MINIMAL SPANNING TREES

Networks

A *network* $G = (V, E, w)$ (also called *weighted graph*) is a graph (V, E) with a weight function $w : E \to \mathbb{R}$, $e \mapsto w(e)$.

The weighted adjacency matrix of a network is

$$A_w(G) = (w(v_i, v_j))_{1 \leq i,j \leq |V|}.$$

We extend the weight function to subsets $F \subseteq E$ of the edge set (or spanning subgraphs of G) by setting

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

A spanning tree (forest) T is called minimal if its weight

$$w(T) := \sum_{e \in E(T)} w(e)$$

is minimal among all spanning trees (forests).

Networks

In a directed or undirected network $G = (V, E, w)$ for any $x, y \in V$ we set

$$M_{x,y} := \{w(P) \,:\, P = (V_P, E_P) \text{ is a path from } x \text{ to } y\},$$

where

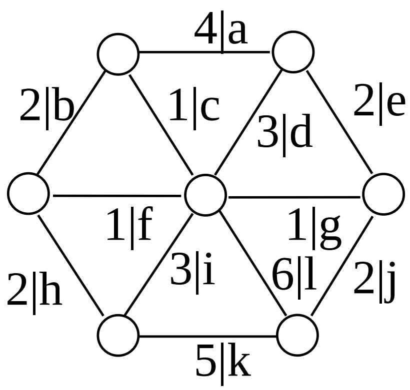
$$w(P) = \sum_{e \in E_P} w(e).$$

Then we define the *distance* from x to y by

$$d(x,y) = \begin{cases} \min M_{x,y} & \text{if } M_{x,y} \neq \emptyset, \\ \infty & \text{if } M_{x,y} = \emptyset. \end{cases}$$

KRUSKAL'S ALGORITHM

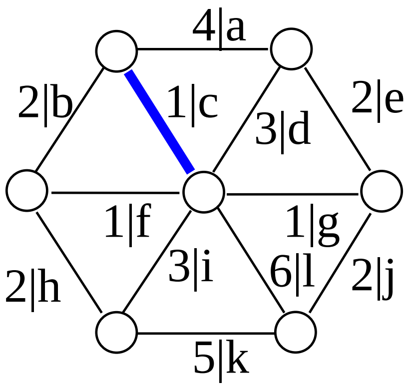
Kruskal's Algorithm



1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. WHILE $(|E'| < |V| - 1$ AND $j < m)$ DO
   IF $(V, E' \cup \{e_j\})$ acyclic THEN
   $E' := E' \cup \{e_j\}$
   END IF
   $j := j + 1$
   END DO

Kruskal's Algorithm



$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \emptyset$
$j = 1$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. `WHILE` $(|E'| < |V| - 1$ `AND` $j < m)$ `DO`

   `IF` $(V, E' \cup \{e_j\})$ acyclic `THEN`

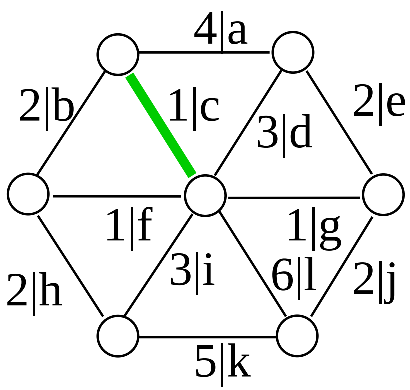
   $E' := E' \cup \{e_j\}$

   `END IF`

   $j := j + 1$

   `END DO`

Kruskal's Algorithm

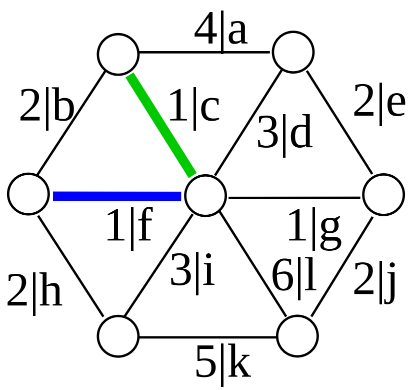


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c\}$
$j = 1$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. `WHILE` $(|E'| < |V| - 1$ `AND` $j < m)$ `DO`
   `IF` $(V, E' \cup \{e_j\})$ acyclic `THEN`
   $E' := E' \cup \{e_j\}$
   `END IF`
   $j := j + 1$
   `END DO`

