

# Graph Theory

Rosen 8<sup>th</sup> ed., ch. 10

## **10.1 图的概念/Introduction of Graph**

## **10.2 图的术语/Graph Terminology**

## **10.3 图的表示与同构/**

### **Representing Graph and Graph Isomorphism**

## **10.4 连通性/Connectivity**

## **10.5 欧拉道路与哈密尔顿道路/**

### **Euler and Hamilton Paths**

## **10.6 最短道路问题/Shortest Path Problem**

## **10.7 平面图/Planar Graphs**

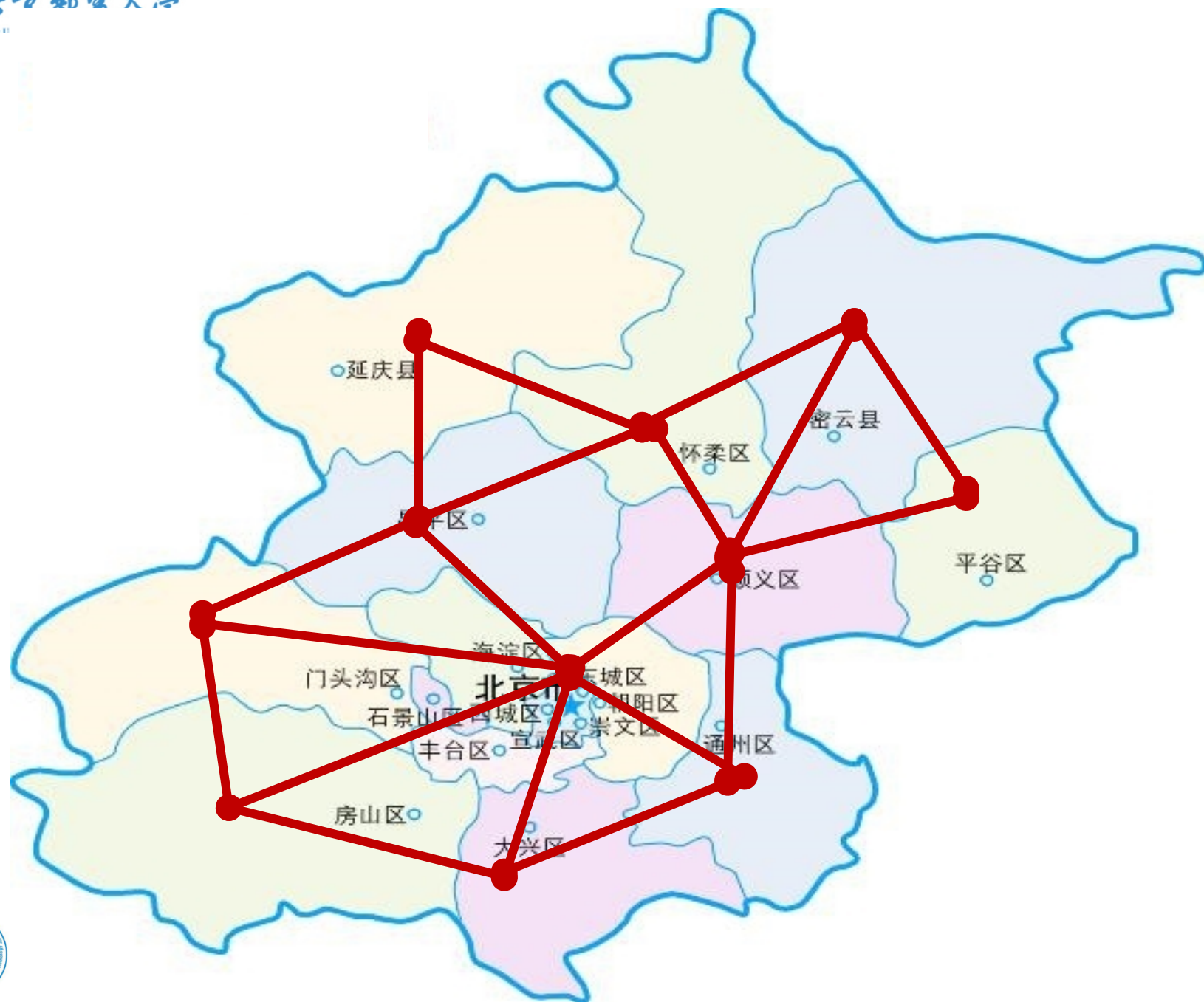
## **10.8 图的着色/Graph Coloring**

## **Transport networks传输网流量问题**

# Coloring graphs

- Chromatic number of  $G$
- Chromatic polynomial of  $G$





# Coloring graphs

- Suppose that  $G = (V, E, \gamma)$  is a graph with no multiple edges, and  $C = \{c_1, c_2, \dots, c_n\}$  is any set of  $n$  “colors”.
- Any function  $f: V \rightarrow C$  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  $v$ .

$$\chi(G)$$

- A coloring is *proper* if any two adjacent vertices  $v$  and  $u$  have different colors.
- The smallest number of colors needed to produce a proper coloring of a graph  $G$  is called the *chromatic number of  $G$*  ( $G$ 的着色数), denoted by  $\chi(G)$ .

点着色 (coloring the vertices)

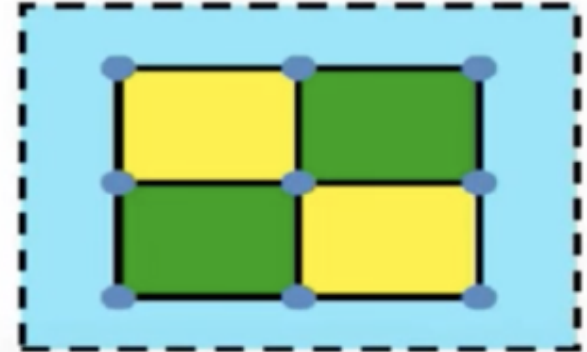
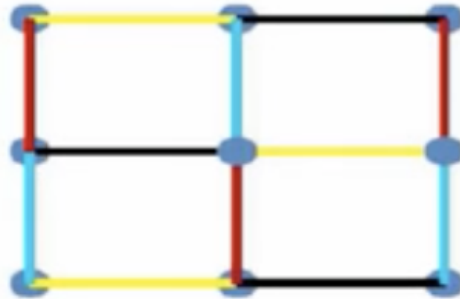
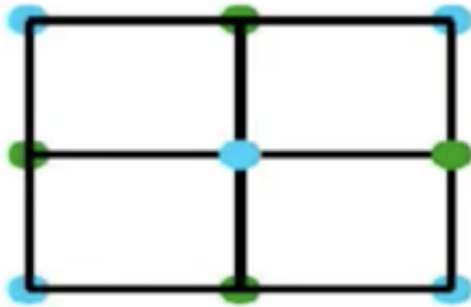
边着色 (coloring the edges)

面着色 (coloring the regions)



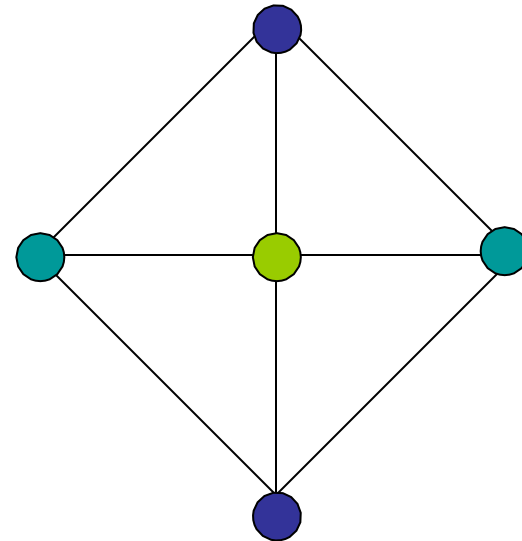
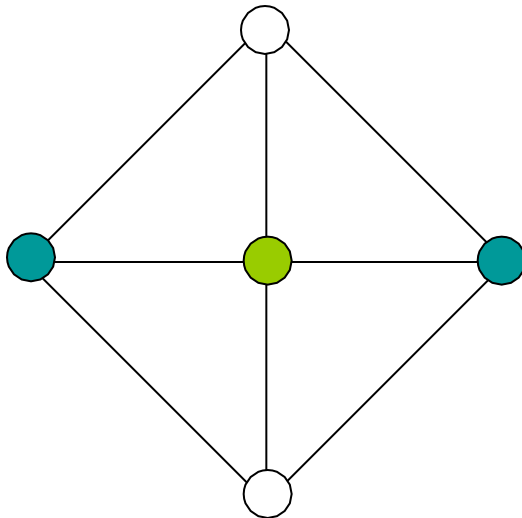
# Example

$$\chi(G)=2, \chi'(G)=4, \chi^*(G)=3$$

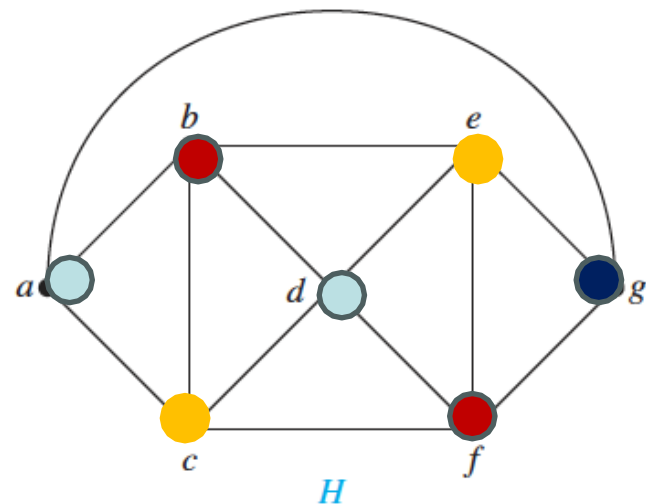
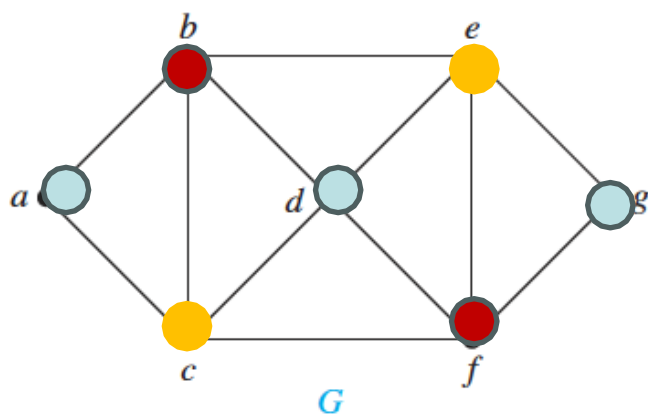


# Example 1

- $\chi(G) = 3$



# Example 4



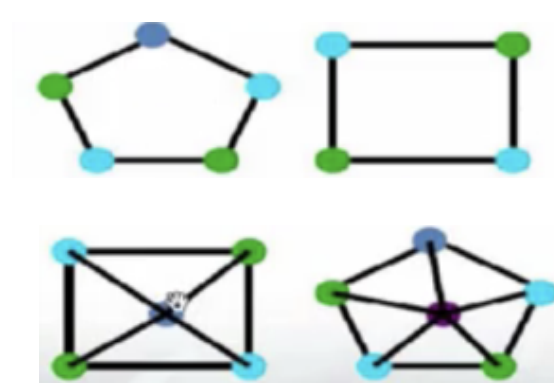
**FIGURE 3** The simple graphs  $G$  and  $H$ .

$$\chi(G) = 3$$

$$\chi(H) = 4$$

# 点着色数

- $\chi(G) = ? \iff G$  是零图(无边只有顶点)
- $\chi(K_n) = ?$
- $\chi(G) = ? \iff G$  是非零二部图
- $\chi(C_n) = ?$
- $\chi(W_n) = ?$



定理（补）：简单图的点着色数小于等于最大度加1.  $\chi(G) \leq \Delta(G) + 1$

# *four colors*

Conjecture - 四色猜想

- ***Four colors are always enough to color any map drawn on a plane***

**THE FOUR COLOR THEOREM** : The chromatic number of a planar graph is no greater than four.

1852年，弗南西斯.格思里，发现每幅地图都可以用四种颜色着色。

1852年，他的弟弟请教他的老师、数学家德.摩尔根

数学家哈密尔顿爵士请教。直到1865年哈密尔顿逝世为止，问题也没有能够解决。

1872年，英国当时数学家凯利正式向伦敦数学学会提出了这个问题，于是四色猜想成了世界数学界关注的问题

1878~1880年两年间，律师兼数学家肯普和泰勒两人分别提交了证明四色猜想的论文，宣布证明了四色定理

1890年，数学家赫伍德（利于四色定理的思想证明五色定理）以自己的精确计算指出肯普的证明是错误的。不久，泰勒的证明也被人们否定了

# Appel-Haken Four-Color Theorem [1976]

## 阿佩尔-黑肯四色定理

- The vertices of any planar graph can be 4-colored in such a way that no two adjacent vertices receive the same color.
- 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 proof known that does not depend on computer checking.

# Color Theorem

## 着色定理

- 五色定理：任何平面图可以用五种颜色着色。
- 六色定理：任何平面图可以用六种颜色着色。



# Lemma

- Every Planar Graph Contains a Node of Degree  $\leq 5$
- Proof
  - If every node has degree at least 6, then the number of edges would be  $3n$ , which would contradict our upper bound of  $3n-6$  edges in an  $n$ -node planar graph.

# 六色定理证明

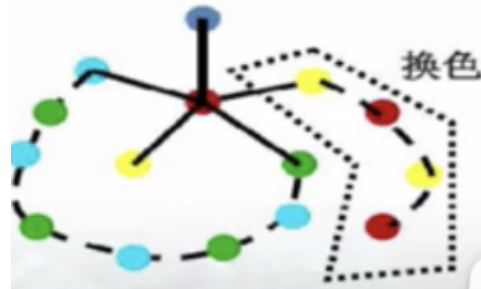
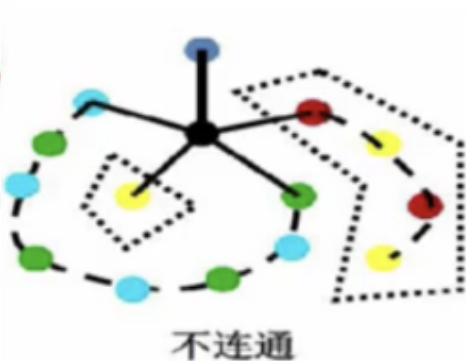
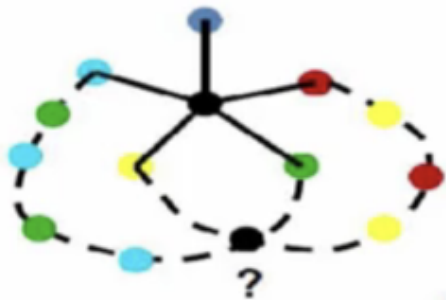
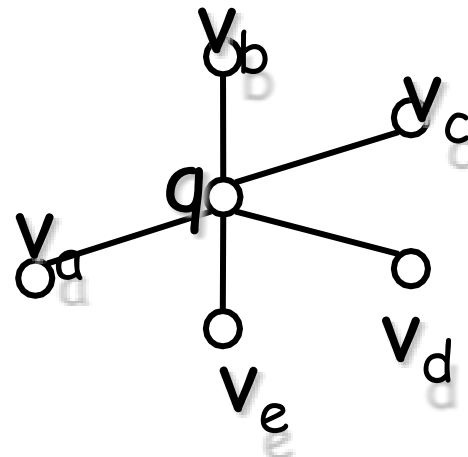
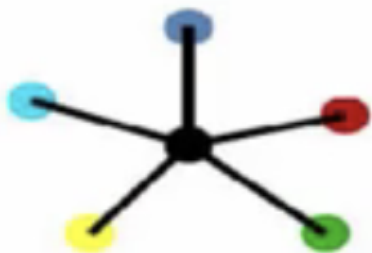
证明：用归纳法

1)  $n \leq 6$ 时，为真

2) 设 $n=k(\geq 7)$ ，为真。

- $n=k+1$  时，存在一个 $\leq 5$ 度的节点 $v$ ，即 $\deg(v) \leq 5$ 。
- 根据归纳假设图  $G' = G - v$  为真
- 把节点 $v$ 再加入原图， $v$ 最多与 $G'$ 中5个不同颜色的点连接，剩余的第6个颜色留个自己。

# 五色定理证明



利用引理1，换色证明

# Graph-coloring

- Graph-coloring problems also arise from counting problems.

# Example 3

- **Fifteen different foods** are to be held in refrigerated compartments within the same refrigerator.
- Construct a graph  $G$  as follows.
  - Construct one vertex for each food and connect two with an edge if they must be kept in separate compartments in the refrigerator.
  - Then  $\chi(G)$  is the smallest number of separate containers needed to store the 15 foods properly.

# Applications

**1 SCHEDULING FINAL EXAMS**

**2 FREQUENCY ASSIGNMENT**

# Another problem

## 着色的方法数

- Computing the total number of different proper colorings of a graph  $G$  using a set  $\{c_1, c_2, \dots, c_n\}$  of colors.

# Chromatic Polynomials

## 色多项式

- If  $G$  is a graph and  $k \geq 0$  is an integer, let  $P_G(k)$  be the number of ways to color  $G$  properly using  $k$  or fewer colors.
- $P_G$  is called the *chromatic polynomial* of  $G$ .



# Linear graph $L_4$

- Suppose we have  $x$  colors

- $P_{L_4} = x(x-1)^3$

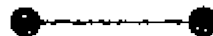
- $P_{L_4}(0) = 0$

- $P_{L_4}(1) = 0$

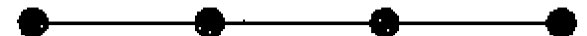
- $P_{L_4}(2) = 2$

- $P_{L_4}(3) = 24$

- So,  $\chi(L_4) = 2$



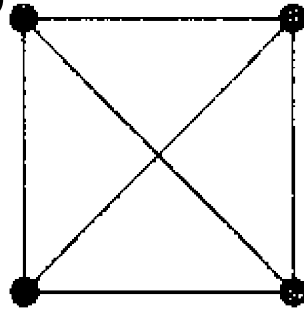
$L_2$



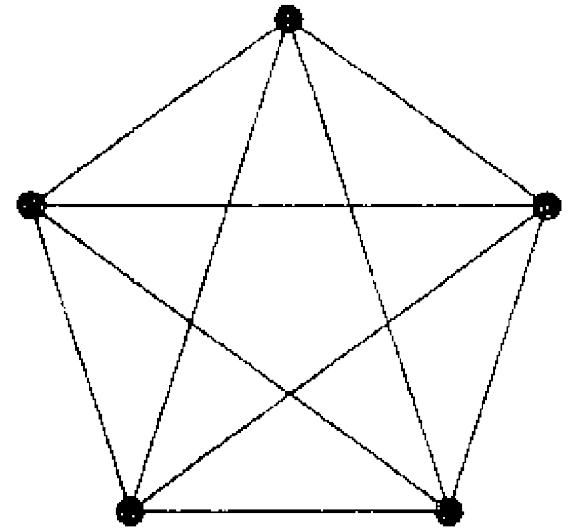
$L_4$

# Complete graph $K_n$

- Suppose we have  $x$  colors, if  $x < n$ , no proper coloring is possible.
- If  $x \geq n$  :
  - $P_{K_5}(x) = x(x-1)(x-2) \dots (x-n+1)$
  - $P_{K_5}(5) = 5!$
  - So,  $\chi(K_5) = 5$



$K_4$



$K_5$

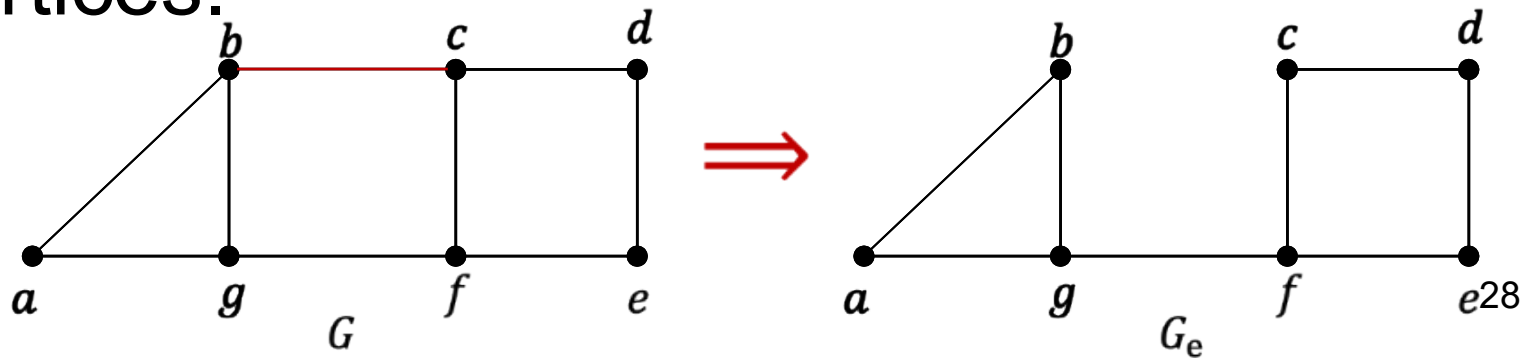
# Theorem 1

## 非联通图着色

- If  $G$  is a disconnected graph with components  $G_1, G_2, \dots, G_m$ , then  $P_G(x)$  is the product of the chromatic polynomials for each component
  - $P_G(x) = P_{G_1}(x) P_{G_2}(x) \dots P_{G_m}(x)$

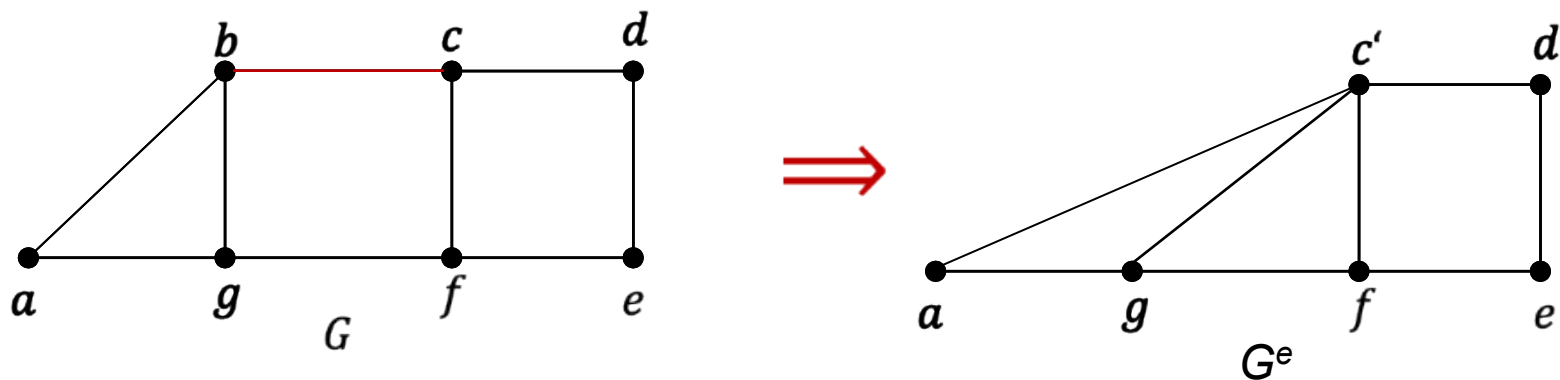
# Subgraph

- One of the most important subgraphs is the one that arises by deleting one edge and no vertices.
- If  $G = (V, E, \gamma)$  is a graph and  $e \in E$ , then we denote by  $G_e$  the subgraph obtained by omitting the edge  $e$  from  $E$  and keeping all vertices.



