DISCRETE MATHEMATICS
（离散数学）

云南大学数学系
李 建 平

Chapter 8 Topics in Graph Theory

8.1 GRAPHS

A **graph** G consists of a finite set V of objects called **vertices**, a finite set E of objects called **edges**, and a function γ that assigns to each edge a subset $\{v, w\}$, where v and w are vertices (and may be the same). We write $G = (V, E; \gamma)$. If e is an edge, and $\gamma(e) = \{v, w\}$, we say that e is an edge between v and w that e is determined by v and w. The vertices v and w are called the **end-vertices** of e. If there is only one edge between v and w, we often identify e with the set $\{v, w\}$.

Graphs are usually represented by pictures using a point for each vertex and a line for each edge.

An edge between v_i and v_j indicates a connection between the objects v_i and v_j.

The **degree** of a vertex is the number of edges having that vertex as an end-vertex. A graph may contain an edge from a vertex to itself; such an edge is referred to as a **loop**. A loop contributes 2 to the degree of a vertex. A vertex with degree 0 will be called an **isolated vertex**. A pair of vertices that determine an edge are **adjacent vertices**.

A **path** π in a graph G consists of a pair (V_π, E_π) of sequences; a vertex sequence $V_\pi : v_1, v_2, \cdots, v_k$ and an edge sequence $E_\pi : e_1, e_2, \cdots, e_{k-1}$ for which:

(1). Each successive pair v_i, v_{i+1} of vertices is adjacent in G, and edge e_i has v_i and v_{i+1} as end-vertices, for $i = 1, 2, \cdots, k-1$.

(2). No edge occurs more than once in the edge sequence.

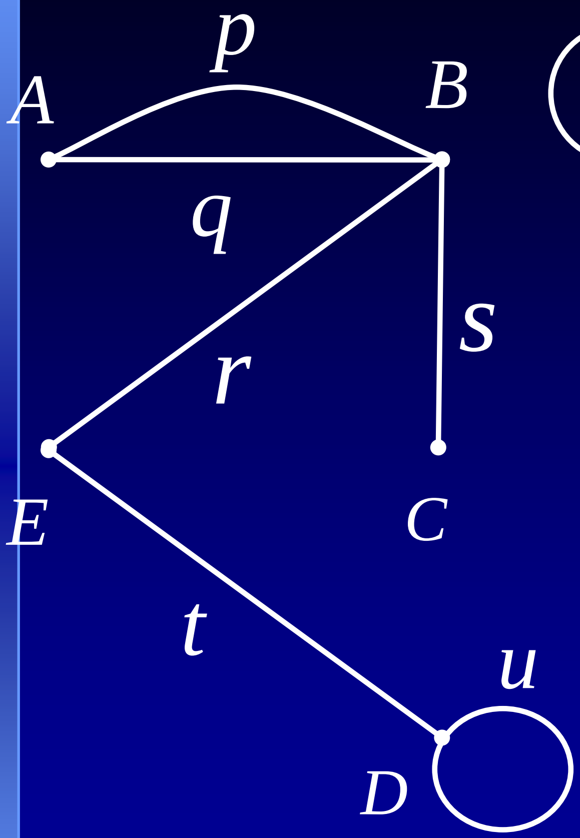


Figure 8.4

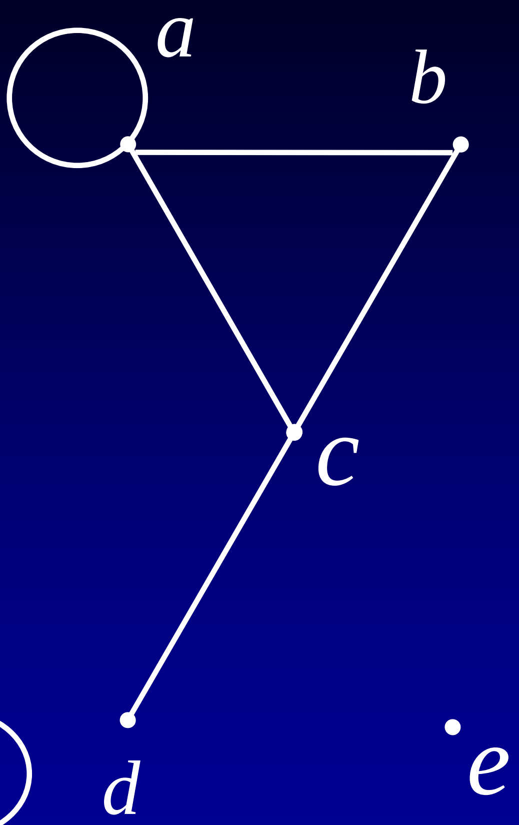


Figure 8.5

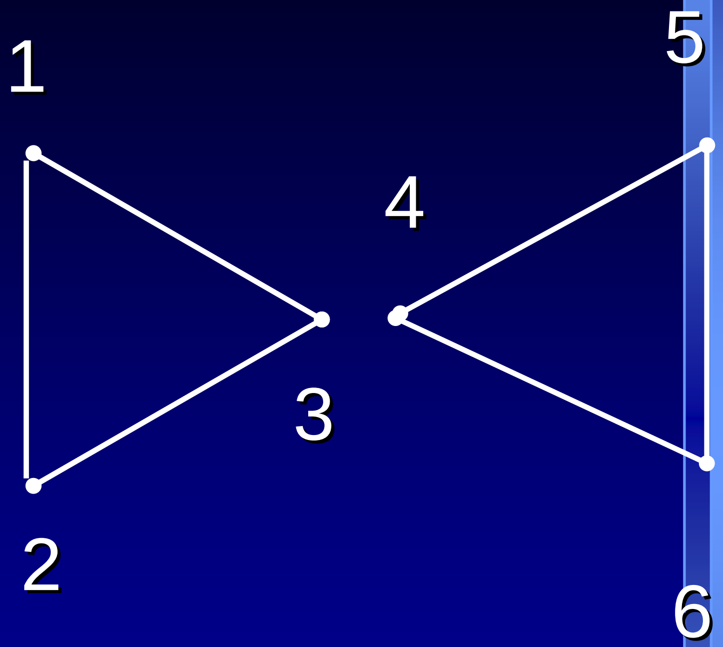


Figure 8.6

A **circuit** is a path that begins and ends at the same vertex. A path is called **simple** if no vertex appears more than once in the vertex sequence, except possibly if $v_1 = v_k$.

A graph is called **connected** if there is a path from any vertex to any other vertex in the graph. Otherwise, the graph is **disconnected**. If the graph is disconnected, the various connected pieces are called the **components** of the graph.

Some important special families of graphs will be useful in our discussions. We present them here.

1. For each integer $n \geq 1$, we let K_n^c or U_n denote the graph with n vertices and no edges. We called K_n^c the **discrete (or empty) graph** on n vertices.

2. For each integer $n \geq 1$, let K_n denote the graph with vertices $\{v_1, v_2, \cdots, v_n\}$ and with an edge $\{v_i, v_j\}$ for every i and j. The graph K_n is called the **complete graph** on n vertices. If each vertex of a graph has the same degree as every other vertex, the graph is called **regular**（正则的）.

3. For each integer $n \geq 1$, we let L_n denote the graph with n vertices $\{v_1, v_2, \cdots, v_n\}$ and with edges

$\{v_i, v_{i+1}\}$ for $1 \le i < n$. We call L_n the **linear graph** on n vertices.

- Subgraphs and Quotient Graphs

Suppose that $G=(V,E;\gamma)$ is a graph. Choose a subset E_1 of the edges in E and a subset V_1 of the vertices in V, so that V_1 contains (at least) all the end-vertices of edges in E_1. Then $H=(V_1,E_1;\gamma_1)$ is called a subgraph of G.

If $G=(V,E;\gamma)$ is a graph and $e \in E$, then we denote by $G-\{e\}$ or G_e the subgraph obtained by omitting the edge e from E and keeping all vertices in G .

Suppose that $G=(V,E;\gamma)$ is a simple graph and that R is an equivalence relation on the set V. We construct the **quotient graph**（商图）G^R. The vertices of G^R are the equivalence classes of V produced by R. If [*v*] and [*w*] are the equivalence classes of vertices v and w of G, then there is an edge in G^R from [v] to [w] if and only if some vertex in [v] is connected to some vertex in [w] in G.

EXAMPLE 9

Let G be the graph of Figure 8.14 (which has no multiple edges), and let R be the equivalence relation on V defined by the partition

$$\{\{a,m,i\},\{b,f,j\},\{c,g,k\},\{d,h,l\}\}$$

Then G^R is shown in Figure 8.15.

If S is also an equivalence relation on V defined by the partition

$$\{\{i,j,k,l\},\{a,m\},\{f,b,c\},\{d\},\{g\},\{h\}\}$$

then the quotient graph G^S is shown in Figure 8.16.

