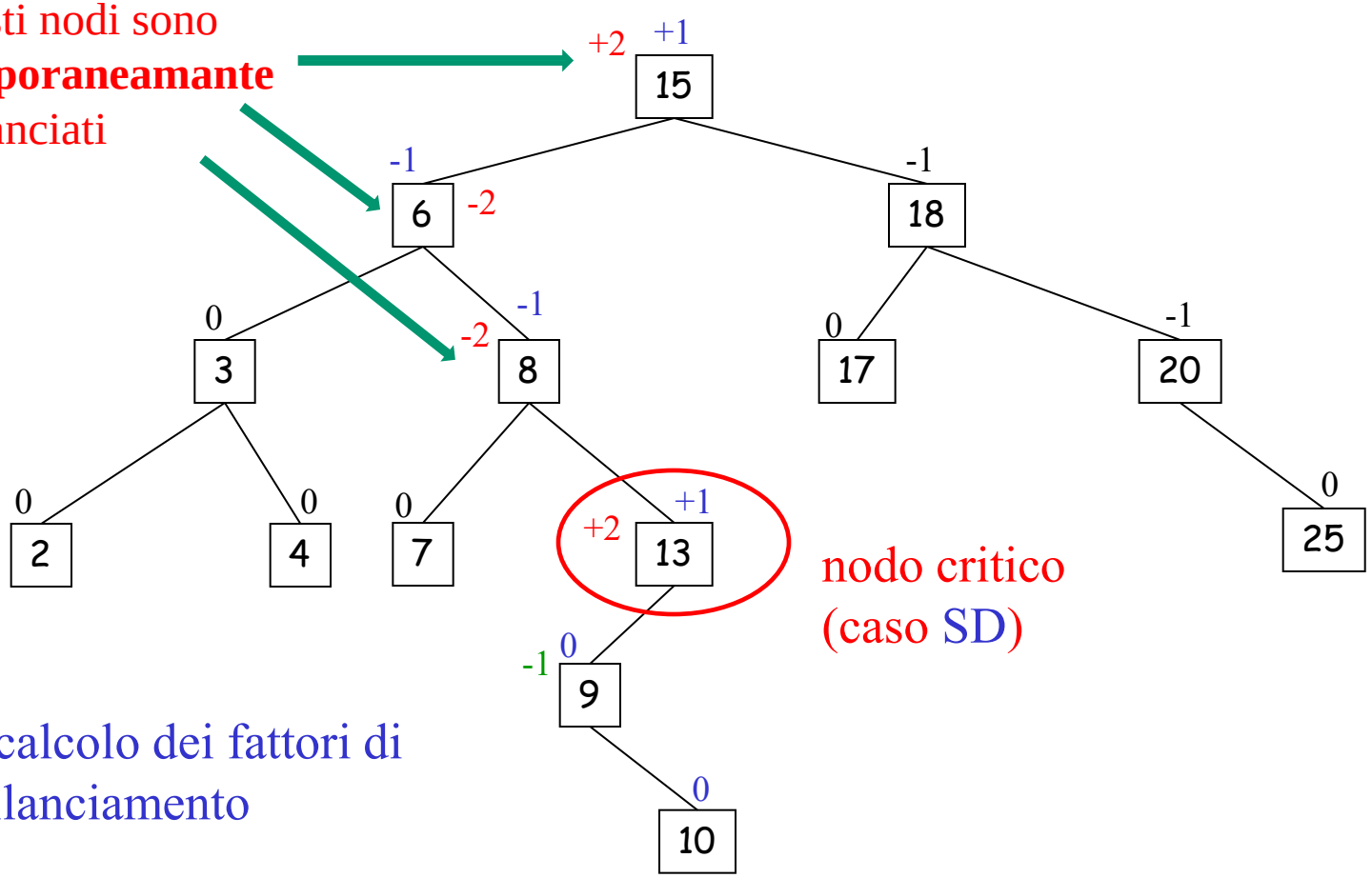
insert(elem e, chiave k)

