UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Corso di Introduzione agli algoritmi
Prof.ssa Tiziana Calamoneri

Dizionari: Alberi Rosso-Neri

Un ABR di altezza *h* contenente *n* nodi supporta le operazioni fondamentali dei dizionari in tempo *O(h)*, ma purtroppo non si può escludere che *h* sia *O(n)*, con conseguente degrado delle prestazioni.

Viceversa, tutte le operazioni restano efficienti se si riesce a garantire che l’altezza dell’albero sia piccola, in particolare sia *O(log n)*.

Per raggiungere tale scopo esistono varie tecniche, dette di ***bilanciamento***.

Bilanciamento dell’altezza (2)

Le tecniche di ***bilanciamento*** sono tutte basate sull’idea di riorganizzare la struttura dell’albero se essa, a seguito di un’operazione di inserimento o di eliminazione di un nodo, viola determinati requisiti.

In particolare, il requisito da controllare è che, per ciascun nodo dell’albero, l’altezza dei suoi due sottoalberi non sia “troppo differente”.

Ciò che rende non banali queste tecniche è che si vuole aggiungere agli ABR una proprietà (il bilanciamento) senza peggiorare il costo computazionale delle operazioni.

In letteratura vi sono vari approcci. Noi ne descriveremo uno.

Alberi Rosso-Neri (1)

Un albero rosso-nero (RB-albero) è un ABR i cui nodi hanno un campo aggiuntivo, il ***colore***, che può essere solo rosso o nero.

Alla struttura si aggiungono foglie fittizie (che non contengono chiavi) per fare in modo che tutti i nodi "veri" dell'albero abbiano ***esattamente due figli***.

Un RB-albero è un ABR che soddisfa le seguenti proprietà aggiuntive:

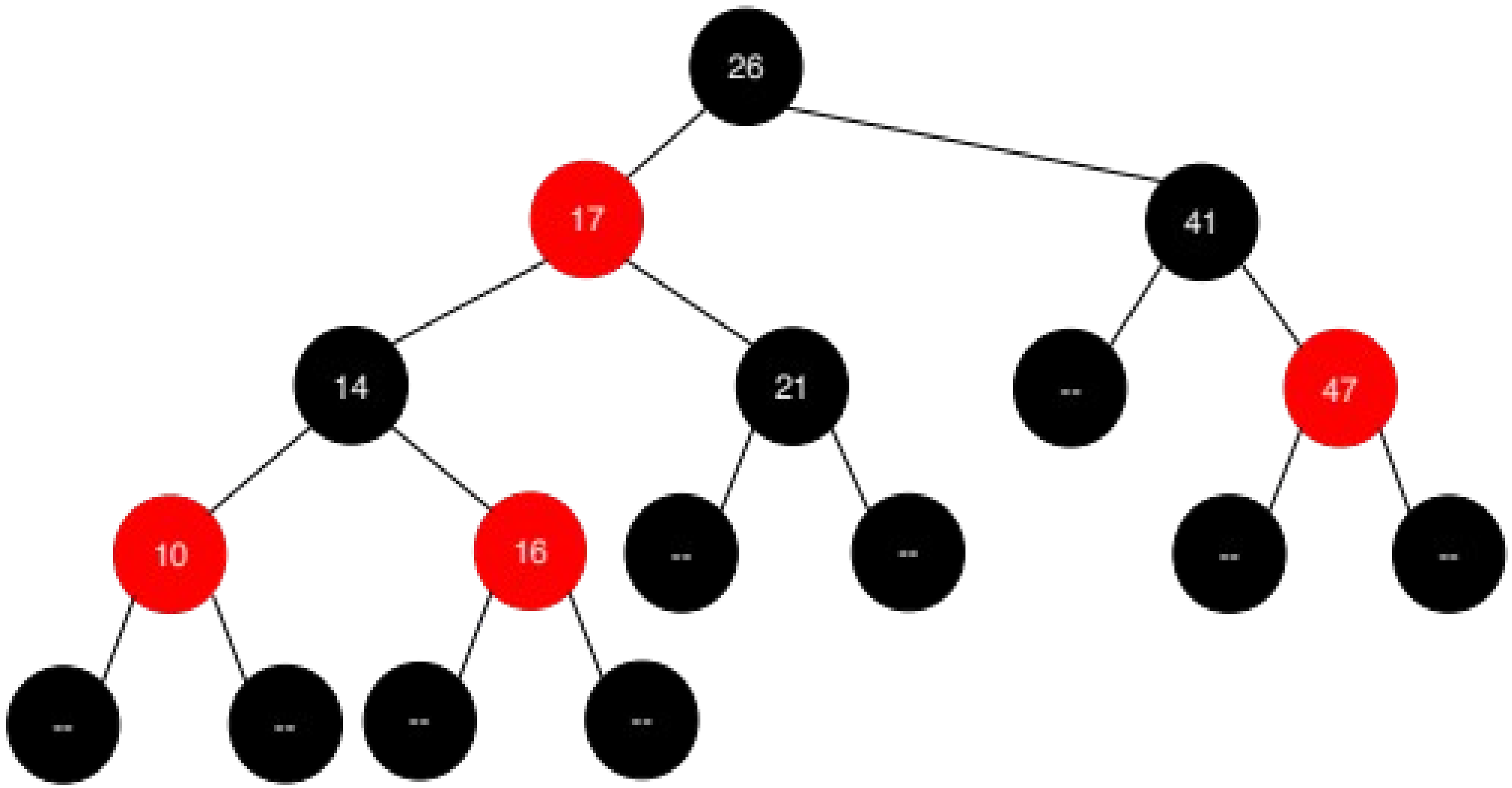
1. ciascun nodo è rosso o nero;
2. ciascuna foglia fittizia è nera;
3. se un nodo è rosso i suoi figli sono entrambi neri;
4. ogni cammino da un nodo a ciascuna delle foglie del suo sottoalbero contiene lo stesso numero di nodi neri.

