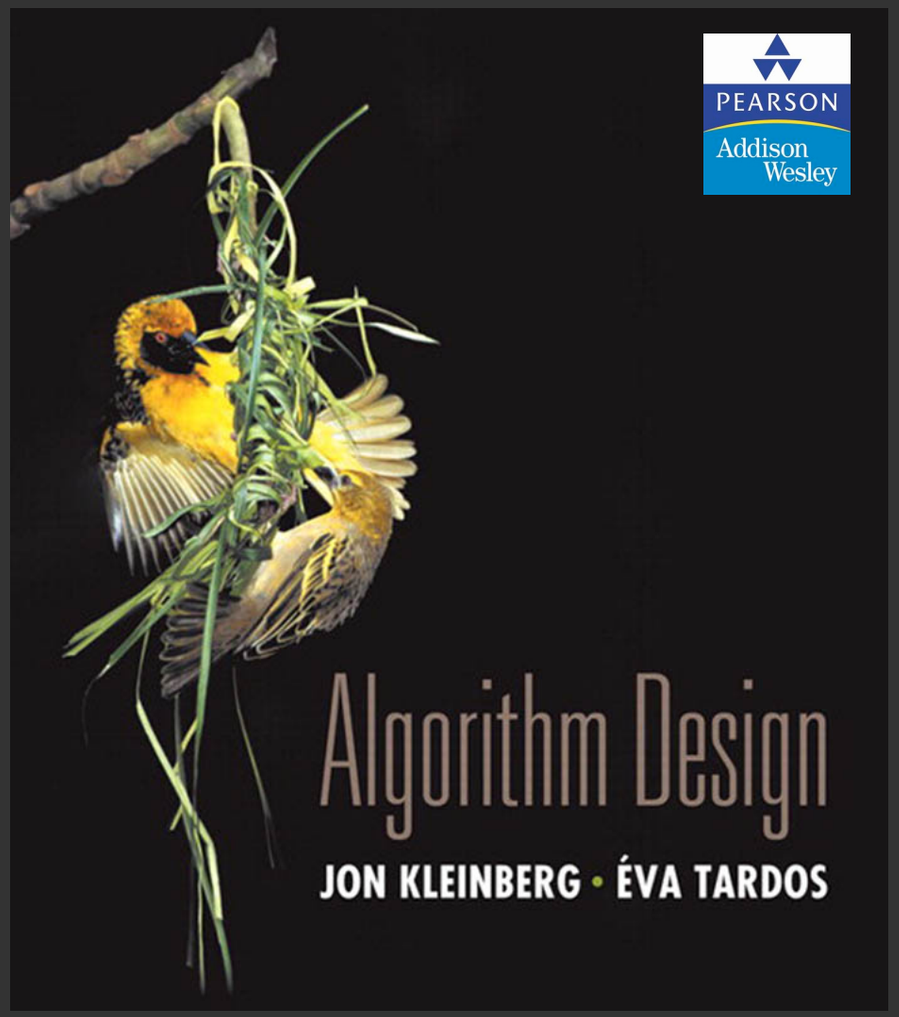


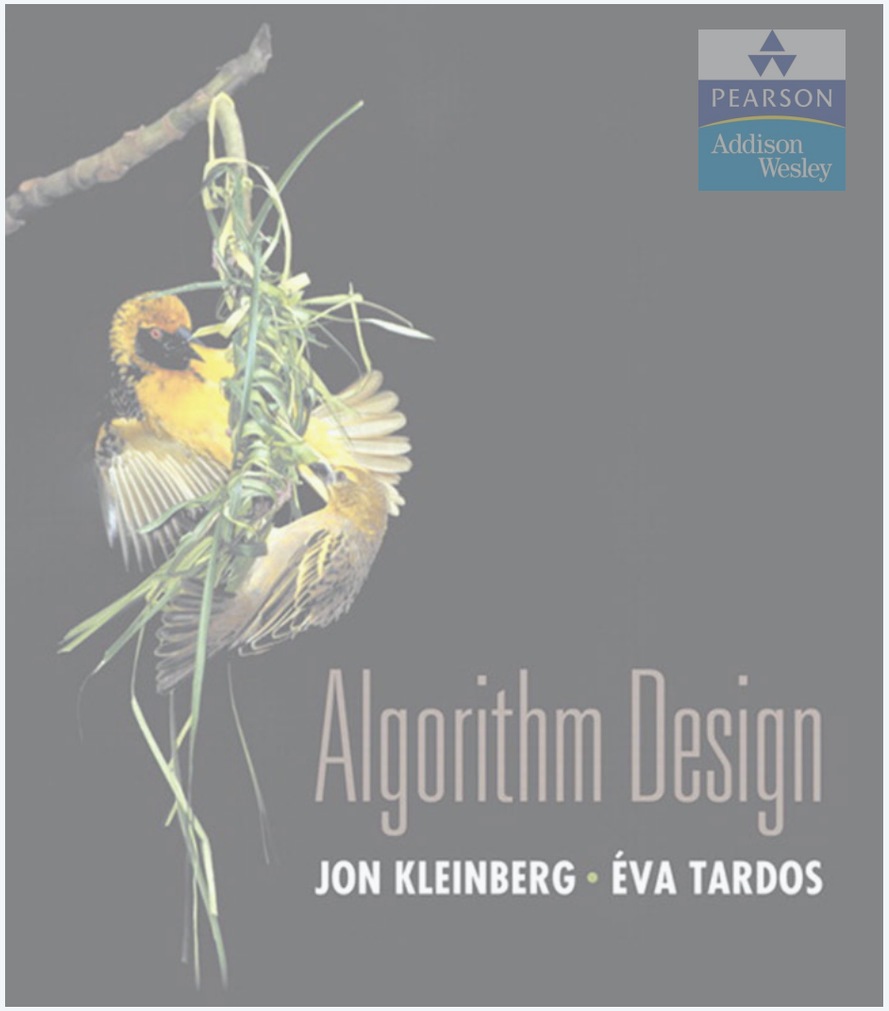

# 6. Programmazione Dinamica II

- *algoritmo Bellman–Ford–Moore*
- *protocolli distance-vector*
- *cicli negativi*

