

# SUMMARY SHEET – DECISION MATHS

## Algorithms

The main ideas are covered in

<b>AQA</b>	<b>Edexcel</b>	<b>MEI</b>	<b>OCR</b>
<b>D1</b>	<b>D1</b>	<b>D1</b>	<b>D1</b>

The main ideas in this topic are

Understanding and implementing a variety of algorithms expressed as lists of instructions, flow charts or in pseudo code.

### What is an algorithm?

An algorithm must have the following properties

- it is a set of precisely defined instructions.
- it has generality: it will work for all valid inputs.
- it is finite: it has a stopping condition.
- it may be an iterative process: you may need to follow the procedure a number of times in order to reach the best solution.

### Bin Packing

These are examples of HEURISTIC algorithms. This means that none of these algorithms necessarily lead you to the best or optimal solution of the problem.

#### 1. Full-Bin Algorithm

Look for combinations of boxes to fill bins. Pack these boxes. For the remainder, place the next box to be packed in the first available slot that can take that box.

Note – the full bin algorithm does not always lead to the same solution of the problem. In other words, two people could apply the full bin algorithm perfectly correctly and end up with their boxes packed differently.

#### 2. First-Fit Algorithm

Taking the boxes in the order listed, place the next box to be packed in the first available slot that can take that box.

#### 3. First-Fit Decreasing Algorithm

- Re-order the boxes in order of decreasing size.
- Apply the First-Fit algorithm to this reordered list.

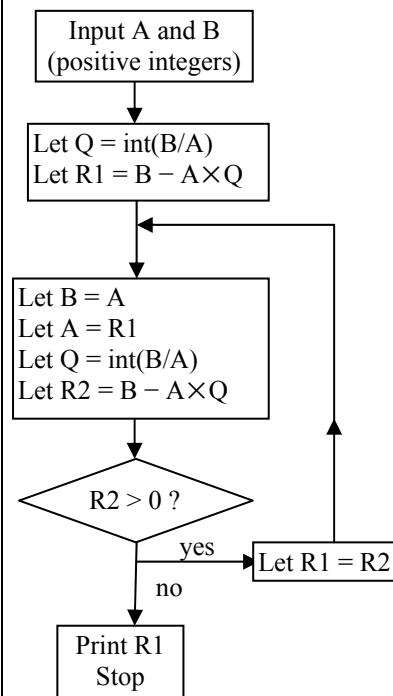
You should be able to form a judgement about the relative efficiency of these algorithms. The First-Fit Decreasing Algorithm requires a sort to be made before applying the First-Fit Algorithm so, in terms of computation, it requires more resources than the First-Fit Algorithm alone.

### Before the exam you should know:

- The three bin packing algorithms. These are the Full-Bin Algorithm, the First-Fit Algorithm and the First-Fit Decreasing Algorithm.
- The sorting algorithms. Make sure you know which of these algorithms you need to learn by heart.
- How to count the number of comparisons and swaps in each pass and know the maximum number of passes that are required for a list of a given length.
- The different ways algorithms are presented and make sure you practice following unfamiliar algorithms.
- What is meant by efficiency of an algorithm.

### Presenting and Implementing Algorithms

An algorithm is a well-defined, finite sequence of instructions to solve a problem. They can be communicated in various ways, including written English, pseudo code and flowcharts. Make sure you are experienced in all possible formats.



### Example

- What is the output of the algorithm when  $A = 84$  and  $B = 660$ ?
- What does the algorithm achieve?

### Solution

a)

A	84	72	12
B	660	84	72
Q	7	1	6
R1	72		12
R2		12	0

PRINT 12

- b) It finds the highest common factor of A and B.

**Example:** Show how the following items are to be packed into boxes each of which has a capacity of 10Kg.

Item	A	B	C	D	E	F
Weight (kg)	2	4	6	3	3	5

**1. Full Bin**

6+4=10, 5+3+2=10, 3 3 bins needed

**2. First-Fit**

3kg	3kg	
4kg	6kg	
2kg		5kg

**3. First-Fit Decreasing**

4kg	2kg	
6kg	3kg	
	5kg	
		3kg

Notice that in this example the First-Fit Decreasing Algorithm gives the same result as the Full Bin Algorithm. This will not always be the case.

## Sorting Algorithms

There are many sorting algorithms, so you must check carefully to see which, if any, you need to memorise for the examination.

Questions often ask about the relative efficiency of sorting algorithms by comparing the number of comparisons (c) and swaps that are made to sort the same list of numbers, as seen in this example:

List	1 <sup>st</sup> pass	2 <sup>nd</sup> pass	3 <sup>rd</sup> pass
6	1	1	1
1	3	3	3
3	6	5	5
7	5	<b>6</b>	6
5	<b>7</b>	<b>7</b>	7
<i>c</i>	4	3	2
<i>s</i>	3	1	0

total number of comparisons: **9**

total number of swaps: **4**

### Bubble Sort

**First pass:** the first number in the list is compared with the second and whichever is smaller assumes the first position. The second number is then compared with the third and the smaller is placed in the second position, and so on. At the end of the first pass, the largest number will be at the bottom. For the list of five numbers on the right, this involves 4 comparisons and 3 swaps.

**Second pass:** repeat first pass but exclude the last number (on the third pass the last two numbers are excluded and so on).

The list is repeatedly processed in this way until **no swaps take place in a pass**.

For a list of 5 numbers, the list will definitely be sorted after the 4<sup>th</sup> pass (why?), so this is the maximum number of passes. The maximum number of comparisons is 4+3+2+1=10 and the maximum number of swaps is 10. You should be able to generalise this to a list of *n* numbers.

list	1 <sup>st</sup> pass	2 <sup>nd</sup> pass	3 <sup>rd</sup> pass	4 <sup>th</sup> pass
6	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>
1	<b>3</b>	<b>3</b>	<b>3</b>	<b>3</b>
<b>3</b>	6	<b>6</b>	<b>5</b>	<b>5</b>
7	<b>7</b>	5	<b>6</b>	<b>6</b>
5	5	<b>7</b>	<b>7</b>	<b>7</b>
<i>c</i>	4	2	1	0
<i>s</i>	1	1	1	0

total number of comparisons: **7**

total number of swaps: **3**

### Quick Sort

Select a pivot – usually the middle item in the list

**First pass:** numbers are sorted into two sub lists, those smaller than the pivot element and those greater than the pivot element. The pivot element is now fixed in its correct position in the list.

**Second pass:** choose a pivot element in each of the two sub lists and repeat the sorting procedure.

Continue this process until all numbers are fixed and the list is sorted.

In this case the quick sort takes fewer comparisons and swaps than the bubble sort, though it does take one more pass to achieve the sort. It is worth noting that the relative efficiency of the different types of algorithm will vary depending on how “mixed up” the list is.

# SUMMARY SHEET – DECISION MATHS

## CRITICAL PATH ANALYSIS

### The main ideas are covered in

<b>AQA</b>	<b>Edexcel</b>	<b>MEI</b>	<b>OCR</b>
<b>D2</b>	<b>D1</b>	<b>D1</b>	<b>D2</b>

### The main ideas in this topic are

- Drawing Activity or Precedence Networks
- Performing Forward and Backward Passes and Identifying Critical Activities
- Drawing Cascade Charts and Resource Levelling

### Before the exam you should know

- How to draw precedence networks, as you possibly can.
- When you need to use dummy activities.
- How to perform forward and backward passes on a precedence network to calculate early and late start times.
- How to find the critical activities.
- How to calculate the various types of float.
- How to draw a cascade chart and construct a resource histogram.
- Where resource levelling is required and how to make effective use of float to improve efficiency.
- What is meant by crashing a network.

