SCSI 1013: Discrete Structure

CHAPTER 4 – PART 1

GRAPH THEORY

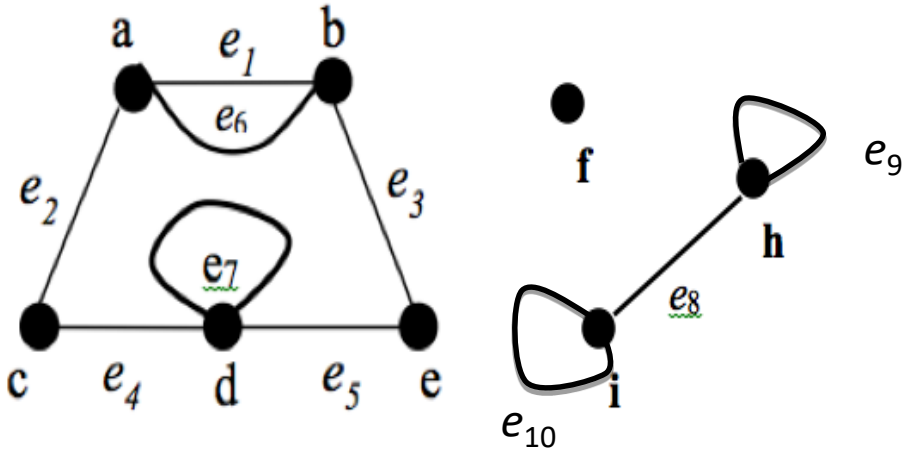
Definition of Graph

Definition

- A graph G consists of two finite sets:
  - a nonempty set $V(G)$ of vertices.
  - a set $E(G)$ of edges, where each edge is associated with a set consisting of either one or two vertices called its endpoints.
  - f is a function, called an incidence function, that assign to each edge, $e \in E$, a one element subset $\{v\}$ or two elements subset $\{v, w\}$, where v and w are vertices.
- We can write G as (V, E, f) or (V, E) or simply as G.

Example 1

Given a graph as shown below,



a) Write a vertex set and the edge set, and give a table showing the edge-endpoint function.

a) Find all edges that are incident on **a**, all vertices that are adjacent to **a**, all edges that are adjacent to e_2, all loops, all parallel edges, all vertices that are adjacent to themselves and all isolated vertices.

Note: Solution – Refer module page 91

Example 2

- Let,
  - $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$
  - $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$
- And *f* be defined by:
  - $f(e_1) = f(e_2) = \{v_1, v_2\}$
  - $f(e_3) = \{v_4, v_3\}$
  - $f(e_4) = f(e_6) = f(e_6) = \{v_6, v_3\}$
  - $f(e_5) = \{v_2, v_4\}$

Question: What is the pictorial representation of *G*?

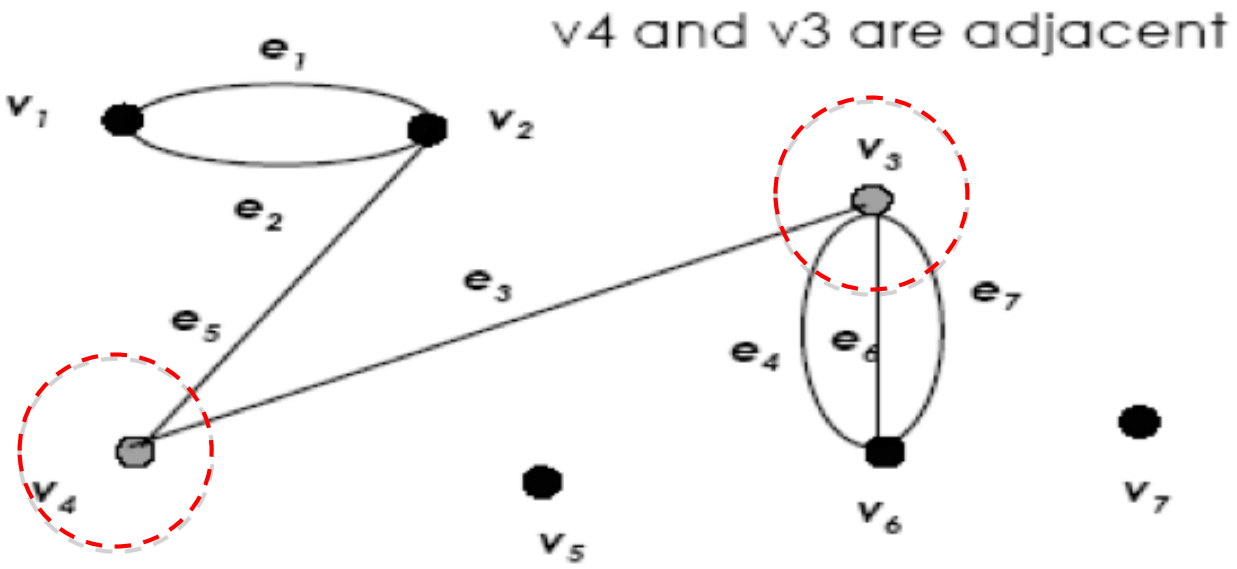
* Solution – refer module (Fig. 4.5)

Characteristics of Graph

(a)Adjacent Vertices

Two vertices that are connected by an edge are called adjacent; and a vertex that is an endpoint of a loop is said to be adjacent to itself.

Example

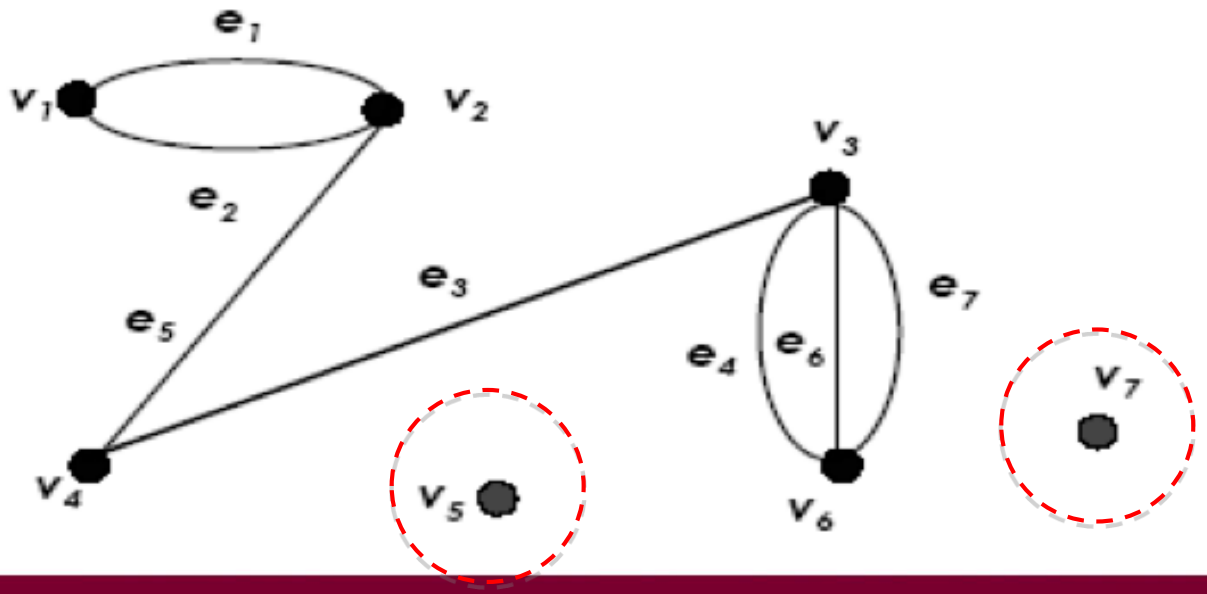


b) Isolated Vertex

Let G be a graph and *v* be a vertex in G. We say that *v* is an isolated vertex if it is not incident with any edge.

- v5 and v7 are isolated vertices.

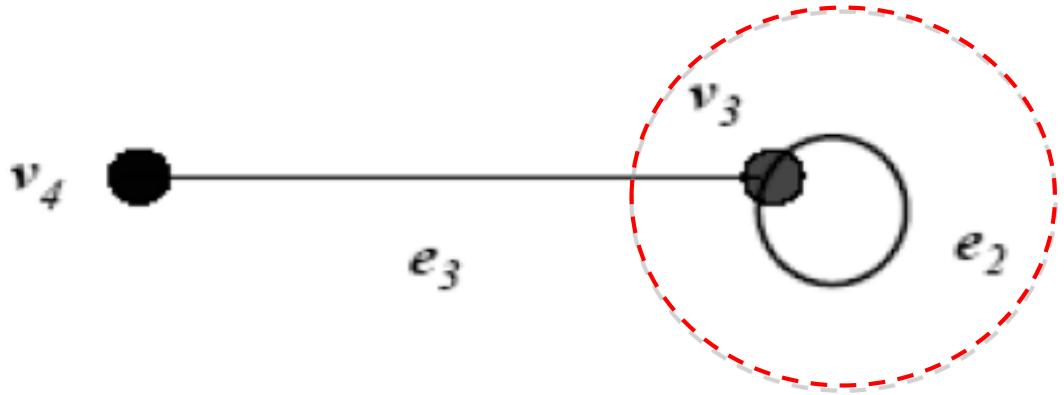
Example



c) Loop

An edge with just one endpoint is called a loop.

Example



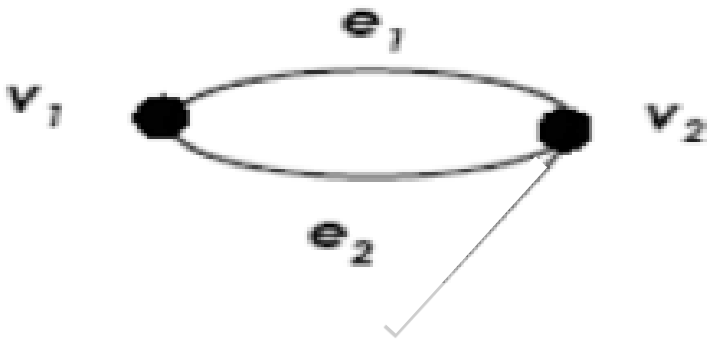
- e2 is a loop

d) Parallel Edges

Two or more distinct edges with the same set of endpoints are said to be parallel.

- e_1 and e_2 are parallel.

Example

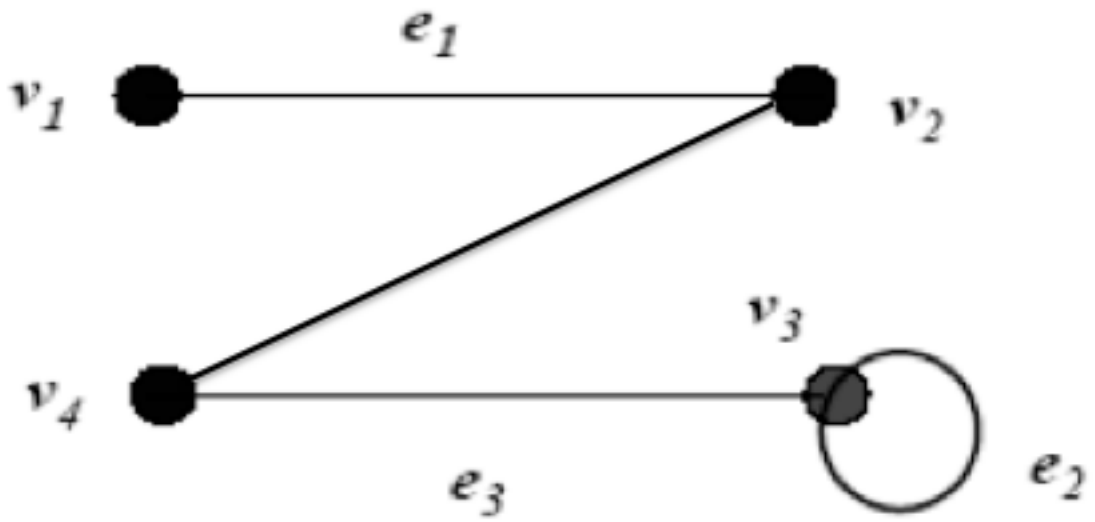


The Concept of Degree

- Let G be a graph and v be a vertex in G.
- The degree of v, written $\deg(v)$ or $d(v)$ is the number of edges incident with v.
- Each **loop** on a vertex v contributes 2 to the degree of v.

Example

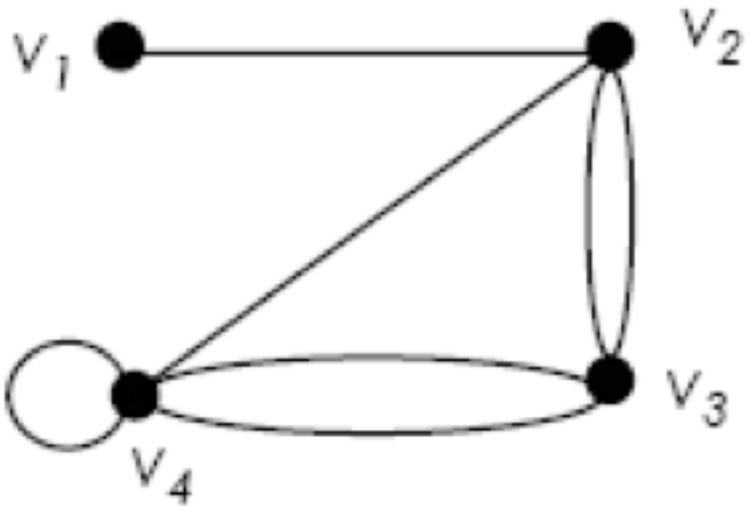
State the degree of each vertex for the following graph.



$\deg(v_1) = 1$; $\deg(v_2) = 2$; $\deg(v_3) = 3$; $\deg(v_4) = 2$

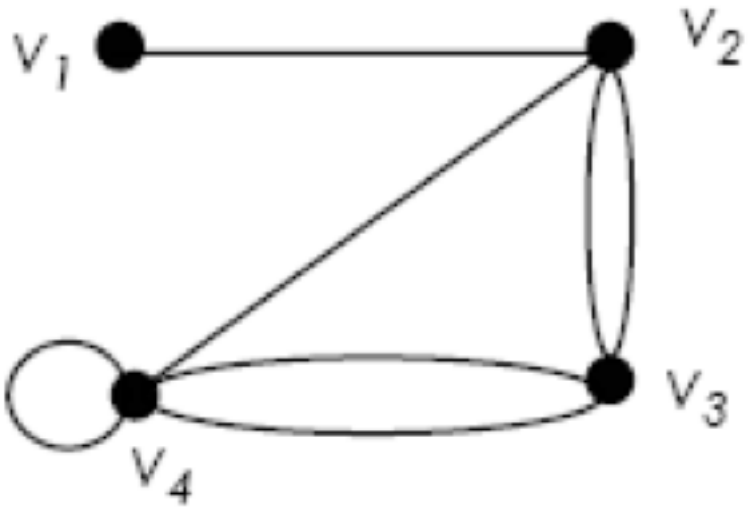
Exercise

- Find the degree of each vertex in the graph.



Exercise - Solution

- Find the degree of each vertex in the graph.



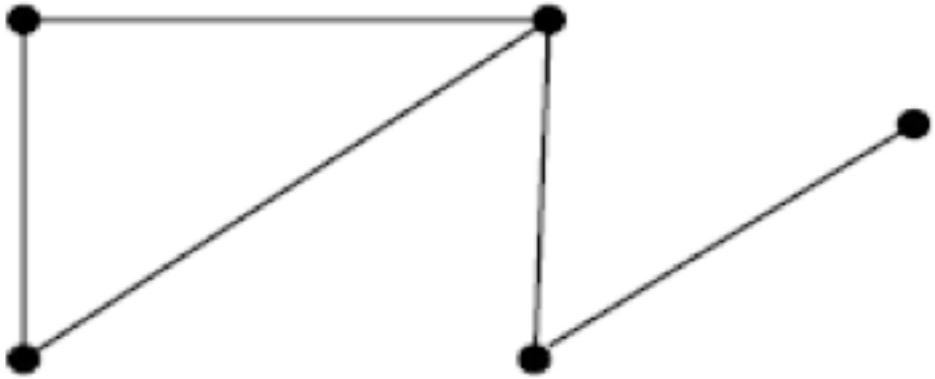
Solution: deg(v_1)= 1; deg(v_2)= 4; deg(v_3)= 4; deg(v_4)= 5

Types of Graphs

a) Simple Graph

A graph G is called a simple graph if G does not contain any parallel edges and any loops.

Example



b) Regular Graph

Let G be a graph and k be a nonnegative integer. G is called a k-regular graph if the degree of each vertex of G is k.

Example

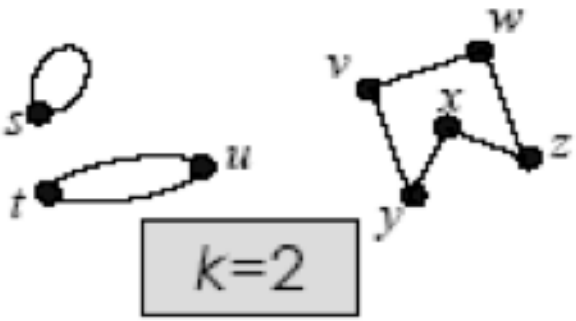


Fig.1: Graph A

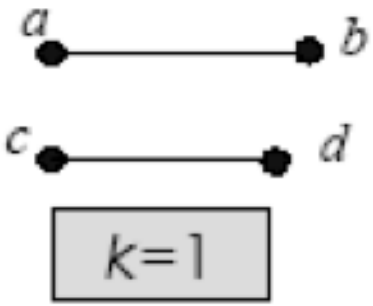


Fig.2: Graph B

c) Connected Graph

A graph G is connected if given any vertices v and w in G, there is a path from v to w.

Example

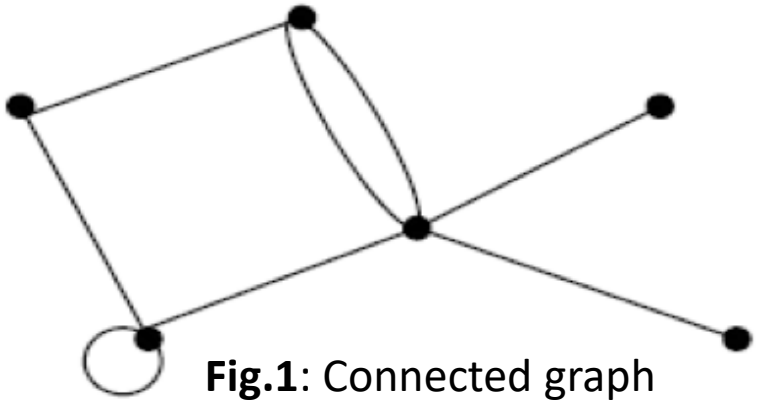


Fig.1: Connected graph

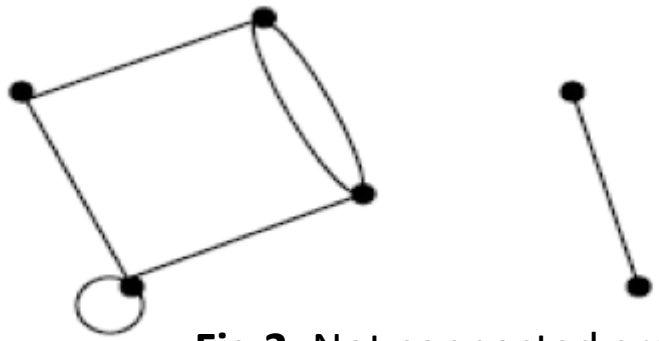
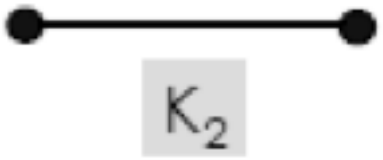


Fig.2: Not connected graph

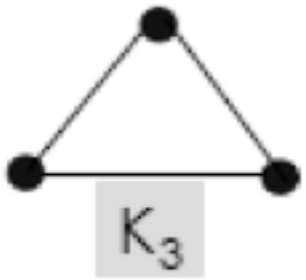
d) Complete Graph

A simple graph with n vertices in which there is an edge between every pair of distinct vertices is called a complete graph on n vertices. This is denoted by k_n .

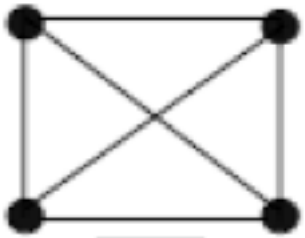
Example



K_2



K_3

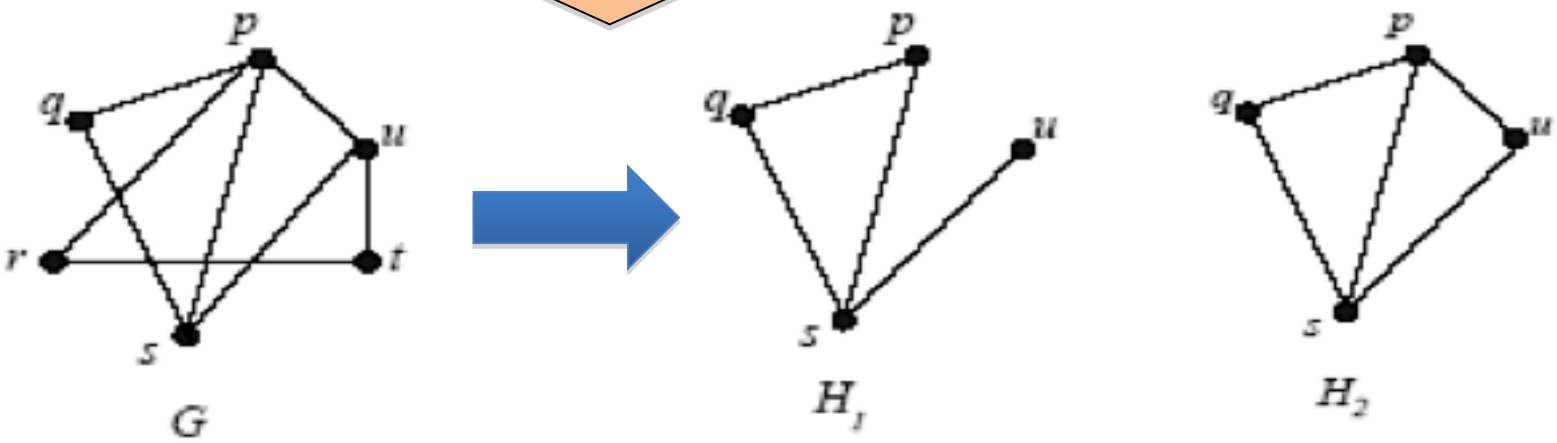


K_4

e) Subgraph

A graph H is said to be a subgraph of a graph G if, every vertex in H is also a vertex in G, every edge in H is also an edge in G, and every edge in H has the same endpoints as it has in G.

Example



Graph Representation

- To write programs that process and manipulate graphs, the graphs must be stored, that is, represented in computer memory.
- A graph can be represented (in computer memory) in several ways.
- 2-dimensional array: adjacency matrix and incidence matrix.

Adjacency Matrix

- Let G be a graph with n vertices.

- The adjacency matrix, A_G is an $n \times n$ matrix $[a_{ij}]$ such that,

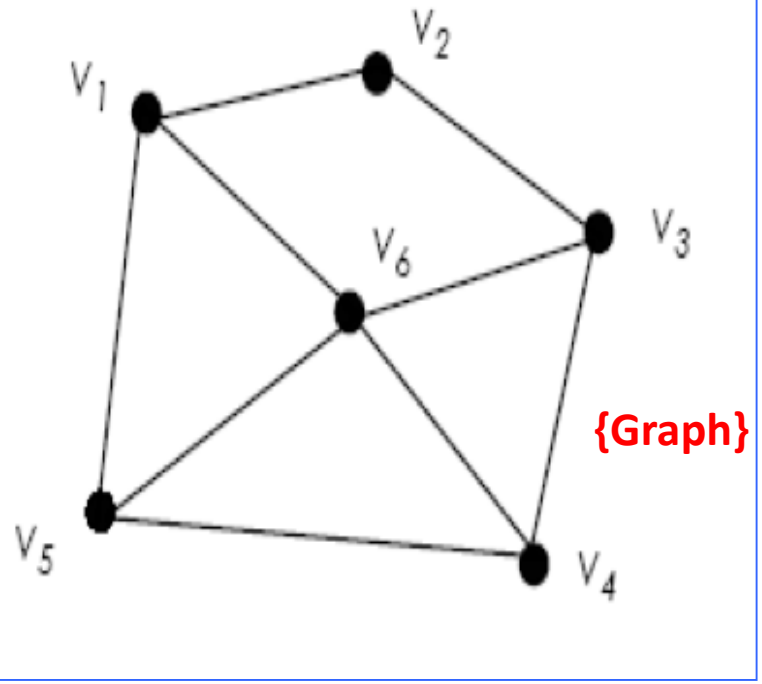
  a_{ij}= the number of edges from v_i to v_j, {undirected G}

  or,

  a_{ij}= the number of arrows from v_i to v_j, {directed G}

  for all i, j = 1,2,….., n.

Example 1

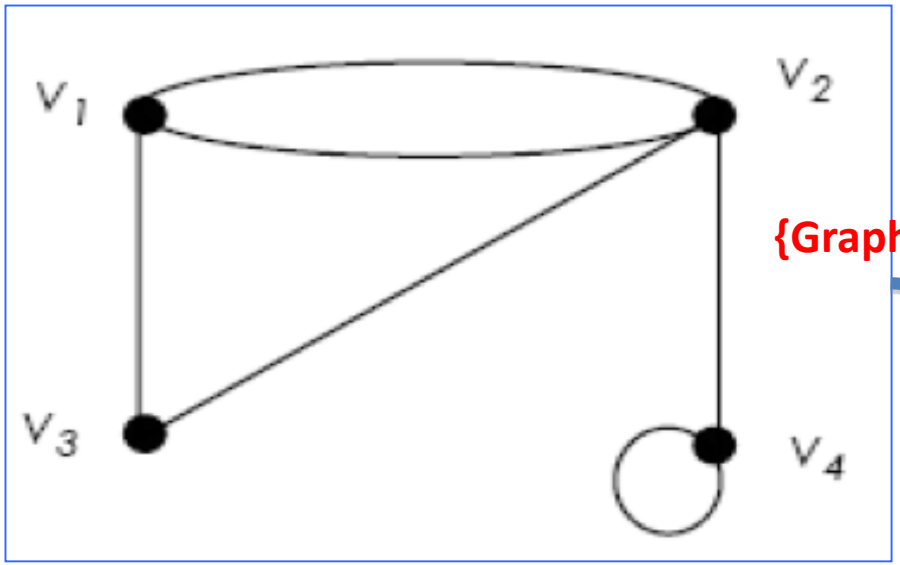


{Graph}

[Matrix]

$$A_G = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \end{array} \begin{array}{c} \begin{array}{cccccc} V_1 & V_2 & V_3 & V_4 & V_5 & V_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Example 2



{Graph}

[Matrix]

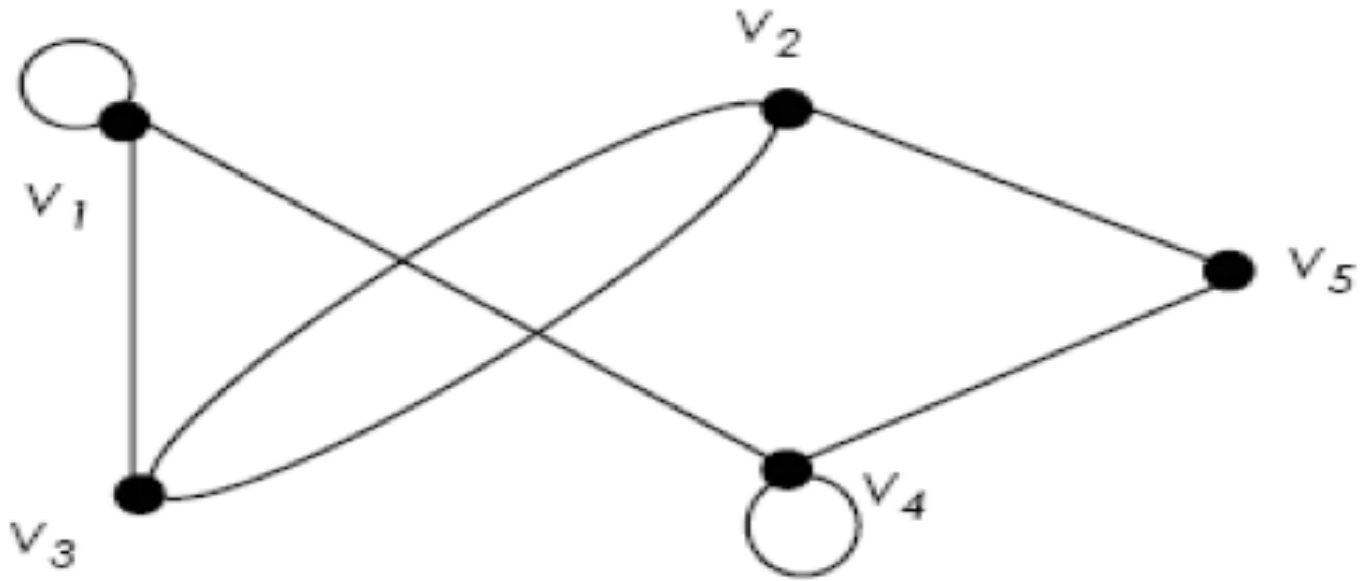
$$A_G = \begin{array}{c} \begin{matrix} V_1 & V_2 & V_3 & V_4 \end{matrix} \\ \begin{bmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Example 3

Draw the graph based on the following matrix:

$$A_G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 2 & 0 & 1 \\ 1 & 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Example 3 - Solution



- Adjacency matrix is a **symmetric matrix** if it is representing an undirected graph, where

$$a_{ij} = a_{ji}$$

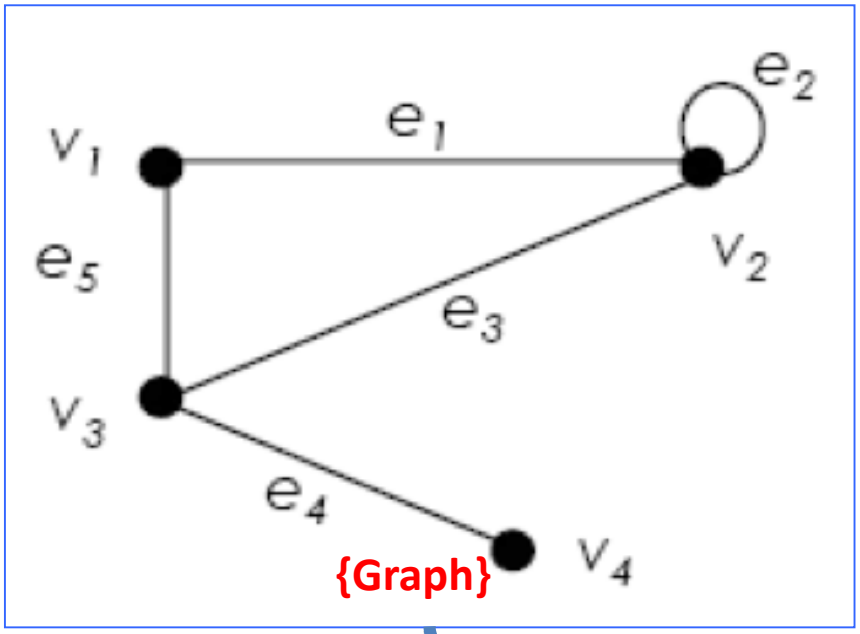
- If the graph is directed graph, the presented matrix is not symmetrical.

Incidence Matrix

- Let G be a graph with *n* vertices and *m* edges.
- The incidence matrix, I_G is an $n \times m$ matrix $[a_{ij}]$ such that,

$$a_{ij} = \begin{cases} 0 & \text{if } v_i \text{ is not an end vertex of } e_j, \\ 1 & \text{if } v_i \text{ is an end vertex of } e_j, \text{but } e_j \text{ is not a loop} \\ 2 & \text{if } e_j \text{ is a loop at } v_i \end{cases}$$

Example



[Matrix]

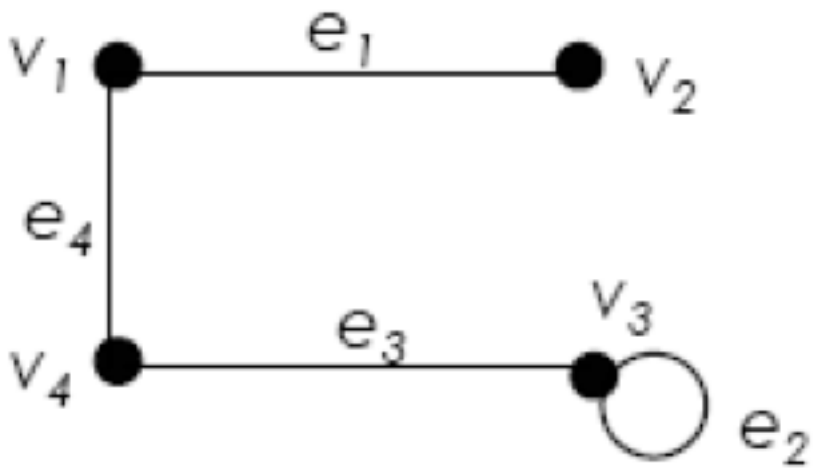
$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

$\deg(v_1) = 2;$
$\deg(v_2) = 4;$
$\deg(v_3) = 3;$
$\deg(v_4) = 1$

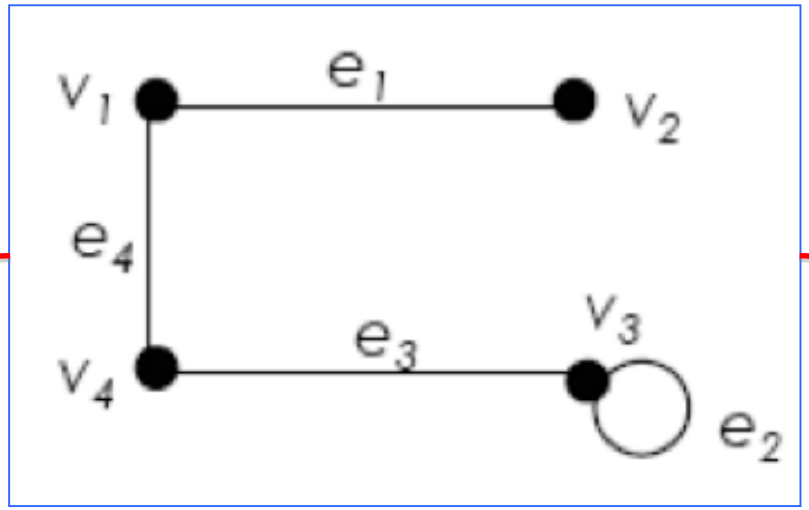
Notice that the sum of the i-th row is the degree of v_i.

Exercise

- Find the adjacency matrix and the incidence matrix of the graph.



Exercise - Solution



Adjacency matrix

Incidence matrix

$$A_G = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

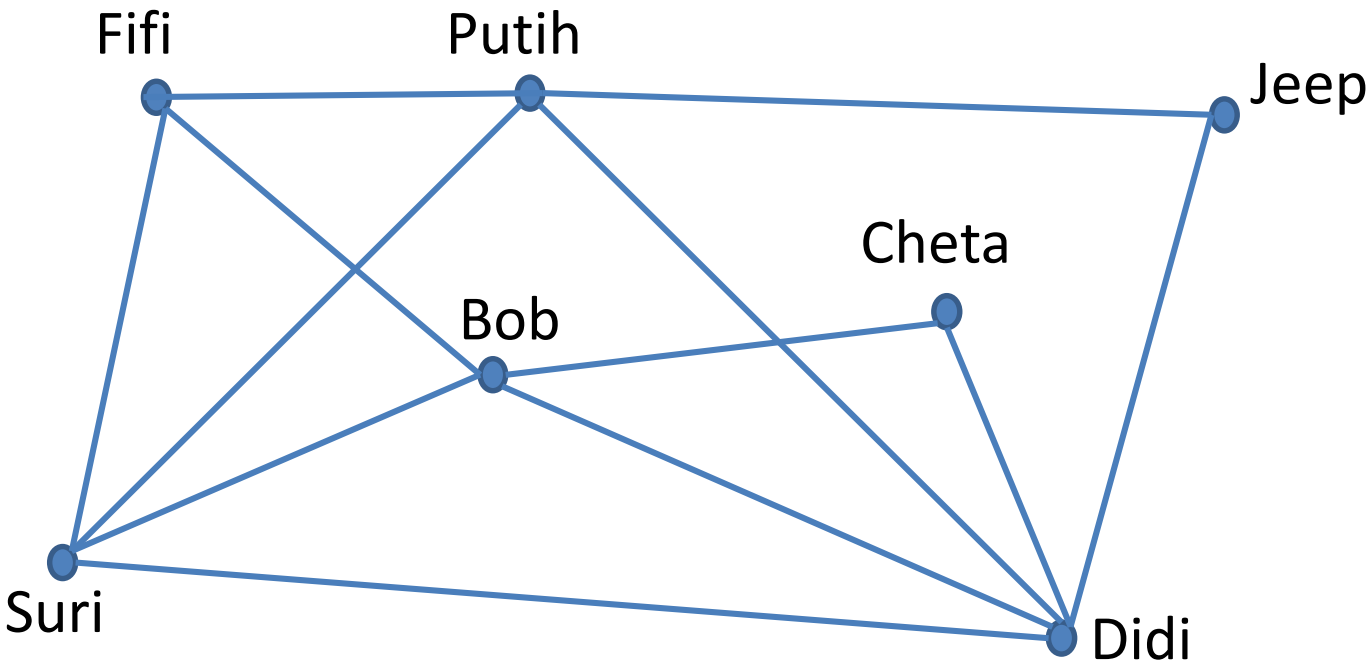
$$I_G = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} e_1 & e_2 & e_3 & e_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

Exercise
Past Year 2015/2016

A cat show is being judged from pictures of the cats. The judges would like to see pictures of the following pairs of cats next to each other for their final decision: Fifi and Putih, Fifi and Suri, Fifi and Bob, Bob and Cheta, Bob and Didi, Bob and Suri, Cheta and Didi, Didi and Suri, Didi and Putih, Suri and Putih, Putih and Jeep, Jeep and Didi.

