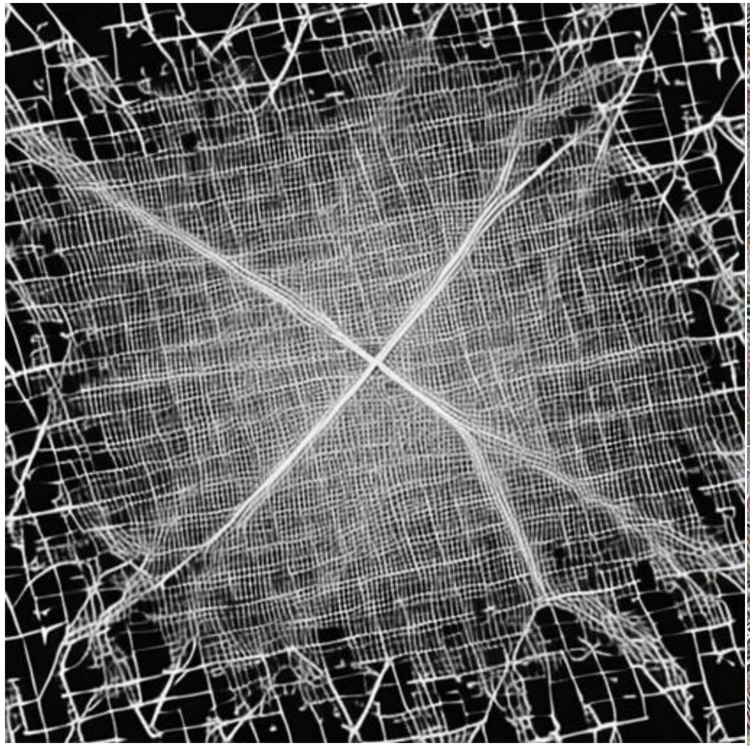
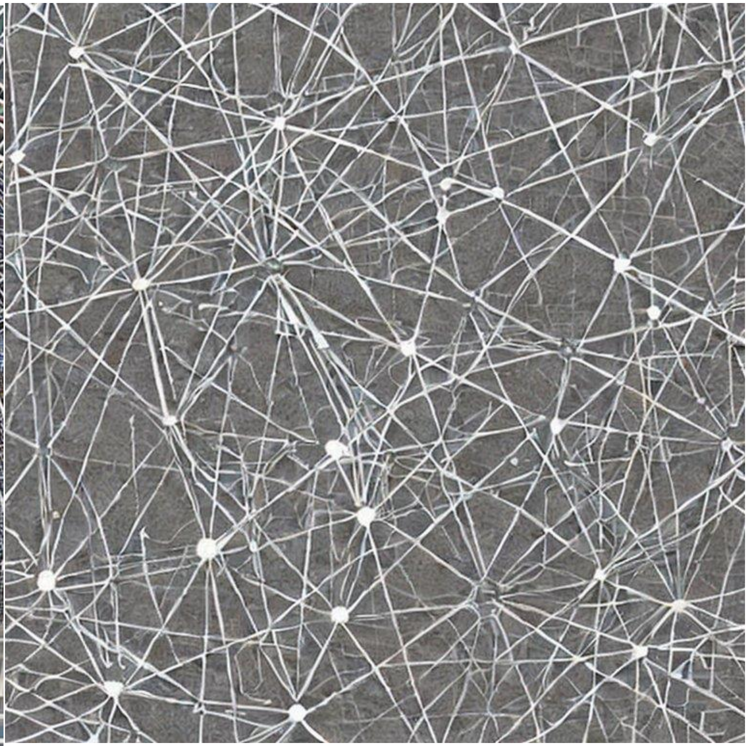
Connectivity in graphs

Mohammed Brahimi

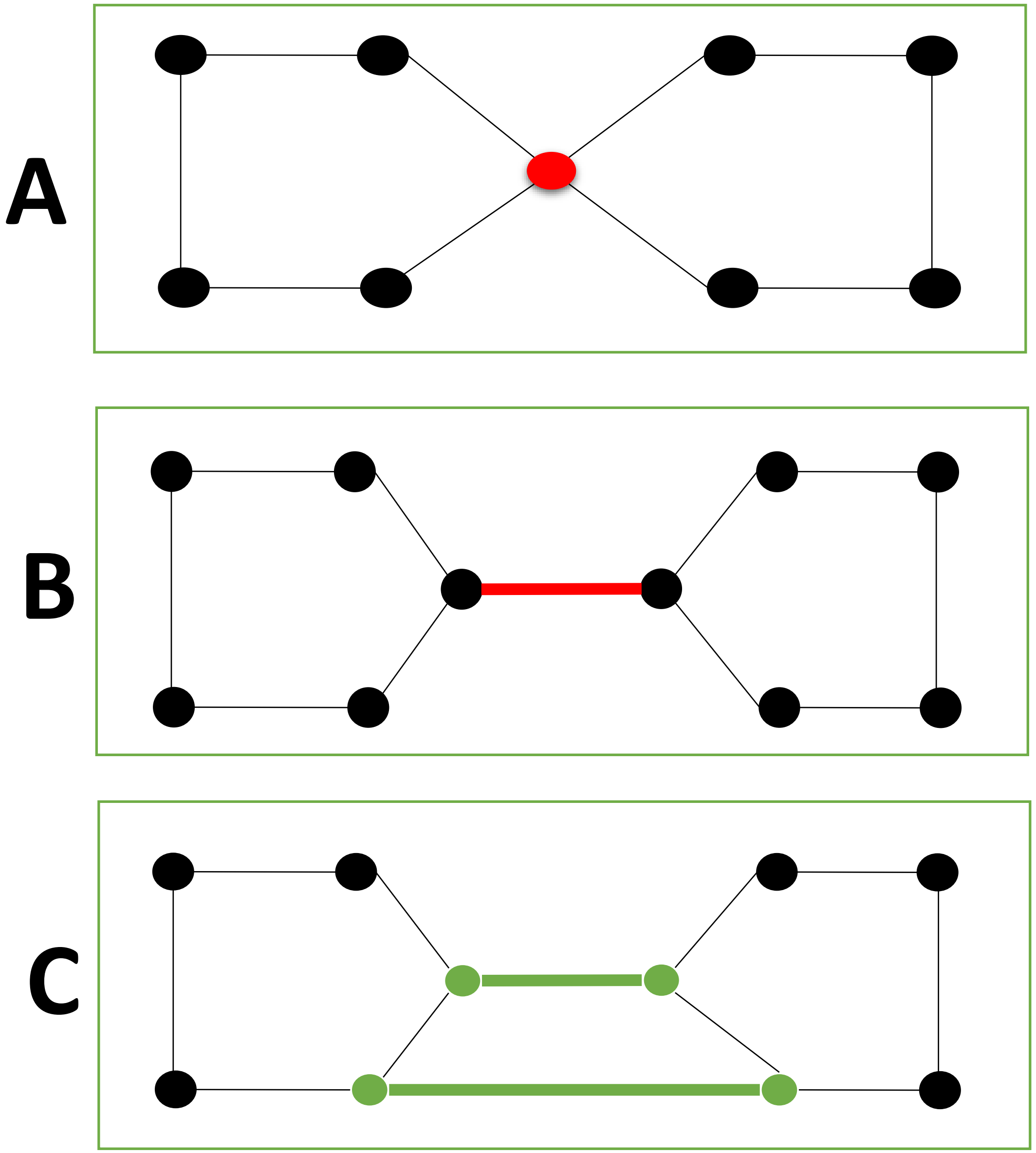




Motivation example

- **Communication** networks represented by graphs **A**, **B**, **C**.
- Vertices are communication centres.
- Edges are communication channels.

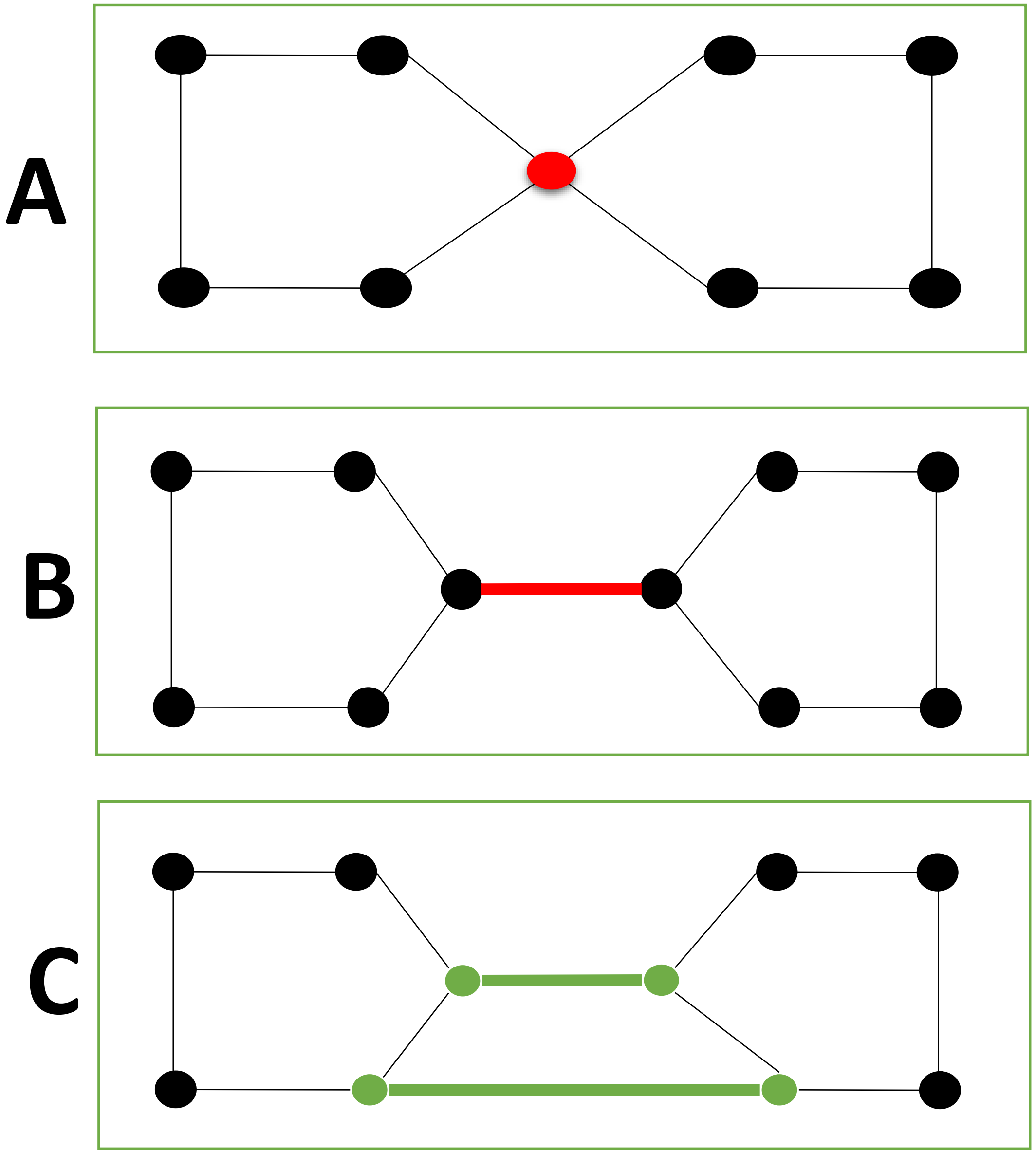
Which of these networks is better and why?