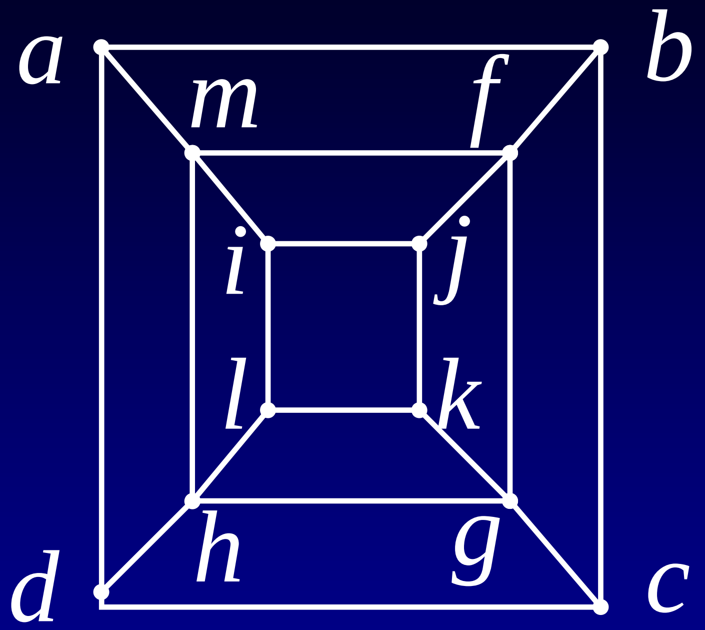


Figure 8.14

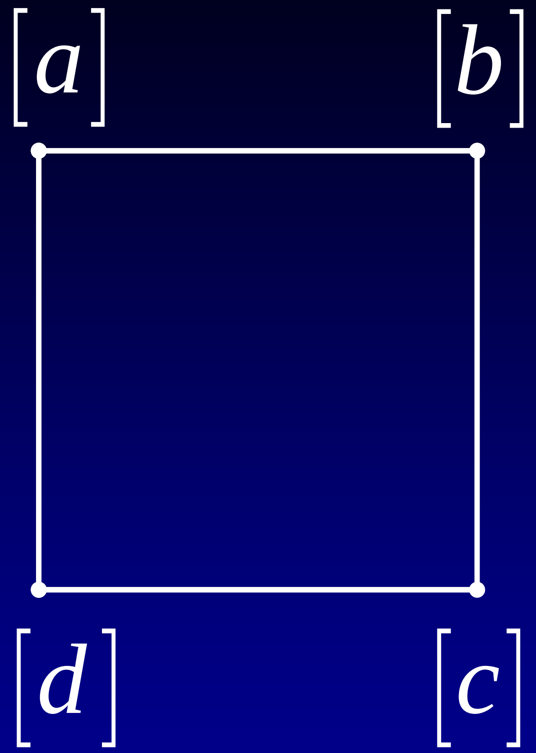


Figure 8.15

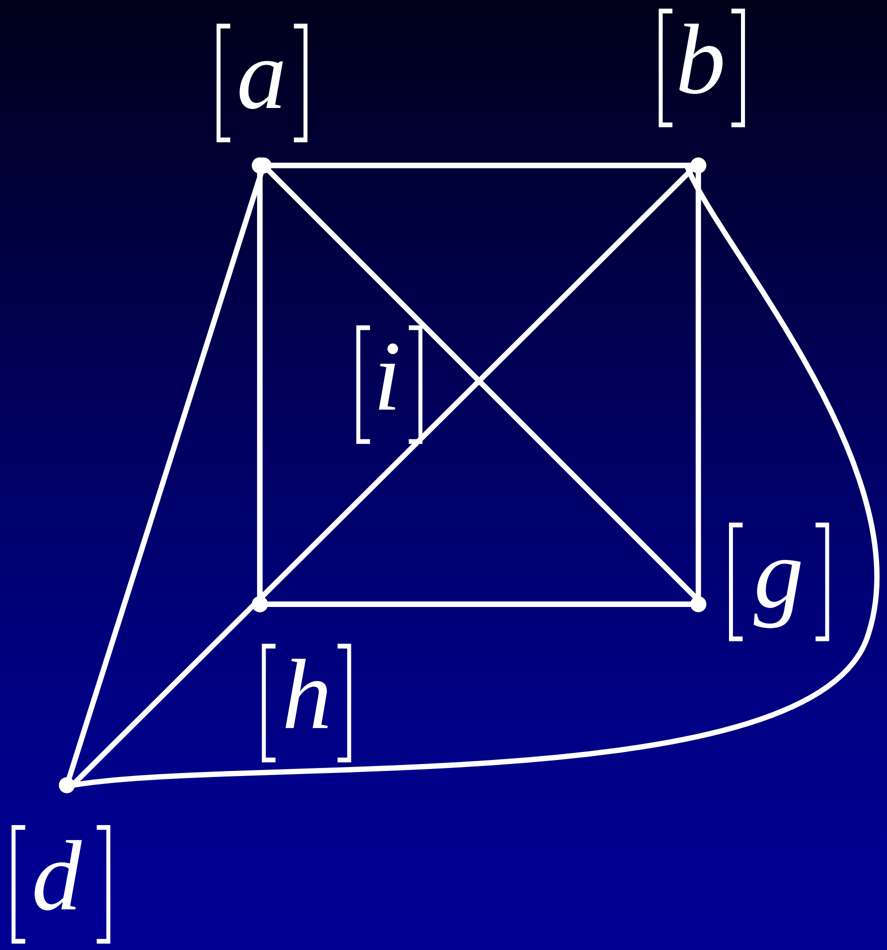


Figure 8.16

8.2 EULER PATHS AND CIRCUITS

A path in a graph G is called an **Euler path**（欧拉路）if it includes every edge **exactly** once. An **Euler circuit**（欧拉回路）is an Euler path that is a circuit.

Theorem 1

(a) If a graph G has a vertex of odd degree, there exists no Euler circuit in G.

(b) If G is a connected graph and every vertex has even degree, then there is an Euler circuit in G.

Theorem 2

(a) If a graph G has more than two vertices of odd degree, then there exists no Euler path in G.

(b) If G is connected and has exactly two vertices of odd degree, there is an Euler path in G. Any Euler path in G must begin at one vertex of odd degree and end at the other.

We give an algorithm that produces an Euler circuit for a connected graph with no vertices of odd degree. An edge is a **bridge**（桥）in a connected graph G if deleting it would create a disconnected graph.

FLEURY'S ALGORITHM

Let $G = (V, E; \gamma)$ be a connected graph with each vertex of even degree.

Step 1 Select an edge e_1 that is not a bridge in G. Let its vertices be v_1, v_2. Let π be specified by $V_\pi : v_1, v_2$ and $E_\pi : e_1$. Remove e_1 from E and v_1 and v_2 from V to create G_1.

Step 2 Suppose that $V_\pi : v_1, v_2, \cdots, v_k$ and $E_\pi : e_1, e_2, \cdots, e_{k-1}$ have been constructed so far, and that all of these edges and vertices have been removed from V and E to from G_{k-1}. Since v_k has even degree, and e_{k-1}

ends there, there must be an edge e_k in G_{k-1} that also has v_k as a vertex. If there is more than one such edge, select one that is not a bridge for G_{k-1}.
Denote the vertex of e_k other than v_k by v_{k+1}, and extend V_π and E_π to $V_\pi : v_1, v_2, \cdots v_k, v_{k+1}$ and
$E_\pi : e_1, e_2, \cdots, e_{k-1}, e_k$.

Step 3 Repeat Step 2 until no edges remain in E.

End of Algorithm

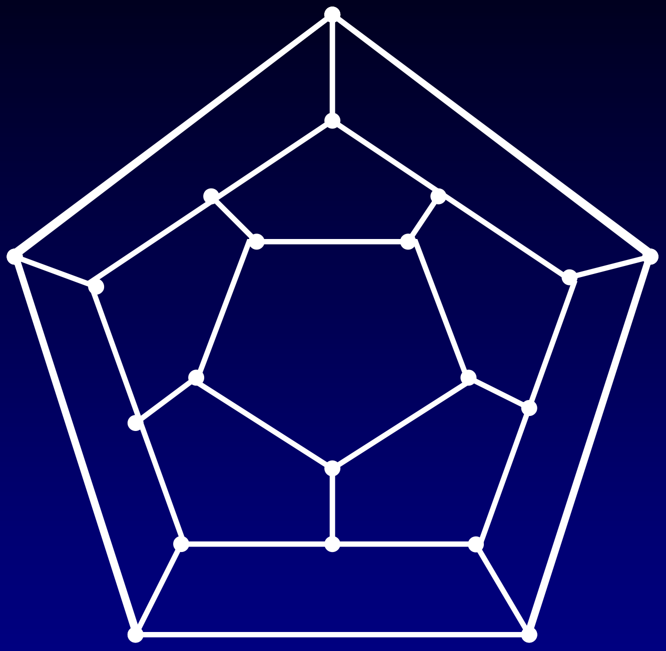
Example

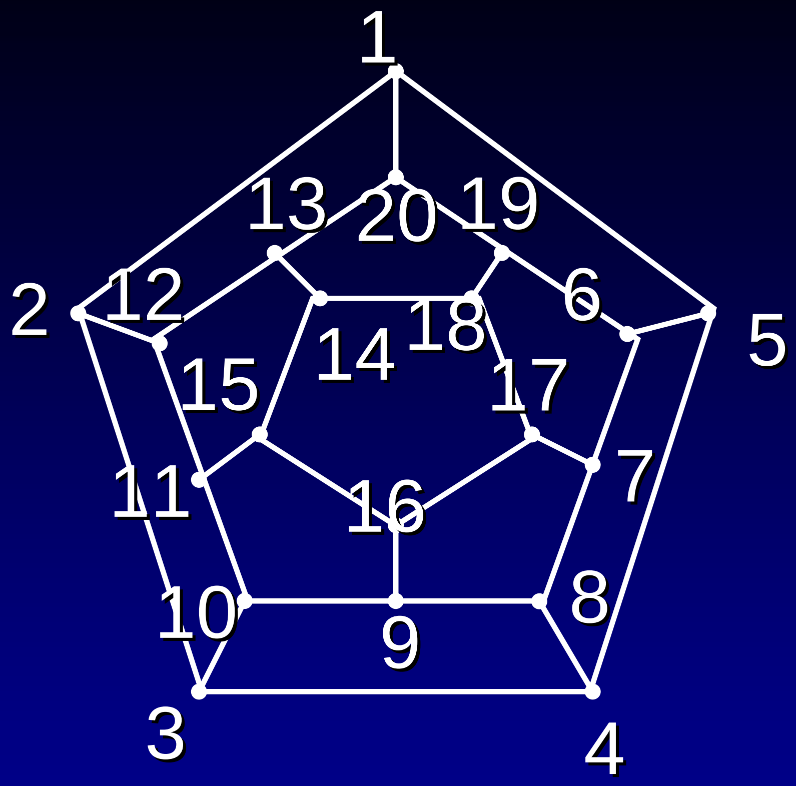
Chinese Postman Problem (管梅谷，1959/60)

8.3 HAMILTONIAN PATHS AND CIRCUITS

A **Hamiltonian path** is a path that contains each vertex **exactly** once. A **Hamiltonian circuit** is a circuit that contains each vertex exactly once except for the first vertex, which is also the last.

A planar version of this solid is shown in Figure 8.54(a), with a Hamiltonian circuit (one of many) shown in Figure 8.54(b) by the consecutively numbered vertices.


(a)



(b)

Figure 8.54

EXAMPLE 2 Any complete graph K_n has H-path.

Question: Is it possible to determine whether a Hamiltonian path or circuit exists?

Theorem 1 (O. Ore,1960) Let G be a connected graph with n vertices, $n \geqslant 2$, and no loops or multiple edges. If for any two vertices u and v of G that are not adjacent, the degree of u plus the degree of v is greater than or equal to n, then G has a Hamiltonian circuit.

Theorem 2 Let the number of edges of G be m. If $m \geq \frac{1}{2}(n^2 - 3n + 6)$, then G has a Hamiltonian circuit. (where n is the number of vertices).

EXAMPLE 3

The converses of Theorems 1 and 2 given above are not true; that is, the conditions given are sufficient. Consider the graph represented by the figure 8.59.

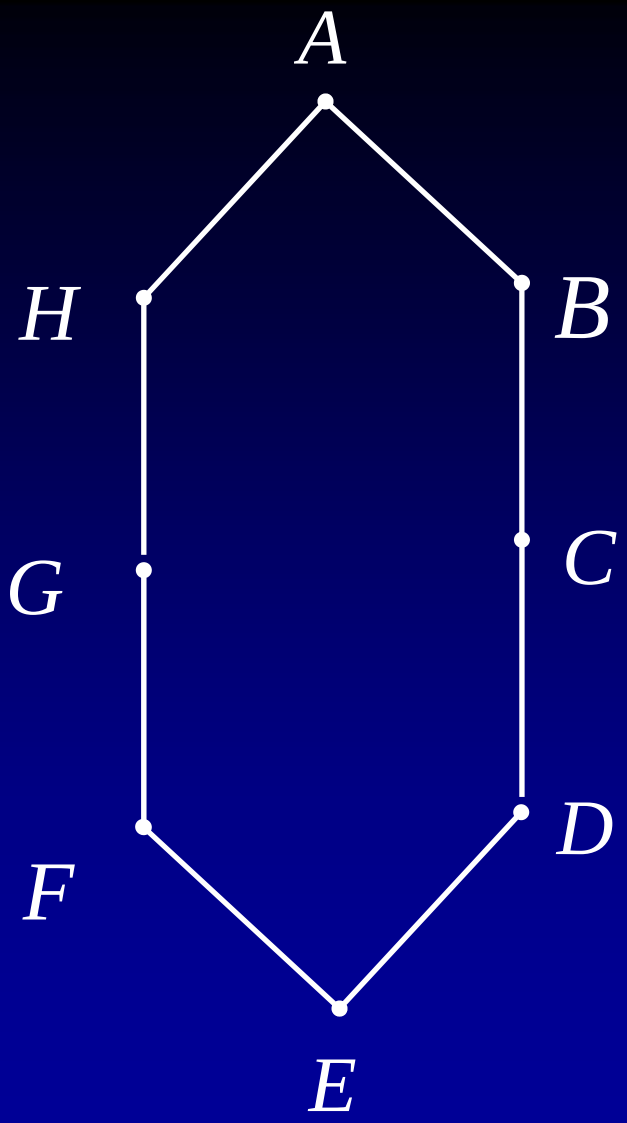


Figure 8.59

8.4 TRANSPORT NETWORKS

A **transport network** (运输网络), or **network**, is a connected digraph N with the following properties:

(a) There is a unique node, the **source**, that has in-degree 0, we generally label the source node 1.

