Year 12 Mathematics

IAS 2.5

Use Networks in Solving Problems

Robert Lakeland & Carl Nugent

Contents

NuLake Ltd
Innovative Publisher of Mathematics Texts

NCEA 2 Internal Achievement Standard 2.5 – Networks

This achievement standard involves applying network methods in solving problems.

Achievement	Achievement with Merit	Achievement with Excellence
• Apply network methods in solving problems.	• Apply network methods, using relational thinking, in solving problems.	• Apply network methods, using extended abstract thinking, in solving problems.

- This achievement standard is derived from Level 7 of The New Zealand Curriculum, and is related to the achievement objective:
  - ❖ choose appropriate networks to find optimal solutions

  in the Mathematics strand of the Mathematics and Statistics Learning Area.

- Apply network methods in solving problems involves:
  - ❖ selecting and using methods
  - ❖ demonstrating knowledge of concepts and terms
  - ❖ communicating using appropriate representations.

  Relational thinking involves one or more of:
  - ❖ selecting and carrying out a logical sequence of steps
  - ❖ connecting different concepts or representations
  - ❖ demonstrating understanding of concepts
  - ❖ forming and using a model;

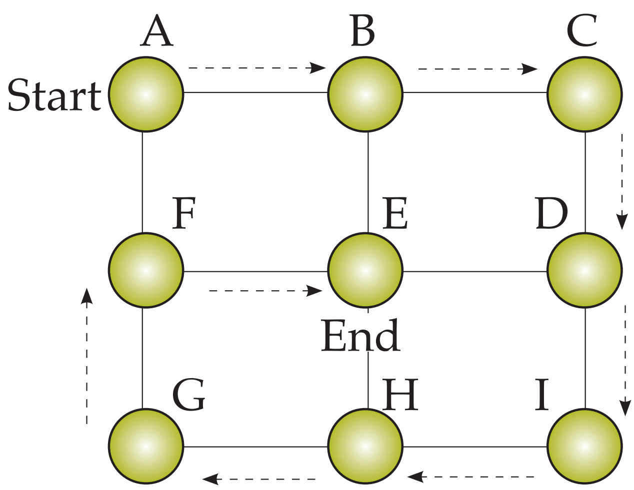
  and also relating findings to a context, or communicating thinking using appropriate mathematical statements.

  Extended abstract thinking involves one or more of:
  - ❖ devising a strategy to investigate a situation
  - ❖ identifying relevant concepts in context
  - ❖ developing a chain of logical reasoning, or proof
  - ❖ forming a generalisation;

  and also using correct mathematical statements, or communicating mathematical insight.

- Problems are situations that provide opportunities to apply knowledge or understanding of mathematical concepts and methods. Situations will be set in real-life or mathematical contexts.

- Methods include a selection from those related to:
  - ❖ shortest path
  - ❖ traversability
  - ❖ minimum spanning tree.

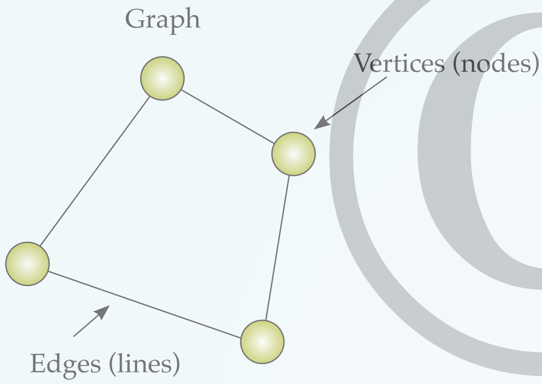


Introduction

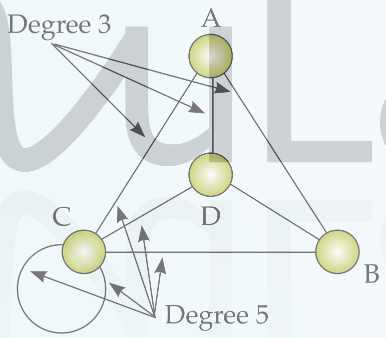
Graph Theory Basics

The graphs we will study in this Achievement Standard are different from those you have previously studied and drawn using grid paper.