Alberi Rosso-Neri (2)

La ***b-altezza*** di un nodo ***x*** (***black height*** di *x*, ***bh(x)***) è il numero di nodi neri sui cammini dal nodo x (non incluso) alle foglie sue discendenti (è uguale per tutti i cammini, proprietà 4).

La *b-altezza* di un RB-albero è la b-altezza della sua radice.

Alberi Rosso-Neri (3)



Alberi Rosso-Neri (4)

Proprietà: nessun cammino dalla radice ad una foglia può essere lungo più del doppio di un cammino dalla radice ad una qualunque altra foglia.

Dim.

- il numero di nodi neri è il medesimo lungo tutti i cammini dalla radice ad una qualunque foglia (proprietà 4);
- il numero di nodi rossi lungo un cammino non può essere maggiore del numero di nodi neri lungo lo stesso cammino (proprietà 3);
- un cammino che contiene il massimo numero possibile di nodi rossi non può essere lungo più del doppio di un cammino composto solo di nodi neri. CVD

Alberi Rosso-Neri (5)

Lemma. Il sottoalbero radicato in un qualsiasi nodo *x* contiene almeno $2^{bh(x)}-1$ nodi interni.

Dim. Ragioniamo per induzione sulla distanza di *x* dalle foglie, sia essa *d(x)*.

Se $d(x)=0$, allora *x* è una foglia ed il suo sottoalbero contiene almeno $2^{bh(x)}-1=2^0-1=0$ nodi interni.

Se *x* è un nodo interno con $d(x)>0$. I suoi figli hanno b-altezza pari o a *bh(x)* o a *bh(x)-1*, a seconda che *x* sia rosso o nero. Hp. induttiva sui figli, il sottoalbero di ciascuno dei quali contiene almeno $2^{bh(x)-1}-1$ nodi interni.

I nodi interni del sottoalbero radicato in *x* sono almeno $2(2^{bh(x)-1}-1)+1=2^{bh(x)}-1$. CVD

Alberi Rosso-Neri (6)

Th. Un RB-albero con n nodi interni ha altezza h al più 2 log(n+1).

Dim. Sia *h* l'altezza dell'RB-albero ed *r* la sua radice.

Sappiamo già che $bh(r) \geq h/2$.

Dal lemma precedente, il numero di nodi interni dell'RB-albero, pari al numero di nodi interni nel sottoalbero radicato in *r*, è:

$n \geq 2^{bh(r)}-1 \geq 2^{h/2} -1$ da cui $n+1 \geq 2^{h/2}$ cioè $2 \log(n+1) \geq h$.

