



SAPIENZA
UNIVERSITÀ DI ROMA

Ottimizzazione su reti

Volume 2 — Alberi e cammini

Alberi ricoprenti di costo minimo (Kruskal, Prim, proprietà del taglio e del ciclo) e cammini minimi (grafi aciclici, Bellman–Ford, Dijkstra), con i potenziali come certificato di ottimalità.

Fabio Furini

fabiofurini.github.io

Dipartimento di Ingegneria informatica, automatica e gestionale — Sapienza Università di Roma

Versione online e materiale scaricabile:
fabiofurini.github.io/ottimizzazione-su-reti/
github.com/fabiofurini/ottimizzazione-su-reti

Anno accademico 2026–2027

Indice

2	Alberi ricoprenti di costo minimo	2
2.1	Il problema dell'albero di copertura minimo	3
2.2	Tagli e cicli	5
2.3	La proprietà di taglio e la proprietà di ciclo	6
2.4	L'algoritmo greedy generico	9
2.5	Algoritmo di Kruskal	10
2.6	Algoritmo di Prim	13
2.7	Unicità dell'albero di copertura minimo	15
2.8	Errori tipici	17
2.9	In sintesi	17
3	Cammini minimi	19
3.1	Problemi del percorso più breve	20
3.2	Condizioni e potenziali di ottimalità	23
3.3	La formulazione della programmazione lineare	25
3.4	Percorsi più brevi nei grafici aciclici	28
3.5	L'algoritmo Bellman–Ford	31
3.6	Algoritmo di Dijkstra	35
3.7	Errori tipici	38
3.8	In sintesi	38

Alberi ricoprenti di costo minimo

Prerequisiti

- Il capitolo [«Grafici»](#): percorsi e cicli, componenti connesse, le caratterizzazioni di un albero e i cicli fondamentali.
- La notazione $O(\cdot)$ per il tempo di esecuzione di un algoritmo e l'ordinamento dei numeri m nel tempo $O(m \log m)$.

Obiettivi

Alla fine di questo capitolo sarai in grado di:

- indica il problema dell'albero ricoprente minimo e lo riconosce nelle applicazioni;
- dimostrare la proprietà di taglio e la proprietà di ciclo e utilizzarle per certificare che uno albero ricoprente è ottimale;
- esegue gli algoritmi di Kruskal e di Prim, dimostra che sono corretti e analizza il loro tempo di esecuzione;
- decide se l'albero ricoprente minimo è unico e trova gli archi che appartengono a ogni albero ottimale.

2.1 Il problema dell'albero di copertura minimo

Definizione 2.1: problema dell'albero ricoprente minimo

Sia $G = (V, E)$ un grafo connesso non orientato con vertici $n = |V|$ e archi $m = |E|$, e sia ogni arco $e \in E$ hanno un **peso** $w_e \in \mathbb{R}$. Il **peso** di una serie di lati $F \subseteq E$ è

$$w(F) = \sum_{e \in F} w_e.$$

Il **problema dell'albero ricoprente minimo** (MST) richiede uno albero ricoprente $\tilde{T} = (V, \tilde{F})$ di G di minimo peso:

$$w(\tilde{F}) \leq w(F) \quad \text{per ogni albero ricoprente } (V, F) \text{ of } G.$$

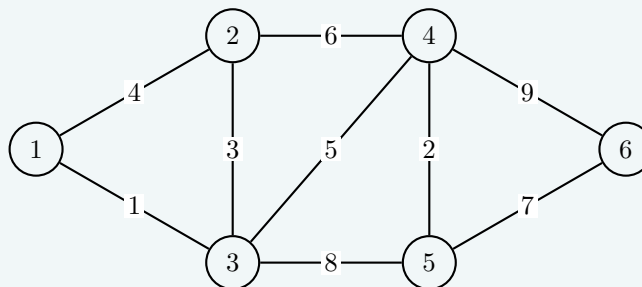
Uno albero ricoprente di peso minimo è uno **albero ricoprente minimo**. Scriviamo w_{uv} per il peso del lato $\{u, v\}$ e identifichiamo uno albero ricoprente con il suo insieme di lati.

- Un grafo connesso ha uno albero ricoprente e ne ha un numero finito: uno albero ricoprente minimo è sempre esiste. Ogni albero ricoprente ha lati $n - 1$.
- I pesi possono essere negativi. Aggiungendo la stessa costante a ogni peso si aggiunge $(n - 1)$ volte quella costante a peso di ogni albero ricoprente, quindi gli albero ricoprente minimi non cambiano.
- Uno **maximum albero ricoprente** è un albero ricoprente minimo per i pesi $-w_e$.

Esempio 2.1: il grafo di riferimento ponderato

Prendiamo il grafo non orientato sottostante del grafo di riferimento del capitolo «Graf», con vertici $n = 6$, lati $m = 9$ e i pesi

lato	$\{1, 2\}$	$\{1, 3\}$	$\{2, 3\}$	$\{2, 4\}$	$\{3, 4\}$	$\{3, 5\}$	$\{4, 5\}$	$\{4, 6\}$	$\{5, 6\}$
w_e	4	1	3	6	5	8	2	9	7

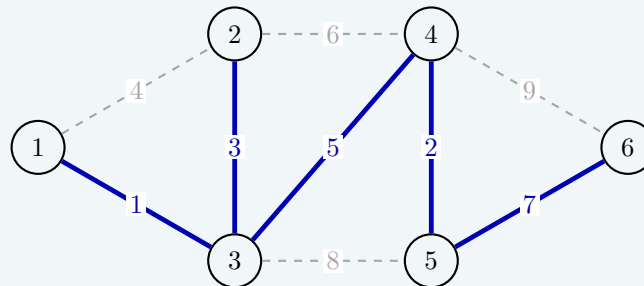


Chiameremo questo grafo **grafo di riferimento pesato**: tutti gli esempi del capitolo lo utilizzano. I pesi sono gli interi distinti $1, 2, \dots, 9$.

Esempio 2.2: l'albero di copertura minimo del grafo di riferimento

Il grafo di riferimento ponderato ha albero ricoprente 55. Enumerandoli tutti, il peso minimo è 18, raggiunto solo vicino all'albero (blu nella figura)

$$\tilde{F} = \{\{1, 3\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}\}, \quad w(\tilde{F}) = 1 + 3 + 5 + 2 + 7 = 18.$$



Lo albero ricoprente $\{1, 2\}, \{1, 3\}, \{2, 4\}, \{3, 5\}, \{4, 6\}$ trovato con la ricerca in ampiezza nel capitolo «Graphs» pesa $4 + 1 + 6 + 8 + 9 = 28$ e uno albero ricoprente massimo pesa 34.

- Enumerare gli albero ricoprente è pratico solo su grafi piccoli: il grafo di riferimento, con vertici 6 e Lati 9, dispone già di 55. Gli algoritmi di questo capitolo trovano un albero di copertura minimo in tempo polinomiale.

2.1.1 Applicazioni

- **Progettazione della rete.** I vertici sono siti (uffici, case, stabilimenti) e w_{uv} è il costo di un collegamento diretto collegamento (un cavo, un tubo, una strada); la rete più economica che collega tutti i siti è uno albero ricoprente minimo.
- **Clustering.** La rimozione dei lati più pesanti $k - 1$ di uno albero ricoprente minimo divide i vertici in k gruppi di punti vicini (clustering a collegamento singolo).
- **Sottoproblem.** Gli alberi di copertura minimi forniscono limiti inferiori per il problema del commesso viaggiatore e un punto di partenza per algoritmi euristici per la progettazione di reti.

Un albero di copertura minimo collega tutti i vertici di un grafo pesato connesso con lati $n - 1$ di totale minimo peso; sul grafo di riferimento ponderato è unico e pesa 18.

2.2 Tagli e cicli

Definizione 2.2: cut

Sia $G = (V, E)$ un grafo non orientato e $S \subseteq V$ con $\emptyset \neq S \neq V$. Il **cut** definito da S è l'insieme dei lati con esattamente un punto finale in S :

$$\delta(S) = \{ \{u, v\} \in E : u \in S, v \notin S \}.$$

Un lato di $\delta(S)$ **attraversa** il taglio. Per un singolo vertice, $\delta(\{v\}) = \delta(v)$ è l'insieme dei lati incidente a v .

- Un grafo è connesso se e solo se ogni taglio $\delta(S)$ non è vuoto: un taglio vuoto separa S da $V \setminus S$ e un percorso da $u \in S$ a $v \notin S$ deve utilizzare un lato di $\delta(S)$.
- Nel grafo di riferimento ponderato, $\delta(\{1, 2, 3\}) = \{\{2, 4\}, \{3, 4\}, \{3, 5\}\}$ e $\delta(\{6\}) = \{\{4, 6\}, \{5, 6\}\}$.

Lemma 2.3: un ciclo attraversa un taglio un numero pari di volte

Sia C un ciclo di G e $\delta(S)$ un taglio. Allora $|C \cap \delta(S)|$ è pari. In particolare, se un ciclo contiene un lato di $\delta(S)$, contiene almeno un altro lato di $\delta(S)$.

Dimostrazione. Lascia che il ciclo visiti $v_0, v_1, \dots, v_k = v_0$. Il lato $\{v_{t-1}, v_t\}$ attraversa il taglio se e solo se v_{t-1} e v_t giacciono su lati diversi. Percorrendo la ciclante il versante cambia ad ogni incrocio e solo lì; il la passeggiata termina a v_0 , sul lato da cui è iniziata, quindi il numero di cambi è pari. $\square$

Definizione 2.4: taglio fondamentale

Sia T un albero ricoprente di un grafo connesso $G = (V, E)$ e $e = \{u, v\} \in T$. Il grafo $T \setminus \{e\}$ ne ha due componenti collegati; lascia che S_e sia quello che contiene u . Il **taglio fondamentale** di e è $\delta(S_e)$.

- $T \setminus \{e\}$ ha lati $n - 2$ e nessun ciclo, quindi ha componenti $n - (n - 2) = 2$ (**Corollario «lati e componenti»**); v non è in S_e , altrimenti il percorso da u a v in T non sarebbe univoco.
- L'unico lato di T nel taglio fondamentale di e è e stesso: ogni altro lato di T si trova all'interno di uno dei le due componenti.

Lemma 2.5: scambio

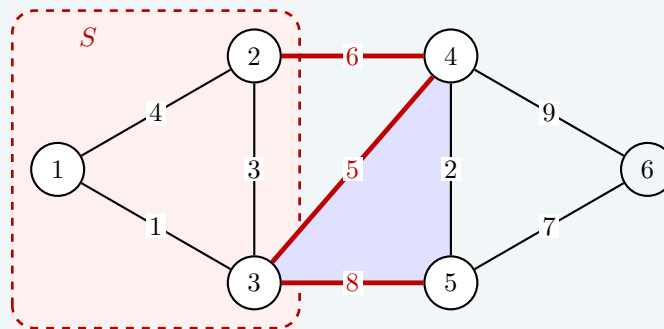
Sia T un albero ricoprente di un grafo connesso $G = (V, E)$.

1. Se $e \notin T$ e f è un arco del ciclo fondamentale $C(T, e)$, allora $T \cup \{e\} \setminus \{f\}$ è un albero di copertura.
2. Se $f \in T$ e $e \notin T$ attraversano il taglio fondamentale di f , allora $f \in C(T, e)$ e $T \cup \{e\} \setminus \{f\}$ è uno albero ricoprente.

- Dimostrazione.*
1. Il resto di $C(T, e)$ si unisce agli estremi di f , quindi rimuovendo f dal grafo connesso $T \cup \{e\}$ lascia sì è connesso. Il risultato è connesso con gli lati $n - 1$: è uno albero ricoprente tramite la [caratterizzazione di un albero](#), (b).
 2. I punti finali di e si trovano su lati diversi del taglio fondamentale $\delta(S_f)$. Il percorso tra di loro in T fa parte di $C(T, e)$ e deve attraversare $\delta(S_f)$ con un lato di T ; l'unico è f . La parte 1 si conclude. $\square$

Esempio 2.3: tagli e cicli del grafo di riferimento

In figura è mostrato il taglio $\delta(\{1, 2, 3\})$ (lati rossi) e il ciclo 3, 4, 5 (ombreggiato in blu): il ciclo utilizza i lati $\{3, 4\}$ e $\{3, 5\}$ del taglio, un numero pari.



Per l'albero di copertura minimo $\tilde{F}$ di [Esempio 2.2](#), il taglio fondamentale di $\{3, 4\}$ è $\delta(\{1, 2, 3\})$ e il ciclo fondamentale di $\{3, 5\}$ è $C(\tilde{F}, \{3, 5\}) = 3, 4, 5, 3$.

