MATLAB
开发实例系列图书

在线交流 有问必答

图论算法及其MATLAB实现

王海英 黄 强 李传涛 褚宝增 编著

北京航空航天大学出版社

图论算法及其 MATLAB 实现

王海英　黄　强　李传涛　褚宝增　编著

北京航空航天大学出版社

内 容 简 介

本书系统介绍了图论重要算法的思想及其 MATLAB 实现。

全书分为相对独立的 9 章，每章都是解决一类问题的算法思想及其 MATLAB 实现，首先介绍有关基础知识，然后给出相关著名实际问题及解决此问题的算法思想，最后给出 MATLAB 实现。第 1 章主要介绍图论的基础知识，同时也给出了可达矩阵的计算，以及关联矩阵和邻接矩阵的相互转换等重要算法及其 MATLAB 实现；第 2～8 章分别介绍最短路、连通图、树、Euler 图和 Hamilton图、匹配、网络中的流、最小费用流等相关问题，而且均给出了有关问题的解决算法及其 MATLAB 实现；第 9 章主要介绍染色问题，本章不仅介绍了几种传统的染色思想，而且还给出了当今研究领域中非常活跃的非传统染色思想，并分别给出其 MATLAB 实现。

本书可供数学、计算机科学、工程科学等学科中相关专业的大学生、研究生阅读，也可供相关专业研究人员参考。

图书在版编目(CIP)数据

图论算法及其 MATLAB 实现/王海英等编著. —北京
:北京航空航天大学出版社，2010.2
ISBN 978-7-81124-940-8

Ⅰ. ①图… Ⅱ. ①王… Ⅲ. ①图论算法②计算机辅助计算—软件包，MATLAB Ⅳ. ①0157.5②TP391.75

中国版本图书馆 CIP 数据核字(2009)第 186090 号

图论算法及其 MATLAB 实现

王海英 黄 强 李传涛 褚宝增 编著

责任编辑 宋淑娟

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100191) 发行部电话：(010)82317024 传真：(010)82328026

http://www.buaapress.com.cn E-mail：bhpress@263.net

印刷有限公司印装 各地书店经销

*

开本：787×1092 1/16 印张：10.25 字数：262 千字

2010 年 2 月第 1 版 2010 年 2 月第 1 次印刷 印数：4 000 册

ISBN 978-7-81124-940-8 定价：24.00 元

前　言

图论算法广泛应用于物理、化学、运筹学、计算机科学、电子学、信息论、控制论、网络理论、管理科学、社会科学等众多学科领域。随着这些学科的发展，特别是计算机科学的快速发展，又大大促进了图论和其他学科的发展。

图论算法是计算机科学的核心。近几年，随着强有力的 MATLAB 等数学软件的迅速发展，图论算法在数学和计算机等各学科方面的应用越来越广泛，从而使各学科的研究者越来越多地重视图论算法及其 MATLAB 实现和典型案例，而市场上又缺少这方面的指导性书籍。

本书将图论的基础知识、图论的著名问题以及相应的 MATLAB 程序代码和简单实例完美地结合在一起，力求语言简洁易懂，问题广泛有趣，算法科学，实例浅显，增强 MATLAB 实现的技巧性和操作性。读者可以通过简单案例，把图论的重要算法与 MATLAB 编程完美结合。

本书力求内容丰富，各章节相互联系，具备指导性书籍的系统性、科学性、实用性和引导性；同时，各章又相对独立，自成体系，为读者提供极大方便。

本书的创新之处在于，每一章均以著名实际问题为引入点，以图论算法为指导线，运用简单案例达到与 MATLAB 实现的完美结合，真正让各层次的读者学会运用图论理论解决实际问题，从而培养读者的图论思维，使读者惊叹图论方法的美妙与魅力。最后还为读者提供了当今图论标号方面等未解决的问题。

本书将在每一章节中给出著名图论的算法步骤及其一般 MATLAB 程序；同时，紧随每个案例分析，均给出解决问题的 MATLAB 源程序，这样对于初学者来说，具有很强的编程操作性。

本书是在中国地质大学(北京)王海英多年专业讲义的基础上重新修订编写而成，其中，山东体育学院体育运动学校的李传涛完成了本书程序的编写工作；中国科学院数学与系统科学院的黄强完成了本书全部程序的调试、修改和图形绘制等工作，褚宝增教授完成全书的定稿和校正工作。作者相信，此书将开启图论算法与 MATLAB 完美结合的首页，也将为更多有实际需求的读者提供更多的指导。

十分感谢中国地质大学(北京)2008 年教学研究与教学改革立项的支持(项目题目：数学知识、数学建模与 MATLAB 等数学软件在实践中相互结合的理论研究)！感谢北京航空航天大学出版社的认可、建议和关心！此外，本文的算法思想均离不开古今中外图论算法研究的完美理论与应用，感谢这些图论研究者们！

北京航空航天大学出版社联合 MATLAB 中文论坛(http://www.iLoveMatlab.cn)为本书设立了在线交流版块，地址是 http://www.iLoveMatlab.cn/forum-170-1.html，有问必答！作者会第一时间在 MATLAB 中文论坛勘误，也会根据读者要求陆续上传更多案例和相关知识链接，还会随着 MATLAB 版本的升级增添必要的内容以满足读者的需求。希望这本不断“成长”的书能最大限度地解决读者在学习、研究、工作中遇到的与图论相关的问题。

由于水平有限，书中缺陷和错误恳请读者批评指正。最后，再次感谢我们所参考的书籍和文献的作者！

编　者

2009 年 10 月

目　　录

第 1 章

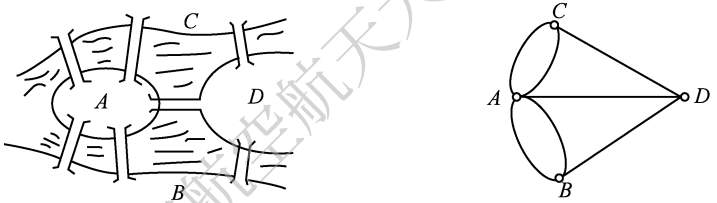
图论的基础知识

图论是一门应用广泛且内容丰富的学科，随着计算机和数学软件的发展，图论越来越多地被人们应用到实际生活和生产中，也成为解决众多实际问题的重要工具。

图论中的概念和定理均与实际问题有关，并起着关键性作用。本章主要介绍图论的基本概念和性质及几个简单算法的 MATLAB 实现。

1.1 图论的起源

图论起源于一个实际问题——柯尼斯堡(Konigsberg)七桥问题。柯尼斯堡位于前苏联的加里宁格勒，普雷格尔河横穿此城堡，河中有两个小岛，记为 A 和 D，并有七座桥连接岛与河岸、岛与岛(图 1.1(a))。当时居民有个有趣的问题：是否存在这样一种走法，要从这四个河岸中的任何一个河岸开始，通过每座桥且恰巧都经过一次，再回到起点。此问题就是著名的柯尼斯堡七桥问题。



(a) 实际图　　(b) 简化图(图G)

图 1.1　七桥问题实际图与简化图

1736 年，瑞典数学家欧拉(Leonhard Euler)解决了柯尼斯堡问题，由此图论诞生。欧拉认为，此问题关键在于河岸与岛所连接的桥的数目，而与河岸和岛的大小、形状以及桥的长度和曲直无关，他用点表示河岸和岛，用连接相应顶点的线表示各座桥，这样就构成一个图 G(图 1.1(b))，此问题就等价于图 G 中是否存在经过该图的每一条边一次且仅一次的“闭路”问题。Euler 不仅论证了此走法是不存在的，而且还推广了这个问题，从此开始了图论理论的研究。

1.2 著名的图论学者——欧拉

本节简介著名图论的创始人——欧拉(图 1.2)。

欧拉(Leonhard Euler，1707—1783)，出生在瑞士的巴塞尔城，13 岁进巴塞尔大学读书，得到当时最著名的数学家约翰·伯努利(Johann Bernoulli，1667—1748)的精心指导。

图 1.2　欧拉像

欧拉是一位科学史上最多产的杰出的数学家。他凭借杰出的智慧、顽强的毅力、孜孜不倦的治学精神，即使双目失明，也从未停止过对数学的研究。据统计，在欧拉不倦的一生中，共撰写 886 本书籍和论文，包括分析学、代数、数论、物理、天文学、弹道学、航海学和建筑学等。欧拉去世后，彼得堡科学院整理他的著作，足足忙碌 47 年才得以完成。高斯曾评价说："研究欧拉的著作永远是了解数学的最好方法。"

欧拉研究问题总是喜欢深入到自然与社会的深层，总是喜欢结合具体问题，被誉为非常出色的理论联系实际的巨匠和应用数学的大师。

欧拉在数学学科方面的建树颇丰，其中之一就是对著名的柯尼斯堡七桥问题的解答，并由此开创了图论学科的研究。每逢欧拉诞辰，国内外的图论学者都会以不同方式的交流来纪念他，追溯他对图论不朽的功绩。