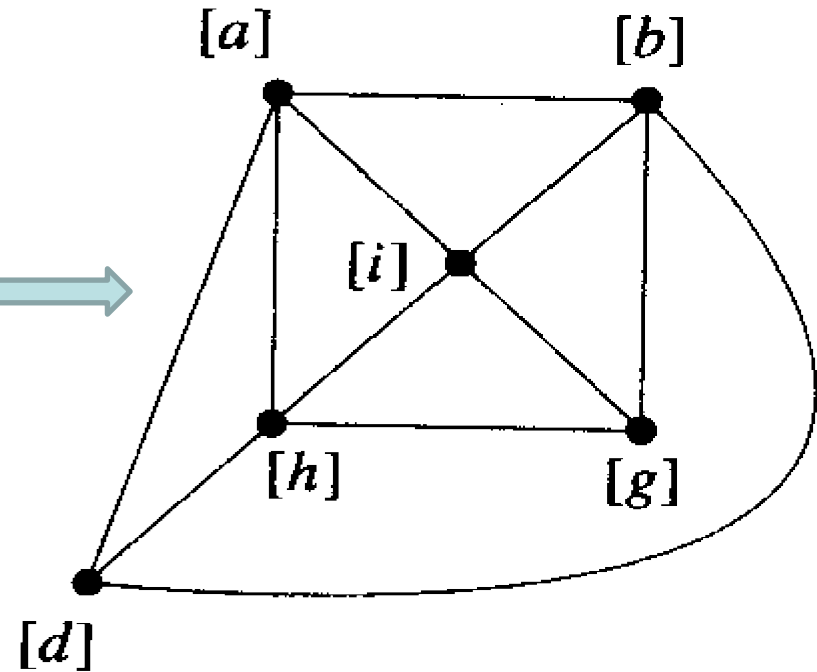
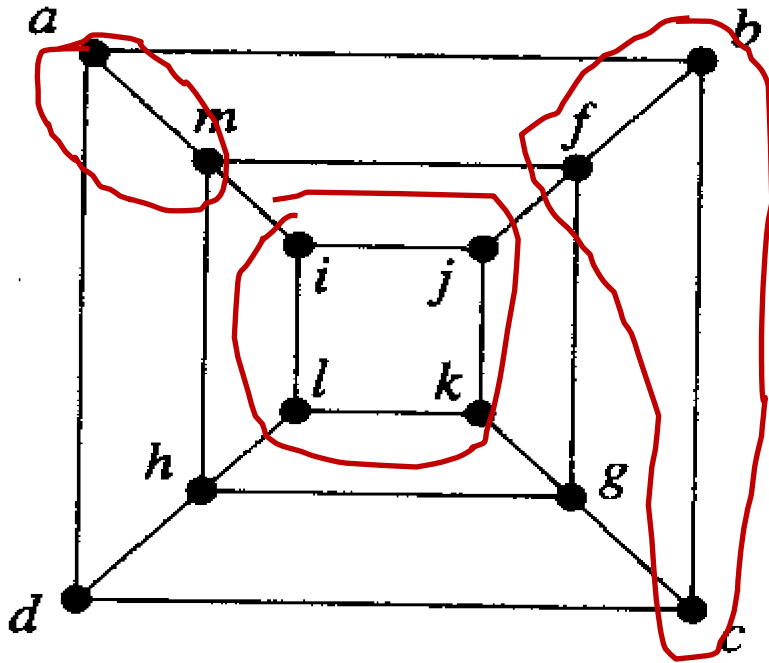
# Quotient graph (商图)

- $G^e$ : the quotient graph of  $G$  obtained by merging the end points of  $e$



# Example

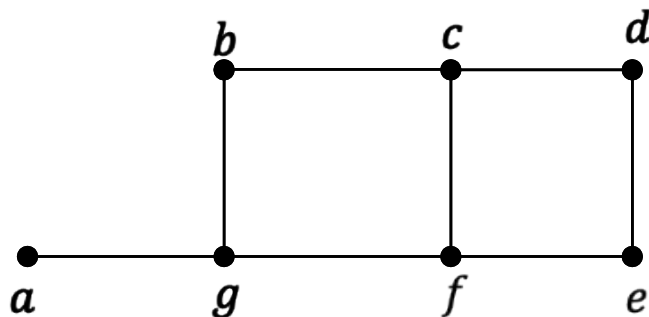
- $R = \{\{i,j,k,l\}, \{a,m\}, \{f,b,c\}, \{d\}, \{g\}, \{h\}\}$



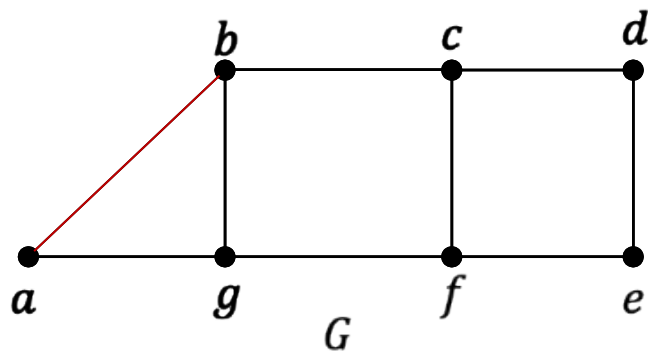
# Theorem 2

- Let  $G = (V, E, \gamma)$  be a graph with no multiple edges, and let  $e \in E$ , say  $e = \{a, b\}$ .
  - $G_e$  : subgraph of  $G$  obtained by deleting  $e$ ,
  - $G^e$  : the quotient graph of  $G$  obtained by merging the end points of  $e$ .
- Then with  $x$  colors:

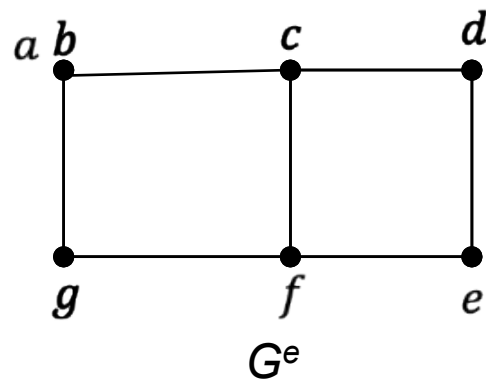
$$P_G(x) = P_{G_e}(x) - P_{G^e}(x).$$



$G_e$



$a, b$  不同颜色



$a, b$  相同颜色



# Proof

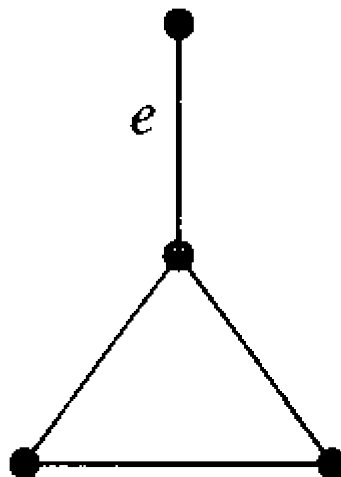
- Consider all the proper colorings of  $G_e$ 
  - $a$  and  $b$  have different colors
  - $a$  and  $b$  have the same color
- A coloring of the first type is also a proper coloring for  $G$ 
  - since  $a$  and  $b$  are connected in  $G$ , and this coloring gives them different colors.

# Proof

- $a$  and  $b$  have different colors
- $a$  and  $b$  have the same color
- A coloring of  $G_e$  of the second type corresponds to a proper coloring of  $G^e$ .
  - In fact, since  $a$  and  $b$  are combined in  $G^e$ , they must have the same color there. All other vertices of  $G_e$  have the same connections as in  $G$ . Thus
  - $P_{G^e}(x) = P_G(x) + P_G^e(x)$ 
    - or
  - $P_G(x) = P_{G^e}(x) - P_G^e(x)$
- Q.E.D

# Example 8

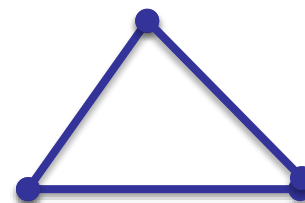
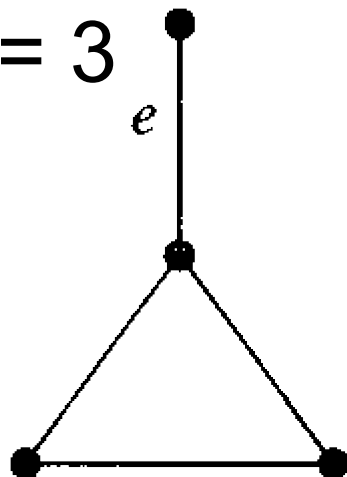
- Let us compute  $P_G(x)$  for the graph  $G$  using the edge  $e$ .
- Then  $G^e$  is  $K_3$  and  $G^e$  has two components, one being a single point and the other being  $K_3$ .



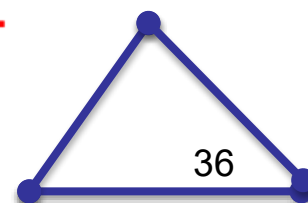
# Example 8

- By Theorem 1
  - $P_{G_e}(x) = x(x(x-1)(x-2)) = x^2(x-1)(x-2)$
  - $P_{G^e}(x) = x(x-1)(x-2)$
- By Theorem 2,
  - $P_G(x) = x^2(x-1)(x-2) - x(x-1)(x-2)$   
 $= x(x-1)^2(x-2)$

- So,  $\chi(G) = 3$



–



36

# Example 8

- By Theorem 1

$$- P_{G_e}(x) = x(x-1)^3$$

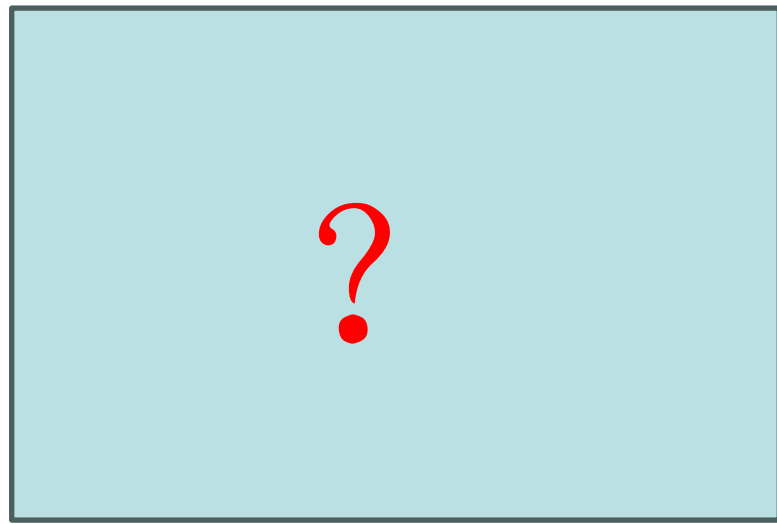
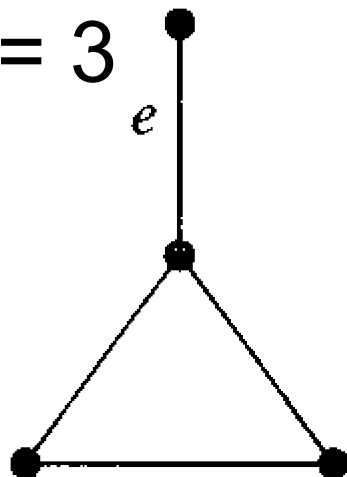
$$- P_{G^e}(x) = x(x-1)^2$$

- By Theorem 2,

$$- P_G(x) = x(x-1)^3 - x(x-1)^2$$

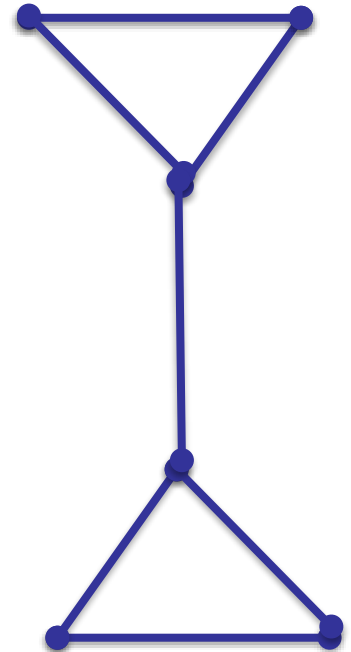
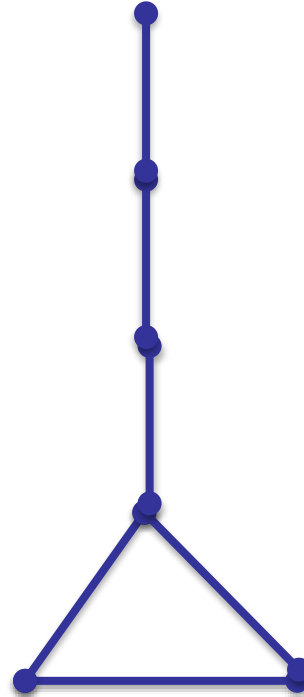
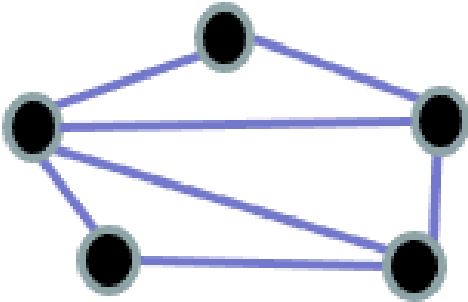
$$= x(x-1)^2(x-2)$$

- So,  $\chi(G) = 3$



# Example

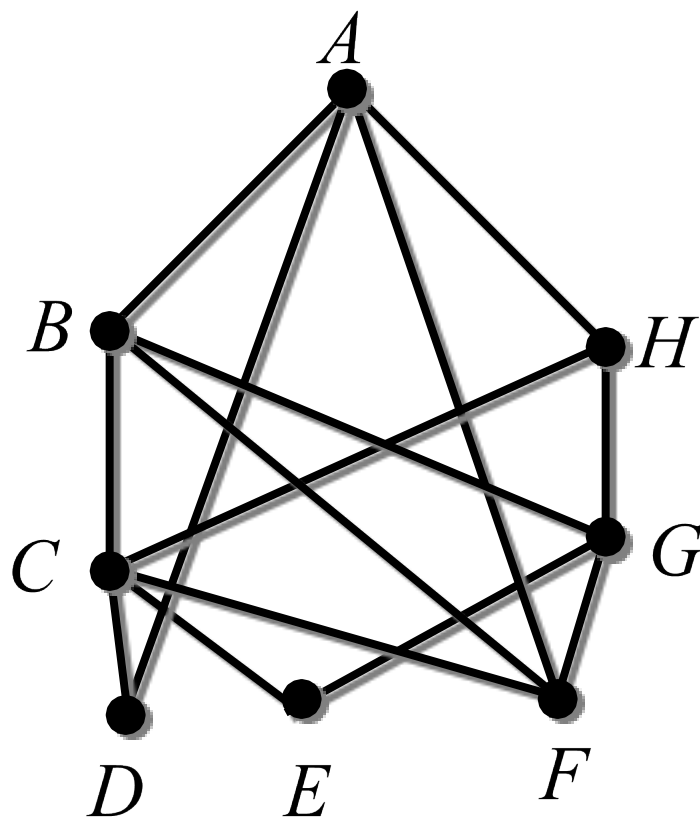
- $P_G(x) = ?$
- $\chi(G) = ?$



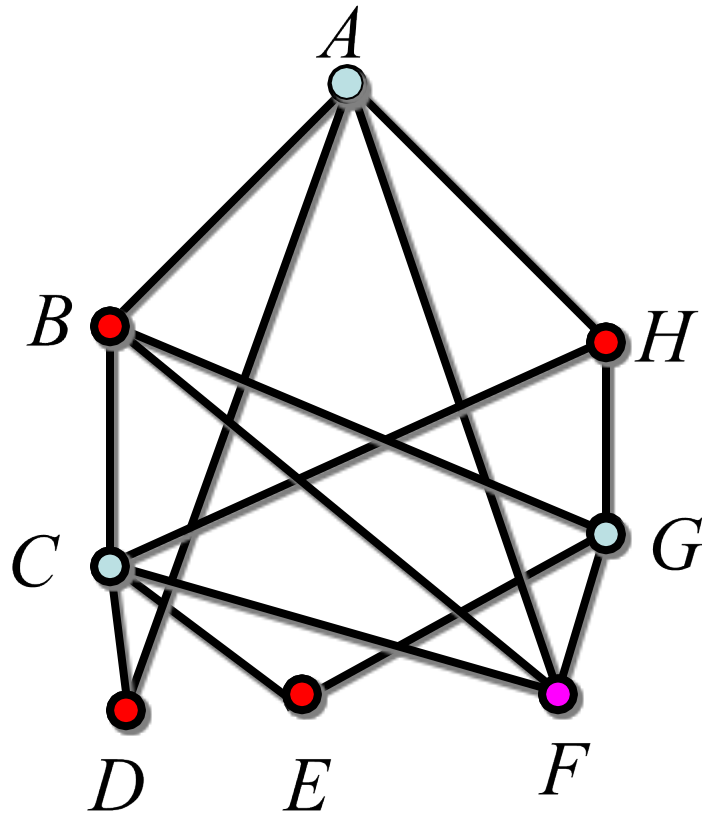
# 点着色的韦尔奇 鲍威尔法 (*Welch Powell*)

- 1) 将图 $G$  中的结点按度数递减的次序进排列(相同度数的结点的排列可随意)。
- 2) 用第一种颜色, 对第一点着色, 并按排列次序对与第一点不相邻的每一点着同样的颜色。
- 3) 用第二种颜色对尚未着色的点重复2), 直到所有的点都着上颜色为止。

例题：试用韦尔奇.鲍威尔法对图进行着色。







1)  $CABFGHDE$

2)  $CAG$

3)  $BDEH$

4)  $F$

$$\chi(G) = 3$$

解：

- 1) 按度数递减次序排列各点  $CABFGHDE$
- 2) 用第一种颜色对  $C$  和与  $C$  不相邻的结点  $A, G$  着色。
- 3) 用第二种颜色对  $B$  和与  $B$  不相邻的结点  $DEH$  着色。
- 4) 用第三种颜色对  $F$  进行着色，所以  $G$  是三色的  $\chi(G) = 3$ 。

# 色多项式递推公式

- 若 $e=(u,v)$ 不是 $G$ 中的边

$$P(G,k)=P(G \cup \langle u,v \rangle)+P(G^e)$$

$$- P_{Ge}(x) = P_G(x) + P_G^e(x)$$

- 若 $e=(u,v)$ 是 $G$ 的边

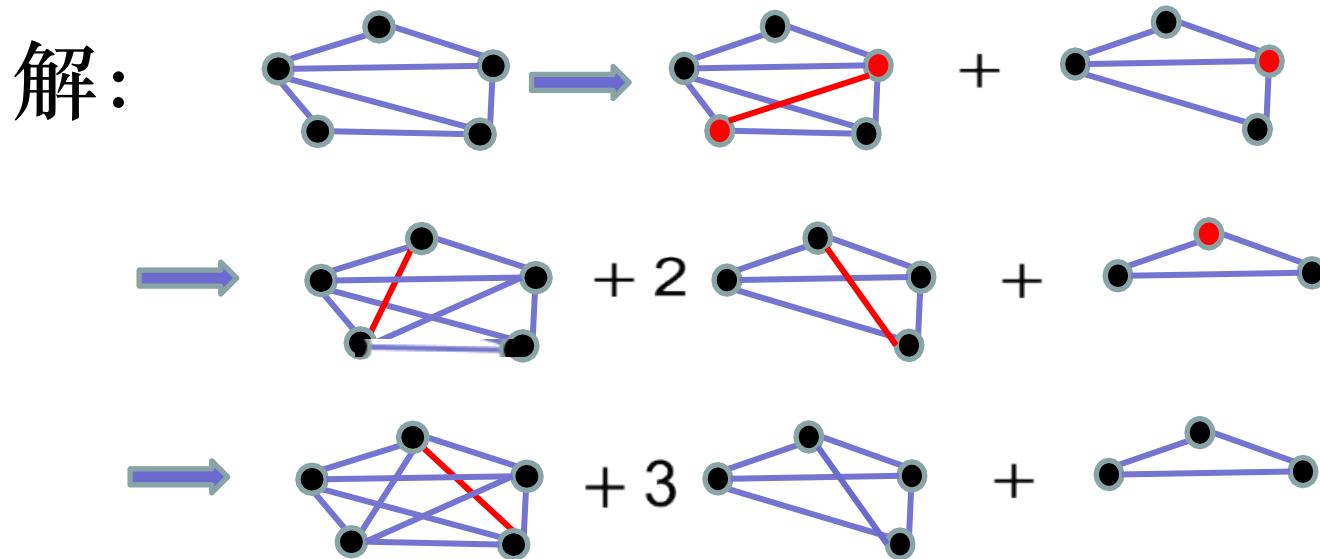
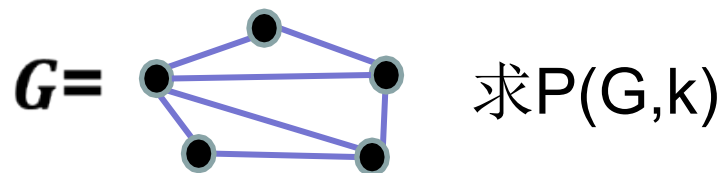
$$P(G,k)=P(G-e)-P(G^e)$$

$$- P_G(x) = P_{Ge}(x) - P_G^e(x)$$

- 推论:

$$P(G,k)=P(K_{n_1})+P(K_{n_2})+\dots+P(K_{n_k})$$

# 例：学生排课



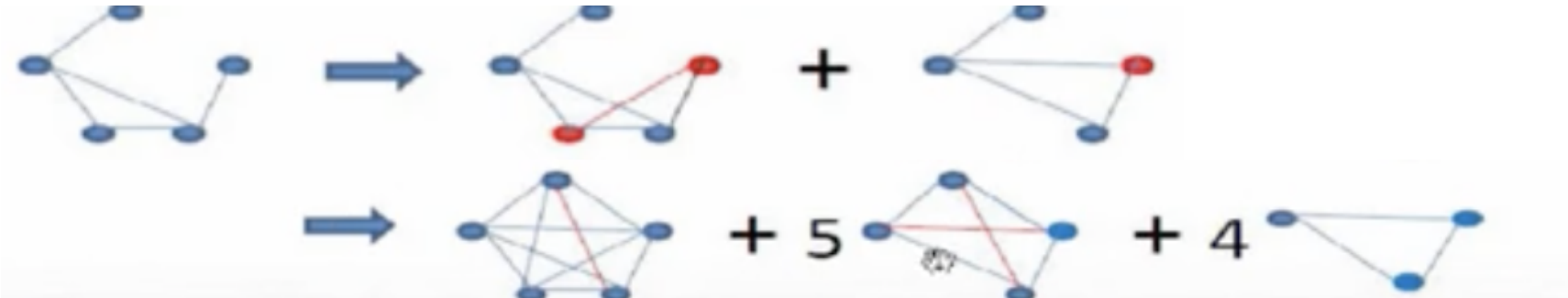
$$P(G, k) = P(K_5, k) + 3P(K_4, k) + \dots + P(K_3, k) = k(k-1)(k-2)^3 = k^5 - 7k^4 + 18k^3 - 20k^2 + 8k$$

$$\chi(k) = \min(5, 4, 3) = 3, \text{ 且 } P(G) = 6$$

# 色多项式性质

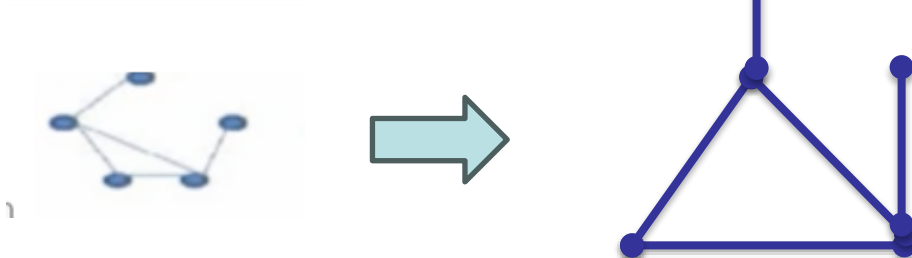
- $n$ 次多项式，系数正负交替
- $k^n$ 系数是1， $k^{n-1}$ 系数绝对值为 $G$ 的边数，常数项为0
- 最低非零项， $k^p$ ， $p$ 为连通分支数
- 不同连通分支相乘

# 考试排课



$$f(G, k) = k(k-1)^3(k-2) = k^5 - 5k^4 + 9k^3 - 7k^2 + 2k.$$

$\chi(k) = \min(5, 4, 3) = 3$ , 且  $P(G) = 24$



# 作业

- §10.8 10, 16, 20, 23, 34

# Transport networks

## 传输网

- Basic concepts
- Labeling algorithm
- The Max flow Min Cut Theorem

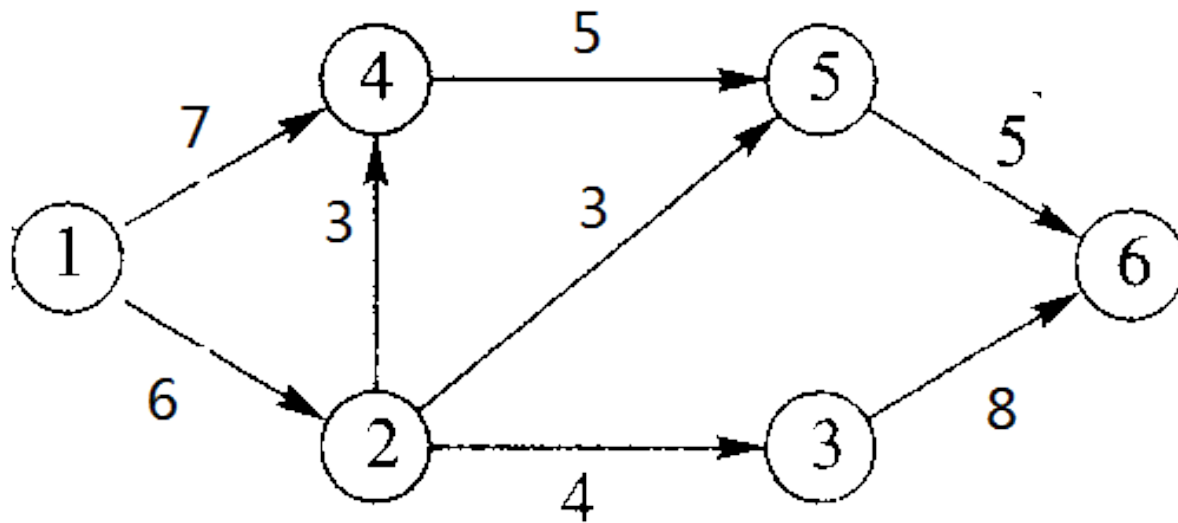


# Directed graph (Digraph)

- An important use of labeled digraphs is to model what are commonly called transport networks.
- The label on an edge represents the maximum flow that can be passed through that edge and is called the *capacity* (容量) of the edge. Many situations can be modeled in this way.