**Traduzione e adattamento di Vincenzo Bonifaci**
**Original lecture slides by Kevin Wayne**


**http://www.cs.princeton.edu/~wayne/kleinberg-tardos**

Last updated on 24/04/2020 09:37




SECTION 6.8

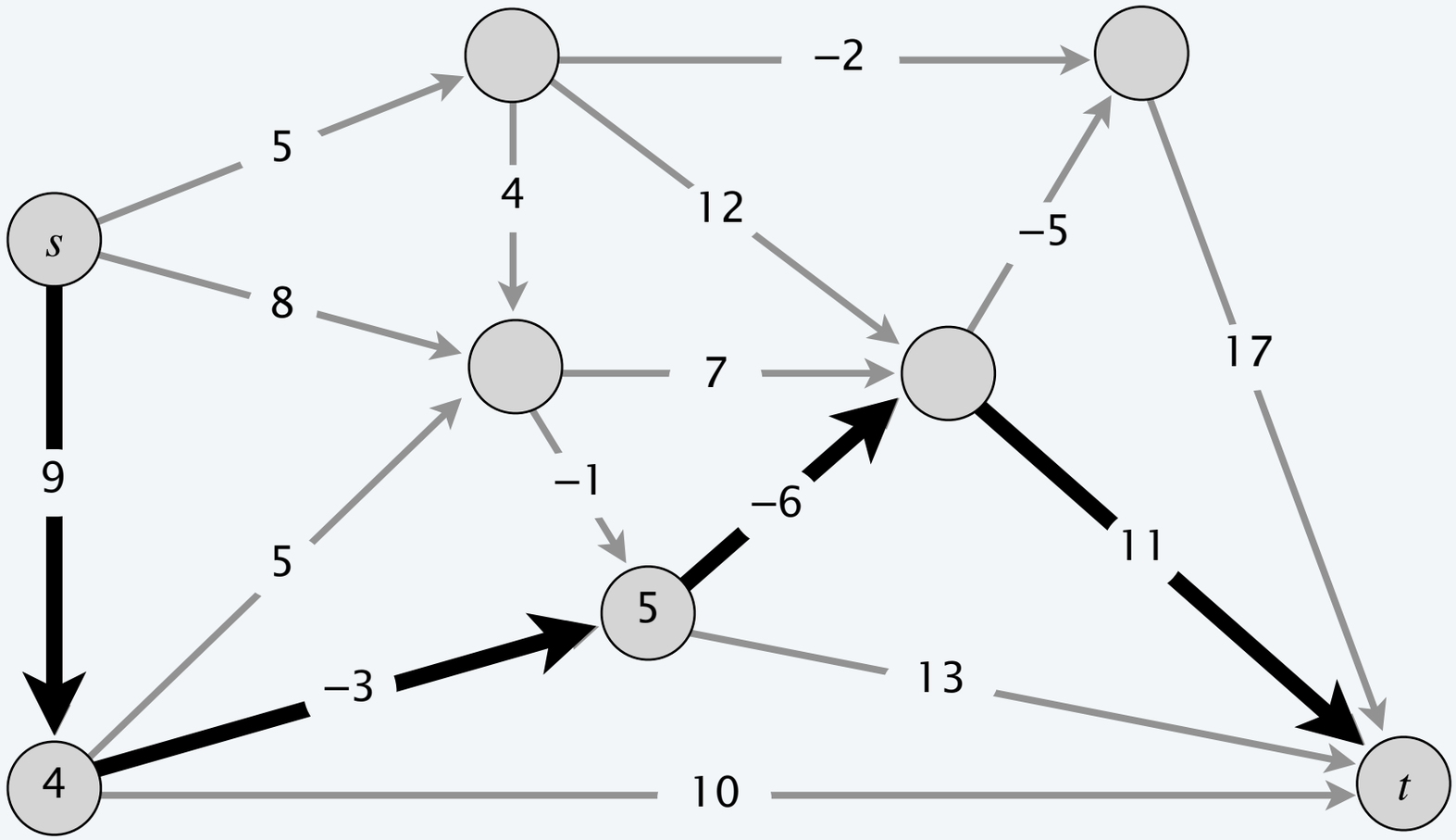
# 6. PROGRAMMAZIONE DINAMICA II

- *sequence alignment*
- *Hirschberg's algorithm*
- *algoritmo Bellman–Ford–Moore*
- *distance-vector protocols*
- *negative cycles*

# Cammino minimo con pesi negativi

Problema del cammino minimo. Dato un digrafo $G = (V, E)$, pesi arbitrari sugli archi $\ell_{vw}$, trovare il cammino minimo da un nodo sorgente $s$ ad un nodo destinazione $t$.

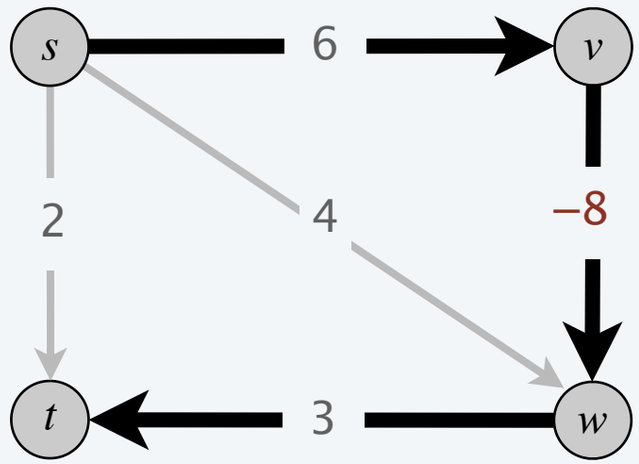
assume l'esistenza di un cammino minimo da $s$ a $t$




**lunghezza del cammino minimo $s \to t$ = 9 − 3 − 6 + 11 = 11**

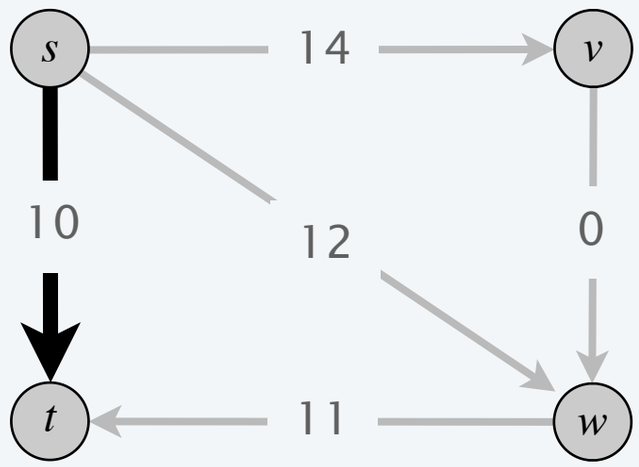
# Cammino minimo con pesi negativi: approcci fallimentari

Dijkstra. Può non trovare il cammino minimo quando i pesi sono negativi.




Dijkstra processa i nodi nell'ordine $s$, $t$, $w$, $v$
Ma il cammino minimo da $s$ a $t$ è $s \to v \to w \to t$.

