利用 TI 图形计算器学习与研究递归算法

广东省中山市东升镇高级中学　(528400)　高建彪

算法是计算科学的基础,是信息时代的现代人应具备的一种数学素养. 在高中数学新课程标准实验教材中,算法是模块《数学3》必修的学习内容,同时算法的思想方法也渗透到了高中数学的其他有关内容之中. 下面我们借助 TI 图形计算器,结合高中数学教材,学习与研究算法中的一种重要算法——递归算法.

一、递归算法的初步认识

递归是指函数、过程或子程序,在运行过程中直接或间接调用自身而产生的重入现象. 递归是计算机科学的一个重要概念,递归的方法是程序设计中有效的方法,采用递归编写程序能使程序变得简洁和清晰. 递归作为一种算法,在程序设计语言中广泛应用.

递归算法解决问题的特点:

(1) 递归就是在过程或函数里调用自身;

(2) 在使用递归策略时,必须有一个明确的递归结束条件,称为递归出口;

(3) 递归算法解题通常显得很简洁,但递归算法解题的运行效率较低,占用时间长;

(4) 在递归调用的过程当中系统为每一层的返回地址、局部变量、参数等开辟了栈来存储,占用的栈空间很大,递归次数过多也容易造成栈溢出等. 所以一般不提倡用递归算法设计程序.

要理解递归算法,可以结合下面三个实例进行浅显的理解.

实例 1　老和尚讲故事

从前有座山,山里有个庙,庙里有个老和尚会讲故事,讲什么呢?从前有座山,山里有个庙,庙里有个老和尚会讲故事 ……

实例 2　两面镜子互映

在每面镜子中都有对面镜子的像,同时在像中也应该有和像同样的场景 ……

实例 3　小学数学趣味题

有一天小猴子摘若干个桃子,当即吃了一半还觉得不过瘾,又多吃了一个. 第二天接着吃剩下桃子中的一半,仍觉得不过瘾又多吃了一个,以后小猴子都是吃尚存桃子一半多一个. 到第 10 天早上小猴子再去吃桃子的时候,看到只剩下一个桃子. 问小猴子第一天共摘下了多少个桃子?

二、高中数学教材中的经典递归

1　斐波那契数列

1202 年,意大利数学家斐波那契(Leonardo Fibonacci,约 1170 – 1250),在他的著作《算盘书》中提出一个著名的"兔子繁殖问题":如果每对兔子每月繁殖一对子兔,而子兔在出生后第二个月就有生殖能力,试问第一月有一对小兔子,第十二月时有多少对兔子?

假设第 n 个月的兔子数目为 $F(n)$,那么 $F(n)=\begin{cases}F(n-1)+F(n-2)\\1,n=1,2\end{cases}$. 对此问题的求解,我们利用TI图形计算器,分别用递归的方法与FOR循环语句进行探索与求解如下.

操作提示:按 ctrl- 4 1 1 可插入一个新问题下的计算页,按 menu 9 1 1 新建一个程序,按 menu 2 1 检查语法并保存,按 ctrl ⇧ 跳转到计算窗口.

对比分析:从语言的简洁性来看,用递归算法写的程序简单多了,然而笔者用以上两个程序,借助秒表工具,进行了对比测试,递归算法计算时,用了将近 30 秒,计算时将像陷入了死循环,而循环结构的程序计算用时不到 1 秒. 为何会如此,我们可以思考一下,递归算法计算 $F(25)$ 时,要调用多少个 $F(n)$ 才能完成运算.

注意一点,循化结构的程序实质是一种迭代,递归与迭代有一定的相似性,但递归是调用自身,迭代只是运用上一计算结果.

2 “辗转相除法”求最大公约数

“辗转相除法”是求两个正整数的最大公约数的一种算法,它出自于古希腊数学家欧几里得(Euclid 活动于公元前 300 年左右)所著的《几何原本》.《几何原本》是一本世界数学名著,在各国流传很广,仅次于基督教的《圣经》. 该著作开辟了一条数学公理化的正确道路,对整个数学发展的影响超过了历史上任何一步著作.

辗转相除法的具体算法:用较大的数 m 除以较小的数 n,得到余数 r,即除式 $m = nq + r(0 \leqslant r \leqslant n)$,再用除数 n 除以余数 r,得到新的余数. 反复执行这一步,当某步余数为 0 是,该步的除数就是最大公约数.

我们利用 TI 图形计算器,分别用递归的方法与当循环的语句进行探索与求解如下.

求最大公约数还有另外一种算法,即“更相减损术”,它出自于中国古代最著名的传世数学名著《九章算术》,成书于公元1 世纪,汇总了中国先秦至汉代的数学成就,代表了东方数学的最高成就. 该书收录了与实际生活紧密相联的许多数学问题,分为九章 246 问 202 术,“术”就是对问题的解法.

更相减损术的具体算法:用较大的数 m 减去较小的数 n,得到差 d,即减式 $m - n = d$,再把差数 d 与小数 n 比较,仍然采用以大减小的方式,反复地辗转相减,直到最后两数相等,则相等的数就是最大公约数.