# Example

- Figure below might as easily represent an oil pipeline, a highway system, a communications network, or an electric power grid.



# Transport network

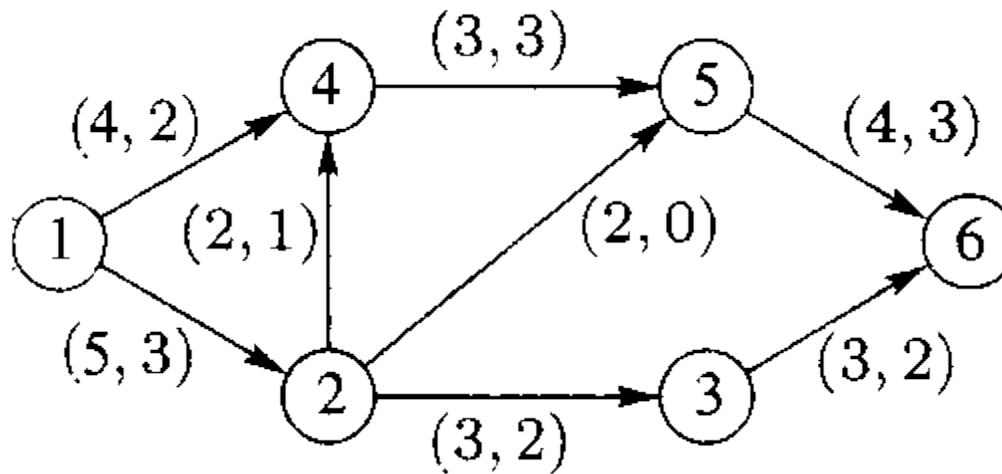
- More formally, a *transport network*, or a *network*, is a connected digraph  $N$  with the following properties:
  - There is a unique node, the *source* (源端), that has in-degree 0. We generally label the source node 1.
  - There is a unique node, the *sink* (宿端), that has out-degree 0. If  $N$  has  $n$  nodes, we generally label the sink as node  $n$ .
  - The graph  $N$  is labeled. The label,  $C_{ij}$ , on edge  $(i, j)$  is a nonnegative number called the *capacity* of the edge.

# Flows

- Mathematically, a *flow* (流量) in a network  $N$  is a function that assigns to each edge  $(i, j)$  of  $N$  a nonnegative number  $F_{ij}$  that does not exceed  $C_{ij}$ .
  - *Conservation of flow* (流量守恒)
  - *Value of the flow*

# Flows

- We can represent a flow  $F$  by labeling each edge  $(i, j)$  with the pair  $(C_{ij}, F_{ij})$



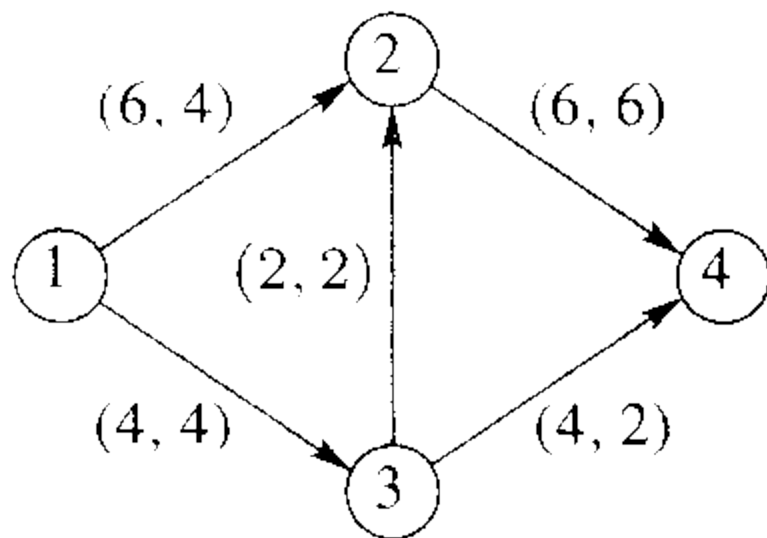
- Here  $\text{value}(F) = 5$

# Maximum Flows (最大流量)

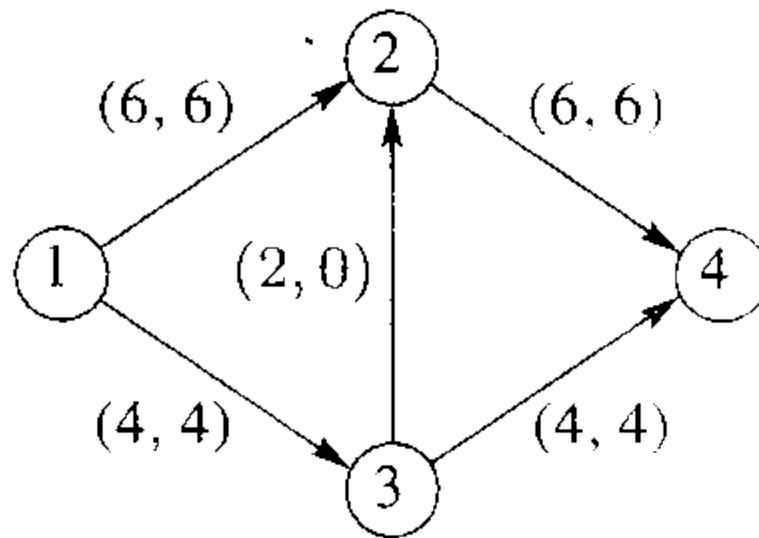
- For any network an important problem is to determine the maximum value of a flow through the network and to describe a flow that has the maximum value.
- For obvious reasons this is commonly referred to the *maximum flow problem*.

# Example 2

- Even for a small network, we need a systematic procedure for solving the maximum flow problem.

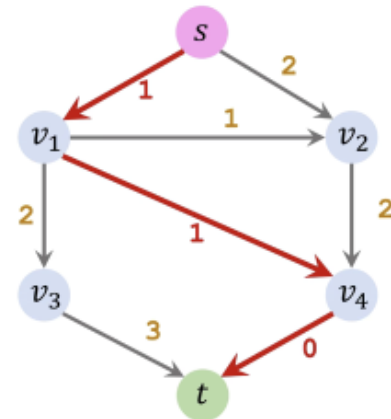
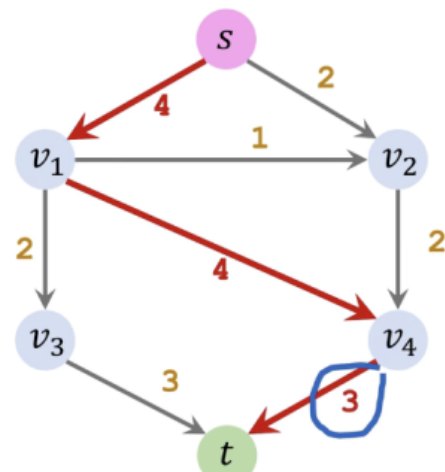
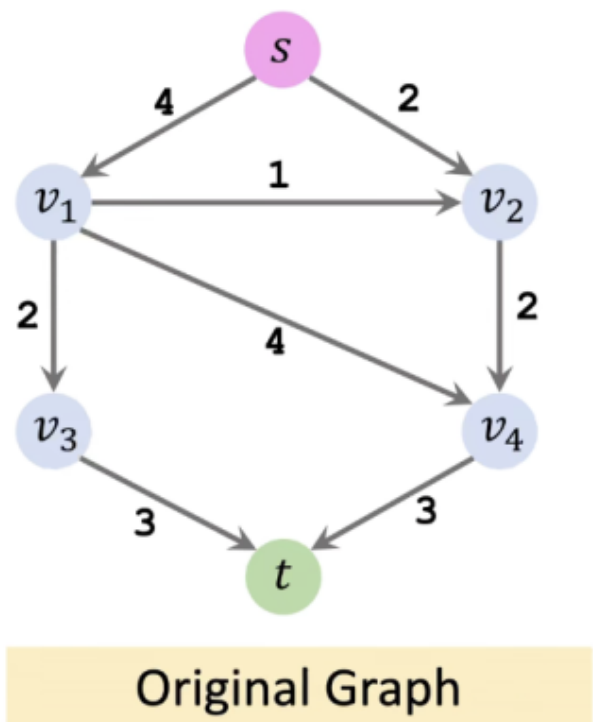


(a)

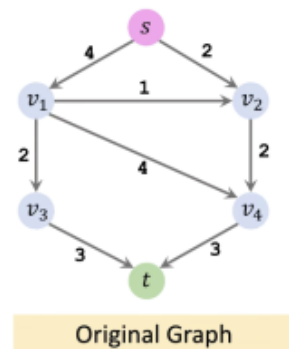


(b)

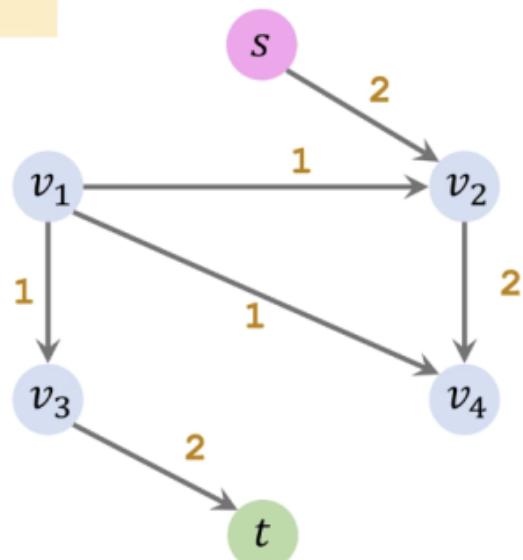
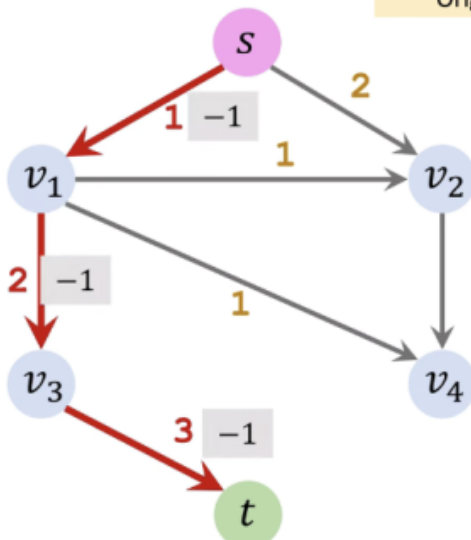
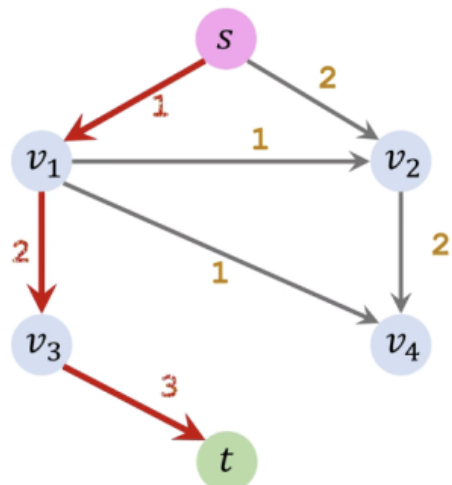
# 求流量的简单算法







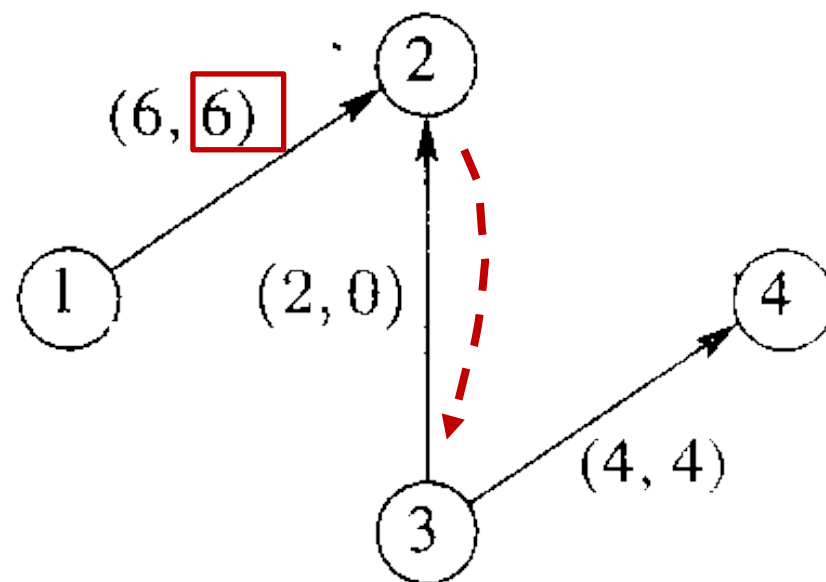
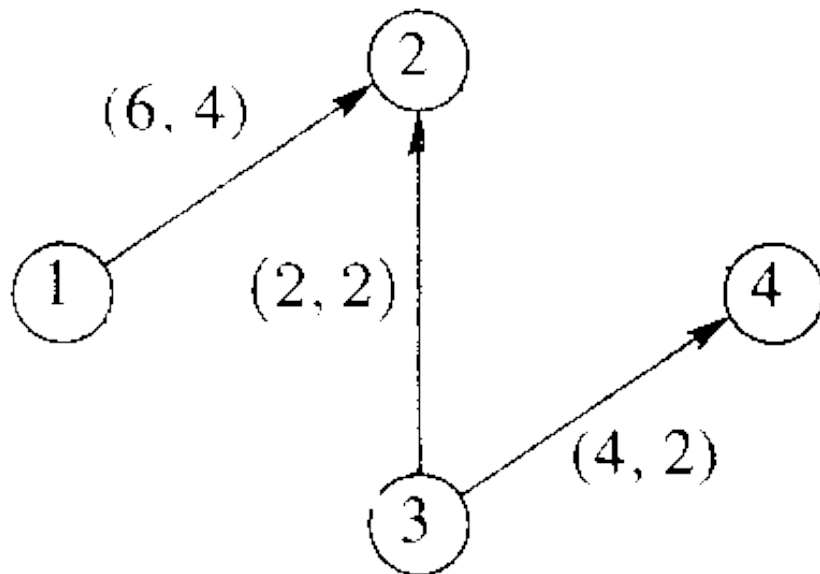
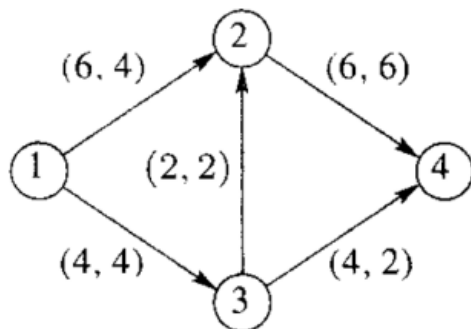
## 第二次循环



第三次循环：无路可达 $t$ ，结束，流量4

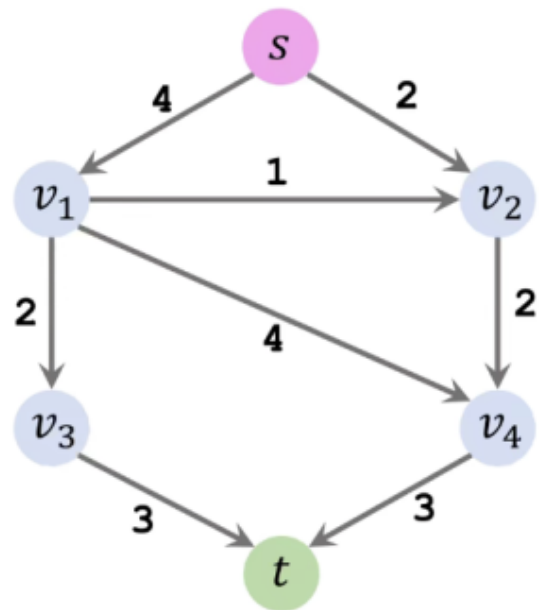
显然不是最大流。原因与每次选择的路径有关。

# Virtual Flow

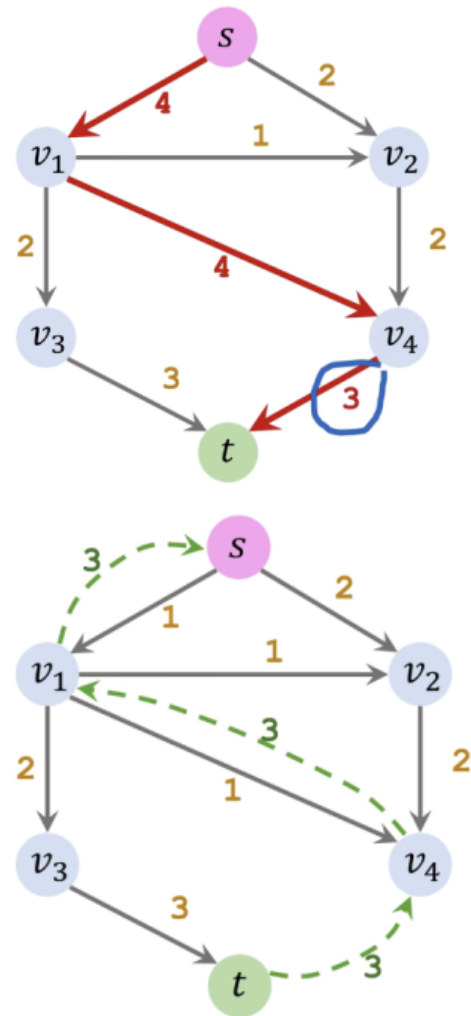


- Let  $N$  be a network and let  $G$  be the symmetric closure of  $N$ .
  - Choose a path in  $G$  and let an edge  $(i,j)$  in this path.
  - If  $(i,j) \in N$  and  $e_{ij} = C_{ij} - F_{ij} > 0$ , then we say this edge has positive excess capacity.
  - If  $(i,j)$  is not an edge of  $N$  then we are traveling this edge in the wrong direction.  $e_{ij} = F_{ji}$  if  $F_{ji} > 0$ . Increasing flow through edge  $(i,j)$  will have the effect of reducing  $F_{ji}$ .

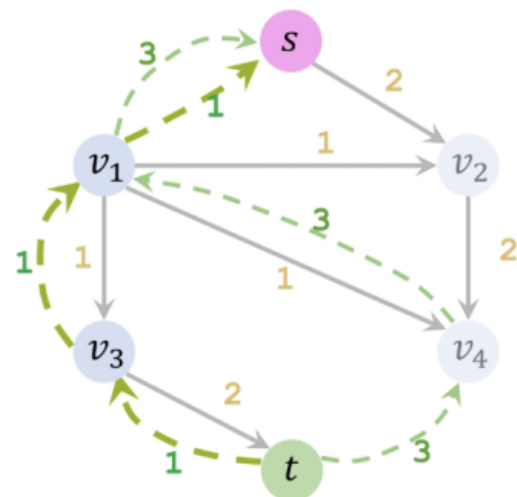
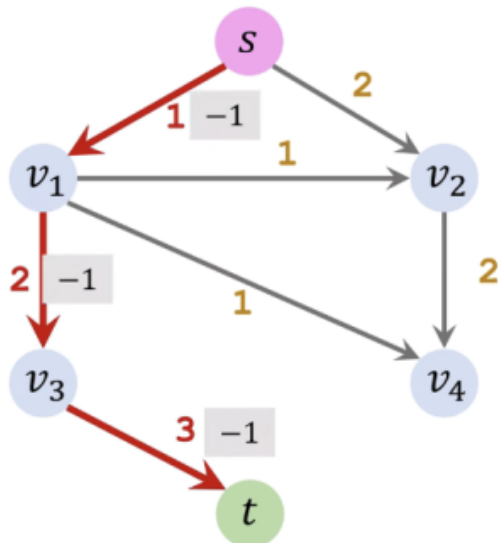
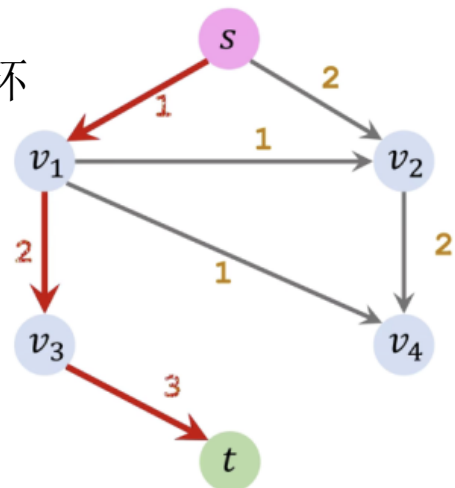
# 改进算法



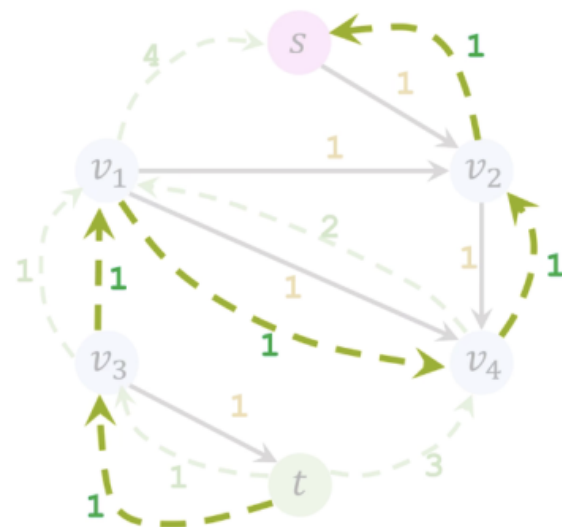
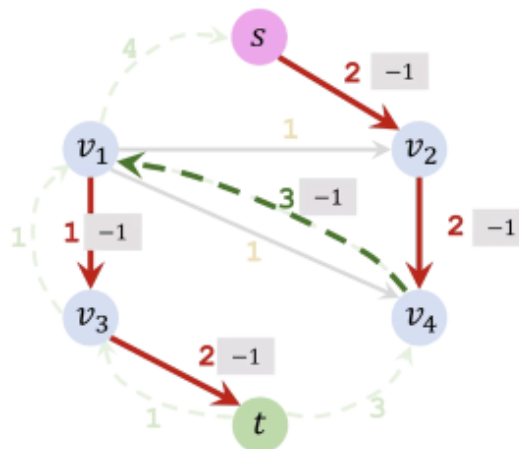
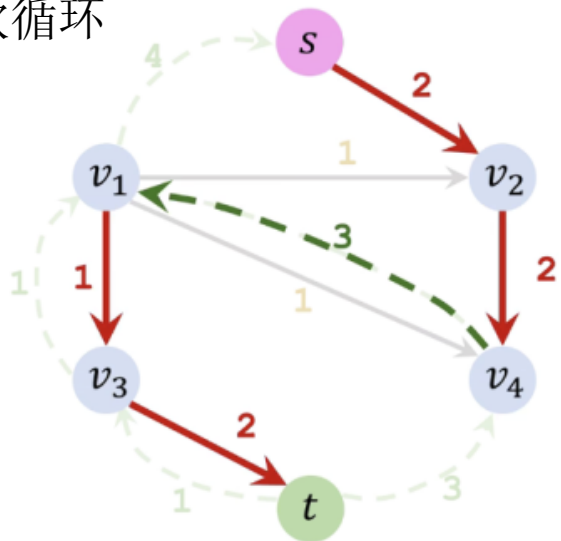
Original Graph