Kruskal's Algorithm



$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c\}$
$j = 2$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. WHILE $(|E'| < |V| - 1$ AND $j < m)$ DO

   IF $(V, E' \cup \{e_j\})$ acyclic THEN

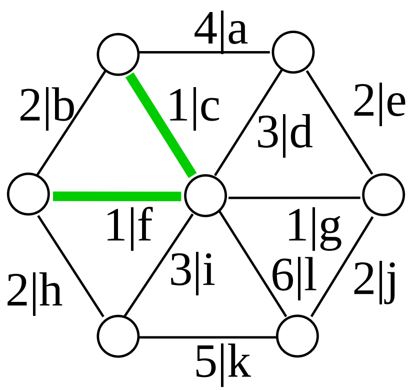
   $E' := E' \cup \{e_j\}$

   END IF

   $j := j + 1$

   END DO

Kruskal's Algorithm

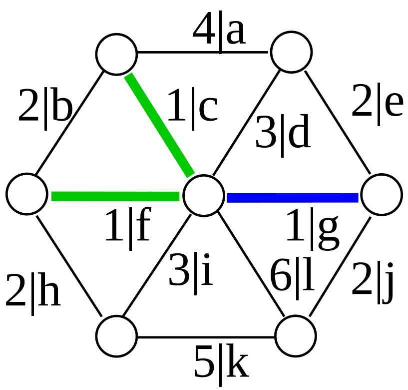


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f\}$
$j = 2$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. WHILE $(|E'| < |V| - 1$ AND $j < m)$ DO
    IF $(V, E' \cup \{e_j\})$ acyclic THEN
        $E' := E' \cup \{e_j\}$
    END IF
    $j := j + 1$
   END DO

Kruskal's Algorithm

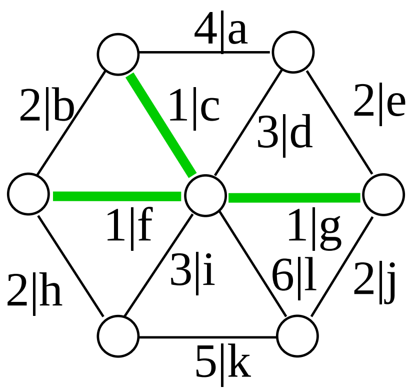


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f\}$
$j = 3$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. WHILE $(|E'| < |V| - 1$ AND $j < m)$ DO
   IF $(V, E' \cup \{e_j\})$ acyclic THEN
   $E' := E' \cup \{e_j\}$
   END IF
   $j := j + 1$
   END DO

Kruskal's Algorithm

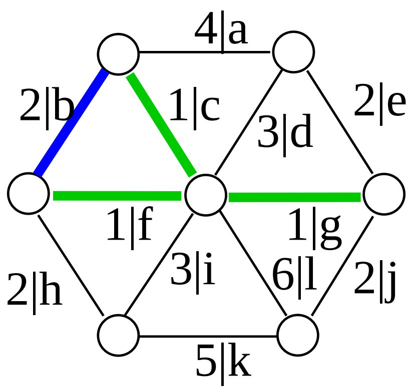


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f, g\}$
$j = 3$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. `WHILE` $(|E'| < |V| - 1$ `AND` $j < m)$ `DO`
   `IF` $(V, E' \cup \{e_j\})$ acyclic `THEN`
   $E' := E' \cup \{e_j\}$
   `END IF`
   $j := j + 1$
   `END DO`

Kruskal's Algorithm

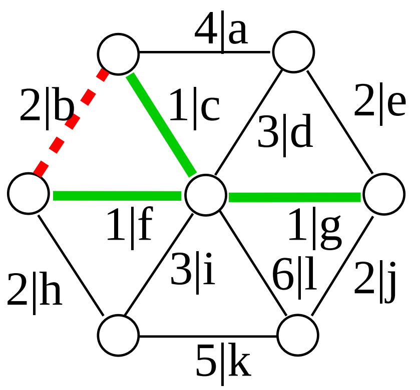


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f, g\}$
$j = 4$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. WHILE $(|E'| < |V| - 1$ AND $j < m)$ DO
   IF $(V, E' \cup \{e_j\})$ acyclic THEN
   $E' := E' \cup \{e_j\}$
   END IF
   $j := j + 1$
   END DO

