

1 The Towers of Hanoi Puzzle

Good puzzles provide an excellent way to log in to the realm of abstract thought inhabited by mathematicians and other theorists. The best puzzles embody themes from this realm; the significance of such themes extends considerably beyond the puzzles themselves.

One such classic puzzle, the *Towers of Hanoi*, suggest two pairs of contrasting themes: recursion and iteration, unity and diversity. Apart from such serious considerations, the puzzle is fun and also provides the neophyte with a satisfying sense of confusion, hallmark of his or her slow entry into the realm of abstract thought.

The Towers of Hanoi consist of three vertical pegs set in a board. A number of disks, graded in size, are initially stacked on one of the pegs so that the smallest disk is uppermost, as shown in Figure 1. The aim of the puzzle is to transfer all the disks from the initial peg to one of the other two pegs. The disks are manipulated according to these two simple rules:

1. Move one disk at a time from one peg to another.
2. No disk may be placed on top of a smaller disk.

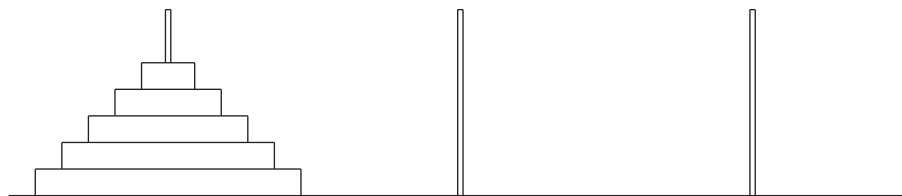


Figure 1: Initial position

The smallest disk must be moved first since it is the only one that is initially accessible. On the next turn there are two moves for the smallest disk (both pointless) and one move for the second-smallest-disk. It goes onto the unoccupied peg since it cannot be placed on top of the smallest disk (Rule 2). On the third turn it is not quite so obvious what to do: should the second disk be returned to the initial peg or should the first disk be moved again—and if so, onto what peg?

From this point on one is faced with a long succession of moves and with many opportunities for wrong choices. Even if all the right choices are made, $2^n - 1$ moves are needed (as we shall see below) to relocate a tower of n disks, one at a time, onto another peg. The surprisingly long time required to solve a puzzle made up of even a moderate number of disks is well illustrated by the following tale quoted from W. W. Rouse Ball's classic puzzle book, *Mathematical Recreations and Essays*:

In the great temple of Benares. . . beneath the dome which marks the centre of the world, rests a brass plate in which are fixed three

diamond needles, each a cubit high and as thick as the body of a bee. On one of these needles, at the creation, God placed sixty-four discs of pure gold, the largest disc resting on the brass plate and the others getting smaller and smaller up to the top one. This is the Tower of Bramah. Day and night unceasingly the priests transfer the discs from one diamond needle to another according to the fixed and immutable laws of Bramah, which require that the priest on duty must not move more than one disc at a time and that he must place this disc on a needle so that there is no smaller disc below it. When the sixty-four discs shall have been thus transferred from the needle on which at the creation God placed them to one of the other needles, tower, temple, and Brahmins alike will crumble into dust, and with a thunderclap the world will vanish.

That the world has not yet vanished attests to the extreme length of time it takes to solve the puzzle: even if the priests move one disk every second, it would take more than 500 billion years to relocate the initial tower of 64 disks!

At this point (and at no risk to the universe) the reader can involve himself or herself more directly by picking up five playing cards, for example the ace through five of hearts, and visualizing three spots on a table. Stack the cards on one of the spots, in order, so that the ace is on top. It is now possible to attempt a solution to the five-disk tower puzzle by moving one card at a time between two spots—but never place a card on one of lower value. Can you complete the relocation of the five-card tower before the end of the world? According to the formula $2^5 - 1$, the transfer should be possible in 31 moves.

Of course, if you're reading this, you're more likely to reach for an HP graphing calculator than a deck of playing cards. So, let's do just that.

2 Solving the Towers of Hanoi on the HP 39g

How does one go about solving a puzzle such as this one?

2.1 Divide-and-Conquer, or Recursion

One technique for solving this problem is a strategy commonly called “divide-and-conquer.” It consists of breaking a problem of size n into smaller problems in such a way that from solutions to smaller problems we can easily construct a solution to the entire problem.

The problem of moving the n smallest disks from A to C can be thought of as consisting of two subproblems of size $n - 1$. First move the $n - 1$ smallest disks from A to B , exposing the n^{th} smallest disk on A . Move that disk from A to C . Then move the $n - 1$ smallest disks from B to C .

So how do we move the $n - 1$ smallest disks from A to B ?

