# Edexcel A Level Further Maths: Decision Maths 1

Your notes

## Minimum Spanning Trees

### Contents

## Networks & Matrices

Your notes

## Introduction to Networks & Matrices

### What is a network?

- A **network** is a **weighted graph**
  - A **network** is used to model real-world situations
  - The **weight** of an arc usually represents a **measure** such as **distance** or **time**

### What is a matrix?

- A **matrix** can be used to **represent a graph** or **network** in the form of a **table**
- The **elements** within the matrix give information about the **connections** between the **different vertices**

Your notes

## Networks & Matrices

### What is an adjacency matrix?

- An **adjacency matrix** is a **square** matrix where all of the vertices in the graph are listed as the headings for both the rows and columns
- An adjacency matrix can be used to show the **number of direct connections** between two vertices
- An entry of **0** in the matrix means that there is **no direct connection** between that pair of vertices
- In a **simple** graph the only entries are either 0 or 1
- A **loop** is indicated in an adjacency matrix with a value in the **leading diagonal** (the line from top left to bottom right)
    - In an **undirected matrix** the value in the leading diagonal will be **2** because you can use the loop to travel out of and into the vertex in two different directions
    - In a **directed matrix**, if the loop has been given a direction, the value in the leading diagonal will be **1** as you can only travel along the loop out of and back into the vertex in one direction
    - For a graph with **no loops** every entry in the leading diagonal will be **0**
- An **undirected graph** will be **symmetrical** in the leading diagonal

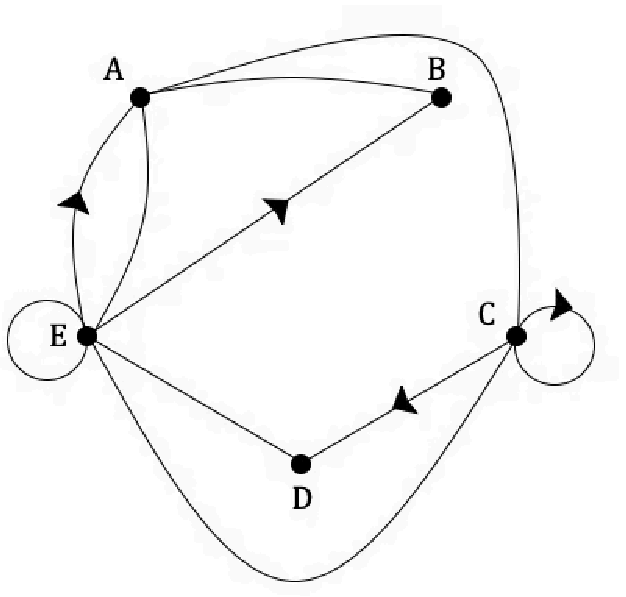
### What is a distance matrix?

- A **distance matrix** is different to an adjacency matrix as the **value** in each cell is the **weight** of the edge connecting that pair of vertices
    - Weight could be cost, distance, time etc.
- An **empty cell** can be used to indicate that there is **no connection** between a pair of vertices
- A directed network is **not** symmetrical along the leading diagonal (the line from top left to bottom right)
- When drawing a network from its distance matrix be careful when labelling the edges
    - For an **un-directed graph** the two cells between a specific pair of vertices will be the same so connect the vertices with **one edge** labelled with the relevant weight
    - For a **directed graph** if the two cells between a specific pair of vertices have different values draw **two lines** between the vertices and label each with the correct weight and direction
- A distance matrix can be used to work out the weight of different **walks** in the network

Your notes

## Worked example

a) Write down the adjacency matrix for the graph shown below.