Un ciclo attraversa ogni taglio un numero pari di volte. Aggiunta di un lato a uno albero ricoprente e rimozione di qualsiasi lato del ciclo si chiude dà un altro albero ricoprente.

2.3 La proprietà di taglio e la proprietà di ciclo

Teorema 2.6: proprietà taglio

Sia $G = (V, E)$ un grafo connesso con pesi w_e , e sia $F \subseteq E$ contenuto in qualche minimo albero di copertura. Sia $\delta(S)$ un taglio con $F \cap \delta(S) = \emptyset$ e sia e un lato di peso minimo in $\delta(S)$.

- (a) $F \cup \{e\}$ è contenuto in un albero ricoprente minimo.
- (b) Se $w_e < w_f$ per ogni $f \in \delta(S) \setminus \{e\}$, allora e appartiene a ogni albero ricoprente minimo.

Dimostrazione. Sia T^* un albero ricoprente minimo con $F \subseteq T^*$; per (b), sia T^* qualsiasi albero di copertura minimo e $F = \emptyset$. Se $e \in T^*$ non c'è nulla da dimostrare; lascia che $e \notin T^*$.

1. Il ciclo fondamentale $C(T^*, e)$ contiene il lato e di $\delta(S)$, quindi contiene un altro lato $f \in \delta(S)$, $f \neq e$ (Lemma 2.3). Tutti i lati diversi da e sono presenti in T^* , quindi $f \in T^*$; e $f \notin F$, da $F \cap \delta(S) = \emptyset$.
2. Di Lemma 2.5, $T' = T^* \cup \{e\} \setminus \{f\}$ è uno albero ricoprente, con $w(T') = w(T^*) + w_e - w_f$.
3. (a) e ha un peso minimo in $\delta(S)$, quindi $w_e \leq w_f$ e $w(T') \leq w(T^*)$: T' è un peso minimo albero ricoprente contenente $F \cup \{e\}$.
4. (b) Qui $w_e < w_f$ e $w(T') < w(T^*)$, contro l'ottimalità di T^* : quindi $e \in T^*$. $\square$

Teorema 2.7: proprietà ciclo

Sia $G = (V, E)$ un grafo connesso con pesi w_e , sia C un ciclo di G e e un arco di peso massimo in C .

- (a) Alcuni albero ricoprente minimi non contengono e .
- (b) Se $w_e > w_f$ per ogni $f \in C \setminus \{e\}$, allora nessun albero ricoprente minimo contiene e .

Dimostrazione. Sia T^* un albero di copertura minimo con $e \in T^*$ (se non ce n'è, valgono entrambe le affermazioni) e sia $\delta(S_e)$ è il taglio fondamentale di e in T^* .

1. Il ciclo C contiene il lato e di $\delta(S_e)$, quindi contiene un altro lato $f \in \delta(S_e)$ (Lemma 2.3); $f \notin T^*$, poiché e è l'unico lato di T^* nella sua taglio fondamentale.
2. Da Lemma 2.5(2), $T' = T^* \cup \{f\} \setminus \{e\}$ è una copertura albero, con $w(T') = w(T^*) + w_f - w_e$.
3. (a) $w_f \leq w_e$, quindi T' è un albero ricoprente minimo senza e . (b) $w_f < w_e$ darebbe $w(T') < w(T^*)$, una contraddizione: nessun albero ricoprente minimo contiene e . $\square$

Esempio 2.4: le due proprietà sul grafo di riferimento

- *Proprietà di taglio.* Il lato $\{3, 4\}$, di peso 5, è l'unico lato più leggero di $\delta(\{1, 2, 3\}) = \{\{2, 4\}, \{3, 4\}, \{3, 5\}\}$, con pesi 6, 5, 8: appartiene ad ogni albero di copertura minimo. Allo stesso modo, $\{5, 6\}$ è l'esclusivo lato più leggero di $\delta(\{6\})$.
- *Proprietà del ciclo.* Il lato $\{1, 2\}$ è l'unico lato più pesante del ciclo 1, 2, 3 (pesi 4, 3, 1), e $\{3, 5\}$ è l'unico lato più pesante del ciclo 3, 4, 5 (pesi 5, 2, 8): non contiene alcun albero di copertura minimo loro. I cicli 2, 3, 4 e 4, 5, 6 escludono $\{2, 4\}$ e $\{4, 6\}$.
- I quattro lati esclusi lasciano esattamente i lati 5 di $\tilde{F}$: le sole due proprietà lo dimostrano $\tilde{F}$ è l'unico albero di copertura minimo.

2.3.1 Condizioni di ottimalità

Teorema 2.8: condizioni di ottimalità

Sia T un albero ricoprente di un grafo connesso $G = (V, E)$ con pesi w_e . Le seguenti dichiarazioni sono equivalenti:

- (i) T è uno albero ricoprente minimo;
- (ii) **condizione ciclo**: per ogni $e \notin T$, $w_e \geq w_f$ per ogni lato f di $C(T, e)$;
- (iii) **condizione di taglio**: per ogni $e \in T$, $w_e \leq w_f$ per ogni lato f del taglio fondamentale di e .

Dimostrazione. Dimostriamo (i) $\Rightarrow$ (iii) $\Rightarrow$ (ii) $\Rightarrow$ (i).

- (i) $\Rightarrow$ (iii). Se f incrocia il taglio fondamentale di $e \in T$ con $w_f < w_e$, allora $T \cup \{f\} \setminus \{e\}$ è uno albero ricoprente (Lemma 2.5(2)) più leggero di T .
- (iii) $\Rightarrow$ (ii). Lascia che $e \notin T$ e $f \in C(T, e)$, $f \neq e$. Il percorso di T tra i gli estremo di e utilizzano f ed è unico, quindi la rimozione di f li separa: e attraversa il taglio fondamentale di f , e $w_f \leq w_e$ da (iii).

(ii) $\Rightarrow$ (i). Tra gli albero ricoprente minimi, sia T^* ad avere il maggior numero di archi in comune con T e presuppone $T^* \neq T$.

1. Prendi $e \in T^* \setminus T$ e lascia che $\delta(S_e)$ sia il suo taglio fondamentale in T^* . Il ciclo $C(T, e)$ contiene e , quindi contiene un altro lato $f \in \delta(S_e)$ (Lemma 2.3), con $f \in T$ e $f \notin T^*$.
2. Di Lemma 2.5(2), $T' = T^* \cup \{f\} \setminus \{e\}$ è una copertura albero e $w_f \leq w_e$ mediante (ii) applicato a $e \notin T$. Quindi $w(T') \leq w(T^*)$.
3. T' è un albero ricoprente minimo con un lato in più in comune con T : una contraddizione. Quindi $T = T^*$. $\square$

Esempio 2.5: certificare l'albero ottimale

Per l'albero $\tilde{F}$ di Esempio 2.2, i quattro lati esterni all'albero e la loro fondamentale i cicli sono:

lato $e \notin \tilde{F}$	$C(\tilde{F}, e)$	pesi dei lati dell'albero del ciclo
$\{1, 2\}, w = 4$	1, 3, 2, 1	1, 3
$\{2, 4\}, w = 6$	2, 3, 4, 2	3, 5
$\{3, 5\}, w = 8$	3, 4, 5, 3	5, 2
$\{4, 6\}, w = 9$	4, 5, 6, 4	2, 7

Ogni lato esterno all'albero è pesante almeno quanto i lati dell'albero del suo ciclo: di Teorema 2.8 l'albero è ottimale, senza enumerare gli albero ricoprente 55.

Esempio 2.6: migliorare un albero che non è ottimale

L'albero BFS $T = \{\{1, 2\}, \{1, 3\}, \{2, 4\}, \{3, 5\}, \{4, 6\}\}$, di peso 28, viola la condizione del ciclo. Per $e = \{4, 5\}$, di peso 2, il ciclo fondamentale $4, 2, 1, 3, 5, 4$ contiene $\{2, 4\}$, $\{1, 2\}$ e $\{3, 5\}$, di pesi 6, 4, 8.

- Rimuovendo il più pesante, $\{3, 5\}$, e aggiungendo $\{4, 5\}$ si ottiene un albero portante del peso $28 + 2 - 8 = 22$.
- La dimostrazione di [Teorema 2.8](#) si trasforma in un algoritmo: ripetere lo scambio mentre un lato viola la condizione. Gli algoritmi delle sezioni successive sono più veloci.

Il lato più leggero di un taglio appartiene a un albero di copertura minimo e il lato più pesante di un ciclo può essere lasciato fuori. Un albero di copertura è ottimale se e solo se ogni arco esterno ad esso è il più pesante del suo ciclo fondamentale.

2.4 L'algoritmo greedy generico

- La proprietà cut suggerisce di costruire uno albero ricoprente minimo un lato alla volta: l'insieme corrente F è sempre contenuto in un albero di copertura minimo e il lato successivo è il lato più leggero di un taglio non attraversato da F .

Algoritmo

Algorithm: Algoritmo greedy generico per MST

Input: un grafo connesso non orientato $G = (V, E)$ con pesi w_e

Output: un albero di copertura minimo F

- 1 **Passaggio 0 (inizializzazione).** Imposta $F := \emptyset$.
- 2 **Passaggio 1 (test di arresto).** Se $|F| = n - 1$, STOP.
- 3 **Passaggio 2 (aumento).** Scegli un taglio $\delta(S)$ con $F \cap \delta(S) = \emptyset$ e aggiungi a F un lato di peso minimo in $\delta(S)$. Vai al passaggio 1.

Teorema 2.9: correttezza dell'algoritmo greedy generico

Ad ogni iterazione dell'algoritmo greedy generico esiste un taglio come nel passaggio 2 che non è vuoto e F è contenuto in un albero di copertura minimo. Alla fine, F è un albero ricoprente minimo.

Dimostrazione. Per induzione sulle iterazioni; $F = \emptyset$ è contenuto in ogni albero ricoprente minimo.

1. Se $|F| < n - 1$, il grafo (V, F) è una foresta (un sottoinsieme di un albero) con componenti $n - |F| \geq 2$. Lascia che S sia uno di essi: nessun lato di F attraversa $\delta(S)$ e $\delta(S) \neq \emptyset$ perché G è connesso.
2. Per la proprietà di taglio ([Teorema 2.6\(a\)](#)), il nuovo insieme $F \cup \{e\}$ è ancora

contenuto in un albero di copertura minimo.

3. Quando $|F| = n - 1$, F è contenuto in uno albero ricoprente minimo T^* con lati $n - 1$, quindi $F = T^*$. $\square$

- Il lato aggiunto $e = \{u, v\}$ ha $u \in S$ e $v \notin S$ e nessun lato di F incrocia $\delta(S)$: u e v sono in diversi componenti di (V, F) , quindi e non chiude un ciclo.
- Gli algoritmi di Kruskal e di Prim sono due modi per scegliere il taglio nello Step 2.

Aggiungendo, $n - 1$ volte, il lato più leggero di un taglio non attraversato dai lati correnti costruisce un albero ricoprente minimo.

2.5 Algoritmo di Kruskal

Algoritmo

Algorithm: Algoritmo di Kruskal

Input: un grafo connesso non orientato $G = (V, E)$ con pesi w_e

Output: un albero di copertura minimo F

- 1 **Passaggio 0 (inizializzazione).** Ordina i lati in base al peso non decrescente, $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$, and set $F := \emptyset$.
- 2 **Passaggio 1 (scansione).** Per $k = 1, 2, \dots, m$: se i punti finali di e_k si trovano in componenti diversi di (V, F) , impostare $F := F \cup \{e_k\}$; altrimenti rifiuta e_k .
- 3 Se $|F| = n - 1$, STOP.

- Un lato è **accettato** se unisce due diversi componenti di (V, F) e **rifiutato** se lo farebbe chiudere un ciclo. I componenti di (V, F) iniziano come singoli vertici n e si fondono ad ogni accettazione.
- In questo modulo, i lati di uguale peso vengono scansionati in ordine lessicografico di $\{u, v\}$, con $u < v$.

Teorema 2.10: correttezza dell'algoritmo di Kruskal

L'algoritmo di Kruskal si ferma con $|F| = n - 1$ e F è uno albero ricoprente minimo.

Dimostrazione. Mostriamo che ogni lato accettato $e_k = \{u, v\}$ è una scelta del passo 2 dell'algoritmo greedy generico. Lasciamo S è il componente di (V, F) che contiene u , subito prima che e_k venga accettato: $v \notin S$ e nessun lato di F incrocia $\delta(S)$.

