# Laboratorio di Algoritmi e Strutture Dati
# 2016/17
# Mini Progetto, Codice: MP4

Docente: Luca Tesei

**Scadenza**: 3 Marzo 2017 ore 23.59

## 1 Implementazione di una classe per grafi orientati

Si scriva il codice di una classe `GraphMatrixDirected<V, E>` che implementi l'interface `Graph<V, E>`, fornita come allegato, definita per rappresentare generici grafi i cui nodi sono etichettati con oggetti (non `null` e tutti distinti) della classe `V` e i cui archi sono etichettati con oggetti (possibilmente `null` e ripetuti) della classe `E`. La classe deve permettere di rappresentare grafi *orientati* (o diretti) e deve usare come rappresentazione una *matrice di adiacenza*. All'interno della classe si potranno utilizzare come variabili istanza sia degli array veri e propri sia delle liste, ad esempio `ArrayList<ArrayList<E>>`. Si noti che le etichette degli archi (oggetti della classe `E`) possono essere `null` anche in presenza dell'arco, pertanto non è possibile sfruttare il fatto che l'etichetta sia `null` per indicare che un certo arco non è presente; è necessario cioè usare un ulteriore bit per la presenza o assenza dell'arco (si creino opportune classi interne per istanziare la classe `E` e/o per mettere come tipo degli array).

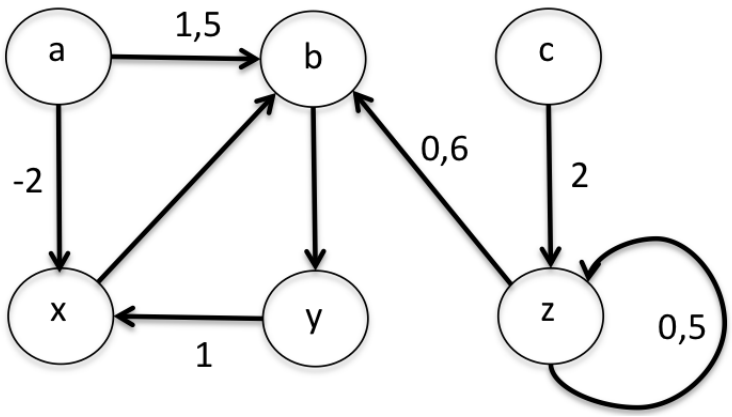
## 2 Implementazione della visita in profondità

Si scriva una classe `GenericGraphDFS<V,E>` che implementi la visita in profondità di un generico grafo espresso tramite una classe che implementi l'interfaccia `Graph<V, E>`. Oltre alla colorazione dei nodi (disponibile già usando i metodi dell'interfaccia) i nodi devono contenere, alla fine della visita, due numeri interi; il primo indica il tempo (discreto) a cui il nodo è stato scoperto e il secondo il tempo (discreto) a cui il nodo è stato completamente esplorato in profondità. Il tempo viene incrementato in maniera discreta ogni volta che si scopre un nodo e ogni volta che un nodo è stato completamente esplorato in profondità. Si impongano dei vincoli opportuni alla classe `V` in `GenericGraphDFS<V,E>` in modo che sia obbligata a fornire metodi per gestire questi due numeri in ogni

nodo (ad esempio si definisca una interfaccia che permette di manipolare questi numeri).

## 3 Test

Si scriva una classe di test creando il seguente grafo:




attraverso la classe `GraphMatrixDirected<V, E>`. Gli archi che non hanno etichetta si assumono ad etichetta `null` e le etichette dei nodi sono semplici stringhe. Si chiami poi l'algoritmo di visita DFS generico implementato (si usino classi opportunamente definite per istanziare `V` ed `E`). Una volta eseguita la visita si faccia stampare il grafo (con i tempi di scoperta e visita completa in profondità di ogni nodo e con i valori di ogni arco) sullo standard output attraverso una opportuna ridefinizione del metodo `toString()` della classe `GraphMatrixDirected<V, E>`.

Le classi dovranno essere completamente autodocumentate tramite commenti interpretabili dall'utility `javadoc` e con commenti privati. Codice non adeguatamente commentato sarà valutato negativamente.

## Modalità di Consegna

I file `.java` per le classi, senza indicazione di package (cioè appartenenti al package di default) e senza il codice già fornito per il framework devono essere caricati entro la data di scadenza in una cartella Google Drive dal nome

`ASDL1617MP4-CognomeStudente-NomeStudente`

che deve essere condivisa, in sola lettura, tramite l'account

`nome-studente.cognome-studente@studenti.unicam.it`

con gli account:

- `luca.tesei@unicam.it` (docente) e

- `matteo.micheletti@unicam.it` (tutor).

Per la scadenza, farà fede la data dei file su Google Drive.

## Allegati

- `Graph.zip` - Contiene il codice commentato dell'interfaccia `Graph<V, E>` e della classe `Edge<V, E>`.