World Academy of Science, Engineering and Technology
Vol:5 2011-01-24

On the Solution of the Towers of Hanoi Problem

Hayedeh Ahrabian, Comfar Badamchi, and Abbass Nowzari-Dalini

Abstract—In this paper, two versions of an iterative loopless algorithm for the classical towers of Hanoi problem with $O(1)$ storage complexity and $O(2^n)$ time complexity are presented. Based on this algorithm the number of different moves in each of pegs with its direction is formulated.

Keywords—Loopless algorithm, Binary tree, Towers of Hanoi.

I. Introduction

THE towers of Hanoi problem is an old puzzle concerned with moving n disks of decreasing diameter all initially stacked on one peg to another peg in a minimal number of moves. Disks must be moved one at a time, only the topmost disk on each peg can be moved, and a larger disk may never be placed on top of smaller one. A third spare peg is available for the intermediate placement of the disks. The three pegs are numbered 1, 2 and 3. It is assumed that the starting peg is 1, the goal peg is 3 and the spare peg is 2. Disks are numbered consecutively from the smallest to the largest from 1 to n. Initially they are all stacked on peg 1, disk n stands at its bottom and disk 1 on the top.

Since the publication of a recursive solution to this problem [5], many attempts have been made by various authors to find simple and efficient iterative solutions [2], [6], [13], [17], [23], [25]. The main idea of the all the iterative algorithms of the towers of Hanoi is based on the disk transfers which can occur in cyclically moves in two directions. For an even n the direction of disk transfers with odd-numbers is $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$, and for even-numbered disks is $1 \rightarrow 3 \rightarrow 2 \rightarrow 1$. Reversely for an odd n the directions are reversed: For even-numbered disks is $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$, and for odd-numbered disks is $1 \rightarrow 3 \rightarrow 2 \rightarrow 1$.

Er [9] presented a loopless algorithm that moves a disk in a constant time independent of the number of disks. When a disk is moved, the least significant 1 at position i in the binary number x (which shows the step number) is changed to 0, and all 0's in between positions 1 and $(i-1)$ inclusively are set to 1's. To model this fact, an auxiliary array is used such that the contents of this array indicates the disk number of a disk to move in each step. The computation of each element of this array in each step depends to the previous step. The time complexity of this algorithm is $O(2^n)$ and the space complexity of the algorithm is $O(n)$. It should be noted that this algorithm is performed independent of the simulation of towers by arrays.

Then, Atkinson [1] and Gedeon [11] presented cyclic version of tower of Hanoi, and suggested algorithms for solving the problem. Also, Sniedovich [20] reviewed the tower of Hanoi problem from an Operational Research perspective. He showed that this problem provides an excellent environment for illustrating a number of fundamental Operational Research problem solving concepts in general and dynamic programming concepts in particular.

The towers of Hanoi problem was extended to four pegs by Dudeney [4] in 1907 and to any arbitrary $k \geq 3$ pegs by Stewart [21] in 1939. In 1941, Frame [10] and Stewart [22] independently proposed an algorithm to the towers of Hanoi problem with $k \geq 4$ pegs. Solutions are also given for the extended problem [12], [26], [24]. Also, for obtaining the minimum number of disk moves, solutions are discussed in [14]. In addition a conjecture with this respect is suggested in [15], and Chen and Shen presented an order for this conjecture [3].

In this paper two versions of an iterative algorithm for the solution of towers of Hanoi is presented. For designing this algorithm the ideas in the Er [7], [8], [9], Mayer and Perkins [16] and Meyer [18] are combined. The solution is based on the binary tree to which the recursive procedure is associated. First an iterative version of the algorithm with an inner loop is presented, and later a loopless version of the algorithm is given. Technically, the storage complexity of this algorithm is $O(1)$ and each step of this algorithm is independently computed and does not depend to the former computed numbers. It is proved that the time complexity of the presented algorithm is $O(2^n)$. Consequently, according to the presented binary tree the number of each move with different directions is formulated.

II. Iterative Algorithm

A full binary tree with $2^n - 1$ nodes can be associated to the solution of the towers of Hanoi with n disks [16], [18]. Each node of this tree can be labeled and each label presents a move. There are six different possible moves and each move is labeled. The corresponding labels are shown in Table I. Without the execution of any algorithm the corresponding full binary tree which its nodes are labeled, can be constructed. As it is illustrated in Fig. 1, the moves in each level has a special pattern. For the odd levels $zy'x'$ are repeated and for even levels xyz' are repeated. The inorder traversal of this binary tree solves the problem. We associate a binary number defined

TABLE I
The six move and corresponding labels.

from 1 to 2	x	from 2 to 1	x'
from 2 to 3	y	from 3 to 2	y'
from 1 to 3	z	from 3 to 1	z'

