Analisi algoritmi ricorsivi

Esercizio1(A,i,j)
input: A è un vettore di interi e i e j sono interi non negativi.

```
1. k ← i
2. while (k ≤ j) do
3.         h ← i
4.     while (h ≤ j) do
5.                 A[h] ← A[h] +1
6.                 h ← h +1
7.         k ← k +1
```

```
Esercizio2 (A,i, j)
if ( i<j )
   then numElem ← j - i+1
Esercizio2( A, i, i + numElem/4)
Esercizio2( A, j - numElem/4,j )
Esercizio1( A, i, j )
```

Posto j-i+1 = n

ABR di altezza logaritmica

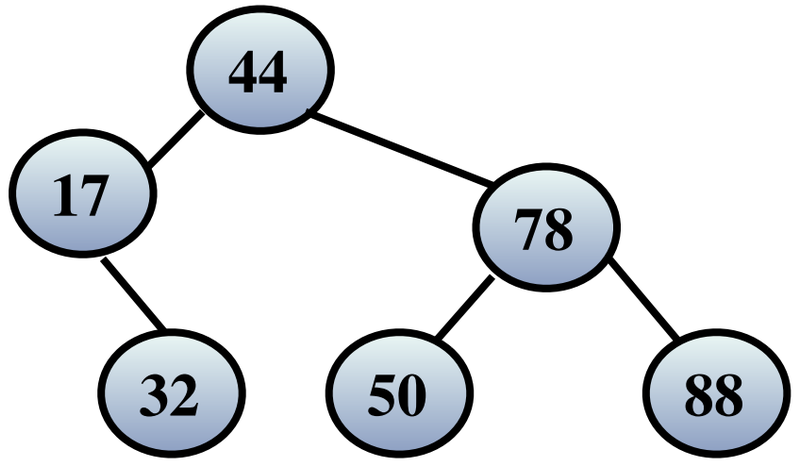
Esistono vari criteri di bilanciamento di alberi binari di ricerca che ne garantiscono l'altezza logaritmica.

Chiamiamo bilanciato ogni albero binario di altezza logaritmica nel numero dei nodi.

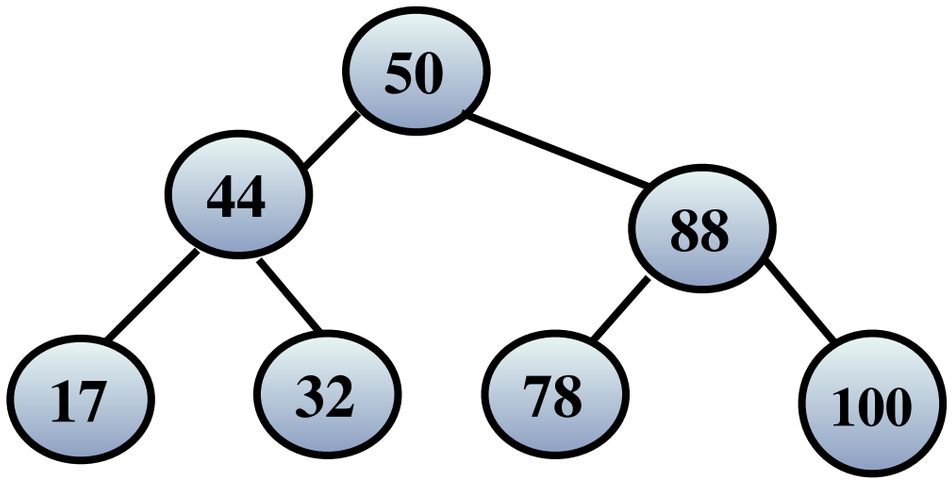
In seguito gli alberi spesso saranno disegnati mettendo in evidenza i nodi di interesse e disegnando come triangoli i sotto alberi di cui non sappiamo la dimensione e che potrebbero essere anche vuoti. Anche se i triangoli rappresentanti sotto alberi diversi dovessero risultare uguali, questo non vuol dire che i sotto alberi rappresentati lo sono.

ABR di altezza logaritmica

Potremmo pensare di utilizzare alberi completi fino al penultimo livello:

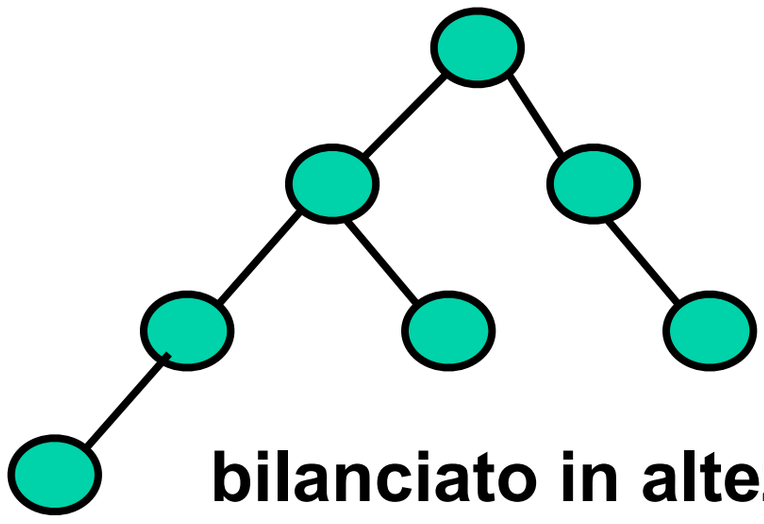


Ma se volessimo inserire 100, per mantenere la proprietà dovremmo ristrutturare tutto l'albero!



ABR bilanciati in altezza

Un ABR si dice bilanciato in altezza se per ogni nodo v la differenza in valore assoluto tra l'altezza del sottoalbero sinistro di v e l' altezza del sottoalbero destro di v è ≤ 1.



bilanciato in altezza

Gli ABR bilanciati in altezza si chiamano brevemente alberi AVL, sigla dalle iniziali degli autori Georgy Maximovich Adel'son-Vel'skii e Yevgeniy Mikhailovich Landis che li hanno introdotti nel 1962.

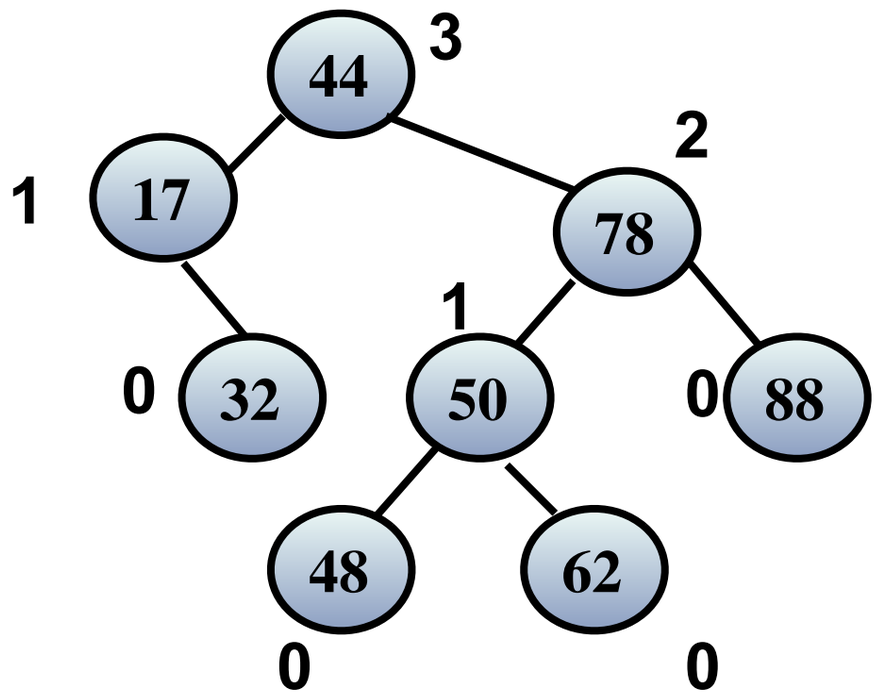
N.B. Prendiamo come altezza il massimo numero di archi in un cammino radice-foglia.

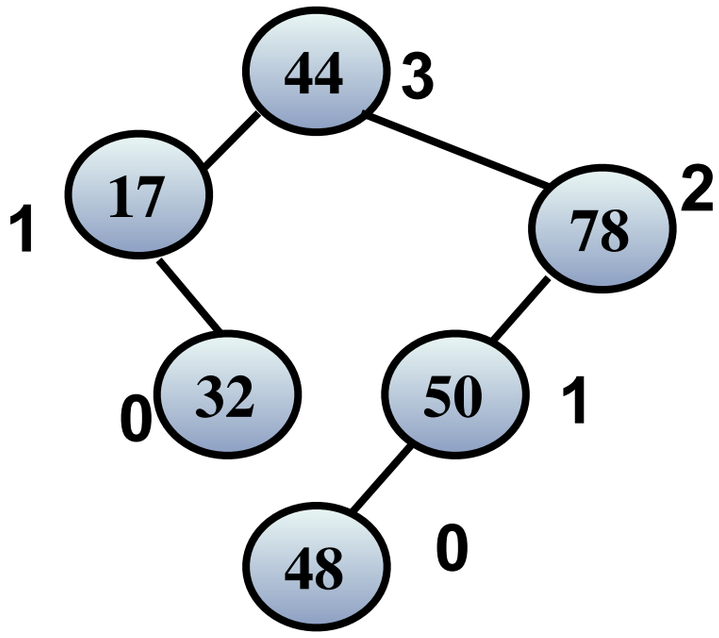
Quindi l'albero vuoto ha altezza -1.

Alberi AVL: esempi

bilanciamento in altezza:

per ogni nodo la differenza in valore assoluto tra l'altezza del sottoalbero sinistro di v e quella del sottoalbero destro di v è ≤ 1



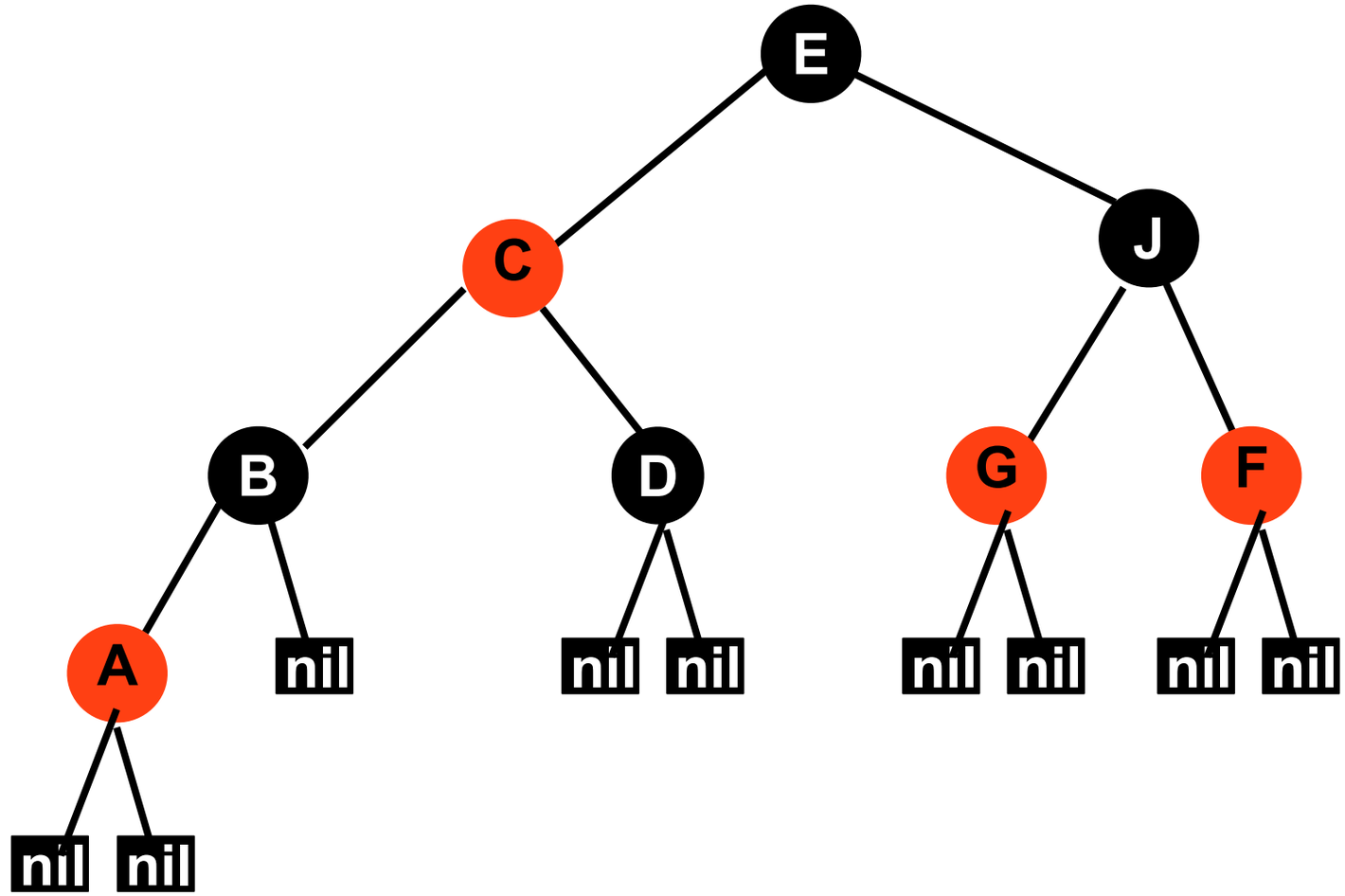


NON bilanciato: il nodo 78 ha un sotto albero destro di altezza -1, mentre il sotto albero sinistro ha altezza 1, quindi la differenza è 2.

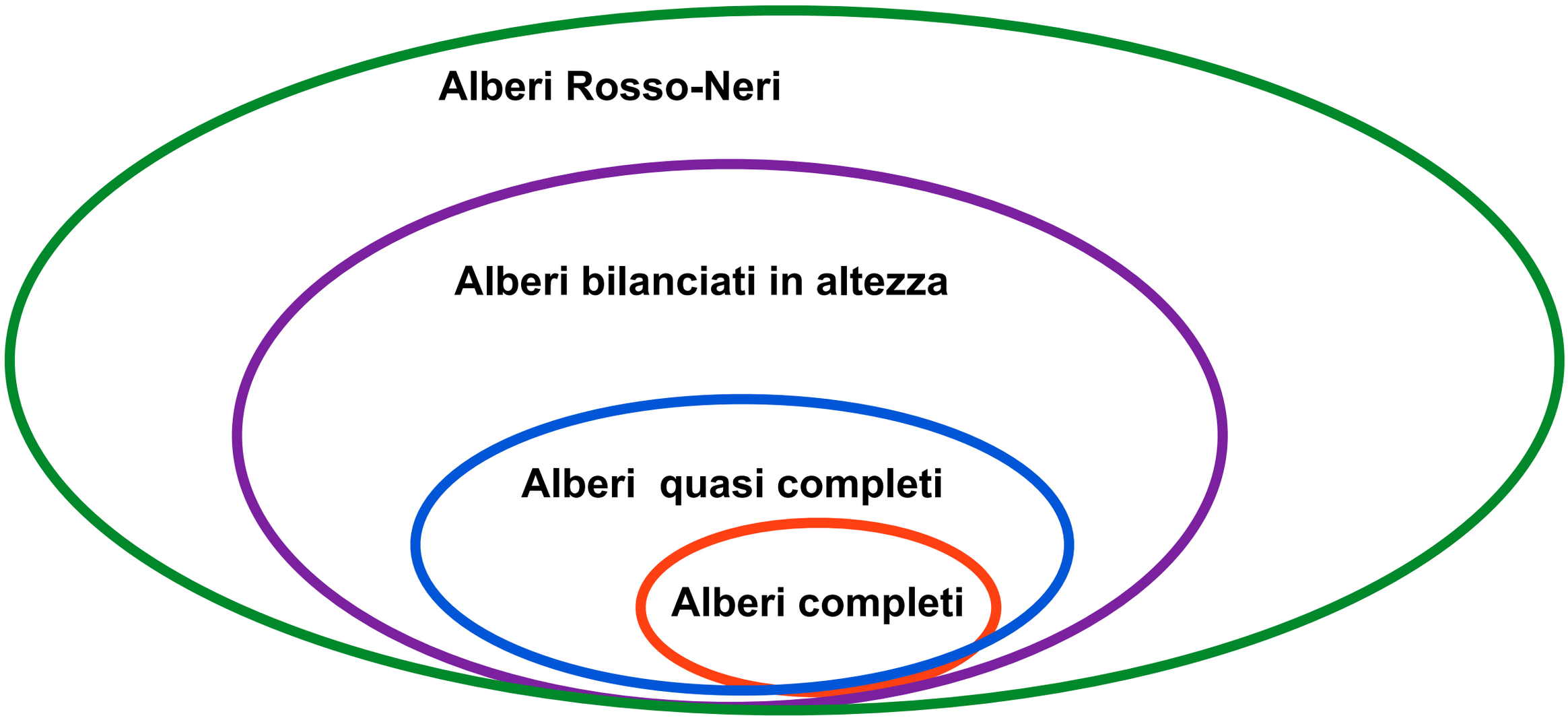
ABR Rosso Neri

- **alberi rosso-neri, introdotti nel 1972 da Rudolf Bayer, professore alla Technische Universität München, con il nome di "B-alberi simmetrici". Leonidas J. Guibas, Professore alla Stanford University e Robert Sedgewick, professore alla Princeton University ne provano molte proprietà, chiamandoli alberi rosso-neri, in un successivo articolo, apparso nel 1978.**

Ogni nodo in un albero rosso-nero è colorato di rosso o di nero, ma la colorazione deve soddisfare dei vincoli che garantiscono che nessun percorso radice-foglia è lungo più del doppio di ogni altro, così l'albero è abbastanza bilanciato da avere altezza logaritmica nel numero dei nodi.



Relazioni tra classi di alberi



Le operazioni

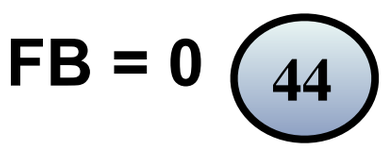
Perchè le operazioni di ricerca, inserimento e cancellazione in un ABR siano di complessità O(lg n), dove n è il numero dei nodi dell'albero bisogna modificarle in modo che le eventuali operazioni di ribilanciamento dell'albero siano anch'esse di complessità O(lg n).

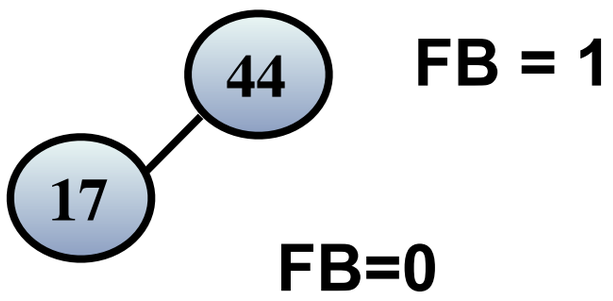
Gli alberi bilanciati in altezza o alberi AVL (Adel'son-Vel'skii, Landis, 1962), sono i primi per i quali si ottiene questo risultato poi ottenuto anche per gli alberi rosso-neri (Bayer,1972) e per gli Splay trees (Sleator,Tarjan 1985), che hanno la proprietà che i nodi ricercati sono spostati verso la radice per poterli poi trovare più efficientemente in seguito.

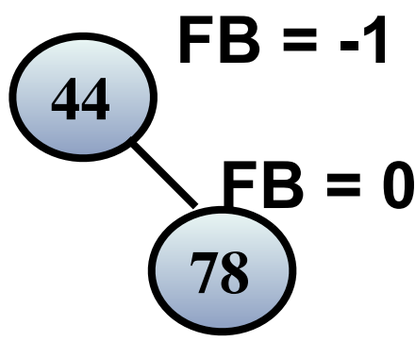
Esistono vari altri alberi di ricerca (anche non binari) con i quali possiamo implementare un dizionario di n elementi con operazioni in tempo O(lg n).

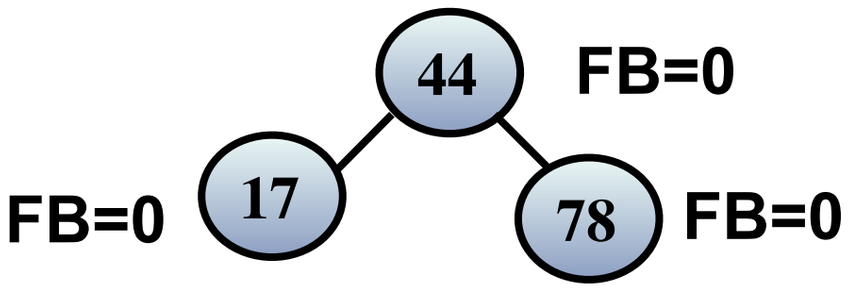
Costruzione AVL

Detto FB il fattore di bilanciamento di un nodo, cioè la differenza tra l' altezza del sotto albero sinistro e quella del destro, vediamo alcuni esempi di AVL, cominciando dai più piccoli in termini di numero dei nodi.

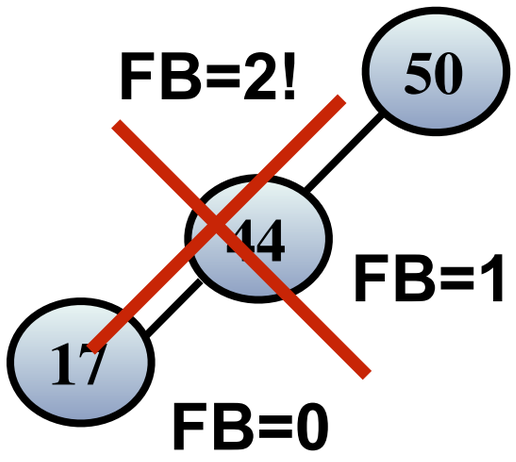




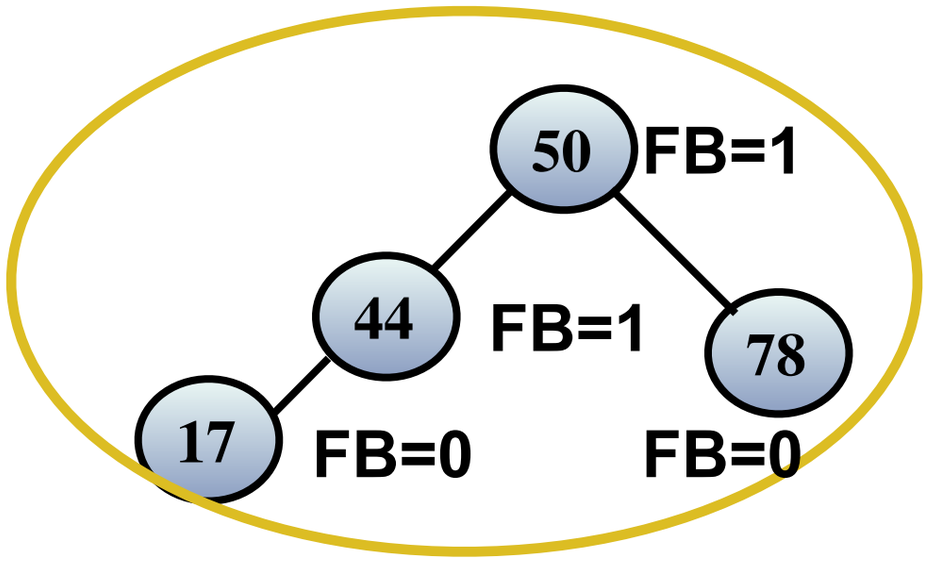


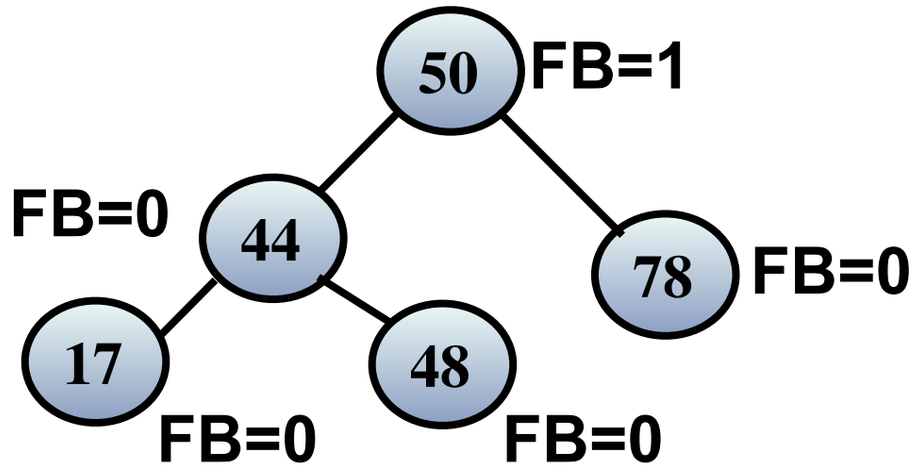


L'unico AVL con tre nodi!

