Esercizi su grafi in linguaggio Python

Corso Ottimizzazione Combinatoria (IN440)

Prof. M. Liverani

Visita di un grafo random

Costruzione di un grafo random G con n vertici e probabilità p che siano presenti gli spigoli

Implementare un algoritmo di visita utilizzando una **coda** e poi una **pila** per rappresentare la struttura dati Q e costruire i due alberi di visita T_1 e T_2

Confrontare gli alberi di visita ottenuti nei due casi

Algoritmo 11 VISITA(G, s)

Input: Il grafo G ed una sorgente $s \in V(G)$
Output: La sequenza di vertici visitati su G a partire da s

```
1: sia Q = ⟨s⟩ una coda o una pila, sia T = ({s}, ∅) un albero
2: per ogni v ∈ V(G) ripeti
3:     colore(v) = bianco
4: fine-ciclo
5: colore(s) = grigio
6: fintanto che Q ≠ ∅ ripeti
7:     sia u il primo elemento di Q
8:     per ogni v ∈ N(u) ripeti
9:         se colore(v) = bianco allora
10:            colore(v) = grigio
11:            aggiungi v a Q
12:            aggiungi il vertice v e lo spigolo (u, v) all'albero T
13:        fine-condizione
14:        sia u il primo elemento di Q
15:    fine-ciclo
16:    estrai da Q il primo elemento u
17:    colore(u) = nero
18: fine-ciclo
```

Visita di un grafo random

```
from pythonds import *
from graphics import *
from random   import *
import numpy as np

def randomGraph(g, n, p):
    ...

def printGraph(g):
    ...

def plotGraph(g):
    ...

def visitaBFS(g, t, s):
    ...

def visitaDFS(g, t, s):
    ...
```

```
from IN440LIB import *

G = Graph()
T1 = Graph()
T2 = Graph()
n = int(input("N. vertici:"))
p = float(input("Probabilità:"))
randomGraph(G, n, p)
visitaBFS(G, T1, 1)
visitaDFS(G, T2, 1)

printGraph(G)
plotGraph(G)

printGraph(T1)
plotGraph(T1)

printGraph(T2)
plotGraph(T2)
```

Cammini di costo minimo tra tutte le coppie di vertici

Costruire un grafo random G con pesi $w_{i,j} < 100$ assegnati agli spigoli in modo casuale.
Calcolare il costo del cammino di costo minimo tra ogni coppia di vertici $v_i, v_j \in V(G)$

Algoritmo 24 FLOYDWARSHALL(G, w)

Input: Un grafo G orientato con funzione peso $w : E(G) \to \mathbb{R}$

Output: La matrice dei pesi dei cammini minimi per ciascuna coppia di vertici in $V(G)$

1: $W := (w_{ij})$ con $w_{ij} := \begin{cases} 0 & \text{se } i = j \\ w(i,j) & \text{se } (i,j) \in E(G) \\ \infty & \text{altrimenti} \end{cases}$

2: $D^{(0)} := W$

3: **per** $k := 1, \ldots, n$ **ripeti**

4: **per** $i := 1, \ldots, n$ **ripeti**

5: **per** $j := 1, \ldots, n$ **ripeti**

6: $d_{ij}^{(k)} := \min\left\{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \right\}$

7: **fine-ciclo**

8: **fine-ciclo**

9: **fine-ciclo**

Cammini di costo minimo tra tutte le coppie di vertici

```
def definisciD(G, D):
   for i in range(1, n+1):
      for j in range(1, n+1):
         if i == j: D[0][i][j] = 0
         else:      D[0][i][j] = np.Inf
   for u in G:
      for v in u.getConnections():
         D[0][u.getId()][v.getId()] = u.getWeight(v)
   return
```

```
from IN440LIB import *

G = Graph()
n = int(input("N. vertici:"))
p = float(input("Probabilità:"))
W = int(input("Massimo peso:"))

randomWeightedGraph(G, n, p, W)
D = np.empty((n+1, n+1, n+1), dtype=float)
definisciD(G, D)
for k in range(1, n+1):
   for i in range(1, n+1):
      for j in range(1, n+1):
         if D[k-1][i][j] < D[k-1][i][k] + D[k-1][k][j]:
            D[k][i][j] = D[k-1][i][j]
         else:
            D[k][i][j] = D[k-1][i][k] + D[k-1][k][j]
print(D[n])
```