Edexcel A Level Further Maths: Decision Maths 1

Your notes

The Travelling Salesman Problem

Contents

The Travelling Salesman Problem

Your notes

The Classical & Practical Travelling Salesman Problem

What is the Travelling Salesman Problem?

- In the **Travelling Salesman Problem**, the objective is to find the **tour of least weight** in a given network
  - A **tour** is a **walk** that **starts at one vertex** and **visits every other vertex** in the network before returning to the **start vertex**
- The **classical** travelling salesman problem requires that **each vertex** is visited **exactly once** before returning to the start vertex
- The **practical** travelling salesman problem requires that **each vertex** must be visited **at least once** before returning to the start vertex
- There isn't an algorithm that will definitely provide an optimal solution to the travelling salesman problem
  - However the methods used will provide a **lower** and **upper** bound for the **tour** of **least weight**
  - If the **lower** and **upper** bound happen to be **equal,** an **optimal** solution has been found

What is a table of least distances?

- A **table of least distances** shows the **shortest** distance between any **two vertices** in a graph
  - In some cases, the **direct route** between two vertices may **not** be the **shortest**
- The table of least distances creates a **complete** network and so it will contain a **Hamiltonian cycle**
- The table of least distances turns a **practical travelling salesman problem** into a **classical travelling salesman problem**

How do I find the table of least distances?

- In practice, for smaller networks, the values in the **table of least distances** can usually be found by **inspection** of the network
  - The shortest route between two vertices may not be the most direct route
  - A more robust method of finding the shortest routes between pairs of vertices would be to use **Dijkstra's or Floyd's algorithm**
    - An exam question will be clear if this is required but given time constraints this is uncommon
- Mathematically, a **table of least distances** is created by ensuring every entry satisfies the **triangle inequality**
  - the **longest side** of a triangle **≤** the **sum** of the lengths of the **other two sides**
  - ('Sides of the triangle' could be made from one or more edges in a network - there would be lots of them, even in small networks!)
- To construct a table of least distances follow the steps below
- **STEP 1**
  Put a line through each cell along the leading diagonal
  - there will be no loops!

- **STEP 2**
  Fill in the information for a particular vertex in the graph with the distance between that vertex and each vertex that is **adjacent** to it in the network (i.e. vertices that are directly connected to it)
    - At this stage check (by inspection) if the direct connections are actually the **shortest** distances and change as necessary
- **STEP 3**
  Complete the rest of the connections between that particular vertex and the remaining (non-adjacent) vertices in the network by finding the shortest route that can be travelled between each pair
- **STEP 4**
  Copy the values in the completed row down the corresponding column
    - a complete network is undirected so the table of least distances will have symmetry
- **STEP 5**
  If the table is **not complete**, return to STEP 2 and continue with the another vertex, otherwise STOP

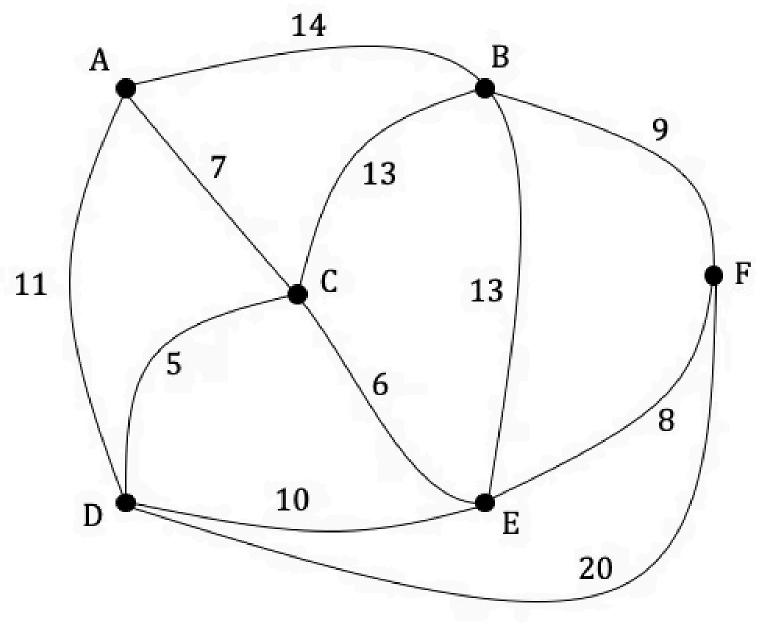
Examiner Tip

- Remember that the table of least values has a **line of symmetry** along the leading diagonal for an **undirected network**, so complete one half carefully first, then map over to the second half

Your notes

Worked example

The network below contains six vertices representing villages and the edges (roads) that connect them. The weighting of the edges represents the time, in minutes, that it takes to walk along a particular road between two villages.



a) Explain why the network in the diagram is not a complete network.

