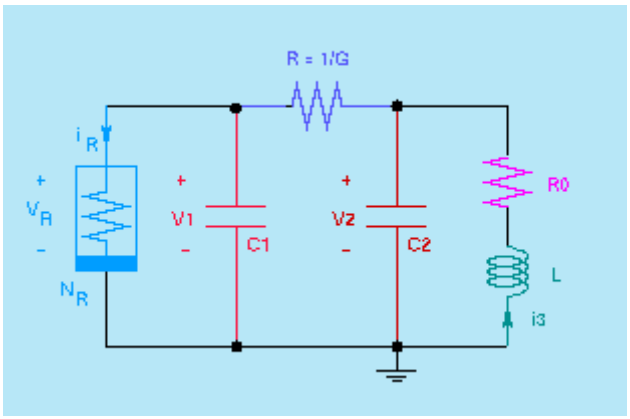
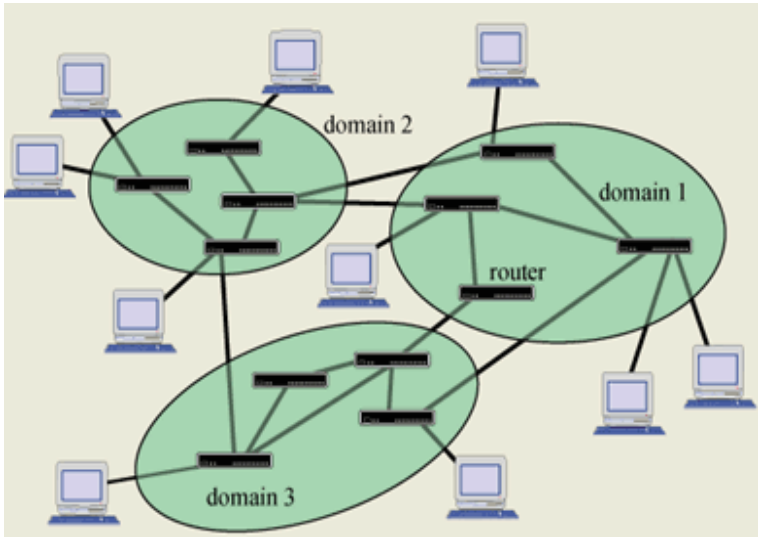
2 A Brief Introduction to Graph Theory

2.1 What is a graph?

A ***graph*** is a diagrammatical representation of some physical structure such as a circuit (Fig. 2-1 (a)), or a computer network (Fig. 2-1 (b)), even a certain type of human relationship (Fig. 2-1 (c)), and so on.



(a)



(b)

(c)

Fig. 2-1 Some examples of graph

As mentioned in Section 1.2.1, Chapter 1, the idea of representing a physical problem by a graph and then solving it by mathematical analysis and computation may be traced back to the eighteenth century when the great mathematician Leonhard Eüler (1707-1783) studied and solved the famous seven-bridge problem in a town named Königsburg, which is now located inside the territory of Russia.

As shown in Fig. 2-2 [1], there are two small islands in the river Pregel passing trough the town Königsburg, and there are seven bridges over the river. The then residents were always amazed as if one could walk through all the seven bridges once and once only and finally return to the starting point.

In 1736, Eüler had a new idea to describe this real-world problem with an abstract graph, using points *A, B, C, D* to represent the four pieces of lands separated by the river in town and then using curves *a, b, c, d, e, f, g* to represent the seven bridges that connect the four points *A, B, C, D* (Fig. 2-3 [1]). Thus, Eüler was able to convert the physical problem to the following mathematical problem: In the graph shown in Fig. 2-3, starting from any point, is there any possible loop leading back to the starting point such that it passes all the seven lines once and once only?

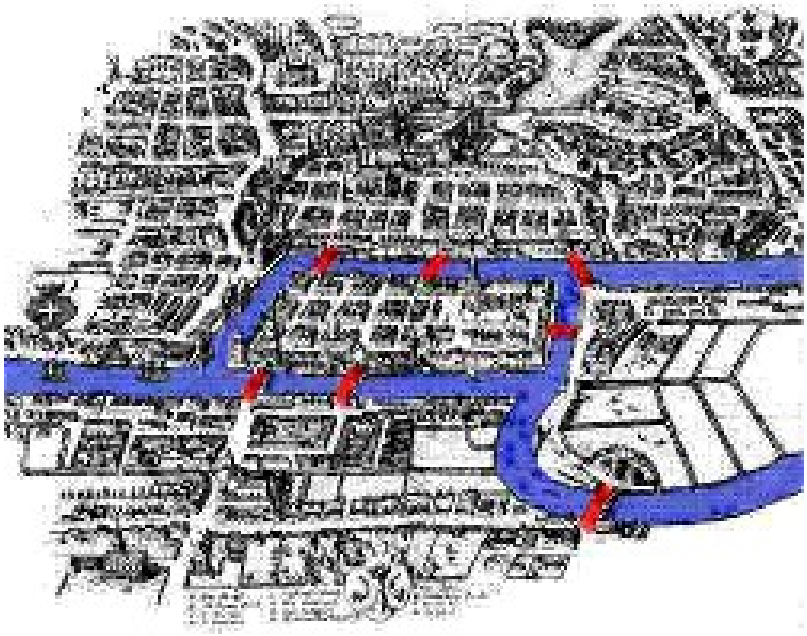


Fig. 2-2 The town Königsburg and the seven bridges in 1736 [1]

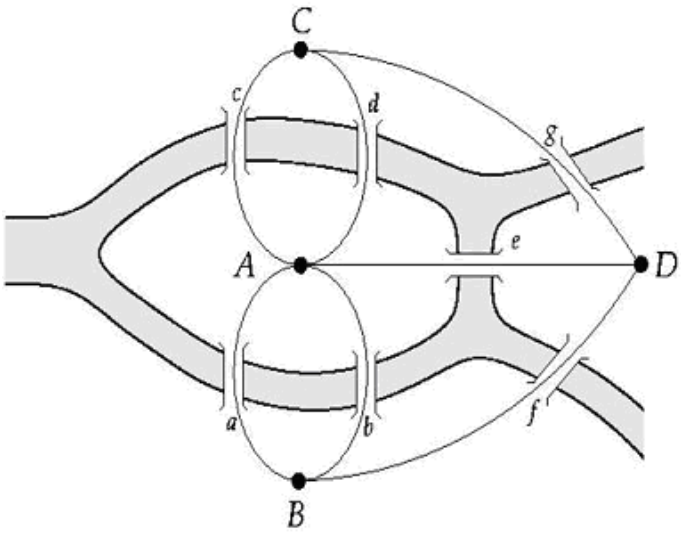


Fig. 2-3 Graph of the Königsburg seven-bridge problem [1**]**

2.2 Notation and Definitions

Let G be a non-empty graph with at least one ***node*** (or, *vertex*). In a non-isolated case, G has at least one ***edge*** (or, *link*), where the edge is not allowed to has an open end without connecting to any node, therefore every edge has two end nodes. Let $N = N(G)$ and $M = M(E)$ denote the sets of its nodes and edges, respectively. Only a finite setting is considered, where $N(G)$ is a non-empty finite set of distinct nodes and $M(E)$ is a

finite set of unordered pairs of distinct nodes in $N(G)$. Such a non-trivial pair $(N(G), M(E))$ is referred to as a ***graph***. In the trivial case where $M(E)$ is an empty set, the graph has only isolated nodes; it is a totally disconnected graph called a ***null graph***.

Figure 2-4 is an example of a non-trivial graph G with four nodes $N(G) = \{A, B, C, D\}$ and four edges $M(E) = \{AB, AC, BC, CD\}$, in which AB denotes an edge joining node A and node B, and so on. In Fig. 2-4, the triangle ABC is a ***subgraph*** of the whole graph, and so are the lines AB and CD, etc. Here, a closed connection like the triangle ABC is called a ***loop*** in the graph. A connected subgraph in a graph is also called a (***connected***) ***component*** of the graph. Figure 2-4 has only one component, while Fig. 2-8 (a) and Fig. 2-12 (b), shown below, each has two.

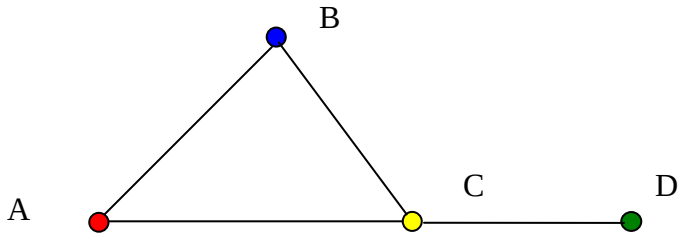


Fig. 2-4 An example of a non-trivial graph

If two nodes are allowed to have more than one edge joining them, or a node can have a ***self-loop***, i.e., an edge joining the same node at both ends, as shown in Fig. 2-5, then the graph is called a ***general graph***, which is considered no longer ***simple***. In Fig. 2-5, the triangle ABC and the lines AB and CD are also subgraphs (and connected components) of the whole graph. In this chapter, hence this whole text, only simple graphs are considered.

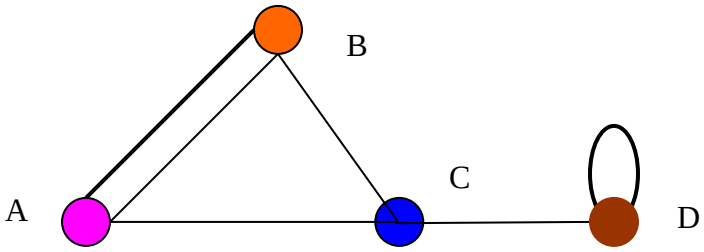


Fig. 2-5 An example of a general graph

In a graph, if some edges are ordered, i.e., are directed (or, have directions), then the graph is called a ***digraph***, as shown by Fig. 2-6 (a). Digraphs are classified into simple and general digraphs, in a similar manner, so the digraph shown in Fig. 2-6 (a) is a ***simple digraph*** while the one shown in Fig. 2-6 (b) is a ***general digraph***. In this chapter, hence this whole text, only non-directed graphs are considered.

Throughout the rest of the text, a "graph" always refers to a simple (without self-loop and repeated edges) and undirected graph, unless otherwise indicated.

