

Problema dei cammini minimi

Mauro Passacantando

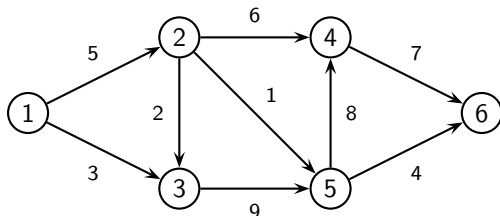
Dipartimento di Informatica, Università di Pisa
mauro.passacantando@unipi.it

Corso di Ricerca Operativa A
Laurea in Informatica - Università di Pisa - a.a. 2019/20

Problema del cammino minimo – definizione

Sia (N, A) un grafo orientato in cui è definito un costo c_{ij} per ogni arco $(i, j) \in A$. Dati un nodo origine $s \in N$ e un nodo destinazione $t \in N$, trovare un cammino orientato da s a t di costo minimo, dove il costo di un cammino è definito come la somma dei costi degli archi da cui è formato.

Esempio 1. Trovare un cammino di costo minimo dal nodo 1 al nodo 6 sul seguente grafo (sugli archi sono riportati i costi).



Problema del cammino minimo – modello

Variabili: per ogni $(i, j) \in A$, definiamo x_{ij} = numero di volte che si attraversa l'arco (i, j) [flusso sull'arco (i, j)].

Funzione obiettivo:
$$\sum_{(i,j) \in A} c_{ij} x_{ij}$$

Vincoli:

$$\sum_{(i,s) \in A} x_{is} - \sum_{(s,j) \in A} x_{sj} = -1 \quad (1 \text{ unità di flusso deve uscire da } s)$$

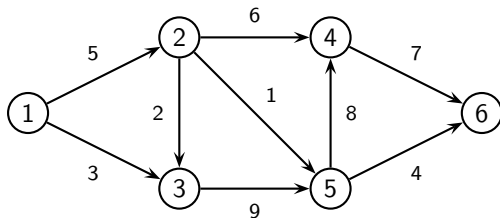
$$\sum_{(i,t) \in A} x_{it} - \sum_{(t,j) \in A} x_{tj} = 1 \quad (1 \text{ unità di flusso deve entrare in } t)$$

$$\sum_{(i,k) \in A} x_{ik} - \sum_{(k,j) \in A} x_{kj} = 0 \quad \forall k \in N \setminus \{s, t\}$$

(i nodi diversi da s e t sono nodi di transito)

Problema del cammino minimo – modello

Esempio 1. Trovare un cammino di costo minimo dal nodo 1 al nodo 6:



Modello:

$$\left\{ \begin{array}{ll} \min \sum_{(i,j) \in A} c_{ij} x_{ij} & \\ -x_{12} - x_{13} = -1 & \text{(nodo 1)} \\ x_{46} + x_{56} = 1 & \text{(nodo 6)} \\ x_{12} - x_{23} - x_{24} - x_{25} = 0 & \text{(nodo 2)} \\ x_{13} + x_{23} - x_{35} = 0 & \text{(nodo 3)} \\ x_{24} + x_{54} - x_{46} = 0 & \text{(nodo 4)} \\ x_{25} + x_{35} - x_{54} - x_{56} = 0 & \text{(nodo 5)} \\ x_{ij} \geq 0, \quad x_{ij} \in \mathbb{Z} & \end{array} \right.$$

Problema del cammino minimo – modello

Il modello può essere scritto in modo più compatto utilizzando la **matrice di incidenza** del grafo.