1. Lascia che $e_h \in \delta(S)$ con $h < k$. Quando e_h è stato scansionato non è stato accettato, poiché $F \cap \delta(S) = \emptyset$; quindi è allora gli estremo erano nello stesso componente e lo sono ancora, perché i componenti si uniscono solo. Ciò contraddice $e_h \in \delta(S)$.

2. Quindi ogni lato di $\delta(S)$ è un certo e_h con $h \geq k$ e $w_{e_h} \geq w_{e_k}$: e_k ha un minimo peso in $\delta(S)$.
3. Se la scansione terminasse con $|F| < n - 1$, la foresta (V, F) avrebbe due componenti uniti da un lato di G e quel lato sarebbe stato accettato una volta scansionato, secondo l'argomentazione del punto 1.

Secondo [Teorema 2.9](#), F è uno albero ricoprente minimo. □

2.5.1 Implementazione e tempo di esecuzione

- I componenti di (V, F) sono mantenuti in una struttura **union-find**: ogni vertice v memorizza l'etichetta $comp(v)$ del suo componente e ogni componente memorizza l'elenco dei suoi vertici e la sua dimensione.
- Testare se u e v si trovano in componenti diversi è il confronto $comp(u) \neq comp(v)$, in $O(1)$ tempo. L'unione di due componenti rietichetta i vertici di *più piccolo* (**unione per dimensione**).

Teorema 2.11: tempo di esecuzione dell'algoritmo di Kruskal

Con la struttura unione-trova per etichette e unione per dimensione, l'algoritmo di Kruskal viene eseguito nel tempo $O(m \log n)$.

Dimostrazione. 1. L'ordinamento dei lati richiede tempo $O(m \log m)$; da $m \leq n^2$, $\log m \leq 2 \log n$ e questo è $O(m \log n)$.

2. La scansione esegue i test m , ciascuno nel tempo $O(1)$.

3. Quando un vertice cambia rappresentante, il suo componente si fonde con un componente di dimensione almeno pari. La dimensione del componente almeno raddoppia e non supera mai n ; quindi ogni vertice cambia rappresentante al massimo $\log_2 n$ volte. Tutte le fusioni richiedono complessivamente tempo $O(n \log n)$.

G è connesso, quindi $n - 1 \leq m$ e il totale è $O(m \log n + n \log n) = O(m \log n)$. □

- Il tempo di esecuzione è dominato dall'ordinamento. Con l'unione: trova la struttura per alberi, con l'unione per rango e compressione del percorso, la scansione richiede un tempo quasi lineare; il limite rimane $O(m \log n)$ a causa dell'ordinamento.
- L'algoritmo utilizza solo confronti tra pesi: ogni trasformazione crescente dei pesi dà il stesso albero.

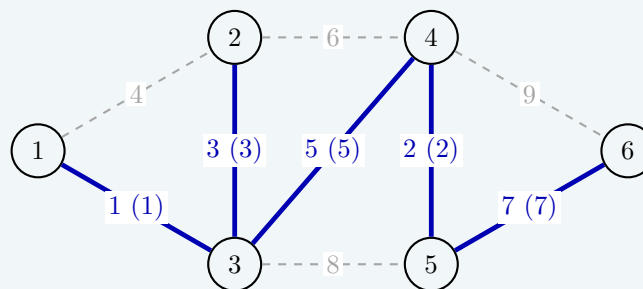
2.5.2 Algoritmo di Kruskal sul grafo di riferimento

Esempio 2.7: Algoritmo di Kruskal

Eseguiamo l'algoritmo di Kruskal sul grafo di riferimento ponderato. I lati in ordine di peso sono $\{1, 3\}, \{4, 5\}, \{2, 3\}, \{1, 2\}, \{3, 4\}, \{2, 4\}, \{5, 6\}, \{3, 5\}, \{4, 6\}$. Ogni riga rappresenta un lato scansionato, con i componenti di (V, F) dopo il passaggio:

k	lato e_k	w_{e_k}	decisione e componenti di (V, F)
1	$\{1, 3\}$	1	accetta: $\{1, 3\}, \{2\}, \{4\}, \{5\}, \{6\}$
2	$\{4, 5\}$	2	accetta: $\{1, 3\}, \{2\}, \{4, 5\}, \{6\}$
3	$\{2, 3\}$	3	accetta: $\{1, 2, 3\}, \{4, 5\}, \{6\}$
4	$\{1, 2\}$	4	rifiuta (1 e 2 sono entrambi in $\{1, 2, 3\}$)
5	$\{3, 4\}$	5	accetta: $\{1, 2, 3, 4, 5\}, \{6\}$
6	$\{2, 4\}$	6	rifiuta (2 e 4 sono entrambi in $\{1, 2, 3, 4, 5\}$)
7 e $\{5, 6\}$ e 7 e accetta: $\{1, 2, 3, 4, 5, 6\}$; $ F = 5$, STOP			

L'algoritmo si ferma dopo 7 dei lati 9, con $F = \{\{1, 3\}, \{4, 5\}, \{2, 3\}, \{3, 4\}, \{5, 6\}\}$ di peso 18: l'albero $\tilde{F}$ di [Esempio 2.2](#). La figura mostra, su ciascun lato dell'albero, il suo peso e il peso tra parentesi il passo k della sua accettazione.



- I lati scartati $\{1, 2\}$ e $\{2, 4\}$ sono i lati più pesanti dei cicli $1, 2, 3$ e $2, 3, 4$, poiché la proprietà del ciclo prevede; i lati $\{3, 5\}$ e $\{4, 6\}$ non vengono mai scansionati.

L'algoritmo di Kruskal scansiona i lati con peso non decrescente e accetta quelli che uniscono due componenti; con union: trova che viene eseguito nel tempo $O(m \log n)$, dominato dall'ordinamento.

2.6 Algoritmo di Prim

- L'algoritmo di Prim fa crescere un singolo albero a partire da un vertice iniziale s : l'insieme S dei vertici dell'albero definisce il taglio $\delta(S)$ del passaggio 2 e il lato più leggero del taglio viene trovato attraverso un **label** $d(v)$ per ogni vertice $v \notin S$.

Algoritmo

Algorithm: Algoritmo di Prim

Input: un grafo connesso non orientato $G = (V, E)$ con pesi w_e , dati dalle sue liste di adiacenza; un vertice iniziale $s \in V$

Output: un albero ricoprente minimo F

- 1 **Passaggio 0 (inizializzazione).** Imposta $S := \{s\}$ e $F := \emptyset$. Per ogni $v \neq s$: se $\{s, v\} \in E$, impostare $d(v) := w_{sv}$ e $pred(v) := s$; altrimenti impostare $d(v) := +\infty$.
- 2 **Passaggio 1 (selezione).** Se $S = V$, STOP. Altrimenti selezionare $v \notin S$ con minimo $d(v)$, e impostare $S := S \cup \{v\}$, $F := F \cup \{\{pred(v), v\}\}$.
- 3 **Passaggio 2 (aggiornamento).** Per ogni $u \notin S$ adiacente a v con $w_{vu} < d(u)$, impostare $d(u) := w_{vu}$ e $pred(u) := v$. Vai al passaggio 1.

- In questo modulo il vertice iniziale è $s = 1$ e le legature di valore nel passaggio 1 sono interrotte dall'indice più piccolo.

Lemma 2.12: significato delle etichette

Ad ogni esecuzione del passo 1 dell'algoritmo di Prim, per ogni $v \notin S$,

$$d(v) = \min \{ w_{uv} : u \in S, \{u, v\} \in E \}$$

($+\infty$ se v non ha vicini in S) e quando $d(v) < +\infty$ il minimo viene raggiunto da $u = pred(v)$.

Dimostrazione. Per induzione. Alla prima esecuzione $S = \{s\}$, e il passo 0 imposta esattamente $d(v) = w_{sv}$, oppure $+\infty$. Assumere l'affermazione prima dell'inserimento di un vertice v in S . Per $u \notin S \cup \{v\}$, il nuovo set $S \cup \{v\}$ aggiunge il candidato w_{vu} quando $\{v, u\} \in E$ e il passo 2 sostituisce $d(u)$ e $pred(u)$ esattamente quando questo candidato è più piccolo. Le altre etichette e predecessori rimangono invariate. $\square$

Teorema 2.13: correttezza dell'algoritmo di Prim

L'algoritmo di Prim si ferma con $|F| = n - 1$ e F è uno albero ricoprente minimo.

Dimostrazione. Tutti i lati di F hanno entrambi i punti finali in S , quindi $F \cap \delta(S) = \emptyset$.

1. Per Lemma 2.12, $\min_{v \notin S} d(v)$ è il peso minimo di un lato di $\delta(S)$, ottenuto da $\{pred(v), v\}$ per il v selezionato; è finito perché G è connesso.
2. Quindi ogni iterazione è una scelta del Passo 2 dell'algoritmo greedy generico,

con il taglio $\delta(S)$.

3. Ogni iterazione aggiunge un vertice a S e un lato a F , quindi l'algoritmo si interrompe dopo le iterazioni $n - 1$ con $|F| = n - 1$; [Teorema 2.9](#) conclude. $\square$

Teorema 2.14: tempo di esecuzione dell'algoritmo di Prim

Con le etichette memorizzate in un array, l'algoritmo di Prim viene eseguito in tempo $O(n^2)$.

Dimostrazione. Sono disponibili selezioni $n - 1$ e ciascuna scansione le etichette al massimo dei vertici n : $O(n^2)$ in totale. L'aggiornamento dopo l'inserimento di v esegue la scansione dell'elenco di adiacenze di v nel tempo $O(\deg(v))$ e $\sum_{v \in V} \deg(v) = 2m$. Il totale è $O(n^2 + m) = O(n^2)$, poiché $m \leq n^2$. $\square$

- Sui grafi densi, con m dell'ordine di n^2 , il limite $O(n^2)$ è lineare nella dimensione del grafo.
- Con le etichette in un heap binario, ogni selezione e ogni aggiornamento impiegano tempo $O(\log n)$, per $O(m \log n)$ in totale: migliore su grafi sparsi.

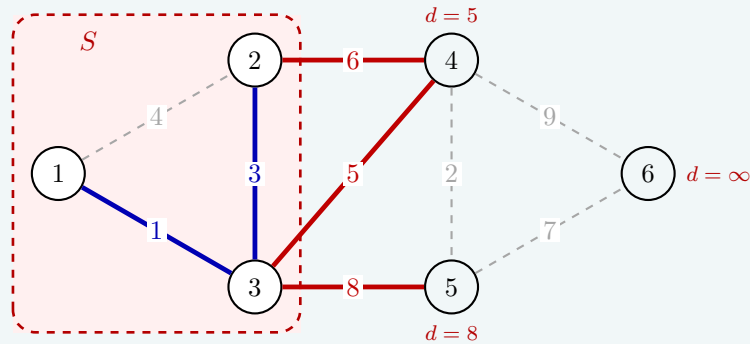
2.6.1 Algoritmo di Prim sul grafo di riferimento

Esempio 2.8: Algoritmo di Prim dal vertice 1

Eseguiamo l'algoritmo di Prim sul grafo di riferimento ponderato di $s = 1$. Ogni riga mostra il vertice selezionato, il lato aggiunto a F e alle etichette $d(v)$ dei vertici esterni a S dopo l'aggiornamento, con $\text{pred}(v)$ tra parentesi; un trattino segna i vertici già in S .

iterazione	lato selezionato e aggiunto	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$d(6)$
0	1	–	4 (1)	1 (1)	∞	∞
1	3	{1, 3}	3 (3)	–	5 (3)	8 (3)
2	2	{2, 3}	–	–	5 (3)	8 (3)
3	4	{3, 4}	–	–	–	2 (4)
4	5	{4, 5}	–	–	–	7 (5)
5	6	{5, 6}	–	–	–	–

All'iterazione 1 l'etichetta di 2 scende da 4 a 3, perché $w_{32} = 3 < 4$; all'iterazione 3 l'etichetta di 5 scende da 8 a 2. La figura mostra lo stato dopo l'iterazione 2: il taglio $\delta(\{1, 2, 3\})$ è attraversato da $\{2, 4\}$, $\{3, 4\}$, $\{3, 5\}$ e il lato successivo è $\{3, 4\}$, con $d(4) = 5$.



L'albero finale è $\{1, 3\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}$, di peso 18: lo stesso albero dell'algoritmo di Kruskal, con i lati aggiunti in un ordine diverso.

- L'algoritmo di Kruskal aggiunge $\{4, 5\}$ per secondo, mentre l'algoritmo di Prim lo aggiunge per quarto: Kruskal fa crescere una foresta, Prim un singolo albero contenente s .

