ACM-ICPC程序设计系列

图论及应用

主编 冯林 金博 于瑞云

哈爾濱工業大學出版社
HARBIN INSTITUTE OF TECHNOLOGY PRESS

ACM-ICPC程序设计系列

图论及应用

● 主　编　冯　林　金　博　于瑞云

● 副主编　姚翠莉　孟繁军　鲁静轩

哈爾濱工業大學出版社

HARBIN INSTITUTE OF TECHNOLOGY PRESS

内容简介

本书主要介绍 ACM－ICPC 比赛中涉及的图论，其中包括许多实际问题的抽象表示与求解，以及部分图论理论内容的证明。全书共分 6 章，第 1 章介绍了图论的基础知识，包括基础概念、存储方法和遍历方法；第 2 章介绍了有关树的问题，着重讲解生成树和一些树上特殊点集的求法；第 3 章介绍了最短路径问题，包括几种通用算法和特殊图上的算法；第 4 章介绍图论中有关连通性的问题，包括有向图的强连通、无向图的双连通及其扩展问题；第 5 章介绍网络流解法，包括几种常用的网络流算法和对于问题如何抽象成网络流模型的经验方法；第 6 章介绍二分图的相关问题，重点为二分图的匹配及其变种问题。本书的内容基本满足 ACM－ICPC 比赛对于图论方面的要求，讲解清晰易懂，代码规范，例题丰富。

图书在版编目(CIP)数据

图论及应用/冯林，金博，姚翠莉主编. —哈尔滨：哈尔滨工业大学出版社，2012.3

(ACM－ICPC 程序设计系列)

ISBN 978-7-5603-3291-8

Ⅰ.①图… Ⅱ.①冯… ②金… ③姚… Ⅲ.①图论 Ⅳ.O157.5

中国版本图书馆 CIP 数据核字(2011)第 093069 号

策划编辑 赵文斌 杜 燕
责任编辑 范业婷
出版发行 哈尔滨工业大学出版社
社 址 哈尔滨市南岗区复华四道街 10 号 邮编 150006
传 真 0451-86414749
网 址 http://hitpress.hit.edu.cn
印 刷 哈尔滨市工大节能印刷厂
开 本 787mm×960mm 1/16 印张 15.75 字数 311 千字
版 次 2012 年 3 月第 1 版 2012 年 3 月第 1 次印刷
书 号 ISBN 978-7-5603-3291-8
定 价 32.00 元

《ACM－ICPC程序设计系列》编委会

序　言

ACM－ICPC(ACM 国际大学生程序设计竞赛)被称为计算机领域的奥林匹克,是计算机领域高水平的智力角斗场。ACM－ICPC 于 1970 年起源于北美,因涉及知识面广、注重实践,并有着公正的竞赛机制和颇有趣味的比赛过程,在全球范围内迅速流行。其命题的考查范围除了程序设计、算法、数据结构、操作系统、计算机网络、编译原理等计算机专业相关学科知识外,还有离散数学、初等数论、组合数学、图论、计算几何等相关的数学知识,而且范围还在持续扩大之中。

ACM－ICPC 程序设计系列是编者们在多年参与 ACM－ICPC 的过程中,对竞赛、训练的有关数学内容进行总结,从学科专题的角度集结而成。本套丛书包含组合数学、初等数论、计算几何、图论和程序设计训练题解等内容,从竞赛的角度出发,介绍其中的基本概念和研究方法,主要以应用为目的。

组合数学源远流长,源自古代的数学游戏和美学消遣,并以无穷的魅力激发人们的聪明才智和数学兴趣。组合数学涉及内容广泛,知识点庞杂,与很多数学分支有交叉,主要研究将一些元素安排成种种集合的问题,被广泛地应用在计算机科学、编码和密码学、物理、生物等学科以及交通管理、城市规划、企业管理等领域。初等数论是研究数的规律,是研究整数的性质的数学分支。初等数论就是用初等、朴素的方法去研究数论,是数论的一个最古老的分支。数论在计算机科学和应用数学的发展中得到了广泛应用。比如,在计算方法、代数编码、组合论等方面都广泛使用了初等数论范围内的许多研究成果。计算几何于 20 世纪 70 年代末从算法设计与分析中分化而来,主要研究“几何图形信息(曲面和三维实体)的计算机表示、分析、修改和综合”。计算几何已经成长为一个被广泛认同的学科,在众多的应用领域(如计算机图形学、地理信息系统和机器人学等)都发挥着重要的作用。图论作为一个数学分支,有一套完整的体系和广泛的内容。它以图为研究对象,其应用范围非常广泛,不但应用于自然科学,且在社会科学中也有应用。当用图论来解决实际问题时,几乎都能引出复杂的图论模型,而这些模型一般情况下如果没有计算机的帮助很难分析出来,因此图论的快速发展和推广与计算机科学和信息科学的快速发展是分不开的。

