RB-Alberi

I vantaggi degli ABR sono evidenti su alberi non troppo sbilanciati

Un RB-albero è un albero binario che mantiene un buon bilanciamento

Struttura del nodo

color colore del nodo: rosso o nero

key chiave

left puntatore al figlio destro

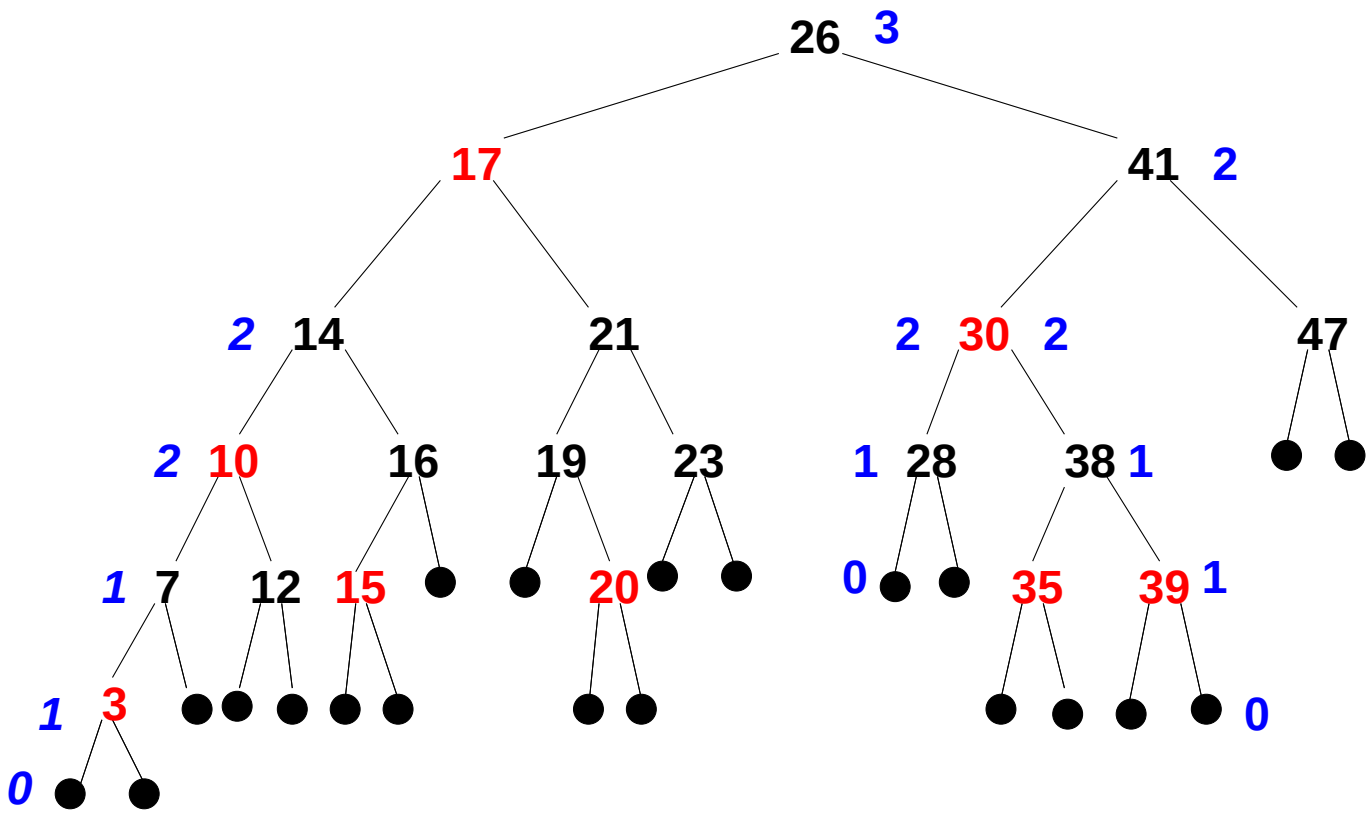
right puntatore al figlio sinistro

p puntatore al padre (NIL per la radice)

Proprietà

- Ad ogni nodo è associato il colore rosso o nero
- Le foglie sono tutte nere
- Un nodo rosso ha entrambi i figli neri
- Ogni cammino, da un nodo ad una sua qualunque foglia contiene lo stesso numero di nodi neri

