CHAPTER 4

GRAPH THEORY

CHAPTER 4

GRAPH THEORY

CHAPTER 4

GRAPH THEORY

CHAPTER 4

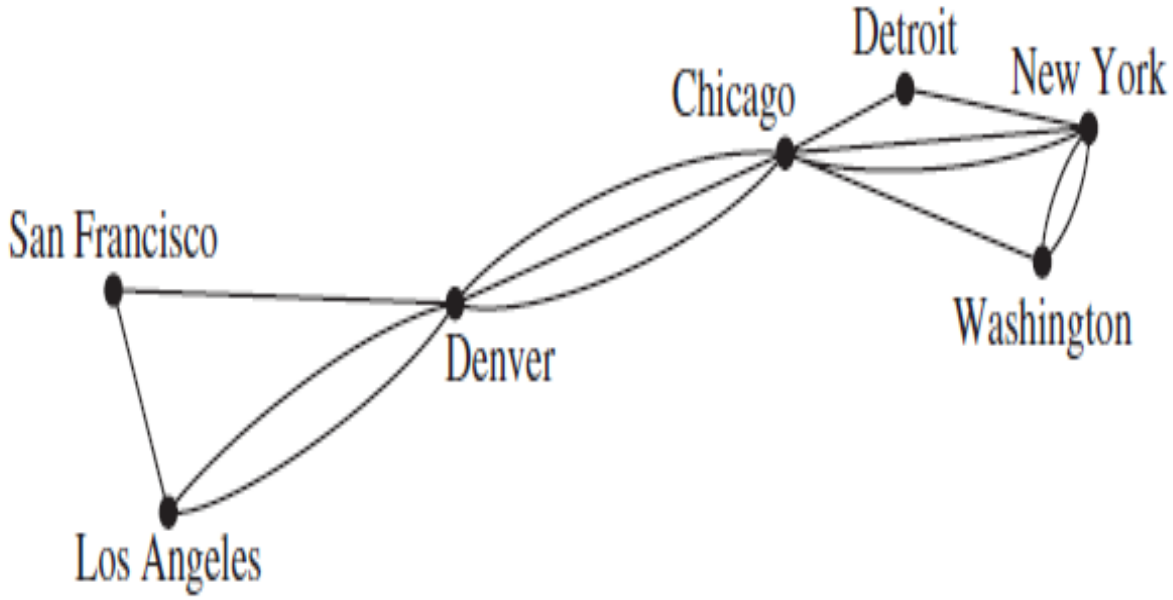
GRAPH THEORY

Introduction

- Graph theory studies relationship (edges) between objects (node)
- Why study graph theory?
- Model real-world systems such as social networks, computer network, transportation routes, and biological system
- Forms the basis for many algorithms in computer science, data science, optimization and AI
- Helps analyze connectivity, flow, paths, and network structure

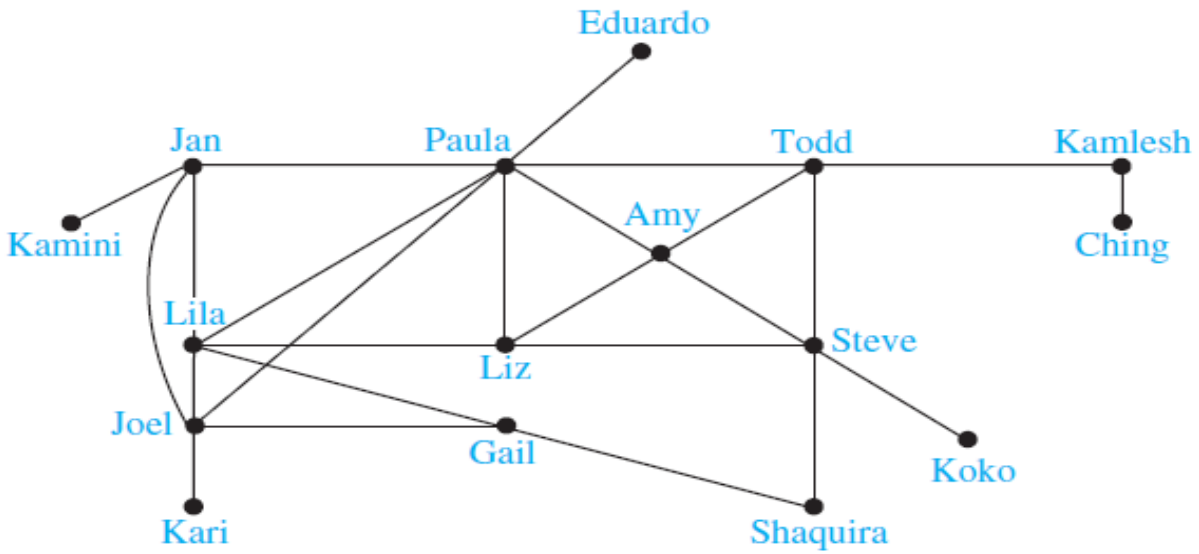
Graph Models

Example



Modeling computer network with multiple link with data centre. (undirected graph)

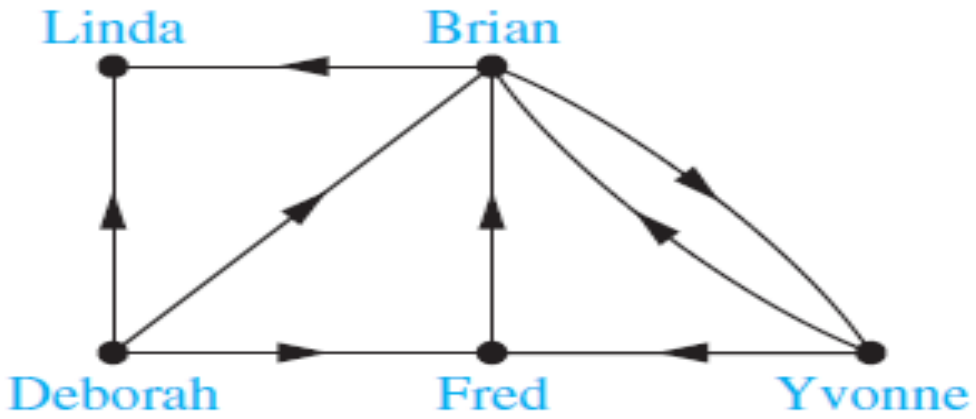
Example



Social networking , e.g. Facebook

(undirected graph)

Example



Influence graph (directed graph),

e.g., Deborah influenced Linda, fred and Brian

Definition, Notation, Key Concepts

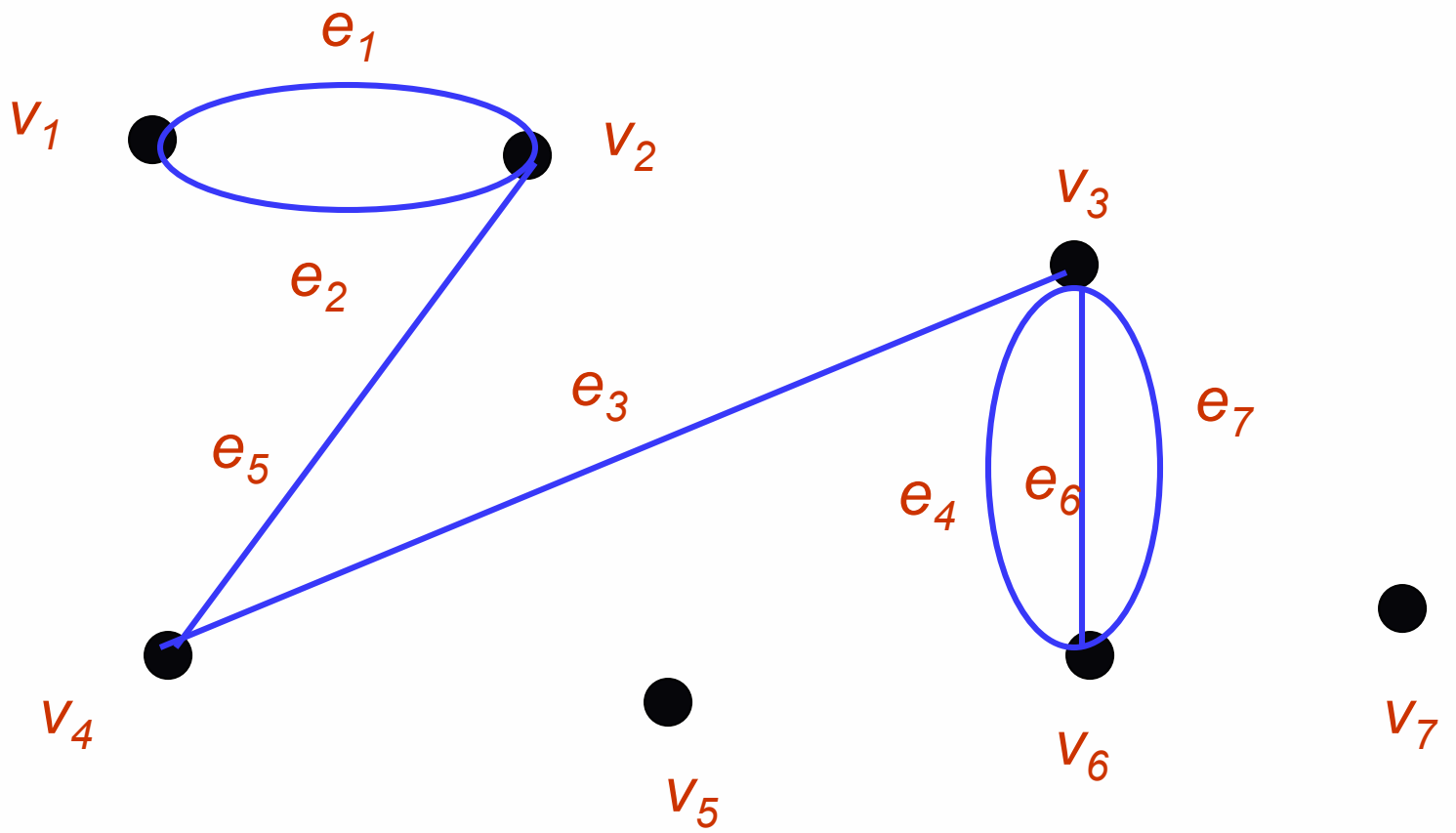
Definition

- A graph G is a triple (V, E, f), where
  - V is a finite nonempty set, called the set of vertices
  - E is a finite set (may be empty), called the set of edges
  - f is a function, called an incidence function, that assign to each edge, $e \in E$, a one-element subset $\{v\}$ or a two-element 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

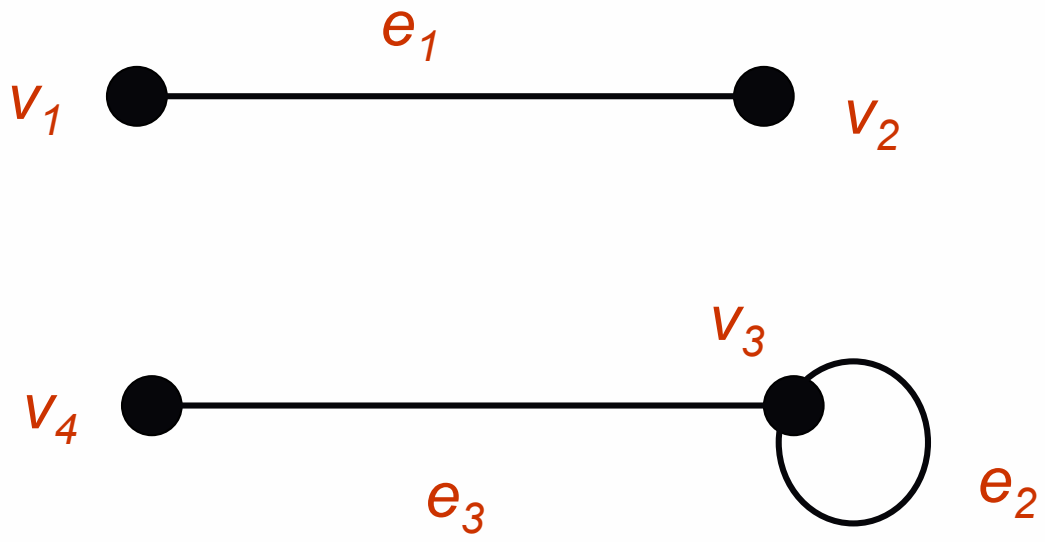
- Let,
  - $V=\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$
  - $E=\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$
- and f be defined by
  - $f(e_1)= f(e_2)=\{v_1, v_2\}$
  - $f(e_3)= \{v_4, v_3\}$
  - $f(e_4)= f(e_6)= f(e_7)=\{v_6, v_3\}$
  - $f(e_5)= \{v_2, v_4\}$
- Then $G=(V,E,f)$ is a graph

Example



Example

- Let $V=\{v_1, v_2, v_3, v_4\}$, $E=\{e_1, e_2, e_3\}$ and
  - $f(e_1)=\{v_1, v_2\}$
  - $f(e_2)=\{v_3, v_3\}$
  - $f(e_3)=\{v_3, v_4\}$
- Then $G=(V,E,f)$ is a graph



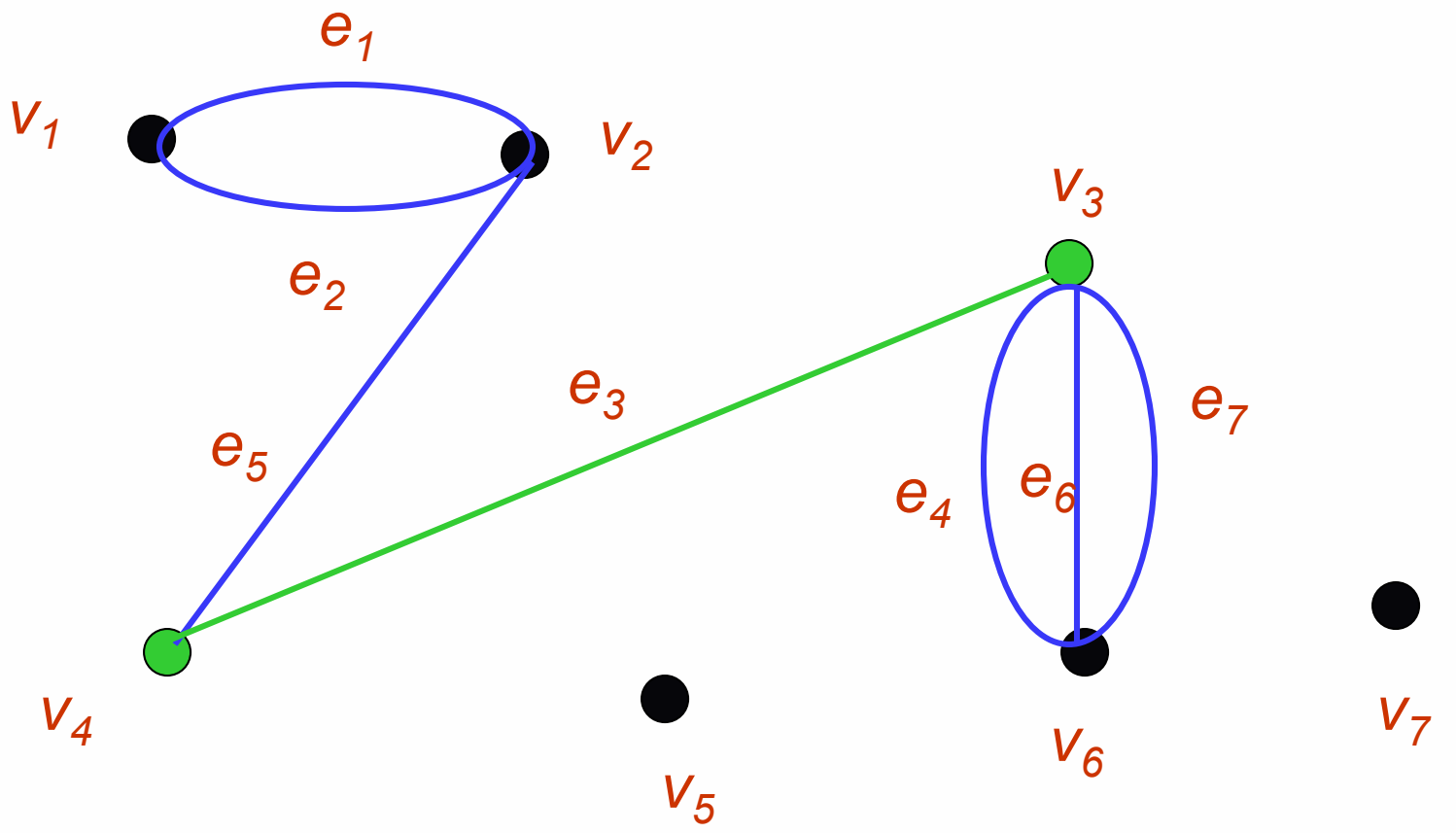
Adjacent Vertices

- An edge *e* in a graph that is associated with the pair of vertices *v* and *w* is said to be incident on *v* and *w*, and *v* and *w* are said to be incident on *e* and to be adjacent vertices.

- A vertex that is an endpoint of a loop is said to be adjacent to itself.

Example:

v_4 and v_3 are adjacent

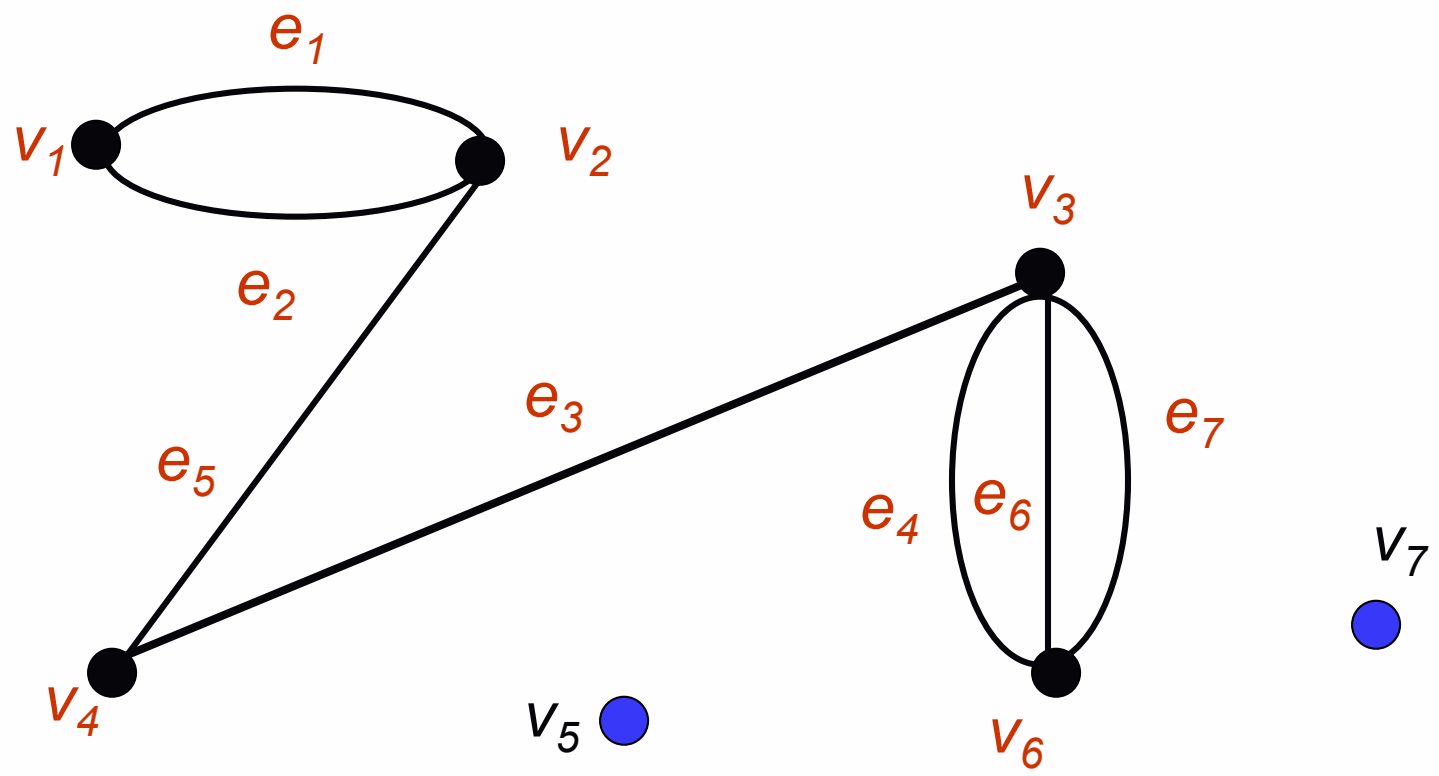


Isolated Vertex

- 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.

Example

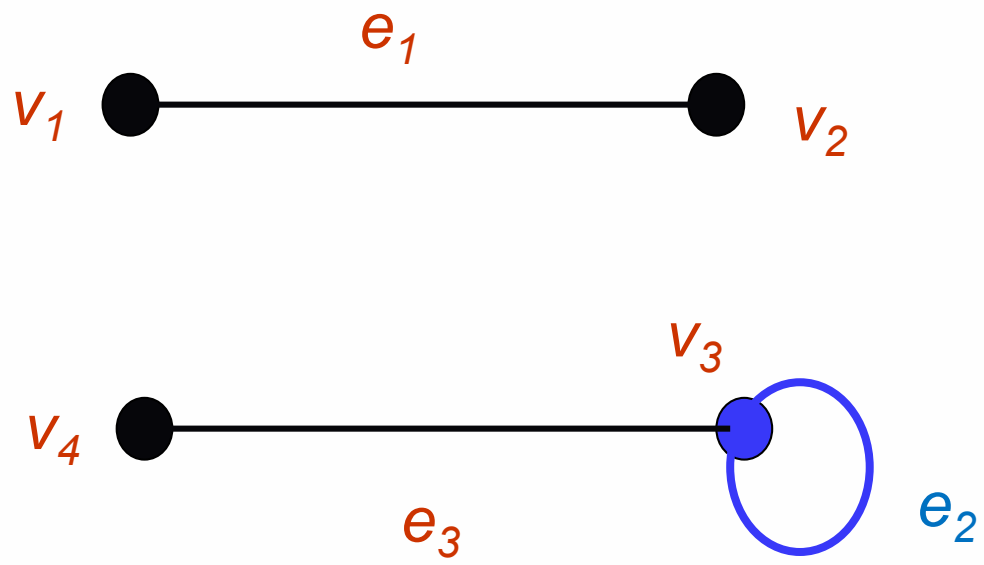
- v_5 and v_7 are isolated vertices.



Loop

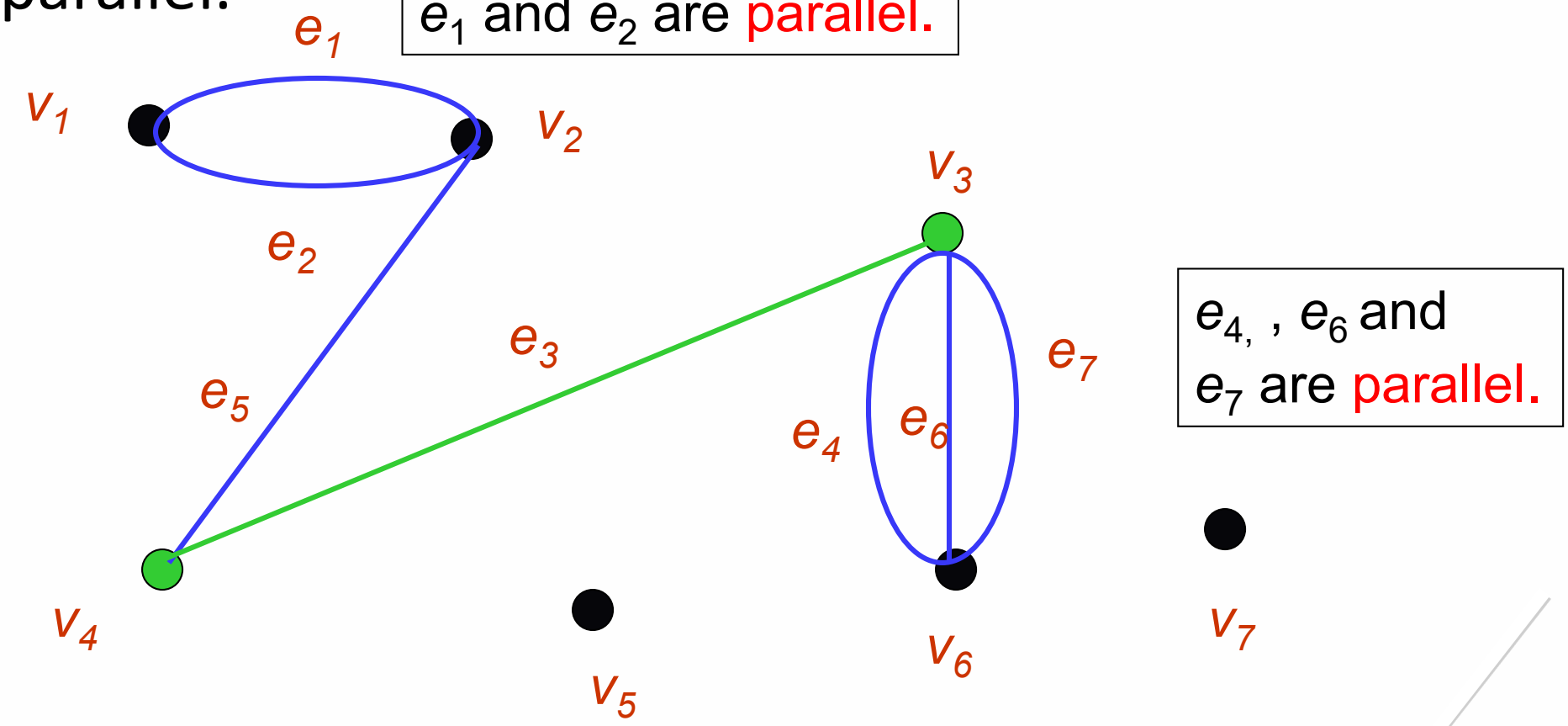
- An edge incident on a single vertex is called a loop.

  Example: e_2 is a loop

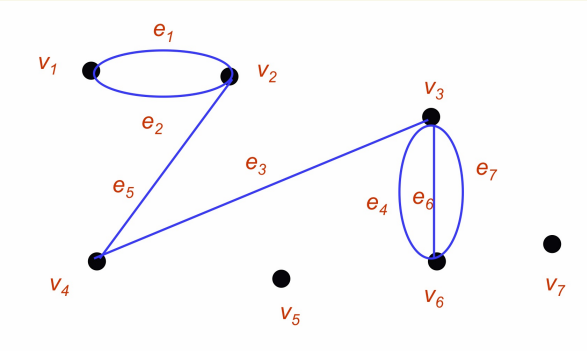


Parallel Edges

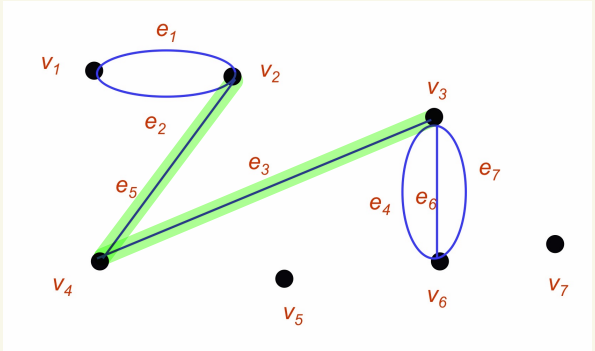
Two or more distinct edges with the same set of endpoints are said to be parallel.



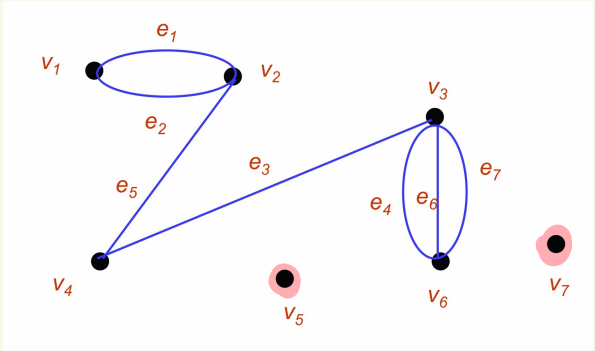
Notation



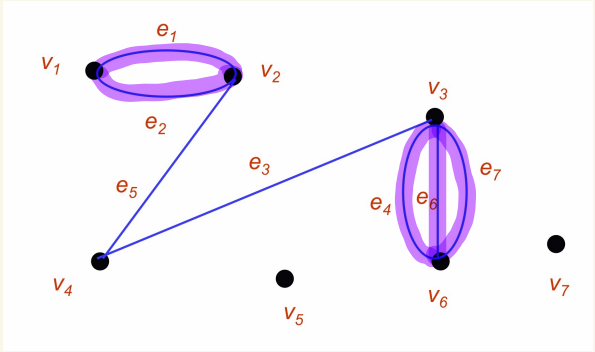
Adjacent vertices



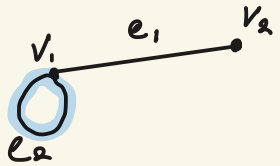
isolated vertices



Parallel edges

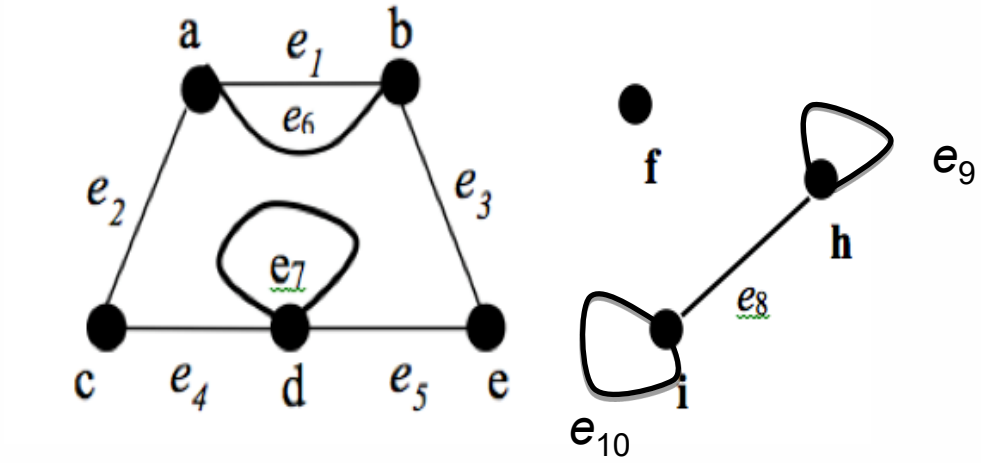


Loop



Example

Given a graph as shown below.



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

b) 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.

a) $V = \{a, b, c, d, e, f, h, i\}$

$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\}$

$f(e_1) = f(e_6) = \{a, b\}$

$f(e_2) = \{a, c\}$

$f(e_3) = \{b, e\}$

$f(e_4) = \{c, d\}$

$f(e_5) = \{d, e\}$

$f(e_7) = \{d, d\}$

$f(e_8) = \{i, h\}$

$f(e_9) = \{h, h\}$

$f(e_{10}) = \{i, i\}$

b)

i) incident on $a = e_1, e_2, e_6$

ii) Vertices that adjacent to $a = c, b$

iii) edges that adjacent to $e_2 = e_1, e_4, e_6$

iv) Loops $= e_7, e_9, e_{10}$

v) parallel = e_1 and e_6 are parallel

vi)

Example

Solution:

a) Vertex set, V = {a, b, c, d, e, f, i, h} and
the set of edges, E = {e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}}

Edge	Endpoints
e_1	{a, b}
e_2	{a, c}
e_3	{b, e}
e_4	{c, d}
e_5	{d, e}
e_6	{a, b}
e_7	{d}
e_8	{i, h}
e_9	{h}
e_{10}	{i}

Example

Solution:

b)

Edges incident on a : e1, e2, e6
Vertices adjacent to a : c, b
Edges adjacent to e_2 : e1, e4, e6
Loops : e7, e9, e10
parallel edges : e1, e6
adjacent to themselves: i , h, d
isolated vertices : f

Try this

- Can we modelled MRT/LRT system in a city as a graph?
- What do the nodes represent?
- What do the edges represent?
- Directed or directed graph?
- Should loop(s) allowed?
- Should multiple edges allowed?