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

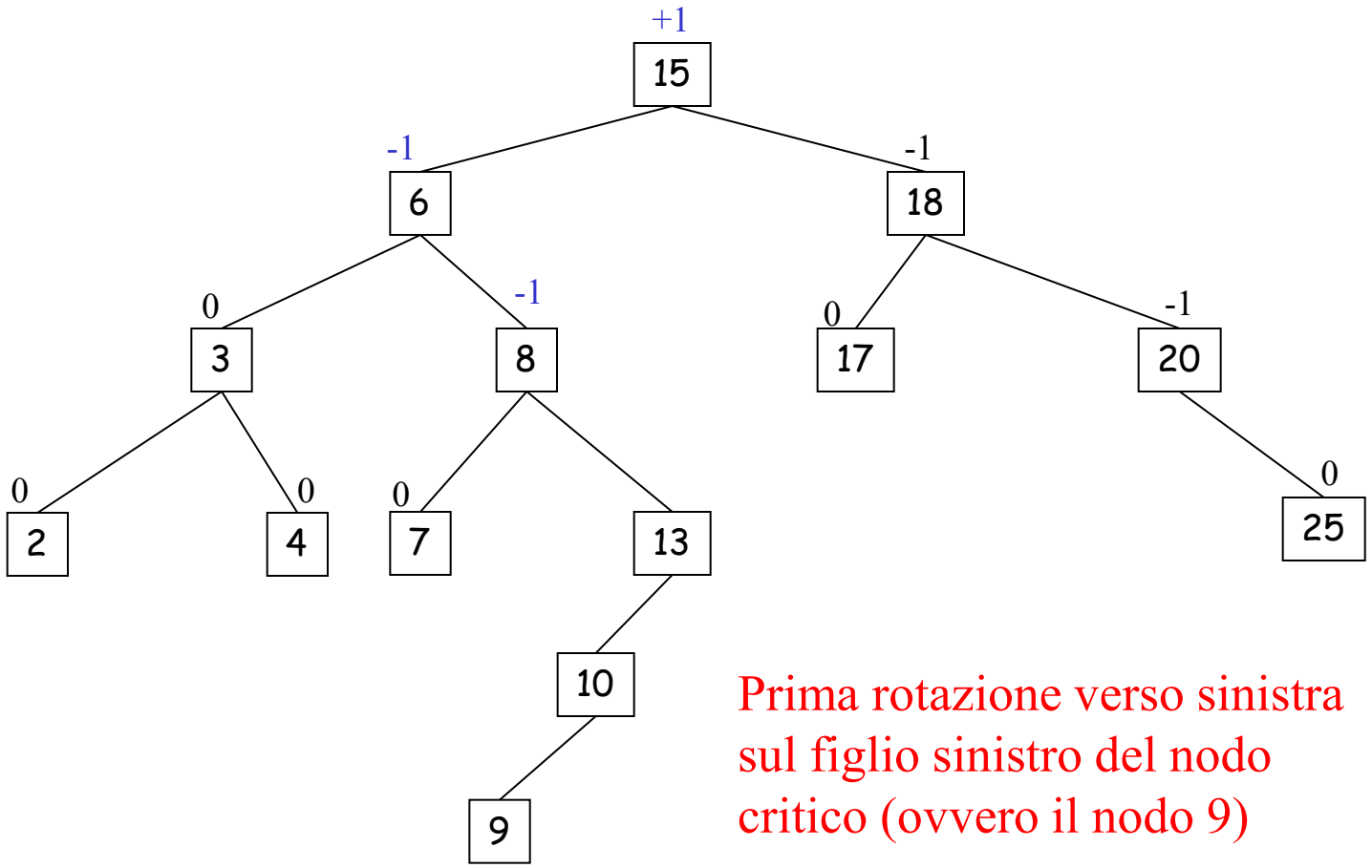
Osservazione: un solo ribilanciamento è sufficiente, poiché l'altezza del sottoalbero critico coinvolto nella/e rotazione/i torna ad essere uguale all'altezza che aveva prima dell'inserimento (vedi tabella precedente), ribilanciando così automaticamente tutti i nodi eventualmente sbilanciati nel cammino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

Esempio: insert (10,e)

