高等工科数学系列课程教材

计算技术与程序设计

总主编　孙振绮
主　编　金承日　孙振绮
副主编　丁效华

机械工业出版社

本书介绍了计算机上常用的数值算法和程序设计技术，取材适当，由浅入深，通俗易懂，便于教学．全书共分 8 章，包括误差与算法、程序设计、方程求根、线性代数方程组的解法、代数插值与曲线拟合、数值积分、常微分方程的数值解法、数学物理方程的差分解法．每一节都配有一定数量的习题，书末还附有数值算例的 C 程序．

本书可作为高等学校工科各专业计算方法课程的教材，也可供工程技术人员及其他科技人员参考．

图书在版编目（CIP）数据

计算技术与程序设计/金承日，孙振绮主编．—北京：机械工业出版社，2004.9
（高等工科数学系列课程教材）
ISBN 7-111-15160-7

Ⅰ.计… Ⅱ.①金…②孙… Ⅲ.①计算技术－高等学校－教材②程序设计－高等学校－教材 Ⅳ.O121②TP311.1

中国版本图书馆 CIP 数据核字（2004）第 086430 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
责任编辑：郑 丹 版式设计：冉晓华 责任校对：王 欣
封面设计：鞠 杨 责任印制：李 妍
北京机工印刷厂印刷·新华书店北京发行所发行
2004 年 10 月第 1 版第 1 次印刷
1000mm×1400mm B5·5.75 印张·219 千字
定价：15.00 元

前　言

为适应科学技术进步的要求，培养高素质人才，必须改革工科数学课程体系与教学方法．为此，我们进行了十多年的教学改革实践．先后在哈尔滨工业大学、黑龙江省教委立项，长期从事“高等工科数学教学过程的优化设计”课题研究．该课题曾获哈尔滨工业大学优秀教学研究成果奖，本套系列课程教材正是这一研究成果的最新总结，包括：《工科数学分析教程》（上，下）、《空间解析几何与线性代数》、《概率论与数理统计》、《复变函数论与运算微积》、《数学物理方程》、《最优化方法》、《计算技术与程序设计》等．

这套教材在编写上广泛吸取国内外知名大学的教学经验，特别是吸取了莫斯科理工学院、乌克兰人民科技大学（原基辅工业大学）等的教学改革经验，提高了知识起点，适当地扩大了知识信息量，加强了基础，并突出了对学生的数学素质与学习能力的培养．具体地，①加强对传统内容的理论叙述；②适当运用近代数学观点来叙述古典工科数学内容，加强了对重要的数学思想方法的阐述；③加强了系列课程内容之间的相互渗透与交叉，注重培养学生综合运用数学知识解决实际问题的能力；④把精选教材内容与编写典型计算题有机结合起来，从而加强了知识间的联系，形成课程的逻辑结构，扩展了知识的深广度，使内容具有较高的系统性和逻辑性；⑤强化对学生的科学工程计算能力的培养；⑥加强了对学生数学建模能力的培养；⑦突出工科特点，增加了许多现代工程应用数学方法；⑧注意到课程内容与工科研究生数学的衔接与区别．

本套教材由孙振绮任总主编．

随着科学技术的不断进步以及电子计算机的迅速发展，计算技术在科学研究与工程设计中发挥着越来越大的作用，科学计算已经成为与理论分析和科学实验并列的第三种科学研究手段．因此，计算技术与程序设计已经成为科研工作者和工程技术人员必须掌握的基本技能．《计算技术与程序设计》就是在校内讲义的基础上经过多次修改完善而成的．

本书除了介绍常用的数值算法及理论外，还特别注重算法的实现．为此，专门用一章的篇幅介绍了程序设计方面的知识，书末还附有数值算例的 C 程序，以便读者在实际编写程序进行计算时参考．本书可作为高等学校工科各专业计算方法课程的教材，也可供工程技术人员及其他科技人员参考．讲授全书大约需要 50 学时．

本书由金承日、孙振绮任主编，丁效华任副主编．参加本书编写的还有李

宝家、邹巾英、孙建邵．崔明根教授仔细审阅了全书，并提出了许多宝贵的意见和建议．

本书在编写与试用过程中，得到了学校有关领导和专家的大力支持，在此深表谢意．

虽然编者认真撰写仔细校对，但由于水平有限，不妥以及错漏之处在所难免，恳请广大读者批评指正．

编　者

目　录

第 1 章　误差与算法

1.1　误差知识

1.1.1　误差的来源

在用数值计算方法解决科学技术中的具体问题时，经常会遇到各种各样的误差，这些误差的来源主要有以下四个途径.

1. 模型误差

将一个实际问题转化成数学模型时，往往抓主要因素，忽略次要因素. 例如，用抛物线来描述抛射体的运动轨迹时忽略了空气阻力. 这种在一定条件下理想化了的模型，与实际问题之间总存在误差，这种误差称为模型误差或描述误差.

