

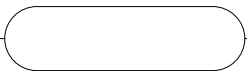
Teoria dei Grafi

Parte I

Alberto Caprara

DEIS - Università di Bologna

acaprara@deis.unibo.it

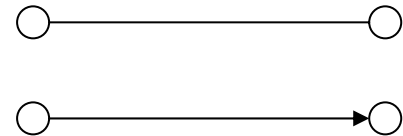


Teoria dei Grafi

- Paradigma di rappresentazione di problemi
- Grafo G : coppia (V,E)

V = insieme di vertici

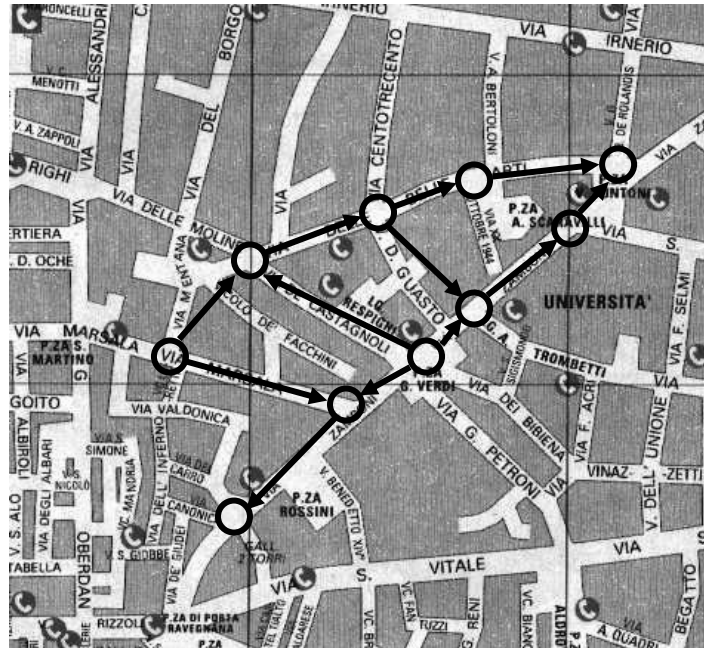
E = insieme di lati (o archi, A)



- Consente di modellare “naturalmente”
 - problemi di scelta di percorsi
 - problemi di connessione di punti mediante reti

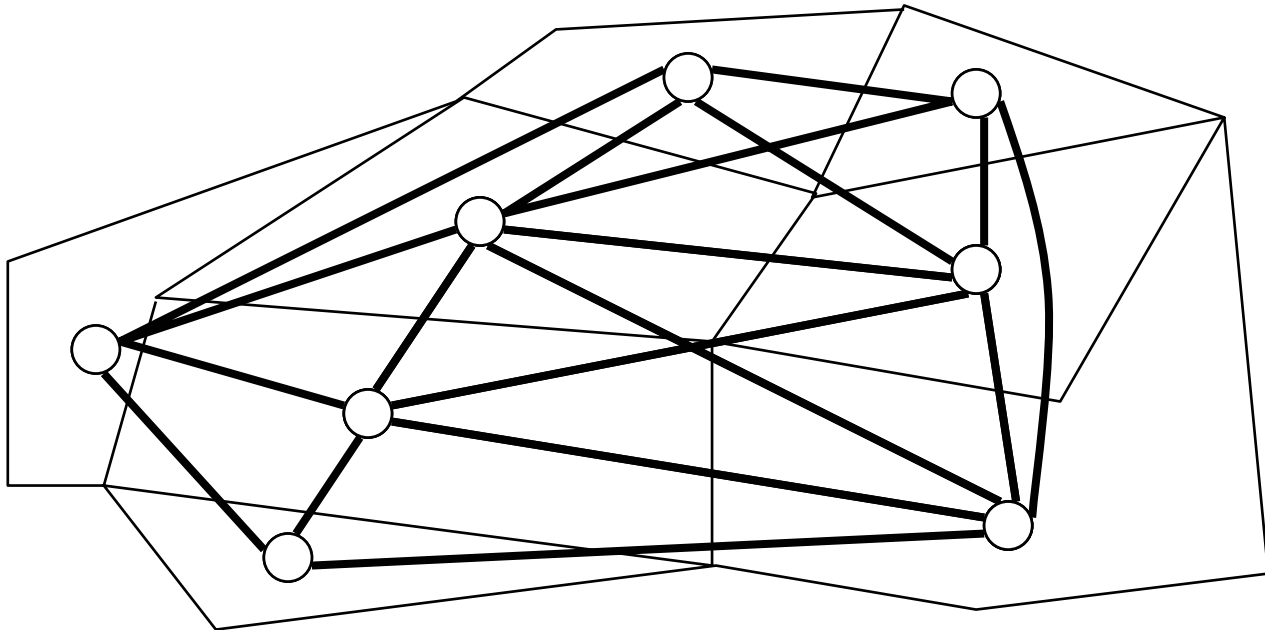
Esempio: rete stradale

- Nodi: incroci
- Archi: tratti di strada (orientati e non orientati)



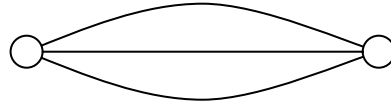
Esempio 3: Coloring

- Data una mappa, determinare il minimo numero di colori necessario affinché regioni adiacenti abbiano colori diversi

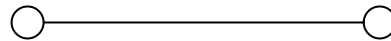


Grafi multipli e semplici

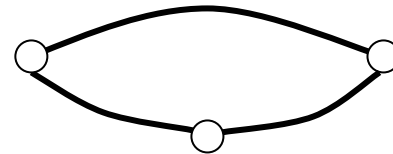
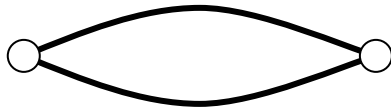
- Grafi multipli: più lati tra due vertici



semplici: solo un lato tra due vertici



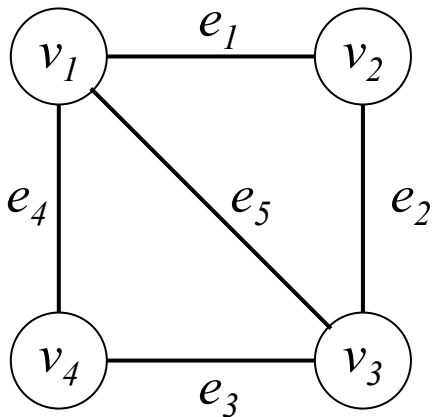
Considereremo solo grafi semplici



grafo semplice equiv. (1 lato ed 1 vertice in più)

Grafi non orientati (simmetrici)

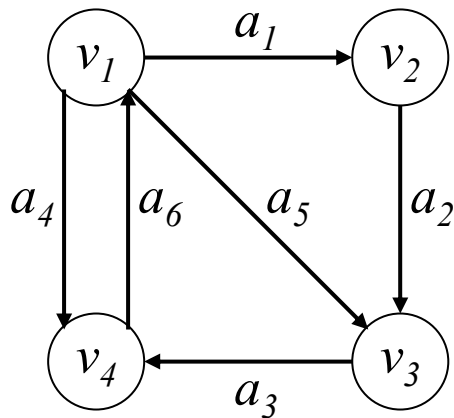
- $G = (V, E)$ $|V| = n$ $|E| = m$
- Vertici $V = \{v_1, \dots, v_n\}$
- Lati $E = \{e_1, \dots, e_m\}$
 $e_k = (v_i, v_j) \equiv (v_j, v_i)$ coppie di vertici



$$V = \{v_1, v_2, v_3, v_4\}$$
$$E = \{e_1, e_2, e_3, e_4, e_5\}$$
$$= \{(v_1, v_2), (v_1, v_3), \dots\}$$

Grafi orientati (asimmetrici)

- $G = (V, A) \quad |V| = n \quad |A| = m$
- Vertici $V = \{v_1, \dots, v_n\}$
- Archi $A = \{a_1, \dots, a_m\}$
 $a_k = (v_i, v_j) \neq (v_j, v_i)$ coppie ordinate
 v_i coda (tail), v_j testa (head)



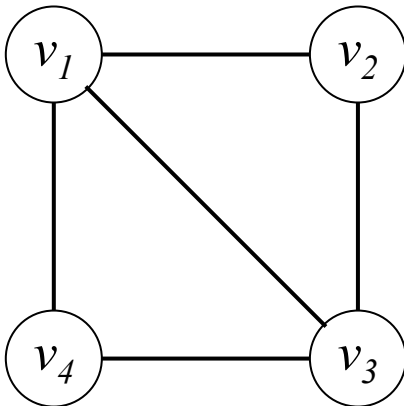
$$V = \{v_1, v_2, v_3, v_4\}$$

$$A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$$
$$= \{\dots (v_1, v_4), (v_4, v_1), \dots\}$$

Sommario

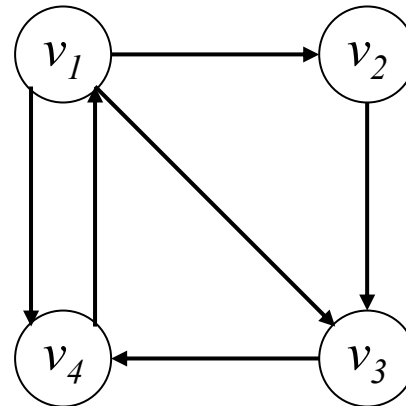
Grafi non orientati (simmetrici)

- $G = (V, E)$
- $V = \{v_1, \dots, v_n\}$
- $E = \{e_1, \dots, e_m\}$
 $e_k = (v_i, v_j) \equiv (v_j, v_i)$



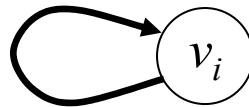
Grafi orientati (asimmetrici)

- $G = (V, A)$
- $V = \{v_1, \dots, v_n\}$
- $A = \{a_1, \dots, a_m\}$
 $a_k = (v_i, v_j) \neq (v_j, v_i)$

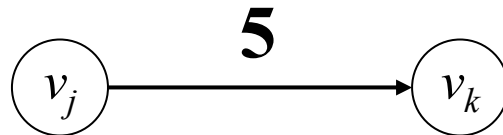


Terminologia

- Autoanelli: lati o archi (v_i, v_i)



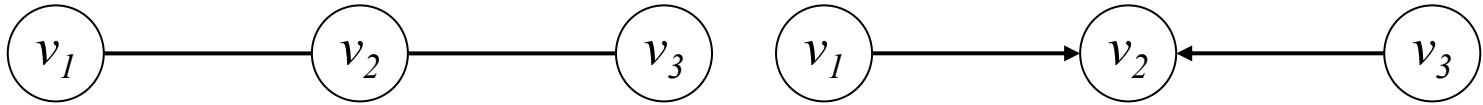
- Grafo pesato (orientato e non):
ad ogni lato (arco) è associato un peso (costo, ...)



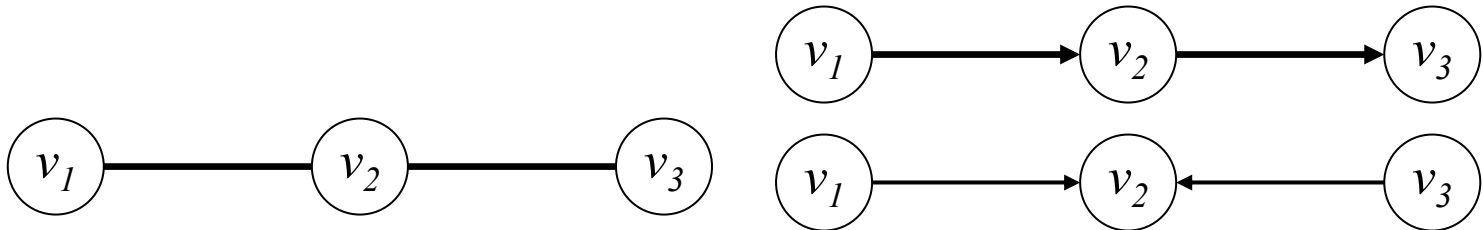
$$c(v_j, v_k) = c(e_i) = c(a_i) = c_i = c_{jk} = 5$$

Terminologia (2)

- v_i e v_j sono adiacenti se $(v_i, v_j) \in E$ ($\in A$)

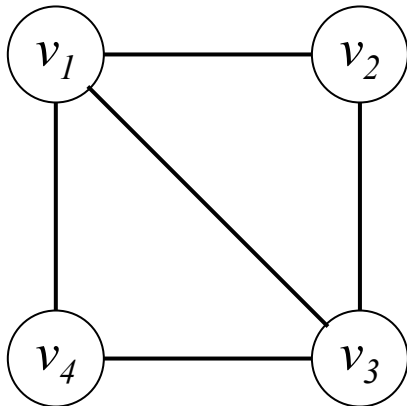


- due lati sono adiacenti (o consecutivi) se hanno un vertice comune
- due archi sono adiacenti (o consecutivi) se la testa del primo coincide con la coda del secondo



Grafi non orientati

- Star di v $\Gamma(v) = \{v_j : (v, v_j) \in E\}$
- Grado di v $d(v) = |\Gamma(v)|$



$$\Gamma(v_2) = \{v_1, v_3\}$$

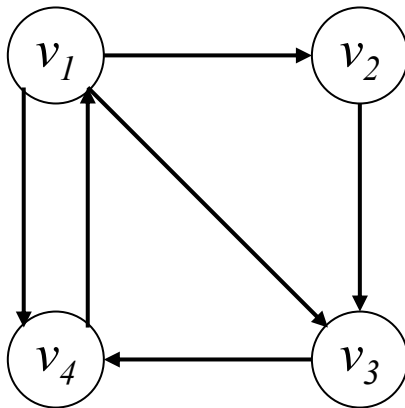
$$d(v_2) = 2$$

$$\Gamma(v_3) = \{v_1, v_2, v_4\}$$

$$d(v_3) = 3$$

Grafi orientati

- Forward star di v $\Gamma^+(v) = \{v_j : (v, v_j) \in A\}$
- Backward star di v $\Gamma^-(v) = \{v_j : (v_j, v) \in A\}$
- Semigrado esterno di v $d^+(v) = |\Gamma^+(v)|$
- Semigrado interno di v $d^-(v) = |\Gamma^-(v)|$



$$\Gamma^+(v_3) = \{v_4\}$$

$$\Gamma^-(v_3) = \{v_1, v_2\}$$

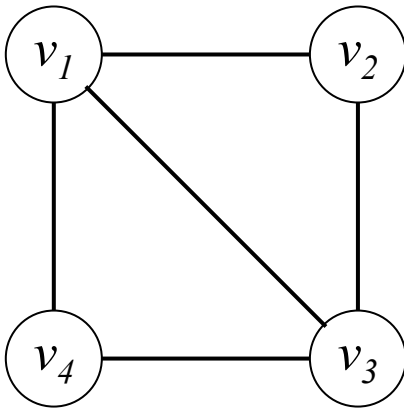
$$d^+(v_3) = 1$$

$$d^-(v_3) = 2$$

Sommario

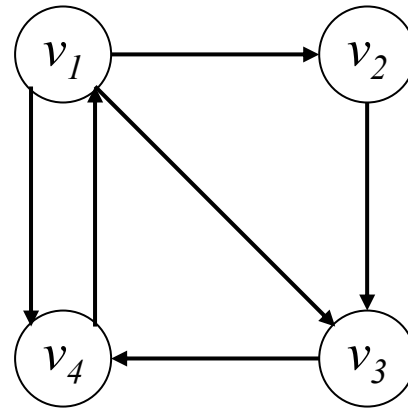
Grafi non orientati

- $\Gamma(v) = \{v_j : (v, v_j) \in E\}$
- $d(v) = |\Gamma(v)|$



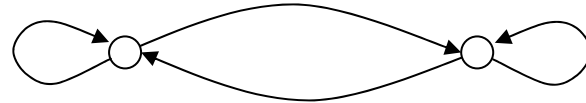
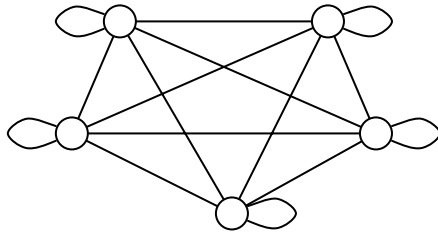
Grafi orientati

- $\Gamma^+(v) = \{v_j : (v, v_j) \in A\}$
- $\Gamma^-(v) = \{v_j : (v_j, v) \in A\}$
- $d^+(v) = |\Gamma^+(v)|$
- $d^-(v) = |\Gamma^-(v)|$

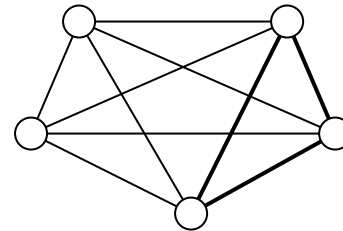
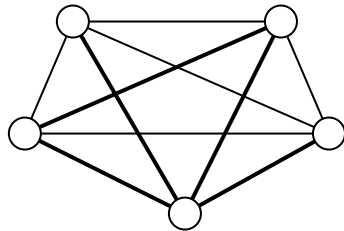


Terminologia (3)

- G è completo se $\forall v_i, v_j \in V, (v_i, v_j) \in E$ ($\in A$)
(con/senza autoanelli)



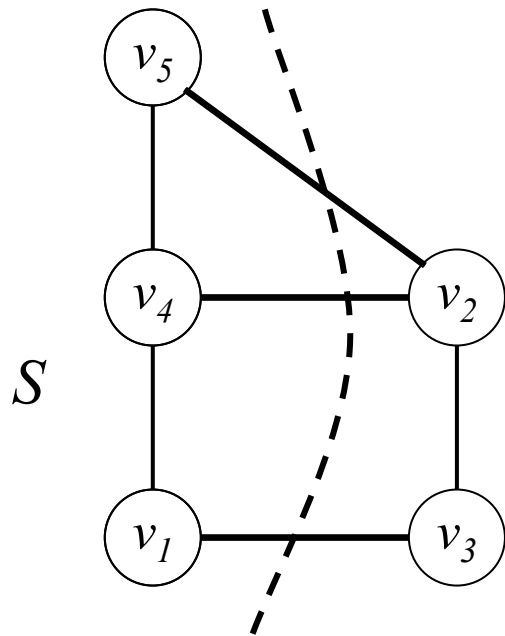
- Grafo parziale $G' = (V, E')$ con $E' \subseteq E$
- Sottografo $G' = (V', E')$ con $V' \subseteq V, E' \subseteq E$



Grafi non orientati

- Dato $S \subset V$, taglio (cut) associato ad S :

$$\delta(S) = \{(v_i, v_j) : |S \cap \{v_i, v_j\}| = 1\}$$



$$S = \{v_1, v_4, v_5\}$$

$$\delta(S) = \{(v_2, v_5), (v_2, v_4), (v_1, v_3)\}$$

Grafi orientati

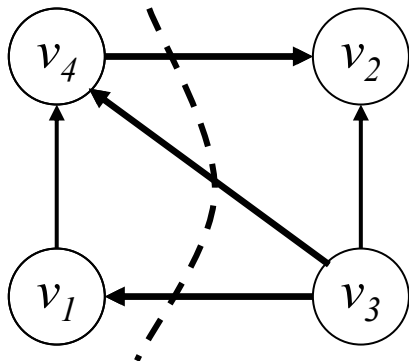
- Dato $S \subset V$, taglio orientato associato ad S

$$\delta^+(S) = \{(v_i, v_j) : v_i \in S, v_j \notin S\}$$

$$\delta^-(S) = \{(v_i, v_j) : v_i \notin S, v_j \in S\}$$

$$\delta^+(S) \equiv \delta^-(V \setminus S)$$

- Archi interni $\sigma(S) = \{(v_i, v_j) : v_i \in S, v_j \in S\}$



$$S = \{v_1, v_4\} \quad \sigma(S) = \{(v_1, v_4)\}$$

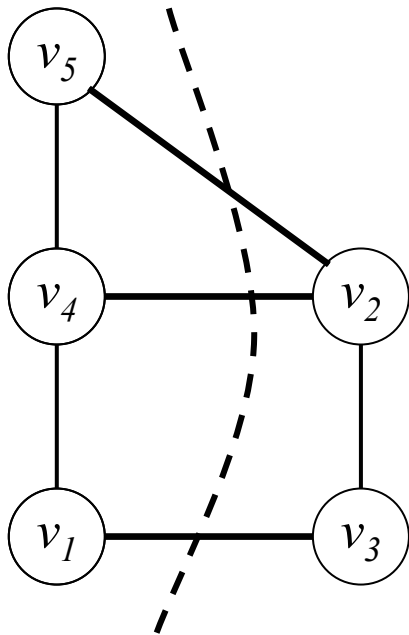
$$\delta^+(S) = \{(v_4, v_2)\}$$

$$\delta^-(S) = \{(v_3, v_4), (v_3, v_1)\}$$

Sommario

Grafi non orientati

$$\delta(S) = \{(v_i, v_j) : |S \cap \{v_i, v_j\}| = 1\}$$



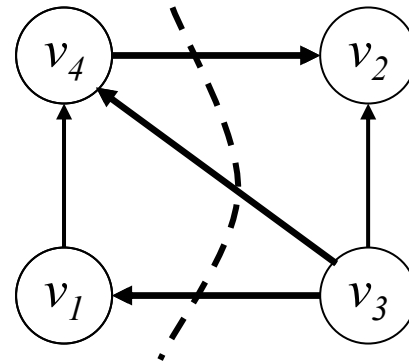
Grafi orientati

$$\delta^+(S) = \{(v_i, v_j) : v_i \in S, v_j \notin S\}$$

$$\delta^-(S) = \{(v_i, v_j) : v_i \notin S, v_j \in S\}$$

$$\delta^+(S) \equiv \delta^-(V \setminus S)$$

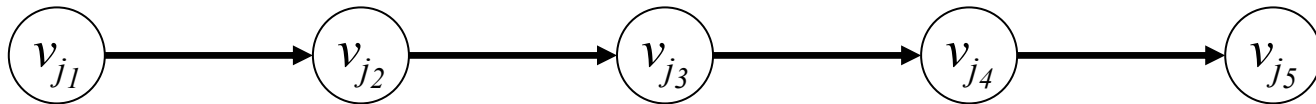
$$\sigma(S) = \{(v_i, v_j) : v_i \in S, v_j \in S\}$$



Terminologia (4)

- Cammino (path):

sequenza di lati (archi) consecutivi

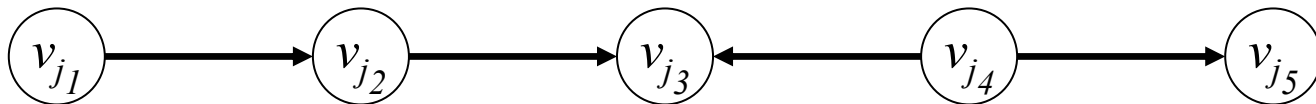


$$\{(v_{j_1}, v_{j_2}), (v_{j_2}, v_{j_3}), (v_{j_3}, v_{j_4}), (v_{j_k}, v_{j_{k+1}})\}$$

- Ciclo o circuito (cycle): cammino con $v_{j_{k+1}} = v_{j_1}$
 - semplice: senza ripetizione di vertici
 - elementare: senza ripetizione di archisemplice \Rightarrow elementare

Terminologia (5)

- Grafo aciclico: non contiene cicli
- In un grafo orientato una catena è una sequenza di archi con in comune la testa e la coda (non necessariamente consecutivi)



- Grafo connesso: $\forall v_i, v_j \in V \exists$ cammino da v_i a v_j

G non orientato:

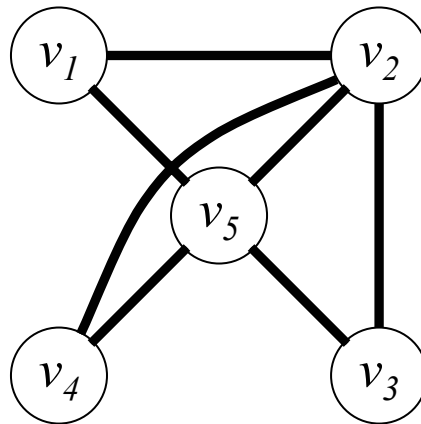
G non connesso \Leftrightarrow costituito da *componenti connesse* separate

