【文章编号】 1004-1540(2005)03-0212-06

Hanoi 塔问题非递归算法的比较与研究

仇苏恺，卢芳芳
（中国计量学院 信息工程学院，浙江 杭州 310018）

【摘 要】 作者对 Hanoi 塔问题现有的五种递归算法和非递归算法进行了时间测试，发现现有的非递归算法随着盘子数 n 的增大，在时间效率上不如递归算法，在空间效率上的优势也不明显. 作者采用编码的方法提出了一种新的非递归算法. 本算法在时间效率上较大地优于现有的非递归算法，也明显地优于递归算法. 并且可以根据需要，增大编码的重数，从而使时间效率成倍提高.

【关键词】 Hanoi 塔；非递归算法；递归算法；编码；时间效率
【中图分类号】 TP301.6 【文献标识码】 A

Comparason and study on the non-recursive algorithms of Hanoi tower

QIU Su-kai, LU Fang-fang
(College of Information Engineering, China Jiliang University, Hangzhou 310018, China)

Abstract: Present five solutions to Hanoi tower problem and calculates their running time have studied and compared in this paper. It is discovered that as n increases, the four non-recursire algorithms of papers cost much more time than the recursire algorithm. This paper proposes a new non-recursive algorithm which bases on coding. This new algorithm will cost less time than the present non-recursive algorithms and also the recursive algorithm. Furthermore, you can code many times and the time cost can be reduced consequently.

Key words: Hanoi tower; non-recursion algorithm; recursion algorithm; coding; time efficiency

Hanoi 塔问题的经典解法是递归算法. 通常递归算法是低效的，原因在于递归算法存在着大量的重复调用，即对相同问题的重复计算，Fibonacci 问题是个很好的例子. 但 Hanoi 塔问题却不同，无论是在形式上还是在实际执行过程中，它的递归算法都是简洁的. 这是因为 Hanoi 塔问题的重复调用并非“绝对的”重复，在求解 n 个盘子的 Hanoi 塔问题时对 $n-1$ 个盘子的 Hanoi 塔问题的两次调用虽然都是移动 $n-1$ 个盘子，但源柱和目的柱并不完全相同. 而且每次调用时仅执行一条简单的输出语句. 但当输入规模 n 较大时，递归算法的执行速度越来越慢. 问题不仅在于重复调用，还在于解的输出. 输入规模为 n 的 Hanoi 塔问题的解有 2^n-1 次移动[2]. 倘若不考虑进栈出栈过程，只进行输出. 若每次只输出一个移动步骤，那么当 n 大于 30 时，要输出 2^n-1 个移动步

【收稿日期】 2005-05-09
【作者简介】 仇苏恺(1980－)，女，山东德州人，英国 Glasgo 大学硕士研究生. 主要研究方向为计算机图形与图像处理等.

骤，其执行时间就要以“天”来计算. 现有的非递归算法也存在类似的问题，而且还要进行复杂的变换或计算，在时间效率上不如递归算法. 当然，递归算法由于参数需要多次进栈出栈，空间效率不高. 但非递归算法中凡是使用显式定义栈或数组存放结果的，空间效率和递归算法差不多. 因此，要想在时间效率上有所突破，就需要批量输出，在空间上有所突破，就需要压缩存储. 本文采用数据编码的思想方法[5]，通过编码较好地解决了这一问题.

1　编码

1.1　递推算法

本文用 Hanoi(n,A,B,C)表示 n 个盘子从 A 借助 B 移到 C，用“A －> C”简记盘子从 A 移到 C. 于是，n 层 Hanoi 塔与(n－1)层 Hanoi 塔问题之间的关系可以用下式表示：

Hanoi(n,A,B,C)＝ Hanoi(n－1,A,C,B)＋“A－>C”＋ Hanoi(n－1,B,A,C).　　(1.1)

记 N 为所求 Hanoi 塔问题的盘子总数. 我们定义初值为递推计算 Hanoi 塔问题时的起点.

1.1.1　当 N ＝ 1 时，解为“A －> C”，记为初值.

