

Algoritmi e Strutture Dati

29 Agosto 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 (4 punti) Sia data la seguente equazione di ricorrenza:

$$T(n) = T(n-1) + \log n$$

Si dimostri che $T(n) = O(n \log n)$.

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
37	4	12	6	9	17	8

Spiegare il processo di costruzione del codice.

Domanda C (5 punti) Realizzare una funzione `RevCountingSort(A,B,n,k)` che, dato un array $A[1..n]$ contenente interi nell'intervallo $[0..k]$, restituisce in $B[1..n]$ una sua permutazione ordinata in modo decrescente utilizzando una variante del counting sort. Valutare la complessità.

Esercizi

Esercizio 1 (7 punti) Realizzare una procedura `BST(A)` che dato un array $A[1..n]$ di interi, ordinato in modo crescente, costruisce un albero binario di ricerca di altezza minima che contiene gli elementi di A e ne restituisce la radice. Per allocare un nuovo nodo dell'albero si utilizzi una funzione `mknod(k)` che dato un intero k ritorna un nuovo nodo con `x.key=k` e figlio destro e sinistro `x.left = x.right = nil`. Valutarne la complessità.

Esercizio 2 (11 punti) Si supponga di avere una scacchiera $n \times n$. Si vuole spostare un pezzo dall'angolo in basso a sinistra $(1,1)$ a quello in alto a destra (n,n) .

Il pezzo può muoversi di una casella verso l'alto (\uparrow) o verso destra (\rightarrow). Un passo dalla casella (i,j) ha un costo $u(i,j)$ se verso l'alto e $r(i,j)$ se verso destra. Realizzare un algoritmo `MinPath(u,r,n)` che dati in input gli array $u[1..n,1..n]$ e $r[1..n,1..n]$ dei costi dei singoli passi fornisce il cammino minimo. Più in dettaglio:

- i. fornire una caratterizzazione ricorsiva del costo minimo di un cammino $C(i,j)$ per andare dalla casella (i,j) alla casella (n,n)
- ii. tradurre tale definizione in un algoritmo `MinPath(u,r,n)` (bottom up o top down con memoization) che determina il costo di un cammino minimo da $(1,1)$ a (n,n)
- iii. trasformare l'algoritmo in modo che stampi la sequenza di passi di costo minimo;
- iv. valutare la complessità dell'algoritmo.

