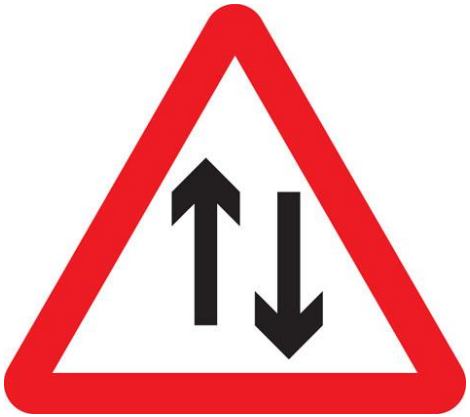
Floyd's algorithm



Shortest Paths – beyond Dijkstra

- Single Source to all other vertices
- Dijkstra's algorithm
- Bellman Ford Algorithm and Dynamic Programming (to handle negative weights)
- What happens if you want to find **all** the shortest paths in a network?
- Floyd-Warshall algorithm

Floyd-Warshall algorithm

- The Floyd–Warshall algorithm is an example of dynamic programming.
- It was published in its current form by Robert Floyd in 1962.
- It is essentially the same as algorithms previously published by Bernard Roy in 1959 and by Stephen Warshall in 1962.

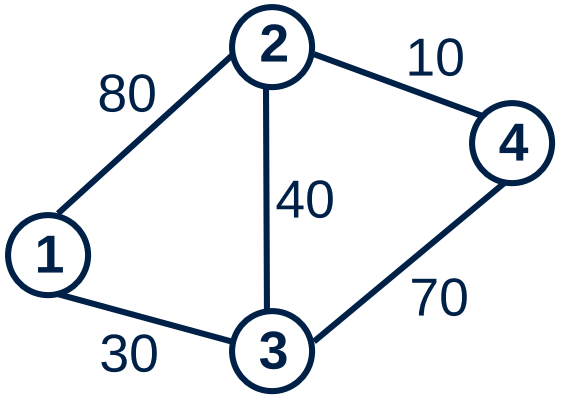
Professor Robert Floyd (1936 – 2001)
Stanford University, computer science

Floyd's algorithm

Floyd's Algorithm finds the shortest path between **every pair** of vertices in the Network.

Example:

Find the shortest paths between each pair of vertices in the following network:



Floyd's algorithm – two versions!

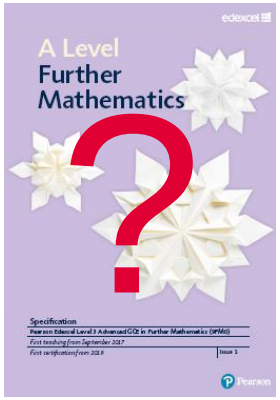
Floyd's Algorithm was on the MEI specification before the new 2017 specifications.

INTEGRAL resources also included it in the same standard form.

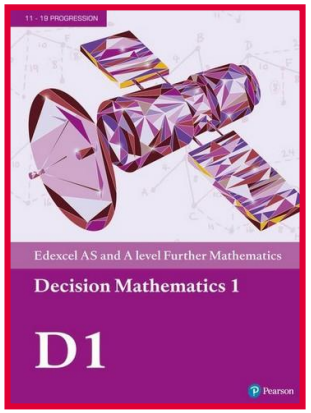
The Pearson Edexcel text book contains a new variant of Floyd's Algorithm which uses a different process for updating the route matrix and reading off the final route.

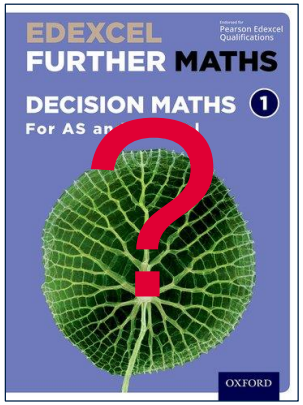
Floyd's algorithm – two versions!

Floyd's Algorithm: Edexcel variant

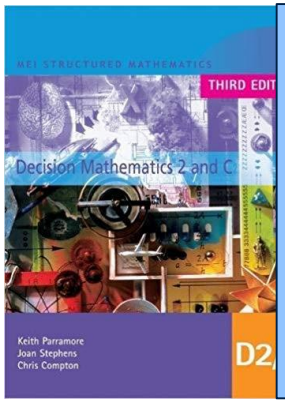


Edexcel SAMs



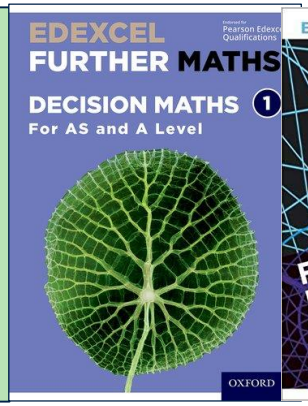


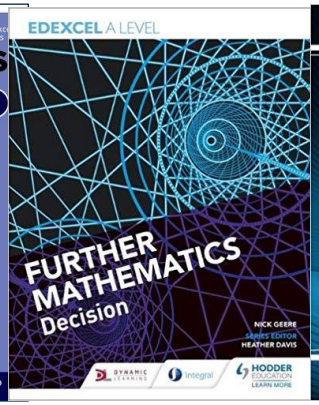
Floyd's Algorithm: standard version

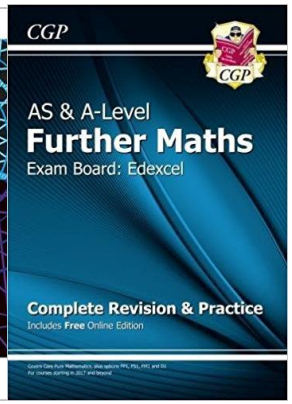


Integral Maths online resources

Edexcel SAMs



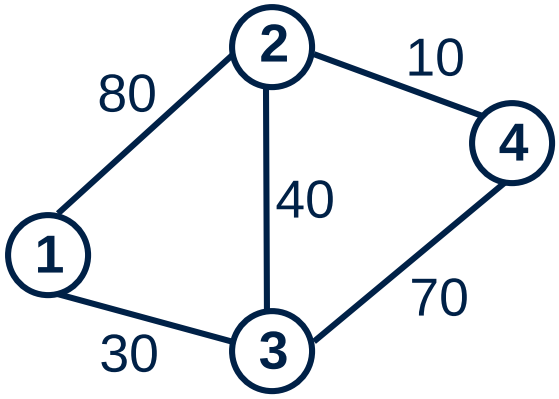




Distance matrix

	1	2	3	4
1	∞	80	30	∞
2	80	∞	40	10
3	30	40	∞	70
4	∞	10	70	∞

Both versions treat the distance matrix the same way, but Edexcel has dashes on the leading diagonal.



Distance and Route matrices

Initial distance matrix

	1	2	3	4
1	∞	80	30	∞
2	80	∞	40	10
3	30	40	∞	70
4	∞	10	70	∞

Initial route matrix

	1	2	3	4
1	1	2	3	4
2	1	2	3	4
3	1	2	3	4
4	1	2	3	4

The main difference occurs in the **route matrix**.

Floyd's Algorithm: standard method

Key point

Floyd's algorithm

For a network with n vertices, produce $n \times n$ distance and route matrices D_0 and R_0

Step 1 Take $k = 1$ and A as the 'active' vertex V

Step 2 Highlight the row and column for V. These values will be unaffected.

Step 3 For all vertices X, $Y \neq V$

If $XV + VY < XY$, then:

- replace row X, column Y of D with $(XV + VY)$
- replace row X, column Y of R with the value from row X, column V.

