Alberi Bilanciati di Ricerca

Damiano Macedonio
Università Ca' Foscari di Venezia

mace@unive.it

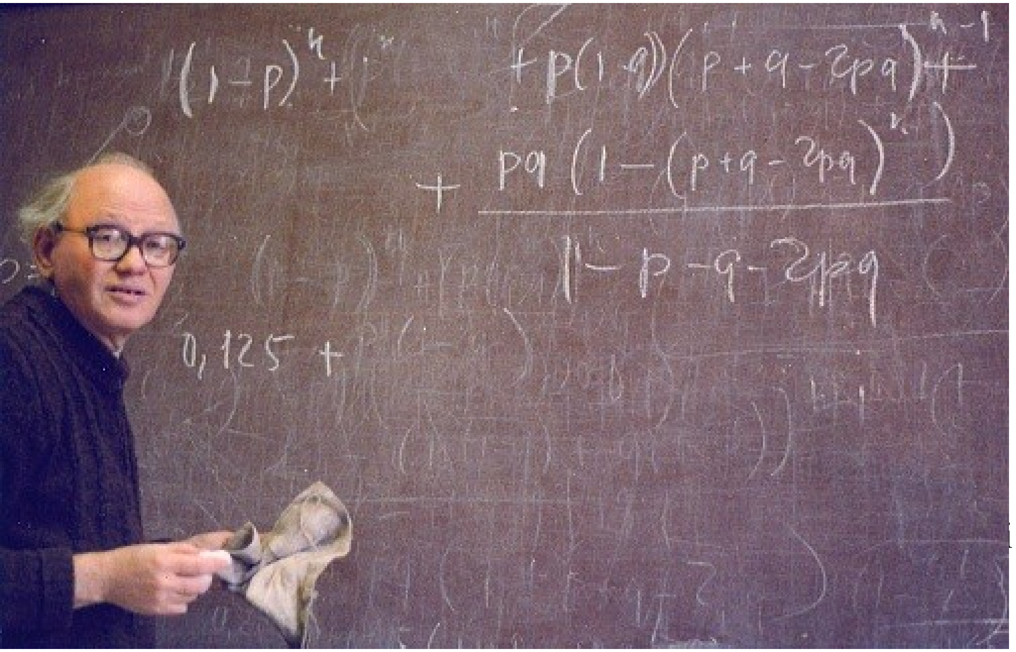
Introduzione

- Abbiamo visto che in un ABR è possibile inserire, rimuovere e individuare nodi data la corrispondente chiave in tempo $O(h)$ con h=altezza dell'albero
  - Un albero binario completo con n nodi ha altezza $h=\Theta(\log n)$
- Tuttavia, inserimenti e rimozioni di nodi possono “sbilanciare” l'albero
  - Domanda: individuare una sequenza di *n* inserimenti in un ABR inizialmente vuoto tali che al termine, l'albero risultante abbia altezza $\Theta(n)$
- Il nostro obbiettivo: mantenere bilanciato un ABR, anche a seguito di inserimenti/rimozioni di nodi

Alberi AVL

- Un albero AVL è un albero di ricerca (quasi) bilanciato
  - Un albero AVL con n nodi supporta le operazioni insert(), delete(), lookup() con costo O(log n) nel caso pessimo
  - Adelson-Velskii, G.; E. M. Landis (1962). *"An algorithm for the organization of information"*. Proceedings of the USSR Academy of Sciences 146: 263–266

Georgy Maximovich Adelson-Velsky (1922—)
http://chessprogramming.wikispaces.com/Georgy+Adelson-Velsky



Evgenii Mikhailovich Landis (1921—1997)
http://en.wikipedia.org/wiki/Yevgeniy_Landis

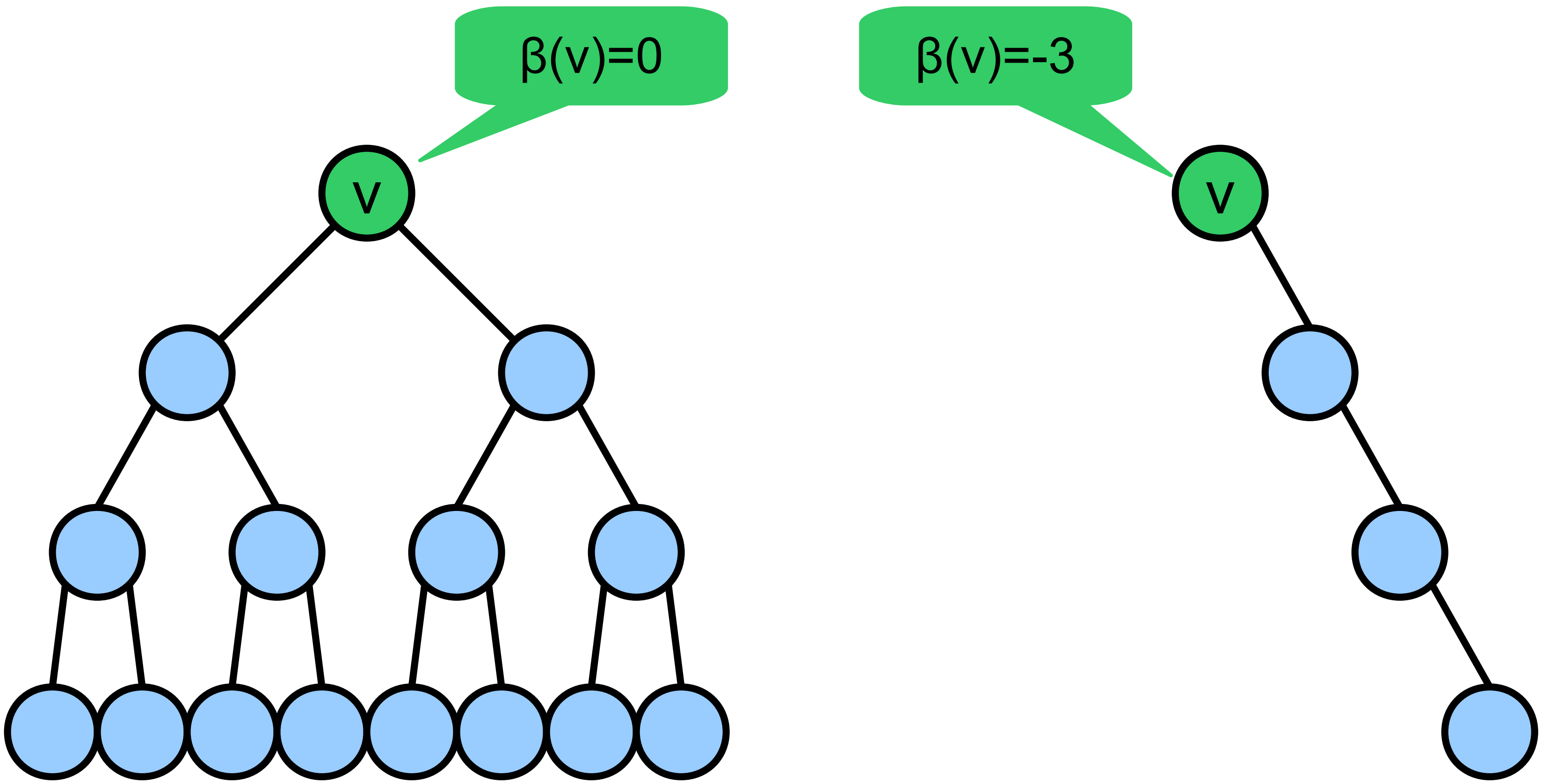
Alcune definizioni

- Fattore di bilanciamento
  - Il *fattore di bilanciamento* β(v) di un nodo v è dato dalla differenza tra l'altezza del sottoalbero sinistro e del sottoalbero destro di v:

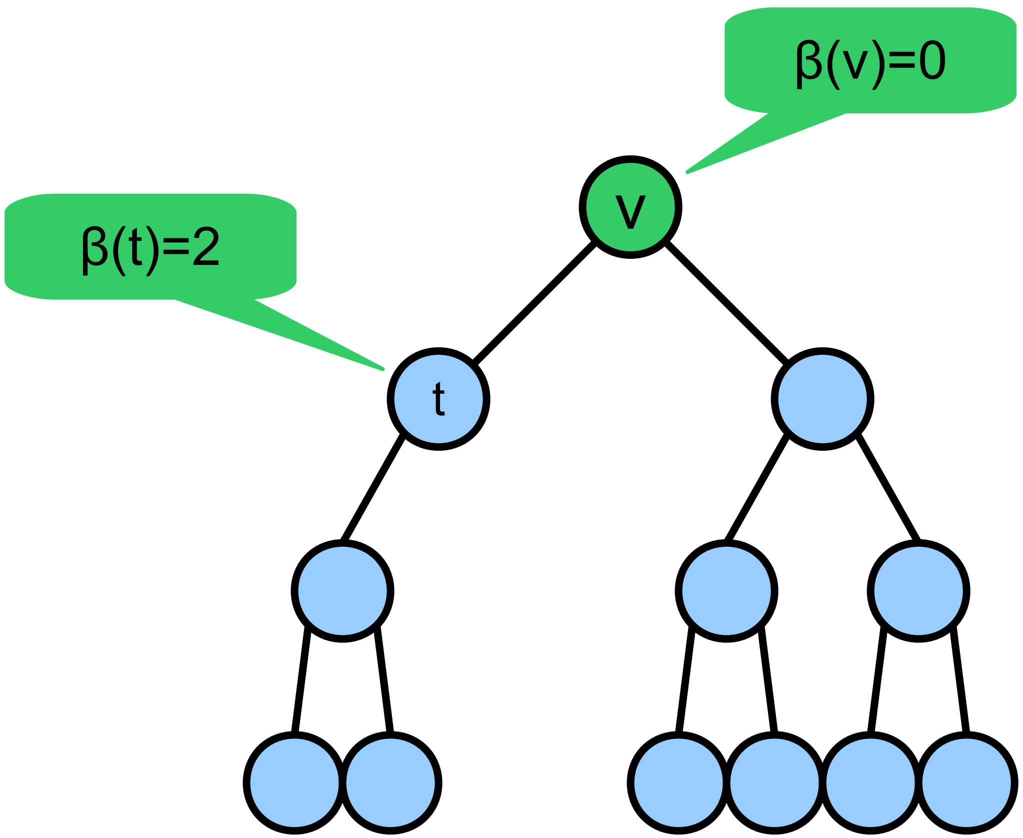
$$\beta(v) = \text{altezza}(\text{sin}(v)) - \text{altezza}(\text{des}(v))$$

- Bilanciamento in altezza
  - Un albero si dice bilanciato in altezza se le altezze dei sottoalberi sinistro e destro di ogni nodo differiscono al più di uno
  - In altre parole, un albero è bilanciato in altezza se per ogni suo nodo v, si ha $|\beta(v)| \leq 1$
- Definizione: un albero AVL è un ABR bilanciato in altezza

Esempio

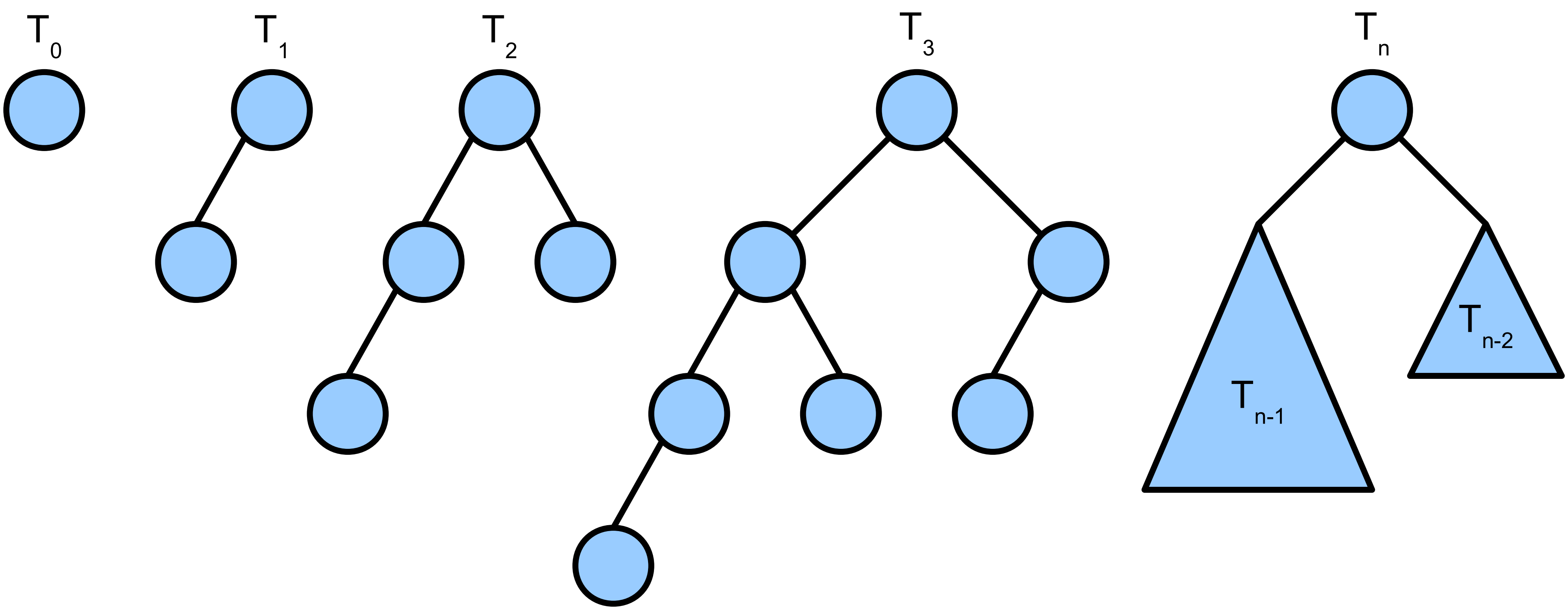


Esempio



Altezza di un albero AVL

- Per valutare l'altezza di un albero AVL, consideriamo gli alberi "più sbilanciati" che si possano costruire
- Alberi di Fibonacci



Altezza di un albero Fibonacci

- Consideriamo un albero di Fibonacci di altezza h. Sia n_h il numero dei nodi
- Per costruzione si ha

$$n_h = n_{h-1} + n_{h-2} + 1$$

- Dimostriamo che

$$n_h = F_{h+3} - 1$$

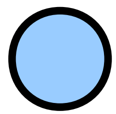
ove F_n è l'n-esimo numero di Fibonacci

Altezza di un albero Fibonacci

$$n_h = F_{h+3} - 1$$

- Base: h=0
  - $n_0 = 1$
  - $F_3 = 2$

T_0



- Passo induttivo

$$\begin{aligned} n_h &= n_{h-1} + n_{h-2} + 1 \\ &= (F_{h+2} - 1) + (F_{h+1} - 1) + 1 \\ &= F_{h+2} + F_{h+1} - 1 \\ &= F_{h+3} - 1 \end{aligned}$$

