NEW VARIATIONS ON THE TOWER OF HANOI

PAUL K. STOCKMEYER AND FRED LUNNON

Abstract. The Tower of Hanoi puzzle, invented by Édouard Lucas in 1883, is well known to students of discrete mathematics and computer science. Many variations have been proposed as exercises and challenges over the past 125 years, including some with more than three pegs which remain unsolved.

In this paper we pose several new variations, all involving two or more stacks of disks, identical except for color. The goal in each variation is to move each stack of disks from its initial location to its final location. As usual, disks must be moved one at a time, and a disk can never sit above a disk of equal or smaller diameter, regardless of color.

Hints are provided, along with solutions.

1. Introduction

The Tower of Hanoi, a puzzle invented by French mathematician Édouard Lucas in 1883, consists of three vertical pegs and some number n of disks, all with a hole in the center that allows them to be stacked on the pegs. The disks are all of different radii, and we imagine that they are numbered 1 through n in increasing order of size. They are initially stacked in order on one of the pegs, with disk n on the bottom and 1 on the top. The object is to transport this tower from the starting peg to a destination peg, moving one disk at a time, and always placing a disk on either an empty peg or on a larger disk. The third peg is used to temporarily hold disks during the process. See [1] or [5] for an early discussion of this puzzle.

It is well known that there is a unique minimum-move solution that takes 2^n-1 steps. It is most easily described recursively: for $n > 0$, recursively move a tower of $n-1$ disks from the initial peg to the temporary peg; move disk n from the initial peg to the destination peg; and then recursively move the tower of $n-1$ disks from the temporary peg to the destination peg. If $M_0(n)$ is the number of moves made, we have the recurrence relation $M_0(n) = M_0(n-1) + 1 + M_0(n-1)$ for $n > 0$ with $M_0(0) = 0$, which has the solution $M_0(n) = 2^n - 1$ given above.

Many variations on this simple puzzle soon followed, with features such as colored disks, multiple pegs, and restricted moves. In fact, Lucas himself invented a version with five pegs and a stack of disks of four different colors [3]. The puzzles presented here, most of which were invented and patented by Victor Mascolo of Bayville, New York, all involve multiple stacks of disks. Each stack has a unique color, but otherwise the stacks are identical. As before, we imagine that the disks in each stack are numbered from 1 to n in increasing order of size. Each stack must be moved from its assigned initial peg to its destination peg, always moving just one disk at a time. Unlike the multi-stack Towers of Antwerpen puzzle posed by Derrick Wood [7] and solved by Steven Minsker [4], we enforce a strict size rule: a disk can only be placed on an empty peg or on top of a strictly larger disk of any color. For $n > 1$ this necessitates that we have at least two more pegs than stacks. For each puzzle, a minimum-move algorithm is desired, along with a proof of minimality.

2. The Puzzles

We present our puzzles in increasing order of difficulty, starting with the easiest one.

Puzzle 1: There are four pegs, numbered from 0 to 3. A stack of n white disks, stack 0, is initially on peg 0, and a stack of n black disks, stack 1, is initially on peg 2. The destination pegs for the white stack and the black stack are pegs 2 and 0, respectively. In other words, the goal is to interchange the two stacks, using pegs 1 and 3 as temporary holding pegs.

One obvious algorithm is to use the standard tower algorithm to move the white stack from peg 0 to peg 1, using 3 as the temporary peg; then moving the black stack from peg 2 to peg 0, again using 3 as the temporary peg; and finally moving the white stack from peg 2 to peg 3, once more using 3 as the temporary peg. Clearly this takes $3(2^n-1)$ moves. Can you do better?

Puzzle 2: This is actually the first puzzle we explored, and the first puzzle invented by Mr. Mascolo. It is sometimes referred to as *Turtle*. It is the same as Puzzle 1 except that peg 1 can hold only white disks (from stack 0), while peg 3 can hold only black disks (from stack 1). Thus the white disks can use only pegs 0, 1, and 2 in moving from 0 to 2, while the black disks can use only pegs 2, 3, and 0 in moving from 2 to 0.

It is not immediately obvious that this puzzle can always be solved. Certainly the moves of the white disks must be carefully intertwined with the moves of the black disks in any possible solution.

Puzzle 3: For this puzzle we have s stacks, $s \geq 3$, numbered 0 through $s-1$, and $m = 2s$ pegs, numbered 0 through $2s-1$. Stack i starts on peg $2i$, can temporarily make use of peg $2i+1$, and must end up on peg $2i+2$, where all numbers are reduced modulo $2s$. Thus peg $2i$ serves as both the starting peg for stack i and the destination peg for stack $i-1$, and peg $2i+1$ is for the exclusive use of stack i. (Puzzle 2 was this puzzle with $s = 2$ stacks, but the puzzle for $s \geq 3$ behaves somewhat differently.)