总之，要感谢欧拉，要永远学习他身上所具备的高尚的科学道德。

1.3　图

一个(无向)**图** G 是指一个有序三元组 $(V(G),E(G),\psi_G)$，其中 $V(G)$ 为非空的顶点集，$E(G)$ 为不与 $V(G)$ 相交的边集，而 ψ_G 是关联函数，使得 G 的每条边都对应于 G 的无序顶点对 uv(未必互异)，简记为图 $G=(V(G),E(G))$ 或 $G=(V,E)$。

根据上述图的定义，可以用平面上的图形来表示图。图 G 的每个顶点均用一个小圆点(或实点)表示，每条边 $\psi_G(e)=uv$ 均用以顶点 u,v 为端点的线段表示，例如，图 1.3 表示以 $V(G)=\{v_1,v_2,v_3,v_4\}$ 为顶点集、以 $E(G)=\{v_1v_2,v_1v_3,v_1v_4,v_2v_3,v_2v_4,v_3v_4\}$ 为边集的无向图。

设图 $G=(V,E)$，分别称 $|V|$ 和 $|E|$ 为图 G 的**顶点数**(或阶)和**边数**。若图 G 的顶点数和边数都是有限集，则称 G 为**有限图**；否则称为**无限图**。仅有一个顶点的图称为**平凡图**，其他所有图均称为**非平凡图**。

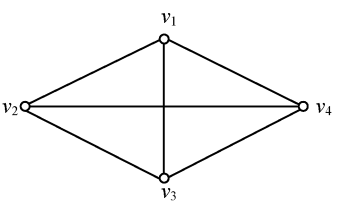


图 1.3　有四个顶点的图

设图 $G=(V,E)$。若 e 是图 G 的一条边，而 u 和 v 是满足 $\psi_G(e)=uv$ 的顶点，则称 u 和 v 为 e 的两个**端点**，称 e **关联**于 u,v，又称顶点 u,v 相互**邻接**。同样，称与同一个顶点关联的若干条互异边是**相邻**的。在图 G 中，两端点重合为一个顶点的边称为**环**；关联于相同两个端点的两条或两条以上的边称为**多重边**。若图 G 没有环和多重边，则称 G 为**简单图**。

设图 $G=(V,E)$，与顶点 v 相关联的边数(每个环计算两次)，称为顶点 v 的**度**，记为 $d_G(v)$ 或 $d(v)$；此外，用 $\delta(G)$ 和 $\Delta(G)$ 分别表示图 G 中所有顶点中最小的度和最大的度；度为零的顶点称为**孤立顶点**；度为奇数的顶点称为**奇点**，度为偶数的顶点称为**偶点**。

此外，设 S 是 $V(G)$ 的一个非空子集，v 是 G 的任一顶点，称 $N_s(v)=\{u\mid u\in S,uv\in E(G)\}$

为 v 在 S 中的**邻域**。特别地，若取 $S=V(G)$，则 $N_G(v)$ 简记为 $N(v)$。显然，当 G 是简单图时，$d(v)=|N(v)|$。

本书仅讨论有限简单图 G，若无特殊说明，不再赘述。

1.4 特殊图类

在研究和描述一般图的性质过程中，特殊图类起着很重要的作用。本节将集中给出常用特殊图的定义。

设两个简单图 $G=(V,E)$ 和 $H=(V',E')$。若 $V'\subseteq V$ 和 $E'\subseteq E$，则称 H 是 G 的**子图**，记为 $H\subseteq G$。若 H 是 G 的子图，并且 $V(H)=V(G)$，则称 H 是 G 的**生成子图**。若 H 是 G 的子图，其中 $V(H)=V(G)$ 和 $E(H)=E(G)$ 至少有一个不成立，就称 H 是 G 的**真子图**。

设图 $G=(V,E)$。假设 V' 是 V 的一个非空真子集，则以 $G-V'$ 表示从 G 中删去 V' 内的所有顶点以及与这些顶点相关联的边所得到的子图。特别地，当 $V'=\{v\}$ 时，常把 $G-\{v\}$ 简记为 $G-v$，并且用 $G[V']$ 表示 $G-(V-V')$，称为 G 的**由 V' 导出的子图**。

对于边子集也有类似的定义。设 E' 是 $E(G)$ 的子集，以 $G-E'$ 表示在 G 中删去 E' 中所有的边所得到的子图，而在图 G 中加上边集 E'' 内的所有边所得到的图记为 $G+E''$，其中 $E''\cap E(G)=\varnothing$。同样，用 $G-e$ 和 $G+f(f\notin E(G))$ 分别表示 $G-\{e\}$ 和 $G+\{f\}$；用 $G[E']$ 表示以 E' 为边集、以 E' 内的边的端点为顶点集合所构成的图，称为 G 的**由 E' 导出的子图**。

设简单图 $G=(V(G),E(G))$，它的**途径**是一个有限非空序列 $W=v_0e_1v_1e_2\cdots e_kv_k$，也就是，顶点和边的交错序列，其中 $e_i\in E(G)$，$v_j\in V(G)$，e_i 分别与 v_{i-1}，v_i 关联，$1\leqslant i\leqslant k$，$0\leqslant j\leqslant k$，记为 (v_0,v_k)-途径，称顶点 v_0，v_k 分别为途径 W 的**起点**和**终点**，$v_1,v_2,\cdots,v_{k-1}$ 称为途径 W 的内顶点，k 称为 W 的**长**。若途径 W 中的 $e_1,e_2,\cdots,e_k$ 两两互异，称之为**迹**；若迹 W 中 $v_0,v_1,v_2,\cdots,v_{k-1},v_k$ 两两互异，称之为**路**。若途径(迹，路)的起点和终点相同，则称为**闭途径**(**闭迹**，**闭路**)。闭迹亦称为**圈**，长为 k 的圈称为 k -圈，并且当 k 为偶数时称为**偶圈**，当 k 为奇数时称为**奇圈**。

若简单图 G 的两顶点 u，v 间存在路，则称 u 和 v 是连通的；并且称顶点 u，v 间的最短路的长为 u，v 间的距离，记作 $d(u,v)$。若简单图 G 的任意互异顶点均是连通的，则称 G 是**连通图**。显然，顶点的连通性是一个等价关系，顶点集 V 的每一等价类的导出子图称为图 G 的一个**连通分支**，记图 G 的连通分支个数为 $\omega(G)$。若图 G 是连通的，则 $\omega(G)=1$；否则，图 G 是不连通的。

简单图 G 及其**补图** G^c 满足有相同顶点集 V、在 G 中相邻的两个顶点均在 G^c 中不相邻。此外，从图 G 中删去所有的环，并使每一对相邻的顶点只留下一条边，即可得到 G 的一个生成子图，称为 G 的**基础简单图**。

设两个简单图 $G_1=(V_1,E_1)$ 和 $G_2=(V_2,E_2)$。若 $V_1\cap V_2=\varnothing$，则称 G_1 与 G_2 是**不相交的**；若 $E_1\cap E_2=\varnothing$，则称 G_1 与 G_2 是**边不重的**。定义 $G_1+G_2=(V_1\cup V_2,E_1\cup E_2\cup E_3)$，$G_1\cup G_2=(V_1\cup V_2,E_1\cap E_2)$，分别称为图 G_1 和 G_2 的**和图**、**并图**，其中 $E_3=\{uv|u\in V_1,v\in V_2\}$。

设 n 阶简单图 $G=(V,E)$。若 V 可划分为 m 个非空子集 $V_1,V_2,\cdots,V_m$，使得对每一个 i，$G[V_i]$ 是空图，其中 $1\leqslant i\leqslant m$，则称 G 为**m 部图**，记为 $G=(V_1,V_2,\cdots,V_m;E)$。特别地，若 $|V_1|=|V_2|=\cdots=|V_m|$，则称 G 为**等 m 部图**。特别地，设 $G=(V_1,V_2,\cdots,V_m;E)$ 是 m 部图。若对任

意顶点 $u\in V_i$ 和 $v\in V_j$，均有 $uv\in E$，其中 $1\leqslant i,j\leqslant m$，$i\neq j$，则称 $G=(V_1,V_2,\cdots,V_m;E)$ 为**完全 m 部图**，记为 $K_{n_1,n_2,\cdots,n_m}$，这里 $|V_i|=n_i$，$i=1,2,\cdots,m$。