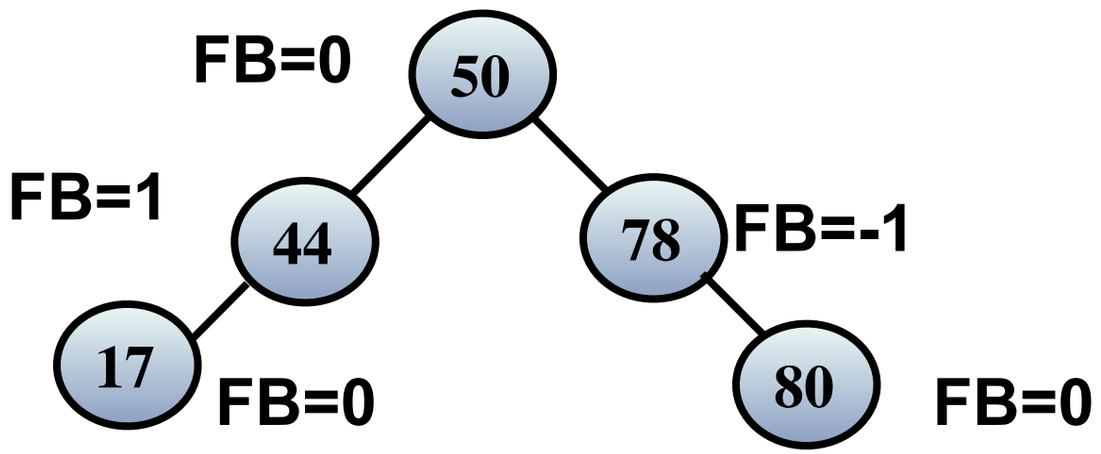


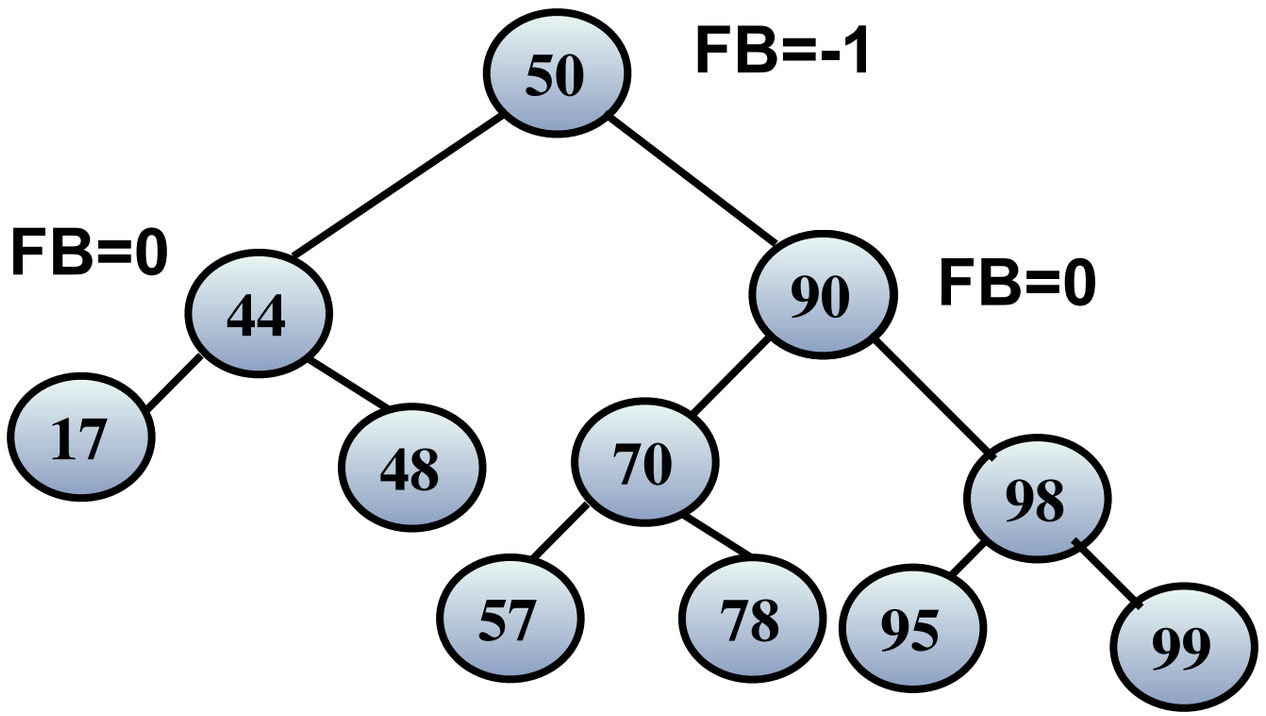
Esempi di AVL





tra quelli di altezza 2 ha il minor numero di nodi

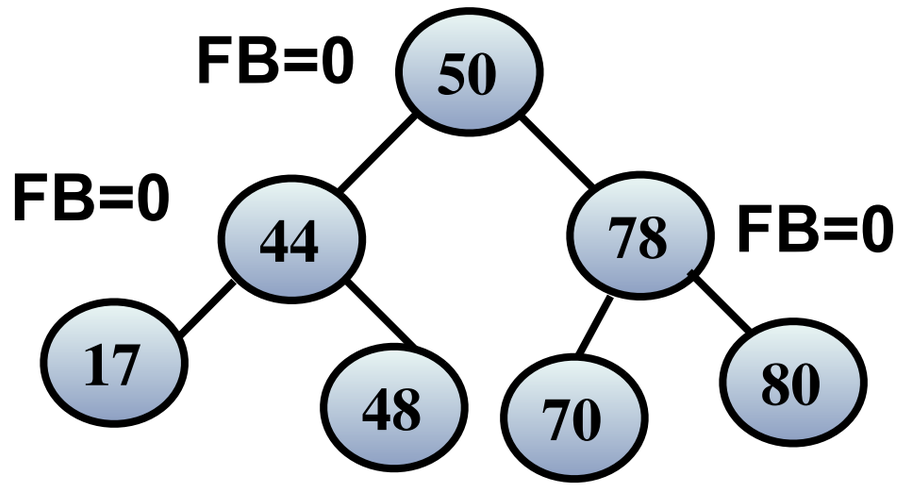




Massimo nodi, minima altezza

L'albero completo è un AVL di altezza logaritmica, e tutti i suoi nodi interni hanno FB = 0

(le foglie avendo sempre FB =0).



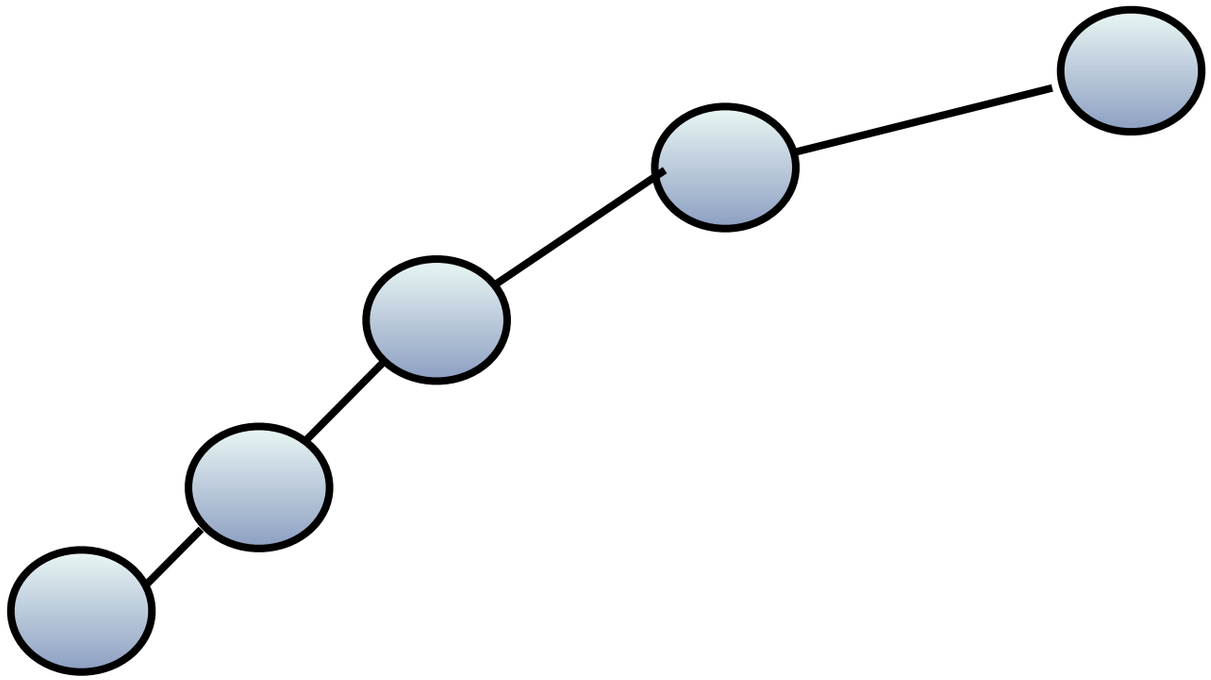
É l'albero AVL che a parità di altezza ha il massimo numero di nodi.

Quindi se un albero qualunque ha altezza h, il suo numero di nodi n è minore o uguale a 2^{h+1}-1, che è il numero di nodi di un albero completo di altezza h. Da qui abbiamo dedotto il limite inferiore logaritmico sull'altezza di un qualsiasi albero.

Ora vogliamo dimostrare che il <u>limite superiore all'altezza di un AVL</u> di n nodi è lg n.

Massima altezza, minimo nodi

<u>Ricordiamo</u> che un albero degenere è il più sbilanciato tra tutti gli alberi binari ed è quello di altezza massima a parità di numero di nodi, o anche è quello che <u>a parità di altezza ha il minimo numero di nodi</u> (un solo nodo su ogni livello).



Questo tipo di albero ci ha fornito il limite superiore all'altezza di un albero qualunque, infatti ogni albero con n nodi avrà altezza minore o uguale n-1, che è quella dell'albero degenere.

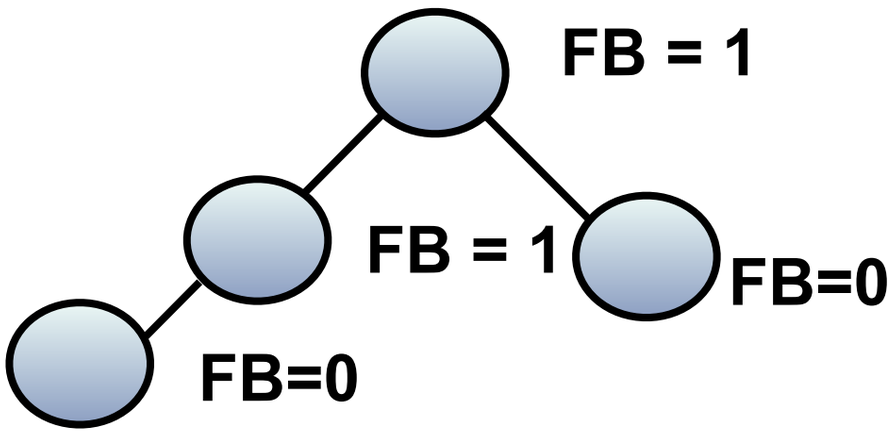
Massima altezza, minimo nodi per AVL

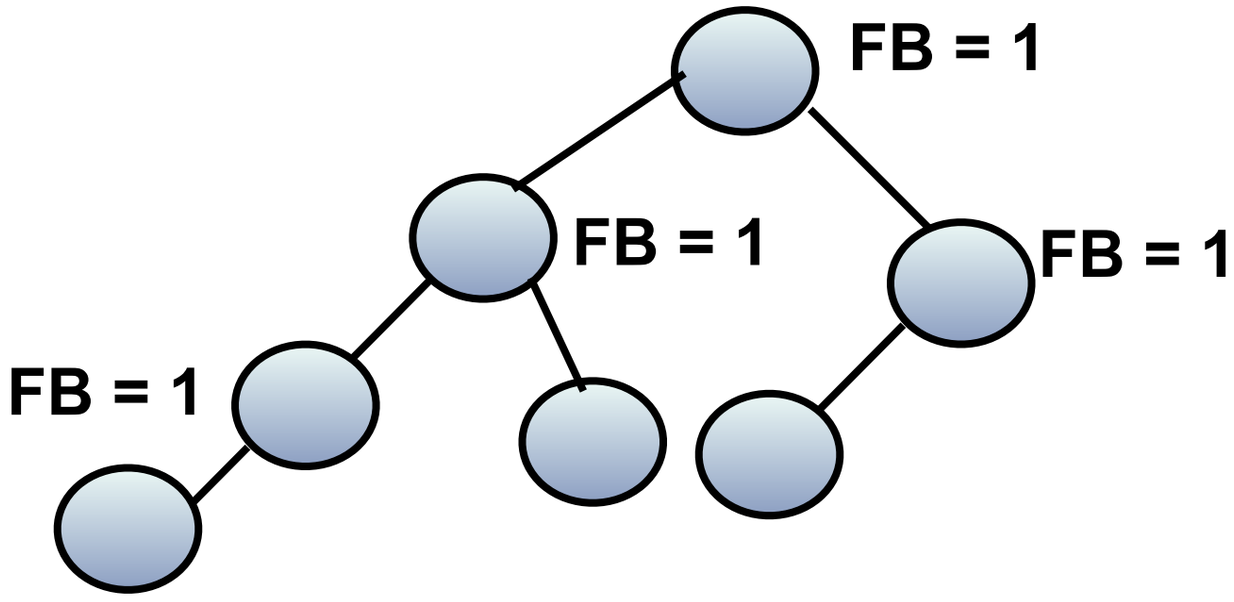
Cerchiamo di determinare gli AVL più sbilanciati che possiamo costruire, cioè cerchiamo gli AVL che a parità di altezza hanno il minimo numero di nodi.

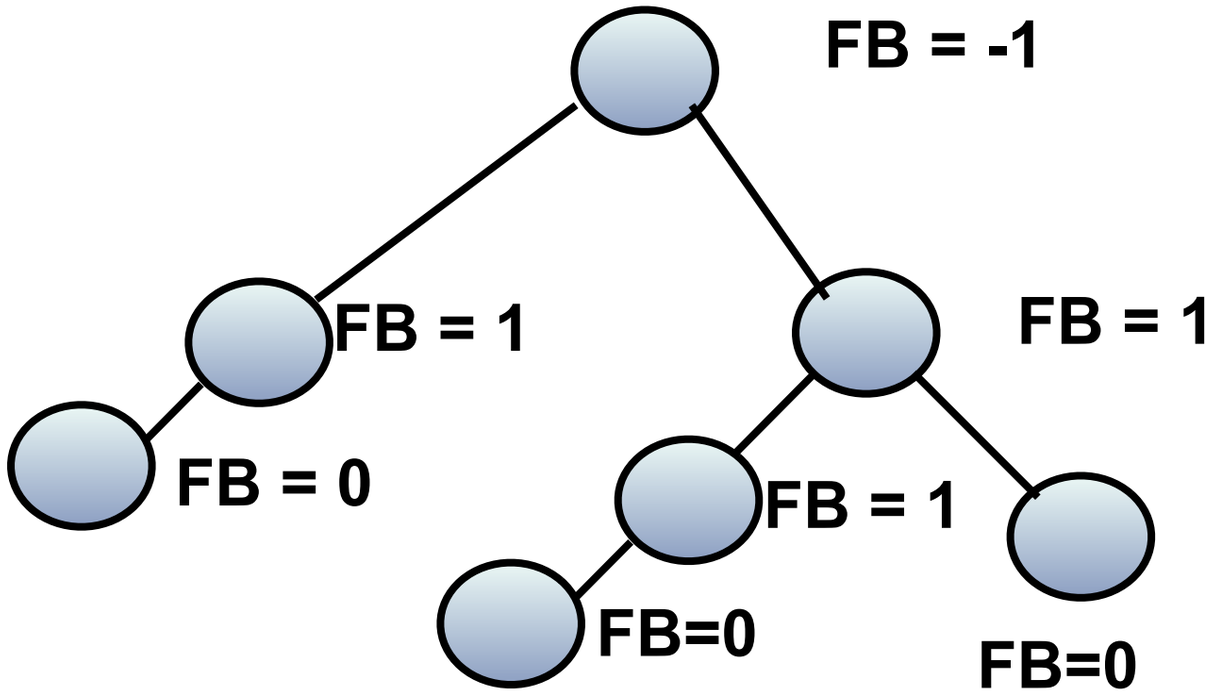
Il massimo sbilanciamento si ottiene imponendo a tutti i nodi non foglia di avere un FB diverso da 0.

Per semplicità stabiliamo che debba essere 1, se prendessimo sempre -1 si avrebbe un risultato equivalente.

Alberi AVL il più “sbilanciati” possibile

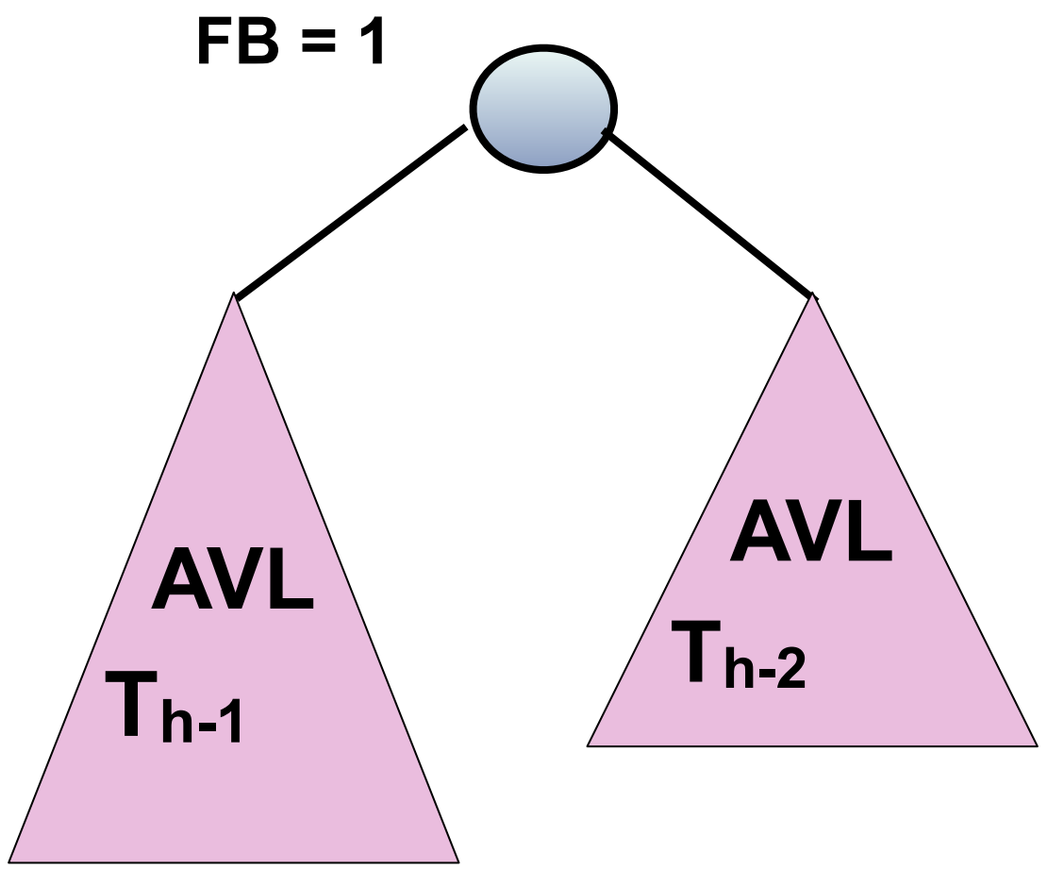






Una scelta “mista” di FB pari a 1 o a -1 è solo più difficile da gestire, volendo determinare il rapporto tra altezza e numero dei nodi.

Alberi AVL più sbilanciati



Definizione ricorsiva:

Base: l'albero vuoto è un AVL più sbilanciato a sinistra di altezza -1, T_{-1}, l'albero con un solo nodo è un AVL più sbilanciato a sinistra di altezza 0, T_0,

Passo induttivo: dati due AVL più sbilanciati a sinistra, di altezza h-1 e h-2 rispettivamente, T_{h-1} e T_{h-2} , un nuovo AVL di altezza h si costruisce prendendo un nuovo nodo come radice e come sotto albero sinistro T_{h-1} e come sotto albero destro T_{h-2}.

Per costruzione tutti i nodi interni di un AVL più sbilanciato a sinistra hanno FB = 1