Puzzle 4: This is a 2-stack 5-peg puzzle, similar to Puzzle 2 but with an additional peg 4 that can temporarily hold disks of either color. The white disks can use pegs 0, 1, 2, and 4 in moving from peg 0 to peg 2, while the black disks can use pegs 2, 3, 0, and 4 in moving from peg 2 to peg 0.

The traditional 4-peg single stack puzzle, often called the Reve's Puzzle, is still an open problem. The name is due to British puzzle king Henry Dudeney, who posed it in his collection of Canterbury Puzzles [2]. There is a presumed minimum-move solution to the Reve's puzzle, but a proof of minimality is still lacking. So perhaps the most we can hope for with our Puzzle 4 is an algorithm that seems likely to be optimal, but without a proof of optimality.

3. The Hints

In this section we provide a specific hint for each puzzle.

Hint 1: We derive a lower bound for the number of moves needed for Puzzle 1, using the "small disk" induction principle popularized by the second author. Suppose we have an algorithm for two stacks of $n-1$ disks each, and we seek an algorithm for stacks of n disks. We can use the known algorithm for the $2(n-1)$

larger disks (disks numbered 2 through n in each stack) and intersperse moves of the smallest disks (the two disks numbered 1) when necessary.

The constraints of the puzzle force both the first move and the last move to use one of the smallest disks. Moreover, in any minimum-move solution, there must be at least one move of a smallest disk between any two moves of larger disks. To see this, note that when moving larger disks, the two smallest disks tie up two of the four pegs. We can move a larger disk between the other two pegs, but then we must move one of the smallest disks, since the only alternative is to reverse the move of the larger disk, which would clearly not be optimal. We conclude that the number of moves of the smallest disks is at least one more than the total number of moves of all larger disks.

Now let $M_1(n)$ be the number of moves made in a minimum-move algorithm for this puzzle. We know that the larger disks must collectively make at least $M_1(n-1)$ moves, since those moves must constitute a solution to the puzzle with stacks of size $n-1$. Moreover, from the previous paragraph we know that the smallest disks must then collectively make at least $M_1(n-1)+1$ moves, for a total of at least $2M_1(n-1)+1$ moves for $n \geq 2$, with $M_1(1) = 3$. This implies that $M_1(n) \geq 2^{n+1} - 1$.

Can you find an algorithm that attains this lower bound?

Hint 2: We can derive a lower bound for Puzzle 2 the same way we did for Puzzle 1. This time the constraints of the puzzle require that the first *two* moves and the last *two* moves must all use one of the smallest disks. And as before, there must be at least one move of a smallest disk between any two moves of the larger disks. So the number of moves of the smallest disks must be at least three more than the total number of moves of the larger disks.

If $M_2(n)$ is the number of moves made in a minimum-move algorithm, we know that the larger disks must make at least $M_2(n-1)$ moves, while the smallest disks must make at least $M_2(n-1) + 3$ moves, for a total of at least $2M_2(n-1) + 3$ moves for $n \geq 2$, with $M_2(1) = 3$. This implies that $M_2(n) \geq 3 \cdot 2^n - 3 = 3\,(2^n - 1)$ for this puzzle.

If this lower bound is attainable, you can use it to help generate an optimal algorithm. Start with a 3-move solution to the $n = 1$ puzzle. Consider these to be the moves of the larger disks of an $n = 2$ puzzle and see if you can find 6 moves of the smallest disks to combine with this moves, in a pattern `SSLSLSLSS`, where `S` indicates a move of one of the two smallest disks and `L` indicates a move of one of the two larger disks. If this works, you can try for $n = 3$ by using this 9-move sequence as the moves of the 4 larger disks and searching for 12 moves of the two smaller disks to combine with them.

Hint 3: The lower bound technique we used above does not work well for Puzzle 3, because it is possible to have consecutive moves of larger disks in an optimal solution, as well as consecutive moves of smallest disks. However, each stack is limited to three pegs, and must move from one to another. So $s(2^n - 1)$ is an obvious lower bound. On the other hand, the optimal algorithm for Puzzle 2 might suggest that this puzzle can be solved in $(s + 1)(2^n - 1)$ moves. Perhaps the real answer is somewhere between these two formulas.

