Laurea in "Informatica"
Corso di "Algoritmi e Strutture di Dati"
18 Febbraio 2014

1. *Tempo disponibile 180 minuti.*
2. *Non è possibile consultare appunti, libri o persone, né uscire dall'aula.*
3. *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)*
4. *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, con $n > 0$, specificando se tale complessità è polinomiale o esponenziale nella *dimensione* dell'input:

```
integer mistery(integer n)
    integer k ← 6
    for integer i ← n + 2 downto n − 1 do
        ⌊ k ← k − 1
    if n > 13 then
        | return ⌊n/k⌋ · mistery(⌊n/k⌋) + k · mistery(⌊n/2⌋) + mistery(2)
    else
        ⌊ return ⌊n/k⌋
```

2. Data una lista L di interi, si vuole togliere da L ogni elemento pari e inserirlo in una nuova lista M, mantenendo in entrambe le liste l'ordine originario degli elementi (p.e. se in input $L = 2, 5, 1, 4, 5, 7, 4$, allora in output $L = 5, 1, 5, 7$ e $M = 2, 4, 4$). Si scriva lo pseudo-codice di un algoritmo che abbia tempo *ottimo* utilizzando gli *operatori* per le liste.

3. Dato un albero binario T i cui nodi contengono valori interi, si vuole cancellare ogni *foglia* che è figlio sinistro e contiene un valore pari. Si scriva lo pseudo-codice di un algoritmo che abbia tempo *ottimo*, assumendo T realizzato con *puntatori*.

4. Dato un vettore di n interi, si vuole trovare in tempo *ottimo* il valore massimo. Si scriva lo pseudo-codice di un algoritmo di tipo *divide et impera* che ad ogni ricorsione divida il vettore in *tre parti bilanciate*, impostando e risolvendo l'equazione di ricorrenza della complessità (per semplicità si può assumere che n sia una potenza di tre).

5. Si scriva la procedura HEAPSORT vista a lezione e la si esegua (a mano) per ordinare i 10 elementi: 9, 10, 3, 5, 8, 2, 6, 4, 7, 1. Si illustri con disegni, passo dopo passo, il contenuto dello heap durante l'esecuzione.

6. In un grafo non orientato $G = (N, A)$, un *insieme indipendente* è un sottoinsieme S di nodi tale che per nessuna coppia di nodi i e j di S esiste un arco $[i, j]$ nel grafo G. Si scriva lo pseudo-codice di un algoritmo *non deterministico* di complessità polinomiale che, dati in input un intero k ed un grafo G rappresentato con *matrice di adiacenza* nodi-nodi, decide se esiste un insieme indipendente di cardinalità maggiore o uguale a k.