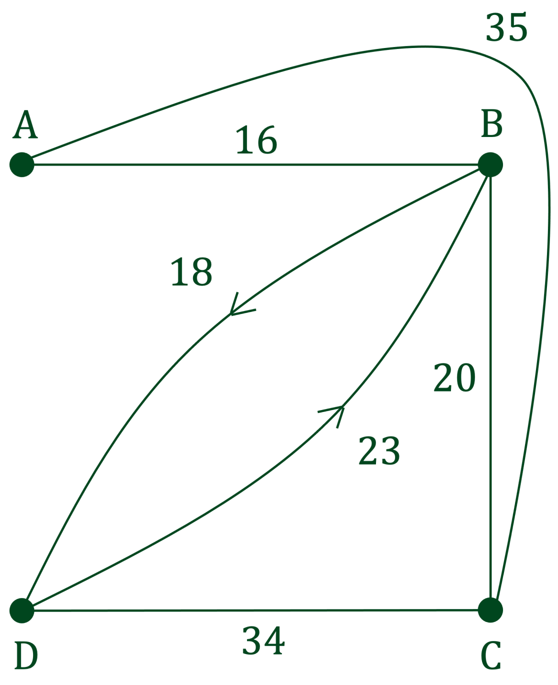
$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 1 & 0 & 1 \\ B & 1 & 0 & 0 & 0 & 0 \\ C & 1 & 0 & 1 & 1 & 1 \\ D & 0 & 0 & 0 & 0 & 1 \\ E & 2 & 1 & 1 & 1 & 2 \end{array}$$

b) The table below shows the time taken in minutes to travel by car between 4 different towns.

| | A | B | C | D |
|---|---|---|---|---|
| A | | 16 | 35 | |
| B | 16 | | 20 | 18 |
| C | 35 | 20 | | 34 |
| D | | 23 | 34 | |

Draw the network described by this distance matrix.

Your notes



## Kruskal's Algorithm

## Kruskal's Algorithm

### What is Kruskal's algorithm?

- In a situation that can be modelled by a **network**, **Kruskal's algorithm** is a mathematical tool that can be used to connect all of the vertices in a way that **reduces** costs, materials or time by finding the minimum spanning tree (MST).

### Why do we use Kruskal's Algorithm?

- **Kruskal's algorithm** is a series of steps that when followed will produce the **minimum spanning tree** for a **network**
- Finding the minimum spanning tree is useful in a lot of practical applications as it connects **all of the vertices** in the **most efficient way** possible
- The **number of edges** in a minimum spanning tree will always be **one less** than the **number of vertices** in the graph
- A minimum spanning tree **cannot** contain any cycles.

### What are the steps of Kruskal's Algorithm?

- **STEP 1**
  Sort the **edges** in terms of **increasing weight**

- **STEP 2**
  Select the **edge of least weight** (if there is more than one edge of the same weight, either may be used)

- **STEP 3**
  Select the **next edge of least weight** that has not already been chosen and **add it to your tree** provided that it does not make a cycle with any of the previously selected edges

- **STEP 4**
  Repeat STEP 3 until **all of the vertices** in the graph are **connected**

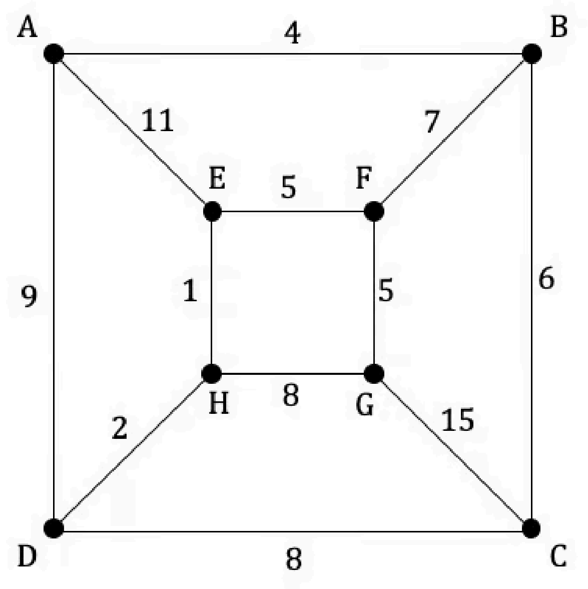
#### Examiner Tip

- Draw out the edges included in the minimum spanning tree as you go along as it will make it easier to spot potential cycles
- When using any of the algorithms for finding the minimum spanning tree, make sure that you state the **order** in which the edges are selected to get full marks for working!
- In Kruskal's algorithm, the edges are **sorted** into **increasing weight** first
  - there are algorithms for that too !

