FI2 (12 CFU), ASD Nettuno (6 CFU) - Compito di esame del 18-6-2012 – a.a. 2011-12
tempo a disposizione: 90 minuti

Problema 1

Si considerino i seguenti metodi:

```
static void m1(int[] a) {
    m1(a, 0);
}

static void m1(int[] a, int i) {
    if(i == a.length - 1) return;
    m1(a, i+1);
    m2(a, i);
}

static void m2(int[] a, int i) {
    if((i == a.length-1)||(a[i] <= a[i+1])) return;
    int e = a[i];
    a[i] = a[i+1];
    a[i+1] = e;
    m2(a, i+1);
}
```

(a) Calcolare, motivandolo adeguatamente, il costo computazionale del metodo `m2(int[], int)` in funzione della dimensione dell'input.

(b) Calcolare, motivandolo adeguatamente, il costo computazionale del metodo `m1(int[])` in funzione della dimensione dell'input.

(c) Chiarire se gli algoritmi `m1` e `m2` lavorano *in-place* o richiedo l'uso di memoria extra di dimensione non costante.

Problema 2

Con riferimento agli alberi binari di ricerca, i cui nodi contengono una chiave `int`, risolvere i punti seguenti.

(a) Definire due classi Java, `BST` e `BSTNode`, per rappresentare alberi binari di ricerca e per risolvere i successivi punti. Le classi debbono specificare variabili membro e costruttori; per quanto riguarda i metodi, sarà sufficiente l'indicazione delle loro firme.

(b) Dati un BST e un intervallo di chiavi $I = [\mathtt{a}, \mathtt{b}]$, definire un metodo Java `void potatura(int a, int b)` che rimuova da `this` BST tutti i nodi contenenti chiavi esterne all'intervallo I. Determinare il costo computazionale del metodo.

(c) Dato un BST in cui possono essere presenti chiavi duplicate, definire un metodo Java `int dict2map()` che rimuova da `this` BST tutti i nodi contenenti chiavi duplicate e restituisca il numero di nodi rimossi.
[N.B. Fra tutti i nodi contenenti la stessa chiave, scegliere arbitrariamente quale sia quello da non rimuovere.]

Problema 3

La nuova rete sociale ReSo utilizza un grafo semplice per rappresentare le relazioni di amicizia fra i suoi partecipanti. Il grafo è *sparso*, ovvero il numero degli spigoli m è lineare nel numero di vertici n: $m \in O(n)$. Ciascun vertice contiene il `nickname` (unico) dell'utente, la sua locazione istantanea (coppia di coordinate) e un riferimento al profilo dell'utente. Si risolvano i punti seguenti.

(a) Definire una classe Java per la rappresentazione per il grafo in questione, anche per consentire una maggiore efficienza nelle operazioni di cui ai punti successivi. Motivare le scelte rispetto ad altre possibilità.

(b) Definire un metodo Java che, dato un nickname, determini il numero di utenti (non amici) ai quali potrebbe essere "presentato" (attraverso una catena di presentazioni). Determinare il costo computazionale dell'algoritmo.

(c) Supponendo che siano disponibili due metodi `boolean compatibili(String nick1, String nick2)` (stabilisce se due utenti sono compatibili in base alla caratteristiche dei loro rispettivi profili) e `double distanzaIstantanea(String nick1, String nick2)` (calcola la distanza in linea d'aria in km fra due utenti), definire un metodo Java che, dati un `nickname` e una distanza d, costruisca e restituisca una lista di utenti compatibili con `nickname`, a una distanza non superiore a d. Determinare il costo computazionale dell'algoritmo.