4.4.2 AVL: alberi binari di ricerca bilanciati

L'**albero AVL** (acronimo derivato dalle iniziali degli autori russi Adel'son-Velsky e Landis che lo inventarono negli anni '60) è un albero binario di ricerca che garantisce avere un'altezza $h = O(\log n)$ per n elementi memorizzati nei suoi nodi. Oltre alla proprietà di ricerca menzionata nel Paragrafo 4.4.1, l'albero AVL soddisfa la proprietà di essere **1-bilanciato** al fine di garantire l'altezza logaritmica.

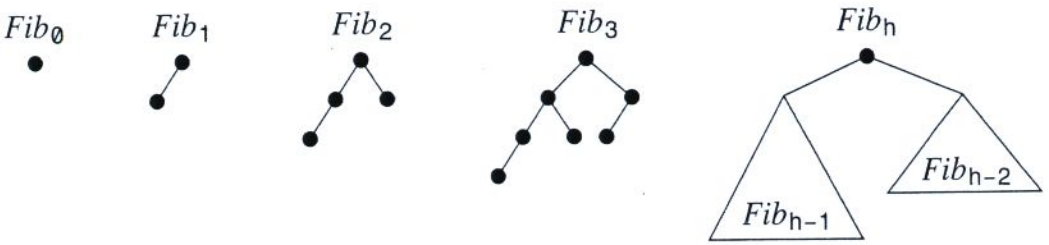


Figura 4.1 Alberi di Fibonacci.

Dato un nodo u, indichiamo con h(u) la sua altezza, che identifichiamo con l'altezza del sottoalbero radicato in u, dove h(null) = -1 (Paragrafo 2.4). Un nodo u è **1-bilanciato** se le altezze dei suoi due figli differiscono per al più di un'unità

$$|h(u.sx) - h(u.dx)| \leq 1 \tag{4.2}$$

Un albero binario è 1-bilanciato se *ogni* suo nodo è 1-bilanciato. Gli alberi dell'Esempio 5.3 sono alberi 1-bilanciati.

La connessione tra l'essere 1-bilanciato e avere altezza logaritmica non è immediata e passa attraverso gli **alberi di Fibonacci**, che sono un sottoinsieme degli alberi 1-bilanciati con il *minor* numero di nodi a *parità* di altezza. In altre parole, indicato con Fib_h un albero di Fibonacci di altezza h e con n_h il suo numero di nodi, eliminando un solo nodo da Fib_h otteniamo che l'altezza diminuisce o che l'albero risultante non è più 1-bilanciato: nessun albero 1-bilanciato con n nodi e altezza h può dunque avere meno di n_h nodi, ossia $n \geq n_h$.

L'albero di Fibonacci Fib_h di altezza h è definito ricorsivamente (Figura 4.1): per $h = 0$, abbiamo un solo nodo, per cui $n_0 = 1$, e, per $h = 1$, abbiamo un albero con $n_1 = 2$ nodi (la radice e un solo figlio, che nella figura è quello sinistro). Per $h > 1$, l'albero Fib_h è costruito prendendo un albero Fib_{h-1} e un albero Fib_{h-2}, le cui radici diventano i figli di una nuova radice (quella di Fib_h). Di conseguenza, abbiamo che $n_h = n_{h-1} + n_{h-2} + 1$, relazione ricorsiva che ricorda quella dei numeri di Fibonacci[5] motivando così il nome di tali alberi.

Teorema 4.3 *Sia* T *un albero AVL di* n *nodi e altezza* h, *allora* $h = O(\log n)$.

Dimostrazione Dimostreremo che Fib_h è l'albero 1-bilanciato di altezza h col minimo numero di nodi. Mostrando che $n_h \geq c^h$ per un'opportuna costante $c > 1$, deriviamo che $n \geq c^h$ e, quindi, che $h = O(\log n)$.

