Algoritmi e Strutture Dati (16/11/2021)

* Aumento di strutture dati

→ statistiche d'ordine dinamiche

→ alberi di intervalli

* Statistiche d'ordine dinamiche

alberi binari di ricerca con due operazioni addizionali

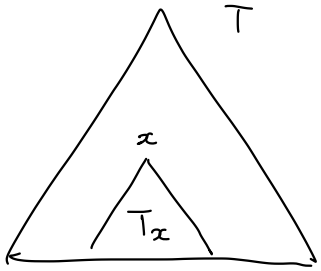
* Select (T, i) — i-mo elemento in una visita inorder

inorder $x_1 \dots x_i \dots x_n$

↑ Select(T, i)

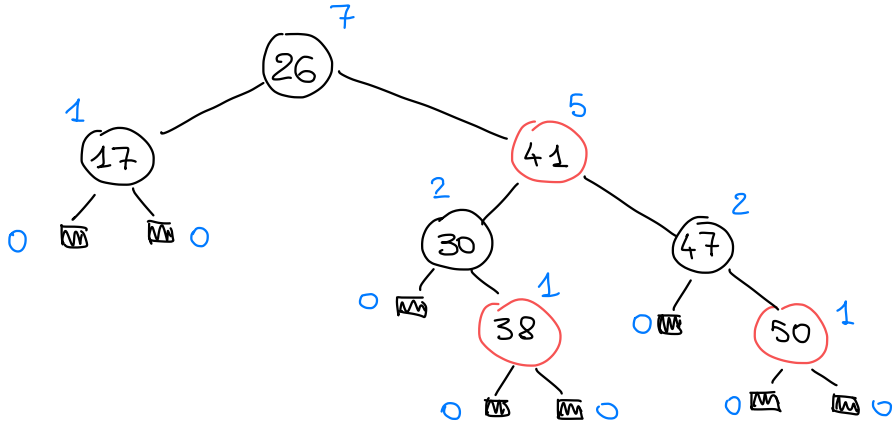
* Rank (T, x) = posizione di x nella lista in order

RB - trees



nodi x con campo addizionale

$x.size$ = # nodi interni in T_x

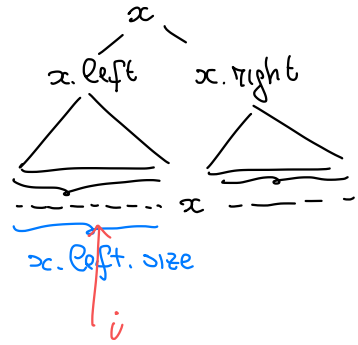


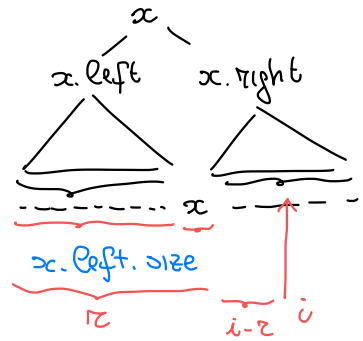
T.nil.size = 0

$x.size = x.left.size + x.right.size + 1$

* Select

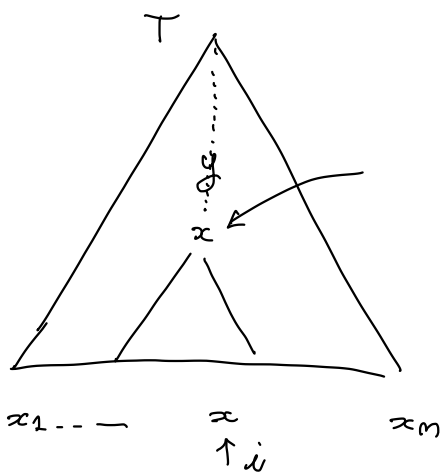
```
select (x, i)                    // i^mo modo nella visita in order di T_x
    r = x.left.size + 1          (1 ≤ i ≤ x.size)
    if (i = r)
        return x
    else if i < r
        return select(x.left, i)
    else
        return select(x.right, i - r)
```





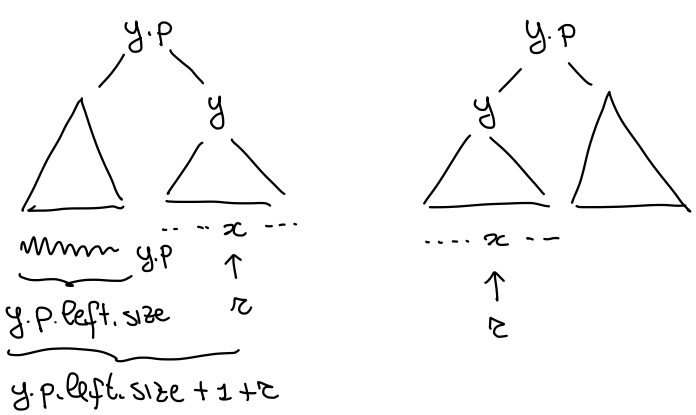

```
Select (T, i)
    return select (T.root, i)
```

complenità $O(h) = O(\log n)$



Rank

```
Rank (T, x)
    r = x.left.size + 1
    y = x          // r posizione di x
                   nel sotto albero
                   radicato in y

    while (y ≠ T.root)
        if y = y.p.right
            r = r + y.p.left.size + 1

        y = y.p

    return r
```



