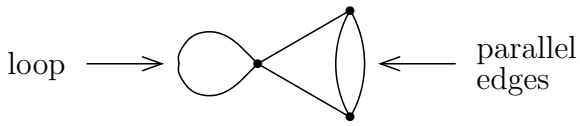
Chapter 1 Graphs and Multigraphs

1.1. Basic definitions. A *pseudograph* has *vertices* (a.k.a. *nodes* or *points*) and *edges* (*lines*). Every edge joins two vertices.



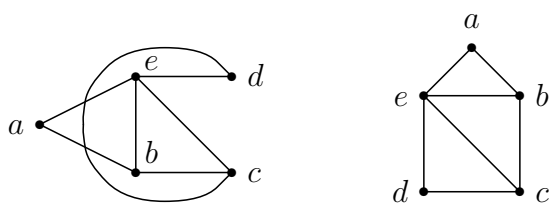
A pseudograph

A *loop* joins a vertex to itself. *Multiple edges* or *parallel edges* are two or more edges joining the same two vertices. A *multigraph* has no loops. A *graph*, or *simple graph* for emphasis, has no loops or parallel edges. Every graph is a multigraph; every multigraph is a pseudograph.

A multigraph (not a graph)

A graph (also a multigraph)

A multigraph G has *vertex-set* $V(G)$ and *edge-set* $E(G)$. We write $G = (V, E)$ if $V(G) = V$ and $E(G) = E$. Two graphs G and H are *isomorphic* (Greek 'same form', essentially the same), written $G \cong H$, if there is a $1:1$ correspondence between $V(G)$ and $V(H)$ such that two vertices of G are *adjacent* (i.e., joined by an edge) iff (if and only if) the corresponding vertices of H are.



$V(G) = \{a, b, c, d, e\}$
$E(G) = \{ab, ae, bc, be, cd, ce, de\}$

Two isomorphic graphs

Edges ab and bc are *adjacent*. Vertex a and edge ab are

incident. The *degree* $\deg(v) = \deg_G(v)$ of a vertex v is the number of edges incident with it. E.g., in the above example, $\deg(a) = \deg(d) = 2$, $\deg(b) = \deg(c) = 3$, and $\deg(e) = 4$.

$|X|$ (or $\#X$) denotes the number of elements in the set X. So $|V(G)|$ is the number of vertices in G, called its *order*, and $|E(G)|$ is its number of edges.

Theorem 1.1. (The *handshaking lemma*.) Let $G = (V, E)$ be a multigraph. Then $\sum_{v \in V} \deg(v) = 2|E|$.

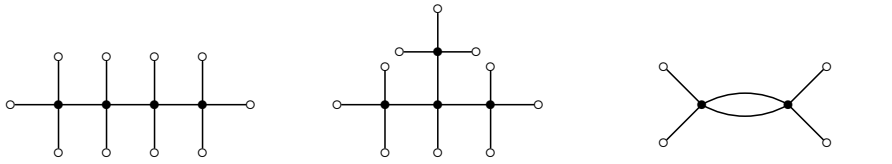
Proof. Count in two ways the number of ends of edges, call it x. The number of ends of edges at v is $\deg(v)$, so $x = \sum_{v \in V} \deg(v)$. But every edge has two ends, so $x = 2|E|$. //

Corollary 1.1.1. Every multigraph has an *even* number of vertices of odd degree. //

Graphs can represent lots of different things:

1. Physical networks: road, rail, river, canal, telephone, gas, electricity, ... (Sometimes each edge has a number indicating its length; if not, assume every edge has length 1.)
2. Vertices = people, edge = 'knows'. ('Six degrees of separation.')
3. Vertices = students, edge = 'willing to share a double room'.
4. Vertices = examinations, edge = 'cannot take place simultaneously'.
5. Chemical molecules: vertices = atoms, edges = chemical bonds.

