

MA014G
Algebra and Discrete Mathematics A

Lecture Notes 6
Autumn 2007

Pia Heidtmann
070811

GRAPH THEORY EXAMPLES

Some graphs from real life:

① vertices: stations on the London Underground

edge between two stations \Leftrightarrow there are no stops in between the two stations.

② vertices: classes needed to be scheduled

edge between two classes \Leftrightarrow some students/lecturer must attend both

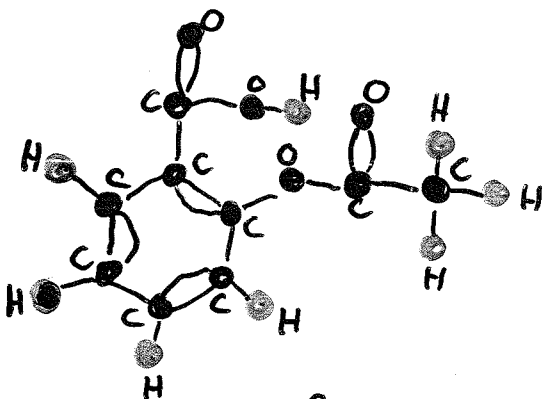
③ vertices: cities on a map of Sweden

edge between two cities \Leftrightarrow there is a road from one to the other which goes through no other city.

You could even label the edge with the distance between the two cities via this road.

④ vertices = atoms in a molecule

edge between two atoms \Leftrightarrow there is a bond between the two atoms.



- - oxygen degree 2
- - carbon degree 4
- - Hydrogen degree 1

An aspirin molecule.

Defining a graph

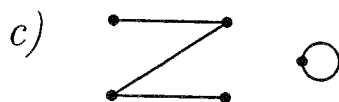
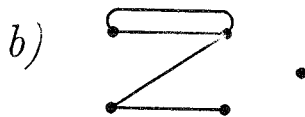
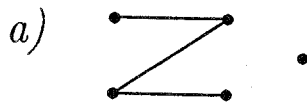
A graph consists of a set of points, called ^{hörn}**vertices**, and a set of lines connecting these vertices, called ^{kanter}**edges**.

If two or more edges are joining the same pair of vertices, this is called a ^{multipel kant}**multiple edge**.

An edge that connects a vertex to itself is called a ^{loop}**loop**.

A ^{enkel}**simple graph** is a graph *without* loops or multiple edges.

Example 6.1 *Example of a simple graph, a graph with multiple edges, and a graph with a loop:*



Note: unless otherwise specified, when we refer to a graph, we mean a simple graph.

Formal definition of a simple graph

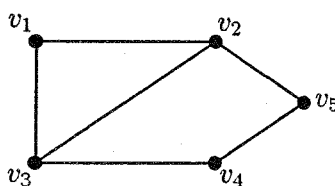
A simple graph $G = (V, E)$ consists of a non-empty set V of points, called **vertices** and a set E of unordered pairs of vertices called **edges**.

Example 6.2 *For the following graph the vertex set*

$$V = \{v_1, v_2, v_3, v_4, v_5\}$$

and the edge set is

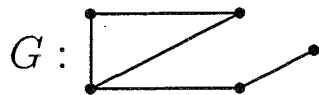
$$E = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_5), (v_3, v_4), (v_4, v_5)\}.$$



Note that the edges are unordered pairs of vertices, so for example the edge (v_1, v_3) could also have been written as (v_3, v_1) .

~~defn.~~
 A *subgraph* H of a graph G is a graph whose sets of vertices and edges are subsets of the sets of vertices and edges respectively of G .

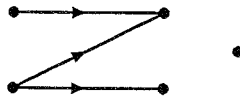
Example 6.3 *In the following, H is a subgraph of G*



~~directed graph~~
Directed graphs

A directed graph (or ^{digraph}digraph) is a graph in which every edge is given a direction. These directed edges are called ^{edges}arcs.

Example 6.4 *Example of a digraph:*

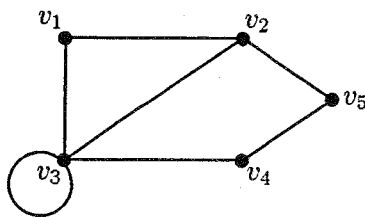


The Degree of a Vertex

Two vertices are said to be **adjacent** if there is an edge that joins them. An edge e is said to be **incident** to a vertex v if e joins v to some other vertex.

Each vertex in a graph has a **degree**; this is the number of edges that are incident to it. If a vertex v has degree r , we write $\delta(v) = r$.

Example 6.5 Any number of vertices in a graph may have the same degree.



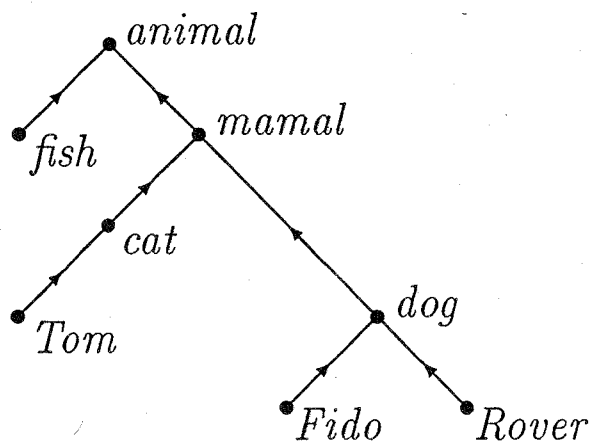
$$\delta(v_1) = 2, \quad \delta(v_2) = 3, \quad \delta(v_3) = 5, \quad \delta(v_4) = 2, \quad \delta(v_5) = 2$$

Note that this graph is not simple. The loop contributes 2 towards the degree of v_3 . This is because we think of the loop as being an edge (v_3, v_3) .

In a digraph each vertex has two types of degree; *indegree* and *outdegree*. The indegree is the number of arcs coming into the vertex while the outdegree is the number of arcs leaving the vertex.

Graphs can be used to model a number of situations in the real world.

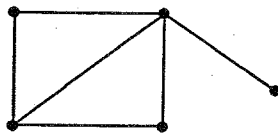
Example 6.6 *Graphs can be used to represent real world relationships such as 'is a':*



^{gradfølgen} The Degree Sequence

The degree sequence of a graph is the list of the degrees in non-decreasing order.

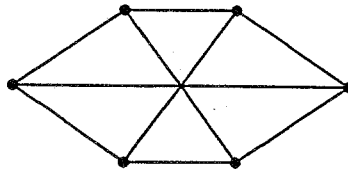
Example 6.7 *Degree sequences:*



The degree sequence is: 1, 2, 2, 3, 4

A ^{regulär} **regular** graph is a graph in which every degree is the same.