H. Ahrabian, C. Badamchi, and A. Nowzari-Dalini are with Center of excellence in Biomathematics, School of Mathematics, Statistics and Computer Science, University of Tehran, Tehran, Iran. e-mail: {ahrabian, badamchi, nowzari}@ut.ac.ir

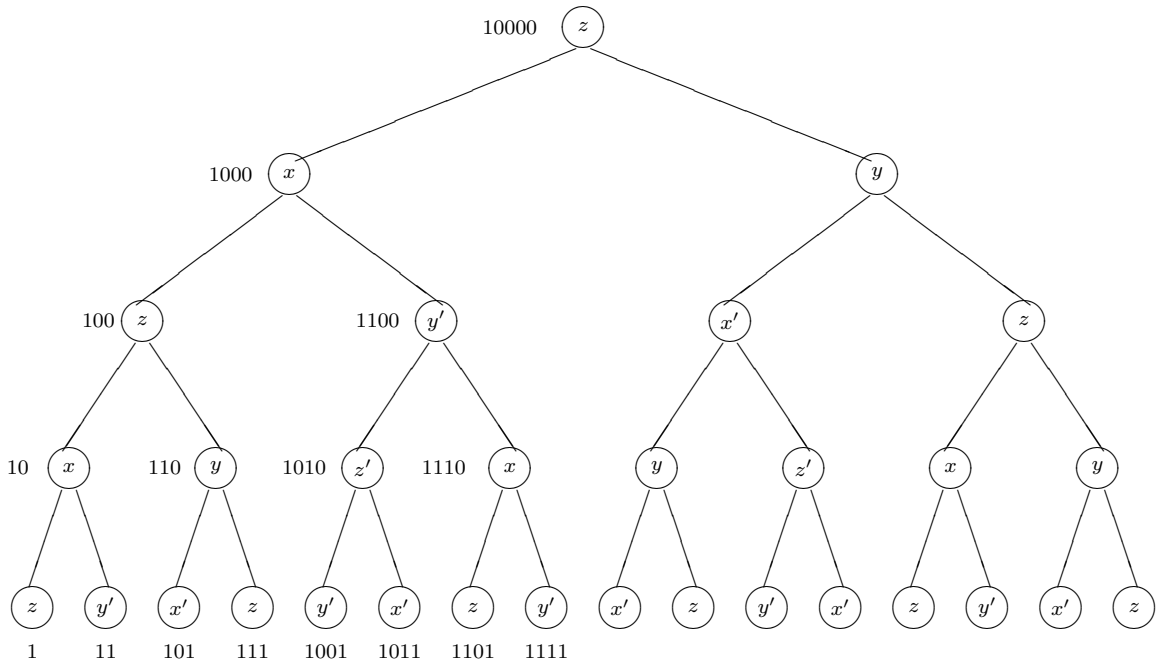

Fig. 1. The binary tree associated to the towers of Hanoi solution for $n = 5$.

by j to each node of the binary tree, which this number is the same as the node number in an inorder traversal of the tree. Easily the disk number and the label of the corresponding node which shows the different moves can be computed from j.

As it is obvious, the inorder numbering of the nodes starts from the last level (nth level). Therefore, if n is odd the pattern of the odd's level would be $zy'x'$ and for an even n the pattern is xyz' and this pattern alters every other one. The algorithm should be designed to choose the correct pattern and then the number j is converted to 0, 1 and 2, such that in a odd pattern 0 stands for z, 1 for y' and 2 for x', and in a even pattern 0 stands for x, 1 for y and 2 for z'. For realizing the level number of j, the maximum i which $2^i \mid j$ (2^i is a divisor of j) is computed. Then $n - i$ shows the level number. It can be easily noticed that for the odd numbers, i is equal to 0 and therefore all the odd numbers will appear in the nth level. After computing the value of i, the variable j is shifted $i + 1$ times and the result is divided by 3. The reminder will be 0, 1 and 2 which is mentioned in above. The computations for i can be performed either with loop or loopless. The algorithm illustrated in Algorithm 1 presents this computation with an inner loop. It is proved that the time complexity of the algorithm is $O(2^n)$.

Algorithm 1 The iterative algorithm.

```
void hanoi-1(int n)
{
  int i, j, k, l, q ;
  int move=(1 << n) - 1 ;
  unsigned char pegs[6]={ 1, 2, 3, 1, 3, 2 } ;
  for (j=1; j <=move; j++) {
    i=0 ;
    while ( !(j >> i & 1) )
      i++ ;
    l=(n - i) & 1 ;
    k=(j >> ++i) % 3 ;
    k=pegs[k + (l * 3)] ;
    q=(k + l) % 3 + 1 ;
    printf ("move disk %d from %d to %d", i, k, q) ;
  }
}
```

Theorem 1. *The time complexity of the algorithm hanoi-1 is* $O(2^n)$.

