ESERCITAZIONI DI *INTRODUZIONE AGLI ALGORITMI*
(A.A. 08/09)

DISPENSA N. 6

Esercizi su alberi di ricerca e AVL

Notazione: Per un albero T scriviamo $|T|$ per indicare il numero dei nodi di T e $h(T)$ per indicare l'altezza di T. Se A e B sono due alberi binari di ricerca, scriviamo $A \prec B$ se tutte le chiavi in A sono minori di tutte le chiavi in B.

Esercizio 1

(a) Scrivere un algoritmo che prende in input due alberi binari di ricerca T_1 e T_2 tali che $T_1 \prec T_2$ o $T_2 \prec T_1$ e restituisce in output un albero binario di ricerca contenente tutte le chiavi di T_1 e tutte le chiavi di T_2. Stimare la complessità.

(b) Scrivere un algoritmo che prende in input due alberi binari di ricerca T_1 e T_2 qualunque restituisce in output un albero binario di ricerca contenente tutte le chiavi di T_1 e tutte le chiavi di T_2. Stimare la complessità.

(c) Nel caso non l'abbiate già fatto al punto (b), scrivere un algoritmo che risolva il problema al punto (b) con complessità $O(|T_1| + |T_2|)$.

Soluzione parte (a) Basta osservare quanto segue. Se $T_1 \prec T_2$, allora, in particolare, il minimo di T_2 è maggiore di tutte le chiavi di T_1. Se appendiamo la radice di T_1 come figlio sinistro al minimo di T_2 rispettiamo la proprietà di ricerca e abbiamo finito. Il caso $T_2 \prec T_1$ è duale: il minimo di T_1 è maggiore di tutte le chiavi di T_2. Se appendiamo la radice di T_2 come figlio sinistro al minimo di T_1 rispettiamo la proprietà di ricerca e abbiamo finito. Per capire se siamo nel caso $T_1 \prec T_2$ oppure $T_2 \prec T_1$ basta confrontare le radici di T_1 e di T_2 (tempo costante). Una volta capito in che caso siamo, possiamo scegliere l'albero che va sotto, T_{sotto} e l'albero che va sopra T_{sopra}. A questo punto basta trovare il minimo di T_{sopra} (costo $O(h(T_{\text{sopra}}))$) e, una volta trovato, appendervi come figlio sinistro la radice di T_{sotto} (costo costante). Il costo di tutta la procedura è dunque $O(h(T_{\text{sopra}}))$. Nel caso peggiore (albero degenerato in lista), questa complessità è $O(|T_{\text{sopra}}|)$.

Soluzione parte (b) In questo caso non sappiamo nulla delle chiavi di T_1 e T_2. Una soluzione immediata è la seguente: inseriamo i nodi di T_1 uno per uno in T_2. Possiamo scegliere di visitare T_1 con una qualunque delle visite che conosciamo. Visitare T_1 costa $h(T_1)$. Inserire il primo nodo di T_1 in T_2 costa un inserimento

Note preparate da Lorenzo Carlucci, carlucci@di.uniroma1.it.

in T_2, ossia $O(|T_2|)$; inserire il secondo nodo di T_1 costa $O(|T_2|+1)$, etc. Il costo dell'inserimento dei nodi di T_1 in T_2 è dato dalla somma

$$\sum_{i=0}^{|T_1|} |T_2| + i = |T_1| \cdot |T_2| + O(|T_1|^2).$$

Il costo della procedura è dunque quadratico in $|T_1|$ se $|T_1| > |T_2|$ e $O(|T_1| \cdot |T_2|)$ altrimenti.

Adottando un approccio leggermente diverso - che ci tornerà utile nella parte (c) - possiamo scrivere una procedura per visitare i nodi di T_1 e di T_2 in ordine crescente (usando la funzione di successore) e inserirli uno alla volta in un nuovo albero T.

```
fondi(ABR T1, ABR T2) → ABR {
  correnteT1 = minimo(T1)
  correnteT2 = minimo(T2)
  T = nuovo ABR ()
  while (correnteT1 != NULL && correnteT2 != NULL) {
    if (correnteT1.valore < correnteT2.valore)
      insert(correnteT1.valore, T)
      correnteT1 = successore(correnteT1)
    else
      insert(correnteT2.valore, T)
      correnteT2 = successore(correnteT2)
  }
  while (correnteT1 != NULL && correnteT2 == NULL) {
    insert(correnteT1.valore, T)
    correnteT1 = successore(correnteT1)
  }
  while (correnteT1 == NULL && correnteT2 != NULL) {
    insert(correnteT2.valore, T)
    correnteT2 = successore(correnteT2)
  }
}
```

Il costo dominante di questa procedura è dato dagli inserimenti in T. L'albero T è vuoto all'inizio e alla fine contiene $|T_1| + |T_2|$ nodi. Il costo degli inserimenti successivi in T è dato dalla sommatoria seguente.

$$\sum_{i=0}^{|T_1|+|T_2|} i = O((|T_1| + |T_2|)^2).$$

La complessità è quadratica nella somma del numero dei nodi di T_1 e di T_2. Nel caso peggiore la complessità è la stessa della procedura vista prima.