Altezza di un albero di Fibonacci

- Quindi ricapitolando: un albero di Fibonacci di altezza h ha $F_{h+3} - 1$ nodi
- Ricordiamo che

$$F_h = \Theta(\phi^h), \phi \approx 1.618$$

da cui otteniamo

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

e possiamo quindi concludere che

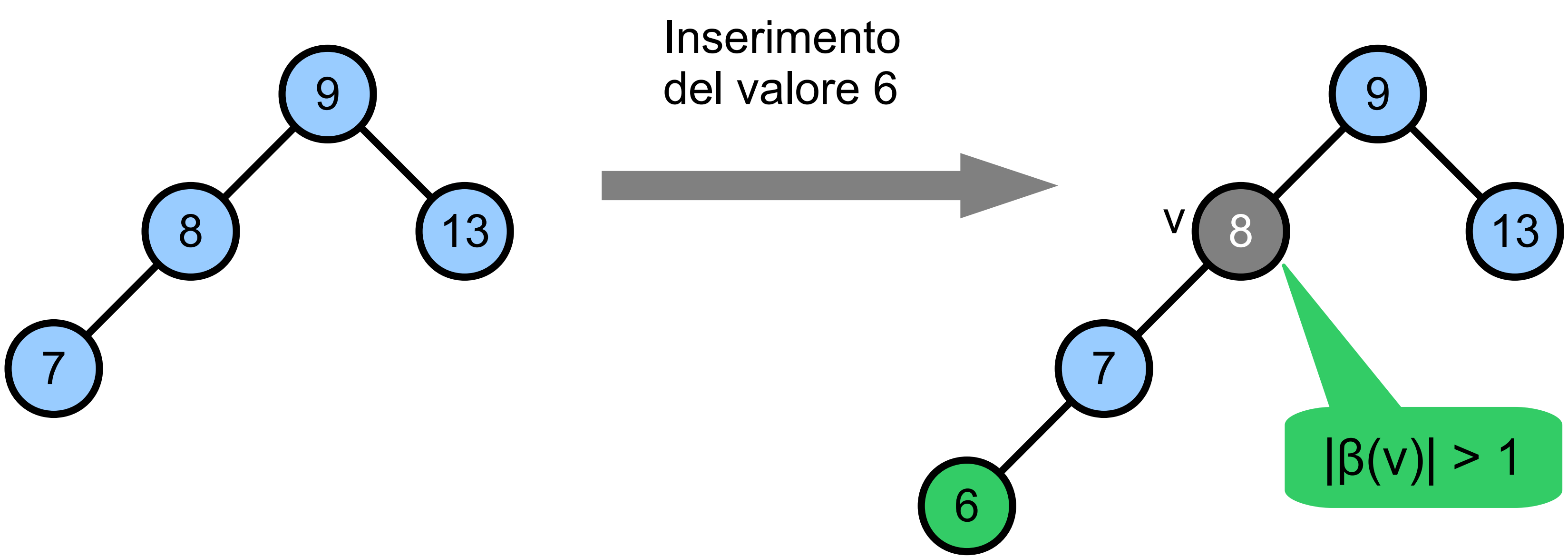
$$h = \Theta(\log n_h)$$

Conclusione

- Poiché...
  - l'albero di Fibonacci con n nodi è quello che tra tutti gli alberi AVL con n nodi ha altezza massima;
  - l'altezza di un albero di Fibonacci con n nodi è proporzionale a (log n)
- ...si conclude che:
  - l'altezza di un albero AVL con n nodi è O(log n)

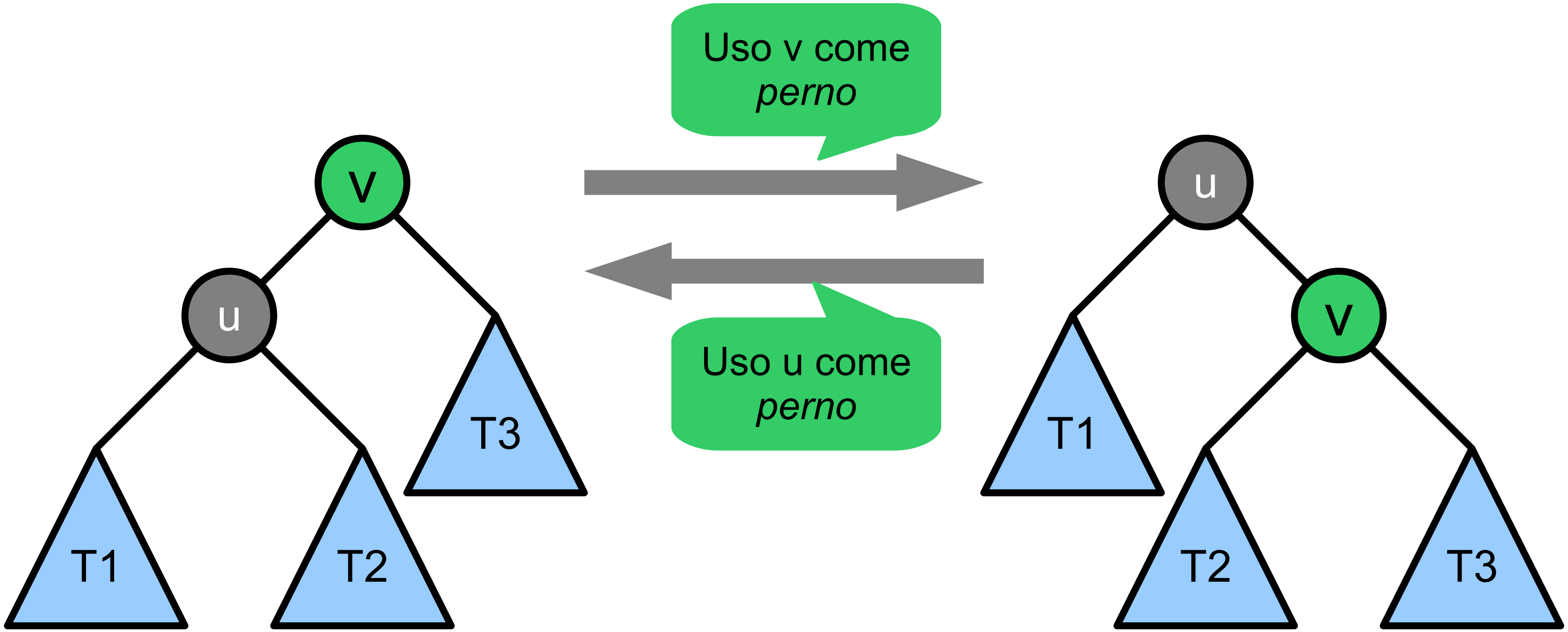
Mantenere il bilanciamento

- La ricerca in un albero AVL viene effettuata come in un generico ABR
- Inserimenti e rimozioni invece richiedono di essere modificati per mantenere il bilanciamento dell'albero
- Esempio