Proof As it is mentioned earlier, the algorithm traverses all the nodes of the full binary tree in an inorder traversal. The main loop of the algorithm is repeated for 2^n times, and j is an index variable in the main loop. For each odd j the internal loop is not performed. According to the full binary tree, the number of odd js is equal to 2^{n-1}. For any even j the internal loop is repeated i times, where i is the maximum value such that $2^i \mid j$. Obviously the number of such js is equal to 2^{n-i-1} which is equal to the number of all nodes in the $(n-i)$th level of the corresponding full binary tree. Assume $T(n)$ be the time complexity of the algorithm, therefore we can write:

$$T(n) = 1.2^{n-1} + 1.2^{n-2} + 2.2^{n-3} + \cdots + (n-1).2^0,$$
$$T(n) = 2^{n-1} + 2^{n-1}(\tfrac{1}{2} + \tfrac{2}{2^2} + \cdots + \tfrac{n-1}{2^{n-1}}),$$

since we have:

$$\sum_{i=1}^{\infty} \frac{i}{2^i} = O(2),$$

therefore:

$$T(n) = 2^{n-1} + O(2^n),$$
$$T(n) = O(2^n).$$

□

As mentioned before, for computing the level number $n-i$, first i should be computed. We can also compute i without any loop. With small modification in the algorithm the inner loop in the main loop is eliminated. In jth step of the algorithm the level number $n-i$ can be easily obtained by the following formula: $\mathbf{log_2}\ (j\ \&\ !\ (j-1))$. The algorithm illustrated in Algorithm 2 shows these computations. It is a loopless version of the previous algorithm. As it is obvious from the algorithm, the time complexity of the algorithm depends on the single loop of the algorithm which shows the number of steps $O(2^n)$. According to both algorithms, no auxiliary array of size n is used and all the computations can be performed with constant storage, thus without considering the allocation space of index j (move counter), the space complexity of the algorithms is $O(1)$.

III. Counting the number of different moves

Since the full binary tree representing the performance of the algorithm is fixed and with increasing of n by 1, just one level is added to the binary tree, therefore the number of six moves z, y', x', x, y and z' can be counted. This can be done by counting the number of nodes in each level of the corresponding binary tree. Therefore, if count(x) shows the number of x moves, we have:

$$count(z) = \sum_{k=0}^{\lceil n/2\rceil-1} \lceil\frac{2^{2k}}{3}\rceil,$$
$$count(y') = \sum_{k=0}^{\lceil n/2\rceil-1} \lceil\frac{2^{2k}-1}{3}\rceil,$$
$$count(x') = \sum_{k=0}^{\lceil n/2\rceil-1} \lceil\frac{2^{2k}-2}{3}\rceil,$$

Algorithm 2 The loopless version algorithm.

```
void hanoi-2(int n)
{
  int i, j, k, l, q ;
  int move=(1 << n) - 1 ;
  unsigned char pegs[6]={ 1, 2, 3, 1, 3, 2 } ;
  for (j=1; j <=move; j++) {
     i= log2 ( j & ! (j − 1)) ;
     l=(n − i) & 1 ;
     k=(j >> ++i) % 3 ;
     k=pegs[k + (l ∗ 3)] ;
     q=(k + l) % 3 + 1 ;
     printf ("move disk %d from %d to %d", i, k, q) ;
  }
}
```

$$count(x) = \sum_{k=0}^{\lfloor n/2\rfloor-1} \lceil\frac{2^{2k+1}}{3}\rceil,$$
$$count(y) = \sum_{k=0}^{\lfloor n/2\rfloor-1} \lceil\frac{2^{2k+1}-1}{3}\rceil,$$
$$count(z') = \sum_{k=0}^{\lfloor n/2\rfloor-1} \lceil\frac{2^{2k+1}-2}{3}\rceil.$$

Clearly:

$$count(z) + count(y') + count(x') + count(x) + count(y) + count(z') = 2^n - 1,$$

which enumerates the total number of moves in the solution of the towers of Hanoi problem. For example for $n = 7$, we have:

$$\begin{aligned} count(z) &= 1+2+6+22 = 31,\\ count(y') &= 0+1+5+21 = 27,\\ count(x') &= 0+1+5+21 = 27,\\ count(x) &= 1+3+11 = 15,\\ count(y) &= 1+3+11 = 15,\\ count(z') &= 0+2+10 = 12, \end{aligned}$$

and

$$\sum_{k=z,y',x',x,y,z'} count(k) = 31+27+27+15+15+12 = 2^7-1.$$

Also, it is well know that, the number of moves of ith disk is equal to the number of node in $(n-i+1)$th level of the full binary tree, which is equal to 2^{n-i}.