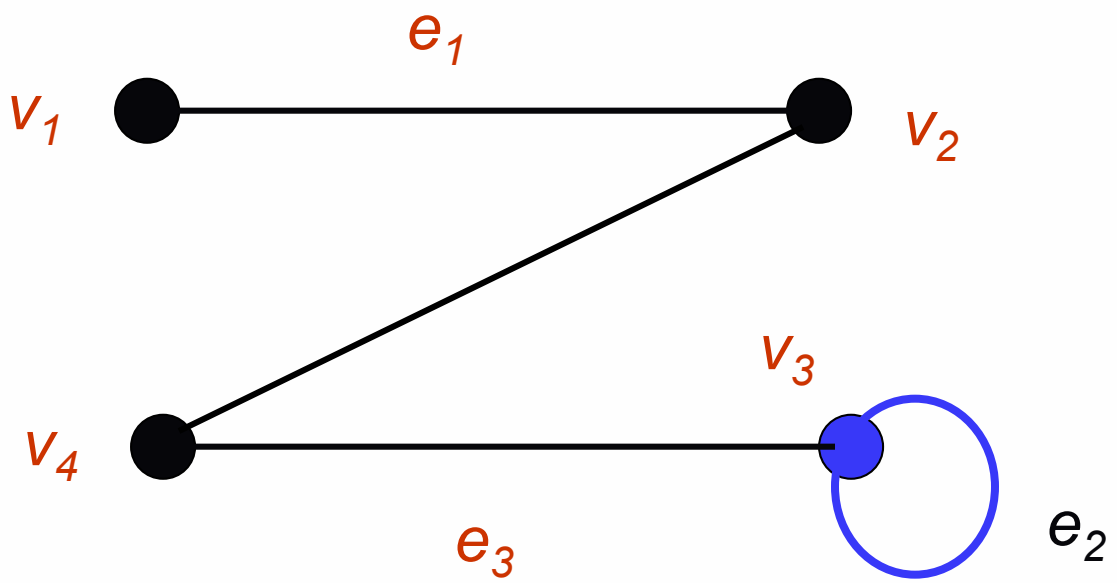
The Concept of Degree

Degree of a vertex

- Let G be a graph and v be a vertex of 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.

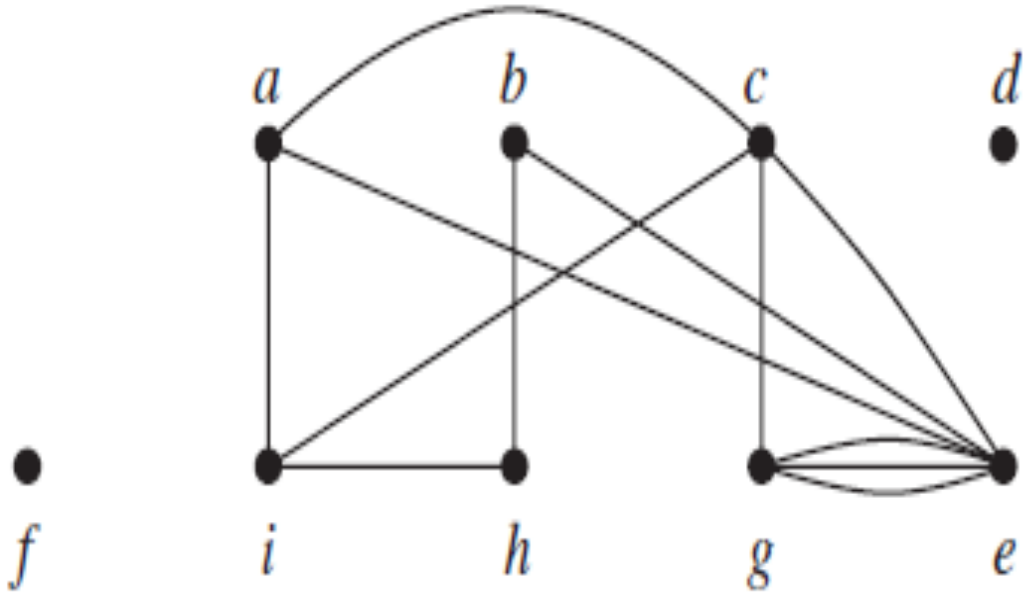
Example

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



Try this

3.



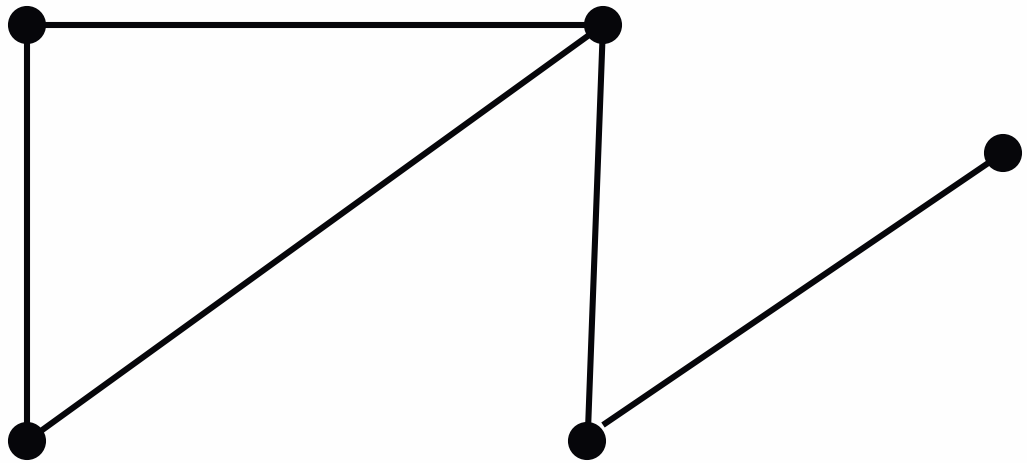
Find

1. Number of vertices
2. Parallel edges
3. Vertices adjacent to *b*
4. Isolated vertices

Types of Graphs

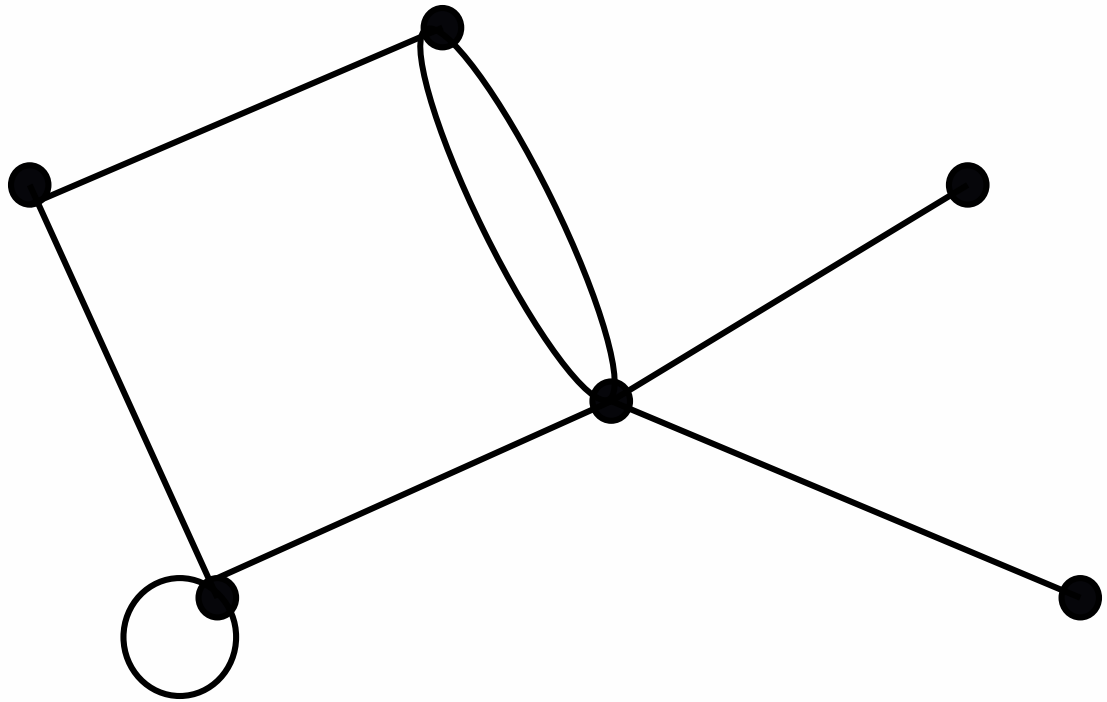
Simple Graphs

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



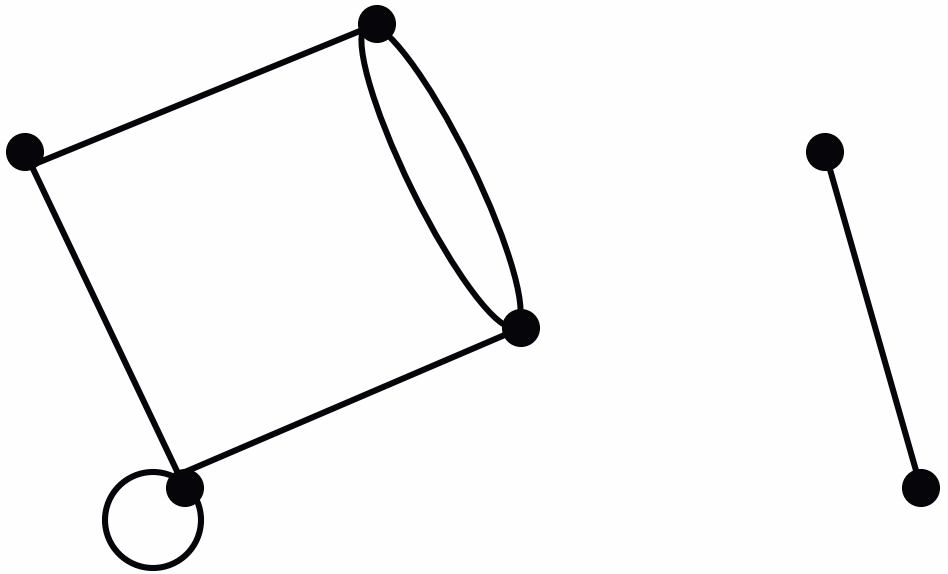
Connected Graph

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



Example

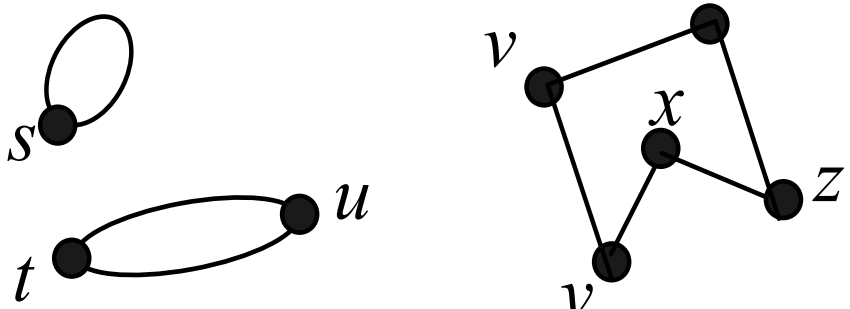
- not connected



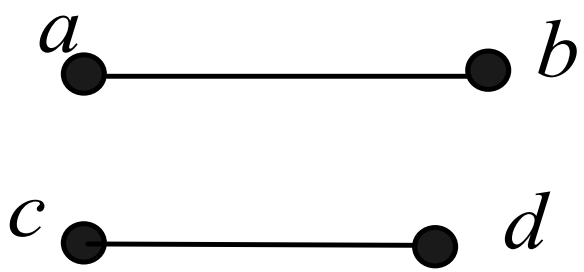
Regular Graphs

- 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



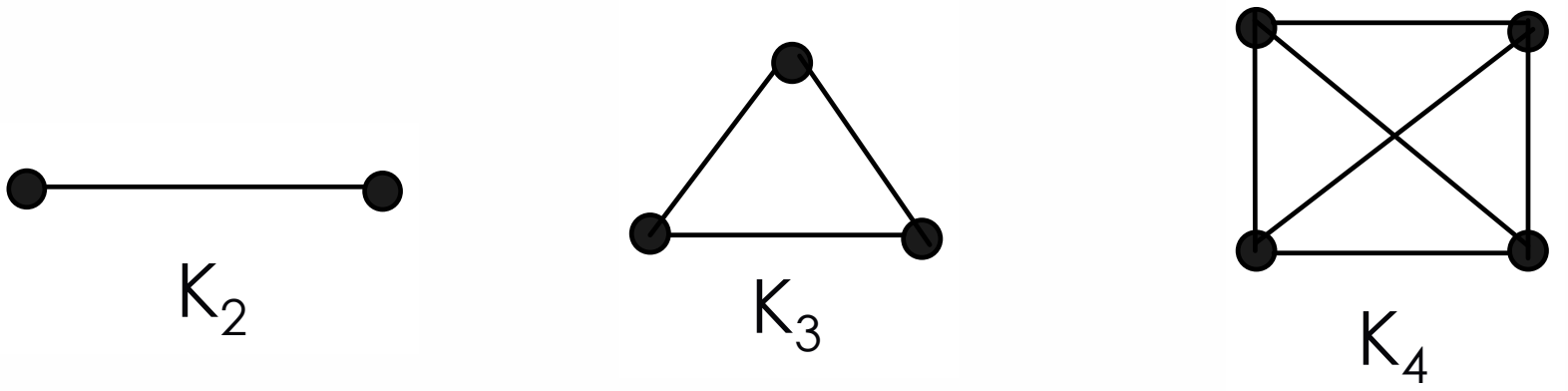
$k=2$



$k=1$

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**

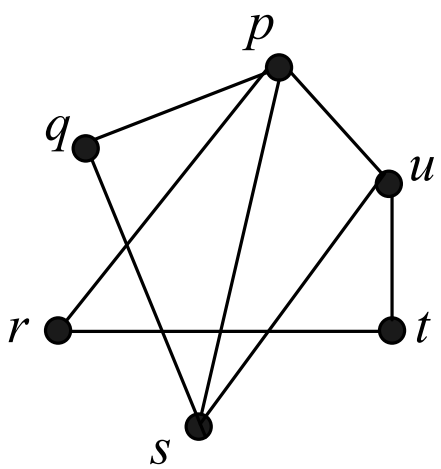


Subgraph

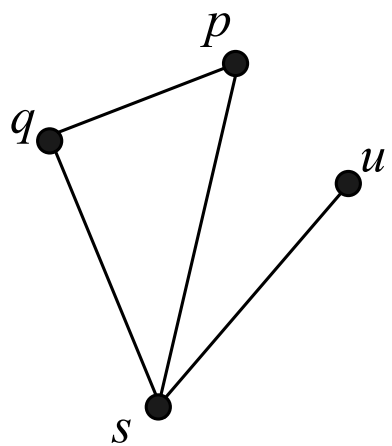
- Let $G=(V,E)$ be a graph.

- $H=(U,D)$ is a subgraph of G if
  - $U \subseteq V$ and $D \subseteq E$
  - for every edge $e \in D$, if e is incident on v and w, then $v,w \in V$.

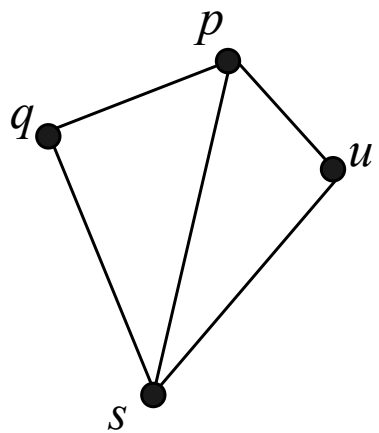
Example



G



H_1

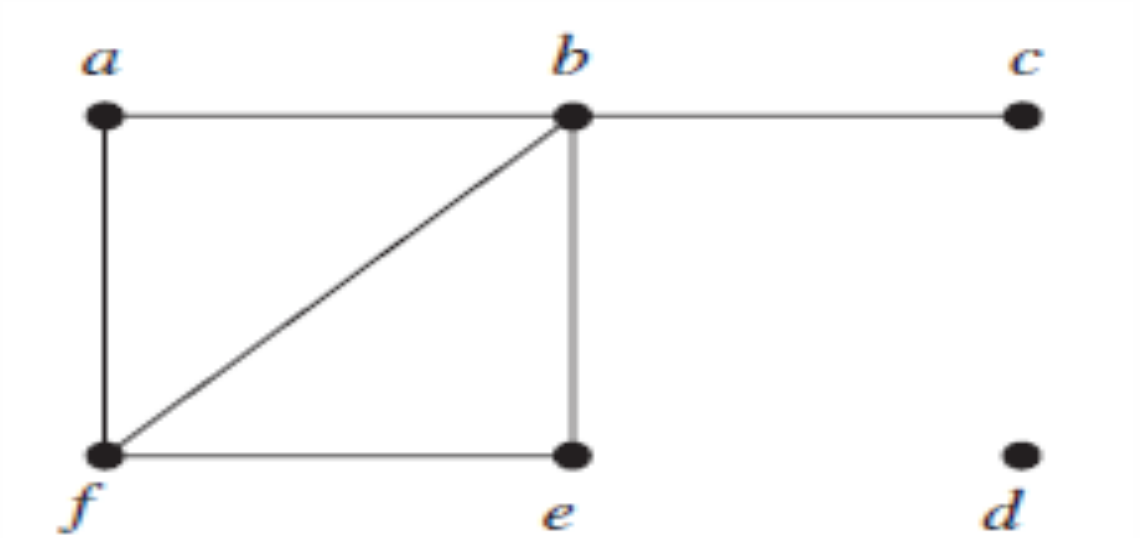


H_2

H_1 and H_2 subgraph G

Try this!

- Add edge(s) to make the graph becomes simple graph
- Remove edges to make it 2-regular graph



Graph Representation

Matrix Representation of a Graph

- 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 Matrices

- Let *G* be a graph with *n* vertices.
- The adjacency matrix, A_G is an $n \times n$ matrix $[a_{ij}]$ such that,

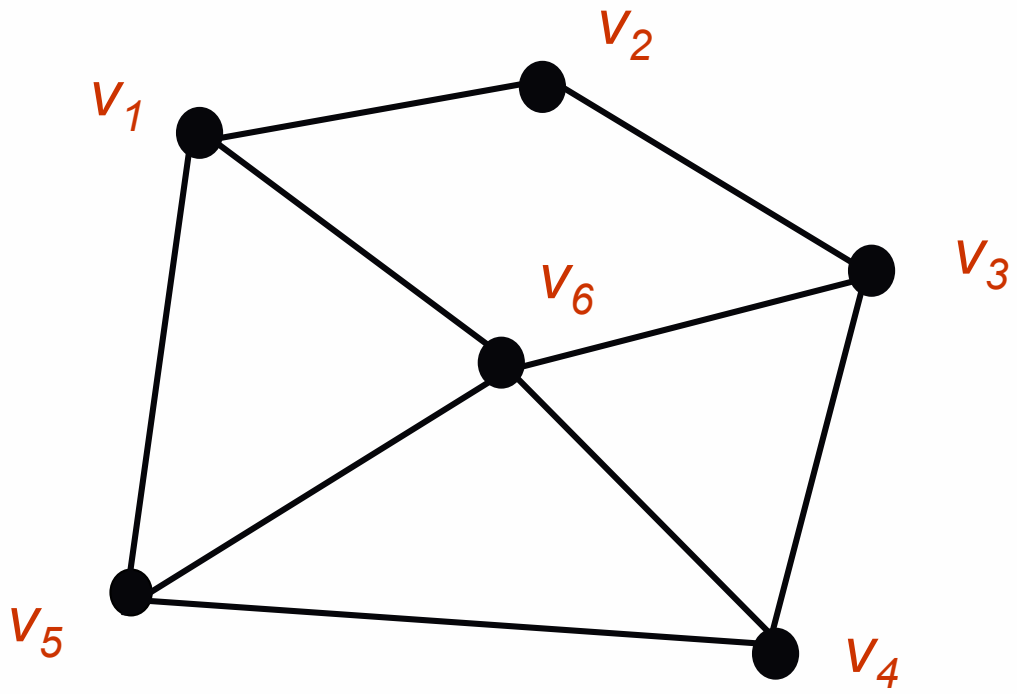
  a_{ij}= the number of edges from v_i to v_j, {undirected G}

  or,

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

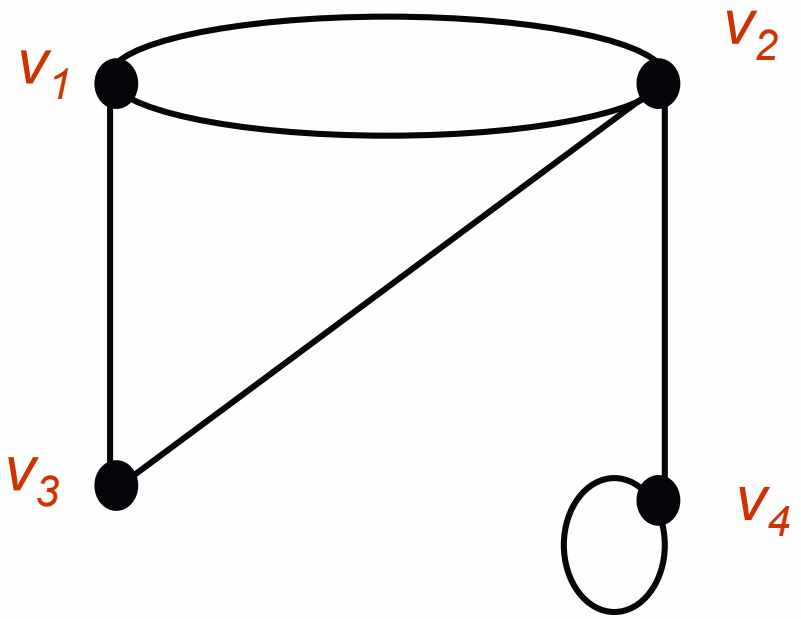
  for all i, j = 1,2,….., n.

Example



$$A_G = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

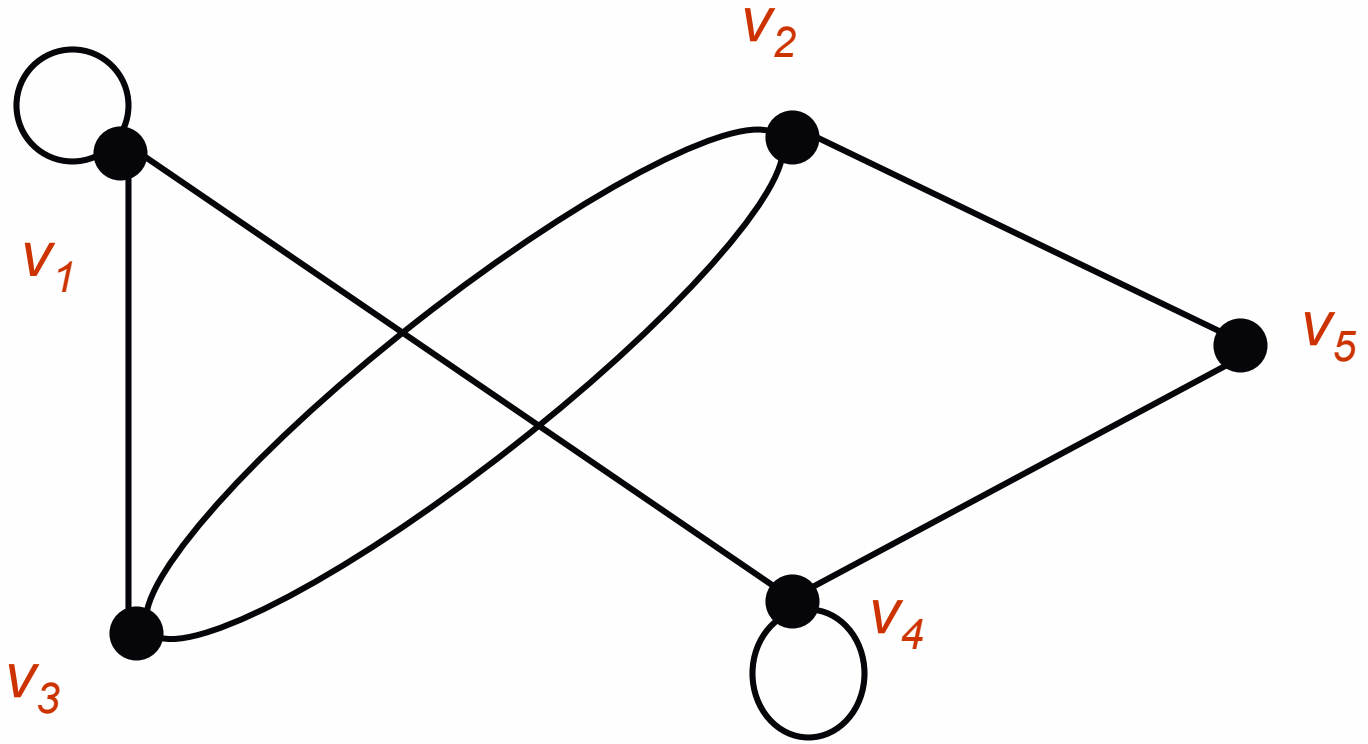
Example



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

Example

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



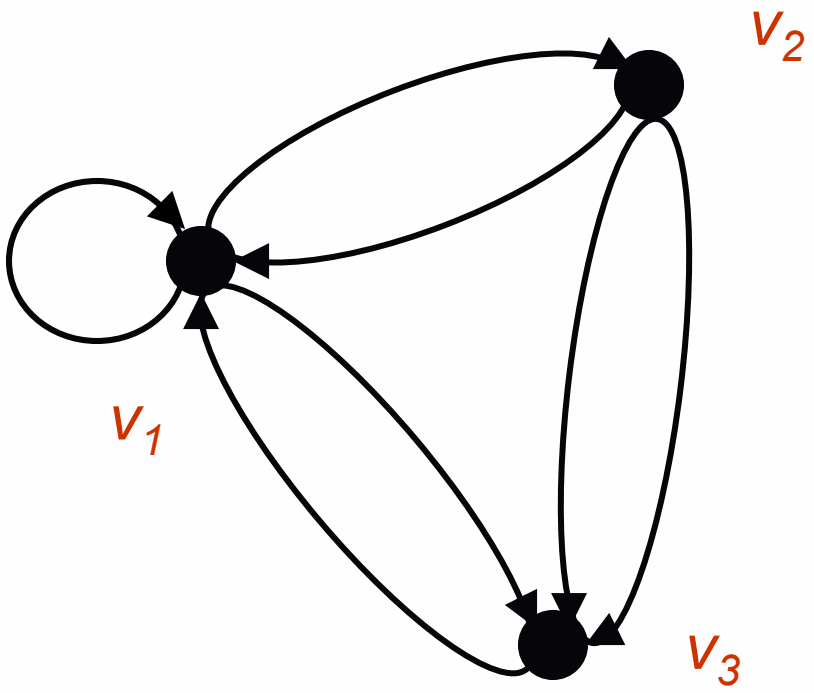
Adjacency Matrices

- Notice that the matrix A_G is a **symmetric matrix** if it is representing an undirected graph, where

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

- If G is a directed graph (**digraph**), then A_G need not be a symmetric matrix.

Example symmetry matrix



$a_{12}=1$

$a_{21}=1$

$$\begin{bmatrix} 1 & \boxed{1} & 1 \\ \boxed{1} & 0 & 2 \\ 1 & 0 & 0 \end{bmatrix}$$

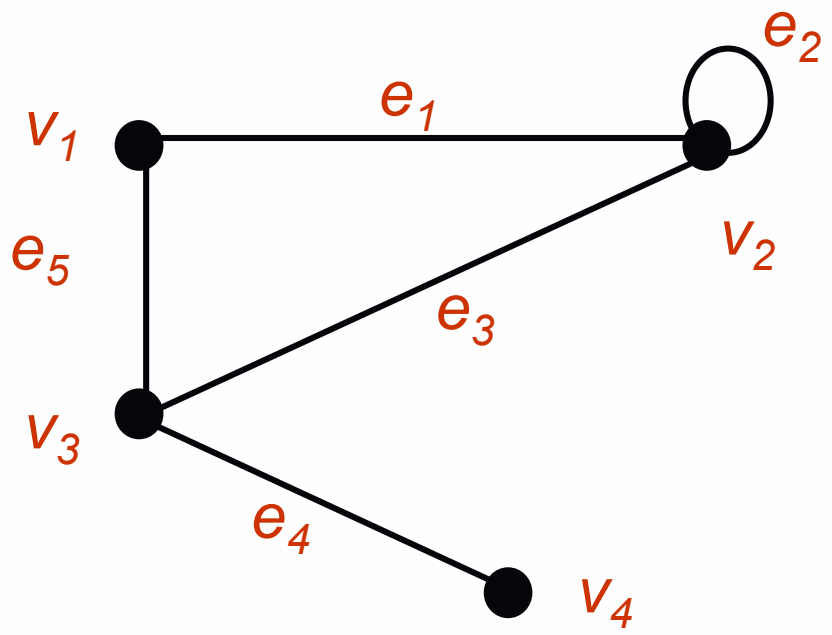
Incidence Matrices

- 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



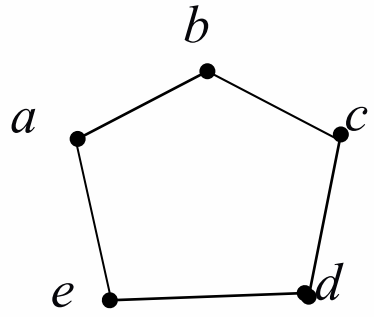
$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Notice that the sum of the *i*th row is the degree of v_i

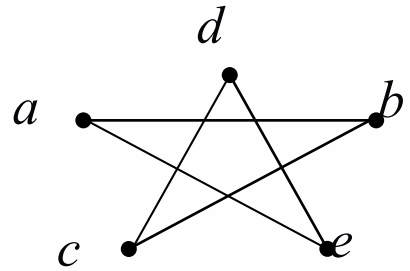
Isomorphisms

Isomorphism

- Are these 2 graphs the same?



G_1



G_2

- When we say that 2 graphs are the same mean they are isomorphic to each other.
- In simple term, Isomorphic graphs are structurally identical, even if they look different.

Isomorphism

Definition

- The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there exists a one to-one and onto function f from V_1 to V_2 with the property that a and b are adjacent in G_1 if and only if $f(a)$ and $f(b)$ are adjacent in G_2, for all a and b in V_1. Such a function f is called an *isomorphism*.

Isomorphism

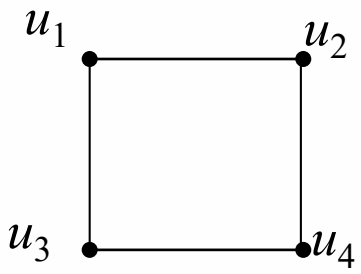
- An edge e is incident on v and w in G_1 if and only if the edge $g(e)$ is incident on $f(v)$ and $f(w)$ in G_2.
- The pair of functions f and g is called an isomorphism of G_1 onto G_2.
- Graphs G_1 and G_2 are isomorphic if and only if for some ordering of their vertices, their adjacency matrices are equal.