Draw a graph modeling this situation. (3 marks)

Exercise Solution
Past Year 2015/2016



Exercise
Past Year 2015/2016

Given a graph as shown in Figure 1.

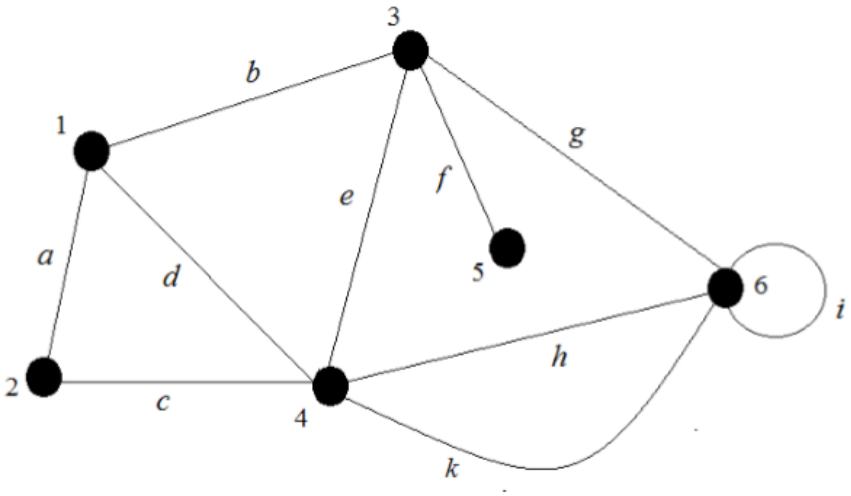


Figure 1

i. Find the incidence matrix of the graph. (4 marks)

ii. Find the adjacency matrix of the graph. (3 marks)

Exercise Solution
Past Year 2015/2016

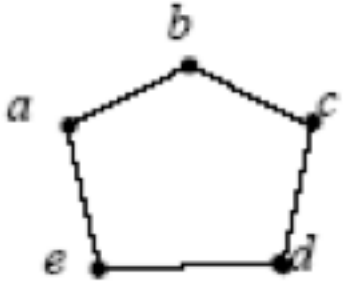
(i) Incident Matrix, I_G =

	a	b	c	d	e	f	g	h	i	k
1	1	1	0	1	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0
3	0	1	0	0	1	1	1	0	0	0
4	0	0	1	1	1	0	0	1	0	1
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	1	2	1

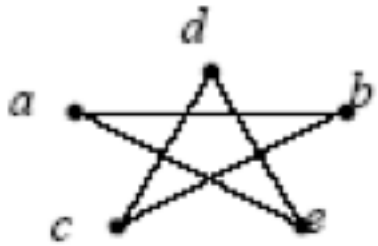
(ii) Adjacency Matrix , A_G =

	1	2	3	4	5	6
1	0	1	1	1	0	0
2	1	0	0	1	0	0
3	1	0	0	1	1	1
4	1	1	1	0	0	2
5	0	0	1	0	0	0
6	0	0	1	2	0	1

Isomorphisms



G_1



G_2

- Are these two graphs (G_1 and G_2) are same?
- When we say that 2 graphs are the same mean they are isomorphic to each other.

Definition

Let $G = \{V, E\}$ and $G' = \{V', E'\}$ be graphs. G and G' are said to be isomorphic if there exist a pair of functions $f : V \rightarrow V'$ and $g : E \rightarrow E'$ such that f associates each element in V with exactly one element in V' and vice versa; g associates each element in E with exactly one element in E' and vice versa, and for each $v \in V$, and each $e \in E$, if v is an endpoint of the edge e, then $f(v)$ is an endpoint of the edge $g(e)$.

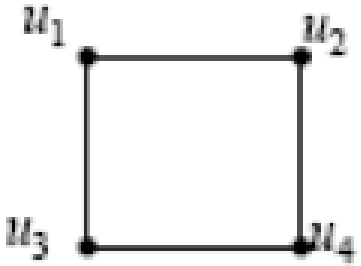
◆ If two graphs is isomorphic, they must have:

- the same number of vertices and edges,
- the same degrees for corresponding vertices,
- the same number of connected components,
- the same number of loops and parallel edges,
- both graphs are connected or both graph are not connected,
- pairs of connected vertices must have the corresponding pair of vertices connected.

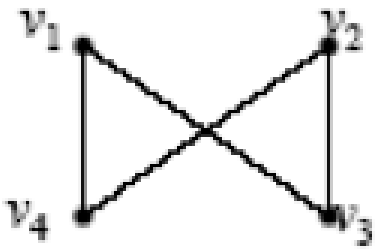
◆ In general, it is easier to prove two graphs are not isomorphic by proving that one of the above properties fails.

Example 1

- Determine whether G is isomorphic to H.

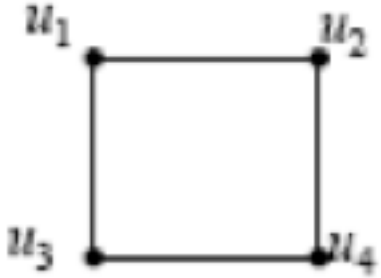


G

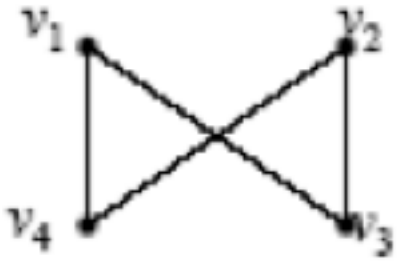


H

Example 1 - Solution

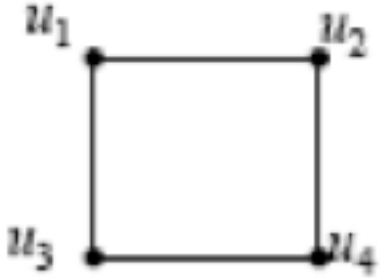


G

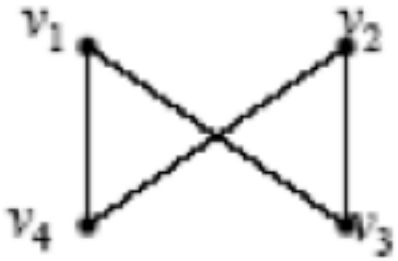


H

- Both graphs are simple and have the same number of vertices and the same number of edges.
- All the vertices of both graphs have degree 2.
- Define $f : U \rightarrow V$, where $U = \{ u_1, u_2, u_3, u_4 \}$ and $V = \{ v_1, v_2, v_3, v_4 \}$;
  $f(u_1) = v_1$; $f(u_2) = v_4$; $f(u_3) = v_3$; $f(u_4) = v_2$.



G



H

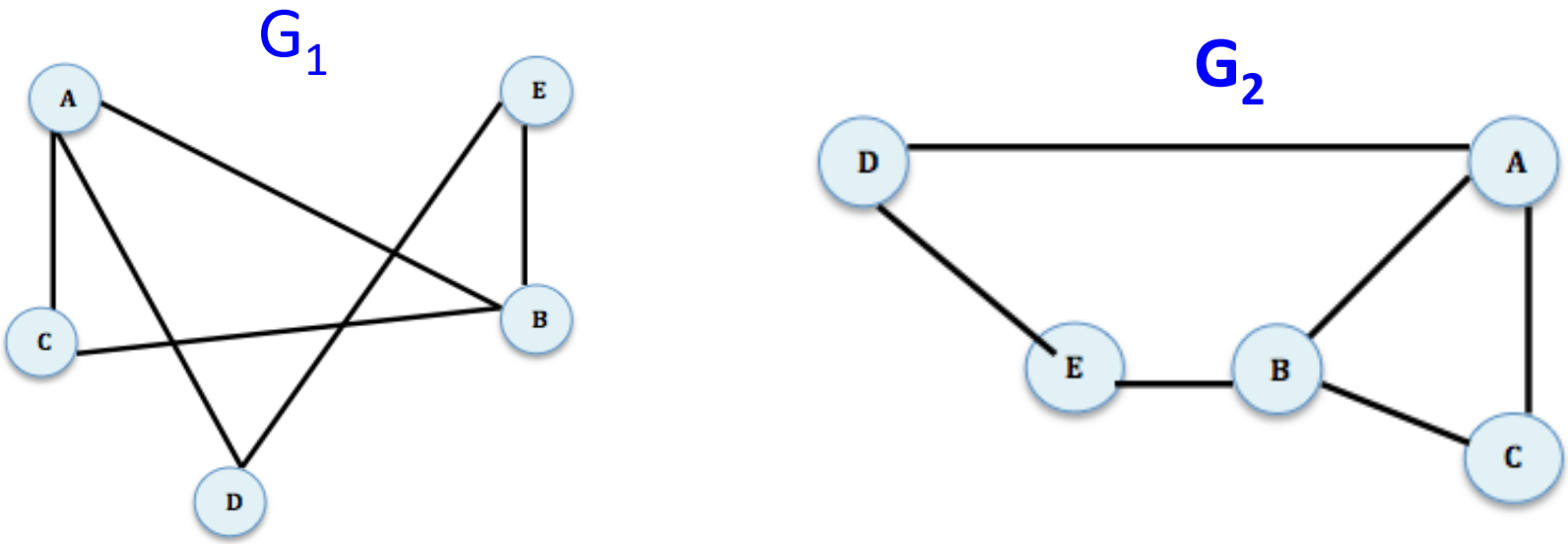
- To verify whether G and H are isomorphic, we examine the adjacency matrix A_G with rows and columns labeled in the order u_1, u_2 , u_3, u_4 , and the adjacency matrix A_H with rows and columns labeled in the order v_1, v_2 , v_3, v_4 .

- A_G and A_H are the same, G and H are isomorphic.

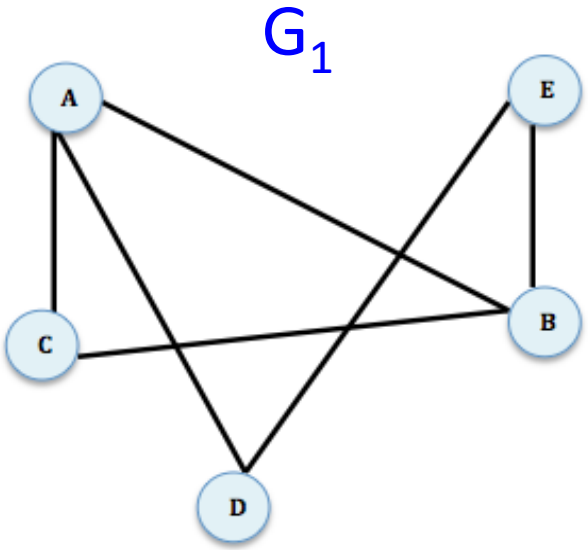
$$A_G = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{array} \begin{array}{c} \begin{array}{cccc} u_1 & u_2 & u_3 & u_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array} \qquad A_H = \begin{array}{c} \\ v_1 \\ v_4 \\ v_3 \\ v_2 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_4 & v_3 & v_2 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

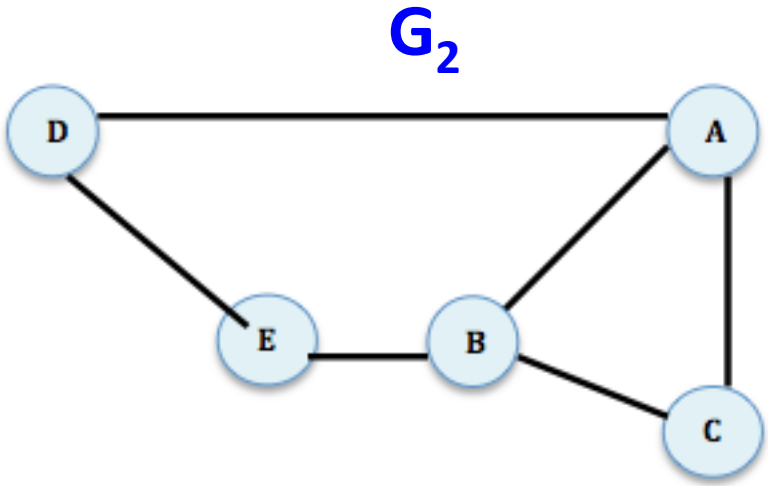
Exercise

Q: Show that the following two graphs are isomorphic.



Exercise Solution



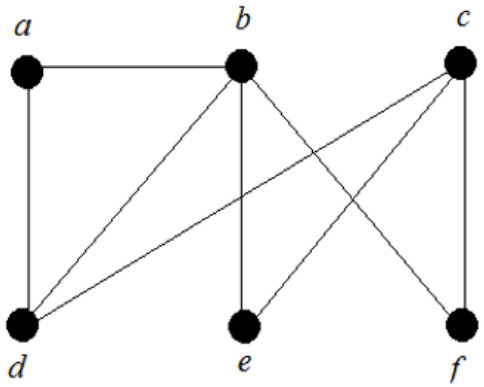


- Both have 5 vertices and 6 edges
- Both are connected and simple graph
- Both have 2 vertices with 3 degree and 3 vertices with 2 degree
- $f(A_{G_1}) = A_{G_2}$ $f(B_{G_1}) = B_{G_2}$
  $f(C_{G_1}) = C_{G_2}$ $f(D_{G_1}) = D_{G_2}$
  $f(E_{G_1}) = E_{G_2}$

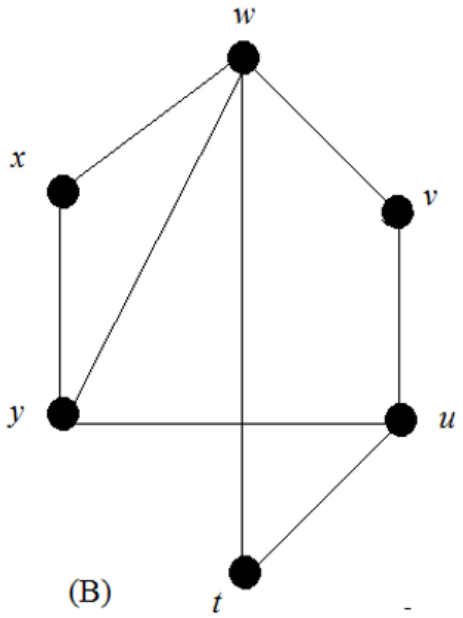
$\therefore$ G_1 and G_2 are isomorphic

Exercise
Past Year 2015/2016

Determine whether the graphs in Figure 2 (A and B) are isomorphic. If the graphs are isomorphic, find their adjacency matrices; otherwise, give an invariant that the graphs do not share. (6 marks)

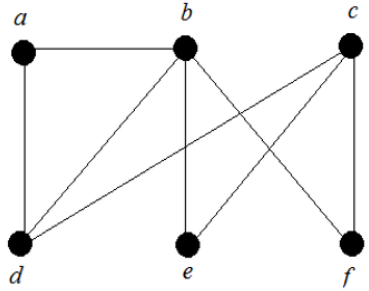


(A)

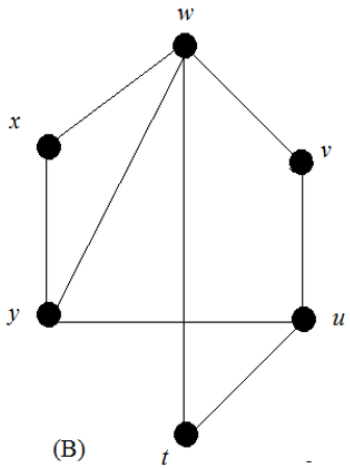


(B)

Exercise Solution
Past Year 2015/2016



(A)



(B)

Adjacency Matrix , A_A =

	a	b	c	d	e	f
a	0	1	0	1	0	0
b	1	0	0	1	1	1
c	0	0	0	1	1	1
d	1	1	1	0	0	0
e	0	1	1	0	0	0
f	0	1	1	0	0	0

Adjacency Matrix , A_B =

	x	w	u	y	t	v
x	0	1	0	1	0	0
w	1	0	0	1	1	1
u	0	0	0	1	1	1
y	1	1	1	0	0	0
t	0	1	1	0	0	0
v	0	1	1	0	0	0

- Both have 6 vertices and 8 edges
- Both are connected and simple graph
- Both have 1 vertex with 4 degree, 2 vertices with 3 degree and 3 vertices with 2 degree
- $f(a_A) = x_B$ $f(b_A) = w_B$
  $f(c_A) = u_B$ $f(d_A) = y_B$
  $f(e_A) = t_B$ $f(f_A) = v_B$

$\therefore$ A and B are isomorphic

Trails, Paths & Circuits

Term and Description

- A walk from v to w is a finite alternating sequence of adjacent vertices and edges of G. Thus a walk has the form

$$(v_0, e_1, v_1, e_2, v_2, \ldots\ldots, v_{n-1}, e_n, v_n)$$

where the v's represent vertices, the e's represent edges, $v = v_0$, $w = v_n$, and for $i = 1,2,\ldots,n$. v_{i-1} and v_i are the endpoints of e_i.

- A trivial walk from v to w consist of the single vertex v.

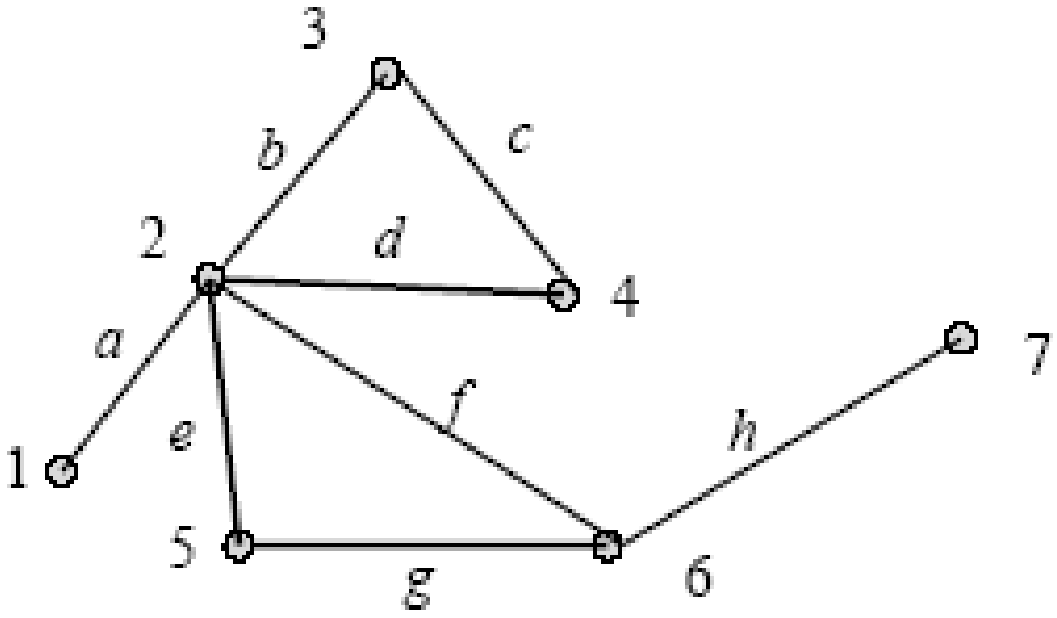
- The length of a walk is the number of edges it has.

Term and Description (cont.)

- A trail from v to w is a walk from v to w that does not contain a repeated edge.
- A path from v to w is a trail from v to w that does not contain a repeated vertex.
- A closed walk is a walk that start and ends at the same vertex.
- A circuit/cycle is a closed walk that contains at least one edge and does not contain a repeated edge.
- A simple circuit is a circuit that does not have any other repeated vertex except the first and the last.

Example 1 – Trail & Path

- (1, a, 2, b, 3, c, 4, d, 2, e, 5) is a trail.
- (6, g, 5, e, 2, d, 4) is a path.



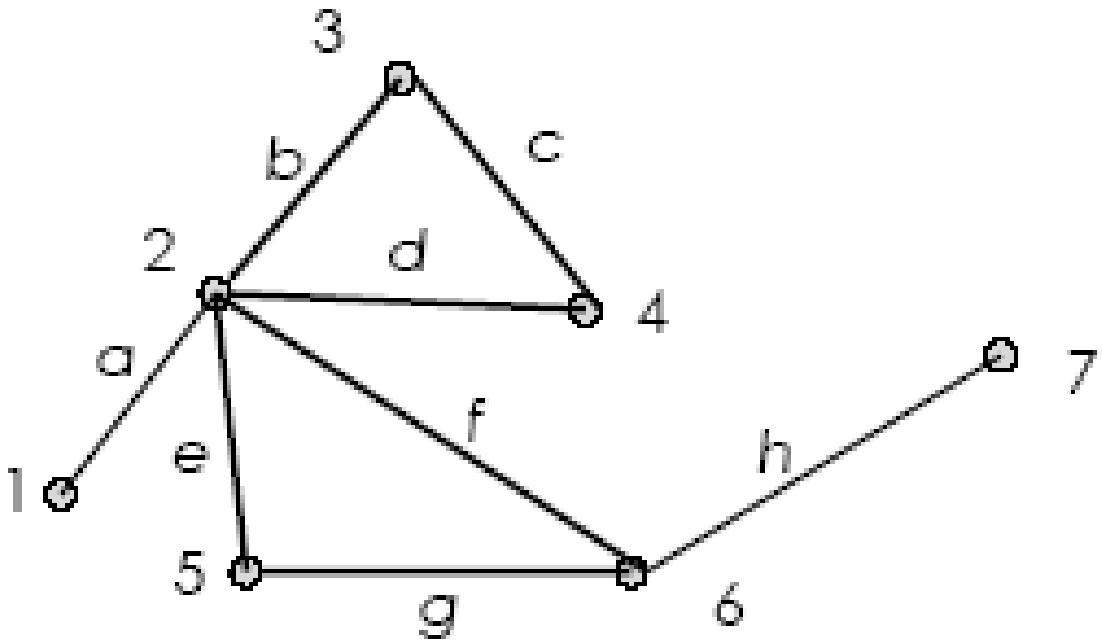
Note:

Trail: No repeated edge (can repeat vertex).

Path: No repeated vertex and edge.

Example 2 – Cycle/circuit

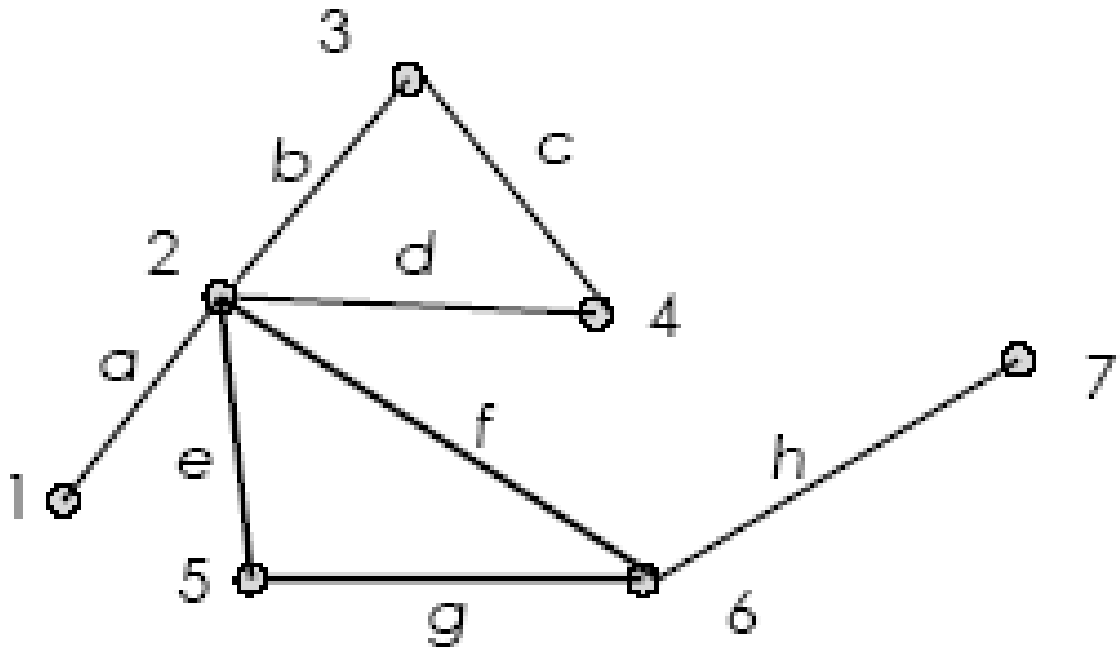
- (2, f, 6, g, 5, e, 2, d, 4, c, 3, b, 2) is a cycle.



Note: cycle –> start and end at same vertex, no repeated edge.

Example 3 – Simple Cycle

- (5, g, 6, f, 2, e, 5) is a simple cycle.

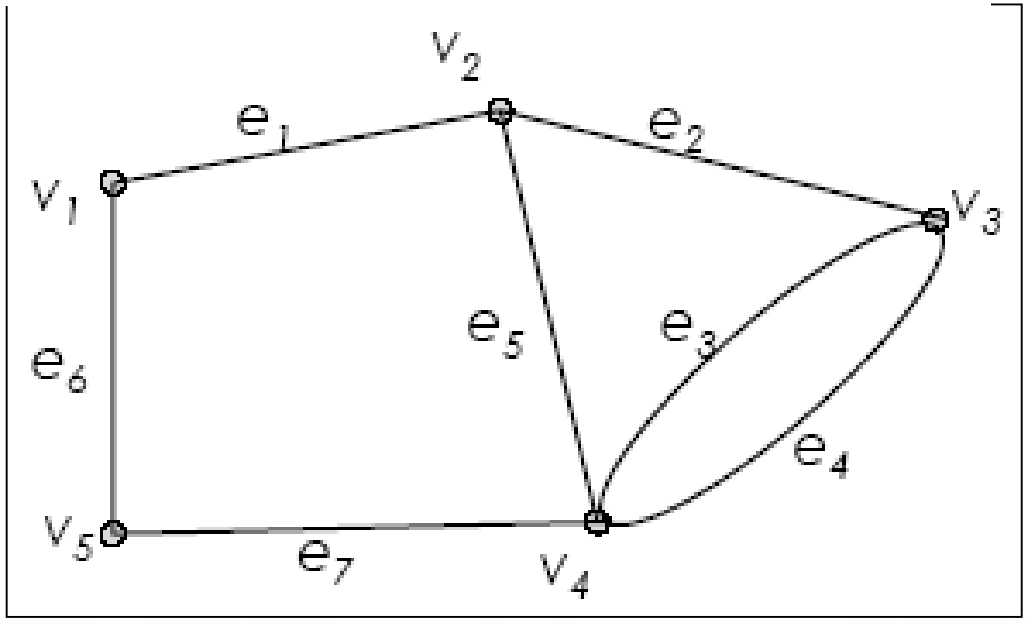


Note: Simple cycle –> start and end at same vertex, no repeated edge or vertex except for the start and end vertex.

Exercise

Tell whether the following is either a walk, trail, path, cycle, simple cycle, closed walk or none of these.

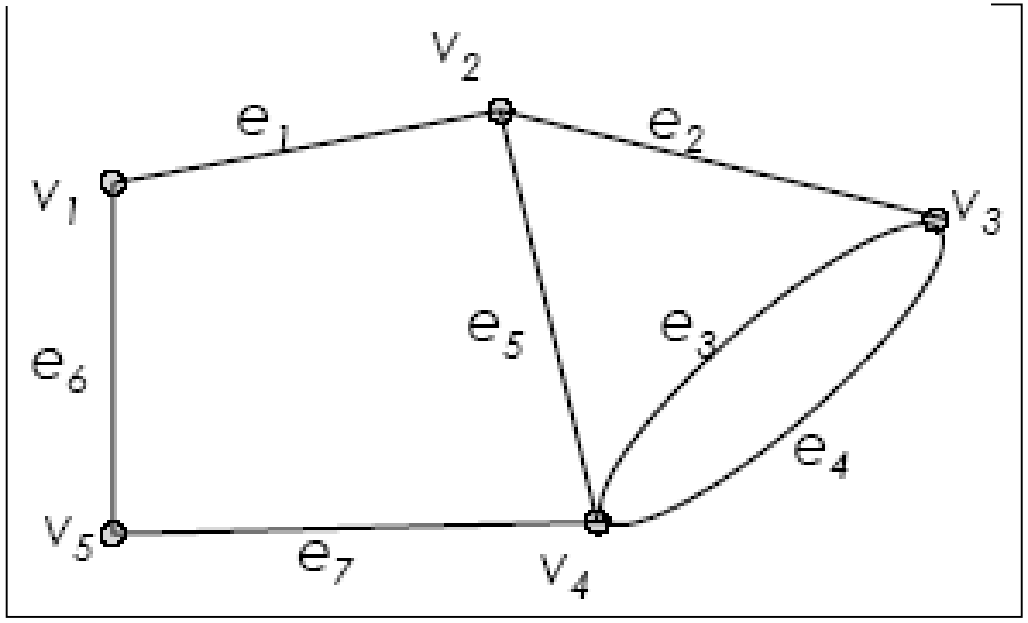
- (v_1, e_1, v_2)
- $(v_2, e_2, v_3, e_3, v_4, e_4, v_3)$
- $(v_4, e_7, v_5, e_6, v_1, e_1, v_2, e_2, v_3, e_3, v_4)$
- $(v_4, e_4, v_3, e_3, v_4, e_5, v_2, e_1, v_1, e_6, v_5, e_7, v_4)$



Exercise - Solution

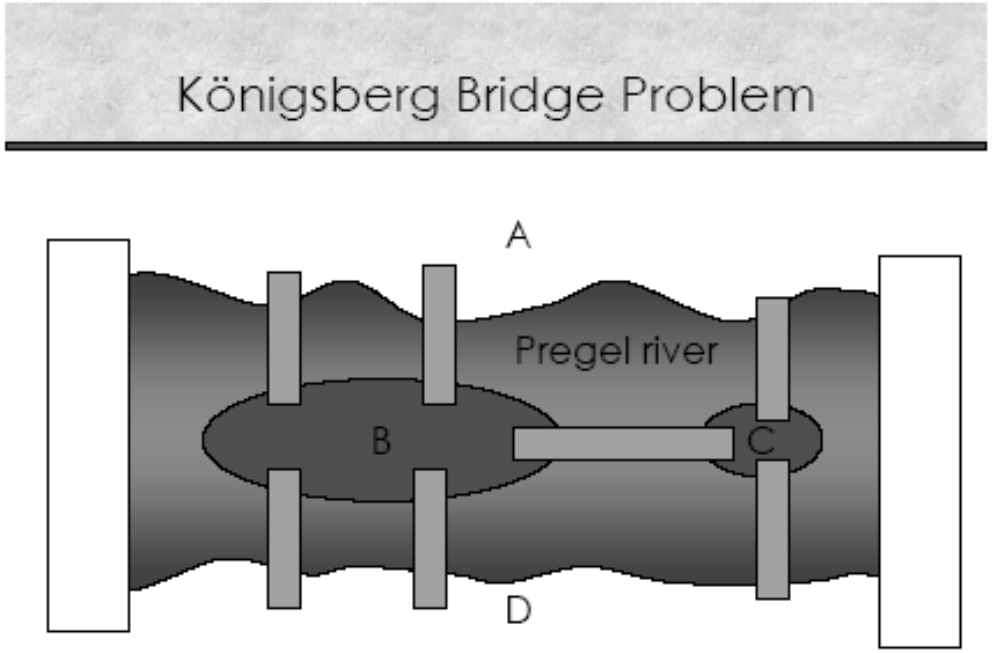
Tell whether the following is either a walk, trail, path, cycle, simple cycle, closed walk or none of these.

- (v_1, e_1, v_2) Trail
- $(v_2, e_2, v_3, e_3, v_4, e_4, v_3)$ Walk; Trail
- $(v_4, e_7, v_5, e_6, v_1, e_1, v_2, e_2, v_3, e_3, v_4)$ Simple cycle
- $(v_4, e_4, v_3, e_3, v_4, e_5, v_2, e_1, v_1, e_6, v_5, e_7, v_4)$ Cycle



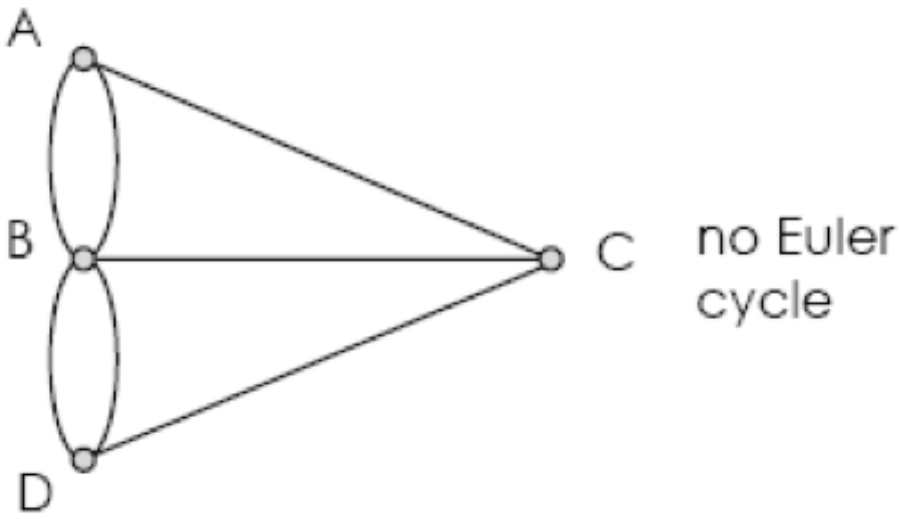
Euler Path & Circuit

The town of Konigsberg in Prussia (now Kaliningrad in Russia) was built at a point where two branches of the Pregel River came together. It consisted of an island and some land along the river banks. These were connected by seven bridges as shown in figure below:



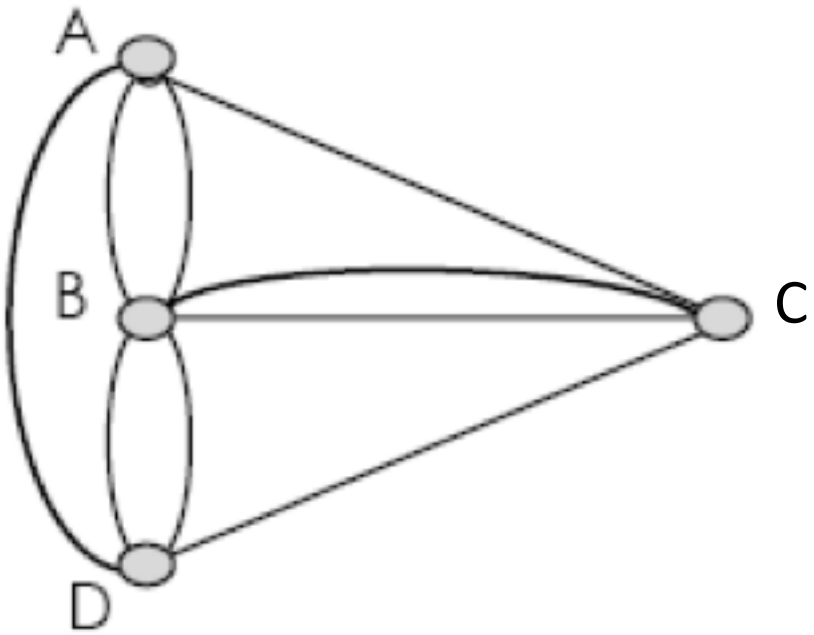
Problem: Starting at one land area, is it possible to walk across all of the bridges exactly once and return to the starting land area?

- Graph of the Königsberg Bridge Problem



It is not possible to walk across all of the bridges exactly once and return to the starting land area.

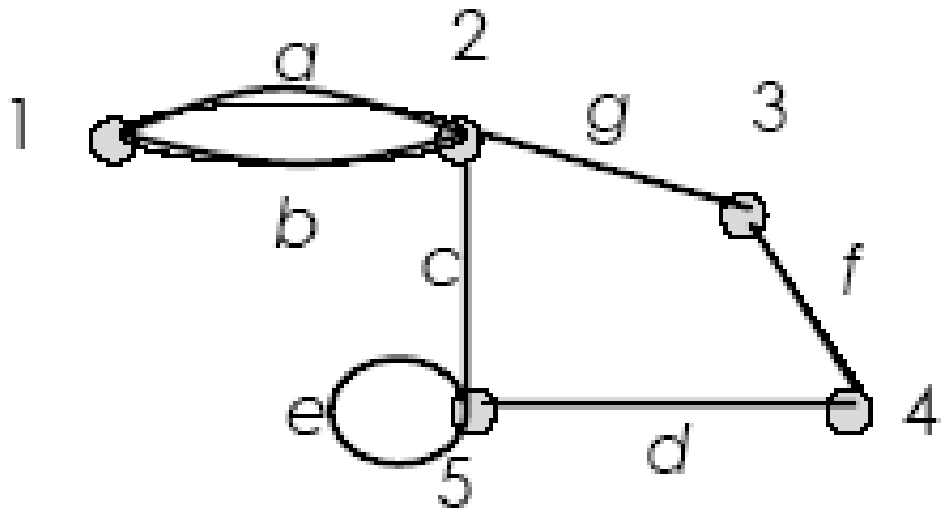
Solution: Two additional bridges have been constructed on the Pregel river.



Euler Circuit/Cycle

Let G be a graph. An Euler circuit for G is a circuit that contains every vertex and every edges of G. That is, an Euler circuit for G is a sequence of adjacent vertices and edges in G that has at least one edges, starts and ends at the same vertex, uses **every** vertex of G **at least** once, and uses **every** edge of G **exactly** once.

