Corso di Laboratorio di Programmazione I

A.A. 2013/2014

Progetto

Regole per lo svolgimento:

- Il progetto deve essere svolto singolarmente o in gruppi di al massimo 2 persone.
- Il progetto deve essere consegnato **almeno 5 giorni prima** della prova scritta della parte di Programmazione relativa all'appello in cui si desidera registrare l'esame.
- NON è permesso utilizzare nessuna libreria del linguaggio Java.
- Il progetto deve contenere i metodi richiesti rispettandone esattamente il nome, il tipo e l'ordine dei parametri formali, ed il tipo di ritorno.
- Si è naturalmente liberi di sviluppare metodi aggiuntivi di supporto ai metodi richiesti, laddove lo si ritenga utile.
- La consegna (dei 2 file Insiemi.java e Multiinsiemi.java) va effettuata all'indirizzo **luca.moscardelli@gmail.com**, mettendo come oggetto **[jmm] consegna progetto laboratorio 2014**, e nel testo dell'email il/i nome/i dello/degli studente/i che consegna/no

Testo del progetto:

Sviluppare in **Java** (non in Java--) una libreria per la gestione di insiemi e di multiinsiemi di numeri interi.

Prima parte:

Gli insiemi devono essere gestiti in Java come array di int, contenente tutti gli interi presenti nell'insieme.

La libreria deve essere composta dai seguenti metodi, tutti inclusi nella stessa classe **Insiemi** salvata nel file **Insiemi.java**:

- `public static int[] rimuoviDuplicati (int[] a)`
  prende come parametro un array di interi e restituisce un nuovo array di interi in cui sono presenti tutti gli elementi di a senza duplicati.
- `public static int cardinalita (int[] a)`
  prende come parametro un array di interi e restituisce la cardinalità dell'insieme da esso rappresentato.
- `public static boolean appartiene (int[] a, int n)`
  prende come parametro un array di interi ed un intero n, e restituisce ***true*** se n appartiene all'insieme rappresentato da a, ***false*** altrimenti.
- `public static int[] creaSingleton (int n)`
  prende come parametro un intero n, e restituisce un array che rappresenta un insieme formato dal solo elemento n.

- `public static int[] unione (int[] a, int[] b)`
  prende come parametri due array di interi, e restituisce un nuovo array di interi corrispondente all'insieme ottenuto facendo l'unione degli insiemi rappresentati da a e b. [si sfrutti il metodo `rimuoviDuplicati`]
- `public static int[] intersezione (int[] a, int[] b)`
  prende come parametri due array di interi, e restituisce un nuovo array di interi corrispondente all'insieme ottenuto facendo l'intersezione degli insiemi rappresentati da a e b.
- `public static int[] differenza (int[] a, int[] b)`
  prende come parametri due array di interi, e restituisce un nuovo array di interi corrispondente all'insieme ottenuto facendo la differenza insiemistica tra l'insieme rappresentato da a e quello rappresentato da b.
- `public static boolean sottoinsieme (int[] a, int[] b)`
  prende come parametri due array di interi, e restituisce ***true*** se l'insieme rappresentato da a è un sottoinsieme di quello rappresentato da b, ***false*** altrimenti.
- `public static boolean uguale (int[] a, int[] b)`
  prende come parametri due array di interi, e restituisce ***true*** se l'insieme rappresentato da a è lo stesso di quello rappresentato da b, ***false*** altrimenti.
  [si sfrutti il metodo `sottoinsieme`]

Seconda parte:

Un multi-insieme è una collezione finita di elementi non ordinati. Gli elementi in un multi-insieme possono essere ripetuti e si chiama **molteplicità** di un elemento x in un multi-insieme A il numero di volte in cui x compare in A.

I multiinisiemi devono essere gestiti come array di `Elem`, dove il tipo riferimento `Elem` è definito nel modo seguente:

```
class Elem{
   int valore;
   int molteplicita;
}
```

Si noti come, in tale array, non devono esistere due elementi aventi lo stesso campo valore.

