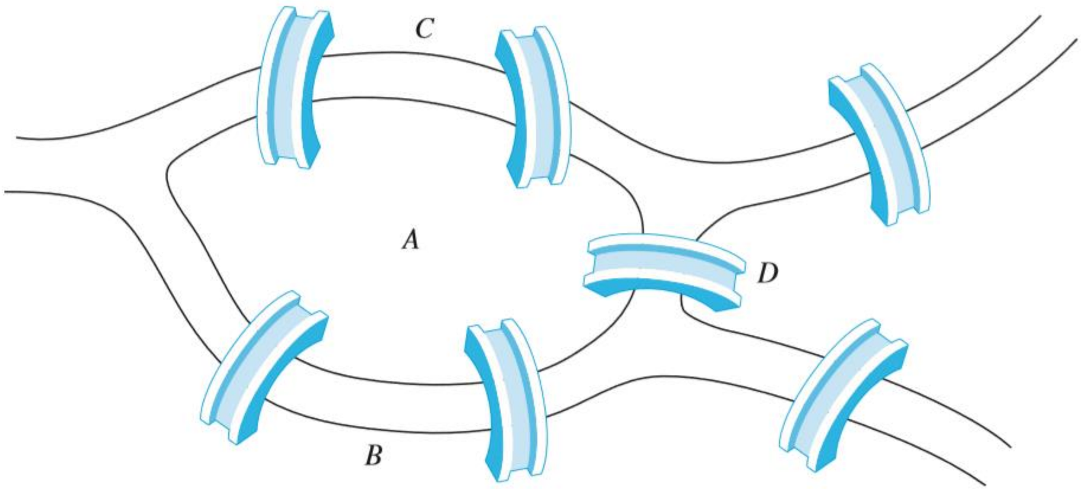
Chapter 9-Graphs

"One graph is worth a thousand logs."

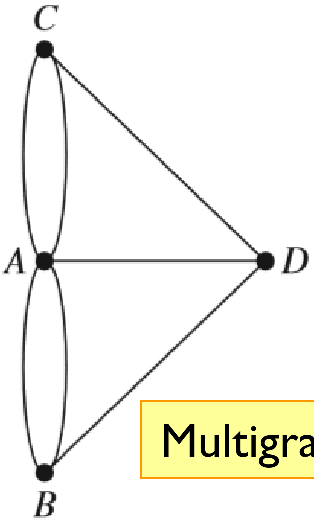
Michal Aharon, Gilad Barash, Ira Cohen and Eli Mordechai.

Introduction



The Seven Bridges of Königsberg.

LEONHARD **EULER**
(1707–1783)



Multigraph Model

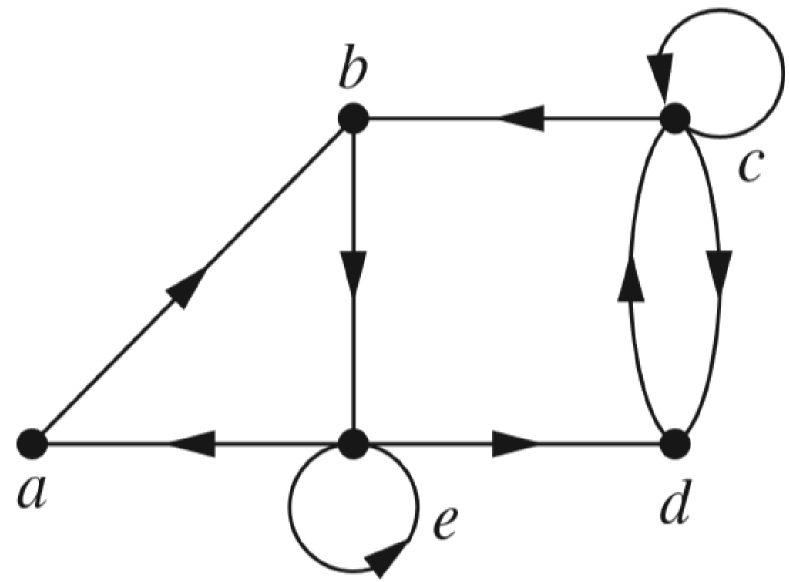
Introduction

Web Graph
Google PageRank

```
> G<-make_graph(e,v)
> page.rank(G)
$vector
[1] 0.1064619 0.1922395 0.2532244 0.1782088 0.2698655

$value
[1] 1

$options
NULL
```



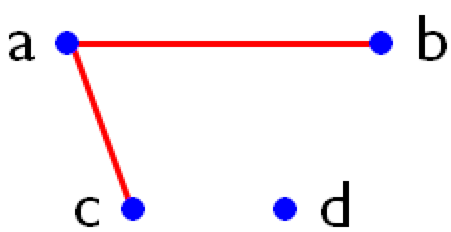
A *network* with ***5 websites*** and ***links***

Objectives

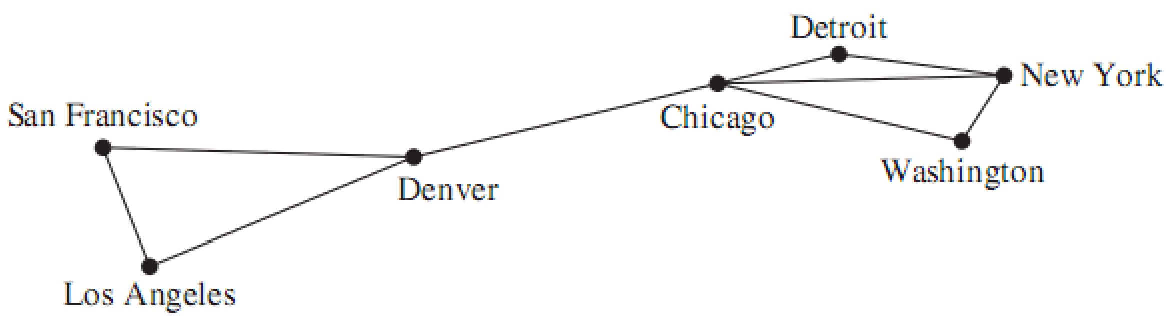
- 9.1- Graphs and Graph Models
- 9.2- Graph Terminology and Special Types of Graphs
- 9.3- Representing Graphs and Graph Isomorphism
- 9.4- Connectivity
- 9.5- Euler and Hamilton Paths
- 9.6- Shortest Path Problems

Graphs

- Graphs = (vertices, edges) ➔ G = (V, E)

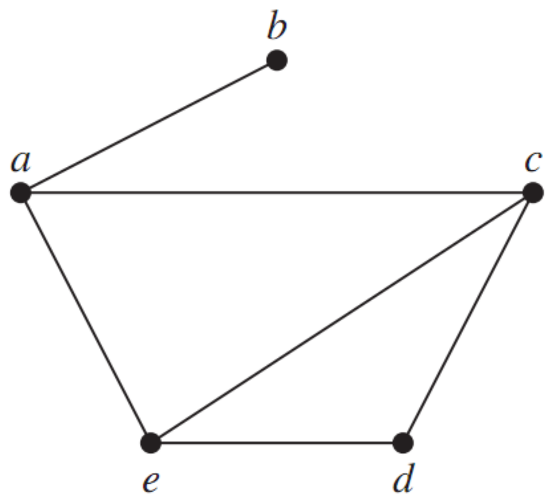


V = vertices = { a, b, c, d }
E = edges = { {a, b}, {a, c} }

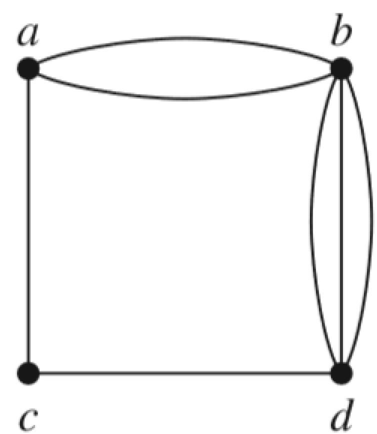


Computer network

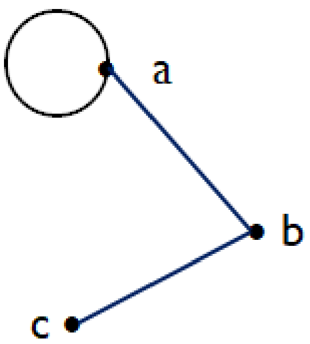
Simple graphs



Simple graph

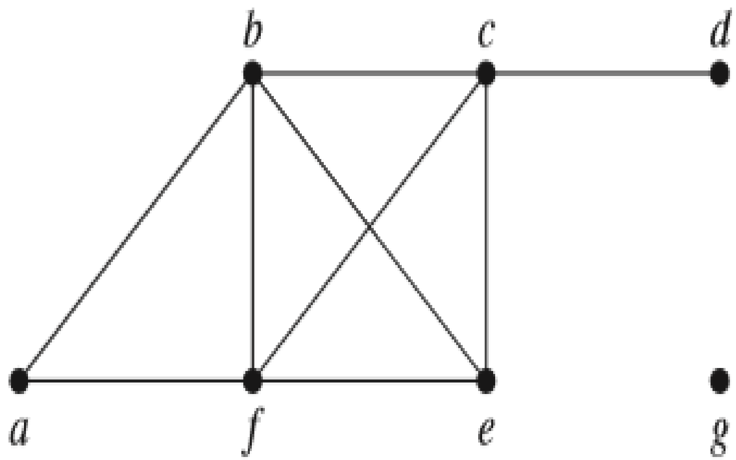


Non-simple graph with *multi-edges*

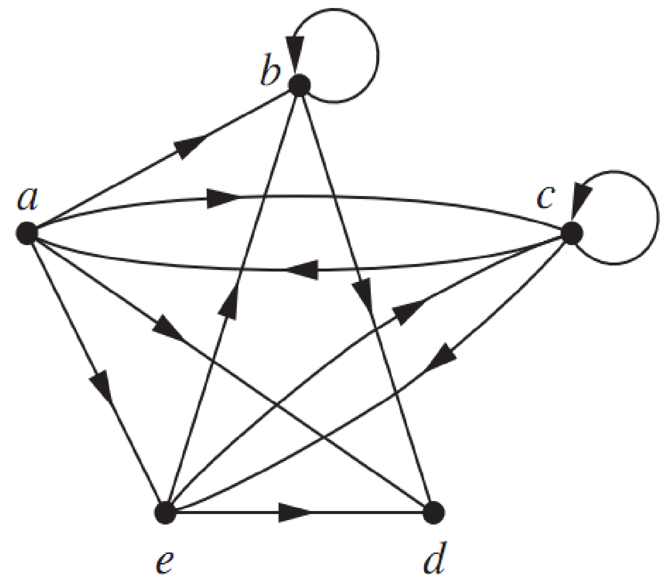


Non-simple graph with *loops*

Types of graphs



An **undirected** graph



A **directed** graph
(*digraph*)

Graphs and Graph Models….

TABLE 1 Graph Terminology.

Type	*Edges*	*Multiple Edges Allowed?*	*Loops Allowed?*
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple directed graph	Directed	No	No
Directed multigraph	Directed	Yes	Yes
Mixed graph	Directed and undirected	Yes	Yes

Graphs and Graph Models....

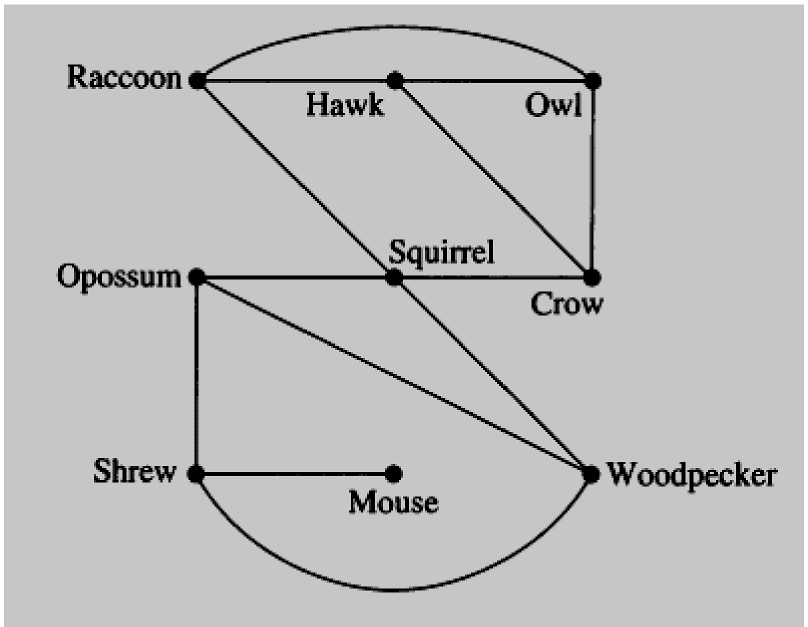


FIGURE 6 A Niche Overlap Graph.

