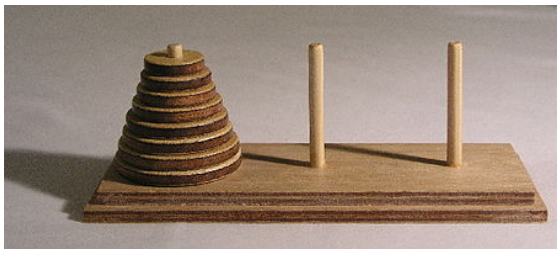
WIKIPEDIA

Tower of Hanoi

The **Tower of Hanoi** (also called the **Tower of Brahma** or **Lucas' Tower**[1] and sometimes pluralized as **Towers**) is a mathematical game or puzzle. It consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape.



A model set of the Tower of Hanoi (with 8 disks)

The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
3. No larger disk may be placed on top of a smaller disk.

An animated solution of the **Tower of Hanoi** puzzle for *T*(4, 3)

With 3 disks, the puzzle can be solved in 7 moves. The minimal number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where n is the number of disks.

Tower of Hanoi interactive display at the Universum museum in Mexico City

Contents

Origins

The puzzle was invented by the French mathematician Édouard Lucas in 1883. Numerous myths regarding the ancient and mystical nature of the puzzle popped up almost immediately.[2] There is a story about an Indian temple in Kashi Vishwanath which contains a large room with three time-worn posts in it, surrounded by 64 golden disks. Brahmin priests, acting out the command of an ancient prophecy, have been moving these disks in accordance with the immutable rules of Brahma since that time. The puzzle is therefore also known as the Tower of Brahma puzzle. According to the legend, when the last move of the puzzle is completed, the world will end.[3]

If the legend were true, and if the priests were able to move disks at a rate of one per second, using the smallest number of moves it would take them $2^{64} - 1$ seconds or roughly 585 billion years to finish,[4] which is about 42 times the current age of the universe.

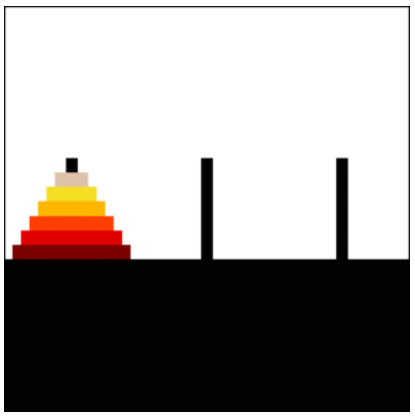
There are many variations on this legend. For instance, in some tellings the temple is a monastery, and the priests are monks. The temple or monastery may be said to be in different parts of the world—including Hanoi, Vietnam—and may be associated with any religion. In some versions other elements are introduced, such as the fact that the tower was created at the beginning of the world, or that the priests or monks may make only one move per day.

Solution

The puzzle can be played with any number of disks, although many toy versions have around 7 to 9 of them. The minimal number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where n is the number of disks.[5] This is precisely the nth Mersenne number.

Iterative solution

A simple solution for the toy puzzle is to alternate moves between the smallest piece and a non-smallest piece. When moving the smallest piece, always move it to the next position in the same direction (to the right if the starting number of pieces is even, to the left if the starting number of pieces is odd). If there is no tower position in the chosen direction, move the piece to the opposite end, but then continue to move in the correct direction. For example, if you started with three pieces, you would move the smallest piece to the opposite end, then continue in the left direction after that. When the turn is to move the non-smallest piece, there is only one legal move. Doing this will complete the puzzle in the fewest moves.[6]



Animation of an iterative algorithm solving 6-disk problem

Simpler statement of iterative solution

For an even number of disks:

- make the legal move between pegs A and B (in either direction),
- make the legal move between pegs A and C (in either direction),
- make the legal move between pegs B and C (in either direction),
- repeat until complete.

For an odd number of disks:

- make the legal move between pegs A and C (in either direction),
- make the legal move between pegs A and B (in either direction),
- make the legal move between pegs B and C (in either direction),
- repeat until complete.

In each case, a total of $2^n - 1$ moves are made.

Equivalent iterative solution

Another way to generate the unique optimal iterative solution:

Number the disks 1 through *n* (largest to smallest).

- If *n* is odd, the first move is from peg A to peg C.
- If *n* is even, the first move is from peg A to peg B.

Now, add these constraints:

- No odd disk may be placed directly on an odd disk.
- No even disk may be placed directly on an even disk.
- There will sometimes be two possible pegs: one will have disks, and the other will be empty. Place the disk on the non-empty peg.
- Never move a disk twice in succession.

Considering those constraints after the first move, there is only one legal move at every subsequent turn.

The sequence of these unique moves is an optimal solution to the problem equivalent to the iterative solution described above.[7]

Recursive solution

The key to solving a problem recursively is to recognize that it can be broken down into a collection of smaller sub-problems, to each of which *that same general solving procedure that we are seeking* applies, and the total solution is then found in some *simple* way from those sub-problems' solutions. Each of thus created sub-problems being "smaller" guarantees that the base case(s) will eventually be reached. Thence, for the Towers of Hanoi:

- label the pegs A, B, C,
- let *n* be the total number of disks,
- number the disks from 1 (smallest, topmost) to *n* (largest, bottom-most).

Assuming all *n* disks are distributed in valid arrangements among the pegs; assuming there are *m* top disks on a *source* peg, and all the rest of the disks are larger than *m*, so they can be safely ignored; to move *m* disks from a source peg to a *target* peg using a *spare* peg, without violating the rules:

1. Move *m* − 1 disks from the **source** to the **spare** peg, by *the same general solving procedure*. Rules are not violated, by assumption. This leaves the disk *m* as a top disk on the source peg.
2. Move the disk *m* from the **source** to the **target** peg, which is guaranteed to be a valid move, by the assumptions — *a simple step*.
3. Move the *m* − 1 disks that we have just placed on the spare, from the **spare** to the **target** peg by *the same general solving procedure*, so they are placed on top of the disk *m* without violating the rules.
4. The base case being to move *0* disks (in steps 1 and 3), that is, do nothing – which obviously doesn't violate the rules.

The full Tower of Hanoi solution then consists of moving n disks from the source peg A to the target peg C, using B as the spare peg.