You now have new distance and route matrices D_k and R_k.

Step 4 Stop if $k = n$, otherwise let $k = k + 1$, choose a new vertex V and repeat from Step 2

Floyd's Algorithm: standard method

Key point

Floyd's algorithm

For a network with n vertices, produce $n \times n$ distance and route matrices D_0 and R_0

Step 1 Take $k = 1$ and A as the 'active' vertex V

Step 2 Highlight the row and column for V. These values will be unaffected.

Step 3 For all vertices X, $Y \neq V$

If $XV + VY < XY$, then:

- replace row X, column Y of D with $(XV + VY)$
- replace row X, column Y of R with the value from row X, column V.

You now have new distance and route matrices D_k and R_k.

Step 4 Stop if $k = n$, otherwise let $k = k + 1$, choose a new vertex V and repeat from Step 2

EDEXCEL FURTHER MATHS: Decision Maths for AS and A Level

Oxford University Press

Distance and Route matrices

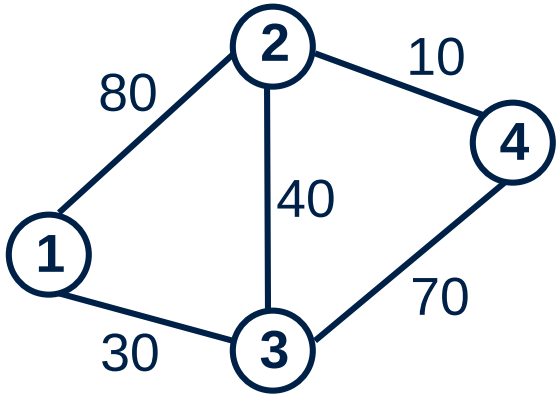
D(0)

	1	2	3	4
1	∞	80	30	∞
2	80	∞	40	10
3	30	40	∞	70
4	∞	10	70	∞

R(0)

	1	2	3	4
1	1	2	3	4
2	1	2	3	4
3	1	2	3	4
4	1	2	3	4

First iteration:



Highlight the 1st row and column.

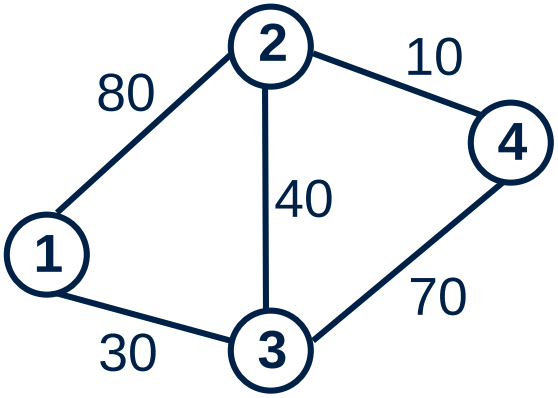
D(0)

	1	2	3	4
1	∞	80	30	∞
2	80	∞	40	10
3	30	40	∞	70
4	∞	10	70	∞

R(0)

	1	2	3	4
1	1	2	3	4
2	1	2	3	4
3	1	2	3	4
4	1	2	3	4

First iteration:



80+30 > 40
don't replace

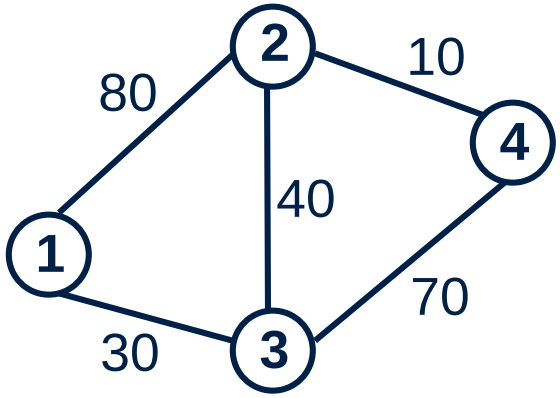
	1	2	3	4
1	∞	80	30	∞
2	80	**160**	40	10
3	30	40	**60**	70
4	∞	10	70	∞

80+80 < ∞
replace with 160

	1	2	3	4
1	1	2	3	4
2	1	**1**	3	4
3	1	2	**1**	4
4	1	2	3	4

replace with the row entry in the highlighted column

After first iteration:



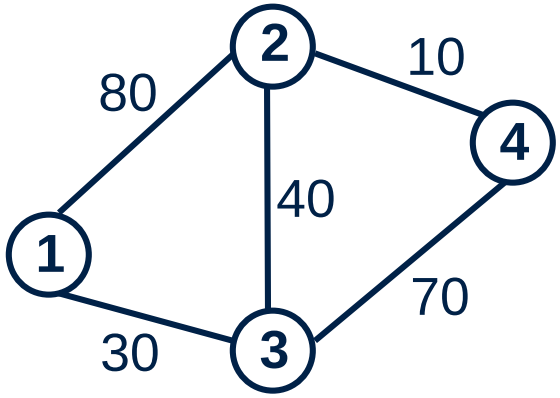
D(1)

	1	2	3	4
1	∞	80	30	∞
2	80	160	40	10
3	30	40	60	70
4	∞	10	70	∞

R(1)

	1	2	3	4
1	1	2	3	4
2	1	1	3	4
3	1	2	1	4
4	1	2	3	4

Second iteration:



Highlight the second row and column.

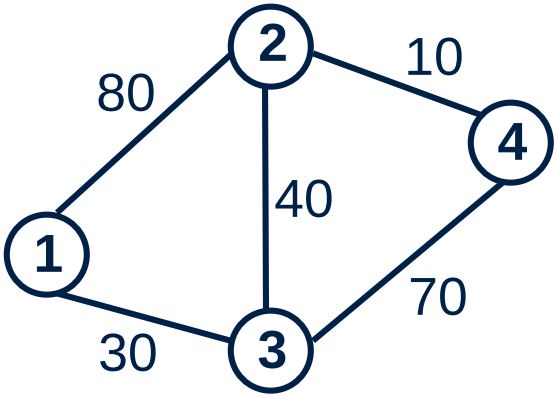
D(1)

	1	2	3	4
1	∞	80	30	∞
2	80	160	40	10
3	30	40	60	70
4	∞	10	70	∞

R(1)

	1	2	3	4
1	1	2	3	4
2	1	1	3	4
3	1	2	1	4
4	1	2	3	4

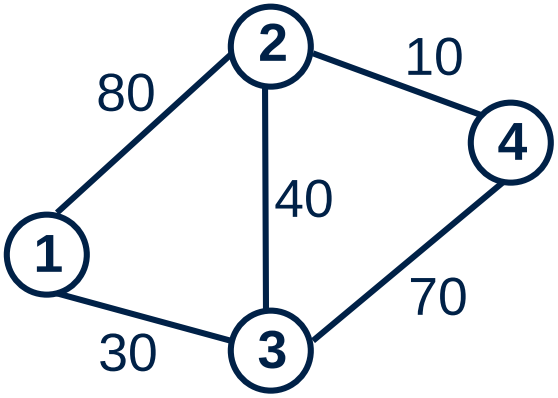
Second iteration:



	1	2	3	4
1	**160**	80	30	**90**
2	80	160	40	10
3	30	40	60	**50**
4	**90**	10	**50**	**20**

	1	2	3	4
1	**2**	2	3	**2**
2	1	1	3	4
3	1	2	1	**2**
4	**2**	2	**2**	**2**

After second iteration:



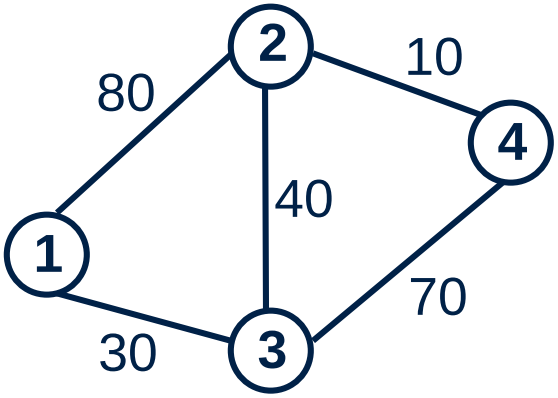
D(2)

	1	2	3	4
1	160	80	30	90
2	80	160	40	10
3	30	40	60	50
4	90	10	50	20

R(2)

	1	2	3	4
1	2	2	3	2
2	1	1	3	4
3	1	2	1	2
4	2	2	2	2

Third iteration:



Highlight the third row and column.