Your notes

## Worked example

Consider the network below.



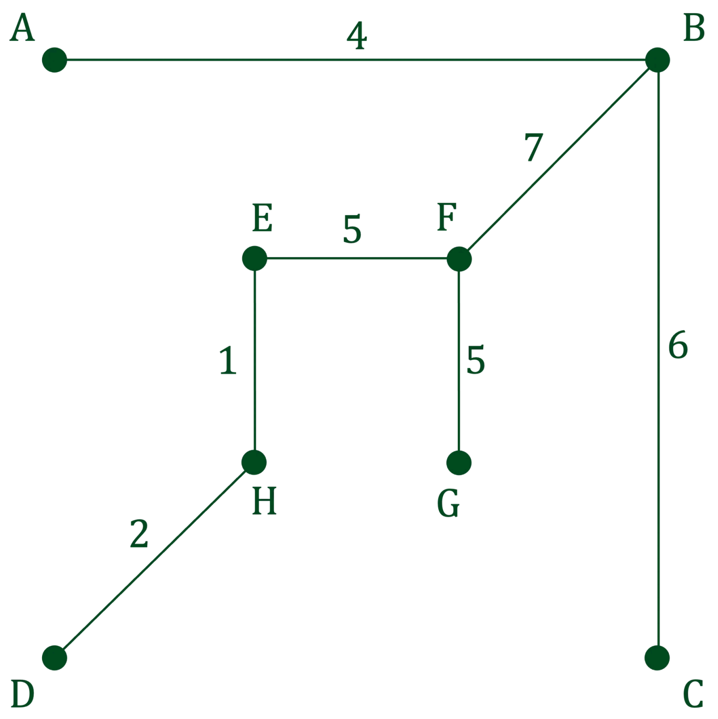
a) Use Kruskal's algorithm to find the minimum spanning tree. Show each step of the algorithm clearly.

- **STEP 1**
  List the edges in order of increasing weight

EH (1)
DH (2)
AB (4)
EF (5)
FG (5)
BC (6)
BF (7)
CD (8)
GH (8)
AD (9)
AE (11)
CG (15)

- **STEP 2**
  Add the edge of least weight (EH)

- **STEP 3**
  Add the next edge of least weight (DH)
- **STEP 4**
  Repeat Step 3
  Repeat adding the next edge of least weight each time provided it does not create a cycle until all vertices are connected



**Edges are added in the order: EH (1), DH (2), AB (4), EF (5), FG (5), BC (6), BF (7)**
Note that the edges EF and FG are of equal weight so could be added in either order

b) State the total weight of the minimum spanning tree.

Add up the weights of the edges in the minimum spanning tree

$1 + 2 + 4 + 5 + 5 + 6 + 7 = 30$

**Total weight = 30**

Your notes

Your notes

## Prim's Algorithm

### Introduction to Prim's Algorithm

#### What is Prim's algorithm?

- **Prim's algorithm** is another method of finding the minimum spanning tree in a **network**
- It can be used when the information for a **network** is given as a **graph** or in a **matrix**

## Prim's Algorithm using Edges & Vertices

Your notes

### What are the steps in Prim's Algorithm when using a graph?

- **Prim's algorithm** involves adding edges from vertices that are **already connected** to the tree.
- Cycles are **avoided** by only adding edges that are **not** already connected at one end.

