Cammini di costo minimo tra tutte le coppie di vertici in linguaggio Python

Corso Ottimizzazione Combinatoria (IN440)

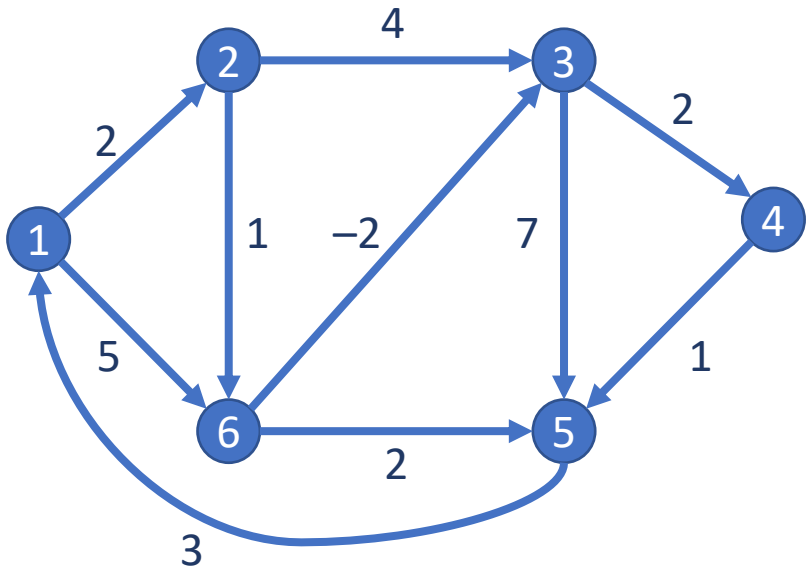
Prof. M. Liverani

Cammini di costo minimo

- Dato un grafo casuale con n vertici e pesi non negativi assegnati agli spigoli, si vuole trovare il cammino di costo minimo tra tutte le coppie di vertici del grafo
- Dopo aver trovare tutti i cammini, dato un vertice u di partenza ed uno v di arrivo, visualizzare, se esiste, la sequenza di vertici che compongono il cammino di costo minimo $p : u \rightsquigarrow v$
- Utilizzeremo l'**algoritmo di Floyd-Warshall**, che utilizza la tecnica della programmazione dinamica

Dati in input

- Un grafo $G = (V, E)$
- Una funzione w che assegna pesi agli spigoli del grafo (i pesi possono essere anche negativi, ma non sono consentiti cicli di costo negativo) $w : E(G) \to \mathbf{R}$



- Sia $V(G) = \{1, 2, 3, \ldots, n\}$

Output dell'algoritmo

- Una matrice quadrata $n \times n$ con i costi dei cammini minimi per ogni coppia di vertici del grafo:

  $D^{(n)}$ con $d_{i,j}^{(n)}$ pari al costo di un cammino di costo minimo da i a j

- Indichiamo con $D^{(k)}$ la matrice che riporta il **costo del cammino minimo** per ogni coppia di vertici i e j con vertici intermedi scelti nell'insieme $\{1, 2, 3, ..., k\}$

Algoritmo di Floyd-Warshall

- La matrice $D^{(0)}$ rappresenta il costo del cammino da i a j con nessun vertice intermedio: quindi è basata sul costo $w_{i,j}$ assegnato ad ogni spigolo del grafo (infinito se lo spigolo non esiste)
- L'algoritmo costruisce una sequenza di matrici: si passa dalla matrice $D^{(k-1)}$ alla matrice $D^{(k)}$ selezionando il minimo tra il costo del cammino minimo da i a j con vertici in $\{1, ..., k-1\}$ e il costo del cammino di costo minimo da i a j passante per k con vertici intermedi in $\{1, ..., k-1\}$

Algoritmo 28 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: $d_{ij}^{(k)} := \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$