Well, we can do that by moving the $n - 2$ smallest disks from A to C , exposing the $(n - 1)^{th}$ smallest disk on A , moving that disk to B , then moving the $n - 2$ smallest disks from C to B .

See the pattern here?

Moving all n disks is accomplished by a recursive application of the method. As the n disks involved in the moves are smaller than any other disks, we need not concern ourselves with what lies below them on pegs A , B , or C . Although the actual movement of individual disks is not obvious, and hand simulation is hard because of the stacking of recursive calls, the algorithm is conceptually simple to understand, to prove correct and, we would like to think, to invent in the first place. It is probably the ease of discovery of divide-and-conquer algorithms that makes the technique so important.

Unfortunately, as simple as the algorithm may be, the HP-BASIC language as implemented on the HP-39g does not lend itself easily to recursive algorithms.

2.2 Another Algorithm—Iteration

There is also a non-recursive algorithm for solving the Towers of Hanoi puzzle.

Imagine the pegs arranged in a triangle. On odd-numbered moves, move the smallest disk one peg clockwise. On even-numbered moves make the only legal move not involving the smallest disk.

The above algorithm is concise, and correct. But in contrast to the earlier divide-and-conquer algorithm, it is hard to understand how it works, and hard to invent on the spur of the moment.

Programming this algorithm requires quite a bit of bookkeeping. The main data object is the matrix `M9`. The three rows of the matrix represent the three pegs. The entry in the first column points to the top disk on the peg. Each disk is represented by a number corresponding to its size, for convenience in drawing. For further convenience, the second column contains a virtual disk, larger than any real disk. According to the algorithm, this disk will never be moved. The table below shows the initial configuration for the case of five disks.

7	99	41	39	37	35	33
2	99					
2	99					

Because HP-BASIC doesn't provide for subroutines, we end up with two helper programs: one to draw/erase one disk and another to move one disk from peg A to peg B .

.HANOI.DRAW

```
DRAWLINE 43Z;3M9(Z,1);43Z+M9(Z,M9(Z,1));3M9(Z,1):
DRAWLINE 43Z;3M9(Z,1)+1;43Z+M9(Z,M9(Z,1));3M9(Z,1)+1
```

.HANOI.MOVE

```
A#Z:RUN ".HANOI.DRAW":
M9(B,1)+1#M9(B,1):
M9(A,M9(A,1))#M9(B,M9(B,1)):
B#Z:RUN ".HANOI.DRAW":
M9(A,1)-1#M9(A,1)
```

These subprograms are used by the main program, which prompts for the number of disks, sets up the initial configuration, then proceeds to solve it.

HANOI

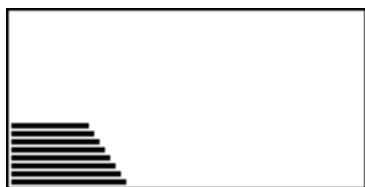
```

INPUT N;"TOWERS OF HANOI";"DISKS";"ENTER NUMBER";5:
INT(MAX(2,MIN(21,N)))N:
REDIM M9;3,N+2
2M9(2,1):99M9(2,2):
2M9(3,1):99M9(3,2):
99M9(1,2):
43XMIN:173XMAX:
9YMIN:72YMAX:
ERASE:
1Z:
FOR I=1 TO N:
  I+2M9(1,1):43-2IM9(1, I+2):
  RUN ".HANOI.DRAW"
END:
1+N MOD 2D:
1A:1+DDB:
RUN ".HANOI.MOVE":BBS:
DO
  1+(1==S)A:
  3-(3==S)B:
  IF M9(A,M9(A,1))>M9(B,M9(B,1))
  THEN AAI:BA:IAB:
  END:
  RUN ".HANOI.MOVE":
  SBA:1+(D+S-1)MOD 3B:
  RUN ".HANOI.MOVE":BBS:
UNTIL
  M9(3,1)==N+2:
END:
REDIM M9;1,1

```

Unfortunately, this program is ridiculously slow, requiring more than one second per move.

3 The Hanoi Aplet



The included aplet is written in System RPL, which avoids many of the shortcomings of HP-BASIC. It implements the first, recursive algorithm, though there is little way of telling that by observation from the outside.

Transfer the aplet to your HP 39g using whatever method you normally use for transferring aplets. Upon starting, it will prompt you for the number of disks. You can accept the default of five or provide your own number. It will then proceed to show the solution at a pace considerably more rapid than the HP-BASIC program, performing about 18 moves per second.

The aplet cannot handle the full sixty-four disks of the temple of Benares, being limited by the small screen to only twenty-one disks, but transferring even that number of disks will require more than a full day to solve.

