Alberi ricoprenti di costo minimo

Indice

Il problema dell'albero ricoprente di costo minimo
    Applicazioni

Tagli e cicli

La proprietà del taglio e la proprietà del ciclo
    Condizioni di ottimalità

L'algoritmo greedy generico

Algoritmo di Kruskal
    Implementazione e tempo di esecuzione
    Algoritmo di Kruskal sul grafo di riferimento

Algoritmo di Prim
    Algoritmo di Prim sul grafo di riferimento

Unicità dell'albero ricoprente di costo minimo
    Pesi con valori uguali

Errori tipici

Obiettivi

Alla fine di questo capitolo sarai in grado di:

- enunciare il problema dell'albero ricoprente di costo minimo e riconoscerlo nelle applicazioni;
- dimostrare la proprietà del taglio e la proprietà del ciclo, e usarle per certificare che un albero ricoprente è ottimo;
- eseguire gli algoritmi di Kruskal e di Prim, dimostrare che sono corretti e analizzarne il tempo di esecuzione;
- stabilire se l'albero ricoprente di costo minimo è unico e trovare gli spigoli che appartengono a ogni albero ottimo.

Prerequisiti

- Il capitolo «Grafi»: cammini e cicli, componenti connesse, le caratterizzazioni di un albero e i cicli fondamentali.
- Dal modulo *Algoritmi*, i capitoli «Notazione asintotica» e «Algoritmi ricorsivi»: la notazione $O(\cdot)$ per il tempo di esecuzione di un algoritmo, e l'ordinamento di m numeri in tempo $O(m \log m)$ (merge sort).

Il problema dell'albero ricoprente di costo minimo

Introduzione: il problema dell'albero ricoprente di costo minimo

- Vogliamo collegare tutti i vertici di un grafo pesato con un insieme di spigoli di peso totale minimo
- La rete connessa più economica è sempre un albero ricoprente: il problema chiede il migliore
- In questa sezione enunciamo il problema, fissiamo il grafo di riferimento pesato usato in tutti gli esempi, e vediamo dove il problema compare nelle applicazioni

Definizione 1: problema dell'albero ricoprente di costo minimo

*Sia $G = (V, E)$ un grafo non orientato connesso con $n = |V|$ vertici e $m = |E|$ spigoli, e sia ogni spigolo $e \in E$ dotato di un **peso** $w_e \in \mathbb{R}$. Il **peso** di un insieme di spigoli $F \subseteq E$ è*

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

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

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

Definizione: problema dell'albero ricoprente di costo minimo (continua)

Un albero ricoprente di peso minimo è un ***albero ricoprente di costo minimo****. Scriviamo* w_{uv} *per il peso dello spigolo* $\{u, v\}$*, e identifichiamo un albero ricoprente con il suo insieme di spigoli.*

Prime osservazioni

- Un grafo connesso ha un albero ricoprente, e ne ha un numero finito: un albero ricoprente di costo minimo esiste sempre. Ogni albero ricoprente ha $n - 1$ spigoli.
- I pesi possono essere negativi. Aggiungere la stessa costante a ogni peso aggiunge $(n - 1)$ volte quella costante al peso di ogni albero ricoprente, quindi gli alberi ricoprenti di costo minimo non cambiano.
- Un **albero ricoprente di costo massimo** è un albero ricoprente di costo minimo per i pesi $-w_e$.

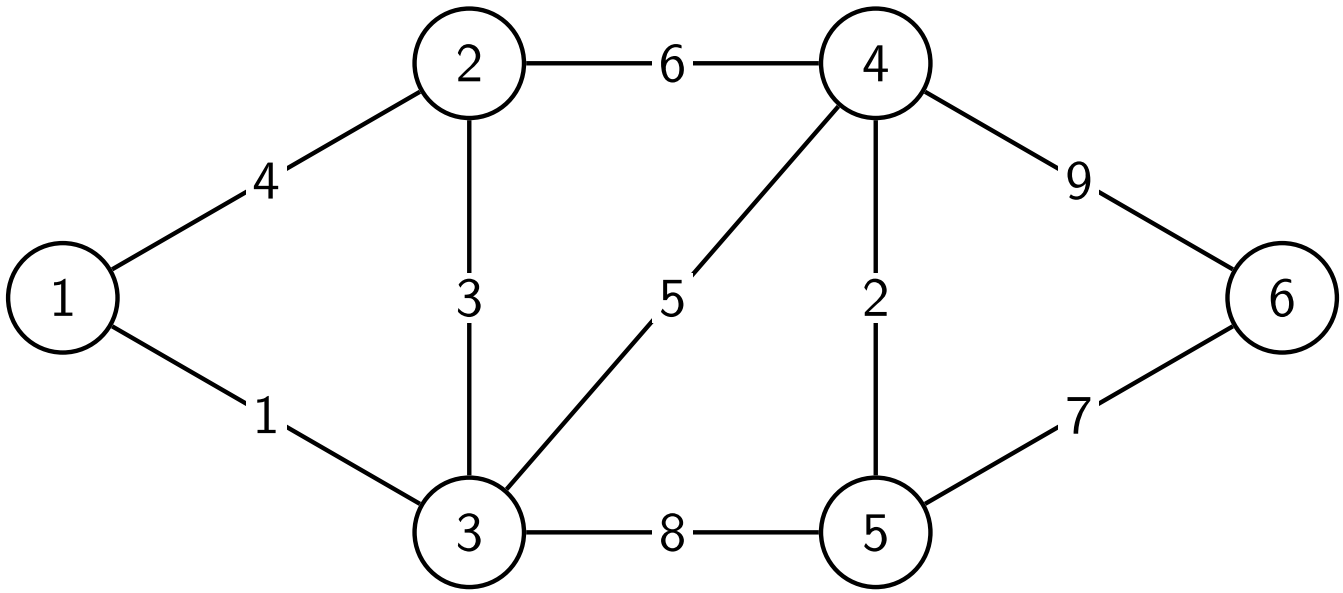
Prima di studiare il problema, fissiamo un grafo pesato:
tutti gli esempi del capitolo lo usano.

Esempio: il grafo di riferimento pesato (1/2)

- Prendiamo il grafo non orientato soggiacente al grafo di riferimento del capitolo «Grafi», con $n = 6$ vertici e $m = 9$ spigoli
- I pesi sono

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

Esempio: il grafo di riferimento pesato (2/2)



- Chiamiamo questo grafo **grafo di riferimento pesato**: tutti gli esempi del capitolo lo usano.
- I pesi sono gli interi distinti $1, 2, \ldots, 9$.

Esempio: l'albero ricoprente di costo minimo del grafo di riferimento (1/2)

- Il grafo di riferimento pesato ha 55 alberi ricoprenti
- Enumerandoli tutti, il peso minimo è 18, raggiunto solo dall'albero

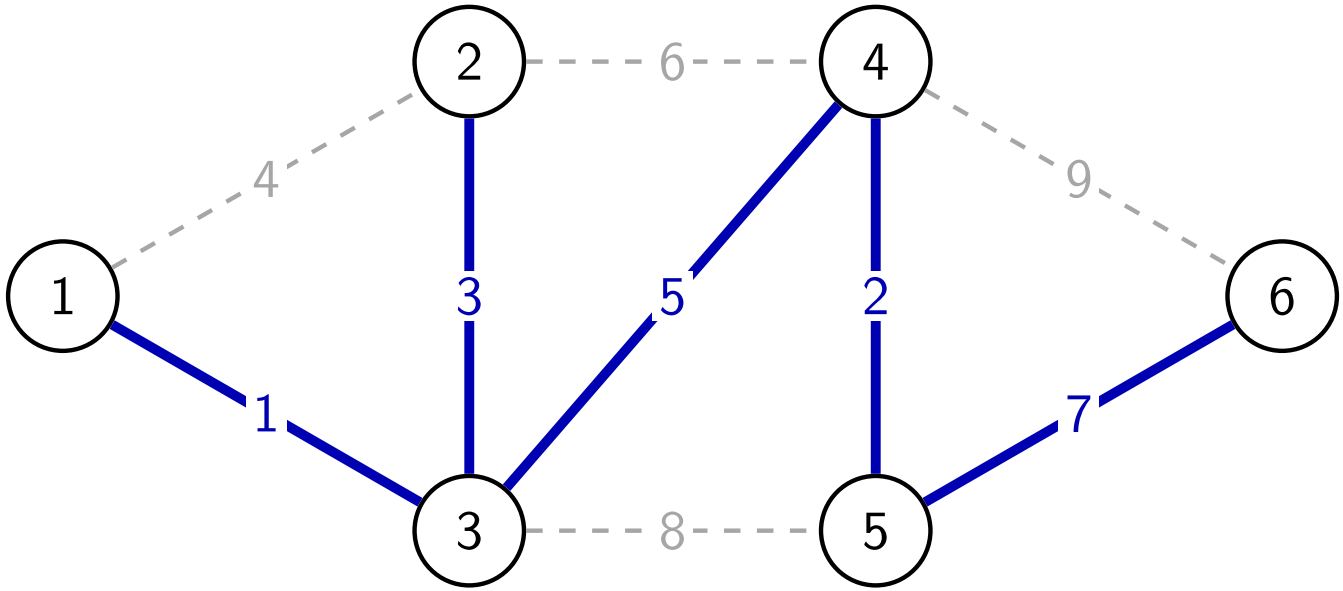
$$\tilde{F} = \{\{1,3\},\{2,3\},\{3,4\},\{4,5\},\{5,6\}\}$$

con

$$w(\tilde{F}) = 1+3+5+2+7 = 18$$

Esempio: l'albero ricoprente di costo minimo del grafo di riferimento (2/2)

L'albero $\tilde{F}$ è in blu nella figura.



- L'albero ricoprente $\{1,2\},\{1,3\},\{2,4\},\{3,5\},\{4,6\}$ trovato dalla visita in ampiezza nel capitolo «Grafi» pesa $4+1+6+8+9=28$
- Un albero ricoprente di costo massimo pesa 34

Perché servono gli algoritmi

- Enumerare gli alberi ricoprenti è praticabile solo su grafi piccolissimi
- Il grafo di riferimento, con 6 vertici e 9 spigoli, ne ha già 55
- Gli algoritmi di questo capitolo trovano un albero ricoprente di costo minimo in tempo polinomiale

Applicazioni

- **Progettazione di reti.** I vertici sono siti (uffici, case, stabilimenti) e w_{uv} è il costo di un collegamento diretto (un cavo, un tubo, una strada); la rete più economica che collega tutti i siti è un albero ricoprente minimo.

- **Clustering.** Togliendo i $k-1$ spigoli più pesanti di un albero ricoprente di costo minimo, i vertici si dividono in k gruppi di punti vicini (clustering a collegamento singolo).

- **Sottoproblema.** Gli alberi ricoprenti di costo minimo danno limitazioni inferiori per il problema del commesso viaggiatore e un punto di partenza per algoritmi euristici di progettazione di reti.

L'albero ricoprente di costo minimo in una frase

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

Riepilogo

Il problema:

- Trovare un albero ricoprente di peso totale minimo di un grafo pesato connesso
- Esiste sempre e ha $n-1$ spigoli; pesi negativi e alberi ricoprenti di costo massimo sono ammessi

Il grafo di riferimento pesato:

- $n = 6$, $m = 9$, pesi $1, 2, \ldots, 9$, 55 alberi ricoprenti
- Un unico albero ricoprente di costo minimo $\tilde{F}$, di peso 18

Tagli e cicli

Introduzione: tagli e cicli

- Tagli e cicli sono i due strumenti dietro a ogni risultato di questo capitolo
- Definiamo i tagli e i tagli fondamentali, e dimostriamo che un ciclo attraversa un taglio un numero pari di volte
- Dal capitolo «Grafi»: per un albero ricoprente T e uno spigolo $e \notin T$, il **ciclo fondamentale** $C(T, e)$ è l'unico ciclo di $T \cup \{e\}$
- Il lemma dello scambio dice come passare da un albero ricoprente a un altro

Definizione 2: taglio

Sia $G = (V, E)$ un grafo non orientato e $S \subseteq V$ con $\emptyset \neq S \neq V$. Il ***taglio*** *definito da S è l'insieme degli spigoli con esattamente un estremo in S:*

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

Uno spigolo di $\delta(S)$ ***attraversa*** *il taglio. Per un singolo vertice, $\delta(\{v\}) = \delta(v)$ è l'insieme dei spigoli incidenti in v.*

Tagli e connessione

- Un grafo è connesso se e solo se ogni taglio $\delta(S)$ è non vuoto:
  - un taglio vuoto separa S da $V \setminus S$;
  - un cammino da $u \in S$ a $v \notin S$ deve usare uno spigolo di $\delta(S)$.

- Nel grafo di riferimento pesato

$$\delta(\{1,2,3\}) = \{\{2,4\},\{3,4\},\{3,5\}\}$$

e

$$\delta(\{6\}) = \{\{4,6\},\{5,6\}\}$$

Lemma 1: 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 uno spigolo di $\delta(S)$, contiene almeno un altro spigolo di $\delta(S)$.

Dimostrazione

- Il ciclo visiti $v_0, v_1, \ldots, v_k = v_0$
- Lo spigolo $\{v_{t-1}, v_t\}$ attraversa il taglio se e solo se v_{t-1} e v_t stanno da parti diverse
- Percorrendo il ciclo, la parte cambia a ogni spigolo che attraversa il taglio, e solo lì
- Il percorso termina in v_0, dalla parte da cui era partito, quindi il numero di cambi è pari

□

Definizione 3: taglio fondamentale

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

Perché il taglio fondamentale è ben definito

- $T \setminus \{e\}$ ha $n-2$ spigoli e nessun ciclo, quindi ha $n-(n-2)=2$ componenti (Corollario «spigoli e componenti»)
- v non sta in S_e, altrimenti il cammino da u a v in T non sarebbe unico
- L'unico spigolo di T nel taglio fondamentale di e è e stesso: ogni altro spigolo di T sta dentro una delle due componenti

Con i cicli fondamentali e i tagli fondamentali
possiamo ora scambiare uno spigolo di un albero ricoprente
con un altro.

