

# 数学建模算法与应用

## 第4章 图与网络模型及方法



## 循环比赛的名次

- $n$ 支球队循环赛，每场比赛只计胜负，没有平局。
- 根据比赛结果排出各队名次

**方法1：**寻找按箭头方向通过全部顶点的路径。

**312456**      **146325**      .....



**无法排名**

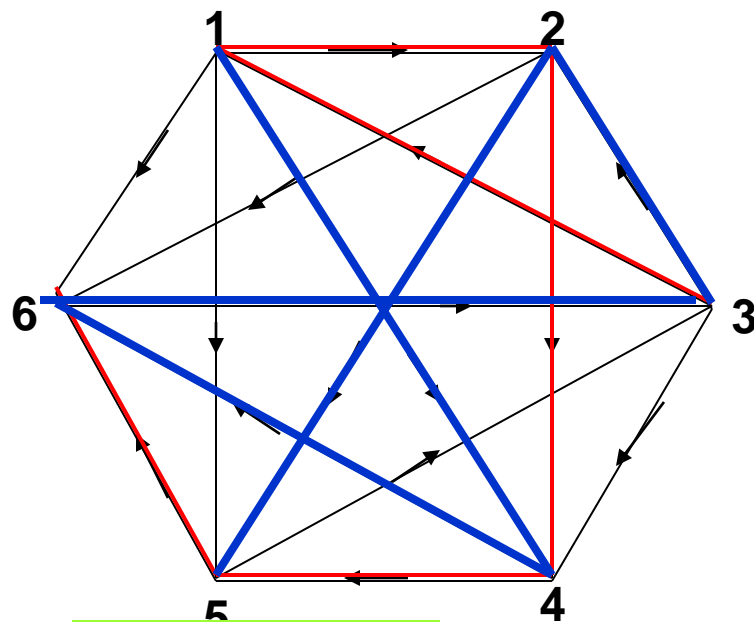
**方法2：**计算得分：1队胜4场，2, 3队各胜3场，4, 5队各胜2场，6队胜1场。2, 3队，4, 5队无法排名

**3→2, 4→5**



**排名 132456 合理吗**

## 6支球队比赛结果



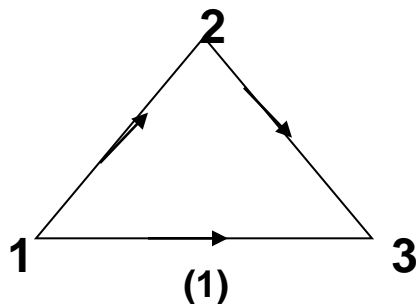


# 循环比赛的结果——竞赛图

每对顶点间都有边相连的有向图

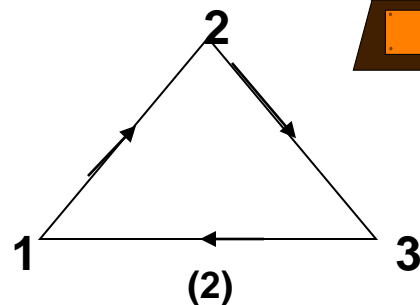


3个顶点的  
竞赛图



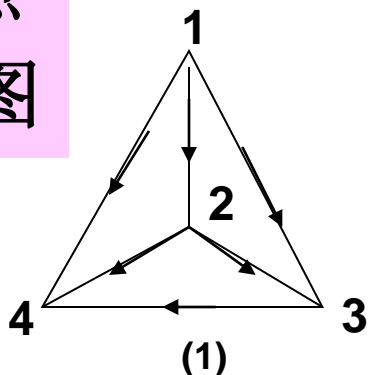
名次

{1, 2, 3}



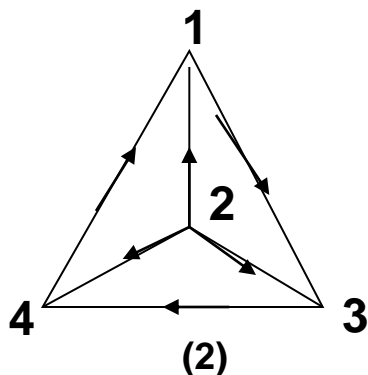
{{1,2,3}}并列

4个顶点的  
竞赛图

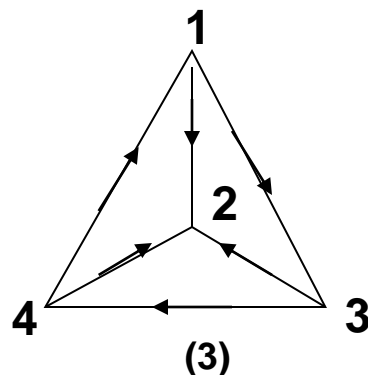


名次

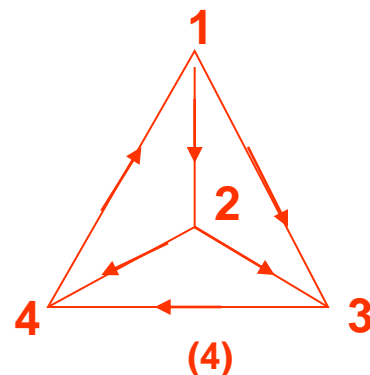
{1, 2, 3, 4}



{2, (1,3,4)}



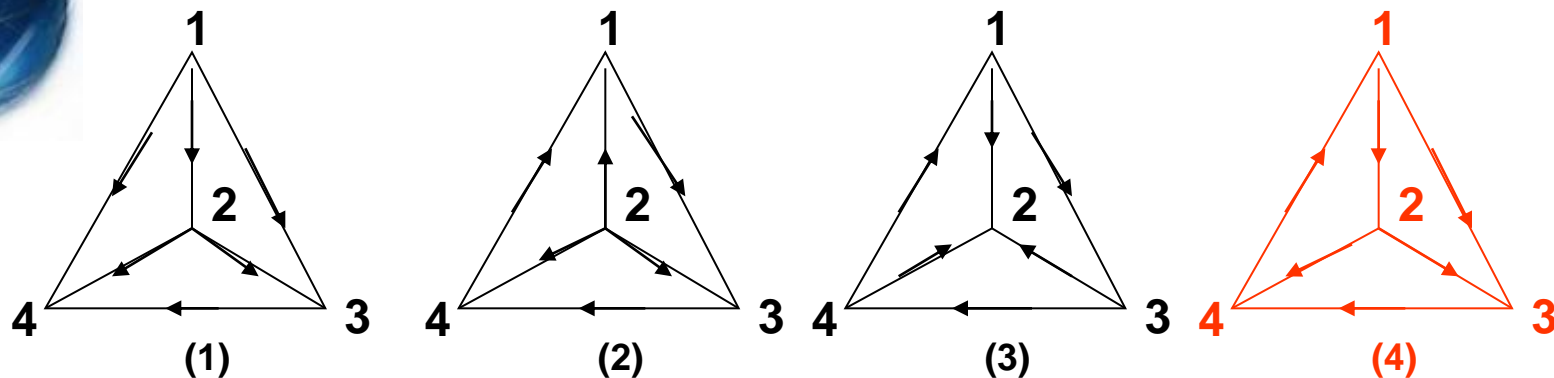
{{(1,3,4), 2}



{{(1,2),(3,4)}

{1, 2, 3, 4}?





## 竞赛图的 3种形式

- 具有唯一的完全路径，如(1)；
- **双向连通图**——任一对顶点存在两条有向路径相互连通，如(4)；
- 其他，如(2)，(3)。

## 竞赛图 的性质

- 必存在完全路径；
- 若存在唯一的完全路径，则由它确定的顶点顺序与按得分排列的顺序一致，如(1)。

# 双向连通竞赛图 $G=(V,E)$ 的名次排序

邻接矩阵

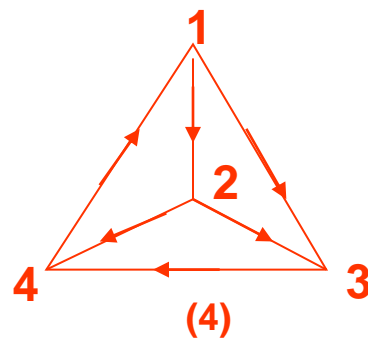
$$a_{ij} = \begin{cases} 1, & v_i v_j \in E \\ 0, & v_i v_j \notin E \end{cases}$$

