Corso di laurea in Informatica
Introduzione agli Algoritmi
Didattica blended

Dizionari: Alberi Rosso-Neri

Angelo Monti

SAPIENZA
UNIVERSITÀ DI ROMA

Sulla base delle slides a cura di T. Calamoneri e G. Bongiovanni per il corso di informatica generale AA 2019/2020

Bilanciamento dell'altezza

Un ABR di altezza h contenente n nodi supporta le operazioni di ricerca, inserimento e cancellazione 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***.

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.

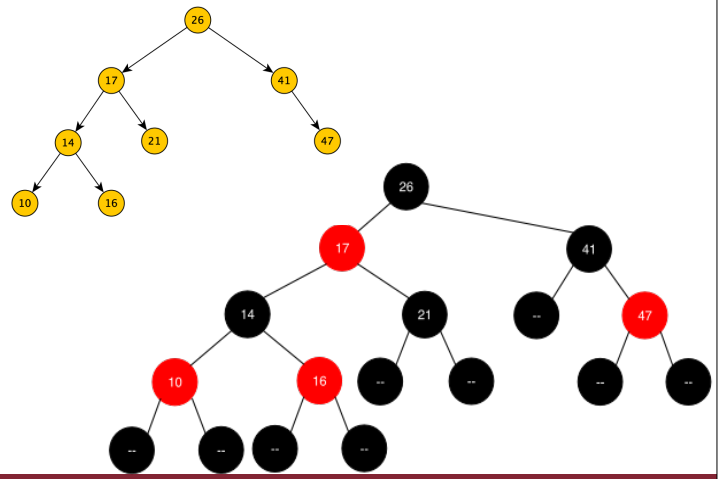
Alberi Rosso-Neri

Un *albero rosso-nero (RB-albero)* è un ABR i cui nodi hanno un campo aggiuntivo, il ***colore***, che può essere solo o 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. la radice è un nodo nero;
3. ciascuna foglia fittizia è nera;
4. se un nodo è rosso i suoi figli sono entrambi neri;
5. ogni cammino da un nodo a ciascuna delle foglie del suo sottoalbero contiene lo stesso numero di nodi neri.

Esempio di albero di ricerca (ABR) e di albero di ricerca Rosso-Nero (RB-albero) con $n = 8$ chiavi



Proprietà degli Alberi Rosso-Neri:

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

Dimostrazione:

- il numero di nodi neri è il medesimo lungo tutti i cammini dalla radice ad una qualunque foglia (proprietà 5). Sia n_1 questo numero.
- **il cammino più corto radice-foglia** ha almeno n_1 nodi.
- **il cammino più lungo radice-foglia** ha meno di $2 \cdot n_1$ nodi. Infatti per proprietà 4 ogni nodo rosso ha figlio nero quindi sul cammino per ogni nodo rosso c'è un nodo nero (suo figlio) inoltre la radice è nera. Questo significa che il numero di nodi rossi del cammino è minore di n_1. Il totale dei nodi del cammino è quindi inferiore ad $n_1 + n_1 = 2 \cdot n_1$;
- dai due punti precedenti segue la proprietà.

CVD

Alberi Rosso-Neri

Definiamo altezza-nera di un nodo ***x*** *che indicheremo con* $bh(x)$, il numero di nodi neri sui cammini dal nodo x (non incluso) alle foglie sue discendenti (è uguale per tutti i cammini, proprietà 5).

L'altezza-nera di un RB-albero è l'altezza nera della sua radice.

Alberi Rosso-Neri

Lemma

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

Dimostrazione

Ragioniamo per induzione sull'altezza dell'albero radicato in x, sia essa $d(x)$.

Se $d(x) = 0$, allora x è una foglia per cui $bh(x) = 0$.

Quindi il suo sottoalbero contiene almeno

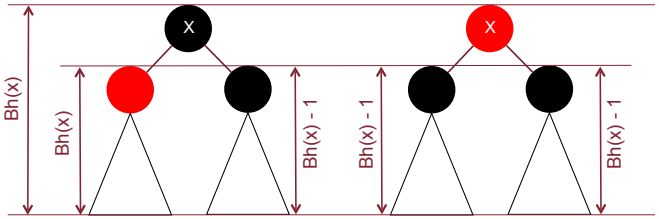
$$2^{bh(x)} - 1 = 2^0 - 1 = 1 - 1 = 0$$

nodi interni.

Dimostrazione (continua)

Sia x un nodo interno, con $d(x) > 0$.