(a) (b)

Fig. 2-6 Examples of simple and general digraphs

The ***degree*** of a node is defined to be the number of edges that the node has, which oftentimes is referred to as ***node degree*** in order to distinguish it from other types of degrees. Let $k(v)$ denote the degree of node v in a graph. Thus, in Fig. 2-3, $k(A)=5$ and $k(B)=3$, and so on. About the node degrees, a simple result is the following:

Theorem 2-1 (Handshaking Lemma) *The total node degree of a graph is always an even number.*

Proof. Since every edge joins two nodes, this total node degree is twice the number of edges in any graph. □

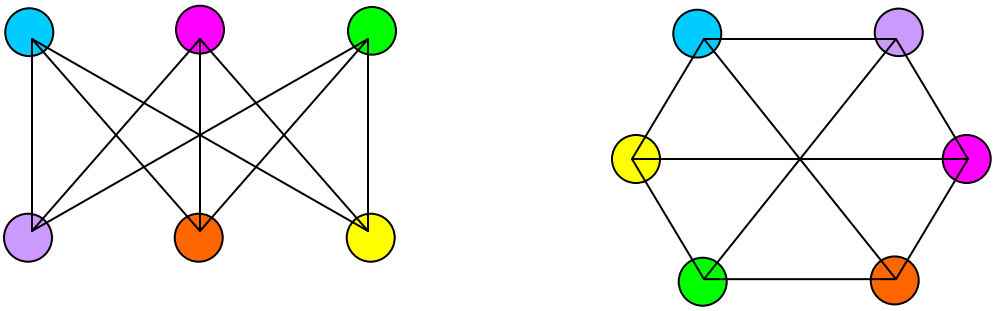
Corollary 2-2 *In any graph, the number of nodes with odd degrees must be even.*

Next, some more notation and concepts are introduced.

Two graphs G_1 and G_2 are said to be ***isomorphic***, if there is a one-one correspondence between the nodes of G_1 and those of G_2 with the property that the number of edges joining any two nodes of G_1 is equal to the number of edges joining the two corresponding nodes of G_2. For example, the two graphs shown in Figs. 2-7 (a) and (b) are isomorphic.

Note that in this and all other pictures throughout, as a convention in drawing, two edges are not crossed if there is no node at the "crossing point." Thus, there are no crossings in between the top and bottom groups of nodes in Fig. 2-7 (a), and no crossings at the center of the graph shown in Fig. 2-7 (b).

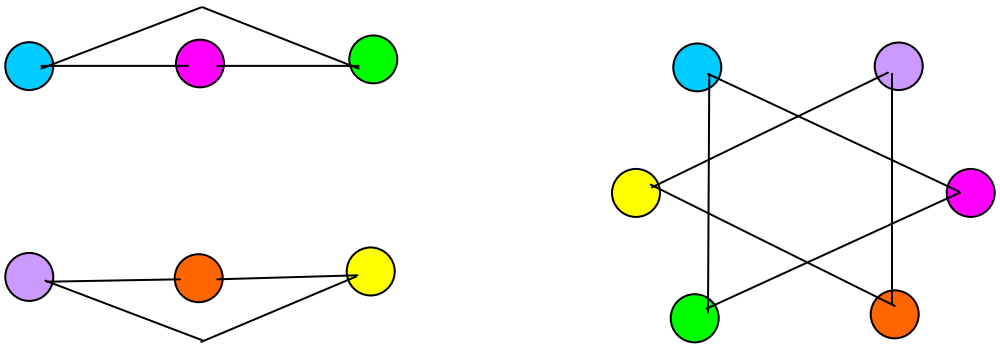
The graph G shown in Fig. 2-7 (a) is quite special, in which all the nodes can be split into two disjoint sets, V_1 and V_2, in such a way that every edge of G joins a node in V_1 to a node in V_2. Another typical example is a star-shaped graph, where the central node belongs to one set and all the other nodes belong to another set. This kind of graph is called a ***bipartite graph***.



(a) (b)

Fig. 2-7 An example of two isomorphic graphs

For a given graph G, its ***complementary graph*** G^c is the graph containing all the nodes of G and all the edges that are not in G. For example, the complementary graphs of the two graphs in Fig. 2-7 (a) and (b) are shown in Fig. 2-8 (a) and (b), respectively. Clearly, the complementary graph of a connected graph may not be connected.



(a) (b)

Fig. 2-8 Two complementary graphs

For a graph G with n nodes, labeled as $N(G)=\{1,2,...,n\}$, its ***adjacency matrix*** A is defined to be the $n\times n$ constant matrix those ij th entry is the number of edges joining node i and node j. An adjacency matrix is always symmetrical. If the edges of the graph are labeled as $M(E)=\{1,2,...,m\}$, then the ***incidence matrix*** M of the graph is defined to be the $n\times m$ constant matrix whose ij th entry is 1 if node i connects edge j; and is 0 otherwise. An incidence matrix usually is non-square, unless the number of nodes is equal to the number of edges. Thus, for the graph shown in Fig. 2-9, one has

$$A=\begin{bmatrix}0&1&0&1\\1&0&1&1\\0&1&0&1\\1&1&1&0\end{bmatrix} \quad \text{and} \quad M=\begin{bmatrix}1&0&0&1&0\\1&1&0&0&1\\0&1&1&0&0\\0&0&1&1&1\end{bmatrix}.$$

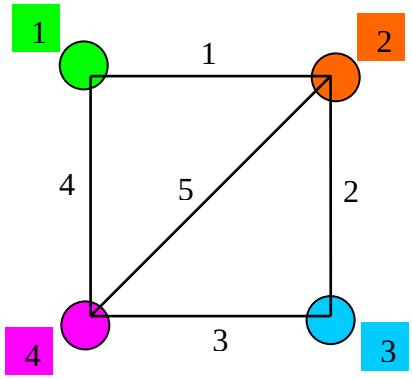


Fig. 2-9 An example for the adjacency and incidence matrices

For a graph, another important concept is the ***Laplacian matrix*** (or, called ***admittance matrix*** or ***Kirchhoff matrix***), denoted $L=\lfloor L_{ij}\rfloor$, which is defined as follows:

$$L_{ij}=\begin{cases}k(v_i) & if \quad i=j\\ -1 & if \quad i\neq j,\quad v_i \quad adjacent \quad v_j\\ 0 & otherwise\end{cases}$$

where $k(v_i)$ is the degree of node v_i, $i,j=1,...,n$. Thus, for the graph shown in Fig. 2-9, the corresponding Laplacian matrix is

$$L=\begin{bmatrix}2&-1&0&-1\\-1&3&-1&-1\\0&-1&2&-1\\-1&-1&-1&3\end{bmatrix}=\begin{bmatrix}2&0&0&0\\0&3&0&0\\0&0&2&0\\0&0&0&3\end{bmatrix}-A$$

where the diagonal matrix has all node degrees on the diagonal and A is the adjacency matrix of the graph given above.

Laplacian matrix L has many useful properties, such as the following:

1) L is always semi-positive definite;
2) L has eigenvalues $\lambda_0\le\lambda_1\le\cdots\le\lambda_n$, with $\lambda_0=\cdots=\lambda_q=0$ where q is the number of disjoint subgraphs in the graph, therefore if the graph is connected, i.e., $q=1$, then $0=\lambda_0<\lambda_1\le\cdots\le\lambda_n$.

In the above, λ_1 is called the ***algebraic connectivity*** and the smallest non-zero eigenvalue of L is called the ***spectral gap*** of the corresponding graph.

A graph in which every node has the same degree, i.e., the same number of connecting edges, is called a ***regular graph***; if every node has degree r then the graph is called a regular ***graph of degree*** r. For example, the graphs shown in Fig. 2-7 are regular graphs of degree 3.

Theorem 2.3 *A regular graph of degree* r *with* N *nodes has* $\frac{1}{2}r\,N$ *edges.*

Proof. Since every node connects with r edges, there are $r\,N$ connecting edges. However, each edge has been doubly counted, so it should be divided by 2. □

For example, the two graphs in Fig. 2-7 are regular graphs of degree 3, with 6 nodes, which both have $\frac{1}{2}\times 3\times 6 = 9$ edges.

Of particular interest are the cubic (or trivalent) graphs, which are regular graphs of degree 3, such as those shown in Fig. 2-7 and Fig. 2-10.

In graph theory, a fully-connected graph is also called a ***complete graph***. A complete graph of N nodes is denoted by K_N. The two graphs shown in Fig. 2-10 are ***complete regular graphs***, both are isomorphic, namely both are K_4, but those shown in Fig. 2-7 are not complete.

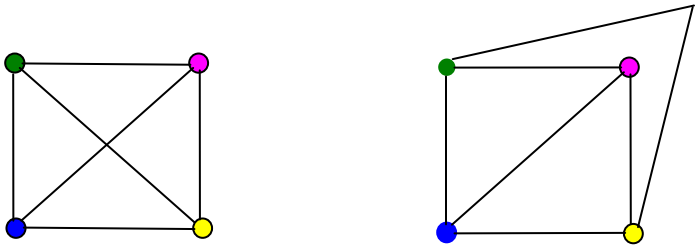
(a) (b)

Fig. 2-10 Examples of complete regular graphs of degree 3

Note that, as mentioned above, in a graph like the one shown in Fig. 2-10 (b), although two edges are visually crossed but they are not, as a convention in all graphic diagrams throughout. Nevertheless, to avoid possible confusion it is much more convenient to show a graph on the plane or in the three-dimensional space, called the ***Euclidean 3-space***, in such a way that no visual crossings will exist. This can be done, for example, by redrawing some edges, as illustrated by Fig. 2-11. If a graph can be redrawn in such a way and then shown in the Euclidean 3-space without visual crossings, then it is said that

the graph is being "embedded" into the Euclidean 3-space (including the plane, which as a special case is an Euclidean 2-space).

Before formally define an embedding of a graph, recall the concept of a ***Jordan curve*** on the plane, which is a continuous curve with no self-crossings on the plane. Thus, all curves shown in Fig. 2-11 (b) are Jordan curves on the plane, but the two in the middle of Fig. 2-11 (a) are not.



(a) (b)

Fig. 2-11 Illustration of avoiding visual crossings in a graph

Now, it is said that a graph G can be ***embedded*** in the Euclidean 3-space if it is isomorphic to a graph drawn in the space with points representing nodes of G and Jordan curves representing its edges (i.e., there are no crossings in the resulting graph diagram in the Euclidean 3-space).

Theorem 2-4 *Every graph can be embedded in the Euclidean 3-space.*

Proof. A constructive proof is given here. First, place all the nodes of the graph at distinct points of the x-axis of the Euclidean 3-space. Then, for each edge, choose a plane passing though the x-axis in such a way that distinct edges of the graph correspond to distinct planes. This is always possible, since there are only finitely many edges. Now, the desired embedding is obtained as follows: For each loop of the graph, on the corresponding plane, draw a small circle passing through the relevant node; for each edge joining two distinct nodes, on the corresponding plane, draw a semi-circle connecting these two relevant nodes. Clearly, all these curves will not intersect since they lie on different planes. The resulting diagram is the embedded graph. □

Of particular interest are the ***planar graphs***, which will be further discussed in Section 2.7 below. All planar graphs can be embedded on planes (within the Euclidean 3-space).