Si noti che anche questi nodi sono **temporaneamante** sbilanciati



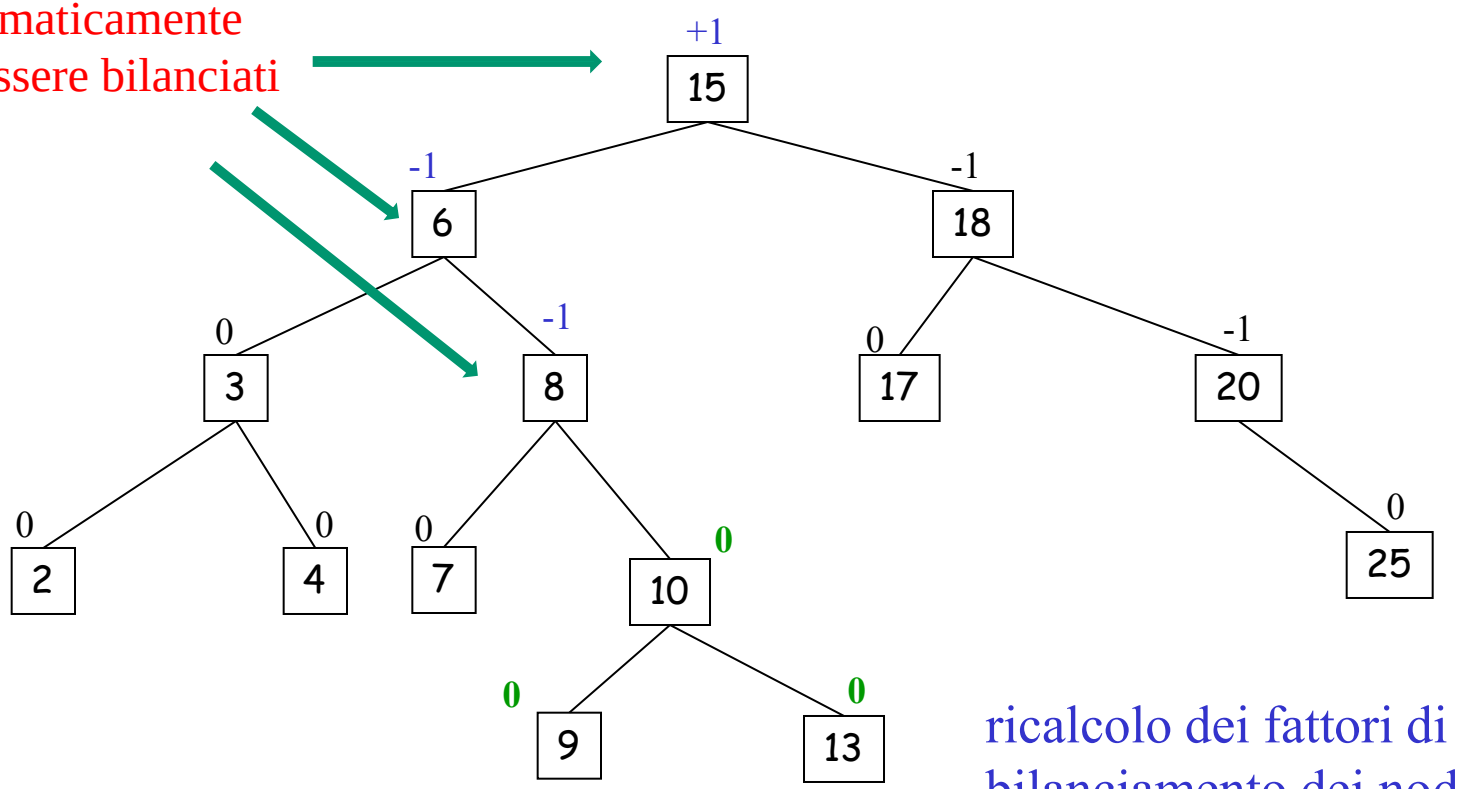
Esempio: insert (10,e)



Prima rotazione verso sinistra sul figlio sinistro del nodo critico (ovvero il nodo 9)

Esempio: insert (10,e)