• = carbon
◦ = hydrogen



Butane, C_4H_{10} Isobutane, C_4H_{10} Ethylene, $CH_2{=}CH_2$

There are also *directed graphs* or *digraphs* (the edges have arrows on them):

1. Vertices = players in an 'all-play-all' tournament, directed edge from u to v if u beats v.
2. Pecking order of hens.
3. Vertices = species, edge $\vec{uv}$ if species u preys on species v.
4. Family trees: vertices = people, edge $\vec{uv}$ if u is a parent of v.
5. Critical path analysis: edges = activities, edges $\vec{uv}$ and $\vec{vw}$ mean that uv must be completed before vw can start.

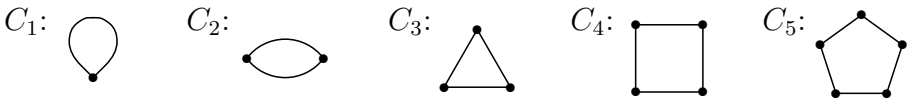
A *mixed graph* can have both directed and undirected edges (e.g., a town plan with some one-way streets).

In this module we will consider only finite undirected graphs and (occasionally) multigraphs.

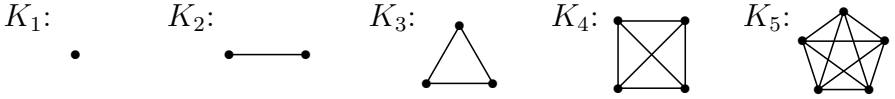
Some special graphs have names.

1. The *path* P_n has n vertices, $n-1$ edges and *length* $n-1$.

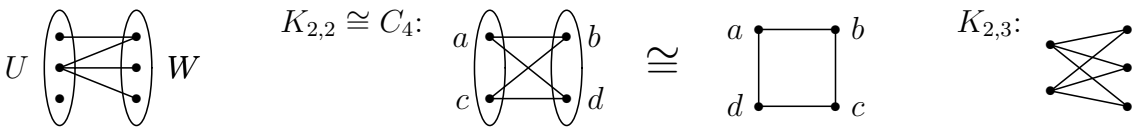
2. The *cycle* (or *circuit*) C_n has n vertices, n edges and *length* n. (C_2 is not a graph; C_1 is not even a multigraph.)



3. The *complete graph* K_n has n vertices, all adjacent.



4. A graph is *bipartite* if its vertices can be partitioned into two disjoint sets U and W, its *partite sets*, where every edge joins vertices in different sets. The *complete bipartite graph* $K_{r,s}$ has two disjoint sets of r and s vertices and all possible edges between them.



5. A graph is *planar* if it can be drawn in the plane without edges crossing. All paths and cycles are planar, as is K_n if $n \leqslant 4$.

(Question: When is $K_{r,s}$ planar?)

1.2. Connectedness. If u and v are vertices of a multigraph G, a *walk* of *length* l from u to v is a sequence

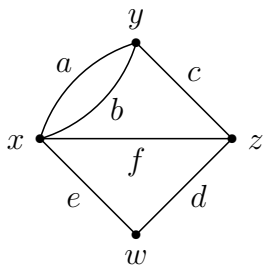
$$u = v_0, e_1, v_1, e_2, v_2, \ldots, e_l, v_l = v$$

of vertices and edges such that each edge e_i joins v_{i-1} and v_i. It is *closed* if $u = v$, and *open* otherwise. It *connects* u and v, which are its *endvertices*; $v_1, \ldots, v_{l-1}$ are its *interior vertices*. In a *graph*, it suffices to list the vertices; the walk could be written as $v_0 v_1 v_2 \ldots v_l$.

A *trail* is a walk where the edges are all different.

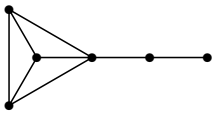
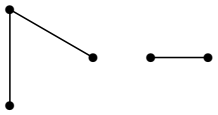
A *path* is a trail where the vertices are all different.

A *cycle* (or *circuit*) is a closed trail where the vertices are all different except that $v_0 = v_l$.



