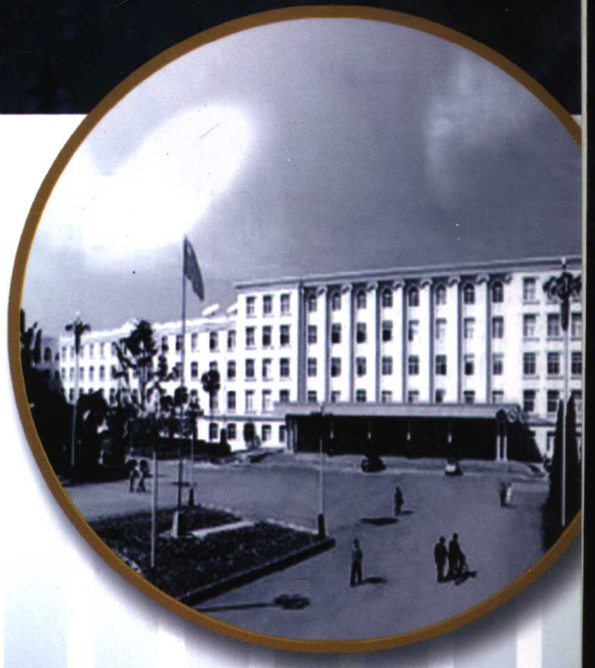
重点大学计算机教材

HZ BOOKS

计算复杂性

顾小丰　孙世新　卢光辉　编著

电　子　科　技　大　学



机械工业出版社
China Machine Press

重点大学计算机教材

计算复杂性

顾小丰　孙世新　卢光辉　编著
电　子　科　技　大　学

机械工业出版社
China Machine Press

本书全面、系统地介绍了计算复杂性理论的基本内容和基本方法。内容涉及数值计算的复杂性，主要包括Kuhn算法设计、正确性证明和复杂性分析；算法复杂性和计算模型；贪心法、动态规划、回溯法和分枝限界法等问题的算法设计方法以及P类、NP类和NPC类问题及其证明方法、若干NPC问题的近似算法。

本书可作为计算机专业及数学专业的本科生或研究生的教材，也可供从事数学和计算机科学的教师和研究人员参考。

图书在版编目（CIP）数据

计算复杂性 / 顾小丰等编著. -北京：机械工业出版社，2005.1
（重点大学计算机教材）
ISBN 7-111-15314-6

Ⅰ.计… Ⅱ.顾… Ⅲ.计算复杂性－高等学校－教材 Ⅳ.TP301.5

中国版本图书馆CIP数据核字（2004）第098875号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：杨　敏　　王镇元
北京昌平奔腾印刷厂印刷 · 新华书店北京发行所发行
2005年1月第1版第1次印刷
787mm × 1092mm 1/16 · 10.5印张
印数：0 001-4 000册
定价：19.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

前　言

计算复杂性理论分为两个方面：计算机科学的复杂性理论和数值计算的复杂性理论。

数值计算的复杂性理论是计算机科学蓬勃发展的一个必然结果，它主要是研究如何更好地利用机器进行数值计算的问题。对于一种算法，不仅要考虑它是否有收敛性的保证,而且还必须考虑它的计算成本。如果计算成本随问题规模的增加而增加的速度是指数级的，则这种算法将被认为是在实践中难以接受的，因而希望算法的计算成本随问题规模的增加而增加的速度是多项式的。

计算机科学的复杂性理论，尤其是其中的NP完全问题是计算机科学理论中最重要的问题。自从1971年S.A.Cook提出P类问题是否等价NP类问题以来，许多著名科学家对这个问题进行了深入研究，有的试图证明它们等价，更多的人猜测并试图证明它们不等价。但迄今为止，人们并没有获得完全的成功，并且越来越多的迹象表明，这是一个十分困难的问题。对该问题的研究带动了对许多其他问题的研究，并逐渐发展成为一个庞大的理论系统。这些研究无论是对增进人们对P和NP本质的了解，还是对许多问题复杂性的解决，都有着重大的意义。