1.1.2　当 N ＝ n 时，其中 n> 1，则 N ＝n 的解可从 N ＝ n－1 的解变换而来，变换规则如下：

(1)前一次的结果中 A 不变，B 与 C 互换，记为Ⅰ1.

(2)在结果中加入 “A －> C”.

(3)前一次的结果中 C 不变，A 与 B 互换，记为Ⅰ2.

1.2　编码规则

1.2.1　将 N ＝ 1 时的解作为初值

当 N＝2 时，根据 1.1.2，我们将 N＝1 的解“A→C”做变换Ⅰ1，得到移动“A→B”. 然后加入“A→C”. 再将该解做变换Ⅰ2，得到移动“B→C”. 于是得到 N＝2 时的解，即：“A→B”，“A→C”，“B→C”. 新增了移动“A→B”，“B→C”.

当 N＝3 时，根据 1.1.2，我们将 N＝2 的解做变换，于是得到 N＝4 时的解，即：“A→C”，“A→B”，“C→B”，“A→C”，“B→A”，“B→C”，“A→C”. 新增了移动“C→B”，“B→A”.

同理可得，当 N＝4 时，新增了移动“C→A”. 当 N＝5 时，就没有新的移动了. 由此可见，Hanoi 塔问题的解中仅含六种移动：“A→B”，“A→C”，“B→A”，“C→A”，“B→C”，“ C→B”. 于是根据 1.1.2，从 N＝n－1 的解推出 N＝n 的解的变换规则可重新描述如下：

1.2.1.1

(1)遍历 N＝n－1 的解，每种移动按如下规则变换(其中＝＝>表示变换为)：

$$\begin{cases} \text{"A}->\text{B"}==>\text{"A}->\text{C"} \\ \text{"A}->\text{C"}==>\text{"A}->\text{B"} \\ \text{"B}->\text{A"}==>\text{"C}->\text{A"} \\ \text{"C}->\text{A"}==>\text{"B}->\text{A"} \\ \text{"B}->\text{C"}==>\text{"C}->\text{B"} \\ \text{"C}->\text{B"}==>\text{"B}->\text{C"} \end{cases} \quad (\text{变换}\text{Ⅱ}_1)$$

(2)在结果中加入 “A －>C”.

(3)遍历 N＝n－1 的解，每种移动按如下规则变换：

$$\begin{cases} \text{"A}->\text{B"}==>\text{"B}->\text{A"} \\ \text{"A}->\text{C"}==>\text{"B}->\text{C"} \\ \text{"B}->\text{A"}==>\text{"A}->\text{B"} \\ \text{"C}->\text{A"}==>\text{"C}->\text{B"} \\ \text{"B}->\text{C"}==>\text{"A}->\text{C"} \\ \text{"C}->\text{B"}==>\text{"C}->\text{A"} \end{cases} \quad (\text{变换}\text{Ⅱ}_2)$$

我们用字符“1”－“6”分别表示以上六步，编码规则如下(其中<＝＝>表示等价于)：

$$\begin{cases} \text{"A}->\text{B"}<==>\text{"1"} \\ \text{"A}->\text{C"}<==>\text{"2"} \\ \text{"B}->\text{A"}<==>\text{"3"} \\ \text{"C}->\text{A"}<==>\text{"4"} \\ \text{"B}->\text{C"}<==>\text{"5"} \\ \text{"C}->\text{B"}<==>\text{"6"} \end{cases}$$

于是根据 1.2.1.1，从 N ＝n－1 的解推出 N＝n 的解的变换规则可简化如下：

1.2.1.2

(1)遍历 N ＝n－1 的解，每种移动按如下规则变换：

$$\begin{cases} \text{"1"}<==>\text{"2"} \\ \text{"2"}<==>\text{"1"} \\ \text{"3"}<==>\text{"4"} \\ \text{"4"}<==>\text{"3"} \\ \text{"5"}<==>\text{"6"} \\ \text{"6"}<==>\text{"5"} \end{cases} \quad (\text{变换}\text{Ⅲ}_1)$$

(2)在结果中加入“2”.

