Laurea in "Informatica"

Corso di "Algoritmi e Strutture di Dati"

7 Giugno 2011

1. *Tempo disponibile 180 minuti. È ammesso ritirarsi entro 90 minuti.*
2. *Sono ammessi al più 3 scritti consegnati per l'A.A. 2010/11 (Giugno 2011-Febbraio 2012)*
3. *Non è possibile consultare appunti, libri o persone, né uscire dall'aula.*
4. *Le soluzioni degli esercizi devono:*
   - *spiegare a parole l'algoritmo usato (anche con l'aiuto di eventuali disegni)*
   - *fornire e commentare lo pseudocodice (dettagliando a parole il significato delle variabili)*
   - *giustificare la correttezza e la complessità (con tutti i passaggi matematici necessari)*
5. *Un esercizio può ammettere più soluzioni: a soluzioni più efficienti sono assegnati punteggi maggiori*

1. Si calcoli la complessità $T(n)$ della seguente procedura:

```
integer mistery(integer[ ] A, integer n)
    integer k ← 0
    for integer i ← 1 to n do
        k ← k + A[i]
    if n < 13 then
        return k
    else
        return 4 · mistery(A, ⌊n/2⌋) + k
```

2. Sia dato un albero binario T realizzato con puntatori, in cui ogni nodo contiene un valore intero. Scrivere lo pseudocodice di una procedura di *complessità ottima* che modifichi T cancellando ogni foglia che ha un fratello e contiene un valore uguale a quello del fratello.

3. Dati due vettori A e B (non ordinati), ciascuno dei quali contiene n elementi interi positivi distinti, si vuole determinare se esiste un elemento che compare sia in A che in B. Fornire un algoritmo efficiente, scriverne lo pseudocodice e discuterne la complessità (nota: ne esiste uno che richiede tempo O(n) nel caso medio).

4. Si scriva lo pseudocodice della procedura *Depth-First-Search (DFS)* vista a lezione. Si esegua la procedura *DFS* sul grafo *non orientato* $G = (N, A)$, $N = \{1, 2, 3, 4, 5\}$, $A = \{[1,2], [1,3], [1,5], [2,5], [3,2], [3,4]\}$ a partire dal nodo 2, assumendo che gli insiemi di adiacenza siano ordinati in modo *crescente*, mostrando l'ordine di visita dei nodi e degli archi.

5. Sia G un grafo *non orientato, connesso e pesato* del quale si conosce un minimo albero di copertura T. Si decrementi di 1 il peso di un arco di G e si consideri il problema di determinare, dato T, un minimo albero di copertura del grafo modificato. Si definisca (a parole) un algoritmo efficiente valutandone correttezza e complessità.

6. Dato un vettore V contenente n elementi interi *positivi e distinti*, si definiscano i seguenti due problemi computazionali:

- *min-gap* (differenza minima): determinare una coppia di indici $1 \le i < j \le n$ tali che $|V[i] - V[j]| \le |V[k] - V[h]|$ per ogni $1 \le k < h \le n$
- *max-gap* (differenza massima): determinare una coppia di indici $1 \le i < j \le n$ tali che |

$|V[i] - V[j]| \geq |V[k] - V[h]|$ per ogni $1 \leq k < h \leq n$

Proporre algoritmi efficienti per risolvere i due problemi, scriverne lo pseudocodice e discuterne la complessità.