2. 观测误差

在各种计算公式中，经常含有一些参量，如光速、声速、重力加速度或其他一些参量，这些参量往往都是由观测或实验得到的，和实际值之间有误差，这种误差称为观测误差.

3. 舍入误差

由于计算机的字长有限，所以在计算过程中对超过计算机字长的数字要进行舍入. 例如用 2.71828 代替 e，用 1.414 代替$\sqrt{2}$，用 2.518736 代替 2.518736204187931081 等，这样产生的误差叫做舍入误差.

4. 截断误差

在科学计算中经常遇到超越运算，它们要求用极限或无穷过程来求得，而实际计算只能用有限次运算来求得其近似值. 例如，取

$$\mathrm{e} \approx 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{100!}$$

这种由有限过程逼近无限过程所产生的误差称为截断误差或方法误差.

在误差的来源当中，前两种误差是计算工作者不能独立解决的，因此下面着重讨论后两种误差.

1.1.2　绝对误差与相对误差

1. 绝对误差

定义 1.1　设 $\tilde{x}$ 代表准确值 x 的一个近似值，则称

$$e = x - \tilde{x} \tag{1.1}$$

为近似值 $\tilde{x}$ 的绝对误差，简称误差. 误差的值可正可负，所以不要将绝对误差认为是误差的绝对值.

一般地，准确值 x 是未知的，所以绝对误差 e 也无法得到，只能根据具体测量或计算的情况估计出 e 的范围，即估计 $|e|$ 的上界.

定义 1.2 如果有已知的常数 $\varepsilon>0$，使得

$$|e| = |x - \tilde{x}| \leqslant \varepsilon \tag{1.2}$$

则称 ε 为近似值 $\tilde{x}$ 的绝对误差限，简称误差限.

显然，一个近似值 $\tilde{x}$ 的误差限有无穷多个. 在实际应用中，误差限应取得尽量小，而且要简洁明了.

此外，有时还用

$$x = \tilde{x} \pm \varepsilon \tag{1.3}$$

表示近似值 $\tilde{x}$ 的误差限是 ε，此时准确值 x 的取值范围是 $[\tilde{x}-\varepsilon, \tilde{x}+\varepsilon]$.

绝对误差是有量纲的量，它反映误差的大小情况.

2. 相对误差

上述绝对误差的概念还不能完全反映近似值的准确程度. 例如，设

$$x = 10 \pm 1$$

$$y = 1000 \pm 3$$

则近似值 $\tilde{x}=10$ 的误差限 $\varepsilon(x)=1$ 虽然比近似值 $\tilde{y}=1000$ 的误差限 $\varepsilon(y)=3$ 小，但考虑到近似值本身的大小，还是觉得 $\tilde{y}=1000$ 更准确. 这就说明，在考虑一个近似值的准确度时，不仅要看其误差值的大小，还要看其近似值本身的大小. 为此，引入相对误差的概念.

定义 1.3 设 $\tilde{x}$ 代表准确值 x 的一个近似值，则称

$$e_r = \frac{e}{\tilde{x}} = \frac{x - \tilde{x}}{\tilde{x}} \tag{1.4}$$

为近似值 $\tilde{x}$ 的相对误差.

相对误差是无量纲的量，它说明了 $\tilde{x}$ 的误差 e 与 $\tilde{x}$ 本身比较起来所占的比例(常用百分比来表示)，因此可以反映近似值的准确程度.

与绝对误差一样，通常我们不能求出相对误差的大小，只能估计它的取值范围.

定义 1.4 如果有已知的常数 $\varepsilon_r>0$，使得

$$|e_r| = \left|\frac{e}{\tilde{x}}\right| \leqslant \varepsilon_r \tag{1.5}$$

则称 ε_r 为近似值 $\tilde{x}$ 的相对误差限.

因为

$$\left|\frac{e}{\tilde{x}}\right| \leqslant \frac{\varepsilon}{|\tilde{x}|}$$

所以相对误差限可以取

$$\varepsilon_r = \frac{\varepsilon}{|\tilde{x}|} \tag{1.6}$$

由此定义，上例中 $\tilde{x}$ 与 $\tilde{y}$ 的相对误差限分别为

$$\varepsilon_r(\tilde{x}) = \frac{1}{10} = 10\%$$

$$\varepsilon_r(\tilde{y}) = \frac{3}{1000} = 0.3\%$$

可见，$\tilde{y}$ 比 $\tilde{x}$ 更精确.

1.1.3 有效数字

在表示一个近似数时，为了同时反映它的准确程度，经常用到“有效数字”的概念.

