Hashing

$h(K, i)$ $\qquad 0 \leq i < m$

$K \in U$

$h : U \times \{0, 1, 2, \dots, m-1\} \to \{0, 1, 2, \dots m-1\}$

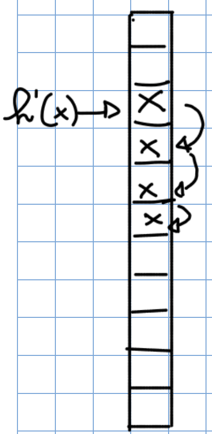
tecniche per fare l'hashing

Lineare

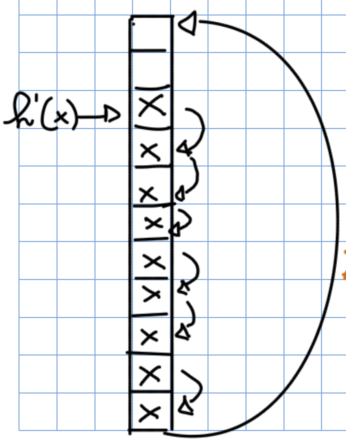
$h' : U \to \{0, 1, \dots m-1\}$

$h(K, i) = (h'(K) + i) \bmod (m)$ $\to$ $m!$ scansioni delle tabelle

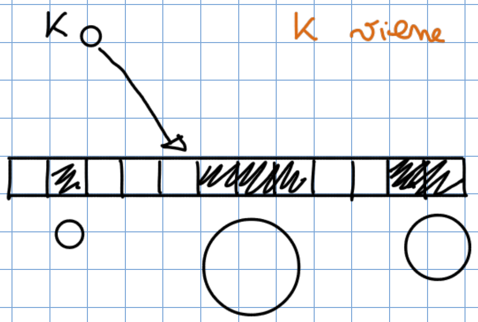
minore di m $\to$ numero di percorsi possibili



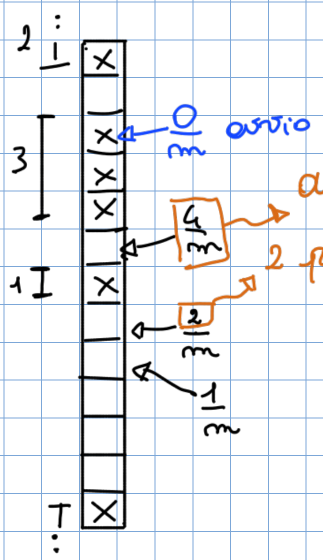
tutte le prove che faccio



qui entra in gioco il modulo



K viene attratto dalle celle con "massa più grande"



aumenta la probabilità perchè è un blocco grande

2 perchè deve finire in quella sopra e scende nella successiva

Questo fenomeno si chiama agglomerazione primaria

qui non c'è

Quadratica

$$h(k,i) = (h'(k) + c_1 + c_2 i^2) \bmod (m)$$

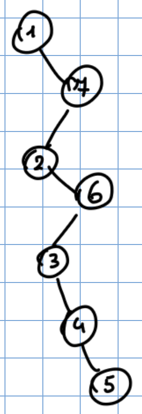
anche in questo caso abbiamo m scansioni

hashing doppio

due funzioni hash

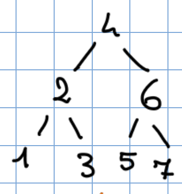
$$h(k,i) = (h'(k) + i\, h''(k)) \bmod m$$

in questo caso abbiamo m^2 scansioni



→ albero sbilanciato → caso peggiore da evitare

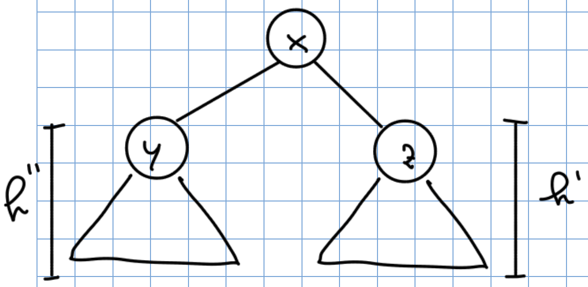
↓ è praticamente una lista



albero bilanciato

Vogliamo trovare un modo per rendere l'albero autobilanciante

Definizione "albero bilanciato":