得分向量

$$s = (s_1, s_2, \dots, s_n)^T$$

$$s = Ae, e = (1, 1, \dots, 1)^T$$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$



$$s^{(1)} = Ae = (2, 2, 1, 1)^T \sim \text{1级得分向量}$$

$$s^{(2)} = As^{(1)} = (3, 2, 1, 2)^T \sim \text{2级得分向量}$$

$$s^{(3)} = (3, 3, 2, 3)^T, \quad s^{(4)} = (5, 5, 3, 3)^T$$

$$s^{(5)} = (8, 6, 3, 5)^T, \quad s^{(6)} = (9, 8, 5, 8)^T$$

$$s^{(7)} = (13, 13, 8, 9)^T, \quad s^{(8)} = (21, 17, 9, 13)^T$$

$$s^{(k)} = As^{(k-1)} = A^k e$$

$$k \rightarrow \infty, s^{(k)} \rightarrow ?$$

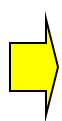
# 双向连通竞赛图的名次排序

$$s^{(k)} = As^{(k-1)} = A^k e$$

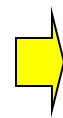
- 对于  $n(>3)$  个顶点的 双向连通竞赛图，存在正整数  $r$ ，使邻接矩阵  $A$  满足  $A^r > 0$ ， $A$  称 **素阵**

- 素阵  $A$  的最大特征根为 正单根  $\lambda$ ，对应 正特征向量  $s$ ，且

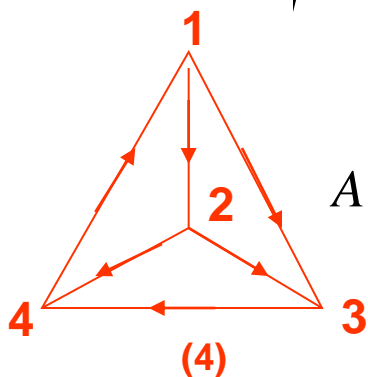
$$\lim_{k \rightarrow \infty} \frac{A^k e}{\lambda^k} = s$$



$$k \rightarrow \infty, s^{(k)} \text{ (归一化后) } \rightarrow s$$



用  $s$  排名



$\{1, 2, 3, 4\}?$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$



$$\lambda = 1.4,$$

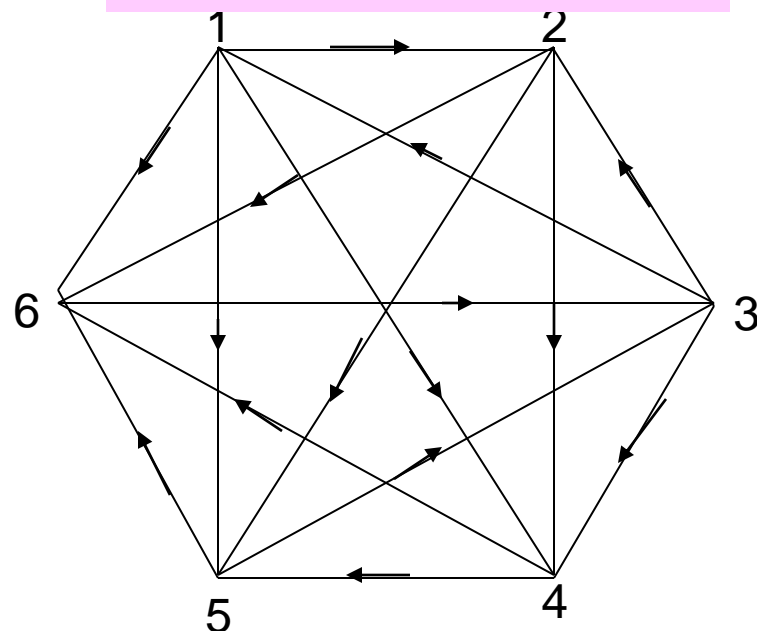
$$s = (0.323, 0.280, 0.167, 0.230)^T$$

排名为  $\{1, 2, 4, 3\}$



$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

## 6支球队比赛结果



$$s^{(1)} = (4, 3, 3, 2, 2, 1)^T,$$

$$s^{(2)} = (8, 5, 9, 3, 4, 3)^T$$

$$s^{(3)} = (15, 10, 16, 7, 12, 9)^T, \quad s^{(4)} = (38, 28, 32, 21, 25, 16)^T$$

$$\lambda = 2.232, \quad s = (0.238, 0.164, 0.231, 0.113, 0.150, 0.104)^T$$

排名次序为{1, 3, 2, 5, 4, 6}





**问题 1** 某公司在六个城市 $c_1, c_2, \dots, c_6$ 中有分公司, 从 $c_i$ 到 $c_j$ 的直接航程票价记在下述矩阵的 $(i, j)$ 位置上。 $(\infty$ 表示无直接航路), 请帮助该公司设计一张城市 $c_1$ 到其它城市间的票价最便宜的路线图。

0	50	$\infty$	40	25	10
50	0	15	20	$\infty$	25
$\infty$	15	0	10	20	$\infty$
40	20	10	0	10	25
25	$\infty$	20	10	0	55
10	25	$\infty$	25	55	0





**问题 2** 某人带狼、羊以及蔬菜渡河，一小船除需人划外，每次只能载一物过河。而人不在场时，狼要吃羊，羊要吃菜，问此人应如何过河？





**问题 3** 已知 SV 地区各城镇之间距离见表 4.4，某公司计划在 SV 地区做广告宣传，推销员从城市 1 出发，经过各个城镇，再回到城市 1。为节约开支，公司希望推销员走过这 10 个城镇的总距离最少。

表 4.4 城镇之间的距离

	2	3	4	5	6	7	8	9	10
1	8	5	9	12	14	12	16	17	22
2		9	15	17	8	11	18	14	22
3			7	9	11	7	12	12	17
4				3	17	10	7	15	18
5					8	10	6	15	15
6						9	14	8	16
7							8	6	11
8								11	11
9									10





图论起源于 18 世纪。第一篇图论论文是瑞士数学家欧拉于 1736 年发表的“哥尼斯堡的七座桥”。1847 年,克希霍夫为了给出电网络方程而引进了“树”的概念 1857 年,凯莱在计数烷 $C_nH_{2n+2}$ 的同分异构物时,也发现了“树”。哈密尔顿于 1859 年提出“周游世界”游戏,用图论的术语,就是如何找出一个连通图中的生成圈、近几十年来,由于计算机技术和科学的飞速发展,大大地促进了图论研究和应用,图论的理论和方法已经渗透到物理、化学、通讯科学、建筑学、运筹学、生物遗传学、心理学、经济学、社会学等学科中。



图论中所谓的“图”是指某类具体事物和这些事物之间的联系。如果我们用点表示这些具体事物，用连接两点的线段(直的或曲的)表示两个事物的特定的联系，就得到了描述这个“图”的几何形象。图论为任何一个包含了一种二元关系的离散系统提供了一个数学模型，借助于图论的概念、理论和方法，可以对该模型求解。哥尼斯堡七桥问题就是一个典型的例子。在哥尼斯堡有七座桥将普莱格尔河中的两个岛及岛与河岸联结起来，问题是要从这四块陆地中的任何一块开始通过每一座桥正好一次，再回到起点。

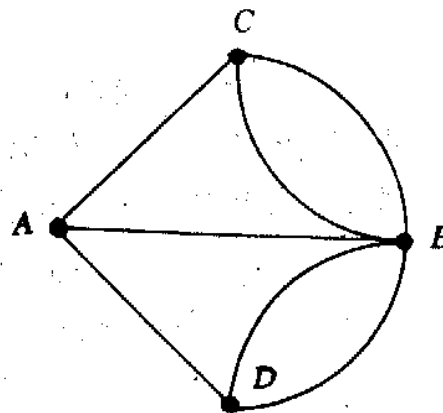
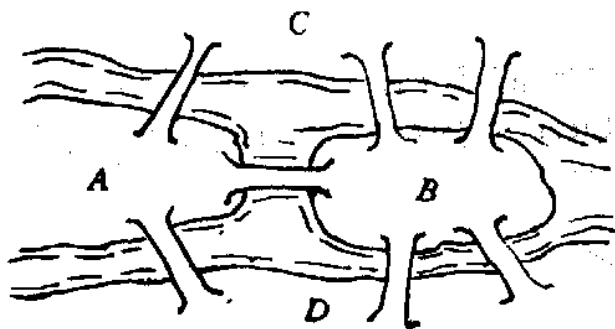


图 4.1 哥尼斯堡七桥问题



## 4.1 图的基本概念与数据结构

### 4.1.1 基本概念

所谓的图，直观地讲就是在平面上 $n$ 个点，把其中的一些点对用曲线或直线连接起来，不考虑点的位置与连线曲直长短，这样形成一个关系结构就是一个图。记成 $G=(V,E)$ ， $V$ 是以上述点为元素的顶点集， $E$ 是以上述连线为元素的边集。