Motivation example

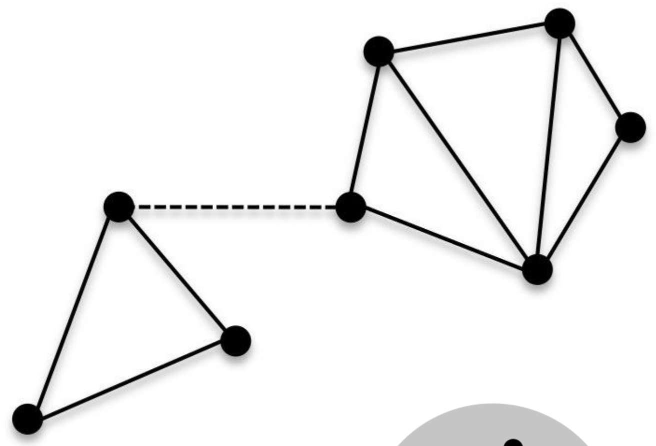
- **Transport** networks represented by graphs **A**, **B**, **C**.
- Vertices are cities.
- Edges are roads.

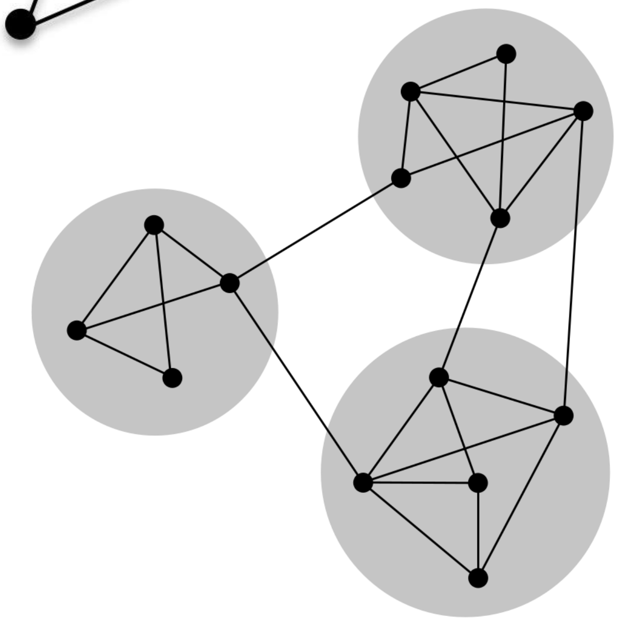
Which of these networks is better and why?



Why study connectivity in graphs ?

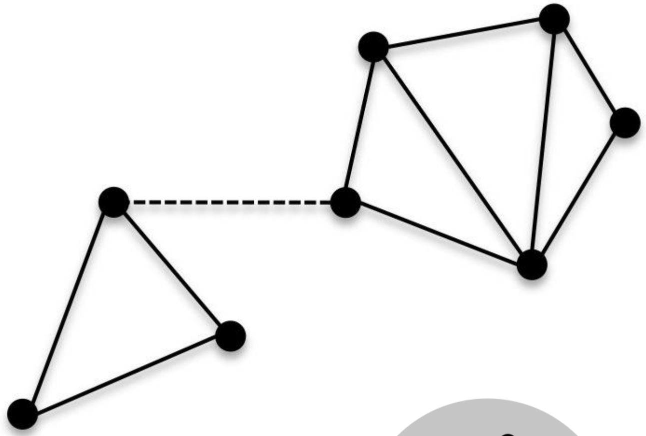
- Connectivity serves as a basis for understanding **the structure** and **behavior** of various networks.
- Connectivity relates to a graph's ability to stay connected after removing vertices or edges.
- Disconnected graphs can hinder information/resource transmission between subgraphs.

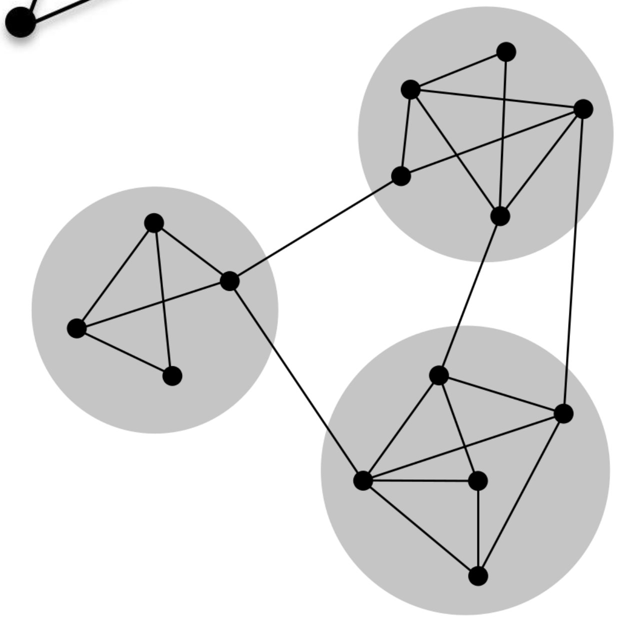




Why study connectivity in graphs ?

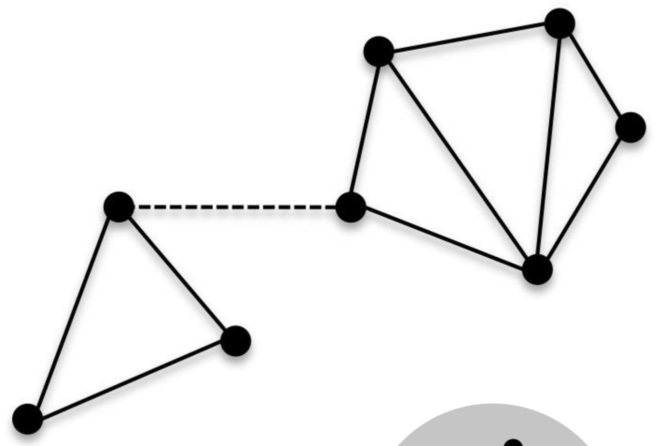
- Studying connectivity in graphs can help us answer questions such as:
  - How many components in the graph?
  - What is the minimum set of vertices/edges to disconnect a graph ?
  - Is there a path between any two vertices in the graph?
  - How easily information / resources be transmitted through the graph?

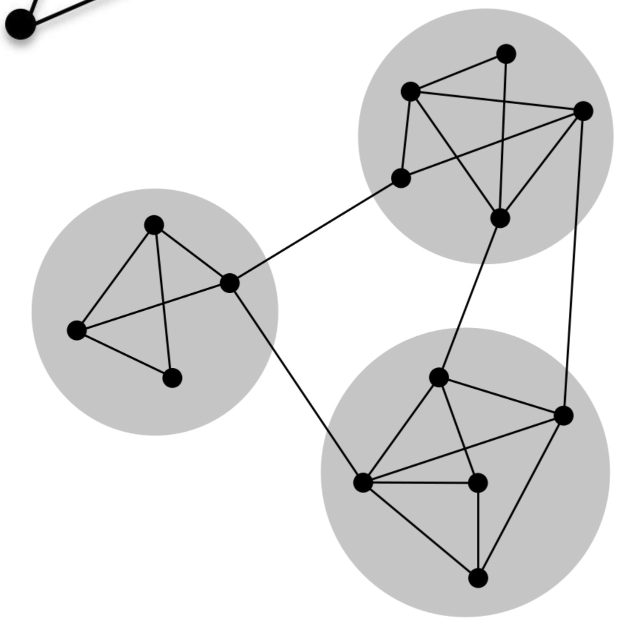




Why study connectivity in graphs ?

- **Network robustness**
  - Resilience assessment.
- **Social network analysis**
  - Relationship strength evaluation.
- **Cluster detection**
  - Identifying communities.
- **Real-world applications**
  - Crucial in diverse fields: biology, power grids, transport ...





Connectivity

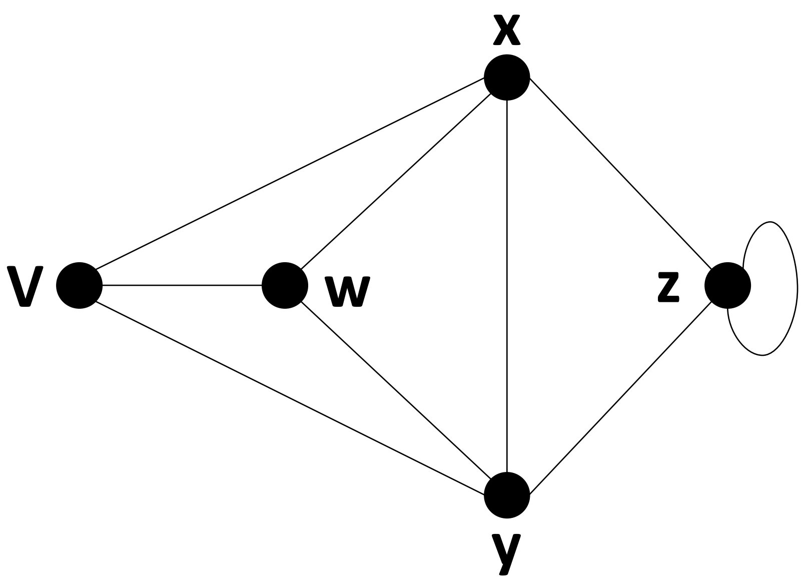
- **Walk**
  - General graph traversal without restrictions.
- **Trail**
  - Walk with distinct edges.
- **Path**
  - Trail with distinct vertices $(v_0, v_1, \dots, v_m)$, except possibly $v_0 = v_m$.
- **Closed Walk/Trail/Path**:
  - $v_0 = v_m$.
- **Cycle**
  - Closed path with at least one edge.

