# Edexcel A Level Further Maths: Decision Maths 1

Your notes

## Graphs

### Contents

* Graph Theory
* Eulerian & semi-Eulerian Graphs
* Planarity Algorithm

# Graph Theory

Your notes

## Introduction to Graph Theory

### What is graph theory?

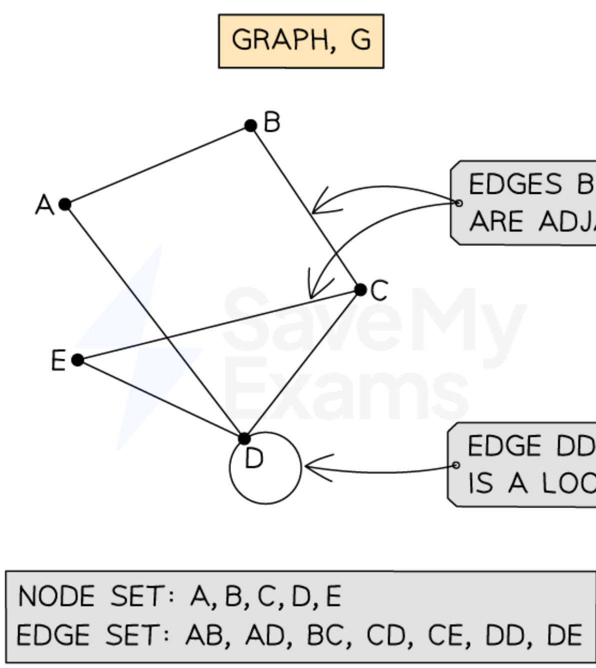
- **Graph theory** is the study of **graphs**, which are mathematical structures used to represent **objects** and the **connections** between them
- They can be used in modelling many **real-life applications**, e.g. electrical circuits, flight paths, maps etc

## Edges & Vertices

Your notes

### What are the different parts of a graph?

- A **graph** is made up of a number of points (**vertices** or **nodes**) that are connected by lines (**edges** or **arcs**)
- A **vertex** (**node**) can represent an object, a place or a person
  - **Adjacent vertices** are connected by an **edge**
  - Vertices are usually labelled with a **letter**, e.g. *A, B, C*...
    - The list of vertices in a graph is sometimes called the **vertex set**
- An **edge** (**arc**) forms a connection between **two vertices**
  - Edges are described by the **nodes they connect**, e.g. *AB, AC, BE*...
    - The list of edges in a graph is sometimes called the **edge set**
  - **Adjacent edges** share a common vertex
  - There may be **multiple edges** connecting two vertices
  - An edge that starts and ends at the same vertex is called a **loop**
- Typically a graph will be drawn such that **edges do not overlap**
  - Note that if a graph is drawn with **overlapping edges**, the edges are **not connected** at points where they overlap
  - Edges are only connected at **vertices**



# Properties of Graphs

Your notes

## What are the properties of graphs?

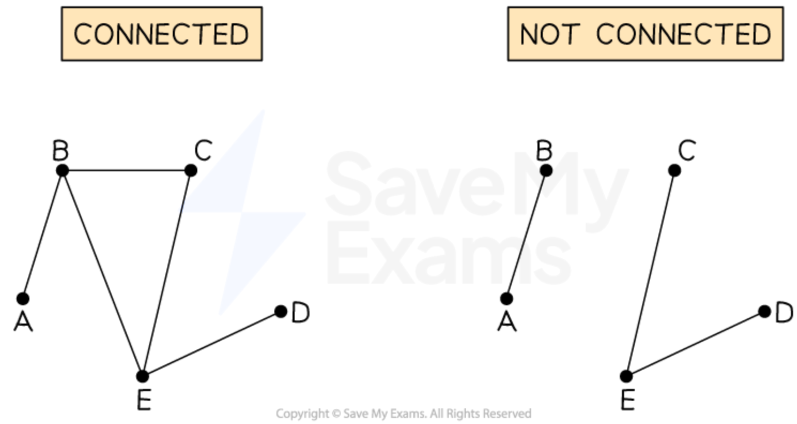
- The **edges** of a graph may be assigned a numerical value
  - This value is called the **weight** (of an edge)
  - Weight is often a measure such as distance, time or money
- A **walk** is a **finite series of edges** in a graph that starts at one vertex and moves from vertex to vertex
  - The (total) **weight** of a **walk** is the **sum of the weights** of the **edges** that it consists of
- A **path** is a **walk** where no **vertex** is visited more than once
- A **trail** is a **walk** where no **edge** is visited more than once
  - Every path is also a trail but not every trail is a path
- A **cycle** (or **circuit**) is a path that starts and finishes at the **same vertex**
  - It is also known as a **closed path**
- A **tour** is a **walk** that visits **every vertex** and **returns** to its **start vertex**