Isomorphism

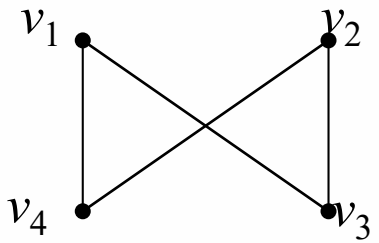
- Key properties of isomorphic graphs
  - the same number of vertices and edges,
  - the same degrees for corresponding vertices,
  - the same number of connected components,
  - the same number of loops and parallel edges,
  - both graphs are connected or both graph are not connected,
  - pairs of connected 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

- Determine whether *G* is isomorphic to *H.*



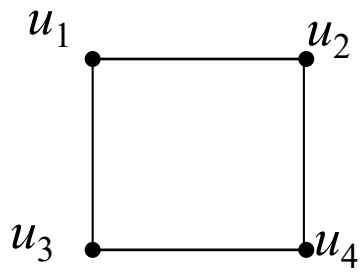
G



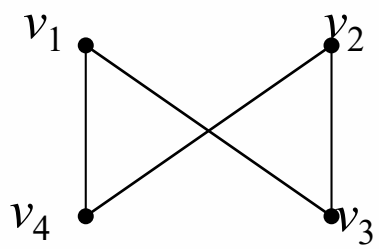
H

Example

- Both graphs are simple and have the same number of vertices and the same number of edges.



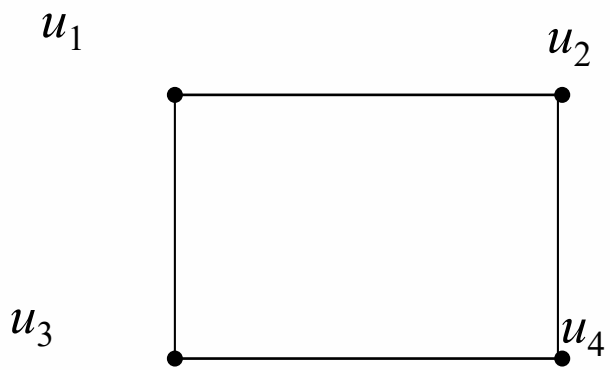
G



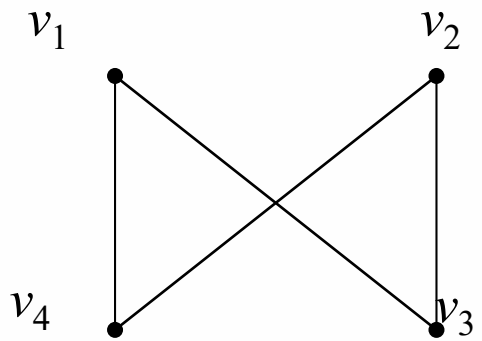
H

Example

- All the vertices of both graphs have degree 2.



G

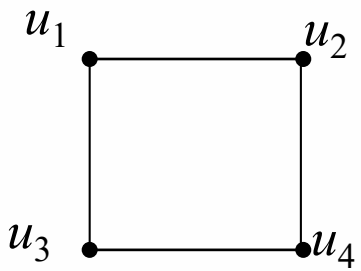


H

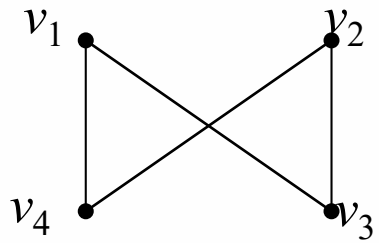
example

- Define $f: U \rightarrow V$, where $U=\{u_1, u_2, u_3, u_4\}$ and $V=\{v_1, v_2, v_3, v_4\}$

$f(u_1)=v_1, \quad f(u_2)=v_4, \quad f(u_3)=v_3, \quad f(u_4)=v_2$



G



H

Example

- 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_4, v_3, v_2.

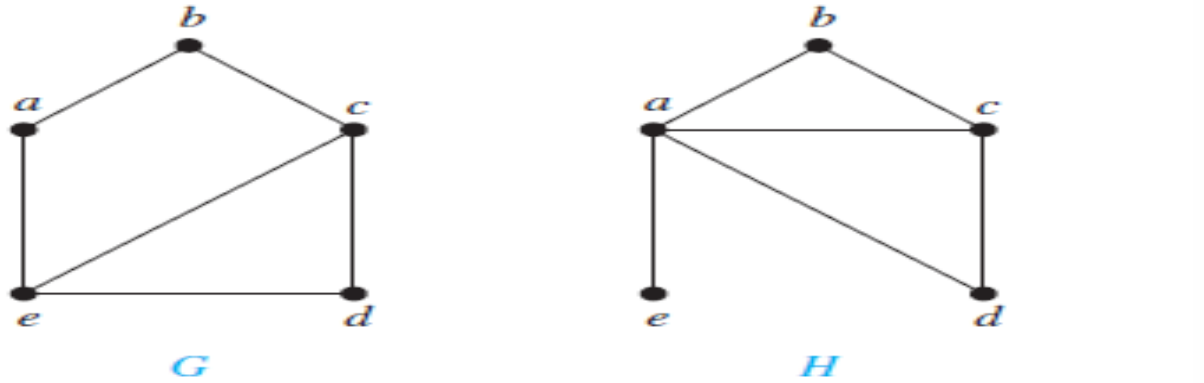
example

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

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

Example

- Show following graphs are not isomorphic



- Both *G* and *H* have five vertices and six edges. However, *H* has a vertex of degree one, namely, *e*, whereas *G* has no vertices of degree one. It follows that *G* and *H* are not 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 walk contains zero edges (has length zero)

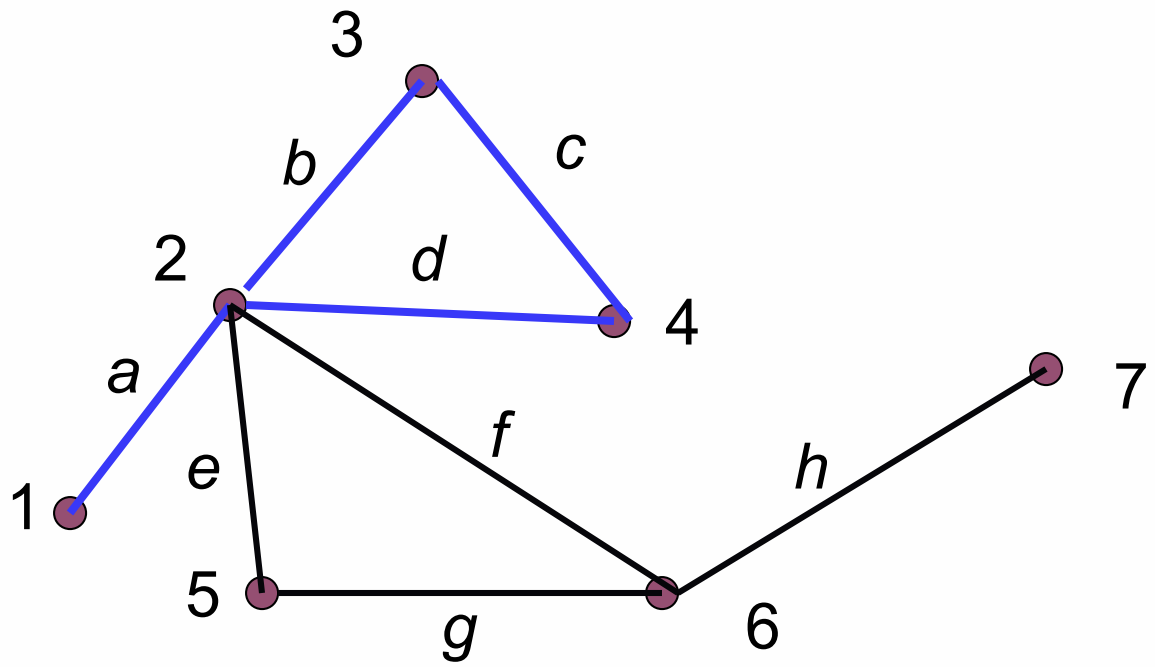
- The length of a walk is the number of edges it has.

Term and Description

- 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.

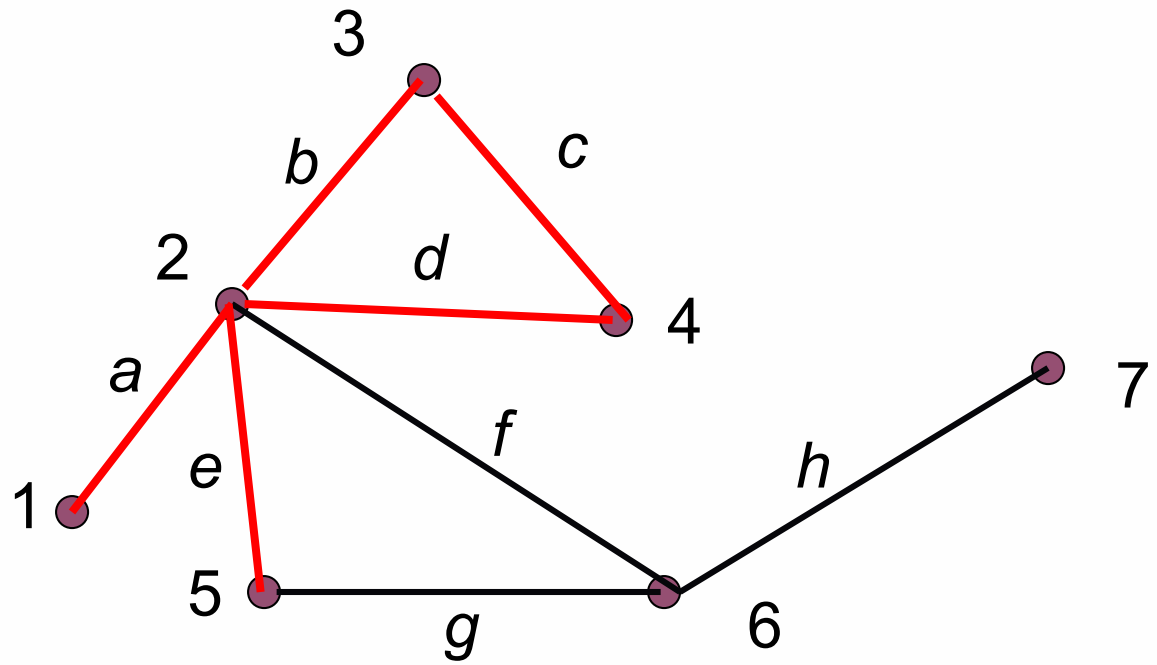
example

- (1, *a*, 2, *b*, 3, *c*, 4, *d*, 2) is a walk of length 4 from vertex 1 to vertex 2.



example

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

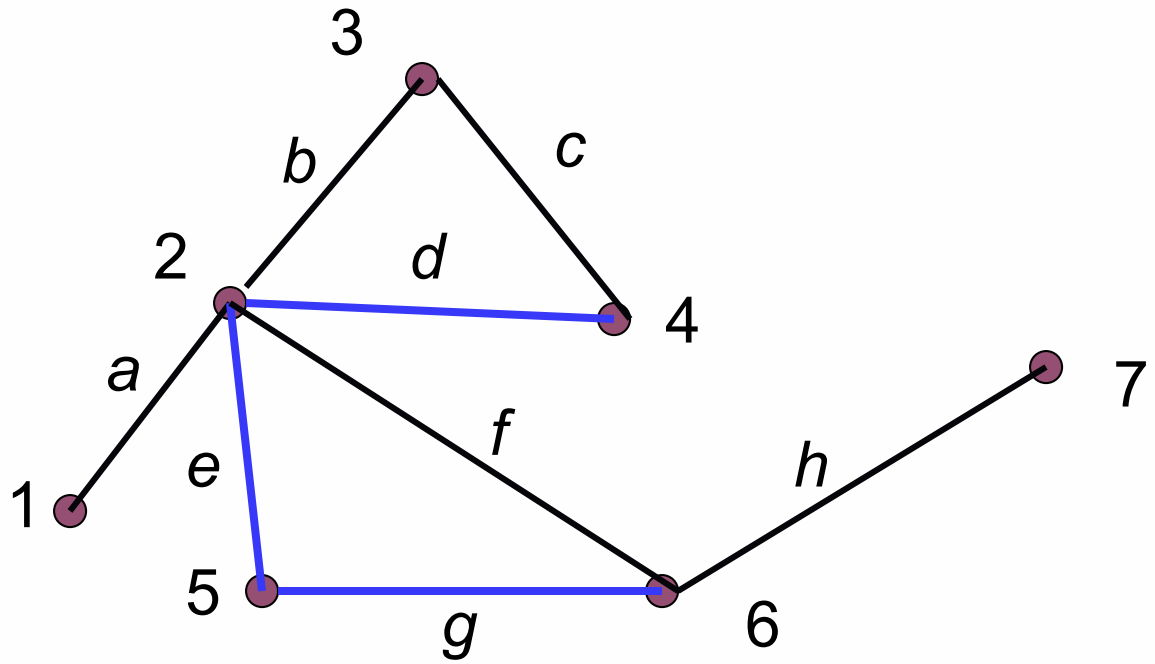


Note:

Trail: No repeated edge (can repeat vertex).

example

- (6, g, 5, e, 2, d, 4) is a path.

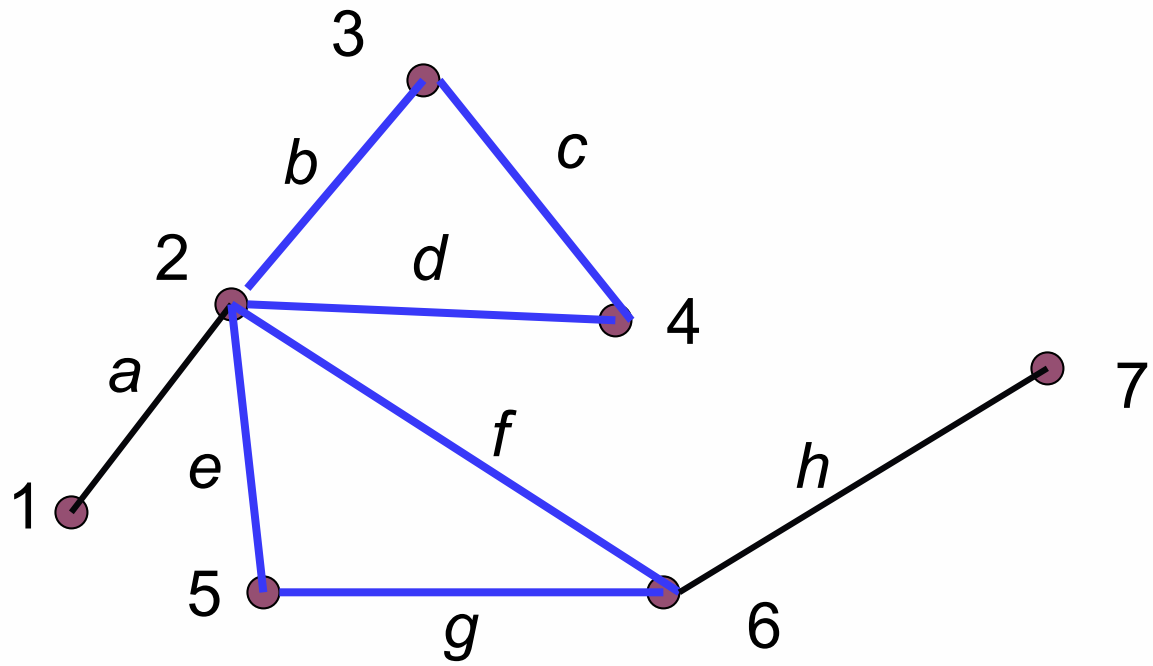


Note:

Path: No repeated vertex and edge.

example

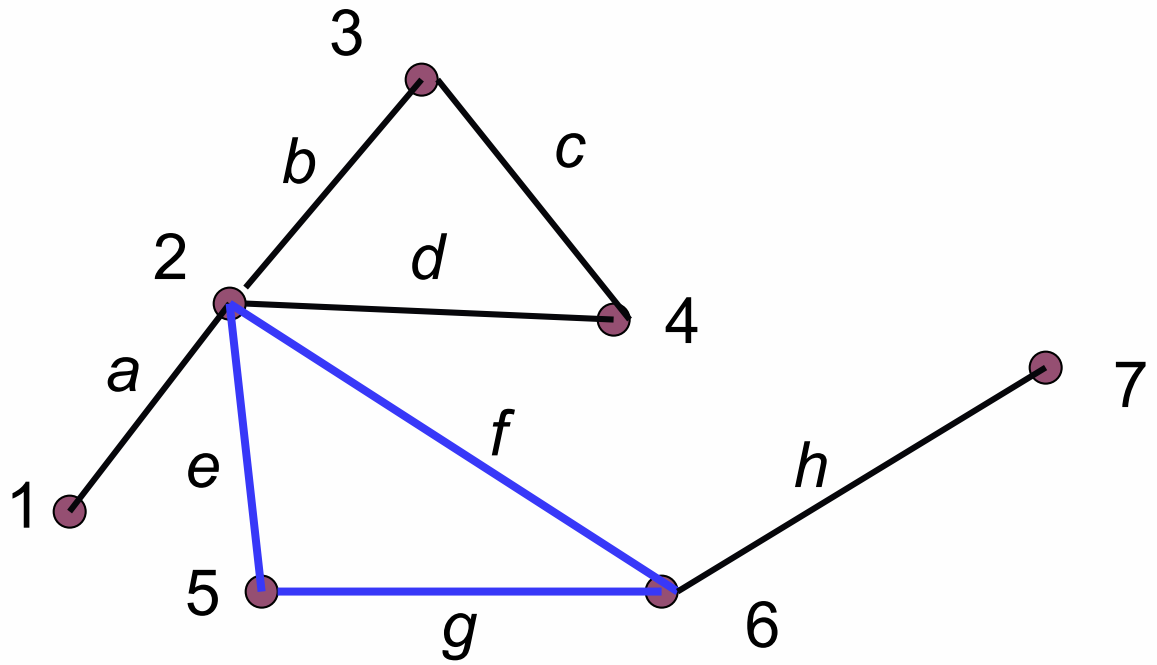
- (2, *f*, 6, *g*, 5, *e*, 2, *d*, 4, *c*, 3, *b*, 2) is a circuit/cycle.



Note: circuit → start and end at same vertex, no repeated edge.

example

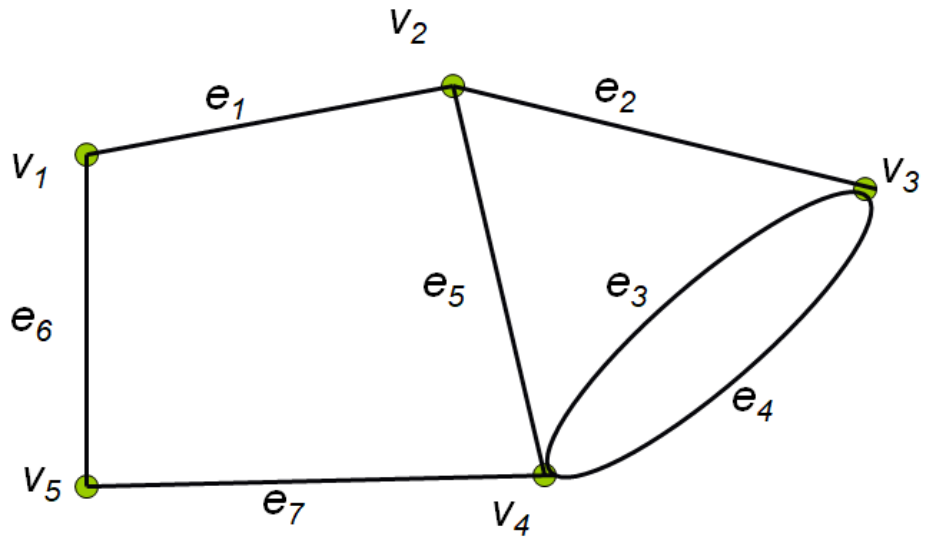
- (5, *g*, 6, *f*, 2, *e*, 5) is a simple circuit.



Note: Simple circuit →start and end at same vertex, no repeated edge or vertex except for the start and end vertex.

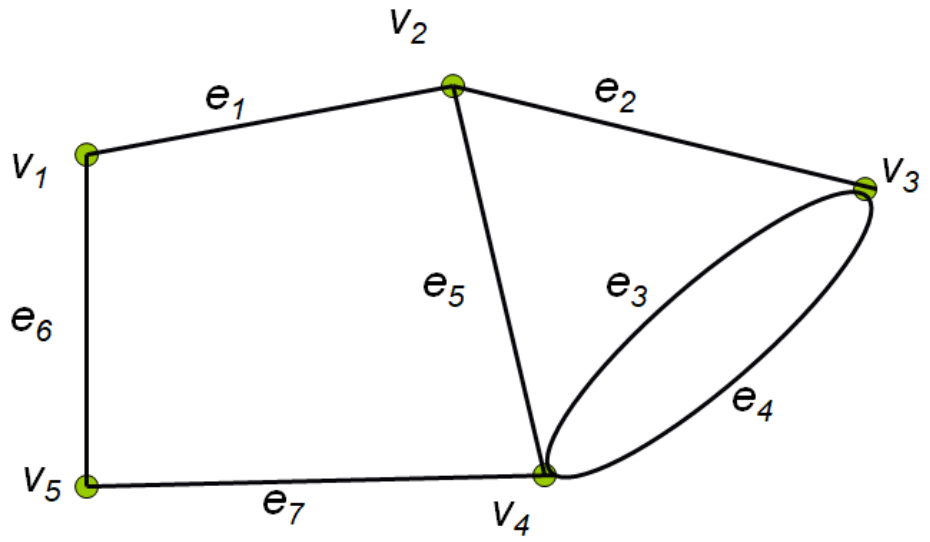
Try this

- Tell whether the following is either a walk, trail, path, circuit, simple circuit, closed walk or none of these.
  - (v_1, e_1, v_2)
  - $(v_2, e_2, v_3, e_3, v_4, e_4, v_3)$
  - $(v_4, e_7, v_5, e_6, v_1, e_1, v_2, e_2, v_3, e_3, v_4)$
  - $(v_4, e_4, v_3, e_3, v_4, e_5, v_2, e_1, v_1, e_6, v_5, e_7, v_4)$



Solution

- Tell whether the following is either a walk, trail, path, circuit, simple circuit, closed walk or none of these.
  - (v_1, e_1, v_2) Path
  - $(v_2, e_2, v_3, e_3, v_4, e_4, v_3)$ 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



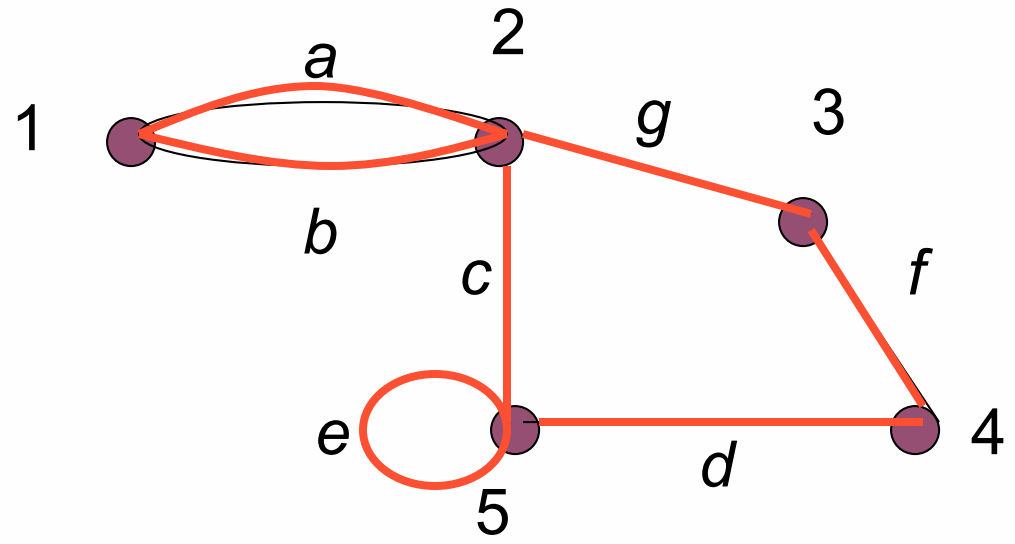
Euler Trail & Circuit

Euler Circuits

- A circuit in a graph that includes all the edges of the graph is called an Euler circuit.

- 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



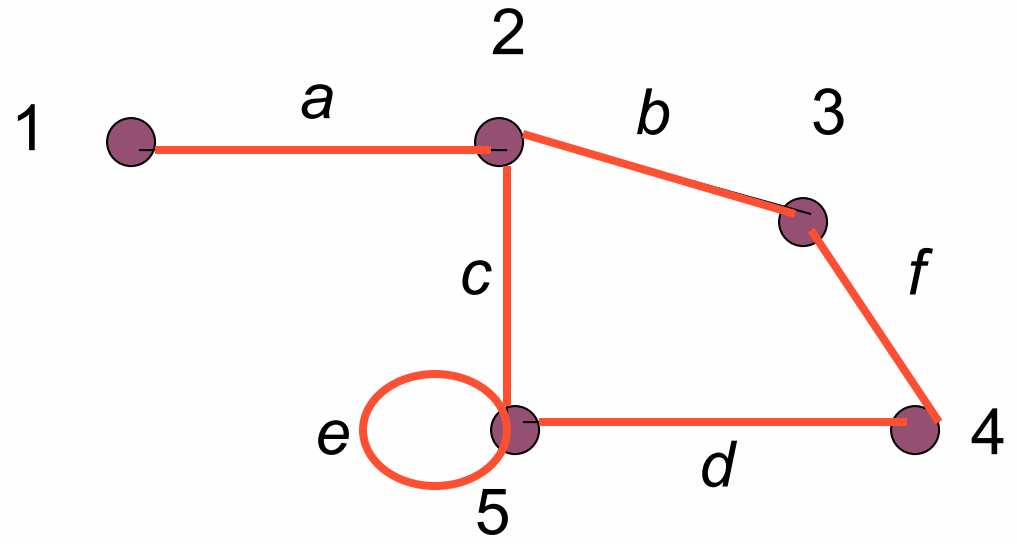
(1, *a*, 2, *c*, 5, *e*, 5, *d*, 4, *f*, 3, *g*, 2, *b*, 1)
is an Euler circuit

Euler Trail

- A trail from v to w ($v \neq w$) with no repeated edges is called an Euler trail if it **contains all the edges and all the vertices**.

- 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



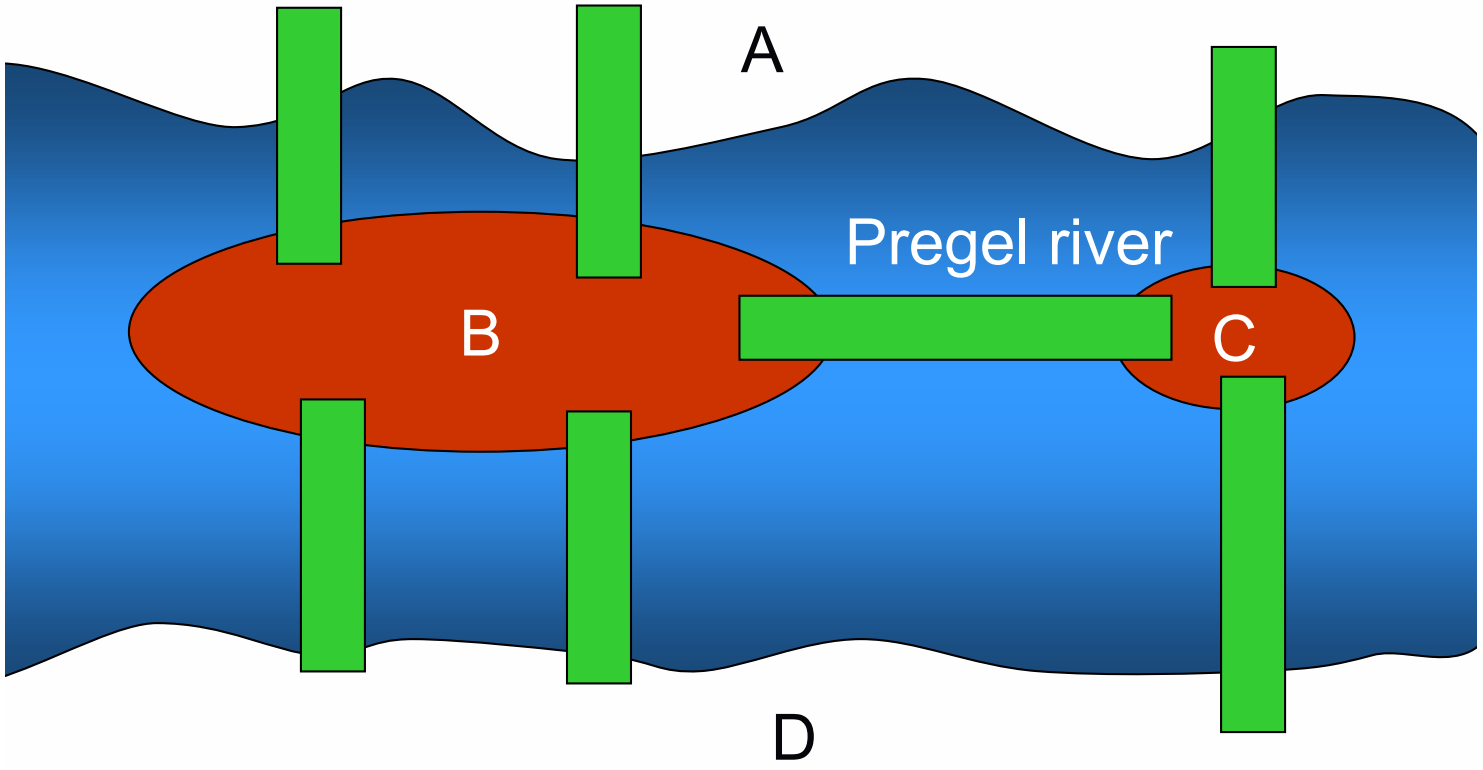
(1, *a*, 2, *c*, 5, *e*, 5, *d*, 4, *f*, 3, *b*, 2)
is an Euler trail

Theorem

- 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≠w) if and only if it is connected and ***v* and *w* are the only vertices having odd degree**.

