Discrete maths:

Graphs and Networks

Student Handout

Name:_____

Please ensure that you keep this booklet with you for all Discrete lessons, and bring it with you when you return to school.

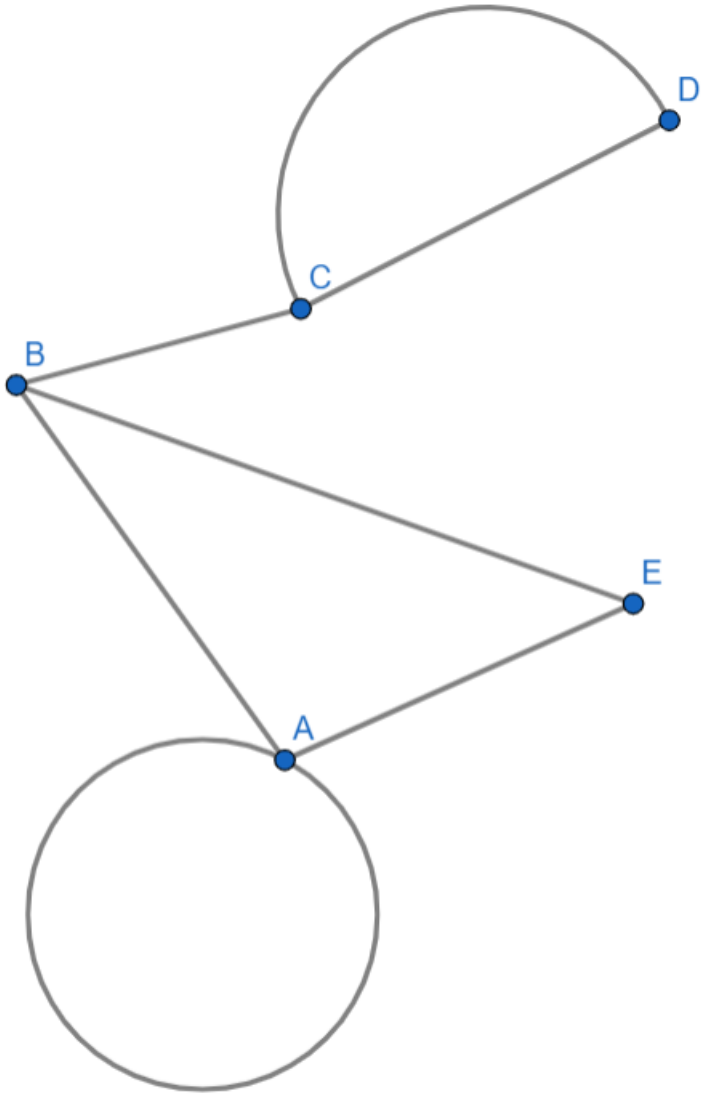
Contents:

Graphs Section 1: Elements of Graphs

This is a graph. Graphs in discrete maths do not look like the graphs you have seen previously. They consist of points called **vertices** joined together by lines called **edges**.

Label the following elements of the graph:

- Vertex
- Edge
- Multiple edge
- Loop



A **trail** describes a path that takes you from a starting vertex, along a series of edges, to a finishing vertex.

An example of a trail on the graph above is E A B C.

Write down another trail from the above graph _____

A **cycle** is a special type of trail that starts and ends at the same vertex.

An example of a cycle on the above graph is B E A B.

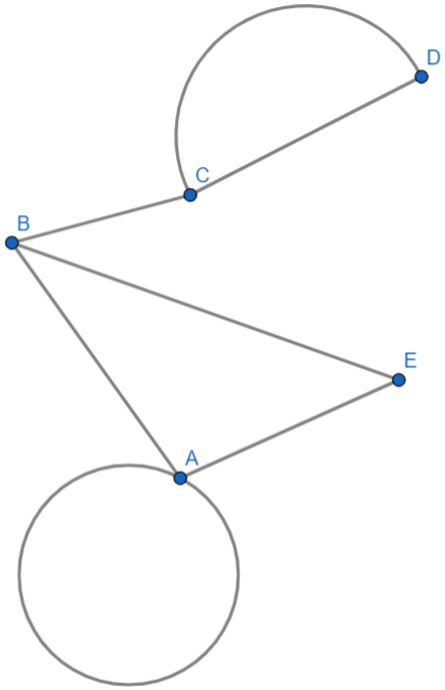
Write down another cycle from the above graph _____

The **degree** of a vertex is the number of edges that come out of the vertex.

The degree of vertex E is 2.

Label each of the vertices with its degree.

Graph G



Graph G is a **connected graph** as it is possible to find a trail that connects each vertex to every other vertex.

Draw a graph below that isn't connected.

A **subgraph** is a graph that is entirely contained within another graph.

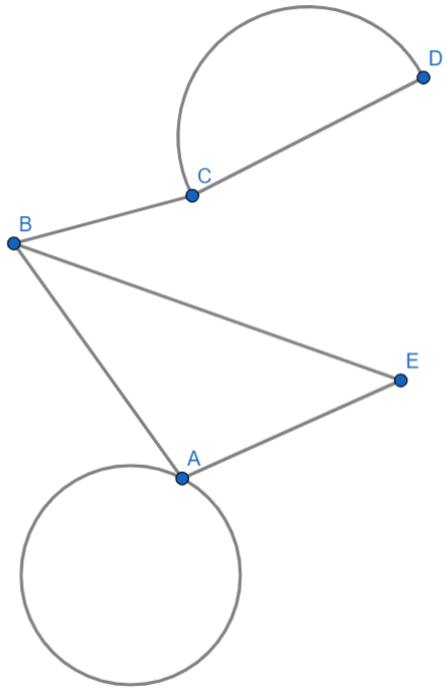
Draw a subgraph of graph G below.

A **subdivision** is a graph that can be created by splitting one (or more) of the edges in a graph by adding in additional vertices.

Draw a subdivision of graph G below.

Graphs Section 2: Types of Graphs

Graph G



A **simple graph** is a graph that contains no loops or multiple edges.

Draw a subgraph of graph G that is a simple graph below.