La matrice di incidenza E di un grafo orientato (N, A) ha dimensione $|N| \times |A|$, cioè ha una riga per ogni nodo $k \in N$ ed una colonna per ogni arco $(i, j) \in A$, ed è così definita:

$$E_{k,(i,j)} = \begin{cases} -1 & \text{se } k = i, \\ 1 & \text{se } k = j, \\ 0 & \text{altrimenti.} \end{cases}$$

Il modello può essere scritto nella forma compatta

$$\begin{cases} \min c^T x \\ Ex = b \\ x \geq 0, \quad x \in \mathbb{Z}^{|A|} \end{cases}$$

dove c è il vettore dei costi, x è il vettore dei flussi e b il vettore dei bilanci ai nodi

$$b_k = \begin{cases} -1 & \text{se } k = s, \\ 1 & \text{se } k = t, \\ 0 & \text{altrimenti.} \end{cases}$$

Problema dei cammini minimi – modello

Consideriamo ora il problema più generale di trovare un cammino di costo minimo da un nodo $r \in N$ a tutti gli altri nodi.

[Possiamo sempre supporre che esista un cammino da r ad ogni altro nodo $i \neq r$, eventualmente aggiungendo al grafo un arco fittizio (r, i) di costo

$$M > (n - 1) \max_{(i,j) \in A} c_{ij}]$$

Come deve essere modificato il modello di ottimizzazione?

Problema dei cammini minimi – modello

Consideriamo ora il problema più generale di trovare un cammino di costo minimo da un nodo $r \in N$ a tutti gli altri nodi.

[Possiamo sempre supporre che esista un cammino da r ad ogni altro nodo $i \neq r$, eventualmente aggiungendo al grafo un arco fittizio (r, i) di costo

$$M > (n - 1) \max_{(i,j) \in A} c_{ij}]$$

Come deve essere modificato il modello di ottimizzazione?

Basta modificare il vettore b dei bilanci ponendo

$$b_i = \begin{cases} -(n - 1) & \text{se } i = r, \\ 1 & \text{se } i \neq r, \end{cases}$$

e considerare il modello

$$\begin{cases} \min c^T x \\ Ex = b \\ x \geq 0, \quad x \in \mathbb{Z}^{|A|} \end{cases}$$

Esistenza di soluzioni ottime

Teorema

Il problema dei cammini minimi ammette (almeno) una soluzione ottima se e solo se non esistono cicli orientati di costo negativo.

[In seguito vedremo in modo algoritmico come fare a verificare se un grafo contiene un ciclo orientato di costo negativo (vedi algoritmo di Bellman-Ford)]

Se non esistono cicli orientati di costo negativo, allora esiste un insieme di cammini minimi da r ad ogni altro nodo che forma un **albero di copertura radicato in r e orientato**.

Condizioni di ottimalità

Come riconoscere una soluzione ottima?

Se T è un albero di copertura radicato in r e orientato, allora esiste un unico cammino da r ad ogni nodo $i \neq r$.

Indichiamo con π_i il costo di tale cammino. [π_i è chiamato potenziale del nodo i]
Per calcolare i potenziali π basta fare una visita dell'albero T e porre $\pi_j = \pi_i + c_{ij}$ se $\text{pred}(j) = i$.

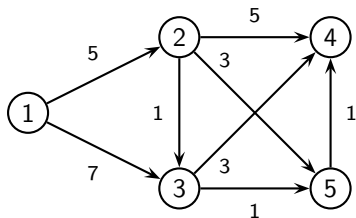
Teorema (condizioni di Bellman)

Sia T un albero di copertura radicato in r e orientato, π il corrispondente vettore dei potenziali dei nodi.

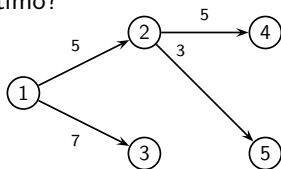
T è un albero dei cammini minimi **se e solo se** $\pi_j \leq \pi_i + c_{ij}, \quad \forall (i, j) \notin T.$

Condizioni di ottimalità

Esempio 2. Consideriamo il problema dei cammini minimi di radice 1 sul grafo:



Il seguente albero è ottimo?



