Cammini minimi tra tutte le coppie di vertici

Come abbiamo già discusso possiamo trovare i cammini minimi da una sorgente a tutti i vertici di un grafo $G=(V,E)$ $|V|=n$ $|E|=m$

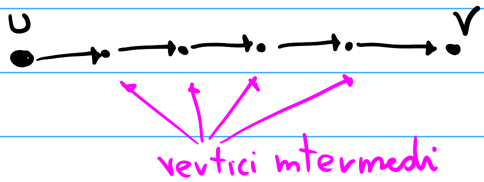
$m \leq n^2$

- pesi ≥ 0 Dijkstra in $O(m \log n)$
- pesi arbitrari Bellman-Ford in $O(n \cdot m)$

Dunque possiamo sempre ripetere l'algoritmo per ogni sorgente a costo di pagare n volte la complessità originaria

Vediamo invece un algoritmo alternativo che calcola i cammini minimi tra tutte le coppie, di complessità $O(n^3)$

Vertici intermedi di un cammino



Consideriamo un cammino da u a v.

I vertici intermedi di un cammino sono quelli diversi da u e v, incontrati nel cammino

Classifichiamo un cammino in base al vertice MASSIMO consentito tra i vertici intermedi nel cammino ($V = \{0, 2, 3, \ldots, n-1\}$)

E.s. vertice intermedio massimo 5

Sì: $7 \to 2 \to 3 \to 0 \to 4$

Sì: $0 \to 6$

Sì: $8 \to 2 \to 5 \to 3 \to 10$

NO: $2 \to 4 \to 7 \to 3 \to 8$ (7: NON VALIDO)

Nessun Vertice Intermedio tra u e v

$u = v$ $\qquad$ $u \to v$

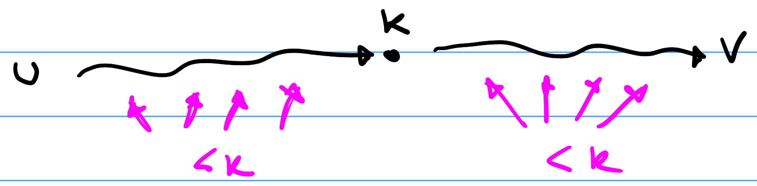
Cammini senza limitazioni $\equiv$ Vertici intermedi $< n$

① Caratterizzazione della soluzione ottima

Se p è un cammino minimo $u \rightsquigarrow v$ con vertici intermedi $\leq k$ abbiamo due casi:

- p non contiene il vertice intermedio k $\qquad p = u \rightsquigarrow v$ ($< k$)

- p contiene il vertice intermedio k



② Calcolo della soluzione ottima con schema ricorsivo

[ASSUMIAMO CHE NON CI SIANO CICLI NEGATIVI]

- Sia $\delta^k(u,v)$ = costo del cammino minimo tra u e v con vertici intermedi strettamente $< k$

allora $\delta(u,v) = \delta^n(u,v)$

- Possiamo calcolare $\delta^k(u,v)$ ricorsivamente

$$\delta^0(u,v) = \begin{cases} 0 & \text{se } u = v \\ W[u,v] & \text{se } (u,v) \in E(G) \\ +\infty & \text{altrimenti} \end{cases}$$

NESSUN VERTICE INTERMEDIO

$k > 0$

$$\delta^{k+1}(u,v) = \min\left\{\delta^k(u,v),\ \delta^k(u,k) + \delta^k(k,v)\right\}$$

- Possiamo quindi calcolare una sequenza $D^{(0)}, D^{(1)}, D^{(2)}, \dots, D^{(n)}$ di matrici dove

$D^{(k)}$ è una matrice $n \times n$ e $D^{(k)}[u,v] = \delta^k(u,v)$

alla fine $D^{(n)}[u,v] = \delta(u,v)$

OSS Per ora ci concentriamo sul calcolo del COSTO delle soluzioni, e non sui cammini effettivi

3 Calcolo bottom-up dei costi

È abbastanza semplice osservare che per calcolare $D^{(k)}$ è necessario e sufficiente aver calcolato $D^{(k-1)}$.

Vediamo l'algoritmo di Floyd-Warshall, che di fatto non è altro che il calcolo della sequenza delle $D^{(k)}$.