每对互异顶点恰有一条边相连接的图称为**完全图**，阶为 n 的完全图常记为 K_n。完全图也是图论中较重要的一类特殊图。

1.5 有向图

有向图 D 是指有序三元组 $(V(D),A(D),\psi_D)$，其中 $V(D)$ 是非空的顶点集；$A(D)$ 是不与 $V(D)$ 相交的有向边集；而 ψ_D 是关联函数，使得 D 的每条有向边对应于 D 的一个有序顶点对（不必相异）。若 a 是 D 的一条有向边，而 u 和 v 是满足 $\psi_D(a)=(u,v)$ 的顶点，则称 a 为从 u 连接到 v 的一条**弧**（或**有向边**），称 u 是 a 的**始点**，v 是 a 的**终点**，在不产生混淆的情况下，可简记有向边 a 为 uv。若有向图没有环，并且任何两条弧都不具有相同方向和相同端点，则称该有向图是**严格的**。

设有向图 $D=(V(D),A(D),\psi_D)$。若 $V(D')\subseteq V(D)$，$A(D')\subseteq A(D)$，并且 ψ'_D 是 ψ_D 在 $A(D')$ 上的限制，则称有向图 $D'=(V(D'),A(D'),\psi'_D)$ 是 D 的**有向子图**。

有向图 D 的**有向途径**是指一个有限非空序列 $W=(v_0,a_1,v_1,\cdots,a_k,v_k)$，其各项交替地是顶点和有向边，使得对于 $i=1,2,\cdots,k$，有向边 a_i 有始点 v_i 和终点 v_{i-1}。有向途径 $(v_0,a_1,v_1,\cdots,a_k,v_k)$ 常简单地用其顶点序列 $(v_0,v_1,\cdots,v_k)$ 表示。**有向迹**是指本身是迹的有向途径；**有向路**和**有向圈**可类似定义。

若有向图 D 中存在有向路 (u,v)，则顶点 v 称为在 D 中从顶点 u 出发**可到达**；若 D 中任意两个顶点 u,v，顶点 u 可达 v 或 v 可达 u，则称 D 为**单向连通有向图**；若任意两个顶点在 D 中互相可到达，则称 D 为**双向连通有向图**。当然，也可从另一个角度来理解双向连通。双向连通在 D 的顶点集上是一个等价关系。根据双向连通关系确定的 $V(D)$ 的一个分类 $(V_1,V_2,\cdots,V_m)$ 所导出的有向图 $D[V_1],D[V_2],\cdots,D[V_m]$ 称为 D 的**双向分支**；若恰有一个双向分支，则称有向图 D 为**双向连通的**。

有向图 D 中以顶点 v 为终点的有向边的数目称为顶点 v 的**入度**，简记为 $d^-(v)$；以顶点 v 为始点的有向边的数目称为顶点 v 的**出度**，简记为 $d^+(v)$。在有向图 D 中，$\delta^-(D)$、$\delta^+(D)$、$\Delta^-(D)$、$\Delta^+(D)$ 分别表示 D 中所有顶点的最小和最大的入度和出度，即

$$\delta^+(D)=\min\{d_D^+(u)\mid u\in V(D)\}$$
$$\delta^-(D)=\min\{d_D^-(u)\mid u\in V(D)\}$$
$$\Delta^+(D)=\max\{d_D^+(u)\mid u\in V(D)\}$$
$$\Delta^-(D)=\max\{d_D^-(u)\mid u\in V(D)\}$$

设有向图 D，可以在相同顶点集上作一个图 G，使得对于 D 的每条有向边，G 均有一条具有相同端点的边与之对应，此图 G 称为 D 的**基础图**。反之，给定任意（无向）图 G，对于它的每条边，均给其端点指定一个顺序，从而确定一条有向边，由此得到一个有向图 D，此有向图 D 称为 G 的一个**定向图**。

类似于无向图，有向图也有简单的图形表示。一个有向图可以用它的基础图连同它边上的箭头所组成的图形来表示，其中每个箭头均指向对应弧的终点。图 1.4 表示一个有向图及其基础图。若有向图 D 的基础图是连通的，则称 D 为**弱连通有向图**。

(a) 有向图D　　(b) D的基础图

图 1.4　某有向图及其基础图示意图

为了叙述方便,本书中 D 表示有向图,G 表示它的基础图。

1.6　图的矩阵表示

图 $G=(V,E)$,一方面,由它的顶点与边之间的关联关系唯一确定,也由它的顶点与顶点之间的邻接关系唯一确定;另一方面,图 $G=(V,E)$在计算机中存储的数据结构必须完全等价于图本身的顶点与边之间的结构关系,而图的矩阵表示就能够承担这种重要的"中介"角色。此外,可通过对图的表示矩阵进行讨论来得到图本身的若干性质。

1.6.1　邻接矩阵

定义 1.1　设(无向)图 $G=(V,E)$,其中顶点集 $V=\{v_1,v_2,\cdots,v_n\}$,边集 $E=\{e_1,e_2,\cdots,e_\varepsilon\}$。用 a_{ij} 表示顶点 v_i 与顶点 v_j 之间的边数,可能取值为 0,1,2,称所得矩阵 $\boldsymbol{A}=\boldsymbol{A}(G)=(a_{ij})_{n\times n}$ 为图 G 的**邻接矩阵**。

根据图邻接矩阵的定义,易得若干性质如下:

① $\boldsymbol{A}(G)$为对称矩阵;

② 若 G 为无环图,则 $\boldsymbol{A}(G)$中第 i 行(列)的元素之和等于顶点 v_i 的度;

③ 两图 G 和 H 同构的充分必要条件是存在置换矩阵 $\boldsymbol{P}$ 使得 $\boldsymbol{A}(G)=\boldsymbol{P}^{\mathrm{T}}\boldsymbol{A}(H)\boldsymbol{P}$。

类似地,有向图 D 的邻接矩阵 $\boldsymbol{A}(D)=(a_{ij})_{n\times n}$ 的元素 a_{ij} 定义为:元素 a_{ij} 表示从始点 v_i 到终点 v_j 的有向边的条数,其中 v_i 和 v_j 为 D 的顶点。

例 1.1　求图 1.5 所示简单图的邻接矩阵。

解:根据定义,可求得该无向图的邻接矩阵为

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

注:邻接矩阵是描述图的一种常用的矩阵表示。在本书中,有时给出图的邻接矩阵,而不是画出该图来。

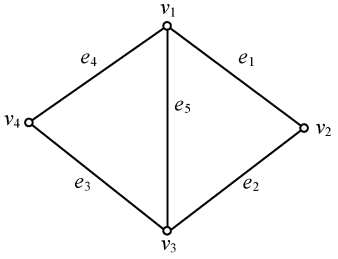


图 1.5　例 1.1 图

1.6.2　关联矩阵

定义 1.2　设任意(无向)图 $G=(V,E)$,其中顶点集 $V=\{v_1,v_2,\cdots,v_n\}$,边集 $E=\{e_1,e_2,\cdots,e_\varepsilon\}$。用 m_{ij} 表示顶点 v_i 与边 e_j 关联的次数,可能取值为 0,1,2,称所得矩阵 $\boldsymbol{M}(G)=(m_{ij})_{n\times\varepsilon}$ 为图 G 的关联矩阵。

类似地，有向图 D 的关联矩阵 $\boldsymbol{M}(D)=(m_{ij})_{n\times\varepsilon}$ 的元素 m_{ij} 定义为

$$m_{ij}=\begin{cases}1, & v_i \text{ 是有向边 } a_j \text{ 的始点}\\ -1, & v_i \text{ 是有向边 } a_j \text{ 的终点}\\ 0, & v_i \text{ 是有向边 } a_j \text{ 的不关联点}\end{cases}$$

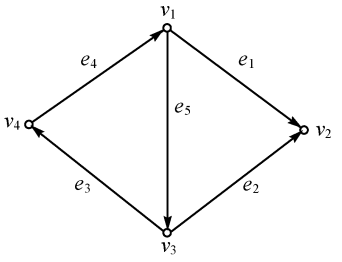


图 1.6　例 1.2 图

例 1.2　求图 1.6 所示有向图的邻接矩阵和关联矩阵。

解：根据定义，可求得该有向图的邻接矩阵和关联矩阵分别为

$$\begin{bmatrix}0&1&1&0\\0&0&0&0\\0&1&0&1\\1&0&0&0\end{bmatrix} \quad 和 \quad \begin{bmatrix}1&0&0&0&1\\-1&-1&0&0&0\\0&1&1&0&-1\\0&0&-1&1&0\end{bmatrix}$$

注：关联矩阵是描述图的另一种矩阵表示。

1.7　图论的基本性质和定理

本节主要介绍图论中最基本的定理，它是欧拉 1736 年在解决柯尼斯堡七桥问题时建立的第一个图论结果，后来的很多重要结论都与它有关。

定理 1.1(握手定理)　对每个图 $G=(V,E)$，均有 $\sum\limits_{v\in V}d(v)=2|E|$。

证明：根据顶点度的定义，在计算点度时每条边对于它所关联的顶点被计算了两次。因此，图 G 中点度的总和恰为边数 $|V|$ 的 2 倍。证毕。

推论 1.1　在任何图 $G=(V,E)$ 中，奇点的个数为偶数。

定理 1.2　对任意有向图 $D=(V,A)$ 均有 $\sum\limits_{u\in V(D)}d_D^+(u)=\sum\limits_{u\in V(D)}d_D^-(u)=|A|$。

