

PROPRIETA'

Sia T un AVL minimo di altezza h , per ogni nodo i due sottoalberi destro e sinistro sono AVL minimi di altezza $h-2$ e $h-1$

Dimostrazione

- Per assumere uno dei due sottoalberi non è un AVL minimo di altezza $h-1$.

Allora sicuramente esiste un AVL di altezza $h-1$ che ha meno nodi di questo. Sostituiamo allora questo AVL minimo al nostro sottoalbero AVL.

L'albero che abbiamo ottenuto ha sicuramente meno nodi del precedente, ma questo è un **ASSURDO** perché per ipotesi l'albero era un AVL minimo

- Per assumere l'altro sottoalbero non è un AVL minimo di altezza $h-2$.

Se non fosse minimo potremmo applicare lo stesso ragionamento di prima e riottenere l'**ASSURDO**

PROPRIETA'

$$\forall T \in \text{AVL} \quad h(T) \leq c \lg_2 m \quad m = |T| \quad (h(T) = O(\lg_2 m))$$

DIMOSTRAZIONE

Definiamo ogni numero di Fibonacci come:

$$F_{10}(h) = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^h - \left(\frac{1-\sqrt{5}}{2} \right)^h \right] \underset{\substack{h \rightarrow \infty \\ \rightarrow 0}}{\cong} \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^h$$

Quindi sappiamo che $mn(h) = F_{10}(h+3) - 1 \Rightarrow mn(h) + 1 \cong \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^{h+3}$

$$\sqrt{5} (mn(h) + 1) \cong \left(\frac{1+\sqrt{5}}{2} \right)^{h+3} \rightarrow h+3 \cong \lg_{\frac{1+\sqrt{5}}{2}} [\sqrt{5} (mn(h) + 1)] \rightarrow$$
$$\rightarrow h \cong \frac{\lg_2 \sqrt{5} (mn(h) + 1)}{\lg_2 \frac{1+\sqrt{5}}{2}} - 3$$

Consideriamo il denominatore come costante moltiplicativa, abbiamo espresso il numero di nodi con una funzione logaritmica per una costante

INSEGNAMENTO IN AVL

Abbiamo dimostrato che l'altezza degli AVL è limitata superiormente da $\lg_2 n$; vogliamo trovare un algoritmo per inserire i nodi mantenendo le proprietà degli AVL e che sia efficiente.

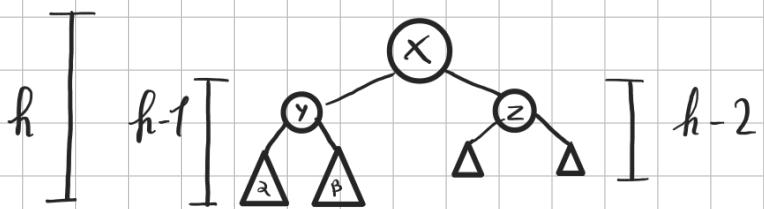
Per controllare le violazioni usiamo un algoritmo di ROTAZIONE

ROTAZIONE (DI UN AVL)

Intendiamo per **ROTAZIONE** (di un AVL in questo caso)

un algoritmo che ri-bilancia le altezze dei sottosalberi "ruotando" i nodi che modificando l'altezza violano la proprietà degli AVL

Immaginiamo di avere il seguente AVL



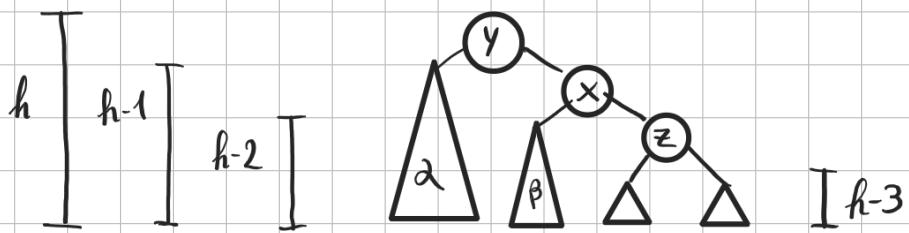
Se dobbiamo inserire un nodo sotto y l'altezza di quell'albero aumentera' di 1, facendo sì che x violi la condizione di AVL.

Cioè che possiamo fare e aumentare l'altezza dell'albero di z e diminuire quella dell'albero di y attraverso la **ROTAZIONE**

ROTAZIONE SINGOLA SINISTRA

Vogliamo diminuire di un livello il sottosalbero α (per tornare ad avere altezza $h-1$) per aumentare l'altezza del sottosalbero β .

Per mantenere la proprietà di ABR possiamo così ridistribuire:



l'albero, al costo di cambiare la radice e ridistribuire i sottoalberi, non solo rispetta le proprietà di AVL, bensì ora i due sottoalberi di y hanno stessa altezza.