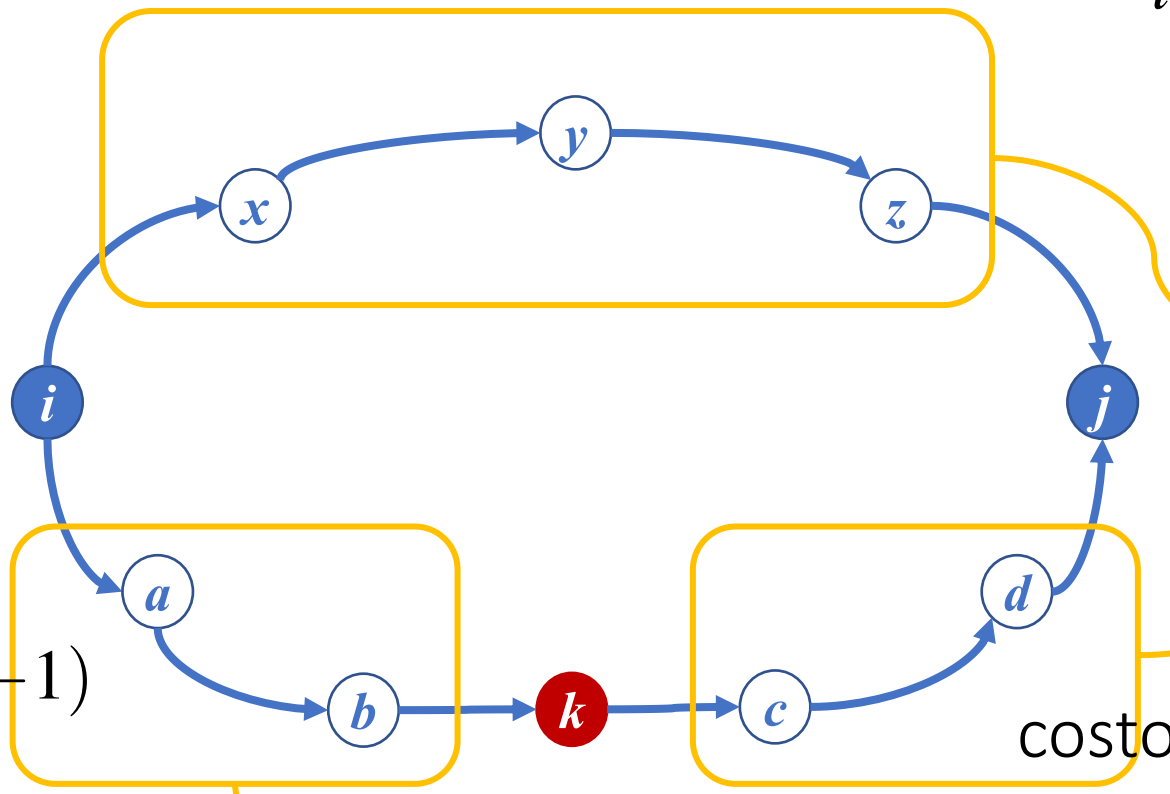
7: **fine-ciclo**

8: **fine-ciclo**

9: **fine-ciclo**

Algoritmo di Floyd-Warshall

costo: $d_{i,j}^{(k-1)}$



scelti in $\{1,\ldots,k-1\}$

costo: $d_{i,k}^{(k-1)}$

costo: $d_{k,j}^{(k-1)}$

costo: $d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}$

Quindi:

$$d_{i,j}^{(k)} = \min \begin{cases} d_{i,j}^{(k-1)} \\ d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \end{cases}$$

Inizializzazione

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** $k := 1,2,3,4,5,6$ **ripeti**