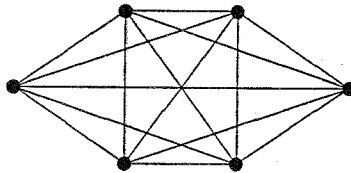
Example 6.8 *A regular graph:*



This graph is 3-regular, i.e. every vertex has degree 3.

The ^{vollständige grafen} complete graph K_n is simply the (simple) graph on n vertices where every pair of distinct vertices are joined by an edge.

Example 6.9 *The complete graph K_n is $(n - 1)$ -regular. Here is K_6 :*



This graph is 5-regular.

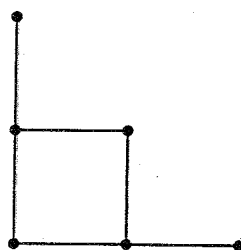
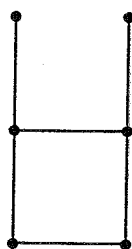
Theorem (Hand Shaking Lemma) [2] Thm. 6.2.21

In any graph the sum of the degree sequence is $2 \times$ the number of edges.

Thus a graph with degree sequence 1,1,2,2,3,3 must have 6 edges, as the sum of the degree sequence is 12.

Note that the degree sequence does not uniquely define the graph. Two different graphs may have the same degree sequence, as we shall see in the next example.

Example 6.10 *Two different graphs that both have degree sequence 1,1,2,2,3,3:*



Example 6.11 *Prove that there cannot be a graph with degree sequence $2, 2, 3, 3, 3$.*

Solution

The sum of the degrees is $2+2+3+3+3=13$. This is odd but the sum of the degree sequences of a graph is twice the number of edges and so is even. Therefore this degree sequence cannot represent a graph.

Example 6.12 *Prove that there cannot be a graph with degree sequence $0, 2, 2, 3, 4, 5$.*

Solution

Only five of the vertices are connected by one or more edges and thus each vertex can be connected to at most four other vertices. This means that the degree of a vertex can be at most four, but one vertex has degree five. This contradicts this degree sequence coming from a graph.

Corollary 1

The sum of the degree sequence of a graph is an even number.

Corollary 2

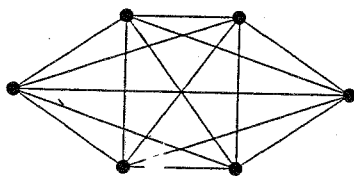
In any graph the number of vertices of odd degree is even.

Example 6.13

If G is a regular graph of degree r on n vertices, then G has $nr/2$ edges.

The complete graph K_n is $(n - 1)$ -regular and thus has $n(n - 1)/2$ edges.

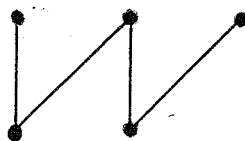
So for example K_6 has 15 edges.



bipartite graph
Bipartite graphs

A bipartite graph is a graph in which the vertices can be partitioned into two sets such that each vertex in either part is connected only to vertices in the other part.

Example 6.14 *A ^{bipartite} bipartite graph:*



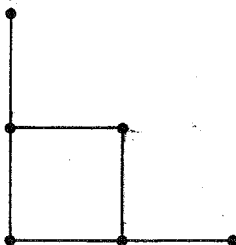
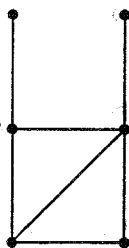
An easy way to test if a given graph G is bipartite is to use the following result:

Theorem

A graph G is bipartite if and only if there is a 2-colouring of the vertices of the graph so that no two adjacent vertices have the same colour.

A 2-colouring is just a colouring using 2 colours only.

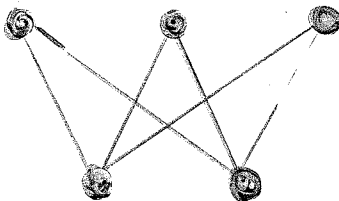
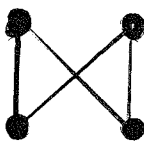
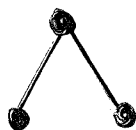
Example 6.15 *Are these graphs bipartite?*



A bipartite graph is a graph in which the vertices can be partitioned into two sets V_1 and V_2 such that each vertex in either part is connected only to vertices in the other part.

The bipartite graph is said to be a ^{fullständiga bipartit graf} **complete bipartite graph** if every vertex of V_1 is adjacent to every vertex of V_2 . If $|V_1| = m$ and $|V_2| = n$ we call the graph $K_{m,n}$.

Example 6.16 Here are $K_{1,2}$, $K_{2,2}$ and $K_{2,3}$.



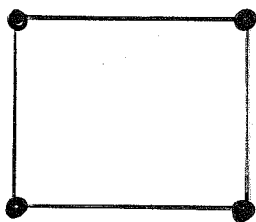
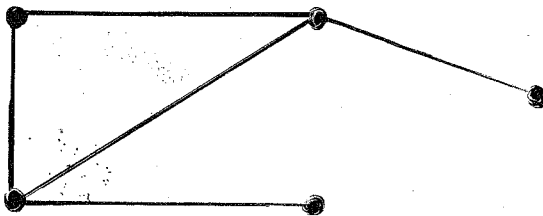
The complement of a graph.

The complement \overline{G} of a graph $G = (V, E)$ is the graph $\overline{G} = (V, \overline{E})$, where \overline{E} is the set of all edges that are *not* in E .

That is

$$(v_1, v_2) \in \overline{E} \iff (v_1, v_2) \notin E.$$

Example 6.17 Find the complement of the following two graphs.



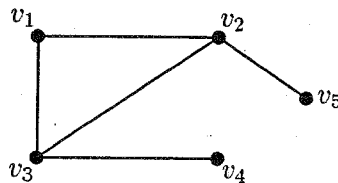
graph matrix

Adjacency Matrices

It is possible to represent a graph with n vertices by an $n \times n$ matrix that states which vertices are adjacent; this is called an adjacency matrix.

If two vertices v_i and v_j are connected in the graph then there is a 1 placed in the positions (i, j) and (j, i) in the adjacency matrix, otherwise these locations contain a 0. The matrix is therefore symmetric about the diagonal.

Example 6.18 For the graph $G = (V, E)$ where
 $V = \{v_1, v_2, v_3, v_4, v_5\}$ and
 $E = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_5), (v_3, v_4)\}$.



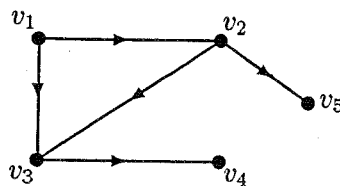
the adjacency matrix is:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Adjacency Matrices for digraphs

The concepts of an adjacency matrix ~~for a graph~~ for a graph can be generalised to include digraphs.

Example 6.19 *For the following digraph:*



The adjacency matrix is:

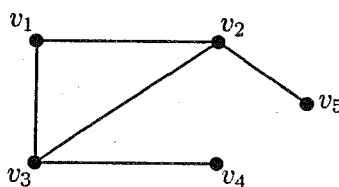
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

For a digraph the adjacency matrix need not be symmetric about the diagonal.