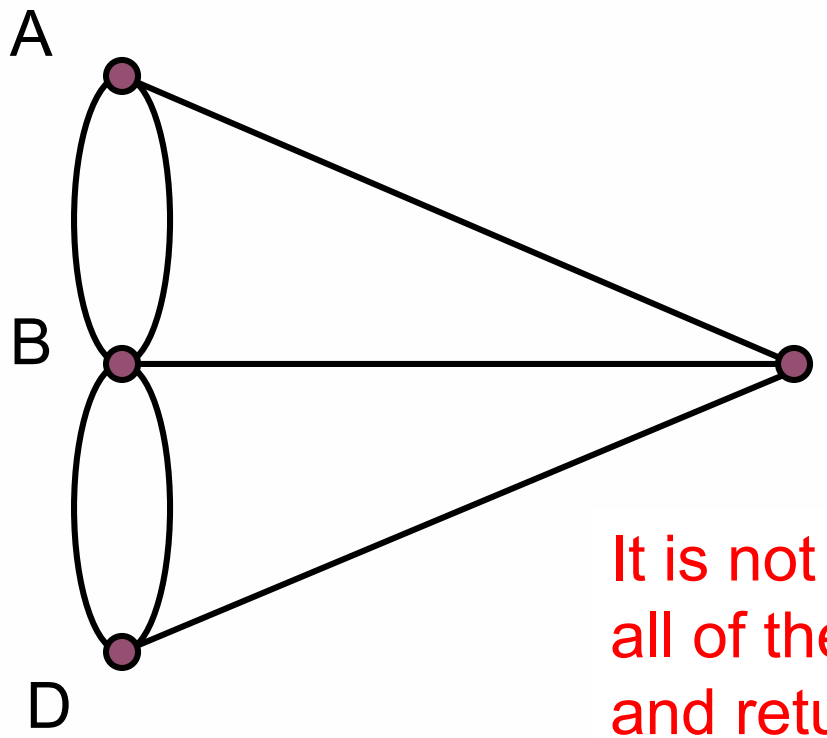
Königsberg Bridge Problem



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

Königsberg Bridge Problem

- Graph of the Königsberg Bridge Problem

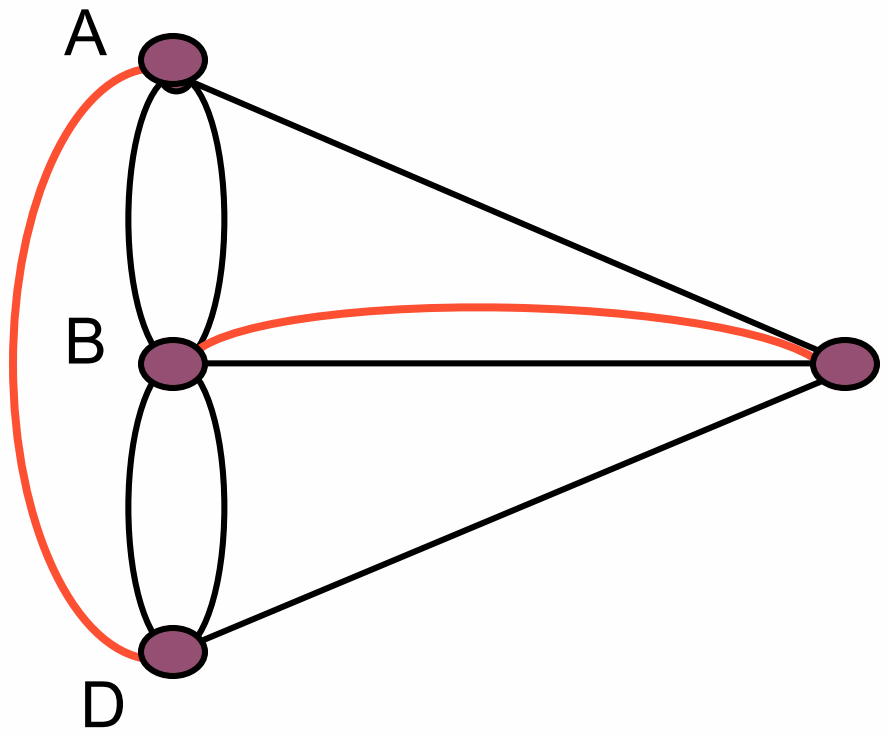


no Euler circuit

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

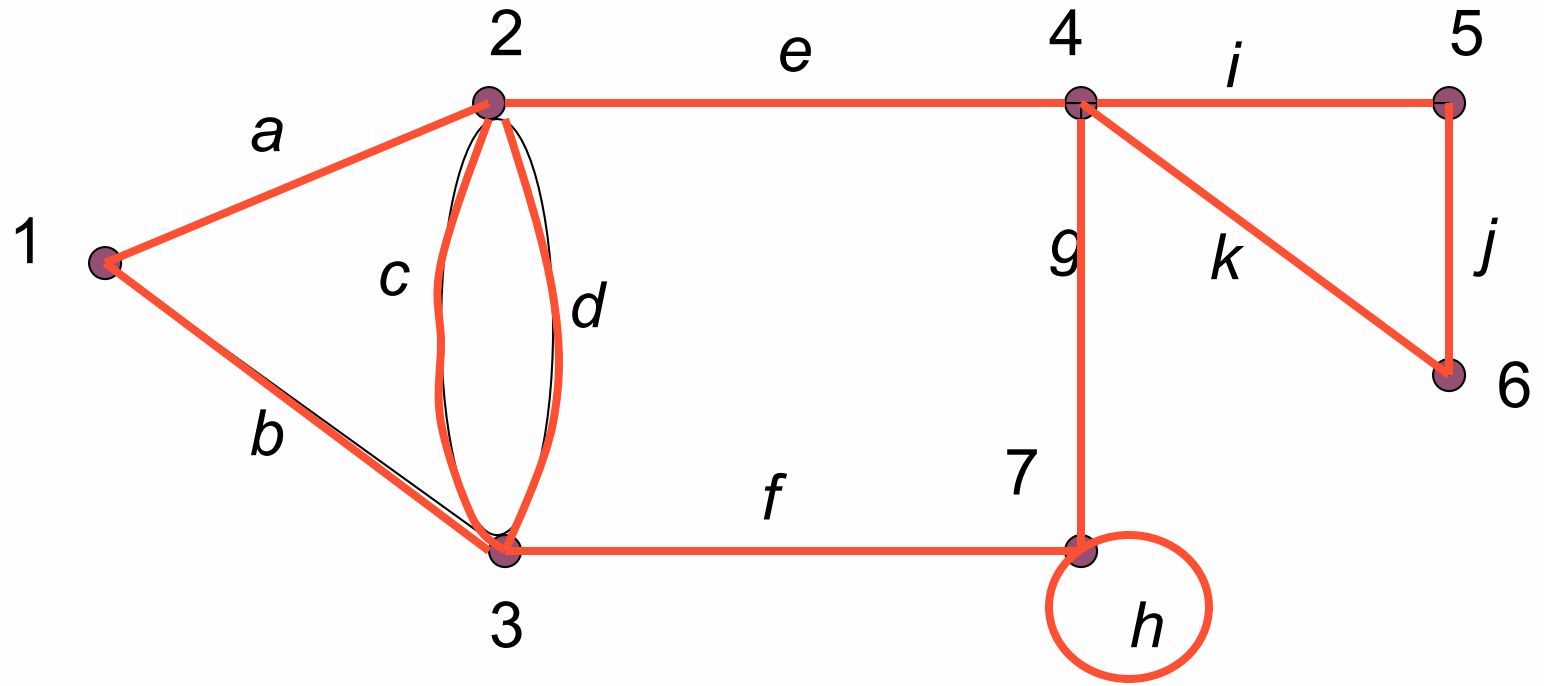
Königsberg Bridge Problem

- Since 1736, two additional bridges have been constructed on the Pregel river.



example

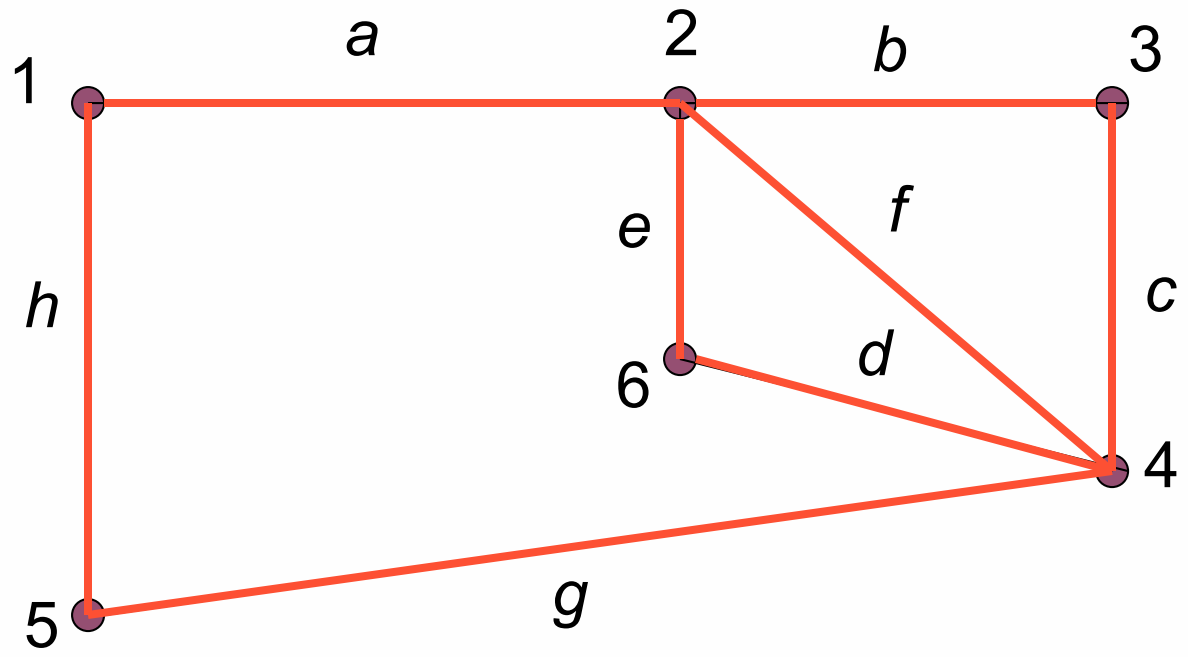
Vertex	1	2	3	4	5	6	7
Degree	2	4	4	4	2	2	4



This graph has an Euler circuit

example

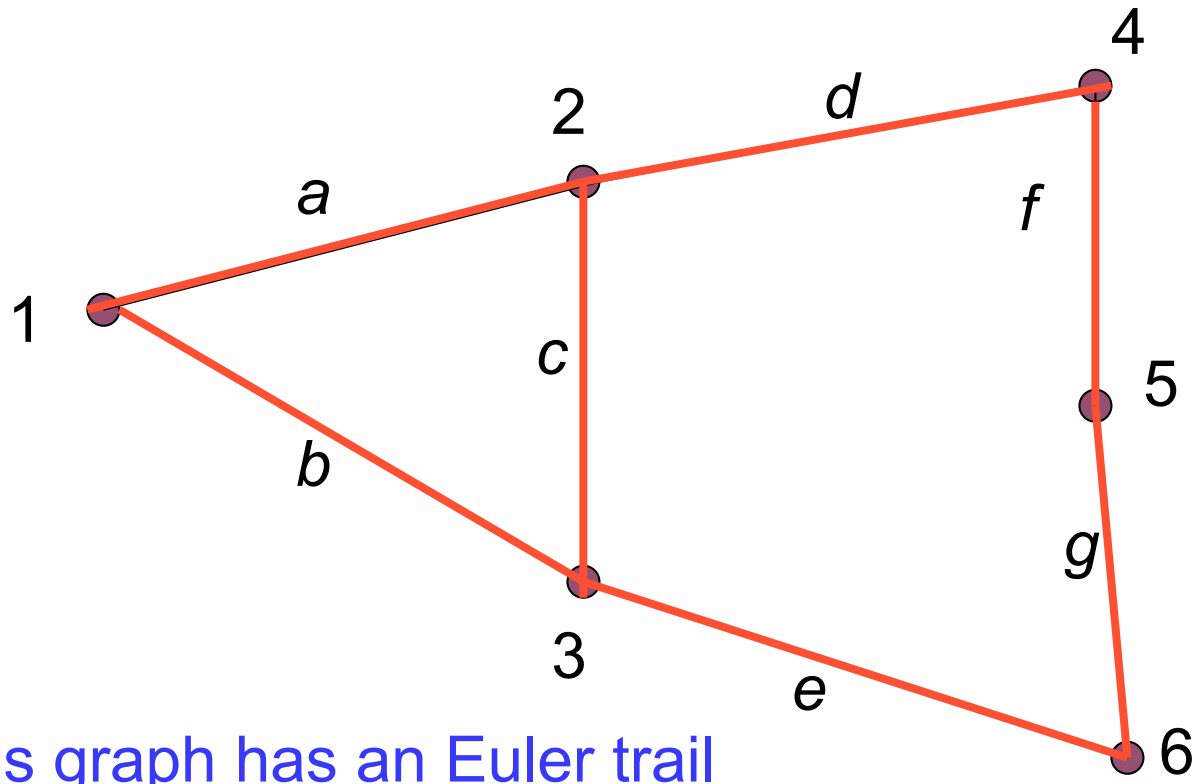
Vertex	1	2	3	4	5	6
Degree	2	4	2	4	2	2



This graph has an Euler circuit

example

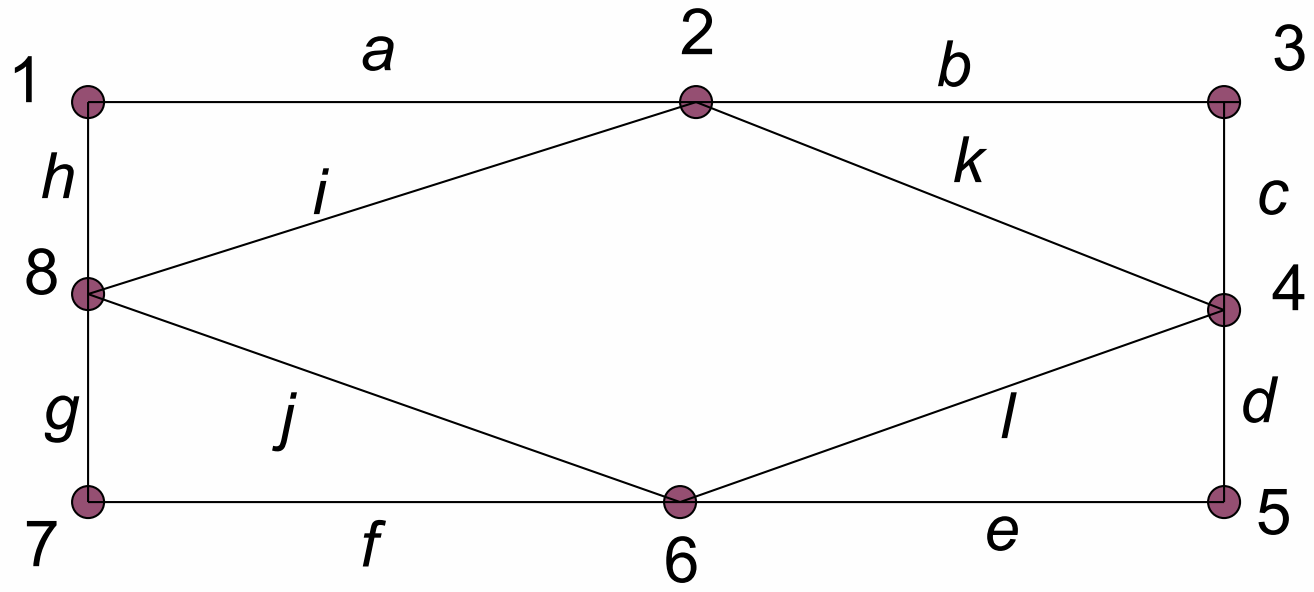
Vertex	1	2	3	4	5	6
Degree	2	3	3	2	2	2



This graph has an Euler trail

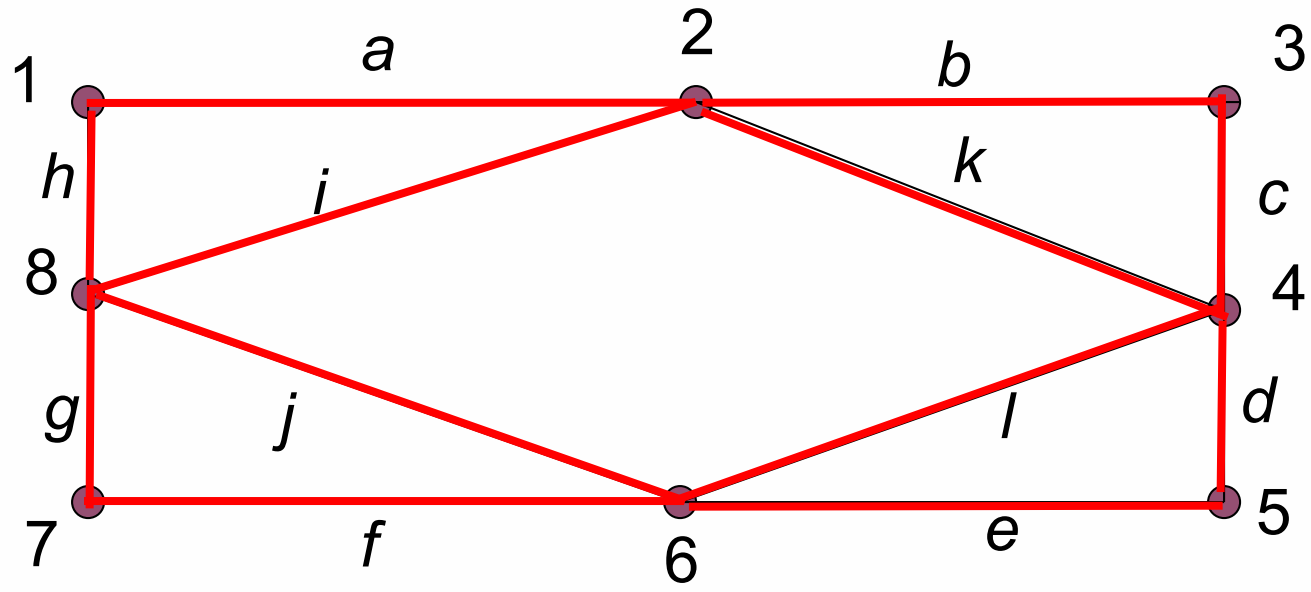
Try this

- Decide whether the graph has an Euler circuit. If the graph has an Euler circuit, exhibit one.



Solution

- Decide whether the graph has an Euler circuit. If the graph has an Euler circuit, exhibit one.

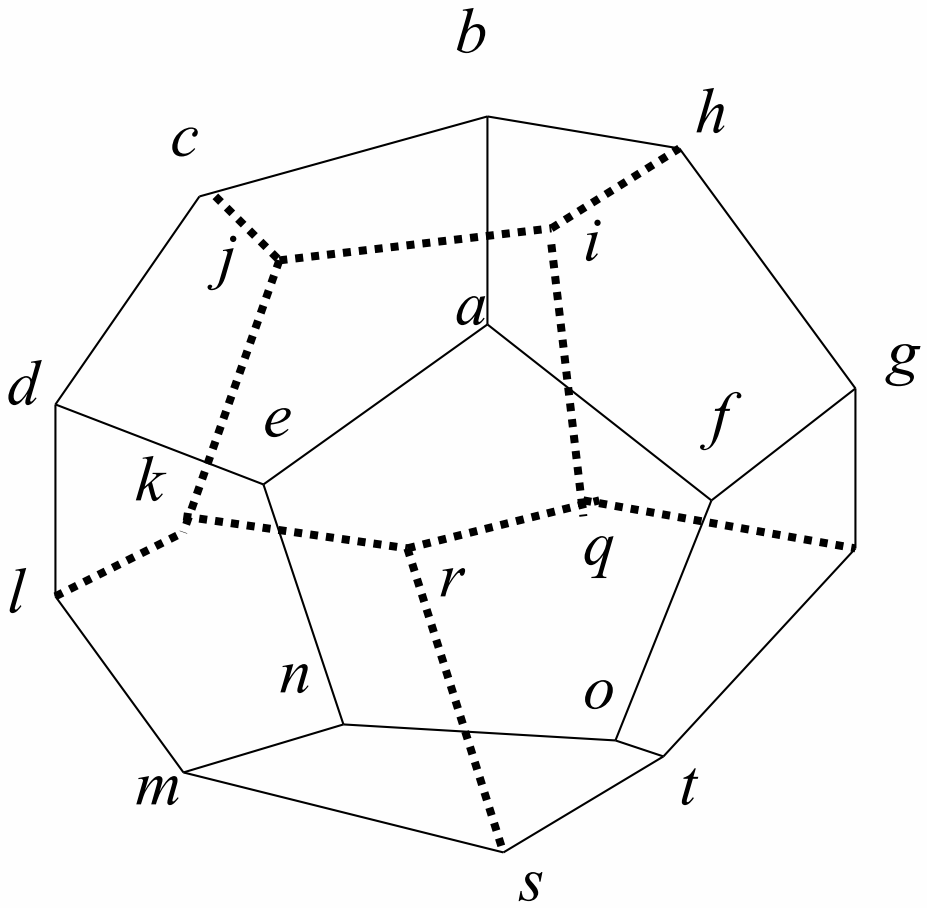


Hamiltonian Circuit

- A circuit in a graph *G* is called a Hamiltonian circuit if it contains each vertex of *G*.

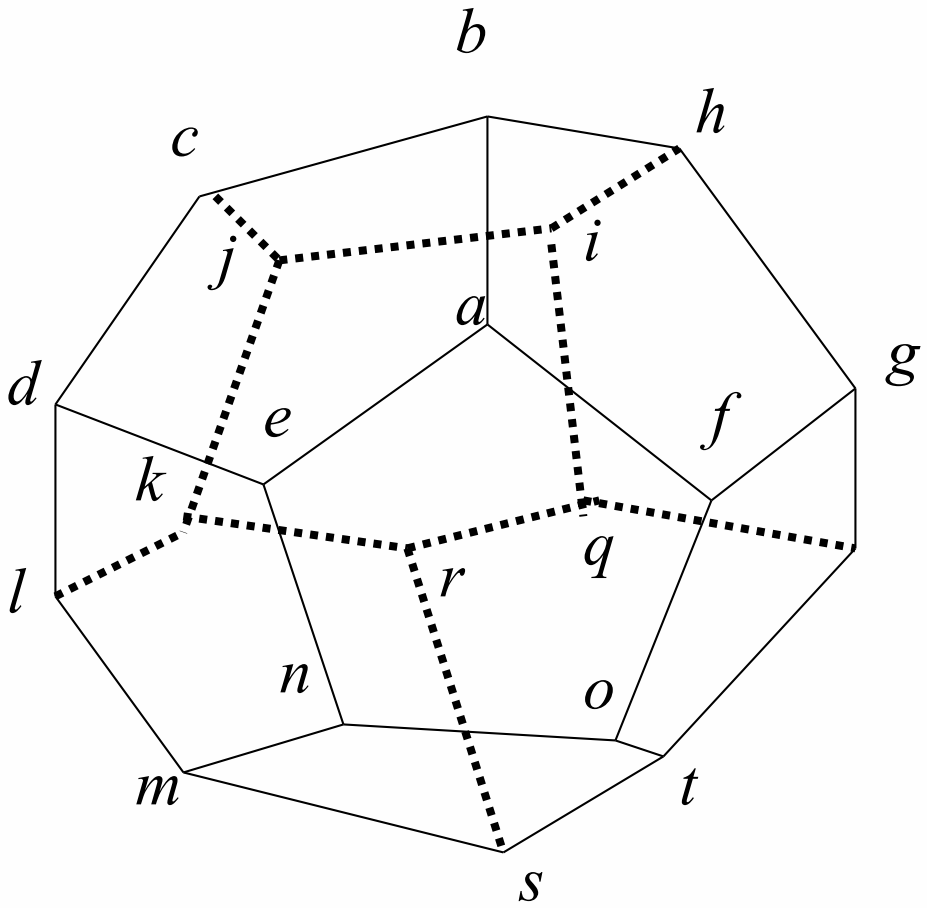
- Given a graph G, a Hamiltonian circuit for G is **a simple circuit that includes every vertex of G** (but doesn't need to include all edges). That is, a Hamiltonian circuit for G is a sequence of adjacent vertices and distinct edges in which every vertex of G appears exactly once, except for the first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

Around the world game



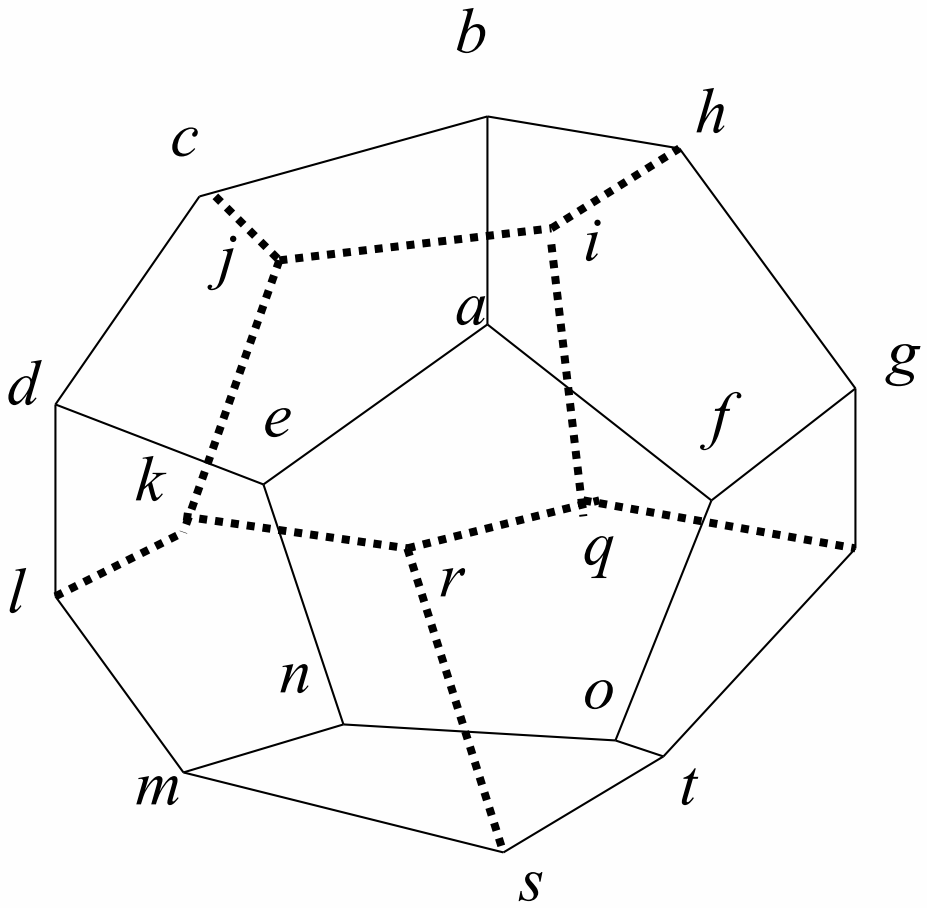
Sir William Rowan Hamilton marketed a puzzle in the mid-1800s in the form of dedocahedron

Around the world game



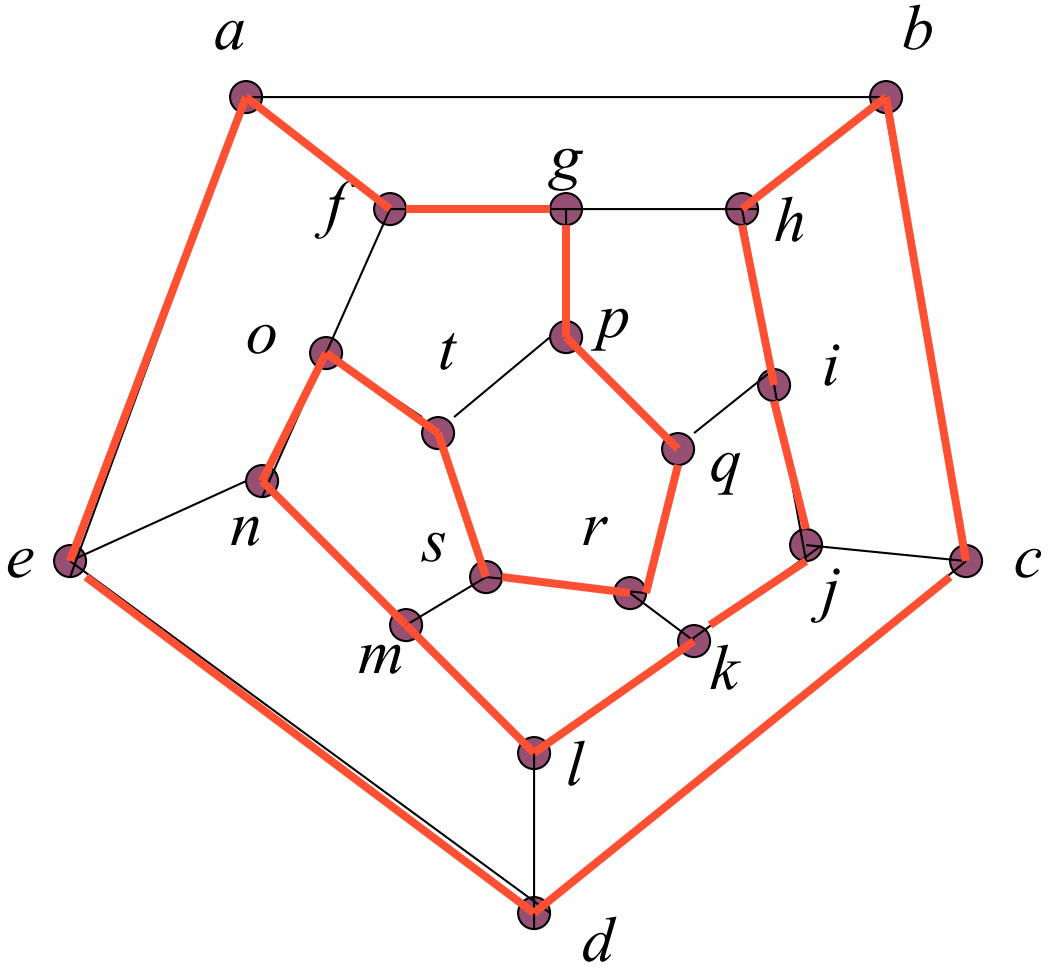
Each corner bore the name of a city

Around the world game

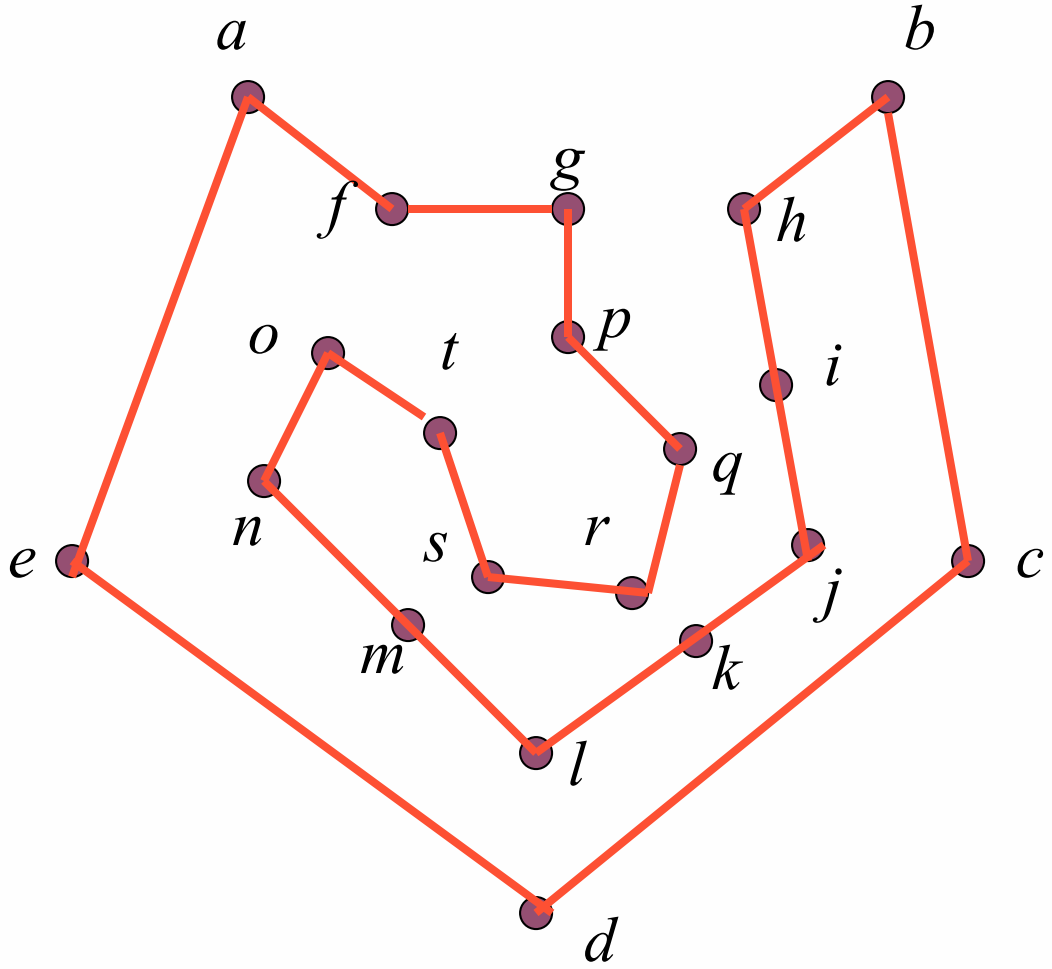


The problem was to start at any city, travel along the edges, visit each city exactly one time and return to the initial city

