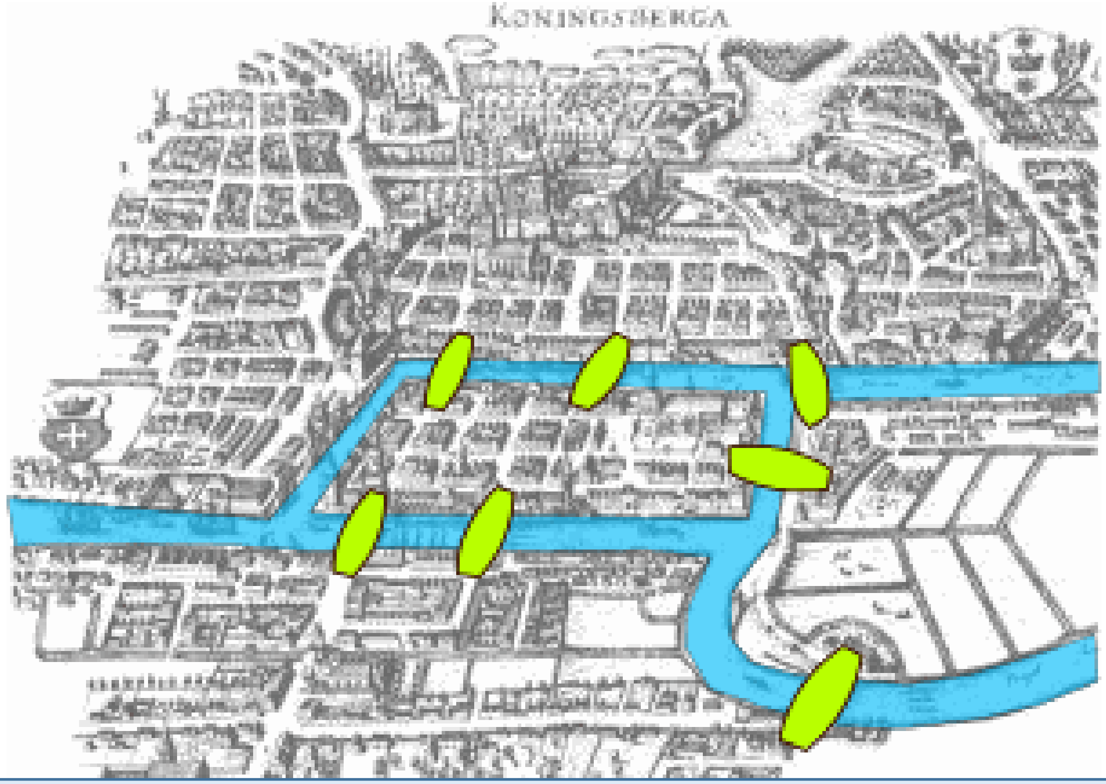
Chapter 8 Graph Theory 图论

Lu Han

hl@bupt.edu.cn

8.1 Introduction 简介

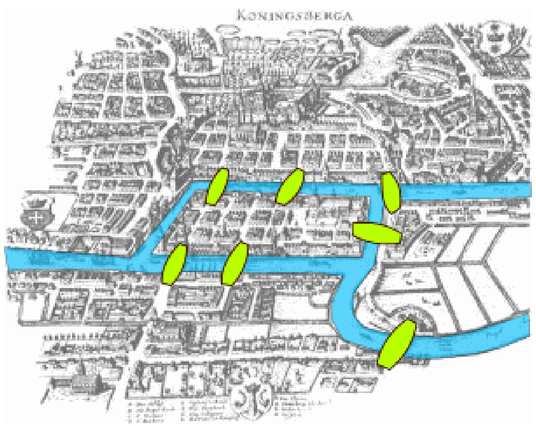
Seven Bridges of Königsberg 柯尼斯堡七桥问题



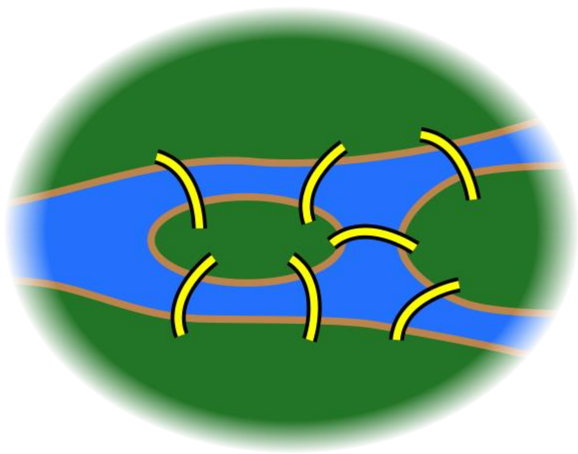
Is it possible to walk with a route that crosses each bridge exactly once?

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题



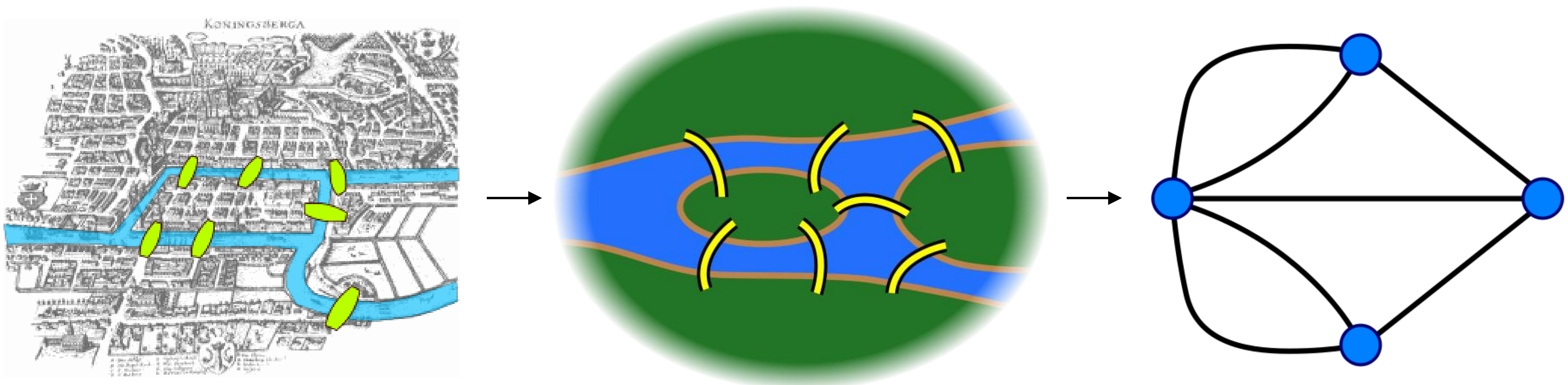
→



Forget unimportant details.

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题

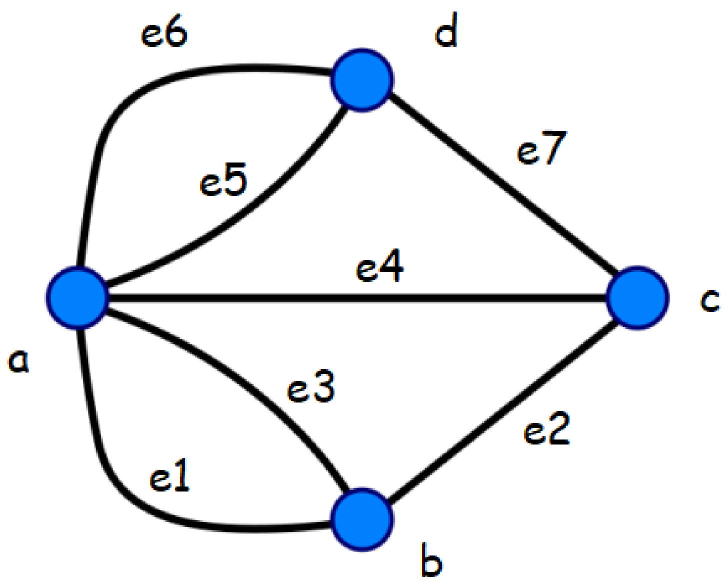


Forget unimportant details.

Forget even more.

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题

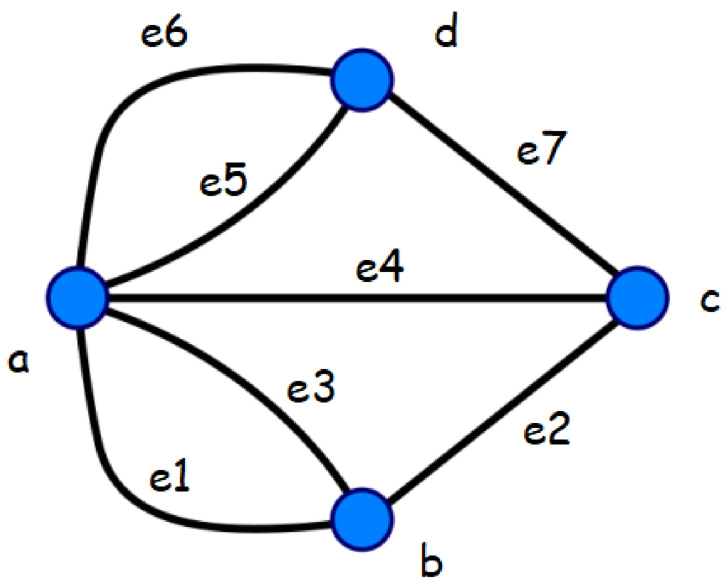


So, what is the **"Seven Bridges of Königsberg"** problem now?

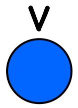
Is it possible to find a walk that visits each edge exactly once?

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题



What is the property of every "intermediate" point v?

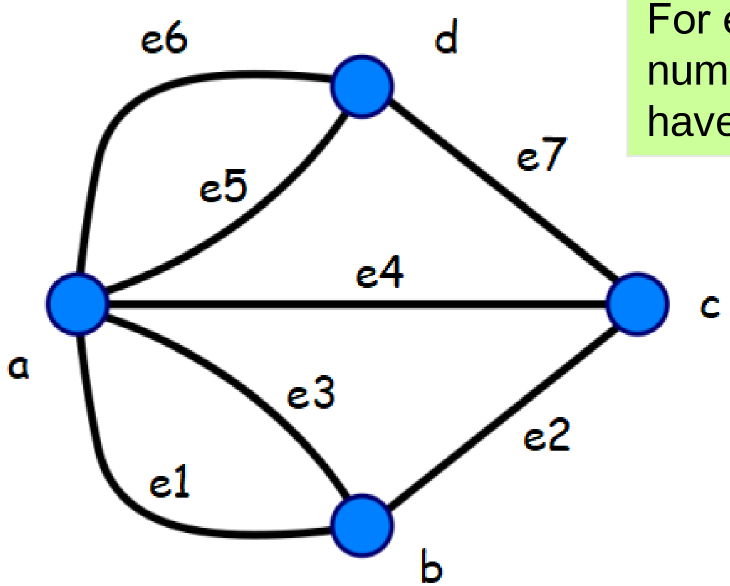


Suppose there is such a walk, there is a starting point and an endpoint point.

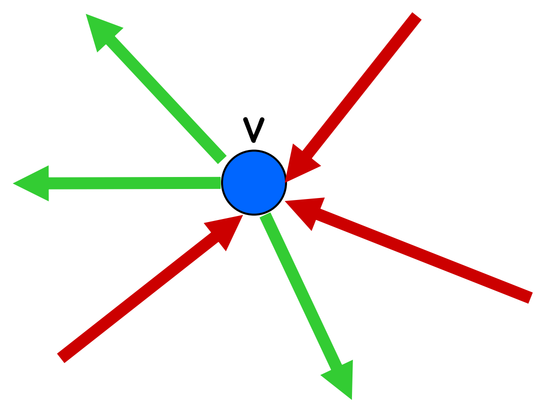
Is it possible to find a walk that visits each edge exactly once?

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题



For every “intermediate” point v, there must be the same number of incoming and outgoing edges, and so v must have an **even number of edges**.

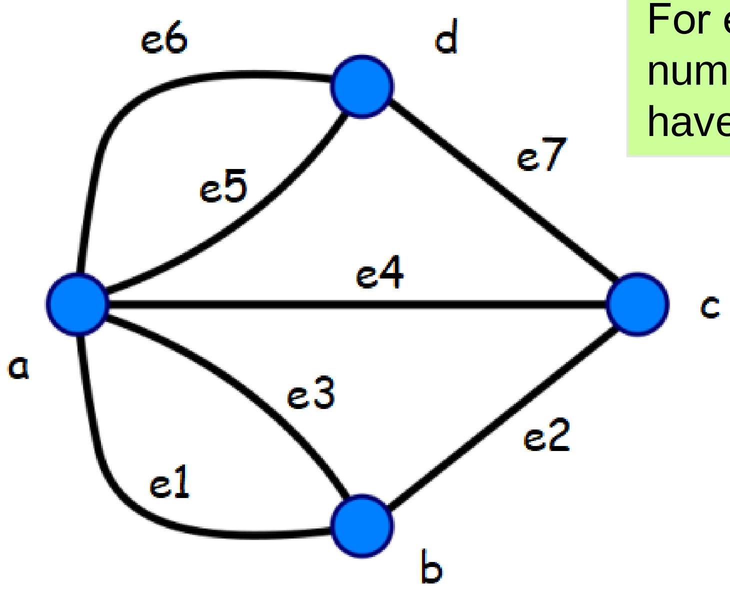


Suppose there is such a walk, there is a starting point and an endpoint point.

Is it possible to find a walk that visits each edge exactly once?

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题



For every “intermediate” point v, there must be the same number of incoming and outgoing edges, and so v must have an **even number of edges**.

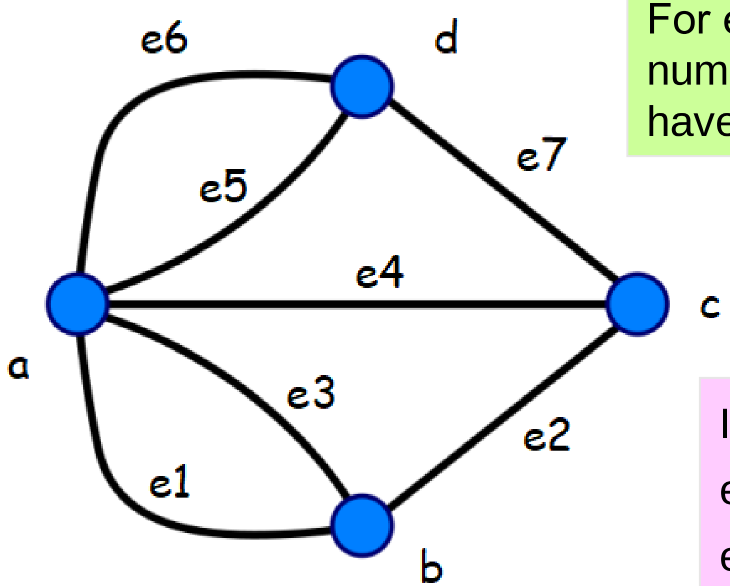
So, at most **two** vertices can have odd number of edges.

Suppose there is such a walk, there is a starting point and an endpoint point.

Is it possible to find a walk that visits each edge exactly once?

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题



For every “intermediate” point v, there must be the same number of incoming and outgoing edges, and so v must have an **even number of edges**.

So, at most **two** vertices can have odd number of edges.

In this graph, every vertex has only an odd number of edges, and so there is no walk which visits each edge exactly one.

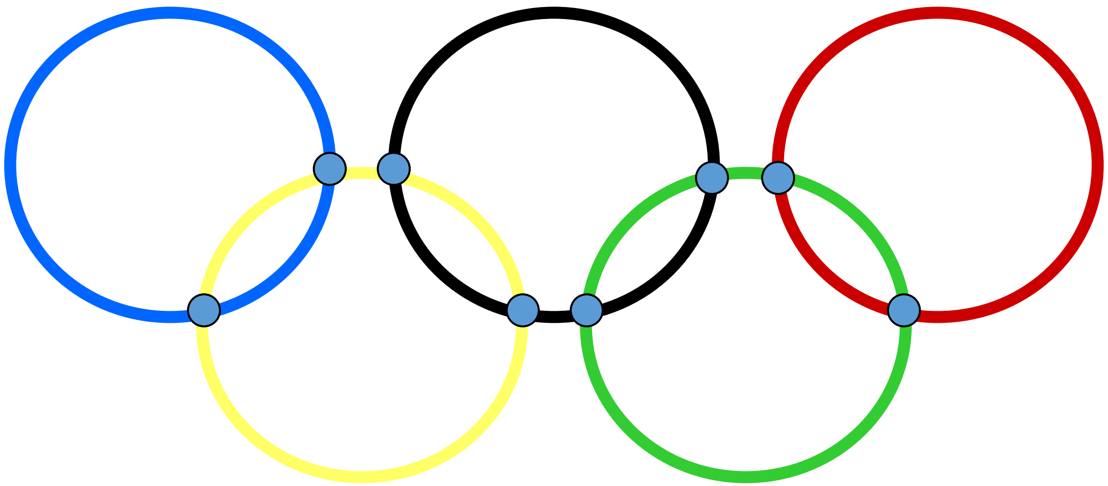
Suppose there is such a walk, there is a starting point and an endpoint point.

Is it possible to find a walk that visits each edge exactly once?

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题

So Euler showed that the **"Seven Bridges of Königsberg"** is unsolvable.

