

第十二章 图的模型及算法初步

§ 12.1 导言

在工农业生产，交通运输，通讯和电力领域经常都能看到许多网络，如河道网，灌溉网，管道网，公路网，铁路网，电话线网，计算机通讯网，输电线网等等。还有许多看不见的网络，各种关系网，如状态转移关系，事物的相互冲突关系，工序的时间先后次序关系等等，这些网络都可归结为图论的研究对象——图。其中存在大量的优化问题需要我们解决。还有象生产计划，投资计划，设备更新等问题也可以转化为网络优化的问题。

基本的网络优化问题有：最短路径问题，最小生成树问题，最大流问题和最小费用流问题。在图论这个数学分支，我们已经有有效的算法来解决这些问题。当然这当中好些问题都可以建立线性规划的模型，但有时若变量特别多，约束也特别多，用线性规划的方法求解效率不高甚至不能在可忍受的时间内解决。而根据这些问题的特点，采用网络分析的方法去求解可能会非常有效。例如，在 1978 年，美国财政部的税务分析部门，在对卡特尔税制改革动议做评估的过程中，就有一个 100,000 个约束以上，25,000,000 个变量的问题，若用普通的线性规划求解，预计要花 7 个月的时间。他们利用网络分析的方法，将其分解为 6 个子问题，利用特殊的网络计算机程序，花了大约 7 个小时问题就得到解决。

本实验的目的是让大家掌握如何建立图的模型；了解图的存储结构；学会图的表示与矩阵表示之间的转化；对算法及其复杂性建立一个初步的认识，树立算法有效性的观点。

§ 12. 2 一种表示工具——图

改变问题的描述方式，往往是创造性的启发式解决问题的手段。一种描述方式就好比我们站在一个位置和角度观察目标，有的东西被遮挡住了，但如果我们换一个位置和角度，原来隐藏着的东西就可能被发现。采用一种新的描述方式，可能会产生新思想。图论中的图提供了一种直观，清晰地表达已知信息的方式。它有时就象解小学数学应用题中的线段图一样，能使我们用语言描述时未显示的或不易观察到的特征，关系，直观形象地呈现在我们

们面前，帮助我们分析和思考，激发我们的灵感。

12.2.1 七桥问题

18 世纪东普鲁士哥尼斯堡被普列格尔河分为四块，它们通过七座桥相互连接，如图 12.1(a)。当时，这城里的市民热衷于这样一个游戏：“一个散步者怎样才能从某块陆地出发，经每座桥一次且仅一次回到出发点？”

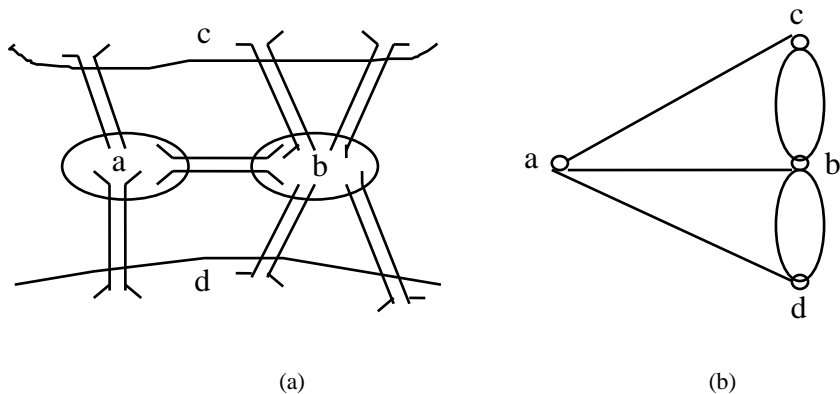


图 12.1

这问题似乎不难，谁都想试一试，但是没人找到答案。后来，有人写信告诉了当时的著名数学家欧拉。千百人的失败，使欧拉猜想，也许，那样的走法根本不可能。1736 年，他证明了自己的猜想。

Euler 把南北两岸和两个岛 a,b,c,d 抽象成四个点，将连接这些陆地的桥用连接相应两点的一条线来表示，于是得到图 12.1(b)。问题就转化为在图 12.1(b)中是否存在从某点（称为顶点）出发经过每条线（称为边）一次且仅一次最终回到出发点的路线。