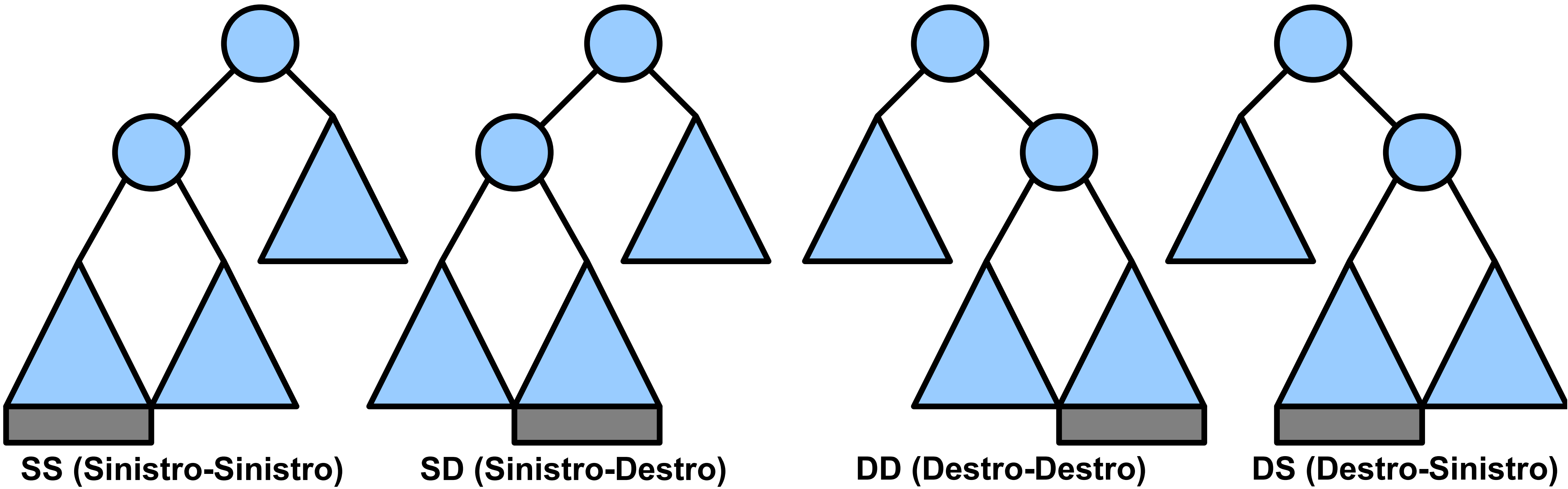
Rotazioni

- L'operazione fondamentale per ribilanciare l'albero è la rotazione semplice
  - Domanda: dimostrare che la rotazione semplice preserva la proprietà d'ordine degli ABR



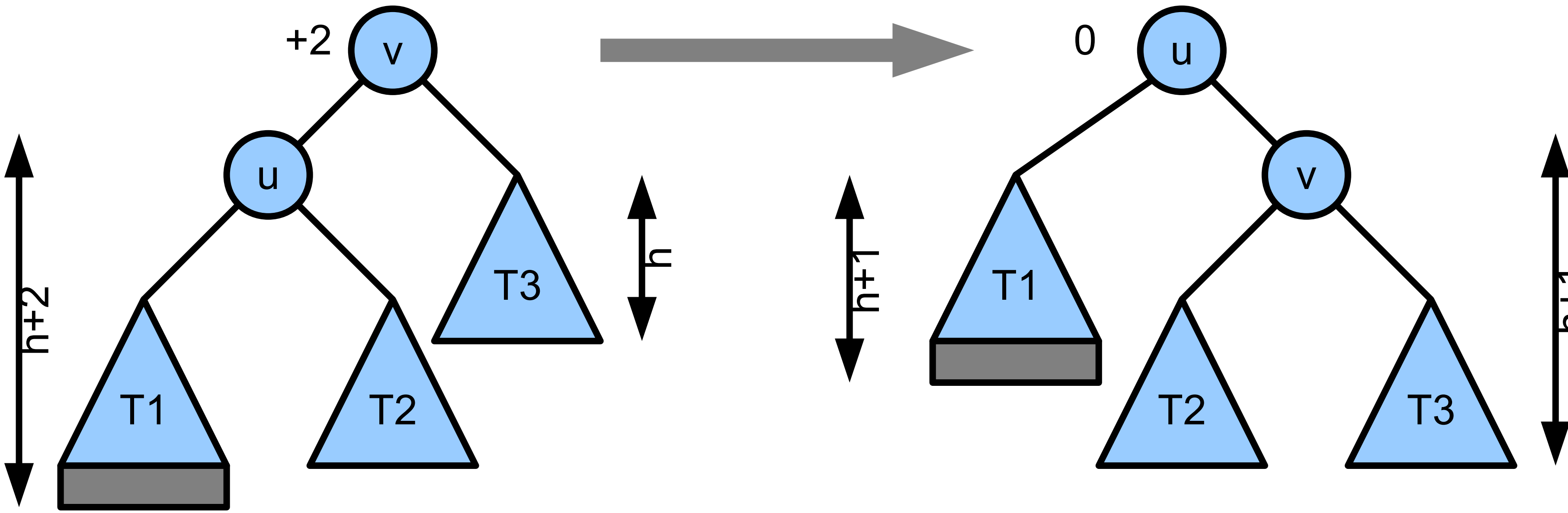
Rotazioni

- Supponiamo che a seguito di un inserimento o cancellazione, una parte dell'albero sia sbilanciata
- Abbiamo quattro casi (simmetrici due a due)