Lemma 2: scambio

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

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

Dimostrazione (1/2)

Parte 1.

- Il resto di $C(T, e)$ collega gli estremi di f
- Quindi togliere f dal grafo connesso $T \cup \{e\}$ lo lascia connesso
- Il risultato è connesso con $n - 1$ spigoli: è un albero ricoprente per le caratterizzazioni di un albero, (b)

Dimostrazione (2/2)

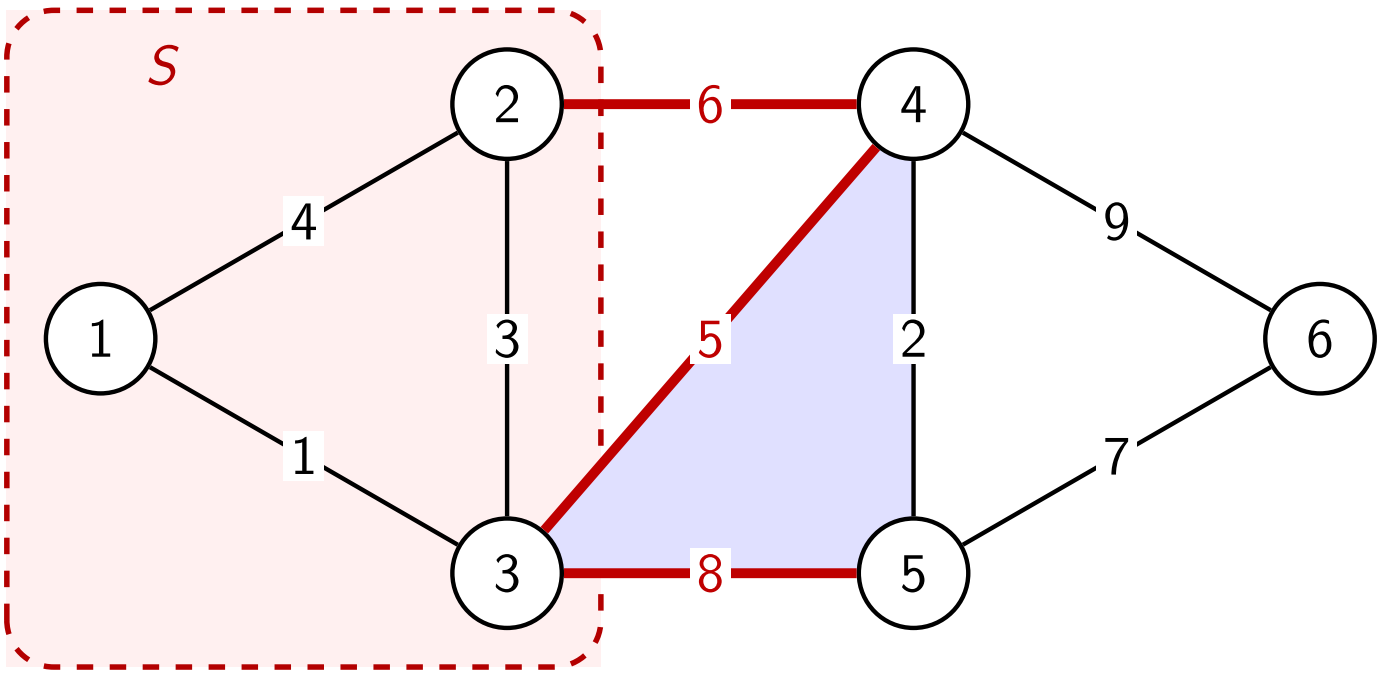
Parte 2.

- Gli estremi di e stanno da parti diverse del taglio fondamentale $\delta(S_f)$
- Il cammino fra loro in T fa parte di $C(T, e)$, e deve attraversare $\delta(S_f)$ con uno spigolo di T
- L'unico è f, quindi $f \in C(T, e)$
- La parte 1 conclude

□

Esempio: tagli e cicli del grafo di riferimento (1/2)

La figura mostra il taglio $\delta(\{1,2,3\})$ (spigoli rossi) e il ciclo $3,4,5$ (ombreggiato in blu).



Il ciclo usa gli spigoli $\{3,4\}$ e $\{3,5\}$ del taglio, un numero pari.

Esempio: tagli e cicli del grafo di riferimento (2/2)

Per l'albero ricoprente di costo minimo
$\tilde{F} = \big\{\{1,3\},\{2,3\},\{3,4\},\{4,5\},\{5,6\}\big\}$ del grafo di riferimento:

- il taglio fondamentale di $\{3,4\}$ è

$$\delta(\{1,2,3\})$$

- il ciclo fondamentale di $\{3,5\}$ è

$$C(\tilde{F},\{3,5\}) = 3,4,5,3$$

Tagli, cicli e scambi

Un ciclo attraversa ogni taglio un numero pari di volte.

Aggiungendo uno spigolo a un albero ricoprente e togliendo uno spigolo qualsiasi del ciclo che esso chiude, si ottiene un altro albero ricoprente.

Riepilogo

Tagli:

- $\delta(S)$: gli spigoli con esattamente un estremo in S; G è connesso se e solo se ogni taglio è non vuoto
- Il taglio fondamentale di $e \in T$ contiene e e nessun altro spigolo di T

Cicli e scambi:

- Un ciclo attraversa un taglio un numero pari di volte
- Lemma dello scambio: $T \cup \{e\} \setminus \{f\}$ è un albero ricoprente quando $f \in C(T, e)$

La proprietà del taglio e la proprietà del ciclo

Introduzione: la proprietà del taglio e la proprietà del ciclo

- Due regole semplici dicono quali spigoli si possono prendere senza rischio e quali si possono lasciare fuori
- La **proprietà del taglio**: lo spigolo più leggero di un taglio; la **proprietà del ciclo**: lo spigolo più pesante di un ciclo
- Insieme danno le condizioni di ottimalità: un certificato che un albero ricoprente è ottimo, senza enumerare tutti gli alberi ricoprenti
- Tutti gli algoritmi del capitolo poggiano su questi risultati

Teorema 1: proprietà del taglio

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

(a) *$F \cup \{e\}$ è contenuto in qualche albero ricoprente di costo minimo.*

(b) *Se $w_e < w_f$ per ogni $f \in \delta(S) \setminus \{e\}$, allora e appartiene a ogni albero ricoprente minimo.*

La dimostrazione è uno scambio:
se un albero ottimo non contiene e, scambiamo e con un altro spigolo del taglio, e il peso non cresce.

Dimostrazione (1/2)

- Sia $T^\star$ un albero ricoprente di costo minimo con $F \subseteq T^\star$; per (b), sia $T^\star$ un albero ricoprente minimo qualsiasi e $F = \emptyset$
- Se $e \in T^\star$ non c'è niente da dimostrare; sia $e \notin T^\star$

1. Il ciclo fondamentale $C(T^\star, e)$ contiene lo spigolo e di $\delta(S)$, quindi contiene un altro spigolo $f \in \delta(S)$, $f \neq e$ (Lemma 1). Tutti i suoi spigoli diversi da e stanno in $T^\star$, quindi $f \in T^\star$; e $f \notin F$, perché $F \cap \delta(S) = \emptyset$.

Dimostrazione (2/2)

2. Per il Lemma 2, $T' = T^\star \cup \{e\} \setminus \{f\}$ è un albero ricoprente, con
$$w(T') = w(T^\star) + w_e - w_f$$
3. (a) e ha peso minimo in $\delta(S)$, quindi $w_e \leq w_f$ e $w(T') \leq w(T^\star)$: T' è un albero ricoprente minimo che contiene $F \cup \{e\}$.
4. (b) Qui $w_e < w_f$ e $w(T') < w(T^\star)$, contro l'ottimalità di $T^\star$: quindi $e \in T^\star$.

□

La proprietà del ciclo è l'immagine speculare della proprietà del taglio:
dice quali spigoli si possono lasciare fuori.

Teorema 2: proprietà del ciclo

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

(a) *Qualche albero ricoprente di costo minimo non contiene e.*

(b) *Se $w_e > w_f$ per ogni $f \in C \setminus \{e\}$, allora nessun albero ricoprente di costo minimo contiene e.*

Dimostrazione (1/2)

- Sia $T^\star$ un albero ricoprente di costo minimo con $e \in T^\star$ (se non ce n'è nessuno, valgono entrambe le affermazioni)
- Sia $\delta(S_e)$ il taglio fondamentale di e in $T^\star$

1. Il ciclo C contiene lo spigolo e di $\delta(S_e)$, quindi contiene un altro spigolo $f \in \delta(S_e)$ (Lemma 1); $f \notin T^\star$, perché e è l'unico spigolo di $T^\star$ nel suo taglio fondamentale.

Dimostrazione (2/2)

2. Per il Lemma 2(2), $T' = T^\star \cup \{f\} \setminus \{e\}$ è un albero ricoprente, con

$$w(T') = w(T^\star) + w_f - w_e$$

3. (a) $w_f \leq w_e$, quindi T' è un albero ricoprente di costo minimo senza e.

   (b) $w_f < w_e$ darebbe $w(T') < w(T^\star)$, una contraddizione: nessun albero ricoprente di costo minimo contiene e.

□

Esempio: le due proprietà sul grafo di riferimento (1/3)

Proprietà del taglio.

- Lo spigolo $\{3,4\}$, di peso 5, è l'unico spigolo più leggero di

$$\delta(\{1,2,3\}) = \{\{2,4\},\{3,4\},\{3,5\}\},$$

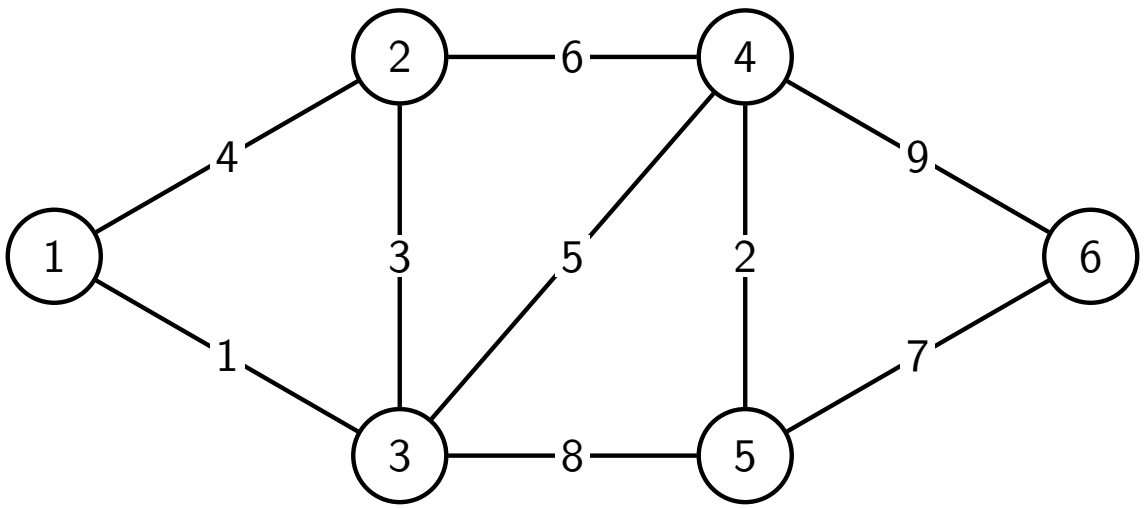
  con pesi $6,5,8$: appartiene a ogni albero ricoprente di costo minimo.

- Allo stesso modo, $\{5,6\}$ è l'unico spigolo più leggero di $\delta(\{6\})$.

Esempio: le due proprietà sul grafo di riferimento (2/3)

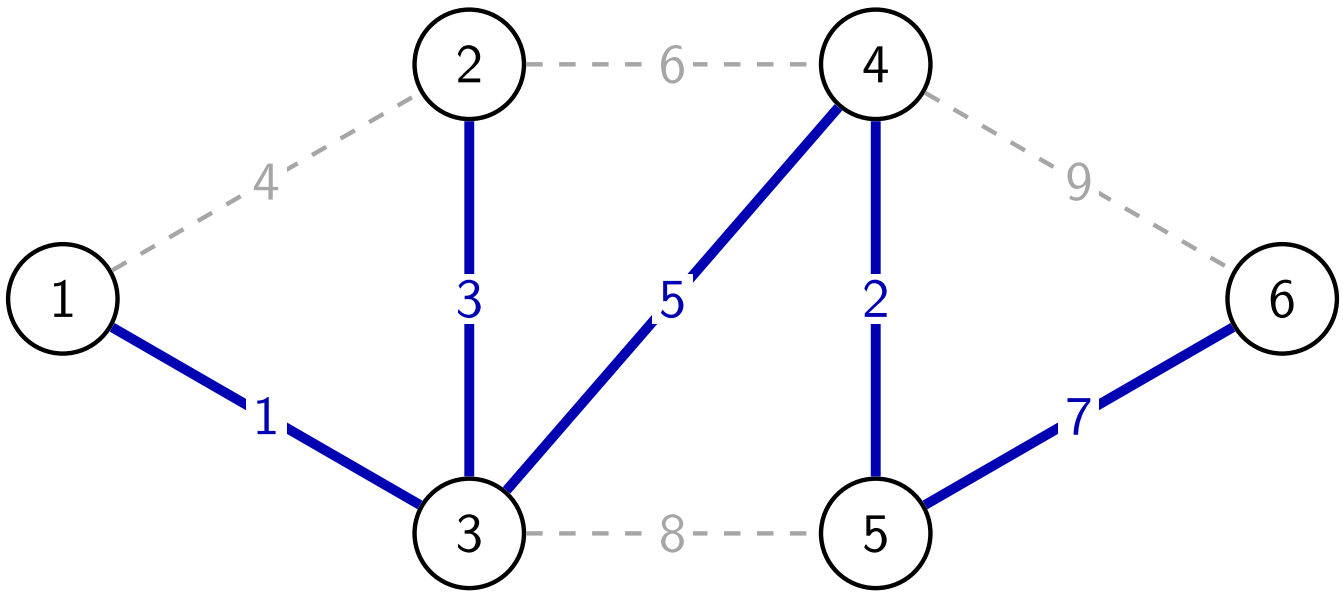
Proprietà del ciclo.

