Alberi AVL

Francesco Gori

July 3, 2024

1 Introduzione

Gli alberi AVL, dal nome dei loro inventori Adelson-Velsky e Landis, sono una classe di alberi binari di ricerca auto-bilancianti simili agli alberi rosso-neri.
Durante l'inserimento e la cancellazione di nuovi elementi può essere necessario dover fare modifiche all'albero per mantenere l'altezza dell'albero bilanciata; questo fa sì che le operazioni abbiano una complessità temporale specifica.
Le operazioni di inserimento, cancellazione, minimo, massimo, successore, predecessore e ricerca si eseguono in tempo $O(lg\, n)$

2 Proprietà degli Alberi AVL

Gli elementi di un AVL hanno, oltre agli attributi di un ABR normale, un attributo h che ne specifica l'altezza. Questa è definita come il numero di archi nel percorso più lungo dalla radice a una foglia. La proprietà principale degli alberi AVL è che, per ogni nodo, l'altezza dei suoi sottoalberi differisce al massimo di 1. In formule:

$$|BF(x)| \leq 1 \quad \text{dove} \quad BF(x) = x.left.h - x.right.h$$

2.1 Altezza Massima e Numero Minimo di Nodi

L'altezza massima di un AVL con n nodi interni è $2lg(n+1)$.
Definendo *n(h)* come il numero minimo di nodi in un AVL di altezza h, avremo il seguente **teorema**:

$$\forall h > 1 \quad \text{si ha} \quad n(h) \geq 2^{h/2} - 1$$

Dimostrazione: induzione su h

- Casi base:
  - $h = 1 : \ n(1) = 1 > 2^{1/2} - 1 = \sqrt{2} - 1$
  - $h = 2 : \ n(2) = 2 > 2^{1/1} - 1 = 1$
- Passo induttivo (per $h \geq 2$):
  da definizione di AVL (un sottoalbero ha altezza $h-1$, l'altro almeno $h-2$)
  $n(h) \geq 1 + n(h-1) + n(h-2)$ **ipotesi induttiva:**
  $n(h) \geq 1 + 2^{\frac{h-1}{2}} - 1 + 2^{\frac{h-2}{2}} - 1 = (2^{-1/2} + 2^{-1})2^{h/2} - 1 > 2^{h/2} - 1 \Rightarrow n = n(h) \geq 2^{h/2} - 1$
- $\Rightarrow lg(n+1) \geq h/2 \Rightarrow h \leq 2lg(n+1)$

3 Operazioni sugli Alberi AVL

3.1 Inserimento

Durante l'inserimento di un nuovo nodo, si procede come in un normale albero binario di ricerca. Dopo l'inserimento, si risale l'albero dai nodi figli ai nodi antenati, aggiornando i fattori di bilanciamento e applicando le rotazioni necessarie per mantenere l'albero bilanciato.

3.2 Rotazioni

Servono a mantenere gli alberi bilanciati, cambiando i puntatori degli elementi. Le rotazioni possono essere a sinistra o a destra. La rotazione mantiene l'ordinamento delle chiavi, bilanciando l'albero. Dato che modificano un numero costante di puntatori e aggiornano gli attributi h, le rotazioni avvengono in tempo $O(1)$.

RVL-Insert(T,z)

```
y = T.NIL
x = T.root
while x ≠ T.NIL do
    y = x
    if z.key < x.key then
        x = x.left
    else
        x = x.right
    end if
end while
z.p = y
if y = T.NIL then
    T.root = z
else if z.key < y.key then
    y.left = z
else
    y.right = z
end if
z.left = T.NIL
z.right = T.NIL
z.h = 1
call AVL-Insert-Fixup(T, z)
```

Left-Rotate(T,x)

```
y = x.right                                ▷ Imposta y
x.right = y.left        ▷ Sposta sottoalbero sx di y in dx di x
x.h = max(x.left.h, x.right.h) + 1
if y.left ≠ T.NIL then
    y.left.p = x
end if
y.p = x.p                    ▷ Collega il padre di x a y
if x.p = T.NIL then
    T.root = y
else if x = x.p.left then
    x.p.left = y
else
    x.p.right = y
end if
y.left = x                                 ▷ Pone x a sx di y
y.h = max(y.left.h, y.right.h) + 1
x.p = y
```