### Terminology

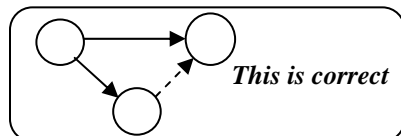
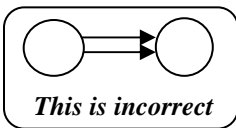
**An activity** is a task which needs to be done and takes an amount of time/resources to complete.

**Precedence tables** show the activities that need to be done together with their duration and their immediate predecessors.

**Precedence networks** show the sequence of the activities. The network must have one start node and one end node.

An **event** is the start/finish of one or more activities.

**Dummy activities** are used to keep the correct logic and to ensure each activity is **uniquely** defined by (i, j) where i is its starting event and j is the finishing event.



It can be a good idea to do an initial sketch as it's often possible to make your diagram clearer by repositioning activities to avoid them crossing over one another.

**Forward pass** establishes the earliest times that events can happen.

**Backward pass** establishes the latest time that an event can happen.

**Critical activities** are those whose timing is critical if the project is to be completed in the minimum time. The critical activities will form a path through the network

**Float** is the amount of time by which an activity can be delayed or extended.

**Independent float** does not affect other activities.

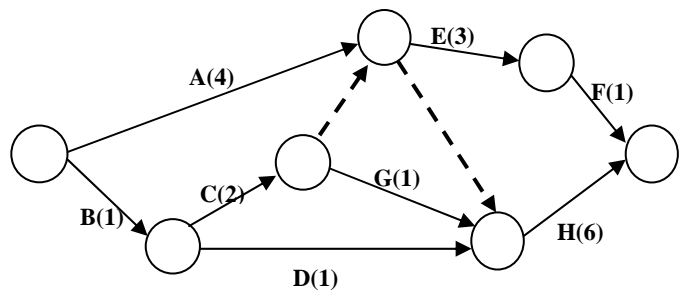
**Interfering float** is shared between two or more activities.

### Example:

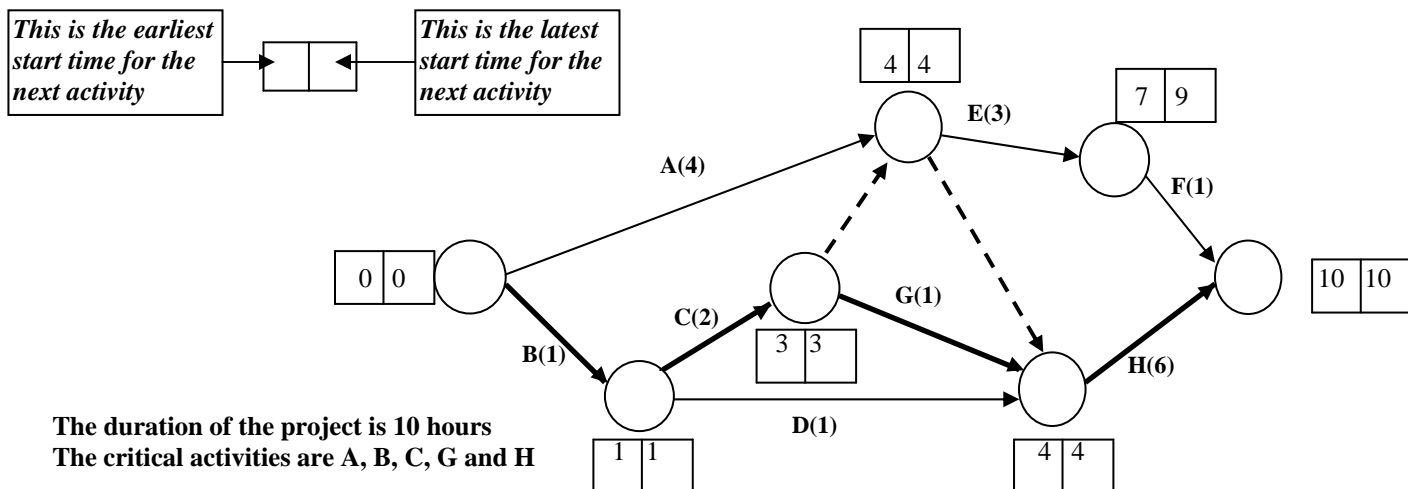
The table shows the activities involved in creating a small patio in a garden.

Activity Name	Task	Time (hrs)	Preceding Activities
A	Clear Garden	4	
B	Measure area	1	
C	Design Patio	2	B
D	Purchase fencing	1	B
E	Buy pots and plants	3	A,C
F	Plant all pots	1	E
G	Purchase paving	1	C
H	Construct Garden	6	A, D,G

### The network for this precedence table



### The forward and backward pass



### Float

activity	float	type
D	2 hours	independent
E	2 hours	Interfering (with F)
F	2 hour	Interfering (with E)

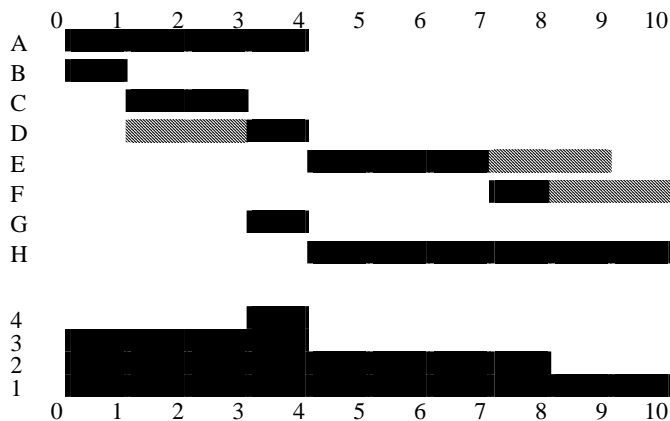
*In this example there are two hours of float shared between activities E and F*

### Cascade Chart and Resources levelling



A Cascade Chart shows each activity set against a time line. Float time is sometimes shown by using shading. Dependencies are shown by vertical lines. The cascade chart can be adjusted by using the float times to make use of resources more efficient.

If activity A needs two people and all the rest can be done by one person, then the resource histogram looks like this (note that 4 people are needed in the second hour).



If only three people are available for the first three hours, but a fourth friend can then come and help for an hour, we could move activity D within its float time to make this possible.

This would make the cascade chart look like this

The resource histogram would now look like this

## SUMMARY SHEET – DECISION MATHS

## Graph Theory

## The main ideas are covered in

AQA	Edexcel	MEI	OCR
D1	D1	D1	D1

## The main ideas in this topic are

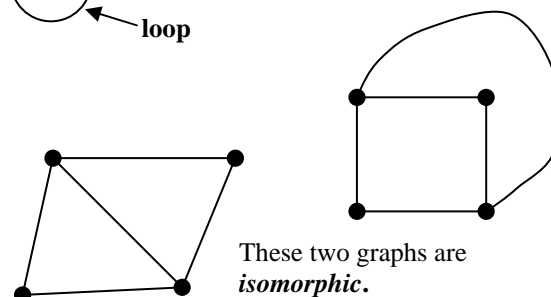
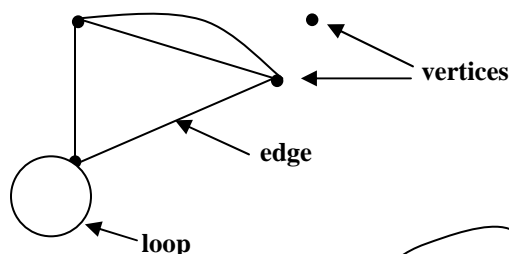
- The definition of a graph and the associated vocabulary.
- Mathematical modeling with graphs.