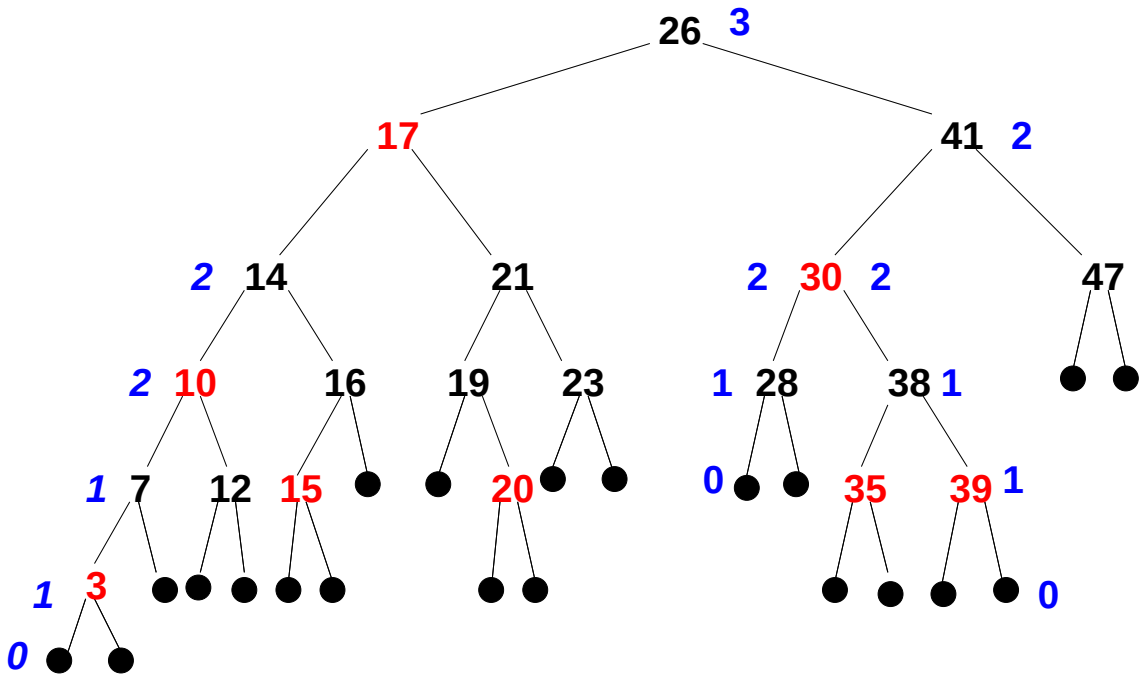
Esempio



La b-altezza di un nodo x è il numero di nodi neri (x escluso) nel cammino da x ad una sua qualunque foglia

La b-altezza dell'albero è l'altezza della sua radice

Proprietà di bilanciamento



- poiché un nodo rosso può avere solo figli neri al massimo c'è una alternanza di nodi rossi e neri
- poiché ogni cammino da un nodo ad una foglia contiene lo stesso numero di nodi neri
- poiché la b-altezza dell'albero è data dal solo numero di nodi neri presenti nel cammino dalla radice ad una foglia (esempio $bh(21) = 2$)

se h è la profondità della foglia meno profonda, la foglia più profonda si trova al massimo a profondità 2h

Altezza di un RB-albero

In termini del numero di nodi dell'albero

Definizione Un nodo interno è un nodo non foglia.

Osservazione Se un nodo x ha altezza h i suoi figli hanno altezza $h-1$, un nodo foglia ha altezza 0.

Lemma Un sottoalbero con radice in un qualsiasi nodo x contiene almeno $2^{bh(x)}-1$ nodi interni.

Si dimostra per induzione sull'altezza dell'albero

Base $h(x)=0$, $x=NIL$ è una foglia e il sottoalbero con radice in x ha $bh(x)=0$ quindi ha $2^0-1=0$ nodi.

Induzione Sia x un nodo interno, quindi con 2 figli.

Sia h l'altezza del sottoalbero con radice in x e con b-altezza $bh(x)$

I suoi figli destro e sinistro hanno altezza $h-1$ e b-altezza $bh(x)$ se il nodo è rosso oppure $bh(x)-1$ se il nodo è nero (perchè la b-altezza non considera il nodo in questione).