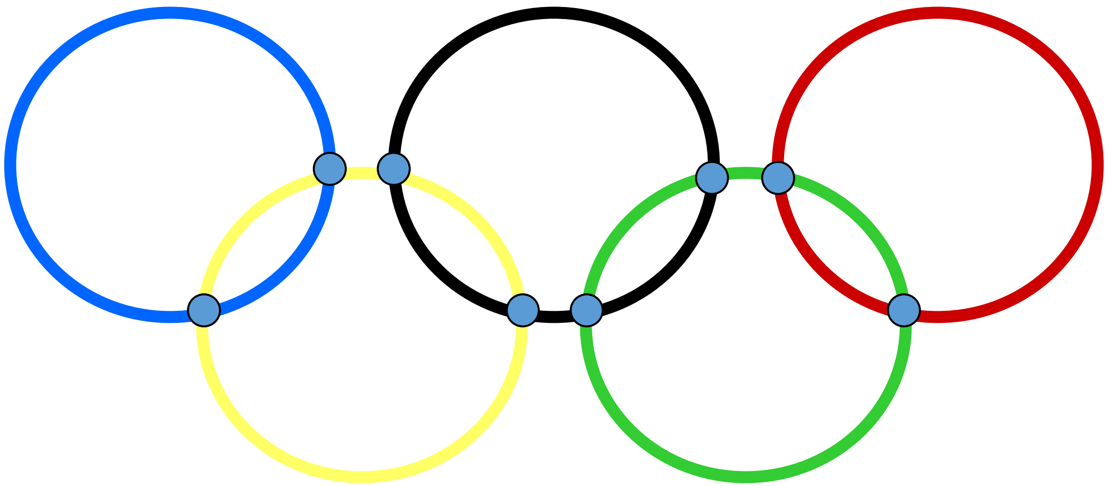


8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题

So Euler showed that the **“Seven Bridges of Königsberg”** is unsolvable.

When is it possible to have a walk that visits every edge exactly once?



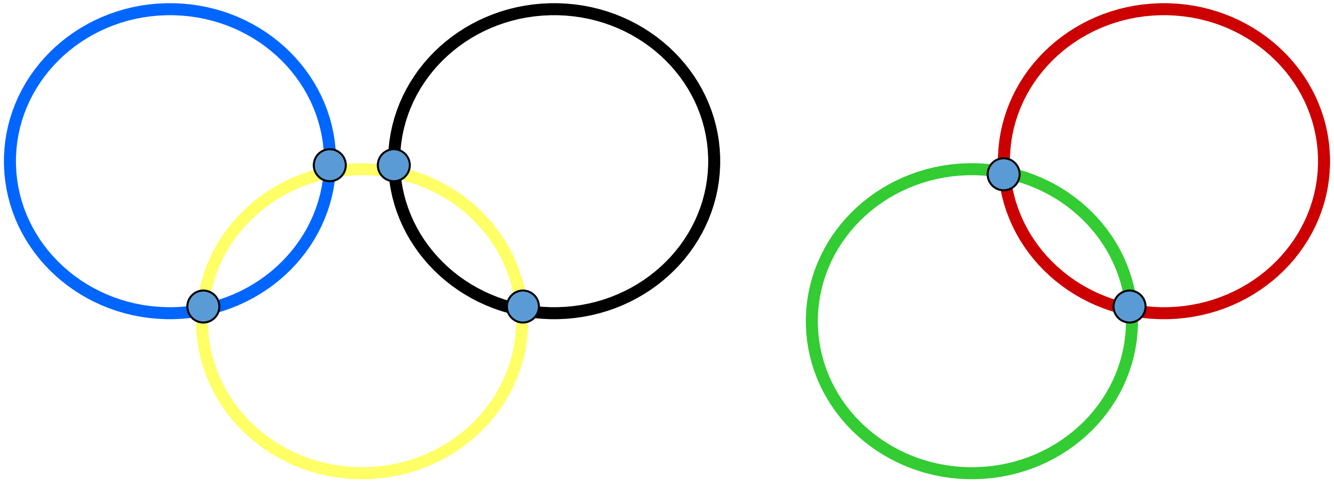
Is it always possible to find such a walk if there is **at most two** vertices with odd number of edges?

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题

So Euler showed that the **"Seven Bridges of Königsberg"** is unsolvable.

When is it possible to have a walk that visits every edge exactly once?



Is it always possible to find such a walk if there is **at most two** vertices with odd number of edges?

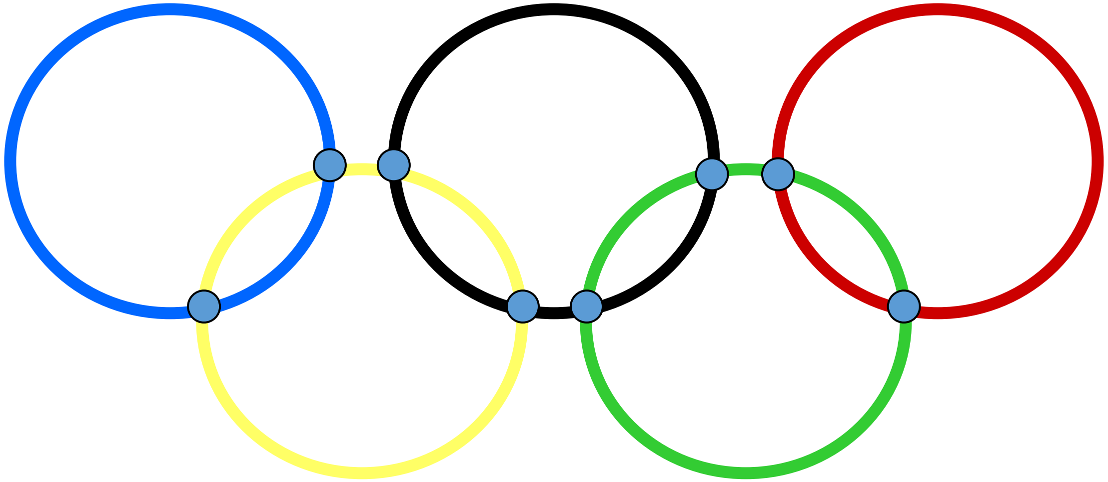
NO!

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题

So Euler showed that the **"Seven Bridges of Königsberg"** is unsolvable.

When is it possible to have a walk that visits every edge exactly once?



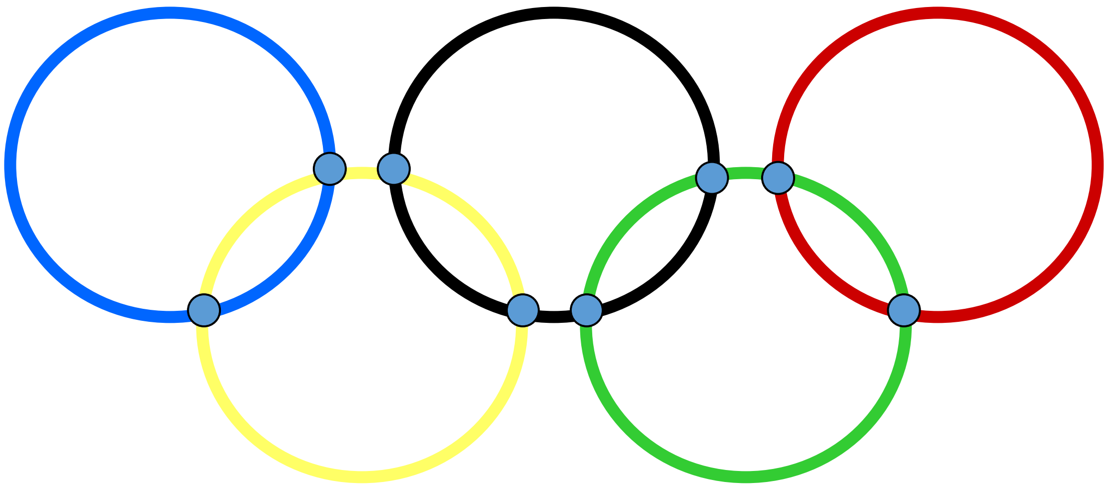
Is it always possible to find such a walk if the graph is **"connected"** and there are **at most two** vertices with odd number of edges?

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题

So Euler showed that the **"Seven Bridges of Königsberg"** is unsolvable.

When is it possible to have a walk that visits every edge exactly once?



Is it always possible to find such a walk if the graph is **"connected"** and there are **at most two** vertices with odd number of edges?

YES!

8.1 Introduction 简介

Seven Bridges of Königsberg 柯尼斯堡七桥问题

So Euler showed that the **"Seven Bridges of Königsberg"** is unsolvable.

When is it possible to have a walk that visits every edge exactly once?

Eulerian Path

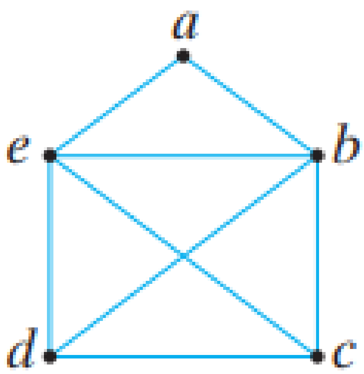
Euler's theorem: A graph has an Eulerian path if and only if it is "connected" and has at most two vertices with an odd number of edges.

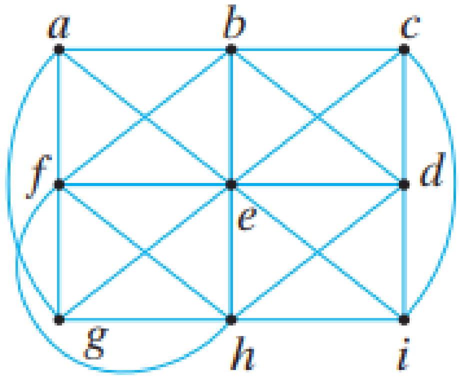
This theorem was proved in 1736,

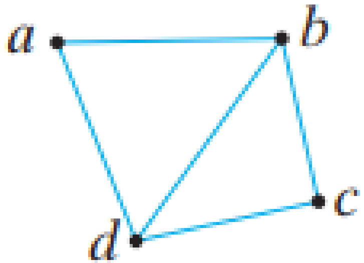
and was regarded as the starting point of graph theory.

8.1 Introduction 简介

Exercise Does there exists a path that visits each edge exactly once?

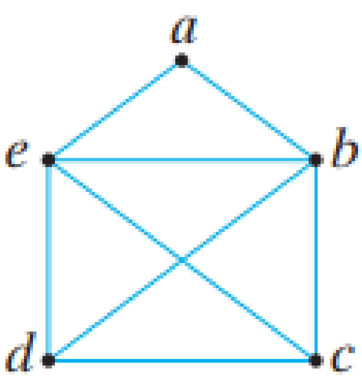


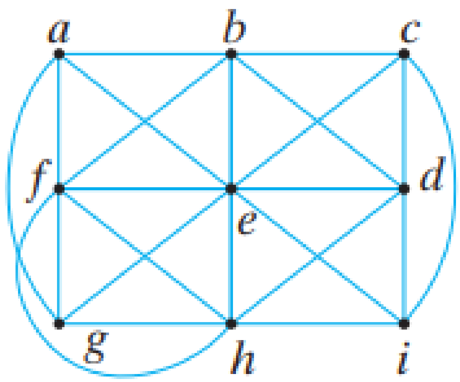


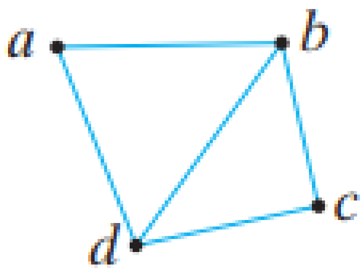


8.1 Introduction 简介

Exercise Does there exists a path from a to a that visits each edge exactly once?

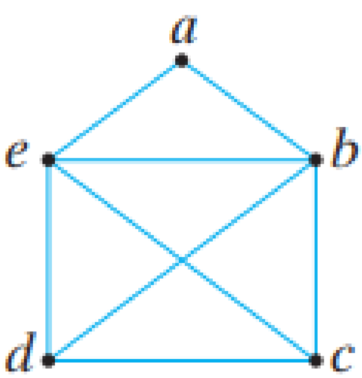


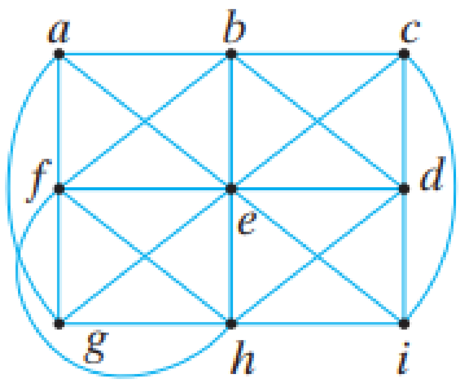


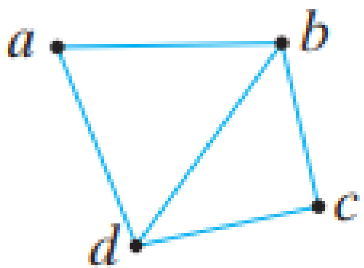


8.1 Introduction 简介

Exercise Does there exists a path from a to a that visits each edge exactly once?



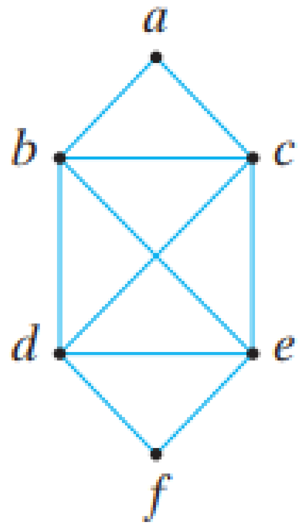


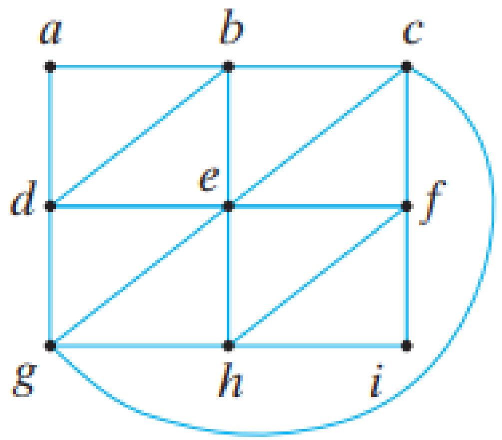


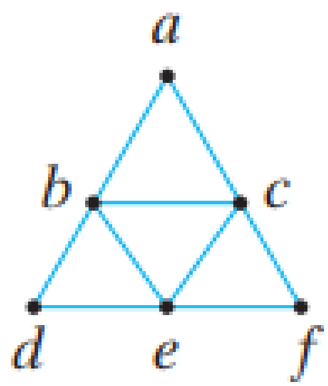
No!

8.1 Introduction 简介

Exercise Does there exists a path from a to a that visits each edge exactly once?







8.1 Introduction 简介

Definition 8.1.1 A **graph (or undirected graph) (无向图)** G consists of a set V of **vertices (or nodes) (顶点)** and a set E of edges (or arcs) **(边)** such that each edge $e \in E$ is associated with an unordered pair of vertices.

If there is a unique edge e associated with the vertices v and w, we write $e = (v,w)$ or $e = (w, v)$.

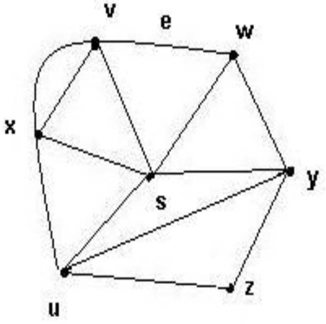
In this context, (v,w) denotes an edge between v and w in an undirected graph and not an ordered pair.

8.1 Introduction 简介

Definition 8.1.1 A **graph (or undirected graph) (无向图)** G consists of a set V of **vertices (or nodes) (顶点)** and a set E of edges (or arcs) **(边)** such that each edge $e \in E$ is associated with an unordered pair of vertices.

If there is a unique edge e associated with the vertices v and w, we write $e = (v,w)$ or $e = (w, v)$.

In this context, (v,w) denotes an edge between v and w in an undirected graph and not an ordered pair.



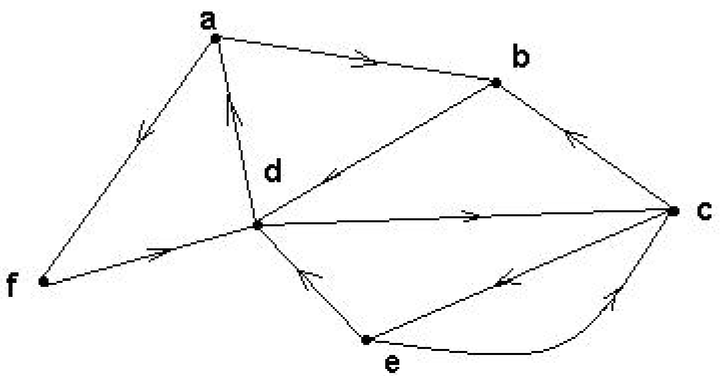
V = {s, u, v, w, x, y, z}

E = {(x,s), $(x,v)_1$, $(x,v)_2$, (x,u), (v,w), (s,v), (s,u), (s,w), (s,y), (w,y), (u,y), (u,z),(y,z)}

8.1 Introduction 简介

A **directed graph (or digraph) (有向图)** G consists of a set V of **vertices (or nodes) (顶点)** and a set E of edges (or arcs) **(边)** such that each edge $e \in E$ is associated with an unordered pair of vertices.

