

Fondamenti di Informatica I (12 cfu) - A.A. 2014-2015

Corsi di Laurea in Ingegneria Gestionale

Sapienza Università di Roma

Prova al calcolatore

Esercitazione 22 Maggio 2015 - Durata 1h 45'

Create una cartella `Esercitazione10` sul Desktop e copiate lì tutti i file. Per ogni esercizio avete un file dal nome `Esercit10ProgX.py` (dove **X** è il numero dell'esercizio) con lo schema della soluzione, un file `ExX.pyc` ed uno `ProvaExX.py`. Per l'esercizio **X**, completate la funzione contenuta nel file `Esercit10ProgX.py`. Per testare un esercizio DOVETE usare i programmi `ProvaExX.py`. I file `ExX.pyc` non sono leggibili,

Esercizio 1

Completare la funzione Python `admat(g, v)` (contenuta nel file `Esercit10Prog1.py`) che preso in ingresso un dizionario **g** rappresentante la topologia di un grafo e un secondo dizionario **v** che rappresenta il vettore dei nodi, restituisca la matrice di adiacenza del grafo, rappresentata come lista di liste. In particolare, la generica chiave di **v** è l'etichetta di un nodo, mentre il valore è un intero che identifica la riga (e anche la colonna) della matrice di adiacenza associata al grafo. Ad esempio, il valore di **g** per il grafo in Figura 1 è (a parte l'ordine in cui compaiono le chiavi): `{'1': ['2', '4'], '2': ['3'], '4': ['2', '3']}`. Qualora il valore di **v** fosse (di nuovo a parte l'ordine delle chiavi) `{'1': 2, '2': 0, '3': 1, '4': 3}`, allora la funzione dovrebbe restituire la matrice `[0, 1, 0, 0], [0, 0, 0, 0], [1, 0, 0, 1], [1, 1, 0, 0]`.

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

Esercizio 2

Completare la funzione Python `conn2(g, u, v)` (contenuta nel file `Esercit10Prog2.py`) che presa in ingresso la matrice di adiacenza 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 suppona che **u** e **v** siano interi e identifichino le corrispondenti righe (colonne) nella matrice di adiacenza. Ad esempio, nel grafo in figura, è possibile raggiungere il nodo 2 dal nodo 1 in esattamente 2 passi, mentre ciò non è possibile a partire dal nodo 4.

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

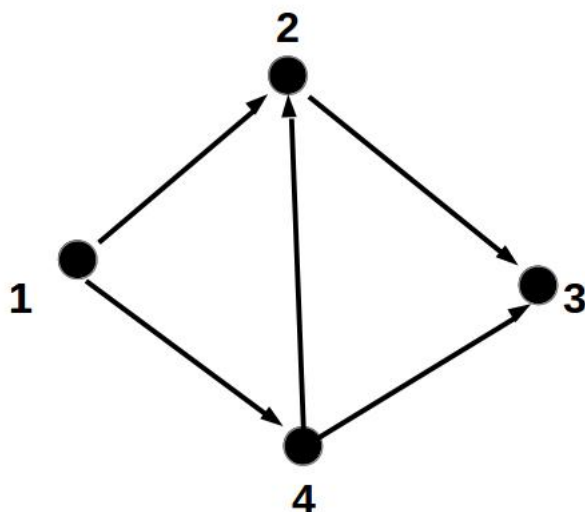


Figure 1: Esempio di grafo diretto

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 *decreasing* 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 `Esercit10Prog2.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.

Esercizio 4

Scrivere una funzione Python `somma_righe(matrice, estremi)` (contenuta nel file `Esercit10Prog4.py`) che prese in ingresso una matrice $n \times m$ di interi `matrice` e una matrice $n \times 2$ di interi `estremi`, restituisca una lista di interi in cui l'elemento i -esimo contiene la somma degli elementi della riga i -esima di `matrice` compresi fra gli elementi contenuti rispettivamente nella posizione 0 e 1 della riga i -esima della matrice `estremi`. Si considerino a titolo di esempio le matrici che seguono:

$$m = \begin{pmatrix} 1 & 2 & 0 & 3 \\ 1 & -1 & 2 & 1 \\ 0 & 2 & 4 & 0 \end{pmatrix} e = \begin{pmatrix} 0 & 3 \\ 1 & 2 \\ 1 & 1 \end{pmatrix} \text{ somma_righe} = \begin{pmatrix} 6 \\ 1 \\ 2 \end{pmatrix}$$

Nell'esempio riportato sopra, il valore 6 è ottenuto sommando gli elementi della prima riga di m a partire dall'indice di colonna 0 fino all'indice di colonna 3. Analogamente il valore 1 (seconda riga) è ottenuto sommando gli elementi di m a partire dall'indice di colonna 1 fino all'indice di colonna 2 incluso. Infine il valore 2 (terza riga) è ottenuto sommando gli elementi di m a partire dall'indice 1 fino all'indice 1. Scrivere la soluzione in modo da poter eseguire il programma di prova `ProvaEx4.py`, senza doverlo modificare.

Esercizio 5

Scrivere una funzione Python RICORSIVA `stringaPariDispari(s)` (contenuta nel file `Esercit10Prog5.py`) che una stringa, restituisca una nuova stringa che contenga prima le lettere di posto pari e poi quelle di posto dispari.

Ad esempio, sia s uguale a `tormenta`, la funzione deve restituire la stringa `tretomna`.