It is not a complete network as each vertex is not connected to every other vertex by a single edge
e.g., there is no single edge that connects vertices B and D

b) Complete the table of least distances for the network below.

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

Your notes

- **STEP 1**
  Put a line through each cell on the leading diagonal
- **STEP 2**
  Fill in the table with the direct connections from the graph, starting from vertex A
  - Check that the direct connection is the shortest route in all cases

	A	B	C	D	E	F
A	–	14	7	11		
B		–				
C			–			
D				–		
E					–	
F						–

- **STEP 3**
  Find the shortest routes between vertices A and E, and A and F, then fill these values in the table

$$A \rightarrow E:\ A \rightarrow C \rightarrow E = 13$$

$$A \rightarrow F:\ A \rightarrow C \rightarrow E \rightarrow F = 21$$

	A	B	C	D	E	F
A	–	14	7	11	13	21
B		–				
C			–			
D				–		
E					–	
F						–

- **STEP 4**
  Complete column A with the values recorded in row A.

	A	B	C	D	E	F
A	–	14	7	11	13	21
B	14	–				

Your notes

C	7		-			
D	11			-		
E	13				-	
F	21					-

- **STEP 5**
  The table is not complete so return to Step 2
- **STEP 2**
  Fill in the table with the direct connections from vertex B

	A	B	C	D	E	F
A	-	14	7	11	13	21
B	14	-	13		13	9
C	7		-			
D	11			-		
E	13				-	
F	21					-

- **STEP 3**
  Complete the table with the shortest route between the non-adjacent vertices B and D

$$B \rightarrow D: \quad B \rightarrow C \rightarrow D = 18$$

	A	B	C	D	E	F
A	-	14	7	11	13	21
B	14	-	13	18	13	9
C	7		-			
D	11			-		
E	13				-	
F	21					-

- **STEP 4**
  Complete column B with the values recorded in row B

	A	B	C	D	E	F
A	–	14	7	11	13	21
B	14	–	13	18	13	9
C	7	13	–			
D	11	18		–		
E	13	13			–	
F	21	9				–

- **STEP 5**
  The table is not complete so return to Step 2
- **STEP 2**
  Fill in the table with the direct connections from vertex C

	A	B	C	D	E	F
A	–	14	7	11	13	21
B	14	–	13	18	13	9
C	7	13	–	5	6	
D	11	18		–		
E	13	13			–	
F	21	9				–

- **STEP 3**
  Complete the table with the shortest route between the non-adjacent vertices C and F

CF: $C \rightarrow E \rightarrow F = 14$

	A	B	C	D	E	F
A	–	14	7	11	13	21
B	14	–	13	18	13	9
C	7	13	–	5	6	14
D	11	18		–		
E	13	13			–	
F	21	9				–

Your notes

Your notes

- **STEP 4**
  Complete column C with the values recorded in row C

	A	B	C	D	E	F
A	-	14	7	11	13	21
B	14	-	13	18	13	9
C	7	13	-	5	6	14
D	11	18	5	-		
E	13	13	6		-	
F	21	9	14			-

- **STEP 5**
  The table is not complete so return to Step 2
- **STEP 2**
  Fill in the table with the direct connections from vertex D
  The direct route between D and F is not the shortest route

$$D \rightarrow F = 20$$

$$D \rightarrow E \rightarrow F = 18$$

	A	B	C	D	E	F
A	-	14	7	11	13	21
B	14	-	13	18	13	9
C	7	13	-	5	6	14
D	11	18	5	-	10	18
E	13	13	6		-	
F	21	9	14			-

- **STEP 3**
  There are no non-adjacent vertices for D left to complete
- **STEP 4**
  Complete column D with the values recorded in row D