If there is a unique edge e associated with the ordered pair (v,w), we write $e = (v, w)$, which denotes an edge from v to w.



8.1 Introduction 简介

An edge e associated with the pair of vertices v and w is said to be

- **incident on (相关联的)** v and w.

The vertices v and w are said to be

- **incident on (相关联的)** e and to be **adjacent vertices (相邻顶点)**.

If G is a graph with vertices V and edges E, we write $G=(V, E)$.

Unless specified otherwise, the set E and V are assumed to be finite and V is assumed to be nonempty.

8.1 Introduction 简介

- **parallel edges (平行边/并行边)**

Two or more edges associated with a same pair of vertices.

- **loop (圈)**

An edge incident on a single vertex,

or an edge that starts and ends at the same vertex.

- **isolated vertex (孤立顶点)**

A vertex that is not incident on any edge.

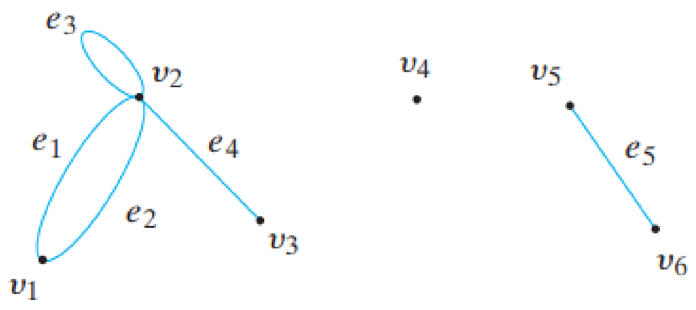
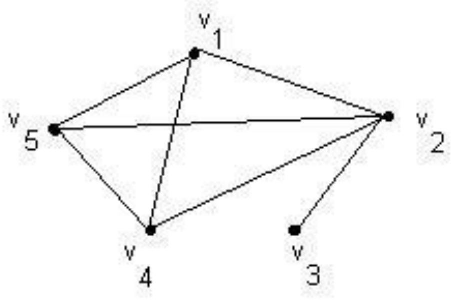


Figure 8.1.5 A graph with parallel edges and loops.

8.1 Introduction 简介

- **simple graph (简单图)**

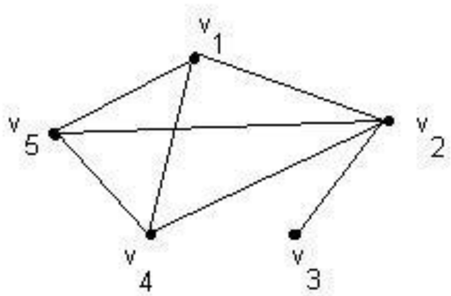
A graph with neither loops nor parallel edges.



8.1 Introduction 简介

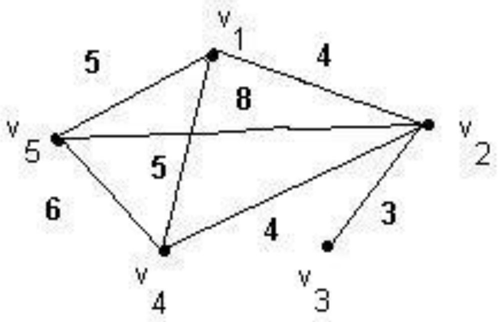
- **simple graph (简单图)**

A graph with neither loops nor parallel edges.



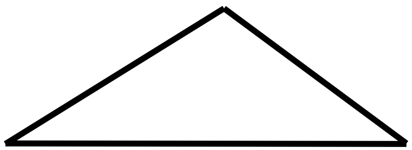
- **weighted graph (权重图)**

A graph where each edge is assigned a numerical label or “weight”.

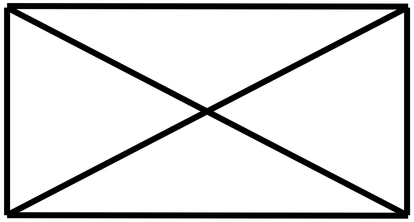


8.1 Introduction 简介

Definition 8.1.9 The **complete graph (完全图)** on n vertices, denote $\boldsymbol{K_n}$, is the simple graph with n vertices in which there is an edge between every pair of distinct vertices.



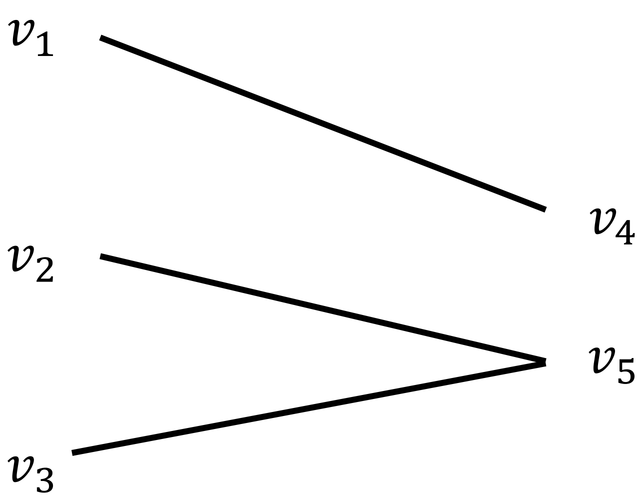
K_3



K_4

8.1 Introduction 简介

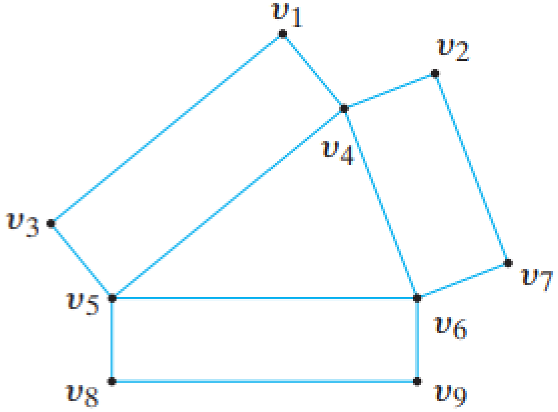
Definition 8.1.11 A graph $G=(V,E)$ is **bipartite (二部图 / 二分图)** if there exist subsets V_1 and V_2 (either possibly empty) of V such that $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$, and each edge in E is incident on one vertex in V_1 and one vertex in V_2.



8.1 Introduction 简介

Definition 8.1.11 A graph $G = (V, E)$ is **bipartite (二部图 / 二分图)** if there exist subsets V_1 and V_2 (either possibly empty) of V such that $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$, and each edge in E is incident on one vertex in V_1 and one vertex in V_2.

Example 8.1.13 Bipartite?



8.1 Introduction 简介

Definition 8.1.11 A graph $G = (V, E)$ is **bipartite (二部图 / 二分图)** if there exist subsets V_1 and V_2 (either possibly empty) of V such that $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$, and each edge in E is incident on one vertex in V_1 and one vertex in V_2.

Example 8.1.13 K_1 is Bipartite?

8.1 Introduction 简介

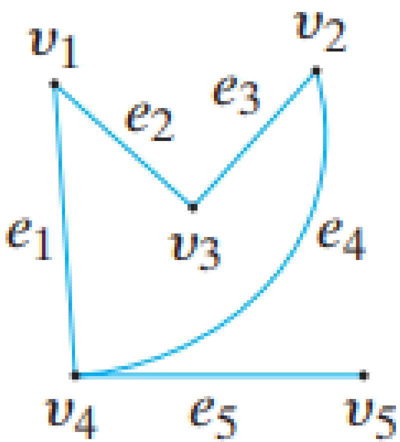
Definition 8.1.15 A **complete bipartite graph (完全二部图)** on m and n vertices, denoted $K_{m,n}$, is the simple graph whose vertex set is partitioned into sets V_1 with m vertices and V_2 with n vertices in which the edge set consists of all edges of the form (v_1, v_2) with $v_1 \in V_1$ and $v_2 \in V_2$.

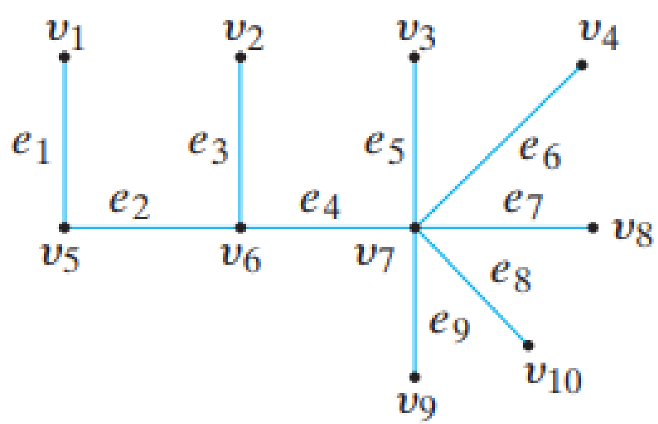
Example 8.1.16 Draw $K_{2,4}$.

8.1 Introduction 简介

Definition 8.1.15 A **complete bipartite graph (完全二部图)** on m and n vertices, denoted $K_{m,n}$, is the simple graph whose vertex set is partitioned into sets V_1 with m vertices and V_2 with n vertices in which the edge set consists of all edges of the form (v_1, v_2) with $v_1 \in V_1$ and $v_2 \in V_2$.

Exercise Draw $K_{2,4}$.





8.1 Introduction 简介

graph (or undirected graph) (无向图)

digraph (or directed graph) (有向图)

incident on (相关联的)

adjacent vertices (相邻顶点)

parallel edges (平行边/并行边)

loop (圈)

isolated vertex (孤立顶点)

simple graph (简单图)

weighted graph (权重图)

complete graph (完全图)

bipartite (二部图 / 二分图)

complete bipartite graph (完全二部图)

8.1 Introduction 简介

graph (or undirected graph) (无向图)

digraph (or directed graph) (有向图)

incident on (相关联的)

adjacent vertices (相邻顶点)

parallel edges (平行边/并行边)

loop (圈)

isolated vertex (孤立顶点)

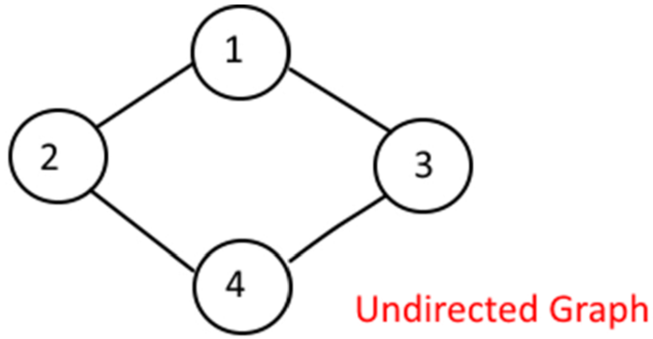
simple graph (简单图)

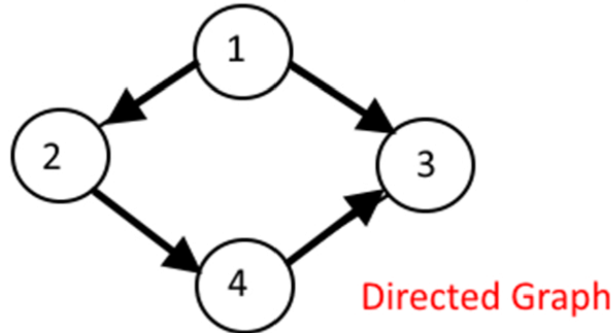
weighted graph (权重图)

complete graph (完全图)

bipartite (二部图 / 二分图)

complete bipartite graph (完全二部图)





8.1 Introduction 简介

graph (or undirected graph) (无向图)

digraph (or directed graph) (有向图)

incident on (相关联的)

adjacent vertices (相邻顶点)

parallel edges (平行边/并行边)

loop (圈)

isolated vertex (孤立顶点)

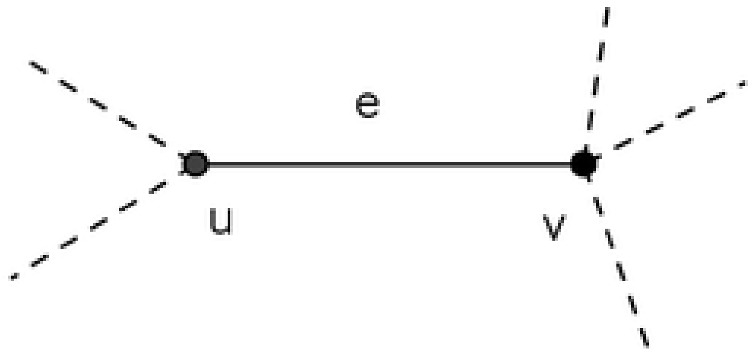
simple graph (简单图)

weighted graph (权重图)

complete graph (完全图)

bipartite (二部图 / 二分图)

complete bipartite graph (完全二部图)



- Two vertices are called **adjacent** if they are connected by an edge.
- Two edges are called **incident**, if they share a vertex.
- A vertex and an edge are called **incident**, if the vertex is one of the two vertices the edge connects.

8.1 Introduction 简介

graph (or undirected graph) (无向图)

digraph (or directed graph) (有向图)

incident on (相关联的)

adjacent vertices (相邻顶点)

parallel edges (平行边/并行边)

loop (圈)

isolated vertex (孤立顶点)

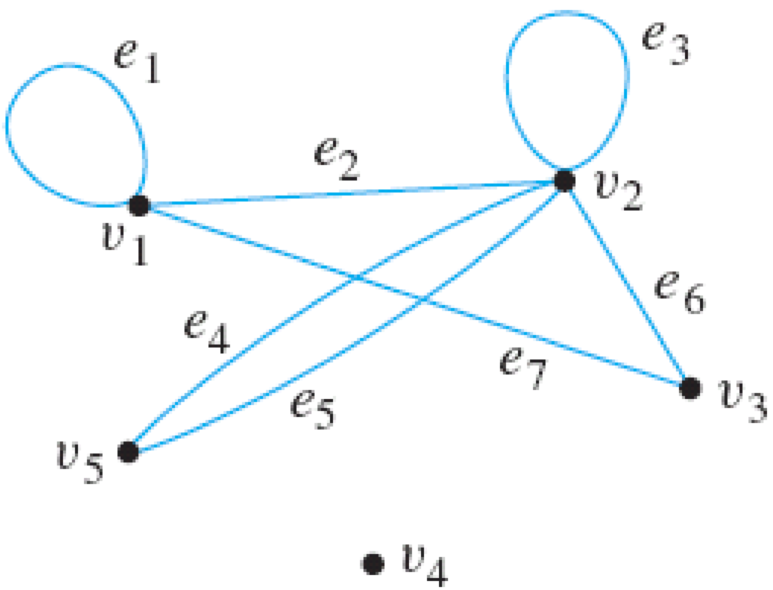
simple graph (简单图)

weighted graph (权重图)

complete graph (完全图)

bipartite (二部图 / 二分图)

complete bipartite graph (完全二部图)



8.1 Introduction 简介

graph (or undirected graph) (无向图)

digraph (or directed graph) (有向图)

incident on (相关联的)

adjacent vertices (相邻顶点)

parallel edges (平行边/并行边)

loop (圈)

isolated vertex (孤立顶点)

simple graph (简单图)

weighted graph (权重图)

complete graph (完全图)

bipartite (二部图 / 二分图)

complete bipartite graph (完全二部图)

8.2 Paths and Cycles 路径和回路

Definition 8.2.1 Let v_0 and v_n be vertices in a graph. A **path (路径)** from v_0 to v_n of length n is an alternating sequence of $n + 1$ vertices and n edges beginning with vertex v_0 and ending with vertex v_n,

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

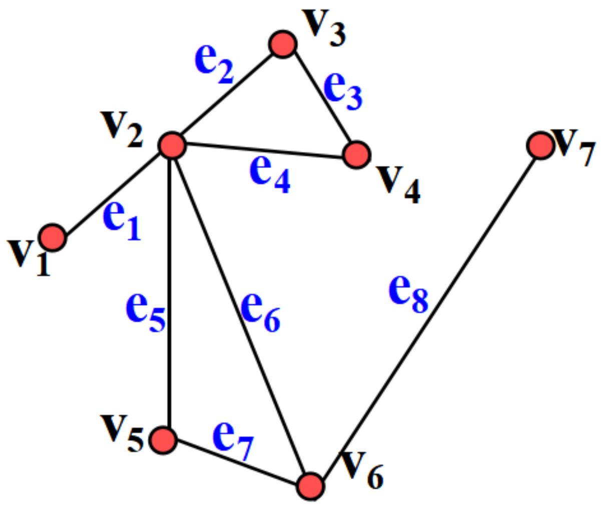
in which edge e_i is incident on vertices v_{i-1} and v_i for $i = 1, \ldots, n$.

8.2 Paths and Cycles 路径和回路

Definition 8.2.1 Let v_0 and v_n be vertices in a graph. A **path (路径)** from v_0 to v_n of length n is an alternating sequence of $n + 1$ vertices and n edges beginning with vertex v_0 and ending with vertex v_n,

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

in which edge e_i is incident on vertices v_{i-1} and v_i for $i = 1, \ldots, n$.