Kruskal's Algorithm

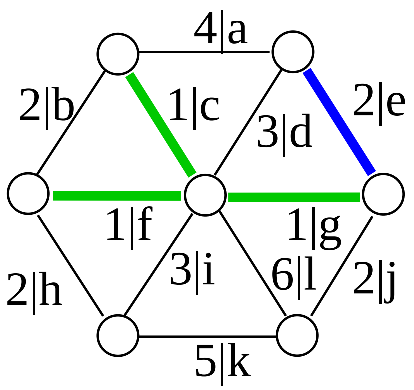


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f, g\}$
$j = 4$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. `WHILE` $(|E'| < |V| - 1$ `AND` $j < m)$ `DO`
   `IF` $(V, E' \cup \{e_j\})$ acyclic `THEN`
   $E' := E' \cup \{e_j\}$
   `END IF`
   $j := j + 1$
   `END DO`

Kruskal's Algorithm

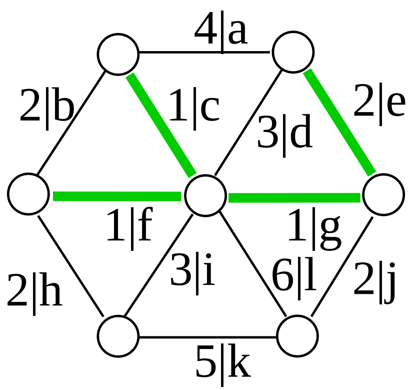


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f, g\}$
$j = 5$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. WHILE $(|E'| < |V| - 1$ AND $j < m)$ DO
   IF $(V, E' \cup \{e_j\})$ acyclic THEN
   $E' := E' \cup \{e_j\}$
   END IF
   $j := j + 1$
   END DO

Kruskal's Algorithm

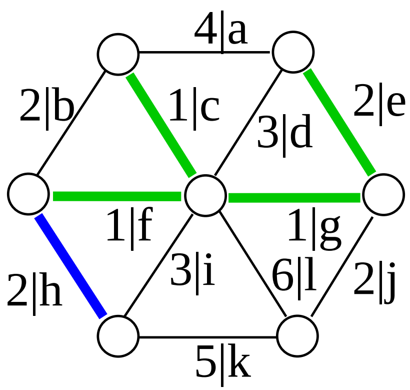


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f, g, e\}$
$j = 5$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. WHILE $(|E'| < |V| - 1$ AND $j < m)$ DO
   IF $(V, E' \cup \{e_j\})$ acyclic THEN
   $E' := E' \cup \{e_j\}$
   END IF
   $j := j + 1$
   END DO

Kruskal's Algorithm

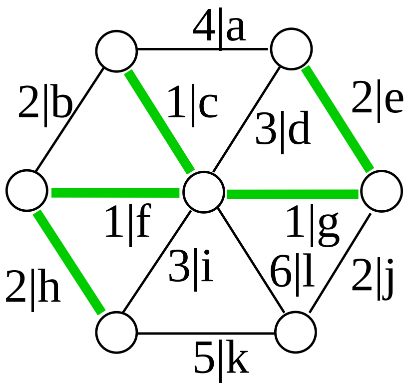


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f, g, e\}$
$j = 6$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. WHILE $(|E'| < |V| - 1$ AND $j < m)$ DO
   IF $(V, E' \cup \{e_j\})$ acyclic THEN
   $E' := E' \cup \{e_j\}$
   END IF
   $j := j + 1$
   END DO

Kruskal's Algorithm

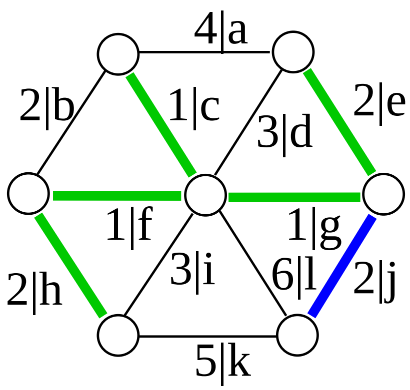


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f, g, e, h\}$
$j = 6$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. `WHILE` $(|E'| < |V| - 1$ `AND` $j < m)$ `DO`
   `IF` $(V, E' \cup \{e_j\})$ acyclic `THEN`
   $E' := E' \cup \{e_j\}$
   `END IF`
   $j := j + 1$
   `END DO`