L'algoritmo di Prim fa crescere un albero da s e aggiunge il lato più leggero lasciandolo; le etichette $d(v)$ danno quel vantaggio Tempo $O(n)$, per $O(n^2)$ in totale.

2.7 Unicità dell'albero di copertura minimo

Teorema 2.15: pesi distinti

Se i pesi w_e , $e \in E$, sono distinti a coppie, il grafo connesso $G = (V, E)$ ha esattamente uno spanning minimo albero.

Dimostrazione. Supponiamo che $T_1 \neq T_2$ siano alberi di copertura minimi. Hanno lo stesso numero di lati, quindi la differenza simmetrica $(T_1 \setminus T_2) \cup (T_2 \setminus T_1)$ non è vuoto; lascia che e sia il suo lato più leggero e di $e \in T_1 \setminus T_2$.

1. Il ciclo $C(T_2, e)$ non è contenuto nell'aciclico T_1 , quindi ha un fronte $f \notin T_1$; $f \neq e$ e $f \in T_2$, quindi $f \in T_2 \setminus T_1$.
2. f è nella differenza simmetrica e $f \neq e$, quindi $w_f > w_e$, essendo i pesi distinti.
3. $T_2 \cup \{e\} \setminus \{f\}$ è uno albero ricoprente (Lemma 2.5) di peso $w(T_2) + w_e - w_f < w(T_2)$: una contraddizione. $\square$

- Con pesi distinti, algoritmo di Kruskal, algoritmo di Prim da qualsiasi vertice iniziale e ogni esecuzione del l'algoritmo greedy generico restituisce lo stesso albero.
- Sul grafo di riferimento, l'algoritmo di Prim da $s = 6$ aggiunge $\{5, 6\}, \{4, 5\}, \{3, 4\}, \{1, 3\}, \{2, 3\}$: ancora una volta il albero $\tilde{F}$.

2.7.1 Pesì con valori di parità

Proposizione 2.16: edge in ogni o nessun albero ricoprente minimo

Sia T uno albero ricoprente minimo di $G = (V, E)$.

- (a) Un lato $e \in T$ appartiene ad ogni albero di copertura minimo se e solo se $w_e < w_f$ per ogni $f \neq e$ in il taglio fondamentale di e .
- (b) Un lato $e \notin T$ non appartiene a nessun albero di copertura minimo se e solo se $w_e > w_f$ per ogni $f \neq e$ in $C(T, e)$.

In particolare, T è l'unico albero di copertura minimo se e solo se ogni lato esterno a T è strettamente più pesante di gli altri lati del suo ciclo fondamentale.

Dimostrazione. (a) Se le disuguaglianze valgono, e si trova in ogni albero di copertura minimo secondo la proprietà cut ([Teorema 2.6\(b\)](#)). Altrimenti alcuni $f \neq e$ nel taglio fondamentale hanno $w_f \leq w_e$ e $w_f \geq w_e$ dalla condizione di taglio ([Teorema 2.8](#)): quindi $w_f = w_e$ e $T \cup \{f\} \setminus \{e\}$ è un albero ricoprente ([Lemma 2.5\(2\)](#)) dello stesso peso, senza e .

- (b) Se le disuguaglianze valgono, e non si trova in alcun albero di copertura minimo secondo la proprietà cycle ([Teorema 2.7\(b\)](#)). Altrimenti alcuni $f \neq e$ in $C(T, e)$ hanno $w_f = w_e$, in base al ciclo condizione e $T \cup \{e\} \setminus \{f\}$ è uno albero ricoprente minimo contenente e .

Per l'ultima affermazione: T è unico se e solo se nessun arco esterno a T appartiene ad un albero di copertura minimo, cioè (b) vale per ogni $e \notin T$. $\square$

Esempio 2.9: legami sul grafo di riferimento

- Lascia che $w_{12} = 3$ invece di 4, in modo che $w_{12} = w_{23} = 3$. Il ciclo fondamentale di $\{1, 2\}$ in $\tilde{F}$ è $1, 3, 2, 1$, con $w_{23} = w_{12}$: di [Proposizione 2.16\(b\)](#), l'albero Anche $\{1, 2\}, \{1, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}$ è ottimale, con peso 18.
- L'enumerazione conferma che questi sono gli unici due albero ricoprente minimi. Algoritmo di Kruskal, scansione $\{1, 2\}$ prima di $\{2, 3\}$, restituisce quello con $\{1, 2\}$.
- Non sempre i legami creano più alberi ottimali: con $w_{35} = w_{46} = 9$ l'albero $\tilde{F}$ è ancora l'unico albero di copertura minimo, perché ciascuno dei quattro lati esterni ad esso rimane rigorosamente il più pesante del suo ciclo.

Con pesi distinti l'albero di copertura minimo è unico. Con i legami, un albero è l'unico ottimo se e solo se ogni lato esterno ad esso è strettamente il più pesante del suo ciclo fondamentale.

2.8 Errori tipici

Attenzione

- **Utilizzo della proprietà di taglio con un taglio attraversato da F .** Il lato più leggero di $\delta(S)$ è sicuro solo se non il lato già scelto incrocia $\delta(S)$.
- **Accettare un lato nell'algoritmo di Kruskal perché tocca un nuovo vertice.** Il test è se i due estremi si trovano in componenti diversi: $\{3, 4\}$ è accettato sebbene sia 3 che 4 siano già coperti.
- **Aggiornamento delle etichette nell'algoritmo di Prim con le lunghezze dei percorsi.** La nuova etichetta è il peso w_{vu} di un lato singolo, non $d(v) + w_{vu}$: questo è l'algoritmo di Dijkstra per i percorsi più brevi.
- **Dimenticando che le etichette possono solo diminuire.** Nell'algoritmo di Prim $d(u)$ cambia solo quando $w_{vu} < d(u)$.
- **Rivendicare l'unicità dall'algoritmo.** Kruskal e Prim restituiscono un albero; con cravatte altre ottimali gli alberi possono esistere, e si trovano attraverso le uguaglianze nei cicli fondamentali.
- **Applicazione degli algoritmi ai grafi orientati.** Uno albero ricoprente minimo è definito sui grafi non orientati; in un grafo orientato l'orientamento degli archi viene ignorato.

2.9 In sintesi

In sintesi

- Uno albero ricoprente minimo è uno albero ricoprente di peso totale minimo di un grafo pesato connesso; ha $n - 1$ lati.
- Proprietà taglio: il lato più leggero di un taglio non attraversato da F estende F entro un certo albero ricoprente minimo. Proprietà del ciclo: il lato più pesante di un ciclo può essere tralasciato e, se è strettamente il più pesante, non è ottimale.
- Uno albero ricoprente è ottimale se e solo se ogni arco esterno ad esso è il più pesante del suo ciclo fondamentale, oppure equivalentemente ogni lato dell'albero è il più leggero del suo taglio fondamentale.
- L'algoritmo di Kruskal ordina gli lati e accetta quelli che uniscono due componenti, nel tempo $O(m \log n)$; Quello di Prim L'algoritmo fa crescere un albero con le etichette $d(v)$, nel tempo $O(n^2)$.
- Con pesi distinti l'albero di copertura minimo è unico.

Il grafo di riferimento ponderato in sintesi.

vertici, lati, albero ricoprente e $n = 6$, $m = 9$, 55

albero di copertura minimo

$\{1, 3\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}$, peso 18 (unico)

Kruskal e accetta $\{1, 3\}, \{4, 5\}, \{2, 3\}, \{3, 4\}, \{5, 6\}$; rifiuta $\{1, 2\}, \{2, 4\}$

Prim da 1 e seleziona 3, 2, 4, 5, 6

Cammini minimi

Prerequisiti

- Da [Capitolo 1 «Grafì»](#): cammini orientati, percorsi e cicli, la matrice di incidenza nodo-arco, gli alberi e il grafo di riferimento.
- Dal modulo Programmazione lineare, [Capitolo 9 «Teoria della dualità»](#): il duale di un modello LP e il teorema della dualità debole.

Obiettivi

Alla fine di questo capitolo sarai in grado di:

- indica i problemi $s-t$ e i problemi del percorso più breve a sorgente singola e riconosce quando si verifica un ciclo negativo loro mal posati;
- utilizza l'ottimalità dei sottopercorsi, le equazioni di Bellman e l'albero dei percorsi più brevi;
- scrivi il modello LP di un percorso minimo come flusso unitario, scrivi il suo duale con i potenziali π_i e usa potenziali ammissibili come certificato di ottimalità;
- esegue l'algoritmo per grafi aciclici, l'algoritmo Bellman–Ford (con il rilevamento di un ciclo negativo) e L'algoritmo di Dijkstra dimostra la loro correttezza e fornisce il loro tempo di esecuzione.

3.1 Problemi del percorso più breve

Definizione 3.1: problemi sul percorso più breve

Sia $G = (N, A)$ un grafo orientato con una **lunghezza** $d_{ij} \in \mathbb{R}$ per ogni arco $(i, j) \in A$, di qualsiasi segno.

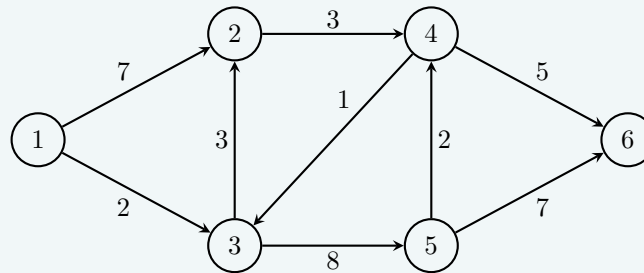
- La lunghezza di un cammino diretto $W = i_0, i_1, \dots, i_k$ è la somma delle lunghezze dei suoi archi, $d(W) = \sum_{t=1}^k d_{i_{t-1}i_t}$.
- Dati due nodi s e t , il problema del percorso più breve $s-t$ richiede un percorso diretto da s a t di lunghezza minima.
- Dato un nodo s , **sorgente, problema del percorso più breve a sorgente singola** richiede un percorso diretto da s a ogni nodo $j \in N$.

La lunghezza minima di un percorso diretto da s a j è **distanza** $\tilde{d}(j)$ di j ; abbiamo impostato $\tilde{d}(j) = +\infty$ se j non è raggiungibile da s .

- In questo capitolo scriviamo $n = |N|$, $m = |A|$, e “percorso” e “ciclo” indicano sempre percorso diretto e percorso diretto ciclo.
- Un algoritmo per il problema della singola origine risolve anche il problema $s-t$, per ogni t contemporaneamente.

Esempio 3.1: il grafo di riferimento con lunghezze

Diamo il grafo di riferimento di [Capitolo 1 «Graf»](#) le lunghezze d_{ij} scritte sugli archi, tutte non negative; la fonte è $s = 1$.



Tre dei percorsi da 1 a 6 e le relative lunghezze sono:

$$d(1, 2, 4, 6) = 7+3+5 = 15, \quad d(1, 3, 2, 4, 6) = 2+3+3+5 = 13, \quad d(1, 3, 5, 6) = 2+8+7 = 17.$$

Le distanze dal nodo 1, calcolate in [Sezione 3.6](#), sono

$$\tilde{d} = (0, 5, 2, 8, 10, 13),$$

quindi 1, 3, 2, 4, 6 è il percorso più breve da 1 a 6.

3.1.1 Cicli negativi

Definizione 3.2: ciclo negativo

Un **ciclo negativo** è un ciclo diretto C con $d(C) < 0$. Nel resto del capitolo ne citeremo due ipotesi:

- (A1) ogni nodo di G è raggiungibile da s ;
- (A2) nessun ciclo negativo di G è raggiungibile da s .

Lemma 3.3: passeggiate e sentieri

Sotto (A2), per ogni cammino diretto W da s a j esiste un percorso diretto P da s a j , con archi di W , tale che $d(P) \leq d(W)$.

Dimostrazione. Per induzione sul numero di archi di W . Se W ha nodi distinti, $P = W$.

1. Altrimenti W ripete un nodo: $W = s, \dots, i_a, \dots, i_b, \dots, j$ con $i_a = i_b$, $a < b$ e $b - a$ minimo. Il pezzo $C = i_a, \dots, i_b$ è un ciclo, raggiungibile da s lungo W .
2. Di (A2), $d(C) \geq 0$. L'eliminazione di C da W consente di eseguire una passeggiata W' da s a j con meno archi e $d(W') = d(W) - d(C) \leq d(W)$.
3. Per induzione W' contiene un percorso P con $d(P) \leq d(W') \leq d(W)$. □

- Sotto l'ipotesi (A2), la lunghezza minima di un cammino da s a j coincide con la distanza $\tilde{d}(j)$. Gli algoritmi di questo capitolo calcolano le lunghezze minime dei cammini; il lemma precedente permette di ricondurle a percorsi.
- Se esiste un ciclo negativo C raggiungibile da s e da cui si può raggiungere j , percorrere C ripetutamente produce cammini da s a j di lunghezza arbitrariamente negativa. In questo caso gli algoritmi rilevano C e non calcolano distanze finite.