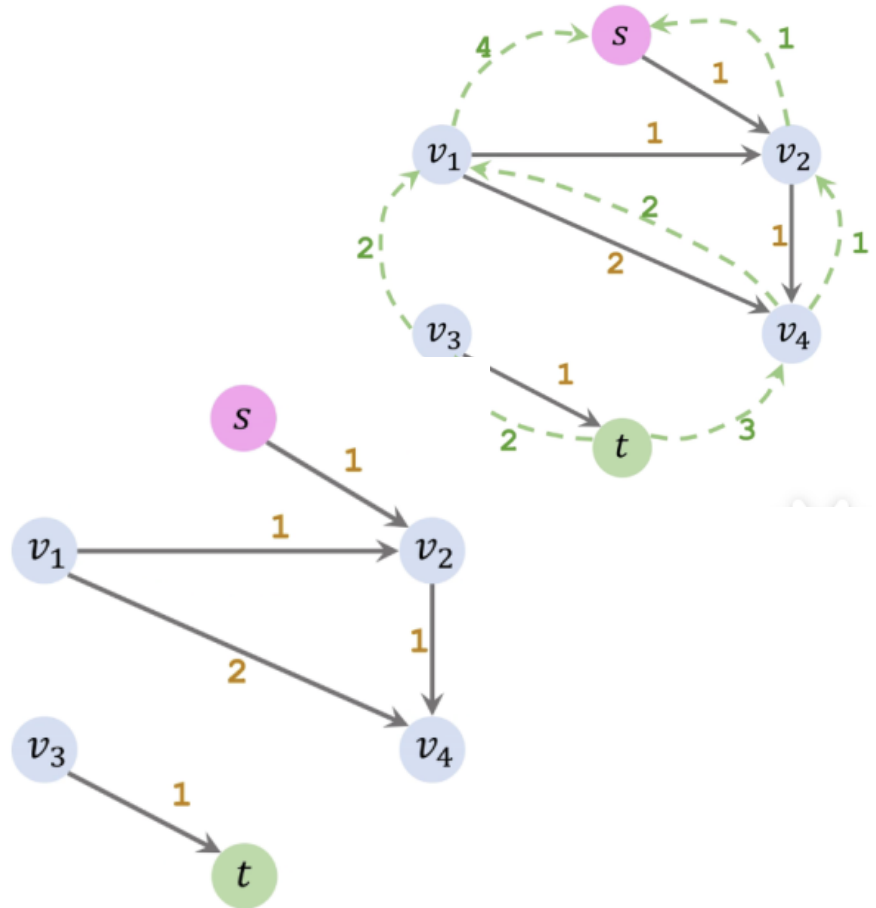
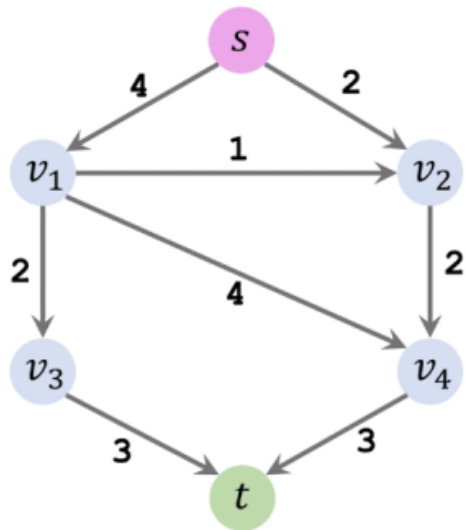
## 第二次循环



## 第三次循环



## 第四次循环—结束



# LABELING ALGORITHM

## Ford - Fulkerson

- The algorithm we present is due to Ford and Fulkerson and is often called the *labeling algorithm* (标记算法). The labeling referred to is an additional labeling of nodes. We have used integer capacities for simplicity, but Ford and Fulkerson show that this algorithm will stop in a finite number of steps if the capacities are rational numbers.

# LABELING ALGORITHM

- Let  $N$  be a network with  $n$  nodes and  $G$  be the symmetric closure of  $N$ . All edges and paths used are in  $G$ .
  - Begin with all flows set to 0.
  - As we proceed, it will be convenient to track the excess capacities in the edges and how they change rather than tracking the increasing flows.
  - When the algorithm terminates, it is easy to find the maximum flow from the final excess capacities.



# Step 1

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

## 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 edge 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 edge  $(j, k)$ .
- When all the nodes in  $N_2(j)$  are labeled in this way, repeat this process for the other nodes in  $N_1$ . Let

$$N_2 = \bigcup_{j \in N_1} N_2(j)$$

# Step 3

- Repeat Step 2, labeling all previously unlabeled nodes  $N_3$  that can be reached from a node in  $N_2$  by an edge having positive excess capacity.
- Continue this process forming sets  $N_4, N_5, \dots$  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; remember that the source is not labeled. or
  - (ii) the sink has been labeled.

# Step 4

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

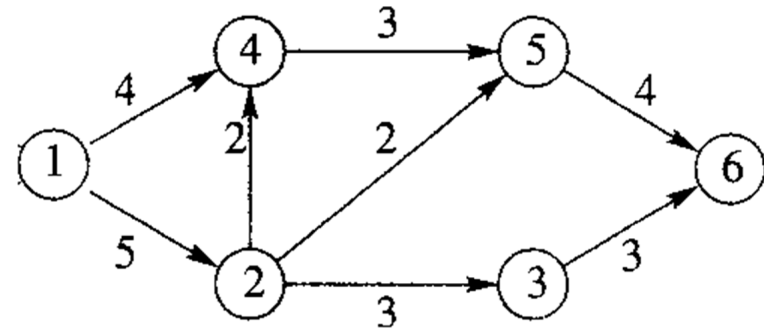
# Step 5

- In case (ii) the sink, node  $n$ , has been labeled with  $[E_n, m]$  where  $E_n$  is the amount of extra flow that can be made to reach the sink through a path  $\pi$ .

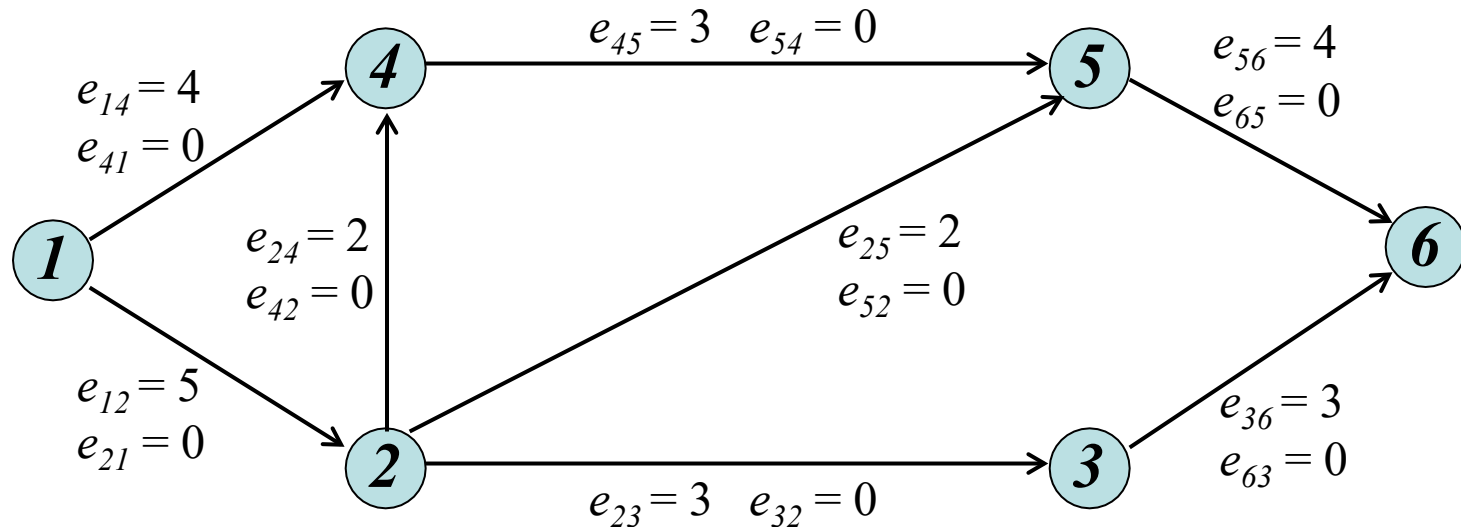
# Step 5

- We examine  $\pi$  in reverse order.
  - If edge  $(i, j) \in N$ , 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) edge  $(j, i)$  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 Step 1.

# Example 3

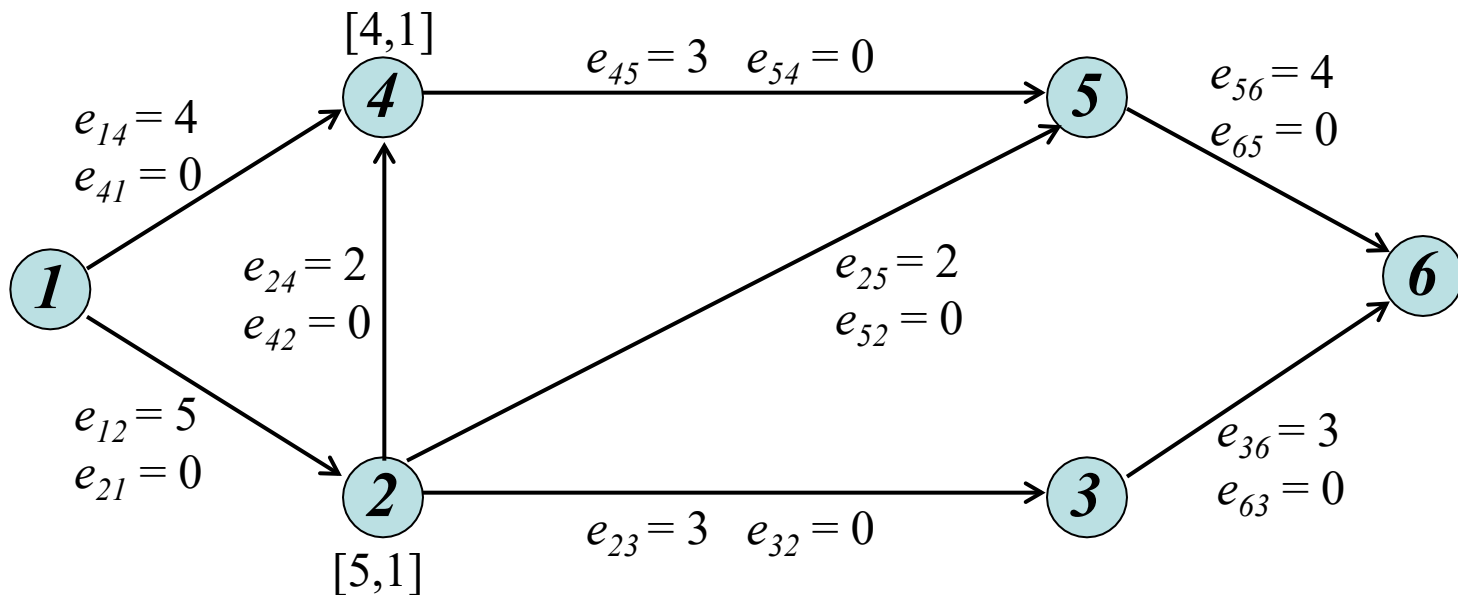


- The initial flow in all edges is zero



# Example 3: step 1

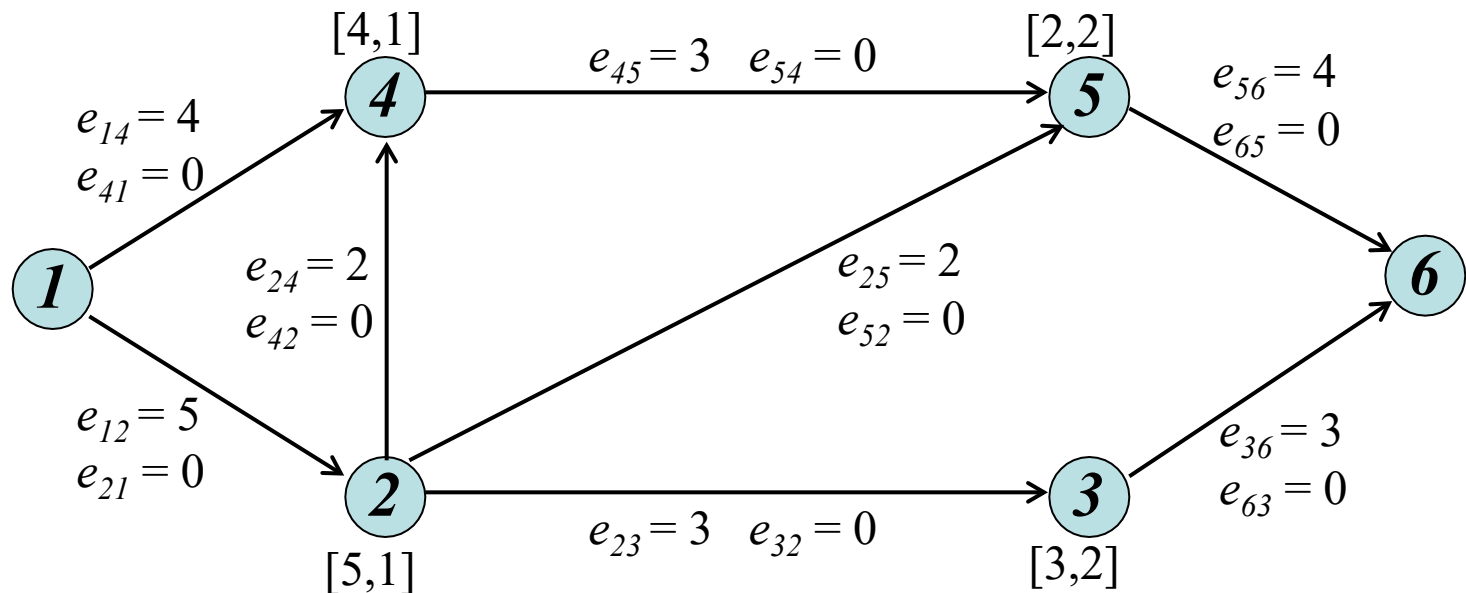
- Starting at the source, we can reach nodes 2 and 4 by edges having excess capacity, so  $N_1 = \{2, 4\}$





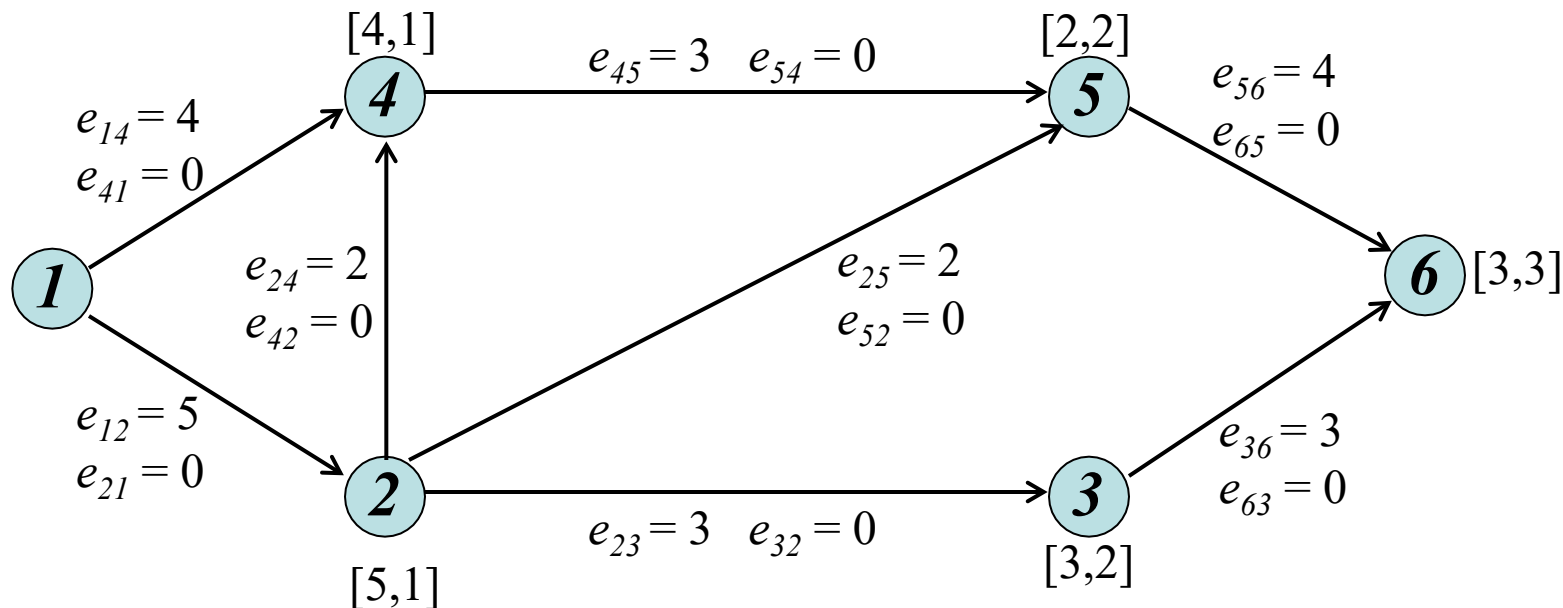
# Example 3: step 2

- From node 2 we can reach nodes 5 and 3
- Label node 5 and node 3
- We cannot travel from node 4 to any unlabeled node by one edge. Thus,  $N_2 = \{3, 5\}$



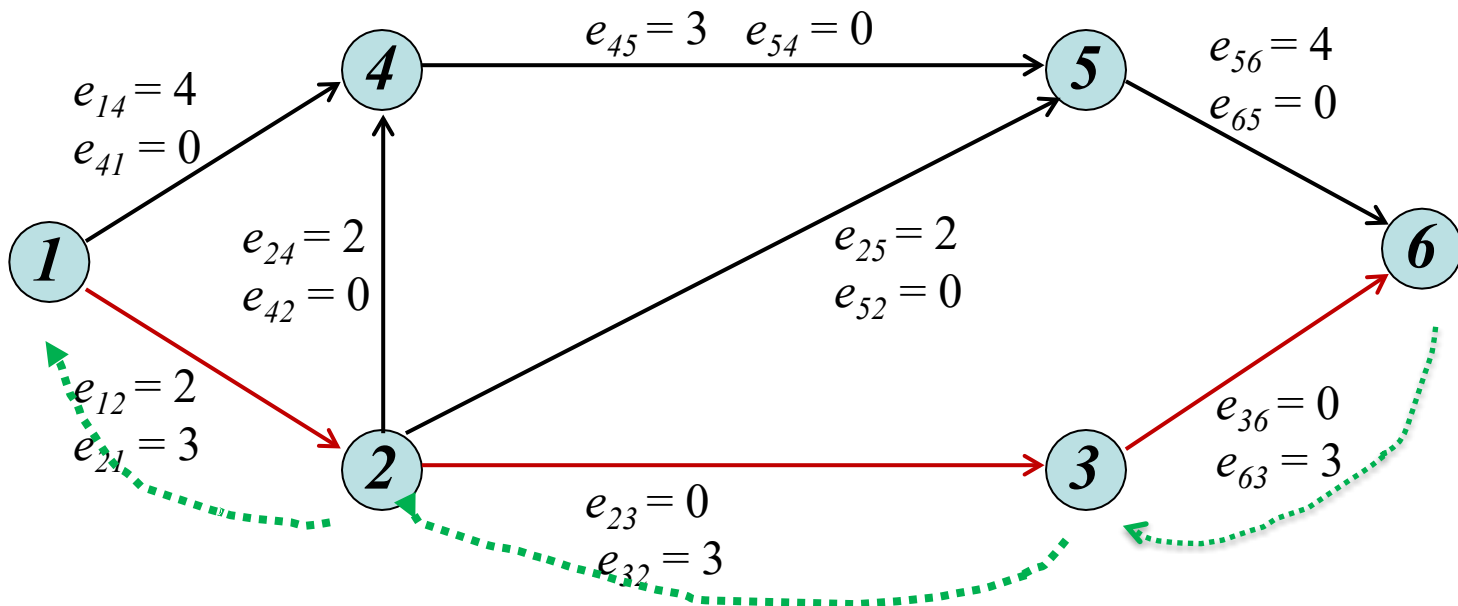
# Example 3: step 3

- We repeat Step 2 using  $N_2$ . We can reach the sink from node 3 and 3 units through edge (3, 6). Thus the sink is labeled with [3, 3]



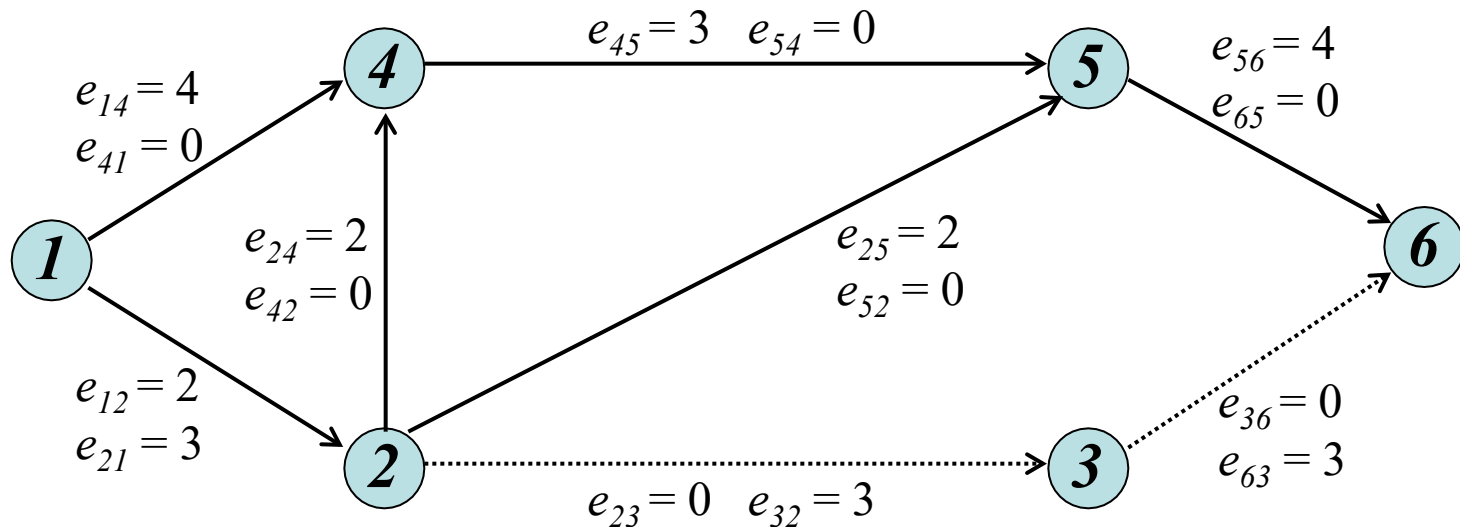
# Example 3 : step 5

- the path is 1, 2, 3, 6
- subtract 3 from the excess capacity of each edge, indicating an increased flow through that edge, and adding an equal amount to the excess capacities of the (virtual) edges.