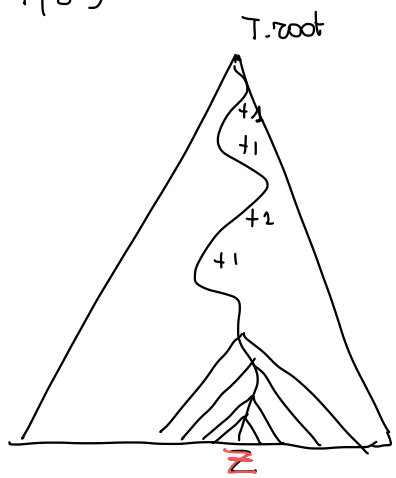
complenità : $O(h) = O(\log n)$

* Come mantenere aggiornato il campo size?

* Insert su RB

1. Insert(T, z)
2. RB-Insert-FixUp(T, z)

1. Insert(T, z)



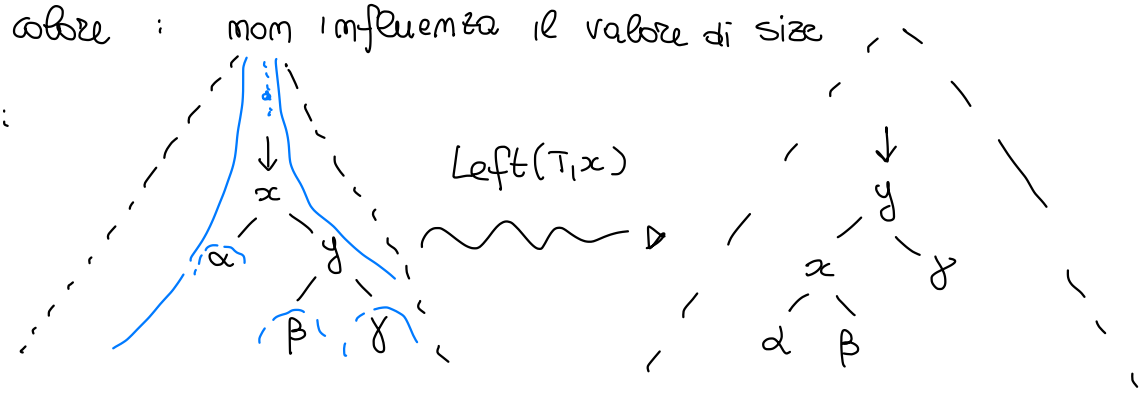

```
Insert (T, z)
    x = T.root
    y = T.nil
    while x ≠ T.nil
        x.size = x.size + 1
        y = x
        if z.key < x.key
            x = x.left
        else
            x = x.right

    z.p = y
    if y = T.nil
        T.root = z
    else if z.key < y.key
            y.left = z
        else y.right = z

    z.color = RED

    z.size = 1
```

$O(\log n)$

② RB-Insert-FixUp(T, z)

→ cambi di colore : non influenza il valore di size

→ <u>rotazioni</u> :



Left(T, x)

[come prima

$O(1)$

y.size = x.size

x.size = x.left.size + x.right.size + 1

costo <u>RB-Insert</u>

$O(\log n)$

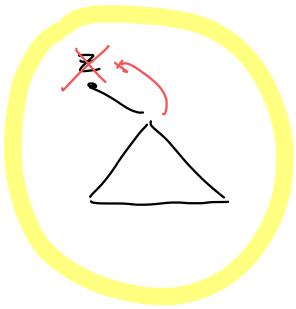
* RB-Delete

① Delete (T, z)

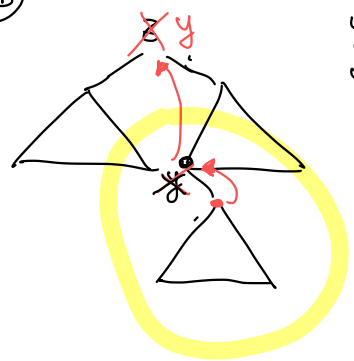
② RB-Delete-FixUp(T, z)

① Delete (T, z)