CVD

Operazioni su RB-alberi (1)

Il teorema precedente garantisce che le operazioni di:

- ricerca di una chiave
- ricerca del massimo o del minimo
- ricerca del predecessore o del successore

sono tutte eseguite con un costo computazionale *O(log n).*

Operazioni su RB-alberi (2)

Discorso a parte va fatto invece per gli inserimenti e le cancellazioni.

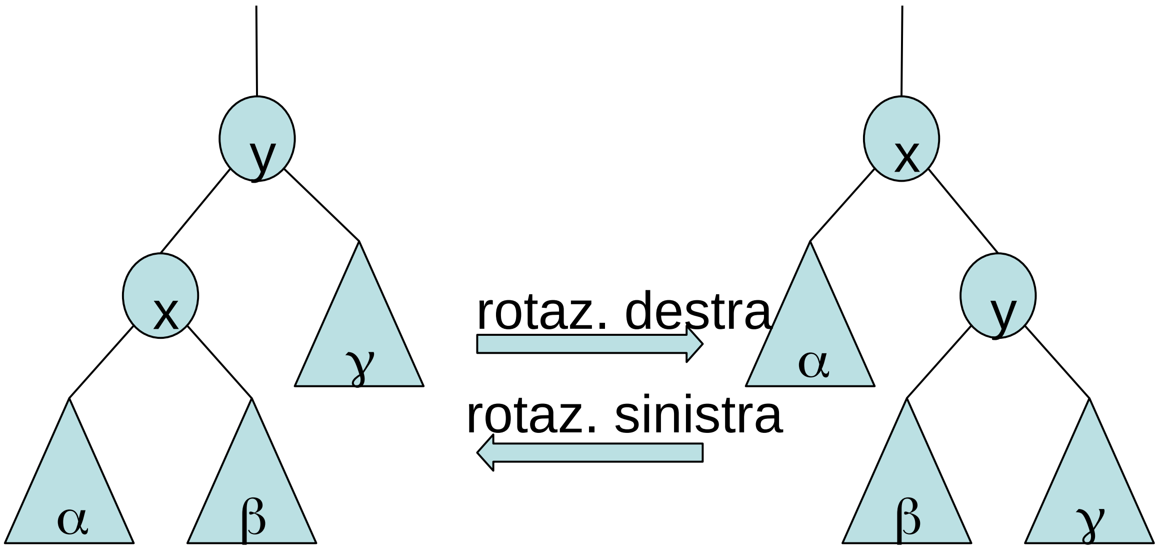
Infatti, l'esigenza di mantenere le proprietà di RB-albero implica che, dopo un inserimento o una cancellazione, la struttura dell'albero possa dover essere riaggiustata, in termini di:

- colori assegnati ai nodi;
- struttura dei puntatori (ossia, collocazione dei nodi nell'albero).

A tal fine sono definite apposite operazioni, dette ***rotazioni***, che permettono di ripristinare in tempo *O(log n)* le proprietà del RB-albero dopo un inserimento o una cancellazione.

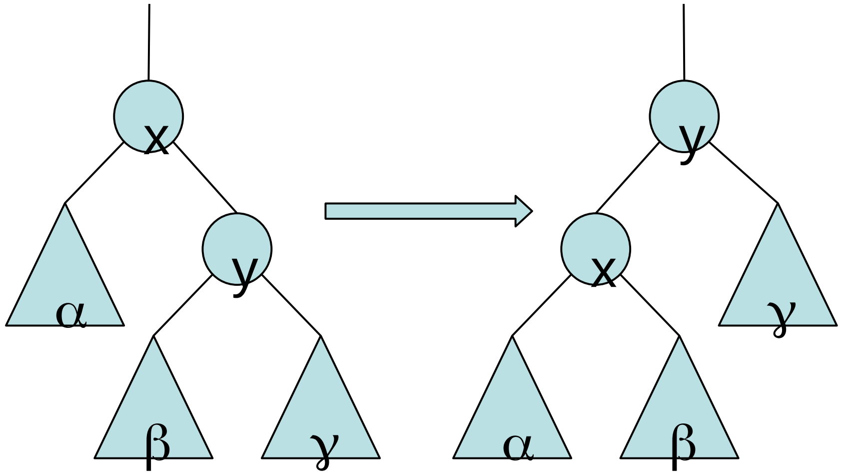
Rotazioni (1)

Le rotazioni possono essere destre o sinistre e sono operazioni locali che non modificano l'ordinamento delle chiavi secondo la visita in-ordine.



