Laurea in "Informatica"

Corso di "Algoritmi e Strutture di Dati"

20 Settembre 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 ricorsiva e si determini se tale complessità sia polinomiale o esponenziale nella dimensione dell'input:

```
integer mistery (integer n)
    integer k ¬ n/2
    for integer i ¬ n downto n − 2 do
        |_ k ¬ k + n · i
    if n < 31 then
        | return k
    else
        |_ return 3 · mistery (⌊n/2⌋) + 2 · mistery (⌊n/2⌋) + k
```

2. Data una lista L di n interi, si vuole togliere da L ogni elemento che compare più di una volta, mantenendo l'ordine originario degli elementi (p.e. se L = 2, 5, 1, 4, 5, 7, 4, 5 allora il risultato è L = 2, 1, 7). Si scriva lo pseudo-codice di una procedura di complessità polinomiale utilizzando gli operatori delle liste visti a lezione.

3. Si riconsideri il problema dell'Esercizio 2. Utilizzando opportune strutture di dati, si descrivano due algoritmi che lo risolvano, rispettivamente, in tempo:

- O(n log n) nel caso pessimo;
- O(n) nel caso medio.

4. Si scriva lo pseudo-codice della procedura *Breadth-First-Search (BFS)* vista a lezione. Si esegua la procedura *BFS* 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 4, assumendo che gli insiemi di adiacenza siano ordinati in modo *crescente*, mostrando l'ordine di visita dei nodi e degli archi.

5. Un esame scritto contiene n esercizi, ognuno dei quali vale $p[i]$ punti e richiede $t[i]$ minuti per essere risolto, ed è superato raggiungendo la sufficienza S. Si vuole trovare il sottoinsieme di esercizi che permette di passare l'esame nel minor tempo possibile.

- Posto $T(i; v)$ il numero minimo di minuti richiesti per ottenere almeno v punti risolvendo un qualunque sottoinsieme dei primi i esercizi, si scriva un'espressione ricorsiva per $T(i; v)$;
- Si progetti un algoritmo basato sulla programmazione dinamica e se ne valuti la complessità.

6. Dato un insieme A, di n interi distinti, si vuole decidere se esistono due sottoinsiemi

disgiunti S e T di A tali che il doppio del prodotto degli elementi in S sia uguale a tre volte la somma degli elementi in T. Scrivere lo pseudo-codice di un algoritmo non deterministico che richieda tempo polinomiale.