$(v_1, e_1, v_2, e_2, v_3, e_3, v_4)$

$(v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_2)$

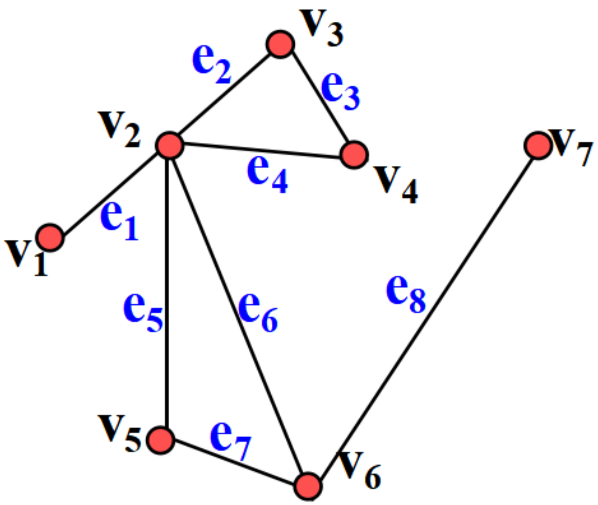
$(v_2, e_2, v_3, e_3, v_4, e_4, v_2)$

8.2 Paths and Cycles 路径和回路

Definition 8.2.1 Let v_0 and v_n be vertices in a graph. A **path (路径)** from v_0 to v_n of length n is an alternating sequence of $n + 1$ vertices and n edges beginning with vertex v_0 and ending with vertex v_n,

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

in which edge e_i is incident on vertices v_{i-1} and v_i for $i = 1, \ldots, n$.



$(v_1, e_1, v_2, e_2, v_3, e_3, v_4)$

$(v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_2)$

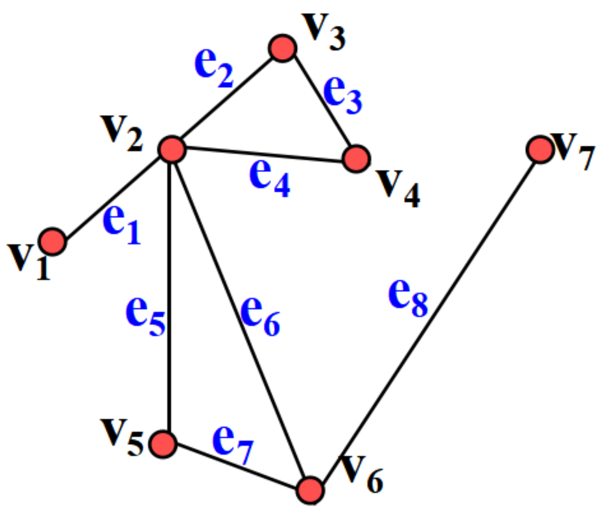
$(v_2, e_2, v_3, e_3, v_4, e_4, v_2)$ closed path

8.2 Paths and Cycles 路径和回路

Definition 8.2.1 Let v_0 and v_n be vertices in a graph. A **path (路径)** from v_0 to v_n of length n is an alternating sequence of $n + 1$ vertices and n edges beginning with vertex v_0 and ending with vertex v_n,

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

in which edge e_i is incident on vertices v_{i-1} and v_i for $i = 1, \ldots, n$.



$(v_1, e_1, v_2, e_2, v_3, e_3, v_4)$

$(v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_2)$

$(v_2, e_2, v_3, e_3, v_4, e_4, v_2)$ closed path

(v_6)

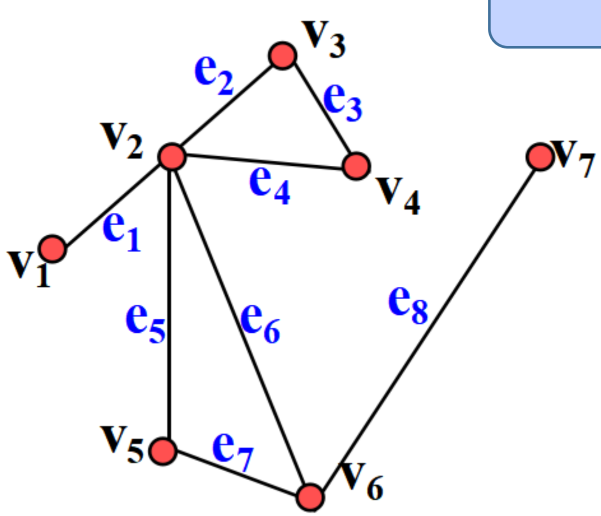
8.2 Paths and Cycles 路径和回路

Definition 8.2.1 Let v_0 and v_n be vertices in a graph. A **path (路径)** from v_0 to v_n of length n is an alternating sequence of $n + 1$ vertices and n edges beginning with vertex v_0 and ending with vertex v_n,

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

in which edge e_i is incident on vertices v_{i-1} and v_i for $i = 1, \ldots, n$.

In absence of parallel edges, in denoting a path we may suppress the edges.



$(v_1, e_1, v_2, e_2, v_3, e_3, v_4) \rightarrow (v_1, v_2, v_3, v_4)$

$(v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_2)$

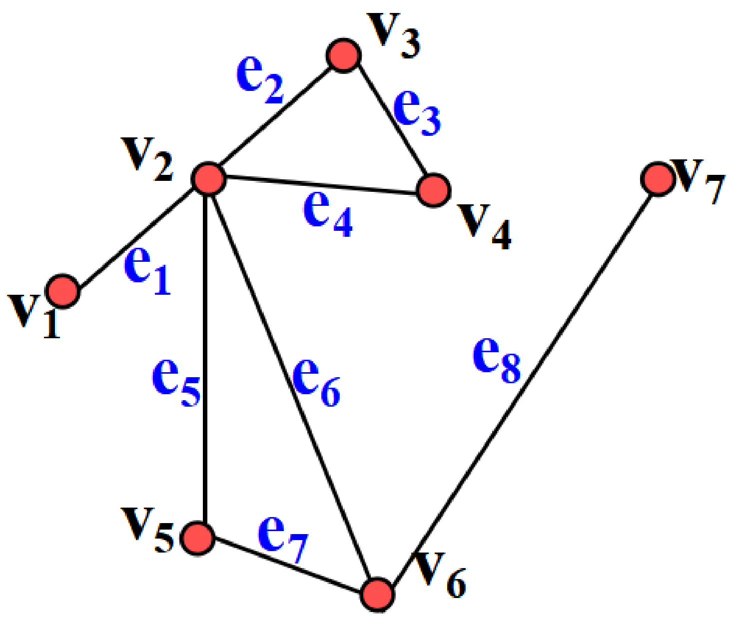
$(v_2, e_2, v_3, e_3, v_4, e_4, v_2)$ closed path

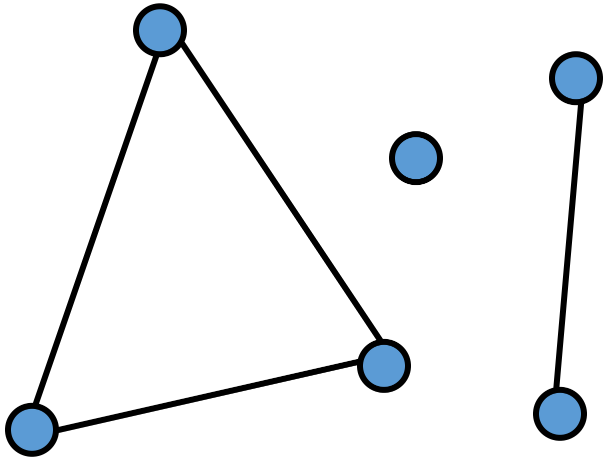
(v_6)

8.2 Paths and Cycles 路径和回路

connected graph (连通图)

Definition 8.2.4 A graph G is **connected (连通的)** if given any vertices v and w in G, there is a path from v to w.





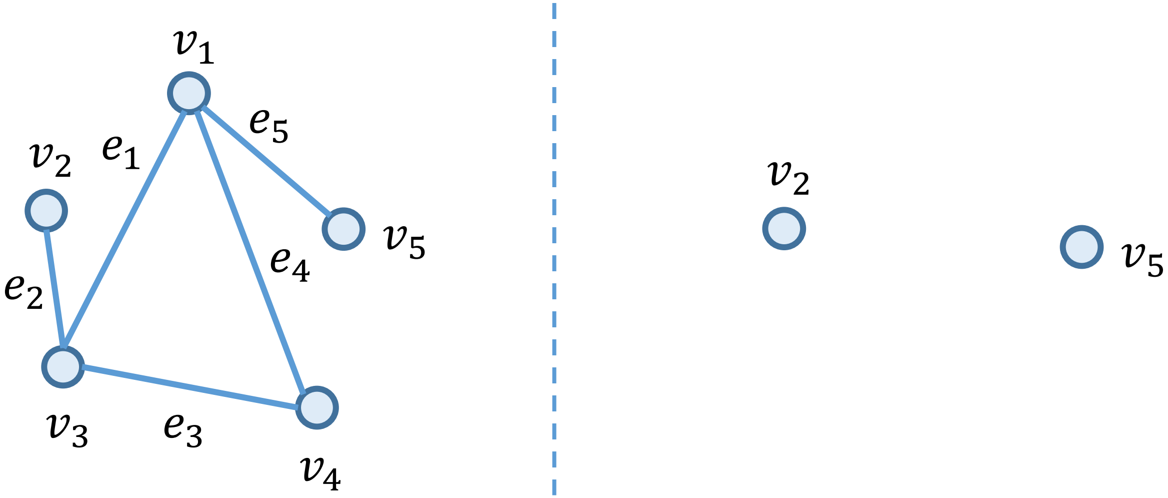
8.2 Paths and Cycles 路径和回路

Definition 8.2.8 Let $G = (V, E)$ be a graph, we call (V', E') a **subgraph (子图)** G if

(a) $V' \quad V$ and $E' \subseteq E$.

(b) For every edge $e' \in E'$, if e' is incident on v' and w', then $v', w' \in V'$.

Example



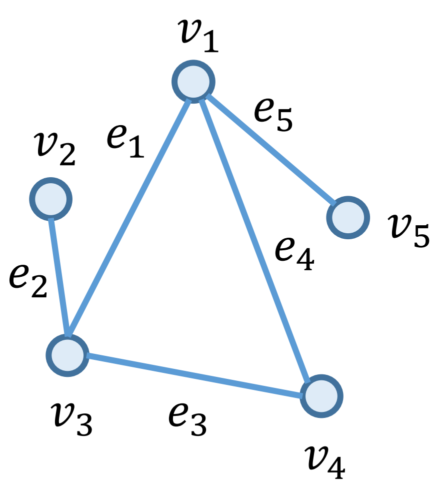
8.2 Paths and Cycles 路径和回路

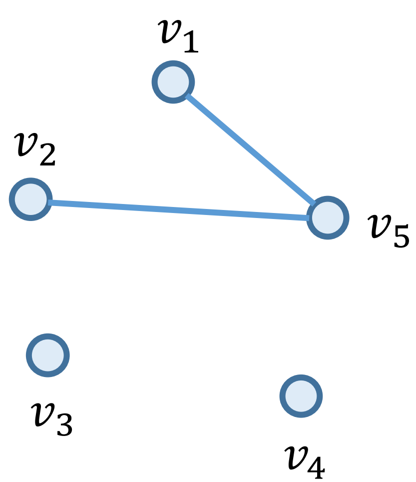
Definition 8.2.8 Let $G = (V, E)$ be a graph, we call (V', E') a **subgraph (子图)** G if

(a) V' V and $E' \subseteq E$.

(b) For every edge $e' \in E'$, if e' is incident on v' and w', then $v', w' \in V'$.

Example





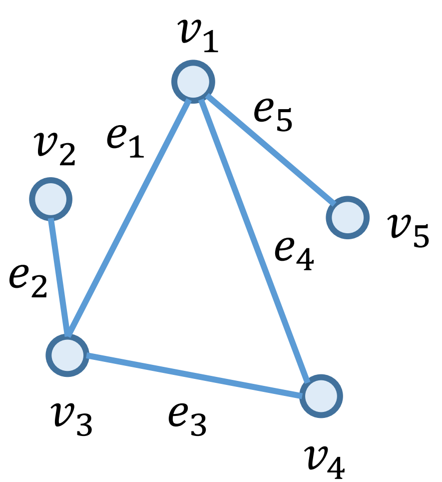
8.2 Paths and Cycles 路径和回路

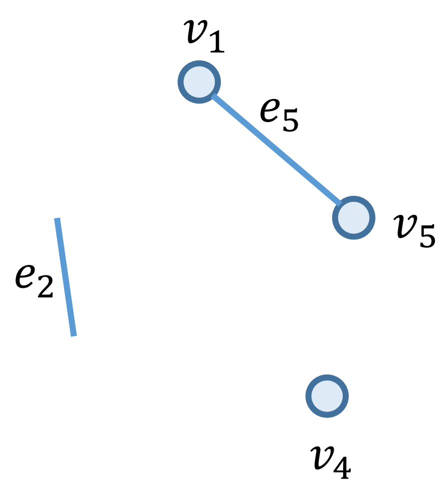
Definition 8.2.8 Let $G = (V, E)$ be a graph, we call (V', E') a **subgraph (子图)** G if

(a) V' V and $E' \subseteq E$.

(b) For every edge $e' \in E'$, if e' is incident on v' and w' , then $v', w' \in V'$.

Example





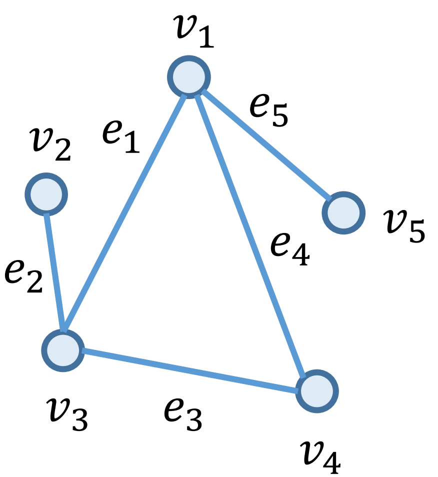
8.2 Paths and Cycles 路径和回路

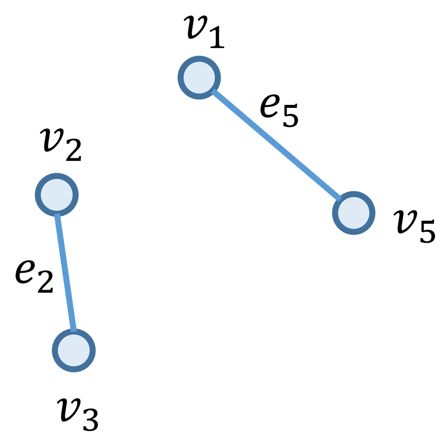
Definition 8.2.8 Let $G = (V, E)$ be a graph, we call (V', E') a **subgraph (子图)** G if

(a) V' V and $E' \subseteq E$.

(b) For every edge $e' \in E'$, if e' is incident on v' and w', then $v', w' \in V'$.

Example





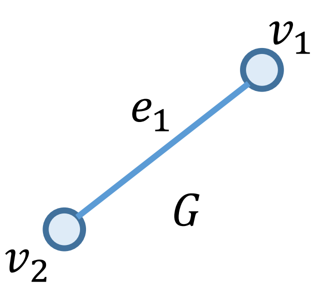
8.2 Paths and Cycles 路径和回路

Definition 8.2.8 Let $G = (V, E)$ be a graph, we call (V', E') a **subgraph (子图)** G if

(a) V' V and $E' \subseteq E$.

(b) For every edge $e' \in E'$, if e' is incident on v' and w', then $v', w' \in V'$.

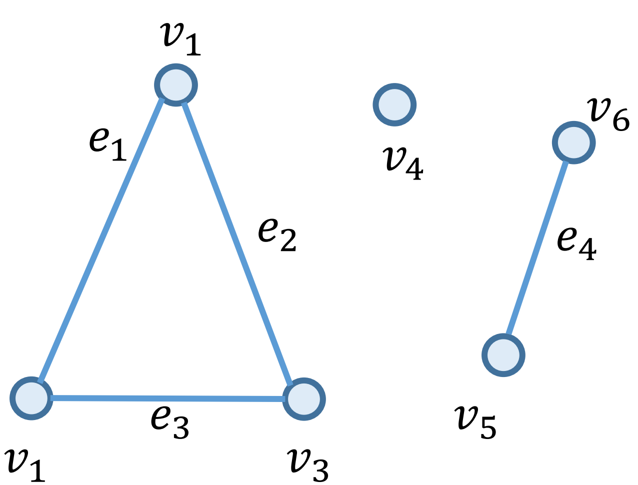
Example 8.2.10 Find all subgraphs of the graph G having at least one vertex.



8.2 Paths and Cycles 路径和回路

Definition 8.2.11 Let G be a graph and let v be a vertex in G. The subgraph G' of G consisting of all edges and vertices in G that are contained in some path beginning at v is called the **component (分支)** of G containing v.

Example 8.2.13 The components of G



The component of G containing v_3?

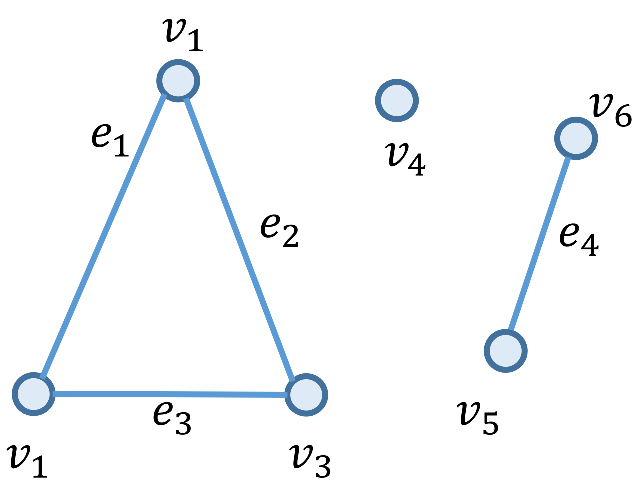
The component of G containing v_3?