	A	B	C	D	E	F
A	-	14	7	11	13	21

Your notes

B	14	–	13	18	13	9
C	7	13	–	5	6	14
D	11	18	5	–	10	18
E	13	13	6	10	–	
F	21	9	14	18		–

- **STEP 5**
  The table is not complete so return to Step 2
- **STEP 2**
  Fill in the table with the direct connections from vertex E

	A	B	C	D	E	F
A	–	14	7	11	13	21
B	14	–	13	18	13	9
C	7	13	–	5	6	14
D	11	18	5	–	10	18
E	13	13	6	10	–	8
F	21	9	14	18		–

- **STEP 3**
  There are no other connections for vertex E
- **STEP 4**
  Complete column E with the values recorded in row E
- **STEP 5**
  The table is complete so stop

	A	B	C	D	E	F
A	**–**	**14**	**7**	**11**	**13**	**21**
B	**14**	**–**	**13**	**18**	**13**	**9**
C	**7**	**13**	**–**	**5**	**6**	**14**
D	**11**	**18**	**5**	**–**	**10**	**18**
E	**13**	**13**	**6**	**10**	**–**	**8**
F	**21**	**9**	**14**	**18**	**8**	**–**

Your notes

Upper & Lower Bounds for the Travelling Salesman Problem

Finding an Upper Bound using a Minimum Connector

Why do I need to find an upper bound?

- There is **no efficient algorithm** to find the optimal solution for the travelling salesman problem
  - An **upper and lower bound** for the solution to the problem can be found
  - The **optimal solution** (the shortest route) will lie **within** these bounds
- The aim is to find the **minimum value for the upper bound** to reduce the size of interval between the upper and lower bounds

How do I find the upper bound?

- You can find an **upper bound** for the **practical travelling salesman problem** using the **minimum spanning tree** of a network

- STEP 1
  Find the **minimum spanning tree** for the network using either **Prim's** or **Kruskal's** algorithm
  - If you have the **table of least distances** - use Prim's algorithm
  - If you have the **graph** only - use Kruskal's algorithm

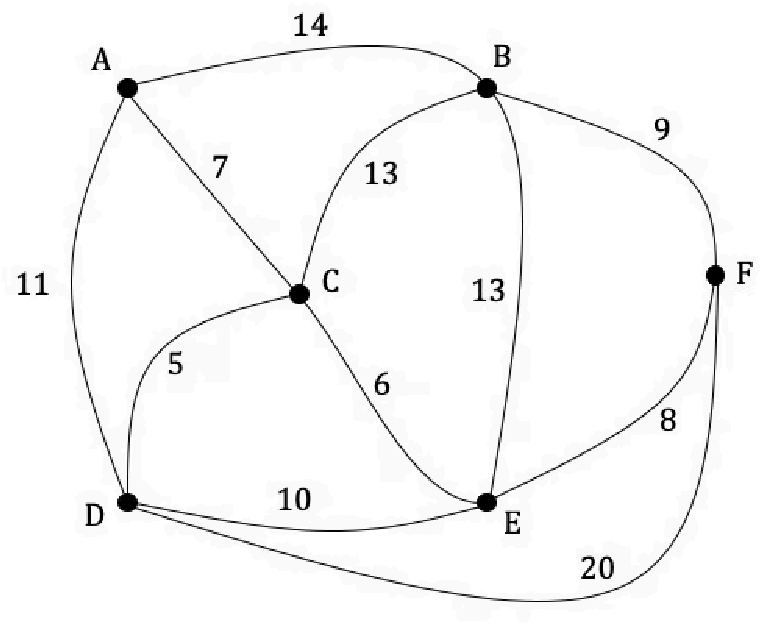
- STEP 2
  Double the weight of the minimum spanning tree
  - Each vertex can therefore definitely be visited and the starting vertex returned to

- STEP 3
  Reduce the weight of the upper bound by finding shortcuts
  - Use some of the **edges** that are **not included** in the minimum spanning tree to reduce the weight of the upper bound

Your notes

Worked example

The network below contains six vertices representing villages and the edges (roads) that connect them. The weighting of the edges represents the time, in minutes, that it takes to walk along a particular road between two villages.



a) Find an initial upper bound for the travelling salesman problem.