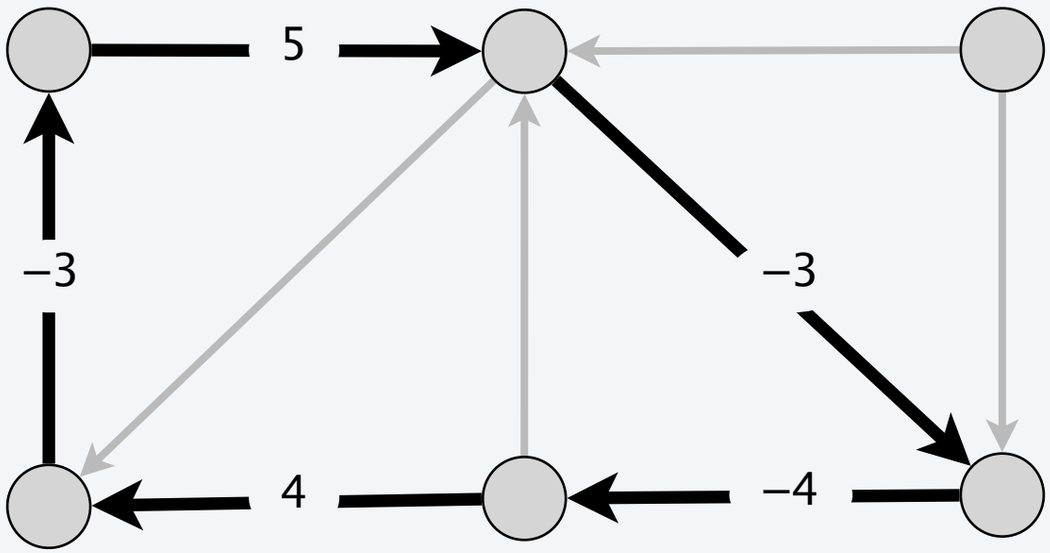
Ripesatura. Aggiungere una costante ad ogni arco non fa necessariamente sì che Dijkstra trovi il cammino minimo.




Aggiungere 8 ad ogni peso cambia il cammino minimo da $s \to v \to w \to t$ a $s \to t$.

# Cicli negativi

Def. Un ciclo negativo è un ciclo orientato per il quale la somma delle lunghezze degli archi è negativa.

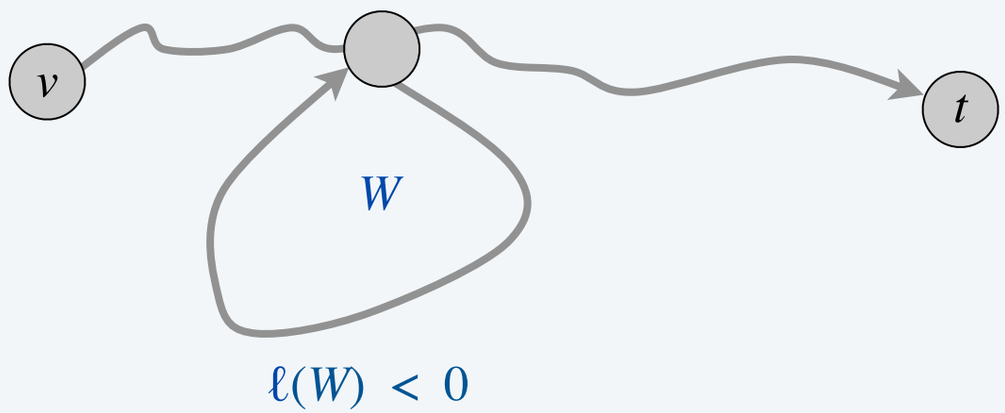



**un ciclo negativo W :** $\ell(W) = \sum_{e \in W} \ell_e < 0$

# Cammini minimi e cicli negativi

Lemma 1. Se qualche cammino $v \rightsquigarrow t$ contiene un ciclo negativo, allora non esiste un cammino minimo $v \rightsquigarrow t$.

Dim. Se esiste un ciclo siffatto $W$, allora si può costruire un cammino $v \rightsquigarrow t$ di lunghezza arbitrariamente negativa girando per $W$ tante volte quanto lo si desidera. ▪

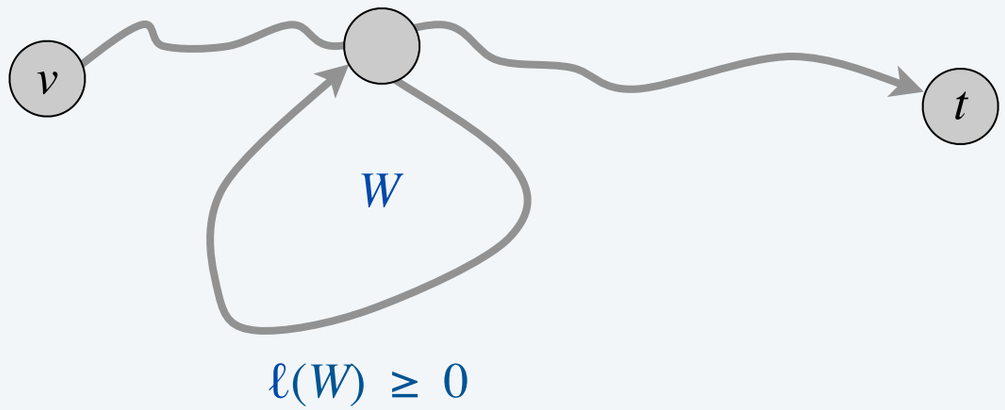


# Cammini minimi e cicli negativi

Lemma 2. Se $G$ non ha cicli negativi, allora esiste un cammino minimo $v \rightsquigarrow t$ che è un cammino semplice (ed ha $\leq n - 1$ archi).

Dim.

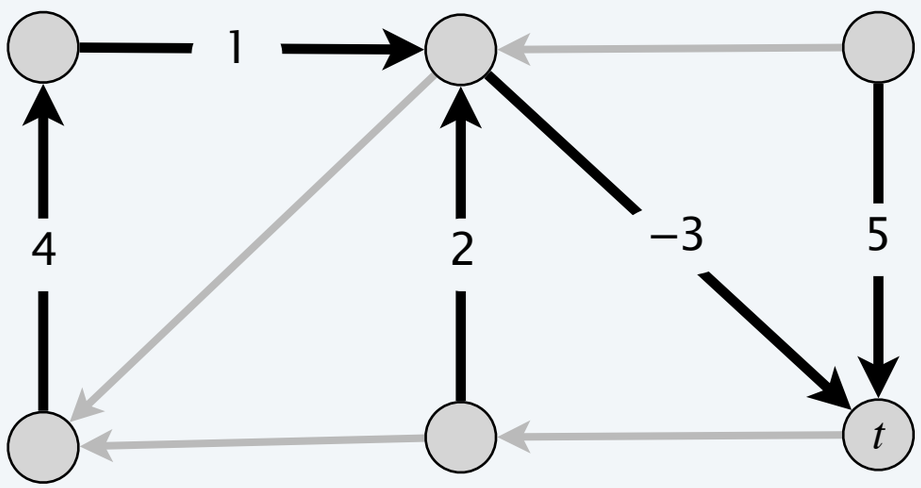
- Tra tutti i cammini minimi $v \rightsquigarrow t$, consideriamone uno che usa il minor numero di archi.
- Se quel cammino $P$ contiene un ciclo orientato $W$, possiamo rimuovere la porzione di $P$ corrispondente a $W$ senza aumentare la lunghezza del cammino. ▪