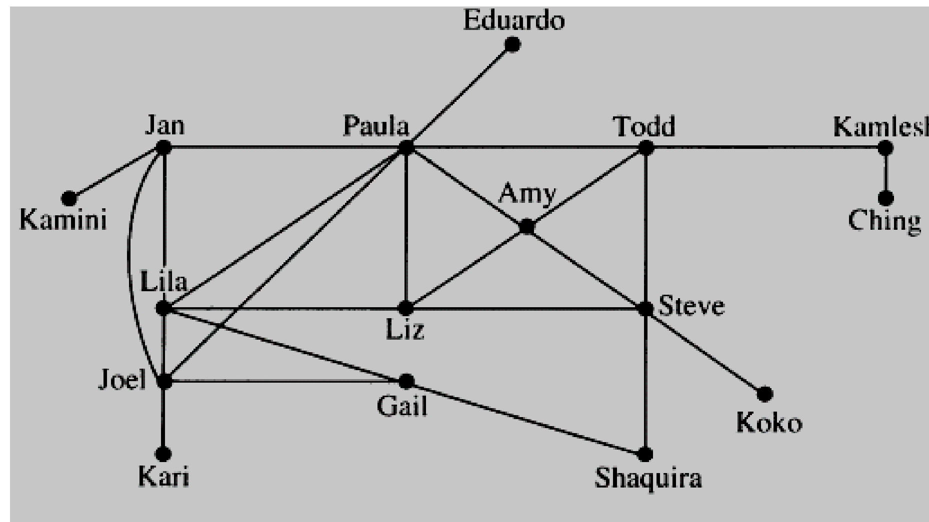
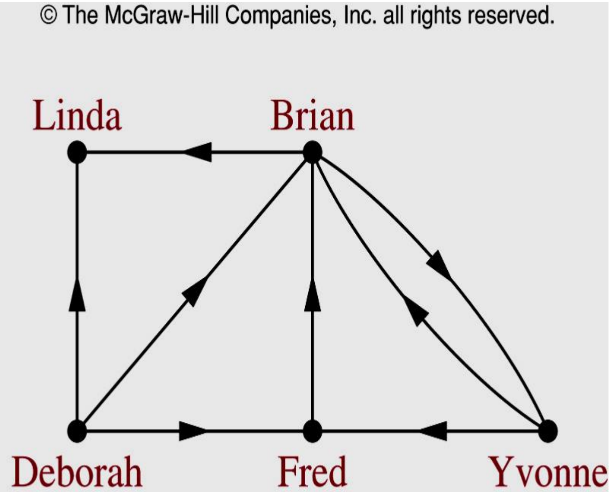


FIGURE 7 An Acquaintanceship Graph.

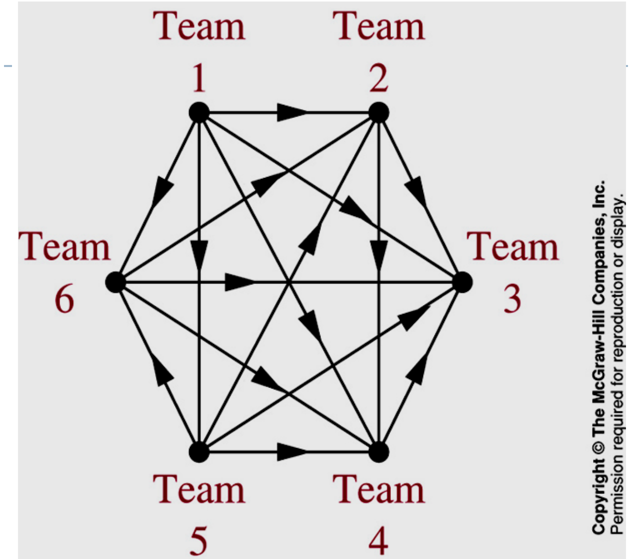
Niche Overlap Graph in Ecology
(sinh thái học) – Đồ thị lấn tổ

Acquaintanceship Graph
Đồ thị cho mô hình quan hệ giữa người

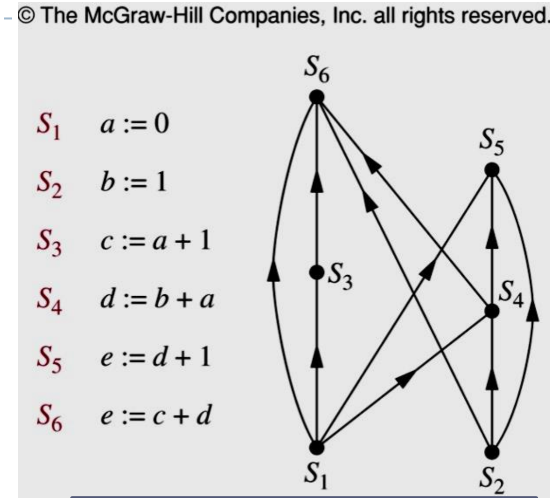
Graphs and Graph Models….



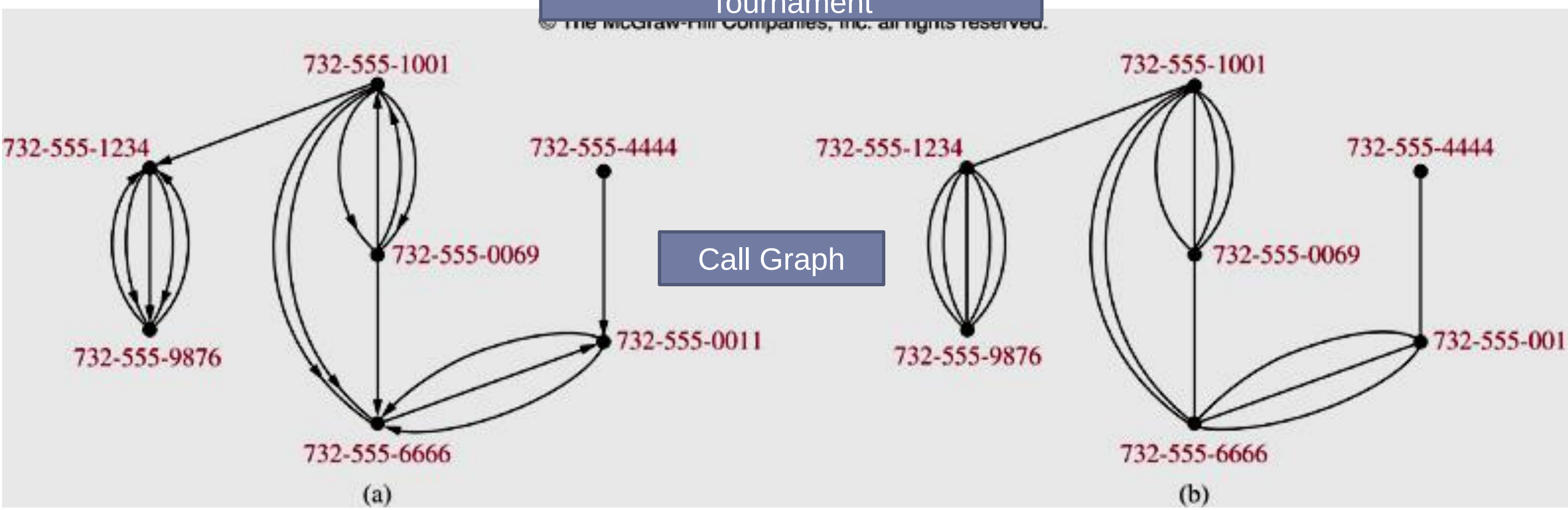
An Influence Graph



A Graph Model of a Round-Robin Tournament

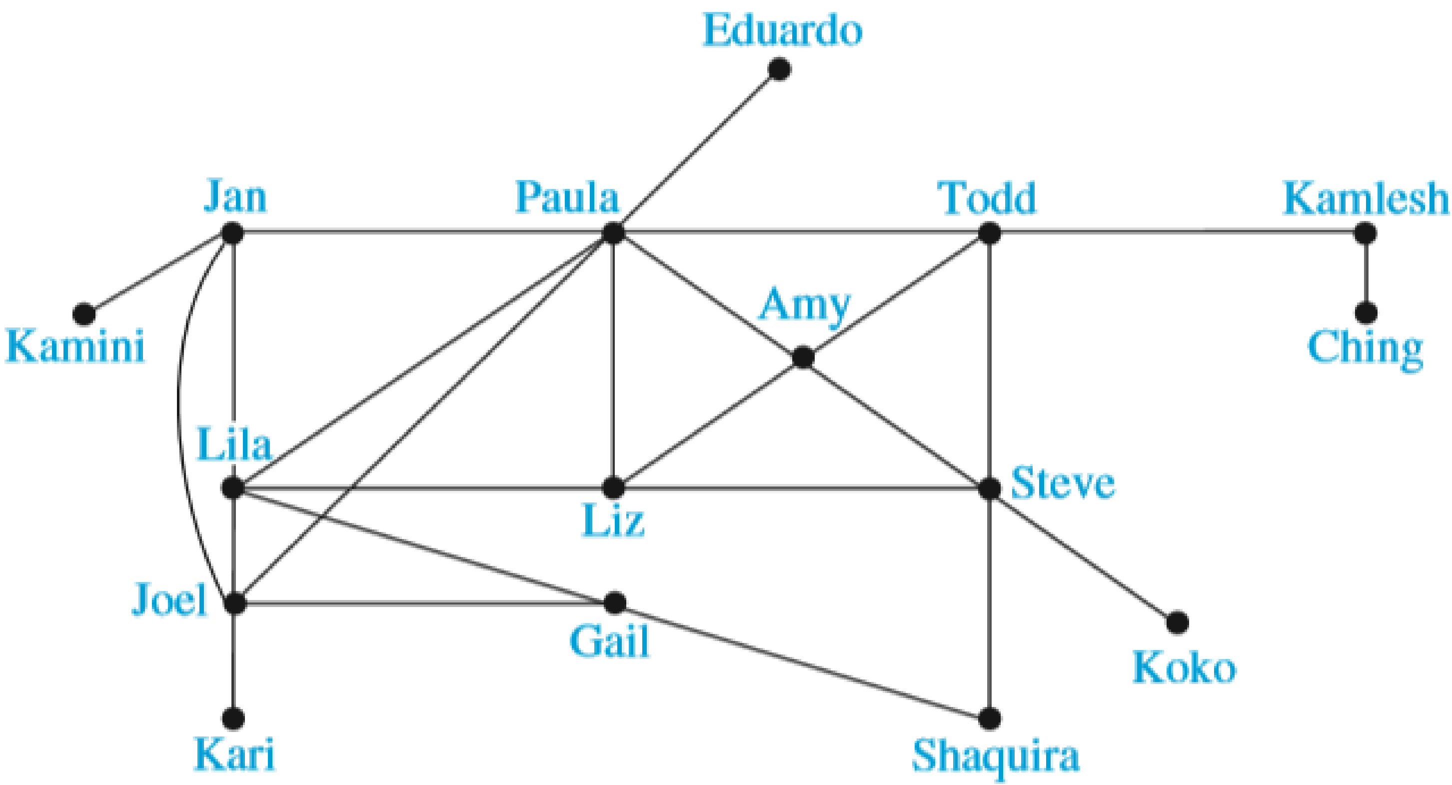


An Precedence Graph

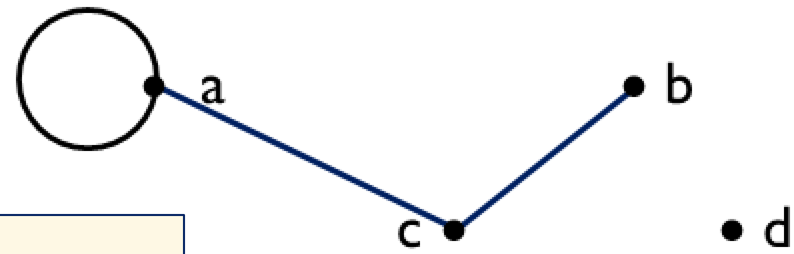


Call Graph

Graph models - Social networks



Basic *Terminology*



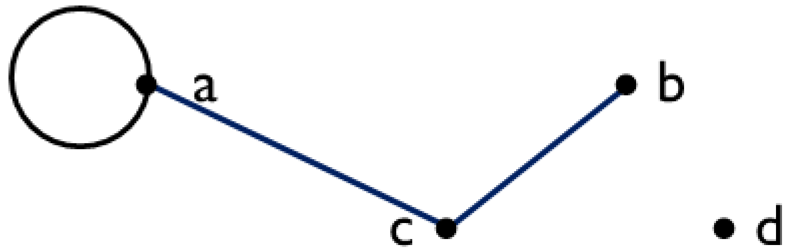
If an *edge* **{u, v}** exists, then *u and v* are called ***adjacent***

vertex	*adjacency* list
a	a, c
b	c
c	a, b
d	

The *edge {u, v}* is called ***incident*** with u and v

edge	*incident* vertices
{a, a}	a
{a, c}	a, c
{c, b}	c, b

Basic Terminology



The **degree** of a vertex v:

= the number of edges *incident* with v, except that *a loop* at a vertex *contributes twice*.