I suoi figli hanno altezza-nera pari a $bh(x)$ o a $bh(x) - 1$, a seconda che essi siano rossi o neri:



Dimostrazione (continua)

Dato che i figli di x sono a distanza $d(x) - 1$ dalle foglie, per essi vale l'ipotesi induttiva.

Quindi il sottoalbero di ciascuno dei due figli contiene almeno $2^{bh(x)-1} - 1$ nodi interni.

Da ciò segue che i nodi interni del sottoalbero radicato in x sono almeno

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

CVD

Teorema.

Un RB-albero con n nodi interni ha altezza limitata da $2 \cdot \log(n+1)$ ed è quindi bilanciato

Dimostrazione

Sia r la radice dell'albero ed h la sua altezza.

Dal lemma precedente abbiamo:

$$n \geq 2^{bh(r)} - 1$$

sappiamo che almeno la metà dei nodi lungo ogni percorso è nero quindi

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

Deduciamo quindi

$$n \geq 2^{\frac{h}{2}} - 1$$

da cui: $2 \cdot \log(n+1) \geq h$

Operazioni su RB-alberi

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

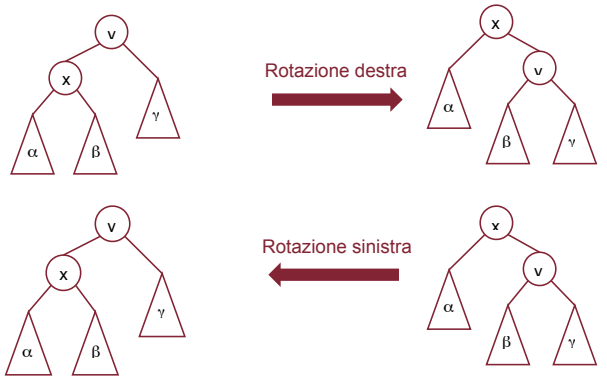
Discorso a parte va fatto invece per gli inserimenti e le cancellazioni.

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).

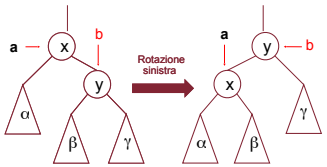
A tal fine sono definite apposite operazioni, dette ***rotazioni***, che permettono di ripristinare in tempo $O(\log n)$ le proprietà del RB-albero dopo un inserimento o una cancellazione.

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



Rotazione sinistra

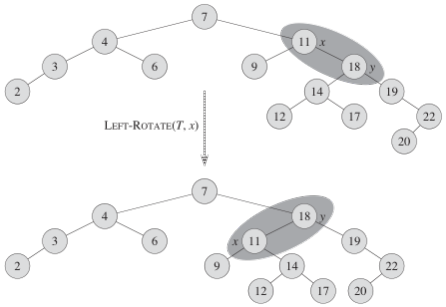
```
def RBA_rotaz_sinistra(p, a):
   b =  a.right
   a.right = b.left
   if b.left!= None:
      b.left.parent = a
   b.left = a
   b.parent = a.parent
   if a == p:
      p = b
   elif a==a.parent.left:
      a.parent.left = b
   else:
       a.parent.right = b
   a.parent = b
   return p
```



Costo computazionale: $\Theta(1)$.

Lo pseudocodice della rotazione a destra è analogo.

Esempio Rotazione a sinistra

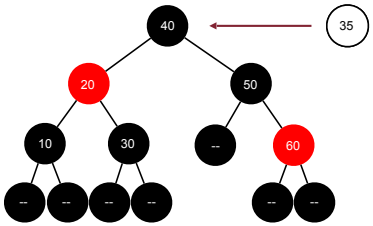


Nota: la visita inorder prima e dopo la rotazione produce lo stesso elenco di chiavi!

Inserimento

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

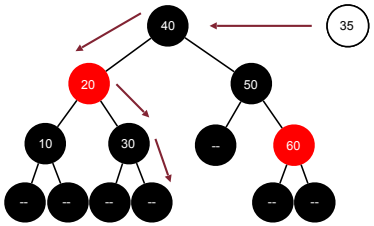
Esempio:



Inserimento (1)

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

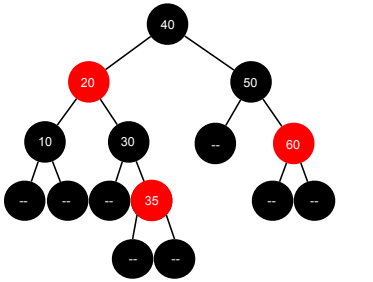
Esempio:



Inserimento (1)