In Graph Theory a graph consists of a set of vertices (nodes) and edges (straight or curved lines). Each edge joins one vertex (node) to another.

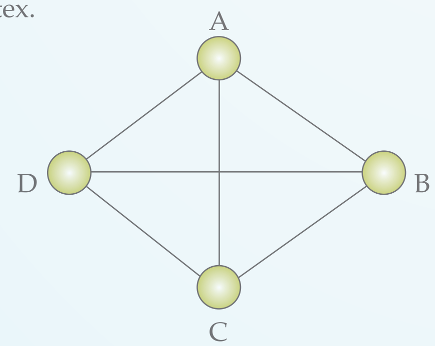


The degree of a vertex (node) is the number of edges (lines) which start or end at the vertex.



Vertex A has degree 3 because it has three edges (lines) 'coming or going' from it whereas vertex C has degree 5 because it has five edges (lines) 'coming or going' from it.

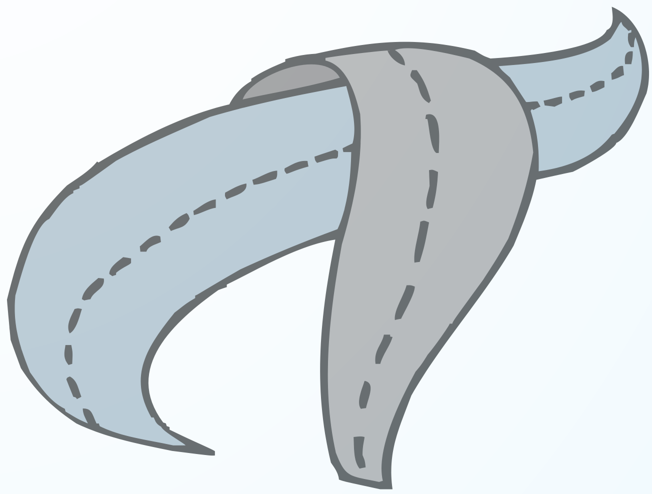
In some graphs it is possible that two lines will cross but unless they are marked as a vertex they should not be treated as such. See the diagram below. The intersection of lines AC and DB is not a vertex.



Extra Notes

In this Achievement standard reference is made to graph theory and graphs. Networks are an application of graph theory. The authors use the term, graph, as this is the accepted practice, but it is possible to call these graphs, networks, in order to differentiate them from Cartesian graphs.

Think of two lines that cross, but not at a vertex, as similar to an underpass on a motorway.

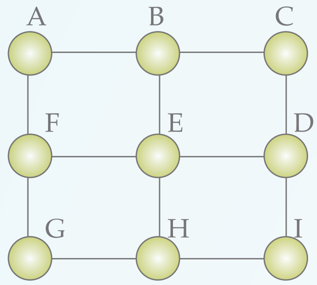


Hamiltonian Paths and Circuits

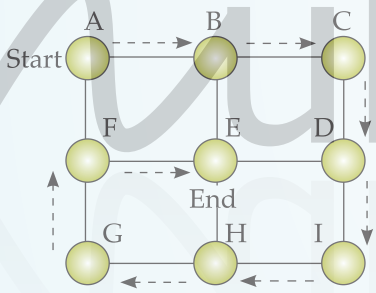
A Hamiltonian path, named after the Irish mathematician Sir William Rowan Hamilton, is one that visits each vertex (node) only once. Each edge does not have to be traversed.

A Hamiltonian circuit or cycle is one that visits each vertex (node) only once **but** also returns to its starting vertex (node). Each edge does not have to be traversed in a Hamiltonian circuit.

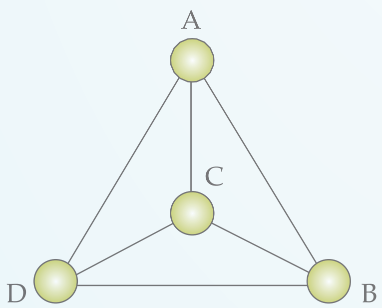
An example of a Hamiltonian path is drawn below.