Example 1

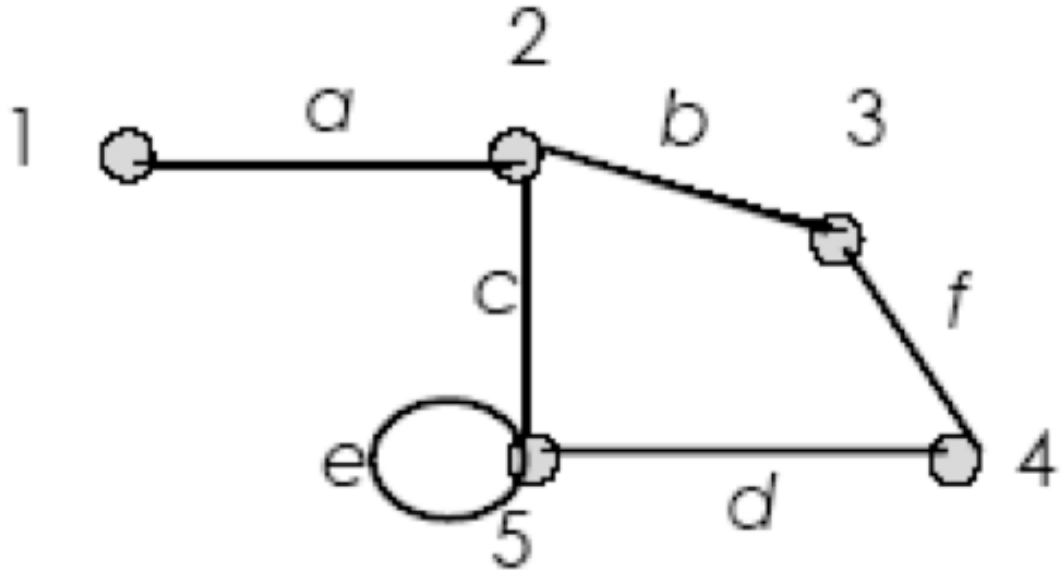


(1, a, 2, c, 5, e, 5, d, 4, f, 3, g, 2, b, 1) is an Euler cycle

Euler Trail

Let G be a graph, and let *v* and *w* be two distinct vertices of G. An Euler trail from *v* to *w* is a sequence of adjacent vertices and edges that starts at *v* and ends at *w*, passes through every **vertex** of G **at least** once, and traverses every **edge** of G **exactly** once.

Example 2



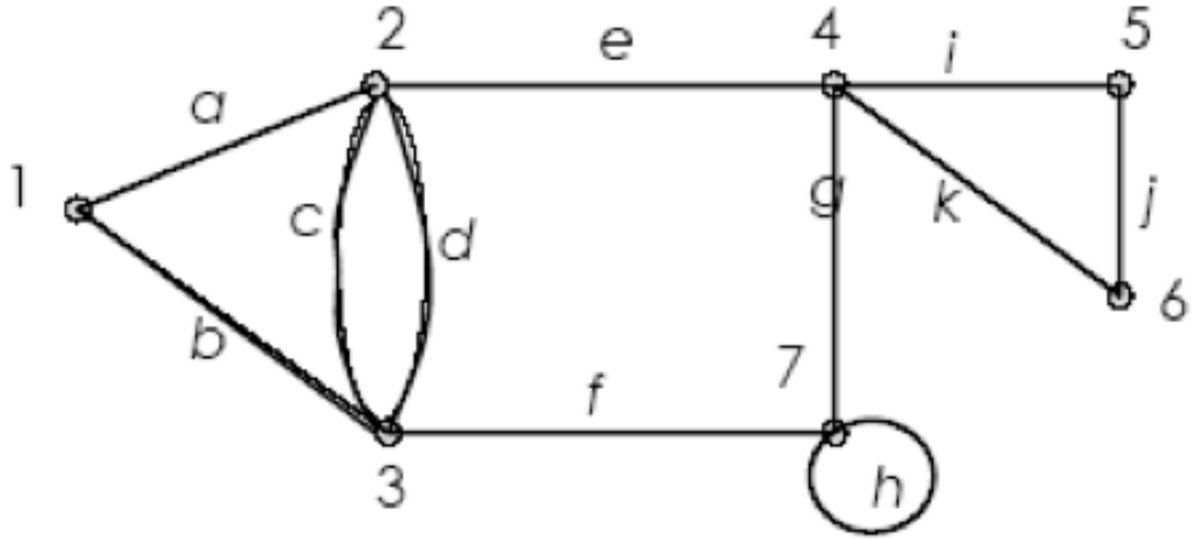
(1, a, 2, c, 5, e, 5, d, 4, f, 3, b, 2)
is an Euler trail

Theorem - Euler

- If G is a connected graph and every vertex has even degree, then G has an Euler circuit.

- A graph has an Euler trail from v to w ($v \neq w$) if and only if it is connected and v and w are the only vertices having odd degree.

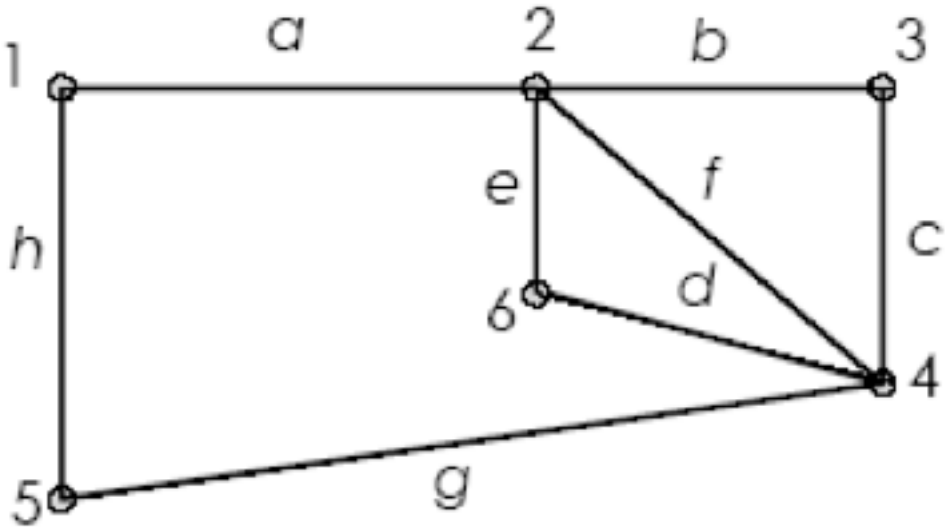
Example 3



This graph has an Euler cycle

Vertex	1	2	3	4	5	6	7
Degree	2	4	4	4	2	2	4

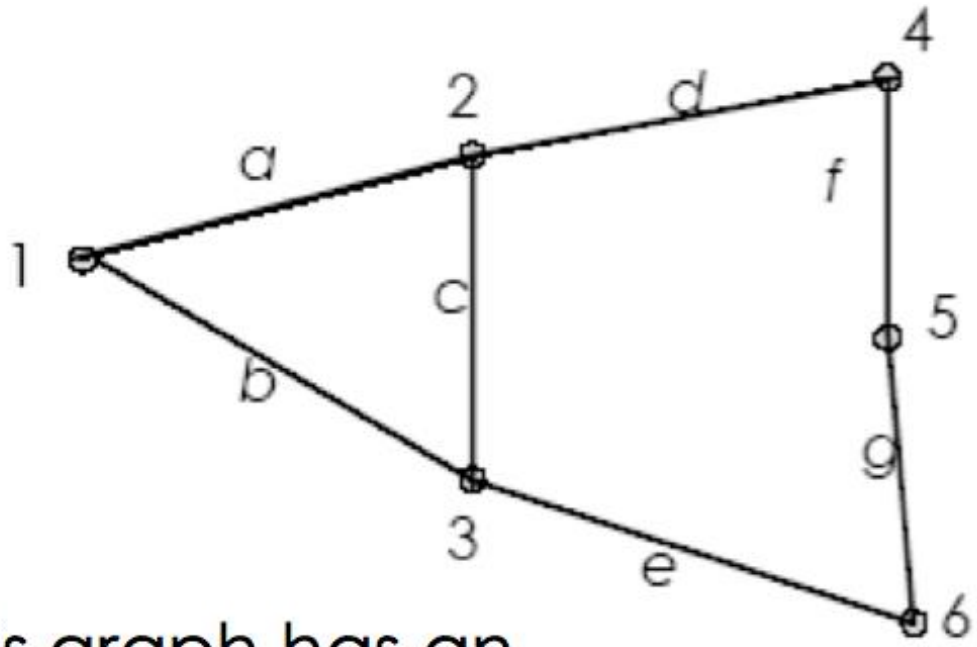
Example 4



This graph has an Euler cycle

Vertex	1	2	3	4	5	6
Degree	2	4	2	4	2	2

Example 5



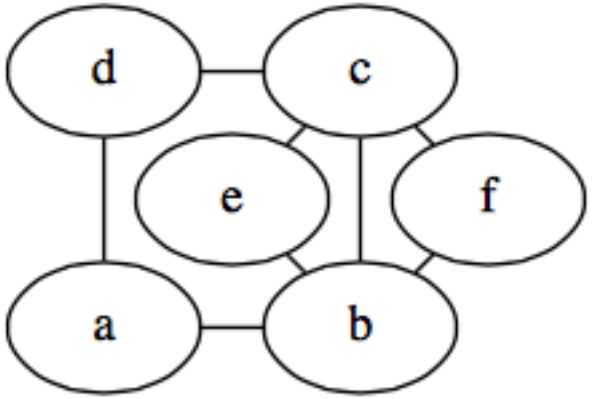
This graph has an Euler trail.

Vertex	1	2	3	4	5	6
Degree	2	3	3	2	2	2

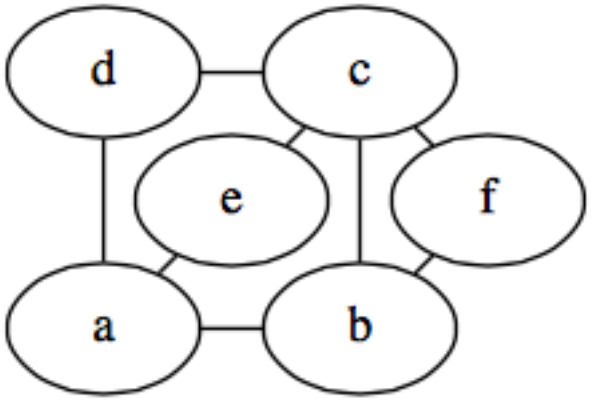
Exercise

Q: Which of the following graphs has Euler circuit?
Justify your answer.

G_1

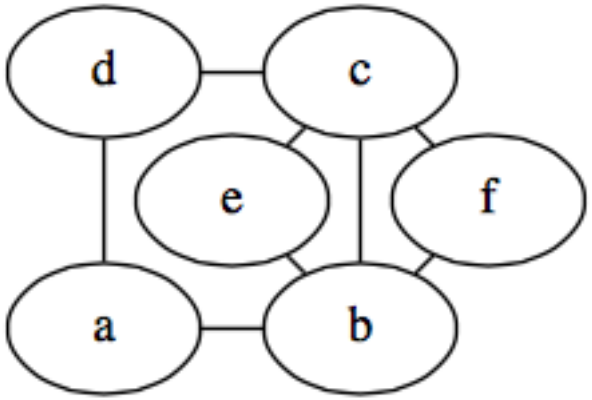


G_2

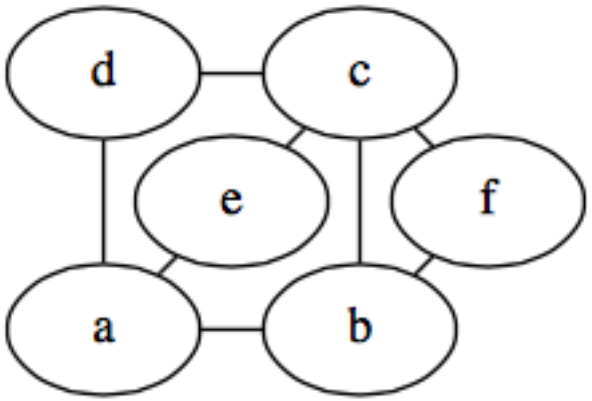


Exercise Solution

G_1



G_2



Q: Which of the following graphs has Euler circuit?
Justify your answer.

- G_1 has Euler circuit since all vertices have even degree
- G_2 does not has Euler circuit since 2 vertices (vertex *a* & *b*) have odd degree

Exercise
Past Year 2015/2016

Determine whether the graph in Figure 3 has an Euler cycle or Euler path. If the graph has an Euler cycle or Euler path, exhibit one; otherwise, give an argument that shows there is no Euler path. (4 marks)

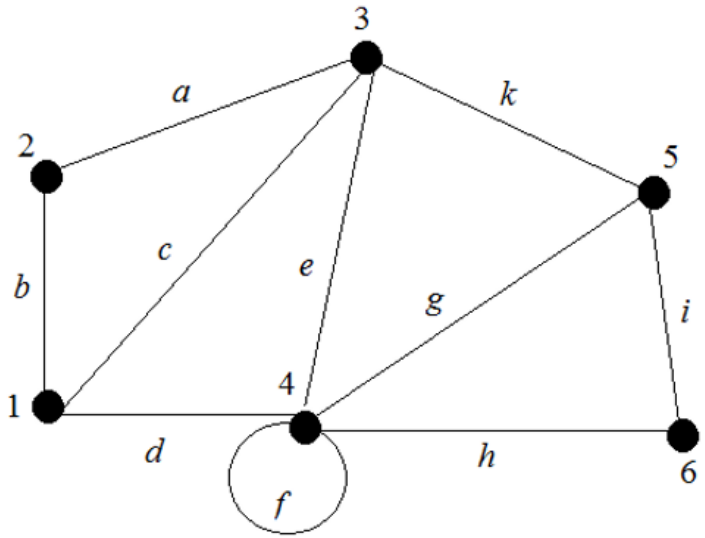
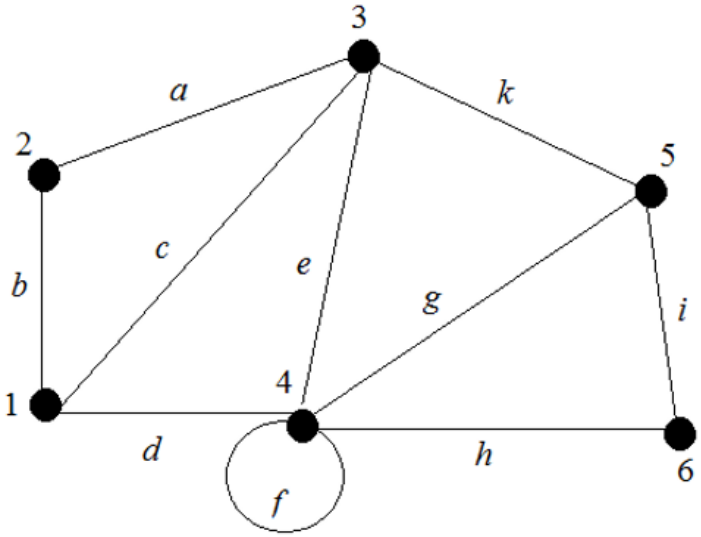


Figure 3

Exercise Solution
Past Year 2015/2016



- The graph does not has Euler circuit because 2 vertices (vertex *1* & *5*) have odd degree
- Hence, the graph has Euler path
- *1-b-2-a-3-c-1-d-4-f-4-e-3-k-5-g-4-h-6-i-5*

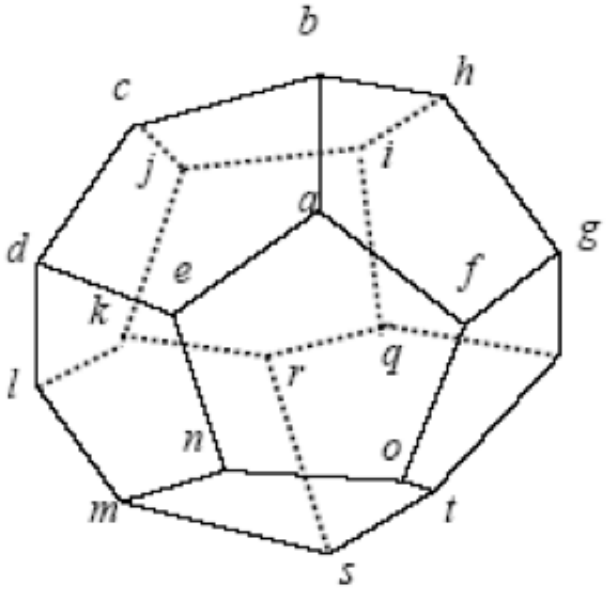
Hamilton Circuits

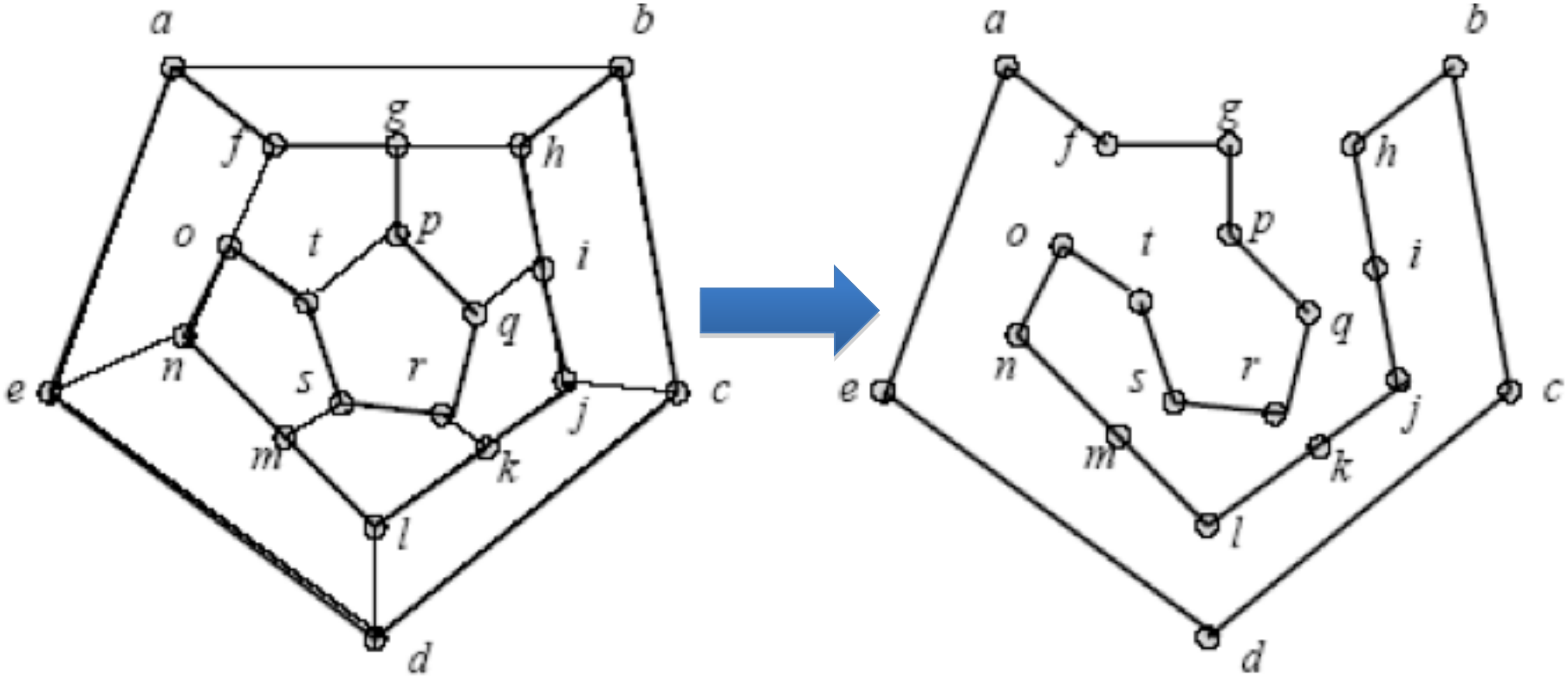
Hamiltonian Circuits

Given a graph G, a Hamiltonian circuit for G is a simple circuit that includes every vertex of G (but doesn't need to include all edges). That is, a Hamiltonian circuit for G is a sequence of adjacent vertices and distinct edges in which every vertex of G appears **exactly** once, except for the first and the last, which are the same.

Example

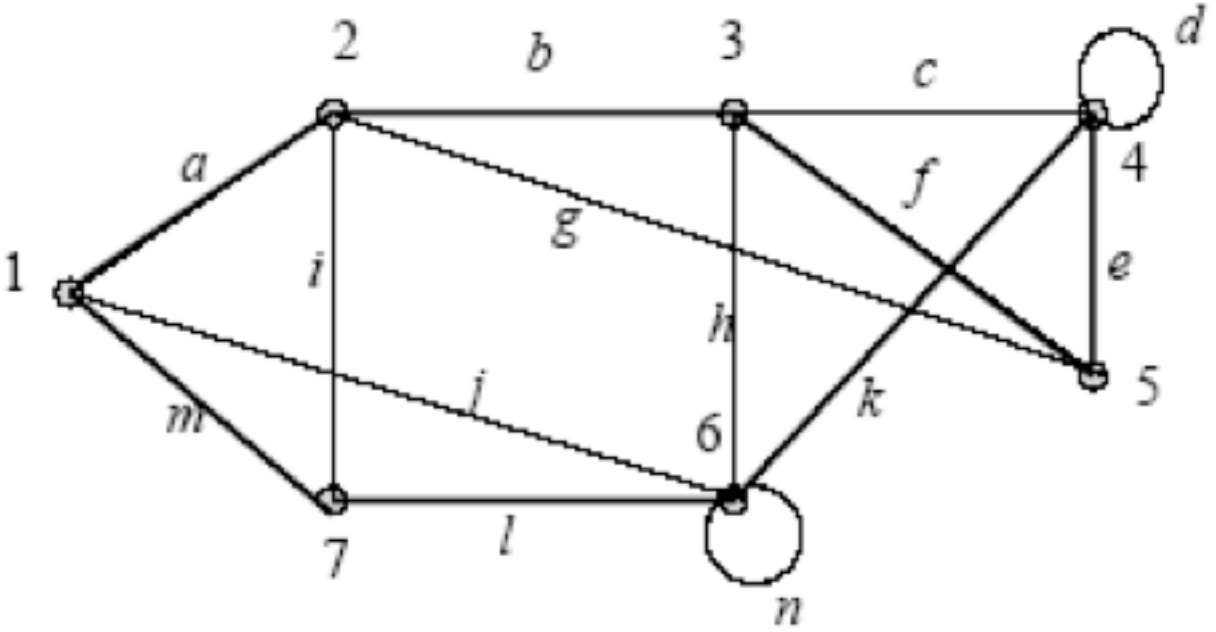
- Sir William Rowan Hamilton marketed a puzzle in the mid-1800s in the form of dedocahedron.
- Each corner bore the name of a city.
- The problem was to start at any city, travel along the edges, visit each city exactly one time and return to the initial city.





(a): The graph (b): Hamilton circuit

Example



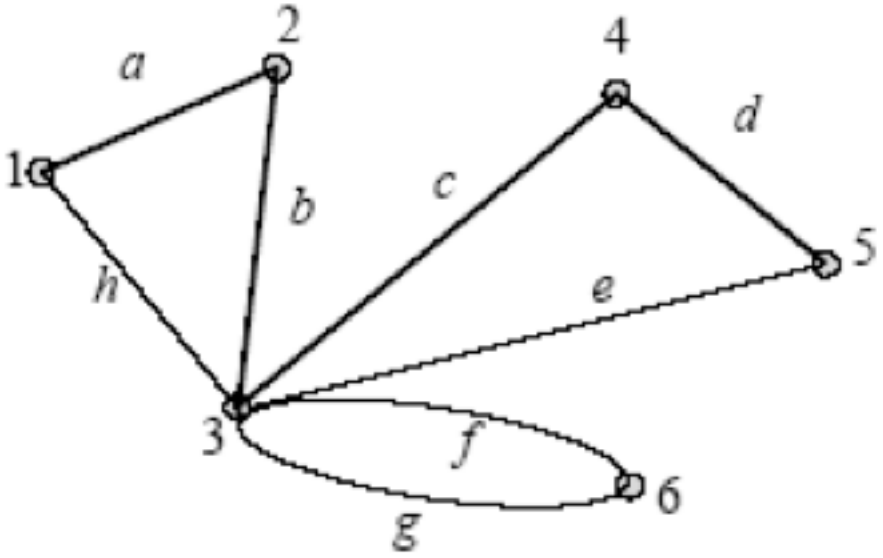
This graph has a Hamilton circuit.

(1, a, 2, b, 3, f, 5, e, 4, k, 6, l, 7, m, 1)

- Visit each vertex just once.

Example

This graph does not contain Hamilton circuit.

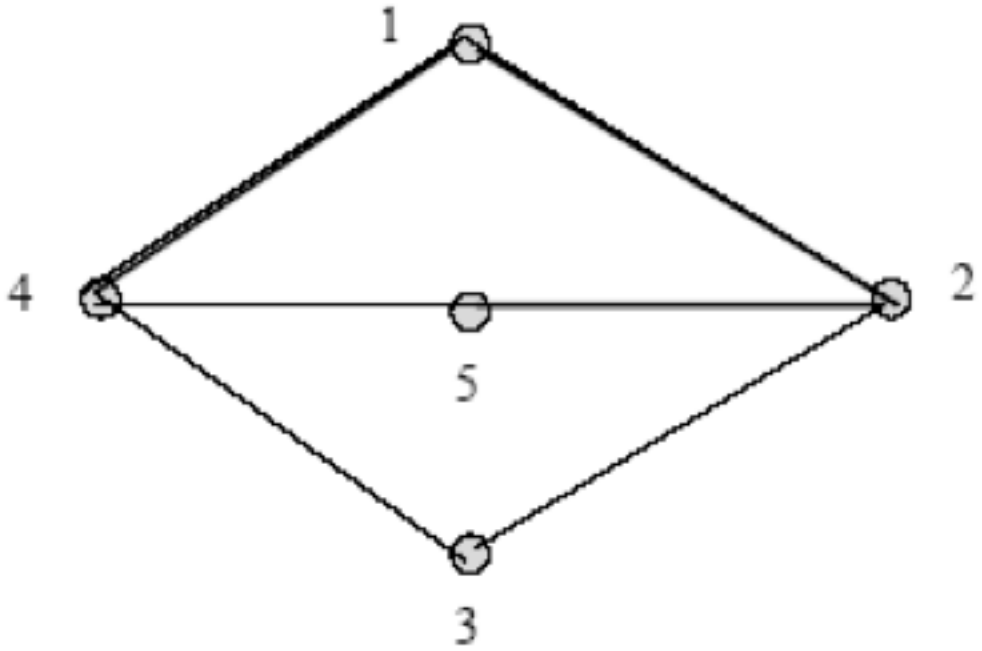


(1, a, 2, b, 3, g, 6, f, 3, e, 5, d, 4, c, 3, h, 1)

- Vertex (3) has to be visited more than once.

Example

Question: Is this graph has Hamiltonian cycle?

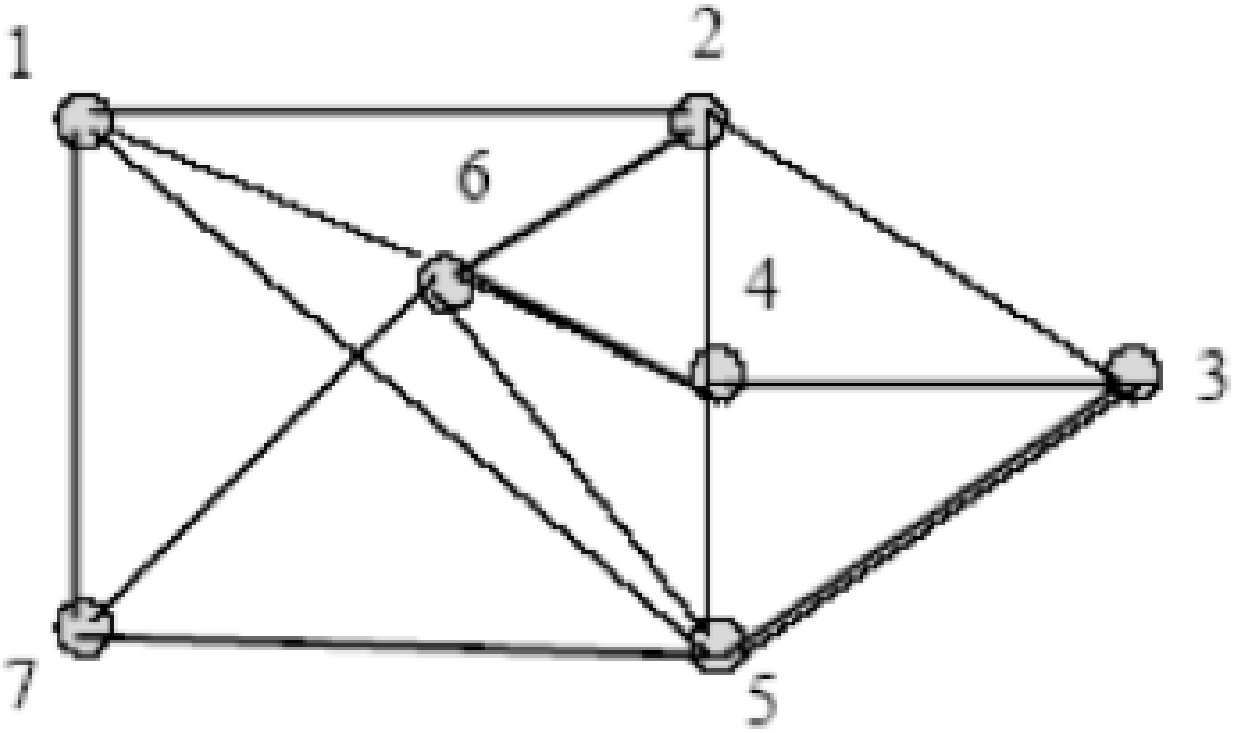


Solution: No.→ (1, 4, 3, 2, 5, 4, 1)

- Vertex (4) has to be visited more than once.

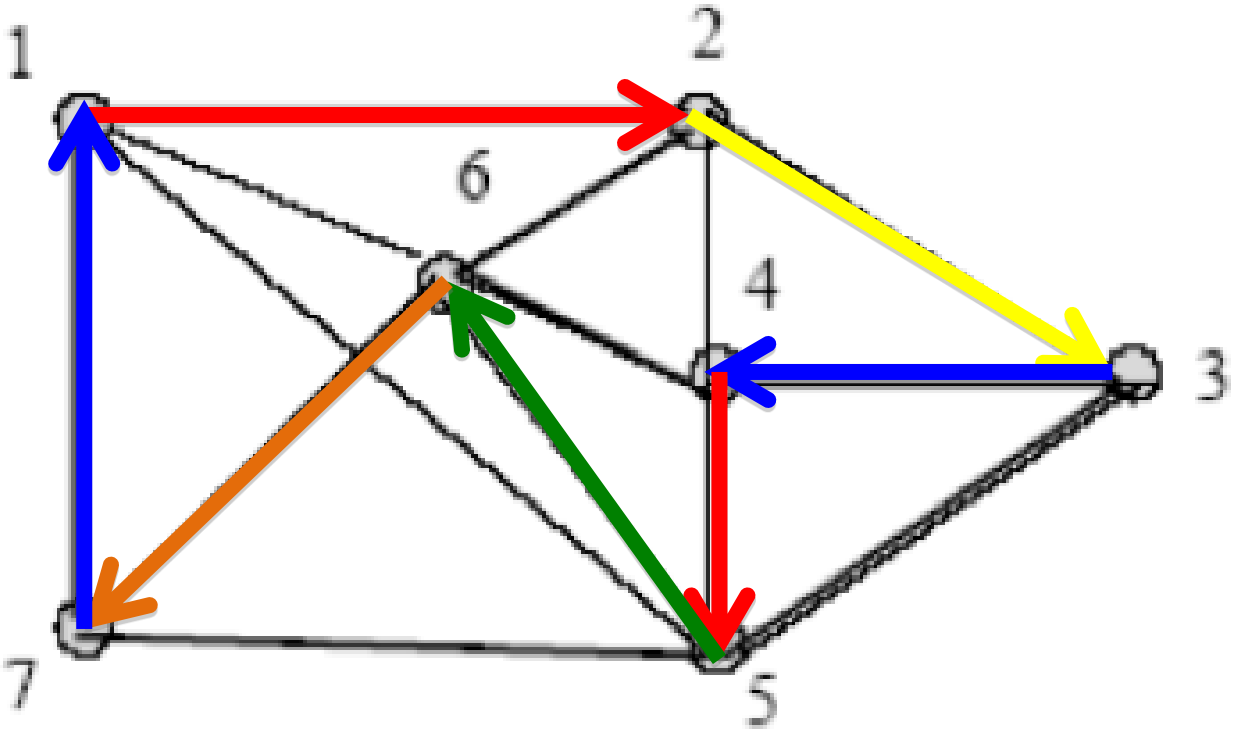
Exercise

Question: Is this graph has Hamiltonian cycle?



Exercise - Solution

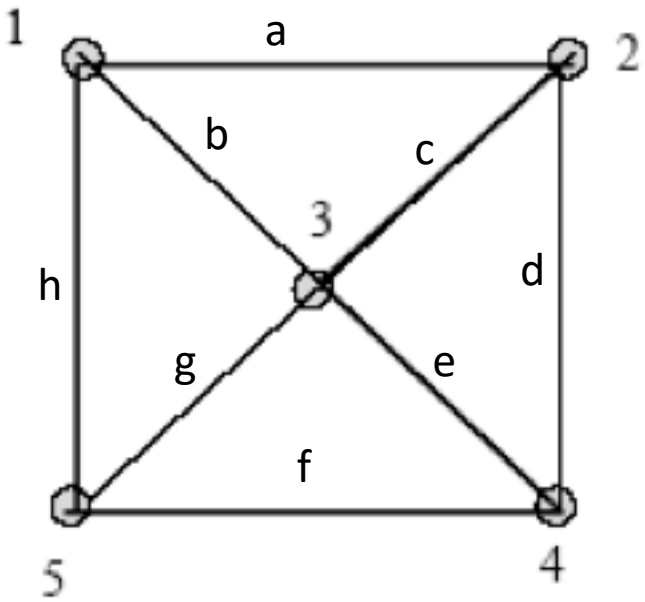
Manually check: This graph has Hamiltonian cycle.



Each vertex has to be visited only once.
For example: (1, 2, 3, 4, 5, 6, 7, 1)

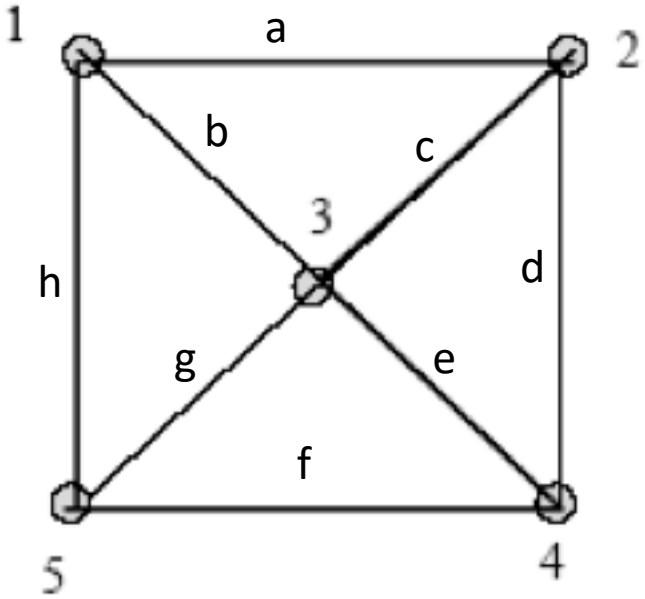
Exercise

Question: Prove that this graph has Hamiltonian cycle.



Exercise - Solution

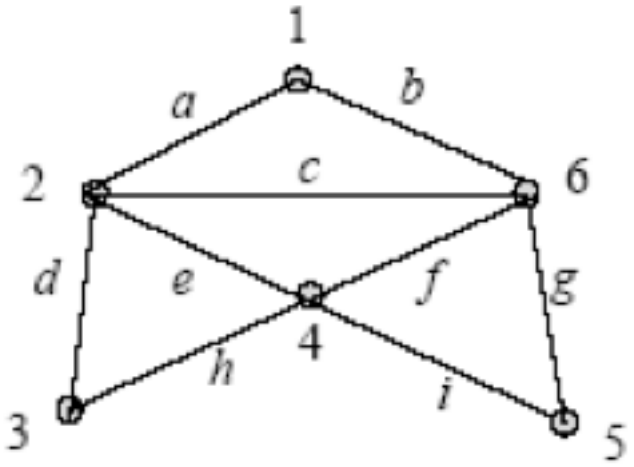
Question: Prove that this graph has Hamiltonian cycle.



Solution: (1, b, 3, c, 2, d, 4, f, 5, h, 1)

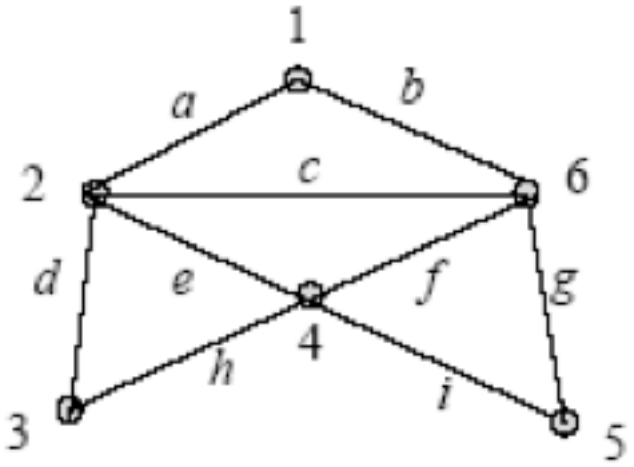
Exercise

- Find a Hamiltonian cycle in this graph.



Exercise - Solution

- Find a Hamiltonian cycle in this graph.



Solution: (1, b, 6, g, 5, i, 4, h, 3, d, 2, a, 1)

Exercise
Past Year 2015/2016

Determine whether the graph in Figure 4 has an Hamiltonian cycle. If yes, exhibit one.