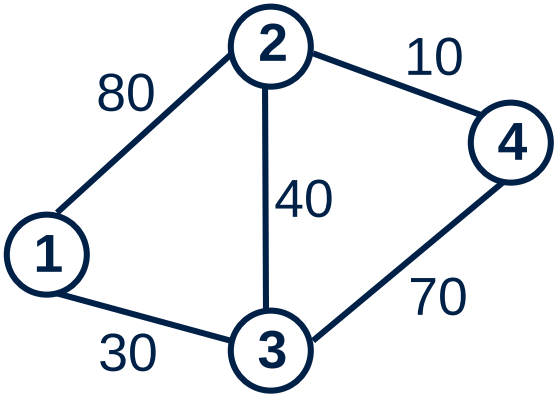
D(2)

	1	2	3	4
1	160	80	30	90
2	80	160	40	10
3	30	40	60	50
4	90	10	50	20

R(2)

	1	2	3	4
1	2	2	3	2
2	1	1	3	4
3	1	2	1	2
4	2	2	2	2

Third iteration:

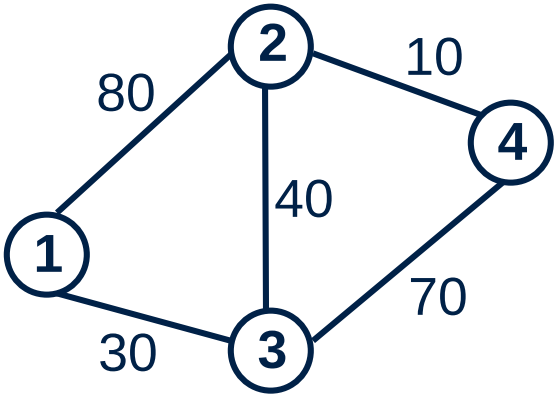


	1	2	3	4
1	60	70	30	80
2	70	80	40	10
3	30	40	60	50
4	80	10	50	20

	1	2	3	4
1	3	3	3	3
2	3	3	3	4
3	1	2	1	2
4	2	2	2	2

Note: the variant method does this step differently

After third iteration:



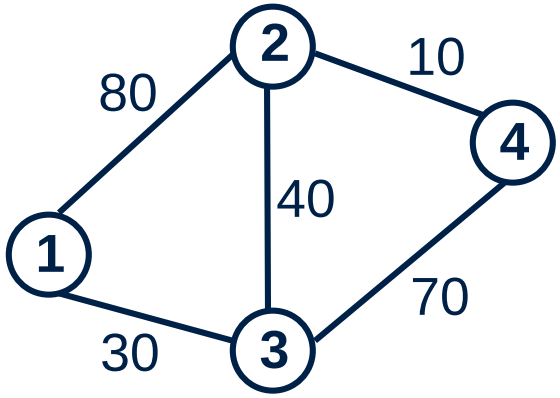
D(3)

	1	2	3	4
1	60	70	30	80
2	70	80	40	10
3	30	40	60	50
4	80	10	50	20

R(3)

	1	2	3	4
1	3	3	3	3
2	3	3	3	4
3	1	2	1	2
4	2	2	2	2

Fourth iteration:



Highlight the fourth row and column.

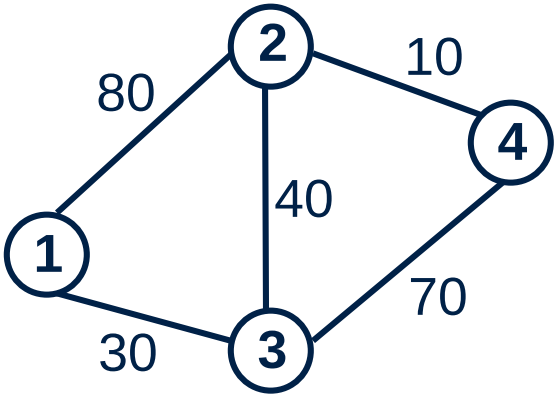
D(3)

	1	2	3	4
1	60	70	30	80
2	70	80	40	10
3	30	40	60	50
4	80	10	50	20

R(3)

	1	2	3	4
1	3	3	3	3
2	3	3	3	4
3	1	2	1	2
4	2	2	2	2

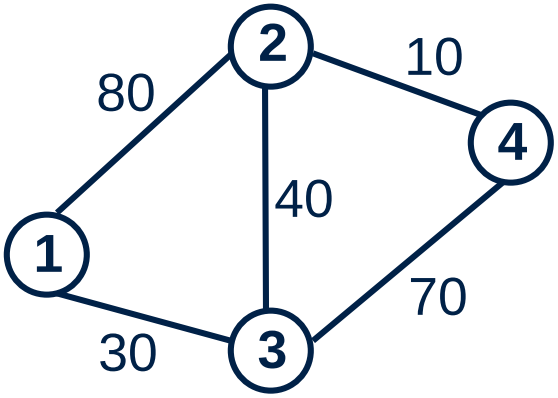
Fourth iteration:



	1	2	3	4
1	60	70	30	80
2	70	**20**	40	10
3	30	40	60	50
4	80	10	50	20

	1	2	3	4
1	3	3	3	3
2	3	**4**	3	4
3	1	2	1	2
4	2	2	2	2

After fourth iteration:



D(4)

	1	2	3	4
1	60	70	30	80
2	70	20	40	10
3	30	40	60	50
4	80	10	50	20

R(4)

	1	2	3	4
1	3	3	3	3
2	3	4	3	4
3	1	2	1	2
4	2	2	2	2

Using the matrices: standard version

Find the shortest route from 1 to 4.

D(4)

	1	2	3	4
1	60	70	30	80
2	70	20	40	10
3	30	40	60	50
4	80	10	50	20

R(4)

	1	2	3	4
1	~~3~~	~~3~~	~~3~~	3
2	3	4	3	4
3	1	2	1	2
4	2	2	2	2

Row 1 to col 4 gives 3, so route directly from 1 to 3;

so 1-3 …

Using the matrices: standard version

Find the shortest route from 1 to 4.

D(4)

	1	2	3	4
1	60	70	30	80
2	70	20	40	10
3	30	40	60	50
4	80	10	50	20

R(4)

	1	2	3	4
1	3	3	3	3
2	3	4	3	4
3	1	2	1	2
4	2	2	2	2

Check row 3 to column 4, 3 goes directly to 2;

so 1-3-2 …

Using the matrices: standard version

Find the shortest route from 1 to 4.

D(4)

	1	2	3	4
1	60	70	30	80
2	70	20	40	10
3	30	40	60	50
4	80	10	50	20

R(4)

	1	2	3	4
1	3	3	3	3
2	3	4	3	4
3	1	2	1	2
4	2	2	2	2

Check row 2 to column 4, 2 goes directly to 4;

so 1-3-2-4

Using the matrices: standard version

Find the shortest route from 1 to 4.