Cammini euleriani

- Un cammino o ciclo elementare è euleriano se visita tutti i lati

Teorema di Eulero:

G ammette un ciclo euleriano $\Leftrightarrow d(v)$ pari $\forall v \in V$

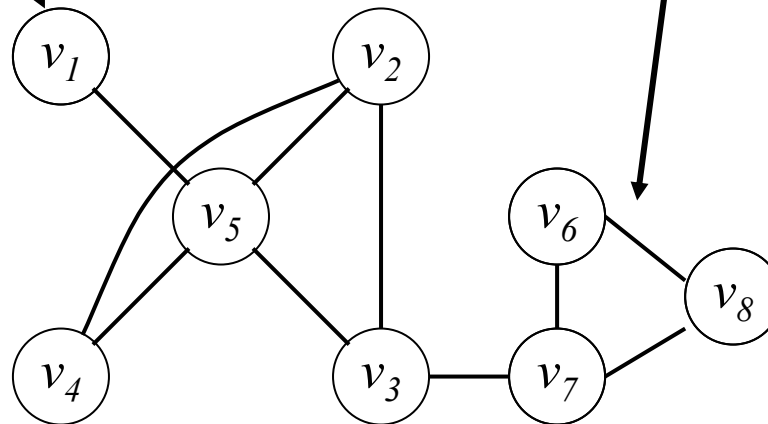


Cammini hamiltoniani

- Un cammino o ciclo semplice è hamiltoniano se visita tutti i vertici del grafo

Condizioni necessarie per l'esistenza di un c. hamiltoniano

G connesso con $d(v) \geq 2 \quad \forall v \in V$ e t.c. $|\delta(S)| \geq 2 \quad \forall S \subset V$



Memorizzazione di grafi

Grafi densi ($m \approx n^2$)

Non pesati

matrice di adiacenza

$[a_{ij}] (n \times n)$

$$a_{ij} = \begin{cases} 1 & \text{se } (v_i, v_j) \in A (\in E) \\ 0 & \text{altrimenti} \end{cases}$$

Pesati

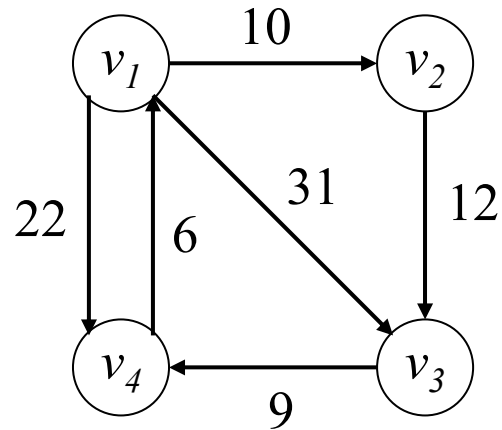
matrice dei pesi

$[c_{ij}] (n \times n)$

$$c_{ij} = \begin{cases} c(v_i, v_j) & \text{se } (v_i, v_j) \in A (\in E) \\ \infty & \text{altrimenti} \end{cases}$$

$[a_{ij}]$, $[c_{ij}]$ simmetriche per grafi non orientati

Esempio



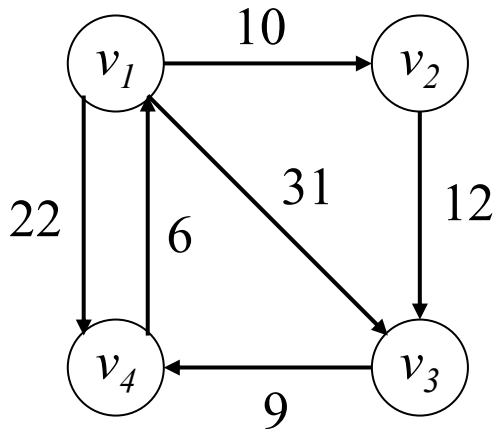
$$a_{ij} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$c_{ij} = \begin{bmatrix} \infty & 10 & 31 & 22 \\ \infty & \infty & 12 & \infty \\ \infty & \infty & \infty & 9 \\ 6 & \infty & \infty & \infty \end{bmatrix}$$

Memorizzazione di grafi

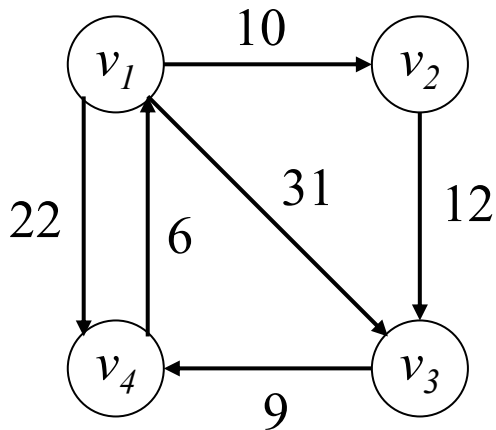
Grafi sparsi ($m \ll n^2$)

- Meglio memorizzare solo gli archi esistenti



$$f' = \begin{array}{|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 2 & 3 & 4 \\ \hline \end{array}$$
$$u' = \begin{array}{|c|c|c|c|c|c|} \hline 2 & 3 & 4 & 3 & 4 & 1 \\ \hline \end{array}$$
$$c' = \begin{array}{|c|c|c|c|c|c|} \hline 10 & 31 & 22 & 12 & 9 & 6 \\ \hline \end{array}$$

Memorizzazione di grafi sparsi



$$c' = \begin{array}{|c|c|c|c|c|c|} \hline 10 & 31 & 22 & 12 & 9 & 6 \\ \hline \end{array}$$

$$u' = \begin{array}{|c|c|c|c|c|c|} \hline 2 & 3 & 4 & 3 & 4 & 1 \\ \hline \end{array}$$

$$p' = \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 5 & 6 & 7 \\ \hline \end{array}$$

$$f' = \begin{array}{|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 2 & 3 & 4 \\ \hline \end{array}$$

Memorizzazione di grafi sparsi

Non pesati

forward star

- vettore $p(n+1)$ di puntatori:

$$p_1=1, p_{n+1}=m+1$$

- vettore $u(m)$:

$(u_{p_i}, \dots, u_{p_{(i+1)}-1})$ indici vertici

$v : \exists$ l'arco (v_i, v)

Pesati

forward star + vettore $c(m)$

$c_k =$ peso dell'arco
individuato da u_k

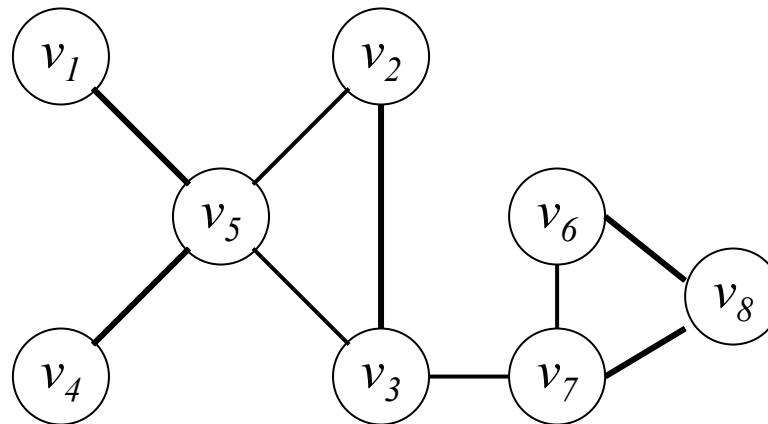
Foreste ed alberi

Dato un grafo non orientato $G=(V,E)$

- Albero (tree):

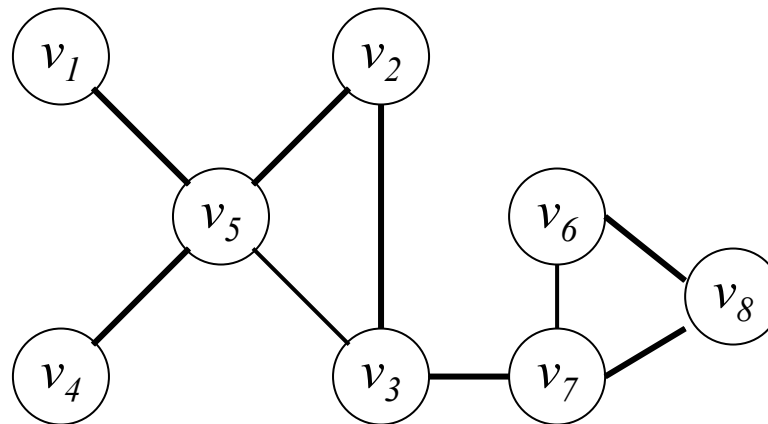
sottografo connesso aciclico $G'=(V',E')$

$\Leftrightarrow \forall v_i, v_j \in V'$ in G' grafo parziale aciclico $G=(V,E)$ ed un solo cammino



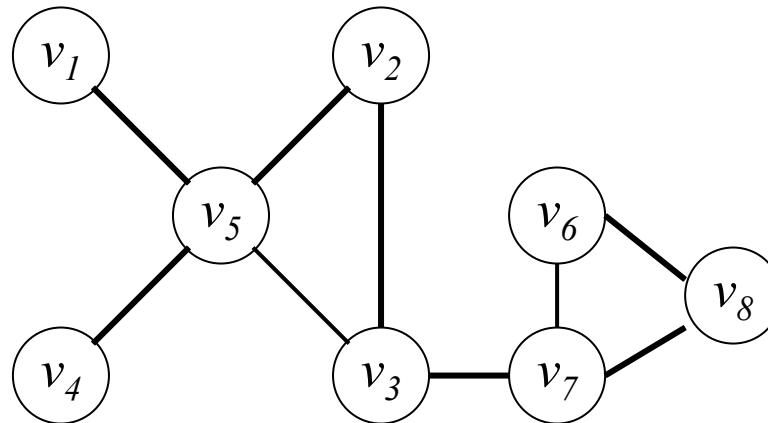
Foreste ed alberi

- Foresta massimale:
ogni arco in $E \setminus E'$ chiude un ciclo con E'
- Albero ricoprente (Spanning Tree, ST):
foresta massimale connessa



Foreste ed alberi

- Albero ricoprente (Spanning Tree, ST):
 - foresta massimale connessa
 - Grafo parziale connesso aciclico $G'=(V,E')$
 - $\forall v_i, v_j \in V \exists$ in G' uno ed un solo cammino
 - è connesso e contiene $|V|-1 = n-1$ lati



Problema del più corto ST

Shortest ST, SST

- Dato $G=(V,E)$ pesato e connesso, trovare lo ST $G'=(V,E')$ tale che il costo $\sum_{e \in E'} c(e)$ sia minimo

Applicazioni:

- combinare i terminali di un circuito elettrico con minima lunghezza di filo (per ridurre effetti parassiti)
- collegare città mediante condutture a costo minimo senza punti di giunzione esterni
- sottoproblema di altri problemi più complessi

Modello di PLI di SST

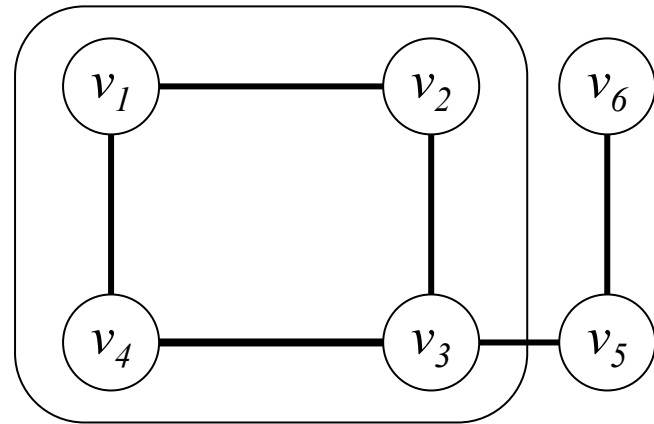
$$c_j = \text{peso del lato } e_j \quad x_j = \begin{cases} 1 & \text{se } e_j \text{ è nell'albero} \\ 0 & \text{altrimenti} \end{cases}$$

$$\min \sum_{j=1,m} c_j x_j$$

$$\sum_{j=1,m} x_j = n - 1$$

$$x_j \in \{0,1\}$$

$$\sum_{j \in \sigma(S)} x_j \leq |S| - 1 \quad \forall S \subseteq V, S \neq \emptyset$$



Subtour elimination constraints: $O(2^n)$

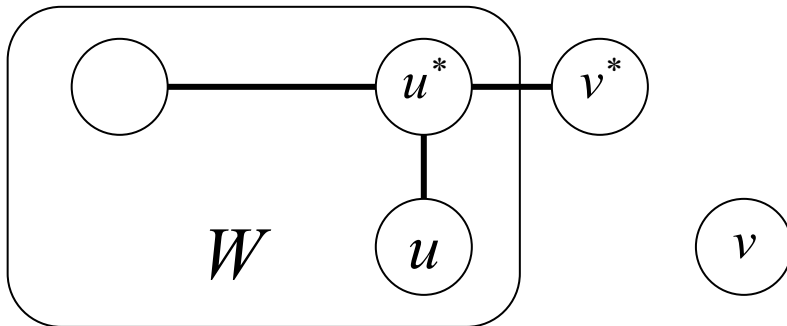
Teorema di Prim (1957)

Dati $G = (V, E)$

ed un albero parziale (W, E') , $W \subset V$, $E' \subset E$,

sia (u^*, v^*) il più corto (u, v) : $u \in W$, $v \in V \setminus W$.

Fra tutti gli ST di G che contengono E' ne esiste uno ottimo che contiene anche (u^*, v^*) .



Se (W, E') è ottimo,
 $(W \cup \{v^*\}, E' \cup \{(u^*, v^*)\})$ è ottimo

Dimostrazione Teorema di Prim

Sia ST^* il più corto ST contenente E' ,

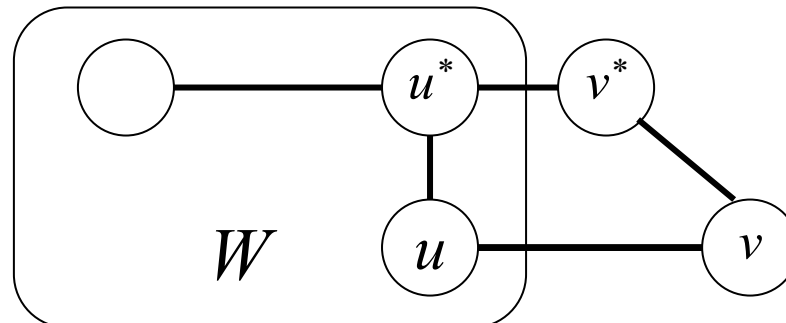
si supponga, per assurdo, che non contenga (u^*, v^*) .

In ST^* deve esistere un cammino tra u^* e v^* ,

che conterrà un lato (u, v) , con $u \in W$, $v \in V \setminus W$.

Rimuovendo $(u, v) \Rightarrow$ due alberi parziali,

aggiungendo $(u^*, v^*) \Rightarrow ST$ più corto di ST^*



Algoritmo di Prim (I versione)

- parti con uno ST ottimo costituito dal vertice v_1

$$W := \{v_1\}; E' := \emptyset;$$

- finchè non si ha uno ST completo ($|W| = n$):

- determina (u^*, v^*) , lato a costo minimo che collega un vertice dello ST con un vertice non ancora raggiunto

- aggiungi il lato (u^*, v^*) allo ST

$$W := W \cup \{v^*\}; E' := E' \cup \{(u^*, v^*)\}$$

Algoritmo di Prim (I versione)

begin

$W := \{v_1\}; E' := \emptyset;$

while $|W| < n$ **do**

begin

determina (u^*, v^*) ;

$W := W \cup \{v^*\};$

$E' := E' \cup \{(u^*, v^*)\};$

end

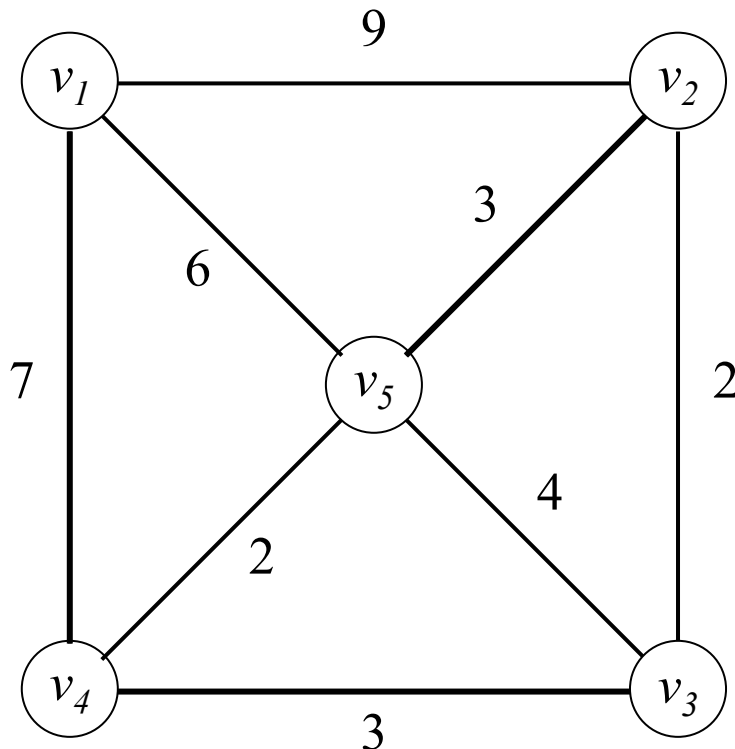
end.



- $n-1$ iterazioni
- \forall iterazione numero operazioni $\propto a |E|$
- tempo $O(n^3)$

Algoritmo di Prim (I versione)

$$(u^*, v^*) = \min \{(u, v) : u \in W, v \in V \setminus W\}.$$



- $W = \{v_1, v_2\}$
- $v^* = v_3$

Basta verificare se v_3
è più vicino a v_4 o v_5

Algoritmo di Prim (II versione)

- $b(v)$ vertice di W più vicino a $v \in V \setminus W$

Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

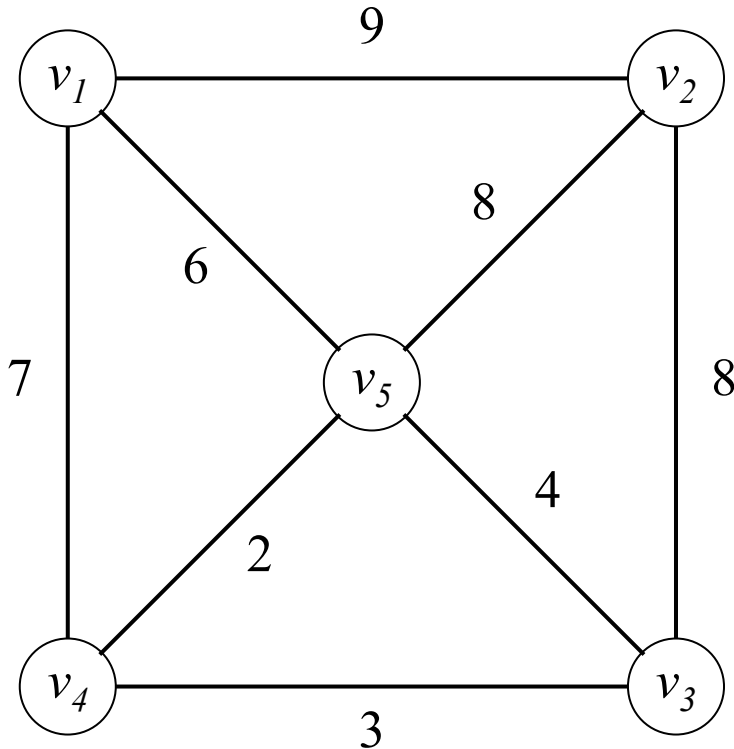
end

end.



- $n-1$ iterazioni
- \forall iterazione numero operaz. $\propto a |V \setminus W|$
- tempo $O(n^2)$

Esempio



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

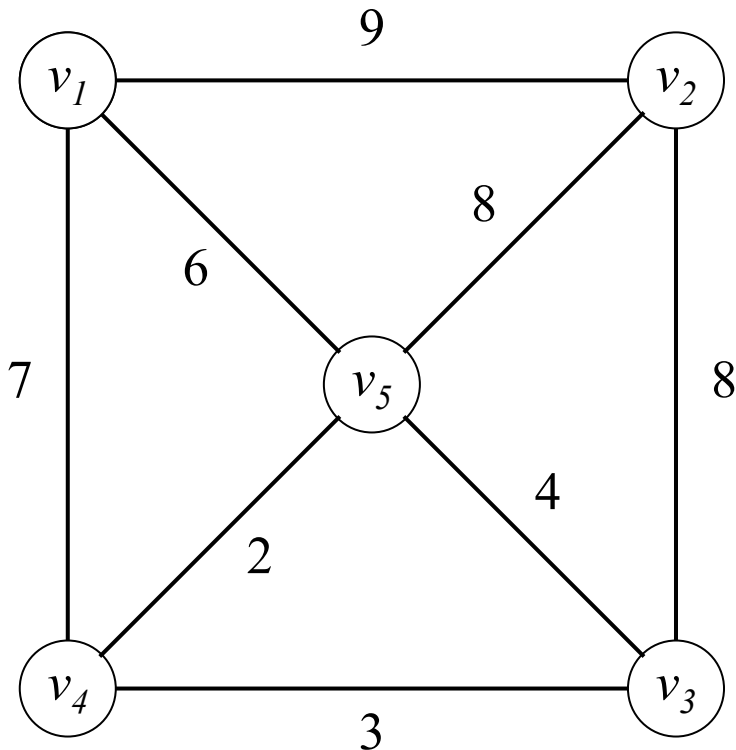
for each $v \in V \setminus W$ **do**

if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

Inizializzazione insiemi



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

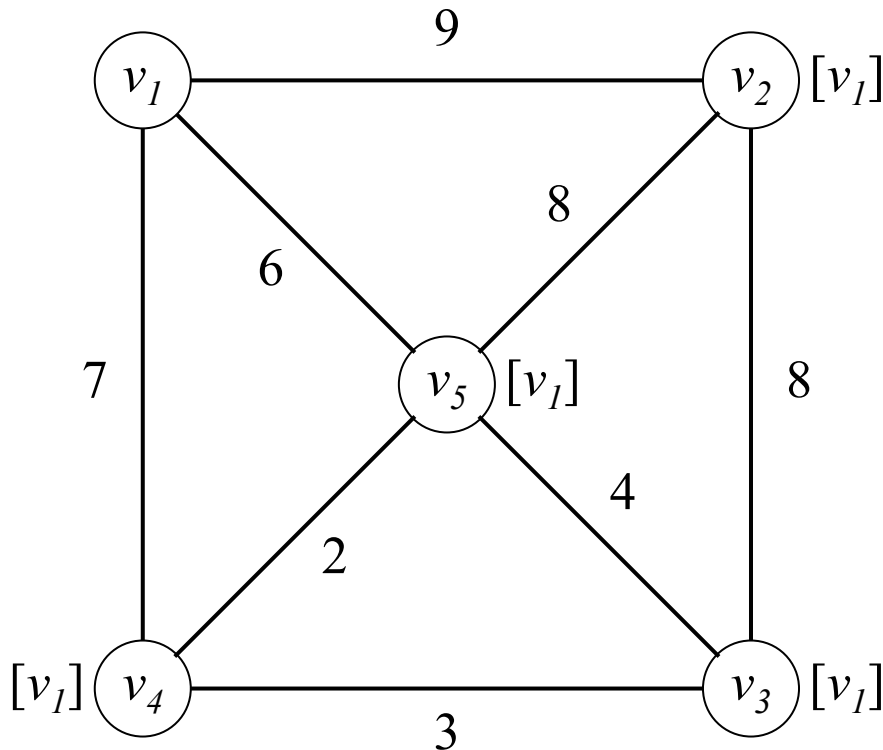
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1\}$
- $E' = \emptyset$