Questi nodi tornano automaticamente ad essere bilanciati



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

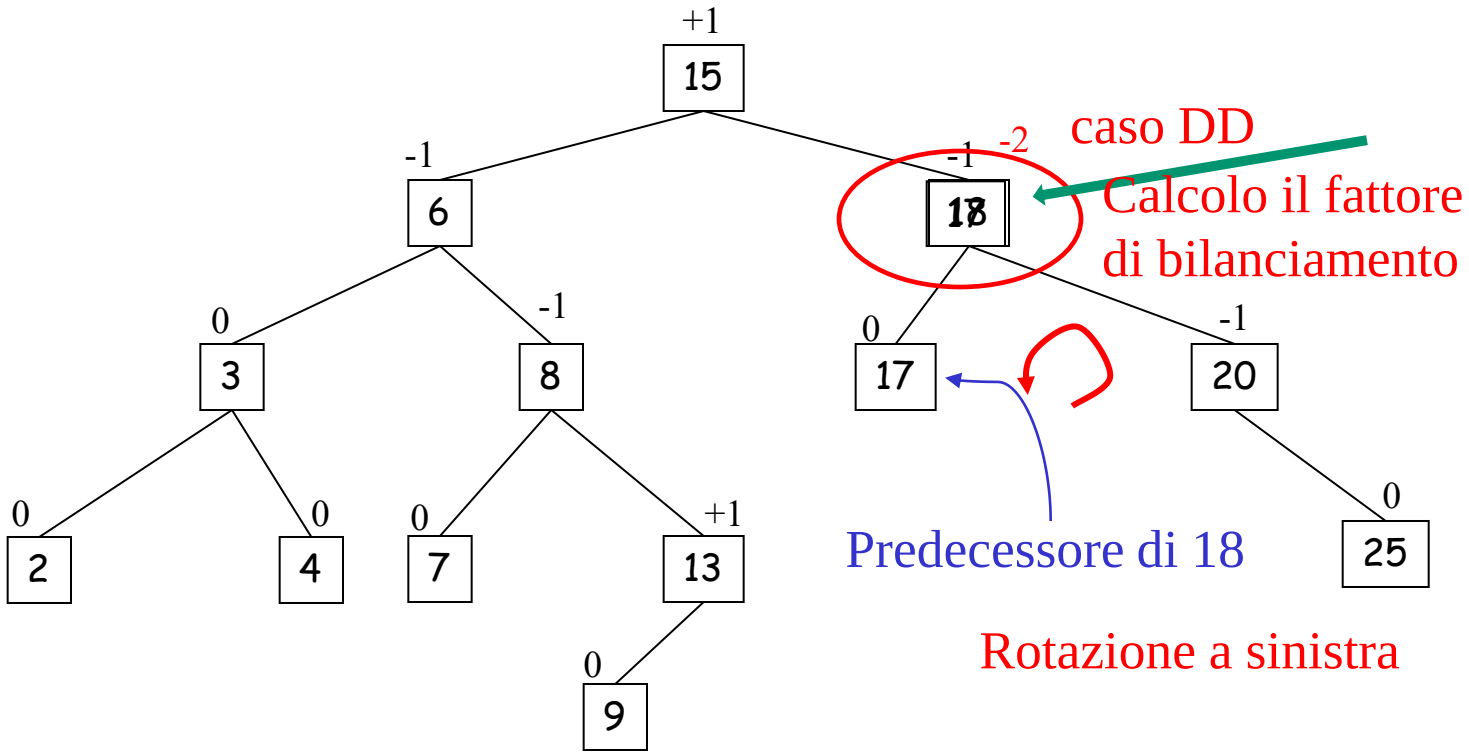
ricalcolo dei fattori di bilanciamento dei nodi coinvolti nella rotazione

delete(elem e)