Notation: deg(v)

vertex	degree
a	3
b	1
c	2
d	0

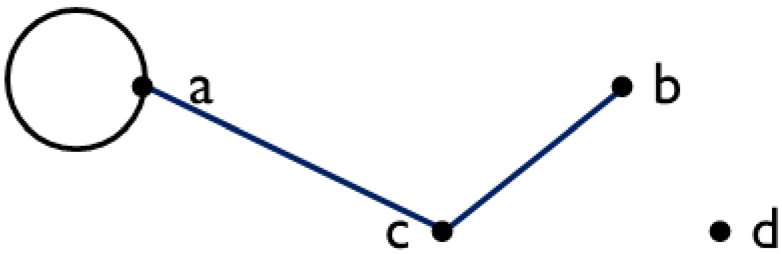
b is called ***pendant***

d is called ***isolated***

$\sum_{\text{degree}} = 6$

Remark: $\sum_{\text{degree}} = 2|\text{ Edges }|$

THE HANDSHAKING THEOREM (for undirected graphs)



3 edges: {a, a}, {c, b}, {a, c}

deg(a) = 3
deg(b) = 1
deg(c) = 2
deg(d) = 0

$\Sigma_{deg} = 6$

THE HANDSHAKING THEOREM:

(one edge = two degrees)

$$\Sigma_{v \in V} \deg(v) = 2|E|$$

always EVEN

"the sum of the degrees is twice the number of edges".

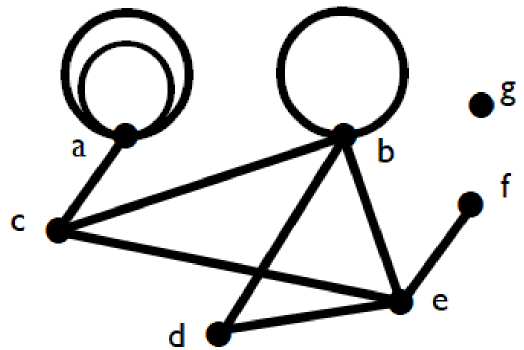
THE HANDSHAKING THEOREM - *Examples*

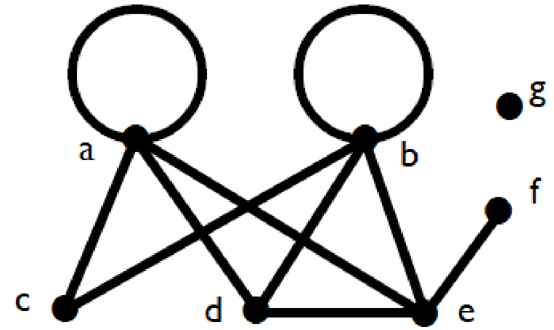
Ex2. *How many edges* does a graph have if its *degree sequence* is 5, 5, 4, 3, 2, 1, 0 ?

Draw a such graph.

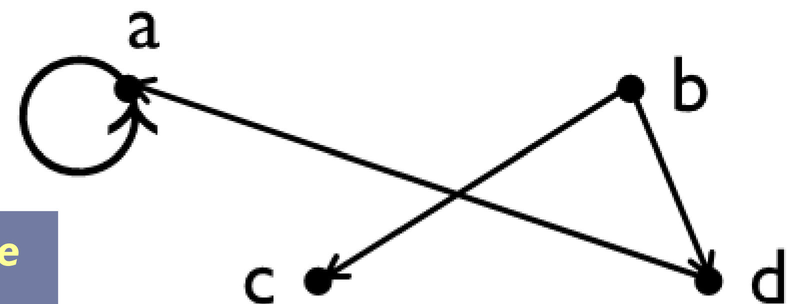
$$\sum_v \deg(v) = 5 + 5 + 4 + 3 + 2 + 1 + 0$$

$$= 20 = 2.|E| \quad \rightarrow |E| = 10.$$





Directed graphs -Basic Terminology



Vertex	*In-degree* deg^-	*Out-degree* deg^+
a	2	1
b	0	2
c	1	0
d	1	1

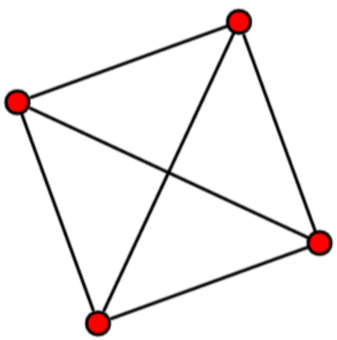
initial vertex terminal vertex

(b, c) ~~{b, c}~~ ~~(c, b)~~

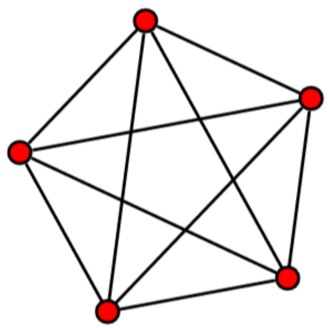
$\boxed{\sum_{deg-} = \sum_{deg+}}$ = 4 *directed edges*

Special simple graphs

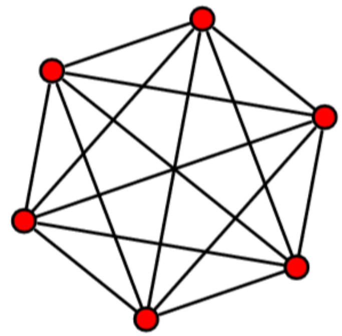
Complete graphs K_n $(n \geq 1)$



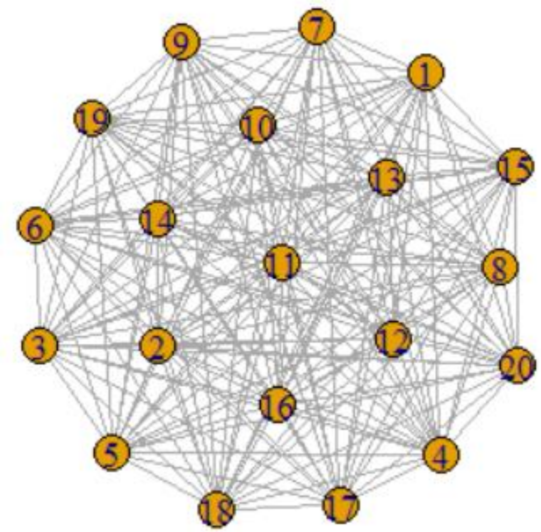
K_4



K_5

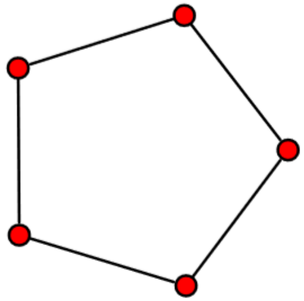


K_6

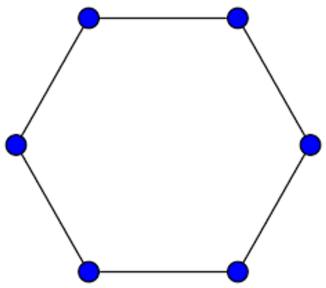


K_{20}

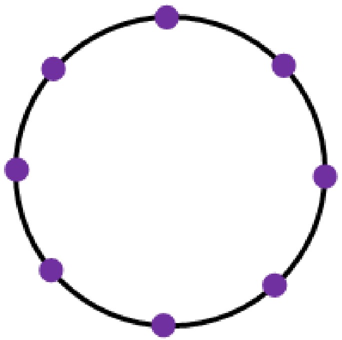
Cycles C_n $(n \geq 3)$



C_5



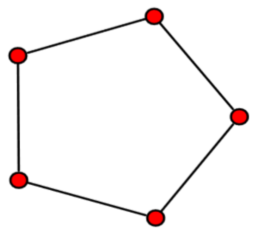
C_6



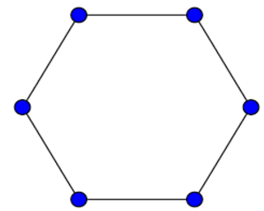
C_8

Wheels W_n

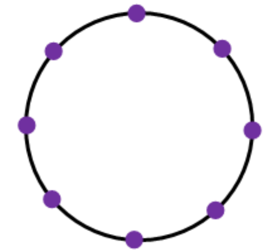
Cycles
C_n



C_5

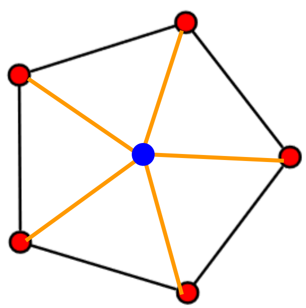


C_6

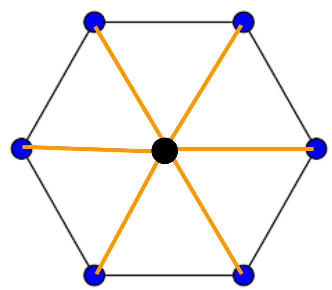


C_8

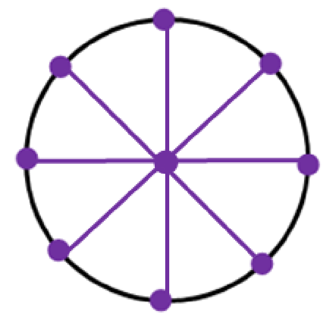
Wheels
W_n $(n \geq 3)$



W_5



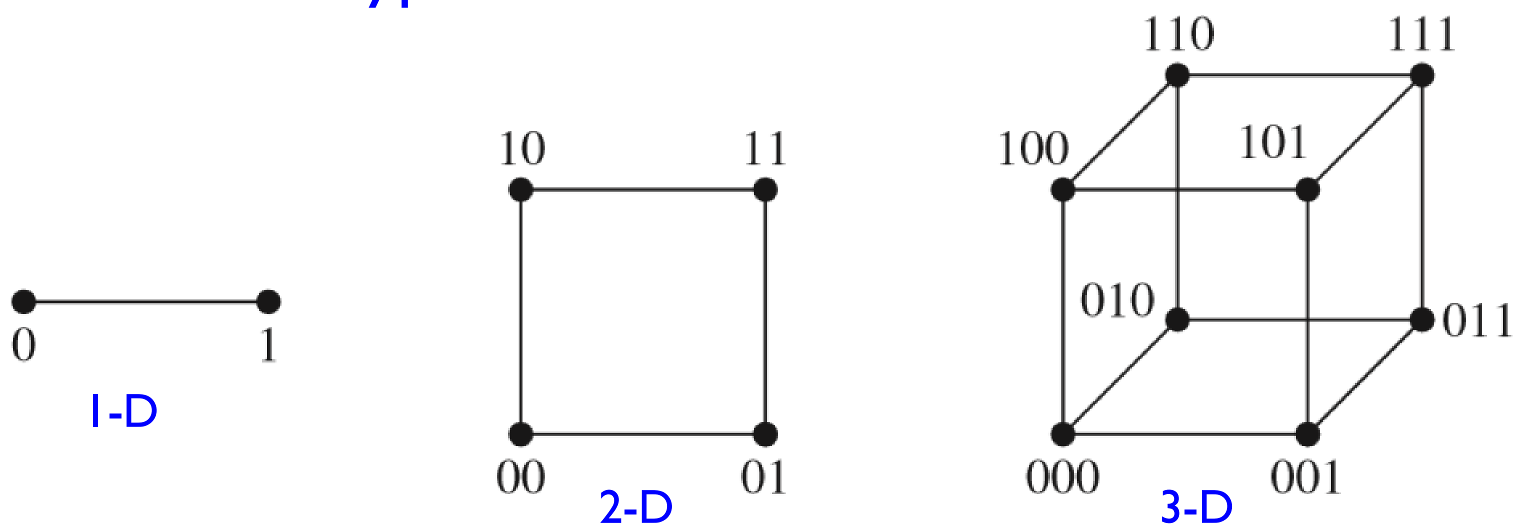
W_6



W_8

n-cubes Q_n

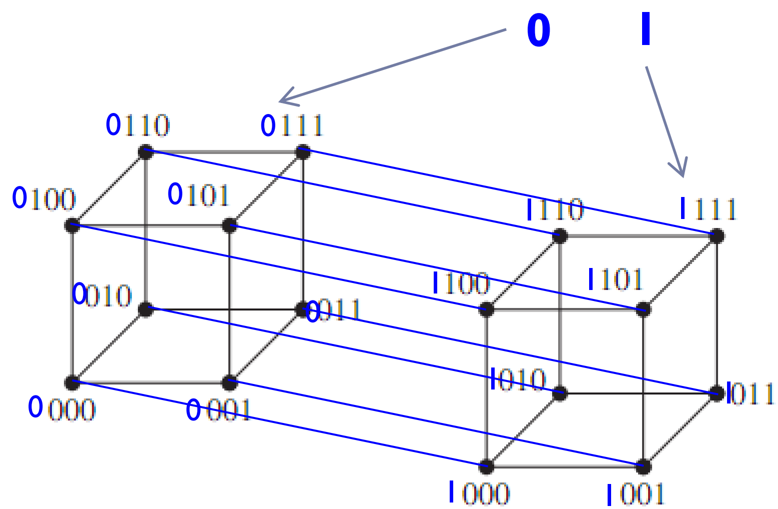
n-dimensional hypercube



Graph	$|V|$ = number of vertices	$|E|$ = number of edges
Q_n	2^n	$n.2^{n-1}$

n-cube Q_n

- Construct Q_4 from two copies of Q_3



Bipartite **graphs**

- A simple graph G = (**V**, E) is called ***bipartite*** if:
  - **V** = $V1 \cup V2$, $V1 \cap V2 = \varnothing$
  - *no edge* connects two vertices in V1
  - *no edge* connects two vertices in V2
- We call the pair (V1,V2) a *bipartition* of V.