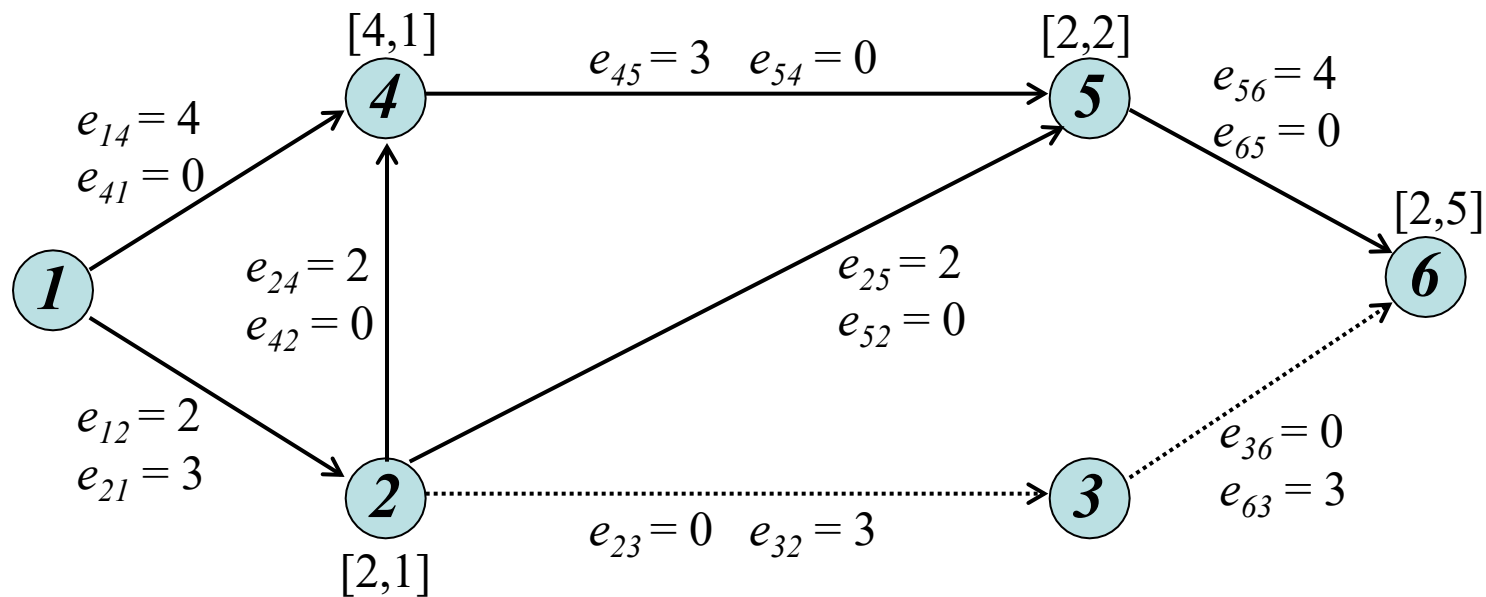


# Example 3

- We now return to Step I with the situation shown as

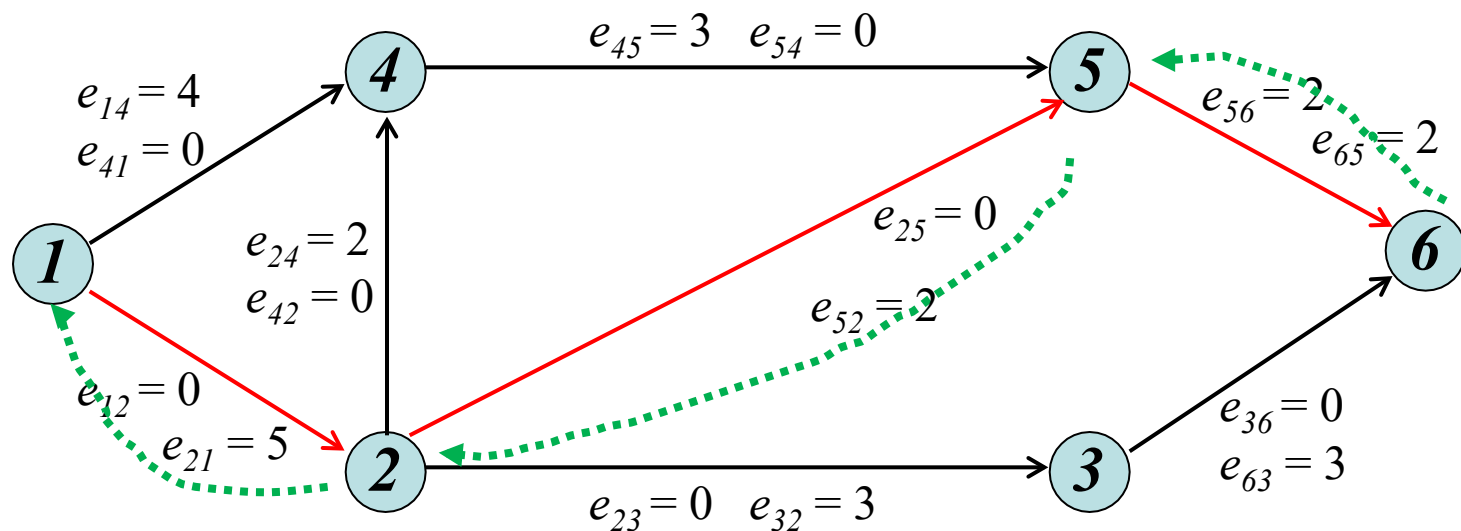


# Steps 1,2,3



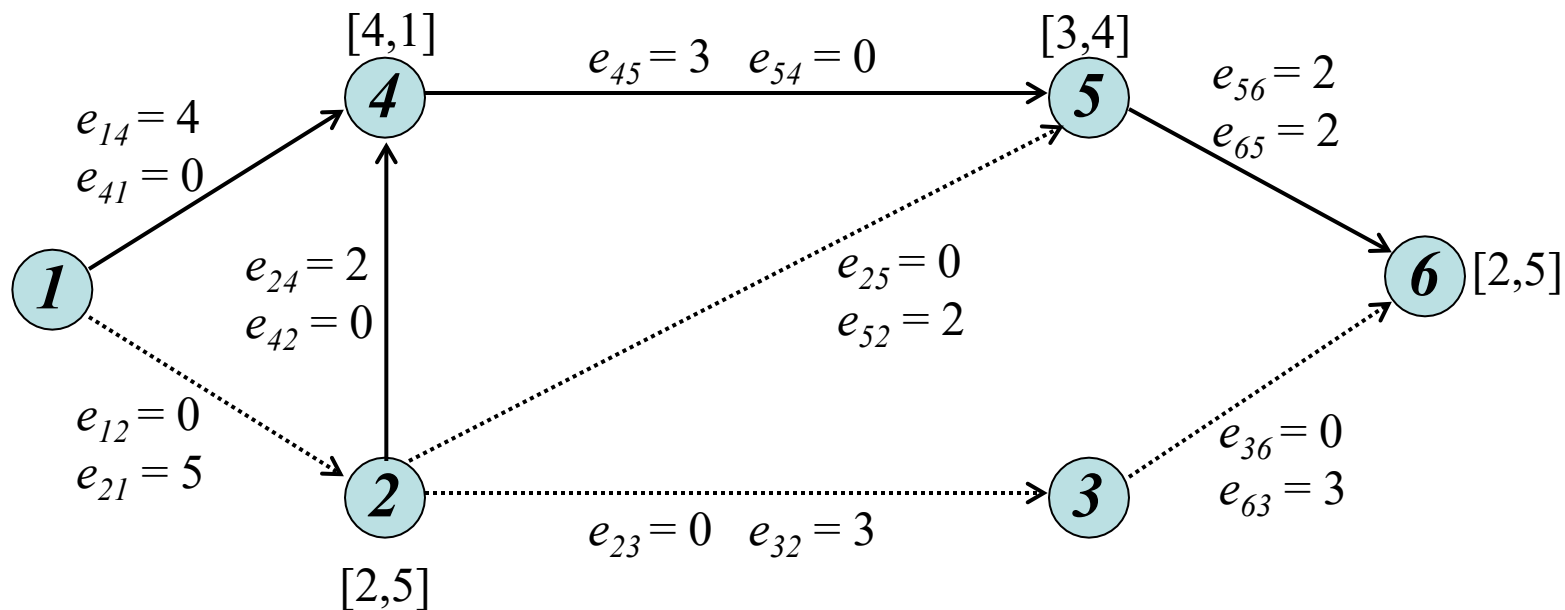
# Step 5

- Work back along path 1,2,5,6, subtracting 2 from the excess capacities of these edges
- And return to Step 1.



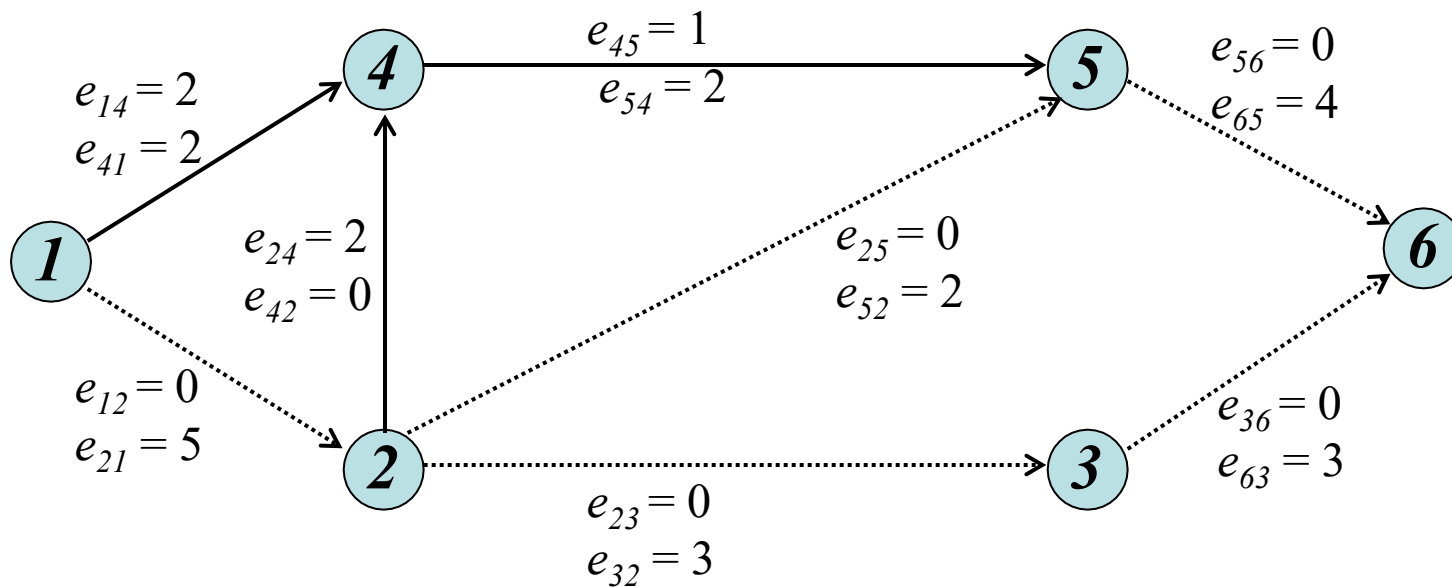
# Step 1,2,3

–  $N_1=\{4\}, N_2=\{5\}, N_3=\{2,6\}$



# Step 5

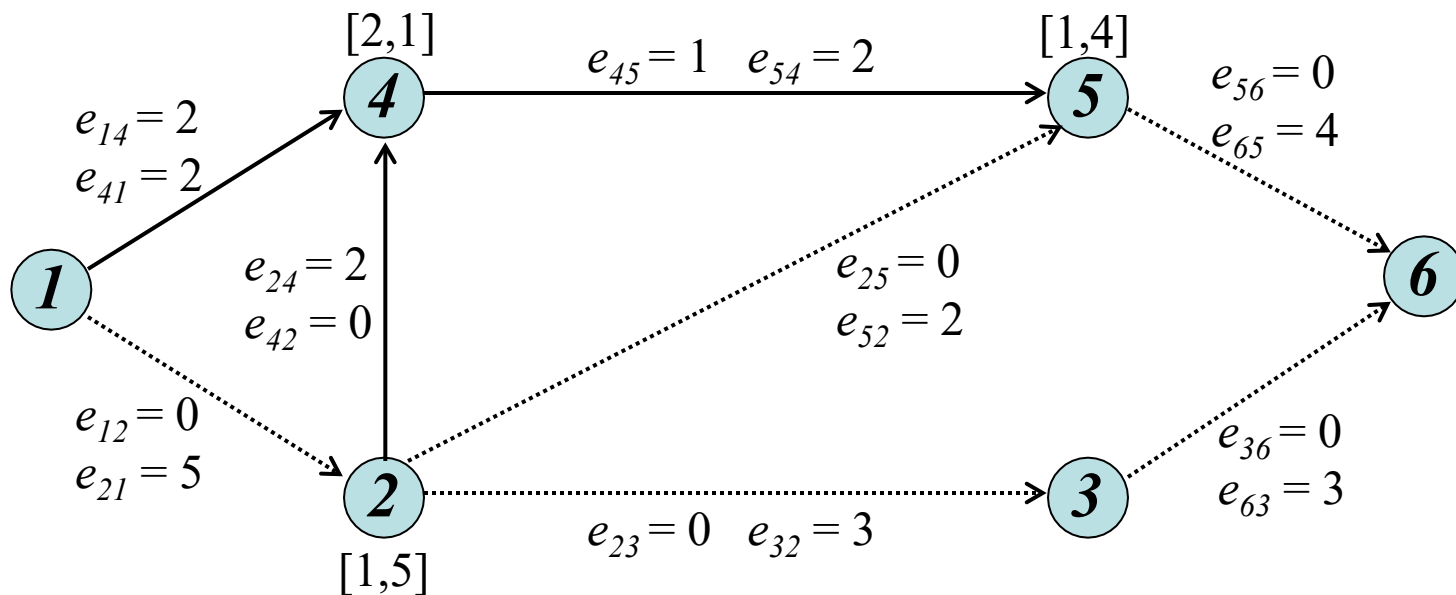
- Work back along the path 1,4,5,6, adjusting excess capacities.
- Return to Step 1.





# Go further?

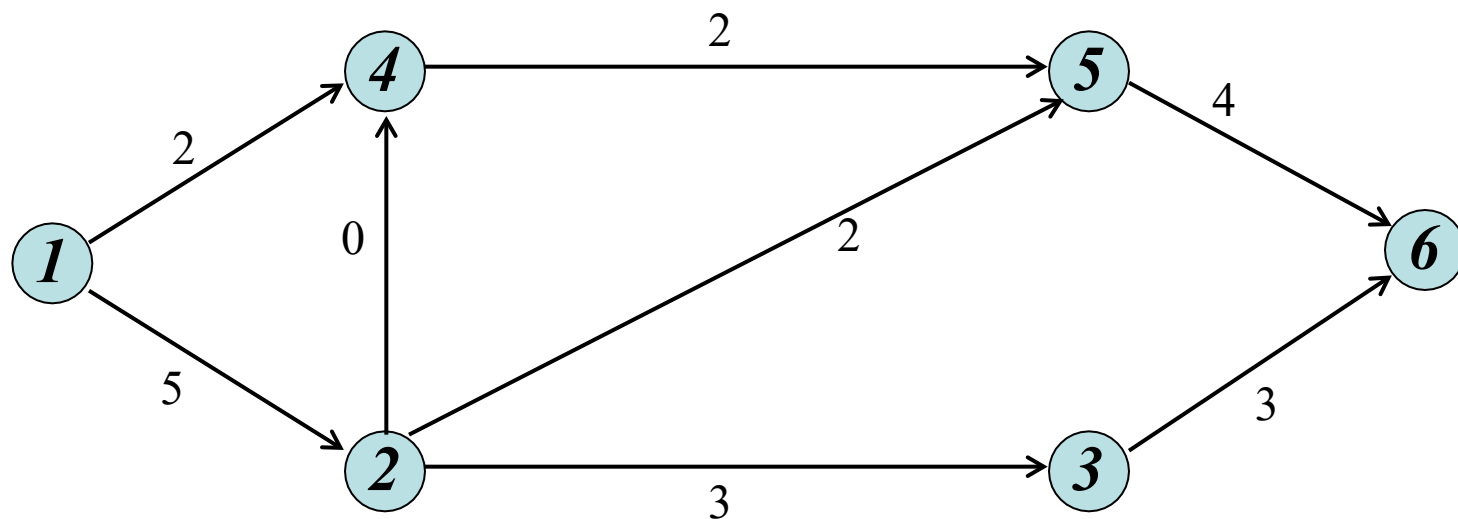
– Step 1,2,3



# Step 4

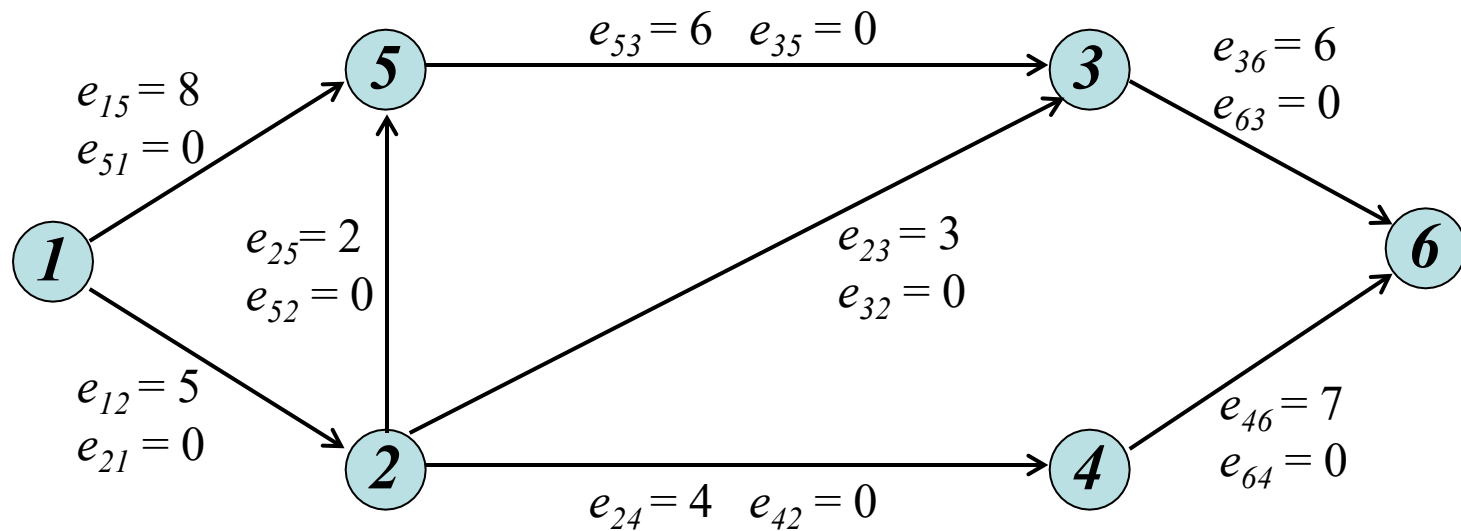
– Terminated with the final overall flow 7.

– Q.E.D.

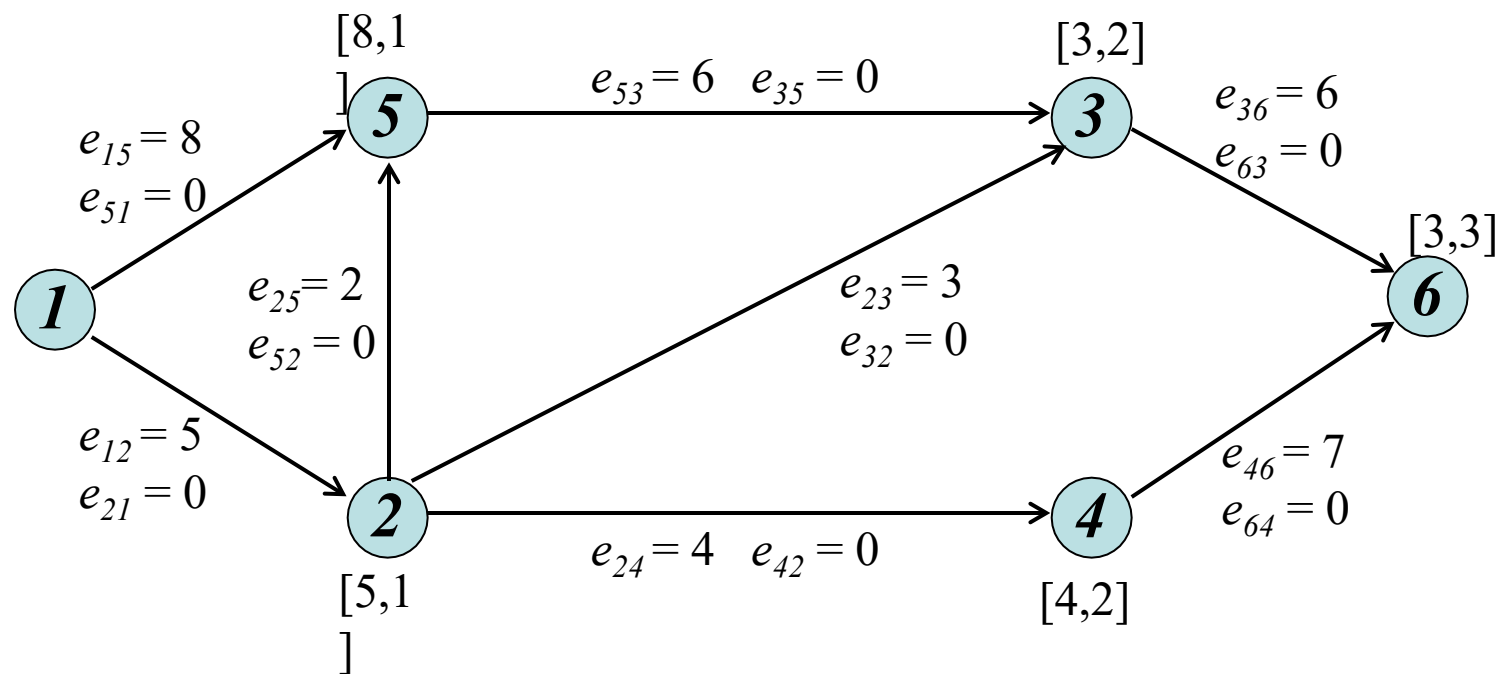


[Go back](#)