Result

The sum of the entries in the i th row (or column) of the adjacency matrix of a simple graph G is the degree of the vertex v_i corresponding to that row (or column).

Example 6.20 *For the graph $G = (V, E)$ where $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $E = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_2, v_5), (v_3, v_4)\}$.*



we found the adjacency matrix to be

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

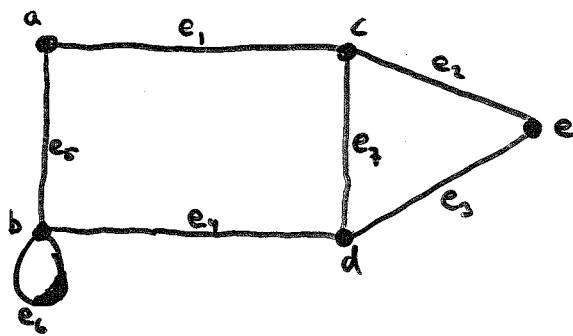
Note that the sum of the entries in row/column 2 is 3 because $\delta(v_2) = 3$.

The incidence matrix of a graph G with n vertices and m edges is an $n \times m$ -matrix whose rows are labelled by the vertices of G and whose columns are labelled by the edges of G .

The entry in row i , column j is 1 if vertex i is on edge j , and 0 otherwise.

EXAMPLE

The following graph



has incidence matrix

$$\begin{matrix} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{matrix}$$

Note that in a graph without loops each column has precisely two 1s.

(A column corresponding to a loop has just one 1.)

PATHS and CYCLES

A ^{simple} path in a graph is an alternating sequence of vertices and edges

$$(v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_5, \dots, e_{k-1}, v_k)$$

such that $e_i = (v_i, v_{i+1})$ is an edge of the graph for all $1 \leq i \leq k-1$

Example

$$(a, e_1, b, e_2, c, e_3, b, e_4, e)$$

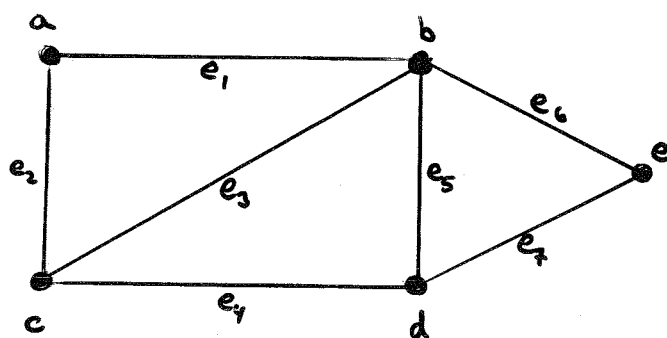
is a path from vertex a to vertex e in the graph below.

When the graph is simple, we do not list the edges in the path, so for example the path from the example above would be given as

$$(a, b, c, b, e)$$

A simple path in a graph is a path with no repeated vertices.

Example (a, b, c, d, e) is a simple path in the graph below.



The length of a (simple) path is the number of edges in the path

A ^{cycle} in a graph is a path (v_1, v_2, \dots, v_k) where $v_1 = v_k$ which has no repeated edges and whose length ≥ 1 .

Example

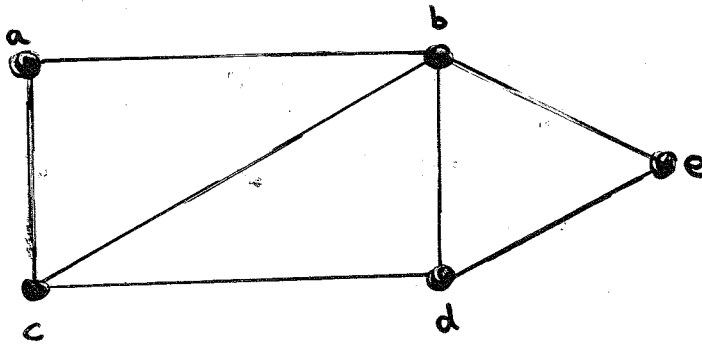
(a, b, d, e, b, c, a)

is a cycle in the graph below.

A simple cycle in a graph is a cycle (v_1, v_2, \dots, v_k) with no repeated vertex except $v_1 = v_k$.

Example

(a, b, c, a) is a simple cycle in the graph below



CONNECTED GRAPH

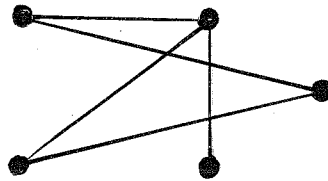
A graph G is ^{zusammenhängend} connected if there is a path between every pair of vertices of G .

A graph which is not connected is said to be ^{nicht-zusammenhängend} disconnected, such a graph can be partitioned into connected subgraphs called components.

^{komponenten}

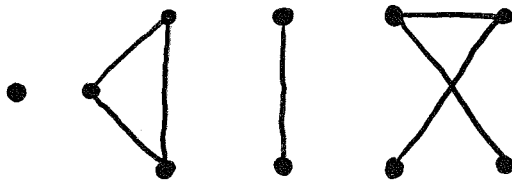
Example

A connected graph



Example

A disconnected graph with 4 components:

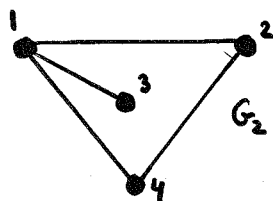
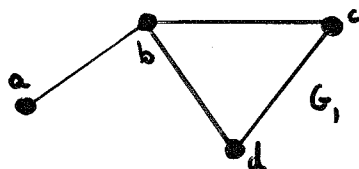


isomorphi:
ISOMORPHISM OF GRAPHS

When are two graphs the 'same'?

EXAMPLE

The two graphs



can be considered to be 'the same' in the following way:
If we relabel the vertices of G_1 , we get G_2 such that
the edges in G_1 correspond to the edges in G_2 .

$$b \rightarrow 1 \quad a \rightarrow 3 \quad c \rightarrow 2 \quad d \rightarrow 4$$

edge (a,b) corresponds to $(1,3)$

edge (b,c) corresponds to $(1,2)$

edge (b,d) corresponds to $(1,4)$

edge (c,d) corresponds to $(2,4)$

DEFINITION

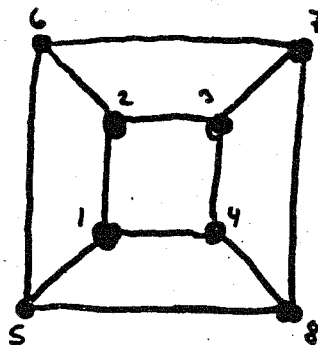
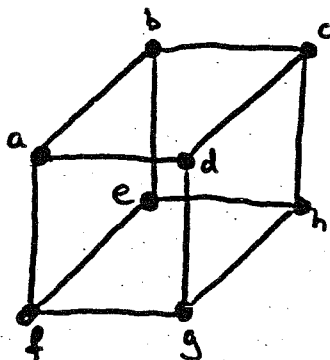
Let G_1 and G_2 be two simple graphs.