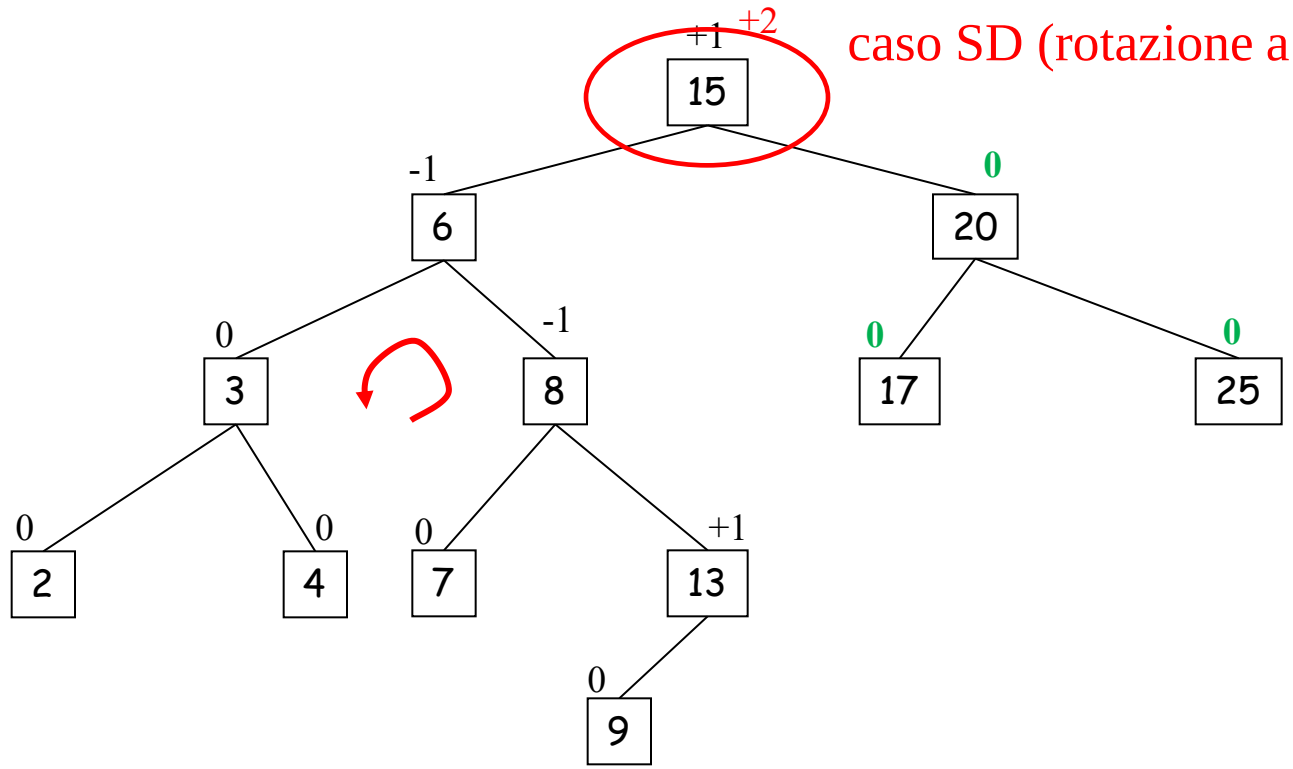
1. Cancella il nodo come in un ABR
2. Ricalcola il fattore di bilanciamento del padre del nodo eliminato fisicamente (che potrebbe essere quello contenente il predecessore di e, se e ha due figli), ed esegui l'opportuna rotazione semplice (casi SS e DD) o doppia (casi SD e DS) ove necessario
3. Se l'altezza 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)

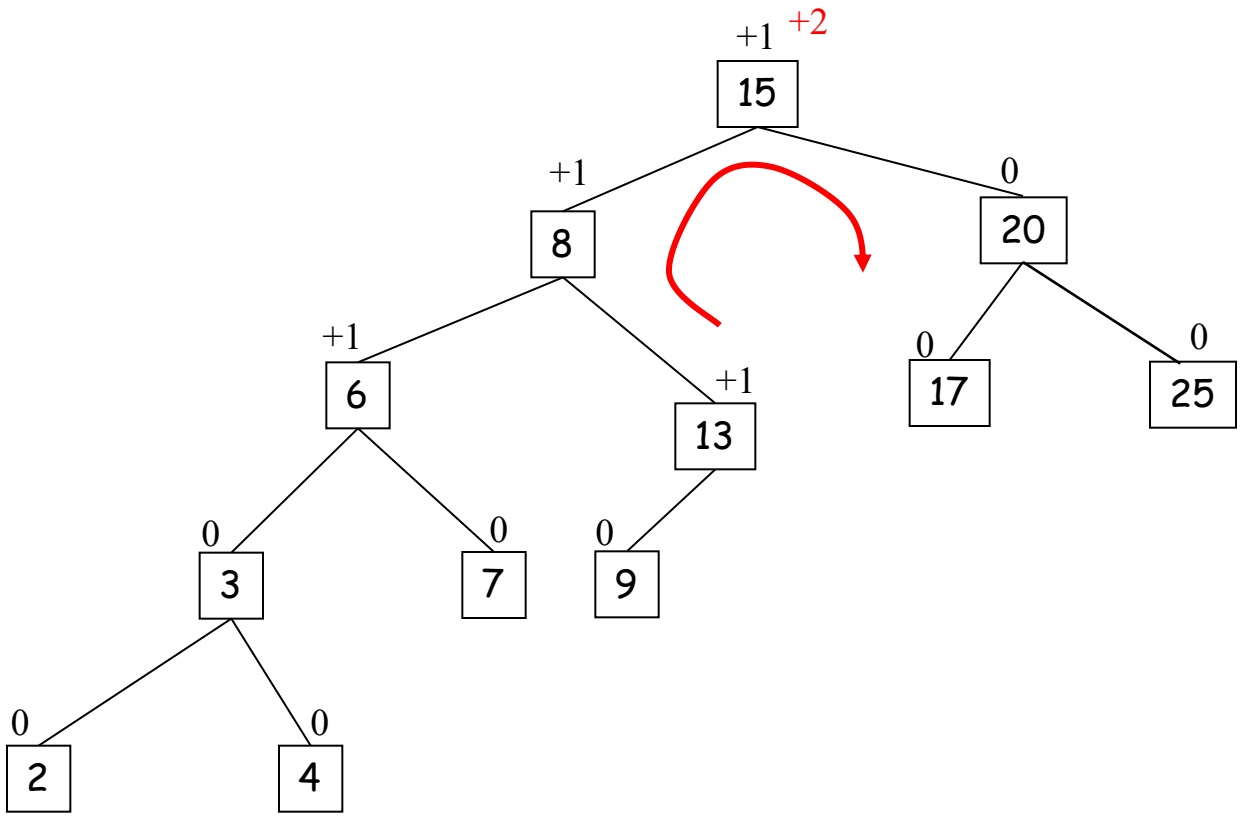
Esempio: delete (elemento con chiave 18)