如果各条边都加上方向，则称为有向图，否则称为无向图。如果有的边有方向，有的边无方向，则称为混合图。

如果任两定点间最多有一条边，且每条边的两个端点皆不重合的图，称为简单图。





如果图的两顶点间有边相连，则称此两顶点相邻，每一对顶点都相邻的图称为完全图，否则称为非完全图，完全图记为  $K_{|V|}$ 。

若  $V = X \cup Y$ ， $X \cap Y = \Phi$ ， $|X| \cdot |Y| \neq 0$ （这里  $|X|$  表示顶点集  $X$  中元素的个数），且  $X$  中无相邻的顶点对， $Y$  中亦然，则称图  $G$  为二分图；特别地，若对任意  $u \in X$ ， $u$  与  $Y$  中每个顶点相邻，则称图  $G$  为完全二分图，记为  $K_{|X|,|Y|}$ 。





设 $v \in V$ ，是边 $e \in E$ 的端点，则称 $v$ 与 $e$ 相关联，与顶点 $v$ 关联的边数称为该顶点的度，记为 $d(v)$ ，度为奇数的顶点称为奇顶点，度为偶数的顶点称为偶顶点。可以证明  $\sum_{v \in V(G)} d(v) = 2|E|$ ，即所有顶点的度数之和是边数的两倍，且由此可知奇顶点的总数是偶数。







设  $W = v_0 e_1 v_1 e_2 \cdots e_k v_k$ , 其中  $e_i \in E, 1 \leq i \leq k, v_j \in V, 0 \leq j \leq k, e_i$  与  $v_{i-1}$  和  $v_i$  关联, 称  $W$  是图  $G$  的一条道路,  $k$  为路长,  $v_0$  为起点,  $v_k$  为终点; 各边相异的道路称为迹; 各顶点相异的道路称为轨道。若  $W$  是一轨道, 可记为  $P(v_0, v_k)$ ; 起点与终点重合的道路称为回路; 起点与终点重合的轨道称为圈, 即对轨道  $P(v_0, v_k)$ , 当  $v_0 = v_k$  时成为一圈; 图中任两顶点之间都存在道路的图, 称为连通图。图中含有所有顶点的轨道称为 Hamilton 轨, 闭的 Hamilton 轨称为 Hamilton 圈; 含有 Hamilton 圈的图称为 Hamilton 图。





称两顶点 $u, v$ 分别为起点和终点的最短轨道之长为顶点 $u, v$ 的距离；在完全二分图 $K_{|X|, |Y|}$ 中， $X$ 中两顶点之间的距离为偶数， $X$ 中的顶点与 $Y$ 中的顶点的距离为奇数。

赋权图是指每条边都有一个（或多个）非负实数对应的图，这个（些）实数称为这条边的权（每条边可以具有多个权）。赋权图在实际问题中非常有用。根据不同的实际情况，权数的含义可以各不相同。例如，可用权数代表两地之间的实际距离或行车时间，也可用权数代表某工序所需的加工时间等。





### 4.1.2 图与网络的数据结构

为了在计算机上实现网络优化的算法，首先我们必须有一种方法（即数据结构）在计算机上来描述图与网络。一般来说，算法的好坏与网络的具体表示方法，以及中间结果的操作方案是有关系的。这里我们介绍计算机上用来描述图与网络的 2 种主要表示方法：邻接矩阵表示法和稀疏矩阵表示法。在下面数据结构讨论中，首先假设  $G=(V,E)$  是一个简单无向图，顶点集合  $V=\{v_1,\cdots,v_n\}$ ，边集  $E=\{e_1,\cdots,e_m\}$ ，记  $|V|=n$ ， $|E|=m$ 。





## 1. 邻接矩阵表示法

邻接矩阵是表示顶点之间相邻关系的矩阵，邻接矩阵记作  $W = (w_{ij})_{n \times n}$ ，当  $G$  为赋权图时

$$w_{ij} = \begin{cases} \text{权值, 当 } v_i \text{ 与 } v_j \text{ 之间有边时,} \\ 0 \text{ 或 } \infty, \text{ 当 } v_i \text{ 与 } v_j \text{ 之间无边时.} \end{cases}$$

当  $G$  为非赋权图时，

$$w_{ij} = \begin{cases} 1, \text{ 当 } v_i \text{ 与 } v_j \text{ 之间有边时,} \\ 0, \text{ 当 } v_i \text{ 与 } v_j \text{ 之间无边时.} \end{cases}$$

采用邻接矩阵表示图，直观方便，通过查邻接矩阵元素的值可以很容易地查找图中任两个顶点  $v_i$  和  $v_j$  之间有无边，以及边上的权值。当图的边数  $m$  远小于顶点数  $n$  时，邻接矩阵表示法会造成很大的空间浪费。





## 2. 稀疏矩阵表示法

稀疏矩阵是指矩阵中零元素很多，非零元素很少的矩阵。对于稀疏矩阵，只要存放非零元素的行标、列标、非零元素的值即可，可以按如下方式存储（非零元素的行地址，非零元素的列地址），非零元素的值。

在 Matlab 中无向图和有向图邻接矩阵的使用上有很大的差异。

对于有向图，只要写出邻接矩阵，直接使用 Matlab 的命令 `sparse` 命令，就可以把邻接矩阵转化为稀疏矩阵的表示方式。





对于无向图，由于邻接矩阵是对称阵，Matlab 中只需使用邻接矩阵的下三角元素，即 Matlab 只存储邻接矩阵下三角元素中的非零元素。

稀疏矩阵只是一种存储格式。Matlab 中，普通矩阵使用 `sparse` 命令变成稀疏矩阵，稀疏矩阵使用 `full` 命令变成普通矩阵。





## 4.2 最短路问题

### 4.2.1 两个指定顶点之间的最短路径

问题如下，给出了一个连接若干个城镇的铁路网络在这个网络的两个指定城镇间，找一条最短铁路线。

构造赋权图  $G=(V,E,W)$ ，其中顶点集  $V=\{v_1,\cdots,v_n\}$ ，这里  $v_1,\cdots,v_n$  表示各个小城镇， $E$  为边的集合，邻接矩阵  $W=(w_{ij})_{n\times n}$ ，这里  $w_{ij}$  表示顶点  $v_i$  和  $v_j$  之间直通铁路的距离，若顶点  $v_i$  和  $v_j$  之间无铁路，则  $w_{ij}=\infty$ 。问题就是求赋权图  $G$  中指定的两个顶点  $u_0,v_0$  间的具有最小权的路。这条路叫做  $u_0,v_0$  间的最短路，它的权叫做  $u_0,v_0$  间的距离，亦记作  $d(u_0,v_0)$ 。



求最短路已有成熟的算法,如**迪克斯特拉**(Dijkstra)算法,其基本思想是按距 $u_0$ 从近到远为顺序,依次求得 $u_0$ 到 $G$ 的各顶点的最短路和距离,直至 $v_0$ (或直至 $G$ 的所有顶点),算法结束。为避免重复并保留每一步的计算信息,采用了标号算法。下面是该算法。







(1) 令 $l(u_0) = 0$ , 对 $v \neq u_0$ , 令 $l(v) = \infty$ ,  $S_0 = \{u_0\}$ ,  $i = 0$ 。

(2) 对每个 $v \in \bar{S}_i$  ( $\bar{S}_i = V \setminus S_i$ ), 用

$$\min_{u \in S_i} \{l(v), l(u) + w(uv)\}$$

代替 $l(v)$ , 这里 $w(uv)$ 表示顶点 $u$ 和 $v$ 之间边的权值。计算 $\min_{v \in \bar{S}_i} \{l(v)\}$ , 把达到这个最小值的一个顶点记为 $u_{i+1}$ , 令

$$S_{i+1} = S_i \cup \{u_{i+1}\}.$$

(3) 若 $i = |V| - 1$ , 停止; 若 $i < |V| - 1$ , 用 $i + 1$ 代替 $i$ , 转(2)。

算法结束时, 从 $u_0$ 到各顶点 $v$ 的距离由 $v$ 的最后一次标号 $l(v)$ 给出。在 $v$ 进入 $S_i$ 之前的标号 $l(v)$ 叫 T 标号,  $v$ 进入 $S_i$ 时的标号 $l(v)$ 叫 P 标号。



**例 4.1** 某公司在六个城市 $c_1, c_2, \dots, c_6$ 中有分公司，从 $c_i$ 到 $c_j$ 的直接航程票价记在下述矩阵的 $(i, j)$ 位置上。  
( $\infty$ 表示无直接航路)，请帮助该公司设计一张城市 $c_1$ 到其它城市间的票价最便宜的路线图。