(3)遍历 $N=n-1$ 的解,每种移动按如下规则变换:

$$\begin{cases}"1"<==>"3"\\"2"<==>"5"\\"3"<==>"1"\\"4"<==>"6"\\"5"<==>"2"\\"6"<==>"4"\end{cases}\quad(\text{变换}\mathrm{III}_2)$$

由于从“A->B”,“A->C”,“B->A”,“C->A”,“B->C”,“C->B”仅经过一次变换就得到了最终的编码“1”,“2”,“3”,“4”,“5”,“6”,因此我们称此时的编码为一重编码.若我们用 K 表示编码的重数,则一重编码可简记为 $K=1$.一重编码后,初值仍取 $N=1$ 时的解,记为“2”.

1.2.2 *进行一重编码之后,$N=2$ 时的移动就可表示成“125”.把它作为初值.*

当 $N=3$ 时,根据 1.2.1.2,我们将 $N=2$ 的解“125”做变换 III_1,得到移动“216”.然后加入“2”.再将该解做变换 III_2,得到移动“352”.于是得到 $N=3$ 时的解,它由以下三个字符串组成:“216”,“2”,“352”.新增了移动“216”,“352”,以及“2”.

当 $N=4$ 时,根据 1.2.1.2,我们将 $N=3$ 的解做变换,于是得到 $N=4$ 的解,即:“125”,“1”,“461”,“2”,“534”,“5”,“125”.新增了移动“461”,“534”,“1”,“5”.

同理可得,当 $N=5$ 时,新增了移动“643”,“3”,“6”.当 $N=6$ 时,新增了移动“4”.当 $N=7$ 时,就没有新的移动了.由此可见,Hanoi 塔问题所有可能的移动方式由两部分组成:

一部分由一重编码的字符“1”-“6”组成.它们之间的变换关系为变换 III_1,变换 III_2;另一部分仅六种情形:“216","125","534","643","352","461"

它们之间的变换关系如下:

$$\begin{cases}"216"==>"125"\\"125"==>"216"\\"534"==>"643"\\"643"==>"534"\\"352"==>"461"\\"461"==>"352"\end{cases}\quad(\text{变换}\mathrm{IV}_1)$$

$$\begin{cases}"216"==>"534"\\"125"==>"352"\\"534"==>"125"\\"643"==>"461"\\"352"==>"125"\\"461"==>"643"\end{cases}\quad(\text{变换}\mathrm{IV}_2)$$

我们用仍用字符“1”-“6”分别表示以上六步,于是编码的规则如下:

$$\begin{cases}"216"<==>"1"\\"125"<==>"2"\\"534"<==>"3"\\"643"<==>"4"\\"352"<==>"5"\\"461"<==>"6"\end{cases}\quad(\text{编码 2})$$

它们之间的变换关系为变换 III_1,变换 III_2.

综合(1)(2),从 $N=n-1$ 的解推出 $N=n$ 的解的变换规则同 1.2.1.2.

由于从“A -> B”,“A -> C”,“B -> A”,“C-> A”,“B -> C”,“C -> B”这原始的六步仅经过两次变换就得到了最终的编码“216","125","534",“643", "352", "461",因此我们称此时的编码为二重编码.简记为 $K=2$.

二重编码后,初值取 $N=2$ 的移动,可标记为“2”.从 $N=n-1$ 的解推出 $N=n$ 的解的变换规则同 1.2.1.2.此时,求得的解中,第奇数个字符代表的是二重编码的字符,第偶数个字符代表的是一重编码的字符.因此将二重编码的形式变换为一重编码的形式,规则如下:

1.2.2.1

(1)第奇数个字符做如下替换:

$$\begin{cases}"1"==>"216"\\"2"==>"125"\\"3"==>"534"\\"4"==>"643"\\"5"==>"352"\\"6"==>"461"\end{cases}\quad(\text{变换}\mathrm{V}_1)$$

(2)第偶数个字符不变.