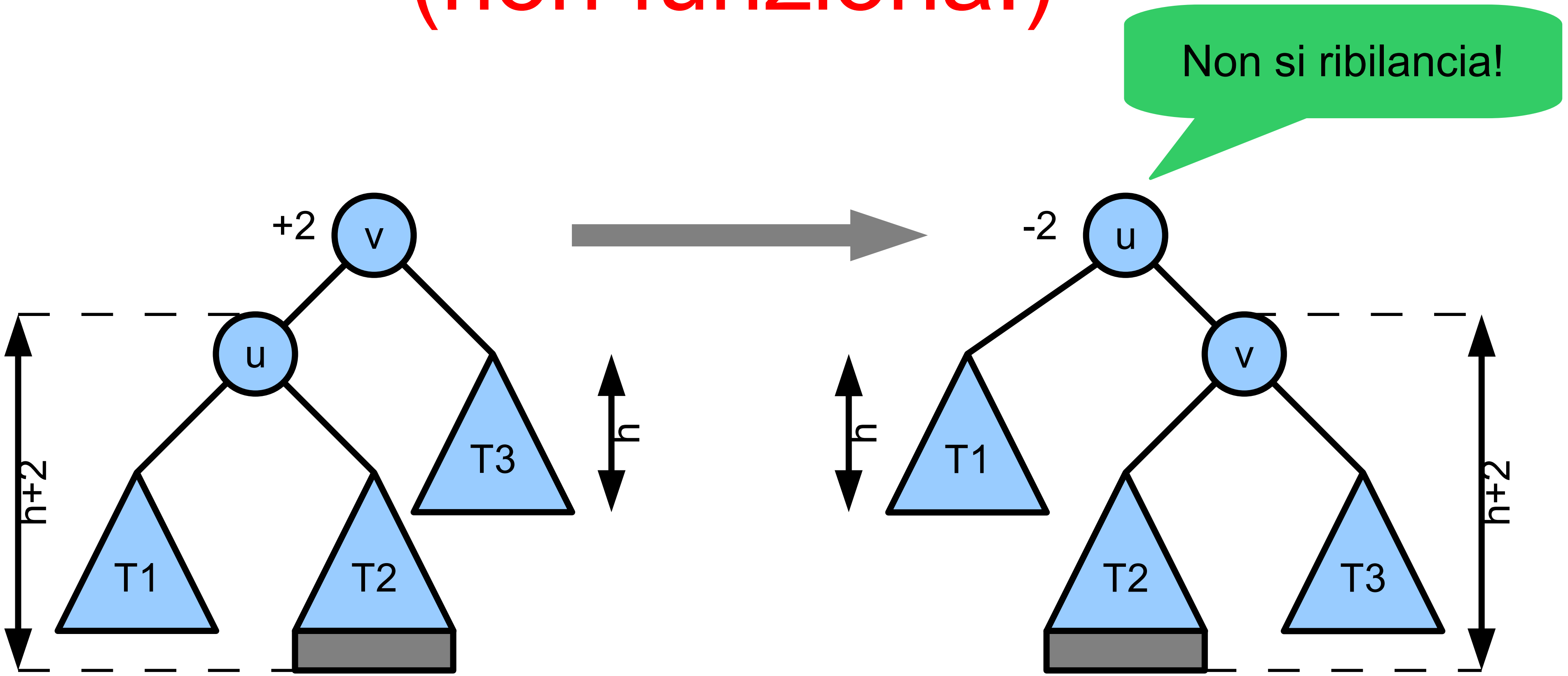


Ribilanciamento: rotazione SS

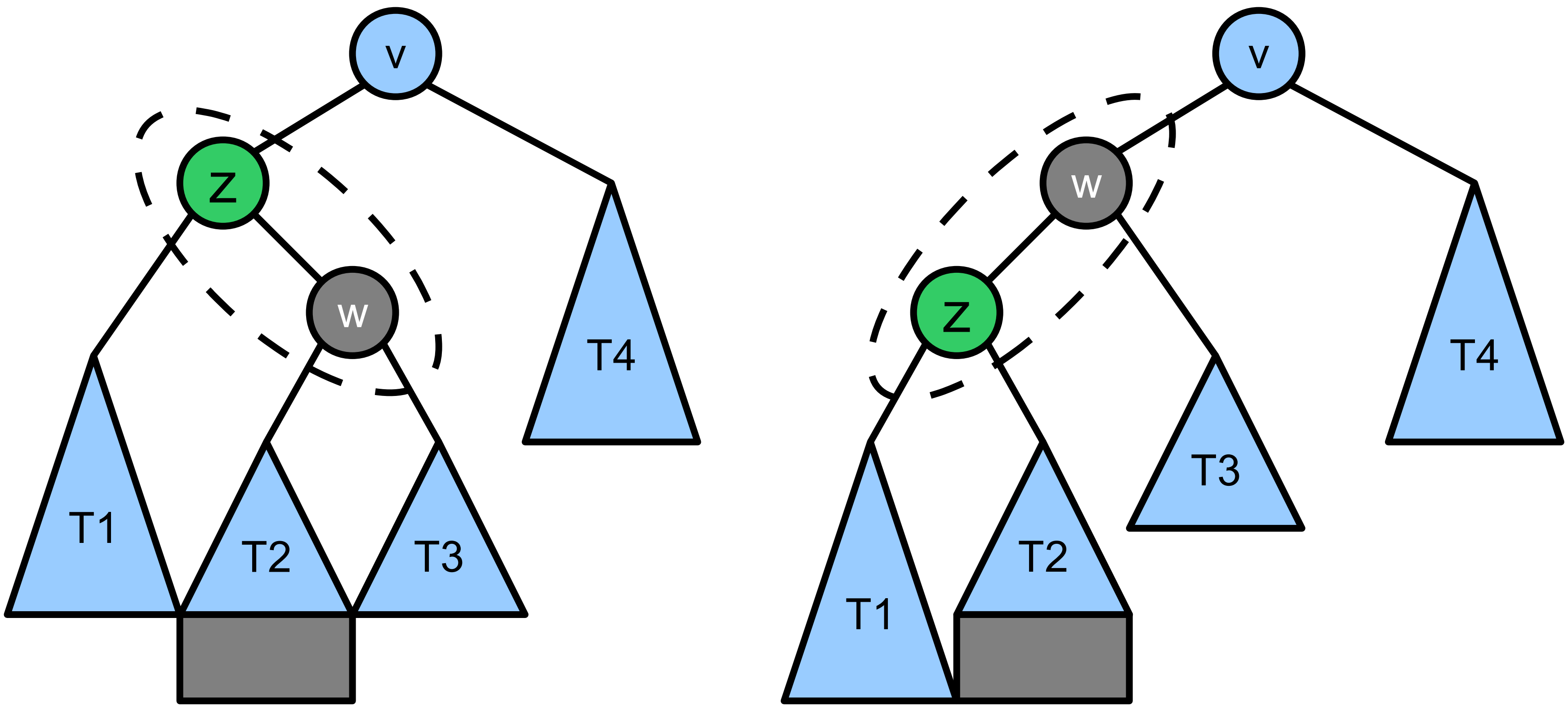
- Si applica una rotazione semplice verso destra su v
- Ha costo O(1)



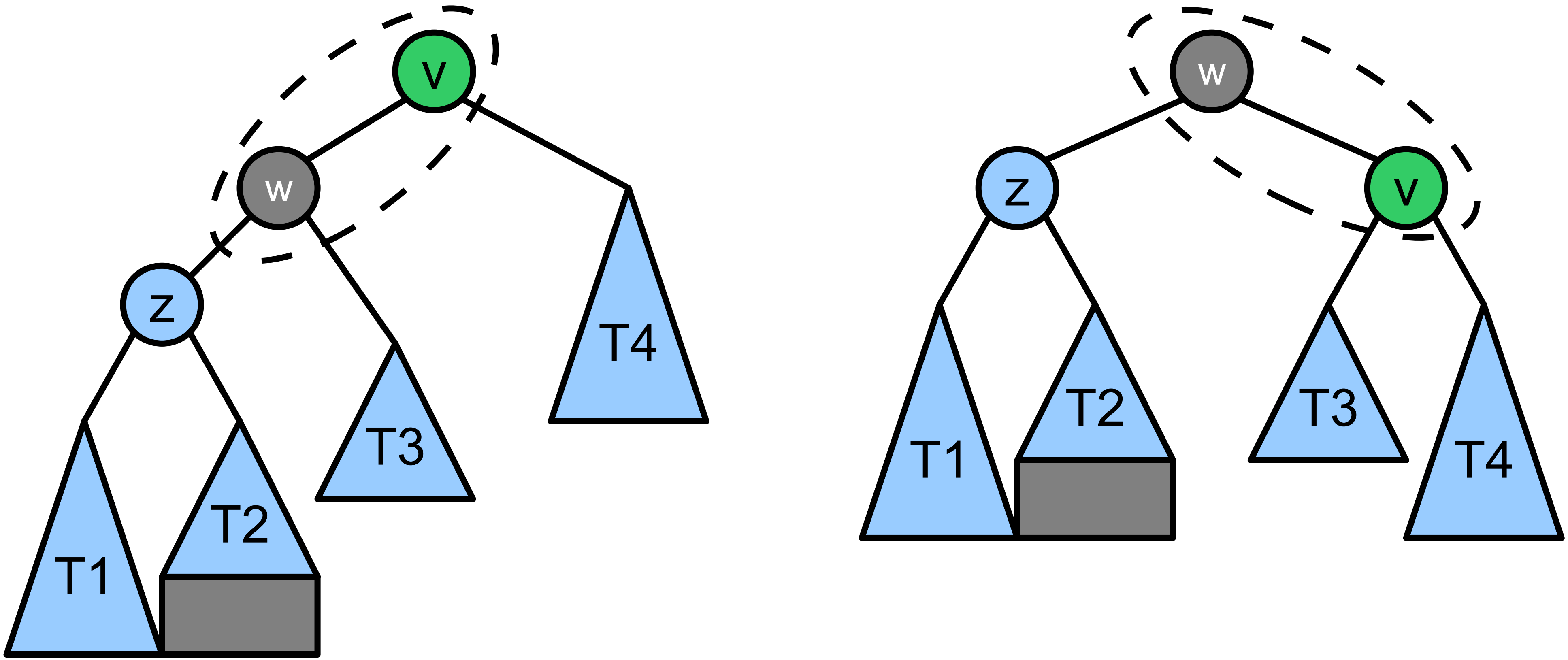
Ribilanciamento: rotazione SD (non funziona!)