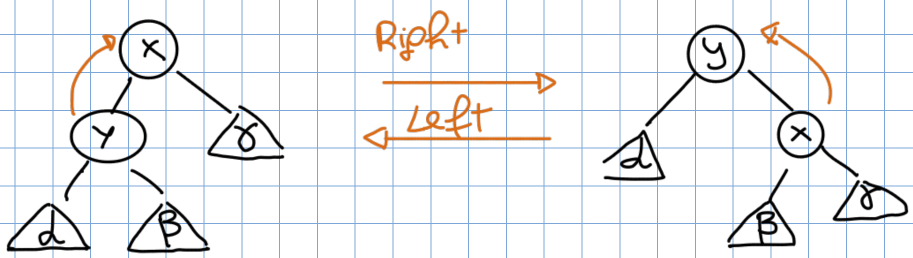
$|h' - h''| \le 2$ oppure $|\#(y) - \#(z)| \le \frac{n}{2}$

Basta che nei due sottoalberi ci sia asintoticamente lo stesso lavoro

Alberi Rosso - Neri (RB - TREE)

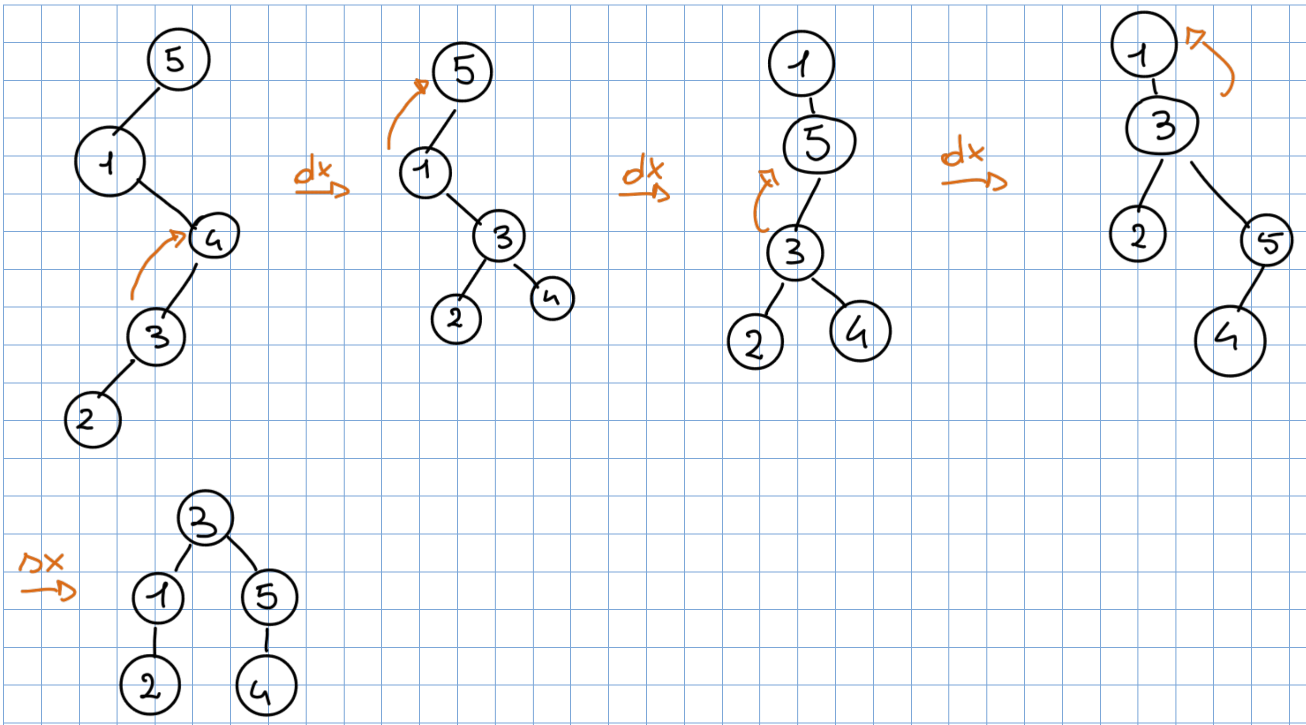