The component of G containing v_5?

8.2 Paths and Cycles 路径和回路

Definition 8.2.11 Let G be a graph and let v be a vertex in G. The subgraph G' of G consisting of all edges and vertices in G that are contained in some path beginning at v is called the **component (分支)** of G containing v.

Example 8.2.13 The components of G

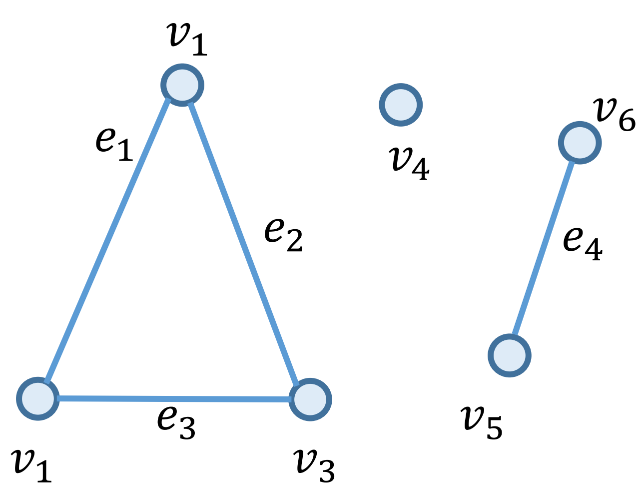


A graph is **connected** if and only if it has only **1 connected component**.

8.2 Paths and Cycles 路径和回路

Definition 8.2.11 Let G be a graph and let v be a vertex in G. The subgraph G' of G consisting of all edges and vertices in G that are contained in some path beginning at v is called the **component (分支)** of G containing v.

Example 8.2.13 The components of G



If we define a relation R on the set of vertices V by the rule

vRw if there is a path from v to w

then R is equivalence relation on V .

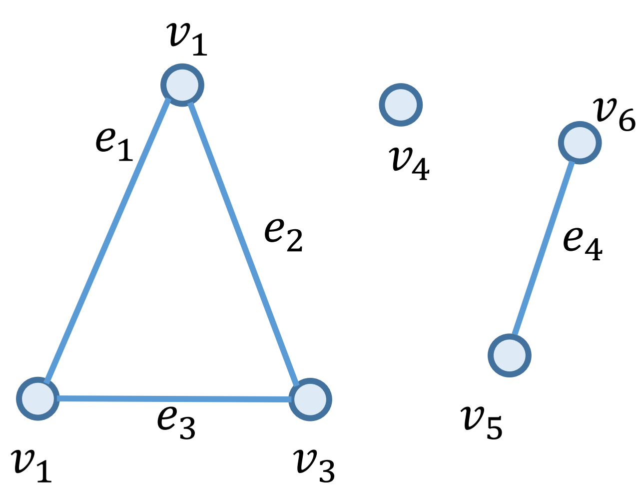
If $v \in V$, the set of vertices in the componeent containing v is the equivalence class

$$[v] = \{w \in V \mid wRv\}.$$

8.2 Paths and Cycles 路径和回路

Definition 8.2.11 Let G be a graph and let v be a vertex in G. The subgraph G' of G consisting of all edges and vertices in G that are contained in some path beginning at v is called the **component (分支)** of G containing v.

Example 8.2.13 The components of G



If we define a relation R on the set of vertices V by the rule

vRw if there is a path from v to w

then R is equivalence relation on V .

$[v_1] =$
$[v_4] =$
$[v_5] =$

8.2 Paths and Cycles 路径和回路

Definition 8.2.14 Let v and w be vertices in a graph G.

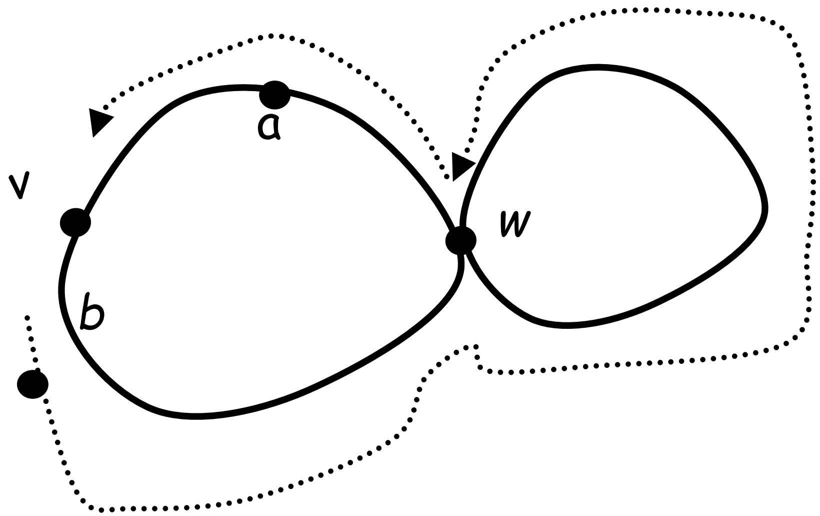
- A **simple path（简单路径）** from v to w is a path from v to w with no repeated vertices.

No repeated edges?

8.2 Paths and Cycles 路径和回路

Definition 8.2.14 Let v and w be vertices in a graph G.

- A **simple path（简单路径）** from v to w is a path from v to w with no repeated vertices.
- A **cycle (or circuit) (回路或者环路)** is a path of nonzero length from v to v with no repeated edges.

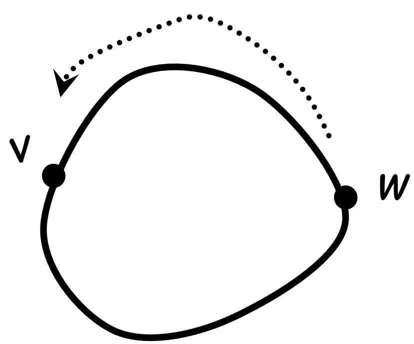


cycle: $v \cdots b \cdots w \cdots w \cdots a \cdots v$

8.2 Paths and Cycles 路径和回路

Definition 8.2.14 Let v and w be vertices in a graph G.

- A **simple path（简单路径）** from v to w is a path from v to w with no repeated vertices.
- A **cycle (or circuit) (回路或者环路)** is a path of nonzero length from v to v with no repeated edges.
- A **simple cycle（简单回路）** is a cycle from v to v in which, except for the beginning and ending vertices that are both equal to v, there are no repeated vertices.



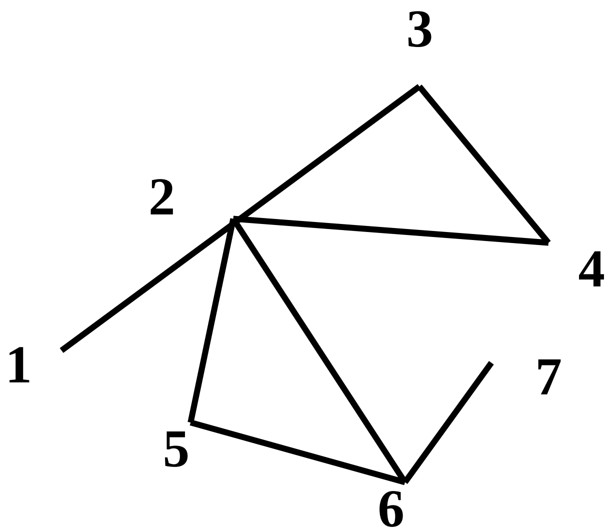
cycle: $v \ \cdots w \ \cdots$
v

In a simple cycle, every vertex is of degree exactly 2.

8.2 Paths and Cycles 路径和回路

Definition 8.2.14 Let v and w be vertices in a graph G.

- A **simple path(简单路径)** from v to w is a path from v to w with no repeated vertices.
- A **cycle (or circuit) (回路或者环路)** is a path of nonzero length from v to v with no repeated edges.
- A **simple cycle(简单回路)** is a cycle from v to v in which, except for the beginning and ending vertices that are both equal to v, there are no repeated vertices.

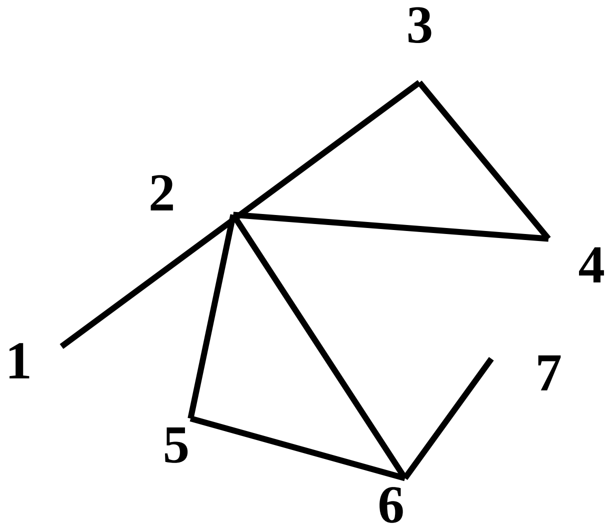


Path	Simple path	Cycle	Simple cycle
6 5 2 4 3 2 1			
6 5 2 4			
2 6 5 2 4 3 2			
5 6 2 5			
7			

8.2 Paths and Cycles 路径和回路

Definition 8.2.14 Let v and w be vertices in a graph G.

- A **simple path（简单路径）** from v to w is a path from v to w with no repeated vertices.
- A **cycle (or circuit) (回路或者环路)** is a path of nonzero length from v to v with no repeated edges.
- A **simple cycle（简单回路）** is a cycle from v to v in which, except for the beginning and ending vertices that are both equal to v, there are no repeated vertices.



Path	Simple path	Cycle	Simple cycle
6 5 2 4 3 2 1	no	no	no
6 5 2 4	yes	no	no
2 6 5 2 4 3 2	no	yes	no
5 6 2 5	no	yes	yes
7	yes	no	no

8.2 Paths and Cycles 路径和回路

Eulerian Path (欧拉路径)

Euler Cycle (欧拉回路): a cycle in a graph G that includes all of the edges and all of the vertices of G.

The **degree of a vertex (顶点度)** v, denoted by $\delta(v)$, is the number of edges incident on v.

Each loop on v contributes 2 to the degree of v.

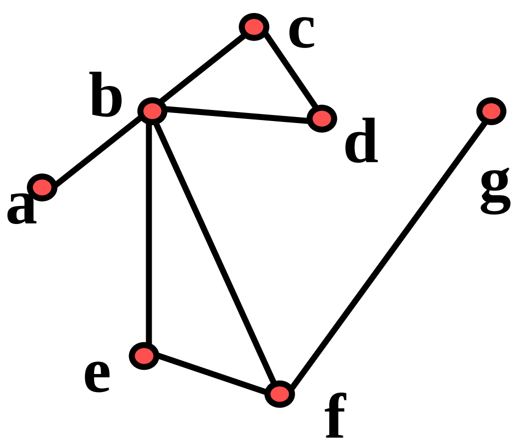
8.2 Paths and Cycles 路径和回路

Eulerian Path (欧拉路径)

Euler Cycle (欧拉回路)： a cycle in a graph G that includes all of the edges and all of the vertices of G.

The **degree of a vertex (顶点度)** v, denoted by $\delta(v)$, is the number of edges incident on v.

Each loop on v contributes 2 to the degree of v.



δ(a) =
δ(b) =
δ(c) =
δ(d) =
δ(e) =
δ(f) =
δ(g) =

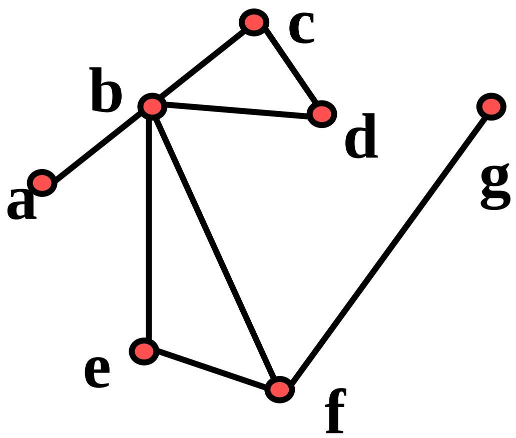
8.2 Paths and Cycles 路径和回路

Eulerian Path (欧拉路径)

Euler Cycle (欧拉回路): a cycle in a graph G that includes all of the edges and all of the vertices of G.

The **degree of a vertex (顶点度)** v, denoted by $\delta(v)$, is the number of edges incident on v.

Each loop on v contributes 2 to the degree of v.



δ(a) = 1,
δ(b) = 5,
δ(c) = 2,
δ(d) = 2,
δ(e) = 2,
δ(f) = 3,
δ(g) =1.

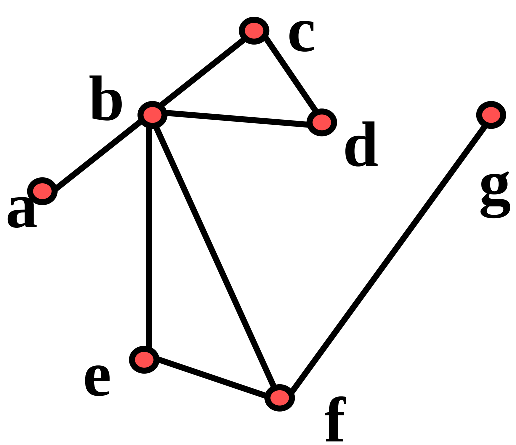
8.2 Paths and Cycles 路径和回路

Eulerian Path (欧拉路径)

Euler Cycle (欧拉回路): a cycle in a graph G that includes all of the edges and all of the vertices of G.

The **degree of a vertex (顶点度)** v, denoted by $\delta(v)$, is the number of edges incident on v.

Each loop on v contributes 2 to the degree of v.



δ(a) = 1,
δ(b) = 5,
δ(c) = 2,
δ(d) = 2,
δ(e) = 2,
δ(f) = 3,
δ(g) =1.

The ***neighbour set*** $N(v)$ of a vertex v is the set of vertices adjacent to it.

The degree of a vertex v = the number of neighbours of v?

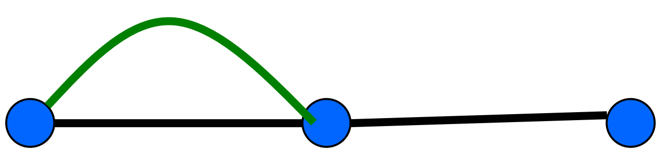
8.2 Paths and Cycles 路径和回路

Eulerian Path (欧拉路径)

Euler Cycle (欧拉回路): a cycle in a graph G that includes all of the edges and all of the vertices of G.

The **degree of a vertex (顶点度)** v, denoted by $\delta(v)$, is the number of edges incident on v.

Each loop on v contributes 2 to the degree of v.



NO!

The ***neighbour set*** $N(v)$ of a vertex v is the set of vertices adjacent to it.

The degree of a vertex v = the number of neighbours of v?

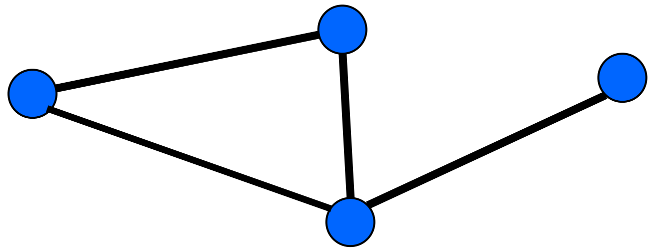
8.2 Paths and Cycles 路径和回路

Eulerian Path (欧拉路径)

Euler Cycle (欧拉回路)： a cycle in a graph G that includes all of the edges and all of the vertices of G.

The **degree of a vertex (顶点度)** v, denoted by $\delta(v)$, is the number of edges incident on v.

Each loop on v contributes 2 to the degree of v.



Yes for any simple graph.

The ***neighbour set*** $N(v)$ of a vertex v is the set of vertices adjacent to it.

The degree of a vertex v = the number of neighbours of v?

8.2 Paths and Cycles 路径和回路

Eulerian Path (欧拉路径)

Euler Cycle (欧拉回路)： a cycle in a graph G that includes all of the edges and all of the vertices of G.