定义 1.5 设准确值 x 的某一近似值为

$$\tilde{x} = \pm (0.a_1a_2\cdots a_n\cdots) \times 10^m$$

其中，a_1，a_2，$\cdots a_n$，$\cdots$都是 0，1，2，$\cdots$，9 这十个数字之一，且 $a_1 \neq 0$，n 是正整数，m 是整数. 若 $\tilde{x}$ 的绝对误差满足

$$0.05 \times 10^{m-n} = 0.5 \times 10^{m-n-1} < |x - \tilde{x}| \leqslant 0.5 \times 10^{m-n} \tag{1.7}$$

则称 $\tilde{x}$ 作为 x 的近似值具有 n 位有效数字，其中 a_1，a_2，$\cdots$，a_n 均称为 $\tilde{x}$ 的有效数字. 此时，也称 $\tilde{x}$ 是具有 n 位有效数字的近似值，或称 $\tilde{x}$ 准确到第 n 位.

由以上定义可知，如果近似值

$$\tilde{x} = \pm (0.a_1a_2\cdots a_n) \times 10^m \qquad (a_1 \neq 0) \tag{1.8}$$

是由精确值 x 经过四舍五入得到的，则 a_1，a_2，$\cdots$，a_n 均为有效数字. 另外，由定义 1.5 还可以知道，若 x 的某一近似值 $\tilde{x}$ 的绝对误差限是 k 位的半个单位，则 $\tilde{x}$ 中从左往右第一个非零数字起直到 k 位为止的所有数字均为有效数字.

例 1.1 设 $\tilde{x}_1 = 3.14$，$\tilde{x}_2 = 3.141$，$\tilde{x}_3 = 3.142$ 均为 $x = 3.1415926\cdots$ 的近似值，试问 $\tilde{x}_1$，$\tilde{x}_2$，$\tilde{x}_3$ 分别有几位有效数字.

解 因为

$$\tilde{x}_1 = 0.314 \times 10^1 \qquad (m = 1)$$

$$\tilde{x}_2 = 0.3141 \times 10^1 \qquad (m = 1)$$

$$\tilde{x}_3 = 0.3142 \times 10^1 \qquad (m = 1)$$

$$0.05 \times 10^{1-3} < |x - \tilde{x}_1| = 0.00159\cdots \leqslant 0.5 \times 10^{1-3} \qquad (n = 3)$$

$$0.05 \times 10^{1-3} < |x - \tilde{x}_2| = 0.00059\cdots \leqslant 0.5 \times 10^{1-3} \qquad (n = 3)$$

$$0.05\times10^{1-4}<|x-\tilde{x}_3|=0.000407\cdots\leqslant0.5\times10^{1-4}\qquad(n=4)$$

所以 $\tilde{x}_1$ 和 $\tilde{x}_2$ 具有三位有效数字，而 $\tilde{x}_3$ 具有四位有效数字.

例 1.2 设 $\tilde{x}_1=36.9$，$\tilde{x}_2=36.900$，$\tilde{x}_3=36.9000000$ 均为 $x=36.8999978$ 的近似值，则它们分别有三位、五位和七位有效数字. 这是因为 $\tilde{x}_1$ 和 $\tilde{x}_2$ 均可由 x 经四舍五入得到. 而

$$|x-\tilde{x}_3|=0.0000022=0.22\times10^{-5}$$

不超过小数点后第五位的半个单位 0.5×10^{-5}，所以 $\tilde{x}_3$ 准确到小数点后第五位，加上个位和十位，共有七位有效数字. 值得注意的是，在 $\tilde{x}_3$ 小数点后六个零中前四个是有效数字，而后两个则不是有效数字.

在本书中如不特别说明，则所出现的所有近似值均准确到最末位数字. 但要注意，近似值 2100 与 2.1×10^3 的精确度是不同的，前者具有四位有效数字，而后者只有两位有效数字.

1.1.4 误差估计

设 $\tilde{x}$ 和 $\tilde{y}$ 分别为精确值 x 和 y 的近似值，而且误差限分别为 $\varepsilon(\tilde{x})$ 和 $\varepsilon(\tilde{y})$，即

$$|e(\tilde{x})|=|x-\tilde{x}|\leqslant\varepsilon(\tilde{x})$$
$$|e(\tilde{y})|=|y-\tilde{y}|\leqslant\varepsilon(\tilde{y})$$

1. 和、差的误差估计

因为

$$e(\tilde{x}\pm\tilde{y})=(x\pm y)-(\tilde{x}\pm\tilde{y})=(x-\tilde{x})\pm(y-\tilde{y})=e(\tilde{x})\pm e(\tilde{y})$$
$$|e(\tilde{x}\pm\tilde{y})|\leqslant|e(\tilde{x})|+|e(\tilde{y})|\leqslant\varepsilon(\tilde{x})+\varepsilon(\tilde{y})\qquad(1.9)$$

所以

$$\varepsilon(\tilde{x}\pm\tilde{y})=\varepsilon(\tilde{x})+\varepsilon(\tilde{y})\qquad(1.10)$$

即和、差的绝对误差限等于各近似值的绝对误差限之和.