证明：由于每一条有向边均有一个始点和一个终点，故结论成立。证毕。

1.8　计算有向图的可达矩阵的算法及其 MATLAB 实现

设有向图 $D=(V,E)$，顶点集 $V=\{v_1,v_2,\cdots,v_n\}$。定义矩阵 $\boldsymbol{P}=(p_{ij})_{n\times n}$ 为

$$p_{ij}=\begin{cases}0, & v_i \text{ 到 } v_j \text{ 不可达}\\ 1, & v_i \text{ 到 } v_j \text{ 可达}\end{cases}$$

称矩阵 $\boldsymbol{P}$ 是图 D 的**可达矩阵**。

【算法用途】

有向图的可达矩阵的计算。

【算法思想】

一般地，设 n 阶有向图 D 的邻接矩阵为 $\boldsymbol{A}$，由 $\boldsymbol{A}$ 可得到图 D 的可达矩阵，不妨设为 $\boldsymbol{P}$，其步骤如下：首先，求出 $\boldsymbol{B}_n=\boldsymbol{A}+\boldsymbol{A}^2+\cdots+\boldsymbol{A}^n$，然后，把矩阵 $\boldsymbol{B}_n$ 中不为 0 的元素改为 1，而为 0 的元素不变，这样所改换的矩阵就为图 D 的可达矩阵 $\boldsymbol{P}$。

【程序参数说明】

A 表示图的邻接矩阵。

P 表示图的可达矩阵。

【算法的 MATLAB 程序】

```
% 计算图的可达矩阵
function P = dgraf(A)
n = size(A,1);
P = A;
 %  计算矩阵 Bn
for i = 2:n
    P = P + A^i;
end

P(P~ = 0) = 1;   % 将不为 0 的元素变为 1
P;
```

例 1.3 求图 1.7 的可达矩阵。

解:根据邻接矩阵的定义,可知该图的邻接矩阵为

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

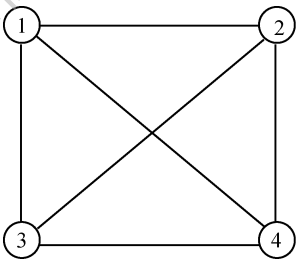


图 1.7 例 1.3 图

运行以下程序。

```
>> A = [0 1 1 1;1 0 1 1;1 1 0 1;1 1 1 0];
>> P = dgraf(A)   % 调用函数 dgraf
```

运行结果如下。

```
P =
     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

1.9 关联矩阵和邻接矩阵的相互转换算法及其 MATLAB 实现

邻接矩阵和关联矩阵均是图的不同形式的矩阵表示,它们均代表着此图的拓扑结构,重要的是,无论是无向图还是有向图,它们均能够相互转换。本节将分别给出其算法理论基础及其 MATLAB 实现。

【算法用途】

无向图或有向图的关联矩阵和邻接矩阵的相互转换。

【程序参数说明】

F 表示所给出的图的相应矩阵。

W 表示程序运行结束后的结果，即由已知矩阵所转换的结果。

当 f=0 时，要完成邻接矩阵转换为关联矩阵，此时 F 表示邻接矩阵，W 表示关联矩阵。

当 f=1 时，要完成关联矩阵转换为邻接矩阵，此时 F 表示关联矩阵，W 表示邻接矩阵。

【算法的 MATLAB 程序】

```
%无向图的关联矩阵和邻接矩阵的相互转换
function W = incandadf(F,f)

if f == 0                                   %邻接矩阵转换为关联矩阵
    m = sum(sum(F))/2;                      %计算图的边数
    n = size(F,1);
    W = zeros(n,m);
    k = 1;
    for i = 1:n
        for j = i:n
            if F(i,j)~ = 0
                W(i,k) = 1;                 %给边的始点赋值为 1
                W(j,k) = 1;                 %给边的终点赋值为 1
                k = k + 1;
            end
        end
    end
elseif f == 1                               %关联矩阵转换为邻接矩阵
    m = size(F,2);
    n = size(F,1);
    W = zeros(n,n);
    for i = 1:m
        a = find(F(:,i)~ = 0);
        W(a(1),a(2)) = 1;                   %存在边，则邻接矩阵的对应值为 1
        W(a(2),a(1)) = 1;
    end
else
    fprint('please input the right value of f');
end
W;
```


例 1.4 已知图 G 的邻接矩阵 $\boldsymbol{F}=\begin{bmatrix}0&1&1&1\\1&0&1&1\\1&1&0&1\\1&1&1&0\end{bmatrix}$，求图 G 的关联矩阵。

解:运行以下程序。

```
>> F=[0 1 1 1;1 0 1 1;1 1 0 1;1 1 1 0];
>> W=incandadf(F,0)
```

运行结果如下。

```
W =
     1     1     1     0     0     0
     1     0     0     1     1     0
     0     1     0     1     0     1
     0     0     1     0     1     1
```

例 1.5 已知图 G 的关联矩阵 $\boldsymbol{F}=\begin{bmatrix}1&1&1&0&0&0\\1&0&0&1&1&0\\0&1&0&1&0&1\\0&0&1&0&1&1\end{bmatrix}$,求图 G 的邻接矩阵。

解:运行以下程序。

```
>> F=[1 1 1 0 0 0;1 0 0 1 1 0;0 1 0 1 0 1;0 0 1 0 1 1];
>> W=incandadf(F,1)
```

运行结果如下。

```
W =
     0     1     1     1
     1     0     1     1
     1     1     0     1
     1     1     1     0
```

【算法的 MATLAB 程序】

```
% 有向图的关联矩阵和邻接矩阵的转换
function W=mattransf(F,f)

if f==0                                  %邻接矩阵转换为关联矩阵
    m=sum(sum(F));
    n=size(F,1);
    W=zeros(n,m);
    k=1;
    for i=1:n
        for j=i:n
            if F(i,j)~=0                 %由 i 发出的边,有向边的始点
                W(i,k)=1;                %关联矩阵始点值为 1
                W(j,k)=-1;               %关联矩阵终点值为-1
                k=k+1;
```

```
            end
        end
    end
elseif f == 1                          % 关联矩阵转换为邻接矩阵
    m = size(F,2);
    n = size(F,1);
    W = zeros(n,n);
    for i = 1:m
        a = find(F(:,i)~ = 0);         % 有向边的两个顶点
        if F(a(1),i) == 1
            W(a(1),a(2)) = 1;          % 有向边由 a(1)指向 a(2)
        else
            W(a(2),a(1)) = 1;          % 有向边由 a(2)指向 a(1)
        end
    end
else
    fprint('please input the right value of f');
end
W;
```

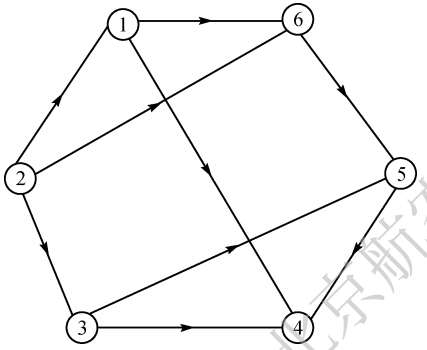


图 1.8　例 1.6 图

例 1.6　建立图 1.8 所示有向图的邻接矩阵，并将其转换为关联矩阵。

解：根据定义，该图的邻接矩阵为

$$\boldsymbol{F}=\begin{bmatrix}0&0&0&1&0&1\\1&0&1&0&0&1\\0&0&0&1&1&0\\0&0&0&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\end{bmatrix}$$

运行以下程序。

```
>>  F = [0 0 0 1 0 1;1 0 1 0 0 1;0 0 0 1 1 0;0 0 0 0 0 0;0 0 0 1 0 0;0 0 0 0 1 0];
>>  W = mattransf(F,0)
```

运行结果如下。

```
W =
     1     1     0     0     0     0     0     0     0
     0     0     1     1     0     0     0     0     0
     0     0    -1     0     1     1     0     0     0
    -1     0     0     0    -1     0     0     0     0
     0     0     0     0     0    -1     0     0     0
     0    -1     0    -1     0     0     0     0     0
```

例 1.7 建立图 1.9 所示有向图的关联矩阵，并求该图的邻接矩阵。

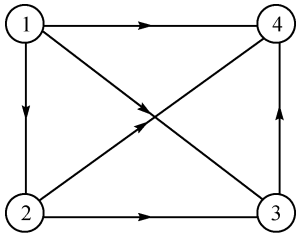


图 1.9 例 1.7 图

解：根据定义，该图的关联矩阵为

$$\boldsymbol{F}=\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

运行以下程序。