## Terminology for Graph Theory

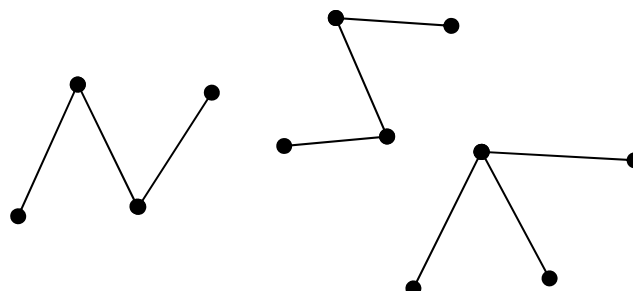
- **Graph** – collection of vertices & edges.
- **Vertex/Node** – the dots in a graph (usually where 2 or more edges meet, but not necessarily).
- **Edge/Arc** – a line between two vertices.
- **Tree** – a graph with no cycles.
- **Order (degree) of a vertex** – the number of edges starting or finishing at that vertex.
- **Simple graph** – a graph with no loops or multiple edges.
- **A path** – a route from one vertex to another which does not repeat any edge.
- **A cycle** – a route starting and finishing at the same vertex.
- **Connected graph** – a graph in which there is a route from each vertex to any other vertex (i.e. the graph is in one part).
- **Complete graph** – a simple graph in which every pair of vertices is connected by an edge.
- **Bipartite graph** – one in which the vertices are in two sets and each edge has a vertex from each set.
- **Planar graph** – one which can be drawn with no edges crossing.
- **Sub graph** – any set of edges & vertices taken from a graph is a sub-graph.
- **Hamiltonian cycle** – a cycle that visits every vertex of the graph.
- **Eulerian cycle** – a cycle that travels along every edge of the graph.
- **Eulerian graph** – a graph with no odd vertices.
- **Di-graph** – a graph in which the edges indicate direction.
- **Incidence matrix** – a matrix representing the edges in a graph.

Before the exam you should know:

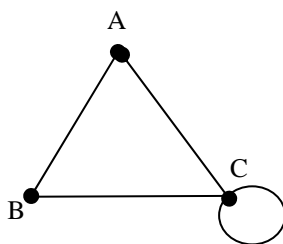
- The terms vertices (nodes), edges (arcs), digraphs, trees and paths.
- All the other vocabulary used to describe ideas in graph theory.
- How to draw a graph from an incidence matrix.
- How to model problems using graphs (e.g. Konigsberg Bridges).
- What is meant by a tree.
- How to recognise isomorphic graphs.
- What is meant by a Hamiltonian cycle.
- What is meant by an Euler cycle.



## These diagrams all show trees of the graph above

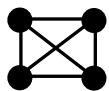


This shows a graph and its Incidence matrix.

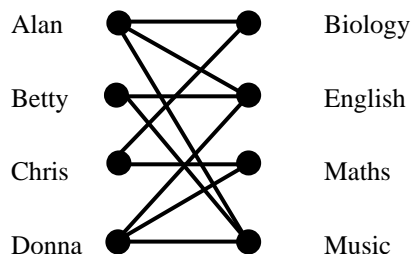
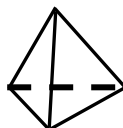


		To		
		A	B	C
From	A	-	1	1
	B	1	-	1
	C	1	1	2

Graphs can be used to represent many different things



This graph represents a tetrahedron



This *bipartite graph* shows which subjects four students study.

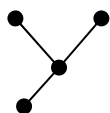
**Example**

The table shows the number of vertices of degree 1, 2, 3 and 4 for three different graphs. Draw an example of each of these graphs.

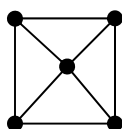
Order of vertex	1	2	3	4
Graph 1	3	0	1	0
Graph 2	0	0	4	1
Graph 3	0	2	2	1

**solution**

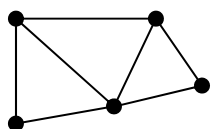
graph 1



graph 2



graph 3



Find the number of edges and the sum of the degrees of all the vertices of the graphs. What do you notice?

- Graph 1: number of edges 3    sum of degrees of vertices  $1+1+1+3 = 6$
- Graph 2: number of edges 8    sum of degrees of vertices  $3+3+3+3+4=16$
- Graph 3: number of edges 7    sum of degrees of vertices  $2+2+3+3+4 = 14$

*The sum of the degrees of the vertices is always twice the number of edges.*

*Also note that there are always an even number of odd vertices.*

# SUMMARY SHEET – DECISION MATHS

## GRAPHICAL LINEAR PROGRAMMING

The main ideas are covered in

AQA	Edexcel	MEI	OCR
D1	D1	D1	D1

The main ideas in this chapter are

Formulating a problem as a linear programming problem, solving a Linear Programming Problem (maximisation and minimisation) and Integer Programming.

### Formulating a problem as a Linear Programming Problem

**First:** identify the variables about which a decision is to be made. These are sometimes called the decision variables. For example if your problem is to decide how many chairs to make and how many tables to make to maximise profit, begin with a statement like – let  $x$  be the number of chairs and let  $y$  be the number of tables. If your problem is to work out how many grams of wheatgerm and how grams of oat flour there should be in a new food product to meet nutritional requirements and minimise cost then let  $x$  be the number of grams of wheatgerm and let  $y$  be the number of grams of oat flour.

**Next:** Decide what the objective function is (this is the value you are trying to maximise or minimise) and what the constraints are as inequalities involving  $x$  and  $y$ .

Be careful to use the same units consistently. For example it's possible that some distances appearing in a problem are given in metres and some are given in centimetres. Or some times they could be given in seconds with some given in minutes. Choose one type of units and convert everything into those units.

#### Example:

A clothing retailer needs to order at least 200 jackets to satisfy demand over the next sales period. He stocks two types of jacket which cost him £10 and £30 to purchase. He sells them at 20 pounds and 50 pounds respectively. He has 2700 pounds to spend on jackets.

The cheaper jackets are bulky and each need 20cm of hanging space. The expensive jackets need only 10cm each. He has 40m of hanging space for jackets.

The retailer wishes to maximise profit. Assuming that all jackets will be sold, formulate a linear program, the solution of which will indicate how many jackets of each type should be ordered.

#### Before the exam you should:

- Practice formulating linear programming problems. This can often be the trickiest part of the problem. Remember to be consistent with units.
- Learn the terminology – the OBJECTIVE FUNCTION is what you have to maximise or minimise subject to a number of CONSTRAINTS.
- Make sure you are able to draw straight line graphs quickly from the constraints by considering where they cross the  $x$  and  $y$  axes.
- You must be able to find the solution to problems from the graph. Make sure you can draw graphs accurately.
- Remember to shade OUT the unacceptable region to keep the feasible region clear and easy to identify.
- You must be able to find correct solutions to problems where the answer must be an integer.

#### Formulation as a linear program

The decision is about how many of two types of jacket need to be ordered.

Let  $x$  = number of cheaper jackets ordered

Let  $y$  = number of expensive jackets ordered

The profit,  $P$ , given by selling all of these, is  $P = 10x + 20y$ , since the profit made on a cheaper jacket is 10 pounds and the profit made on an expensive one is 20 pounds.

The constraints are:

1. "needs to order at least 200" giving  $x + y \geq 200$
2. "cost him 10 pounds and 30 pounds" and "has 2700 pounds to spend" giving  $10x + 30y \leq 2700$
3. "20cm of hanging space" and "10cm" and "has 40m of hanging space" giving  $0.2x + 0.1y \leq 40$

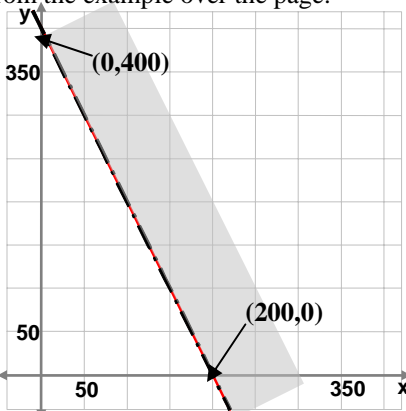
# Solving a Linear Programming Problem

Draw a graph in which each constraint is represented by a line with shading. The unacceptable side of the line should be shaded. This leaves a “feasible region”. The solution of the problem will be one of the vertices of the feasible region. These can be checked to find the best. We do this below for the example introduced over the page.