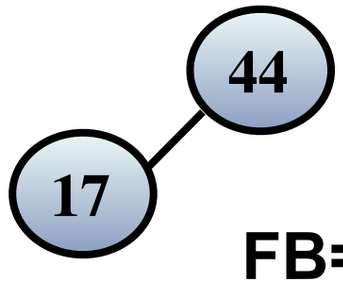
Passi iniziali della costruzione

Il più piccolo albero AVL più sbilanciato a sinistra non vuoto è

l'albero T_0 con h = 0 e n_1 = 1

FB = 0

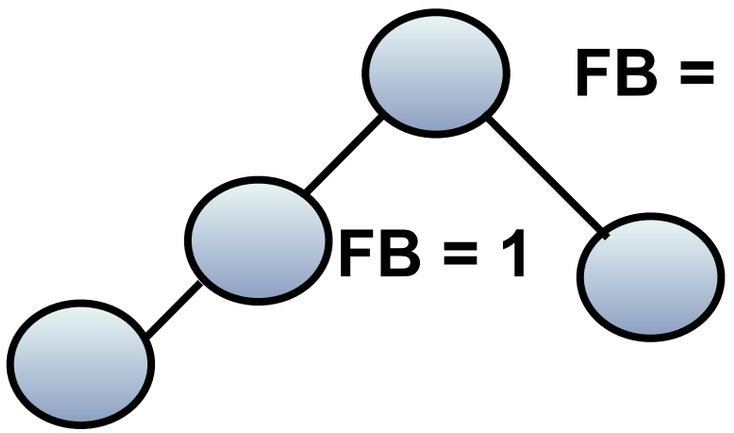
l'albero T_1 con h = 1 e n_1 = 2



FB = 1

FB=0

l'albero T_2 costruito a partire da T_0 e T_1



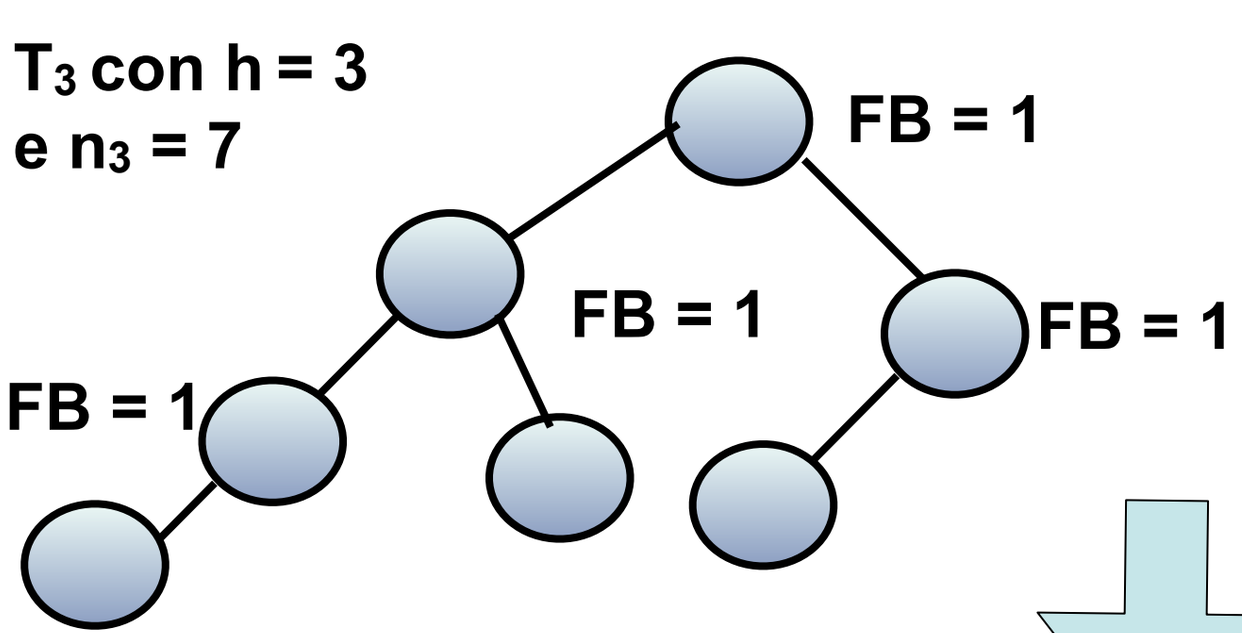
FB = 1

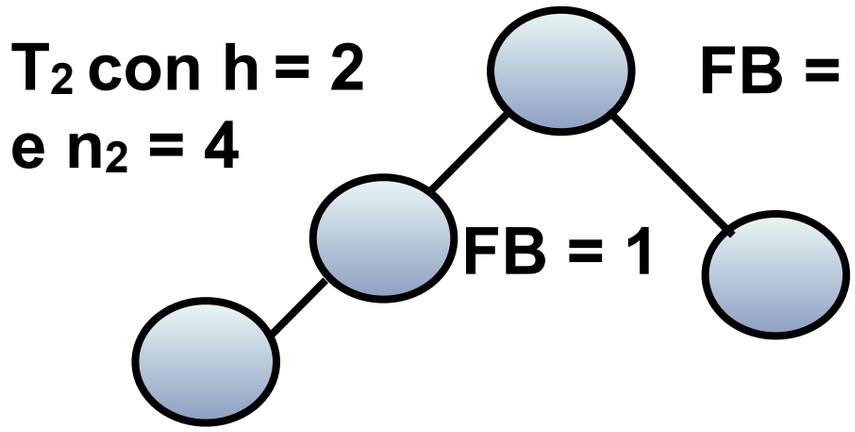
FB = 1

FB=0

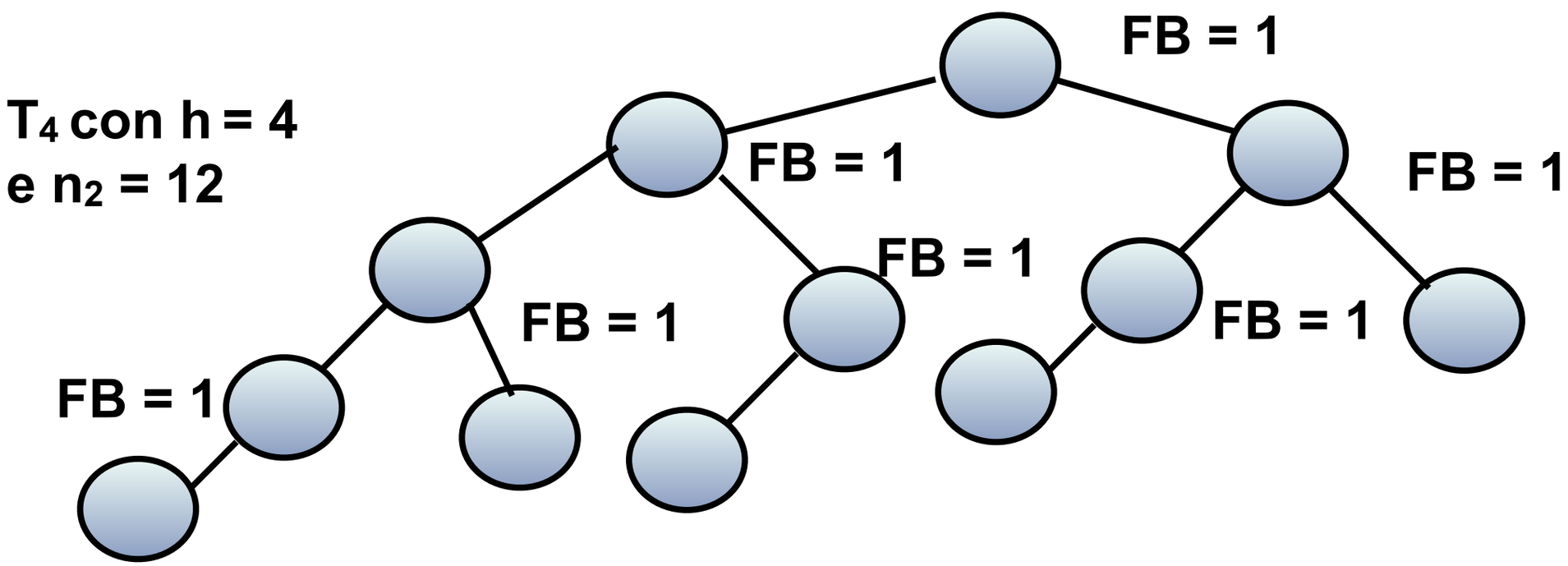
FB=0

Esempio costruzione dei più sbilanciati

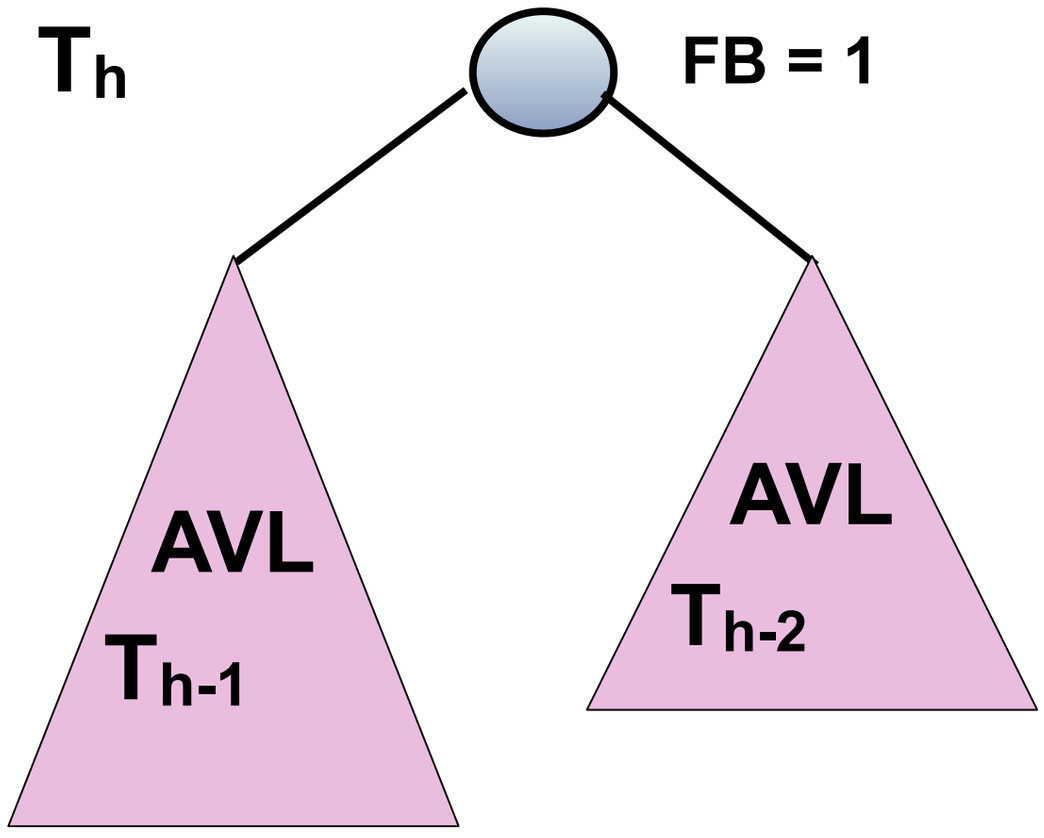




ogni nodo interno ha un FB = + 1



Alberi più sbilanciati calcolo numero dei nodi



Numero nodi in un AVL più sbilanciato a sinistra di altezza h:

$n_0 = 1$

$n_1 = 2$

$n_h = n_{h-1} + n_{h-2} + 1$ per $h \geq 2$

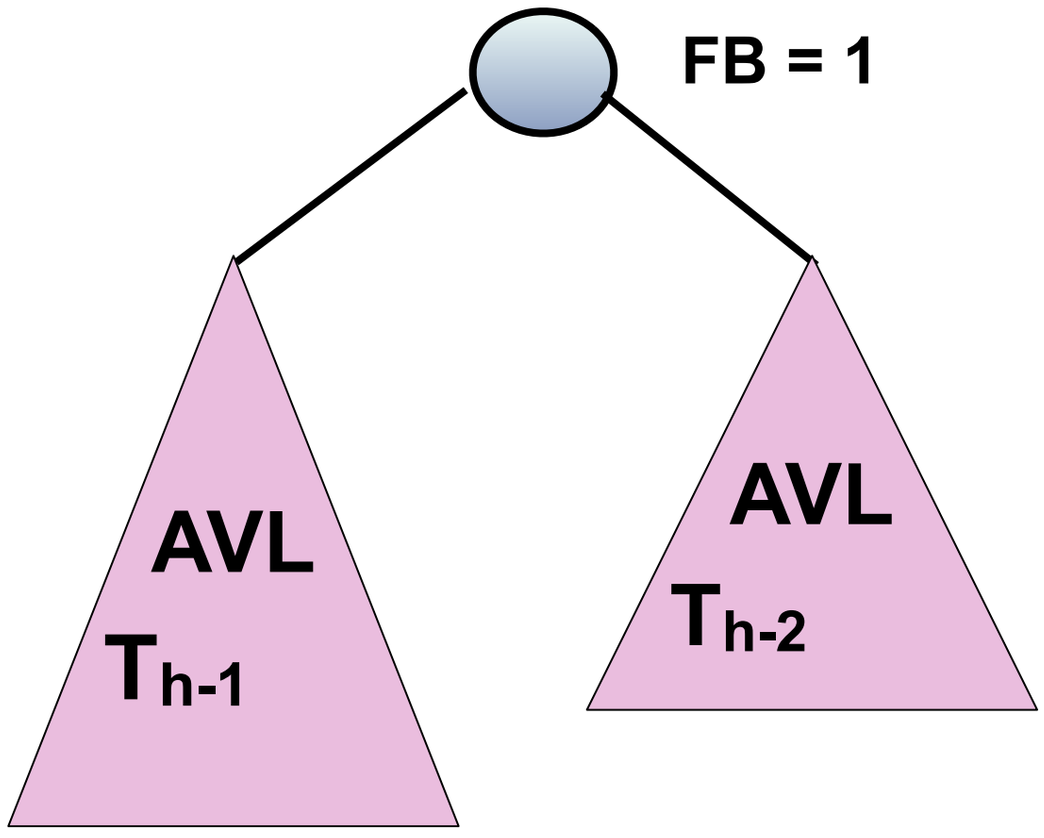
Serie di Fibonacci

$Fib_1 = 1$

$Fib_2 = 1$

$Fib_h = Fib_{h-1} + Fib_{h-2}$

Alberi di Fibonacci calcolo numero dei nodi



Numero nodi in un AVL più sbilanciato a sinistra di altezza h:

$n_0 = 1$

$n_1 = 2$

$n_h = n_{h-1} + n_{h-2} + 1$ per $h \geq 2$

Fibonacci

$Fib_1 = 1$

$Fib_2 = 1$

$Fib_h = Fib_{h-1} + Fib_{h-2}$

$Fib_1 = 1$ e $n_1 = 2$

Per $h \geq 1$, si ha $n_h > Fib_h$

Si dimostra facilmente per induzione su h

Alberi di Fibonacci: conclusione

Calcolo del limite superiore per l'altezza di un AVL:

$n_0 = 1$

$n_1 = 2$

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

Poiché per $h \geq 2$

$n_{h-1} \geq n_{h-2}$

possiamo dire che

se h è pari $n_h \geq 2n_{h-2} \geq 2*2\ n_{h-4} \geq 2*2*2\ n_{h-6} \ldots \geq 2^{h/2} n_0 = 2^{h/2}$

altrimenti $n_h \geq 2n_{h-2} \geq 2*2\ n_{h-4} \geq 2*2*2\ n_{h-6} \ldots \geq 2^{h/2} n_1 \geq 2^{h/2}$

Quindi per un qualsiasi AVL con n nodi e altezza h si ha

$$n \geq n_h \geq 2^{h/2}$$

applicando il logaritmo si ottiene $\lg n \geq h/2$,

per cui si può concludere $h = O(\lg n)$.

Si può dimostrare che $h \leq 1.44 \lg n$.

Alberi AVL: altezza in θ(lgn)

Per un qualsiasi albero AVL T di altezza h si ha $h = O(\lg n)$.

Il limite è stato ottenuto seguendo i seguenti passi:
1. abbiamo individuato gli alberi di Fibonacci come i più sbilanciati "a sinistra" tra gli AVL, cioè gli alberi che a parità di altezza hanno il minor numero di nodi.

2. Abbiamo dimostrato che $2^{h/2}$ è un limite inferiore al numero dei nodi di un albero di Fibonacci di altezza h.

3. Da 1 sappiamo che un qualsiasi AVL di altezza h ha un numero di nodi, n, maggiore o uguale a quello di un albero di Fibonacci della stessa altezza, quindi abbiamo dedotto che $n \geq 2^{h/2}$, da cui la conclusione.

Poiché per un qualsiasi albero binario T di altezza h si ha che $h = \Omega(\lg n)$, concludiamo che se T è un qualsiasi albero AVL di altezza h, vale

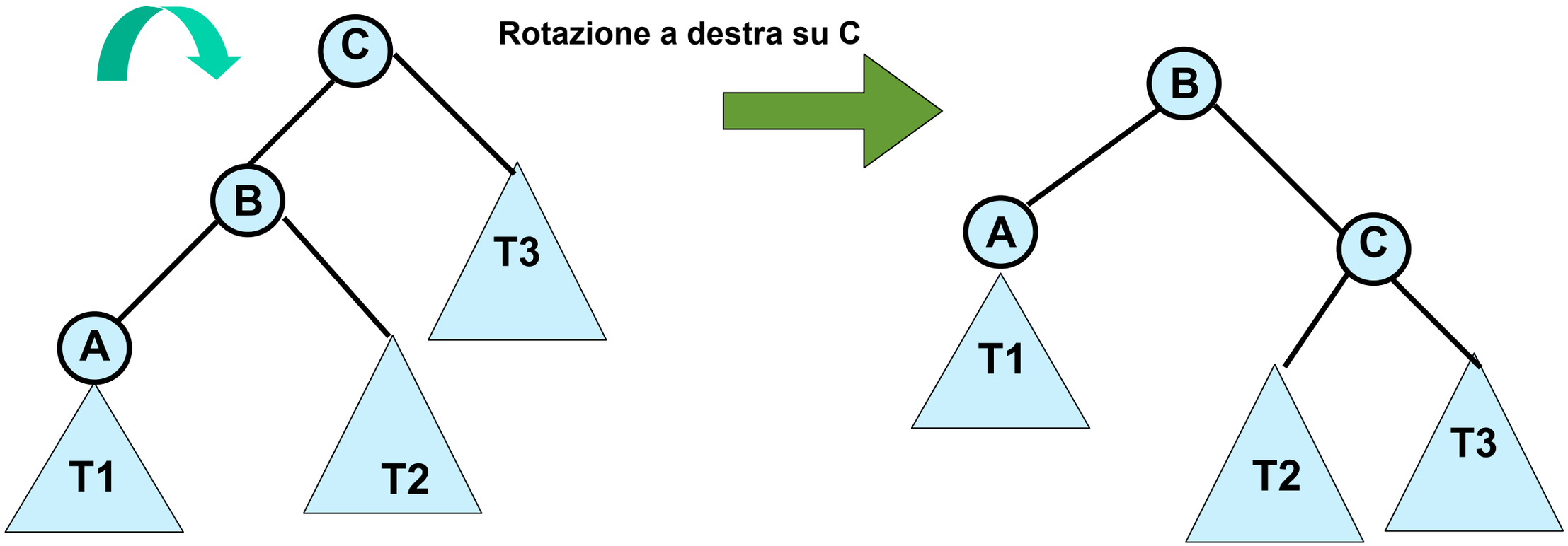
$$h = \theta(\lg n)$$

Operazioni di ribilanciamento

Le operazioni di inserimento e cancellazione sono le stesse usate nel caso degli ABR qualunque, ma seguite dai passi necessari per ribilanciare l'albero, se necessario.

Si usano le rotazioni.

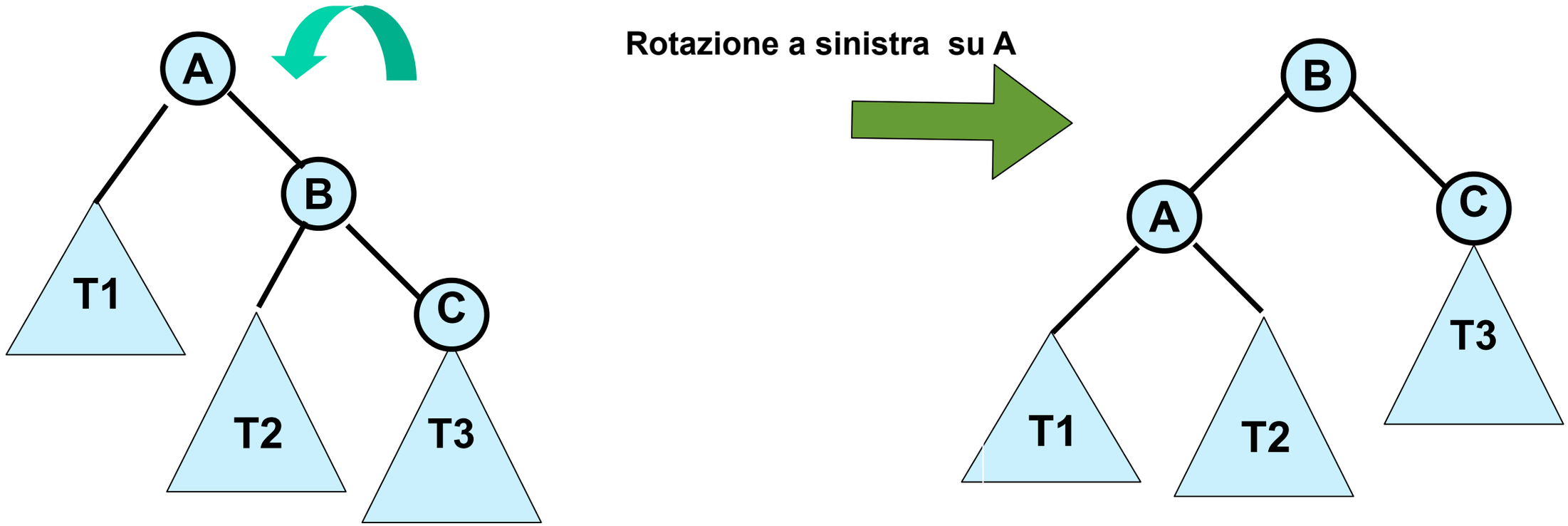
Rotazioni su ABR



Dopo la rotazione l'albero è ancora un ABR, perchè se a è un nodo in T1, b in T2 e c in T3 allora vale a < A < B < b < C < c anche dopo la rotazione.

Richiede tempo O(1)

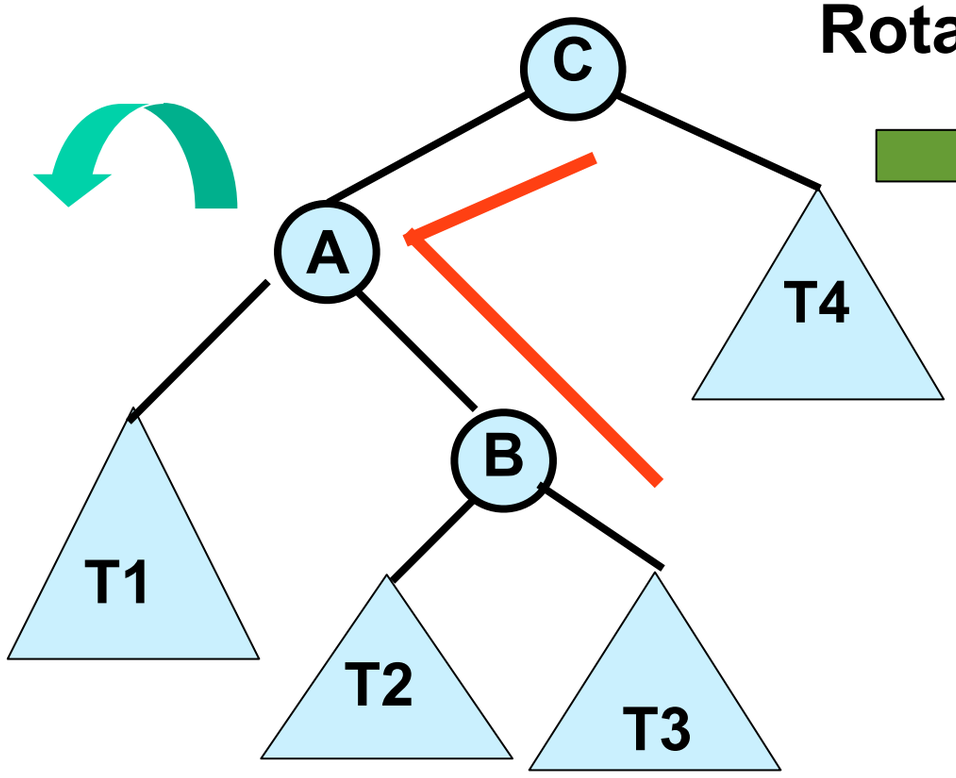
Rotazioni su ABR



Dopo la rotazione l'albero è ancora un ABR, perchè se a è un nodo in T1, b in T2 e c in T3 allora vale a < A < b < B < C < c anche dopo la rotazione.

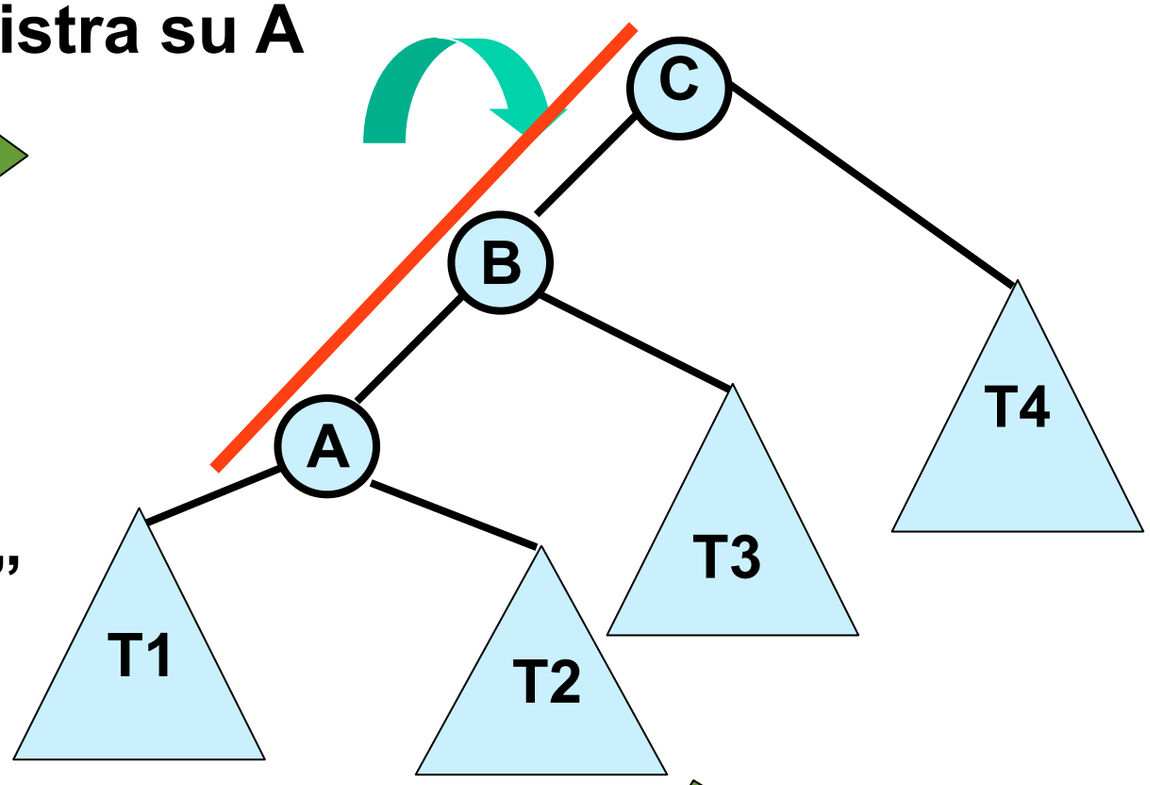
Richiede tempo O(1)

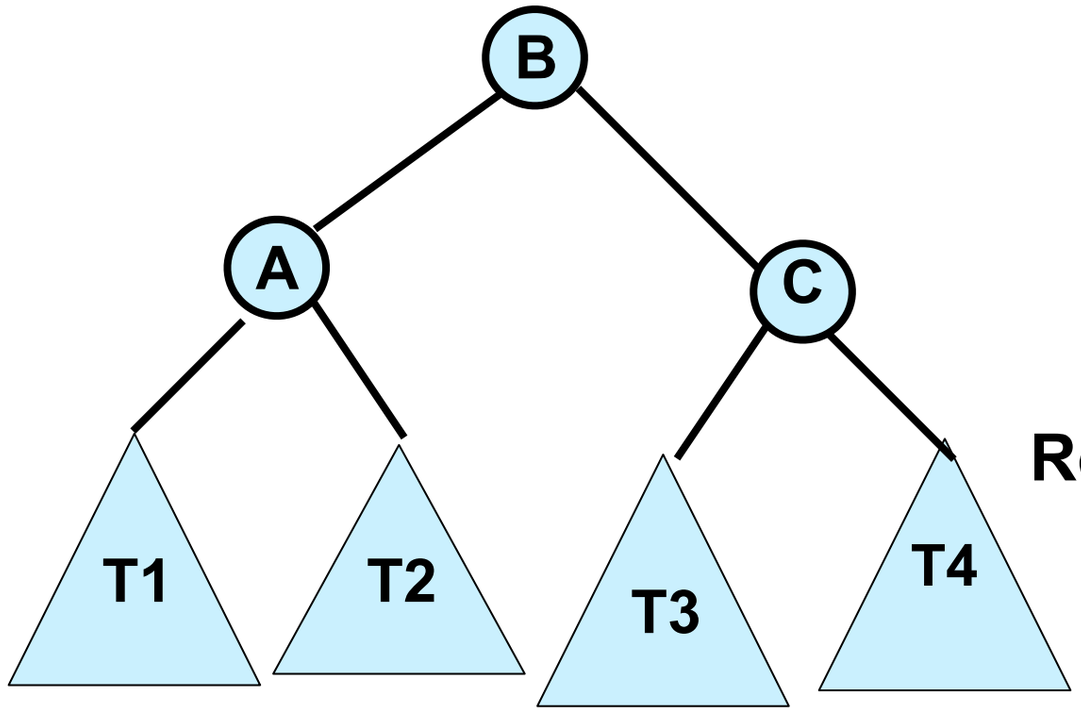
La doppia rotazione: sinistra poi destra