The graph



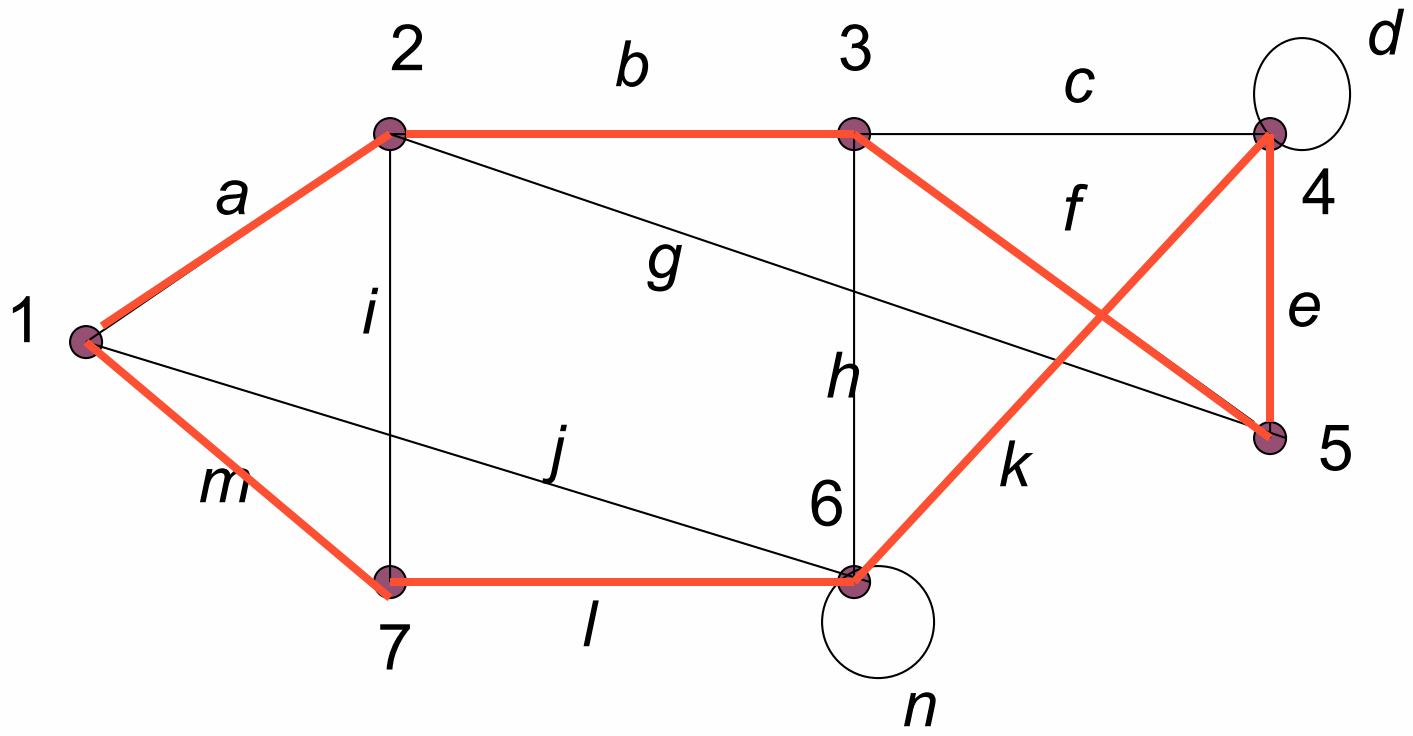
Hamiltonian Circuit



a-f-g-p-q-r-s-t-o-n-m-l-k-j-i-h-b-c-d-e-a

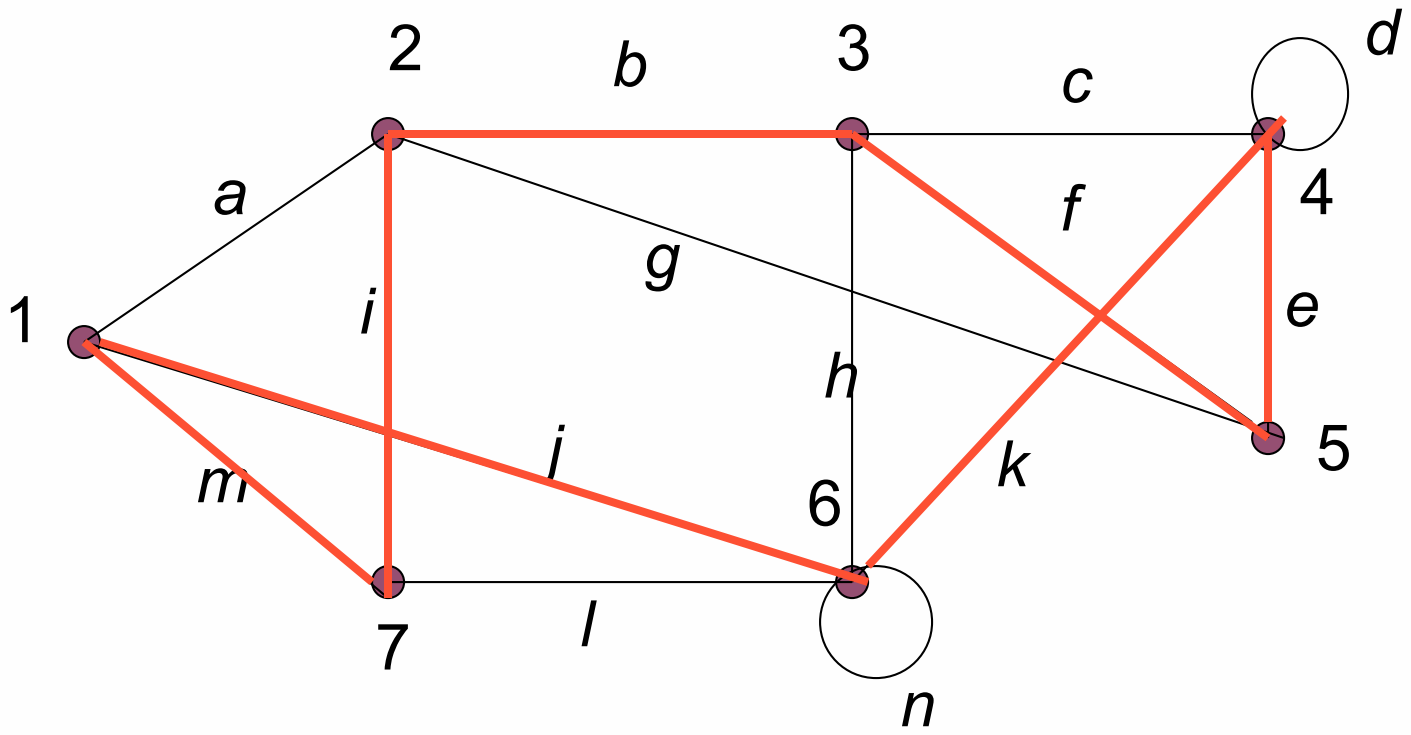
example

This graph has a Hamiltonian circuit



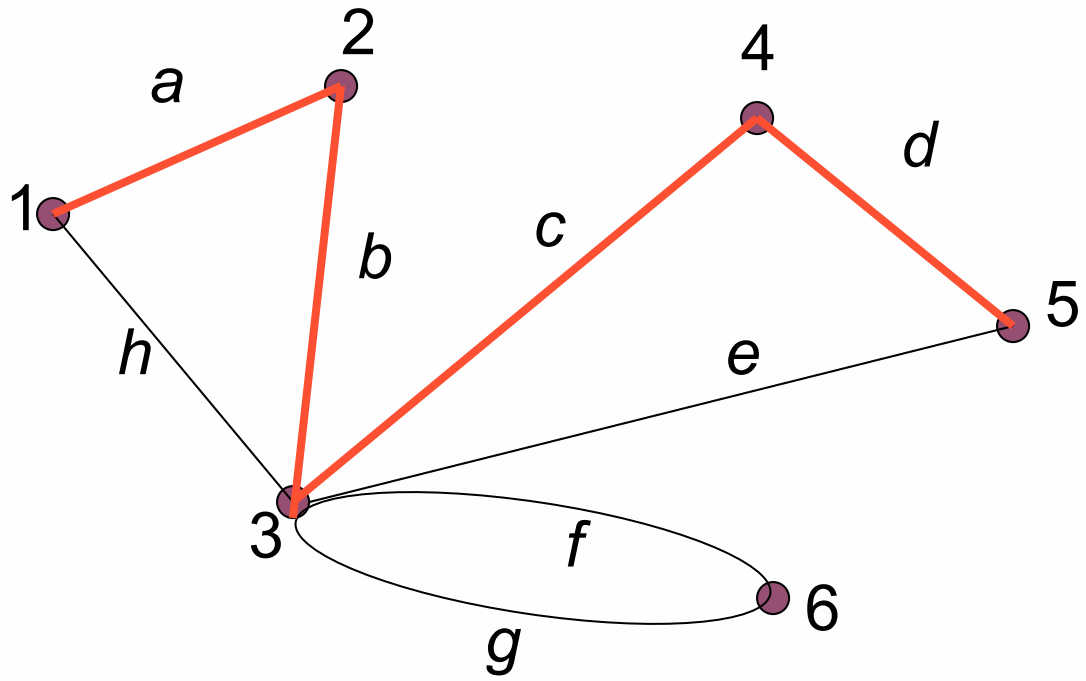
1-a-2-b-3-f-5-e-4-k-6-l-7-m-1

example



1-j-6-k-4-e-5-f-3-b-2-i-7-m-1

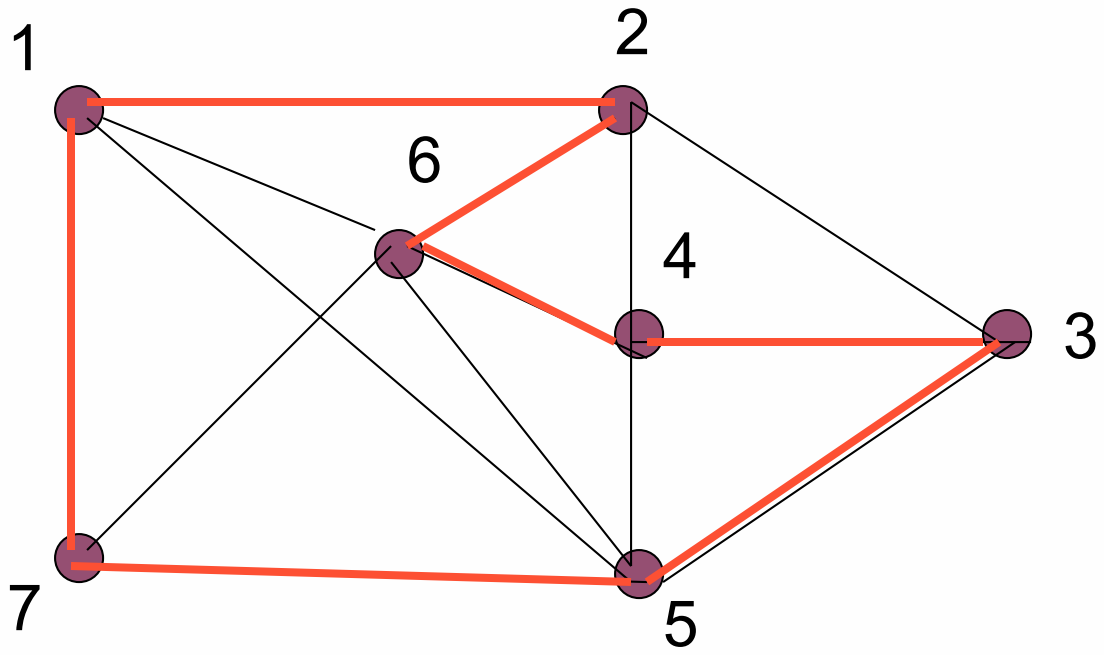
example



no Hamiltonion circuit

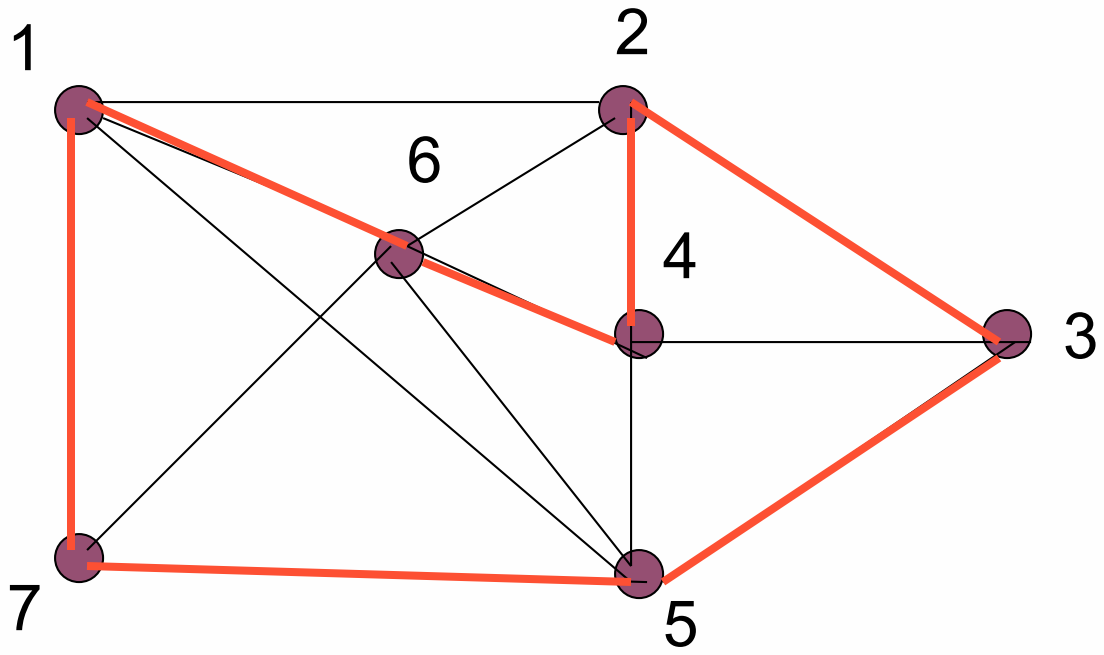
example

This graph has a Hamiltonian circuit



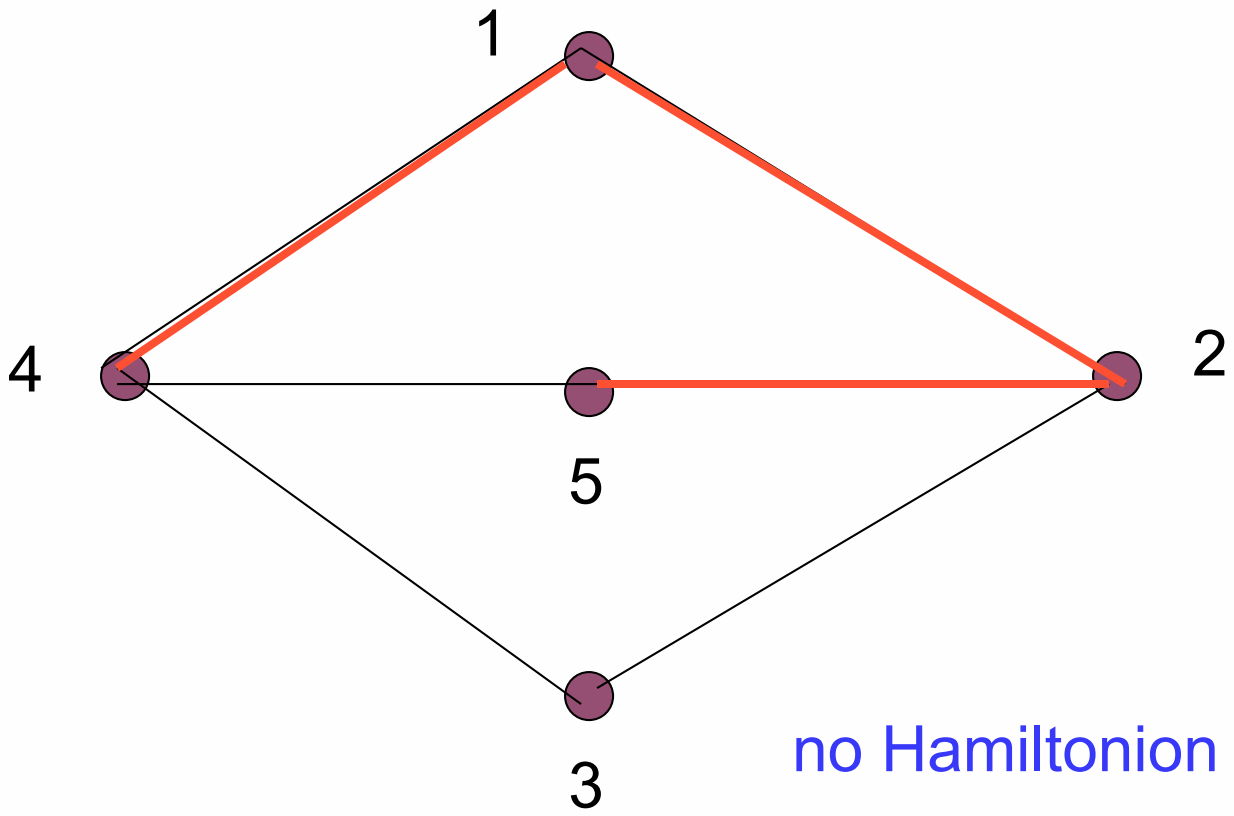
1-2-6-4-3-5-7-1

example



1-6-4-2-3-5-7-1

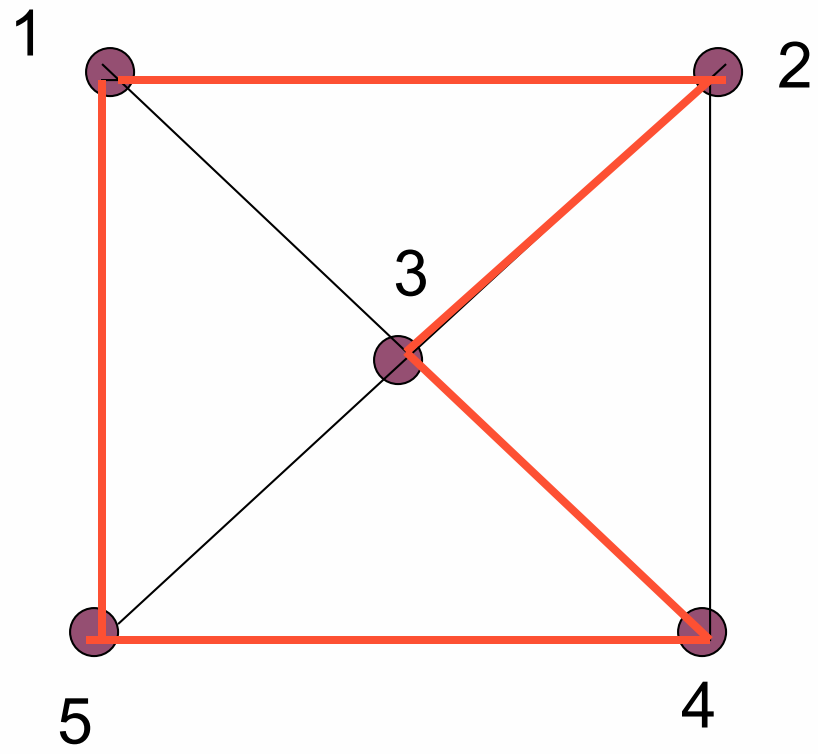
example



no Hamiltonion circuit

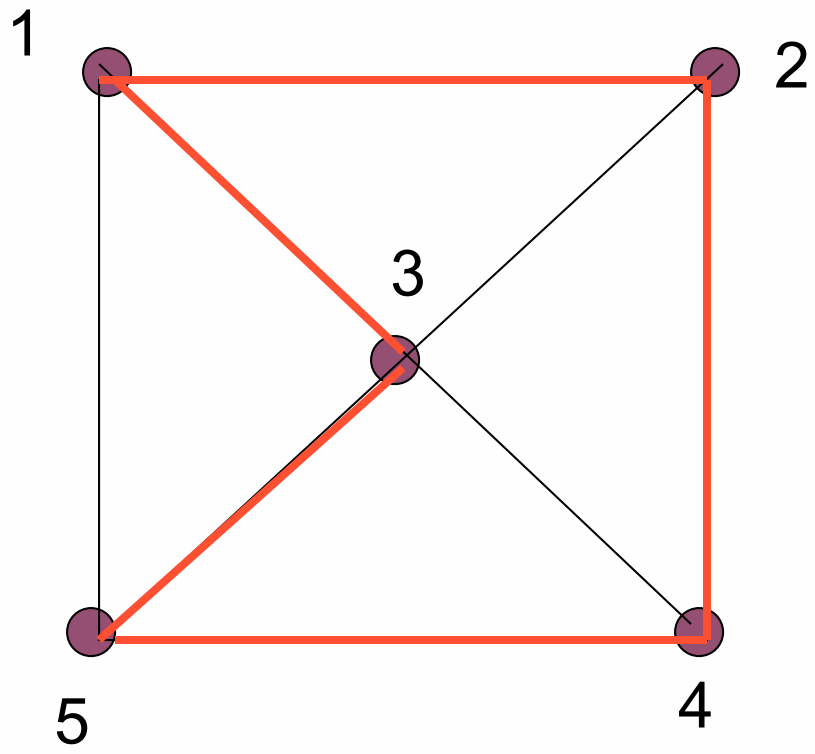
example

This graph has a Hamiltonian circuit



1-2-3-4-5-1

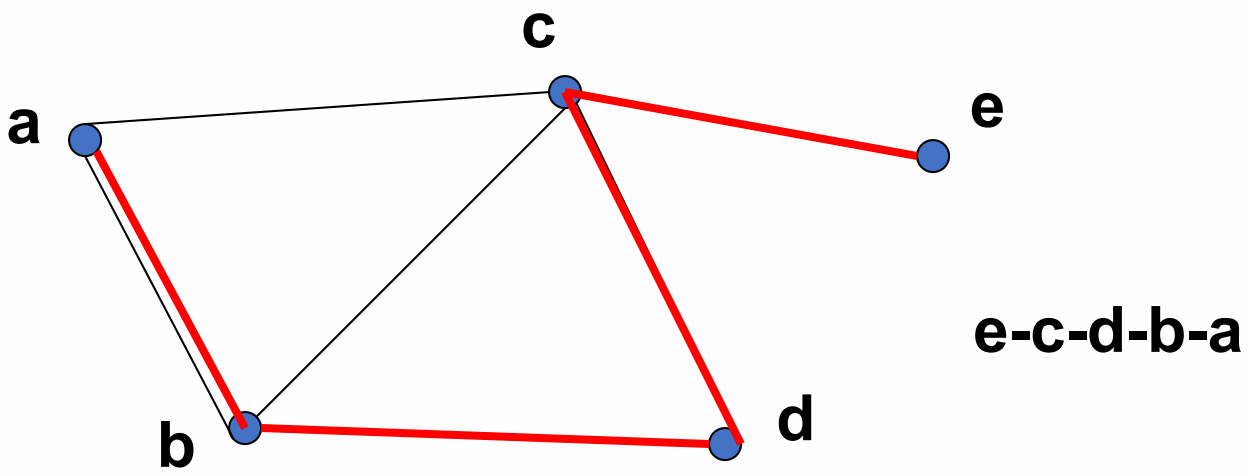
example



3-5-4-2-1-3

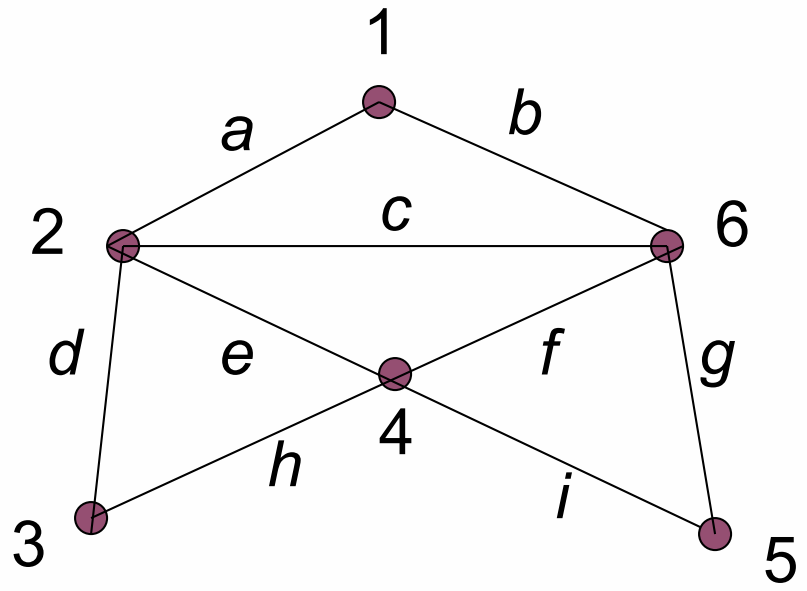
Hamiltonian Path

- A path in a graph *G* is called a Hamiltonian path if it contains each vertex of *G*.
- Example:



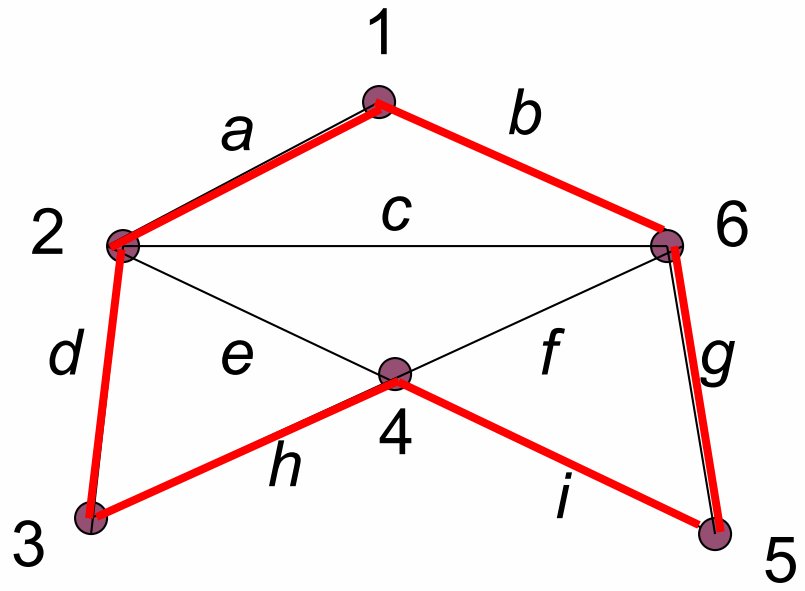
Try this

- Find a Hamiltonian circuit in this graph.



Solution

- Find a Hamiltonian circuit in this graph.

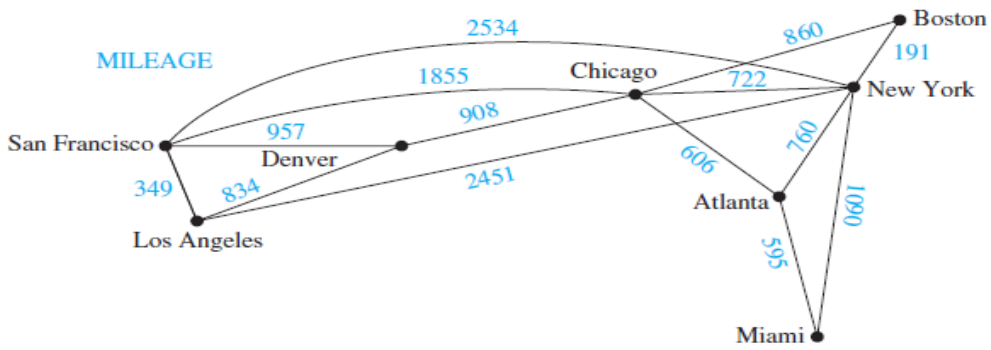


1-b-6-g-5-i-4-h-3-d-2-a-1

Shortest Path Problem

Shortest Path Problem

- Many problems can be modeled using graphs with weights assigned to their edges.
- For example, problems involving distances can be modeled by assigning distances between cities to the edges.

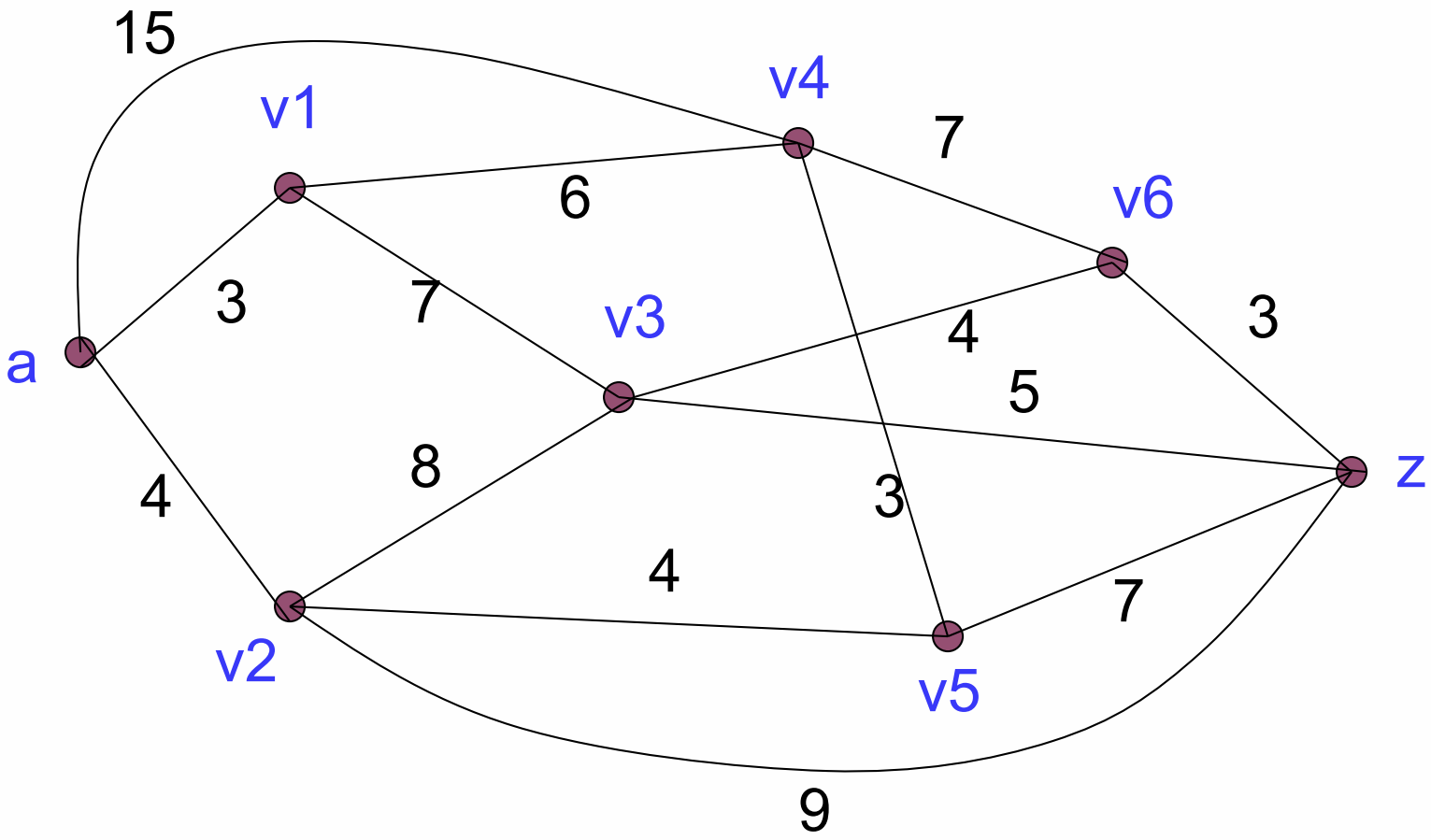


- Given a weighted graph, find the minimum-cost path between two cities?

