Capitolo 12

Cammini minimi fra tutte le coppie

Consideriamo il problema dei cammini minimi fra tutte le coppie in un grafo $G = (V, E, w)$ orientato, pesato, dove possono essere presenti archi (ma non cicli) di peso negativo; assumiamo, inoltre, che gli n vertici siano identificati da numeri interi da 1 a n.

12.1 Cammini minimi e moltiplicazione di matrici

Matrice di adiacenza pesata

Si tratta di una matrice $n \times n$ che, oltre a specificare se esiste l'arco (i, j), indica anche il peso di tale arco.

$$w_{i,j} = \begin{cases} 0 & \text{se } i = j \\ w(i,j) & \text{se } i \neq j \wedge (i,j) \in E \\ +\infty & \text{se } i \neq j \wedge (i,j) \notin E \end{cases}$$

Matrice delle distanze

È una matrice $n \times n$, aggiornata dall'algoritmo, dove $d_{i,j}$ contiene la stima della distanza tra i e j; al termine dell'esecuzione dell'algoritmo, vogliamo che $d_{i,j} = \delta(i,j), \forall i, j \in V$.

Moltiplicazione di matrici

Siano $i, j \in V$; definiamo:

$$\mathcal{C}_{i,j} = \{p = \langle x_0, ..., x_q \rangle : x_0 = i \wedge x_q = j \wedge (x_{l-1}, x_l) \in E, l = 1..q\}$$

La distanza tra i e j può essere definita come segue:

$$\delta(i,j) = \min_{p \in \mathcal{C}(i,j)} w(p)$$

Fissiamo $m \geq 1$:

$$\mathcal{C}_{i,j}^{(m)} = \{p = \langle x_0, ..., x_q \rangle : x_0 = i \wedge x_q = j \wedge (x_{l-1}, x_l) \in E, l = 1..q \wedge q \leq m\}$$

$$d_{i,j}^{(m)} = \min_{p \in \mathcal{C}_{i,j}^{(m)}} w(p)$$

Sono valide le seguenti relazioni:

$$\mathcal{C}_{i,j}^{(1)} \subseteq \mathcal{C}_{i,j}^{(2)} \subseteq ... \subseteq \mathcal{C}_{i,j}^{(m)} \subseteq \mathcal{C}_{i,j}^{(m+1)} \subseteq ... \subseteq \mathcal{C}_{i,j}$$

$$\mathcal{C}_{i,j} = \bigcup_{m \in \mathbb{N}} \mathcal{C}_{i,j}^{(m)}$$

$$d_{i,j}^{(1)} \geq d_{i,j}^{(2)} \geq ... \geq d_{i,j}^{(m)} \geq ... \geq \delta(i,j)$$

Poiché non sono presenti, per ipotesi, cicli negativi, possiamo assumere che i cammini minimi siano cammini semplici, dunque formati al più da $n-1$ archi; ossia, si ha $d_{i,j}^{(n-1)} = d_{i,j}^{(n)} = d_{i,j}^{(n+1)} = ... = \delta(i,j)$. L'idea è di progettare un algoritmo per il calcolo di $d_{i,j}^{(m)}$ in modo che, quando arriviamo a $d_{i,j}^{(n-1)}$, otteniamo proprio le distanze; cerchiamo una formula ricorsiva per il calcolo di $d_{i,j}^{(m)}$, per induzione.

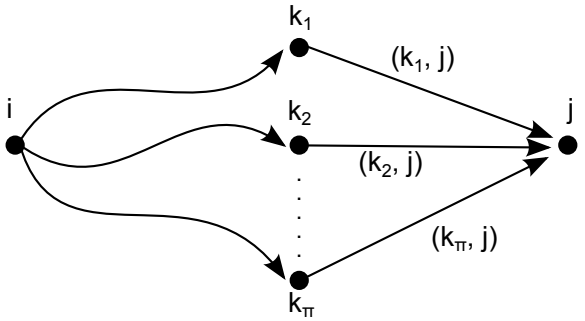
Caso base: si ha $m = 1$

$$\mathcal{C}_{i,j}^{(1)} = \begin{cases} \{\langle i \rangle\} & \text{se } i = j \\ \{\langle i,j \rangle\} & \text{se } i \neq j \wedge (i,j) \in E \\ \emptyset & \text{se } i \neq j \wedge (i,j) \notin E \end{cases}$$

$$d_{i,j}^{(1)} = \begin{cases} 0 & \text{se } i = j \\ w(i,j) & \text{se } i \neq j \wedge (i,j) \in E \\ +\infty & \text{se } i \neq j \wedge (i,j) \notin E \end{cases}$$

Ipotesi induttiva: supponiamo di essere riusciti a calcolare la matrice delle distanze $D^{(m-1)}$).