# Types of Graph

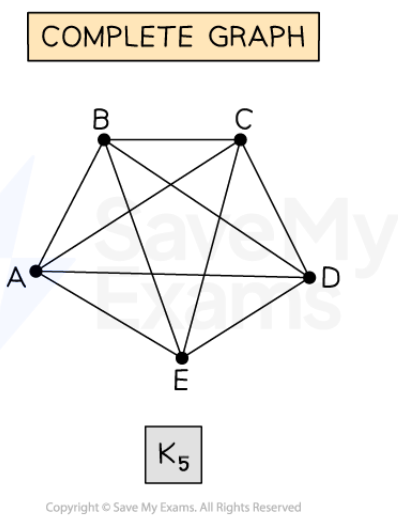
Your notes

## What are the types of graph?

- A **connected graph** is a graph in which all of its vertices are connected to each other
  - Two vertices are connected if there is a **path** between them
    - There does not need to be an edge connecting the pair of vertices

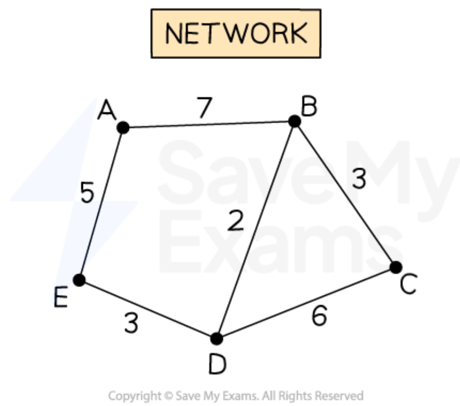



- A **complete graph** is a graph in which each vertex is connected by an edge to each of the other vertices
  - A complete graph with $n$ vertices is labelled $K_n$

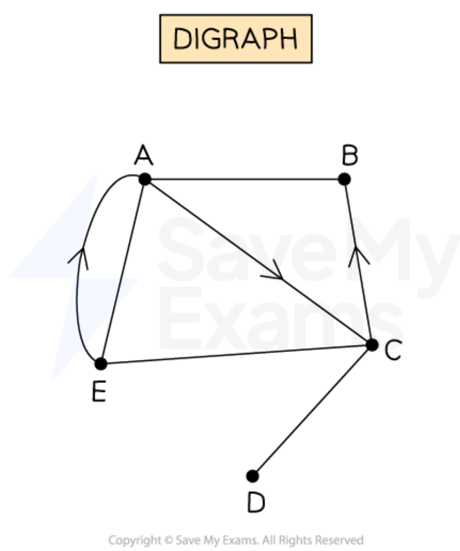



- If the edges of a graph have a **weight,** the graph is known as a **weighted graph** (or **network**)
  - Networks are **not** usually **drawn to scale**