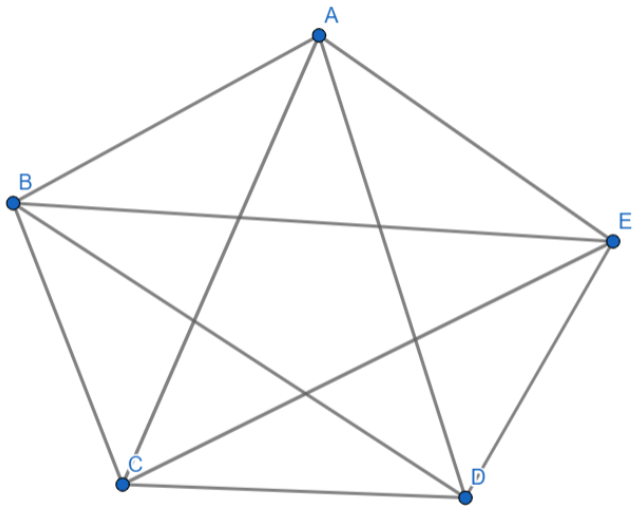
A **planar graph** is one that can be drawn on a flat surface with no lines crossing over.

Draw a graph below that is planar.

Draw a graph below that is not planar

A **complete graph** is one where every vertex is connected to every other. They are denoted with a capital letter K and a subscript that shows the number of vertices they have.

Here is the graph of K_5.



Draw a graph of K_4 below

How many edges does the graph K_4 have?

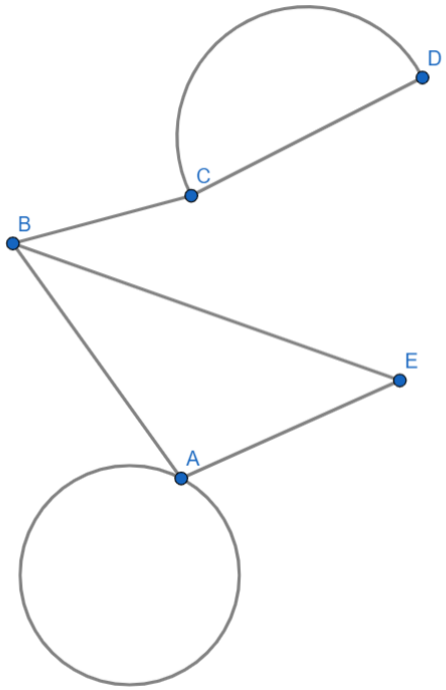
How many edges does the graph K_5 have?

How many edges does the graph K_n have?

A **tree** is a connected graph that contains no cycles.

Draw a graph below that is a tree.

Graph G

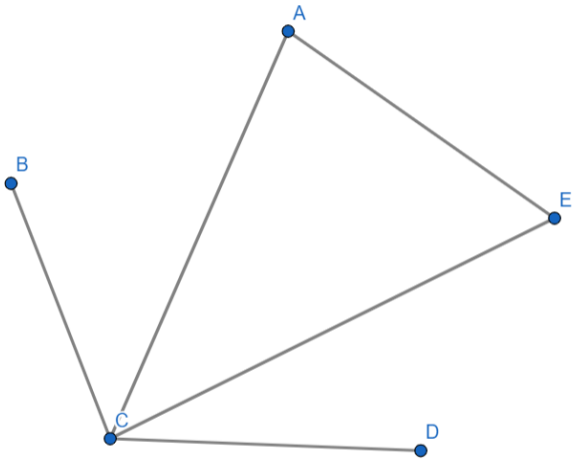


An **adjacency matrix** is a grid that shows which vertices are connected to which other vertices.

Fill in the adjacency matrix for graph G below:

	A	B	C	D	E
A					
B					
C					
D					
E					

Graph H



Complements are graphs that together, form the complete graph with the same number of vertices.

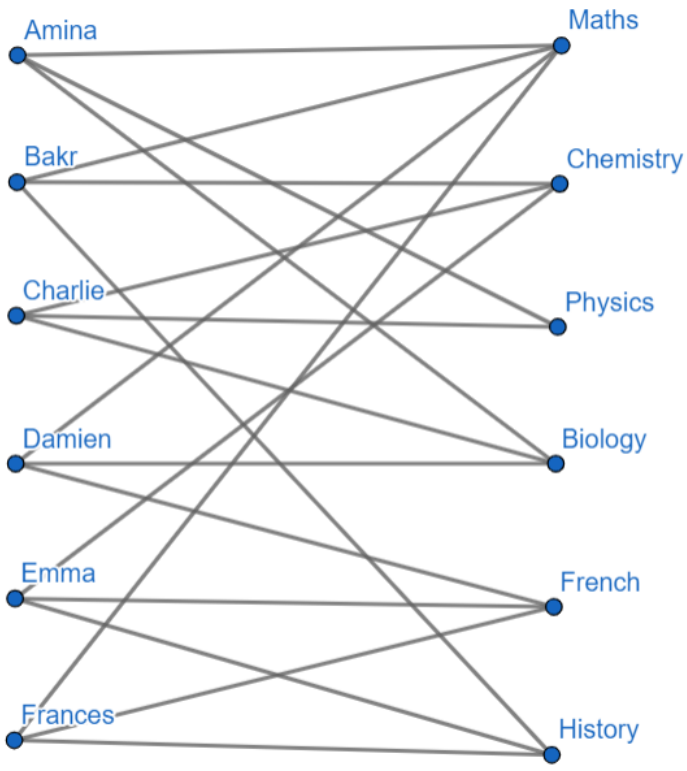
The complement of Graph H, denoted by H' contains the same vertices as graph H, but all the edges that graph H doesn't have.

Draw the complement of graph H below.

A **bipartite graph** is a type of graph that has two distinct sets of vertices. Each vertex can only be connected to vertices in the other set.

Here is an example showing which A-Levels a group of six students are studying:

Graph B



Complete the adjacency matrix for graph B below

	M	C	P	B	F	H
A						
B						
C						
D						
E						
F						

5 more students are making their A-Level choices.

George does English, Psychology and Criminology.

