Laurea in "Informatica"
Corso di "Algoritmi e Strutture di Dati"
17 Gennaio 2012

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 computazionalmente più efficienti e/o concettualmente più semplici sono assegnati punteggi maggiori*

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

```
integer mistery(integer [ ] A, integer n)
    integer k ← ⌊n/2⌋
    for integer i ← n downto ⌊n/2⌋ do
        ⌊ k ← k·A[i]
    if n < 17 then
        | return k
    else
        ⌊ return k · mistery(A, ⌊n/2⌋) + ⌊n/2⌋ · mistery(A, ⌊n/2⌋)
```

2. Progettare un algoritmo divide-et-impera che, preso un vettore ordinato A di n interi relativi distinti, determini in tempo $O(\log n)$ se esiste un *punto fisso*, ovvero un indice i tale che $A[i] = i$.

3. In un albero binario T, *l'altezza* di un nodo v è la massima distanza (cioè il numero di archi) tra v e una foglia del sottoalbero radicato in v (pertanto, l'altezza di una foglia di T è zero, mentre l'altezza della radice di T è pari alla profondità dell'albero). Si scriva lo pseudo-codice di un algoritmo che, dati un albero binario T di n nodi ed un intero k, calcola in tempo $O(n)$ quanti nodi di T hanno altezza k, assumendo T realizzato con puntatori.

4. Si scriva lo pseudo-codice della procedura *Depth-First-Search (DFS)* vista a lezione. Si esegua la procedura *DFS* sul grafo *non orientato* $G = (N, A)$, $N = \{1, 2, 3, 4, 5\}$, $A = \{[1,3], [1,4], [1,5], [2,3], [3,4], [4,5]\}$ 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. Si scriva lo pseudo-codice dell'algoritmo di Kruskal visto a lezione per determinare il minimo albero di copertura di un grafo orientato pesato e se ne discuta la complessità. Successivamente, si esegua (a mano) l'algoritmo sul grafo dell'Esercizio 4, assumendo che i pesi degli archi siano $w[1,3]=1$, $w[1,4]=3$, $w[1,5]=4$, $w[2,3]=6$, $w[3,4]=5$, $w[4,5]=2$, e mostrando ad ogni passo il contenuto delle strutture di dati usate.

6. Dato un insieme A di n interi distinti, si vuole decidere se esistono due sottoinsiemi non vuoti S e T di A tali che il prodotto degli elementi in S sia uguale al doppio della somma degli elementi in T. Scrivere lo pseudo-codice di un algoritmo non deterministico che richieda tempo polinomiale.