D(4)

	1	2	3	4
1	60	70	30	80
2	70	20	40	10
3	30	40	60	50
4	80	10	50	20

R(4)

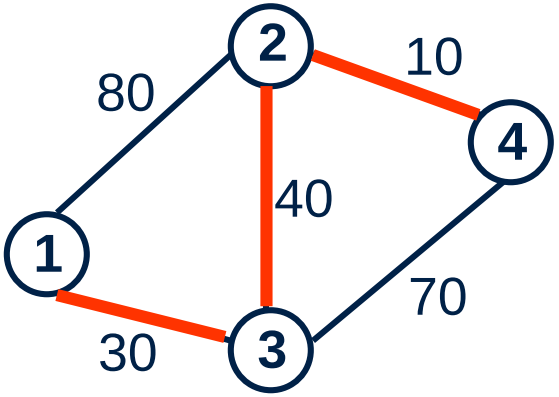
	1	2	3	4
1	3	3	3	3
2	3	4	3	4
3	1	2	1	2
4	2	2	2	2

The shortest route from 1 to 4 is 80 by going: 1-3-2-4

Using the matrices: standard version

	1	2	3	4
1	60	70	30	**80**
2	70	20	40	10
3	30	40	60	50
4	80	10	50	20

	1	2	3	4
1	3	3	3	3
2	3	4	3	4
3	1	2	1	2
4	2	2	2	2



The shortest route from 1 to 4 is 80 by going: 1-3-2-4

Floyd's Algorithm: Edexcel variant

1 Complete an initial **distance table** for the network. If there is no direct route from one vertex to another, label the distance with the infinity symbol.

2 Complete an initial **route table** by making every entry in the first column the same as the label at the top of the first column, making every entry in the second column the same as the label at the top of the second column and so on.

3 In the first iteration, copy the first row and the first column values of the distance table into a new table. Lightly shade these values.

	A	*B*	*C*	*D*
A	–			
B		–		
C			–	
D				–

Floyd's Algorithm: Edexcel variant

4 Consider each unshaded position in turn. Compare the value in this position in the previous table with the sum of the corresponding shaded values.

Distance table

	A	B	C	D
A	–		Y	
B	X	–	Z	
C			–	
D				–

Route table

	A	B	C	D
A	A	B	C	D
B	A	B	[A]	D
C	A	B	C	D
D	A	B	C	D

Watch out You compare Z with the sum of the shaded values in the **same row and column**. You are always comparing an unshaded value with the sum of two shaded values.

If $X + Y \geqslant Z$ then copy Z into the new table, i.e. there is no change – the object is to keep the smallest values in the table.

If $X + Y < Z$, copy $X + Y$ into the new table and write A in the corresponding position in the route table. Once all values in the unshaded area have been considered, the first iteration is complete.

A is written into the route table to show that the direct distance BC has been replaced by $BA + AC$, i.e. the route has taken a detour through A. Any other changes in this iteration will also result in a detour through A and so A is written in the route table, in each case, in the position corresponding to the changed value.

Floyd's Algorithm: Edexcel variant

5 For the second iteration, copy the second row and the second column from the last iteration into a new distance table. Lightly shade these values.

	A	*B*	*C*	*D*
A	–			
B		–		
C			–	
D				–

6 Repeat step **4** with the new unshaded positions. This time any changes will result in a detour through *B* and so you should write *B* in the new route table, in each case, in the position corresponding to the changed value.

7 If there are *n* vertices then completing the algorithm will require *n* iterations continuing in the same way.

EDEXCEL FURTHER MATHS: Decision Maths for AS and A Level

Pearson

Distance and Route matrices

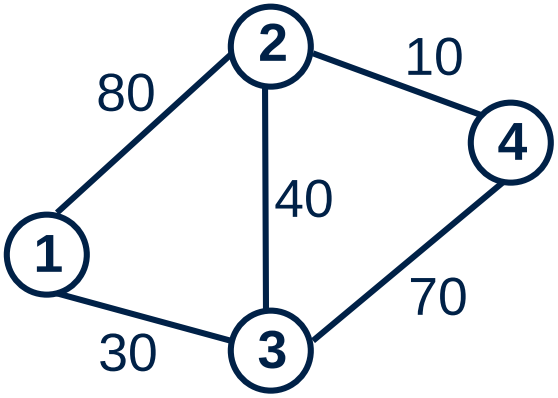
D(0)

	1	2	3	4
1	-	80	30	∞
2	80	-	40	10
3	30	40	-	70
4	∞	10	70	-

R(0)

	1	2	3	4
1	1	2	3	4
2	1	2	3	4
3	1	2	3	4
4	1	2	3	4

First iteration:



Highlight the 1st row and column.

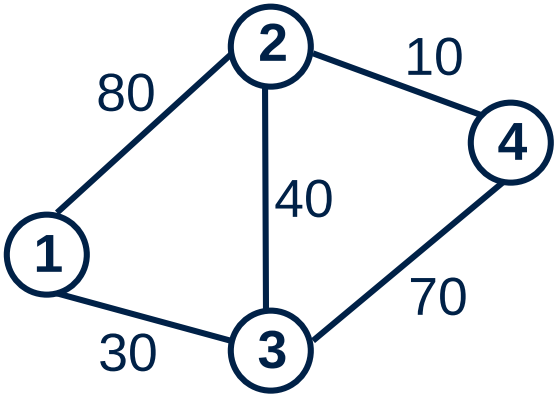
D(0)

	1	2	3	4
1	-	80	30	∞
2	80	-	40	10
3	30	40	-	70
4	∞	10	70	-

R(0)

	1	2	3	4
1	1	2	3	4
2	1	2	3	4
3	1	2	3	4
4	1	2	3	4

First iteration:



80+30 > 40
don't replace

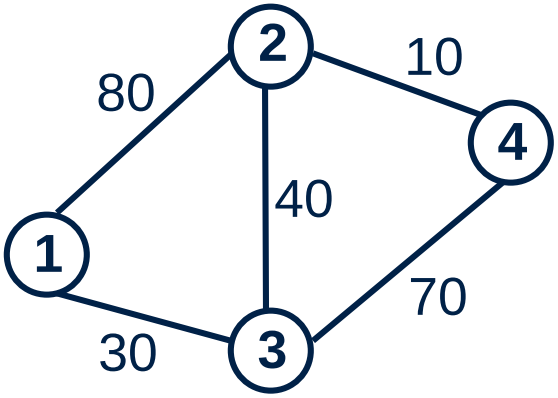
	1	2	3	4
1	-	80	30	∞
2	80	-	40	10
3	30	40	-	70
4	∞	10	70	-

80+80 > "-"
don't replace

	1	2	3	4
1	1	2	3	4
2	1	2	3	4
3	1	2	3	4
4	1	2	3	4

No changes needed
to route matrix

After first iteration:



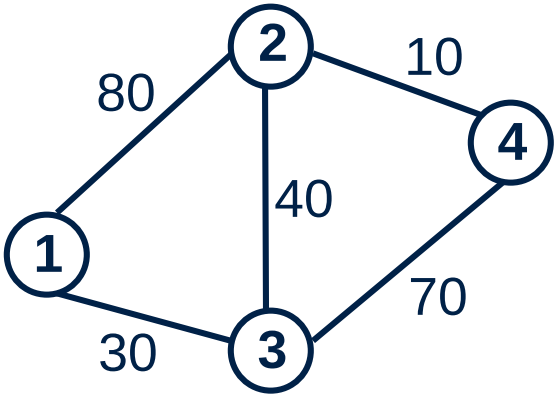
D(1)

	1	2	3	4
1	-	80	30	∞
2	80	-	40	10
3	30	40	-	70
4	∞	10	70	-

R(1)

	1	2	3	4
1	1	2	3	4
2	1	2	3	4
3	1	2	3	4
4	1	2	3	4

Second iteration:



Highlight the second row and column.