(b) There is a unique node, the **sink**, that has out-degree 0. If N has n nodes, we generally label the sink as node n.

(c) The graph N is labeled. The label, C_{ij}, on arc (i, j) is a nonnegative number called the **capacity** (容量) of the arc.

Flows（流）

A **flow** in a network *N* is a function *F* that assigns to each arc (i, j) of *N* a nonnegative number F_{ij} that does not exceed C_{ij}, i.e., $0 \leqslant F_{ij} \leqslant C_{ij}$. We call F_{ij} as the flow through the arc (i, j). We also require that for each node other than the source and sink, the sum of the F_{ik} on arcs entering node k must be equal to the sum of the F_{kj} on arcs leaving node k. This is called **conservation**（守恒）**of flow**.

A consequence of this requirement is that the sum of the flows leaving the source must be equal to the sum of the flows entering the sink. This sum is called the **value of the flow** *F*, written value(*F*). We can represent a flow *F* by labeling each arc (i,j) with the pair (C_{ij}, F_{ij}).

A flow *F* in the network *N* represented by Figure 8.70 is shown in Figure 8.71.

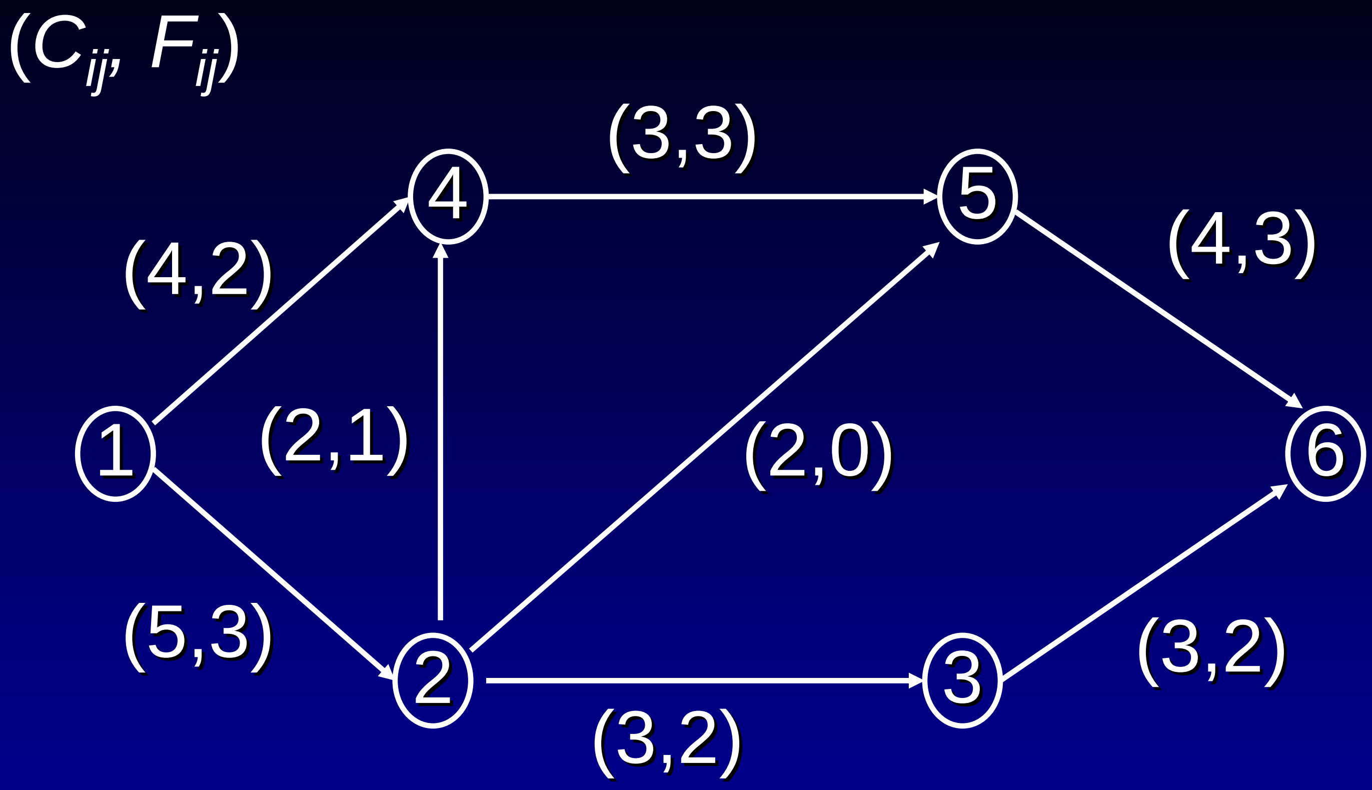
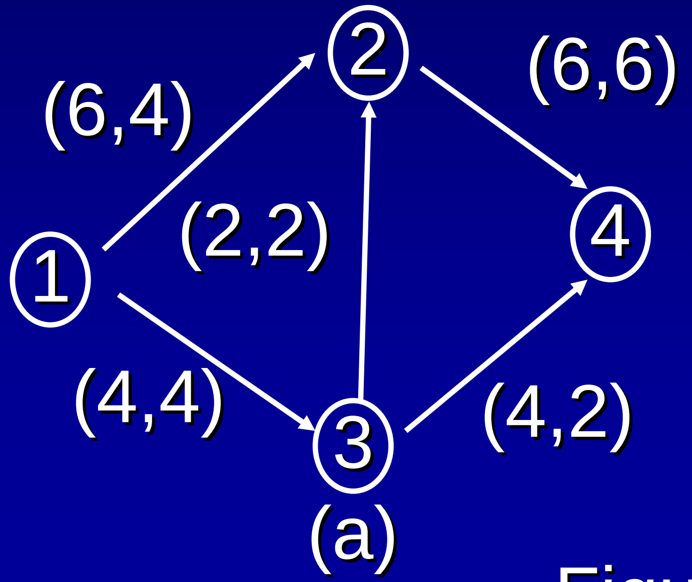


Figure 8.71

Maximum Flows

For any network， an important problem is to determine the maximum value of a flow through the network and to describe a flow that has the maximum value.

EXAMPLE 2



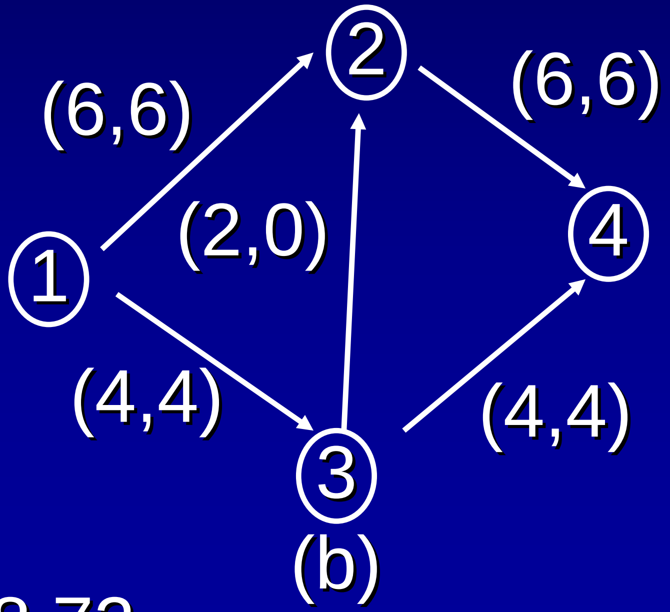


Figure 8.72

We now describe this improvement in general terms. Let N be a network and let G be the symmetric closure of N. Choose a path in G and an edge (i, j) in this path. If (i, j) belongs to N, then we say this arc has positive excess capacity（通过容量）if $e_{ij} = C_{ij} - F_{ij} > 0$. If (i, j) is not an arc of N, then we are traveling this arc in the wrong direction. In this case we say (i, j) has excess capacity $e_{ij} = F_{ji}$ if $F_{ji} > 0$. Then increasing flow through arc (i, j) will have the effect of reducing F_{ji}.

A Maximum Flow Algorithm

The algorithm due to Ford and Fulkerson is often called the **labeling algorithm**（标号算法）.

Algorithm (THE LABELING ALGORITHM)

Step 1 Let N_1 be the set of all nodes connected to the source by an arc with positive excess capacity. Label each j in N_1 with $[E_j, 1]$, where E_j is the excess capacity e_{1j} of arc $(1, j)$. The 1 in the label indicates that j is connected to the source, node 1.

Step 2 Let node j in N_1 be the node with smallest node number and let $N_2(j)$ be the set of all unlabeled nodes, other than the source, that are joined to node j and have positive excess capacity. Suppose that node k is in $N_2(j)$ and (j,k) is the arc with positive excess capacity. Label node k with $[E_k, j]$, where E_k is the minimum of E_j and the excess capacity e_{jk} of arc (j,k) . When all the nodes in $N_2(j)$ are labeled in this way, repeat this process for the other nodes in N_1. Let $N_2 = \cup_{j\in N_1} N_2(j)$

Step 3 Repeat Step 2, labeling all previously unlabeled nodes N_3 that can be reached from a node in N_2 by an arc having positive excess capacity. Continue this process forming sets $N_4, N_5, \cdots$ until after a finite number of steps either

(i) the sink has not been labeled and no other nodes can be labeled. It can happen that no nodes have been labeled; or

(ii) the sink has been labeled.

Step 4 In case (i), the algorithm terminates and then the total flow is a maximum flow.

Step 5 In case (ii) the sink, node n , has been labeled with $[E_n, m]$, where E_n is the amount extra flow that can be made to reach the sink through a path π . We examine π in reverse order. If arcs $(i, j) \in N$, then we increase the flow in (i, j) by E_n and decrease the excess capacity e_{ij} by the same amount. Simultaneously, we increase the excess capacity of the (virtual) arc (j, i) by E_n since there is that much more flow in (i, j) to reverse. If, on the other hand, $(i, j) \notin N$, we decrease the flow in (j, i)

by E_n and increase its excess capacity by E_n. We simultaneously decrease the excess capacity in (i, j) by the same amount, since there is less flow in (i, j) to reverse. We now have a new flow that is E_n units greater than before and we return to the step 1.

EXAMPLE 3

Use the labeling algorithm to find a maximum flow for the network in Figure 8.70.

