A Non-recursive Algorithm for 4-Peg Hanoi Tower

Jun Wang[1] Junpeng Liu[2] Guoying Yue[1] Liangshan Shao[2] Sukui Lu[3]

[1]Zhejiang Water Conservancy and Hydropower College, Hangzhou 310018, P. R. China

[2]Liaoning Technical University, Fuxin 123000, P. R. China

[3]Hebei University, Baoding 071102, P. R. China

Abstract

The compound relationship between different moving-orders and different plates of 4-peg Hanoi Tower problem is analyzed in this paper, which is regarded as a new cross correlation pattern without fixed span, not the abutting correlation pattern of 3-peg problem. It is rearranged to a binary and hierarchical tree good for understanding and using. A new idea to solve the 4-peg with a non-recursive algorithm is proposed by forming an iteration path with those nodes that are related to the target problem directly and iterating according this path. By abandoning the storage and computing of useless node, the efficiency of the non-recursive algorithm is improved great.

Keywords*:* 4-peg Hanoi Tower, Cross correlation, Iteration path

1. Introduction

Hanoi Tower is an old mathematic problem, which is firstly proposed in 1883 by Edouard Lucas, a France mathematician. The algorithm to solve this problem has been an essential model in computer fundamental courses such as data structure, algorithm analysis and so on. It have 3 pegs and there are n plates placed in a small-to-big order on one peg which may call A. It is required to move those all plates from A to anther peg, which may call C, taking advantage of one pig which may call B, at such a regulation that no big is plated on a small plate at any time. Such a problem has been solved well, including recursive [6] and non-recursive [3]-[5] solutions. The model of recursive algorithm of traditional Hanoi Tower is as follows:

$$F^3(n,A,B,C)=\begin{cases}F^3(n-1,A,C,B)\\ A\rightarrow C\\ F^3(n-1,B,A,C)\end{cases} \quad (1)$$

Where n is the number of plates, A, B and C are the 3 pegs and A→C means to move one plate from A to C. It can be inferred that the moving order of n plates from A to C can be formed by the moving order of n-1 plates which means that if the moving order of n-1 plates are given, the moving order of n plates can be available by replacing some alphabet of moving order of n-1 plates on some principle. Therefore, non-recursive algorithm of Hanoi Tower with 3 pegs takes good advantage of such feature and loops from 1 to n and replaces one by one to get the moving order of n plates.

The 4-peg problem is just to insert one peg into the traditional model and at the same regulation which looks like figure 1. One more peg means by far less moving steps from initial peg to target peg, but by far more complex meanwhile. 3-peg problem has unique moving order while 4-peg problem has not because one more buffer peg means optional moving selection and therefore different moving orders. But they are equivalent with just different roles acting by different peg. J.S.Frame propose one algorithm to solve this problem in *American Mathematic Monthly* in 1941, and in 2004, the algorithm is proved by YangKai and XuChuan using mathematical induction in *Acta Scientiarum Naturalium Universitatis Pekinensis* [1]. And afterward LiuDuo and DaiYiqi provide mathematic prove of the minimum moving steps about even more general problem with multi-pegs [2].

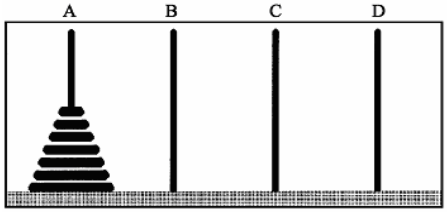

Fig. 1: The illustration of 4-peg Hanoi Tower.

2. Formula and Related Data

There have different moving orders for a 4-peg

problem with n plates but they are equivalent. Therefore, we could list a formula to express the problem with n plates as follows [1]:

$$F^4(n,A,B,C,D)=\begin{cases}F^4(n-R(n),A,C,D,B)\\ F^3(R(n),A,C,D)\\ F^4(n-R(n),B,A,C,D)\end{cases} \quad (2)$$