- Lo spigolo $\{1, 2\}$ è l'unico spigolo più pesante del ciclo $1, 2, 3$ (pesi $4, 3, 1$)
- $\{3, 5\}$ è l'unico spigolo più pesante del ciclo $3, 4, 5$ (pesi $5, 2, 8$)
- Nessun albero ricoprente di costo minimo li contiene
- I cicli $2, 3, 4$ e $4, 5, 6$ escludono $\{2, 4\}$ e $\{4, 6\}$



Esempio: le due proprietà sul grafo di riferimento (3/3)

- I quattro spigoli esclusi lasciano esattamente i 5 spigoli di $\tilde{F}$
- Le due proprietà da sole dimostrano che $\tilde{F}$ è l'unico albero ricoprente di costo minimo



Le due proprietà diventano un test di ottimalità:
per un albero dato, basta controllare i suoi cicli fondamentali,
o i suoi tagli fondamentali.

Teorema 3: condizioni di ottimalità

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

(i) *T è un albero ricoprente di costo minimo;*

(ii) ***condizione del ciclo**: per ogni $e \notin T$, $w_e \geq w_f$ per ogni spigolo f di $C(T, e)$;*

(iii) ***condizione del taglio**: per ogni $e \in T$, $w_e \leq w_f$ per ogni spigolo f del taglio fondamentale di e.*

Dimostrazione (1/3)

Dimostriamo (i)⇒(iii)⇒(ii)⇒(i).

- **(i)⇒(iii).** Se f attraversa il taglio fondamentale di $e \in T$ con $w_f < w_e$, allora $T \cup \{f\} \setminus \{e\}$ è un albero ricoprente (Lemma 2(2)) più leggero di T.

- **(iii)⇒(ii).** Siano $e \notin T$ e $f \in C(T, e)$, $f \neq e$. Il cammino di T fra gli estremi di e usa f, ed è unico, quindi togliere f li separa: e attraversa il taglio fondamentale di f, e $w_f \leq w_e$ per (iii).

Dimostrazione (2/3)

(ii)⇒(i). Fra gli alberi ricoprenti di costo minimo, sia $T^\star$ uno con il massimo numero di spigoli in comune con T, e supponiamo $T^\star \neq T$.

1. Prendiamo $e \in T^\star \setminus T$, e sia $\delta(S_e)$ il suo taglio fondamentale in $T^\star$.

   Il ciclo $C(T, e)$ contiene e, quindi contiene un altro spigolo $f \in \delta(S_e)$ (Lemma 1), con $f \in T$ e $f \notin T^\star$.

Dimostrazione (3/3)

2. Per il Lemma 2(2), $T' = T^\star \cup \{f\} \setminus \{e\}$ è un albero ricoprente, e $w_f \leq w_e$ per (ii) applicata a $e \notin T$. Quindi

$$w(T') \leq w(T^\star)$$

3. T' è un albero ricoprente di costo minimo con uno spigolo in più in comune con T: una contraddizione. Quindi $T = T^\star$.

□

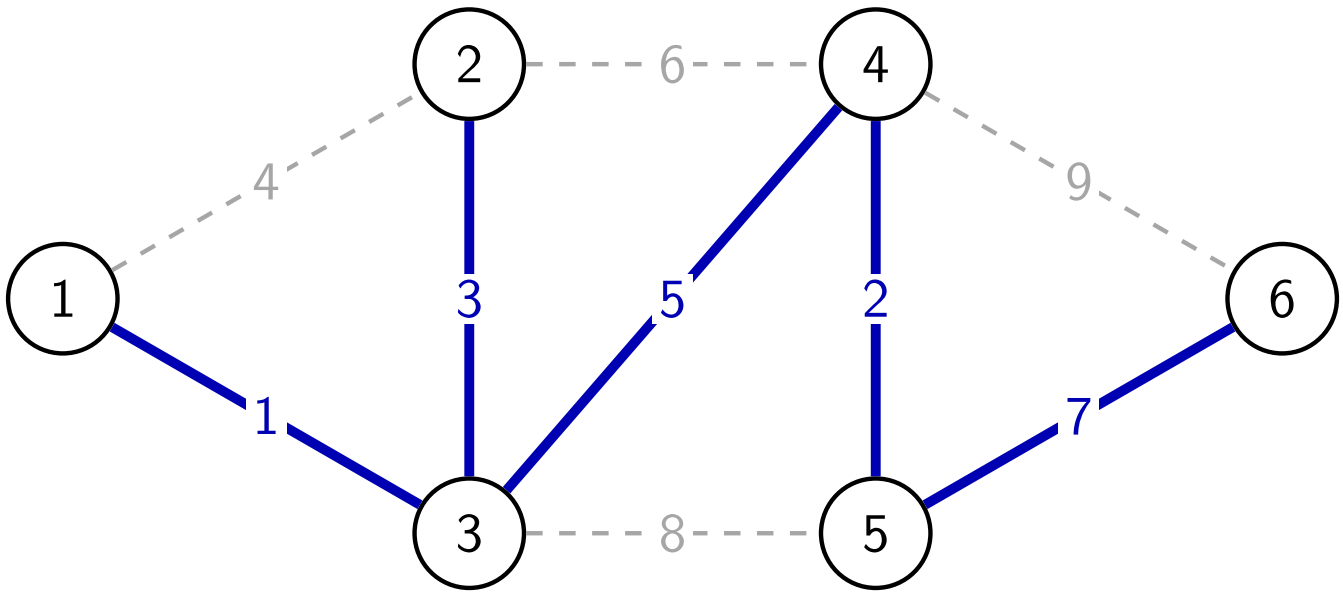
Esempio: certificare l'albero ottimo (1/2)

Per l'albero $\tilde{F}$ del grafo di riferimento, i quattro spigoli fuori dall'albero e i loro cicli fondamentali sono:

spigolo $e \notin \tilde{F}$	$C(\tilde{F}, e)$	pesi degli spigoli dell'albero nel ciclo
$\{1,2\}$, $w = 4$	$1,3,2,1$	$1,\ 3$
$\{2,4\}$, $w = 6$	$2,3,4,2$	$3,\ 5$
$\{3,5\}$, $w = 8$	$3,4,5,3$	$5,\ 2$
$\{4,6\}$, $w = 9$	$4,5,6,4$	$2,\ 7$

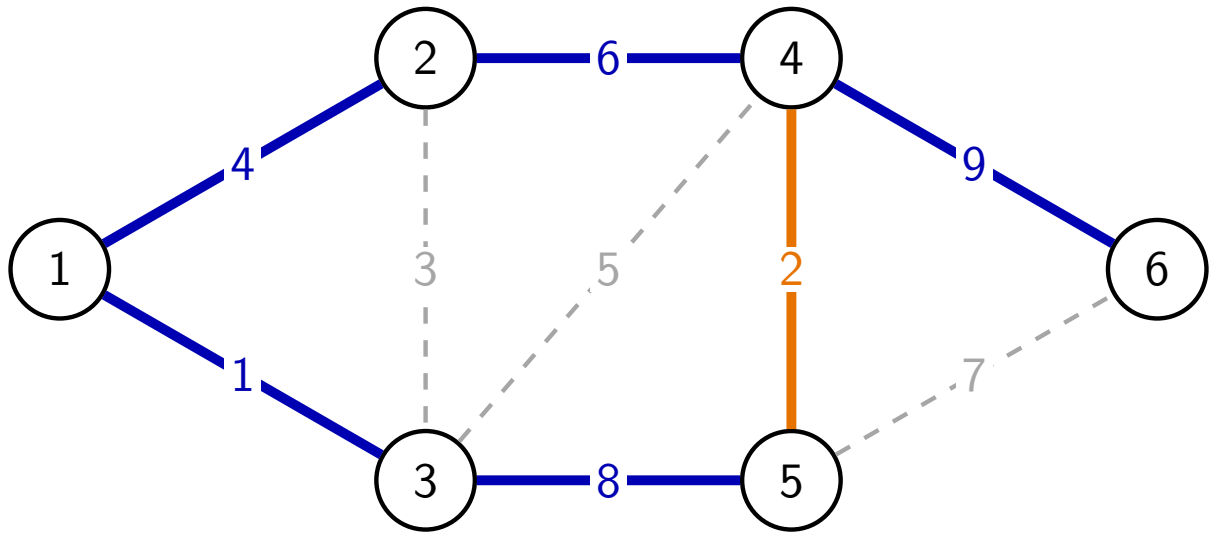
Esempio: certificare l'albero ottimo (2/2)

- Ogni spigolo fuori dall'albero è pesante almeno quanto gli spigoli dell'albero del suo ciclo
- Per il Teorema 3 l'albero è ottimo, senza enumerare i 55 alberi ricoprenti



Esempio: migliorare un albero che non è ottimo (1/2)

- L'albero della 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 $6,4,8$



in blu: l'albero della BFS T; in arancione: lo spigolo $e = \{4,5\}$

Esempio: migliorare un albero che non è ottimo (2/2)

- Togliendo il più pesante, $\{3, 5\}$, e aggiungendo $\{4, 5\}$ si ottiene un albero ricoprente di peso

$$28 + 2 - 8 = 22$$

- La dimostrazione del Teorema 3 diventa un algoritmo: ripetere lo scambio finché uno spigolo viola la condizione

- Gli algoritmi delle prossime sezioni sono più veloci

Taglio, ciclo e ottimalità

Lo spigolo più leggero di un taglio appartiene a qualche albero ricoprente di costo minimo, e lo spigolo più pesante di un ciclo si può lasciare fuori.

Un albero ricoprente è ottimo se e solo se ogni spigolo fuori dall'albero è il più pesante del suo ciclo fondamentale.

Riepilogo

Proprietà del taglio e di ciclo:

- Lo spigolo più leggero di un taglio non attraversato da F estende F dentro qualche albero ricoprente di costo minimo
- Lo spigolo più pesante di un ciclo si può lasciare fuori; se è strettamente il più pesante, non sta in nessun ottimo

Condizioni di ottimalità:

- Ottimo $\iff$ ogni spigolo fuori da T è il più pesante del suo ciclo fondamentale
- $\iff$ ogni spigolo dell'albero è il più leggero del suo taglio fondamentale

L'algoritmo greedy generico

Introduzione: l'algoritmo greedy generico

- La proprietà del taglio diventa direttamente un algoritmo
- Descriviamo un algoritmo greedy generico e dimostriamo che restituisce sempre un albero ricoprente di costo minimo
- Gli algoritmi di Kruskal e di Prim sono due casi particolari: dimostrare corretto quello generico li dimostra corretti entrambi

L'idea

- La proprietà del taglio suggerisce di costruire un albero ricoprente di costo minimo uno spigolo alla volta
- L'insieme corrente F è sempre contenuto in qualche albero ricoprente di costo minimo
- Lo spigolo successivo è lo spigolo più leggero di un taglio non attraversato da F

Algorithm: Algoritmo greedy generico per l'MST

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

Output: un albero ricoprente di costo minimo F

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

Teorema 4: correttezza dell'algoritmo greedy generico

A ogni iterazione dell'algoritmo greedy generico esiste un taglio come nel Passo 2, ed è non vuoto, e F è contenuto in qualche albero ricoprente di costo minimo. Alla fine, F è un albero ricoprente di costo minimo.

Dimostrazione (1/2)

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

1. Se $|F| < n - 1$, il grafo (V, F) è una foresta (un sottoinsieme di un albero) con $n - |F| \geq 2$ componenti.

   Sia S una di esse: nessuno spigolo di F attraversa $\delta(S)$, e $\delta(S) \neq \emptyset$ perché G è connesso.

Dimostrazione (2/2)

2. Per la proprietà del taglio (Teorema 1(a)), il nuovo insieme $F \cup \{e\}$ è ancora contenuto in qualche albero ricoprente di costo minimo.

3. Quando $|F| = n - 1$, F è contenuto in un albero ricoprente di costo minimo $T^\star$ con $n - 1$ spigoli, quindi $F = T^\star$.

□

Osservazioni

- Lo spigolo aggiunto $e = \{u, v\}$ ha $u \in S$ e $v \notin S$, e nessuno spigolo di F attraversa $\delta(S)$
- Quindi u e v stanno in componenti diverse di (V, F), ed e non chiude un ciclo
- Gli algoritmi di Kruskal e di Prim sono due modi di scegliere il taglio nel Passo 2

Il principio greedy

Aggiungendo $n - 1$ volte lo spigolo più leggero di un taglio non attraversato dagli spigoli correnti si costruisce un albero ricoprente minimo.

Riepilogo

Algoritmo greedy generico:

- Parte da $F = \emptyset$ e aggiunge $n - 1$ volte lo spigolo più leggero di un taglio non attraversato da F
- Un taglio adatto esiste sempre: una componente della foresta (V, F)

Correttezza:

- Per la proprietà del taglio, F resta sempre dentro qualche albero ricoprente di costo minimo
- Kruskal e Prim sono due modi di scegliere il taglio

Algoritmo di Kruskal

Introduzione: algoritmo di Kruskal

- L'algoritmo di Kruskal esamina gli spigoli dal più leggero al più pesante, e tiene uno spigolo ogni volta che non chiude un ciclo
- Dimostriamo che è un caso particolare dell'algoritmo greedy generico
- Con una struttura union–find richiede tempo $O(m \log n)$
- Lo eseguiamo passo per passo sul grafo di riferimento pesato

Algorithm: Algoritmo di Kruskal

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

Output: un albero ricoprente di costo minimo F

1 **Passo 0 (inizializzazione).** Ordina gli spigoli per peso non decrescente, $w_{e_1} \leq w_{e_2} \leq \cdots \leq w_{e_m}$, e poni $F := \emptyset$.

2 **Passo 1 (scansione).** Per $k = 1, 2, \ldots, m$: se gli estremi di e_k stanno in componenti diverse di (V, F), poni $F := F \cup \{e_k\}$; altrimenti rifiuta e_k.