Theorem 2-5 A *graph is planar if and only if it can be embedded on the surface of a sphere.*

Proof. Let G be a graph embedded on the surface of a sphere. Place the sphere to sit on a horizontal plane in such a way that the "south pole" is the only point of contact. Then,

connect the "north pole" to each node of the graph, which is on the surface of the sphere, and then extend the connection line outward until it intersects the plane on the base. Mark the intersection point as a new node on the plane, which is merely the stereographic projection of the original node on the plane. The desired planar representation of the original graph is then obtained by joining each pair of the new nodes with a new edge if there is an edge between the corresponding pair of nodes on the sphere. The converse can be similarly proved. □

Let $N(G) = \{v_1, v_2, ..., v_n\}$ be the set of the nodes of in a graph G. A finite sequence of edges in the form of $v_1v_2, v_2v_3, ..., v_{n-1}v_n$, if they exists, is called a ***walk*** in G. Such a walk is denoted by $v_1 \to v_2 \to \cdots \to v_n$ and the number of edges in the walk is called the ***length*** of the walk. A walk in which all edges are distinct is called a ***trail***. A trail is called a ***path*** if all its nodes are distinct, except perhaps $v_1 = v_n$ which is called a ***closed path***. A closed path is often called a ***circuit*** (or, sometimes, a ***cycle***) in this text, which contains at least two edges because no self-loop is allowed. A graph with no circuits is called a ***tree***. Of more interest are the circuits with an even number of edges.

Theorem 2-6 *In any bipartite graph, every circuit has an even number of path length.*

Proof. Let $v_1 \to v_2 \to \cdots \to v_n \to v_1$ be a circuit in the bipartite graph $G = G(V_1, V_2)$. Assume, without loss of generality, that $v_1 \in V_1$. Then, since G is bipartite, one must have $v_2 \in V_2, v_3 \in V_1$, and so on. Finally, one must have $v_n \in V_2$ for a circuit, yielding an even number of path length. □

Theorem 2-7 *If a simple graph G with N nodes has K components, then the number of edges, M, of G satisfies*

$$N - K \le M \le \frac{1}{2}(N-K)\ (N-K+1).$$

In particular, for a connected graph it reduces to

$$N - 1 \le M \le \frac{1}{2}N(N-1).$$

Proof. To show that $M \ge N - K$, an induction on the number of edges of G may be used, as follows: If G is a null graph, the result holds trivially. Suppose that G has the smallest possible number of edges, say M_0. Then, removing any edge of G will increase the number of components by one, and the remaining graph will have N nodes, $K+1$ components, and $M_0 - 1$ edges. It then follows from the induction hypothesis that $M_0 - 1 \ge N - (K+1)$, implying that $M_0 \ge N - K$, as expected.

To show that $M \le \frac{1}{2}(N-K)\ (N-K+1)$, consider the case where M is the largest possible, and assume that each component of G is a fully-connected subgraph. Pick any

two components, C_i and C_j, having N_i and N_j nodes, respectively, with $N_i \geq N_j > 1$. If these two components are replaced by two new ones having $N_i + 1$ and $N_j - 1$ nodes respectively, then the total number of nodes remains the same, but the number of edges is increased by

$$\frac{1}{2}\{N_i(N_i+1) - N_i(N_i-1)\} - \frac{1}{2}\left\{ N_j(N_j-1) - (N_j-1)(N_j-2) \right\} = N_i - N_j + 1,$$

which is a positive number. Thus, in order to attain the maximum number of edges, G must consist of a fully-connected component with $N - K + 1$ nodes and $K - 1$ isolated nodes. This yields the formula.

A connected graph has only one component, $K = 1$, so the result follows immediately. □

Corollary 2-12 *Any simple graph with N nodes and more than $M = \frac{1}{2}(N-1)(N-2)$ edges must be connected.*

Finally in this section, the important concept of ***graph connectivity*** is discussed. This is to try to answer such a question as "how many edges or nodes must be removed in order to disconnect an originally connected graph?" Obviously, if a node is removed, then all edges joining it will also be removed; but the converse may not be true.

For a given graph G, if after a set of edges, $M_0(E)$, is removed, G becomes unconnected, then $M_0(E)$ is called a ***disconnecting set*** of edges of the graph G. The smallest disconnecting set, i.e., no proper subset of which is a disconnecting set, is called a ***cut-set***. For example, in Fig. 2-12, $M_0^1(E) = \{e_1, e_2\}$, $M_0^2(E) = \{e_1, e_2, e_5\}$ and $M_0^3(E) = \{e_3, e_6, e_7, e_8\}$ are disconnecting sets, in which both $M_0^1(E)$ and $M_0^3(E)$ are cut-sets. A cut-set with only one edge is called a ***bridge***, such as the cut-set $\{e\}$ shown in Fig. 2-13.

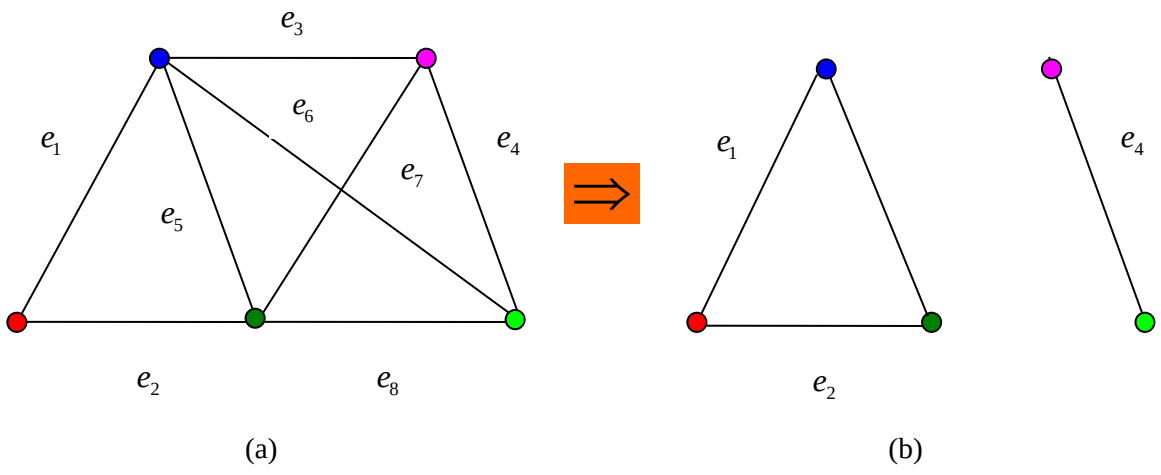


Fig. 2-12 An example of disconnecting set and cut-set

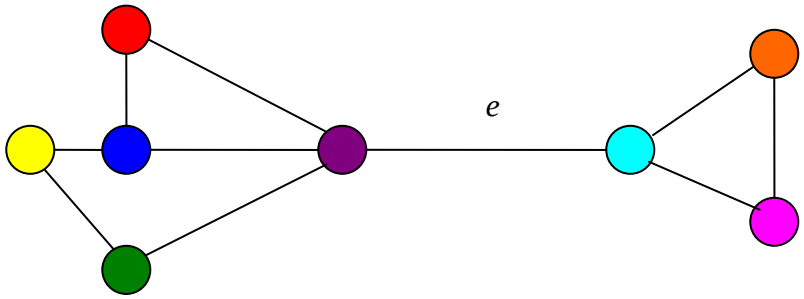


Fig. 2-13 An example of bridge

2.3 Eulerian and Hamiltonian graphs

2.3.1 Eulerian graphs

A connected graph is called an ***Eulerian graph*** if it has a closed trail (i.e., circuit, cycle) that traverses each edge once and once only. There may be more than one such trail, but any of which is called an ***Eulerian trail***. A graph is called a ***semi-Eulerian graph*** if it has a trail, need not be closed, that traverses each edge once and once only. Figures 2-14 (a), (b), and (c) show an Eulerian, a semi-Eulerian and a non-Eulerian graph, respectively.

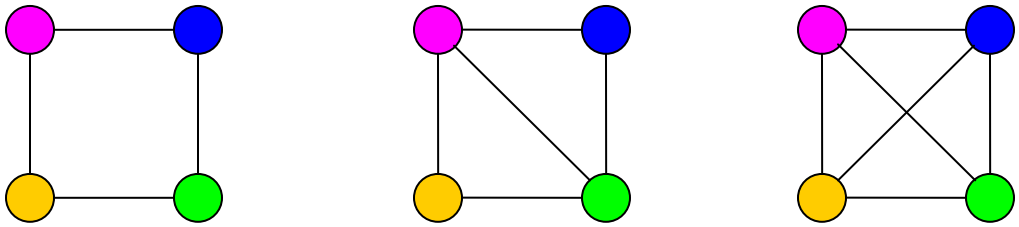


Fig. 2-14 Examples of Eulerian, semi-Eulerian and non-Eulerian graphs

Lemma 2-13 *If every node in a graph has degree larger than or equal to two, then this graph contains a circuit.*

Proof. Starting from any node v_0 of the graph, construct a walk $v_0 \to v_1 \to v_2 \to \cdots$ in such a way that v_1 is any adjacent node of v_0 and, for $i = 1,2,...,$ v_{i+1} is any (except v_{i-1}) adjacent node of v_i. Since every node has degree larger than or equal to two, such a node v_{i+1} exists. Since the graph has finitely many nodes, the walk eventually connects to a node that has been chosen before. This walk yields a circuit in the graph. □

Theorem 2-14 *A connected graph is Eulerian if and only if the degree of every node of the graph is an even number.*

Proof. Suppose that the connected graph G is Eulerian, containing a trail T. Since every edge connecting to a node in T must leave the node, the edge contributes a number two to the degree of the node. Since every edge in T is to be traversed once and once only, the degree of every node must be an integer multiple of two.

Conversely, suppose that the degree of every node in G is an even number. Since G is a connected graph, every node has degree at least two (i.e., even but non-zero), so it follows from Lemma 2-13 that G contains a circuit C. If C contains all edges of G, the proof is complete. If not, remove the edges of C from G, and denote the resulting (probably disconnected) graph by R. Since C does not contain all edges of G, only part of edges of G have been removed, so R is not totally disconnected without any edge. Therefore, although R may contain some isolated nodes, it must contain at least one non-trivial component of G. It is clear that each component of R has at least one node in common with C and, more importantly, every node in R still has an even degree so that Lemma 2-13 implies that R contains a circuit. Thus, one can create a walk as follows: start from a common node of C and R, follow the edges of C until a non-isolated node in R is reached, trace an Eulerian trail in the component of R which contains that node, then continue along the edges of C until a node belonging to another component of R is reached, and so on, and finally stop when the walk returns to the initial node. This walk is an Eulerian trail, therefore G is an Eulerian graph. □

Corollary 2-15 *The Königsburg seven-bridge problem shown in Fig. 2-3 has no solutions.*

Proof. It follows from Theorem 2-14 that the graph of the Königsburg seven-bridge problem shown in Fig. 2-3 is not Eulerian, therefore it is impossible to traverse through each of its edges once and once only and finally return to the starting point. (See Problem 2-20 for a more rigorous argument.) □

Corollary 2-16 (Fleury Algorithm) *In any given Eulerian graph G, an Eulerian trail can be found by the following procedure:*

Start from any node in G. Walk along the edges of G in an arbitrary manner, subject to the following rules:

1) *erase the edges as they are traversed;*
2) *erase the resulting isolated nodes;*
3) *walk through a bridge only if there are no other alternatives.*

Proof. Basic idea can be gained from the proof of Theorem 2-14. A rigorous proof is tedious therefore omitted here. □

The Fleury algorithm is illustrated by the simple example shown in Fig. 2-15.