If we want to solve the problem with n plates, firstly we should move the upper n-R (n) plates from A to B with the buffer of C and D, then we move the rest plates from A to D with C as a buffer, and last we move the n-R (n) plates from B to D with the buffer of A and C. R(n) is the core of the whole problem and the moving step is minimum [1] when the following formula is right that

$$R(n)=\left\lfloor \frac{\sqrt{8*n+1}-1}{2} \right\rfloor .$$

Then minimum step is

$$[n-\frac{R^2(n)-R(n)+2}{2}]*2^{R(n)}+1$$

That result has been mentioned in reference 2. Some data of 4-peg problem are list in table 1 as follows:

n	F(n)	R(n)	MS	CI
1	F(1,A,B,C,D)	1	1	F(1, A,B,C,D)=AD
2	F(2,A,B,C,D)	1	3	F(1,A,C,D,B),f(1,A,C,D), F(1,B,A,C,D)
3	F(3,A,B,C,D)	2	5	F(1,A,C,D,B), f(2,A,C,D), F(1,B,A,C,D)
4	F(4,A,B,C,D)	2	9	F(2,A,C,D,B), f(2,A,C,D), F(2,B,A,C,D)
5	F(5,A,B,C,D)	2	13	F(3,A,C,D,B), f(2,A,C,D), F(3,B,A,C,D)
6	F(6,A,B,C,D)	3	17	F(3,A,C,D,B), f(3,A,C,D), F(3,B,A,C,D)
7	F(7,A,B,C,D)	3	25	F(4,A,C,D,B), f(3,A,C,D), F(4,B,A,C,D)
8	F(8,A,B,C,D)	3	33	F(5,A,C,D,B), f(3,A,C,D), F(5,B,A,C,D)
9	F(9,A,B,C,D)	3	41	F(6,A,C,D,B), f(3,A,C,D), F(6,B,A,C,D)
…	…	…	…	…

Table 1: Sample Data of 4-peg Hanoi Tower.
MS is short for Moving Steps.
CI is short for Correlation Items.

3. Non-recursive Solution

As we all know, the moving orders of those that have different initial, path and target, could be transformed from one to another just by replacing some alphabets according to some rule, which has been proved enough in traditional 3-peg problem. Therefore, we could regard those problems with same plates but different origins and targets as one type of problem, called Fn in order to simplify our problem. Similarly, we could use fn to express those problems of 3-peg with n plates. After table 1 could be simplified to table 2 as follows:

Target	Correlation	Target	Correlation	Target	Correlation
F1	f1	**F8**	F5,f3	**F15**	F10,f5
F2	F1,f1	**F9**	F6,f3	**F16**	F11,f5
F3	F1,f2	**F10**	F6,f4	**F17**	F12,f5
F4	F2,f2	**F11**	F7,f4	**F18**	F13,f5
F5	F3,f2	**F12**	F8,f4	**F19**	F14,f5
F6	F3,f3	**F13**	F9,f9	**F20**	F15,f5
F7	F4,f3	**F14**	F10,f4	**F21**	F15,f6

Table 2: Correlation of Fn.

In fact, all the problems of 4-peg could be transformed into 3-peg problems, but we couldn't get the direct formula because there have many relations among 4-peg problems. So, the type of 4-peg problems is far more difficult and complex than that of 3-peg problems. In view of good solution of 3-peg problems, we may take them in constant and now we get a correlation table where Fn is the only variable. It's an iteration relationship in some form.

If R(n) is a parameter to 3-peg problems, R(n) is 1 all the time whatever n values, which means a direct iteration relationship between problems with n plates and those with n-1 plates. We may call it adjacent correlation pattern. However, R(n)>1 and varies with n in 4-pet problems, which means that Fn is not adjacent to F(n-1) but F(n-R(n)) in the solving process of 4-peg problems to work out the certain moving order. We may call it a cross correlation pattern.

We take good advantage of the idea to solve 3-peg problems to solve the 4-peg problems still and how to

make sure of and deal with R(n) are the key steps of the whole algorithm. There are two possible ways in detail as follows:

One is that forming an array of moving order string useful to future calculation to save the middle results of iteration. It is inferred in the formula of F(n)=F(n-R(n))+f(R(n))+F(n-R(n)) that results among one problem with n-R(n)+1 plates to that with n plates should be prestored in the loop process because those results must be used in the succedent iteration. And therefore, an string array with at least R(n) elements is needed.

