Grafi: cammini minimi

Fulvio CORNO - Matteo SONZA REORDA
Dip. Automatica e Informatica
Politecnico di Torino

Sommario

- Introduzione
- Algoritmo di Dijkstra
- Algoritmo di Bellman-Ford.

Sommario

- Introduzione
- Algoritmo di Dijkstra
- Algoritmo di Bellman-Ford.

Introduzione

Si consideri un grafo orientato e pesato G=(V,E), con una funzione peso w: E→***R***.

Il peso w(p) di un cammino p è la somma dei pesi degli archi che lo compongono.

Il peso δ(u,v) di cammino minimo dal vertice u al vertice v è definito come il peso del cammino di peso minimo da u a v se tale cammino esiste, ∞ altrimenti.

Cammini minimi con sorgente singola

Due problemi classici nella teoria dei grafi sono:

- L'identificazione dei pesi di cammini minimi dei vertici di un grafo a partire da un vertice sorgente
- L'identificazione dei vertici che compongono tali cammini minimi dalla sorgente ai vertici del grafo.

Rappresentazione dei cammini minimi

Per rappresentare i cammini minimi si usa normalmente un vettore π che contiene per ogni vertice l'indice del predecessore, oppure NIL se questo non esiste.

Il sottografo $G_\pi=(V_\pi,E_\pi)$ indotto da π è noto come sottografo dei predecessori.

Si può dimostrare che l'insieme dei cammini minimi da un vertice radice costituisce un albero.

Albero dei cammini minimi

Un albero di cammini minimi con radice s è un sottografo orientato G'=(V',E'), in cui V'⊆V e E'⊆E tale che

- V' è l'insieme dei vertici raggiungibili da s
- G' è un albero con radice in s
- per ogni v∈V', l'unico cammino semplice da s a v in G' è un cammino minimo da s a v in G.

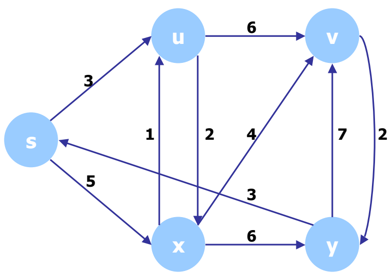
Varianti

Esiste una categoria di problemi riconducibili alla determinazione dei cammini minimi che collegano un vertice sorgente a tutti gli altri vertici in un grafo.

Altri problemi simili sono:

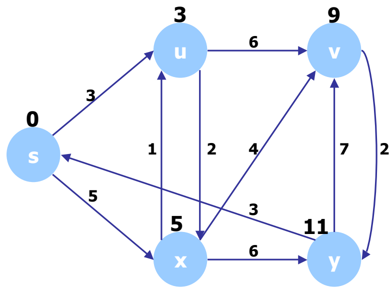
- Identificazione dei cammini minimi con destinazione singola
- Identificazione del cammino minimo tra una coppia di vertici
- Identificazione dei cammini minimi tra tutte le coppie.

Esempio



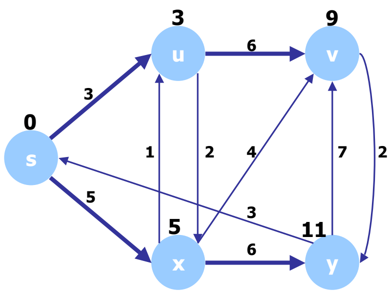
Esempio

Pesi di cammino minimo dal vertice s



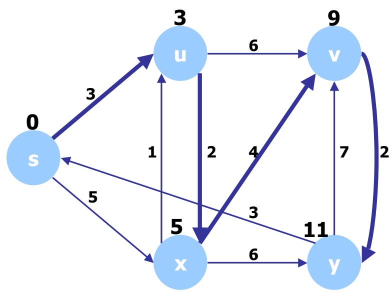
Esempio

Albero dei cammini minimi dal vertice s



Esempio

Altro albero dei cammini minimi dal vertice s



Nota

Nel caso di grafi non pesati, l'identificazione dell'albero dei cammini minimi può essere eseguita tramite la procedura di visita in ampiezza.