## Drawing the line representing a constraint.

As an example, take the constraint  $0.2x + 0.1y \leq 40$  from the example over the page.

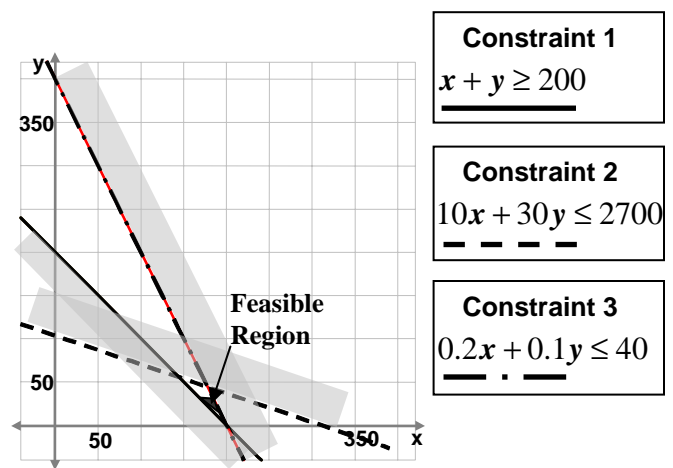
The initial aim is to draw the line  $0.2x + 0.1y = 40$ . We know this is a straight line so it's enough to find two points on the line and join them. When  $x = 0$ ,  $y = 400$  and when  $y = 0$ ,  $x = 200$ . So the points  $(0, 400)$  and  $(200, 0)$  are on the line.



Then shade out, the unacceptable region. To find the unacceptable region just test a point to see if it satisfies the constraint or not. For example, in this case  $(10, 10)$  clearly satisfies the constraint and so is in the acceptable region.

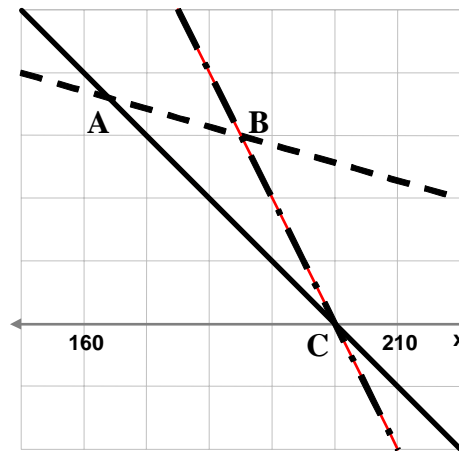
## The feasible region.

Once you have drawn all the constraints, the feasible region is the intersection of the acceptable regions for all of them.



## Finding the solution

The solution of the problem will be at one of the vertices of the feasible region. You will need to solve simultaneous equations to find the co-ordinates of these vertices. Then each vertex must be checked to find the best. For example in the above we have a feasible region as in the diagram on the right. The coordinates of point A are found by solving  $x + y = 200$  and  $10x + 30y = 2700$  simultaneously. The solutions are  $x = 165$  and  $y = 35$ . So the point is  $(165, 35)$  and the profit at that point is  $P = 10x + 20y = 1650 + 700 = 2350$ . Similarly it can be seen point B is  $(186, 28)$  giving a profit of 2420. Point C is  $(200, 0)$  giving a profit of 2000. So the best profit that can be made is by buying 165 cheap coats and 35 expensive coats.



## Considering Gradients.

By calculating the gradients of each of the constraints and the gradient of the objective function, it's possible to predict in advance which vertex will give the optimal solution.

**Minimisation problems** are solved in exactly the same way. Just remember that this time you are looking for the vertex which makes the objective function the lowest.

## Integer Programming

If the solution to the problem has to have integer values then points with integer value coordinates, close to the optimal point can be checked. This is likely to reveal the optimal solution but it is not guaranteed to. For example suppose the Objective Function is  $2x + 3y$  and that this should be maximised. The optimal point may be  $(30.6, 40.8)$  but do not assume that  $(30, 40)$  will give the best solution; you must look at all the points with integer coordinates that are nearby:  $(31, 40)$ ,  $(30, 41)$ ,  $(30, 40)$  and  $(31, 41)$ .

However  $(31, 41)$  and  $(31, 40)$  are not in the feasible region. You can check this by substituting in the values into the constraints. Of the two points nearby which are in the feasible region, namely  $(30, 41)$  and  $(30, 40)$ , it can be seen that  $(30, 41)$  provides the best profit.



# REVISION SHEET – DECISION MATHS

## Linear programming: the Simplex algorithm 1

### The main ideas are covered in

<b>AQA</b>	<b>Edexcel</b>	<b>MEI</b>	<b>OCR</b>
<b>D2</b>	<b>D1</b>	<b>D2</b>	<b>D1</b>

### The main idea in this topic is:

Using the Simplex algorithm to deal with linear programming problems with more than two variables.

### Before the exam you should know:

- How to formulate a linear programming problem to maximise the objective function, subject to the given constraints.
- How to use slack variables to convert inequality constraints into equations.
- How to set up the initial simplex tableau.
- How to perform the Simple algorithm for maximising an objective function.
- How to identify initial, intermediate and final tableaux and know when the solution is optimal.
- How to interpret the values of the variables and the objective function at any stage in the Simplex method.
- That you must state the solution in the context of the original problem.

## Simplex Method for Maximisation Problems

### Getting started: Formulation

Translating a real life problem into a linear programming problem is called formulating the problem and is an example of mathematical modelling. Each problem must have clearly defined variables, an objective function and is subject to certain constraints

### Slack Variables

In order to enable problems to be converted into a format that can be dealt with by computer, slack variables are introduced to change the constraint inequalities into equalities. Each vertex of the feasible region would then be defined by the intersection of lines where some of these variables equal zero.

### The Simplex Method

The Simplex Method starts at one vertex and systematically moves round all the vertices of the feasible region, increasing the objective function as it goes, until it reaches the one with the optimal solution. This is easy to visualise on a 2 dimensional problem, but can be generalised to include more variables. Once there are more than two variables, a graphical approach is no longer appropriate, so we use the simplex tableau, a tabular form of the algorithm which uses row reduction to solve the problem.

### The Simplex Algorithm

1. Represent the problem in a tableau.
2. Use the objective row to find the pivot column.
3. Use the ratio test to find the pivot element.
4. Divide through the pivot row by the pivot element.
5. Add/subtract multiples of the transformed pivot row to/from the other rows to create zeros in the pivot column.
6. Repeat until no negatives in objective row.
7. Read the solution from the table.

#### Note on finding pivot column (step 2)

You can choose any variable in the objective row with a negative coefficient, but it is usual to pick the most negative. Give priority to the original rather than slack variables.

#### Note on ratio test (step 3)

Divide each R.H.S. value by the corresponding element in the pivot column, ignore negative ratios and division by zero. Choose row with the smallest ratio as the pivot row.

**Example:**

A manufacturer makes three products  $x$ ,  $y$  and  $z$  which give profits of £160, £120 and £120 per tonne respectively. Production is constrained by availability of staff and storage as summarised in this table:

	Staff time (hours /tonne)	Storage (m <sup>3</sup> /tonne)
$x$	5	5
$y$	5	3
$z$	6	4
availability	30	20

Formulate a linear programming problem.