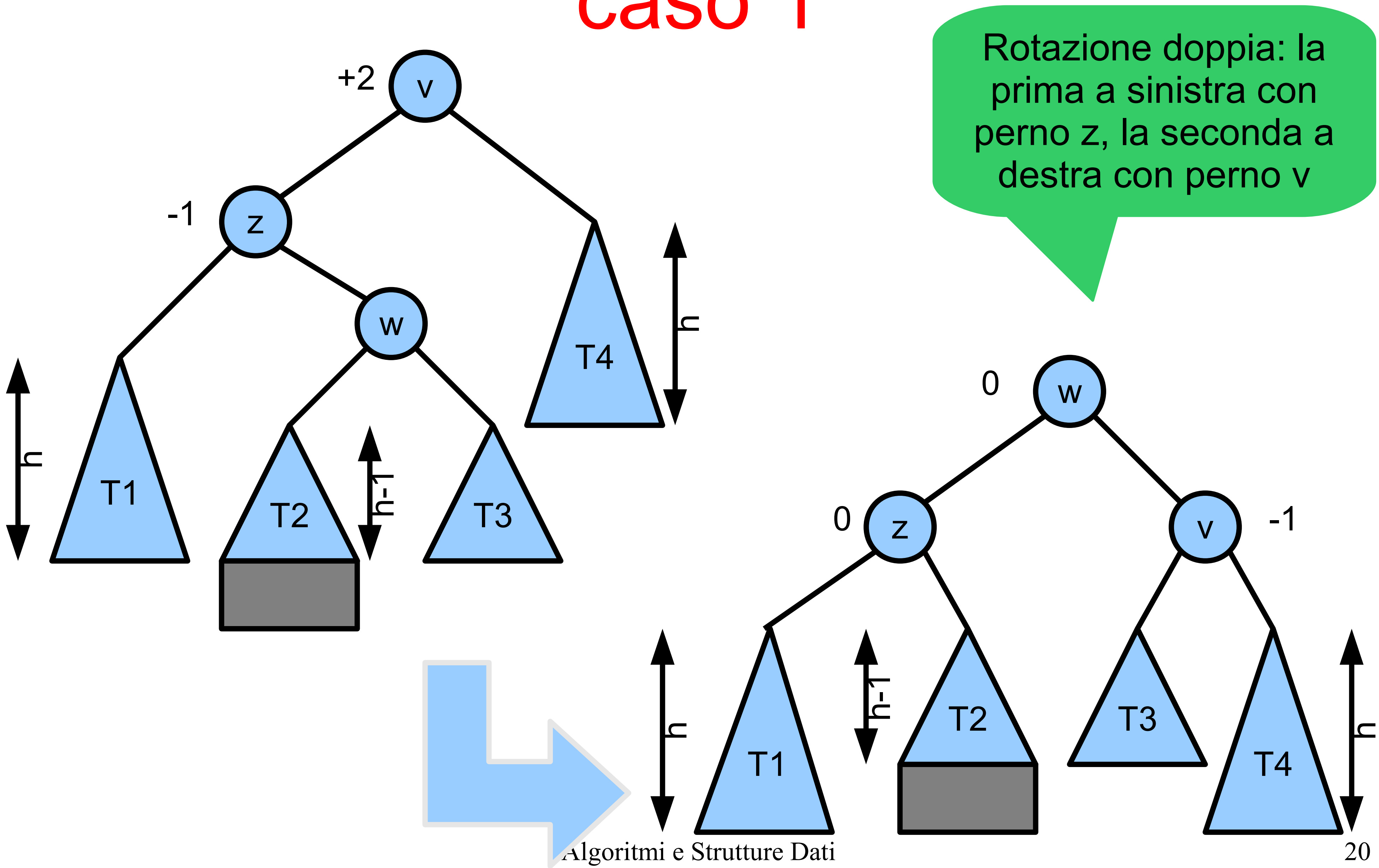
Ribilanciamento: rotazione SD primo passo



Ribilanciamento: rotazione SD secondo passo

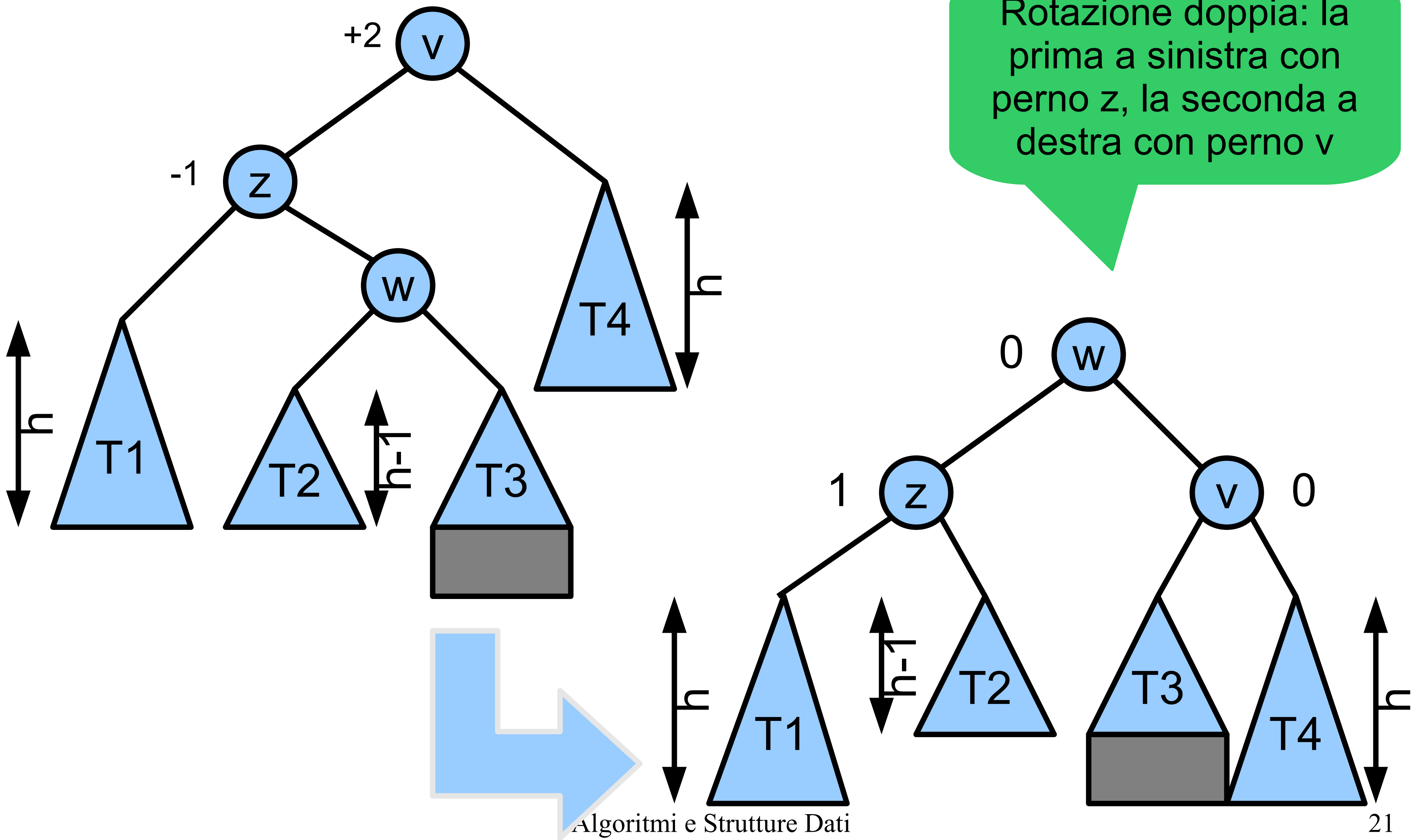


Ribilanciamento: rotazione SD caso 1



Ribilanciamento: rotazione SD caso 2

Rotazione doppia: la prima a sinistra con perno z, la seconda a destra con perno v