G_1 and G_2 are said to be ^(isomorphic) isomorphic if and only if
there is a bijection φ between the vertices of G_1 and
the vertices of G_2 such that

if (x,y) is an edge of G_1 , then $(\varphi(x), \varphi(y))$ is an edge of G_2 .

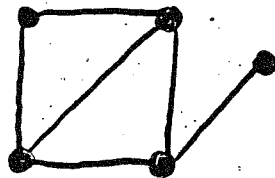
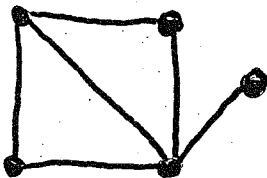
Example

Which of the following pairs of graphs are isomorphic?

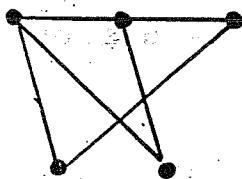
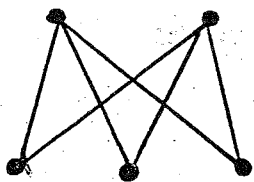


yes

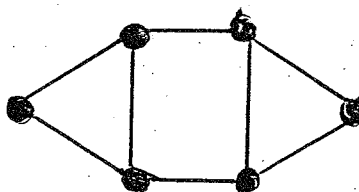
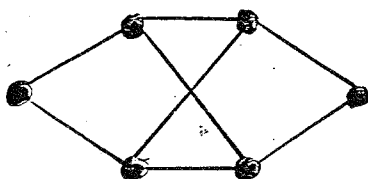
a	→	1
b	→	2
c	→	3
d	→	4
e	→	6
f	→	5
g	→	8
h	→	7




no, they have not got the same degree sequence



no, one is bipartite, the other one is not



no, one has  as subgraph, the other has not.

Result

If two graphs G_1 and G_2 are isomorphic, then

- they have the same number of vertices
- they have the same number of edges
- they have the same degree sequence.
- they have the same structure, so if one is
 - bipartite, so is the other
 - if one has a \triangle -subgraph, so has the other
 -
 -
 -

! Note The above result is only useful for proving that two graphs are non-isomorphic, if one of the bullet points is not satisfied.

The trouble is that two graphs satisfying all bullet points may still be non-isomorphic.

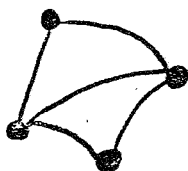
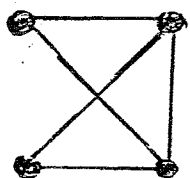
To show that they are isomorphic we must set up a bijection between vertices mapping edges to edges.

Result

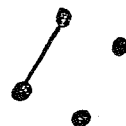
Two graphs are isomorphic \iff their complements are isomorphic

Example

- This result is useful if the graphs have a lot of edges:



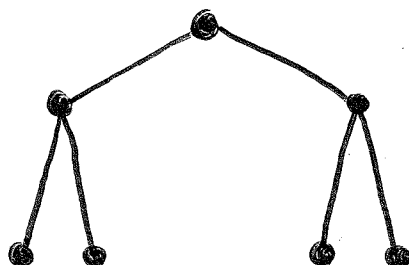
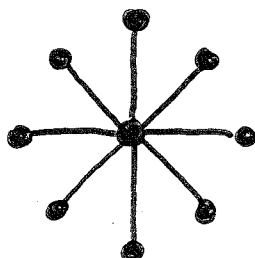
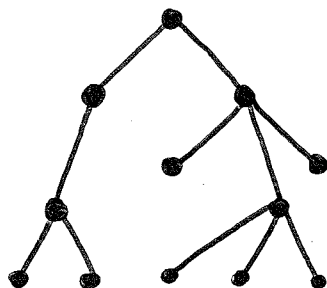
these are isomorphic as both have complement



TREES

A tree is a connected graph with no cycles.

EXAMPLES



trees

THEOREM

The following are equivalent:

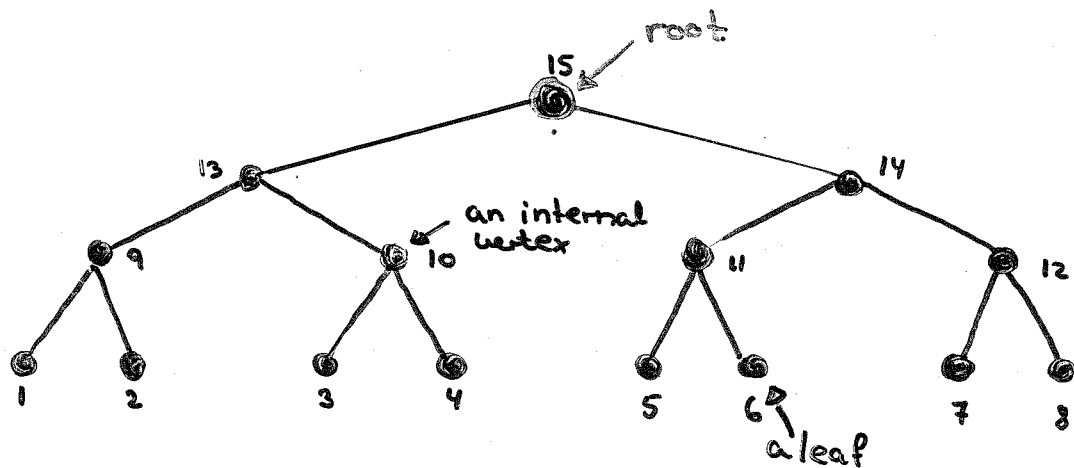
- ① T is a connected graph with no cycles
- ② T is a simple graph with the property that there is a unique path between any pair of vertices v and w .

THEOREM

Let T be a tree on n vertices, then T has $n-1$ edges.

A rooted tree is one in which we have specified one of the vertices to be the root.

EXAMPLE Some terminology for rooted trees:



15 is at level 0

13, 14 are at level 1

9, 10, 11, 12 are at level 2

The height of this tree is 3

11 and 12 are the children of 14

13 is the parent of 10

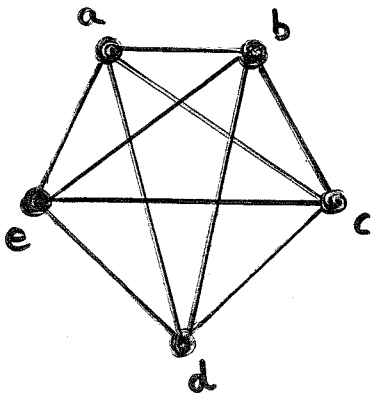
13 is an ancestor of 2

13 is NOT an ancestor of 6

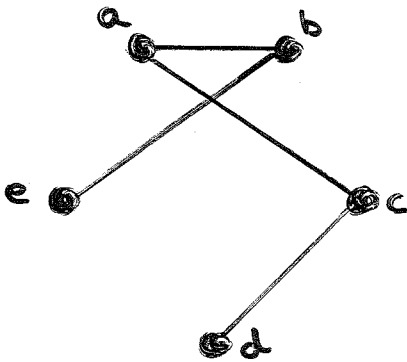
DEFINITION

A ^(uppspännande träd) spanning tree of a graph G is a subgraph of G which
(i) contains all vertices of G and
(ii) is a tree.