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

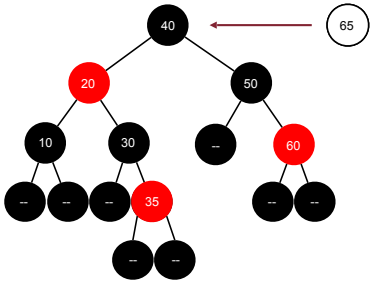
Esempio:



Inserimento (1)

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

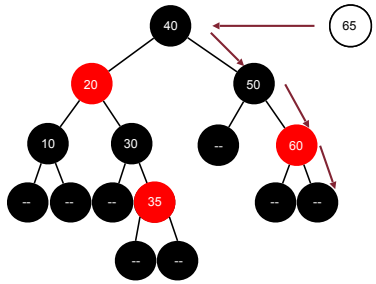
Esempio:



Inserimento (1)

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

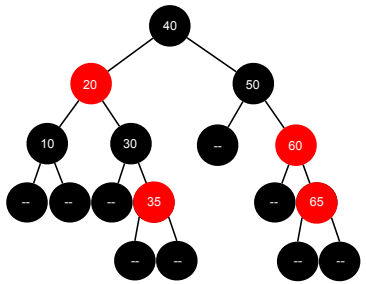
Esempio:



Inserimento (1)

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

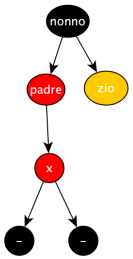
Esempio:



Inserimento

- Le regole 1. (ciascun nodo è rosso o nero) e 3. (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.
- Le uniche regole che possono essere infrante sono:
  - la regola 2. (la radice è un nodo nero)
  - la regola 5. (entrambi i figli di un nodo rosso sono neri), infatti il nuovo nodo (rosso) potrebbe essere inserito come figlio di un nodo rosso.

Inserimento: cose di cui siamo certi dopo il passo preliminare dell'inserimento del nodo x in un albero RB non vuoto quando quando si crea la violazione (vale a dire padre e figlio di colore rosso)



- il nonno esiste (infatti un nodo rosso non può essere radice in albero RB).
- il nonno è nero (infatti un nodo rosso non può avere avere un figlio rosso in un albero RB).
- lo zio esiste (infatti ogni nodo interno ha sempre due figli in un albero RB).

Passo di aggiustamento per Inserimento:

Va eseguito solo se il nuovo nodo è stato inserito come radice (rossa) o come figlio (rosso) di un nodo rosso.

4 possibili eventualità:

Caso 0: la radice è rossa

Caso 1: il nodo x inserito è figlio sinistro di un nodo rosso e **lo zio è nero.**

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

Caso 3: il nodo x inserito è figlio di un nodo rosso e **lo zio è rosso.**

N.B considereremo qui i casi in cui lo zio è figlio DESTRO del nonno gli altri casi sono simmetrici.

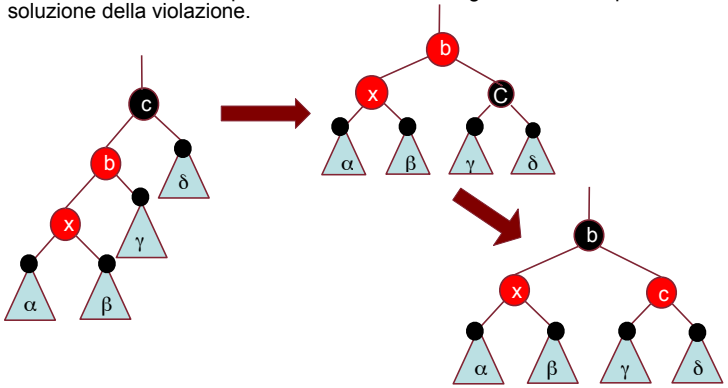
Inserimento

Caso 0: Il nodo x che crea la violazione è la radice.

Si colora di nero la radice.
Ora, la violazione è stata eliminata (fine inserimento)

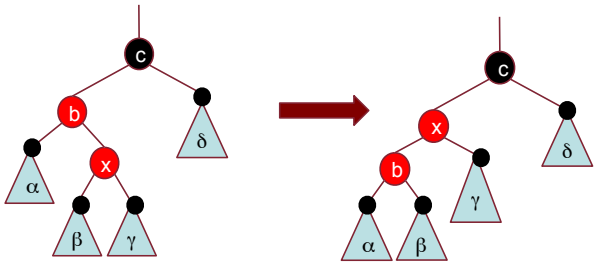
Caso 1: Il nodo x che crea la violazione è figlio sinistro di un nodo rosso e **lo zio è nero**.

Si effettua una rotazione ***a destra*** imperniata sul **nonno** e si scambiano i colori del padre e del nonno. Ciò garantisce sempre la soluzione della violazione.



