FI2 (12 CFU) - Compito di esame del 19 gennaio 2012 – a.a. 2011-12, *tempo a disposizione: 90 minuti*

Problema 1

Si considerino i seguenti metodi:

```
public static int[] cip(int[] v, int n) {
      if(n < v.length) return cucu(v);
      v = ciop(v);
      v = ciop(v);
      return cip (v, n);
}

private static int[] ciop(int[] v) {
      int[] w = new int[2*v.length];
      int i = 0, j = v.length - 1, k = 0;
      while(i < v.length) {
              w[k++] = v[i++];
              w[k++] = v[j--];
      }
      return w;
}

private static int[] cucu(int[] v) {
      int[] w = new int[1];
      w[0] = cucu(v, 0);
      return w;
}

private static int cucu(int[] v, int i) {
      if(i >= v.length) return 0;
      int r = 0;
      if(i < v.length) r += v[i];
      return r + cucu(v, i+1);
}
```

(a) Calcolare, motivandolo adeguatamente, il costo computazionale del metodo `cip(int[] v, int n)` in funzione della dimensione dell'array `v` e del valore `n`.

(b) Esprimere il costo calcolato al punto precedente in funzione della dimensione dell'input.

(c) Cosa restituisce `cip(a, 10)` quando l'array `a` ha solo una cella, contenente il valore `1`?.

Problema 2

(a) Definire una interfaccia Java `Dict` che rappresenti il tipo astratto *dizionario* e definire una classe Java `BinTreeDict` (e le eventuali altri classi necessarie) che implementi `Dict` basandosi su un albero binario di ricerca. Supporre intere le chiavi e includere nelle classi solo variabili, costruttori e firme dei metodi.

(b) Realizzare il metodo della classe `BinTreeDict` per l'inserimento di una nuova chiave nel dizionario. Calcolarne upper e lower bound.

(c) Realizzare il metodo della classe `BinTreeDict` per cercare nel dizionario tutte le chiavi uguali a una data. Calcolarne upper e lower bound.

Problema 3

(a) Si consideri un grafo nondiretto G i cui nodi, oltre ad avere un identificativo numerico, abbiano anche una etichetta info (stringa). Si scriva lo pseudocodice di un algoritmo che, preso in input una stringa S, cerchi nel grafo se esiste un nodo il cui campo INFO é pari ad S, stampando il corrispondente identificativo (il primo che si trova).

(b) Si supponga ora che gli archi del grafo G siano etichettati in modo da distinguere gli archi TREE appartenenti ad un suo spanning tree (gli altri hanno etichetta NONTREE). Si risolva, in questa nuova situazione, lo stesso problema dato al punto (a), ossia cercare un nodo con una certa etichetta S. [Con qualche espediente, es., numerazione delle righe, ci si puó limitare a scrivere le variazioni apportate all'algoritmo.]

(c) Si supponga di dover risolvere ripetutamente, moltissime volte, il problema dato al punto (a) sullo stesso grafo. Organizzare una struttura dati per risolvere questo problema in modo piú efficiente, fornendo la pseudocodifica.