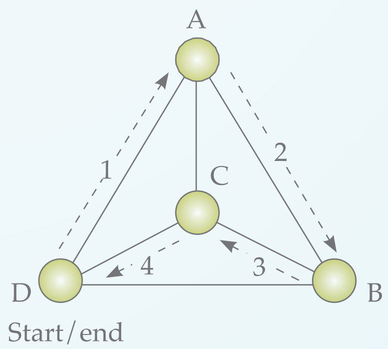
A possible path that visits each vertex (node) only once but does not return to the start is drawn below. Note also that not all edges have been traversed which is not a requirement of a Hamiltonian path.



An example of a Hamiltonian circuit is drawn below.



A possible path that starts and ends at the same point and visits each vertex (node) is drawn on the right. Note also that not all edges have been traversed which is not a requirement for a Hamiltonian circuit.



Extra Notes

A circuit (cycle) is a closed path, i.e. it finishes where it starts.

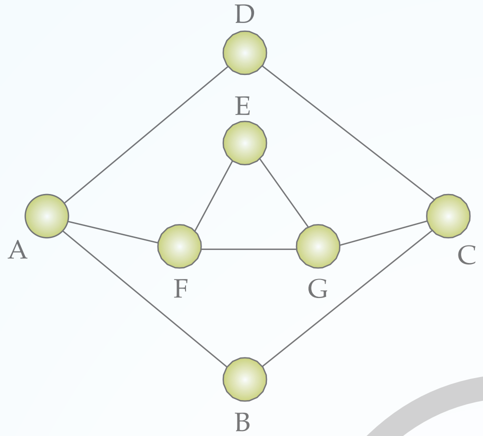
Hamiltonian circuits are useful in applications where it is necessary to visit each 'vertex' for example, a salesman's clients in a town.

Unfortunately there is not an equivalent formula to Euler's to identify whether or not a graph has a Hamiltonian path or circuit.

In an Euler path you can visit each vertex (node) more than once but each edge only once. In a Hamiltonian path you can only visit each node once.

In problems involving Hamiltonian circuits there are many equivalent reversal routes. When we don't want to count reversal routes we use the term 'distinct' routes.

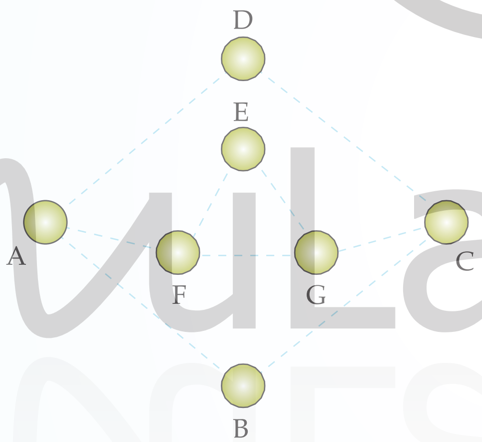
37.



Circuit or path?

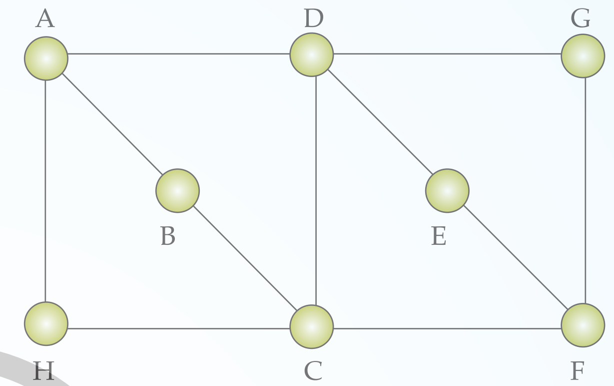
Justification:

Draw the Hamiltonian path or circuit below.
Label your start and end points.
A line that you mark as solid will be one that you need not traverse.