Esempio di applicazione

Si supponga di voler determinare il percorso più breve (in termini di kilometri) che collega due luoghi A e B.

Se si modella la carta stradale su cui compaiono A e B come un grafo in cui:

- I vertici sono gli incroci
- Gli archi sono i tratti di strada tra incroci successivi
- I pesi degli archi sono le distanze in kilometri di ciascun arco

Il problema si riconduce a quello dell'identificazione del cammino minimo dal vertice corrispondente ad A al vertice corrispondente a B.

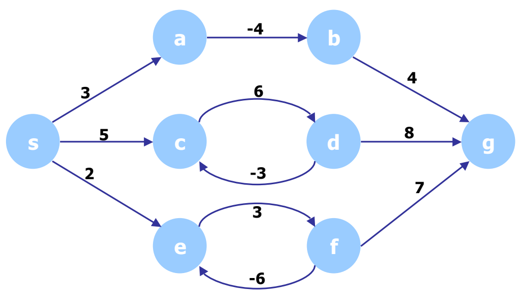
Cicli con pesi negativi

Il problema dell'identificazione dei cammini minimi con sorgente singola non è ben definito se il grafo contiene dei cicli con peso negativo raggiungibili dalla sorgente.

In tal caso non esiste nessun cammino minimo dalla sorgente ad un vertice del ciclo, poiché percorrendo un'ulteriore volta il ciclo il peso del cammino diminuisce.

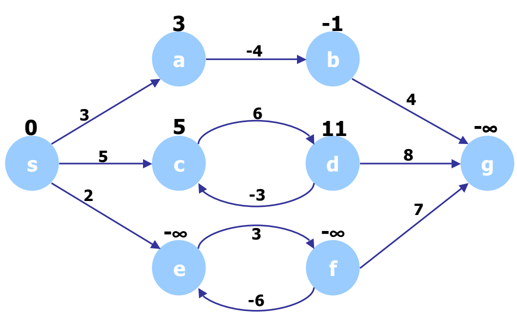
Si usa allora assegnare al peso di cammino minimo del vertice il valore $-\infty$.

Esempio



Esempio

Pesi di cammino minimo dal vertice s



Lemma

Si consideri un grafo orientato e pesato $G=(V,E)$, con una funzione peso $w: E \rightarrow \boldsymbol{R}$.

Sia $p=<v_1, v_2, \ldots, v_k>$ un cammino minimo dal vertice v_1 al vertice v_k.

Per ogni i e j tali che $1 \leq i \leq j \leq k$ sia $p_{ij}=<v_i, v_{i+1}, \ldots, v_j>$ il sottocammino di p dal vertice v_i al vertice v_j.

Allora p_{ij} è un cammino minimo da v_i a v_j.

Corollario

Si consideri un grafo orientato e pesato $G=(V,E)$, con una funzione peso $w: E \rightarrow \boldsymbol{R}$.

Si supponga che un cammino minimo p da una sorgente s ad un vertice v possa essere decomposto in

- un sottocammino da s ad un vertice u
- un arco (u,v).

Allora

$$\delta(s,v)=\delta(s,u)+w(u,v)$$

Lemma

Si consideri un grafo orientato e pesato $G=(V,E)$, con una funzione peso $w: E \rightarrow \boldsymbol{R}$.

Per ogni arco $(u,v) \in E$ vale che

$$\delta(s,v) \leq \delta(s,u)+w(u,v)$$

Rilassamento

È la tecnica sulla quale si basano gli algoritmi seguenti.

Consiste nel

- Mantenere una stima d[u] del peso di cammino minimo relativo a ciascun nodo u
- Rilassare (ossia aggiornare) il valore di d[v] (e conseguentemente di π[v]) verificando se non convenga considerare il cammino che porta a v passando attraverso un certo arco (u,v).