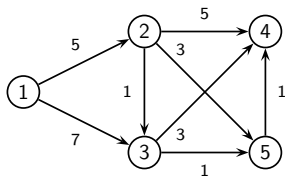
Calcoliamo i potenziali dei nodi: $\pi = (0, 5, 7, 10, 8)$

e controlliamo le condizioni di Bellman:

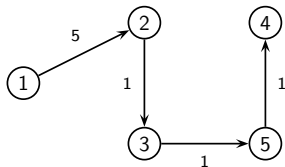
arco $(2, 3)$: $7 = \pi_3 \leq \pi_2 + c_{23} = 5 + 1$? NO, quindi l'albero non è ottimo.

Condizioni di ottimalità

Esempio 3. Consideriamo il problema dei cammini minimi di radice 1 sul grafo:



Il seguente albero è ottimo?



I potenziali dei nodi sono $\pi = (0, 5, 6, 8, 7)$ Controlliamo le condizioni di Bellman:

arco $(1, 3)$: $6 = \pi_3 \leq \pi_1 + c_{13} = 0 + 7$? SI.

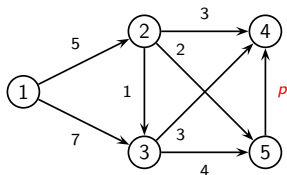
arco $(2, 4)$: $8 = \pi_4 \leq \pi_2 + c_{24} = 5 + 5$? SI.

arco $(2, 5)$: $7 = \pi_5 \leq \pi_2 + c_{25} = 5 + 3$? SI.

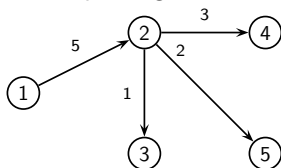
arco $(3, 4)$: $8 = \pi_4 \leq \pi_3 + c_{34} = 6 + 3$? SI. Pertanto l'albero è ottimo.

Condizioni di ottimalità

Esempio 4. Consideriamo il problema dei cammini minimi di radice 1 sul grafo:



Per quali valori del parametro p il seguente albero è ottimo?



I potenziali dei nodi sono $\pi = (0, 5, 6, 8, 7)$ Condizioni di Bellman:

arco (1, 3): $6 = \pi_3 \leq \pi_1 + c_{13} = 0 + 7$ sempre

arco (3, 4): $8 = \pi_4 \leq \pi_3 + c_{34} = 6 + 3$ sempre

arco (3, 5): $7 = \pi_5 \leq \pi_3 + c_{35} = 6 + 4$ sempre

arco (5, 4): $8 = \pi_4 \leq \pi_5 + c_{54} = 7 + p$ per $p \geq 1$.

Pertanto l'albero è ottimo se e solo se $p \geq 1$.

Unicità della soluzione ottima

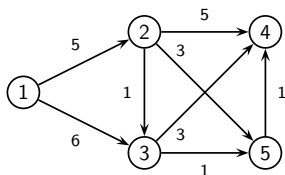
Teorema

Supponiamo che nel grafo non esistano cicli orientati di costo ≤ 0 . Sia T un albero di copertura radicato in r e orientato, π il corrispondente vettore dei potenziali dei nodi. Allora

T è l'unico albero dei cammini minimi se e solo se $\pi_j < \pi_i + c_{ij}, \quad \forall (i,j) \notin T$.

Nell'esempio 3 le condizioni di Bellman valgono tutte con il segno $<$ quindi l'albero ottimo è unico.

Mentre nel grafo



esistono due alberi ottimi:

$T_1 = \{(1,2), (2,3), (3,5), (5,4)\}$ e $T_2 = \{(1,2), (1,3), (3,5), (5,4)\}$.

Algoritmo di Dijkstra

L'algoritmo di Dijkstra trova un albero ottimo nel caso in cui $c_{ij} \geq 0$ per ogni $(i, j) \in A$.