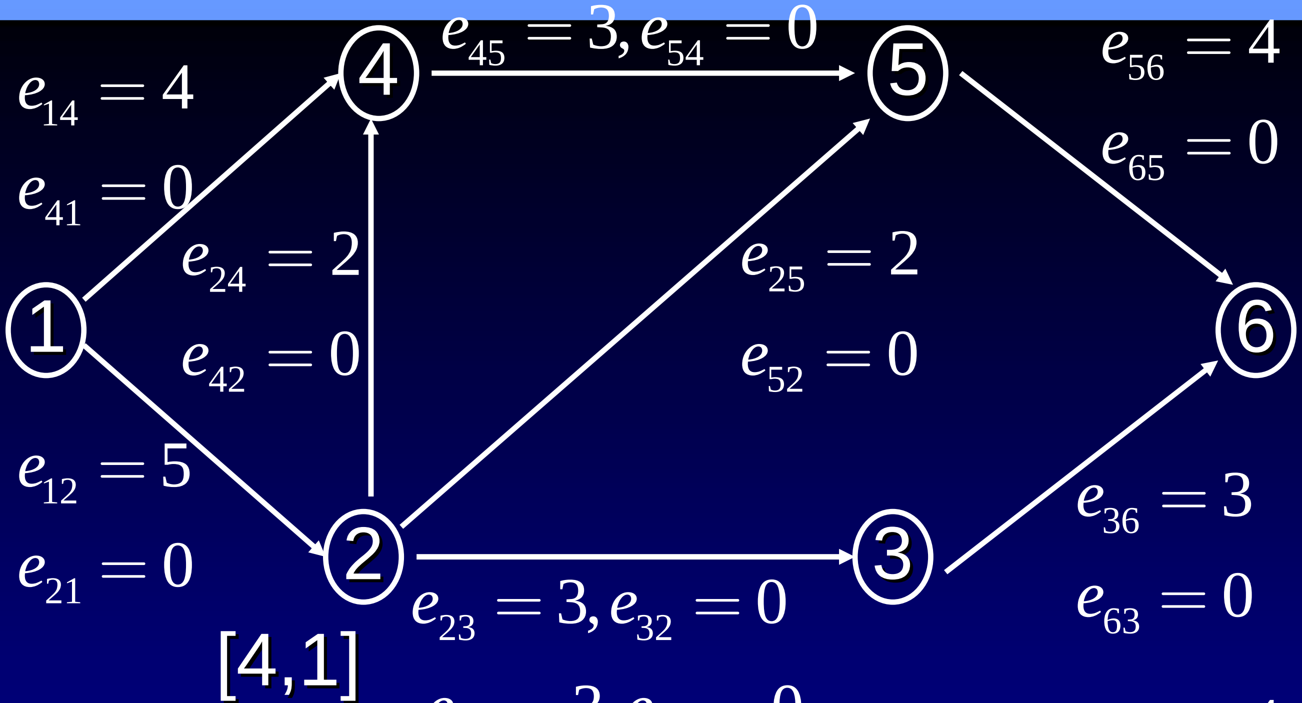


Figure 8.74

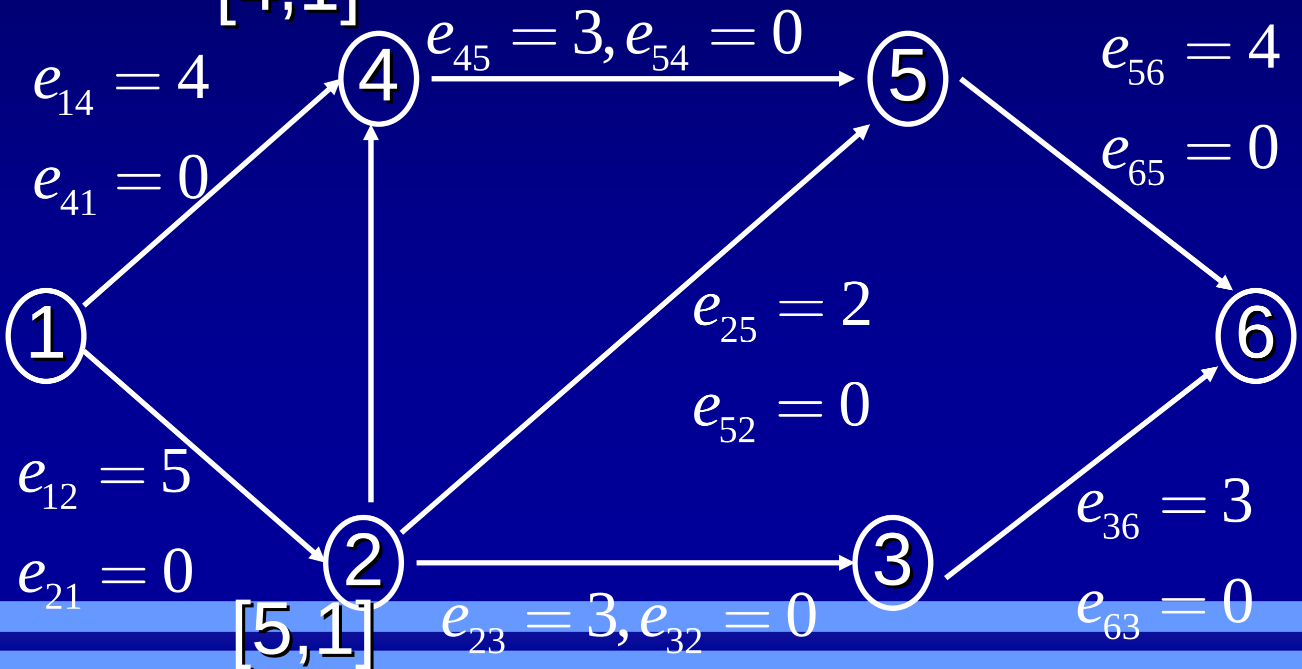


Figure 8.75

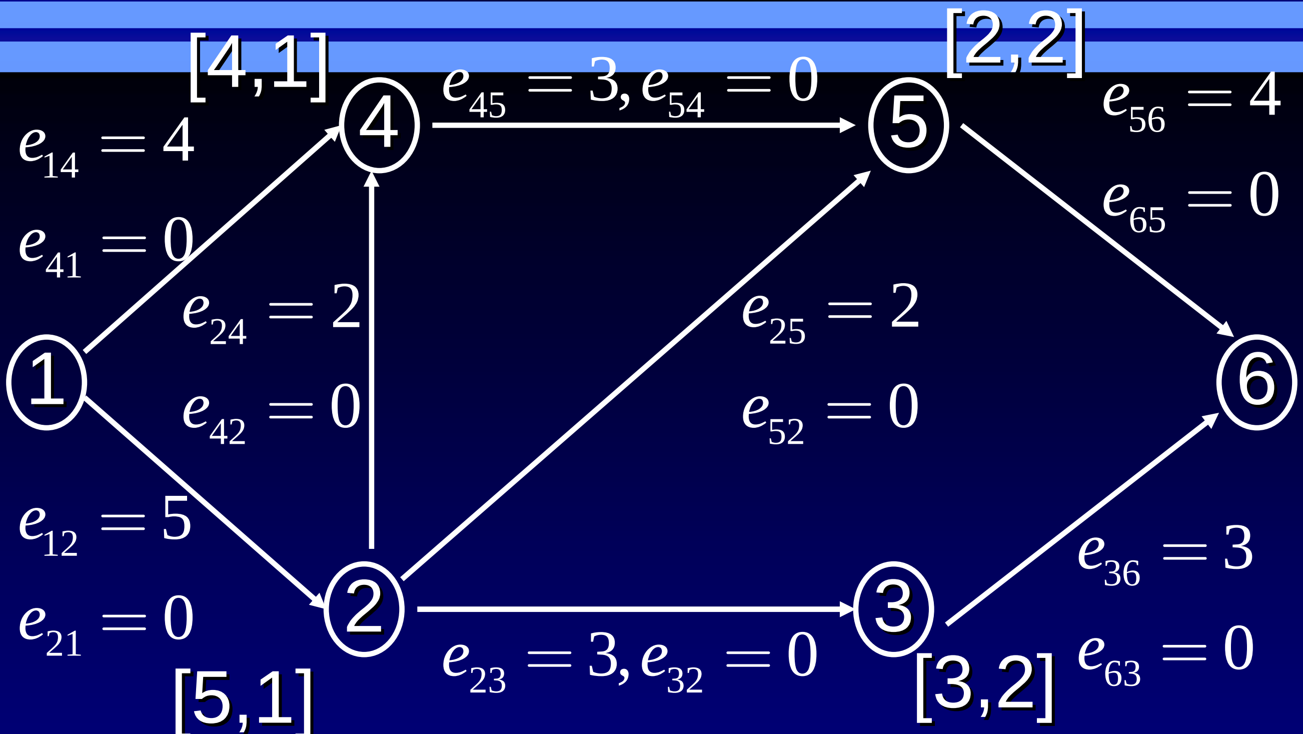


Figure 8.76

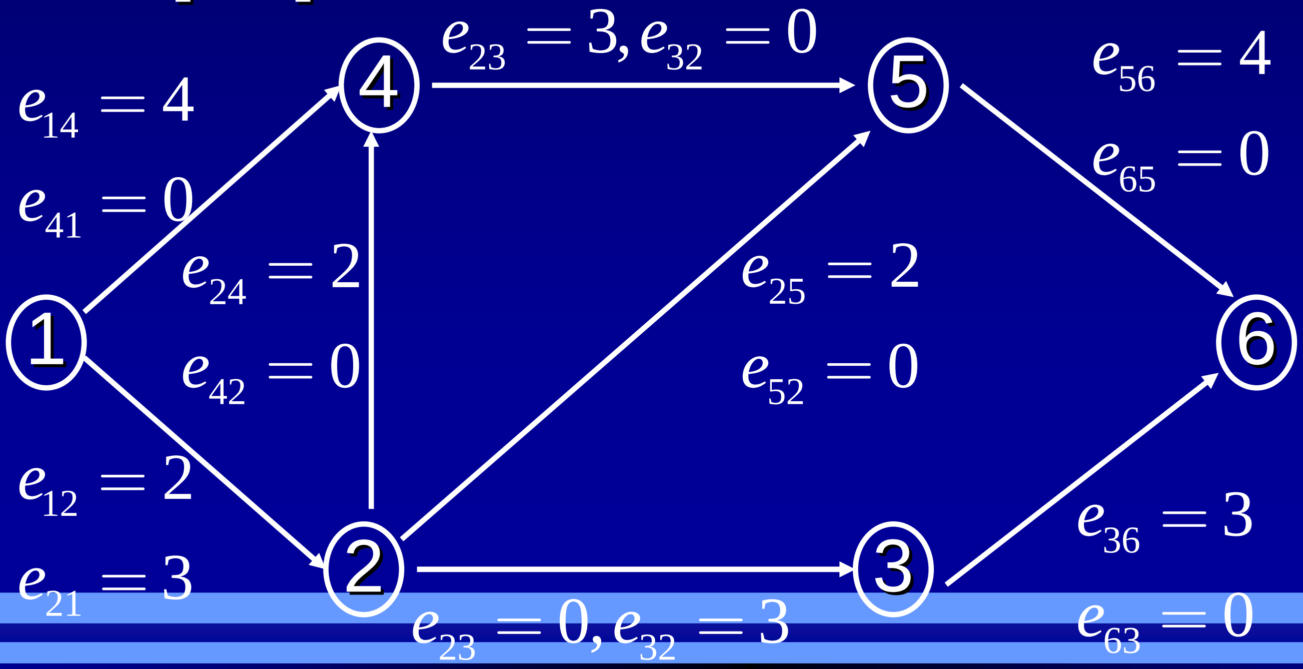


Figure 8.77

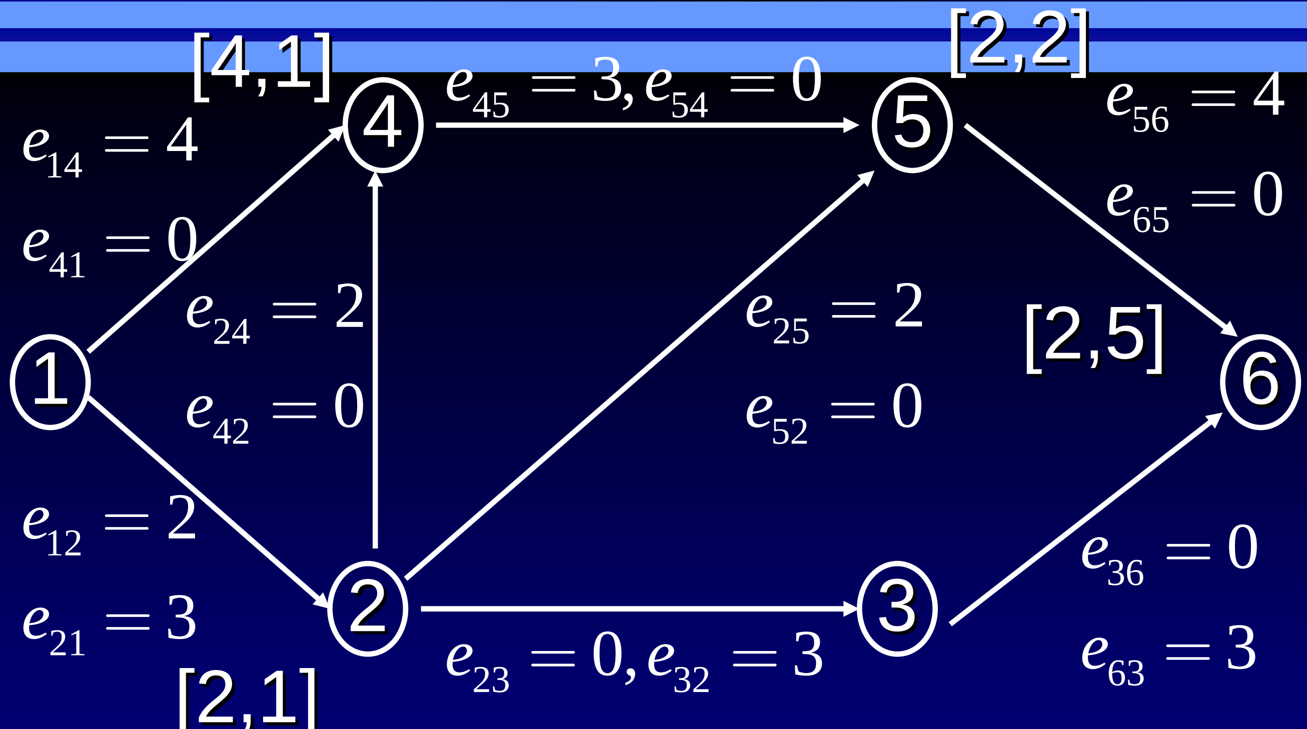


Figure 8.78

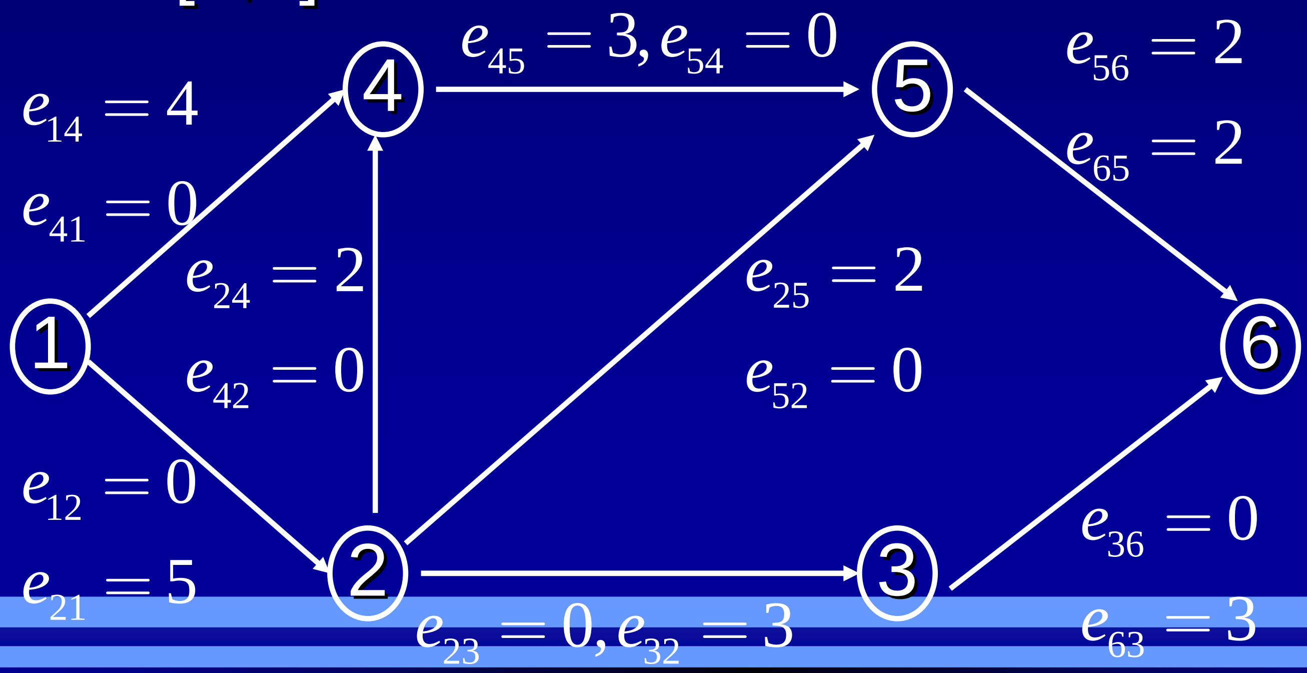


Figure 8.79

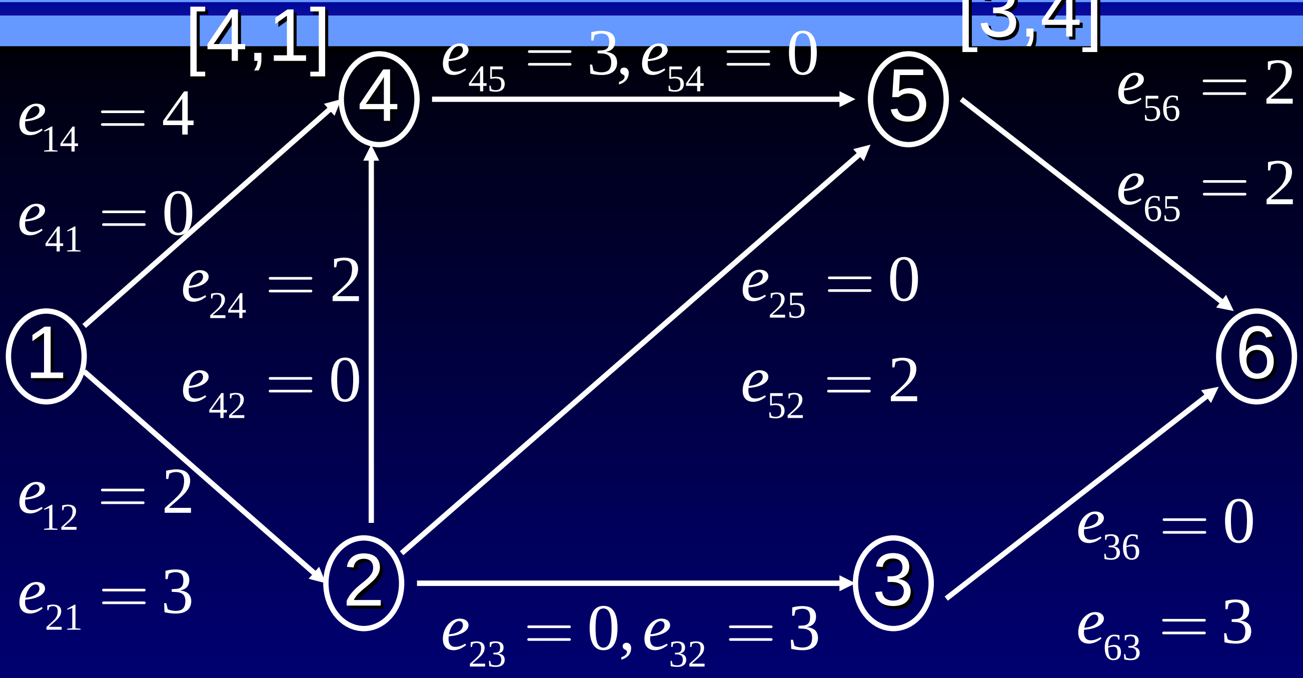


Figure 8.80

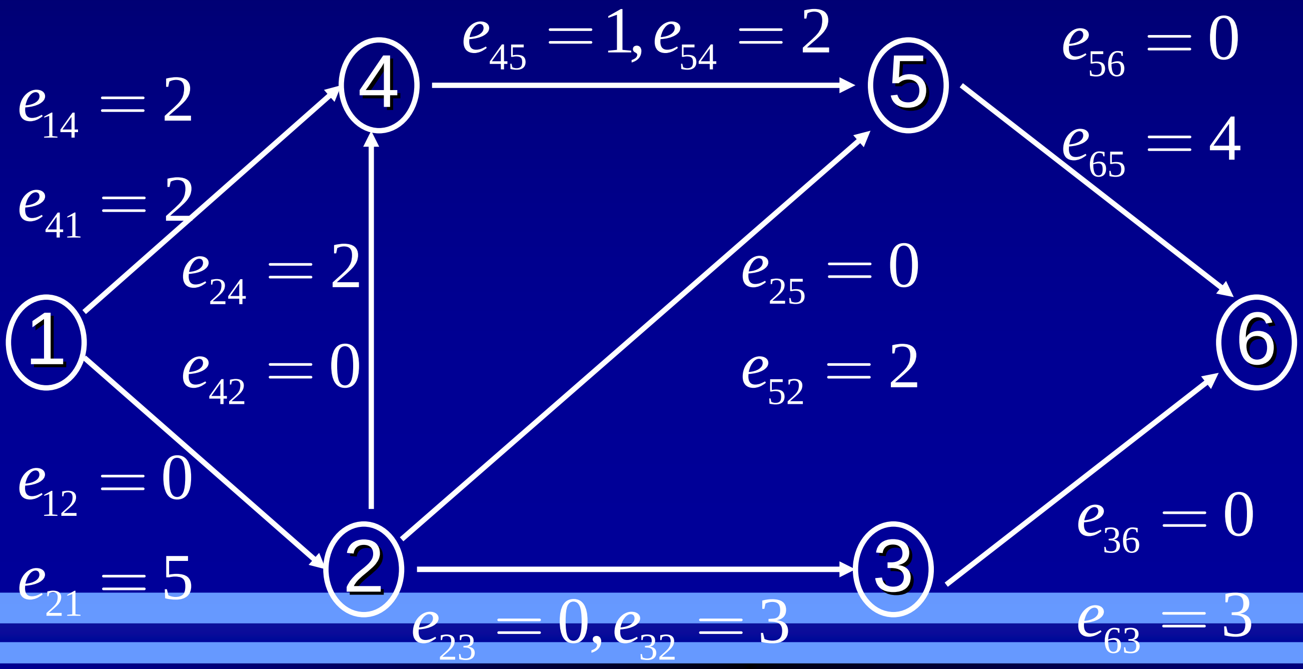


Figure 8.81

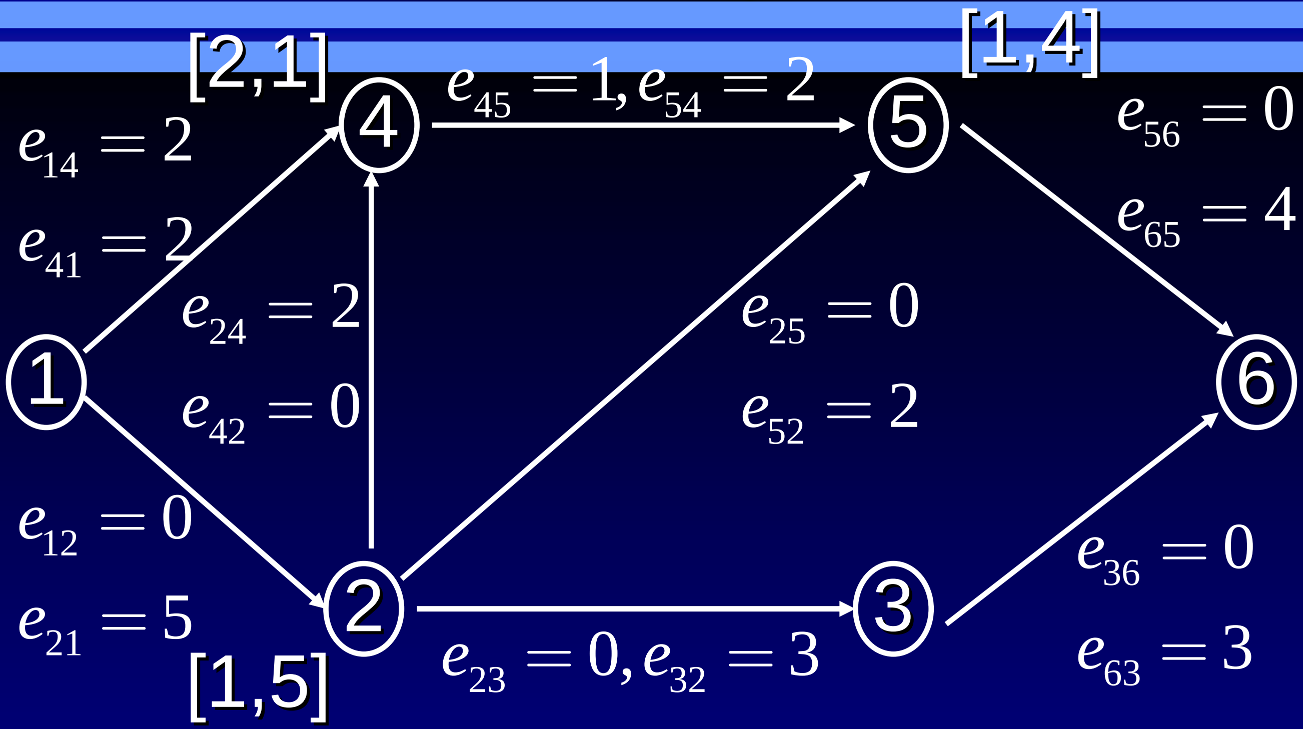


Figure 8.82

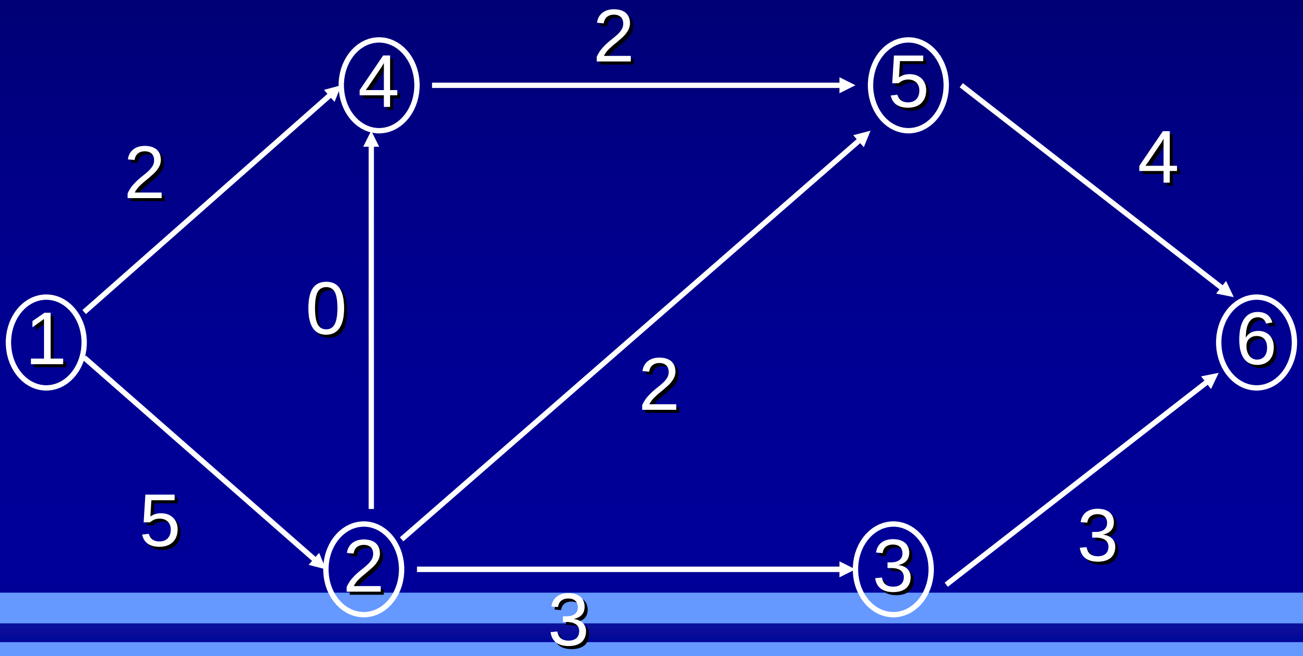


Figure 8.83

We define a **cut** in a network *N* as a set *K* of arcs having the property that every path from the source to the sink contains at least one arc from K. A cut does "cut" a digraph into two pieces, one containing the source and one containing the sink.

The **capacity of a cut** K,(截集K的截量) , $c(K)$ is the sum of the capacity of all arcs in K.

EXAMPLE 4

Figure 8.84 shows two cuts for the network given by Figure 8.70. Verify that $c(K_1) = 9$, and $c(K_2) = 7$.

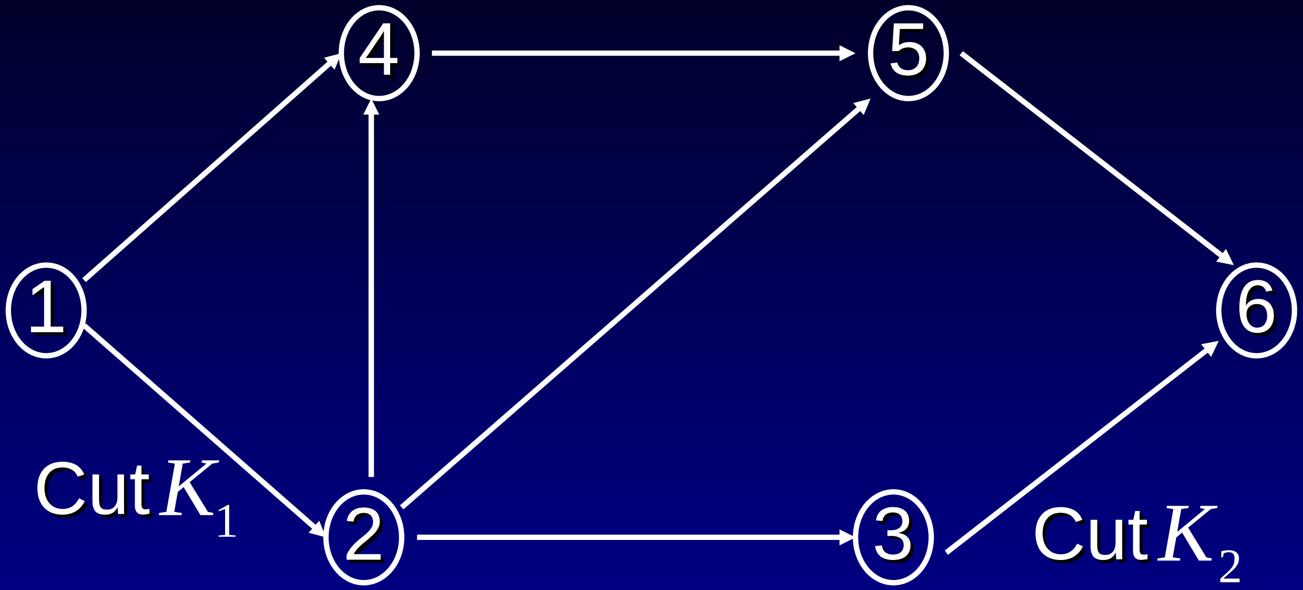


Figure 8.84