Path/Circuit:

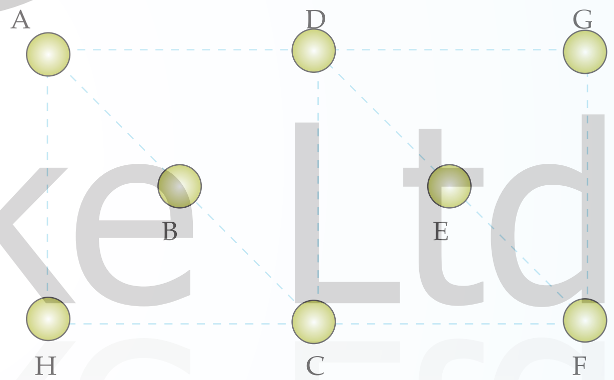
38.



Circuit or path?

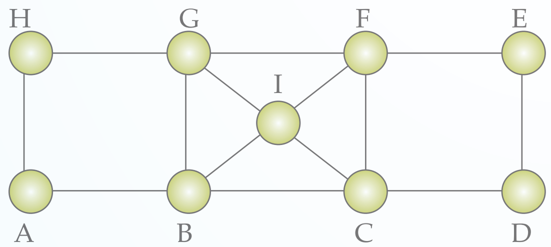
Justification:

Draw the Hamiltonian path or circuit below.
Label your start and end points.
A line that you mark as solid will be one that you need not traverse.



Path/Circuit:

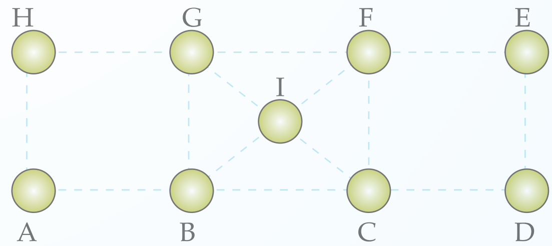
39.



Circuit or path?

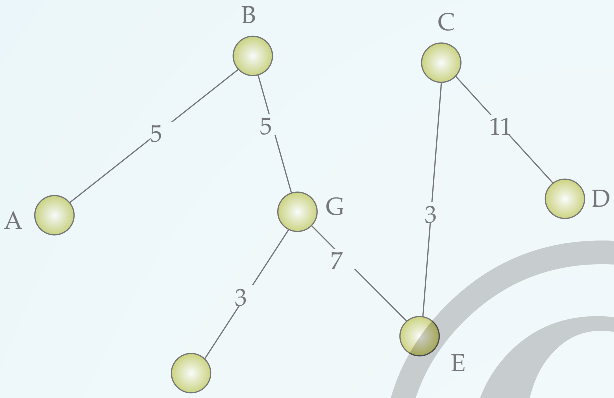
Justification:

Draw the Hamiltonian path or circuit below.
Label your start and end points.
A line that you mark as solid will be one that you need not traverse.



Path/Circuit:

Kruskal's Algorithm cont...



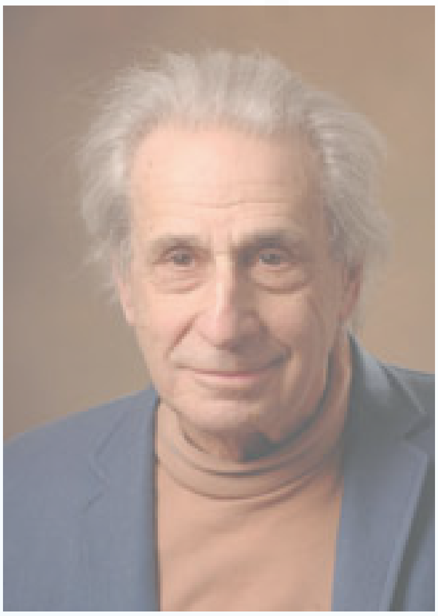
Every suburb (node) of our graph is now touched by a link (fibre optics). This is now our minimum spanning tree with a minimum cost of:

50 000 + 50 000 + 30 000 + 70 000 + 30 000 + 110 000

= $340 000

When using Kruskal's Algorithm, if there are two equal cheapest links to choose from, it will not matter which one you select. The same minimum spanning tree will be generated but in a different order.