Possiamo osservare nella Figura 4.1 che Fib_0 e Fib_1 sono alberi 1-bilanciati di altezza 0 e 1, rispettivamente, con il *minimo* numero di nodi possibile. Ipotizzando che, per induzione, tale proprietà valga per ogni $\ell < h$ con $h > 1$, mostriamo come ciò sia vero anche per Fib_h ragionando per assurdo. Supponiamo di poter rimuovere un

[5] Ricordiamo che la successione dei numeri di Fibonacci $\{F_i\}_{i \geq 0}$ è definita ricorsivamente nel seguente modo: $F_0 = 0$, $F_1 = 1$ e, per ogni $i \geq 2$, $F_i = F_{i-1} + F_{i-2}$.

nodo da Fib_h mantenendo la sua altezza h e garantendo che rimanga 1-bilanciato: non potendo rimuovere la radice, tale nodo deve appartenere a Fib_{h-1} oppure a Fib_{h-2} per costruzione. Ciò è impossibile in quanto questi ultimi sono minimali per ipotesi induttiva e, se Fib_{h-1} cambiasse altezza, anche Fib_h la cambierebbe, mentre se Fib_{h-2} cambiasse altezza, Fib_h non sarebbe più 1-bilanciato. Quindi, anche Fib_h è minimale e concludiamo che ogni albero 1-bilanciato di altezza h con n nodi soddisfa $n \geq n_h$.

Tabulando i primi 15 valori di n_h e altrettanti numeri di Fibonacci F_h, possiamo verificare per ispezione diretta che vale la relazione $n_h = F_{h+3} - 1$:

h	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
n_h	1	2	4	7	12	20	33	54	88	143	232	376	609	986	1596
F_h	0	1	1	2	3	5	8	13	21	34	55	89	144	233	377

Utilizzando la nota forma chiusa dei numeri di Fibonacci

$$F_h = \frac{\phi^h - (1-\phi)^h}{\sqrt{5}}, \quad \text{dove } \phi = \frac{1+\sqrt{5}}{2} \approx 1,6180339\cdots$$

possiamo affermare che $F_h > \frac{\phi^h - 1}{\sqrt{5}}$ e che, quindi, esiste una costante $c > 1$ tale che $F_h > c^h$ per $h > 2$. Pertanto, $n_h = F_{h+3} - 1 \geq c^h$ e, poiché $n \geq n_h$, possiamo concludere che $n \geq c^h$: in altre parole, ogni albero 1-bilanciato di n nodi e altezza h verifica $h = O(\log n)$. ❑

Per implementare gli alberi AVL, introduciamo un ulteriore campo `u.altezza` nei suoi nodi `u`, tale che `u.altezza = h(u)`. Notiamo che l'operazione di ricerca negli alberi AVL rimane identica a quella descritta nel Codice 4.5. Mostriamo quindi nel Codice 4.7 come estendere l'inserimento per garantire che l'albero AVL rimanga 1-bilanciato.

Codice 4.7 Algoritmo per l'inserimento di un elemento e in un albero AVL con radice u.

```
1  Inserisci( u, e ):
2    IF (u == null) {
3      RETURN f = NuovaFoglia( e );
4    } ELSE IF (e.chiave < u.dato.chiave) {
5      u.sx = Inserisci( u.sx, e );
6      IF (Altezza(u.sx) - Altezza(u.dx) == 2) {
7        IF (e.chiave>u.sx.dato.chiave) u.sx=RuotaAntiOraria(u.sx);
8        u = RuotaOraria( u );
9      }
10   } ELSE IF (e.chiave > u.dato.chiave) {
11     u.dx = Inserisci( u.dx, e );
12     IF (Altezza(u.dx) - Altezza(u.sx) == 2) {
```

```
13        IF (e.chiave < u.dx.dato.chiave) u.dx = RuotaOraria(u.dx);
14        u = RuotaAntiOraria( u );
15      }
16    }
17    u.altezza = max( Altezza(u.sx), Altezza(u.dx) ) + 1;
18    RETURN u;
```

```
1  Altezza( u ):
2    IF (u == null) {
3      RETURN -1;
4    } ELSE {
5      RETURN u.altezza;
6    }
```

```
1  NuovaFoglia( e ):
2    u = NuovoNodo();
3    u.dato = e;
4    u.altezza = 0;
5    u.sx = u.dx = null;
6    RETURN u;
```

Dopo la creazione della foglia `f` contenente l'elemento `e` (riga 3), aggiorniamo le altezze ricorsivamente nei campi `u.altezza` degli antenati `u` di `f`, essendo questi ultimi i soli nodi che possono cambiare altezza (riga 17). Allo stesso tempo, controlliamo se il nodo corrente è 1-bilanciato: definiamo **nodo critico** il minimo antenato di `f` che viola tale proprietà.