0	50	$\infty$	40	25	10
50	0	15	20	$\infty$	25
$\infty$	15	0	10	20	$\infty$
40	20	10	0	10	25
25	$\infty$	20	10	0	55
10	25	$\infty$	25	55	0



用矩阵 $a_{n \times n}$  ( $n$ 为顶点个数) 存放各边权的邻接矩阵, 行向量 $pb$ 、 $index_1$ 、 $index_2$ 、 $d$ 分别用来存放 $P$ 标号信息、标号顶点顺序、标号顶点索引、最短通路的值。其中分量

$$pb(i) = \begin{cases} 1 & \text{当第}i\text{顶点的标号已成为}P\text{标号;} \\ 0 & \text{当第}i\text{顶点的标号未成为}P\text{标号;} \end{cases}$$

$index_2(i)$  存放始点到第 $i$ 顶点最短通路中第 $i$ 顶点前一顶点的序号;

$d(i)$  存放由始点到第 $i$ 顶点最短通路的值。

求得 $c_1$ 到 $c_2, \dots, c_6$ 的最便宜票价分别为35, 45, 35, 25, 10。





## 4.2.2 两个指定顶点之间最短路问题的数学规划模型

假设有向图有 $n$ 个顶点，现要求从顶点 $v_1$ 到顶点 $v_n$ 的最短路。设 $W = (w_{ij})_{n \times n}$ 为邻接矩阵，其分量为

$$w_{ij} = \begin{cases} \text{边 } v_i v_j \text{ 的权值, } v_i v_j \in E, \\ \infty, & \text{其它,} \end{cases}$$

决策变量为 $x_{ij}$ ，当 $x_{ij} = 1$ ，说明弧 $v_i v_j$ 位于顶点 $v_1$ 至顶点 $v_n$ 的最短路上；否则 $x_{ij} = 0$ 。其数学规划表达式为

$$\begin{aligned} \min \quad & \sum_{v_i v_j \in E} w_{ij} x_{ij}, \\ \text{s.t.} \quad & \sum_{\substack{j=1 \\ v_i v_j \in E}}^n x_{ij} - \sum_{\substack{j=1 \\ v_j v_i \in E}}^n x_{ji} = \begin{cases} 1, & i = 1, \\ -1, & i = n, \\ 0, & i \neq 1, n, \end{cases} \\ & x_{ij} = 0 \text{ 或 } 1. \end{aligned}$$



**例 4.2** 在图 4.2 中，用点表示城市，现有  $A, B_1, B_2, C_1, C_2, C_3, D$  共 7 个城市。点与点之间的连线表示城市间有道路相连。连线旁的数字表示道路的长度。现计划从城市  $A$  到城市  $D$  铺设一条天然气管道，请设计出最小长度管道铺设方案。

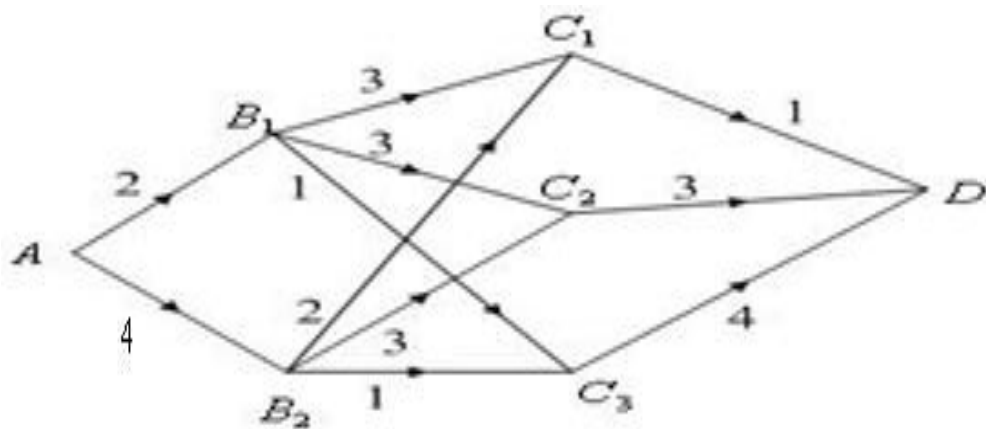


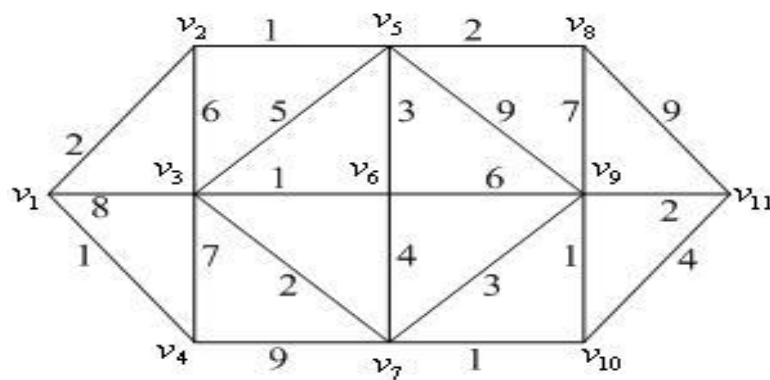
图 4.2 7 个城市间的连线图

求得最短铺设方案是铺设  $AB_1, B_1C_1, C_1D$  段，最短铺设长度为 6。



**例 4.3**（无向图的最短路问题）求图 4.3 中 $v_1$ 到 $v_{11}$ 的最短路。

**分析** 例 4.2 处理的问题属于有向图的最短路问题，本例是处理无向图的最短路问题，在处理方式上与有向图的最短路问题有一些差别，这里选择赋权邻接矩阵的方法编写 LINGO 程序。



**图 4.3** 赋权无向图



与有向图相比较，在程序中只增加了一个语句  
 $\text{@sum(cities(j):x(j,1))=0}$ ，即从顶点 1 离开后，再不能回到该顶点。

求得的最短路径为  $1 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 3 \rightarrow 7 \rightarrow 10 \rightarrow 9 \rightarrow 11$ ，最短路径长度为 13。





### 4.2.3 每对顶点之间的最短路径

计算赋权图中各对顶点之间最短路径，显然可以调用 Dijkstra 算法。具体方法是：每次以不同的顶点作为起点，用 Dijkstra 算法求出从该起点到其余顶点的最短路径，反复执行  $n-1$  次这样的操作，就可得到从每一个顶点到其它顶点的最短路径。这种算法的时间复杂度为  $O(n^3)$ 。第二种解决这一问题的方法是由 Floyd, R. W. 提出的算法，称之为 Floyd 算法。







对于赋权图  $G = (V, E, A_0)$ , 其中顶点集  $V = \{v_1, \dots, v_n\}$

邻接矩阵

$$A_0 = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix},$$

这里

$$a_{ij} = \begin{cases} \text{权值, 当 } v_i \text{ 与 } v_j \text{ 之间有边时,} \\ \infty, \text{ 当 } v_i \text{ 与 } v_j \text{ 之间无边时,} \end{cases} \quad (i \neq j)$$

$$a_{ii} = 0, \quad i = 1, 2, \dots, n.$$

对于无向图,  $A_0$  是对称矩阵,  $a_{ij} = a_{ji}$ 。





Floyd 算法的基本思想是递推产生一个矩阵序列  $A_1, \dots, A_k, \dots, A_n$ , 其中矩阵  $A_k$  的第  $i$  行第  $j$  列元素  $A_k(i, j)$  表示从顶点  $v_i$  到顶点  $v_j$  的路径上所经过的顶点序号不大于  $k$  的最短路径长度。

计算时用迭代公式

$$A_k(i, j) = \min(A_{k-1}(i, j), A_{k-1}(i, k) + A_{k-1}(k, j)),$$

$k$  是迭代次数,  $i, j, k = 1, 2, \dots, n$ 。

最后, 当  $k = n$  时,  $A_n$  即是各顶点之间的最短通路值。





## 例4.4 用Floyd算法求解例4.1。





## 4.3 最小生成树问题

### 4.3.1 基本概念

连通的无圈图叫做树，记之为 $T$ ；其度为 1 的顶点称为叶子顶点；显然有边的树至少有两个叶子顶点。

若图  $G = (V(G), E(G))$  和树  $T = (V(T), E(T))$  满足  $V(G) = V(T)$ ,  $E(T) \subset E(G)$ , 则称 $T$ 是 $G$ 的生成树。图 $G$ 连通的充分必要条件为 $G$ 有生成树，一个连通图的生成树的个数很多。