Inizializzazione etichette



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

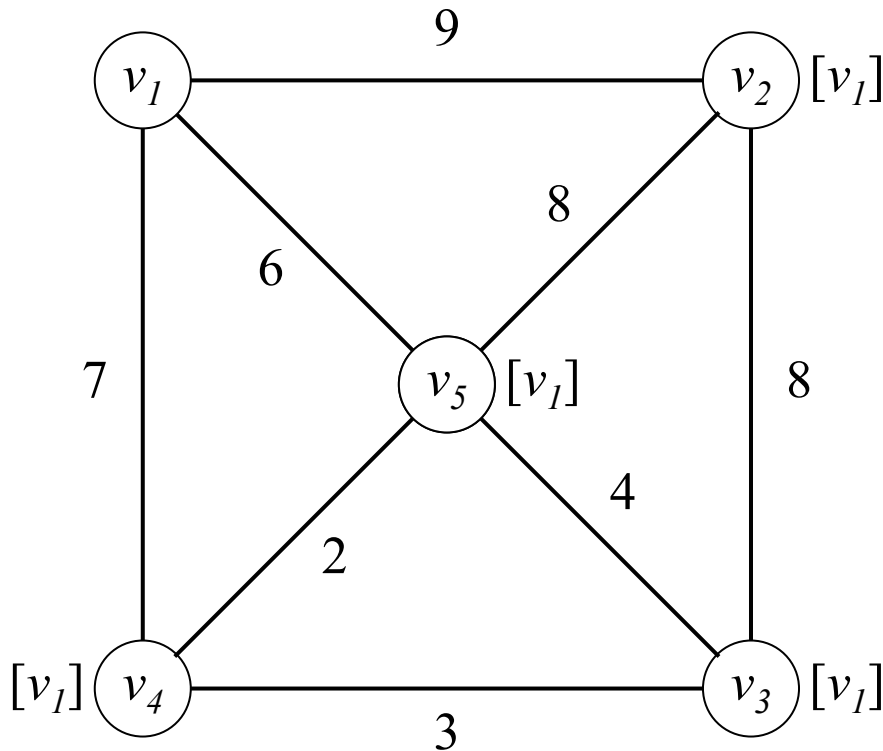
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1\}$
- $E' = \emptyset$

Selezione del vertice (1)



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

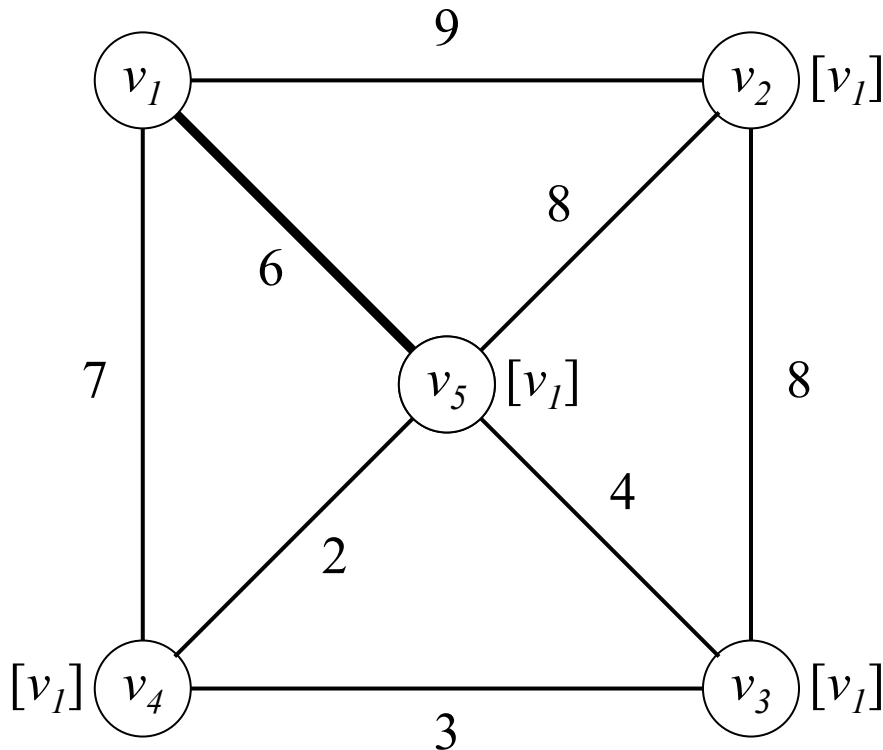
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1\}$
- $E' = \emptyset$

Aggiornamento insiemi (1)



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

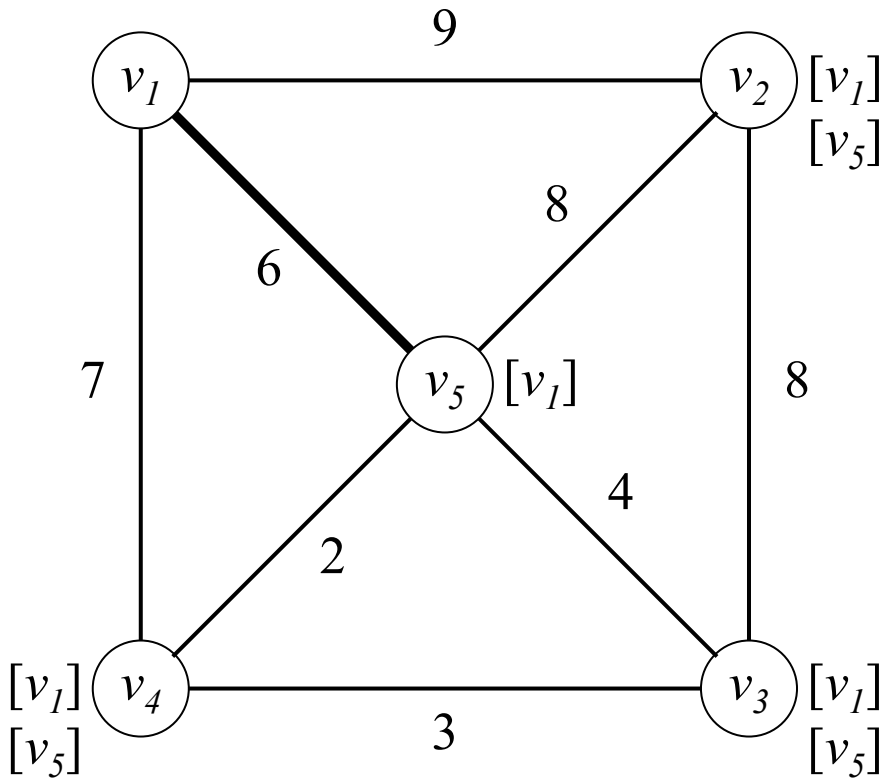
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1, v_5\}$
- $E' = \{(v_5, v_1)\}$

Aggiornamento etichette (1)



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

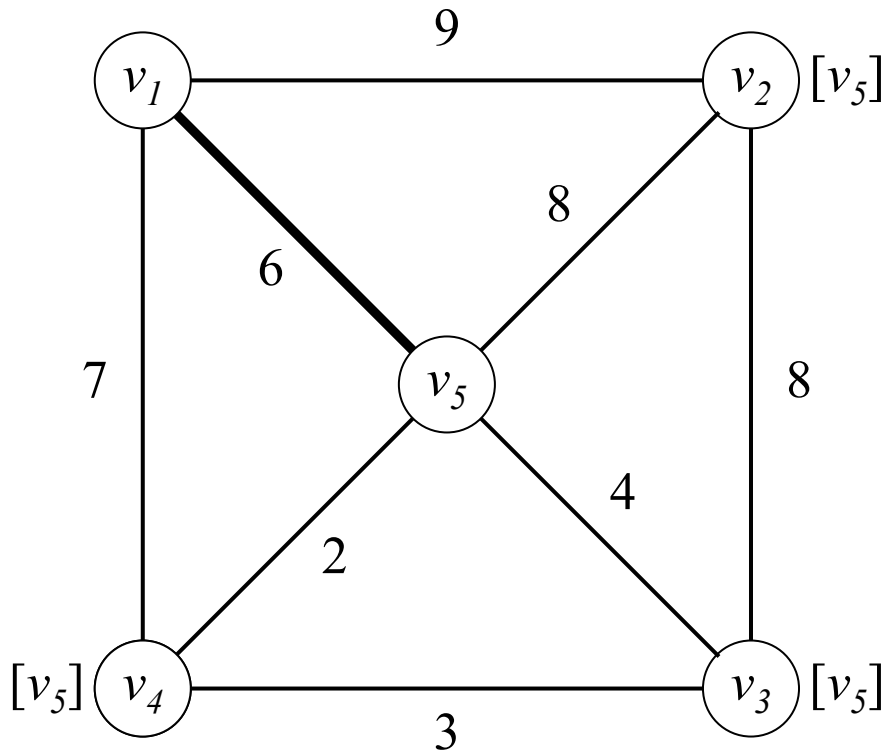
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1, v_5\}$
- $E' = \{(v_5, v_1)\}$

Selezione del vertice (2)



Procedure SST_Prim

begin

$W := \{v_l\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_l\}$ **do** $b(v) := v_l;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

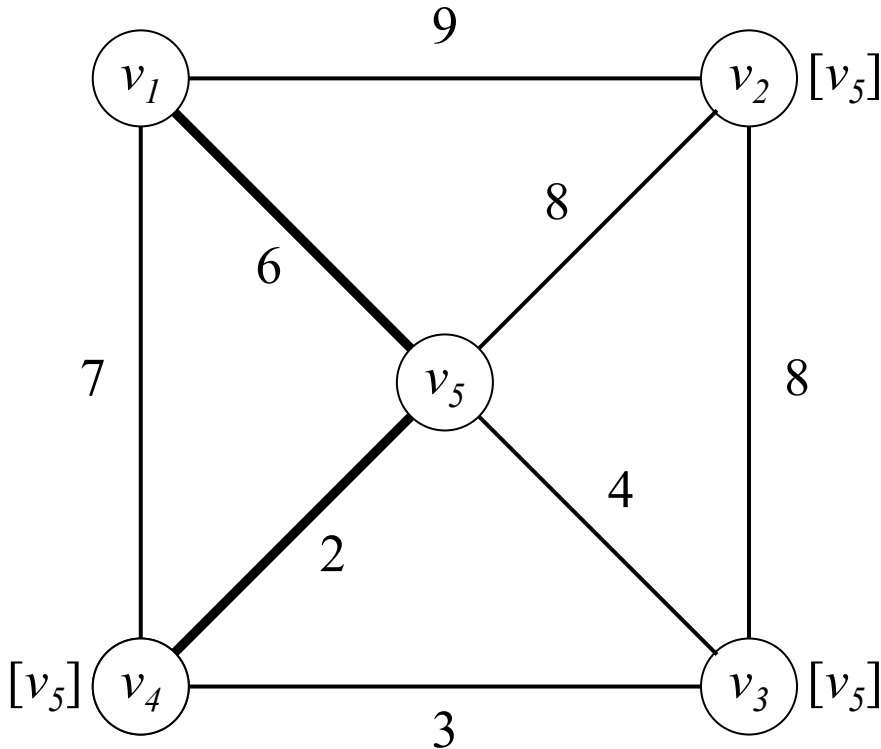
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_l, v_5\}$
- $E' = \{(v_5, v_l)\}$

Aggiornamento insiemi (2)



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

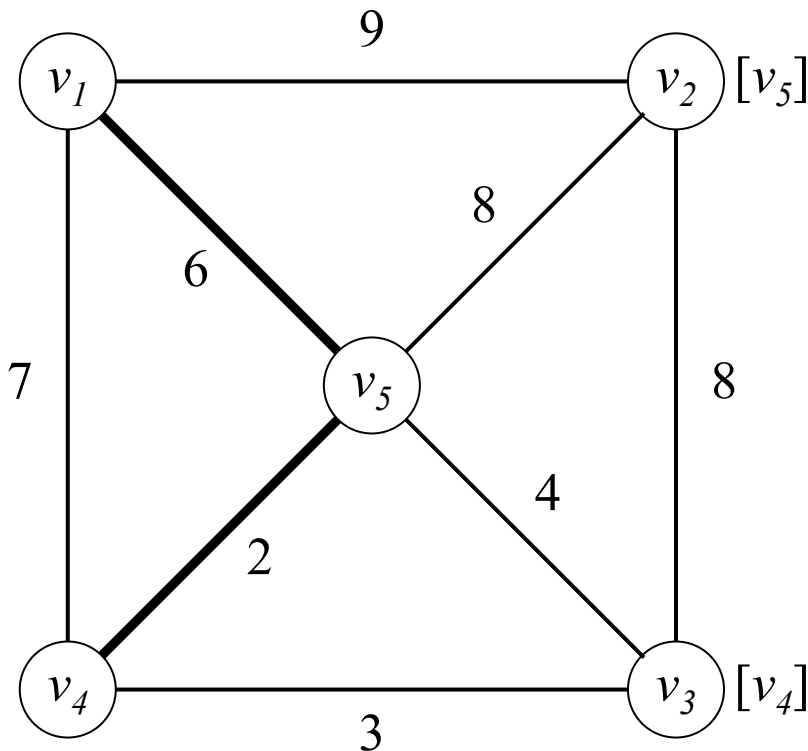
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1, v_5, v_4\}$
- $E' = \{(v_5, v_1), (v_4, v_5)\}$

Aggiornamento etichette (2)



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

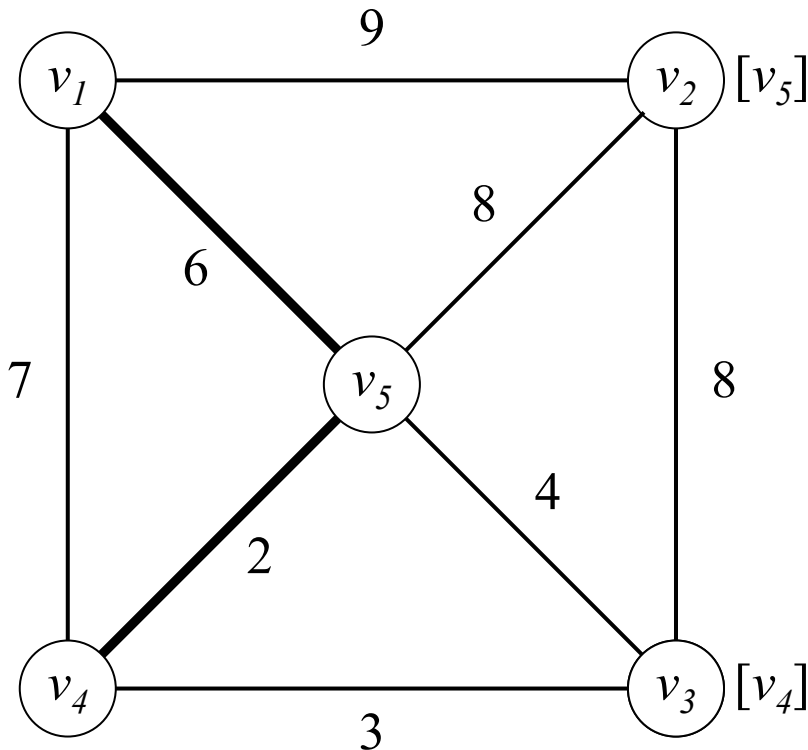
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1, v_5, v_4\}$
- $E' = \{(v_5, v_1), (v_4, v_5)\}$

Selezione del vertice (3)



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

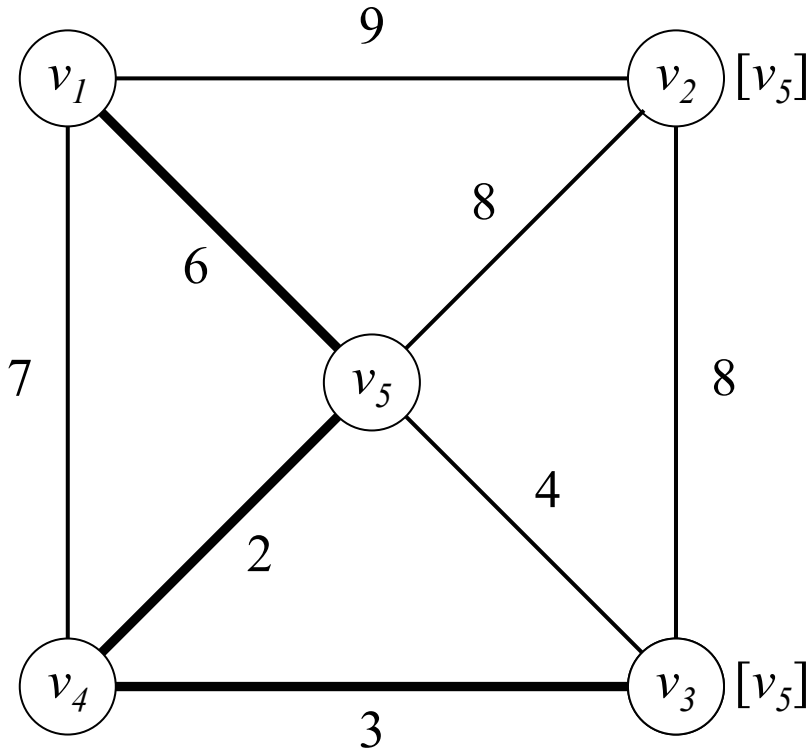
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1, v_5, v_4\}$
- $E' = \{(v_5, v_1), (v_4, v_5)\}$

Aggiornamento insiemi (3)



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

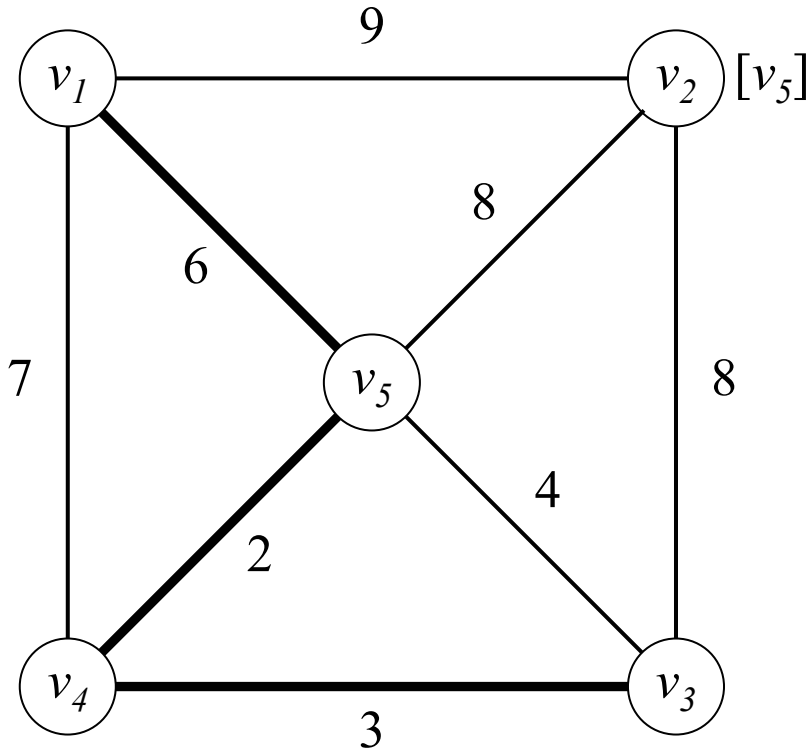
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1, v_5, v_4, v_3\}$
- $E' = \{(v_5, v_1), (v_4, v_5), (v_3, v_4)\}$

Aggiornamento etichette (3)



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

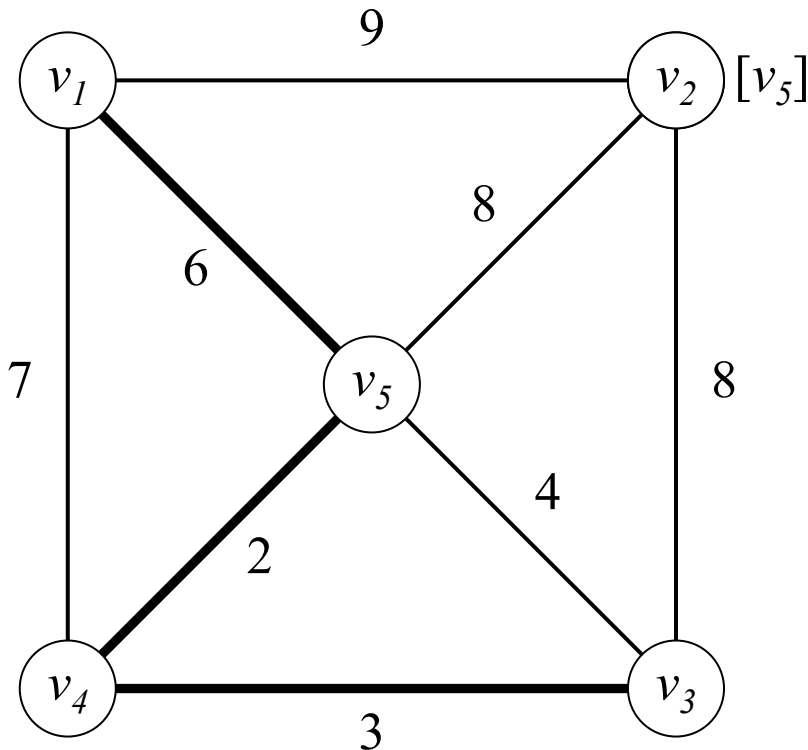
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1, v_5, v_4, v_3\}$
- $E' = \{(v_5, v_1), (v_4, v_5), (v_3, v_4)\}$

Selezione del vertice (4)



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

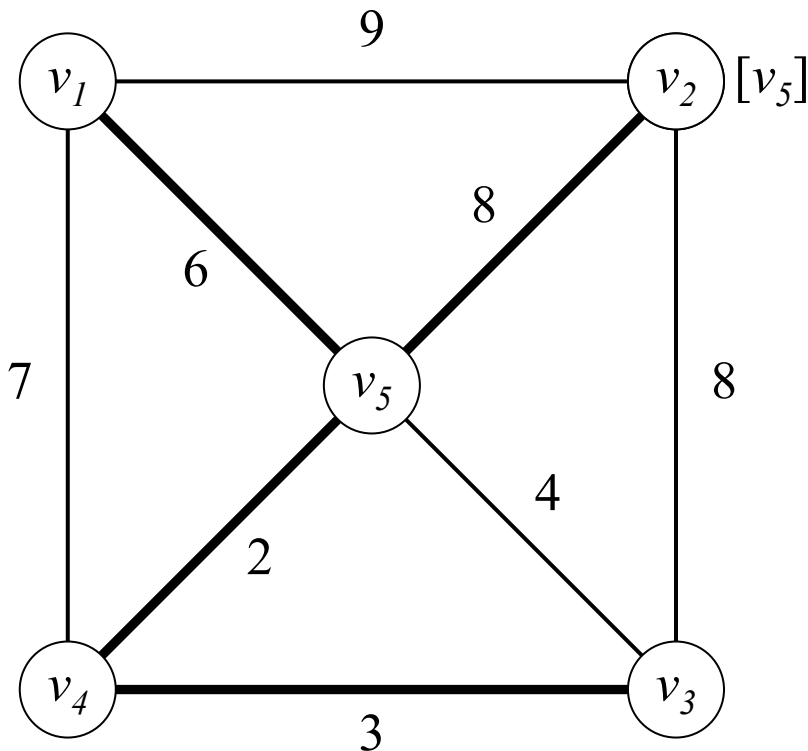
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1, v_5, v_4, v_3\}$
- $E' = \{(v_5, v_1), (v_4, v_5), (v_3, v_4)\}$

Aggiornamento insiemi (4)



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

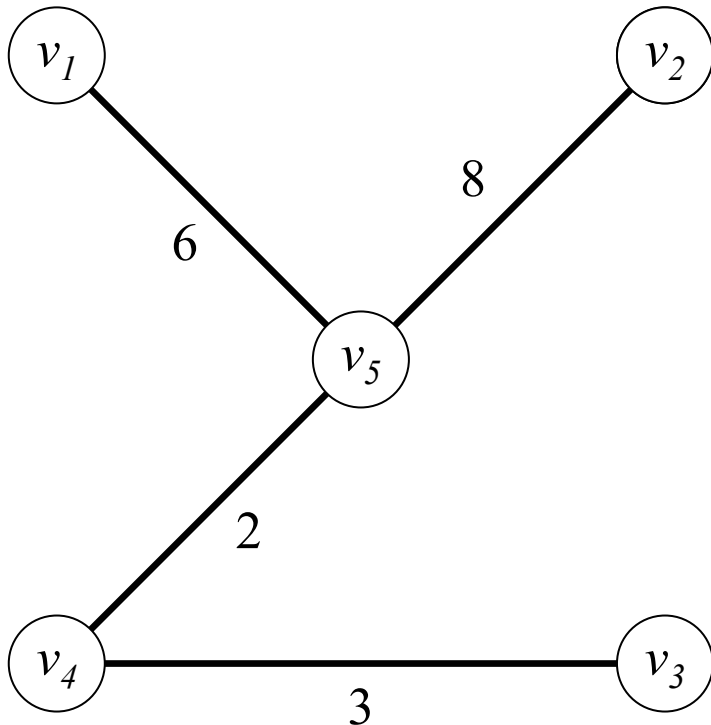
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

- $W = \{v_1, v_5, v_4, v_3, v_2\}$
- $E' = \{(v_5, v_1), (v_4, v_5), (v_3, v_4), (v_2, v_5)\}$

Soluzione



Procedure SST_Prim

begin

$W := \{v_1\}; E' := \emptyset;$

comment $b(v) = \text{vertrice} \in W: c(v, b(v)) = \min_{r \in W} \{c(v, r)\};$

for each $v \in V \setminus \{v_1\}$ **do** $b(v) := v_1;$

while $|W| < n$ **do begin**

$v^* \in V \setminus W: c(v^*, b(v^*)) = \min_{v \in V \setminus W} \{c(v, b(v))\};$

$W := W \cup \{v^*\}; E' := E' \cup \{(v^*, b(v^*))\};$

for each $v \in V \setminus W$ **do**

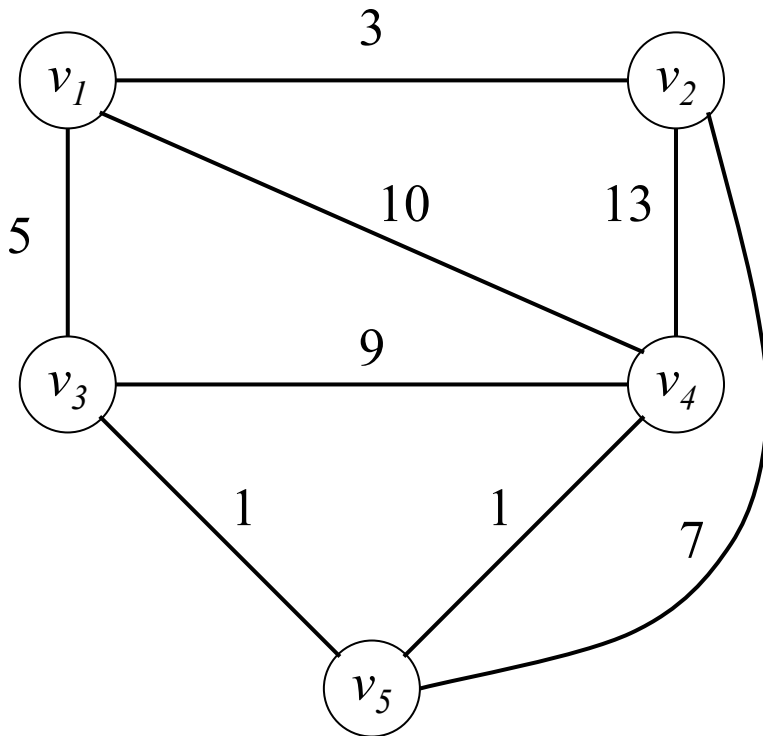
if $c(v, v^*) < c(v, b(v))$ **then** $b(v) := v^*;$

end

end.