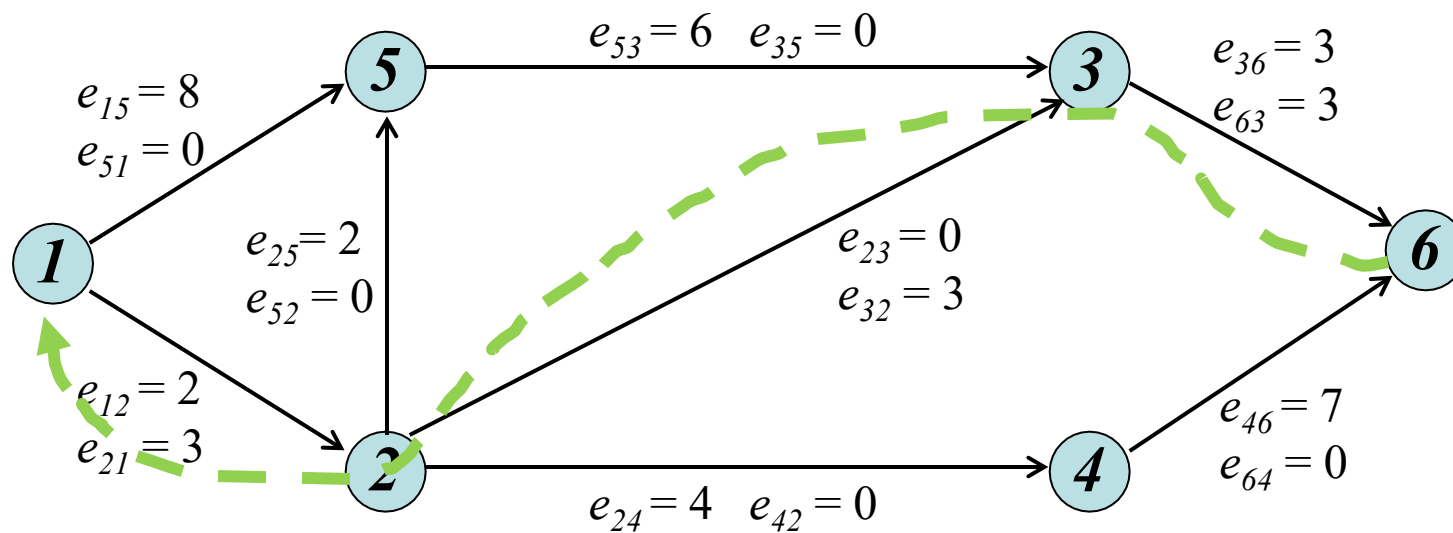
# Example 4



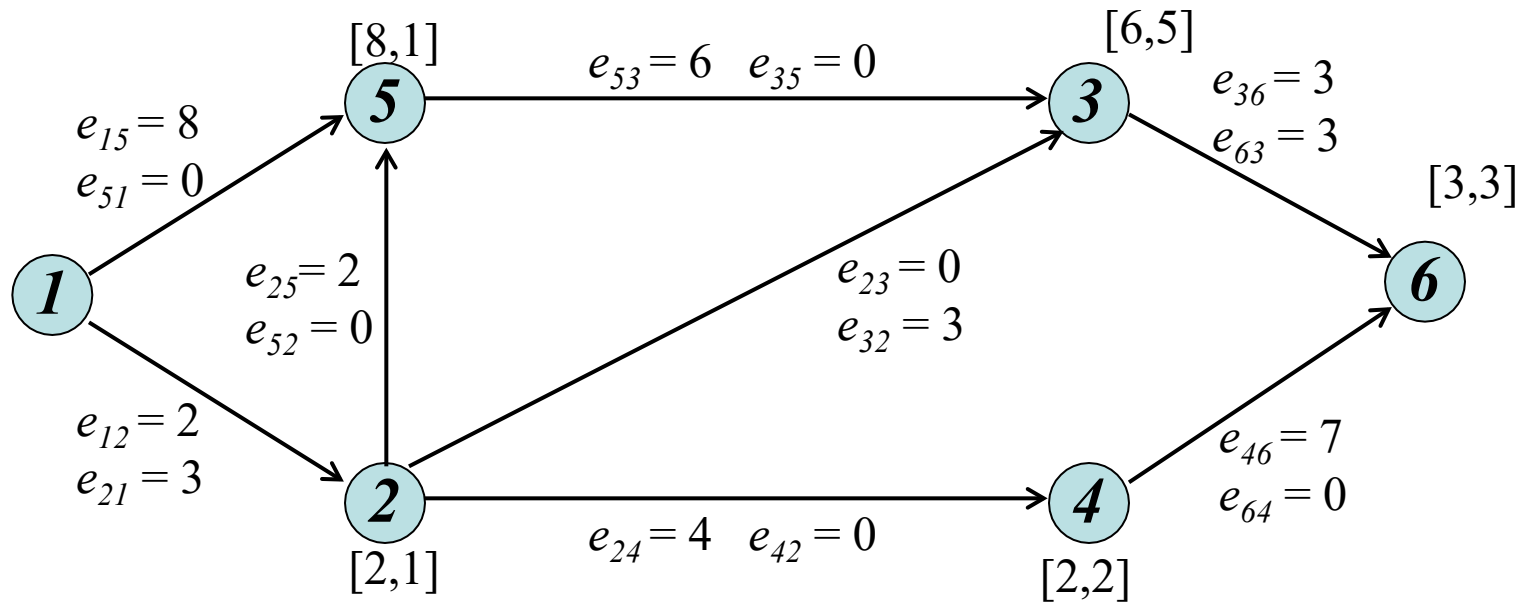
# Step 1,2,3



# Step 5

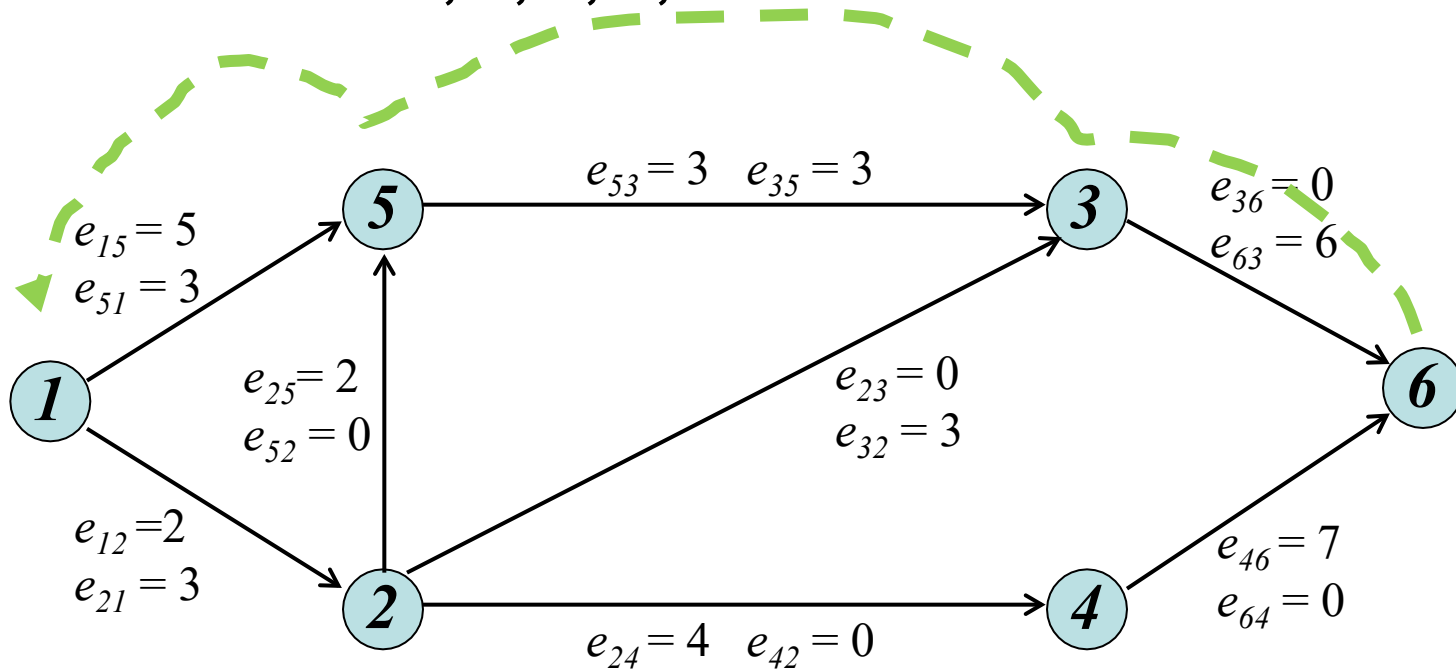


# Step 1,2,3



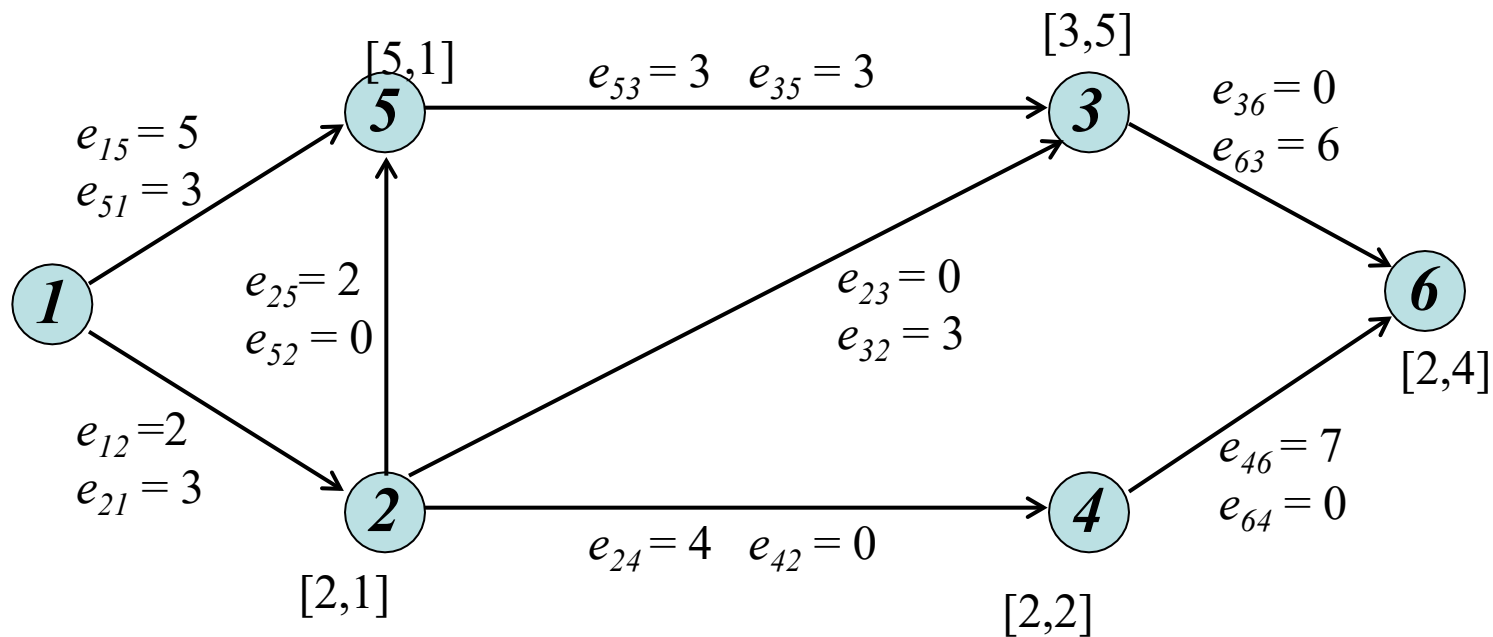
# Step 5

- reverse: 1,5,3,6; subtract 2.



# Step 1,2,3

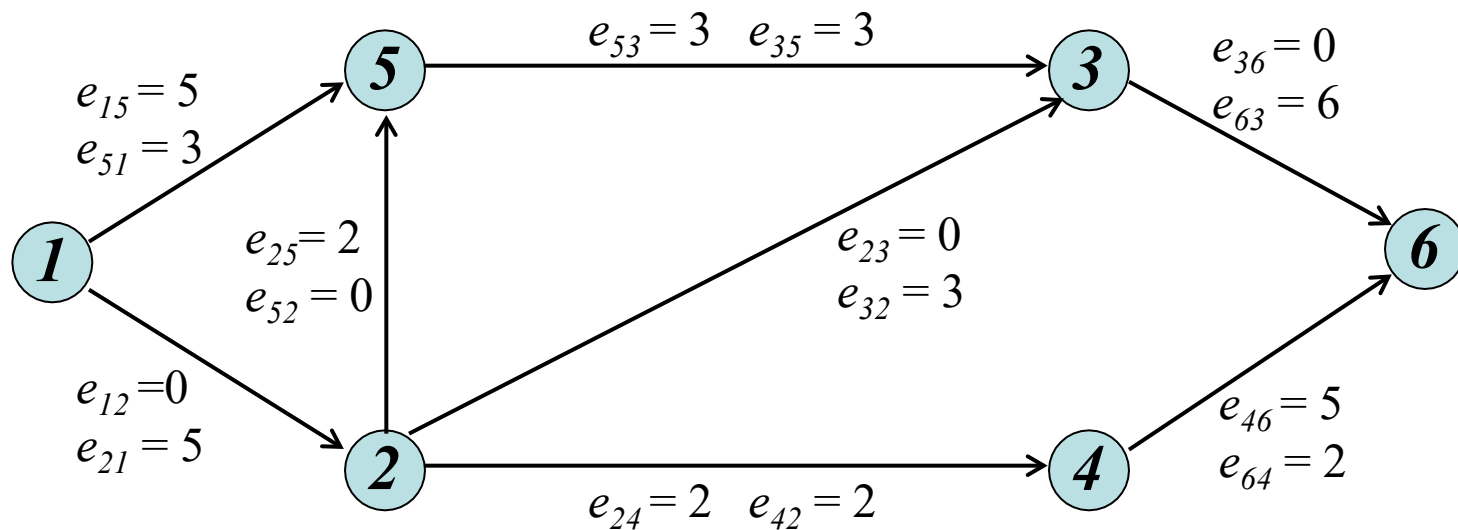
- reverse:





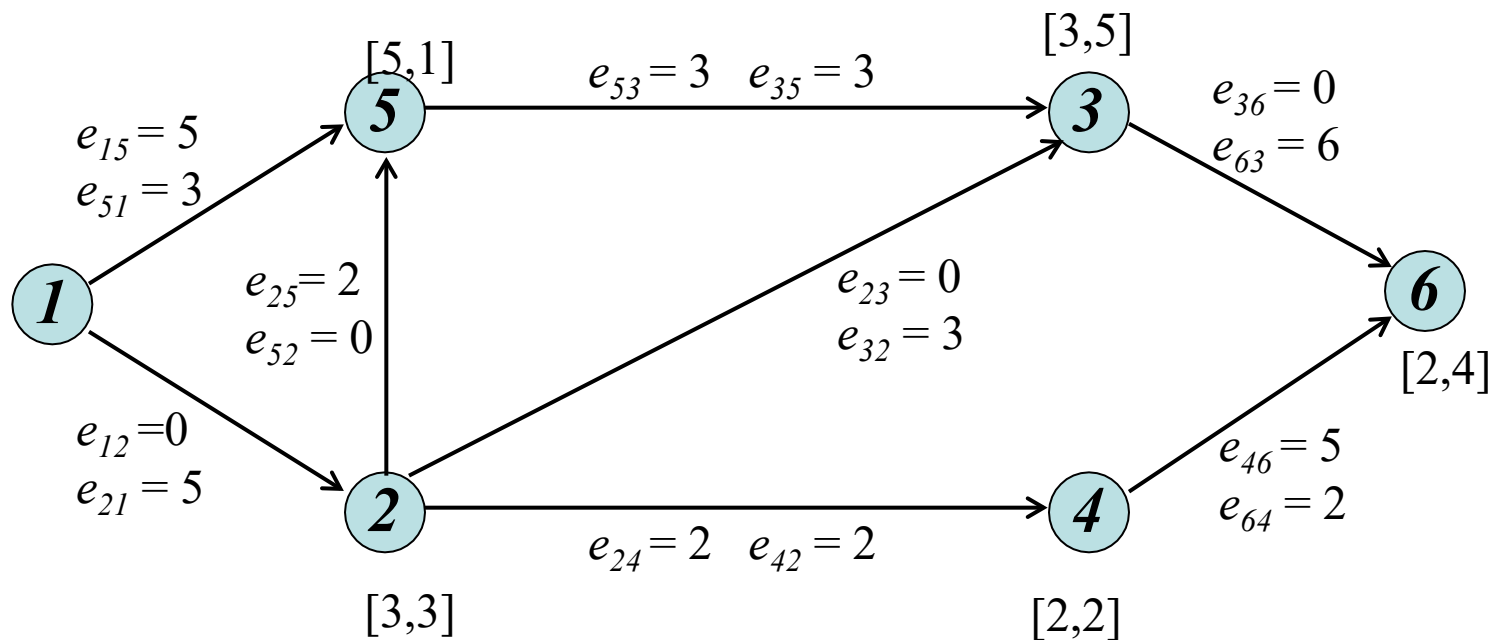
# Step 5

- reverse: 1, 2,4,6



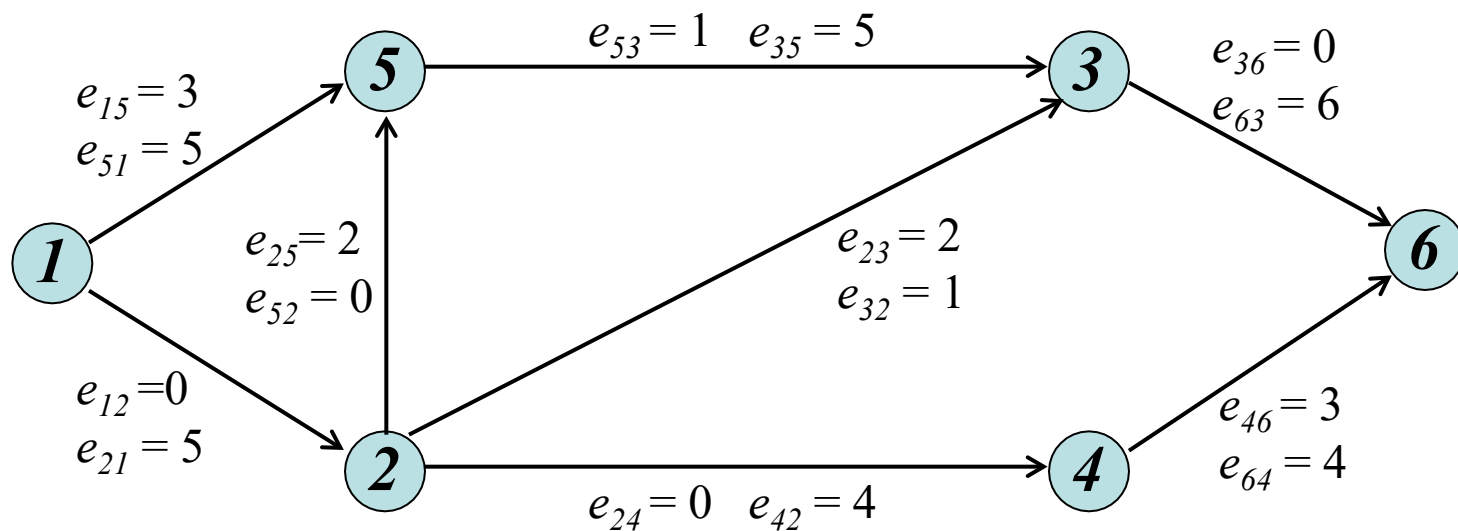
# Step 123

- reverse:



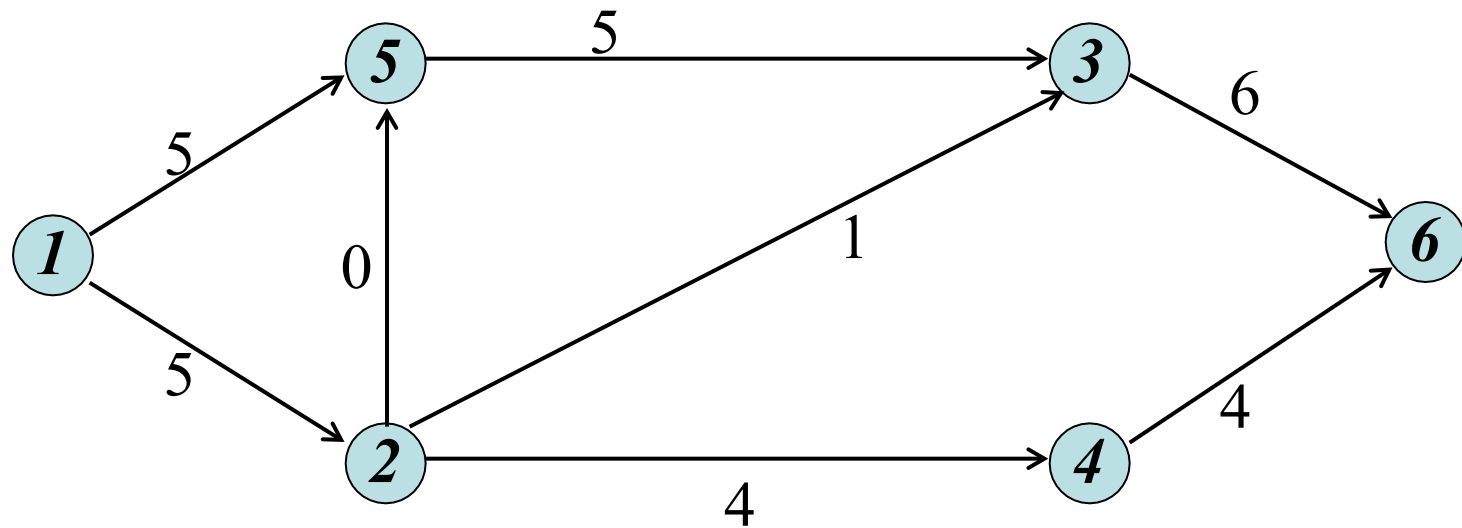
# Step 4

- reverse: 1, 5, 3, 2, 4, 6



# Step 4

- Terminated with the final overall flow 10.

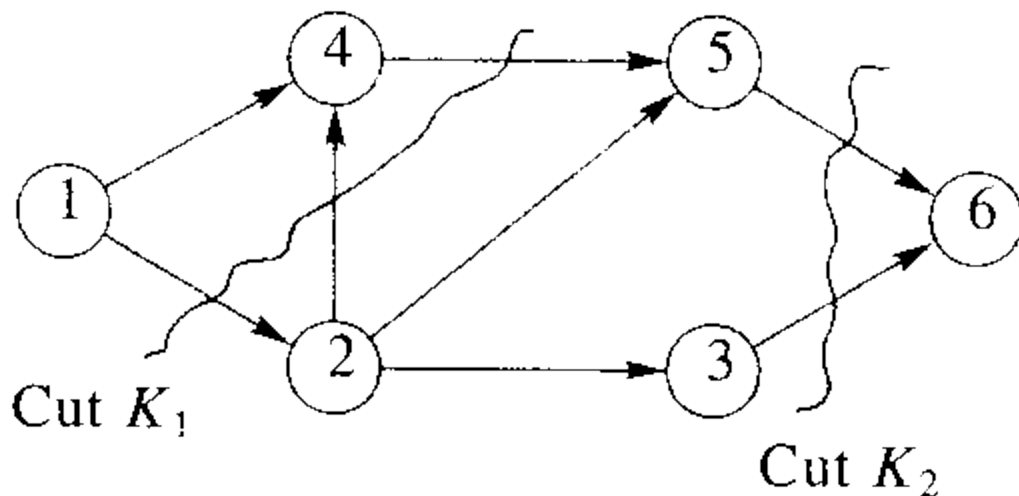


# Cut (割集)

- A *cut* in a network  $N$  is a set  $K$  of edges having the property that every path from the source to the sink contains at least one edge from  $K$ .

$$K_1 = \{ (4,5), (2,4), (1,2) \}:$$

$$K_2 = \{ (5,6), (3,6) \}:$$



# Capacity of a cut $K$

- The *capacity* of a cut  $K$ ,  $c(K)$ , is the sum of the capacities of all edges in  $K$ .
- If  $F$  is any flow and  $K$  is any cut, then
  - $value(F) \leq c(K)$ .