Your notes

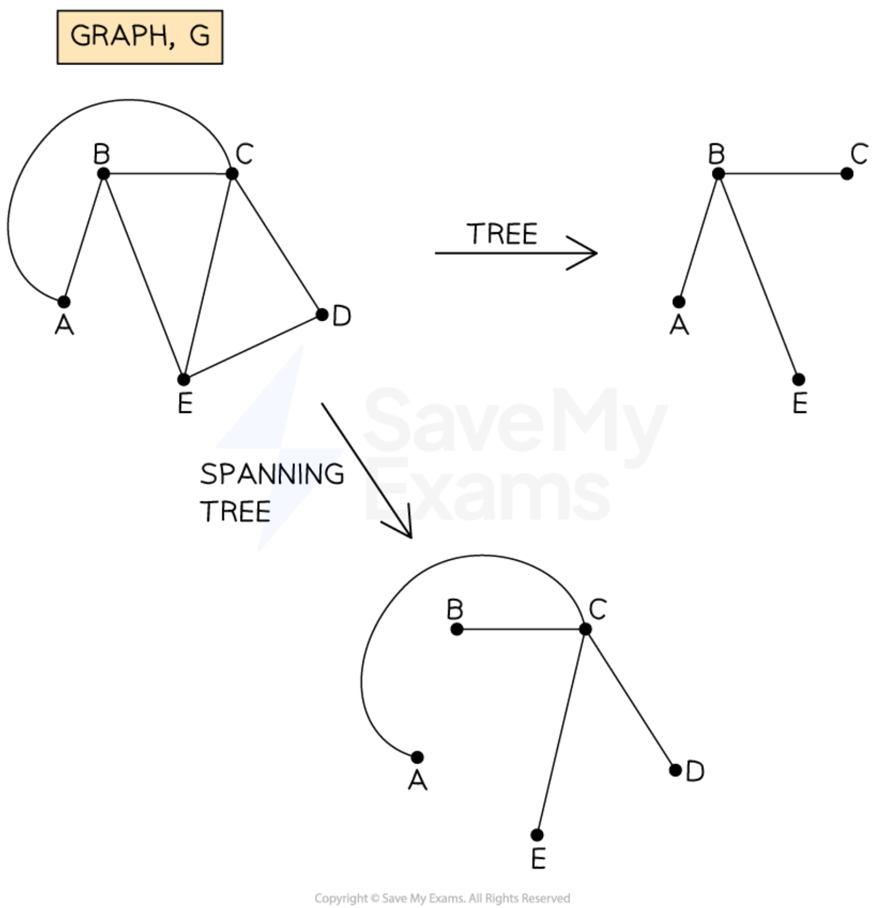



- If the edges of a graph are assigned a direction, they are known as **directed edges** and the graph is known as a **digraph**
  - the directed edges of a graph can only be **traversed in the direction indicated**

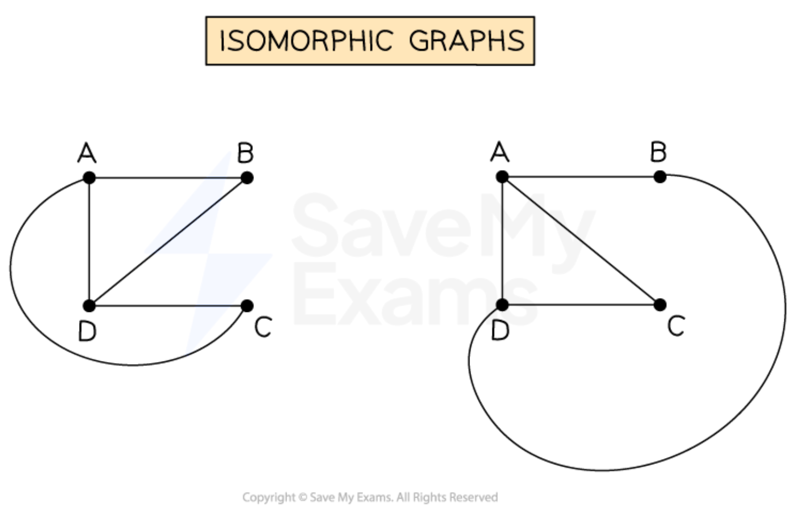



- A **simple graph** is **undirected** and **unweighted** and contains **no loops** or **multiple edges**
- Given a graph G, a **subgraph** will only contain edges and vertices that appear in G
- A **tree** is a connected graph that does not contain any cycles
- A **spanning tree** is a subgraph, which is also a tree, of a graph G that contains all the vertices from G

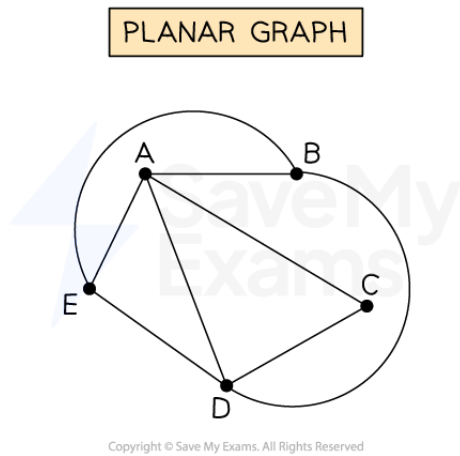
Your notes




- An **isomorphic graph** is a graph that shows the same information (number of vertices and valency of vertices) but is drawn in a different way



- A **planar graph** is a graph where no two edges meet each other except for at a vertex