If F is any flow and K is any cut, then $value(F) \leq c(K)$. Suppose for some flow F and some cut K, $value(F) = c(K)$. Then F would be a flow with maximum value, K must be a minimum capacity cut.

Theorem 1 (The Max Flow Min Cut Theorem)

A maximum flow F in a network has value equal to the capacity of a minimum cut of the network.

Suppose that the algorithm has been run and has stopped at the step 4. Then the sink has not been labeled. Divide the nodes into two sets, M_1 and M_2, where M_1 contains the source and all nodes that have been labeled, and M_2 contains all unlabeled nodes. Let K consists of all arcs of the network N that connect a node in M_1 with a node in M_2. If i is the last node in π that belongs to M_1 and j is the node that follows i in the path, then j belongs to M_2 so (i, j) is in K.

Suppose that (i, j) is an arc in K, so that $i \in M_1$ and $j \in M_2$. The final flow F produced by the algorithm must result in (i, j) carrying its full capacity. Thus the value of the final flow of the algorithm is equal to the capacity $c(K)$, and so F is a maximum flow.

- Minimum Cost Flows

  Given a digraph N=(V,A;s,t;C,P) and a positive number k, where capacity $C:A\rightarrow R^+$, cost $P:A\rightarrow R^+$, find a flow $F:A \rightarrow R_0^+$ such that its flow value value(F)=k and its cost $P(F)=\Sigma_{e\in A}F(e)P(e)$ is minimum.

8.5 MATCHING PROBLEMS (匹配问题)

The first example is to allow a network to have multiple sources or multiple sinks as many real network do.

We want to maximize the flow from all sources taken together to all the sinks taken together. To find the maximum flow in a general network N, we change the problem into a single-source, single-sink network problem by enlarging N to N' as follows. We add two nodes, node a is the source for N' and is connected to all nodes that are sources in N, node b is the sink for N', all nodes that were sinks in N are connected to it. Nodes a and b are called, respectively, a **supersource** (超级发点) and a **supersink** (超级收点).

By adding a supersource and a supersink to a network, we can apply the labeling algorithm to find a maximum flow for the enlarged network. This flow will also be maximal for the original network.

EXAMPLE 2 Find the maximum flow for the network N given in Figure 8.92.