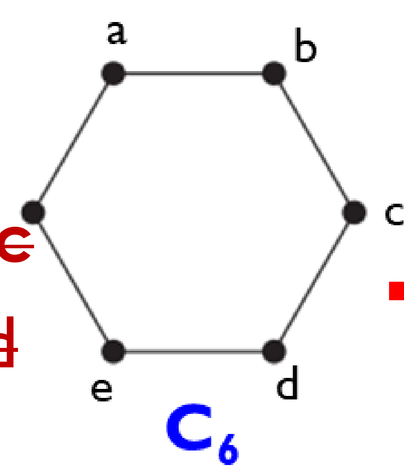
Ex. Study graph $\mathbf{C_6}$ (cycle)

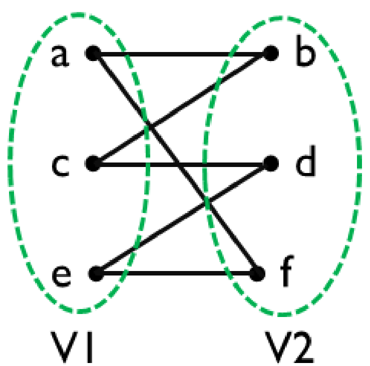
V = { a, b, c, d, e, f }

V1 = {a, e, c} ~~ae~~ ~~ac~~ ~~ec~~

V2 = {b, f, d} ~~bf~~ ~~bd~~ ~~fd~~

➔ C_6 is *bipartite*

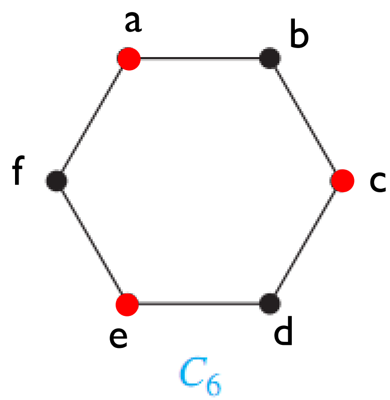




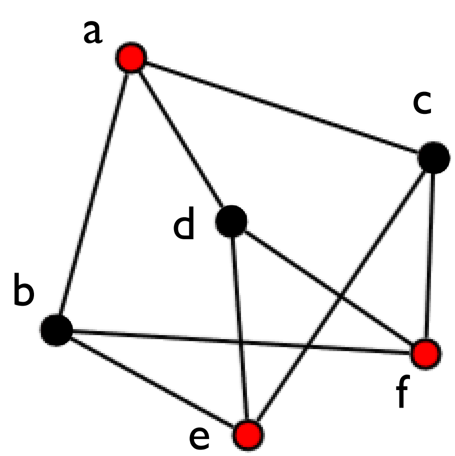
Bipartite graphs

- How to check?
- ***Ex.***

> - Let **color** the vertices using *2 different colors*
> - Two ***adjacent vertices*** must have ***different colors*** (e.g., **red** and **black**)



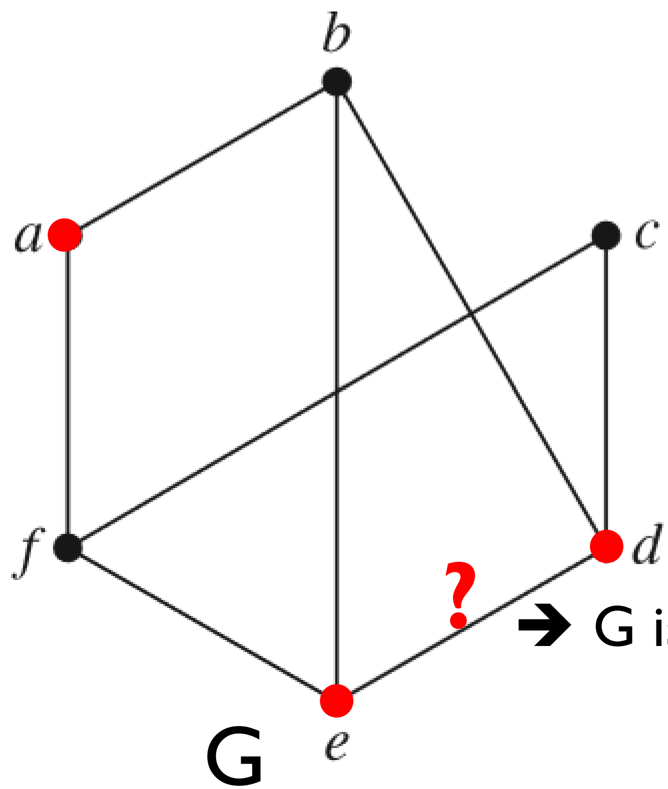
→ C_6 is ***bipartite***
V1 = {a, c, e}
V2 = {b, d, f}



→ *bipartite*
V1 = {a, e, f}
V2 = {b, c, d}

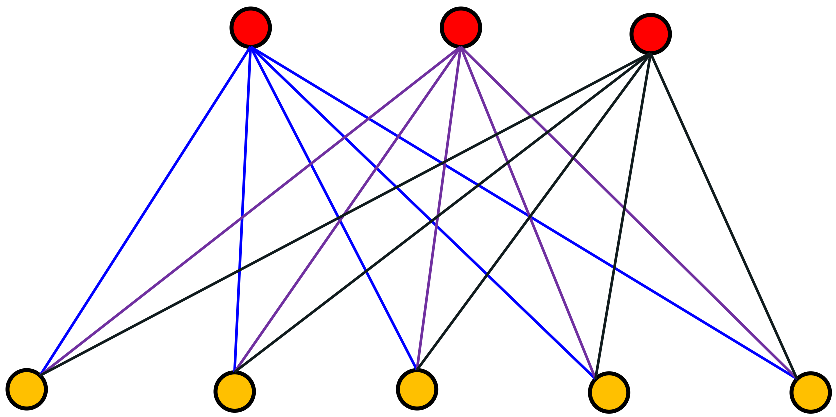
Example

Ex. Is the graph G *bipartite*?



➔ G is *not bipartite*

$K_{m,n}$ - *Complete bipartite* graphs

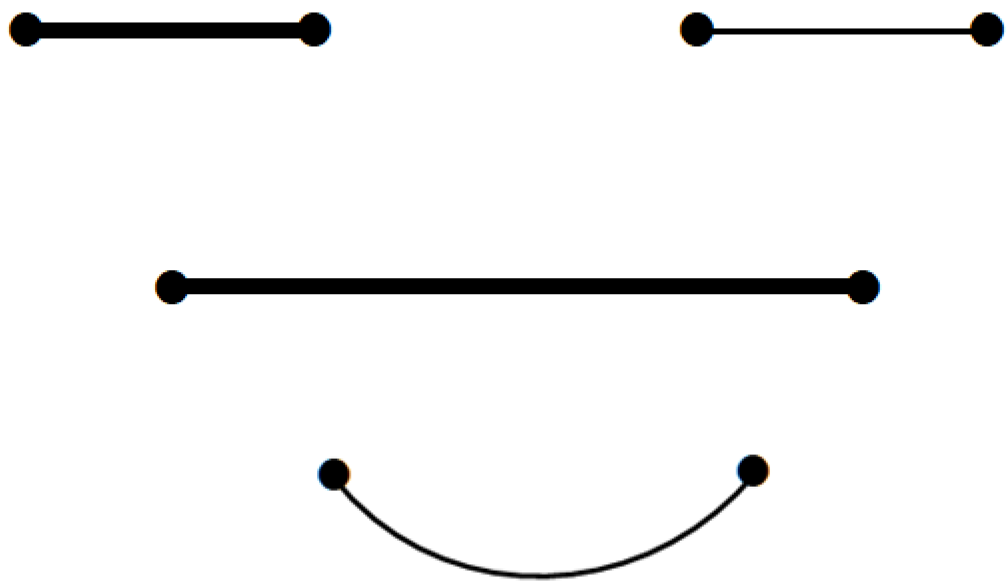


m = 3

n = 5

$K_{3,5}$

Isomorphism



THE SAME

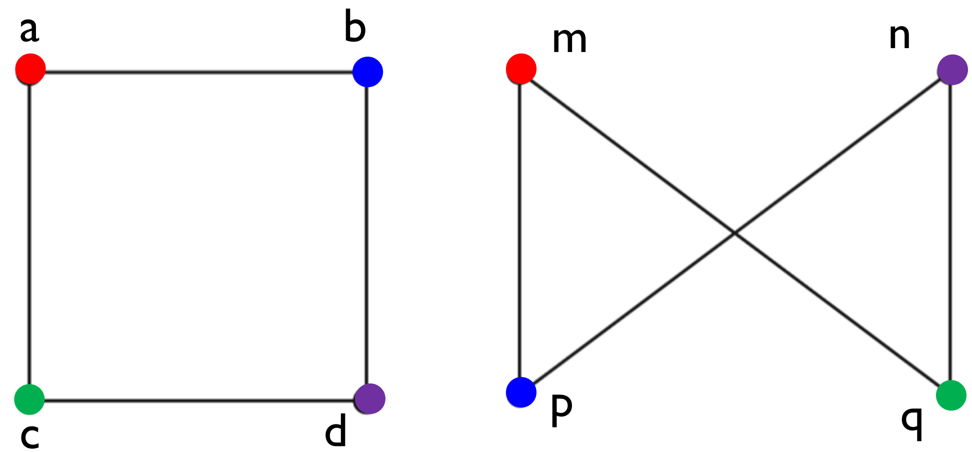
Isomorphism

$G1 = (V1, E1)$, $G2 = (V2, E2)$

G1 and G2 are called ***isomorphic*** if

$\exists$ **function** **f:** $V1 \rightarrow V2$

- One-to-one
- Onto
- a, b are *adjacent* in V1
  $\Leftrightarrow$ f(a), f(b) are *adjacent* in V2

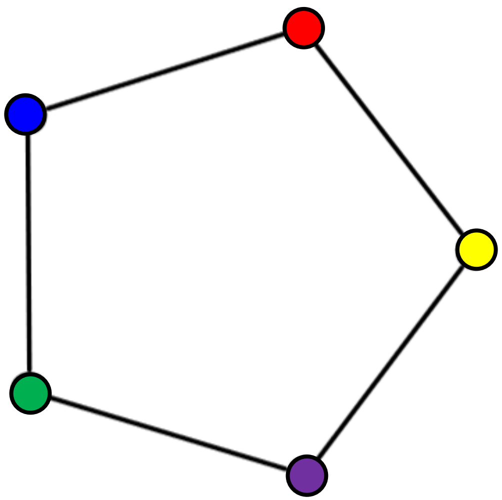


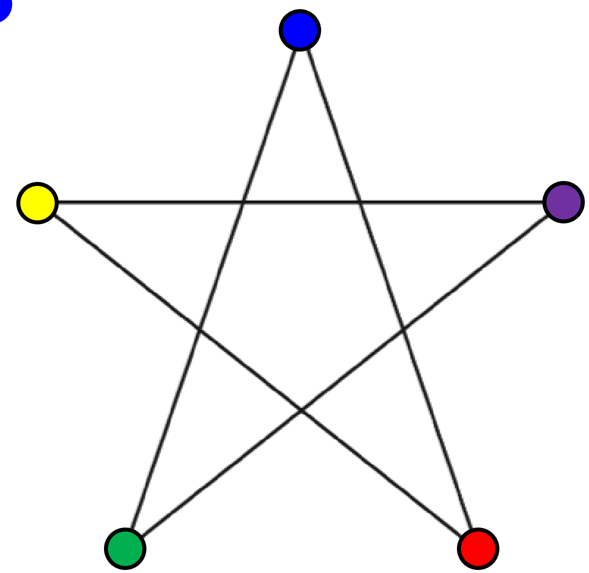
function f = {(a, m); (b, p); (d, n); (c, q)}

➔ Two graphs are called ***isomorphic***

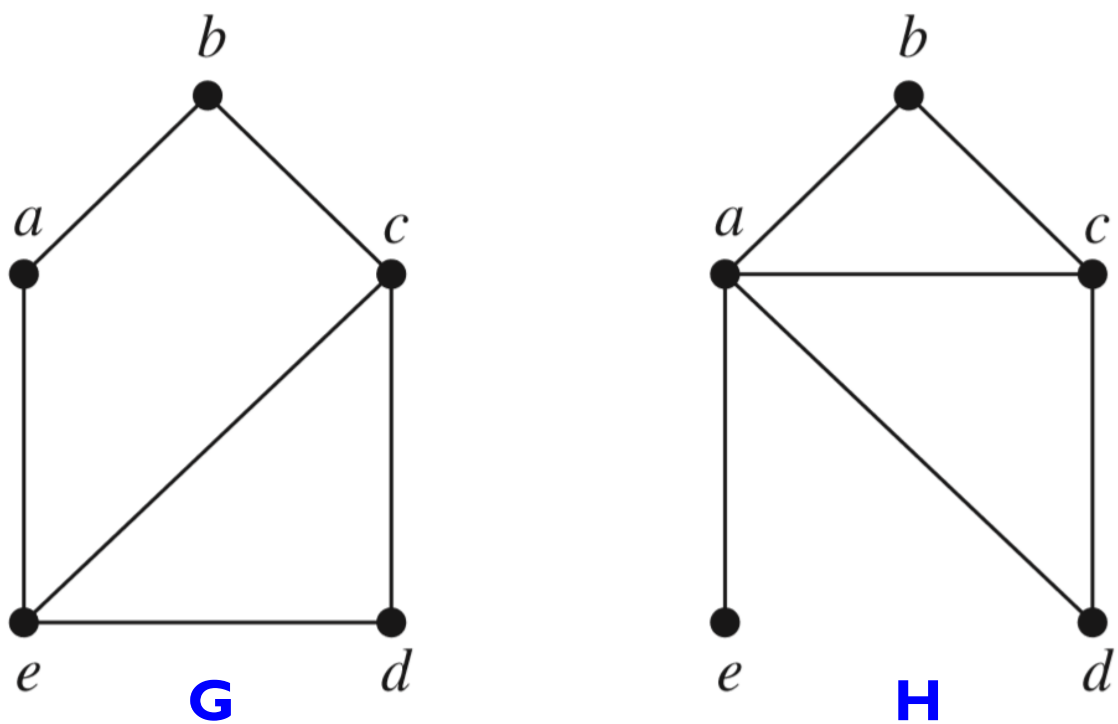
Isomorphic?