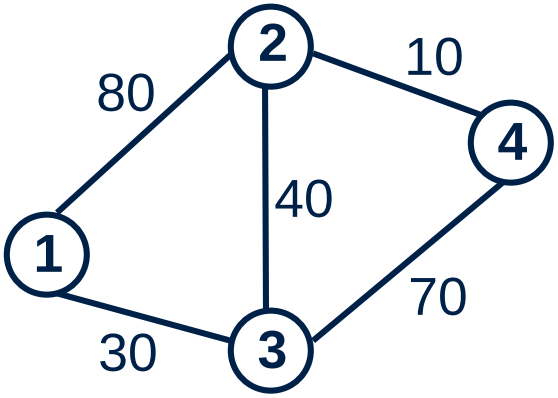
D(1)

	1	2	3	4
1	-	80	30	∞
2	80	-	40	10
3	30	40	-	70
4	∞	10	70	-

R(1)

	1	2	3	4
1	1	2	3	4
2	1	2	3	4
3	1	2	3	4
4	1	2	3	4

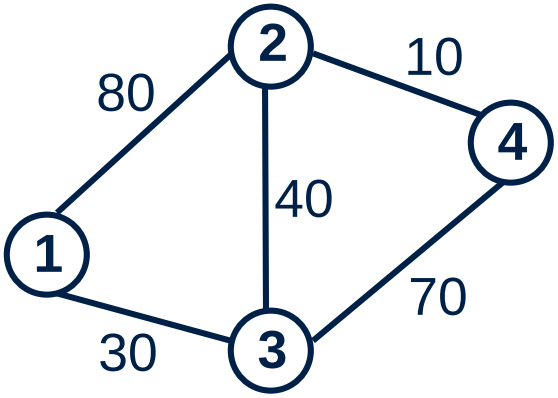
Second iteration:



	1	2	3	4
1	-	80	30	**90**
2	80	-	40	10
3	30	40	-	**50**
4	**90**	10	**50**	-

	1	2	3	4
1	1	2	3	**2**
2	1	2	3	4
3	1	2	3	**2**
4	**2**	2	**2**	4

After second iteration:



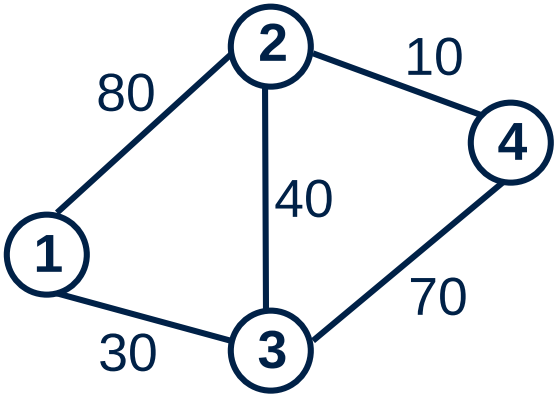
D(2)

	1	2	3	4
1	-	80	30	90
2	80	-	40	10
3	30	40	-	50
4	90	10	50	-

R(2)

	1	2	3	4
1	1	2	3	2
2	1	2	3	4
3	1	2	3	2
4	2	2	2	4

Third iteration:



Highlight the third row and column.

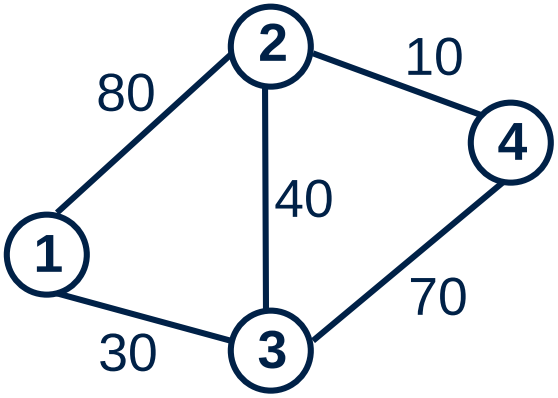
D(2)

	1	2	3	4
1	-	80	30	90
2	80	-	40	10
3	30	40	-	50
4	90	10	50	-

R(2)

	1	2	3	4
1	1	2	3	2
2	1	2	3	4
3	1	2	3	2
4	2	2	2	4

Third iteration:

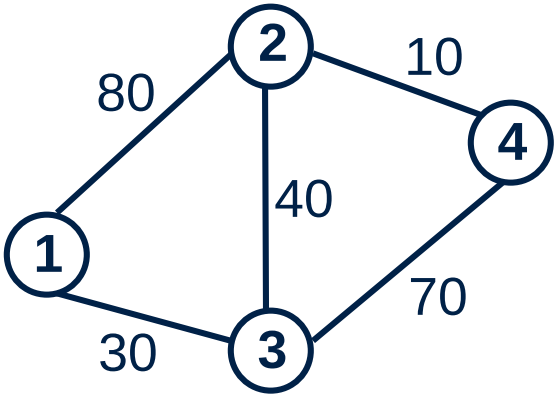


	1	2	3	4
1	-	70	30	80
2	70	-	40	10
3	30	40	-	50
4	80	10	50	-

	1	2	3	4
1	1	3	3	3
2	3	2	3	4
3	1	2	3	2
4	3	2	2	4

Note: the standard method would do this step differently

After third iteration:



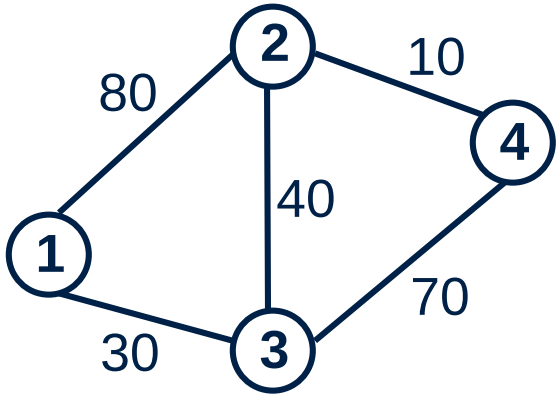
D(3)

	1	2	3	4
1	-	70	30	80
2	70	-	40	10
3	30	40	-	50
4	80	10	50	-

R(3)

	1	2	3	4
1	1	3	3	3
2	3	2	3	4
3	1	2	3	2
4	3	2	2	4

Fourth iteration:



Highlight the fourth row and column.

