Exchanging Disks in the Tower of Hanoi

Paul K. Stockmeyer
Department of Computer Science
College of William and Mary
Williamsburg, Virginia 23187-8795

C. Douglass Bateman, James W. Clark
Cyrus R. Eyster, Matthew T. Harrison
Nicholas A. Loehr, Patrick J. Rodriguez
and Joseph R. Simmons III
1993 Virginia Governor's School for the Gifted
in Science, Mathematics, and Technology *

September 1984

Abstract

We examine a variation of the famous Tower of Hanoi puzzle posed but not solved in a 1944 paper by Scorer *et al.* [5]. In this variation, disks of adjacent sizes can be exchanged, provided that they are at the top of their respective stacks. We present an algorithm for solving this variation, analyze its performance, and prove that it is optimal. Several exercises are listed at the end, ranging in difficulty from elementary to research level.

1 Introduction

The Tower of Hanoi puzzle, invented in 1883 by Édouard Lucas and described by him in [4, pp. 55–59], has been undergoing a dramatic revival in popularity during the past 10 years, largely due to its use as a programming exercise in elementary computer courses. Many variations on the original puzzle also have been proposed and solved. For an excellent survey, with an extensive bibliography, see [3]. One variation posed in 1944 [5], however, appears not to have been studied. In this paper we present an algorithm for solving this variation, analyze the performance of our algorithm, and prove that our solution is optimal.

The traditional Tower of Hanoi consists of three posts called A, B, and C, and a set of n, typically 8, pierced disks of differing diameters that can be stacked on the posts. The disks are usually numbered from 1 to n in order of increasing size. The tower is formed initially by stacking the disks onto post A in decreasing order from bottom to top. The challenge is to transport the tower to post C by moving the disks one at a time from post to post,

*Funded by the Commonwealth of Virginia

subject to the rule that no disk can ever be placed on top of a smaller disk. This game can be solved by first recursively transporting the sub-tower consisting of disks 1 through $n-1$ from post A to post B, next moving disk n from post A to post C, and then recursively transporting the sub-tower from post B to post C.

The game proposed in [5] is similar, but the rules governing the moves are different. They are:

Rule 1. The smallest disk (number 1) can be moved to any post, at any time;

Rule 2. For d satisfying $1 < d \leq n$, disks $d-1$ and d can be exchanged, provided each is at the top of its respective stack.

No other moves are allowed. Note that these rules ensure that the disks on each post are always stacked in decreasing order of size, as with the original game. We encourage the reader to explore this version for small values of n before continuing.

The presentation and discussion of our algorithm will be simplified by first establishing some notation and terminology. We refer to any proper distribution of disks onto posts as a *state*. Each state will be denoted by a sequence of length n from the set {A, B, C}, naming the posts that the disks occupy, *from the largest to the smallest*. (Some authors have used the opposite order, but we feel our order is in many ways more natural. The position of the largest disk is the most significant aspect of a state, and most forms of information, including numbers, are written with the most significant part first in the western world.) Each move of the puzzle transforms it from one state to another. Thus for $n = 8$, the starting state is AAAAAAAA, and the final state is CCCCCCCC. In the traditional version of the game, the move of disk 8 at the half-way point transforms the puzzle from state ABBBBBBB to state CBBBBBBB. For the variation we are considering, the middle move is an interchange of disks 7 and 8, transforming the puzzle from state ACBBBBBB to state CABBBBBB.

These ideas can be made clearer by examining the *state graph* of this puzzle, another concept first introduced in [5]. In this graph, the vertices are the 3^n possible states, and the edges are the $(3^{n+1}-3)/2$ allowable moves between states. It is claimed in [5] that this graph can be drawn in the plane with all edges parallel to the sides of an equilateral triangle, as can the state graph of the original Tower of Hanoi puzzle. In this new version, however, the lines are not all of the same length. These claims are illustrated in FIGURE 1, which displays the state graph for the case $n = 5$. We will make frequent use of standard graph theory terminology in what follows, and will use the words *vertex* and *state* interchangeably, as well as the words *edge* and *move*. Any definitions that we omit can be found in standard works such as [2].

