Programmazione dinamica (IV parte)

Progettazione di Algoritmi a.a. 2023-24

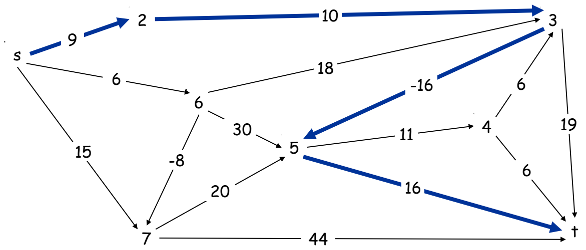
Matricole congrue a 1

Docente: Annalisa De Bonis

Cammini minimi

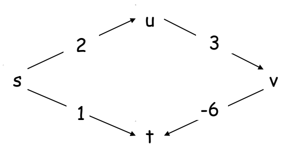
- Problema del percorso piu` corto. Dato un grafo direzionato G = (V, E), con pesi degli archi $c_{vw}$, trovare il percorso piu` corto da s a t.

- Esempio. I nodi rappresentano agenti finanziari e c_{vw} e` il costo (eventualmente <0) di una transazione che consiste nel comprare dall'agente v e vendere immediatamente a w.

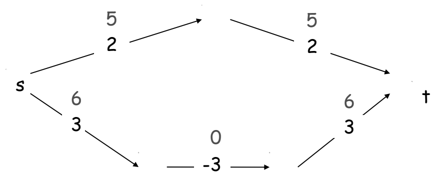


Cammini minimi in presenza di archi con costo negativo

- Dijkstra. Puo` fallire se ci sono archi di costo negativo

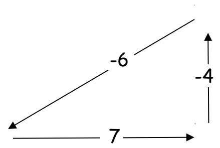


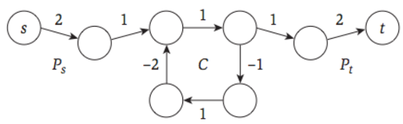
- Re-weighting. Aggiungere una costante positiva ai pesi degli archi potrebbe non funzionare.



Cammini minimi in presenza di archi con costo negativo

- Ciclo di costo negativo.



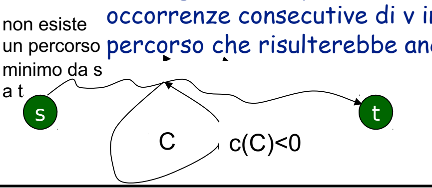


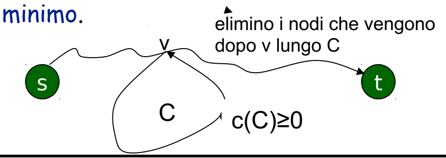
Cammini minimi in presenza di archi con costo negativo

Osservazione. Se qualche percorso da s a t contiene un ciclo di costo negativo allora non esiste un percorso minimo da s a t. In caso contrario esiste un percorso minimo da s a t che e` semplice (nessun nodo compare due volte sul percorso).

- Dim. Se esiste un percorso P da s a t con un ciclo C di costo negativo –c allora gni volta che attraversiamo il ciclo riduciamo il costo del percorso di un valore pari a c. Cio` rende impossibile definire il costo del percorso minimo perche` dato un percorso riusciamo sempre a trovarne uno di costo minore attraversando il ciclo C (osservazione questa che avevamo gia` fatto in precedenti lezioni).

Supponiamo ora che nessun percorso da s a t contenga cicli negativi e sia P un percorso minimo da s a t (ovviamente P e` privo di cicli di costo negativo). Supponiamo che un certo vertice v appaia almeno due volte in P. C'e` quindi in P un ciclo che contiene v e che per ipotesi deve avere costo **non negativo**. In questo caso potremmo rimuovere le porzioni di P tra due occorrenze consecutive di v in P senza far aumentare il costo del percorso che risulterebbe ancora minimo.





Cammini minimi: Programmazione dinamica

Sia t la destinazione a cui si vuole arrivare.

Def. OPT(i, v) = lunghezza del cammino piu` corto P per andare da v a t che consiste di al piu` i archi

Per computare OPT(i,v) quando i>0, esiste un arco uscente da v e v≠t, osserviamo che

- il percorso ottimo P deve contenere almeno un arco (che ha come origine v).
- se (v, w) e` il primo arco di P allora P e` formato da (v, w) e d percorso piu` corto da w a t di al piu` i-1 archi.
- siccome non sappiamo quale sia il primo arco di P allora computiamo $OPT(i,w) = \min_{(v,w)\in E}\{OPT(i-1,w) + c_{vw}\}$

La formula di ricorrenza è quindi:

$$OPT(i,v) = \begin{cases} 0 & \text{se } v = t \\ \infty & \text{se } v \neq t \text{ e } i = 0 \text{ o se } v \neq t \text{ e } v \text{ non ha archi uscenti} \\ \min_{(v,w)\in E}\{(c(v,w) + OPT(i-1,w)\} & \text{altrimenti} \end{cases}$$

Cammini minimi: Programmazione dinamica

$$OPT(i,v) = \begin{cases} 0 & \text{se } v = t \\ \infty & \text{se } v \neq t \text{ e } i = 0 \text{ o se } v \neq t \text{ e } v \text{ non ha archi uscenti} \\ \min_{(v,w)\in E}\{(c(v,w) + OPT(i-1,w)\} & \text{altrimenti} \end{cases}$$

Notiamo che nel secondo caso usiamo ∞ per indicare che se possiamo attraversare 0 archi o se non ci sono archi uscenti da v, non è possibile raggiungere t.