Per bilanciarsi fa uso delle Rotazione

△ → sotto albero

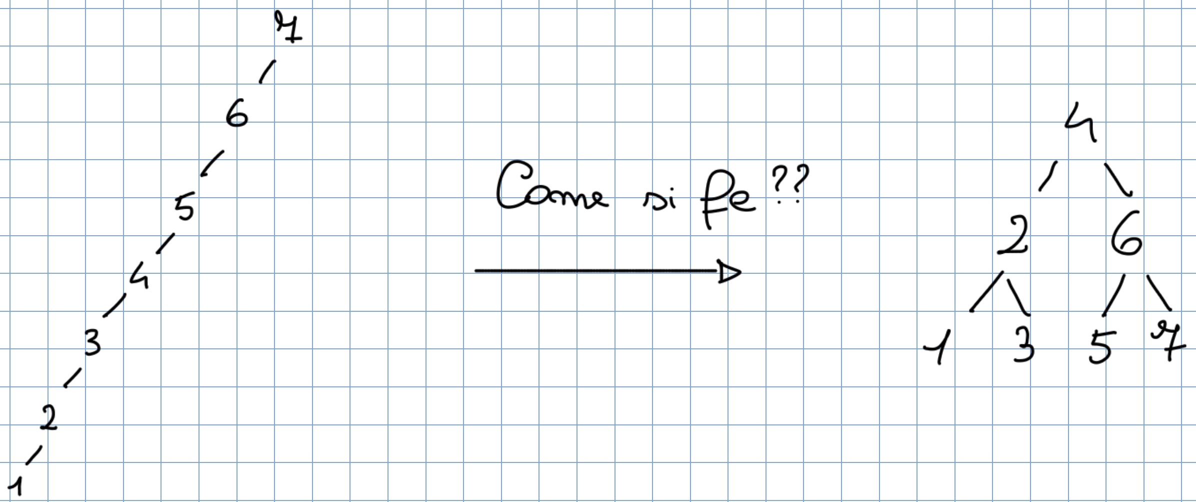


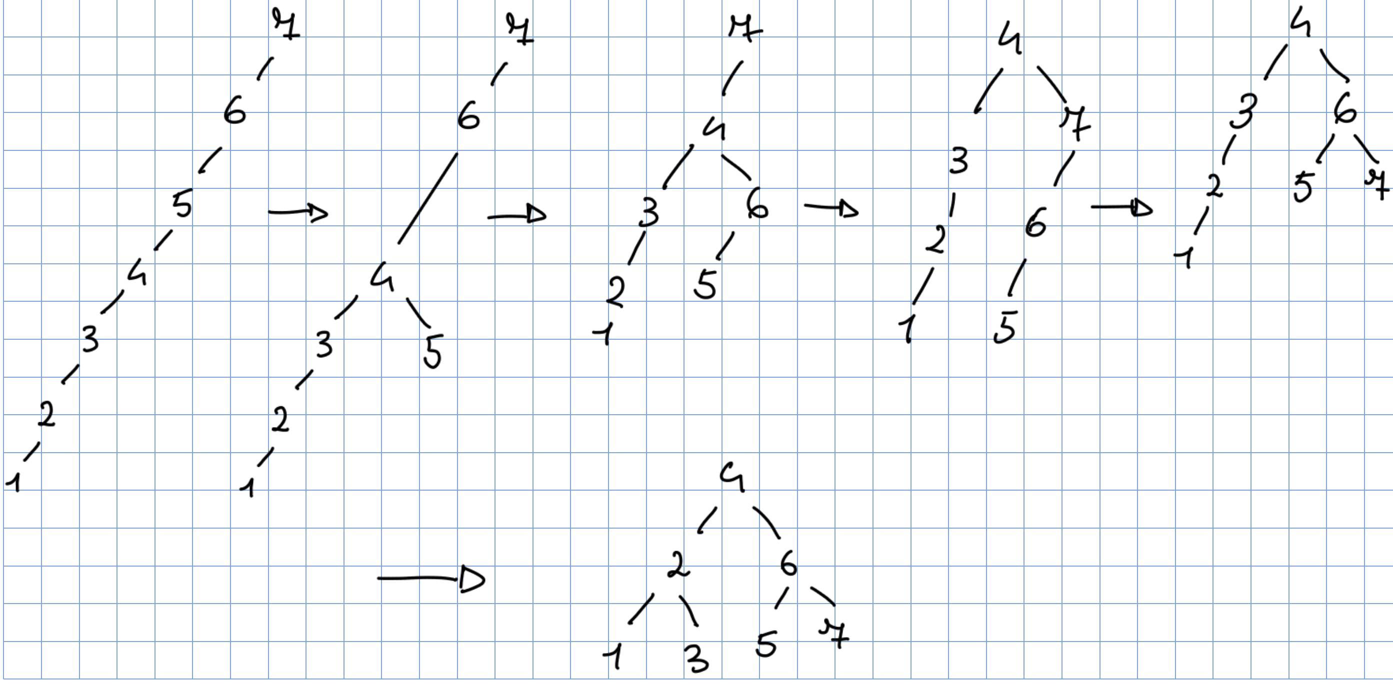
$\alpha \le y \le \beta \le x \le \gamma$ $\alpha \le y \le \beta \le x \le \gamma$