La libreria deve essere composta dai seguenti metodi, tutti inclusi nella stessa classe **Multiinsiemi** salvata nel file **Multiinsiemi.java**:

- `public static Elem[] trasforma (int[] a)`
  prende come parametro un array di interi contenente una collezione di elementi (con possibili ripetizioni) restituisce un nuovo array di Elem in cui ad ogni intero è associata l'opportuna molteplicità. Ad esempio **a** potrebbe essere la collezione {{1, 6, 2, 3, 6, 6, 3, 4, 4}} contenente 1 e 2 con molteplicità 1, 3 e 4, con molteplicità 2 e 6, con molteplicità 3.
- `public static Elem[] normalizza (Elem[] a)`
  prende come parametro un array di Elem , e restituisce un nuovo array di Elem in cui non

sono presenti due elementi aventi lo stesso campo valore. Se nell'array **a** erano presenti due o più elementi con lo stesso campo valore, nel nuovo array deve essere presente un solo elemento con quel campo valore, la cui molteplicità è pari alla somma delle molteplicità degli elementi che in **a** avevano lo stesso campo valore.

- `public static int molteplicita (Elem[] a, int n)`
  prende come parametro un array di Elem ed un intero n, e restituisce la molteplicità di n nel multiinsieme rappresentato da a. Se n non appartiene a tale multiinsieme, la sua molteplicità è 0.
- `public static Elem[] creaSingleton (int n)`
  prende come parametro un intero n, e restituisce un array che rappresenta un multiinsieme formato dal solo elemento n (con molteplicità 1).
- `public static Elem[] unione (Elem[] a, Elem[] b)`
  prende come parametri due array di Elem, e restituisce un nuovo array di Elem corrispondente al multiinsieme ottenuto facendo l'unione dei multiinsiemi rappresentati da a e b. L'unione di due multi-insiemi si ottiene prendendo tutti gli elementi (comuni e non comuni) ai 2 multiinsiemi, con la **massima** molteplicità.
- `public static Elem[] intersezione (Elem[] a, Elem[] b)`
  prende come parametri due array di Elem, e restituisce un nuovo array di Elem corrispondente al multiinsieme ottenuto facendo l'intersezione dei multiinsiemi rappresentati da a e b. L'intersezione di due multiinsiemi si ottiene prendendo solo gli elementi comuni ai due multiinsiemi, con la **minima** molteplicità.
- `public static Elem[] somma (Elem[] a, Elem[] b)`
  prende come parametri due array di Elem, e restituisce un nuovo array di Elem corrispondente al multiinsieme ottenuto facendo la somma dei multiinsiemi rappresentati da a e b. La somma di due multi-insiemi si ottiene prendendo tutti gli elementi (comuni e non comuni) ai 2 multiinsiemi, e **sommando** le loro molteplicità.
  [si sfrutti il metodo `normalizza`]
- `public static Elem[] differenza (Elem[] a, Elem[] b)`
  prende come parametri due array di Elem, e restituisce un nuovo array di Elem corrispondente al multiinsieme ottenuto facendo la differenza dei multiinsiemi rappresentati da a e b. La differenza di due multi-insiemi si ottiene prendendo tutti gli elementi del primo multiinsieme e sottraendo dalla loro molteplicità la molteplicità del corrispondente elemento (con lo stesso campo valore, se è presente) del secondo multiinsieme. Se tale differenza è minore o uguale a 0, l'elemento viene rimosso dal multiinsieme risultato.
- `public static boolean sottoinsieme (Elem[] a, Elem[] b)`
  prende come parametri due array di interi, e restituisce ***true*** se il multiinsieme rappresentato da a è un sottoinsieme di quello rappresentato da b (ovvero ogni elemento presente in a è presente anche in b con una molteplicità almeno uguale a quella che l'elemento ha in a), ***false*** altrimenti.
- `public static boolean uguale (Elem[] a, Elem[] b)`
  prende come parametri due array di interi, e restituisce ***true*** se il multiinsieme rappresentato da a è lo stesso di quello rappresentato da b, ***false*** altrimenti.
  [si sfrutti il metodo `sottoinsieme`]