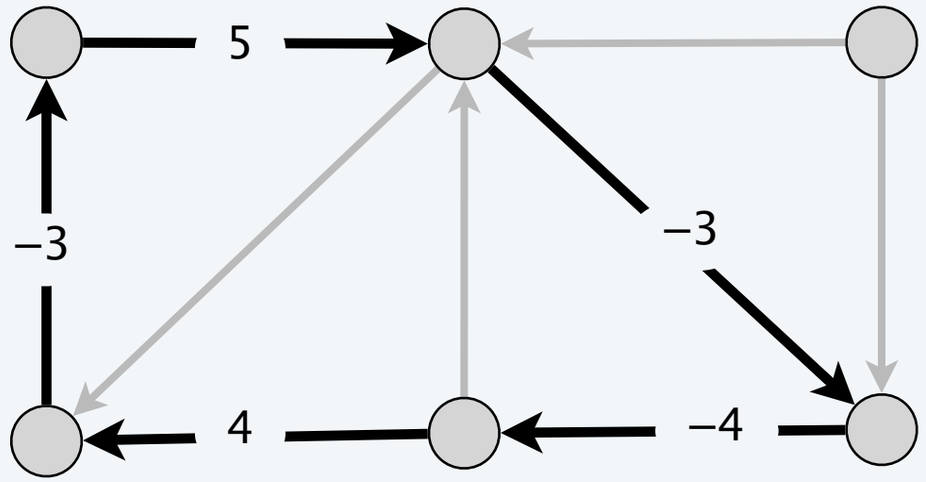
# Problemi di cammini minimi e di cicli negativi

Problema dei cammini minimi a destinazione singola. Dato un digrafo $G = (V, E)$ con lunghezze degli archi $\ell_{vw}$ (ma senza cicli negativi) e un nodo $t$, trovare un cammino minimo $v \rightsquigarrow t$ per ciascun nodo $v$.

Problema del ciclo negativo. Dato un digrafo $G = (V, E)$ con lunghezze degli archi $\ell_{vw}$, trovare un ciclo negativo (se ne esiste uno).




**albero dei cammini minimi**




**ciclo negativo**

# Cammini minimi con pesi negativi: programmazione dinamica

Def. $OPT(i, v)$ = lunghezza di un cammino minimo $v \rightsquigarrow t$ che usa $\leq i$ archi.

Scopo. $OPT(n-1, v)$ per ogni $v$.

dal Lemma 2, se non ci sono cicli negativi, esiste un cammino minimo $v \rightsquigarrow t$ che è semplice

Caso 1. Il cammino minimo $v \rightsquigarrow t$ usa $\leq i-1$ archi.

- $OPT(i, v) = \mathrm{OPT}(i-1, v)$.

proprietà di sottostruttura ottima

Caso 2. Il cammino minimo $v \rightsquigarrow t$ usa esattamente $i$ archi.

- Se $(v, w)$ è il primo arco in un cammino siffatto $v \rightsquigarrow t$, incorri costo $\ell_{vw}$.
- Dopodiché, scegli il miglior cammino $w \rightsquigarrow t$ che usa $\leq i-1$ archi.

Equazione di Bellman.

$$OPT(i,v) = \begin{cases} 0 & \text{if } i = 0 \text{ and } v = t \\ \infty & \text{if } i = 0 \text{ and } v \neq t \\ \min\left\{ OPT(i-1,v),\ \min\limits_{(v,w)\in E} \{OPT(i-1,w) + \ell_{vw}\} \right\} & \text{if } i > 0 \end{cases}$$

# Cammini minimi con pesi negativi: implementazione

SHORTEST-PATHS($V$, $E$, $\ell$, $t$)

FOREACH nodo $v \in V$ :

$M[0, v] \leftarrow \infty$.

$M[0, t] \leftarrow 0$.

FOR i = 1 TO $n - 1$

FOREACH nodo $v \in V$ :

$M[i, v] \leftarrow M[i-1, v]$.

FOREACH arco $(v, w) \in E$ :

$M[i, v] \leftarrow \min \{ M[i, v],\ M[i-1, w] + \ell_{vw} \}$.

# Cammini minimi con pesi negativi: implementazione

Teorema 1. Dato un digrafo $G = (V, E)$ senza cicli negativi, l'algoritmo di PD calcola la lunghezza di un cammino minimo $v \rightsquigarrow t$ per ciascun nodo $v$ in tempo $\Theta(mn)$ e spazio $\Theta(n^2)$.

Dim.

- La tabella richiede spazio $\Theta(n^2)$.
- Ogni iterazione $i$ prende tempo $\Theta(m)$ perché ogni arco è esaminato una volta. ▪

Ricostruire i cammini minimi.

- Approccio 1: Mantenere un $successor[i, v]$ che punta al nodo successivo su un cammino minimo $v \rightsquigarrow t$ con $\leq i$ archi.
- Approccio 2: Calcolare le lunghezze minime $M[i, v]$ e considerare solo gli archi con $M[i, v] = M[i - 1, w] + \ell_{vw}$. Qualunque cammino orientato in questo sottografo è un cammino minimo.

# Programmazione dinamica: quiz 6

**È facile modificare l'algoritmo di PD per i cammini minimi per…**

**A.** Calcolare le lunghezze dei cammini in tempo $O(mn)$ e spazio $O(m + n)$.

**B.** Calcolare i cammini minimi in tempo $O(mn)$ e spazio $O(m + n)$.

**C.** Sia A che B.

**D.** Né A né B.

# Cammini minimi con pesi negativi: miglioramenti pratici

Ottimizzazione dello spazio. Manteniamo due array 1D (invece dell'array 2D).

- $d[v]$ = lunghezza del cammino minimo $v \rightsquigarrow t$ trovato finora.
- $successor[v]$ = nodo successivo sul cammino $v \rightsquigarrow t$.

Ottimizzazione della performance. Se $d[w]$ non è stato modificato all'iterazione $i - 1$, non è necessario considerare gli archi entranti in $w$ all'iterazione $i$.