Connectivity

- **Loop**
  - Cycle of length 1.
- **Multiple edges**
  - Cycle of length 2.
- **Triangle**
  - Cycle of length 3.
- **Connected Graph**
  - It exists **a path** between **each pair of vertices**.
- **Disconnected Graph**
  - Several disconnected subgraphs called **components**.

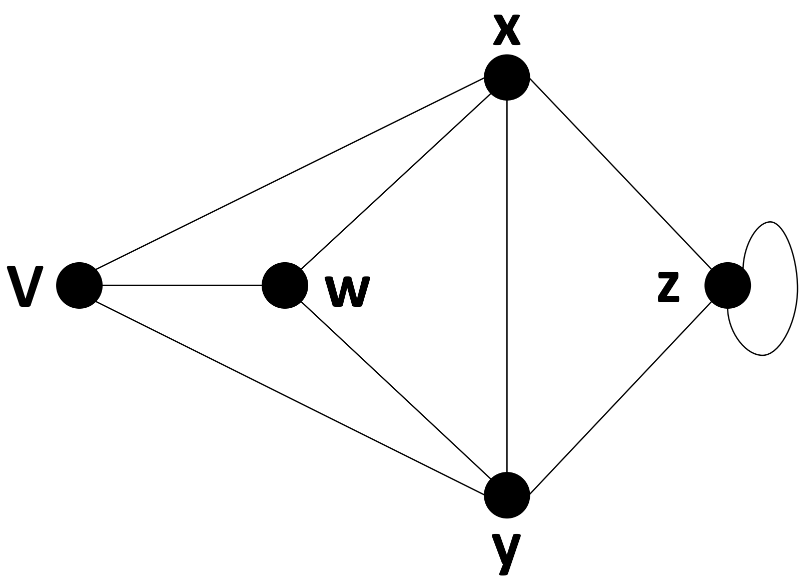
Examples

- $v \ \rightarrow w \rightarrow x \rightarrow y \rightarrow z \rightarrow z \rightarrow x$ is ....
- $v \ \rightarrow w \rightarrow x \rightarrow y \rightarrow z$ is ....
- $v \ \rightarrow w \rightarrow x \rightarrow y \rightarrow z \rightarrow x \rightarrow v$ is ....
- $v \ \rightarrow w \rightarrow x \rightarrow y \rightarrow v$ is ....
- $v \ \rightarrow w \rightarrow x \rightarrow v$ is ....

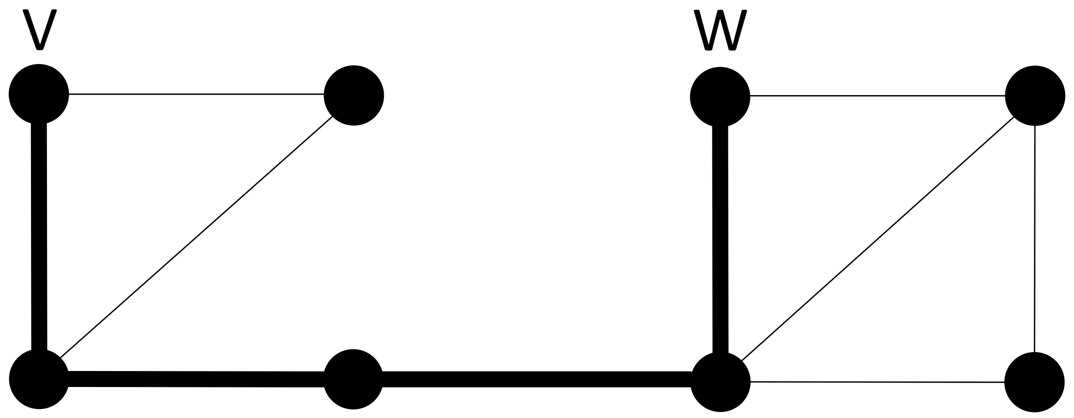


Example

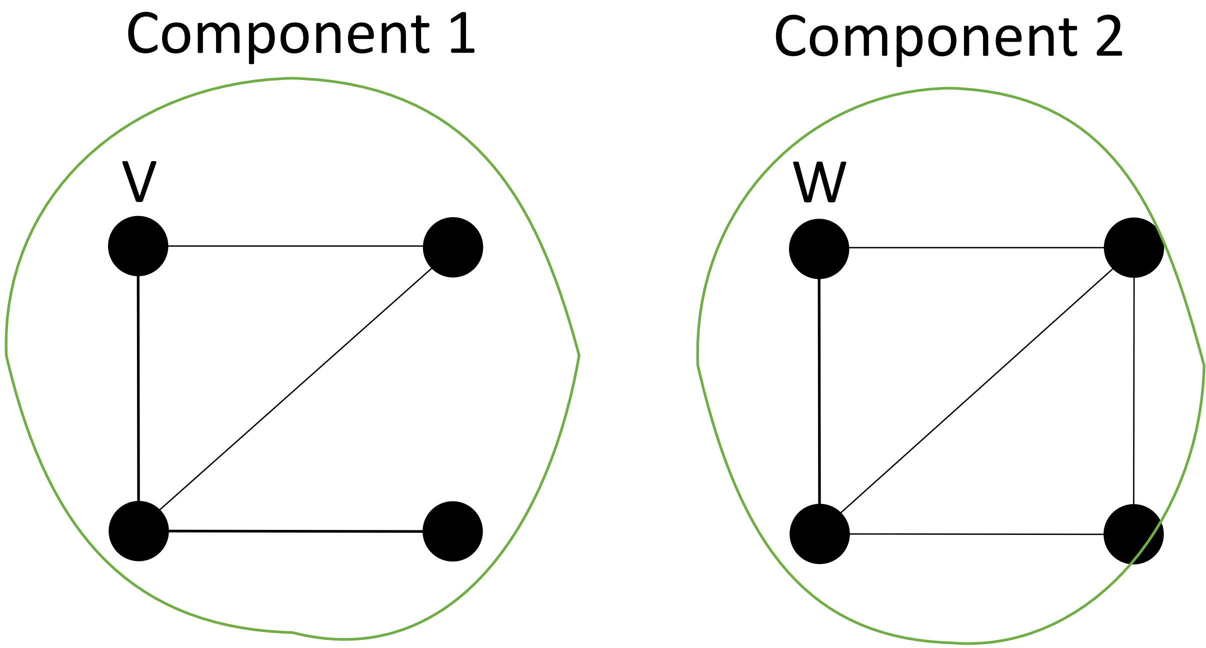
- $v \rightarrow w \rightarrow x \rightarrow y \rightarrow \boldsymbol{z} \rightarrow \boldsymbol{z} \rightarrow x$ is **Trail**
- $v \rightarrow w \rightarrow x \rightarrow y \rightarrow z$ is **Path**
- $v \rightarrow w \rightarrow \boldsymbol{x} \rightarrow y \rightarrow z \rightarrow \boldsymbol{x} \rightarrow v$ is **Closed trail**
- $v \rightarrow w \rightarrow x \rightarrow y \rightarrow v$ is **Cycle (closed path)**
- $v \rightarrow w \rightarrow x \rightarrow v$ is **Triangle**



Examples



Connected graph



Disconnected graph

Cycles and bipartite graphs

THEOREM 1

A graph G is bipartite ***if and only if*** *every cycle of G has even length*

Proof:

- The direction ➔ is easy, when we assume the graph is bipartite.
- The direction 🡐 is tricky, proofing any bipartite graph should not contains cycles with odd length.

Proof of theorem 1:

If the graph is bipartite the each cycle should have even lenght

- Bipartite graphs can be split into two disjoint sets of vertices such that every edge connects a vertex from one set to another.
- If there is a cycle in a bipartite graph and we assume one vertex is in set A, then every other vertex in the cycle alternates between set A and set B.
- Since the cycle must end on a vertex in the opposite set, the length of the cycle must be even.

Proof of theorem 1:

If no even cycle exists then the graph is bipartite

- Take a vertex v
  - A : the set of vertices w for which the shortest path from $\boldsymbol{v}$ **to** $\boldsymbol{w}$ has even length.
  - B : the set of vertices w for which the shortest path from $\boldsymbol{v}$ **to** $\boldsymbol{w}$ has odd length.
- If vertices in set A (or set B) are adjacent ⇒ The shortest paths to vertex v have odd length cycle.
- Therefore, each edge of G connects a vertex in set A and a vertex in set B ⇒ G is bipartite

Number of edges bounds

THEOREM 2

Let G be a simple graph on n vertices. If G has k components, then the number m of edges of G satisfies:

$$n - k \leq m \leq \frac{(n-k)(n-k+1)}{2}$$

Proof

- The intuition of the proof :
  - What is the minimum number of edges to keep k components ?
  - Determine the minimum bound $n-k$
  - What is the maximum number of edges with k components ?
  - Determine the maximum bound $\frac{(n-k)(n-k+1)}{2}$
- Proof by induction in the number of edges.

Number of edges bounds

COROLLARY 3

Any simple graph with n *vertices and more than* $\frac{(\boldsymbol{n}-\mathbf{1})(\boldsymbol{n}-\mathbf{2})}{\mathbf{2}}$ *edges is connected.*

- By using this corollary, one can establish the connectivity of a graph by simply counting its edges.
- The number of edges can be readily determined from the graph's matrix representations

Edge/ Vertex Deleted Subgraphs

Let $G(V, E)$ be a graph

- $F \subseteq E$ be a set of edges of G.
  - The graph $G - F$ is subgraph of G obtained by removing all **edges in $\boldsymbol{F}$** from G.
  - $G - F \ = \ (V, E - F)$.