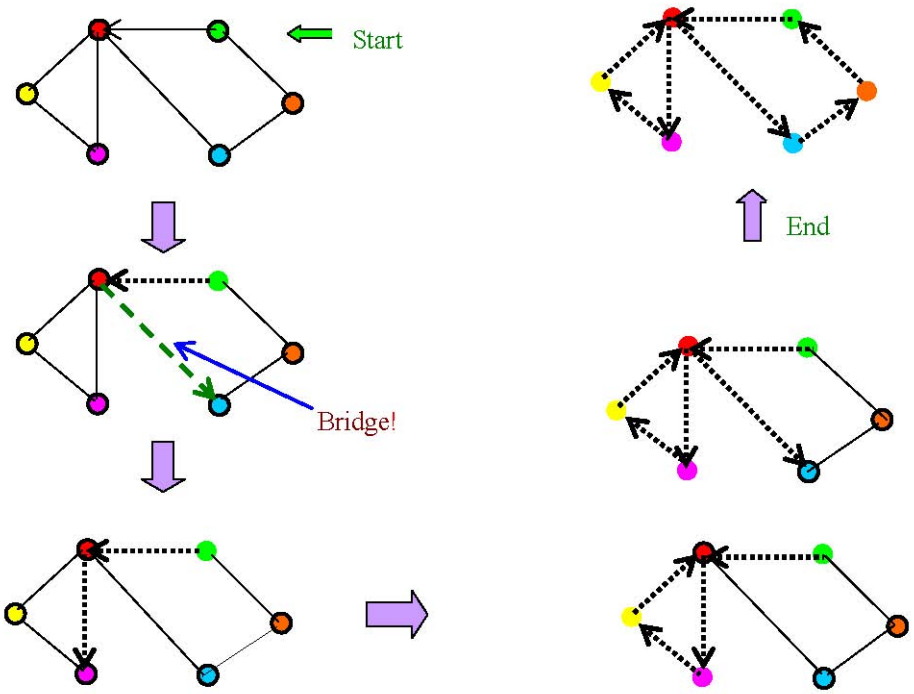
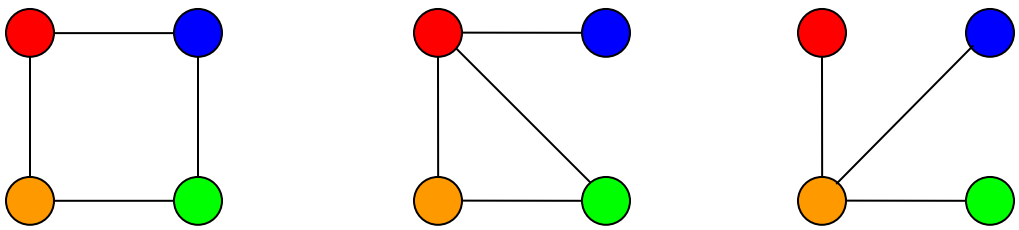


Fig. 2-15 Illustration of the Fleury algorithm

2.3.2 Hamiltonian graphs

A graph is called a ***Hamiltonian graph*** if it has a closed trail (i.e., circuit, or cycle) that traverses each node once and once only. It is clear that the trivial case is a single node and a non-trivial Hamiltonian graph must be a circuit, called a ***Hamiltonian circuit***. A graph is called ***semi-Hamiltonian graph*** if it has a trail, need not be closed, that traverses each node once and once only. Figures 2-16 (a), (b), (c) show a Hamiltonian, a semi-Hamiltonian and a non-Hamiltonian graph, respectively.



(a) (b) (c)

Fig. 2-16 Examples of Hamiltonian, semi-Hamiltonian and non-Hamiltonian graphs

Let $k(v)$ denote the degree of node v in a graph.

Theorem 2-17 *Let G be a simple graph with N (≥ 3) nodes. If, for every pair of non-adjacent nodes v and u, it is always true that $k(v)+k(u)\geq N$, then G is a Hamiltonian graph.*

Proof. Suppose the assertion is not true. Then, by adding extra edges if necessary, which does not violate the inequality condition, one may assume that G is "only just" non-Hamiltonian in the sense that the addition of any more edge will result in a Hamiltonian graph. Let $v_1 \to v_2 \to \cdots \to v_n$ be a walk that traverses all nodes of G. Since G is non-Hamiltonian, the nodes v_1 and v_n will not be adjacent, and so by the assumption of the theorem one has $k(v_1)+k(v_n)\geq N$. Since $N\geq 3$, one must have $k(v_1)\geq 2$ or $k(v_n)\geq 2$. Suppose $k(v_1)\geq 2$. Then, other than v_2 there must be a node, denoted v_i, adjacent to v_1. However, if v_i is not adjacent to v_n then one has $k(v_i)+k(v_n)\geq N\geq 3$, so there must be another node, say v_j, adjacent to v_i. Since the graph has finitely many nodes, this process will stop at some node which appeared previously. This closed loop yields a Hamiltonian circuit. □

2.4 The Chinese postman problem

The problem of interest here is for a postman to deliver all letters in such a way that he passes every street at least once and finally returns to the starting point (the post office), traversing a shortest possible total path-length.

Since it requires the postman to traverse through every street, this is an Eulerian-type of graph but it differs a perfect Eulerian graph in that it allows multiple passing of some streets (otherwise the postman may not be able to return to the post office) and requires the total traveling path-length be minimal. This problem was solved in 1962 by a Chinese mathematician Professor Mei-ko Kwan (1934—) from Shanghai, China, thereby gaining the name as a "Chinese postman problem".

Clearly, if all nodes in such a graph have even degrees, then the graph is Eulerian therefore the problem is trivially solved by the Fleury algorithm. It is also clear that if there are odd-degree nodes in the graph, the postman has to traverse through such nodes for one more time, which makes them of even degrees by means of adding one more edge to each of them. In so doing, however, some originally even-degree nodes would become of odd degrees instead, so that one more edge has to be added to each of the latter to make them even, until all nodes have even degrees. The question is how to do this, so that the total traveling path-length is minimal?

Corollary 2-17 (Kwan's Algorithm) *In any given connected graph G, a Chinese postman trail can be found by the following off-line procedure beforehand:*

Identify the initial node in G, and then do the following:

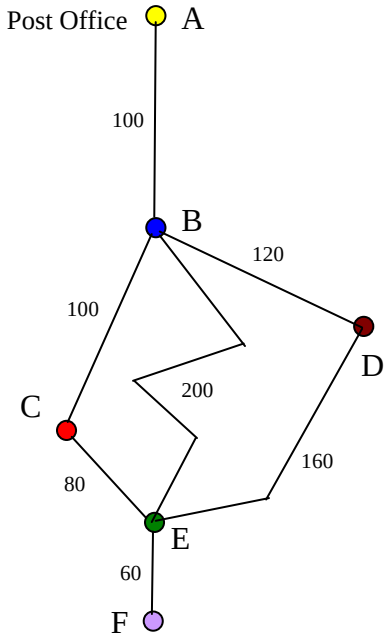
4) *for each odd-degree node in G , add an edge to connect it to one of its neighboring node;*
5) *repeat the above, until all nodes in G have even degrees;*
6) *check all resulting reconfigurations - if the increment of the total path-length of a loop is not longer than one half of the total path-length of the corresponding original loop, then keep this loop as a solution.*

The Kwan algorithm is illustrated by the following example of postman route inspection problem.

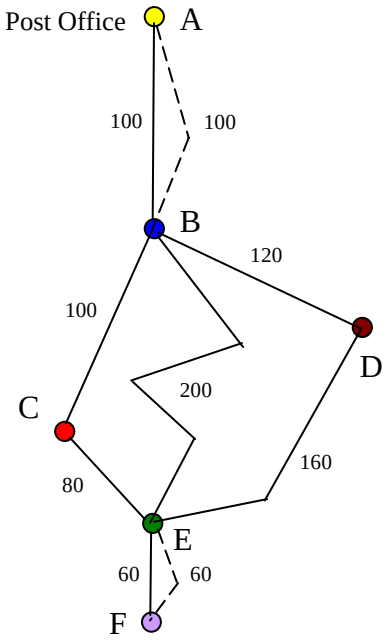
Consider a postman route map shown in Fig. 2-17 (a), where node A is the post office.

First of all, in order to be able to return to node A, one edge has to be added to it from node B, and for the same reason another one is added between node E and node F, as shown in Fig. 2-17 (b).

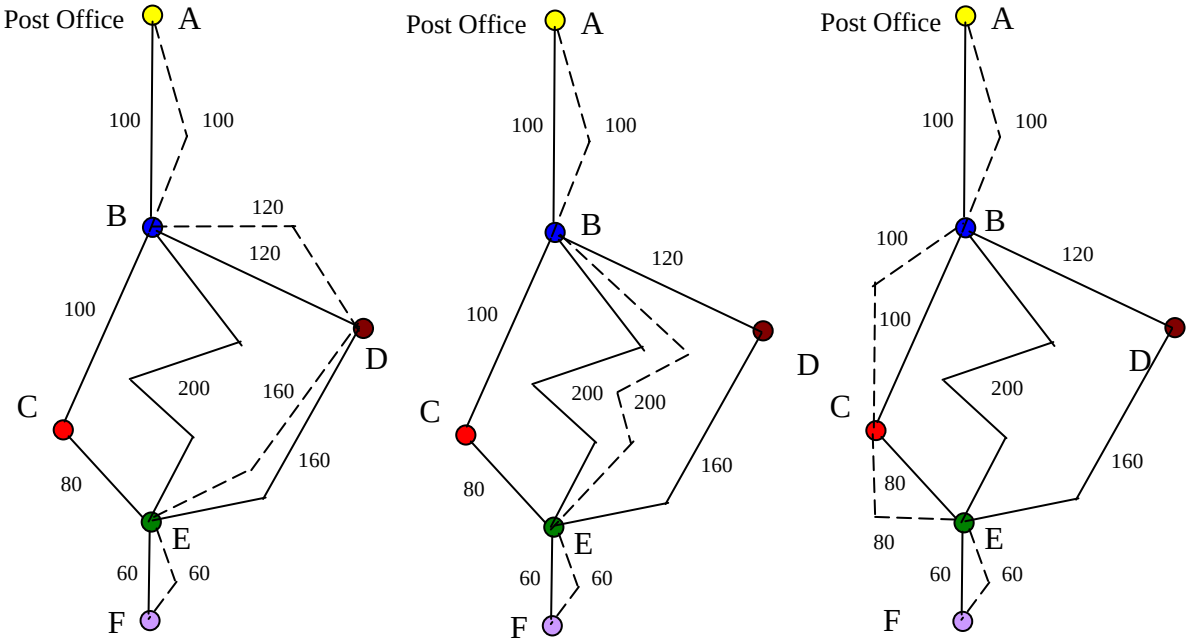
After the above adding, however, the degrees of nodes B and E are changed from even to odd. Consequently, one more edge has to be added to node B and node E, respectively. There are three possible ways to do so, as shown in Fig. 2-17 (c), (d) and (e), respectively.