Ad ogni iterazione l'algoritmo mantiene:

- ▶ un albero di copertura radicato in r e orientato (memorizzato in un vettore p di predecessori)
- ▶ un vettore π di potenziali dei nodi corrispondente all'albero memorizzato in p
- ▶ un insieme $U \subseteq N$ tale che gli archi uscenti da nodi di U potrebbero violare le condizioni di Bellman.

Algoritmo di Dijkstra

Algoritmo

0. Poni

$$p_i = \begin{cases} 0 & \text{se } i = r \\ -1 & \text{se } i \neq r \end{cases} \quad \pi_i = \begin{cases} 0 & \text{se } i = r \\ +\infty & \text{se } i \neq r \end{cases} \quad U = N$$

(l'albero inizialmente è costituito solo da archi fittizi da r agli altri nodi)

1. Se $U = \emptyset$ allora stop
2. Seleziona un nodo $u \in U$ con potenziale minimo: $u = \arg \min_{i \in U} \pi_i$
3. Per ogni arco $(u, v) \in A$ controlla la condizione di Bellman:
se $\pi_v > \pi_u + c_{uv}$ allora $p_v = u$, $\pi_v = \pi_u + c_{uv}$
4. $U = U \setminus \{u\}$ e torna al passo 1.

Algoritmo di Dijkstra

Teorema

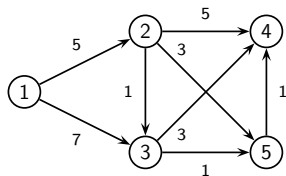
Se $c_{ij} \geq 0$ per ogni $(i, j) \in A$, allora l'algoritmo di Dijkstra trova un albero dei cammini minimi dopo $|N|$ iterazioni.

Dim. (sketch)

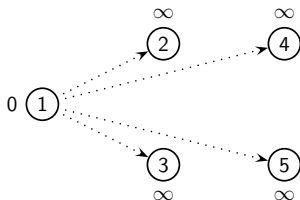
- ▶ durante l'esecuzione dell'algoritmo π è il vettore dei potenziali dei nodi associato all'albero memorizzato nel vettore p .
- ▶ il potenziale di ogni nodo non può mai aumentare durante l'esecuzione dell'algoritmo
- ▶ quando un nodo u viene estratto da U , il suo potenziale diventa definitivo e gli archi uscenti da u soddisfano le condizioni di Bellman fino alla fine dell'algoritmo.

Algoritmo di Dijkstra

Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



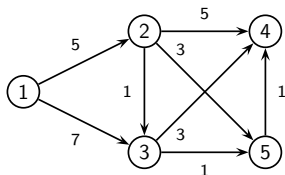
Iterazione 0.



$$U = \{1, 2, 3, 4, 5\}$$

Algoritmo di Dijkstra

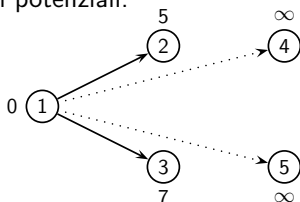
Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



Iterazione 1. Estrai nodo 1 da U ed esamina gli archi uscenti da 1:

$$\begin{cases} \infty = \pi_2 > \pi_1 + c_{12} = 5 \Rightarrow p_2 = 1, \pi_2 = 5 \\ \infty = \pi_3 > \pi_1 + c_{13} = 7 \Rightarrow p_3 = 1, \pi_3 = 7 \end{cases}$$

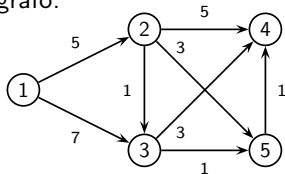
Aggiorno l'albero ed i potenziali:



$$U = \{2, 3, 4, 5\}$$

Algoritmo di Dijkstra

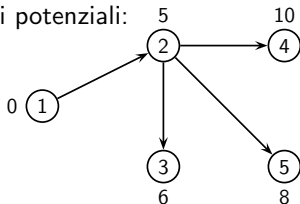
Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