(3 marks)

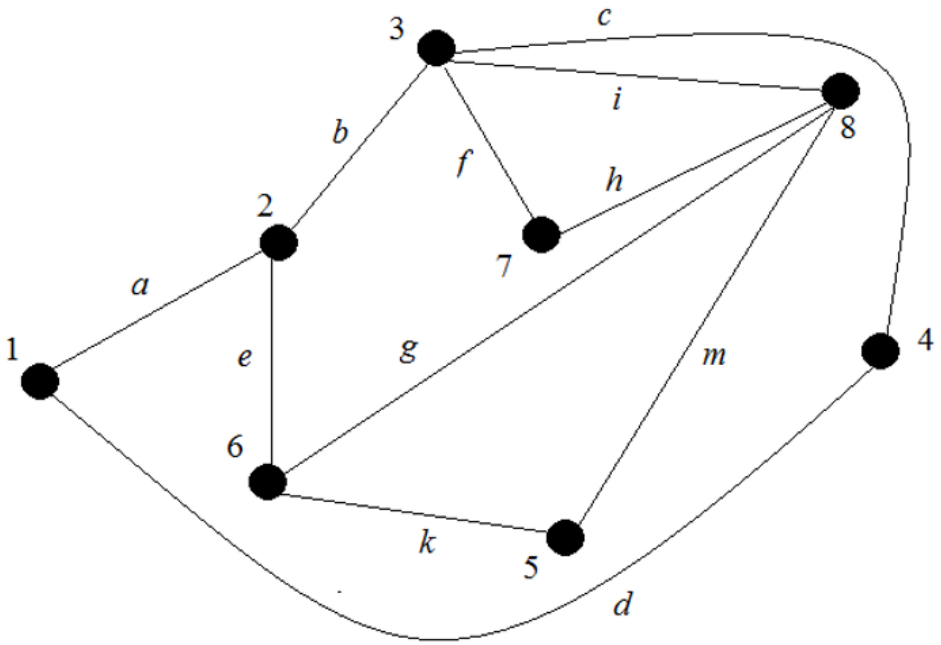
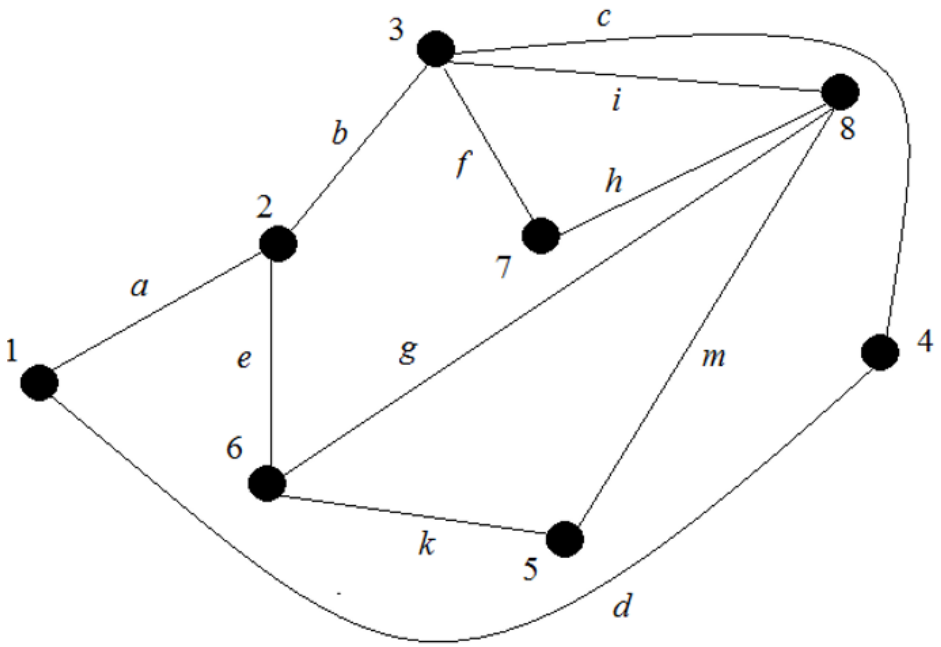


Figure 4

Exercise Solution
Past Year 2015/2016



- The graph has Hamiltonian cycle
- *7-f-3-c-4-d-1-a-2-e-6-k-5-m-8-h-7*

Shortest Path Problem

Shortest Path

- Let ***G*** be a weighted graph.
- Let ***u*** and ***v*** be two vertices in ***G***, and let ***P*** be a path in ***G*** from ***u*** to ***v***.
- The length of path ***P***, written ***L*(*P*)**, is the sum of the weights of all the edges on path ***P***.
- A **shortest path** from a vertex to another vertex is a path with the shortest length between the vertices.

Dijkstra's Shortest Path Algorithm

- 1. $S := \emptyset$
- 2. $N := V$
- 3. For all vertices, $u \in V$, $u \neq a$, $L(u) := \infty$
- 4. $L(a) := 0$

- 5. While $z \notin S$ do,

5.a :Let $v \in N$ be such that

$$L(v) = \min\{L(u) \mid u \in N\}$$

5.b : $S := S \in \{v\}$

5.c : $N := N - \{v\}$

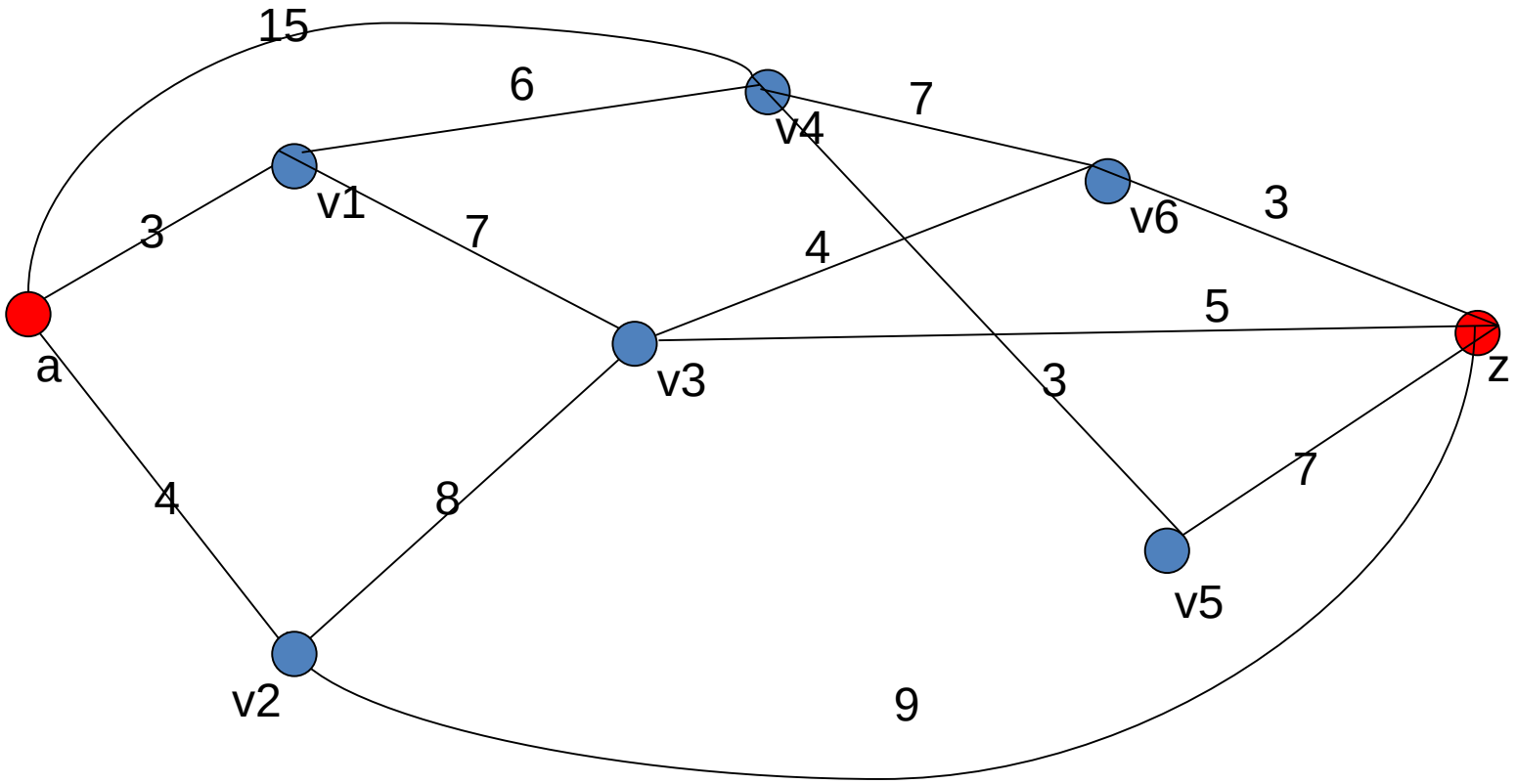
5.d : For all $w \in N$ such that there is an edge from v to w

5.d.1: If $L(v) + W[v,w] < L(w)$ then $L(w) = L(v) + W[v,w]$

http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

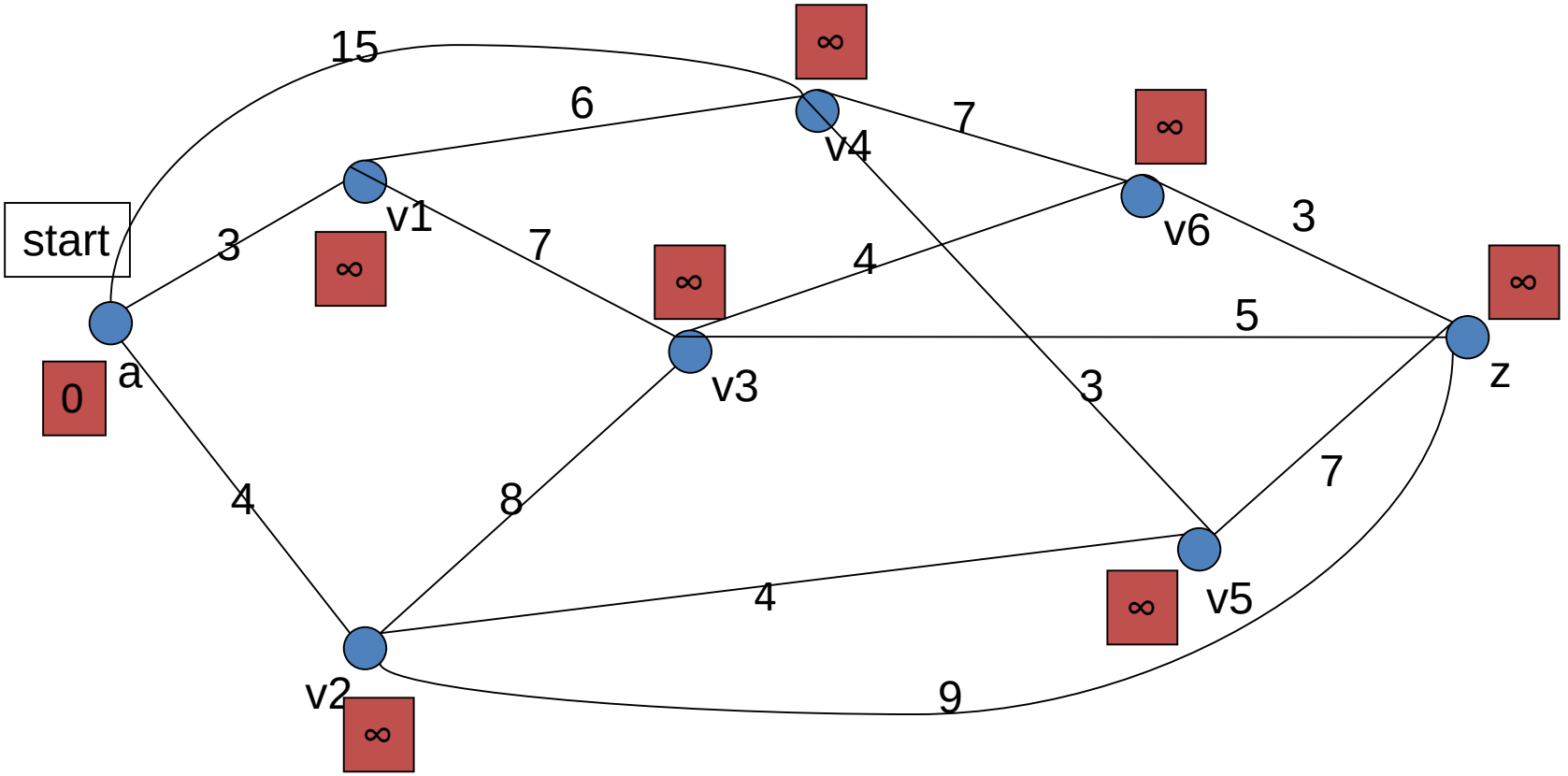
Example

What is the shortest path from **a** to **z**?



$S=\emptyset$

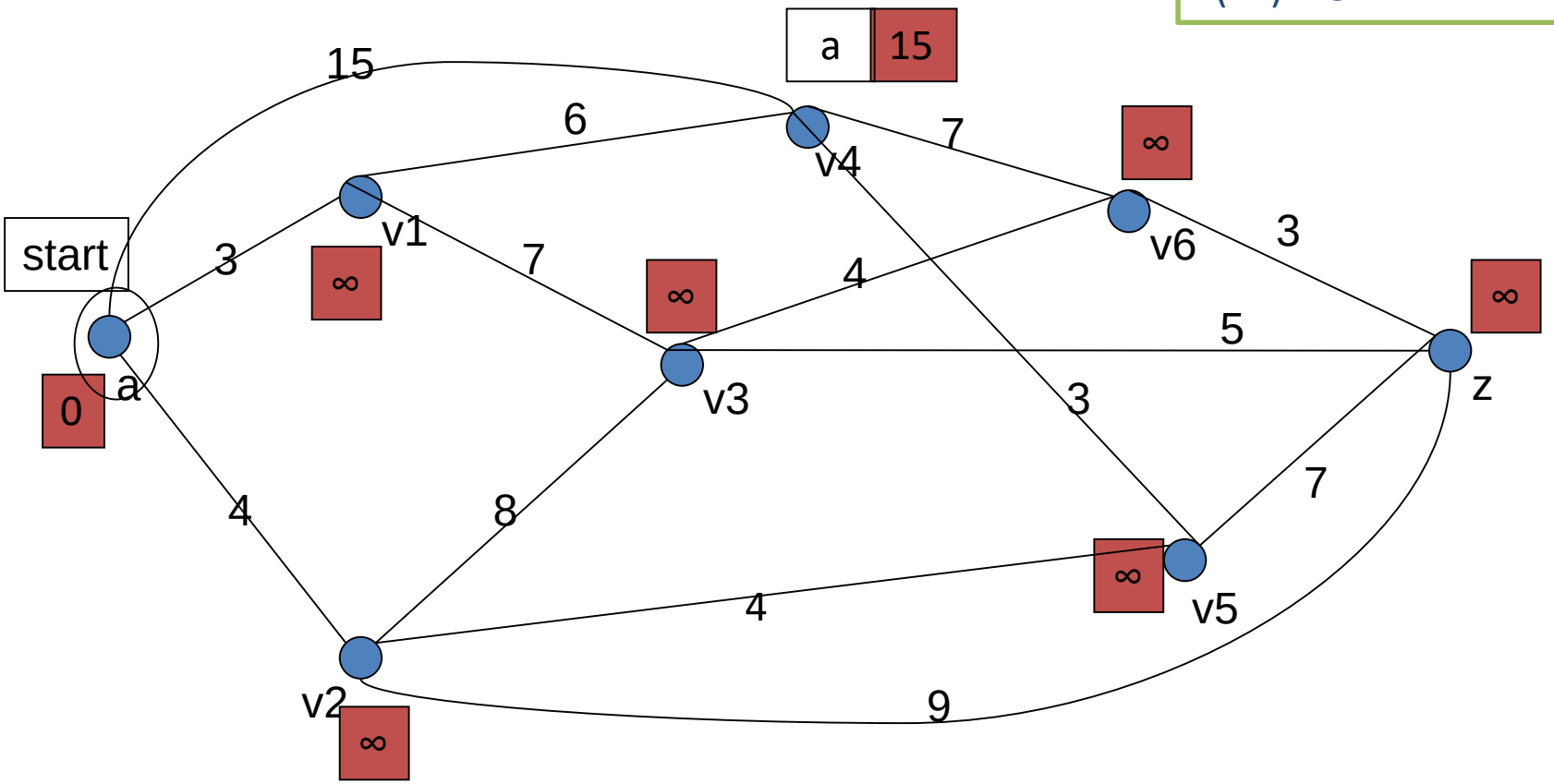
$N=\{a,v1,v2,v3,v4,v5,v6,z\}$



$S=\{a\}$

$N=\{v1,v2,v3,v4,v5,v6,z\}$

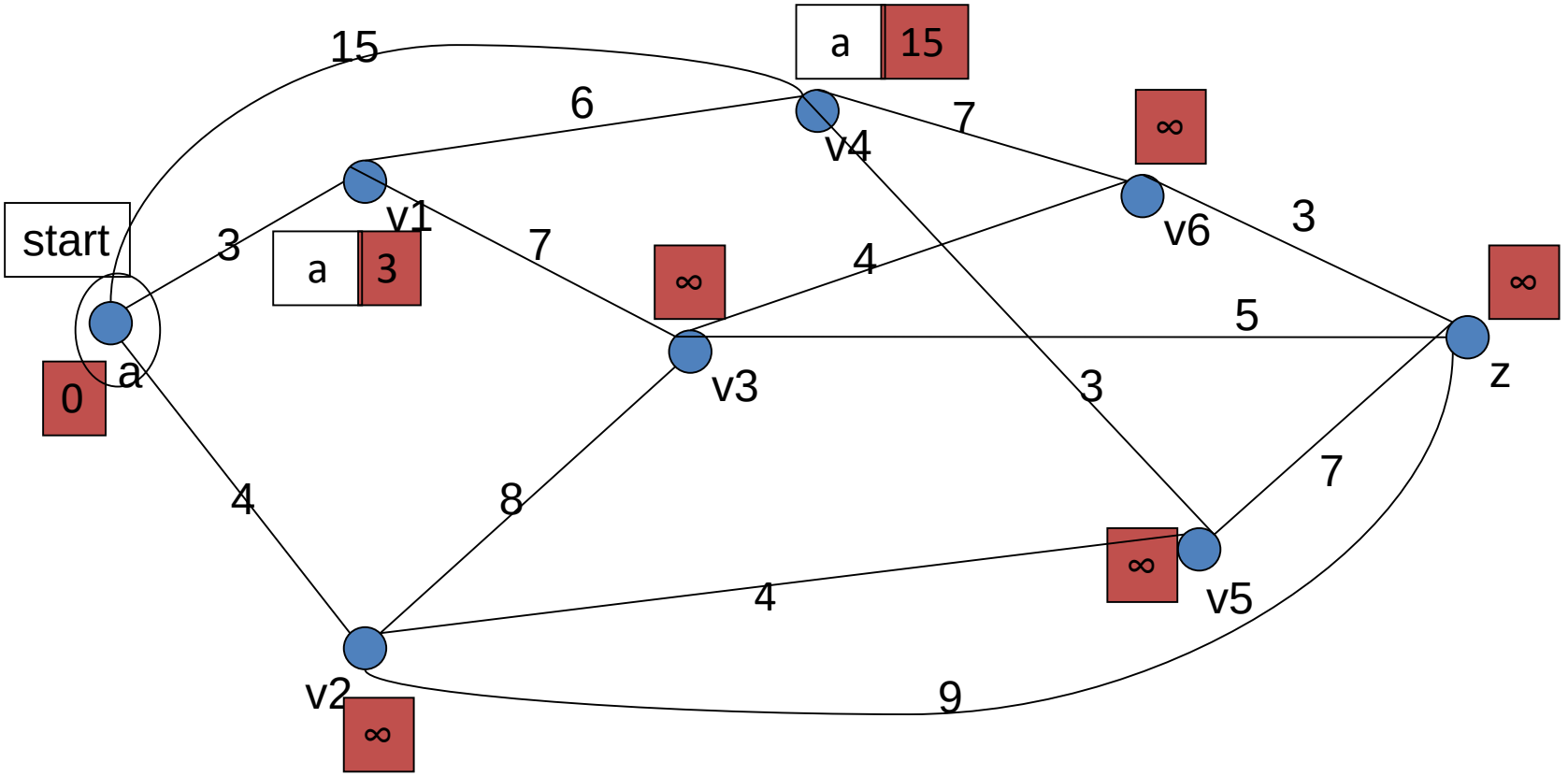
$L(a)+W[a,v4] < L(v4)$
$0+15=15 < \infty$
$L(v4)=15$



$S=\{a\}$

$N=\{v1,v2,v3,v4,v5,v6,z\}$

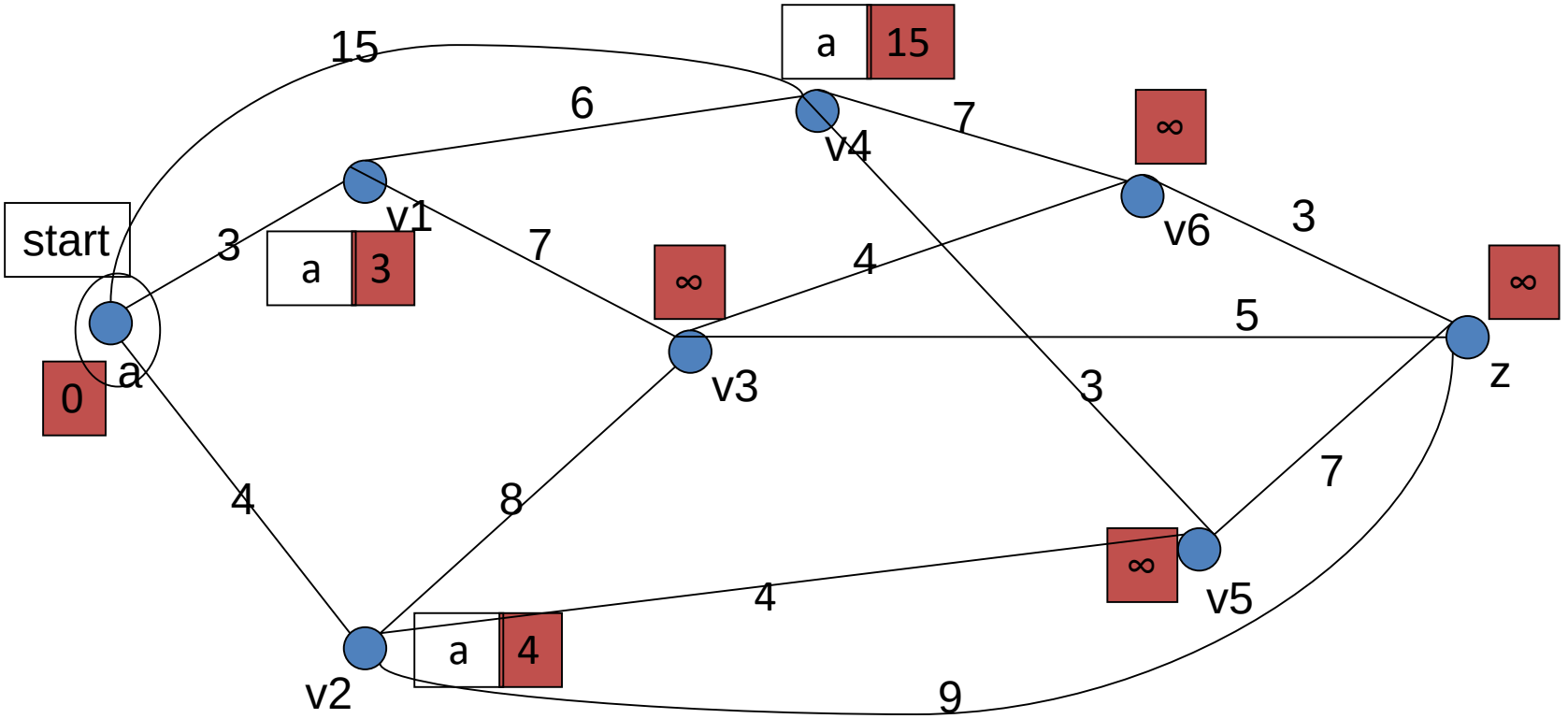
$L(a)+W[a,v1] < L(v1)$
$0+3=3 < \infty$
$L(v1)=3$



$S=\{a\}$

$N=\{v1,v2,v3,v4,v5,v6,z\}$

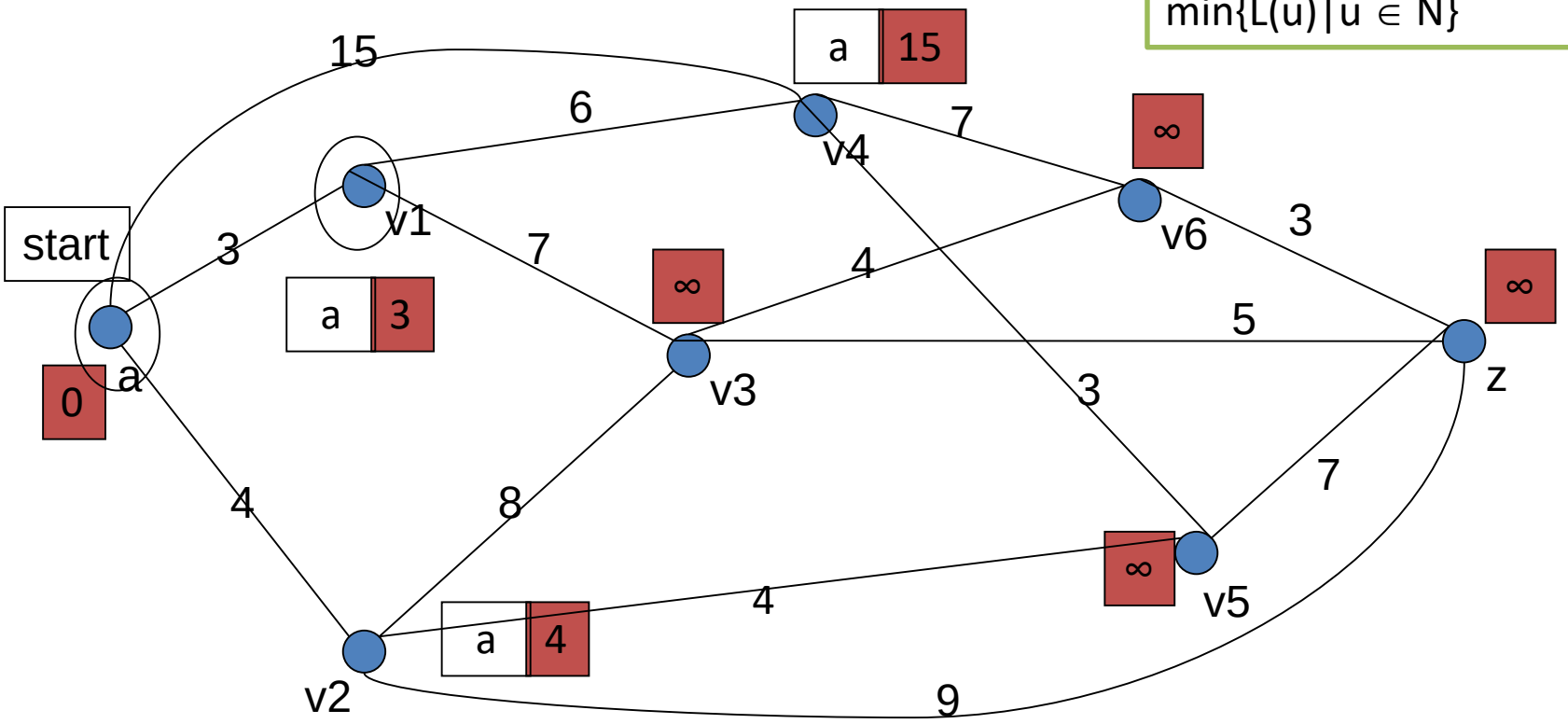
$L(a)+W[a,v2] < L(v2)$
$0+4=4 < \infty$
$L(v2)=4$



$S=\{a\}$

$N=\{v1,v2,v3,v4,v5,v6,z\}$

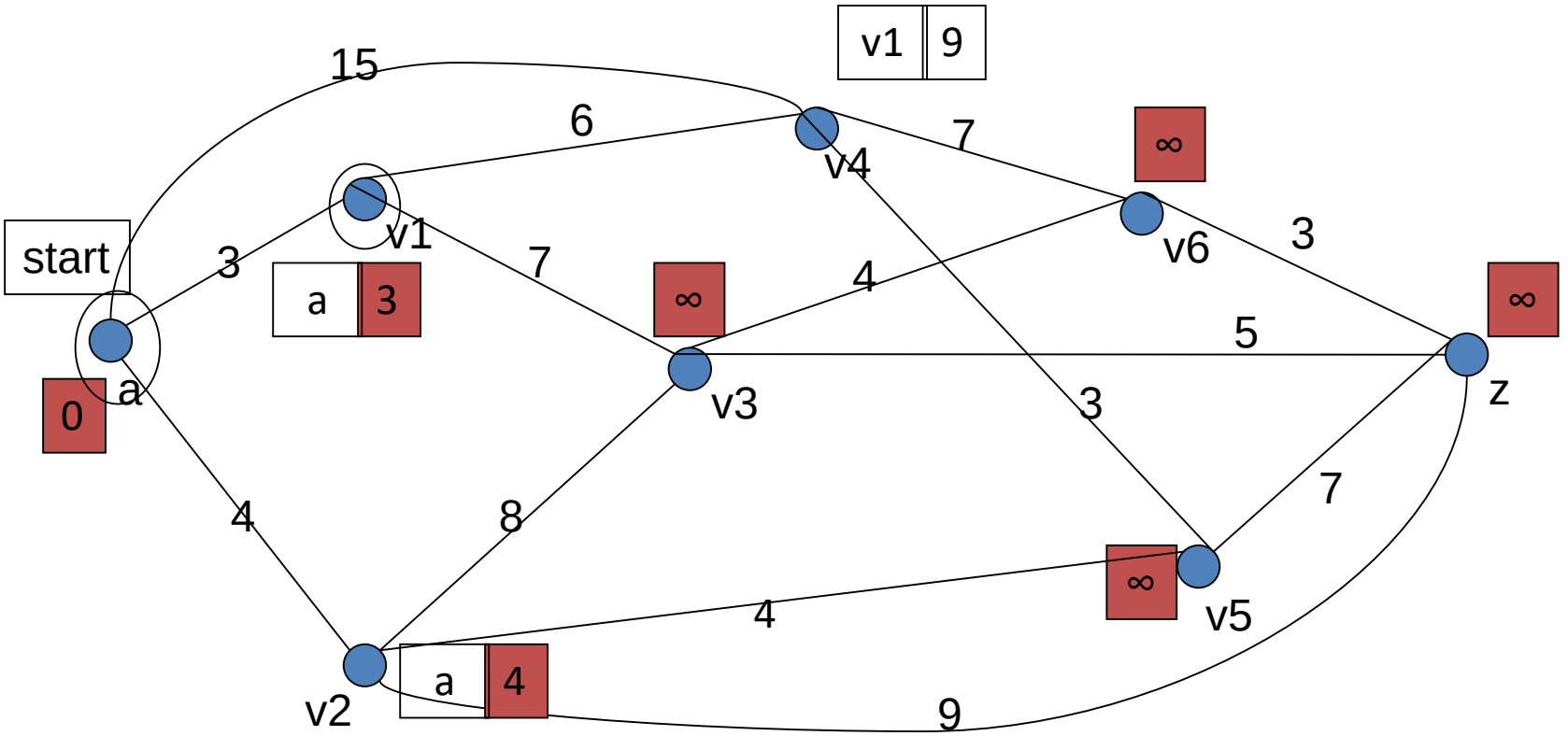
choose $v1$
because
$L(v1)=3$ =
$\min\{L(u) \mid u \in N\}$



$S=\{a, v1\}$

$N=\{v2,v3,v4,v5,v6,z\}$

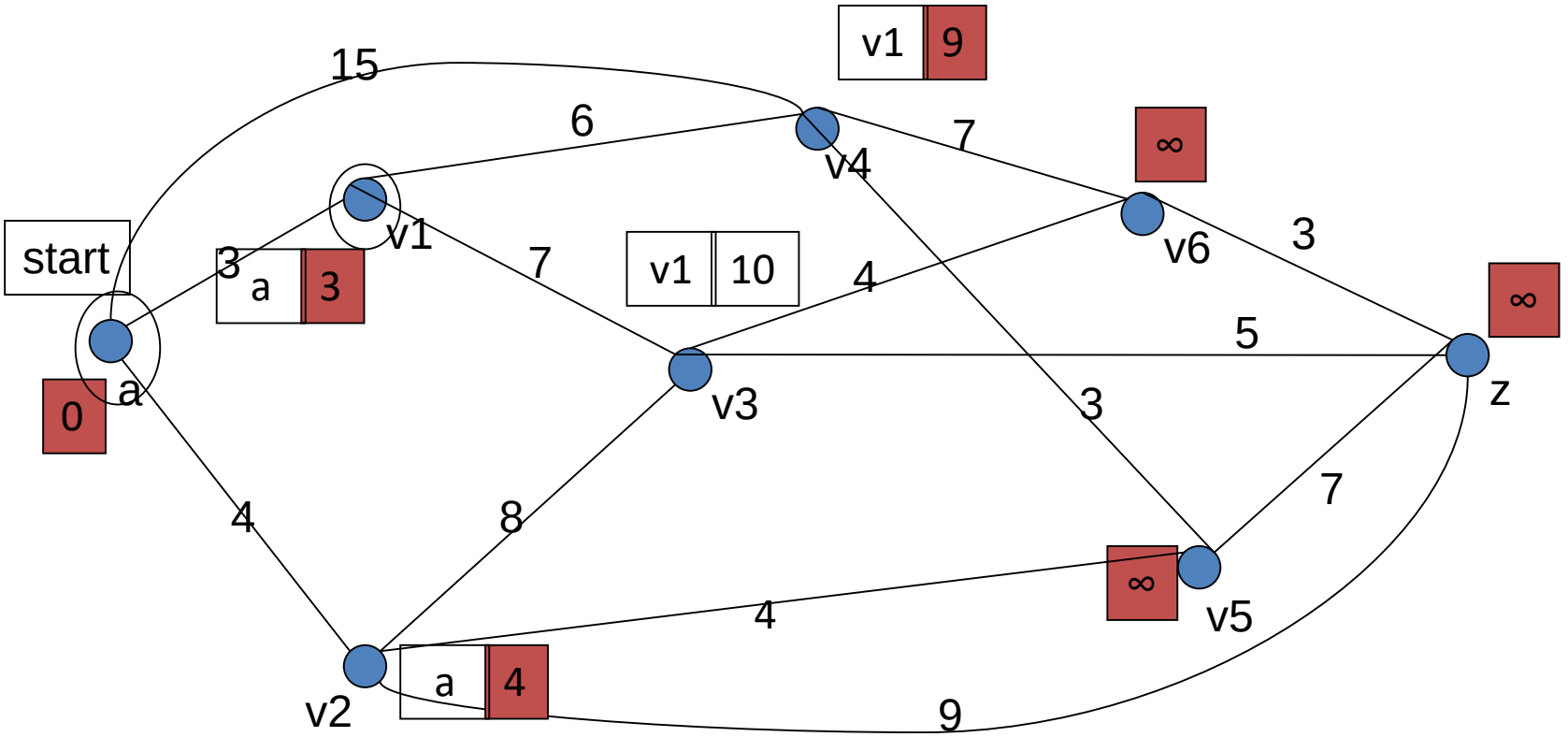
$L(v1)+W[v1,v4]<L(v4)$
$3+6= 9 < 15$
$L(v4)=9$



$S=\{a, v1\}$

$N= \{v2,v3,v4,v5,v6,z\}$

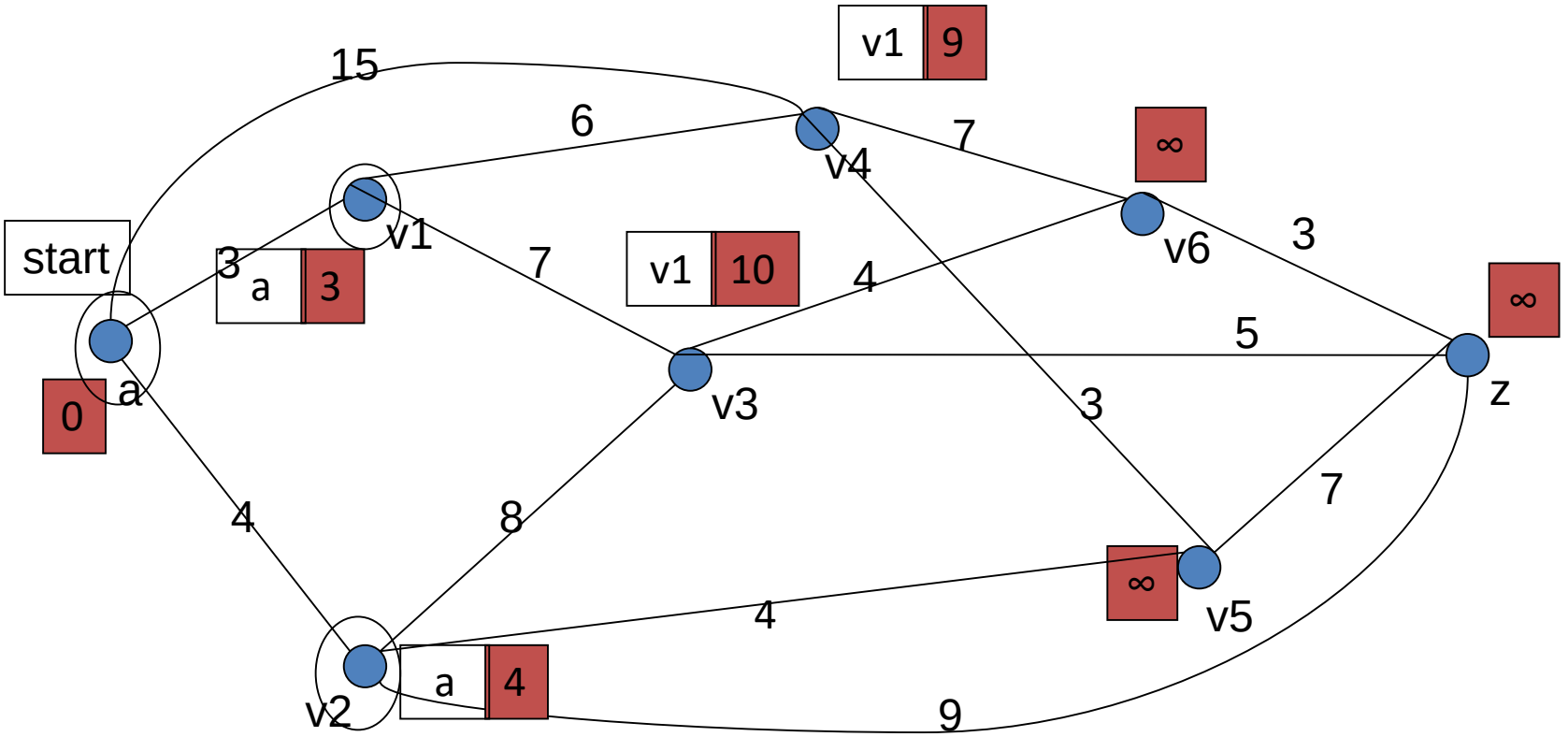
$L(v1)+W[v1,v3]<L(v3)$
$3+7= 10 < \infty$
$L(v4)=10$