3 Se $|F| = n - 1$, `STOP`.

Spigoli accettati e spigoli rifiutati

- Uno spigolo è **accettato** se unisce due componenti diverse di (V, F), e **rifiutato** se chiuderebbe un ciclo
- Le componenti di (V, F) partono come gli n vertici isolati e si fondono a ogni accettazione
- In questo modulo, gli spigoli di peso uguale si esaminano in ordine lessicografico di $\{u, v\}$, con $u < v$

Teorema 5: correttezza dell'algoritmo di Kruskal

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

- L'idea: ogni spigolo accettato è una scelta del Passo 2 dell'algoritmo greedy generico

Dimostrazione (1/3)

- Mostriamo che ogni spigolo accettato $e_k = \{u, v\}$ è una scelta del Passo 2 dell'algoritmo greedy generico
- Sia S la componente di (V, F) che contiene u, subito prima che e_k sia accettato
- Allora $v \notin S$, e nessuno spigolo di F attraversa $\delta(S)$

Dimostrazione (2/3)

1. Sia $e_h \in \delta(S)$ con $h < k$. Quando e_h è stato esaminato non è stato accettato, perché $F \cap \delta(S) = \emptyset$; quindi i suoi estremi stavano allora nella stessa componente, e ci stanno ancora, perché le componenti si fondono soltanto. Questo contraddice $e_h \in \delta(S)$.

2. Quindi ogni spigolo di $\delta(S)$ è un e_h con $h \geq k$, e $w_{e_h} \geq w_{e_k}$: e_k ha peso minimo in $\delta(S)$.

Dimostrazione (3/3)

3. Se la scansione finisse con $|F| < n - 1$, la foresta (V, F) avrebbe due componenti unite da uno spigolo di G, e quel spigolo sarebbe stato accettato quando è stato esaminato, per l'argomento del punto 1.

Per il Teorema 4, F è un albero ricoprente di costo minimo. □

La struttura union–find

- Le componenti di (V, F) sono mantenute in una struttura **union–find**:
  - ogni vertice v memorizza l'etichetta $comp(v)$ della sua componente;
  - ogni componente memorizza la lista dei suoi vertici e la sua dimensione.

- Verificare se u e v stanno in componenti diverse è il confronto

$$comp(u) \neq comp(v),$$

  in tempo $O(1)$
- Fondere due componenti rietichetta i vertici della *più piccola* (**unione per dimensione**)

Teorema 6: tempo di esecuzione dell'algoritmo di Kruskal

Con la struttura union–find per etichette e l'unione per dimensione, l'algoritmo di Kruskal richiede tempo $O(m \log n)$.

Dimostrazione (1/2)

1. Ordinare gli spigoli richiede tempo $O(m \log m)$; poiché $m \leq n^2$, $\log m \leq 2 \log n$ e questo è $O(m \log n)$.

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

3. Quando un vertice viene rietichettato, la sua componente si fonde con una almeno altrettanto grande, quindi la dimensione della sua componente almeno raddoppia.

Dimostrazione (2/2)

- La dimensione non supera mai n, quindi ogni vertice viene rietichettato al più $\log_2 n$ volte: tutte le fusioni richiedono tempo $O(n \log n)$
- G è connesso, quindi $n - 1 \leq m$, e il totale è

$$O(m \log n + n \log n) = O(m \log n)$$

□

Osservazioni sul tempo di esecuzione

- Il tempo di esecuzione è dominato dall'ordinamento
- Con la struttura union–find ad alberi, con unione per rango e compressione dei cammini, la scansione richiede un tempo quasi lineare; il limite resta $O(m \log n)$ a causa dell'ordinamento
- L'algoritmo usa solo confronti fra pesi: ogni trasformazione crescente dei pesi dà lo stesso albero

Esempio: algoritmo di Kruskal (1/10)

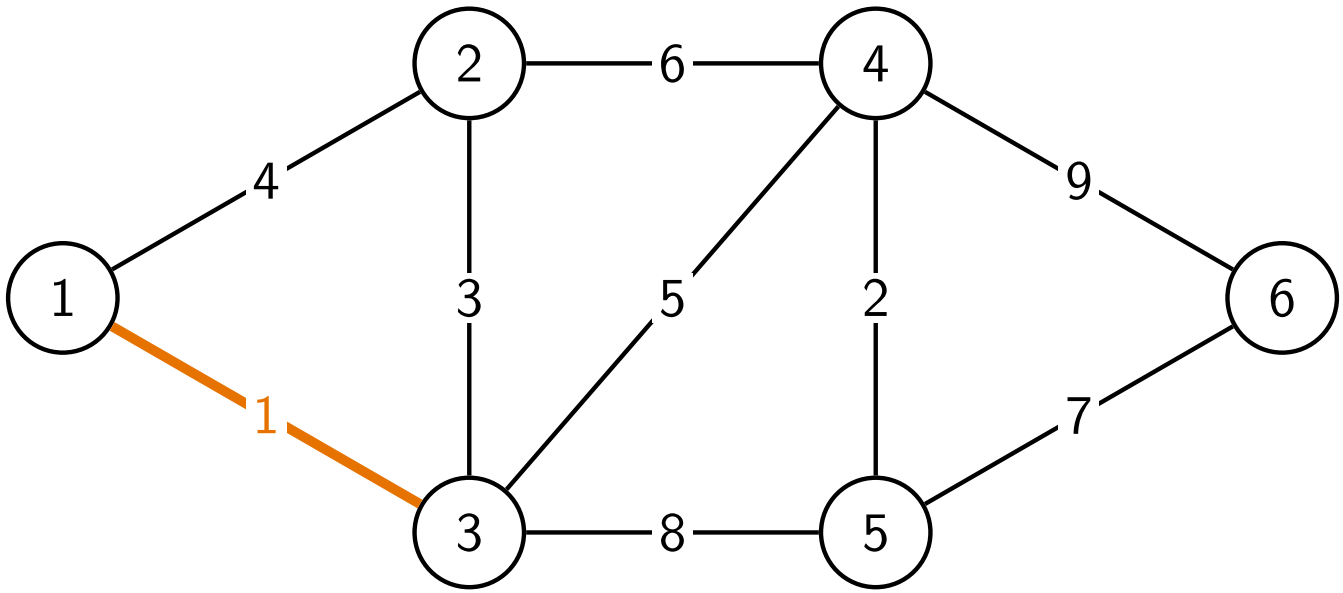
- Eseguiamo l'algoritmo di Kruskal sul grafo di riferimento pesato
- Gli spigoli in ordine di peso sono

$$\{1,3\},\{4,5\},\{2,3\},\{1,2\},\{3,4\},\{2,4\},\{5,6\},\{3,5\},\{4,6\}$$

- Nelle prossime slide, una slide per ogni spigolo esaminato:
  - **in blu**: gli spigoli accettati, in F;
  - **in arancione**: lo spigolo esaminato ora;
  - tratteggiati in grigio: gli spigoli rifiutati.

Esempio: algoritmo di Kruskal (2/10)

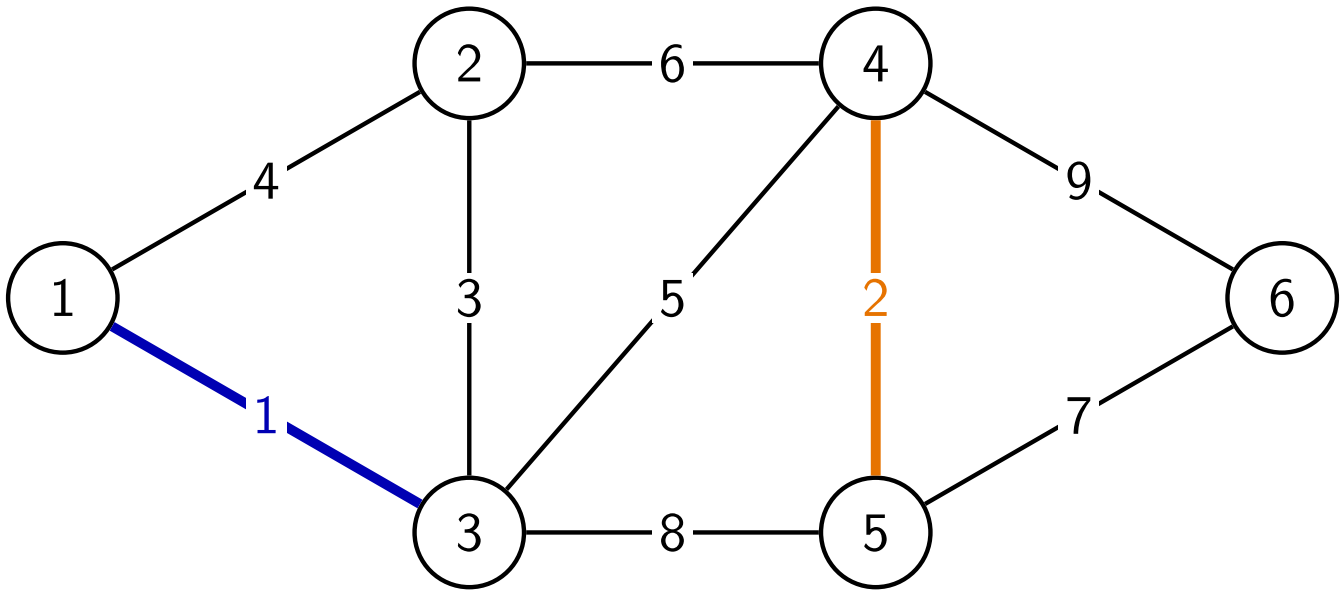
$k = 1$: spigolo $\{1, 3\}$, $w = 1$. **Accetta**.



Componenti di (V, F): $\{1, 3\}, \{2\}, \{4\}, \{5\}, \{6\}$

Esempio: algoritmo di Kruskal (3/10)

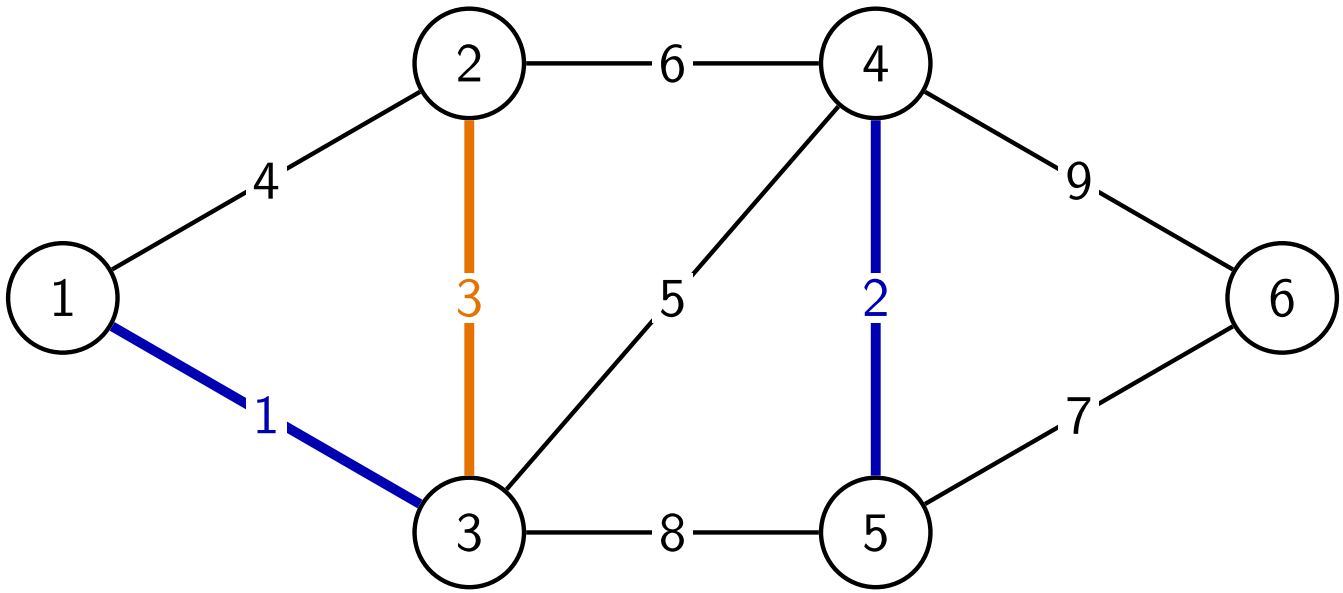
$k = 2$: spigolo $\{4, 5\}$, $w = 2$. **Accetta**.



Componenti di (V, F): $\{1, 3\}, \{2\}, \{4, 5\}, \{6\}$

Esempio: algoritmo di Kruskal (4/10)

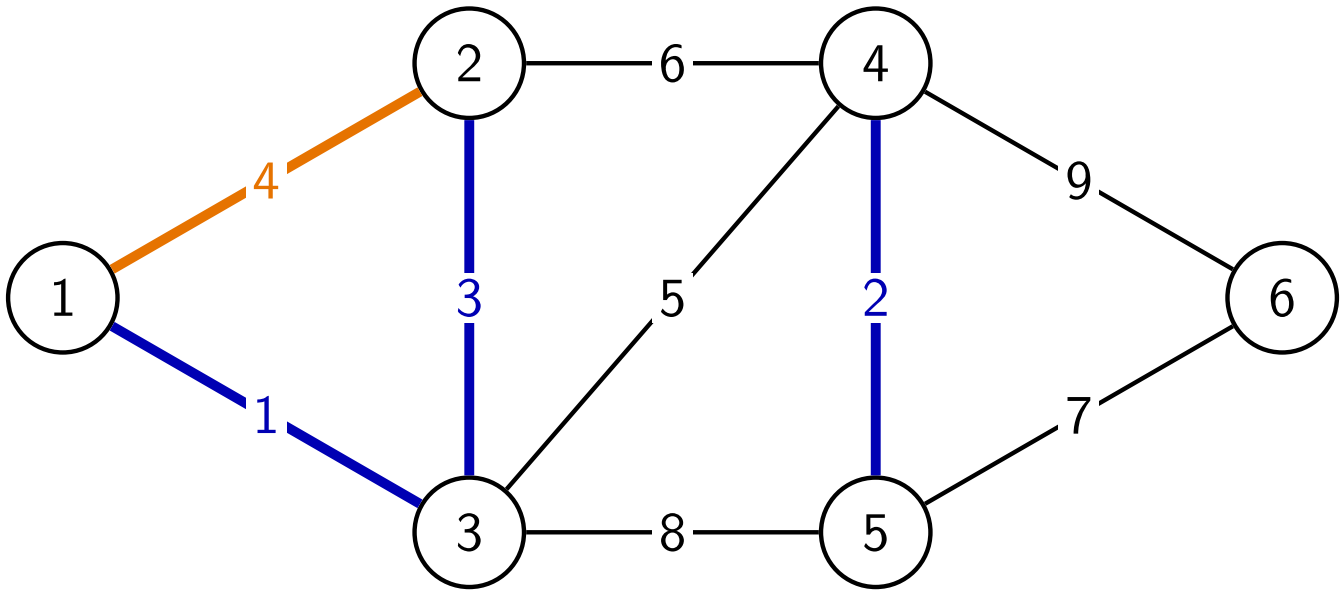
$k = 3$: spigolo $\{2, 3\}$, $w = 3$. **Accetta**.