```
>> F = [1 1 1 0 0 0; -1 0 0 1 1 0; 0 -1 0 -1 0 1; 0 0 -1 0 -1 -1];
>> W = mattransf(F,1)
```

运行结果如下。

```
W =
     0     1     1     1
     0     0     1     1
     0     0     0     1
     0     0     0     0
```


习题一

1－1 给定图 1.10 的邻接矩阵，试求出其可达矩阵。

1－2 求图 1.11 的关联矩阵和邻接矩阵，并试用程序进行转换。

图 1.10 习题 1－1 图

图 1.11 习题 1－2 图

1－3 试将某图的关联矩阵 $\boldsymbol{F}=\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \\ -1 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$ 转换为邻接矩阵。

第 2 章

最短路

随着计算机技术的飞速发展，人们对效率的要求越来越高，最短路问题也逐渐成为计算机、运筹学、地理信息科学等领域的研究热点之一。由于在实际中得到广泛的应用，因此所求最短路问题的算法以及提高该算法的求解效率也就具有其重大的现实意义，国内外诸多行业的专家均对此问题进行了深入研究。

2.1 路

在图论理论中，路具有特殊的重要性，古往今来，许多学者均对它进行过深入研究。本节主要介绍简单图 $G=(V,E)$ 中有关路和连通性的简单性质。

定理 2.1 若图 G 中有一条 (u,v)-途径，则 G 中也存在一条 (u,v)-路。

证明：事实上，由 u 出发沿 (u,v)-途径走，若遇到相同点，则把相同点间的那段途径去掉，然后继续沿 (u,v)-途径往下走，一直走到终点 v 为止。按照此做法可知，最终所得的 (u,v)-途径即为图 G 中的一条 (u,v)-路。证毕。

定理 2.2 设 G 为简单图，且最小度 $\delta(G)\geqslant k$，则 G 中存在长为 k 的路。

证明：设 P 为简单图 G 中的一条最长路，其长为 l，则 $l>k$。进一步设 P 为 $v_1v_2\cdots v_lv_{l+1}$，由假设 $d_G(v_1)\geqslant\delta\geqslant l>k$，得在 P 外存在一个顶点 v_0 与 v_1 邻接，这样就得到 $v_0v_1v_2\cdots v_lv_{l+1}$ 是图 G 的另外一条路，并且长于 P，这与 P 的最长性矛盾。

从上述可知，$l\geqslant k$，这样，取 P 的长为 k 的一段作为所求。证毕。

定理 2.3 在连通图中任意两条最长的路都有公共顶点。

证明：设 (v_1,v_2)-路，(v_1',v_2')-路均是 G 中之最长路，且无公共顶点。由于 G 是连通的，故存在 (v_2,v_2')-路，令 v_3 为由 v_2 出发沿 (v_2,v_2')-路前进，最后一个和 (v_1,v_2)-路相交的顶点，v_3' 为由 v_3 出发，沿 (v_2,v_2')-路前进，第一个和 (v_1',v_2')-路相交的顶点。

不失一般性，(v_1,v_2)-路中的 (v_1,v_3)-段的长度不小于 (v_1',v_2')-路中 (v_1',v_3')-段的长度，从而，从 v_1 开始，经过 (v_1,v_2)-路中的 (v_1,v_3)-段，然后经 (v_2,v_2')-路中的 (v_3,v_3')-段，最后经 (v_1',v_2')-路中的 (v_3',v_2')-段到达 v_2'。显然，此路要比原来的 (v_1,v_2)-路长，这与 (v_1,v_2)-路是最长的矛盾。故 (v_1,v_2)-路和 (v_1',v_2')-路有公共点。证毕。

定理 2.4 若顶点 u,v 在 G 中是连通的，则定义 G 中最短的 (u,v)-路的长度为 G 中 u,v 之间的距离，记为 $d_0(u,v)$；若 u,v 在 G 中不连通，则定义 $d_0(u,v)$ 为无穷。证明对于任意三个顶点有 $d(u,v)+d(v,w)\geqslant d(u,w)$。

证明：$d(u,v)$ 是在不引起混淆的情况下 $d_0(u,v)$ 的简记。

① 当 u,v,w 连通时，因为由 (u,v)-路和 (v,w)-路合并起来构成 (u,w)-途径，又因为最短的 (u,w)-路的长度不超过 (u,w)-路径的长度，故由距离的定义可知，结论成立。

② 当 u,v,w 不连通时，$d(u,v),d(v,w),d(u,w)$ 中至少两个为无穷，结论也成立。

证毕。

定理 2.5 图 G 的**直径**是 G 中两顶点之间的最大距离。试证：若 G 有大于 3 的直径，则 G^c 的直径小于 3。

证明：对任意一对 $u,v\in V(G)$，若边 $uv\notin E(G)$，则 $uv\in E(G^c)$，故 $d_0(u,v)=1$。若 $uv\in E(G)$，则 $uv\notin E(G^c)$，这时分两种情况讨论：

① $V(G)$中任意顶点至少与 u,v 中一个顶点相邻。此时任意的 $x,y\in V(G)$有 $d_0(u,v)\leqslant 3$，这与假设是矛盾的，故情况①不可能。

② $V(G)$中存在一顶点 w 使得 $uw,wv\notin E(G)$，则 $vw,wv\in E(G^c)$。此时 $d_0(u,v)=2$。综合上述情况，G^c 的直径小于 3。

证毕。

定理 2.6 若 G 是连通单图，但不是完全图，则 G 存在如下三个顶点 u,v,w，满足 $uv,vw\in E,uw\notin E$。

证明：由假定 G 不是完全图，故存在 $u,w_1\in V,uw_1\notin E$；由于 G 是连通的，故存在一条连接 u,w_1 的最短路，设为 $uu_1u_2\cdots u_nw_1,n\geqslant 1$。显然 $uu_2\notin E$，否则与最短路假定矛盾。于是令 $u_1=v,u_2=w$(当 $n=1$ 时，$u_2=w_1$)。即为所求。证毕。

定理 2.7 若最小度 $\delta\geqslant 2$，则简单图 G 含有圈。

证明：由于 $\delta\geqslant 2$，从而由 v_0 出发到 v_k 的路可以向前延伸，又由于 G 有有限个顶点，从而延伸到某一点后再往下延伸时，必然要与已走过的顶点相重，于是得到 G 中的一个圈。证毕。

定理 2.8 若 G 是简单图，且最小度 $\delta\geqslant 2$，则 G 含有长最小为 $\delta+1$ 的圈。

证明：设 G 中最长路为(v_0,v_k)-路，其顶点依次为 $v_0,v_1,v_2,\cdots,v_k$。显然，v_0 的所有邻点均在(v_0,v_k)-路上，不然它与最长路矛盾。取 v_0 所有邻点 v_i 中的下标最大者，并记为 l，显然 $l\geqslant\delta$。于是 $v_0v_1v_2\cdots v_lv_0$ 是 G 中的一个长不小于 $\delta+1$ 的圈。证毕。

有关简单图的路和连通性的其他性质均可参见文献内容。

2.2 最短路问题

最短路问题是重要的最优化问题之一，也是图论研究中的一个经典算法问题，它不仅直接应用于解决生产实践中的众多问题，如管道的铺设、线路的安排、厂区的选址和布局、设备的更新等，而且也经常被作为一种基本工具，用于解决其他的最优化问题以及预测和决策问题。

从数学角度考虑，大量优化问题等价于在一个图中找最短路的问题。在图论中，最短路算法比任何其他算法都解决得更彻底。

定义 2.1 对简单图 G 的每一边 e 赋予一个实数，记为 $w(e)$，称为边 e 的**权**，而每边均赋予权的图称为**赋权图**。

定义 2.2 (u,v)-路的边权之和称为该路的**长**，而 u,v 间路长达到最小的路称顶点 u 和 v 的最短路。

在给定赋权图 G 中，求两个互异顶点间的最短路(径)，简记为**最短路(径)问题**。求最短路问题的应用背景广泛，研究此问题具有实用价值。

最短路问题一般归为两类：一类是求从某个顶点(源点)到其他顶点(终点)的最短路径；另一类是求图中每一对顶点间的最短路径。关于最短路径的研究，目前已有很多算法，但基本上均是以 Dijkstra 和 Floyd 两种算法为基础，因此，对 Dijkstra 算法和 Floyd 算法进行本质的研

究非常必要。

结合图在计算机中的存储形式，在程序运行以后，赋权图的任意两点间的最短距离可用一个矩阵形式表示。设 n 阶有向或无向连通赋权图 $G=(V,E)$，其中顶点集 $V=\{v_1,v_2,\cdots,v_n\}$，边集 $E=\{e_1,e_2,\cdots,e_n\}$。不妨设两顶点 v_i,v_j 间的最短距离为 d_{ij}，其中 $i,j=1,2,\cdots,n$，则图 G 的**最短距离矩阵** $\boldsymbol{d}=(d_{ij})_{n\times n}$ 的元素定义为 $d_{ij}=\begin{cases}d_{ij}, & \text{若 } i\neq j\\ 0, & \text{若 } i=j\end{cases}$。

显然，只要计算机能输出该图的最短距离矩阵 $\boldsymbol{d}$，也就解决了最短路径问题。

2.3 求连通图最短距离矩阵的算法及其 MATLAB 实现

求最短距离矩阵的算法，在图论中是一个应用十分广泛的方法，它不仅可直接应用到运输网路的分析中，而且在图上最优选址类问题中应用也很重要。

设 n 阶无向或有向连通赋权图 $G=(V,E)$，顶点集 $V=\{v_1,v_2,\cdots,v_n\}$，边集 $E=\{e_1,e_2,\cdots,e_n\}$，并且边或弧 v_iv_j 上的权值为 $f(v_iv_j)$，其中 $i,j=1,2,\cdots,n$。

无向图 G 的**权值矩阵** $\boldsymbol{W}=(d_{ij})_{n\times n}$ 定义为

$$d_{ij}=\begin{cases}f(v_iv_j), & \text{若 } v_i \text{ 到 } v_j \text{ 存在边}\\ \infty, & \text{若 } v_i \text{ 到 } v_j \text{ 不存在边}\\ 0, & \text{若 } i=j\end{cases}$$

显然，对于无向图来说，对任意 $i,j=1,2,\cdots,n$，有 $d_{ij}=d_{ji}$，故该图的权值矩阵 $\boldsymbol{W}$ 为对称矩阵。

类似地，可定义有向图的权值矩阵。

【算法用途】

连通赋权图最短距离矩阵的求法。

【算法思想】

首先，输入该赋权图的权值矩阵 $\boldsymbol{W}=(d_{ij})_{n\times n}$，然后，为了求出两点间的最短距离，可按照 Dijkstra 方法，只要反复使用迭代公式 $d_{ij}^{(k)}=\min\limits_{i,j,k\in\{1,2,\cdots,n\}}\{d_{ij}^{(k-1)},d_{ik}^{(k-1)}+d_{kj}^{(k-1)}\}$，就可以得到最终结果，记为

$$\boldsymbol{D}^{(n)}=\begin{pmatrix}d_{11}^{(k)} & \cdots & d_{1n}^{(n)}\\ d_{21}^{(n)} & \cdots & d_{2n}^{(n)}\\ \vdots & & \vdots\\ d_{n1}^{(n)} & \cdots & d_{nn}^{(n)}\end{pmatrix}$$

其中 $d_{ij}^{(n)}$ 为从顶点 v_i 到顶点 v_j 的最短距离的计算结果，称 $\boldsymbol{D}^{(n)}$ 为**最短距离矩阵**。以上亦是图的最短距离矩阵的求解思路。

【程序参数说明】

W 表示图的权值矩阵。

D 表示图的最短距离矩阵。

【算法的 MATLAB 程序】