This approach can be given a rigorous mathematical proof with mathematical induction and is often used as an example of recursion when teaching programming.

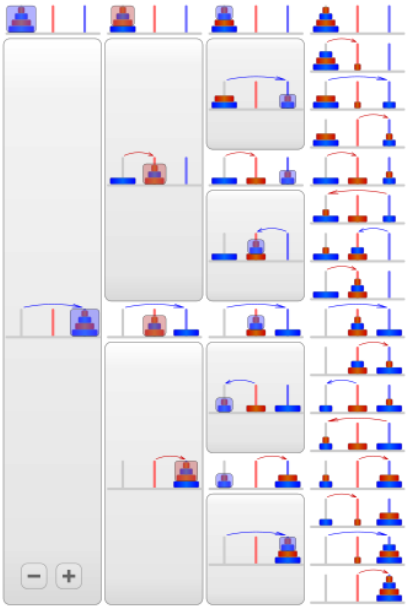


Illustration of a recursive solution for the Towers of Hanoi puzzle with 4 disks

Logical analysis of the recursive solution

As in many mathematical puzzles, finding a solution is made easier by solving a slightly more general problem: how to move a tower of h (height) disks from a starting peg f = **A** (from) onto a destination peg t = **C** (to), **B** being the remaining third peg and assuming $t \neq f$. First, observe that the problem is symmetric for permutations of the names of the pegs (symmetric group S_3). If a solution is known moving from peg **A** to peg **C**, then, by renaming the pegs, the same solution can be used for every other choice of starting and destination peg. If there is only one disk (or even none at all), the problem is trivial. If $h = 1$, then simply move the disk from peg **A** to peg **C**. If $h > 1$, then somewhere along the sequence of moves, the largest disk must be moved from peg **A** to another peg, preferably to peg **C**. The only situation that allows this move is when all smaller $h - 1$ disks are on peg **B**. Hence, first all $h - 1$ smaller disks must go from **A** to **B**. Then move the largest disk and finally move the $h - 1$ smaller disks from peg **B** to peg **C**. The presence of the largest disk does not impede any move of the $h - 1$ smaller disks and can be temporarily ignored. Now the problem is reduced to moving $h - 1$ disks from one peg to another one, first from **A** to **B** and subsequently from **B** to **C**, but the same method can be used both times by renaming the pegs. The same strategy can be used to reduce the $h - 1$ problem to $h - 2$, $h - 3$, and so on until only one disk is left. This is called recursion. This algorithm can be schematized as follows.

Identify the disks in order of increasing size by the natural numbers from 0 up to but not including h. Hence disk 0 is the smallest one, and disk $h - 1$ the largest one.

The following is a procedure for moving a tower of h disks from a peg **A** onto a peg **C**, with **B** being the remaining third peg:

1. If $h > 1$, then first use this procedure to move the $h - 1$ smaller disks from peg **A** to peg **B**.
2. Now the largest disk, i.e. disk h can be moved from peg **A** to peg **C**.
3. If $h > 1$, then again use this procedure to move the $h - 1$ smaller disks from peg **B** to peg **C**.

By means of mathematical induction, it is easily proven that the above procedure requires the minimal number of moves possible, and that the produced solution is the only one with this minimal number of moves. Using recurrence relations, the exact number of moves that this solution requires can be calculated by: $\mathbf{2^h - 1}$. This result is obtained by noting that steps 1 and 3 take T_{h-1} moves, and step 2 takes one move, giving $T_h = 2T_{h-1} + 1$.

Recursive implementation

The following Python code highlights an essential function of the recursive solution, which may be otherwise misunderstood or overlooked. That is, with every level of recursion, the first recursive call inverts the *target* and *auxiliary* stacks, while in the second recursive call the *source* and *auxiliary* stacks are inverted.

```
A = [3, 2, 1]
B = []
C = []

def move(n, source, target, auxiliary):
```

```
    if n > 0:
        # Move n - 1 disks from source to auxiliary, so they are out of the way
        move(n - 1, source, auxiliary, target)

        # Move the nth disk from source to target
        target.append(source.pop())

        # Display our progress
        print(A, B, C, '###############', sep='\n')

        # Move the n - 1 disks that we left on auxiliary onto target
        move(n - 1, auxiliary, target, source)

# Initiate call from source A to target C with auxiliary B
move(3, A, C, B)
```

The following code implements more recursive functions for a text-based animation:

```
import time

A = [i for i in range(5, 0, -1)]
height = len(A) - 1  # Stable height value for animation
B = []
C = []

def move(n, source, target, auxiliary):
    if n > 0:
        # Move n - 1 disks from source to auxiliary, so they are out of the way
        move(n - 1, source, auxiliary, target)

        # Move the nth disk from source to target
        target.append(source.pop())

        # Display our progress using a recursive function to draw it out
        draw_disks(A, B, C, height)
        print("")  # Provide spacing
        time.sleep(0.3)  # Pause for a moment to animate

        # Move the n - 1 disks that we left on auxiliary onto target
        move(n - 1, auxiliary, target, source)

def draw_disks(A, B, C, position, width=2 * int(max(A))):
    # width parameter defaults to double of the largest sized disk in the initial tower.
    if position >= 0:
        # If A has a value in the list at the given position, create a disk at its position (height)
        valueInA = " " if position >= len(A) else create_disk(A[position])
        # Same for B and C
        valueInB = " " if position >= len(B) else create_disk(B[position])
        valueInC = " " if position >= len(C) else create_disk(C[position])

        # Print each row
        print("{0:^{width}}{1:^{width}}{2:^{width}}".format(valueInA, valueInB, valueInC, width=width))

        # Recursively call this method again to the next position (height)
        draw_disks(A, B, C, position - 1, width)
    else:
        # When done with recursive, print column labels
        print("{0:^{width}}{1:^{width}}{2:^{width}}".format("A", "B", "C", width=width))

def create_disk(size):
    """Simple recursive method to create a slanted disk."""
    if size == 1:
        return "/\\"
    else:
        return "/" + create_disk(size - 1) + "\\"

# Initiate call from source A to target C with auxiliary B
move(len(A), A, C, B)
```

Non-recursive solution

The list of moves for a tower being carried from one peg onto another one, as produced by the recursive algorithm, has many regularities. When counting the moves starting from 1, the ordinal of the disk to be moved during move *m* is the number of times *m* can be divided by 2. Hence every odd move involves the smallest disk. It

can also be observed that the smallest disk traverses the pegs *f*, *t*, *r*, *f*, *t*, *r*, etc. for odd height of the tower and traverses the pegs *f*, *r*, *t*, *f*, *r*, *t*, etc. for even height of the tower. This provides the following algorithm, which is easier, carried out by hand, than the recursive algorithm.