Componenti di (V, F): $\{1, 2, 3\}, \{4, 5\}, \{6\}$

Esempio: algoritmo di Kruskal (5/10)

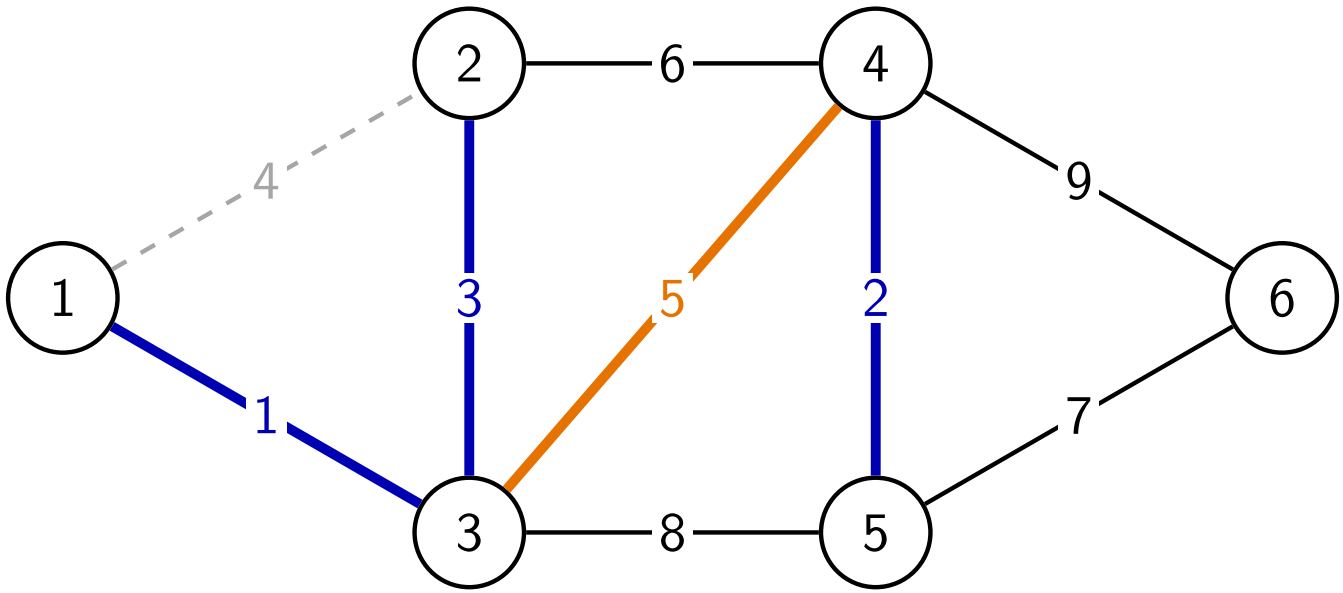
$k = 4$: spigolo $\{1, 2\}$, $w = 4$. **Rifiuta**: 1 e 2 stanno entrambi in $\{1, 2, 3\}$.



Componenti di (V, F): $\{1, 2, 3\}, \{4, 5\}, \{6\}$

Esempio: algoritmo di Kruskal (6/10)

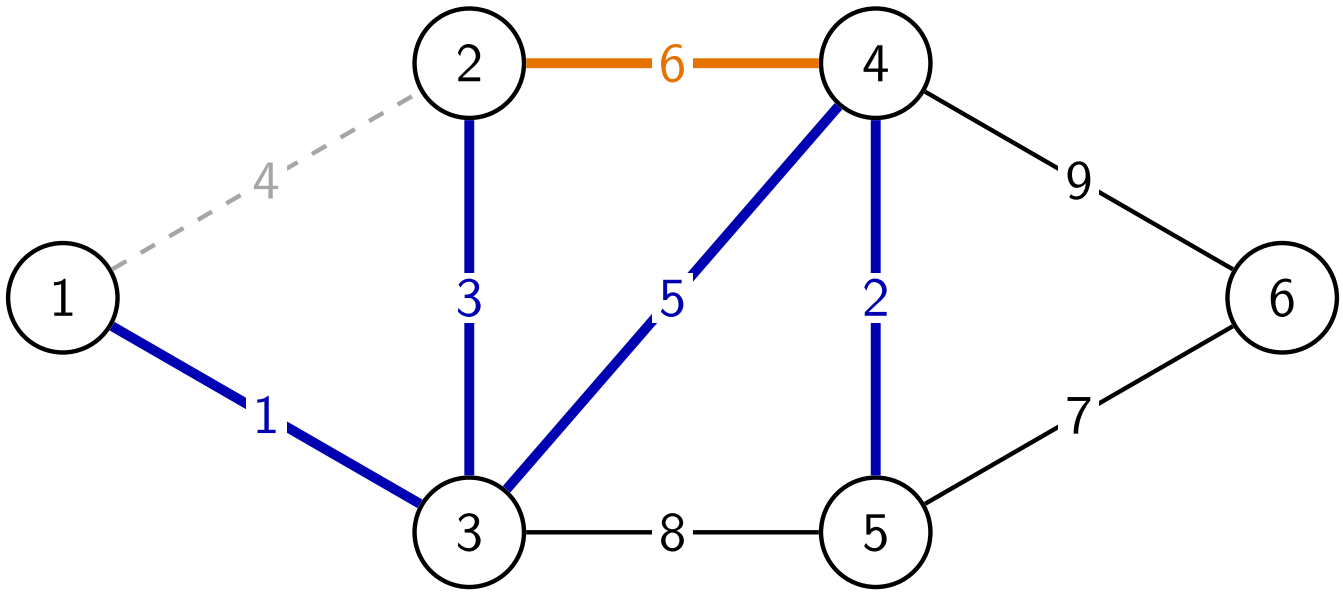
$k = 5$: spigolo $\{3, 4\}$, $w = 5$. **Accetta**.



Componenti di (V, F): $\{1, 2, 3, 4, 5\}, \{6\}$

Esempio: algoritmo di Kruskal (7/10)

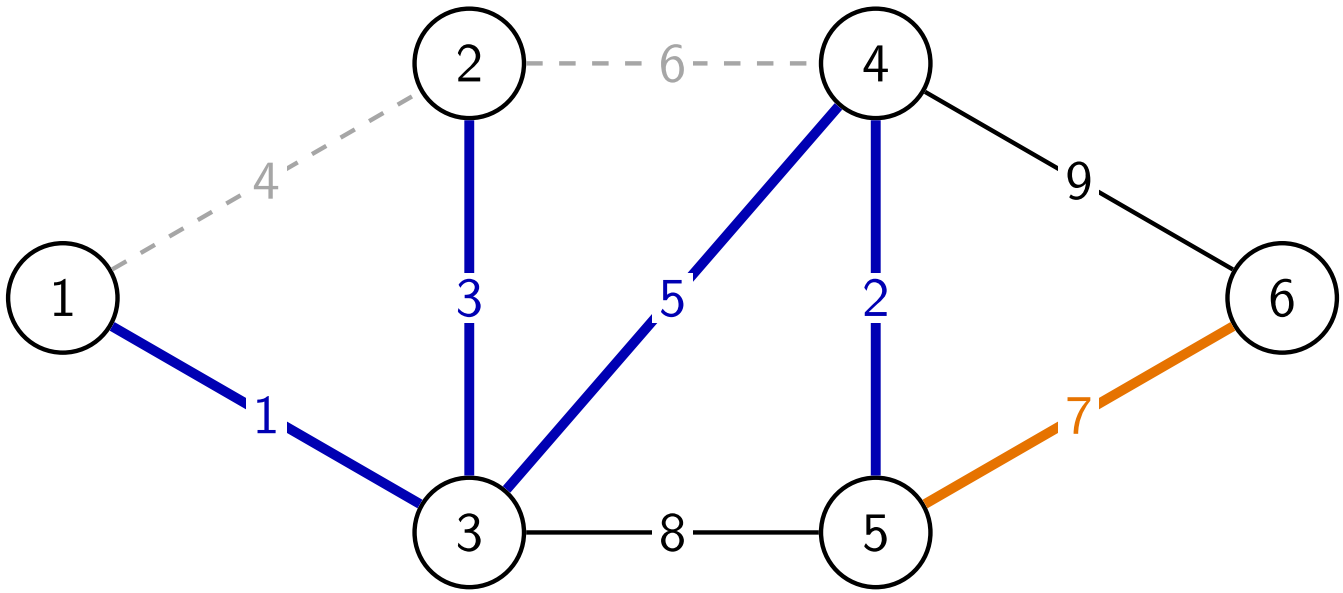
$k = 6$: spigolo $\{2, 4\}$, $w = 6$. **Rifiuta**: 2 e 4 stanno entrambi in $\{1, 2, 3, 4, 5\}$.



Componenti di (V, F): $\{1, 2, 3, 4, 5\}, \{6\}$

Esempio: algoritmo di Kruskal (8/10)

$k = 7$: spigolo $\{5, 6\}$, $w = 7$. **Accetta**: $|F| = 5$, `STOP`.



Componenti di (V, F): $\{1, 2, 3, 4, 5, 6\}$

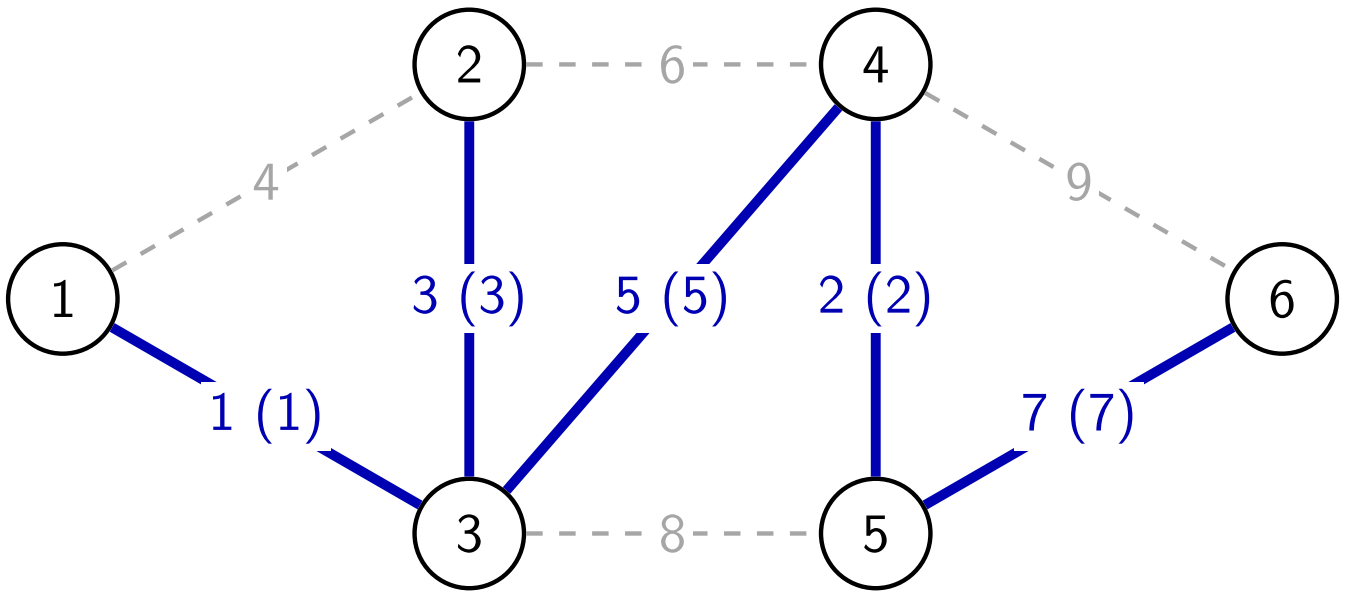
Esempio: algoritmo di Kruskal (9/10)

Ogni riga è uno spigolo esaminato, con le componenti di (V, F) dopo il passo:

k	spigolo 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 stanno entrambi in $\{1,2,3\}$)
5	$\{3,4\}$	5	accetta: $\{1,2,3,4,5\},\{6\}$
6	$\{2,4\}$	6	rifiuta (2 e 4 stanno entrambi in $\{1,2,3,4,5\}$)
7	$\{5,6\}$	7	accetta: $\{1,2,3,4,5,6\}$; $\lvert F\rvert = 5$, `STOP`

Esempio: algoritmo di Kruskal (10/10)

- L'algoritmo si ferma dopo 7 dei 9 spigoli, con $F = \{\{1,3\},\{4,5\},\{2,3\},\{3,4\},\{5,6\}\}$ di peso 18: l'albero $\tilde{F}$
- Su ogni spigolo dell'albero: il suo peso e, fra parentesi, il passo k in cui è accettato



Kruskal e la proprietà del ciclo

- Gli spigoli rifiutati $\{1, 2\}$ e $\{2, 4\}$ sono gli spigoli più pesanti dei cicli $1, 2, 3$ e $2, 3, 4$, come prevede la proprietà del ciclo
- Gli spigoli $\{3, 5\}$ e $\{4, 6\}$ non vengono mai esaminati

L'algoritmo di Kruskal in una frase

L'algoritmo di Kruskal esamina gli spigoli per peso non decrescente e accetta quelli che uniscono due componenti; con union–find richiede tempo $O(m \log n)$, dominato dall'ordinamento.