Iterazione 2. Estrai nodo 2 da U ed esamina gli archi uscenti da 2:

$$\begin{cases} 7 = \pi_3 > \pi_2 + c_{23} = 6 \Rightarrow p_3 = 2, \pi_3 = 6 \\ \infty = \pi_4 > \pi_2 + c_{24} = 10 \Rightarrow p_4 = 2, \pi_4 = 10 \\ \infty = \pi_5 > \pi_2 + c_{25} = 8 \Rightarrow p_5 = 2, \pi_5 = 8 \end{cases}$$

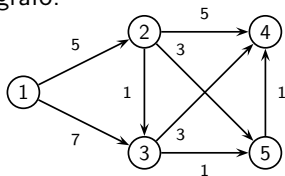
Aggiorno l'albero ed i potenziali:



$$U = \{3, 4, 5\}$$

Algoritmo di Dijkstra

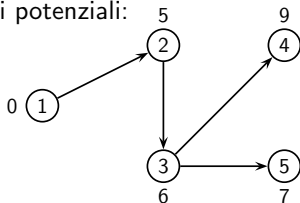
Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



Iterazione 3. Estrai nodo 3 da U ed esamina gli archi uscenti da 3:

$$\begin{cases} 10 = \pi_4 > \pi_3 + c_{34} = 9 \Rightarrow p_4 = 3, \pi_4 = 9 \\ 8 = \pi_5 > \pi_3 + c_{35} = 7 \Rightarrow p_5 = 3, \pi_5 = 7 \end{cases}$$

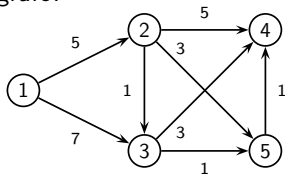
Aggiorno l'albero ed i potenziali:



$$U = \{4, 5\}$$

Algoritmo di Dijkstra

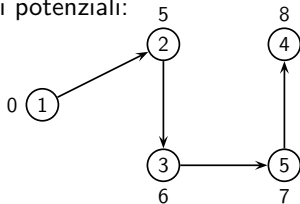
Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



Iterazione 4. Estrai nodo 5 da U ed esamina gli archi uscenti da 5:

$$9 = \pi_4 > \pi_5 + c_{54} = 8 \Rightarrow p_4 = 5, \pi_4 = 8$$

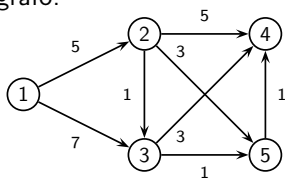
Aggiorno l'albero ed i potenziali:



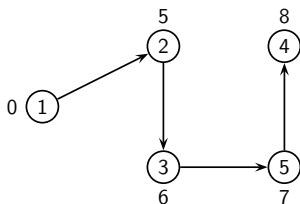
$$U = \{4\}$$

Algoritmo di Dijkstra

Esempio 5. Applicare l'algoritmo di Dijkstra per trovare l'albero dei cammini minimi di radice 1 sul grafo:



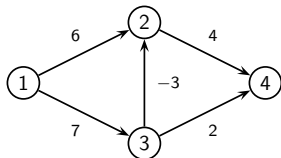
Iterazione 5. Estrai nodo 4 da U , non ci sono archi uscenti da 4, stop.
Un albero dei cammini minimi è



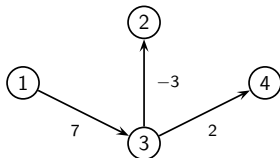
Algoritmo di Dijkstra

Se nel grafo esistono archi di costo negativo, l'algoritmo di Dijkstra non trova necessariamente un albero dei cammini minimi.

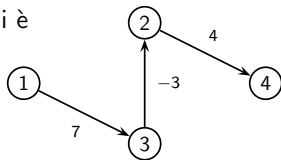
Esempio 6. Dato il grafo



l'algoritmo di Dijkstra trova l'albero



che non è ottimo, perché l'albero dei cammini minimi è



Algoritmo di Bellman-Ford

L'algoritmo di Bellman-Ford trova un albero ottimo (se esiste) anche nel caso in cui nel grafo ci siano archi di costo negativo.

