FI2 (12 CFU), ASD (5 CFU)
Compito di esame del 12-4-2013 – a.a. 2012-13 (appello straordinario)
tempo a disposizione: 90 minuti

Problema 1

Si considerino i seguenti metodi Java

```
static int[] recurse(int[] a) {
    return recurse(a, 0, a.length-1);
}

static int[] recurse(int[] a, int i, int j) {
    int[] r = new int[2];
    if(i == j) r[0] = r[1] = a[i];
    else {
        int k = (i + j) / 2;
        int[] s = recurse(a, i, k);
        int[] t = recurse(a, k+1, j);
        join(r, s, t);
    }
    return r;
}

static void join(int[] r, int[] s, int[] t) {
    if(s[1] > t[1]) r[1] = s[1]; else r[1] = t[1];
    if(s[0] < t[0]) r[0] = s[0]; else r[0] = t[0];
}
```

(a) Calcolare, motivandolo adeguatamente, il costo computazionale del metodo `recurse(int[])`, in funzione della dimensione dell'input.

(b) Quale array verrà restituito dal metodo `recurse(int[])` se questo riceve in input l'array `{0, 1, -1, -2, 2, 3, -3, -4, 5, 6, 7, 8, -9}`?

(c) Descrivere un algoritmo non ricorsivo che calcoli la stessa funzione di `recurse(int[])`.

Problema 2

Con riferimento agli alberi binari i cui nodi contengono un campo `int` (inizializzato a 0), risolvere i seguenti punti:

(a) Definire due classi Java, `BinTree` e `BinaryNode`, per rappresentare alberi della tipologia specificata. Nelle classi specificare le variabili membro e le firme di costruttori e metodi.

(b) Definire un metodo Java `fromRoot()` della classe `BinTree` che, visitando `this` albero, assegni al campo intero di ciascun nodo la sua distanza dalla radice. Determinare il costo computazionale del metodo.

(c) Definire un metodo Java `toFurthestLeaf()` della classe `BinTree` che, visitando `this` albero, assegni al campo intero di ciascun nodo la sua distanza dalla foglia (nel suo sottoalbero) più lontana. Determinare il costo computazionale del metodo.

Problema 3

Due grafi semplici G_1 e G_2 sono definiti sullo stesso insieme di vertici V: $G_1 = (V, E_1)$ e $G_2 = (V, E_2)$.

(a) Definire una rappresentazione per G_1 e G_2 idonea a risolvere i punti successivi.

(b) Progettare un algoritmo (pseudo-codice oppure Java) che, dati $G_1 = (V, E_1)$ e $G_2 = (V, E_2)$, restituisca il numero di componenti connesse di $G = (V, E_1 \cup E_2)$. Determinare il costo computazionale dell'algoritmo.

(c) Progettare un algoritmo (pseudo-codice oppure Java) che, dati $G_1 = (V, E_1)$ e $G_2 = (V, E_2)$, restituisca il numero di componenti connesse di $G = (V, E_1 \cap E_2)$. Determinare il costo computazionale dell'algoritmo.