In the iteration process, in the first time we may work out that F1 is A→D and we should save it because it's useful to F2 and F3; in the second time we work out F2 and F3 with help of F1 and we must save F2 and F3 and may abandon F1 because F2 is useful to F4, F3 is useful to F5 and F6 and F1 is no longer useful; and similarly we must save F4, F5 and F6 and abandon F2 and F3. The rest may be deduced by analogy. The cost of storage would become huge and it is a difficult and time consuming job to decide which should be stored and deleted and when should move elements of the array, when we met with a little more complex problem simply with much more plates. Obviously it is not a efficient and feasible solution.

The other is just to try to make sure of correlation items with the target problem and therefore transform the cross correlation pattern to adjacent correlation pattern.

In the view of cross correlation of iteration process, and according to the formula ①, table 1 and table 2, draw an arrow from the correlated node to the target node and if we draw out all possible arrows, we get a graphic as Fig 2 shows.

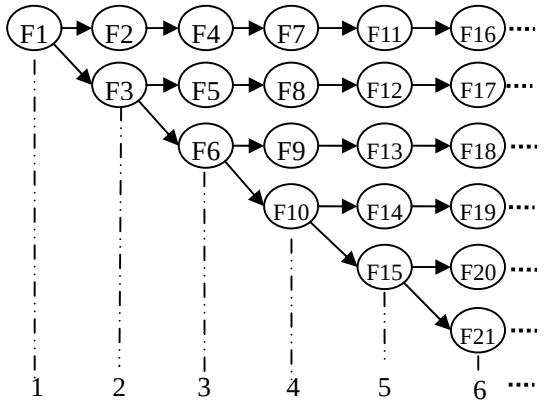

Fig. 2: Correlation Tree of Fn.

Nodes at each end of one arrow have a direct correlation, which means that the moving orders of the node that the arrow points could be iterated directly from the moving order of the node that the arrow starts. We can conclude from the Fig2 that Fn is a traditional binary tree and there is only one margin between the number of nodes in adjacent layer.

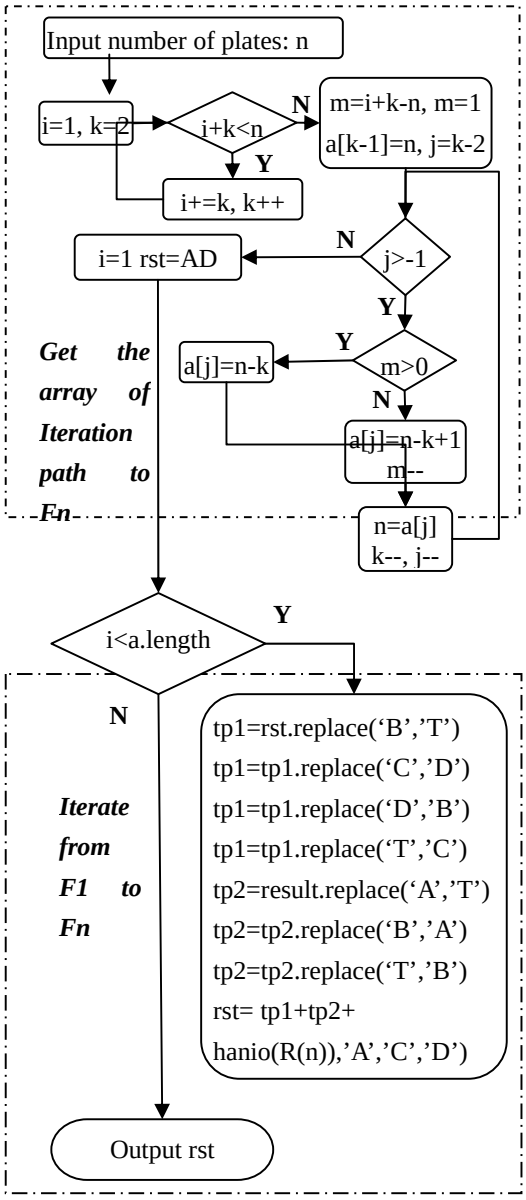

Fig. 3: The Flow Chat of Nun-recursive Algorithm to 4-peg Problems.