Your notes

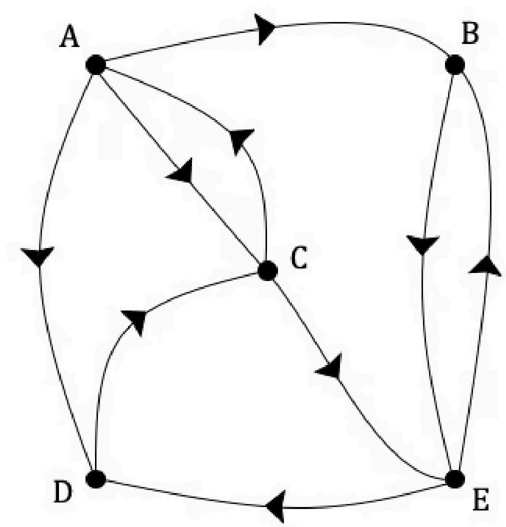
## Examiner Tip

- There are a lot of specific terms involved in graph theory and you are often asked to describe these terms in an exam, so make sure you learn the definitions - note that in many cases there are two terms that describe the same thing!
- Make sure that any graphs you draw are big and clear so they are easy for the examiner to read

Your notes

## Worked example

The graph G shown below is a connected, unweighted, directed graph with 5 vertices.




a) Write down a cycle, starting at vertex A and visiting exactly 2 other vertices.

A cycle starts and ends at the same vertex, with no other vertex visited more than once
Two other vertices (from the 4 possibilities) need to be included
The directed edges mean, in this case, there is only one such cycle

**ADCA**

b) Write down a path, longer than 1 edge, that starts at vertex A and ends at vertex B.

A path is a walk that does not repeat any nodes.
You must only traverse edges in the directions indicated.

**ADCEB**

A path does not have to visit every node (so ACEB is also acceptable)

Your notes

# Eulerian & semi-Eulerian Graphs

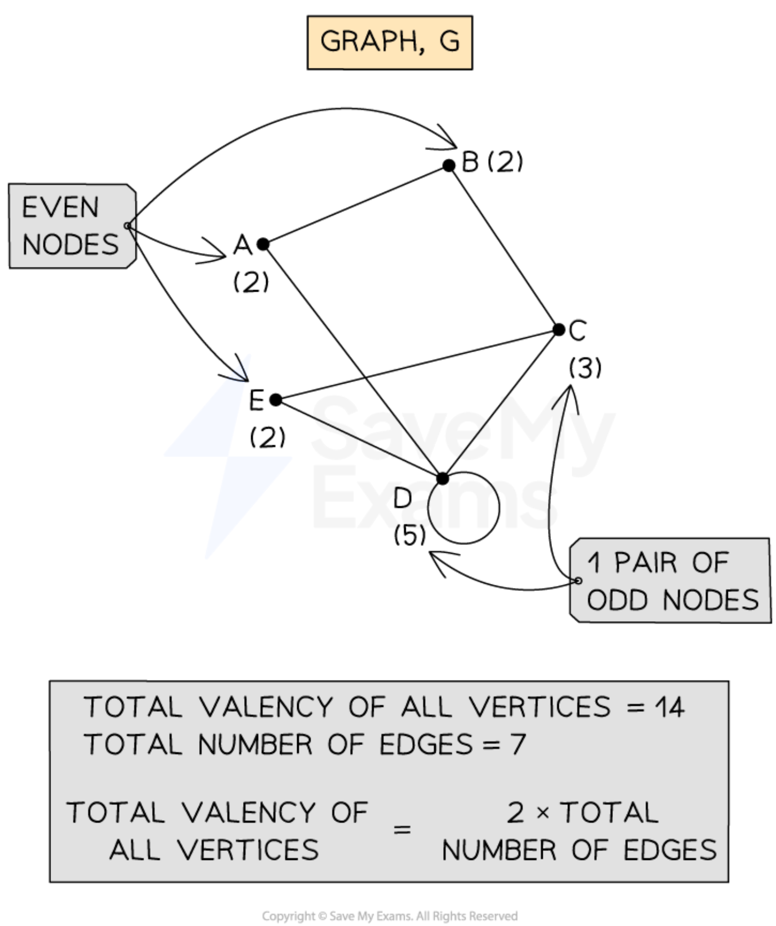
## Order (Degree) of a Node

### What is meant by the order (degree) of a node?

- The **degree** or **valency** of a vertex can be defined by **how many edges** are **incident** (connected) to it
- A vertex can be described as being **odd** or **even:**
  - It has **odd degree** if there are an odd number of edges connected to it
  - It has **even degree** if there are an even number of edges connected to it