# Bellman–Ford–Moore: implementazione efficiente

BELLMAN–FORD–MOORE($V$, $E$, $c$, $t$)

FOREACH node $v \in V$ :

  $d[v] \leftarrow \infty$.

  $successor[v] \leftarrow null$.

$d[t] \leftarrow 0$.

FOR $i = 1$ TO $n - 1$

  FOREACH nodo $w \in V$ :

    IF ($d[w]$ è stato modificato nell'iterazione precedente)

      FOREACH arco $(v, w) \in E$ :

        IF ($d[v] > d[w] + \ell_{vw}$)

          $d[v] \leftarrow d[w] + \ell_{vw}$.

          $successor[v] \leftarrow w$.

  IF (nessun valore $d[\cdot]$ modificato nell'iterazione $i$) STOP.

passata $i$-esima
tempo $O(m)$

# Bellman-Ford-Moore: analisi

Lemma 3. Per ogni nodo $v$ : $d[v]$ è la lunghezza di qualche cammino $v \rightarrow t$.

Lemma 4. Per ogni nodo $v$ : $d[v]$ è monotona non crescente nel tempo.

Lemma 5. Dopo l'iterazione $i$, $d[v] \leq$ lunghezza di un cammino minimo $v \rightarrow t$ che usa $\leq i$ archi.

Dim. [ per induzione su $i$ ]

- Caso base: $i = 0$.
- Assumiamo sia vero dopo il passo $i$.
- Sia $P$ un qualunque cammino $v \rightarrow t$ con $\leq i + 1$ archi.
- Sia $(v, w)$ il primo arco di $P$ e sia $P'$ il sottocammino da $w$ a $t$.
- Per ipotesi induttiva, alla fine dell'iterazione $i$, $d[w] \leq \ell(P')$ perché $P'$ è un cammino $w \rightarrow t$ con $\leq i$ archi.
- Poiché $(v, w)$ è stato considerato nell'iterazione $i + 1$:

e per il Lemma 4, $d[w]$ non aumenta

$$
\begin{aligned}
d[v] &\leq \ell_{vw} + d[w] \\
&\leq \ell_{vw} + \ell(P') \\
&= \ell(P) \quad \blacksquare
\end{aligned}
$$

e per il Lemma 4, $d[v]$ non aumenta

# Bellman–Ford–Moore: analisi

Teorema 2. In assenza di cicli negativi, Bellman–Ford–Moore calcola le lunghezze dei cammini minimi $v \rightsquigarrow t$ in tempo $O(mn)$ e in spazio $\Theta(n)$.

Dim. Lemma 2 + Lemma 5. ▪

il cammino minimo esiste ed ha al più $n-1$ archi

dopo $i$ iterazioni, $d[v] \leq$ lunghezza di un cammino minimo che usa $\leq i$ archi

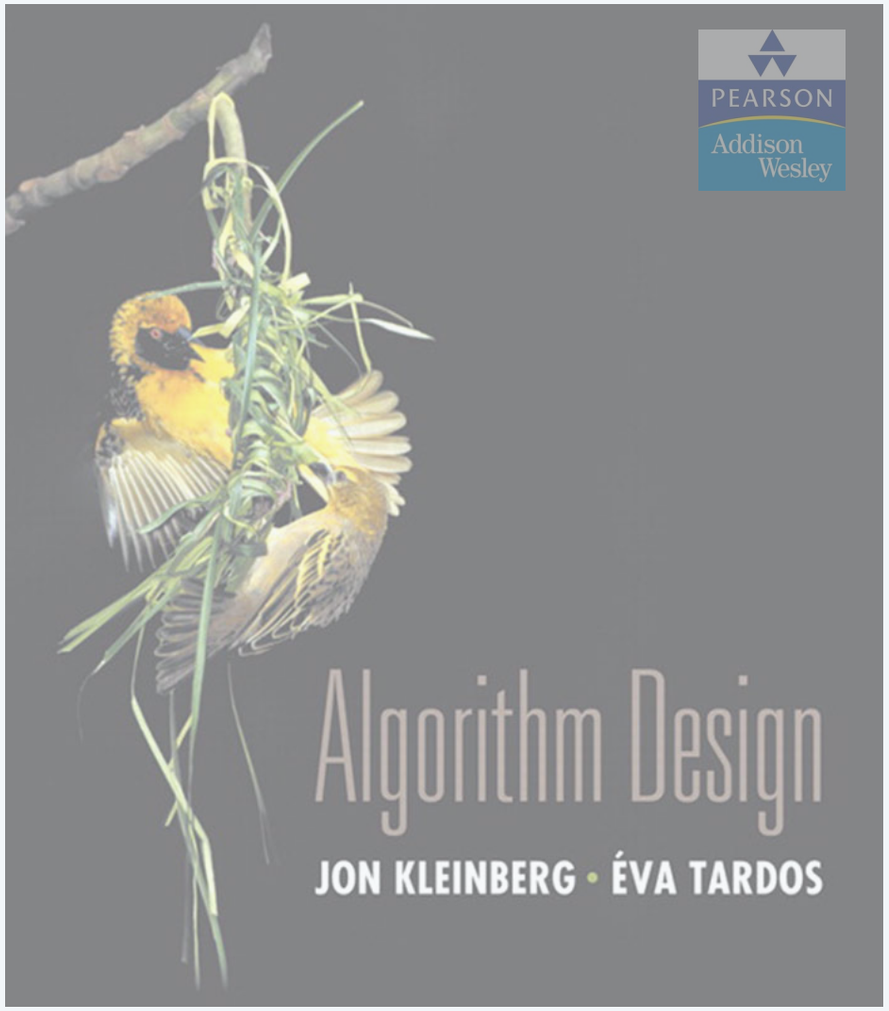
Nota. Bellman–Ford–Moore è di norma più efficiente in pratica di quanto suggerisce il Teorema.

- L'arco $(v, w)$ è considerato all'iterazione $i + 1$ solo se $d[w]$ è aggiornato all'iterazione $i$.
- Se il cammino minimo ha $k$ archi, l'algoritmo lo trova dopo $\leq k$ iterazioni.

# Cammini minimi a singola destinazione con pesi negativi

| anno | caso peggiore | scoperto da |
|---|---|---|
| **1955** | $O(n^4)$ | Shimbel |
| **1956** | $O(m\, n^2\, W)$ | Ford |
| **1958** | $O(m\, n)$ | Bellman, Moore |
| **1983** | $O(n^{3/4}\, m \log W)$ | Gabow |
| **1989** | $O(m\, n^{1/2} \log(nW))$ | Gabow–Tarjan |
| **1993** | $O(m\, n^{1/2} \log W)$ | Goldberg |
| **2005** | $O(n^{2.38}\, W)$ | Sankowsi, Yuster–Zwick |
| **2016** | $\tilde{O}(n^{10/7} \log W)$ | Cohen–Mądry–Sankowski–Vladu |
| **20xx** |  | |

**cammini minimi a singola destinazione con pesi tra –W e W**




SECTION 6.9

# 6. PROGRAMMAZIONE DINAMICA II

- *sequence alignment*
- *Hirschberg's algorithm*
- *Bellman–Ford–Moore algorithm*
- *protocolli distance-vector*
- *negative cycles*

# Protocolli di instradamento basati su vettori delle distanze

Rete di comunicazione.