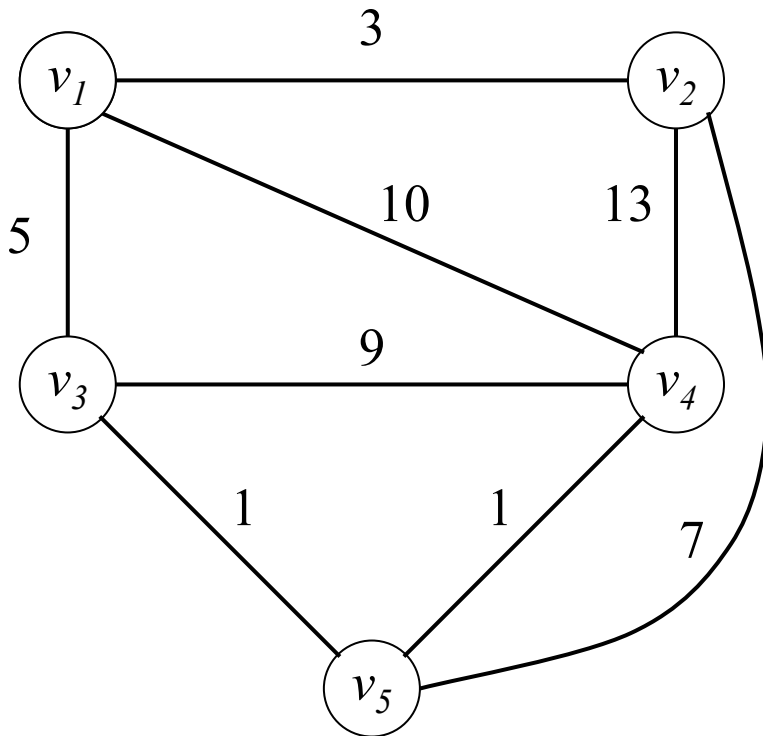
- $W = \{v_1, v_5, v_4, v_3, v_2\}$
- $E' = \{(v_5, v_1), (v_4, v_5), (v_3, v_4), (v_2, v_5)\}$
- **costo: 19**

Esempio



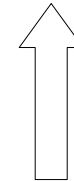
	v_1	v_2	v_3	v_4	v_5
$b(v)$
$c(v, b(v))$

Iterazione 1



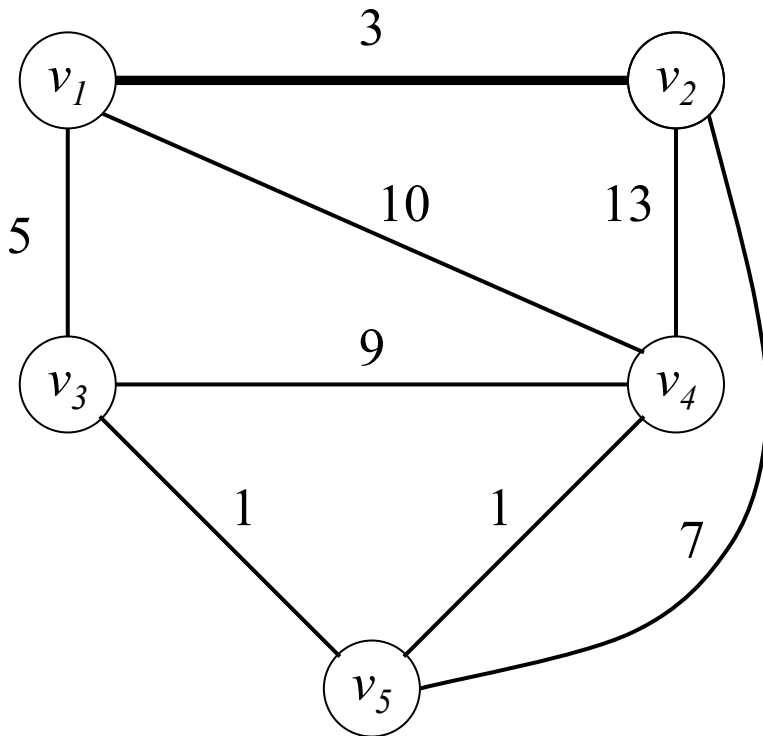
$$W = \{ 1 \}$$

	v_1	v_2	v_3	v_4	v_5
$b(v)$.	1	1	1	1
$c(v, b(v))$.	3	5	10	∞



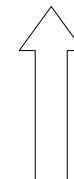
Iterazione 2

$$W = \{ 1, 2 \}$$



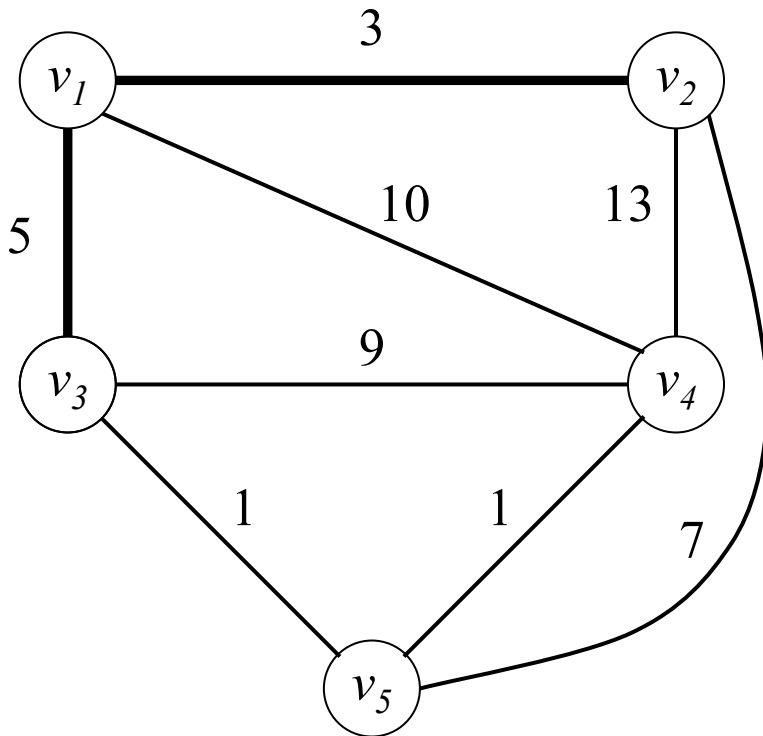
	v_1	v_2	v_3	v_4	v_5
$b(v)$.	1	1	1	1
$c(v, b(v))$.	3	5	10	∞

	v_1	v_2	v_3	v_4	v_5
$b(v)$		1	1	1	2
$c(v, b(v))$		3	5	10	7



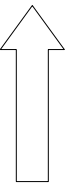
Iterazione 3

$$W = \{ 1, 2, 3 \}$$



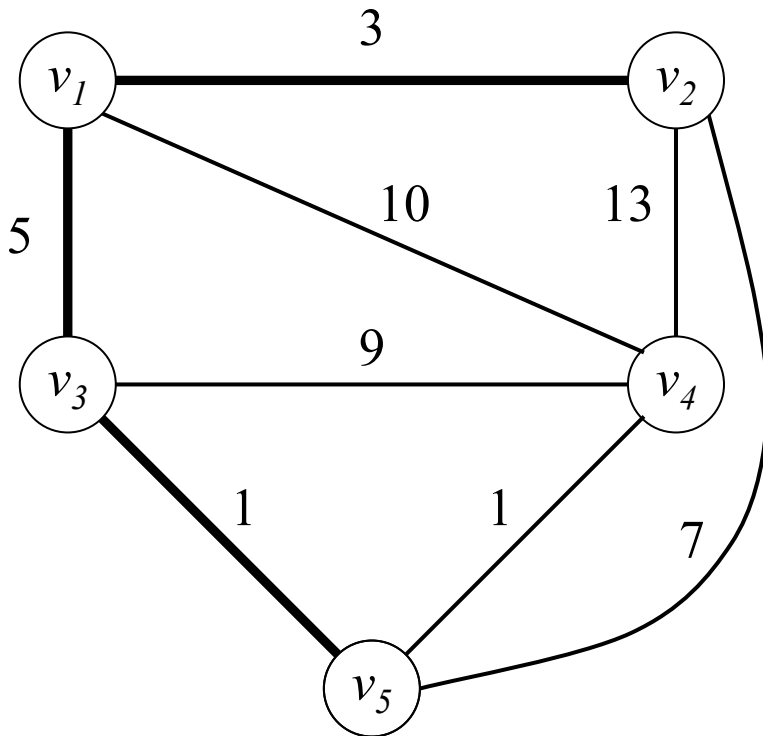
	v_1	v_2	v_3	v_4	v_5
$b(v)$		1	1	1	2
$c(v, b(v))$		3	5	10	7

	v_1	v_2	v_3	v_4	v_5
$b(v)$		1	1	3	3
$c(v, b(v))$		3	5	9	1



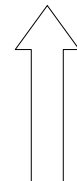
Iterazione 4

$$W = \{ 1, 2, 3, 5 \}$$

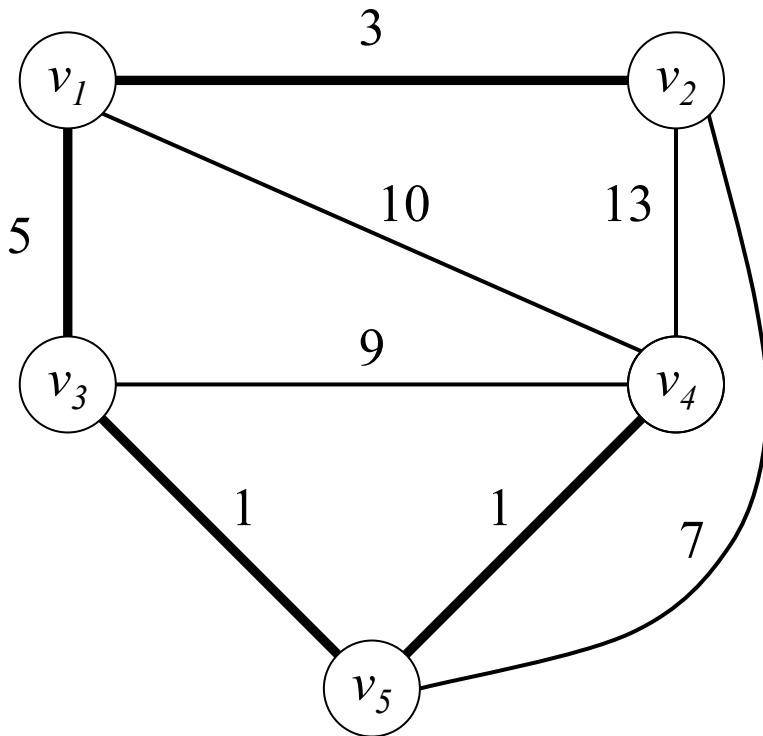


	v_1	v_2	v_3	v_4	v_5
$b(v)$		1	1	3	3
$c(v, b(v))$		3	5	9	1

	v_1	v_2	v_3	v_4	v_5
$b(v)$		1	1	5	3
$c(v, b(v))$		3	5	1	1



Iterazione 5



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

	v_1	v_2	v_3	v_4	v_5
$b(v)$		1	1	5	3
$c(v, b(v))$		3	5	1	1

$$\text{Costo} = 10$$

Algoritmo di Kruskal (1956)

- ordina E per costi non decrescenti;
- inizializza una soluzione vuota ($E' := \emptyset$);
- finchè non si ha uno ST completo ($|E'| = n - 1$):
 - considera il prossimo lato (e_j) nell'ordine
 - se e_j assieme agli archi di E' non chiude un circuito allora inseriscilo in soluzione
($E' := E' \cup \{ e_j \}$)
 - altrimenti scartalo

Algoritmo di Kruskal (1956)

Algoritmo di tipo greedy (le scelte sono basate su un criterio locale e non sono riconsiderate successivamente)

begin

$E' := \emptyset;$

ordina E per costi non decrescenti;

repeat

individua il lato (e_j) di costo minimo; $E := E \setminus \{ e_j \};$

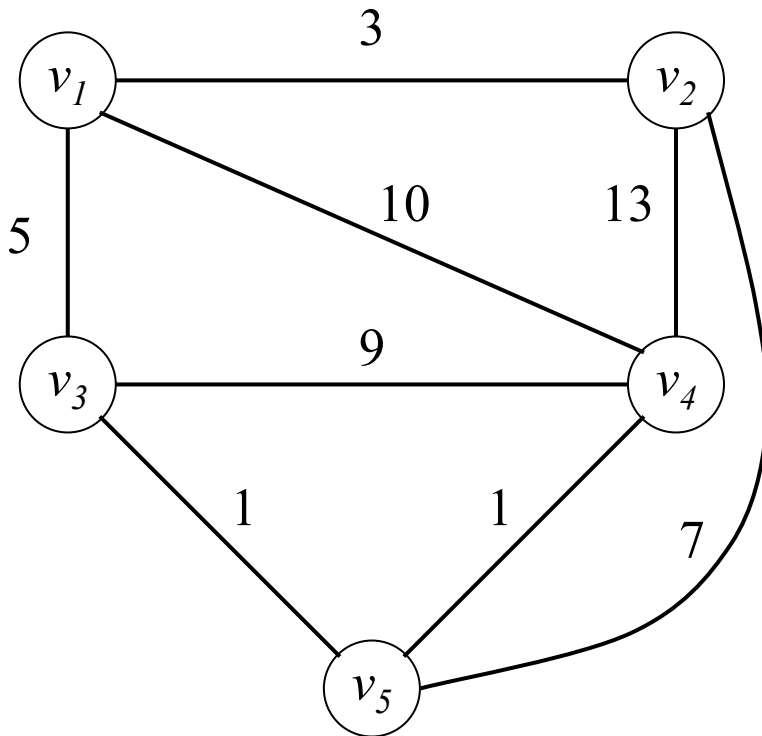
if $E' \cup \{ e_j \}$ non ha circuiti **then** $E' := E' \cup \{ e_j \};$

until $|E'| = n - 1;$

end.

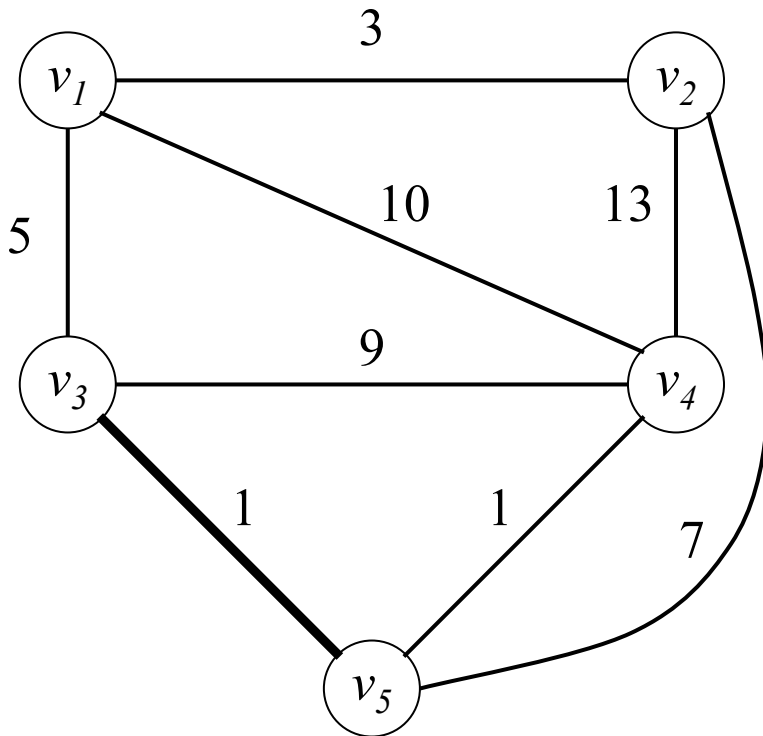
Complessità: $O(m \log m)$

Esempio



e_j	$c(e_j)$
(v_3, v_5)	1
(v_4, v_5)	1
(v_1, v_2)	3
(v_1, v_3)	5
(v_2, v_5)	7
(v_3, v_4)	9
(v_1, v_4)	10
(v_2, v_4)	13

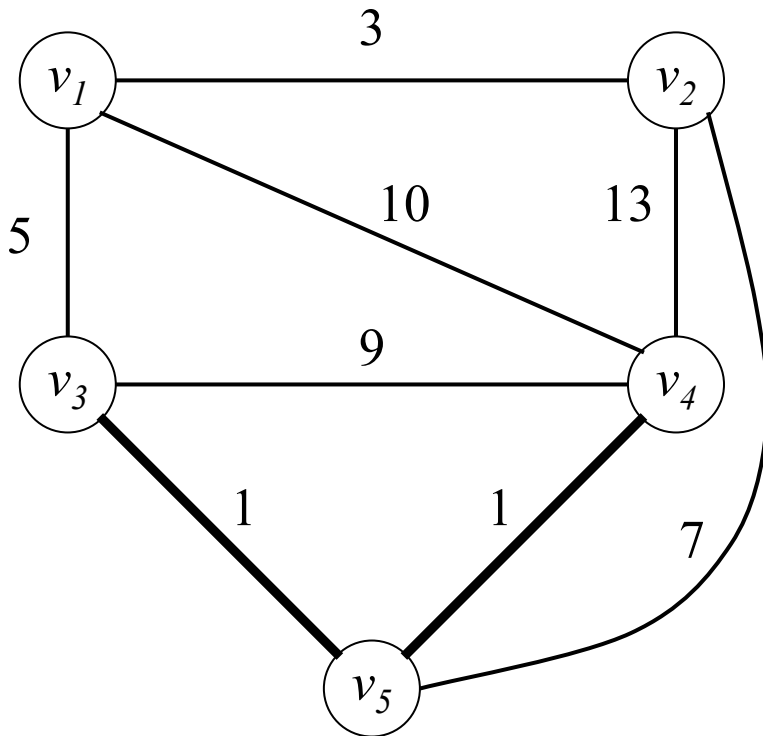
Iterazione 1



e_j	$c(e_j)$
(v_3, v_5)	1
(v_4, v_5)	1
(v_1, v_2)	3
(v_1, v_3)	5
(v_2, v_5)	7
(v_3, v_4)	9
(v_1, v_4)	10
(v_2, v_4)	13

$(v_3, v_5) \leftarrow 1$

Iterazione 2

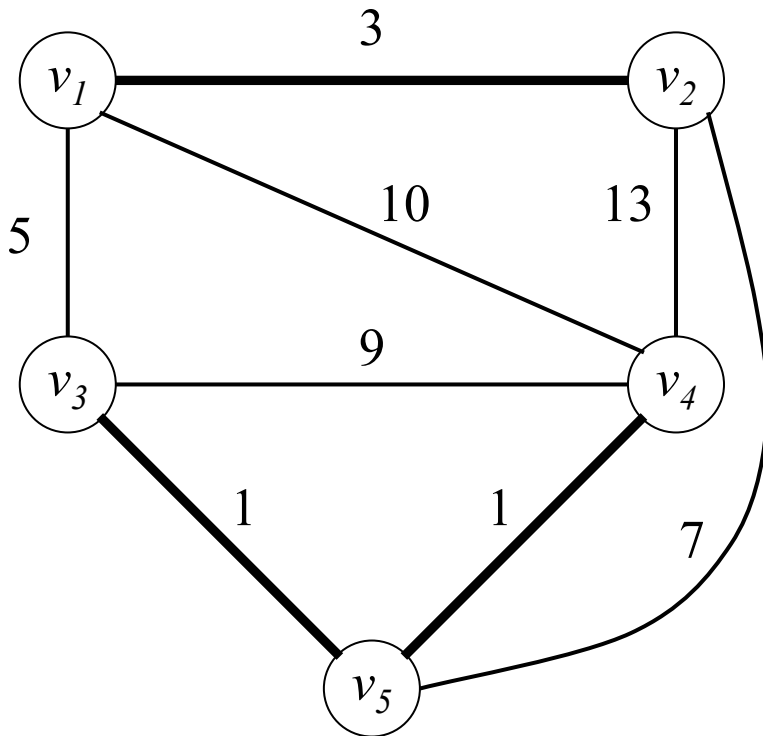


e_j	$c(e_j)$
(v_4, v_5)	1
(v_1, v_2)	3
(v_1, v_3)	5
(v_2, v_5)	7
(v_3, v_4)	9
(v_1, v_4)	10
(v_2, v_4)	13

(v_3, v_5)