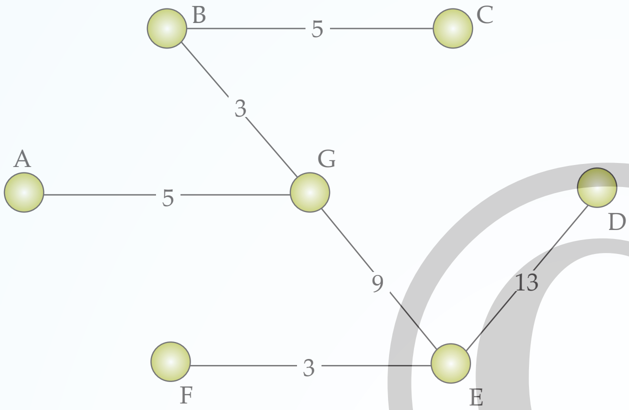
Kruskal's Algorithm is often called a 'Greedy Algorithm'. It repeatedly adds the next cheapest link that doesn't produce a cycle. It was named after Joseph Kruskal an American mathematician who lived from 1928 – 2010.



Extra Notes

cont...

The completed minimum spanning tree has a total minimum weight of 5 + 5 + 3 + 9 + 3 + 13 = 38.



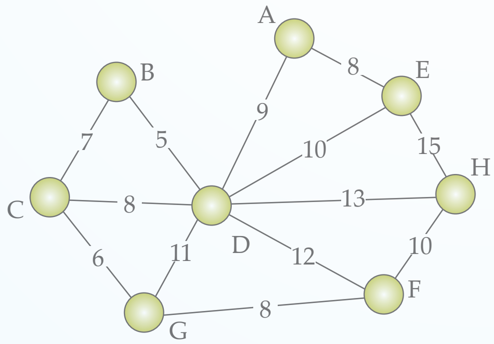
Example

A farmer wishes to develop a system of tracks around his farm so that he can service all parts of the farm. The distance in hundreds of metres between key points on the farm is given in the table below.

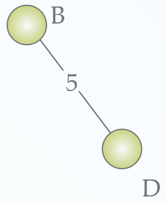
Draw up a graph first representing this situation and then use Kruskal's Algorithm to find a minimum spanning tree for the farmer so he can develop a system of tracks covering the minimum distance.

	A	B	C	D	E	F	G	H
A	-	-	-	9	8	-	-	-
B	-	-	7	5	-	-	-	-
C	-	7	-	8	-	-	6	-
D	9	5	8	-	10	12	11	13
E	8	-	-	10	-			15
F	-	-		12	-	-	8	10
G	-	-	6	11	-	8	-	-
H	-	-	-	13	15	10	-	-

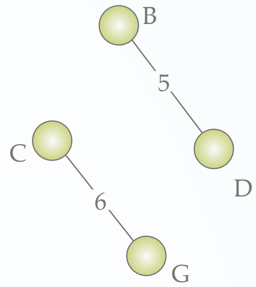
Drawing up a graph using the distances from the table we get the following.



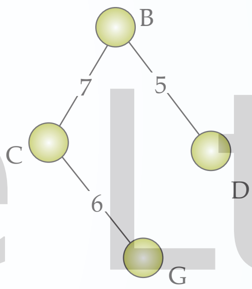
We begin by finding the shortest piece of track which is B to D, with a weight of 5 (500 metres).



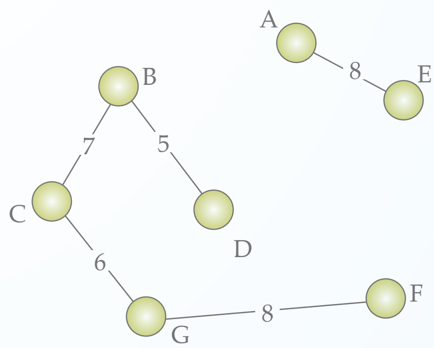
The next shortest piece of track is C to G with a distance of 600 metres so we include this link.



