Cammino minimo per tutte le coppi di nodi

Il problema del cammino minimo per tutte le coppie di nodi è il seguente: dato un grafo pesato, per ogni coppia di nodi u e v, trovare un cammino minimo da u a v.

Posso risolvere questo problema invocando Bellman-Ford per ogni nodo del grafo. Questa soluzione ha complessità $\Theta(n^2 m)$, quindi $\Theta(n^4)$ se il grafo è denso.

Rappresentazione del grafo

Assumiamo di lavorare su un grafo pesato $G = (V, E, w)$ con pesi reali (anche negativi) privo di cicli di peso negativo. Assumiamo che $V = \{1, 2, \ldots, n\}$.

Rappresentiamo G con una matrice di adiacenza $W = (w_{i,j})$. W è una matrice quadrata $n \times n$ tale che:

- $w_{i,j}$ è il peso dell'arco (i, j) se $(i, j) \in E$ e $i \neq j$;
- $w_{i,j} = \infty$ se $(i, j) \notin E$ e $i \neq j$;
- $w_{i,j} = 0$ se $i = j$.

Rappresentazione dei cammini minimi

- Le **distanze** (i pesi dei cammini minimi) vengono memorizzati in una matrice $D = (d_{i,j})$ tale che $d_{i,j}$ contiene la distanza che separa i e j;
- i **cammini minimi** vengono memorizzati in una matrice $\Pi = (\pi_{i,j})$ tale che $\pi_{i,j}$ contiene il prececessore del nodo j sul cammino minimo con sorgente i, oppure NIL se $i = j$ oppure non esiste alcun cammino da i a j.

Nodi intermedi di un cammino

Dato un cammino semplice $p = \langle v_1, v_2, \ldots, v_r \rangle$, un **nodo intermedio** di p è un qualsiasi vertice v_i con $i \neq 1, r$.

Fissato $k \in V$, sia $V^{(k)} = \{1, 2, \ldots, k\}$. Consideriamo tutti i cammini minimo con nodi intermedi appartenenti a $V^{(k)}$.

Se $k = 0$, allora $V^{(k)} = \emptyset$, e quindi il cammino minimo da i a j è l'arco (i, j) se tale arco esiste.

Se invece $k = n$, allora $V^{(k)} = V$ e quindi un cammino minimo da i a j è un cammino minimo da i a j nel grafo.

Algoritmo di Floyd-Warshall

Per $1 \leq k \leq n$, sia $D^{(k)} = (d_{i,j}^{(k)})$ la matrice delle distanze e $\Pi^{(k)} = (\pi_{i,j}^{(k)})$ la matrice dei predecessori per cammini con **nodi intermedi** in $V^{(k)}$.

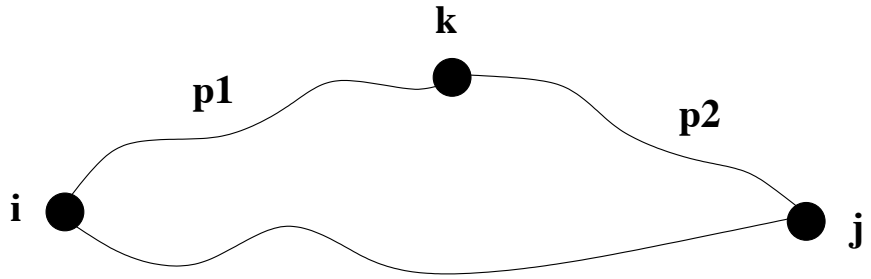
L'algoritmo di **Floyd-Warshall**:

1. inizializza opportunamente $D^{(0)}$ e $\Pi^{(0)}$;
2. calcola, per k che va da 1 a n, le matrici $D^{(k)}$ e $\Pi^{(k)}$. Ad ogni passo le matrici vengono calcolate in funzione delle matrici al passo precedente;
3. restituisce $D^{(n)}$ e $\Pi^{(n)}$.

Algoritmo di Floyd-Warshall

L'algoritmo di Floyd-Warshall si basa sulla seguente osservazione. Sia p il cammino minimo da i a j con nodi intermedi in $V^{(k)}$. Allora:

- se k è un nodo intermedio di p, allora sia p_1 il sottocammino da i a k e p_2 il sottocammino da k a j. Allora p_1 è un cammino minimo da i a k con nodi intermedi in $V^{(k-1)}$ e p_2 è un cammino minimo da k a j con nodi intermedi in $V^{(k-1)}$;
- se invece k non è un nodo intermedio di p, allora p è un cammino minimo da i a j con nodi intermedi in $V^{(k-1)}$.



Quindi il **cammino minimo** da i a j con nodi intermedi in $V^{(k)}$ è:

1. il cammino da i a j che non passa per k se

$$d_{i,j}^{(k-1)} \leq d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}$$

2. il cammino ottenuto concatenando i cammini p_1 e p_2, altrimenti.

In formule, abbiamo detto che:

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

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

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

```
1:  Floyd-Warshall(G)
2:  for i ← 1 to n do
3:      for j ← 1 to n do
4:          D^(0)[i, j] ← W[i, j]
5:          if i = j or W[i, j] = ∞ then
6:              Π^(0)[i, j] ← NIL
7:          else
8:              Π^(0)[i, j] ← i
9:          end if
10:     end for
11: end for
```

12: **for** $k \leftarrow 1$ **to** n **do**
13: **for** $i \leftarrow 1$ **to** n **do**
14: **for** $j \leftarrow 1$ **to** n **do**
15: **if** $D^{(k-1)}[i,j] \leq D^{(k-1)}[i,k] + D^{(k-1)}[k,j]$ **then**
16: $D^{(k)}[i,j] \leftarrow D^{(k-1)}[i,j]$
17: $\Pi^{(k)}[i,j] \leftarrow \Pi^{(k-1)}[i,j]$
18: **else**
19: $D^{(k)}[i,j] \leftarrow D^{(k-1)}[i,k] + D^{(k-1)}[k,j]$
20: $\Pi^{(k)}[i,j] \leftarrow \Pi^{(k-1)}[k,j]$
21: **end if**
22: **end for**
23: **end for**
24: **end for**

Complessità

- la fase di inizializzazione costa $\Theta(n^2)$;
- il corpo principale costa $\Theta(n^3)$.

Quindi Floyd-Washall costa

$$\Theta(n^2 + n^3) = \Theta(n^3)$$

Lo spazio utilizzato dall'algoritmo è $\Theta(n^3)$ ma può essere ridotto a $\Theta(n^2)$ riutilizzando le matrici (ogni matrice fa uso solo della matrice al passo precedente).

Ricostruzione dei cammini minimi

PrintPath(G, i, j)

```
1: if i = j then
2:    print i
3: else
4:    if Π[i, j] = NIL then
5:       print  "no path exists"
6:    else
7:       PrintPath(G, i, Π[i, j])
8:       print j
9:    end if
10: end if
```