Inizializzazione

Avviene attraverso la seguente procedura

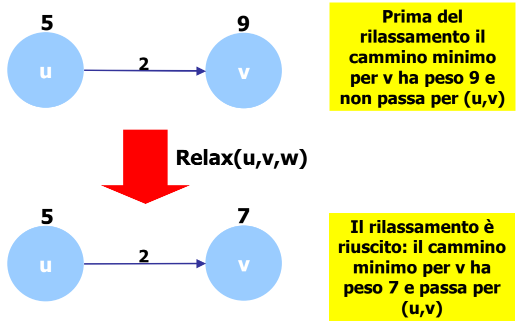
```
Initialize-Single-Source(G,s)
   for ogni vertice v ∈ V[g]
      do  d[v]←∞
          π[v]←NIL
   d[s]←0
```


Rilassamento

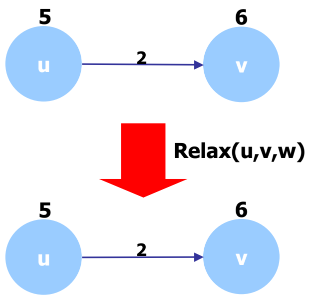
Avviene considerando un arco (u,v) di peso w attraverso la seguente procedura

```
Relax(u,v,w)
      if d[v]>d[u]+w(u,v)
         then d[v]←d[u]+w(u,v)
              π[v]←u
```


Esempio



Esempio



Prima del rilassamento il cammino minimo per v ha peso 6 e non passa per (u,v)

Il rilassamento non è riuscito: il cammino minimo per v ha sempre peso 6 e non passa per (u,v)

Lemma

Si consideri un grafo orientato e pesato G=(V,E), con una funzione peso w: E→***R***.

Sia (u,v) un arco di G.

Dopo il rilassamento di (u,v) si ha che

$$d[v] \le d[u] + w(u,v)$$

Lemma

Si consideri un grafo orientato e pesato G=(V,E), con una funzione peso w: E→***R*** e con vertice sorgente s∈V. G non contiene cicli di peso negativo raggiungibili da s.

Allora dopo che si è chiamata la procedura Initialize-Single-Source(G,s) il sottografo dei predecessori G_π forma un albero radicato avente s come radice.

Inoltre qualunque sequenza di passi di rilassamento sugli archi di G non distrugge questa proprietà.

Lemma

Si consideri un grafo orientato e pesato G=(V,E), con una funzione peso w: E→***R***.

Si è chiamata la procedura Initialize-Single-Source.

Sia s∈V un vertice sorgente.

Allora qualunque sia la sequenza degli archi su cui si è eseguito il rilassamento e per ogni vertice v si ha che

$$d[v] \ge \delta(s,v)$$

Una volta che d[v] raggiunge il valore di $\delta(s,v)$ non cambia più.

Uso del rilassamento

Gli algoritmi che seguono si basano sul rilassamento.

Essi differiscono per

- numero di volte in cui ogni arco è rilassato
- ordine con cui si rilassano gli archi.

Sommario

- Introduzione
- Algoritmo di Dijkstra
- Algoritmo di Bellman-Ford.

Algoritmo di Dijkstra

Risolve il problema del calcolo dei cammini minimi con sorgente singola su un grafo orientato e pesato nel caso in cui tutti i pesi degli archi siano non negativi.

L'algoritmo usa una strategia di tipo greedy.

Funzionamento

L'algoritmo mantiene un insieme S contenente i vertici il cui peso di cammino minimo da s è già stato determinato.

Ad ogni passo l'algoritmo

- seleziona il vertice u in V-S con la minima stima di cammino minimo
- inserisce u in S
- rilassa tutti gli archi uscenti da u.

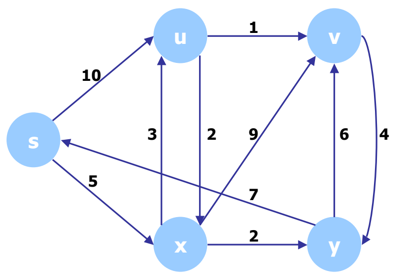
I vertici in V-S sono mantenuti in una coda prioritaria Q avente come chiave per ogni vertice u il valore d[u].