Riepilogo

Algoritmo di Kruskal:

- Ordina gli spigoli; accetta uno spigolo se unisce due componenti di (V, F), altrimenti lo rifiuta
- Ogni accettazione è una scelta dell'algoritmo greedy generico: l'algoritmo è corretto

Tempo di esecuzione:

- Union–find per etichette con unione per dimensione: $O(m \log n)$, dominato dall'ordinamento

Sul grafo di riferimento:

- Accetta $\{1,3\}, \{4,5\}, \{2,3\}, \{3,4\}, \{5,6\}$; rifiuta $\{1,2\}, \{2,4\}$

Algoritmo di Prim

Introduzione: algoritmo di Prim

- L'algoritmo di Prim fa crescere un solo albero da un vertice di partenza, aggiungendo a ogni passo lo spigolo più leggero che esce dall'albero
- Le etichette sui vertici rendono veloce la scelta di quello spigolo: tempo $O(n^2)$ in totale
- Lo dimostriamo corretto come caso particolare dell'algoritmo greedy generico, e lo eseguiamo sul grafo di riferimento pesato

L'idea

- L'algoritmo di Prim fa crescere un solo albero da un vertice di partenza s
- L'insieme S dei vertici dell'albero definisce il taglio $\delta(S)$ del Passo 2
- Lo spigolo più leggero del taglio si trova attraverso un'**etichetta** $d(v)$ per ogni vertice $v \notin S$

Algorithm: Algoritmo di Prim

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

Output : un albero ricoprente di costo minimo F

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

Convenzioni e significato delle etichette

- In questo modulo il vertice di partenza è $s = 1$, e le parità nel Passo 1 si risolvono a favore dell'indice più piccolo
- Il prossimo lemma dice che cosa sono le etichette: il peso dello spigolo più leggero da v all'albero corrente

Lemma 3: significato delle etichette

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

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

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

Dimostrazione

Per induzione.

- Alla prima esecuzione $S = \{s\}$, e il Passo 0 pone esattamente $d(v) = w_{sv}$, oppure $+\infty$
- Supponiamo l'affermazione vera prima dell'inserimento di un vertice v in S
- Per $u \notin S \cup \{v\}$, il nuovo insieme $S \cup \{v\}$ 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 gli altri predecessori non cambiano

□

Teorema 7: correttezza dell'algoritmo di Prim

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

Dimostrazione

Tutti gli spigoli di F hanno entrambi gli estremi in S, quindi $F \cap \delta(S) = \emptyset$.

1. Per il Lemma 3, $\min_{v \notin S} d(v)$ è il peso minimo di uno spigolo di $\delta(S)$, raggiunto 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 uno spigolo a F, quindi l'algoritmo si ferma dopo $n-1$ iterazioni con $|F| = n-1$; il Teorema 4 conclude.

□

Teorema 8: tempo di esecuzione dell'algoritmo di Prim

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

Dimostrazione

- Ci sono $n-1$ selezioni, e ciascuna scorre le etichette di al più n vertici: $O(n^2)$ in totale
- L'aggiornamento dopo l'inserimento di v scorre la lista di adiacenza di v in tempo $O(\deg(v))$, e

$$\sum_{v \in V} \deg(v) = 2m$$

- Il totale è $O(n^2 + m) = O(n^2)$, poiché $m \leq n^2$

□

Grafi densi e grafi sparsi

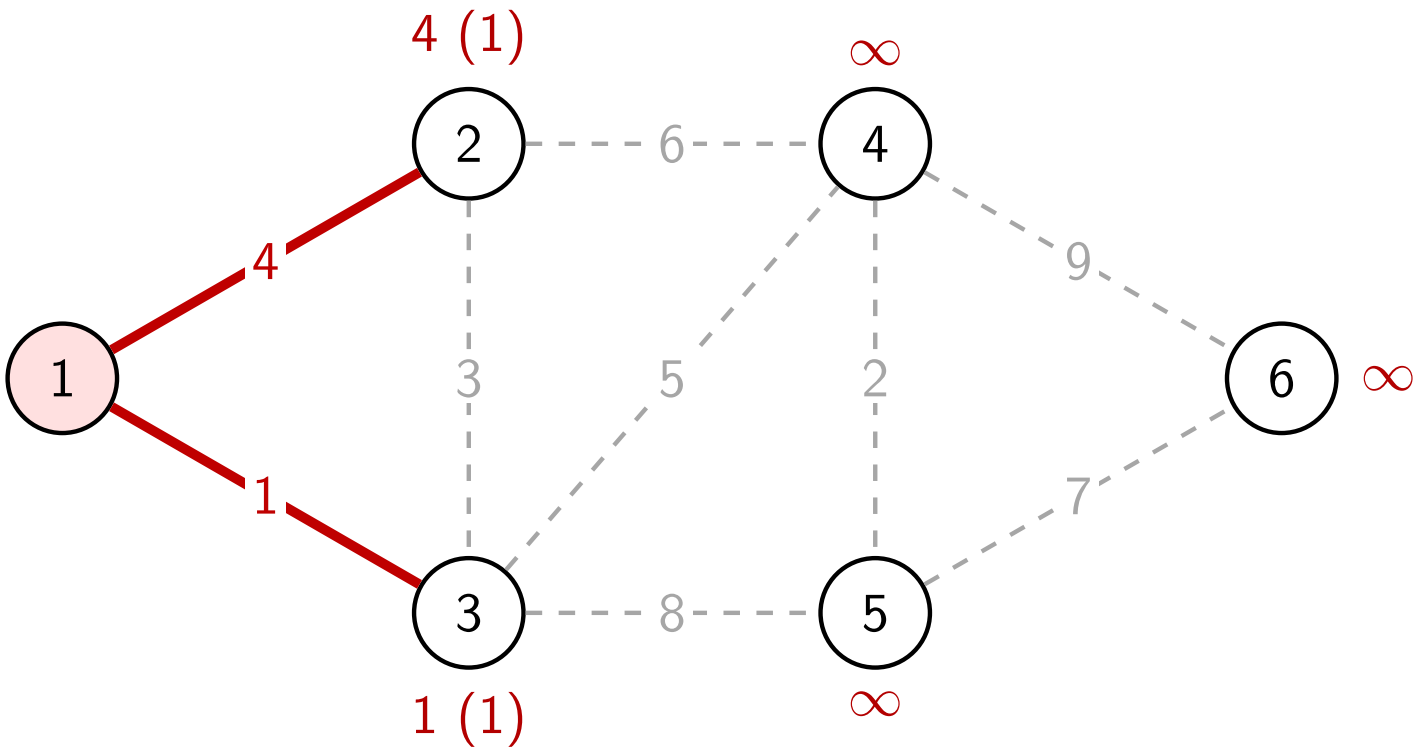
- Sui grafi densi, con m dell'ordine di n^2, il limite $O(n^2)$ è lineare nella dimensione del grafo
- Con le etichette in uno heap binario, ogni selezione e ogni aggiornamento richiedono tempo $O(\log n)$, per $O(m \log n)$ in totale: meglio sui grafi sparsi

Esempio: algoritmo di Prim dal vertice 1 (1/10)

- Eseguiamo l'algoritmo di Prim sul grafo di riferimento pesato da $s = 1$
- Nelle prossime slide, una slide per iterazione, dopo l'aggiornamento:
  - i vertici di S sono ombreggiati;
  - **in blu**: gli spigoli di F;
  - **in rosso**: gli spigoli del taglio $\delta(S)$;
  - accanto a ogni vertice $v \notin S$: la sua etichetta $d(v)$, con $pred(v)$ fra parentesi.

Esempio: algoritmo di Prim dal vertice 1 (2/10)

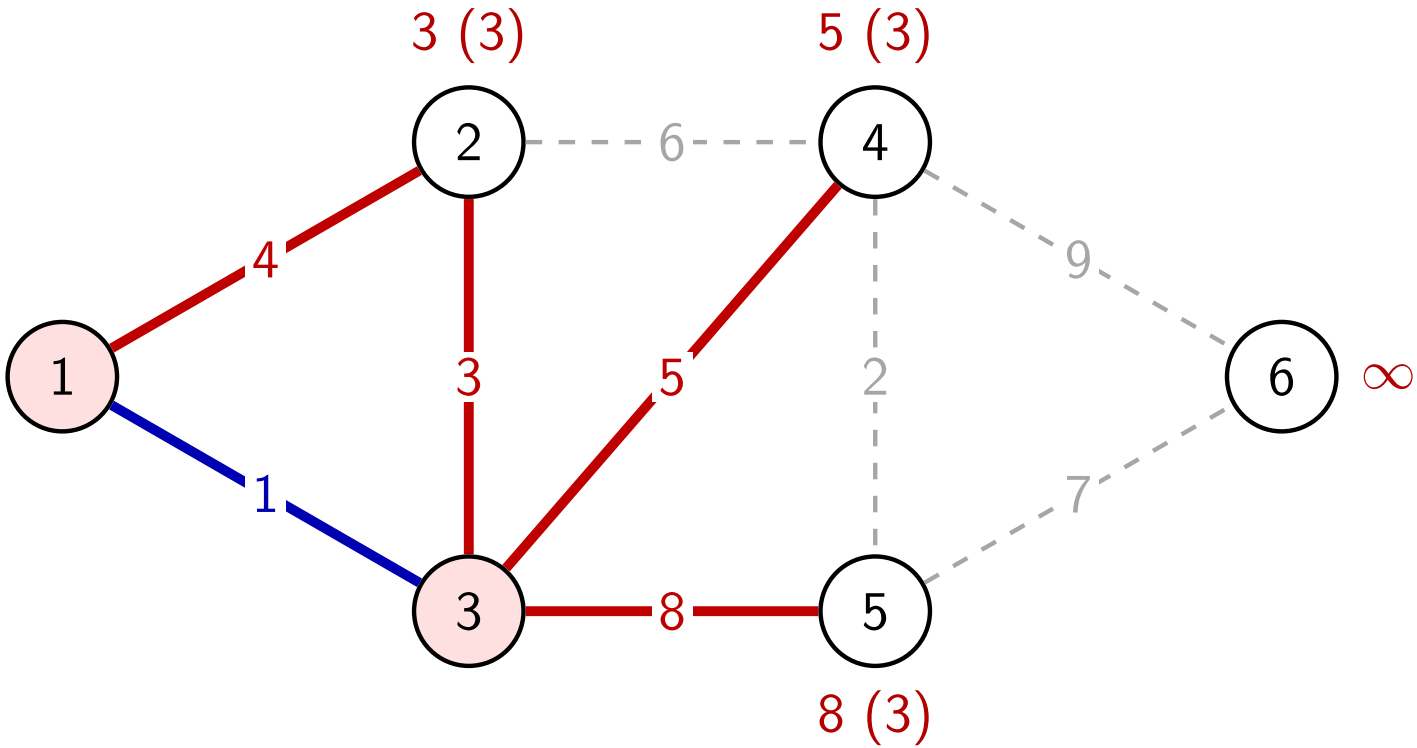
Iterazione 0: $S = \{1\}$, $F = \emptyset$.



Il prossimo vertice è 3, con $d(3) = 1$.

Esempio: algoritmo di Prim dal vertice 1 (3/10)

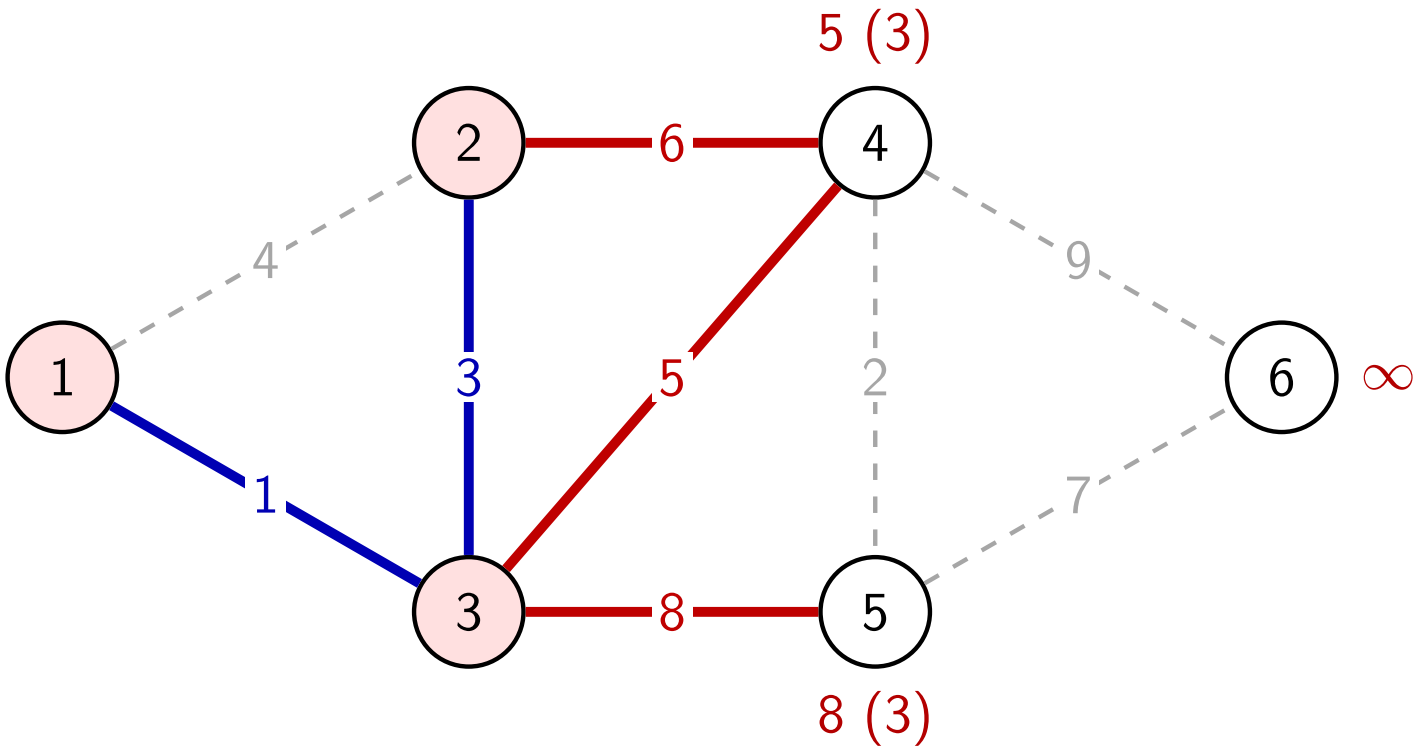
Iterazione 1: seleziona 3, aggiungi $\{1, 3\}$.