$(v_4, v_5) \leftarrow 2$

Iterazione 3



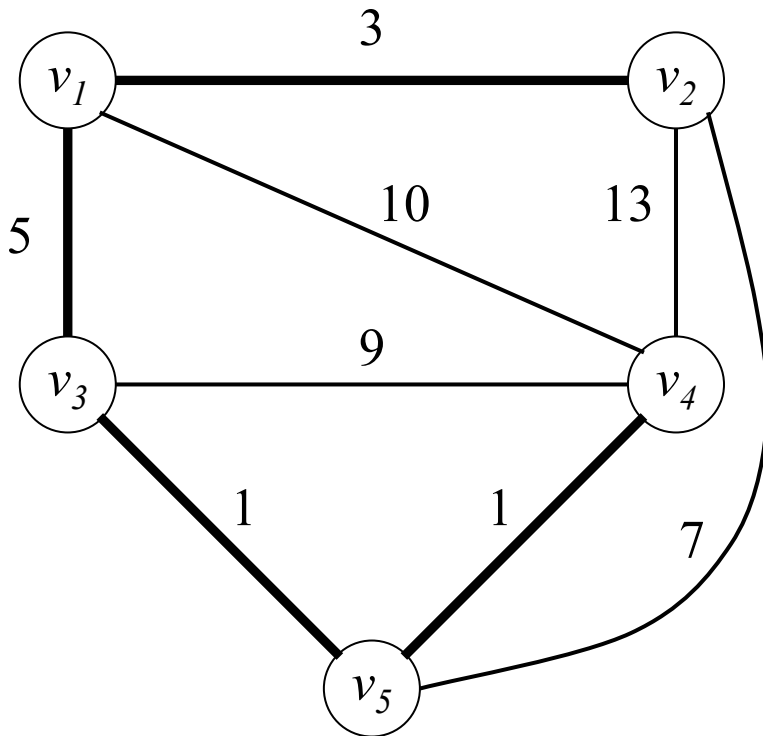
e_j	$c(e_j)$
(v_1, v_2)	3
(v_1, v_3)	5
(v_2, v_5)	7
(v_3, v_4)	9
(v_1, v_4)	10
(v_2, v_4)	13

(v_3, v_5)

(v_4, v_5)

$(v_1, v_2) \leftarrow 3$

Iterazione 4



e_j	$c(e_j)$
(v_1, v_3)	5
(v_2, v_5)	7
(v_3, v_4)	9
(v_1, v_4)	10
(v_2, v_4)	13

(v_3, v_5)

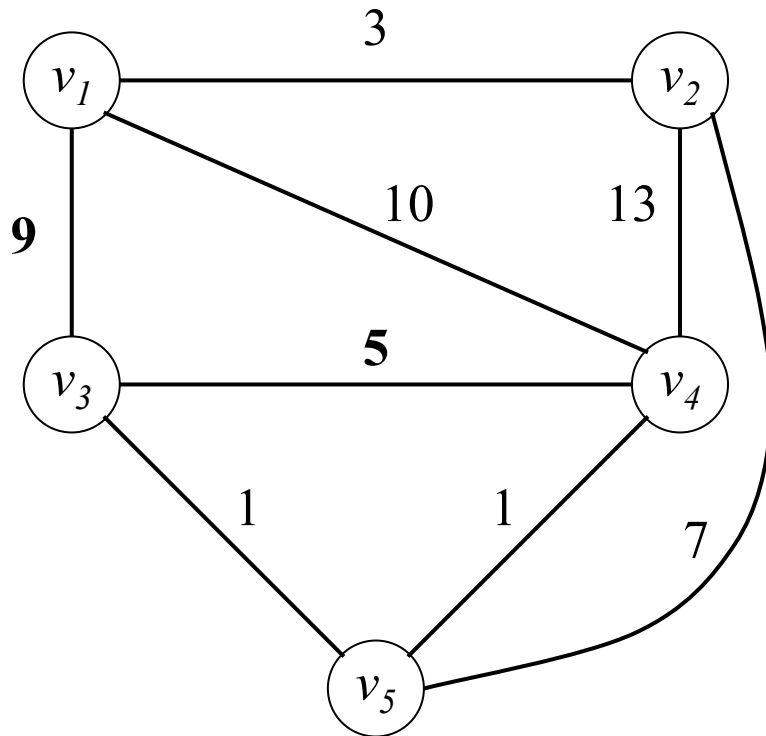
(v_4, v_5)

(v_1, v_2)

$(v_1, v_3) \leftarrow n-1$

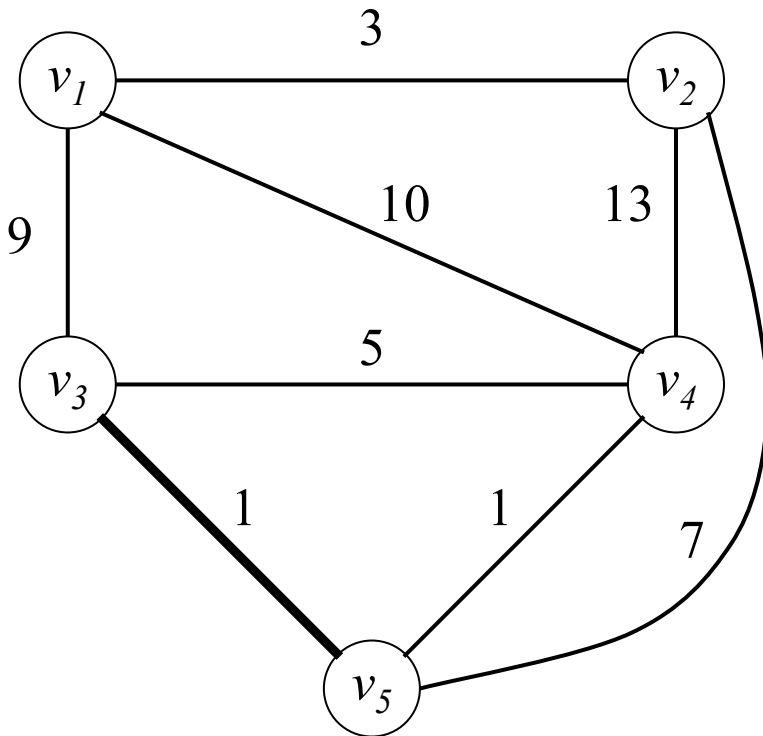
Costo = 10

Esempio: variante



e_j	$c(e_j)$
(v_3, v_5)	1
(v_4, v_5)	1
(v_1, v_2)	3
(v_3, v_4)	5
(v_2, v_5)	7
(v_1, v_3)	9
(v_1, v_4)	10
(v_2, v_4)	13

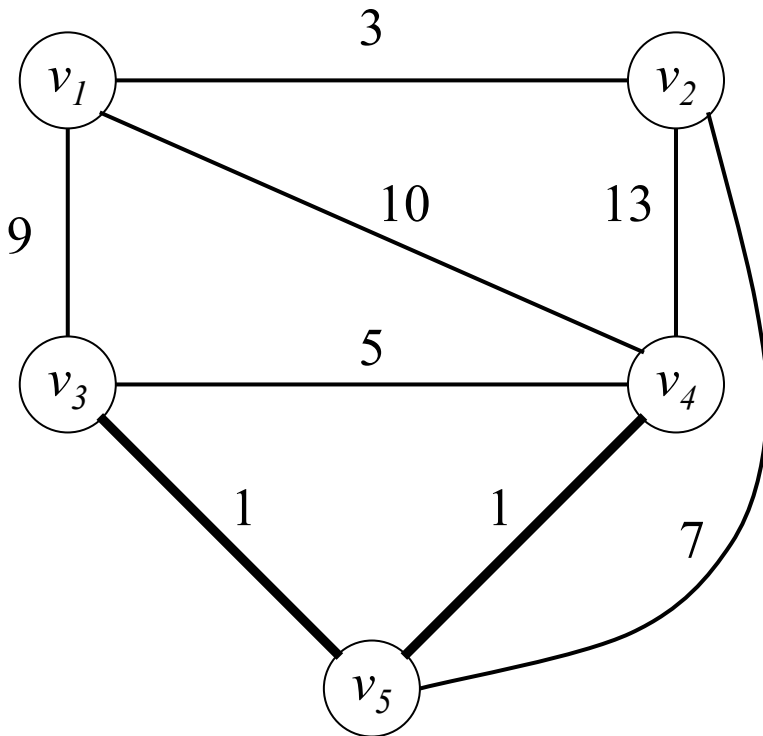
Iterazione 1



e_j	$c(e_j)$
(v_3, v_5)	1
(v_4, v_5)	1
(v_1, v_2)	3
(v_3, v_4)	5
(v_2, v_5)	7
(v_1, v_3)	9
(v_1, v_4)	10
(v_2, v_4)	13

$(v_3, v_5) \leftarrow 1$

Iterazione 2

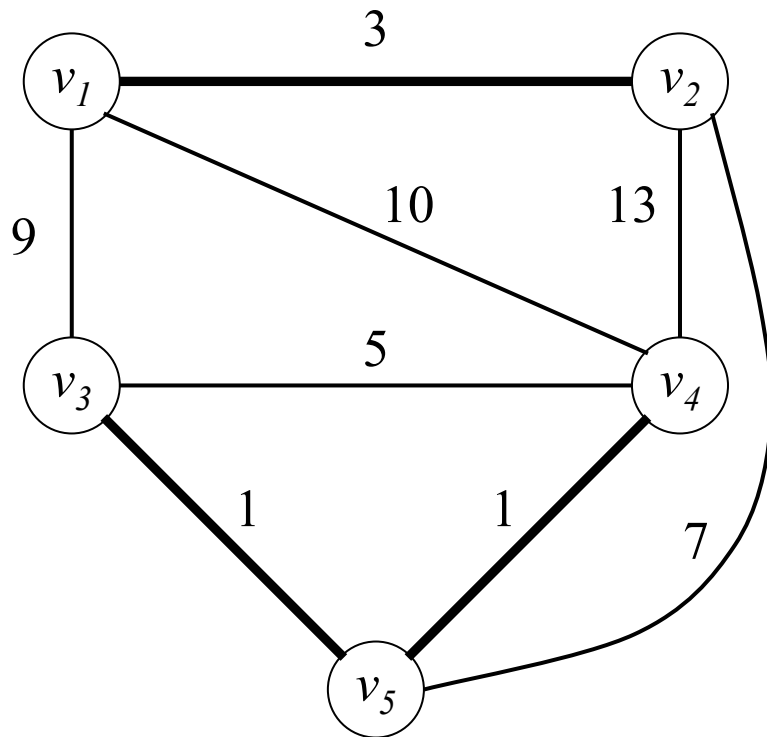


e_j	$c(e_j)$
(v_4, v_5)	1
(v_1, v_2)	3
(v_3, v_4)	5
(v_2, v_5)	7
(v_1, v_3)	9
(v_1, v_4)	10
(v_2, v_4)	13

(v_3, v_5)

$(v_4, v_5) \leftarrow 2$

Iterazione 3



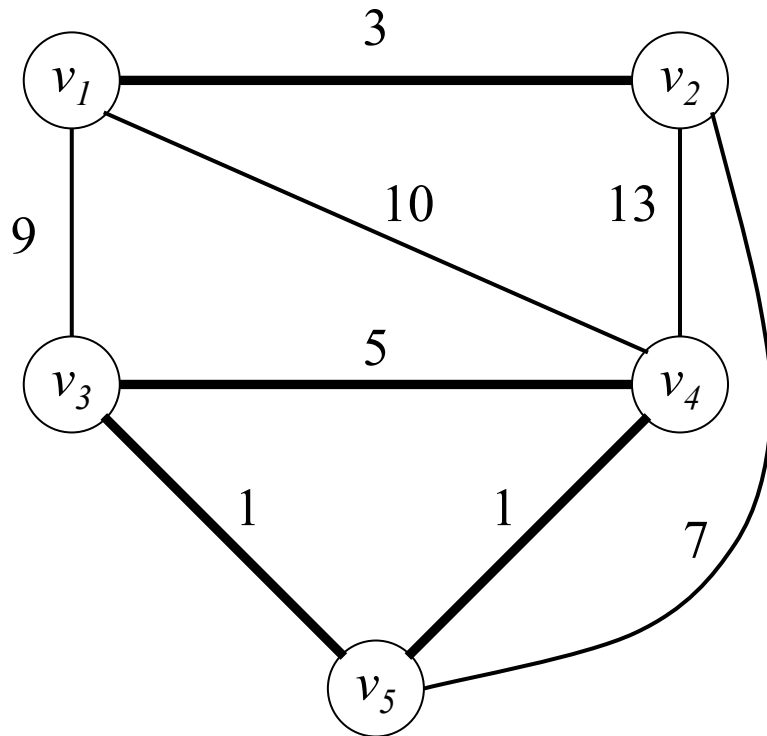
e_j	$c(e_j)$
(v_1, v_2)	3
(v_3, v_4)	5
(v_2, v_5)	7
(v_1, v_3)	9
(v_1, v_4)	10
(v_2, v_4)	13

(v_3, v_5)

(v_4, v_5)

$(v_1, v_2) \leftarrow 3$

Iterazione 4



e_j	$c(e_j)$
(v_3, v_4)	5
(v_2, v_5)	7
(v_1, v_3)	9
(v_1, v_4)	10
(v_2, v_4)	13

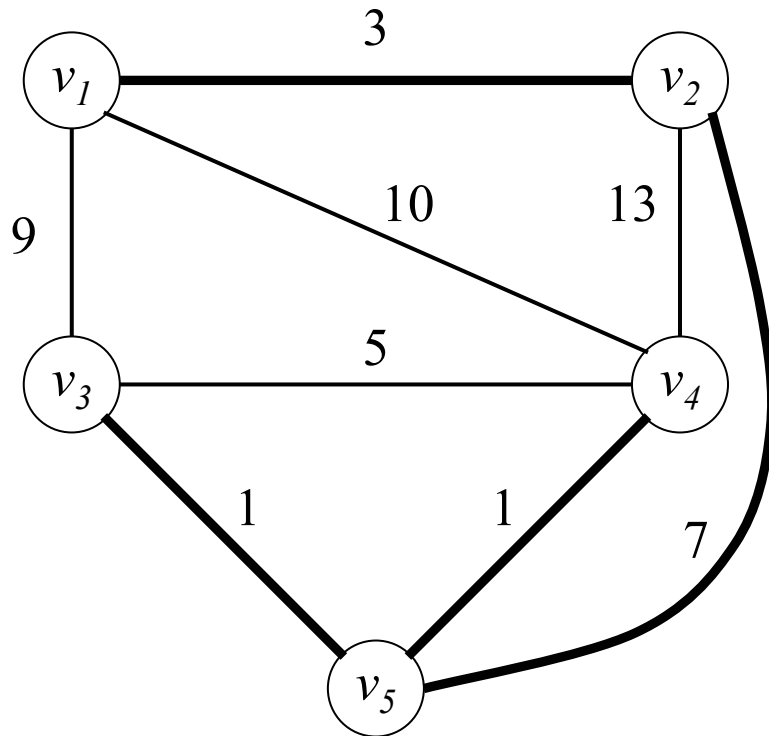
(v_3, v_5)

(v_4, v_5)

(v_1, v_2)

(v_3, v_4) NO

Iterazione 5



e_j	$c(e_j)$
(v_2, v_5)	7
(v_1, v_3)	9
(v_1, v_4)	10
(v_2, v_4)	13

(v_3, v_5)

(v_4, v_5)

(v_1, v_2)

$(v_2, v_5) \leftarrow n-1$

Costo = 12

Confronto Prim-Kruskal

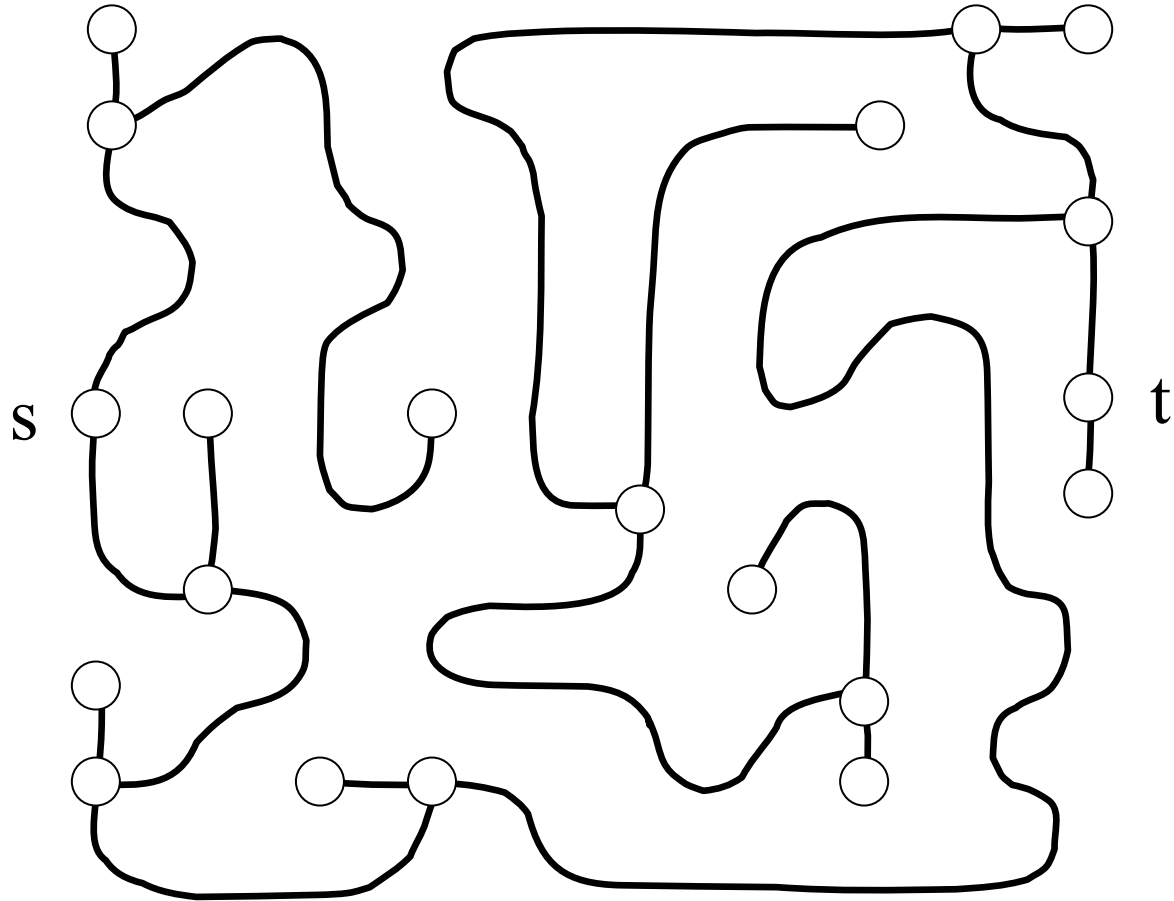
- Complessità
 - Prim : $O(n^2)$
 - Kruskal : $O(m \log m)$
- Per grafi densi $m \cong n^2$
 - \Rightarrow meglio Prim $O(n^2) < O(n^2 \log n)$
- Per grafi sparsi $m \ll n^2$
 - \Rightarrow meglio Kruskal $O(n \log n) < O(n^2)$

Cammini

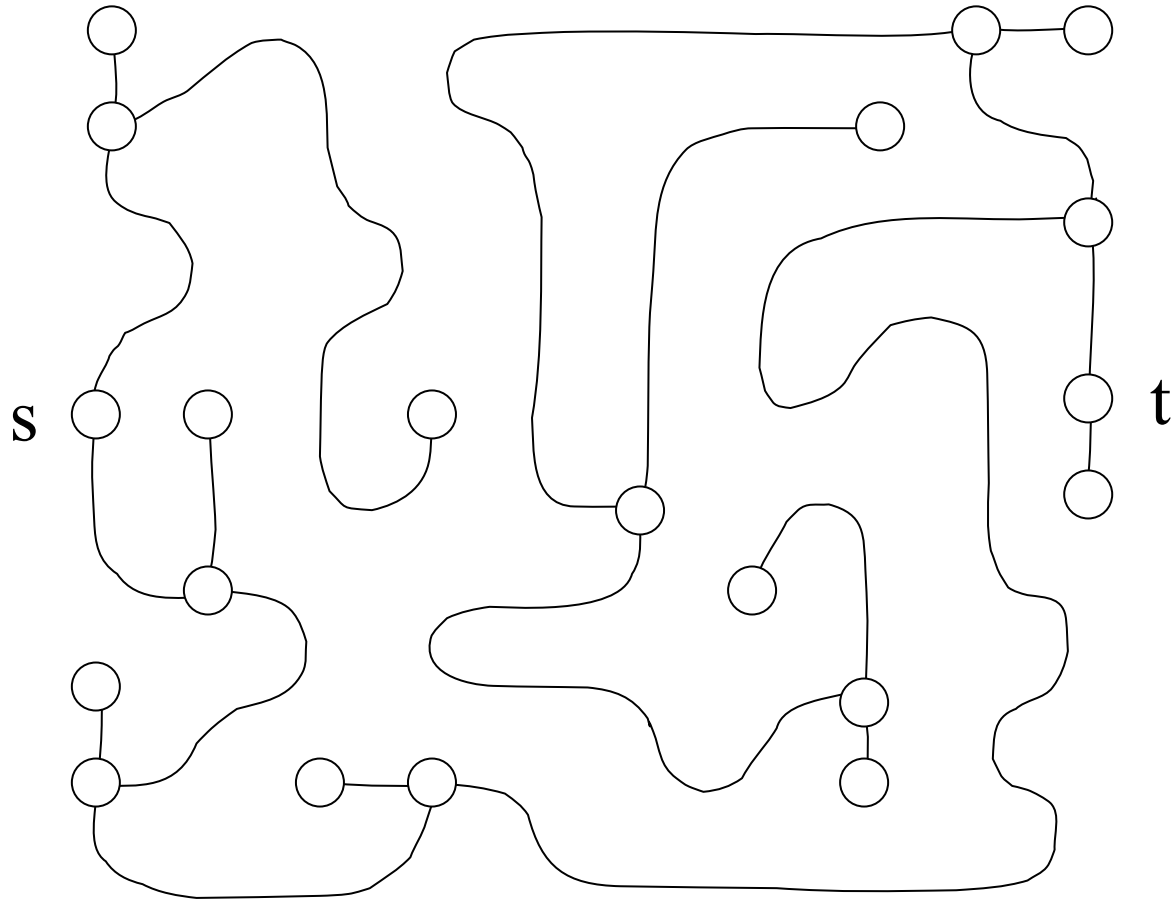
- Ricerca di cammini su grafo:
 - verifica esistenza di un percorso da un vertice ad un altro (uscita da un labirinto)
 - ricerca del percorso (di costo minimo) tra due località in un grafo (pesato) rappresentante una rete stradale
- Considereremo grafi orientati:
 - \Rightarrow maggiore generalità

(gli algoritmi presentati si adattano anche ai grafi non orientati)

Esempio

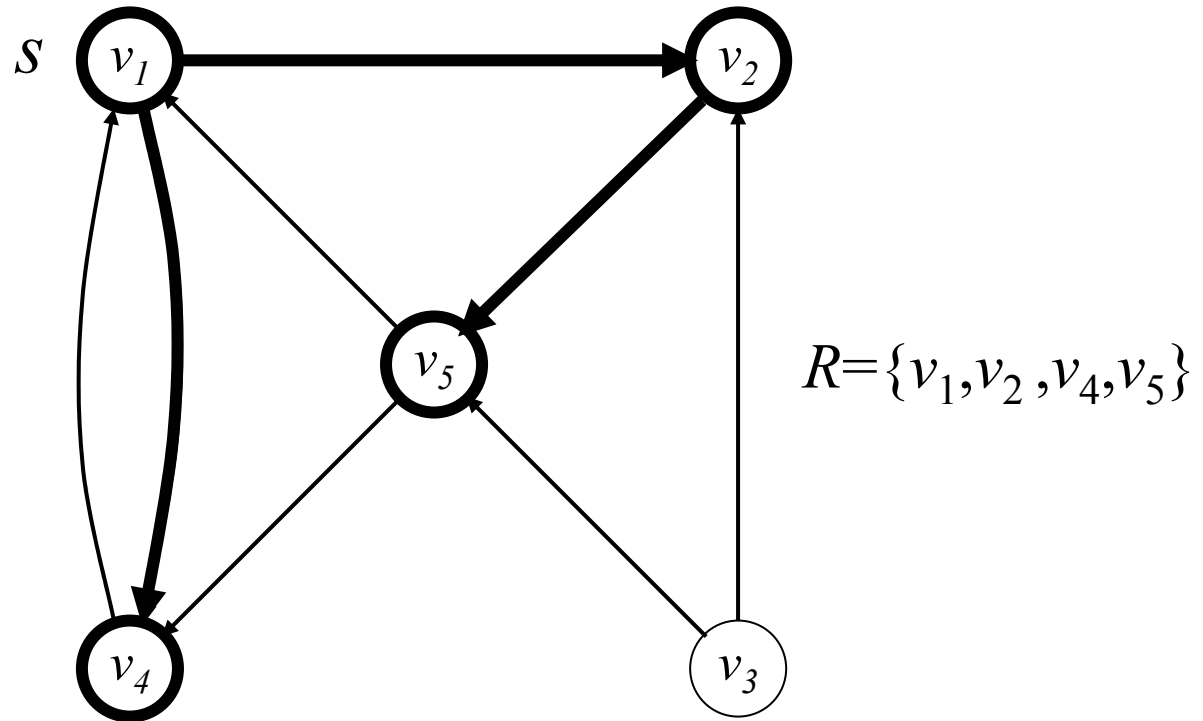


Esempio



Raggiungibilità

- Dato un grafo orientato $G=(V,A)$ determinare l'insieme R dei vertici raggiungibili da un vertice s assegnato