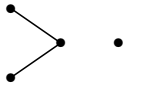
In the multigraph shown,
$x, a, y, b, x, b, y, c, z$ is a walk of length 4 (not a trail — b is repeated).
x, a, y, b, x, f, z is a trail of length 3 (not a path — x is repeated).
x, e, w, d, z, c, y, or just $xwzy$, is a path of length 3.
$x, a, y, c, z, d, w, e, x$ is a cycle of length 4.

G_1 is a *subgraph* (or *submultigraph*) of G_2, written $G_1 \subseteq G_2$, if $V(G_1) \subseteq V(G_2)$ and $E(G_1) \subseteq E(G_2)$, i.e, every vertex of G_1 is a vertex of G_2, and every edge of G_1 is an edge of G_2.

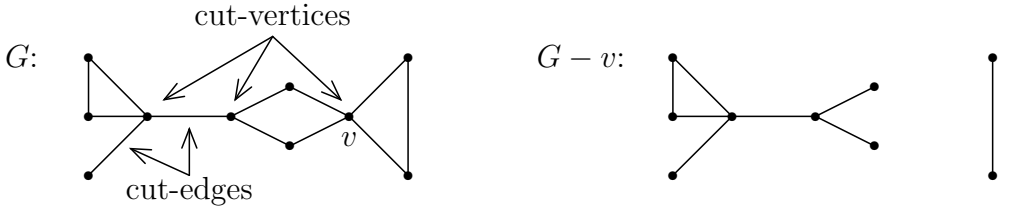
E.g.,  has  as a subgraph. [Must be a multigraph.]

We sometimes think of paths and cycles as subgraphs, i.e., sets of vertices and edges, with no starting point or direction.

If $u \neq v$ and there is a walk from u to v in G, then there is a path from u to v in G. (If any vertex x is repeated, leave out the part of the walk between the first and last occurrences of x; repeat until there are no repeated vertices.)

A graph G is *connected* if, for every two vertices u and v, there is a path from u to v (i.e., G is 'all in one piece'). A *component* of G is a maximal connected subgraph of G (so the components are the 'connected pieces' of G). There is a path from u to v iff u and v are in the same component. A graph is connected iff it has exactly one component. A *trivial component* has only one vertex (an *isolated vertex*). The graph  has three components, one trivial; they are isomorphic to P_3, K_1 and K_2.

A *cut-vertex* or *cut-edge*, $=$ *bridge*, is a vertex or edge, respectively, whose removal disconnects the component in which it lies. (When you remove a vertex, you must remove also all edges incident with it.)



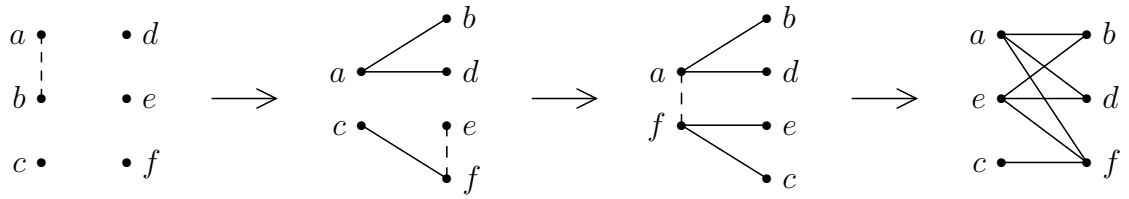
Theorem 1.2. An edge uv is a cut-edge of G
$\iff$ there is no path between u and v in $G - uv$
$\iff$ there is no cycle containing uv in G. //

Theorem 1.3. A multigraph G is bipartite if and only if every cycle in G has even length.

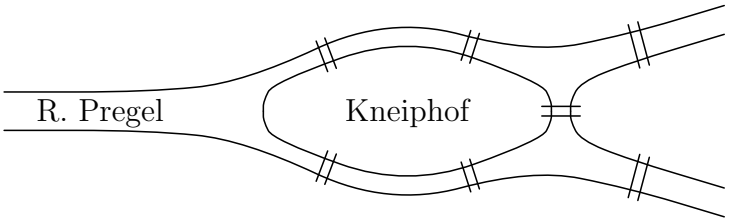
Proof. If G is bipartite, with partite sets U and W, then every walk of odd length starting in U ends in W, and so every cycle must have even length.