$S=\{a, v1\}$

$N= \{v2,v3,v4,v5,v6,z\}$

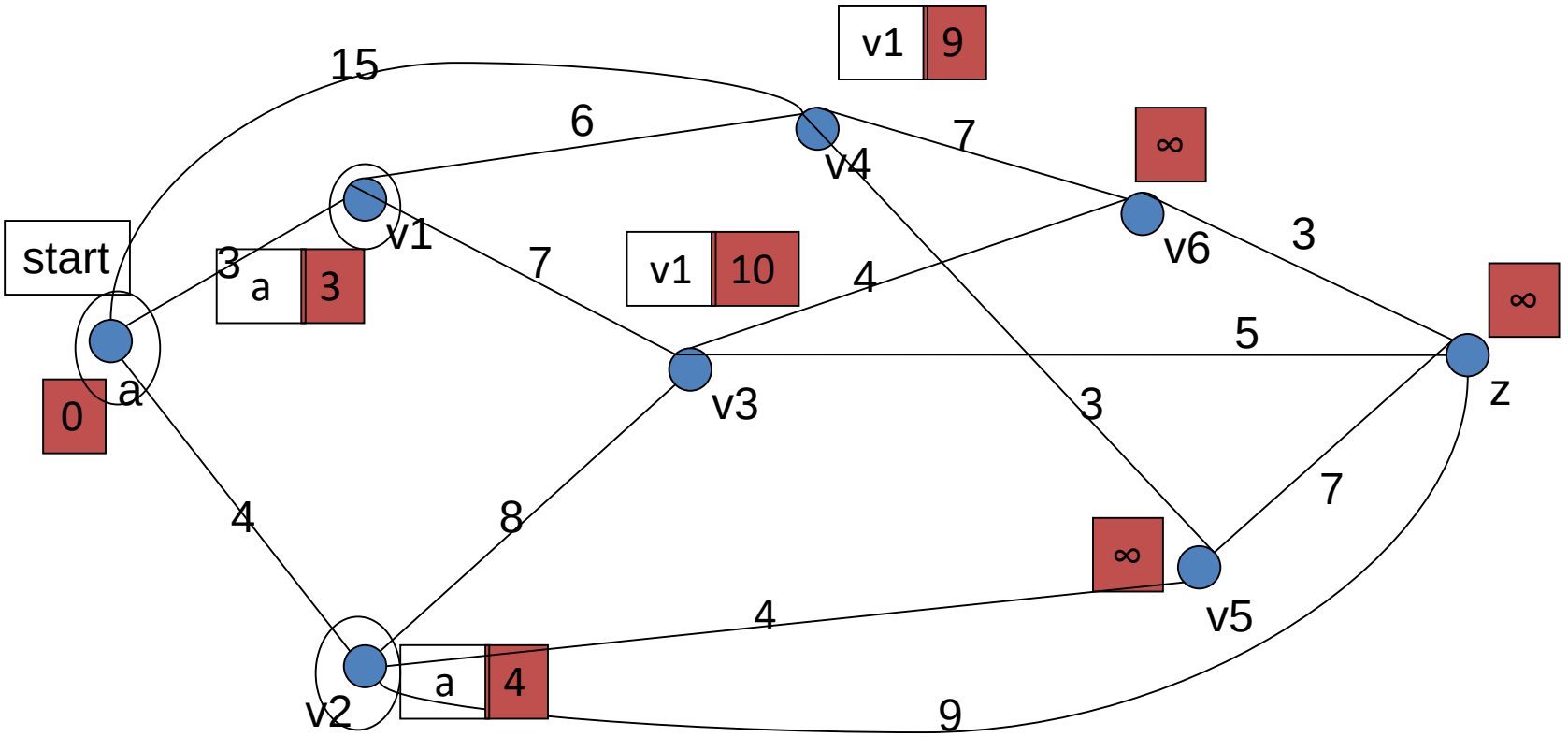
choose *v*2
because
$L(v2)=4 = \min\{L(u) \mid u \in N\}$



$S=\{a, v1, v2\}$

$N= \{v3,v4,v5,v6,z\}$

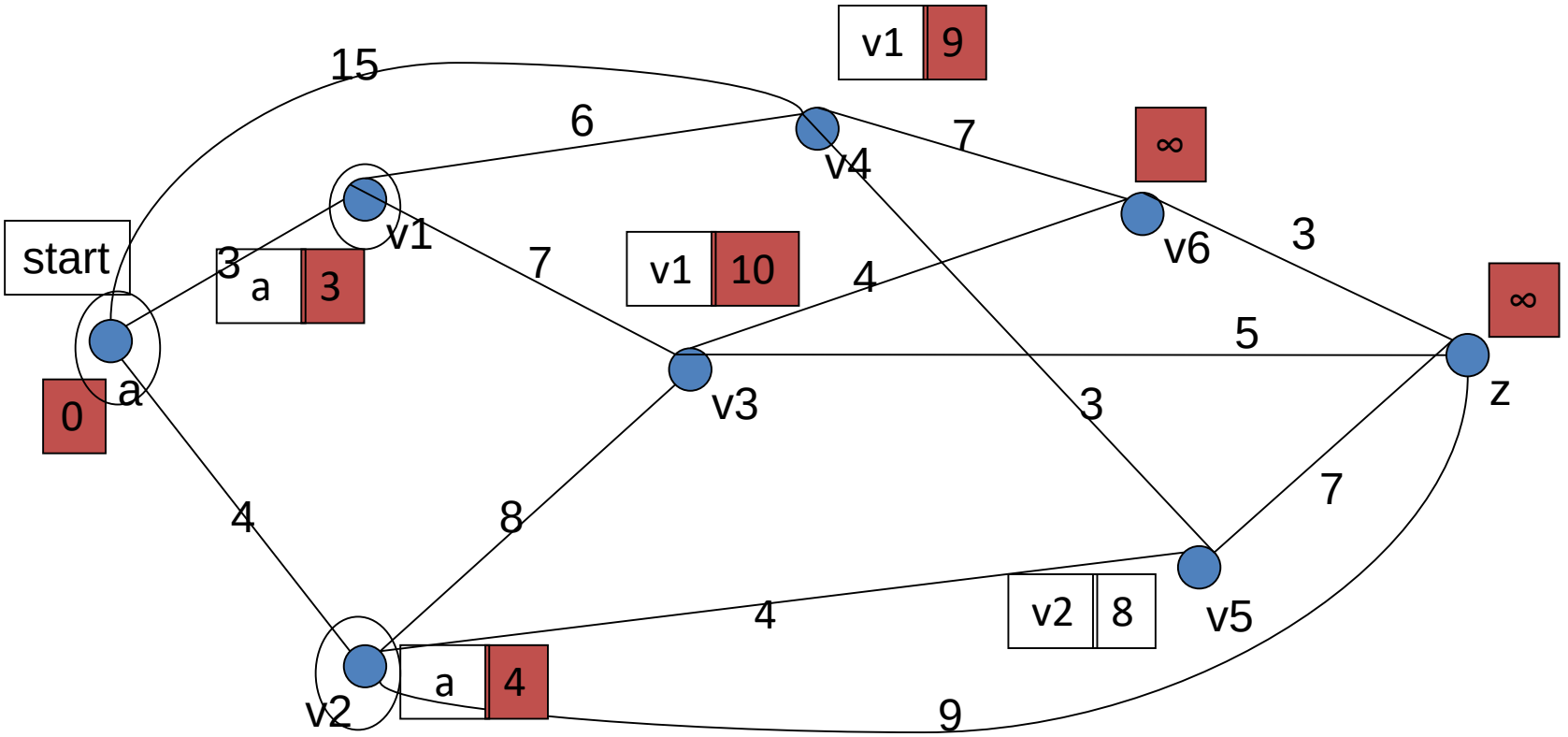
$L(v2)+W[v2,v3]<L(v3)$
$4+8= 12 > 10$
$L(v3)$ remains the same.



$S=\{a, v1, v2\}$

$N=\{v3,v4,v5,v6,z\}$

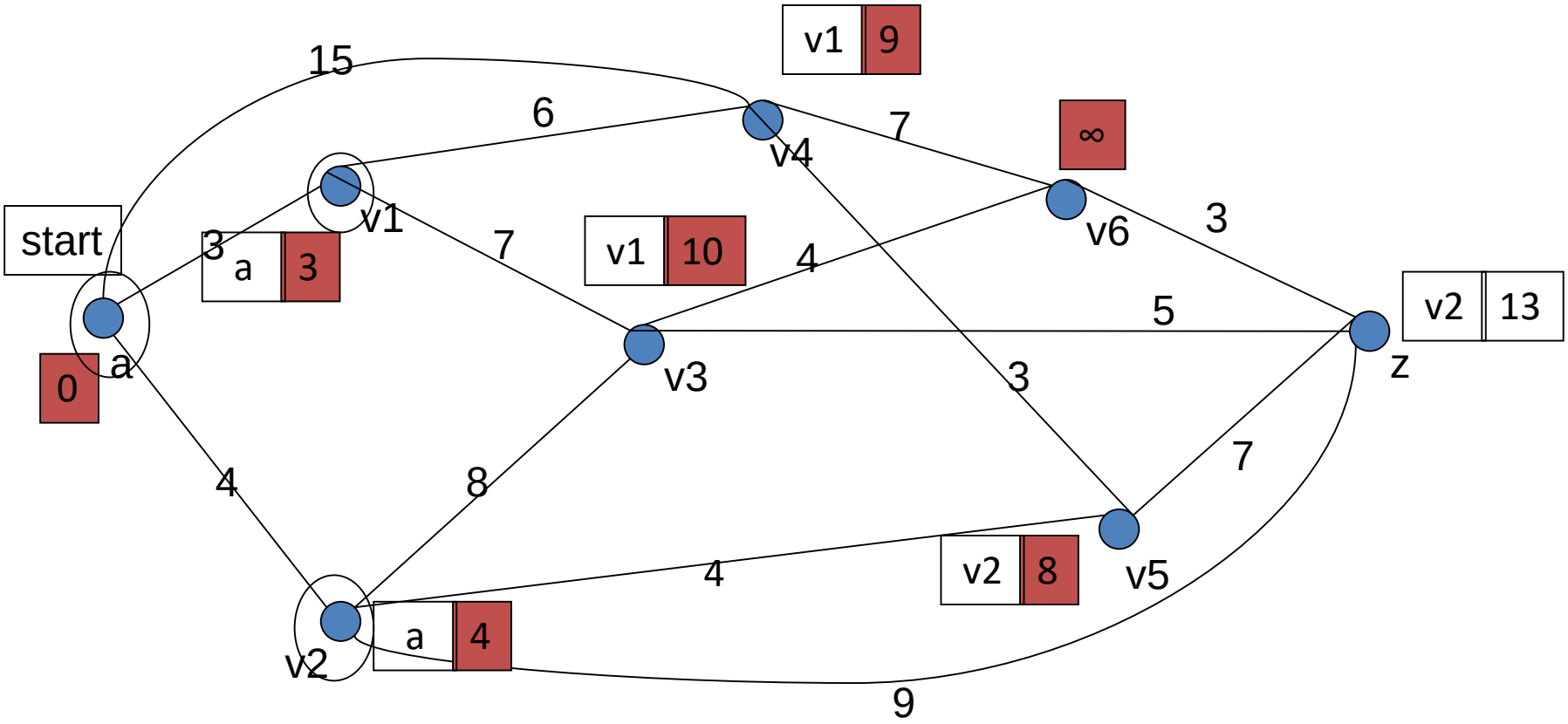
$L(v2)+W[v2,v5]<L(v5)$
$4+4=8<\infty$
$L(v5)=8$



$S=\{a, v1, v2\}$

$N= \{v3,v4,v5,v6,z\}$

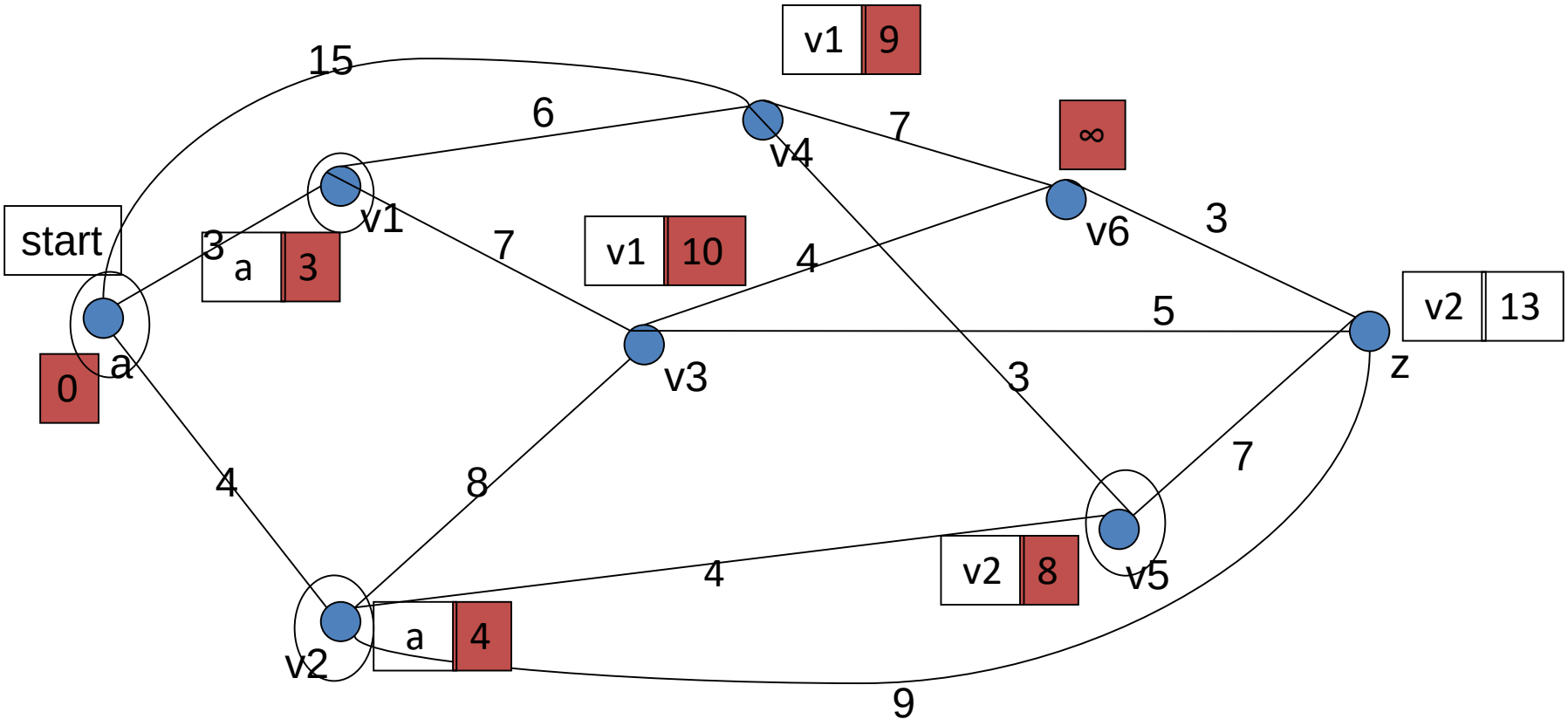
$L(v2)+W[v2, z]<L(z)$
$4+9= 13 < \infty$
$L(z)=13$



$S=\{a, v1, v2\}$

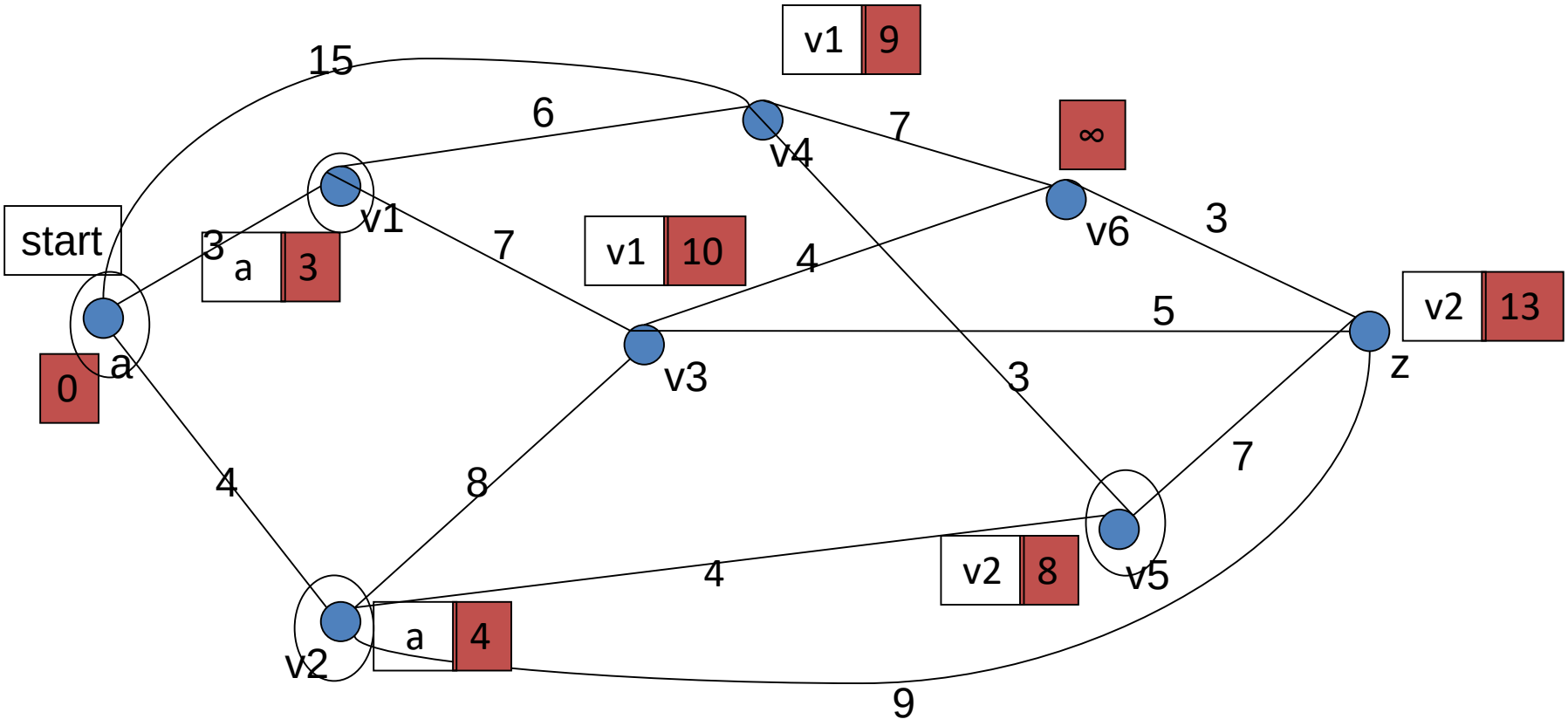
$N=\{v3,v4,v5,v6,z\}$

choose $v5$
because
$L(v5)=8 = \min\{L(u) \mid u \in N\}$



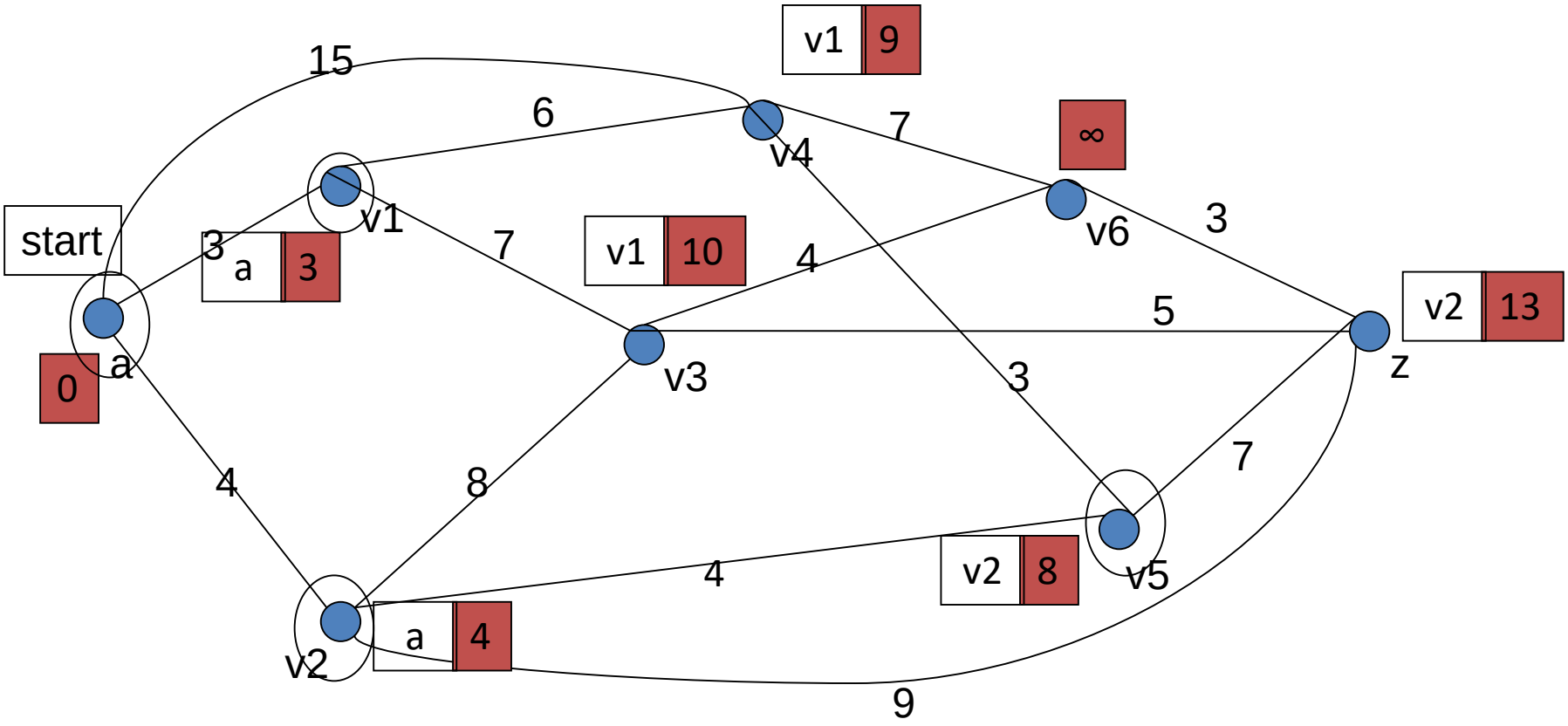
$S=\{a, v1, v2, v5\}$
$N=\{v3, v4, v6, z\}$

$L(v5)+W[v5, v4]<L(v4)$
$8+3= 11 > 9$
$L(v4)$ remains the same



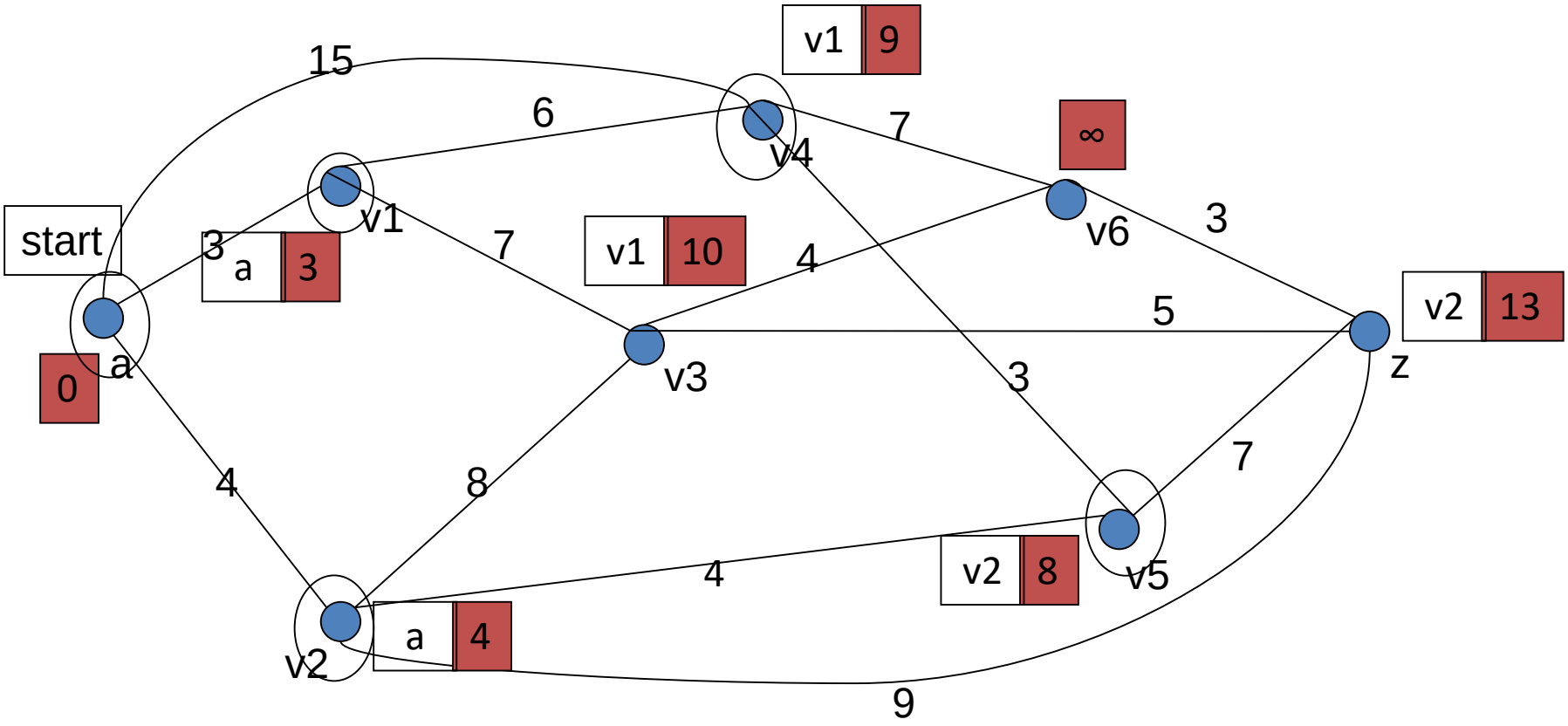
$S=\{a, v1, v2, v5\}$
$N=\{v3, v4, v6, z\}$

$L(v5)+W[v5, z] < L(z)$
$8+7= 15 > 13$
$L(z)$ remains the same



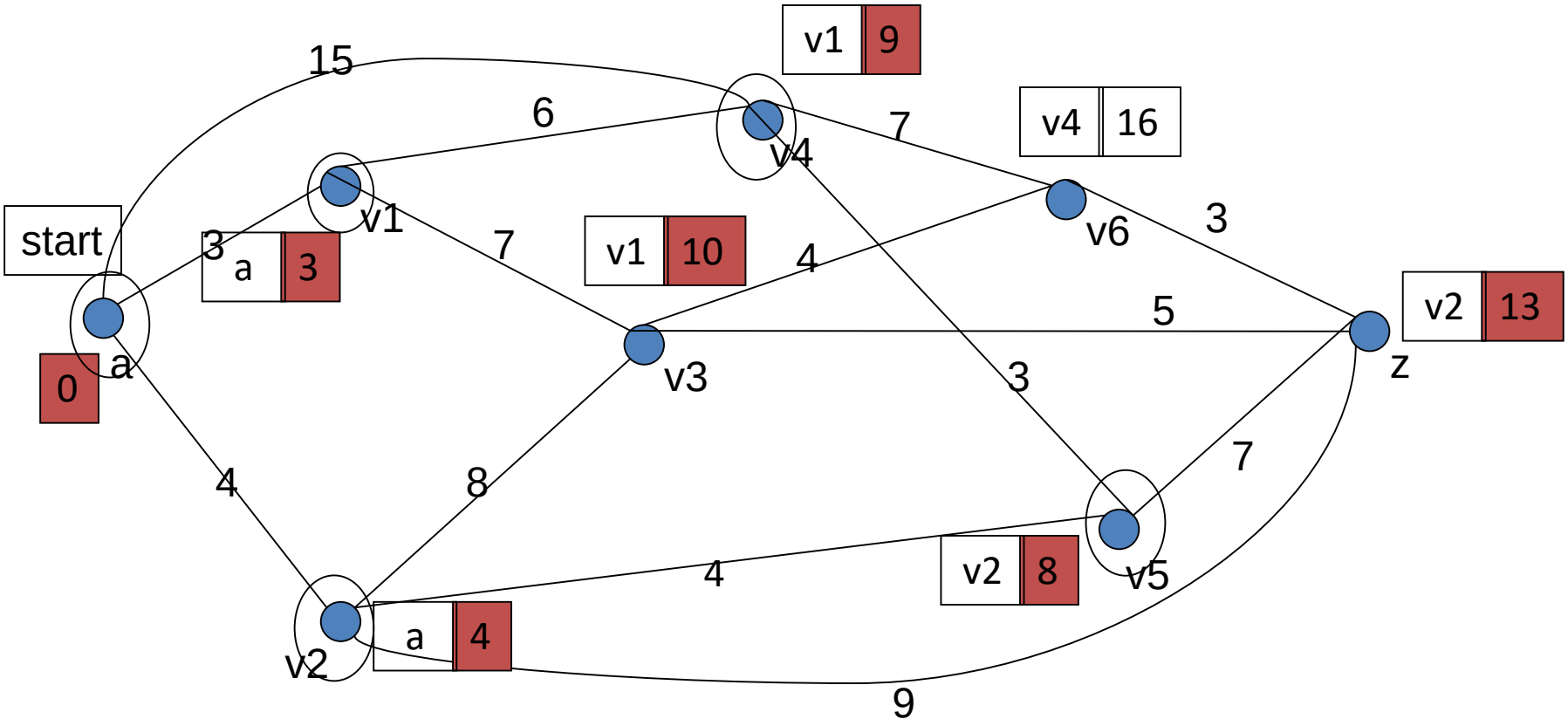
$S=\{a, v1, v2, v5\}$
$N=\{v3, v4, v6, z\}$

choose *v*4
because
$L(v4)=9 = \min\{L(u) \mid u \in N\}$



$S=\{a, v1, v2, v5, v4\}$
$N=\{v3, v6, z\}$

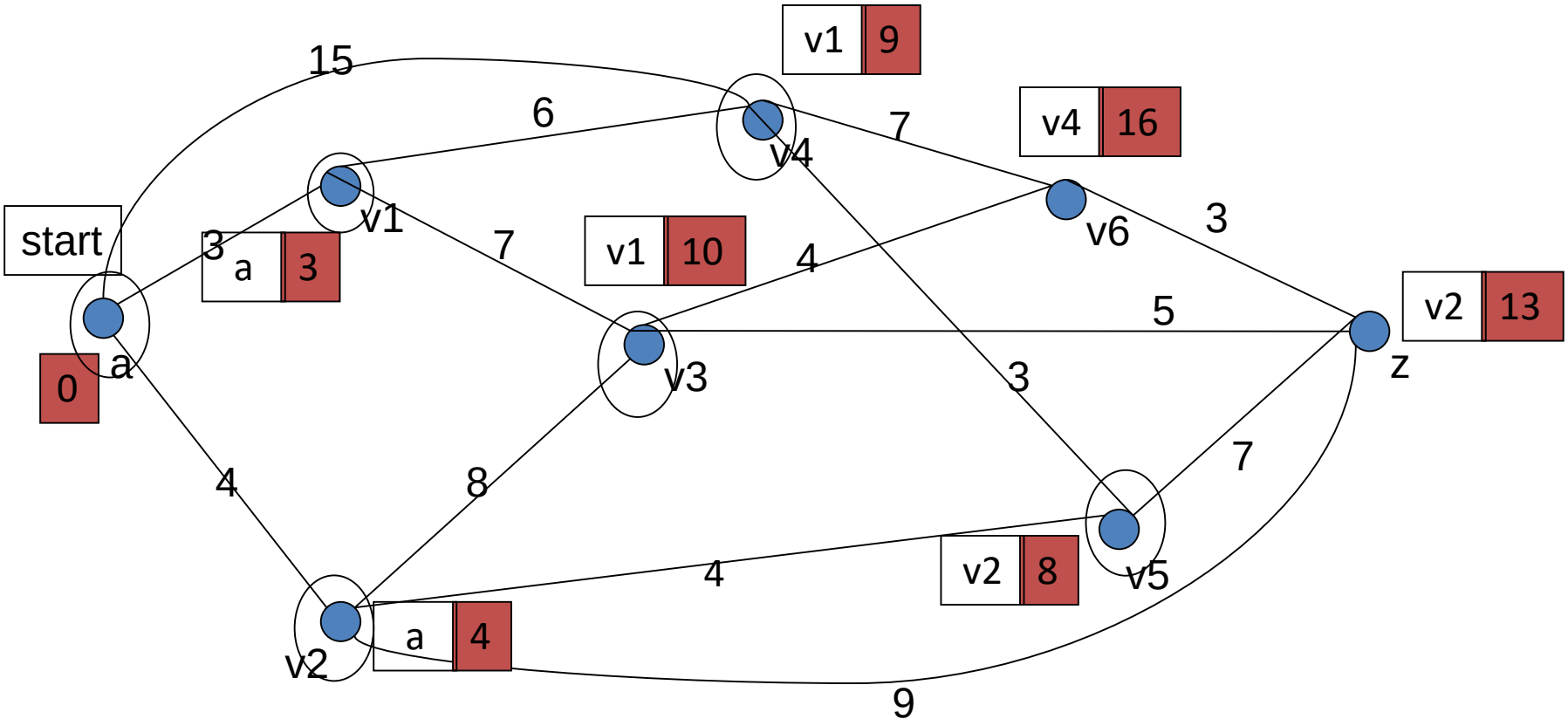
$L(v4)+W[v4, v6]<L(v6)$
$9+7= 16 < \infty$
$L(v6) = 16$



S={a, v1, v2, v5,v4}

N= {v3, v6, z}

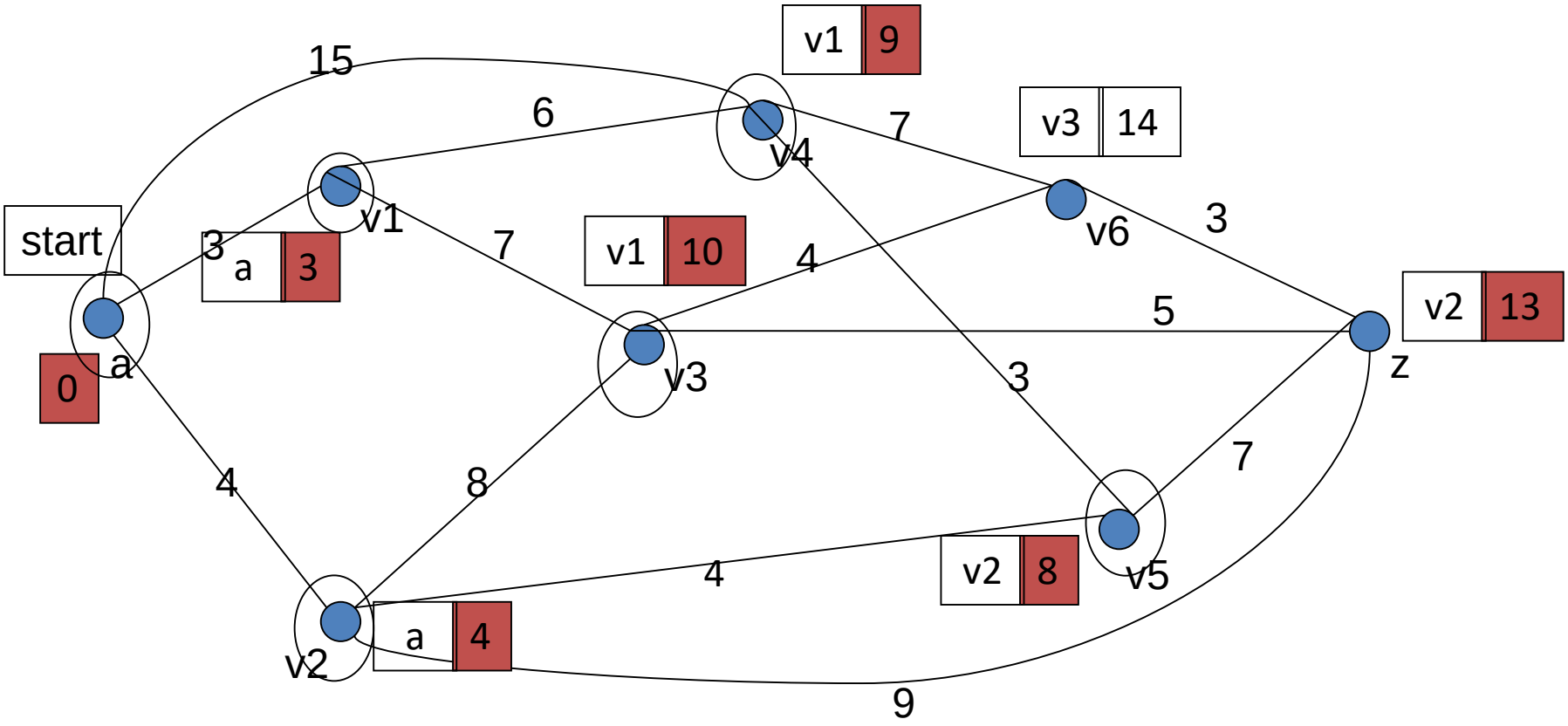
choose $v3$
because
$L(v3)=10 = \min\{L(u) \mid u \in N\}$



