Esercizi di Riepilogo

ASD 2008-2009 - I Quadrimestre

Sequenze

1. Dimostrare che non esiste un programma `TerminaZero`, che, preso in input un programma A, restituisce (in tempo finito) un valore di verità per indicare che A termina o meno quando viene eseguito con input 0.

2. Progettare un algoritmo ricorsivo per generare tutti i sottoinsiemi di taglia n ottenibili da un insieme di m elementi, in cui il numero delle chiamate ricorsive effettuate é proporzionale al numero dei sottoinsiemi generati.

3. Progettare un algoritmo di costo in tempo $O(\log n)$ per il calcolo dell'n-simo numero di Fibonacci F_n.

4. É data una sequenza a di n interi non necessariamente distinti. Diciamo che un elemento é di maggioranza se appare in a almeno $\lceil n/2 \rceil$ volte.

   i) Progettare un algoritmo di complessità $\Theta(n^2)$ per stabilire se a contiene un elemento di maggioranza e, in caso affermativo, lo restituisca.

   ii) Progettare un algoritmo di complessità $\Theta(n \log n)$ per lo stesso problema.

5. Dati due array a e b, di n ed m interi distinti, costruire l'array c che rappresenta l'insieme intersezione. Discutere la complessità dell'algoritmo proposto.

6. Scrivere una procedura che fa il MergeSort per $n > k$ e InsertionSort per $n \leq k$. Si assuma che $n = 2^k$. Calcolare la complessità dell'algoritmo proposto.

7. Scrivere un algoritmo di tipo divide et impera che, dato un array a **ordinato** di interi **distinti** (anche negativi), verifichi se esiste un indice i tale che $a[i] = i$. Analizzare la complessità dell'algoritmo proposto.

8. Dato l'insieme $A = \{2, 6, 4, 3, 5\}$ di somma $2s = 20$, simulare l'algoritmo di programmazione dinamica `Partizione`, mostrando il contenuto della tabella che l'algoritmo riempie dinamicamente e la soluzione trovata dall'algoritmo.

9. Progettare un algoritmo di ordinamento che si comporti come MergeSort, ma che divida ricorsivamente l'array in tre parti anzicché in due. Scrivere lo pseudocodice della nuova procedura `MergeSort3`. Descrivere a parole la nuova procedura `Fusione3`. Impostare e risolvere l'equazione di ricorrenza associata.

Alberi binari

1. Dato un albero binario i cui nodi sono colorati di *rosso* e di *nero*, progettare un algoritmo efficiente che calcoli il numero di nodi aventi lo stesso numero di discendenti rossi e neri (Nota: un nodo é discendente di se stesso).

2. Dato un array a di n elementi, progettare un algoritmo che costruisca ricorsivamente in tempo $O(n)$ un albero binario bilanciato tale che $a[i]$ sia l'$(i+1)$-esimo campo $u.dato$ in ordine di visita anticipata. Considerare anche gli algoritmi per le altre visite.

3. Un nodo u di un albero binario é detto **centrale** se la somma delle chiavi contenute nei nodi del sottoalbero di cui u é radice é uguale alla somma delle chiavi contenute nei nodi sul percorso che collega u alla radice dell'albero (u escluso).

   i) Progettare un algoritmo efficiente che, dato un albero binario A, stampi tutti i nodi centrali.

   ii) Discutere la complessità della soluzione trovata.

4. Dato un albero binario T di n nodi, progettare un algoritmo efficiente che restituisce *true* se per ogni nodo u vale la seguente proprietà: il sottoalbero sinistro di u ha una dimensione almeno doppia di quella del sottoalbero destro di u (la dimensione di un sottoalbero é il numero di nodi in esso contenuti). Discutere la soluzione proposta.

5. Dato un albero binario con chiavi intere positive e negative, progettare un algoritmo efficiente che restituisca la somma totale di tutte le chiavi dell'albero. Definire un secondo algoritmo che restituisca il massimo peso di tutti i sottoalberi , dove il peso di un sottoalbero é la somma di tutte le chiavi dei suoi nodi. Valutare la complessità dei due algoritmi.