Rotaioni (2)

```
Funzione RBA_rotaz_sinistra (p: puntatore alla radice;
   x: puntatore al nodo intorno a cui avviene la rotaz.)
y ←  right[x]; right[x] ← left[y]
if left[y] is not NULL
   parent[left[y]] ← x
parent[y] ← parent[x]
if parent[x]=NULL
   p ← y
else if x=left[parent[x]]
      left[parent[x]] ← y
     else right[parent[x]] ← y
left[y] ← x
parent[x] ← y
```



Lo pseudocodice della rotazione a destra è analogo.

Inserimento (1)

Passo Preliminare: Si inserisce l'elemento seguendo le regole dell'inserimento in un ABR, attribuendo sempre colore rosso al nuovo nodo.

N.B. le regole 1. (ciascun nodo è rosso o nero) e 2. (le foglie fittizie sono nere) si mantengono sempre vere nella struttura; anche la regola 4. (ogni cammino da un nodo a ciascuna foglia ha lo stesso numero di nodi neri) rimane vera dopo un inserimento, visto che il nuovo nodo è sempre colorato di rosso.

L'unica regola che può essere infranta è la 3. (entrambi i figli di un nodo rosso sono neri), infatti il nuovo nodo (rosso) potrebbe essere inserito come figlio di un nodo rosso.

Inserimento (2)

Passo di aggiustamento: Va eseguito solo se il nuovo nodo è stato inserito come figlio (rosso) di un nodo rosso.

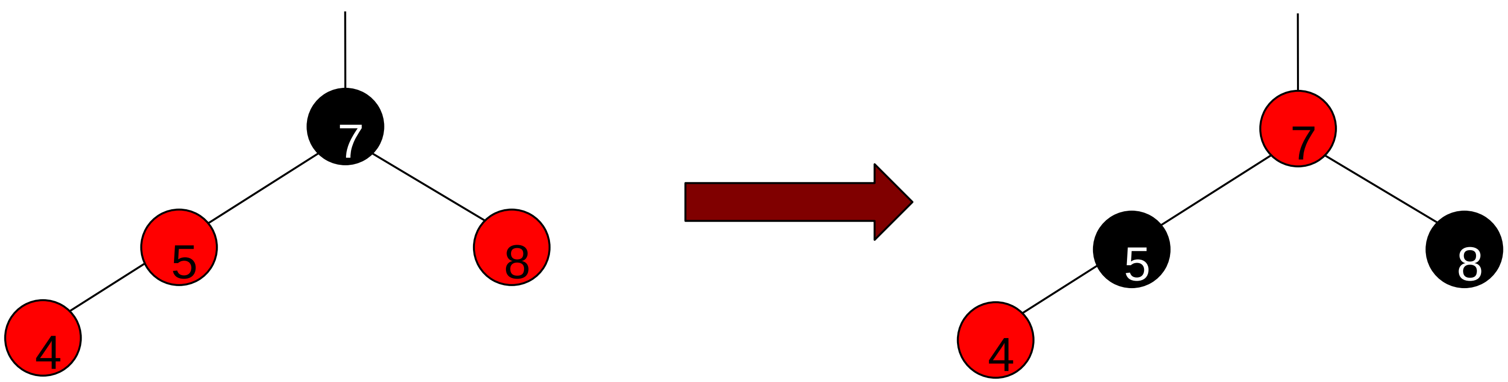
3 possibili eventualità:
Caso 1. Il nodo inserito è figlio di un nodo rosso e lo zio è rosso.
Caso 2. Il nodo inserito è figlio destro di un nodo rosso e lo zio è nero.
Caso 3. Il nodo inserito è figlio sinistro di un nodo rosso e lo zio è nero.

N.B. in tutti e tre i casi, il nonno del nuovo nodo è nero, poiché l'unico punto dell'albero in cui la regola 3. non è rispettata è quello in cui il nodo è stato inserito.