这个问题是一个几何问题，但它却是欧几里德几何学没有研究过的。欧几里德几何研究的都是与几何图形的长度，面积，角度，位置等有关的性质，而在这里所引出的图形，线段的长短，曲直，顶点的准确位置都无关紧要，重要的是点线之间的连接关系。欧拉认为这是一门新的数学分支，他把这样的图形称为线图，也就是图论的研究对象“图”。

欧拉指出：一个线图中存在通过每边一次且仅一次回到出发点的路线的充要条件是

1) 图要是连通的(即任意两点可由图中的一些边连接起来)。

2) 与图中每一顶点相连的边必须是偶数条。

于是得出结论: 七桥问题无解。

欧拉由七桥问题所引发的, 对图的研究论文是图论的开篇之作, 因此称欧拉为图论之父。

象图 12.1(b)那样的由一些小圆圈(称为顶点 vertex)和一些连接它们的线(称为边 edge)所构成的图称为无向图(undirected graph), 一般用大写字母 G, H 表示。图中顶点的位置, 边的曲直长短都无关紧要, 重要的是顶点与顶点之间的连接关系。图 12.1(b)可以只需列出它的顶点 a, b, c, d 和边的端点对

$e_1=(a, b), e_2=(a, c), e_3=(a, d), e_4=(b, c), e_5=(b, c), e_6=(b, d), e_7=(b, d)$

便可确定。通常用 V, E 分别表图的顶点集, 边集, 图 G 可用 (V, E) 表示。在图 12.1(b)中, $V=\{a, b, c, d\}, E=\{(a, b), (a, c), (a, d), (b, c), (b, c), (b, d), (b, d)\}$ 。

任何元素集及其元素对之间的关系都可用图来描述。例如, 参加某次聚会的人之间的相识关系, 以人为顶点, 当且仅当两顶点所代表的两人相识时, 两顶点间连一边, 所得之图便形象地表达出这群人中哪些人相识。又如设 V 为一个城市中街道的交汇点的集合, E 代表所有连接交汇点的街道段, 则 $G=(V, E)$ 便是图。顶点的位置不必是实际的物理位置, 边的长度也不必与真实的物理长度成比例。还有铁路网中的火车站, 道路交叉点, 道路尽头看作顶点, 铁路看作边, 铁路网就是图。其它的什么公路网, 灌溉网, 管道网, 公路网, 电话线网, 计算机通讯网, 输电线网统统都是图。



一个州的立法机关由许多委员会组成。某些资深的立法委员身兼数职, 因而各委员会的委员互相交迭。说明怎样用图来描述这种委员会委员的互相交迭。如果委员会 A 有委员(编号) 1, 3, 5, 6; B 有 2, 4, 8, 10; C 有 1, 7, 9; D 有 2, 5, 8; E 有 2, 4, 10; F 有 11, 12, 13。试绘出描述委员会委员互相交迭的图。

关于图中顶点与边的几个术语

1) 若边 e 的端点为 u, v , 则称 e 与顶点 u, v 相关联。

2) 若顶点 u, v 之间有边相连, 则称 u 与 v 相邻。

3) 若边 e_1, e_2 与同一顶点相关联, 则称相邻。

4) 端点相同的两边称为重边。两端点为同一个点的一条边称为环。

例如, 在图 12.2 中 e_1 与 v_1 相关联; v_1 与 v_2 相邻, e_1 与 e_2 相邻, e_4 与 e_5 是重边, e_6 是环。

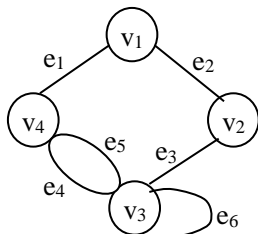


图 12.2

两种特殊图

- 1) 简单图: 无重边, 无环的图。
- 2) 完全图: 任两顶点之间皆有边相连的简单图, 记为 K_n , n 为图的顶点数。

若每条边都有方向, 则称为有向图(directed graph)。你能给出一个有向图描述的实际例子吗?