Harry does English, Politics and Sociology

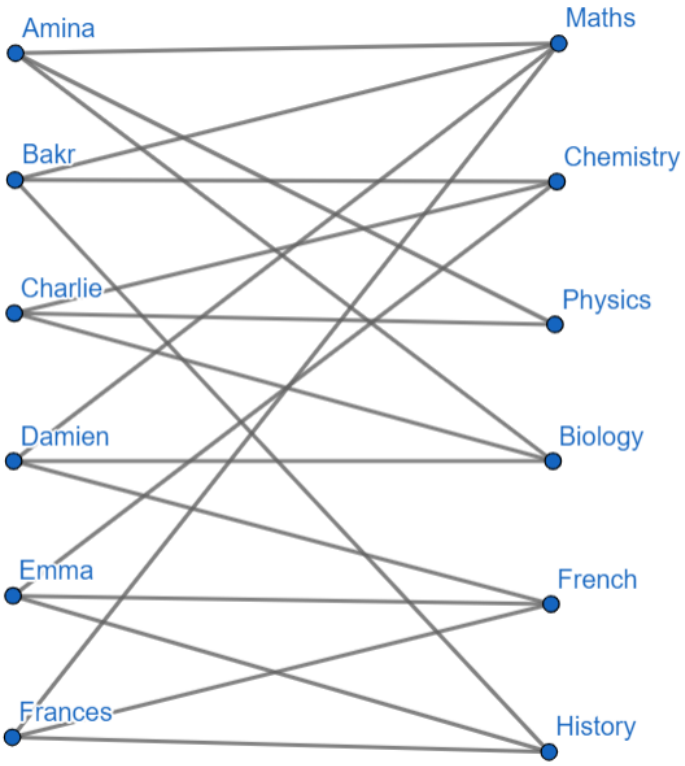
Iara does Politics, Criminology and Psychology

Jamal does English, Sociology and Psychology

Kiran does Politics, Sociology and Criminology

Draw a bipartite graph to represent this information.

Graph B

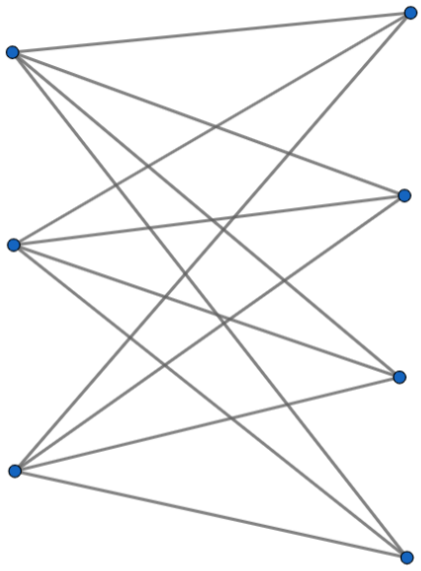


A **full matching** is when it is possible to connect each of the vertices in one set with one unique vertex in the other set.

Write down a full matching for graph B

A **complete bipartite graph** is one where each vertex is connected to all the vertices in the other set. They are denoted by a capital K, then a subscript with the number of vertices in each set, separated by a comma.

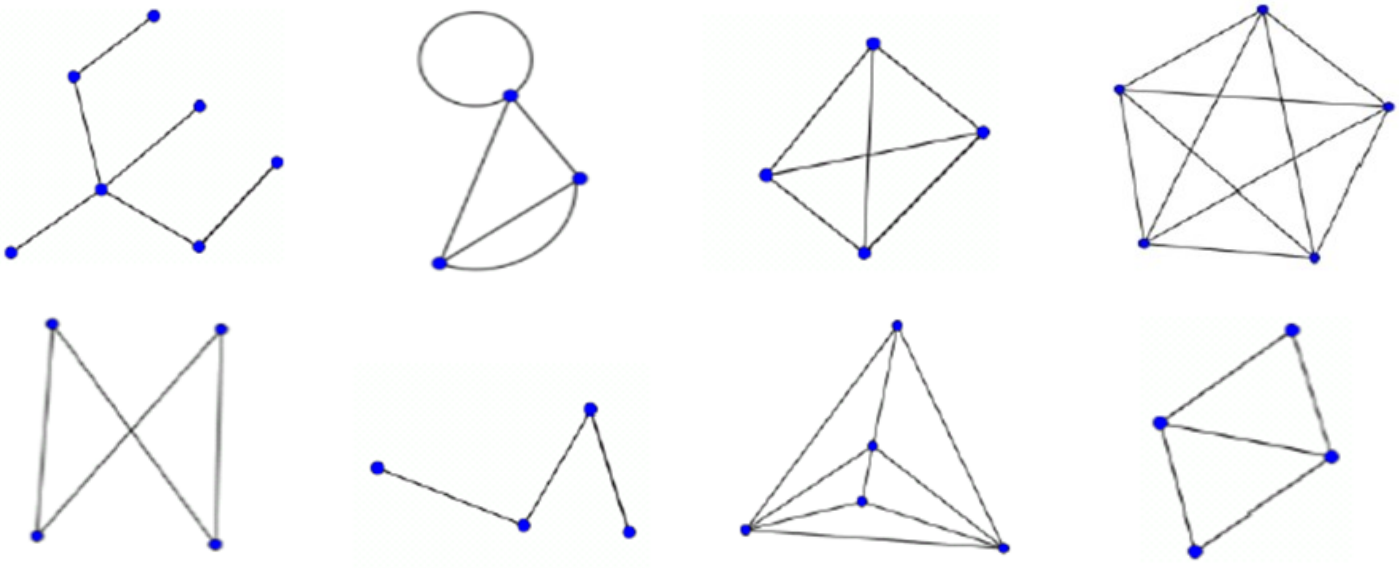
Below is the graph of $K_{3,4}$



Draw the graph of $K_{3,3}$ below

Graphs Section 3: Properties of Graphs

An **Eulerian trail** is one that covers each edge once and once only. For each of the graphs below, see if it is possible to trace the whole graph with your pencil without lifting your pencil, and without going over the same edge more than once.

