GRAFI: ALGORITMO DI FLOYD WARSHALL

[Deme, seconda edizione] cap. 14

Sezione 14.6

Calcolare i cammini minimi tra tutte le coppie di vertici

Fino ad ora, ci siamo confrontati solo con il problema di trovare i cammini minimi tra **un vertice di partenza** s **e tutti gli altri vertici** del grafo.

In alcune applicazioni (ad esempio la **propagazione di vincoli** numerici), è necessario trovare i cammini minimi tra **tutte le coppie di vertici** del grafo.

In queste slide vedremo un algoritmo, basato sulla **programmazione dinamica**, che calcola i cammini minimi tra tutte le coppie di vertici del grafo in tempo polinomiale.

Nota: possiamo anche applicare Bellman-Ford (o Dijkstra, o BF-DAG) su tutti gli n nodi usando ciascun nodo come sorgente. Ad esempio, la complessità di Bellman-Ford su tutti i nodi è $O(mn^2)$ che con grafi densi è circa $O(n^4)$

Distanze e cammini minimi k-vincolati

Denotiamo i vertici del grafo come $v_1, v_2, \dots v_n$.

Per un k fissato con $1 \le k \le n$ definiamo

Cammino minimo k-vincolato tra x e y (denotato con π_{xy}^k) il cammino che va **da x a y** di **costo minimo** tra tutti quelli che **non contengono i vertici** $\{v_{k+1}, \dots v_n\}$ (esclusi x e y estremi del cammino).

Distanza k-vincolata (denotata con d_{xy}^k) è **il peso** $W(\pi_{xy}^k)$ se π_{xy}^k esiste, ∞ **altrimenti**.

È facile notare che

$\pi_{xy}^0 = (x,y)$ e $d_{xy}^0 = W(x,y)$ **se** $(x,y) \in E$,

$\pi_{xy}^0 = \{\}$ e $d_{xy}^0 = 0$ **se** $x = y$,

π_{xy}^0 non esiste e $d_{xy}^0 = \infty$ **altrimenti**.

Infine, $d_{xy}^n = \delta(x,y)$

Nota: qui, per brevità, per le distanze si è usata la notazione del [Deme]. Dovessimo usare la solita notazione indicheremmo d_{xy}^k con $\delta^k(x,y)$.

Grafo k-vincolato

È immediato notare che un cammino minimo k-vincolato è un cammino minimo in un grafo (k-vincolato) $G_k = (V_k, E_k)$ in cui

$V_k = (V - \{v_{k+1}, \dots v_n\}) \cup \{x, y\}$ e
$E_k = E \cap (V_k \times V_k)$

Essendo G_k un grafo, anche per esso esisteranno dei cammini minimi (che sono i **cammini minimi k-vincolati**) e varrà la proprietà della **sottostruttura ottima** (ogni sottocammino di un cammino minimo k-vincolato è esso stesso un cammino minimo k-vincolato).

Quindi potremo applicare le tecniche di **programmazione dinamica**.

Relazione tra distanze k-vincolate

Per ogni 1 ≤ k ≤ n e per ogni coppia x, y definiamo **l'equazione ricorsiva**

$$d_{xy}^{k} = \min(d_{xy}^{k-1}, d_{xv_k}^{k-1} + d_{v_k y}^{k-1})$$

DIMOSTRAZIONE: abbiamo 2 casi:

CASO 1: $v_k \notin \pi_{xy}^{k}$. Allora π_{xy}^{k} **è anche un cammino minimo (k-1)-vincolato**. Se così non fosse esisterebbe un cammino minimo π'^{k-1}_{xy} (k-1)-vincolato di peso minore di π_{xy}^{k}. Ma allora π'^{k-1}_{xy} sarebbe anche un cammino k-vincolato e avrebbe peso minore di π_{xy}^{k} (**assurdo**).

CASO 2: $v_k \in \pi_{xy}^{k}$. I sottocammini da **x a v_k** e da **v_k a y** sono cammini minimi k-vincolati (per la sottostruttura ottima). Poiché non contengono internamente v_k, essi sono anche i **cammini minimi (k-1)-vincolati**. Quindi possiamo dire che $d_{xy}^{k} = d_{xv_k}^{k-1} + d_{v_k y}^{k-1}$

Più semplicemente

La formula ricorsiva

$$d_{xy}^{k} = \min(d_{xy}^{k-1}, d_{xv_k}^{k-1} + d_{v_k y}^{k-1})$$

Può sembrare complicata, ma in realtà ciò che fa è semplicissimo.