L'etichetta di 2 scende da 4 a 3, perché $w_{32} = 3 < 4$.

Esempio: algoritmo di Prim dal vertice 1 (4/10)

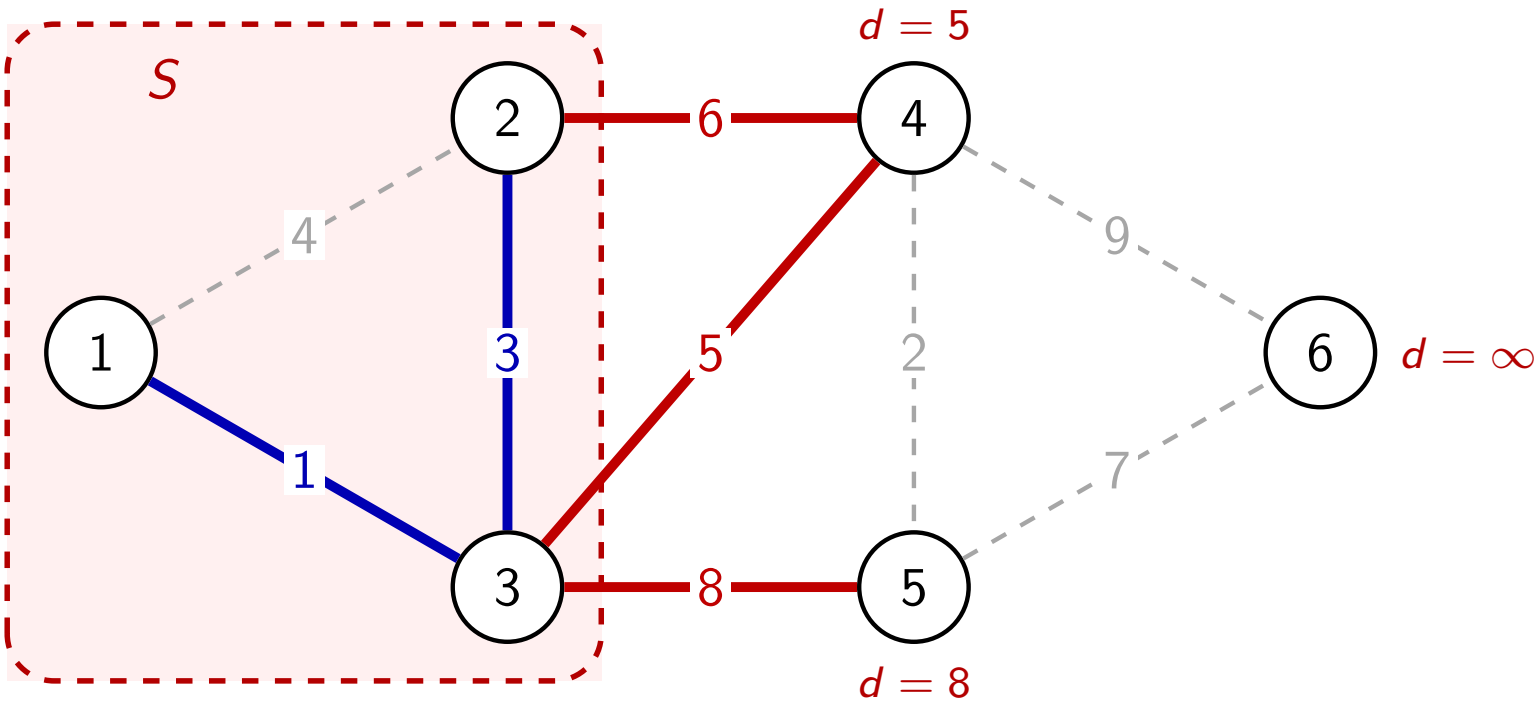
Iterazione 2: seleziona 2, aggiungi $\{2, 3\}$.



Nessuna etichetta cambia: $w_{24} = 6 > d(4) = 5$.

Esempio: algoritmo di Prim dal vertice 1 (5/10)

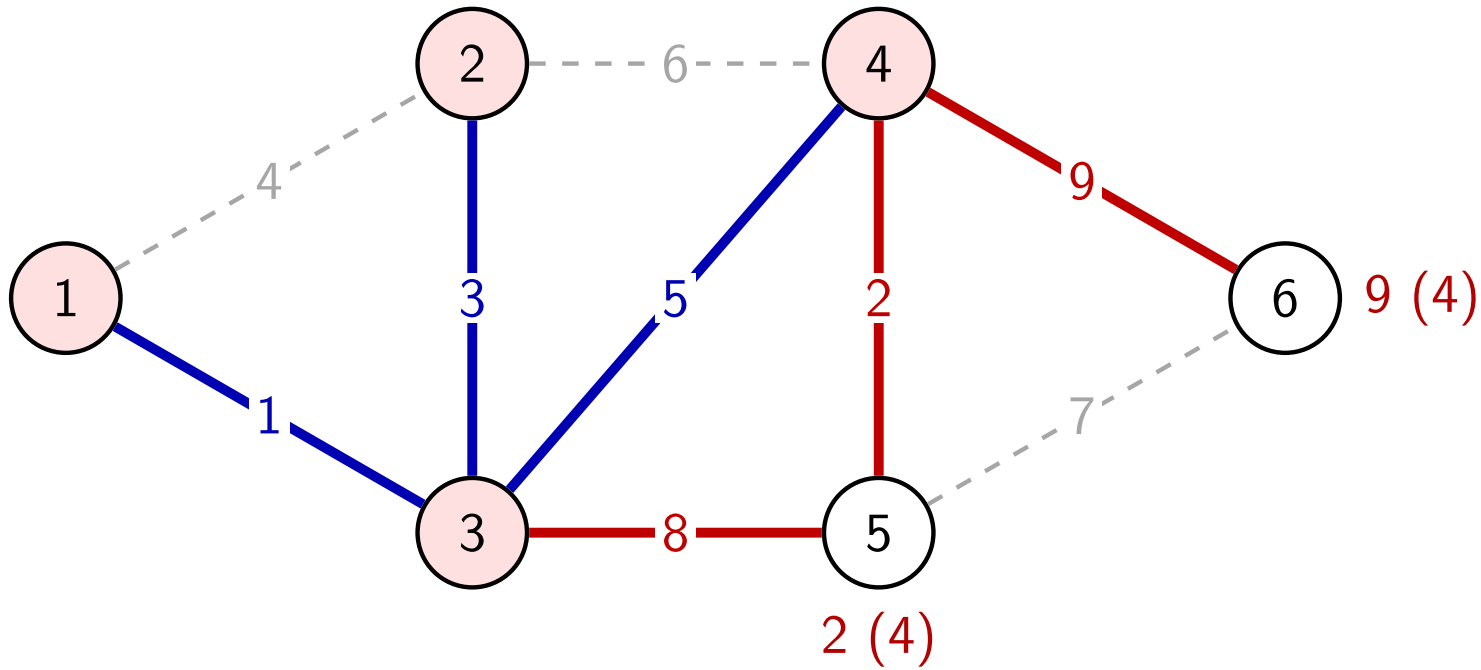
Lo stato dopo l'iterazione 2, con l'insieme S e il taglio $\delta(S)$:



Il taglio $\delta(\{1,2,3\})$ è attraversato da $\{2,4\}$, $\{3,4\}$, $\{3,5\}$, e il prossimo spigolo è $\{3,4\}$, con $d(4) = 5$.

Esempio: algoritmo di Prim dal vertice 1 (6/10)

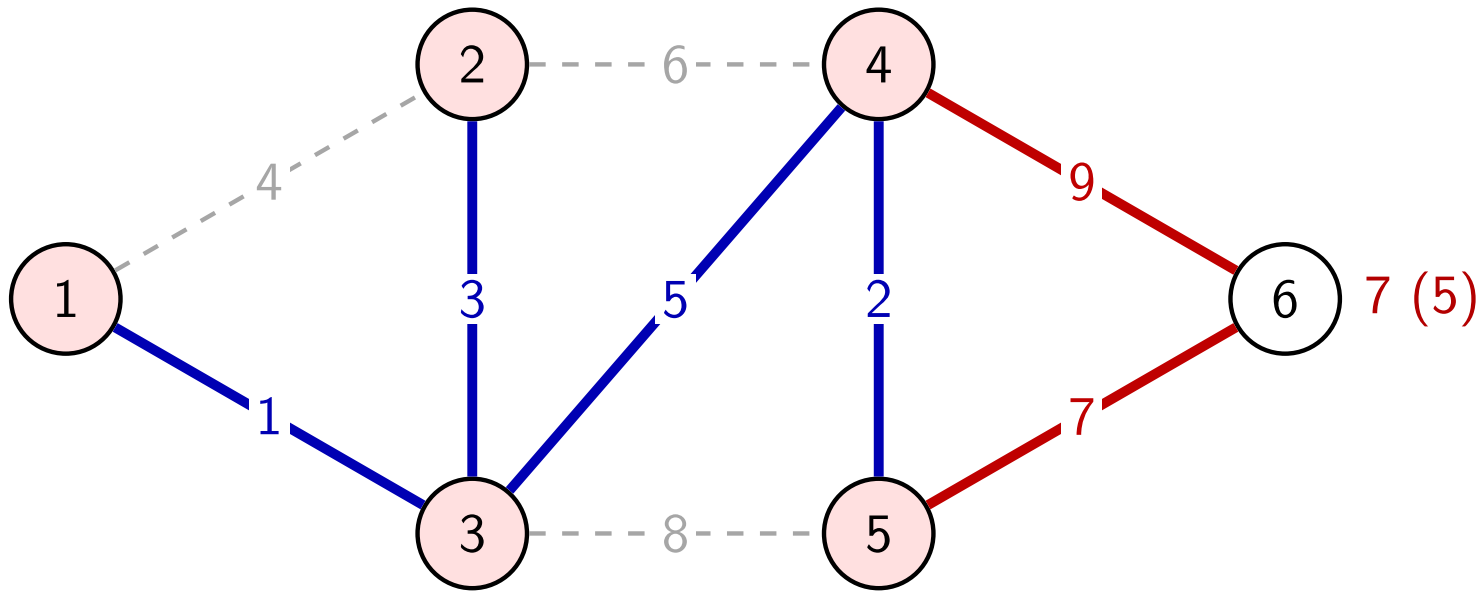
Iterazione 3: seleziona 4, aggiungi $\{3, 4\}$.



L'etichetta di 5 scende da 8 a 2.

Esempio: algoritmo di Prim dal vertice 1 (7/10)

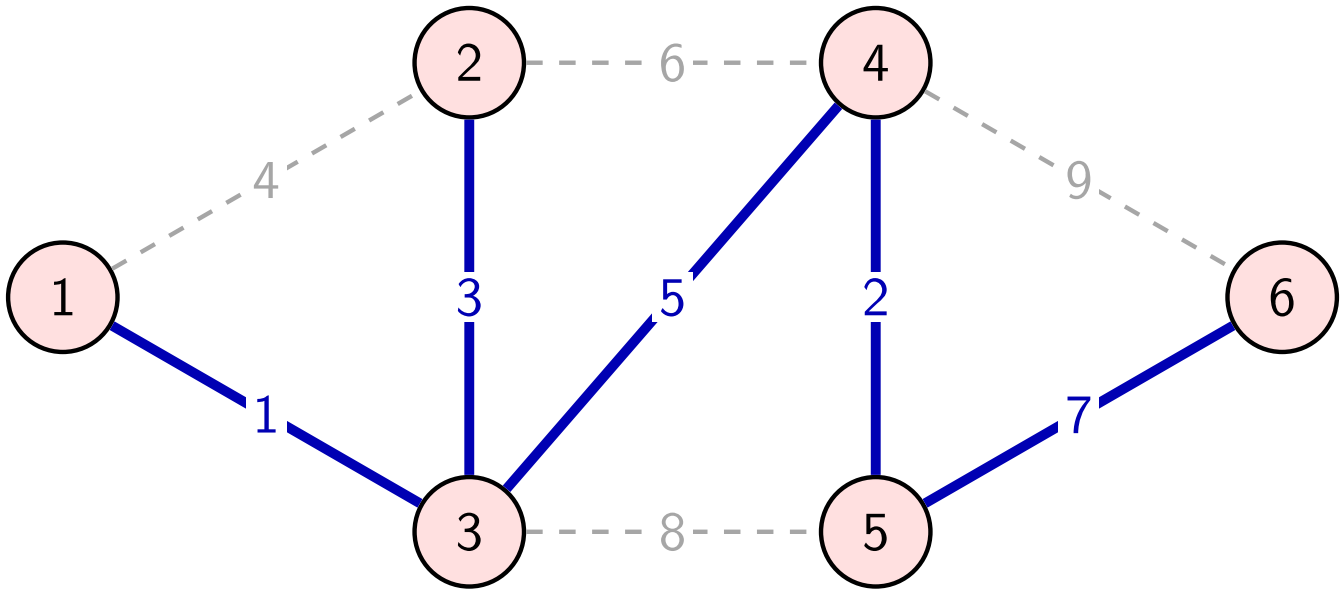
Iterazione 4: seleziona 5, aggiungi $\{4, 5\}$.



L'etichetta di 6 scende da 9 a 7.