Shortest Path Problems

- 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.

example



Dijkstra's Shortest Path Algorithm

1. $S := \varnothing$
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 $S := S \cup \{v\}$

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

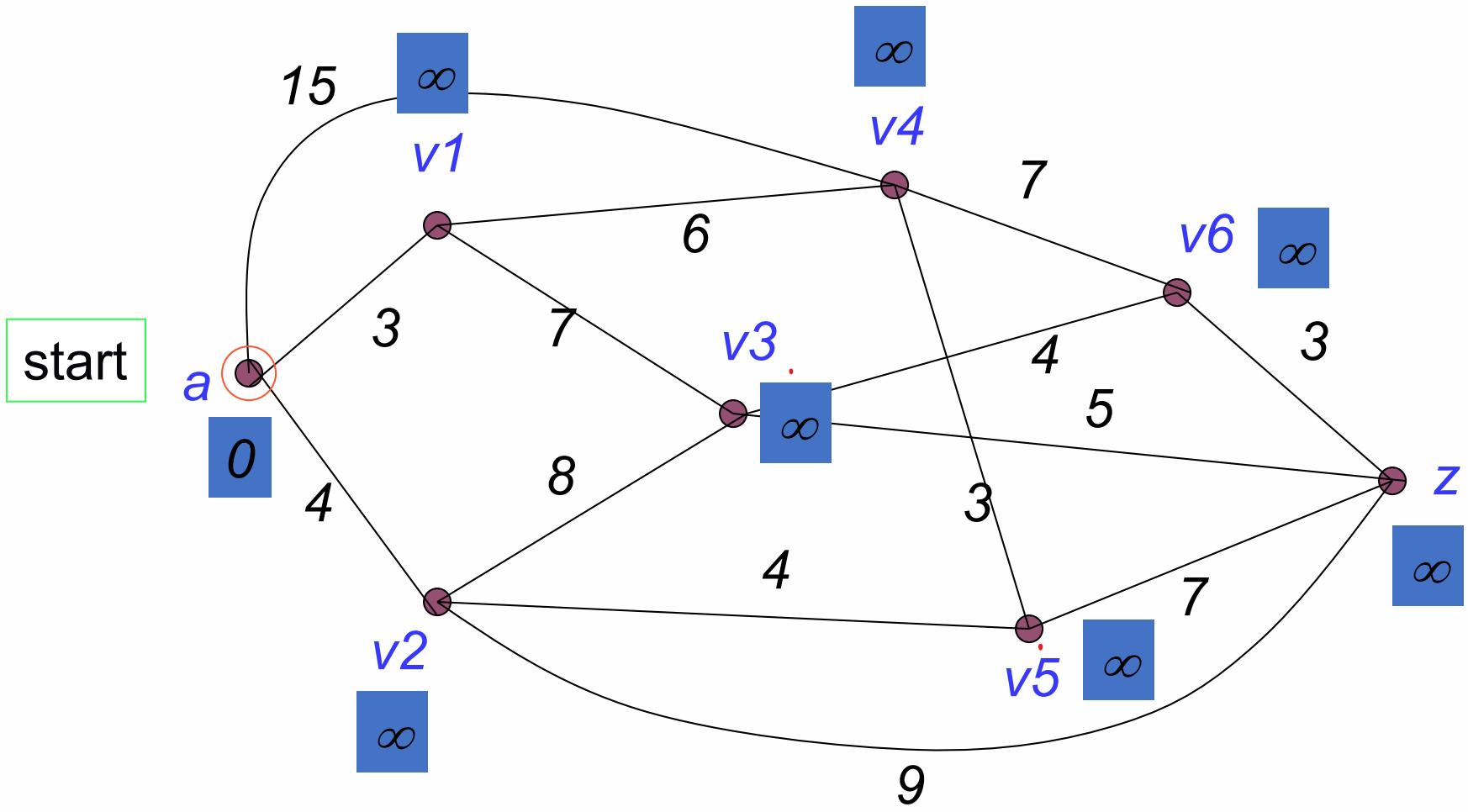
   5.d For all $w \in N$ such that there is an edge from v to w

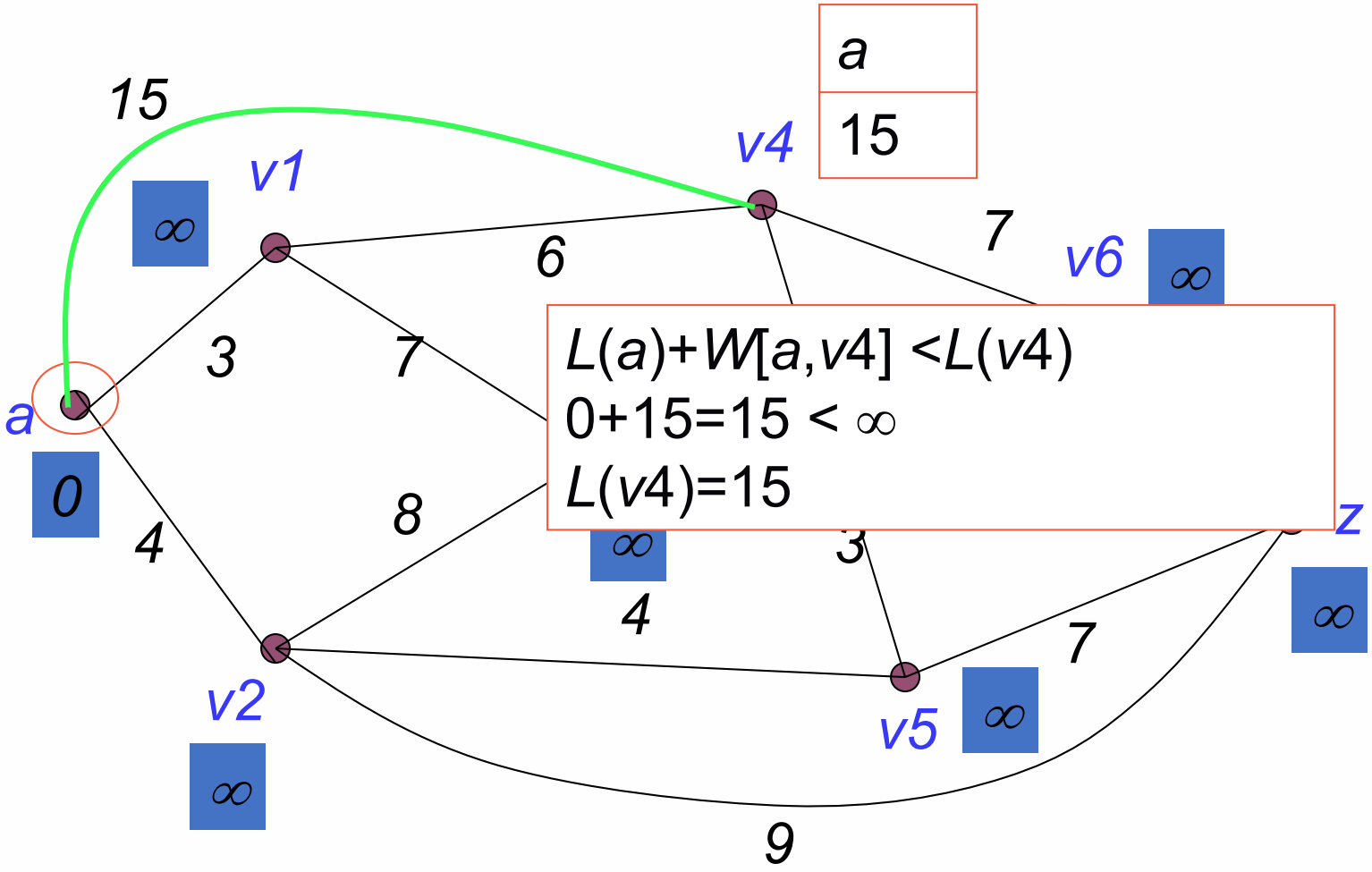
   5.d.1 if $L(v) + W[v,w] < L(w)$ then

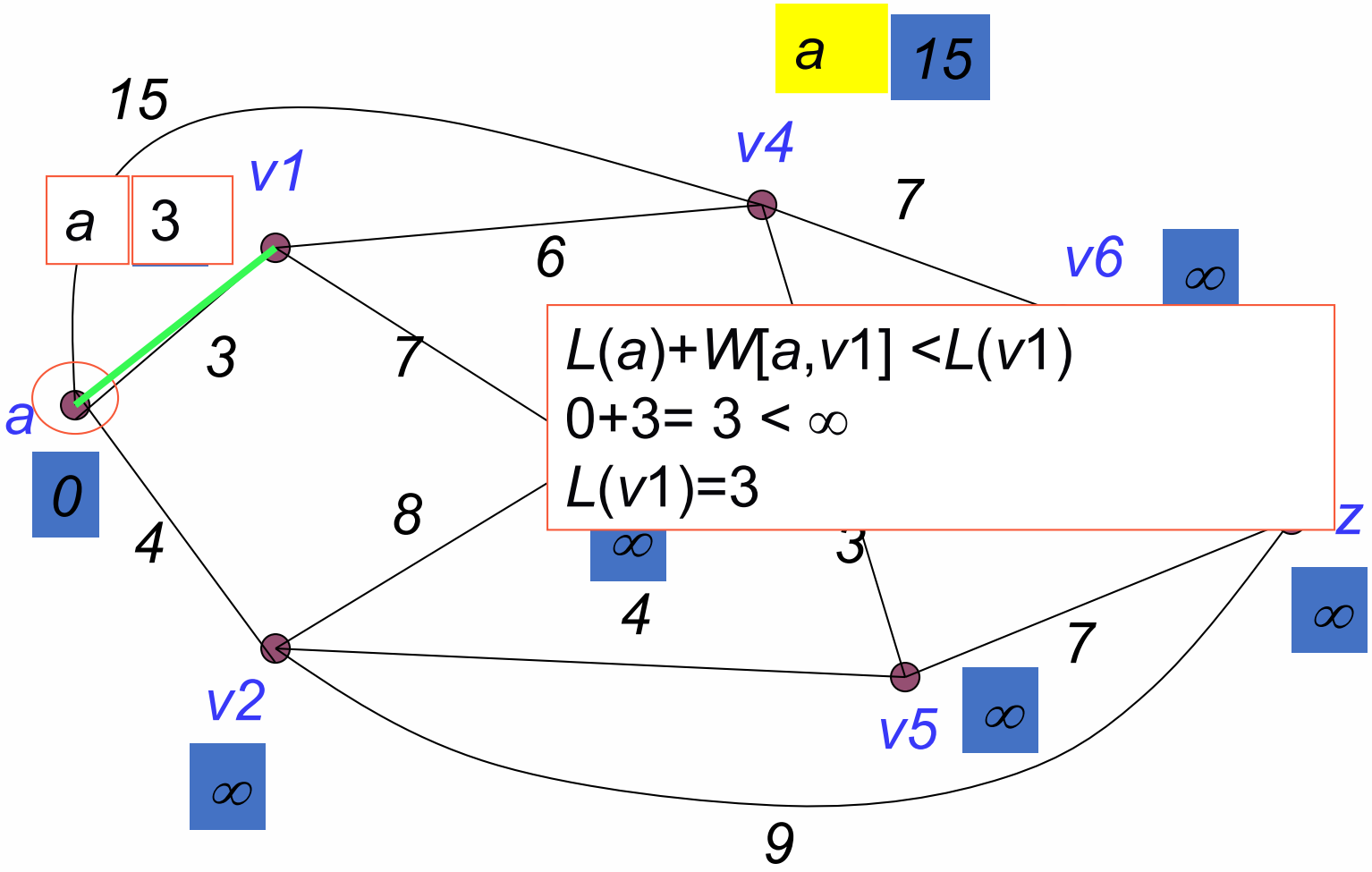
   $L(w) = L(v) + W[v,w]$

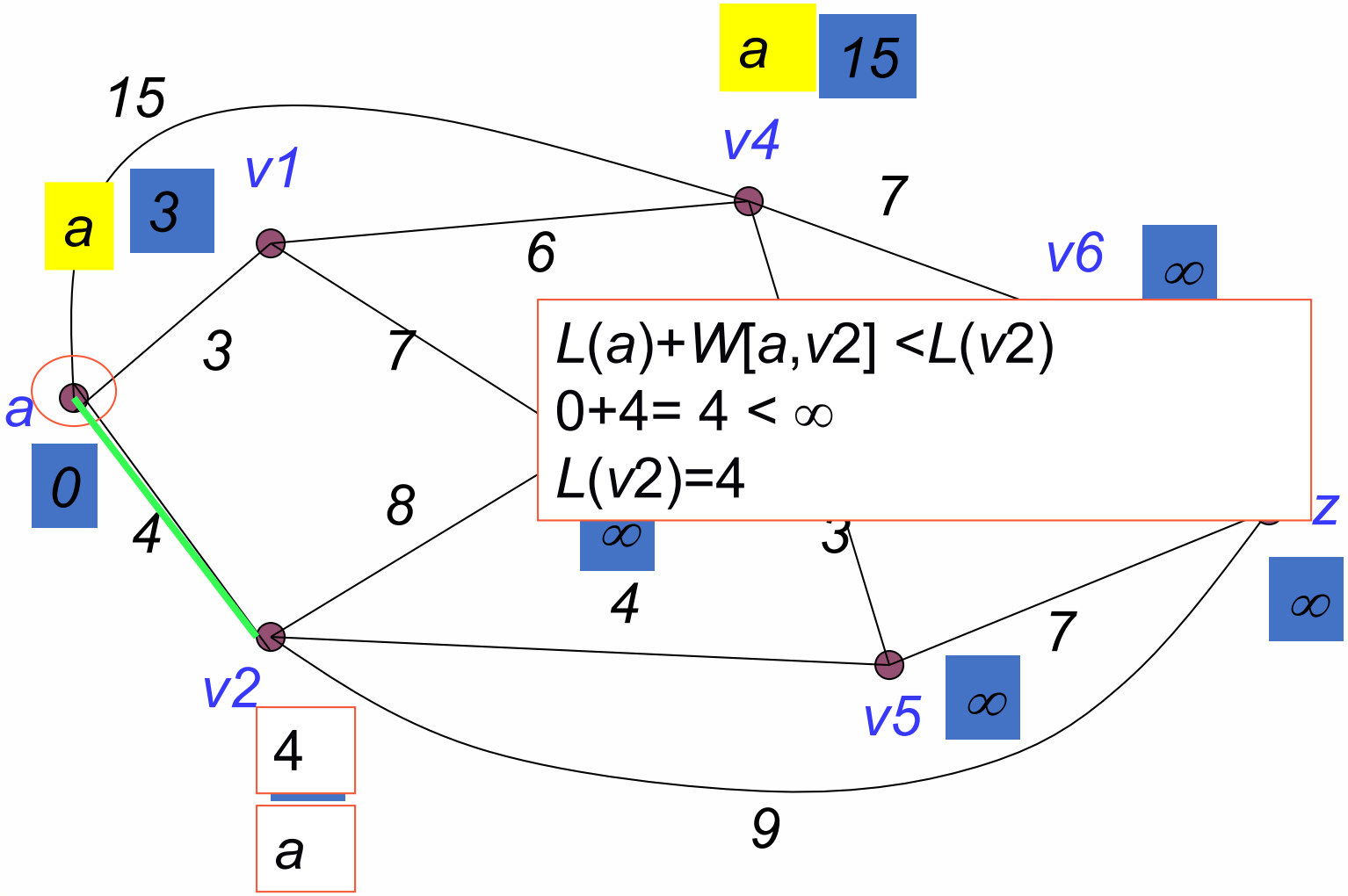
$S=\emptyset$

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




a
15
15
v4
v1
∞
6
7
v6
∞
3
7
L(a)+W[a,v4] <L(v4)
0+15=15 < ∞
L(v4)=15
a
0
4
8
∞
4
3
z
∞
7
v2
∞
v5
∞
9

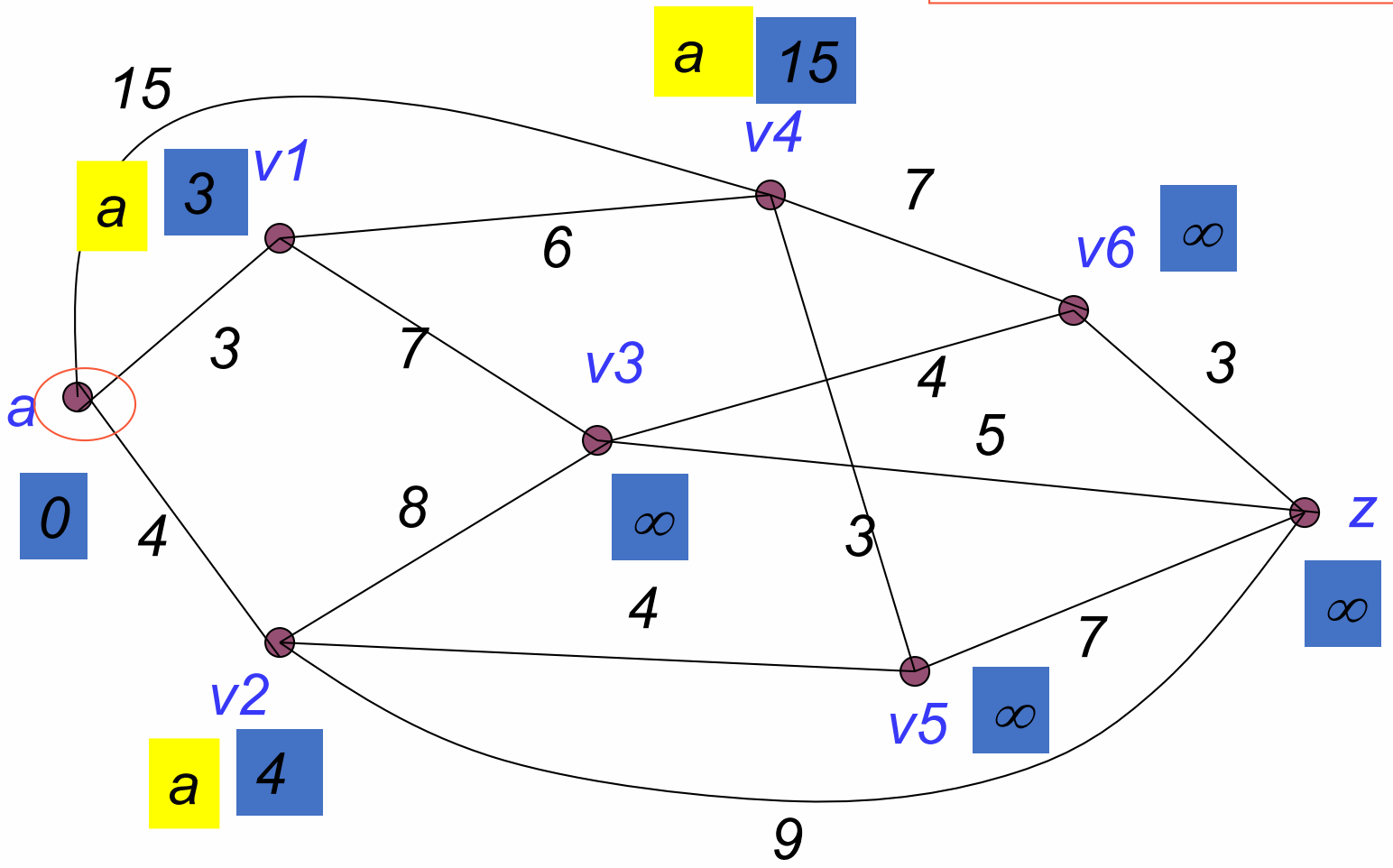

a
15
15
v4
v1
a
3
7
6
v6
∞
L(a)+W[a,v1] <L(v1)
0+3= 3 < ∞
L(v1)=3
3
7
a
0
8
4
∞
4
3
z
∞
7
v2
∞
v5
∞
9


a
15
15
v4
v1
a
3
7
6
v6
∞
L(a)+W[a,v2] <L(v2)
0+4= 4 < ∞
L(v2)=4
3
7
a
0
8
z
4
∞
3
4
7
∞
v2
v5
∞
4
a
9