Preso un k, si va a vedere se **la concatenazione dei cammini minimi da x a v_k e da v_k a y ha peso minore del cammino minimo da x a y** (senza considerare v_k, v_{k+1}, v_n). Quindi si va a verificare la (non verifica della) disuguaglianza triangolare

D(x,y) > D(x,k) + D(k,y)

Se la disuguaglianza triangolare non è verificata, si applica un **rilassamento** facendo passare il cammino minimo attraverso v_k.

D(x,y) <- D(x,k) + D(k,y)

Struttura di memoizzazione

Dobbiamo modellare, per ogni coppia di vertici x e y (ordinata in caso di grafi orientati) la lunghezza del cammino provvisorio da x a y.

Lo dobbiamo fare per ogni k-esima iterazione dell'algoritmo, con $k \leq n$

Quindi ci servono n matrici **D^k di dimensione nxn** (quindi lo **spazio richiesto** è **$O(n^3)$**)

D^0	v_1	...	v_n
v_1	$d^0_{v_1v_1}$	...	$d^0_{v_1v_n}$
...	...	...	...
v_n	$d^0_{v_nv_1}$	...	$d^0_{v_nv_n}$

D^1	v_1	...	v_n
v_1	$d^1_{v_1v_1}$	...	$d^1_{v_1v_n}$
...	...	...	...
v_n	$d^1_{v_nv_1}$	...	$d^1_{v_nv_n}$

...

D^{k-1}[i,j] contiene la **distanza** tra il vertice v_i e v_j in un grafo $\boldsymbol{G^{k-1}}$ in cui $V^{k-1} = G - \{v_k, v_{k+1}, \dots v_n\} \cup \{v_i, v_j\}$

D^{k-1}	v_1	v_j	v_n
v_1	…	…	…
v_i	…	…	…
v_n	…	…	…

D^{k}[i,j] contiene la **distanza** tra il vertice v_i e v_j in un grafo $\boldsymbol{G^{k}}$ in cui $V^{k} = G - \{v_{k+1}, \dots v_n\} \cup \{v_i, v_j\}$.

Avendo aggiunto solo v_k rispetto al grafo precedente, le uniche strade diverse passano da v_k.
Qual è il miglior cammino da v_i a v_j che passa da v_k in G^k? È a concatenazione del miglior cammino da v_i a v_k ed il migliore da v_k a v_j. Ma questi 2 già ce li ho nella matrice D^{k-1}. Infatti sono D^{k-1} [i,k] e D^{k-1} [k,j].

Allora, in G^k il miglior cammino da v_i a v_j è quello con distanza minore tra D^{k-1} [i,j] (quella del vecchio cammino finora trovato) e D^{k-1} [i,k] + D^{k-1} [k,j] (il cammino nuovo in G^k, che prima non c'era).

Floyd-Warshall – versione «facile»

***Floyd-Warshall* (G, W)**

```
   for i = 1..n
     for j = 1..n
        if i = j     D^0[i,j] <- 0
        else if (i,j) ∈ E     D^0[i,j] <- W(i,j)
        else D^0[i,j] <- ∞
  for k = 1..n
     for i = 1..n
        for j = 1..n
           D^k[i,j] <- D^k-1[i,j]
           if D^k[i,j] > D^k-1[i,k] + D^k-1[k,j] then D^k[i,j] <- D^k-1[i,k] + D^k-1[k,j]
end
```

Floyd-Warshall – ottimizzazione

Lo **spazio** richiesto dalla prima versione dell'algoritmo è **cubico**, vogliamo capire se è possibile fare di meglio.

Vorremmo usare **una sola matrice aggiornata ad ogni iterazione**, come facevamo per il vettore d negli algoritmi uno-a-molti.