Pseudo-codice

Dijkstra(G,w,s)

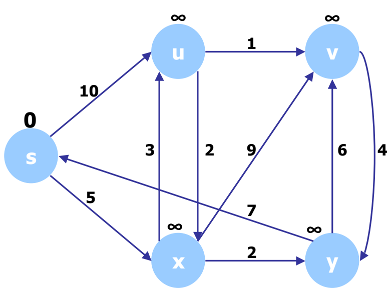
Initialize-Single-Source(G,s)
$S \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
    do $u \leftarrow$ Extract-Min(Q)
        $S \leftarrow S \cup \{u\}$
        for ogni vertice $v \in Adj[u]$
            do Relax(u,v,w)

Esempio



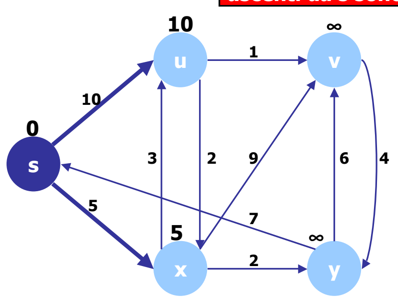
Esempio (0)

Inizializzazione: accanto ad ogni vertice v è rappresentato il valore d[v].



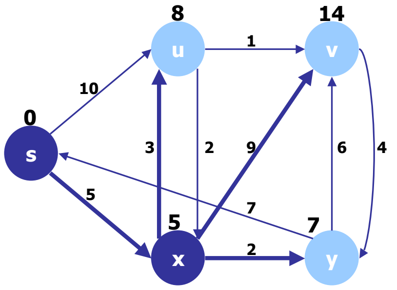
Esempio (1)

Dalla coda si estrae il vertice s, che viene inserito in S; tutti gli archi uscenti da s sono rilassati.



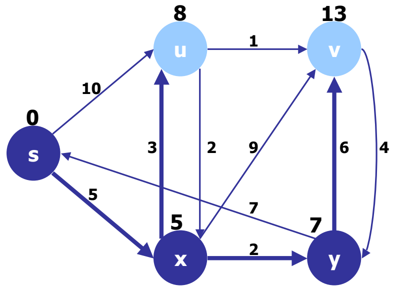
Esempio (2)

Dalla coda si estrae il vertice x, che viene inserito in S; tutti gli archi uscenti da x sono rilassati.



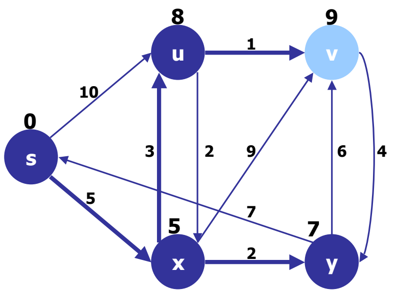
Esempio (3)

Dalla coda si estrae il vertice y, che viene inserito in S; tutti gli archi uscenti da y sono rilassati.



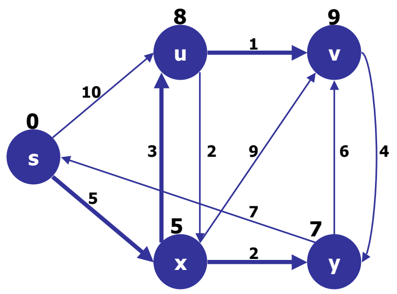
Esempio (4)

Dalla coda si estrae il vertice u, che viene inserito in S; tutti gli archi uscenti da u sono rilassati.



Esempio (5)

Dalla coda si estrae il vertice v, che viene inserito in S; tutti gli archi uscenti da v sono rilassati.



Complessità

La coda è inizializzata con tutti i vertici. Nessun vertice entra nella coda in seguito. Quindi il ciclo viene eseguito V volte.

