An Algorithm for Tower of Hanoi with Four or More Poles

By Itsuo TAKANAMI* and Katsushi INOUE*

(Received July 6, 1978)

Abstract

For the case of four or more poles, we propose a recursive procedure not using the dynamic programmming 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 $m \geqq N+1$. 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 next section, 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 $N \leqq 64$ 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 tremendous 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 monotonously 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 $m \geqq N+1$. 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

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

expression for its solution. This result shows that the modified algorithm is near optimal.

2. 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

$$2\sigma(n_i, m)+\sigma(n_{i-1}-n_i, m-1)\geqq\sigma(n_{i-1}, m) \qquad i\geqq 1, \quad n_0=N \tag{1}$$

Therefore, we have

$$2^i\sigma(n_i, m)+2^{i-1}\sigma(n_{i-1}-n_i, m-1)\geqq 2^{i-1}\sigma(n_{i-1}, m) \qquad i\geqq 1, \quad n_0=N \tag{2}$$

Summing up (2) from $i=1$ to q, we have

$$2^q\sigma(n_q, m)+\sum_{i=1}^{q} 2^{i-1}\sigma(n_{i-1}-n_i, m-1)\geqq\sigma(N, m) \tag{3}$$

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

$$(2^q-1)\sigma(d, m-1)=(2^q-1)\sigma(N/q, m-1)\geqq\sigma(N, m) \tag{4}$$

Using (4) successively, we have

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

$(2^{q_2}-1)\sigma(N/q_1q_2, m-2)\geqq\sigma(N/q_1, m-1)$ if N is divided by q_1q_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

$$(2^{q_1}-1)(2^{q_2}-1)\cdots(2^{q_{m-3}}-1)\sigma(N/q_1q_2\cdots q_{m-3},\ 3)\geqq\sigma(N,\ m)$$

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

Finally, we have

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

if N is divided by $q_1\cdots q_{m-3}$. (5)

If $N^{1/(m-2)}$ is an integer, we put $q_1=q_2=\cdots=q_{m-3}=N^{1/(m-2)}$. Then (5) becomes

$$(2^{N^{1/(m-2)}}-1)^{m-2}\geqq\sigma(N,\ m) \tag{6}$$

In general, we have

$$\begin{aligned}(2^{\lceil N^{1/(m-2)}\rceil}-1)^{m-2}&\geqq\sigma(N,\ m) \quad \text{for} \quad m-2<\log_2 N\\ 3^{\lceil\log_2 N\rceil}&\geqq\sigma(N,\ m) \quad \text{for} \quad m-2\geqq\log_2 N\end{aligned} \tag{7}$$

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 monotonously in terms of m ($\geqq 3$).

The proof is in the Appendix.

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 we want to make the number of moves become $2N-1$ for $m\geqq N+1$. From (1) to (5), $n_1=N-N^{1-1/(m-2)}=N(1-N^{-1/(m-2)})$. We will modify n_1 so that $n_1=1$ for $m=N+1$. To do so, we choose $n_1=\lfloor N(1-N^{-1/(m-2)}+N^{-1/(N-1)}-1)\rfloor+1=\lfloor N(N^{-1/(N-1)}-N^{-1/(m-2)})\rfloor+1$. Now, we introduce a modified algorithm in which the number of moves becomes $2N-1$ for $m\geqq N+1$.

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

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

Algorithm. A recursive algorithm for the Tower of Hanoi with three or more poles,

not using the dynamic 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 no poles of the set S.

```
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

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
```

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 $m \geqq 4$

$$f(N, m)=\begin{cases} 2N-1 & \text{for } N+1 \leqq m \\ 2f(n, m)+f(N-n, m-1) & \text{for } N+1>m \text{ where} \\ & n=\lfloor N(N^{-1/(N-1)}-N^{-1/(m-2)})\rfloor+1 \end{cases} \tag{8}$$

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

N \ m	4	5	6
2	3 (3)	3 (3)	3 (3)
3	5 (5)	5 (5)	5 (5)
4	9 (9)	7 (7)	7 (7)
5	13 (13)	11 (11)	9 (9)
10	57 (49)	35 (31)	29 (29)
20	353 (289)	127 (111)	89 (89)
30	1153 (1025)	303 (271)	185 (169)
40	2945 (2817)	559 (511)	313 (289)
50	6913 (6657)	943 (831)	473 (449)
60	15361 (14337)	1471 (1279)	697 (629)
100	176129 (172033)	5855 (4863)	2017 (1729)
200	15204353 (14680065)	53887 (36863)	10257 (7297)
300	478150657 (385875969)	251903 (143359)	30305 (19457)
500		2478079 (1015807)	131905 (68097)