Raggiungibilità

Strutture dati

- R insieme dei vertici raggiungibili da s
 - $pred(j) =$ vertice che precede j
in un cammino da s a j
- si noti che $R = \{j : pred(j) \neq 0\}$
- $Q =$ insieme dei vertici raggiungibili da s
e non ancora elaborati

Algoritmo CAMMINI

- parti dal vertice s e marca tutti i vertici come non raggiunti

$pred(j) := 0$ ($j=1, \dots, n$); $R := \emptyset$;

- poni $Q := \{ s \}$
(insieme dei vertici raggiunti e non esaminati)
- per ogni vertice h di Q :
 - marca come raggiunti tutti i vertici j collegati ad h ($j \in \Gamma^+(h)$) e non raggiunti ($pred(j) = 0$)
 - inserisci tali vertici in Q

Algoritmo CAMMINI

begin

for $j:=1$ **to** n **do** $pred(j) := 0$;

$pred(s) := s$; $Q := \{ s \}$; $R := \emptyset$;

while $Q \neq \emptyset$ **do** $\{ \text{vertice raggiungibile } h \in Q \}$

 scegli vertice $h \in Q$; $Q := Q \setminus \{ h \}$;

for each $j \in \Gamma^+(h)$ **do**

if $pred(j) = 0$ **then**

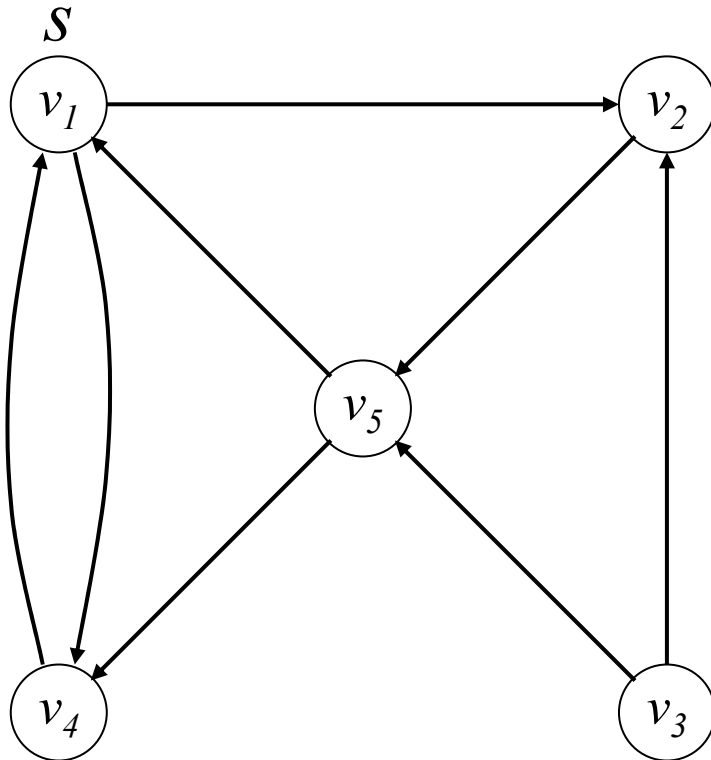
$pred(j) := h$; $Q := Q \cup \{ j \}$; $R := R \cup \{ j \}$;

end

end.

Complessità: $O(m)$

Esempio



procedure CAMMINI

begin

for $j:=1$ **to** n **do** $pred(j) := 0$;

$pred(s) := s$; $Q := \{ s \}$; $R := \emptyset$;

while $Q \neq \emptyset$ **do**

 scegli vertice $h \in Q$; $Q := Q \setminus \{ h \}$;

for each $j \in \Gamma^+(h)$ **do**

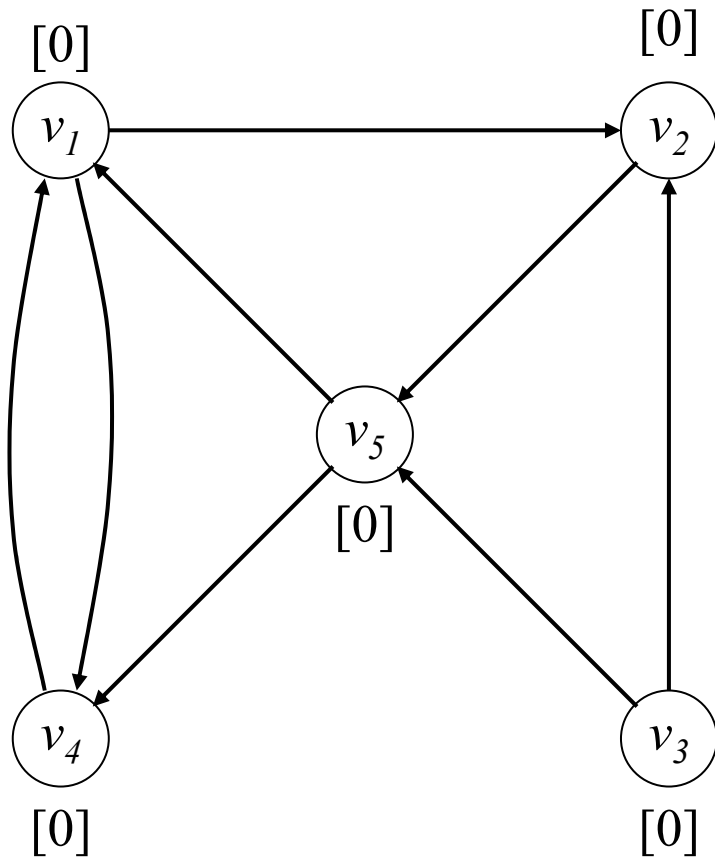
if $pred(j) = 0$ **then**

$pred(j) := h$; $Q := Q \cup \{ j \}$; $R := R \cup \{ j \}$;

end

end.

Inizializzazione etichette



procedure CAMMINI

begin

for $j:=1$ **to** n **do** $pred(j) := 0$;

$pred(s) := s$; $Q := \{ s \}$; $R := \emptyset$;

while $Q \neq \emptyset$ **do**

 scegli vertice $h \in Q$; $Q := Q \setminus \{ h \}$;

for each $j \in \Gamma^+(h)$ **do**

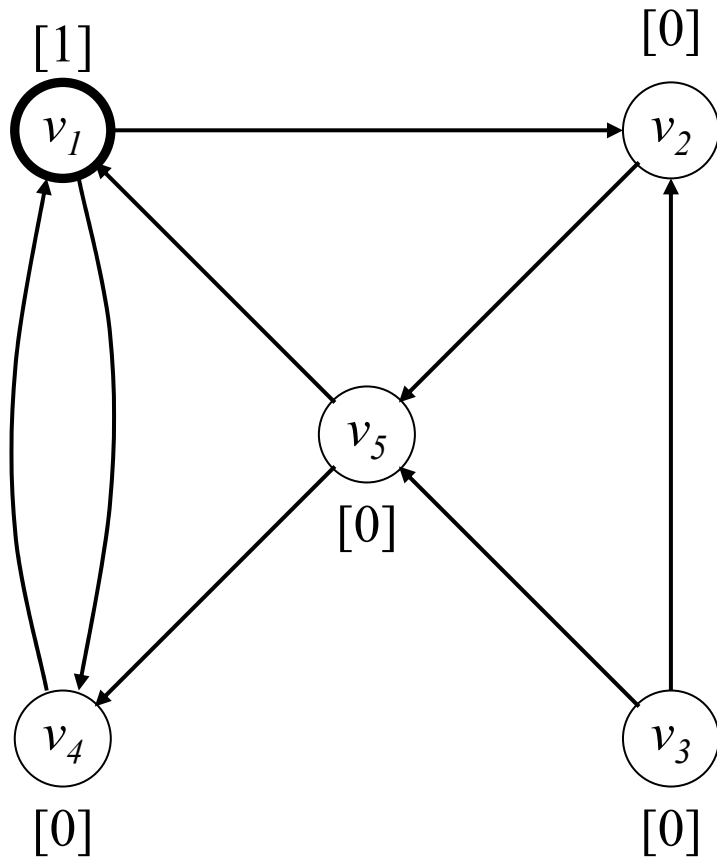
if $pred(j) = 0$ **then**

$pred(j) := h$; $Q := Q \cup \{ j \}$; $R := R \cup \{ j \}$;

end

end.

Inizializzazione insiemi



procedure CAMMINI

begin

for $j:=1$ **to** n **do** $pred(j) := 0$;

$pred(s) := s$; $Q := \{ s \}$; $R := \emptyset$;

while $Q \neq \emptyset$ **do**

 scegli vertice $h \in Q$; $Q := Q \setminus \{ h \}$;

for each $j \in \Gamma^+(h)$ **do**

if $pred(j) = 0$ **then**

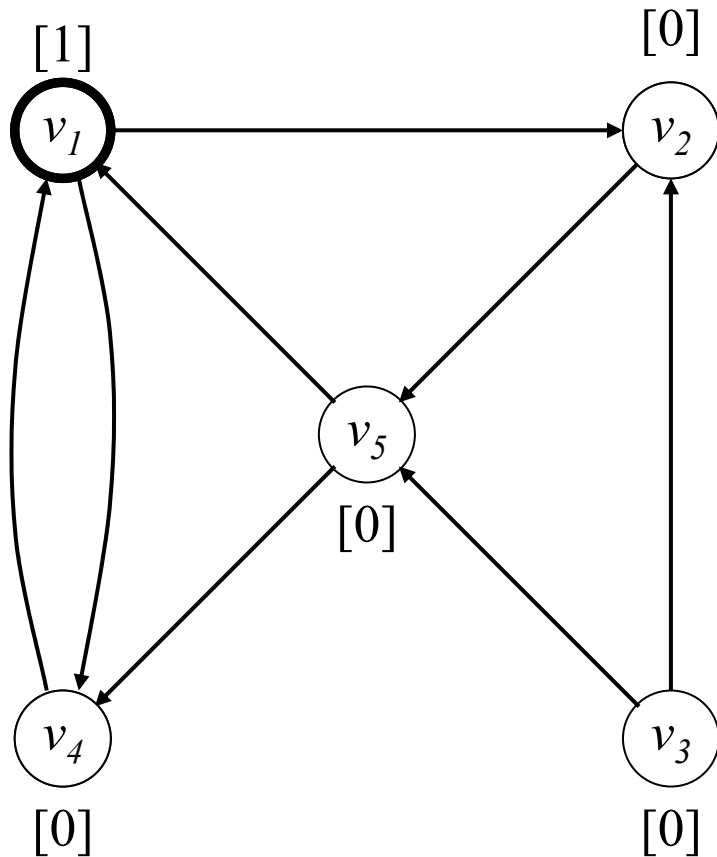
$pred(j) := h$; $Q := Q \cup \{ j \}$; $R := R \cup \{ j \}$;

end

end.

- $Q = \{ v_1 \}$
- $R = \emptyset$

Selezione vertice (1)



procedure CAMMINI

begin

for $j:=1$ **to** n **do** $pred(j) := 0$;

$pred(s) := s$; $Q := \{ s \}$; $R := \emptyset$;

while $Q \neq \emptyset$ **do**

 scegli vertice $h \in Q$; $Q := Q \setminus \{ h \}$;

for each $j \in \Gamma^+(h)$ **do**

if $pred(j) = 0$ **then**

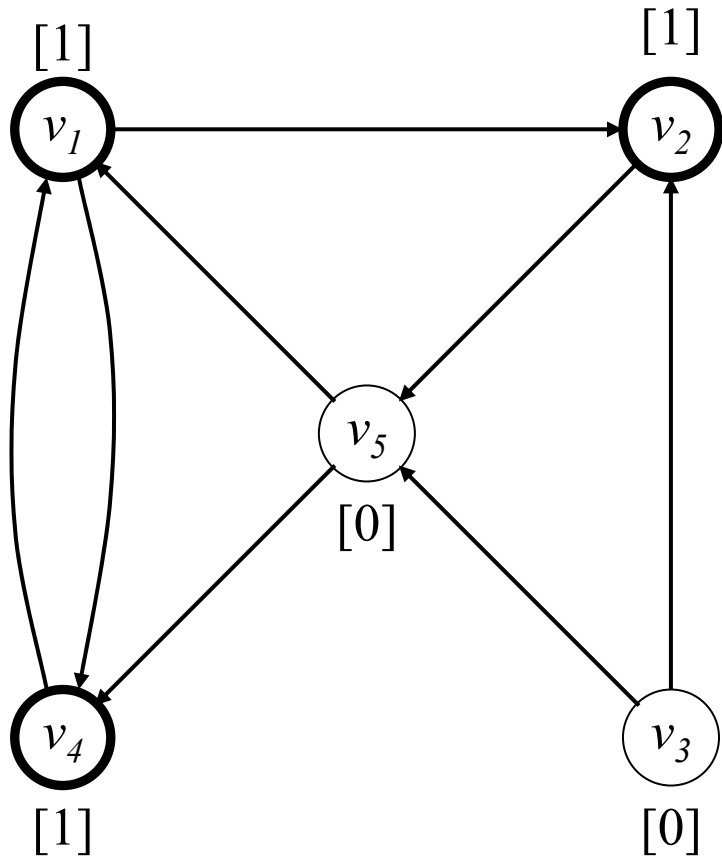
$pred(j) := h$; $Q := Q \cup \{ j \}$; $R := R \cup \{ j \}$;

end

end.

- $Q = \emptyset$
- $R = \emptyset$
- $\Gamma^+(v_1) = \{ v_2, v_4 \}$

Aggiornamento (1)



procedure CAMMINI

begin

for $j:=1$ **to** n **do** $pred(j) := 0$;

$pred(s) := s$; $Q := \{ s \}$; $R := \emptyset$;

while $Q \neq \emptyset$ **do**

 scegli vertice $h \in Q$; $Q := Q \setminus \{ h \}$;

for each $j \in \Gamma^+(h)$ **do**

if $pred(j) = 0$ **then**

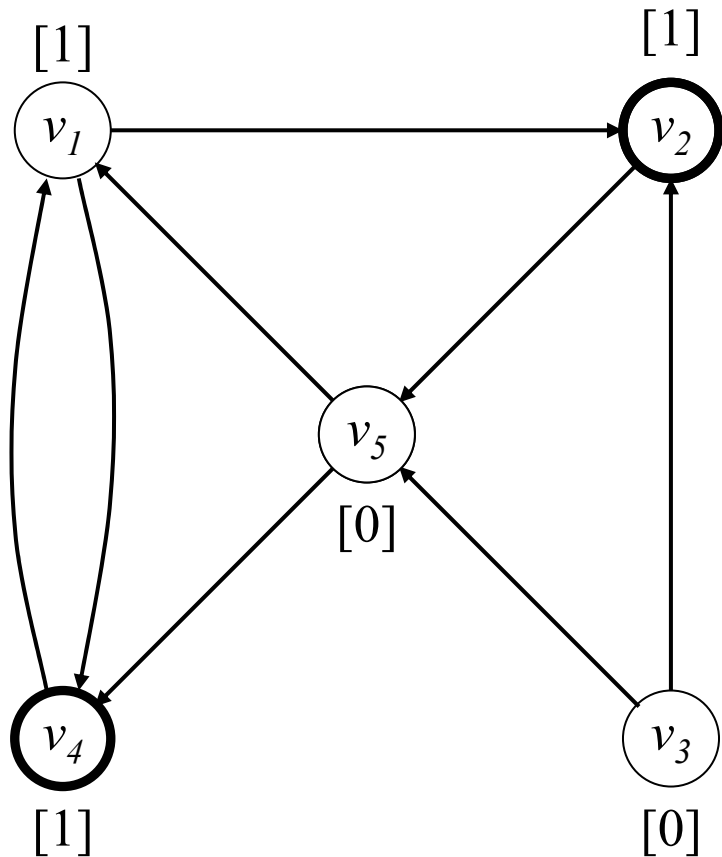
$pred(j) := h$; $Q := Q \cup \{ j \}$; $R := R \cup \{ j \}$;

end

end.

- $Q = \{ v_2, v_4 \}$
- $R = \{ v_2, v_4 \}$
- $\Gamma^+(v_1) = \{ v_2, v_4 \}$

Selezione vertice (2)



procedure CAMMINI

begin

for $j:=1$ **to** n **do** $pred(j) := 0$;

$pred(s) := s$; $Q := \{ s \}$; $R := \emptyset$;

while $Q \neq \emptyset$ **do**

 scegli vertice $h \in Q$; $Q := Q \setminus \{ h \}$;

for each $j \in \Gamma^+(h)$ **do**

if $pred(j) = 0$ **then**

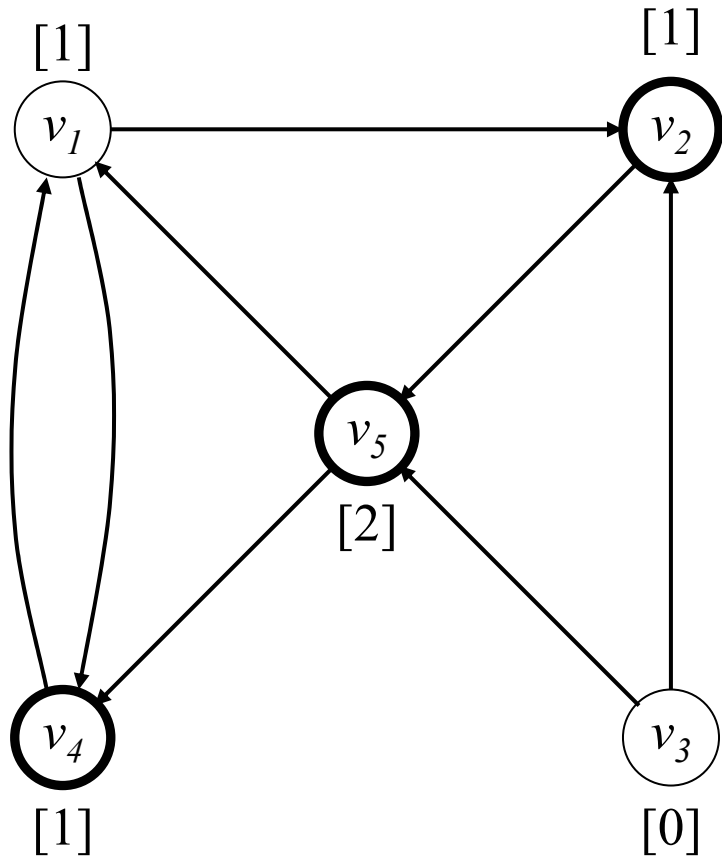
$pred(j) := h$; $Q := Q \cup \{ j \}$; $R := R \cup \{ j \}$;

end

end.

- $Q = \{ v_4 \}$
- $R = \{ v_2, v_4 \}$
- $\Gamma^+(v_2) = \{ v_5 \}$

Aggiornamento (2)



procedure CAMMINI

begin

for $j:=1$ **to** n **do** $pred(j) := 0$;

$pred(s) := s$; $Q := \{ s \}$; $R := \emptyset$;

while $Q \neq \emptyset$ **do**

 scegli vertice $h \in Q$; $Q := Q \setminus \{ h \}$;

for each $j \in \Gamma^+(h)$ **do**

if $pred(j) = 0$ **then**

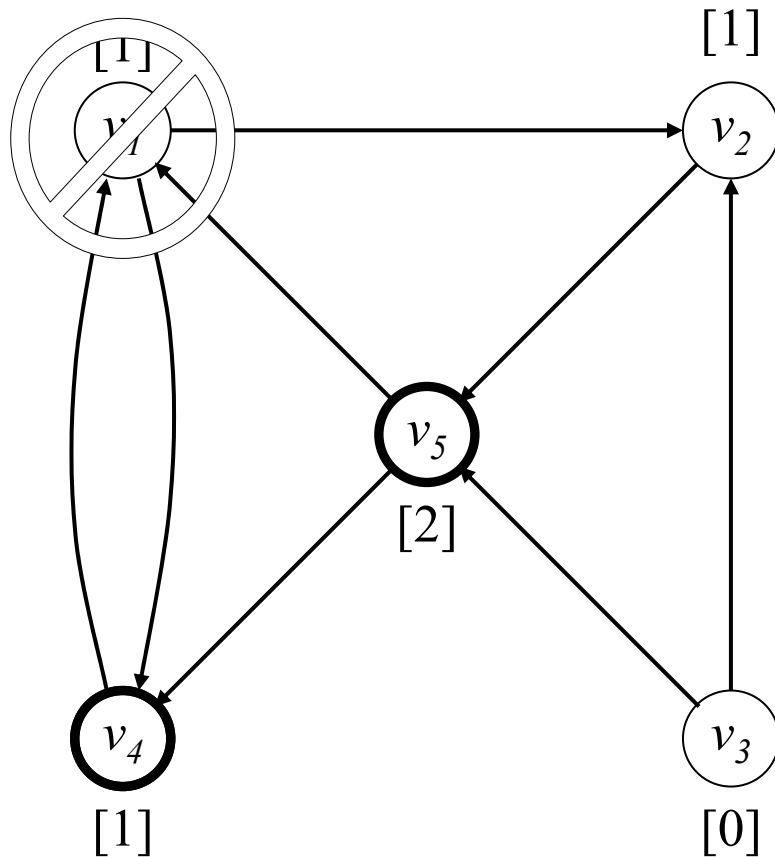
$pred(j) := h$; $Q := Q \cup \{ j \}$; $R := R \cup \{ j \}$;

end

end.

- $Q = \{ v_4, v_5 \}$
- $R = \{ v_2, v_4, v_5 \}$
- $\Gamma^+(v_2) = \{ v_5 \}$

Selezione vertice (3)



procedure CAMMINI

begin

for $j:=1$ **to** n **do** $pred(j) := 0;$

$pred(s) := s; Q := \{ s \}; R := \emptyset;$

while $Q \neq \emptyset$ **do**

 scegli vertice $h \in Q; Q := Q \setminus \{ h \};$

for each $j \in \Gamma^+(h)$ **do**

if $pred(j) = 0$ **then**

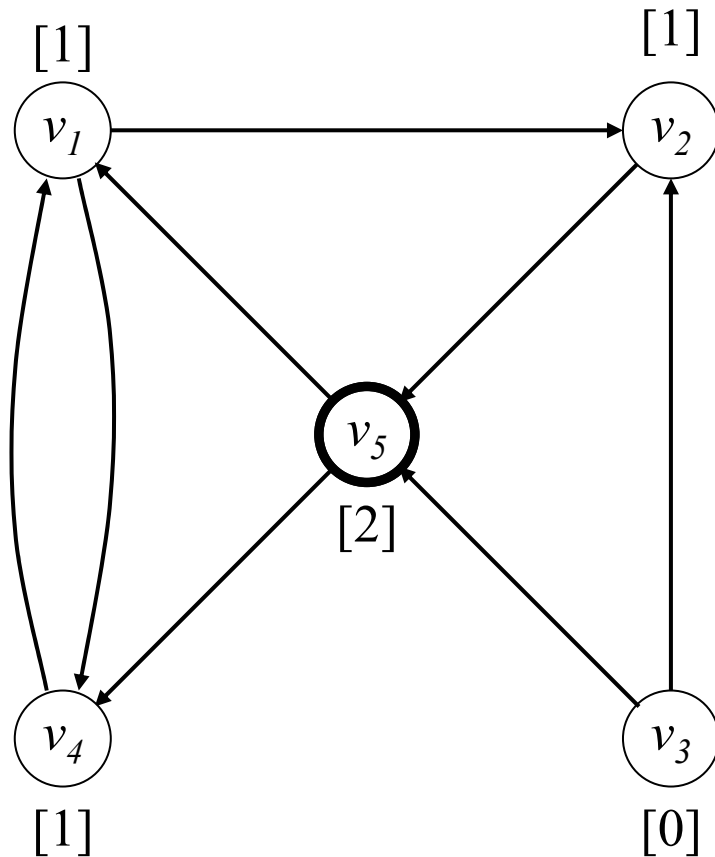
$pred(j) := h; Q := Q \cup \{ j \}; R := R \cup \{ j \};$

end

end.