Rotazione a sinistra su A

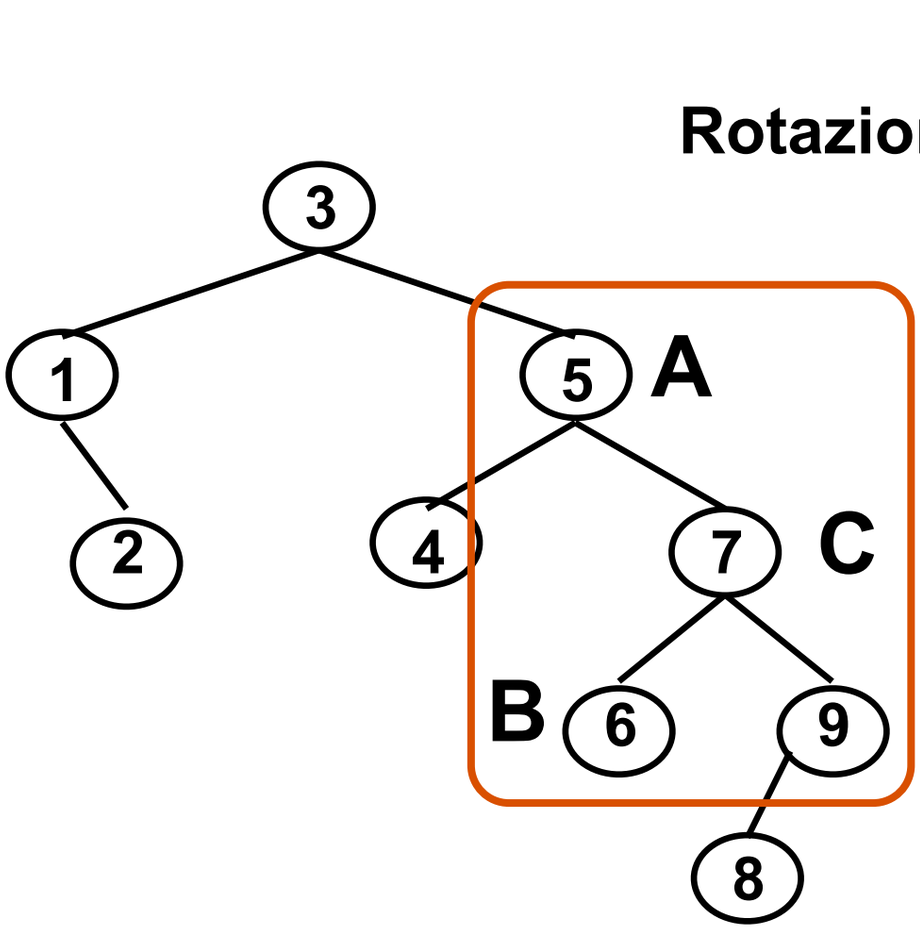
Si rientra nel caso precedente eliminando lo “spigolo” o “zig-zag”



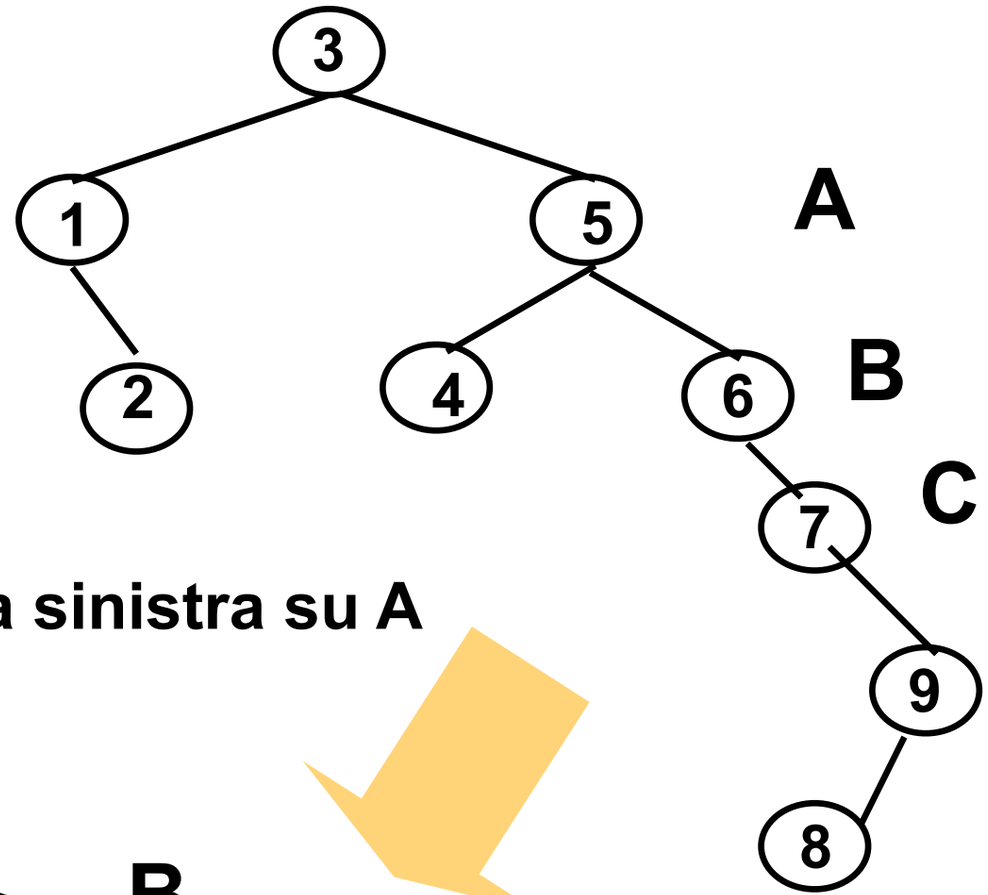


Rotazione a destra su C

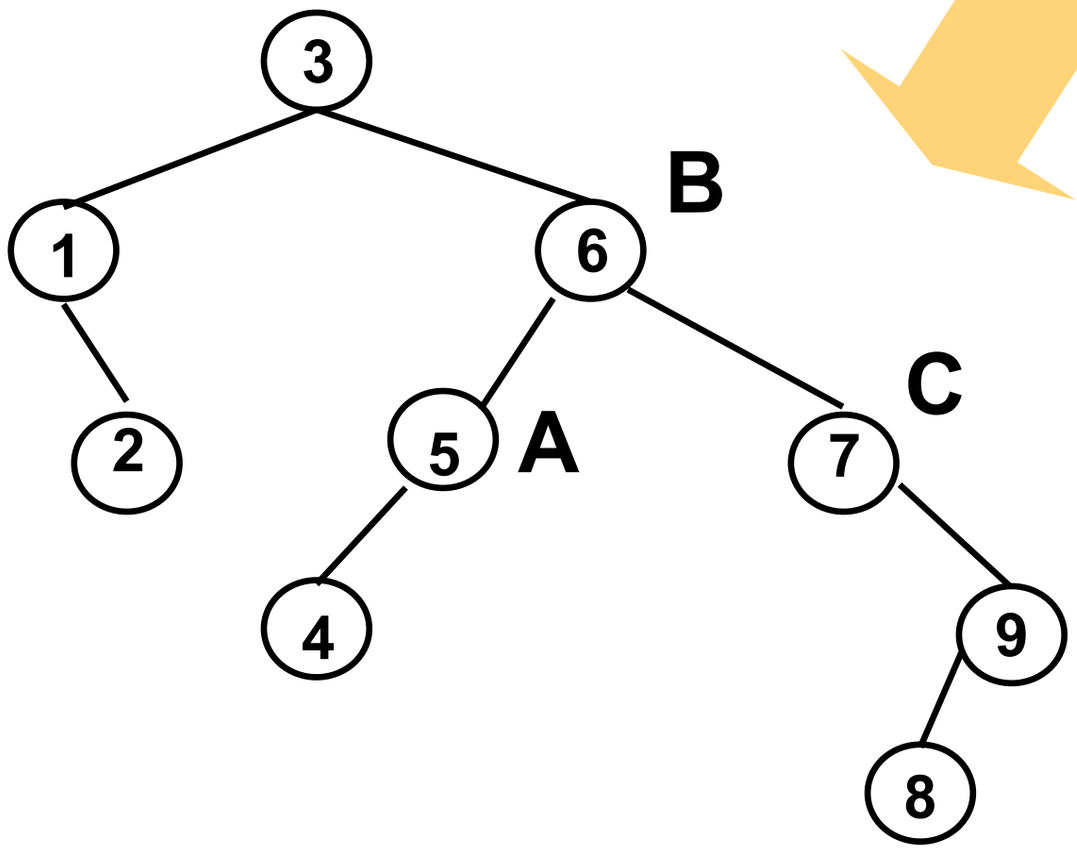
Esempio di doppia rotazione su 5



Rotazione a destra su C



Rotazione a sinistra su A



Operazioni su ABR in O(lg n)

In un ABR con n nodi e di altezza h le operazioni di

- ricerca
- inserimento
- cancellazione

hanno una complessità asintotica O(h),

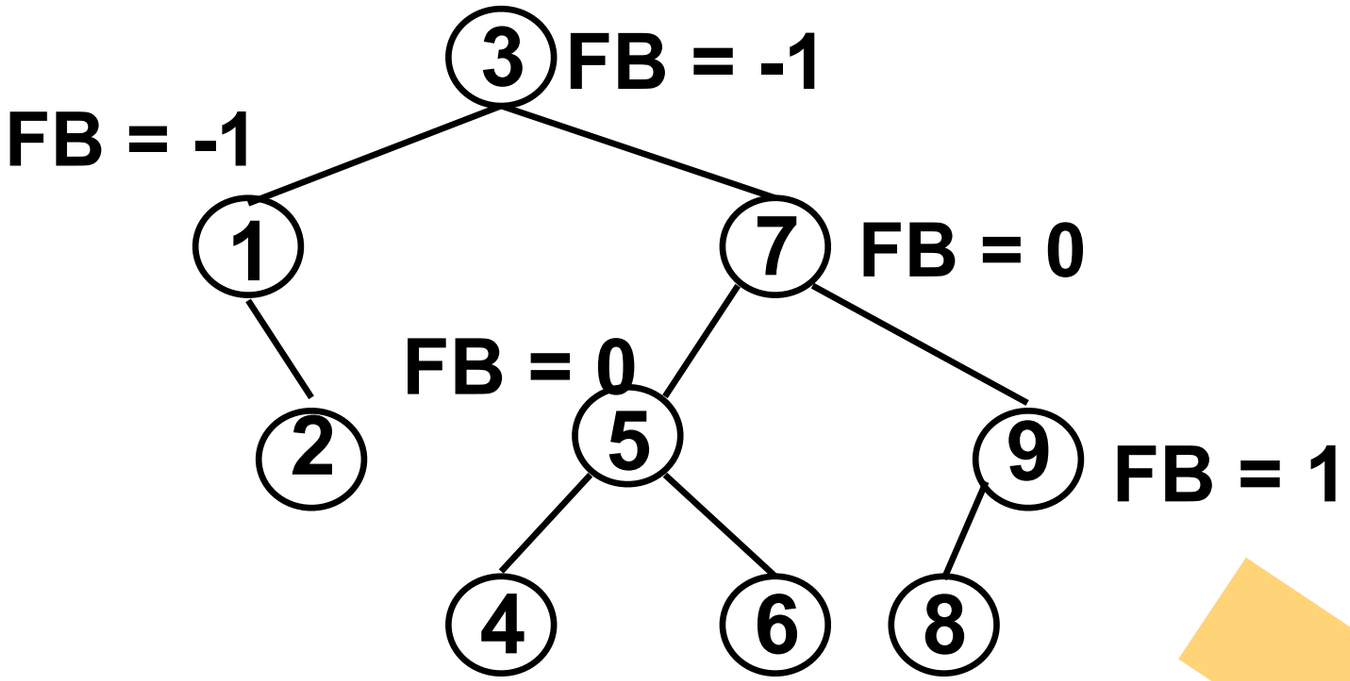
con lg n ≤ h < n

in un albero bilanciato in altezza dimostreremo che hanno una complessità asintotica O(lg n), perchè l'altezza è O(lg n) e il ribilanciamento dell'albero può essere fatto in O(lg n).

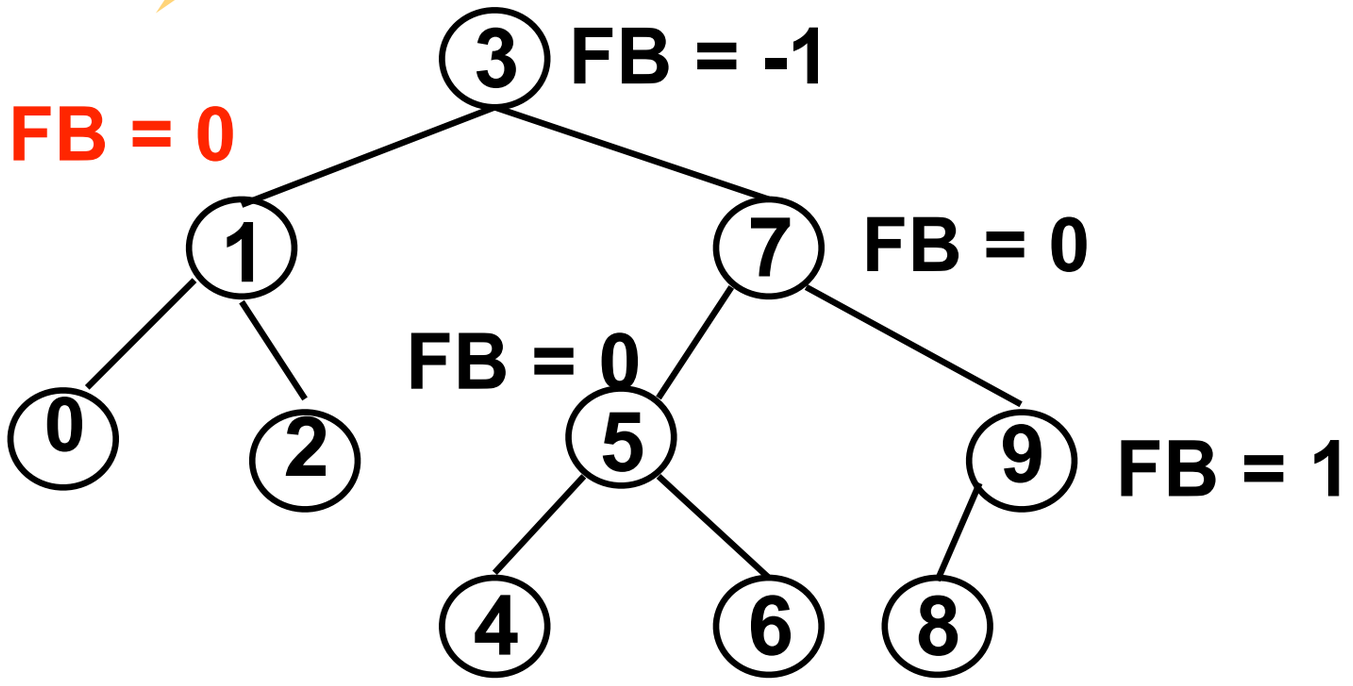
Per ottenere questo si adotta una struttura dati aumentata, in cui in ogni nodo oltre ai campi key,left,right,p, si ha il campo FB, la differenza tra l'altezza del sotto albero sinistro meno quella del destro, che quindi assume solo i valori -1,0 e 1.

Per FB bastano 2 bits.