- Nodo ≈ router.
- Arco ≈ link di comunicazione diretta.
- Lunghezza di un arco ≈ latenza del link. ← non-negativa, ma Bellman–Ford–Moore è usato comunque

Algoritmo di Dijkstra. Richiede informazioni globali sulla rete.

Bellman–Ford–Moore. Usa solo informazioni locali dai nodi vicini.

Sincronizzazione. Non si può pretendere che i router siano sincronizzati. L'ordine in cui gli archi sono processati in Bellman–Ford–Moore non è importante. Inoltre, l'algoritmo converge anche se l'elaborazione è asincrona.

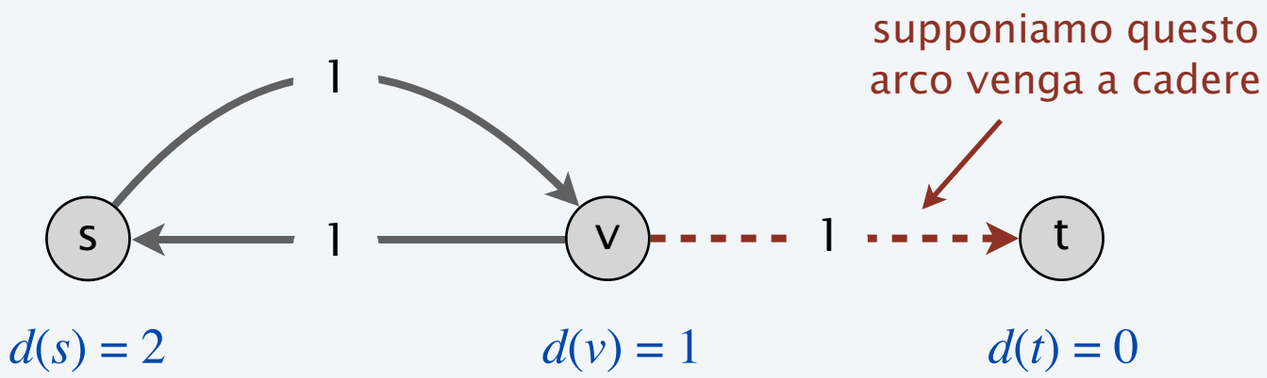
# Protocolli distance-vector

Protocolli di instradamento distance-vector. [ "routing by rumor" ]

- Ogni router mantiene un vettore di lunghezze di cammini minimi verso ogni altro nodo (distanze) e il primo passo di ogni cammino (direzioni).
- Algoritmo: ogni router effettua $n$ calcoli separati, uno per ogni potenziale nodo di destinazione.

Es. RIP, Xerox XNS RIP, Novell's IPX RIP, Cisco's IGRP, DEC's DNA Phase IV, AppleTalk's RTMP.

Caveat. Le lunghezze possono cambiare durante l' algoritmo (o divenire infinite).




**"conteggio all'infinito"**

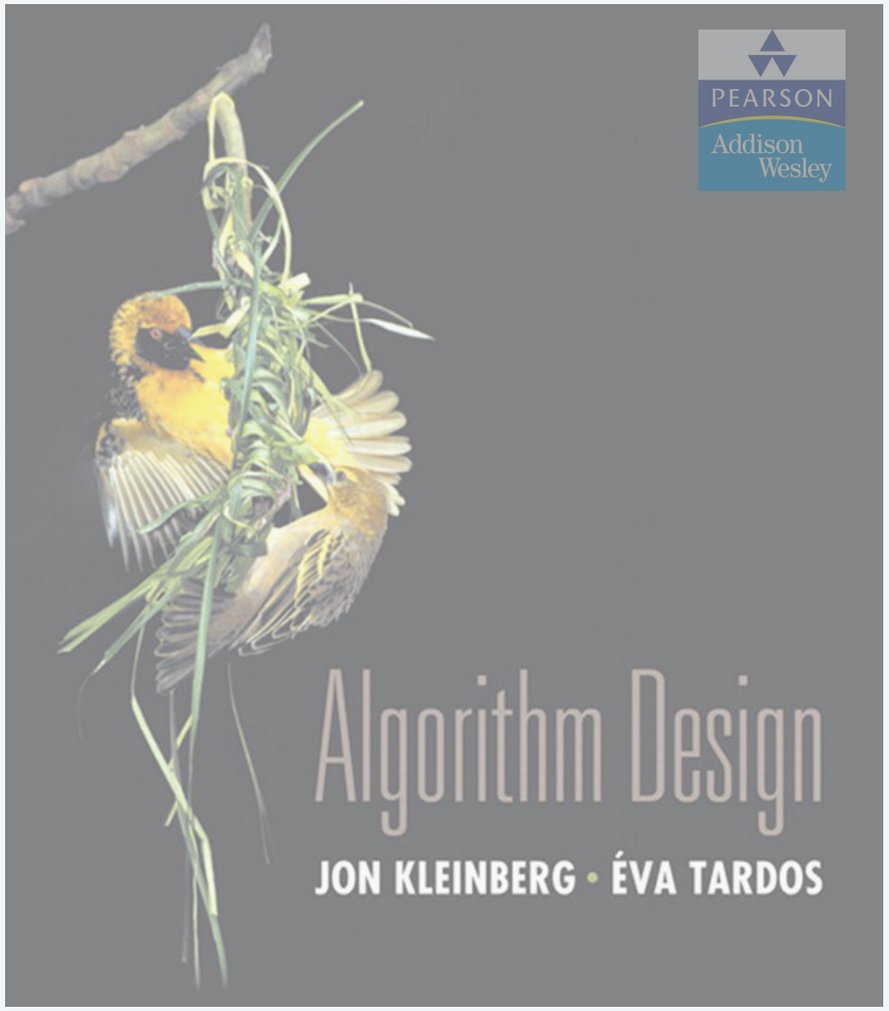
# Protocolli di instradamento path-vector

Protocolli di instradamento path-vector.

non solo la distanza
e il primo passo

- Ogni router memorizza l'intero percorso (o la topologia della rete).
- Basati sull'algoritmo di Dijkstra.
- Evitano il problema del “conteggio all'infinito” e le relative difficoltà.
- Richiedono molta più memoria.