- $W \ \subseteq V(G)$ be a set of vertices of G.
  - The graph $G - F$ is subgraph of G obtained by removing all **vertices in $\boldsymbol{W}$** from G.
  - If a vertex is removed from a graph, its connected edges are also removed.
  - $G - F \ = \ (V - F, E')$.

Removing edges and vertices

- An approach to study connected graphs is to assess the level of connectivity.

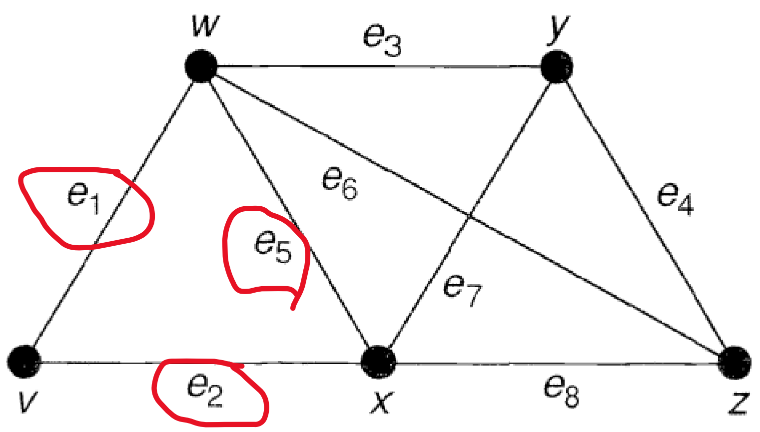
- Study the vulnerability of certain networks, like transport networks.

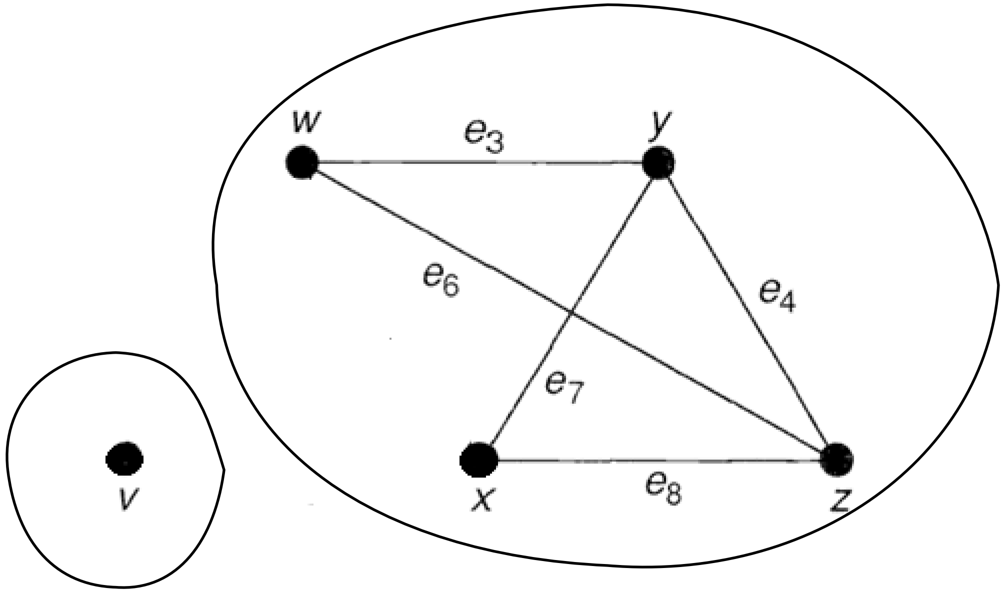
- Determine the minimum number of **edges /vertices** that need to be out of service **for the network to become disconnected**.

- Disconnecting/separating sets help in study these questions .

Removing edges: Disconnecting set

- **Disconnecting set $\boldsymbol{F}$** in a connected graph G is a **set of edges** whose deletion disconnects G. $G - F$ is disconnected.

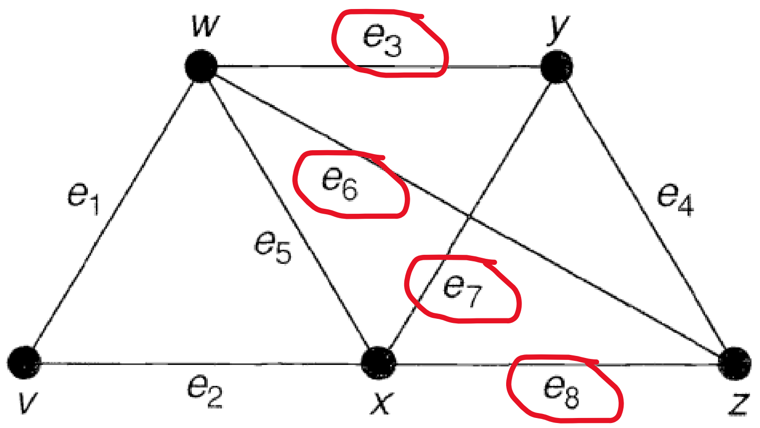


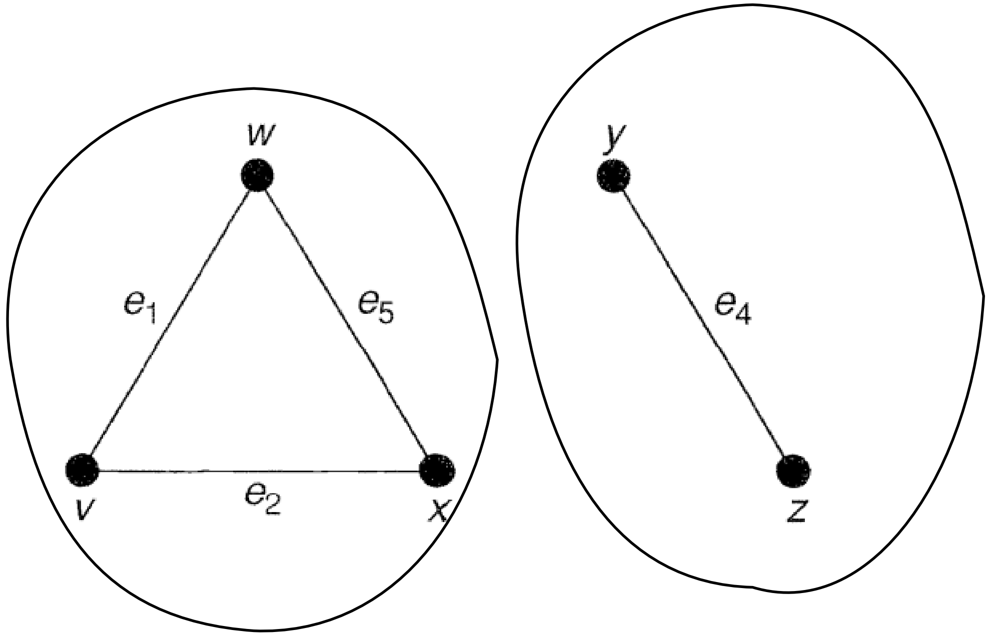


$\{e_1, e_2, e_5\}$ is disconnecting set

Removing edges: Disconnecting set

- **Disconnecting set** in a connected graph G is a set of edges whose deletion disconnects G.

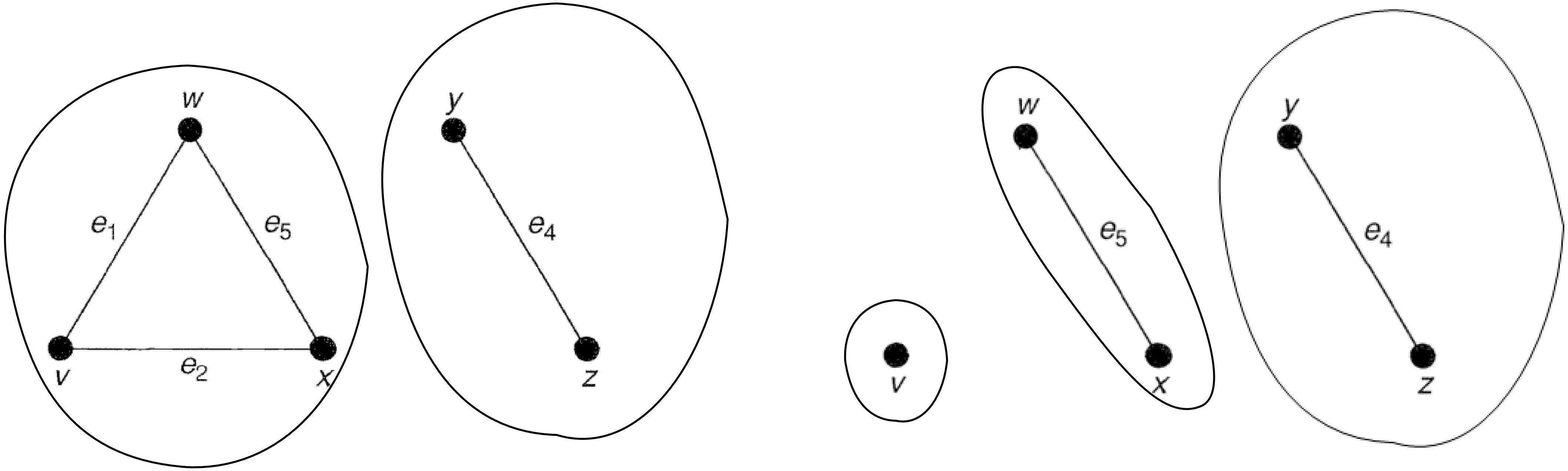




$\{e_3, e_6, e_7, e_8\}$ is disconnecting set

Disconnecting set in disconnected graph

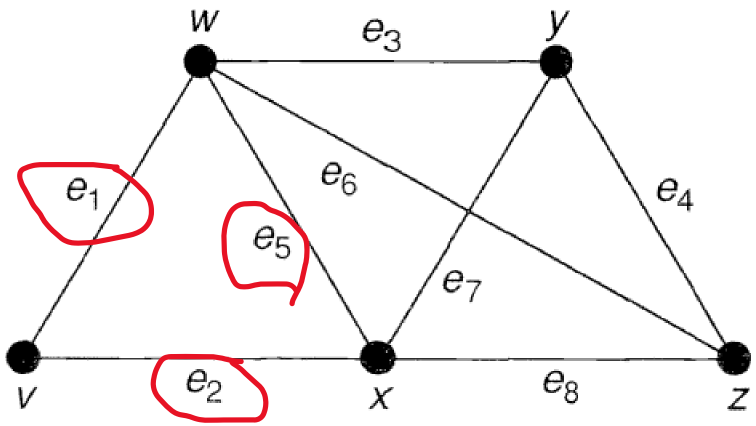
- A disconnecting set in a **disconnected graph** G is a set of edges whose deletion **increases the number of components** of G.