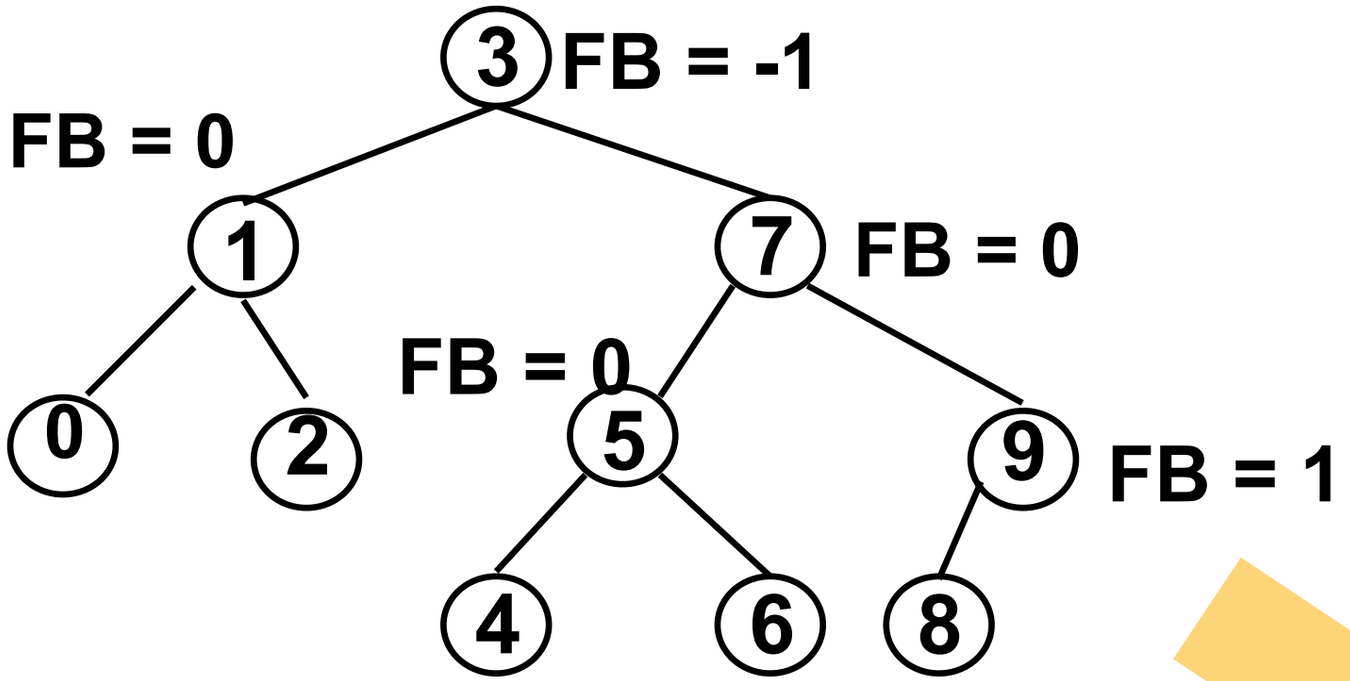
Esempio inserimento in AVL



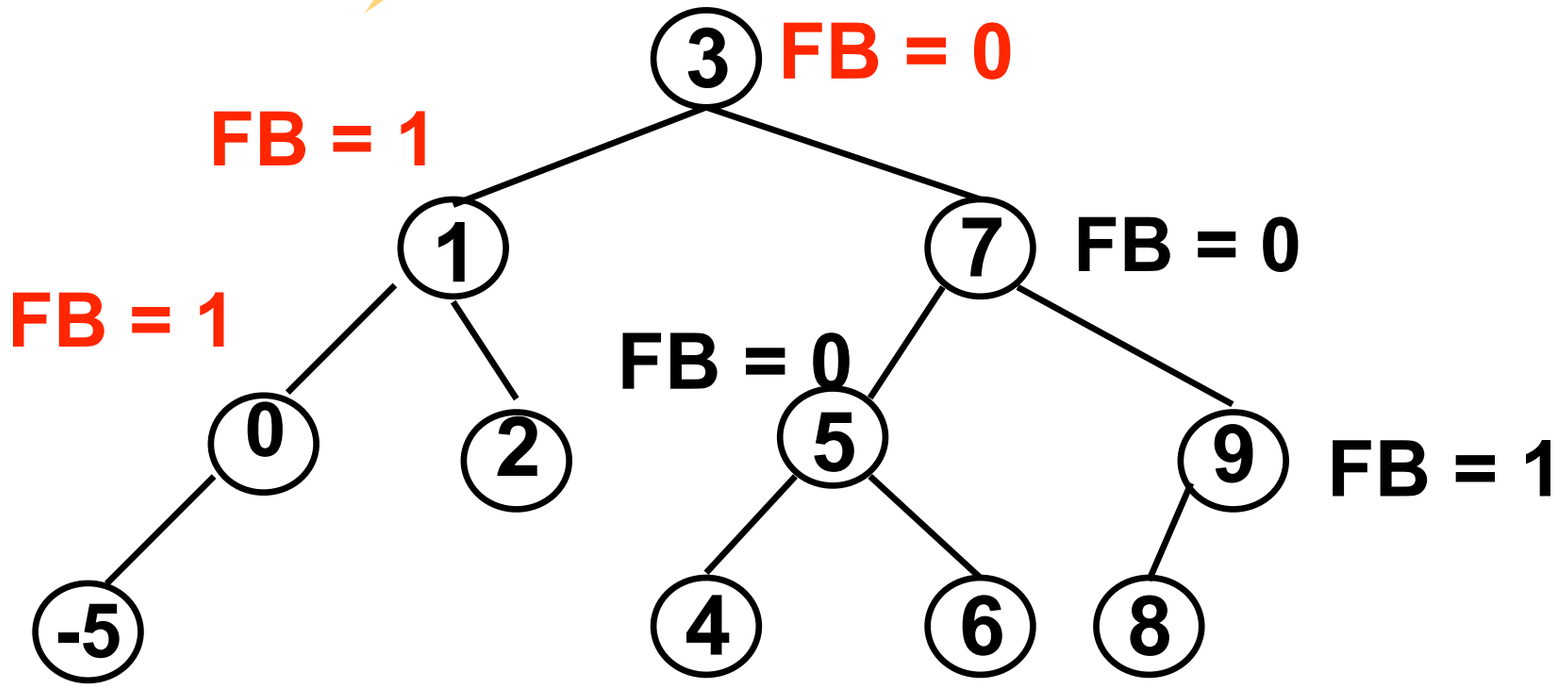
Inserimento di 0



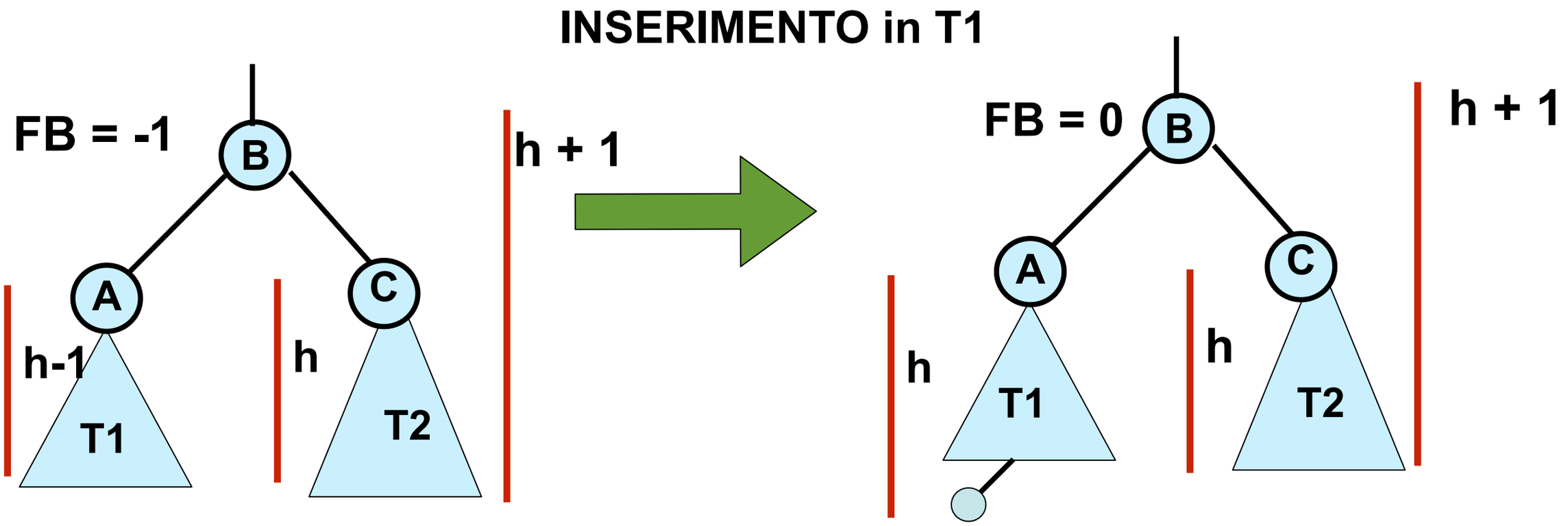
Esempio inserimento in AVL



Inserimento di -5

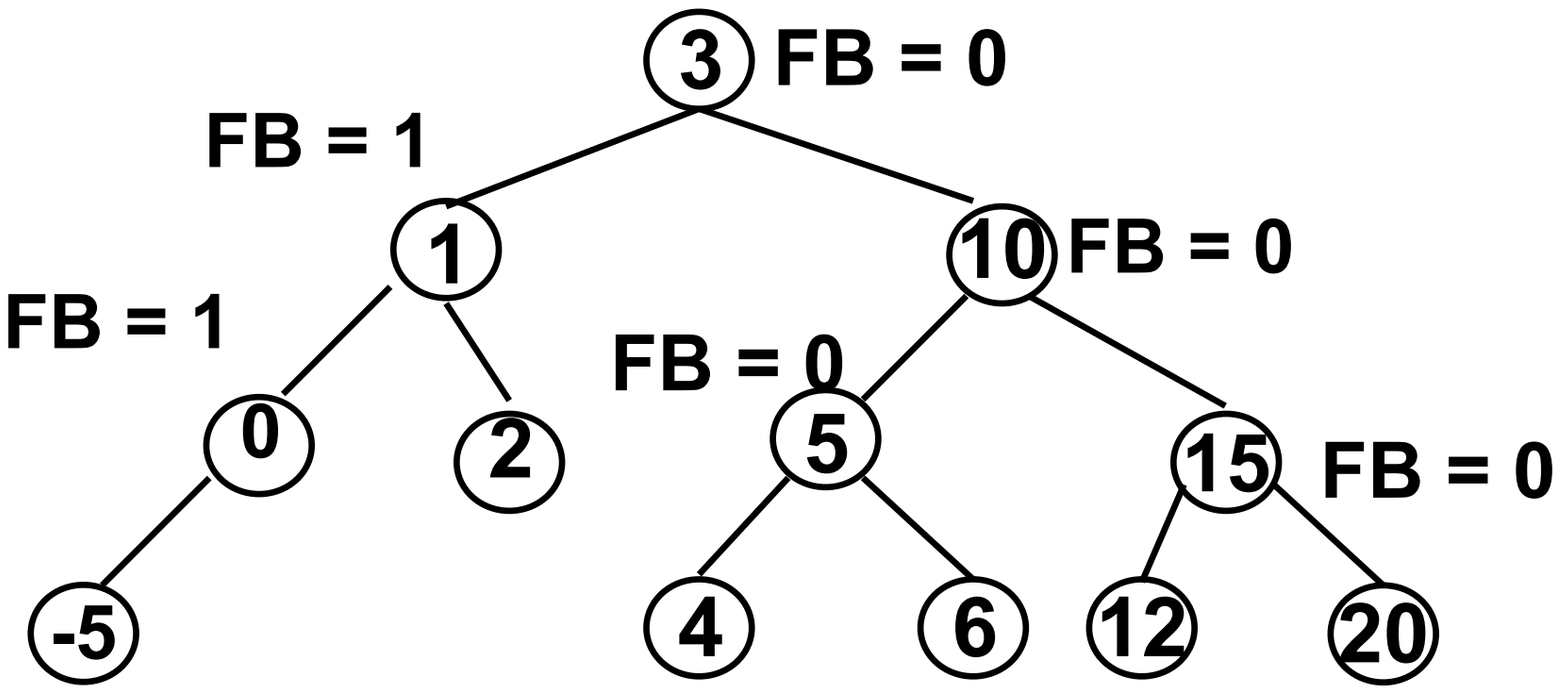


INSERIMENTO: aggiornamento FB

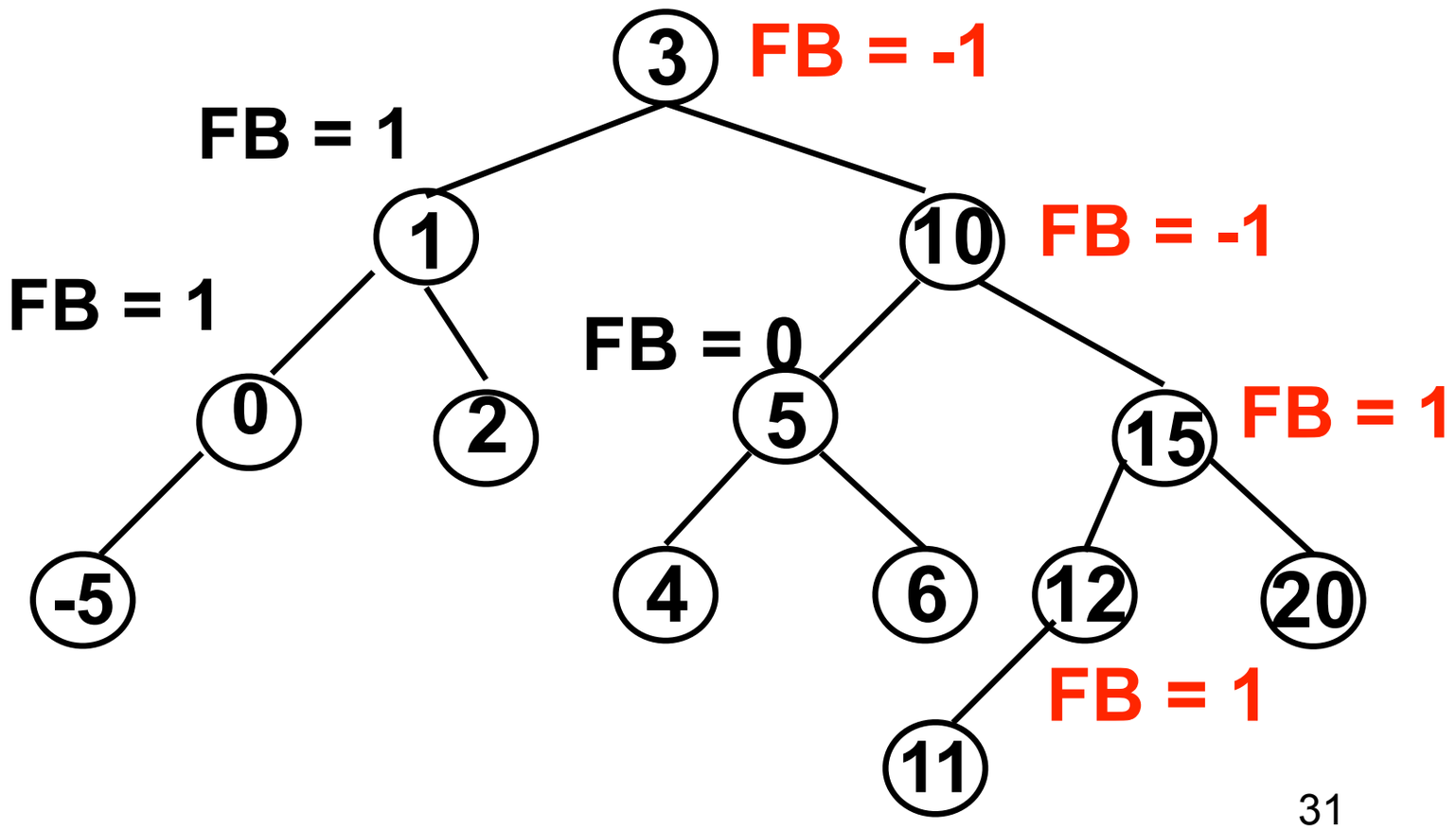


Quando risalendo verso la radice il fattore di bilanciamento passa da -1 a 0, possiamo concludere che l'inserimento ha aumentato l'altezza del sottoalbero sinistro che però era più basso di 1 di quello destro. Dunque l'altezza del sottoalbero radicato in B , il nodo in cui il FB cambia da -1 a 0, non cambia e si può fermare il processo di aggiornamento dei FB.
Il caso simmetrico è analogo.

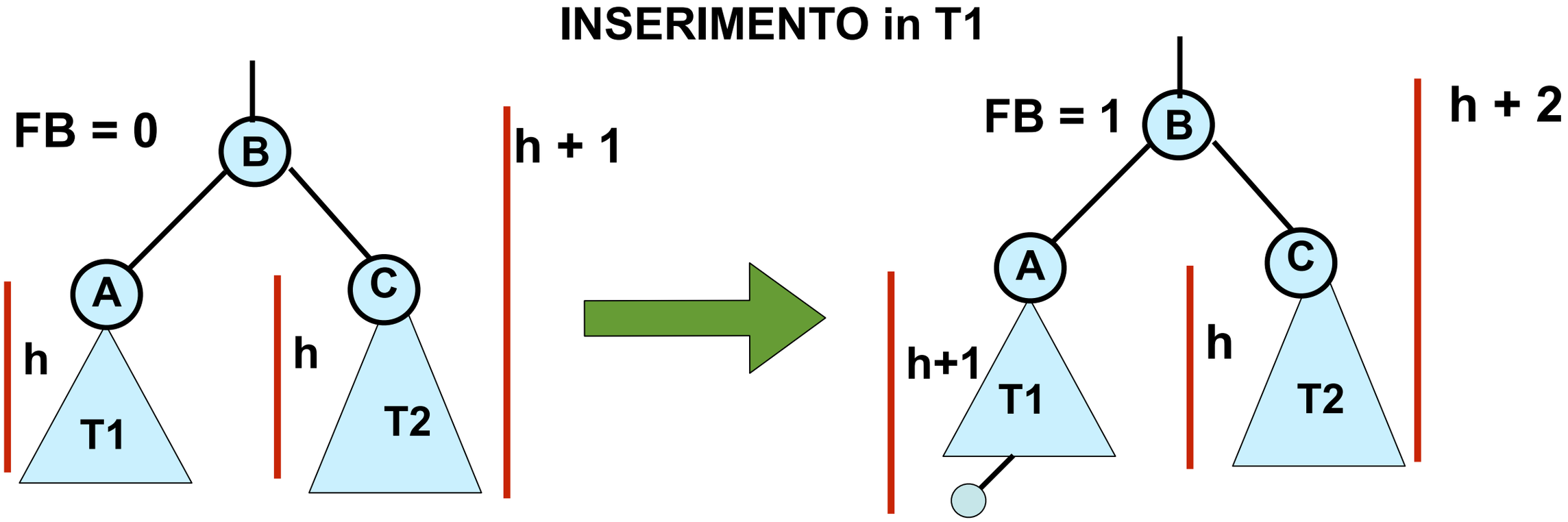
Esempio inserimento in AVL



Inserimento di 11

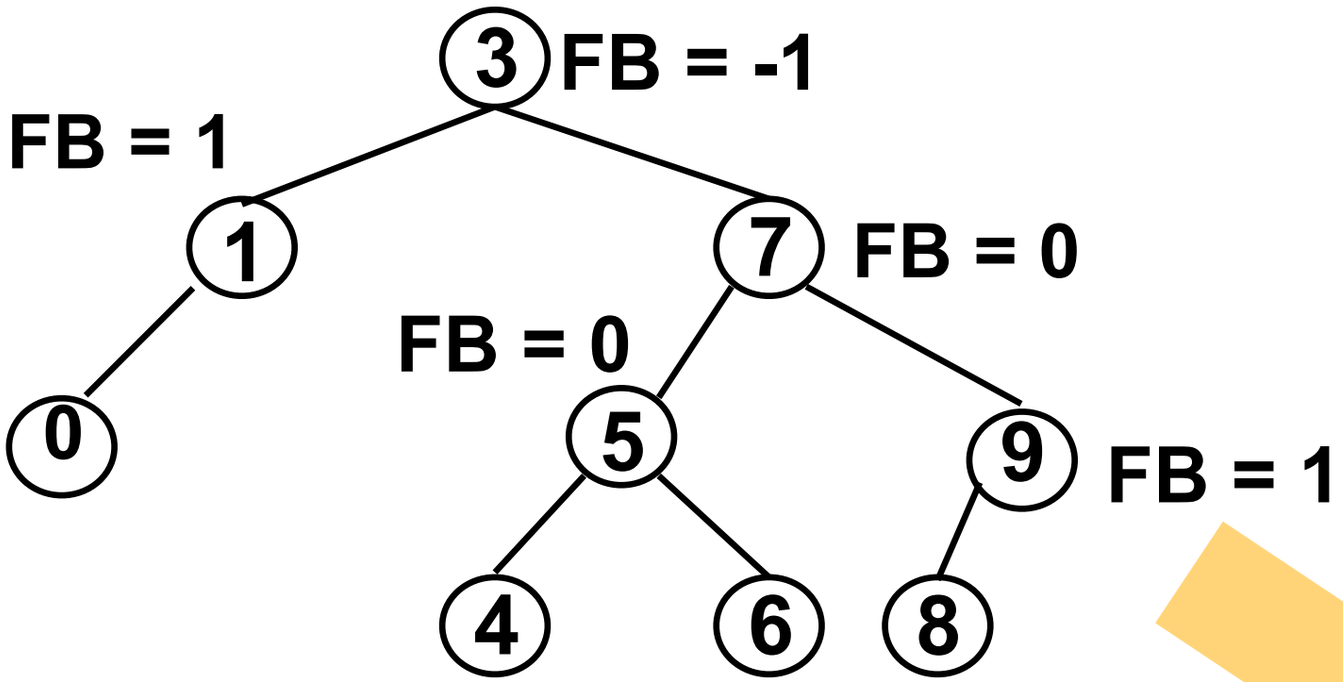


INSERIMENTO: aggiornamento FB

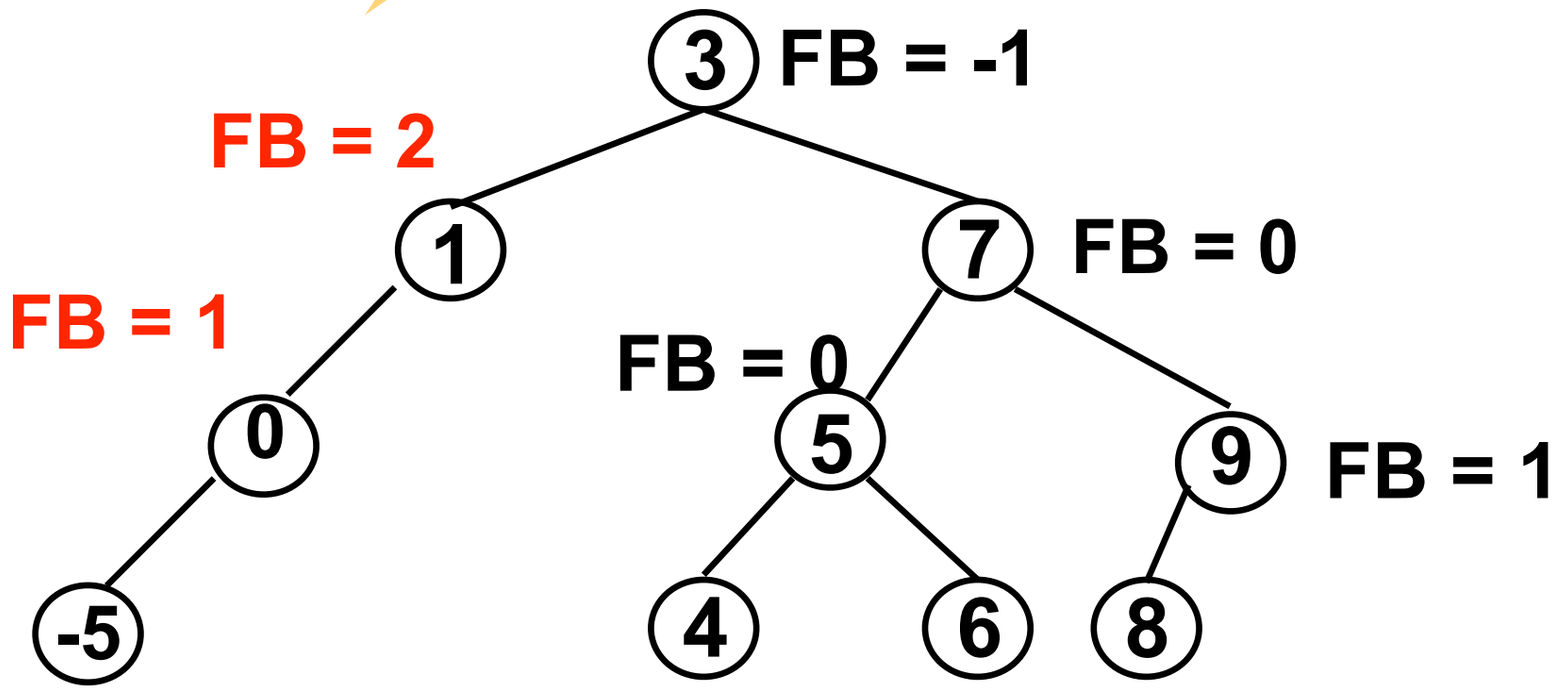


Risalendo verso la radice il fattore di bilanciamento passa da 0 a 1 perchè il sottoalbero sinistro ha aumentato la sua altezza e complessivamente tutto il sottoalbero radicato in B, quindi si deve controllare il padre di B.