3 二分法求方程近似解

方程 $f(x) = 0$ 的近似解,就是函数 $f(x)$ 的零点,给定精度 ε,用二分法求函数 $f(x)$ 的零点近似值的步骤如下:

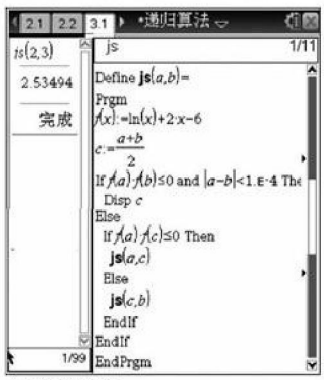
*S*1 确定区间 $[a,b]$,验证 $f(a) \cdot f(b) < 0$,给定精度 ε;

*S*2 求区间 (a,b) 的中点 x_1;

*S*3 计算 $f(x_1)$:若 $f(x_1) = 0$,则 x_1 就是函数的零点;若 $f(a) \cdot f(x_1) < 0$,则令 $b = x_1$(此时零点 $x_0 \in (a,x_1)$);若 $f(x_1) \cdot f(b) < 0$,则令 $a = x_1$(此时零 $x_0 \in (x_1,b)$ 点);

*S*4 判断是否达到精度 ε,若 $|a-b| < \varepsilon$,则得到零点值 a(或 b);否则重复步骤 *S*2 ~ *S*4.

下面举一例说明,利用二分法的思想,编程求方程 $\ln x + 2x - 6 = 0$ 的近似解(精确到0.0001). 我们分别用递归方法与循环结构编程如下.

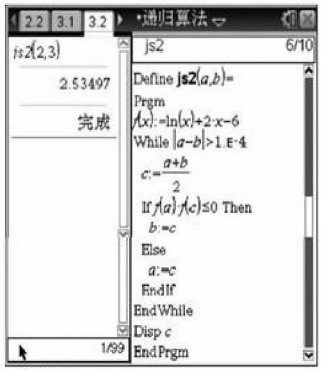


4 汉诺塔问题

在人教 *A* 版高中数学教材《选修 2 - 2》第 75 页,例 4 所研究问题的拓展就是起源于印度一个古老传说的汉诺塔问题,问题描述如下:

古代有一个梵塔,塔内有 3 个基座 *A*、*B*、*C*,开始

时 A 基座上有 64 个盘子，盘子大小不等，大的在下，小的在上. 有一个老和尚想把这 64 个盘子从 A 座移到 B 座，但每次只允许移动一个盘子，且在移动过程中在 3 个基座上的盘子都始终保持大盘在下，小盘在上. 在移动过程中可以利用 C 基座做辅助. 请编程求解移动过程.



算法分析：

用归纳法解此题，约定盘子自上而下的编号为 $1,2,3,\cdots\cdots,n$.

首先看一下 2 阶汉诺塔问题的解，不难理解以下移动过程：

初始状态为：**A**(1,2)　**B**(　)　**C**(　)

第一步后：　**A**(2)　**B**(　)　**C**(1)

第二步后：　**A**(　)　**B**(2)　**C**(1)

第三步后：　**A**(　)　**B**(1,2)　**C**(　)

如何找出大规模问题与小规模问题的关系，从而实现递归呢？

把 n 个盘子抽象地看作“两个盘子”，上面“一个”由 1—$n-1$ 号组成，下面“一个”就是 n 号盘子，从而移动过程如下：

第一步：先把上面“一个”(1—$n-1$ 号) 盘子以 a 基座为起点借助 b 基座移到 c 基座；

第二步：把下面“一个”盘子(n 号) 从 a 基座移到 b 基座；

第三步：再把 c 基座上的 $n-1$ 盘子借助 a 基座移到 b 基座.

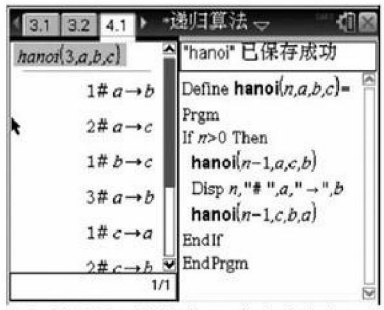
把 n 阶的汉诺塔问题的模块记作 hanoi(n,a,b,c)，a 代表每一次移动的起始基座，b 代表每一次移动的终点基座，c 代表每一次移动的辅助基座，则汉诺塔问题 hanoi(n,a,b,c) 等价于以下三步：

第一步，hanoi($n-1$,a,c,b)；

第二步，把下面“一个”盘子(n 号) 从 a 基座移到 b 基座；

第三步，hanoi($n-1$,c,b,a).

至此找出了大规模问题与小规模问题的关系. 用 TI 图形计算器编写的程序如下.



（注：若用循环结构编程，将十分复杂）

三、小结语

递归是一种强有力的算法设计方法，是一种比循环更强、更好用的实现“重复操作”的机制. 由于递归不需要编程人员自己构造“循环体”，而只需要找出递归关系和最小问题的解，因而递归在很多算法策略中得以运用. 在文章最后，我们再展示出阶乘的递归算法(下图)，希望能透过简单的实例理解递归算法.

同时，我们把递归与非递归进行比较，得出以下结论：

	程序可读性	代码量大小	运行时间	占用空间	适用范围	设计难度
递归	易	小	长	大	广	易
非递归	难	大	短	小	窄	难