Notiamo che l'ostacolo principale all'uso di una sola matrice è l'assegnazione $D^k[i,j] <- D^{k-1}[i,k] + D^{k-1}[k,j]$ che richiede l'uso di **$D^{k-1}[i,k]$ e $D^{k-1}[k,j]$ durante la k-esima iterazione**. Ma possiamo notare che $D^{k-1}[k,k] = 0$ (il cammino minimo da un vertice a se stesso è vuoto) e quindi

$$\mathbf{D^k[i,k]} = \min(D^{k-1}[i,k], D^{k-1}[i,k] + \mathbf{D^{k-1}[k,k]}) = \min(D^{k-1}[i,k], D^{k-1}[i,k] + 0) \mathbf{= D^{k-1}[i,k]}$$

E analogamente

$$\mathbf{D^k[k,j]} = \min(D^{k-1}[k,j], \mathbf{D^{k-1}[k,k]} + D^{k-1}[k,j]) = \min(D^{k-1}[k,j], 0 + D^{k-1}[k,j]) \mathbf{= D^{k-1}[k,j]}$$

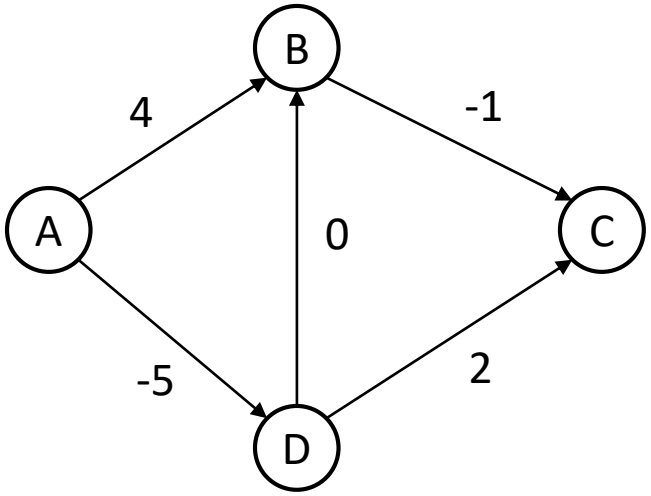
Di conseguenza, $D^{k-1}[i,k]$ e $D^{k-1}[k,j]$ non cambieranno valore dall'iterazione k-1 a quella k, quindi possiamo usare un'unica matrice D.

Algoritmo di Floyd-Warshall

```
Floyd-Warshall (G, W)
    for i = 1..n
      for j = 1..n
         if i = j D[i,j] <- 0
         else if (i,j) ∈ E D[i,j] <- W(i,j)
         else D[i,j] <- ∞
   for k = 1..n
      for i = 1..n
         for j = 1..n
            if D[i,j] > D[i,k] + D[k,j] then D[i,j] <- D[i,k] + D[k,j]
end
```

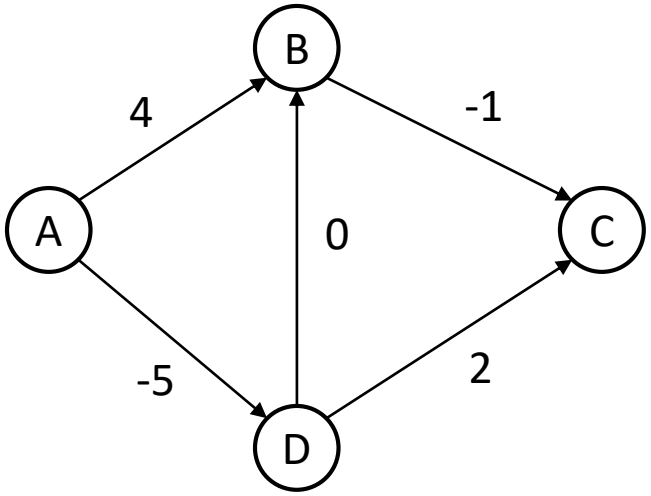
Questa versione ha **complessità spaziale $O(n^2)$** e **temporale $O(n^3)$**.

Esempio



	A	B	C	D
A	0	4	∞	-5
B	∞	0	-1	∞
C	∞	∞	0	∞
D	∞	0	2	0

Esempio



	A	B	C	D
A	0	4	∞	-5
B	∞	0	-1	∞
C	∞	∞	0	∞
D	∞	0	2	0

i	j	K
A	A	A

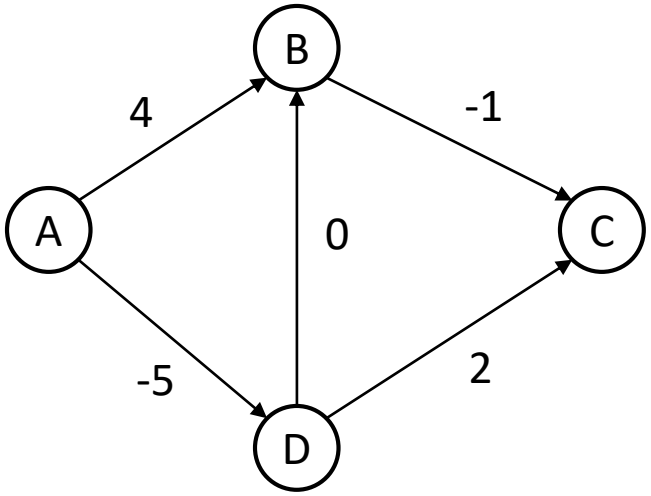
D[A,A] > D[A,A] + D[A,A]?