The association of state names with vertices is relatively simple for this puzzle. The first letter specifies the major triangular subgraph to which the state belongs, the second letter specifies the triangular subgraph within this subgraph, and so forth. At each level, A denotes the lower left triangle, B the upper triangle, and C the lower right triangle. We encourage the reader to trace and name the vertices in FIGURE 1 along the unique minimal path, of length 25, from vertex AAAAA at the lower left corner to vertex CCCCC at the lower right. (The length of a path in a graph is the number of edges it contains, regardless of how long these edges are drawn.)

Figure 1. State Graph of the 5-Disk Puzzle.

2 The Algorithm

We present our algorithm in the form of a Pascal procedure. When invoked by the statement `new_hanoi(n)`, it displays instructions for transporting a tower of n disks from post A to post C. The reader may wish to trace the action of each of the component sub-procedures on the graph in FIGURE 1.

```
procedure new_hanoi(n: integer);
{ Moves a tower of n disks from post A to post C, }
{ following the rules of the interchange version. }

   type post = 'A' .. 'C';

   procedure move(d: integer; X, Y: post);
   { Moves disk d from post X to post Y. }
   begin {move}
     if d = 1 then
       writeln('Move disk 1 from post ', X, ' to post ', Y)
     else
       writeln('Swap disk ', d-1, ' on post ', Y, ' with disk ',
                               d, ' on post ', X)
   end;  {move}

   procedure initial(k: integer; X, Y, Z: post); forward;
   procedure final(k: integer; X, Y, Z: post); forward;
   procedure early_middle(k: integer; X, Y, Z: post); forward;
   procedure late_middle(k: integer; X, Y, Z: post); forward;
```

```
procedure total(k: integer; X, Y, Z: post);
{ Converts state X^k to state Y^k. }
begin {total}
  if k >= 1 then
    begin
      initial(k-1, X, Y, Z);
      move(k, X, Y);
      final(k-1, X, Y, Z)
    end
end;  {total}

procedure initial(k: integer; X, Y, Z: post);
{ Converts state X^k to state YZ^(k-1). }
begin {initial}
  if k >= 1 then
    begin
      initial(k-1, X, Y, Z);
      move(k, X, Y);
      early_middle(k-1, X, Z, Y)
    end
end;  {initial}

procedure final(k: integer; X, Y, Z: post);
{ Converts state XZ^(k-1) to state Y^k. }
begin {final}
  if k >= 1 then
    begin
      late_middle(k-1, Z, Y, X);
      move(k, X, Y);
      final(k-1, X, Y, Z)
    end
end;  {final}

procedure early_middle(k: integer; X, Y, Z: post);
{ Converts state XY^(k-1) to state Y^k. }
begin {early_middle}
  if k >= 1 then
    begin
      total(k-2, Y, Z, X);
      move(k, X, Y);
      final(k-1, X, Y, Z)
    end
end;  {early_middle}
```

```
  procedure late_middle(k: integer; X, Y, Z: post);
  { Converts state X^k to state YX^(k-1). }
  begin {late_middle}
    if k >= 1 then
      begin
        initial(k-1, X, Y, Z);
        move(k, X, Y);
        total(k-2, Z, X, Y)
      end
  end;  {late_middle}

begin {new_hanoi}
  total(n, 'A', 'C', 'B');
end;  {new_hanoi}
```

In this program, the procedure `move(d, X, Y)` writes out an instruction for transferring disk d from post X to post Y. If d equals 1, then this is a simple move; otherwise it is an exchange with disk $d-1$. Somewhat more formally, the PRECONDITIONS for this procedure are: $\{X, Y\} \subset \{$'A', 'B', 'C'$\}$ with $X \neq Y$; disk d is on post X; disk $d-1$ is on post Y (if $d > 1$); and for i satisfying $1 \leq i < d-1$, disk i is on the remaining post (if $d > 2$). The POSTCONDITIONS are: disk d is on post Y; disk $d-1$ is on post X (if $d > 1$); and all other disks are unchanged. Clearly the action of this procedure is consistent with the rules of this game.

Each of the remaining five procedures generates instructions for a sequence of moves that performs some prescribed transformation of the puzzle. They are co-routines, calling each other, and all have a similar format. The first formal parameter, k, is the number of the largest disk moved by the procedure. The parameters X, Y, and Z are, respectively, the post that disk k occupies before execution, the post that disk k occupies after execution, and the remaining post. The code for each procedure consists of three statements, the middle of which is the one that moves disk k to its desired location. The first statement is a procedure call that transforms the first $k-1$ disks to make this possible, while the third is a procedure call that then transforms the first $k-1$ disks to their specified positions.