Ad ogni iterazione k l'algoritmo mantiene:

- ▶ un albero di copertura radicato in r e orientato (memorizzato in un vettore p di predecessori)
- ▶ un vettore π^k di etichette associate ai nodi

Algoritmo di Bellman-Ford

Algoritmo

0. Poni

$$p_i = \begin{cases} 0 & \text{se } i = r \\ -1 & \text{se } i \neq r \end{cases} \quad \pi_i^0 = \begin{cases} 0 & \text{se } i = r \\ +\infty & \text{se } i \neq r \end{cases} \quad k = 1$$

(l'albero inizialmente è costituito solo da archi fittizi da r agli altri nodi)

1. Per ogni nodo $j \in N$:

trova $u = \arg \min_{i \in BS(j)} \{\pi_i^{k-1} + c_{ij}\}$

($BS(j) = \{i \in N : \text{esiste un arco } (i, j) \in A\}$ è la stella entrante in j)

se $\pi_j^{k-1} > \pi_u^{k-1} + c_{uj}$ allora $p_j = u$, $\pi_j^k = \pi_u^{k-1} + c_{uj}$

altrimenti $\pi_j^k = \pi_j^{k-1}$

2. Se $\pi^k = \pi^{k-1}$ allora stop (p fornisce un albero ottimo)

3. Se $k = |N|$ allora stop (p fornisce un ciclo orientato di costo negativo)

altrimenti $k = k + 1$ e torna al passo 1.

Algoritmo di Bellman-Ford

Teorema

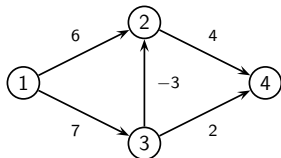
L'algoritmo di Bellman-Ford trova un albero dei cammini minimi oppure un ciclo orientato di costo negativo dopo al più $|N|$ iterazioni.

Dim. (sketch)

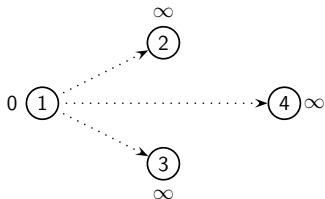
- ▶ Per ogni nodo j l'etichetta π_j^k è uguale al costo del cammino minimo da r a j che contiene al più k archi.
- ▶ Se all'iterazione $k \leq |N|$ le etichette dei nodi non sono state modificate ($\pi^k = \pi^{k-1}$), allora rimarranno costanti in tutte le iterazioni successive, quindi π_j^k è il costo del cammino minimo da r a j senza limitazioni sul numero di archi e p fornisce un albero ottimo.
- ▶ Se all'iterazione $|N|$ viene modificata l'etichetta di un nodo j , allora è stato trovato un nuovo cammino da r a j contenente $|N|$ archi (cioè passante due volte su uno stesso nodo) di costo inferiore rispetto a tutti i cammini semplici da r a j . Pertanto non può esistere un cammino minimo da r a j , mentre nel nuovo cammino trovato esiste un ciclo di costo negativo che può essere ricavato dal vettore p .

Algoritmo di Bellman-Ford

Esempio 7. Applicare l'algoritmo di Bellman-Ford per trovare l'albero dei cammini minimi di radice 1 sul grafo:

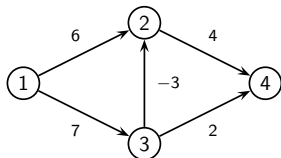


Iterazione 0.



Algoritmo di Bellman-Ford

Esempio 7. Applicare l'algoritmo di Bellman-Ford per trovare l'albero dei cammini minimi di radice 1 sul grafo:



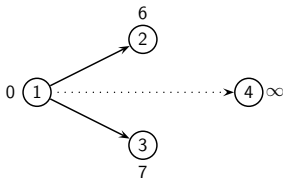
Iterazione $k = 1$.