Suppose conversely that every cycle has even length. Remove all the edges from G and assign its vertices to U and W arbitrarily. Now restore the edges one at a time, keeping the graph bipartite at each stage. If we want to add an edge uv, and u and v are in *different* sets, we can just add it. So suppose u and v are in the *same* set. Then there is no path connecting them, since such a path P would have even length, and so $P + uv$ would be an odd cycle, $\Rightarrow\Leftarrow$. Thus u and v are in different components C_u and C_v. Swap over all vertices of C_v from U to W or vice versa, and then add uv. Continue until all edges have been added. //

E.g., $V(G) = \{a, b, c, d, e, f\}$, $E(G) = \{ab, ad, cf, ef, af, de, be\}$:

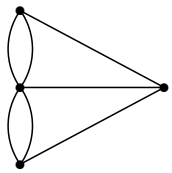


1.3. Eulerian multigraphs. In the 1730s the town of Königsberg in East Prussia (now Kaliningrad in Russia) had seven bridges.



The citizens wondered if it was possible to go for a walk, crossing every bridge exactly once. In 1736, Euler heard of the problem and showed it was impossible, thereby inventing graph theory!

A trail in a multigraph is *eulerian* if it uses every edge (exactly once) and every vertex. The Königsberg bridge problem is equivalent to finding a (closed?) eulerian trail in the following multigraph.



Theorem 1.4. (L. Euler, 1736; C. F. B. Hierholzer, 1873.) A multigraph G has a closed eulerian trail if and only if it is connected and every vertex has even degree.

Proof. The conditions are necessary, since every time you pass through a vertex you use two edges, one in and one out. So suppose the conditions hold.

Start at any vertex u and follow edges, using no edge twice, until you get stuck. When you arrive at any vertex $v \neq u$, you have used an odd number of edges at v (one more in than out), and so there must be an unused edge you can leave by. So the only place you can get stuck is at u. You have now completed a closed trail T_0.

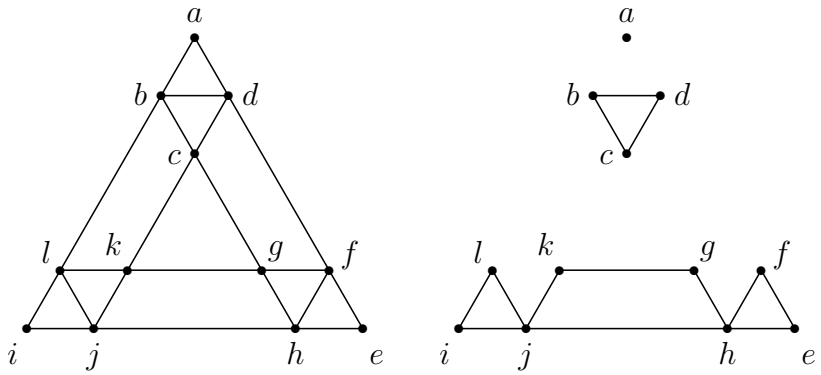
If there are any edges you have not used, then every component of $G - E(T_0)$ has a vertex in common with T_0 (otherwise G

would be disconnected). So pick a vertex u_1 in both T_0 and a non-trivial component C_1 of $G - E(T_0)$. Note that every vertex of C_1 still has even degree. Use the same method as before to construct a closed trail W_1 in C_1 starting from u_1. Then insert W_1 in T_0 to form a longer closed trail T_1 in G. Continue in this way until you reach a closed trail T_k containing all edges of G. //

Example.

$T_0 = adfgcklba,$

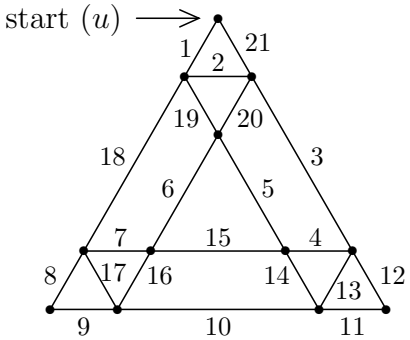
$T_1 = adfgcklb(cdb)a,$



$T_2 = adf(ehf)gcklbcdba, \quad T_3 = adfehfgckl(ijkghjl)bcdba.$

A multigraph is *eulerian* if it has a closed eulerian trail. There is an algorithm to construct a closed eulerian trail in an eulerian multigraph.