二重编码分别由一重编码经过 1(=$K-1$)次变换得到.即:

“1” == >“216"

“2” == > "125"

“3” == > "534"

“4” == > "643"

“5” == >“352”

“6” == > "461"

1.2.3 我们进行二重编码之后，$N=3$ 时的移动就可表示成“125”. 我们以它作为初值. 推导过程同 $K=2$ 的情况，按编码 2 进行编码. 但此时“216”,"125","534",“643","352", "461"是二重编码时的形式，根据 1.2.2.1，我们将其变成一重编码时的形式：

"216" == > "125 1 461"

"125" == > "216 2 352"

"534" == > "352 3 643"

"643" == > "461 4 534"

"352" == > "534 5 125”

"461" == > "643 6 216"

于是得到了三重编码的最后编码：

"125 1 461"<==>"1"

"216 2 352"<==>"2"

"352 3 643"<==>"3"

"461 4 534"<==>"4"

"534 5 125"<==>"5"

"643 6 216"<==>"6"

由于从“A -> B”,“A -> C”,”B -> A”,“C-> A”,“B -> C”,“ C -> B”这原始的六步仅经过三次变换就得到了最终的编码“125 1 461","216 2 352","352 3 643","461 4 534",“534 5 125”,"643 6 216",因此我们称此时的编码为三重编码. 简记为 $K=3$.

三重编码后，初值取 $N=3$ 的移动，可标记为“2”. 从 $N=n-1$ 的解推出 $N=n$ 的解的变换规则同 2.1.1.2. 此时，求得的解中，第奇数个字符代表的是三重编码的字符，第偶数个字符代表的是一重编码的字符. 因此将三重编码的形式变换为一重编码的形式，规则如下：

1.2.3.1

(1)第奇数个字符做如下替换：

"1"==>"125 1 461"

"2"==>"216 2 352"

"3"==>"352 3 643"

"4"==>"461 4 534"

"5"==>"534 5 125"

"6"==>"643 6 216"

(2)第偶数个字符不变.

三重编码分别由一重编码经过 2(=$K-1$)次变换得到. 每一次变换的规则如下：

(1)第奇数个字符做变换Ⅵ.

(2)第偶数个字符不变. 即：

"1" < == > “216” < == > "125 1 461"

"2" < == > "125" < == > "216 2 352"

"3" < == > "534" < == > "352 3 643"

"4" < == > "643" < == > "461 4 534"

"5" < == > "352" < == > "534 5 125"

"6" < == > "461" < == > "643 6 216"

1.2.4 在三重编码的基础上，以 $N=4$，移动为“125”作为初值. 推导过程同 $K=3$ 的情况，按编码 2 进行编码. 但此时“216","125","534",“643", "352", "461"是三重编码时的形式，根据 1.2.3.1，将其变成一重编码时的形式：

"216" == > “216 2 352 1 643 6 216"

"125" == > "125 1 461 2 534 5 125"

"534" == > "534 5 125 3 461 4 534"

"643" == > "643 6 216 4 352 3 643"

"352" == > "352 3 643 5 216 2 352"

"461" == > "461 4 534 6 125 1 461"

于是得到了四重编码的最后编码：

"1"==>"125 1 461"

"2"==>"216 2 352"

"3"==>"352 3 643"

"4"==>"461 4 534"

"5"==>"534 5 125"

"6"==>"643 6 216"

由于从“A -> B”,“A -> C”,”B -> A”,“C-> A”,“B -> C”,“ C -> B”这原始的六步仅经过四次变换就得到了最终的编码“216 2 352 1 643 6 216","125 1 461 2 534 5 125","534 5 125 3 461 4 534","643 6 216 4 352 3 643","352 3 643 5 216 2 352","461 4 534 6 125 1 461",因此我们称此时的编码为四重编码. 简记为 $K=4$.