```
% 连通图中各顶点间最短距离的计算
function D = shortdf(W)
% 对于 W(i,j),若两顶点间存在弧,则为弧的权值,否则为 inf;当 i = j 时,W(i,j) = 0
```

```
n = length(W);
D = W;
m = 1;
while m<=n
    for i = 1:n
        for j = 1:n
            if D(i,j)>D(i,m) + D(m,j)
                D(i,j) = D(i,m) + D(m,j);      % 距离进行更新
            end
        end
    end
    m = m + 1;
end
D;
```

例 2.1 求图 2.1 所示无向图的最短距离矩阵。

解:根据定义,该图的权值矩阵为

$$\boldsymbol{W}=\begin{bmatrix}0 & 1 & 3 & 4\\ 1 & 0 & 2 & \infty\\ 3 & 2 & 0 & 5\\ 4 & \infty & 5 & 0\end{bmatrix}$$

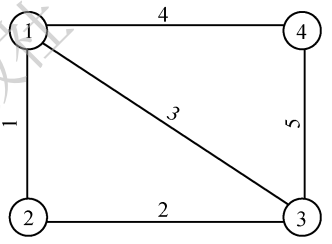


图 2.1 例 2.1 图

运行以下 MATLAB 程序。

```
>> W = [0 1 3 4;1 0 2 inf;3 2 0 5;4 inf 5 0]
>> D = shortdf(W)
```

运行结果如下。

```
D =
     0     1     3     4
     1     0     2     5
     3     2     0     5
     4     5     5     0
```

注:该程序最终输出赋权图的最短距离矩阵 $\boldsymbol{D}$,也就是给出了该图任意两点间的最短路径问题的解决方法。

2.4 求两点间最短路的 Dijkstra 算法及其 MATLAB 实现

最短路径问题要解决的就是求加权图 $G=(V,E,W)$ 中两个给定顶点之间的最短路径。求单源点最短路径的一个著名算法就是 Dijkstra 算法。

设源点为 u_0,目标点为 v_0。Dijkstra 算法的基本思想是:按距离 u_0 由近及远为顺序,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。

2.4.1 Dijkstra 算法

算法步骤如下:

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

步骤 2 对每个 $v\in\overline{S}_i(\overline{S}_i=V\backslash S_i)$,用 $\min\limits_{u\in S_i}\{l(v),l(u)+w(uv)\}$ 代替 $l(v)$,当 u,v 不相邻时,$w(uv)=\infty$。计算 $\min\limits_{v\in\overline{S}_i}\{l(v)\}$,把达到这个最小值的一个顶点记为 u_{i+1},令 $S_{i+1}=S_i\cup\{u_{i+1}\}$。

步骤 3 若 $i=|V|-1$,则停止;若 $i<|V|-1$,则用 $i+1$ 代替 i,转至步骤 2。

算法结束时,从 u_0 到各顶点 v 的距离由 v 的最后一次的标号 $l(v)$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。算法就是不断修改各个点的 T 标号,直至获得 P 标号。若在算法运行过程中,将每一顶点获得 P 标号所由来的边在图上标明,则当算法结束时,u_0 至各个点的最短路也在图上标示出来了。

2.4.2 Dijkstra 算法的 MATLAB 实现

Dijkstra 算法的程序实现如下。

```
% 两点间最短路的 Dijkstra 算法
function [d index1 index2] = Dijkf(a)
% a  表示图的权值矩阵
% d  表示所求最短路的权和
% index1  表示标号顶点顺序
% index2  表示标号顶点索引

%参数初始化
M = max(max(a));
pb(1:length(a)) = 0;
pb(1) = 1;
index1 = 1;
index2 = ones(1,length(a));
d(1:length(a)) = M; d(1) = 0; temp = 1;
%更新 l(v),同时记录顶点顺序和顶点索引
while sum(pb)<length(a)                          %重复步骤 2,直到满足停止条件
    tb = find(pb == 0);
    d(tb) = min(d(tb),d(temp) + a(temp,tb));     %更新 l(v)
    tmpb = find(d(tb) == min(d(tb)));
    temp = tb(tmpb(1));
    pb(temp) = 1;
    index1 = [index1,temp];                      %记录标号顺序
    index = index1(find(d(index1) == d(temp) - a(temp,index1)));
    if length(index)>= 2
        index = index(1);
```

```
    end
    index2(temp) = index;                    % 记录标号索引
end
d;
index1;
index2;
```

例 2.2 某公司在六个城市 $c_1,c_2,\cdots,c_6$ 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(∞表示无直接航路)。请帮助该公司设计一张从城市 c_1 到其他城市间票价最低的路线图。

$$\begin{bmatrix} 0 & 50 & \infty & 40 & 25 & 10 \\ 50 & 0 & 15 & 20 & \infty & 25 \\ \infty & 15 & 0 & 10 & 20 & \infty \\ 40 & 20 & 10 & 0 & 10 & 25 \\ 25 & \infty & 20 & 10 & 0 & 55 \\ 10 & 25 & \infty & 25 & 55 & 0 \end{bmatrix}$$

解题思路：用矩阵 $\boldsymbol{a}_{n\times n}$($n$ 为顶点个数)存放各边权的邻接矩阵，行向量 **pb**，**index1**，**index2**，***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

$$pb_i = \begin{cases} 1, & \text{当第 } i \text{ 顶点已标号} \\ 0, & \text{当第 } i \text{ 顶点未标号} \end{cases}$$

index2_i 存放从始点到第 i 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 点最短通路的值。

经过 MATLAB 求解，得到

$$\boldsymbol{d} = [0,35,45,35,25,10]$$
$$\textbf{index1} = [1,6,5,2,4,3]$$
$$\textbf{index2} = [1,6,5,6,1,1]$$

以从 c_1 到 c_2 为例说明如何使用这一结果。c_1 到 c_2 的最短距离是 35，c_1 到 c_2 最短路径的第二个顶点之前的一个顶点是 6，c_1 到 c_6 最短路径的第六个顶点之前的一个顶点是 1，那么 c_1 到 c_2 的最短路径是 $c_1 \to c_6 \to c_2$。

附：求从第一个城市 c_1 到其他城市最短路径的 MATLAB 源程序如下。