若给每条边赋予一个或多个实数, 这样的图称为网络。这些数字可以代表距离, 费用, 可靠性或其他的相关参数。

在无向图中, 与顶点 v 相关联的边的数目称为 v 的度数, 记为 $d(v)$; 一般用 $v(G)$ 和 $\varepsilon(G)$ 分别表示图 G 的顶点数和边数。在有向图中, 从顶点 v 引出的边的数目称为 v 的出度, 记为 $d^+(v)$; 指向顶点 v 的边的数目称为 v 的入度, 记为 $d^-(v)$ 。



在一个足球赛季中, 某足球联合会内每两队间比赛一次。假定每次比赛结束后总有一方获胜 (即无平局)。如何用一有向图来表示该赛季所有比赛的结局 (即每次那队获胜)。设 A, B, C, D 为该足球联合会各队, A 胜 B, D ; B 胜 C, D ; C 胜 A ; D 胜 C , 试绘出相应的图。根据所绘之图, 确定比赛名次。

12. 2. 2 一个时间安排问题

学校要为一年级的研究生开设六门基础数学课: 统计(S), 数值分析(N), 图论(G), 矩阵论(M), 随机过程(R)和数理方程(P)。按培养计划, 注册的学生必须选修其中的一门以上, 你作为教务管理人员, 要设法安排一个课表, 使每个学生所选的课程, 在时间上不会发生冲突。由于他们都要必修的外语和自然辩证法课程占用了许多时间, 可供排课的时段不多, 可供使用的教室足够多, 因此, 有些课可能需要安排在同样的时段。上述六门数学课中任何两

门，只要不被同一个学生选择，可以安排在同样的时间段。

假设六门数学课的选课名单如表 12.1，

表 12.1

S	N	G	M	R	P
李春兰	陈奇峰	化范文	张星	赵小民	许茂
郑文国	刘云	李出荣	夏雯	息志强	陈俊
姚南	刘元元	张惠	邵桂芳	陈修建	周清武
陈奇峰	黄大度	赵云	王学权	邹鑫	樊雪峰
王润惠	董舟	曹林军	单富民	刘元兵	刘伟
邹文燕	邹鑫	胡志强	董舟	杨成宝	甄军
万华	赵云	张敏	杨欣	邱吉洲	姜永东
李祖军	王凯	陈修建	吴军		
黄大度	李白彤	欧阳金	查小辉		
史武军	甄军	李晓	王坚		
刘昆	李欣	李白彤	程静波		
欧阳金	陈俊	万华	邹文燕		
郭志伟	于洪	曾光伟	卫迎新		

哪些课程可以安排在同样的时段上，将六门课在四个不同时段上的安排填入表 12.2。

表 12.2

时段	课程
1	
2	
3	
4	

如果只提供三个不同的时段，是否存在这样的安排，使每个学生都能上到他所选的课程。要作出这样的安排的最少时段数是多少？

我们可以借助于图来分析和解决这个问题。用六个顶点代表六门课，当且仅当两门课同时被一个以上学生选择而不能安排在同样的时段时，在相应的两顶点间连边，例如，S 和 N 同时被黄大度同学选修，所以在 S 与 N 之间要连一条边（见图 12.3）

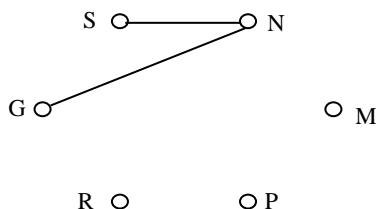


图 12.3



当且仅当两门课同时有一个以上学生选修时，在相应的两顶点间连一条边，完成图 12.3 中的图。

现在我们在作好的图上来操作，用 1, 2, 3, 4 这四个数来标记该图的顶点，使标记相同的顶点不相邻，则标记相同的顶点对应的课程可安排在这样的时段上，因此标记为 1, 2, 3, 4 的顶点对应的课程可分别安排在四个不同的时段。

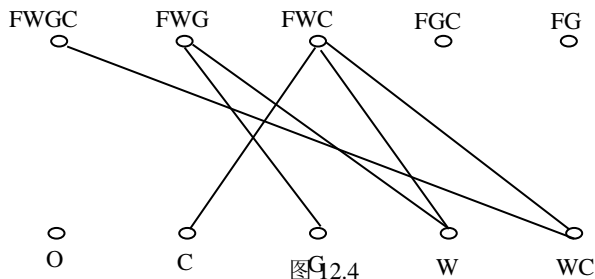


用为顶点标记的方法，求出使学生上课不发生冲突且要求所花的时段数尽可能少的课程时间安排。

12. 2. 3 人，狼，羊，菜渡河问题

一个摆渡人 F 希望用一条小船把一只狼 W，一头羊 G 和一篮白菜 C 从一条河的北岸渡到南岸去，而船小只能容纳 F, W, G, C 中的两个，绝不能在无人看守的情况下，留下狼和羊在一起，羊和白菜在一起，应怎样渡河才能将狼，羊，白菜都运过去？