本书是在作者多年为电子科技大学计算机专业硕士研究生讲授计算复杂性课程的基础之上经过修改、编写而成的。全书共分九章，前四章主要介绍数值计算的复杂性理论，重点讨论了解代数方程的Kuhn算法。后五章主要介绍计算机科学的复杂性理论，重点是讨论P与NP及NP完全问题。

由于作者水平有限，错误和疏漏在所难免，敬请读者批评指正。

作　者

2004年6月于电子科技大学

作者简介

顾小丰，1966年生，1991年于兰州大学数学系获理学硕士学位。现任电子科技大学计算机学院高级工程师，硕士研究生导师。主要从事网络计算及应用、并行计算研究和教学。参与完成了“八五”、“九五”军事预研项目两项，获2001年度“国防科学技术进步奖”二等奖，撰写《离散数学》和《离散数学及其应用》两本教材。

孙世新，1940年生，1966年毕业于四川大学数学系，现任电子科技大学计算机学院教授，博士研究生导师，享受政府特殊津贴专家，全国并行计算专家委员会委员。

1984年起，分别在法国格勒诺贝尔第一大学和贡比涅大学、意大利罗马大学以及香港科技大学作访问学者和客座研究员。并先后赴美国、加拿大、法国、比利时、德国、瑞典等国进行学术访问。主要从事计算机科学理论与应用的研究与教学工作。其主要研究方向为网络计算技术、并行/分布式计算及其应用、信息压缩技术、数值计算与组合算法等。主持并参与了“九五”军事预研项目、国家高性能计算基金、“863”计划等多项课题研究。自1988年至今，在国内外著名期刊杂志发表论文70余篇，其中30余篇被国际著名的三大检索系统SCI、EI、ISTP以及美国的著名检索杂志“M.R.”等收录评论，出版《组合数学》教材一部。获省科技进步三等奖、“国防科学技术进步”二等奖、校优秀教材奖、优秀论文奖和摩托罗拉教师奖教金和托普基础课教师奖多项。

卢光辉，男，1971生，1996年于四川师大数学系获硕士学位，2002年于电子科技大学计算机学院获工学博士学位，现任电子科技大学计算机学院副教授，硕士研究生导师。主要从事网格计算、并行处理、计算机网络和图像处理等方面的科研与教学工作，在一级学报上发表学术论文多篇，参与完成了“九五”军事预研项目一项，获2001年度“国防科学技术进步奖”二等奖。

目　录

第一部分　数值计算的复杂性

第1章　代数方程和数值计算的复杂性理论简介

1.1　代数方程的不动点迭代算法

我们知道，形如 $ax^2+bx+c=0\ (a\neq 0)$ 的一元二次方程，它的两个根可以按照

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

这个求根公式计算出来。形如 $ay^3+by^2+cy+d=0\ (a\neq 0)$ 的一元三次方程，可作代换 $y=x-\frac{b}{3a}$ 化为无平方项的简化形式 $x^3+px+q=0$，对简化形式作代换 $x=z-\frac{p}{3z}$，可将其化为 $z^6+qz^3-\frac{p^3}{27}=0$，此方程可看作 z^3 的二次方程，不难解出 z，再代回去，可得到

$$x_1=A+B;\ \ x_2=A\omega+B\omega^2;\ \ x_3=A\omega^2+B\omega$$

其中

$$A=\sqrt[3]{-\frac{q}{2}+\sqrt{\frac{1}{4}q^2+\frac{1}{27}p^2}};\ \ B=\sqrt[3]{-\frac{q}{2}-\sqrt{\frac{1}{4}q^2+\frac{1}{27}p^2}}$$

$\omega=\cos 120°+\mathrm{i}\cdot\sin 120°$（$\omega$是1的立方根之一）。

而形如 $x^4+ax^3+bx^2+cx+d=0$ 的一元四次方程，可以配方成

$$\left(x^2+\frac{ax}{2}+\frac{t}{2}\right)^2=\left(\frac{a^2}{4}-b+t\right)x^2+\left(\frac{at}{2}-c\right)x+\left(\frac{t^2}{4}-d\right)$$

其中 t 是待定系数。令 $f(x,t)=x^2+\frac{ax}{2}+\frac{t}{2}$，上式的左边为 $[f(x,t)]^2$。为了使上式右边为 $[g(x,t)]^2$ 要选择适当的 t，使判别式为0，即

$$\left(\frac{at}{2}-c\right)^2-4\left(\frac{a^2}{4}-b+t\right)\left(\frac{t^2}{4}-d\right)=0$$