注意：$\tilde{x}-\tilde{y}$ 的相对误差

$$\left|\frac{e(\tilde{x}-\tilde{y})}{\tilde{x}-\tilde{y}}\right|\leqslant\frac{\varepsilon(\tilde{x})+\varepsilon(\tilde{y})}{|\tilde{x}-\tilde{y}|}=\left|\frac{\varepsilon(\tilde{x})}{\tilde{x}}\right|\left|\frac{\tilde{x}}{\tilde{x}-\tilde{y}}\right|+\left|\frac{\varepsilon(\tilde{y})}{\tilde{y}}\right|\left|\frac{\tilde{y}}{\tilde{x}-\tilde{y}}\right|$$

当 $\tilde{x}$ 与 $\tilde{y}$ 比较接近时，$\left|\frac{\tilde{x}}{\tilde{x}-\tilde{y}}\right|$ 或 $\left|\frac{\tilde{y}}{\tilde{x}-\tilde{y}}\right|$ 可能很大，这时 $\tilde{x}-\tilde{y}$ 的相对误差也可能很大. 由此可见，两个相近的近似值相减，可能会造成有效数字的严重损失. 例如，若用四位有效数字计算，得

$$\sqrt{1001}-\sqrt{1000}\approx31.64-31.62=0.02$$

而精确值为 $\sqrt{1001}-\sqrt{1000}=0.0158074\cdots$. 可见，近似值 0.02 只有一位有效数字. 为了提高运算精度，可采用如下方法（不用减法！）计算：

$$\sqrt{1001}-\sqrt{1000}=\frac{1}{\sqrt{1001}+\sqrt{1000}}\approx\frac{1}{31.64+31.62}\approx 0.01581$$

这个结果具有四位有效数字.

2. 乘积的误差估计

$$\begin{aligned} e(\tilde{x}\,\tilde{y}) &= xy-\tilde{x}\,\tilde{y}=xy-\tilde{x}\,y+\tilde{x}\,y-\tilde{x}\,\tilde{y}\\ &= ye(\tilde{x})+\tilde{x}\,e(\tilde{y})\\ &= [\tilde{y}+e(\tilde{y})]e(\tilde{x})+\tilde{x}\,e(\tilde{y})\\ &= \tilde{y}\,e(\tilde{x})+\tilde{x}\,e(\tilde{y})+e(\tilde{x})e(\tilde{y}) \end{aligned}$$

$$|e(\tilde{x}\,\tilde{y})|\leqslant|\tilde{y}|\varepsilon(\tilde{x})+|\tilde{x}|\varepsilon(\tilde{y})+\varepsilon(\tilde{x})\varepsilon(\tilde{y}) \tag{1.11}$$

故乘积 $\tilde{x}\tilde{y}$ 的绝对误差限为

$$\varepsilon(\tilde{x}\,\tilde{y})=|\tilde{y}|\varepsilon(\tilde{x})+|\tilde{x}|\varepsilon(\tilde{y})+\varepsilon(\tilde{x})\varepsilon(\tilde{y}) \tag{1.12}$$

3. 商的误差估计

$$\begin{aligned} e\left(\frac{\tilde{y}}{\tilde{x}}\right) &= \frac{y}{x}-\frac{\tilde{y}}{\tilde{x}}=\frac{y\tilde{x}-x\tilde{y}}{x\tilde{x}}=\frac{y\tilde{x}-\tilde{x}\,\tilde{y}+\tilde{x}\,\tilde{y}-x\tilde{y}}{x\tilde{x}}\\ &= \frac{\tilde{x}\,e(\tilde{y})-\tilde{y}\,e(\tilde{x})}{[\tilde{x}+e(\tilde{x})]\tilde{x}}=\frac{\tilde{x}\,e(\tilde{y})-\tilde{y}\,e(\tilde{x})}{\tilde{x}^2}\,\frac{\tilde{x}}{\tilde{x}+e(\tilde{x})}\\ &= \frac{\tilde{x}\,e(\tilde{y})-\tilde{y}\,e(\tilde{x})}{\tilde{x}^2}\,\frac{1}{1+e_{\mathrm{r}}(\tilde{x})}\\ &= \frac{\tilde{y}}{\tilde{x}}\,\frac{e_{\mathrm{r}}(\tilde{y})-e_{\mathrm{r}}(\tilde{x})}{1+e_{\mathrm{r}}(\tilde{x})} \end{aligned}$$

可见，当 $|\tilde{y}/\tilde{x}|$ 很大时，$e\left(\frac{\tilde{y}}{\tilde{x}}\right)$ 也可能很大，因此在计算过程中应尽量避免大数与小数（按绝对值）之间的除法运算.