Dijkstra(G,w,s)
Initialize-Single-Source(G,s)
S←∅
Q←V[G]
while Q≠∅
do u←Extract-Min(Q)
S←S∪{u}
for ogni vertice v∈Adj[u]
do Relax(u,v,w)

Se per realizzare la coda con priorità si usa uno heap l'operazione ha un costo O(lg V)

Complessità

Il ciclo viene ripetuto al più una volta per ogni arco, quindi |E| volte.

Dijkstra(G,w,s)
Initialize-Single-Source(G,s)
S←∅
Q←V[G]
while Q≠∅
do u←Extract-Min(Q)
S←S∪{u}
for ogni vertice v∈Adj[u]
do Relax(u,v,w)

Modifica i campi d dei vertici nella coda; l'operazione ha un costo O(lg V)

Complessità

L'algoritmo di Dijkstra richiede un tempo di esecuzione che è O((V+E) lg V).

Sommario

- Introduzione
- Algoritmo di Dijkstra
- Algoritmo di Bellman-Ford.

Algoritmo di Bellman-Ford

L'algoritmo di Bellman-Ford risolve il problema dell'identificazione dei cammini minimi con sorgente singola nel caso più generale in cui i pesi degli archi possano essere negativi.

L'algoritmo restituisce

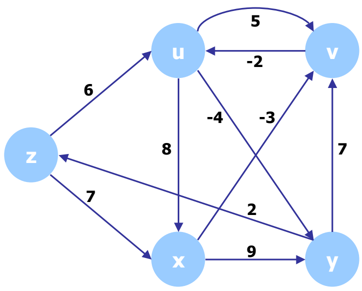
- il valore FALSE se trova un ciclo di peso negativo raggiungibile dalla radice
- il valore TRUE e l'albero dei cammini minimi in caso contrario.

Pseudo-codice

```
Bellman-Ford(G,w,s)
  Initialize-Single-Source(G,s)
  for i←1 to |V(G)|-1
     do for ogni arco (u,v)∈E(G)
         do relax(u,v,w)
  for ogni arco (u,v)∈E(G)
     do if d[v]>d[u]+w(u,v)
         then  return FALSE
  return TRUE
```

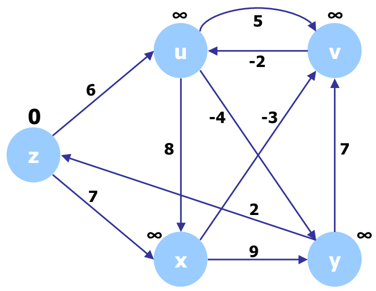

Esempio

5 vertici
10 archi
Sorgente z
Nessun ciclo di peso negativo



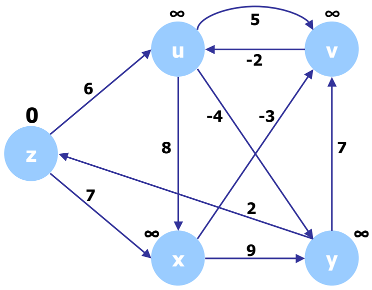
Esempio (0)

Inizializzazione: accanto ad ogni vertice v è rappresentato il valore d[v].



Esempio (1)

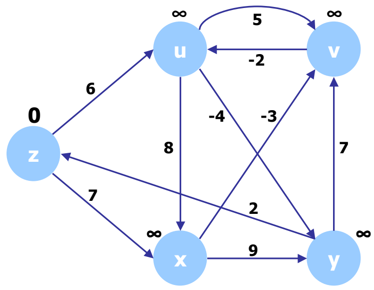
Ciclo 1.1
Gli archi sono rilassati in ordine lessicografico: arco (u,v)



→ (u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (2)

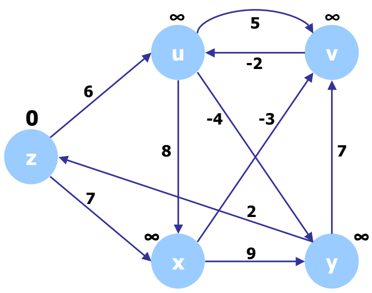
Ciclo 1.2
Gli archi sono rilassati in ordine lessicografico: arco (u,x)



(u,v)
→ (u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (3)

