

Algoritmi e Strutture Dati

22 Giugno 2016

Cognome Nome Matricola

Note

1. La leggibilità è un prerequisito: parti difficili da leggere potranno essere ignorate.
2. Quando si presenta un algoritmo è fondamentale spiegare l'idea sottostante e motivarne la correttezza.
3. L'efficienza è un criterio di valutazione delle soluzioni proposte.
4. Si consegnano tutti i fogli, con nome, cognome, matricola e l'indicazione *bella copia* o *brutta copia*.

Domande

Domanda A (5 punti) Sia data la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 3T(n/5) + T(n/6) + n & \text{se } n > 1 \end{cases}$$

Si fornisca un limite asintotico stretto per la soluzione.

Domanda B (4 punti) Indicare il codice prefisso ottenuto utilizzando l'algoritmo di Huffman per l'alfabeto $\{a, b, c, d, e, f, g\}$, supponendo che ogni simbolo appaia con le seguenti frequenze.

a	b	c	d	e	f	g
6	21	12	8	3	23	8

Spiegare il processo di costruzione del codice.

Domanda C (5 punti) Scrivere una funzione `IsABR(A)` che dato in input un array di interi $A[1..n]$, interpretato come albero binario (come nel caso degli heap, ogni $A[i]$ è un nodo con figlio sinistro e destro $A[2i]$ e $A[2i+1]$) verifica se A è un albero binario di ricerca. Valutarne la complessità.

Esercizi

Esercizio 1 (7 punti) Sia dato un albero i cui nodi contengono una chiave intera $x.key$, oltre ai campi $x.l$, $x.r$ e $x.p$ che rappresentano rispettivamente il figlio sinistro, il figlio destro e il padre. Si definisce *grado di squilibrio* di un nodo il valore assoluto della differenza tra la somma delle chiavi nei nodi foglia del sottoalbero sinistro e la somma delle chiavi dei nodi foglia del sottoalbero destro. Il grado di squilibrio di un albero è il massimo grado di squilibrio dei suoi nodi.

Fornire lo pseudocodice di una funzione `sdegree(T)` che calcola il grado di squilibrio dell'albero T (si possono utilizzare funzioni ricorsive di supporto). Valutare la complessità della funzione.

Esercizio 2 (11 punti) Si supponga di dover pagare una certa somma s . Per farlo si hanno a disposizione le banconote b_1, \dots, b_n ciascuna di valore v_1, \dots, v_n . Si vuole determinare, se esiste, un insieme di banconote b_{i_1}, \dots, b_{i_k} che totalizzi esattamente la somma richiesta e che minimizzi il numero k di banconote utilizzate.

- i. mostrare che vale la proprietà della sottostruttura ottima e fornire una caratterizzazione ricorsiva del costo $c(s', j)$ della soluzione ottima per il sottoproblema di dare una somma pari a s' con le banconote in b_1, \dots, b_j , con $j \leq n$;
- ii. tradurre tale definizione in un algoritmo (bottom up o top down con memoization) che determina il costo della soluzione ottima;
- iii. trasformare l'algoritmo in modo che permetta anche di individuare la soluzione, non solo il suo costo;
- iv. valutare la complessità dell'algoritmo.