(a) The routing map



(b) Making nodes A and F be of even degree



(c) Possible route 1 (d) Possible route 2 (e) Possible route 3

Fig. 2-17 A postman route inspection problem

A quick inspection on Fig. 2-17 reveals the following:

(i) There are three existing loops:

(i.1) Total path-length of loop B-C-E-D-B = 120 + 160 + 80 + 100 = 460

(i.2) Total path-length of loop B-C-E-B = 100 + 80 + 200 = 380

(i.3) Total path-length of loop B-D-E-B = 120 + 160 + 200 = 480

(ii) Possible choices of routs are:

(ii.1) Fig. 2-17 (c): New increment of total path-length = 120 + 160 = 280, which is longer than one half of the corresponding loop B-C-E-D-B = 460/2 = 230, and also longer than one half of the corresponding loop B-D-E-B = 480/2 = 240. Hence, this is not a solution.

(ii.2) Fig. 2-17 (d): New increment of total path-length = 200, which is longer than one half of the corresponding loop B-C-E-B = 380/2 = 190, and also longer than one half of the corresponding loop B-D-E-B = 480/2 = 240. Hence, this is not a solution.

(ii.1) Fig. 2-17 (e): New increment of total path-length = 100 + 80 = 180, which is shorter than one half of the corresponding loop B-C-E-B = 380/2 = 190, and also shorter than one half of the corresponding loop B-C-E-D-B = 460/2 = 230. Hence, this is a solution, giving A-B-C-E-F-E-D-B-E-C-B-A.

Clearly, solutions are not unique, at least how to traverse a solution route is not unique.

2.5 The shortest path length problem

Consider the graph shown in Fig. 2-18, where nodes $A-L$ denote cities, which are connected by roads (edges). If the distances (lengths) of the roads are as marked, what is the shortest path length from node A to node L?

A few remarks are in order. First, an upper bound for the shortest path can be easily found: arbitrarily connecting A to L certainly gives an upper bound, although may not be the largest upper bound. Second, the numbers shown in the figure do not necessarily refer to distances; they can be considered as other physical means such as time, cost, traffic flows, bandwidth, etc., and may be referred to as ***weights***, giving a ***weighted graph*** in general. The problem is then to find a path from A to L with the minimum total weight. Finally, there are several different methods that can be used to solve the problem, while a typical one is the ***Dijkstra algorithm*** introduced below.

It should be noted that the Dijkstra algorithm is a greedy algorithm, namely, it searches through all possible ways to find a shorts path, therefore has very high computational complexity: for a network of N nodes and M edges, its complexity is $O(M + N\ln N)$.

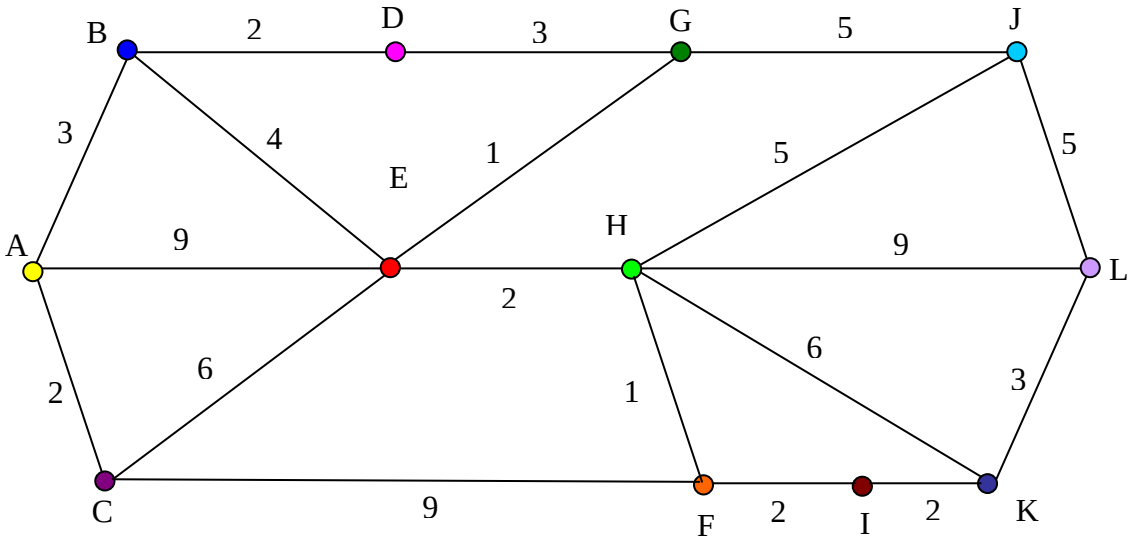


Fig. 2-18 Example of the shortest path length problem

A natural mathematical approach to solving the problem is to walk from A to L, associating each intermediate node V with a number $l(V)$ that is equal to the shortest path length from A to V. For example, when walking from A to J, one has

$l(J)=l(G)+5$ or $l(J)=l(H)+5$ or even $l(J)=l(L)+5$, whichever is shorter. Thus, the procedure is carried out as follows:

Starting from A with $l(A)=0$, moving toward L, one has $l(B)=l(A)+3=3$, $l(E)=l(A)+9=9$ and $l(C)=l(A)+2=2$; therefore node C is chosen with $l(C)=2$.

Looking at the nodes adjacent to B, one has $l(D)=l(B)+2=5$ and $l(E)=l(B)+4=7$; looking at the nodes adjacent to C, one has $l(E)=l(C)+6=8$ and $l(F)=l(C)+9=11$. Here, $l(D)=5$ is uniquely determined, but $l(E)=7$, or = 8, or = 9, so node E is chosen with the shortest: $l(E)=l(B)+4=7$.

Continuing this way, one finally found the shortest lengths of all nodes in the graph: $l(A)=0$, $l(B)=3$, $l(C)=2$, $l(D)=5$, $l(E)=7$, $l(F)=10$, $l(G)=8$, $l(H)=9$, $l(I)=12$, $l(J)=13$, $l(K)=14$, $l(L)=17$. This means that the shortest path length is 17, which cannot be further reduced.

For the example shown in Fig. 2-18, one solution is given in Fig. 2-19. But it should be noted that such a possible path may not be unique, especially for a large-sized graph.

It should also be noted that the numbers on the graph do not have to be “distances”; instead, they can be costs, weights, time, etc.

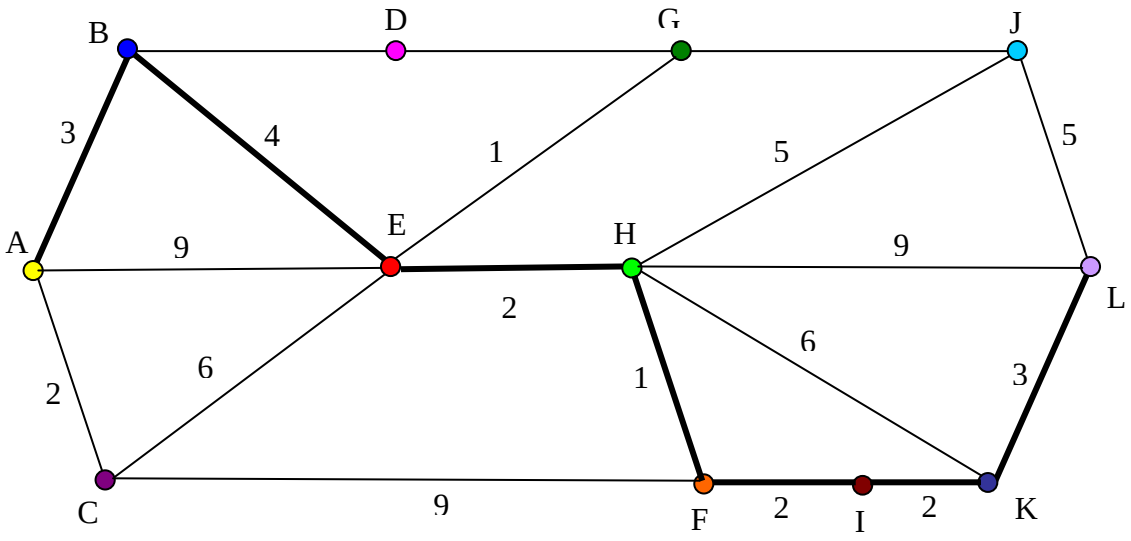
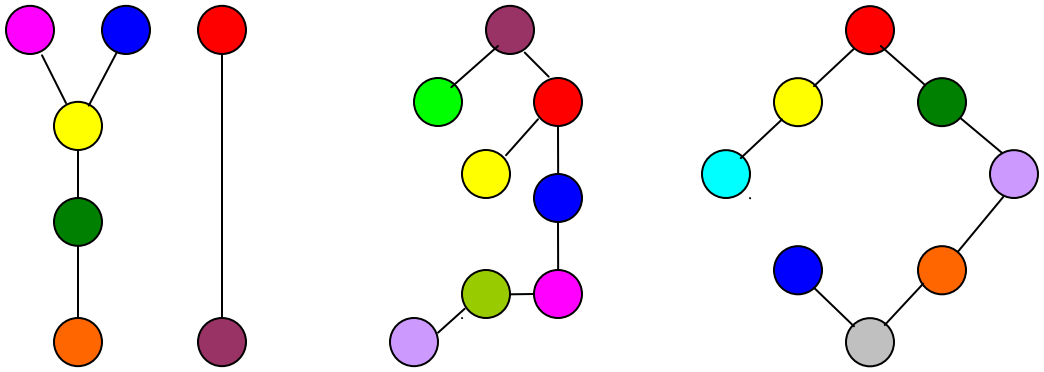


Fig. 2-19 One solution to the shortest path length problem

2.6 Trees

A graph containing no circuits is called a ***forest***, while a connected forest is called a ***tree***. In other words, a forest is a family of trees. Some examples of forest are shown in Fig. 2-20, where (b) and (c) are trees.