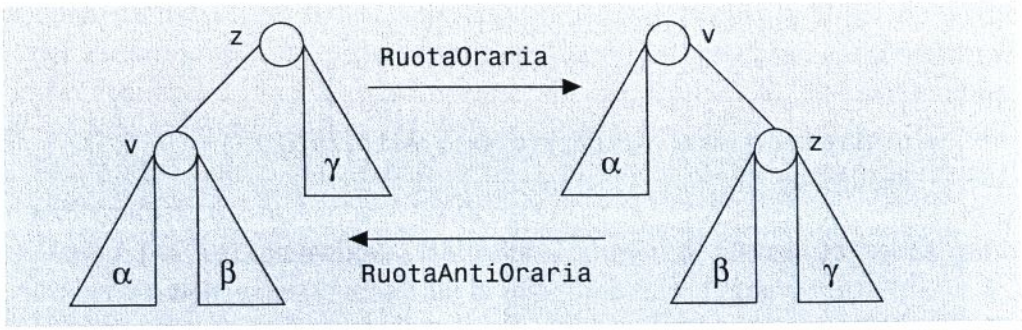
A tal fine, percorriamo in ordine inverso le chiamate ricorsive sugli antenati di `f` (righe 5 e 11), fino a individuare il nodo critico `u`, se esiste: in tal caso, se `f` discende da `u.sx`, l'altezza del sottoalbero sinistro di `u` differisce di due rispetto a quella del destro (riga 6), mentre se `f` discende da `u.dx`, abbiamo che è l'altezza del sottoalbero destro di `u` a differire di due rispetto a quella del sinistro (riga 12). Comunque vada, aggiorniamo l'altezza del nodo prima di terminare la chiamata attuale (riga 17).

Discutiamo ora come ristrutturare l'albero in corrispondenza del nodo critico `u`, utilizzando le **rotazioni** orarie e antiorarie per rendere nuovamente `u` un nodo 1-bilanciato (righe 7-8 e 13-14): tali rotazioni sono illustrate e descritte nel Codice 4.8. Notiamo che esse richiedono tempo `O(1)` e preservano la proprietà di ricerca: le chiavi in α sono minori di `v.dato.chiave`; quest'ultima è minore delle chiavi in β, le quali sono minori di `z.dato.chiave`; infine, quest'ultima è minore delle chiavi in γ.

Utilizziamo le rotazioni per trattare i quattro casi che si possono presentare (individuati con un semplice codice mnemonico), in base alla posizione di `f` rispetto ai nipoti del nodo critico `u` (Figura 4.2):

1. caso SS: la foglia `f` appartiene al sottoalbero α radicato in `u.sx.sx`;
2. caso SD: la foglia `f` appartiene al sottoalbero β radicato in `u.sx.dx`;
3. caso DS: la foglia `f` appartiene al sottoalbero γ radicato in `u.dx.sx`;
4. caso DD: la foglia `f` appartiene al sottoalbero δ radicato in `u.dx.dx`.