Hint 4: Perhaps it is possible to construct an algorithm for Puzzle 4 based on the presumed minimum-move solution to the Reve's Puzzle. That solution, like that of the Tower of Hanoi, is most easily described recursively. For $n > 0$ first compute a value k. Recursively move the top $n-k$ disks of the tower from the initial peg to one

of the two temporary pegs; then move the bottom k disks to the destination peg, using the standard Tower of Hanoi algorithm; and then recursively move the top $n-k$ disks from their temporary location to the destination peg. The value k should be chosen so as to minimize the number of moves. It is known that $k = \{\sqrt{2n}\,\}$ will always do so, where $\{x\}$ denotes the integer nearest to x—see [6] for example.

Now try incorporating these ideas into the solution to Puzzle 2.

4. The Solutions

In this section we give pseudo-code for algorithms for each of the puzzles. For Puzzles 1, 2, and 3, these are provably minimum-move solutions; for Puzzles 4 we conjecture that our algorithm is optimal. Many of these algorithms will utilize a very general algorithm for the standard Tower of Hanoi, which we list here as the procedure `Hanoi`. Also, we use the notation `Move(disk, stack, from, to)` for an instruction to move a designated disk of a stack from one peg to another.

```
procedure Hanoi(bottom, top, stack, from, via, to)
  if(top<=bottom)
    Hanoi(bottom-1, top, stack, from, to, via);
    Move(bottom, stack, from, to);
    Hanoi(bottom-1, top, stack, via, from, to);
```

This algorithm moves disks numbered from `bottom` up to `top` (always 1 in the classical Tower puzzle) of the indicated stack between pegs, using peg `via` as the temporary peg. It is assumed that disks numbered less than `top`, if any, are on some peg other than the three listed.

Algorithm 1: We list `procedure Free` as a minimum-move algorithm for Puzzle 1, where `n` is the number of disks in each stack. We assume here that n is positive.

```
procedure Free(n)
  Hanoi(n-1, 1, 0, 0, 3, 1);
  Move(n, 0, 0, 3);
  Hanoi(n, 1, 1, 2, 3, 0);
  Move(n, 0, 3, 2);
  Hanoi(n-1, 1, 0, 1, 3, 2);
```

Clearly the number of moves made by this algorithm is $(2^{n-1}-1)+1+(2^n-1)+1+(2^{n-1}-1) = 2^{n+1}-1$, in agreement with the lower bound found earlier. We see that the disks of stack 0 make a total of 2^n moves while the disks of stack 1 make 2^n-1 moves. The total is just one move more than required for two separate Hanoi puzzles. Alternatively, it is the number of moves required for a Hanoi puzzle of $n+1$ disks.

Algorithm 2: In this section we present an algorithm for Puzzle 2 developed by the first author. Later we will give an alternative version crafted by the second author. We again assume that `n` is positive, and that all stack parameters are reduced modulo 4.

A key property of this algorithm can be proved by induction: at the end of each iteration of the top loop, disks from `1` down to `i+1` of the current stack are on their temporary peg while disks `i+2` down to `n` are still on their initial peg. For the

```
procedure Turtle(n)
  stack = 0;
  Move(1, 0, 0, 1);
  for i from 1 to n-1 do
    stack = 1-stack;
    Move(i, stack, 2*stack, 2*stack+2);
    Hanoi(i-1, 1, stack, 2*stack+1, 2*stack, 2*stack+2);
    Move(i+1, stack, 2*stack, 2*stack+1);
    Hanoi(i, 1, stack, 2*stack+2, 2*stack, 2*stack+1);
  stack = 1-stack;
  Move(n, stack, 2*stack, 2*stack+2);
  for i from n-1 down to 1 do
    stack = 1-stack;
    Hanoi(i, 1, stack, 2*stack+1, 2*stack+2, 2*stack);
    Move(i+1, stack, 2*stack+1, 2*stack+2);
    Hanoi(i-1, 1, stack, 2*stack, 2*stack+2, 2*stack+1);
    Move(i, stack, 2*stack, 2*stack+2);
  stack = 1-stack;
  Move(1, 0, 1, 2);
```

opposite stack, disks from `1` down to `i` are on their temporary peg and disks `i+1` down to `n` are still on their initial peg. An analogous property holds at the end of each iteration of the bottom loop: disks from `1` down to `i-1` of the current stack are on their temporary peg while disks `i` down to `n` are settled on their destination peg. For the opposite stack disks from `1` down to `i` are on their temporary pegs and disks `i+1` down to `n` are settled on their destination peg. It follows from these properties that all moves are legal and that the code correctly solves the puzzle.