树有下面常用的五个充要条件。

**定理 4.1** (1)  $G = (V, E)$  是树当且仅当  $G$  中任二顶点之间有且仅有一条轨道。

(2)  $G$  是树当且仅当  $G$  无圈, 且  $|E| = |V| - 1$ 。

(3)  $G$  是树当且仅当  $G$  连通, 且  $|E| = |V| - 1$ 。

(4)  $G$  是树当且仅当  $G$  连通, 且  $\forall e \in E, G - e$  不连通。

(5)  $G$  是树当且仅当  $G$  无圈,  $\forall e \notin E, G + e$  恰有一个圈。





### 4.3.2 最小生成树

欲修筑连接 $n$ 个城市的铁路，已知 $i$ 城与 $j$ 城之间的铁路造价为 $c_{ij}$ ，设计一个线路图，使总造价最低。

上述问题的数学模型是在连通赋权图上求权最小的生成树。赋权图的具最小权的生成树叫做最小生成树。



### 4.3.2.1 prim 算法构造最小生成树

构造连通赋权图  $G = (V, E, W)$  的最小生成树，设置两个集合  $P$  和  $Q$ ，其中  $P$  用于存放  $G$  的最小生成树中的顶点，集合  $Q$  存放  $G$  的最小生成树中的边。令集合  $P$  的初值为  $P = \{v_1\}$ （假设构造最小生成树时，从顶点  $v_1$  出发），集合  $Q$  的初值为  $Q = \Phi$ （空集）。prim 算法的思想是，从所有  $p \in P$   $v \in V - P$  的边中，选取具有最小权值的边  $pv$ ，将顶点  $v$  加入集合  $P$  中，将边  $pv$  加入集合  $Q$  中，如此不断重复，直到  $P = V$  时，最小生成树构造完毕，这时集合  $Q$  中包含了最小生成树的所有边。



prim 算法如下

(1)  $P = \{v_1\}$ ,  $Q = \Phi$ ;

(2) while  $P \neq V$

找最小边  $pv$ , 其中  $p \in P, v \in V - P$ ;

$P = P + \{v\}$ ;

$Q = Q + \{pv\}$ ;

end





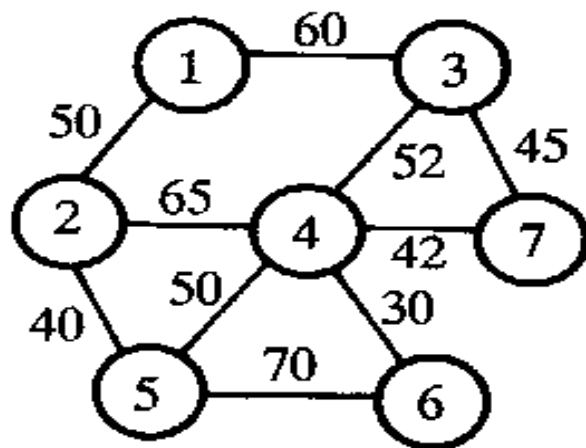


图 4.4 最小生成树问题

例 4.5 用 prim 算法求图 4.4 的最小生成树。

求得最小生成树的边集为  $\{v_1v_2, v_2v_5, v_5v_4, v_4v_6, v_4v_7, v_7v_3\}$ 。



### 4.3.2.2 Kruskal 算法构造最小生成树

科茹斯克尔 (Kruskal) 算法是一个好算法。Kruskal 算法如下

- (1) 选  $e_1 \in E(G)$ , 使得  $e_1$  是权值最小的边。
- (2) 若  $e_1, e_2, \dots, e_i$  已选好, 则从  $E(G) - \{e_1, e_2, \dots, e_i\}$  中选取  $e_{i+1}$ , 使得
  - i)  $\{e_1, e_2, \dots, e_i, e_{i+1}\}$  中无圈, 且
  - ii)  $e_{i+1}$  是  $E(G) - \{e_1, e_2, \dots, e_i\}$  中权值最小的边。
- (3) 直到选得  $e_{|V|-1}$  为止。





**例 4.6** 用 Kruskal 算法构造例 4.5 的最小生成树。

用  $index_{2 \times n}$  存放各边端点的信息，当选中某一边之后，就将此边对应的顶点序号中较大序号  $u$  改为此边的另一序号  $v$ ，同时把后面边中所有序号为  $u$  的改为  $v$ 。此方法的几何意义是将序号  $u$  的这个顶点收缩到  $v$  顶点， $u$  顶点不复存在。后面继续寻查时，发现某边的两个顶点序号相同时，认为已被收缩掉，失去了被选取的资格。

求解结果和例 4.5 相同。





## 4.4 网络最大流问题

### 4.4.1 基本概念与基本定理

#### 1. 网络与流

**定义 4.1** 给一个有向图  $D = (V, A)$ , 其中  $A$  为弧集, 在  $V$  中指定了一点, 称为发点 (记为  $v_s$ ), 和另一点, 称为收点 (记为  $v_t$ ), 其余的点叫中间点, 对于每一个弧  $(v_i, v_j) \in A$ , 对应有一个  $c(v_i, v_j) \geq 0$  (或简写为  $c_{ij}$ ), 称为弧的容量。通常我们就把这样的有向图  $D$  叫作一个网络, 记作  $D = (V, A, C)$ , 其中  $C = \{c_{ij}\}$ 。

所谓网络上的流, 是指定义在弧集合  $A$  上的一个函数  $f = \{f_{ij}\} = \{f(v_i, v_j)\}$ , 并称  $f_{ij}$  为弧  $(v_i, v_j)$  上的流量。



## 2. 可行流与最大流

**定义 4.2** 满足下列条件的流  $f$  称为可行流

(1) 容量限制条件：对每一弧  $(v_i, v_j) \in A$ ,  $0 \leq f_{ij} \leq c_{ij}$ ;

(2) 平衡条件

对于中间点，流出量=流入量，即对于每个  $i (i \neq s, t)$  有

$$\sum_{j:(v_i, v_j) \in A} f_{ij} - \sum_{j:(v_j, v_i) \in A} f_{ji} = 0,$$

对于发点  $v_s$ ，记

$$\sum_{(v_s, v_j) \in A} f_{sj} - \sum_{(v_j, v_s) \in A} f_{js} = v(f),$$

对于收点  $v_t$ ，

$$\sum_{(v_t, v_j) \in A} f_{tj} - \sum_{(v_j, v_t) \in A} f_{jt} = -v(f),$$

式中  $v(f)$  称为这个可行流的流量，即发点的净输出量。

可行流总是存在的，例如零流。



最大流问题可以写为如下的线性规划模型

$$\max v(f)$$

s.t.

$$\sum_{j:(v_i, v_j) \in A} f_{ij} - \sum_{j:(v_j, v_i) \in A} f_{ji} = \begin{cases} v(f), & i = s, \\ -v(f), & i = t, \\ 0, & i \neq s, t, \end{cases} \quad (4.1)$$
$$0 \leq f_{ij} \leq c_{ij}, \quad \forall (v_i, v_j) \in A.$$





### 3. 增广路

若给一个可行流  $f = \{f_{ij}\}$ ，把网络中使  $f_{ij} = c_{ij}$  的弧称为饱和弧，使  $f_{ij} < c_{ij}$  的弧称为非饱和弧。把  $f_{ij} = 0$  的弧称为零流弧， $f_{ij} > 0$  的弧称为非零流弧。

若  $\mu$  是网络中联结发点  $v_s$  和收点  $v_t$  的一条路，我们定义路的方向是从  $v_s$  到  $v_t$ ，则路上的弧被分为两类：一类是弧的方向与路的方向一致，叫做前向弧。前向弧的全体记为  $\mu^+$ 。另一类弧与路的方向相反，称为后向弧。后向弧的全体记为  $\mu^-$ 。





**定义 4.3** 设  $f$  是一个可行流,  $\mu$  是从  $v_s$  到  $v_t$  的一条路, 若  $\mu$  满足: 前向弧是非饱和弧, 后向弧是非零流弧, 则称  $\mu$  为 (关于可行流  $f$ ) 一条增广路。





## 4.4.2 寻求最大流的标号法 (Ford—Fulkerson)

从 $v_s$ 到 $v_t$ 的一个可行流出发（若网络中没有给定 $f$ ，则可以设 $f$ 是零流），经过标号过程与调整过程，即可求得从 $v_s$ 到 $v_t$ 的最大流。这两个过程的步骤分述如下。

### (A) 标号过程

在下面的算法中，每个顶点 $v_x$ 的标号值有两个， $v_x$ 的第一个标号值表示在可能的增广路上， $v_x$ 的前驱顶点； $v_x$ 的第二个标号值记为 $\delta_x$ ，表示在可能的增广路上可以调整的流量。