4: **per** $i := 1,2,3,4,5,6$ **ripeti**

5: **per** $j := 1,2,3,4,5,6$ **ripeti**

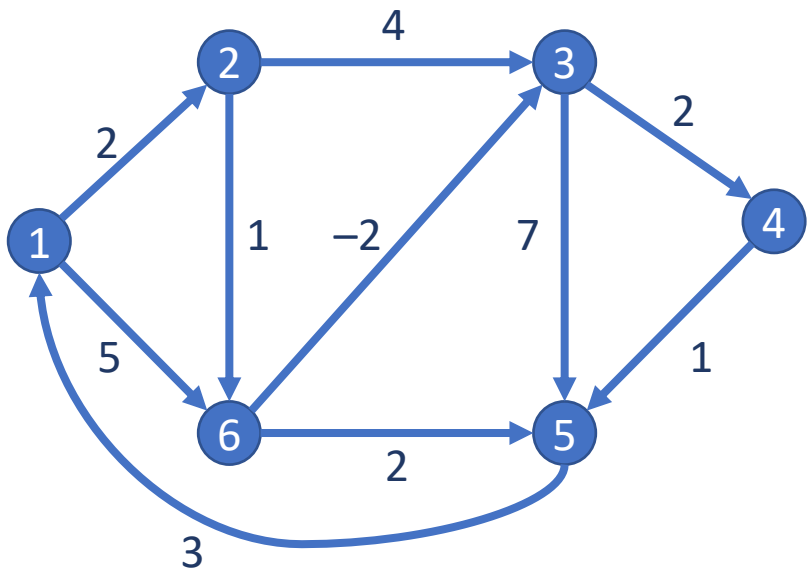
6: $d_{ij}^{(k)} := \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$

7: **fine-ciclo**

8: **fine-ciclo**

9: **fine-ciclo**

- Viene inizializzata la matrice $D^{(0)}$ con i dati della matrice W



$D^{(0)}$

	1	2	3	4	5	6
1	0	2	∞	∞	∞	5
2	∞	0	4	∞	∞	1
3	∞	∞	0	2	7	∞
4	∞	∞	∞	0	1	∞
5	3	∞	∞	∞	0	∞
6	∞	∞	−2	∞	2	0

Cammini di costo minimo con vertici intermedi in {1}

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** $k := 1,2,3,4,5,6$ **ripeti**

4: **per** $i := 1,2,3,4,5,6$ **ripeti**

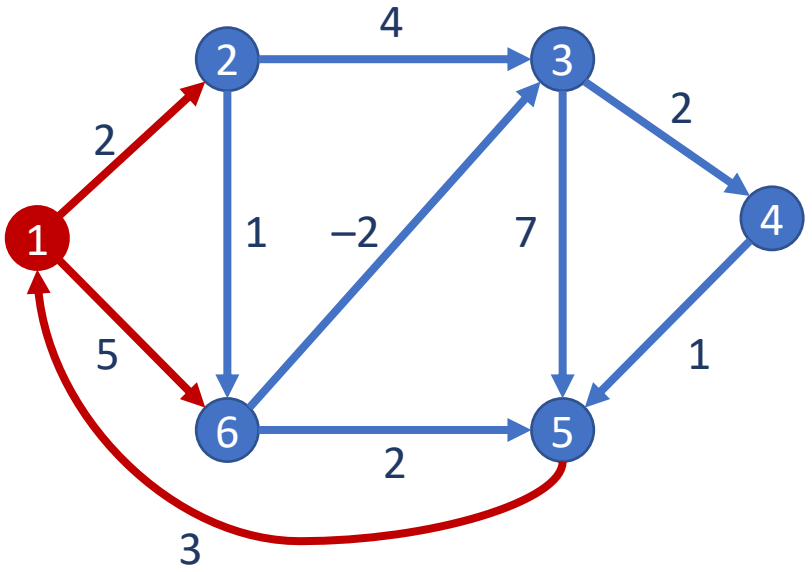
5: **per** $j := 1,2,3,4,5,6$ **ripeti**

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**



- Viene calcolata la matrice $D^{(1)}$ con i costi degli eventuali cammini di costo minimo con vertici intermedi in $\{1\}$

$D^{(1)}$

	1	2	3	4	5	6
1	0	2	∞	∞	∞	5
2	∞	0	4	∞	∞	1
3	∞	∞	0	2	7	∞
4	∞	∞	∞	0	1	∞
5	3	**5**	∞	∞	0	**8**
6	∞	∞	−2	∞	2	0

Cammini di costo minimo con vertici intermedi in {1, 2}

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** $k := 1,2,3,4,5,6$ **ripeti**