Passo induttivo: sia $m > 1$ e proviamo a calcolare $D^{(m)}$ usando l'ipotesi. Prendiamo $i, j \in V$: $p \in \mathcal{C}_{i,j}^{(m)}$ può essere scomposto come $p = p', j$, con $p' \in \mathcal{C}_{i,k}^{(m-1)}$; in generale, i cammini minimi in $\mathcal{C}_{i,j}^{(m)}$ sono scomponibili come segue:



da cui:

$$d_{i,j}^{(m)} = \min_{p \in \mathcal{C}_{i,j}^{(m)}} w(p) = \min_{k \in V \text{ e } (k,j) \in E} \left\{ d_{i,k}^{(m-1)} + w(k,j) \right\}$$

Sia $h \in V$: se $(h,j) \notin E$, allora $d_{i,h}^{(m-1)} + w(h,j) = +\infty$.

La formula cercata è la seguente:

$$d_{i,j}^{(m)} = \min_{k \in V} \left\{ d_{i,k}^{(m-1)} + w(k,j) \right\}$$

Per implementare la formula ricorsiva, definiamo la seguente funzione:

funzione `extendShortestPath` (*matrice delle distanze* $D^{(m-1)}$*, matrice di adiacenza pesata* W) $\rightarrow$ *matrice delle distanze* $D^{(m)}$

```
1 n = rows(D)
2 Dm : matrice[n][n]
3 for i = 1 to n do
4   for j = 1 to n do
5     Dm[i][j] = + ∞
6     for k = 1 to n do
7       Dm[i][j] = min(Dm[i][j], D[i][k] + W[k][j])
8 return Dm
```

Complessità: $\Theta(n^3)$.

La funzione definisce un nuovo tipo di prodotto tra matrici $D \otimes W = D'$, dove D' è la matrice calcolata da `extendShortestPath`:

algoritmo `showAllPairsShortestPath` (*matrice di adiacenza pesata* W) $\rightarrow$ *matrice delle distanze* $D^{(n-1)}$

```
1 n = rows(W)
2 D(1) = W
3 for m = 2 to n - 1 do
4   D(m) = D(m - 1) ⊗ W
5 return D(n - 1)
```

Complessità: $\Theta(n^4)$.

12.1.1 Ottimizzazione

L'algoritmo `showAllPairsShortestPath` calcola le matrici $D^{(m)}$ in maniera incrementale:

$$\begin{array}{rcl} D^{(1)} & = & W \\ D^{(2)} & = & D^{(1)} \otimes W \quad = \quad W^2 \\ D^{(3)} & = & D^{(2)} \otimes W \quad = \quad W^3 \\ & \vdots & \\ D^{(n-1)} & = & D^{(n-2)} \otimes W \quad = \quad W^{(n-1)} \end{array}$$

Quello che interessa è avere la matrice $W^{(n-1)}$. Osserviamo che, in assenza di cicli negativi, $W^{(n-1)} = W^{(n)} = W^{(n+1)} = \ldots$ Ci basta quindi ottenere una potenza W^m per qualche $m \geq (n-1)$. Modifichiamo il nostro punto di vista e calcoliamo le potenze in questo modo:

$$\begin{array}{rcl} W^2 & = & W \otimes W \\ W^4 & = & W^2 \otimes W^2 \\ W^8 & = & W^4 \otimes W^4 \\ & \vdots & \\ W^{2m} & = & W^m \otimes W^m \end{array}$$

... fino a superare $(n-1)$.