YES





Isomorphic?



G and H are **NOT** ***isomorphic***

(in H: $deg(e) = 1$, no vertex in G has degree 1)

Representing graphs

- *Adjacency* matrix

Vertex 1 vertex 2 … vertex n

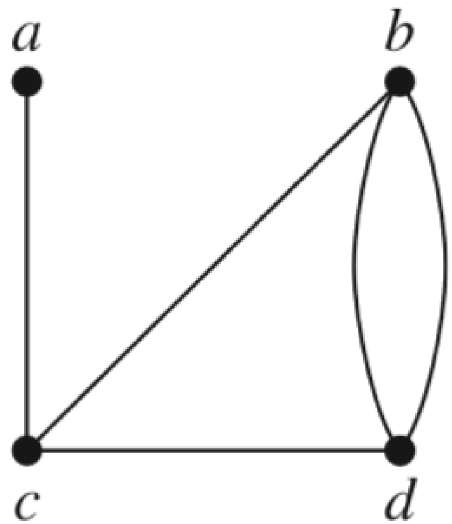
Vertex 1
Vertex 2
…
Vertex n

- *Incidence* matrix

edge 1 *edge* 2 … *edge* m

Vertex 1
Vertex 2
…
Vertex n

Adjacency matrices



	a	b	c	d
a	0	0	1	0
b	0	0	1	2
c	1	1	0	1
d	0	2	1	0

Adjacency matrix.

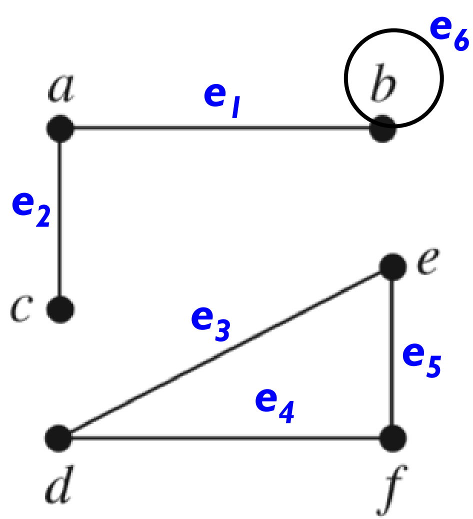
$A = [a_{ij}]$,

where a_{ij} = the *number of edges* that are associated to $\{v_i, v_j\}$

In 2010 the ***Web graph*** was estimated to have at least *55 billion vertices* and one *trillion edges*.

➔ More than ***40 TB*** of computer memory would have been needed to represent its *adjacency matrix*.

Incidence matrices



Vertices = { a, b, c, d, e, f }
Edges = { e_1, e_2, e_3, e_4, e_5, e_6 }

edges

	e_1	e_2	e_3	e_4	e_5	e_6
a	1	1	0	0	0	0
b	1	0	0	0	0	1
c	0	1	0	0	0	0
d	0	0	1	1	0	0
e	0	0	1	0	1	0
f	0	0	0	1	1	0
g	0	0	0	0	0	0

Incidence matrices

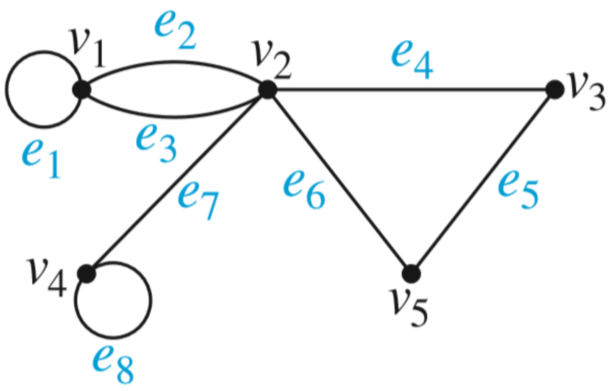
If the edge e_j is incident with the vertex v_i

the (v_i, e_j)-entry = 1

Else

the (v_i, e_j)-entry = 0

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$



A path of length n

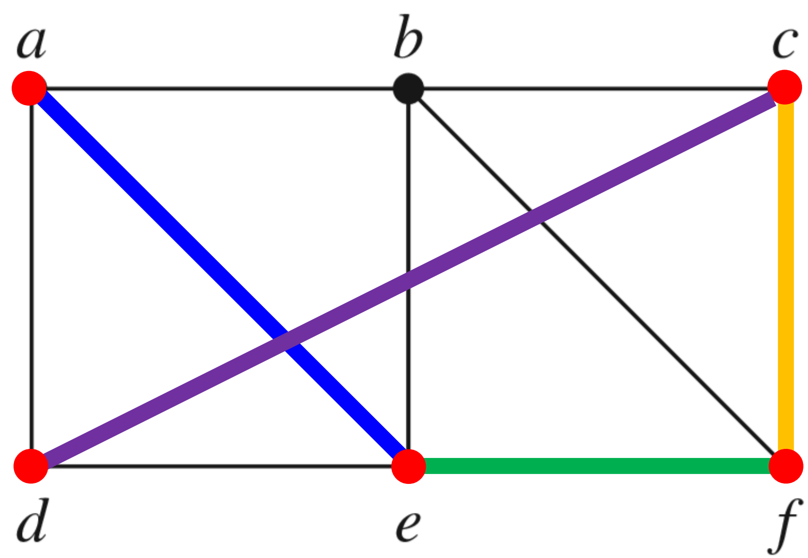
- A *path of length n* from u to v:

A sequence of *n consecutive edges*

- ***Ex.***

a, e, f, c, d

is a *path of* ***length 4***.

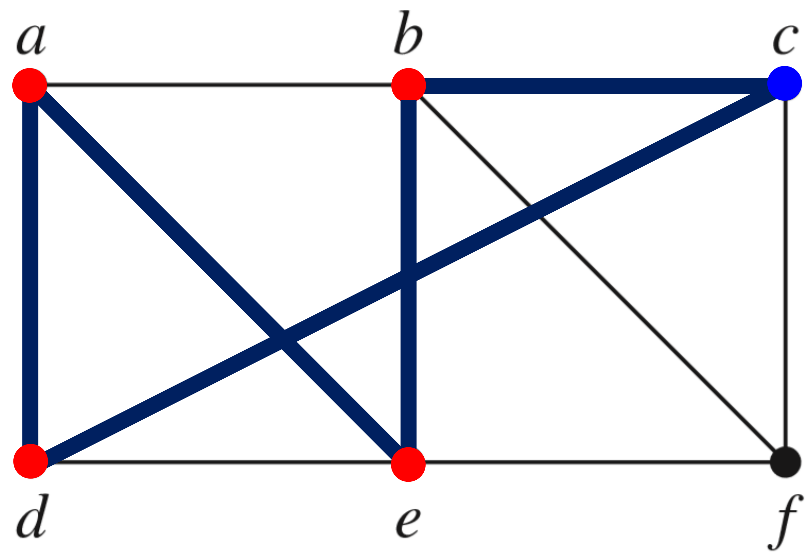


Circuits

- A ***circuit*** is a path of *length greater than zero* that ***starts*** and ***ends*** at the *same vertex*.

- ***Ex.***

  c, b, e, a, d, **c** is ***circuit***.



Simple *paths/ circuits*

- A path/circuit is ***simple*** if it does *not* contain the *same edge more than once.*

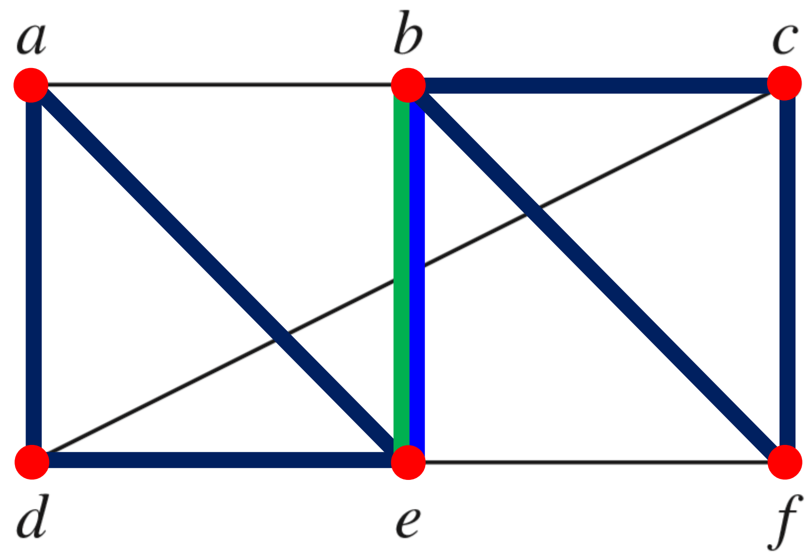
- ***Ex.***

  b, e, a, b, f, c

is a ***simple path***.

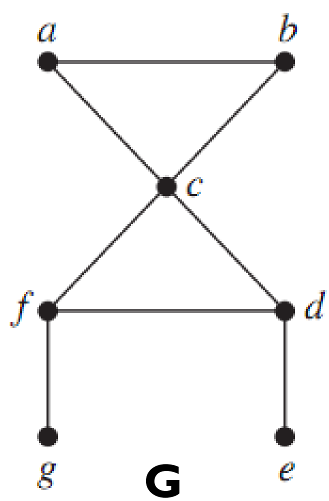
  c, b, e, a, d, e, b, f, c

is NOT a *simple circuit*.

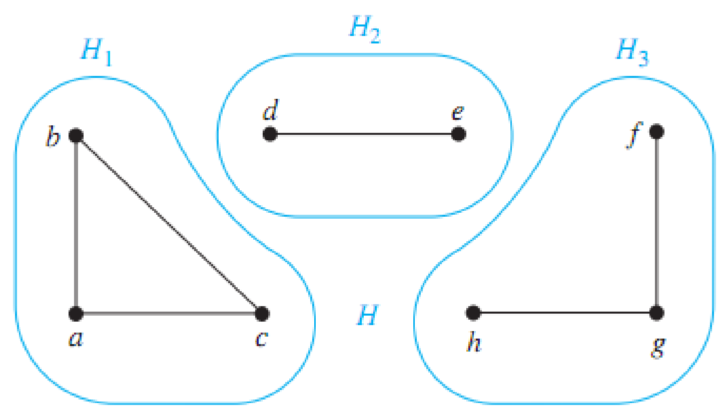


Connectedness **in Undirected Graphs**

- Connected = **there is a path** between every pair of distinct vertices of the graph.
- Not connected = **disconnected**.