S={a, v1, v2, v5,v4, v3}

N= {v6,z}

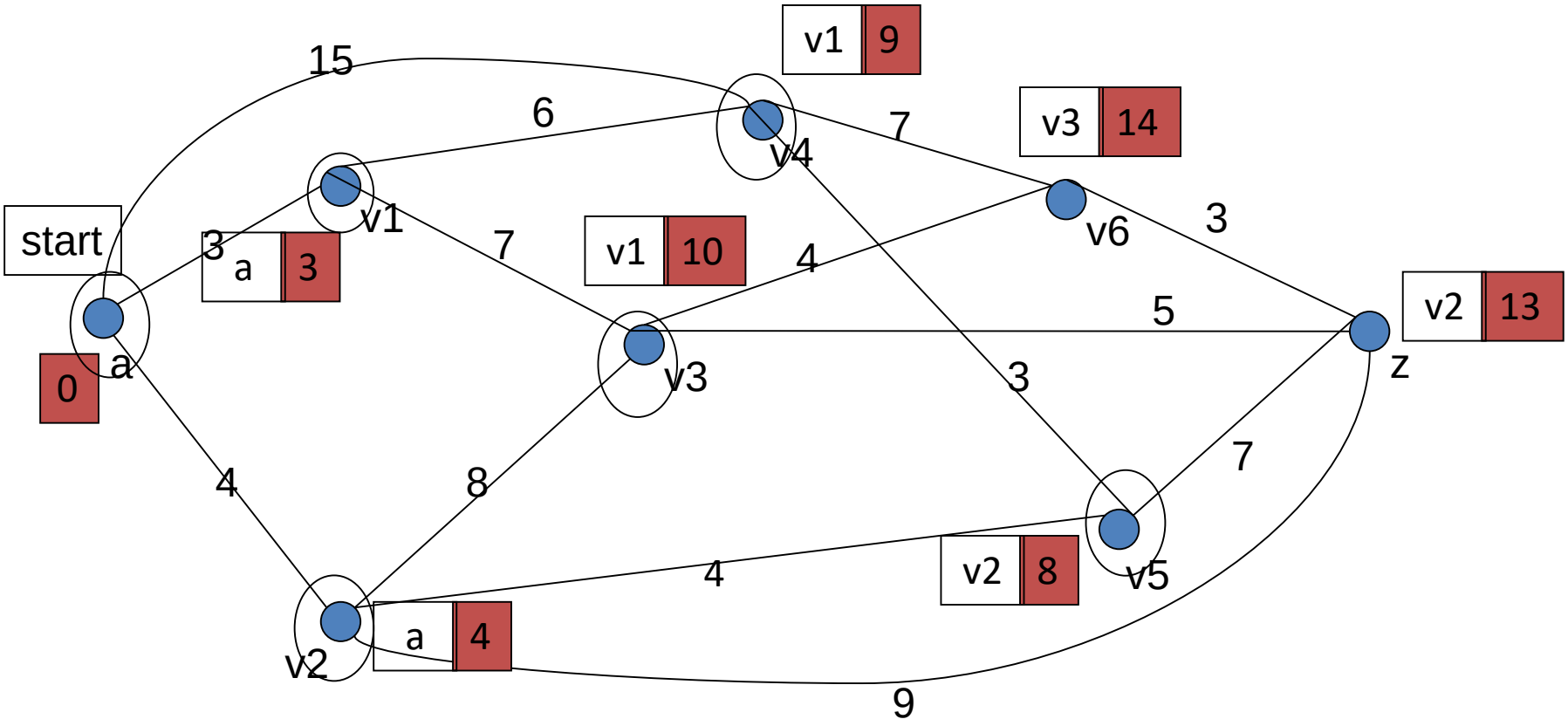
$L(v3)+W[v3, v6]<L(v6)$
10+4= 14 < 16
$L(v6)$ = 14



S={a, v1, v2, v5,v4,v3}

N= {v6,z}

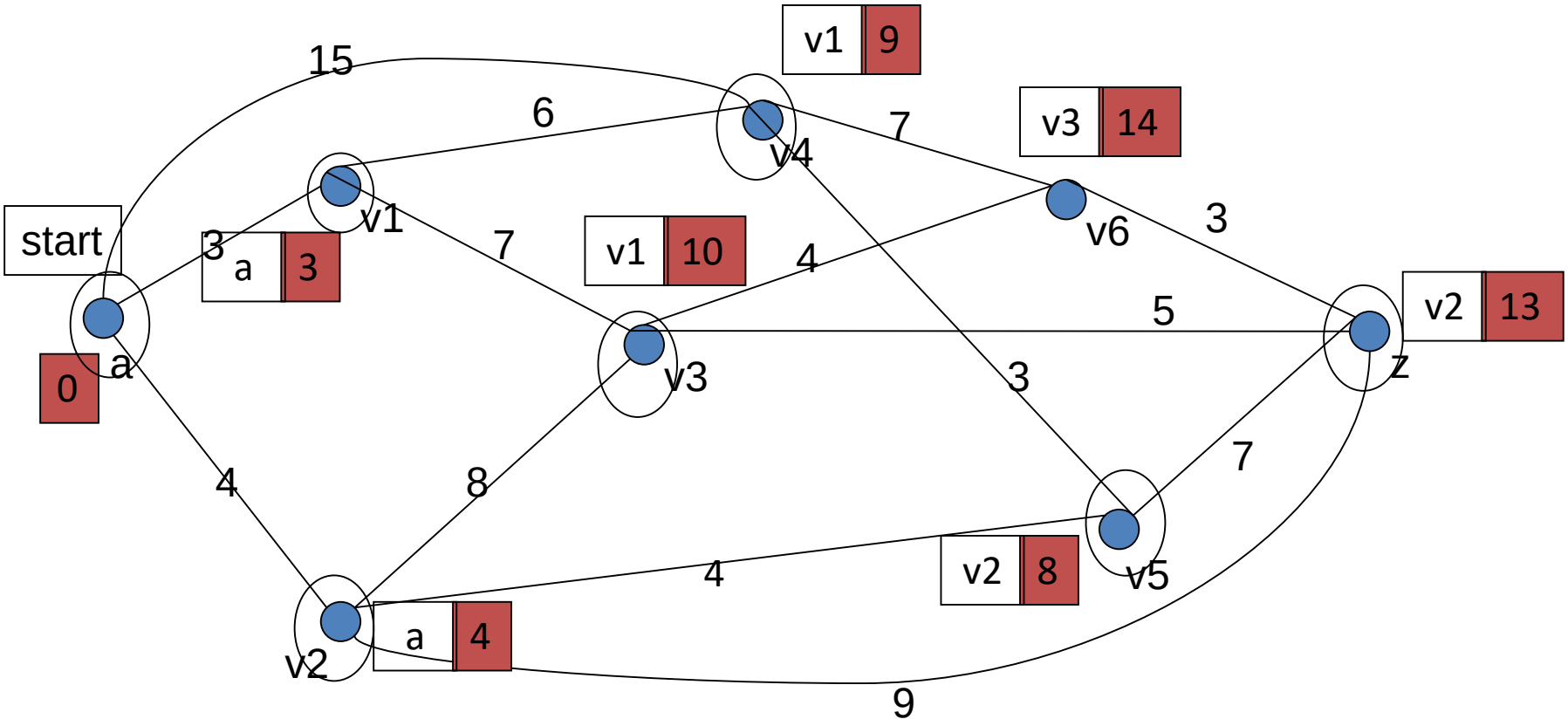
$L(v3)+W[v3, z] < L(z)$
10+5= 15 > 13
$L(z)$ remains the same



S={a, v1, v2, v5,v4,v3}

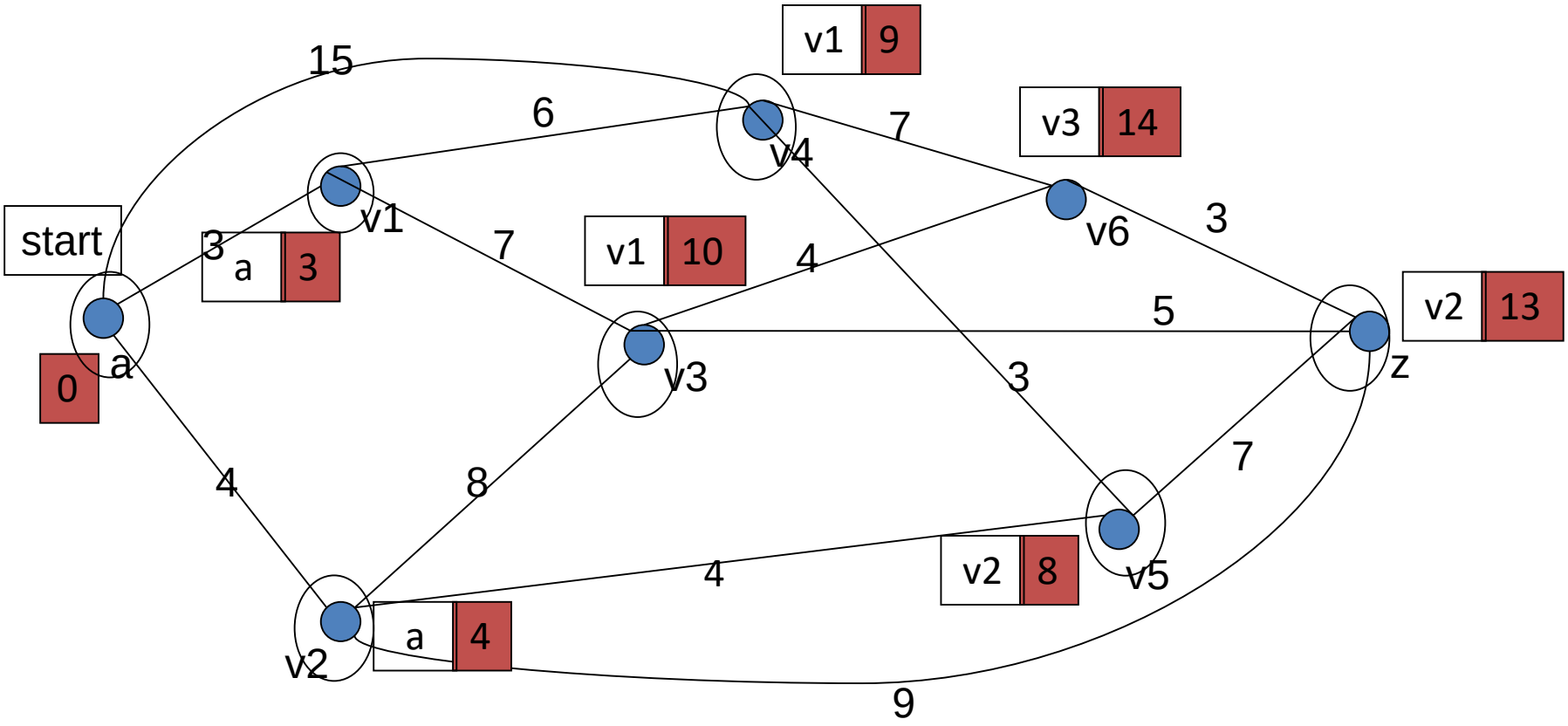
N= {v6,z}

choose z
because
$L(z)=13 = \min\{L(u) \mid u \in N\}$



S={a, v1, v2, v5,v4,v3,z}
N= {v6}

The loop terminates because $z \in S$



Shortest path from *a* to z is a -> v2-> z, with the length 13.

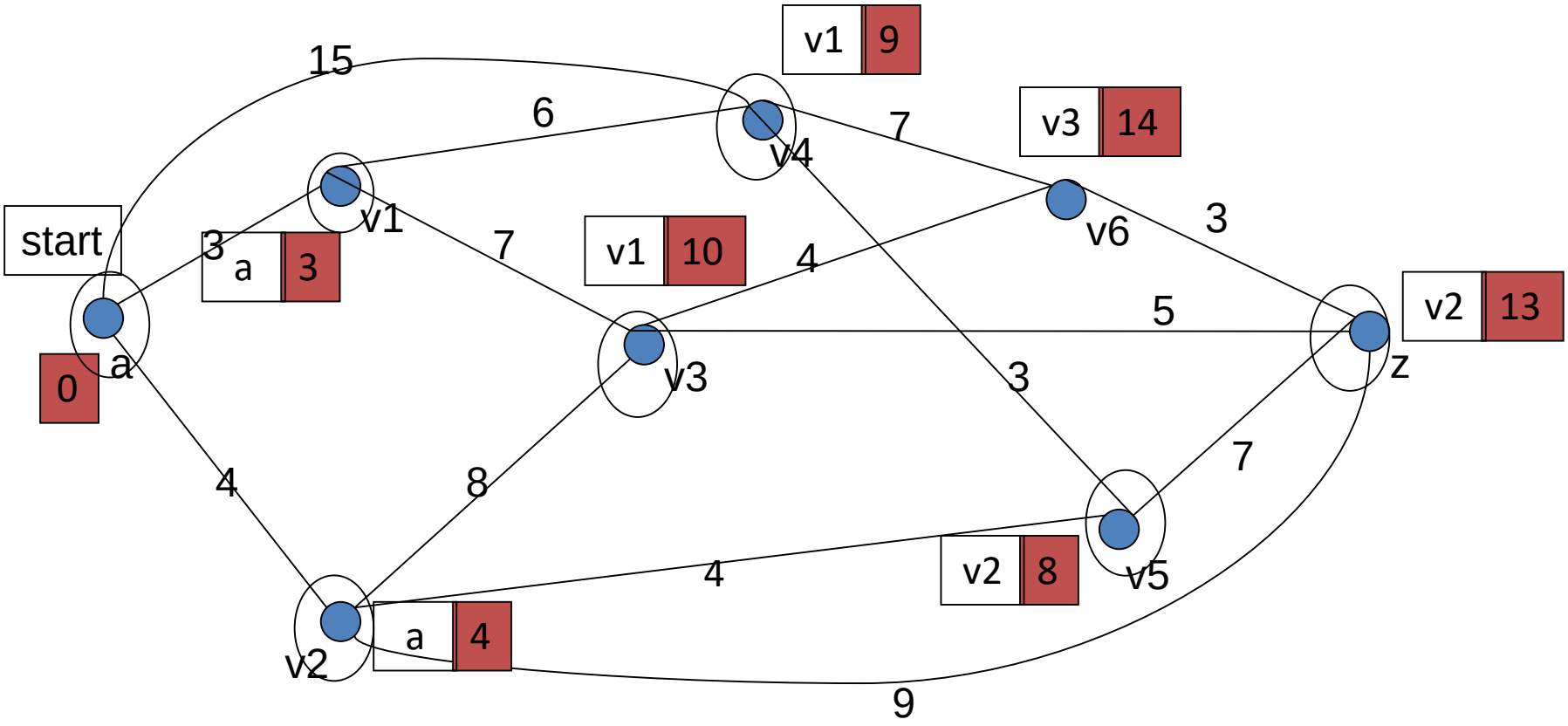
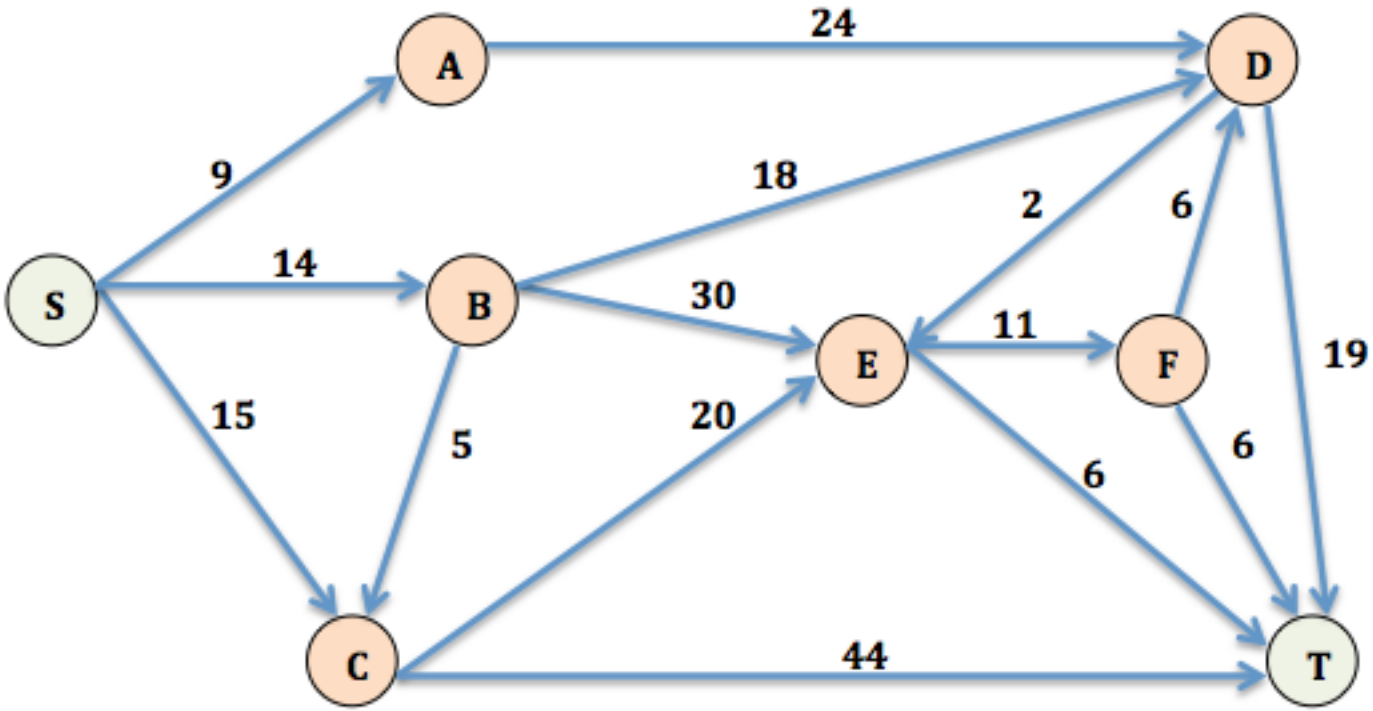


Table – Djikstra Algorithm

No.	S	N	$L(a)$	$L(V_1)$	$L(V_2)$	$L(V_3)$	$L(V_4)$	$L(V_5)$	$L(V_6)$	$L(z)$
0	{ }	{a, V_1, V_2, V_3, V_4, V_5, V_6, z }	0	∞	∞	∞	∞	∞	∞	∞
1	{a}	{V_1, V_2, V_3, V_4, V_5, V_6, z }		3	4	∞	15	∞	∞	∞
2	{a, V_1}	{V_2, V_3, V_4, V_5, V_6, z }		**3**	4	10	9	∞	∞	∞
3	{a, V_1, V_2}	{V_3, V_4, V_5, V_6, z }			**4**	10	9	8	∞	13
4	{a, V_1, V_2, V_5}	{V_3, V_4, V_6, z }				10	9	**8**	∞	13
5	{a, V_1, V_2, V_5, V_4}	{V_6, z }				10	**9**		16	13
6	{a, V_1, V_2, V_5, V_4, V_3,}	{V_6, z }				**10**			14	13
7	{a, V_1, V_2, V_5, V_4, V_3, z}	{V_6 }							14	**13**

Exercise

Q: Given a weighted digraph, find the shortest path from **S** to **T**, using Djikstra Algorithm.



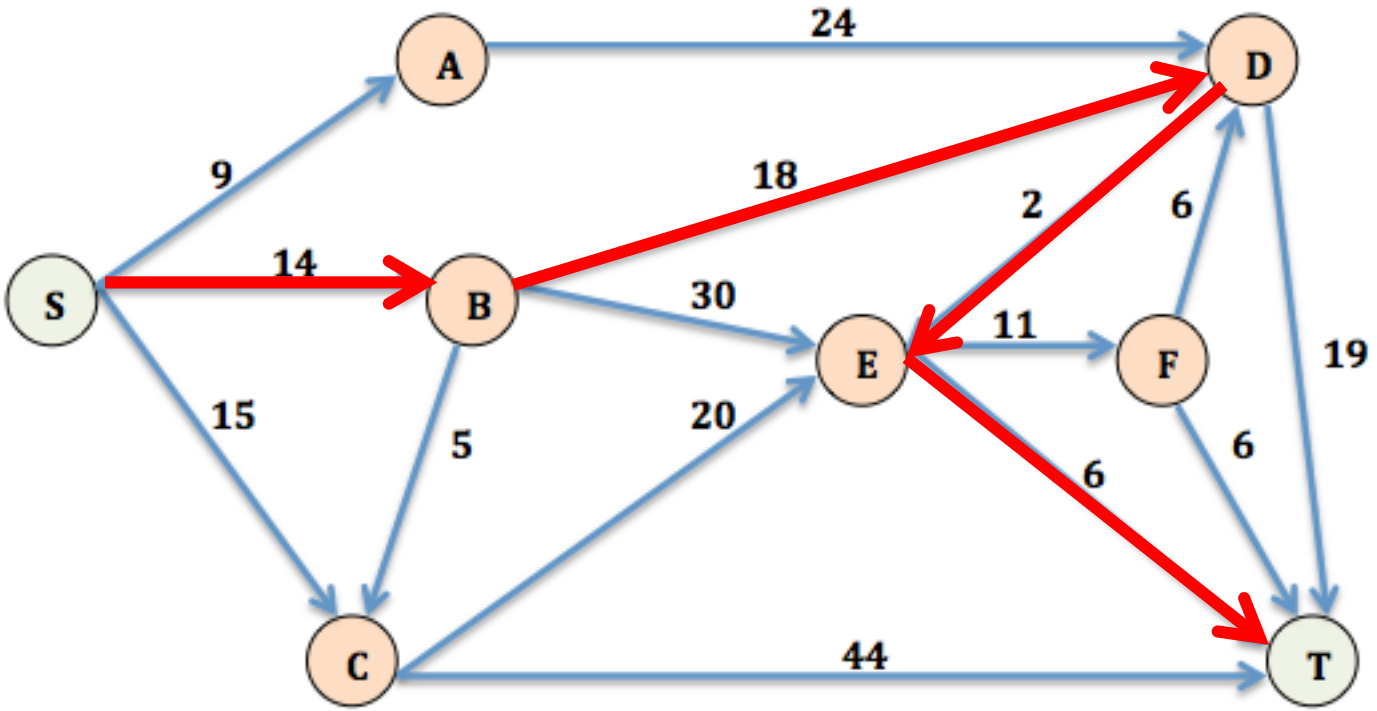
Note: Weights are arbitrary numbers (i.e., not necessarily distances).

Exercise - Solution

i	S	N	L(S)	L(A)	L(B)	L(C)	L(D)	L(E)	L(F)	L(T)
0	Ø	{S,A,B,C,D,E,F,T}	0	∞	∞	∞	∞	∞	∞	∞
1	{S}	{A,B,C,D,E,F,T}	0	9	14	15	∞	∞	∞	∞
2	{S,A}	{B,C,D,E,F,T}	0	9	14	15	33	∞	∞	∞
3	{S,A,B}	{C,D,E,F,T}	0	9	14	15	32	44	∞	∞
4	{S,A,B,C}	{D,E,F,T}	0	9	14	15	32	35	∞	59
5	{S,A,B,C,D}	{E,F,T}	0	9	14	15	32	34	∞	51
6	{S,A,B,C,D,E}	{F,T}	0	9	14	15	32	34	45	40
7	{S,A,B,C,D,E,T}	{F}	0	9	14	15	32	34	45	40

Exercise Solution

The shortest path from **S** to **T**, having weight 40, is S – B – D – E – T



Note: Weights are arbitrary numbers (i.e., not necessarily distances).

Exercise Past Year 2015/2016

The network in Figure 5 gives the distances in miles between pairs of cities A, B, …, and H.

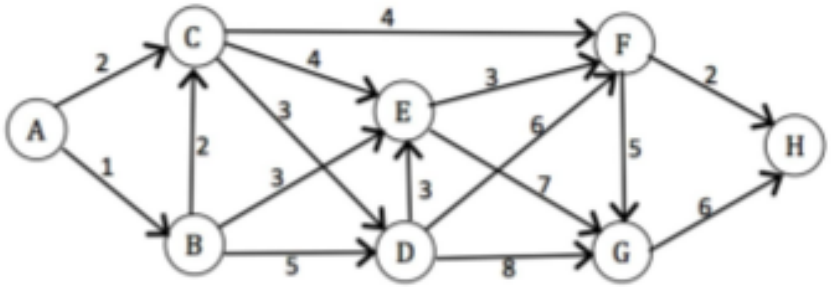


Figure 5

a) Based on Dijkstra's algorithm, complete Table 1 to find the shortest path from city A to city H. (Note: Copy Table 1 into your answer booklet).

(8 marks)

Table 1

Iteration	S	N	L(A)	L(B)	L(C)	L(D)	L(E)	L(F)	L(G)	L(H)
0										
1										
2										
3										
4										
5										
6										
7										

b) State the minimum distance and the shortest path from city 1 to city 8.

(2 marks)

Exercise Solution
Past Year 2015/2016

i	S	N	L(A)	L(B)	L(C)	L(D)	L(E)	L(F)	L(G)	L(H)
0	Ø	{A,B,C,D,E,F,G,H}	0	∞	∞	∞	∞	∞	∞	∞
1	{A}	{B,C,D,E,F,G,H}	0	1	2	∞	∞	∞	∞	∞
2	{A,B}	{C,D,E,F,G,H}	0	1	2	6	4	∞	∞	∞
3	{A,B,C}	{D,E,F,G,H}	0	1	2	5	4	6	∞	∞
4	{A,B,C,E}	{D,F,G,H}	0	1	2	5	4	6	11	∞
5	{A,B,C,E,D}	{F,G,H}	0	1	2	5	4	6	11	∞
6	{A,B,C,E,D,F}	{G,H}	0	1	2	5	4	6	11	8
7	{A,B,C,E,D,F,H}	{G}	0	1	2	5	4	6	11	8

- Minimum distance from city *A* to city *H* is 8
- The shortest path is *A – C – F - H*

SCSI 1013: Discrete Structure

CHAPTER 4 – PART 2

TREE

Introduction

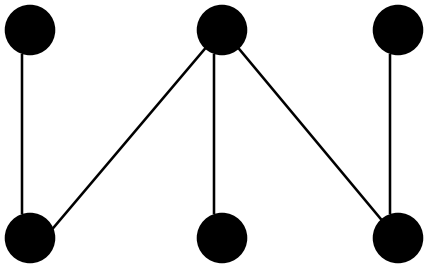
Definition 1. A tree is a connected undirected graph with no simple circuits.

Theorem 1. An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

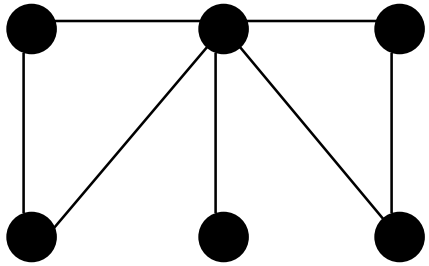
Theorem 2 . A tree with m-vertices has m-1 edges

Which graphs are trees?

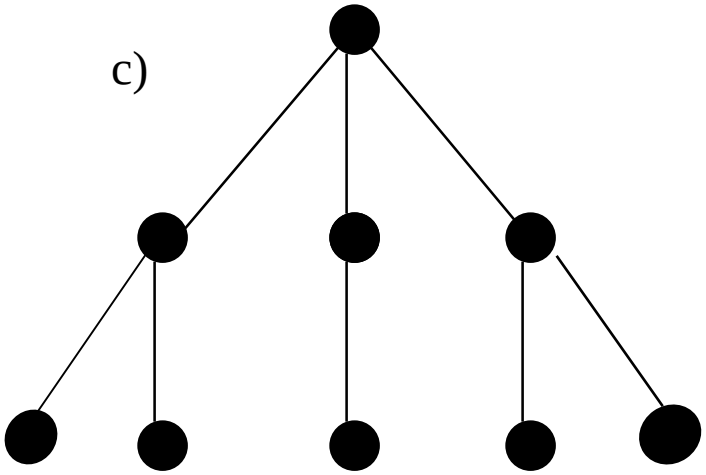
a)



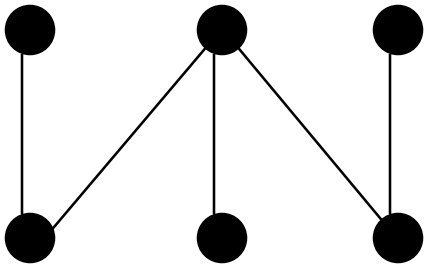
b)



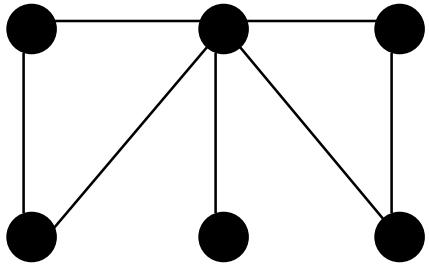
c)



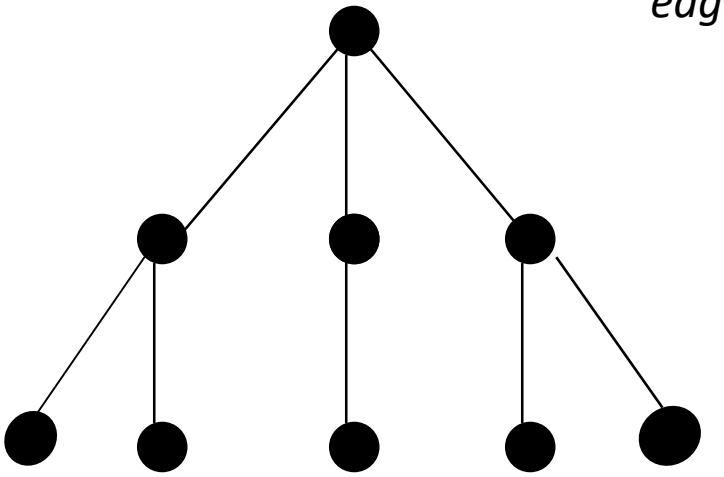
Solution



tree
vertices = 6
edges = 5



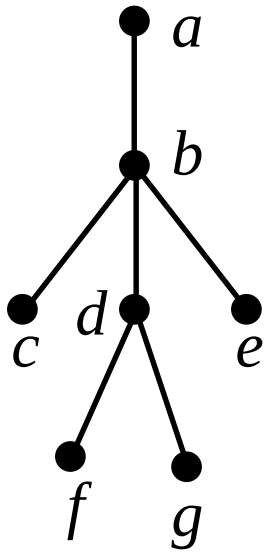
Not a tree
vertices = 6
edges = 7



tree
vertices = 9
edges = 8

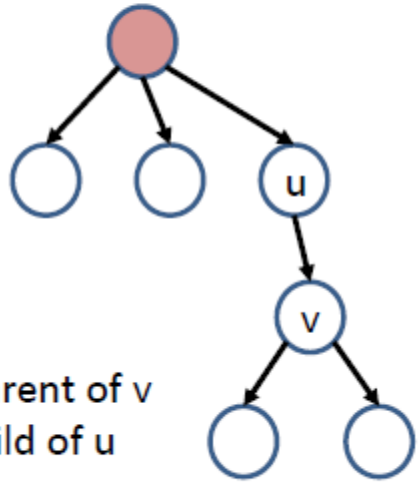
Rooted tree

Definition 2. A **rooted tree** is a tree in which one vertex has been designed as the **root** and every edge is directed away from the root.

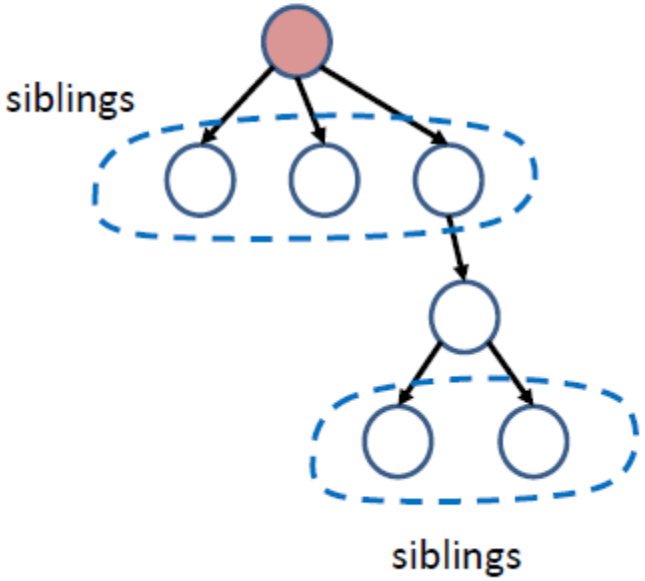


Rooted Tree - Terminologies

- Each edge is from a parent to a child
- Vertices with the same parent are siblings

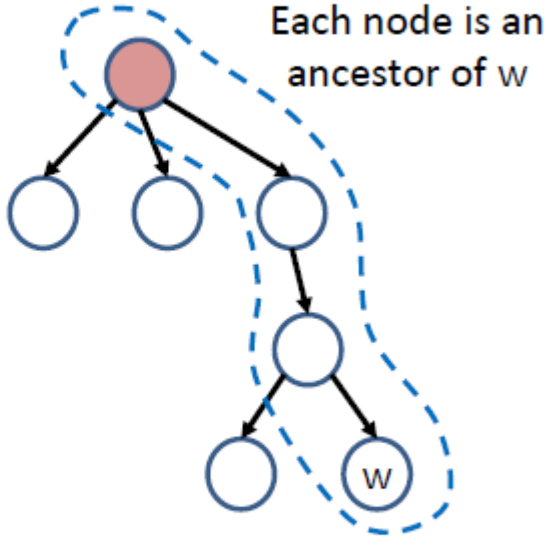


u is the parent of v
v is a child of u



Rooted Tree - Terminologies

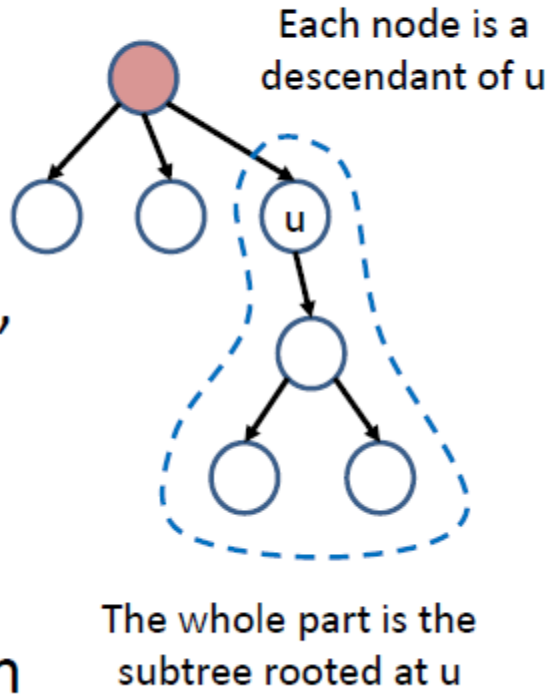
- The ancestors of a vertex w include all the nodes in the path from the root to w
- The proper ancestors of a vertex w are the ancestors of w, but excluding w



The whole part forms a path from root to w

Rooted Tree - Terminologies

- The descendants of a vertex u include all the nodes that have u as its ancestor
- The proper descendants of a vertex u are the descendants of u, but excluding u
- The subtree rooted at u includes all the descendants of u, and all edges that connect between them

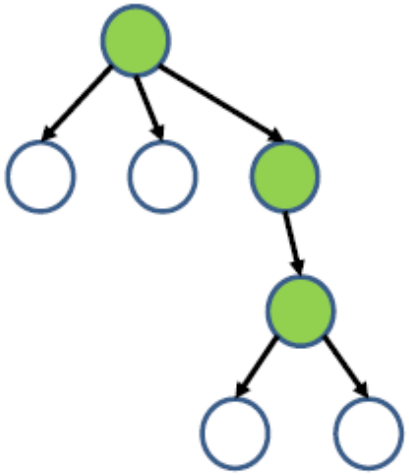


Rooted Tree - Terminologies

- Vertices with no children are called leaves ;

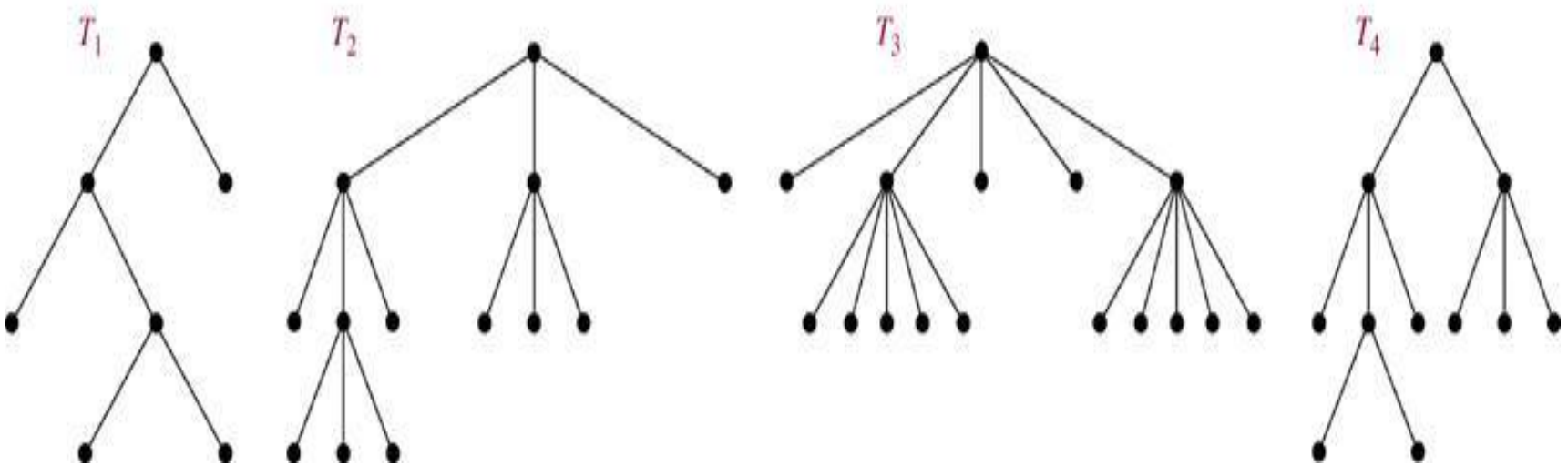
  Otherwise, they are called internal nodes
- If every internal node has no more than m children, the tree is called an m-ary tree
  - Further, if every internal node has exactly m children, the tree is a full m-ary tree

All internal nodes are colored



The tree is ternary (3-ary), but not full

Examples



full binary tree

full 3-ary tree

full 5-ary tree

not full 3-ary tree

Properties of Trees

- Theorem : A tree with n nodes has n-1 edges
- Theorem : A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices.