Nodo 1: non ci sono archi entranti in 1, $\pi_1^1 = 0$.

Nodo 2: $u = \arg \min \{ \pi_1^0 + c_{12}, \pi_3^0 + c_{32} \} = 1$,
 $\infty = \pi_2^0 > \pi_1^0 + c_{12} = 6 \implies p_2 = 1, \pi_2^1 = 6$

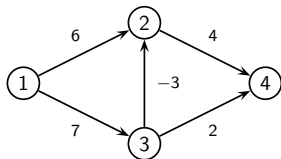
Nodo 3: $u = 1, \infty = \pi_3^0 > \pi_1^0 + c_{13} = 7 \implies p_3 = 1, \pi_3^1 = 7$

Nodo 4: $u = \arg \min \{ \pi_2^0 + c_{24}, \pi_3^0 + c_{34} \} = 2. \infty = \pi_4^0 = \pi_2^0 + c_{24} = \infty \implies \pi_4^1 = \infty$



Algoritmo di Bellman-Ford

Esempio 7. Applicare l'algoritmo di Bellman-Ford per trovare l'albero dei cammini minimi di radice 1 sul grafo:



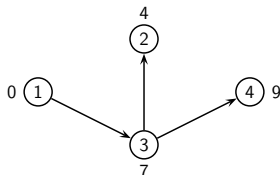
Iterazione $k = 2$.

Nodo 1: non ci sono archi entranti in 1, $\pi_1^2 = 0$

Nodo 2: $u = \arg \min\{\pi_1^1 + c_{12}, \pi_3^1 + c_{32}\} = 3$, $6 = \pi_2^1 > \pi_3^1 + c_{32} = 4 \implies p_2 = 3$,
 $\pi_2^2 = 4$

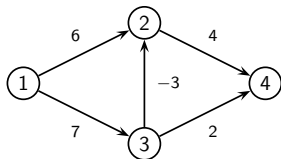
Nodo 3: $u = 1$, $7 = \pi_3^1 = \pi_1^1 + c_{13} = 7 \implies \pi_3^2 = 7$

Nodo 4: $u = \arg \min\{\pi_2^1 + c_{24}, \pi_3^1 + c_{34}\} = 3$, $\infty = \pi_4^1 > \pi_3^1 + c_{34} = 9 \implies p_4 = 3$,
 $\pi_4^2 = 9$



Algoritmo di Bellman-Ford

Esempio 7. Applicare l'algoritmo di Bellman-Ford per trovare l'albero dei cammini minimi di radice 1 sul grafo:



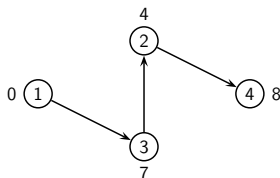
Iterazione $k = 3$.

Nodo 1: non ci sono archi entranti in 1, $\pi_1^3 = 0$

Nodo 2: $u = \arg \min\{\pi_1^2 + c_{12}, \pi_3^2 + c_{32}\} = 3$ $4 = \pi_2^2 = \pi_3^2 + c_{32} = 4 \implies \pi_2^3 = 4$

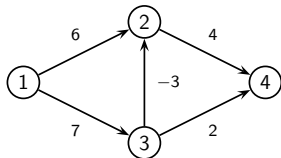
Nodo 3: $u = 1$, $7 = \pi_3^2 = \pi_1^2 + c_{13} = 7 \implies \pi_3^3 = 7$

Nodo 4: $u = \arg \min\{\pi_2^2 + c_{24}, \pi_3^2 + c_{34}\} = 2$ $9 = \pi_4^2 > \pi_2^2 + c_{24} = 8 \implies p_4 = 2$,
 $\pi_4^3 = 8$



Algoritmo di Bellman-Ford

Esempio 7. Applicare l'algoritmo di Bellman-Ford per trovare l'albero dei cammini minimi di radice 1 sul grafo:



Iterazione $k = 4$.