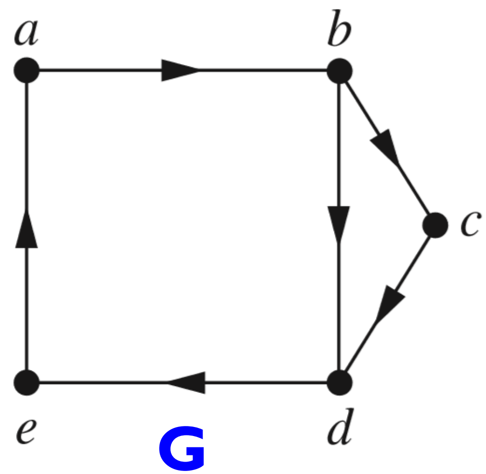
G is **connected**

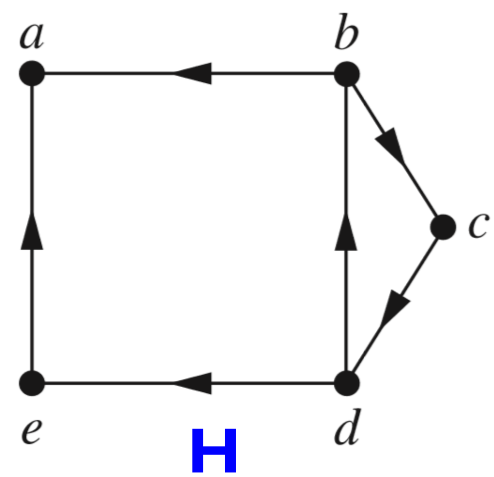


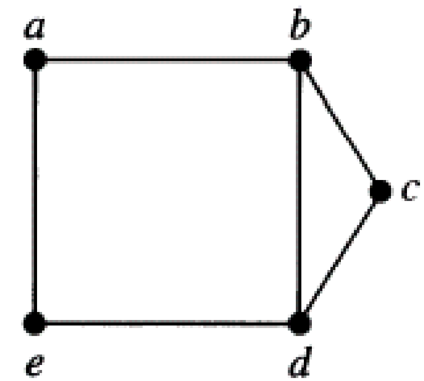
A disconnected with
3 connected **components**

Connectedness in *Directed Graphs*

Strongly connected vs weakly connected







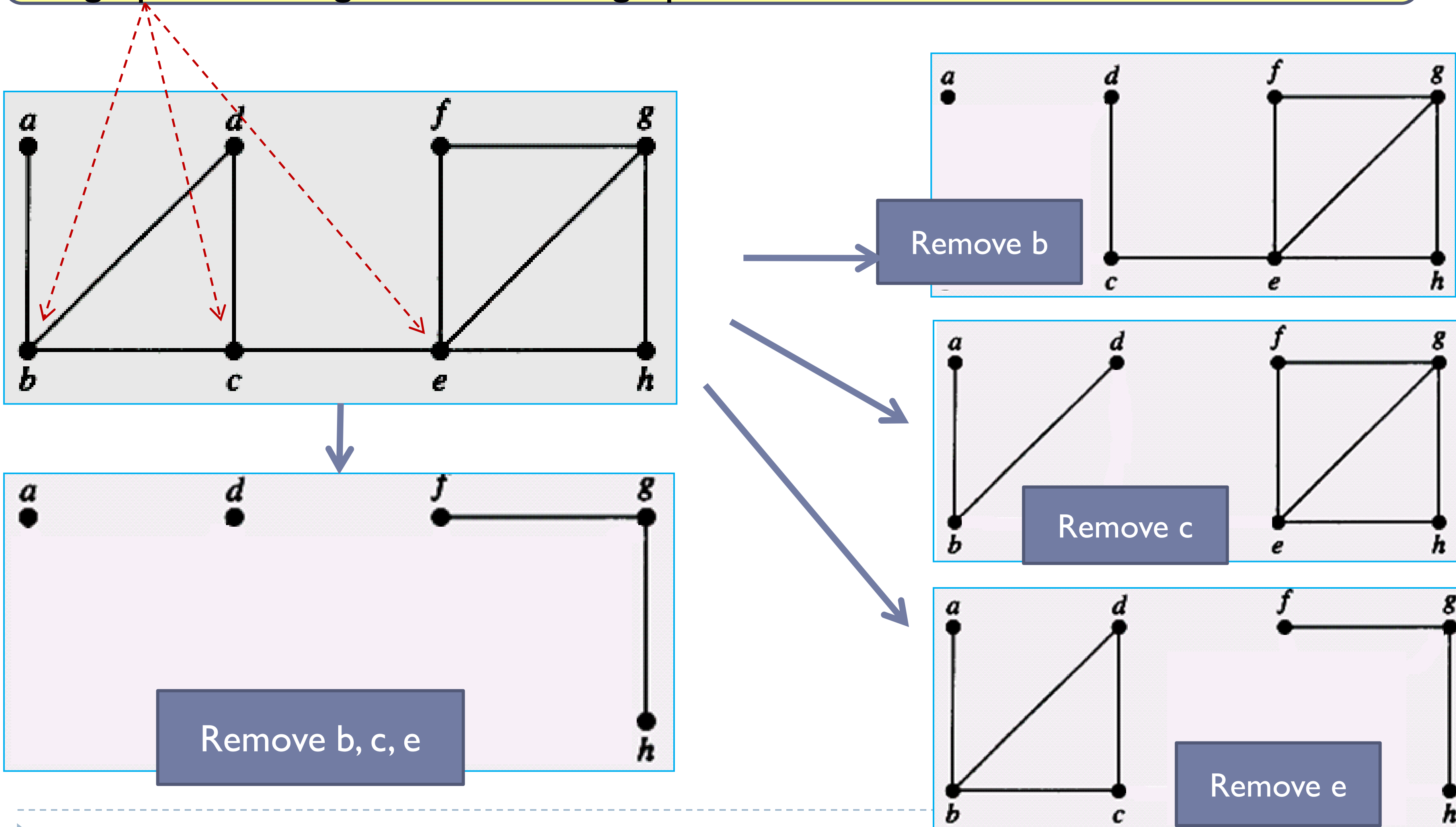
underlying undirected graph of *H*

G: ***strongly connected***
➔ weakly connected

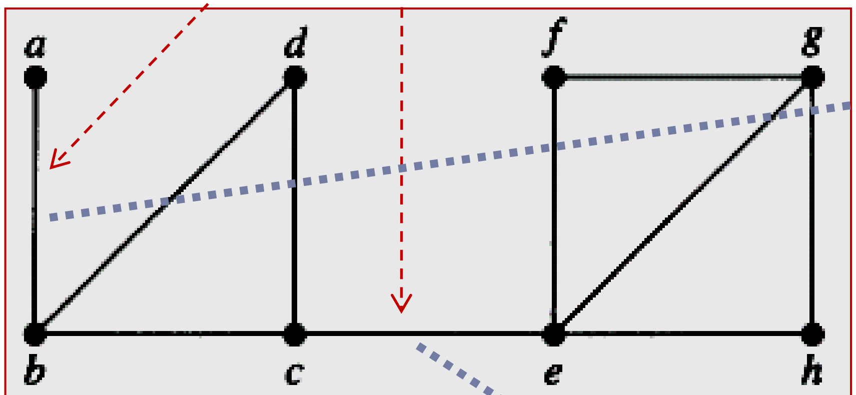
H: ***weakly connected*** ⬅ connected

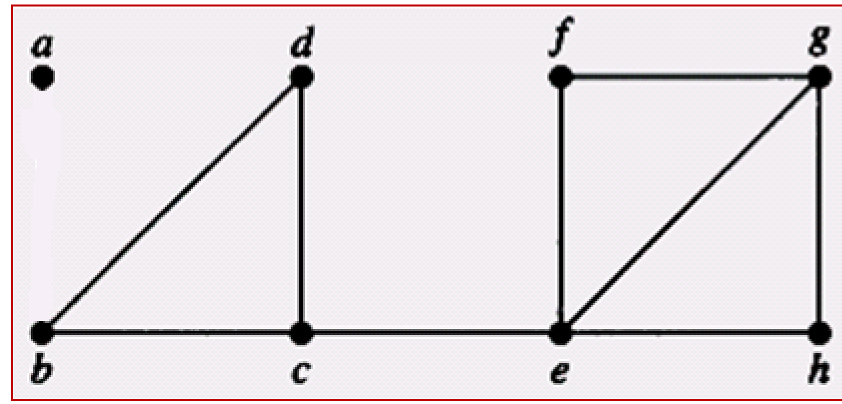
Cut vertex, cut edge

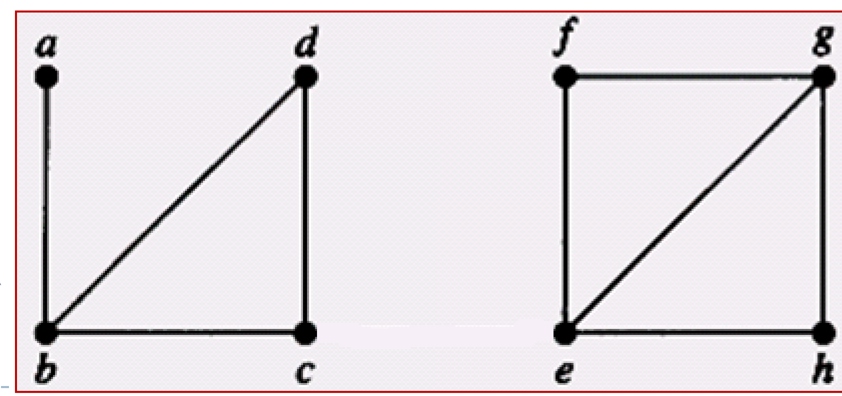
Cut vertex (articulation point): It’s removal will produce disconnected subgraph from original connected graph.



Cut edge (bridge): It's removal will produce subgraphs which are more connected components (thành phần liên thông) than in the original graph

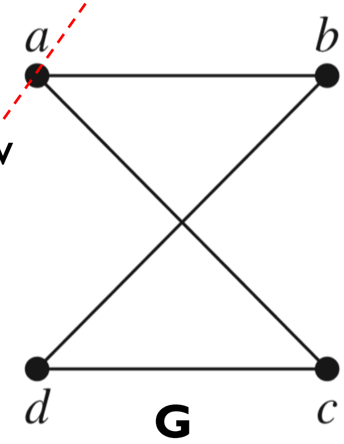






Counting **Paths Between Vertices**

How many *paths of length* ***four*** from *a to d* in the simple graph G?



The *adjacency matrix* of G is given below *(ordering the vertex as a, b, c, d)*

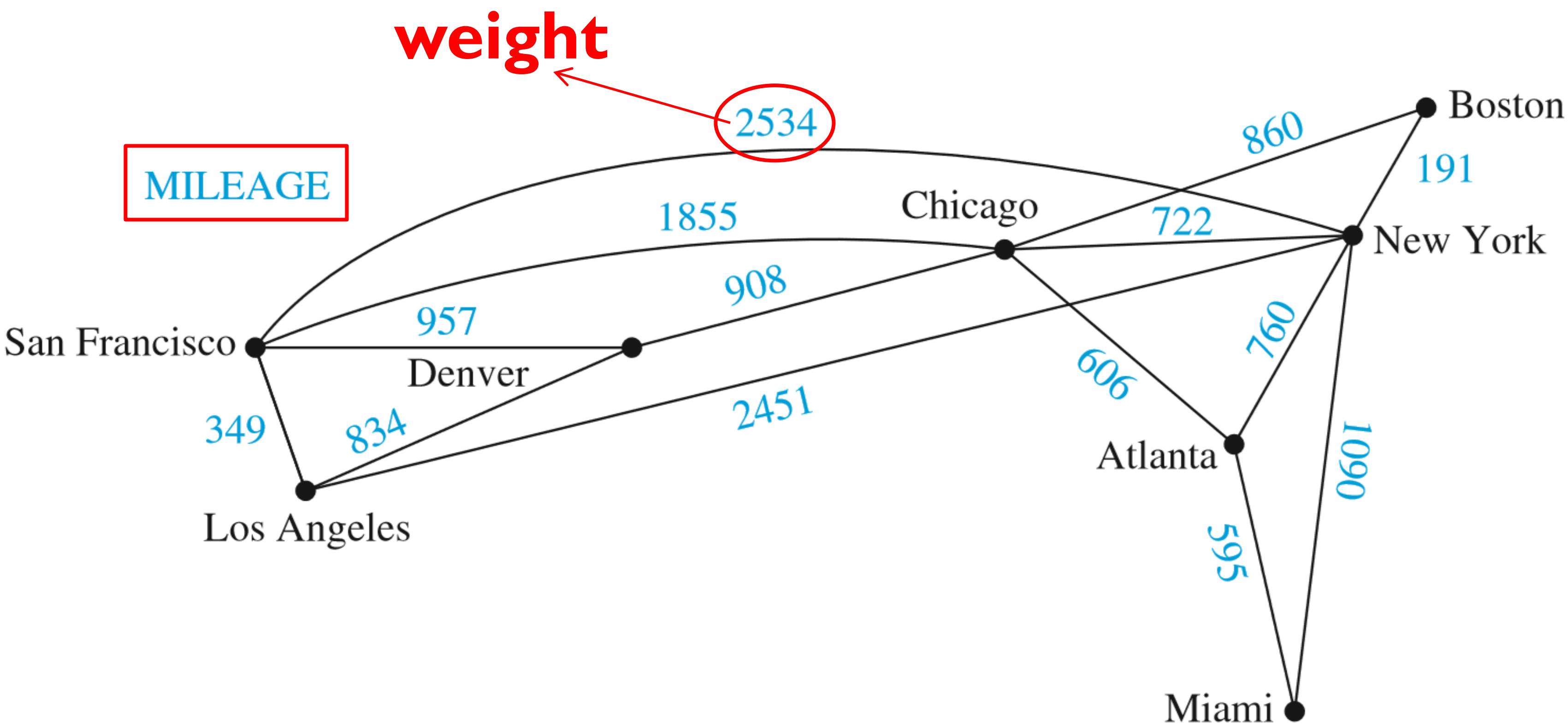
$$\mathbf{A} = \begin{array}{cc} & \textbf{to} \\ \textbf{from} & \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

$$\mathbf{A}^{\boxed{4}} = \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

