Laurea in "Informatica"
Corso di "Algoritmi e Strutture Dati"
19 Luglio 2006

1. *Tempo disponibile 180 minuti. È ammesso ritirarsi entro 90 minuti.*
2. *Sono ammessi al più 3 scritti consegnati tra Giugno 2006 e Febbraio 2007.*
3. *Non è possibile consultare appunti, libri o persone, né uscire dall'aula.*
4. *Per raggiungere la sufficienza occorrono* almeno *3 esercizi risolti senza alcun errore.*
5. *Le soluzioni degli esercizi devono:*
   a. *spiegare a parole l'algoritmo usato (anche con eventuali disegni)*
   b. *commentare l'eventuale procedura Pascal (dettagliando il significato delle variabili)*
   c. *giustificare la correttezza e tutti i passaggi matematici*
   d. *dimostrare la complessità (con equazioni di ricorrenza se necessario)*

1. Si valuti la complessità $T(n)$ della seguente funzione Pascal:

```
function PIPPO(n: integer): integer;
  var i, j: integer;
  begin
     for  j := 2  to  4  do i := j;
     if  n < 10
        then PIPPO := 2
        else  if  n > 33
                 then  PIPPO :=  4*PIPPO(n div i) + n*i
                 else   PIPPO :=  4*PIPPO(n - i) + 5*i;
  end;
```

2. Siano dati un albero binario A implementato con puntatori, in cui ogni nodo contiene un elemento intero, ed un intero k. Scrivere una funzione Pascal di complessità ottima per contare il numero di nodi di A che contengono un elemento maggiore di k.

3. Data una lista L di interi, si vuole riordinarla in modo che tutti gli elementi dispari precedano, nello stesso ordine che avevano inizialmente in L, tutti gli elementi pari (p.e., se in ingresso L = 3, 7, 8, 1, 4, allora si ottiene in uscita L = 3, 7, 1, 8, 4). Si scriva una procedura Pascal di *complessità ottima* utilizzando gli operatori per le liste visti a lezione.

4. Si scriva la procedura Pascal COMPONENTICONNESSE per determinare le componenti connesse di un grafo non orientato e la si esegua sul grafo $G = (N, A)$, dove N = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11} e A = {[1,4], [1,5], [2,3], [2,6], [3,4], [4,5], [5,6], [7,8], [7,9], [8,9], [10,11]}. Si assuma che i vettori di adiacenza siano ordinati in modo crescente e se ne mostri il contenuto.

5. Si scriva la procedura Pascal INSERTIONSORT vista a lezione per ordinare gli n elementi A[1],...,A[n] di un vettore e se ne dimostri la complessità.

6. Dato un insieme A di n interi distinti, si vuole decidere se esiste un sottoinsieme S di A tale che il prodotto degli elementi in S sia uguale alla somma degli elementi in $A - S$. Scrivere (in pseudo-codice) un algoritmo non deterministico che richieda tempo polinomiale.