- $Q = \{ v_5 \}$
- $R = \{ v_2, v_4, v_5 \}$
- $\Gamma^+(v_4) = \{ v_1 \}$

Selezione vertice (4)



procedure CAMMINI

begin

for $j:=1$ **to** n **do** $pred(j) := 0$;

$pred(s) := s$; $Q := \{ s \}$; $R := \emptyset$;

while $Q \neq \emptyset$ **do**

 scegli vertice $h \in Q$; $Q := Q \setminus \{ h \}$;

for each $j \in \Gamma^+(h)$ **do**

if $pred(j) = 0$ **then**

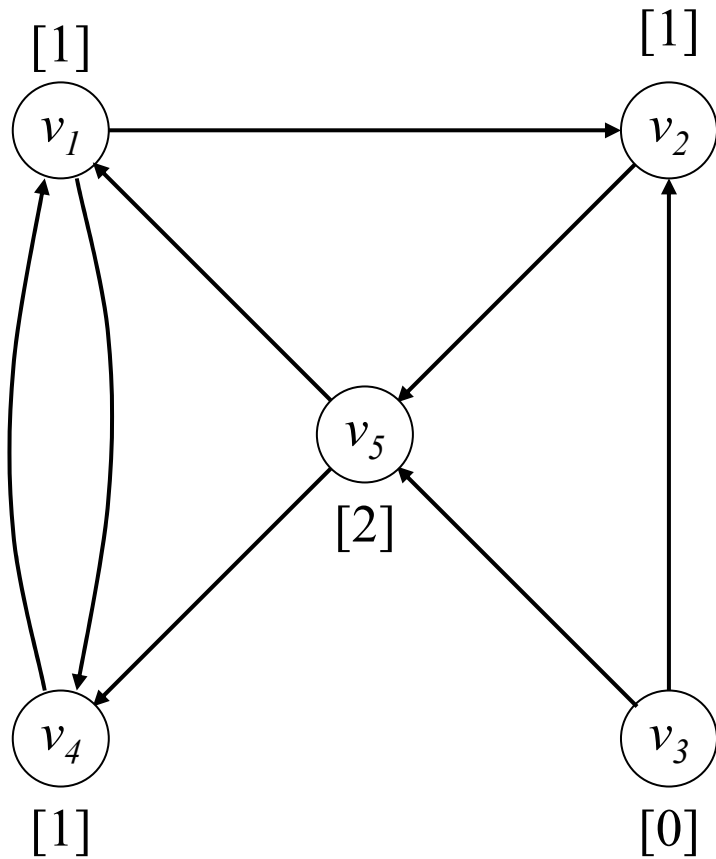
$pred(j) := h$; $Q := Q \cup \{ j \}$; $R := R \cup \{ j \}$;

end

end.

- $Q = \emptyset$
- $R = \{ v_2, v_4, v_5 \}$
- $\Gamma^+(v_5) = \{ v_1, v_4 \}$

Soluzione



$$S = v_1$$

$$R = \{ v_2, v_4, v_5 \}$$

$$v_2 \leftarrow v_1$$

v_3 non raggiunto

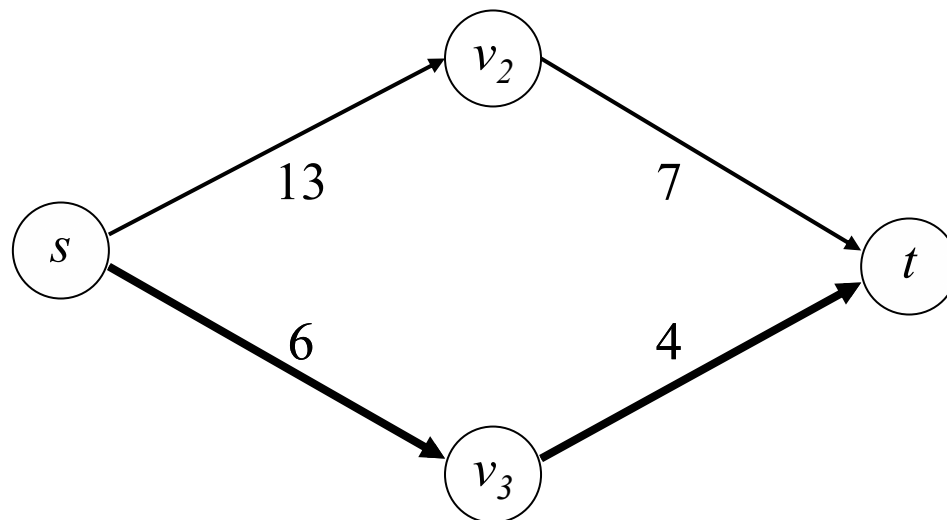
$$v_4 \leftarrow v_1$$

$$v_5 \leftarrow v_2 \leftarrow v_1$$

Problema del cammino minimo

Shortest Path Problem, SPP

- Dato un grafo orientato $G=(V,A)$, pesato sugli archi con costi c_{ij} , e due vertici $s,t \in V$, determinare il cammino semplice di costo minimo dal vertice s al vertice t



Complessità SPP

Th.: Se i costi degli archi sono qualsiasi, SPP è NP-hard

Dim.:

- Un cammino semplice da un vertice a se stesso ha al più n archi e se ne ha esattamente n è un circuito hamiltoniano
- Determinare se $G=(V,A)$ possiede un circuito hamiltoniano è NP-completo
- Dato un grafo $G=(V,A)$ definendo $c_{ij} = -1$ per ogni $(i,j) \in A$, se il cammino semplice di costo minimo da un qualunque vertice v a se stesso ha costo $-n$, tale cammino è un circuito hamiltoniano

Casi particolari

Esistono casi in cui il problema è polinomiale

- $c_{ij} \geq 0$ per ogni $(i, j) \in A$

Algoritmo di Dijkstra

complessità $O(n^2)$ – $O(n \log n)$

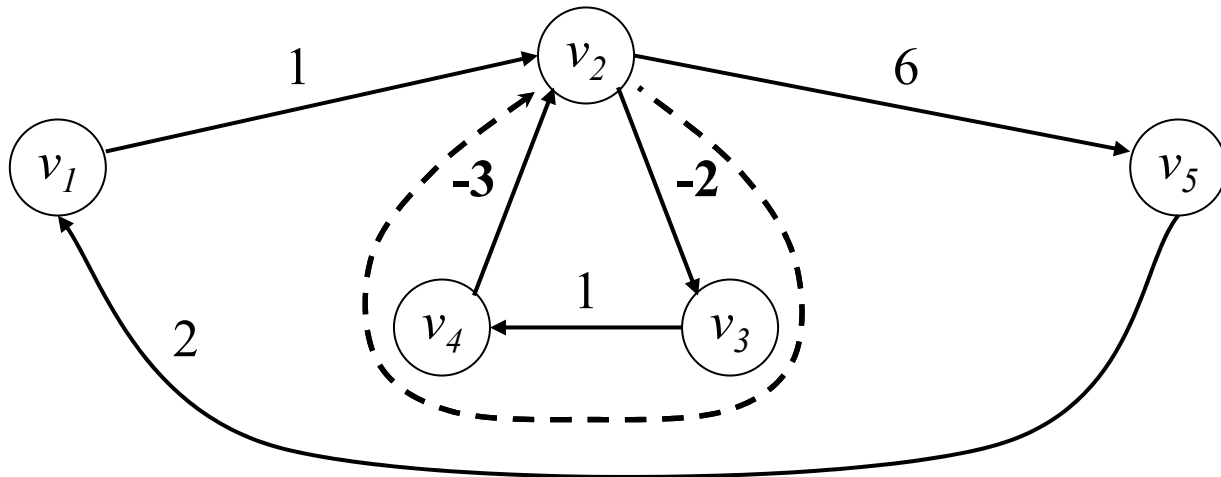
- c_{ij} qualsiasi per ogni $(i, j) \in A$, ma non esistono circuiti in G di costo negativo

Algoritmo di Floyd-Warshall

complessità $O(n^3)$

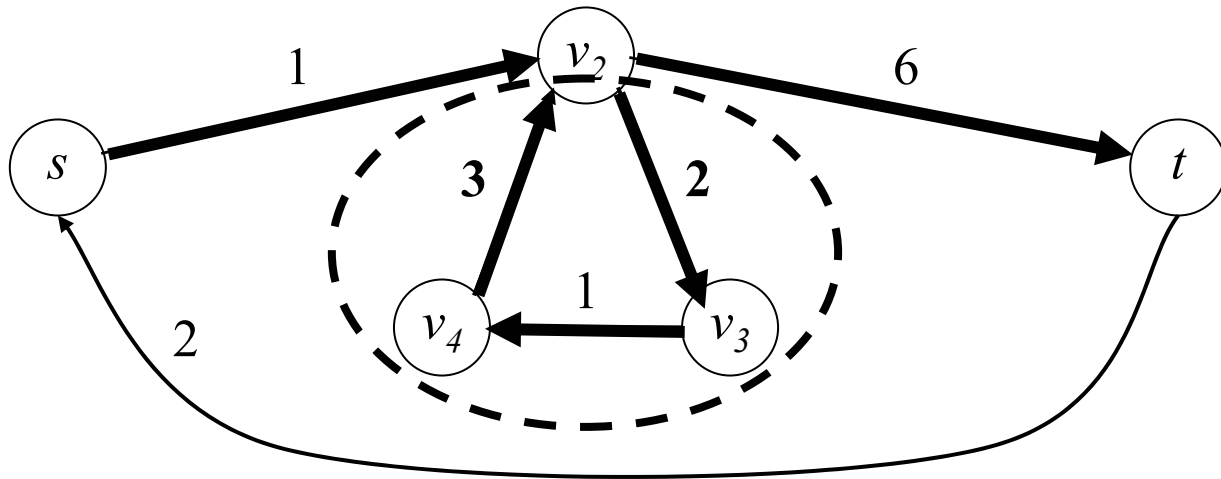
Proprietà SP (1)

- Se i costi possono essere negativi il cammino minimo non è necessariamente semplice ed elementare
- *Se esiste un circuito a costo negativo il cammino minimo lo percorre infinite volte*



Proprietà SP (2)

- Se i costi degli archi sono non negativi il cammino di costo minimo è semplice ed elementare



Concatenazione cammini

Propr.: Se il cammino minimo da v_i a v_k passa per v_j , allora esso è formato dal cammino minimo da v_i a v_j concatenato al cammino minimo da v_j a v_k

Dim.:

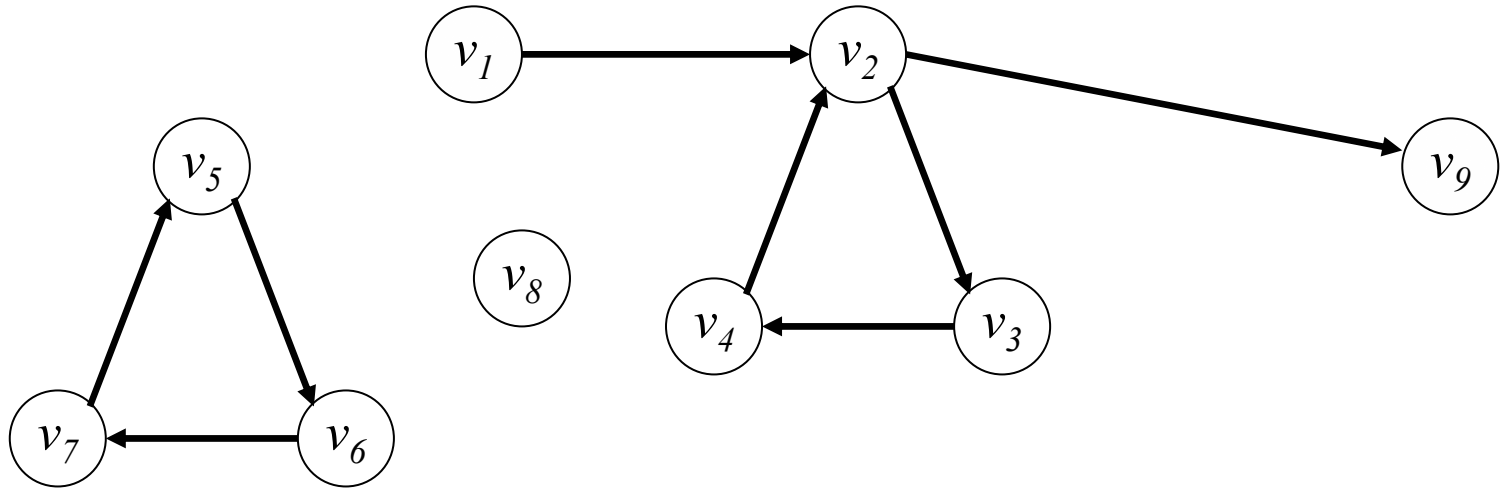
- Se esistesse un cammino più breve da v_i a v_j (o da v_j a v_k) esso verrebbe usato anche nel cammino minimo da v_i a v_k

Modello PLI di SPP

$$\begin{array}{l}
 \min \sum_{(i,j) \in A} c_{ij} x_{ij} \\
 \sum_{(h,j) \in \Gamma^+(h)} x_{hj} - \sum_{(i,h) \in \Gamma^-(h)} x_{ih} = \begin{cases} 1 & \text{se } h = s \\ -1 & \text{se } h = t \\ 0 & \text{se } h \in V \setminus \{s, t\} \end{cases} \\
 N. \text{ archi in } S \longrightarrow \sum_{(i,j): i,j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subseteq V, S \neq \emptyset \\
 0 \leq x_{ij} \leq 1 \quad \text{intero } (i,j) \in A
 \end{array}$$

Costo cammino
N. archi uscenti
N. archi entranti

Vincoli subtour elimination



$$\sum_{(i,j):i,j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subseteq V, S \neq \emptyset$$

Se $c_{ij} \geq 0$ il cammino minimo è semplice

\Rightarrow ridondanti

Algoritmo di Dijkstra (1959)

Strutture dati

- $W =$ insieme dei vertici raggiunti in modo permanente da s
- $L(v) =$ costo del cammino minimo da s a v passante solo per $j \in W$
- $pred(v) =$ vertice che precede v nel cammino da s a v

Procedure DIJKSTRA

begin

$W := \{ s \}; L(s) := 0; pred(s) := 0;$

for each $v \in V \setminus \{ s \}$ **do**

$L(v) := c(s, v); pred(v) := s;$

while $|W| < n$ **do**

trova $v^* \in V \setminus W : L(v^*) = \min_{v \in V \setminus W} \{ L(v) \};$

$W := W \cup \{ v^* \};$

for each $v \in V \setminus W$ **do**

if $L(v^*) + c(v^*, v) < L(v)$ **then**

$L(v) := L(v^*) + c(v^*, v); pred(v) := v^*;$

end

end.



- $n-1$ iterazioni
- \forall iterazione
numero operaz.
 $\propto a |V \setminus W|$
- tempo $O(n^2)$

Teorema

- Se $L(v^*) = \min_{v \in V \setminus W} \{ L(v) \}$ il cammino minimo da s a v^* è lungo $L(v^*)$

Dimostrazione

Dimostriamo che qualunque cammino P da s a v^* è lungo almeno $L(v^*)$:

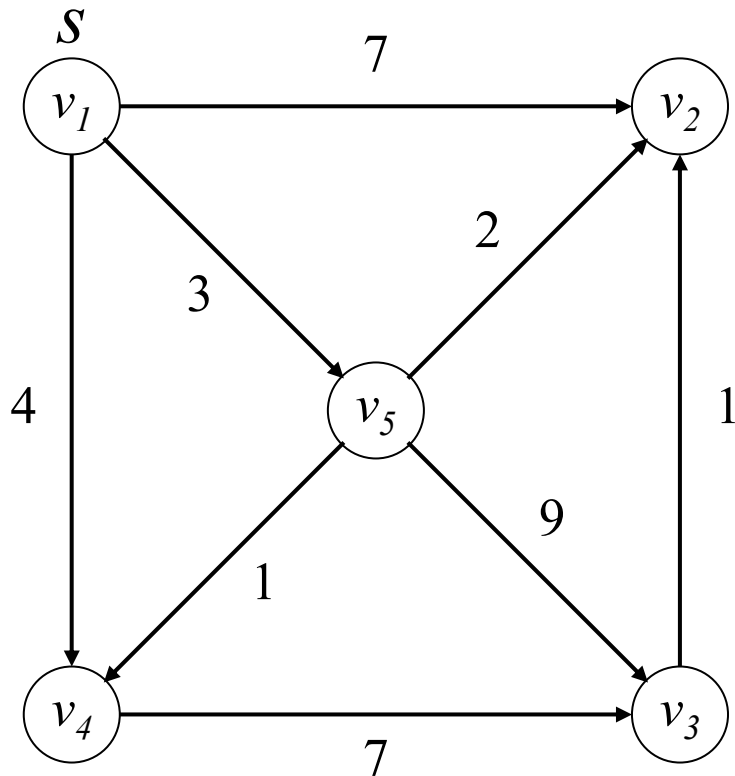
- Se P passa solo per vertici di W , vero per definizione di $L(\cdot)$;
- altrimenti sia h il primo vertice $\notin W$ che si incontra in P

$P \equiv$ cammino da s a h (lungo almeno $L(h) \geq L(v^*)$)

+ cammino da h a v^* di lunghezza ≥ 0

Dimostrazione non valida se ci sono distanze negative

Esempio



procedure DIJKSTRA

begin

$W := \{ s \}; L(s) := 0; pred(s) := 0;$

for each $v \in V \setminus \{ s \}$ **do**

$L(v) := c(s, v); pred(v) := s;$

while $|W| < n$ **do**

trova $v^* \in V \setminus W : L(v^*) = \min_{v \in V \setminus W} \{ L(v) \};$

$W := W \cup \{ v^* \};$

for each $v \in V \setminus W$ **do**

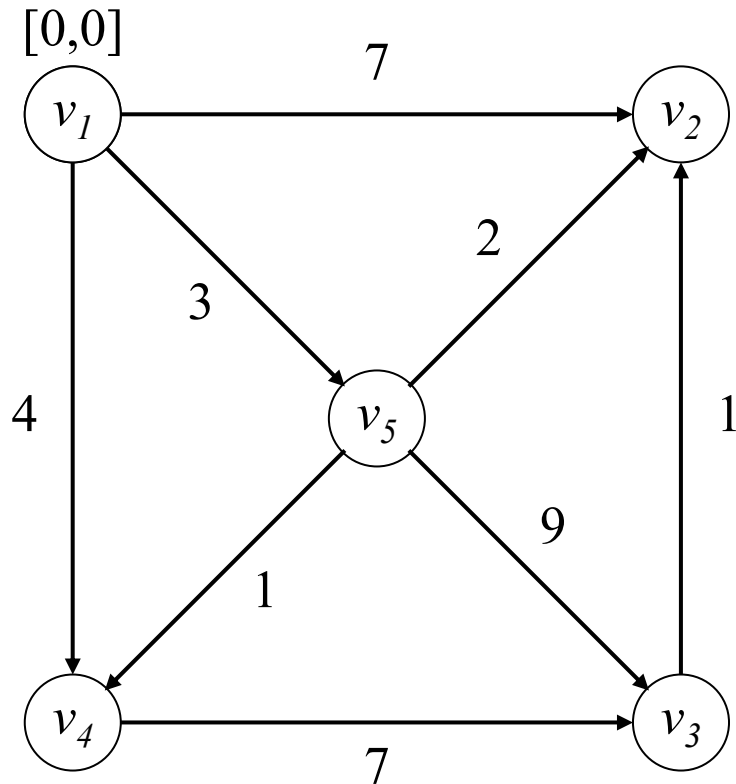
if $L(v^*) + c(v^*, v) < L(v)$ **then**

$L(v) := L(v^*) + c(v^*, v); pred(v) := v^*;$

end

end.

