CANCELLAZIONE IN AVL

E' il caso duale dell' inserimento: se cancello a destra dovrò eventualmente ruotare a destra, analogamente a sinistra

```
CANCELLA_AVL(T,K)
IF T≠ NIL THEN
   IF T->Key < K THEN
      T->dx = CANCELLA_AVL(T->dx, K)
      T= BILANCIA_SX(T)
   ELSE
      IF T->Key > K THEN
         T->sx = CANCELLA_AVL(T->sx, K)
         T= BILANCIA_DX(T)
   ELSE
         T= CANCELLA_ROOT(T)
RETURN T
```

```
CANCELLA_ROOT(T)

IF T->Sx ≠ NIL AND T->dx ≠ NIL THEN
    TMP = STACCA_MIN_AVL(T->dx, T)
    T->Key = TMP->Key
    T = BILANCIA_SX(T)
ELSE
    TMP = T
    IF T->Sx = NIL THEN
        T = T->dx
    ELSE
        T = T->Sx
dealloca(TMP)
RETURN T
```

è l'omologo di quello per gli ABR

Sovrascrivi il nodo da eliminare con il successore

Se stacca min viola le proprietà di AVL

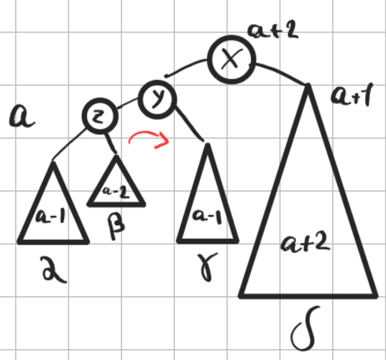
In questi casi non avrò mai sbilanciamenti

Eliminiamo la radice

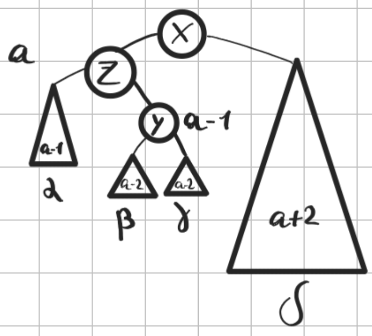
In TMP salviamo sempre la variabile da eliminare

```
STACCA_MIN_AVL (T,P)

IF T ≠ NIL THEN

    IF T->Sx ≠ NIL THEN

        TMP=STACCA_MIN_AVL (T->Sx,T)      Ritorna il minimo, potrebbe ridurre
                                          l'altezza e violare quindi AVL
        NEWROOT= BILANCIA_DX (T)          Ritorna la nuova radice post-bilancio

    ELSE                                  Siamo arrivati al minimo

        NEWROOT = T->dx                   Salviamo la radice del sottoalbero che
                                          dovremo attaccare al padre

    IF T=P->Sx THEN                       Troviamo dov'era attaccato T rispetto
                                          al padre P, e puntiamo a NEWROOT
        P->Sx = NEWROOT

    ELSE

        P->dx = NEWROOT

RETURN TMP                                TMP è il minimo, che verrà deallocato
```

Es.



Supponiamo una cancellazione in γ, che avrà quindi altezza a-3 provocando una violazione. Quando chiameremo bilancia questo effettuerà una rotazione sinistra.



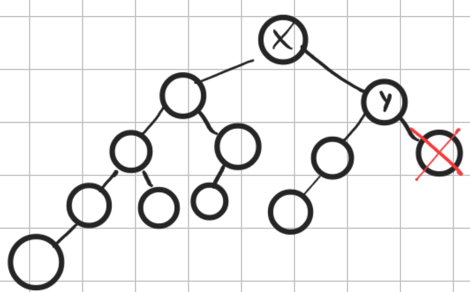
Bilancia restituirà z e la chiamata sospesa aggancerà x a z. Abbiamo ancora una violazione

A differenza dell' dell' inserimento, dopo la rotazione abbiamo ancora una violazione da gestire dopo la cancellazione.

ROTAZIONI MASSIME PER CANCELLAZIONE

Nell' esempio precedente abbiamo dovuto effettuare più di una rotazione perché siamo andati a cancellare un nodo nel sottoalbero che già era più basso dell' altro.

In generale il caso peggiore si ha cancellando in un AVL minimo, che è fortemente instabile (tutti gli alberi hanno una differenza di altezza altezza di uno)



Cancellando la foglia meno profonda avremo una violazione in y, richiedendo una rotazione che però genererà a sua volta una violazione in x perché riduciamo l'altezza dell'albero in y di uno. Nel caso peggiore il numero di rotazioni necessarie è pari alla profondità della foglia meno profonda