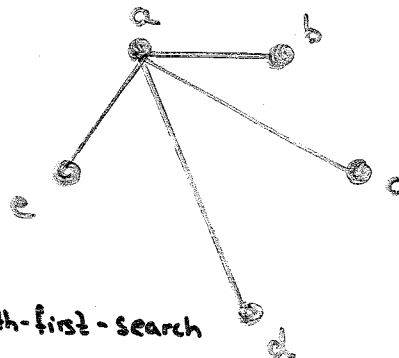
EXAMPLE



A graph G

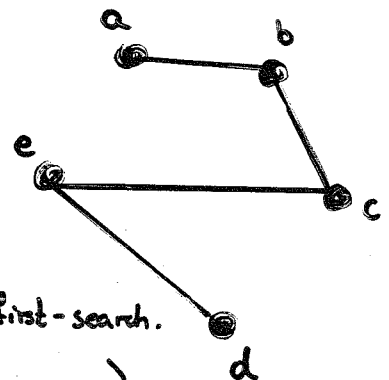


A spanning tree of G



breadth-first-search

A BFS spanning tree of G
rooted at a



depth-first-search.

A DFS Spanning tree
of G rooted at a

ALGORITHM FOR GROWING A DEPTH-FIRST-SEARCH SPANNING TREE T :

Let G be a connected graph on vertices $v_1, v_2, v_3, \dots, v_n$.

INITIAL STEP: Put $x_1 := v_1$ and let T be a tree with
 $V(T) = \{x_1\}$ and $E(T) = \emptyset$.

RECURSIVE STEP: Suppose $V(T) = \{x_1, x_2, \dots, x_i\}$ for some $i \geq 1$.

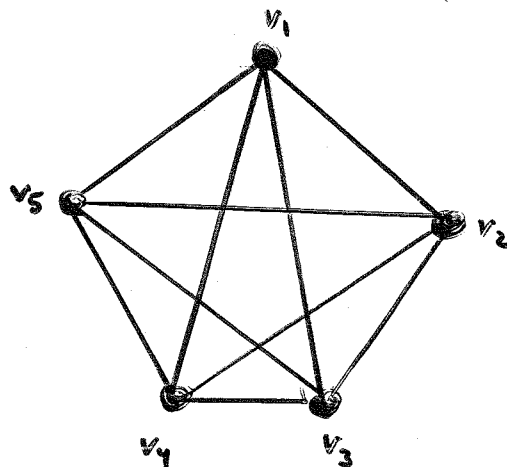
While there are still vertices in $V(G) - V(T)$
adjacent to some $x_j \in V(T)$,

choose a vertex $v \in V(G) - V(T)$ adjacent
to $x_j \in V(T)$ with j as big as possible
and put $x_{i+1} := v$,

add vertex x_{i+1} and edge (x_{i+1}, x_j) to T .

EXAMPLE

We construct a DFS spanning tree in the following graph,
rooted at v_1 :



ALGORITHM FOR GROWING A BREADTH-FIRST-SEARCH SPANNING TREE

Let G be a connected graph on vertices $v_1, v_2, v_3, \dots, v_n$.

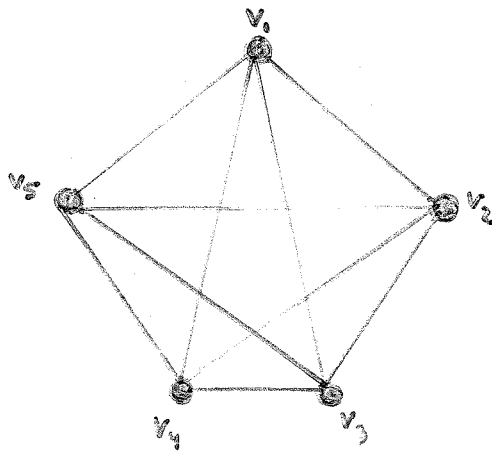
INITIAL STEP: Put $x_1 := v_1$ and let T be a tree with
 $V(T) = \{x_1\}$ and $E(T) = \emptyset$

RECURSIVE STEP: Suppose $V(T) = \{x_1, x_2, \dots, x_i\}$ for some $i \geq 1$.

While there are still vertices in $V(G) - V(T)$
adjacent to some $x_j \in V(T)$,
choose a vertex $v \in V(G) - V(T)$ adjacent
to $x_j \in V(T)$ with j as small as possible
and put $x_{i+1} := v$,
add vertex x_{i+1} and edge (x_j, x_{i+1}) to T .

EXAMPLE

We construct a BFS spanning tree in the following graph,
rooted at v_1



WEIGHTED GRAPHS

Let G be a graph with vertex set V and edge set E .

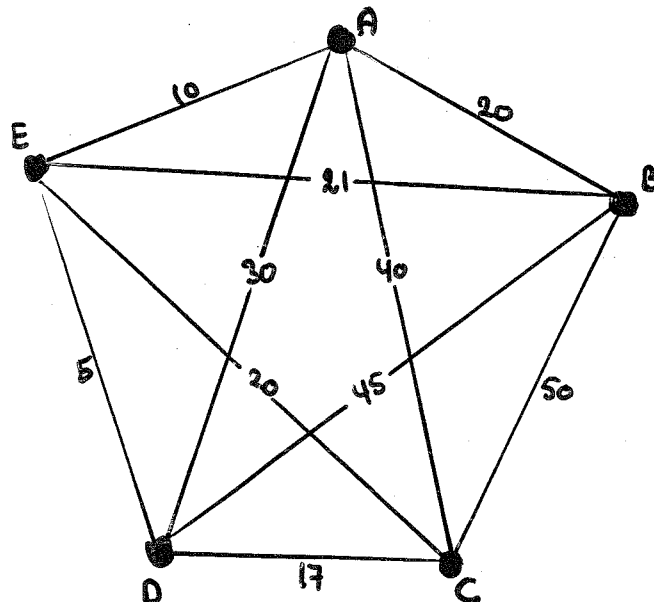
If we have defined a function $w: E \rightarrow \mathbb{R}$, so that $w(e)$ is the weight of the edge e (The weight could be a cost or a distance for example), then G is a weighted graph.