Es. Border Gateway Protocol (BGP), Open Shortest Path First (OSPF).



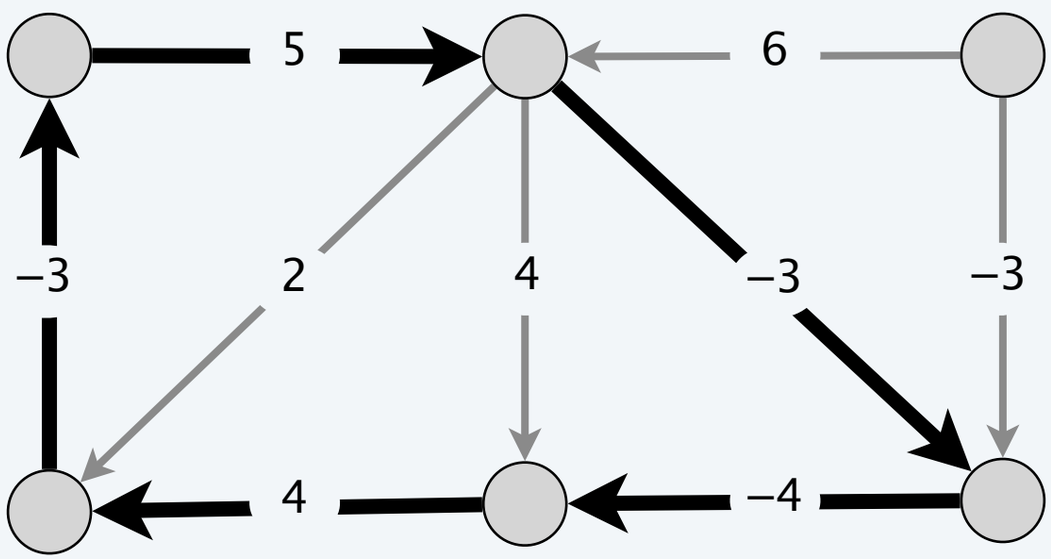

SECTION 6.10

# 6. PROGRAMMAZIONE DINAMICA II

- *sequence alignment*
- *Hirschberg's algorithm*
- *Bellman–Ford–Moore algorithm*
- *distance vector protocol*
- *cicli negativi*

# Individuare cicli negativi

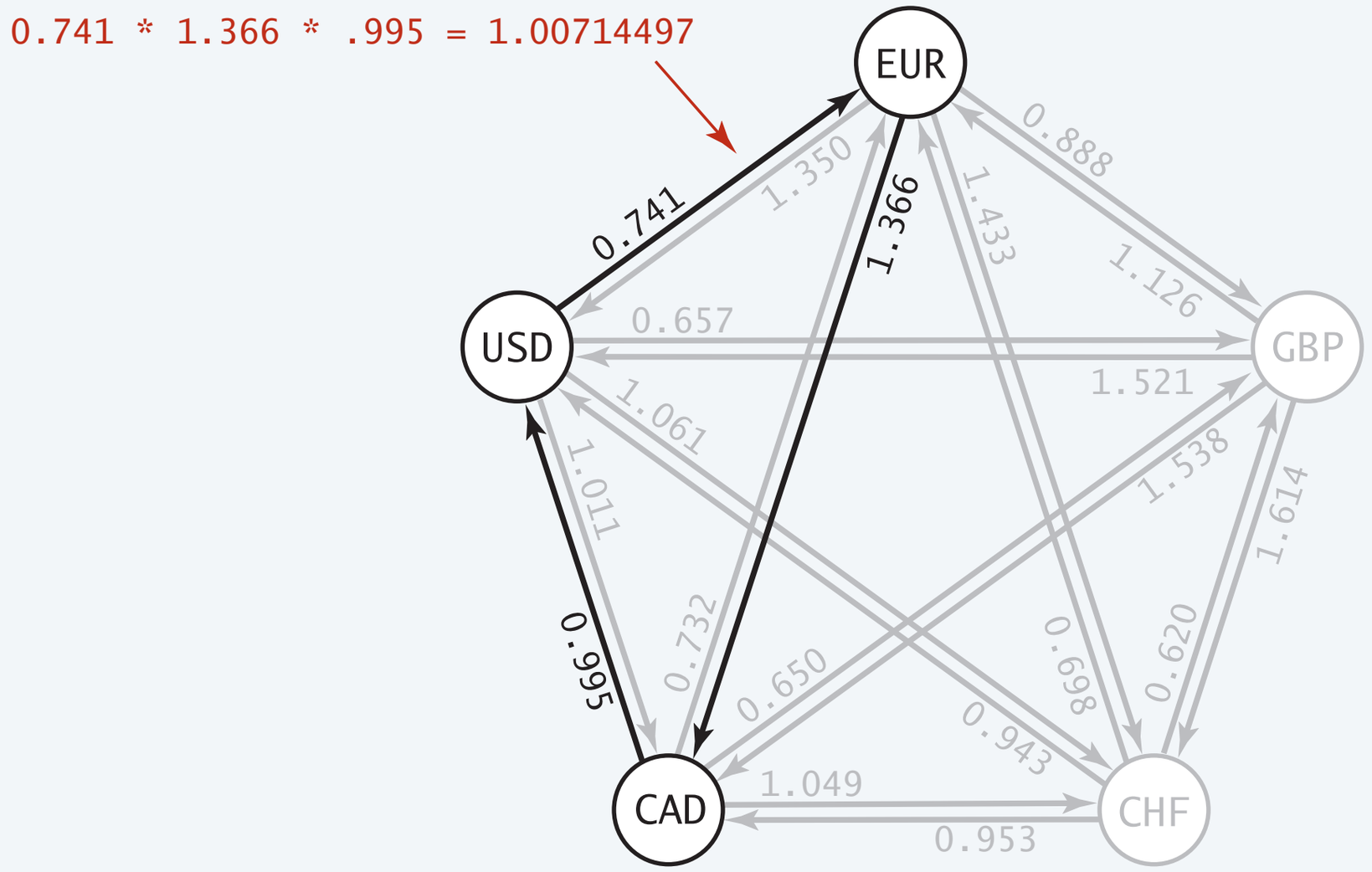
Problema del ciclo negativo. Dato un digrafo $G = (V, E)$, con lunghezze degli archi $\ell_{vw}$, trovare un ciclo negativo (se ne esiste uno).



# Individuare cicli negativi: applicazione

Cambi di valuta. Date $n$ valute e i tassi di cambio tra coppie di valute, esiste una possibilità di arbitraggio?

Nota. L'algoritmo più veloce per questo problema ha un certo valore!



# Individuare cicli negativi

Lemma 7. Se $OPT(n, v) = \mathrm{OPT}(n-1, v)$ per ogni $v$, non esistono cicli negativi.

