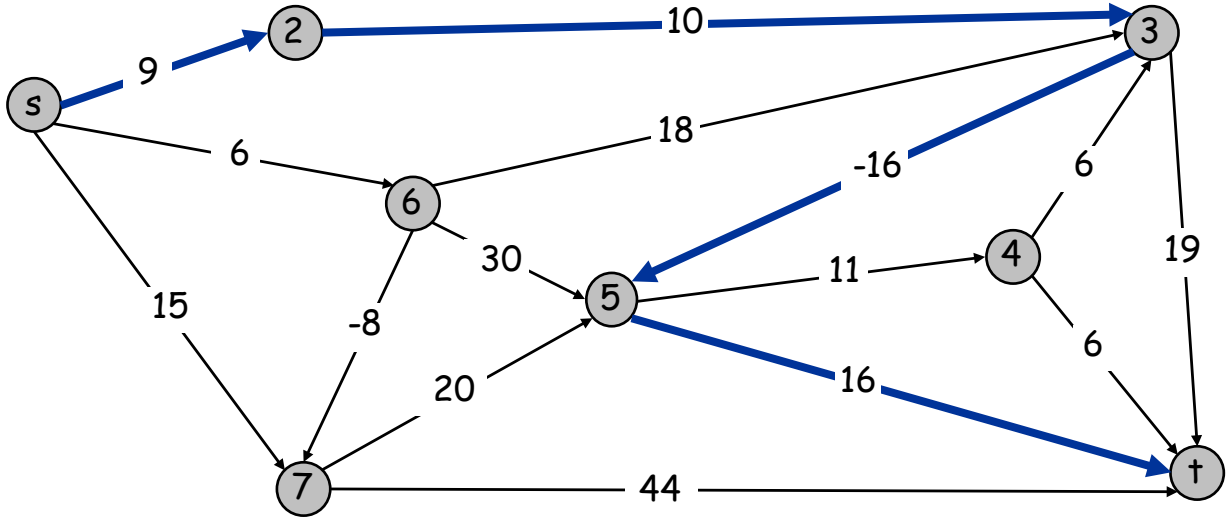
Cammini minimi

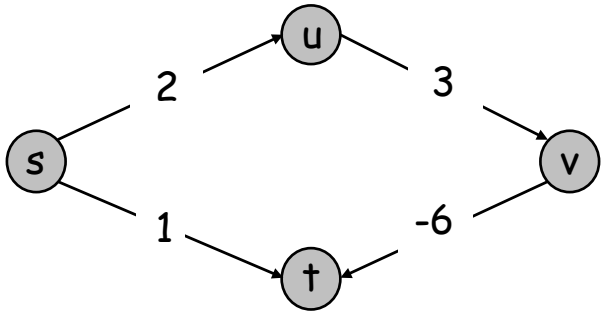
- Problema del percorso piu` corto. Dato un grafo direzionato $G = (V, E)$, con pesi degli archi $c_{vw}$, trovare il percorso piu` corto da s a t.