4: **per** $i := 1,2,3,4,5,6$ **ripeti**

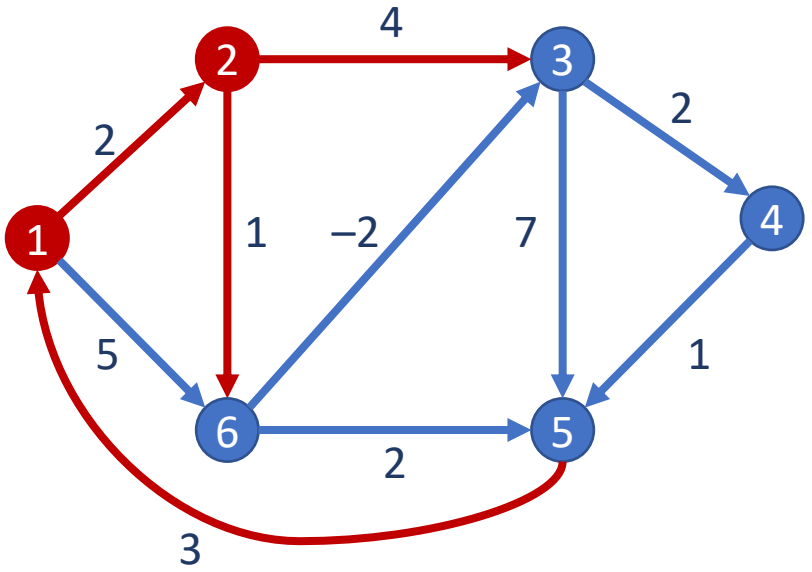
5: **per** $j := 1,2,3,4,5,6$ **ripeti**

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**



- Viene calcolata la matrice $D^{(2)}$ con i costi degli eventuali cammini di costo minimo con vertici intermedi in $\{1, 2\}$

$D^{(2)}$

	1	2	3	4	5	6
1	0	2	**6**	∞	∞	**3**
2	∞	0	4	∞	∞	1
3	∞	∞	0	2	7	∞
4	∞	∞	∞	0	1	∞
5	3	5	**9**	∞	0	**6**
6	∞	∞	−2	∞	2	0

Cammini di costo minimo con vertici intermedi in {1, 2, 3}

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** $k := 1,2,3,4,5,6$ **ripeti**

4: **per** $i := 1,2,3,4,5,6$ **ripeti**

5: **per** $j := 1,2,3,4,5,6$ **ripeti**

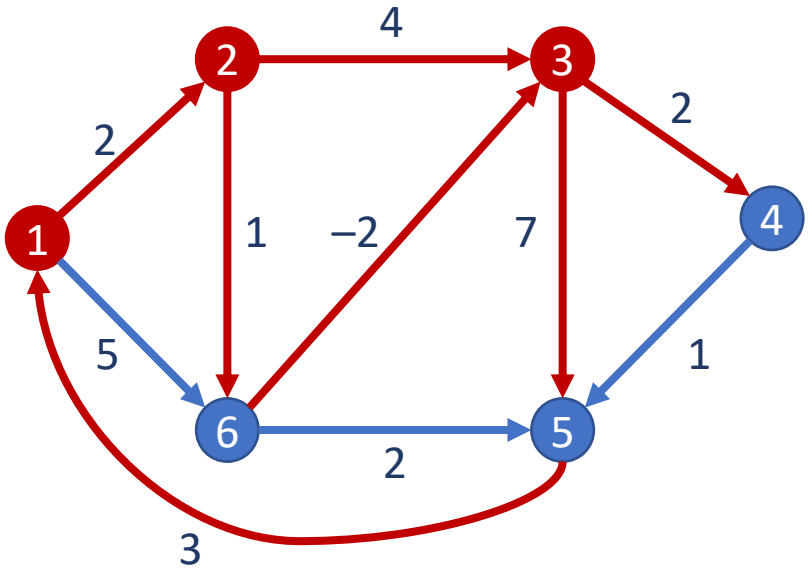
6: $d_{ij}^{(k)} := \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$

7: **fine-ciclo**

8: **fine-ciclo**

9: **fine-ciclo**

- Viene calcolata la matrice $D^{(3)}$ con i costi degli eventuali cammini di costo minimo con vertici intermedi in $\{1, 2, 3\}$



