一种基于数组的递归算法

A Recursive Algorithm Based on Array

汤 薇,秦 华,廖 瑞

Tang Wei,Qin Hua,Liao Rui

(桂林空军学院训练部教育技术中心,广西桂林 541000)

(Center of Education and Technology, The Department of Training, Guilin Air Force Academy,Guilin,Guangxi,541000,China)

摘要:以Hanoi塔问题为例,分析递归程序运行速度慢的原因,提出一种基于数组的递归算法。该算法可以使计算机程序的计算速度提高到最快。

关键词:递归 算法 数组

中图法分类号:TP301 **文献标识码**:A **文章编号**:1002-7378(2005)04-0202-02

Abstract: The Hanoi tower question is used as an example to analyze the causes of slow operation of the recursive routine. A recursive algorithm based on array is presented. As shown by the tests, the recursive algorithm can speed up the computational speed of procedure.

Key words: recursion, algorithm, array

递归是计算机程序设计中经常用到的方法,因为数学中的许多函数是用递归方法定义的。用递归方法编写出来的程序具有简洁、清晰、易理解的特点[1]。但是,递归程序的计算速度慢,如果问题规模稍大,计算时间会很长,特别是对于实时控制,根本无法实现。为此,人们做出各种努力去寻找新的算法。到目前为止,主要采用的算法是用堆栈代替递归[2]。这样计算速度也许会快一些,但是本质上没有变化,而且使原递归程序的简洁易懂的优点丧失殆尽,计算速度也没有从根本上解决。针对这个问题,本文提出一种基于数组的递归算法,既保留原递归程序的简洁易懂特点,又使程序运行速度提高到较快水平。

1 传统递归算法

Hanoi塔问题是个典型的递归问题,现扩大问题规模,把"3柱"变成"多柱"。

假设$X(>=3)$个柱,第1柱上有$Y(>=1)$个盘子(盘子大小各不相同,且只许大盘在下,小盘在上),把第1柱的Y个盘子移动到第X个柱上最少需多少步(规定每次只能移动1个盘子,且保证大盘在下,小盘在上)。

设利用函数hanoi (int x,int y) 实现之。

$$hanoi(x,1) = 1, \quad (1)$$

$$hanoi(3,y) = 2^y - 1, \quad (2)$$

$$hanoi(x,y) = \min(2 * hanoi(x,k) + hanoi(x-1,y-k)), k = 1,2,3,\cdots,y-1。 \quad (3)$$

根据上面的分析,不难写出以下递归程序。

```
/*Hanoi 塔——递归程序*/
long hanoi(int x,int y)
{int k ; long m,n;
if (y= =1) m=1;
else if(x= =3) m= (1L<<y)-1;
else
   {m=hanoi(x,1)*2+hanoi(x-1,y -1);
   for(k=2;k<y;k++)
   {n=hanoi(x,k)*2+hanoi(x-1,y-k);
   if(n<m) m=n;
   }
   }
   return m;
}
main( )
{int x,y;
printf("输入柱子数(X>=3):");
scanf("%d",&x);
```

收稿日期:2005-06-27

作者简介:汤 薇(1979-),女,广西全州人,助教,主要从事计算机教学和研究工作。

```
printf("输入盘子数(Y>=1):");
scanf("%d",&y);
printf("\n 最少需要的移动的步数:%ld",hanoi(x,y));
}
```

此程序利用典型的传统递归算法编写,它的致命缺点是:递归调用次数多,重复调用次数多,所以运算速度很慢。如:计算hanoi(4,5),共递归调用了37次,它们是:hanoi(4,5)1次;hanoi(4,4)、hanoi(3,4)各2次;hanoi(4,3)、hanoi(3,3)各4次;hanoi(4,2)、hanoi(3,2)各4次;hanoi(4,1)、hanoi(3,1)各8次。又如,计算hanoi(4,10)共需递归调用1 023次,而计算hanoi(4,23)共需调用8388607次,如果x、y的值更大些,计算时间就更长了。由此可见,传统递归算法中采用的数据结构和调用方式不利于有效解决递归算法耗时长的问题。

2 基于数组的递归算法

将递归调用改为对数组元素的操作,即先计算出函数值,将其存入数组中,以后需要时,不再计算,直接从数组中取出该值即可。这样就能彻底消除递归,将计算复杂度为指数的函数运算转变为多项式的计算,算法流程见图1所示。

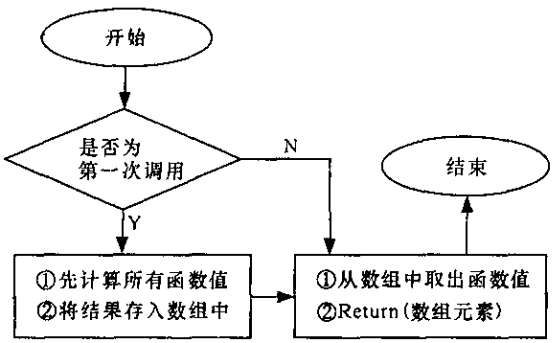

图1 基于数组的递归算法流程

程序如下:

```
/*Hanoi 塔——基于数组的递归程序*/
long a[31][31];
long hanoi(int x,int y)
{static int f=1;
  int h,j,k;
  long m,n;
  if(f)
  {f=0;
    for(h=3;h<=30;h++)
      a[h][1]=1L;
    for(j=2;j<=30;j++)
      a[3][j]=(1L<<j)-1;
    for(h=4;h<=30;h++)
    for(j=2;j<=30;j++)
      {m=a[h][1]*2+a[h-1][j-1];
        for(k=2;k<j;k++)
          {n=a[h][k]*2+a[h-1][j-k];
            if(n<m) m=n;
          }
      a[h][j]=m;
      }
    }
  return a[x][y];
  }
  main()
    {int x,y;
      printrf("输入柱子数(3<=X<=50):");
      scanf("%d",&x);
      printf("输入盘子数(1<=Y<=50):");
      scanf("%d",&y);
      printf("\n 最少需要的移动的步数:%ld",hanoi(x,y));
    }
```

程序执行中,先判断变量f,若其值为非零,表示是第一次调用,然后根据公式(1)、(2)、(3),计算所有函数值,并存入数组a中的相应位置,再返回$a[x][y]$。如果f为零,直接返回$a[x][y]$即可。利用这种算法编写的程序执行速度很快,几乎可以达到忽略不计的程度。

3 结束语

以上利用Hanoi塔问题作为例子介绍基于数组的递归调用算法的思想和实现方法。这个例子很具有代表性,Hanoi塔问题是二元的,对于多元递归函数也可照此处理。当然,速度的提高是要付出代价的,即该算法的代价是增加了内存的开销。

参考文献:

[1] 谭浩强.C程序设计[M].第2版.北京:清华大学出版社,1999.

[2] 王海深.递归程序设计的理论基础探讨[J].小型微型计算机系统,1997,18(2):77-80.

(责任编辑:邓大玉)