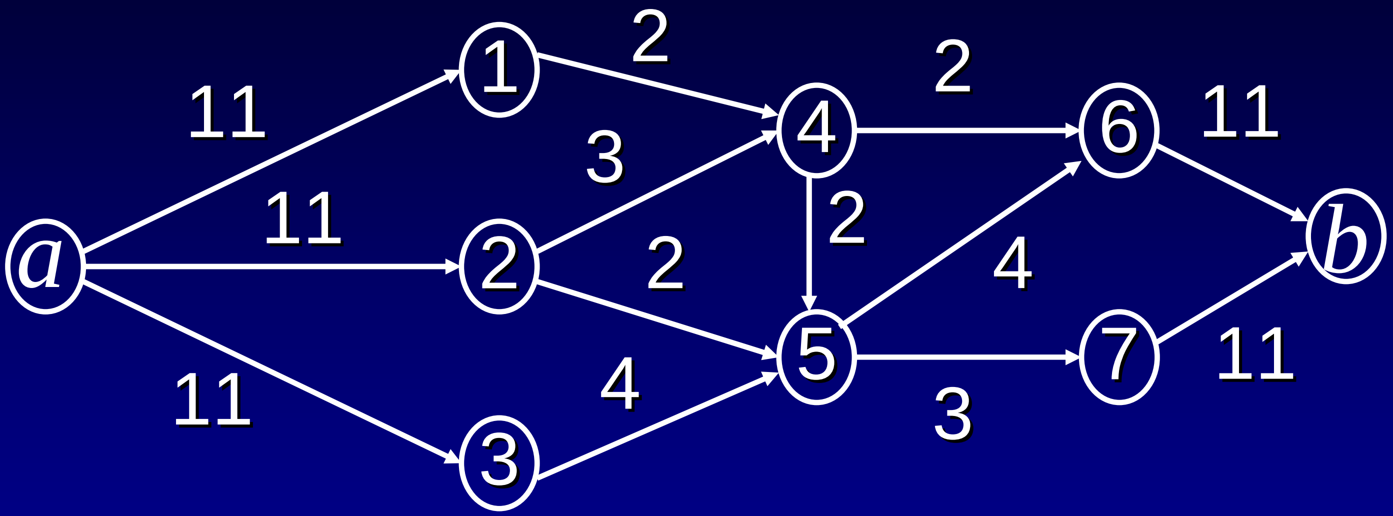


Figure 8.93

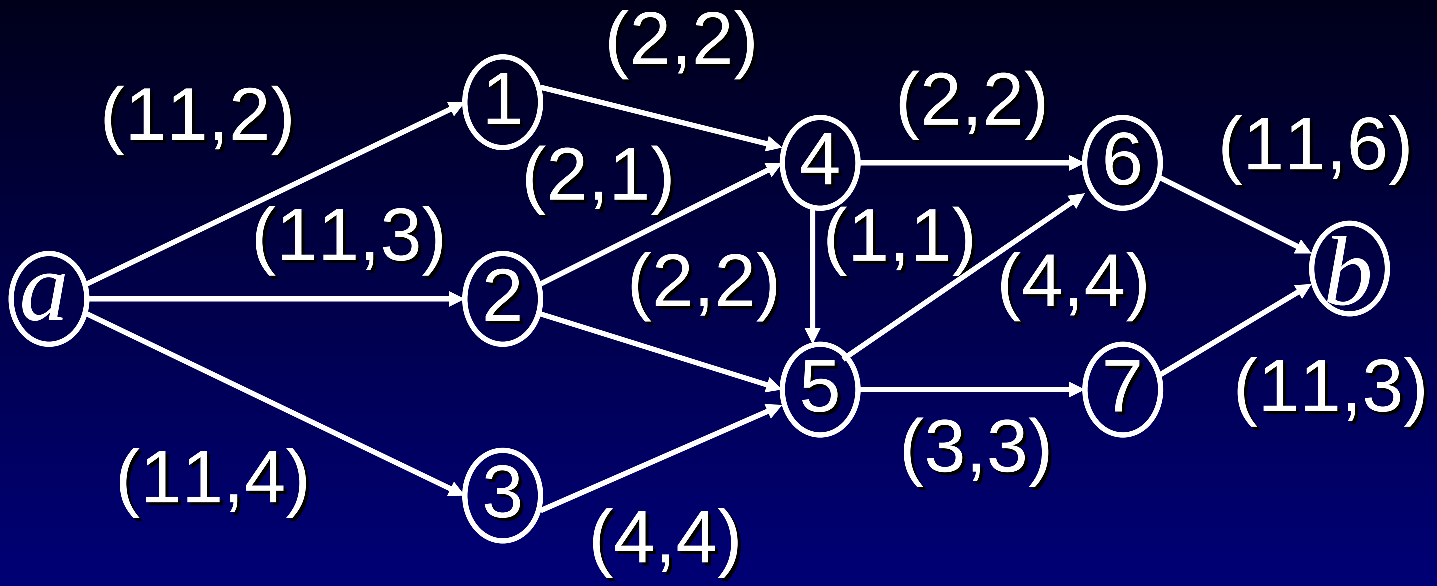


Figure 8.94

- The Matching Problem (匹配问题)

We begin with two finite sets A and B a relation R from A to B. A **matching function** M is one-to-one function from a subset of A to a subset of B. We say a is matched with b if $M(a) = b$. A matching function M is **compatible**（相容的）**with** R if $M \subseteq R$; that is, if $M(a) = b$, then $a\,R\,b$.

EXAMPLE 4

Let $A = \{s_1, s_2, s_3, s_4, s_5\}$ be a set of students working on a research project and $B = \{b_1, b_2, b_3, b_4, b_5\}$ be a set of reference books on reserve in the library for the project. Define R by $s_i\ R\ b_k$ if and only if student s_i wants to sign out book b_k. A matching of students to books would be compatible with R if each student is matched with a book that he or she wants to sign out.

Given any relation R from A to B, a matching M that is compatible with R is called **maximum** if its domain if as large as possible and is **perfect** if its domain is A. Matching problems can be solved using networks. We create a network to model the situation by using the elements of A as sources and the elements of B as sinks. There is an directed edge or arc (a,b) if and only if aRb. Each arc is assigned capacity 1.

EXAMPLE 5

Let A, B, and R be as in Example 4. Suppose student s_1 wants books b_2 and b_3; s_2 wants b_1, b_2, b_3, b_4, s_3 wants b_2, b_3; s_4 wants b_2, b_3, b_4; s_5 wants b_2, b_3. Then the network N that represents this situation is given in Figure 8.95.

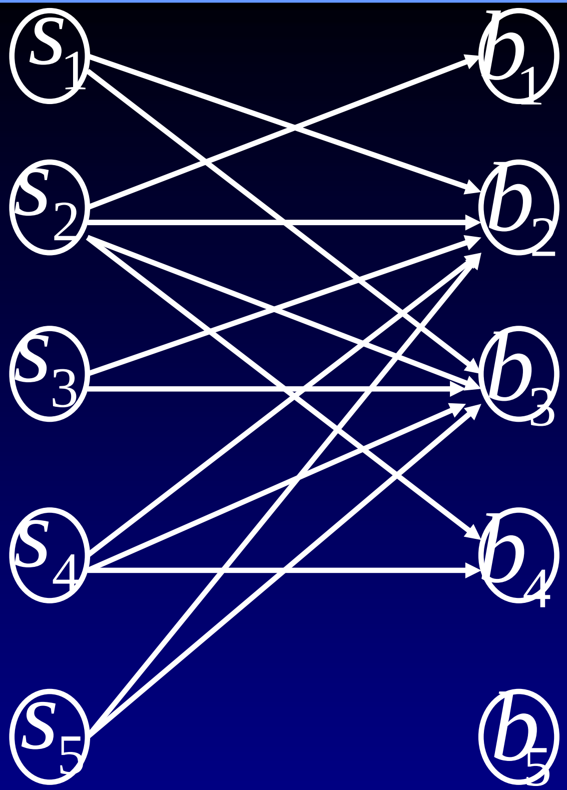


Figure 8.95

In Figure 8.95, a supersource x and a supersink y have been provided and new arcs have been assigned capacity 1 to create N'.

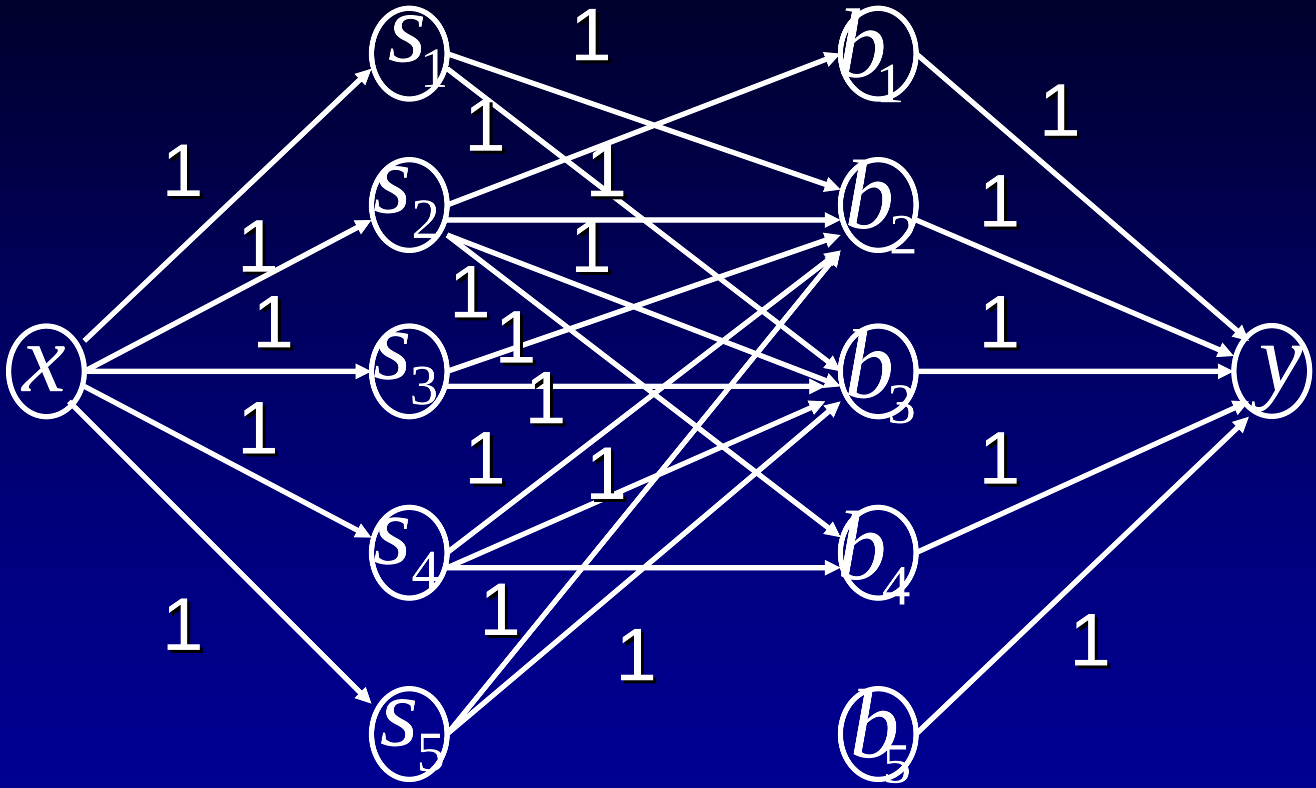


Figure 8.96

Suppose that F is a flow in N. If the flow into node s_m is 1, then the flow out must be 1, so the flow from s_m(if any) can go to only one node, say b_n. Flow can enter a node b_n from at most one s_m since the flow out of b_n is 1. We match s_m to b_n if and only if there is flow between these two nodes. The matching function M that we construct is clearly compatible with R.

If we have an R-compatible matching M, we can define a flow F by letting the flow be 1 between any two matched nodes, 1 from x to each student matched with a book, and from each matched book to y , and 0 on all other edges. This flow yields the matching M again.

Theorem 1 (Hall's Marriage Theorem)

Let R be a relation from A to B. Then there exists a complete matching M if and only if for each $X \subseteq A,\ |X| \le |R(X)|$.

Proof If a complete matching M exists, then $M(X) \subseteq R(X)$ for every subset X of A. But M is one to one, so $|X| = |M(X)| \le |R(X)|$.

Conversely, suppose that for any $X \subseteq A, |X| \le |R(X)|$

construct the network N that corresponds to R. Suppose $|A| = n$. We want to show that there is a flow in N with value n, which will correspond to a complete matching. We know by the Max-Flow-Min-Cut theorem that it is sufficient to show that the minimum cut in N has value n. A typical situation is shown in Figure 8.97, with the wavy line representing a cut in N.