$D^{(3)}$

	1	2	3	4	5	6
1	0	2	6	**8**	**13**	3
2	∞	0	4	**6**	**11**	1
3	∞	∞	0	2	7	∞
4	∞	∞	∞	0	1	∞
5	3	5	9	**11**	0	6
6	∞	∞	−2	**0**	2	0

Cammini di costo minimo con vertici intermedi in {1, 2, 3, 4}

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** $k := 1,2,3,4,5,6$ **ripeti**

4: **per** $i := 1,2,3,4,5,6$ **ripeti**

5: **per** $j := 1,2,3,4,5,6$ **ripeti**

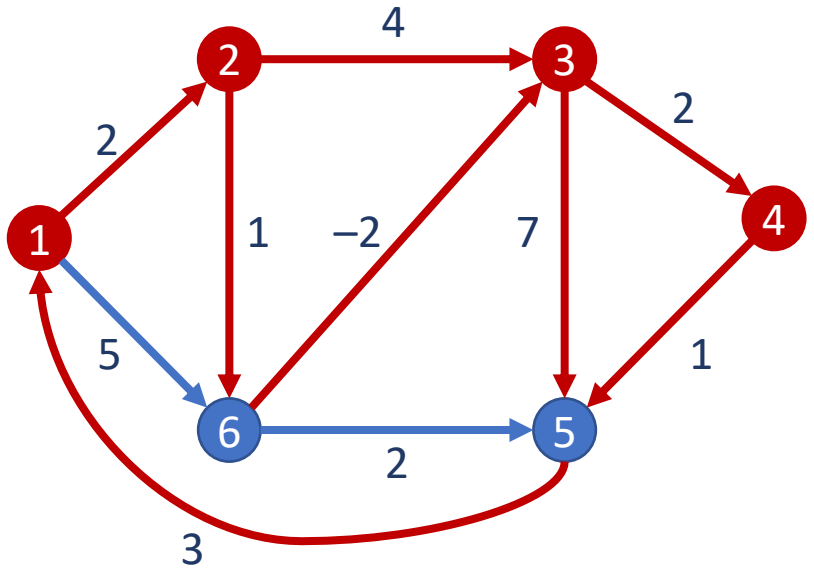
6: $d_{ij}^{(k)} := \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$

7: **fine-ciclo**

8: **fine-ciclo**

9: **fine-ciclo**

- Viene calcolata la matrice $D^{(4)}$ con i costi degli eventuali cammini di costo minimo con vertici intermedi in $\{1, 2, 3, 4\}$



$D^{(4)}$

	1	2	3	4	5	6
1	0	2	6	8	**9**	3
2	∞	0	4	6	**7**	1
3	∞	∞	0	2	**3**	∞
4	∞	∞	∞	0	1	∞
5	3	5	9	11	0	6
6	∞	∞	−2	0	**1**	0

Cammini di costo minimo con vertici intermedi in {1, 2, 3, 4, 5}

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** $k := 1,2,3,4,5,6$ **ripeti**

4: **per** $i := 1,2,3,4,5,6$ **ripeti**

5: **per** $j := 1,2,3,4,5,6$ **ripeti**

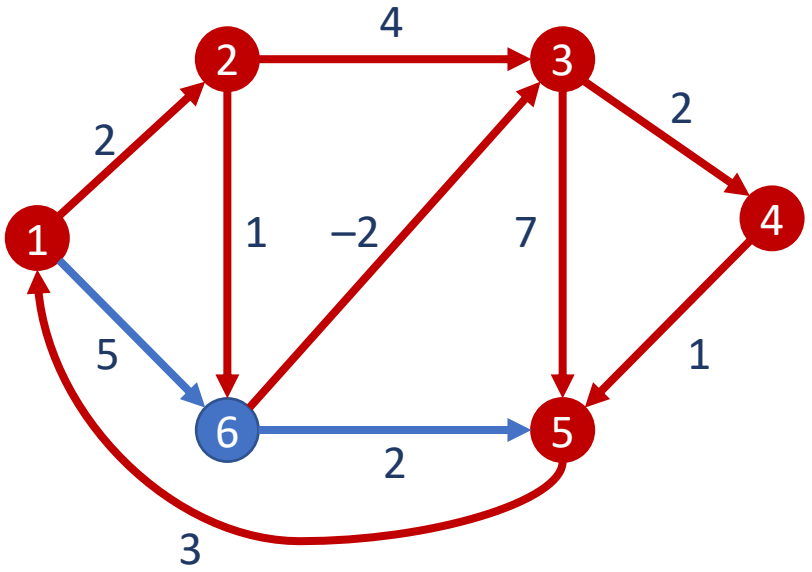
6: $d_{ij}^{(k)} := \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$