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

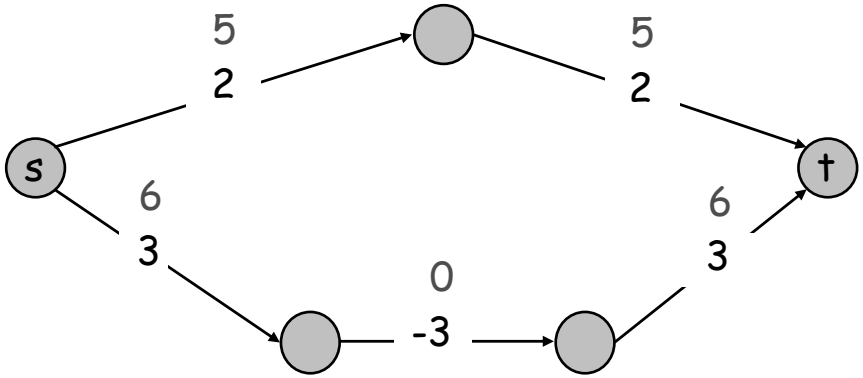


Cammini minimi in presenza di archi con costo negativo

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

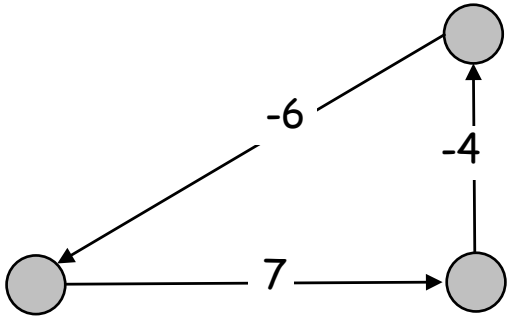


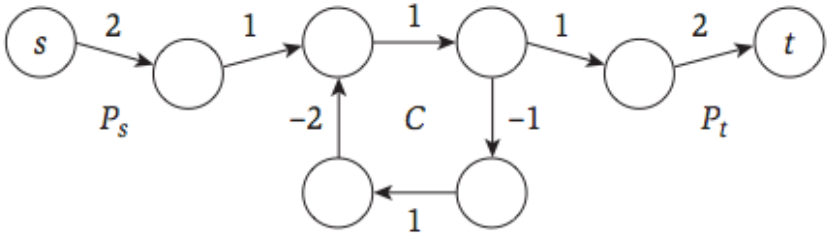
- Re-weighting. Aggiungere una costante positiva ai pesi degli archi potrebbe non funzionare.



Cammini minimi in presenza di archi con costo negativo

- Ciclo di costo negativo.



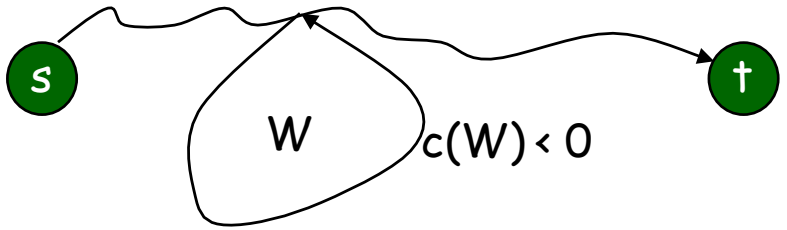


Cammini minimi in presenza di archi con costo negativo

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. Sia P un percorso da s a t con un ciclo C di costo negativo –c. Ogni 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 .

Sia ora P un percorso da s a t 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 l' ipotesi deve avere costo positivo. In questo caso potremmo rimuovere le porzioni di P tra due occorrenze consecutive di v in P senza far aumentare il costo del percorso.



Cammini minimi: Programmazione dinamica

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

  - Caso 1: P usa al piu` i-1 archi.
    - OPT(i, v) = OPT(i-1, v)

  - Caso 2: P usa esattamente i archi.
    - se (v, w) e` il primo arco allora P e` formato da (v, w) e dal percorso piu` corto da w a t di al piu` i-1 archi

$$OPT(i,v) = \min_{(v,w)\in E} \left\{ OPT(i-1,\ w) + c_{vw} \right\}$$

$$OPT(i,v) = \begin{cases} 0 & \text{se } i = 0 \text{ e } v = t \\ \infty & \text{se } i = 0 \text{ e } v \neq t \\ \min\left\{ OPT(i-1,\ v),\ \min_{(v,w)\in E} \left\{ OPT(i-1,\ w) + c_{vw} \right\} \right\} & \text{altrimenti} \end{cases}$$

Cammini minimi: Programmazione dinamica

$$OPT(i,v) = \begin{cases} 0 & \text{se } i = 0 \text{ e } v = t \\ \infty & \text{se } i = 0 \text{ e } v \neq t \\ \min\left\{ OPT(i-1,\ v),\ \min_{(v,w)\in E} \left\{ OPT(i-1,\ w) + c_{vw} \right\} \right\} & \text{altrimenti} \end{cases}$$

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

Ecco dove….