Esempio 3.2: un ciclo negativo

Il grafo di riferimento ha due cicli, 2, 4, 3, 2 e 3, 5, 4, 3. Con le lunghezze di [Esempio 3.1](#) le loro lunghezze sono 7 e 11. Se la lunghezza dell'arco (5, 4) scende da 2 a $d''_{54} = -10$, il ciclo 3, 5, 4, 3 ha lunghezza

$$8 - 10 + 1 = -1,$$

e le passeggiate 1, 3, 5, 6, 1, 3, 5, 4, 3, 5, 6, 1, 3, 5, 4, 3, 5, 4, 3, 5, 6, ... da 1 a 6 hanno lunghezze 17, 16, 15, ...: nessuna passeggiata è la più breve. Utilizziamo nuovamente queste lunghezze d'' in [Section 3.5](#).

3.1.2 Ottimità dei sottopercorsi

Lemma 3.4: ottimalità dei sottopercorsi

In (A2), sia P il percorso più breve da s a j e sia i un nodo di P . Quindi il sottopercorso P_{si} di P da s a i è il percorso più breve da s a i .

Dimostrazione. Scrivi P come P_{si} seguito dal sottopercorso P_{ij} da i a j , quindi $d(P) = d(P_{si}) + d(P_{ij})$. Assumere da contraddizione che un percorso Q da s a i abbia $d(Q) < d(P_{si})$.

1. Q seguito da P_{ij} è una passeggiata da s a j di lunghezza $d(Q) + d(P_{ij}) < d(P)$.
2. Da Lemma 3.3 contiene un percorso da s a j più corto di P : un contraddizione. $\square$

Teorema 3.5: Equazioni di Bellman

Per (A1) e (A2), le distanze sono soddisfatte

$$\tilde{d}(s) = 0, \quad \tilde{d}(j) = \min_{(i,j) \in \delta^-(j)} \{ \tilde{d}(i) + d_{ij} \}, \quad \forall j \in N \setminus \{s\},$$

e $\tilde{d}(j) \leq \tilde{d}(i) + d_{ij}$ per ogni arco $(i, j) \in A$.

Dimostrazione. Tutte le distanze sono finite da (A1) e $\tilde{d}(s) = 0$: il percorso s ha lunghezza 0 e un percorso da s a s con archi sarebbe un ciclo, di lunghezza ≥ 0 per (A2).

- *Disuguaglianza.* Un percorso più breve da s a i seguito dall'arco (i, j) è una passeggiata di lunghezza $\tilde{d}(i) + d_{ij}$; di Lemma 3.3, $\tilde{d}(j) \leq \tilde{d}(i) + d_{ij}$.
- *Equazione.* Per $j \neq s$, sia P il percorso più breve da s a j e (i, j) il suo ultimo arco. Di Lemma 3.4, $\tilde{d}(j) = d(P) = \tilde{d}(i) + d_{ij}$, quindi il minimo è raggiunto. $\square$

Proposizione 3.6: shortest path tree

Sotto (A1) e (A2), c'è uno albero ricoprente T di G , lo **albero del percorso più breve**, tale che per ogni $j \in N$ il percorso da s a j in T è diretto ed è il percorso più breve.

Dimostrazione. Per ogni j , sia $k(j)$ il numero minimo di archi di un percorso minimo da s a j ; $k(s) = 0$.

1. Per $j \neq s$, scegliere un percorso più breve P da s a j con archi $k(j)$ e lasciare che $pred(j)$ sia il nodo prima di j su P .
2. Il sottopercorso di P da s a $pred(j)$ è il più breve (Lemma 3.4) con $k(j) - 1$ archi, quindi $k(pred(j)) \leq k(j) - 1$ e $\tilde{d}(j) = \tilde{d}(pred(j)) + d_{pred(j)j}$.

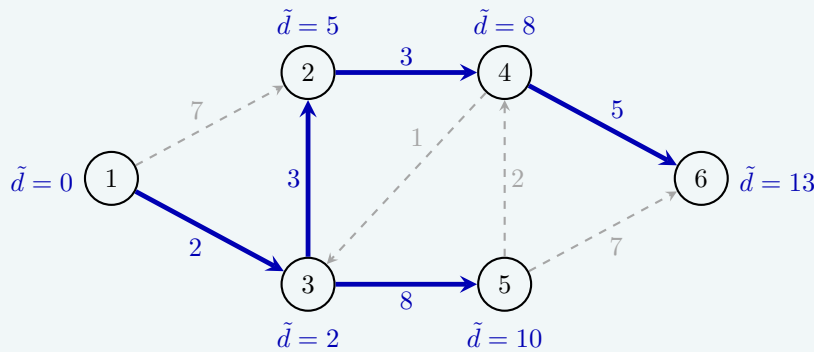
3. Passando da j a $\text{pred}(j)$, il numero k diminuisce rigorosamente, quindi la sequenza termina con s . Gli archi $(\text{pred}(j), j)$ sono $n-1$ e collegano ogni nodo a s : formano un albero ricoprente (Capitolo 1, Teorema «caratterizzazioni di un albero»).
4. Lungo il percorso dell'albero $s = i_0, i_1, \dots, i_r = j$ ogni arco soddisfa $\tilde{d}(i_t) = \tilde{d}(i_{t-1}) + d_{i_{t-1}i_t}$; sommando, la sua lunghezza è $\tilde{d}(j)$. $\square$

Esempio 3.3: Equazioni di Bellman sul grafo di riferimento

Con le distanze $\tilde{d} = (0, 5, 2, 8, 10, 13)$ di Esempio 3.1:

- Il nodo 4 ha gli archi entranti $(2, 4)$ e $(5, 4)$ e $\tilde{d}(4) = \min\{5 + 3, 10 + 2\} = 8$;
- Il nodo 6 ha gli archi entranti $(4, 6)$ e $(5, 6)$ e $\tilde{d}(6) = \min\{8 + 5, 10 + 7\} = 13$.

Gli archi che raggiungono il minimo formano l'albero dei percorsi più brevi $(1, 3)$, $(3, 2)$, $(2, 4)$, $(3, 5)$, $(4, 6)$, in blu; successivo ad ogni nodo la sua distanza.



Senza cicli negativi raggiungibili dalla sorgente, ogni sottopercorso di un percorso più breve è la distanza più breve soddisfare le equazioni di Bellman e uno albero ricoprente contiene il percorso più breve per ogni nodo.

3.2 Condizioni e potenziali di ottimalità

Definizione 3.7: potenziali e costi ridotti

Un vettore $\pi \in \mathbb{R}^n$ fornisce un **potenziale** π_i ad ogni nodo $i \in N$. **costo ridotto** dell'arco (i, j) rispetto a π è

$$\bar{c}_{ij} = d_{ij} - \pi_i + \pi_j.$$

I potenziali sono **ammissibile** se $\bar{c}_{ij} \geq 0$ per ogni arco $(i, j) \in A$, e un arco con $\bar{c}_{ij} = 0$ è **stretto**.

- Dati **etichette** $d(j)$, lunghezze di cammino da s a j , utilizziamo i potenziali $\pi_i = -d(i)$. Poi $\bar{c}_{ij} = d_{ij} + d(i) - d(j)$ e i potenziali sono ammissibili se e solo se $d(j) \leq d(i) + d_{ij}$ per ogni arco.

- Lungo una passeggiata $W = i_0, i_1, \dots, i_k$ il telescopio dei potenziali:

$$d(W) = \sum_{t=1}^k \bar{c}_{i_{t-1}i_t} + \pi_{i_0} - \pi_{i_k}.$$

Teorema 3.8: potenziali come certificato di ottimalità

Sia π un potenziale ammissibile e sia P un percorso da s a t i cui archi siano tutti stretti. Allora P è un percorso più breve da s a t e $d(P) = \pi_s - \pi_t$. Inoltre, G non ha ciclo negativo.

Dimostrazione. Usiamo la formula telescopica sopra.

1. Gli archi di P hanno $\bar{c}_{ij} = 0$, quindi $d(P) = \pi_s - \pi_t$.
2. Per ogni percorso Q da s a t , $d(Q) = \sum_{(i,j) \in Q} \bar{c}_{ij} + \pi_s - \pi_t \geq \pi_s - \pi_t = d(P)$.
3. Per ogni ciclo C il primo e l'ultimo nodo coincidono, quindi $d(C) = \sum_{(i,j) \in C} \bar{c}_{ij} \geq 0$. $\square$

Corollario 3.9: condizioni di ottimalità per le etichette

Lasciamo che le etichette $d(j)$, $j \in N$ siano finite con $d(s) = 0$ e assumiamo che

- (i) $d(j) \leq d(i) + d_{ij}$ per ogni arco $(i, j) \in A$;
- (ii) per ogni $j \neq s$ esiste un percorso da s a j di lunghezza $d(j)$.

Quindi $d(j) = \tilde{d}(j)$ per ogni $j \in N$.

Dimostrazione. Per (i) i potenziali $\pi_i = -d(i)$ sono ammissibili, quindi ogni percorso Q da s a j ha $d(Q) \geq \pi_s - \pi_j = d(j)$ con l'argomento [Teorema 3.8](#). Per (ii) il valore $d(j)$ si raggiunge tramite un percorso, quindi $d(j) = \tilde{d}(j)$. $\square$

Esempio 3.4: un certificato sul grafo di riferimento

Con le distanze di [Esempio 3.1](#) i potenziali sono $\pi = -\tilde{d} = (0, -5, -2, -8, -10, -13)$, and the reduced costs $\bar{c}_{ij} = d_{ij} - \pi_i + \pi_j$ sono:

arco	(1, 2)	(1, 3)	(2, 4)	(3, 2)	(3, 5)	(4, 3)	(4, 6)	(5, 4)	(5, 6)
d_{ij}	7	2	3	3	8	1	5	2	7
$\bar{c}_{ij}$	2	0	0	0	0	7	0	4	4

- Tutti i costi ridotti sono non negativi e gli archi stretti sono gli archi dell'albero del percorso più breve. Il percorso 1, 3, 2, 4, 6 è stretto, quindi è il più corto, con lunghezza $\pi_1 - \pi_6 = 13$.
- L'albero della ricerca in ampiezza, (1, 2), (1, 3), (2, 4), (3, 5), (4, 6), fornisce le etichette

$(0, 7, 2, 10, 10, 15)$. Con i loro potenziali, $\bar{c}_{32} = 3 - (-2) + (-7) = -2 < 0$: i potenziali non sono realizzabili e entrando nel nodo 2 attraverso $(3, 2)$ si accorcia il suo percorso di 2.

I potenziali ammissibili ($\bar{c}_{ij} = d_{ij} - \pi_i + \pi_j \geq 0$ su ogni arco) e un percorso fatto di archi stretti lo dimostrano il percorso è più breve. Con etichette, $\pi_i = -d(i)$, e la fattibilità recita $d(j) \leq d(i) + d_{ij}$.

3.3 La formulazione della programmazione lineare

- Un percorso da s a t invia un'unità di flusso da s a t . Usiamo una variabile $x_{ij} \geq 0$ per ogni arco, il flusso sull'arco e il **bilanci**

$$b_s = 1, \quad b_t = -1, \quad b_i = 0, \quad \forall i \in N \setminus \{s, t\}.$$

Utilizzando queste variabili, un modello LP per il problema del percorso più breve s - t è il seguente:

$$\min \sum_{(i,j) \in A} d_{ij} x_{ij} \quad (3.1a)$$

$$\text{subject to } \sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = b_i, \quad \forall i \in N, \quad (3.1b)$$

$$x_{ij} \geq 0, \quad \forall (i,j) \in A. \quad (3.1c)$$

Questo modello LP è denominato modello $SP(s, t)$.

- Descrizione della funzione obiettivo e dei vincoli del modello $SP(s, t)$:
 - la funzione obiettivo (3.1a) minimizza la lunghezza totale del flusso;
 - Vincoli (3.1b) sono i vincoli di conservazione del flusso: un'unità esce da s , un'unità entra in t , e ogni altro nodo invia ciò che riceve;
 - infine, i vincoli (3.1c) definiscono le variabili del modello LP.
- In forma matriciale il modello è $\min\{d'x : Bx = b, x \geq 0\}$, con il matrice di incidenza nodo-arco B di [Capitolo 1](#): +1 nella coda e -1 nella testa di ogni arco.
- Il **vettore di incidenza** x^P di un percorso P da s a t , con $x_{ij}^P = 1$ sugli archi di P e 0 altrove, è fattibile e il suo valore è $d(P)$.