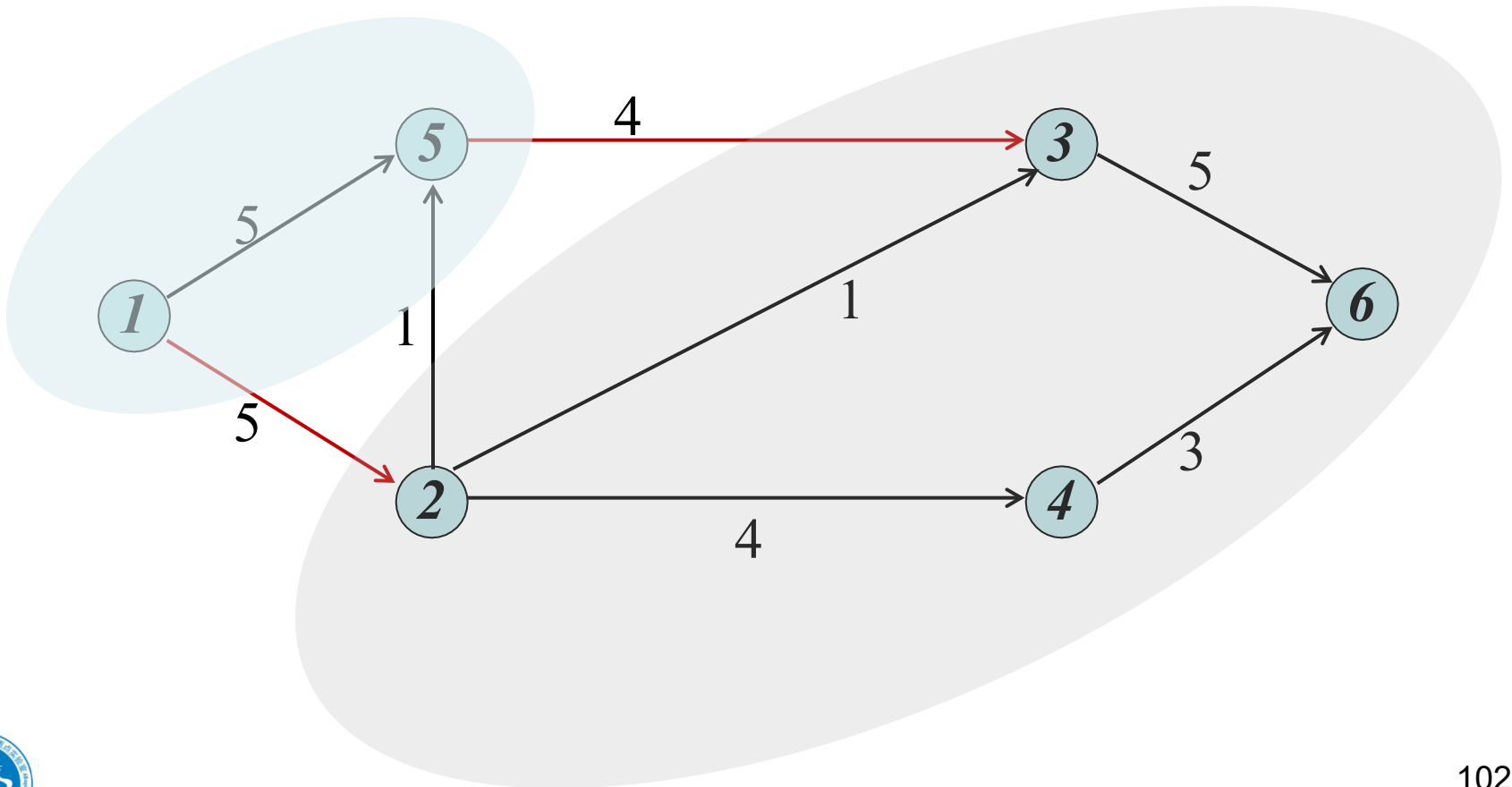
# Theorem

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

# example

$$K = \{ (5,3), (1,2) \}$$

$$c(K) = 4 + 5 = 9$$

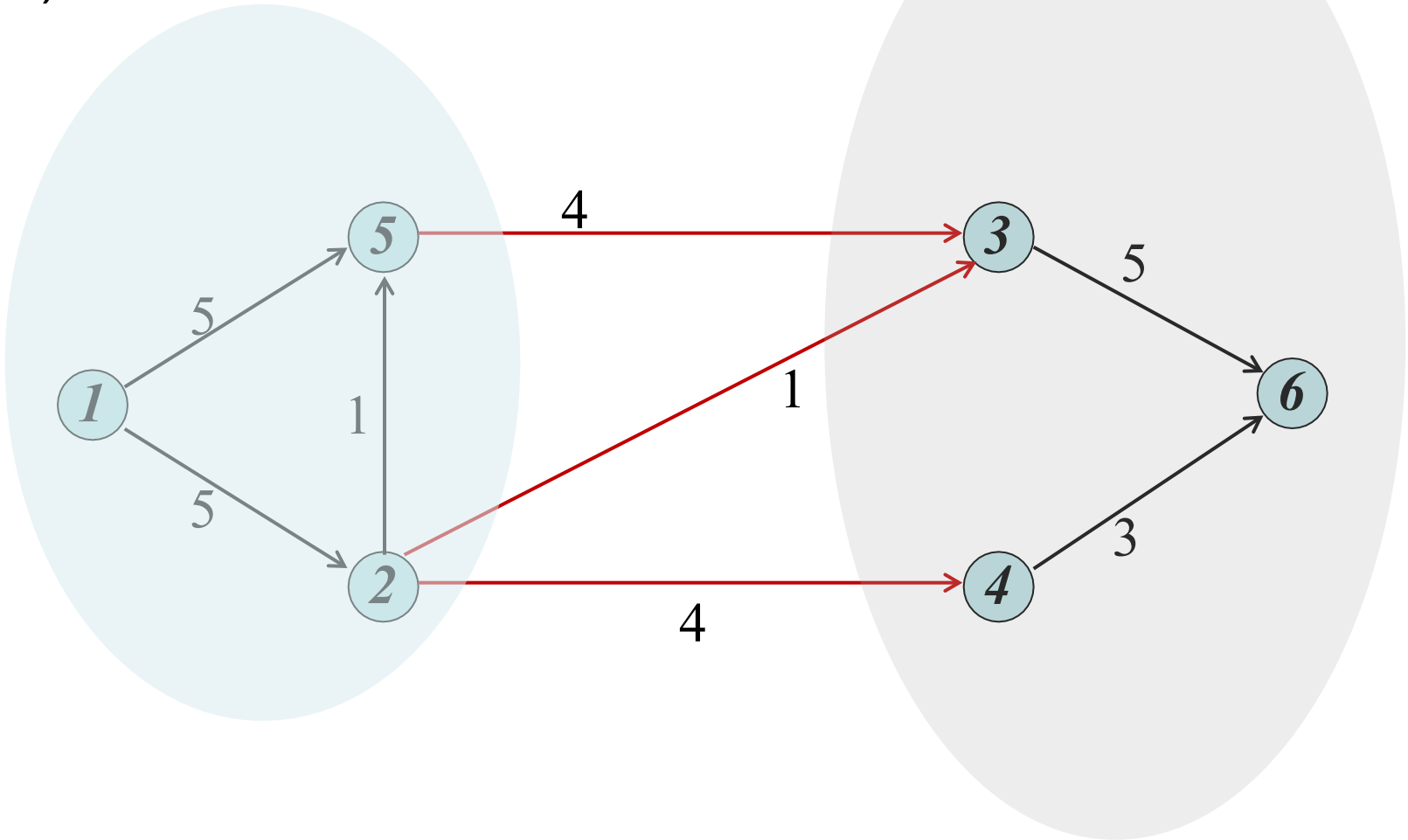




# example

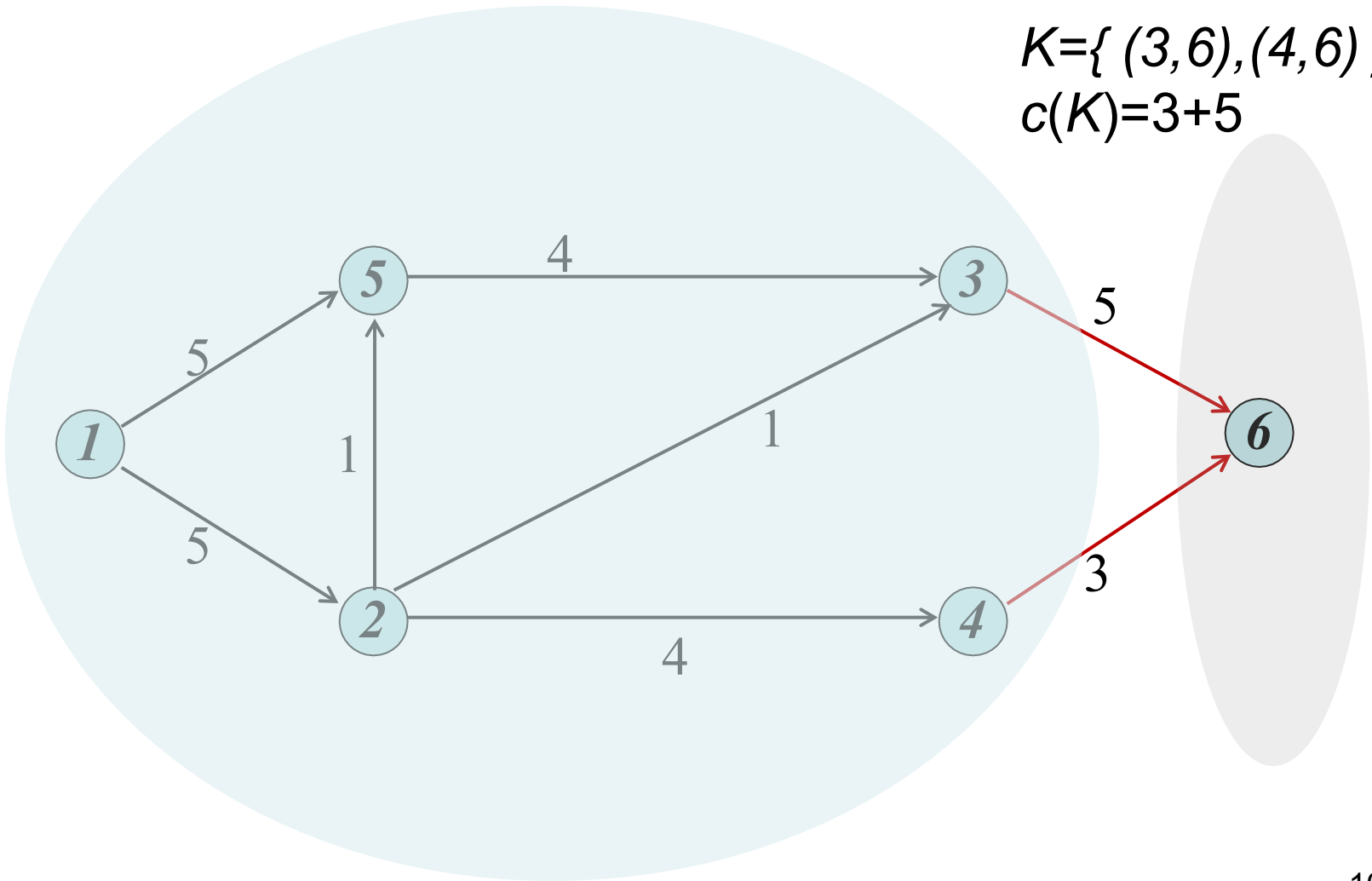
$$K = \{ (5,3), (2,3), (2,4) \}$$

$$c(K) = 4 + 1 + 4 = 9$$

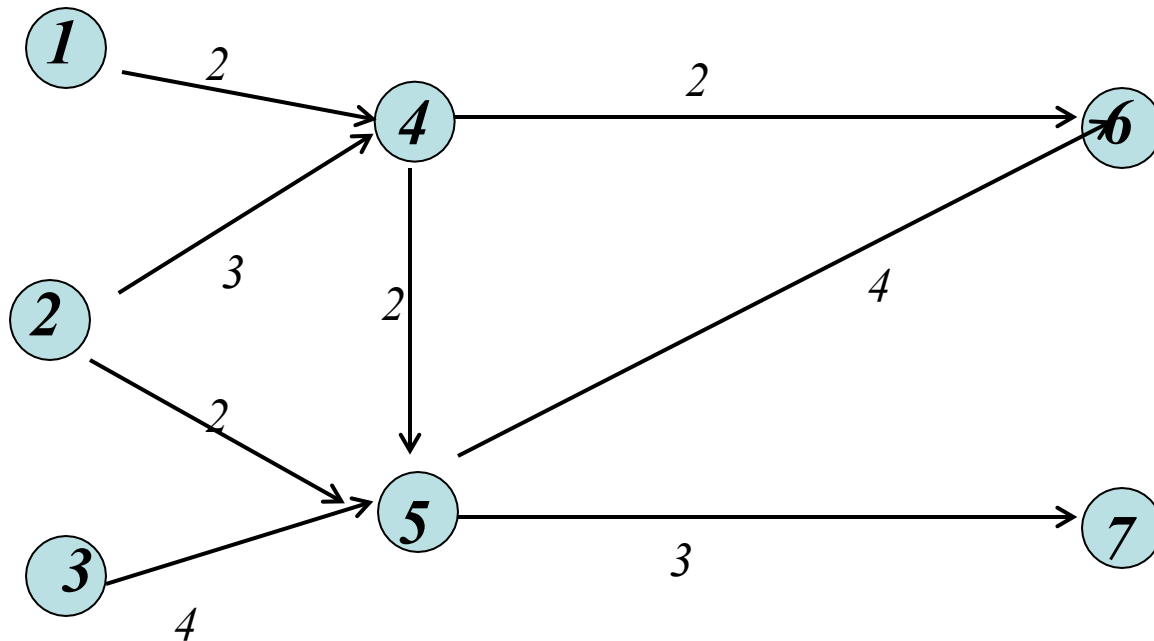


# example

$$K = \{ (3, 6), (4, 6) \}$$
$$c(K) = 3 + 5$$



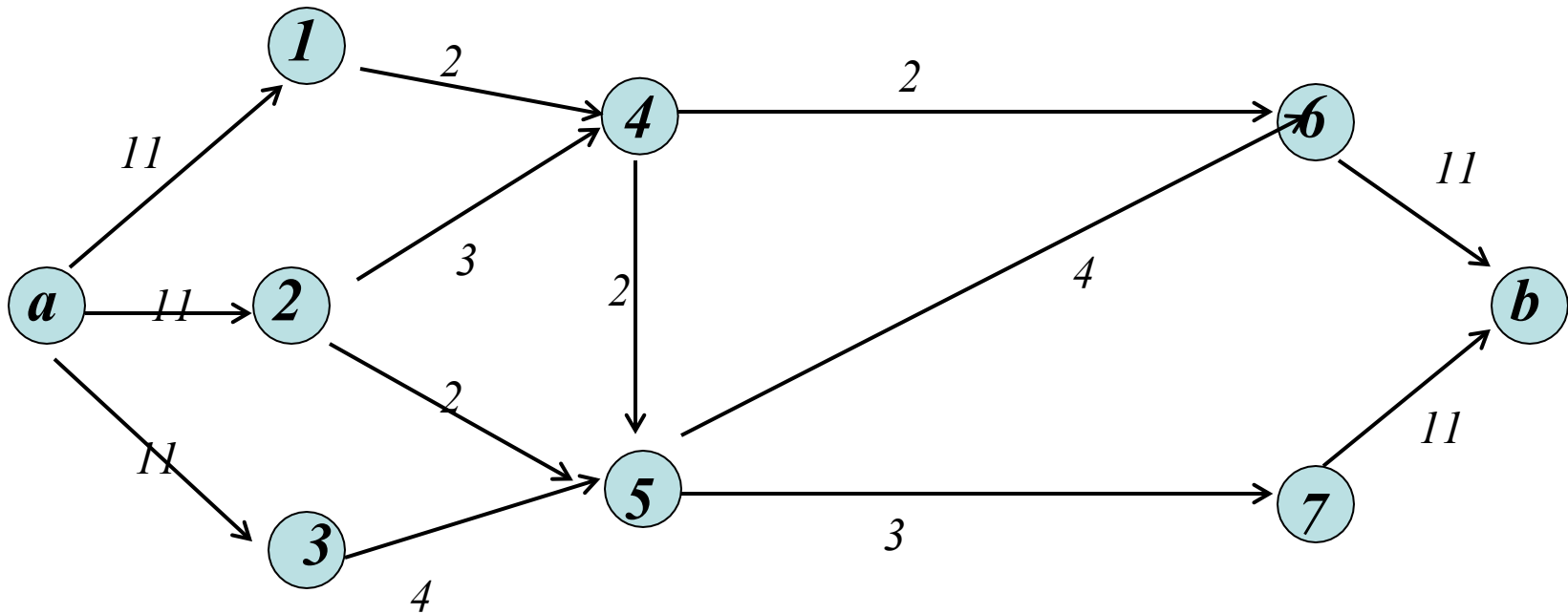
# Example

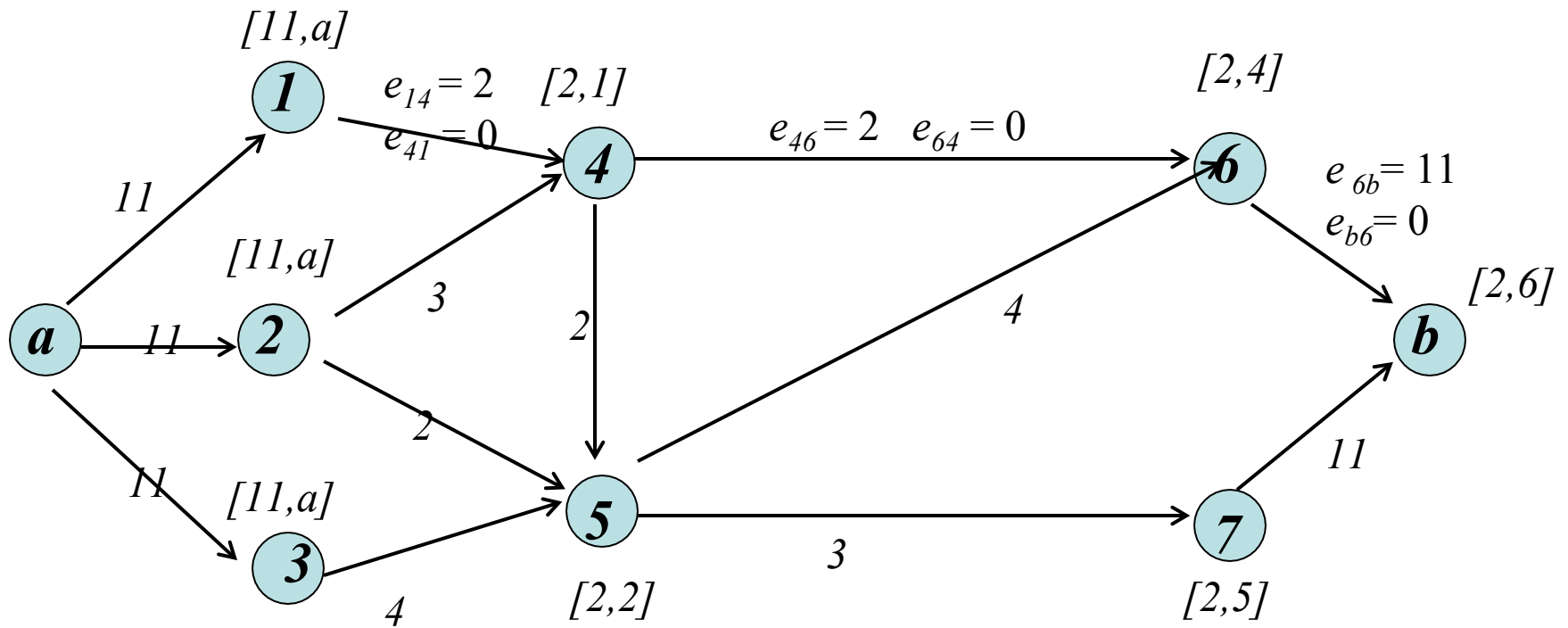


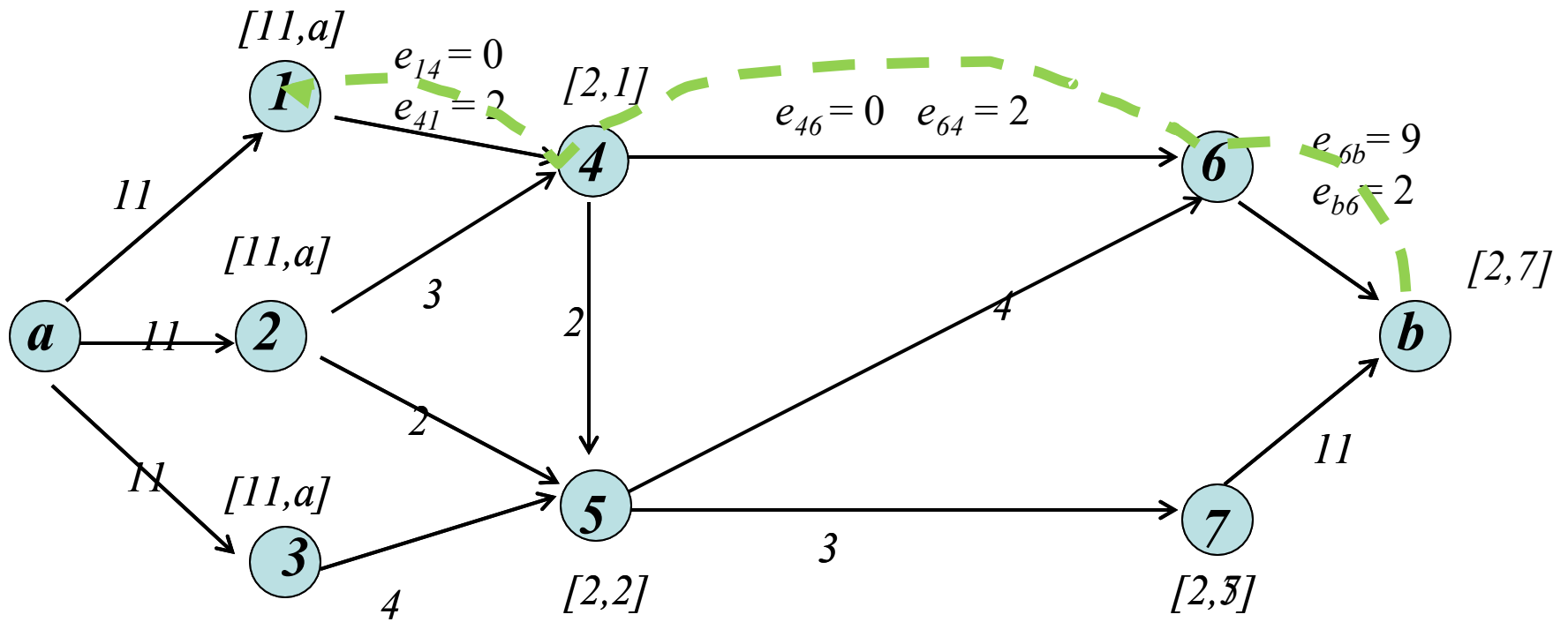
# Basic concepts

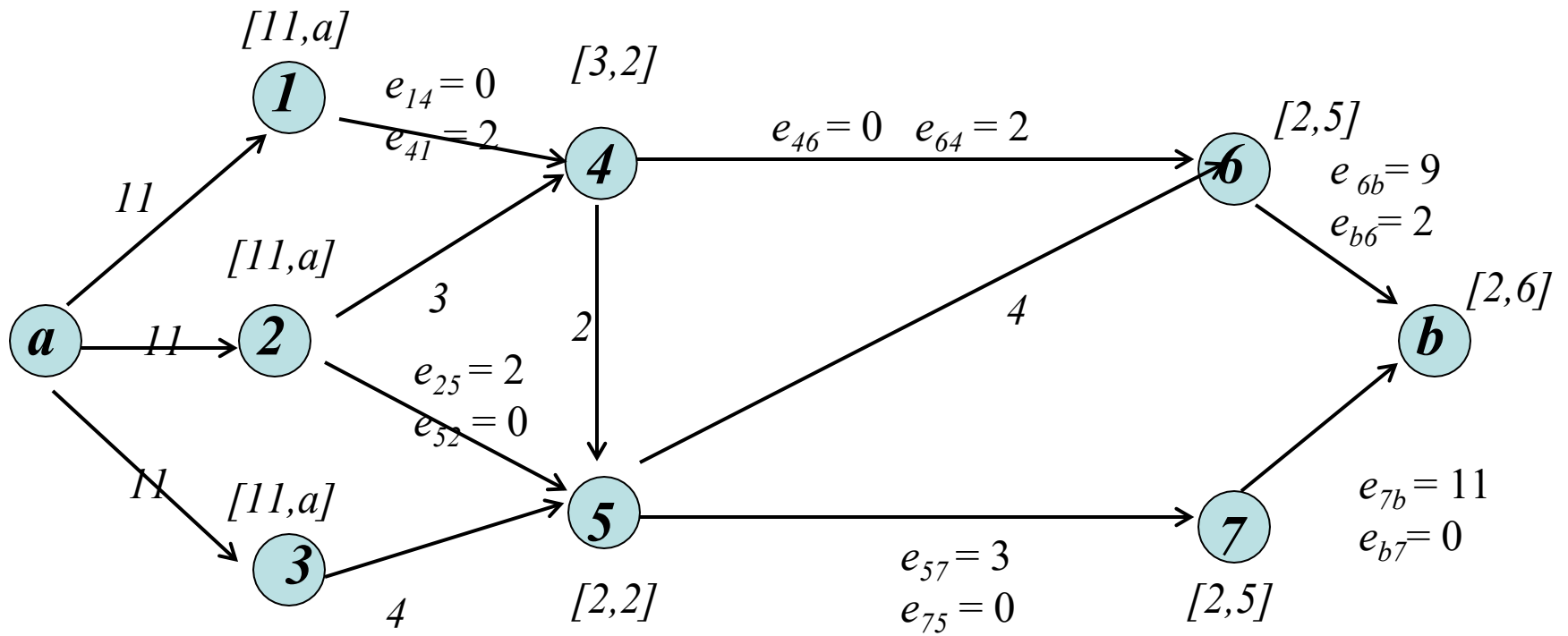
- Transport Networks      digraphs whose edges labeled with capacities.
- Source
- Sink
- Super source(超源)
- Super sink(超宿(汇))
- Flows      conservation of flow ( $C_{ij}, F_{ij}$ )
- Maximum flows      virtual flow