AVL Codice 4.8 Rotazioni oraria e antioraria.




```
1  RuotaOraria( z ):
2    v = z.sx;
3    z.sx = v.dx;
4    v.dx = z;
5    z.altezza = max( Altezza(z.sx), Altezza(z.dx) ) + 1;
6    v.altezza = max( Altezza(v.sx), Altezza(v.dx) ) + 1;
7    RETURN v;
```

```
1  RuotaAntiOraria( v ):
2    z = v.dx;
3    v.dx = z.sx;
4    z.sx = v;
5    v.altezza = max( Altezza(v.sx), Altezza(v.dx) ) + 1;
6    z.altezza = max( Altezza(z.sx), Altezza(z.dx) ) + 1;
7    RETURN z;
```

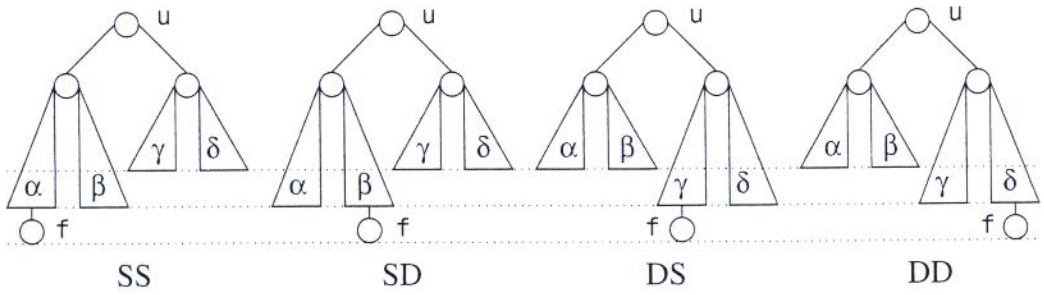


Figura 4.2 I quattro casi possibili di sbilanciamento del nodo critico u a causa della creazione della foglia f (contenente la chiave k).

In tutti e quattro i casi, lo sbilanciamento conduce l'albero AVL in una configurazione in cui due sottoalberi hanno un dislivello pari a due. Con riferimento all'inserimento descritto nel Codice 4.7, trattiamo il caso SS con una rotazione oraria effettuata sul nodo critico u, riportando l'altezza del sottoalbero a quella imme-

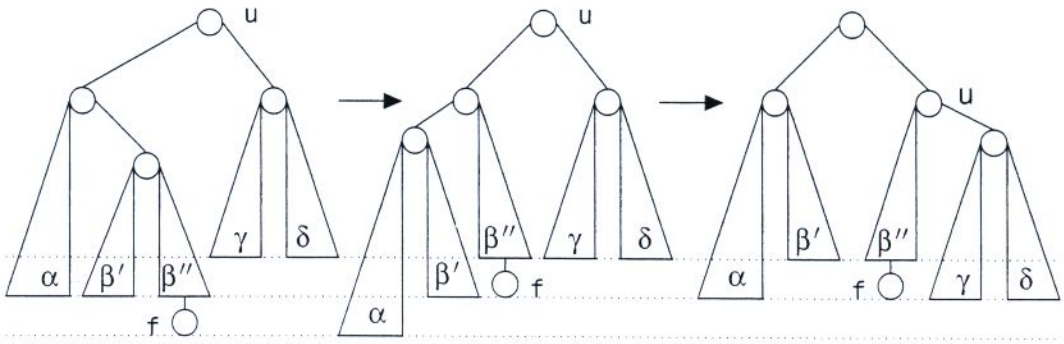
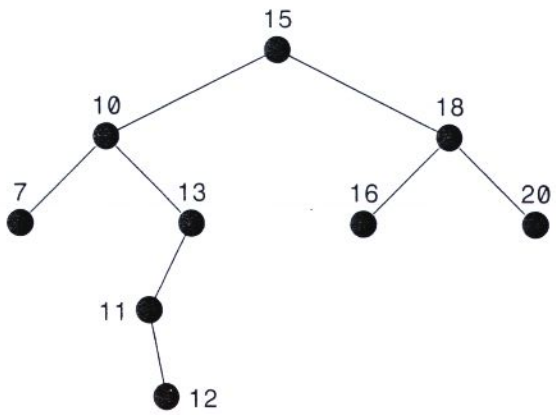


Figura 4.3 Effetto delle rotazioni sul nodo critico u per il caso SD.