(1) 初始化, 给发点 $v_s$ 标号为 $(0, \infty)$ 。

(2) 若顶点 $v_x$ 已经标号, 则对 $v_x$ 的所有未标号的邻接顶点 $v_y$ 按以下规则标号

i) 若 $(v_x, v_y) \in A$ , 且 $f_{xy} < c_{xy}$ 时, 令 $\delta_y = \min\{c_{xy} - f_{xy}, \delta_x\}$ 则给顶点 $v_y$ 标号为 $(v_x, \delta_y)$ , 若 $f_{xy} = c_{xy}$ , 则不给顶点 $v_y$ 标号。

ii)  $(v_y, v_x) \in A$ , 且 $f_{yx} > 0$ , 令 $\delta_y = \min\{f_{yx}, \delta_x\}$ , 则给 $v_y$ 标号为 $(-v_x, \delta_y)$ , 这里第一个标号值 $-v_x$ , 表示在可能的增广路上,  $(v_y, v_x)$ 为反向弧; 若 $f_{yx} = 0$ , 则不给 $v_y$ 标号。



(3) 不断地重复步骤 (2) 直到收点  $v_t$  被标号, 或不再有顶点可以标号为止。当  $v_t$  被标号时, 表明存在一条从  $v_s$  到  $v_t$  的增广路, 则转向增流过程 (B)。如若  $v_t$  点不能被标号, 且不存在其它可以标号的顶点时, 表明不存在从  $v_s$  到  $v_t$  的增广路, 算法结束, 此时所获得的流就是最大流。

### (B) 增流过程

(1) 令  $v_y = v_t$ 。

(2) 若  $v_y$  的标号为  $(v_x, \delta_t)$ , 则  $f_{xy} = f_{xy} + \delta_t$ ; 若  $v_y$  的标号为  $(-v_x, \delta_t)$ , 则  $f_{yx} = f_{yx} - \delta_t$ 。

(3) 若  $v_y = v_s$ , 把全部标号去掉, 并回到标号过程 (A)。否则, 令  $v_y = v_x$ , 并回到增流过程 (2)。



**例4.7** 现需要将城市 $s$ 的石油通过管道运送到城市 $t$ ，中间有4个中转站 $v_1, v_2, v_3$ 和 $v_4$ ，城市与中转站的连接以及管道的容量如图4.5所示，求从城市 $s$ 到城市 $t$ 的最大流。

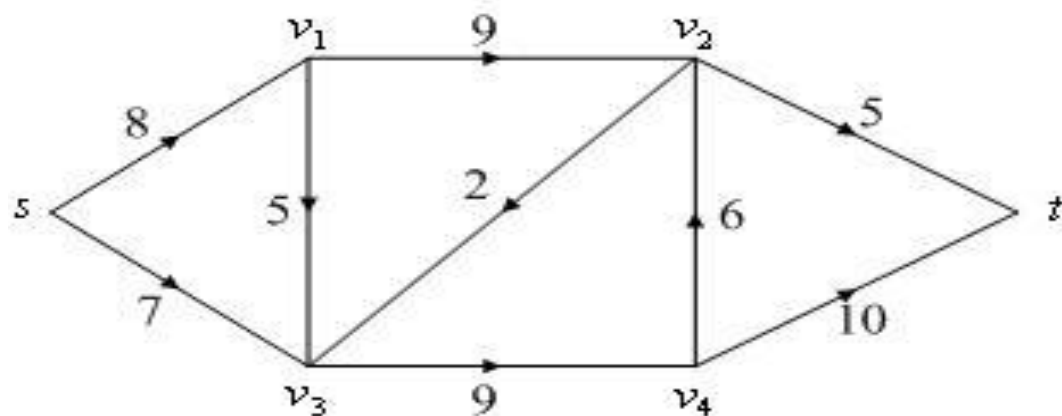


图4.5 网络图

## 4.5 最小费用最大流问题

### 4.5.1 最小费用最大流

给定网络  $D = (V, A, C)$ ，每一弧  $(v_i, v_j) \in A$  上，除了已给容量  $c_{ij}$  外，还给了一个单位流量的费用  $b(v_i, v_j) \geq 0$ （简记为  $b_{ij}$ ）。所谓最小费用最大流问题就是要求一个从发点  $v_s$  到收点  $v_t$  的最大流，使流的总输送费用  $\sum_{(v_i, v_j) \in A} b_{ij} f_{ij}$  取最小值。

最小费用最大流问题可以归结为两个线性规划问题，首先用 (4.1) 的线性规划模型求出最大流量  $v(f_{\max})$ ，然后用如下的线性规划模型求出最大流对应的最小费用。



$$\min \sum_{(v_i, v_j) \in A} b_{ij} f_{ij},$$

$$\text{s.t.} \quad 0 \leq f_{ij} \leq c_{ij}, \quad \forall (v_i, v_j) \in A, \quad (4.2)$$

$$\sum_{j: (v_i, v_j) \in A} f_{ij} - \sum_{j: (v_j, v_i) \in A} f_{ji} = d_i,$$

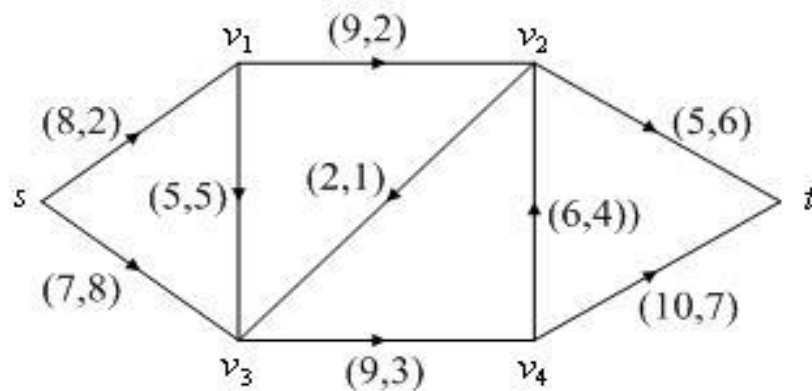
其中

$$d_i = \begin{cases} v(f_{\max}), & i = s, \\ -v(f_{\max}), & i = t, \\ 0, & i \neq s, t. \end{cases}$$

这里  $v(f_{\max})$  表示 (4.1) 线性规划模型求得的最大流的流量。



**例 4.8 (最小费用最大流问题) (续例 4.7)** 由于输油管道的长短不一或地质等原因,使每条管道上运输费用也不相同,因此,除考虑输油管道的最大流外,还需要考虑输油管道输送最大流的最小费用。图 4.6 所示是带有运费的网络,其中第 1 个数字是网络的容量,第 2 个数字是网络的单位运费。



**图 4.6 最小费用最大流的网络图**

求得最大流的最小费用是 205。



## 4.5.2 求最小费用流的一种迭代方法

这里所介绍的求最小费用流的迭代方法，是由 Busacker 和 Gowan 在 1961 年提出的。其主要步骤如下

- (1) 求出从发点到收点的最小费用通路  $\mu(s, t)$ 。
- (2) 对该通路  $\mu(s, t)$  分配最大可能的流量

$$\bar{f} = \min_{(v_i, v_j) \in \mu(s, t)} \{c_{ij}\}$$

并让通路上的所有边的容量相应减少  $\bar{f}$ 。这时，对于通路上的饱和边，其单位流费用相应改为  $\infty$ 。







(3) 作该通路 $\mu(s, t)$ 上所有边 $(v_i, v_j)$ 的反向边 $(v_j, v_i)$ 。令

$$c_{ji} = \bar{f}, \quad b_{ji} = -b_{ij}$$

(4) 在这样构成的新网络中，重复上述步骤 (1), (2), (3), 直到从发点到收点的全部流量等于指定的 $v(f)$ 为止 (或者再也找不到从 $v_s$ 到 $v_t$ 的最小费用道路)。





## 4.6 Matlab 的图论工具箱

### 4.6.1 Matlab 图论工具箱的命令

Matlab 图论工具箱的命令见表 4.1。

表 4.1 Matlab 图论工具箱的相关命令

命令名	功能
graphallshortestpaths	求图中所有顶点对之间的最短距离
graphconncomp	找无向图的连通分支, 或有向图的强 (弱) 连通分支
graphisomorphism	确定两个图是否同构, 同构返回 1, 否则返回 0





命令名	功能
<b>graphisspanntree</b>	确定一个图是否是生成树，是返回 1，否则返回 0
<b>graphmaxflow</b>	计算有向图的最大流
<b>graphminspanntree</b>	在图中找最小生成树
<b>graphpred2path</b>	把前驱顶点序列变成路径的顶点序列
<b>graphshortestpath</b>	求图中指定的一对顶点间的最短距离和最短路径
<b>graphtopoorder</b>	执行有向无圈图的拓扑排序
<b>graphtraverse</b>	求从一顶点出发，所能遍历图中的顶点