Cor. A full m-ary tree with n vertices contains $(n-1)/m$ internal vertices, and hence $n - (n-1)/m = ((m-1)n+1)/m$ leaves

Properties of Trees

Theorem – A full **m-ary** tree with

- ***n*** vertices has ***i = (n-1)/m*** internal vertices and ***l = [(m-1)n+1]/m*** leaves
- ***i*** internal vertices has ***n =mi+1*** vertices and ***l = (m-1)i + 1*** leaves
- *l* leaves has *n =(ml-1)/(m-1)* vertices and *i =(l-1)/(m-1)* internal vertices

Example

Ex : Peter starts out a chain mail. Each person receiving the mail is asked to send it to four other people. Some people do this, and some don't

Now, there are 100 people who received the letter but did not send it out

Assuming no one receives more than one mail. How many people have sent the letter ?

Solution

- The chain letter can be represented using **4-ary** tree. The internal vertices correspond to people who sent out the letter, and the leaves correspond to people who did not send it out. Since 100 people did not send out the letter, the number of leaves in this rooted tree is, l=100. The number of people have seen the letter is $n=(4x100-1)/(4-1)=133$. The number of internal vertices is $133-100=33$, people sent the letter.

Exercise

- How many matches are played in a tennis tournament of 27 players

Solution

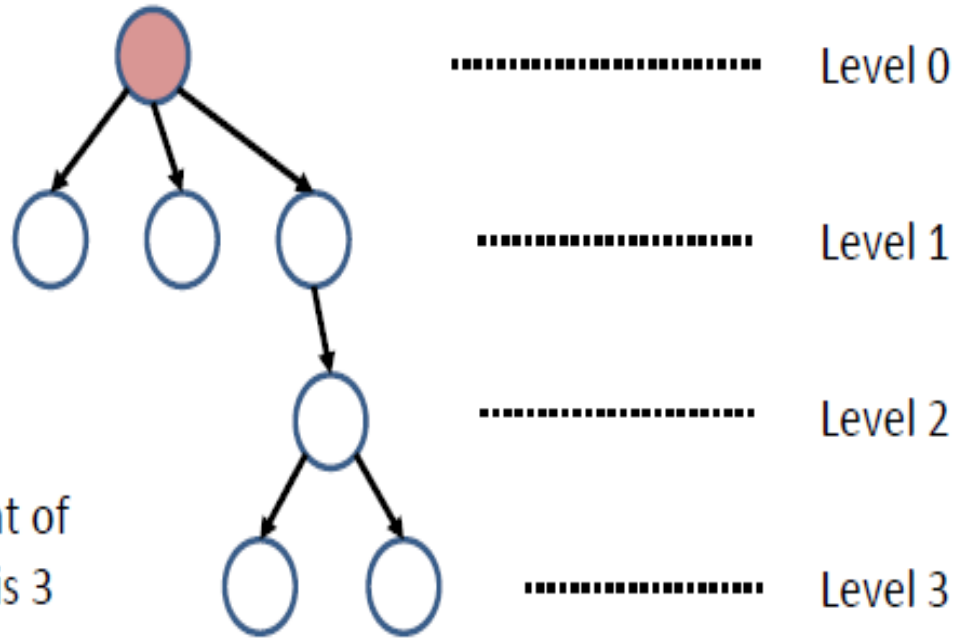
- A leaf for each player, *l*=27
- An internal node for each matches: *m=2*
- Number of matches: $\frac{l-1}{m-1} = \frac{27-1}{2-1} = 26$

Properties of Trees

- The level of a vertex v in a rooted tree is the length of the unique path from the root to this vertex.
  The level of the root is defined to be zero.
  The height of a rooted tree is the maximum of the levels of vertices.

Example

- Ex :



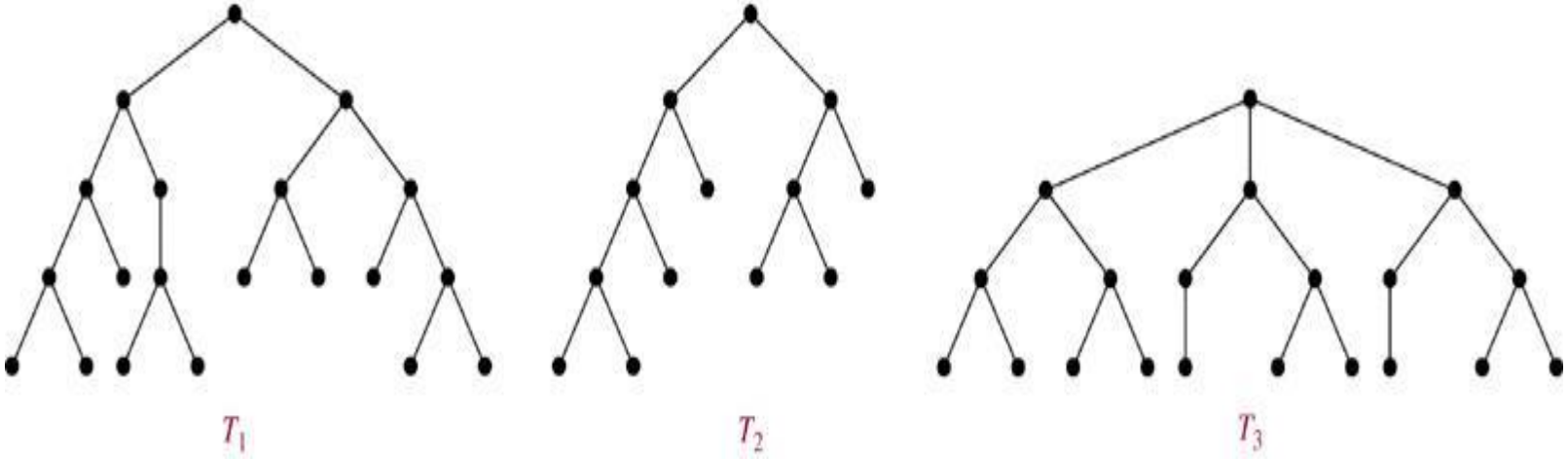
Properties of Trees

- **Definition:** A rooted m-ary tree of height h is balanced if all leaves are at levels h or $h-1$.

- **Theorem.** There are at most m^h leaves in an m-ary tree of height h.

Example

Which of the rooted trees shown below are balanced?



Sol. T_1, T_3

Tree Traversal

- Inorder – left subtree, root, right sub-tree
- Preorder: root, left-subtree, right subtree
- Post-order : left subtree, right sub-tree, root

Preorder Traversal

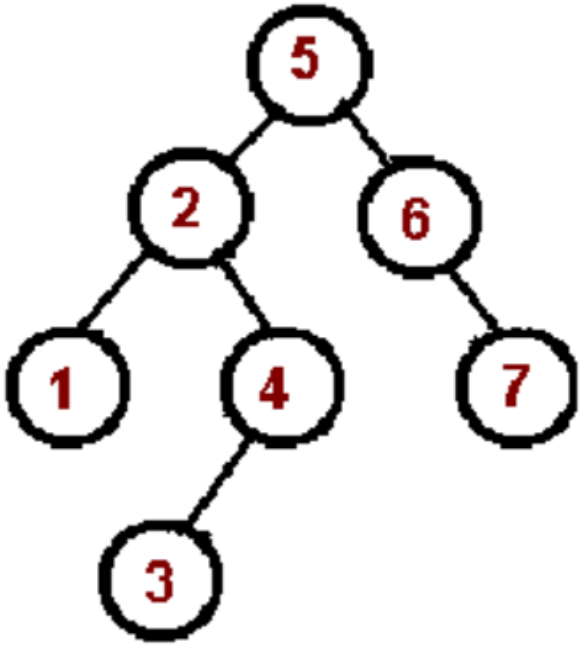
Procedure *preorder*(*T*: ordered rooted tree)
r := root of *T*
list *r*
for each child *c* of *r* from left to right
begin
    T(*c*) := subtree with *c* as its root
    preorder(*T*(*c*))
end

Inorder Traversal

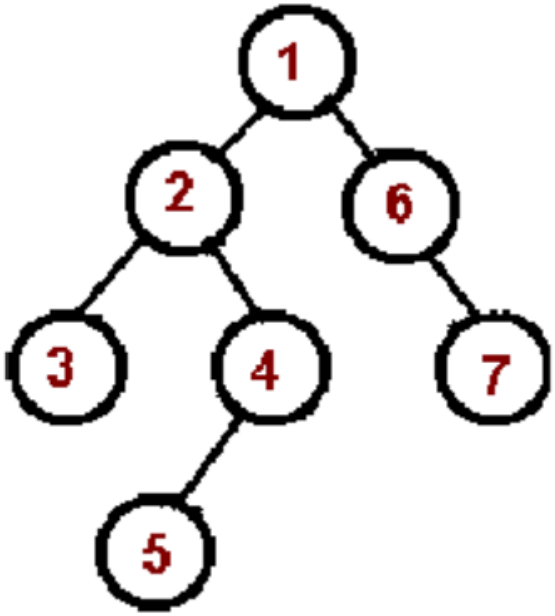
Procedure *inorder*(*T*: ordered rooted tree)
r := root of *T*
If *r* is a leaf **then** list *r*
else
begin
    l := first child of *r* from left to right
    T(*l*) := subtree with *l* as its root
    inorder(*T*(*l*))
    list *r*
    for each child *c* of *r* except for *l* from left to right
        T(*c*) := subtree with *c* as its root
        inorder(*T*(*c*))
end

Postorder Traversal

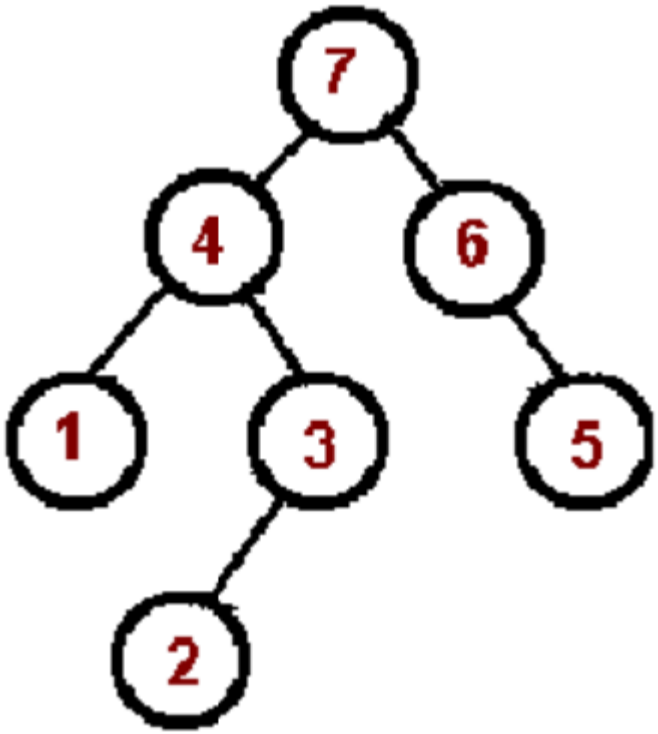
Procedure *postorder*(*T*: ordered rooted tree)
r := root of *T*
for each child *c* of *r* from left to right
begin
    T(*c*) := subtree with *c* as its root
    postorder(*T*(*c*))
end
list *r*



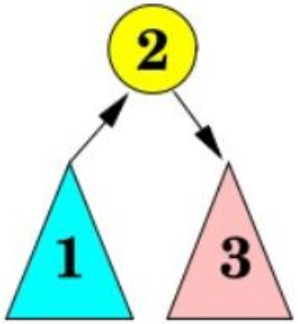
inorder



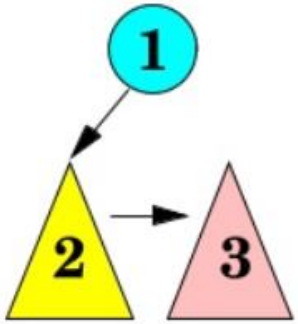
preorder



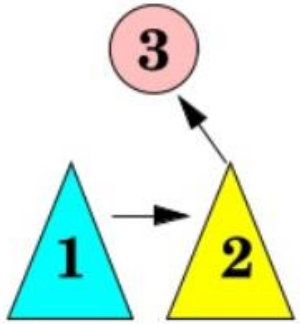
postorder



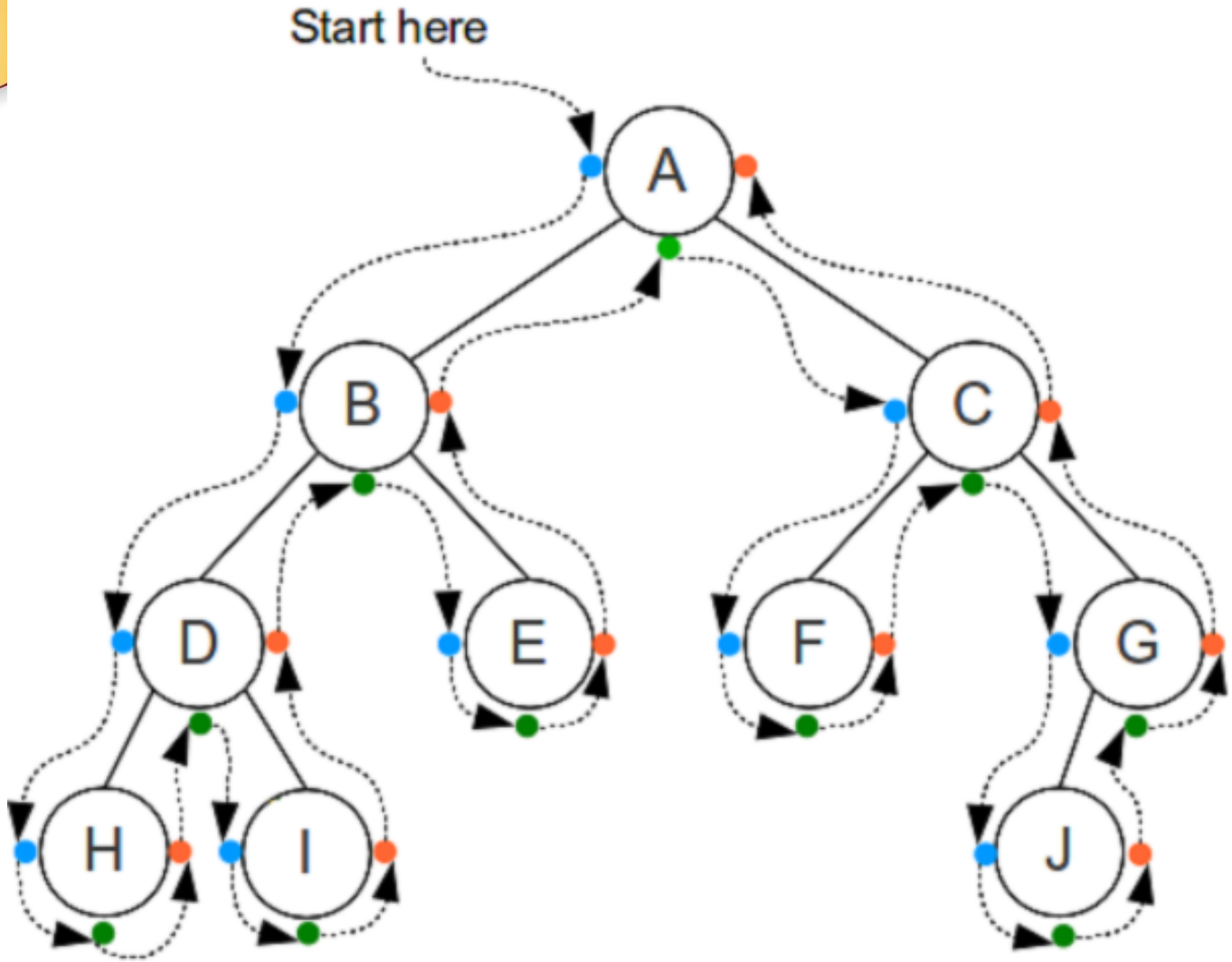
inorder



preorder



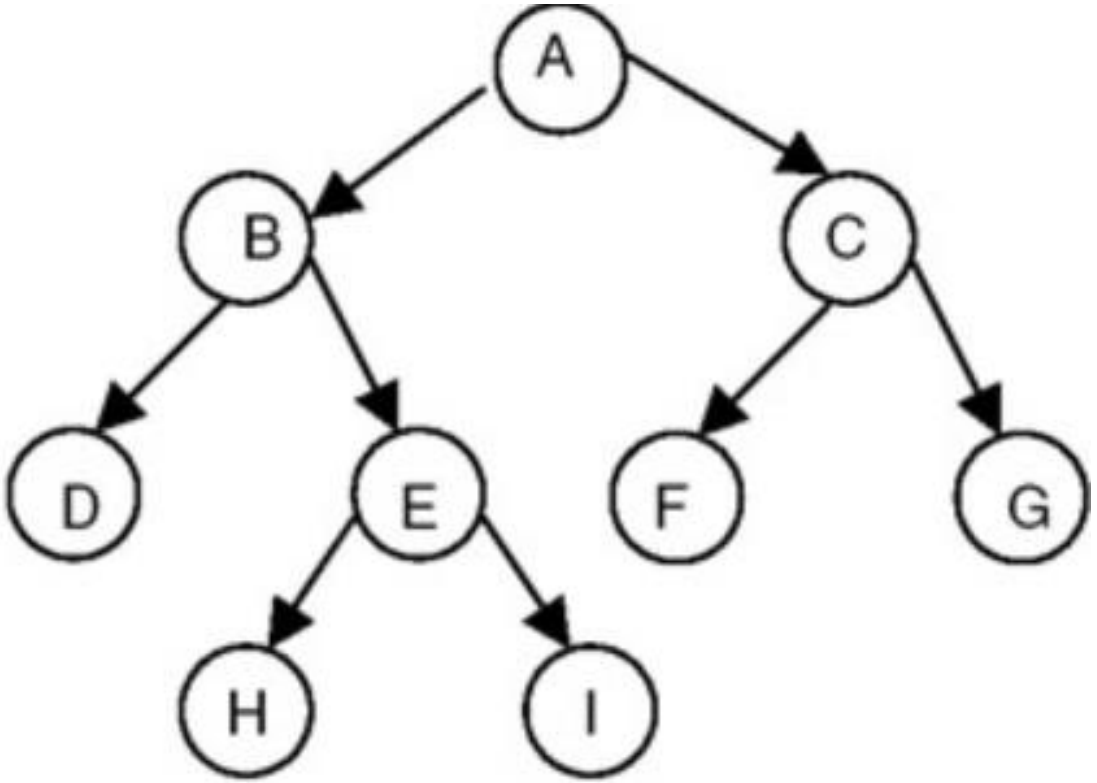
postorder



Pre-Order	ABDHIECFGJ
In-Order	HDIBEAFCJG
Post-Order	HIDEBFJGCA

Exercise

Give the inorder, preorder, and postorder traversals for the following tree.



Solution

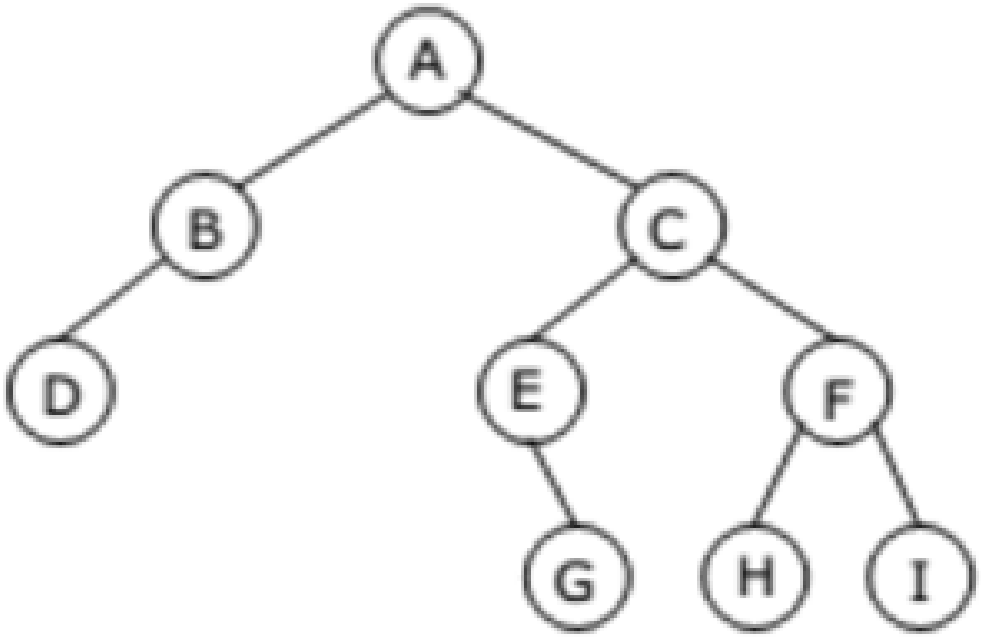
Inorder : DBHEIAFCG

Preorder : ABDEHICFG

Postorder : DHIEBFGCA

Exercise

Trace the inorder, preorder, and postorder traversals for the following tree.

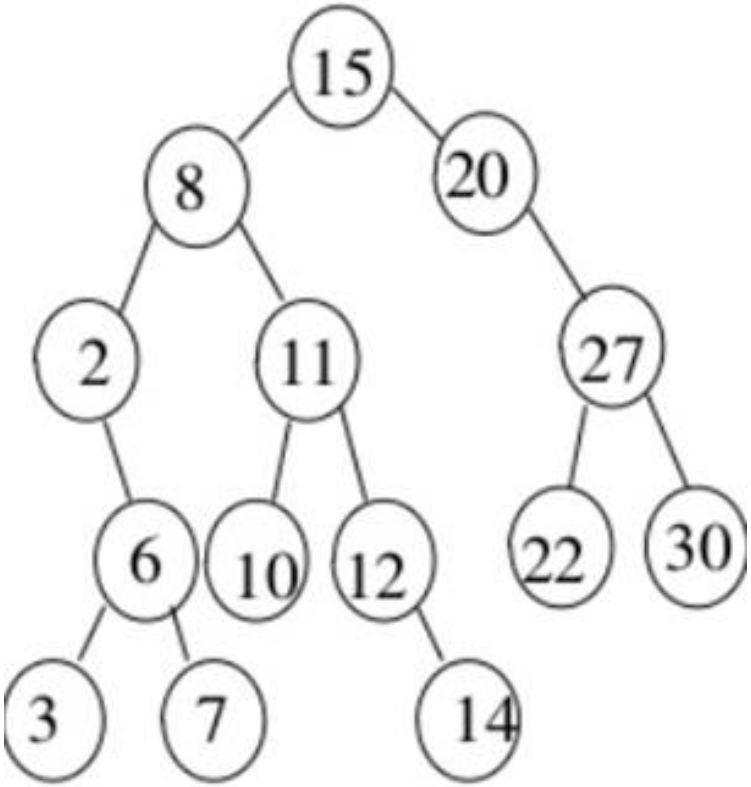


Solution

- Preorder traversal yields:
  A, B, D, C, E, G, F, H, I
- Postorder traversal yields:
  D, B, G, E, H, I, F, C, A
- Inorder traversal yields:
  D, B, A, E, G, C, H, F, I

Exercise

Find the inorder, preorder, and postorder traversals for the following tree.



Solution

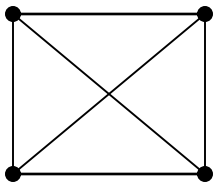
Preorder: 15 8 2 6 3 7
11 10 12 14 20 27 22 30

Inorder: 2 3 6 7 8 10 11
12 14 15 20 22 27 30

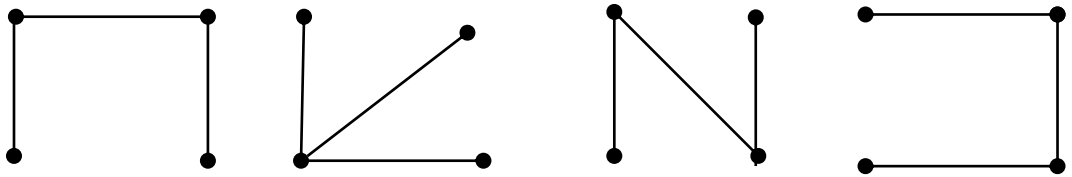
Postorder: 3 7 6 2 10 14
12 11 8 22 30 27 20 15

Spanning Trees

- A spanning tree is a simple graph that is a subgraph of ***G*** and contains every vertex of ***G*** and is a tree.



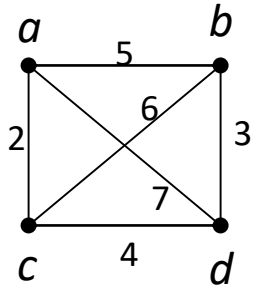
A connected undirected graph

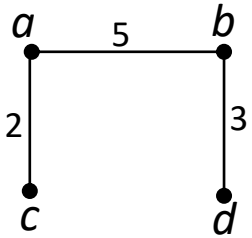


Four spanning trees of the graph

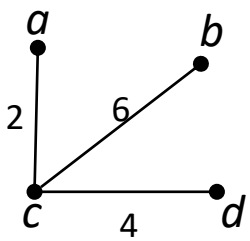
Minimum Spanning Tree (MST)

- A Minimum Spanning Tree is a spanning tree on a weighted graph that has minimum total weight.
- Example

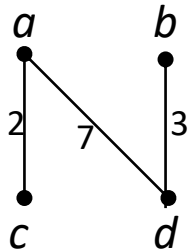




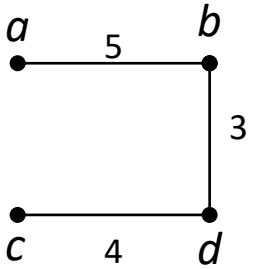
$T_1 = 10$



$T_2 = 12$



$T_3 = 12$



$T_4 = 12$

Application of MST: an example

- In the design of electronic circuitry, it is often necessary to make a set of pins electrically equivalent by wiring them together.
- Running cable TV to a set of houses. What's the least amount of cable needed to still connect all the houses?

Finding MST

- Kruskal's algorithm: start with no nodes or edges in the spanning tree and repeatedly add the cheapest edge that does not create a cycle

Kruskal algorithm

Procedure Kruskal (*G*: weighted connected undirected graph with *n* vertices)

T:= empty graph

for i := 1 to *n*-1

begin

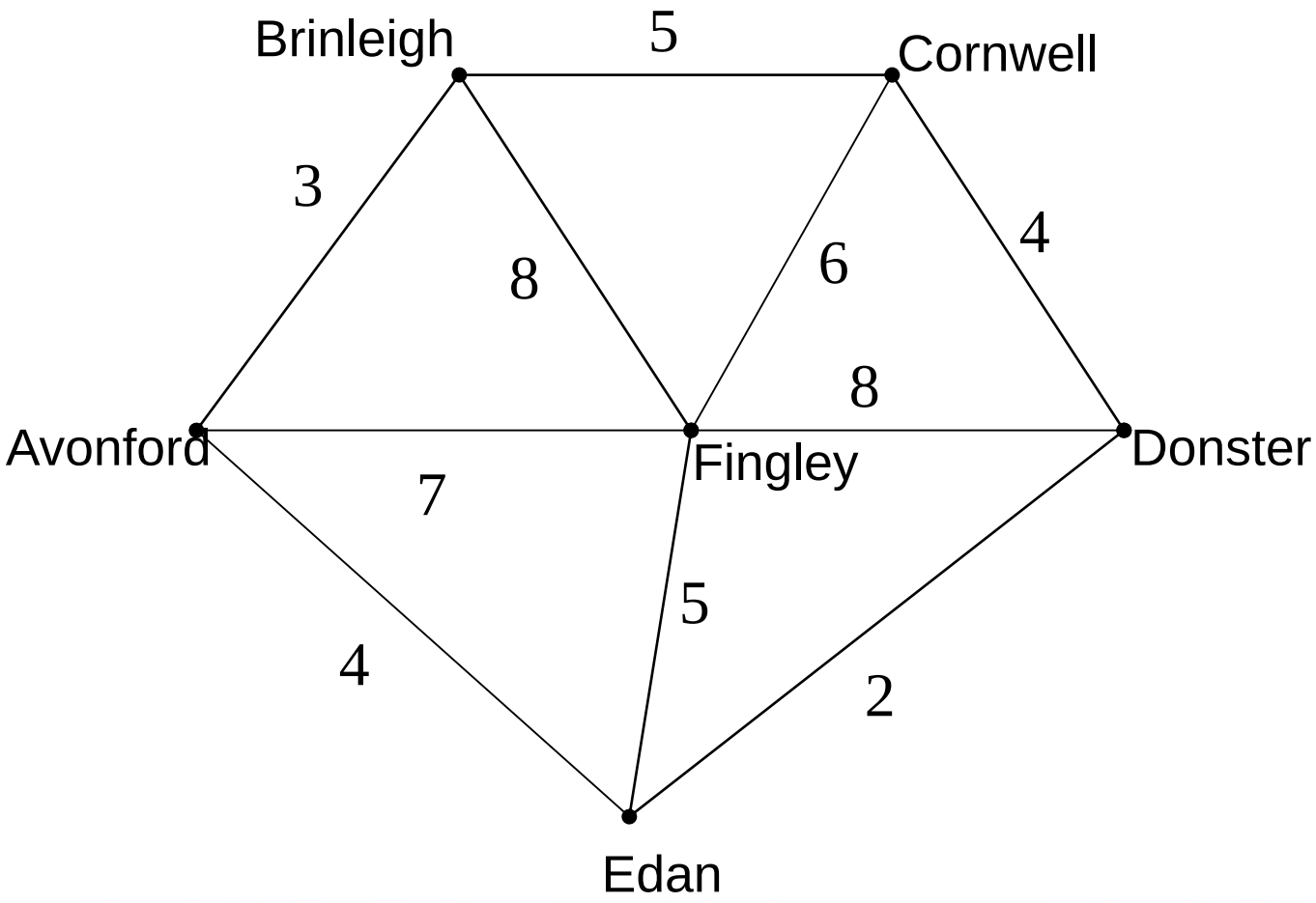
e:= any edge in *G* with smallest weight that does not form a simple circuit when added to *T*

T:= *T* with e added

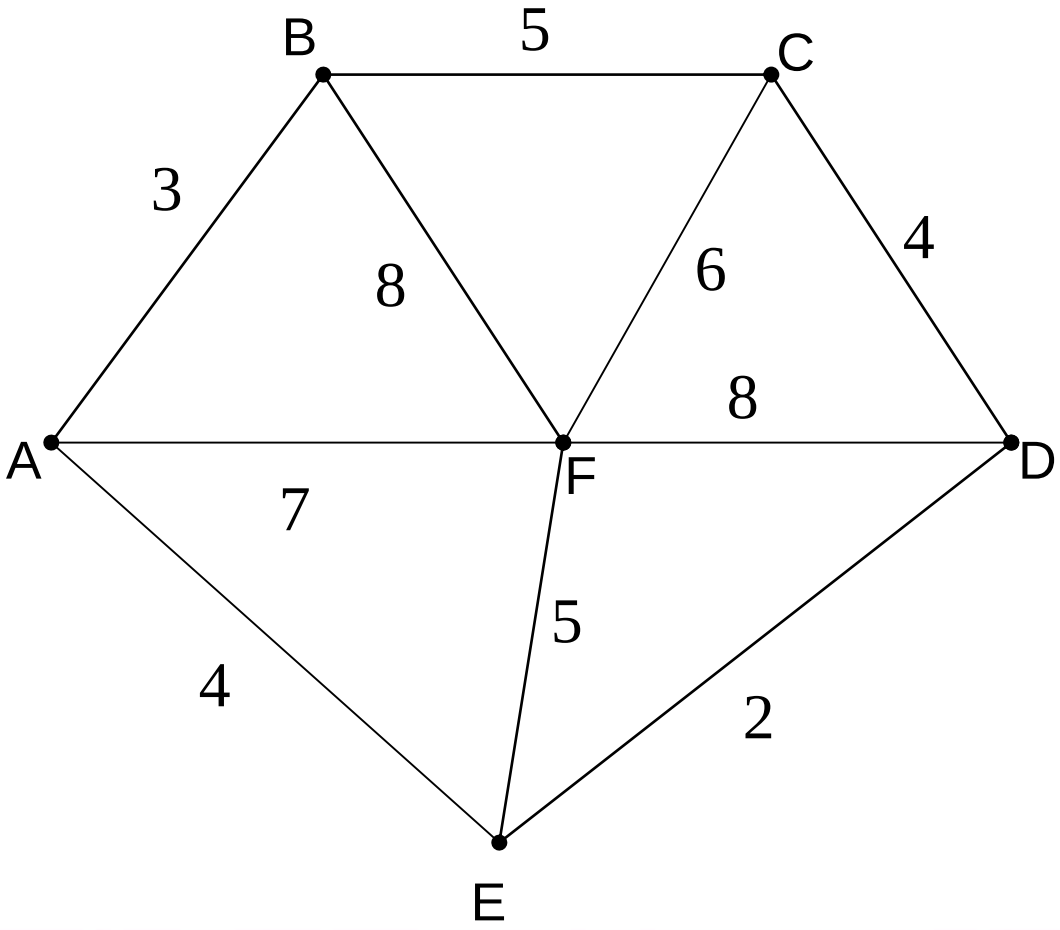
end (*T* is a minimum spanning tree of *G*)

Example

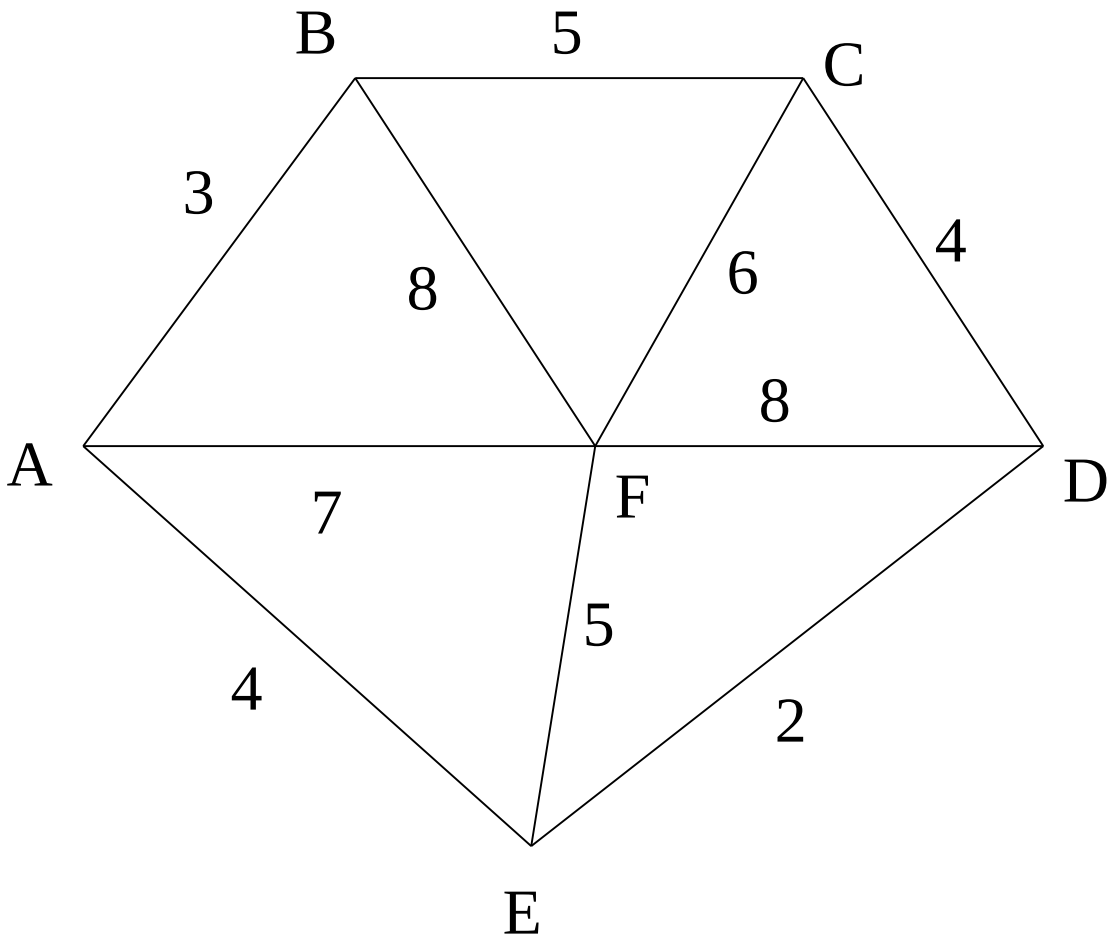
A cable company want to connect five villages to their network which currently extends to the market town of Avonford. What is the minimum length of cable needed?