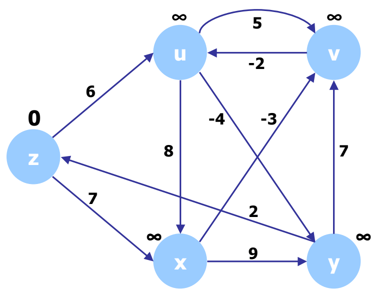
Ciclo 1.3
Gli archi sono rilassati in ordine lessicografico: arco (u,y)



(u,v)
(u,x)
→ (u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (4)

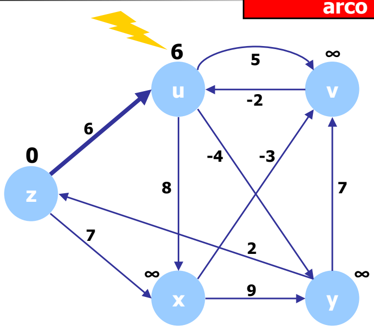
Ciclo 1.4
Gli archi sono rilassati in ordine lessicografico: arco (v,u)



(u,v)
(u,x)
(u,y)
→ (v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (10)

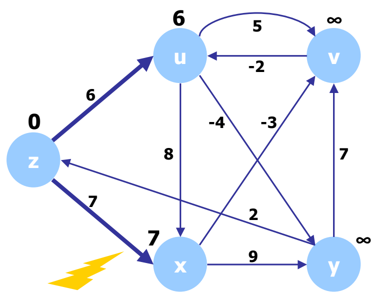
Ciclo 1.9
Gli archi sono rilassati in ordine lessicografico: arco (z,u)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
→ (z,u)
(z,x)

Esempio (11)

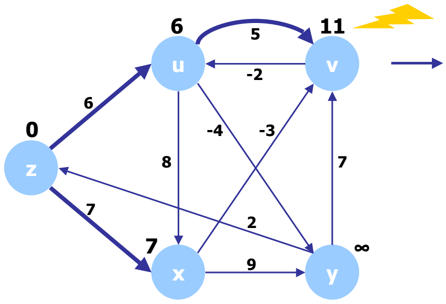
Ciclo 1.10
Gli archi sono rilassati in ordine lessicografico: arco (z,x)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
→ (z,x)

Esempio (12)

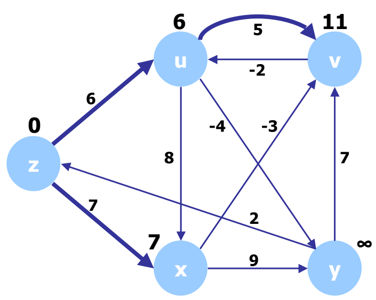
Ciclo 2.1
Gli archi sono rilassati in ordine lessicografico: arco (u,v)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (13)

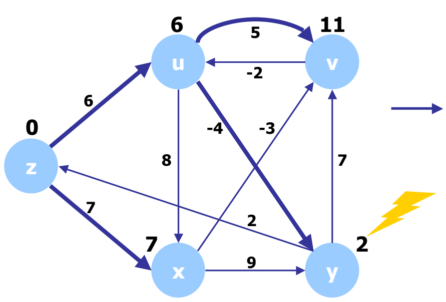
Ciclo 2.2
Gli archi sono rilassati in ordine lessicografico: arco (u,x)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (14)

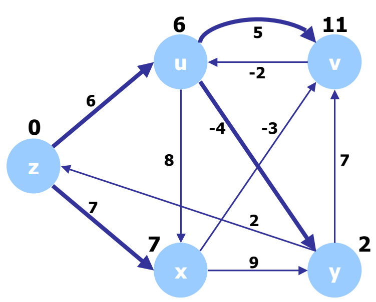
Ciclo 2.3
Gli archi sono rilassati in ordine lessicografico: arco (u,y)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (15)