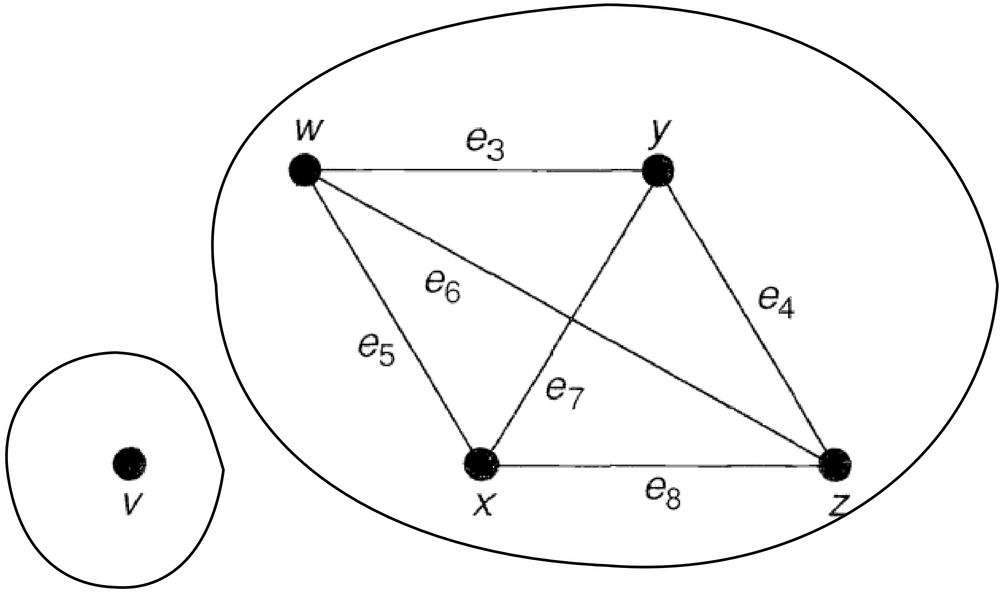


$\{e_1, e_2\}$ is disconnecting set

Disconnecting sets and edge connectivity

- **Cutset** is minimal disconnecting set, no proper subset of a cutset can disconnect the graph.



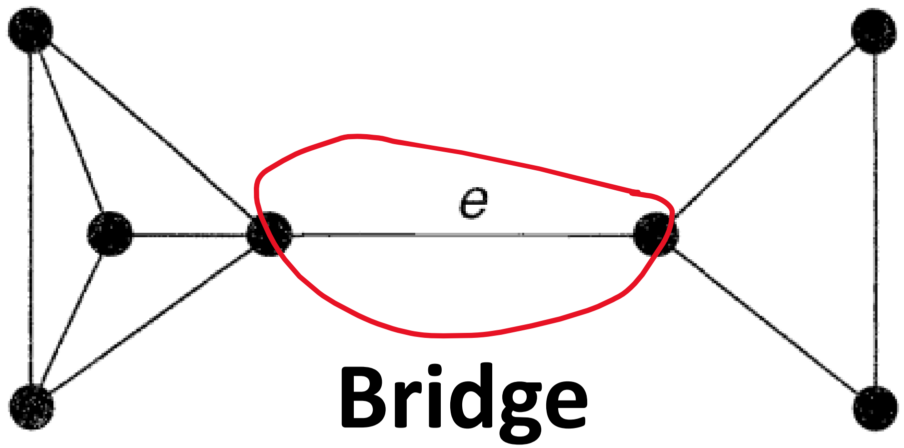


$\{e_1, e_2, e_5\}$ isn't a cutest

$\{e_1, e_2\}$ is a cutest

Disconnecting sets and edge connectivity

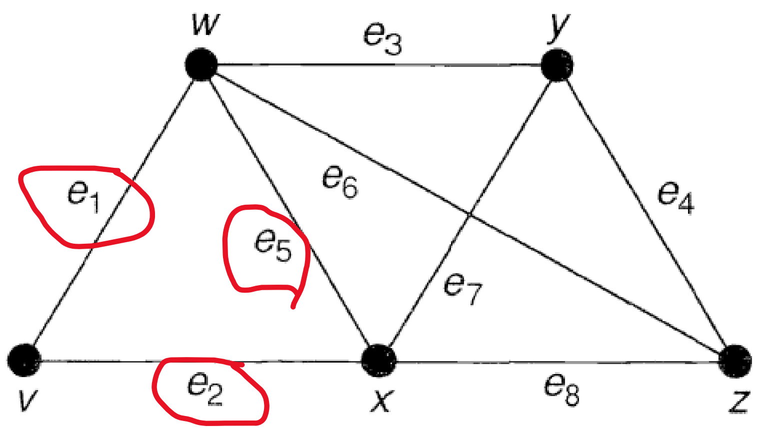
- **Cutset** is minimal disconnecting set, no proper subset of a cutset can disconnect the graph.
- If a cutset has **only one edge** e, we call e a **bridge**.

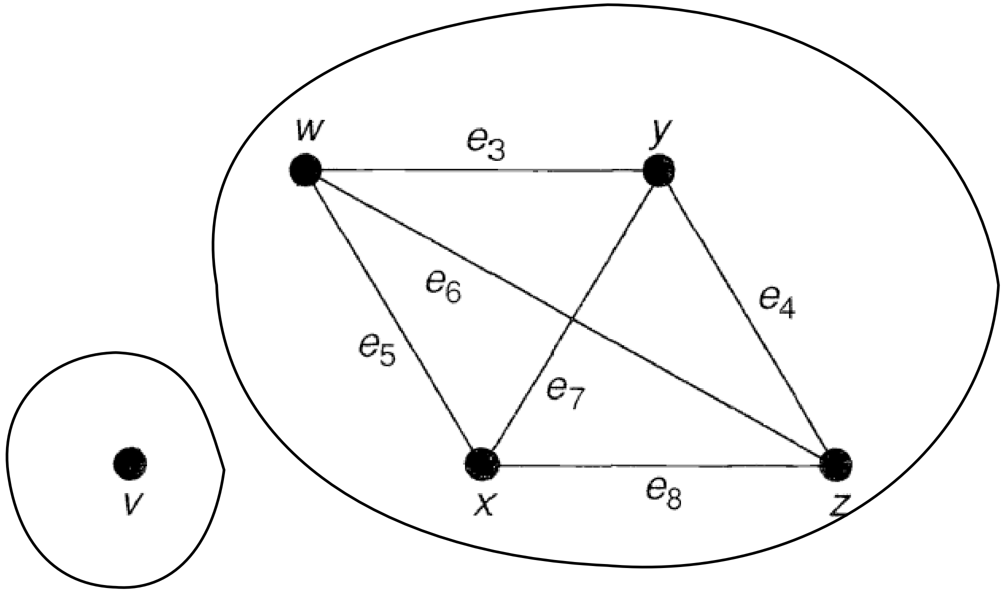


Bridge

Disconnecting sets and edge connectivity

- If G is connected, its edge connectivity $\boldsymbol{\lambda(G)}$ is the size of the **smallest cutset** in G.
- $\boldsymbol{\lambda(G)}$ is the minimum number of edges that we need to delete in order to disconnect G.





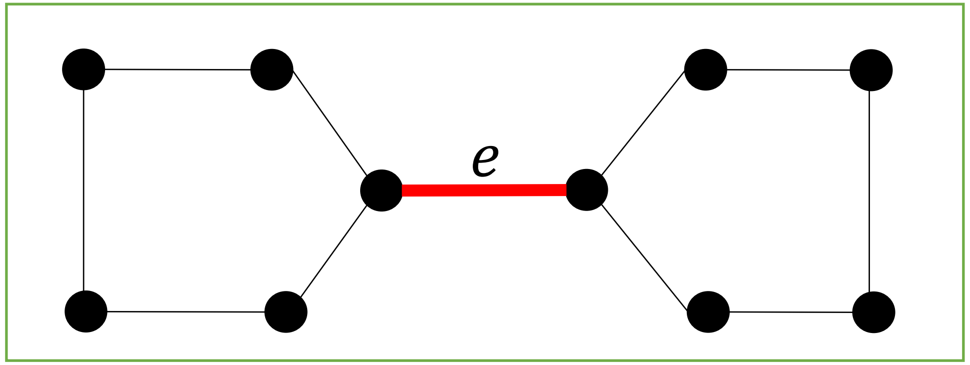
$\{e_1, e_2\}$ is the smallest cutest

$\boldsymbol{\lambda(G)}$ **=2**

Disconnecting sets and edge connectivity

- If G is connected, its edge connectivity $\boldsymbol{\lambda(G)}$ is the size of the **smallest cutset** in G.
- $\boldsymbol{\lambda(G)}$ is the minimum number of edges that we need to delete in order to disconnect G.