本系列图书以介绍这些学科专题的基本概念和方法为基准,目的是使读者能在短时间

内了解专题的主要内容。整套图书适用于参加初级、中级 ACM - ICPC 和信息学竞赛的学生，对计算机程序设计和算法感兴趣的读者同样有指导意义，同时对在该方向上有所研究的人士也有一定的参考意义，可作为竞赛代表队的培训教材，也可作为相关课程实践教学的教材。

本系列图书是由 ACM - ICPC 中国·东北地区组织委员会组织策划，由哈尔滨工程大学、大连理工大学、东北大学、东北林业大学、东北师范大学、内蒙古师范大学等学校的老师（教练）群策群力共同完成。ACM - ICPC 中国·东北地区组织委员会一直关注着竞赛与实践教学的结合，使大学生们通过这个活动掌握更多的学科知识和提高分析问题、解决问题的能力。本系列图书的出版能够将一些有实际意义的学科专题知识以简洁而清晰的方式最快地介绍给大学生们，可以引起更多程序设计爱好者的兴趣，可以成为 ACM - ICPC 参赛队员攀登道路上的一块垫脚石。

ACM - ICPC 中国·东北地区组织委员会主席
第 34 届 ACM - ICPC 全球总决赛执行主席
哈尔滨工程大学副校长

2012 年 1 月

前　言

图论是数学的一个分支，它以图为研究对象。图论中的图是由若干给定的点及连接两点的线所构成的图形，这种图形通常用来描述某些事物之间的某种特定关系，用点代表事物，用连接两点的线表示相应两个事物间具有这种关系。

本书主要介绍 ACM/ICPC 比赛中涉及的图论，其中包括许多实际问题的抽象表示与求解，以及部分图论理论内容的证明。全书共分 6 章，第 1 章介绍了图论的基础知识，包括基础概念，存储方法和遍历方法；第 2 章介绍了有关树的问题，着重讲解生成树和一些树上特殊点集的求法；第 3 章介绍了最短路径问题，包括几种通用算法和特殊图上的算法；第 4 章主要介绍图论中有关连通性的问题，包括有向图的强连通、无向图的双连通以及其扩展问题；第 5 章主要介绍网络流解法，包括几种常用的网络流算法和对于问题如何抽象成网络流模型的经验方法；第 6 章主要介绍二分图的相关问题，重点为二分图的匹配及其变种问题。本书的内容基本满足 ACM/ICPC 比赛对于图论方面的要求，讲解清晰易懂，代码规范，例题丰富。

本书既可以作为高等院校信息与计算科学、计算机专业及数学相关专业的图论教材，也可以作为高等学校计算机竞赛的培训教材，还可供计算机软硬件研发人员参考。

本书由冯林、金博、姚翠莉主编。第 1 ~ 3 章由辽宁省 ACM/ICPC 程序设计竞赛指导金博、高品编写；第 4 章由姚翠莉、张超编写；第 5 章由姚翠莉、胡骏编写；第 6 章由姚翠莉、陈洋编写。全书由大连理工大学创新实验学院院长冯林教授统稿。此外，本书还得到了内蒙古师范大学孟繁军老师和北华大学鲁静轩老师的大力支持与帮助。

本书在编写中参考了很多国内外相关文献资料，同时得到了大连理工大学创新实验学院领导、东北地区 ACM 竞赛指导委员会的大力支持。同时感谢大连理工大学 ACM 集训队的队员们对本书的算法进行的测试，他们为本书的出版作出了辛勤的劳动，贡献很大。

由于时间和水平所限，书中难免会出现陈述不妥或说明错误之处，欢迎读者批评指正。如果在阅读过程中发现问题，请通过电子邮件或者书信联系我们，我们会尽快改进，并在再版时修正。