Applichiamo l'ipotesi induttiva considerando che i due sottolaberi di altezza $h-1$ abbiano b-altezza più piccola $=bh(x)-1$.

Il numero di nodi dell'albero radicato in x di altezza h è

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

LEMMA Un RB-albero con n nodi ha al più altezza $2lg(n+1)$.
DIMOSTRAZIONE Sia h l'altezza dell'albero con n nodi.

Consideriamo un cammino dalla radice ad una foglia. Nel cammino ci possono essere tutti nodi neri oppure almeno la metà di nodi neri (due rossi consecutivi mai), di conseguenza la b-altezza deve essere almeno $h/2$ (non può essere meno di $h/2$).

Per il LEMMA precedente l'albero ha un numero di nodi n uguali almeno a $2^{h/2} - 1$

$$n \geq 2^{h/2} - 1$$
$$n + 1 \geq 2^{h/2}$$

Applicando il logaritmo

$$lg(n+1) \geq lg2^{h/2}$$
$$lg(n+1) \geq \frac{h}{2}$$
$$2lg(n+1) \geq h$$

$$h \leq 2lg(n+1)$$

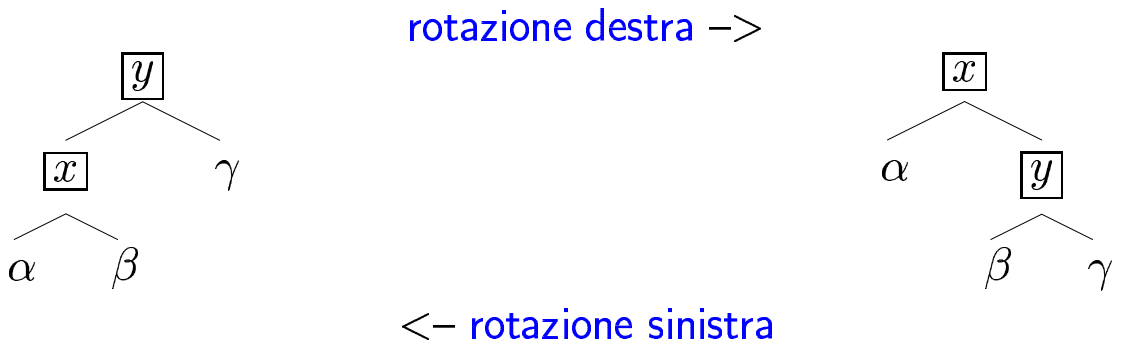
L'altezza di un RB-albero con n nodi è proporzionale al $lg\ n$

Mantenimento di un RB-albero: Rotazioni

Le operazioni Tree-Insert e Tree-Delete modificano la struttura dell'RB-albero, che deve essere ripristinata, anche cambiando eventualmente colore ai nodi

Una rotazione deve:

- Mantenere la proprietà ABR



rotazione destra

- y è maggiore di tutti gli elementi di sinistra quindi deve trovarsi nella parte destra
- tutti gli elementi di β sono maggiori di x ma minori di y quindi β diventa sottoalbero sinistro di y
- x diventa la radice dell'albero