4. 一般运算的误差估计

设 $\tilde{x}_i$ 是精确值 x_i 的近似值，$u=f(x_1,x_2,\cdots,x_n)$，则由多元函数的 Taylor 展开式得误差的近似估计式

$$\begin{aligned} e(\tilde{u}) &= f(x_1,x_2,\cdots,x_n)-f(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n)\\ &\approx \sum_{i=1}^{n} f'_{x_i}(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n)\cdot e(\tilde{x}_i) \end{aligned} \tag{1.13}$$

其中，f'_{x_i} 是多元函数 f 关于变量 x_i 的一阶偏导数. 由式（1.13）可得 $\tilde{u}$ 的误差限

$$\varepsilon(\tilde{u})=\sum_{i=1}^{n}|f'_{x_i}(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n)|\cdot\varepsilon(\tilde{x}_i) \tag{1.14}$$

例 1.3 测得一个长方形的长度为 $\tilde{x}=120$ cm，宽度为 $\tilde{y}=60$ cm，已知其误差限分别为 $\varepsilon(\tilde{x})=0.2$ cm 和 $\varepsilon(\tilde{y})=0.1$ cm. 求此长方形的面积 A，并给出

相对误差限.

解 面积的近似值

$$\bar{A} = \tilde{x}\,\tilde{y} = 120 \times 60\ \text{cm}^2 = 7200\ \text{cm}^2$$

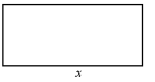

图 1.1

由式 (1.12) 求得其绝对误差限

$$\varepsilon(\bar{A}) = (60 \times 0.2 + 120 \times 0.1 + 0.2 \times 0.1)\ \text{cm}^2 = 24.02\ \text{cm}^2$$

故相对误差限

$$\varepsilon_r(\bar{A}) = \frac{\varepsilon(\bar{A})}{|\bar{A}|} = \frac{24.02}{7200} \approx 0.3336\%$$

如果用式 (1.14) 计算，则

$$\varepsilon(\bar{A}) = (60 \times 0.2 + 120 \times 0.1)\ \text{cm}^2 = 24\ \text{cm}^2$$

$$\varepsilon_r(\bar{A}) = \frac{24}{7200} \approx 0.3333\%$$

可见，利用近似估计式 (1.14) 仍然可以获得很精确的估计.

习 题 1.1

1. 下面各近似值的绝对误差限是最末位的半个单位，试指出它们各有几位有效数字.

$\tilde{x}_1 = 0.002$，　$\tilde{x}_2 = 25.30$，　$\tilde{x}_3 = 4 \times 10^{-3}$，

$\tilde{x}_4 = 8 \times 10^4$，　$\tilde{x}_5 = 2.010$，　$\tilde{x}_6 = 0.001 \times 10^6$

2. 设近似值 $\tilde{x}_1 = 126.385$，$\tilde{x}_2 = -16.20$，$\tilde{x}_3 = 5.1$ 均准确到末位数字，试分别求 $\tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3$，$\tilde{x}_1\tilde{x}_2$，$\dfrac{\tilde{x}_1}{\tilde{x}_3}$ 的相对误差限.

3. 设 $\tilde{x}_1 = 58.24$，$\tilde{x}_2 = 58.25$，$\tilde{x}_3 = 58.100$，$\tilde{x}_4 = 58.200$，$\tilde{x}_5 = 58.20000000$，$\tilde{x}_6 = 0.5820 \times 10^2$ 均为精确值 $x = 58.1999527$ 的近似值，试问它们各有几位有效数字.

4. 设近似值 $\tilde{x} = 2.63847$ 具有四位有效数字，试问 $\tilde{x}$ 的相对误差限是多少.

5. 设近似值 $\tilde{x} = 1.24867$ 的相对误差限是 0.4%，试问 $\tilde{x}$ 具有几位有效数字.

6. 计算球的体积使其相对误差限为 1%，试问测量半径 R 时 (假设 π 取准确值) 允许的相对误差限是多少.

7. 设近似值 $\tilde{x} = \pm(0.a_1a_2\cdots a_n\cdots) \times 10^m$ 的相对误差限小于 $\dfrac{1}{2(a_1+1)} \times 10^{1-n}$ (这里 $a_1 \neq 0$)，证明 $\tilde{x}$ 至少有 n 位有效数字.

8. 测得圆柱体半径 $R \approx 13.5\text{cm}$，高 $h \approx 27.0\text{cm}$，取圆周率 $\pi \approx 3.14$ 计算此圆柱体体积，其绝对误差限是多少？

1.2 算法的概念

“算法”是一组 (有限个) 规则，它提供了解决某一问题的运算序列. 通俗地说，

“算法”是对解题方案完整而准确的描述.

算法具有多种描述形式,它既可以用计算框图来描述,也可以用计算机程序语言来描述,本书中我们更多的是用数学语言 (即一系列数学符号和数学公式) 来描述算法.