$S=\{a\}$

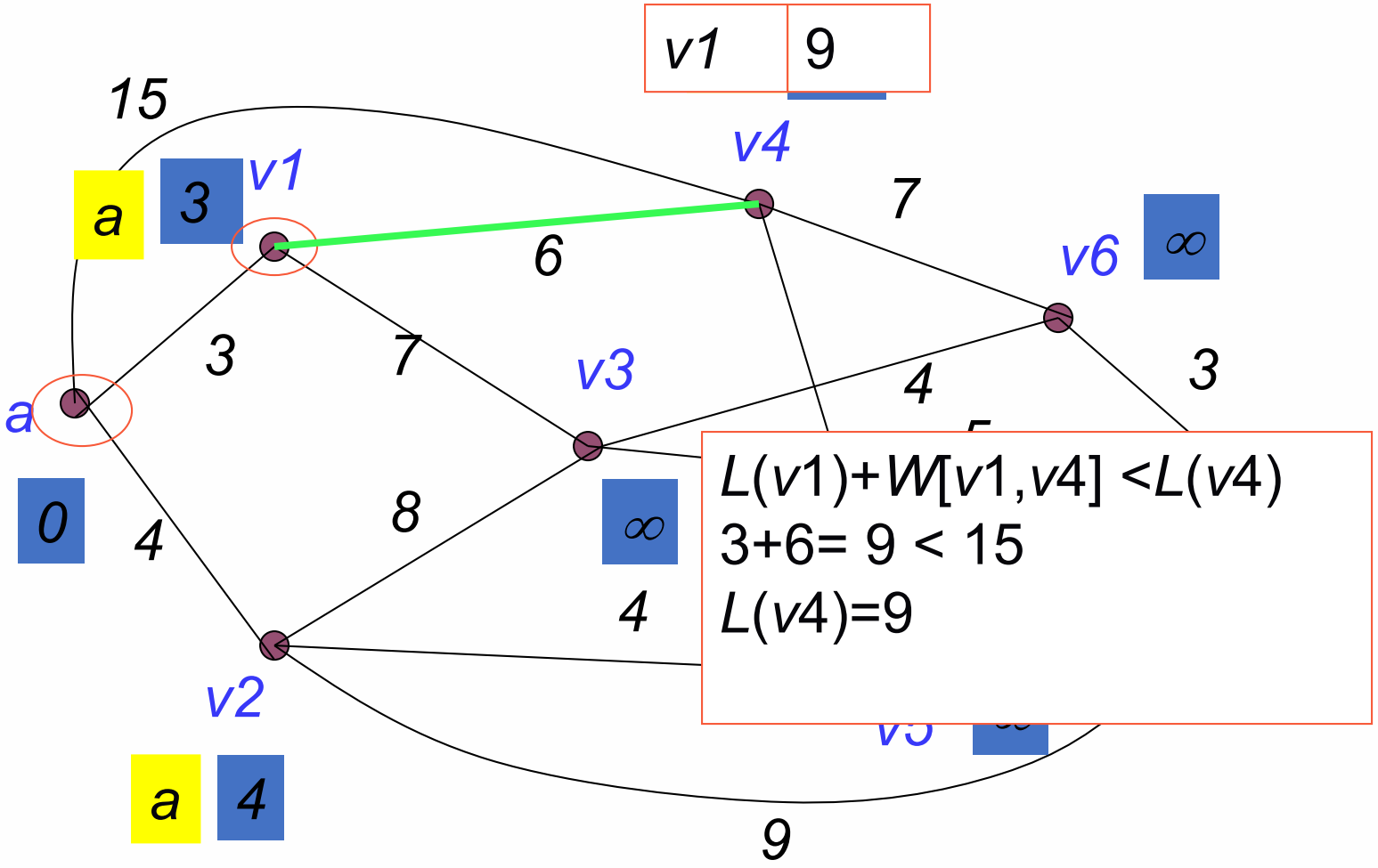
$N=\{v1,v2,v3,v4,v5,$

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



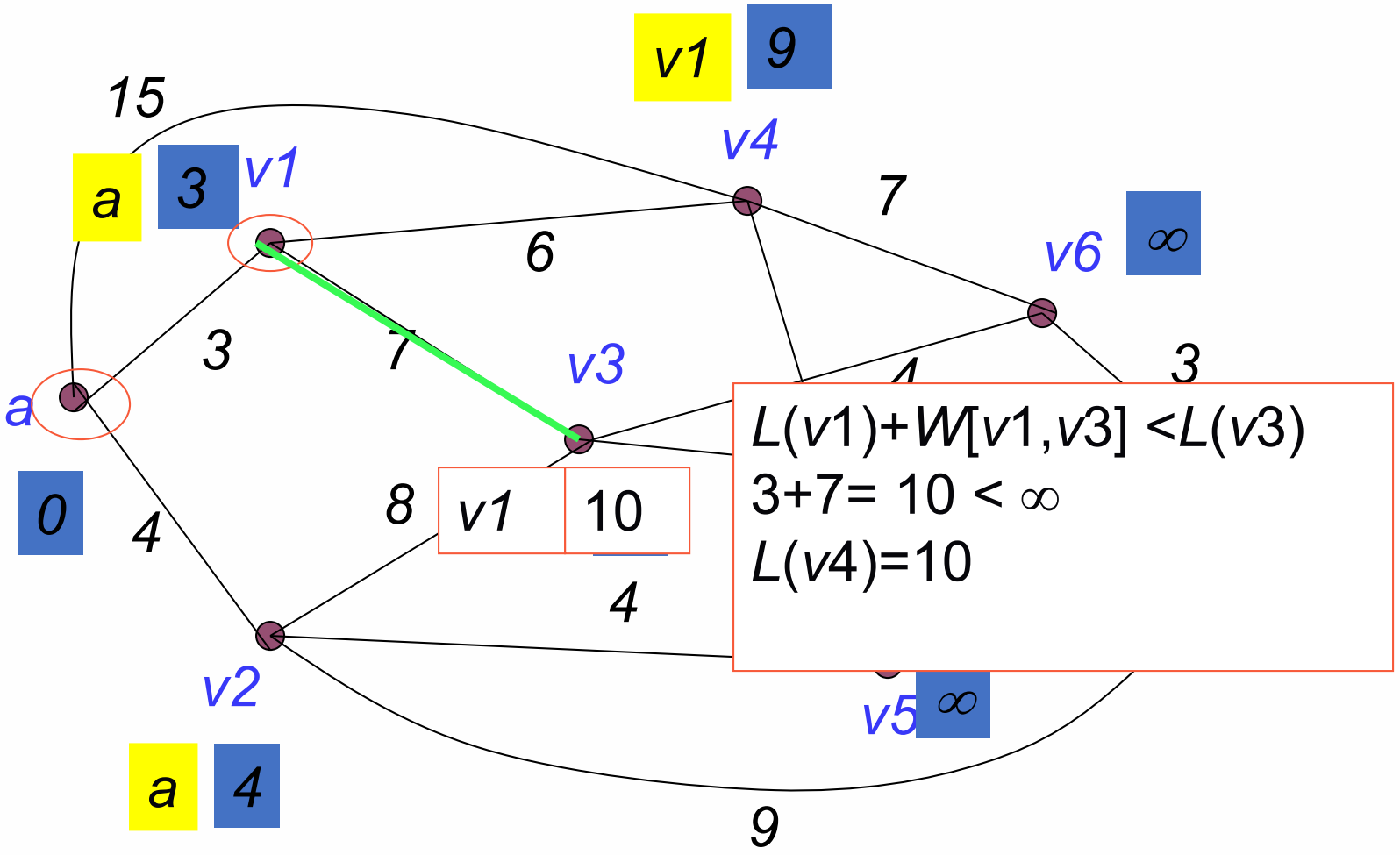
$S=\{a,v1\}$

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



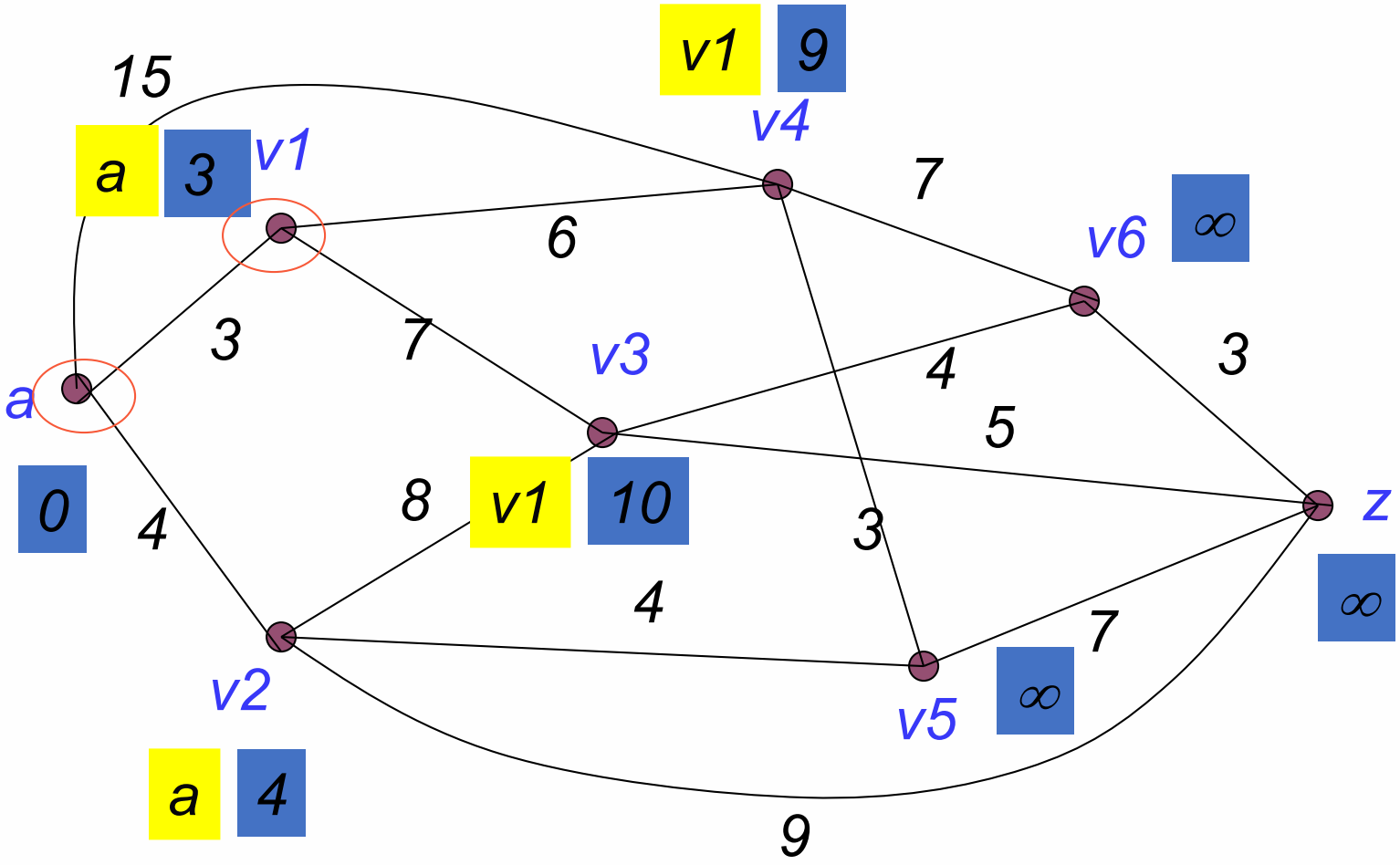
$S=\{a,v1\}$

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



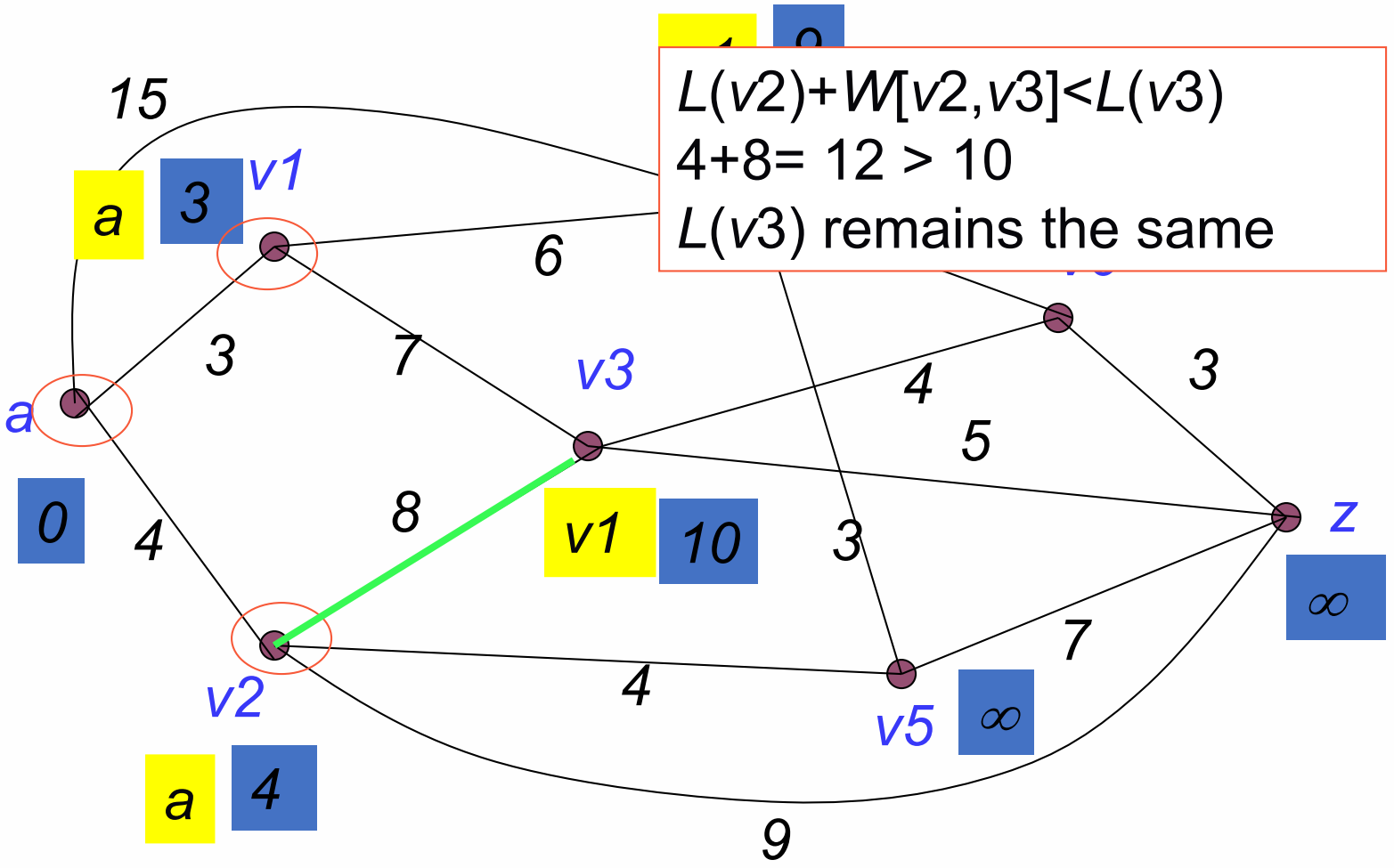
$S=\{a, v1\}$
$N=\{v2,v3,v4,v5,v6,z\}$

choose $v2$
because $L(v2)=4$ =
$\min\{L(u)|u\in N\}$



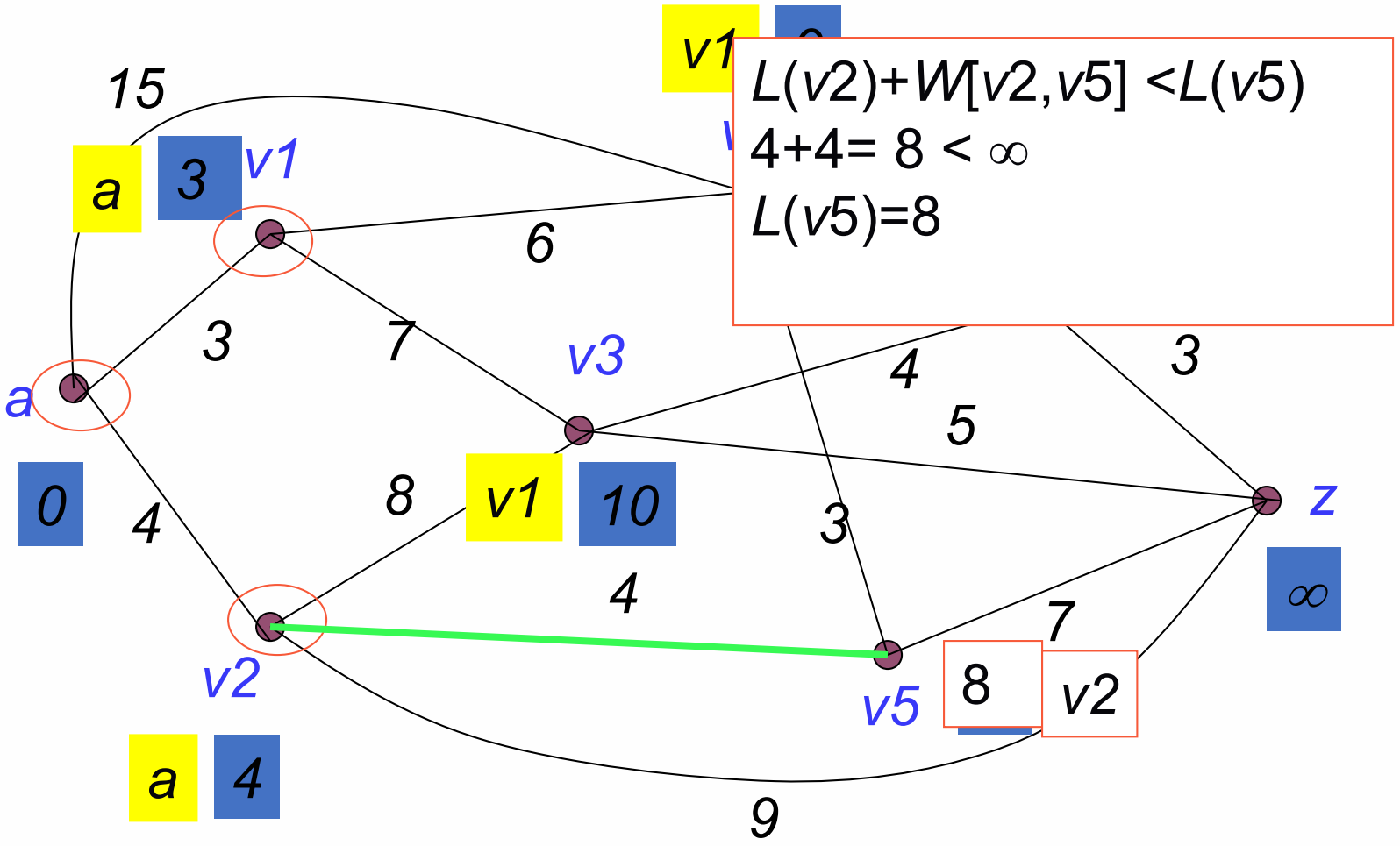
$S=\{a, v1, v2\}$

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

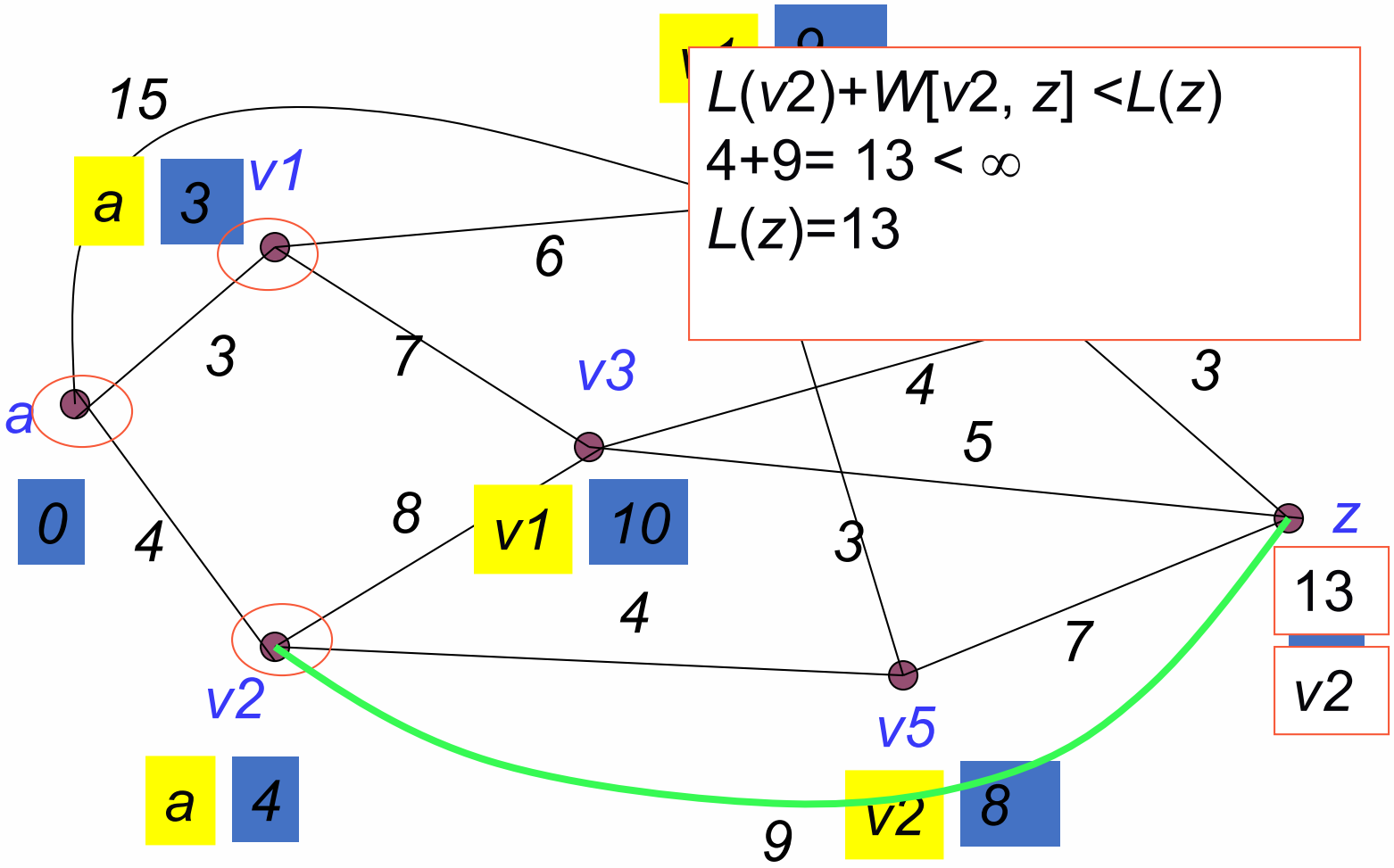


$S=\{a, v1, v2\}$

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



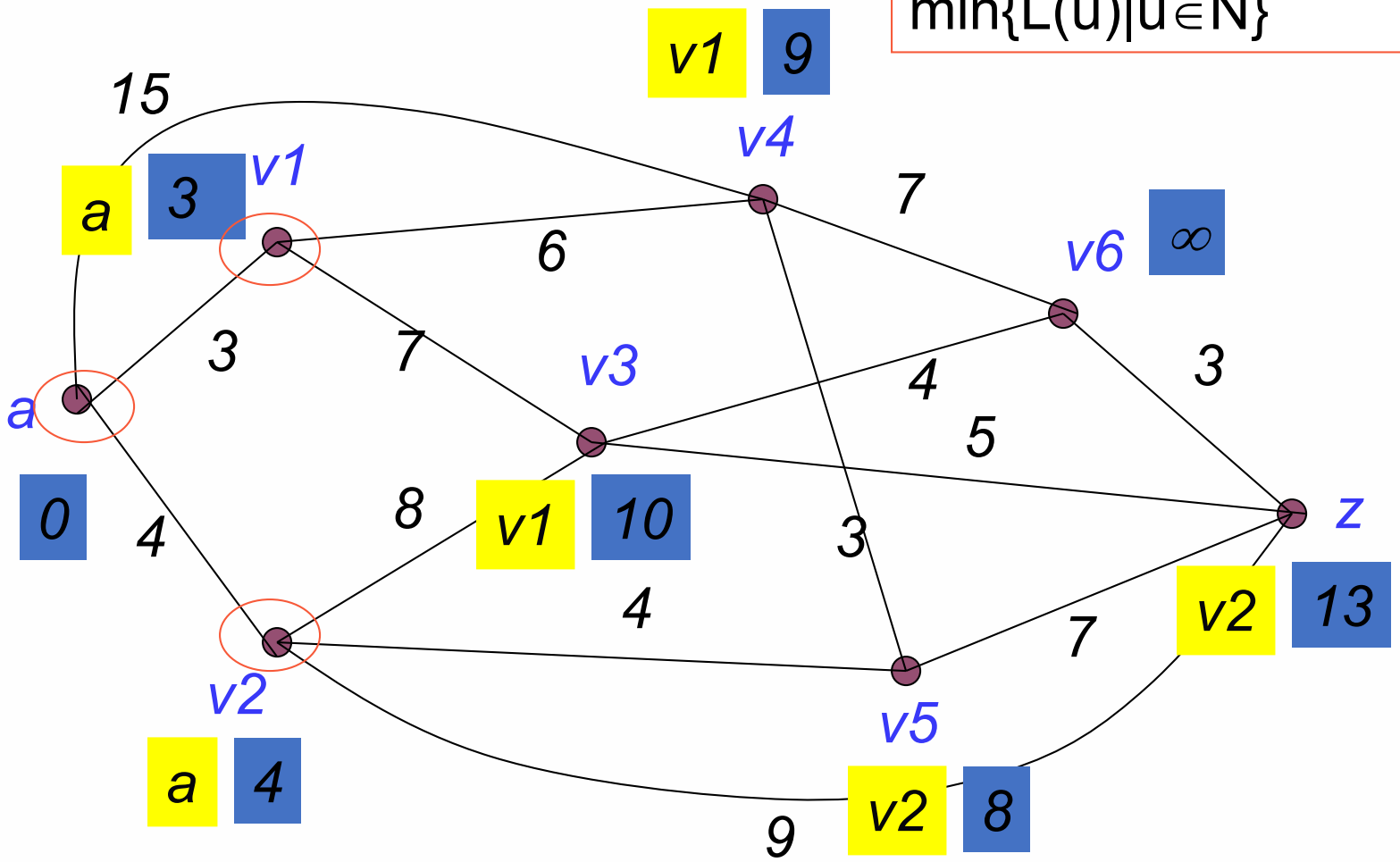
$S=\{a, v1, v2\}$
$N=\{v3,v4,v5,v6,z\}$



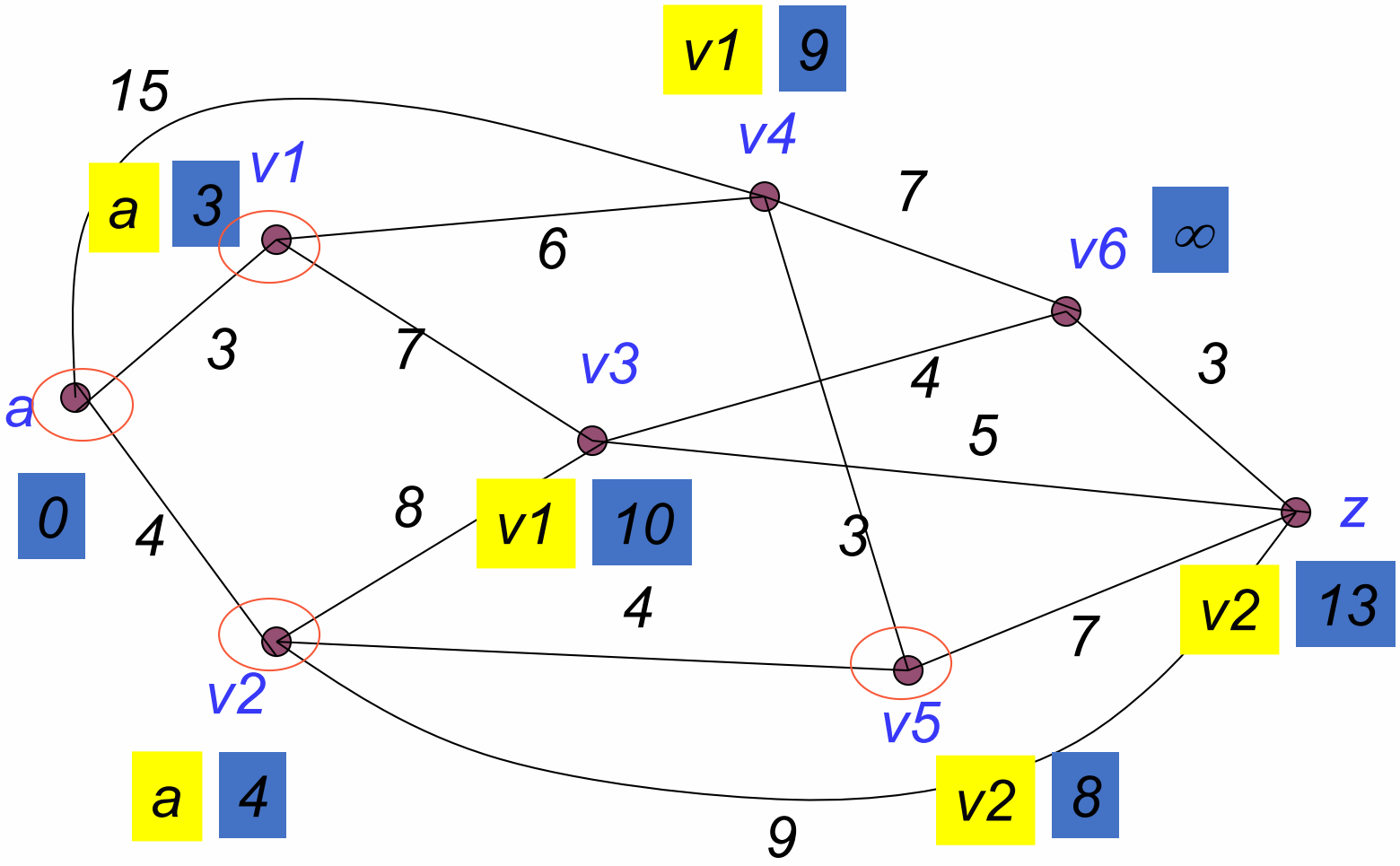
$L(v2)+W[v2, z] < L(z)$
$4+9= 13 < \infty$
$L(z)=13$

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

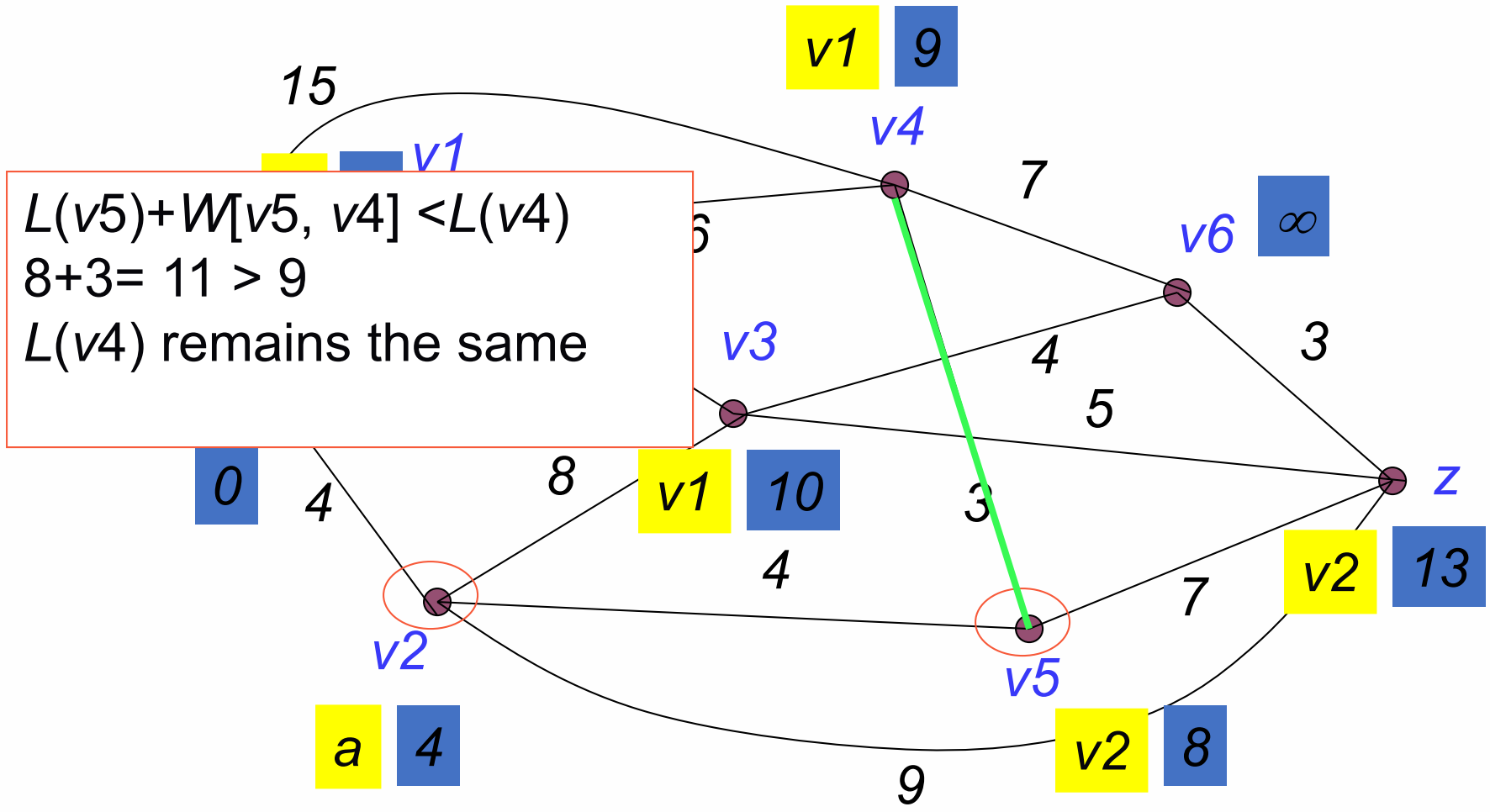
choose $v5$ because $L(v5)=8 = \min\{L(u)|u\in N\}$