EXAMPLE

The distance between 5 towns A, B, C, D and E in miles is given by the table

	A	B	C	D	E
A	-	20	40	30	10
B	20	-	50	45	21
C	40	50	-	17	20
D	30	45	17	-	5
E	10	21	20	5	-

A weighted graph can be used to model this information:



Q: How do we find the shortest route from A to C ?

Dijkstra's Shortest-Path Algorithm

for finding the shortest path between any two vertices S and T in a connected, weighted graph in which all weights are positive.

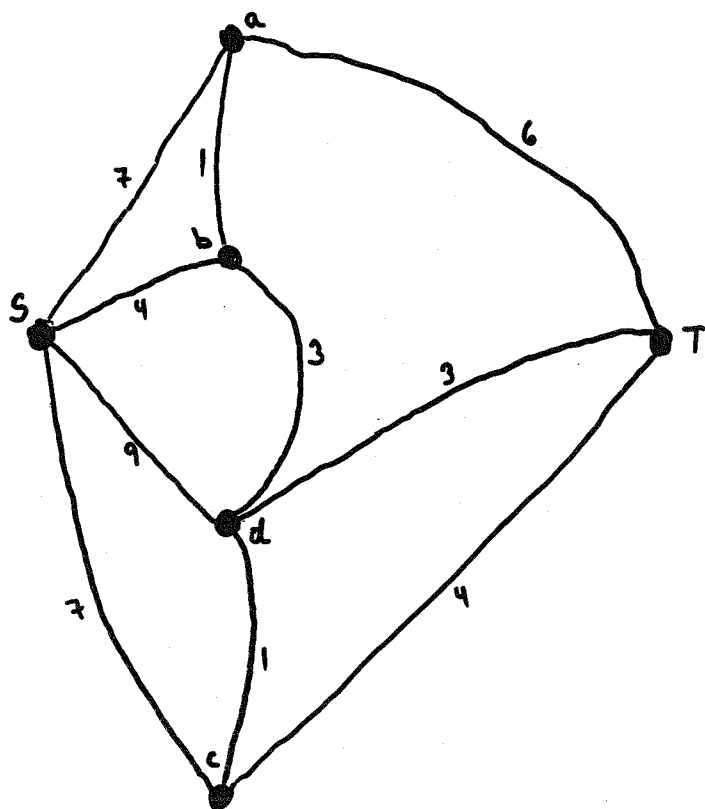
- ① Initialise the graph by assigning the label $(0, S)$ to vertex S and the label (∞, S) to all other vertices. Make sure all vertices are unmarked.

REPEAT steps ②-④

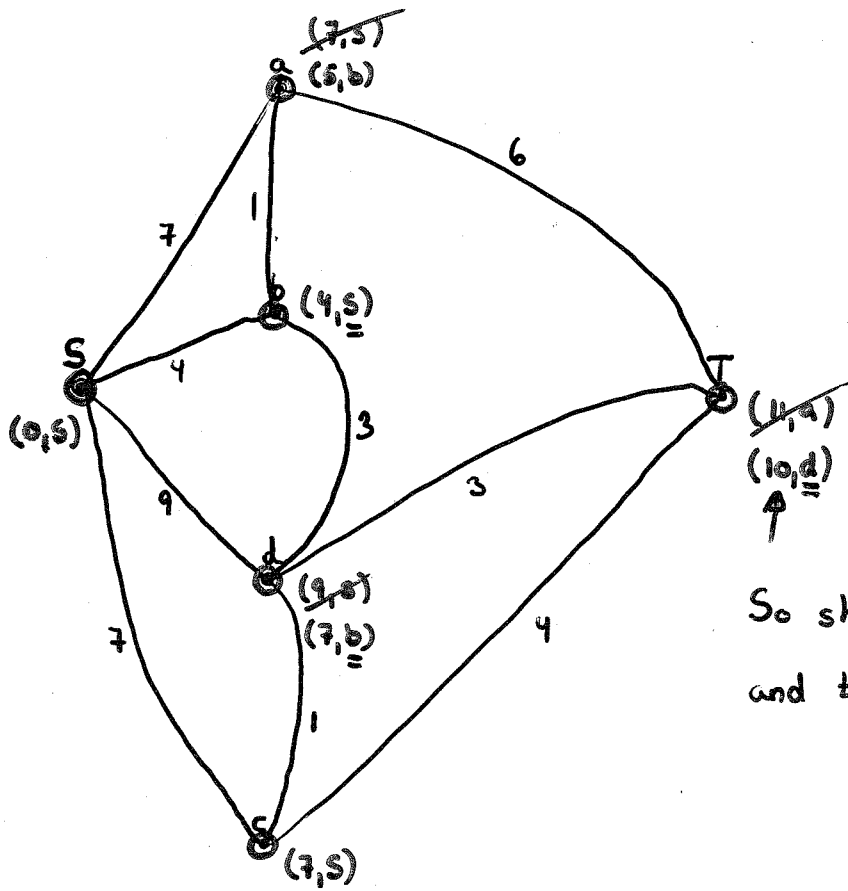
- ② Choose an unmarked vertex v whose label $(d(v), m)$ such that d is as small as possible.
- ③ Mark vertex v (e.g. circle it)
- ④ Update any unmarked neighbour w of v :
If its label is $(d(w), m)$, its new label is
 $(d(v) + \text{weight of edge } (v, w), v)$
if $d(v) + \text{weight of edge } (v, w) < d(w)$
otherwise the label is unchanged

UNTIL vertex T is marked

- ⑤ When T is marked and has label $(d(T), x)$ the shortest path from S to T has length $d(T)$ and the previous vertex in the path was x .



EXAMPLE



So shortest path has length 10

and the path is

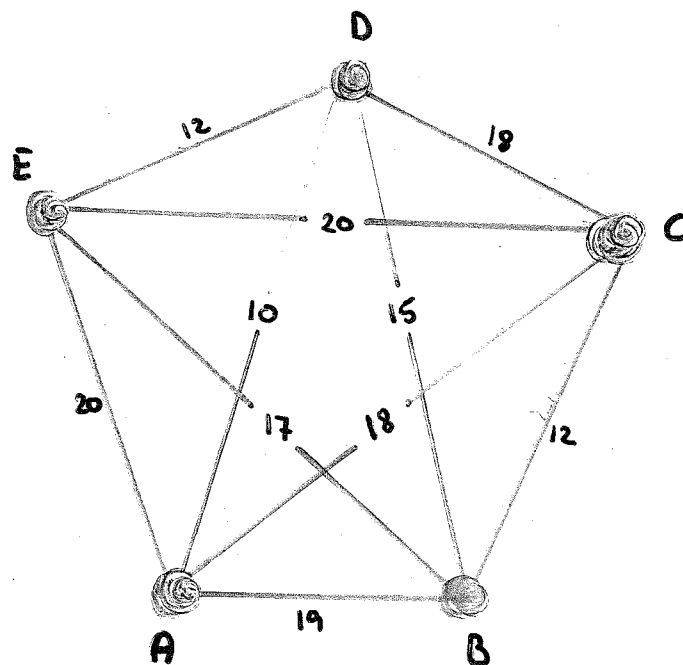
(S, b, d, T)

EXAMPLE

The distances between 5 towns A, B, C, D and E in miles is given as weights on the edges in the following graph.

