FI2 - Compito di esame del 06 maggio 2011 – a.a. 2010-11, *tempo a disposizione: 90 minuti*

Problema 1

Si considerino i seguenti metodi:

```
static long smartExp(int base, int n) {
    if (n == 0) return 1;
    if (n % 2 == 1) return base * smartExp(base, n / 2) * smartExp(base, n / 2);
    return smartExp(base, n / 2) * smartExp(base, n / 2);
}
static long fastPower(int base, int n) {
    long f = 1;
    long b = base;
    while (n > 0) {
        int lsb = 0x1 & n;
        n >>= 1;
        if (lsb != 0) f *= b;
        b *= b;
    }
    return f;
}
```

(a) Calcolare, motivandolo adeguatamente, il costo computazionale del metodo `fastPower` in funzione del valore di `n`.

(b) Calcolare, motivandolo adeguatamente, il costo computazionale del metodo `smartExp` in funzione del valore di `n`.

(c) Calcolare, motivandolo adeguatamente, il costo computazionale dei due metodi precedenti in funzione della dimensione dell'input.

Problema 2

Con riferimento agli alberi binari di ricerca (BST) contenente chiavi di tipo intero e valori di tipo `String` si richiede quanto segue:

(a) Definire il TDA albero binario di ricerca elencando i metodi principali che lo caratterizzano. Realizzare poi le classi `BST` e `BSTNode` che rappresentano rispettivamente un albero binario di ricerca e il suo generico nodo (solo variabili di istanza e firma dei metodi pubblici del TDA e costruttori).

(b) Realizzare il metodo di istanza della classe `BST` la cui firma è `List<Integer> chiaviOrdinate()` che restituisce una lista di tutte le chiavi presenti nell'albero, ordinate in ordine crescente. *Il metodo dovrà avere costo lineare in funzione del numero di nodi dell'albero.*

(c) Realizzare il metodo di istanza della classe `BST` la cui firma è `boolean verificaBilanciamento()` che restituisce true se l'albero `this` è bilanciato secondo il concetto di bilanciamento definito nell'AVL.

Problema 3

Si vuole rappresentare il grafo stradale della città di Roma. I nodi (che indicano gli incroci) sono descritti da un valore intero (tra 0 e N-1) mentre gli archi orientati (che rappresentano i tratti di strada che uniscono gli incroci) sono caratterizzati dalle seguenti informazioni: *lunghezza* (in metri), *tempo medio di percorrenza* (in minuti), flag *marciapiede* (che indica se il tratto di strada ha un marciapiede per pedoni). Si richiede quanto segue:

(a) Descrivere la classe `GrafoRoma` (e le eventuali classi relative ad archi e nodi) che rappresenta il grafo stradale di Roma, utilizzando la rappresentazione che si ritiene più idonea. Le classi dovranno contenere, oltre alle variabili di istanza, anche le firme di tutti i metodi pubblici che si ritiene fondamentali per risolvere i problemi successivi.

(b) Realizzare un metodo della classe `GrafoRoma` la cui firma è `boolean percorsoPedonale(int source,int dest)` che restituisce `true` se esiste almeno un percorso pedonale che, a partire dal nodo `source` permette di raggiungere `dest`. Un percorso pedonale è definito come un tragitto che attraversa solo archi con marciapiede. Descrivere poi il costo computazionale dell'algoritmo, motivandolo opportunamente.

(c) Realizzare un metodo della classe `GrafoRoma` la cui firma è `int tempoMinimo(int source, int destination)` che restituisce il tempo minimo necessario a raggiungere il nodo `dest` a partire dal `source`. Restituire `-1` se il percorso non esiste. Descrivere poi il costo computazionale dell'algoritmo, motivandolo opportunamente.

matricola	*Nominativo*