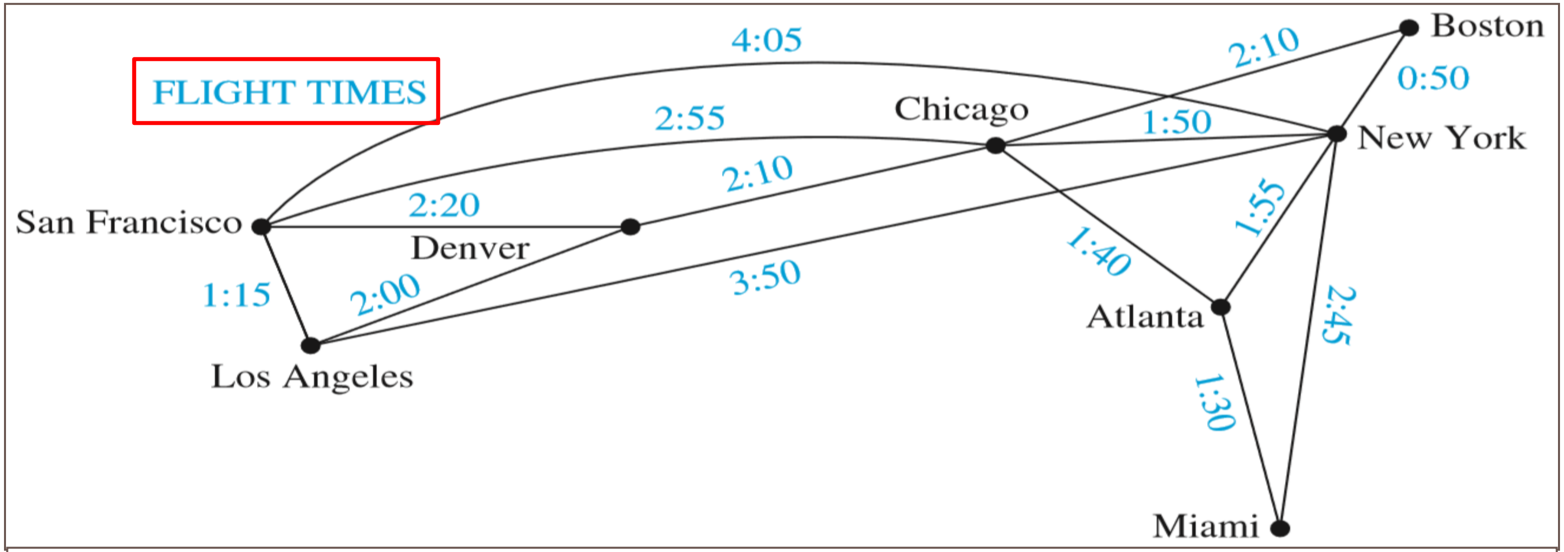
Result = 8

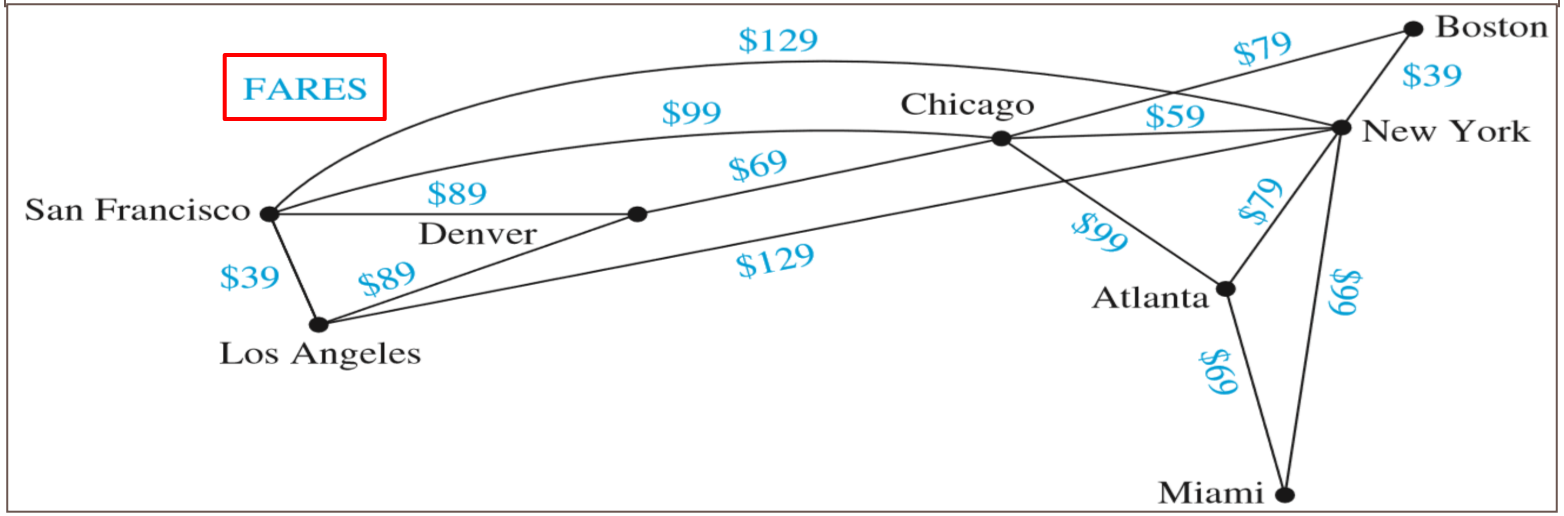
a, b, a, b, d
a, b, a, c, d
a, b, d, b, d
a, b, d, c, d
a, c, a, b, d
a, c, a, c, d
a, c, d, b, d
a, c, d, c, d

Shortest-path problems

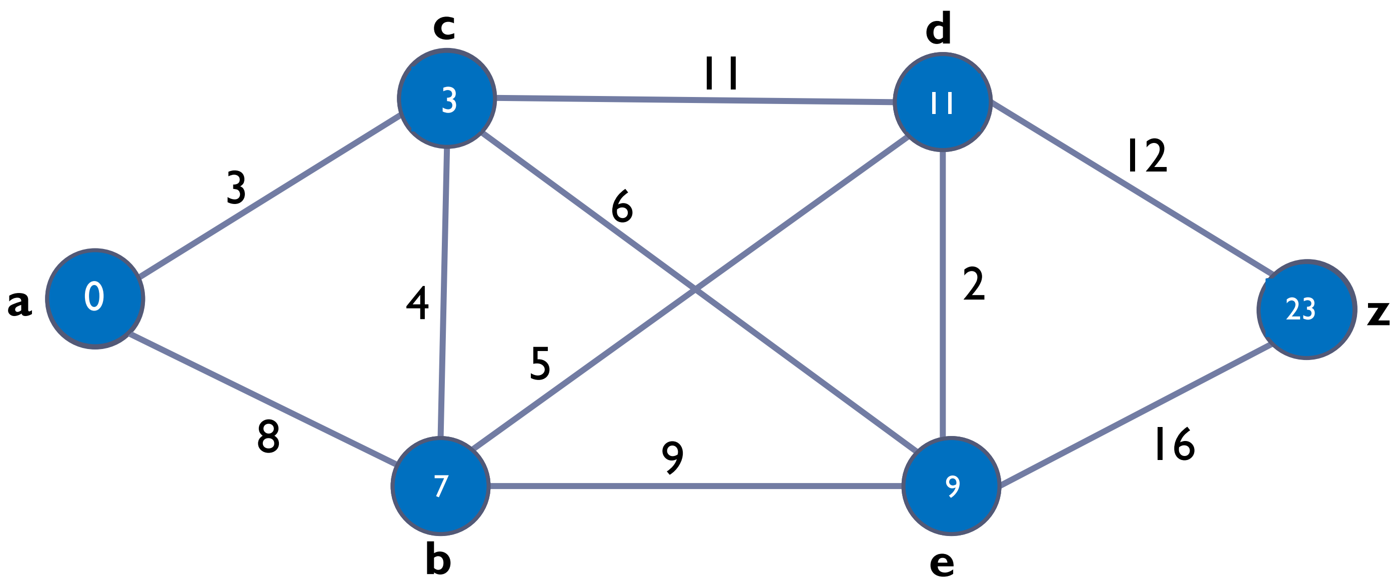


Shortest-path problems





Dijkstra's algorithm



S = {a, c, b, e, d, z}

The ***shortest path*** from a to z: a, c, e, d, z

Dijkstra's Algorithm

ALGORITHM 1 Dijkstra's Algorithm.

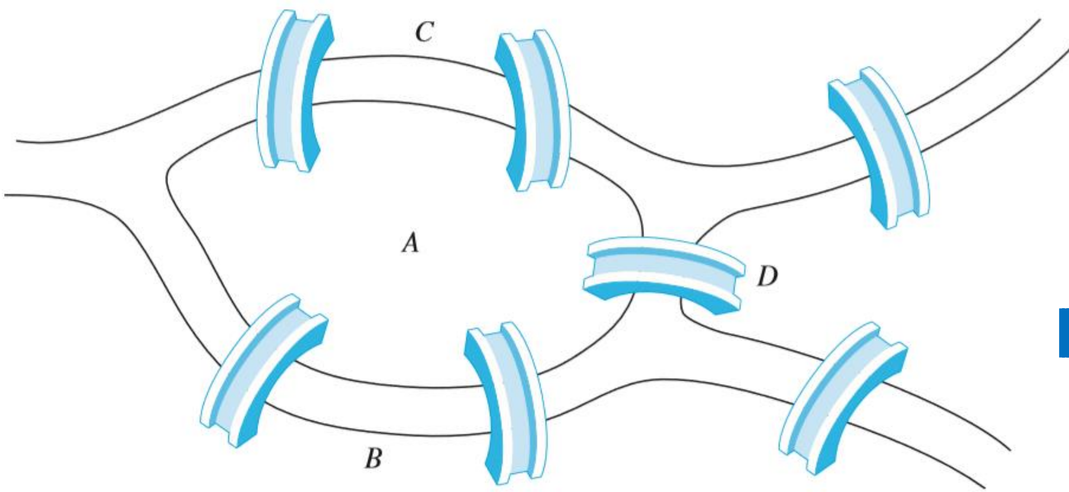
procedure *Dijkstra*(G: weighted connected simple graph, with all weights positive)
{G has vertices $a = v_0, v_1, \ldots, v_n = z$ and weights $w(v_i, v_j)$ where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all other labels are ∞, and S is the empty set}
while $z \notin S$
begin
    $u :=$ a vertex not in S with $L(u)$ minimal
    $S := S \cup \{u\}$
    for all vertices v not in S
        if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
    {this adds a vertex to S with minimal label and updates the labels of vertices not in S}
end {$L(z)$ = length of a shortest path from a to z}

O(n^2)
time complexity

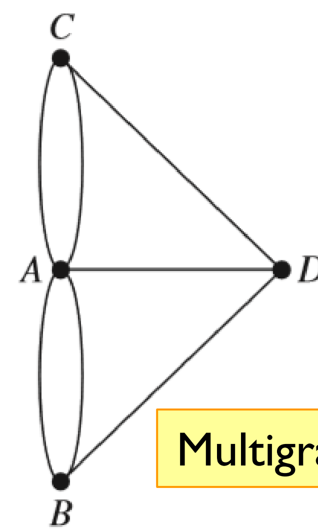
Euler and Hamilton paths - introduction

Can one travel across *all the bridges once* and *return* to the starting point?

LEONHARD **EULER**
(1707–1783)



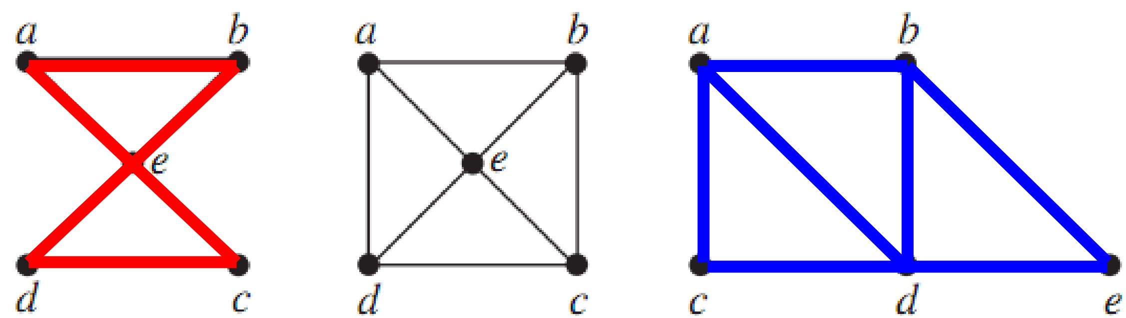
The Seven Bridges of Königsberg.



Multigraph Model

Euler circuit/path - definitions

- An ***Euler path/circuit*** in a graph G is a *simple path/circuit* containing *every edge* of G.



