数理解析研究所講究録
第 322 巻 1978 年 218-229

A Note on Algorithms for Tower of Hanoi

by

Itsuo Takanami+

and

Katsushi Inoue+

+ Department of Electronics
Faculty of Engineering
Yamaguchi University
Ube, 755 Japan

ABSTRACT

For the case of the traditional three poles, we give a straight-line algorithm in the sense that it is not recursive. This algorithm only needs a constant memory not depending on the number of disks N. For the case of four or more poles, we propose a recursive procedure not using the dynamic programming technique. Then in a certain proposed algorithm we derive an explicit expression for the number of moves of disks as a function of N disks and m poles. In this algorithm the number of moves decreases monotoneously in terms of m but its limiting value is $3^{\lceil \log_2 N \rceil}$ although 2N+1 is the minimum number of moves for $m \geq N+1$. So we give a modified algorithm and its associated recurrence equation for the number of moves. This equation is solved numerically since it is difficult to derive the explicit expression for its solution. This result shows that the modified algorithm is near optimal.

1. Introduction

A study on programs or algorithms for the traditional Tower of Hanoi puzzle may be considered to become an appropriate object in the fields of the artificial intelligence and the complexity of algorithms. Some would say that this problem is traditional and already settled. However, as far as the authors know, little attention is paid to storage spaces and computation steps on executing this problem by computers. We focus attention on this point.

In Section 2, for the case of the traditional three poles, we give a straight-line algorithm in the sense that it is not recursive. This algorithm only needs a constant memory not depending on the number of disks N.

In Section 3, we discuss the case of four or more poles. This case is considered in [1] where the minimum number of moves of N disks for $N \leq 64$ is computed by the dynamic programming technique and the implicit expression for the number is given as an exterpolation of this result without proof. Furthermore, for the case of six or more poles, the implicit expression for the number of minimum moves is also given as a conjecture from the foregoing result. In [2], the minimum number of moves for the case of four poles is also computed by the dynamic programming technique. Since these studies depend on the dynamic programming technique, one might be afraid that trimendous memories and computation steps need in order to perform moves of disks by computer for a large number of disks. Hence, we propose a recursive program not using the dynamic programming technique. Then in a certain proposed algorithm we derive an explicit expression for the number of moves of disks as a function of N disks and m poles. In this algorithm the number of moves decreases monotoneously in terms of m but its limiting value is $3^{\lceil \log_2 N \rceil}$. However, for large N, $3^{\lceil \log_2 N \rceil}$ is much greater than the minimum number of moves 2N+1 for $m \geq N+1$. So we give a modified algorithm and its associated recurrence equation for the number of moves. This recurrence equation is solved numerically since it is difficult to derive the explicit expression for its solution. This result shows that the modified algorithm is near optimal.

2. Straight-line algorithm

The Tower of Hanoi puzzle: there are three poles P_1, P_2, P_3 with N different sized disks stacked on P_1. The disks are arranged in decreasing order with the largest one on the bottom and the smallest one on the top of the stack. Then move the disks one at a time from one pole to another, never putting a larger one on a smaller one, and eventually transferring the N disks from P_1 to P_3.

Let $\Pi_r(x,y)$ denote the moving sequence of r disks from P_x to P_y. Then a solution of the above problem is expressed by the following recursive equation

$$\Pi_N(1,3)=\Pi_{N-1}(1,2)\cdot\Pi_1(1,3)\cdot\Pi_{N-1}(2,3) \qquad (1)$$

If we accord meekly with the equation, the program will become recursive. If we can not use a recursive program, we probably ask for the case of N=1, and then for the case of N=2 and so on. Many memories and computation steps will need in either procedure.

Let $M(i)_k$ denote the total number of moves done until the disk D_i has been moved k times and let M(r) denote the length (the number of moves) of the sequence $\Pi_r(x,y)$ $(x \neq y)$. (Note that the length of the sequence $\Pi_r(x,y)$ $(x \neq y)$ does not depend on x and y). Then $M(i)_k$ can be described by $M(i)_{k-1}$ moves done until the (k-1)st move of D_i has been done, and M(i-1) moves to stack $D_1,\ldots,D_{i-1}$ on D_i, followed by one move of D_{i+1} and then M(i-1) moves to transfer $D_1,\ldots,D_{i-1}$ to some pole not having D_{i+1}, followed by one move to stack D_i on D_{i+1}. Therefore, we have the following equation

$$M(i)_k = M(i)_{k-1} + 2(M(i-1)+1)$$
$$= M(i)_{k-1} + 2 \qquad (2)$$

(It is well known that $M(i)=2^i-1$)

The equation is recursive and the following equation is easily derived.

$$M(i)_k = M(i)_1 + 2^i (k-1) \qquad (3)$$

On the other hand, since $M(i)_1 = M(i-1)+1 = 2^{i-1}$,

$$M(i)_k = 2^{i-1}(2k-1) \qquad (4)$$

Conversely, when an arbitrary positive integer t is given, we can uniquely determine the disk which is moved at the t-th move and the number of the time of the move in the successive moves of the disk. Let $D_{a(t)}$ be such the disk and let k(t) be such the number of the time. Of course, $t=2^{a(t)-1}(2k(t)-1)$. a(t) can be determined by $\log_2 t$ divisions and k(t) by one division. Later we discuss the determination of a(t).