```
>> M = 10000;
a(1,:) = [0,50,M,40,25,10];
a(2,:) = [zeros(1,2),15,20,M,25];
a(3,:) = [zeros(1,3),10,20,M];
a(4,:) = [zeros(1,4),10,25];
a(5,:) = [zeros(1,5),55];
a(6,:) = zeros(1,6);
a = a + a';
>> [d index1 index2] = Dijkf(a)
```

运行结果如下。

```
d =
     0    35    45    35    25    10
index1 =
     1     6     5     2     4     3
index2 =
     1     6     5     6     1     1
```

2.5 求两点间最短路的改进的 Dijkstra 算法及其 MATLAB 实现

由于经典的 Dijkstra 算法存在着许多不足，尤其是当节点数很大时，该算法会占用大量存储空间，并且该算法需要计算从起点到每一个节点的最短路径，这也降低了程序运行的效率。

针对 Dijkstra 算法存在的问题，研究人员提出了两个改进的 Dijkstra 算法。

2.5.1 Dijkstra 矩阵算法 Ⅰ

Dijkstra 矩阵算法 Ⅰ 比 Dijkstra 算法更容易在计算机上实现，它能够计算加权图中任意两顶点之间的最短距离。该算法的基本思想是将加权图 $G(V,E)$ 存储在矩阵 $\boldsymbol{A}=(a_{ij})_{n\times n}$ 里，该矩阵可定义为

$$a_{ij}=\begin{cases}0, & \text{若 } i=j\\ w_{ij}, & \text{若 } i\neq j,\text{顶点 } v_i \text{ 与 } v_j \text{ 有连边}\\ \infty, & \text{若 } i\neq j,\text{顶点 } v_i \text{ 与 } v_j \text{ 无连边}\end{cases}$$

其中，n 为图 G 的顶点个数，w_{ij} 为边 v_iv_j 的权重。

将 Dijkstra 算法的思想应用于此矩阵的第 k 行，可求出顶点 v_k 到其他各顶点的最短距离，将最短距离仍保存在矩阵 $\boldsymbol{A}$ 的第 k 行，其中 $k=1,2,\cdots,n$。当算法结束时，矩阵 $\boldsymbol{A}$ 的元素值就是任意两顶点之间的最短距离。

2.5.2 Dijkstra 矩阵算法 Ⅱ

Dijkstra 矩阵算法 Ⅰ 只是简单地将 Dijkstra 算法的思想应用到矩阵的每一行，这样做有很多的重复计算，效率不高。为了提高效率，又提出下面的**Dijkstra 矩阵算法 Ⅱ**，步骤如下。

步骤 1 输入加权图，存储在矩阵 $\boldsymbol{A}=(a_{ij})_{n\times n}$ 里。

步骤 2 对矩阵 $\boldsymbol{A}$ 进行操作，求任意两顶点间的最短距离。算法的求解过程如下。

```
循环。k 以 1 为步长，从 1 到 n-1，执行
    (2.1)
    b←{1,2,…,k-1,k+1,k+2,…,n};
    kk←length(b);
    a_id←k;
    b1←{k+1,k+2,…,n};
    kk1←length(b1);
    (2.2)
```

```
    循环。反复执行下列语句,直到 kk = 0。
        (2.2.1)
        循环。j 以 1 为步长,从 1 到 kk1,执行
            te←a(k,a_id) + a(a_id,b1(j));
            若 te<a(k,b1(j)),a(k,b1(j))←te。
        (2.2.2)
        miid←1
        (2.2.3)
        循环。j 以 1 为步长, 从 2 到 kk,执行
            若 a(k,b(j))<a(k,b(miid)),miid←j。
        (2.2.4)
        a_id←b(miid);
        b←[b(1:miid - 1),b(mid + 1:kk)];
        % 在数组 b[]中去掉一个元素
        kk←length(b);
        % 数组 b[]的长度减少了 1
        若 a_id>k,执行
            miid1←find(b1 == a_id);
            b1←[b1(1:miid1 - 1),b1(miid1 + 1:kk1)];
            kk1←length(b1);
    (2.3)
    循环。j 以 1 为步长,从 k + 1 到 n,执行
        a(j,k)←a(k,j)。
```

步骤 3 算法结束。

算法Ⅱ与算法Ⅰ比较,在步骤(2.2.1)处循环的次数随着 k 的增加而减少,循环体的执行总次数会减少一半。

Dijkstra 矩阵算法Ⅱ的 MATLAB 实现如下。

```
function a = dij2_m(a)
% a(输入量)表示图的邻接矩阵
% a(输出量)表示所求最短路的距离矩阵

% 建立邻接矩阵,若不是对称矩阵,则变为对称矩阵
n = length(a);
for i = 2:n
    for j = 1:(i - 1)
        a(i,j) = a(j,i);
    end
end
% The main program
for k = 1:(n - 1)
    % 步骤(2.1)
    b = [1:(k - 1),(k + 1):n];
```

```
    kk = length(b);
    a_id = k;
    b1 = [(k + 1):n];
    kk1 = length(b1);
    % 步骤(2.2)
    while kk>0
        % 步骤(2.2.1)
        for j = 1:kk1
            te = a(k,a_id) + a(a_id,b1(j));
            if te<a(k,b1(j))
                a(k,b1(j)) = te;
            end
        end
        % 步骤(2.2.2)
        miid = 1;
        % 步骤(2.2.3)
        for j = 2:kk
            if a(k,b(j))<a(k,b(miid))
                miid = j;
            end
        end
        % 步骤(2.2.4)
        a_id = b(miid);
        b = [b(1:(miid - 1)),b((miid + 1):kk)];
        kk = length(b);
        if a_id>k
            miid1 = find(b1 == a_id);
            b1 = [b1(1:(miid1 - 1)) b1((miid1 + 1):kk1)];
            kk1 = length(b1);
        end
    end
    % 步骤(2.3)
    for j = (k + 1):n
        a(j,k) = a(k,j);
    end
end
a;
```


例 2.3 计算图 2.2 中顶点 v_1 到其他顶点的最短距离。

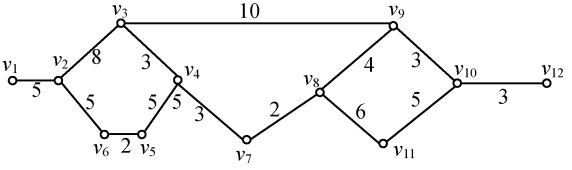


图 2.2　例 2.3 图

用 MATLAB 求解的主程序如下。

```
n = 12;
a = ones(n) + inf;
for i = 1:n
    a(i,i) = 0;
end
a(1,2) = 5;
a(2,3) = 8;
a(2,6) = 5;
a(3,4) = 3;
a(3,9) = 10;
a(4,5) = 5;
a(4,7) = 3;
a(5,6) = 2;
a(7,8) = 2;
a(8,9) = 4;
a(8,11) = 6;
a(9,10) = 3;
a(10,11) = 5;
a(10,12) = 3;
>> dij2_m(a)
```

计算结果如下。