We model the situation as a network, then the problem is to find the minimum connector for the network



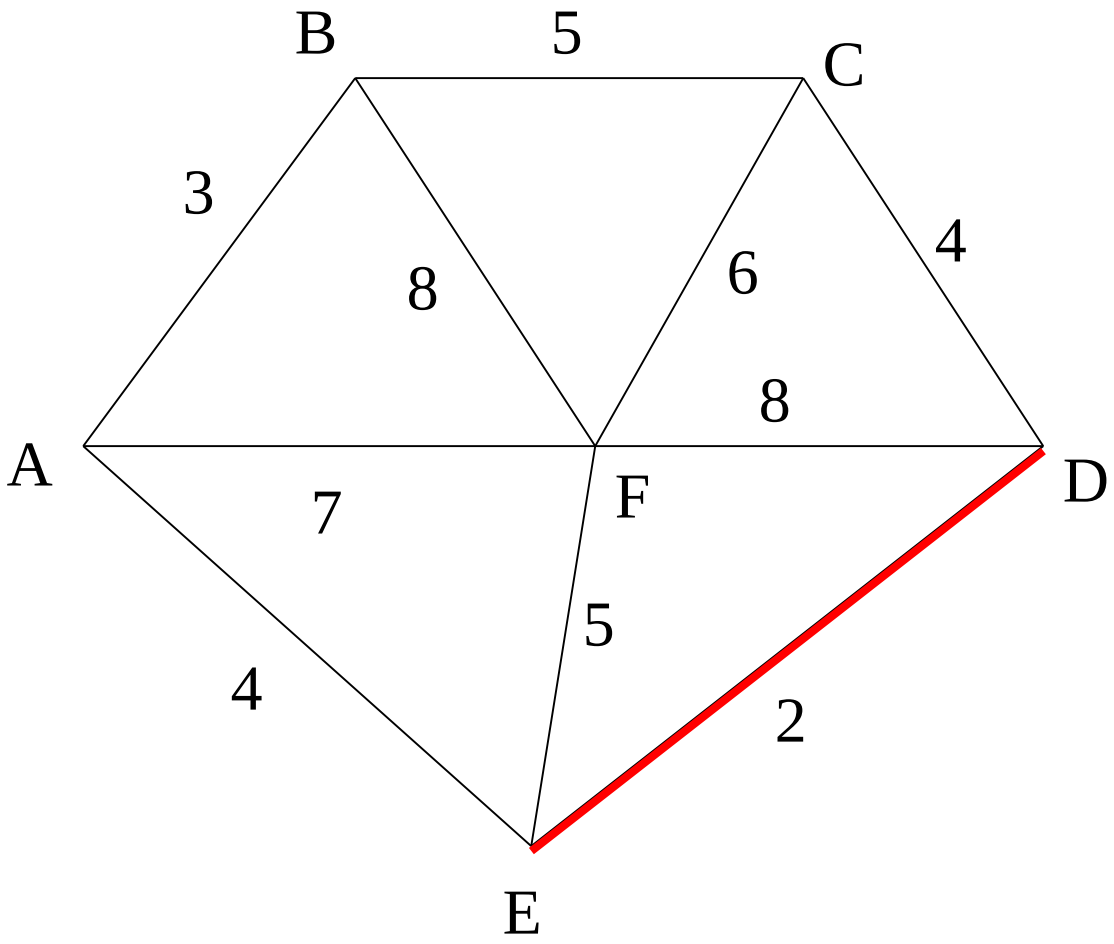
Kruskal's Algorithm



List the edges in order of size:

ED 2
AB 3
AE 4
CD 4
BC 5
EF 5
CF 6
AF 7
BF 8
CF 8

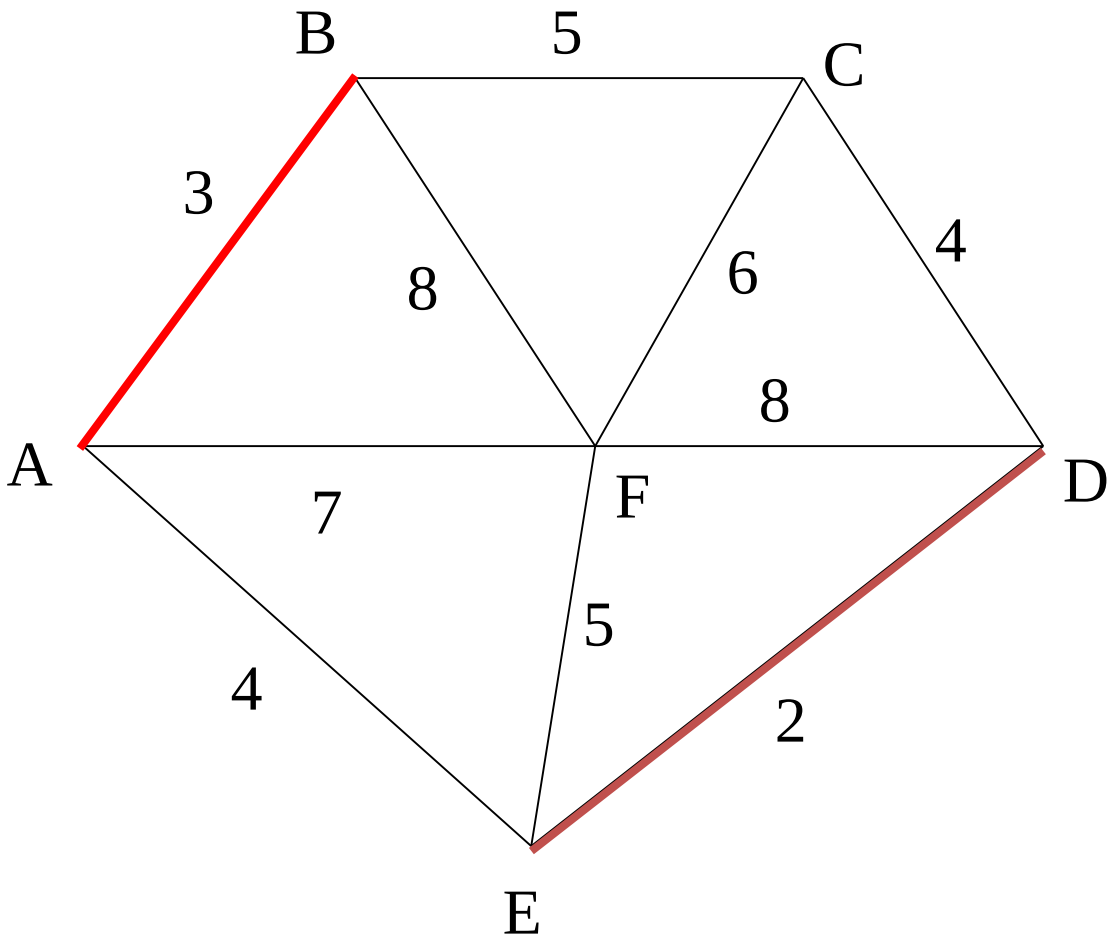
Kruskal's Algorithm



Select the shortest edge in the network

ED 2

Kruskal's Algorithm

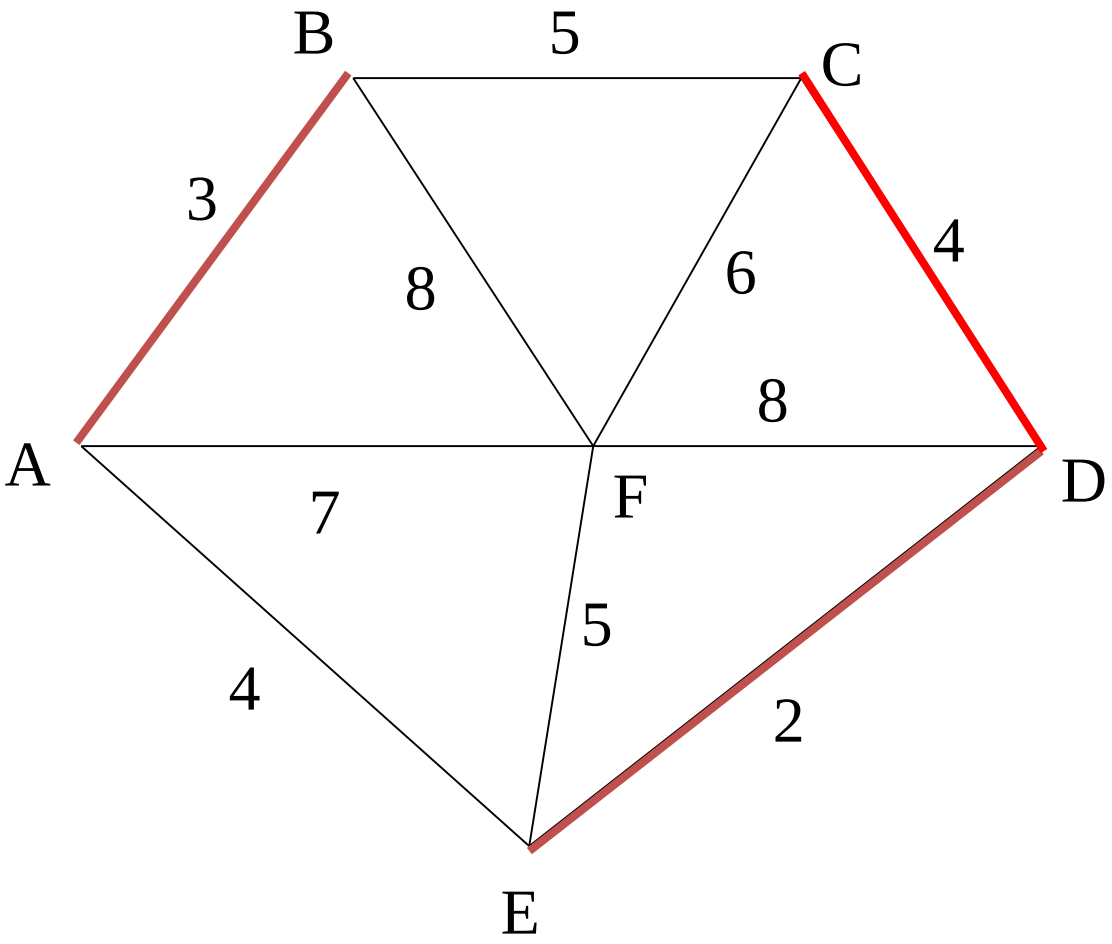


Select the next shortest edge which does not create a cycle

ED 2

AB 3

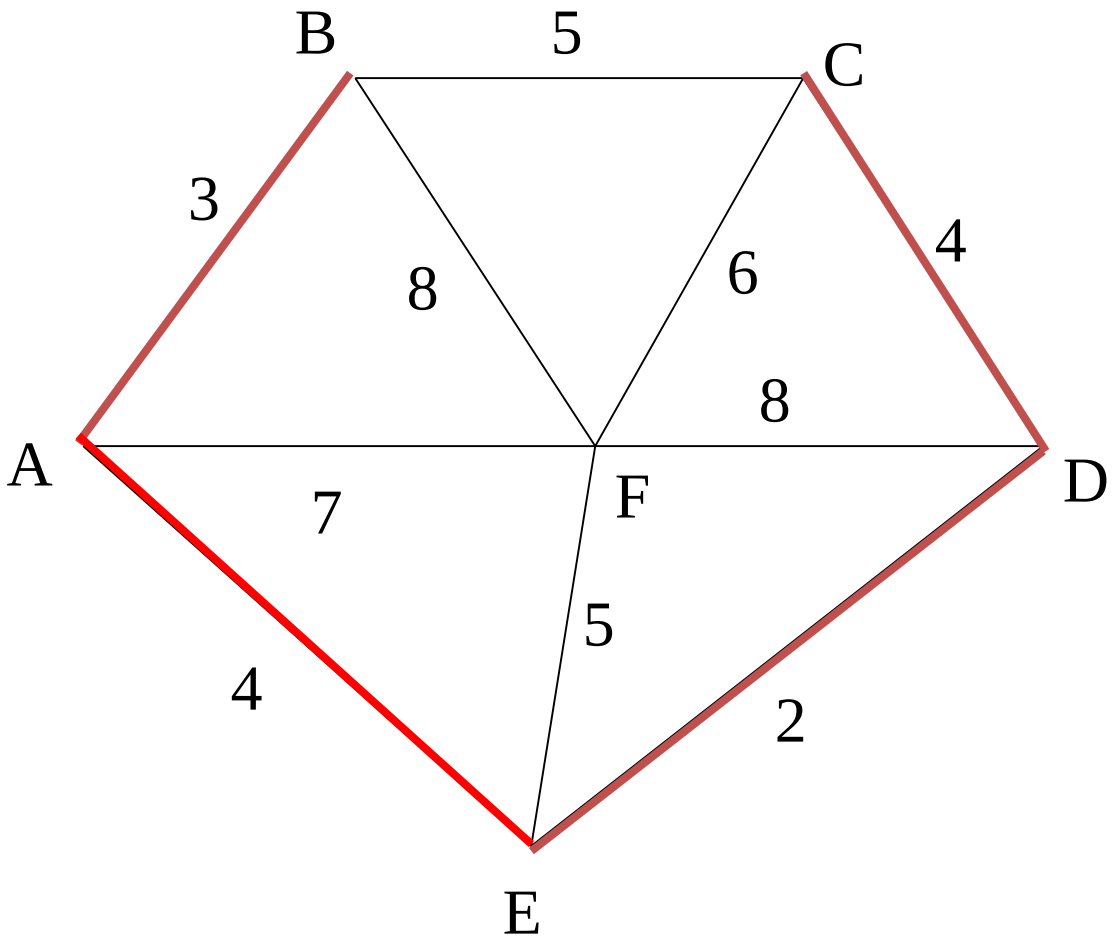
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3
CD 4 (or AE 4)

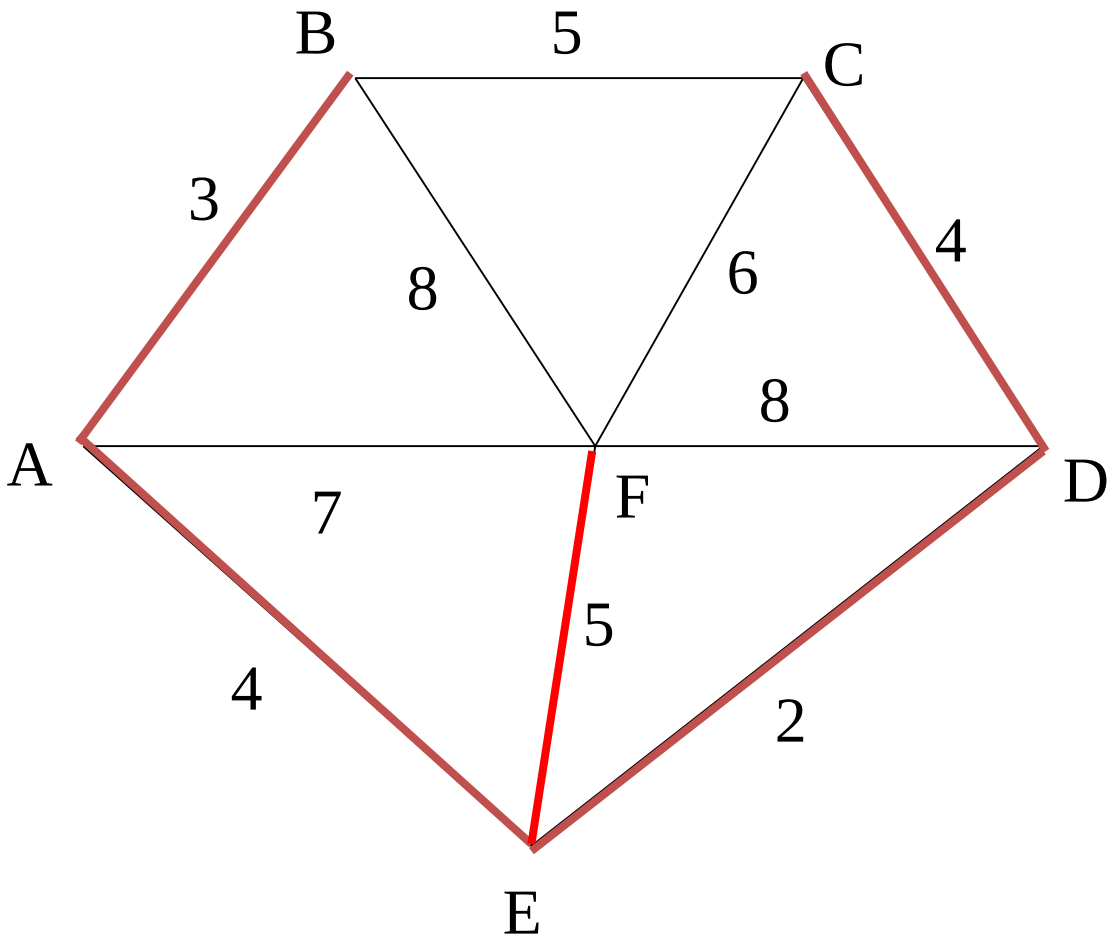
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3
CD 4
AE 4

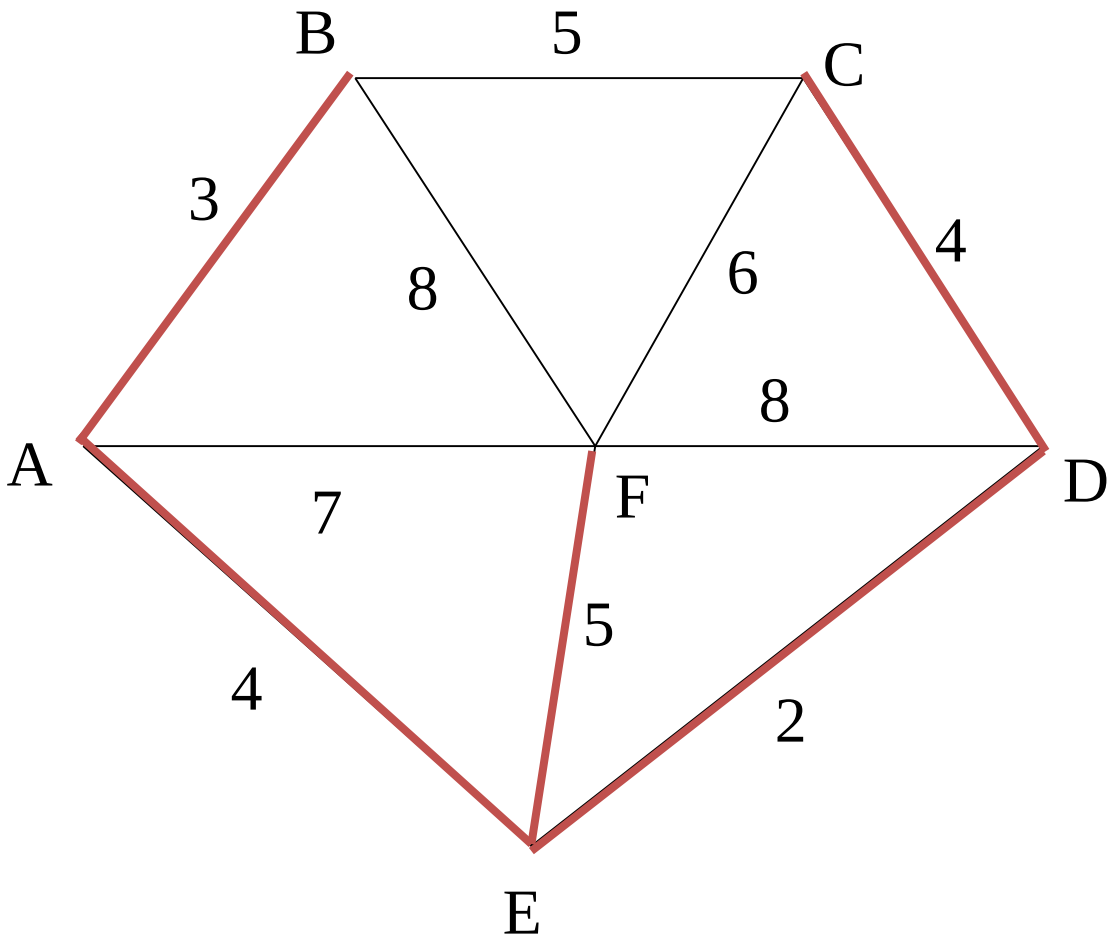
Kruskal's Algorithm



Select the next shortest edge which does not create a cycle

ED 2
AB 3
CD 4
AE 4
BC 5 – forms a cycle
EF 5

Kruskal's Algorithm



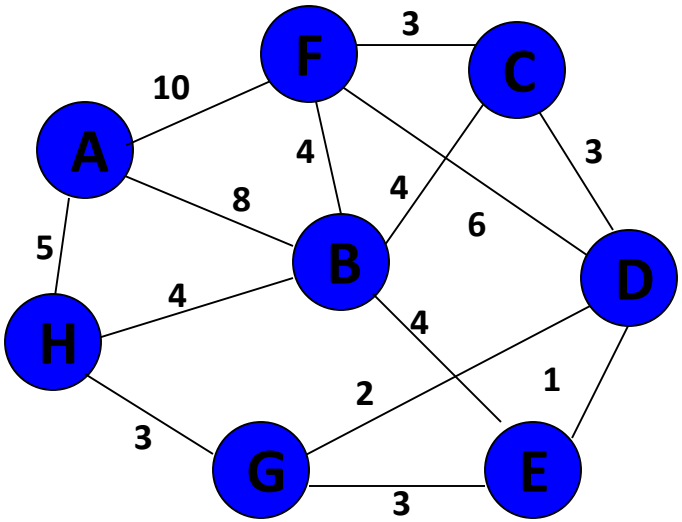
All vertices have been connected.

The solution is

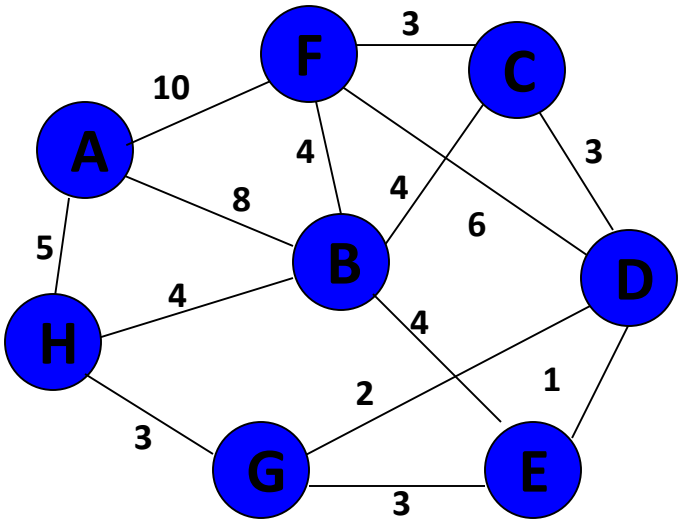
ED 2
AB 3
CD 4
AE 4
EF 5

Total weight of tree: 18

Walk-Through



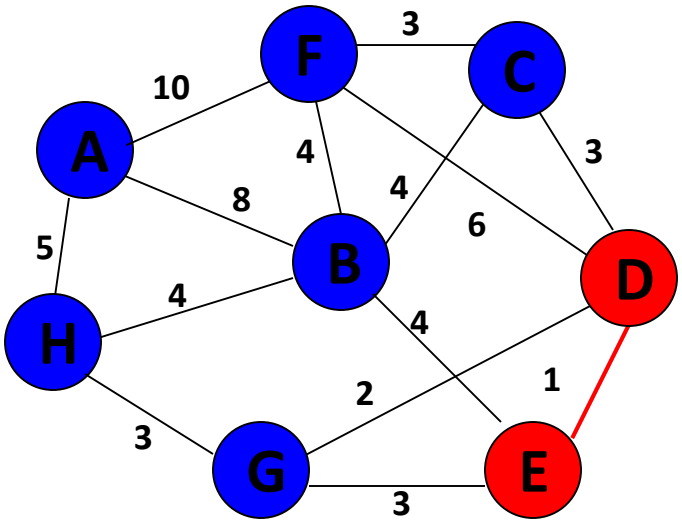
Consider an undirected, weight graph



Sort the edges by increasing edge weight

edge	d_v	
(D,E)	1	
(D,G)	2	
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

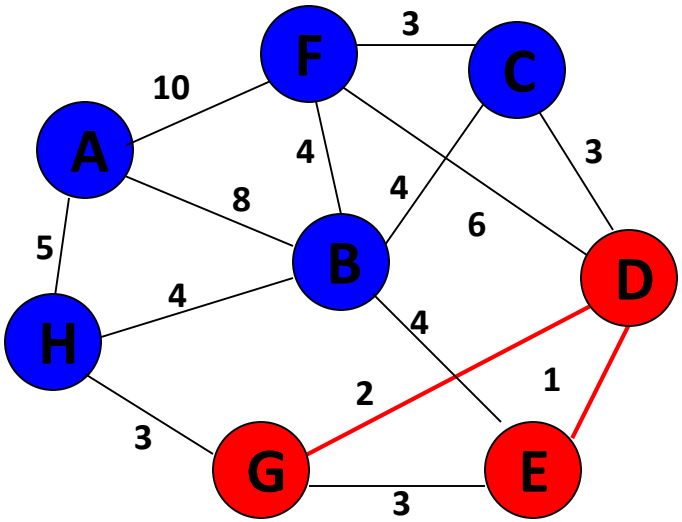
edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	



Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

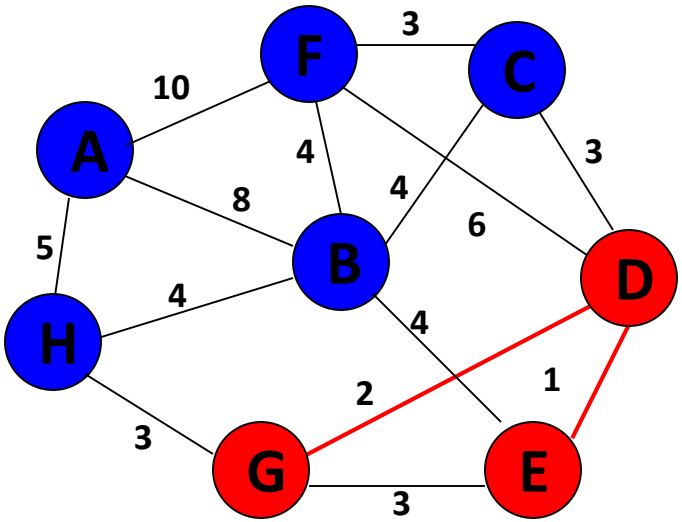
edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	



Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

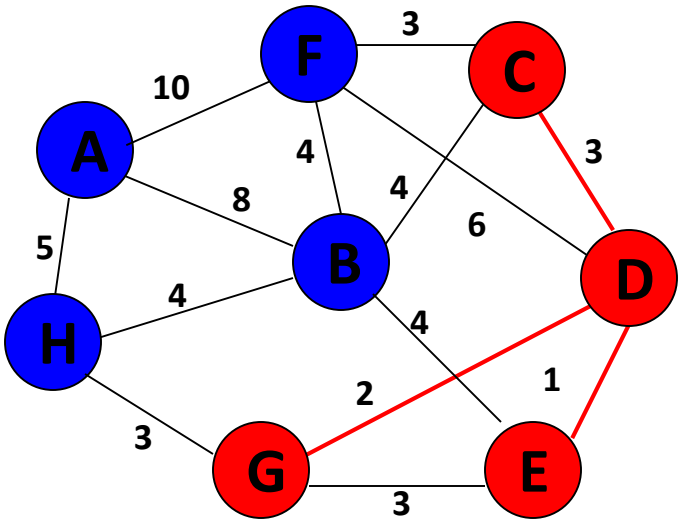


Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

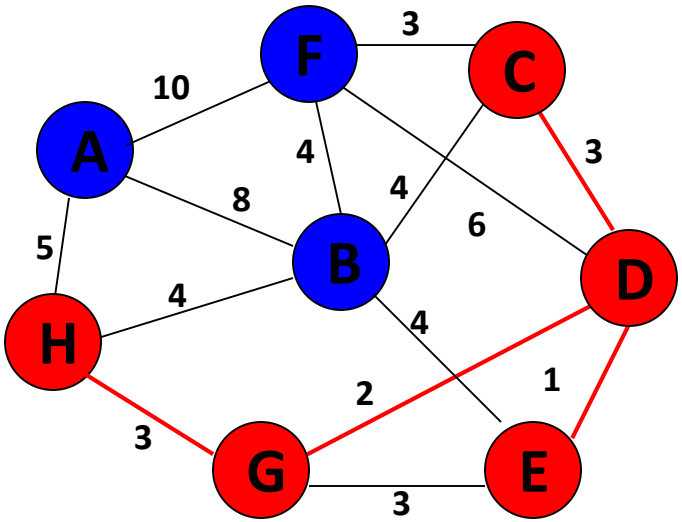
Accepting edge (E,G) would create a cycle



Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	
(C,F)	3	
(B,C)	4	

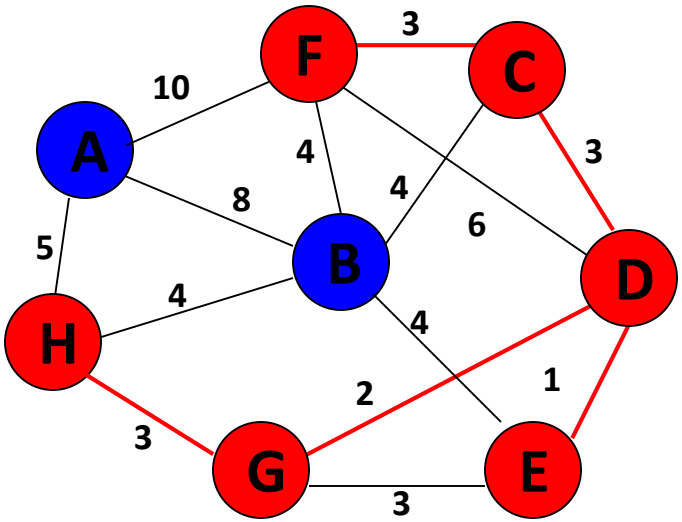
edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	



Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	
(B,C)	4	

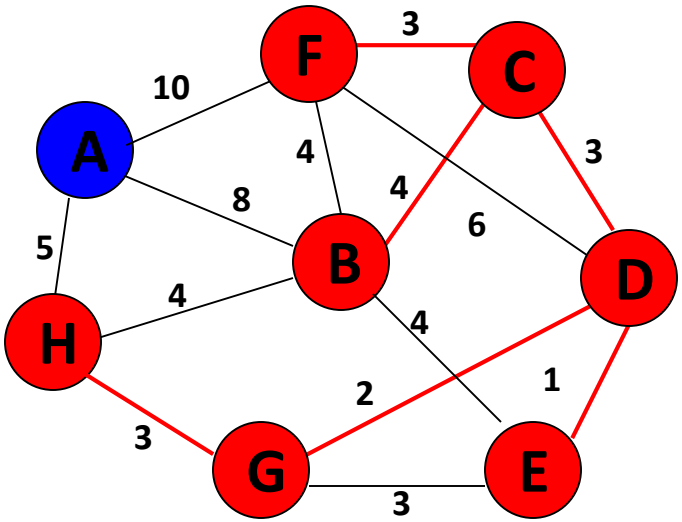
edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	



Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	

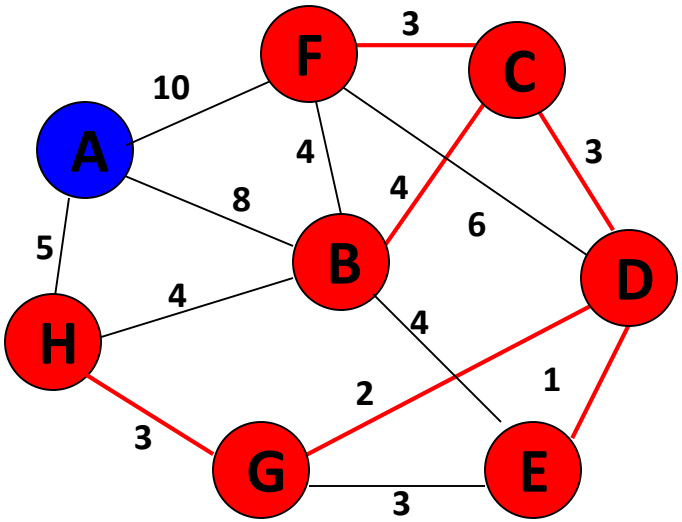
edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	



Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

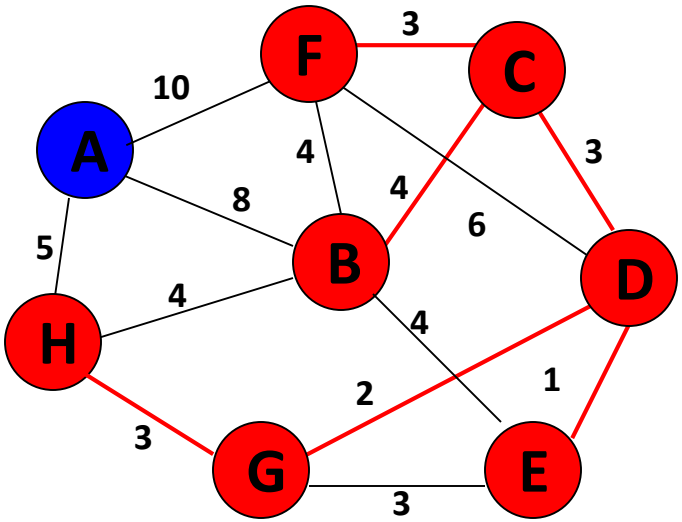
edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	



Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

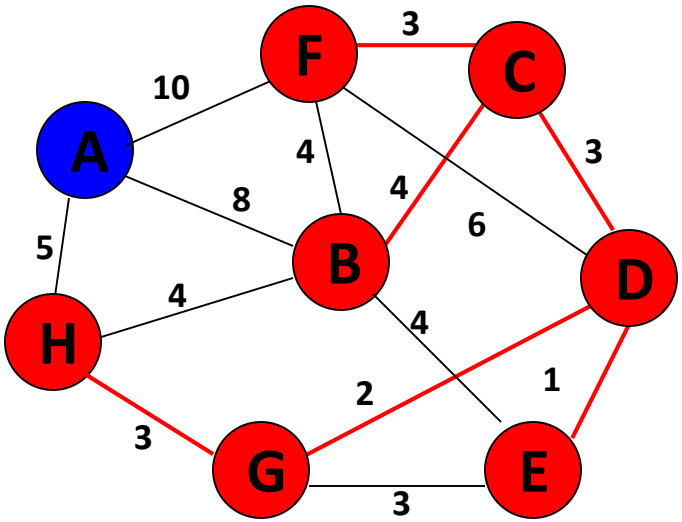
edge	d_v	
(B,E)	4	χ
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	



Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

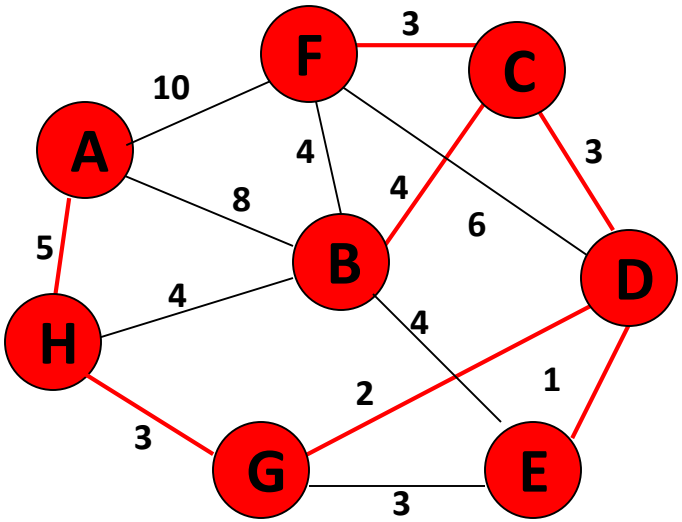
edge	d_v	
(B,E)	4	χ
(B,F)	4	χ
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	



Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

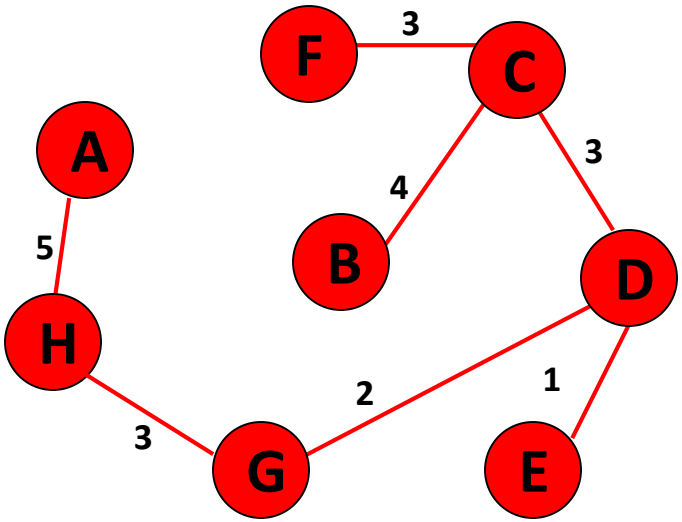
edge	d_v	
(B,E)	4	χ
(B,F)	4	χ
(B,H)	4	χ
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	



Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

edge	d_v	
(B,E)	4	χ
(B,F)	4	χ
(B,H)	4	χ
(A,H)	5	√
(D,F)	6	
(A,B)	8	
(A,F)	10	



Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

edge	d_v	
(B,E)	4	χ
(B,F)	4	χ
(B,H)	4	χ
(A,H)	5	√
(D,F)	6	
(A,B)	8	
(A,F)	10	

} not considered

Done

Total Cost = $\sum d_v = 21$