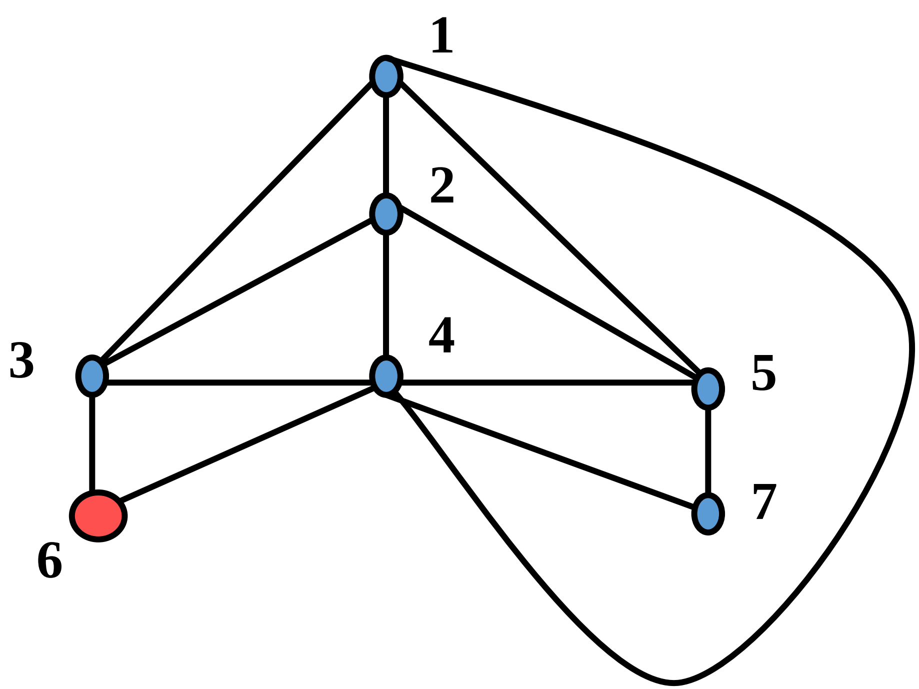
The **degree of a vertex (顶点度)** v, denoted by $\delta(v)$, is the number of edges incident on v.

Each loop on v contributes 2 to the degree of v.

Theorem 8.2.7 If a graph G has an Euler cycle, then G is connected and every vertex has even degree.

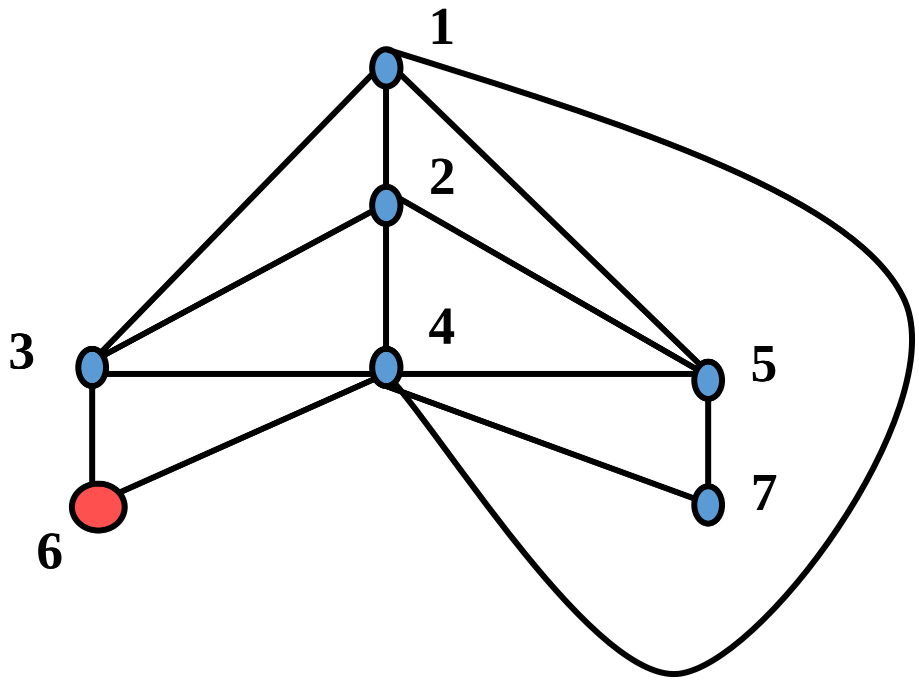
8.2 Paths and Cycles 路径和回路

Theorem 8.2.18 If G is a connected graph and every vertex has even degree, then G has an Euler cycle.



8.2 Paths and Cycles 路径和回路

Theorem 8.2.18 If G is a connected graph and every vertex has even degree, then G has an Euler cycle.



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

8.2 Paths and Cycles 路径和回路

A graph G is an **Euler graph (欧拉图)** if it has an Euler cycle.

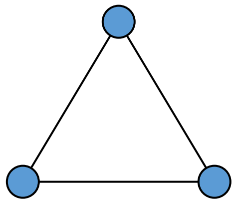
Theorems 8.2.17 and 8.2.18: G is an Euler graph if and only if G is connected and all its vertices have even degree.

Theorem 8.2.7 If a graph G has an Euler cycle, then G is connected and every vertex has even degree.

Theorem 8.2.18 If G is a connected graph and every vertex has even degree, then G has an Euler cycle.

8.2 Paths and Cycles 路径和回路

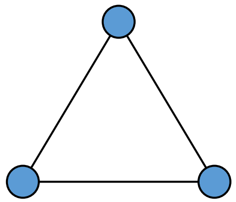
Is there a graph with degree sequence (2, 2, 2)? YES.



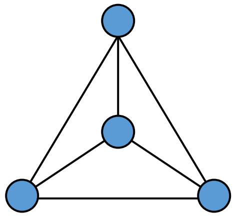
Is there a graph with degree sequence (3, 3, 3, 3)?

8.2 Paths and Cycles 路径和回路

Is there a graph with degree sequence (2, 2, 2)? YES.

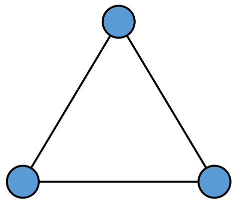


Is there a graph with degree sequence (3, 3, 3, 3)? YES.

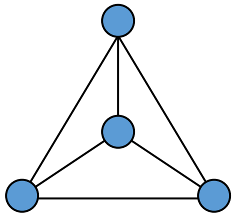


8.2 Paths and Cycles 路径和回路

Is there a graph with degree sequence (2, 2, 2)? YES.



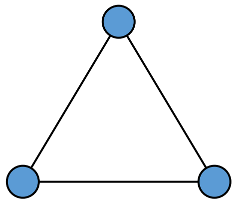
Is there a graph with degree sequence (3, 3, 3, 3)? YES.



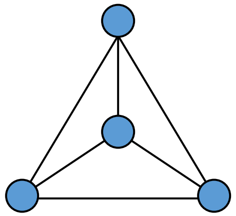
Is there a graph with degree sequence (2, 2, 1)?

8.2 Paths and Cycles 路径和回路

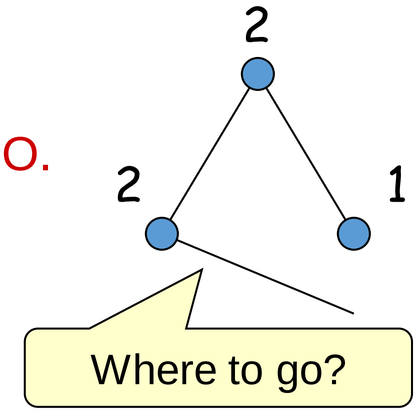
Is there a graph with degree sequence (2, 2, 2)? YES.



Is there a graph with degree sequence (3, 3, 3, 3)? YES.

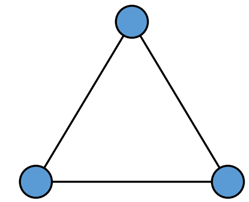


Is there a graph with degree sequence (2, 2, 1)? NO.

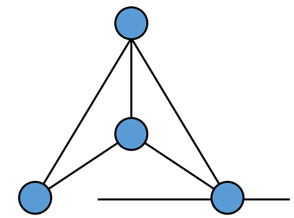


8.2 Paths and Cycles 路径和回路

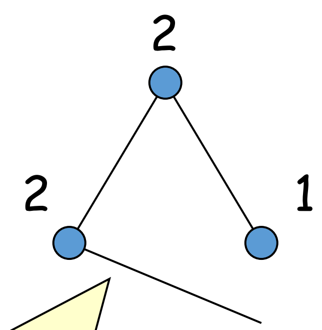
Is there a graph with degree sequence (2, 2, 2)? YES.



Is there a graph with degree sequence (3, 3, 3, 3)? YES.



Is there a graph with degree sequence (2, 2, 1)? NO.



Is there a graph with degree sequence (2, 2, 2, 2, 1)?

Where to go?

8.2 Paths and Cycles 路径和回路

Theorem 8.2.21 The handshaking theorem

If several people shake hands, the total number of hands shake must be even.

8.2 Paths and Cycles 路径和回路

Theorem 8.2.21 The handshaking theorem

If G is a graph with m edges and n vertices $v_1, v_2, \ldots, vn$, then

$$\sum_{i=1}^{n} \delta(v_i) = 2m$$

In particular, the sum of the degrees of all the vertices of a graph is even.

If several people shake hands, the total number of hands shake must be even.

8.2 Paths and Cycles 路径和回路

Theorem 8.2.21 The handshaking theorem

If G is a graph with m edges and n vertices $v_1, v_2, \ldots, vn$, then

$$\sum_{i=1}^{n} \delta(v_i) = 2m$$

In particular, the sum of the degrees of all the vertices of a graph is even.

Proof When we sum over the degrees of all the vertices, we count each edge (vi, vj) twice—once when we count it as (vi, vj) in the degree of vi and again when we count it as (vj, vi) in the degree of vj. The conclusion follows.

If several people shake hands, the total number of hands shake must be even.

8.2 Paths and Cycles 路径和回路

Theorem 8.2.21 The handshaking theorem

If G is a graph with m edges and n vertices $v_1, v_2, \ldots, vn$, then

$$\sum_{i=1}^{n} \delta(v_i) = 2m$$

In particular, the sum of the degrees of all the vertices of a graph is even.

Examples

Is there a graph with degree sequence (2, 2, 1)?

2+2+1 = odd, so impossible.

Is there a graph with degree sequence (2, 2, 1)?

2+2+2+2+1 = odd, so impossible.

If several people shake hands, the total number of hands shake must be even.

8.2 Paths and Cycles 路径和回路

Theorem 8.2.21 The handshaking theorem

If G is a graph with m edges and n vertices $v_1, v_2, \ldots, vn$, then

$$\sum_{i=1}^{n} \delta(v_i) = 2m$$

In particular, the sum of the degrees of all the vertices of a graph is even.

Examples

Is there a graph with degree sequence (1, 2, 3)?

Question. Given a degree sequence, if the sum of degree is even, is it true that there is a simple graph with such a degree sequence?

8.2 Paths and Cycles 路径和回路

Theorem 8.2.21 The handshaking theorem

If G is a graph with m edges and n vertices $v_1, v_2, \ldots, vn$, then

$$\sum_{i=1}^{n} \delta(v_i) = 2m$$

In particular, the sum of the degrees of all the vertices of a graph is even.

Examples

Is there a graph with degree sequence (1, 2, 3)?

NO! (3, 3, 3, 1), (3, 5)...

Question. Given a degree sequence, if the sum of degree is even, is it true that there is a simple graph with such a degree sequence?

8.2 Paths and Cycles 路径和回路

Theorem 8.2.21 The handshaking theorem

If G is a graph with m edges and n vertices $v_1, v_2, \ldots, vn$, then

$$\sum_{i=1}^{n} \delta(v_i) = 2m$$

In particular, the sum of the degrees of all the vertices of a graph is even.

Examples

Is there a graph with degree sequence (1, 2, 3)?

NO! (3, 3, 3, 1), (3, 5)...

Question. Given a degree sequence, if the sum of degree is even, is it true that there is a ~~simple~~ graph with such a degree sequence?

8.2 Paths and Cycles 路径和回路

Theorem 8.2.21 The handshaking theorem

If G is a graph with m edges and n vertices $v_1, v_2, \ldots, vn$, then

$$\sum_{i=1}^{n} \delta(v_i) = 2m$$

In particular, the sum of the degrees of all the vertices of a graph is even.

Examples

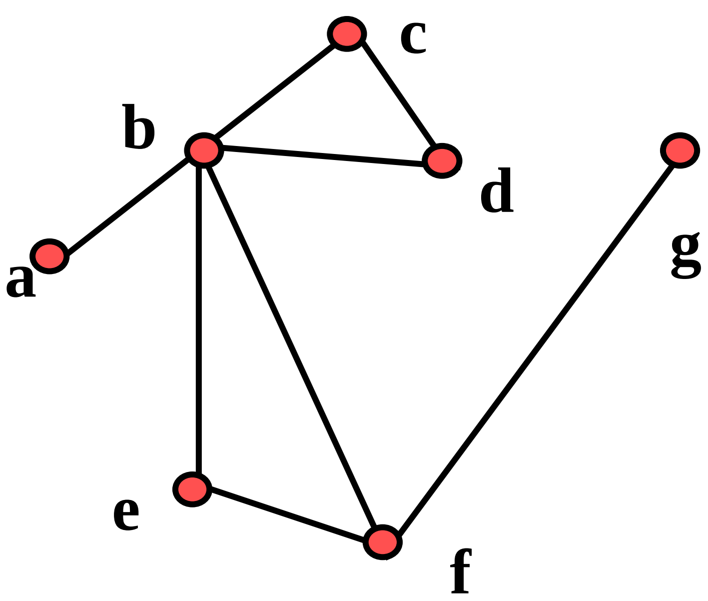
Is there a graph with degree sequence (1, 2, 3)?

YES!

Question. Given a degree sequence, if the sum of degree is even, is it true that there is a ~~simple~~ graph with such a degree sequence?

8.2 Paths and Cycles 路径和回路

Corollary 8.2.22 In any graph, the number of vertices of odd degree is even.



$\delta(a) = 1$,
$\delta(b) = 5$,
$\delta(c) = 2$,
$\delta(d) = 2$,
$\delta(e) = 2$,
$\delta(f) = 3$,
$\delta(g) = 1$.

8.2 Paths and Cycles 路径和回路

Corollary 8.2.22 In any graph, the number of vertices of odd degree is even.

Proof Let us divide the vertices into tow groups: those with even degree $x_1, \ldots, x_m$ and those with odd degree $y_1, \ldots, y_n$.
Let $S = \delta(x_1) + \delta(x_2) + \ldots + \delta(x_m)$, $T = \delta(y_1) + \delta(y_2) + \ldots + \delta(y_n)$.
By **Theorem 8.2.21 (The handshaking theorem)**, $S + T$ is even. Since S is the sum of even numbers, S is even. Thus T is even. But T is the sum of n odd numbers, and therefore n is even.

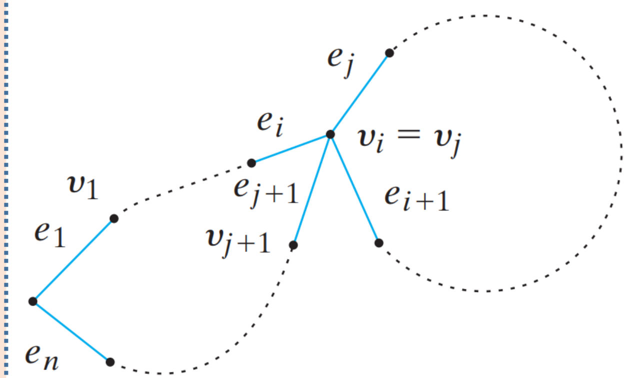
8.2 Paths and Cycles 路径和回路

Theorem 8.2.23 A graph has a path with no repeated edges from v to w ($v \neq w$) containing all the edges and vertices if and only if it is connected and v and w are the only vertices having odd degree.

8.2 Paths and Cycles 路径和回路

Theorem 8.2.24 If a graph G contains a cycle from v to v, G contains a simple cycle from v to v.

Proof Let
$C = (v_0, e_1, v_1, \ldots, e_i, v_i, e_{i+1}, \ldots, e_j, v_j, e_{j+1}, v_{j+1}, \ldots, e_n, v_n)$
be a cycle from v to v where $v = v_0 = v_n$. If C is not a simple cycle, then $v_i = v_j$, for some $i < j < n$. We can replace C by the cycle
$C \quad = (v_0, e_1, v_1, \ldots, e_i, v_i, e_{j+1}, v_{j+1}, \ldots, e_n, v_n)$
If C is not a simple cycle from v to v, we repeat the previous procedure. Eventually we obtain a simple cycle from v to v.



A **cycle (or circuit) (回路或者环路)** is a path of nonzero length from v to v with no repeated edges.

Problem-Solving Corner

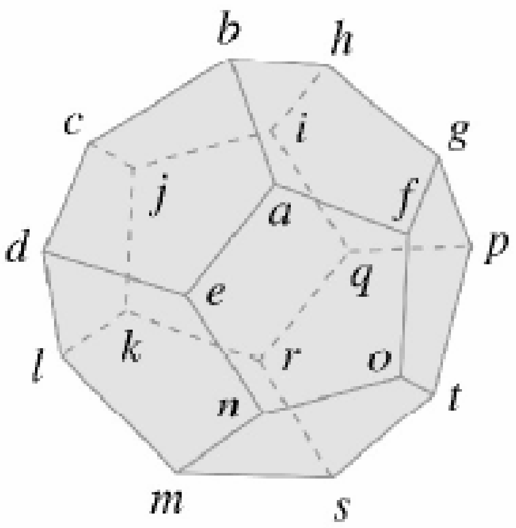
Is it possible in a department of 25 person，racked by dissension, for each person to get along with exactly four others?