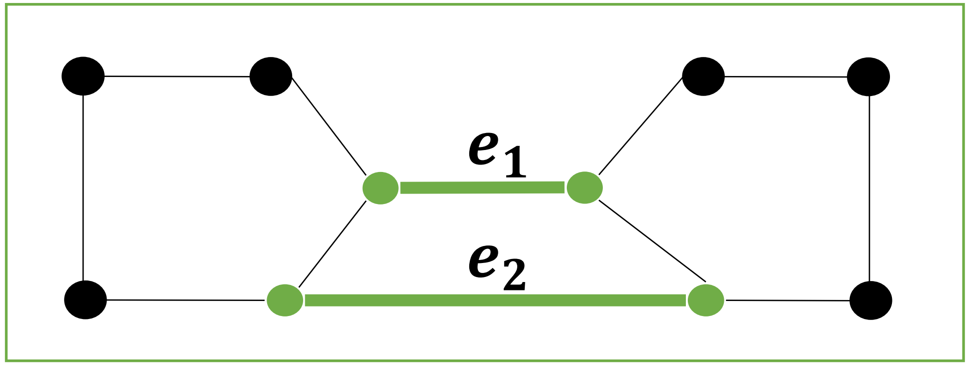
B



The bridge $\{e\}$ is the smallest cutest

$\boldsymbol{\lambda(G)}$ **= 1**

C

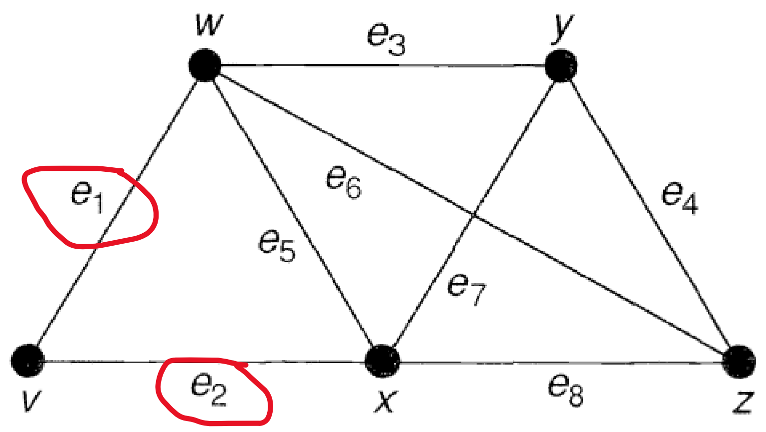


$\{e_1, e_2\}$ is one of the smallest cutsets

$\boldsymbol{\lambda(G)}$ **= 2**

Disconnecting sets and edge connectivity

- If G is connected, its edge connectivity $\boldsymbol{\lambda(G)}$ is the size of the **smallest cutset** in G.
- **K-edge connected** if $\boldsymbol{\lambda(G)}$≥ k.



$\boldsymbol{\lambda(G)}$ **= 2**

G is **1-edge connected** and **2-edge connected**

but

not 3-edge connected

Examples of edge connectivity $\boldsymbol{\lambda(G)}$

Graph type	$\lambda(G)$
Null graph (Trivial graph,)	
Complete graph K_n	
Path graph	
Cycle graph	
Complete bipartite graph K_{mn}	

Examples of edge connectivity $\boldsymbol{\lambda(G)}$

Graph type	$\boldsymbol{\lambda(G)}$
Null graph (Trivial graph)	**0**
Complete graph $\boldsymbol{K_n}$	$\boldsymbol{n-1}$
Path graph	**1**
Cycle graph	**2**
Complete bipartite graph $\boldsymbol{K_{mn}}$	$\boldsymbol{\min(m,n)}$

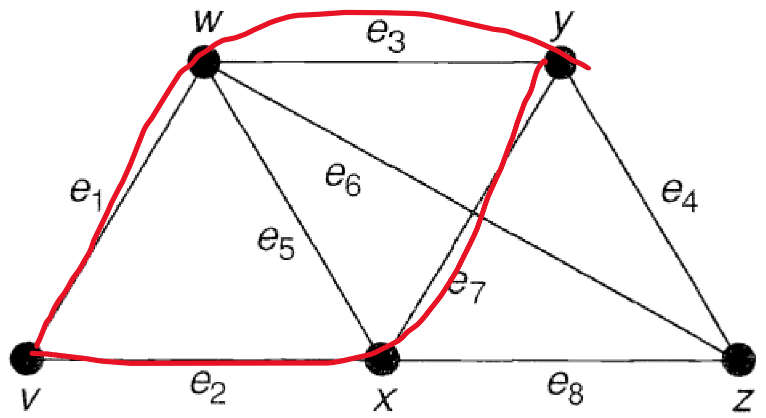
K-edge-connected: Menger's theorem

THEOREM 3

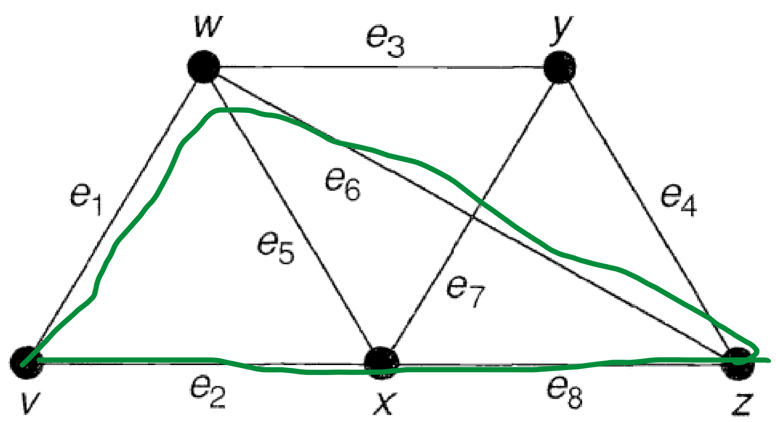
*A graph G is K-edge-connected **if and if only if** any two distinct vertices are joined by at least k paths, no two of which have any edges in common.*

- It can be proved that a graph is 2-edge-connected if and only if any two distinct vertices are joined by at least two paths with no edges in common.

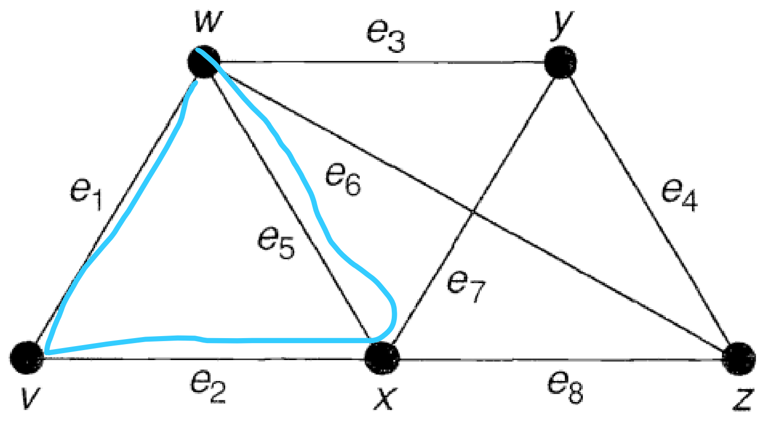
Example



Example

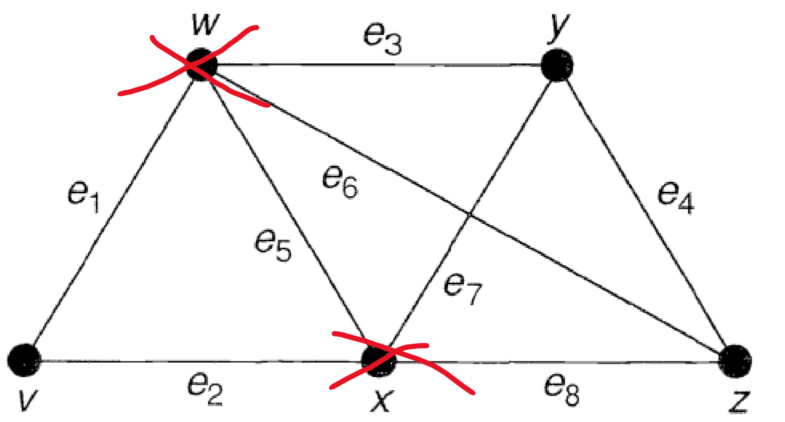


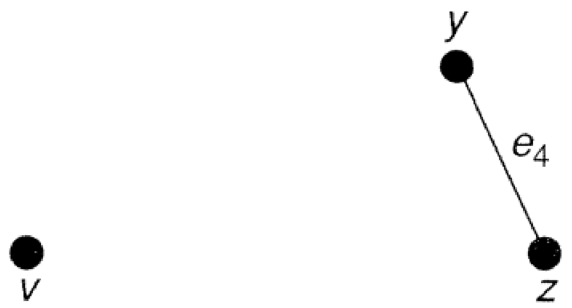
Example



Removing vertices: Separating set

- **Separating set $\boldsymbol{W}$** in a connected graph G is a **set of vertices** whose deletion disconnects G. $G - W$ is disconnected.
- If a vertex is deleted, then **its incident edges are also removed**.





$\{w, x\}$ isn't a separating set

Removing vertices: Separating set

- **Separating set $\boldsymbol{W}$** in a connected graph G is a **set of vertices** whose deletion disconnects G. $G - W$ is disconnected.
- **Cut-vertex** is Separating set with only one vertex

v is a cut-vertex

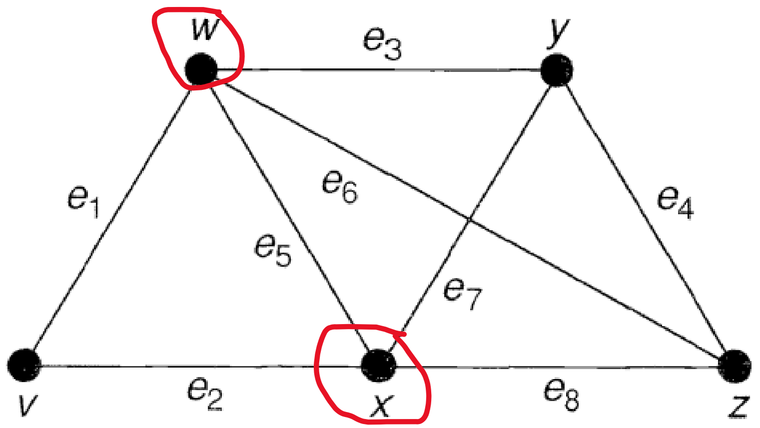
Removing vertices: Separating set

- **Separating set $\boldsymbol{W}$** in a connected graph G is a **set of vertices** whose deletion disconnects G. $G - W$ is disconnected.
- **Cut-vertex** is Separating set with only one vertex

This definition can be extended to disconnected graphs like Disconnecting edges

Separating sets and vertex connectivity

- (vertex) **connectivity** $\boldsymbol{\kappa(G)}$ is the size of the smallest separating set in G.
- $\boldsymbol{\kappa(G)}$ is the minimum number of vertices that we need to delete in order to disconnect G.
- **k-connected** if $\boldsymbol{\kappa(G) \geq k}$



$\boldsymbol{\kappa(G)}$ **= 2**

G is **1 connected** and **2-edge connected**

but not 3-edge connected

Examples of vertex connectivity $\boldsymbol{\kappa(G)}$

Graph type	$\kappa(G)$
Null graph (Trivial graph)	
Complete graph K_n	
Path graph	
Cycle graph	
Complete bipartite graph K_{mn}	

Examples of vertex connectivity $\boldsymbol{\kappa(G)}$

Graph type	$\boldsymbol{\kappa(G)}$
Null graph (Trivial graph)	**0**
Complete graph K_n	**n-1 (here the definition should be modified)**
Path graph	**1**
Cycle graph	**2**
Complete bipartite graph K_{mn}	**!!!!**

K-connected: Menger's theorem

THEOREM 3

A graph G with at least $k+1$ is K-connected ***if and if only if*** *any two vertices are joined by at least k paths, no two of which have any edges in common.*

It can be proved that a graph with at least three vertices is 2-connected if and only if any two distinct vertices are joined by at least two paths with no edges in common (Exercise).