For example, the procedure call `total(k, X, Y, Z)` will transform the puzzle from state X^k to state Y^k, as claimed by the comment in the listing of this procedure. To accomplish this, the procedure first calls `initial(k-1, X, Y, Z)` to transform the puzzle from state X^k to state XYZ^{k-2}. This makes it possible to `move(k, X, Y)` next, which produces state YXZ^{k-2}. Then the call `final(k-1, X, Y, Z)` completes the action, yielding state Y^k. The two co-routines called by this procedure each behave in an analogous manner.

The transformation accomplished by each of the five co-routines should be clear from its comment, but the reader who prefers more formal descriptions can easily convert the comments into PRE- and POSTCONDITIONS. For example, the PRECONDITIONS for the procedure `early_middle` are: $\{X, Y, Z\} = \{$'A', 'B', 'C'$\}$; if $k \geq 1$, then disk k is on post X; and if $k \geq 2$, then for i satisfying $1 \leq i < k$, disk i is on post Y. The POSTCONDITIONS are: if $k \geq 1$, then for all i satisfying $1 \leq i \leq k$, disk i is on post Y; and all disks numbered greater than k remain where they were. It is then a routine exercise to prove simultaneously by induction that all procedures do indeed perform as claimed. Thus the procedure call

`total(n, 'A', 'C', 'B')` at the end of the listing will generate a written list of moves that solve this version of the n-disk puzzle, tracing out a path from vertex A^n to vertex C^n in the state graph.

3 Analysis

We now investigate the number of moves that are performed by our algorithm. First, note that procedures `initial` and `final` are essentially mirror images of each other, and hence generate equal numbers of moves. The same is true for procedures `early_middle` and `late_middle`. We define $f(n)$, $g(n)$, and $h(n)$ to be the number of moves generated by calls to procedures `total`, `initial` (or `final`), and `early_middle` (or `late_middle`), respectively, with first parameter n. We seek a closed form expression for $f(n)$.

An examination of the code reveals that all three functions have value 0 if $n \leq 0$, while for $n \geq 1$ we have

$$f(n) = 2g(n-1) + 1, \tag{1}$$
$$g(n) = g(n-1) + h(n-1) + 1, \tag{2}$$
$$\text{and } h(n) = f(n-2) + g(n-1) + 1. \tag{3}$$

Values of these functions for small positive values of n can be calculated easily from these equations.

Replacing $f(n-2)$ in equation (3) with its equivalent from equation (1) yields

$$\begin{aligned} h(n) &= [2g(n-3) + 1] + g(n-1) + 1 \\ &= g(n-1) + 2g(n-3) + 2, \end{aligned}$$

for $n \geq 3$. Now using this expression to remove $h(n-1)$ from equation (2) we get

$$\begin{aligned} g(n) &= g(n-1) + [g(n-2) + 2g(n-4) + 2] + 1 \\ &= g(n-1) + g(n-2) + 2g(n-4) + 3, \end{aligned}$$

for $n \geq 4$. Finally, using this expression in equation (1) produces

$$\begin{aligned} f(n) &= 2[g(n-2) + g(n-3) + 2g(n-5) + 3] + 1 \\ &= [2g(n-2) + 1] + [2g(n-3) + 1] + [4g(n-5) + 2] + 3 \\ &= f(n-1) + f(n-2) + 2f(n-4) + 3, \end{aligned} \tag{4}$$

for $n \geq 5$. In fact, equation (4) is valid for $n = 4$ as well. Thus $f(n)$ is completely determined for all non-negative values of n by the recurrence relation

$$f(n) - f(n-1) - f(n-2) - 2f(n-4) = 3, \tag{5}$$

together with the boundary values $f(0) = 0$, $f(1) = 1$, $f(2) = 3$, and $f(3) = 7$.