Kruskal's Algorithm

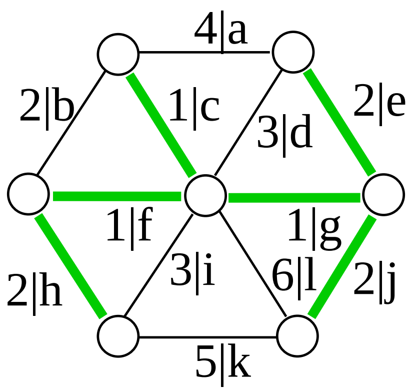


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f, g, e, h\}$
$j = 7$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. `WHILE` $(|E'| < |V| - 1$ `AND` $j < m)$ `DO`
    `IF` $(V, E' \cup \{e_j\})$ acyclic `THEN`
        $E' := E' \cup \{e_j\}$
    `END IF`
    $j := j + 1$
   `END DO`

Kruskal's Algorithm

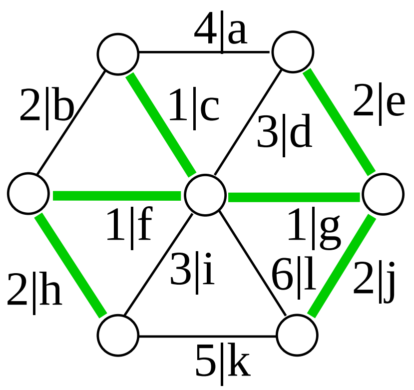


$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f, g, e, h, j\}$
$j = 7$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. WHILE $(|E'| < |V| - 1$ AND $j < m)$ DO
   IF $(V, E' \cup \{e_j\})$ acyclic THEN
   $E' := E' \cup \{e_j\}$
   END IF
   $j := j + 1$
   END DO

Kruskal's Algorithm



$E = \{c, f, g, b, e, h, j, d, i, a, k, l\}$
$E' = \{c, f, g, e, h, j\}$
$j = 7$
$|E'| = 6 \rightsquigarrow END$

1. Sort edges by weight; $E' := \emptyset$; $j := 1$;

2. `WHILE` $(|E'| < |V| - 1$ `AND` $j < m)$ `DO`

   `IF` $(V, E' \cup \{e_j\})$ acyclic `THEN`

   $E' := E' \cup \{e_j\}$

   `END IF`

   $j := j + 1$

   `END DO`

PRIM'S ALGORITHM

Prim's Algorithm

Store graph as adjacency lists:

$$V = \{v_1, \dots, v_n\}, \qquad A_i = \Gamma(v_i),\ i = 1, \dots, n$$

1. $g(v_1) := 1$; $S := \emptyset$; $T := \emptyset$;

2. `FOR` i = 2 to n `DO`: $g(v_i) := \infty$: `END`;

3. `WHILE` $S \neq V$ `DO`:

    choose $v_i \in V \setminus S$ such that $g(v_i)$ minimal;

    $S := S \cup \{v_i\}$;

    `IF` $i \neq 1$ `THEN` $T := T \cup \{e_i\}$: `END`;

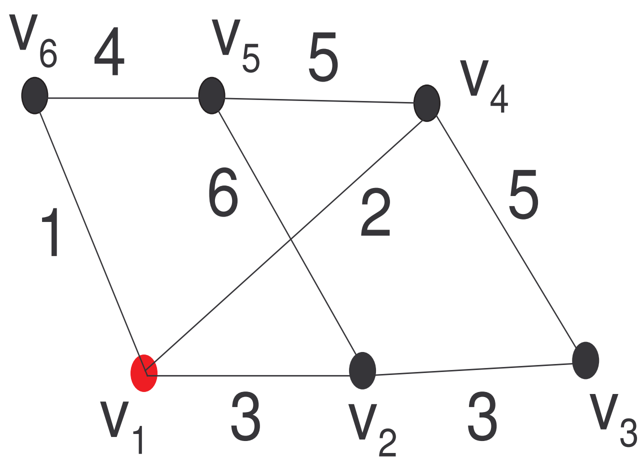
    `FOR` $v_j \in A_i \cap (V \setminus S)$ `DO`:

        `IF` $g(v_j) > w(v_i v_j)$ `THEN` $g(v_j) := w(v_i v_j)$: $e_j := v_i v_j$: `END`

    `END`;

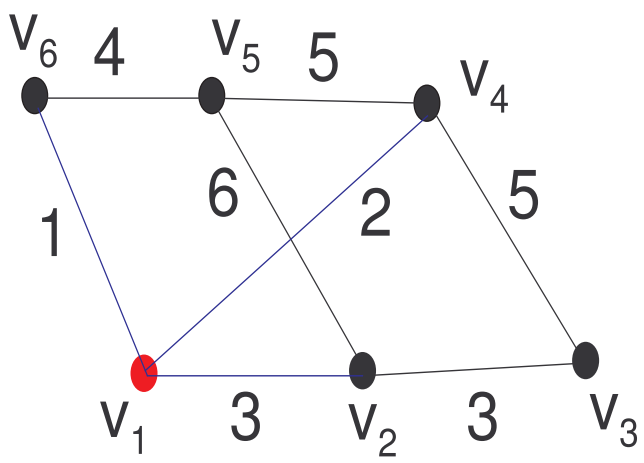
   `END`;

Prim's Algorithm



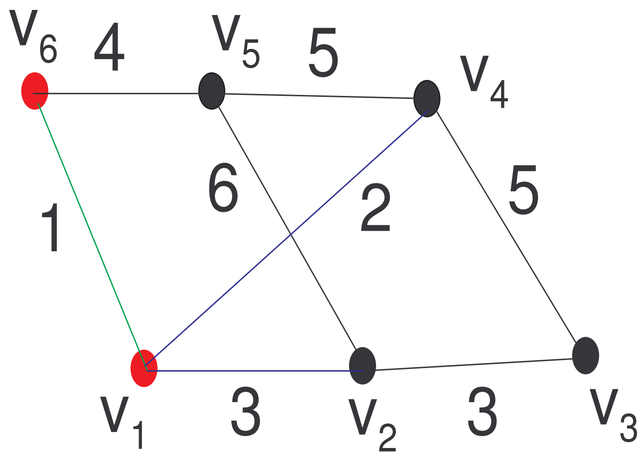
	$g(v_2)$	$g(v_3)$	$g(v_4)$	$g(v_5)$	$g(v_6)$	
v_1						

Prim's Algorithm



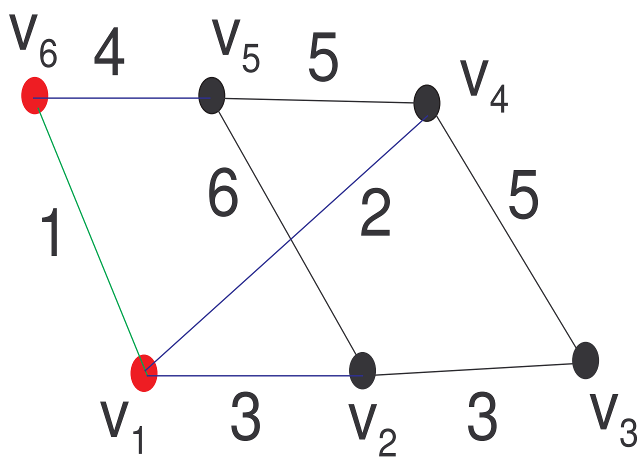
	$g(v_2)$	$g(v_3)$	$g(v_4)$	$g(v_5)$	$g(v_6)$	
v_1	3	∞	2	∞	$\boxed{1}$	$e_2 := 3,\ e_4 := 2,\ e_6 := 1$