Inizializzazione



procedure DIJKSTRA

begin

$W := \{ s \}; L(s) := 0; pred(s) := 0;$

for each $v \in V \setminus \{ s \}$ **do**

$L(v) := c(s, v); pred(v) := s;$

while $|W| < n$ **do**

trova $v^* \in V \setminus W : L(v^*) = \min_{v \in V \setminus W} \{ L(v) \};$

$W := W \cup \{ v^* \};$

for each $v \in V \setminus W$ **do**

if $L(v^*) + c(v^*, v) < L(v)$ **then**

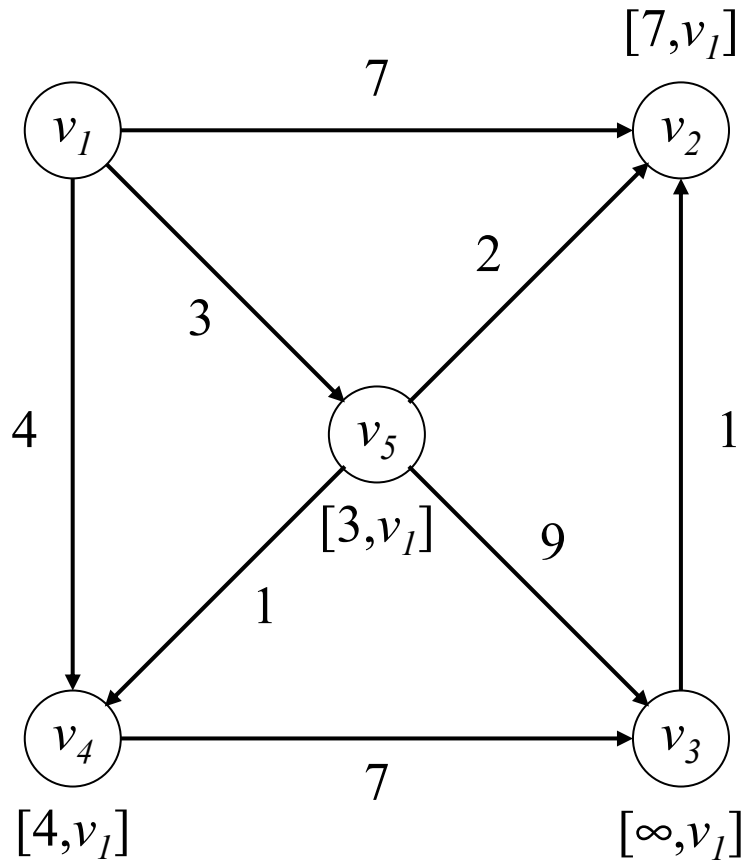
$L(v) := L(v^*) + c(v^*, v); pred(v) := v^*;$

end

end.

$$W = \{ v_1 \}$$

Inizializzazione etichette



procedure DIJKSTRA

begin

$W := \{ s \}; L(s) := 0; pred(s) := 0;$

for each $v \in V \setminus \{ s \}$ **do**

$L(v) := c(s, v); pred(v) := s;$

while $|W| < n$ **do**

trova $v^* \in V \setminus W : L(v^*) = \min_{v \in V \setminus W} \{ L(v) \};$

$W := W \cup \{ v^* \};$

for each $v \in V \setminus W$ **do**

if $L(v^*) + c(v^*, v) < L(v)$ **then**

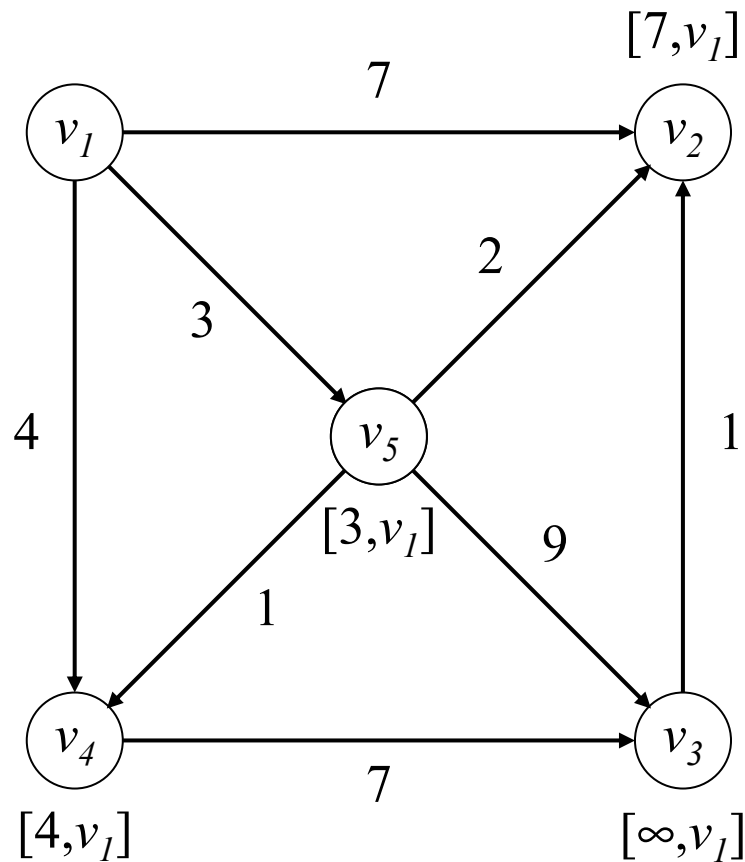
$L(v) := L(v^*) + c(v^*, v); pred(v) := v^*;$

end

end.

$$W = \{ v_1 \}$$

Selezione vertice (1)



procedure DIJKSTRA

begin

$W := \{ s \}; L(s) := 0; pred(s) := 0;$

for each $v \in V \setminus \{ s \}$ **do**

$L(v) := c(s, v); pred(v) := s;$

while $|W| < n$ **do**

trova $v^* \in V \setminus W : L(v^*) = \min_{v \in V \setminus W} \{ L(v) \};$

$W := W \cup \{ v^* \};$

for each $v \in V \setminus W$ **do**

if $L(v^*) + c(v^*, v) < L(v)$ **then**

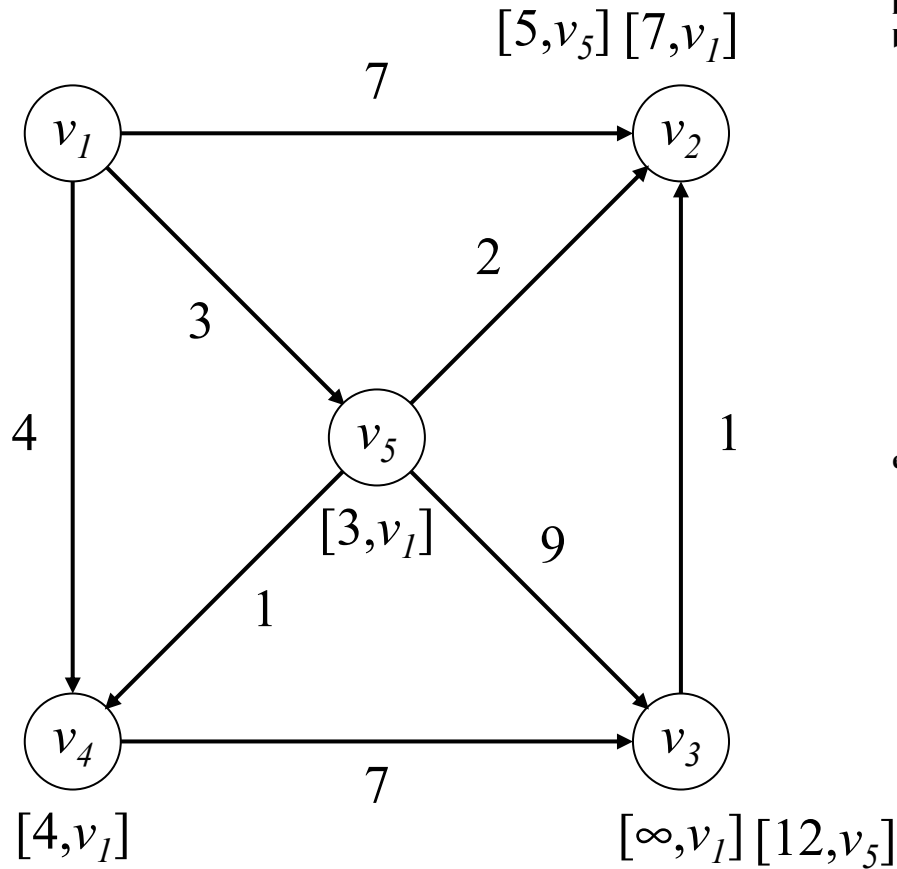
$L(v) := L(v^*) + c(v^*, v); pred(v) := v^*;$

end

end.

$$W = \{ v_1, v_5 \}$$

Aggiornamento etichette (1)



procedure DIJKSTRA

begin

$W := \{ s \}; L(s) := 0; pred(s) := 0;$

for each $v \in V \setminus \{ s \}$ **do**

$L(v) := c(s, v); pred(v) := s;$

while $|W| < n$ **do**

trova $v^* \in V \setminus W : L(v^*) = \min_{v \in V \setminus W} \{ L(v) \};$

$W := W \cup \{ v^* \};$

for each $v \in V \setminus W$ **do**

if $L(v^*) + c(v^*, v) < L(v)$ **then**

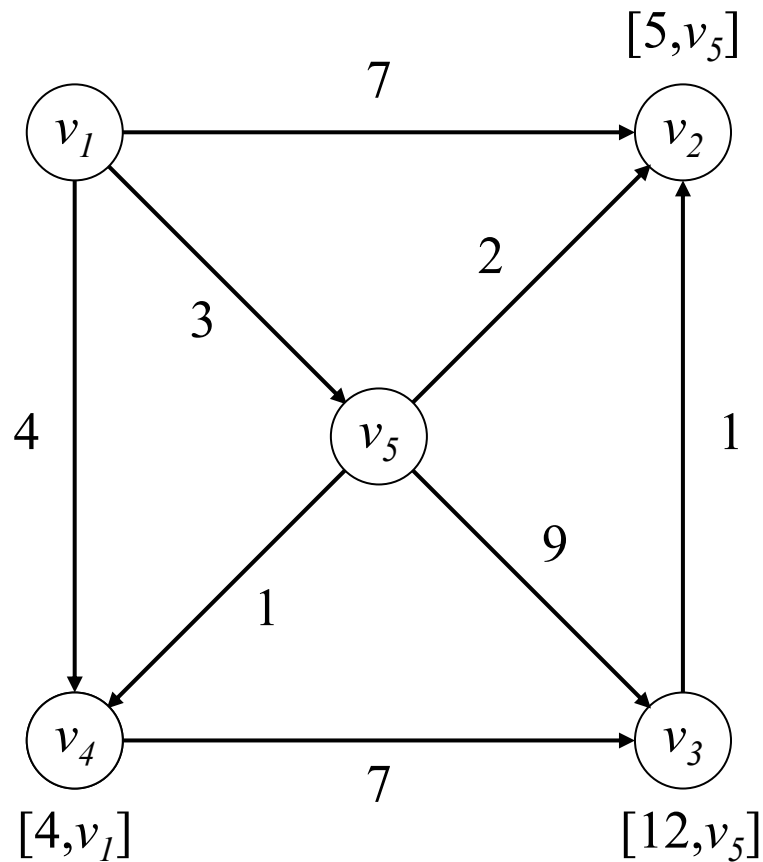
$L(v) := L(v^*) + c(v^*, v); pred(v) := v^*;$

end

end.

$$W = \{ v_1, v_5 \}$$

Selezione vertice (2)



procedure DIJKSTRA

begin

$W := \{ s \}; L(s) := 0; pred(s) := 0;$

for each $v \in V \setminus \{ s \}$ **do**

$L(v) := c(s, v); pred(v) := s;$

while $|W| < n$ **do**

trova $v^* \in V \setminus W : L(v^*) = \min_{v \in V \setminus W} \{ L(v) \};$

$W := W \cup \{ v^* \};$

for each $v \in V \setminus W$ **do**

if $L(v^*) + c(v^*, v) < L(v)$ **then**

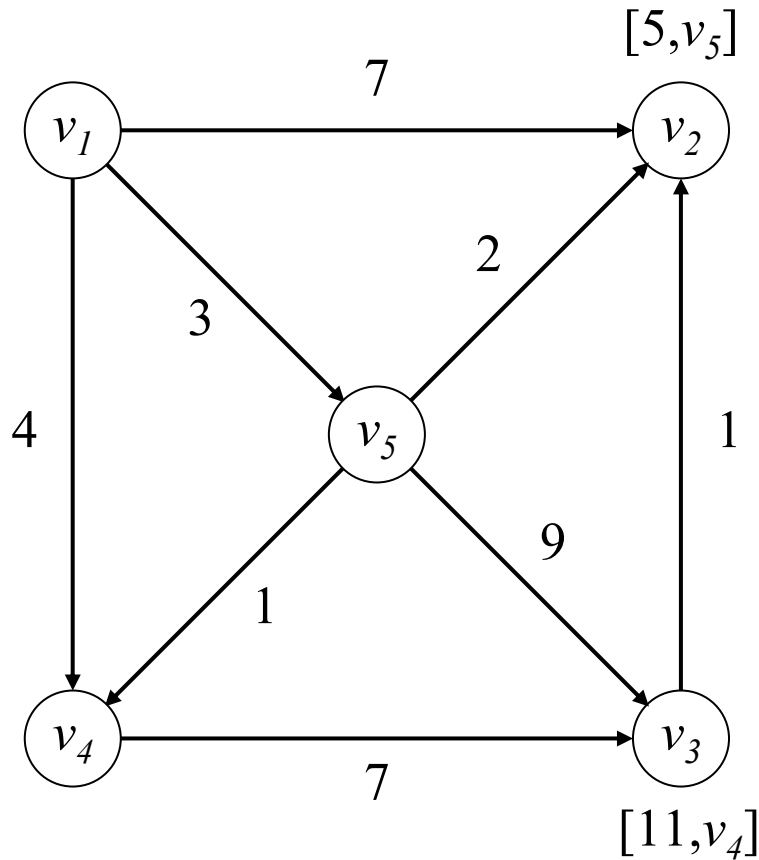
$L(v) := L(v^*) + c(v^*, v); pred(v) := v^*;$

end

end.

$$W = \{ v_1, v_5, v_4 \}$$

Aggiornamento etichette (2)



procedure DIJKSTRA

begin

$W := \{ s \}; L(s) := 0; pred(s) := 0;$

for each $v \in V \setminus \{ s \}$ **do**

$L(v) := c(s, v); pred(v) := s;$

while $|W| < n$ **do**

trova $v^* \in V \setminus W : L(v^*) = \min_{v \in V \setminus W} \{ L(v) \};$

$W := W \cup \{ v^* \};$

for each $v \in V \setminus W$ **do**

if $L(v^*) + c(v^*, v) < L(v)$ **then**

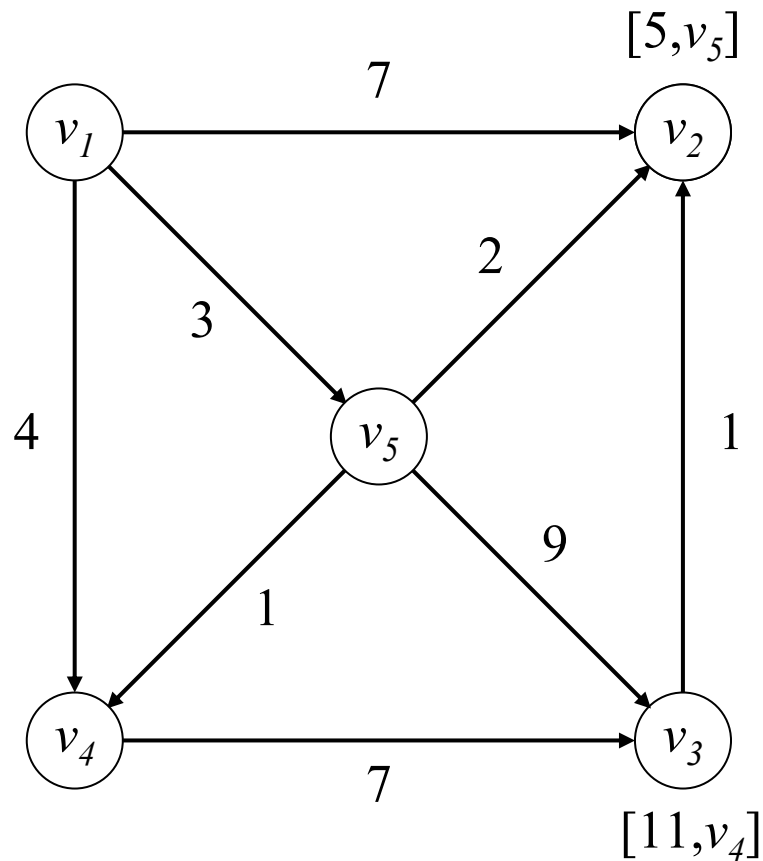
$L(v) := L(v^*) + c(v^*, v); pred(v) := v^*;$

end

end.

$$W = \{ v_1, v_5, v_4 \}$$

Selezione vertice (3)



procedure DIJKSTRA

begin

$W := \{ s \}; L(s) := 0; pred(s) := 0;$

for each $v \in V \setminus \{ s \}$ **do**

$L(v) := c(s, v); pred(v) := s;$

while $|W| < n$ **do**

trova $v^* \in V \setminus W : L(v^*) = \min_{v \in V \setminus W} \{ L(v) \};$

$W := W \cup \{ v^* \};$

for each $v \in V \setminus W$ **do**

if $L(v^*) + c(v^*, v) < L(v)$ **then**

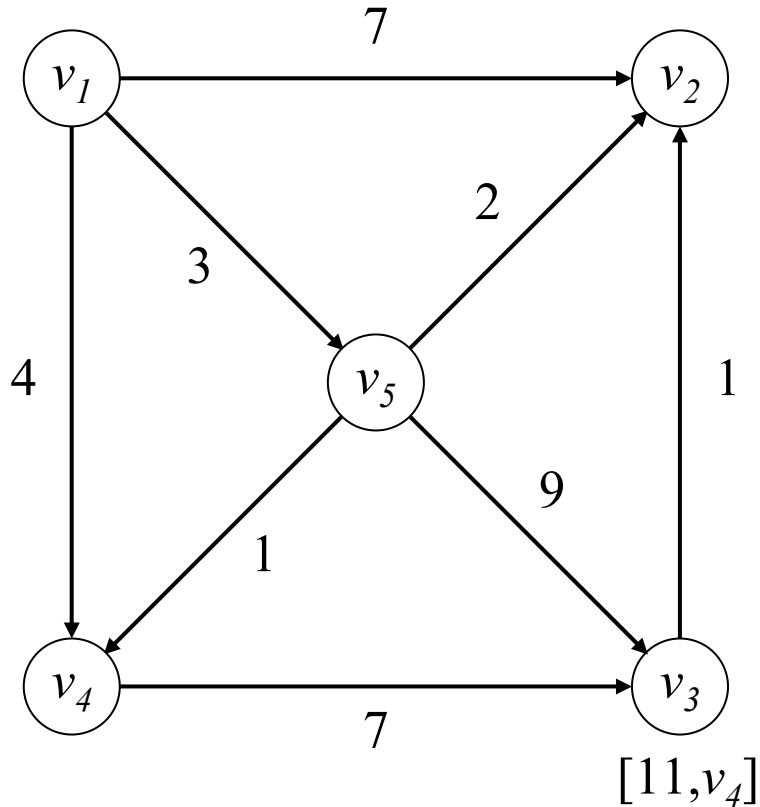
$L(v) := L(v^*) + c(v^*, v); pred(v) := v^*;$

end

end.

$$W = \{ v_1, v_5, v_4, v_2 \}$$

Aggiornamento etichette (3)



procedure DIJKSTRA

begin

$W := \{ s \}; L(s) := 0; pred(s) := 0;$

for each $v \in V \setminus \{ s \}$ **do**

$L(v) := c(s, v); pred(v) := s;$

while $|W| < n$ **do**

trova $v^* \in V \setminus W : L(v^*) = \min_{v \in V \setminus W} \{ L(v) \};$

$W := W \cup \{ v^* \};$

for each $v \in V \setminus W$ **do**

if $L(v^*) + c(v^*, v) < L(v)$ **then**

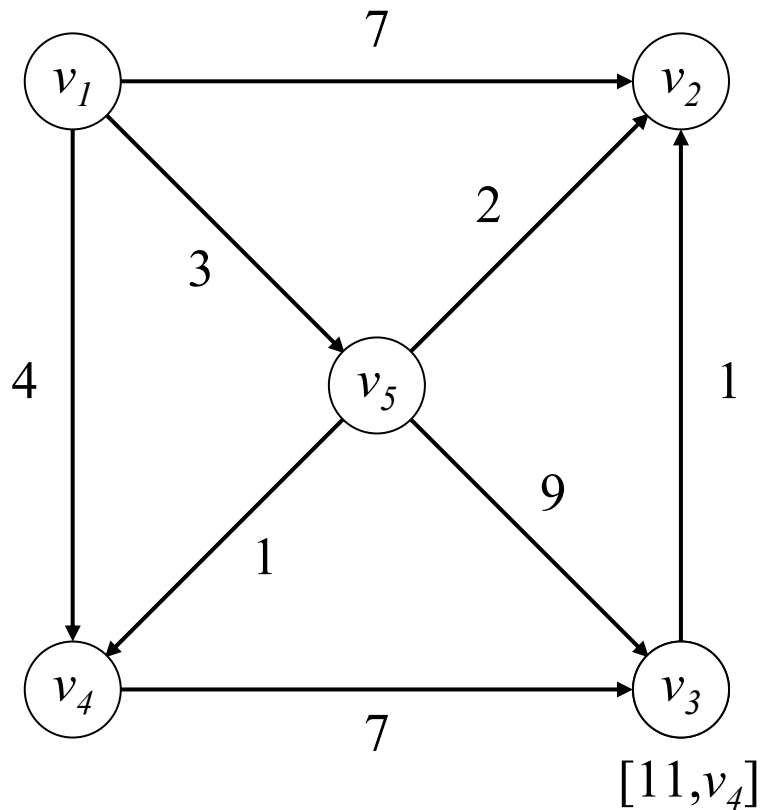
$L(v) := L(v^*) + c(v^*, v); pred(v) := v^*;$

end

end.

$$W = \{ v_1, v_5, v_4, v_2 \}$$

Selezione vertice (4)



procedure DIJKSTRA

begin

$W := \{ s \}; L(s) := 0; pred(s) := 0;$

for each $v \in V \setminus \{ s \}$ **do**

$L(v) := c(s, v); pred(v) := s;$

while $|W| < n$ **do**

trova $v^* \in V \setminus W : L(v^*) = \min_{v \in V \setminus W} \{ L(v) \};$

$W := W \cup \{ v^* \};$

for each $v \in V \setminus W$ **do**

if $L(v^*) + c(v^*, v) < L(v)$ **then**

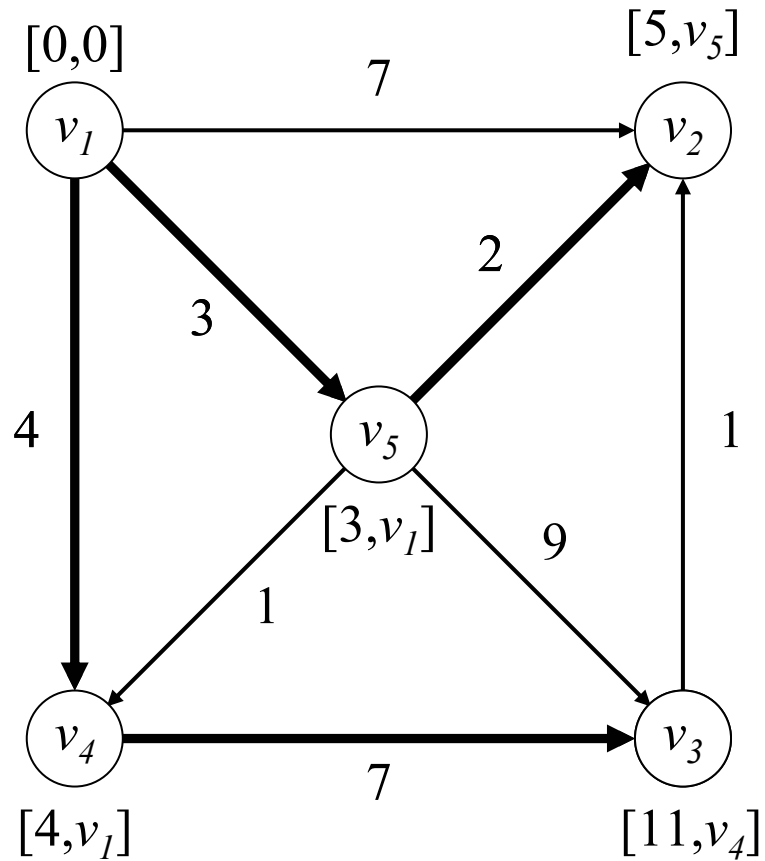
$L(v) := L(v^*) + c(v^*, v); pred(v) := v^*;$

end

end.

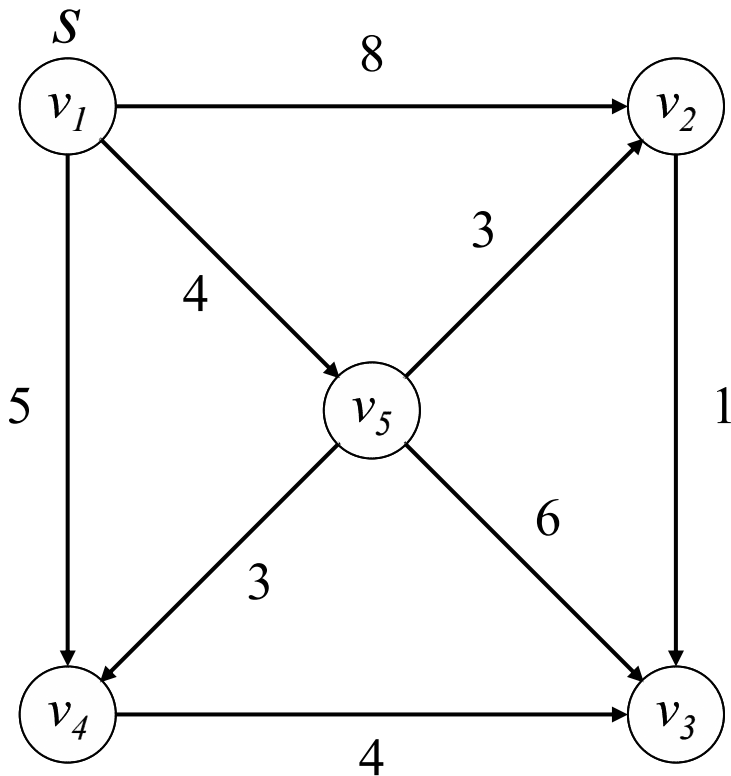
$$W = \{ v_1, v_5, v_4, v_2, v_1 \}$$

Soluzione



v	$L(v)$	$pred(v)$
v_1	0	0
v_2	5	v_5
v_3	11	v_4
v_4	4	v_1
v_5	3	v_1

Esempio



W	$L(v)$
$\{1\}$	$(0, 8, \infty, 5, 4)$
$\{1, 5\}$	$(0, 7, 10, 5, 4)$
$\{1, 5, 4\}$	$(0, 7, 9, 5, 4)$
$\{1, 5, 4, 2\}$	$(0, 7, 8, 5, 4)$
$\{1, 5, 4, 2, 3\}$	$(0, 7, 8, 5, 4)$