Abbiamo modificato la struttura dell'albero ma non il suo significato



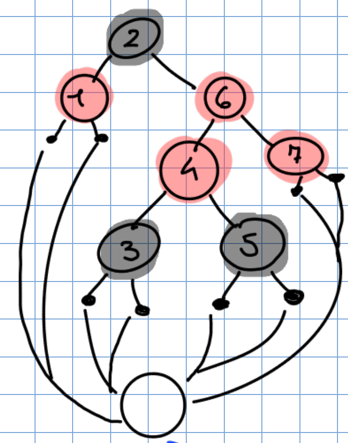
Abbiamo ridotto l' altezza senza modificarne il significato





Le 5 regole dei Rosso-Neri

= nero   = rosso



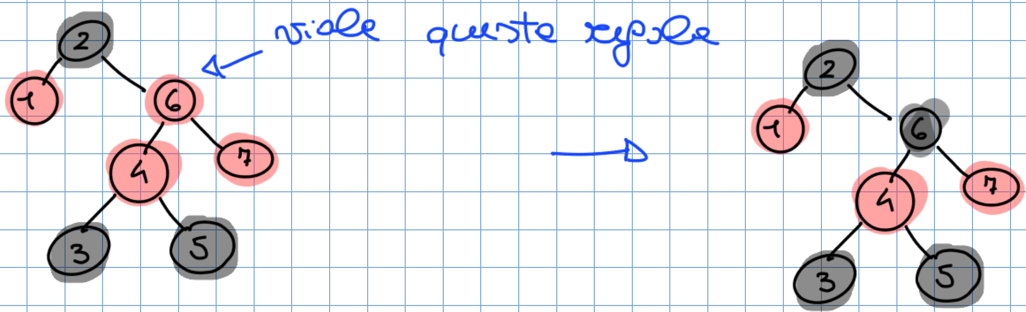
implementato puntano tutti ad un singolo nodo null

1. Ogni nodo è rosso o nero
2. La radice è nera
3. Le foglie sono nere

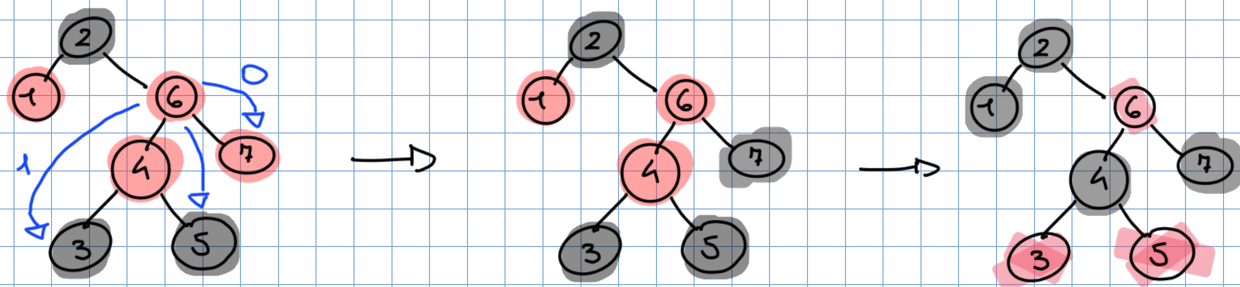
→ proprietà rilassabile (può anche non essere rispettata) se consideriamo le foglie sempre essenti nodi NULL facendo così però passo da un albero: $n \rightarrow 2n+1$

non cambia il comportamento asintotico

4. Un nodo rosso ha figli neri

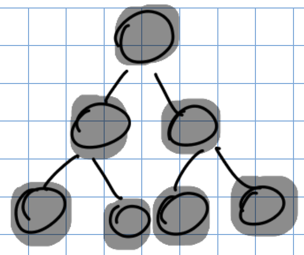


5. Un qualsiasi cammino da un nodo ad una foglia incontra lo stesso numero di nodi neri

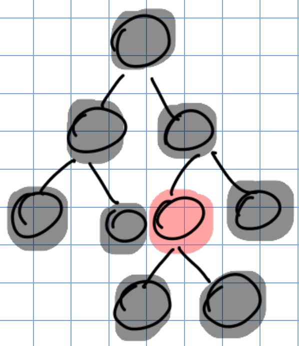


Non esiste un solo modo di colorare un albero ma diversi, basta che proprietà vengono rispettate