- STEP 1
  **Start at any vertex** and choose the **edge of least weight** that is **connected** to it

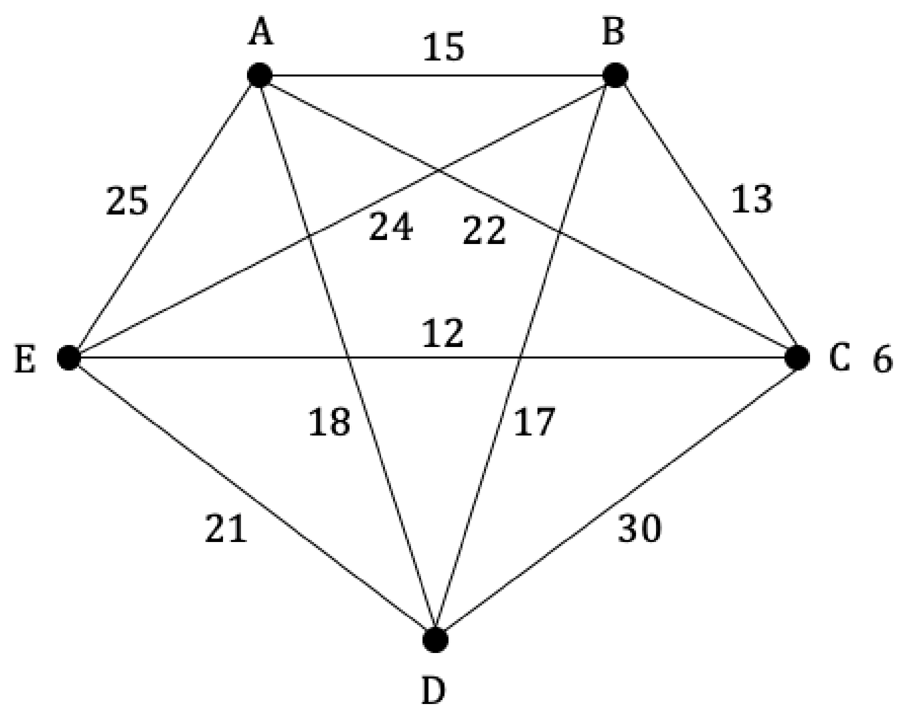
- STEP 2
  Choose the **edge of least weight** that is **incident** (connected) to **any of the vertices already connected** and does not connect to another vertex that is already in the tree
  - If there is a choice of the edges of equal weight, either can be selected

- STEP 3
  Repeat STEP 2 until **all of the vertices** are added to the tree

Your notes

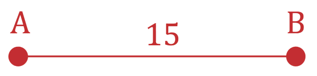
## Worked example

Consider the network, G, below.



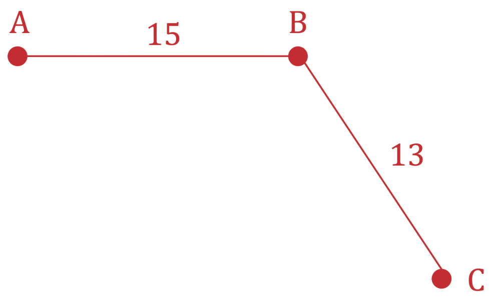
a) Using Prim's algorithm, find the minimum spanning tree.

- **STEP 1**
  Select a starting vertex (A) and choose the edge of least weight that is connected to it



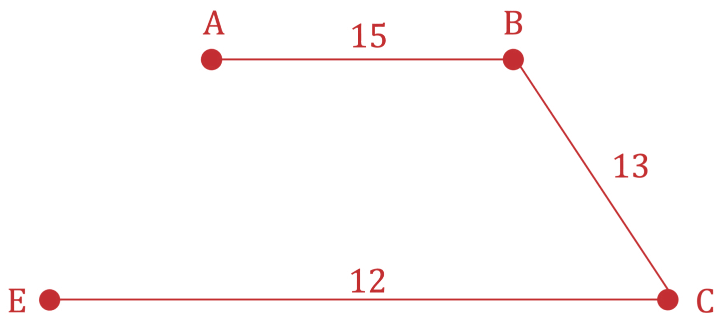
AB (15)

- **STEP 2**
  Select the next edge of least weight that is incident (connected) to either of the vertices already in the tree



