The Four-Peg Tower of Hanoi Puzzle

I-Ping Chu
Richard Johnsonbaugh
Department of Computer Science and Information Systems
DePaul University
Chicago, IL 60604
chu@depaul.edu, johnsonbaugh@depaul.edu

Abstract

We discuss a version of the Tower of Hanoi puzzle in which there are four pegs rather than three. The four-peg puzzle provides a rich source of exercises (samples of which are included) for students after the familiar three-peg version has been presented. We give an algorithm that solves the four-peg puzzle in the claimed minimum number of moves (see [2,4]). Our algorithm solves the four-peg puzzle in $O(4^{\sqrt{n}})$ moves whereas the best algorithm for the three-peg puzzle requires $2^n - 1$ moves. As far as we know, the minimum number of moves required to solve the four-peg puzzle is an open question.

The Original Puzzle

The Tower of Hanoi puzzle is a favorite example of a problem for which a recursive solution can be obtained readily and elegantly (see, e.g., [1,2,3,4]). The original puzzle consists of three pegs mounted on a board and n disks of various sizes with holes in their centers (see Figure 1). If a disk is on a peg, only a disk of smaller diameter can be placed on top of it. Given all the disks properly stacked on one peg as in Figure 1, the problem is to transfer the disks to another designated peg by moving one disk at a time. Subsequently, we refer to this puzzle as the three-peg puzzle.

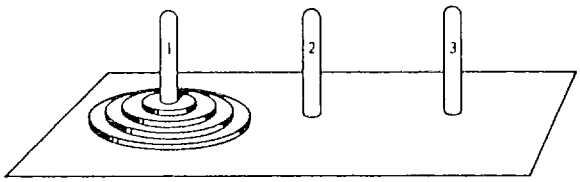

Figure 1: The three-peg Tower of Hanoi puzzle.

The three-peg puzzle was invented by the French mathematician François-Édouard-Anatole Lucas (1842–1891). Lucas edited a classic four-volume treatise on recreational mathematics which is still regularly referenced. He was the first person to call the sequence $1, 1, 2, 3, 5, \ldots$ the Fibonacci sequence. (A related sequence $1, 3, 4, 7, 11, \ldots$ is known as the Lucas sequence.)

Lucas created the following myth to accompany the Tower of Hanoi puzzle (and, one assumes, to help market the puzzle). The puzzle was said to be derived from a mythical gold tower which consisted of 64 disks. The 64 disks were to be transferred by monks according to the rules set forth previously. It was said that before the monks finished moving the tower, it would collapse and the world would end in a clap of thunder. Since at least 18,446,744,073,709,551,615 moves are required to solve the 64-disk Tower of Hanoi puzzle, we can be fairly certain something would happen to the tower before it was completely moved.

The traditional solution to the n-disk three-peg problem is as follows. We assume that the pegs are numbered $1, 2, 3$, and that the problem is to move the n disks, which are initially stacked on peg 1, to peg 3. We fix the bottom disk. We then recursively move the $n - 1$ top disks to peg 2. Next, we move the remaining disk on peg 1 to peg 3. Finally, we recursively move the $n-1$ disks on peg 2 to peg 3. It can be shown (see [3]) that this solution requires $2^n - 1$ moves and that it is optimal (i.e., the problem cannot be solved in fewer than $2^n - 1$ moves).

The Four-Peg Puzzle

The four-peg Tower of Hanoi puzzle is the same as the original puzzle except that there is an extra peg. (The puzzle has been generalized to an arbitrary number of pegs, see [2,4]). The problem, as with the three-peg puzzle, is to transfer the disks from one peg to another designated peg by moving one disk at a time. Of course, we could simply ignore one peg and use the solution to

the three-peg problem. As before, this solution would require $2^n - 1$ moves. However, as we will show, the extra peg allows moves different from the three-peg problem and, in fact, allows us to construct an algorithm that uses dramatically fewer moves.

Exercises

1. Construct an algorithm for the four-peg puzzle that is faster than the standard three-peg algorithm.
2. Why doesn't the proof of optimality of the three-peg solution go through for the four-peg problem?

An Algorithm for the Four-Peg Puzzle

We now present our algorithm to solve the four-peg Tower of Hanoi puzzle. We assume that the pegs are numbered 1, 2, 3, 4 and that the problem is to move n disks, which are initially stacked on peg 1, to peg 4. We fix k disks at the bottom of peg 1. (We describe how to determine the value of k later.) We then recursively invoke this algorithm to move the $n-k$ disks at the top of peg 1 to peg 2. During this part of the algorithm, the k bottom disks on peg 1 remain fixed. Next, we move the k disks on peg 1 to peg 4 by invoking the optimal three-peg algorithm and using only pegs 1, 3, and 4. Finally, we again recursively invoke this algorithm to move the $n-k$ disks on peg 2 to peg 4. During this part of the algorithm, the k disks on peg 4 remain fixed.

The preceding paragraph describes our solution to the four-peg puzzle except for specifying the value of k. Notice that any choice of a value for k between 1 and $n-1$, inclusive, gives an algorithm. (The three-peg algorithm resembles our four-peg algorithm with $k = 1$.) The idea is to choose a value for k between 1 and $n-1$ that uses the fewest moves. We wrote a program to compute the minimal k for various values of n (see Table 1). Notice that an interesting pattern develops in the column that lists the values of k: there are three 2's, followed by four 3's, followed by five 4's, and so on. This suggests that we should choose k to be the largest integer satisfying

$$\sum_{i=1}^{k} i \leq n. \qquad (1)$$

To simplify the notation, we let $SUMINT(j)$ denote the sum of the first j positive integers. Throughout the remainder of this article, for a given value of n, we *define* k to be the largest integer for which $SUMINT(k) \leq n$.