Next, we need the information of from which pole to which pole the disk should be moved. The following property is known.

In Fig.1, let the counter clockwise move (from P_1 to P_2, from P_2 to P_3 or from P_3 to P_1) denote by +1 and let the

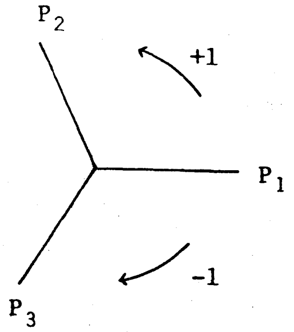

Fig.1

clockwise move denote by -1. Then for given number of disks N the move direction is determined for each disk. That is, that of D_i is $(-1)^{N+i+1}$. Therefore, the k-th move of each disk becomes as follows.

(i) If the move direction is +1, from $P_{(k-1) \bmod 3+1}$ to $P_{k \bmod 3+1}$

(ii) If the move direction is -1, from $P_{2(k-1) \bmod 3+1}$ to $P_{2k \bmod 3+1}$.

Summarizing the above result, we could give a straight-line algorithm. However, we discuss a little about deciding a(t) since it takes relatively long if it is done directly.

From $t=2^{a(t)-1}(2k(t)-1)$,

(i) if t is odd, $a(t)=1$ and $k(t)=(t+1)/2$, and

(ii) if t is even and

(a) if $a(t) \geq 3$, $a(t+2)=2$ since $t+2=2^{a(t)-1}(2k(t)-1)+2=2(2(2^{a(t)-3}(2k(t)-1)+1)-1)$,

(b) if $a(t)=2$, $a(t+2)=\alpha+3$ and $k(t+2)=\beta$ where $k(t)=2^{\alpha}(2\beta-1)$, since $t+2=2(2k(t)-1)+2=2^2k(t)=2^{\alpha+3-1}(2\beta-1)$.

Now, summarizing the above consideration, we give the following algorithm.

Algorithm 1. A straight-line algorithm for the Tower of Hanoi with three poles

Input. The number of disks N and the set of poles {1,2,3}.

Output. The sequence of moves of disks consisting of pairs of poles. (i,j) means 'from pole i to pole j.'

Method. The algorithm consists of a procedure call, HANOI(N), in which a procedure AK(t,a,k) is used to decide a(t) and k(t) for t.

```
procedure HANOI(N)
begin
     t ← 1;
     while t < 2^N do
          AK(t,a,k);
          i ← {((-1)^(a+N) +3)(k-1)/2} mod 3 + 1
          j ← {((-1)^(a+N) +3)k/2} mod 3 + 1
          print (i,j)
          t ← t + 1
end

procedure AK(t,a,k)
begin
     if t is odd then return a ← 1 and k ← (t+1)/2;
     else
        begin
            if t=2 then return a ← 2 and k ← 1:
            else
               if (t-2)/2 is odd then AK(t/4,p,q);
                                            return a ← p+2 and k ← q;
               else return a ← 2 and k ← (t+2)/4;
        end
end
```

3. Tower of Hanoi with four or more poles

We consider the general problem: Given m poles with N disks stacked in decreasing order of size on pole P_1. Move the N disks one at a time from one pole to another, never putting a larger one on a smaller one, and eventually transferring the N disks from P_1 to P_m , in steps as small as possible.

Let $\sigma(N,m)$ denote the minimum number of steps (moves of disks) Consider the following algorithm: First, take n_1 disks from the top on P_1 and by using m poles construct a tower consisting of them on some pole (denoted P_k) except P_m. Next, transfer the remaining $N-n_1$ disks on P_1 to P_m by using m-1 poles except P_k, and then the n_1 disks on P_k to P_m by using m poles, completing a final tower.