algoritmo `fasterAllPairsShortestPath` (*matr di adiacenza pesata* W) $\rightarrow$ *matr delle distanze* $D^{(n-1)}$

```
1 n = rows(W)
2 D(1) = W
3 m = 1
4 while m < n - 1 do
5     D(2m) = D(m) ⊗ D(m)
6     m = 2m
7 return D(m)
```

Complessità: $\Theta(n^3 \log n)$. [Esericizio]

12.2 Algoritmo di Floyd-Warshall

Consideriamo un grafo che goda delle proprietà descritte ad inizio capitolo; vogliamo determinare $\delta(i,j)$ per ogni coppia i, j. Costruiamo $\mathcal{C}_{i,j}$ in maniera incrementale; sia $0 \leq k \leq n$ e definiamo:

$$P_{i,j}^{(k)} = \{p = \langle x_0, ..., x_q \rangle : x_0 = i \wedge x_q = j \wedge (x_{l-1}, x_l) \in E, l = 1..q \wedge x_h \leq k, h = 1..(q-1)\}$$

Si tratta dell'*insieme dei cammini* da i a j i cui *nodi interni* sono in $1..k$; osserviamo che $P_{i,j}^{(n)} = \mathcal{C}_{i,j}$ quindi, se definiamo:

$$d_{i,j}^{(k)} = \min_{p \in P_{i,j}^{(k)}} w(p)$$

si ha:

$$d_{i,j}^{(n)} = \delta(i,j)$$

L'obiettivo è calcolare $d_{i,j}^{(n)}$ per ogni coppia di vertici in V; procediamo per induzione su k.

Caso base: si ha $k = 0$; $P_{i,j}^{(0)}$ è l'insieme dei cammini da i a j che *non* hanno nodi interni, dunque:

$$P_{i,j}^{(0)} = \begin{cases} \{\langle i \rangle\} & \text{se } i = j \\ \{\langle i,j \rangle\} & \text{se } i \neq j \wedge (i,j) \in E \\ \emptyset & \text{se } i \neq j \wedge (i,j) \notin E \end{cases}$$

$$d_{i,j}^{(0)} = \begin{cases} 0 & \text{se } i = j \\ w(i,j) & \text{se } i \neq j \wedge (i,j) \in E \\ +\infty & \text{se } i \neq j \wedge (i,j) \notin E \end{cases}$$

Se chiamiamo $D^{(k)} = (d_{i,j}^{(k)})$, si ha $D^{(0)} = W$.

Ipotesi induttiva: supponiamo di conoscere $P_{i,j}^{(k-1)}$.

Passo induttivo: sia $k \geq 1$; usando l'ipotesi induttiva, cerchiamo di ottenere $P_{i,j}^{(k)}$. Innanzitutto, osserviamo che $P_{i,j}^{(k-1)} \subseteq P_{i,j}^{(k)}$ e definiamo $\widehat{P}_{i,j}^{(k)} = P_{i,j}^{(k)} - P_{i,j}^{(k-1)} = \left\{ p \in P_{i,j}^{(k)} : p \text{ passa per il vertice } k \right\}$. Si ha:

$$\begin{aligned} d_{i,j}^{(k)} &= \min_{p \in P_{i,j}^{(k)}} w(p) \\ &= \min\left(\min_{p \in P_{i,j}^{(k-1)}} w(p), \min_{p \in \widehat{P}_{i,j}^{(k)}} w(p) \right) \end{aligned}$$

Il primo parametro della funzione *minimo* è:

$$\min_{p \in P_{i,j}^{(k-1)}} w(p) = d_{i,j}^{(k-1)}$$

Per quanto riguarda il secondo, dobbiamo trovare il peso del cammino più leggero da i a j che può passare per i nodi $1..k$ e passa sicuramente per k; consideriamo cammini semplici, in quanto non sono presenti cicli negativi: il cammino che cerchiamo può essere scomposto in due parti:

1. cammino da i a k che può passare per i nodi $1..k-1$;
2. cammino da k a j che può passare per i nodi $1..k-1$.