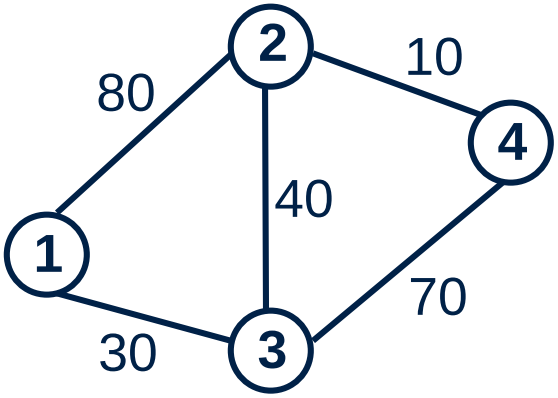
D(3)

	1	2	3	4
1	-	70	30	80
2	70	-	40	10
3	30	40	-	50
4	80	10	50	-

R(3)

	1	2	3	4
1	1	3	3	3
2	3	2	3	4
3	1	2	3	2
4	3	2	2	4

Fourth iteration:



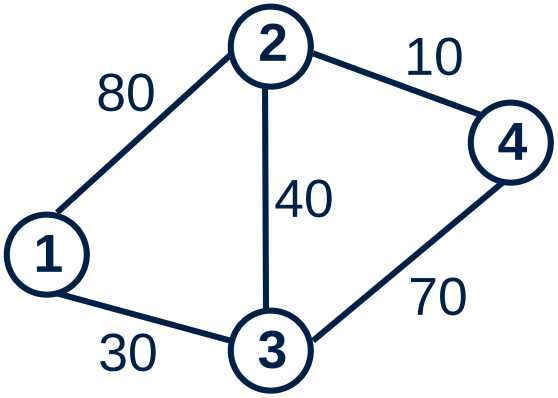
Check all unhighlighted cells.

	1	2	3	4
1	-	70	30	80
2	70	-	40	10
3	30	40	-	50
4	80	10	50	-

	1	2	3	4
1	1	3	3	3
2	3	2	3	4
3	1	2	3	2
4	3	2	2	4

(There are no further changes.)

After fourth iteration:



D(4)

	1	2	3	4
1	-	70	30	80
2	70	-	40	10
3	30	40	-	50
4	80	10	50	-

R(4)

	1	2	3	4
1	1	3	3	3
2	3	2	3	4
3	1	2	3	2
4	3	2	2	4

Using the matrices: Edexcel version

Find the shortest route from 1 to 4.

D(4)

	1	2	3	4
1	-	70	30	80
2	70	-	40	10
3	30	40	-	50
4	80	10	50	-

R(4)

	1	2	3	4
1	1	3	3	3
2	3	2	3	4
3	1	2	3	2
4	3	2	2	4

Row 1 to col 4 gives 3, so route from 1 to 4 goes *via 3*;

so 1…3…4

Using the matrices: Edexcel version

Find the shortest route from 1 to 4.

D(4)

	1	2	3	4
1	-	70	30	80
2	70	-	40	10
3	30	40	-	50
4	80	10	50	-

R(4)

	1	2	3	4
1	1	3	3	3
2	3	2	3	4
3	1	2	3	2
4	3	2	2	4

Check row 3 to column 4, 3 to 4 goes via 2;

so 1…3…2…4

Using the matrices: Edexcel version

Find the shortest route from 1 to 4.

D(4)

	1	2	3	4
1	-	70	30	80
2	70	-	40	10
3	30	40	-	50
4	80	10	50	-

R(4)

	1	2	3	4
1	1	3	3	3
2	3	2	3	4
3	1	2	3	2
4	3	2	2	4

Check row 2 to column 4, 2 to 4 goes via 4 (i.e. direct).

The shortest route: 1…3…2…4

Using the matrices: Edexcel version

Find the shortest route from 1 to 4.

D(4)

	1	2	3	4
1	-	70	30	80
2	70	-	40	10
3	30	40	-	50
4	80	10	50	-

R(4)

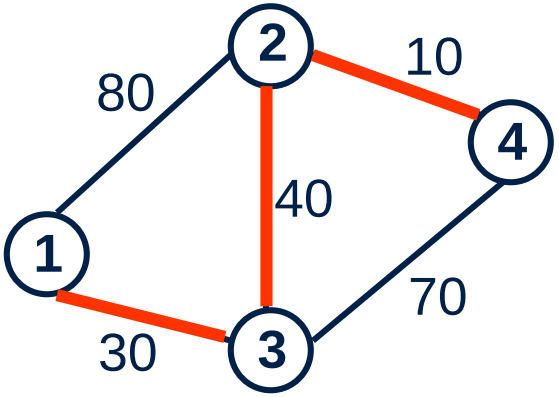
	1	2	3	4
1	1	3	3	3
2	3	2	3	4
3	1	2	3	2
4	3	2	2	4

The shortest route from 1 to 4 is 80 by going: 1-3-2-4

Using the matrices: standard version

	1	2	3	4
1	-	70	30	80
2	70	-	40	10
3	30	40	-	50
4	80	10	50	-

	1	2	3	4
1	1	3	3	3
2	3	2	3	4
3	1	2	3	2
4	3	2	2	4



The shortest route from 1 to 4 is 80 by going: 1-3-2-4

Activity: Floyd's Algorithm card sort

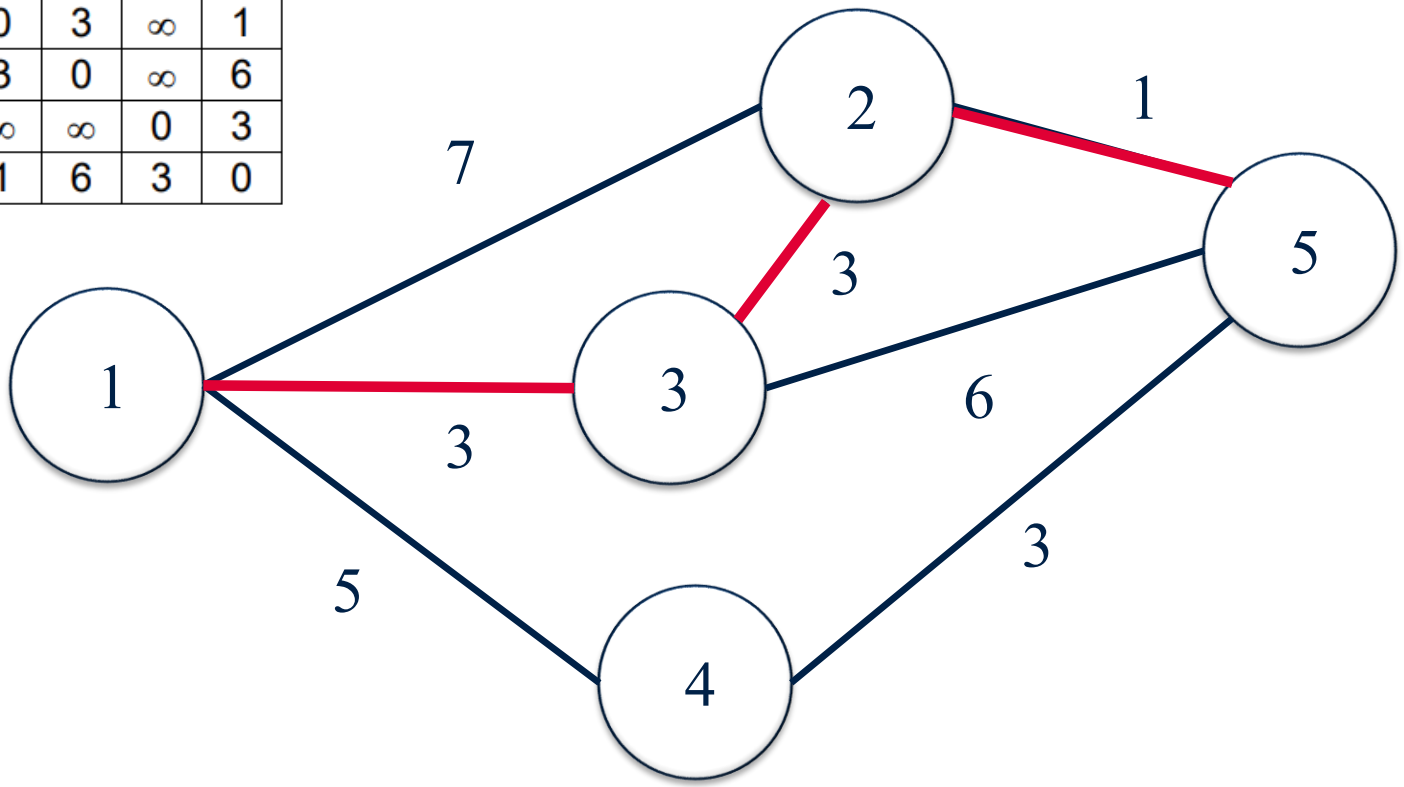
Each set is a fully worked solution to Floyd's algorithm.

