Corso di Algoritmi 2 - Prova scritta: esercizio di programmazione

Una società di atletica ha bisogno di un software per organizzare la partecipazione dei propri atleti alle gare. Ogni gara è costituita da prove di discipline diverse, che si effettuano una dopo l'altra, denotate con indentificativi numerici in ordine crescente 0,1, 2,..., n-1 (la prova i è immediatamente seguita dalla prova i+1). Per ogni atleta esiste un indice di rendimento per ogni disciplina (piú alto è l'indice di rendimento in una certa disciplina piú è probabile che l'atleta vinca una prova in quella disciplina). *Il rendimento totale previsto per un atleta ad una gara è la somma degli indici di rendimento delle prove di discipline diverse alle quali l'atleta partecipa.* **L'atleta non può mai partecipare a due prove consecutive** (si deve riposare). Si vuole un software che permetta alla società, per una data gara, di scegliere tra un elenco di atleti quello che ha rendimento totale previsto massimo tra tutti (per la versione dimostrativa del software che dovete implementare voi, la scelta è limitata a due atleti).

Scrivete una classe Java **Atletica** con il costruttore:

public Atletica (int *numeroDiscipline***)**

dove *numeroDiscipline* è il numero di discipline per cui ci saranno prove nella gara.

e implementate il metodo:

public int scelta(int[] *rendAtleta1***, int[]** *rendAtleta2***)**

dove *rendAtleta1* e *rendAtleta2* sono due array che indicano, rispettivamente per l'atleta 1 e per l'atleta 2, per ciascuna disciplina l'indice di rendimento; il metodo restituisce **1** se alla società conviene che partecipi l'atleta **1**; **2** se alla società conviene che partecipi l'atleta **2**; **0** se è indifferente; se per uno dei due atleti la lunghezza dell'array *rendAtleta* non corrisponde a *numeroDiscipline* (è maggiore o minore), solleva una java.lang.IllegalArgumentException. Le prove delle diverse discipline sono in ordine temporale 0,1,..., *numeroDiscipline* -1.

Esempi:
se *numeroDiscipline* = 5, allora **scelta**(*[8,4,2,6,3], [3,10,7,7,4])* = **2***;*
se *numeroDiscipline* = 5, allora **scelta**(*[8,4,2,6,3], [3,10,7,4,4])* = **0***;*
se *numeroDiscipline* = 5, allora **scelta**(*[8,4,2,6,3,1], [3,10,7,7,4])* solleva una **IllegalArgumentException**.

TEST: Lavorate implementando anche una classe test JUnit con almeno due test **significativi**, va bene anche se sono semplici.

Struttura del progetto Java e consegna: La classe richiesta e la classe test devono essere contenute in un **package** il cui nome è **il vostro nomecognome** (senza punti né spazi, tutto minuscolo). Consegnate solo il package. Se non rispettate le istruzioni di consegna, questo influirà sulla valutazione.

Suggerimenti e commenti generali per tutti i progetti:
0) Dovete risolvere il problema rigorosamente applicando uno degli algoritmi visti a lezione.
1) Potete consultare la documentazione Java: http://master.edu-al.unipmn.it/docs/TIJ/TIJ3.htm per il linguaggio e http://master.edu-al.unipmn.it/docs/java2se/docs-1.8/api/index.html per le API.
2) Se serve, potete usare la libreria graphLib.jar (che sarà comunque sempre fornita). Alcune note:
- è allegato un file con le istruzioni per visualizzare i commenti Javadoc per i metodi;
- la sintassi per la descrizione di un grafo nella libreria graphLib è del tipo: "3; 0 1 4; 1 2 5" dove 3 è il numero dei nodi, (0,1) e (1,2) sono archi di peso 4 e 5 rispettivamente;
- nella versione della libreria che avete a disposizione, il metodo per ottenere gli archi incidenti o uscenti da un nodo è `getOutEdges(int nodo)`;
- osservate che il metodo **union** delle implementazioni di UnionFind che trovate nella libreria vuole come argomenti *dei rappresentanti* di insiemi, non degli elementi qualunque.

3) Potete aggiungere tutti i metodi private che volete.
4) **NON selezionate la cartella consegna come vostro workspace**; inserite nella cartella consegna solo il vostro package per la consegna o per consegne parziali (che vi proporrò durante il compito).
5) Verranno valutatati: l'algoritmo utilizzato, la correttezza e l'aderenza alle specifiche (comprese la gestione degli input particolari e l'inizializzazione, nonché le modalità di consegna), la qualità dell'implementazione e la presenza di una classe test con test significativi.