联系方式：jinbo@ dlut. edu. cn

编　者

2012 年 1 月于大连

目　　录

第1章 图

本章要点 本章主要介绍与图相关的基本概念，图的存储结构和图的遍历方法及其相关问题。1.1节主要介绍图的定义，介绍几种特殊的图；1.2节主要介绍几种图的表示方法，将一个图存储到计算机中；1.3节主要介绍图的遍历方法及其相关问题。

1.1 图的定义和术语

本书的研究对象是图，那么什么是图呢？

1.1.1 图的定义

定义 1.1 由若干个不同顶点与连接其中某些顶点的边组成的图形称为图。

如图1.1所示的形状就是一个图，图由点和边的集合组成。在图结构中常常将点称为顶点，边是顶点的有序偶对，若两个顶点之间存在一条边，就表示这两个顶点具有相邻关系。需要注意的是，在图的定义中，顶点的位置以及边的曲直长短都无关紧要，而且也没有假定这些顶点和边都要在一个平面内，其需要表示的就是顶点与顶点之间的连接关系。

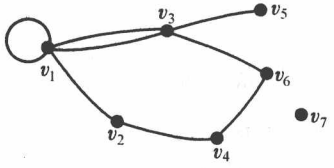

图1.1 图

通常用一个大写字母G表示图，用V表示其所有顶点的集合，E表示其所有边的集合，并记做$G=(V,E)$。V中顶点的个数叫做图的阶。如果V和E都是有限集合，则G称为有限图，否则称为无限图。在ACM竞赛中，只限于有限图相关的问题。

通常，图是实际问题的抽象。如在表示城市之间的道路建设这个问题上，可以把每个城

市看做一个点，连接城市的道路看做一条边。点集 V 和边集 E 组成了整个图 G，用来表示城市之间的道路交通。

为了解决实际问题，引入权的概念。权有两种，点权和边权。在一类问题中，计算从一个城市经过一定道路到另一个城市这个实际问题上，用点表示城市，边表示城市之间的道路，每条路的长度可以作为图上对应边的权值，也就是边权。而在另一类问题中，如果用道路上的收费站作为图中的点，连接收费站的道路作为图中的边，通过收费站所要支付的费用就可以作为图中的点所具有的权值，就是点权。

在描述有关图的算法效率的时候，通常以图中顶点的个数 n 和边的个数 m 来度量输入的规模。

1.1.2 特殊的图

定义 1.2 如果图 G 的点集 V 只含有一个节点，则称 G 为平凡图。

定义 1.3 如果对于图 $G=(V,E)$ 和 $G'=(V',E')$，G' 的顶点集 V' 是 G 的顶点集 V 的一个子集，G' 的边集 E' 是 G 的边集 E 的一个子集，则 G' 是 G 的子图。如果 $G'\neq G$，则图 G' 是图 G 的真子图。如果 $V'=V$，则 G' 是 G 的生成子图。

平凡图是任何图的子图。

定义 1.4 如果一条边两端是图中的同一个顶点，称这条边为自环。

定义 1.5 如果图 G 中没有自环，且每两个顶点之间最多有一条边，称图 G 为简单图。

定义 1.6 如果图 G 为简单图且图中任意两点之间都有一条边，就称 G 为完全图。

定义 1.7 如果图 G 为简单图，且它的顶点集合 V 是由两个没有公共元素的子集 $X=\{x_1,x_2,\cdots,x_n\}$ 与 $Y=\{y_1,y_2,\cdots,y_m\}$ 组成的，并且 x_i 与 $x_j(1\leqslant i,j\leqslant n)$，$y_s$ 与 $y_t(1\leqslant s,t\leqslant m)$ 之间没有边连接，称 G 为二分图，也称二部图。如果 X 集合中的点与 Y 集合中的每个点都有边相连，则称图 G 为完全二部图。

定义 1.8 如果图 G 是一个 N 个顶点的简单图，从 N 阶完全图中把属于 G 的边全部去掉后，得到的称为 G 的补图，记做 $\overline{G}$。$\overline{G}$ 和 G 互为补图。

定义 1.9 如果图 $G_1=(V_1,E_1)$ 和图 $G_2=(V_2,E_2)$ 的顶点之间可以建立起一对一的对应，并且当且仅当 G_1 的顶点 v_i 与 v_j 之间有 k 条边相连时，G_2 的相应的 u_i 与 u_j 之间也有 k 条边相连，则 G_1 与 G_2 为同构。