Queste proprietà possono essere rispettate solo su un albero bilanciato



→ rispetta tutte le proprietà

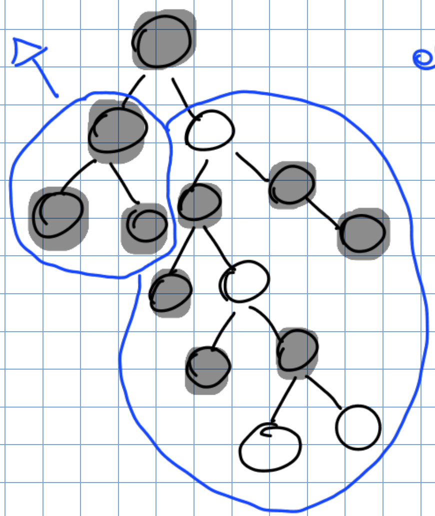


→ rispetta le proprietà anche aggiungendo un rosso

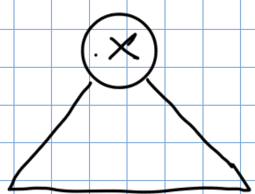
$bh(x)$ → altezza nera

altezza minima : K

altezza massima : 2K + 1



Più di così non posso sbilanciare



$$I(x) \geq 2^{bh(x)} - 1$$

$h(x)$ → altezza

$bh(x)$ → altezza nera

$\} \; h(x) \geq bh(x)$

ma non troppo infatti

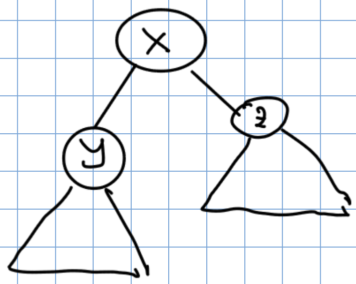
$$bh(x) \geq \frac{h(x)}{2}$$

$h(x) = 0$

$2^0 - 1 = 1 - 1 = 0$

la proprietà vale

$h(x) > 0$



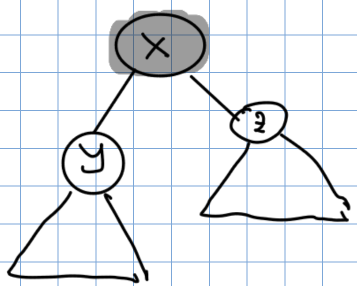
numero nodi interni

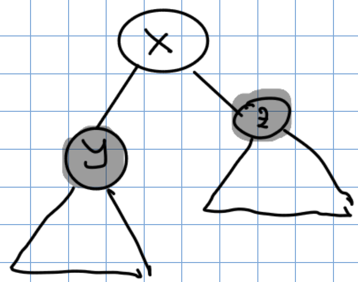
$I(y) \geq 2^{bh(y)} - 1$

$I(z) \geq 2^{bh(z)} - 1$

$h(y) < h(x)$

$h(z) < h(x)$





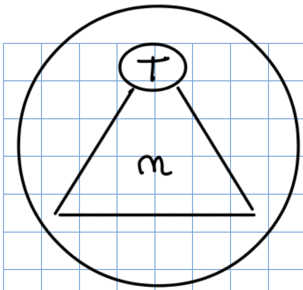
$$\underbrace{bh(y) = bh(x) \qquad\qquad bh(y) = bh(x) - 1}$$

