Algoritmi e Strutture Dati
16 Giugno 2003

- *Tempo disponibile 180 minuti (è ammesso ritirarsi entro 90 minuti)*
- *Sono ammessi al più 3 scritti consegnati per A.A.*
- *Non è possibile consultare appunti, libri o persone, né uscire dall'aula*
- *Le soluzioni degli esercizi devono:*
  - *Spiegare a parole l'algoritmo usato (anche con eventuali disegni)*
  - *commentare l'eventuale procedura Pascal (dettagliando il significato delle variabili)*
  - *giustificare la correttezza e tutti i passaggi matematici*
  - *dimostrare la complessità (con equazioni di ricorrenza se necessario)*
  - *Nella valutazione dello scritto ogni esercizio conta 6 punti (e quindi si raggiunge 18 con 3 esercizi risolti correttamente e 30 con 5 esercizi risolti correttamente)*

1. Si valuti l'ordine di grandezza della complessità $T(n)$ della seguente funzione Pascal:

```
function PIPPO(n: integer): integer;
  var i, j: integer;
  begin
      for  j := 2  to  n div 4  do i := j;
      if n ≤ 10 then
          PIPPO  :=  2
      else if n > 313 then
          PIPPO  :=  5 * PIPPO(i) + n*i
      else
          PIPPO  :=  7 * PIPPO(n–2) + 5*i
  end;
```

2. Data una lista L di interi, la si vuole riordinare 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.

3. Siano dati un intero k ed un albero binario i cui nodi contengono interi positivi. Si vuole aggiungere un figlio destro, che contenga un intero arbitrario, ad ogni foglia per la quale la *somma* degli interi contenuti nei nodi del percorso dalla radice alla foglia sia uguale a k. Si scriva una procedura Pascal ricorsiva di complessità *ottima* assumendo l'albero *realizzato con puntatori*.

4. Modificando una visita *DFS*, si scriva una funzione Pascal di complessità *ottima* per verificare se un grafo orientato è *aciclico* oppure no. La si esegua (a mano) sul grafo $G = (N, A)$, dove $N = \{1, 2, 3, 4\}$, $A = \{(1,4), (2,1), (2,4), (3,1), (3,4)\}$, dopo aver mostrato la memorizzazione di G con vettori di adiacenza.

5. Dati un insieme M di n "mazzette" ed un insieme P di n "politici" appartenenti al "Partito Popolare dei Lavoratori" (PPL) oppure al "Partito Cristiano Democratico" (PCD), si vogliono assegnare tutte le mazzette a tutti i politici in modo che il PPL riceva complessivamente il doppio del PCD. Si progetti un algoritmo non deterministico di complessità polinomiale.

6. Si progettino un algoritmo euristico di tipo *greedy* ed uno di *ricerca locale* per risolvere il problema della BISEZIONE in forma di ottimizzazione: "Dato un grafo non orientato $G = (N, A)$ di n nodi, con n pari, trovare una partizione di N in due sottoinsiemi N_0 ed N_1 di $n/2$ nodi ciascuno tale che il *numero di archi* con un estremo in N_0 e l'altro estremo in N_1 sia *minimo*."