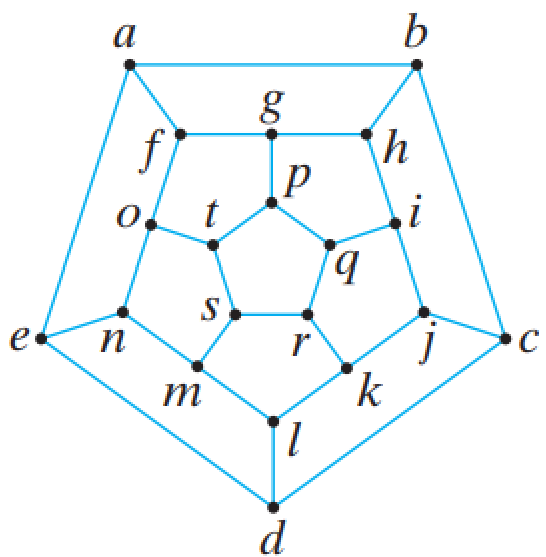
Is it possible in a department of 25 person，racked by dissension, for each person to get along with exactly four others

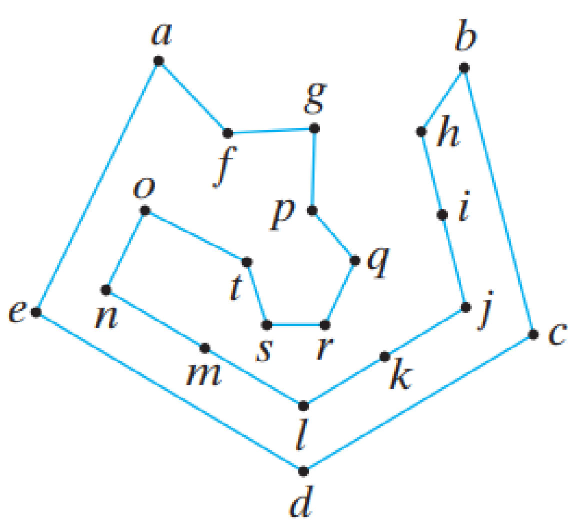
8.3 Hamiltonian Cycles and the Traveling Salesperson Problem
哈密顿回路和旅行商问题

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

Hamilton’s puzzle (哈密顿难题): Can we find a cycle in the graph of the dedecahedron that contains each vertex exactly once？







8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

- A **cycle (or circuit) (回路或者环路)** is a path of nonzero length from v to v with no repeated edges.
- **Euler Cycle (欧拉回路)：** a cycle in a graph G that includes all of the edges and all of the vertices of G.

8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

(Unlike the situation for Euler cycles, no easily verified necessary and sufficient conditions are known for the existence of a Hamiltonian cycle in a graph.)

Theorems 8.2.17 and 8.2.18: G is an Euler graph if and only if G is connected and all its vertices have even degree.

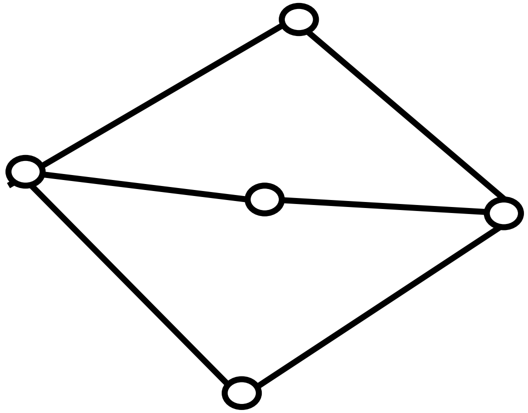
- A **cycle (or circuit) (回路或者环路)** is a path of nonzero length from v to v with no repeated edges.
- **Euler Cycle (欧拉回路)**: a cycle in a graph G that includes all of the edges and all of the vertices of G.

8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

(Unlike the situation for Euler cycles, no easily verified necessary and sufficient conditions are known for the existence of a Hamiltonian cycle in a graph.)

Example 8.3.2 Does the following graph contain a Hamiltonian cycle?

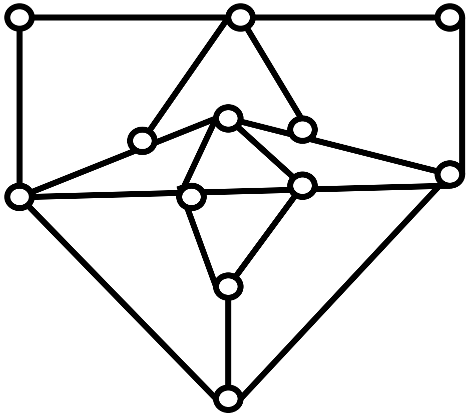


8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

(Unlike the situation for Euler cycles, no easily verified necessary and sufficient conditions are known for the existence of a Hamiltonian cycle in a graph.)

Example 8.3.3 Does the following graph contain a Hamiltonian cycle?

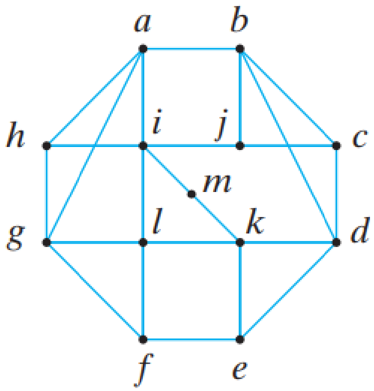


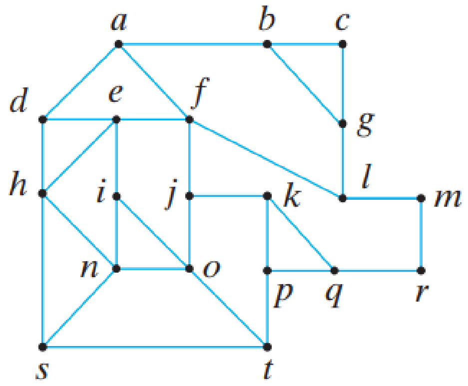
8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

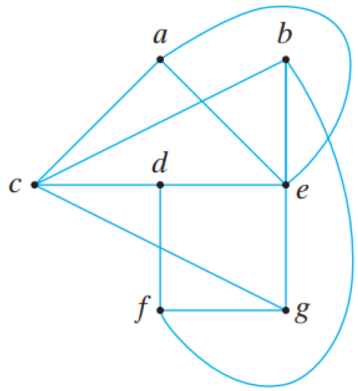
Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

(Unlike the situation for Euler cycles, no easily verified necessary and sufficient conditions are known for the existence of a Hamiltonian cycle in a graph.)

Exercise Does the following graphs contain a Hamiltonian cycle?







8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

Euler Cycle (欧拉回路)： a cycle in a graph G that includes all of the edges and all of the vertices of G.

Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

1. Given an example of a graph that has an Euler cycle and a Hamiltonian cycle.

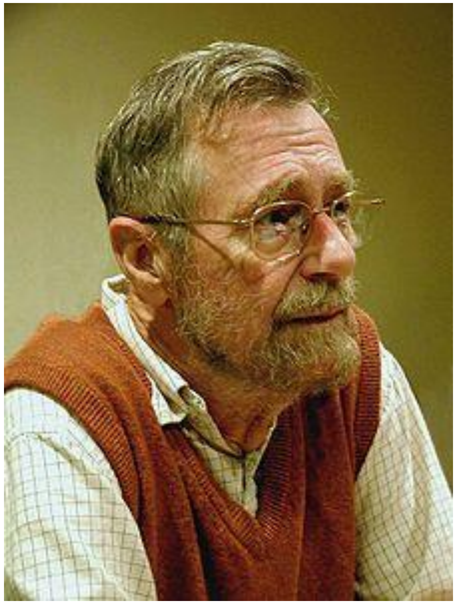
2. Given an example of a graph that has an Euler cycle but not a Hamiltonian cycle.

3. Given an example of a graph that has a Hamiltonian cycle but not an Euler cycle.

4. Given an example of a graph that has neither an Euler cycle nor a Hamiltonian cycle.

8.4 A Shortest-Path Algorithm 最短路径算法

Due to **Edsger W. Dijkstra**, Dutch computer scientist born in 1930



Edsger W. Dijkstra (1930–2002) was born in The Netherlands. He was an early proponent of programming as a science. So dedicated to programming was he that when he was married in 1957, he listed his profession as a programmer. However, the Dutch authorities said that there was no such profession, and he had to change the entry to "theoretical physicist." He won the prestigious Turing Award in 1972.

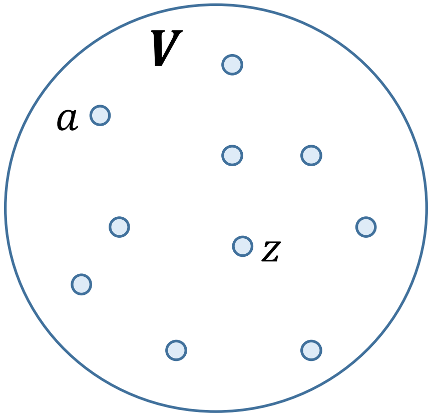
Dijkstra's algorithm (狄克斯特拉算法) finds the length of the shortest path from a single vertex to any other vertex in a connected weighted graph.

Dijkstra's algorithm (狄克斯特拉算法)

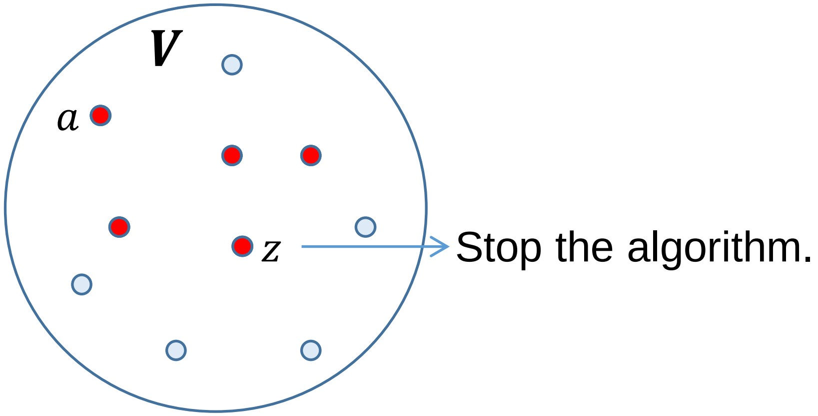
The given graph G is **connected, weighted graph**. Assume that **the weights are positive numbers**. We want to find a shortest path from vertex a to vertex z.

Dijkstra's algorithm (狄克斯特拉算法)

The given graph G is **connected, weighted graph**. Assume that **the weights are positive numbers**. We want to find a shortest path from vertex a to vertex z.



Initialization. Temporarily labelled each vertex $v \in V$ with a value $L(v)$.



Each iteration changes the status of one temporairly labelled vertex from temporary to permanent. Update the label of some related vertices.

Dijkstra's algorithm (狄克斯特拉算法)

The given graph G is **connected, weighted graph**. Assume that **the weights are positive numbers**. We want to find a shortest path from vertex a to vertex z.

This algorithm finds the length of a shortest path from vertex a to vertex z in a connected, weighted graph. The weight of edge (i, j) is $w(i, j) > 0$ and the label of vertex x is $L(x)$. At termination, $L(z)$ is the length of a shortest path from a to z.

Dijkstra's algorithm (狄克斯特拉算法)

Input: A connected, weighted graph in which all weights are positive; vertices a and z.

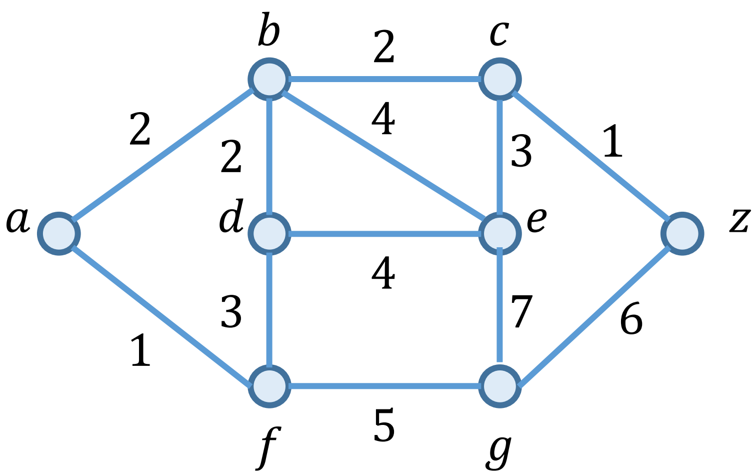
Output: $L(z)$, the length of a shortest path from a to z.

```
procedure dijkstra(w, a, z, L)
    L(a) := 0
    for each node x≠a do
    L(x) :=  ∞
   T := set of all nodes
   //T is the set of vertices whose shortest
   // distance from a has not been found
  while z   T do
       chose v   T with minimum L(v)
       T := T − {v}
       for each x   T adjacent to v do
          L(x) := min {L(x), L(v) + w(v, x)}
      end
   end dijkstra
```

Dijkstra's algorithm (狄克斯特拉算法)

Example 8.4.2

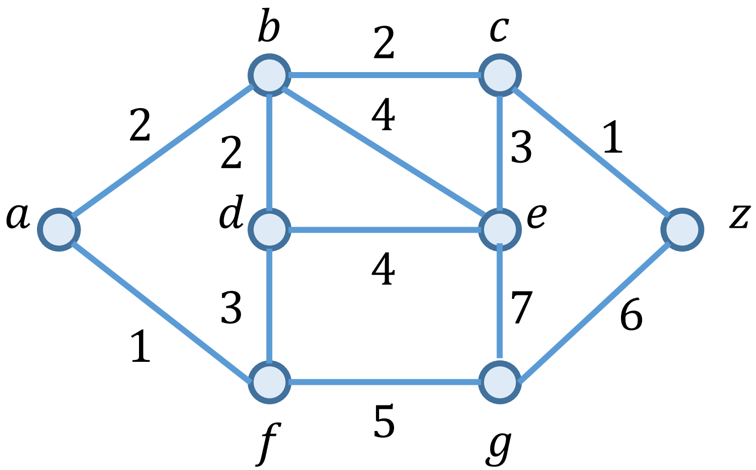
procedure $dijkstra(w, a, z, L)$
  $L(a) := 0$
  for each node $x \neq a$ **do**
  $L(x) := \infty$
 T:= set of all nodes
 //T is the set of vertices whose shortest
 // distance from a has not been found
 while z T **do**
   chose v T with minimum $L(v)$
   $T := T - \{v\}$
   for each x T adjacent to v **do**
    $L(x) := min\ \{L(x), L(v) + w(v, x)\}$
   end
 end dijkstra



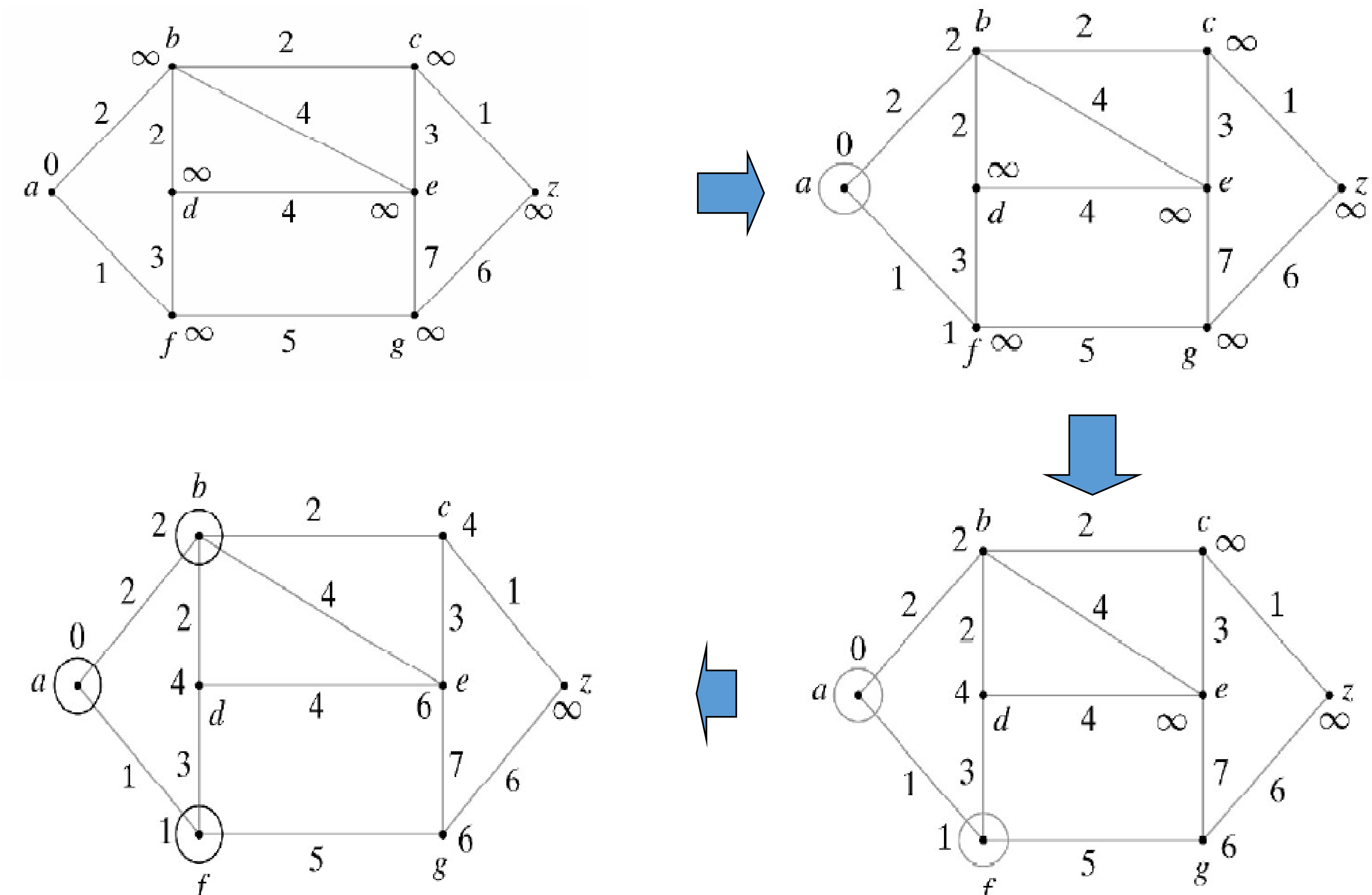
Dijkstra's algorithm (狄克斯特拉算法)

Example 8.4.2

```
procedure dijkstra(w, a, z, L)
    L(a) := 0
    for each node x≠a do
    L(x) := ∞
  T:= set of all nodes
  //T is the set of vertices whose shortest
  // distance from a has not been found
 while z    T do
       chose v    T with minimum L(v)
       T := T − {v}
       for each x    T adjacent to v do
          L(x) := min {L(x), L(v) + w(v, x)}
      end
  end dijkstra
```

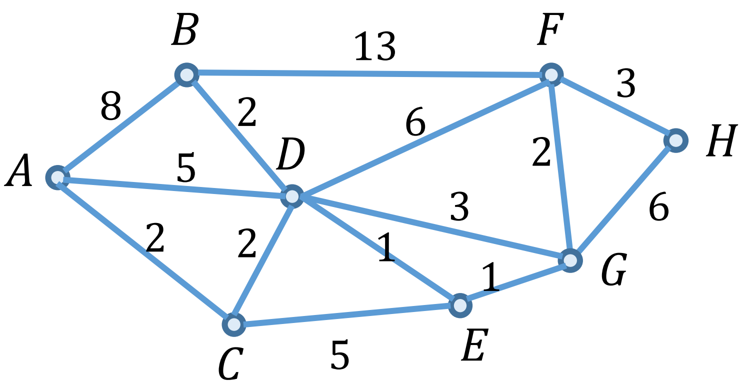


a b c d e f g z



Dijkstra's algorithm (狄克斯特拉算法)

Exercise Which vertex has the largest shortest path starting from vertex A?