ALBERI RED-BLACK

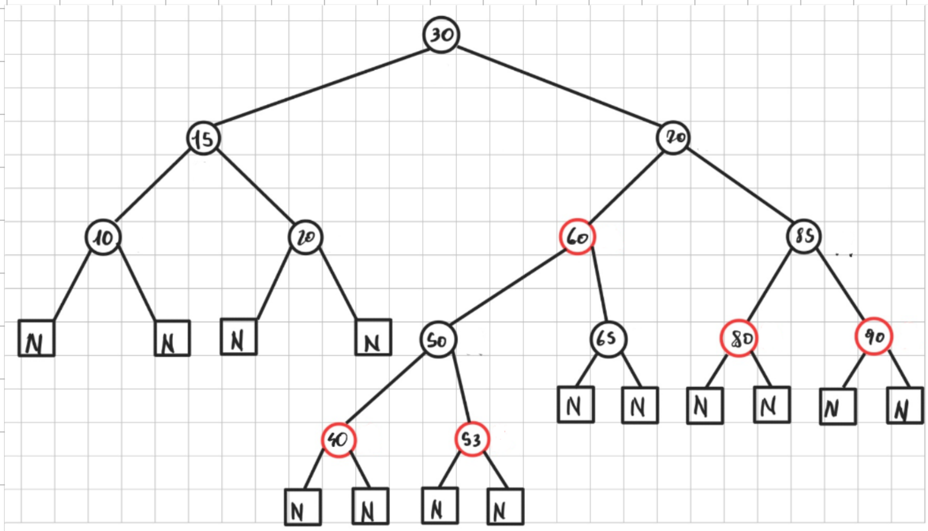
Sono ABR superclassi di APB e AVL in cui:

1. Ogni nodo interno ha due figli
2. Tutti i nodi sono o rossi o neri
3. Ogni nodo rosso ha figli neri
4. $\forall x \in T$ il numero di nodi neri da x a una qualsiasi foglia è sempre lo stesso
5. I nodi NIL sono tutti neri

Per la proprietà 4 ogni nodo ha una caratteristica detta ALTEZZA NERA ovvero il numero di nodi neri tra il nodo e le foglie (nodo escluso)

Se un nodo interno non ha due figli gli viene aggiunto un nodo fittizio (vuoto) detto NODO NIL, e saranno tutte e sole le foglie

Indichiamo con quadrati i nodi NIL e vediamo un esempio:



PROPRIETA'

$\forall T \in ARB$ con $|T| = n$ nodi interni

① $H(T) = O(\lg_2 n)$ con $n = |T|$

(1.5) $H(T) \leq 2\lg_2(n+1)$

③ $n \geq 2^{bh(T)} - 1$ (bh è l'altezza nera)

DIMOSTRAZIONE ③

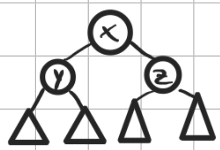
Vogliamo dimostrare per induzione che

$\forall h \geq 0$ se $H(x) = h \Rightarrow n \geq 2^{bh(x)} - 1$ con $x \in ARB$ e n nodi interni

• $n = 0$

è banalmente 0 perché avremo solo un nodo NIL e quindi $bh = 0$

• $n > 0$



$n(x) = 1 + n(y) + n(z)$

Dobbiamo però esprimere $ni(y)$ e $ni(z)$ in funzione di x.

Applicando l'ipotesi induttiva avremo

$n(x) \geq \cancel{1} + 2^{bh(y)} \cancel{-1} + 2^{bh(z)} - 1$

Prendiamo in esame l'albero radicato in y (analogamente vale per z):

• Se y è rosso $bh(x) = bh(y)$ (non aggiunge nulla)

• Se y è nero $bh(x) = bh(y) + 1$

Possiamo dire che $bh(x) - 1 \leq bh(y) \leq bh(x)$

Per monotonicità $bh(y) \geq bh(x) - 1 \Rightarrow 2^{bh(y)} \geq 2^{bh(x)-1}$

Analogamente varrà per x, sostituiamo nella prima espressione le nostre quantità ancora maggiori. Per transitività avremo:

$n(x) \geq 2^{bh(x)-1} + 2^{bh(x)-1} - 1 = 2^{bh(x)} - 1$ ✓

DIMOSTRAZIONE ①

Dalla proprietà ③ sappiamo che dato un albero Red-black T, i nodi interni sono $ni(T) \geq 2^{bh(T)} - 1$.

$ni(T) = n$ (nodi che contengono informazioni, le foglie sono nodi NIL)

$bh(T)$ può essere uguale ad h (albero tutto nero), il massimo numero di nodi rossi che possiamo avere è $n/2$ (nodi rossi e neri alternati), di conseguenza il numero minimo di nodi neri è $n/2$ e quindi bh è almeno $h/2$.

$$h/2 \leq bh \leq h$$

Per monotonia $2^{h/2} \leq 2^{bh}$ e banalmente $2^{h/2} - 1 \leq 2^{bh} - 1$

Per transitività sulla prop. ③ $n \geq 2^{bh} - 1 \geq 2^{h/2} - 1$

Portiamo -1 a sinistra e calcoliamo il logaritmo

$$n+1 \geq 2^{h/2} \rightarrow \lg_2(n+1) \geq \frac{h}{2} \Rightarrow 2\lg_2(n+1) \geq h$$

Che è la nostra prop. (1.5) e per definizione $h = O(\lg_2)$ ✓