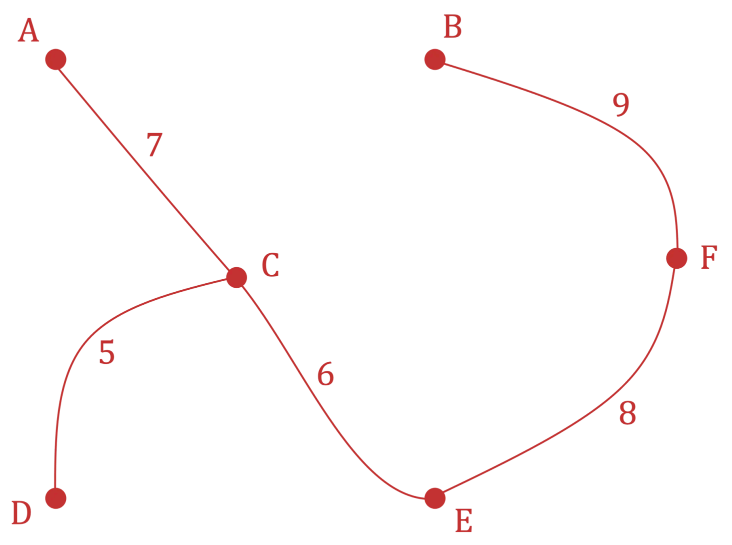
- **STEP 1**
  The question contains the graph but you have not been asked to find the table of least distances, so apply Kruskal's algorithm to find the minimum spanning tree of the network
  List the edges in order of increasing weight

CD (5)
CE (6)
AC (7)
EF (8)
BF (9)
DE (10)
AD (11)
BC (13)
BE (13)
AB (14)
DF (20)

Your notes

Add the edge of least weight (CD), then the next edge of least weight (CE)
Repeat adding the next edge of least weight each time provided it does not create a cycle until all vertices are connected

Edges added in the order: CD (5), CE (6), AC (7), EF (8), BF (9)



- **STEP 2**
  Find the weight of the minimum spanning tree

$$5 + 6 + 7 + 8 + 9 = 35$$

Double the weight of the minimum spanning tree to find the initial upper bound of the solution

$$35 \times 2 = 70$$

Initial upper bound = 70 minutes

b) Show that the initial upper bound can be reduced to 54 by finding one shortcut.

- **STEP 3**
  Identify which edges could be used to replace repeated edges in the minimum spanning tree

Single edge: AB (14)

Your notes

Repeated pathway: AC + CE + EF + FB (30)

Subtract the weight of the single edge from the repeated pathway to calculate the weight saved by using the short cut

30 - 14 = 16

Subtract the weight saving from the initial upper bound

70 - 16 = 54

Improved upper bound = 54 minutes

This can also be seen as 70 - 30 + 14 = 54
(subtract the 'repeated pathway', then add the 'shortcut')

Your notes

Finding a Lower Bound using a Minimum Connector

Why do I need to find the lower bound?

- There is **no efficient algorithm** to find the **optimal** solution for the travelling salesman problem
  - An **upper and lower bound** for the solution to the problem can be found
  - The **optimal solution** (the shortest route) will lie **within** these bounds
- The aim is to find the **maximum value** for the **lower bound** to reduce the size of interval between the upper and lower bounds
- An **optimal solution** is found if
  - the **lower bound** gives a **Hamiltonian cycle**
  - the **lower bound** has the **same value** as the **upper bound**

How do I find the lower bound?

- You can find a **lower bound** for the **classical travelling salesman problem** using the **minimum spanning tree** of a network
  - To find a solution for the **practical travelling salesman problem** you need to apply the steps of the algorithm below to a **table of least distances**

- **STEP 1**
  Remove a **vertex** and all of its **associated edge**s from the network

- **STEP 2**
  Find the **minimum spanning tree** for the remaining **network** and write down its **total weight**
  - This spanning tree is known as the **residual minimum spanning tree** (RMST)

- **STEP 3**
  Identify the **two shortest individual edges** that can **connect** the **missing vertex** to the **residual minimum spanning tree**
  Calculate the **sum** of the **total weight** of the **residual minimal spanning tree** and the **two connecting edges**

- **STEP 4**
  Repeat Steps 1 to 3 for **each vertex** in the network

- **STEP 5**
  When the network has been tested for **each missing vertex**, identify the **maximum value**
  - This maximum value is the **greatest possible lower bound**

- In many problems Steps 4 and 5 will not be required as a question will dictate which vertex to delete and there will be no need to do them all
- This method may be referred to as the **deleted vertex method**

Your notes

Worked example

The table below contains six vertices representing villages and the roads that connect them. The weighting of the edges represents the time in minutes that it takes to walk along a particular road between two villages.

