Chapter 04: Cut-Sets and Cut-Vertices

Table of Contents

[Properties of Cut-Sets 2](#_Toc138883066)

[Fundamental Cut-Sets 4](#_Toc138883067)

[Fundamental Circuits and Cut-Sets 5](#_Toc138883068)

[Edge Connectivity and Vertex Connectivity 6](#_Toc138883069)

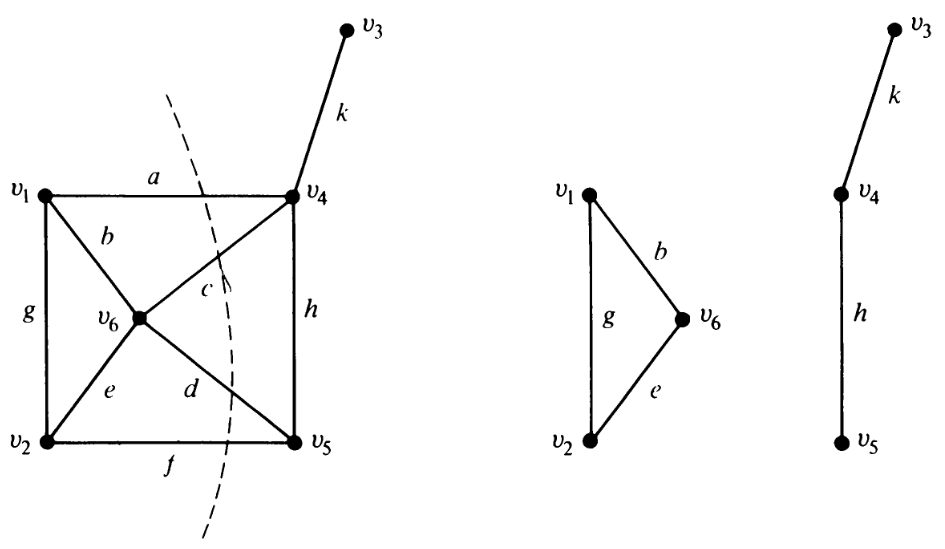
[Flow 9](#_Toc138883070)

[1-Isomorphic Graphs 10](#_Toc138883071)

[2-Isomorphic Graphs 11](#_Toc138883072)

[Circuit Correspondence 12](#_Toc138883073)

A **cut-set** is a set of edges in a connected graph , the removal of which causes to become **disconnected**. Additionally, the set should not have any **proper subset** which can be removed to disconnect .



For the first graph above, is a cut-set. This is not necessarily the only cut-set however. is also a cut-set. However, is not a cut-set even though it would disconnect the graph, since it has a proper subset, , which is a cut-set.

As we remove one cut-set of a graph, the number of **vertices** remains unchanged. However, the number of **components** increases by 1. Thus, the **rank**, , decreases by 1.

For **trees**, every edge is a cut-set.

## Properties of Cut-Sets

Consider a **spanning tree** , which by definition connects all the vertices of a graph, and an arbitrary cut-set of the same graph. It is not possible for to be empty, since that would mean that removing from the graph does not disconnect the graph.

Theorem 4.1: Every cut-set in a connected graph must contain at least one branch from every spanning tree of .

The reverse of this is also true. Consider a set which is the minimal set of edges containing at least one branch from every spanning tree of . If we remove from , the graph will becoming disconnected. This makes a **cut-set**, and more specifically, a **minimal cut-set**.

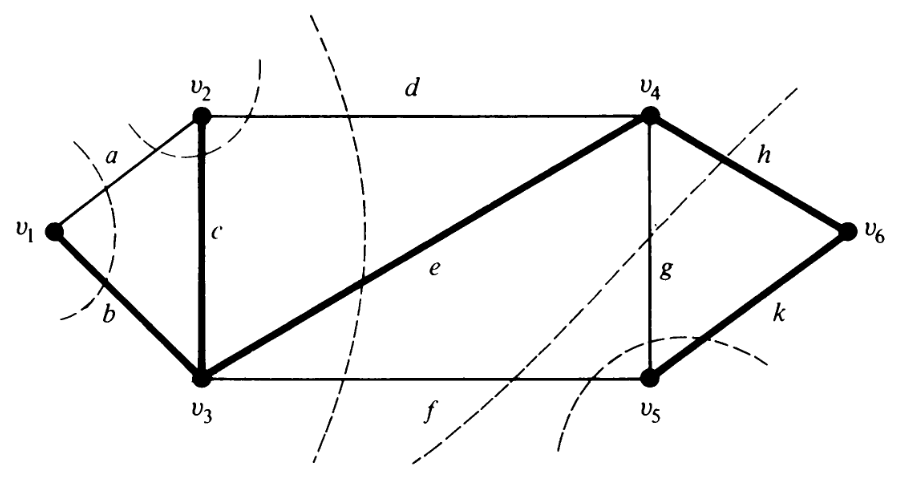
Theorem 4.2: In a connected graph , any minimal set of edges containing at least one branch of every spanning tree of is a cut-set.