```
a =
     0     5    13    16    12    10    19    21    23    26    27    29
     5     0     8    11     7     5    14    16    18    21    22    24
    13     8     0     3     8    10     6     8    10    13    14    16
    16    11     3     0     5     7     3     5     9    12    11    15
    12     7     8     5     0     2     8    10    14    17    16    20
    10     5    10     7     2     0    10    12    16    19    18    22
    19    14     6     3     8    10     0     2     6     9     8    12
    21    16     8     5    10    12     2     0     4     7     6    10
    23    18    10     9    14    16     6     4     0     3     8     6
    26    21    13    12    17    19     9     7     3     0     5     3
    27    22    14    11    16    18     8     6     8     5     0     8
    29    24    16    15    20    22    12    10     6     3     8     0
```


2.6 求两点间最短路的 Warshall－Floyd 算法及其 MATLAB 实现

设图 $G=(V,E)$，顶点集记作$(v_1,v_2,\cdots,v_n)$，G 的每条边赋有一个权值，w_{ij} 表示边 v_iv_j 上的权，若 v_i,v_j 不相邻，则令 $w_{ij}=+\infty$。

Warshall－Floyd 算法简称**Floyd 算法**，它利用了动态规划算法的基本思想，即若 d_{ik} 是顶点 v_i 到顶点 v_k 的最短距离，d_{kj} 是顶点 v_k 到顶点 v_j 的最短距离，则 $d_{ij}=d_{ik}+d_{kj}$ 是顶点 v_i 到顶点 v_j 的最短距离。

2.6.1 Floyd 算法的基本思想

对于任何一个顶点 $v_k \in V$，顶点 v_i 到顶点 v_j 的最短路经过顶点 v_k 或者不经过顶点 v_k。比较 d_{ij} 与 $d_{ik}+d_{kj}$ 的值。若 $d_{ij}>d_{ik}+d_{kj}$，则令 $d_{ij}=d_{ik}+d_{kj}$，保持 d_{ij} 是当前搜索的顶点 v_i 到顶点 v_j 的最短距离。重复这一过程，最后当搜索完所有顶点 v_k 时，d_{ij} 就是顶点 v_i 到顶点 v_j 的最短距离。

2.6.2 Floyd 算法的基本步骤

令 d_{ij} 是顶点 v_i 到顶点 v_j 的最短距离，w_{ij} 是顶点 v_i 到 v_j 的权。Floyd 算法的步骤是：

步骤 1 输入图 G 的权矩阵 $\boldsymbol{W}$。对所有 i,j，有 $d_{ij}=w_{ij}$，$k=1$。

步骤 2 更新 d_{ij}。对所有 i,j，若 $d_{ik}+d_{kj}<d_{ij}$，则令 $d_{ij}=d_{ik}+d_{kj}$。

步骤 3 若 $d_{ii}<0$，则存在一条含有顶点 v_i 的负回路，停止；或者 $k=n$ 停止，否则转到步骤 2。

2.6.3 Warshall－Floyd 算法的 MATLAB 实现

Floyd 算法的 MATLAB 程序如下。

```
function [P u] = f_path(W)
% W 表示权值矩阵
% P 表示最短路
% u 表示最短路的权和

%初始化，步骤 1
n = length(W);
U = W;
m = 1;
%  步骤 2
while m<=n                                          %判断是否满足停止条件
    for i = 1:n
        for j = 1:n
            if U(i,j)>U(i,m) + U(m,j)
                U(i,j) = U(i,m) + U(m,j);           %更新 dij
            end
        end
    end
    m = m + 1;
end
u = U(1,n);
```

```
% 输出最短路的顶点
P1 = zeros(1,n);
k = 1;
P1(k) = n;
V = ones(1,n) * inf;
kk = n;
while kk~ = 1
    for i = 1:n
        V(1,i) = U(1,kk) - W(i,kk);
        if V(1,i) = = U(1,i)
            P1(k + 1) = i;
            kk = i;
            k = k + 1;
        end
    end
end
k = 1;
wrow = find(P1~ = 0);
for j = length(wrow):( - 1):1
    P(k) = P1(wrow(j));
    k = k + 1;
end
P;
```

例 2.4 计算图 2.3 中顶点 v_0 到顶点 v_7 的最短距离和最短路。

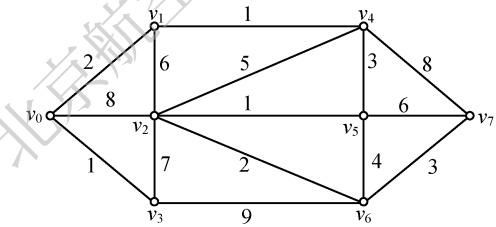


图 2.3 例 2.4 图

MATLAB 实现如下。

```
% 建立权值矩阵
>> W = [0 2 8 1 Inf Inf Inf Inf;
2 0 6 Inf 1 Inf Inf Inf;
8 6 0 7 5 1 2 Inf;
1 Inf 7 0 Inf Inf 9 Inf;
Inf 1 5 Inf 0 3 Inf 8;
Inf Inf 1 Inf 3 0 4 6;
Inf Inf 2 9 Inf 4 0 3;
```

```
Inf Inf Inf Inf 8 6 3 0];
>> [P u] = f_path(W)
```

计算结果如下。

```
P =
     1     2     5     8
u =
    11
```

由此得到顶点 v_0 到顶点 v_7 的最短距离为 11，最短路为 $v_0 \to v_1 \to v_4 \to v_7$（注：在程序中顶点的标号是 1～8）。

例 2.5 求图 2.4 的顶点 v_1 到顶点 v_{11} 的最短距离和最短路。

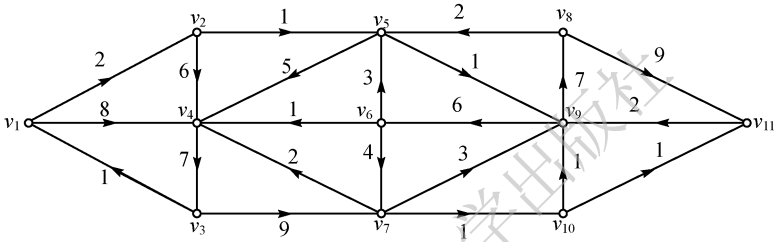


图 2.4 例 2.5 图

MATLAB 实现如下。

```
>> W = [0 2 Inf 8 Inf Inf Inf Inf Inf Inf Inf;
Inf 0 Inf 6 1 Inf Inf Inf Inf Inf Inf;
1 Inf 0 Inf Inf Inf 9 Inf Inf Inf Inf;
Inf Inf 7 0 Inf Inf Inf Inf Inf Inf Inf;
Inf Inf Inf 5 0 Inf Inf 1 Inf Inf;
Inf Inf Inf 1 3 0 4 Inf Inf Inf Inf;
Inf Inf Inf 2 Inf Inf 0 Inf 3 1 Inf;
Inf Inf Inf Inf 2 Inf Inf 0 Inf Inf 9;
Inf Inf Inf Inf Inf 6 Inf 7 0 Inf Inf;
Inf Inf Inf Inf Inf Inf Inf Inf 1 0 1;
Inf Inf Inf Inf Inf Inf Inf Inf 2 Inf 0];
>> [P u] = f_path(W)
```

计算结果如下。

```
P =
     1     2     5     9     6     7    10    11
u =
    16
```

由此得到顶点 v_1 到顶点 v_{11} 的最短距离为 16，最短路为 $v_1 \to v_2 \to v_5 \to v_9 \to v_6 \to v_7 \to v_{10} \to v_{11}$。

例 2.6 求图 2.5 所示无向图的最短距离。

解:根据定义,该图的权值矩阵为

$$\boldsymbol{W}=\begin{bmatrix} 0 & 23 & \infty & 12 & \infty & 6 \\ 23 & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 8 & \infty & \infty \\ 12 & \infty & 8 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & 3 \\ \infty & \infty & \infty & \infty & 3 & 0 \end{bmatrix}$$

运行以下 MATLAB 程序。

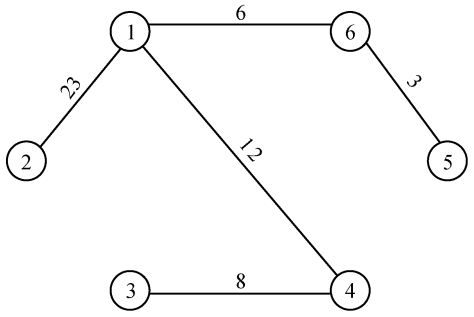


图 2.5 例 2.6 图

```
>> W=[0 23 Inf 12 Inf 6;23 0 Inf Inf Inf Inf; Inf Inf 0 8 Inf Inf; 12 Inf 8 0 Inf Inf ; Inf Inf Inf Inf 0 3; Inf Inf Inf Inf 3 0];
>> [P u]=f_path(W)
```

运行结果如下。

```
P =
     1     6
u =
     6
```


2.7 求任意两点间最短路的算法及其 MATLAB 实现

对于连通的无向或有向赋权图,任意两个顶点间是否都存在路或有向路?若存在不止一条,哪条的边权和最小?这条最短有向路或最短路顺序经过哪些顶点?以上这些问题经常是交通运输网路等实际生活中急需解决的。

已知图 $G=(V,E)$的各顶点之间的距离矩阵 $\boldsymbol{W}$ 及任意两顶点 k_1 和 k_2,不妨设 k_1 为始点、k_2 为终点,本节的目的是:找到一条由始点 k_1 到终点 k_2 的最短路,并给出该总路的长 u 及此路所经过的全部顶点的标号。

【算法用途】

图中任意两点间最短路的求法。

【算法思想】

利用求最短路的 Warshall-Floyd 算法的思想。首先,求得最短距离矩阵;然后,求任意给定两个顶点间的最短路所包含的顶点。

【程序参数说明】

W 表示权值矩阵。

k1 表示始点。

k2 表示终点。

P 表示两顶点 k1,k2 之间的最短路,顶点以经过次序进行排序。

u 表示运行后所得最短路的距离。