一个算法必须具备以下四个特征:

⑴ 有穷性,即一个算法必须经过有限步就得结束.因为计算机的运算速度有限,所以在实际应用中往往要求算法的计算过程“非常”有穷.如果一个算法虽然具备有穷性,但需要花费几十年甚至上千年的时间才能算出结果,那么这种算法就失去了使用价值.

⑵ 一义性,即算法的每一步都必须有明确的定义,准确地告诉人们应该做什么.为了做到算法的确定性,必须使用精确的语言来描述算法,这种语言不应该有二义性.

⑶ 有效性,即算法中的每一种运算都是可以做得到的.

⑷ 每一个算法都要有输出.

具有上述四个特征的叫做算法,具有上述后三个特征 (即不要求具备有穷性) 的叫做计算方法.例如,利用公式

$$e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots$$

求出 e 值的方法是计算方法,而不是算法.但是,若要求出具有 5 位有效数字的近似值 $\tilde{e}$,则它满足有穷性,因而这种方法是算法.

不同的问题有不同的算法,但设计算法的思路可以分成以下几类,统称为基础算法.

1.2.1 枚举法

所谓枚举法就是一一列举判别法.为了判别一个命题是否为真,可以对所有可能情况进行一一列举判断,只要没有遇到反例,就可以断定本命题为真,这就是枚举证明.

枚举法是比较笨拙、原始的方法,它的弱点是运算量大.因此,枚举法常常用于局部问题,比如,在图中寻找路径、查找、搜索等.

1.2.2 归纳法

所谓归纳法,就是由特殊关系归纳出一般规律的方法.通过仔细观察找出共同的特点,并用数学语言或其他方法描述出来.但是,归纳出一般规律并不容易,也没有一定的规则可遵循.

1. 数学归纳法

通过精心观察提出的归纳假设,需要用数学归纳法证明其正确性.数学归纳法陈述如下:

设 $P(n)$ 是关于自然数的命题，有

(1) 对于 $n=n_0$，证明 $P(n_0)=$‘真’　　(归纳基础)

(2) 如果由假设 $n=k$ $(k\geqslant n_0)$ 时，$P(k)=$‘真’，可以推出 $P(k+1)=$‘真’　　(归纳步骤)

(3) 则对任何自然数 $n\geqslant n_0$，都有 $P(n)=$‘真’　　(结论)

数学归纳法在程序设计、程序证明中极其有用. 以自然数为自变量的函数称为数函数，其定义域是自然数集，其值域是求解问题的值域.

将数学归纳法应用于数函数特别有效，我们就是在数函数的概念上进行归纳.

2. 递推

利用计算机解题往往采用逐步求解的方法. 如果对某一种运算找到了前一步与后一步之间的关系，只要起始条件清楚，问题就好解决，让计算机一步一步地计算就是了. 计算机不怕重复计算，程序设计经常追求重复运算 (迭代)，这样才能真正发挥计算机的效益. 递推在数值计算中是极为常见的.

许多函数都可以在数函数的概念上建立递推关系，如

阶乘函数
$$\begin{cases} f(0)=0!=1 & n=0 \\ f(n)=nf(n-1) & n\geqslant 1 \end{cases}$$

幂函数
$$\begin{cases} f(0)=x^0=1 & n=0 \\ f(n)=xf(n-1) & n\geqslant 1 \end{cases}$$

如果要求解，只要沿 $n=0,1,2,\cdots$ 逐一计算即可.

这些函数更一般的情况是

$$f(n,k)=\begin{cases} h(k) & n=0 \\ g(n,k,f(n-1)) & n\geqslant 1 \end{cases}$$

其中，k 是参数. 此时称 f 是以 h 为基始函数、以 g 为递推函数的函数 (g 是已知函数).

3. 递归

如果一个未知函数 f 用其自身构成的已知函数 g 来定义

$$\begin{cases} f(0)=a & n=0 \\ f(n)=g(n,f(n-1)) & n\geqslant 1 \end{cases}$$

则称 f 为递归定义函数，简称递归函数，称已知数 a 为递归边界.

递归定义和递推定义有时看上去是一样的，但其实现方式大不相同. 递推一定从基始函数出发，而递归则从函数本身到达递归边界. 例如，设函数 $f(n)$ 是由

$$\begin{cases} f(0) = 2 \\ f(n) = n + f(n-1) \quad n \geqslant 1 \end{cases}$$

递归定义的，则

$$\begin{aligned} f(n) &= n + f(n-1) = n + (n-1) + f(n-2) \\ &= \cdots \\ &= n + (n-1) + (n-2) + \cdots + 2 + 1 + f(0) \\ &= \frac{n(n+1)}{2} + 2 \end{aligned}$$

1.2.3 回溯法

对于某些问题，我们很难归纳出一般规律，又不能漫无边际地枚举. 此时，可以根据以往的经验和一些已知的线索进行试探. 如果不成功，那就返回到问题的出发点，换一种办法重新去试. 如此反复，直到所有的办法都试过了，问题也许就能解决. 这就是所谓的回溯法，它的本质是枚举和试探.