Theorem 4.3: Every circuit has an even number of edges in common with any cut-set.

Consider the graph shown below which has a cut-set which partitions the graph into two mutually exclusive and disjoint subsets and . If a circuit is contained entirely within or entirely without , it has 0 edges in common with , which is an even number. If a circuit has one edge going from to , it must also have another edges going back from to due to the closed nature of a circuit. Thus, the number of edges in common with must be even.



## Fundamental Cut-Sets



For the graph above, all possible cut-sets are shown with dotted lines and the bold lines show a spanning tree. To disconnect the vertex , we must take the cut-set . This cut-set contains one branch from the spanning tree and one chord. If a cut-set contains a single branch while the rest of the edges are chords, it is called a **fundamental cut-set**.

Theorem 4.4: The ring sum of any two cut-sets is either a third cut-set or an edge disjoint union of cut-sets.

For the graph above, , which is another cut-set, while , which is the edge disjoint union of the two cut-sets and .

## Fundamental Circuits and Cut-Sets

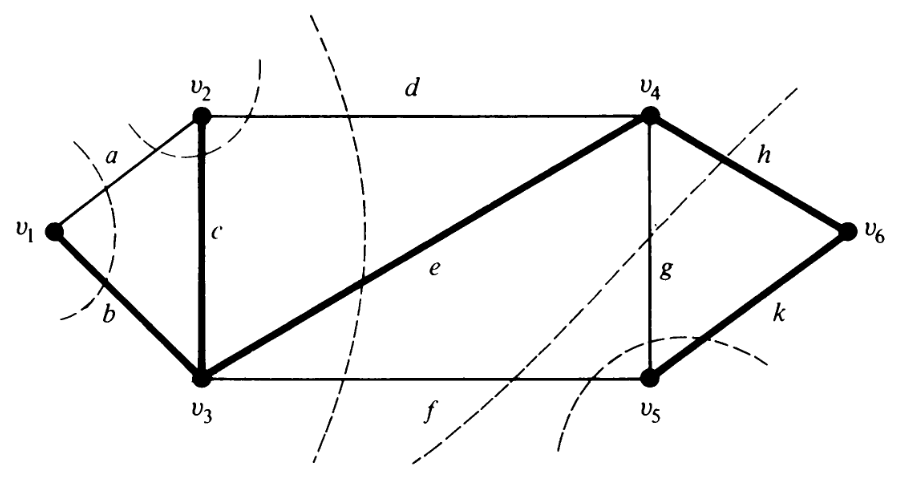
Suppose a graph has a spanning tree with the branches , , and let one of the chords be . We can create a **fundamental circuit** using the spanning tree and .

Every branch of a spanning tree has a **fundamental cut-set** associated with it. Let be a fundamental cut-set formed using and other chords, , , , . From theorem 4.3, we know that and must have an even number of edges in common. One of these edges is of course. The other must therefore be . The same argument can be made for every other fundamental cut-set we can create using each of the other branches.

In addition, it is also not possible for any cut-set to exist which contains but not any of the branches from . Otherwise, there would be only one edge in common with and , contradicting theorem 4.3.

Theorem 4.5: A chord that determines a fundamental circuit occurs in every fundamental cut-set associated with the branches of and in no other cut-set.