亦即

$$t^3 - bt^2 - (ac - 4d)\,t - a^2d + 4bd - c^2 = 0$$

先解关于t的三次方程，求出t，进而配方得到$[g(x,t)]^2$，再解两个二次方程$f(x,t) + g(x,t) = 0$和$f(x,t) - g(x,t) = 0$即可得到一元四次方程的根。

上述这种把代数方程的根用方程系数经有限次加、减、乘、除和开方运算表示出来的方法，叫做*代数方程的代数解法（或公式解法）*。

但是，数学家已经证明，五次和更高次方程没有普遍适用的代数解法，也就是说，没有用方程系数经有限次加、减、乘、除和开方运算把方程的根表示出来的公式。这种“无公式解”的本性是和五次以下的方程不同的。由于这个原因，以后我们只把五次和高于五次的代数方程叫做*高次方程*。

高次方程虽然没有普遍适用的代数解法，但是却有一些非代数的或者说非公式的解法。下面介绍高次方程的不动点迭代解法。

代数方程都可以表示成

$$f(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-1}x + a_n = 0,\quad a_0 \neq 0$$

这里$f(x)$是一个n次多项式。如果能够把方程

$$f(x) = 0$$

改写成

$$x = \phi(x)$$

的形式，并能够找到一个x^*，使得

$$x^* = \phi(x^*)$$

那么，x^*就是原代数方程的一个解。

把方程$f(x) = 0$该写成$x = \phi(x)$的形式，非常容易，也有许多方式。例如，可以写成

$$x = -a_0x^n - a_1x^{n-1} - a_2x^{n-2} - \cdots - (a_{n-1} - 1)\,x - a_n$$

也可以写成

$$x = -\frac{a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-1}x + a_n}{a_0x^{n-1}}$$

因为新方程是从$f(x) = 0$变换来的，所以新方程的解也就是原方程的解。

x^*是新方程的解，就是说$x^* = \phi(x^*)$。一个函数就表示一个对应，或者说表示一个变换。函数$\phi(x)$是x把变成$\phi(x)$的对应。现在$x^* = \phi(x^*)$，就是把x^*变成$\phi(x^*) = x^*$自己。换一个说法，就是x^*经过ϕ这个变换没有动，由于这个原因，使得$x^* = \phi(x^*)$的点x^*叫做函数ϕ的*不动点*，形如$x = \phi(x)$的方程也就叫做*不动点方程*。

由上可知，把代数方程改写成不动点方程是容易的，但难的是怎样得到不动点x^*。为此，我们采用迭代方法：找一个点，记作x_0，代入函数ϕ，得到$\phi(x_0)$，记作x_1，再代入函数ϕ，得到$\phi(x_1)$，记作x_2，……，如此循环下去，可以得到一个序列

$$x_0, x_1, x_2, \cdots, x_n, \cdots$$

其迭代关系可以表示为

$$x_{n+1} = \phi(x_n), n = 0, 1, 2, \cdots$$

有趣的是，这个迭代序列有时候可以帮助我们找到所要的不动点，这就是不动点迭代方法。

先试一试，考虑五次方程

$$x^5 - 17x + 2 = 0$$

首先把它变成不动点方程

$$x = \frac{x^5 + 2}{17} \equiv \phi(x)$$

这里的“≡”表示 $\frac{x^5+2}{17}$ 就是$\phi(x)$。选$x_0 = 0$进行迭代，得

$$x_1 = \frac{2}{17} = 0.1176483$$

$$x_2 = \frac{0.1176483^5 + 2}{17} = 0.1176483$$

$$x_3 = \frac{0.1176483^5 + 2}{17} = 0.1176483$$

$x^* = 0.1176483$就是ϕ的一个不动点，所以也是原方程的一个解。

熟悉这方面内容的读者可能已经看出，2也是原方程的一个解。但是如果你不懂迭代法，或者虽然懂但不动手算，就无论如何看不出0.1176483这个解。

刚才，我们选$x_0 = 0$开始迭代，获得成功，这是不是巧合？是不是接受了什么暗示？提出怀疑是完全合理的，应当多做几次试验。下面分别从$x_0 = 1$、$x_0 = -1$、$x_0 = 2$、$x_0 = -2$开始迭代，4个迭代序列如下：

n	x_n	x_n	x_n	x_n
0	2.0	1.0	−1.0	−2.0
1	2.0	0.1764705	0.0588235	−1.7647059
2		0.1176571	0.1176471	−0.8890822
3		0.1176483	0.1176483	0.0849686
4		0.1176483	0.1176483	0.1176473
5				0.1176483
6				0.1176483