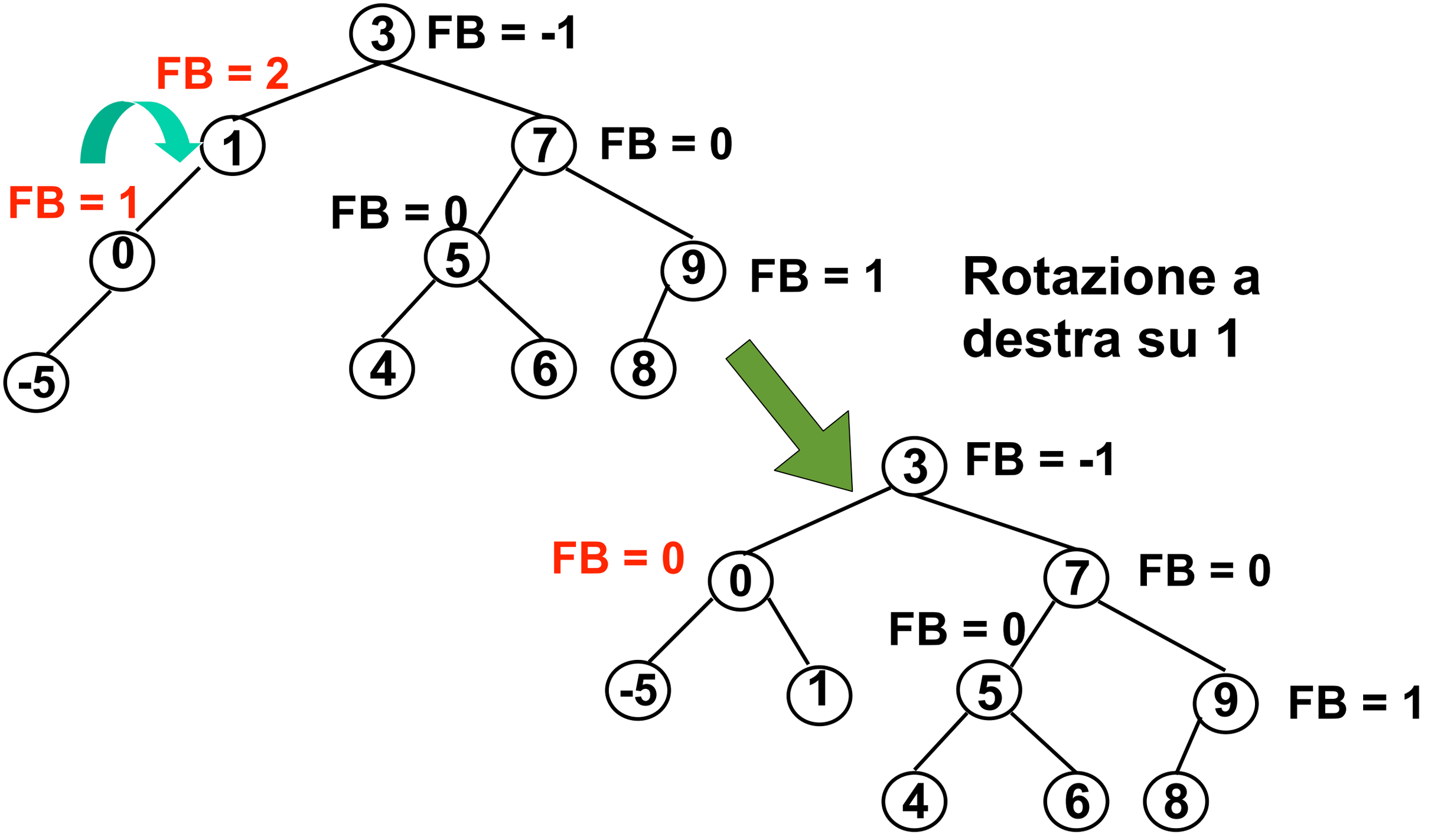
Inserimento: esempio ribilanciamento



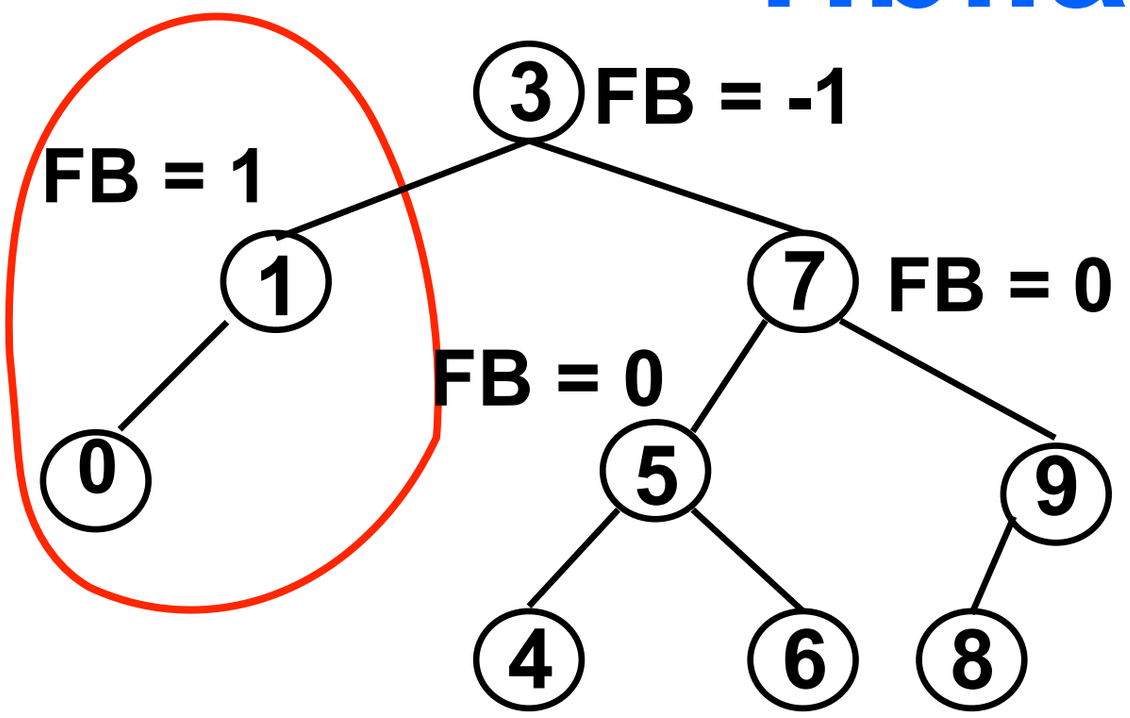
Inserimento di -5



Inserimento: esempio ribilanciamento

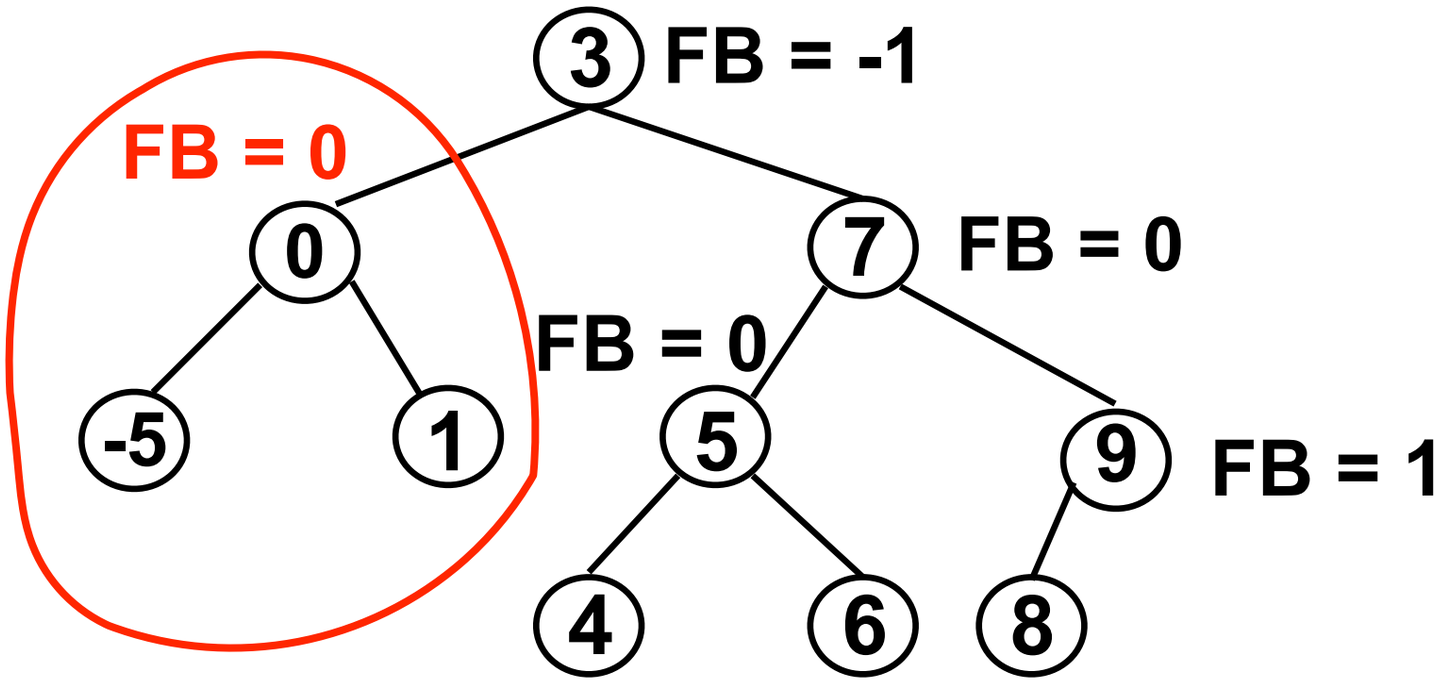


Inserimento: esempio ribilanciamento



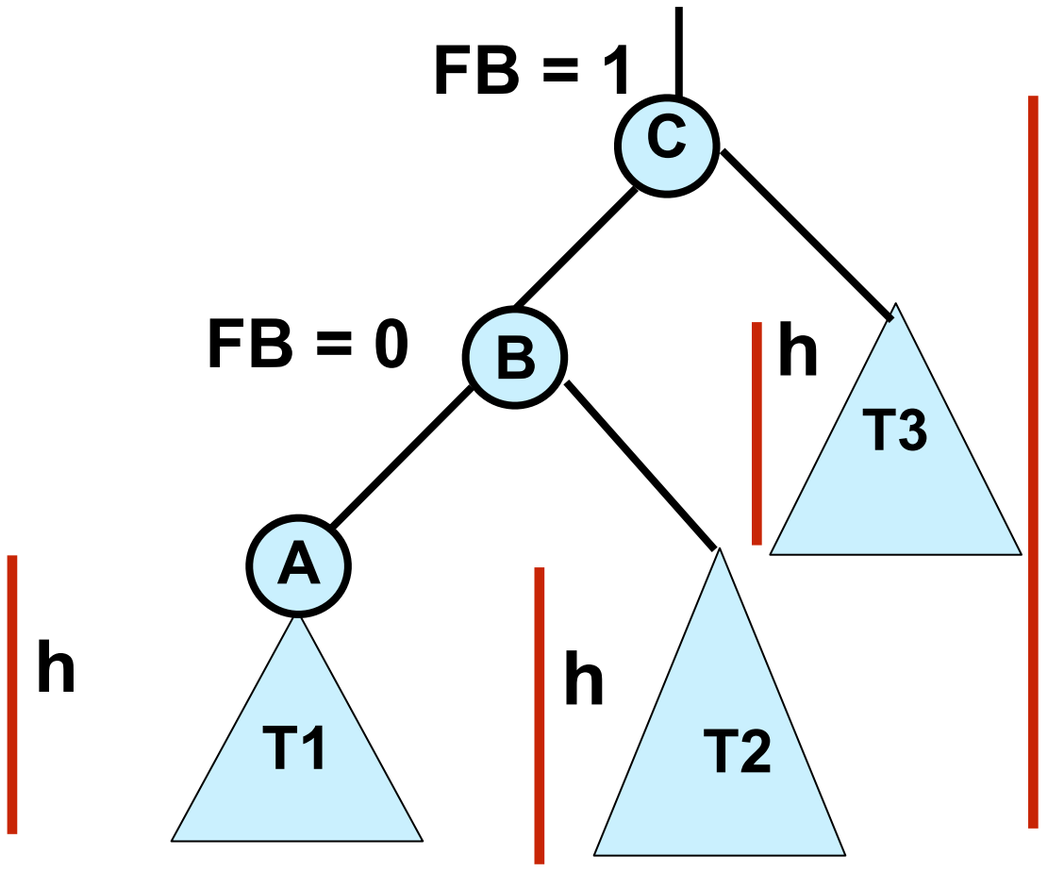
Prima dell'inserimento

I due sotto alberi hanno la stessa altezza, quindi non si dovrà risalire verso la radice, nemmeno per aggiornare i FB

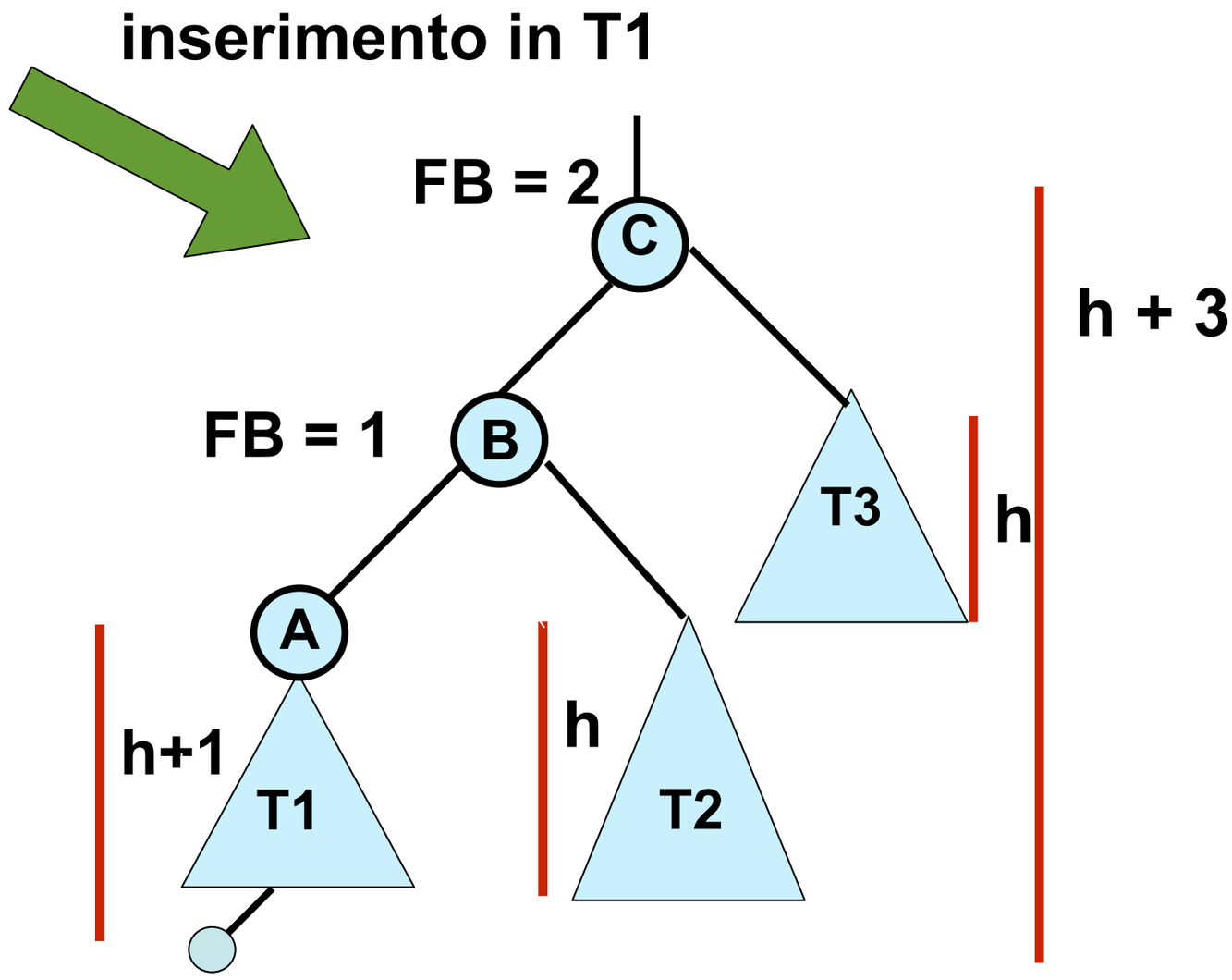


Dopo l'inserimento.

Inserimento: CASO LEFT-LEFT

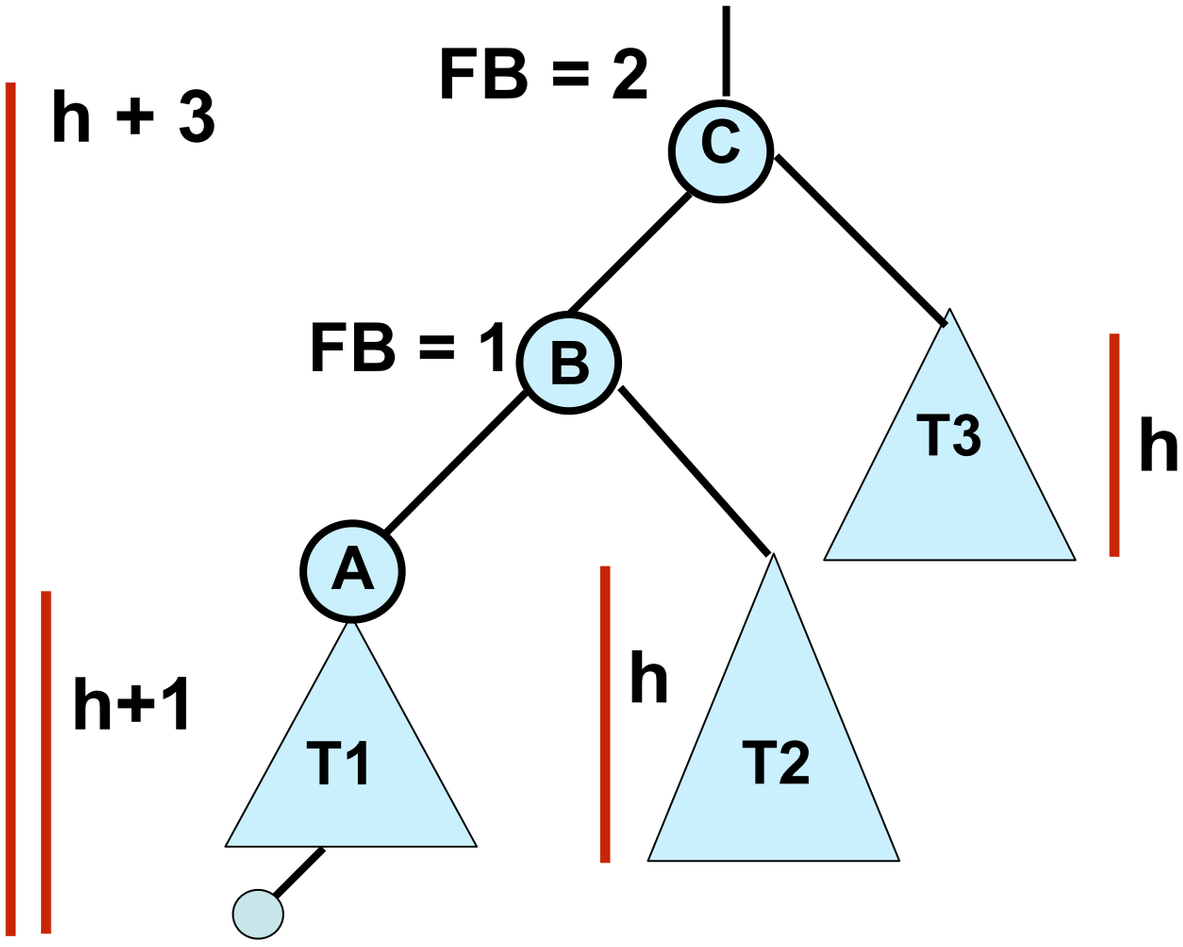


Inserimento nel <u>sotto albero sinistro del sotto albero sinistro</u>. Il nodo C può essere figlio sinistro o destro.

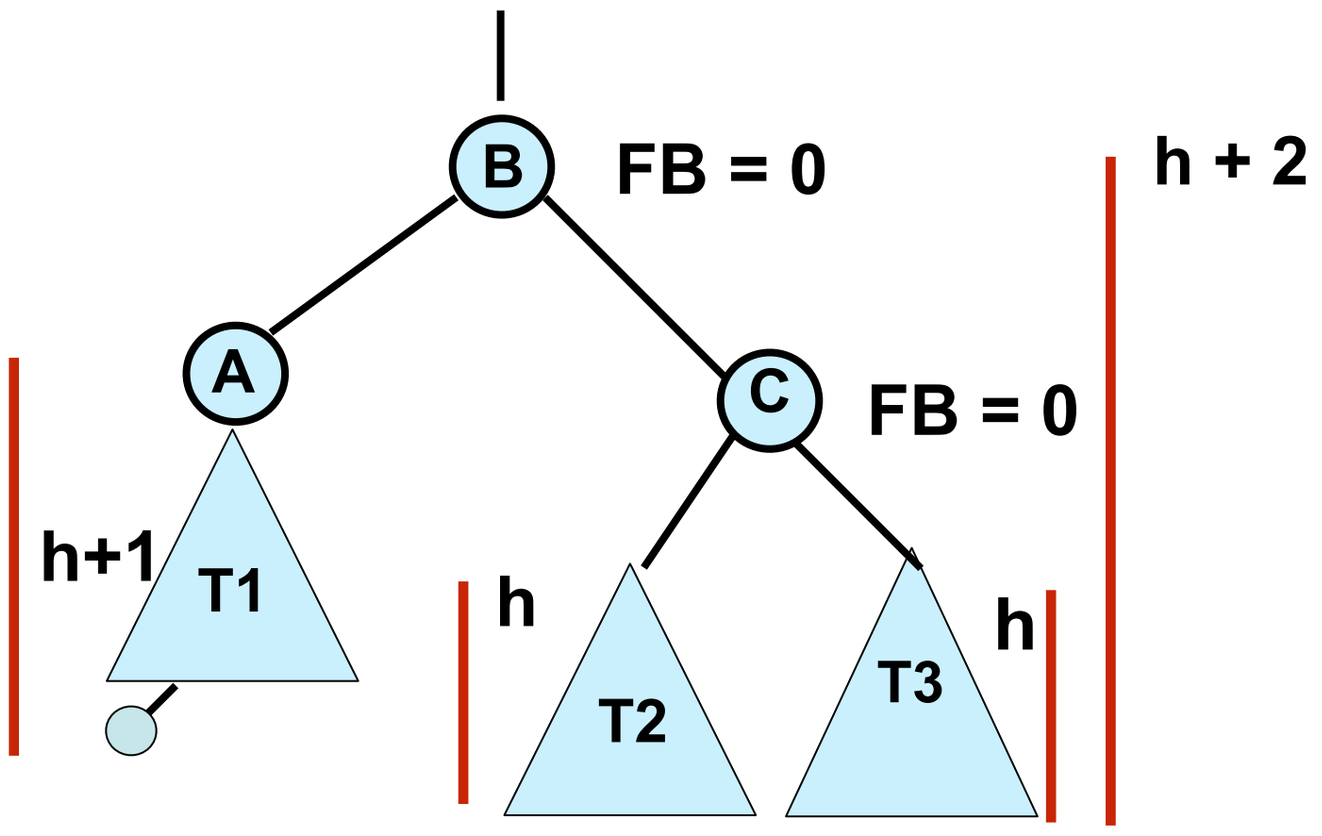


L'inserimento provoca un aumento di altezza nel sotto albero sinistro di C, nodo nel quale il fattore di bilanciamento diventa illegale. Risalendo da un nodo con FB = 1, passiamo a uno con FB = 2, questo identifica il caso: left-left.

CASO LEFT-LEFT

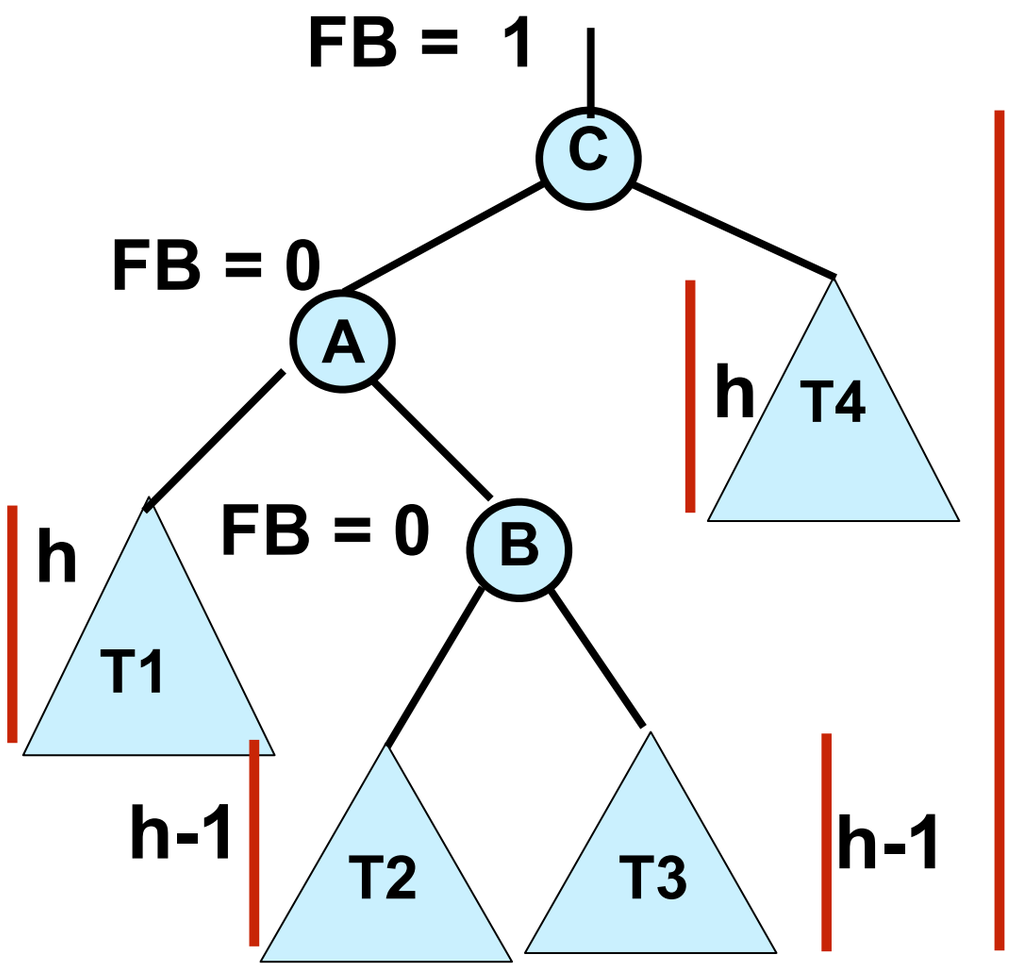


Rotazione a destra su C

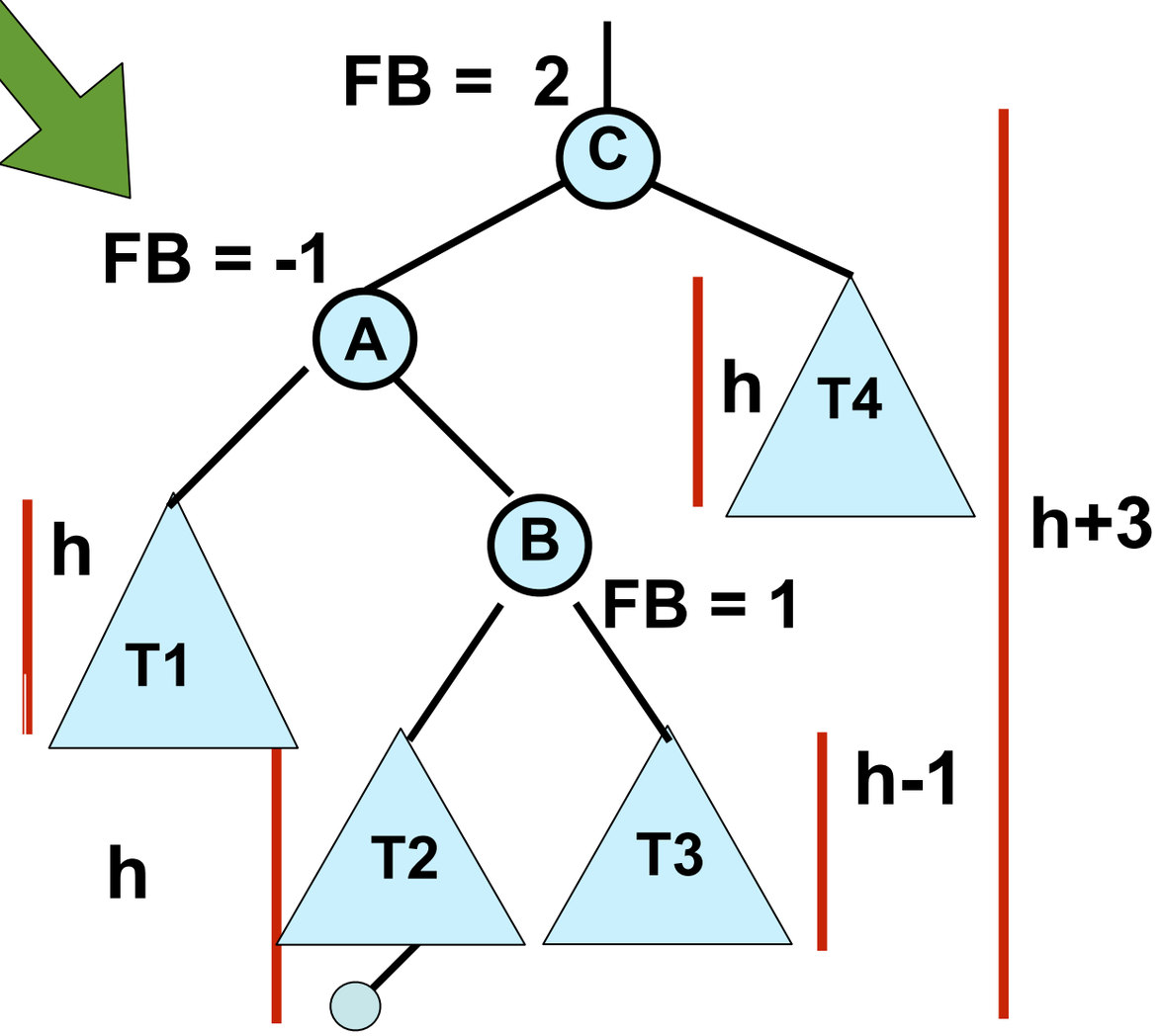


Dopo la rotazione l'altezza della radice del sottoalbero, B è quella del nodo C, <u>prima dell'inserimento</u>, quindi <u>non</u> è necessario risalire ancora verso la radice per ribilanciare l'albero!