- Prima dell'esecuzione x.right $\neq T$.NIL

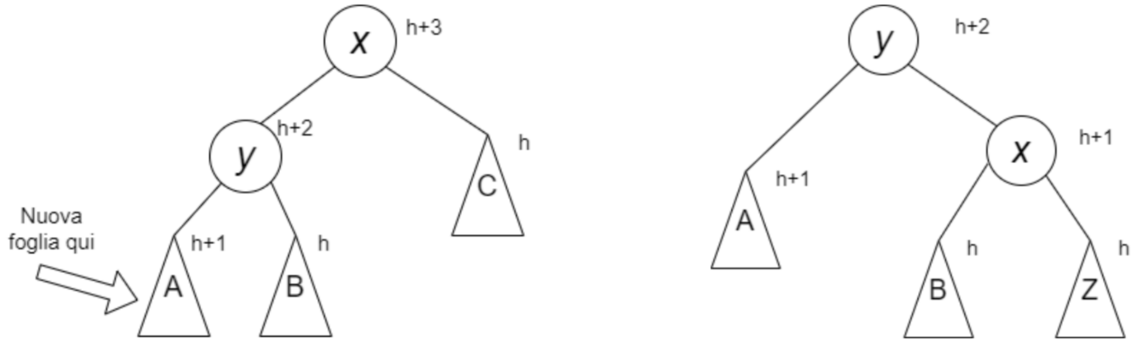
3.3 Problematiche

Al termine dell'inserimento solo gli antenati del nuovo nodo possono essere sbilanciati (ovvero gli unici nodi con sottoalberi modificati).
$\forall x$ antenato di z vale: $\quad -2 \leq BF(x) \leq 2 \quad$ (prima dell'inserimento era $-1 \leq BF(x) \leq 1$)
È un problema se $BF(x) = \pm 2$, dato che uno dei due sottoalberi sarà più alto dell'altro di 2 livelli.
Consideriamo il caso $BF(x) = 2$ (=-2 è simmetrico), questo si divide in due sottocasi:

- Il nuovo nodo è stato inserito in $x.left.left \quad \Rightarrow BF(x.left) = 1$
- Il nuovo nodo è stato inserito in $x.left.right \quad \Rightarrow BF(x.left) = -1$

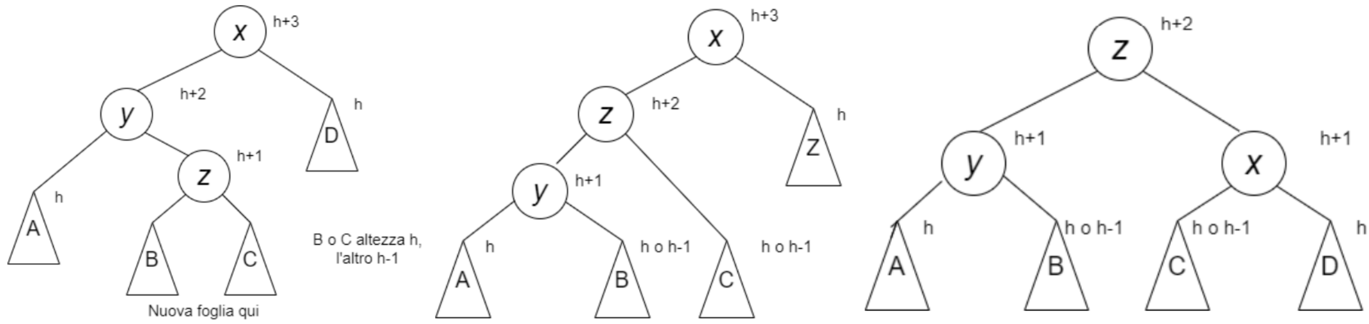
3.3.1 BF(x)=2 nuovo nodo in x.left.left BF(x.left)=1

Per risolvere questa situazione è sufficiente una rotazione a destra sul nodo x.



3.3.2 BF(x)=2 nuovo nodo in x.left.right BF(x.left)=-1

In questo caso servono 2 rotazioni: una a sinistra sul nodo y e una a destra sul nodo x.



3.3.3 Algoritmo Fixup

AVL-Insert-Fixup(T,x)

1: $x \leftarrow x.p$
2: **while** $x \neq T.NIL$ **do**
3: $\quad x.h \leftarrow \max(x.left.h, x.right.h) + 1$
4: $\quad$ **if** $(x.left.h - x.right.h) = 2$ **then** ▷ BH(x)
5: $\quad\quad$ **if** $(x.left.left.h - x.left.right.h) = -1$ **then** ▷ BH(x.left)
6: $\quad\quad\quad$ Left-Rotate(T, x.left)
7: $\quad\quad$ **end if**
8: $\quad\quad$ Right-Rotate(T, x)
9: $\quad\quad x \leftarrow x.p$ ▷ è ancora radice del sottoalbero, 9-12 simmetrico con 4-7
10: $\quad$ **else if** $(x.left.h - x.right.h) = -2$ **then** ▷ BH(x)
11: $\quad\quad$ **if** $(x.right.left.h - x.right.right.h) = 1$ **then** ▷ BH(x.right)
12: $\quad\quad\quad$ Right-Rotate(T, x.right)
13: $\quad\quad$ **end if**
14: $\quad\quad$ Left-Rotate(T, x)
15: $\quad\quad x \leftarrow x.p$ ▷ è ancora radice del sottoalbero
16: $\quad$ **end if**
17: $\quad x \leftarrow x.p$
18: **end while**

3.3.4 Correttezza e Tempo

Invariante di ciclo: all'inizio di ogni iterazione in AVL c'è al massimo una violazione del bilanciamento.
Tempo:

- AVL-Insert $O(lg\, n)$
- AVL-Insert-Fixup $O(1)$ per ogni livello, $O(lg\, n)$ livelli $\Rightarrow O(lg\, n)$

In totale quindi un inserimento occupa $O(lg\, n)$