

# Esercizi di Algoritmi e Strutture Dati

Ultimo aggiornamento: 6 dicembre 2009

Moreno Marzolla  
marzolla@cs.unibo.it

5 dicembre 2009

## 1 Hashing/1

(Esercizio 7.5 del libro di testo) Supponiamo che nell'hashing con liste di collisione, le liste siano mantenute ordinate. Che effetto ha questa modifica sul tempo di esecuzione delle operazioni `search`, `insert` e `delete`?

## 2 Hashing/2

(Problema 7.1 del libro di testo) Proporre una funzione hash perfetta nel caso in cui le chiavi siano stringhe di lunghezza 3 composte dai caratteri appartenenti all'insieme  $\{ 'A', 'B', 'C' \}$ .

## 3 Rotazioni semplici in ABR

Si consideri l'operazione di *rotazione semplice* applicata ad un Albero Binario di Ricerca (ABR). Dimostrare che l'operazione di rotazione semplice, come definita a lezione, preserva la proprietà di ordinamento degli ABR. In altre parole, dimostrare che un ABR, dopo una singola operazione di rotazione semplice rispetto ad un qualsiasi nodo  $x$ , è ancora un ABR.

## 4 Costruzione di ABR

Dimostrare che qualsiasi algoritmo basato su confronti per la costruzione di un ABR con  $n$  nodi ha complessità asintotica  $\Omega(n \log n)$ .

## 5 Visita di un ABR

L'operazione di visita di un ABR con  $n$  nodi può essere implementata determinando l'elemento minimo dell'ABR, e poi invocando  $n - 1$  volte l'operazione `successor()`. Fornire una giustificazione intuitiva del fatto che questo algoritmo di visita abbia complessità asintotica  $\Theta(n)$ .

## 6 Incremento di chiavi in un albero AVL

Si consideri un albero AVL contenente  $n$  chiavi numeriche. Supponiamo che le chiavi siano tutte distinte tra loro. Vogliamo implementare l'operazione `incrementaChiave(k, d)` il cui scopo è quello di incrementare il valore della chiave  $k$  di una quantità  $d$  (che potrebbe anche essere negativa), facendolo diventare  $k + d$ . Al termine di questa operazione, la struttura dati risultante deve ancora essere un albero AVL. Per l'ipotesi di unicità delle chiavi, il nodo contenente la chiave  $k$ , se esiste, è sempre unico; supponiamo anche che il valore  $k + d$  sia unico. Descrivere un algoritmo per realizzare l'operazione `incrementaChiave(k,d)`, stimandone poi il costo computazionale.

## 7 Implementazione di un albero AVL

A lezione abbiamo visto che per una corretta implementazione degli alberi AVL è necessario conoscere l'altezza dei sottoalberi radicati in ciascun nodo. Infatti, questa informazione consente poi di capire quali sono i sottoalberi "pesanti", e quindi procedere alle operazioni di ribilanciamento appropriate. Ricordiamo che l'altezza di un albero è la massima profondità cui si trova una sua foglia. L'albero composto da un singolo nodo (la radice) ha altezza 0.

1. Consideriamo innanzitutto un generico ABR (non bilanciato). Supponiamo che ciascun nodo  $v$  abbia un attributo intero  $v.h$  che corrisponde all'altezza del sottoalbero radicato in  $v$ . Mostrare come sia possibile estendere le operazioni di inserimento e rimozione di nodi di un ABR per mantenere in modo efficiente il valore corretto di  $v.h$  per ciascun nodo. Tale modifica non deve alterare il costo computazionale delle operazioni di inserimento e rimozione, che devono mantenersi  $O(h)$  nel caso pessimo, essendo  $h$  l'altezza totale dell'albero.
2. Consideriamo ancora un generico ABR. Dimostrare come l'operazione di *rotazione semplice* può essere estesa per mantenere il valore corretto di  $v.h$  per ciascun nodo. Dopo tale modifica, il costo dell'operazione di rotazione semplice deve essere  $O(h)$  nel caso pessimo, essendo  $h$  l'altezza totale dell'albero.
3. Usare i due punti precedenti per dimostrare come sia possibile mantenere l'informazione sull'altezza di ciascun sottoalbero in un albero AVL senza alterare il costo computazionale delle operazioni di inserimento e rimozione di nodi.