## 4.6.2 应用举例

**例 4.9** 用 Matlab 工具箱求图 4.7 中从  $v_1$  到  $v_{11}$  的最短路和最短路径。

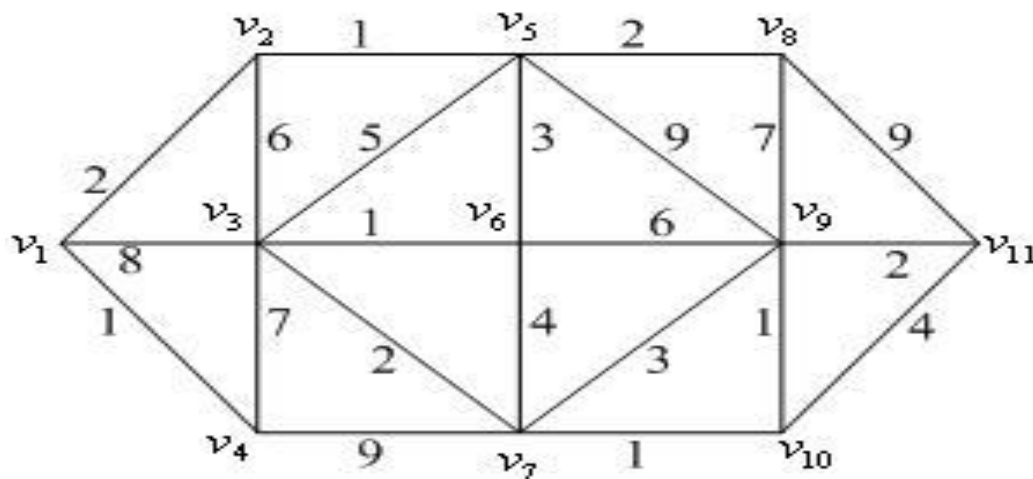


图 4.7 无向图的最短路

求得最短路径为

$$v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_3 \rightarrow v_7 \rightarrow v_{10} \rightarrow v_9 \rightarrow v_{11},$$

最短路径的长度为 13。



**例 4.10 (渡河问题)** 某人带狼、羊以及蔬菜渡河，一小船除需人划外，每次只能载一物过河。而人不在场时，狼要吃羊，羊要吃菜，问此人应如何过河？

**解** 该问题可以使用图论中的最短路算法进行求解。可以用四维向量来表示状态，其中第一分量表示人，第二分量表示狼，第三分量表示羊，第四分量表示蔬菜；当人或物在此岸时相应分量取 1，在对岸时取 0。





根据题意，人不在场时，狼要吃羊，羊要吃菜，因此，人不在场时，不能将狼与羊，羊与蔬菜留在河的任一岸。例如，状态  $(0, 1, 1, 0)$  表示人和菜在对岸，而狼和羊在此岸，这时人不在场狼要吃羊，因此，这个状态是不可行的。





通过穷举法将所有可行的状态列举出来，可行的状态有  $(1, 1, 1, 1)$  ,  $(1, 1, 1, 0)$  ,  $(1, 1, 0, 1)$  ,  $(1, 0, 1, 1)$  ,  $(1, 0, 1, 0)$  ,  $(0, 1, 0, 1)$  ,  $(0, 1, 0, 0)$  ,  $(0, 0, 1, 0)$  ,  $(0, 0, 0, 1)$  ,  $(0, 0, 0, 0)$  可行状态共有十种。每一次的渡河行为改变现有的状态。现构造赋权图  $G = (V, E, W)$  , 其中顶点集合  $V = \{v_1, \dots, v_{10}\}$  中的顶点（按照上面的顺序编号）分别表示上述十个可行状态，当且仅当对应的两个可行状态之间存在一个可行转移时两顶点之间才有边连接，并且对应的权重取为1，当两个顶点之间不存在可行转移时，可以把相应的权重取为  $\infty$ 。





因此问题变为在图 $G$ 中寻找一条由初始状态  $(1, 1, 1, 1)$  出发, 经最小次数转移达到最终状态  $(0, 0, 0, 0)$  的转移过程, 即求从状态  $(1, 1, 1, 1)$  到状态  $(0, 0, 0, 0)$  的最短路径。这就将问题转化成了图论中的最短路问题。

该题的难点在于计算邻接矩阵, 由于摆渡一次就改变现有的状态, 为此再引入一个四维状态转移向量, 用它来反映摆渡情况。用 1 表示过河, 0 表示未过河。例如,  $(1, 1, 0, 0)$  表示人带狼过河。状态转移只有四种情况, 用如下的向量表示  $(1, 0, 0, 0)$ ,  $(1, 1, 0, 0)$ ,  $(1, 0, 1, 0)$ ,  $(1, 0, 0, 1)$ 。







现在规定状态向量与转移向量之间的运算为

$$0 + 0 = 0, 1 + 0 = 1, 0 + 1 = 1, 1 + 1 = 0$$

通过上面的定义，如果某一个可行状态加上转移向量得到的新向量还属于可行状态，则这两个可行状态对应的顶点之间就存在一条边。用计算机编程时，可以利用普通向量的异或运算实现。





赋权图 $G$ 之间的状态转移关系见图 4.8，最终求得的状态转移顺序为

1      6      3      7      2      8      5      10，

经过 7 次渡河就可以把狼，羊，蔬菜运过河，第一次运羊过河，空船返回；第二次运菜过河，带羊返回；第三次运狼过河，空船返回；第四次运羊过河。

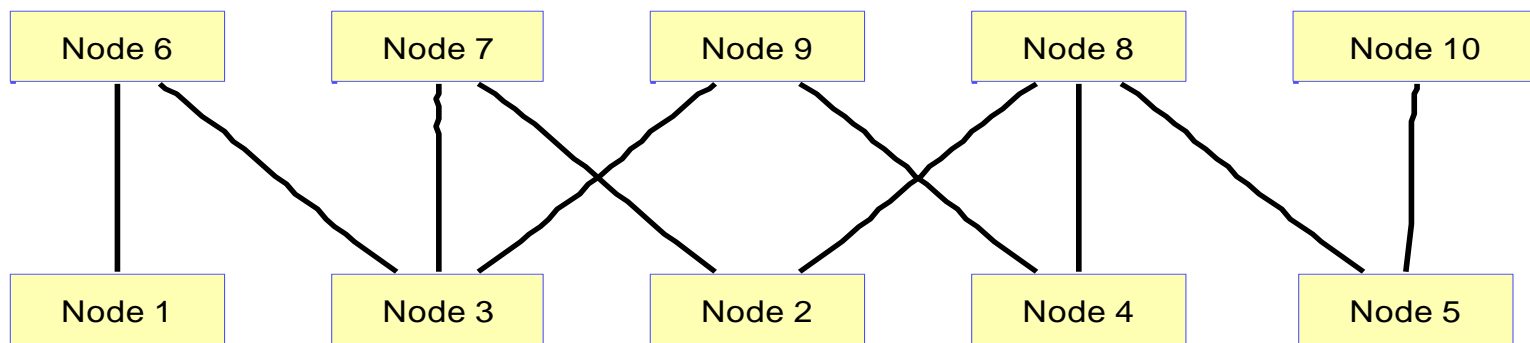
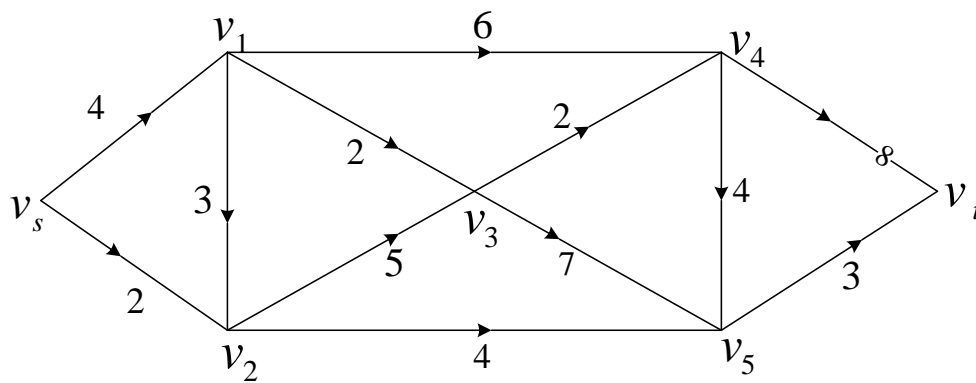


图 4.8 可行状态之间的转移



**例 4.11** 求图 4.9 所示有向图中 $v_s$ 到 $v_t$ 的最短路径及长度



**图 4.9 有向图的最短路**





解 该赋权有向图中顶点集  $V = \{v_s, v_1, \dots, v_5, v_t\}$  中  
总共有 7 个顶点，邻接矩阵

$$W = \begin{bmatrix} 0 & 4 & 2 & \infty & \infty & \infty & \infty \\ \infty & 0 & 3 & 2 & 6 & \infty & \infty \\ \infty & \infty & 0 & 5 & \infty & 4 & \infty \\ \infty & \infty & \infty & 0 & 2 & 7 & \infty \\ \infty & \infty & \infty & \infty & 0 & 4 & 8 \\ \infty & \infty & \infty & \infty & \infty & 0 & 3 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{bmatrix}.$$





**例 4.12** 设有 9 个节点  $v_i (i = 1, \dots, 9)$ , 他们的坐标分别为  $(x_i, y_i)$ , 具体数据见表 4.2。任意两个节点之间的距离为

$$d_{ij} = |x_i - x_j| + |y_i - y_j|,$$

问怎样连接电缆, 使每个节点都连通, 且所用的总电缆长度为最短?

