Informatica Generale
Homework 3: Grafi

docente: Ivano Salvo
Sapienza Università di Roma

pubblicazione: 26.V.2016 - consegna 13.VI.2016

Una relazione binaria R su un insieme V finito, di fatto, rappresenta un grafo diretto. Se la relazione R è simmetrica, può rappresentare un grafo non diretto. Anche in questo homework, tutti gli esercizi leggeranno in input un numero intero n e poi $n \times n$ interi (che in realtà saranno solo 0 e 1) che rappresentano la *matrice di adiacenza* di un grafo diretto.

Esercizio 1: (Labirinto) Scrivere un programma C che letta una matrice come sopra descritto, (numerando i nodi da 0 a $n-1$) scriva 1 se esiste un cammino che parte dal nodo 0 e arriva al nodo $n-1$. Il programma deve scrivere 0 altrimenti.

Assumete $n > 1$ (ci sono quindi almeno 2 nodi).

[Suggerimento: fare una qualsiasi visita a partire dal nodo 0. Se nella visita incontrate il nodo $n-1$ allora potete stampare 1. Se la visita finisce senza incontrare $n-1$ stampate 0.

Se temete di dover gestire personalmente una pila o coda, potete fare una visita in profondità ricorsiva con l'avvertenza di marcare i nodi già visitati per evitare di ciclare su nodi già visitati. Potete, allo scopo, usare un vettore che passate come parametro e che marca i nodi già visitati.

Chi avesse fatto l'esercizio 3 dell'Homework 2, può trovare il modo di reciclare quella soluzione ☺]

Esercizio 2: Scrivere un programma C che letta una matrice come sopra descritto, stampi 1 se il grafo rappresentato dalla matrice contiene almeno un ciclo, e 0 altrimenti.

[Suggerimento: Come nell'esercizio precedente, potete fare una visita in profondità partendo da un nodo. Non appena trovate un nodo già visitato *nel cammino in esame* potete concludere che c'è un ciclo e produrre 1 in output. Marcate anche in qualche modo *tutti* i nodi visitati.

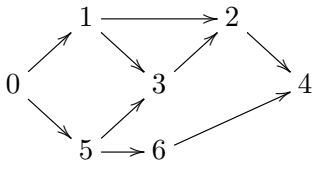

Figura 1: Esercizio 3, Esempio 1

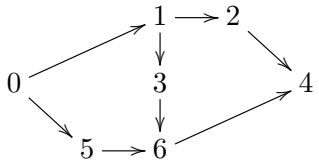

Figura 2: Esercizio 3, Esempio 2

Quando avete finito, dovete ripartire da un nodo mai visitato (se esiste) e ricominciare. Se avete esaminato tutti i nodi e tutti i cammini senza mai trovare un ciclo, potete finalmente scrivere 0.

Come nell'esercizio precedente, si può trovare il modo di trarre vantaggio dall'esercizio 3 dell'Homework 2.]

Esercizio 3: (Topological Sort – Facoltativo) In un grafo diretto acicliclo, i nodi possono essere ordinati secondo il seguente ordine parziale: $u \preceq v$ se esiste un cammino da u a v. I minimali di questo ordine sono i nodi senza archi entranti. I massimali sono i nodi senza archi uscenti.

Diciamo che una sequenza $a_1, a_2, \ldots, a_n$ di nodi rispetta l'ordine topologico se $i \leq j$ implica $a_j \not\preceq a_i$ nell'ordine topologico. Non si tratta di un ordine totale e infatti ci sono molte sequenze che rispettano l'ordine topologico. Prendiamo ad esempio il grafo in Fig. 1. Sia la sequenza 0 5 1 3 6 2 4 che la sequenza 0 1 5 3 2 6 4 rispettano $\preceq$.

Il vostro programma letto un grafo rappresentato con la sua matrice di adiacenza come sopra descritto, dovrà stampare la sequenza che rispetta l'ordine topologico minima rispetto all'ordine lessicografico delle sequenze (un singolo nodo è minore di un altro se ha indice minore). Assumete il grafo aciclico.

Nell'esempio di Fig. 1, la sequenza minima è proprio 0 1 5 3 2 6 4. Nell'esempio di Fig. 2, la sequenza minima è 0 1 2 3 5 6 4.

[Suggerimento: potete procedere come segue. Ad ogni iterazione prendete il nodo senza archi entranti di indice minimo. A questo punto "cancellate" tale nodo, rimuovendo tutti i suoi archi uscenti e riapplicate il ragionamento finchè non avete scritto tutti i nodi.]