3.3.1 Il modello duale e i potenziali

- Il vincolo del nodo i è un'equazione, quindi la sua variabile duale π_i è libera. La colonna di x_{ij} in $\mathbf{B}$ ha $+1$ nella riga i e -1 nella riga j , quindi il vincolo duale di $x_{ij} \geq 0$ è $\pi_i - \pi_j \leq d_{ij}$ (Lineare Programmazione, Capitolo 9 «Teoria della dualità»).

Utilizzando una variabile duale libera π_i per ciascun vincolo (3.1b), il modello duale del modello $\text{SP}(s, t)$ è:

$$\max \quad \pi_s - \pi_t \quad (3.2a)$$

$$\text{subject to} \quad \pi_i - \pi_j \leq d_{ij}, \quad \forall (i, j) \in A, \quad (3.2b)$$

$$\pi_i \in \mathbb{R}, \quad \forall i \in N. \quad (3.2c)$$

Questo modello LP è denominato modello $D(\text{SP}(s, t))$.

- Vincolo (3.2b) è $\bar{c}_{ij} = d_{ij} - \pi_i + \pi_j \geq 0$: le soluzioni ammissibili del duale sono i potenziali realizzabili di Definizione 3.7 e $\bar{c}_{ij}$ è il ridotto costo di x_{ij} nel senso LP.

Teorema 3.10: LP e percorsi più brevi

Sotto (A1) e (A2), il modello $\text{SP}(s, t)$ ha una soluzione ottimale e $z(\text{SP}(s, t)) = \tilde{d}(t)$. L'incidenza il vettore di un percorso minimo da s a t è ottimale per $\text{SP}(s, t)$ e i potenziali $\pi_i = -\tilde{d}(i)$ sono ottimale per $D(\text{SP}(s, t))$.

Dimostrazione. Sia P il percorso più breve da s a t , che esiste per (A1).

1. $\mathbf{x}^P$ è fattibile per $\text{SP}(s, t)$, con valore $d(P) = \tilde{d}(t)$.
2. Per Teorema 3.5, $\tilde{d}(j) \leq \tilde{d}(i) + d_{ij}$ per ogni arco, quindi $\pi_i = -\tilde{d}(i)$ soddisfa $\pi_i - \pi_j \leq d_{ij}$; il suo valore è $\pi_s - \pi_t = 0 + \tilde{d}(t)$.
3. Una soluzione primale ammissibile e una soluzione duale ammissibile con lo stesso valore sono entrambe ottimali, per dualità debole. $\square$

Proposizione 3.11: un ciclo negativo rende il modello illimitato

Se $\text{SP}(s, t)$ è fattibile e G ha un ciclo negativo C , raggiungibile da s o meno, allora $\text{SP}(s, t)$ non ha limiti.

Dimostrazione. Sia $\bar{\mathbf{x}}$ ammissibile e sia $\mathbf{x}^C$ il vettore di incidenza di C . Le colonne degli archi di C aggiungono fino a zero (Capitolo 1, Lemma «colonne di un ciclo»), quindi $\mathbf{B}\mathbf{x}^C = \mathbf{0}$. Per ogni $\lambda \geq 0$ il vettore $\bar{\mathbf{x}} + \lambda\mathbf{x}^C$ è fattibile, con valore $\mathbf{d}'\bar{\mathbf{x}} + \lambda d(C)$, che tende a $-\infty$. $\square$

Il segno dei potenziali

Con la matrice di incidenza del percorso (+1 in coda) il vincolo duale è $\pi_i - \pi_j \leq d_{ij}$, e i potenziali ottimali sono $\pi_i = -\tilde{d}(i)$, fino ad una costante additiva. Il modulo $\pi_j - \pi_i \leq d_{ij}$, con $\pi_i = \tilde{d}(i)$, appartiene alla convenzione dei segni opposti (-1 nella coda).

3.3.2 Il modello a fonte unica

- Per trovare il percorso più breve da s a ogni nodo, s invia un'unità a ciascuno degli altri nodi $n - 1$:

$$b_s = n - 1, \quad b_i = -1, \quad \forall i \in N \setminus \{s\}.$$

Con queste bilance il modello (3.1) si chiama $SP(s)$; il suo duale massimizza $(n - 1)\pi_s - \sum_{i \neq s} \pi_i$ con gli stessi vincoli (3.2b).

Proposizione 3.12: il modello a sorgente singola

Sotto (A1) e (A2), sia T un albero dei cammini minimi e, per ogni $j \neq s$, sia T_j l'insieme dei nodi il cui percorso dell'albero passa per j (j incluso). Il flusso con $x_{ij} = |T_j|$ sugli archi (i, j) di T e 0 altrove è ottimale per $SP(s)$, i potenziali $\pi_i = -\tilde{d}(i)$ sono ottimali per il suo duale, e

$$z(SP(s)) = \sum_{j \in N} \tilde{d}(j).$$

Dimostrazione. 1. *Fattibilità.* In un nodo $j \neq s$ il flusso in entrata sull'arco dell'albero è $|T_j|$, e il flusso in uscita è $\sum |T_h|$ sopra gli archi dell'albero (j, h) , cioè $|T_j| - 1$: il saldo è -1 . In s il flusso in uscita copre tutto gli altri nodi: $n - 1$.

2. *Valore.* Il flusso è la somma dei vettori di incidenza dei percorsi dell'albero $n - 1$, poiché l'arco (i, j) giace sui percorsi dell'albero dei nodi $|T_j|$ di T_j . Il suo valore è $\sum_j \tilde{d}(j)$.

3. *Dual.* $\pi_i = -\tilde{d}(i)$ è fattibile come in [Teorema 3.10](#), con valore $(n-1) \cdot 0 + \sum_{i \neq s} \tilde{d}(i)$. Per dualità debole entrambe le soluzioni sono ottimali. $\square$

- In [Capitolo 4 «Flusso di rete modelli»](#) i vertici di questi poliedri risultano interi, per la totale unimodularità di B .

Esempio 3.5: i modelli LP sul grafo di riferimento

Risolviamo i modelli con le lunghezze [Esempio 3.1](#), $s = 1$ e $t = 6$ (valori verificati con Gurobi).

- $SP(1, 6)$: $z(SP(1, 6)) = 13$, con $x_{13} = x_{32} = x_{24} = x_{46} = 1$ e tutte le altre variabili 0: il percorso 1, 3, 2, 4, 6. Il valore duale è $\pi_1 - \pi_6 = 13$.
- $SP(1)$: $z(SP(1)) = 0 + 5 + 2 + 8 + 10 + 13 = 38$, con i flussi sull'albero dei percorsi più brevi

$$x_{13} = 5, \quad x_{32} = 3, \quad x_{24} = 2, \quad x_{46} = 1, \quad x_{35} = 1.$$

Ad esempio $T_3 = \{2, 3, 4, 5, 6\}$, quindi $x_{13} = 5$; il costo è $2 \cdot 5 + 3 \cdot 3 + 3 \cdot 2 + 5 + 8 = 38$.
I duali ottimali, normalizzati con $\pi_1 = 0$, sono $\pi = (0, -5, -2, -8, -10, -13)$.

- Con le lunghezze d'' di [Esempio 3.2](#) il modello SP(1, 6) non ha limiti: ogni unità di il flusso attorno al ciclo 3, 5, 4, 3 abbassa il valore di 1.

Un percorso più breve è un flusso unitario da s a t : $\min d'x$ con $Bx = b$, $x \geq 0$. Its dual has the constraints $\pi_i - \pi_j \leq d_{ij}$, and $\pi_i = -\tilde{d}(i)$ è ottimale; un negativo ciclo ovunque in G rende il modello illimitato.

3.4 Percorsi più brevi nei grafici aciclici

Definizione 3.13: grafo aciclico e ordine topologico

Un grafo diretto senza cicli diretti è un **grafo aciclico diretto** (DAG). Un **ordine topologico** di $G = (N, A)$ è un ordinamento $v_1, v_2, \dots, v_n$ dei suoi nodi tale che ogni arco vada avanti:

$$(v_a, v_b) \in A \implies a < b.$$

Proposizione 3.14: ordini topologici

Un grafo diretto ha un ordine topologico se e solo se è aciclico.

Dimostrazione. Se G avesse un ordine topologico, un ciclo $i_0, i_1, \dots, i_k = i_0$ andrebbe avanti su ogni arco e ritornerebbe al suo primo nodo: impossibile. Viceversa, sia G aciclico; discutiamo per induzione su n .

1. G ha un nodo senza arco entrante: altrimenti, camminare all'indietro lungo archi entranti da qualsiasi nodo, dopo n passi che un nodo ripete e il cammino contiene un ciclo.
2. Metti prima un nodo di questo tipo v_1 . Il grafo senza v_1 è aciclico e per induzione ha un ordine topologico $v_2, \dots, v_n$; nessun arco entra in v_1 , quindi $v_1, v_2, \dots, v_n$ è un ordine topologico di G . $\square$

Algoritmo**Algorithm:** Percorsi più brevi in un grafo aciclico**Input:** un grafo aciclico $G = (N, A)$ con lunghezze d_{ij} di qualsiasi segno, una sorgente $s \in N$ **Output:** le distanze $d(j) = \tilde{d}(j)$ e i predecessori $pred(j)$

- 1 **Passaggio 0 (ordine).** Trova un ordine topologico: rimuovi ripetutamente un nodo senza inserire archi (il più piccolo indice, se più di uno).
- 2 Imposta $d(s) := 0$ e $d(j) := +\infty$ per $j \neq s$.
- 3 **Passaggio 1 (etichette).** Per ciascun nodo i in ordine topologico con $d(i) < +\infty$ e ciascun arco $(i, j) \in \delta^+(i)$:
- 4 se $d(i) + d_{ij} < d(j)$, impostare $d(j) := d(i) + d_{ij}$ e $pred(j) := i$.
- 5 STOP.

Teorema 3.15: correttezza e tempo di esecuzione su grafi aciclici

Su un grafo aciclico con lunghezze di qualsiasi segno, l'algoritmo «Percorsi più brevi in un grafo aciclico» termina con $d(j) = \tilde{d}(j)$ per ogni $j \in N$ e viene eseguito in tempo $O(n + m)$.

Dimostrazione. Un grafo aciclico non ha alcun ciclo, quindi vale la (A2). Ogni etichetta finita è la lunghezza di una passeggiata da s , quindi $d(j) \geq \tilde{d}(j)$ di [Lemma 3.3](#); le etichette diminuiscono solo.

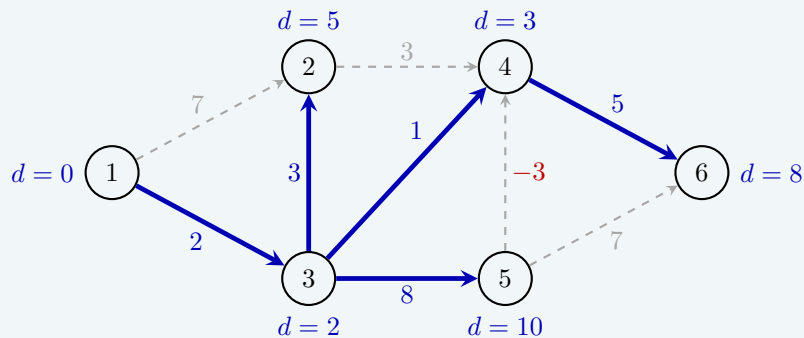
1. Lascia che j sia raggiungibile e $s = i_0, i_1, \dots, i_k = j$ un percorso più breve; i suoi nodi appaiono in questo ordine nel file ordine topologico, poiché ogni arco va avanti.
2. Per induzione su t , dopo l'elaborazione di i_t $d(i_t) \leq \tilde{d}(i_t)$: il percorso verso i_{t-1} è il più breve ([Lemma 3.4](#)), e l'elaborazione i_{t-1} restituisce $d(i_t) \leq \tilde{d}(i_{t-1}) + d_{i_{t-1}i_t} = \tilde{d}(i_t)$.
3. I nodi non raggiungibili da s mantengono $d(j) = +\infty = \tilde{d}(j)$.