我们考虑在人狼羊菜渡河的过程中河北岸状态的变化情况，最初的状态是人狼羊菜，最终的状态是空状态，中间的状态为人狼羊菜的不同组合。一种渡河方案就对应了一条从人狼羊菜状态经过一些中间状态到空状态的状态转移链。为了找出最佳渡河方案(对应于人狼羊菜状态到空状态的最短链)，需考虑河北岸的所有可能状态，以及这些状态间经一次摆渡的相互转移关系。



人，狼，羊，菜的任意不同组合共有：16 种，其中狼羊菜，狼羊，羊

菜不允许,从而人,人狼,人菜三种情况也不会出现(因它们分别对应于南岸的三个不允许状态)。剩下 10 个允许状态,用顶点表示河北岸的各允许状态,两顶点连线当且仅当相应的两状态可经一次摆渡相互转移,例如,在北岸为 FWGC 状态下,摆渡人 F 载着狼 G 到河的南岸,北岸就变成 WC,反之,在北岸为 WC 状态下,摆渡人 F 载着狼 G 回到河的北岸,北岸就变成 FWGC,因此要在 FWGC 与 WC 之间连一边,见图 12.4, O 表示空状态。



当且仅当两状态可经一次摆渡相互转移时,对应的两顶点之间连一条边,完成图 11-4 中的图。象 FWGC—WC—FWC—W—

FWG 这样首尾相连的边构成的链路,称为**路径**。寻求图中从顶点“FWGC”到顶点“O”的最短路径,这样的路径有几条? 求出最优的渡河方案。



1. 后面要专门介绍求图中最短路径的一般方法,使我们能用计算机求解,从而具有推广价值。

2. 用顶点代表状态,边代表状态间的转移,这样的状态转移图模型颇具代表性。任何具有有限个状态的多步决策问题都可以类似地建立图的模型,利用图论工具来解决。下一章还可看到一些例子。

§ 12.3 图的矩阵表示方法

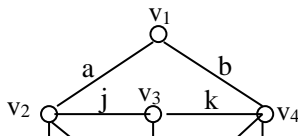
任何事物群体及其元素之间的二元关系均可用图来描述,非常直观,形象,能使一些抽象的关系跃然纸上,助你思考,激发灵感。另一方面,若想借助计算机来解决有关图的问题,又需将形转化为数,矩阵或数组就是一种表示图的很好工具,可作为图在计算机里的存储结构。

12.3.1 邻接矩阵

无向图的邻接矩阵: $A=(a_{ij})_{n \times n}$, 其中

$$a_{ij} = \begin{cases} 1 & \text{当 } v_i \text{ 与 } v_j \text{ 相邻,} \\ 0 & \text{当 } v_i \text{ 与 } v_j \text{ 不相邻} \end{cases}$$

图 12.5 中的图对应的邻接矩阵为





写出 MATLAB 环境下建立带权邻接矩阵的命令 M 文件。并由此产生图 12.7 的带权邻接矩阵。

11. 3. 2 关联矩阵

无向图的关联矩阵: $M=(m_{ij})_{n \times m}$, 其中

$$m_{ij} = \begin{cases} 1 & \text{若 } v_i \text{ 与 } e_j \text{ 相关联,} \\ 0 & \text{若 } v_i \text{ 与 } e_j \text{ 不相关联} \end{cases}$$

图 12.5 中无向图的关联矩阵为:

$$\begin{array}{c} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} \begin{bmatrix} a & b & c & d & e & f & g & h & i & j & k \\ \begin{matrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{matrix} \end{bmatrix} \end{array}$$

注意关联矩阵每行的元素之和为对应顶点的次数, 每列的元素之和为 2。

有向图的关联矩阵: $M=(m_{ij})_{n \times m}$, 其中 m_{ij} 的取值为 1, -1, 0, 分别对应于 v_i 是 e_j 的起点, v_i 是 e_j 的终点和 v_i 不是 e_j 的端点三种情形。