In alternate moves:

- Move the smallest disk to the peg it has not recently come from.
- Move another disk legally (there will be only one possibility).

For the very first move, the smallest disk goes to peg *t* if *h* is odd and to peg *r* if *h* is even.

Also observe that:

- Disks whose ordinals have even parity move in the same sense as the smallest disk.
- Disks whose ordinals have odd parity move in opposite sense.
- If *h* is even, the remaining third peg during successive moves is *t*, *r*, *f*, *t*, *r*, *f*, etc.
- If *h* is odd, the remaining third peg during successive moves is *r*, *t*, *f*, *r*, *t*, *f*, etc.

With this knowledge, a set of disks in the middle of an optimal solution can be recovered with no more state information than the positions of each disk:

- Call the moves detailed above a disk's "natural" move.
- Examine the smallest top disk that is not disk 0, and note what its only (legal) move would be: if there is no such disk, then we are either at the first or last move.
- If that move is the disk's "natural" move, then the disk has not been moved since the last disk 0 move, and that move should be taken.
- If that move is not the disk's "natural" move, then move disk 0.

Binary solution

Disk positions may be determined more directly from the binary (base-2) representation of the move number (the initial state being move #0, with all digits 0, and the final state being with all digits 1), using the following rules:

- There is one binary digit (bit) for each disk.
- The most significant (leftmost) bit represents the largest disk. A value of 0 indicates that the largest disk is on the initial peg, while a 1 indicates that it's on the final peg (right peg if number of disks is odd and middle peg otherwise).
- The bitstring is read from left to right, and each bit can be used to determine the location of the corresponding disk.
- A bit with the same value as the previous one means that the corresponding disk is stacked on top the previous disk on the same peg.

  (That is to say: a straight sequence of 1s or 0s means that the corresponding disks are all on the same peg.)

- A bit with a different value to the previous one means that the corresponding disk is one position to the left or right of the previous one. Whether it is left or right is determined by this rule:
  - Assume that the initial peg is on the left.
  - Also assume "wrapping" – so the right peg counts as one peg "left" of the left peg, and vice versa.
  - Let *n* be the number of greater disks that are located on the same peg as their first greater disk and add 1 if the largest disk is on the left peg. If *n* is even, the disk is located one peg to the right, if *n* is odd, the disk located one peg to the left (in case of even number of disks and vice versa otherwise).

For example, in an 8-disk Hanoi:

- Move 0 = 00000000.
  - The largest disk is 0, so it is on the left (initial) peg.
  - All other disks are 0 as well, so they are stacked on top of it. Hence all disks are on the initial peg.
- Move $2^8 - 1$ = 11111111.
  - The largest disk is 1, so it is on the middle (final) peg.
  - All other disks are 1 as well, so they are stacked on top of it. Hence all disks are on the final peg and the puzzle is complete.
- Move 216_{10} = 11011000.
  - The largest disk is 1, so it is on the middle (final) peg.
  - Disk two is also 1, so it is stacked on top of it, on the middle peg.
  - Disk three is 0, so it is on another peg. Since *n* is odd (*n* = 1), it is one peg to the left, i.e. on the left peg.
  - Disk four is 1, so it is on another peg. Since *n* is odd (*n* = 1), it is one peg to the left, i.e. on the right peg.
  - Disk five is also 1, so it is stacked on top of it, on the right peg.
  - Disk six is 0, so it is on another peg. Since *n* is even (*n* = 2), the disk is one peg to the right, i.e. on the left peg.
  - Disks seven and eight are also 0, so they are stacked on top of it, on the left peg.

The source and destination pegs for the *m*th move can also be found elegantly from the binary representation of *m* using bitwise operations. To use the syntax of the C programming language, move *m* is from peg `(m & m - 1) % 3` to peg `((m | m - 1) + 1) % 3`, where the disks begin on peg 0 and finish on peg 1 or 2 according as whether the number of disks is even or odd. Another formulation is from peg `(m - (m & -m)) % 3` to peg `(m + (m & -m)) % 3`.

Furthermore, the disk to be moved is determined by the number of times the move count (*m*) can be divided by 2 (i.e. the number of zero bits at the right), counting the first move as 1 and identifying the disks by the numbers 0, 1, 2 etc. in order of increasing size. This permits a very fast non-recursive computer implementation to find the positions of the disks after m moves without reference to any previous move or distribution of disks.

The operation, which counts the number of consecutive zeros at the end of a binary number, gives a simple solution to the problem: the disks are numbered from zero, and at move *m*, disk number count trailing zeros is moved the minimal possible distance to the right (circling back around to the left as needed).[8]

Gray-code solution

The binary numeral system of Gray codes gives an alternative way of solving the puzzle. In the Gray system, numbers are expressed in a binary combination of 0s and 1s, but rather than being a standard positional numeral system, Gray code operates on the premise that each value differs from its predecessor by only one (and exactly one) bit changed.

If one counts in Gray code of a bit size equal to the number of disks in a particular Tower of Hanoi, begins at zero, and counts up, then the bit changed each move corresponds to the disk to move, where the least-significant bit is the smallest disk, and the most-significant bit is the largest.

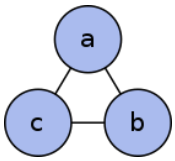
> Counting moves from 1 and identifying the disks by numbers starting from 0 in order of increasing size, the ordinal of the disk to be moved during move *m* is the number of times *m* can be divided by 2.