Tempo. L'ordine topologico impiega $O(n + m)$ tempo con un contatore degli archi entranti di ogni nodo, decrementato quando viene rimossa una coda; Il passaggio 1 esegue la scansione di ogni arco una volta. $\square$

- La regola dell'indice più piccolo fissa un ordine per gli esempi; qualsiasi nodo senza inserire archi può essere rimosso.
- Con le lunghezze $-d_{ij}$ l'algoritmo calcola **percorsi più lunghi**, poiché un grafo aciclico non ha ciclo ripetere. I percorsi più lunghi nei grafi aciclici sono il nucleo di [Capitolo 9 «Progetto gestione»](#).

Esempio 3.6: un'istanza aciclica

Invertendo l'arco $(4, 3)$ del grafo di riferimento in $(3, 4)$, con $d_{34} = 1$, si ottiene un grafo aciclico. Lo diamo le lunghezze di [Esempio 3.1](#) tranne $d_{54} = -3$ (rosso).



Il passo 0 fornisce l'ordine topologico 1, 3, 2, 5, 4, 6: dopo 1, solo 3 non ha arco entrante; allora 2 e 5 lo sono gratuito e 2 inizia per primo. Ogni riga della tabella elabora un nodo e mostra le etichette dopo di esso:

gradino e nodo i	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$d(6)$
1	1	0	7	2	∞	∞
2	3	0	5	2	3	10
3	2	0	5	2	3	10
4	5	0	5	2	3	10
5	4	0	5	2	3	10
6	6	0	5	2	3	10

I predecessori danno l'albero $(1, 3)$, $(3, 2)$, $(3, 4)$, $(3, 5)$, $(4, 6)$ (blu). L'arco negativo $(5, 4)$ non lo è utilizzato: $d(5) - 3 = 7 > 3 = d(4)$. Con le lunghezze $-d_{ij}$ le lunghezze del percorso più lunghe da 1 sono $(0, 7, 2, 10, 10, 17)$.

Un grafo è aciclico se e solo se ha un ordine topologico. Scansionando i nodi in quest'ordine, ogni etichetta è definitiva quando il suo nodo viene elaborato: percorsi più brevi nel tempo $O(n + m)$, con lunghezze di qualsiasi segno.

3.5 L'algoritmo Bellman–Ford

Definizione 3.16: etichette con al massimo k archi

Per $k \geq 0$ e $j \in N$, $d^k(j)$ è la lunghezza minima di un percorso da s a j con al massimo k archi e $d^k(j) = +\infty$ se non ce n'è.

Lemma 3.17: ricorsione delle etichette

$d^0(s) = 0$, $d^0(j) = +\infty$ per $j \neq s$ e per ogni $k \geq 1$

$$d^k(j) = \min \left\{ d^{k-1}(j), \min_{(i,j) \in \delta^-(j)} (d^{k-1}(i) + d_{ij}) \right\}, \quad \forall j \in N.$$

Dimostrazione. L'unico cammino con zero archi è quello che parte e termina in s . Per $k \geq 1$, un cammino da s a j con al massimo k archi o ne contiene al massimo $k-1$, e ha lunghezza minima $d^{k-1}(j)$, oppure contiene esattamente k archi e termina con un arco (i, j) . In quest'ultimo caso, prima percorre un cammino da s a i con $k-1$ archi; la lunghezza minima è quindi $\min_{(i,j) \in A} (d^{k-1}(i) + d_{ij})$. $\square$

Algoritmo

Algorithm: Algoritmo Bellman–Ford

Input: un grafo diretto $G = (N, A)$ con lunghezze d_{ij} di qualsiasi segno, una sorgente $s \in N$

Output: le distanze $\tilde{d}(j)$ e i predecessori, oppure un ciclo negativo raggiungibile da s

- 1 **Passaggio 0 (inizializzazione).** Imposta $d^0(s) := 0$, $d^0(j) := +\infty$ per $j \neq s$ e $k := 1$.
- 2 **Passaggio 1 (passa k).** Calcola $d^k(j)$ per ogni $j \in N$ con la ricorsione di [Lemma 3.17](#);
- 3 se $d^k(j) < d^{k-1}(j)$, imposta $pred(j)$ alla coda i dell'arco che raggiunge il minimo.
- 4 **Passaggio 2 (test).** Se $d^k = d^{k-1}$, STOP: $\tilde{d} = d^k$.
- 5 Se $k = n$, STOP: un ciclo negativo è raggiungibile da s . Altrimenti imposta $k := k + 1$ e vai al passo 1.

- Ogni passaggio dipende solo dalle etichette del passaggio precedente: se $d^k = d^{k-1}$, tutti i passaggi successivi si ripetono le stesse etichette e l'algoritmo può fermarsi.
- Negli esempi, i pari merito nel minimo vengono interrotti dalla coda più piccola i e il predecessore cambia solo quando l'etichetta diminuisce rigorosamente.

Teorema 3.18: correttezza di Bellman–Ford

Se non è raggiungibile alcun ciclo negativo da s , allora $d^{n-1}(j) = \tilde{d}(j)$ per ogni $j \in N$ e l'algoritmo si ferma al passaggio $k \leq n$ con $d^k = d^{k-1} = \tilde{d}$.

Dimostrazione. Un percorso ha al massimo $n - 1$ archi, quindi $d^{n-1}(j) \leq \tilde{d}(j)$. Di [Lemma 3.3](#) (che necessita solo dei cicli raggiungibili da s) ogni cammino da s a j contiene un percorso non più lungo, quindi $d^k(j) \geq \tilde{d}(j)$ per ogni k .

- Quindi $d^{n-1} = d^n = \tilde{d}$.
- Le etichette non aumentano mai e l'algoritmo si ferma al primo passaggio k con $d^k = d^{k-1}$, che è al massimo n ; da quel passaggio in poi le etichette non cambiano più, quindi $d^k = d^{n-1} = \tilde{d}$. $\square$

Teorema 3.19: rilevamento di un ciclo negativo

Un ciclo negativo è raggiungibile da s se e solo se $d^n \neq d^{n-1}$. In tal caso, lascia che $d^n(j) < d^{n-1}(j)$; leggendo i predecessori all'indietro da j , una riga per passaggio, partendo dalla riga n della tabella, dà una passeggiata da s a j con archi n che contengono un ciclo negativo.

Dimostrazione. Se non è possibile raggiungere alcun ciclo negativo da s , allora $d^n = d^{n-1}$ mediante [Teorema 3.18](#). Al contrario, lascia che $d^n(j) < d^{n-1}(j)$.

1. Ogni cammino verso j di lunghezza $d^n(j)$ con al massimo n archi ha esattamente n archi, altrimenti $d^{n-1}(j) \leq d^n(j)$.
2. Lascia che i sia il predecessore di j nella riga n , quindi $d^n(j) = d^{n-1}(i) + d_{ij}$. Una passeggiata realizzando $d^{n-1}(i)$ seguito da (i, j) realizza $d^n(j)$, quindi ha esattamente archi $n - 1$: $d^{n-1}(i) < d^{n-2}(i)$, e i ha un predecessore nella riga $n - 1$.
3. Ripetendo l'argomento fino alla riga 1, il cui predecessore è s , si ottiene una passeggiata W da s a j con n archi e $d(W) = d^n(j)$.
4. W ha nodi $n + 1$, quindi ripete un nodo e contiene un ciclo C . L'eliminazione di C dà una passeggiata a j con meno rispetto agli archi n , di lunghezza $d(W) - d(C) \geq d^{n-1}(j) > d^n(j) = d(W)$. Pertanto $d(C) < 0$ e C sono raggiungibili da s lungo W . $\square$

Proposizione 3.20: l'albero predecessore di Bellman–Ford

Se nessun ciclo negativo è raggiungibile da s , alla fermata gli archi $(pred(j), j)$, per i nodi $j \neq s$ con finito etichetta, forma un albero dei percorsi più brevi dei nodi raggiungibili.

Dimostrazione. Sia $d = \tilde{d}$ l'etichetta finale, e sia $t(j)$ l'ultimo passaggio in cui l'etichetta di j è diminuita; $t(s) = 0$, da $d^k(s) = 0$ per tutti gli k . Lascia che $p = pred(j)$.

1. Nel passaggio $t(j)$, $d(j) = d^{t(j)-1}(p) + d_{pj} \geq d(p) + d_{pj}$, poiché le etichette non aumentano mai.
2. Di [Teorema 3.5](#), $d(j) \leq d(p) + d_{pj}$. Quindi $d(j) = d(p) + d_{pj}$ e $d(p) = d^{t(j)-1}(p)$:

l'etichetta di p non è cambiata dopo il passaggio $t(j) - 1$, quindi $t(p) < t(j)$.

3. Passando al predecessore t diminuisce rigorosamente, quindi la sequenza termina in s , lungo archi stretti: il percorso dell'albero da s a j ha lunghezza $d(j)$. $\square$

Teorema 3.21: tempo di esecuzione di Bellman–Ford

L'algoritmo Bellman–Ford viene eseguito nel tempo $O(nm)$.

Dimostrazione. Un passaggio calcola $d^k(j)$ per ogni j scansionando la stella all'indietro $\delta^-(j)$, quindi scansiona ogni arco una volta: $O(n + m)$ tempo. Sono disponibili al massimo n passaggi e il totale è $O(n(n + m))$, che è $O(nm)$ quando $m \geq n - 1$, come sotto (A1). $\square$

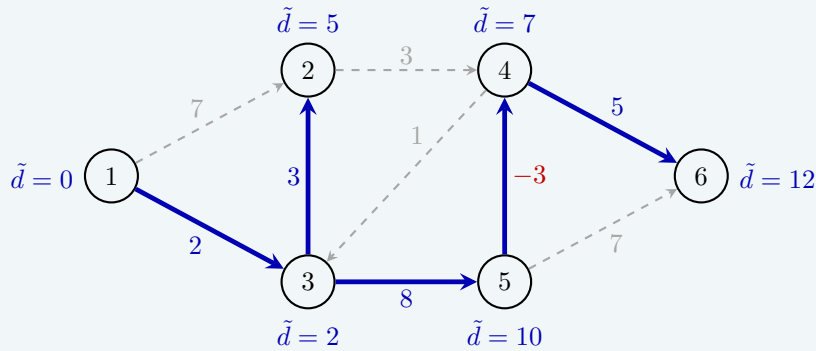
3.5.1 Bellman–Ford sul grafo di riferimento

Esempio 3.7: Bellman–Ford con un arco negativo

Utilizziamo le lunghezze d' di [Esempio 3.1](#) con $d'_{54} = -3$ al posto di 2. I due cicli hanno lunghezze 7 e $8 - 3 + 1 = 6$, quindi non esiste un ciclo negativo. Ogni riga è un passaggio k ; accanto a un'etichetta diminuita, tra parentesi il nuovo predecessore:

k	$d^k(1)$	$d^k(2)$	$d^k(3)$	$d^k(4)$	$d^k(5)$	$d^k(6)$
0	0	∞	∞	∞	∞	∞
1	0	7 (1)	2 (1)	∞	∞	∞
2	0	5 (3)	2	10 (2)	10 (3)	∞
3	0	5	2	7 (5)	10	15 (4)
4	0	5	2	7	10	12 (4)
5	0	5	2	7	10	12

- In passaggio 3, $d^3(4) = \min\{10, d^2(2) + 3, d^2(5) - 3\} = \min\{10, 8, 7\} = 7$, con il predecessore 5.
- Passaggio 5 ripete le etichette del passaggio 4: l'algoritmo si ferma con $\tilde{d} = (0, 5, 2, 7, 10, 12)$, dopo 5 passaggi consentiti invece dello $n = 6$.
- L'albero del percorso più breve ha gli archi $(1, 3)$, $(3, 2)$, $(3, 5)$, $(5, 4)$, $(4, 6)$ e il percorso più breve per 6 è $1, 3, 5, 4, 6$, di lunghezza $2 + 8 - 3 + 5 = 12$.