### What is Euler's handshaking lemma?

- **Euler's handshaking lemma** states that for any **undirected** graph:
  - the **sum of the degrees** of the **vertices** is **equal** to **two lots of the number of edges**
  - the **number** of **odd vertices** must be **even** (or zero)



# Eulerian & Semi-Eulerian Graphs

## What are Eulerian cycles and trails?

- An **Eulerian cycle** starts and ends at the **same vertex** and traverses **every edge** in a graph **exactly once**
  - Unlike a true cycle it may visit a **vertex more than once**
  - An Eulerian cycle is also known as an **Eulerian circuit**
- An **Eulerian trail** traverses **every edge exactly once** but starts and ends at **different vertices**
  - Again, vertices may be visited more than once

## What are Eulerian and semi-Eulerian graphs?

- An **Eulerian graph** is a graph that contains an Eulerian cycle
  - **Every vertex** in an Eulerian graph has an **even valency**
- A **semi-Eulerian graph** is a graph that contains an **Eulerian trail**
  - **Exactly one pair** of vertices in the graph will have **odd** valencies
    - These odd vertices will be the **start** and **finish** points of any **Eulerian trail**
- Eulerian graphs can be used to solve many **practical problems** where the edges should not be traversed more than once
  - A common problem is the **Chinese Postman problem**

### Examiner Tip

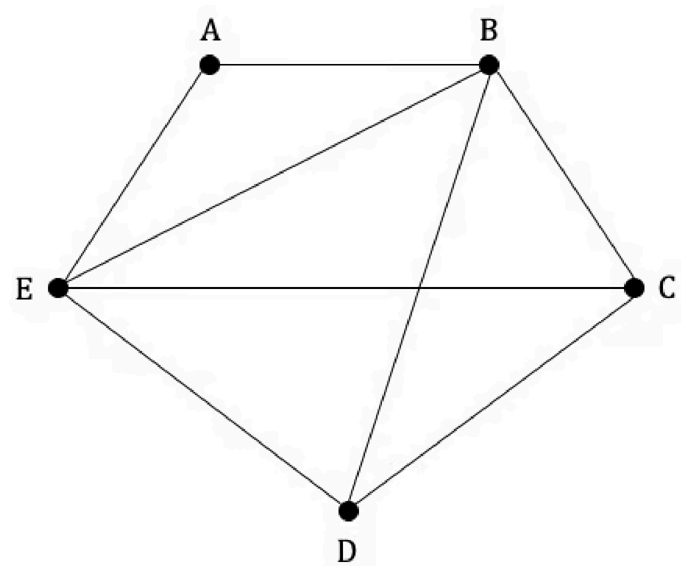
- If you can draw a graph without taking your pen off the paper and without going over any edge more than once then you have an Eulerian or semi-Eulerian graph!

Your notes

Your notes

## Worked example

Let G be the graph shown below.




a) Show that G is a semi-Eulerian graph.

Look at the degree of each vertex.

A: 2
B: 4
C: 3
D: 3
E: 4

**G is a semi-Eulerian graph because it has exactly one pair of odd vertices, C and D**

b) Write down an Eulerian trail for G.

An Eulerian trail must start and end at C/D
There are several possible Eulerian trails, one solution is

**DEABECDBC**

# Planarity Algorithm

Your notes

## Introduction to the Planarity Algorithm

### What is the planarity algorithm?

- A **planar graph** is one that can be drawn in a plane in such a way that **no two edges connect** except for at a vertex
- You can determine whether or not a graph is planar by applying the **planarity algorithm**
- The planarity algorithm can be applied to graphs that contain a **Hamiltonian cycle**

# Identifying a Hamiltonian Cycle

## What are Hamiltonian paths and cycles?

- A **Hamiltonian path** is a path in which each **vertex** in a graph is visited **exactly once**
- A **Hamiltonian cycle** is a cycle which visits each **vertex** in a graph **exactly once** and returns to its **start vertex**
- If a graph contains a **Hamiltonian cycle** then it is known as a **Hamiltonian graph**
- A graph is **semi-Hamiltonian** if it contains a **Hamiltonian path** but not a **Hamiltonian cycle**
- The only way to show that a graph is **Hamiltonian** or **semi-Hamiltonian** is to identify a **Hamiltonian cycle** or **Hamiltonian path**