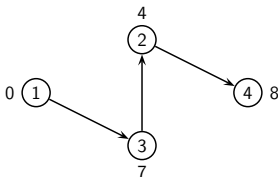
Nodo 1: non ci sono archi entranti in 1, $\pi_1^4 = 0$

Nodo 2: $u = \arg \min\{\pi_1^3 + c_{12}, \pi_3^3 + c_{32}\} = 3$ $4 = \pi_2^4 = \pi_3^3 + c_{32} = 4 \implies \pi_2^4 = 4$

Nodo 3: $u = 1$ $7 = \pi_3^4 = \pi_1^3 + c_{13} = 7 \implies \pi_3^4 = 7$

Nodo 4: $u = \arg \min\{\pi_2^3 + c_{24}, \pi_3^3 + c_{34}\} = 2$. $8 = \pi_4^4 = \pi_2^3 + c_{24} = 8 \implies \pi_4^4 = 8$

Poiché nessuna etichetta è cambiata, cioè $\pi^4 = \pi^3$, l'algoritmo si ferma e l'albero seguente è ottimo



Algoritmo di programmazione dinamica

Se il grafo (N, A) è aciclico, cioè non esistono cicli orientati, allora un albero dei cammini minimi di radice r si può trovare con un algoritmo di programmazione dinamica.

Prima di tutto è necessario fare un **ordinamento topologico** dei nodi, cioè numerare i nodi in modo che

$$(i, j) \in A \implies i < j.$$

Tale ordinamento può essere effettuato con il seguente algoritmo:

1. Assegno 1 alla radice r , elimino dal grafo r e tutti gli archi uscenti da r .
Pongo $k = 2$.
2. Assegno k ad un nodo i che non ha archi entranti.
3. Elimino il nodo i e tutti gli archi uscenti da i .
4. Se il grafo è vuoto, allora stop
altrimenti $k = k + 1$ e torna al passo 2.

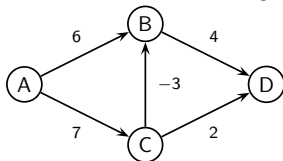
Algoritmo di programmazione dinamica

Dopo aver definito un ordinamento topologico dei nodi, l'algoritmo di programmazione dinamica per trovare un albero dei cammini minimi di radice 1 è il seguente:

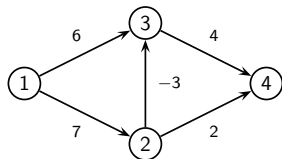
1. Poni $\pi_1 = 0$.
2. Per ogni nodo $j = 2, \dots, n$:
trova $u = \arg \min_{i < j} \{\pi_i + c_{ij}\}$
poni $p_j = u$, $\pi_j = \pi_u + c_{uj}$

Algoritmo di programmazione dinamica

Esempio 8. Applicare l'algoritmo di programmazione dinamica per trovare l'albero dei cammini minimi di radice A sul seguente grafo aciclico:



L'ordinamento topologico dei nodi è il seguente:



Algoritmo di programmazione dinamica:

Nodo 1: $\pi_1 = 0$.

Nodo 2: $u = 1, p_2 = 1, \pi_2 = 7$

Nodo 3: $u = \arg \min\{\pi_1 + c_{13}, \pi_2 + c_{23}\} = 2, p_3 = 2, \pi_3 = 4$

Nodo 4: $u = \arg \min\{\pi_2 + c_{24}, \pi_3 + c_{34}\} = 3, p_4 = 3, \pi_4 = 8$

Esercizio di riepilogo

Applicare l'algoritmo di Dijkstra, l'algoritmo di Bellman-Ford e l'algoritmo di programmazione dinamica per trovare un albero dei cammini minimi di radice 1 sul seguente grafo e confrontare le soluzioni ottenute.