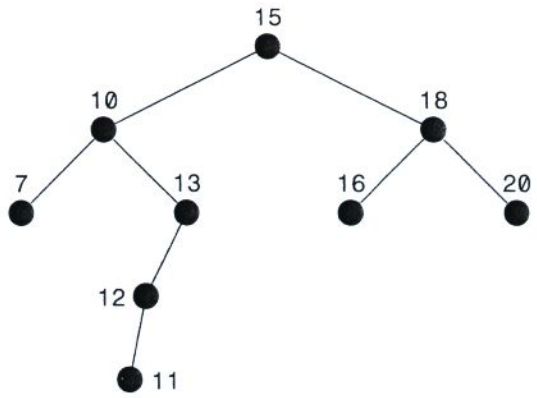
diatamente prima che la foglia `f` venisse creata (riga 8). Se invece incontriamo il caso SD, prima effettuiamo una rotazione antioraria sul figlio sinistro di u (riga 7) e poi una oraria su u stesso (riga 8): anche in tal caso, l'altezza del sottoalbero torna a essere quella immediatamente prima che la foglia `f` venisse creata (Figura 4.3). I casi DD e DS sono speculari: effettuiamo una rotazione antioraria su u (riga 14) oppure una rotazione oraria sul figlio destro di u seguita da una antioraria su u (righe 13 e 14). Siccome ogni caso richiede una o due rotazioni, il costo è tempo $O(1)$ per eseguire le rotazioni (al più due), a cui va aggiunto il tempo di inserimento $O(\log n)$.

ESEMPIO 4.4

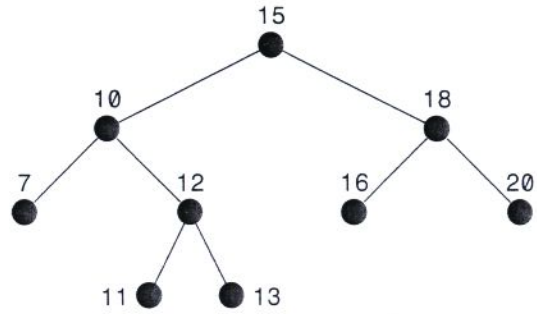
Consideriamo l'inserimento del nodo con chiave 12 nell'ultimo albero mostrato nell'Esempio 5.3 che è un albero AVL. Dopo la prima fase, prima del ribilanciamento, l'albero appare come segue.



I nodi con chiave 12 e 11 risultano essere 1-bilanciati mentre quello con chiave 13 no: quest'ultimo è il nodo critico che chiameremo u. Poiché la chiave inserita è maggiore della chiave del figlio sinistro di u (ovvero 11) viene prima eseguita una rotazione antioraria su figlio sinistro di u (riga 7).



Segue una rotazione oraria su u (riga 8).



Quest'ultima operazione restituisce l'albero 1-bilanciato.

Per la cancellazione, possiamo trattare dei casi simili a quelli dell'inserimento, solo che possono esserci più nodi critici tra gli antenati del nodo cancellato: preferiamo quindi marcare logicamente i nodi cancellati, che vengono ignorati ai fini della ricerca. Quando una frazione costante dei nodi sono marcati come cancellati, ricostruiamo l'albero con le sole chiavi valide ottenendo un costo ammortizzato di `O(log n)`. Possiamo trasformare il costo ammortizzato in un costo al caso pessimo interlacciando la ricostruzione dell'albero, che produce una copia dell'albero attuale, con le operazioni normalmente eseguite su quest'ultimo.

Nonostante siano stati concepiti diverso tempo fa, gli alberi AVL sono tuttora molto competitivi rispetto a strutture di dati più recenti: la maggior parte delle rotazioni avvengono negli ultimi due livelli di nodi, per cui gli alberi AVL risultano molto efficienti se implementati opportunamente (all'atto pratico, la loro forma è molto vicina a quella di un albero completo e quasi perfettamente bilanciato).

4.5 Opus libri: liste invertite e trie

Nei sistemi di recupero dei documenti (**information retrieval**), i dati sono documenti testuali e il loro contenuto è relativamente stabile: pertanto, in tali sistemi lo scopo principale è quello di fornire una risposta veloce alle numerose interrogazioni degli utenti (contrariamente alle basi di dati che sono progettate per garantire