A telephone company wants to wire all of the towns with as small a length of wire as possible.

They want a minimum weight spanning tree for the graph.



Two algorithms exist for solving this problem:

Kruskal's algorithm and Prim's algorithm

KRUSKAL'S ALGORITHM FOR FINDING AN MST.

INITIAL STEP: Choose an edge e_1 of minimum weight in G and let H_1 be the subgraph of G with $E(H_1) = \{e_1\}$.

RECURSIVE STEP: Suppose we have constructed H_i with $E(H_i) = \{e_1, e_2, \dots, e_i\}$ for some $i \geq 1$.

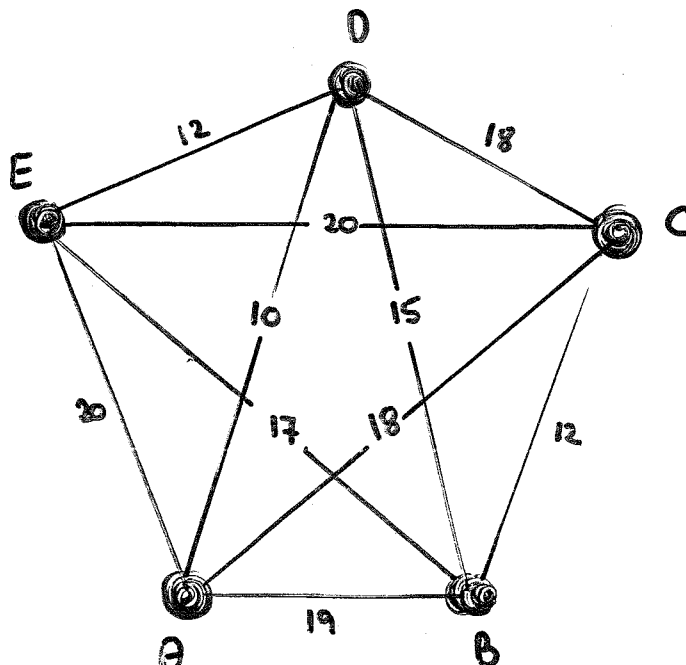
While $i < |V(G)| - 1$ choose an edge e_{i+1} of G such that

- (a) adding e_{i+1} to H_i creates no cycle in H_i
- (b) subject to (a), $w(e_{i+1})$ is as small as possible.

Add e_{i+1} to H_i to form H_{i+1} .

EXAMPLE

Using Kruskal's algorithm, we find an MST in the following graph:



PRIM'S ALGORITHM FOR FINDING AN MST

INITIAL STEP: choose any vertex x_1 of G and let T_1 be the tree with $V(T_1) = \{x_1\}$ and $E(T_1) = \emptyset$

RECURSIVE STEP: Suppose we have constructed T_i with $V(T_i) = \{x_1, x_2, \dots, x_i\}$ for some $i \geq 1$

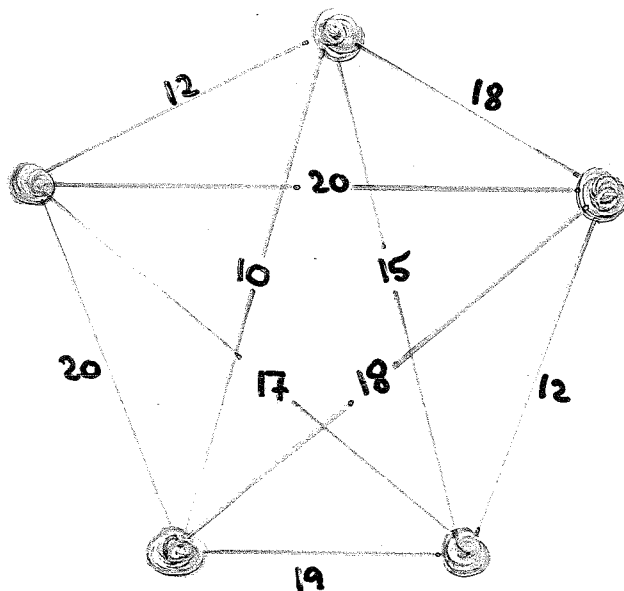
While $i < |V(G)| - 1$ choose an edge (x_j, v) where

- (a) x_j is in T_i and v is not in T_i
- (b) adding (x_j, v) to T_i creates no cycles
- (c) subject to (b), $w(x_j, v)$ is as small as possible.

Add $x_{i+1} = v$ to T_i and the edge (x_j, x_{i+1}) to form T_{i+1} .

EXAMPLE

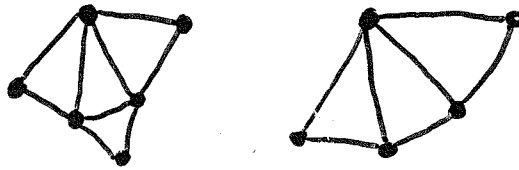
Using Prim's algorithm we find an MST in the following graph:



Hamiltonian and Eulerian graphs

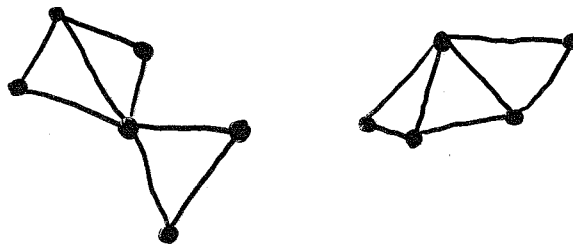
Example

A snow plough needs to clear all the roads in an area. It is preferable that no road is travelled along more than once. Can this be done?



Example

A salesman wants to visit customers in several different towns. He does not want to visit any place more than once. Can this be done?



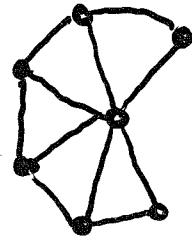
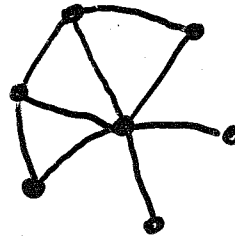
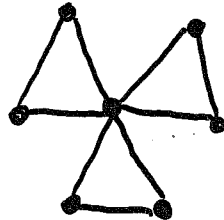
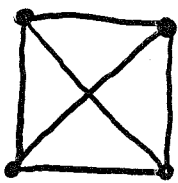
Euler cycle

A cycle which contains every edge of the graph (precisely once) is called an ^{Euler-cycle} Euler cycle.