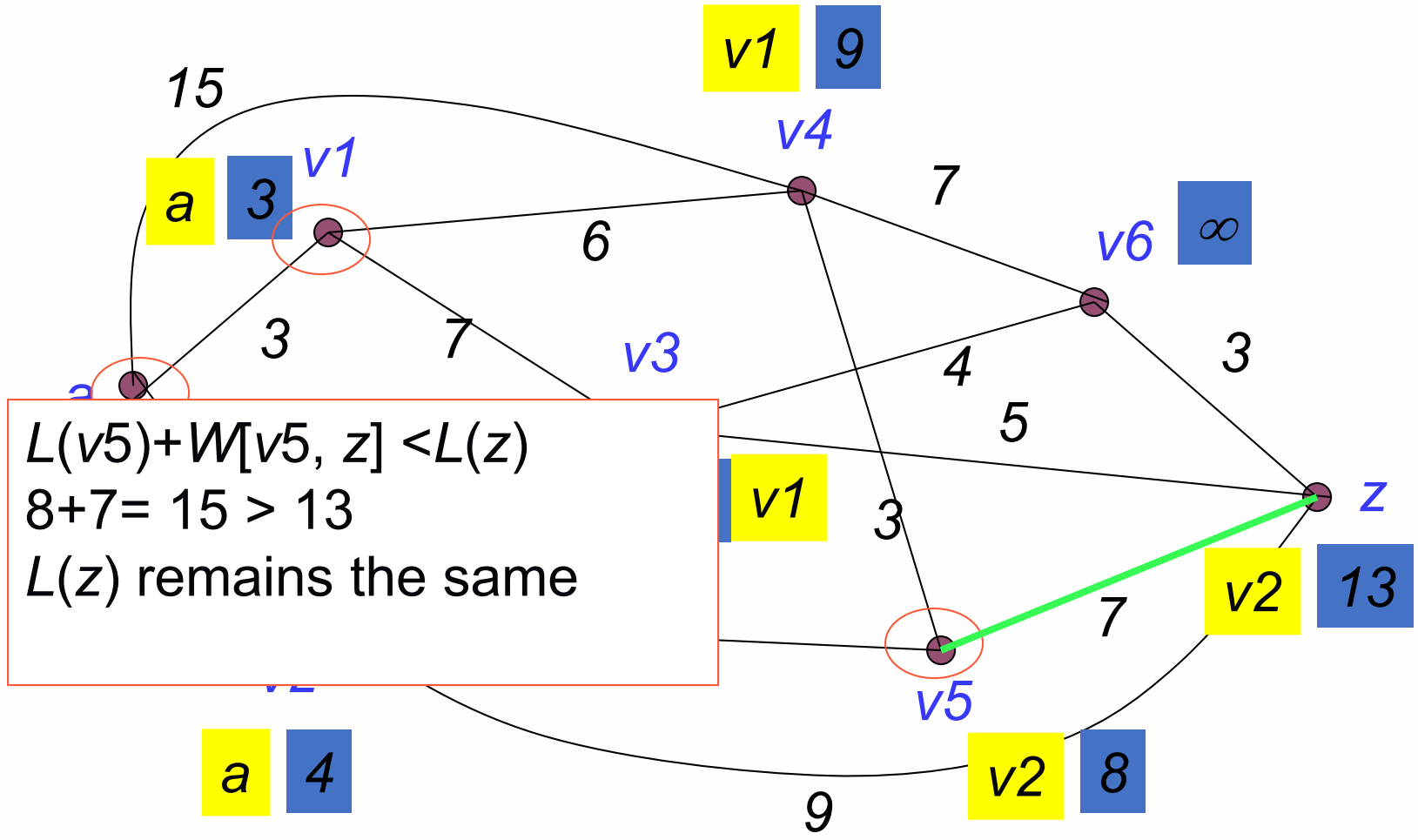
$S=\{a,v1,v2,v5\}$
$N=\{v3,v4,v6,z\}$



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

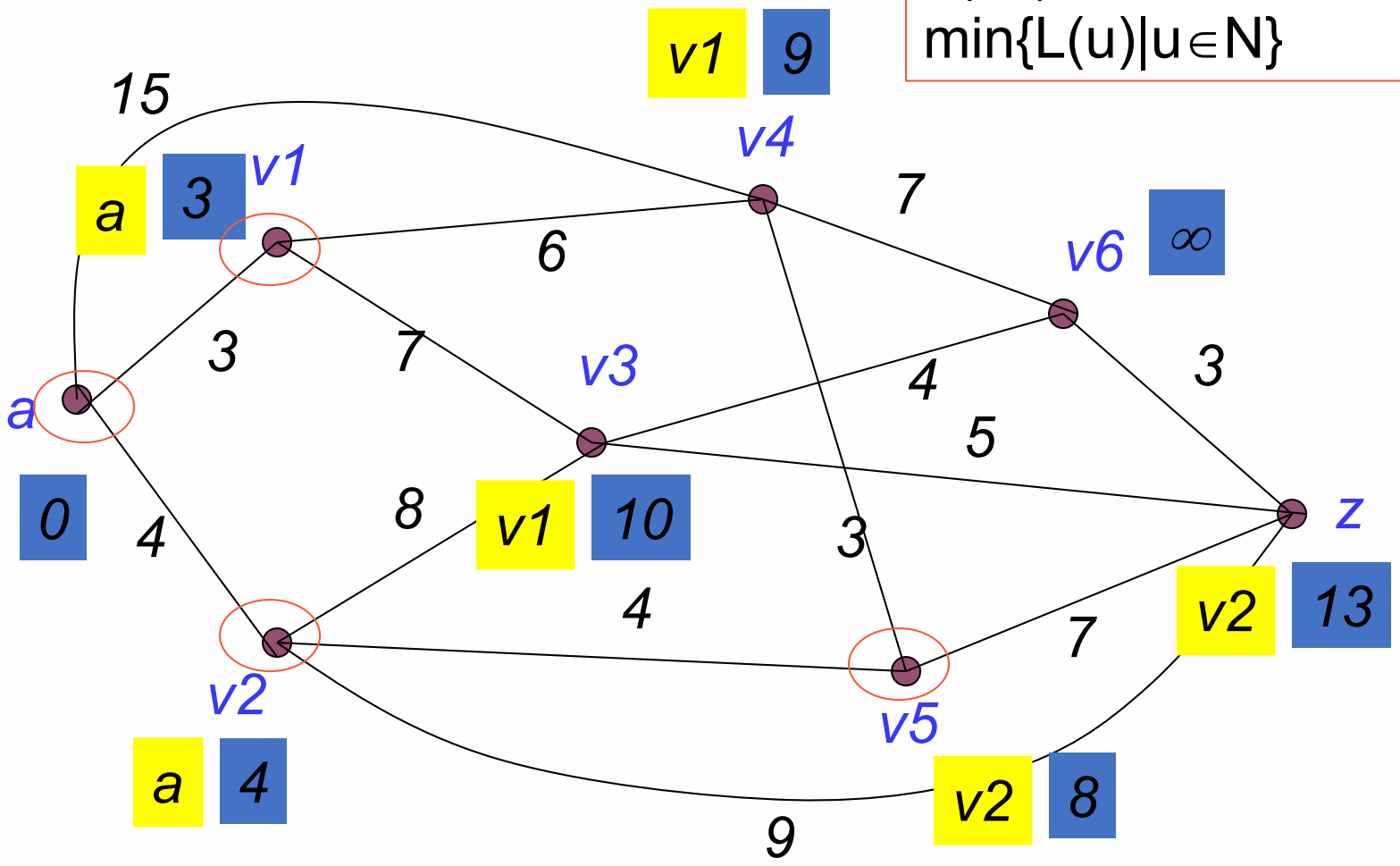


$S=\{a,v1,v2,v5\}$
$N=\{v3,v4,v6,z\}$

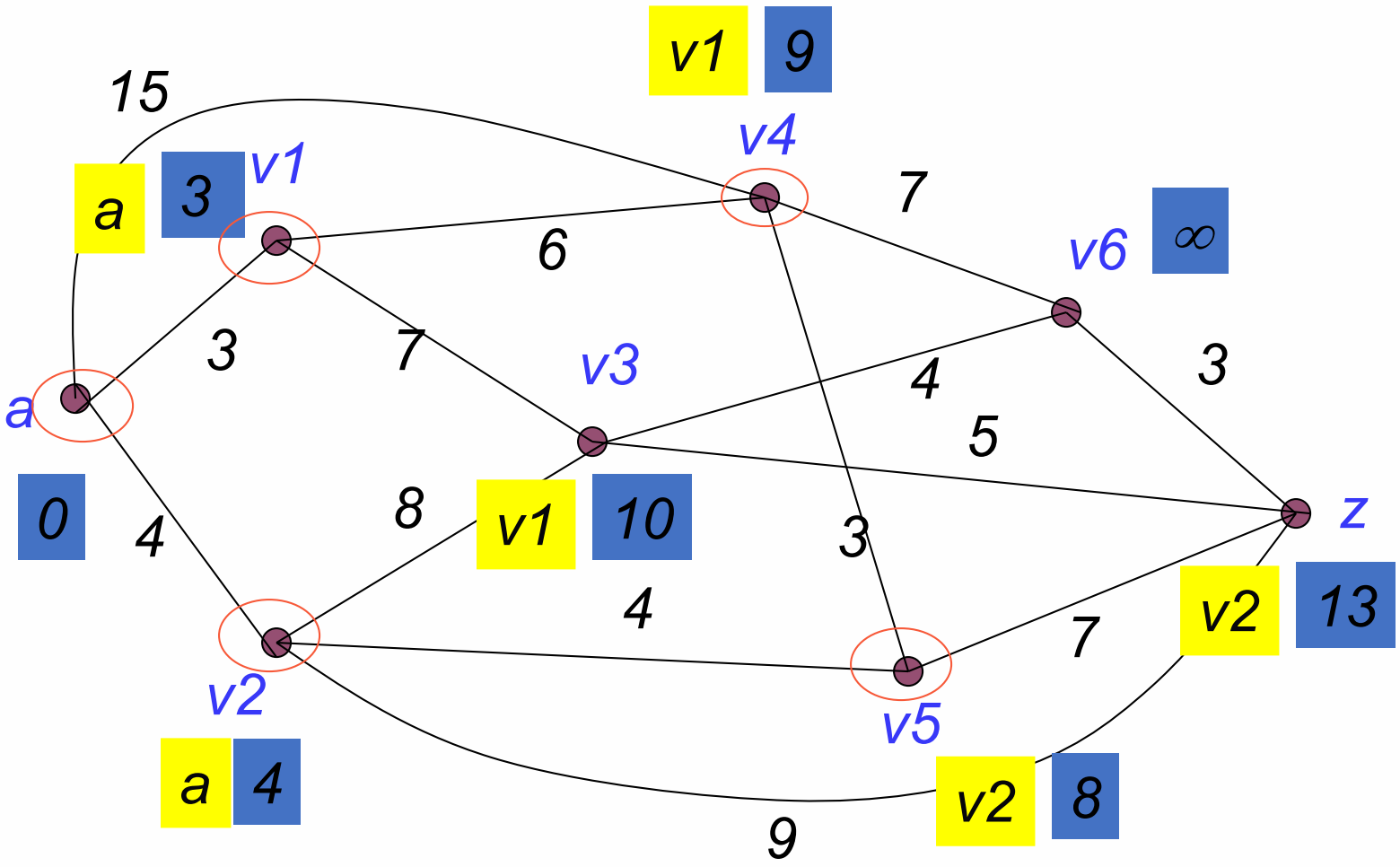


$S=\{a,v1,v2,v5\}$
$N=\{v3,v4,v6,z\}$

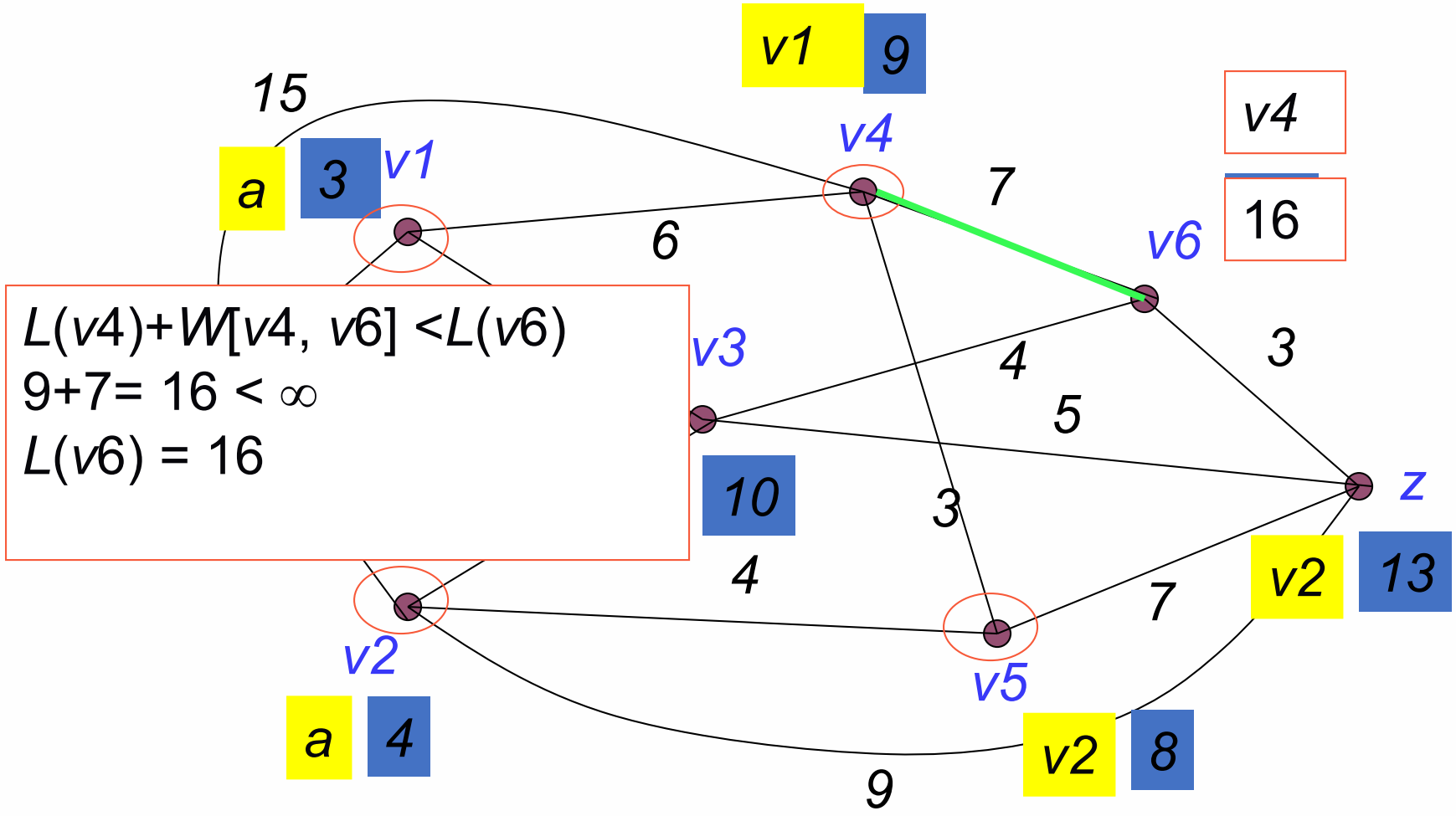
choose $v4$ because
$L(v4)=9 =$
$\min\{L(u)|u\in N\}$



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

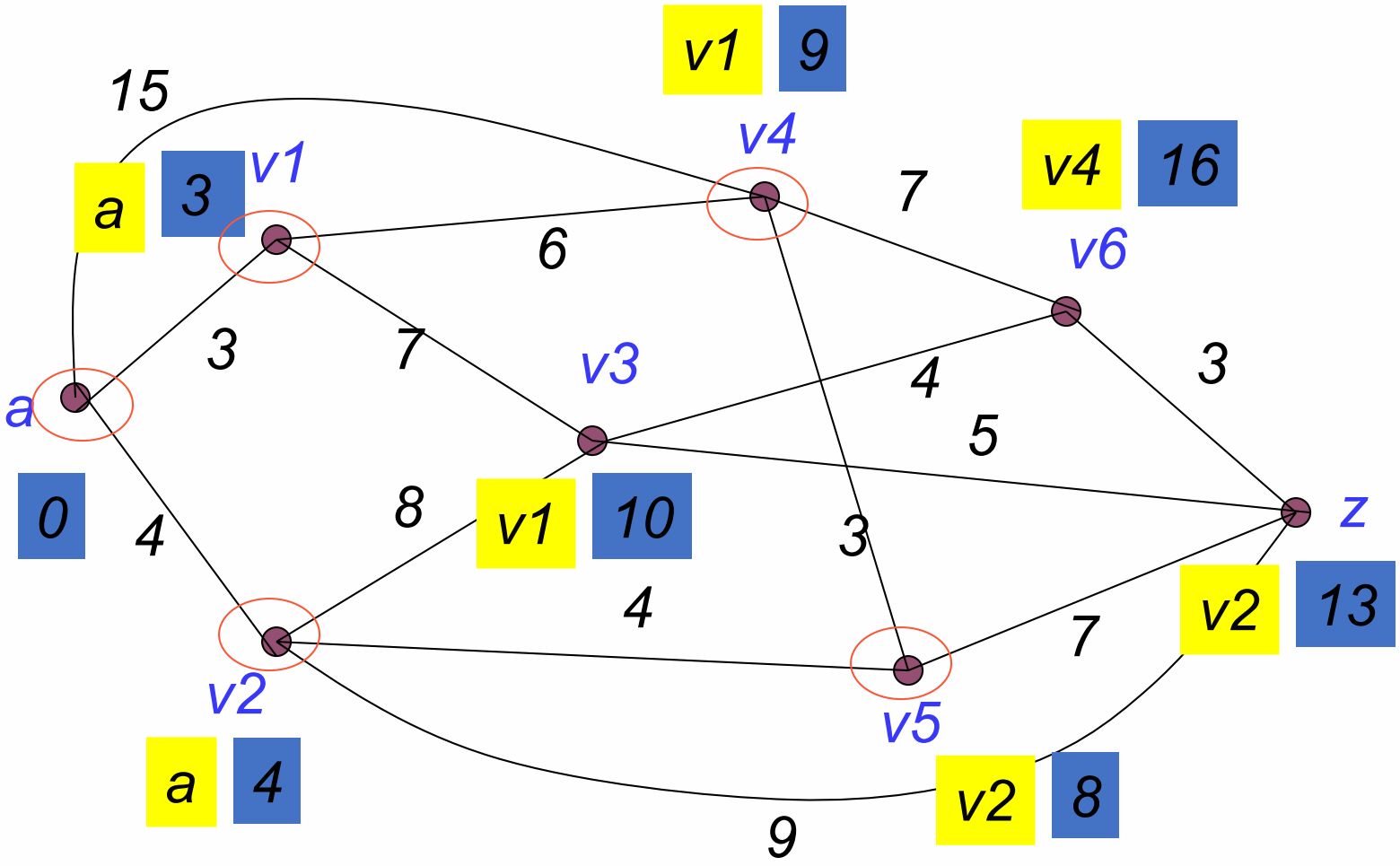


$S=\{a, v1, v2, v4, v5\}$
$N=\{v3, v6, z\}$

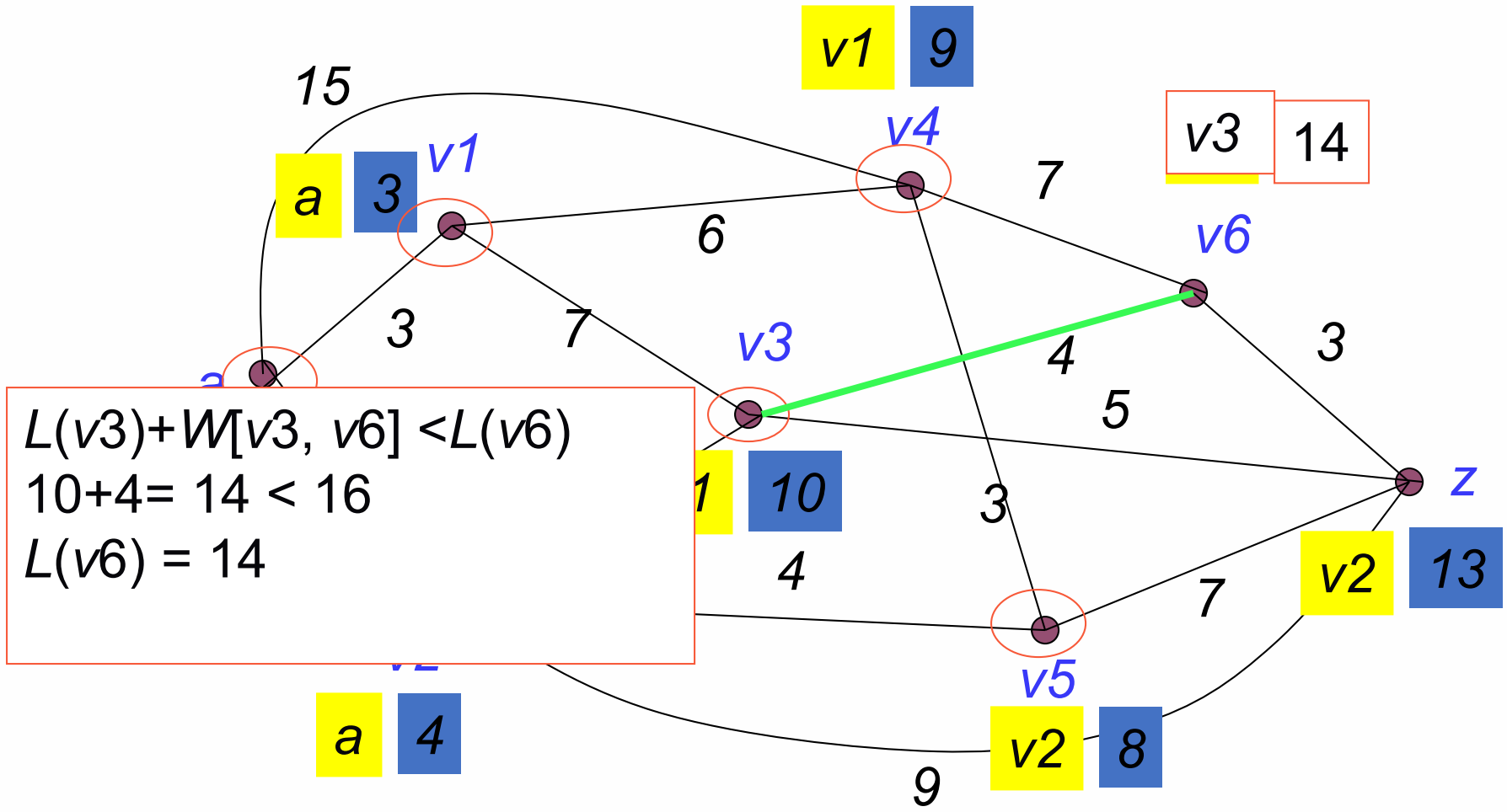


choose *v*3
because *L*(*v*3)=10
= min{L(u)|u∈N}

S={*a*,*v*1,*v*2, *v*4,*v*5}
N= {*v*3,*v*6,*z*}



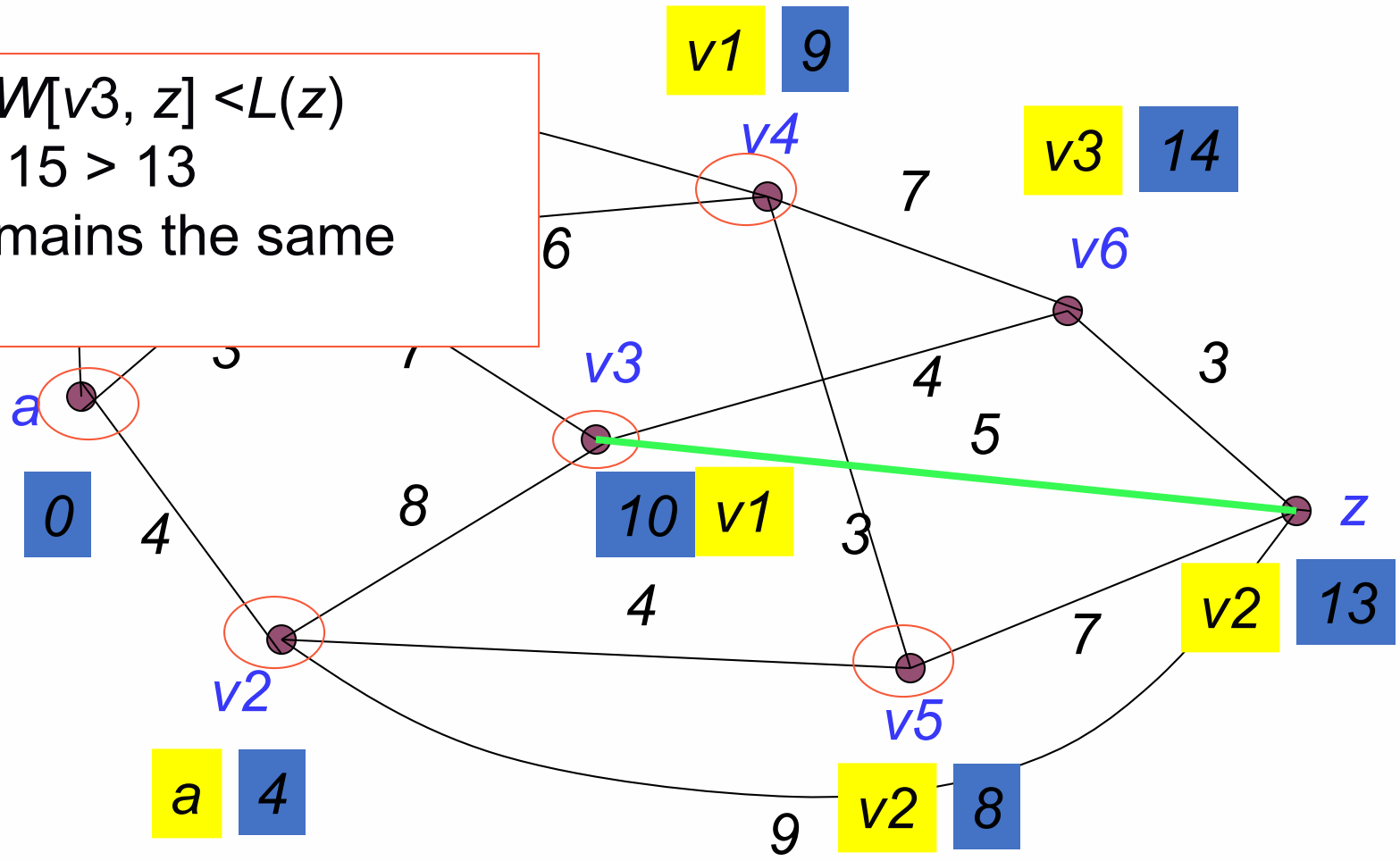
$S=\{a, v1, v2, v3, v4, v5\}$
$N=\{v6, z\}$



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

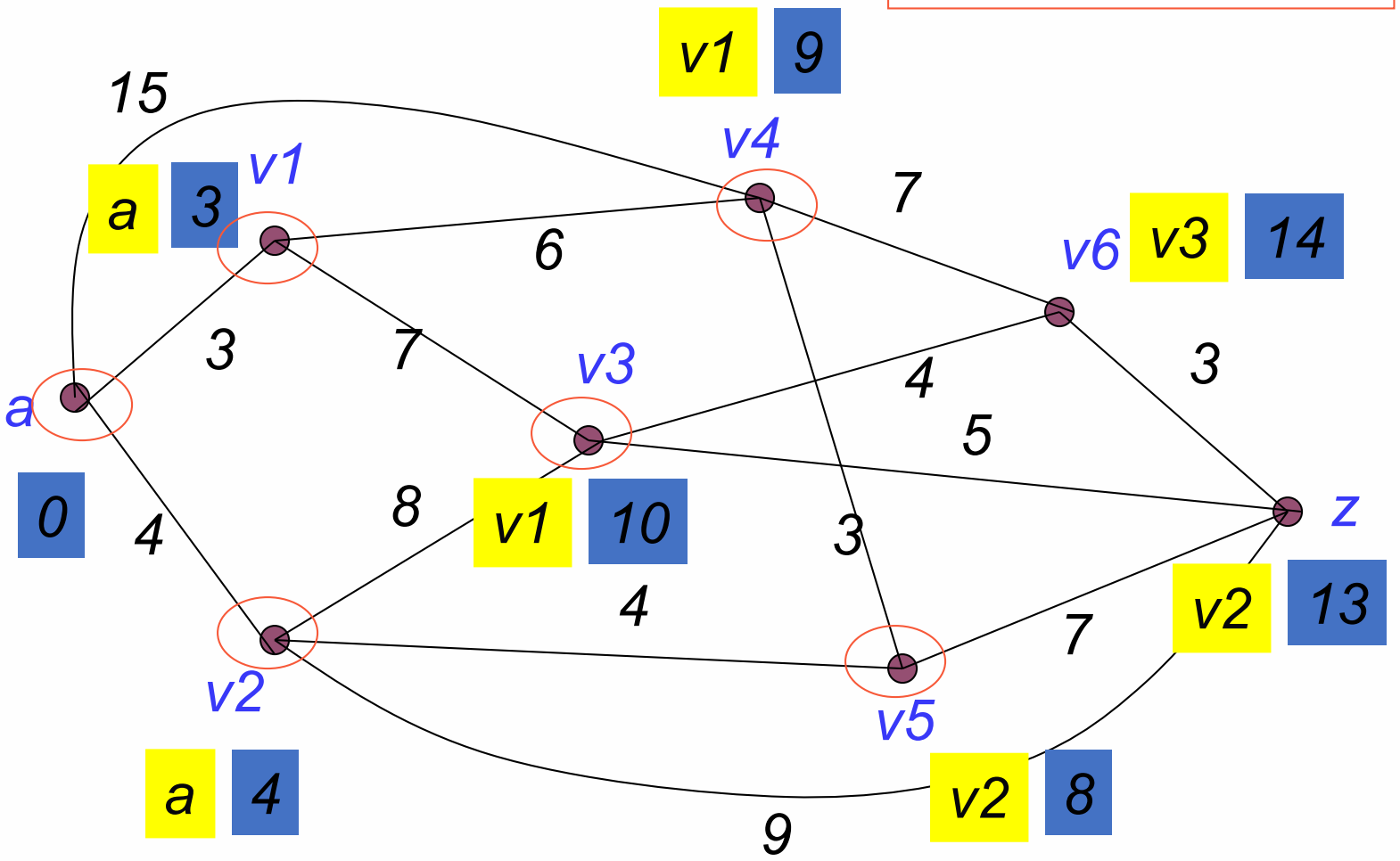
$S=\{a, v1, v2, v3, v4, v5\}$
$N=\{v6, z\}$

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



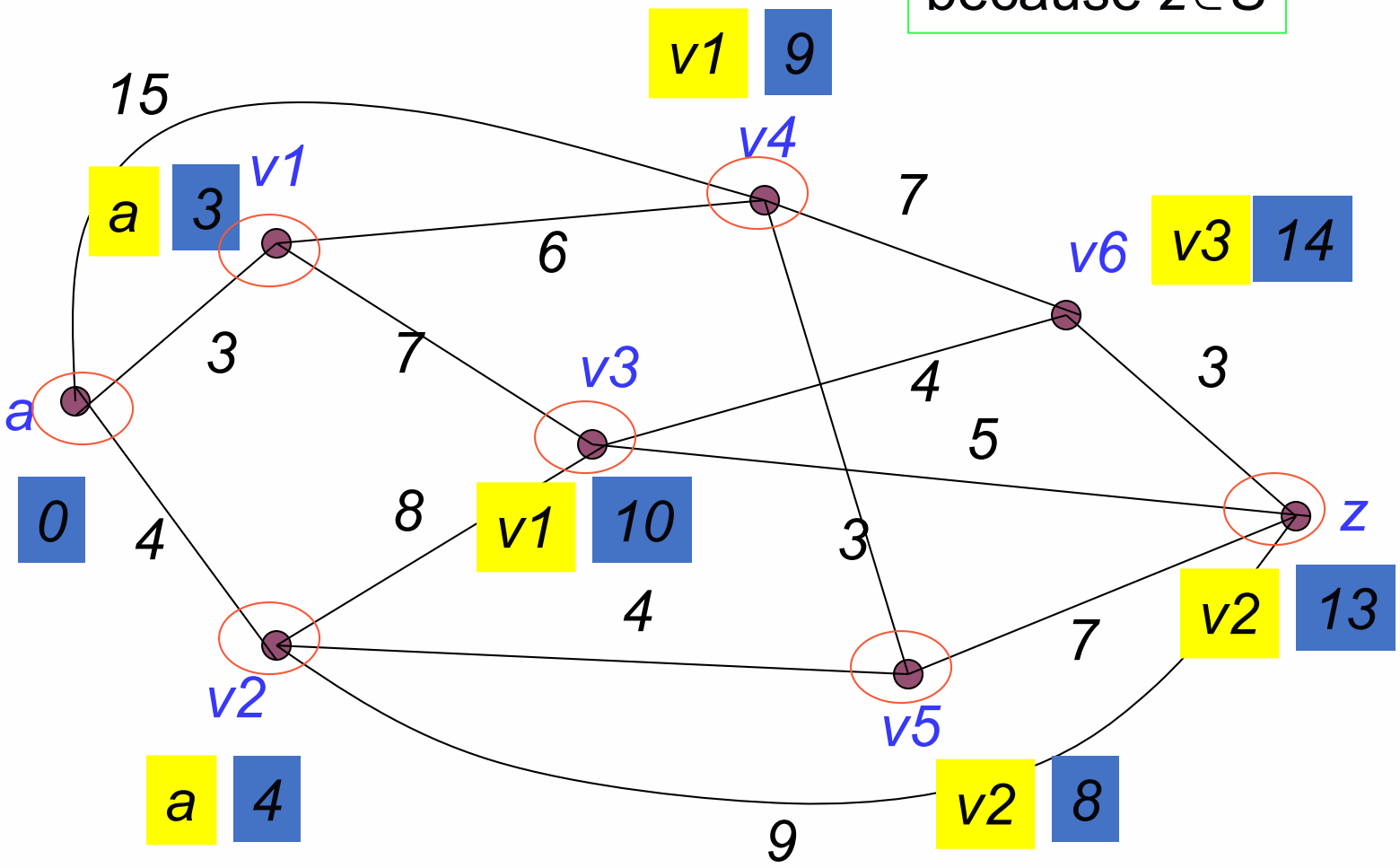
choose z because $L(z)=13 = \min\{L(u)|u \in N\}$

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



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

The loop terminates because $z \in S$



Shortest path from *a* to *z*

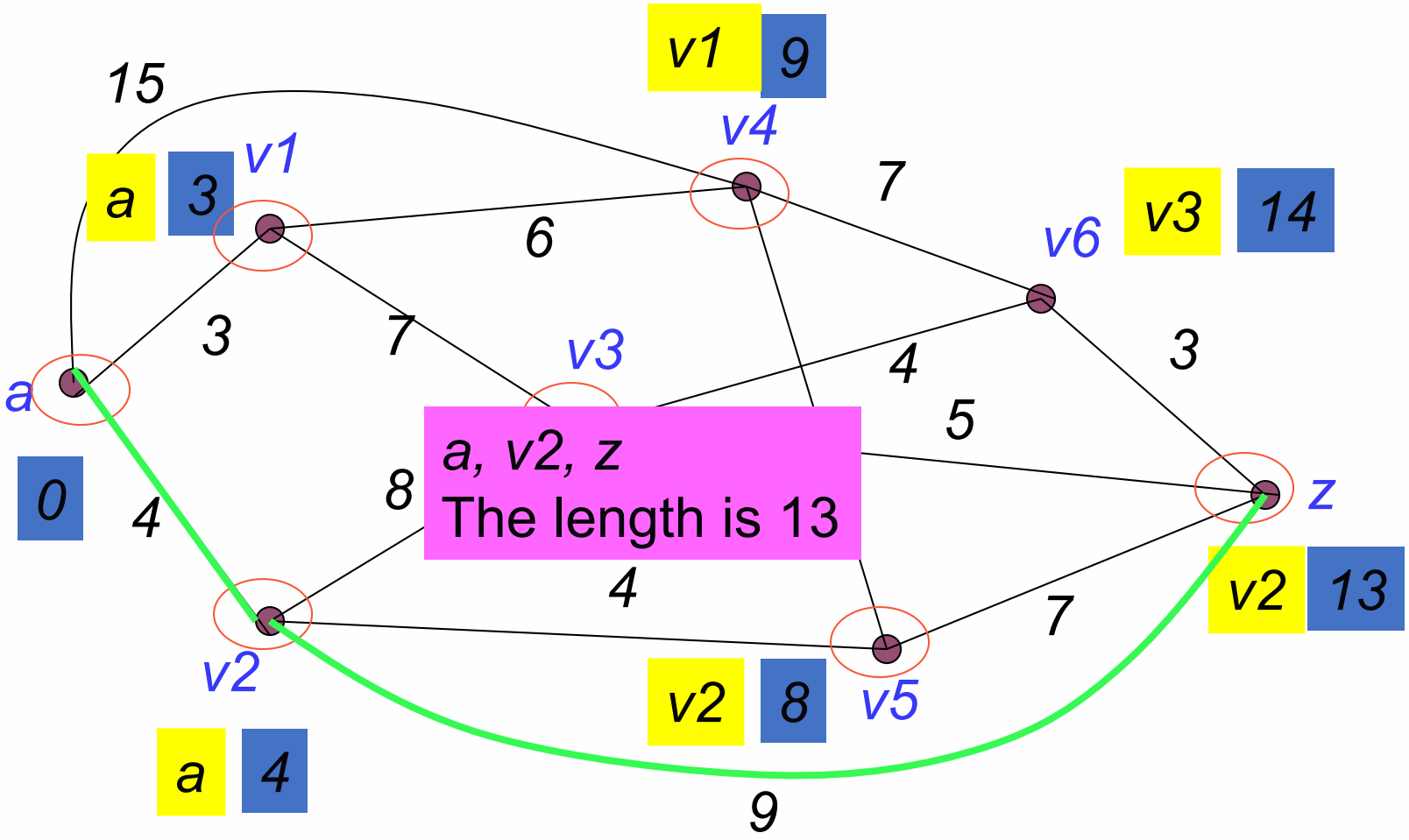


Table – Djikstra Algorithm

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**

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**

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**

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**

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**