i	j	K
A	B	A

D[A,B] > D[A,A] + D[A,B]?

...

Esempio



	A	B	C	D
A	0	4	∞	-5
B	∞	0	-1	∞
C	∞	∞	0	∞
D	∞	0	2	0

i	j	K
A	C	B

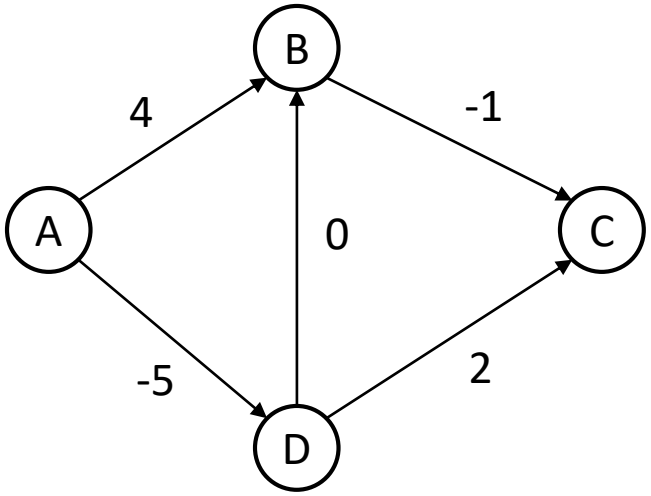
D[A,C] > D[A,B] + D[B,C]?

∞ > 4 + (-1) = 3

	A	B	C	D
A	0	4	3	-5
B	∞	0	-1	∞
C	∞	∞	0	∞
D	∞	0	2	0

...

Esempio



	A	B	C	D
A	0	4	3	-5
B	∞	0	-1	∞
C	∞	∞	0	∞
D	∞	0	2	0

i	j	K
D	C	B

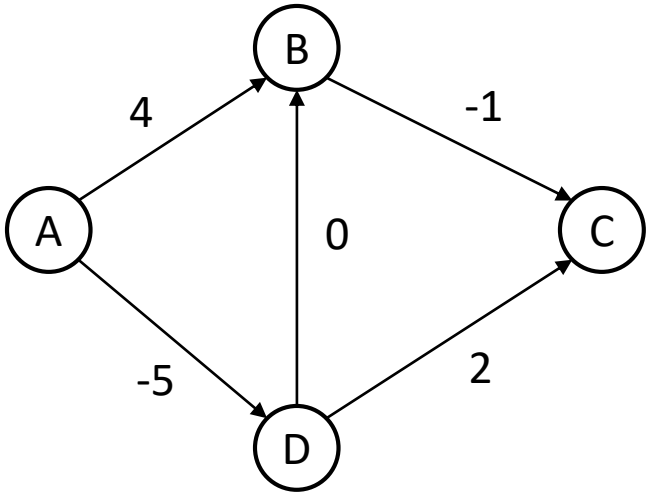
D[D,C] > D[D,B] + D[B,C]?

2 > 0 + (-1) = -1

	A	B	C	D
A	0	4	3	-5
B	∞	0	-1	∞
C	∞	∞	0	∞
D	∞	0	-1	0

...

Esempio



	A	B	C	D
A	0	4	3	-5
B	∞	0	-1	∞
C	∞	∞	0	∞
D	∞	0	-1	0

i	j	K
A	B	D

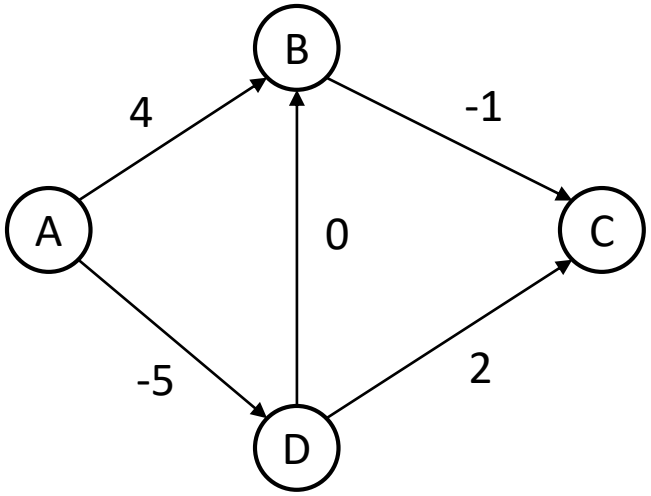
D[A,B] > D[A,D] + D[D,B]?