11. 3. 3 边权矩阵

定义一个 $2 \times m$ 的矩阵 E , 第一, 二行分别存放边的起点和终点。若第 i 条边 e_i 的起点和终点分别为 v_j, v_k , 则 $E(1,i)=j$, $E(2,i)=k$ 。例如, 图 11-5 中的图对应的边权矩阵为:

$$E = \begin{array}{c} \begin{matrix} a & b & c & d & e & f & g & h & i & j & k \end{matrix} \\ \begin{bmatrix} 1 & 1 & 2 & 2 & 3 & 4 & 4 & 5 & 6 & 2 & 3 \\ 2 & 4 & 5 & 6 & 6 & 6 & 7 & 6 & 7 & 3 & 4 \end{bmatrix} \end{array}$$

对加权图, 只需增加一行来存放各条边上的权, 这样的矩阵称为**边权矩阵**。



写出 MatLab 环境下由加权图的边权矩阵表示转化为带权邻

接矩阵表示的 M 文件函数。

11.4 算法

在现代社会，许多问题的解决都离不开计算机程序，而算法则是计算机程序的“心脏”。我们不仅需要算法，而且更需要好的算法。对于一个问题，找到一个解决方法，我们会很高兴。但若只停留于我们最初得到的算法，可能比继续去找到更快，更好算法所花的时间更多。例如，用行列式的定义来求一个 20 阶行列式的值，即使是每秒 1000 万次的计算机也要花大约 15 万年的时间。从定义出发求 n 阶行列式的值的算法不可取，它虽然正确，但是无效。因此，寻找一个问题的有效算法非常重要。如何才能简便地判断一个算法的有效性呢？在什么情况下一个问题找不到有效算法来求出正确解？这便是本节的主要任务。

11.4.1 什么是算法

一个算法就是解决某一特定问题的方法，是一系列确定步骤，它必须在有限的时间内终止。

例如，求两个正整数 m 和 n 的最大公因子的 Euclid 算法

Euclid 算法：

输入 正整数 m, n 。

输出 m 和 n 的最大公因子。

第 1 步 求余数

m 除以 n ，令 r 为所得的余数 ($0 \leq r < n$)。

第二步 判断 r 是否为 0。

若 $r=0$ ，则算法终止；否则，转第三步。

第三步 互换。

置 $m \leftarrow n$ ， $n \leftarrow r$ ，返回第一步。



注意



1. “ \leftarrow ”是代替运算（或叫赋值），“ \leftarrow ”的意思是把右边表达式的值赋予左边的变量。如“ $n \leftarrow n+1$ ”表示 n 的当前值增加 1 后，代替 n 的值。

2. 在算法的第三步中，两个代替运算“ $m \leftarrow n$ ”和“ $n \leftarrow r$ ”的次序不能颠倒。

3. 两个变量的值相互交换，可以写成“交换 $m \leftrightarrow n$ ”



提示

表述一个算法一般有两种方式：步骤描述 框图。前面对 Euclid 算法的描述便属于第一种方式。描述算法的详略程度依赖于阅读并使用这一算法的读者。本书中图论部分的算法均采用步骤描述的方式，应用时，再将其细化，转述为程序。

显然，可用 Euclid 算法求其最大公因数的数对有无穷多，每一对正整数都是确定最大公因数这个一般问题的一个**实例**，因此，我们称所有可能实例的集合为一个**问题**。一个正确算法必须对该问题中每一个实例给出正确解。要证明这一点往往非常困难。

Euclid 算法是一种迭代算法，且在有限步后必定终止。这是因为，每步除法中的余数 r 严格比前一步的小。实际上，算法的循环次数一定不大于 n ，这是因为余数序列的最坏情形也就是 $n-1, n-2, \dots, 1, 0$ 。若假设循环一次耗费一个单位时间，则循环次数越多，耗时越多。



在上面的算法中可加入一个计数器去跟踪算法的循环次数，编写用 Euclid 算法求最大公因数的 MATLAB 程序，选取各种数对 (m, n) ，研究一下，大的数所需的循环次数是否也多？

其中的依赖关系怎样？试构造比 n 更好的上界。

11.4.2 算法的时间复杂性分析

什么是有效算法？衡量算法优劣的标准是什么？

通常，对一个算法，人们常用执行该算法所需的时间，和所需的计算机内存容量（即空间）来评价。即用算法的时间复杂性和空间复杂性去衡量它的效率和计算难度。以下主要介绍算法的时间复杂性。