Edge connectivity VS vertex connectivity

$$\boldsymbol{\kappa(G)} \ \leq \boldsymbol{\lambda(G)} \leq \boldsymbol{\delta(G)}$$

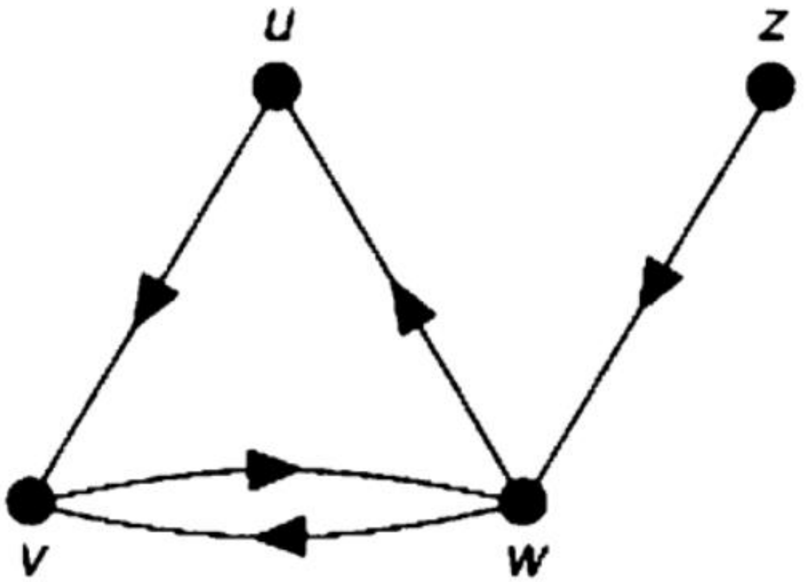
- The vertex connectivity is smaller than edge connectivity.
- $\boldsymbol{\delta(G)}$: is the smallest vertex-degree in G.

Connectivity in digraphs

- **Walk**
  - Finite sequence of arcs without any restriction .
- **Trail**
  - Walk with distinct arcs. It can contain the arcs vw and wv.
- **Path**
  - Trail with distinct vertices $(v_0, v_1, \dots, v_m)$, except possibly $v_0 = v_m$.
- **Closed Walk/Trail/Path**:
  - $v_0 = v_m$.
- **Cycle**
  - Closed path with at least one arc.

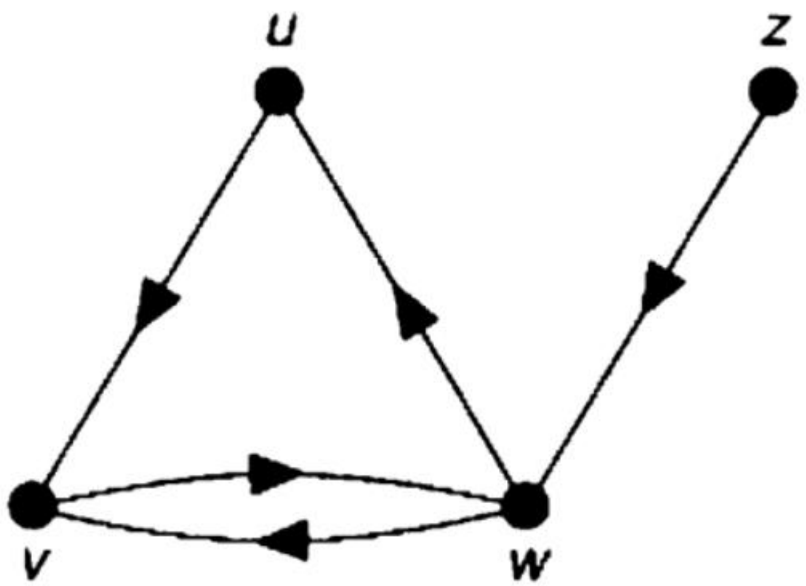
Example

- z → w → v → w → u → v → w is ……
- z → w → v → w → u is ………
- z → w → v → u is ………
- w → u → v → w is ……….



Example

- z → w → v → w → u → v → w is **Walk.**
- z → w → v → w → u is **Trail.**
- z → w → u → v is **Path.**
- w → u → v → w is **Closed path (Cycle).**



Connectivity in digraphs

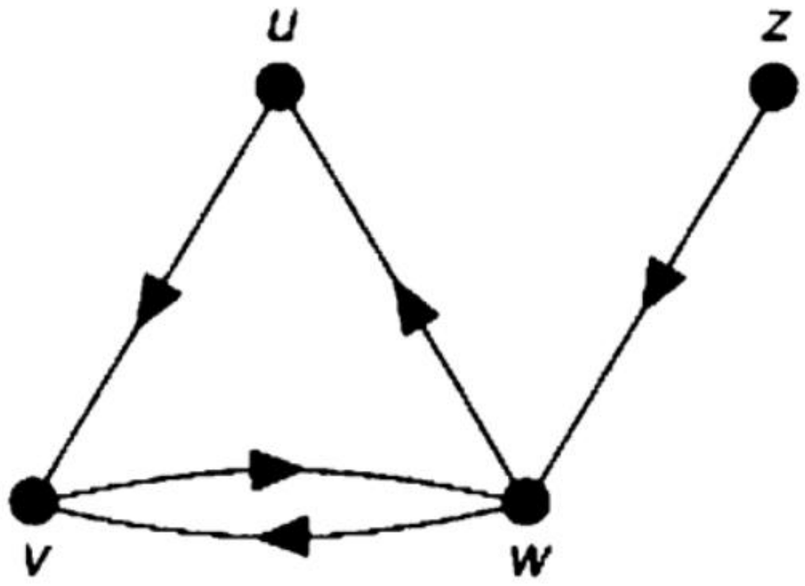
- Connectivity can be defined for digraphs, with two useful types corresponding to whether or not the directions of the arcs are considered.
- **Weak connectivity**:
  - A digraph D is connected if it underlying graph of D is connected.
- **Strong connectivity**:
  - A digraph D is strongly connected if there is a directed path from any vertex to any other vertex.

Connectivity in digraphs

- Every strongly connected digraph is connected, but not all connected digraphs are strongly connected.
- The distinction between connected and strongly connected digraphs can be illustrated with a one-way street map of a city.
- If the map is connected, we can drive from any part of the city to any other, ignoring one-way street directions.
- If it is strongly connected, we can drive from any part of the city to any other, following one-way street directions.

Example

- This graph is **weakly connected** but not **strongly connected.**
- We can not find a path from **u** to **z**.
- If we add an arc from **w** to **z**, the graph becomes **strongly connected**.

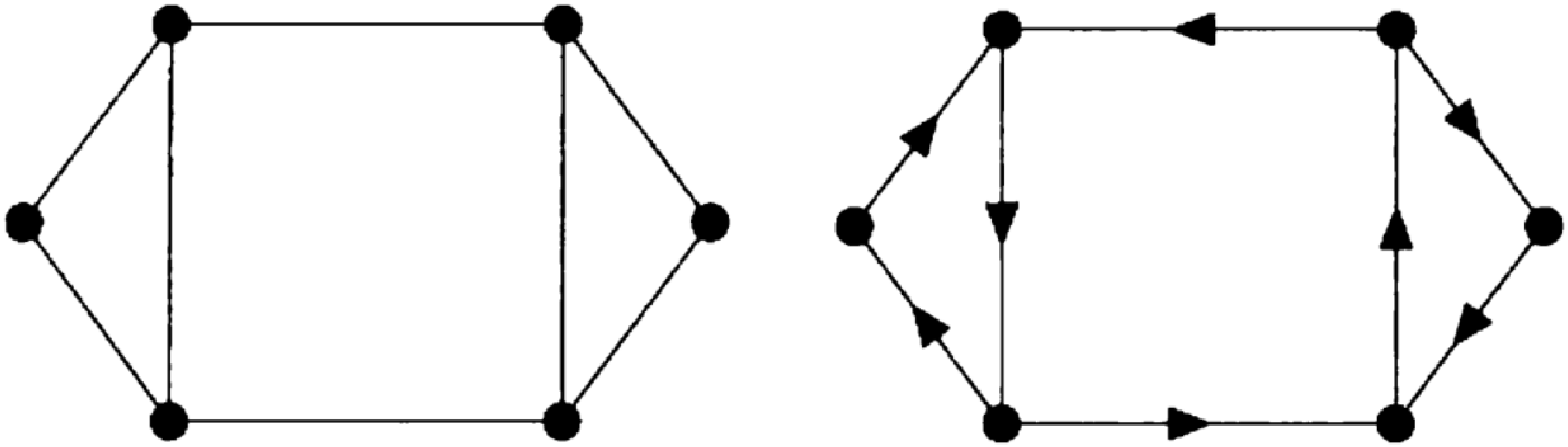


Orientable undirected graphs

- Question:
  - Can a one-way system be implemented on an undirected graph in a way that enables driving from any part of the city to any other?
- It's not always possible
  - If a city consists of two parts connected by a single bridge.
  - If there are no bridges, then a one-way system can always be imposed.
- The presence of bridges may prevent the imposition of a one-way system, cutting off one part of the city.