If we let $T_4(n)$ denote the number of moves required by our algorithm to solve the n-disk four-peg puzzle (with k as defined previously) and we let $T_3(n)$ denote the number of moves required by the traditional (optimal) algorithm to solve the n-disk three-peg puzzle, we see that

$$T_4(n) = 2T_4(n-k) + T_3(k).$$

n	k
3	2
4	2
5	2
6	3
7	3
8	3
9	3
10	4
11	4
12	4
13	4
14	4
15	5

Table 1: The value of k which minimizes the number of moves in the four-peg n-disk Tower of Hanoi algorithm.

n	Number of moves using the 3-peg algorithm	Number of moves using the 4-peg algorithm
1	1	1
2	3	3
3	7	5
4	15	9
5	31	13
6	63	17
7	127	25
8	255	33
9	511	41
10	1023	49
16	65535	161
32	4294967295	1281
64	18446744073709551615	18433

Table 2: Comparison of the number of moves required by two Tower of Hanoi algorithms.

Table 2 compares the number of moves to solve the n-disk Tower of Hanoi puzzle using the optimal three-peg algorithm with the number of moves required by our four-peg algorithm for various values of n. In the following section, we give a formula for $T_4(n)$. The last line of Table 2 shows that if Lucas' monks had had an extra peg, they might have gotten the disks moved.

Exercises

3. Write a program that, given n, computes the largest integer k such that $SUMINT(k) \leq n$.
4. Write a program that lists the moves made by the four-peg algorithm.
5. Write a program to compute $T_4(n)$.
6. Consider a variant of the algorithm in this section in which we always take $k = 2$. Show that if $n \geq 3$, the modified algorithm uses fewer moves than does the optimal three-peg algorithm.

7. Find an explicit formula for the number of moves required by the algorithm of Exercise 6.

Analysis of the Four-Peg Algorithm

We use the notation previously defined. In addition, if k is the largest integer satisfying equation (1), we define r by the equation

$$r = n - SUMINT(k).$$

Notice that if in Table 2 we compute $T_4(n) - T_4(n-1)$, we obtain the sequence $2, 2, 4, 4, 4, 8, 8, 8, 8, \ldots$. That is, we obtain two 2's, three 4's, four 8's, and so on. This suggests that

$$T_4(n) = \sum_{i=1}^{k} i2^{i-1} + r2^r = (k + r - 1)2^k + 1.$$

We verify the conjecture

$$T_4(n) = (k + r - 1)2^k + 1 \qquad (2)$$

by using induction on n.

We use primes to denote the k and r values for the $(n-k)$-disk problem. Specifically, we define k' to be the largest integer satisfying

$$SUMINT(k') \leq n - k$$

and we define

$$r' = n - k - k'.$$

For small values of n, we can directly verify equation (2). We assume that equation (2) is true for values less than n. We note that because k is the largest integer satisfying equation (1), we must have

$$n - k \leq SUMINT(k). \qquad (3)$$

(If $SUMINT(k) < n-k$, then $SUMINT(k+1) \leq n$, so k was not the largest integer for which $SUMINT(k) \leq n$.) We consider two cases.

Case 1: $n - k < SUMINT(k)$.

We always have $SUMINT(k) \leq n$, so

$$SUMINT(k-1) \leq n - k.$$

This last inequality, together with the Case 1 assumption, tells us that $k' = k - 1$. Now

$$r = n - SUMINT(k) = n - k - SUMINT(k-1) = r'.$$

Therefore,

$$\begin{aligned} T_4(n) &= 2T_4(n-k) + T_3(k) \\ &= 2(k' + r' - 1)2^{k'} + 2 + 2^k - 1. \\ &= 2(k - 1 + r - 1)2^{k-1} + 2 + 2^k - 1 \\ &= (k + r - 1)2^k + 1, \end{aligned}$$

as desired.

Case 2: $n - k = SUMINT(k)$.

This time, $k' = k$ and $r' = 0$, and a computation like that for Case 1 again gives the desired conclusion. We have proved equation (2).

Asymptotic Behavior of the Four-Peg Algorithm

Since $SUMINT(k) = k(k+1)/2$, given n, k is the largest integer satisfying

$$\frac{k(k+1)}{2} \leq n.$$

If we solve the equation

$$\frac{x(x+1)}{2} = n$$

for x and take the floor of the positive root, we obtain

$$k = \lfloor \frac{\sqrt{1+8n} - 1}{2} \rfloor.$$

From this formula, it is easy to see that $k \leq \sqrt{2n}$. Inequality (3) shows that $r \leq k$. Therefore,

$$\begin{aligned} T_4(n) &= (k + r - 1)2^k + 1 \\ &< 2k2^k + 1 \leq 2(\sqrt{2n})2^{\sqrt{2n}} + 1 = O(4^{\sqrt{n}}). \end{aligned}$$

Thus the four-peg algorithm is asymptotically faster than the optimal three-peg algorithm [which is $O(2^n)$].

Conclusions

The four-peg Tower of Hanoi puzzle extends the three-peg puzzle typically presented in data structures and discrete mathematics courses. In discussing the four-peg puzzle, several interrelated issues naturally arise: recursion, recurrence relations, mathematical induction, and analysis of algorithms. In addition, we have to formulate and verify conjectures based on brute-force calculations.

References

1. N. Dale and C. Weems, *Introduction to Pascal and Structured Design*, 2nd ed., Heath, Lexington, Mass., 1987.

2. A. M. Hinz, "The Tower of Hanoi," *L'Enseignement Mathématique*, t. 35, 1989, pp. 289–321.

3. R. Johnsonbaugh, *Discrete Mathematics*, 2nd ed., Macmillan, New York, 1990.

4. Problem 3918, *Amer. Math. Mo.*, March 1941, pp. 216–219.