If G has an Euler cycle G is called a ^{Euler's graph} Eulerian graph.

EXAMPLE

Can the following graphs be drawn without repeating an edge or taking the pen off the paper



THEOREM

A connected graph on $n \geq 3$ vertices has an Euler cycle



it contains no vertices of odd degree.

A graph G is called edge-traceable if it contains a path with no repeated edges, which contains every edge of G .

COROLLARY

A connected graph is edge-traceable



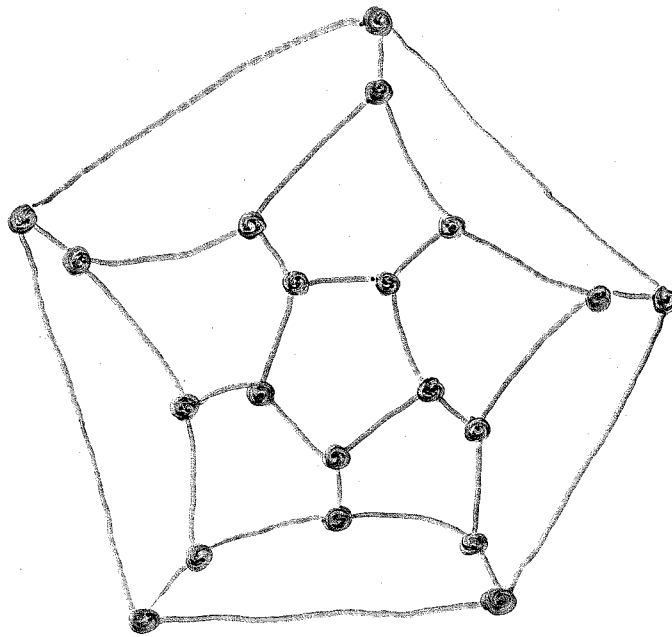
it contains at most two vertices of odd degree.

Hamilton cycle

Hamiltonian cycle

Let G be a graph. If G contains a simple cycle going through every vertex of G , then G is a hamiltonian graph and the cycle is a hamiltonian cycle.

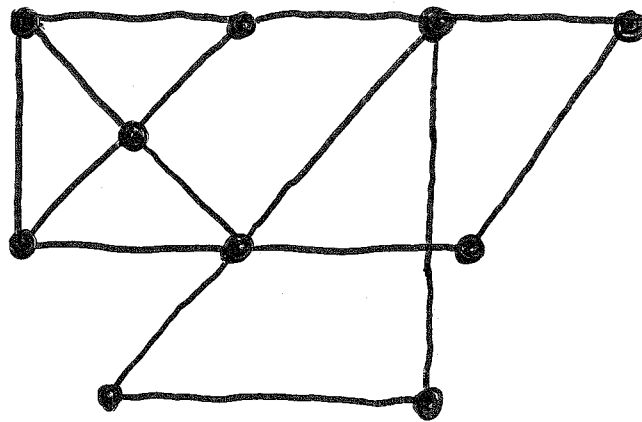
A simple path going through every vertex of G is a hamiltonian path.



Example

Is G hamiltonian?

G :



THEOREM

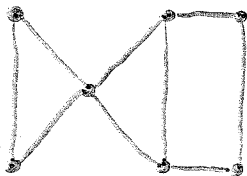
Let G be a hamiltonian graph.

Then for every non-empty proper subset S of the vertex set $V(G)$, there are at most $|S|$ components in $G-S$.

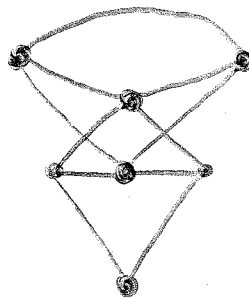
So, remove $|S|$ vertices and you are left with just $|S|$ components.

EXAMPLE

Which are hamiltonian?



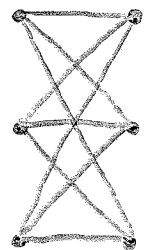
G_1



G_2



$K_{2,3}$



$K_{3,3}$

$K_{n,n}$ is hamiltonian for $n \geq 2$

$K_{n,m}$ where $m \neq n$ is not hamiltonian.

Travelling Salesman Problem

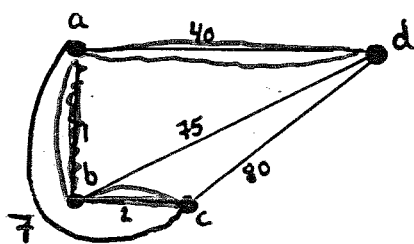
A travelling salesman wants to visit each of his customers, returning to his start point at the end.

He wants to travel the shortest possible distance and visit every customer precisely once.

We model this by a complete graph on n vertices where the n customers live at the n vertices. We weight the graph by assigning the weight d to edge (v_i, v_j) if the distance between customer v_i and customer v_j is d .

NOTES

- ① A visit to every customer is a closed path through all vertices.
- ② We require a closed path through all vertices of smallest weight.
- ③ Since the salesman has as extra condition that he wants to visit every customer precisely once, he is actually demanding a HAMILTONIAN CYCLE of shortest possible weight.
- ④ The salesman is not doing himself any favour in requiring this, for the shortest possible route in the graph for him is not necessarily a Hamiltonian cycle:



shortest route: (a, b, c, b, a, d, a) , length 86

shortest Hamiltonian cycle: (a, b, c, d, a) , length 123

Result

Suppose the graph satisfies the triangle inequality, that is, whenever a, b and c are vertices, then $w(ab) + w(bc) \geq w(ac)$.

Then a shortest route around all customers is a Hamiltonian cycle.

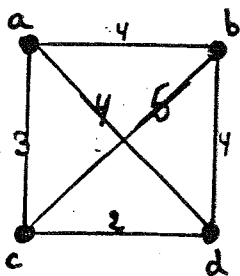
Finding the exact route requires exhaustive search, but we can find a good approximation:

Nearest Neighbour Approximation to a solution of TSP:

Let G be a weighted complete graph satisfying the triangle inequality.

- ① Start at any vertex v_1 . This is P_1 .
- ② Build a path P_{k+1} by adding the nearest neighbour of v_k to P_k .
- ③ Repeat ②, but stop when no vertices are left. If the final path is $P_n = (v_1, v_2, \dots, v_n)$, then the nearest neighbour approximation to a solution is $(v_1, v_2, v_3, \dots, v_n, v_1)$.

EXAMPLE



start at a: $P_1: (a)$ length 0

$P_2: (a, c)$ length 3

$P_3: (a, c, d)$ length 5

$P_4: (a, c, d, b)$ length 9

Approximated solution: (a, c, d, b, a) , length 13

start at c: $P_1: (c)$ $P_2: (c, d)$ $P_3: (c, d, a)$ $P_4: (c, d, a, b)$

Approximated solution (c, d, a, b, c) , length 15

So the solution is not necessarily optimal!