Fleury's Algorithm. Start from any vertex and follow edges subject to these rules.
(a) Erase the edges as they are used.
(b) Never use a cut-edge unless you have to.
(Note: you have to know the whole multigraph in advance.)

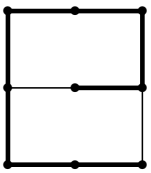


Corollary 1.4.1. A multigraph G has an open eulerian trail from u to v iff G is connected and has exactly two vertices of odd degree, u and v.

Proof. G has an open eulerian trail from u to v iff the graph $G+uv$ has a closed eulerian trail. The result follows from Theorem 1.4. //

[Chinese postman problem, Mei-Ko Kwan, 1962.]

1.4. Hamiltonian graphs. A *hamiltonian path* or *hamiltonian cycle* in a graph G is a path or cycle, respectively, that includes all vertices of G. G is *hamiltonian* if it has a hamiltonian cycle.



[W. R. Hamilton, icosian problem, 1857/59; knight's tour; travelling salesman problem.]

In contrast with eulerian graphs, there is no simple test for whether or not a graph is hamiltonian, and no algorithm (better than trial and error) for constructing a hamiltonian cycle in a hamiltonian graph. But there are several known sufficient conditions for a graph to be hamiltonian.

Theorem 1.5. (G. A. Dirac, 1952.) If G is a graph of order $n \geqslant 3$ and every vertex has degree at least $\frac{1}{2}n$, then G is hamiltonian.

[Dinner-party problem]

Proof. We will assume G is not hamiltonian and get a contradiction. Form H from G by adding extra edges if necessary until the addition of any further edge would make H hamiltonian (but H is not hamiltonian). Then any two nonadjacent vertices of H are connected by a hamiltonian path. Let

$$v_1 v_2 \ldots v_{n-1} v_n$$

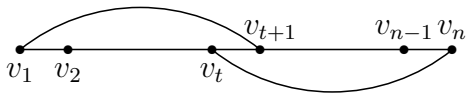
be a hamiltonian path in H. Then

v_n is adjacent to more than half the vertices $v_1, v_2, \ldots, v_{n-1}$,
v_1 is adjacent to more than half the vertices $v_2, v_3, \ldots, v_n$,
i.e., to more than half the vertices v_{t+1} such that $1 \leqslant t \leqslant n-1$.

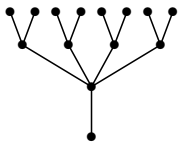
Thus there is a $t \in \{1, 2, \ldots, n-1\}$ such that $v_t v_n$ and $v_1 v_{t+1}$ are both edges of G. Then

$$v_1 v_2 \ldots v_t v_n v_{n-1} \ldots v_{t+1} v_1$$

is a hamiltonian cycle of H. This contradiction shows that G is hamiltonian after all. //

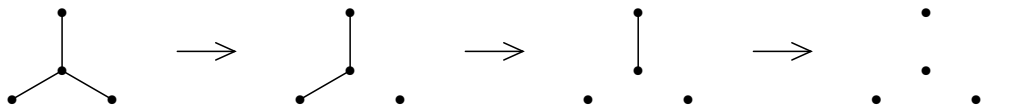


1.5. Trees and edge-weighted graphs. A graph of order n is a *tree* if it satisfies any one of the following equivalent conditions.