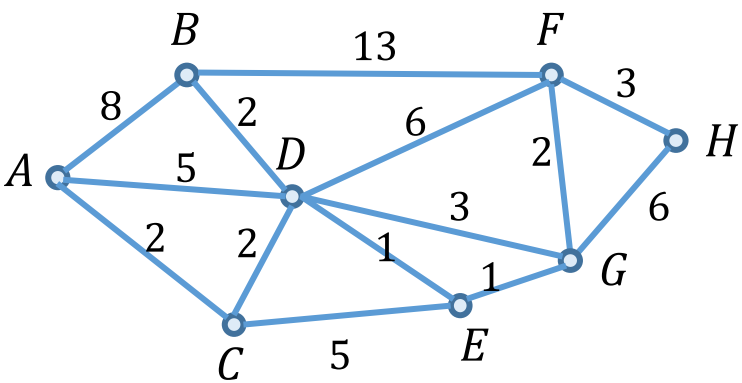



```
procedure dijkstra(w, a, z, L)
    L(a) := 0
    for each node x≠a do
    L(x) := ∞
  T := set of all nodes
  //T is the set of vertices whose shortest
  // distance from a has not been found
  while z ∈ T do
     choose v ∈ T with minimum L(v)
     T := T − {v}
     for each x ∈ T adjacent to v do
        L(x) := min {L(x), L(v) + w(v, x)}
     end
  end dijkstra
```

	A	B	C	D	E	F	G	H
1								
2								
3								
4								
5								
6								
7								
8								

Dijkstra's algorithm (狄克斯特拉算法)

Exercise Which vertex has the largest shortest path starting from vertex A?

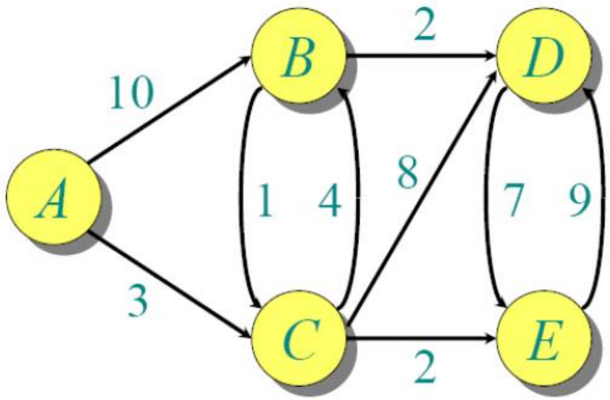



```
procedure dijkstra(w, a, z, L)
    L(a):= 0
    for each node x≠a do
    L(x):=  ∞
  T:= set of all nodes
  //T is the set of vertices whose shortest
  // distance from a has not been found
  while z   T do
     chose v   with minimum L(v)
     T:= T − {v}
     for each x   T adjacent to v do
        L(x):= min {L(x), L(v) + w(v,x)}
     end
  end dijkstra
```

	A	B	C	D	E	F	G	H
1	**0**	∞	∞	∞	∞	∞	∞	∞
2		8_A	$\mathbf{2_A}$	5_A	∞	∞	∞	∞
3		8_A		$\mathbf{4_C}$	7_c	∞	∞	∞
4		6_D			$\mathbf{5_D}$	10_D	7_D	∞
5		$\mathbf{6_D}$				10_D	6_E	∞
6						10_D	$\mathbf{6_E}$	∞
7						$\mathbf{8_G}$		12_G
8								$\mathbf{11_F}$

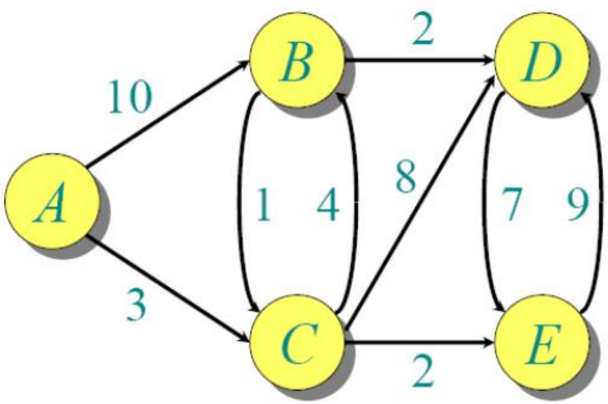
8.4 A Shortest-Path Algorithm 最短路径算法

Exercise Use Dijkstra's Shorest-Path Algorithm to find the length of a shortest path from A to D.



8.4 A Shortest-Path Algorithm 最短路径算法

Exercise Use Dijkstra’s Shorest-Path Algorithm to find the length of a shortest path from A to D.



A	B	C	D	E
$\boxed{0}$	∞	∞	∞	∞
	10	$\boxed{3}$	∞	∞
	7		11	$\boxed{5}$
	$\boxed{7}$		11	
			$\boxed{9}$	

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra’s Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra’s shortest-path algorithm correctly finds the length of a shortest path form a to z.

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

In addition to circle a vertex, we will also label it with the name of the vertex from which it was labeled.

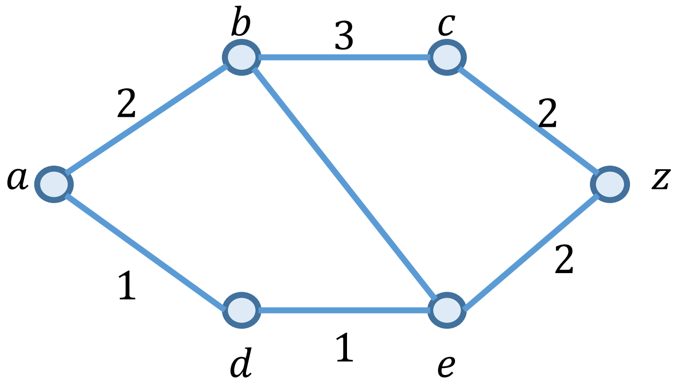
8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

In addition to circling a vertex, we will also label it with the name of the vertex from which it was labeled.

Example 8.4.4



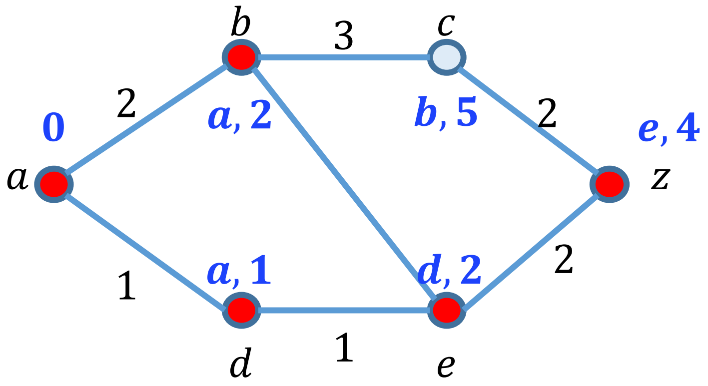
8.4 A Shortest-Path Algorithm 最短路径算法

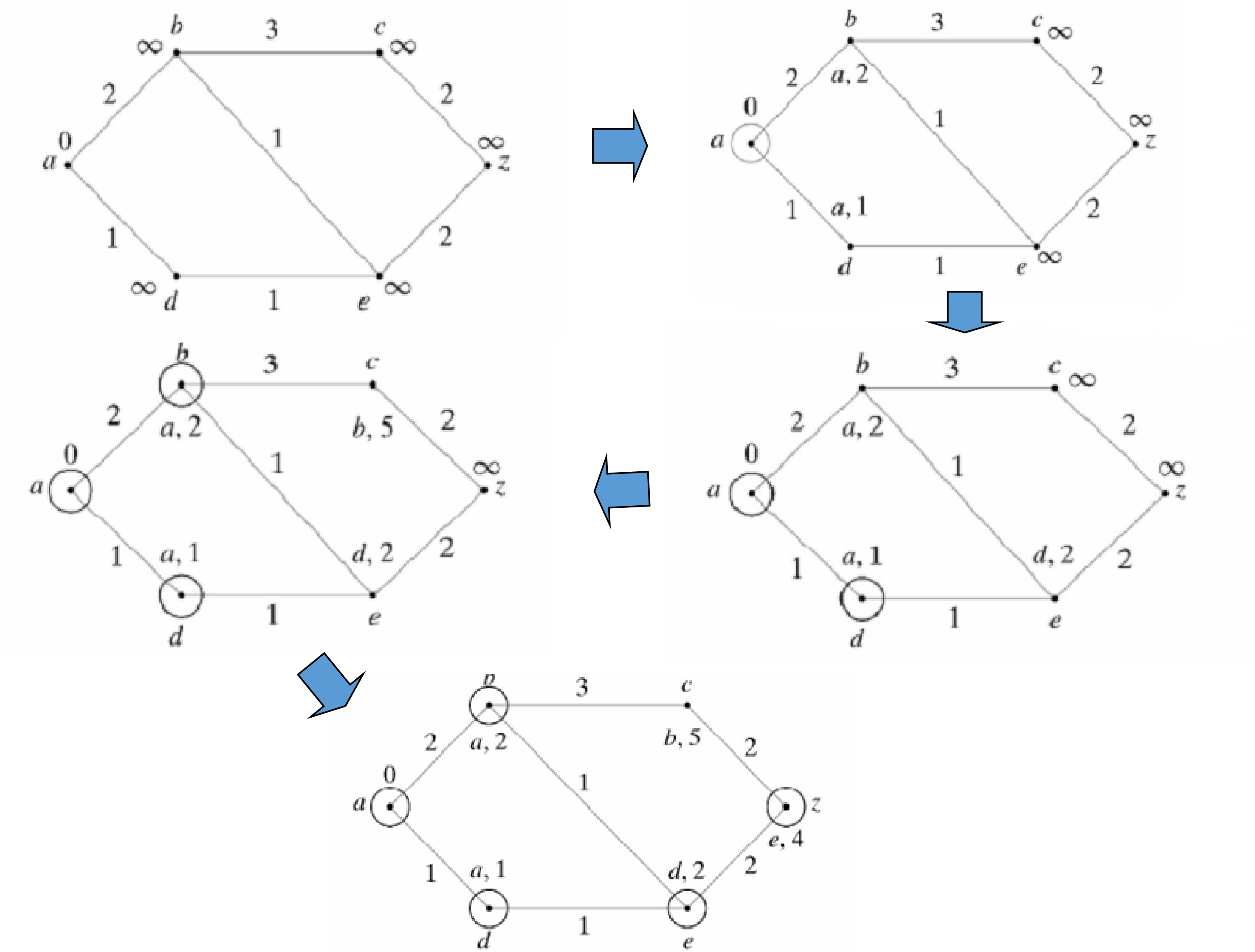
Dijkstra's Shortest-Path Algorithm

Theorem 8.4.3 Dijkstra's shortest-path algorithm correctly finds the length of a shortest path form a to z.

In addition to circling a vertex, we will also label it with the name of the vertex from which it was labeled.

Example 8.4.4




a b c d e z
0 ∞ ∞ ∞ ∞ ∞
2 3 2 1 1 1 2 2
a b c d e z
0 a, 2 ∞ a, 1 ∞ ∞
2 3 2 1 1 1 2 2
a b c d e z
0 a, 2 ∞ a, 1 d, 2 ∞
a b c d e z
0 a, 2 b, 5 a, 1 d, 2 ∞
a b c d e z
0 a, 2 b, 5 a, 1 d, 2 e, 4

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra’s Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra’s shorest-path algorithm correctly finds the length of a shortest path form a to z.

Let P be a shortest path from a to z.

We want to prove that

(i) $L(z) \geq$ length of P

(ii) $L(z) \leq$ length of P

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shorest-Path Algorithm

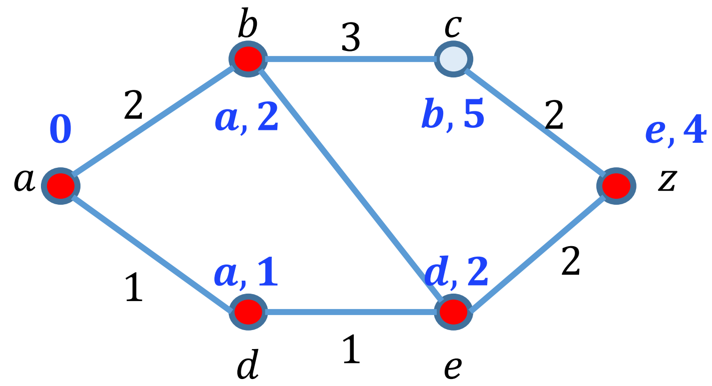
Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

Let P be a shortest path from a to z.

We want to prove that

(i) $L(z) \geq$ length of P

(ii) $L(z) \leq$ length of P



8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

Let P be a shortest path from a to z.

We want to prove that

(i) $L(z) \geq$ length of P

(ii) $L(z) \leq$ length of P

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra’s Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra’s shorest-path algorithm correctly finds the length of a shortest path form a to z.

Let P be a shortest path from a to z. (ii) $L(z) \leq$ length of P

Proof We use mathematical induction on i to prove that the ith time we choose a vertex v with minimum $L(v)$, $L(v)$ is the length of a shortest path from a to v.

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

Let P be a shortest path from a to z. (ii) $L(z) \leq \text{length of } P$

Proof We use mathematical induction on i to prove that the ith time we choose a vertex v with minimum $L(v)$, $L(v)$ is the length of a shortest path from a to v.

Basis Step ($i = 1$)

Inductive Step ($k < i$)
If there is a path from a to w whose length is less than $L(v)$, then w is not in T.

8.4 A Shortest-Path Algorithm 最短路径算法

Modify Dijkstra's shortest-path algorithm so that it accepts a weighted graph that is not necessarily connected. At termination, what is $L(z)$ if there is no path from a to z?

8.4 A Shortest-Path Algorithm 最短路径算法

True or false? When a connected, weighted graph and vertices a and z are input to the following algorithm, it returns the length of a shortest path from a to z.

Algorithm 8.4.6

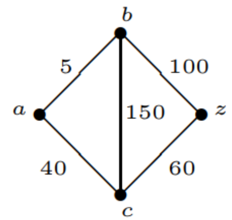
$algor(w, a, z)$ {
    $length = 0$
    $v = a$
    T = set of all vertices
    while ($v \neg = z$) {
        $T = T - \{v\}$
        choose $x \in T$ with minimum $w(v, x)$
        $length = length + w(v, x)$
        $v = x$
    }
    return $length$
}

8.4 A Shortest-Path Algorithm 最短路径算法

True or false? When a connected, weighted graph and vertices a and z are input to the following algorithm, it returns the length of a shortest path from a to z.

Algorithm 8.4.6

```
algor(w, a, z) {
    length = 0
    v = a
    T = set of all vertices
    while (v¬ = z) {
        T = T − {v}
        choose x ∈ T with minimum w(v, x)
        length = length + w(v, x)
        v = x
    }
    return length
}
```



8.4 A Shortest-Path Algorithm 最短路径算法

True or false? Dijkstra’s shorest-path algorithm finds the length of a shortest path in a connected, weighted graph even if some weights are negative. If true, prove it; otherwise, provide a counterexample.

8.4 A Shortest-Path Algorithm 最短路径算法

True or false? Dijkstra's shorest-path algorithm finds the length of a shortest path in a connected, weighted graph even if some weights are negative. If true, prove it; otherwise, provide a counterexample.