- Draw the network that goes with the initial problem
- Put the cards in order to show the solution to the problem
- Give the shortest distance and route from vertex 1 to vertex 5
- Check your solution against your network

Initial Distance Matrix: D(0)

	1	2	3	4	5
1	0	7	3	5	∞
2	7	0	3	∞	1
3	3	3	0	∞	6
4	5	∞	∞	0	3
5	∞	1	6	3	0

Solution



Shortest distance from 1 to 5 is 7

Route: 1 – 3 – 2 – 5

amsp

SAMs

4.

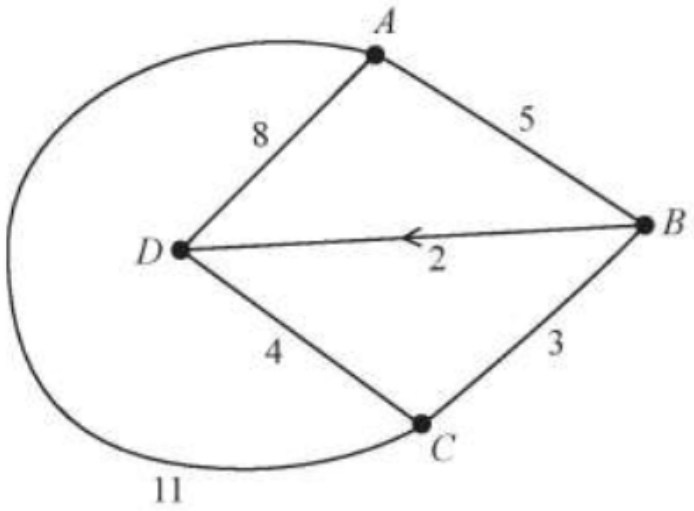


Figure 3

The network in Figure 3 shows the roads linking a depot, D, and three collection points A, B and C. The number on each arc represents the length, in miles, of the corresponding road. The road from B to D is a one-way road, as indicated by the arrow.

(a) Explain clearly why Dijkstra's algorithm cannot be used to find a route from D to A.

(1)

The initial distance and route tables for the network are given in the answer book.

(b) Use Floyd's algorithm to find a table of least distances. You should show both the distance table and the route table after each iteration.

(7)

(c) Explain how the final route table can be used to find the shortest route from D to B. State this route.

SAMs

"Explain how the final route table can be used to find the shortest route from D to B."

	A	**B**	**C**	**D**
A	*A*	*B*	*B*	*B*
B	*A*	*B*	*C*	*D*
C	*B*	*B*	*C*	*D*
D	*A*	*C*	*C*	*D*

Question	Scheme	Marks	AOs
4(a)	Yes Dijkstra's algorithm can be applied to either a directed or undirected network	B1	3.5b
		(1)	
(b)	Initial tables $\begin{bmatrix} - & 5 & 11 & 8 \\ 5 & - & 3 & 2 \\ 11 & 3 & - & 4 \\ 8 & \infty & 4 & - \end{bmatrix}$ $\begin{bmatrix} A & B & C & D \\ A & B & C & D \\ A & B & C & D \\ A & B & C & D \end{bmatrix}$		
	1st iteration $\begin{bmatrix} - & 5 & 11 & 8 \\ 5 & - & 3 & 2 \\ 11 & 3 & - & 4 \\ 8 & [13] & 4 & - \end{bmatrix}$ $\begin{bmatrix} A & B & C & D \\ A & B & C & D \\ A & B & C & D \\ A & [A] & C & D \end{bmatrix}$	M1 A1	1.1b 1.1b
	2nd iteration $\begin{bmatrix} - & 5 & [8] & [7] \\ 5 & - & 3 & 2 \\ [8] & 3 & - & 4 \\ 8 & 13 & 4 & - \end{bmatrix}$ $\begin{bmatrix} A & B & [B] & [B] \\ A & B & C & D \\ [B] & B & C & D \\ A & A & C & D \end{bmatrix}$	M1 A1ft	1.1b 1.1b
	3rd iteration $\begin{bmatrix} - & 5 & 8 & 7 \\ 5 & - & 3 & 2 \\ 8 & 3 & - & 4 \\ 8 & [7] & 4 & - \end{bmatrix}$ $\begin{bmatrix} A & B & B & B \\ A & B & C & D \\ B & B & C & D \\ A & [C] & C & D \end{bmatrix}$	M1 A1ft	1.1b 1.1b
	4th iteration $\begin{bmatrix} - & 5 & 8 & 7 \\ 5 & - & 3 & 2 \\ 8 & 3 & - & 4 \\ 8 & 7 & 4 & - \end{bmatrix}$ $\begin{bmatrix} A & B & B & B \\ A & B & C & D \\ B & B & C & D \\ A & C & C & D \end{bmatrix}$ no changes therefore optimal	A1	1.1b
		(7)	
(c)	Start at D (4th) row and read across to the B (2nd) column, there is a C there		
	so the route starts DC. Look at the C row, B column and you see B	B1	2.4
	The route is therefore DCB	B1	2.2a
		(2)	

SAMs

Question	Scheme	Marks	AOs
(d)	D – C –B – A – B – D	M1	2.2a
	Length 19 (miles)	A1	1.1b
		(2)	
(e)	Dijkstra's algorithm finds the shortest distances from **one** vertex to **all** the others. Floyd's algorithm finds the shortest distance between **every pair** of vertices.	B1 B1	2.5 2.5
		(2)	
			(14 marks)

Question 4 notes:

(a)

B1: cao (must include mention of 'directed' network)

M1: No change in the first row and first column of both tables with at least one value in the distance table reduced and one value in the route table changed

(b)

A1: cao

M1: No change in the second row and second column of both tables with at least two values in the distance table reduced and two values in the route table changed

A1ft: Correct second iteration follow through from the candidate's first iteration

M1: No change in the third row and third column of both tables with at least one value in the distance table reduced and one value in the route table changed

A1ft: Correct third iteration follow through from the candidate's second iteration

A1: cao (no change after the fourth iteration) – all previous marks must have been awarded in this part

(c)

B1: Clear indication of how the final route table can be used to get from D to B (therefore must mention the correct rows and columns in their reasoning)

B1: Completely correct argument + correct route (DCB)

(d)

M1: Deduce correctly their minimum route from their final distance table (dependent on all M marks in (a)) must begin and end at D

A1: cao (length of 19)

(e)

B1: cao – must use correct language '**one** vertex to **all** other vertices'

B1: cao – must use correct language '**every pair** of vertices'

Edexcel's Latest Announcement

"I can confirm that we will accept either variant of the method for Floyd's algorithm and that it will be possible to answer all exam questions using either variant, i.e. if we give a partially completed version of the algorithm, we will ensure that the part we have completed would be the same using either variant."

Rob Cackett, Pearson, 6th February 2019

Teaching Discrete Mathematics

Activity: Floyd's Algorithm card sort (Edexcel text book method)

Cut out the cards and place them in the correct order to give the solution for this problem.