表 4.2 点的坐标数据表

$i$	1	2	3	4	5	6	7	8	9
$x_i$	0	5	16	20	33	23	35	25	10
$y_i$	15	20	24	20	25	11	7	0	3



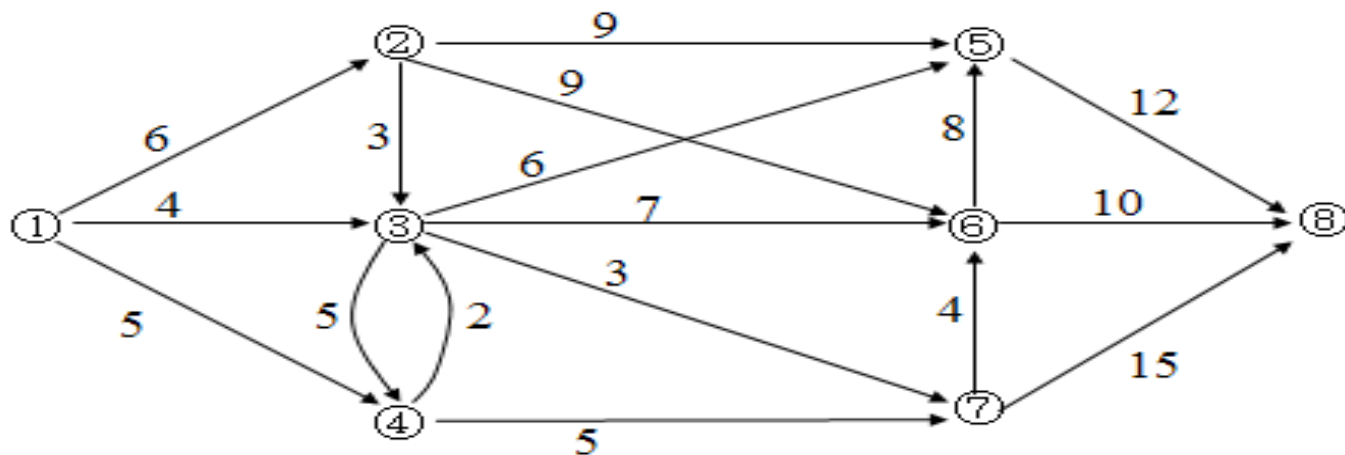


解 以  $V = \{v_1, v_2, \dots, v_9\}$  作为顶点集, 构造赋权图  $G = (V, E, W)$ , 这里  $W = (w_{ij})_{9 \times 9}$  为邻接矩阵, 其中  $w_{ij} = d_{ij}$ ,  $i, j = 1, 2, \dots, 9$ 。求总电缆长度最短的问题实际上就是求图  $G$  的最小生成树。





**例 4.13** 求图 4.10 中从①到⑧的最大流。



**图 4.10 最大流问题的网络图**

**解** Matlab 图论工具箱求解最大流的命令，只能解决权重都为正值，且两个顶点之间不能有两条弧的问题。图 4.10 中顶点 3，4 之间有两条弧，为此，在顶点 4 和顶点 3 之间加入一个虚拟的顶点 9，并添加两条弧，删除顶点 4 到顶点 3 的权重为 2 的弧，加入的两条弧的容量都是 2。



## 4.7 旅行商 (TSP) 问题

### 4.7.1 修改圈近似算法

一名推销员准备前往若干城市推销产品，然后回到他的出发地。如何为他设计一条最短的旅行路线（从驻地出发，经过每个城市恰好一次，最后返回驻地）？这个问题称为旅行商问题。用图论的术语说，就是在一个赋权完全图中，找出一个有最小权的 Hamilton 圈。称这种圈为最优圈。目前还没有求解旅行商问题的有效算法。所以希望有一个方法以获得相当好（但不一定最优）的解。







一个可行的办法是首先求一个 Hamilton 圈 $C$ ，然后适当修改 $C$ 以得到具有较小权的另一个 Hamilton 圈。修改的方法叫做改良圈算法。设初始圈 $C = v_1v_2 \cdots v_nv_1$ 。

(1) 对于 $1 \leq i < i+1 < j \leq n$ ，构造新的 Hamilton 圈

$$C_{ij} = v_1v_2 \cdots v_iv_jv_{j-1}v_{j-2} \cdots v_{i+1}v_{j+1}v_{j+2} \cdots v_nv_1,$$

它是由 $C$ 中删去边 $v_iv_{i+1}$ 和 $v_jv_{j+1}$ ，添加边 $v_iv_j$ 和 $v_{i+1}v_{j+1}$ 而得到的。若 $w(v_iv_j) + w(v_{i+1}v_{j+1}) < w(v_iv_{i+1}) + w(v_jv_{j+1})$ ，则以 $C_{ij}$ 代替 $C$ ， $C_{ij}$ 叫做 $C$ 的改良圈。

(2) 转 (1)，直至无法改进，停止。





**例 4.14** 从北京 (Pe) 乘飞机到东京(T)、纽约(N)、墨西哥城(M)、伦敦(L)、巴黎(Pa)五城市做旅游，每城市恰去一次再回北京，应如何安排旅游线，使旅程最短？用修改圈算法，求一个近似解。各城市之间的航线距离如表 4.3。

**表 4.3 六城市间的距离**

	L	M	N	Pa	Pe	T
L		56	35	21	51	60
M	56		21	57	78	70
N	35	21		36	68	68
Pa	21	57	36		51	61
Pe	51	78	68	51		13
T	60	70	68	61	13	





求得近似圈为 $5 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow 6 \rightarrow 5$ ；近似圈的长度为211。

实际上我们我们可以用下节的数学规划模型求得精确的最短圈长度为 211，这里的近似算法凑巧求出了准确解。



## 4.7.2 旅行商问题的数学规划模型

设城市的个数为 $n$ ， $d_{ij}$ 是两个城市 $i$ 与 $j$ 之间的距离， $x_{ij} = 0$ 或 $1$ （ $1$ 表示走过城市 $i$ 到城市 $j$ 的路， $0$ 表示没有选择走这条路）。则有

$$\min \sum_{i \neq j} d_{ij} x_{ij},$$

$$\text{s.t.} \quad \sum_{j=1}^n x_{ij} = 1, i = 1, 2, \dots, n, \text{ (每个点只有一条边出去),}$$

$$\sum_{i=1}^n x_{ij} = 1, j = 1, 2, \dots, n, \text{ (每个点只有一条边进去),}$$

$$\sum_{i,j \in s} x_{ij} \leq |s| - 1, 2 \leq |s| \leq n - 1, s \subset \{1, 2, \dots, n\}, \text{ 即 } s \text{ 为}$$

$\{1, 2, \dots, n\}$ 的真子集，

(除起点和终点外，各边不构成圈)

$$x_{ij} \in \{0, 1\}, i, j = 1, 2, \dots, n, i \neq j.$$



**例 4.15** 已知 SV 地区各城镇之间距离见表 4.4，某公司计划在 SV 地区做广告宣传，推销员从城市 1 出发，经过各个城镇，再回到城市 1。为节约开支，公司希望推销员走过这 10 个城镇的总距离最少。

**表 4.4 城镇之间的距离**

	2	3	4	5	6	7	8	9	10
1	8	5	9	12	14	12	16	17	22
2		9	15	17	8	11	18	14	22
3			7	9	11	7	12	12	17
4				3	17	10	7	15	18
5					8	10	6	15	15
6						9	14	8	16
7							8	6	11
8								11	11
9									10





求得的最短路径为

$1 \rightarrow 2 \rightarrow 6 \rightarrow 9 \rightarrow 7 \rightarrow 10 \rightarrow 8 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 1,$

最短路径长度为 73。