四重编码后，初值取 $N=4$ 的移动，可标记为“2”. 从 $N=n-1$ 的解推出 $N=n$ 的解的变换规则同 1.2.1.2. 此时，求得的解中，第奇数个字符代表的是四重编码的字符，第偶数个字符代表的是一重编码的字符. 因此将四重编码的形式变换

为一重编码的形式,规则如下:

1.2.4.1

(1)第奇数个字符做如下替换:

"1" ==> "216 2 352 1 643 6 216"

"2" ==> "125 1 461 2 534 5 125"

"3" ==> "534 5 125 3 461 4 534"

"4" ==> "643 6 216 4 352 3 643"

"5" ==> "352 3 643 5 216 2 352"

"6" ==> "461 4 534 6 125 1 461"

(2)第偶数个字符不变.

四重编码分别由一重编码经过 3(=$K-1$)次变换得到.每一次变换的规则如下:

(1)第奇数个字符做变换Ⅵ.

(2)第偶数个字符不变.即:

"1"<=>"216"<=>"125 1 461"
<=> "216 2 352 1 643 6 216

"2"<=>"125"<=>"216 2 352"
<=> "125 1 461 2 534 5 125"

"3"<=>"534"<=>"352 3 643"
<=>"534 5 125 3 461 4 534"

"4"<=>"643"<=>"461 4 534"
<=>"643 6 216 4 352 3 643"

"5"<=>"352"<=>"534 5 125"
<=>"352 3 643 5 216 2 352"

"6"<=>"461"<=>"643 6 216"
<=>"461 4 534 6 125 1 461"

1.2.x 以此类推.我们用数组 p[]存放编码,p[0]— p[5]分别存放字符"1"—"6"代表的六个编码.当 $K=k$ 时,计算编码的规律如下:

(1)$K=1$ 时,p[0] ="1",p[1] ="2",p[2] ="3",p[3] ="4",p[4] ="5",p[5] ="6"(初值)

(2)$K=k$ 时,则从 $K=1$ 的编码递推计算到 $K=K$ 的编码,共进行($K-1$)次变换,才能得到最终的编码.每一次变换的计算规律如下:

(1)将上一次变换得到的编码做如下处理:第奇数个字符(数组 p[]中下标为偶数的元素)做变换Ⅵ.

(2)第偶数个数字(数组 p[]中下标为奇数的元素)保持不变,计算六次,得到本次变换中字符"1"—"6"代表的编码.

编码后,从 $N=n-1$ 的解推出 $N=n$ 的解的变换规则同 1.2.1.2.

2 Hanoi 塔问题的计算(从 $N=1$ 的解递推计算到 $N=n$ 的解)

2.1 符号说明

N:代表盘子数

K:代表编码的重数

temp2[]:结果数组 1,存放刚计算出的盘子数 N 为及奇数时的解

temp2[]:结果数组 2,存放刚计算出的盘子数 N 为偶数时的解

p[0]—p[5]:存放第 1—6 个编码字符串

2.2 计算过程:

2.2.1 每次输入两个值 K,N,应满足($N>K$),设 $K=k$, $N=n$.

2.2.2 计算编码字符串

P[0]—P[5]分别存放 1—6 个编码字符串,初始化为"1"—"6".对 P[0]—P[5]分别做 k — 1 次变换,变换规则如下:

(1)第奇数个字符(数组 p[]中下标为偶数的元素)做变换Ⅵ.

(2)第偶数个数字(数组 p[]中下标为奇数的元素)保持不变.

于是得到本次变换中字符"1"—"6"代表的编码.

2.2.3 递推求解

temp1[]初始化为 $N=k$ 的解,即"2",temp2[]初始化为 $N=k+1$ 的解,即"125". 若 $N>k+1$,则计算从 $n=k+2$ 开始,需递推计算 $n-k-1$ 次. 若 k 为奇数,则最终结果存放在 temp2[]中,否则存放在 temp1[]中.每一次将上次的结果做如下变换:

(1)做变换Ⅲ_1.

(2)加入"2".

(3)做变换Ⅲ_2.

2.2.3 计算完毕后对结果数组再进行如下变换,以输出最终的移动结果.