(a) (b) (c)

Fig. 2-20 Examples of forest and tree

By the handshaking lemma (Theorem 2-1), in a tree the sum of degrees of all nodes is equal to two times the number of edges; that is, for a tree with N nodes, the number of its edges is $N-1$ and the sum of all its node degrees is equal to $2(N-1)$. It follows that if $N \geq 2$ then the tree always contains at least two end-nodes, as can be seen from Figs. 2-20 (b) and (c).

Theorem 2-18 *Let T be a graph with N nodes. Then, the following statements are equivalent:*

1) *T is a tree;*
2) *T has $N-1$ edges but contains no circuits;*
3) *T has $N-1$ edges and is connected;*
4) *T is connected and every edge is a bridge;*
5) *every pair of nodes in T are connected by exactly one path;*
6) *T contains no circuits, but the addition of any new edge creates exactly one circuit.*

Proof. Without loss of generality, assume $N \geq 2$.

1) $\Rightarrow$ 2) Since T is a tree, it contains no circuits. So, the removal of any edge will disconnect T into two subgraphs, each of which is a tree and each tree has one fewer edge than the number of its nodes. This implies that the total number of edges of T is $N-1$.

2) $\Rightarrow$ 3) If T is disconnected, then each component of T is a connected subgraph without circuits. Hence, by part 1), the number of nodes in each component has one more

than the number of its edges. Consequently, in T the total number of nodes is more than the total number of edges at least by two, contradicting the fact that T has $N-1$ edges.

3) $\Rightarrow$ 4) The removal of any edge from T will result in a graph with N nodes and $N-2$ edges, which must be disconnected according to Theorem 2-7.

4) $\Rightarrow$ 5) Since T is connected, each pair of nodes is connected by at least one path. If a given pair of nodes is connected by two paths, then they enclose a circuit, contradicting the fact that every edge is a bridge.

5) $\Rightarrow$ 6) If T contained a circuit, then any pair of nodes in the circuit would be connected by at least two paths, contradicting part 5). If an edge e is added to T, then since the nodes connecting with e are already connected in T, a circuit will be created. Moreover, this circuit is unique since otherwise e would be the common edge of two circuits but them these two circuits remain to be a circuit without e, contradicting the fact that T contains no circuits at the beginning.

6) $\Rightarrow$ 1) If T was disconnected, then by adding an edge to join any two components of T would not create a circuit. Therefore, T must be connected. Moreover, by assumption T contains no circuits, so it is a tree. □

Corollary 2-19 *A forest with N nodes and K components has $N-K$ edges. Consequently, a tree with N nodes has $N-1$ edges.*

Proof. Applying part 3) of Theorem 2-18 to each component of the forest leads to the first conclusion. The second conclusion follows from the fact that a tree has only one component, $K=1$. □

Starting from a given graph, if it has a circuit, then remove one edge from the circuit. Clearly, the resulting graph remains to be connected. Repeat this procedure until no circuits are left out. Then, the final resulting graph is a tree. This tree is called a ***spanning tree*** of the graph given at the beginning. More generally, if the graph has N nodes, M edges and K components, then one can carry out the above procedure on each component of the graph, resulting in a ***spanning forest***. The total number of edges removed throughout this procedure is called the ***circuit bank*** number, which is equal to $M-N+K$.

Theorem 2-20 *Let T be any spanning forest of a graph G. Then,*

1) *every cut-set of G has an edge in common with T;*
2) *every circuit of G has an edge in common with the complementary graph of T.*

Proof. 1) Let C be a cut-set of G. Removing C will split one of the components of G into two subgraphs, G_1 and G_2. Then, since T is a spanning forest, it must contain an edge joining a node of G_1 to a node of G_2, which is the required edge.

2) Let C be a circuit of G that has no edges in common with the complementary graph of T. Then, C must be contained in T, which is a contradiction. □

2.7 The minimum connector problem

Suppose one wants to build a highway network connecting n given cities, in such a way that a car can travel from any city to any other city, but the total mileage of the highways is minimum. This is the so-called ***minimum connector problem***.

Clearly, the graph formed by taking the N cities as nodes and the connecting highways as edges must be a tree because any more highway will be extra. The problem is to find an efficient algorithm to decide which of all the possible trees connecting these cities reaches the minimum total mileage, given that the distance between any pair of cities is known.

It should be noted that the numbers on the graph do not have to be "distances"; instead, they can be costs, weights, time, etc.

Just as the shortest path length problem, discussed above, one may reformulate this problem in terms of a weighted network, where the weight is the distance, and to find which tree has the minimum total weight. The resulting tree should be a minimum-weight spanning tree. It turns out that there is a simple and efficient algorithm, known as the ***Kruskal*** (***greedy***) ***algorithm***, for solving this problem. The computational complexity of this algorithm is $O(M \ln M)$ for a network with M edges.

Theorem 2-21 (Kruskal Algorithm) *Let G be a connected graph with N nodes. Then, the following constructive scheme yields a solution to the minimum connector problem:*

1) *let e_1 be an edge of G with the smallest weight;*
2) *choose $e_2,...,e_{N-1}$ successively by choosing each edge e_i, not previously chosen, with a smallest weight, subject to the condition that it forms no circuit with all the previous edges $\{e_1,...,e_{i-1}\}$;*
3) *repeat the above procedure for $i = 3,...,N$.*

The resulting graph is a spanning tree, the subgraph of G with edges $e_1,...,e_{N-1}$.

Proof. The resulting subgraph T is a spanning tree follows immediately from Theorem 2-18 (ii). It remains to show that the total weight of T, denoted $W(T)$, is minimum.

Suppose that S is a spanning tree of G such that $W(S) < W(T)$. If e_k is the first edge in the above sequence which is not contained in S, then the subgraph of G formed by adding e_k to S contains a unique circuit C that contains e_k. Since C must contain an edge e belonging to S but not to T, the subgraph obtained from S with e being replaced by e_k is still a spanning tree, denoted as S_0. But, by the construction, the edge weight $w(e_k) < w(e)$, therefore $W(S_0) \le W(S)$, and S_0 has one more edge in common

with T than with S. By repeating this procedure, it follows that one can change S into T at each step, with the total weight decreasing at each time. Consequently, $W(T) \le W(S)$, which is a contradiction to the above assumption $W(S) < W(T)$. □

As an example, consider the graph shown in Fig. 2-21. By the greedy algorithm, an optimal result can be found as follows:

1) Find an edge with the smallest weight, which is $e_1 = AB = 2$.
2) Find an edge with the smallest weight from the rest edges, which is $e_2 = DE = 3$.
3) Find an edge with the smallest weight from the rest edges, which is $e_3 = AD = 4$.
4) Find an edge with the smallest weight from the rest edges, which is $e_4 = BC = 7$, since any other choice will form a circuit, or will be longer than 7.
5) Stop, because a spanning tree has been obtained, i.e., adding any more edge will form a circuit.

The final result is given by $\{AB, AD, BC, DE\}$, as shown by the bold edges in Fig. 2-21.

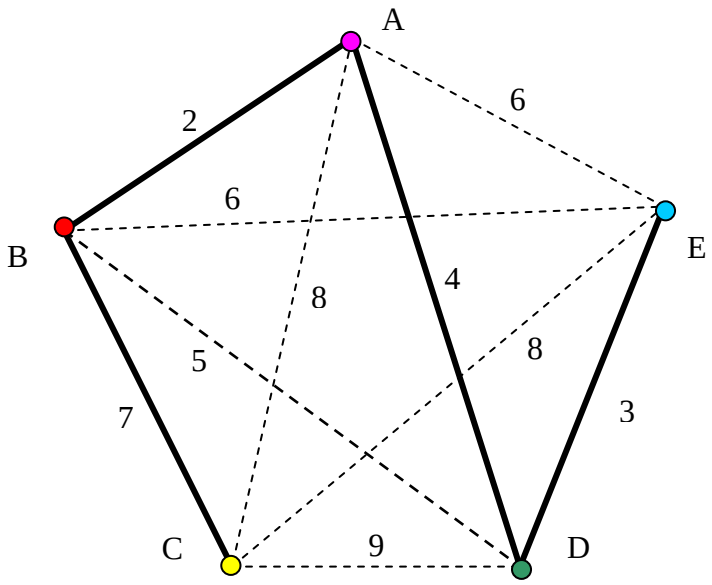
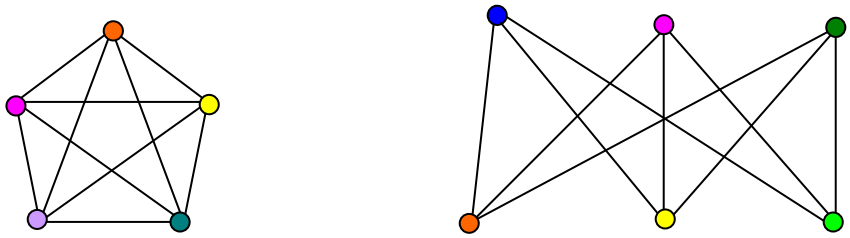


Fig. 2-21 Example of the minimum connector problem

2.8 Plane graphs and planar graphs

A ***plane graph*** is one that can be drawn on the plane without crossing edges. A ***planar graph*** is one that is isomorphic to a plane graph. A plane graph is certainly a planar graph, but the main reason to distinguish them will become clear from Theorem 2-23 below where the concept "face" can be defined for plane graphs but not for planar graphs in general.

A triangle and a square are plane graphs, and a cube in the Euclidean 3-space is a planar graph. However, the two graphs shown in Fig. 2-22 (a) and (b) are both non-planar graphs (therefore no-plane graphs). These two non-planar graphs, called K_5 and $K_{3,3}$ respectively, are very important graphs, as will be seen from Theorem 2-22 below.



(a) Graph "K_5" (b) Graph "$K_{3,3}$"

Fig. 2-22 Two important examples of non-planar graphs

Two graphs are said to be ***homeomorphic*** if they both can be obtained from the same graph by inserting new nodes of degree two into edges. Therefore, they are identical except possibly some nodes of degree two. For example, the two graphs shown in Fig. 2-23 are homeomorphic.

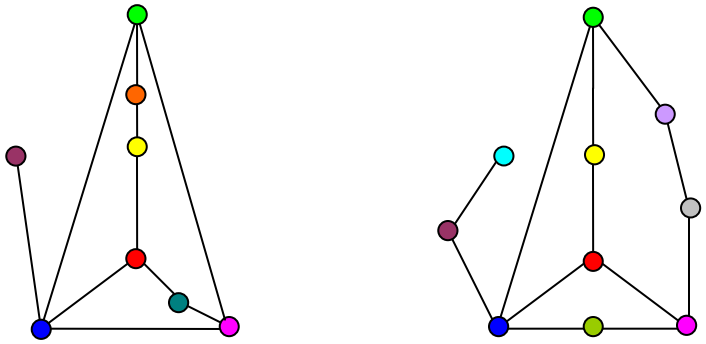


Fig. 2-23 Example of two homeomorphic graphs

Theorem 2-22 (Kuratowski Theorem) *A graph is planar if and only if it contains no subgraphs homomorphic to* K_5 *or* $K_{3,3}$.