到目前为止，5个迭代都是成功的，一共找到2个解。下面再扩大范围试试，从$x_0 = 3$和$x_0 = -3$开始迭代，数据如下：

n:	0	1	2	3
x_n:	3.0	14.411765	36571.122	3.84806×10^{21}
x_n:	−3.0	−14.176471	−33681.402	-2.5497×10^{21}

不必再算下去就可以判断，这两个序列都是没有极限的。迭代公式是

$$x_{n+1} = \phi(x_n) = \frac{x_n^5 + 2}{17}$$

当x_n变大时，x_n^5就会非常大，最后除以17，仍然保持很大。所以，从$x_0 = 3$开始的迭代趋向$+\infty$，从$x_0 = -3$开始的迭代趋向$-\infty$，它们都不收敛。我们说这样的迭代序列是发散的。

不动点迭代方法非常简单，但不一定能够保证成功。迭代是否成功，怎样使迭代成功，我们后面会讨论。为了进一步把上述迭代的情况研究清楚，我们可以画一张迭代图帮助分析。

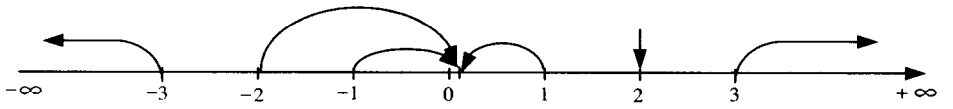


图 1-1

图1-1标明了前面所做的7个迭代过程的结果，1个迭代驻守在2这个点不动，4个迭代收敛到0.1176483，另外两个迭代分别向$+\infty$和$-\infty$发散。这个图启发我们进一步提出问题:

(1) 从−3开始的迭代发散，从−2开始的迭代收敛。−3和−2之间，应当有一个分界点，分界点在哪里？从分界点开始的迭代，究竟怎样进行?

(2) 从1开始的迭代收敛到0.1176483这个不动点，从2开始的迭代驻守在2这个不动点，它们之间也应当有一个分界点，也有同样的问题。

(3) 相应地，2和3之间也有同样的问题。

为此，我们做3个迭代，数据如下:

n	x_n	x_n	x_n
0	−2.1	1.9	2.1
1	−2.2847653	1.5741759	2.5200594
2	−3.5446962	0.6862608	6.0963224
3	−32.801317	0.1266060	495.44315
4	−2233620.6	0.1176489	
5		0.1176483	
6		0.1176483	

看来，点2向右过去一点，迭代就发散到$+\infty$，点2向左过去一点，迭代就收敛到0.1176483;而从−2.1开始的迭代发散到$-\infty$。

更精细的试验可以得出以下数据:

n	x_n	x_n	x_n	x_n
0	−2.0590	−2.0589	1.9999	2.0001
1	−2.052245	−2.0586959	1.9995295	2.0004706
2	−2.0604117	−2.0576176	1.9977868	2.0022158

3	−2.0666974	−2.0519269	1.9896078	2.0104505
4	−2.1002199	−2.0220904	1.9516012	2.0496956
5	−2.2860233	−1.8709824	1.7830008	2.2457759
6	−3.5547898	−1.2310038	1.1776542	3.4779865
7	−33.272681	−0.048636	0.2508887	30.053423
8	−2398777.8	0.117647	0.1177055	1442184.5
9	-4.6719×10^{30}	0.1176483	0.1176483	
10		0.1176483	0.1176483	

从以上试验可以得出两个结论:

(1) 点2是个孤立的、很不稳定的不动点，迭代出发点x_0与点2差一点点，迭代的结果就完全不同。

(2) −2和−3之间的分界点在−2.0590与−2.0589之间。

下面我们用另一种不但一学就会而且必定成功的迭代法把这个分界点确定下来，这就是分区间迭代法。

我们要解的是方程 $f(x)=0$，如果我们已经知道两点a、$b(a<b)$，使得 $f(a)\cdot f(b)<0$，即$f(a)$和$f(b)$符号相反，那么在(a, b)中取中点c，计算$f(c)$。如果$f(c)=0$，则解已求得，不必再迭代下去。否则，如果$f(a)\cdot f(c)<0$，可知$f(c)$与$f(b)$同号，就扔掉原来的b，把c作为新b，继续迭代；如果$f(b)\cdot f(c)<0$，可知$f(c)$与$f(a)$同号，就扔掉原来的a，把c作为新a，继续迭代。这就是同符号顶替的原则。这样，每迭代一次，区间(a, b)的长度就缩小一半，而在区间的两端，函数$f(x)$的符号总是相反的。于是，根据$f(x)$的连续性，这些每次缩短一半的区间最后套住一个点，这个点就是$f(x)=0$的解。

现在我们就来试一试。由前已知，−2.0590与−2.0589之间应该有一个分界点，我们就从$a=-2.0590$和$b=-2.0589$开始。$f(a)=-3.817\times10^{-3}<0$，$f(b)=3.469\times10^{-3}>0$，正好符合$f(a)\cdot f(b)<0$，取区间$(a, b)$的中点$c=-2.05895$，计算$f(c)$，这时不必记录具体结果，只要知道正负就可以了，算得$f(c)<0$，与$f(a)$同号，所以按照同号顶替的原则，取c作为新a，下次迭代就取$(a, b)=(-2.05895, -2.05890)$。如果拘泥于中点，下次就该取$c=-2.058925$。但对分区间有一个好处，$c$取偏一点关系不大，只要不超出$(a, b)$就行了。为了不一下子增加数字的长度，我们取$c=-2.05893$，算得$f(c)>0$，再取$c=-2.05894$，也得$f(c)>0$，接下去

$c=-2.058945$，$f(c)>0$

$c=-2.058948$，$f(c)<0$

$c=-2.0589477$，$f(c)<0$

$c=-2.058947$，$f(c)>0$

$c=-2.0589475$，$f(c)>0$

$c=-2.0589476$，$f(c)>0$

这已经很精确了，我们就取$c=-2.0589476$作为$f(x)=0$的一个近似解，这时$f(c)=1.2\times10^{-6}$，

这个解已经精确了。

至此，我们找到了 $f(x)=0$的三个解，也就是$x_{n+1}=\phi(x_n)$迭代的三个不动点，即2、0.1176483和−2.0589476。经过这样一番研究，我们对$x_{n+1}=\phi(x_n)$迭代的收敛情况有更加准确的了解，这些情况总结如图1-2所示。

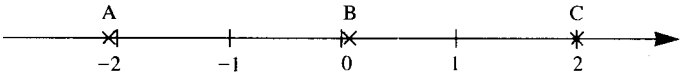


图 1-2

方程 $f(x)=x^5-17x+2=0$共有三个不同的实数解，它们是$x_1=-2.058976$，$x_2=0.1176483$和$x_3=2$。从不动点迭代 $x_{n+1}=\phi(x_n)=\dfrac{x_n^5+2}{17}$ 的角度来看，A和C都是孤立的、不稳定的不动点，在A和C附近开始迭代，出发点差一点点，迭代结果就会面目全非。如果以迭代的结果来划分实数轴，那么(−∞, A)是−∞的地盘，(A, C)是B的地盘，(C, +∞)是+∞的地盘，而A的地盘只有它本身这个点{A}，C的地盘也只有它本身这个点{C}。

下面再试一个例子。

考虑代数方程 $f(x)=x^2-3=0$。这个方程太简单了，但着眼于方法的讨论，我们为什么非要复杂的方程不可呢？找复杂的方程来演示，会本末倒置，花费许多精力在技术细节上会妨碍我们对问题实质的认识，所以，这里宁愿用简单的方程。

我们采用4种方案，把$f(x)=0$变成不动点方程:

(1) $x=x+(x^2-3)$

(2) $x=\dfrac{3}{x}$

(3) $x=\dfrac{x+\dfrac{3}{x}}{2}$

(4) $x=x+\dfrac{x^2-3}{4}$

将不动点方程右端的表达式看作$\phi(x)$，就可以将迭代公式$x_{n+1}=\phi(x_n)$具体写为如下形式:

(1) $x_{n+1}=x_n+(x_n^2-3)$

(2) $x_{n+1}=\dfrac{3}{x_n}$

(3) $x_{n+1}=\dfrac{x_n+\dfrac{3}{x_n}}{2}$

(4) $x_{n+1}=x_n+\dfrac{x_n^2-3}{4}$

这4个迭代都从$x_0=2$开始，迭代情况如下:

n	(1)	(2)	(3)	(4)
0	2.0	2.0	2.0	2.0
1	3.0	1.5	1.75	1.75
2	9.0	2.0	1.732143	1.734375
3	87.0	1.5	1.732051	1.732361
4			1.732051	1.732092
5				1.732056
6				1.732051
7				1.732051

迭代(1)发散到$+\infty$，不收敛；迭代(2)振荡，也不收敛；迭代(3)和(4)都收敛到方程的解。虽然(3)和(4)都收敛，但是很明显，迭代(3)收敛得比迭代(4)快。

大家是否注意到，迭代(1)和(4)都可以表示成统一的形式:

$$x = x + c\,(x^2 - 3)$$

当取$c = 1$时就得(1)，迭代不收敛；当取$c = \frac{1}{4}$时就得(4)，迭代就收敛。这个c的选择很讲究。选得好，就收敛，甚至收敛很快；选得不好，就会收敛得很慢，甚至根本不收敛。大家不妨自己选$c = \pm\frac{1}{2}$或$\pm\frac{1}{8}$等试一试。

从上面得例子可以看出，方程$f(x) = 0$可以改写为多种不动点方程，同一个不动点方程也可以选取多个初值，那么应该怎样构造不动点方程、怎样选择初值呢？下面我们来推导求方程$x = \phi(x)$的根的迭代过程的收敛性定理和误差估计。

定理1-1　设函数$\phi(x)$在有限区间$[a, b]$上满足如下条件:

(1) 当$x \in [a, b]$时，$\phi(x) \in [a, b]$，即$a \leqslant \phi(x) \leqslant b$。

(2) 对任意的$x_1, x_2 \in [a, b]$，恒成立：$|\phi(x_1) - \phi(x_2)| \leqslant L|x_1 - x_2|$。

($\phi(x)$在$[a, b]$上满足Lipschitz条件，L称为Lipschitz常数）且$L < 1$，则方程$x = \phi(x)$在$[a, b]$上的解α存在且惟一，并且对任意$x_0 \in [a, b]$，由迭代过程$x_{n+1} = \phi(x_n)$产生的序列$\{x_k\}$收敛到α。

证明：首先证明$x = \phi(x)$的解存在。作函数

$$g(x) = x - \phi(x)$$

显然$g(x)$在$[a, b]$上连续。由条件(1)可知

$$g(a) = a - \phi(a) \leqslant 0,\quad g(b) = b - \phi(b) \geqslant 0$$

由连续函数根的存在定理可知：必有$\alpha \in [a, b]$使$g(\alpha)=0$，即$\alpha = \phi(\alpha)$。因此，α是方程$x = \phi(x)$的解。

其次证明惟一性，假设存在两个解α_1, α_2，则

$$\alpha_1 = \phi(\alpha_1),\quad \alpha_2 = \phi(\alpha_2),\quad \alpha_1, \alpha_2 \in [a, b]$$

因此，由条件(2)有

$$|\alpha_1 - \alpha_2| = |\phi(\alpha_1) - \phi(\alpha_2)| \leqslant L|\alpha_1 - \alpha_2|,$$

因为$L<1$，所以必有$\alpha_1 = \alpha_2 = \alpha$。

再来证明$\{x_k\}$的收敛性。按迭代过程$x_{n+1} = \phi(x_n)$，有

$$x_k - \alpha = \phi(x_k) - \phi(\alpha)$$

由条件(2)得

$$|x_k - \alpha| = |\phi(x_{k-1}) - \phi(\alpha)| \leqslant L|x_{k-1} - \alpha| \leqslant \cdots \leqslant L^k|x_0 - \alpha|,$$

