Esercizi su grafi in linguaggio Python

Corso Ottimizzazione Combinatoria (IN440)

Prof. M. Liverani

File in Python

- Un «file di testo sequenziale» è una successione di caratteri in codice ASCII memorizzati sul filesystem del computer
- In Python è possibile creare dei file di testo per memorizzare dei dati (numeri o testi: in generale caratteri alfanumerici) e leggere file di testo, per caricare in memoria i dati presenti sul file

`f = open("nomefile", "w")`	apre il file «*nomefile*» in scrittura (w = *write*); «f» rappresenta l'oggetto file aperto
`f.write(stringa)`	scrive sul file «f» una stringa di caratteri
`f = open("nomefile", "r")`	apre il file «*nomefile*» in lettura (r = *read*); «f» rappresenta l'oggetto file aperto
`x = f.readline()`	legge dal file «f» una intera riga e la memorizza nella variabile x come stringa
`f.close()`	chiude il file «f»: da questo momento non è possibile eseguire operazioni sul file

Stringhe di caratteri in Python

- Sui file di testo vengono scritti e letti dati in formato stringa; è utile quindi eseguire operazioni elementari su una stringa di testo:

`len(a)`	calcola la lunghezza (il numero di caratteri) della stringa contenuta nella variabile **a**
`b = a.strip()`	assegna a **b** la stringa **a** priva di spazi all'inizio e alla fine
`c = a.split()`	crea la lista **c** i cui elementi sono le sottostringhe di **a** separate da spazi
`d = int(a)`	assegna alla variabile **d** il valore numerico intero della stringa **a** (es.: int('123') → 123)
`e = float(a)`	assegna alla variabile **e** il valore numerico floating point della stringa **a**
`g = str(d)`	assegna a **g** una stringa con il caratteri formati dal numero **d** (es.: str(123) → '123')

Lettura e scrittura di un file di testo

- Leggere in input alcune righe di testo e memorizzarle su un file

```
filename = input("Nome del file: ")
f = open(filename, "w")
print("Inserisci il testo; lascia
una riga vuota per terminare:")
x = input("> ")
while x != "":
    f.write(x)
    f.write("\n")
    x = input("> ")
f.close()
```

- Leggere da un file le righe di testo in esso contenute e visualizzarle sullo schermo

```
filename = input("Nome del file: ")
f = open(filename, "r")
i = 0
x = f.readline()
x = x.strip()
while x != "":
        i = i+1
        print(i, ": '", x, "'")
        x = f.readline().strip()
f.close()
print("Lette", i, "righe dal file '",
filename, "'")
```

Le matrici e la libreria NUMPY

- Caricamento libreria: **import numpy as np**
- La libreria Python «**numpy**» fornisce tantissime funzioni matematiche:
  - trigonometriche: np.**sin**(x), np.**cos**(x), np.**tan**(x), ...
  - iperboliche: np.**sinh**(x), np.**cosh**(x), np.**tanh**(x), ...
  - di arrotondamento: np.**around**(x, *decimali*), np.**trunc**(x), ...
  - esponenziali e logaritmiche: np.**exp**(x), np.**exp2**(x), np.**log**(x), np.**log10**(x), np.**log2**(x), ...
  - ricerca di valori estremali: np.**maximum**(x_1, x_2, ..., x_n), np.**minimum**(x_1, x_2, ..., x_n), ...
- Consente un utilizzo più semplice di array multi-dimensionali:
  - matrice vuota con *k* dimensioni: A = np.**empty**((n_1, n_2, ..., n_k), dtype=float)
  - matrice nulla con *k* dimensioni: A = np.**zeros**((n_1, n_2, ..., n_k), dtype=int)
  - matrice con *k* dimensioni e valori tutti uguali a *x*: A = np.**full**((n_1, n_2, ..., n_k), x)
  - la numerazione degli indici degli elementi delle matrici parte za zero: elemento nella terza riga e quinta colonna della matrice A: A[2][4]

Matrice di adiacenza su file

- Generare un grafo random e salvare la matrice di adiacenza su file

```
from in440 import *
n = int(input("Numero di vertici: "))
p = float(input("Probabilita': "))
filename = input("Nome file: ")
G = Graph()
randomGraph(G, n, p)
printGraph(G)
A = np.zeros((n,n))
for u in G:
   for v in u.getConnections():
      A[u.getId()-1][v.getId()-1] = 1
f = open(filename, "w")
for i in range(0, n):
   for j in range(0,n):
      f.write(str(int(A[i][j])))
      f.write(" ")
   f.write("\n")
f.close()
```

È possibile importare la matrice di adiacenza di un grafo su Mathematica:

```
A = Import["nomefile", "Table"]
G = AdjacencyGraph[A, VertexLabels -> Automatic]
```

Matrice di adiacenza da file

- Caricare in memoria un grafo a partire da una matrice di adiacenza memorizzata su file (prodotta da un altro programma Python o da Mathematica)

```
from in440 import *
filename = input("Nome file: ")
f.open(filename, "r")
i = 0
x = f.readline().strip()
n = len(x.split())
A = np.empty((n,n), dtype=int)
while x != "":
   A[i] = x.split()
   i = i+1
   x = f.readline().strip()
f.close()
```

segue ↗

continua ↘

```
G = Graph()
for i in range(n):
   G.addVertex(i+1)
for i in range(n):
   for j in range(n):
      if A[i][j] == 1: G.addEdge(i+1, j+1)
printGraph(G)
graphPlot(G)
```

È possibile esportare la matrice di adiacenza di un grafo su Mathematica:

```
G = RandomGraph[{10,20}]
A = AdjacencyMatrix[G]
Export["grafo.txt", A, "Table"]
```