提示

实际上可以有各种不同的标准，如算法的正确性，简单性，最佳性和精确性等。但我们更关心，当问题的规模越来越大时，求解它所需的时间和空间的增长率，即把时间和空间的增长率作为衡量算法优劣的标准。这是因为解题过程不可能无限制地继续下去；计算机的内，外存储器容量也不可能无限大。

1. 时间复杂度

可以用一个整数量表示一个问题输入数据量的大小(或输入长),称为问题的**规模**。例如,行列式的大小可用其阶数 n 来度量,因此 n 就是“求 n 阶行列式值”这个问题的规模;图问题的规模就是其边数或顶点数。

一个算法的**时间复杂度**可以简单地定义为:从输入数据到计算出结果所需的时间。它是问题规模(或输入长) n 的函数,记为 $T(n)$ 。

众所周知,同一算法的计算机执行时间与计算机型号,程序语言及程序员的技术等等有关。因此,这个时间无法精确度量。我们需要基于算法本身来确定算法的时间复杂度 $T(n)$,而不受机型以及这个算法如何编码等等因素的影响。

度量 $T(n)$ 的一个较恰当的方法是,计数这个算法所需的基本操作,这些基本操作包括加、减、比较、乘、除、交换存储位置。更粗略、更宏观地,我们计数这个算法必须执行的步骤数目,它是实例输入长的函数。

例如,求 n 个数最大者的一个算法如下:

```
maxn=-999999
for i=1:n
    if number(i)>maxn
        maxn=number(i)
    end
end
```

n 是该问题的规模(输入长)。第一句的计算时间为一个常数 c_1 ,后面的比较,赋值语句要重复 n 次,所花时间不超过一个常数倍 n ,即 c_2n 。总的计算时间不超过 c_1+c_2n ,这是一个与 n 同阶的量,即有 $c_1+c_2n=O(n)$,这里的“ O ”表示数量级,读作“ n 阶”。我们称 $O(n)$ 为这个算法的时间复杂度。



一般地,若存在正数 C 和 n_0 , 使当 $n \geq n_0$ 时,一个算法的执行时间 $T(n) \leq Cf(n)$, 则称该算法花了 $f(n)$ 阶的时间,记为 $T(n)=O(f(n))$ 。

例:对下面三个简单的程序段,求时间复杂度。

- (1) $x=x+1$
- (2) for $i=1:n$

```

        x=x+1
    end
(3) for i=1:n
        for j=1:n
            x=x+1
        end
    end
end

```

程序(1)只运行一步,执行时间为常数;程序(2)的语句要执行 n 次,计算的时间是 n 的常数倍;程序(3)的语句要执行 n^2 次,计算时间是 n^2 的常数倍。因此,三个程序的计算时间复杂度分别是 $O(1)$, $O(n)$, $O(n^2)$ 。



1. 由于基本操作所需时间不超过一个与 n 无关的常数,这些基本操作包括加,减,乘,除,比较,交换存储位置,则计数这个算法所需的基本操作数 $f(n)$,完成这些基本操作所需时间为 $O(f(n))$ 。



2. 更粗略,更宏观地,若一次循环所需时间不超过一个与 n 无关的常数,则计数这个算法所需的循环次数 $f(n)$,完成这些循环所需时间为 $O(f(n))$ 。

3. $O(f_1(n)) + O(f_2(n)) = O(f_1(n) + f_2(n))$

2. 计算复杂性函数量级的比较

当需要比较两个复杂性函数的量级时,要用到高等数学中关于无穷大量的下述定义:

设有两函数 $f_1(n)$ 与 $f_2(n)$, 令

$$\lim_{n \rightarrow \infty} \frac{f_1(n)}{f_2(n)} = L$$

- (1) 若 $0 < L < \infty$, 称 $f_1(n)$ 与 $f_2(n)$ 同量级, 记为 $O(f_1(n)) = O(f_2(n))$;
- (2) 若 $L = 0$, 则称 $f_1(n)$ 的量级比 $f_2(n)$ 低, 记为 $O(f_1(n)) < O(f_2(n))$;
- (3) 若 $L = \infty$, 则称 $f_1(n)$ 的量级比 $f_2(n)$ 高, 记为 $O(f_1(n)) > O(f_2(n))$ 。