This recurrence is easily solved by standard methods, as found, for example, in [1, pp. 65–72]. The characteristic equation for this relation is

$$x^4 - x^3 - x^2 - 2 = 0,$$

with two real roots $r_1 = 1.85356$ and $r_2 = -1.15673$, and two complex roots $r_3 = \rho(\cos(\theta) + i\sin(\theta))$ and $r_4 = \rho(\cos(\theta) - i\sin(\theta))$, where $\rho = 0.96582$ and $\theta = 1.41320$ (in radians). It is easy to see that $f_p(n) = -1$ is a particular solution to our recurrence relation. This implies that the general solution is of the form

$$f_g(n) = c_1 r_1^n + c_2 r_2^n + c_3 \rho^n \cos(n\theta) + c_4 \rho^n \sin(n\theta) - 1,$$

for arbitrary constants c_1, c_2, c_3, and c_4. Using the boundary values to solve for these constants in our case, we get $c_1 = 1.19188$, $c_2 = -0.08028$, $c_3 = -0.11160$, and $c_4 = -0.29896$. Thus

$$\begin{aligned} f(n) =& 1.19188\,(1.85356)^n - 0.08028\,(-1.15673)^n \\ &- 0.11160\,(0.96582)^n \cos(1.41320\,n) \\ &- 0.29896\,(0.96582)^n \sin(1.41320\,n) - 1, \end{aligned} \tag{6}$$

for $n \geq 0$.

The first term is the dominant one. The second term makes even-numbered terms somewhat smaller, and odd-numbered terms somewhat larger, than they otherwise would be. The next two terms approach 0, and serve only to make $f(n)$ always be an integer. Asymptotically, we have

$$f(n) \sim 1.19188 \times 1.85356^n.$$

4 Proof of Optimality

Consider the shortest sequence of moves that would transform the puzzle from one given state to another. It is tempting to believe that if the largest disk is already on its correct post it would not move at all; and if it is on the wrong post, it would move exactly once, from its initial position to its desired location. In terms of the graph, these beliefs are: the shortest path between two vertices in the same major subtriangle is contained within that subtriangle; and the shortest path between vertices in different major subtriangles never enters the third subtriangle. If these beliefs were true, then it would be relatively easy to prove that our algorithm solves the game in a minimum number of moves. As we shall see, the first belief is in fact true, but the second is not. An analogous belief for the standard Tower of Hanoi puzzle is also false, and several careless mathematicians have been tripped up by such a mistaken belief. It is not obvious that our algorithm is a minimal solution; a proof must be given.

Earlier we defined functions $f(n)$, $g(n)$, and $h(n)$ as the number of moves made by our procedures `total`, `initial`, and `early_middle`, respectively. We now define $F(n)$, $G(n)$, and $H(n)$ as the minimum number of moves to accomplish the corresponding tasks. That is, $F(n)$ is the length of the shortest path from state X^n to state Y^n, $G(n)$ is the distance between states X^n and YZ^{n-1}, and $H(n)$ gives the minimum number of moves between XY^{n-1} and Y^n. We will prove that all of our procedures are optimal: that $F(n) = f(n)$, $G(n) = g(n)$, and $H(n) = h(n)$. For purposes of this section only, we also need to consider transforming state XY^{n-1} to state YX^{n-1}. Such a transformation is always possible — the code `total(n-2, Y, Z, X); move(n, X, Y); total(n-2, Z. X. Y)` will do the job quite nicely. We define $K(n)$ to be the minimum number of moves needed for this task.

LEMMA 1. *In a shortest path between any given pair of states, the largest disk moves*
a) not at all if it is already on the desired post, or
b) either once or twice otherwise.

Proof: We first observe that the shortest route from one state to another will never visit any state more than once. In particular, a path will not cross over from one of the three major triangular subgraphs to another and then later cross back again. Equivalently, the largest disk will never move from one post to another and then move back again.

For part a), we assume without loss of generality that both the start state S_1 and the goal state S_2 have the largest disk on post X. The only possible way in which the largest disk could move at all in a minimal length path from S_1 to S_2 would be to move from post X to another post, say Y, then later to the remaining post Z, and then still later back to post X again. Such a path would look like S_1, ..., XYZ^{n-2}, YXZ^{n-2}, ..., YZX^{n-2}, ZYX^{n-2}, ..., ZXY^{n-2}, XZY^{n-2}, ..., S_2. Now the part of this path from state XYZ^{n-2} to state XZY^{n-2} has length at least $1 + K(n-1) + 1 + K(n-1) + 1$, or $2K(n-1) + 3$. But comments made just prior to the statement of this lemma imply that we can get from XYZ^{n-2} to XZY^{n-2} directly in only $K(n-1)$ moves by ignoring the largest disk and leaving it where it is, on post X. By patching in this shortcut, we get a strictly shorter route from S_1 to S_2. Thus no shortest path can include a move of the largest disk if it is already on its desired post.