$bh(y) \geq bh(x) - 1$

$bh(z) \geq bh(x) - 1$

$$\begin{aligned} I(x) &= I(y) + I(z) + 1 \\ &\geq 2^{bh(y)} - 1 + 2^{bh(x)} \cancel{-1} \cancel{+1} \\ &\geq 2^{bh(x)-1} - 1 + 2^{bh(x)-1} \\ &= 2^{bh(x)} - 1 \end{aligned}$$

allora $I(x) = 2^{bh(x)-1}$

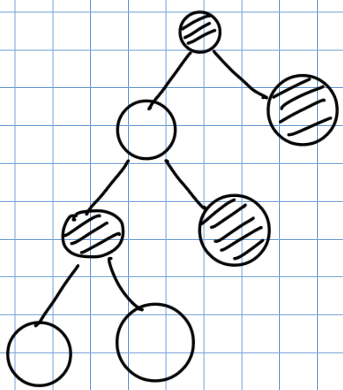


$$I(t) = n \geq 2^{bh(t)} - 1$$

$$n+1 \geq 2^{\frac{h(t)}{2}}$$

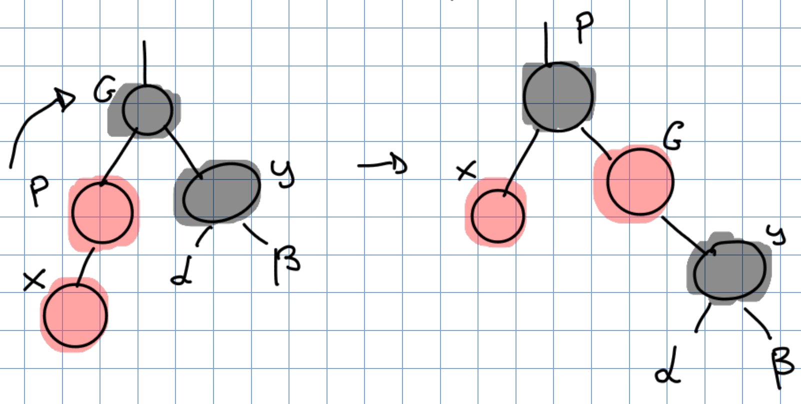
$$\log n+1 \geq \frac{h(t)}{2} \Rightarrow h(t) \leq 2\log(n+1)$$

Inserimento in un albero rosso-nero

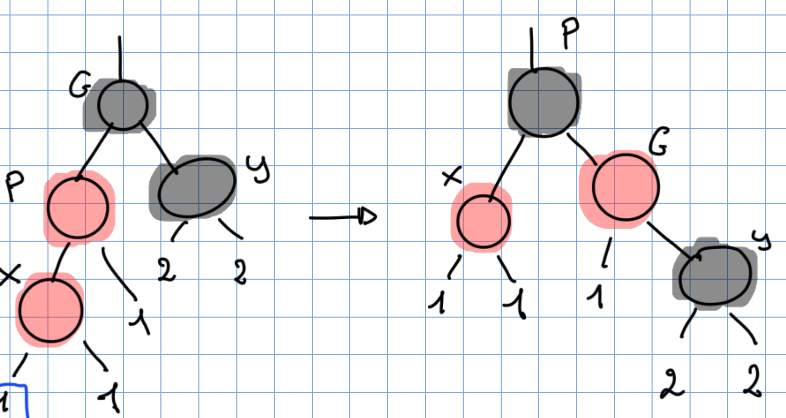


I nodi inseriti sono sempre rossi, così abbiamo meno problemi

CASO 1 y nero, x esterno

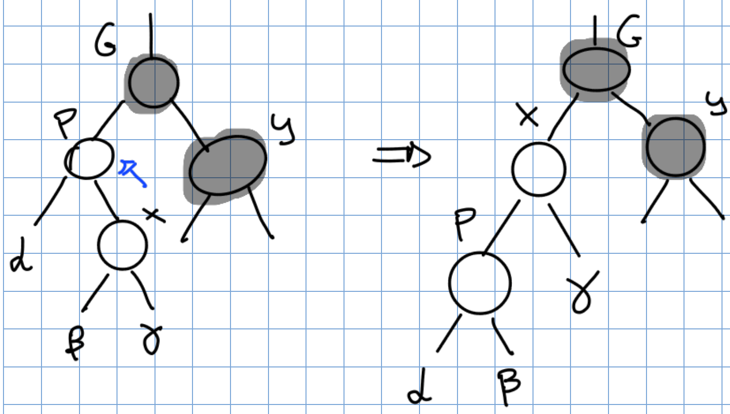


l' albero resta invariato



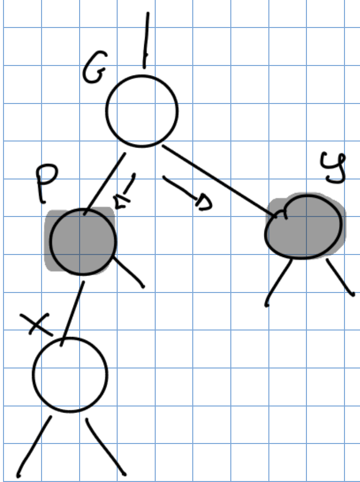
$\boxed{1}$ → numero nodi neri nel cammino