Initial Distance Matrix: D(0)

	1	**2**	**3**	**4**	**5**
1	0	7	3	5	∞
2	7	0	3	∞	1
3	3	3	0	∞	6
4	5	∞	∞	0	3
5	∞	1	6	3	0

Initial Route Matrix: R(0)

	1	**2**	**3**	**4**	**5**
1	1	2	3	4	5
2	1	2	3	4	5
3	1	2	3	4	5
4	1	2	3	4	5
5	1	2	3	4	5

1	3	3	4	3
3	2	3	3	5
1	2	3	1	2
1	3	1	4	5
3	2	2	4	5

0	6	3	5	7
6	0	3	4	1
3	3	0	7	4
5	4	7	0	3
7	1	4	3	0

0	7	3	5	8
7	0	3	12	1
3	3	0	8	4
5	12	8	0	3
8	1	4	3	0

1	2	3	4	5
1	2	3	1	5
1	2	3	1	5
1	1	1	4	5
1	2	3	4	5

0	7	3	5	∞
7	0	3	12	1
3	3	0	8	6
5	12	8	0	3
∞	1	6	3	0

1	3	3	4	3
3	2	3	3	5
1	2	3	1	2
1	3	1	4	5
3	2	2	4	5

1	3	3	4	3
3	2	3	5	5
1	2	3	5	2
1	5	5	4	5
3	2	2	4	5

0	6	3	5	7
6	0	3	11	1
3	3	0	8	4
5	11	8	0	3
7	1	4	3	0

1	2	3	4	2
1	2	3	1	5
1	2	3	1	2
1	1	1	4	5
2	2	2	4	5

0	6	3	5	7
6	0	3	11	1
3	3	0	8	4
5	11	8	0	3
7	1	4	3	0

Teaching Discrete Mathematics

Activity: Floyd's Algorithm card sort solution (Edexcel text book method)

Initial Distance Matrix: D_0

	1	**2**	**3**	**4**	**5**
1	0	7	3	5	∞
2	7	0	3	∞	1
3	3	3	0	∞	6
4	5	∞	∞	0	3
5	∞	1	6	3	0

Initial Route Matrix: R_0

	1	**2**	**3**	**4**	**5**
1	1	2	3	4	5
2	1	2	3	4	5
3	1	2	3	4	5
4	1	2	3	4	5
5	1	2	3	4	5

D_1

0	7	3	5	∞
7	0	3	12	1
3	3	0	8	6
5	12	8	0	3
∞	1	6	3	0

R_1

1	2	3	4	5
1	2	3	1	5
1	2	3	1	5
1	1	1	4	5
1	2	3	4	5

D_2

0	7	3	5	8
7	0	3	12	1
3	3	0	8	4
5	12	8	0	3
8	1	4	3	0

R_2

1	2	3	4	2
1	2	3	1	5
1	2	3	1	2
1	1	1	4	5
2	2	2	4	5

D_3

0	6	3	5	7
6	0	3	11	1
3	3	0	8	4
5	11	8	0	3
7	1	4	3	0

R_3

1	3	3	4	3
3	2	3	3	5
1	2	3	1	2
1	3	1	4	5
3	2	2	4	5

D_4

0	6	3	5	7
6	0	3	11	1
3	3	0	8	4
5	11	8	0	3
7	1	4	3	0

R_4

1	3	3	4	3
3	2	3	3	5
1	2	3	1	2
1	3	1	4	5
3	2	2	4	5

D_5

0	6	3	5	7
6	0	3	4	1
3	3	0	7	4
5	4	7	0	3
7	1	4	3	0

R_5

1	3	3	4	3
3	2	3	5	5
1	2	3	5	2
1	5	5	4	5
3	2	2	4	5

Teaching Discrete Mathematics

Activity: Floyd's Algorithm card sort (standard method)

Cut out the cards and place them in the correct order to give the solution for this problem.

Initial Distance Matrix: D(0)

	1	**2**	**3**	**4**	**5**
1	0	7	3	5	∞
2	7	0	3	∞	1
3	3	3	0	∞	6
4	5	∞	∞	0	3
5	∞	1	6	3	0

Initial Route Matrix: R(0)

	1	**2**	**3**	**4**	**5**
1	1	2	3	4	5
2	1	2	3	4	5
3	1	2	3	4	5
4	1	2	3	4	5
5	1	2	3	4	5

1	3	3	4	3
3	2	3	3	5
1	2	3	1	2
1	1	1	4	5
2	2	2	4	5

0	6	3	5	7
6	0	3	4	1
3	3	0	7	4
5	4	7	0	3
7	1	4	3	0

0	7	3	5	8
7	0	3	12	1
3	3	0	8	4
5	12	8	0	3
8	1	4	3	0

1	2	3	4	5
1	2	3	1	5
1	2	3	1	5
1	1	1	4	5
1	2	3	4	5

0	7	3	5	∞
7	0	3	12	1
3	3	0	8	6
5	12	8	0	3
∞	1	6	3	0

1	3	3	4	3
3	2	3	3	5
1	2	3	1	2
1	1	1	4	5
2	2	2	4	5

1	3	3	4	3
3	2	3	5	5
1	2	3	2	2
1	5	5	4	5
2	2	2	4	5

0	6	3	5	7
6	0	3	11	1
3	3	0	8	4
5	11	8	0	3
7	1	4	3	0

1	2	3	4	2
1	2	3	1	5
1	2	3	1	2
1	1	1	4	5
2	2	2	4	5

0	6	3	5	7
6	0	3	11	1
3	3	0	8	4
5	11	8	0	3
7	1	4	3	0

Teaching Discrete Mathematics

Activity: Floyd's Algorithm card sort solution (standard method)

Initial Distance Matrix: D(0)

	1	**2**	**3**	**4**	**5**
1	0	7	3	5	∞
2	7	0	3	∞	1
3	3	3	0	∞	6
4	5	∞	∞	0	3
5	∞	1	6	3	0

Initial Route Matrix: R(0)

	1	**2**	**3**	**4**	**5**
1	1	2	3	4	5
2	1	2	3	4	5
3	1	2	3	4	5
4	1	2	3	4	5
5	1	2	3	4	5

D(1)

0	7	3	5	∞
7	0	3	12	1
3	3	0	8	6
5	12	8	0	3
∞	1	6	3	0

R(1)

1	2	3	4	5
1	2	3	1	5
1	2	3	1	5
1	1	1	4	5
1	2	3	4	5

D(2)

0	7	3	5	8
7	0	3	12	1
3	3	0	8	4
5	12	8	0	3
8	1	4	3	0

R(2)

1	2	3	4	2
1	2	3	1	5
1	2	3	1	2
1	1	1	4	5
2	2	2	4	5

D(3)

0	6	3	5	7
6	0	3	11	1
3	3	0	8	4
5	11	8	0	3
7	1	4	3	0

R(3)

1	3	3	4	3
3	2	3	3	5
1	2	3	1	2
1	1	1	4	5
2	2	2	4	5

D(4)

0	6	3	5	7
6	0	3	11	1
3	3	0	8	4
5	11	8	0	3
7	1	4	3	0

R(4)

1	3	3	4	3
3	2	3	3	5
1	2	3	1	2
1	1	1	4	5
2	2	2	4	5

D(5)

0	6	3	5	7
6	0	3	4	1
3	3	0	7	4
5	4	7	0	3
7	1	4	3	0

R(5)

1	3	3	4	3
3	2	3	5	5
1	2	3	2	2
1	5	5	4	5
2	2	2	4	5