For part b), we merely observe that for any sequence of moves in which the largest disk moves more than twice and ends up on a different post from the one on which it started, some state must be visited more that once. We recall that such a sequence can not be minimal.

COROLLARY. *For any two states S_1 and S_2, let k be the number of the largest disk for which S_1 and S_2 differ. Then in a shortest path from S_1 to S_2, disk k moves either once or twice, and for all i satisfying $k < i \leq n$, disk i does not move.*

This corollary follows from LEMMA 1 by an easy induction argument. Among other helpful things, it tells us that a minimal length path from X^n to, say, $X^{n-k}YZ^{k-1}$ is essentially the same as a minimal length path from X^k to YZ^{k-1}. We can safely ignore disks numbered greater than k in all of our procedures when proving optimality.

It is not always obvious whether the base disk should move once or twice in a minimal path. But in case anyone at this point is under the mistaken belief that the one-move alternative is always best, we consider paths from XYZ^{n-2} to ZYX^{n-2}. The shortest path with one move of disk n has length $2K(n-1) + 1$, while the best two-move path takes $K(n-1) + 2$ moves. These lengths are equal for $n = 3$, but for $n \geq 4$ the path with two moves of disk n is strictly shorter.

We need to know two more facts before proving our main theorem.

LEMMA 2. *For all $n \geq 1$ we have*
a). $H(n) \leq K(n) + F(n-1)$, and
b). $F(n-1) \leq K(n) + H(n)$.

Proof: a). By definition, $H(n)$ is the minimum number of moves needed to get from state XY^{n-1} to state Y^n. One not necessarily optimal way of carrying out this task is to go

from XY^{n-1} to YX^{n-1} in $K(n)$ moves, and then from YX^{n-1} to $YY^{n-1} = Y^n$ in $F(n-1)$ additional moves. The conclusion follows.

b). We have that $F(n-1)$ is the minimum number of moves from state X^{n-1} to state Y^{n-1}, or equivalently, from YX^{n-1} to Y^n. A not necessarily optimal way of accomplishing this is to transform YX^{n-1} to XY^{n-1} in $K(n)$ moves, and then transform XY^{n-1} to Y^n in an additional $H(n)$ moves.

We are now able to prove our main result of this section.

THEOREM 1. *For all $n \geq 0$ we have*
a). $F(n) = f(n)$,
b). $G(n) = g(n)$, and
c). $H(n) = h(n)$.

Proof: We prove all three claims simultaneously, by induction. When $n = 0$, all six functions take the value 0, while for $n = 1$ the six function values are clearly all 1. Now suppose that the claims are true for all values of the variable less than some integer $n \geq 2$.

a). A minimal sequence of $F(n)$ moves from state X^n to state Y^n must move disk n either once, from X to Y, or twice, from X to Z to Y. Also, subsequences before, after, and perhaps between such moves must be minimal. The first alternative makes $G(n-1)+1+G(n-1)$ moves, while the second makes $G(n-1)+1+K(n-1)+1+G(n-1)$. Clearly the first alternative is the optimal one. Moreover, by our inductive hypothesis, we have $G(n-1) = g(n-1)$. The result then follows from equation (1). To summarize:

$$\begin{aligned} F(n) &= \min \begin{cases} G(n-1) + 1 + G(n-1) \\ G(n-1) + 1 + K(n-1) + 1 + G(n-1) \end{cases} \\ &= 2G(n-1) + 1 \\ &= 2g(n-1) + 1 \\ &= f(n). \end{aligned}$$

b). In finding a minimal sequence of $G(n)$ moves that transforms the puzzle from state X^n to state YZ^{n-1}, we must again consider two alternatives: whether disk n makes 1 or 2 moves. Here Lemma 2.a tells us that the one-move alternative is the optimal one. We have