This technique identifies which disk to move, but not where to move it to. For the smallest disk there are always two possibilities. For the other disks there is always one possibility, except when all disks are on the same peg, but in that case either it is the smallest disk that must be moved or the objective has already been achieved. Luckily, there is a rule that does say where to move the smallest disk to. Let *f* be the starting peg, *t* the destination peg, and *r* the remaining third peg. If the number of disks is odd, the smallest disk cycles along the pegs in the order $f \to t \to r \to f \to t \to r$, etc. If the number of disks is even, this must be reversed: $f \to r \to t \to f \to r \to t$, etc.[9]

The position of the bit change in the Gray code solution gives the size of the disk moved at each step: 1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1, ... (sequence A001511 in the OEIS),[10] a sequence also known as the ruler function, or one more than the power of 2 within the move number. In the Wolfram Language, `IntegerExponent[Range[2^8 - 1], 2] + 1` gives moves for the 8-disk puzzle.

Graphical representation

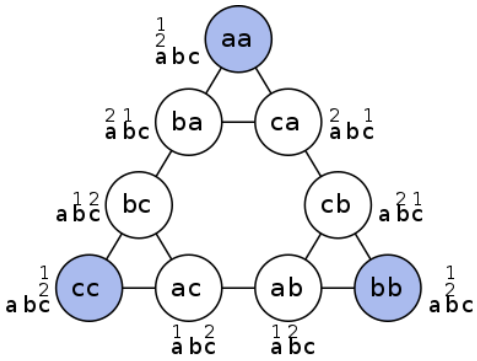
The game can be represented by an undirected graph, the nodes representing distributions of disks and the edges representing moves. For one disk, the graph is a triangle:



The graph for two disks is three triangles connected to form the corners of a larger triangle.

A second letter is added to represent the larger disk. Clearly it cannot initially be moved.

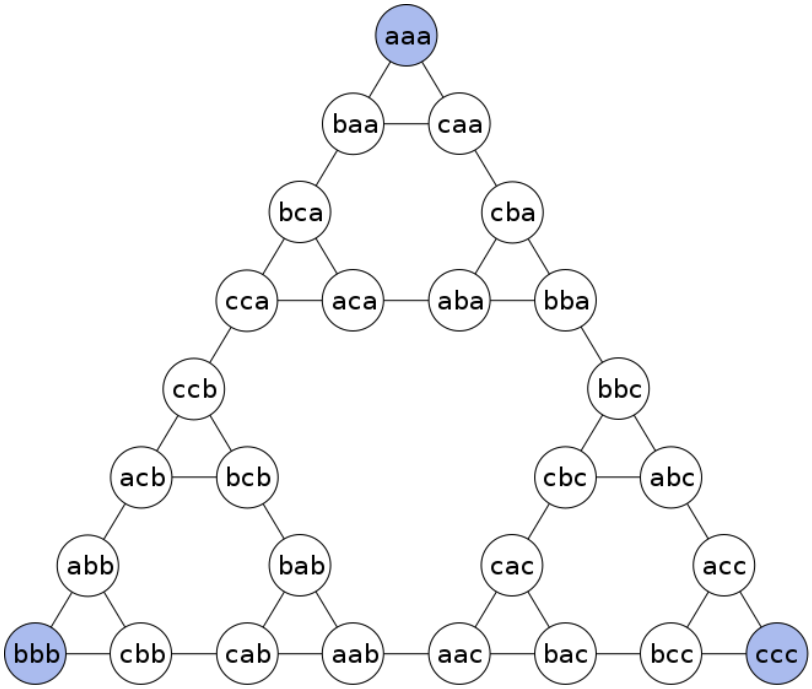
The topmost small triangle now represents the one-move possibilities with two disks:



The nodes at the vertices of the outermost triangle represent distributions with all disks on the same peg.

For h + 1 disks, take the graph of h disks and replace each small triangle with the graph for two disks.

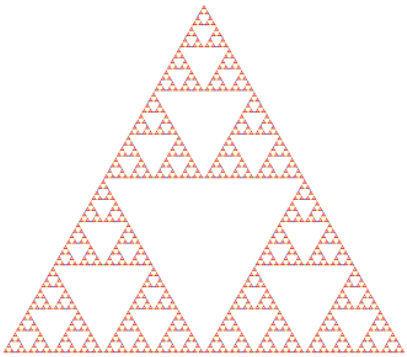
For three disks the graph is:



- call the pegs a, b and c
- list disk positions from left to right in order of increasing size

The sides of the outermost triangle represent the shortest ways of moving a tower from one peg to another one. The edge in the middle of the sides of the largest triangle represents a move of the largest disk. The edge in the middle of the sides of each next smaller triangle represents a move of each next smaller disk. The sides of the smallest triangles represent moves of the smallest disk.

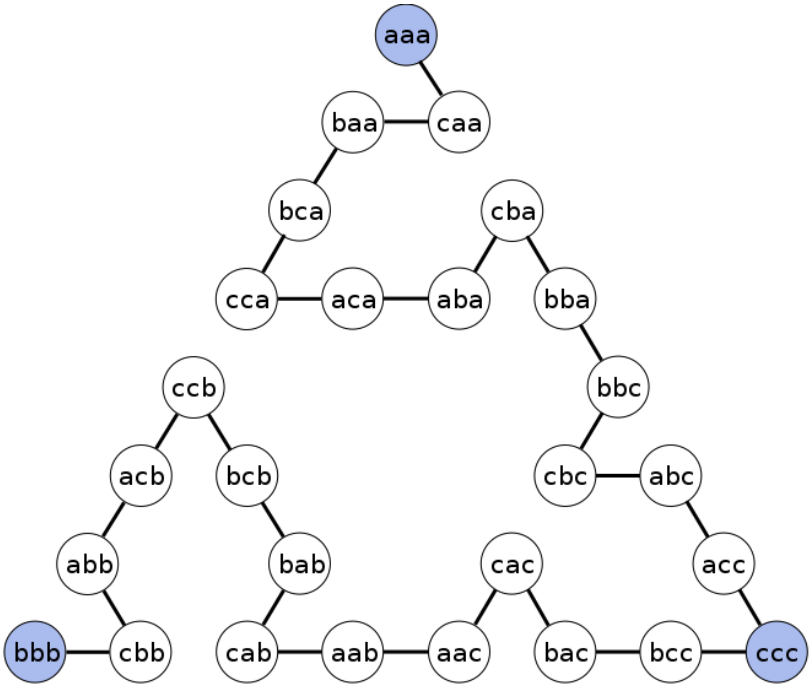
In general, for a puzzle with n disks, there are 3^n nodes in the graph; every node has three edges to other nodes, except the three corner nodes, which have two: it is always possible to move the smallest disk to one of the two other pegs, and it is possible to move one disk between those two pegs *except* in the situation where all disks are stacked on one peg. The corner nodes represent the three cases where all the disks are stacked on one peg. The diagram for $n + 1$ disks is obtained by taking three copies of the n-disk diagram—each one representing all the states and moves of the smaller disks for one particular position of the new largest disk—and joining them at the corners with three new edges, representing the only three opportunities to move the largest disk. The resulting figure thus has 3^{n+1} nodes and still has three corners remaining with only two edges.