因$L<1$，所以

$$\lim_{k\to\infty} x_k = \alpha$$

证毕

推论1-1　若条件(2)改为$\phi'(x)$在$[a, b]$上有界，且

$$当x \in [a, b]时，\ |\phi'(x)| \leqslant L < 1$$

则定理结论成立。

上面证明迭代过程的收敛性，理论上要得到精确解α，一般需要进行无穷次迭代循环，这当然是不现实的，实际应用中也只需求得满足精度要求的近似值，为此，我们要估计经过n次迭代后的误差$\varepsilon_n = x_n - \alpha$。

定理1-2　设函数$\phi(x)$在有限区间$[a, b]$上满足如下条件：

(1) 当$x \in [a, b]$时，$\phi(x) \in [a, b]$，即$a \leqslant \phi(x) \leqslant b$。

(2) $\phi(x)$在$[a, b]$上满足Lipschitz条件，且$L<1$。

则误差估计式

$$|x_n - \alpha| \leqslant \frac{L^n}{1-L}|x_1 - x_0|$$

成立。

证明：对任意正整数$m>n$有

$$x_m - x_n = \sum_{k=n}^{m-1}(x_{k+1} - x_k)$$

由Lipschitz条件得

$$|x_{k+1} - x_k| = |\phi(x_k) - \phi(x_{k-1})| \leqslant L|x_k - x_{k-1}| \leqslant \cdots \leqslant L^k|x_1 - x_0|$$

所以

$$\begin{aligned}|x_m - x_n| &\leqslant \sum_{k=n}^{m-1}|x_{k+1} - x_k| \leqslant \sum_{k=n}^{m-1} L^k|x_1 - x_0| \\ &\leqslant L^n \sum_{k=0}^{\infty} L^k|x_1 - x_0| \leqslant \frac{L^n}{1-L}|x_1 - x_0|\end{aligned}$$

在上式中，令$m\to\infty$，由定理1-1可知 $\lim\limits_{m\to\infty}x_m=\alpha$，所以

$$|\varepsilon_n|=|x_n-\alpha|<\frac{L^n}{1-L}|x_1-x_0|$$

证毕

在实际计算中，上式也可用来估计达到要求精度ε所需要的迭代循环次数，由要求

$$\frac{L^n}{1-L}|x_1-x_0|<\varepsilon$$

得

$$n>\left\lfloor\frac{\ln\dfrac{\varepsilon(1-L)}{|x_1-x_0|}}{\ln L}\right\rfloor$$

这里，$\lfloor x\rfloor$表示不大于x的最大整数。

1.2　收敛性和复杂性——算法优劣判别的两个层次

数学研究最终还是要解决实际问题。如果面对的是方程求解问题，那么首先就要回答方程有没有解这个所谓解的存在性问题。方程有多少个解，解在什么地方等也都属于这类问题。

讨论存在性有两种基本方式：一种是构造性的证明方法，就是具体设计一种方法把解找出来，从而说明解是存在的。找到了的东西当然是存在的东西，这就是构造性方式的哲学。另一种是非构造性的证明方法，其代表就是反证法：假定没有解，然后通过推理，引出与已知事实的矛盾。

反证法在逻辑上常常是漂亮的，但是带给人们的信息较少。例如，面对3只雏鸡，你看也不看就可以作出“其中必有两只雏鸡的性别相同”这样一个存在性的判断，因为不然的话，就和鸡只有雌雄两个性别的已知事实矛盾。这个判断过程就是非构造性的。倘若你是一个识别鸡的行家，确实辨认出其中有两只雌鸡或雄鸡，并由此得到“有两只雏鸡的性别相同”的判断，这个判断的论证过程就是构造性的。两相比较：构造性的判断方式具体指出了两只雌鸡或雄鸡，所以提供的信息较多，而非构造性的判断，在这个雏鸡识别的问题里，没有提供一点儿有实用价值的信息。