显然有:

$$O(1) < O(\log n) < O(n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$$

一个算法的时间复杂性如果是 $O(n^k)$ (k 是有理数), 则称该算法是**多项式时间算法**或**好算法**。一个算法, 若任何多项式都不是其时间复杂度 $T(n)$ 的上界, 则称该算法是**指数时间算法**或**坏算法**。

时间复杂性的量级比较有何益处呢? 设有 4 种算法 A, B, C, D, 其时间复杂度函数 $T(n)$ 分别为: $\log_{10}n, n^2, n^3, 2^n, n!$ 。设所用计算机的计算速度是 100 万次/秒, 各算法在问题实例大小 n 取几种不同值时, 所需的计算时间见表 11-1。当 $n=60$ 时, 指数时间复杂度和阶乘时间复杂度算法所需的时间以百年计, 是不可能在实际有限的时间内完成的, 是无效的坏算法, 这种算法不能解大问题。而多项式时间复杂度算法所需的时间以分秒计, 这是我们所希望的。把以多项式时间为限界的算法称为**有效算法**。

表 11-1 典型 $T(n)$ 算法在每秒百万次的计算机上的执行时间

时间复杂性函数	大小 n					
	10	20	30	40	50	60
$\log_{10}n$	10-6 秒	1.3*10-6 秒	1.5*10-6 秒	1.6*10-6 秒	1.7*10-6 秒	1.8*10-6 秒
n^2	10-4 秒	0.4*10-3 秒	0.9*10-3 秒	1.6*10-3 秒	2.5*10-3 秒	3.6*10-3 秒
n^3	10-3 秒	0.008 秒	0.027 秒	0.064 秒	0.125 秒	0.216 秒
2^n	10-3 秒	1 秒	17.9 分	12.7 天	35.7 年	366 世纪
$n!$	3.6 秒	771.5 世纪	8.4*10 ¹⁶ 世纪	2.6*10 ³² 世纪	9.6*10 ⁴⁸ 世纪	2.6*10 ⁶⁶ 世纪

可能认为, 现代计算机的发展突飞猛进, 计算速度成百成千倍的增长, 算法效率高已没多大意义。其实不然, 表 11-2 给出了提高计算机运算速度, 对不同量级复杂度函数的算法, 所带来的处理能力提高的情况。

表 11-2 典型 $T(n)$ 算法在不同速度计算机上 1 分钟内能处理的问题大小

时间复杂性函数	现有计算机 1 分中能处理的问题大小	快 100 倍的计算机 1 分中能处理的问题大小	快 1 百万倍的计算机 1 分中能处理的问题大小
$\log_{10}n$	N_1	N_1^{100}	$N_1^{1\,000\,000}$
n^2	N_2	$10N_2$	$1\,000\,N_2$
n^3	N_3	$4.64N_3$	$100N_3$
2^n	N_4	$N_4+6.64$	$N_4+19.93$
$n!$	N_5	N_5 与 N_5+3 之间	N_5 与 N_5+9 之间

从表 11-2 可看出, 提高计算机速度, 多项式时间算法所能处理的问题大小 n 成倍增加, 其计算效率明显提高。而对指数时间算法, 虽然计算机速度提高了 100 倍, 甚至 1 百万倍, 但由于算法本身的笨拙, 其处理的问题大小

n 最多增加一个常数，甚至不增加，没有显示出快速计算机的优势来。

1. 无论计算机速度多么高，功能多么强，指数时间算法不能解大型问题。



提示

2. 在较慢的计算机上用好算法要优于在较快的计算机上用坏算法。

3. 时间复杂度的重要意义在于它能告诉我们在现有技术和给定的时间内能解决多大的问题。

3. P, NPC 与 NPH 问题浅说

在近几十年来，算法复杂性的重要性与日俱增的一个原因是：存在许多还没找到有效算法的问题。也许其中最著名的要数图论中的“旅行推销员问题”，简称“TSP”。即“已给一个 n 个点的完全图，每条边都有一个长度，求总长度最短的经过每个顶点正好一次的封闭回路”。Edmonds, Cook 和 Karp 等人发现，这批难题有一个值得注意的性质，对其中一个问题存在有效算法时，每个问题都会有有效算法。这些问题称为 NP 难题（NP-Hard 或 NPH），迄今为止，这类问题中没有一个找到有效算法。目前倾向于接受 NP 完全问题（NP-Complete 或 NPC）和 NP 难题不存在有效算法这一猜想，认为这类问题的大型实例不能用精确算法求解，必须寻求这类问题的有效的近似算法。