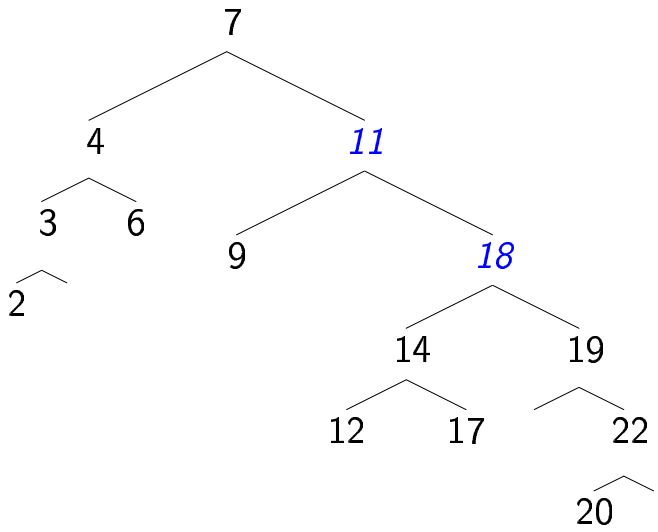
rotazione sinistra è speculare

Le rotazioni non modificano la proprietà degli ABR

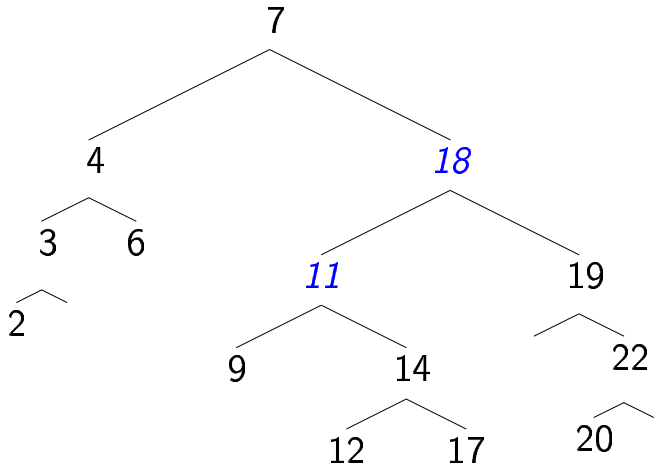
Mantenimento dell'ordinamento delle chiavi

Le rotazioni non modificano l'ordine delle chiavi dell'albero rispetto alla visita simmetrica

Esempio: Rotazione Sinistra



Per diminuire l'altezza dell'ABR viene effettuata la rotazione rispetto ai nodi 11 *e* 18



Codice per la rotazione

```
LEFT-ROTATE(T,x)
1.      y <- right[x]
        ;; appoggio il s.a. destro di x in y
2.      right[x] <- left[y]
        ;; ruoto il s.a. sinistro di y nel s.a. destro di x
3.      if left[y] != NIL
4.          then p[left[y]] <- x
5.      p[y] <- p[x]
6.      if p[x] = NIL
7.          then root[T] <- y
8.          else if x = left[p[x]]
9.                  then left[p[x]] <- y
10.                 else right[p[x]] <- y
11.      left[y] <- x
12.      p[x] <- y
```

Il tempo d'esecuzione di LEFT-ROTATE è costante

Inserzione

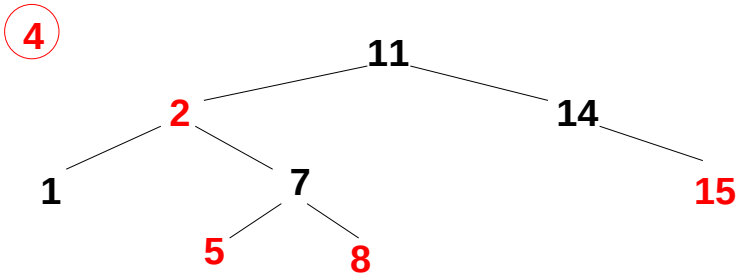
Quando viene inserito un elemento viene effettuata la rotazione dell'albero e l'eventuale ri-colorazione dei nodi

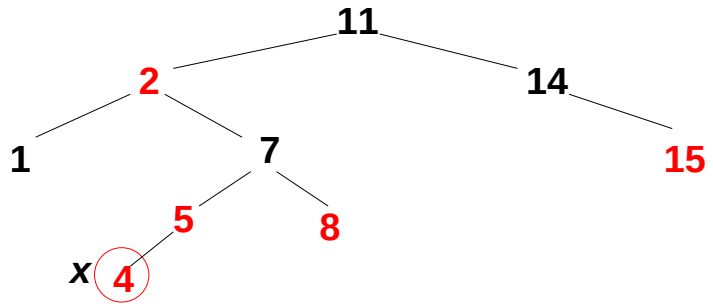
Algoritmo: operazioni principali

1. Si inserisce un nodo come in un ABR
2. Si colora il nodo di rosso
3. Si ricolorano i nodi del padre e dei suoi sottoalberi in modo che venga soddisfatta la proprietà degli RB-alberi (eventualmente eseguendo una rotazione)
4. Si prende in esame il colore del nodo padre si torna al punto 3.

