

# Fondamenti di Informatica I (12 cfu) - A.A. 2013-2014

Corsi di Laurea in Ingegneria Gestionale

Sapienza Università di Roma

## Prova al calcolatore

Esercitazione 30 Maggio 2014 - Durata 1h 30'

### Esercizio 1

Completare la funzione Python `conn2(g, u, v)` (contenuta nel file `Esercit12Prog1.py`) che preso in ingresso un dizionario rappresentante la topologia di un grafo e due nodi  $u$  e  $v$ , restituisca `True` se  $v$  può essere raggiunto a partire da  $u$  attraversando *esattamente* due archi, `False` altrimenti. Si assuma che le etichette dei nodi siano stringhe. Ad esempio, nel grafo in figura, è possibile raggiungere il

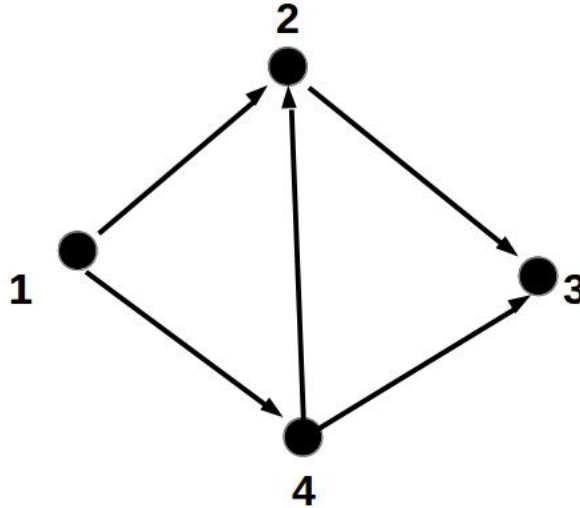


Figure 1: Esempio di grafo diretto

nodo 2 dal nodo 1 in esattamente 2 passi, mentre ciò non è possibile a partire dal nodo 4.

**Suggerimento:** si costruisca la matrice di adiacenza del grafo e poi si considerino le proprietà della matrice  $A$  tale che  $A_{ij} = 1$  se  $j$  è raggiungibile a partire da  $i$  in *esattamente* due passi  $A_{ij} = 0$  altrimenti. Lo studente potrebbe trovare utile la funzione `prodotto(A, B)` (già implementata nel file `Esercit12Prog1.py`) che, date due matrici consistenti  $A$  e  $B$  rappresentate come liste di liste, ne calcola il prodotto.

Scrivere la soluzione in modo da poter eseguire il programma di prova `ProvaEx1.py`, senza doverlo modificare.

### Esercizio 2

Completare la funzione Python `indiceInverso(fin)` (contenuta nel file `Esercit12Prog2.py`) che, preso in ingresso il nome di un file testo contenuto nella directory locale, restituisca un dizionario

avente come chiavi i termini che compaiono nel file testo e per valori le liste dei numeri delle righe in cui ciascun termine appare *almeno una volta*. Si supponga di assegnare alla prima riga del file il numero di ordine 1. Ad esempio, se il testo contenuto nel file fosse:

a rose is a rose
is a
rose is

Allora il dizionario (a parte l'ordine in cui appaiono le chiavi) sarebbe questo:

`{ 'a': [1, 2], 'rose': [1, 3], 'is': [1, 2, 3] }`

Scrivere la soluzione in modo da poter eseguire il programma di prova `ProvaEx2.py`, senza doverlo modificare.

### Esercizio 3

Nel problema della *bisaccia*, abbiamo un insieme  $A = \{O_1, \dots, O_n\}$  di oggetti. Il generico oggetto  $O_i$  ha valore  $v_i$  e peso  $p_i$ . Abbiamo a disposizione una bisaccia che può sopportare un peso complessivo  $P$ . L'obiettivo è quello di riempire la bisaccia con il sottoinsieme di oggetti avente massimo tra quelli il cui peso totale non eccede  $P$ .

Per quanto strano possa sembrare, non sono noti algoritmi che trovino una soluzione ottima per questo problema in tempo polinomiale in tutti i casi. Una semplice euristica, che consente di trovare soluzioni “quasi” ottime (e effettivamente ottime in molti casi di interesse pratico) è la seguente:

1. Ordina gli oggetti secondo valori *decrescenti* dei rapporti  $v_i/p_i$  (dunque, dal più grande al più piccolo).
2. Inserisci gli oggetti nella bisaccia in quest'ordine. Se il prossimo oggetto non può essere inserito perché ciò causerebbe il superamento del peso massimo consentito, passa all'elemento successivo.
3. Termina quando raggiungi il termine della lista.

Ad esempio, se  $P = 10$  e gli oggetti fossero  $O_1 = (3, 5)$ ,  $O_2 = (4, 8)$ ,  $O_3 = (2, 1)$ ,  $O_4 = (1, 4)$ . In questo caso, gli oggetti vengono esaminati nell'ordine  $\{O_3, O_1, O_2, O_4\}$  e gli oggetti  $\{O_3, O_1, O_4\}$  vengono inseriti nella bisaccia (in quest'ordine). Si noti che in questo caso la soluzione calcolata ha valore ottimo.

Supponiamo di rappresentare una bisaccia con un dizionario in questo modo: i) le chiavi sono etichette (stringhe) che identificano gli oggetti della collezione; ii) il valore associato all'etichetta  $e$  (che supponiamo corrispondente all'oggetto  $O_i$ ) è una lista di due elementi, il primo dei quali è  $v_i$ , il secondo  $p_i$ .

Completare la funzione Python `bisaccia(l, P)` (contenuta nel file `Esercit12Prog2.py`) che, presi in ingresso un dizionario rappresentante la collezione di oggetti e il valore  $P$  del peso massimo consentito, restituisca una lista delle chiavi degli oggetti inseriti nella bisaccia.

Scrivere la soluzione in modo da poter eseguire il programma di prova `ProvaEx3.py`, senza doverlo modificare.