The game graph of level 7 shows the relatedness to the Sierpiński triangle.

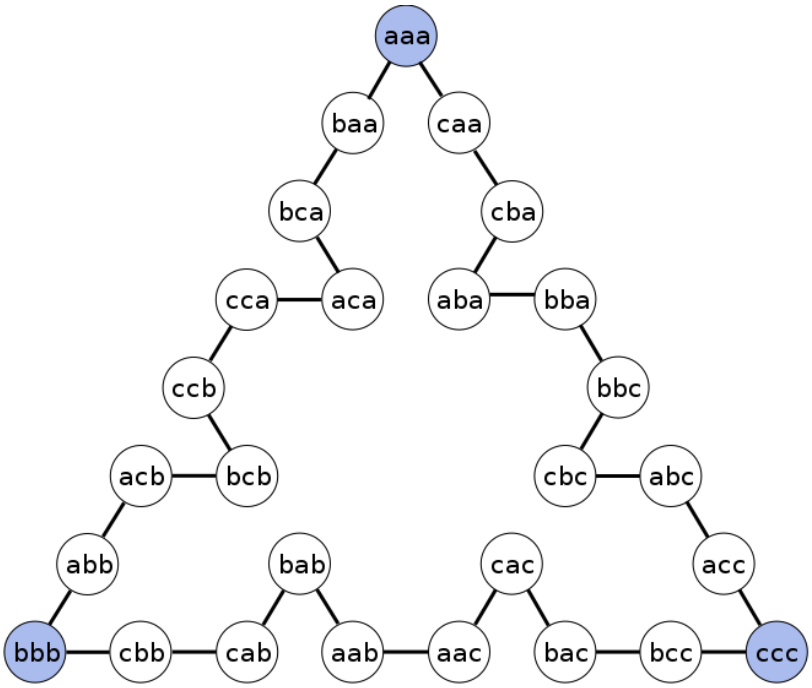
As more disks are added, the graph representation of the game will resemble a fractal figure, the Sierpiński triangle. It is clear that the great majority of positions in the puzzle will never be reached when using the shortest possible solution; indeed, if the priests of the legend are using the longest possible solution (without re-visiting any position), it will take them $3^{64} - 1$ moves, or more than 10^{23} years.

The longest non-repetitive way for three disks can be visualized by erasing the unused edges:



Incidentally, this longest non-repetitive path can be obtained by forbidding all moves from a to b.

The Hamiltonian cycle for three disks is:



The graphs clearly show that:

- From every arbitrary distribution of disks, there is exactly one shortest way to move all disks onto one of the three pegs.
- Between every pair of arbitrary distributions of disks there are one or two different shortest paths.
- From every arbitrary distribution of disks, there are one or two different longest non selfcrossing paths to move all disks to one of the three pegs.
- Between every pair of arbitrary distributions of disks there are one or two different longest non self-crossing paths.
- Let N_h be the number of non-self-crossing paths for moving a tower of h disks from one peg to another one. Then:

- $N_1 = 2$
- $N_{h+1} = (N_h)^2 + (N_h)^3$

This gives N_h to be 2, 12, 1872, 6563711232, ... (sequence A125295 in the OEIS)

Variations

Adjacent pegs

If all moves must be between adjacent pegs (i.e. given pegs A, B, C, one cannot move directly between pegs A and C), then moving a stack of n disks from peg A to peg C takes $3^n - 1$ moves. The solution uses all 3^n valid positions, always taking the unique move that does not undo the previous move. The position with all disks at peg B is reached halfway, i.e. after $(3^n - 1) / 2$ moves.

Cyclic Hanoi

In Cyclic Hanoi, we are given three pegs (A, B, C), which are arranged as a circle with the clockwise and the counterclockwise directions being defined as A – B – C – A and A – C – B – A respectively. The moving direction of the disk must be clockwise.[11] It suffices to represent the sequence of disks to be moved. The solution can be found using two mutually recursive procedures:

To move *n* disks **counterclockwise** to the neighbouring target peg:

1. move *n* − 1 disks **counterclockwise** to the target peg
2. move disk #*n* one step clockwise
3. move *n* − 1 disks **clockwise** to the start peg
4. move disk #*n* one step clockwise
5. move *n* − 1 disks **counterclockwise** to the target peg

To move *n* disks **clockwise** to the neighbouring target peg:

1. move *n* − 1 disks **counterclockwise** to a spare peg
2. move disk #*n* one step clockwise
3. move *n* − 1 disks **counterclockwise** to the target peg

Let C(n) and A(n) represent moving n disks clockwise and counterclockwise, then we can write down both formulas:

```
      C(n) = A(n-1) n A(n-1)    and      A(n) = A(n-1) n C(n-1) n A(n-1).
```

```
 Thus  C(1) = 1                 and      A(1) = 1 1,
       C(2) = 1 1 2 1 1         and      A(2) = 1 1 2 1 2 1 1.
```

The solution for the Cyclic Hanoi has some interesting properties:

1)The move-patterns of transferring a tower of disks from a peg to another peg are symmetric with respect to the center points.

2)The smallest disk is the first and last disk to move.

3)Groups of the smallest disk moves alternate with single moves of other disks.

4)The number of disks moves specified by C(n) and A(n) are minimal.

With four pegs and beyond

Although the three-peg version has a simple recursive solution long been known, the optimal solution for the Tower of Hanoi problem with four pegs (called Reve's puzzle) was not verified until 2014, by Bousch.[12]

However, in case of four or more pegs, the Frame–Stewart algorithm is known without proof of optimality since 1941.[13]

For the formal derivation of the exact number of minimal moves required to solve the problem by applying the Frame–Stewart algorithm (and other equivalent methods), see the following paper.[14]

For other variants of the four-peg Tower of Hanoi problem, see Paul Stockmeyer's survey paper.[15]

Frame–Stewart algorithm

The Frame–Stewart algorithm is described below:

- Let n be the number of disks.
- Let r be the number of pegs.
- Define $T(n, r)$ to be the minimum number of moves required to transfer n disks using r pegs.