Example:

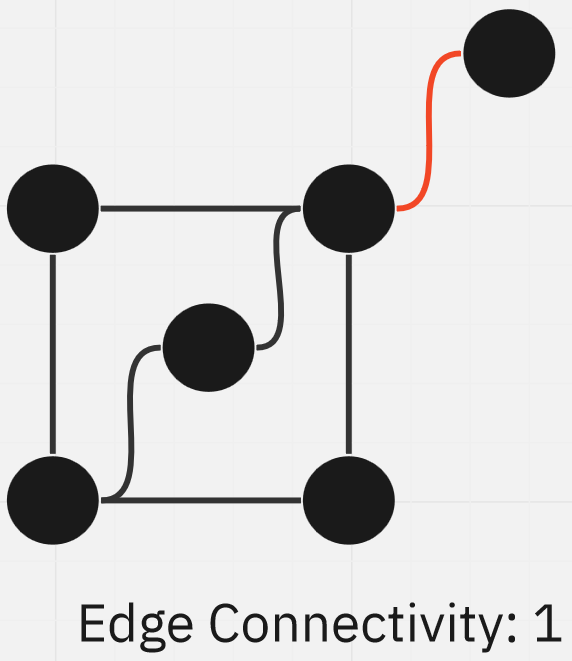
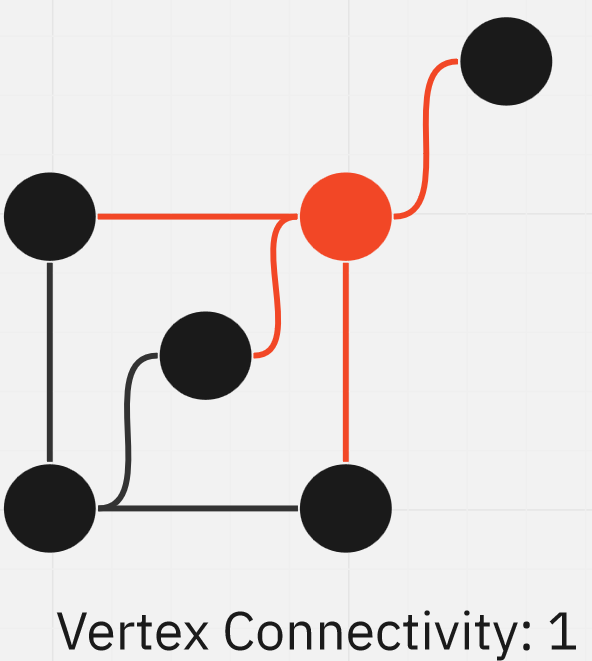


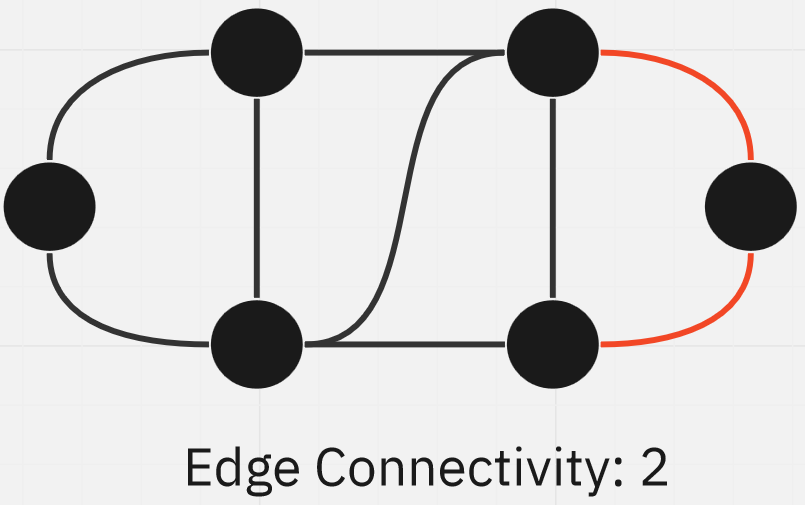
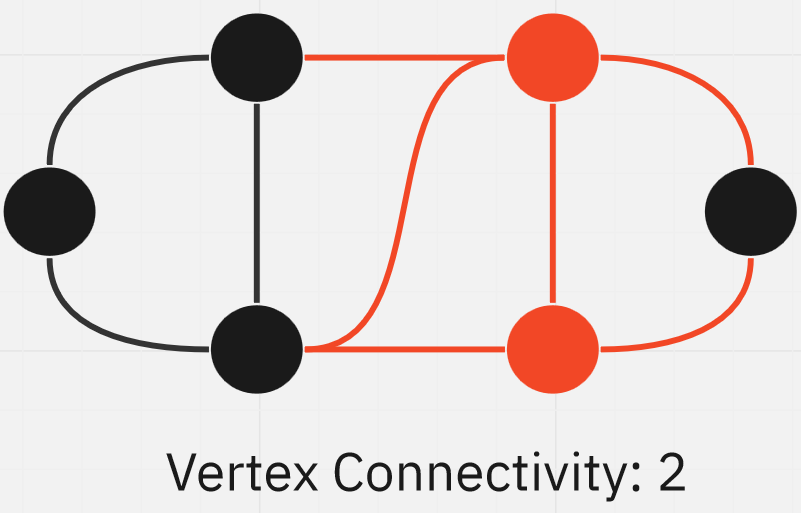
Let . The fundamental cut-sets are thus , and for the branches , and respectively. All of these cut-sets have in common.