7: **fine-ciclo**

8: **fine-ciclo**

9: **fine-ciclo**

- Viene calcolata la matrice $D^{(5)}$ con i costi degli eventuali cammini di costo minimo con vertici intermedi in $\{1, 2, 3, 4, 5\}$



$D^{(5)}$

	1	2	3	4	5	6
1	0	2	6	8	9	3
2	**10**	0	4	6	7	1
3	**6**	**8**	0	2	3	**9**
4	**4**	**6**	**10**	0	1	**7**
5	3	5	9	11	0	6
6	**4**	**6**	−2	0	1	0

Cammini di costo minimo con vertici intermedi in {1, 2, 3, 4, 5, 6}

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** $k := 1,2,3,4,5,6$ **ripeti**

4: **per** $i := 1,2,3,4,5,6$ **ripeti**

5: **per** $j := 1,2,3,4,5,6$ **ripeti**

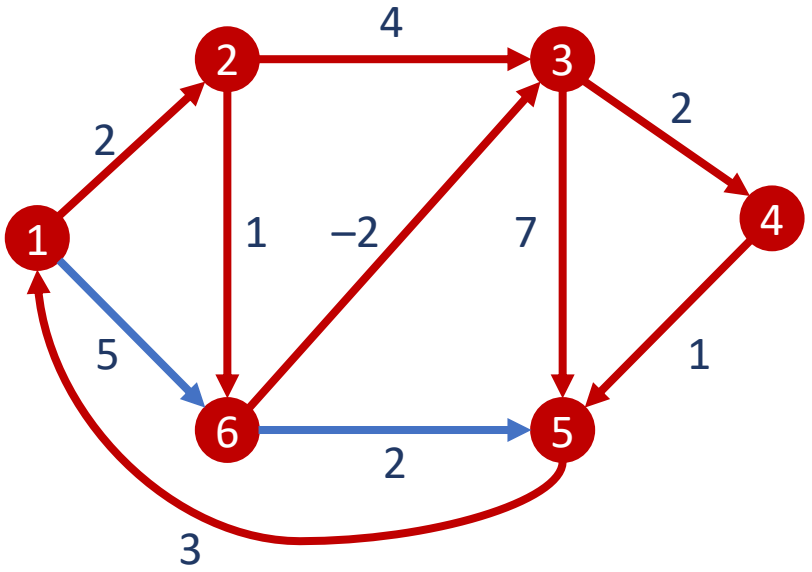
6: $d_{ij}^{(k)} := \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$

7: **fine-ciclo**

8: **fine-ciclo**

9: **fine-ciclo**

- Viene calcolata la matrice $D^{(6)}$ con i costi degli eventuali cammini di costo minimo con vertici intermedi in $\{1, 2, 3, 4, 5, 6\}$



$D^{(6)}$

	1	2	3	4	5	6
1	0	2	**1**	**3**	**4**	3
2	**6**	0	**–1**	**1**	**2**	1
3	6	8	0	2	3	9
4	4	6	**5**	0	1	7
5	3	5	**4**	**6**	0	6
6	4	6	–2	0	1	0

Cammini di costo minimo con vertici intermedi in {1, 2, 3, 4, 5, 6}

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** $k := 1,2,3,4,5,6$ **ripeti**