The algorithm can be described recursively:

1. For some k, $1 \leq k < n$, transfer the top k disks to a single peg other than the start or destination pegs, taking $T(k, r)$ moves.
2. Without disturbing the peg that now contains the top k disks, transfer the remaining $n - k$ disks to the destination peg, using only the remaining $r - 1$ pegs, taking $T(n - k, r - 1)$ moves.
3. Finally, transfer the top k disks to the destination peg, taking $T(k, r)$ moves.

The entire process takes $2T(k, r) + T(n - k, r - 1)$ moves. Therefore, the count k should be picked for which this quantity is minimum. In the 4-peg case, the optimal k equals $n - \lfloor \sqrt{2n + 1} \rceil + 1$, where $\lfloor \cdot \rceil$ is the nearest integer function.[16] For example, in the UPenn CIS 194 course on Haskell, the first assignment page[17] lists the optimal solution for the 15-disk and 4-peg case as 129 steps, which is obtained for the above value of *k*.

This algorithm (with the above choice for k) is presumed to be optimal for any number of pegs; its number of moves is $2^{\Theta(n^{1/(r-2)})}$ (for fixed *r*).

General shortest paths and the number 466/885

A curious generalization of the original goal of the puzzle is to start from a given configuration of the disks where all disks are not necessarily on the same peg, and to arrive in a minimal number of moves at another given configuration. In general it can be quite difficult to compute a shortest sequence of moves to solve this problem. A solution was proposed by Andreas Hinz, and is based on the observation that in a shortest sequence of moves, the largest disk that needs to be moved (obviously one may ignore all of the largest disks that will occupy the same peg in both the initial and final configurations) will move either exactly once or exactly twice.

The mathematics related to this generalized problem becomes even more interesting when one considers the **average** number of moves in a shortest sequence of moves between two initial and final disk configurations that are chosen at random. Hinz and Chan Tat-Hung independently discovered[18][19] (see also [20]:Chapter 1, p. 14) that the average number of moves in an n-disk Tower is given by the following exact formula:

$$\frac{466}{885}\cdot 2^n - \frac{1}{3} - \frac{3}{5}\cdot\left(\frac{1}{3}\right)^n + \left(\frac{12}{59} + \frac{18}{1003}\sqrt{17}\right)\left(\frac{5+\sqrt{17}}{18}\right)^n + \left(\frac{12}{59} - \frac{18}{1003}\sqrt{17}\right)\left(\frac{5-\sqrt{17}}{18}\right)^n.$$

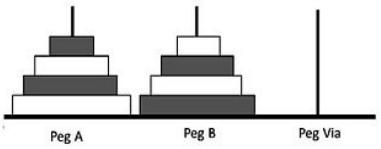
For large enough n, only the first and second terms do not converge to zero, so we get an asymptotic expression: $466/885 \cdot 2^n - 1/3 + o(1)$, as $n \to \infty$. Thus intuitively, we could interpret the fraction of $466/885 \approx 52.6\%$ as representing the ratio of the labor one has to perform when going from a randomly chosen configuration to another randomly chosen configuration, relative to the difficulty of having to cross the "most difficult" path of length $2^n - 1$ which involves moving all the disks from one peg to another. An alternative explanation for the appearance of the constant 466/885, as well as a new and somewhat improved algorithm for computing the shortest path, was given by Romik.[21]

Magnetic Hanoi

In Magnetic Tower of Hanoi, each disk has two distinct sides North and South (typically colored "red" and "blue"). Disks must not be placed with the similar poles together—magnets in each disk prevent this illegal move. Also, each disk must be flipped as it is moved.

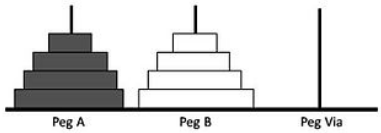
Bicolor Towers of Hanoi

This variation of the famous Tower of Hanoi puzzle was offered to grade 3–6 students at *2ème Championnat de France des Jeux Mathématiques et Logiques* held in July 1988.[22]



Initial configuration of bicolor Towers of Hanoi (n=4)

The rules of the puzzle are essentially the same: disks are transferred between pegs one at a time. At no time may a bigger disk be placed on top of a smaller one. The difference is that now for every size there are two disks: one black and one white. Also, there are now two towers of disks of alternating colors. The goal of the puzzle is to make the towers monochrome (same color). The biggest disks at the bottom of the towers are assumed to swap positions.



Final configuration of bicolor Towers of Hanoi (n=4)

Tower of Hanoy

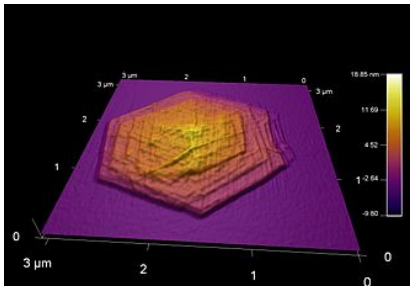
A variation of the puzzle has been adapted as a solitaire game with nine playing cards under the name Tower of Hanoy. It is not known whether the altered spelling of the original name is deliberate or accidental.

Applications

The Tower of Hanoi is frequently used in psychological research on problem solving. There also exists a variant of this task called Tower of London for neuropsychological diagnosis and treatment of executive functions.

Zhang and Norman[24] used several isomorphic (equivalent) representations of the game to study the impact of representational effect in task design. They demonstrated an impact on user performance by changing the way that the rules of the game are represented, using variations in the physical design of the game components. This knowledge has impacted on the development of the TURF framework[25] for the representation of human–computer interaction.

The Tower of Hanoi is also used as a backup rotation scheme when performing computer data backups where multiple tapes/media are involved.



3D AFM topographic image of multilayered palladium nanosheet on silicon wafer, with Tower of Hanoi-like structure.[23]

As mentioned above, the Tower of Hanoi is popular for teaching recursive algorithms to beginning programming students. A pictorial version of this puzzle is programmed into the emacs editor, accessed by typing M-x hanoi. There is also a sample algorithm written in Prolog.