	A	B	C	D	E	F
A	–	14	7	11	13	21
B	14	–	13	18	13	9
C	7	13	–	5	6	14
D	11	18	5	–	10	18
E	13	13	6	10	–	8
F	21	9	14	18	8	–

a) By deleting vertex A and using Prim's algorithm, find a lower bound for the time taken to start at village A, visit each of the other villages and return to village A

- **STEP 1**
  Delete vertex A in the table

- **STEP 2**
  Perform Prim's algorithm on the remaining vertices in the table

		~~A~~	B	C	D	E	F
	~~A~~	~~-~~	~~14~~	~~7~~	~~11~~	~~13~~	~~21~~
1	B	~~14~~	-	~~13~~	18	~~13~~	9
4	C	~~7~~	~~13~~	-	5	~~6~~	~~14~~
5	D	~~11~~	~~18~~	~~5~~	-	~~10~~	~~18~~
3	E	~~13~~	~~13~~	6	10	-	~~8~~
2	F	~~21~~	~~9~~	~~14~~	18	8	-

Edges added in the order: BF (9), EF (8), CE (6), CD (5)

Total weight of the residual minimum spanning tree = 9 + 8 + 6 + 5 = 28

- **STEP 3**
  Identify the two shortest edges that connect A to the minimum spanning tree

AC (7), AD (11)

Add together the total weight of the minimum spanning tree and the weights of the two shortest lengths connecting it to A

28 + 7 + 11 = 46

Lower bound = 46 minutes

b) Show that by deleting vertex E instead, a higher lower bound can be found.

- **STEP 1**
  Delete vertex A in the table

- **STEP 2**
  Perform Prim's algorithm on the remaining vertices in the table

		A	B	C	D	~~E~~	F
1	A	-	~~14~~	7	~~11~~	~~13~~	21
4	B	~~14~~	-	~~13~~	~~18~~	~~13~~	9
2	C	~~7~~	~~13~~	-	5	~~6~~	14
3	D	~~11~~	18	~~5~~	-	~~10~~	18
	~~E~~	~~13~~	~~13~~	~~6~~	~~10~~	~~-~~	~~8~~
5	F	~~21~~	~~9~~	~~14~~	~~18~~	~~8~~	-

Edges added in the order: AC (7), CD (5), BD/DF (18), BF (9)

Total weight of the residual minimum spanning tree = 7 + 5 + 18 + 9 = 39

- **STEP 3**
  Identify the two shortest edges that connect E to the minimum spanning tree

CE (6), EF (8)

Add together the total weight of the minimum spanning tree and the weights of the two shortest lengths connecting it to E

Your notes

$39 + 6 + 8 = 53$

Lower bound = 53 minutes

This is a higher lower bound than found when vertex A was deleted (46 minutes)

Your notes

Nearest Neighbour Algorithm

What is the nearest neighbour algorithm?

- For a **complete** network with at least 3 vertices, performing the **nearest neighbour algorithm** will generate a **low** (but not necessarily least) **weight** Hamiltonian cycle
    - This low weight cycle can be considered the **upper bound**
- The **best upper bound** is the upper bound with the **smallest value**
- The nearest neighbour algorithm can only be used on a graph that is **complete** and satisfies the **triangle inequality**
    - If need be the **table of least distances** should be found first

What are the steps of the nearest neighbour algorithm?

- **STEP 1**
  Choose a **starting vertex**
- **STEP 2**
  Select the **edge of least weigh**t from the **current vertex** to an **adjacent unvisited vertex**
    - if there is more than one edge of least weight pick one at random
    - move to this vertex (column in matrix form)
    - if this is the final vertex to be visited go to Step 3
    - otherwise disregard (cross out in matrix form) any edges that go to vertices already visited and repeat Step 2
- **STEP 3**
  Add a final edge to return to the **starting vertex**
- **STEP 4**
  Find the **weight** of the resulting Hamiltonian cycle to produce an upper bound for the travelling salesman problem
- **STEP 5**
  Repeat Steps 1 to 4 starting with a different **vertex** in the network
  When the network has been tested for **each vertex** as the starting vertex, identify the **minimum value**
  This minimum value is the least **possible lower bound**
- In many problems Step 5 will not be required as a question will dictate which vertex to start at and there will be no requirement to repeat with all vertices as the starting vertex