1.a



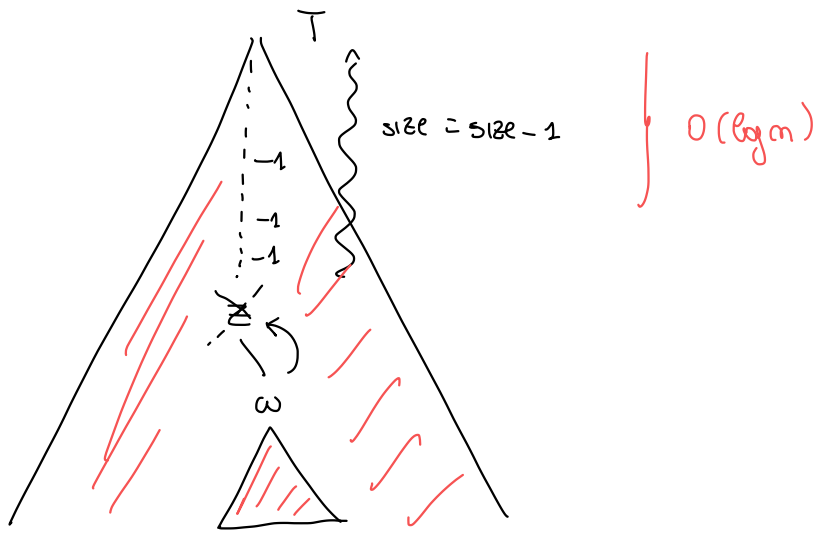
1.b



y.col = z.col

y.size = z.size

vero unico problema



② RB-DeleteFixUp(T, z)

→ cambi di colore
→ 3 max rotazioni

} complessità inalterata $O(\log n)$

* <u>Teorema dell'aumento degli alberi RB</u>

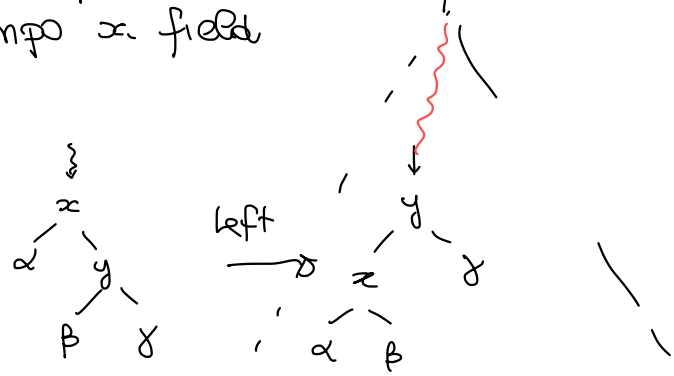
Se modifichiamo i nodi degli alberi RB aggiungendo un nuovo campo

x.field

tale che si possa calcolare in tempo $O(1)$ da

$$\left\{ \begin{array}{l} x \\ x.\text{left} \\ x.\text{right} \end{array} \right\}$$

allora RB-Insert e RB-Delete possono essere modificate di modo da mantenere aggiornato il campo x.field con complessità $O(\log n)$



Ex

x.size = x.left.size + x.right.size + 1

Esempio 2 : Interval Trees

nodi $x.int$ $[x.int.low, x.int.high]$

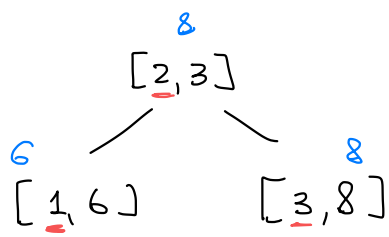
Search (x, i) $\rightsquigarrow$ nodo y in T_x t.c. $y.int \cap i \neq \emptyset$

i ↑ intervallo

RB-Trees

$x.int$ $\begin{cases} \boxed{x.int.low} \text{ (key)} \\ x.int.high \end{cases}$

$\boxed{x.max}$ = massimo $y.int.high$ in nodi di T_x



$$\begin{cases} T.nil.max = 0 \\ x.max = max\{x.left.max, x.right.max, x.int.high\} \end{cases}$$

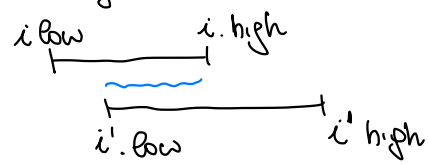
$\rightsquigarrow$ Insert / Delete $O(\log n)$

```
Search (x, i)          // cerca in T_x nodo y t.c y.int ∩ i ≠ ∅
  if (x = T.nil) or (x.int ∩ i ≠ ∅)
      return x
  else if (x.left ≠ T.nil) and (x.left.max ≥ i.low)
      return Search (x.left, i)
```

i' in $x.left$

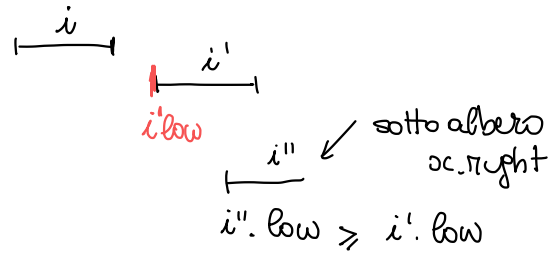
2 possibilità

① $i'.low \leq i.high$



$i \cap i' \neq \emptyset$

② $i'.low > i.high$



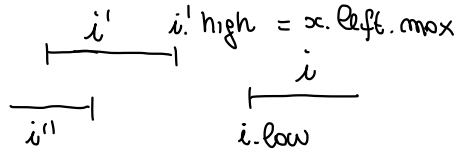
$\Rightarrow i'' \cap i = \emptyset$

else // $\boxed{x.left = T.nil}$

oppure

$\boxed{x.left.max < i.low}$

return Search (x.right, i)



$\forall i''$ in x. left

$i''.high \leq i'.high < i.low$ $\Big\} \Rightarrow i'' \cap i = \emptyset$

complessità $O(h) = O(\log n)$