From the above algorithm, we have

$$2\sigma(n_1,m)+\sigma(N-n_1,m-1) \geqq \sigma(N,m).$$

Similarly, we have

$$2\sigma(n_2,m)+\sigma(n_1-n_2,m-1) \geqq \sigma(n_1,m).$$

In general, we have

$$2\sigma(n_i,m)+\sigma(n_{i-1}-n_i,m-1) \geqq \sigma(n_{i-1},m) \qquad i \geqq 1,\ n_o=N \tag{5}$$

Therefore, we have

$$2^i\sigma(n_i,m)+2^{i-1}\sigma(n_{i-1}-n_i,m-1) \geqq 2^{i-1}\sigma(n_{i-1},m) \qquad i \geqq 1,\ n_o=N \tag{6}$$

Summing up (6) from i=1 to q, we have

$$2^q\sigma(n_q,m)+\sum_{i=1}^{q} 2^{i-1}\sigma(n_{i-1}-n_i,m-1) \geqq \sigma(N,m) \tag{7}$$

If we put $n_{i-1}-n_i=d$ for $1 \leqq i \leqq q$, then $N-n_q=qd$. If $n_q=0$, that is, N is divided by q (or d), N=qd and

$$(2^q-1)\ \sigma(d,m-1)=(2^q-1)\ \sigma(N/q,m-1)$$
$$\geqq \sigma(N,m) \tag{8}$$

Using (8) succesively, we have

$(2^{q_1}-1)\ \sigma(N/q_1,m-1) \geqq \sigma(N,m)$ if N is divided by q_1.

$(2^{q_2}-1)\ \sigma(N/q_1 q_2,m-2) \geqq \sigma(N/q_1,m-1)$ if N is divided by $q_1 q_2$.

$\vdots$

$(2^{q_{m-4}}-1)\ \sigma(N/q_1\cdots q_{m-4},4) \geqq \sigma(N/q_1\cdots q_{m-5},5)$ if N is divided by $q_1\cdots q_{m-4}$.

$(2^{q_{m-3}}-1)\ \sigma(N/q_1\cdots q_{m-3},3) \geqq \sigma(N/q_1\cdots q_{m-4},4)$ if N is divided by $q_1\cdots q_{m-3}$.

Therefore, we have

$$(2^{q_1}-1)(2^{q_2}-1)\ldots(2^{q_{m-3}}-1)\ \sigma(N/q_1 q_2\cdots q_{m-3},3) \geqq \sigma(N,m)$$

if N is divided by $q_1\cdots q_{m-3}$.

Finally, we have

$$(2^{q_1}-1)(2^{q_2}-1)\ldots(2^{q_{m-3}}-1)(2^{N/q_1\cdots q_{m-3}}-1) \geqq \sigma(N,m) \tag{9}$$

if N is divided by $q_1\cdots q_{m-3}$.

If $N^{1/(m-2)}$ is an integer, we put $q_1=q_2=\ldots=q_{m-3}=N^{1/(m-2)}$.

Then (9) becomes

$$(2^{N^{1/(m-2)}}-1)^{m-2} \geqq \sigma(N,m) \tag{10}$$

In general, we have

$$\begin{aligned}(2^{\lceil N^{1/(m-2)}\rceil}-1)^{m-2} &\geqq \sigma(N,m) \quad \text{for } m-2 < \log_2 N\\ 3^{\lceil \log_2 N\rceil} &\geqq \sigma(N,m) \quad \text{for } m-2 \geqq \log_2 N\end{aligned} \tag{11}$$

where $\lceil x\rceil$ is the least integer equal to or more than x.

The following properties hold.

(1) $\lim_{m\to\infty} (2^{N^{1/(m-2)}}-1)^{m-2} = N^{2\log 2} \approx N^{1.38629}$

(2) $(2^{N^{1/(m-2)}}-1)^{m-2}$ decreases monotoneously in terms of m ($\geqq 3$).

It is easily shown that $\sigma(N,m)=2N-1$ for $m \geqq N+1$. Table 1 shows the comparison of $N^{2\log 2}$ with 2N-1.

From the property (2) and Table 1 it seems to be able to conclude the goodness of the above algorithm. But for large N $N^{2\log 2}$ becomes much larger than 2N-1. Hence, we introduce a modified algorithm in which the number of moves becomes 2N-1 for $m \geqq N+1$.

N	$N^{2\log 2}/(2N-1)$
10	1.28097
20	1.63133
50	2.28894
100	2.97668
200	3.88086
500	5.52074
1000	7.21217

Table 1. The comparison of $N^{2\log 2}$ with 2N-1

Algorithm 2. A recursive algorithm for the Tower of Hanoi with three or more poles, not using the dymanic programming technique.

Input. The number of disks N, the number of poles $m \geqq 3$ and the set of poles {1,2,...,m}.

Output. The sequence of moves of disks which are given by pairs of poles. (i,j) means 'from pole i to pole j'.

