FI2 - Compito di esame del 04 marzo 2011 – a.a. 2010-11, *tempo a disposizione: 90 minuti*

Problema 1

Si considerino i seguenti metodi:

```
private static int conta(int[][] mat) {
    int count = 0;
    for (int i = 0; i < mat.length; i++)
        count += mat[i].length;
    return count;
}
public static boolean mistero2(int[][] mat, int value) {
    return mistero2Ric(mat, value, 0);
}
public static boolean mistero2Ric(int[][] mat, int value, int counter) {
    if (counter >= conta(mat)) return false;
    int row = counter / mat[0].length;
    int col = counter % mat[0].length;
    return mat[row][col] == value || mistero2Ric(mat, value, counter + 1);
}
```

Assumendo che l'array di array `mat` descriva una matrice rettangolare di m righe e n colonne, si richiede quanto segue.

(a) Determinare, giustificandolo opportunamente, il costo computazionale del metodo `mistero2(int[][] mat, int value)` in funzione di m ed n.

(b) Esprimere il costo computazionale di cui al punto (a) in funzione della dimensione dell'input.

(c) Modificare l'algoritmo in modo da migliorarne il costo computazionale.

Problema 2

Con riferimento al tipo astratto Mappa si richiede di

(a) Definire il tipo astratto Mappa realizzando una possibile interfaccia Java `Map<K,V>` che lo descrive.

Si intende realizzare la classe Java `TabellaHashString` che implementa l' interfaccia `Map<String,String>` e che rappresenta una mappa con chiavi e valori di tipo `String` implementata tramite *tabella hash*. Dovranno essere impiegati: un *codice hash polinomiale*, una funzione di compressione basata sul *metodo di divisione* e gestione delle collisioni basata su *Linear Probing*.
Si richiede di implementare i seguenti metodi:

(b) `int hash(K key)` che implementa la funzione hash descritta.

(c) il metodo `put` che inserisce una nuova coppia chiave-valore nella mappa. In caso di fallimento lanciare opportuna eccezione.

(d) il metodo `remove` che rimuove la entry avente chiave specificata.

Problema 3

Con riferimento ai grafi orientati con pesi sugli archi si richiede quanto segue:

(a) Definire i concetti di cammino, lunghezza di un cammino e cammino minimo e spiegare la differenza tra lunghezza di un percorso e numero di archi di un percorso.

(b) Supponendo di voler rappresentare un grafo i cui nodi hanno etichette intere (tra 0 e $N-1$) e pesi sugli archi di tipo `double`, definire una possibile rappresentazione del grafo basata su matrice di adiacenza realizzando una classe Java `AdjGraph` contenente tutte le variabili di istanza e le firme di tutti i metodi pubblici ritenuti fondamentali per risolvere il problema successivo.

(c) Realizzare un metodo `List<Integer> searchPath(int src, int dest, double maxWeight)` (classe `AdjGraph`) che restituisce la lista dei nodi del percorso tra `source` e `dest` avente il minimo numero di archi, ciascuno dei quali con peso non superiore a `maxWeight`. Descrivere poi il costo computazionale dell'algoritmo, motivandolo opportunamente.

nominativo e matricola	*esame cui si partecipa*