Dove si usa l'osservazione di prima sul fatto che in assenza di cicli negativi il percorso minimo e` semplice?

Ecco dove…
Affermazione. Se non ci sono cicli di costo negativo allora OPT(n-1,v) = lunghezza del percorso piu` corto da v a t.

Dim. Dall'osservazione precedente se non ci sono cicli negativi allora esiste un percorso di costo minimo da v a t che e` semplice e di conseguenza contiene al piu` n-1 archi

$$OPT(i,v) = \begin{cases} 0 & \text{se } v = t \\ \infty & \text{se } v \neq t \text{ e } i = 0 \text{ o se } v \neq t \text{ e } v \text{ non ha archi uscenti} \\ \min_{(v,w)\in E}\{(c(v,w) + OPT(i-1,w)\} & \text{altrimenti} \end{cases}$$

Algoritmo di Bellman-Ford per i cammini minimi

```
Shortest-Path(G, t) {
   foreach node v ∈ V
      M[0, v] ← ∞
      S[0, v] ← Ø // Ø  indica che non ci sono percorsi
                 //da v a t di al piu` 0 archi
   for i = 0 to n-1
      M[i, t] ← 0
      S[i, t] ← t  //t indica che non ci sono successori di t
                   //lungo il percorso ottimo da t a t
   for i = 1 to n-1
      foreach node v ∈ V
         M[i, v] ← ∞, S[i, v] ← Ø
         foreach edge (v, w) ∈ E
            if M[i-1, w] + cvw < M[i, v]
                M[i, v] ← M[i-1, w] + cvw
                S[i,v] ← w  //serve per ricostruire i
                            //percorsi minimi verso t
}
```

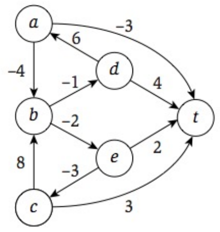
- Assumiamo che per ogni v esista un percorso da v a t → n=O(m)
- Analisi. Tempo $\Theta(mn)$, spazio $\Theta(n^2)$.
- S[i,v]: Memorizza il successore di v lungo il percoso minimo per andare da v a t attraversando al piu` i archi

Algoritmo di Bellman-Ford per i cammini minimi

Osservazione

- L'algoritmo di Bellman-Ford di fatto calcola le lunghezze dei cammini minimi da v a t per ogni v (risolve Single Destination Shortest Paths)
- Queste lunghezze sono contenute nella riga n-1
- L'algoritmo puo` essere scritto in modo che prenda in input un vertice sorgente s ed un vertice destinazione t ma il contenuto della tabella M dipende solo da t.
- In altri termini, una volta costruita la tabella per un certo t, possiamo ottenere la lunghezza del percorso piu` corto da un qualsiasi nodo v al nodo t andando a leggere l'entrata M[n-1,v]

Bellman-Ford: esempio



Come aggiorno la cella M[i,j] e la cella S[i,j], per i>1?
Inizialmente pongo M[i,j]=∞ e S[i,j]=∅.
Nella riga i-1 esamino tutte le entrate M[i-1,k] per cui esiste l'arco (j,k) e per ciascuna di queste entrate computo M[i-1,k]+c_{jk}. Se questo valore è piu` piccolo di M[i,j], pongo M[i,j]= M[i-1,k]+c_{jk} e S[i,j]=k.

M

	t	a	b	c	d	e
0	0	∞	∞	∞	∞	∞
1	0	-3	∞	3	4	2
2	0	-3	0	3	3	0
3	0	-4	-2	3	3	0
4	0	-6	-2	3	2	0
5	0	-6	-2	3	0	0

S

	t	a	b	c	d	e
0	t	∅	∅	∅	∅	∅
1	t	t	∅	t	t	t
2	t	t	e	t	a	c
3	t	b	e	t	a	c
4	t	b	e	t	a	c
5	t	b	e	t	a	c

Algoritmo che produce il cammino minimo

```
FindPath(i,v):
  if S[i,v]= Ø
     output "No path"
      return
  if v= t
     output t
     return
  output v
  FindPath(i-1,S[i,v])
```

prima volta invocato con i=n-1 e v uguale al nodo per il quale vogliamo computare il cammino minimo fino a t

tempo O(n) perche'
- se ignoriamo il tempo per la chiamata ricorsiva al suo interno, il tempo di ciascuna chiamata e` O(1)
- vengono effettuate al piu` n-1 chiamate

69

Bellman-Ford: esempio

S

	t	a	b	c	d	e
0	t	Ø	Ø	Ø	Ø	Ø
1	t	t	Ø	t	t	t
2	t	t	e	t	a	c
3	t	b	e	t	a	c
4	t	b	e	t	a	c
5	t	b	e	t	a	c

Supponiamo di voler conoscere il percorso minimo tra a e t
Invoco FindPath(5,a)

output a e effettua ricorsione con i=4 e v=S[5,a]=b

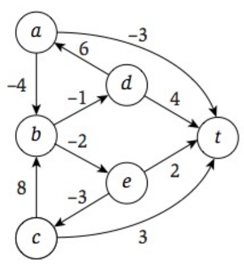
output b e effettua ricorsione con i=3 e v= S[4,b]=e

output e e effettua ricorsione con i=2 e v= S[2,e]=c

output c e effettua ricorsione con i=1 e v= S[1,c]=t

output t ed esci

Il percorso minimo da a verso t e` a,b,e,c,t




```
FindPath(i,v):
  if S[i,v]= Ø
     output "No path"
      return
  if v= t
     output t
     return
  output v
  FindPath(i-1,S[i,v])
```

70

70