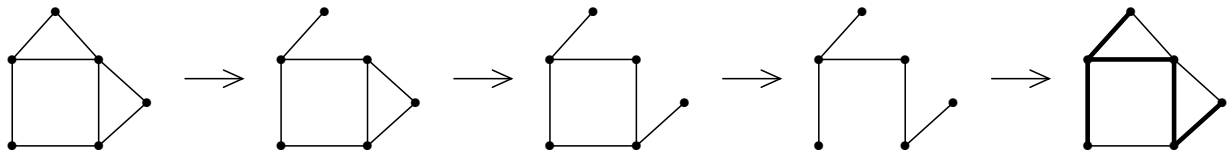
T1. G is connected and has no cycle.
T2. G has $n-1$ edges and no cycle.
T3. G has $n-1$ edges and is connected.
T4. Every two vertices of G are connected by exactly one path.
T5. G is connected but the deletion of any edge disconnects it.
T6. G has no cycle but the addition of any new edge creates one.
T7. The addition of any new edge to G creates exactly one new cycle.

Proof that T1 $\Longrightarrow$ T2 $\Longrightarrow$ T3 $\Longrightarrow$ T1.
(a) T1 $\Longrightarrow$ T2: Since G has no cycle, removing any edge increases the number of components by one. If we start with G (one component) and remove all the edges, we finish with n isolated vertices (n components), and so we must have removed $n-1$ edges.



(b) T2 $\Longrightarrow$ T3: If G is not connected, add edges joining different components to make a connected graph with no cycle and more than $n-1$ edges; this contradicts (a).
(c) T3 $\Longrightarrow$ T1: If G has a cycle, remove edges from cycles to make a connected graph with no cycle and fewer than $n-1$ edges; this contradicts (a). //

A *forest* is a graph with no cycles; so a graph is a forest iff every component of it is a tree. A *spanning subgraph* of a graph G is a subgraph that includes all vertices of G. A *spanning tree* of a connected graph G is a spanning subgraph of G that is a tree. A *maximal forest* or *skeleton* of a (possibly disconnected) graph H consists of a spanning tree in each component of H; it is a maximal subgraph containing no cycles. To form a spanning tree of G one can build it up or build it down: either start with all the vertices and no edges, and add as many edges of G as possible without creating a cycle, or start with G and delete as many edges as possible without disconnecting it.



Suppose now that G is a connected graph and every edge e has a number $w(e) \geqslant 0$ called its *weight*, *length* or *cost*. We will consider two problems.

1.5.1. The minimum connector problem. We wish to find a connected spanning subgraph of G with minimum possible weight (or length, or cost). Clearly it will be a spanning tree.

Kruskal's Algorithm. (J. B. Kruskal, 1956.) Start with a forest T containing all vertices of G and no edges. Consider the edges of G in order of increasing weight, and add each edge to T iff this would

not create a cycle. Stop when T is connected; it is then a minimum-weight spanning tree.

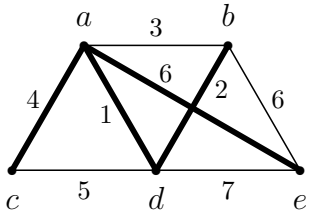
[Greedy algorithm, idea doesn't always work.]

Proof. Let T' be a minimum-weight spanning tree that has as many edges in common with T as possible; we will prove that $T' = T$. Suppose $T' \neq T$. Let $e_1, e_2, \ldots, e_{n-1}$ be the edges of T in the order in which they were added to T. Let e_k be the *first* edge in the list that is not in T'. Adding e_k to T' creates a cycle, which must contain an edge e'_k not in T. Let $T'' := T' + e_k - e'_k$. Then T'' is connected, with $n-1$ edges, and so it is a spanning tree of G.

$$\begin{aligned} T: &\ e_1, e_2, \ldots, e_{k-1}, e_k, \ldots \\ T': &\ e_1, e_2, \ldots, e_{k-1}, e'_k, \ldots \end{aligned}$$

The edges $e_1, \ldots, e_{k-1}, e'_k$ are all in T', and so they do not contain a cycle. Thus $w(e_k) \leqslant w(e'_k)$, since otherwise we would have chosen e'_k instead of e_k when forming T. So $w(T'') \leqslant w(T')$, which means that $w(T'') = w(T')$ and T'' is another minimum-weight spanning tree. But T'' has one more edge in common with T than T' has. This contradicts the choice of T' and so proves the result. //

Example.