# Supersource    supersink

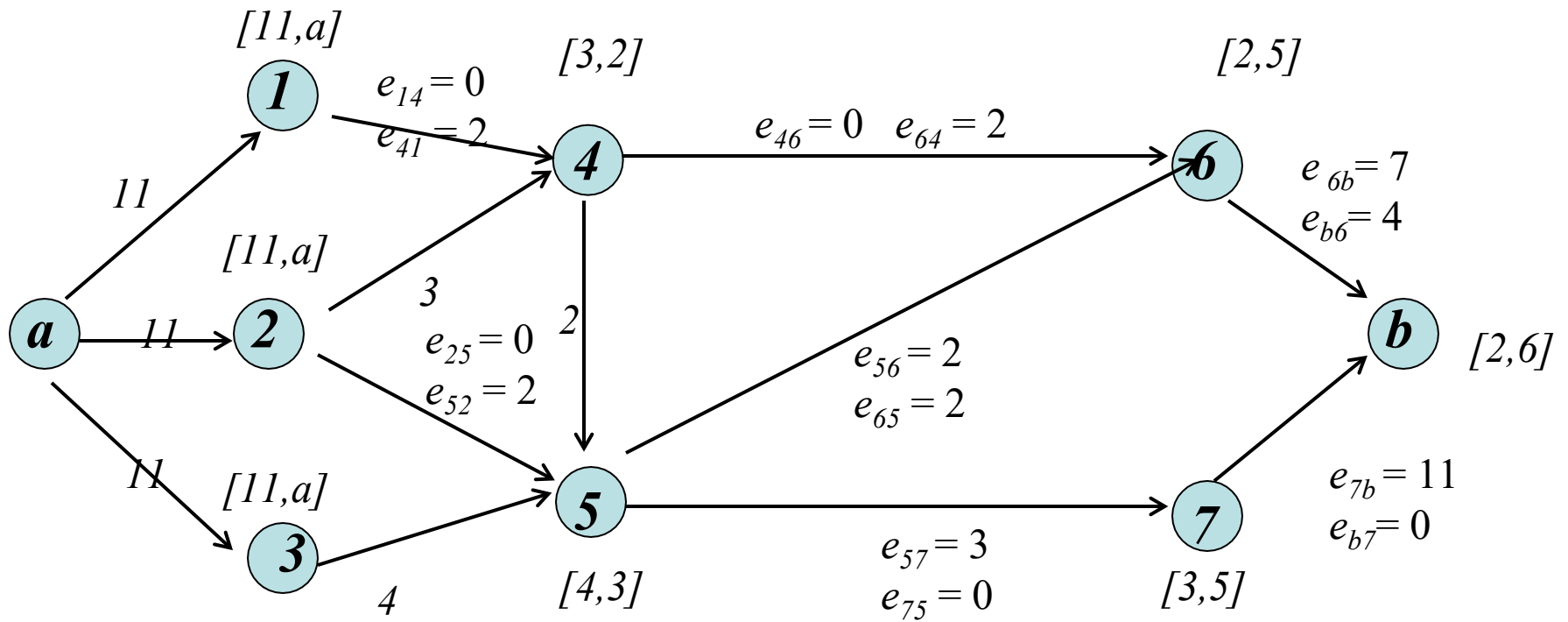


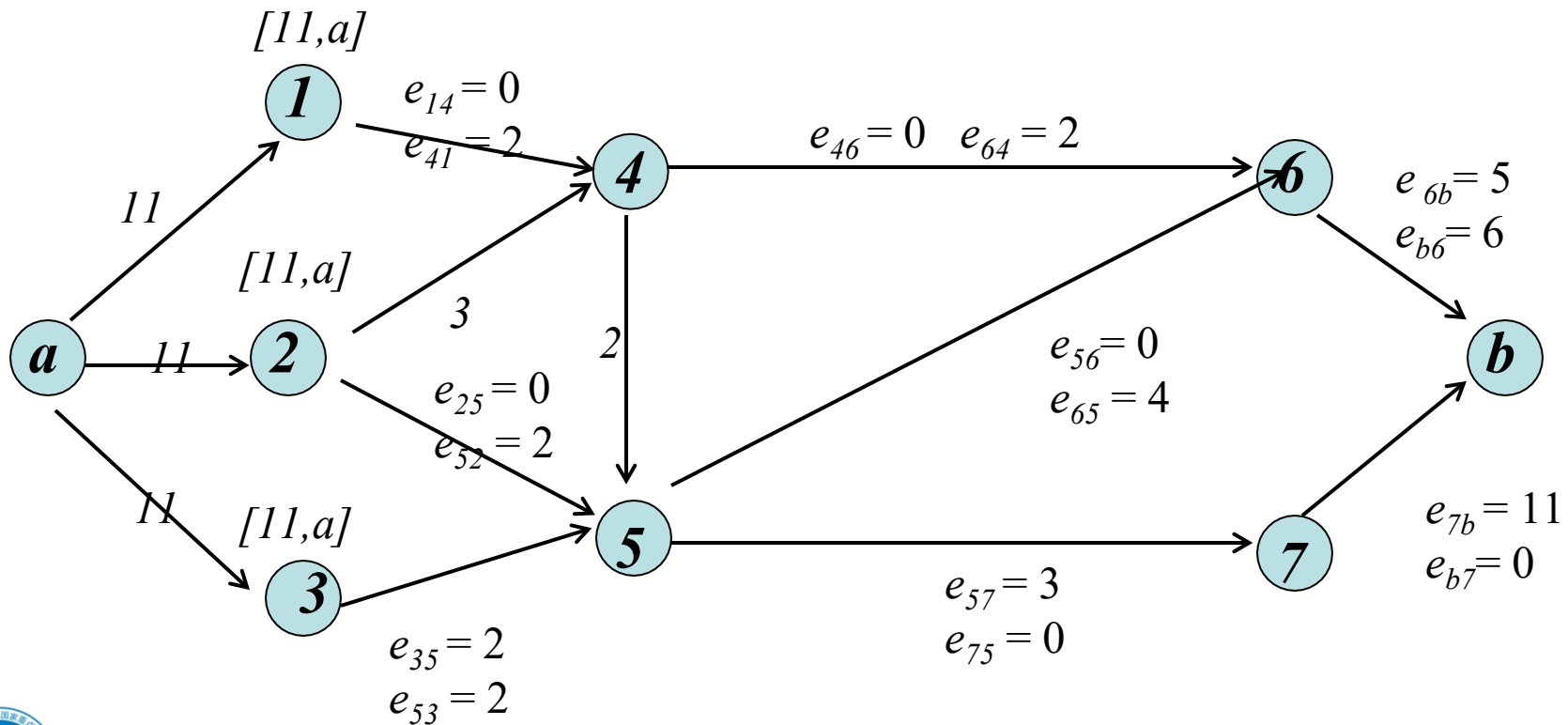


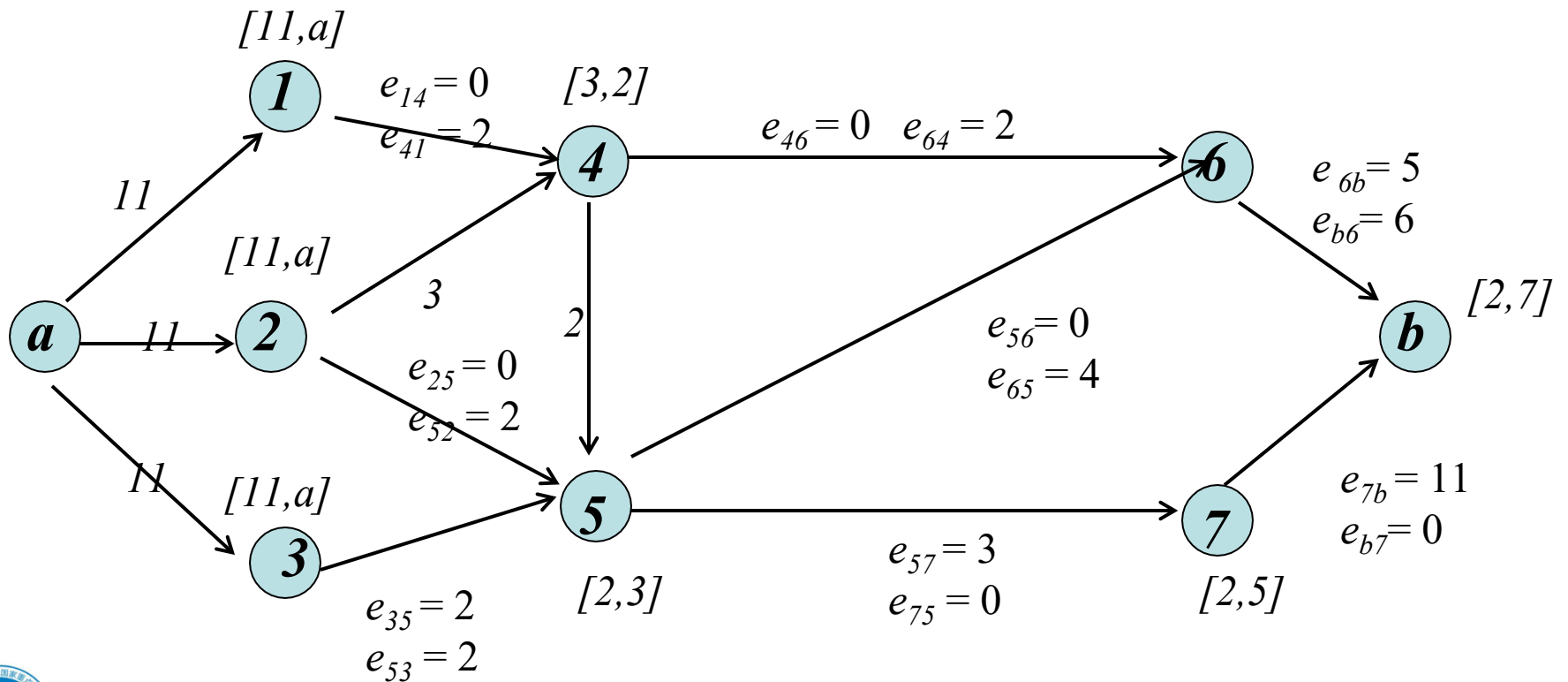


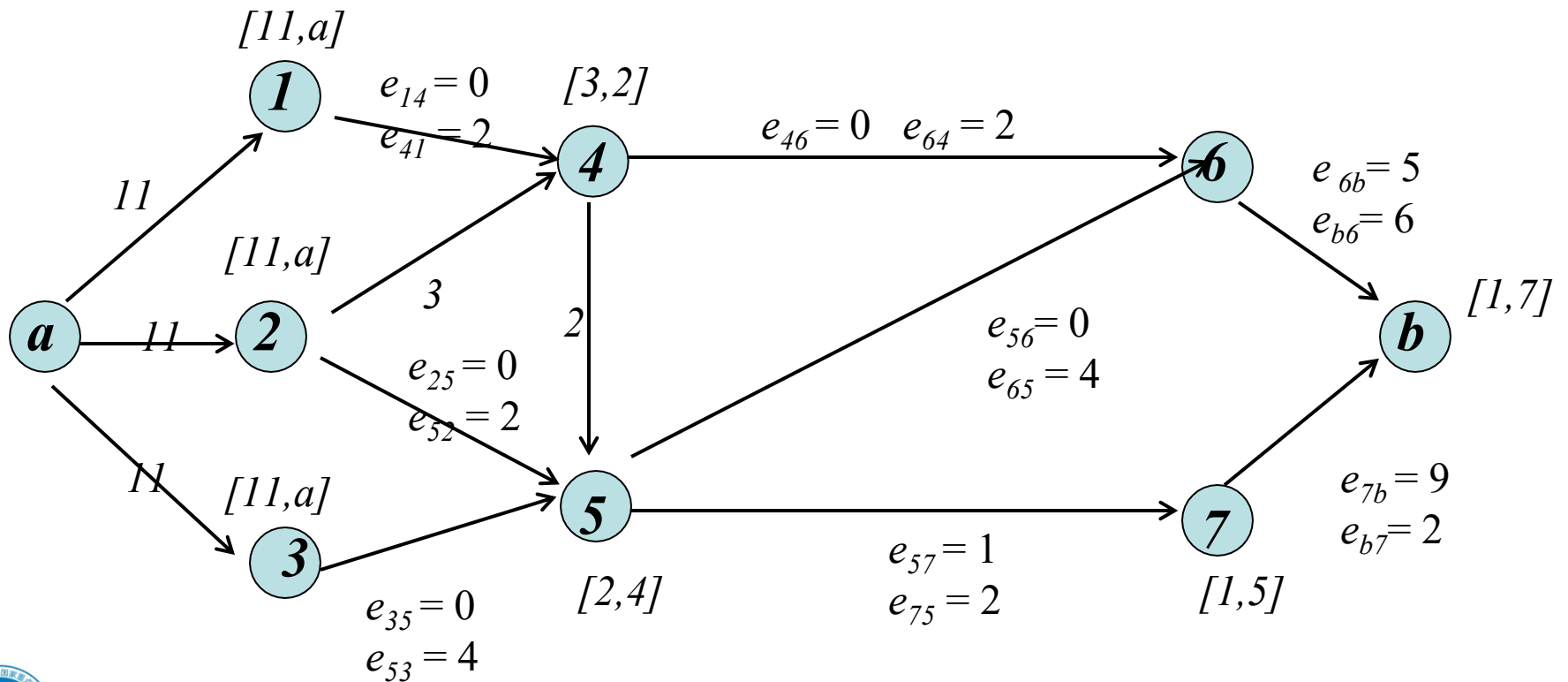


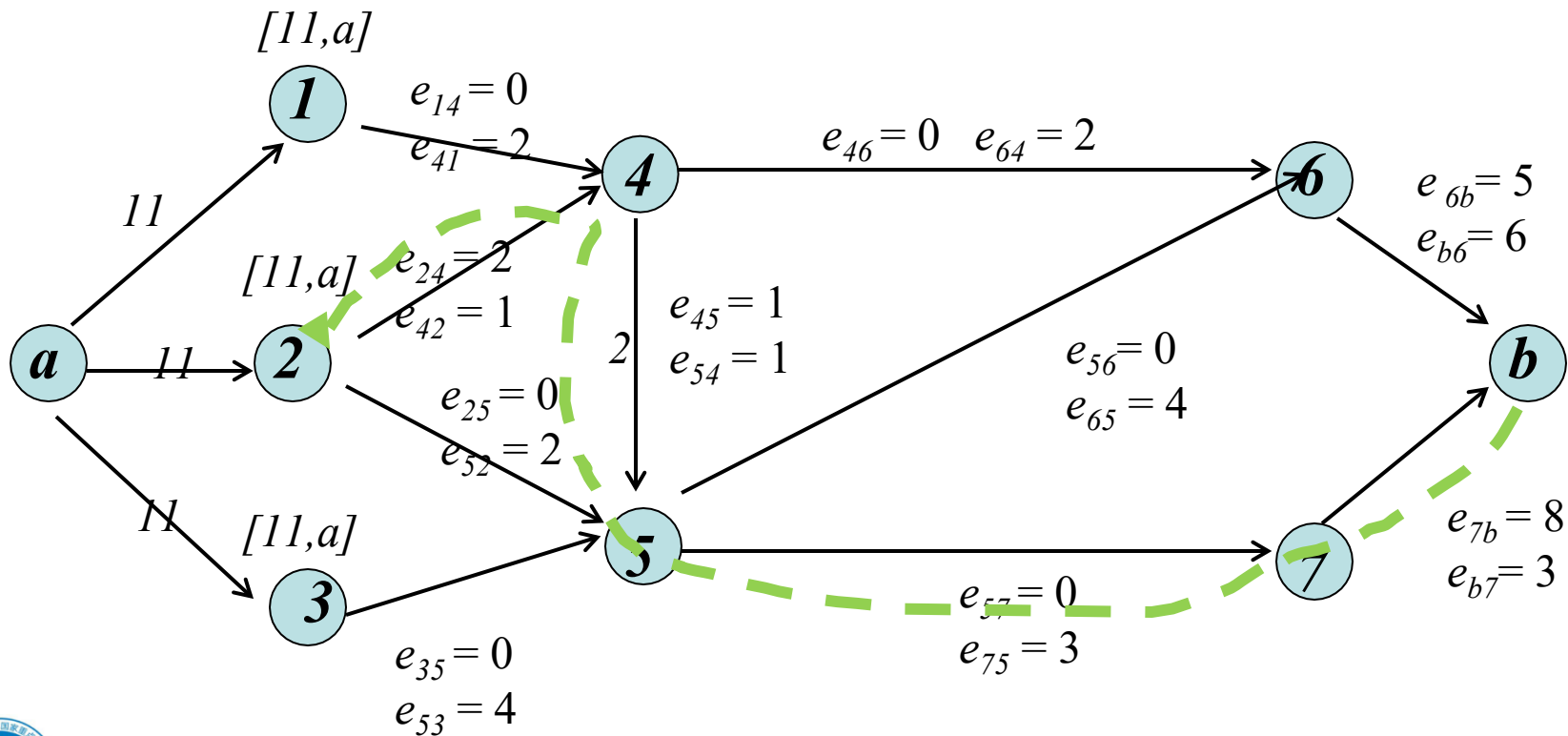




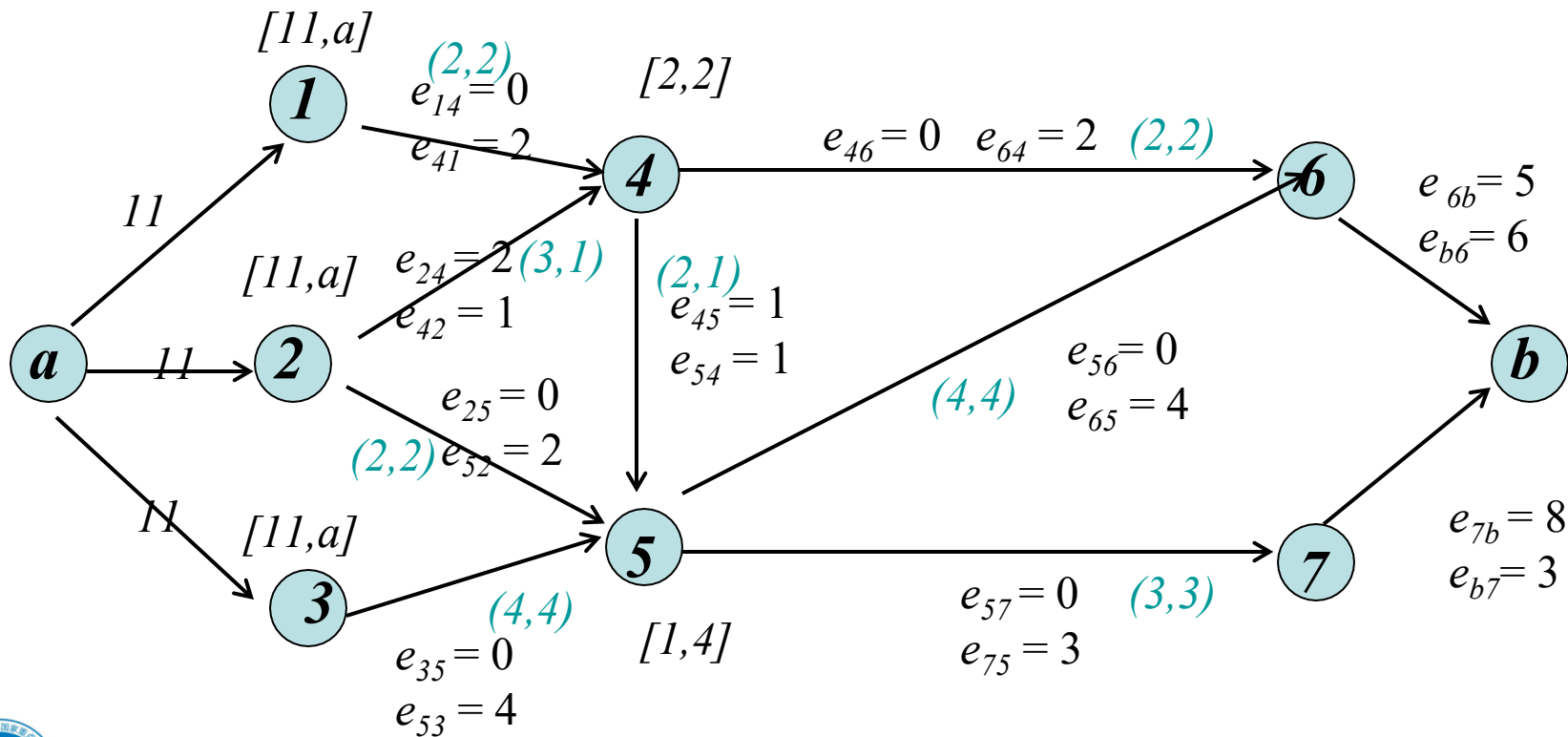








# stop



Find the maximal flow for the network N given in Figure 1.

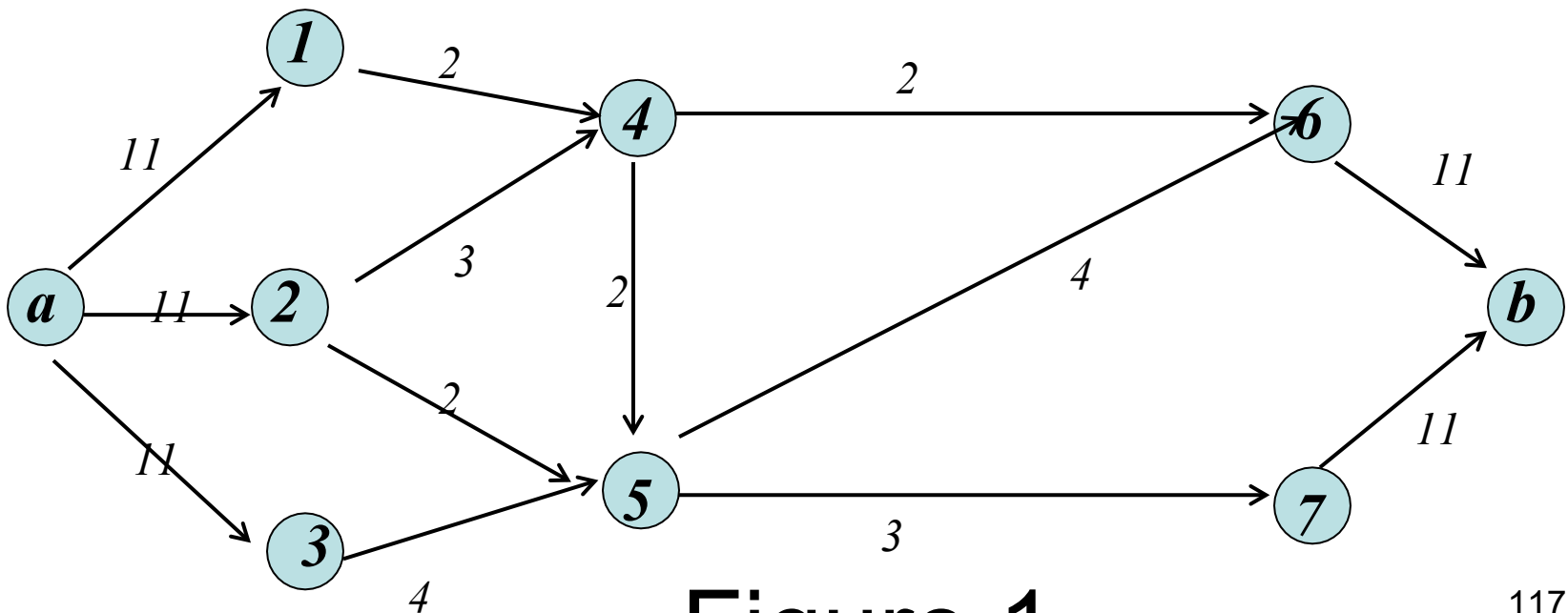


Figure 1

# 作业

- §10.8 10, 16, 20, 23, 34
- Transport network: §8.4 10, 14, 20