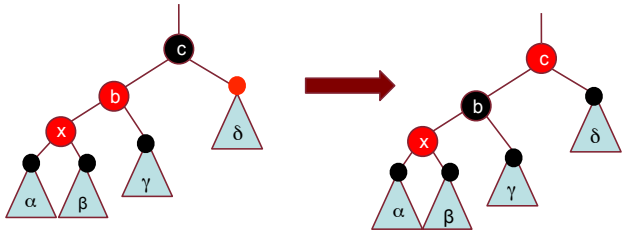
Caso 2: Il nodo x che crea la violazione è figlio destro di un nodo rosso e **lo zio è nero**.

Si effettua una **rotazione a sinistra** imperniata sul padre di x, e questo conduce al Caso 1.

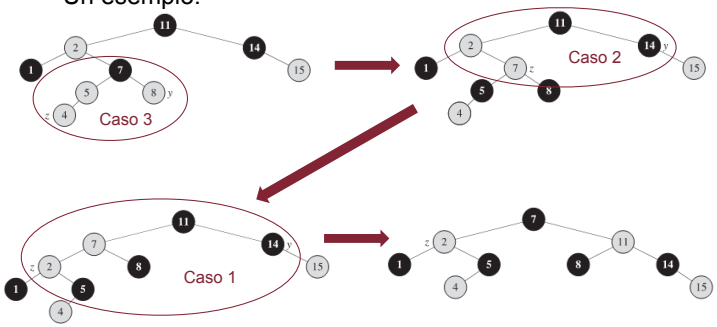


Caso 3: Il nodo x che crea la violazione è figlio di un nodo rosso e **lo zio è rosso.**

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 (a violare potrebbe essere il nonno di x: di nuovo Caso 0, o Caso 1 o Caso 2 o Caso 3).



Un esempio:



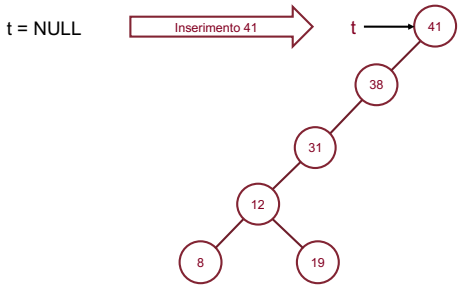
NOTA: il problema si può propagare verso l'alto!

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

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.

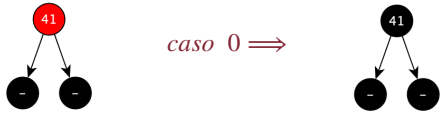
Esercizio 1 svolto.
Costruire un **ABR**, a partire dall'albero vuoto, inserendo successivamente le chiavi **41, 38, 31, 12, 19 ed 8** in quest'ordine.



Esercizio 2 svolto

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

Inserisco 41 e risolvo le violazioni derivanti dall'inserimento:



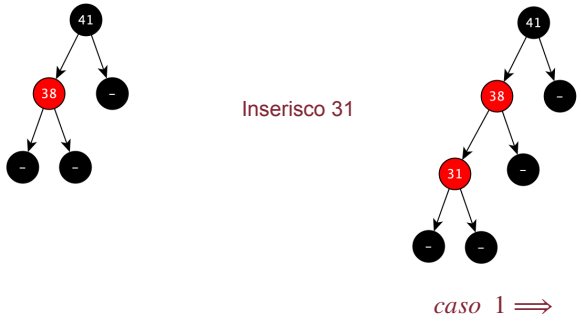
Esercizio 2 svolto, continuo

41, 38, 31, 12, 19, 8

ok!