CASO 2 y nero, x interno



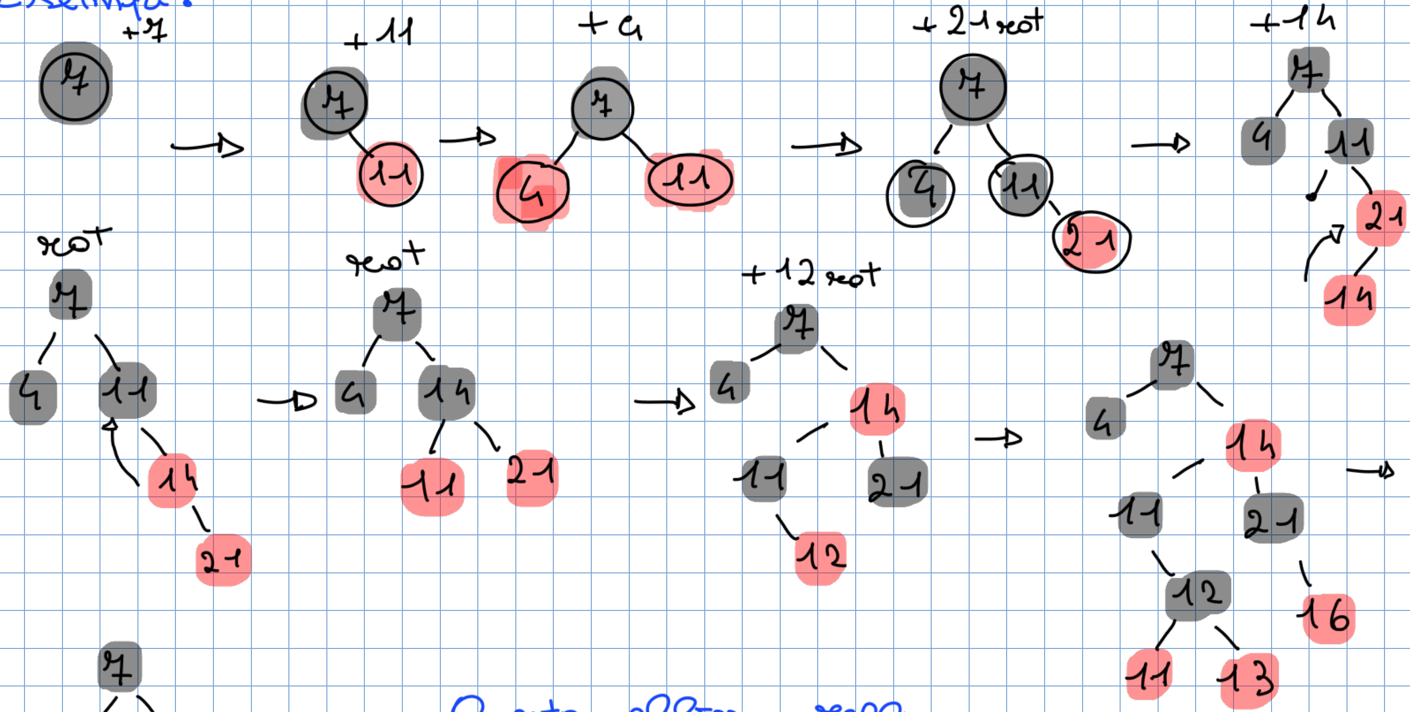
siamo riusciti ad arrivare ad un caso conosciuto

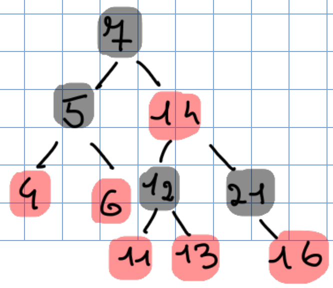
CASO 3 y rosso



C'è qualche problema nel disegno guardare meglio il libro

Esempi:





Questo albero regge lo stress del caso peggiore iniziale?

$bh(t) = 0$ $bh(t) = 1$