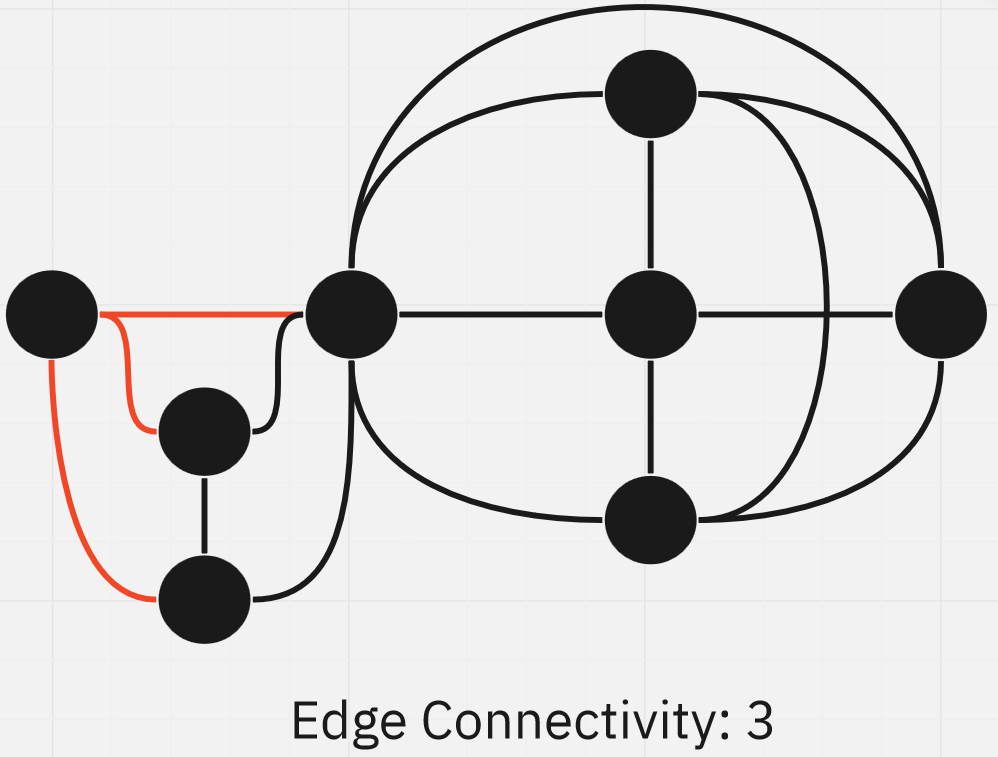
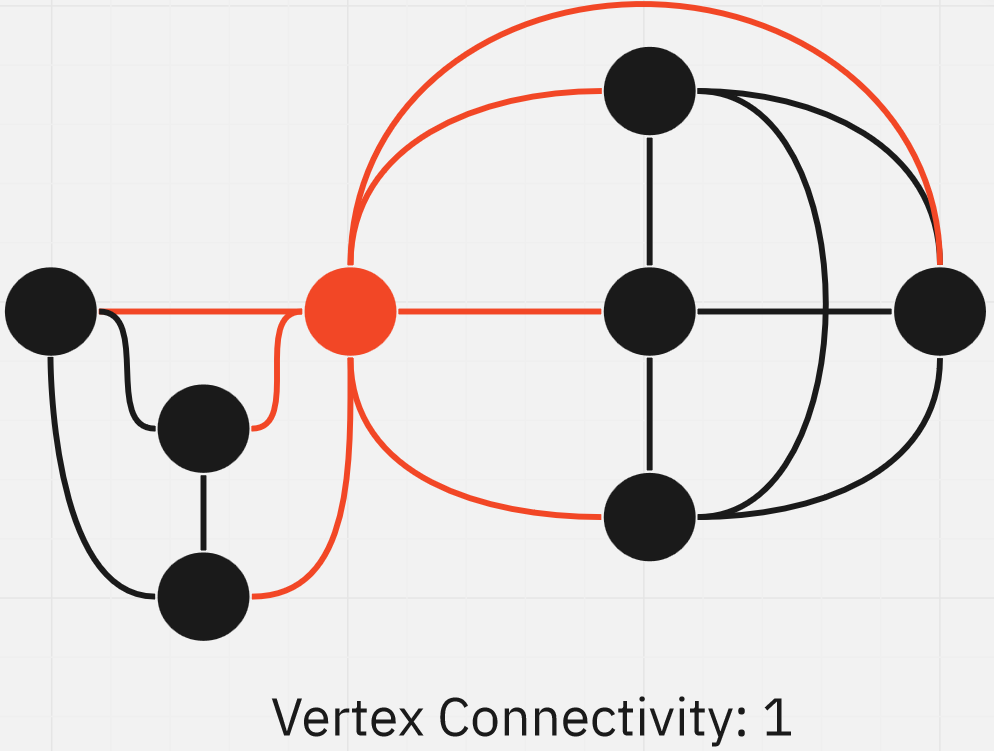
## Edge Connectivity and Vertex Connectivity