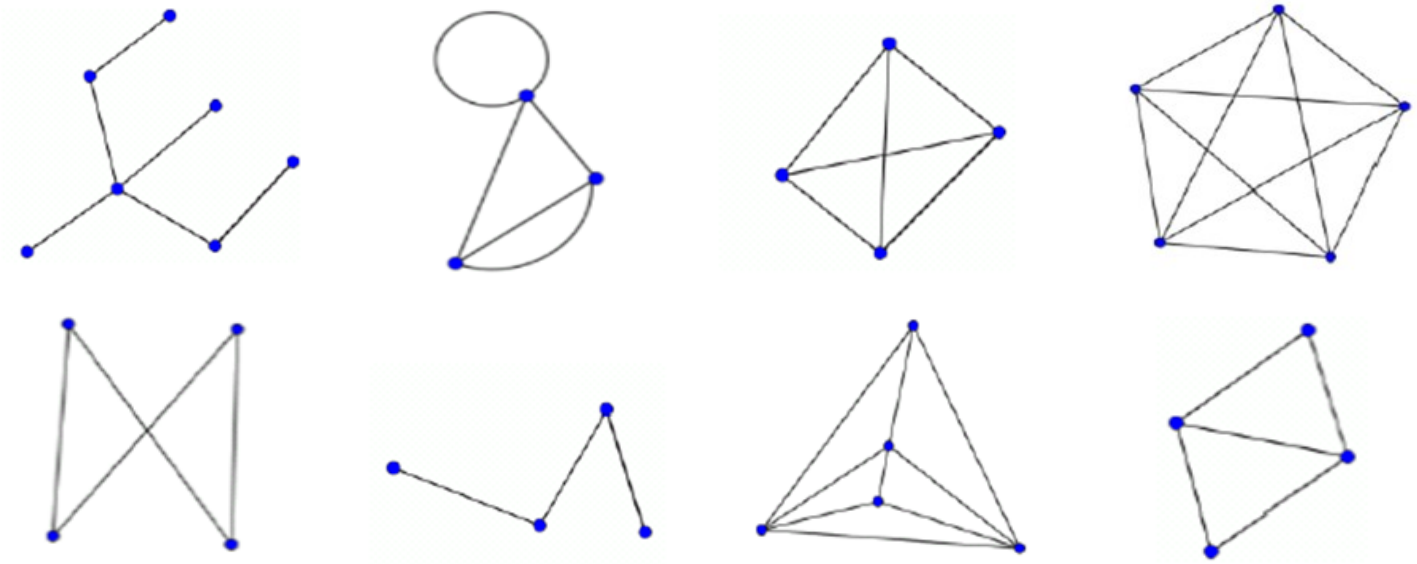


Where this is possible, and the start and end points are the same, then the graph is said to be **Eulerian**.

Where an Eulerian trail exists but the start and end points are different, then the graph is **semi-Eulerian**.

In order for a graph to be **Eulerian:**	In order for a graph to be **semi-Eulerian:**

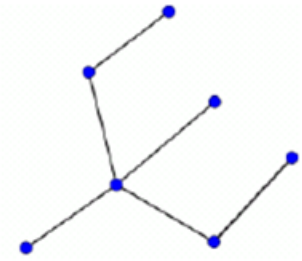
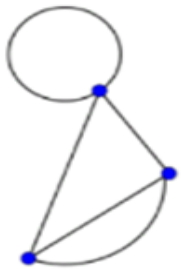
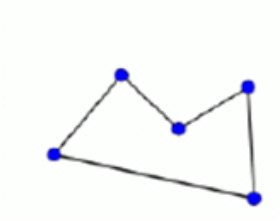
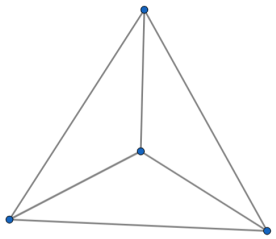
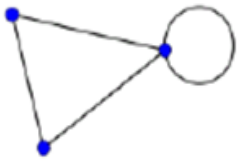
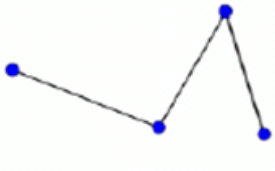
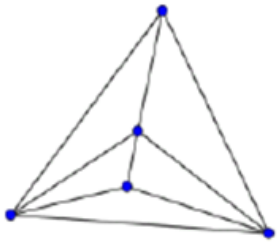
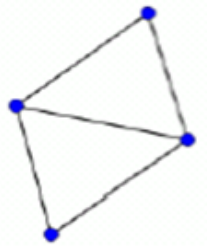
A **Hamiltonian cycle** is one which visits each **vertex** of a graph once and once only, before returning to the starting point. Which of the graphs below contain Hamiltonian cycles?



A graph that contains a Hamiltonian cycle is said to be a **Hamiltonian graph**.

Fill in the table for the graphs below. The first one has been done for you:

A

Graph	Vertices	Edges	Regions
A	7	6	1

(outside of the graph also counts as a region)

For **connected, planar graphs** there is a link between these values.

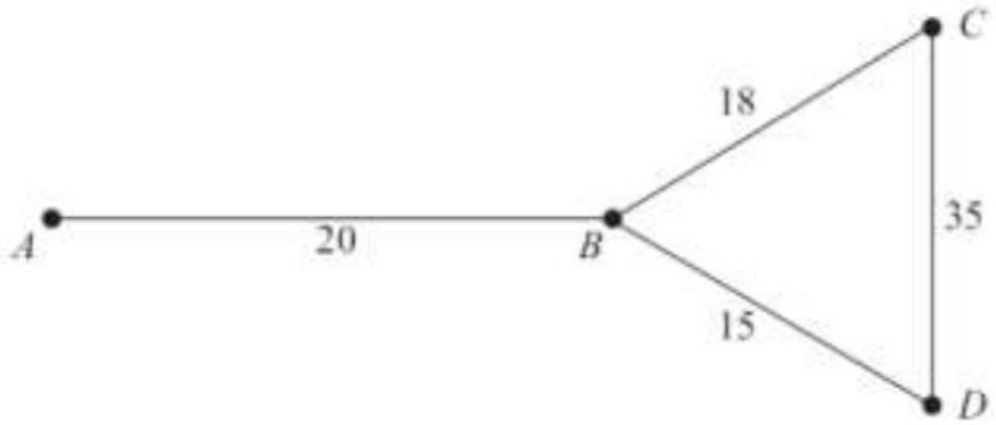
The link is

This is known as **Euler's Formula**.

Networks Section 1: Elements of Networks

This is a **network**. Networks look a lot like graphs, but each of the edges now has a **weight** and is referred to as an **arc**. The weight can refer to a cost, a distance, a time taken, or something else. The context is usually given in the questions.

The vertices in a network are referred to as **nodes**.



The adjacency matrices for networks are called **distance matrices** and instead of the number of arcs connecting two nodes, each cell gives the weight of the arc connecting the two nodes.