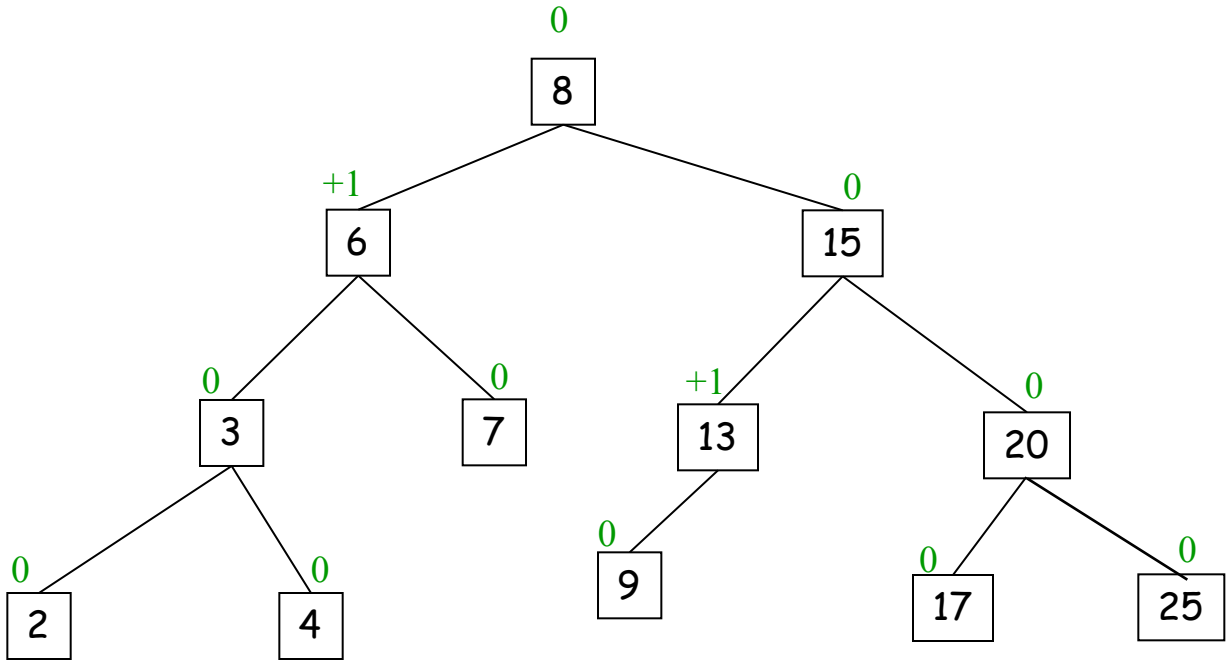
Ribilanciamento DD e aggiornamento del fattore di bilanciamento del padre del sottoalbero ruotato



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 Θ(log n) e ciascuna rotazione richiede solo tempo costante
- **Approfondimento**: Dimostrare che la insert costa Θ(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