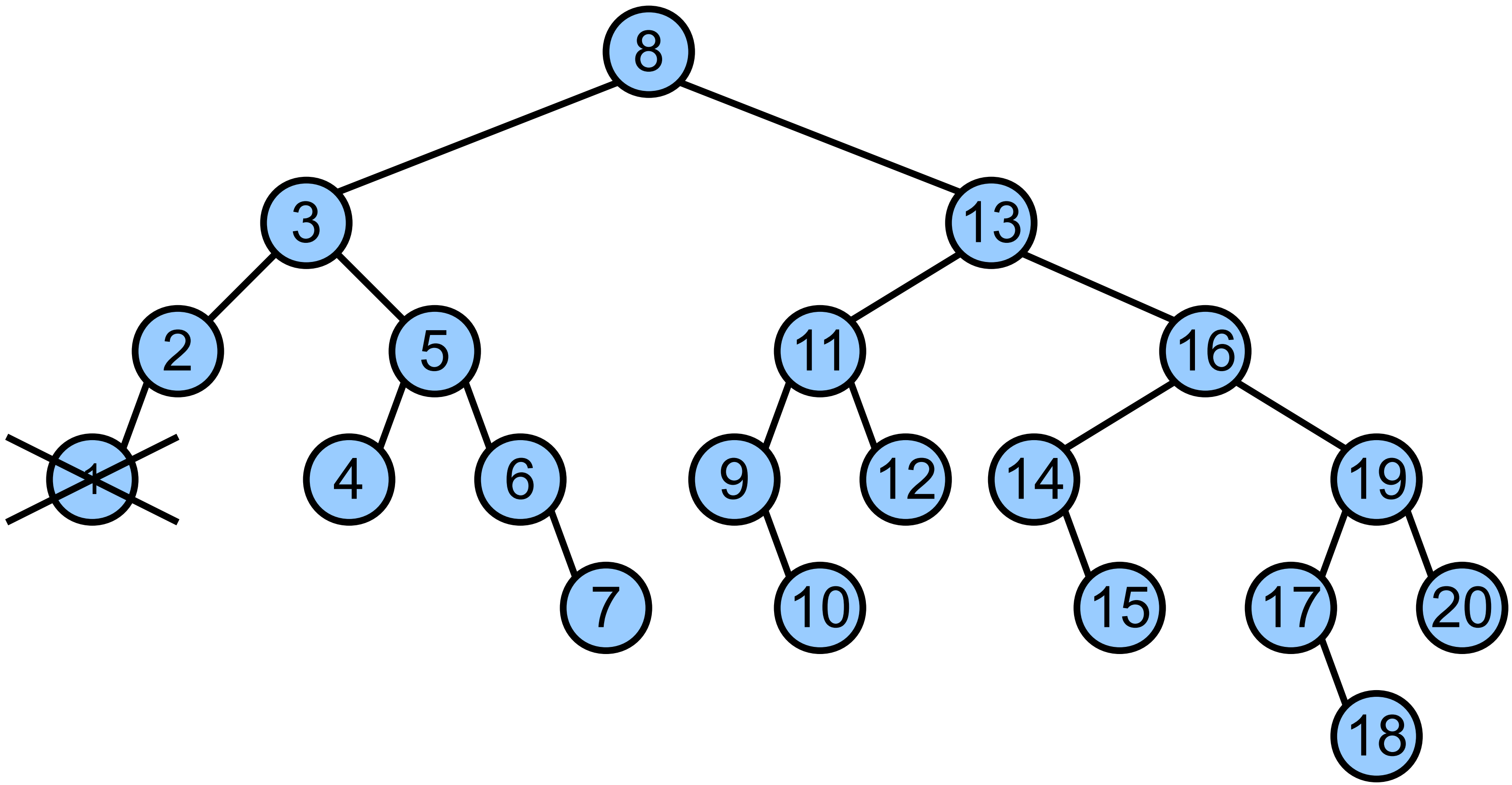
Alberi AVL: Inserimento

- Si inserisce il nuovo valore come per gli ABR
- Si ricalcolano tutti i fattori di bilanciamento mutati
  - Al più il ricalcolo riguarderà un cammino dalla foglia appena inserita fino alla radice, quindi ha costo O(log n)
- Se un nodo presenta fattore di bilanciamento ±2 (nodo critico), occorre ribilanciare l'albero mediante una delle rotazioni viste
  - Nota: in caso di inserimento, il nodo critico è unico
- Costo complessivo: O(log n)

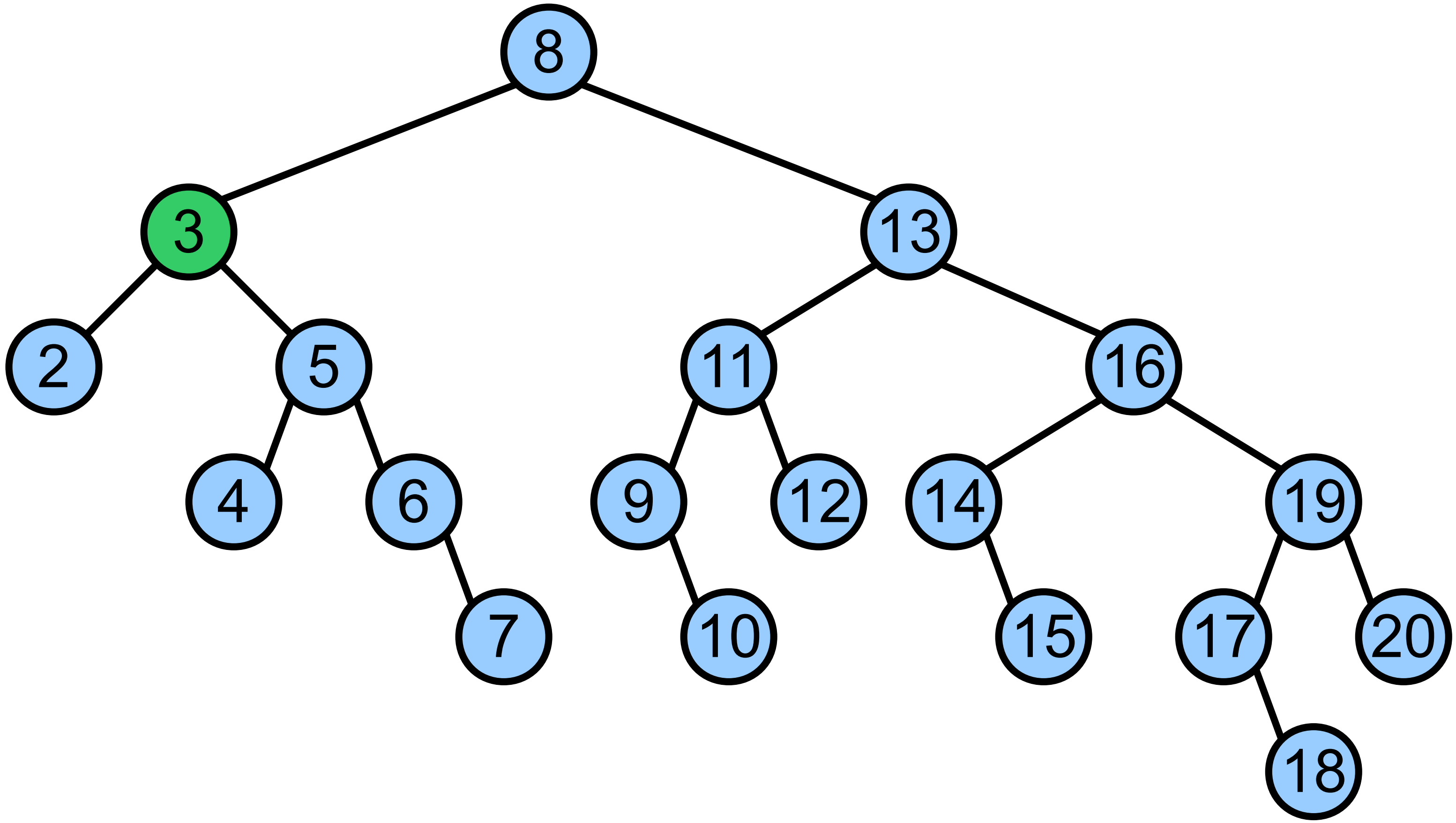
Alberi AVL: Rimozione

- Si rimuove il nodo come per gli ABR
- Si ricalcolano tutti i fattori di bilanciamento mutati
  - Al più il ricalcolo riguarderà un cammino dal padre del nodo eliminato fino alla radice, quindi ha costo O(log n)
- Per ogni nodo con fattore di bilanciamento ±2, occorre ribilanciare l'albero mediante una delle rotazioni viste
  - Nota: nel caso della rimozione, possono comparire più nodi con fattori di bilanciamento ±2
- Costo complessivo: O(log n)