**Objective function**

Objective function is maximise  $P = 160x + 120y + 120z$   
 $\Rightarrow P - 160x - 120y - 120z = 0$

**Constraints**

Subject to:  $5x + 5y + 6z + s_1 = 30$  (staff time)  
 $5x + 3y + 4z + s_2 = 20$  (storage)  
 $x \geq 0, y \geq 0, z \geq 0, s_1 \geq 0, s_2 \geq 0$

**Solving the problem**

Considering the problem in the example on the previous page, we must now set up an initial tableau.

**Setting up the initial tableau**

$P$	$x$	$y$	$z$	$s_1$	$s_2$	RHS	<i>Ratio test</i>
1	-160	-120	-120	0	0	0	
0	5	5	6	1	0	30	$30/5 = 6$
0	5	3	4	0	1	20	$20/5 = 4$

Look for the largest negative value in the objective row. This is in the  $x$  column, so we will choose to increase  $x$  first. Since  $4 < 6$  so we choose 5 in row 3 as the pivot element. The idea now is to make the pivot 1 so divide the pivot row by 5.

**First iteration**

	$P$	$x$	$y$	$z$	$s_1$	$s_2$	RHS	<i>Ratio test</i>
row 1 + 160 × row 3	1	0	-24	8	0	32	640	
row 2 - 5 × row 3	0	0	2	2	1	-1	10	$10/2 = 5$
row 3	0	1	0.6	0.8	0	0.2	4	$4/0.4 = 10$

Look for the largest negative value in the objective row. This is in the  $y$  column, so we will choose to increase  $y$ . Since  $5 < 10$  we choose 2 in row 2 as the pivot element. The idea is to make the pivot 1 so divide the pivot row by 2.

**Second iteration**

	$P$	$x$	$y$	$z$	$s_1$	$s_2$	RHS
row 1 + 24 × row 2	1	0	0	32	12	20	760
row 2	0	0	1	1	0.5	-0.5	5
row 3 - 0.6 × row 2	0	1	0	0.2	-0.3	0.5	1

The solution is optimal since there are no negative values in the objective row.

**Reading the tableau**

The final tableau represents the following set of equations

- Row 1:  $P + 32z + s_1 + s_2 = 760$
- Row 2:  $y + z + 0.5s_1 - 0.5s_2 = 5$
- Row 3:  $x + 0.2z - 0.3s_1 + 0.2s_2 = 1$

The most obvious solution to this is obtained by setting the “basic” variables (columns with zeros and a single 1) equal to the RHS and setting the “non-basic” variables (columns with more than one non-zero entry) equal to 0. This gives the solution  $P = 760, x = 1, y = 5, z = 0, s_1 = 0, s_2 = 0$

You can check your solution by substituting the values obtained for  $x$ ,  $y$  and  $z$  into the original objective function to check that the profit is correct:  
 $P = (160 \times 1) + (120 \times 5) + (120 \times 0)$   
 $= 160 + 600 + 0$   
 $= 760$

**Interpreting the solution**

In order to maximise his profit the manufacturer should make one tonne of product  $x$ , five tones of product  $y$  and no product  $z$ . this would use all the available resources and would generate a profit of £760.00.

## SUMMARY SHEET – DECISION MATHS

## Matchings

The main ideas are covered in

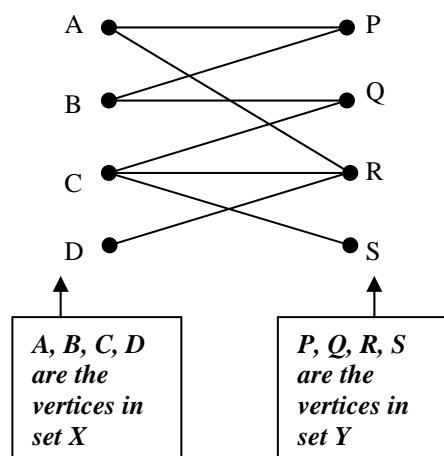
AQA	Edexcel	MEI	OCR
D1	D1		D2

The main ideas in this topic are:

Modelling real situations using bipartite graphs.

Using the maximum matching algorithm to solve problems.

A **bipartite graph** has two sets of vertices, X and Y such that the edges only connect vertices in set X to those in set Y and never to vertices in the same set.



### Example

A college has to fit French, Geography, History, Maths and Science into a single timetable slot. There are five teachers available all of whom can teach two or more of these subjects.

Ann can teach French and Geography.  
 Bob can teach French, Maths and Science.  
 Carol can teach Geography and History.  
 David can teach Geography, Maths and Science.  
 Elaine can teach History Maths and Science.

How should the college allocate the staff so that all subjects are covered?

### Before the exam you should know:

- What is meant by a bipartite graph.
- That a matching maps vertices in one set to vertices in a second set. No vertex may be used more than once.
- For a complete matching the two sets must have the same number of vertices.
- A complete matching pairs every vertex in the first set to one in the second set.
- A maximal matching is one where there is no solution that uses a greater number of edges.
- A complete matching is always maximal, but a maximal matching is not necessarily complete.

### The algorithm for finding a maximum matching

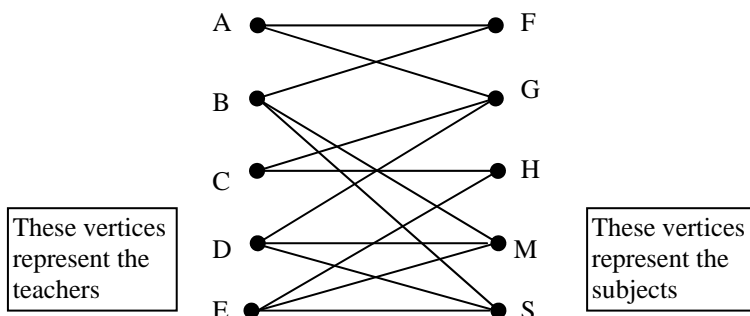
1. Always start with an initial matching.
2. If the matching is not maximal it can be improved by finding an *alternating path*.

An alternating path:

- Starts on an unmatched vertex on the right hand side.
  - Consists of edges alternately *not in* and *in* the matching.
  - Finishes on an unmatched vertex in the second set.
3. If every vertex is now matched so we have a complete matching. If it is not, then repeat step 2.
  4. The solution consists of:
    - Edges *in* the alternating path but *not in* the initial matching.
    - Edges *in* the initial matching but *not in* the alternating path.

**Solution**

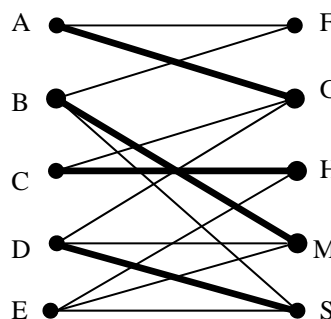
Start by drawing a bipartite graph to model the situation



Start with an initial matching:

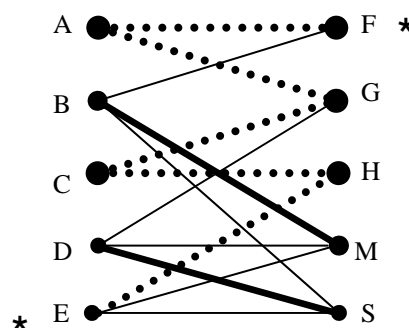
A – G, B – M, C – H, D – S

This is not a maximum matching since Elaine has not been allocated a subject and there is no-one to teach French.



We must try to find an *alternating path*

- Start on an unmatched vertex on the right hand side (F)
- Choose an edge which is not in the initial matching (FA)
- Choose an edge which is in the initial matching (AG)
- Choose an edge which is not in the initial matching (GC)
- Choose an edge which is in the initial matching (CH)
- Choose an edge which is not in the initial matching (HE)



We have now reached E which was not in the initial matching so we have a *breakthrough*.