CASO LEFT-RIGHT



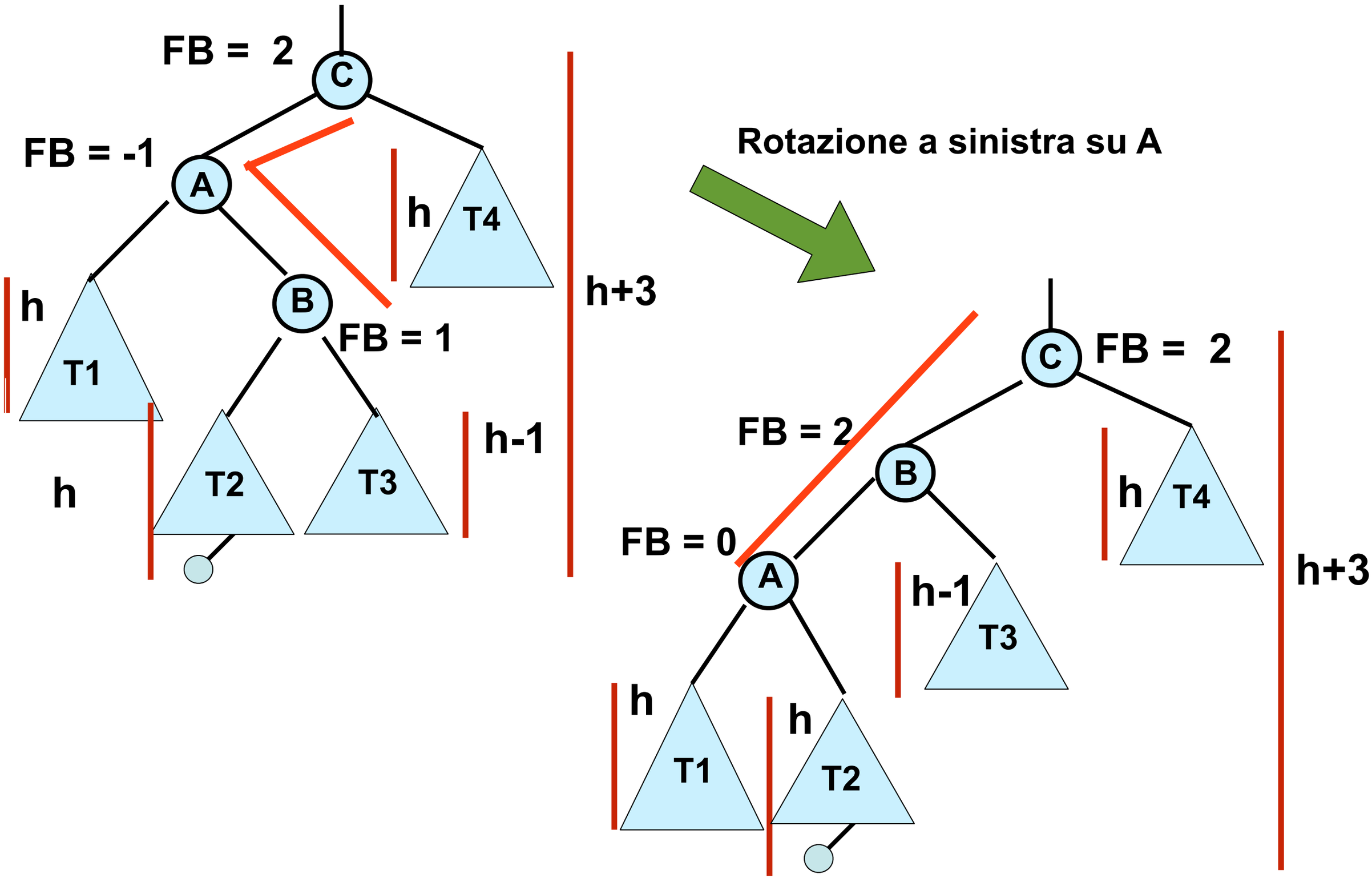
inserimento in T2
(caso simmetrico in T3, scambiando le altezze)



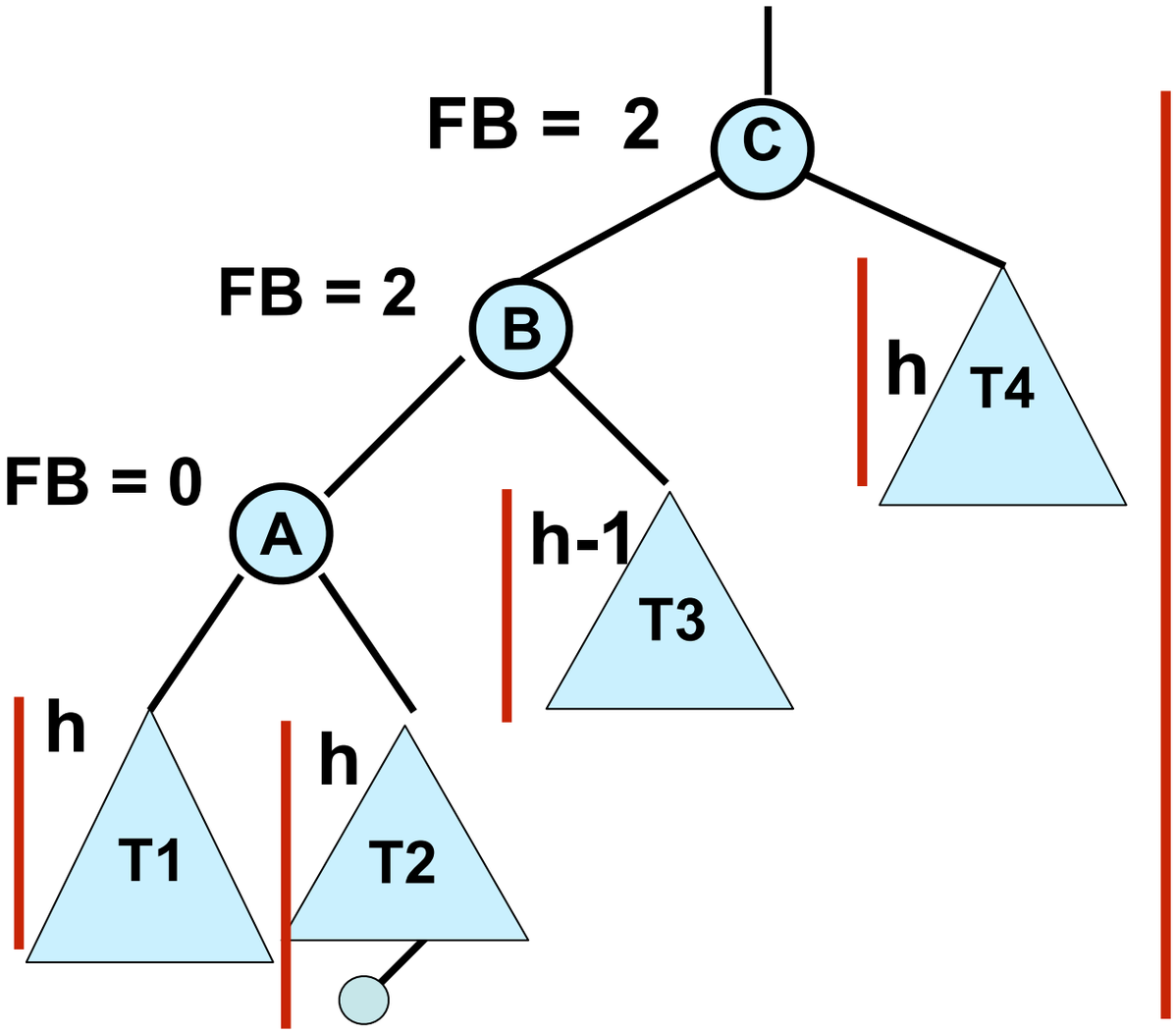
L'inserimento nel <u>sotto albero destro del sotto albero sinistro</u>.

Risalendo da un nodo con FB = -1, passiamo a uno con FB = 2 : questo identifica il caso: left-right.

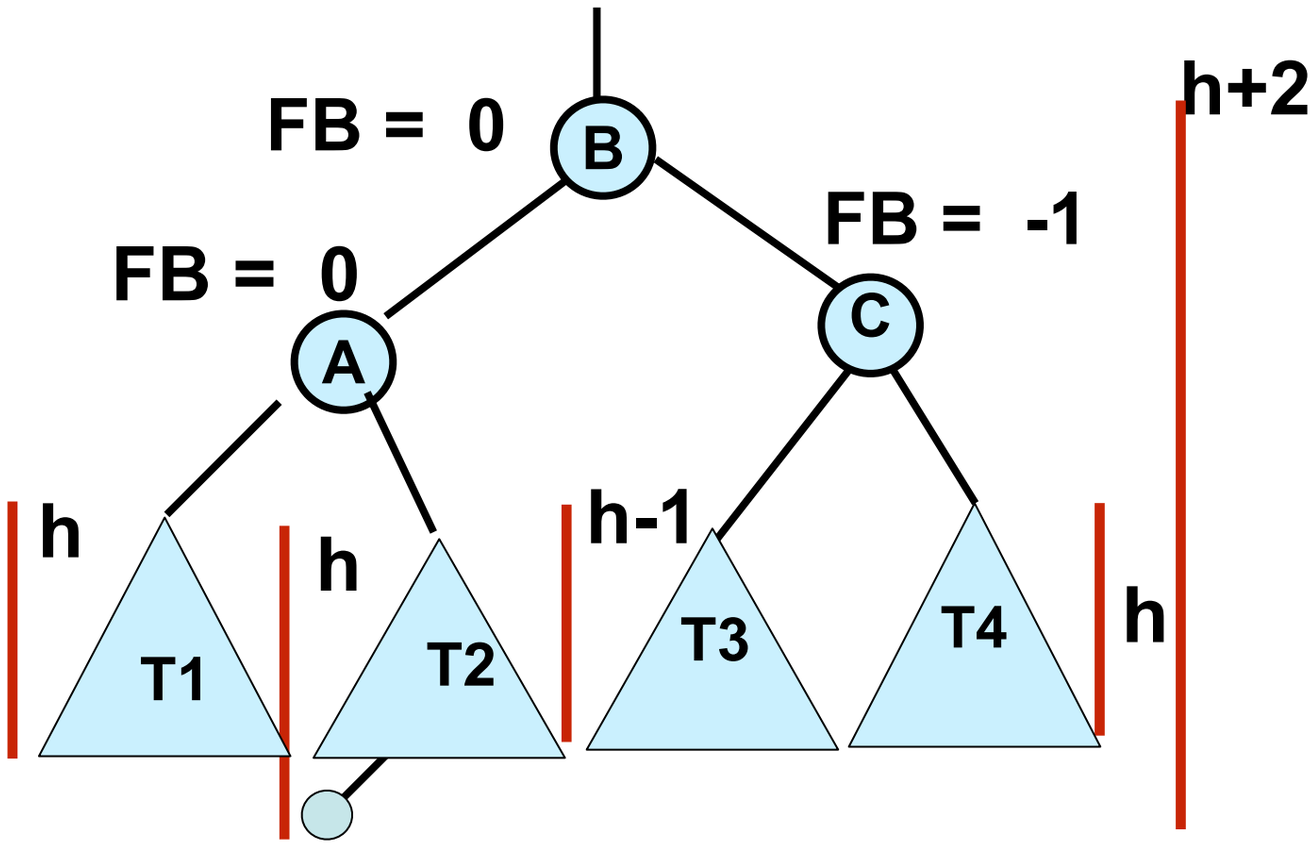
LA PRIMA ROTAZIONE - LEFT



La seconda rotazione - RIGHT

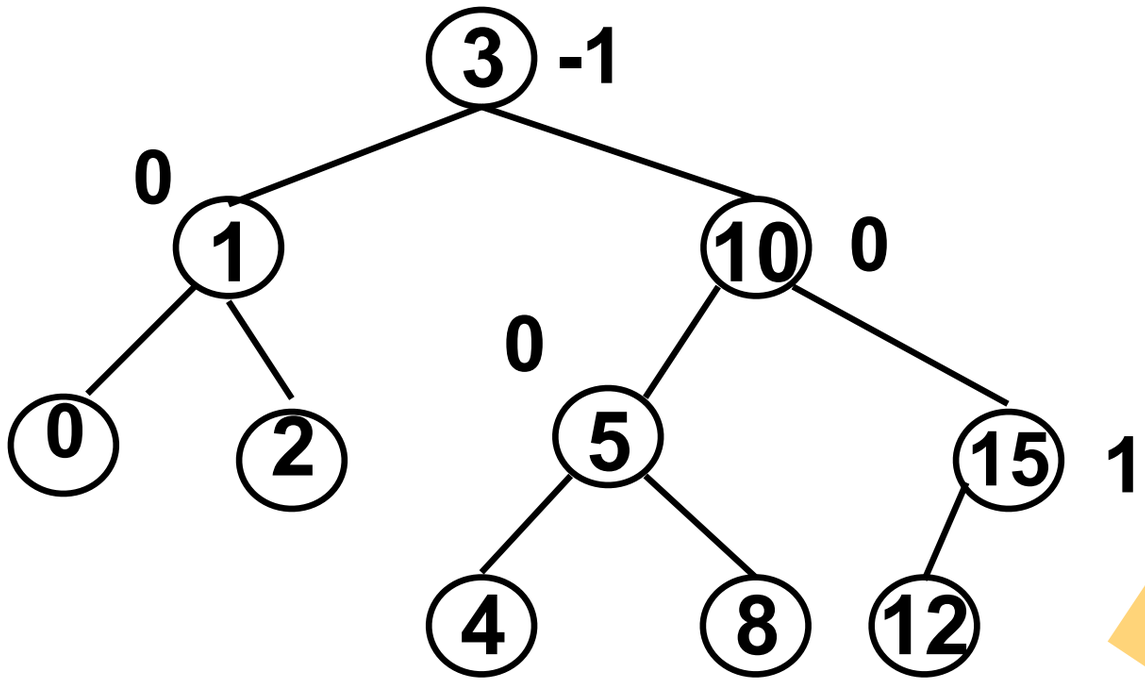


Rotazione a destra su C



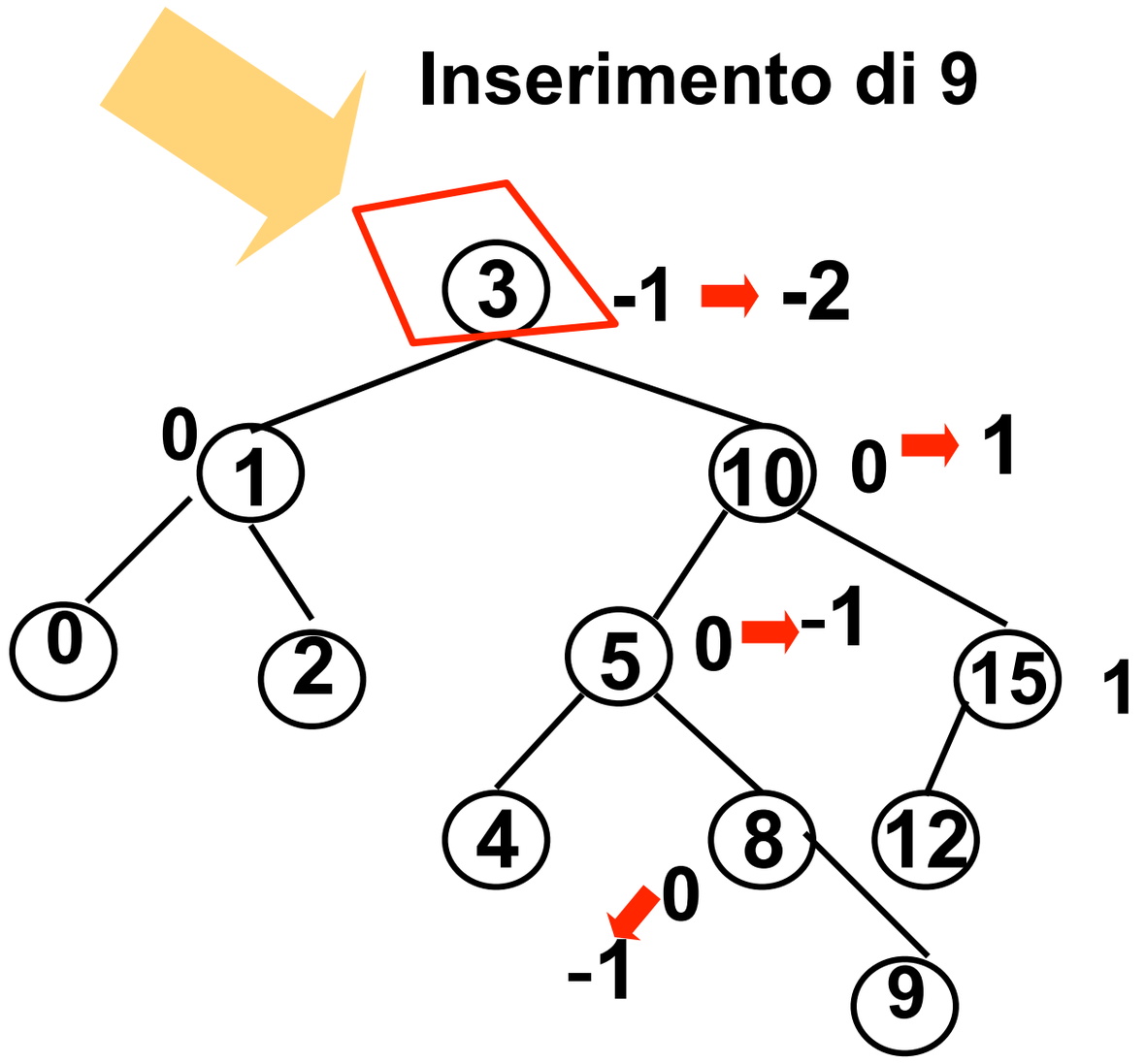
Anche in questo caso, <u>dopo la seconda rotazione l'altezza di B è quella del nodo C prima dell'inserimento</u>, quindi non è necessario risalire ancora verso la radice per ribilanciare l'albero!

Esempio 2 inserimento in AVL

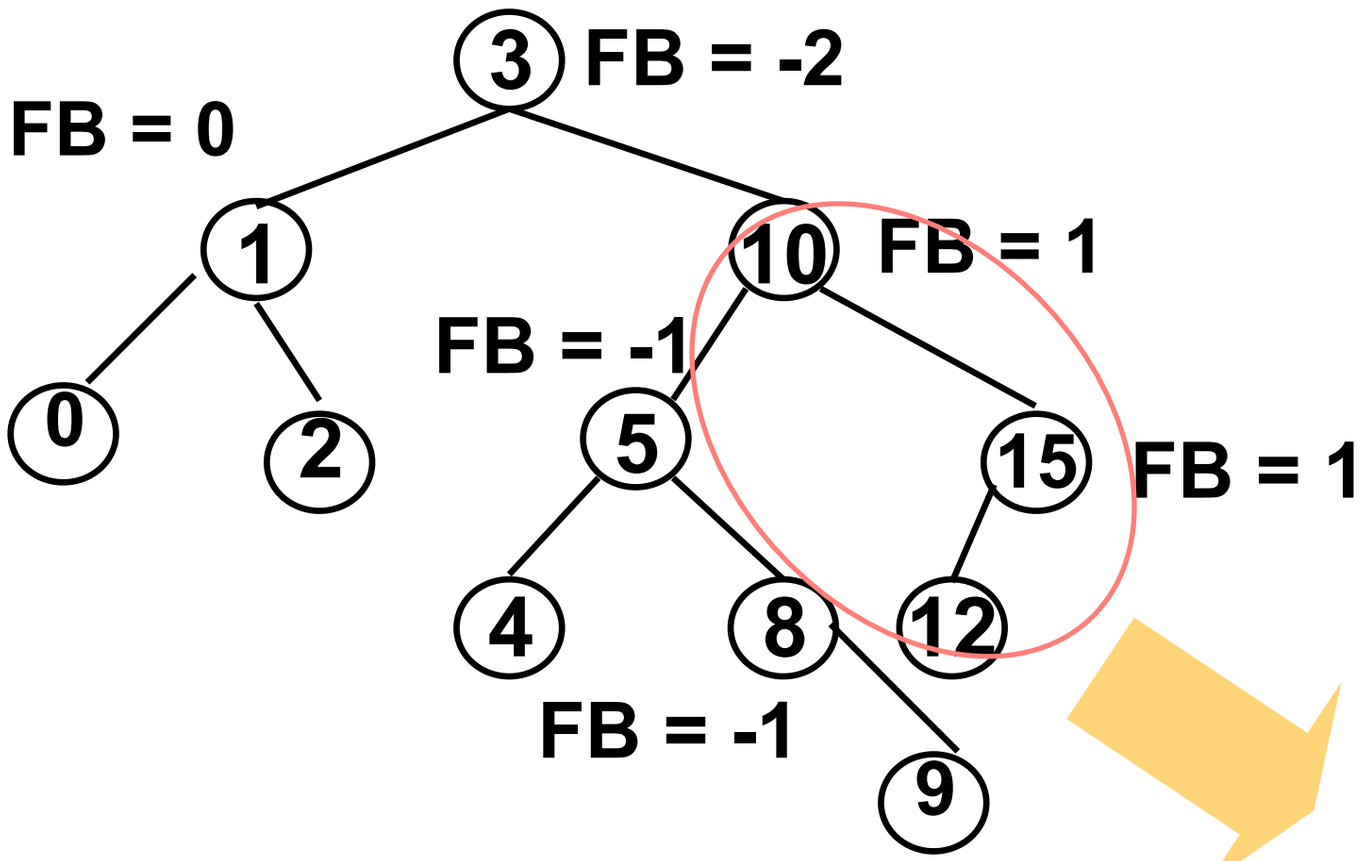


inserimento nel sotto albero <u>sinistro</u> del sotto albero <u>destro</u>

Inserimento di 9

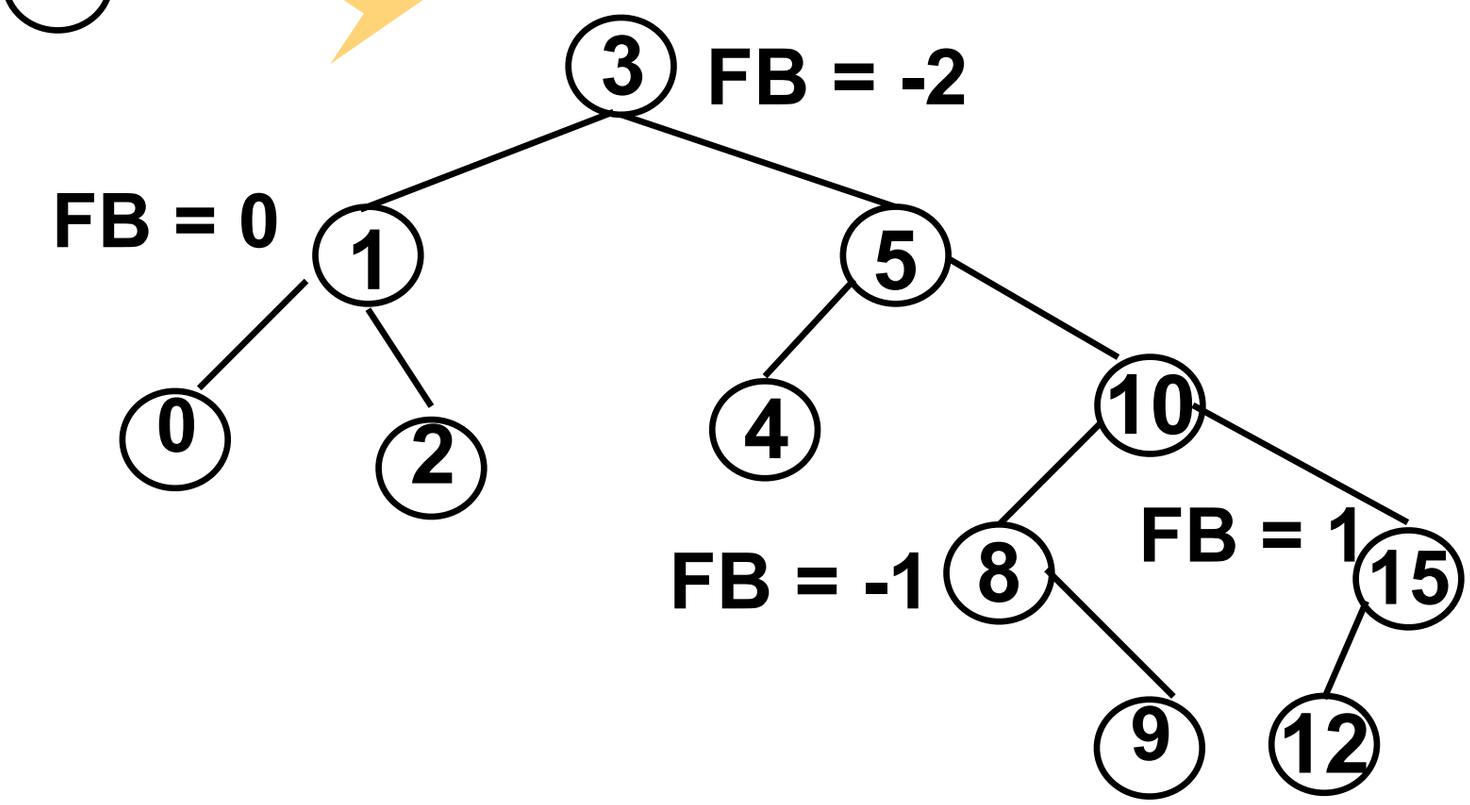


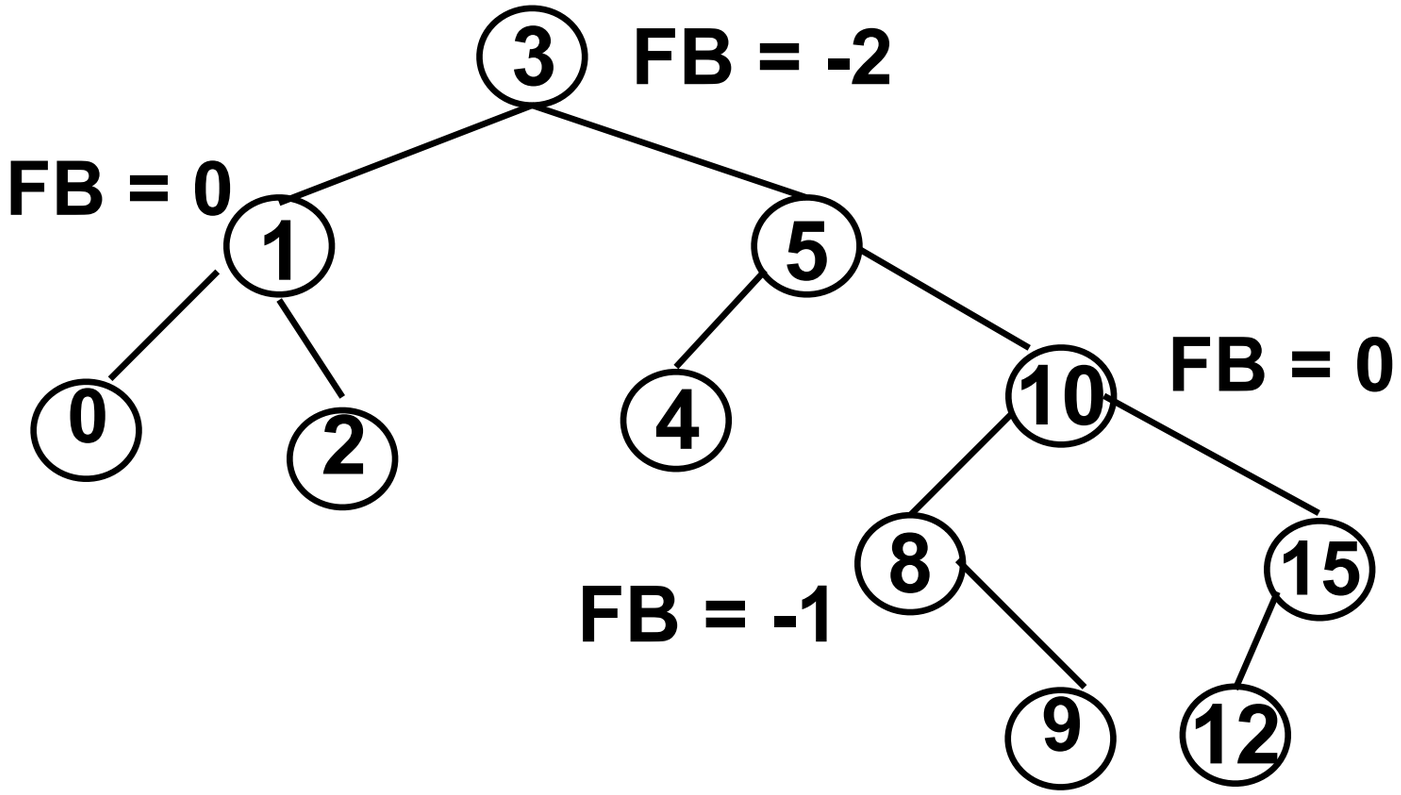
Risalendo da un nodo con FB = 1, passiamo a uno con FB = -2, questo identifica il caso: right-left



inserimento nel sotto albero <u>sinistro</u> del sotto albero <u>destro</u> del nodo sbilanciato