Complete the distance matrix below for the network above:

	A	B	C	D
A	-	20		
B	20	-		
C			-	
D				-

Networks Section 2: Minimum Spanning Trees

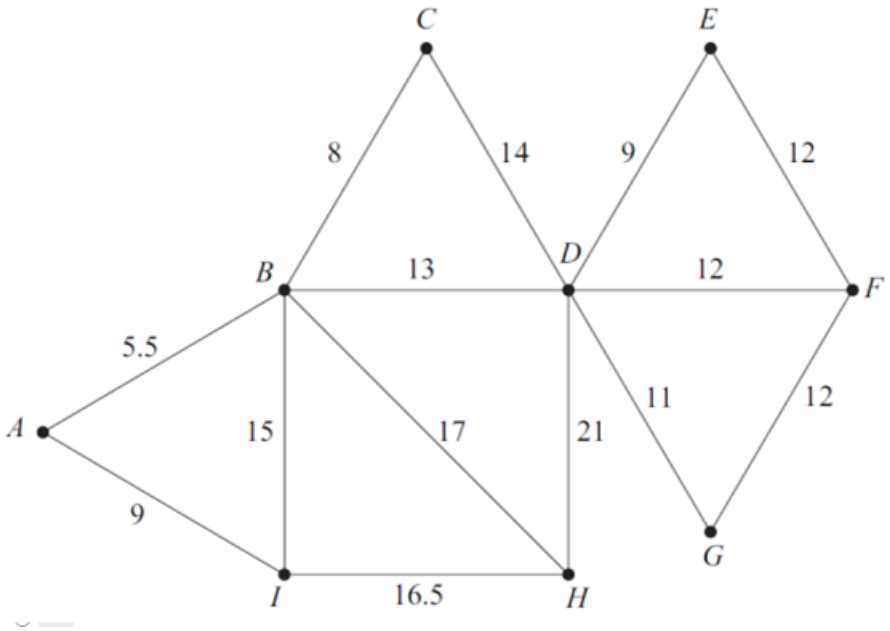
We know from Graphs Section 2: Types of Graphs that a tree is a type of graph that contains no cycles.

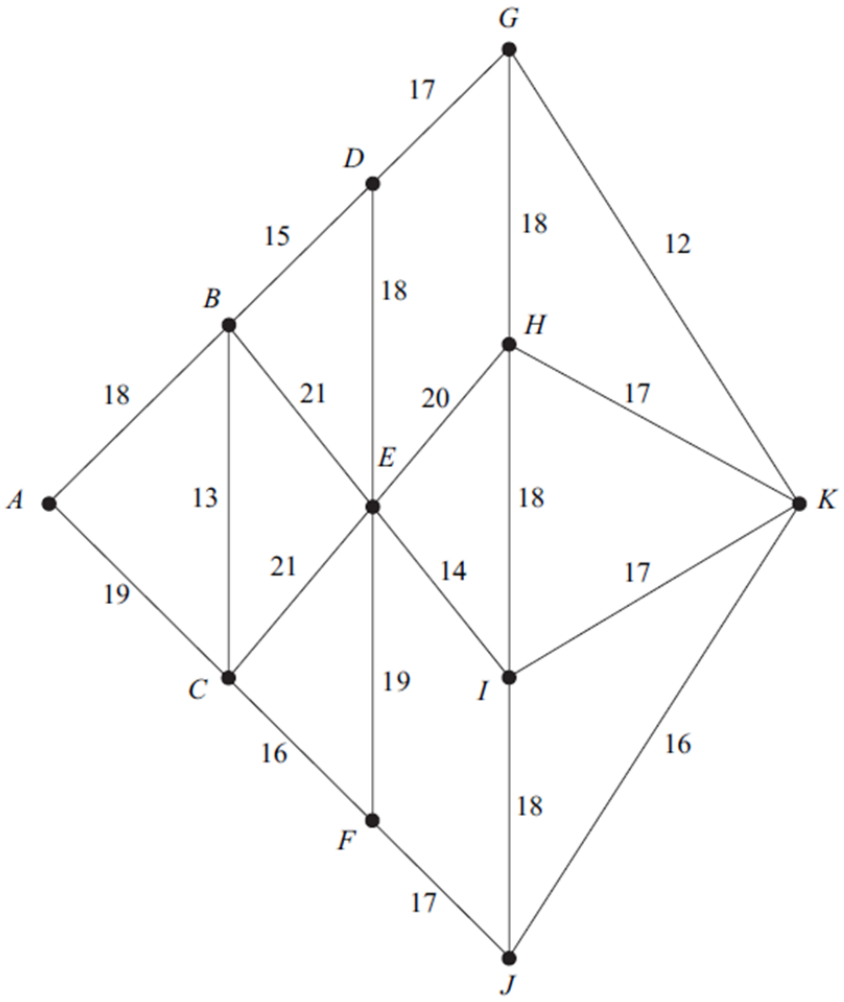
A Minimum Spanning Tree (MST) is the tree with minimum total weight that connects all nodes of a network. As part of discrete maths, we look at 2 algorithms to find the MST of a network.

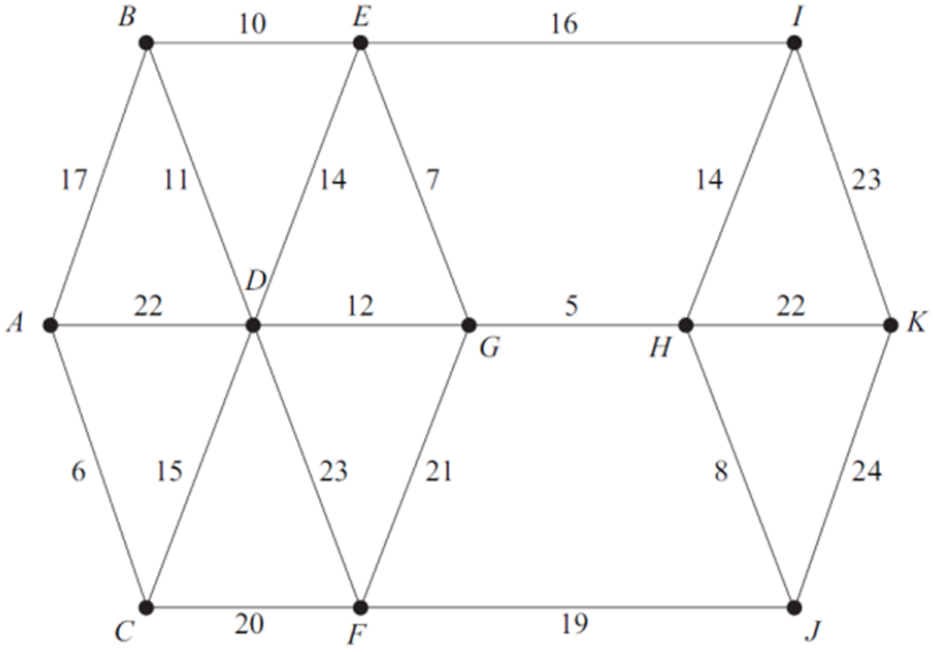
An algorithm is a specific set of instructions that we follow to achieve a desired outcome.