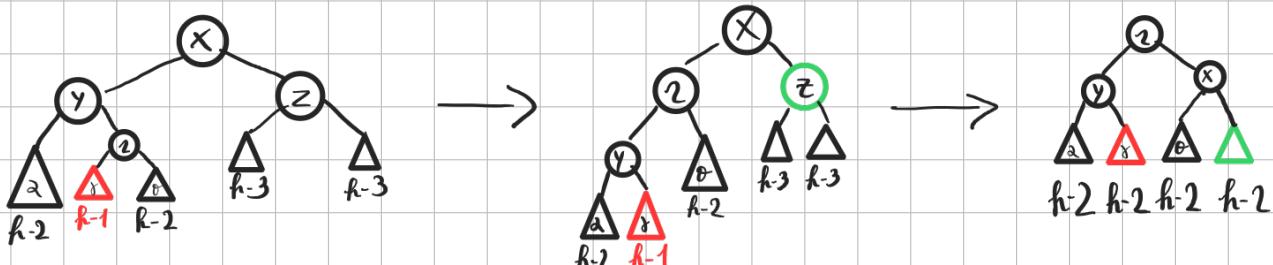
ROTAZIONE DOPPIA SINISTRA

Immaginiamo di dover aggiungere un nodo all'albero β (pre-rotazione)

In questo caso la rotazione-ssx non ci aiuta perché ruota il sottoalbero α senza modificare β che ci crea problemi.

Possiamo pensare però di "spostare il peso" di β ruotandolo a destra per poi ruotare a sinistra il nuovo albero ottenuto.

Sia γ la radice di β e suddividiamo β in $\gamma(sx)$ e $\theta(dx)$ di altezza $h-2$ e aggiungiamo un nodo ad esempio a γ (è indifferente)



ALGORITMO DI ROTAZIONE - SSX

Implementiamo nella struttura un campo che tiene conto del livello del nodo

ROTAZIONE_SSX (T)

$TEMP = T \rightarrow s_x$

$T \rightarrow s_x = TEMP \rightarrow d_x$

metti a sx di \times il d_x di y

$TEMP \rightarrow d_x = T$

settiamo la nuova radice

$T \rightarrow h = 1 + \text{MAX}(H(T \rightarrow s_x), H(T \rightarrow d_x))$

$TEMP \rightarrow h = 1 + \text{MAX}(H(TEMP \rightarrow s_x), H(TEMP \rightarrow d_x))$

RETURN TEMP

ALGORITMO DI ROTAZIONE - DSX

ROTAZIONE_DSX (T)

$T \rightarrow s_x = ROTAZIONE_SSX(T \rightarrow s_x)$

$T = ROTAZIONE_SSX(T)$

RETURN T

ALGORITMO DI INSERIMENTO IN AVL

● INSERT_AVL (T, K)

IF $T \neq \text{NIL}$ THEN

IF $T \rightarrow \text{Key} > K$ THEN

$T \rightarrow s_x = \text{INSERT_AVL}(T \rightarrow s_x, K)$

$T = \text{BILANCIA}_S X(T)$

ELSE IF $T \rightarrow \text{Key} < K$ THEN

$T \rightarrow d_x = \text{INSERT_AVL}(T \rightarrow d_x, K)$

$T = \text{BILANCIA}_D X(T)$

ELSE

$T = \text{AllocaNodo}()$

$T \rightarrow \text{Key} = K$

$T \rightarrow h = 0$

RETURN T

● BILANCIA_SX (T)

IF $(H(T \rightarrow s_x) - H(T \rightarrow d_x)) = 2$ THEN

IF $(H(T \rightarrow s_x \rightarrow s_x) > H(T \rightarrow s_x \rightarrow d_x))$ THEN

$T = \text{ROTAZIONE_SSX}(T)$

il sottobrno sx e' sempre
piu' alto
una violazione

ELSE

$T = \text{ROTAZIONE_DSX}(T)$

due violazioni

ELSE

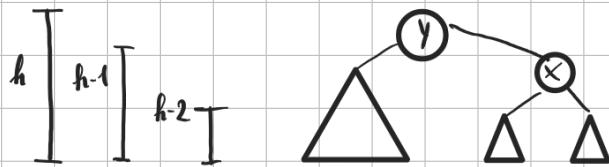
$T \rightarrow h = 1 + \max(H(T \rightarrow s_x), H(T \rightarrow d_x))$ Aggiorna $T \rightarrow h$

RETURN T

ROTAZIONI MASSIME RICHIESTE

In BILANCI_SX l'operazione più costosa è la rotazione, vogliamo quindi sapere al più quale ne saremo richieste.

Immaginiamo di avere una violazione in x che gestiamo ruotando a sinistra. Saremo in questa situazione



L'albero prima dell'inserimento aveva altezza h , dopo aveva altezza $h+1$ e dopo la rotazione di nuovo h .

Questo implica che il padre di x pre inserimento aveva altezza h , post rotazione si ritroverà un albero con altezza h .

Quindi dopo l'inserimento è necessaria **SOLO** una rotazione (costante)