

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

Corsi di Laurea in Ingegneria Gestionale

Sapienza Università di Roma

Prova al calcolatore Esercitazione 15 Maggio - Durata 1h 45'

Create una cartella **Esercitazione9** sul Desktop e copiate lì tutti i file. Per ogni esercizio avete un file dal nome **Esercit9ProgX.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 **Esercit9ProgX.py**. Per testare un esercizio DOVETE usare i programmi **ProvaExX.py**. I file **ExX.pyc** non sono leggibili, ma contengono la nostra soluzione al problema e sono usati da **ProvaExX.py** per controllare se la vostra soluzione sia corretta.

Esercizio 1

Scrivere una funzione Python `nvicini((g, u, v))` (contenuta nel file **Esercit9Prog1.py**) che presi in ingresso un grafo **g** rappresentato come un dizionario e due vertici **u** e **v** (si assume che le etichette dei vertici siano stringhe), restituisca il numero di vicini in comune tra **u** e **v**. Ad esempio, se il grafo, rappresentato come un dizionario, è `{'1': ['3', '6'], '3': ['5', '7'], '2': ['1', '4', '5'], '5': ['4', '7'], '4': ['6'], '7': ['4'], '6': ['1']}`, **u** = `'2'` e **v** = `'6'`, la funzione deve restituire 1 (i due nodi hanno il solo nodo `'1'` in comune).

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

Esercizio 2

Scrivere una funzione Python `inavgdeg(g)` (contenuta nel file **Esercit9Prog2.py**) che presi in ingresso un grafo diretto **g** rappresentato come un dizionario, restituisca il grado medio *entrante*. Ad esempio, se il grafo, rappresentato come un dizionario, è `{'1': ['3', '2'], '2': ['1', '3']}`, il grado medio entrante è $4/3 = 1.3333...$

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

Esercizio 3

Scrivere una funzione Python `conn(g, v, w)` (contenuta nel file **Esercit9Prog3.py**) che presi in ingresso un grafo *diretto* **g** rappresentato come un dizionario e due vertici **v** e **w** (si assume che le etichette dei vertici siano stringhe), restituisca **True** se **w** è raggiungibile a partire da **v** seguendo un cammino diretto (ossia se **v** e **w** appartengono alla stessa componente connessa) e restituisca **False** altrimenti. Ad esempio, se il grafo, rappresentato come un dizionario, è `{'1': ['2', '3'], '3':`

`['2', '4'], '4': ['2']}]`, allora la funzione deve restituire `True` quando `v = '1'` e `w = '4'` e `False` quando `v = '4'` e `w = '1'`.

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

Esercizio 4

Si consideri un file di testo contenente il dettaglio delle vendite azionarie di una società finanziaria: per ogni riga del file ci sarà il nome del titolo azionario venduto e il numero delle azioni vendute separate dal carattere `;`. Scrivere un metodo Python `carica(file)` che dato il nome di un file di testo restituisca un dizionario contenente le coppie titolo / quantità caricate dal file. *Si consideri che lo stesso titolo può apparire più volte nel file: in quel caso i valori delle vendite vanno sommati.* Ad esempio, se il file è il seguente:

```
Aeffe;120
A.S. Roma;90
A2a;20
Acea;269
Acotel Group;158
Acque Potabili;173
Acsm-Agam;264
Acea;234
Aedes;82
Aeffe;12
Aegon;51
Aeffe;127
```

Allora la società ha venduto 259 titoli della società “Aeffe”, 90 di “A.S. Roma” 90 ecc.

Scrivere la soluzione in modo da poter eseguire il programma di prova `ProvaEx4.py`, senza doverlo modificare.