G_1 G_2 G_3

G_1 has an *Euler circuit*
a, b, e, d, c, e, **a**

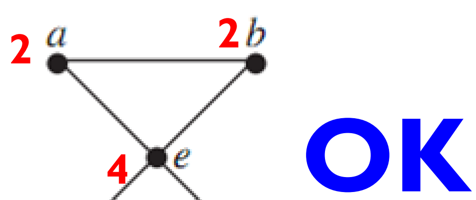
G_3 has *no* an *Euler circuit*,
But G_3 has an *Euler path*
a, c, d, e, b, d, a, **b**

Euler circuit

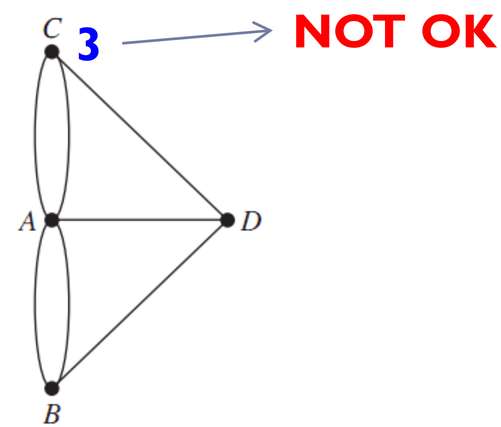
Theorem.

A *connected* multigraph, ≥ 2 vertices,

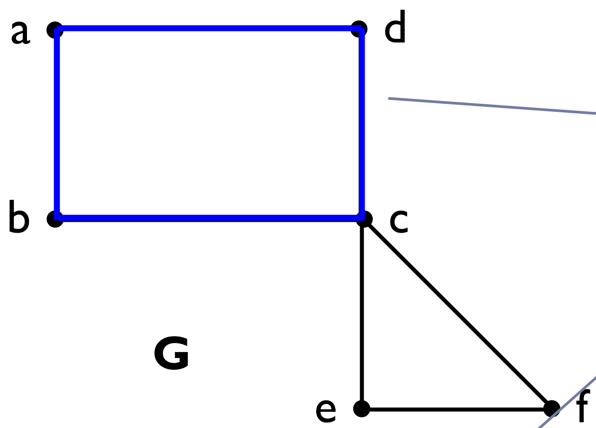
has an *Euler circuit* $\Leftrightarrow$ every vertex has ***even degree***



G_1



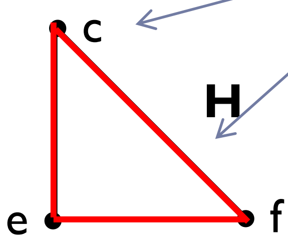
How to construct an Euler circuit?



procedure *Euler*(G: connected, every vertex has even degree)

construct a simple ***circuit*** in G
H:= G – ***circuit*** // remove passed edges
while H has edges
    construct a simple ***subcircuit*** in H beginning at a vertex in ***circuit***
    H: = H – ***subcircuit*** // remove passed edges
    circuit : = add ***subcircuit*** ***to*** ***circuit***

    { ***circuit*** is an *Euler circuit* }



circuit: a, b, c, d, a
subcircuit: c, e, f, c

→ *circuit:* a, b, **c,** e, f, **c,** d, a

Euler path

Theorem.

A connected multigraph has

an **Euler path** but *not an Euler circuit*

$\Leftrightarrow$

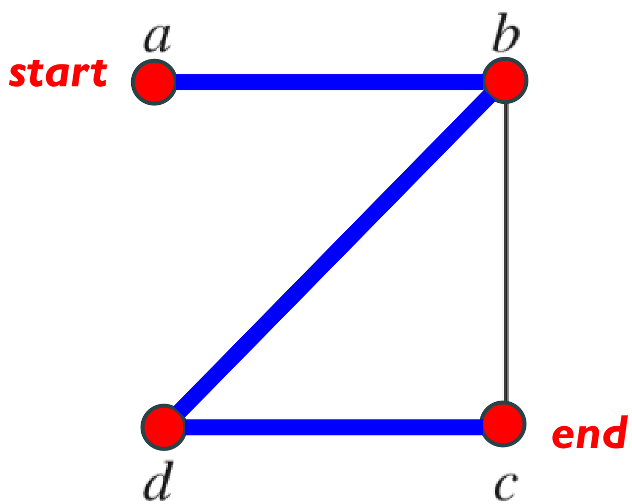
it has *exactly*

two vertices *of odd degree*

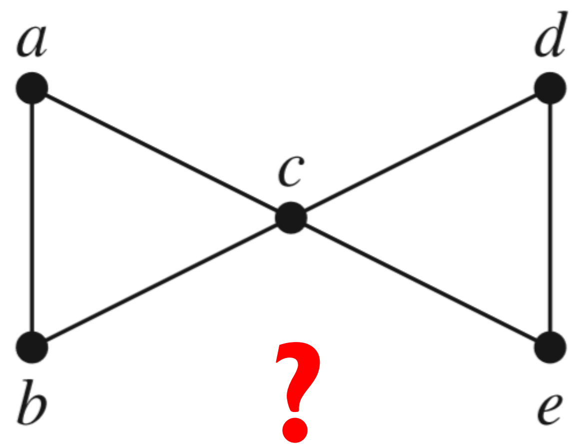
Note that: an Euler circuit is also an Euler path

Hamilton Paths and Circuits

Hamilton circuit/path:
A simple circuit/path passes through ***every vertex*** exactly *once.*

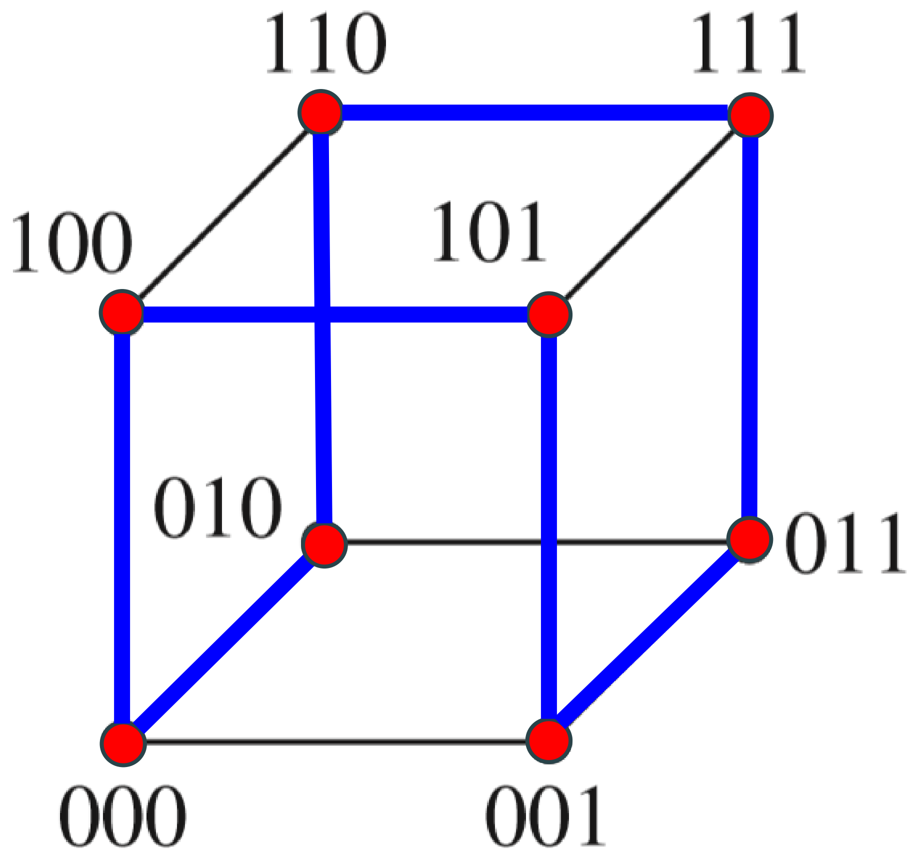


Hamilton ***path***



No *Hamilton circuit*
a,b,c,d,e is a Hamilton path

Example - *A Hamilton* circuit for Q_3



Hamilton Paths and Circuits

There are

NO known simple

***necessary and sufficient* criteria**

for the existence of

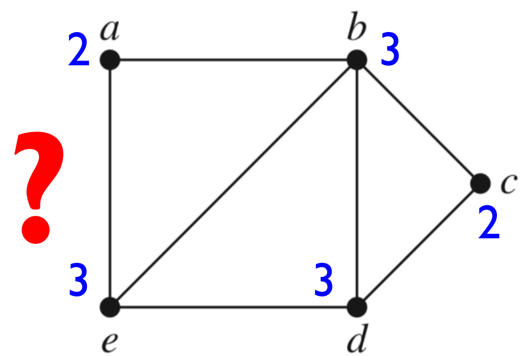
Hamilton circuits

Hamilton circuits - sufficient conditions

Dirac's theorem.

G is a graph:
- *simple*
- n ($\geq$ 3) vertices
- $\forall v_i$, **deg(v_i) $\geq \frac{n}{2}$**

$\Rightarrow$ G has a ***Hamilton circuit***



Ore's theorem.

G is a graph:
- *simple*
- n ($\geq$ 3) vertices
- $\forall$u, $\forall$v, *non-adjacent*
  deg(u) + deg(v) $\geq$ n

$\Rightarrow$ G has a ***Hamilton circuit***

The Traveling Salesman Problem

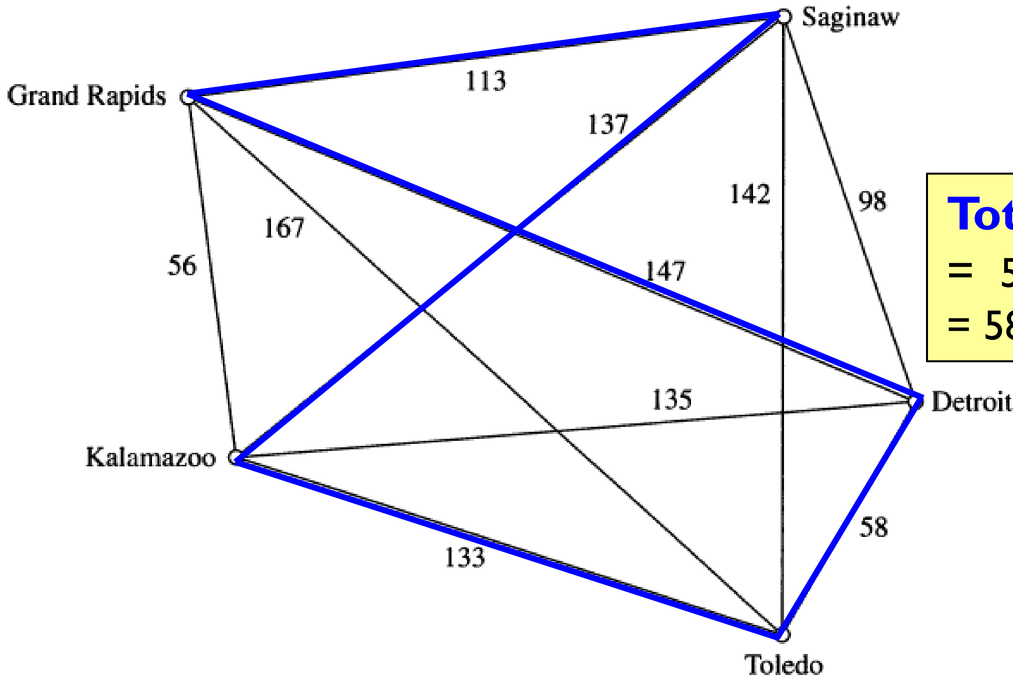


FIGURE 5 **The Graph Showing the Distances between Five Cities.**

Total Distance
= 58 + 133 + 137 + 113 + 147
= 588

Salesman starts in one city (ex. Detroit). He wants to visit n cities *exactly once* and return to his starting point (Detroit). In which order should he visit theses cities to travel the ***minimum total distance*** ?

The Traveling Salesman Problem

Route	*Total Distance (miles)*
Detroit–Toledo–Grand Rapids–Saginaw–Kalamazoo–Detroit	610
Detroit–Toledo–Grand Rapids–Kalamazoo–Saginaw–Detroit	516
Detroit–Toledo–Kalamazoo–Saginaw–Grand Rapids–Detroit	588
Detroit–Toledo–Kalamazoo–Grand Rapids–Saginaw–Detroit	458
Detroit–Toledo–Saginaw–Kalamazoo–Grand Rapids–Detroit	540
Detroit–Toledo–Saginaw–Grand Rapids–Kalamazoo–Detroit	504
Detroit–Saginaw–Toledo–Grand Rapids–Kalamazoo–Detroit	598
Detroit–Saginaw–Toledo–Kalamazoo–Grand Rapids–Detroit	576
Detroit–Saginaw–Kalamazoo–Toledo–Grand Rapids–Detroit	682
Detroit–Saginaw–Grand Rapids–Toledo–Kalamazoo–Detroit	646
Detroit–Grand Rapids–Saginaw–Toledo–Kalamazoo–Detroit	670
Detroit–Grand Rapids–Toledo–Saginaw–Kalamazoo–Detroit	728

$$\frac{4!}{2} = 12$$

Exhaustive search technique // vét cạn

[(n-1) (n-2) (n-2) … 3.2.1]/2 = (n-1) !]/2 ➔ **O((n-1)!)** complexity

➔ **Approximation algorithm**

Summary

- 9.1- Graphs and Graph Models
- 9.2- Graph Terminology and Special Types of Graphs
- 9.3- Representing Graphs and Graph Isomorphism
- 9.4- Connectivity
- 9.5- Euler and Hamilton Paths
- 9.6- Shortest Path Problems

THANKS