Method. The algorithm consists of a procedure call, HANOI(N,m), in which a procedure MOVE(n,m,i,j,S) is used. The procedure MOVE(n,m,i,j,S) gives the sequence of moves of n disks on the top of pole i to pole j, using none of the set S of poles.

```
procedure HANOI(N,m)
begin
    MOVE(N,m,1,m,Ø)
end
```

```
procedure MOVE(n,m,i,j,S)
begin
    if m-|S|=3 then return HANOI(n); where the pole numbers
          1, 2,and 3 in HANOI(n) is renamed by i, j,and k,
          respectively,(k∈{1,2,...,m}-{S∪{i,j}}), and |S|
          denotes the number of elements of S;
    else
       begin
          if n=1 then return print (i,j);
          else
             if n+1 ≤ m-|S| then return
                print (i,k); MOVE(n-1,m,i,j,S∪{k}); print (k,j);
                where k∈{1,2,...,m}-{S∪{i,j}};
             else
                p ← ⌊n(n^(-1/(n-1)) -n^(-1/(m-2-|S|)))⌋ +1;
                return MOVE(p,m,i,k,S);
                       MOVE(n-p,m,i,j,S∪{k});
                       MOVE(p,m,k,j,S);
                       where k∈{1,2,...,m}-{S∪{i,j}} and ⌊x⌋ is
                       the largest integer equal to or less
                       than x;
       end
end
```

Let $f(N,m)$ be the number of moves of disks in HANOI(N,m). Then we readily have the following equation.

$$f(N,3)=2^N-1,$$

for $m \geq 4$

$$f(N,m)=\begin{cases} 2N-1 & \text{for } N+1 \leq m \\ 2f(n,m)+f(N-n,m-1) & \text{for } N+1 > m \text{ where} \end{cases} \qquad (12)$$

$$n=\lfloor N(N^{-1/(N-1)}-N^{-1/(m-2)})\rfloor +1$$

The values of f(N,m) derived by numerically solving the recurrence equation (12) are shown in Table 2 in which the values in parenthesis are given or conjectured in [1]. Table 2 shows that our result is not optimal but near optimal for m=4 and 5. We will consider that our result is also near optimal for all m 6.

4. Conclusion

Analyzing the traditional Tower of Hanoi puzzle, we have given a straight-line algorithm which is not recursive. This has an advantage of only needing a constant memory. Furthermore, we have investigated the Tower of Hanoi with four or more poles and have given a near optimal recursive algorithm not using the dynamic programming technique.

References

1. Brousseau,B.A. Tower of HANOI with more pegs, J.Recreational Mathematics Vol.8(3). (1976) 169-176.

2. Nakamura,G. Puzzle and combinatorial theory, Mathematical Sciences in Japanese, No.115 (Jan. 1973) 5-11.

3. Imamiya,A. and Yuri,H. On realization for permutations through 3-stacks, Technical Report EC 74-24 of IECE of Japan (Sept.1974)

4. Aho,A.V., Hopcroft,J.E. and Ullman,J.D. The desing and analysis of computer algorithms, Addison-Wesley Pub. Co. (1974).

N \ m	4	5	6	7	10	20
2	3 (3)	3 (3)	3 (3)	3 (3)	3 (3)	3 (3)
3	5 (5)	5 (5)	5 (5)	5 (5)	5 (5)	5 (5)
4	9 (9)	7 (7)	7 (7)	7 (7)	7 (7)	7 (7)
5	13 (13)	11 (11)	9 (9)	9 (9)	9 (9)	9 (9)
10	57 (49)	35 (31)	29 (29)	27 (27)	21 (21)	19 (19)
20	353 (289)	127 (111)	89 (89)	75 (67)	61 (61)	41 (41)
30	1153 (1025)	303 (271)	185 (169)	143 (143)	101 (101)	81 (81)
40	2945 (2817)	559 (511)	313 (289)	231 (223)	157 (141)	121 (121)
50	6913 (6657)	943 (831)	473 (449)	343 (303)	225 (201)	161 (161)
60	15361 (14337)	1471 (1279)	697 (629)	471 (415)	297 (281)	201 (201)
100	176129 (172033)	5855 (4863)	2017 (1729)	1215 (1055)	637 (601)	361 (361)
200	15204353 (14680065)	53887 (36863)	10257 (7297)	4863 (3839)	1953 (1681)	993
300	478150657 (385875969)	251903 (143359)	30305 (19457)	11647 (8575)	3897 (3281)	1741
500		2478079 (1015807)	131905 (68097)	38095 (23807)	9425 (6561)	3441

Table 2. The values of f(N,m) (The values in parenthesis are given or conjectured in [1])