The Tower of Hanoi is also used as a test by neuropsychologists trying to evaluate frontal lobe deficits.[26]

In 2010, researchers published the results of an experiment that found that the ant species *Linepithema humile* were successfully able to solve the 3-disk version of the Tower of Hanoi problem through non-linear dynamics and pheromone signals.[27]

In 2014, scientists synthesized multilayered palladium nanosheets with a Tower of Hanoi like structure.[23]

In popular culture

In the science fiction story "Now Inhale", by Eric Frank Russell,[28] a human is held prisoner on a planet where the local custom is to make the prisoner play a game until it is won or lost before his execution. The protagonist knows that a rescue ship might take a year or more to arrive, so he chooses to play Towers of Hanoi with 64 disks. (This story makes reference to the legend about the Buddhist monks playing the game until the end of the world.)

In the 1966 *Doctor Who* story *The Celestial Toymaker*, the eponymous villain forces the Doctor to play a ten-piece 1,023-move Tower of Hanoi game entitled The Trilogic Game with the pieces forming a pyramid shape when stacked.

In 2007, the concept of the Towers Of Hanoi problem was used in *Professor Layton and the Diabolical Box* in puzzles 6, 83, and 84, but the disks had been changed to pancakes. The puzzle was based around a dilemma where the chef of a restaurant had to move a pile of pancakes from one plate to the other with the basic principles of the original puzzle (i.e. three plates that the pancakes could be moved onto, not being able to put a larger pancake onto a smaller one, etc.)

In the film *Rise of the Planet of the Apes* (2011), this puzzle, called in the film the "Lucas Tower", is used as a test to study the intelligence of apes.

The puzzle is featured regularly in adventure and puzzle games. Since it is easy to implement, and easily recognised, it is well-suited to use as a puzzle in a larger graphical game (e.g. *Star Wars: Knights of the Old Republic* and *Mass Effect*).[29] Some implementations use straight disks, but others disguise the puzzle in some other form. There is an arcade version by Sega.[30]

A 15-disk version of the puzzle appears in the game *Sunless Sea* as a lock to a tomb. The player has the option to click through each move of the puzzle in order to solve it, but the game notes that it will take 32767 moves to complete. If an especially dedicated player does click through to the end of the puzzle, it is revealed that completing the puzzle does not unlock the door.

In *Yu-Gi-Oh! VRAINS*, a hacking group called "Knight of Hanoi" create a structure named "Tower of Hanoi" within the eponymous VRAINS virtual reality network.

This was first used as a challenge in survivor Thailand in 2002 but rather than rings, the pieces were made to resemble a temple. Sook Jai threw the challenge to get rid of Jed even though Shii-Ann knew full well how to complete the puzzle. The problem is featured as part of a reward challenge in a 2011 episode of the American version of the *Survivor* TV series. Both players (Ozzy Lusth and Benjamin "Coach" Wade) struggled to understand how to solve the puzzle and are aided by their fellow tribe members.

See also

- ABACABA pattern
- Backup rotation scheme, a TOH application

- Baguenaudier
- Recursion (computer science)

Notes

1. Hofstadter, Douglas R. (1985). *Metamagical Themas : Questing for the Essence of Mind and Pattern* (https://archive.org/details/metamagicalthema0000hofs). New York: Basic Books. ISBN 978-0-465-04540-2.
2. Hinz, Andreas M.; Klavžar, Sandi; Milutinović, Uroš; Petr, Ciril (2013-01-31). *The Tower of Hanoi – Myths and Maths*. ISBN 978-3034802369.
3. Spitznagel, Edward L. (1971). *Selected topics in mathematics* (https://archive.org/details/selectedtopicsin0000spit/page/137). Holt, Rinehart and Winston. p. 137 (https://archive.org/details/selectedtopicsin0000spit/page/137). ISBN 978-0-03-084693-9.
4. Moscovich, Ivan (2001). *1000 playthinks: puzzles, paradoxes, illusions & games*. Workman. ISBN 978-0-7611-1826-8.
5. Petković, Miodrag (2009). *Famous Puzzles of Great Mathematicians*. AMS Bookstore. p. 197. ISBN 978-0-8218-4814-2.
6. Troshkin, M. "Doomsday Comes: A Nonrecursive Analysis of the Recursive Towers-of-Hanoi Problem". *Focus* (in Russian). **95** (2): 10–14.
7. Mayer, Herbert; Perkins, Don (1984). "Towers of Hanoi Revisited". *SIGPLAN Notices*. **19** (2): 80–84. doi:10.1145/948566.948573 (https://doi.org/10.1145%2F948566.948573). S2CID 2304761 (https://api.semanticscholar.org/CorpusID:2304761).
8. Warren, Henry S. (2003). "Section 5-4: Counting Trailing 0's.". *Hacker's delight* (1st ed.). Boston MA: Addison-Wesley. ISBN 978-0-201-91465-8.
9. Miller, Charles D. (2000). "Ch. 4: Binary Numbers and the Standard Gray Code" (https://web.archive.org/web/20040821062630/http://occawlonline.pearsoned.com/bookbind/pubbooks/miller2_awl/chapter4/essay1/deluxe-content.html#tower). *Mathematical Ideas* (http://occawlonline.pearsoned.com/bookbind/pubbooks/miller2_awl/chapter4/essay1/deluxe-content.html) (9 ed.). Addison Wesley Longman. ISBN 978-0-321-07607-6. Archived from the original on 2004-08-21.
10. Gros, L. (1872). *Théorie du Baguenodier*. Lyon: Aimé Vingtrinier.
11. Gedeon, T. D. (1996). "The Cyclic Towers of Hanoi: An Iterative Solution Produced by Transformation" (http://comjnl.oxfordjournals.org/content/39/4/353.short). *The Computer Journal*. **39** (4): 353–356. doi:10.1093/comjnl/39.4.353 (https://doi.org/10.1093%2Fcomjnl%2F39.4.353).
12. Bousch, T. (2014). "La quatrieme tour de Hanoi" (https://pdfs.semanticscholar.org/fb87/0a772baf96a2e11901122a2b04c3dd25596d.pdf) (PDF). *Bull. Belg. Math. Soc. Simon Stevin*. **21**: 895–912. doi:10.36045/bbms/1420071861 (https://doi.org/10.36045%2Fbbms%2F1420071861). S2CID 14243013 (https://api.semanticscholar.org/CorpusID:14243013).
13. Stewart, B. M.; Frame, J. S. (March 1941). "Solution to advanced problem 3819". *American Mathematical Monthly*. **48** (3): 216–9. doi:10.2307/2304268 (https://doi.org/10.2307%2F2304268). JSTOR 2304268 (https://www.jstor.org/stable/2304268).
14. Klavzar, Sandi; Milutinovi, Uro; Petrb, Ciril (2002). "Variations on the Four-Post Tower of Hanoi Puzzle" (https://core.ac.uk/download/pdf/81954097.pdf) (Postscript). *Congressus Numerantium*. **102**.
15. Stockmeyer, Paul (1994). "Variations on the Four-Post Tower of Hanoi Puzzle" (http://www.cs.wm.edu/~pkstoc/boca.ps) (Postscript). *Congressus Numerantium*. **102**: 3–12.
16. "University of Toronto CSC148 Slog" (https://berkeycolortran.wordpress.com/2014/04/05/week-12-update/). April 5, 2014. Retrieved July 22, 2015.
17. "UPenn CIS 194 Introduction to Haskell Assignment 1" (http://www.seas.upenn.edu/~cis194/spring13/hw/01-intro.pdf) (PDF). Retrieved January 31, 2016.
18. Hinz, A. (1989). "The Tower of Hanoi". *L'Enseignement Mathématique*. **35**: 289–321. doi:10.5169/seals-57378 (https://doi.org/10.5169%2Fseals-57378).
19. Chan, T. (1988). "A statistical analysis of the towers of Hanoi problem". *Internat. J. Comput. Math*. **28** (1–4): 57–65. doi:10.1080/00207168908803728 (https://doi.org/10.1080%2F00207168908803728).
20. Stewart, Ian (2004). *Another Fine Math You've Got Me Into...* Courier Dover. ISBN 978-0-7167-2342-4.