- Floyd-Warshall (G, W)

  input grafo orientato pesato

  output Matrice $n \times n$ che alla cella $[u,v]$ contiene $\delta(u,v)$

$$D^{(0)} = \begin{cases} 0 & \text{per } u = v \\ +\infty & \text{se } (u,v) \notin E(G) \\ W[u,v] & \text{se } (u,v) \in E(G) \end{cases}$$

$\rightarrow$ complessità $O(n^2)$

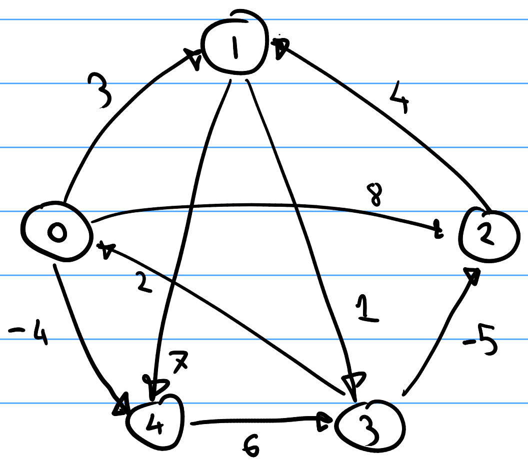
```
for k = 0 a n-1
    for u in {0,...,n-1}
        for v in {0,...,n-1}
            D^(k+1)[u,v] = min(D^(k)[u,v], D^(k)[u,k] + D^(k)[k,v])
```

④ Dal costo delle soluzioni ottime, ai cammini minimi

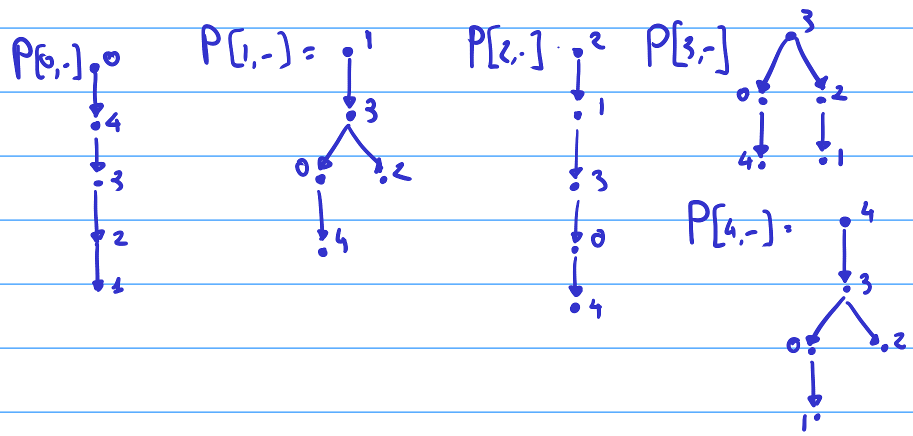
Nel caso di algoritmi a singola sorgente s

$$P = [\, \cdot \; \cdot \; \cdot \; \cdot \; \cdot \; \cdot \,] \quad \text{array dei predecessori}$$

In questo caso P è una matrice $n \times n$ per cui $P[s,\cdot]$ è la riga della matrice che rappresenta i cammini minimi da s



$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \left[\begin{array}{ccccc} - & 2 & 3 & 4 & 0 \\ 3 & - & 3 & 1 & 0 \\ 3 & 2 & - & 1 & 0 \\ 3 & 2 & 3 & - & 0 \\ 3 & 2 & 3 & 4 & - \end{array}\right] \end{array}$$



Come calcolare P ?

Possiamo calcolare $P^{(0)}, P^{(1)}, \dots P^{(n)}$

dove $P^k[u,v] = \begin{cases} \text{"il predecessore di } v \text{ in un cammino } u \rightsquigarrow v \text{ con vertici intermedi} < k \text{ e costo } \delta^k(u,v)\text{"} \\ \text{NIL se non esiste un cammino del genere} \end{cases}$

$D^{(k+1)}[u,v] \to D^k[u,v]$ allora $P^{k+1}[u,v] = P^k[u,v]$

$D^{(k+1)}[u,v] \to D^k[u,k] + D^k[k,v]$ allora $P^{k+1}[u,v] = P^k[k,v]$ $\quad u \rightsquigarrow k \rightsquigarrow v$

Vedere ESEMPIO PRECALCOLATO

Complessità: $O(n^3)$
chiaramente sono 3 cicli annidati di dim n.

Uso della memoria: Vengono calcolate $2n$ matrici $n \times n$, quindi l'analisi sembra indicare il bisogno di $O(n^3)$ celle di memoria

ESERCIZIO: Come ridurre la memoria utilizzata a $O(n^2)$?