4: **per** $i := 1,2,3,4,5,6$ **ripeti**

5: **per** $j := 1,2,3,4,5,6$ **ripeti**

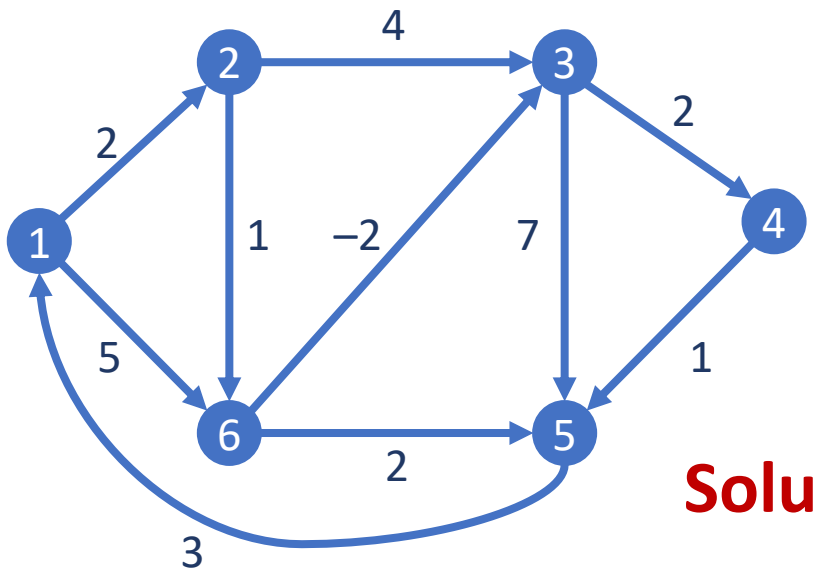
6: $d_{ij}^{(k)} := \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$

7: **fine-ciclo**

8: **fine-ciclo**

9: **fine-ciclo**

- Viene calcolata la matrice $D^{(6)}$ con i costi degli eventuali cammini di costo minimo con vertici intermedi in $\{1, 2, 3, 4, 5, 6\}$



Soluzione!

$n = 6$

$D^{(6)}$

	1	2	3	4	5	6
1	0	2	1	3	4	3
2	6	0	−1	1	2	1
3	6	8	0	2	3	9
4	4	6	5	0	1	7
5	3	5	4	6	0	6
6	4	6	−2	0	1	0

Grafi random con pesi random

- Per il problema del cammino di costo minimo dobbiamo gestire un grafo (casuale) con pesi arbitrari assegnati agli spigoli
- Per creare gli spigoli del grafo usiamo il metodo `addEdge(u, v, w)` reso disponibile dalla libreria `pythonds`, che ci permette di specificare anche il peso w da attribuire allo spigolo (u, v) del grafo che stiamo costruendo
- Definiamo quindi una funzione `randomWeightedGraph`, simile alla funzione `randomGraph`, che aggiungiamo alla libreria `in440.py`

```
def randomWeightedGraph(G, n, P, Wmax):
  for v in range(1,n+1):
    G.addVertex(v)
  for u in range(1,n):
    for v in range(u+1,n+1):
      x = random()
      if u != v and x>0 and x<=P:
        w = randint(1, Wmax)
        G.addEdge(u,v, w)
return
```

Algoritmo di Floyd-Warshall

- Possiamo implementare l'algoritmo precisando con maggiore dettaglio alcune operazioni e calcolando oltre alla matrice dei costi anche la matrice dei predecessori, utile poi per ricostruire i cammini di costo minimo, fissato il vertice di partenza u e quello di arrivo v

Algoritmo 29 FLOYDWARSHALL(G, w)

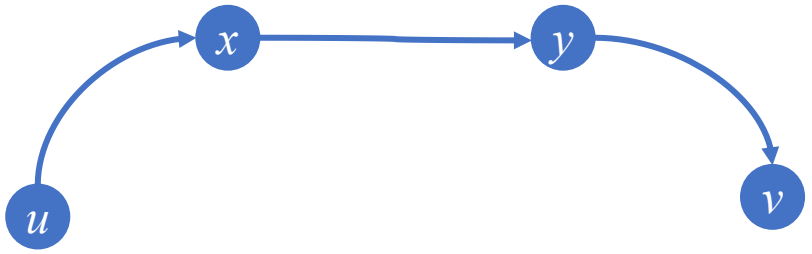
Input: Un grafo G orientato con funzione peso $w : E(G) \to \mathbb{R}$
Output: La matrice dei pesi dei cammini minimi per ciascuna coppia di vertici in $V(G)$