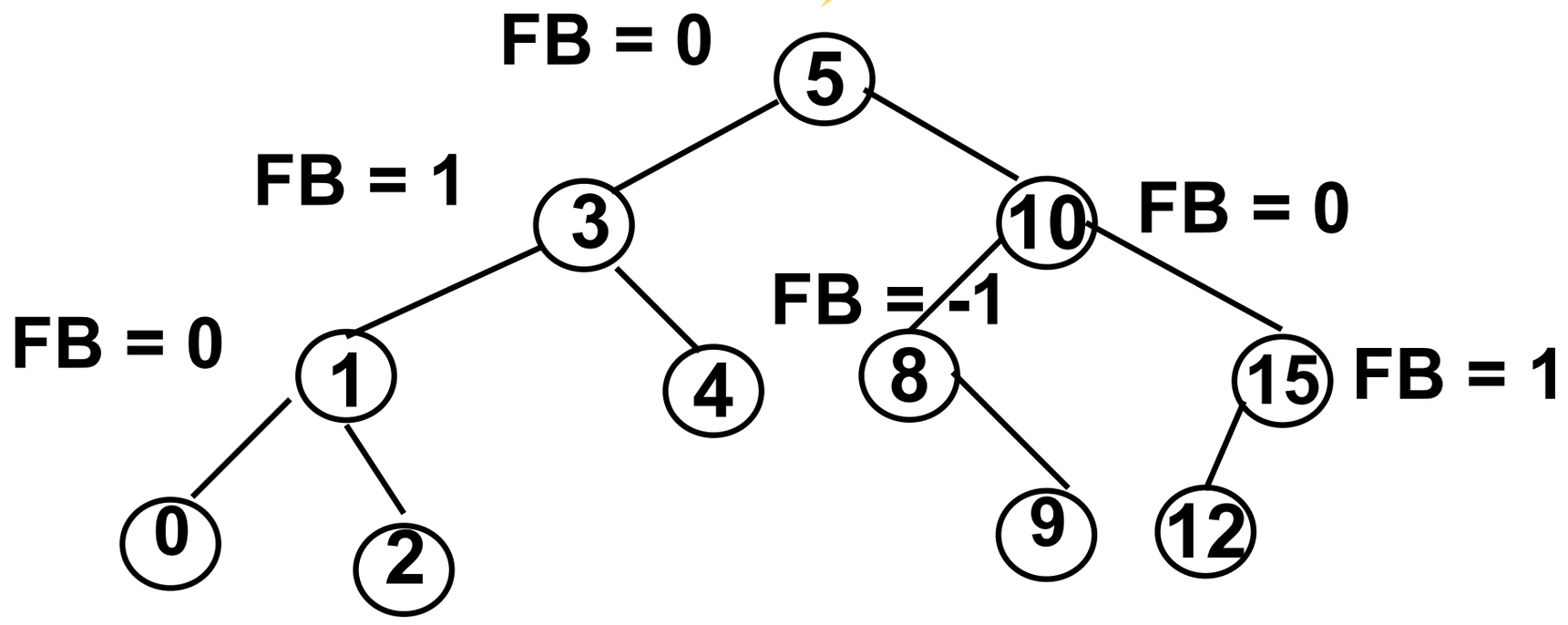
Prima rotazione: a destra su 10





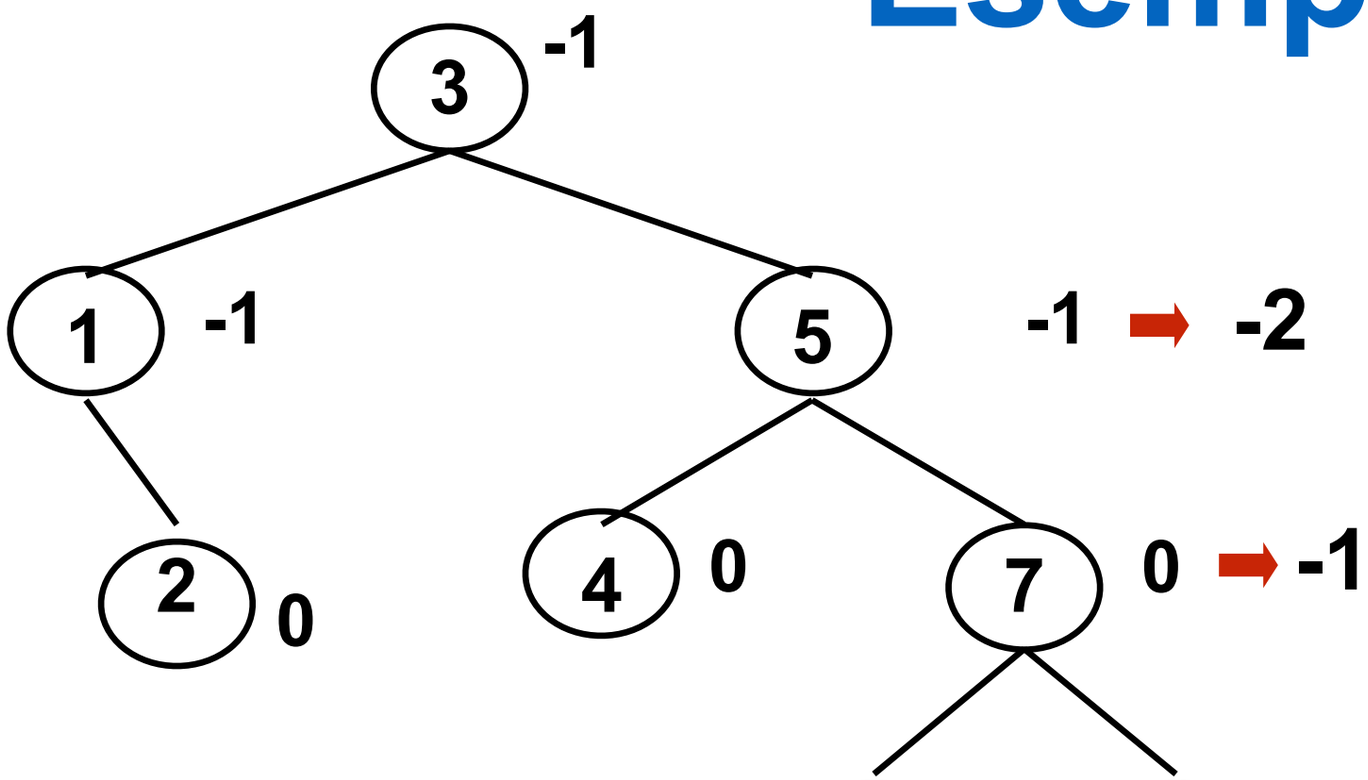
inserimento nel sotto albero <u>sinistro</u> del sotto albero <u>destro</u> del nodo sbilanciato

Seconda rotazione: a sinistra su 3

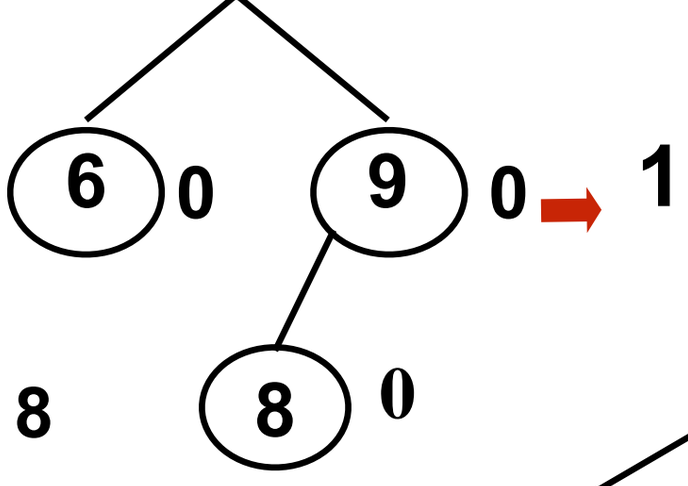


Esempio inserimento

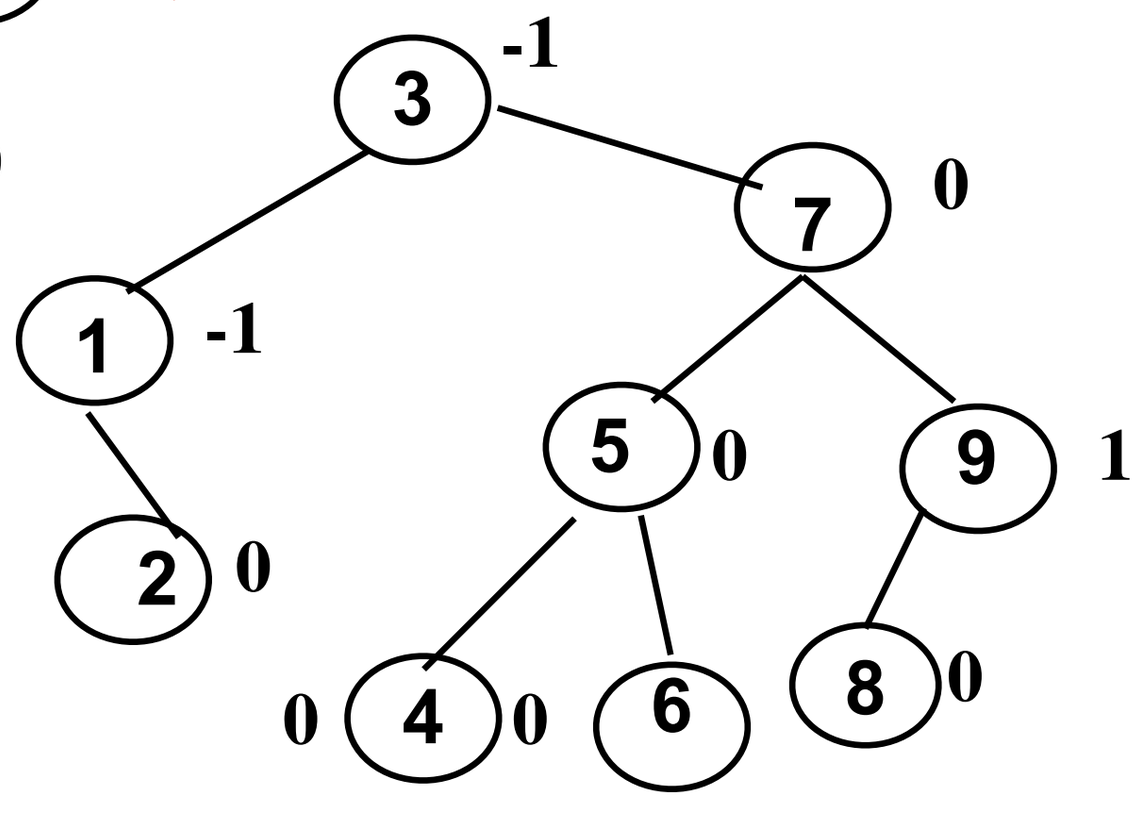
caso right-right



Rotazione a sinistra su 5

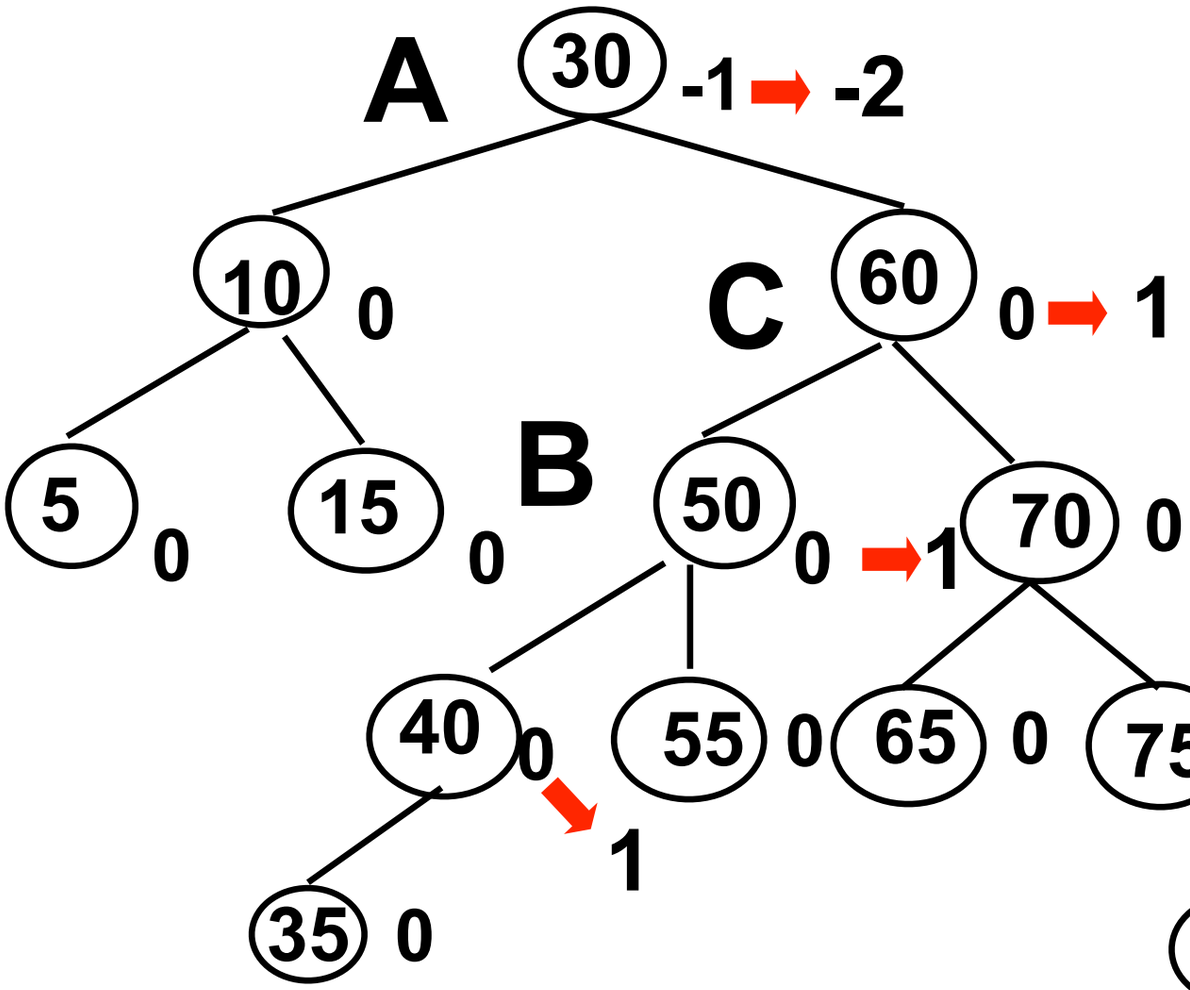


Inserimento del nodo 8



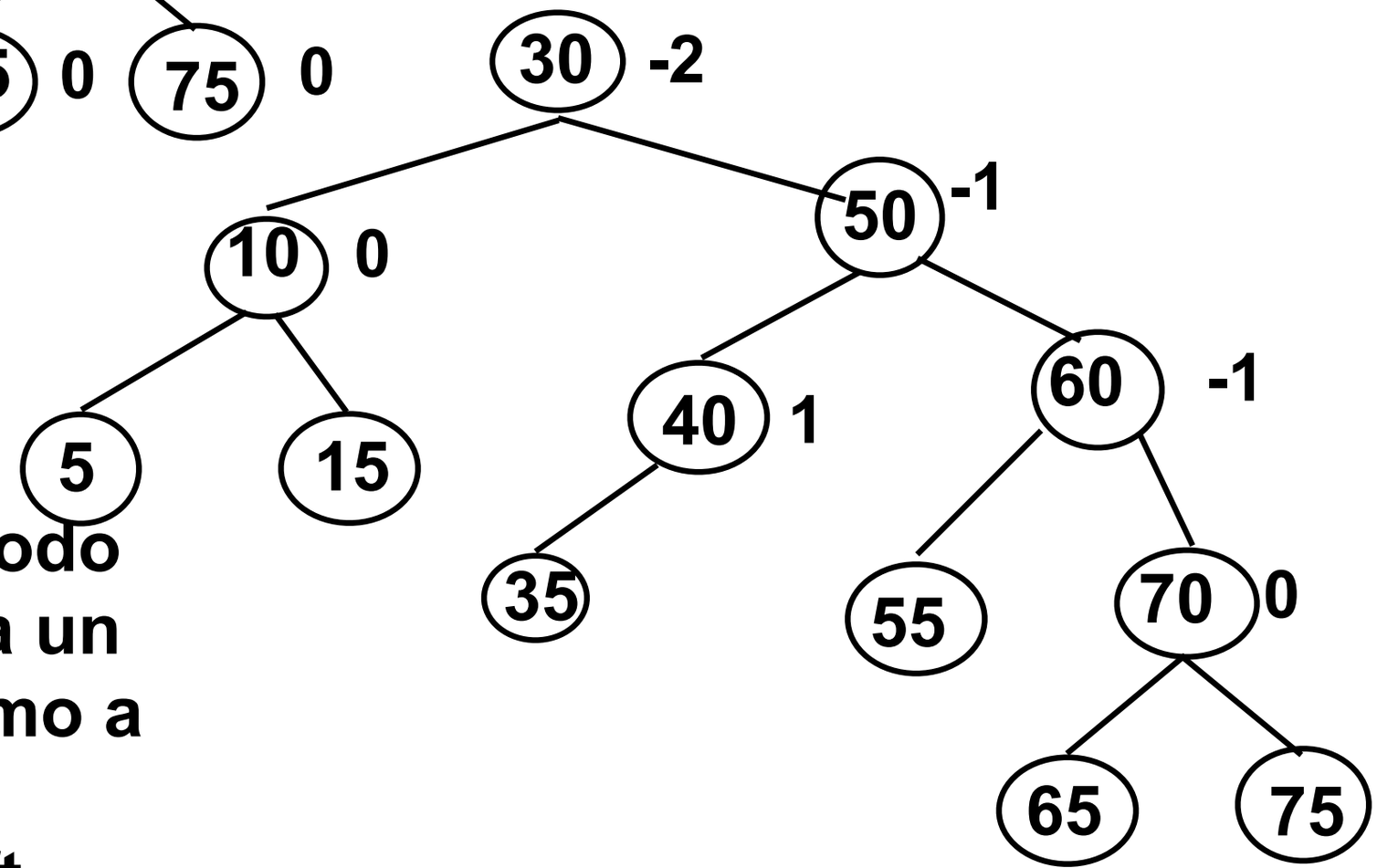
Risalendo da un nodo,B , con FB = -1, passiamo a uno con FB = -2, A, questo identifica il caso: right-right.

Esempio inserimento



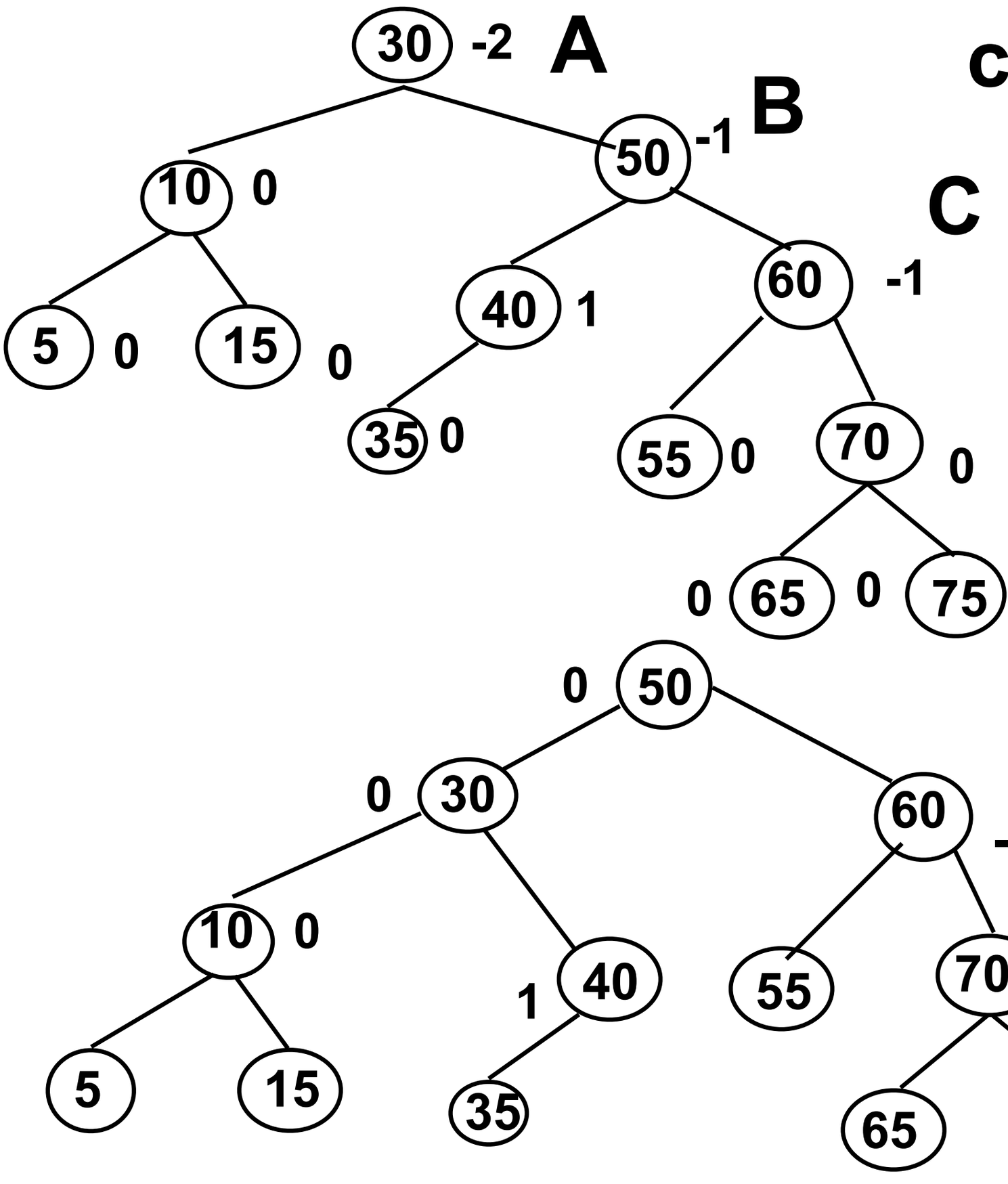
caso RIGHT- LEFT

Rotazione a destra su C



Inserimento del nodo 35, sotto albero sinistro del sotto albero destro del nodo sbilanciato. Risalendo da un nodo con FB = 1, passiamo a uno con FB = -2, questo identifica il caso right-left.

caso RIGHT- LEFT



Rotazione a sinistra su 30

Conclusione

Ci sono 4 casi:

1. **Due richiedono una sola rotazione:**
    1. **left-left: inserimento nel sotto albero sinistro del sotto albero sinistro, rotazione a destra, il caso è identificato dal fatto che da un nodo con FB = 1 saliamo a un nodo con FB = 2**
2. **right-right: inserimento nel sotto albero destro del sotto albero destro, rotazione a sinistra , il caso è identificato dal fatto che da un nodo con FB = -1 saliamo a un nodo con FB = -2**

Conclusione

2. **Due richiedono una rotazione doppia:**
    3. **left-right: inserimento nel sotto albero destro del sotto albero sinistro, rotazione sinistra seguita da una a destra, il caso è identificato dal fatto che da un nodo con FB = -1 saliamo a un nodo con FB = 2**
    4. **right-left: inserimento nel sotto albero sinistro del sotto albero destro, rotazione destra seguita da una a sinistra, il caso è identificato dal fatto che da un nodo con FB = 1 saliamo a un nodo con FB = - 2**

Tempo di esecuzione

L' inserimento da solo prende un tempo O(lg n).

Risalendo verso la radice si aggiornano i fattori di bilanciamento nei nodi e se un fattore diventa illegale (-2 o 2) con al più due rotazioni si ripristina la proprietà del bilanciamento in altezza.

Quindi anche le operazioni di ripristino delle proprietà di bilanciamento hanno un costo O(lg n).

Per un'implementazione in Python:
http://interactivepython.org/runestone/static/pythonds/Trees/AVLTreeImplementation.html