4 > -5 + 0 = -5

	A	B	C	D
A	0	-5	3	-5
B	∞	0	-1	∞
C	∞	∞	0	∞
D	∞	0	-1	0

...

Esempio



	A	B	C	D
A	0	-5	3	-5
B	∞	0	-1	∞
C	∞	∞	0	∞
D	∞	0	-1	0

i	j	K
A	C	D

D[A,C] > D[A,D] + D[D,C]?

3 > -5 + (-1) = -6

	A	B	C	D
A	0	-5	-6	-5
B	∞	0	-1	∞
C	∞	∞	0	∞
D	∞	0	-1	0

La tabella non cambia da qui alla fine dell'algoritmo

…

Floyd-Warshall – cicli negativi

Anche in questo caso abbiamo bisogno di rilevare dei cicli di peso negativo.

In Floyd-Warshall, rilevare questo genere di cicli è relativamente semplice. Infatti **in presenza di un ciclo è possibile raggiungere un vertice v da se stesso con un cammino di distanza $d_{vv} < 0$**.

Ma noi abbiamo d_{vv} per ogni vertice v sulla **diagonale** della nostra matrice D, quindi ci basta aggiungere un controllo

if i = j **and** D[i,j] < 0 **then return errore**

Floyd-Warshall – cicli negativi

```
FW-NEG (G, W)
   for i = 1..n
     for j = 1..n
        if i = j D[i,j] <- 0
        else if (i,j) ∈ E D[i,j] <- W(i,j)
        else D[i,j] <- ∞
  for k = 1..n
     for i = 1..n
        for j = 1..n
           if D[i,j] > D[i,k] + D[k,j] then D[i,j] <- D[i,k] + D[k,j]
           if i = j and D[i,j] < 0 then return errore
end
```

Floyd-Warshall – predecessori

```
FW-PRED (G, W)
   for i = 1..n
      for j = 1..n
         P[i,j] <- -1
         if i = j D[i,j] <- 0
         else if (i,j) ∈ E D[i,j] <- W(i,j) P[i,j] <- i
         else D[i,j] <- ∞
   for k = 1..n
      for i = 1..n
         for j = 1..n
            if D[i,j] > D[i,k] + D[k,j] then
               D[i,j] <- D[i,k] + D[k,j]
               P[i,j] <- P[k,j]
            if i = j and D[i,j] < 0 then return errore
end
```

P[i,j] rappresenta il predecessore di j nel cammino minimo tra i e j

Se il cammino minimo tra i e j passa per il nodo k allora il predecessore di j in i⇝j sarà chiaramente il predecessore di j in k ⇝ j.

Floyd-Warshall – ricostruzione c.m.

Path-reconstruction **(P, x, y)**
  if x = y **return** x
  else return *Path-reconstruction (*P, x, P[x,y]*)* + y

Simulazioni di Floyd Warshall

https://www-m9.ma.tum.de/graph-algorithms/spp-floyd-warshall/index_en.html
(permette di creare grafi personalizzati, mostra solo passaggi con modifiche)

https://www.cs.usfca.edu/~galles/visualization/Floyd.html
(grafi casuali, mostra tutti i passaggi)

Cosa devo aver capito fino ad ora

- Il problema di trovare i cammini minimi tra tutte le coppie di vertici in un grafo
- Cammini minimi e distanze k-vincolati
- Utilizzo delle distanze k-vincolate come sottoproblemi dei cammini minimi (correttezza)
- Strutture di memoizzazione per il problema
- Algoritmo di Floyd-Warshall
  - Complessità temporale e spaziale (2 casi)

…se non ho capito qualcosa

- Alzo la mano e chiedo
- Ripasso sul libro
- Chiedo aiuto sul forum
- Chiedo o mando una mail al docente