MST Algorithm 1: Kruskal's

1. List all the arcs in order of size
2. Pick the smallest one
3. Pick the next smallest one, providing it does not form a cycle anywhere
4. Repeat step 3 until all nodes are connected. If the number of nodes in the network is n then you will need to include n-1 arcs.




G
17
D
15
18
12
B
18
H
18
21
20
17
E
A
13
18
K
21
14
17
19
C
19
I
16
16
18
F
17
J

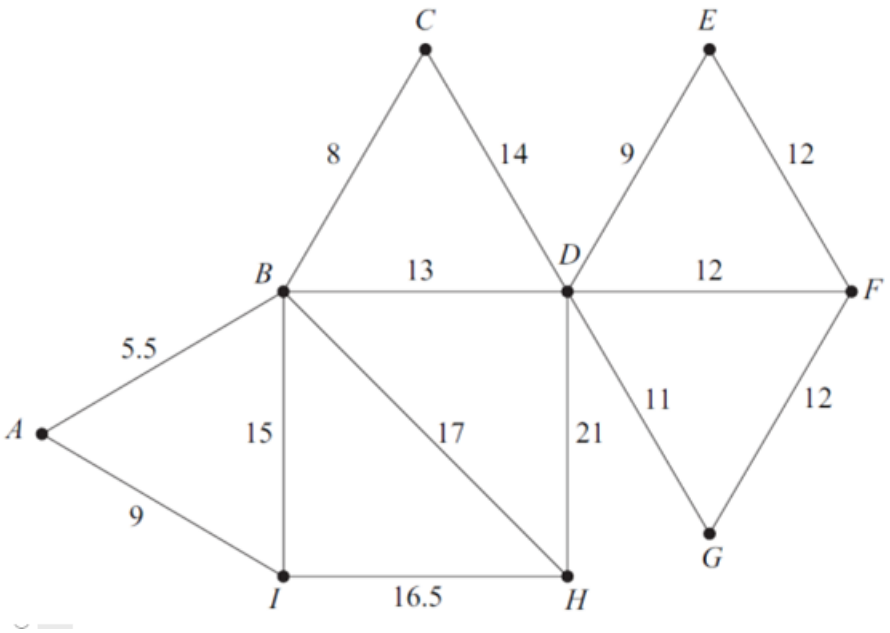

B
10
E
16
I
17
11
14
7
14
23
D
A
22
12
5
22
K
G
H
6
15
23
21
8
24
C
20
F
19
J

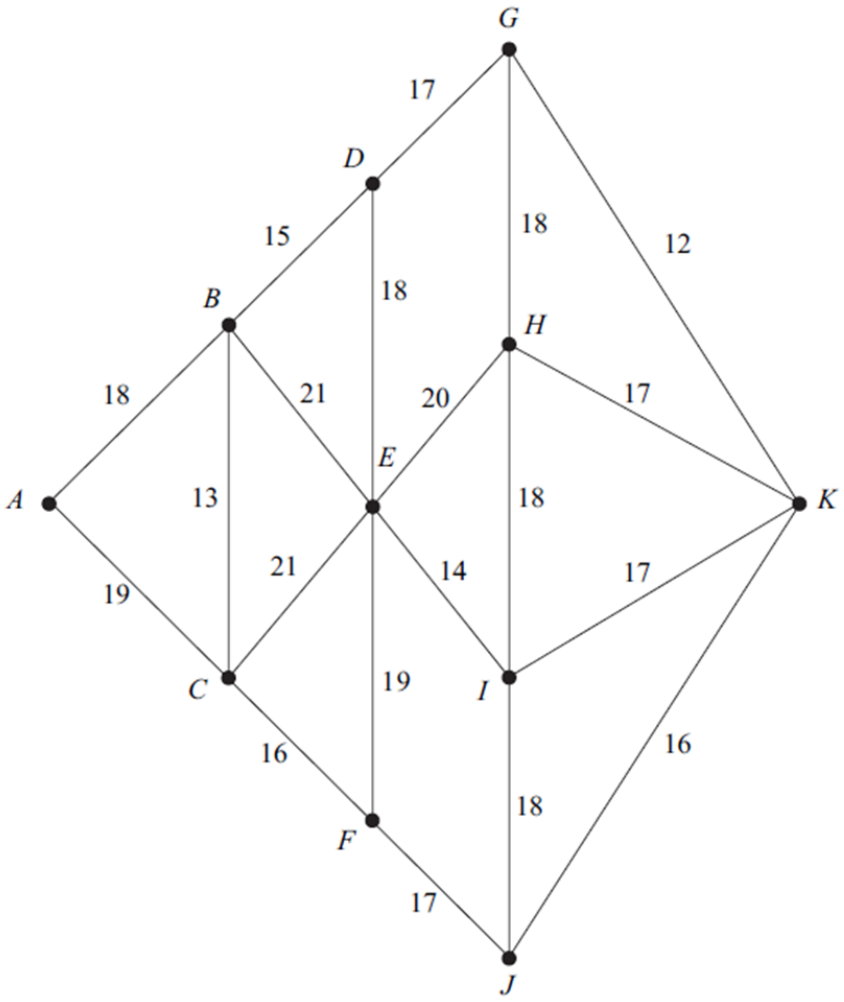
MST Algorithm 2a: Prim's (on a network)