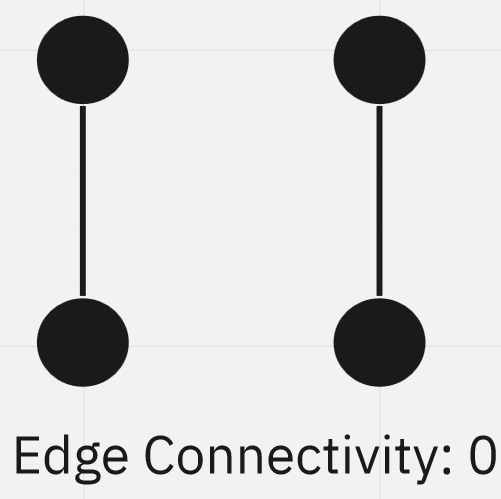
**Edge Connectivity** refers to the number of edges in the smallest cut-set, i.e., the number of edges that must be removed at the minimum to make a graph disconnected.

**Vertex Connectivity** refers to the number of vertices that must be removed at the minimum to make a graph disconnected. Remember that removing a vertex also removes any edges incident on that vertex. If the vertex connectivity of a graph is , the graph is said to be **-connected**.

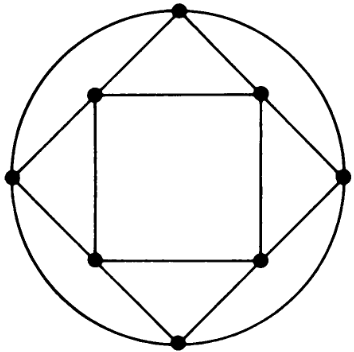


For trees, the edge connectivity and vertex connectivity are both 1. In the case of vertex connectivity, the vertex that is removed is called the **cut-vertex** or the **articulation point**. This must be a **non-pendent** vertex.

A graph that can be disconnected by removing a single vertex, such as a tree, is called a **separable graph**.

Theorem 4.7: In any graph, if there is a vertex such that all paths between two other vertices and go through , then is called a **cut-vertex**. Essentially, if we remove , then the graph is guaranteed to become disconnected.

One example of the use case for all of the information we just learnt is the construction of roads between cities. In the case of an attack on the country, if the enemy must capture the cut-vertices in order to disconnect the city. Ensuring that the number of cut-vertices is high will make this more difficult. For example, in the graph below, both the edge connectivity and the vertex connectivity is .



Theorem 4.8: Edge connectivity cannot exceed the minimum degree of any vertex.