### Examiner Tip

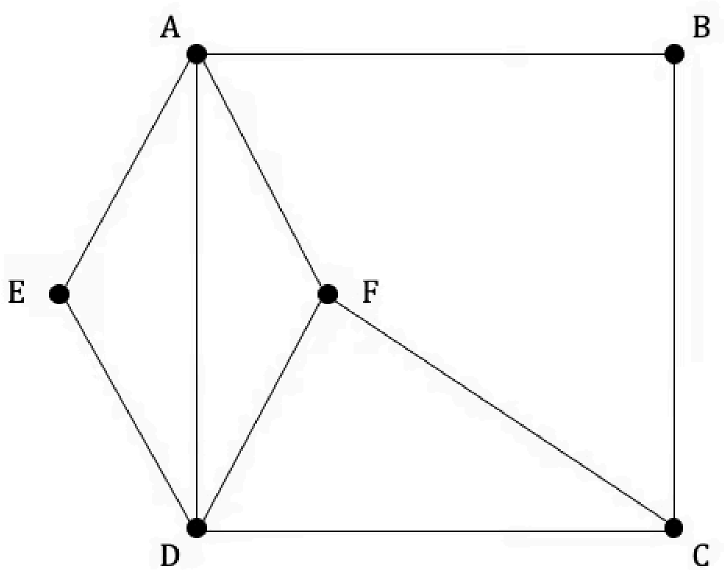
- If you are given an adjacency matrix and are asked to find a Hamiltonian cycle, make sure that you sketch out the graph first

Your notes

Your notes

## Worked example

Let G be the graph shown below.




Show that G is a Hamiltonian graph.

To show that the graph is Hamiltonian, identify a Hamiltonian cycle.

**ABCFDEA**

There is more than one possible correct Hamiltonian cycle in this graph

Your notes

# Applying the Planarity Algorithm

## How is the planarity algorithm applied to a graph?

- STEP 1
  Draw a polygon (roughly regular) with the same number of vertices as the original graph
  - Identify a **Hamiltonian cycle** in the graph
  - **Re-label** the vertices in the **order of the Hamiltonian cycle**

- STEP 2
  Add **all of the other edges** in the original graph to the new graph **inside the polygon**
  - Make a **list** of all of these added edges

- STEP 3
  Choose an edge **inside the polygon** that has **not yet been labelled** and **label it (I)** *(Inside)*
  If **all edges** (inside the polygon) **now have** a **label,** the graph is **planar**

- STEP 4
  Identify any **edges inside the polygon** that **intersect** the edge(s) that has **just been labelled**
  - If there are no such edges, return to **STEP 3**
  - If any of the edges that intersect the 'just labelled' edge **also intersect each other**, then the graph is **non-planar**
  - If the edges that intersect the 'just labelled' edge **do not intersect with each other**, give them each the **opposite label** to the 'just labelled' edge(s)
    - (I) and (O) (*Outside*) are opposite labels
  - If **all edges** (inside the polygon) **now have** a **label,** the graph is **planar**
  - If **any edge** (inside the polygon) remains **unlabelled**, return to the **beginning** of **STEP 4**

- If the algorithm has determined that the graph **is planar,** then you can draw it with all of the edges labelled (**I**) on the **inside of the polygon** and all of the edges labelled (**O**) on the **outside of the polygon**

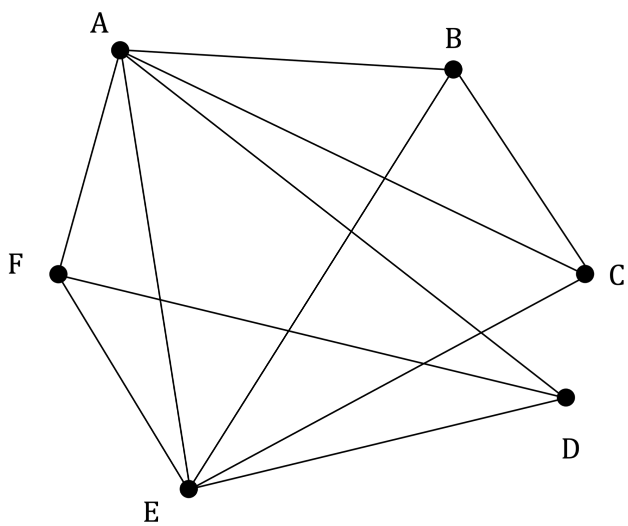
### Examiner Tip

Remember to draw your diagrams in pencil so that it's easy to erase any errors!