The alternating path is F – A – G – C – H – E

The solution consists of:

- Edges *in* the alternating path but *not in* the initial matching: AF, CG, EH
- Edges *in* the initial matching but *not in* the alternating path: BM, DS

So the solution is:

- Ann teaches French
- Bob teaches Maths
- Carol teaches Geography
- David teaches Science
- Elaine teaches History

# SUMMARY SHEET – DECISION MATHS

## Network Flows

### The main ideas are covered in

<b>AQA</b>	<b>Edexcel</b>	<b>MEI</b>	<b>OCR</b>
<b>D2</b>	<b>D1</b>		<b>D2</b>

### The main ideas in this topic are:

Modelling flows using bipartite graphs.  
 Finding the maximum flow through a network.  
 The maximum flow-minimum cut theorem.

### Before the exam you should know

- What is meant by source, sink and capacity.
- What a cut is.
- How to find an initial flow.
- The meaning of a flow augmenting path and how to find them.
- How to use the labeling procedure.
- What is meant by excess capacity and back capacity.
- What is meant by a saturated arc.
- The maximum flow – minimum cut theorem.
- How to insert a super-source and super-sink into a network.

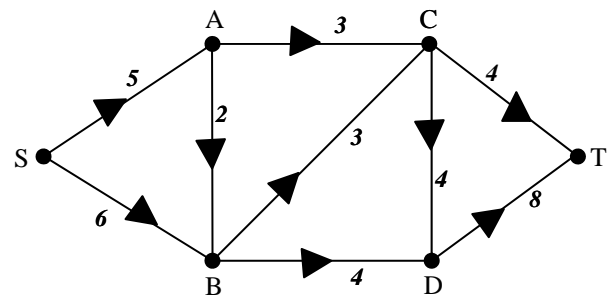
### The algorithm for finding a maximum flow

1. Always start with an initial feasible flow, found by inspection.
2. Label each arc with
  - The flow along it, shown by an arrow pointing back towards the source.
  - The excess capacity, which is the amount by which the flow could be increased, shown by an arrow pointing forward towards the sink.
3. Systematically look for flow augmenting paths and mark these on your network using the labelling procedure.
4. When all paths are blocked by saturated arcs you have found the maximum flow.

### Example

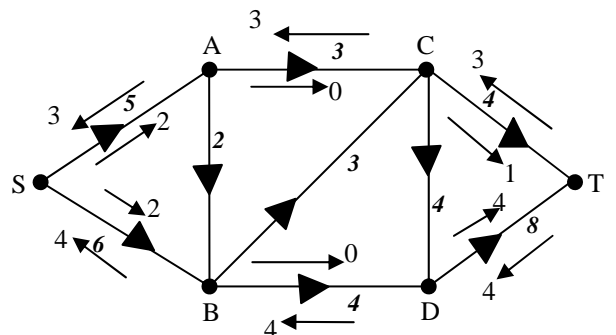
In this directed network:

- a) What is the maximum flow along the path SACT?
- b) Find an initial flow of value 7.
- c) Find the maximum flow in the network.
- d) What are the capacities of these cuts?



### Solution

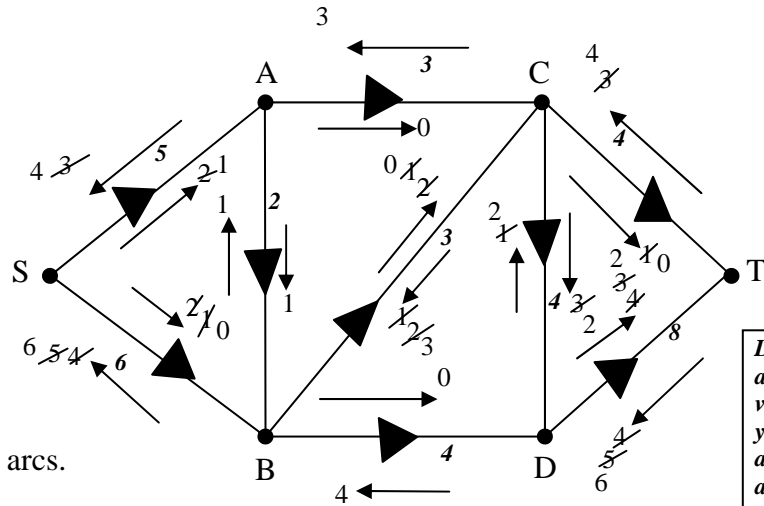
- a) The maximum flow along SACT is 3 (this is determined by the arc of least capacity on the path).
- b) A flow of 7 (shown on the diagram) is
  - SACT with capacity 3
  - SBDT with capacity 4



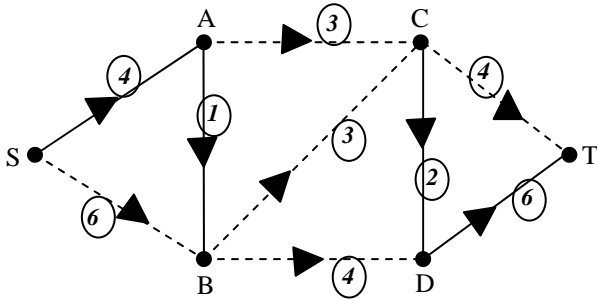
c) Flow augmenting paths

- SBCT with capacity 1
- SBCDT with capacity 1
- SABCDT with capacity 1

This gives a maximum total flow of 10. The flow is shown on this diagram, along with the saturated arcs.



Diagrams showing the flow augmenting paths can be very messy. Try to keep yours as tidy as possible and always list the flow augmenting paths you have used



**Cuts**

A cut partitions the vertices into two sets, one containing the source and one containing the sink. The capacity of a cut is the total of all the cut edges with direction going from source to sink

*Find the capacity of the cuts shown on the directed network:*

Note that only three cuts have been shown here, but there are many more cuts in this network.

$C_1$  is the cut  $\{S\}, \{A, B, C, D, T\}$

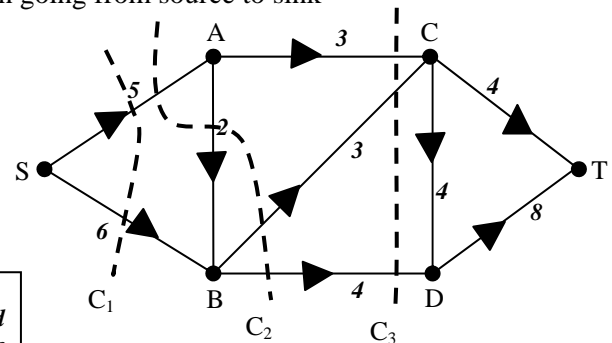
It has capacity  $5 + 6 = 11$

$C_2$  is the cut  $\{S, B\}, \{A, C, D, T\}$

It has capacity  $5 + 0 + 3 + 4 = 12$

$C_3$  is the cut  $\{S, A, B\}, \{C, D, T\}$

It has capacity  $3 + 3 + 4 = 10$



Note that we do not add the capacity of arc AB as it is directed from the sink side of the cut to the source side

**Maximum flow- minimum cut theorem**

The theorem states that the maximum flow in a directed network is equal to the capacity of the minimum cut.

In the example above the cut  $C_3$  is the minimum cut and it has a value 10. This confirms that the flow of 10 found in (c) above is the maximum flow.