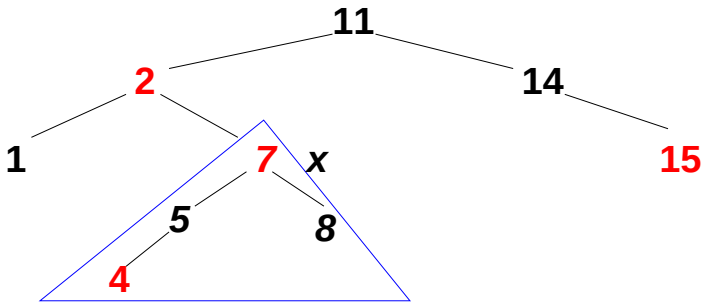
Il ciclo termina quando il nodo in esame è la radice, oppure quando il nodo padre di x ha colore nero (quindi la proprietà sul colore dei nodi non viene violata)

Inserzione - Esempio

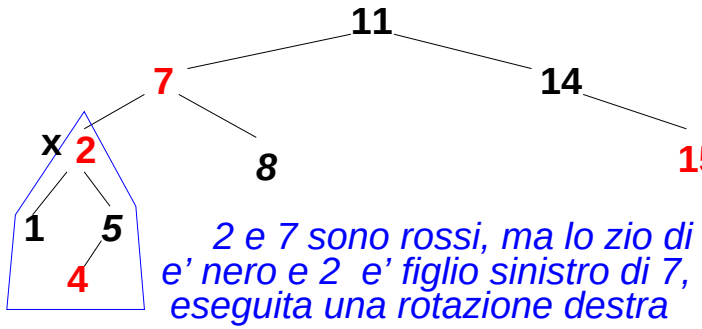




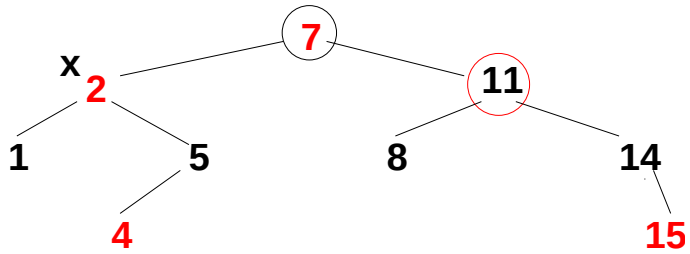
I nodi 5 e 4 violano la proprieta' sui colori Poiche' anche lo zio di 4, 8 e' rosso, 5 e 8 diventano neri e 7 rosso



Si verifica se il nodo 7 soddisfa la proprieta' sui colori: NO Poiche' lo zio di 7, 14, e' nero e poiche' 7 e' figlio destro di 2, viene eseguita una rotazione sinistra



2 e 7 sono rossi, ma lo zio di 2, 14 e' nero e 2 e' figlio sinistro di 7, viene eseguita una rotazione destra

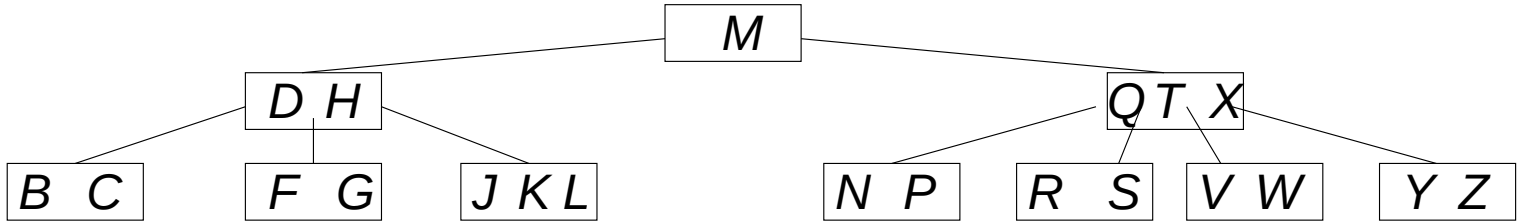


Infine si scambiano i colori di 7 e 11

B-Alberi

Struttura dati progettata per memoria secondaria ad accesso diretto

- Alberi n-ari bilanciati di ricerca
- un nodo di un b-albero contiene un più di una chiave (punti di divisione)
- se un nodo x ha n chiavi allora ha n+1 figli



Con questa struttura si riducono il numero di accessi alla memoria

ESEMPIO

Un b-albero di altezza 2 con 1000 chiavi per nodo può contenere più di un miliardo di chiavi.

Sono necessari al più due accessi al disco per trovare una chiave (la radice deve sempre trovarsi in memoria principale)