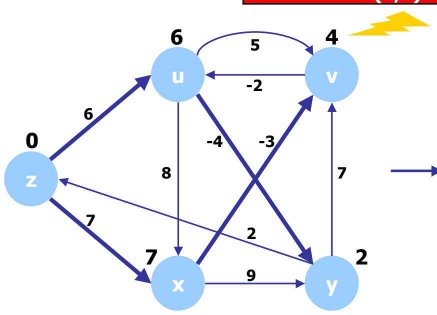
Ciclo 2.4
Gli archi sono rilassati in ordine lessicografico: arco (v,u)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (16)

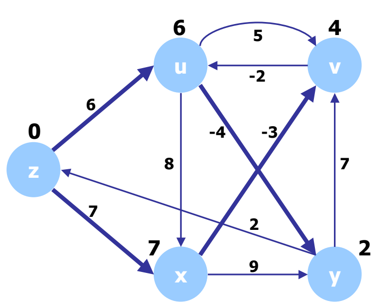
Ciclo 2.5
Gli archi sono rilassati in ordine lessicografico: arco (x,v)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (17)

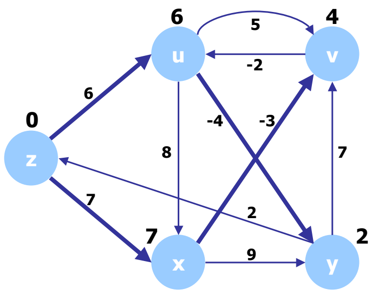
Ciclo 2.6
Gli archi sono rilassati in ordine lessicografico: arco (x,y)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (18)

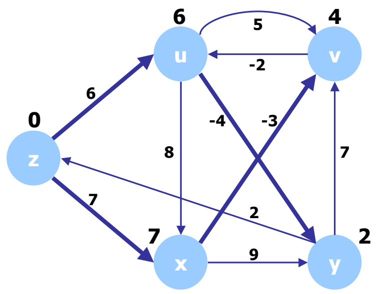
Ciclo 2.7
Gli archi sono rilassati in ordine lessicografico: arco (y,v)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (19)

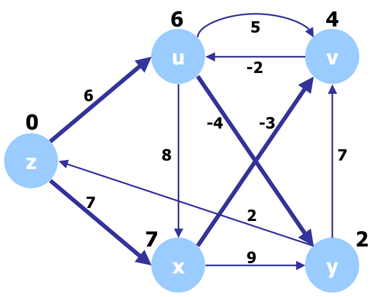
Ciclo 2.8
Gli archi sono rilassati in ordine lessicografico: arco (y,z)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (20)

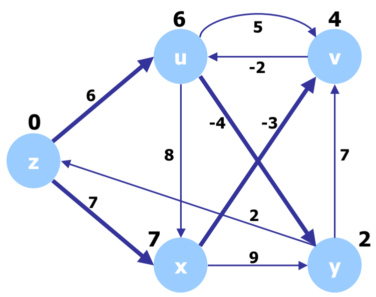
Ciclo 2.9
Gli archi sono rilassati in ordine lessicografico: arco (z,u)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (21)

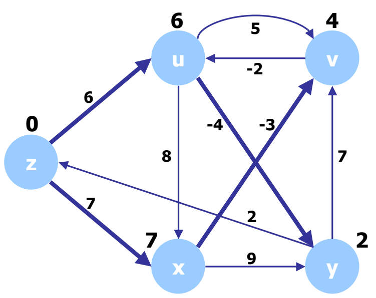
Ciclo 2.10
Gli archi sono rilassati in ordine lessicografico: arco (z,x)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (22)

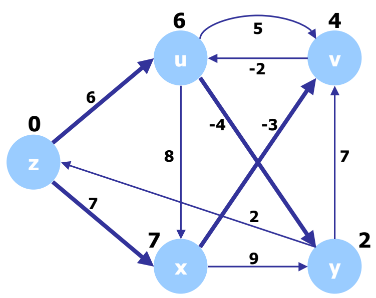
Ciclo 3.1
Gli archi sono rilassati in ordine lessicografico: arco (u,v)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (23)