**Networks with many sources and sinks**

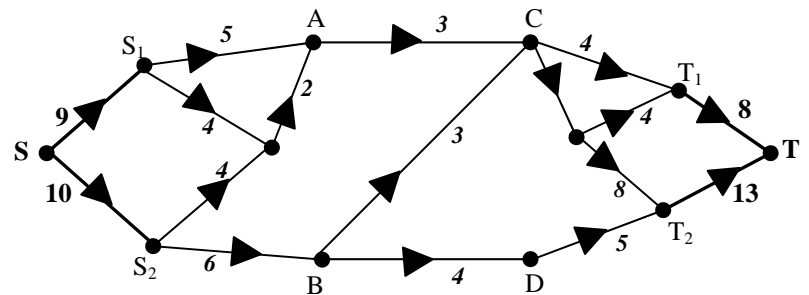
If there is more than one source ( $S_1$  and  $S_2$  on the diagram) or sink ( $T_1$  and  $T_2$  on the diagram) you must introduce *supersource* ( $S$ ) and/or *supersink* ( $T$ ).

$SS_1$  must have a capacity  $5 + 4 = 9$

$SS_2$  must have capacity  $4 + 6 = 10$

$T_1T$  must have capacity  $4 + 4 = 8$

$T_2T$  must have capacity  $8 + 5 = 13$



# SUMMARY SHEET – DECISION MATHS 1

## NETWORKS – Minimum spanning tree and shortest path

The main ideas are covered in

<b>AQA</b>	<b>Edexcel</b>	<b>MEI</b>	<b>OCR</b>
<b>D1</b>	<b>D1</b>	<b>D1</b>	<b>D1</b>

Before the exam you should know:

- How to show all the working clearly, there are more marks for the working than for getting the right answer.
- The distinction between Kruskal’s and Prim’s algorithms.
- How to apply Prim’s algorithm to both a network and a table correctly.
- That Prim’s and Kruskal’s algorithms will usually give the same MST but often select the edges in a different order. Make sure you show sufficient working so that the examiner can see which algorithm you have used.
- How to work with networks or tables and be able to convert between the two.
- That you must always show all the working values as well as the permanent labels when using Dijkstra’s algorithm.

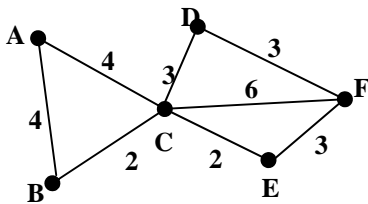
### The main ideas in this topic are

- Applying Kruskal’s and Prim’s Algorithms to find the minimum spanning tree of a network.
- Applying Dijkstra’s Algorithm to find the shortest (or least value) path from one vertex to any other vertex in the network.

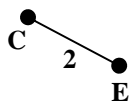
### Minimum Spanning Tree

The minimum connector problem is to make a selection of the available edges so that any one vertex can be reached from any other, and the total length of the chosen edges is as small as possible. A connected set of edges with no loops is called a *tree* and the set which solves the minimum connector problem is the *minimum spanning tree* for the network.

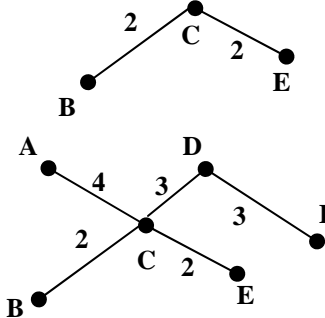
#### Kruskal’s Algorithm



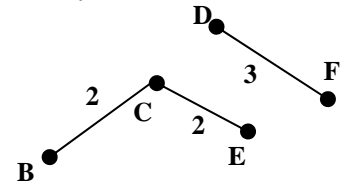
1. Choose the shortest edge (if there is more than one, choose any of the shortest)...



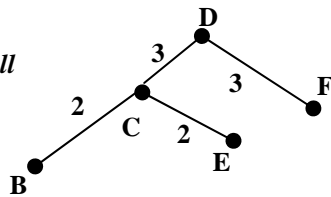
2. Choose the next shortest edge in the network (it doesn't have to be joined to the edges already)...



3. Choose the next shortest edge which does not create a cycle and add it...



4. Repeat step 3 until all the vertices are connected then stop.



Length of minimum spanning tree: 14

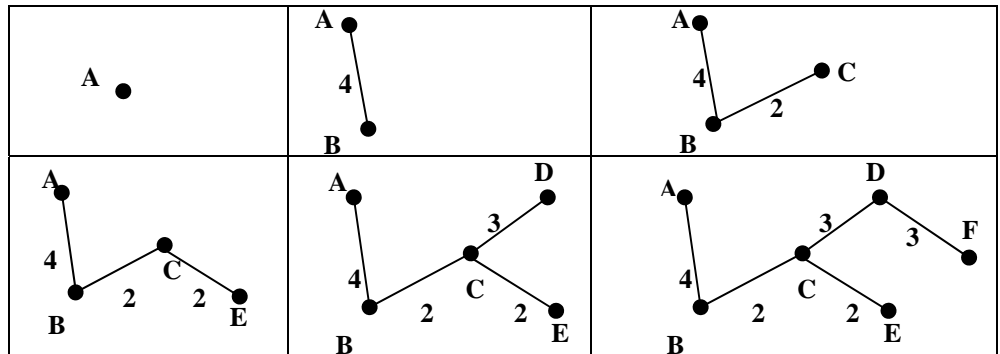
#### Prim’s Algorithm on a network

1. Choose a vertex...

2. Choose the shortest edge from this vertex to any vertex connected directly to it...

3. Choose the nearest vertex not yet in the solution which is connected to any vertex which is in the solution and which does not create a cycle...

4. Repeat step 3 until all the vertices are connected then stop.



### Prim's Algorithm on a Table

1. Choose a column and cross out its row. Here D has been chosen. Delete row D.
2. Choose the smallest number in the column D and circle it. If there is a choice, choose either.
3. For the number you have just circled, cross out its row and put an arrow above its row at the top of the table.
4. Choose the smallest number not already crossed out from the arrowed columns and circle it.
5. For the number you have just circled, cross out its row
6. and put an arrow above it's row at the top of the table.
7. Continue till all vertices have been included in the tree.