Example

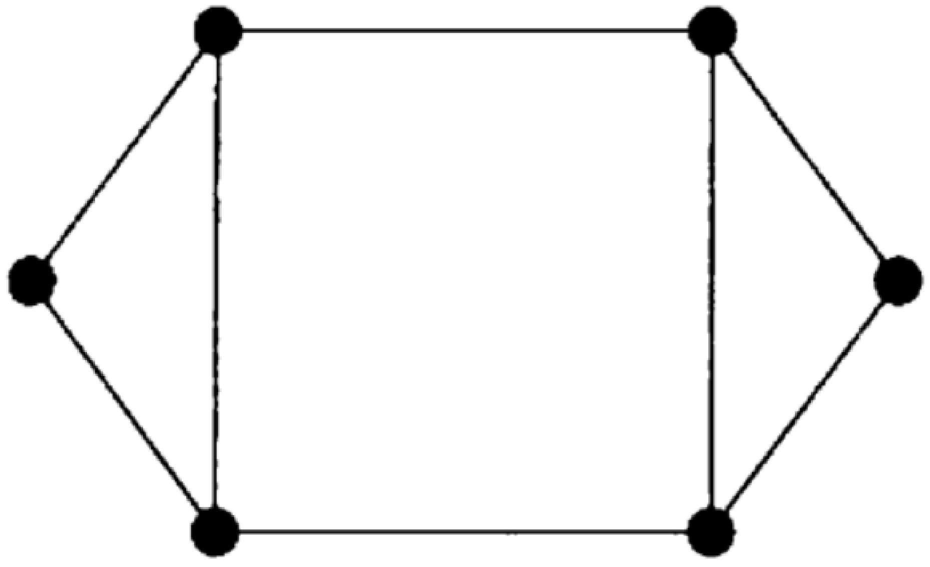
Convert this undirected graph to strongly connected directed graph

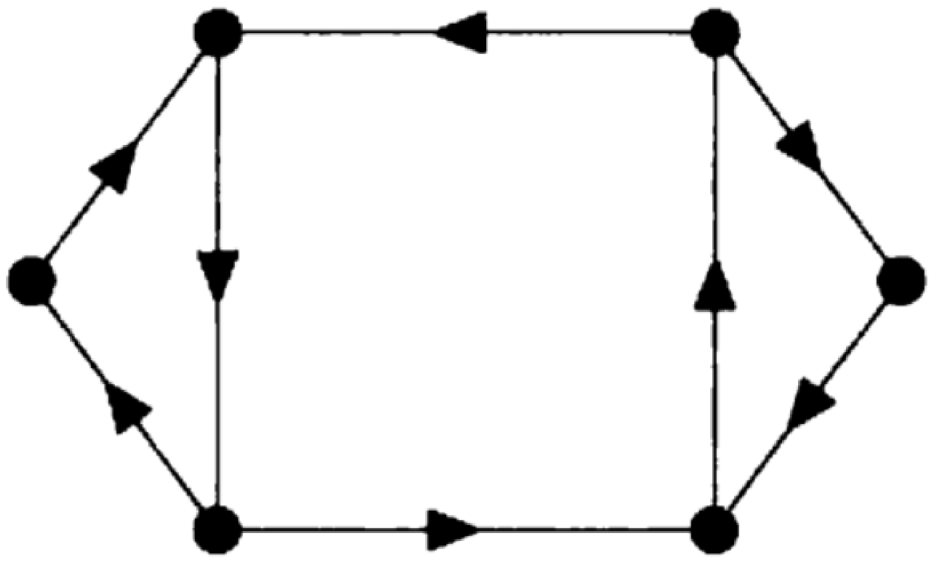


Orientable undirected graphs

THEOREM 4

A connected G graph is orientable if and if only if each edge of G lies at least one cycle.

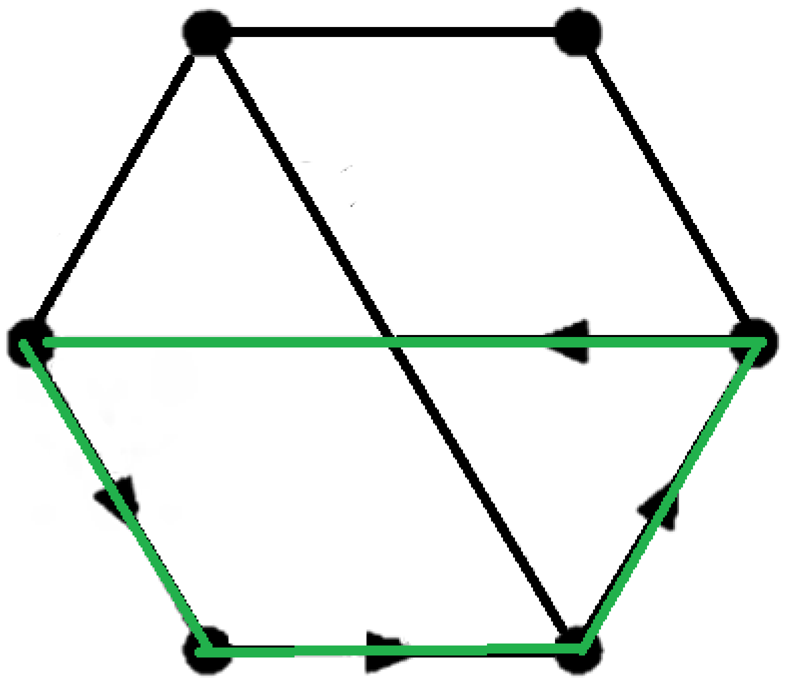




Proof

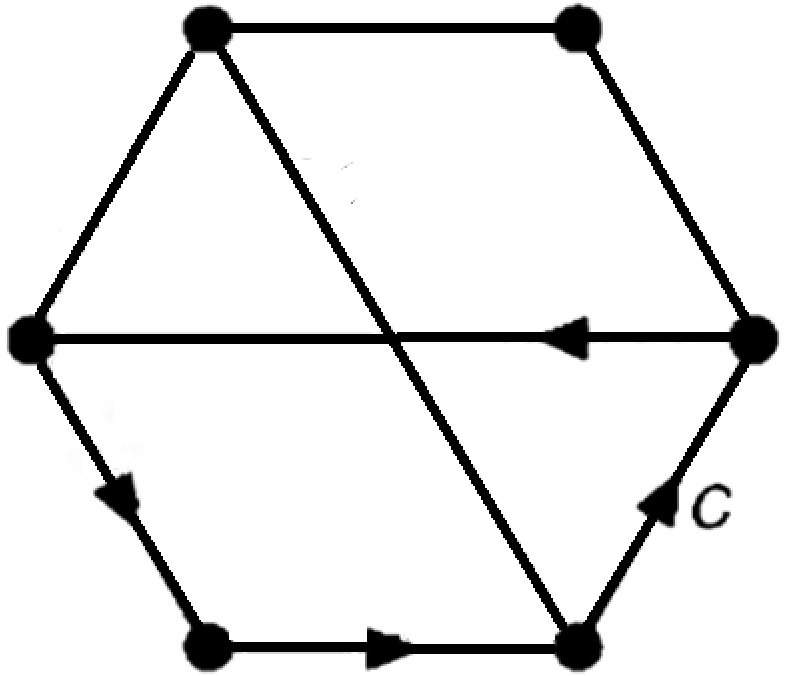
- The necessity of a condition is clear.

- If there is a directed **path from $\boldsymbol{u}$ to $\boldsymbol{v}$** and another **path from $\boldsymbol{v}$ to $\boldsymbol{u}$** $\Rightarrow$ There is a cycle in the underlying graph.



Proof

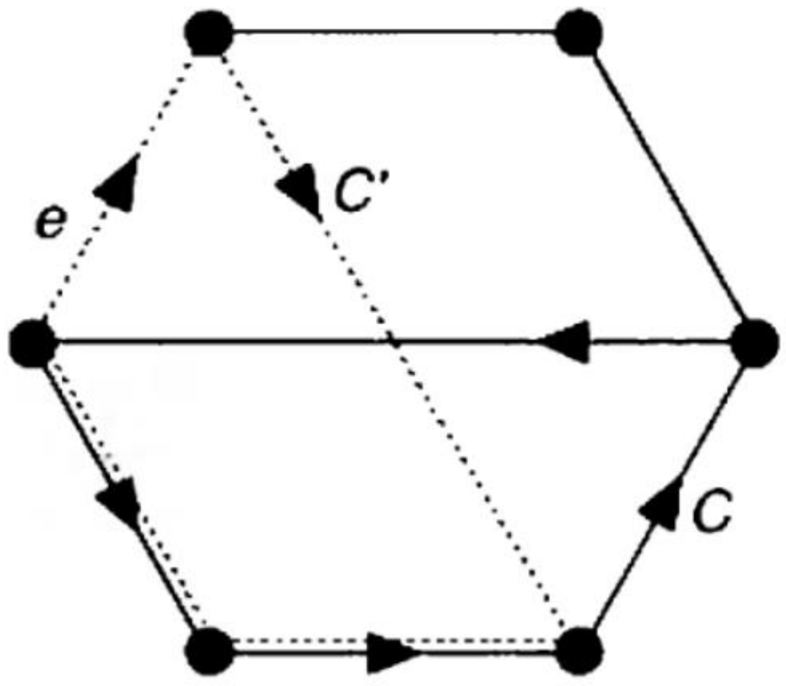
- The necessity of a condition is clear.
- To prove sufficiency, choose any cycle $\boldsymbol{C}$ and direct its edges cyclically.
- If each edge of $\boldsymbol{G}$ is in $\boldsymbol{C}$, the proof is complete.



Proof

- If not, choose any edge e adjacent to an edge of $\boldsymbol{C}$ but not in $\boldsymbol{C}$.
- By hypothesis, $\boldsymbol{e}$ is in some cycle $\boldsymbol{C}$ whose edges we may direct cyclically, **except those already directed.**
- Proceed this way, directing at least one new edge at each stage until all edges are directed
- The digraph must remain strongly connected at each stage

- Thus, the sufficiency of the condition is proved.



Reference text book