1: $W := (w_{ij})$ con $w_{ij} := w(i,j)$ per ogni $i, j = 1, 2, \ldots, n$
2: $D^{(0)} := W$
3: **per** $u := 1, 2, \ldots, n$ **ripeti**
4: **per** $v := 1, 2, \ldots, n$ **ripeti**
5: **se** $u = v$ o $w_{uv} = \infty$ **allora**
6: $\pi_{uv}^{(0)} := null$
7: **altrimenti**
8: $\pi_{uv}^{(0)} := u$
9: **fine-condizione**
10: **fine-ciclo**
11: **fine-ciclo**
12: **per** $k := 1, 2, \ldots, n$ **ripeti**
13: **per** $u := 1, 2, \ldots, n$ **ripeti**
14: **per** $v := 1, 2, \ldots, n$ **ripeti**
15: **se** $d_{uv}^{(k-1)} \leq d_{uk}^{(k-1)} + d_{kv}^{(k-1)}$ **allora**
16: $\pi_{uv}^{(k)} := \pi_{uv}^{(k-1)}, d_{uv}^{(k)} = d_{uv}^{(k-1)}$
17: **altrimenti**
18: $\pi_{uv}^{(k)} := \pi_{kv}^{(k-1)}, d_{uv}^{(k)} = d_{uk}^{(k-1)} + d_{kv}^{(k-1)}$
19: **fine-condizione**
20: **fine-ciclo**
21: **fine-ciclo**
22: **fine-ciclo**

Visualizzazione del cammino di costo minimo

- Per visualizzare il cammino di costo minimo dal vertice u al vertice v utilizziamo la matrice $P^{(n)}$ prodotta dall'algoritmo
- $P^{(n)}{}_{u,\,v}$ indica il predecessore (il padre) del vertice v nel cammino di costo minimo per arrivare da u al vertice v
- Possiamo quindi costruire la seguente procedura ricorsiva

```
def stampaCammino(u, v, P, n):
  if u != v:
    stampaCammino(u, P[n][u][v], P, n)
    print(v, "-->", end="")
  else:
    print()
  return
```



$$P^{(n)}{}_{u,v} = y \quad P^{(n)}{}_{u,y} = x \quad P^{(n)}{}_{u,x} = u$$

Codifica dell'algoritmo in Python

```
from in440 import *

def stampaCammino(u, v, P, n):
  if u != v:
    stampaCammino(u,P[n][u][v],P,n)
    print(v, "-->", end="")
  else:
    print()

g = Graph()
n = int(input("Numero di vertici: "))
p = float(input("Probabilita': "))
w = int(input("Peso massimo: "))
randomWeightedDiGraph(g, n, p, w)

D = np.full((n+1, n+1, n+1), np.Inf, dtype=float)
P = np.zeros((n+1, n+1, n+1), dtype=int)

for u in g:
  D[0][u.getId()][u.getId()] = 0
  for v in u.getConnections():
    D[0][u.getId()][v.getId()] = u.getWeight(v)
    P[0][u.getId()][v.getId()] = u.getId()
```

```
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]:
        P[k][i][j] = P[k-1][i][j]
        D[k][i][j] = D[k-1][i][j]
      else:
        P[k][i][j] = P[k-1][k][j]
        D[k][i][j] = D[k-1][i][k] + D[k-1][k][j]

printGraph(g)
print("Matrice dei pesi:\n", D[0])
print("Matrice costi di cammino minimo:\n", D[n])
print("Matrice dei cammini minimi:\n", P[n])

a = int(input("Vertice di partenza:"))
b = int(input("Vertice di arrivo:  "))
stampaCammino(a, b, P, n)
print()
plotGraph(g, "Grafo", "orientato", True)
```