		1						
	A	B	C	D	E	F		
A	-	4	4					
B	4	-	2					
C	4	2	-	3	2	6		
D				3		3		
E					2	-	3	
F					6	3	3	-

		2	1					
	A	B	C	D	E	F		
A	-	4	4					
B	4	-	2					
C	4	2	-	3	2	6		
D				3		3		
E					2	-	3	
F					6	3	3	-

		3	2	1				
	A	B	C	D	E	F		
A	-	4	4					
B	4	-	2					
C	4	2	-	3	2	6		
D				3		3		
E					2	-	3	
F					6	3	3	-

		3	2	1	4			
	A	B	C	D	E	F		
A	-	4	4					
B	4	-	2					
C	4	2	-	3	2	6		
D				3		3		
E					2	-	3	
F					6	3	3	-

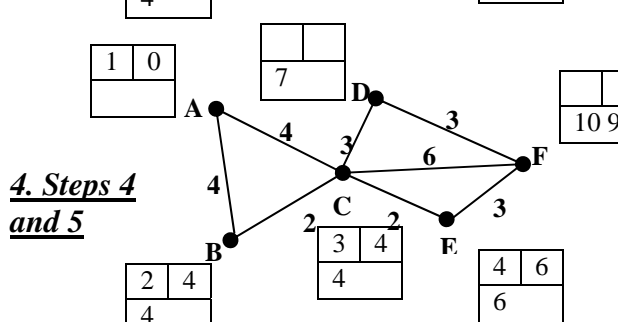
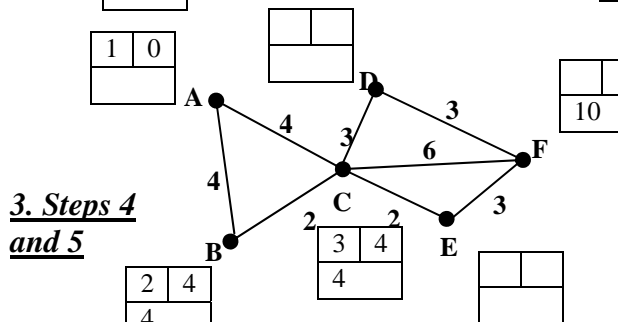
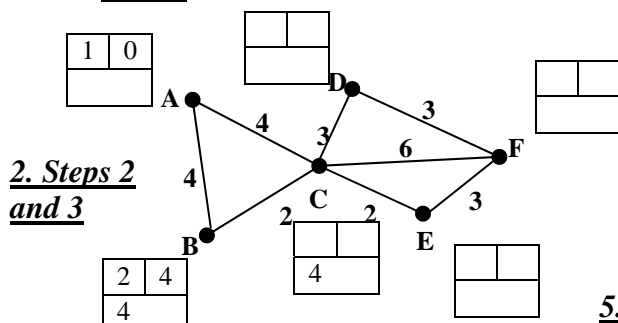
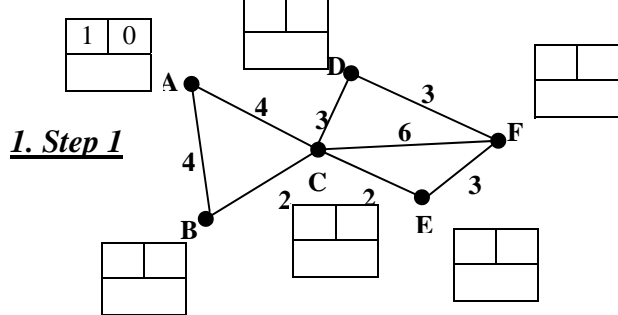
		3	2	1	4	5		
	A	B	C	D	E	F		
A	-	4	4					
B	4	-	2					
C	4	2	-	3	2	6		
D				3		3		
E					2	-	3	
F					6	3	3	-

		6	3	2	1	4	5		
	A	B	C	D	E	F			
A	-	4	4						
B	4	-	2						
C	4	2	-	3	2	6			
D				3		3			
E					2	-	3		
F					6	3	3	-	

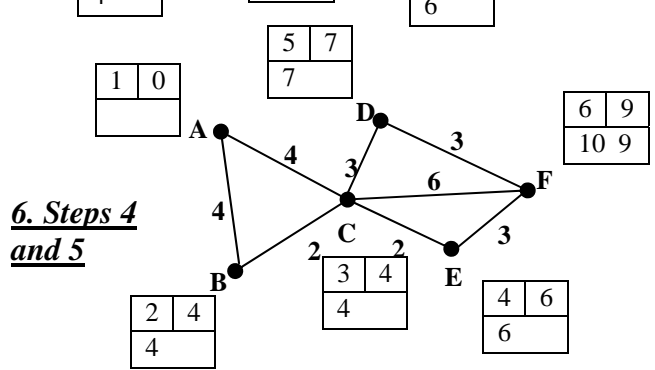
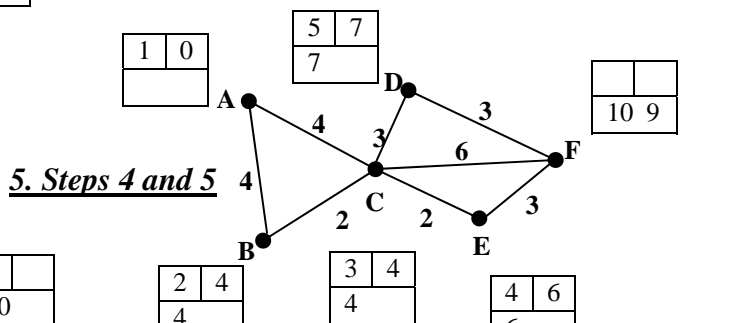
Length of minimum spanning tree 14



### Dijkstra's Algorithm for the shortest path



1. Label the start vertex with permanent label 0 and order label 1.
2. Assign temporary labels to all the vertices that can be reached directly from the start.
3. Select the vertex with the smallest temporary label and make its label permanent. Add the correct order label.
4. Put temporary labels on each vertex that can be reached directly from the vertex you have just made permanent. The temporary label must be equal to the sum of the permanent label and the direct distance from it. If there is an existing temporary label at a vertex, it should be replaced only if the new sum is smaller.
5. Repeat steps 3 and 4 until the finishing vertex has a permanent label.
6. To find the shortest paths(s), trace back from the end vertex to the start vertex. Write the route forwards and state the length.



**7. Step 6**  
**Solution:**  
 Shortest path ACEF  
 Length 9



# REVISION SHEET – DECISION MATHS

## Networks: Route Inspection

The main ideas are covered in

<b>AQA</b>	<b>Edexcel</b>	<b>MEI</b>	<b>OCR</b>
<b>D1</b>	<b>D1</b>	<b>D2</b>	<b>D1</b>

The main ideas in this topic are

An Eulerian network has all its vertices even.

Apply the Chinese Postman Algorithm to obtain the closed trail of minimum weight.

Before the exam you should know

- That an Euler cycle is a tour which travels along every edge of a network.
- The meaning of: order of a vertex (node), traversable graph and Eulerian graph.
- That the direct route between two vertices is not always the shortest.
- That you need to identify ALL the odd vertices in the route inspection problem.

### The Route Inspection Problem

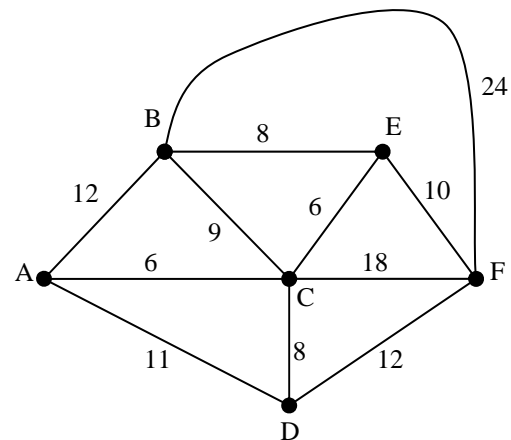
The problem is to find a route of minimum length which goes along each edge in the network once and returns to the starting point. This type of problem arises in contexts such as a rail safety expert needing to inspect every piece of track in a railway system, or a postman needing to walk along every street to deliver mail in the most efficient way possible, hence it is often called the **Chinese Postman problem** because a Chinese mathematician developed the algorithm. For a network to be traversable it must be Eulerian (no odd nodes) or semi-Eulerian (two odd nodes). A network will always have an even number of odd nodes (handshaking theorem). If the network is Eulerian (every vertex is of even order) there are many equal optimum solutions.

The Algorithm can be stated as follows

1. Identify the odd vertices in the network.
2. Consider all the routes joining pairs of odd vertices and select the one with the least weight.
3. Find the sum of the weights on all the edges.
4. Shortest distance is the sum of the weights plus the extra that must be traveled.
5. Find a tour which repeats the edges found in step 2.

**Example:** For the network shown below

Find the length of the shortest closed trail that covers every edge on the network below and write down a suitable route



**Solution:**

(a) Odd vertices are A, C, D and E.

Consider all the possible pairings of odd vertices:

AC = 6	and DE = 14	total = 20
AD = 11	and CE = 6	total = 17
AE = 12	and CD = 8	total = 20

The pairing of least weight is AD and CE = 17.

The sum of the weights in the network is 124.

Repeating AD and CE gives a total weight = 124 + 17 = 141.

A suitable route is A – B – E – F – D – A – C – B – F – C – E – C – D – A.