

Fondamenti di Informatica per Ing. Gestionale

Esercitazione del 31 Maggio 2018

Note

La mancata osservanza delle seguenti regole può comportare la perdita di informazioni necessarie alla valutazione dell'esame.

Cartella di esame e registrazione dei dati dello studente

La cartella Esame contenente il compito da svolgere si trova sul Desktop. Entrare nella cartella e, prima di iniziare il compito, eseguire il programma `registrazione.pyc`. Inserire i dati personali fornendo (separatamente) *Numero di Matricola*, *Cognome* e *Nome*. Il programma genera un file `studente.txt` che non deve essere modificato manualmente. Verificare che i dati nel file `studente.txt` siano corretti; in caso di errore potete rieseguire il programma `registrazione.pyc`.

Svolgimento degli esercizi - Leggere attentamente

Leggere attentamente il testo e risolvere gli esercizi proposti.

Per ogni esercizio avete una cartella `EsercN` che contiene un file dal nome `B_ExN.py` (dove N è il numero dell'esercizio) con lo scheletro della soluzione. Il file `B_S_ExN.py` contiene la soluzione dell'esercizio proposta dal docente. *E' nel vostro diretto interesse non aprire quest'ultimo file prima di aver dedicato tutto il tempo disponibile a risolvere autonomamente l'esercizio.* Non create nuovi file.

Per verificare la correttezza di un esercizio **DOVETE** usare il programma `TestEx.pyc`, che proverà la vostra soluzione su un certo numero di casi di test, mostrandovi il confronto tra i risultati ottenuti con la vostra soluzione e usando la soluzione proposta dal docente. Ciò ha lo scopo di facilitarvi nella ricerca e correzione di eventuale errori

1. **B_Ex1(s) (8 punti)** Completare la funzione Python `B_Ex1(s)` che prende in ingresso una stringa s. La funzione deve restituire una stringa ottenuta da s nel seguente modo. La stringa da restituire è la concatenazione di len(s) stringhe, la i-esima delle quali è ottenuta ripetendo l'i-esimo carattere di s (i+1) volte (quindi 1 volta il carattere in posizione 0, due volte il carattere in posizione 1 e così via). Qualora s sia la stringa nulla la funzione deve restituire la stessa stringa.

Esempio: se la stringa fosse "amaca" la funzione dovrebbe restituire la stringa "ammaaaccccaaaaa".

2. **B_Ex2(m) (8 punti)** Completare la funzione Python `B_Ex2(m)` che riceve in ingresso una matrice m e restituisce una nuova matrice che rappresenta la matrice triangolare superiore estratta da m. Se la matrice m non è quadrata (o vuota), la funzione dovrà restituire la matrice vuota.

Esempio: se la matrice fosse

$$\begin{pmatrix} 2 & 2 \\ 3 & 3 \end{pmatrix}$$

la funzione dovrebbe restituire la matrice

$$\begin{pmatrix} 2 & 2 \\ 0 & 3 \end{pmatrix}$$

Se invece la matrice fosse

$$\begin{pmatrix} 2 & 2 & 1 \\ 3 & 5 & 2 \end{pmatrix}$$

la funzione dovrebbe restituire `[]`.

3. **B_Ex3(file) (8 punti)** Implementare la funzione `B_Ex3(file)` che prende in ingresso il nome di un file, contenente testo privo di punteggiatura. La funzione deve restituire un dizionario avente come chiavi le parole (in lettere minuscole) che compaiono in file e come valori i corrispondenti numeri di occorrenze. Eventuali maiuscole devono essere ignorate, per cui ad esempio "Casa" e "casa" vanno considerate la stessa parola.

Esempio: se il file contenesse il testo:

"Sopra la panca la capra campa sotto la panca la capra muore"

allora la funzione dovrebbe restituire il dizionario `{"sopra": 1, "la": 4, "panca": 2, "capra": 2, "campa": 1, "sotto": 1, "muore": 1}`.

4. **B_Ex4(g, u, v) (8 punti)** Implementare la funzione `B_Ex4(g, u, v)` che (tra gli altri parametri) riceve in ingresso un oggetto `g` della classe `Graph` che rappresenta un grafo diretto, i cui nodi hanno etichette intere. I parametri `u` e `v` sono identificatori di nodi del grafo e sono stringhe che rappresentano etichette intere. I nodi del grafo sono internamente rappresentati come oggetti della classe `Vertex`. Le interfacce delle classi `Graph` e `Vertex` sono descritte sotto.

La funzione deve restituire la somma delle etichette dei nodi che sono raggiungibili sia da `u` che da `v`. Si assuma che un nodo sia sempre raggiungibile a partire da se stesso.

Ad esempio, se il grafo fosse quello della figura sottostante, `u` fosse il nodo 1 e `v` il nodo 3, la funzione dovrebbe restituire 9 (i nodi 3, 4 e 2 sono raggiungibili da entrambi). Se invece `u` fosse il nodo 1 e `v` fosse il nodo 5, la funzione dovrebbe restituire il valore 5 (soltanto 5 è raggiungibile a partire da entrambi).

La classe `Graph` è descritta dalla seguente interfaccia:

```
class Graph:
    def __init__(self):
        ## Inizializza un oggetto di classe Graph vuoto

    def addVertex(self, key):
        ## Aggiunge un vertice dato il suo identificatore (una stringa)

    def getVertex(self, n):
```

```

    ## Dato un identificatore, restituisce il riferimento all'oggetto Vertex
    ## corrispondente se il vertice esiste, None altrimenti

def addEdge(self, f, t, cost=1):
    ## Dati gli identificatori f e t di due vertici, aggiunge l'arco (f, t), creando
    ## gli oggetti di classe Vertex se necessario

def getVertices(self):
    ## Restituisce una sequenza contenente gli identificatori dei vertici

```

I vertici del grafo sono invece oggetti della classe Vertex, avente la seguente interfaccia:

```

class Vertex:
    def __init__(self, key):
        ## Crea un oggetto di classe Vertex, dato l'identificatore del vertice (stringa)

    def addNeighbor(self, nbr, weight=1):
        ## Aggiunge nbr ai vicini del vertice corrente

    def getConnections(self):
        ## Restituisce una sequenza contenente gli identificatori dei vicini del
        ## vertice corrente

    def getId(self):
        ## Restituisce l'identificatore di *questo* vertice

    def getWeight(self, nbr):
        ## Restituisce il peso dell'arco tra *questo* vertice e il vertice avente
        ## identificatore nbr

```

Si noti che probabilmente vi serviranno soltanto alcuni di questi metodi per risolvere il problema