If we want to work out the moving order of F19, we should take good advantage of the iteration path: 1, 3, 6, 10, 14, 19, which may be easily figure out from Fig2. Therefore, we could use F1 to iterate F3 and then use F3 to iterate F6 and so on and so forth and at last

we can work out F19 by only 5 iterations.

So, in this paper, our idea is that firstly try to find out the iteration path from F1 to Fn and store the mark number into an array, and then loop to iterate from the first element of the array to the last and the moving order of Fn is just the result. The key step of this algorithm is to find the iteration path of target problem. We know from Fig.2 that the correlation of Fn is an upper triangular matrix and we can make sure of the exact position of one node by the character of the upper triangular matrix and construction rule of nodes.

The flow chat of solving 4-peg problems with nun-recursive algorithm is shown in Fig3.Fulfill the algorithm in Java and we get our prospective results same to reference 1 and 2 exactly.

We plot the time consuming to calculate and the number of disks in one figure so as to compare the efficiency of the Non-recursive algorithm, NA for short, to the traditional algorithm, TA for short. In view of the strong power of current computer, which makes such problems with few disks solve in far less than 1ms, we regard the total time of 1000 running times for the same problem as comparison criteria. Take time as vertical axis and number of disks as horizontal axis, and we get a time-consuming plotting as Fig.4 shows.

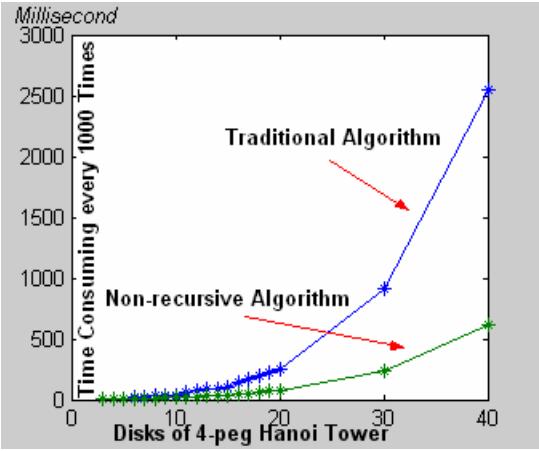

Fig. 4: Contrast Plotting of TA and NA.

As we may see easily from the contrast plotting that the non-recursive algorithm saves running time great much and the more complicated the problem is and the more running time it saves. It is because the non-recursive algorithm works follow the iteration path, omitting such operations to middle nodes with no influence to the result.

4. Conclusions

In this paper, one new method is proposed to solve 4-peg Hanoi Tower problems in recursive algorithm which takes good advantage of special features of 4-peg problems. It requires firstly find the iteration path. And then loop to iterate in a cross iteration way according to the iteration path to work out the moving order with n plates. And in this way, we needn't iterate from 1 to n for the cross iteration and we only need to iterate n-R(n) times at most, which is much less than n and therefore much less storage and calculating of useless nodes.

Acknowledgement

This work is partially supported by National Nature Science Foundation of China (Grant No. 70572070), Zhejiang Provincial Natural Science Foundation (Grant No. 2006C33057) and Hebei Provincial Natural Science Foundation (Grant No. 072135115).

References

[1] K. Yang and C. Xu, The Preliminary Probe of 4-Peg Hanoi Tower, *Acta Scientiarum Naturalium Universitatis Pekinensis(Natural Science Edition)*, 1:99-106, 2004.

[2] D. Liu and Y.Q. Dai, Study of Hanoi Tower Problem with Multi Pegs, *Acta Scientiarum Naturalium Universitatis Pekinensis(Natural Science Edition)*, 1:99-102, 2006.

[3] S.L. Tao, A Profound Inquiry Into The Hanoi Problem, *Journal of Panzhihua University*（*Natural Science Edition*）, 3:34-41, 1994.

[4] R.L. Shi, A Fast Iterative Algorithm for Solving the Towers of Hanoi Problem, *Journal of ChangSha Railway University*, 9:39-43, 1995.

[5] N. Qiu, A Non recursive Algorithm of Hanoi Tower, *Journal of ZheJiang ShuRen University*. 3:117-118, 2005.

[6] Z.C. Li and J. Su, P.D. Java, *China Railway Publishing House*, BeiJing, 51-52, 2004..