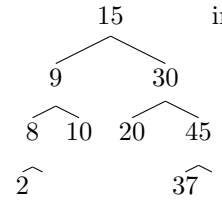
Soluzione parte (c) Per trovare un algoritmo più efficiente per questo problema non possiamo sfruttare la struttura del problema (gli alberi T_1 e T_2 sono qualunque) ma possiamo invece sfruttare la struttura dati in questione, ossia gli alberi binari di ricerca. Come sappiamo, questi hanno una stretta relazione con la ricerca binaria. Proviamo a sfruttare questa relazione. Se visitiamo gli alberi T_1 e

T_2 sfruttando la funzione di successore come abbiamo fatto in (b), ma inseriamo i valori trovati in un vettore anziché in una lista, in $\Theta(|T_1| + |T_2|)$ otteniamo un vettore di lunghezza $|T_1| + |T_2|$ contenente i valori di T_1 e di T_2 in ordine crescente. A questo punto possiamo usare un approccio divide et impera, simile alla ricerca binaria, per costruire un albero binario di ricerca con le chiavi del vettore ordinato. La procedura di ricostruzione prende in input un vettore V e due indici *sin*, *des* che delimitano la porzione del vettore su cui agire (alla prima chiamata saranno 0 e $|T_1| + |T_2| - 1$. Ad ogni passo viene calcolato il punto mediano del vettore. Il valore del punto mediano viene messo nella radice dell'albero, e la procedura viene richiamata ricorsivamente sulla porzione a sinistra e sulla porzione a destra del mediano per costruire i sottoalberi sinistro e destro, rispettivamente.

```
crea(vettore V, indici sin, des) → ABR {
  mediano = (sin+des)/2
  T = nuovo ABR
  T.valore = V[mediano]
  T.sinistro = crea(V,sin,mediano−1)
  T.destro = crea(V,mediano+1,des)
  return T
}
```

Il costo della procedura è dato dalla relazione di ricorrenza $T(n) = 2T(n/2) + O(1)$. Analizzando l'equazione con uno dei metodi noti si trova un costo $O(|T_1| + |T_2|)$. Il costo dell'intera procedura (creazione del vettore ordinato e creazione dell'albero di fusione) è dunque $\Theta(|T_1| + |T_2|)$, come desiderato.

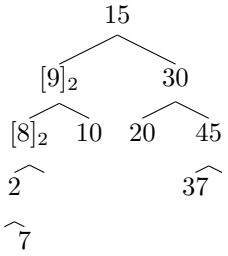
Esercizio 2 Dato l'AVL seguente:



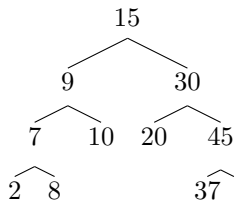
inserire gli elementi $7, 12, 3, 36$ e rimuovere gli elementi $20, 30$. Eseguire le operazioni nell'ordine indicato. Dettagliare i valori di bilanciamento di ogni nodo e le rotazioni effettuate.

Soluzione Indichiamo soltanto i nodi sbilanciati (tra parentesi quadre), e il relativo fattore di sbilanciamento (in pedice).

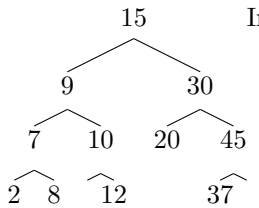
Inserire 7:



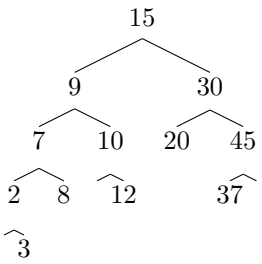
Caso SD (rotazione sinistra in 2, destra in 8):



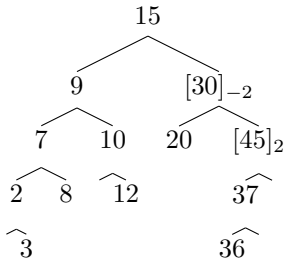
Inserire 12:



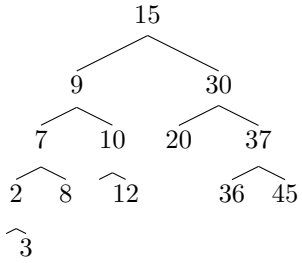
Inserire 3:



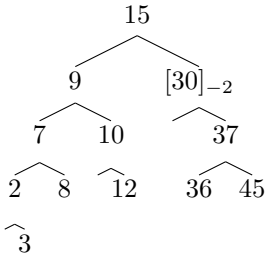
Inserire 36:



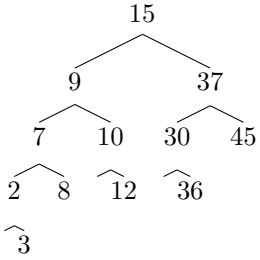
Caso SS (rotazione destra in 45):



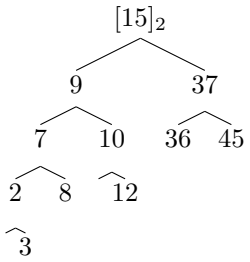
Rimuovi 20:



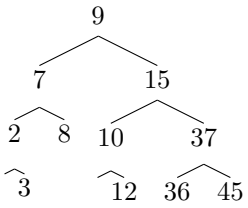
Caso DD (rotazione sinistra in 30):



Rimuovi 30:



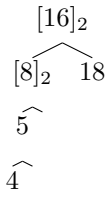
Caso SS (rotazione destra in 15)



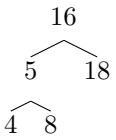
Esercizio 3 Inserire gli elementi $16, 18, 8, 5, 4, 28, 25, 7, 13, 2, 10$ in un AVL.

Soluzione

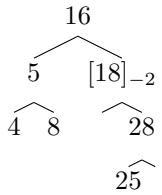
Insert $16, 18, 8, 5, 4$:



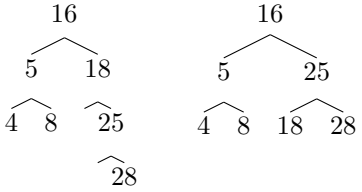
Caso SS (rotazione destra in 8):



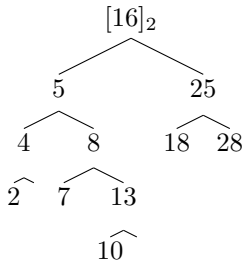
Insert $28, 25$:



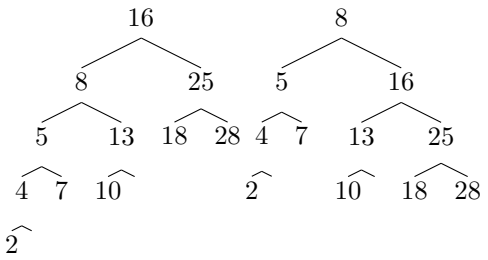
Caso DS (destra in 28, sinistra in 18):



Insert $7, 13, 2, 10$:



Caso SD (sinistra in 5, destra in 16):

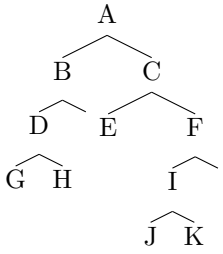


Esercizio 4

(a) Se $GDHBAECJIKF$ è il risultato della visita simmetrica di un albero T e se $ABDGHCEFIJK$ è il risultato della visita in pre-ordine dello stesso albero T, che albero è T?

(b) Scrivere un algoritmo che prenda in input due vettori, v_1, v_2 contenenti rispettivamente il risultato della visita simmetrica di un albero T e il risultato della visita in pre-ordine dello stesso albero T, e che restituisca in output l'albero T.

Soluzione parte (a) L'albero T è il seguente



Soluzione parte (b)(cfr. Esercitazione 6, a.a. 07/08, Dott. Accattoli) L'idea è di usare la sequenza in pre-ordine per individuare la radice, cercarne l'occorrenza nella sequenza simmetrica, ricavare dalla sequenza simmetrica la dimensione del sottoalbero destro e di quello sinistro, usare questa informazione per chiamare ricorsivamente la procedura sulle porzioni a sinistra e a destra della sequenza simmetrica e sulle corrispondenti sezioni della sequenza in pre-ordine. Ad ogni passo, dalla sequenza in pre-ordine ricaviamo l'informazione: *qual è la chiave della radice del sottoalbero corrente?* Dalla sequenza simmetrica ricaviamo l'informazione: *qual è la dimensione del sottoalbero sinistro e del sottoalbero destro del nodo corrente?* Le due informazioni combinate ci permettono di ricostruire univocamente l'albero T.