Proof. A proof is mathematically involved, therefore omitted here.

2.9 Euler formula for plane graphs

Now, one more new concept is introduced: ***face***. In Fig. 2-24, there are four faces: F_1, F_2, F_3 and F_4, where the last one is an infinite face.

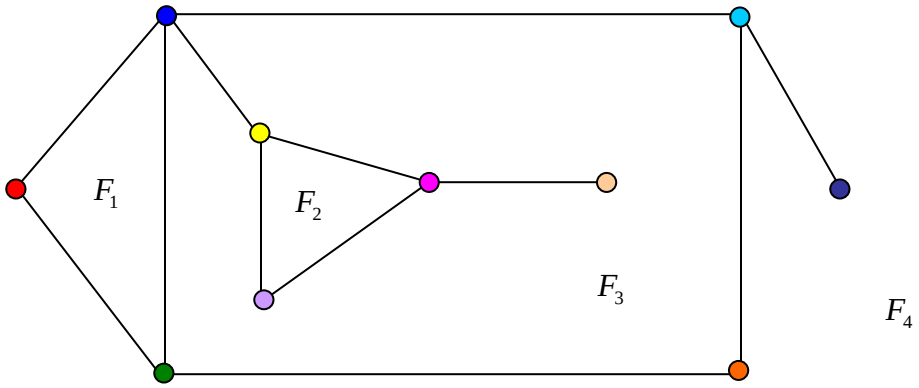


Fig. 2-24 Example of a graph with four faces

Theorem 2-23 (Euler, 1750) *Let N, E and F be the number of nodes, edges and faces of a connected plane graph, respectively. Then,*

$$N - E + F = 2$$

Proof. Apply induction on the number of edges of the graph as follows. If $E = 0$ then $N = 1$, since the graph is connected, and in this case $F = 1$, the infinite face. So the theorem is true. Suppose that the theorem is true for all graphs with at most $E - 1$ edges, and then consider a connected plane graph with E edges. If the graph is a tree, then $E = N - 1$ and $F = 1$, the infinite face, therefore the theorem is true. If the graph is not a tree, then remove an edge E from any circuit in the graph. This will result in a connected plane graph with N nodes, $E - 1$ edges and $F - 1$ faces, so that $N - (E - 1) + (F - 1) = 2$ by the induction hypothesis, which gives $N - E + F = 2$. This completes the induction. □

For example, the connected plane graph shown in Fig. 2-24 has $N = 10$, $E = 12$ and $F = 4$, which verifies Theorem 2-23.

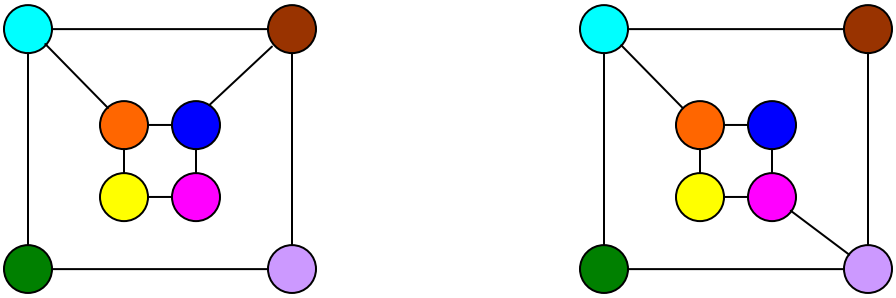
As another example, consider a polyhedron. Any polyhedron can be projected on a sphere, where the resulting graph is called a ***polyhedral graph***, which is isomorphic to a plane graph, therefore is a planar graph. Consequently, Theorem 2-23 holds for any polyhedron: $N - E + F = 2$, which is called the Euler polyhedron formula.

Problems

2-1 Argue that Corollary 2-2 is correct, and show some examples to support the result.

2-2 Verify that the two graphs shown in Fig. 2-7 are isomorphic, by indicating their one-one correspondence.

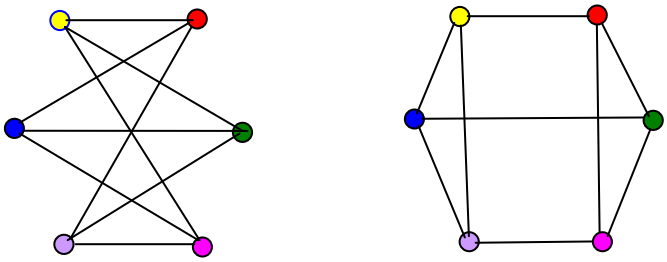
2-3 Are the two graphs shown in Fig. 2-25 isomorphic? Why?



(a) (b)

Fig. 2-25 Are these two graphs isomorphic?

2-4 Find the complementary graphs of the two graphs shown in Fig. 2-26.



(a) (b)

Fig. 2-26 Find the complementary graphs

2-5 Can you find a disconnected graph such that its complementary graph is also disconnected? If so, show an example; if not, tell why.

2-6 (a) Let G be a random graph and G^c be its complementary graph. State three major properties of G^c.

(b) Let G be a scale-free graph and G^c be its complementary graph. State three major properties of G^c.

2-7 Find the adjacency, incidence and Laplacian matrices of the graphs shown in Fig. 2-26.

2-8* Show that Laplacian matrix L has the following properties:

(a) L is always semi-positive definite;

(b) L has eigenvalues $\lambda_0 \le \lambda_1 \le \cdots \le \lambda_n$, with $\lambda_0 = \cdots = \lambda_q = 0$ where q is the number of disjoint subgraphs in the graph; consequently, if the graph is connected, i.e., $q = 1$, then $0 = \lambda_0 < \lambda_1 \le \cdots \le \lambda_n$.

2-9 On the surface of the globe, use a node to represent a continent, so there are 7 nodes in total, labeled by A, B, C, D, E, F, G. Assume that there is an edge joining each pair of nodes, i.e., they are the nodes of a complete regular graph. Is this graph planar, and why? If so, draw an illustrative picture of this planar graph.

2-10 Verify Theorem 2-6 by the example shown in Fig. 2-7.

2-11 Verify Theorem 2-7 by an example of a graph with 3 components and 10 nodes, and by a tree with 10 nodes.

2-12 For an arbitrarily given graph of N nodes, what is the minimal number of edges you need to remove so that you can guarantee to obtain a new graph with at least 2 components? with at least 3 components? and, in general, with at least K ($2 \le K \le N$) components?

2-13 Show all the cut sets in Fig. 2-25 (b).

2-14 Distinguish Eulerian, semi-Eulerian and non-Eulerian graphs shown in Fig. 2-27.

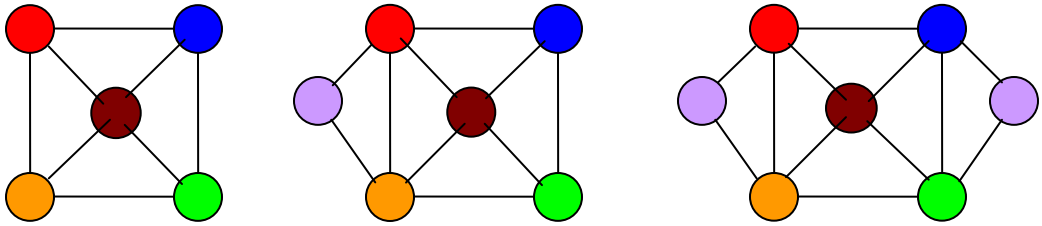


Fig. 2-27 Distinguish the Eulerian, semi-Eulerian and non-Eulerian graphs

2-15 Show that if in a graph all nodes have degrees no less than 4 then this graph contains at least 2 circuits.

2-16 Use the Fleury Algorithm (Corollary 2-16) to find an Eulerian trail in the Eulerian graph that you identified in Fig. 2-27 of Problem **2-14**.

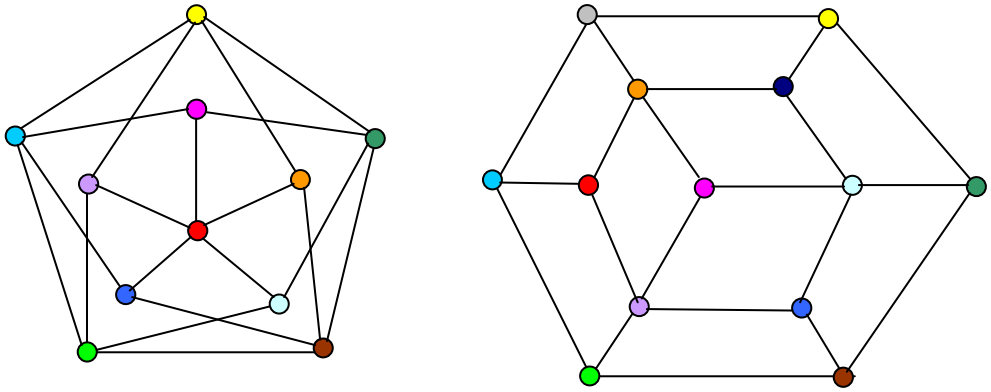


Fig. 2-28 Distinguish Hamiltonian and non-Hamiltonian graphs

2-17 In Fig. 2-28, which is a Hamiltonian graph and which is not?

2-18 In 1856, the English mathematician William R. Hamilton studied the world navigation problem and considered a map with 20 nodes representing cities connected by sailing routes, as depicted by Fig. 2-29. He wanted to find out if one can traverse through every city once and once only, and finally return to the starting city. His study eventually led to the establishment of the now-famous Hamiltonian graph theory. Please try to figure out whether Hamilton was able to solve his problem and show how, or tell why not?

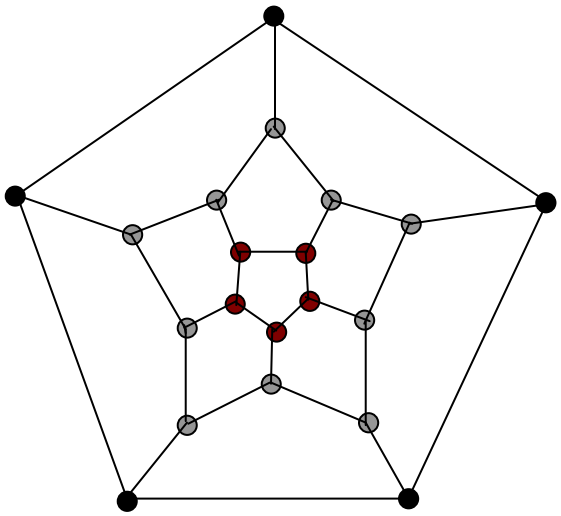


Fig. 2-29 Hamiltonian world navigation problem

2-19 Find a shortest path from A to G in Fig. 2-30.