IV. Conclusion

Two versions of an iterative algorithm for the towers of Hanoi problem are presented. In each iteration, both versions compute the disk number and the direction of each move, the name of pegs are also calculated. The computations in each iteration are completely independent of the previous iterations. The first version of the algorithm uses an inner loop and the second version is a loopless one. In each iteration all the computations are done without using an auxiliary array of size n. The storage complexity of both versions of the algorithm is $O(1)$ and their time complexity is $O(2^n)$. According to the algorithm the number of different moves are formulated.

Acknowledgment

This research was partially supported by University of Tehran.

References

[1] M. D. Atkinson, The cyclic towers of Hanoi, *Inform. Process. Lett.* **13** (1981), 118-119.
[2] P. Buneman and L. Levy, The towers of Hanoi problem, *Inform. Process. Lett.* **10** (1980), 243-244.
[3] X. Chen and J. Shen, On the Frame-Stewart conjecture about the towers of Hanoi, *SIAM J. Comput.* **33** (2004), 584-589.

[4] H. Dudeney, *The Canterbury Puzzles*, Thomas Nelson & Sons, London, 1907.
[5] E. W. Dijkstra, *A Short Introduction to the Art of Programming*, Technisch Hogeschool Eindhoven, EWD 316, 1971.
[6] M. C. Er, A linear space algorithm for the towers of Hanoi problem by using a virtual disc, *Inform. Sci.* **47** (1989), 47-52.
[7] M. C. Er, A loopless and optimal algorithm for the cycle for Hanoi problem, *Inform. Sci.* **42** (1987), 283-287.
[8] M. C. Er, A loopless approach for constructing a fastest algorithm for the towers of Hanoi problem, *Intern. J. Comput. Math.* **20** (1986), 49-54.
[9] M. C. Er, The towers of Hanoi and binary numerals, *J. Inform. Optim. Sci.* **6** (1985), 147–152.
[10] J. S. Frame, Solution to advanced problem 3918, *Amer. Math. Monthly* **48** (1941), 216-217.
[11] T. D. Gedeon, The cyclic towers of Hanoi: An iterative solution produced by transformation, *Copmut. J.* **39** (1996), 353–356.
[12] P. Gupta, P. P. chakrabarti, and S. Ghose, The towers of Hanoi: Generalizations, specializations and algorithms, *Intern. J. Comput. Math.* **46** (1992), 149–161.
[13] P. J. Hayes, A note on the towers of Hanoi problem, *Copmut. J.* **20** (1977), 282-285.
[14] S. Klavžar, U. Milutinović, and C. Petr, On the Frame-Stewart algorithm for the multi-peg towers of Hanoi problem, *Discrete Appl. Math.* **120** (2002), 141-157.
[15] S. Klavžar and U. Milutinović, Simple Explicit Formulas for the Frame-Stewart Numbers, *Ann. Comb.* **6** (2002), 157–167
[16] H. Mayer and D. Perkins, Towers of Hanoi revisited, *SIGPLAN Notices* **19** (1984), 80–84.
[17] S. Maziar, Solution of the tower of Hanoi problem using a binary tree, *SIGPLAN Notice* **20** (1985), 16–20.
[18] B. Meyer, A note on iterative Hanoi, *SIGPLAN Notices* **19** (1984), 123-126.
[19] P. H. Schoute, De ringen van brahma, *Eigen Harrd* **22** (1884), 274-276.
[20] M. Sniedovich, OR/MS games: 2. Towers of Hanoi, *INFORMS Transcations on Education* **3** (2002), 34–51.
[21] B. M. Stewart, Advanced problem 3918, *Amer. Math. Monthly* **46** (1939), 363-363.
[22] B. M. Stewart, Solution to advanced problem 3918, *Amer. Math. Monthly* **48** (1941), 217-219.
[23] P. K. Stockmeyer, C. D. Bateman, J. W. Clark, C. R. Eyster, M. T. Harrison, N. A. Loehr, P. J. Rodriguez, and J. R. Simmons, Exchanging disks in the tower of Hanoi, *Intern. J. Comput. Math.* **59** (1995), 37–47.
[24] M. Saegedy, In how many steps the k peg version of the towers of Hanoi game can be solved, *Lec. Note. Comput. Sci.* **1563** (1999) 356–361.
[25] T. R. Walsh, The towers of Hanoi revisited: Moving the rings by counting the moves, *Inform. Process. Lett.* **15** (1982), 64–67.
[26] L. Xue-miao, A loopless approach to the multipeg towers of Hanoi, *Intern. J. Comput. Math.* **33** (1990) 13–29.