Osservazione. 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 il percorso piu` corto da v a t e` semplice e di conseguenza contiene al piu` n-1 archi

Algoritmo di Bellman-Ford per i cammini minimi

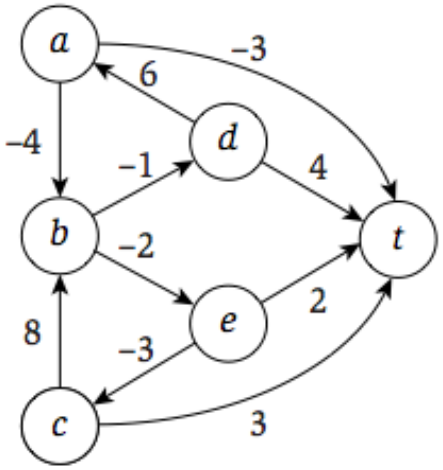
```
Shortest-Path(G, t) {
   foreach node v ∈ V
      M[0, v] ← ∞
   M[0, t] ← 0

   for i = 1 to n-1
      foreach node v ∈ V
         M[i, v] ← M[i-1, v]
         foreach edge (v, w) ∈ E
            M[i, v] ← min { M[i, v], M[i-1, w] + c_vw }
}
```

- Analisi. Tempo $\Theta(mn)$, spazio $\Theta(n^2)$.

- Trovare il percorso minimo. Memorizzare un successore per ogni entrata della matrice

Bellman-Ford: esempio



(a)

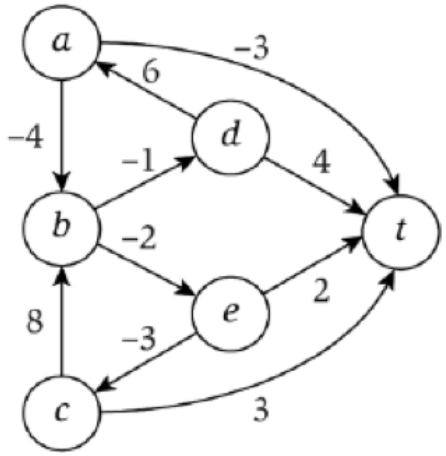
	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	−3	−3	−4	−6	−6
b	∞	∞	0	−2	−2	−2
c	∞	3	3	3	3	3
d	∞	4	3	3	2	0
e	∞	2	0	0	0	0

(b)

In questa tabella qui l'elemento di indice (u,i) corrisponde al valore M(i,u)

Bellman-Ford: esempio

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



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

OPT(5,a) = min(OPT(4,a), min(OPT(4,t)+c_{at}, OPT(4,b)+c_{ab}))
= min(-6, min(0-3,-2-4))

OPT(4,a) = min(OPT(3,a), min(OPT(3,t)+c_{at}, OPT(3,b)+c_{ab})) a-b
= min(-4, min(0-3,-2-4))

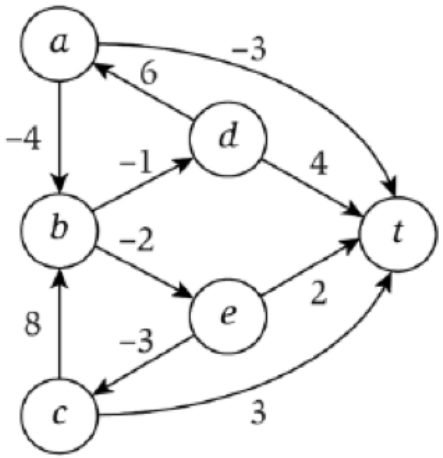
OPT(3,b) = min(OPT(2,b), min(OPT(2,e)+c_{be}, OPT(2,d)+c_{bd})) b-e
= min(0, min(0-2,3-1))