Examiner Tip

- Be careful not to muddle up the nearest neighbour algorithm with Prim's algorithm just because they can both be performed on tables - this is a common mistake!
- Questions may ask you to explain the difference between the two algorithms
  - The nearest neighbour algorithm selects the nearest vertex adjacent to the **current** vertex
  - Prim's algorithm selects the nearest vertex adjacent to **any** vertex already visited

Your notes

Worked example

The table below contains six vertices representing villages and the roads that connect them. The weighting of the edges represents the time in minutes that it takes to walk along a particular road between two villages.

	A	B	C	D	E	F
A	-	14	7	11	13	21
B	14	-	13	18	13	9
C	7	13	-	5	6	14
D	11	18	5	-	10	18
E	13	13	6	10	-	8
F	21	9	14	18	8	-

Starting at village A, use the nearest neighbour algorithm to find the upper bound of the time it would take to visit each village and return to village A.

- **STEP 1**
  Start at vertex A (column A)

- **STEP 2**
  Select the edge of least weight in column A (AC) and write it down

	A	B	C	D	E	F
A	-	14	7	11	13	21
B	14	-	13	18	13	9
C	$\bigcirc\!\!\!\!7$	13	-	5	6	14
D	11	18	5	-	10	18
E	13	13	6	10	-	8
F	21	9	14	18	8	-

AC (7)

Move to vertex/column C
C is not the final vertex to be visited
Disregard (cross out) the edge (value) in column C that would connect C to A as vertex A has already been visited
Repeat Step 2

- **STEP 2**
  Select the edge of least weight (from those remaining) in column C (CD) and write it down

	A	B	C	D	E	F
A	-	14	~~7~~	11	13	21
B	14	-	13	18	13	9
C	7	13	-	5	6	14
D	11	18	5	-	10	18
E	13	13	6	10	-	8
F	21	9	14	18	8	-

AC (7), CD (5)

Move to column D
D is not the final vertex
Cross out the values for the vertices already visited (A and C)
Repeat Step 2

- **STEP 2**
  Select the edge of least weight (from the remaining) in column D (DE) and write it down

	A	B	C	D	E	F
A	-	14	~~7~~	~~11~~	13	21
B	14	-	13	18	13	9
C	7	13	-	~~5~~	6	14
D	11	18	5	-	10	18
E	13	13	6	10	-	8
F	21	9	14	18	8	-

AC (7), CD (5), DE (10)

Move to column E
E is not the final vertex
Cross out the values for the vertices already visited (A, C and D)
Repeat Step 2

- **STEP 2**
  Select the edge of least weight (from the remaining values) in column E (EF) and write it down

	A	B	C	D	E	F
A	-	14	~~7~~	~~11~~	~~13~~	21
B	14	-	13	18	13	9
C	[✓ 7]	13	-	~~5~~	~~6~~	14
D	11	18	[✓ 5]	-	~~10~~	18
E	13	13	6	[✓ 10]	-	8
F	21	9	14	18	[✓ 8]	-

AC (7), CD (5), DE (10), EF (8)

Move to vertex F
F is not the final vertex
Cross out the values for the vertices already visited (A, C, D and E)
Repeat Step 2

- **STEP 2**
  Select the edge of least weight (from the remaining values) in column F (BF) and write it down

	A	B	C	D	E	F
A	-	14	~~7~~	~~11~~	~~13~~	~~21~~
B	14	-	13	18	13	[✓ 9]
C	[✓ 7]	13	-	~~5~~	~~6~~	~~14~~
D	11	18	[✓ 5]	-	~~10~~	~~18~~
E	13	13	6	[✓ 10]	-	~~8~~
F	21	9	14	18	[✓ 8]	-

AC (7), CD (5), DE (10), EF (8), BF (9)

Move to vertex B
B is the final vertex to be visited
Go to Step 3

- **STEP 3**
  Choose the final edge (value) to get back from the last vertex (B) to the starting vertex (A)