The next shortest piece of track is C to B with a distance of 700 metres so we add this link.



The next shortest pieces of track are C to D, A to E and G to F all with a distance of 800 metres. We can not add C to D as we would form a cycle within the tree so we choose to add G to F and then A to E.



The next shortest pieces of track is D to A with a distance of 900 metres so we add this link.

64. A plumber has to run pipes from a series of nine points in a building.
A table of the estimated time, in hours, to lay the pipes between each point is given below.

First draw up a graph representing this situation and then use Kruskal's Algorithm to find a minimum spanning tree for the piping and then calculate its minimum laying time.

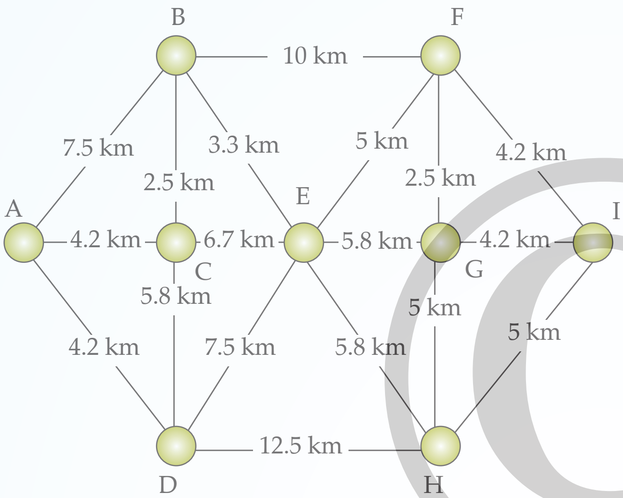
	A	B	C	D	E	F	G	H	I
A	-	5	-	-	-	-	-	9	-
B	5	-	9	-	-	-	-	12	-
C	-	9	-	8	-	5	-	-	3
D	-	-	8	-	10	15	-	-	-
E	-	-	-	10	-	11	-	-	-
F	-	-	5	15	11	-	3	-	-
G	-	-	-	-	-	3	-	2	7
H	9	12	-	-	-	-	2	-	8
I	-	-	3	-	-	-	7	8	-

65. A installer has to lay TV cabling to a neighbourhood of nine houses.
A table of the estimated times, in hours, to lay the cabling between each house is given below.

First draw up a graph representing this situation and then use Kruskal's Algorithm to find a minimum spanning tree for the cabling and then calculate its minimum installing time.

	A	B	C	D	E	F	G	H	I
A	-	10	-	11	15	-	-	-	-
B	10	-	17	-	3	9	-	-	-
C	-	17	-	-	-	12	-	-	-
D	11	-	-	-	14	-	21	-	-
E	15	3	-	14	-	16	19	4	-
F	-	9	12	-	16	-	-	8	2
G	-	-	-	21	19	-	-	5	-
H	-	-	-	-	4	8	5	-	7
I	-	-	-	-	-	2	-	7	-

b) A large water reservoir at village A is to supply water to all the villages. Use Kruskal's algorithm to find a minimum spanning tree for the network of pipes and calculate the total amount of piping required.



c) A traffic officer leaves from village A in the morning and must drive all the roads between the villages as part of his 'run'. Give a route for the traffic officer so that he visits each road only once and ends up at village I.

d) What is the total distance the traffic officer travels in a day if he returns to village A using the quickest possible route?

Page 12

22. Even vertices: A, B, C, D, E, F, G
Odd vertices: None
Classification: Euler circuit
Reason: All vertices are even.

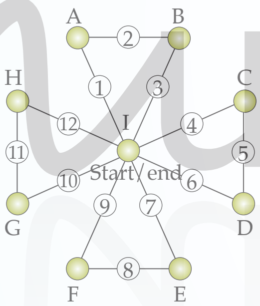
23. Even vertices: A, C, E, F, G
Odd vertices: B, D
Classification: Euler path
Reason: Has two odd vertices and some even vertices.

24. Even vertices: A, B
Odd vertices: C, D
Classification: Euler path
Reason: Has two odd vertices and some even vertices.