21. Romik, D. (2006). "Shortest paths in the Tower of Hanoi graph and finite automata". *SIAM Journal on Discrete Mathematics*. **20** (3): 610–622. arXiv:math/0310109 (https://arxiv.org/abs/math/0310109). doi:10.1137/050628660 (https://doi.org/10.1137%2F050628660). S2CID 8342396 (https://api.semanticscholar.org/CorpusID:8342396).
22. Prasad Vithal Chaugule (2015). "A Recursive Solution to Bicolor Towers of Hanoi Problem" (http://rmm.ludus-opuscula.org/PDF_Files/Chaugule_BicolorHanoi_37_48(4_2015)_low.pdf) (PDF). *Recreational Mathematics Magazine* (4): 37–48. ISSN 2182-1976 (https://www.worldcat.org/issn/2182-1976).
23. Yin, Xi; Liu, Xinhong; Pan, Yung-Tin; Walsh, Kathleen A.; Yang, Hong (November 4, 2014). "Hanoi Tower-like Multilayered Ultrathin Palladium Nanosheets". *Nano Letters*. **14** (12): 7188–94. Bibcode:2014NanoL..14.7188Y (https://ui.adsabs.harvard.edu/abs/2014NanoL..14.7188Y). doi:10.1021/nl503879a (https://doi.org/10.1021%2Fnl503879a). PMID 25369350 (https://pubmed.ncbi.nlm.nih.gov/25369350).
24. Zhang, J (1994). "Representations in distributed cognitive tasks" (http://wexler.free.fr/library/files/zhang%20%281994%29%20representations%20in%20distributed%20cognitive%20tasks.pdf) (PDF). *Cognitive Science*. **18**: 87–122. doi:10.1016/0364-0213(94)90021-3 (https://doi.org/10.1016%2F0364-0213%2894%2990021-3).
25. Zhang, Jiajie; Walji, Muhammad F. (2011). "TURF: Toward a unified framework of EHR usability". *Journal of Biomedical Informatics*. **44** (6): 1056–67. doi:10.1016/j.jbi.2011.08.005 (https://doi.org/10.1016%2Fj.jbi.2011.08.005). PMID 21867774 (https://pubmed.ncbi.nlm.nih.gov/21867774).
26. Beers, S. R.; Rosenberg, D. R.; Dick, E. L.; Williams, T.; O'Hearn, K. M.; Birmaher, B.; Ryan, C. M. (1999). "Neuropsychological study of frontal lobe function in psychotropic-naive children with obsessive-compulsive disorder" (https://ajp.psychiatryonline.org/doi/pdf/10.1176/ajp.156.5.777). *The American Journal of Psychiatry*. **156** (5): 777–9. doi:10.1176/ajp.156.5.777 (https://doi.org/10.1176%2Fajp.156.5.777) (inactive 2020-09-10). PMID 10327915 (https://pubmed.ncbi.nlm.nih.gov/10327915).
27. Reid, C.R.; Sumpter, D.J.; Beekman, M. (January 2011). "Optimisation in a natural system: Argentine ants solve the Towers of Hanoi". *J. Exp. Biol*. **214** (Pt 1): 50–8. CiteSeerX 10.1.1.231.9201 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.231.9201). doi:10.1242/jeb.048173 (https://doi.org/10.1242%2Fjeb.048173). PMID 21147968 (https://pubmed.ncbi.nlm.nih.gov/21147968). S2CID 18819977 (https://api.semanticscholar.org/CorpusID:18819977).
28. Russell, Eric Frank (April 1959). "Now Inhale". *Astounding Science Fiction*.
29. "Tower of Hanoi (video game concept)" (http://www.giantbomb.com/tower-of-hanoi/92-5744/). Giantbomb.com. Retrieved 2010-12-05.
30. "Tower of Hanoi / Andamiro" (https://web.archive.org/web/20120301060428/http://www.segaarcade.com/towerofhanoi). Sega Amusements. Archived from the original (http://www.segaarcade.com/towerofhanoi) on 2012-03-01. Retrieved 2012-02-26.

External links

- Weisstein, Eric W. "Tower of Hanoi" (https://mathworld.wolfram.com/TowerofHanoi.html). *MathWorld*.