Prim's Algorithm



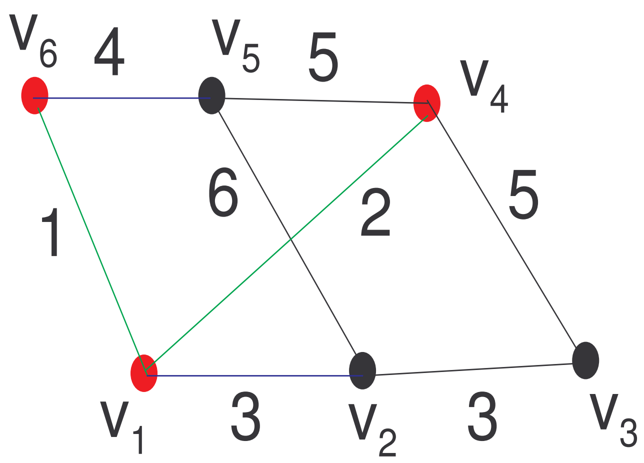
	$g(v_2)$	$g(v_3)$	$g(v_4)$	$g(v_5)$	$g(v_6)$	
v_1	3	∞	2	∞	1	$e_2 := 3,\ e_4 := 2,\ e_6 := 1$
v_6					–	

Prim's Algorithm



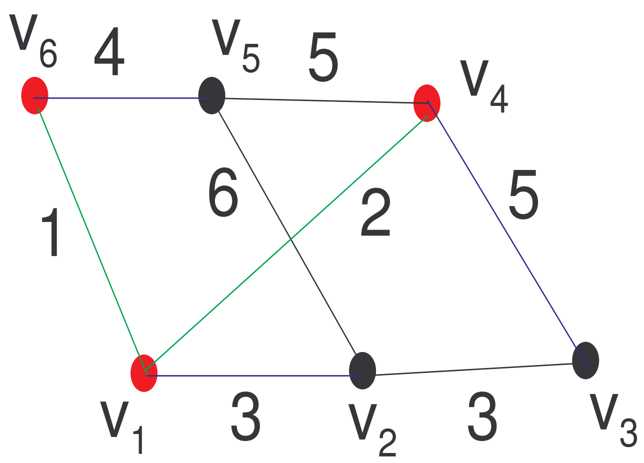
	$g(v_2)$	$g(v_3)$	$g(v_4)$	$g(v_5)$	$g(v_6)$	
v_1	3	∞	2	∞	1	$e_2 := 3,\ e_4 := 2,\ e_6 := 1$
v_6	3	∞	$\boxed{2}$	4	$-$	$e_5 := 4$

Prim's Algorithm



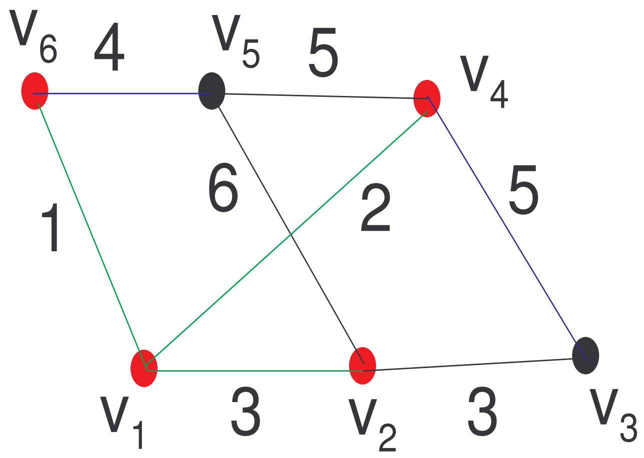
	$g(v_2)$	$g(v_3)$	$g(v_4)$	$g(v_5)$	$g(v_6)$	
v_1	3	∞	2	∞	1	$e_2 := 3,\ e_4 := 2,\ e_6 := 1$
v_6	3	∞	2	4	$-$	$e_5 := 4$
v_4			$-$		$-$	

Prim's Algorithm



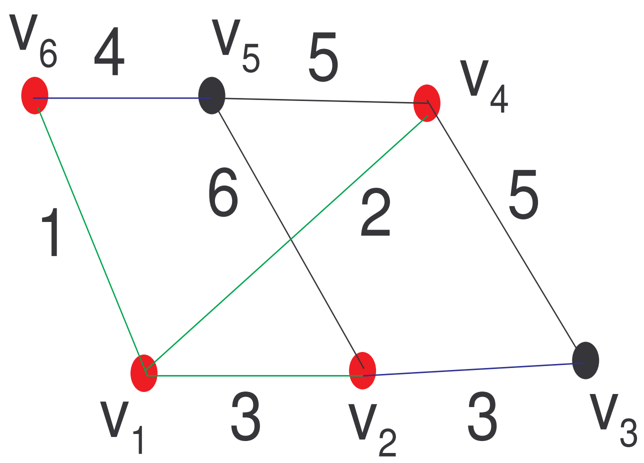
	$g(v_2)$	$g(v_3)$	$g(v_4)$	$g(v_5)$	$g(v_6)$	
v_1	3	∞	2	∞	1	$e_2 := 3,\ e_4 := 2,\ e_6 := 1$
v_6	3	∞	2	4	–	$e_5 := 4$
v_4	$\boxed{3}$	5	–	4	–	$e_3 := 5$