AB (15), BC (13)

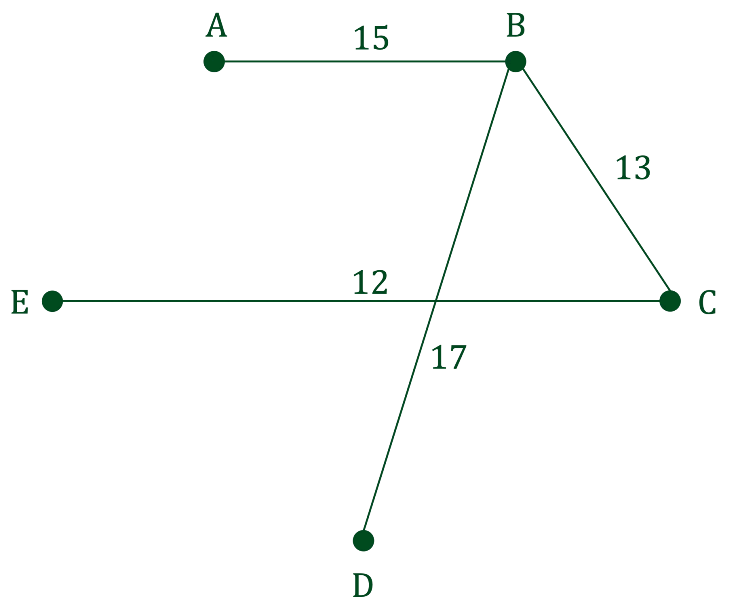
- **STEP 3**
  Continue to select the edge of least weight that is incident to any of the other vertices that are already connected to the tree



AB (15), BC (13), CE (12)

Remember to record the order in which the edges were added

Your notes



**Edges added in the order: AB (15), BC (13), CE (12), BD (17)**

b) State the total weight of the minimum spanning tree.

Add up the weight of the edges in the minimum spanning tree.

$15 + 13 + 12 + 17 = 57$

**Total weight = 57**

## Prim's Algorithm using a Matrix

Your notes

### How do you apply Prim's algorithm to a matrix?

- A minimum spanning tree is built up from the least weight edges that are incident to vertices already in the tree by looking at the **relevant rows** in the **distance matrix**

- STEP 1
  Select **any vertex** to start from, **cross out** the values in the **column** associated with that vertex and **label the row** associated with the vertex 1

- STEP 2
  Circle the **lowest value** in **any cell** along that **row** and **add the edge** to your tree, **cross out** the remaining values in the **column** of the cell that you have circled

- STEP 3
  **Label the row** associated with the same vertex as the column in the previous STEP with the **next number**

- STEP 4
  Circle the **lowest value** in **any cell** along **any of the rows** that have been **labelled** and **add the edge** to your tree, **cross out** the remaining values in the **column** of the cell that you have circled

- STEP 5
  Repeat STEPS 3 and 4 until **all vertices** have been **added** to the tree

- Some versions of Prim's algorithm using a matrix will **cross out rows** and **label columns**

### Examiner Tip

- Look out for questions that ask you to minimise the cost or length etc. from a network – they are implying that they want you to find the minimum spanning tree!
- Be careful not to confuse Prim's algorithm with the Nearest Neighbour algorithm!

Your notes

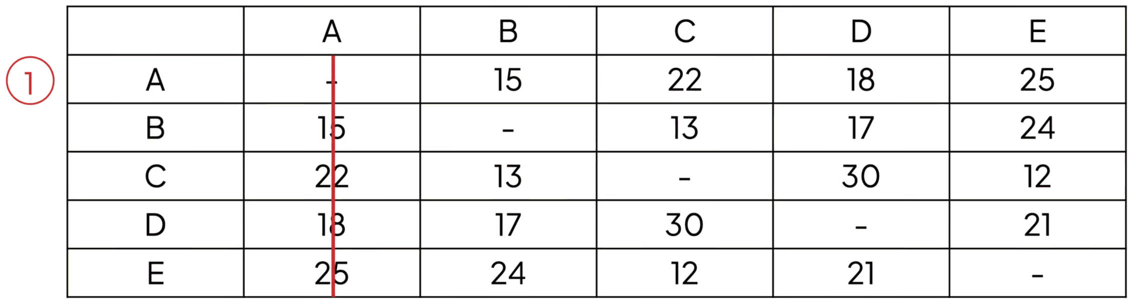
## Worked example

The adjacency matrix of a network, is shown below.