计算复杂性理论源于对判定问题算法的研究。

判定问题：其答案不是“是”就是“否”的问题。如，一个图的两顶点之间存在路径吗？判定问题有三类：P, NP, 和 NPC。

P 类：已有多项式时间算法的判定问题。

NP 类：已有指数时间算法的判定问题，包括 P 类。

NPC 类：是 NP 的一个子集，且其中每一个问题均能由 NP 中的任何问题在多项式时间内转化而成。问题 A 能在多项式时间内转化为问题 B 可理解为，问题 A 有一个算法以问题 B 的算法为子程序，当把每次对 B 算法的调用看作一个基本操作（只花常数时间）时，A 的这个算法是多项式时间的。

在 NPC 问题之外还有一些问题，其难度与 NPC 相当或难度超出 NPC，这就是 NPH 问题。何谓 NPH 问题呢？

NPH 类：若问题 A 不属于 NP 类，已知某一 NPC 问题可在多项式时间之内转化为问题 A，则称 A 为 NP 难题。例如，“TSP”是 NPH 问题。



1. 上述这些还不是严格定义，要准确说明 P, NP 和 NPC 问题的定义需借助确定型图灵机 (deterministic turing machines) 和非确定型图灵机的概念。
2. 在某些文献中也将“NP 难题”与“NP 完全问题”混用，因而也称“TSP”为 NP 完全问题。



1. NPC 类问题是 NP 类问题中最困难的一类问题，若 NPC 类中一个问题有多项式时间算法，则 NP 类中所有问题皆有多项式时间算法。
2. 迄今，已证明为 NPC 的问题已愈千个。其中，比较著名的有：哈密尔顿路径问题，图的着色问题，独立集，顶点覆盖问题，团的问题，三维匹配问题等。

11. 5 实验

11. 5. 1 实验 1：田径赛的时间安排

假设某校的田径选拔赛共设六个项目的比赛，即跳高，跳远，标枪，铅球，100 米和 200 米短跑，规定每个选手至多参加三个项目的比赛。现有七名选手报名，选手所选项目如表 11-3 所示。现在要求设计一个比赛日程安排表，使得在尽可能短的时间内完成比赛。

表 11-3 参赛选手比赛项目表

姓 名	项目 1	项目 2	项目 3
赵 宁	跳 高	跳 远	铅 球
钱 虎	跳 远	100 米	
孙 正	跳 高	200 米	
李 江	200 米	标 枪	铅 球
杨 众	跳 远	铅 球	跳高
刘 平	铅 球	跳 高	200 米
王 跃	标 枪	跳 远	100 米

11. 5. 2 实验 2：国际象棋中马的行走路线

假定有一个 $n \times m$ 矩形国际象棋盘（代替标准的 8×8 棋盘）。是否存在一

连串符合规定的着法，使马可以从某个指定的方格走到另一个指定的方格。

■ 如何画出相应的图，使路中每条路径代表马的一连串符合规定的着法。

■ 试在 3×3 棋盘上画出这样的图。

☞ 在 3×3 棋盘上，是否任意两个方格间都存在这样一连串符合规定的着法？试用图论术语说明。

11. 5. 3 实验 3：寻找有向路径的算法分析

假设我们想确定在一个有向图 G 中，给定一对顶点 s 和 t ，是否存在一条从顶点 s 到顶点 t 的有向路径。解这个问题的一个算法陈述如下：

第一步：Node $\leftarrow s$ 。

第二步：对以顶点 Node 为起点的有向边的终点，标上记号“可达的”，对 Node 标上“已扫描”。

第三步：若顶点 t 已经有了标记“可达的”，则有向路径存在，算法终止。否则，继续。

第四步：在标记为“可达的”，并且还没有扫描的顶点中，任选一点 u ，Node $\leftarrow u$ ，返回第二步。若没有这样的顶点，则有向路径不存在。

■ 若图的存储结构采用关联矩阵，求其时间复杂度。

■ 若图的存储结构采用邻接矩阵，求其时间复杂度。

☞ 这两个由不同数据结构产生的不同算法的时间复杂度不同，哪个算法好？

参考文献：

1. 龚劬，图论与网络最优化算法，重庆大学应用数学系，1998。
2. 周培德，算法设计与分析，机械工业出版社，1991。