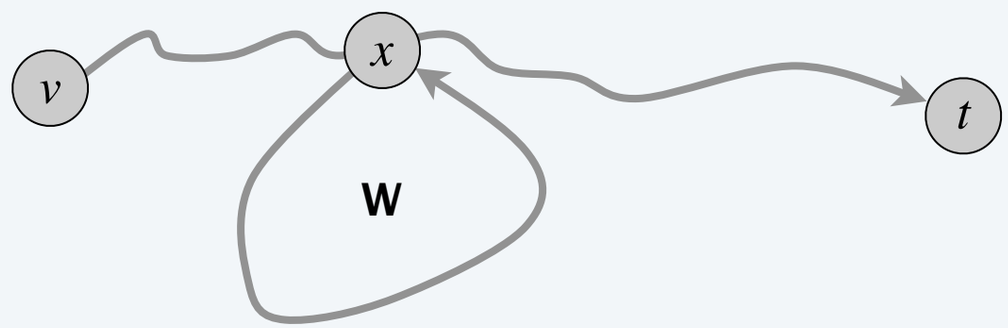
Dim.Se i valori $OPT(n, v)$ si sono stabilizzati $\Rightarrow$ cammino minimo $v \rightsquigarrow t$ esiste. ▪

Lemma 8. Se $OPT(n, v) < \mathrm{OPT}(n-1, v)$ per qualche $v$, allora (ogni) cammino minimo $v \rightsquigarrow t$ di lunghezza $\leq n$ contiene un ciclo $W$, e $W$ è un ciclo negativo.

Dim. [per assurdo]

- Poiché $OPT(n, v) < \mathrm{OPT}(n-1, v)$, sappiamo che il cammino minimo $v \rightsquigarrow t$ $P$ ha esattamente $n$ archi.
- Per il principio della piccionaia, $P$ contiene un nodo ripetuto $x$.
- Sia $W$ un qualunque ciclo di $P$.
- Rimuovere $W$ dà un cammino $v \rightsquigarrow t$ con $< n$ archi $\Rightarrow$ $W$ è un ciclo negativo.

▪



# Individuare cicli negativi

Teorema 4. Si può trovare un ciclo negativo in tempo $\Theta(mn)$ e spazio $\Theta(n^2)$.

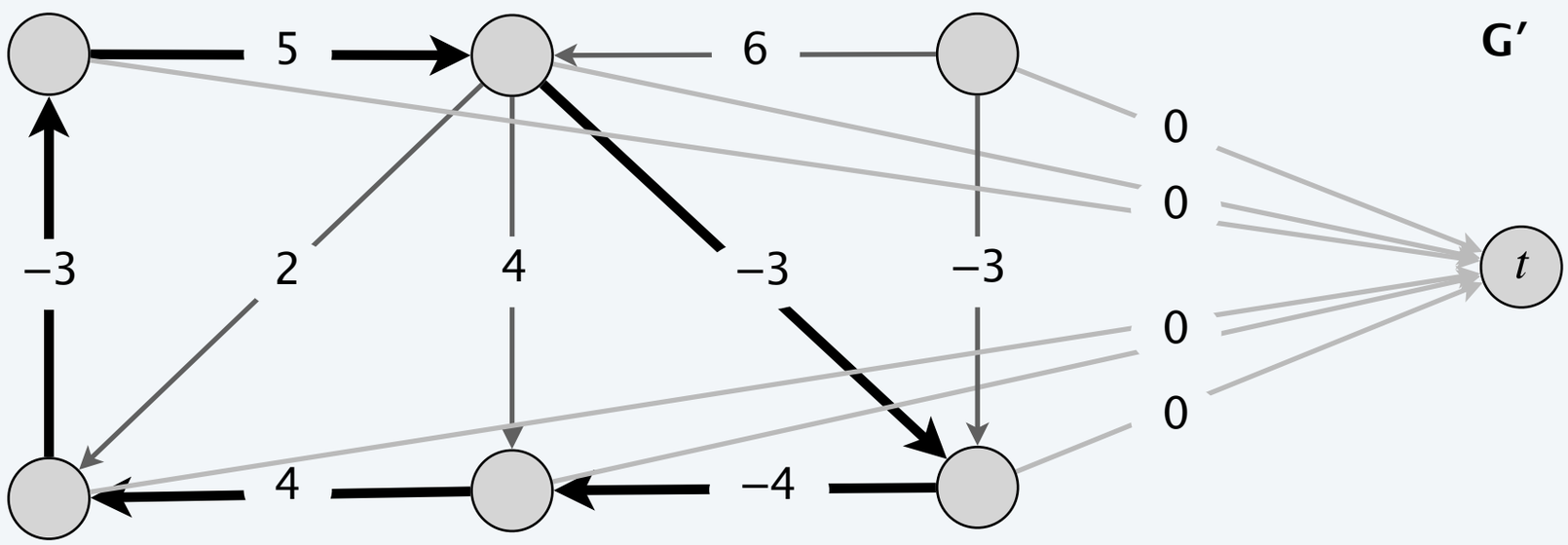
Dim.

- Aggiungiamo un pozzo $t$ e colleghiamo tutti i nodi a $t$ con archi a costo 0.
- $G$ ha un ciclo negativo sse $G'$ ha un ciclo negativo.
- Caso 1. [ $OPT(n, v) = OPT(n-1, v)$ per ogni nodo $v$ ]

  Per il Lemma 7, non ci sono cicli negativi.
- Caso 2. [ $OPT(n, v) < OPT(n-1, v)$ per qualche nodo $v$ ]

  Come nel Lemma 8, si può estrarre un ciclo negativo dal percorso $v \rightsquigarrow t$.

  (il ciclo non può contenere $t$ perché nessun arco esce da $t$) ▪



## Individuare cicli negativi

Teorema 5. Si può trovare un ciclo negativo in tempo $O(mn)$ e spazio aggiuntivo $O(n)$.

Dim.

- Esegui Bellman–Ford–Moore su $G'$ per $n' = n + 1$ passate (invece di $n' - 1$).
- Se nessun valore $d[v]$ è aggiornato nella passata $n'$, non ci sono cicli negativi.
- Altrimenti, supponiamo che $d[s]$ sia aggiornato alla passata $n'$.
- Definiamo $pass(v)$ = ultima passata in cui $d[v]$ è stato aggiornato.
- Notiamo che $pass(s) = n'$ e $pass(successor[v]) \geq pass(v) - 1$ per ogni $v$.
- Seguendo i puntatori di *successor*, prima o poi un nodo si ripete.
- Lemma 6 $\Rightarrow$ il ciclo corrispondente è un ciclo negativo. ▪

Nota. Vedi p. 304 per una versione migliorata.