Esempio 3.8: Bellman–Ford rileva un ciclo negativo

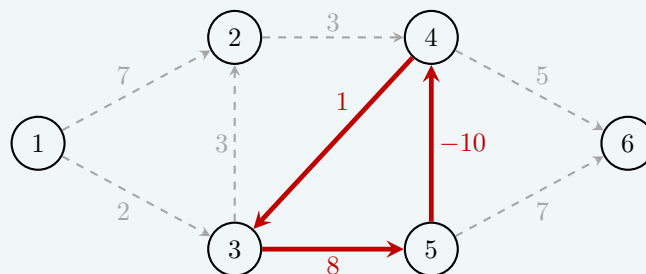
Con le lunghezze d'' di [Esempio 3.2](#) ($d''_{54} = -10$) l'algoritmo esegue tutti i $n = 6$ passa:

k	$d^k(1)$	$d^k(2)$	$d^k(3)$	$d^k(4)$	$d^k(5)$	$d^k(6)$
0	0	∞	∞	∞	∞	∞
1	0	7 (1)	2 (1)	∞	∞	∞
2	0	5 (3)	2	10 (2)	10 (3)	∞
3	0	5	2	0 (5)	10	15 (4)
4	0	5	1 (4)	0	10	5 (4)
5	0	4 (3)	1	0	9 (3)	5
6	0	4	1	-1 (5)	9	5

- Nel passaggio 6 l'etichetta del nodo 4 diminuisce da 0 a -1 : $d^6 \neq d^5$ e viene generato un ciclo negativo raggiungibile da 1.
- Lettura all'indietro dal nodo 4 (voci in grassetto): la riga 6 restituisce $pred = 5$, la riga 5 per il nodo 5 restituisce 3, la riga 4 per il nodo 3 restituisce 4, quindi 5, 3, 1. La passeggiata è

1, 3, 5, 4, 3, 5, 4, di lunghezza $2 + 8 - 10 + 1 + 8 - 10 = -1 = d^6(4)$.

- Nodo 3 ripete: la passeggiata contiene il ciclo 3, 5, 4, 3, di lunghezza -1 (rosso).



L'algoritmo Bellman–Ford calcola d^k , i percorsi più brevi con al massimo archi k , nel tempo $O(nm)$. Senza cicli negativi raggiungibili da s si ferma entro n passa con le distanze; un cambiamento nel passaggio n rivela a ciclo negativo, trovato leggendo i predecessori all'indietro.

3.6 Algoritmo di Dijkstra

- In questa sezione tutte le lunghezze sono non negative: $d_{ij} \geq 0$ per ogni arco. Quindi ogni ciclo ha una lunghezza ≥ 0 e (A2) sono validi.
- I nodi sono **permanente**, quando la loro etichetta è definitiva, o **temporaneo**. Ad ogni iterazione il nodo temporaneo con l'etichetta più piccola diventa permanente.

Algoritmo

Algorithm: Algoritmo di Dijkstra

Input: un grafo diretto $G = (N, A)$ con lunghezze $d_{ij} \geq 0$, una sorgente $s \in N$

Output: le distanze $d(j) = \tilde{d}(j)$ e i predecessori $pred(j)$

- 1 **Passaggio 0 (inizializzazione).** Imposta $d(s) := 0$, $d(j) := +\infty$ per $j \neq s$; tutti i nodi sono temporanei.
- 2 **Passaggio 1 (selezione).** Se nessun nodo temporaneo ha un'etichetta finita, STOP.
- 3 Altrimenti rendere permanente il nodo temporaneo i con il più piccolo $d(i)$ (l'indice più piccolo, se superiore a uno).
- 4 **Passaggio 2 (aggiornamento).** Per ogni arco $(i, j) \in \delta^+(i)$ con j temporaneo: se $d(i) + d_{ij} < d(j)$, impostare $d(j) := d(i) + d_{ij}$ e $pred(j) := i$.
- 5 Vai al passaggio 1.

Teorema 3.22: correttezza dell'algoritmo di Dijkstra

Se $d_{ij} \geq 0$ per ogni arco, ad ogni iterazione dell'algoritmo di Dijkstra:

- (a) ogni nodo permanente i ha $d(i) = \tilde{d}(i)$;
- (b) ogni nodo temporaneo j ha come $d(j)$ la lunghezza minima di un percorso da s a j i cui nodi diversi da j sono tutti permanenti ($+\infty$ se non ce n'è nessuno).

Alla fermata, $d(j) = \tilde{d}(j)$ per ogni $j \in N$.

Dimostrazione. Entrambe le proprietà restano valide dopo il passaggio 0, senza nodo permanente. Lascia che i sia il nodo selezionato nel passaggio 1.

1. $d(i) = \tilde{d}(i)$. Sia P un percorso qualsiasi da s a i e h sia il suo primo nodo temporaneo. Il parte di P fino a h ha lunghezza $\geq d(h)$ di (b) e il resto ha lunghezza ≥ 0 , a partire da $d \geq 0$. Quindi $d(P) \geq d(h) \geq d(i)$ e per (b) il valore $d(i)$ è la lunghezza di un percorso: $d(i) = \tilde{d}(i)$.
2. (b) dopo il passo 2. Sia j temporaneo e Q un percorso verso j i cui altri nodi siano

permanententi, con h il nodo prima di j . Se $h = i$, allora $d(Q) \geq \tilde{d}(i) + d_{ij} = d(i) + d_{ij}$, il valore testato nel passaggio 2.

3. Se $h \neq i$, allora h diventava permanente prima di i , attraverso un percorso di nodi permanententi di lunghezza $\tilde{d}(h)$; quando è stato selezionato h , il passaggio 2 ha prodotto $d(j) \leq \tilde{d}(h) + d_{hj} \leq d(Q)$.

Quindi il nuovo minimo è $\min\{d(j), d(i) + d_{ij}\}$, raggiunto da un percorso, che è l'etichetta lasciata dal passaggio 2. Alla fermata, i nodi temporanei hanno $d(j) = +\infty$: per (b) nessun arco lascia il permanente impostato verso di loro, quindi non sono raggiungibili e $\tilde{d}(j) = +\infty$. $\square$

- Per il problema $s-t$ l'algoritmo può interrompersi non appena t è permanente, mediante (a).
- Gli archi $(pred(j), j)$ formano un albero del percorso più breve: $pred(j)$ è permanente prima di j e quando viene impostato l'ultima volta $d(j) = d(pred(j)) + d_{pred(j)j}$ con entrambe le etichette finali.

Teorema 3.23: tempo di esecuzione dell'algoritmo di Dijkstra

Con le etichette in un array, l'algoritmo di Dijkstra viene eseguito nel tempo $O(n^2)$.

Dimostrazione. Esistono al massimo le iterazioni n e il passaggio 1 esegue la scansione delle etichette temporanee nel tempo $O(n)$. Il passaggio 2 esegue la scansione dell'intero stella del nodo selezionato e ciascun nodo viene selezionato una volta, quindi tutti gli aggiornamenti richiedono tempo $O(m)$. Il totale è $O(n^2 + m) = O(n^2)$, da $m \leq n(n-1)$. $\square$

- Con le etichette temporanee in un heap binario, il passaggio 1 e ogni diminuzione di un'etichetta richiedono tempo $O(\log n)$: il totale è $O((n+m) \log n)$, migliore di $O(n^2)$ su grafi sparsi.

3.6.1 Algoritmo di Dijkstra sul grafo di riferimento

Esempio 3.9: Dijkstra dal nodo 1

Eseguiamo l'algoritmo con le lunghezze [Esempio 3.1](#). Ogni riga è un'iterazione: il nodo selezionato diventa permanente (etichetta in grassetto) e un'etichetta diminuita ha il nuovo predecessore tra parentesi.

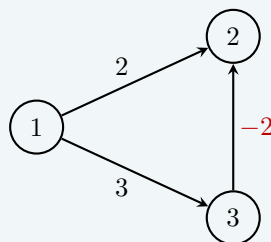
iterazione e i	$d(1)$	$d(2)$	$d(3)$	$d(4)$	$d(5)$	$d(6)$
1	1	0	7 (1)	2 (1)	∞	∞
2	3	0	5 (3)	2	∞	10 (3)
3	2	0	5	2	8 (2)	10
4	4	0	5	2	8	10 13 (4)
5	5	0	5	2	8	10 13
6	6	0	5	2	8	10 13

- Nell'iterazione 2 l'etichetta del nodo 2 scende da 7 a $2 + 3 = 5$; nell'iterazione 5 l'arco $(5, 6)$ offre $10 + 7 = 17 > 13$ e l'arco $(5, 4)$ termina con un nodo permanente.
- I nodi diventano permanenti nell'ordine 1, 3, 2, 4, 5, 6, con $\tilde{d} = (0, 5, 2, 8, 10, 13)$; danno i predecessori l'albero del percorso più breve di [Esempio 3.3](#).

3.6.2 Lunghezze negative

Esempio 3.10: un arco negativo interrompe l'algoritmo di Dijkstra

Nel grafo sottostante, con $s = 1$, l'algoritmo rende permanente il nodo 2 con $d(2) = 2$, quindi il nodo 3 con $d(3) = 3$; l'arco $(3, 2)$ termina con un nodo permanente e viene ignorato. Il percorso 1, 3, 2 ha lunghezza $3 - 2 = 1 < 2$.



Sul grafo di riferimento con le lunghezze d' di [Esempio 3.7](#) ($d'_{54} = -3$), l'algoritmo di Dijkstra rende permanenti i nodi nell'ordine 1, 3, 2, 4, 5, 6 e restituisce $d(4) = 8$ e $d(6) = 13$: il nodo 4 è permanente prima nodo 5, quindi il percorso più breve 1, 3, 5, 4 di lunghezza 7 viene perso e con esso il percorso verso 6 di lunghezza 12.

- La dimostrazione di [Teorema 3.22](#) utilizza $d \geq 0$ in un punto: il resto di P dopo il suo primo nodo temporaneo ha lunghezza ≥ 0 . Un arco negativo interrompe esattamente questo passaggio.

Con lunghezze non negative, l'algoritmo di Dijkstra rende permanente, ad ogni iterazione, il nodo temporaneo con il etichetta più piccola, la cui etichetta è quindi definitiva; funziona in tempo $O(n^2)$. Con un arco negativo può fallire e il Viene invece utilizzato l'algoritmo Bellman–Ford.

3.7 Errori tipici

Attenzione

- **Esecuzione dell'algoritmo di Dijkstra con un arco negativo.** Le sue etichette possono essere errate, come con $d'_{54} = -3$ su il grafo di riferimento; utilizzare Bellman–Ford o l'ordine topologico se il grafo è aciclico.
- **Aggiungere una costante a ogni lunghezza.** Rende le lunghezze non negative ma modifica i percorsi più brevi: un percorso con archi k cresce di k volte la costante.
- **Scambio di segno dei potenziali.** Con $+1$ nella coda di $\mathbf{B}$ i vincoli duali sono $\pi_i - \pi_j \leq d_{ij}$ e i potenziali ottimali sono $\pi_i = -\tilde{d}(i)$.
- **Saltare il test di passaggio n .** Dopo che $n - 1$ ha superato le etichette sono solo le distanze senza negativo cicli; un cambiamento nel passaggio n ne rivela uno.
- **Mi aspetto che Bellman–Ford veda ogni ciclo negativo.** Rileva solo i cicli raggiungibili da s , mentre un ciclo negativo ovunque in G rende il modello LP illimitato.
- **Aggiornamento di un nodo permanente nell'algoritmo di Dijkstra.** La sua etichetta è definitiva; lo sono solo i nodi temporanei aggiornato.
- **Ricerca di un ordine topologico in un grafo con cicli.** Esiste solo nei grafi aciclici: il il grafo di riferimento non ne ha nessuno.

3.8 In sintesi

In sintesi

- La distanza $\tilde{d}(j)$ è la lunghezza del percorso più breve da s a j ; è ben definito quando non il ciclo negativo è raggiungibile da s (A2).
- I sottopercorsi dei percorsi più brevi sono i più brevi; le distanze soddisfano le equazioni di Bellman e un albero dei cammini più brevi esiste.
- I potenziali fattibili, $\bar{c}_{ij} = d_{ij} - \pi_i + \pi_j \geq 0$ e un percorso di archi stretti certificano l'ottimalità.
- Il modello LP è un'unità di flusso, $\mathbf{B}\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$; il suo duale è $\max\{\pi_s - \pi_t : \pi_i - \pi_j \leq d_{ij}\}$, ottimale su $\pi_i = -\tilde{d}(i)$.

- Grafi aciclici: ordine topologico, $O(n + m)$, qualsiasi lunghezza. Bellman–Ford: $O(nm)$, qualsiasi lunghezza, rileva cicli negativi. Dijkstra: $O(n^2)$, lunghezze non negative.

Il grafo di riferimento in sintesi.

lunghezze d (Dijkstra) e $\tilde{d} = (0, 5, 2, 8, 10, 13)$, ordinare 1, 3, 2, 4, 5, 6

Modelli LP e $z(\text{SP}(1, 6)) = 13$, $z(\text{SP}(1)) = 38$

lunghezze d' , $d'_{54} = -3$ (Bellman–Ford) e $\tilde{d} = (0, 5, 2, 7, 10, 12)$, 5 passa

lunghezze d'' , $d''_{54} = -10$ e ciclo negativo 3, 5, 4, 3 della lunghezza -1

grafo aciclico, arco (3, 4) e ordine 1, 3, 2, 5, 4, 6; $\tilde{d} = (0, 5, 2, 3, 10, 8)$