但是在数学发展的漫长历程中，非构造性的讨论方法作用很大，功不可没。我们知道，社会要求和内部动力是数学发展的两大激励因素。非构造性的讨论方式，常常就是数学大系统的内部动力的体现。另外，除了在没有构造性方法时自然欢迎非构造性方法外，我们还应注意，人类的认识都是相互联系的，非构造性方法得到的结论常常可以给研究构造性方法指引方向。最典型的例子就是，数学家已经用非构造性的方法证明了不存在用圆规和直尺三等分任意角的通用方法、不存在高次方程的代数解法。这就使后人可以避免在寻求三等分角和高次方程代数

解方面空耗精力（论证某种东西不存在和论证某种东西存在一样，都属于存在性问题）。相反，如果对于一个问题，数学家已经用非构造性方法证明了解是一定存在的，这就更明确地指引后人去寻求把解具体找出来的构造性方法。最典型的例子就是代数基本定理。

多项式有没有根，有多少个根，这在二百年前就已经清楚了。论述这一事实的定理，就是著名的代数基本定理：**任一n次（复系数）多项式恰好有n个（复）根。**

大家知道，德国数学家高斯从1799年到1850年前后跨越半个世纪的时间里，曾经提出代数基本定理的四种证明。更早，牛顿、麦克劳林、达朗贝尔、欧拉和拉格朗日都从事过这一工作。他们提供的证明基本上都是非构造性的证明，主要思路就是如果没有根，会引出什么样的矛盾。

随着科学的发展，各方面对数学的要求越来越倾向于构造性的解决办法。构造性的方法虽然做起来有时比较辛苦，但它不仅肯定了“存在”的事实，而且还告诉人们怎样把这个存在找出来。构造性方法虽然计算比较繁复，但随着计算机的发展和普及，“繁复”的弱点大多已经不成问题，构造性方法正好可以借助计算机扬长避短，向人们提供满意的答案。我们下一章要介绍的库恩（Kuhn）算法就是代数基本定理的构造性证明方法。

既然我们强调构造性工作的实际应用价值，那么，面对一种算法，第一要问它是否成功，第二要问它的效率如何。不成功的算法不能把解求出来，当然没用。成功但效率很低的算法，也没有多少价值。如果有人告诉你一种算法，并且在数学上证明了按这种算法一定可以找到问题的解，但是求解要在最新的计算机上花费1万年的时间，你对这样一种实际上无法实现的构造性方法会有什么感想？恐怕是啼笑皆非吧。

算法是否成功是收敛性问题，收敛的就成功，不收敛即发散的就不成功。效率如何是算法的复杂性问题。复杂性也是我们介绍的主题。

效率的高低指的是收敛的快慢，这是毫无疑问的。同一种算法，解小规模的问题花时间少，解大规模的问题花时间多，这是很自然的事情。问题是随着问题规模的增长，所需要的计算时间怎样增长？以代数方程求解来说，如果方程的次数为n，求解所需要的时间为T，我们把它暂记为$T(n)$。$T(n)$与n究竟是什么关系呢？即使没有明确的函数关系，也要尽可能把它们的关系反映出来。

因为工作量的大小与工作效率的高低总是可以用工作时间来表示，所以我们称$T(n)$为解法或算法的时间复杂性函数。

不同的算法具有不同的时间复杂性函数，说它们当中哪些“效率足够高”，哪些“效率太低”，这要看当时的情况。但是，计算机科学和计算数学科学家们公认一种简单的区别，这种区别对这些问题提供了重要的透彻的分析，这就是多项式时间算法和指数时间算法的区别。

定义1-1　称一个函数$g(n)$是$O(f(n))$（小于等于$f(n)$的阶），当且仅当存在常数$c>0$和$n_0\geqslant 0$，对一切$n\geqslant n_0$均有$|g(n)|\leqslant c|f(n)|$成立，也称函数$g(n)$以$f(n)$为界或者称$g(n)$囿于$f(n)$。记作$g(n)=O(f(n))$。

按此定义，如果$g(n)=O(n^2)$，则一定有$g(n)=O(n^3)$，为了更精确地说明一个算法的复杂性，有时引入记号$\Theta(f(n))$，表示阶恰好为$f(n)$的函数。