Chiaramente le due parti dovranno essere entrambe minime, affinché il cammino totale lo sia: il loro costo lo conosciamo ed è, rispettivamente, $d_{i,k}^{(k-1)}$ e $d_{k,j}^{(k-1)}$. Dunque, il secondo parametro della funzione *minimo* è:

$$\min_{p \in \widehat{P}_{1,j}^{(k)}} w(p) = d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}$$

La formula ricorsiva per il calcolo di $d_{i,j}^{(k)}$ è dunque la seguente:

$$d_{i,j}^{(k)} = \begin{cases} w(i,j) & \text{se } k = 0 \\ \min(d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}) & \text{se } k > 0 \end{cases}$$

L'algoritmo segue direttamente dalla formula:

algoritmo `floydWarshallCubico` (*matrice di adiacenza pesata* W) $\rightarrow$ *matrice delle distanze* $D^{(n)}$

```
1 n = rows(W)
2 for k = 1 to n do
3   for i = 1 to n do
4     for j = 1 to n do
5       d_{i,j}^(k) = min(d_{i,j}^(k-1), d_{i,k}^(k-1) + d_{k,j}^(k-1))
6 return D^(n)
```

Complessità

Temporale: $\Theta(n^3)$
Spaziale: $\Theta(n^3)$

Costruzione di un cammino minimo

Un metodo per costruire i cammini minimi usando l'algoritmo di Floyd Warshall consiste nell'usare la matrice dei pesi di cammino minimo prodotta e costruire, in tempo $O(n^3)$, la *matrice dei predecessori* Π. Un secondo metodo prevede di calcolare Π 'in linea' con l'algoritmo, o meglio, si calcola una sequenza di matrici $\Pi^{(0)}$, $\Pi^{(1)}$, ..., $\Pi^{(n)}$, dove $\Pi = \Pi^{(n)}$ e $\pi_{i,j}^{(k)}$ è definito come il predecessore di j su un cammino minimo dal vertice i avente tutti i vertici intermedi nell'insieme $\{1, ..., k\}$.
Si può definire induttivamente $\pi_{i,j}^{(k)}$; quando $k = 0$ si ha:

$$\pi_{i,j}^{(0)} = \begin{cases} \text{NIL} & \text{se } i = j \text{ o } w(i,j) = \infty \\ i & \text{se } i \neq j \text{ e } w(i,j) < \infty \end{cases}$$

Per $k > 1$, se prendiamo il cammino da i a j passante per k, allora il predecessore di j è lo stesso vertice che avevamo scelto come predecessore di j in un cammino da k con tutti i vertici intermedi nell'insieme $\{1, ..., k-1\}$; altrimenti, scegliamo lo stesso predecessore di j che avevamo scelto su un cammino minimo da i con tutti i vertici nell'insieme $\{1, ..., k-1\}$. Formalmente:

$$\pi_{i,j}^{(k)} = \begin{cases} \pi_{i,j}^{(k-1)} & \text{se } d_{i,j}^{(k-1)} \leq d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \\ \pi_{k,j}^{(k-1)} & \text{se } d_{i,j}^{(k-1)} > d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)} \end{cases}$$

Miglioramento

L'algoritmo che segue è una versione migliorata di Floyd Warshall ed ha complessità spaziale $\Theta(n^2)$, mentre la complessità temporale non cambia ($\Theta(n^3)$): l'idea consiste nell'utilizzare sempre la stessa matrice, invece di utilizzarne una diversa per ciascun k.

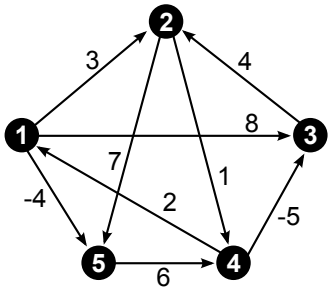
algoritmo `floydWarshallQuadratico` (*matrice di adiacenza pesata* W) $\rightarrow$ *matrice delle distanze* $D^{(n)}$

```
1 n = rows(W)
2 for k = 1 to n do
3   for i = 1 to n do
4     for j = 1 to n do
5       d_{i,j} = min(d_{i,j}, d_{i,k} + d_{k,j})
6 return D
```

Esempio di esecuzione

Simuliamo l'esecuzione sul seguente grafo:

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$



$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$