OPT(2,e) = min(OPT(1,e), min(OPT(1,c)+c_{ec}, OPT(1,t)+c_{et})) e-c
= min(2, min(3-3,0+2))

OPT(1,c) = min(OPT(0,c), min(OPT(0,b)+c_{cb}, OPT(0,t)+c_{ct})) c-t
= min(∞, min(∞+8,0+3))

a - b - e - c - t
-4 -2 -3 +3 = -6

Bellman-Ford: esempio



$OPT(5,a) = \min(OPT(4,a), \min(OPT(4,t)+c_{at}, OPT(4,b)+c_{ab}))$
$= \min(-6, \min(0-3, -2-4))$

$OPT(4,a) = \min(OPT(3,a), \min(OPT(3,t)+c_{at}, OPT(3,b)+c_{ab}))$ a-b
$= \min(-4, \min(0-3, -2-4))$

$OPT(3,b) = \min(OPT(2,b), \min(OPT(2,e)+c_{be}, OPT(2,d)+c_{bd}))$ b-e
$= \min(0, \min(0-2, 3-1))$

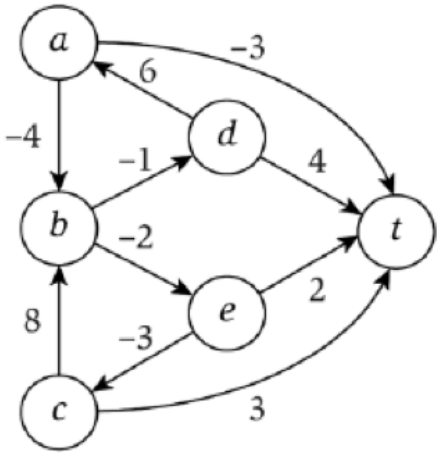
$OPT(2,e) = \min(OPT(1,e), \min(OPT(1,c)+c_{ec}, OPT(1,t)+c_{et}))$ e-c
$= \min(2, \min(3-3, 0+2))$

$OPT(1,c) = \min(OPT(0,c), \min(OPT(0,b)+c_{cb}, OPT(0,t)+c_{ct}))$ c-t
$= \min(\infty, \min(\infty+8, 0+3))$

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

a - b - e - c - t
-4 -2 -3 +3 = -6

Bellman-Ford: esempio



$OPT(5,a) = \min(OPT(4,a), \min(OPT(4,t)+c_{at}, OPT(4,b)+c_{ab}))$
$= \min(-6, \min(0-3,-2-4))$

$OPT(4,a) = \min(OPT(3,a), \min(OPT(3,t)+c_{at}, OPT(3,b)+c_{ab}))$ a-b
$= \min(-4, \min(0-3,-2-4))$

$OPT(3,b) = \min(OPT(2,b), \min(OPT(2,e)+c_{be}, OPT(2,d)+c_{bd}))$ b-e
$= \min(0, \min(0-2,3-1))$

$OPT(2,e) = \min(OPT(1,e), \min(OPT(1,c)+c_{ec}, OPT(1,t)+c_{et}))$ e-c
$= \min(2, \min(3-3,0+2))$

$OPT(1,c) = \min(OPT(0,c), \min(OPT(0,b)+c_{cb}, OPT(0,t)+c_{ct}))$ c-t
$= \min(\infty, \min(\infty+8,0+3))$

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

a - b - e - c - t
-4 -2 -3 +3 = -6

Miglioramento dell'algoritmo

- Usiamo un solo array M[v] = percorso da v a t piu` corto che abbiamo trovato fino a questo momento

$$M[v] = \min(M[v], \min_{w \in V}(c_{vw} + M[w])).$$

Miglioramento dell'algoritmo

- Teorema. Durante l'algoritmo, M[v] e` la lunghezza di un certo percorso da v a t, e dopo i aggiornamenti il valore di m[v] non e` piu` grande della lunghezza del percorso da v a t che usa al piu` i archi

- Conseguenze sullo spazio
  - Memoria: O(n).