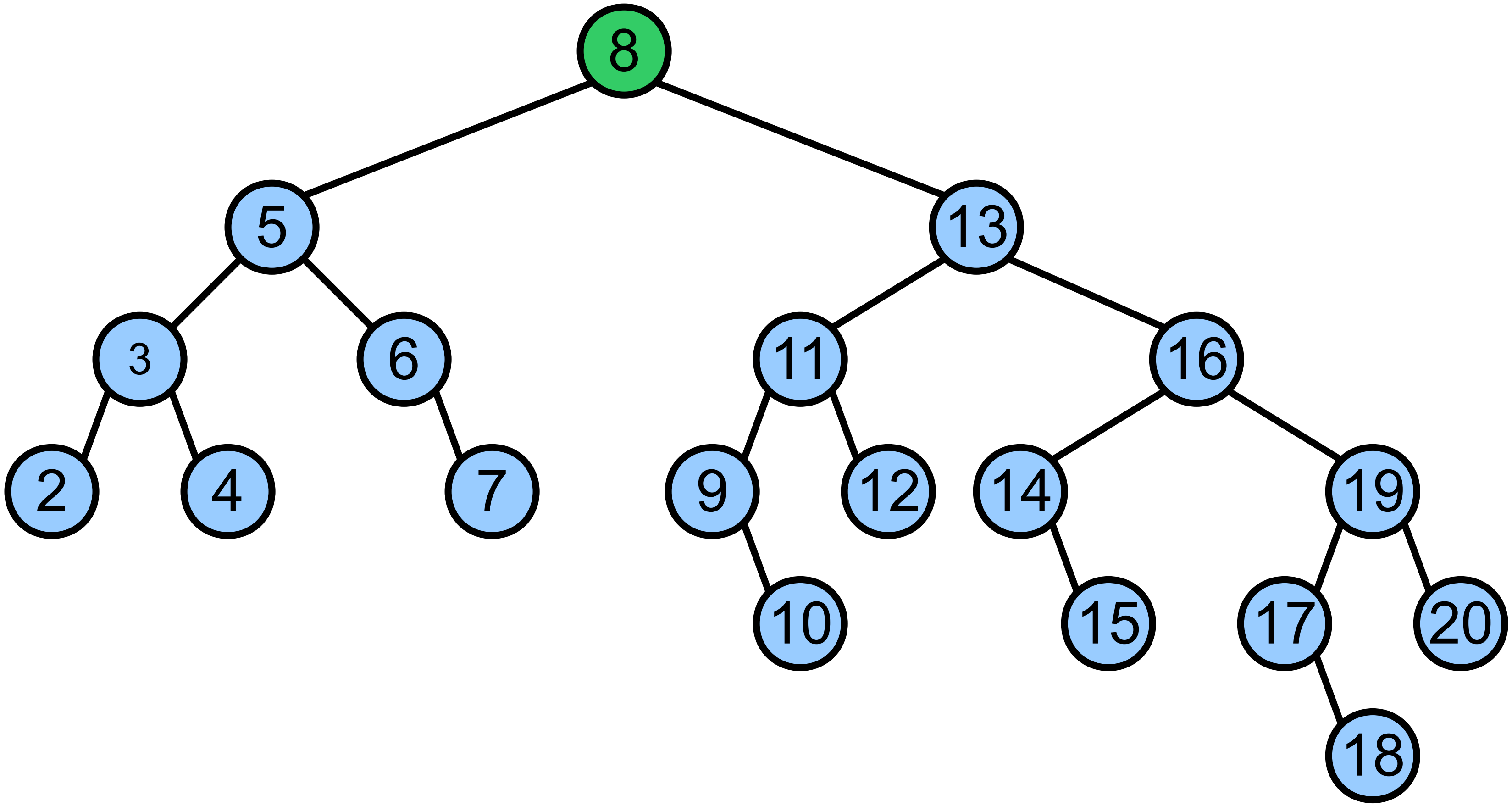
Esempio: cancellazione con rotazioni a cascata



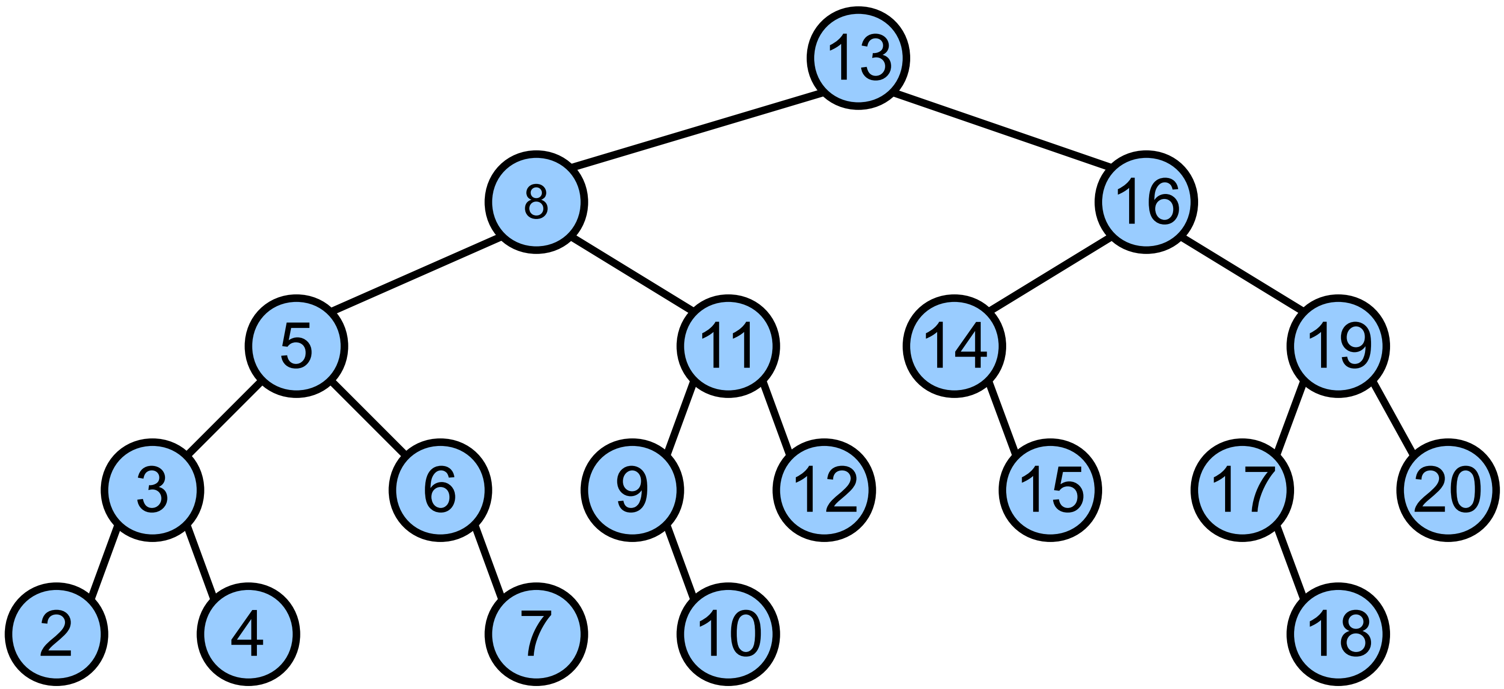
Applicare rotazione a sinistra su 3



Applicare rotazione a sinistra su 8



Albero ribilanciato



Alberi AVL: Riassunto

- search(Key k)
  - O(log n) nel caso peggiore
- insert(Key k, Item t)
  - O(log n) nel caso peggiore
- delete(Key k)
  - O(log n) nel caso peggiore