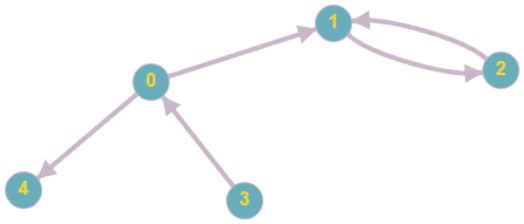
Grafi

Esercizio 0:

Scrivere un programma che utilizzi il metodo setEdge per creare un oggetto della classe Graphm che rappresenti il seguente grafo:



Esercizio 1:

Scrivere una funzione membro reverse() che trasformi un grafo (descritto tramite matrice di adiacenza) nel suo grafo opposto. L'opposto di un grafo è il grafo con il verso di tutti gli archi invertito. Verificare il funzionamento della funzione su un grafo letto da file oppure costruito tramite chiamate alla funzione setEdge.

Esercizio 2:

Letti in input due numeri interi n e k (0 < k < n), scrivere una funzione membro della classe Graphl (grafo con rappresentazione mediante lista di successori) per costruire un grafo orientato G = (V,E) in modo casuale tale che V = {0,1,2, ...,n-1} e ogni vertice abbia al massimo k spigoli uscenti.

Esercizio 3:

Leggere in input un grafo orientato G=(V,E) e rappresentarlo mediante lista di successori. Letto in input un vertice v (0 <= v <= n) eliminare tutti gli spigoli entranti in v.

Esercizio 4:

Scrivere un programma per verificare se, dato un grafo connesso, e date in ingresso le etichette di due nodi a e b, dal primo nodo è possibile arrivare al secondo attraverso un cammino di archi orientati. Procedimento: eseguire un attraversamento DFS partendo dal nodo a. Se durante l'attraversamento si incontra il vertice b allora il percorso esiste.

Esercizio 5:
Scrivere una funzione membro reverse() che trasformi un grafo (descritto tramite lista di successori) nel suo grafo opposto. L'opposto di un grafo è il grafo con il verso di tutti gli archi invertito.

Esercizio 6:
Scrivere una funzione membro complement() che trasformi un grafo (descritto tramite lista di successori) nel suo grafo complementare. Il grafo complementare è il grafo ottenuto eliminando tutti gli archi presenti nel grafo originale ed inserendo tutti gli archi non presenti nel grafo originale.