习 题 1.2

1. 什么是算法？
2. 算法应具备哪些特征？
3. 什么是数函数？
4. 基础算法包括哪些方法？
5. 设 $n = 2^k - 1$，$A(1) = 1$，且对 $k > 1$ 恒有 $A(k) = 2A(k-1) + 2$，证明：

$$A(k) = (3n - 1)/2$$

1.3 算法分析

1.3.1 算法分析的标准

一个有用的算法首先应该是正确的算法. 所以，在分析一个算法的优劣时，首先要考虑算法的正确性. 算法的正确性只能通过公式推导、逻辑推理等数学方法予以证明. 其次，要考虑算法的可读性、稳定性、收敛性、存储量、效率等因素.

需要强调的是，算法分析是针对算法本身而言的，并不是分析算法的某种实现，算法和算法的实现是有区别的. 算法的实现是指针对某种算法，选择一定的数据结构和程序结构，把算法用计算机语言表达出来，并在具体的计算机上运行. 算法实现的任务是把一个算法转换成计算机程序. 由于不同的算法实现会产生不同的效率、存储量、可读性等，所以算法分析的标准应该独立于算法的实现.

在评价一种正确的算法时，人们常把算法的效率放在首位. 但是，随着结构程序思想的发展，人们越来越重视分析哪种算法更简单、清晰、易于编制程序和调试等.

1.3.2 算法的效率

分析算法的效率时，通常只需分析该算法中起决定作用的运算.

一个算法的效率包括占用内存空间（空间复杂性）和计算工作量（时间复杂性）两个方面，我们希望算法少占内存空间且计算量也尽量小.

空间复杂性包括算法程序所占的空间、输入的初始数据所占的空间、算法执行过程中所需要的额外空间（包括算法程序执行过程中的工作单元及某种数据结构所需要的附加存储空间）等. 由于空间复杂性问题一般可以转化到时间复杂性中讨论，所以下面重点讨论时间复杂性问题.

所谓一个算法的时间复杂性，就是执行这个算法的计算工作量（即算法的时间代价）. 对于同一个问题可以有多种不同的算法，它们的时间复杂性也可能不一样. 我们希望选择一种花费计算时间最少的算法，这就遇到了如何度量时间复杂性的问题.

关于时间复杂性的度量标准，最自然、最简单的有以下两种：第一种是用算法的执行时间的长短来衡量，第二种是用算法程序中所含指令数目或语句数目来衡量. 但是，这两种度量标准都是不可取的. 一个算法的执行时间与所使用的计算机有关，而我们不可能针对每一种具体的计算机来讨论算法，所以第一种衡量标准是不可取的. 第二种衡量方法又依赖于所使用的程序语言及程序设计者的水平和风格，因此也是不可取的.

我们希望找到度量工作量的一般标准，它既不依赖于所使用的计算机、程序语言以及编程人员，又不依赖于算法实现过程中的许多细节. 这里所说的细节包括循环下标的计算，数组下标的计算，设置数据结构指针等“簿记”运算以及其他运算. 这就是说，所选择的工作量度量方法不仅是足够精确的，而且还应该具有一般性. 为此，在分析一个算法的计算工作量时，可以从以下几个方面入手.

1. 基本运算

计算机里可实施的运算有四则运算和逻辑运算. 一次乘法比一次加法慢很多，所以不分析什么运算只说运算次数是不可能比较出算法的优劣性的. 因此，在分析一个算法的计算工作量时，要抓住主要工作量，忽略次要工作量. 也就是说，在所要解决的问题中合理地选择基本运算作为度量标准，忽略“簿记”等细节问题. 这样，算法的工作量就可以用基本运算次数来度量. 基本运算是一个算法中运算量最大的一种运算，是该算法运算的主要特征. 例如：

问　　题	基本运算
在一维数组中找出 x	x 与数组元素相比较
实矩阵相乘	实数乘法
排　　序	表中元素相比较

一般来说，每一种算法除了基本运算之外，还有其他运算. 例如，实矩阵

的乘法中除了乘法运算外，还有大量的加法运算，虽然加法运算的计算量与乘法运算相比并不占主导地位，但仍然有一定的工作量. 如果一个算法中乘法运算很少，而加法运算很多，则采用乘法次数和加法次数来度量工作量显然更合理. 另外，由于乘法比加法花费更多的计算时间，所以往往把一个算法改进为少做乘法多做加法的算法，这就给算法的工作量带来一些不确定的因素. 在极端的情况下，对于解决同一问题的所有算法中，可以按照基本运算类型的不同，将这些算法也分成不同的算法类，以便在同一算法类中比较工作量的大小. 但在不同算法类中的两个算法之间很难精确地比较工作量的大小.