|   | A | B | C | D | E |
|---|---|---|---|---|---|
| A | - | 15 | 22 | 18 | 25 |
| B | 15 | - | 13 | 17 | 24 |
| C | 22 | 13 | - | 30 | 12 |
| D | 18 | 17 | 30 | - | 21 |
| E | 25 | 24 | 12 | 21 | - |

a) Starting from vertex A, use Prim's algorithm on the table to find and draw the minimum spanning tree. Show each step of the process clearly.

- **STEP 1**
  Start at the row for vertex A and label it 1
  Cross out the values in column A



|   |   | A | B | C | D | E |
|---|---|---|---|---|---|---|
|   |   | A | B | C | D | E |
| 1 | A | ~~-~~ | 15 | 22 | 18 | 25 |
|   | B | ~~15~~ | - | 13 | 17 | 24 |
|   | C | ~~22~~ | 13 | - | 30 | 12 |
|   | D | ~~18~~ | 17 | 30 | - | 21 |
|   | E | ~~25~~ | 24 | 12 | 21 | - |

- **STEP 2**
  Circle the edge of least weight in row 1 and record which edge it is
  Delete the remaining values in column B

| | | A | B | C | D | E |
|---|---|---|---|---|---|---|
| 1 | A | ~~-~~ | 15 | 22 | 18 | 25 |
| | B | ~~15~~ | ~~-~~ | 13 | 17 | 24 |
| | C | ~~22~~ | ~~13~~ | - | 30 | 12 |
| | D | ~~18~~ | ~~17~~ | 30 | - | 21 |
| | E | ~~25~~ | ~~24~~ | 12 | 21 | - |

AB (15)

- **STEP 3**
  Label the row for vertex B, 2

- **STEP 4**
  Circle the edge of least weight in row 1 and row 2, remembering to record the edge
  Cross out the remaining values in the column of that circled edge

| | | A | B | C | D | E |
|---|---|---|---|---|---|---|
| 1 | A | ~~-~~ | 15 | ~~22~~ | 18 | 25 |
| 2 | B | ~~15~~ | ~~-~~ | 13 | 17 | 24 |
| | C | ~~22~~ | ~~13~~ | ~~-~~ | 30 | 12 |
| | D | ~~18~~ | ~~17~~ | ~~30~~ | - | 21 |
| | E | ~~25~~ | ~~24~~ | ~~12~~ | 21 | - |

AB (15), BC (13)

- **STEP 5**
  Repeat Steps 3 and 4 until all vertices are added

- **STEP 3**
  Label the row for vertex C, 3

- **STEP 4**
  Circle the edge of least weight in rows 1 to 3, remembering to record the edge
  Delete the remaining values in column E

Your notes

| | | A | B | C | D | E |
|---|---|---|---|---|---|---|
| 1 | A | ~~-~~ | 15 | ~~22~~ | 18 | ~~25~~ |
| 2 | B | ~~15~~ | ~~-~~ | 13 | 17 | ~~24~~ |
| 3 | C | ~~22~~ | ~~13~~ | ~~-~~ | 30 | 12 |
| | D | ~~18~~ | ~~17~~ | ~~30~~ | - | ~~21~~ |
| | E | ~~25~~ | ~~24~~ | ~~12~~ | 21 | ~~-~~ |

AB (15), BC (13), CE (12)