$$\begin{aligned} G(n) &= \min \begin{cases} G(n-1) + 1 + H(n-1) \\ G(n-1) + 1 + K(n-1) + 1 + F(n-2) \end{cases} \\ &= G(n-1) + H(n-1) + 1 \\ &= g(n-1) + h(n-1) + 1 \\ &= g(n). \end{aligned}$$

c). In finding a minimal sequence of $H(n)$ moves from state XY^{n-1} to state Y^n, we face the same two alternatives. This time Lemma 2.b tells us that the one-move alternative

is optimal, and we have

$$\begin{aligned} H(n) &= \min \begin{cases} F(n-2) + 1 + G(n-1) \\ H(n-1) + 1 + K(n-1) + 1 + G(n-1) \end{cases} \\ &= F(n-2) + G(n-1) + 1 \\ &= f(n-2) + g(n-1) + 1 \\ &= h(n). \end{aligned}$$

In fact, this proof shows more than was claimed. In each case, the alternative with only 1 move of disk n made strictly fewer moves than the alternative that moved disk n twice. Thus by induction, the minimal move sequences are unique.

COROLLARY *The sequence of $f(n)$ moves generated by the procedure* `new_hanoi(n)` *is the unique minimal solution to this puzzle.*

5 Coda

We close with a few exercises that the reader may enjoy, in approximate order of difficulty. The last two should be considered research problems.

1. Prove that the state graph of this puzzle is connected — that it is possible to get from any state to any other state.

2. Prove that of the $f(n)$ moves needed to solve the n-disk puzzle, $f(n-1)$ are exchanges and $f(n) - f(n-1)$ are simple moves of disk 1. Which is larger?

3. Prove that states A^n and C^n are at maximal distance apart. That is, show that the distance between any two states is at most $f(n)$. Are there any other pairs this far apart, aside from obvious symmetries?

4. Prove that the state graph is non-Hamiltonian for $n \geq 5$. In other words, show that there is no sequence of moves that visits each state exactly once. Find a Hamiltonian circuit for the case $n = 4$.

5. The length of a path in a graph is usually defined as the number of edges the path contains, as we have seen. However, once a graph has been embedded in the plane, we can talk about the physical, or geometrical, distance along a path in the embedding. Let $d(n)$ be the physical distance of the minimal path from vertex A^n to vertex C^n, using the embedding illustrated in FIGURE 1, where the shortest edges are defined to have length 1. Prove that

$$\begin{aligned} d(n) &= \tfrac{7}{2}\,2^n - 3.19914\,(1.85356)^n - 0.10227\,(-1.15673)^n \\ &\qquad - 0.19858\,(0.96582)^n \cos(1.41320\,n) \\ &\qquad - 0.16606\,(0.96582)^n \sin(1.41320\,n) \\ &= \tfrac{7}{2}\,2^n + \Theta(1.85356^n), \end{aligned}$$

while the straight line distance between these two vertices in the embedded graph is $2^n - 1$. Thus in the limit, as n grows large, this path becomes $\frac{7}{2}$ times as long as the base of the graph.

6. Write a computer program that generates the same sequence of moves that our program does but without using recursion, either directly, indirectly through co-routines, or implemented with a stack.

7. Determine the average distance between a pair of randomly chosen states.

References

[1] Brassard, Gilles and Paul Bratley, *Algorithmics — Theory and Practice*. Prentice Hall (Englewood Cliffs, NJ), 1988.

[2] Harary, Frank, *Graph Theory*. Addison-Wesley (Reading, MA), 1969.

[3] Hinz, Andreas M. "The Tower of Hanoi". *Enseign. Math. 35* (1989), 289–321.

[4] Lucas, Édouard, *Récréations Mathématiques, vol.III*. Gauthier-Villars (Paris), 1893. Reprinted, Albert Blanchard (Paris), several years.

[5] Scorer, R. S., P. M. Grundy, and C. A. B. Smith. "Some Binary Games". *Math. Gazette 28* (1944), number 280 (July), 96–103.

Current Affiliations:
Mr. Bateman, Mr. Harrison, and Mr. Rodriguez: University of Virginia;
Mr. Clark and Mr. Eyster: Massachusetts Institute of Technology;
Mr. Loehr: Virginia Polytechnic Institute and State University;
Mr. Simmons: Florida State University.