The edges are considered in the order ad, bd, [ab,] ac, [cd,] ae or be, where the brackets denote edges that are *not* included in the tree. There are two different minimum-weight spanning trees, depending on whether we include ae or be.

1.5.2. The shortest path problem. We wish to find a shortest path between two vertices u and w. The following algorithm finds shortest paths from u to all other vertices; they form a tree, called a

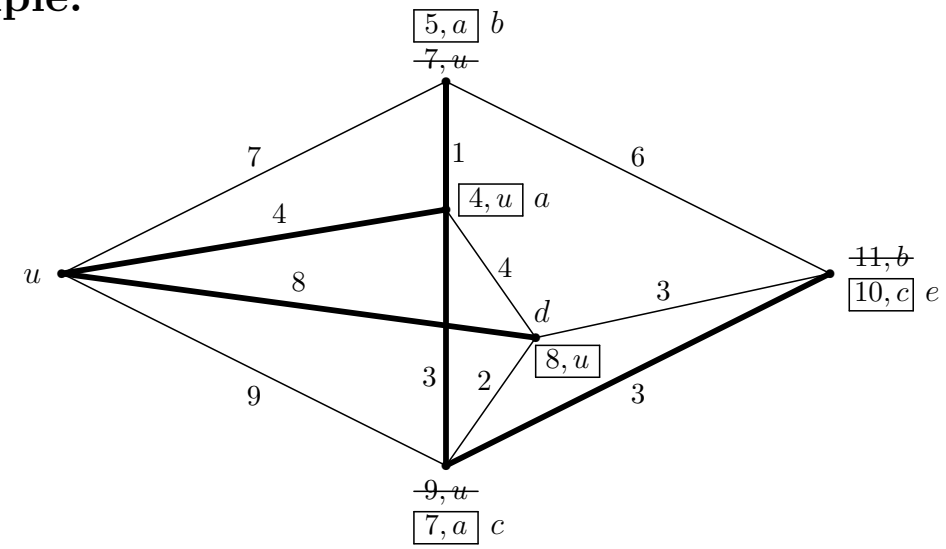
shortest-distance tree rooted at u. For each vertex x, we maintain two 'labels', $L(x)$ and $P(x)$. $L(x)$ is a *number*, the length of the shortest path found so far from u to x, and $P(x)$ is a *vertex*, the vertex preceding x on that path (unless $x = u$). When we are sure that we have found the shortest path to x, we add x and the edge $P(x)x$ into the tree T, and then 'process' x.

Dijkstra's Algorithm. (E. W. Dijkstra, 1959.)

1. (Initialization.) Set $L(u) := 0$, $L(x) := \infty$ for every other vertex x, $T := \{u\}$ (a 1-vertex tree) and $v := u$ (the first vertex to be processed).
2. 'Process v': for each edge vx of G with $x \notin T$, if $L(v)+w(vx) < L(x)$ (i.e, the path from u to v in T, followed by the edge vx, is shorter than the shortest path we previously knew to x), then reset $L(x) := L(v) + w(vx)$ and $P(x) := v$.
3. Choose $x \notin T$ such that $L(x)$ is smallest, and add x and the edge $P(x)x$ to T. (Any other path from u to x must be at least as long as the one we have currently found.)
4. If $V(T) \neq V(G)$, set $v := x$ and return to Step 2. Otherwise stop: for each vertex x, $L(x)$ is the length of a shortest path from u to x, and T is a shortest-distance tree rooted at u.

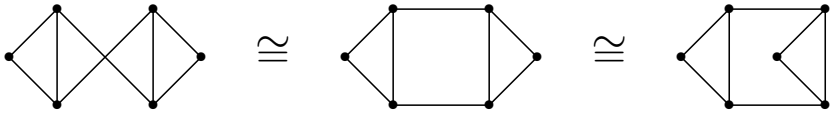
If you only want the shortest distance to a particular vertex w, then stop as soon as $w \in T$.

Example.