总之，当一个算法的计算量主要集中在基本运算时，就以基本运算量来度量该算法的工作量. 当一个算法中一些非基本运算所占的工作量较大时，将这一部分工作量也要考虑到算法的工作量中去. 通常采用加权因子法来处理，也就是将这一部分工作量转化成基本运算量乘上一定的比例系数来度量.

2. 输入尺寸

确定了基本运算，讨论算法的工作量就有了前提. 但为了进一步表示分析的结果，还需要一种衡量一个问题输入尺寸的度量. 这是因为，算法所执行的基本运算次数与输入尺寸有密切的关系. 例如，两个三阶矩阵相乘与两个二十阶矩阵相乘，其基本运算 (实数乘法) 的次数相差很大. 可见，算法的工作量不仅取决于基本运算，还取决于输入尺寸 (即问题的规模). 因此，在分析算法的工作量时首先要确定问题的输入尺寸. 例如：

问　　题	输入尺寸
在一维数组中找出 x	数组元素的个数 n
实数矩阵相乘	矩阵的阶 $m\times l\times n$
排　　序	表元素的个数 n

3. 输入情况

确定了基本运算与输入尺寸，就可以分析算法的工作量，即算法在执行过程中其基本运算的执行次数. 但是，对于一个固定的输入尺寸 n，一个算法的工作量可能与特定的输入有关. 例如，对于“在一维数组中找出等于 x 的数”这个问题，如果输入的一组数据中，第一个数就等于 x 和第 n 个数才等于 x，其查找次数之比为 $1:n$. 可见，对于同一个算法，可以从不同的输入情况分析它的工作量. 但这样讨论，情况过于复杂. 所以，通常从平均情况和最坏情况分析算法的工作量.

当输入尺寸为 n 时，算法执行的运算次数取决于某一特定的输入. 每一种算法都可以有不同的输入. 但从概率统计的观点看，它有一个平均的输入情况. 在平均情况下分析算法的工作量 $A(n)$ 是比较准确的. 但在平均情况下，分析算法的工作量比较麻烦，往往难以估计. 因此，经常在最坏情况下分析算法的工作量.

所谓最坏情况分析，是指在输入尺寸为 n 时，算法所执行的基本运算的最大次数 $W(n)$. 这种分析方法给出的是基本运算的上界，所以是保守的估计. 不过，由于在最坏情况下容易进行分析，而且还可以给出算法的运行时间范围，所以最坏情况分析经常被采用.

需要注意的是，一个算法什么时候发生最坏情况，并不取决于问题本身，而是取决于所选取的算法. 例如，在数组 M 中找出 x 这个问题中，若采用从后向前搜索的倒序算法，则最坏情况就发生在 x 位于数组的首位或 x 不在数组中的时候. 也有一些问题，其算法的工作量与输入情况无关，即当输入尺寸为 n 时，有

$$A(n) = W(n) = \text{在所有情况下基本运算次数}$$

下面的例子就属于这种情况.

例 1.4 设 $\boldsymbol{a} = \{a_1, a_2, \cdots, a_n\}$ 和 $\boldsymbol{b} = \{b_1, b_2, \cdots, b_n\}$ 是两个 n 维实向量，计算数量积 $\boldsymbol{a} \cdot \boldsymbol{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$.

解 其算法如图 1.2 所示.

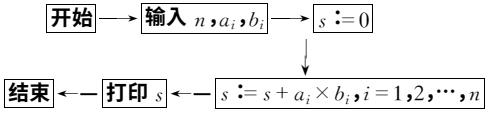

图 1.2

在上述算法中，取实数的乘法作为基本运算，则对输入尺寸 n，不管输入的 a_i 和 b_i 如何，都要做 n 次乘法，故

$$A(n) = W(n) = n$$

4. 工作量的阶

在算法分析中，可能会遇到这种情况，对于同一个问题，有两种算法可供选择，它们在输入尺寸为 n 时的计算量分别为 $W_1(n) = n^2 - n$ 和 $W_2(n) = 10n$，那么哪一个算法的效率更高呢？当 $n < 11$ 时，$W_1(n) < W_2(n)$；当 $n > 11$ 时，$W_1(n) > W_2(n)$.

一般地，假设同一问题的两种不同算法的计算量分别为 $W_1(n)$ 和 $W_2(n)$，如果存在正的实数 c 和 m，使得

$$W_1(n) \leqslant cW_2(n) \qquad (\text{对 } n > m)$$

成立，则称 W_1 是比 W_2 低阶或同阶的. 又如果 W_2 也是比 W_1 低阶或同阶的，则称 W_1 和 W_2 是同阶的，记为 $W_1 = O(W_2)$.

例如，

$$2n^2 = O\left(\frac{n^2}{10}\right)$$

$$3n + 4 = O(n)$$