7	10	20
3 (3)	3 (3)	3 (3)
5 (5)	5 (5)	5 (5)
7 (7)	7 (7)	7 (7)
9 (9)	9 (9)	9 (9)
27 (27)	21 (21)	19 (19)
75 (67)	61 (61)	41 (41)
143 (143)	101 (101)	81 (81)
231 (223)	157 (141)	121 (121)
343 (303)	225 (201)	161 (161)
471 (415)	297 (281)	201 (201)
1215 (1055)	637 (601)	361 (361)
4863 (3839)	1953 (1681)	993 (801)
11647 (8575)	3897 (3281)	1741 (1601)
38095 (23807)	9425 (6561)	3441 (3201)

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 $m \geqq 6$ if the Brousseau's conjecture is true.

3. Conclusion

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 design and analysis of computer algorithms, Addison-Wesley Pub. Co. (1974).

Appendix

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

Proof. Put $m-2=m'$ and $N^{1/m'}=x$. Then $m'=\log_2 N/\log_2 x$ and $x\to 1$ ($m'\to\infty$). Therefore,

$$(2^{N^{1/m'}}-1)^{m'}=((2^x-1)^{1/\log_2 x})^{\log_2 N}$$

Put $\log_2 x=X$, i.e. $x=2^X$. Then $X\to 0$ ($x\to 1$). Therefore,

$$(2^x-1)^{1/\log_2 x}=(2^{2^X}-1)^{1/X}\equiv f(X)$$

$$\log f(X)=\log(2^{2^X}-1)/X \tag{9}$$

$$\lim_{X\to 0}\log f(X)=\lim_{X\to 0}2^{2^X}\log 2\cdot 2^X\log 2/(2^{2^X}-1)=2\cdot(\log 2)^2$$

Thus

$$\lim_{X\to 0}f(X)=\exp(2(\log 2)^2)$$ Q. E. D.

Therefore, we have

$$\lim_{m\to\infty}(2^{N^{1/(m-2)}}-1)^{m-2}=\exp(2(\log 2)^2\log_2 N)=N^{2\log 2}$$

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

Proof. To show that the statement is true, it is sufficient to show that $\log(2^{2^X}-1)/X$ in equation (9) increases monotonously in terms of X for $X>0$.

Let put

$$F_1(x)\equiv\log(2^{2^x}-1)/x$$

We will show that $dF_1(x)/dx\geqq 0$ for $x>0$.

$$dF_1(x)/dx=[x2^{2^x}2^x(\log 2)^2-(2^{2^x}-1)\log(2^{2^x}-1)]/[x^2(2^{2^x}-1)] \tag{10}$$

Since the dominator of (10) is positive for $x>0$, it suffices to show that the denominator

of (10) is nonnegative. Let it be $F_2(x)$.

$$dF_2(x)/dx = 2^{2^x}2^x(\log 2)^2[x2^x(\log 2)^2 + x\log 2 - \log(2^{2^x}-1)]$$

$$F_3(x) \equiv x2^x(\log 2)^2 + x\log 2 - \log(2^{2^x}-1)$$

$$dF_3(x)/dx = \{(2^{2^x}-1)x2^x(\log 2)^3 + [(2^{2^x}-1) - 2^x\log 2]\log 2\}/(2^{2^x}-1)$$

$$F_4(x) \equiv (2^{2^x}-1) - 2^x\log 2$$

$$= 2^{2^x} - 1 - \log 2^{2^x}$$

$$dF_4(x)/d2^{2^x} = 1 - 2^{-2^x} > 0 \qquad (x>0)$$

$$F_4(0) = 1 - \log 2 > 0$$

Therefore, $F_4(x)>0$ $(x>0)$. Then $dF_3(x)/dx>0$ $(x>0)$. Since $F_3(0)=0$, $F_3(x)>0$ $(x>0)$. Then $dF_2(x)/dx>0$. Since $F_2(0)=0$, $F_2(x)>0$ $(x>0)$. Q. E. D.