Prim's Algorithm



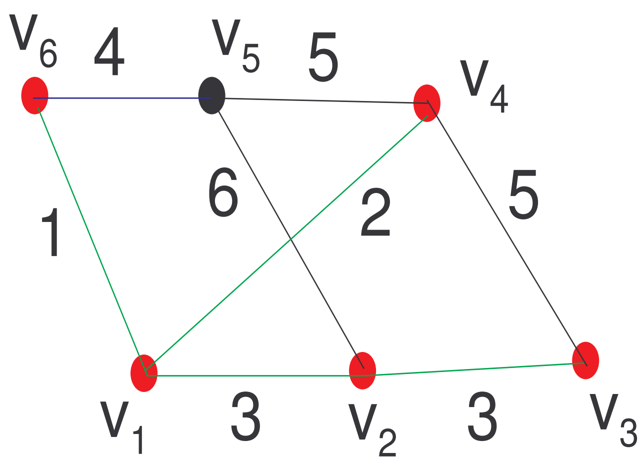
	$g(v_2)$	$g(v_3)$	$g(v_4)$	$g(v_5)$	$g(v_6)$	
v_1	3	∞	2	∞	1	$e_2 := 3,\ e_4 := 2,\ e_6 := 1$
v_6	3	∞	2	4	–	$e_5 := 4$
v_4	3	5	–	4	–	$e_3 := 5$
v_2	–		–		–	

Prim's Algorithm



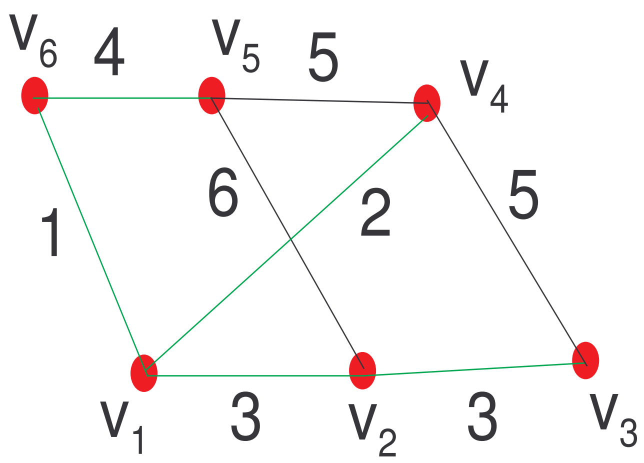
	$g(v_2)$	$g(v_3)$	$g(v_4)$	$g(v_5)$	$g(v_6)$	
v_1	3	∞	2	∞	1	$e_2 := 3,\ e_4 := 2,\ e_6 := 1$
v_6	3	∞	2	4	–	$e_5 := 4$
v_4	3	5	–	4	–	–
v_2	–	$\boxed{3}$	–	4	–	$e_3 := 3$

Prim's Algorithm



	$g(v_2)$	$g(v_3)$	$g(v_4)$	$g(v_5)$	$g(v_6)$	
v_1	3	∞	2	∞	1	$e_2 := 3,\ e_4 := 2,\ e_6 := 1$
v_6	3	∞	2	4	–	$e_5 := 4$
v_4	3	5	–	4	–	–
v_2	–	3	–	4	–	$e_3 := 3$
v_3	–	–	–	$\boxed{4}$	–	

Prim's Algorithm



	$g(v_2)$	$g(v_3)$	$g(v_4)$	$g(v_5)$	$g(v_6)$	
v_1	3	∞	2	∞	1	$e_2 := 3,\ e_4 := 2,\ e_6 := 1$
v_6	3	∞	2	4	–	$e_5 := 4$
v_4	3	5	–	4	–	–
v_2	–	3	–	4	–	$e_3 := 3$
v_3	–	–	–	$\boxed{4}$	–	