同构的两个图，是没有区别的。两个图同构需要建立两个图的对应关系，这个关系是双射函数。同构的两个图，节点数相等，边数相等，度数相等的节点个数相等。事实上，图 G 和其自身同构，即同构具有自反性；如果图 G_1 和 G_2 同构，那么图 G_2 和 G_1 也同构，即同构具有对

称性;如果图 G_1 和 G_2 同构,图 G_2 和 G_3 同构,那么图 G_1 和 G_3 也同构,即同构具有传递性。

图 1.2(a) 是一个由 5 个点组成的完全图,图 1.2(b) 是一个二分图,图 1.2(c) 和图 1.2(d) 互为补图,同时两个图还同构。

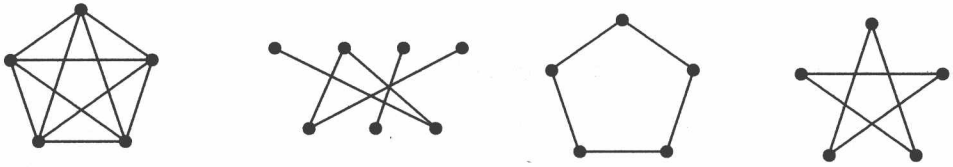

图 1.2 完全图、二分图、补图和同构图

1.1.3 有向图和无向图

定义 1.10 如果给图的每条边规定一个方向,那么得到的图称为有向图。相反,边没有方向的图称为无向图。

在有向图中,与一个节点相关联的边有出边和入边之分,通常将边称做弧,记做 $< v_i, v_j >$,表示从顶点 v_i 到顶点 v_j 有一条边,这条边是顶点 v_i 的一条出边,同时也是顶点 v_j 的一条入边。在无向图中,边记做 (v_i, v_j),它蕴涵着存在 $< v_i, v_j >$ 和 $< v_j, v_i >$ 两条弧。

若有向图中有 n 个顶点,则最多有 $n(n-1)$ 条弧,将具有 $n(n-1)$ 条弧的有向图称做有向完全图。若无向图中有 n 个顶点,则最多有 $n(n-1)/2$ 条弧,将具有 $n(n-1)/2$ 条弧的无向图称做无向完全图。

联结相同两个节点的多于 1 条的无向边称做无向平行边。联结两个节点之间的多于 1 条且方向相同的有向边称做有向平行边。平行边有时亦可称重边。

定义 1.11 顶点的度是指与该点相关联的边的条数。

有向图的顶点的度可分入度和出度。顶点 v 的出边的数目称做顶点 v 的出度,顶点 v 的入边的数目称做顶点 v 的入度。无向图中与顶点 v 相关的边的条数称做顶点 v 的度。

1.1.4 路径与连通

定义 1.12 从 v_0 到 v_k 的一条路径是指一个序列 $e_1, v_1, e_2, v_2, \cdots, e_k$,其中 e_i 的顶点为 v_{i-1} 及 v_i,路径长度是指路径上边或弧的数目 k。

如果一条路径的起止顶点相同,该路径是“闭”的,称为回路,反之,则称为“开”的。如果路径中除起始与终止顶点可以重合外,所有顶点两两不等,这样的路径称为简单路径。

定义 1.13 在无向图中,如果从顶点 v_i 到顶点 v_j 有路径,则称 v_i 和 v_j 连通。如果图中任

意两个顶点之间都连通，则称该图为连通图，否则，将其中的极大连通子图称为连通分量。

定义 1.14　在有向图中，如果对于每一对顶点 v_i 和 v_j，从 v_i 到 v_j 和从 v_j 到 v_i 都有路径，则称该图为强连通图；否则，将其中的极大连通子图称为强连通分量。

1.2　图的存储结构

要将图的信息存到计算机中，需要使用专门设计的数据结构，比较常见的是邻接矩阵、前向星、邻接表、链式前向星这四种方式。另外还有十字链表的方式，由于其建立比较复杂，故在 ACM/ICPC 竞赛中很少用到，这里不再赘述。下面分别使用上面四种方式来存储图1.3所示的图。下文只考虑输入的信息为有向边的信息，如果输入为无向边请读者自行拆成两条有向边处理。