Inserimento (3)

Caso 1: Il nodo inserito è figlio di un nodo rosso e lo zio è rosso.

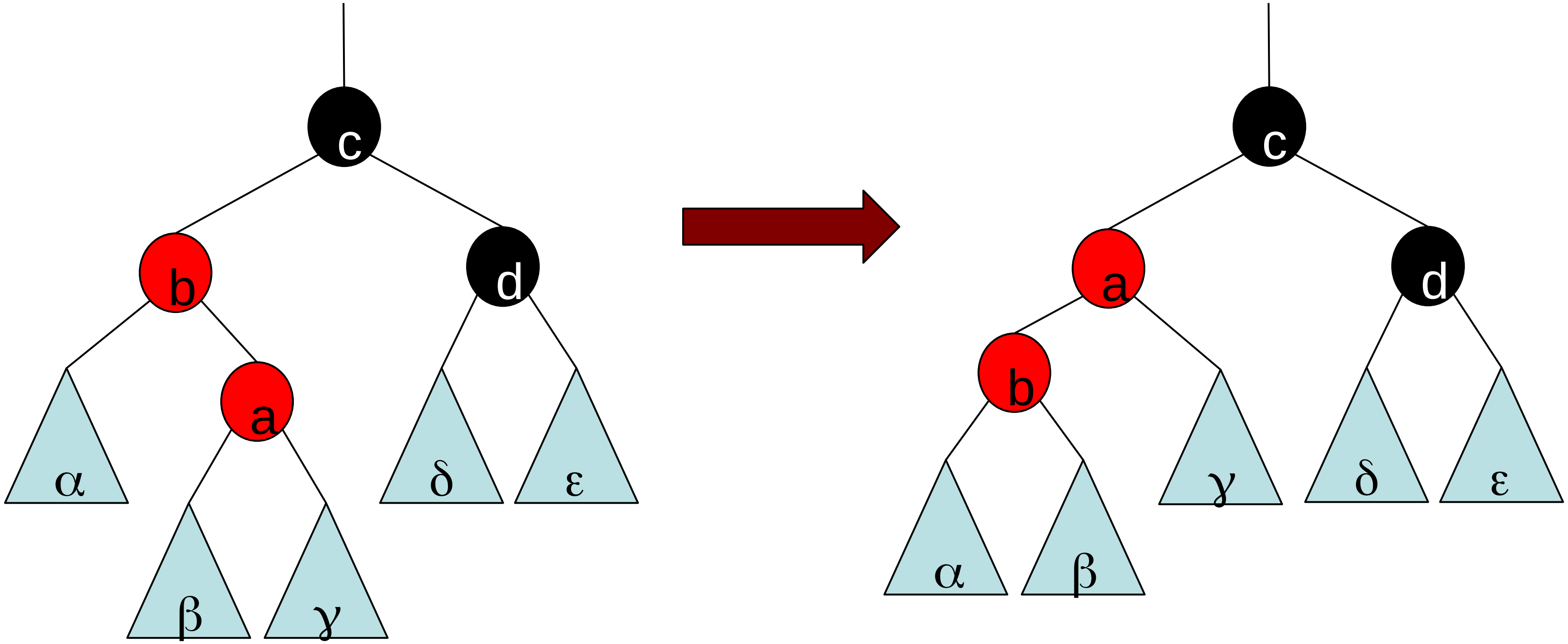
si cambiano i colori di alcuni nodi: detto *x* il nuovo nodo appena inserito, si colorano di nero il padre di *x* e lo zio di *x*, di rosso il nonno di *x*. Ora, o la violazione è stata eliminata (fine inserimento), o è stata portata più in alto (di nuovo Caso 1. a partire dal nonno di *x*, o Caso 2. o Caso 3.).



Inserimento (4)

Caso 2: Il nodo inserito è figlio destro di un nodo rosso e lo zio è nero .