Your notes

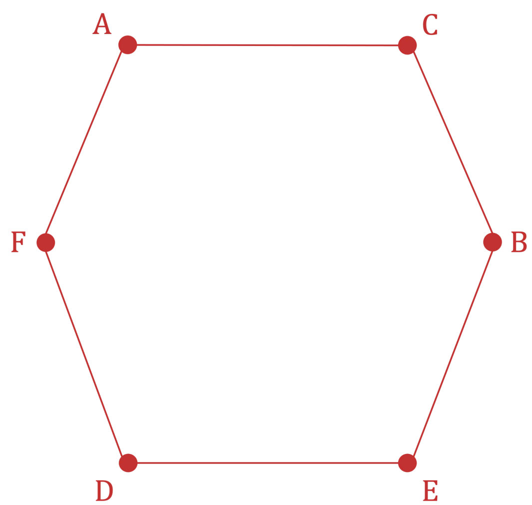
## Worked example

Show that the graph below is a planar graph by applying the planarity algorithm and draw it so that no two edges intersect.

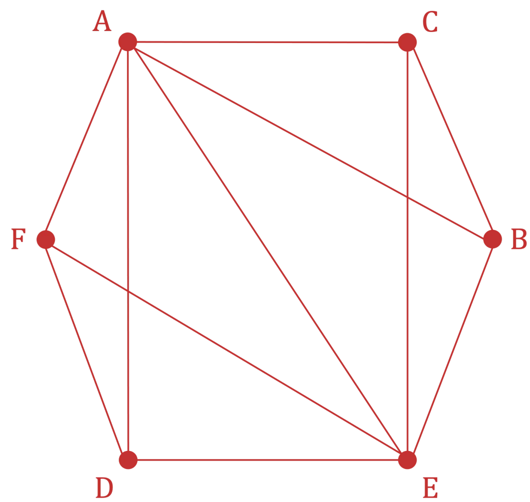



- **STEP 1**
  There are 6 vertices so re-draw these 6 vertices connected with edges to form a hexagon
  Find a Hamiltonian cycle in the original graph and label the vertices of the polygon in the order of the cycle

Hamiltonian cycle: ACBEDFA

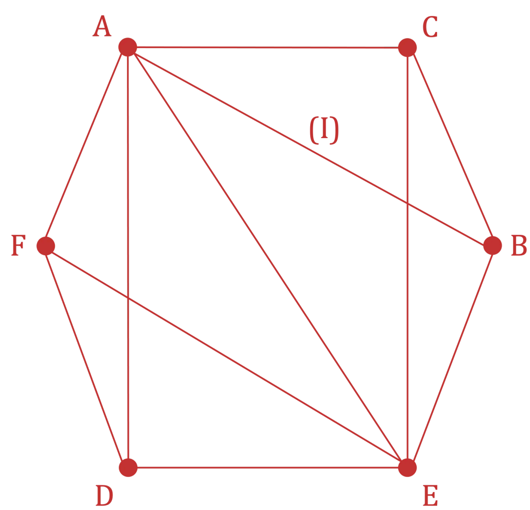


- **STEP 2**
  Draw all of the remaining edges from the original graph on the inside of the polygon and make a list of these edges




Inside edges: AB, AE, AD, CE, EF

- **STEP 3**
  Label the first edge in that list, AB, (I)

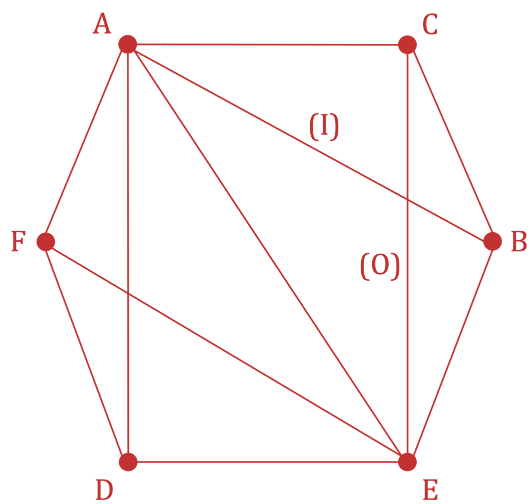



Inside edges: AB (I), AE, AD, CE, EF

- **STEP 4**
  Edge CE is the only edge that intersects AB, so label it with the opposite label, (O)