Iteration i of the top loop makes $1 + (2^{i-1} - 1) + 1 + (2^i - 1) = 3 \cdot 2^{i-1}$ moves, which sums to $3(2^{n-1} - 1)$. The bottom loop makes an equal number of moves, and there are an addition 3 moves outside of the loops, for a total of $3(2^n - 1)$ moves, in agreement with our lower bound.

The move sequence is unique except for deciding which stack's disk 1 should move first.

Algorithm 3: For Puzzle 3 we present an algorithm built on a quartet of co-routines. In these routines we refer to the initial peg of stack i (peg $2i$) as its Home peg, the temporary peg of stack i (peg $2i + 1$) as its Run peg, and the destination peg of stack i (peg $2i + 2$) as its Away peg. The four letters in the name of each routine denote, in order, the peg where the indicated disks of stack 0 reside at the beginning of the routine; the peg where the indicated disks of stack 0 reside at the end of the routine; the peg where the indicated disks of all other stacks reside at the beginning of the routine; and the peg where the indicated disks of all other stacks reside at the end of the routine. For example, the routine `HAHR(n, s)` assumes that there are `s` stacks, and that at the start of the routine the top `n` disks of all stacks are on their Home pegs. At the end of the procedure the top disks of stack 0 will be on their Away peg while the top disks of all other stacks will be on their Run pegs. As before, we assume that each procedure does nothing if `n` is not positive, and that all stack parameters are reduced modulo $2s$.

```
procedure HAHR(n, s)
  HRHA(n-1, s);
  for i from 1 to s-1 do
    Move(n, i, (2*i), (2*i+1));
    Hanoi(n-1, 1, i, (2*i+2), (2*i), (2*i+1));
  Move(n, 0, 0, 2);
  Hanoi(n-1, 1, 0, 1, 0, 2);
procedure HRHA(n, s)
  HAHR(n-1, s);
  Move(n, 0, 0, 1);
  Hanoi(n-1, 1, 0, 2, 0, 1);
  for i from s-1 down to 1 do
    Move(n, i, (2*i), (2*i+2));
    Hanoi(n-1, 1, i, (2*i+1), (2*i), (2*i+2));
procedure HARA(n, s)
  Hanoi(n-1, 1, 0, 0, 2, 1);
  Move(n, 0, 0, 2);
  for i from 1 to s-1 do
    Hanoi(n-1, 1, i, (2*i+1), (2*i+2), (2*i));
    Move(n, i, (2*i+1), (2*i+2));
  RAHA(n-1, s);
procedure RAHA(n, s)
  for i from s-1 down to 1 do
    Hanoi(n-1, 1, i, (2*i), (2*i+2), (2*i+1));
    Move(n, i, (2*i), (2*i+2));
  Hanoi(n-1, 1, 0, 1, 2, 0);
  Move(n, 0, 1, 2);
  HARA(n-1, s);
```

All of these procedures make the same number of moves. There is a call to another one of the procedures, plus s moves and s calls to `Hanoi` in some order. If $H(n)$ is the number of moves made by any of these routines, then we have $H(n) = H(n-1)+s\cdot 2^{n-1}$, with $H(0) = 0$. The solution to this recurrence is $H(n) = s(2^n-1)$. These routines are clearly optimal in what they do, since each must carry out s separate Tower transformations.

Our solution for Puzzle 3 is given by our `HAHA` procedure. We again make our usual assumptions about `n` and the stack parameters. We see that this procedure

```
procedure HAHA(n, s)
  HAHR(n-1, s);
  Move(n, 0, 0, 1);
  for i from s-1 down to 2 do
    Move(n, i, (2*i), (2*i+2));
  Hanoi(n-1, 1, 0, 2, 1, 0);
  Move(n, 1, 2, 4);
  Move(n, 0, 1, 2);
  HARA(n-1, s);
```

makes two calls to earlier H-procedures, one call to Hanoi, and $s+1$ individual moves, for a total of $M_3(n,s) = s(2^n-1)+2^{n-1}$ moves.

To see that this is optimal, we observe that not all of the s disks of size n can move directly from their Home pegs to their Away pegs. At least one such disk (on stack 0 in our code) must move from its Home peg to its Run peg, and later from its Run peg to its Away peg. The top $n-1$ disks of this stack must at least first move from their Home peg to their Away peg before the first move of the largest disk, then back to their Home peg between moves of the largest disk, and then back to their Away peg after the final move of the largest disk. So this stack must make at least $3(2^{n-1}-1)+2 = (2^n-1)+2^{n-1}$ moves. The other $s-1$ stacks must make at least 2^n-1 moves each, for a sum of $M_3(n,s)$.