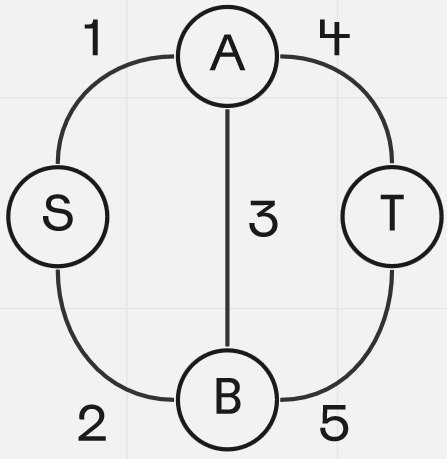
Theorem 4.9: Vertex connectivity cannot exceed the edge connectivity.

Suppose we have vertices and edges. This means that the total degree of the graph is . The best case scenario here is to **distribute the degrees**. This will ensure that no vertex is given a lower degree in order to give some other vertex a higher degree. As such, the maximum degree is , which makes the maximum vertex connectivity .

Theorem 4.10: The maximum vertex connectivity one can achieve with a graph of vertices and edges, where , is .

## Flow

**Flow** refers to the transfer of material from one point to another via a graph, such as the one shown below.



There are a few rules to calculating the flow:

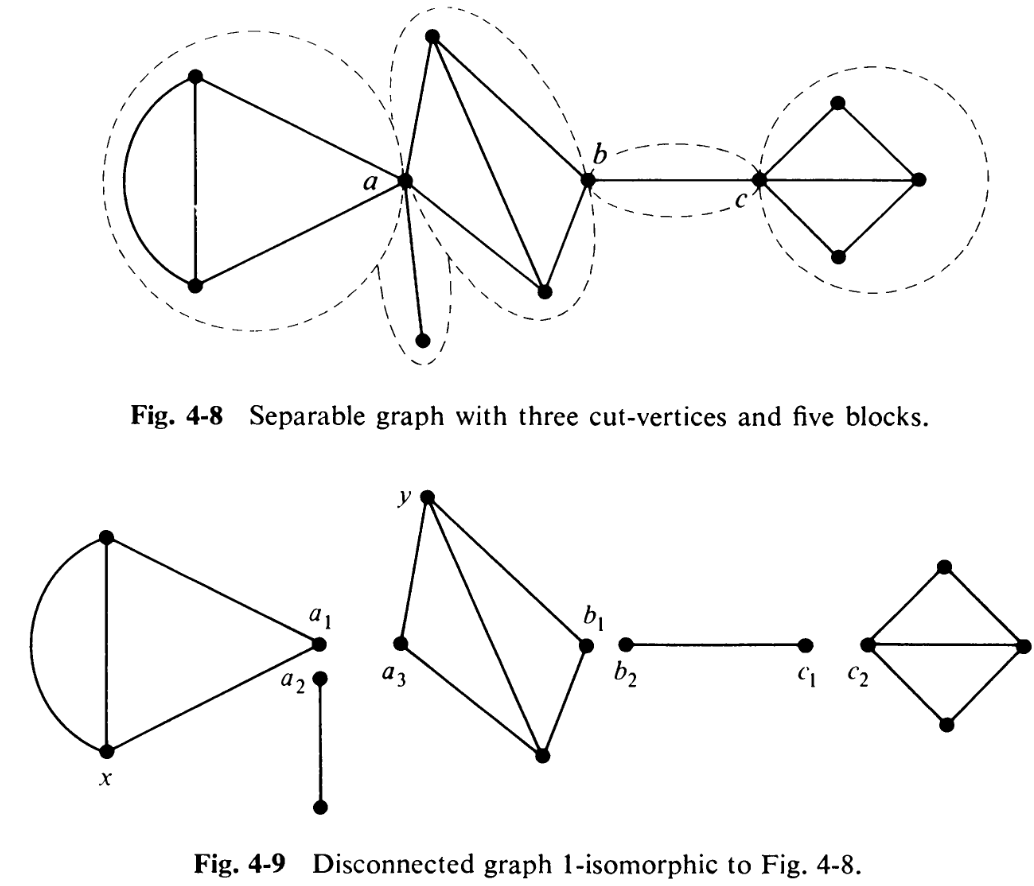
1. Apart from the source and target, the amount of material entering a node must be equal to the amount of material exiting it.
2. The maximum flow is only restricted by the capacity of the pipes, as denoted by their weights.
3. There are no leaks.