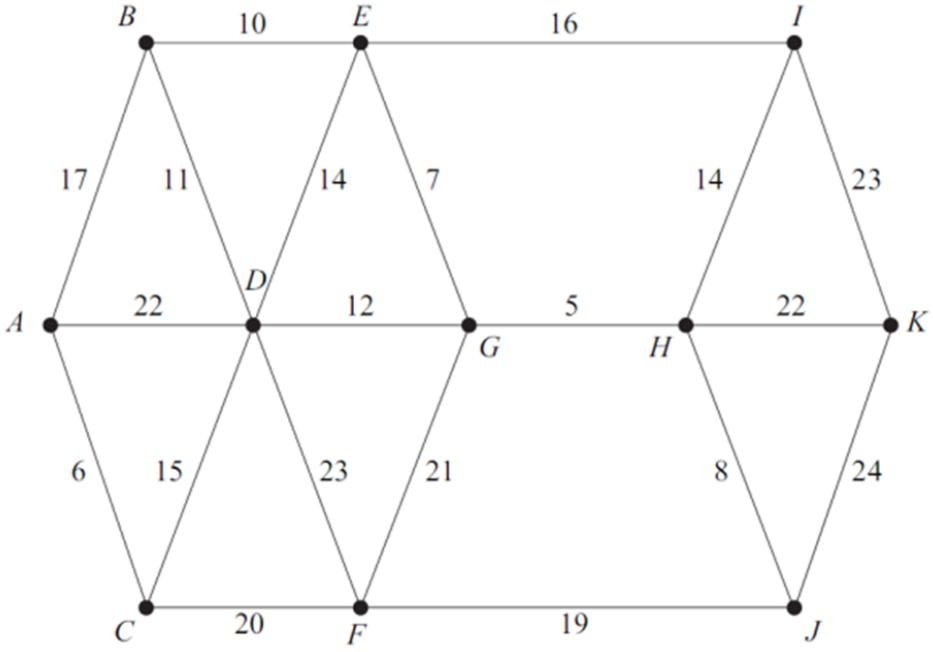
Prim's algorithm was first discovered by Vojtech Jarnik in 1930, later independently rediscovered by Robert Clay Prim in 1957, then again, independently, by Edsger Dijkstra in 1959.
Prim was born in Texas in 1921 and only died in 2012. He worked at Bell Laboratories with Joseph Kruskal where they both developed an algorithm for finding a minimum spanning tree on a weighted graph. Both algorithms serve the same purpose, but each uses a different method.

Prim's algorithm:

1. Select a starting node
2. Choose the shortest arc from that node
3. Continue to choose the shortest arc from any included node that does not form a closed cycle
4. Carry on until all nodes are joined. This will be when the number of arcs selected is 1 fewer than the number of nodes.




G
17
D
15
18
12
B
18
H
18
21
20
17
E
A
13
18
K
21
14
17
19
C
19
I
16
16
18
F
17
J


B
10
E
16
I
17
11
14
7
14
23
D
A
22
12
5
22
K
G
H
6
15
23
21
8
24
C
20
F
19
J

MST Algorithm 2b: Prim's (on a matrix)

Prim's algorithm can also be carried out on a matrix

Prim's algorithm:

1. Choose a starting node
2. Label that *column* 1 and cross out the *row* for that node
3. Find the lowest value in *any* labelled column
4. Circle the value, label the *column* for the node number 2 and cross out the *row* for that node
5. Repeat until all columns are labelled and all rows are crossed out

	U	V	W	X	Y	Z
U	-	27	14	12	13	30
V	27	-	21	23	25	31
W	14	21	-	29	16	18
X	12	23	29	-	20	24
Y	13	25	16	20	-	17
Z	30	31	18	24	17	-

	A	B	C	D	E
A	-	53	47	56	42
B	53	-	60	40	48
C	47	60	-	59	59
D	56	40	59	-	54
E	42	48	59	54	-

	A	***B***	***C***	***D***	***E***	***F***	***G***	***H***
A	–	15	10	12	16	11	14	17
B	15	–	15	14	15	16	16	15
C	10	15	–	11	10	12	14	9
D	12	14	11	–	11	12	14	12
E	16	15	10	11	–	13	15	14
F	11	16	12	12	13	–	14	8
G	14	16	14	14	15	14	–	13
H	17	15	9	12	14	8	13	–

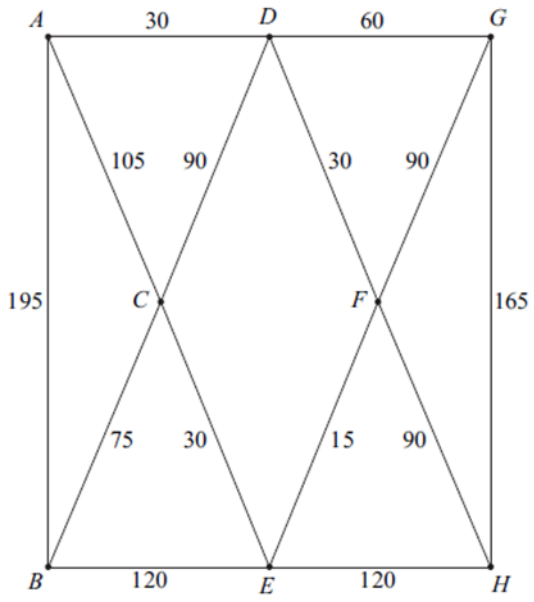
Networks Section 3: Route Inspection

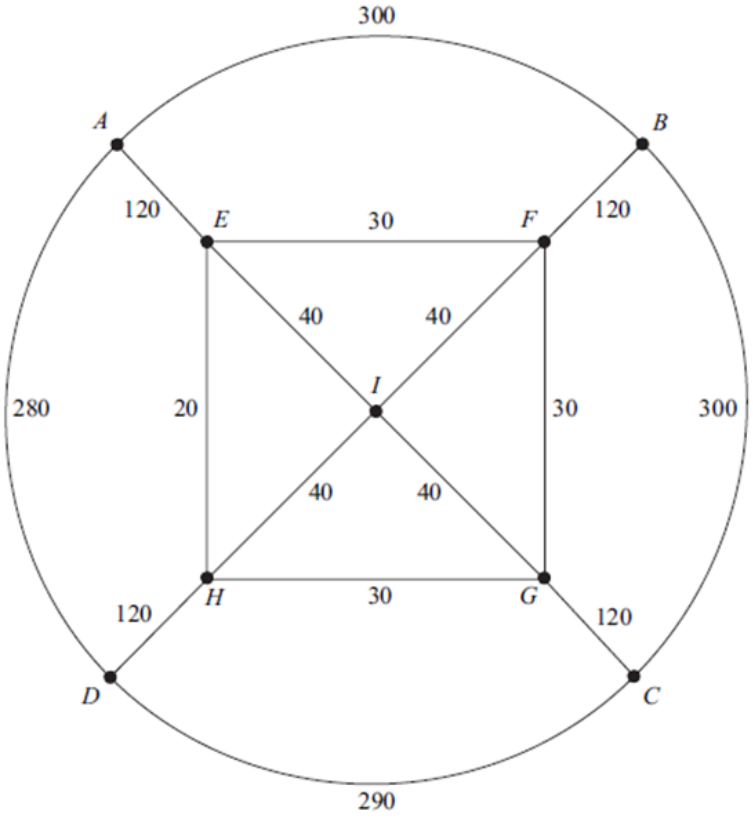
Sometimes, we need to find an Eulerian trail in a network that has the shortest distance. For example, a grit spreader who needs to cover all the roads in an estate, or a postman who needs to walk all the streets.