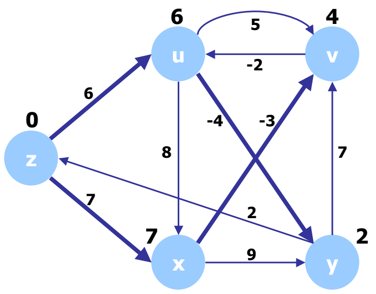
Ciclo 3.2
Gli archi sono rilassati in ordine lessicografico: arco (u,x)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (24)

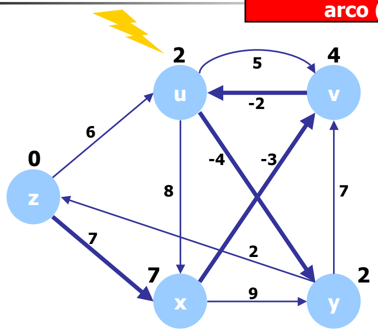
Ciclo 3.3
Gli archi sono rilassati in ordine lessicografico: arco (u,y)



(u,v)
(u,x)
→ (u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (25)

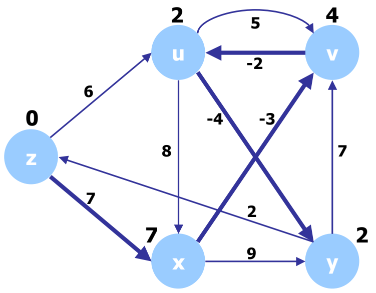
Ciclo 3.4
Gli archi sono rilassati in ordine lessicografico: arco (v,u)



(u,v)
(u,x)
(u,y)
→ (v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (32)

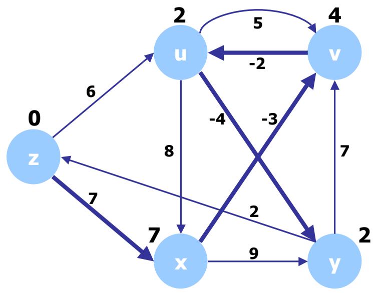
Ciclo 4.1
Gli archi sono rilassati in ordine lessicografico: arco (u,v)



→ (u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (33)

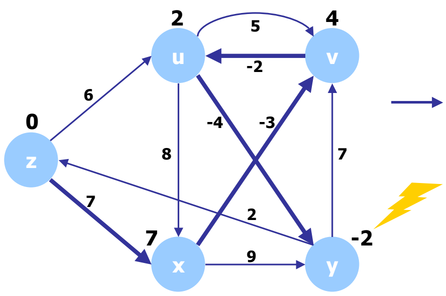
Ciclo 4.2
Gli archi sono rilassati in ordine lessicografico: arco (u,x)



(u,v)
→ (u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (34)

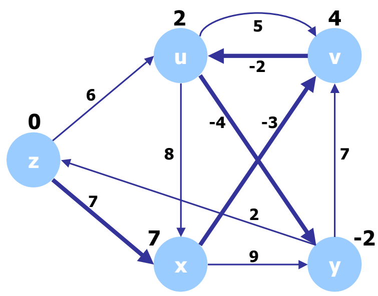
Ciclo 4.3
Gli archi sono rilassati in ordine lessicografico: arco (u,y)



(u,v)
(u,x)
→ (u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
(z,x)

Esempio (41)

Ciclo 4.10
Gli archi sono rilassati in ordine lessicografico: arco (z,x)



(u,v)
(u,x)
(u,y)
(v,u)
(x,v)
(x,y)
(y,v)
(y,z)
(z,u)
→ (z,x)

Complessità

```
Bellman-Ford(G,w,s)
  Initialize-Single-Source(G,s)
  for i←1 to |V(G)|-1
     do for ogni arco (u,v)∈E(G)
          do relax(u,v,w)
  for ogni arco (u,v)∈E(G)
     do if d[v]>d[u]+w(u,v)
          then  return FALSE
  return TRUE
```

O(V)

O(VE)

O(E)

Complessità

L'algoritmo di Bellman-Ford richiede un tempo di esecuzione O(VE).