— o — o — o — o —

- Le 4 Fasi della programmazione dinamica
un concetto su cui torneremo e di cui F·W è un ottimo esempio

1. Caratterizzazione della soluzione ottima
2. Calcolo della soluzione con schema ricorsivo
3. Calcolo bottom-up dei costi
4. Costruzione della soluzione mentre si calcolano i costi

— o — o — o —

Correttezza dell'algoritmo e cicli negativi

- Ricordiamo che $\delta(u,v) = \begin{cases} +\infty & \text{se } u \text{ non raggiunge } v \\ -\infty & \text{se } \exists k \text{ in un ciclo negativo con } u \rightsquigarrow k \rightsquigarrow v \\ \text{costo del cammino minimo tra } u \text{ e } v & \text{(ben definito) altrimenti} \end{cases}$

OSS $\delta^k(u,v)$ è sempre $+\infty$ o un numero reale, per definizione
(è un minimo tra cammini senza vertici ripetuti)

OSS Se $\delta(u,v) > -\infty$ allora $\delta(u,v) = \delta^n(u,v)$

Qual è la relazione tra D^k calcolata dall'algoritmo e $\delta^k(u,v)$

Lemma 1 $D^k[u,v] \le \delta^k(u,v)$

dim per induzione su k.

- $k=0$ D^0 definito come $\delta^0(u,v)$
- $D^{k+1}[u,v] = \min$
  - $D^k[u,v] \le \delta^k(u,v)$
  - $D^k[u,k] + D^k[k,v] \le \delta^k(u,k) + \delta^k(k,v)$

quindi $D^{k+1}[u,v] \le \min\{\delta^k(u,v), \delta^k(u,k) + \delta^k(k,v)\} = \delta^{k+1}(u,v)$

Lemma 2 [Es 25.3-3] Se $\delta(u,v) > -\infty$ allora $D^k[u,v] = \delta^k(u,v)$
e di conseguenza $D^n[u,v] = \delta(u,v)$

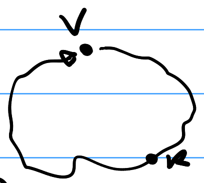
dim Per esercizio: dimostrate che quando $\delta(u,v) > +\infty$
allora ogni volta che $D^k[u,v]$ viene settata ad un valore finito,
$P^k[u,-]$ di fatto contiene un cammino $u \rightsquigarrow v$ di costo $D^k[u,v]$
con vertici intermedi $< k$.

Come riconosciamo i cicli negativi?

Lemma 3 In ogni ciclo negativo esiste un vertice v contenuto nel ciclo
per cui $D^n[v,v] < 0$

oss Questa cosa ovviamente non è possibile per vertici con $\delta(v,v) > -\infty$ [Lemma 2]

dim Consideriamo un ciclo negativo



che coinvolge v, e che k sia il vertice di indice più alto del ciclo

- $\text{costo}(v,k) + \text{costo}(k,v) < 0$ per ipotesi
- $\text{costo}(v,k) \ge \delta^k(v,k) \ge D^k[v,k]$ e $\text{costo}(k,v) \ge \delta^k(k,v) \ge D^k[k,v]$

Al k-esimo ciclo $D^{k+1}[v,v] \le D^k[v,k] + D^k[k,v] < 0$ $\square$

Esercizio (25.3-7)

- Estendete l'algoritmo di F-W in modo tale che calcoli una matrice D per cui

$$D[u,v] = \begin{cases} +\infty & \text{se } v \text{ è irraggiungibile da } u \\ -\infty & \text{se } \exists k \text{ in un ciclo negativo e } u \rightsquigarrow k \rightsquigarrow v \\ \text{costo del cammino minimo tra } u \text{ e } v \end{cases}$$

- E che calcoli una matrice P per cui, se $D[u,v] \neq \pm\infty$ allora
  $P[u,v] =$ il predecessore di v in un cammino da u a v di costo $D[u,v]$

L'algoritmo deve avere complessità $O(n^3)$ e usare spazio $O(n^2)$

Esercizio 25.3-4

Abbiamo già ridotto lo spazio a $O(n^2)$ in un esercizio precedente ma possiamo fare meglio. Usiamo solo una matrice D (e di conseguenza una sola matrice P)

$$D = \begin{cases} 0 & \text{per } u = v \\ +\infty & \text{se } (u,v) \notin E(G) \\ W[u,v] & \text{se } (u,v) \in E(G) \end{cases}$$

```
for k = 0 a n-1
    for u in {0,...,n-1}
        for v in {0,...,n-1}
            D[u,v] = min(D[u,v], D[u,k] + D[k,v])
```

eliminiamo gli indici!!

- Dimostrare che la correttezza dell'algoritmo non è pregiudicata
- E neppure le aggiunte successive: P, la gestione di cicli negativi ecc....