Esempio: algoritmo di Prim dal vertice 1 (8/10)

Iterazione 5: seleziona 6, aggiungi $\{5, 6\}$. Ora $S = V$: `STOP`.



L'albero finale pesa 18.

Esempio: algoritmo di Prim dal vertice 1 (9/10)

Ogni riga: il vertice selezionato, lo spigolo aggiunto e le etichette dopo l'aggiornamento, con $pred(v)$ fra parentesi; un trattino indica i vertici già in S.

iterazione	selezionato	spigolo 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)	9 (4)
4	5	$\{4,5\}$	–	–	–	–	7 (5)
5	6	$\{5,6\}$	–	–	–	–	–

Esempio: algoritmo di Prim dal vertice 1 (10/10)

- L'albero finale è $\{1,3\},\{2,3\},\{3,4\},\{4,5\},\{5,6\}$, di peso 18
- È lo stesso albero dell'algoritmo di Kruskal, con gli spigoli aggiunti in un ordine diverso

Kruskal e Prim

- L'algoritmo di Kruskal aggiunge $\{4, 5\}$ per secondo, l'algoritmo di Prim lo aggiunge per quarto
- Kruskal fa crescere una foresta, Prim un solo albero che contiene s

L'algoritmo di Prim in una frase

L'algoritmo di Prim fa crescere un albero da s e aggiunge lo spigolo più leggero che ne esce; le etichette $d(v)$ danno quello spigolo in tempo $O(n)$, per $O(n^2)$ in totale.

Riepilogo

Algoritmo di Prim:

- Fa crescere un albero da s; il taglio è $\delta(S)$, con S i vertici dell'albero
- $d(v)$ è il peso dello spigolo più leggero da v a S, raggiunto da $\{pred(v), v\}$

Tempo di esecuzione:

- $O(n^2)$ con un array, adatto ai grafi densi; $O(m \log n)$ con uno heap binario

Sul grafo di riferimento:

- Da $s = 1$ seleziona $3, 2, 4, 5, 6$: lo stesso albero $\tilde{F}$ di Kruskal

Unicità dell'albero ricoprente di costo minimo

Introduzione: unicità dell'albero ricoprente di costo minimo

- Gli algoritmi restituiscono un albero ricoprente di costo minimo: ce ne sono altri?
- Con pesi distinti la risposta è no
- Con pesi uguali caratterizziamo gli spigoli che appartengono a tutti, o a nessuno, gli alberi ricoprenti di costo minimo, e stabiliamo se l'ottimo è unico

Teorema 9: pesi distinti

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

Dimostrazione (1/2)

- Supponiamo che $T_1 \neq T_2$ siano alberi ricoprenti di costo minimo
- Hanno lo stesso numero di spigoli, quindi la differenza simmetrica

$$(T_1 \setminus T_2) \cup (T_2 \setminus T_1)$$

è non vuota

- Sia e il suo spigolo più leggero, e diciamo $e \in T_1 \setminus T_2$

Dimostrazione (2/2)

1. Il ciclo $C(T_2, e)$ non è contenuto nell'aciclico T_1, quindi ha uno spigolo $f \notin T_1$; $f \neq e$, e $f \in T_2$, quindi $f \in T_2 \setminus T_1$.
2. f sta nella differenza simmetrica e $f \neq e$, quindi $w_f > w_e$, perché i pesi sono distinti.
3. $T_2 \cup \{e\} \setminus \{f\}$ è un albero ricoprente (Lemma 2) di peso

$$w(T_2) + w_e - w_f < w(T_2)$$

una contraddizione.

□

Conseguenze

- Con pesi distinti, l'algoritmo di Kruskal, l'algoritmo di Prim da qualsiasi vertice di partenza e ogni esecuzione dell'algoritmo greedy generico restituiscono 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 l'albero $\tilde{F}$

Quando alcuni pesi sono uguali, possono esistere più alberi ricoprenti di costo minimo.

I cicli e i tagli fondamentali di un albero ottimo ci dicono quali.

Proposizione 1: spigoli in tutti o in nessun albero ricoprente di costo minimo

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

(a) *Uno spigolo $e \in T$ appartiene a ogni albero ricoprente di costo minimo se e solo se $w_e < w_f$ per ogni $f \neq e$ nel taglio fondamentale di e.*

(b) *Uno spigolo $e \notin T$ non appartiene a nessun albero ricoprente di costo 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 ricoprente di costo minimo se e solo se ogni spigolo fuori da T è strettamente più pesante degli altri spigoli del suo ciclo fondamentale.

Dimostrazione (1/3)

(a)

- Se le disuguaglianze valgono, e sta in ogni albero ricoprente di costo minimo per la proprietà del taglio (Teorema 1(b))
- Altrimenti qualche $f \neq e$ nel taglio fondamentale ha $w_f \leq w_e$, e $w_f \geq w_e$ per la condizione di taglio (Teorema 3): quindi $w_f = w_e$
- $T \cup \{f\} \setminus \{e\}$ è un albero ricoprente (Lemma 2(2)) dello stesso peso, senza e

Dimostrazione (2/3)

(b)

- Se le disuguaglianze valgono, e non sta in nessun albero ricoprente di costo minimo per la proprietà del ciclo (Teorema 2(b))
- Altrimenti qualche $f \neq e$ in $C(T, e)$ ha $w_f = w_e$, per la condizione del ciclo
- $T \cup \{e\} \setminus \{f\}$ è un albero ricoprente di costo minimo che contiene e

Dimostrazione (3/3)

Per l'ultima affermazione:

- T è unico se e solo se nessuno spigolo fuori da T appartiene a un albero ricoprente di costo minimo
- cioè, (b) vale per ogni $e \notin T$

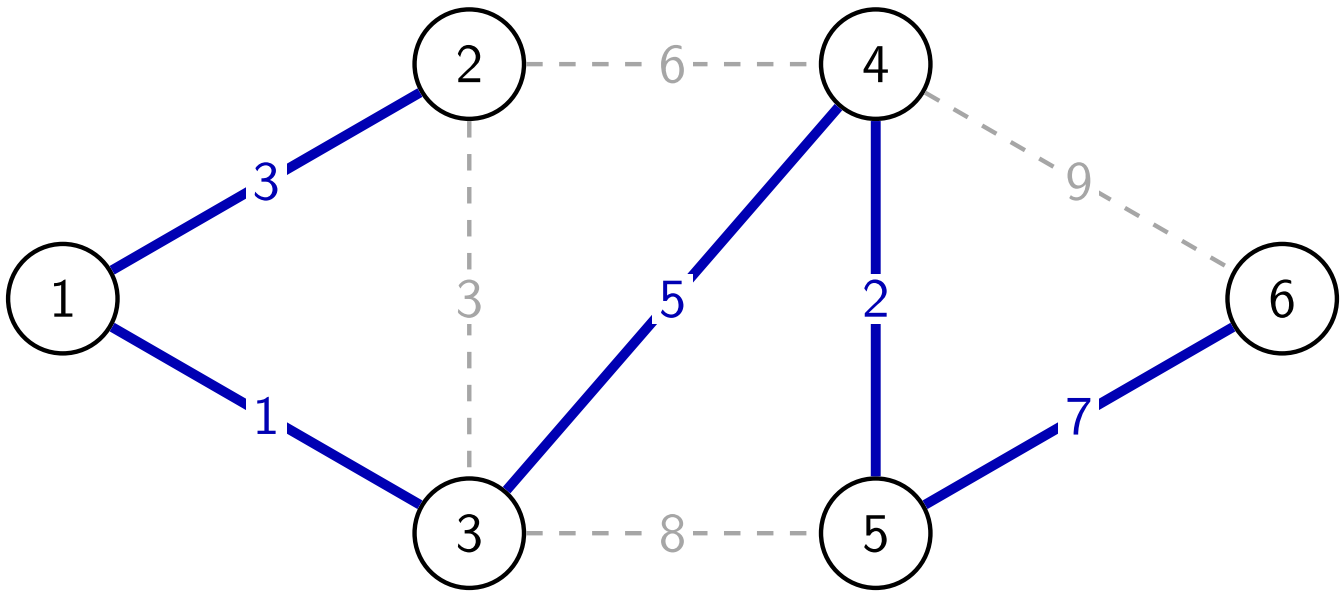
□

Esempio: pesi uguali sul grafo di riferimento (1/4)

- Sia $w_{12} = 3$ invece di 4, così che $w_{12} = w_{23} = 3$
- Il ciclo fondamentale di $\{1, 2\}$ in $\tilde{F}$ è $1, 3, 2, 1$, con $w_{23} = w_{12}$
- Per la Proposizione 1(b), anche l'albero $\{1, 2\}, \{1, 3\}, \{3, 4\}, \{4, 5\}, \{5, 6\}$ è ottimo, con peso 18

Esempio: pesi uguali sul grafo di riferimento (2/4)

Con $w_{12} = 3$, il secondo albero ricoprente di costo minimo $\{1,2\},\{1,3\},\{3,4\},\{4,5\},\{5,6\}$ (in blu), di peso 18:



Esempio: pesi uguali sul grafo di riferimento (3/4)

- L'enumerazione conferma che questi sono gli unici due alberi ricoprenti di costo minimo
- L'algoritmo di Kruskal, che esamina $\{1, 2\}$ prima di $\{2, 3\}$, restituisce quello con $\{1, 2\}$

Esempio: pesi uguali sul grafo di riferimento (4/4)

- I pesi uguali non creano sempre più alberi ottimi
- Con $w_{35} = w_{46} = 9$ l'albero $\tilde{F}$ è ancora l'unico albero ricoprente di costo minimo
- Ciascuno dei quattro spigoli fuori dall'albero resta strettamente il più pesante del suo ciclo

L'unicità in una frase

Con pesi distinti l'albero ricoprente di costo minimo è unico.

Con pesi uguali, un albero è l'unico ottimo se e solo se ogni spigolo fuori dall'albero è strettamente il più pesante del suo ciclo fondamentale.

Riepilogo

Pesi distinti:

- L'albero ricoprente di costo minimo è unico: tutti gli algoritmi restituiscono lo stesso albero

Pesi con valori uguali:

- Uno spigolo dell'albero sta in ogni ottimo se e solo se è strettamente il più leggero del suo taglio fondamentale
- Uno spigolo fuori dall'albero non sta in nessun ottimo se e solo se è strettamente il più pesante del suo ciclo fondamentale
- Le uguaglianze nei cicli fondamentali danno gli altri alberi ottimi

Errori tipici

Introduzione: errori tipici

- Sei errori frequenti quando si applicano i risultati e gli algoritmi di questo capitolo
- Riguardano la proprietà del taglio, i test di Kruskal e di Prim, l'unicità e i grafi orientati
- Controllarli è un modo rapido per non perdere punti

Osservazione 1: errori tipici

- ***Usare la proprietà del taglio con un taglio attraversato da F.*** *Lo spigolo più leggero di $\delta(S)$ è sicuro solo se nessuno spigolo già scelto attraversa $\delta(S)$.*
- ***Accettare uno spigolo nell'algoritmo di Kruskal perché tocca un vertice nuovo.*** *Il test è se i due estremi stanno in componenti diverse: $\{3, 4\}$ è accettato anche se 3 e 4 sono già entrambi coperti.*
- ***Aggiornare le etichette dell'algoritmo di Prim con lunghezze di cammini.*** *La nuova etichetta è il peso w_{vu} di un solo spigolo, non $d(v) + w_{vu}$: quello è l'algoritmo di Dijkstra per i cammini minimi.*

Osservazione: errori tipici (continua)

- ***Dimenticare che le etichette possono solo diminuire.*** *Nell'algoritmo di Prim $d(u)$ cambia solo quando $w_{vu} < d(u)$.*
- ***Dedurre l'unicità dall'algoritmo.*** *Kruskal e Prim restituiscono un albero; con pesi uguali possono esistere altri alberi ottimi, e si trovano attraverso le uguaglianze nei cicli fondamentali.*
- ***Applicare gli algoritmi ai grafi orientati.*** *L'albero ricoprente di costo minimo è definito sui grafi non orientati; in un grafo orientato l'orientamento degli archi si ignora.*

Riepilogo: errori tipici

- La proprietà del taglio vuole un taglio non attraversato da F
- Kruskal controlla le componenti, non i vertici nuovi
- Le etichette di Prim sono pesi di singoli spigoli, e possono solo diminuire
- Un albero restituito non vuol dire un ottimo unico
- Gli alberi ricoprenti di costo minimo si definiscono sui grafi non orientati

Riepilogo (1/3)

- Un albero ricoprente di costo minimo è un albero ricoprente di peso totale minimo di un grafo pesato connesso; ha $n - 1$ spigoli.
- Proprietà del taglio: lo spigolo più leggero di un taglio non attraversato da F estende F dentro qualche albero ricoprente minimo.
- Proprietà del ciclo: lo spigolo più pesante di un ciclo si può lasciare fuori, e se è strettamente il più pesante non sta in nessun ottimo.

Riepilogo (2/3)

- Un albero ricoprente è ottimo se e solo se ogni spigolo fuori dall'albero è il più pesante del suo ciclo fondamentale, o equivalentemente ogni spigolo dell'albero è il più leggero del suo taglio fondamentale.
- L'algoritmo di Kruskal ordina gli spigoli e accetta quelli che uniscono due componenti, in tempo $O(m \log n)$; l'algoritmo di Prim fa crescere un albero con le etichette $d(v)$, in tempo $O(n^2)$.
- Con pesi distinti l'albero ricoprente di costo minimo è unico.

Riepilogo (3/3)

Il grafo di riferimento pesato in sintesi.

vertici, spigoli, alberi ricoprenti	$n = 6$, $m = 9$, 55
albero ricoprente di costo minimo	$\{1,3\}, \{2,3\}, \{3,4\}, \{4,5\}, \{5,6\}$, peso 18 (unico)
Kruskal	accetta $\{1,3\}, \{4,5\}, \{2,3\}, \{3,4\}, \{5,6\}$; rifiuta $\{1,2\}, \{2,4\}$
Prim da 1	seleziona $3, 2, 4, 5, 6$