Your notes

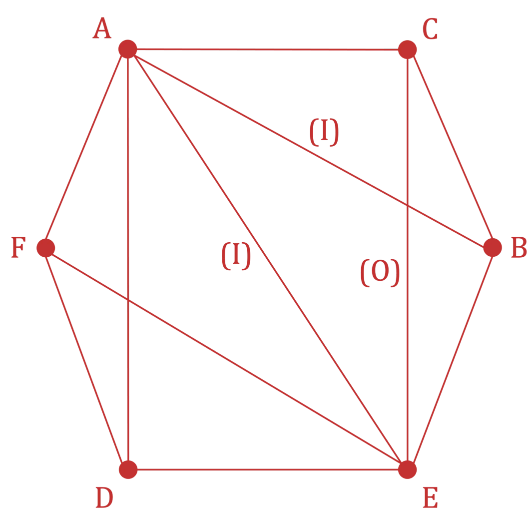
Your notes




Inside edges: AB (I), AE, AD, CE (O), EF

There are no other unlabelled edges that intersect with edge CE, so return to STEP 3

- **STEP 3**
  Choose the next unlabelled edge from the list of inside edges, AE, and label it (I)

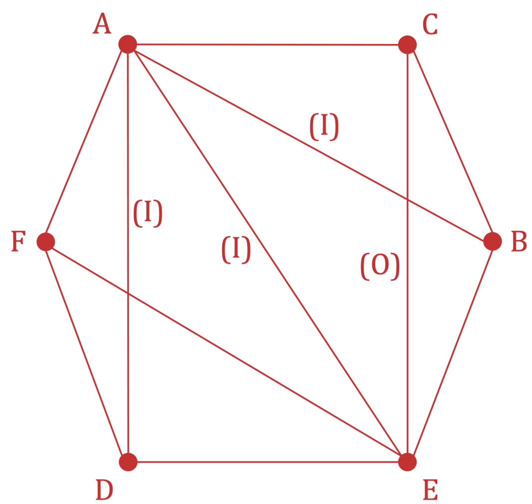



Inside edges: AB (I), AE (I), AD, CE (O), EF

- **STEP 4**
  There are no unlabelled edges that intersect edge AE, so return to STEP 3

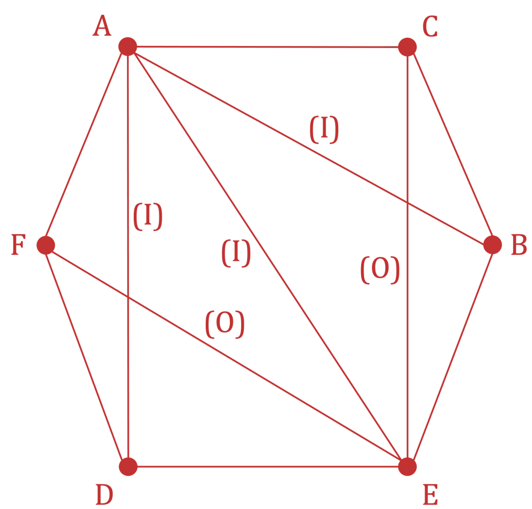
Your notes

- **STEP 3**
  Choose the next unlabelled edges from the list of inside edges, AD, and label it (I)



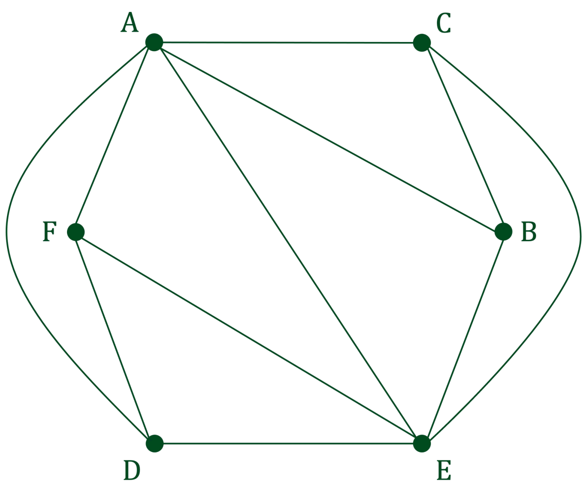

Inside edges: AB (I), AE (I), AD (I), CE (O), EF

- **STEP 4**
  Edge EF is the only edge that intersects AD, so label it with the opposite label, (O)




Inside edges: AB (I), AE (I), AD (I), CE (O), EF (O)

All edges are now labelled so the graph is planar
Re-draw the graph with edges labelled (I) inside the polygon and the edges labelled (O) outside the polygon




Your notes