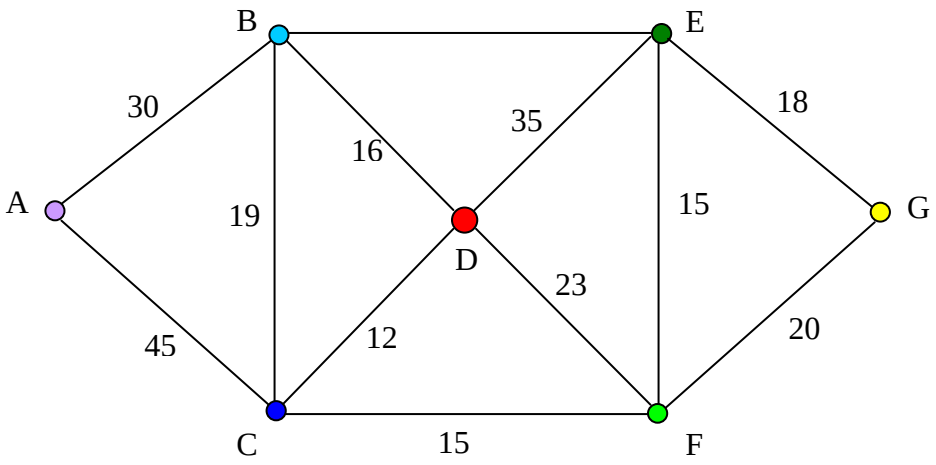


Fig. 2-30 Find a shortest path length

2-20 Find a solution of the Chinese postman problem for the postman route map shown in Fig. 2-31.

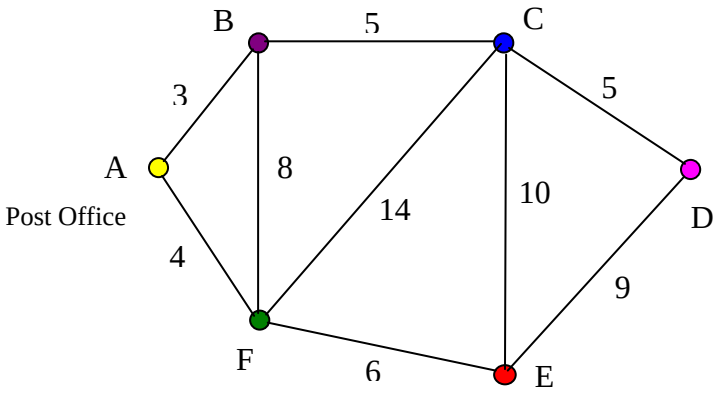


Fig. 2-31 A Chinese postman problem

2-21 Find all spanning trees in the graph shown in Fig. 2-32.

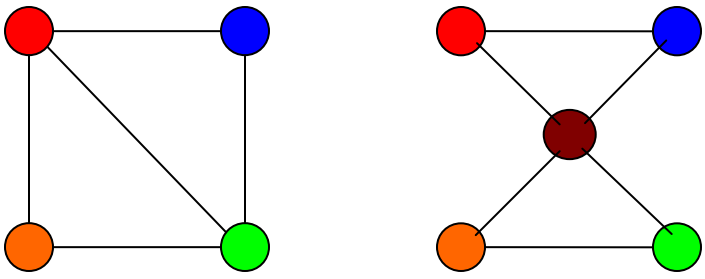


Fig. 2-32 Find all spanning trees

2-22 Apply the greedy algorithm (Theorem 2-21) to solve the minimum connector problem from point A to point B in the graph shown in Fig. 2-33, where the numbers are costs.

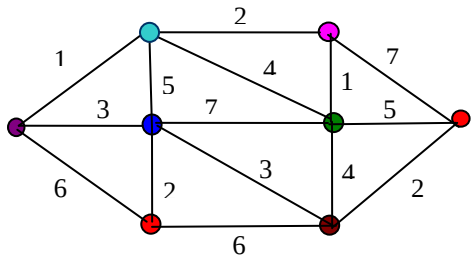


Fig. 2-33 A minimum connector problem

2-23 Consider a computer network in a lab, as shown by Fig. 2-34, where each node is a computer (Router or PC) and the numbers indicate the lengths of optical fibers needed if the computers are connected.

(a) Design a best cost-effective Personal Area Network (PAN) by connecting all nodes together, so that every computer can communicate with every other one while the total optical fiber used is the shortest possible. Explain your steps, and what is the total length of optical fibers you need?

(b) Find the shortest path from computer A to computer B, which uses the shorter possible optical fiber. Show your reasoning.

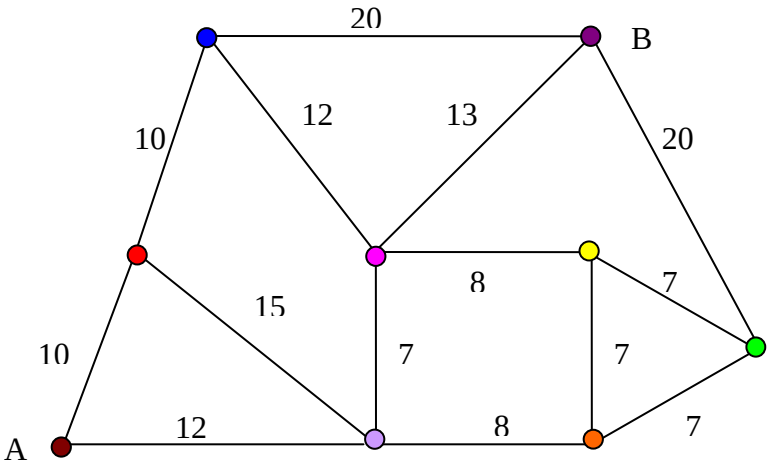


Fig. 2-34 Computer network in a lab

2-24 Verify that the two graphs shown in Fig. 2-22 (a) and (b), namely, K_5 and $K_{3,3}$, are both non-planar graphs. Then, show two more simple examples of non-planar graphs.

2-25 Give some simple circuit examples to verify that any two circuits are homeomorphic.

2-26 Figure 2-3, describing the Euler seven-bridge problem, appears to contain double edges between nodes A, B, C, therefore is not a simple graph. Based on the concept of "homeomorphic graphs" and Theorem 2-14, explain that this seven-bridge problem indeed has no solutions, namely, Corollary 2-15 is perfectly correct.

2-27 Verify that the graph shown in Fig. 2-17 satisfies the Euler formula given in Theorem 2-23.

2-28 Verify that any polyhedron can be mapped to the polyhedral graph shown in Fig. 2-35, and verify that the Euler polyhedron formula holds for the polyhedral graph.

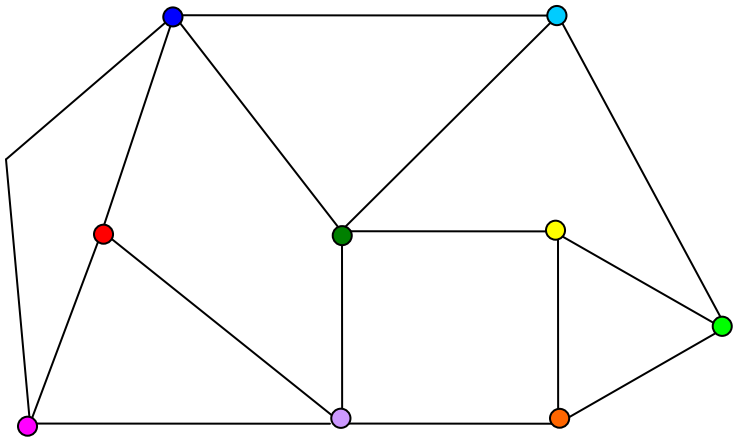


Fig. 2-35 The polyhedral graph

2-29* Prove that for any plane graph with N nodes, E edges, F faces and K components, it is always true that $N - M + F = K + 1$. [*Hint:* Apply Theorem 2-23 to each component separately, noticing that the infinite face should not be counted for more than once.]

2-30 The network shown in Fig. 2-36 is called a *binary tree*.

(a) Starting from the top, there is $2^0 = 1$ node on the first layer, 2^1 nodes on the second layer, 2^2 nodes on the third layer, …, 2^n nodes on the $(n+1)$st layer. For a binary-tree network with m layers, (i) what is the total number of nodes in the network? (ii) what is the total number of edges in the network?

(b) In order to improve the overall robustness against various attacks on this binary-tree traffic network, you are allowed to add an edge either to connect node B with node C, or to connect node A with node D. Which connection is better, or they are the same, and why?

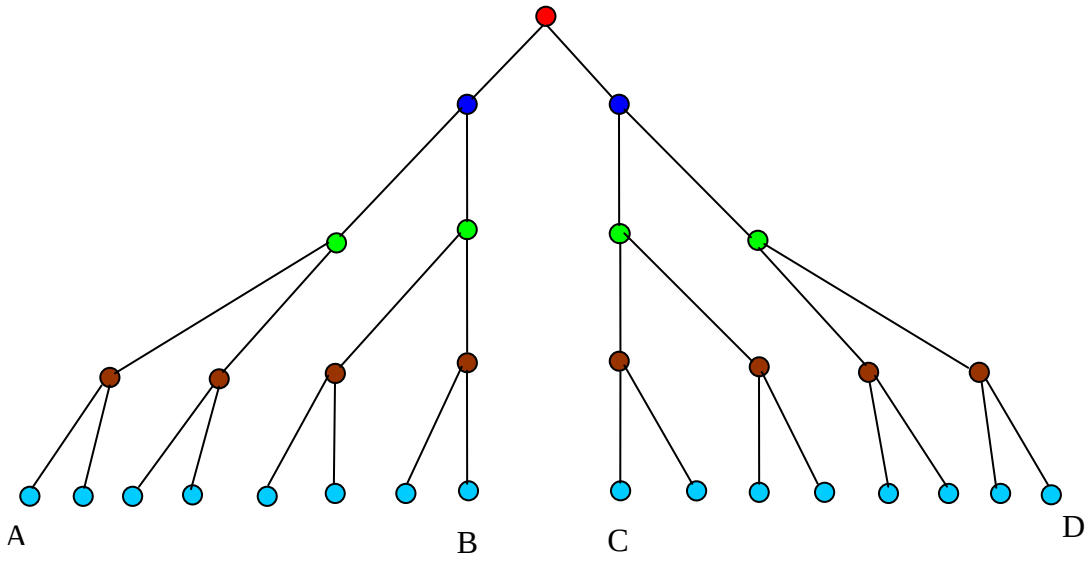


Fig. 2-36 A binary-tree network

References

[1] http://www-history.mcs.st-andrews.ac.uk/Extras/Konigsberg.html
[2] Wilson, R J. *Introduction to Graph Theory*, 3rd edition, Longman House, England, 1985
[3] Thulasiraman K, Swamy M N S. *Graphs: Theory and Algorithms*, Wiley, New York, 1992