- **STEP 3**
  Label the row for vertex E, 4
- **STEP 4**
  Circle the edge of least weight in rows 1 to 4, remembering to record the edge
  Delete the remaining values in column D

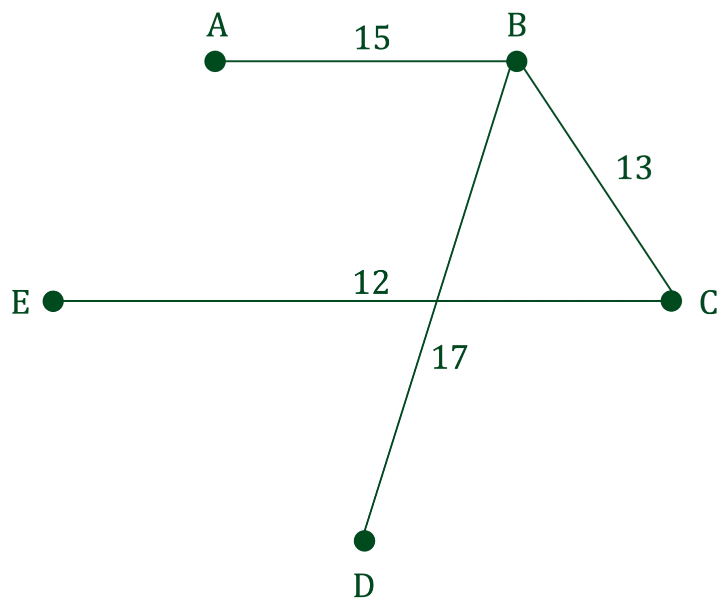
| | | A | B | C | D | E |
|---|---|---|---|---|---|---|
| 1 | A | ~~-~~ | 15 | ~~22~~ | ~~18~~ | ~~25~~ |
| 2 | B | ~~15~~ | ~~-~~ | 13 | 17 | ~~24~~ |
| 3 | C | ~~22~~ | ~~13~~ | ~~-~~ | ~~30~~ | 12 |
| | D | ~~18~~ | ~~17~~ | ~~30~~ | ~~-~~ | ~~21~~ |
| 4 | E | ~~25~~ | ~~24~~ | ~~12~~ | ~~21~~ | ~~-~~ |

AB (15), BC (13), CE (12), DE (17)

All vertices have been selected
You should have a list of the order in which the edges were selected

Your notes

Your notes



**Edges added in the order: AB (15), BC (13), CE (12), BD (17)**

b) State the lowest cost of connecting all of the buildings to the electricity supply.

Add up the weights of the edges in the minimum spanning tree.

$15 + 13 + 12 + 17 = 57$

**Total weight = 57**

## Comparing MST Algorithms

Your notes

### Comparing MST Algorithms

**Which should I use – Prim's or Kruskal's algorithm?**

- Both algorithms find a minimum spanning tree
  - they may give different answers but will both produce a minimal solution
- **Prim's algorithm** can be used in **either graph or matrix** form
  - If an MST is asked for and the information is given in **table/matrix** form, use **Prim's algorithm**
- **Kruskal's algorithm** can be used when the information is in **graph** form (or if a list of arcs is available)
  - If trying to use Kruskal's algorithm from a table/matrix, it would be best to draw a graph first
- Kruskal's algorithm allows arcs to be added in **any** order
  - i.e. the tree can be **'split'** (unconnected) at stages of the algorithm
  - Prim's algorithm 'builds' as arcs are added such that they will connect to arcs **already** in the tree
- **Prim's algorithm** is sometimes considered to be more **efficient** that **Kruskal's algorithm** as
  - the edges **do not need to be ordered** at the start and
  - it does not rely on **checking for cycles** at each step
- An **exam question** will usually specify which method should be used, otherwise either is fine

#### Examiner Tip

- A common exam question is to state that certain edges need to be included in a spanning tree
  - you are then asked to describe which algorithm you should use and how it should be adapted
  - you should start off by drawing in the edges given, and then complete the tree using Kruskal's algorithm!