$K=1$ 时,将结果做如下变换输出:

$$\begin{cases} "1" ==> "p[0]" \\ "2" ==> "p[1]" \\ "3" ==> "p[2]" \\ "4" ==> "p[3]" \\ "5" ==> "p[4]" \\ "6" ==> "p[5]" \end{cases} \quad (\text{变换Ⅴ})$$

$K \neq 1$ 时，将结果中的第奇数个元素做变换 V 后输出. 将结果中的第偶数个数字如下变换：

"1" ==>"A ->B"

"2" ==>"A ->C"

"3" ==>"B ->A"

"4" ==>"C ->A"

"5" ==>"B ->C"

"6" ==>"C ->B"

3　结果与结论

3.1　补充公式

3.1.1　Hanoi(n,A,B,C) = Hanoi(n − 1,A,C,B) + A->C + Hanoi(n − 1,B,A,C)

= Hanoi(n − 2,A,B,C) + A->B+ Hanoi(n − 2,C,A,B) + A->C + Hanoi(n − 2,B,C,A) + B->C + Hanoi(n − 2,A,B,C)

= Hanoi(n − 2,A,B,C) + 1 + Hanoi(n − 2,C,A,B) + 2 + Hanoi(n − 2,B,C,A) + 5+ Hanoi(n − 2,A,B,C)

因此，输入规模为 n 的 Hanoi 问题的解前个元素正是输入规模为 $n-2$ 的 Hanoi 问题的解，其他部分可由输入规模为 $n-1$ 的 Hanoi 问题的解变换得到.

3.1.2　我们用 K 代表编码的重数，用 N 代表输入的盘子数，用 $N(k,n)$ 表示 $K=k, N=n$ 时的移动总次数的计算公式.

$N=n$ 时，若 $n=k$，则 $N(k,k)=1$；

若 $n>k$，则

$$N(k,n) = 2\times N(k,n-1)+1$$
$$= 2\times(2\times N(k,n-2)+1)+1$$
$$= 2^2\times N(k,n-2)+2^1+2^0$$
$$= 2^2\times(2\times N(k,n-3)+1)+2^1+2^0$$
$$= 2^3\times N(k,n-3)+2^2+2^1+2^0$$
$$= \cdots$$
$$= 2^{n-k}\times N(k,k)+2^{n-k-1}+2^{n-2}+\cdots+2^1+2^0$$
$$= 2^{n-k+1}-1.$$

3.1.3　相对时间比较

相对时间比较见表 1.

表 1　新算法的时间效率

算法　盘数	$n=15$	$n=20$	$n=25$	$n=26$	$n=27$	$n=29$	$n=30$	$n=35$	$n=36$	$n=37$	$n=38$
digui	0.03	0.98	0.98	50.70	96.30						
LiuZhenHai	0.02	0.77	0.77	66.1	/	/	/	/	/	/	/
NiAiBing	0.09	3.33	3.33	146.00	346.00	/	/	/	/	/	/
LiZhong	0.047	1.44	45.48	90.92	181.28	759.16	1307.50				
XieXianFei	1.00	2.00	108.00	257.00	/	/	/	/	/	/	/
$K=3$	0.20	3.56	9.41	20.70	20.70	54.50	/	/	/	/	/
$K=9$	0	0	0	0.03	0.15	0.34	1.22	51.40	/	/	
$K=10$	0	0	0	0.08	0.20	1.70	2.77	4.78	63.70	115	/

注：空格表示尚未测试，"/" 表示不能计算；单位 /s

测试结果说明，新的算法的时间效率明显优于现有的递归和非递归算法.

致谢：感谢孙燮华教授的指导

【参　考　文　献】

[1]　李永新. 汉诺塔问题的非递归算法实现[J]. 湖州师范学院学报，2000(6)：43－47.

[2]　宁爱兵，黄　明. Hanoi 问题的非递归算法的形式推导[J]. 计算机工程与科学，2003(25)：66－68.

[3]　刘振海，束长宝. Hanoi 问题的一种非递归算法[J]. 电脑开发与应用，2002(11)：33－34.

[4]　李　忠，尹德辉，孟　林. 递归算法非递归化的一般规律[J]. 四川师范大学学报，2003(2)：209－212.

[5]　孙燮华. 计算机密码学的新进展[M]. 中国计量学院学报，2001(1)：1－18.