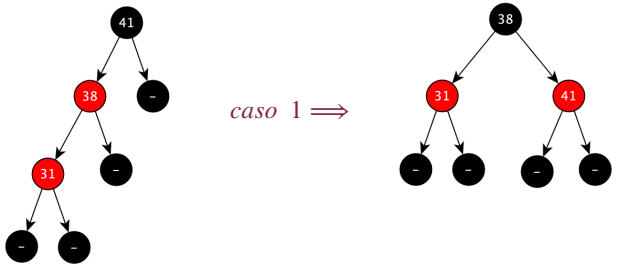
Esercizio 2 svolto continuo

41, 38, 31, 12, 19, 8



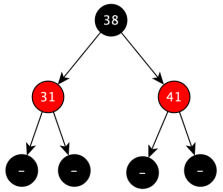
Esercizio 2 svolto continuo

Risolvo le violazioni derivanti dall'inserimento di 31

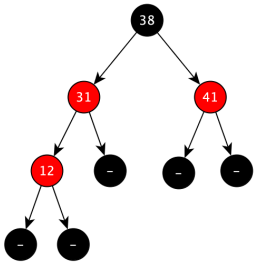


Esercizio 2 svolto continuo

41, 38, 31, 12, 19, 8



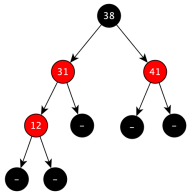
Inserisco 12



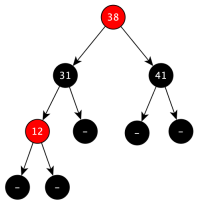
caso 3 $\Longrightarrow$

Esercizio 2 svolto continuo

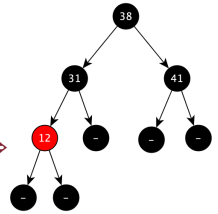
Risolvo le violazioni derivanti dall'inserimento di 12



caso 3 $\Longrightarrow$

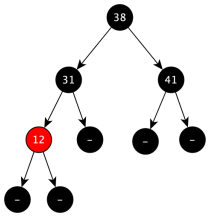


caso 0 $\Longrightarrow$

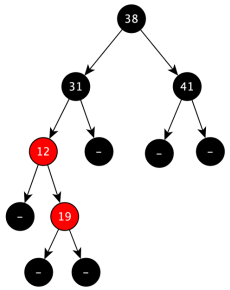


Esercizio 2 svolto continuo

41, 38, 31, 12, 19, 8



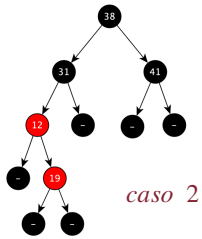
Inserisco 19



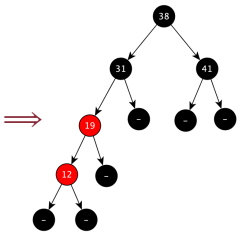
caso 2 $\Longrightarrow$

Esercizio 2 svolto continuo

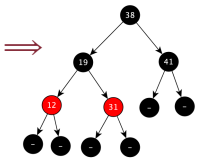
Risolvo le violazioni derivanti dall'inserimento di 19



caso 2 $\Longrightarrow$

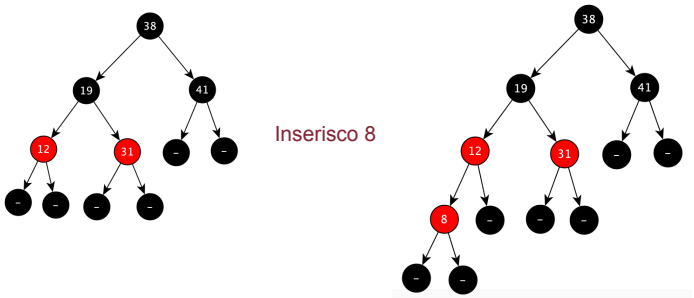


caso 1 $\Longrightarrow$



Esercizio 2 svolto continuo

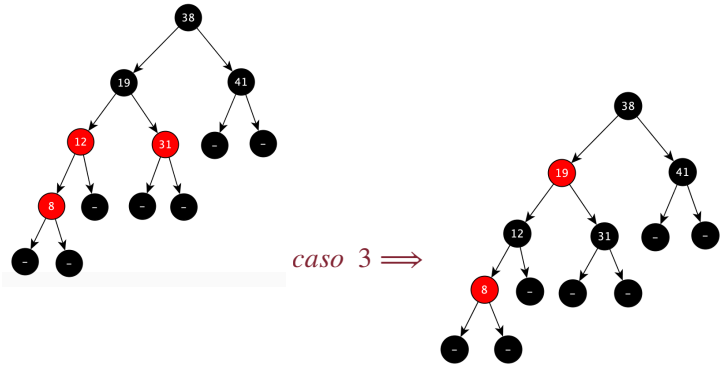
41, 38, 31, 12, 19, 8



caso 3 $\Longrightarrow$

Esercizio 2 svolto continuo

Risolvo le violazioni derivanti dall'inserimento di 8



Corso di laurea in Informatica
Introduzione agli Algoritmi
Didattica blended

Esercizi per casa

SAPIENZA
UNIVERSITÀ DI ROMA

Esercizi

- Scrivere lo pseudocodice di una funzione che ordina un vettore A:
  1. inserendo i suoi elementi in un ABR;
  2. ricopiando su A gli elementi incontrati eseguendo la visita inorder sull'ABR.

  Valutarne il costo computazionale. Se l'ABR è un R&B, cambia qualcosa? se si, cosa?