Chinese Postman Algorithm

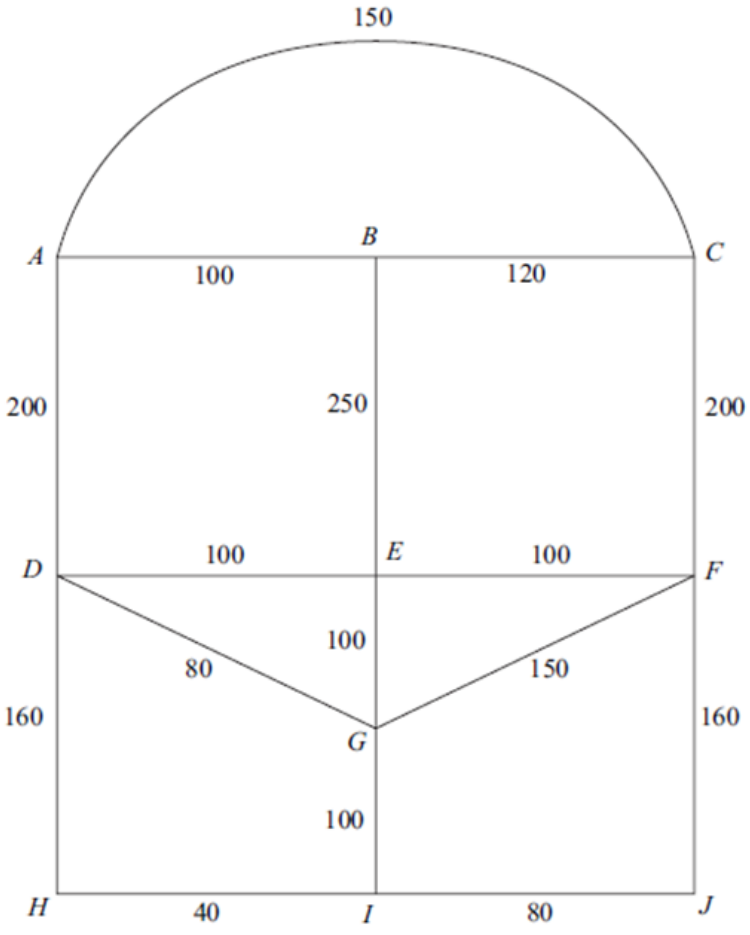
The Chinese Postman Algorithm is named for the Chinese Mathematician who studied the potential routes of a postman. The algorithm is:

1. Write down the orders of all the nodes
2. Identify the odd nodes
3. Find the shortest way of pairing them up
4. Duplicate these edges on the network
5. Find a path that visits all edges





Total Length = 1920 metres



Total = 2090

Networks Section 4: Travelling Salesperson

The **travelling salesperson** problem involves finding a Hamiltonian cycle of optimum weight within a network. One which visits all nodes before returning home, at minimum cost or distance.

An "ideal" algorithm for this problem has yet to be discovered, but there are a couple of algorithms we can use to find an upper bound and lower bound between which the answer to our problem must lie.

Nearest Neighbour Algorithm

1. Choose a starting node
2. Chose the smallest arc from *this node* to a node not yet in the tour
3. Repeat until all nodes are in the tour, then add an arc back to the starting node

This algorithm is far from perfect because the final step could involve choosing the most heavily weighted node in the entire network, and you would have no choice. But it is better than nothing. For this reason, the resulting cycle from this algorithm presents the **upper bound** for the traveling salesperson problem.

	B	*C*	*P*	*T*	*V*
B	–	43	57	52	18
C	43	–	18	13	56
P	57	18	–	8	48
T	52	13	8	–	51
V	18	56	48	51	–

This provides an upper bound because:

	P	Q	R	S	T
P	-	24	35	15	31
Q	24	-	45	18	38
R	35	45	-	32	41
S	15	18	32	-	25
T	31	38	41	25	-

	Grand Hotel (*G*)	**Aker Brygge (*A*)**	**National Theatre (*N*)**	**Parliament House (*P*)**	**Royal Palace (*R*)**
Grand Hotel (*G*)	–	165	185	65	160
Aker Brygge (*A*)	165	–	155	115	275
National Theatre (*N*)	185	155	–	205	125
Parliament House (*P*)	65	115	205	–	225
Royal Palace (*R*)	160	275	125	225	–

The Lower Bound

1. Delete one of the nodes and all attached arcs
2. Find the minimum spanning tree for the remaining nodes and arcs (using Prim's)
3. Reintroduce the deleted node using the two shortest arcs attached to it

	B	*C*	*P*	*T*	*V*
B	–	43	57	52	18
C	43	–	18	13	56
P	57	18	–	8	48
T	52	13	8	–	51
V	18	56	48	51	–

This provides an lower bound because:

	P	Q	R	S	T
P	-	24	35	15	31
Q	24	-	45	18	38
R	35	45	-	32	41
S	15	18	32	-	25
T	31	38	41	25	-

	Grand Hotel (*G*)	**Aker Brygge (*A*)**	**National Theatre (*N*)**	**Parliament House (*P*)**	**Royal Palace (*R*)**
Grand Hotel (*G*)	–	165	185	65	160
Aker Brygge (*A*)	165	–	155	115	275
National Theatre (*N*)	185	155	–	205	125
Parliament House (*P*)	65	115	205	–	225
Royal Palace (*R*)	160	275	125	225	–

	B	*C*	*P*	*T*	*V*
B	–	43	57	52	18
C	43	–	18	13	56
P	57	18	–	8	48
T	52	13	8	–	51
V	18	56	48	51	–

An upper bound for this network, found on page 22 is _____

A lower bound for this network, found on page 24 is _____

We can therefore say that the optimum travelling salesperson route has a total distance that lies between these two values.