[Note: In exercises and exam questions, you must label the graph *exactly* like this.]

1.6. Planar graphs. A *plane graph* is a graph that is *already* drawn in the plane without edges crossing. A *planar graph* is one that *can* be drawn as (i.e., is isomorphic to) a plane graph. A plane graph divides the plane into regions or *faces*, and we write $G = (V, E, F)$ where F is the set of faces (including the outside face).



Theorem 1.6. (Euler's Polyhedron Formula, 1752.) Let $G = (V, E, F)$ be a plane graph with c components. Then

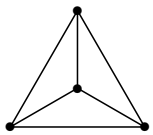
$$|V| - |E| + |F| = c + 1.$$

Proof. If G has a cycle, remove an edge e that lies in a cycle; this reduces both $|E|$ and $|F|$ by one (since it merges the faces either side of e), and so it does not change the value of either $|V| - |E| + |F|$ or $c + 1$. Repeat until G becomes a forest with c components. Then every component is a tree, with one fewer edge than vertices, and so $|E| = |V| - c$. Also $|F| = 1$, and so

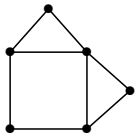
$$|V| - |E| + |F| = |V| - (|V| - c) + 1 = c + 1. \quad //$$

Corollary 1.6.1. If $G = (V, E, F)$ is a plane graph then $|V| - |E| + |F| \geqslant 2$. If G is connected then $|V| - |E| + |F| = 2$. //

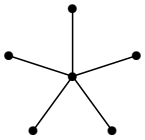
Examples.



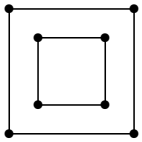
$4 - 6 + 4 = 2$



$6 - 8 + 4 = 2$



$6 - 5 + 1 = 2$



$8 - 8 + 3 = 3$
(not connected!)

Theorem 1.7. A planar graph of order $n \geqslant 3$ has at most $3n - 6$ edges.

Proof. This is true if $|E| \leqslant 1$, so suppose $|E| \geqslant 2$. Then every face has at least three 'sides of edges' in its boundary. Since every edge has two sides, the number of sides of edges is $2|E|$, and it is at least $3|F|$; thus $2|E| \geqslant 3|F|$. Now Euler's formula, multiplied by 3, gives $0 \leqslant 3n - 3|E| + 3|F| - 6$, and so

$$|E| \leqslant 3n - 2|E| + 3|F| - 6 \leqslant 3n - 6. \quad //$$

Theorem 1.8. A planar bipartite graph of order $n \geqslant 3$ has at most $2n - 4$ edges.

Proof. Since G is bipartite, no face can have just three edges in its boundary, and so every face has at least four edges; thus $2|E| \geqslant 4|F|$, or $|E| \geqslant 2|F|$. Now Euler's formula, multiplied by 2, gives $0 \leqslant 2n - 2|E| + 2|F| - 4$, and so

$$|E| \leqslant 2n - |E| + 2|F| - 4 \leqslant 2n - 4. \quad //$$

K_5 has 5 vertices and 10 edges, and so if K_5 is planar then

$$|E| = 10 \leqslant 3n - 6 = 3 \cdot 5 - 6 = 9$$

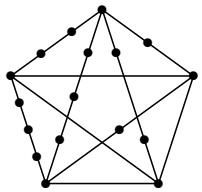
by Theorem 1.7, $\Rightarrow\Leftarrow$. So K_5 is not planar.

$K_{3,3}$ is bipartite, and has 6 vertices and 9 edges, and so if $K_{3,3}$ is planar then

$$|E| = 9 \leqslant 2n - 4 = 2 \cdot 6 - 4 = 8$$

by Theorem 1.8, $\Rightarrow\Leftarrow$. So $K_{3,3}$ is not planar.

A *subdivision* of a graph G is formed by adding vertices of degree 2 subdividing the edges of G.



A subdivision of K_5

Theorem 1.9. (K. Kuratowski, 1930.) A graph is planar if and only if it has no subgraph isomorphic to K_5, $K_{3,3}$, or any subdivision of them. //