The procedure Turtle2 provides an alternative way to generate the optimal move sequence for Puzzle 2, using the H-procedures. We leave the analysis to the reader.

```
procedure Turtle2(n)
  HAHR(n-1, 2);
  Move(n, 0, 0, 1);
  Hanoi(n-1, 1, 0, 2, 0, 1);
  Move(n, 1, 2, 0);
  Hanoi(n-1, 1, 0, 1, 2, 0);
  Move(n, 0, 1, 2);
  HARA(n-1, 2);
```

Algorithm 4: Our algorithm for Puzzle 4, which we conjecture is optimal, uses the procedure Reve(n) that generates the moves of the presumed minimum-move solution to the Reve's Puzzle. In addition to our usual assumption that the procedure does nothing if n is not positive, we assume that k is always the integer nearest to $\sqrt{2n}$.

```
procedure Reve(n, stack, from, via, to, extra)
  Reve(n-k, stack, from, to, via, extra);
  Hanoi(n, n-k+1, stack, from, extra, to);
  Reve(n-k, stack, via, from, to, extra);
```

We will also need Reve versions of the H-procedures used in Puzzle 3. We list the procedure R_HAHR(n) as an example. Note that because there are only two stacks in this puzzle, there are no loops in the procedure.

```
procedure R_HRHA(n)
  R_HAHR(n-k);
  Hanoi(n, n-k+1, 0, 0, 4, 1);
  Reve(n-k, 0, 2, 0, 1, 4);
  Hanoi(n, n-k+1, 1, 2, 4, 0);
  Reve(n-k, 1, 3, 2, 0, 4);
```

Finally, we present the procedure R_HAHA as our solution to Puzzle 4, with the usual assumptions. It is a variation of the procedure Turtle2 above.

We very briefly sketch an analysis of this procedure. Let $R(n)$ be the number of moves made in the presumed minimum-move solution to the Reve's Puzzle, as

```
procedure R_HAHA(n)
  R_HAHR(n-k);
  Hanoi(n-1, n-k+1, 0, 0, 4, 1);
  Reve(n-k, 0, 2, 0, 1, 4);
  Move(n, 0, 0, 4);
  Hanoi(n, n-k+1, 1, 2, 4, 0);
  Move(n, 0, 4, 2);
  Reve(n-k, 0, 1, 2, 0, 4);
  Hanoi(n-1, n-k+1, 0, 1, 4, 2);
  R_HARA(n-k);
```

presented in the procedure `Reve`. We see from our procedure that $R(n) = 2R(n-k) + 2^k - 1$. This recurrence relation has the solution

$$R(n) = (k-1)2^k - \left(\frac{k(k+1)}{2} - n\right) 2^{k-1} + 1,$$

where as usual, k is the integer closest to $\sqrt{2n}$. Each procedure such as `R_HAHR` performs the Reve moves on each of two stacks, for a total of $2R(n)$ moves. In the procedure `R_HAHA`, the disks in stack 1 make a total of $R(n)$ moves, while the disks in stack 0 make a total of $4R(n-k) + 2^k$ moves. It can be shown that this last expression is equal to $2R(n-1) + 2$. Thus our solution to Puzzle 4 takes a total of $M_4(n) = R(n) + 2R(n-1) + 2$ moves.

References

[1] N. Claus (pseudonym for Édouard Lucas), *La Tour d'Hanoï: Jeu de Calcul*, Science et Nature **1** (1884), no. 8 (January 19), 127–128.

[2] Henry Ernest Dudeney, *The Reve's Puzzle*, The Canterbury Puzzles (and other curious problems), Thomas Nelson and Sons, Ltd., London, 1907.

[3] Édouard Lucas, *Nouveaux Jeux Scientifiques de M. Édouard Lucas*, La Nature, 17^{th} year, 2^{nd} semester (1889), no. 855 (October 5), 301–303.

[4] Steven Minsker, *The towers of Antwerpen problem*, Inform. Process. Lett. **38** (1991), 107–111.

[5] Henri de Parville, *Revue des Sciences*, Journal des Débats Politiques et Littéraires (December 27, 1883), 1–2.

[6] Paul K. Stockmeyer, *Variations on the four-post Tower of Hanoi puzzle*, Congr. Numer. **102** (1994), 3–12. (Proceedings of the 25^{th} Southeastern International Conference on Combinatorics, Graph Theory and Computing)

[7] Derick Wood, *The Towers of Brahma and Hanoi revisited*, J. Recreational Math. **14** (1981-1982), no. 1, 17–24.

The College of William and Mary
E-mail address: `stockmeyer@cs.wm.edu`

National University of Ireland, Maynooth
E-mail address: `Fred.Lunnon@may.ie`