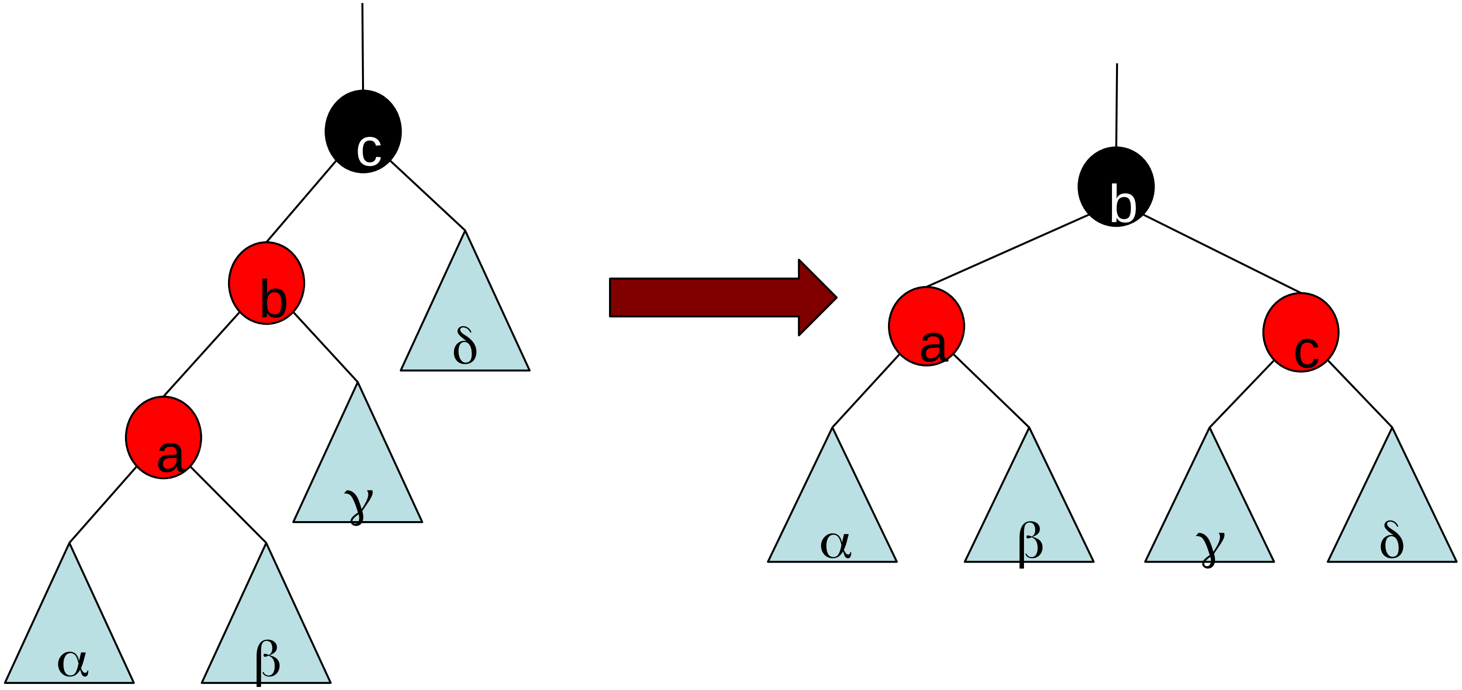
si effettua una rotazione a sinistra imperniata sul padre di *x*, e questo conduce sempre al Caso 3.



Inserimento (5)

Caso 3: Il nodo inserito è figlio sinistro di un nodo rosso e lo zio è nero .

si effettua una rotazione e si cambiano alcuni colori. Ciò garantisce sempre la soluzione della violazione.



Inserimento (6)

Durante un inserimento, è possibile entrare ripetutamente nel caso 1. (ma sempre più in alto nell'albero), eventualmente nel caso 2 e si conclude sempre con il caso 3.

Poiché ogni volta che si presenta una violazione, questa è più in alto nell'albero, e poiché essa è risolta sempre in tempo costante, l'inserimento in un RB albero si effettua in *O(log n)* tempo.

Lo stesso vale per la cancellazione, che decidiamo di omettere qui.

Esercizi

- Costruire un ABR, a partire dall'albero vuoto, inserendo successivamente le chiavi 41, 38, 31, 12, 19 ed 8 in quest'ordine.
- Costruire un RB albero, a partire dall'albero vuoto, inserendo successivamente le chiavi 41, 38, 31, 12, 19 ed 8 in quest'ordine.

/