25. Even vertices: C, D
Odd vertices: A, B, E, F
Classification: Neither
Reason: More than two odd vertices so neither.

Page 14

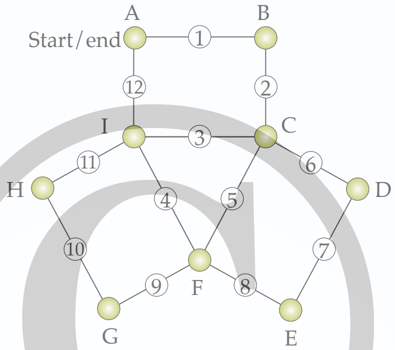
26. Even vertices: A, B, C, D, E, F, G H, I
Odd vertices: None
Traversable: Yes
Start and End points: Start I
End I



27. Even vertices: C, D, E, H
Odd vertices: A, B, F, G
Traversable: No
Start and End points: N/A

Page 15

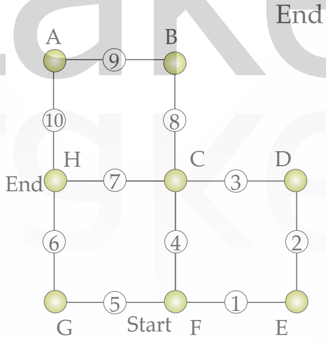
28. Even vertices: A, B, C, D, E, F, G H, I
Odd vertices: None
Traversable: Yes
Start and End points: Start A
End A



29. Even vertices: A, D, E, H
Odd vertices: B, C, F, G, I, J
Traversable: No
Start and End points: N/A

Page 16

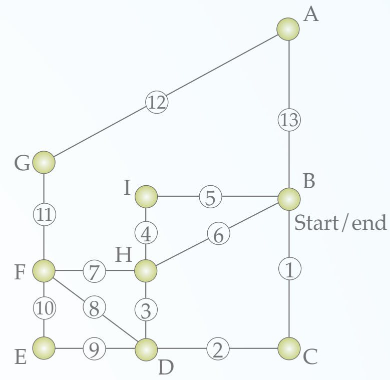
30. Even vertices: A, B, C, D, E, G,
Odd vertices: F, H
Traversable: Yes
Start and End points: Start F
End H



31. Even vertices: A, B, C, D, F, H, K
Odd vertices: E, G, I, J
Traversable: No
Start and End points: N/A

Page 17

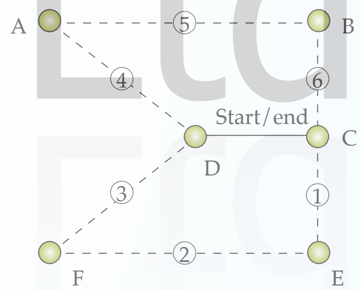
32. Even vertices: A, B, C, D, E, F, G H, I
Odd vertices: None
Traversable: Yes
Start and End points: Start B
End B



33. Even vertices: A, B, D, E, H, J
Odd vertices: C, F, G, I
Traversable: No
Start and End points: N/A

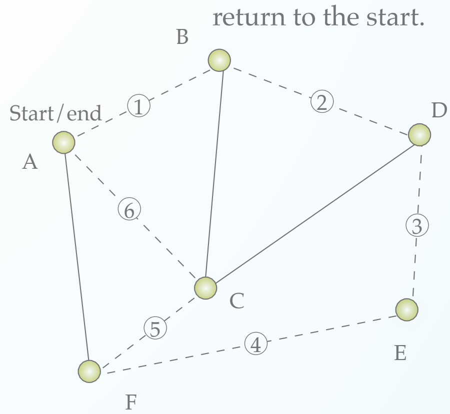
Page 20

34. Circuit/path: Hamiltonian circuit
Justification: Possible to visit each vertex and return to the start.



Path: C, E, F, D, A, B, C.

35. Circuit/path: Hamiltonian circuit
Justification: Possible to visit each vertex and return to the start.



Path: A, B, D, E, F, C, A.