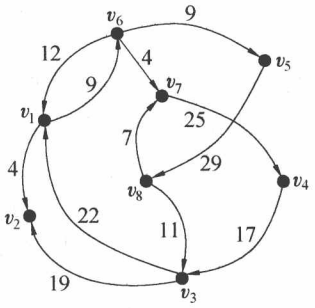

图 1.3　图的存储结构

1.2.1　邻接矩阵

邻接矩阵是表示图的数据结构中最简单也是最常用的一种，对于一个有 n 个点的图，需要一个 $n \cdot n$ 的矩阵，这个矩阵的第 i 行第 j 列的数值表示点 v_i 到 v_j 的距离。对于图 1.3 来说，$n = 8$，在邻接矩阵 map[ ][ ] 中 map[1][2] = 4，map[1][6] = 9，map[3][2] = 19，…，map[8][7] = 7。

邻接矩阵需要初始化，map[i][i] = 0，map[i][j] = INF($i \neq j$)。对于每组读入的 v_i，v_j，w（v_i 为边的起点，v_j 为边的终点，w 为权值），赋值 map[i][j] = w 即可。另外，邻接矩阵的值和边的输入顺序无关，无论以任何顺序输入边，map[ ][ ] 中最后的值是一样的，图 1.3 的邻接矩阵 map[ ][ ] 的值见表 1.1。

表1.1 使用邻接矩阵存储图1.3时各位置的值

Map	1	2	3	4	5	6	7	8
1	0	4	INF	INF	INF	9	INF	INF
2	INF	0	INF	INF	INF	INF	INF	INF
3	22	19	0	INF	INF	INF	INF	INF
4	INF	INF	17	0	INF	INF	INF	INF
5	INF	INF	INF	INF	0	INF	INF	29
6	12	INF	INF	INF	9	0	4	INF
7	INF	INF	INF	25	INF	INF	0	INF
8	INF	INF	11	INF	INF	INF	7	0

对于邻接矩阵来说，初始化需要 $O(n^2)$ 的时间，建图需要 $O(m)$，所以总的时间复杂度是 $O(n^2)$。空间上，邻接矩阵的开销也是 $O(n^2)$，与点的个数有关。

邻接矩阵的优点是实现简单直观，并且可以直接查询点 v_i 与 v_j 间是否有边，如果有，边的权值是多少。缺点是它遍历效率较低，并且不能存储重边；初始化效率低；大图的空间开销大，特别是当 n 比较大（如 $n > 10^5$）的时候，建一个 $n \cdot n$ 的数组是不现实的。对于稀疏图邻接矩阵的空间利用效率也不高，大多数位置为 INF，如表1.1。

1.2.2 前向星

前向星也是一种通过存储边信息的方式存储图的数据结构。它的构造方式非常简单，读入每条边的信息，将边存放在数组中，把数组中的边按照起点顺序排序，前向星就构造完了。为了查询方便，经常会有一个数组存储起点为 v_i 的第一条边的位置。

所需的数据结构如下：

```
int head[maxn];        // 存储起点为 v_i 的第一条边的位置
struct NODE
{
    int from;          // 起点
    int to;            // 终点
    int w;             // 权值
};
NODE edge[maxm];
```

将所有边的信息读入，按照边的起点排序，如果起点相同，对于相同起点的边按终点排序，如果仍有相同，按权值排序。在下面的样例中，使用的 C ++ STL 中的排序函数。

信息存储的主要代码如下。

比较函数：

```
bool cmp(NODE a,NODE b)
{
    if(a.from == b.from && a.to == b.to) return a.w < b.w;
    if(a.from == b.from) return a.to < b.to;
    return a.from < b.from;
}
```

读入数据：

```
cin >> n >> m;
for(i = 0;i < m;i ++) cin >> edge[i].from >> edge[i].to >> edge[i].w;
sort(edge,edge + m,cmp);          // 排序
memset(head, - 1,sizeof(head));
head[edge[0].from] = 0;
for(i = 1;i < m;i ++)
    if(edge[i].from!  = edge[i - 1].from) head[edge[i].from] = i;
                                  // 确定起点为 v_i 的第一条边的位置
```

☞ 遍历代码

```
for(i = 1; i <= n; i ++)
{
    for(k = head[i]; edge[k].from == i && k < m; k ++)
    {
      cout << edge[k].from << ' ' << edge[k].to << ' ' << edge[k].
      w << endl;
    }
}
```