	A	B	C	D	E	F
A	-	14	~~7~~	~~11~~	~~13~~	~~21~~
B	14	-	13	18	13	9
C	7	13	-	~~5~~	~~6~~	~~14~~
D	11	18	5	-	~~10~~	~~18~~
E	13	13	6	10	-	~~8~~
F	21	9	14	18	8	-

AC (7), CD (5), DE (10), EF (8), BF (9), AB (14)

Hamiltonian cycle: ACDEFBA

- **STEP 4**
  Find the weight of the Hamiltonian cycle by adding together the weights of the edges forming it
  Total weight = 7 + 5 + 10 + 8 + 9 + 14 = 53

  This is an upper bound for the travelling salesman problem

Upper bound = 53 minutes

Solving the Travelling Salesman Problem

Your notes

Solving the Travelling Salesman Problem

How do I solve the travelling salesman problem?

- List **all** of the possible **Hamiltonian cycles** and find the cycle of **least weight**
  - A complete graph with 3 vertices will have 2 possible Hamiltonian cycles, 4 vertices will have 6 possible cycles and a graph with 5 vertices will have 24 possible cycles
- There is **no known algorithm** that guarantees finding the shortest Hamiltonian cycle in a graph so this method is only suitable for **small graphs**

Examiner Tip

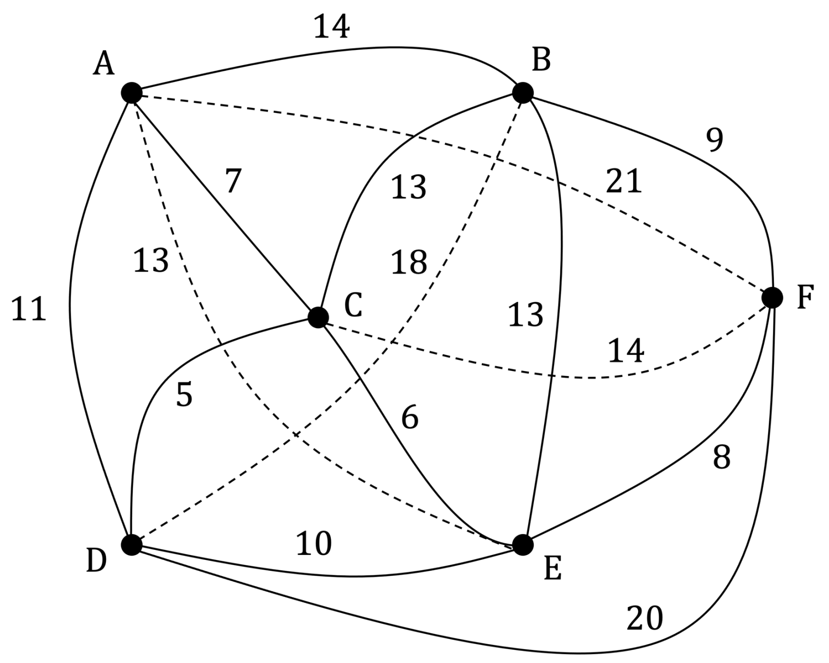
- To remember the difference between the travelling salesman problem and the Chinese postman problem, remember that the salesman is interested in selling at each destination (vertex) whereas the postman wants to walk along every road (edge) in order to deliver the letters

Your notes

Worked example

The network below contains six vertices representing villages and the edges (roads) that connect them. The weighting of the edges represents the time, in minutes, that it takes to walk along a particular road between two villages.

The dashed lines indicate the edges that are to be repeated to convert the network into a complete network.



The lower bound for the time taken to walk a route that visits every vertex and returns to the start vertex is 53 minutes.

Starting from vertex A, find a route that shows that this is the optimal solution.

Because the network has 6 vertices there will be 120 possible different Hamiltonian cycles - this is too many to check!
However, you are told to start at vertex A and you are told to confirm that the route you have found is the optimal solution, this means that it should have the same weight as the lower bound for the problem.

Find a Hamiltonian cycle starting at vertex A.

AB (14)
BF (9)
EF (8)
CF (6)
CD (5)
AD (11)

Calculate the total weight of the cycle.

$$14 + 9 + 8 + 6 + 5 + 11 = 53$$

The route ABFECDA has the same weight (53 minutes) as the lower bound of the problem, therefore it is the optimal solution