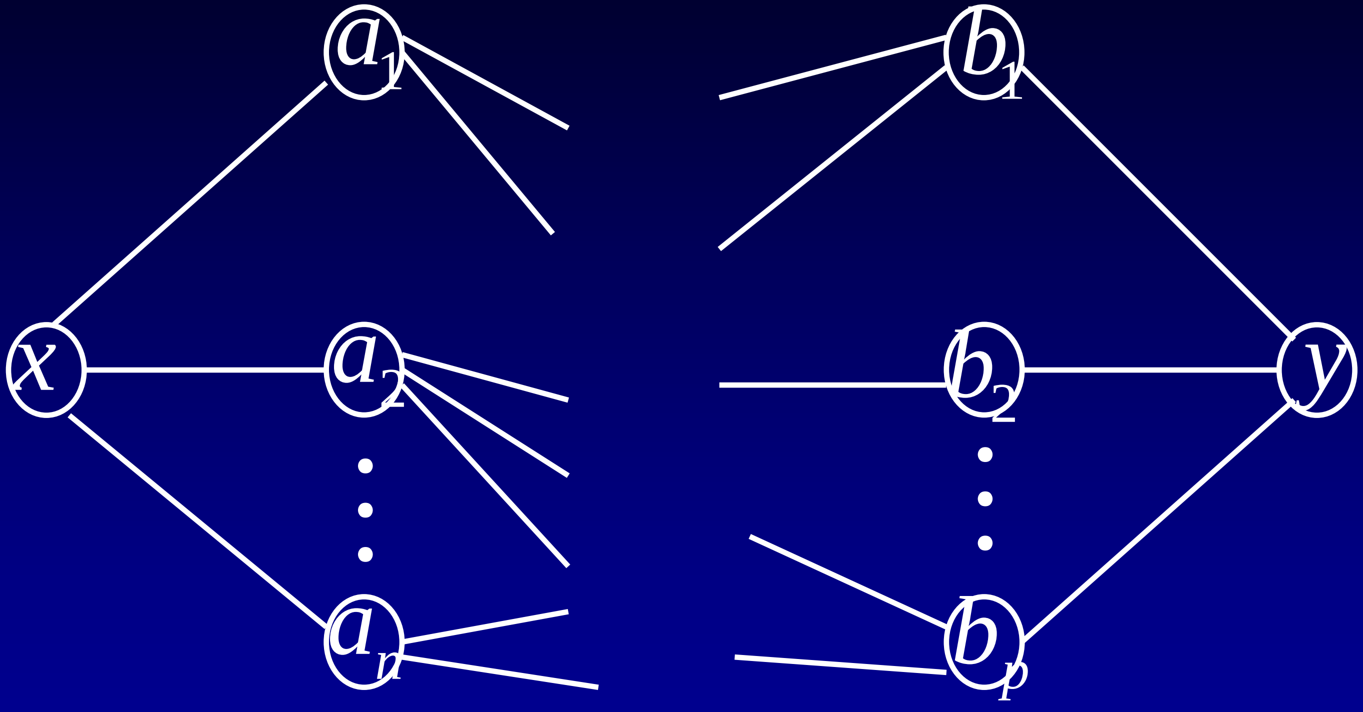


Figure 8.97

Note that this is an example of an existence theorem; no method is given for finding a complete matching, if one exists.

- Hitchcock Problem

8.6 COLORING GRAPHS

Suppose that $G=(V,E;\gamma)$ is a graph with no multiple edges, and $C=\{c_1,c_2,\cdots,c_n\}$ is any set of n “colors”. Any function $f:V\to C$ is called a **coloring** of the graph G using n colors (or using the colors of C). For each vertex v, $f(v)$ is the color of v.

A coloring is **proper** if any two adjacent vertices v and w have different colors.

EXAMPLE 1

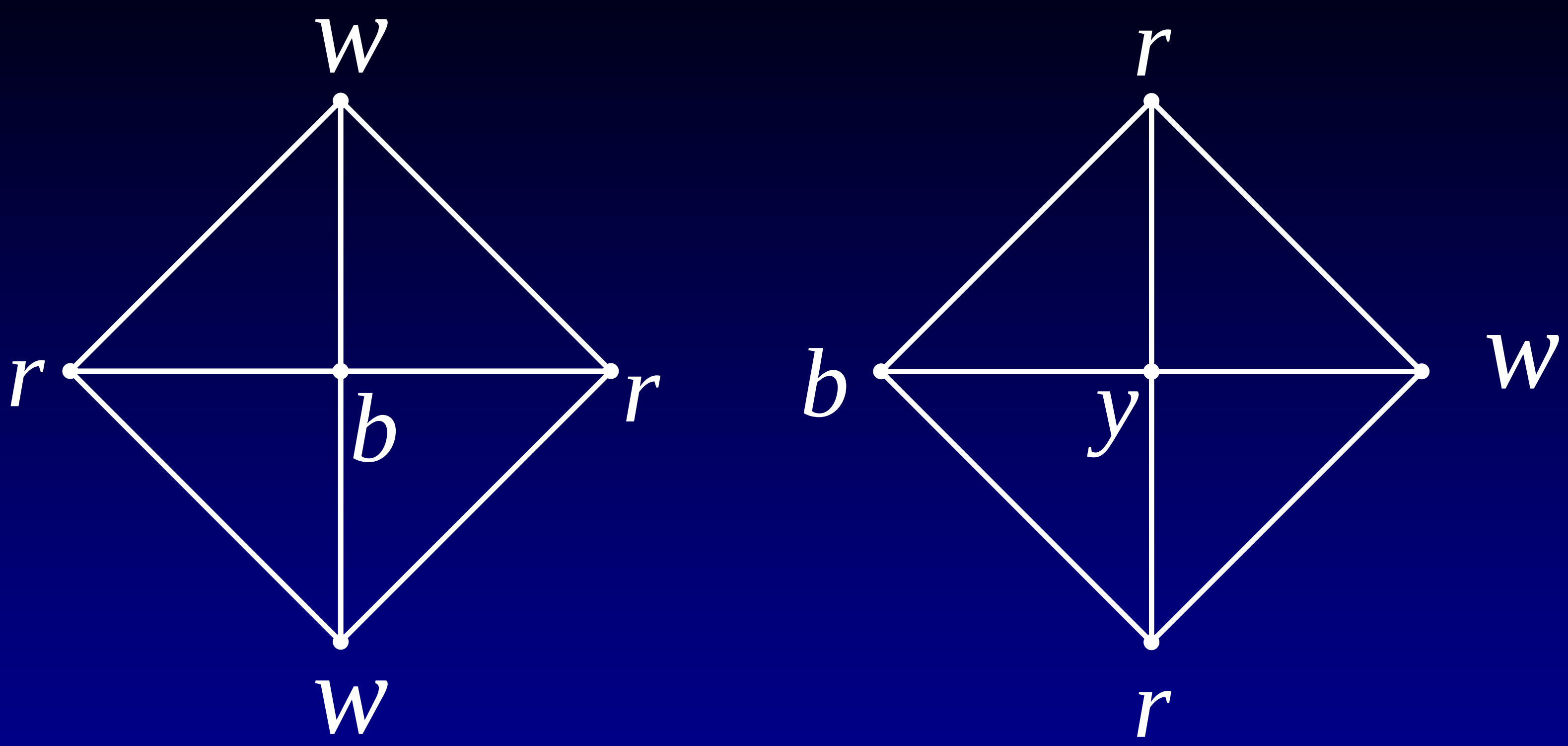


Figure 8.103

The smallest number of colors needed to produce a proper coloring of a graph G is called the **chromatic number**（染色数）of G , denoted by $\chi(G)$.

For the graph *G* of Figure 8.103, our discussion leads us to believe that $\chi(G) = 3$.

Of the many problems that can be viewed as graph-coloring problem, one of the oldest is the map-coloring problem. Consider the map shown in Figure 8.104.

A coloring of a map is a way to color each region (country, state, country, province, etc) so that no two distinct regions sharing a common border have the same color. The map-coloring problem is to find the smallest number of colors that can be used. We can view this problem as a proper graph-coloring problem as follows.

Given a map M, construct a graph G_M with one vertex for each region and an edge connecting any two vertices whose corresponding regions share a common boundary. Then the proper coloring of G_M correspond exactly to the colorings of M.

Figure 8.104

EXAMPLE 2

Consider the map M shown in Figure 8.104. then G_M is represented by Figure 8.105.

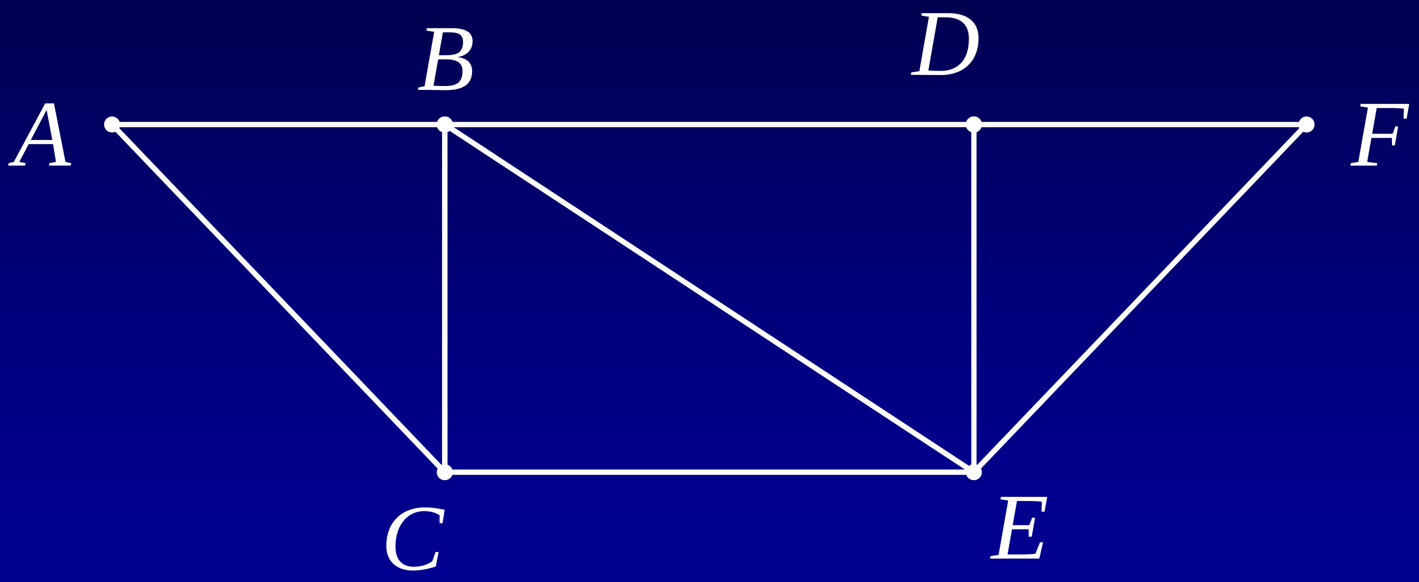


Figure 8.105

A conjecture was that **four colors** are always enough to color any map drawn on a plane. This conjecture was proved to be true in 1976 with the aid of computer computations performed on almost 2,000 configurations of graphs. There is still no **mathematical proof** known that does not depend on computer checking.

The graph corresponding to a map is an example of a **planar graph**, meaning that it can be drawn in a plane so that no edges cross except at vertices.

THE END!