Our goal is to calculate the **maximum flow** and the **path** via which this flow occurs. More formally, for a connected graph , we must find the **cut-set** between two nodes, and . The capacity of the cut-set is the sum of the capacities of each of the edges in the cut-set.

Theorem 4.13: The maximum flow between and is equal to the minimum capacity of all cut-sets between and .

The above theorem is also called the **Min-Cut Max-Flow Theorem**.

## 1-Isomorphic Graphs



Consider the two graphs above. The first graph is a **separable** graph and the second graph is clearly derived from the first graph, but they are **not isomorphic**.

The original separable graph has a **set of non-separable subgraphs**, called **blocks**. The graph is divided into blocks based on where the **cut-vertices** are. The second graph was created by extracting each of these blocks into a **component**. Each of the components are **isomorphic** to one of the blocks. This phenomenon is called **1-Isomorphism**.

In addition to the above, all **isomorphic** graphs are also **1-isomorphic**.

Theorem 4.14: If and are two 1-isomorphic graphs, the rank of equals the rank of and the nullity of equals the nullity of .

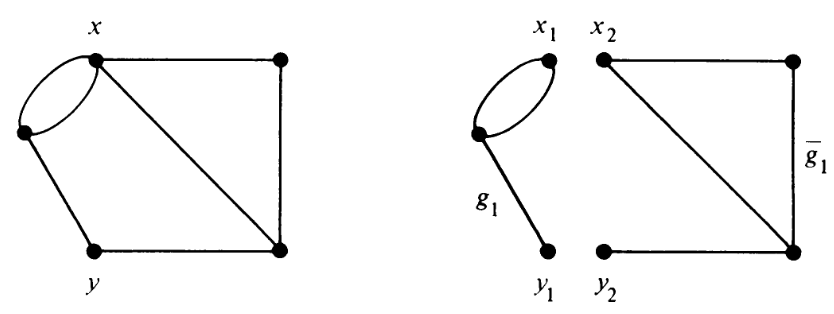
Proof:

Whenever we split into two components at one of the cut-vertices, the total number of vertices increases by and the total number of components also increases by . Since the rank of a graph is the difference between the number of vertices and the number of components, it remains unchanged.

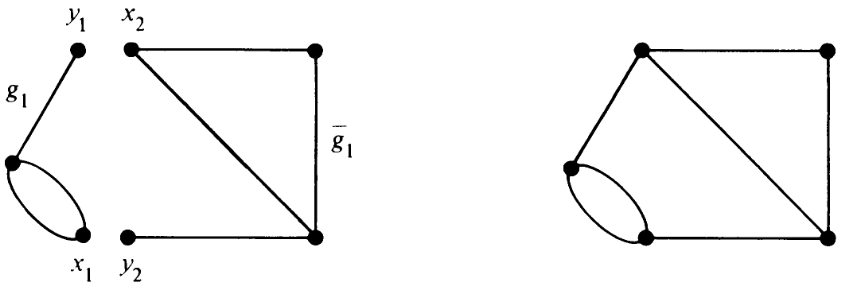
The splitting process does not modify the number of edges. Nullity is the difference between the number of edges and the rank of a graph, which thus remains unchanged as well.

## 2-Isomorphic Graphs

Consider a graph that is 2-connected, i.e., it has a vertex connectivity of 2. For this graph, let the cut-vertices be and . Thus, we can split the graph into two components, as shown below.



The concept of **2-isomorphism** suggests that we now connect to and to , fusing the vertices as shown below.



2-isomorphic graphs are also equal in rank and nullity.

## Circuit Correspondence

Two graphs are said to have a **circuit correspondence** if for any circuit formed in , the same circuit is formed in using the same edges.

**Isomorphic graphs** obviously have circuit correspondence.

For **1-isomorphic graphs**, the original graph is separable, meaning the circuits are confined to the blocks. Since the blocks do not change, circuit correspondence remains intact.

A similar argument can be made for **2-isomorphic graphs**, where the edges surrounding the circuits do not change, even though the order in which those edges appear can change.