FI2 (12 CFU), ASD (5 CFU)
Compito di esame dell'11-6-2013 – a.a. 2012-13
tempo a disposizione: 90 minuti

Problema 1

Si considerino i seguenti metodi Java

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

static int recurse(int[] a, int k, int l, int d) {
  // assumere 0<=k<=a.length
  if((k == 0) && (d < 0)) return l;
  if((k == a.length - 1) && (d > 0)) return l;
  int p = a[k];
  k += d;
  if(a[k] != p) return l;
  else return recurse(a, k, l+1, d);
}
```

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

(b) Quale valore verrà restituito dalla chiamata
`recurse({4 5 2 2 2 3 4 3 4 5 5 5 6 6 6}, 11)` ?

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

Problema 2

Con riferimento agli alberi binari i cui nodi contengono una chiave `int` e un ulteriore 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 `maxInPath()` della classe `BinTree` che, visitando `this` albero, assegni al campo intero di ciascun nodo v la massima chiave presente nel percorso dalla radice al nodo v (inclusi gli estremi). Determinare il costo computazionale del metodo.

(c) Definire un metodo Java `subtreeSize()` della classe `BinTree` che, visitando `this` albero, assegni al campo intero di ciascun nodo v il numero di nodi appartenenti al sottoalbero di radice v. Determinare il costo computazionale del metodo.

Problema 3

Si intende rappresentare la relazione "follower" (Twitter) attraverso un grafo orientato, i cui nodi rappresentanto gli utenti e gli archi rappresentano la relazione, come di seguito precisato: esiste un arco (u, v) se e solo se l'utente u è follower dell'utente v.

(a) Definire una rappresentazione per il grafo G di cui sopra, idonea a risolvere i punti successivi.

(b) Progettare un algoritmo (pseudo-codice oppure Java) che, dati G e un utente u, determini l'utente "followed" da u avente il maggior numero di follower. In caso di più utenti che soddisfano la condizione, restituirne l'elenco completo. Determinare il costo computazionale dell'algoritmo.

(c) Progettare un algoritmo (pseudo-codice oppure Java) che, dati un grafo G e due utenti u e v, restituisca l'elenco degli utenti che seguono (follow) sia u che v (assumere che un utente non segue se stesso). Determinare il costo computazionale dell'algoritmo.